United States of America

To Promote the Progress of Science and Useful Arts

The Director

of the United States Patent and Trademark Office has received an application for a patent for a new and useful invention. The title and description of the invention are enclosed. The requirements of law have been complied with, and it has been determined that a patent on the invention shall be granted under the law.

Therefore, this United States

Patent

grants to the person(s) having title to this patent the right to exclude others from making, using, offering for sale, or selling the invention throughout the United States of America or importing the invention into the United States of America, and if the invention is a process, of the right to exclude others from using, offering for sale or selling throughout the United States of America, products made by that process, for the term set forth in 35 U.S.C. 154(a)(2) or (c)(1), subject to the payment of maintenance fees as provided by 35 U.S.C. 41(b). See the Maintenance Fee Notice on the inside of the cover.

David J. Kappos

Acting Director of the United States Patent and Trademark Office

Maintenance Fee Notice

If the application for this patent was filed on or after December 12, 1980, maintenance fees are due three years and six months, seven years and six months, and eleven years and six months after the date of this grant, or within a grace period of six months thereafter upon payment of a surcharge as provided by law. The amount, number and timing of the maintenance fees required may be changed by law or regulation. Unless payment of the applicable maintenance fee is received in the United States Patent and Trademark Office on or before the date the fee is due or within a grace period of six months thereafter, the patent will expire as of the end of such grace period.

Patent Term Notice

If the application for this patent was filed on or after June 8, 1995, the term of this patent begins on the date on which this patent issues and ends twenty years from the filing date of the application or, if the application contains a specific reference to an earlier filed application or applications under 35 U.S.C. 120, 121, 365(c), or 386(c), twenty years from the filing date of the earliest such application ("the twenty-year term"), subject to the payment of maintenance fees as provided by 35 U.S.C. 41(b), and any extension as provided by 35 U.S.C. 154(b) or 156 or any disclaimer under 35 U.S.C. 253.

If this application was filed prior to June 8, 1995, the term of this patent begins on the date on which this patent issues and ends on the later of seventeen years from the date of the grant of this patent or the twenty-year term set forth above for patents resulting from applications filed on or after June 8, 1995, subject to the payment of maintenance fees as provided by 35 U.S.C. 41(b) and any extension as provided by 35 U.S.C. 156 or any disclaimer under 35 U.S.C. 253.

US012182662B2

(12) **United States Patent**
Selly

(10) **Patent No.: US 12,182,662 B2**
(45) **Date of Patent: *Dec. 31, 2024**

(54) **PROGRAMMABLE QUANTUM COMPUTER**

(71) Applicant: **PANVIA FUTURE TECHNOLOGIES, INC.**, Palo Alto, CA (US)

(72) Inventor: **Roger Selly**, Palo Alto, CA (US)

(73) Assignee: **Panvia Future Technologies Inc.**, San Jose, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 58 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **18/157,562**

(22) Filed: **Jan. 20, 2023**

(65) **Prior Publication Data**

US 2023/0229950 A1 Jul. 20, 2023

Related U.S. Application Data

(63) Continuation-in-part of application No. 16/594,685, filed on Oct. 7, 2019, now Pat. No. 11,561,951, which
(Continued)

(51) **Int. Cl.**
G06F 16/00 (2019.01)
G06F 16/21 (2019.01)
(Continued)

(52) **U.S. Cl.**
CPC ***G06N 10/20*** (2022.01); ***G06F 16/212*** (2019.01); ***G06F 16/2264*** (2019.01); ***G06F 16/24532*** (2019.01); ***G06F 16/258*** (2019.01)

(58) **Field of Classification Search**
CPC G06F 16/2264; G06F 16/212; G06F 16/24532; G06F 16/258
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,144,428 A 9/1992 Okuda
5,161,204 A 11/1992 Hutcheson et al.
(Continued)

FOREIGN PATENT DOCUMENTS

JP 2003141538 A 5/2003
JP 2005078407 A 3/2005

OTHER PUBLICATIONS

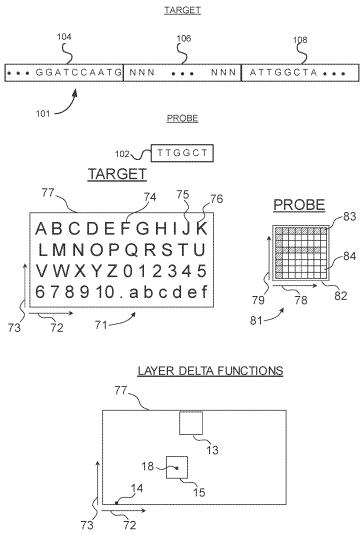
Ezhov, A.A., et al., "Quantum Associative Memory with Distributed Queries", Information Science, vol. 128, p. 271-293, 2000.
(Continued)

Primary Examiner — Alexander Khong

(57) **ABSTRACT**

A method for searching data includes storing a probe data and a target data expressed in a first orthogonal domain. The target data includes potential probe match data each characterized by the length of the target data. The probe data representation and the target data are transformed into an orthogonal domain. In the orthogonal domain, the target data is encoded with modulation functions to produce a plurality of encoded target data, each of the modulation functions having a position index corresponding to one of the potential probe match data. The plurality of encoded target data is interfered with the probe data in the orthogonal domain and an inverse transform result is obtained. If the inverse transform result exceeds a threshold, information is output indicating a match between the probe data and a corresponding one of the potential probe match data.

10 Claims, 39 Drawing Sheets



Related U.S. Application Data

is a continuation-in-part of application No. 14/480,355, filed on Sep. 8, 2014, now Pat. No. 10,438,690, which is a continuation-in-part of application No. 11/914,554, filed as application No. PCT/US2006/018699 on May 15, 2006, now Pat. No. 8,832,139.

(60) Provisional application No. 60/681,374, filed on May 16, 2005.

(51) **Int. Cl.**

G06F 16/22	(2019.01)
G06F 16/2453	(2019.01)
G06F 16/25	(2019.01)
G06N 10/20	(2022.01)

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,706,498	A *	1/1998	Fujimiya	G16B 50/30 435/6.12
6,054,268	A	4/2000	Perlin	
6,317,766	B1	11/2001	Grover	
6,321,182	B1	11/2001	Suzuki	
6,466,923	B1	10/2002	Young	
6,551,784	B2 *	4/2003	Fodor	G03F 7/00 435/6.12
6,578,018	B1	6/2003	Ulyanov	
6,718,309	B1	4/2004	Selly	
6,920,451	B2 *	7/2005	Shaw	G06F 16/358 707/999.102
2002/0152191	A1	10/2002	Hollenberg et al.	
2003/0228591	A1 *	12/2003	Scafe	G16B 25/20 435/6.1
2004/0193789	A1	9/2004	Rudolf	

OTHER PUBLICATIONS

Grover, Lov K., "A Fast Quantum Mechanical Algorithm for Database Search", [On-Line]; Nov. 19, 1996; XP002543984; Retrieved from the Internet: URL:http://arxiv.org/PS_cache/quant-ph/pdf/9605/9605043v3.pdf>: Retrieved on Sep. 2, 2009.

Trugenberger, C.A., "Quantum Pattern Recognition", Quantum Information Processing; vol. 1, No. 6, p. 471-493, Dec. 2002.

Ventura, Dan, et al. "Quantum Associative Memory", Information Sciences, vol. 124, p. 273-296, 2002.

Ventura et al., "Quantum Associative Memory with Exponential Capacity", Neural Networks Proceedings, 1988; IEEE World Congress on Computational Intelligene; The 1998 IEEE International Joint Conference on Anchorage, AK, USA May 4-9, 1998; New York; USA; IEEE; US vol. 1 May 4, 1998; p. 509-513; XP010286557; isbn: 978-0-7803-4859-2.

Von Ahsen, Nicholas, et al., "Application of a Thermodynamic Nearest-Neighbor Model to Estimate Nucleic Acid Staility and Optimize Probe Design: Prediction of Melting Points of Multiple Mutations of Apolipoprotein B-3500 and Factor V with a Hybridization Probe Genotyping Assay on the LightCycler", Clinical Chemistry, vol. 45, No. 12, p. 2094-2101, 1999.

* cited by examiner

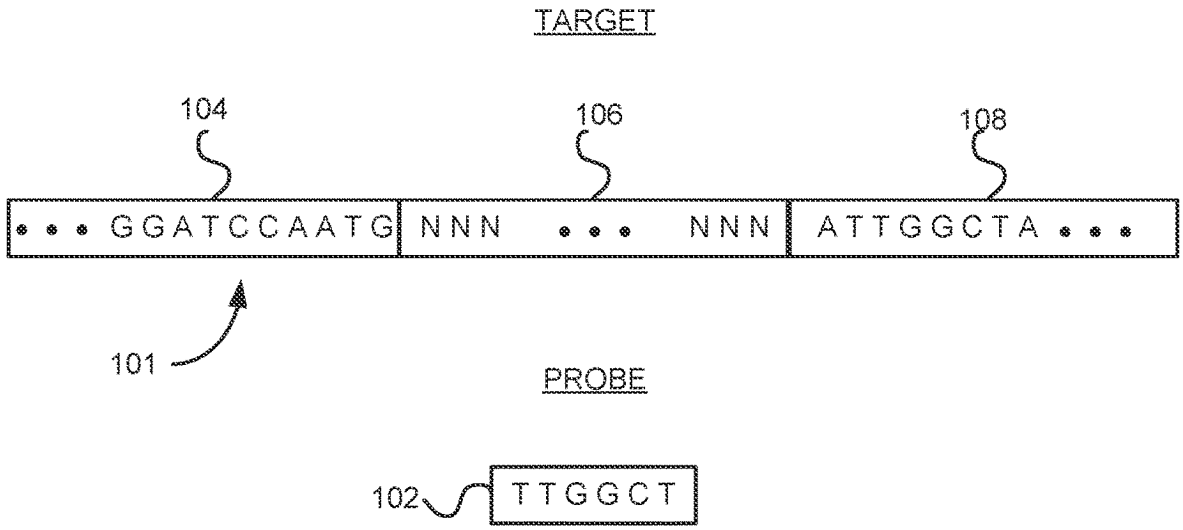

TARGET
104
106
108
• • • GGATCCAATG
NNN • • • NNN
ATTGGCTA • • •
101
PROBE
102
TTGGCT

Figure 1A

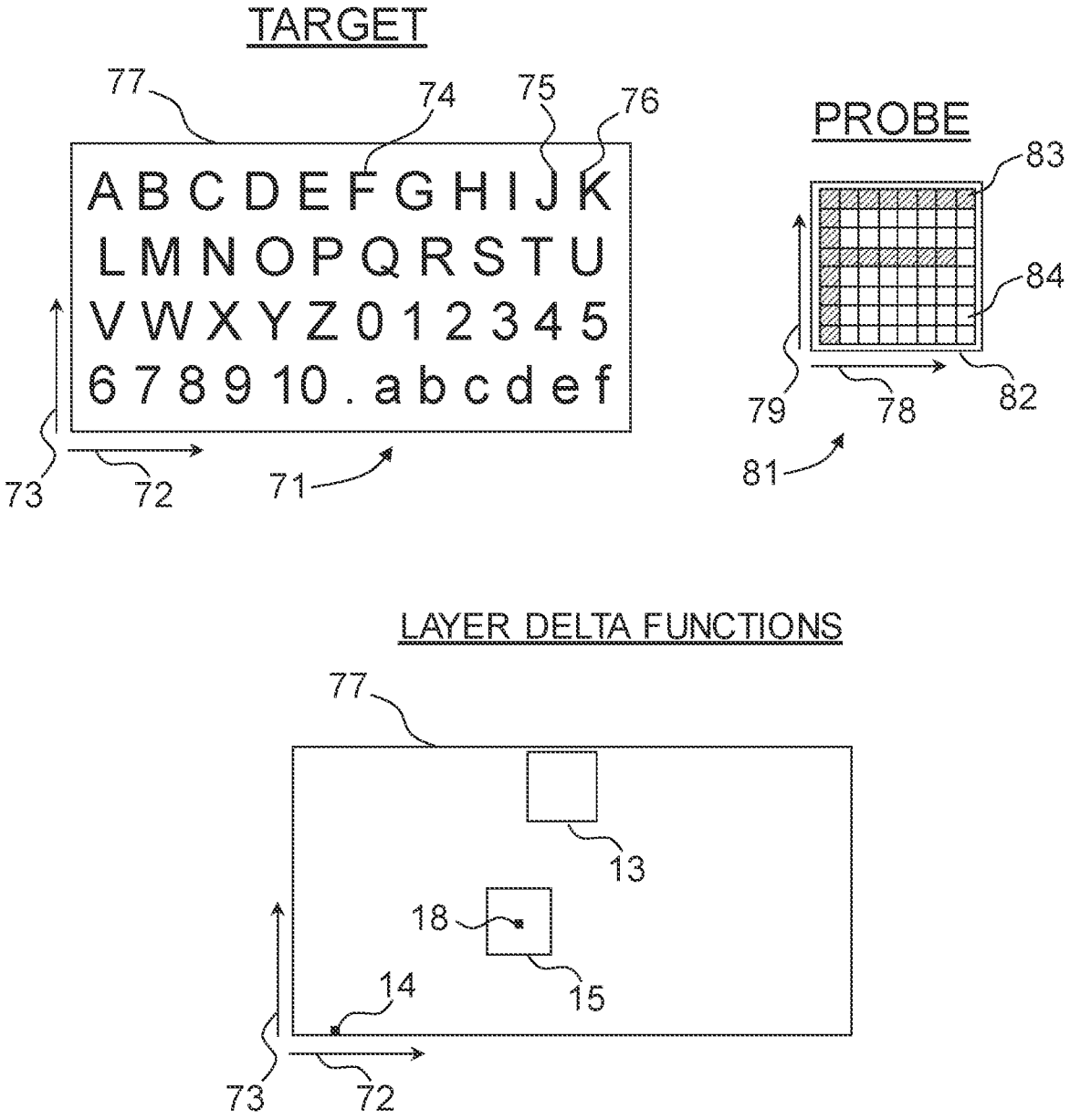

TARGET
77
74
75
76
A B C D E F G H I J K
L M N O P Q R S T U
V W X Y Z 0 1 2 3 4 5
6 7 8 9 10 . a b c d e f
73
72
71
PROBE
83
84
79
78
82
81
LAYER DELTA FUNCTIONS
77
13
18
15
14
73
72

Figure 1B

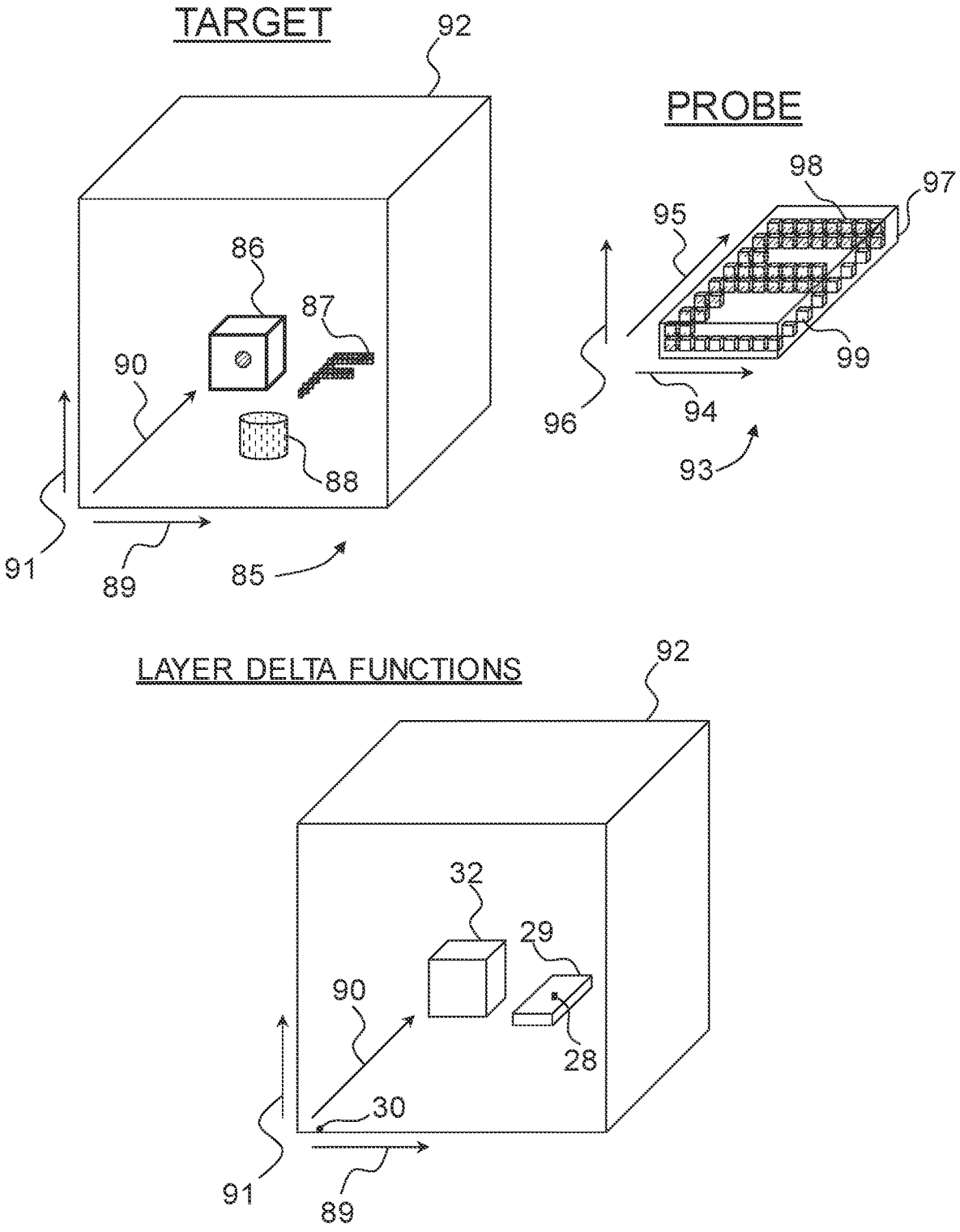

TARGET
92
86
87
90
88
91
89
85
PROBE
98
97
95
99
96
94
93
LAYER DELTA FUNCTIONS
92
32
29
90
28
30
91
89

Figure 1C

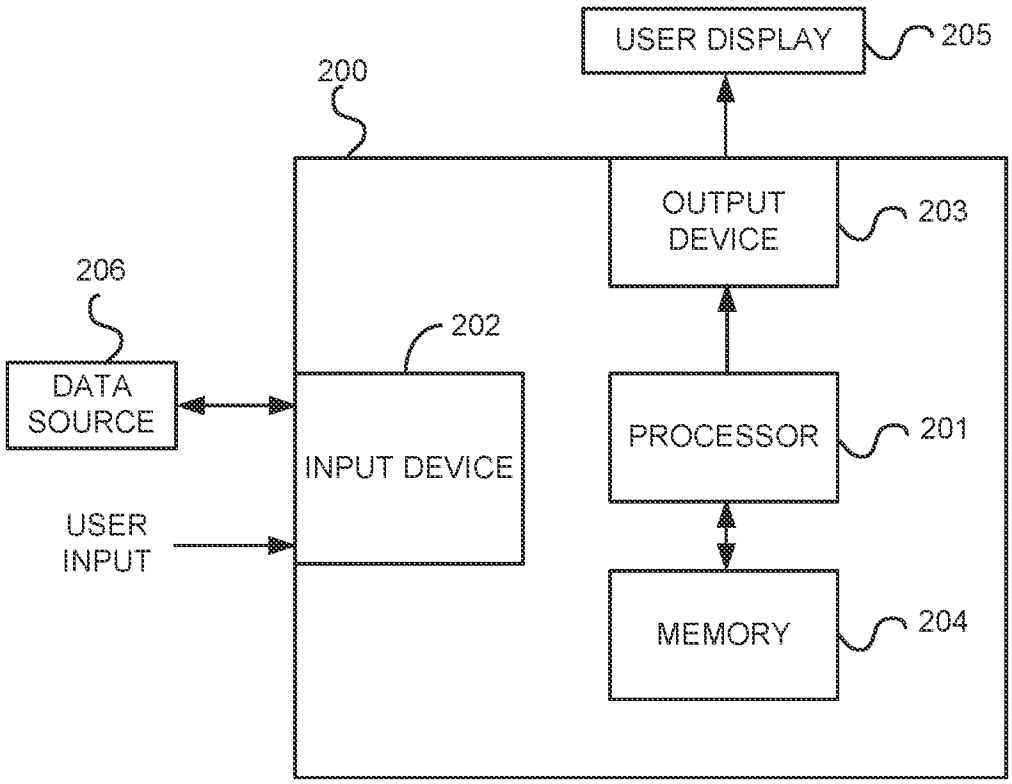

USER DISPLAY
205
200
OUTPUT DEVICE
203
206
202
DATA SOURCE
INPUT DEVICE
PROCESSOR
201
USER INPUT
MEMORY
204

Figure 2

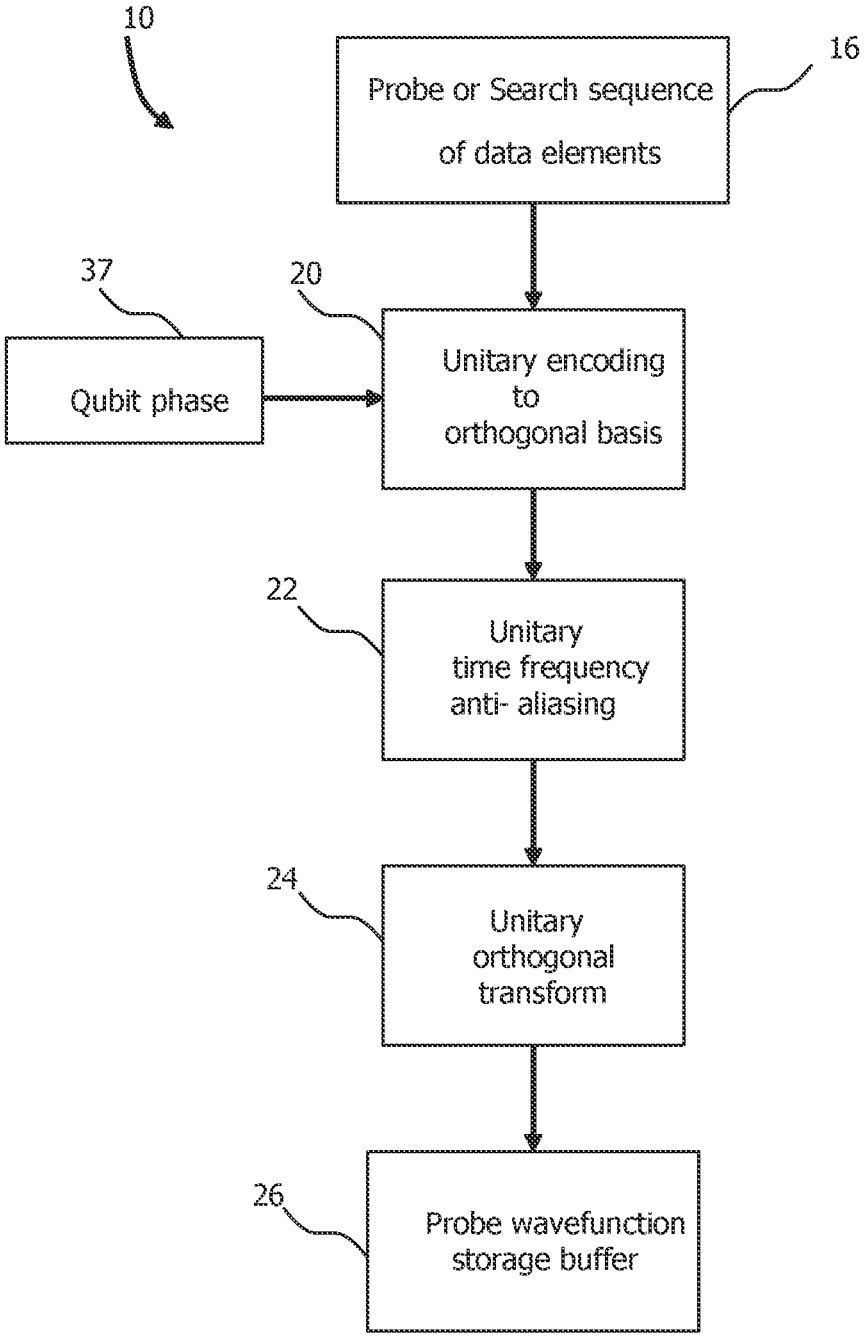

10
Probe or Search sequence of data elements
16
37
20
Qubit phase
Unitary encoding to orthogonal basis
22
Unitary time frequency anti- aliasing
24
Unitary orthogonal transform
26
Probe wavefunction storage buffer

Figure 3A

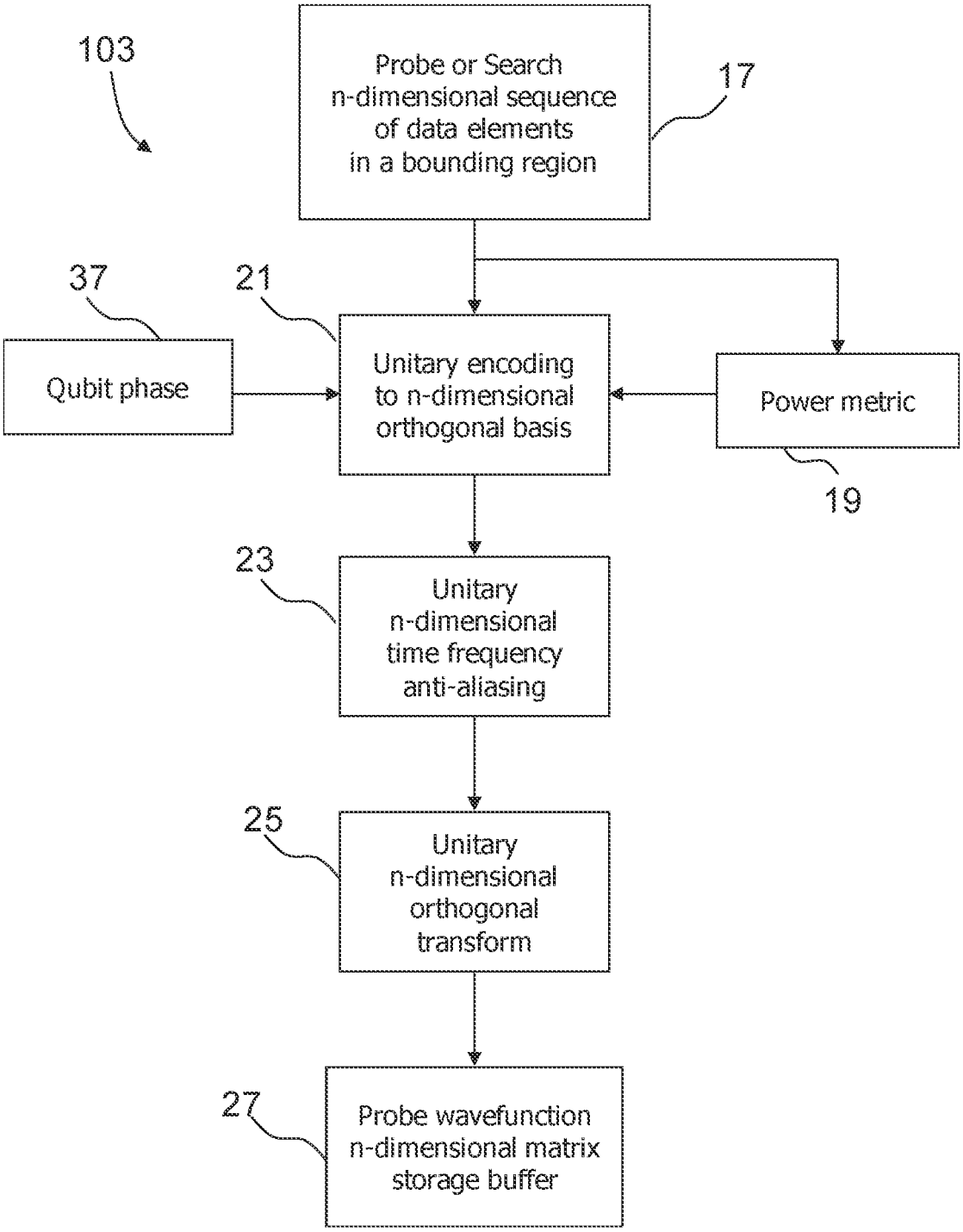

103
Probe or Search n-dimensional sequence of data elements in a bounding region
17
37
21
Qubit phase
Unitary encoding to n-dimensional orthogonal basis
Power metric
19
23
Unitary n-dimensional time frequency anti-aliasing
25
Unitary n-dimensional orthogonal transform
27
Probe wavefunction n-dimensional matrix storage buffer

Figure 3B

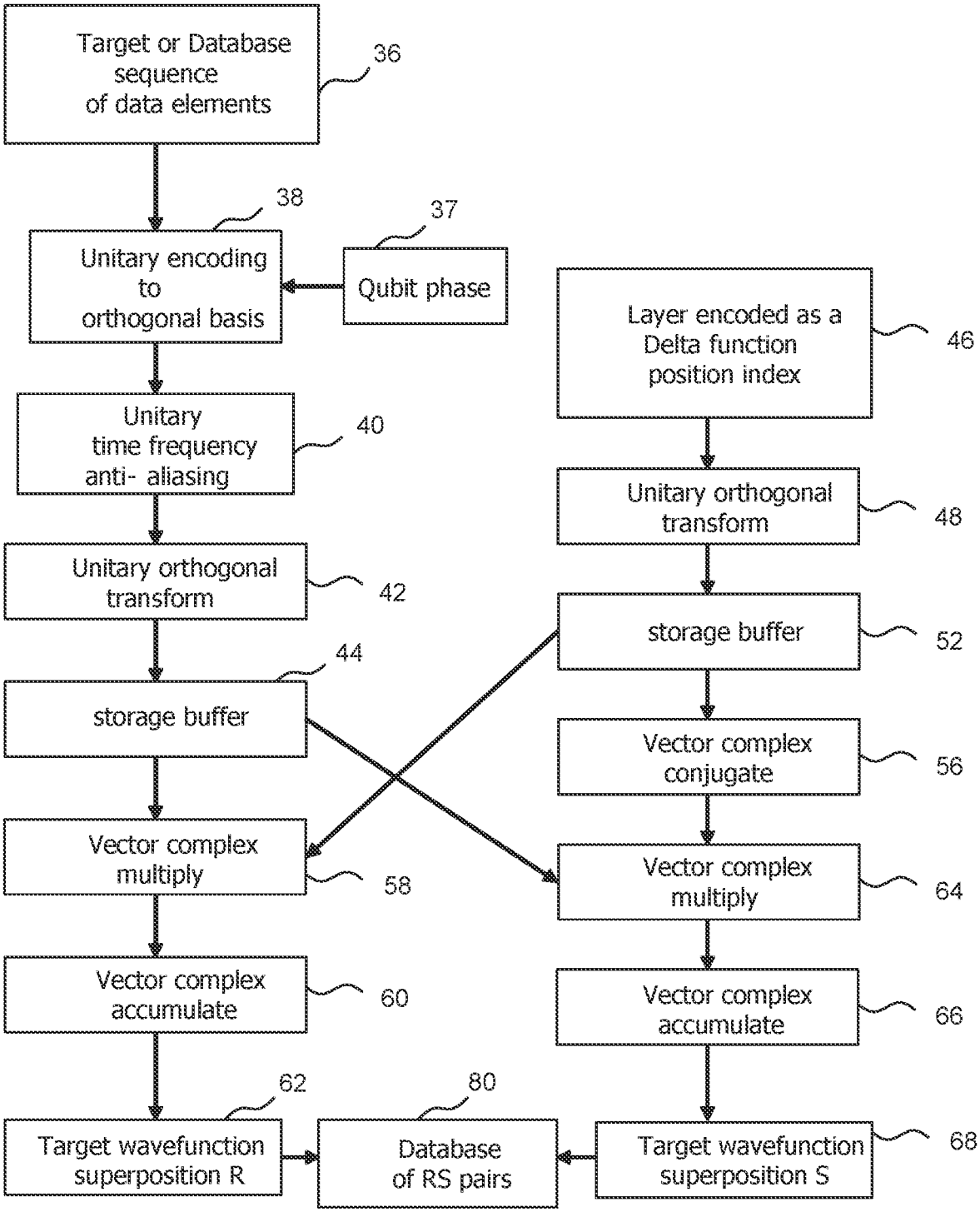

Target or Database sequence of data elements
36
38
37
Unitary encoding to orthogonal basis
Qubit phase
Unitary time frequency anti- aliasing
40
Unitary orthogonal transform
42
44
storage buffer
Vector complex multiply
58
Vector complex accumulate
60
62
Target wavefunction superposition R
Layer encoded as a Delta function position index
46
Unitary orthogonal transform
48
storage buffer
52
Vector complex conjugate
56
Vector complex multiply
64
Vector complex accumulate
66
80
Database of RS pairs
Target wavefunction superposition S
68

Figure 4A

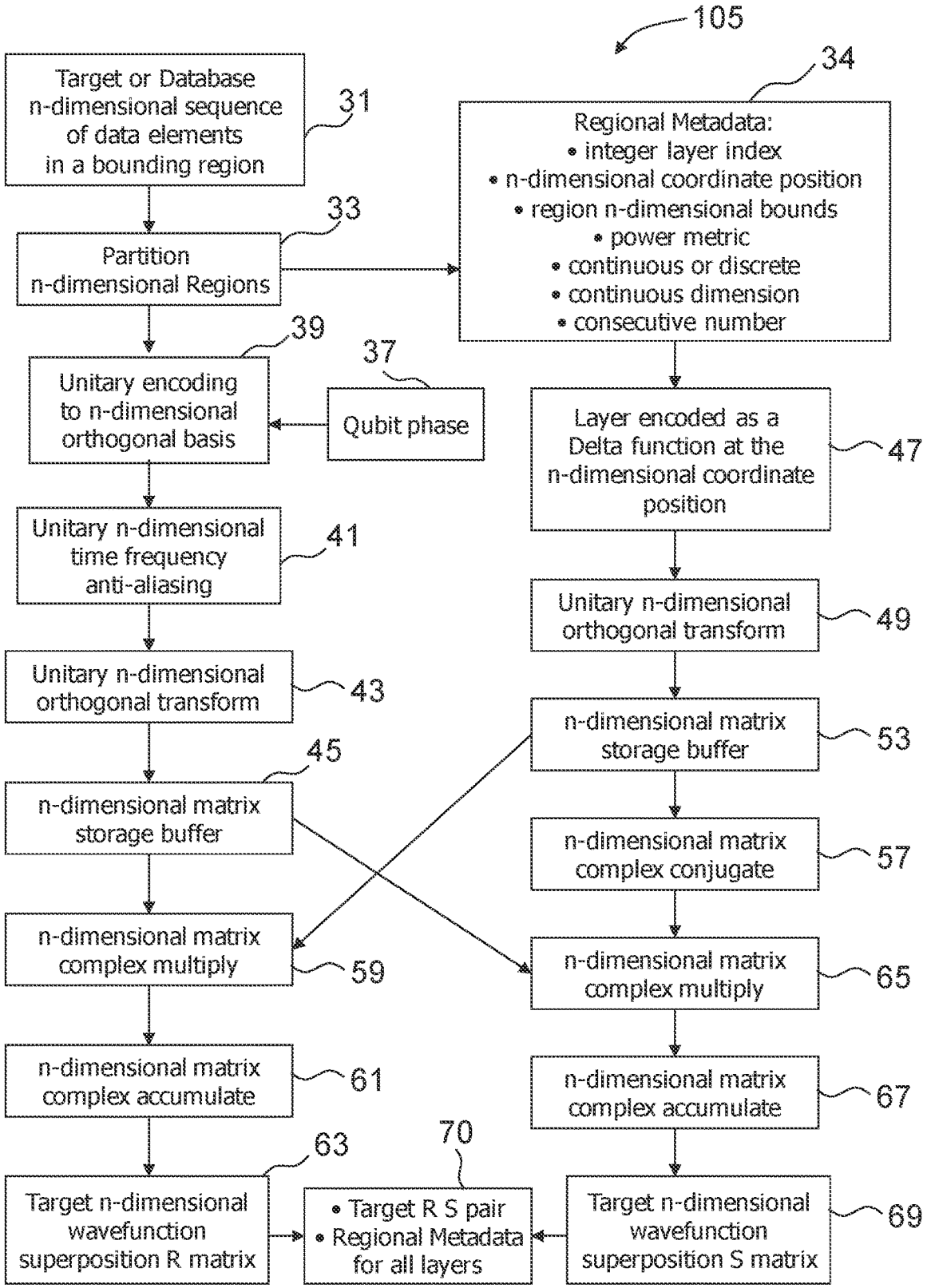

105
Target or Database n-dimensional sequence of data elements in a bounding region
31
34
Regional Metadata:
• integer layer index
• n-dimensional coordinate position
• region n-dimensional bounds
• power metric
• continuous or discrete
• continuous dimension
• consecutive number
33
Partition n-dimensional Regions
39
Unitary encoding to n-dimensional orthogonal basis
37
Qubit phase
Layer encoded as a Delta function at the n-dimensional coordinate position
47
Unitary n-dimensional time frequency anti-aliasing
41
Unitary n-dimensional orthogonal transform
49
Unitary n-dimensional orthogonal transform
43
n-dimensional matrix storage buffer
53
45
n-dimensional matrix storage buffer
n-dimensional matrix complex conjugate
57
n-dimensional matrix complex multiply
59
n-dimensional matrix complex multiply
65
n-dimensional matrix complex accumulate
61
n-dimensional matrix complex accumulate
67
63
70
Target n-dimensional wavefunction superposition R matrix
• Target R S pair
• Regional Metadata for all layers
Target n-dimensional wavefunction superposition S matrix
69

Figure 4B

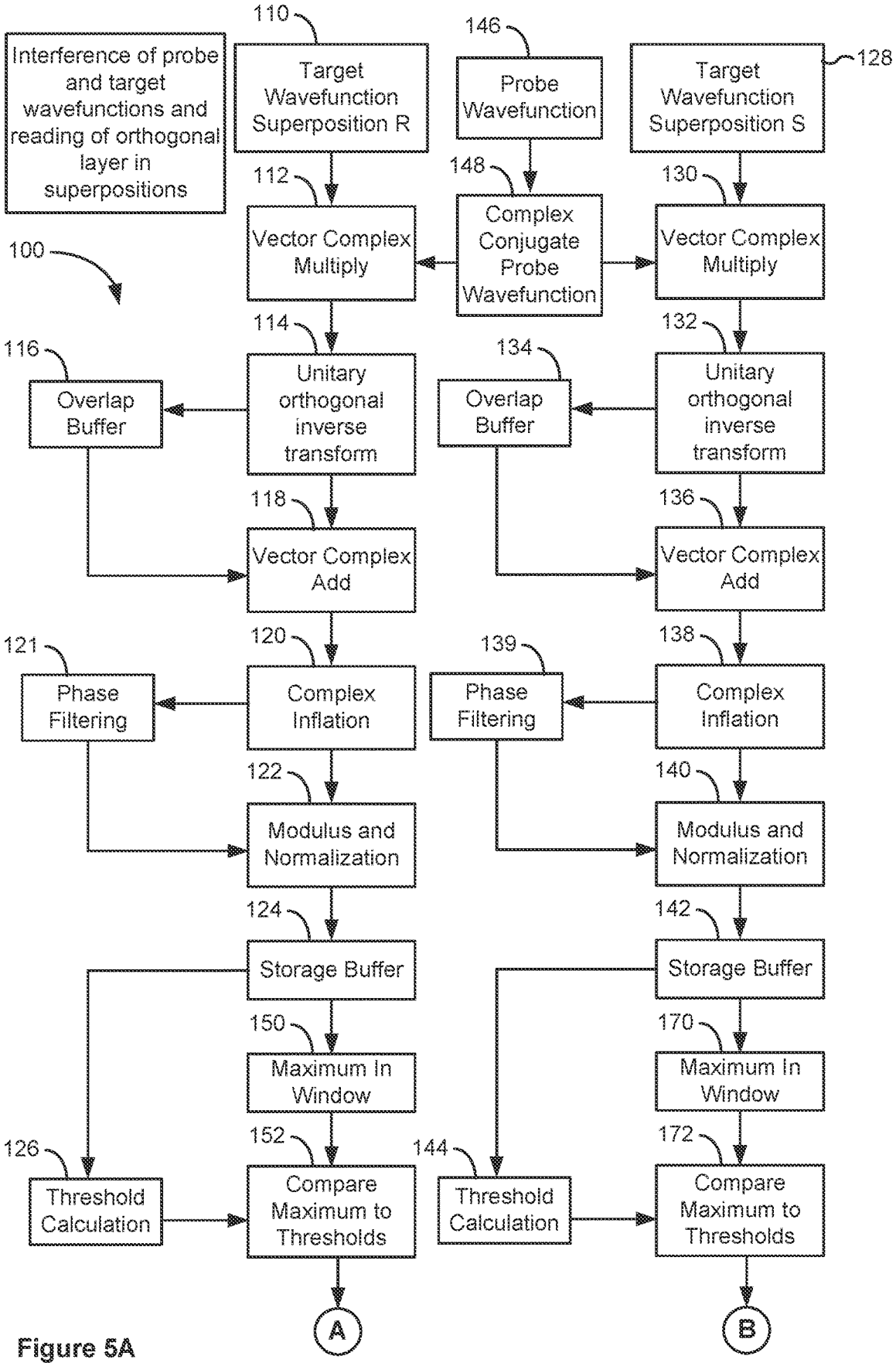

Interference of probe and target wavefunctions and reading of orthogonal layer in superpositions
100
110
Target Wavefunction Superposition R
112
Vector Complex Multiply
114
Unitary orthogonal inverse transform
116
Overlap Buffer
118
Vector Complex Add
120
Complex Inflation
121
Phase Filtering
122
Modulus and Normalization
124
Storage Buffer
150
Maximum In Window
152
Compare Maximum to Thresholds
126
Threshold Calculation
A
146
Probe Wavefunction
148
Complex Conjugate Probe Wavefunction
128
Target Wavefunction Superposition S
130
Vector Complex Multiply
132
Unitary orthogonal inverse transform
134
Overlap Buffer
136
Vector Complex Add
138
Complex Inflation
139
Phase Filtering
140
Modulus and Normalization
142
Storage Buffer
170
Maximum In Window
172
Compare Maximum to Thresholds
144
Threshold Calculation
B

Figure 5A

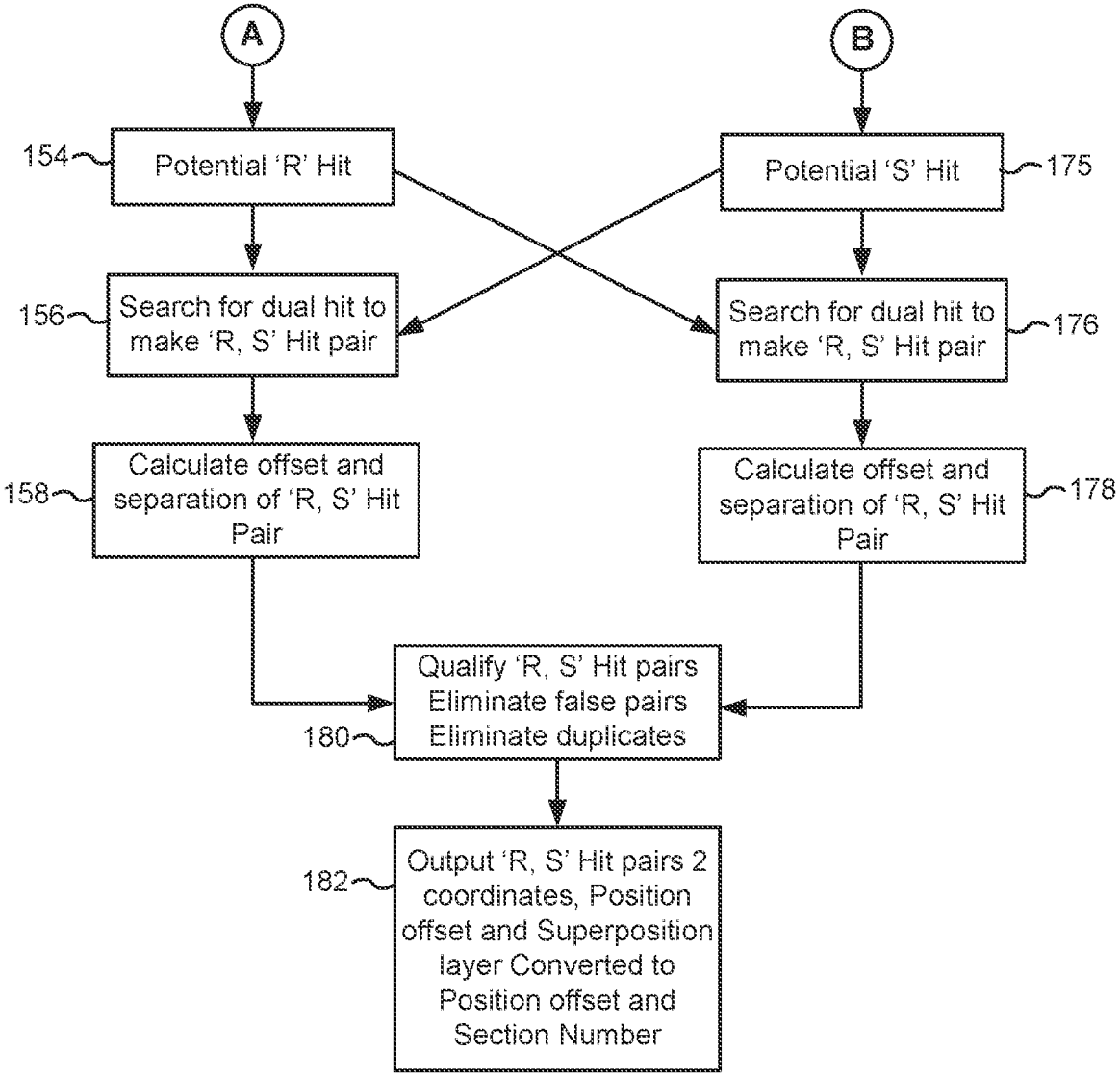

A
B
154
Potential 'R' Hit
Potential 'S' Hit
175
156
Search for dual hit to make 'R, S' Hit pair
Search for dual hit to make 'R, S' Hit pair
176
158
Calculate offset and separation of 'R, S' Hit Pair
Calculate offset and separation of 'R, S' Hit Pair
178
Qualify 'R, S' Hit pairs
Eliminate false pairs
Eliminate duplicates
180
182
Output 'R, S' Hit pairs 2 coordinates, Position offset and Superposition layer Converted to Position offset and Section Number

Figure 5B

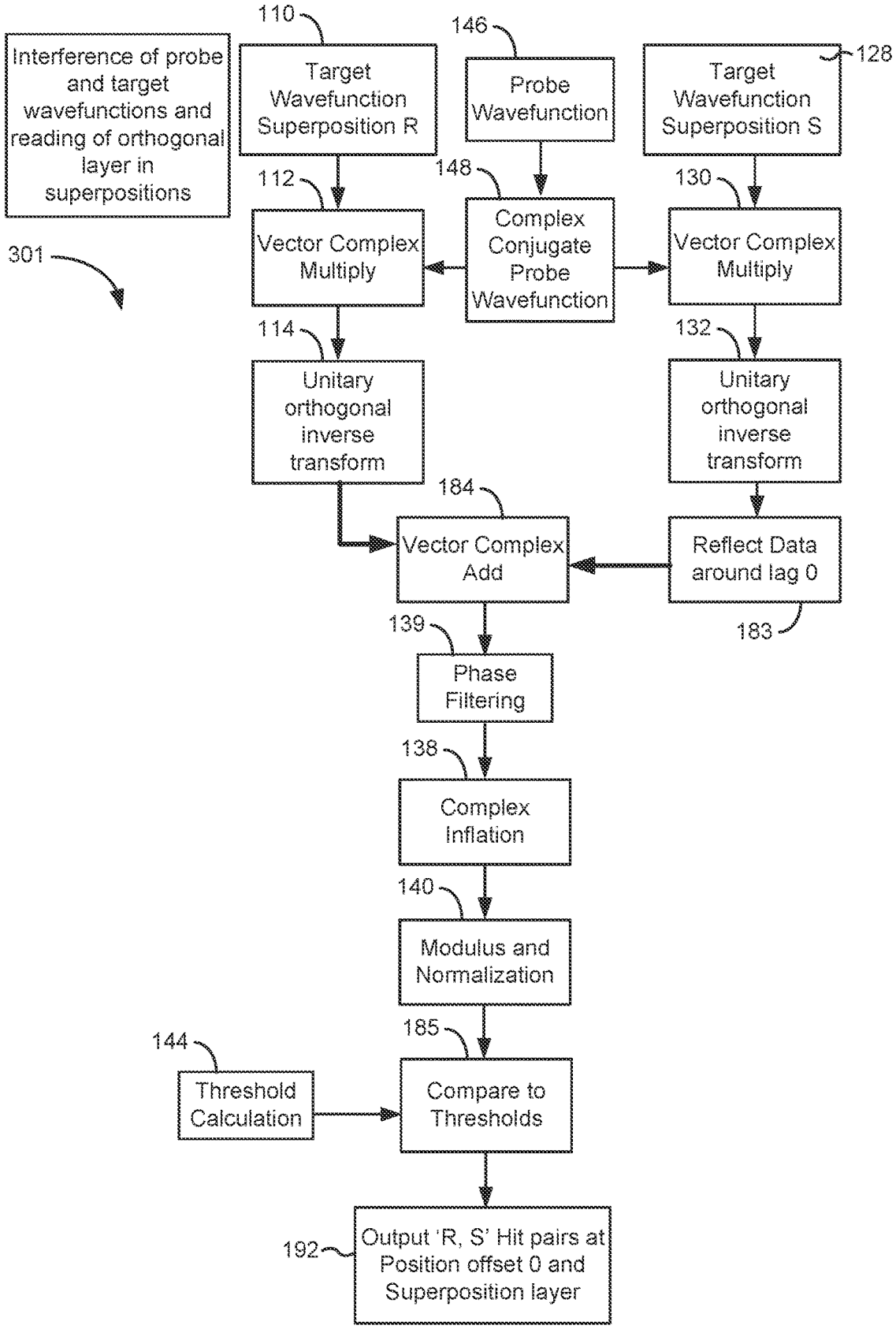

Interference of probe and target wavefunctions and reading of orthogonal layer in superpositions
301
110
Target Wavefunction Superposition R
146
Probe Wavefunction
128
Target Wavefunction Superposition S
112
148
130
Vector Complex Multiply
Complex Conjugate Probe Wavefunction
Vector Complex Multiply
114
132
Unitary orthogonal inverse transform
Unitary orthogonal inverse transform
184
Vector Complex Add
Reflect Data around lag 0
183
139
Phase Filtering
138
Complex Inflation
140
Modulus and Normalization
185
144
Threshold Calculation
Compare to Thresholds
192
Output 'R, S' Hit pairs at Position offset 0 and Superposition layer

Figure 5C

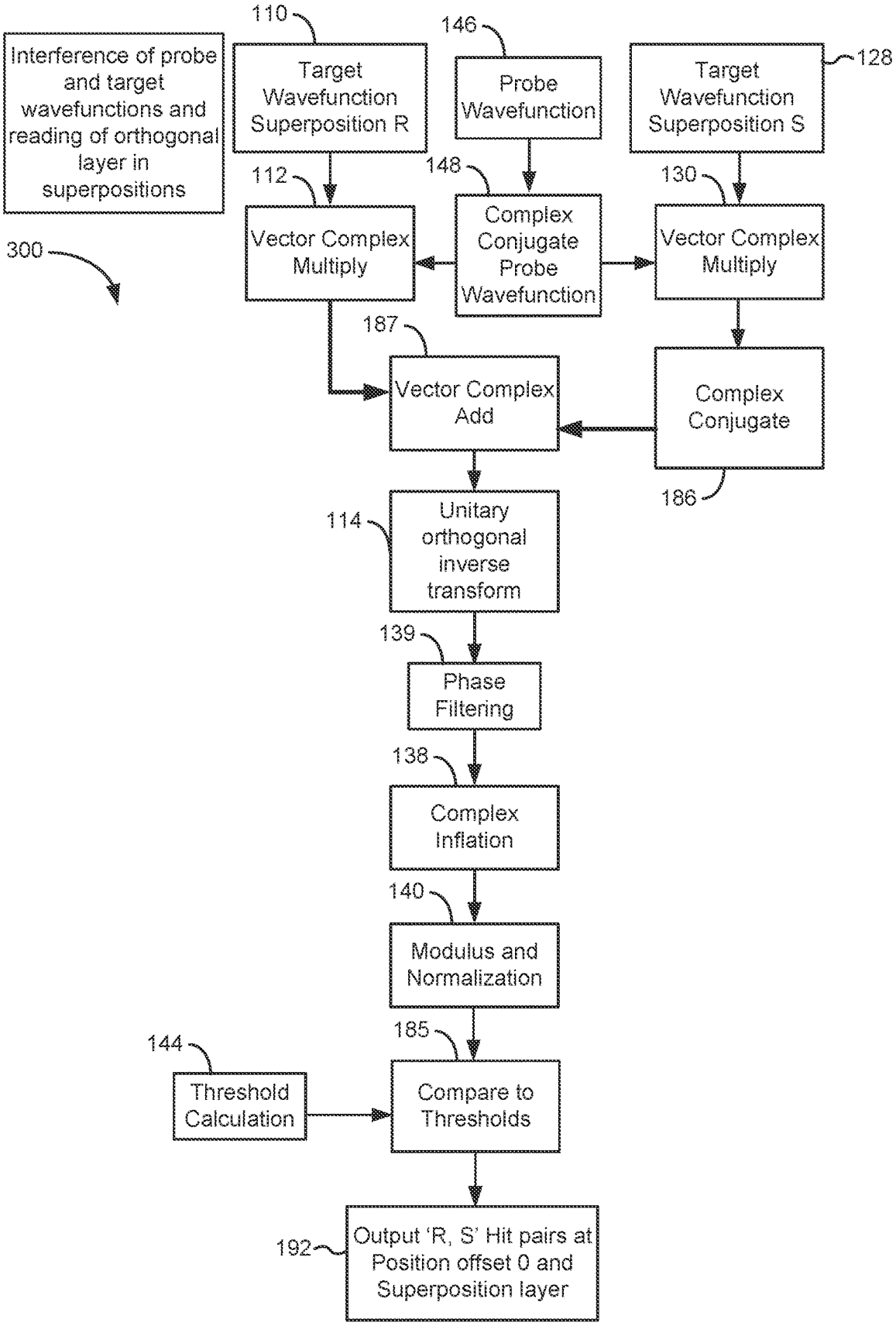

Interference of probe and target wavefunctions and reading of orthogonal layer in superpositions
300
110
Target Wavefunction Superposition R
146
Probe Wavefunction
128
Target Wavefunction Superposition S
112
148
130
Vector Complex Multiply
Complex Conjugate Probe Wavefunction
Vector Complex Multiply
187
Vector Complex Add
Complex Conjugate
186
114
Unitary orthogonal inverse transform
139
Phase Filtering
138
Complex Inflation
140
Modulus and Normalization
185
144
Threshold Calculation
Compare to Thresholds
192
Output 'R, S' Hit pairs at Position offset 0 and Superposition layer

Figure 5D

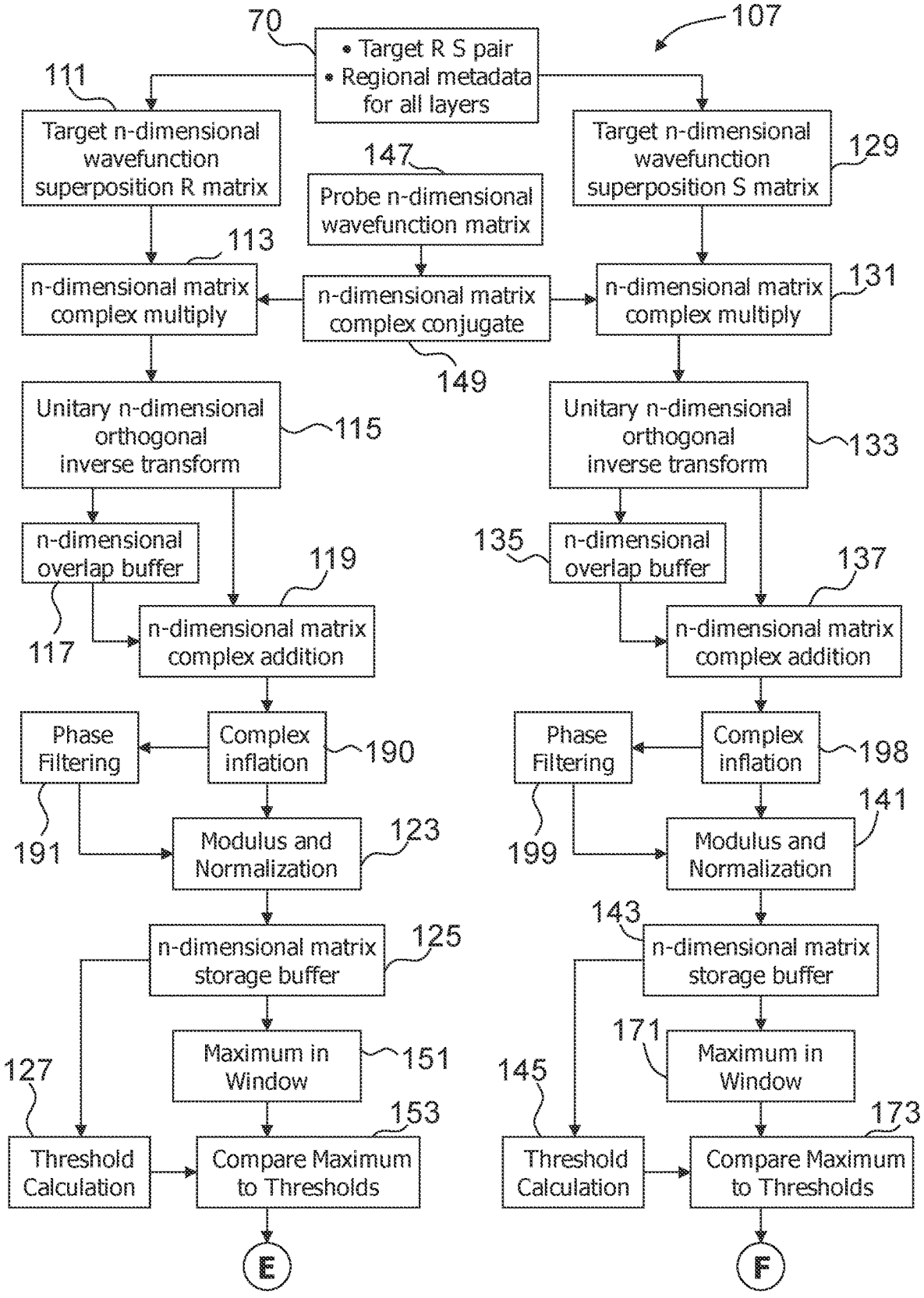

70
• Target R S pair
• Regional metadata for all layers
107
111
Target n-dimensional wavefunction superposition R matrix
147
Probe n-dimensional wavefunction matrix
Target n-dimensional wavefunction superposition S matrix
129
113
n-dimensional matrix complex multiply
n-dimensional matrix complex conjugate
n-dimensional matrix complex multiply
131
149
Unitary n-dimensional orthogonal inverse transform
115
Unitary n-dimensional orthogonal inverse transform
133
n-dimensional overlap buffer
135
n-dimensional overlap buffer
119
137
117
n-dimensional matrix complex addition
n-dimensional matrix complex addition
Phase Filtering
Complex inflation
190
Phase Filtering
Complex inflation
198
191
Modulus and Normalization
123
199
Modulus and Normalization
141
143
n-dimensional matrix storage buffer
125
n-dimensional matrix storage buffer
171
Maximum in Window
151
Maximum in Window
127
145
153
173
Threshold Calculation
Compare Maximum to Thresholds
Threshold Calculation
Compare Maximum to Thresholds
E
F

Figure 5E

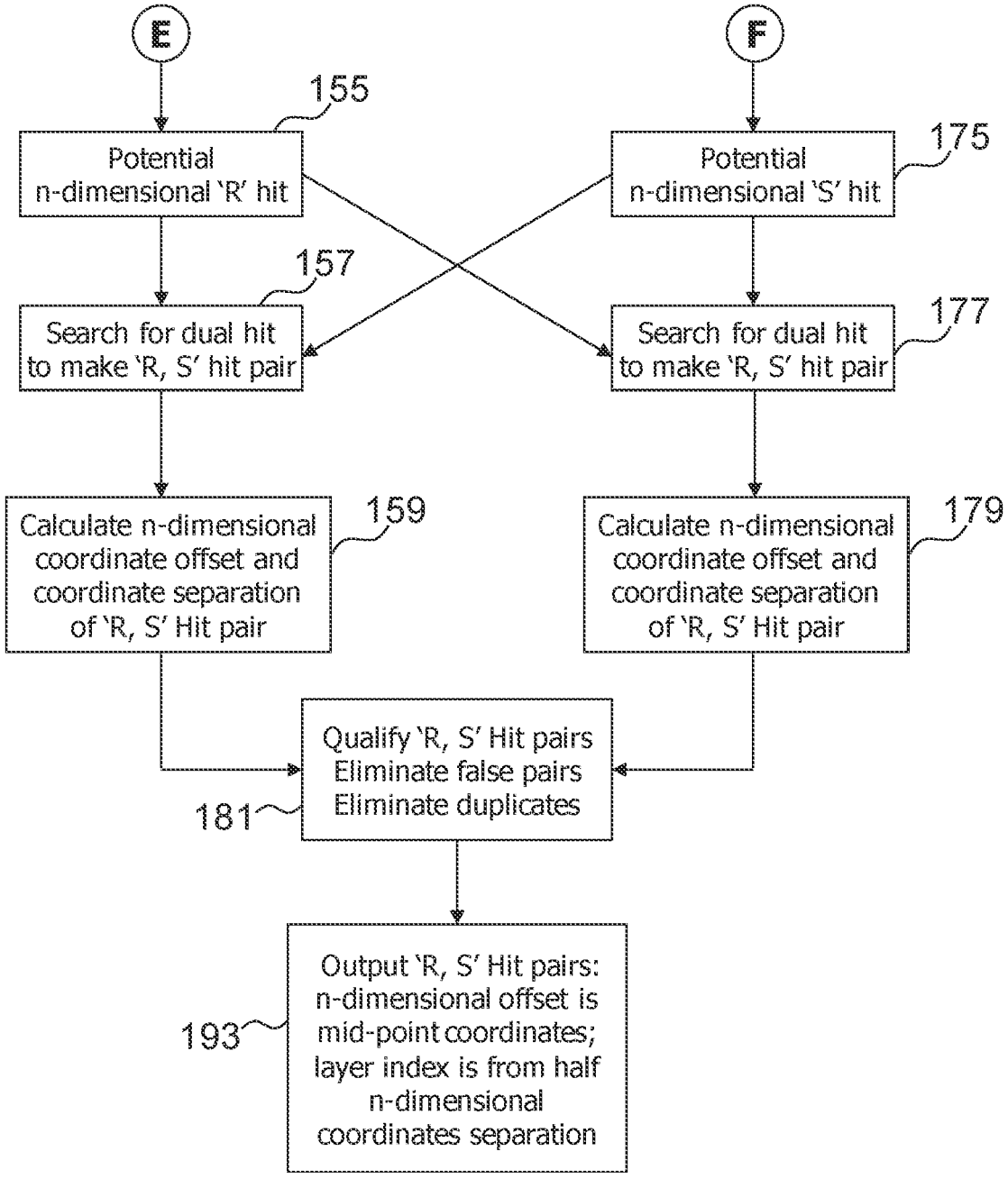

E
F
155
Potential n-dimensional 'R' hit
175
Potential n-dimensional 'S' hit
157
Search for dual hit to make 'R, S' hit pair
177
Search for dual hit to make 'R, S' hit pair
159
Calculate n-dimensional coordinate offset and coordinate separation of 'R, S' Hit pair
179
Calculate n-dimensional coordinate offset and coordinate separation of 'R, S' Hit pair
Qualify 'R, S' Hit pairs
Eliminate false pairs
Eliminate duplicates
181
Output 'R, S' Hit pairs: n-dimensional offset is mid-point coordinates; layer index is from half n-dimensional coordinates separation
193

Figure 5F

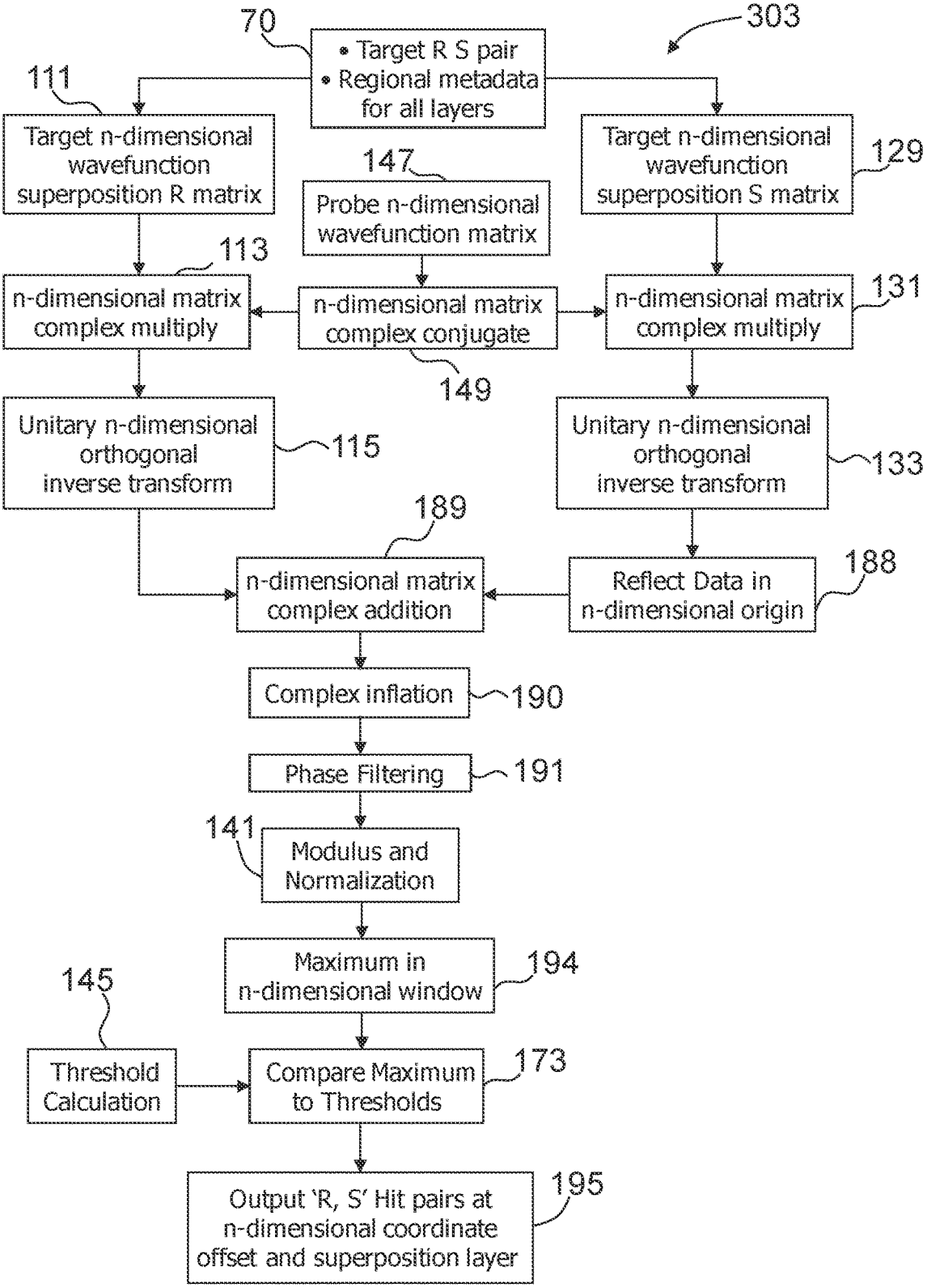

70
303
• Target R S pair
• Regional metadata for all layers
111
Target n-dimensional wavefunction superposition R matrix
147
Probe n-dimensional wavefunction matrix
Target n-dimensional wavefunction superposition S matrix
129
113
n-dimensional matrix complex multiply
n-dimensional matrix complex conjugate
n-dimensional matrix complex multiply
131
149
Unitary n-dimensional orthogonal inverse transform
115
Unitary n-dimensional orthogonal inverse transform
133
189
n-dimensional matrix complex addition
Reflect Data in n-dimensional origin
188
Complex inflation
190
Phase Filtering
191
141
Modulus and Normalization
Maximum in n-dimensional window
194
145
Threshold Calculation
Compare Maximum to Thresholds
173
Output 'R, S' Hit pairs at n-dimensional coordinate offset and superposition layer
195

Figure 5G

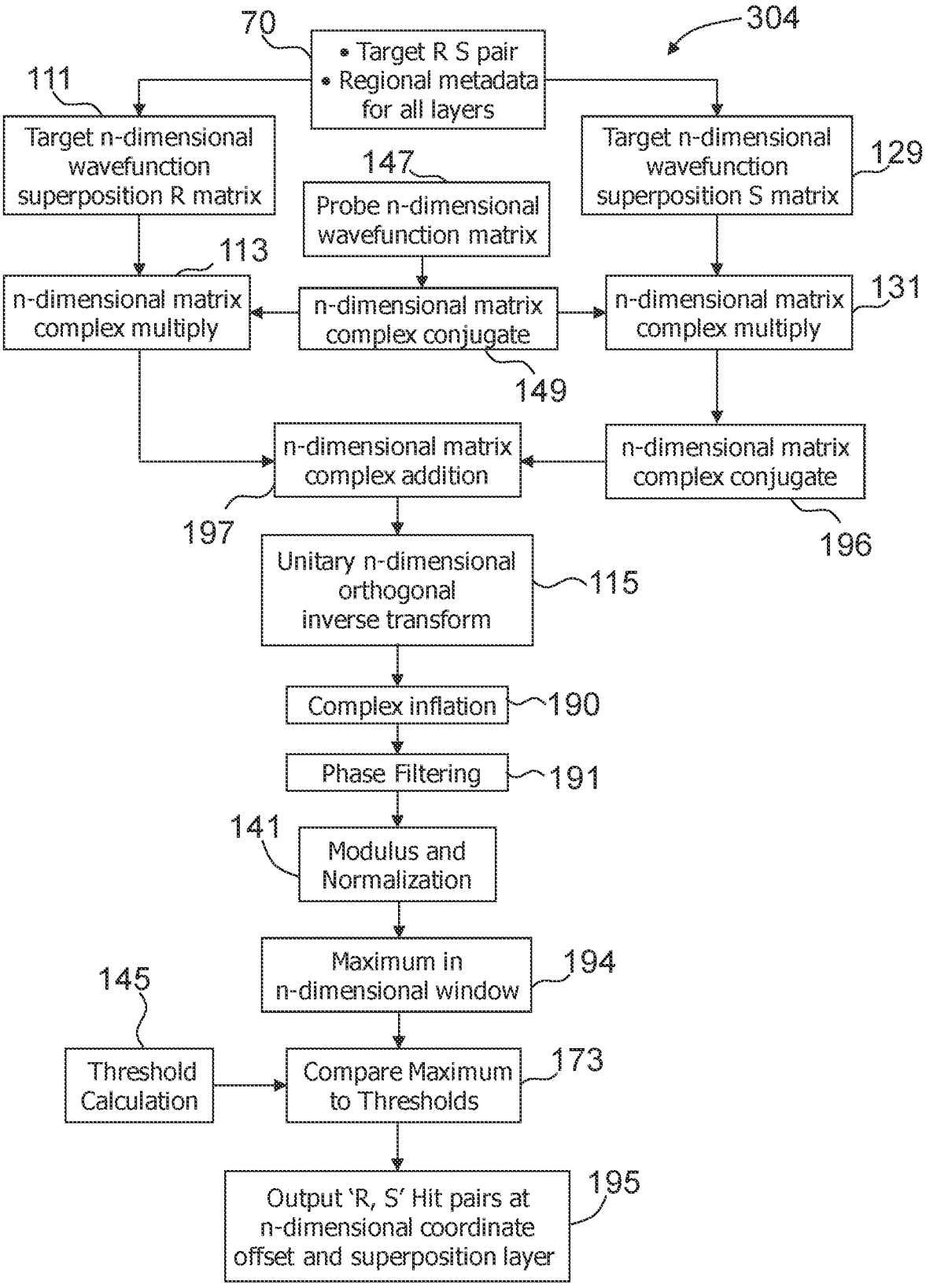
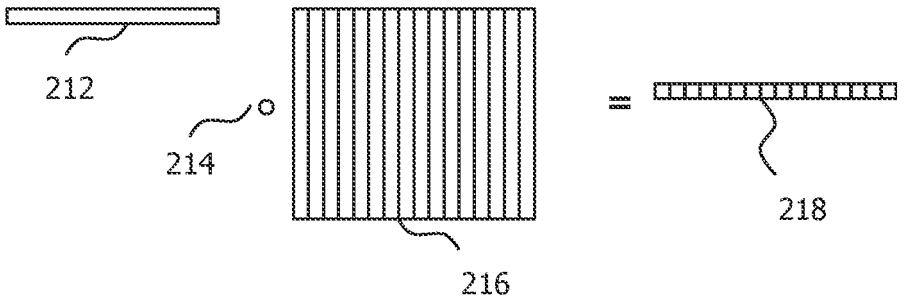

70
• Target R S pair
• Regional metadata for all layers
304
111
Target n-dimensional wavefunction superposition R matrix
147
Probe n-dimensional wavefunction matrix
Target n-dimensional wavefunction superposition S matrix
129
113
n-dimensional matrix complex multiply
n-dimensional matrix complex conjugate
149
n-dimensional matrix complex multiply
131
n-dimensional matrix complex addition
197
n-dimensional matrix complex conjugate
196
Unitary n-dimensional orthogonal inverse transform
115
Complex inflation
190
Phase Filtering
191
141
Modulus and Normalization
Maximum in n-dimensional window
194
145
Threshold Calculation
Compare Maximum to Thresholds
173
Output 'R, S' Hit pairs at n-dimensional coordinate offset and superposition layer
195

Figure 5H

Figure 5I



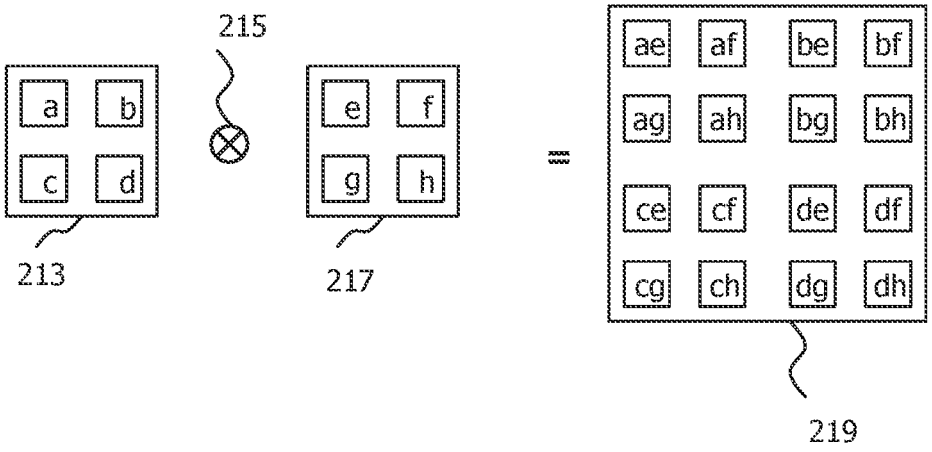


Figure 5I

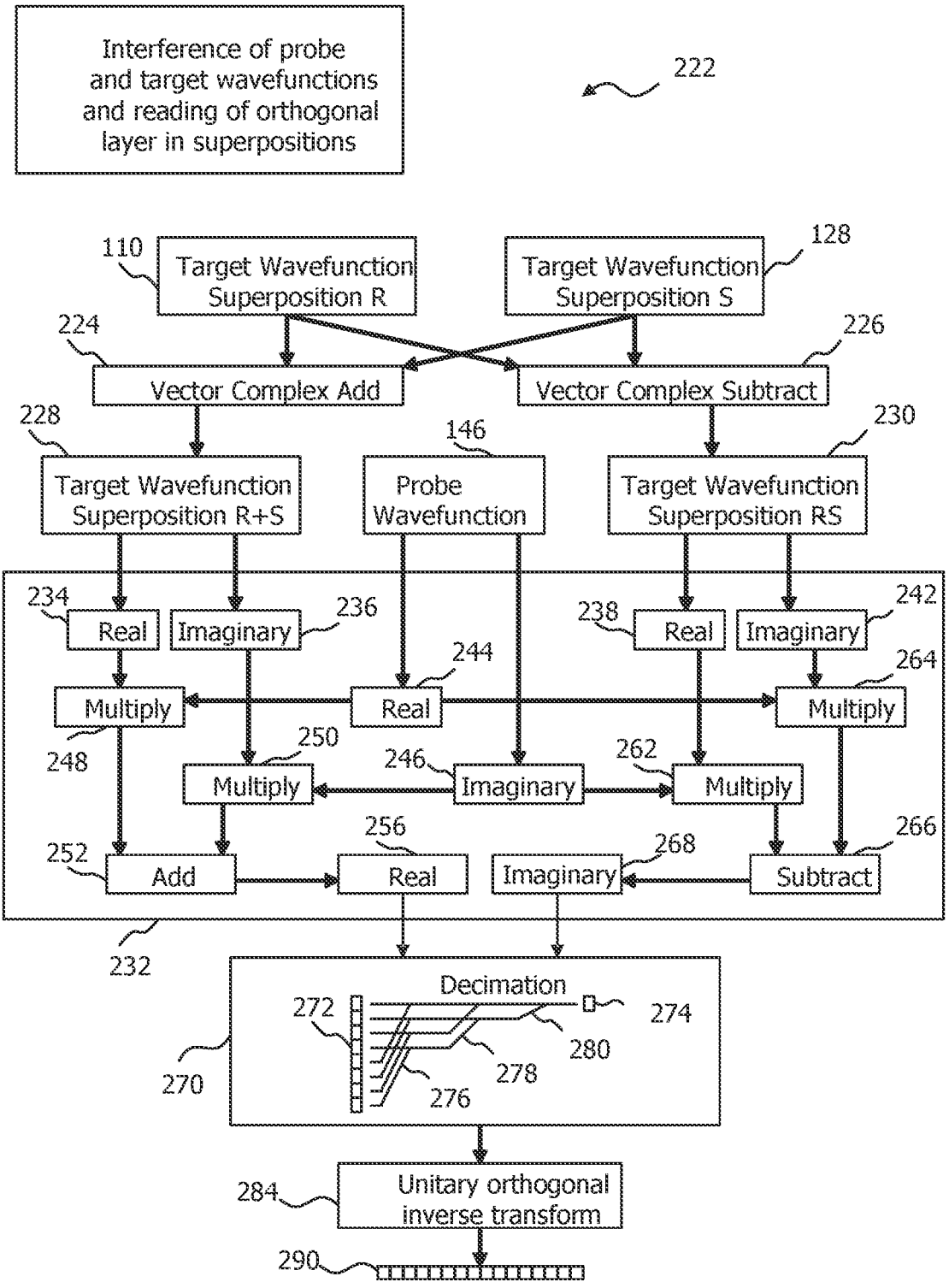

Interference of probe and target wavefunctions and reading of orthogonal layer in superpositions
222
110
Target Wavefunction Superposition R
128
Target Wavefunction Superposition S
224
226
Vector Complex Add
Vector Complex Subtract
228
146
230
Target Wavefunction Superposition R+S
Probe Wavefunction
Target Wavefunction Superposition RS
234
Real
Imaginary
236
238
Real
Imaginary
242
244
264
Multiply
Real
Multiply
248
250
246
262
Multiply
Imaginary
Multiply
256
252
268
266
Add
Real
Imaginary
Subtract
232
Decimation
272
274
280
278
270
276
284
Unitary orthogonal inverse transform
290

Figure 5J

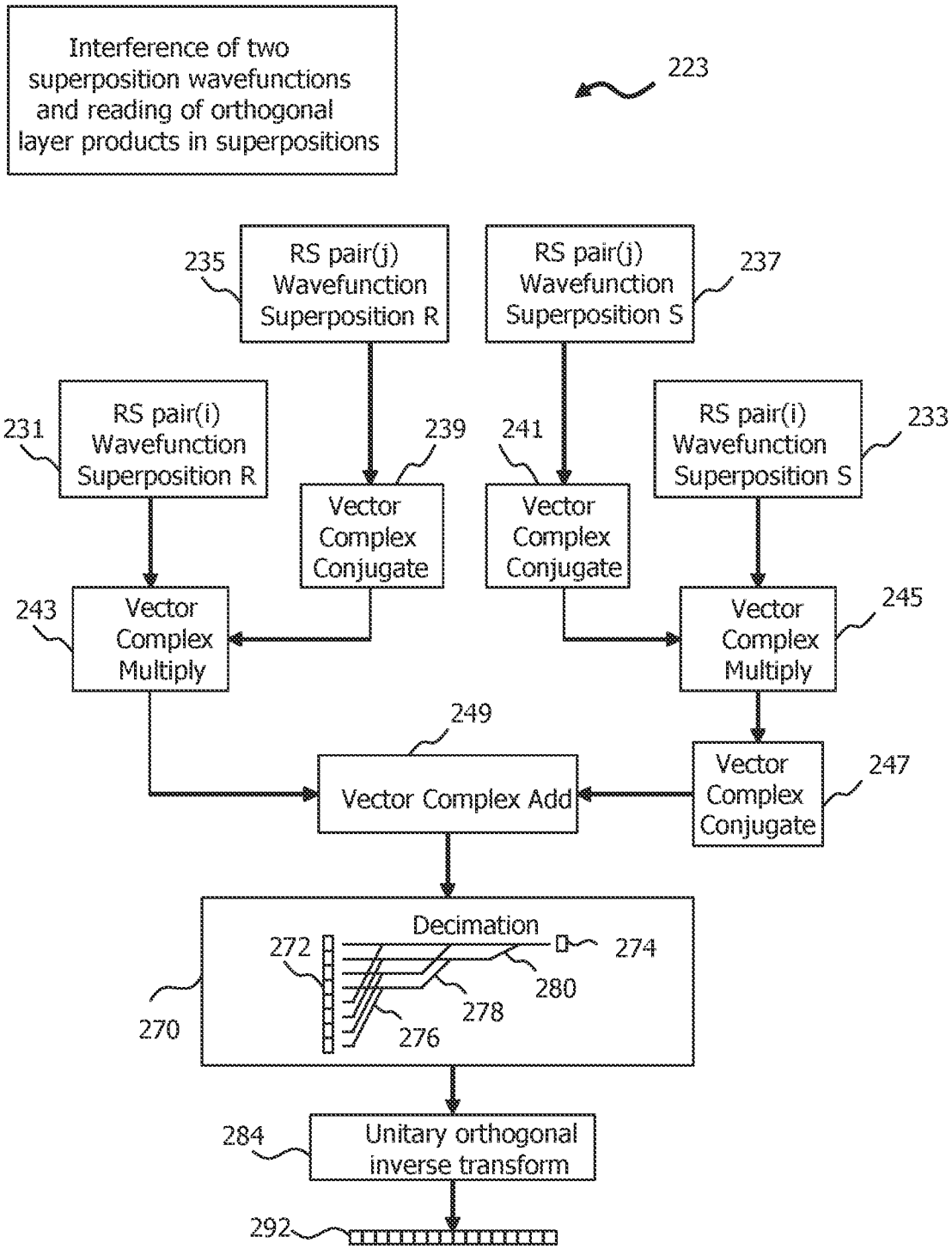

Interference of two superposition wavefunctions and reading of orthogonal layer products in superpositions
223
235
RS pair(j) Wavefunction Superposition R
237
RS pair(j) Wavefunction Superposition S
231
RS pair(i) Wavefunction Superposition R
239
Vector Complex Conjugate
241
Vector Complex Conjugate
233
RS pair(i) Wavefunction Superposition S
243
Vector Complex Multiply
245
Vector Complex Multiply
249
Vector Complex Add
247
Vector Complex Conjugate
Decimation
272
274
280
278
276
270
284
Unitary orthogonal inverse transform
292

Figure 5K

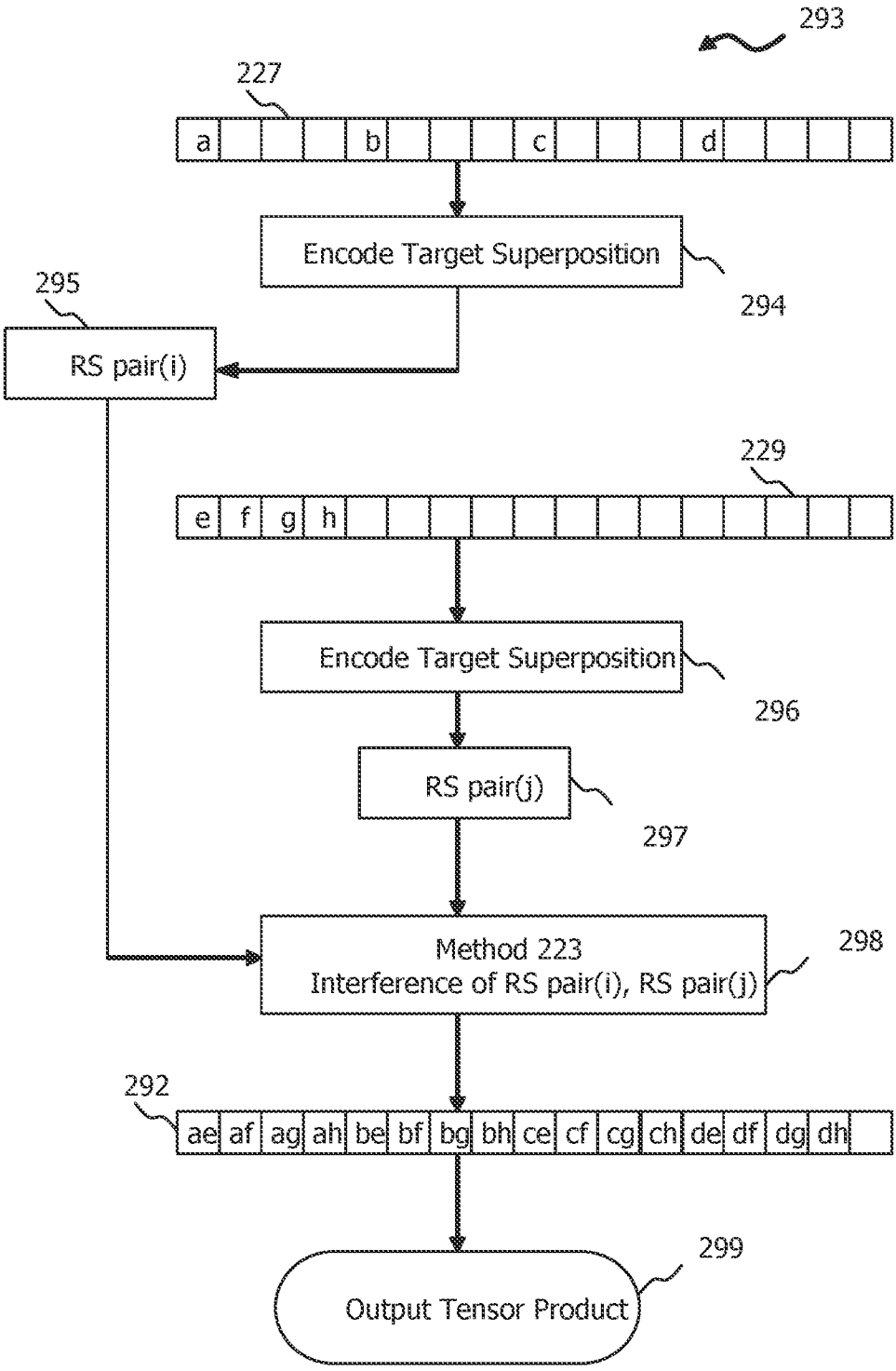

293
227
a
b
c
d
Encode Target Superposition
294
295
RS pair(i)
229
e
f
g
h
Encode Target Superposition
296
RS pair(j)
297
Method 223
Interference of RS pair(i), RS pair(j)
298
292
ae
af
ag
ah
be
bf
bg
bh
ce
cf
cg
ch
de
df
dg
dh
299
Output Tensor Product

Figure 5L

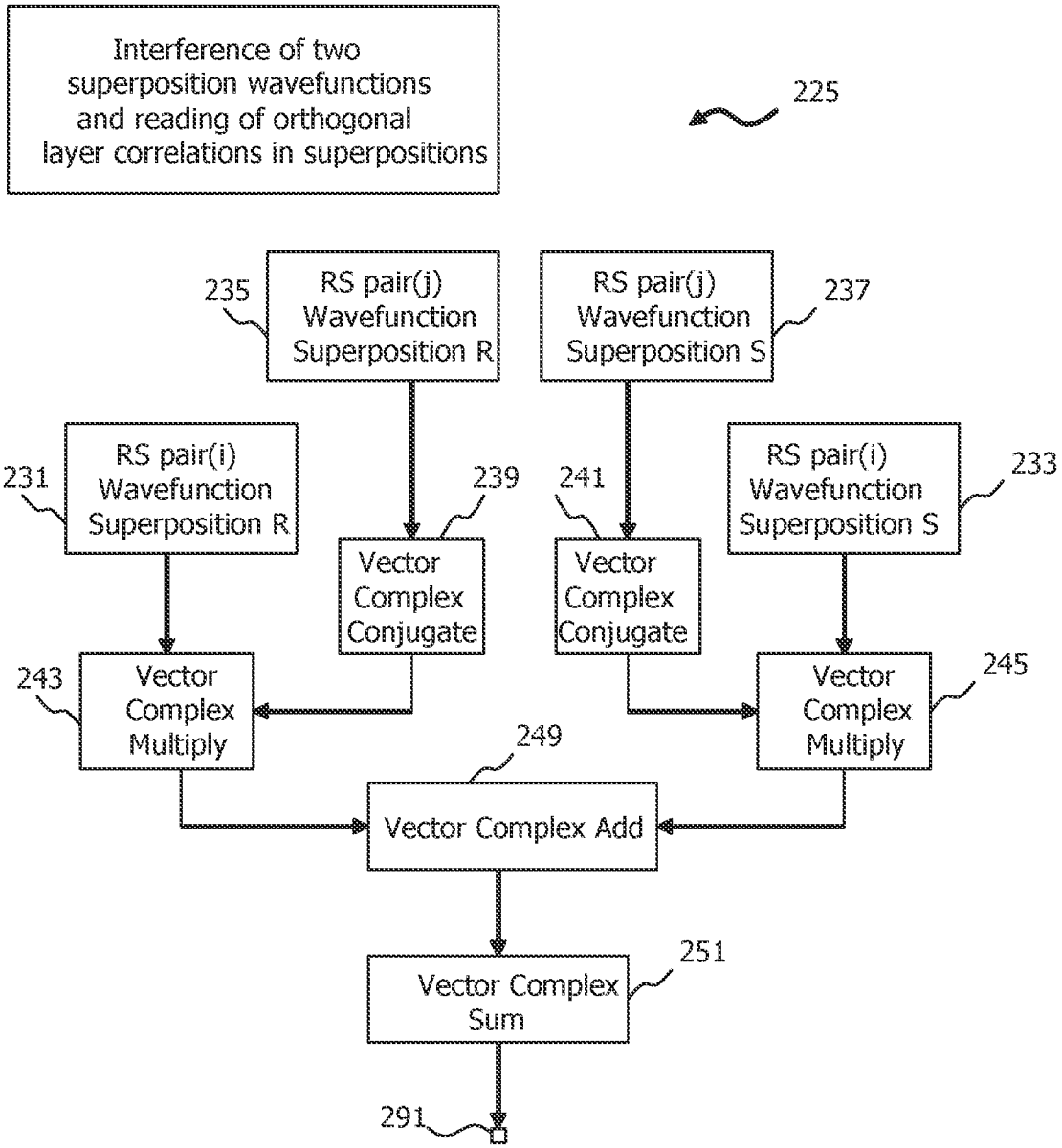

Interference of two superposition wavefunctions and reading of orthogonal layer correlations in superpositions
225
235
RS pair(j) Wavefunction Superposition R
237
RS pair(j) Wavefunction Superposition S
231
RS pair(i) Wavefunction Superposition R
239
Vector Complex Conjugate
241
Vector Complex Conjugate
233
RS pair(i) Wavefunction Superposition S
243
Vector Complex Multiply
245
Vector Complex Multiply
249
Vector Complex Add
251
Vector Complex Sum
291

Figure 5M

505	Row Vector 212 Length	P	
510	Column Vector Length	P	
515	Number of column vectors in matrix 216	M	
520	Wavefunction complex length	Q	
525	Number of layers per RS pair	M	
530	Real Matrix Multiplication (1xP)(PxM)=(1xM)	M×P addition	M×P multiplication
535	Complex Matrix Multiplication (1xP)(PxM)=(1xM)	4M×P addition	4M×P multiplication
540	Interference Butterfly 232	2Q addition	4Q multiplication
545	Decimation 270, d = Q÷ M	2Q addition	
550	Unitary orthogonal inverse transform 284, M complex	3M × log2(M) addition	2M × log2(M) multiplication
555	Total Operations method 222	4Q + 3M×log2(M) addition	4Q + 2M×log2(M) multiplication
560	Quantum Advantage Real Matrix Multiplication	M×P ÷ 4Q + 3M×log2(M) addition	M×P ÷ 4Q + 2M×log2(M) multiplication
565	Quantum Breakeven @ P=Q Real Matrix Multiplication	M = 4 addition	M = 4 multiplication
570	Quantum Advantage Complex Matrix Multiplication	4M×P ÷ 4Q + 3M×log2(M) addition	4M×P ÷ 4Q + 2M×log2(M) multiplication
575	Quantum Breakeven @ 2P=Q Complex Matrix Multiplication	M = 2 addition	M = 2 multiplication
580	Quantum Advantage Data Size	Real: M×P÷ 4Q	Complex: M×P÷2Q

Figure 5N

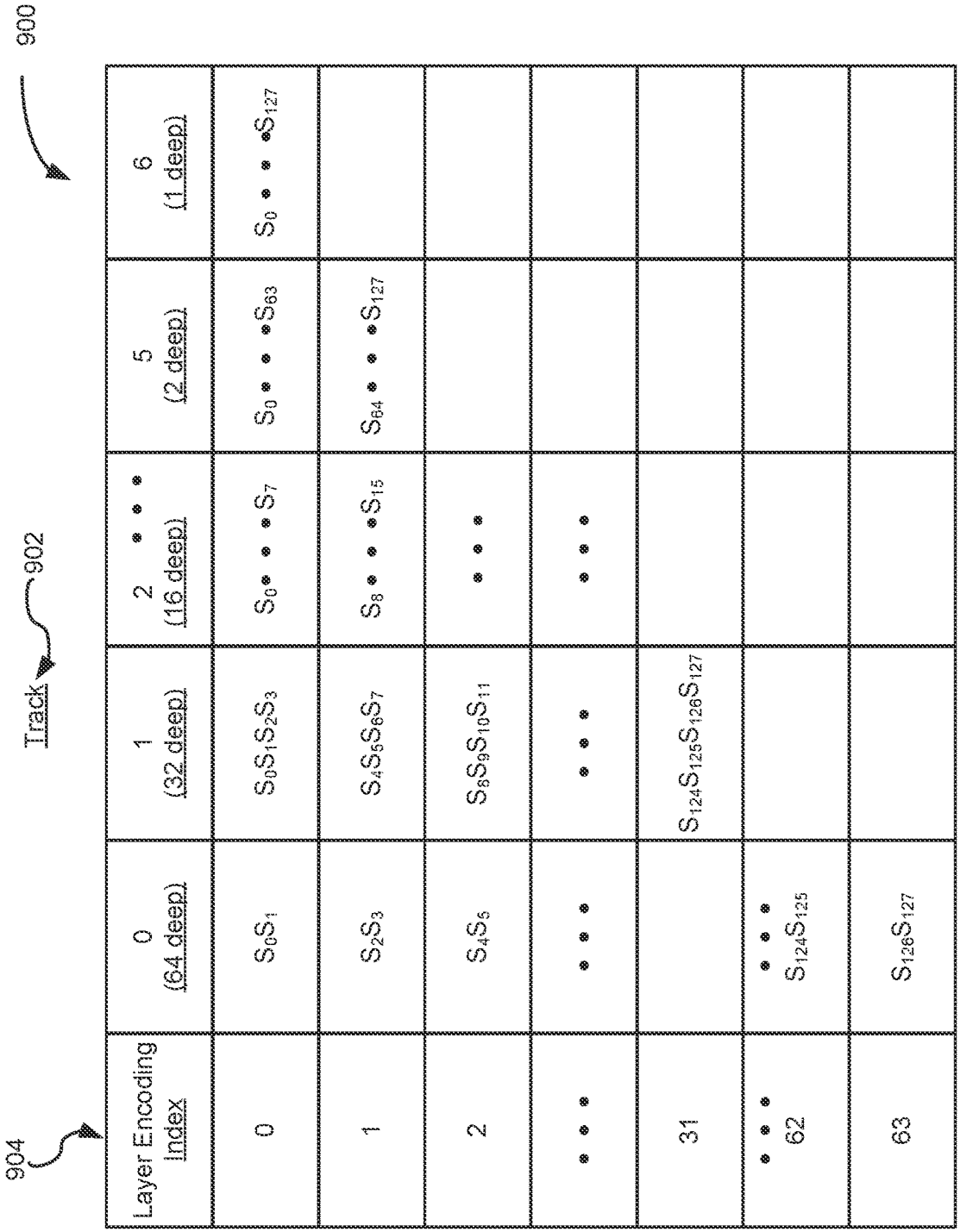

900
902
Track
904
Layer Encoding Index
0 (64 deep)
1 (32 deep)
2 (16 deep)
5 (2 deep)
6 (1 deep)
0
1
2
31
62
63
S_0S_1
$S_0S_1S_2S_3$
S_0S_1
$S_0 \bullet \bullet \bullet S_7$
$S_0 \bullet \bullet \bullet S_{63}$
$S_0 \bullet \bullet \bullet S_{127}$
S_2S_3
$S_4S_5S_6S_7$
$S_8 \bullet \bullet \bullet S_{15}$
$S_{64} \bullet \bullet \bullet S_{127}$
S_4S_5
$S_8S_9S_{10}S_{11}$
$S_{124}S_{125}$
$S_{124}S_{125}S_{126}S_{127}$
$S_{126}S_{127}$

Figure 6

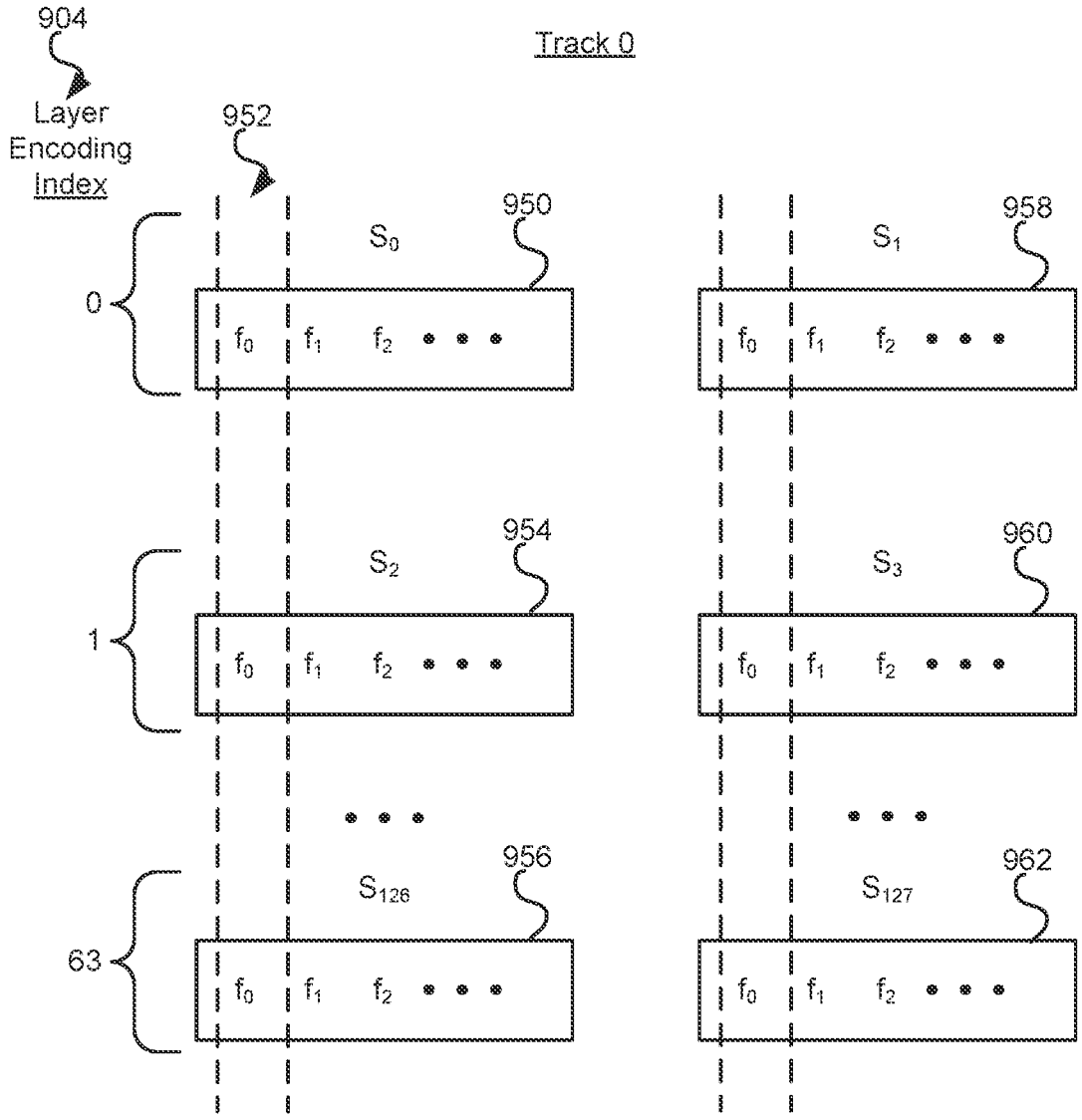

904
Layer Encoding Index
Track 0
952
950
958
954
960
956
962
0
1
63
S_0
S_1
S_2
S_3
S_{126}
S_{127}
f_0 f_1 f_2 • • •

Figure 7

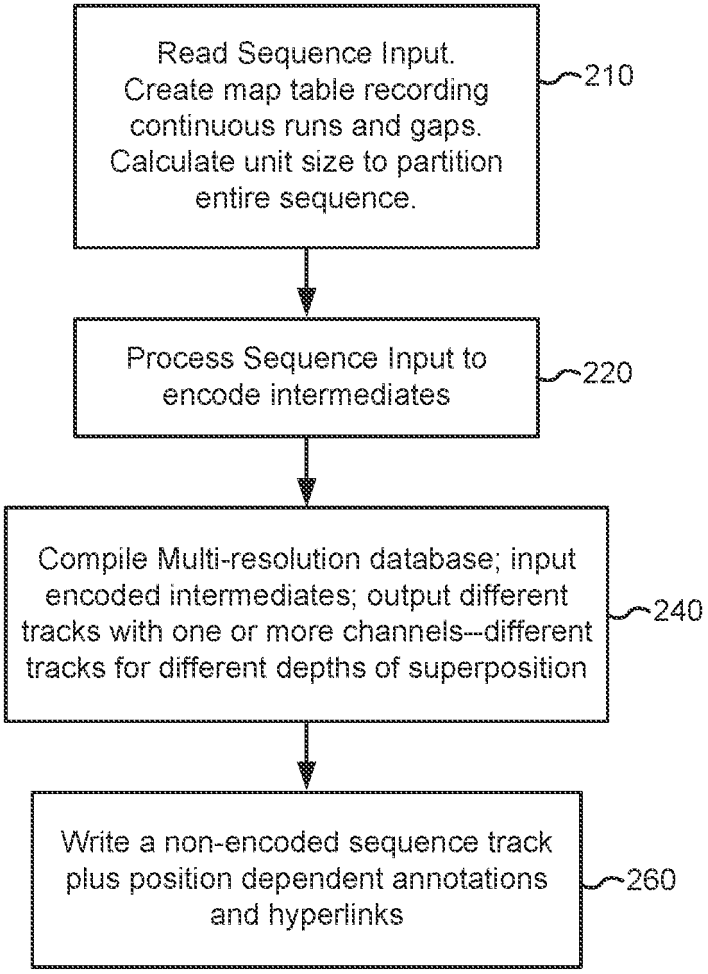

Read Sequence Input. Create map table recording continuous runs and gaps. Calculate unit size to partition entire sequence.
210
Process Sequence Input to encode intermediates
220
Compile Multi-resolution database; input encoded intermediates; output different tracks with one or more channels--different tracks for different depths of superposition
240
Write a non-encoded sequence track plus position dependent annotations and hyperlinks
260

Figure 8

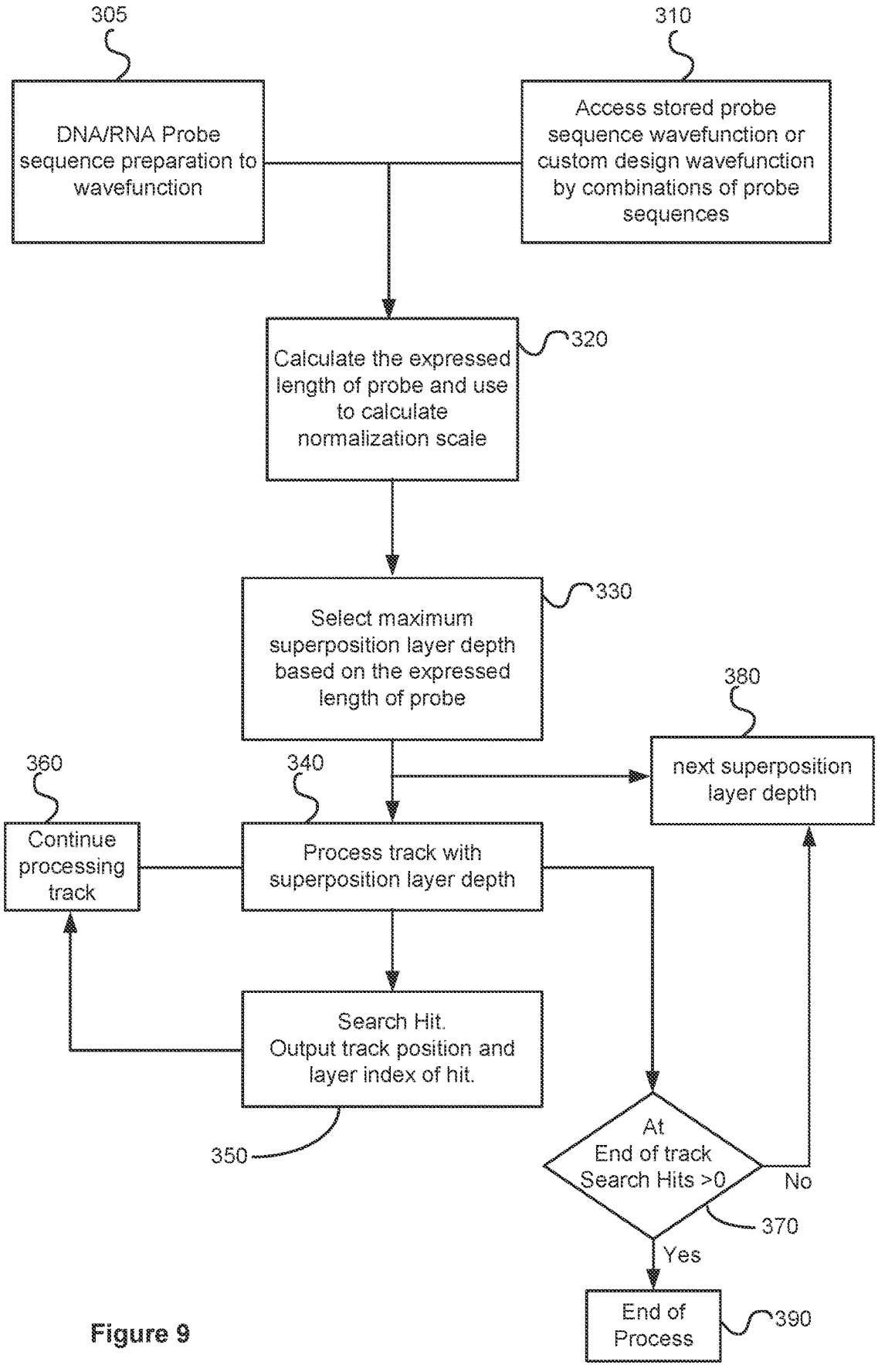

305
DNA/RNA Probe sequence preparation to wavefunction
310
Access stored probe sequence wavefunction or custom design wavefunction by combinations of probe sequences
Calculate the expressed length of probe and use to calculate normalization scale
320
Select maximum superposition layer depth based on the expressed length of probe
330
380
next superposition layer depth
360
Continue processing track
340
Process track with superposition layer depth
Search Hit. Output track position and layer index of hit.
350
At End of track Search Hits >0
No
370
Yes
End of Process
390

Figure 9

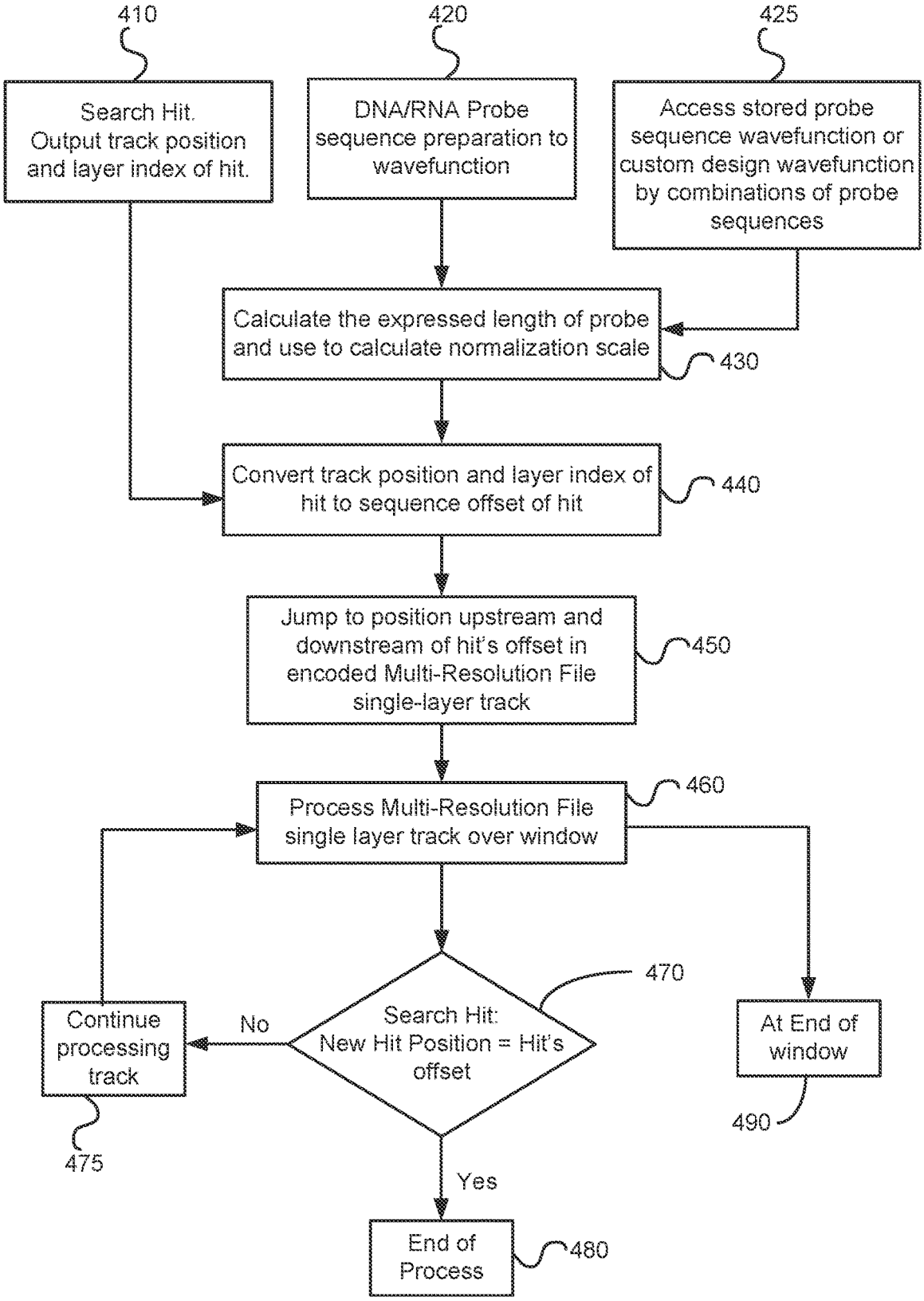

410
Search Hit. Output track position and layer index of hit.
420
DNA/RNA Probe sequence preparation to wavefunction
425
Access stored probe sequence wavefunction or custom design wavefunction by combinations of probe sequences
Calculate the expressed length of probe and use to calculate normalization scale
430
Convert track position and layer index of hit to sequence offset of hit
440
Jump to position upstream and downstream of hit's offset in encoded Multi-Resolution File single-layer track
450
Process Multi-Resolution File single layer track over window
460
Search Hit: New Hit Position = Hit's offset
470
No
Continue processing track
475
Yes
At End of window
490
End of Process
480

Figure 10

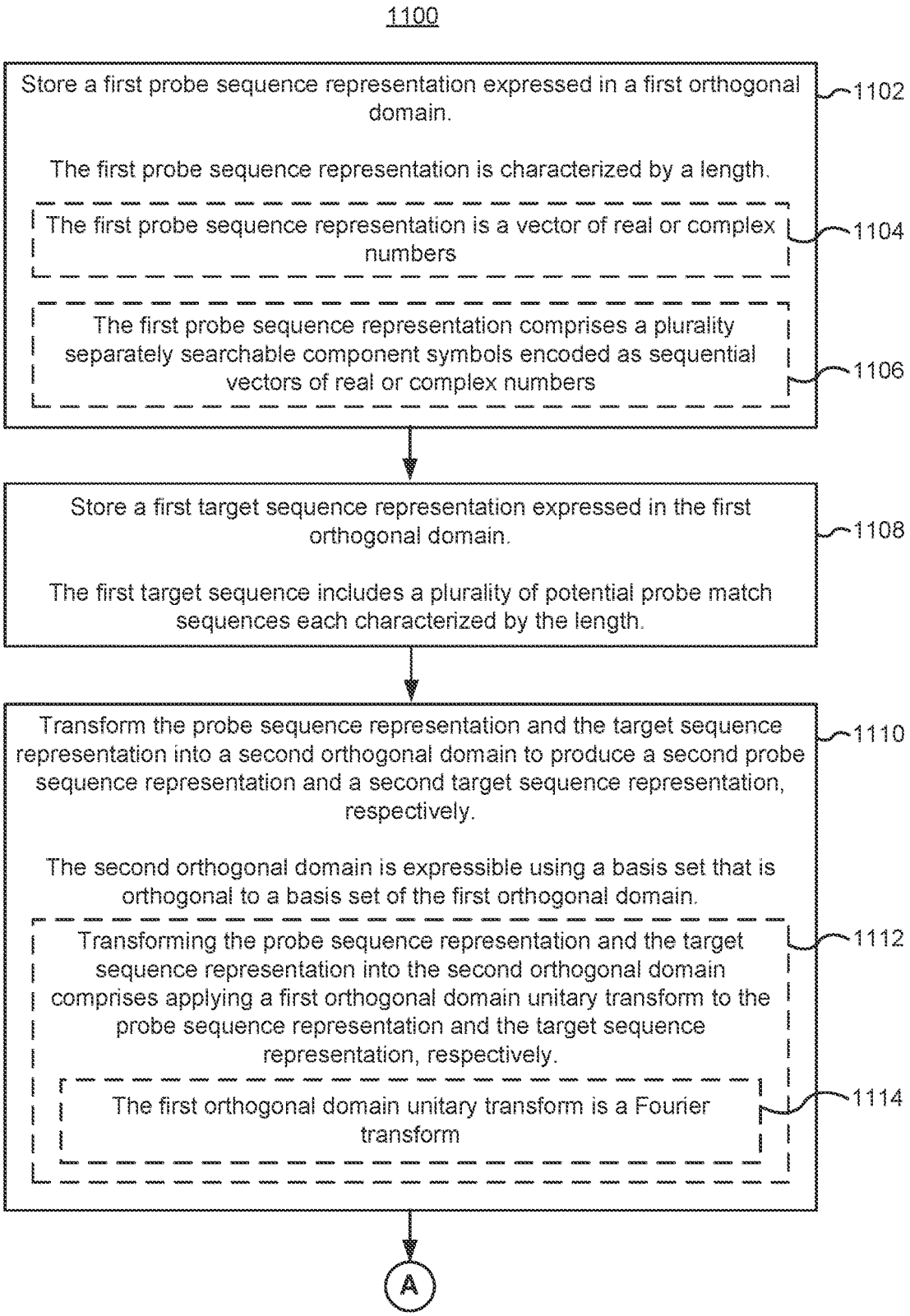

1100
Store a first probe sequence representation expressed in a first orthogonal domain.
The first probe sequence representation is characterized by a length.
1102
The first probe sequence representation is a vector of real or complex numbers
1104
The first probe sequence representation comprises a plurality separately searchable component symbols encoded as sequential vectors of real or complex numbers
1106
Store a first target sequence representation expressed in the first orthogonal domain.
The first target sequence includes a plurality of potential probe match sequences each characterized by the length.
1108
Transform the probe sequence representation and the target sequence representation into a second orthogonal domain to produce a second probe sequence representation and a second target sequence representation, respectively.
The second orthogonal domain is expressible using a basis set that is orthogonal to a basis set of the first orthogonal domain.
1110
Transforming the probe sequence representation and the target sequence representation into the second orthogonal domain comprises applying a first orthogonal domain unitary transform to the probe sequence representation and the target sequence representation, respectively.
1112
The first orthogonal domain unitary transform is a Fourier transform
1114
A

Figure 11A

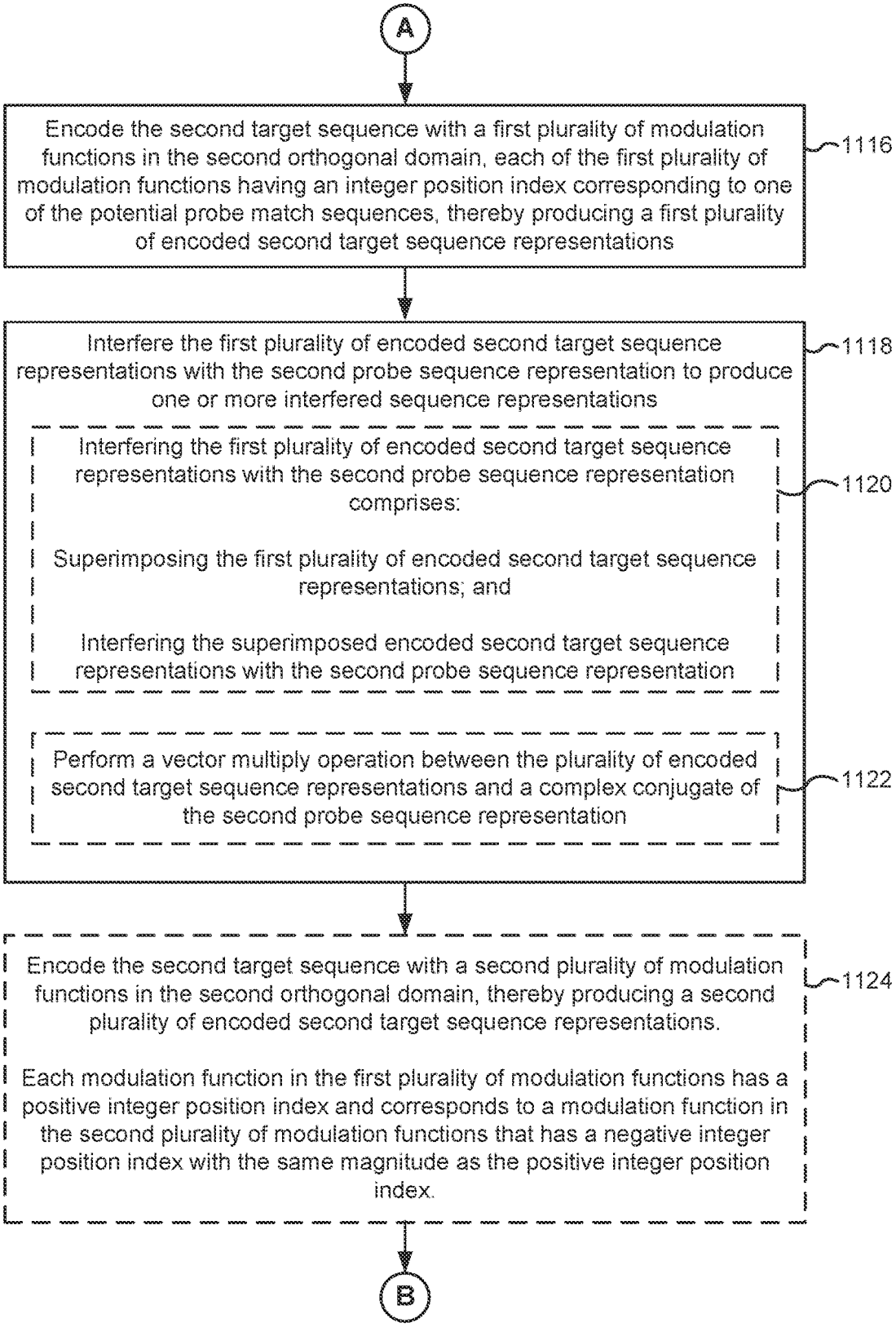

A
Encode the second target sequence with a first plurality of modulation functions in the second orthogonal domain, each of the first plurality of modulation functions having an integer position index corresponding to one of the potential probe match sequences, thereby producing a first plurality of encoded second target sequence representations
1116
Interfere the first plurality of encoded second target sequence representations with the second probe sequence representation to produce one or more interfered sequence representations
1118
Interfering the first plurality of encoded second target sequence representations with the second probe sequence representation comprises:
Superimposing the first plurality of encoded second target sequence representations; and
Interfering the superimposed encoded second target sequence representations with the second probe sequence representation
1120
Perform a vector multiply operation between the plurality of encoded second target sequence representations and a complex conjugate of the second probe sequence representation
1122
Encode the second target sequence with a second plurality of modulation functions in the second orthogonal domain, thereby producing a second plurality of encoded second target sequence representations.
Each modulation function in the first plurality of modulation functions has a positive integer position index and corresponds to a modulation function in the second plurality of modulation functions that has a negative integer position index with the same magnitude as the positive integer position index.
1124
B

Figure 11B

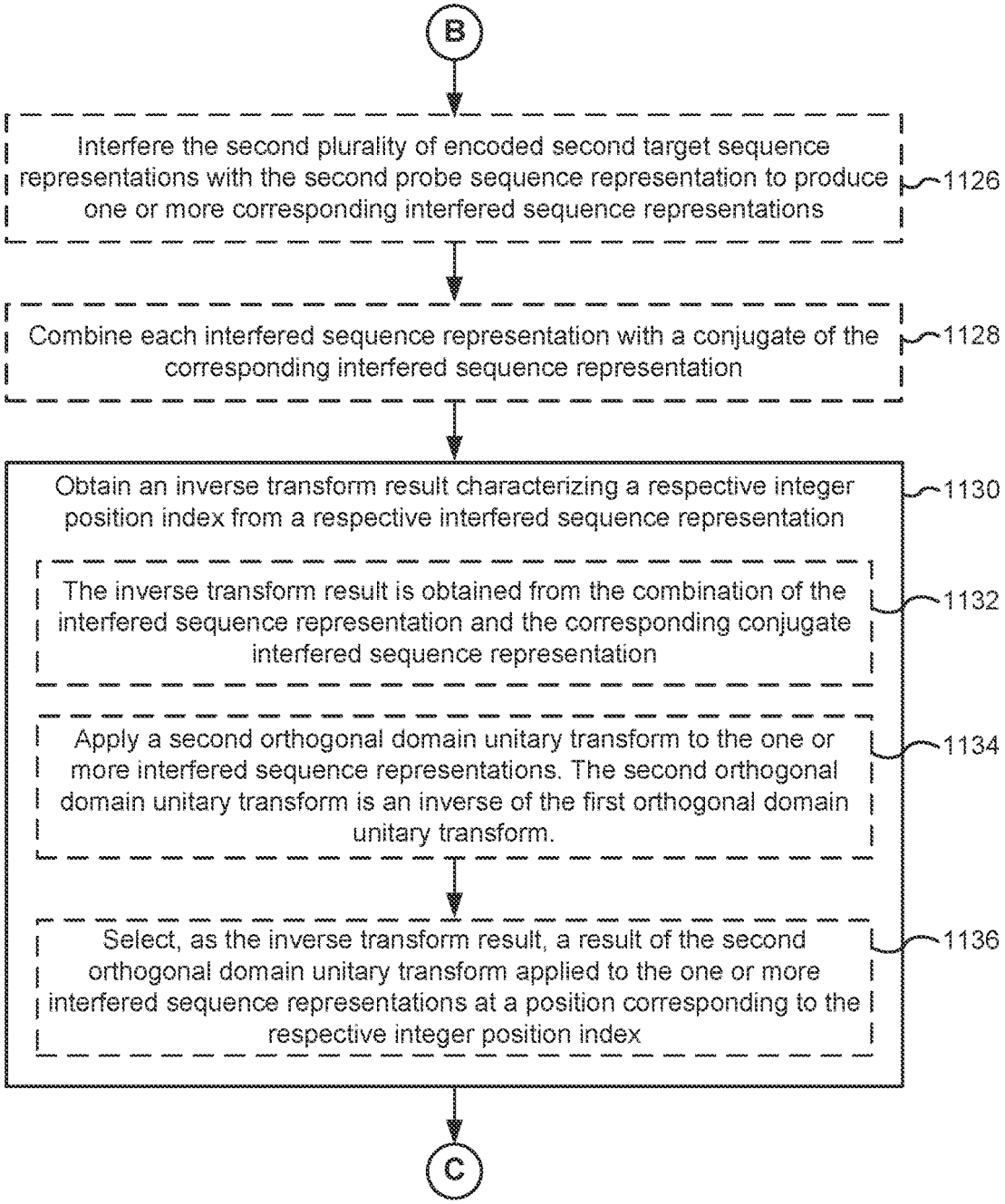

B
Interfere the second plurality of encoded second target sequence representations with the second probe sequence representation to produce one or more corresponding interfered sequence representations
1126
Combine each interfered sequence representation with a conjugate of the corresponding interfered sequence representation
1128
Obtain an inverse transform result characterizing a respective integer position index from a respective interfered sequence representation
1130
The inverse transform result is obtained from the combination of the interfered sequence representation and the corresponding conjugate interfered sequence representation
1132
Apply a second orthogonal domain unitary transform to the one or more interfered sequence representations. The second orthogonal domain unitary transform is an inverse of the first orthogonal domain unitary transform.
1134
Select, as the inverse transform result, a result of the second orthogonal domain unitary transform applied to the one or more interfered sequence representations at a position corresponding to the respective integer position index
1136
C

Figure 11C

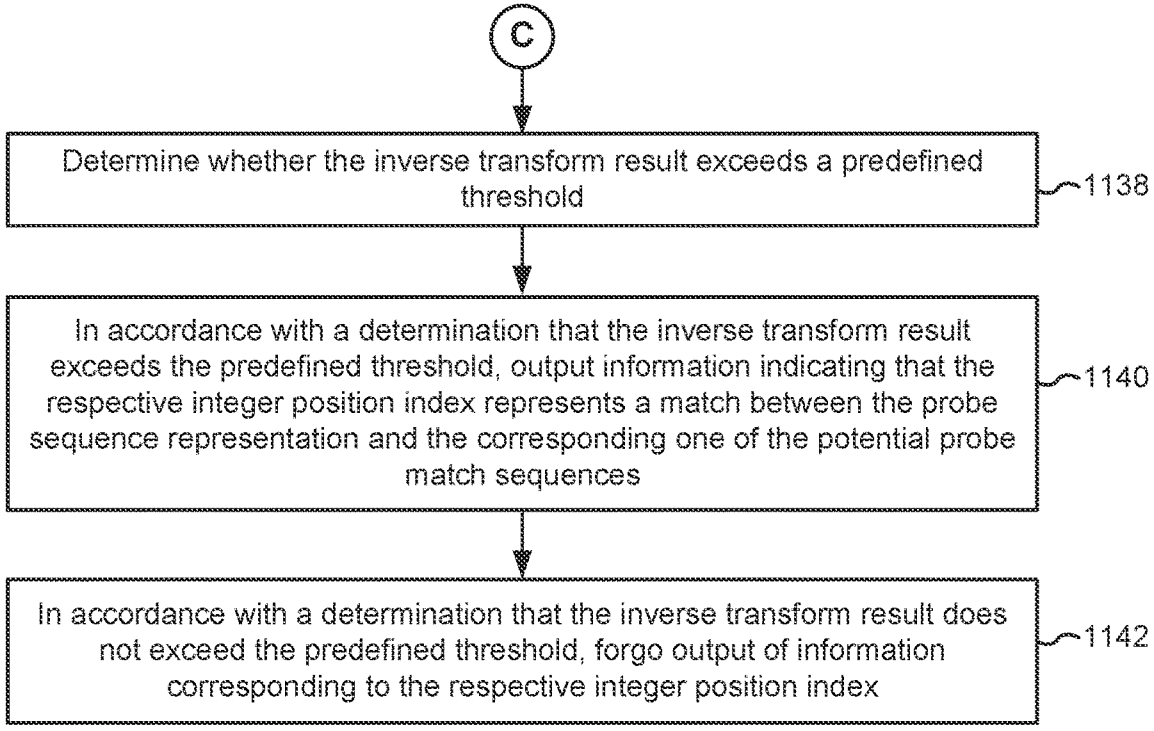

C
Determine whether the inverse transform result exceeds a predefined threshold
1138
In accordance with a determination that the inverse transform result exceeds the predefined threshold, output information indicating that the respective integer position index represents a match between the probe sequence representation and the corresponding one of the potential probe match sequences
1140
In accordance with a determination that the inverse transform result does not exceed the predefined threshold, forgo output of information corresponding to the respective integer position index
1142

Figure 11D

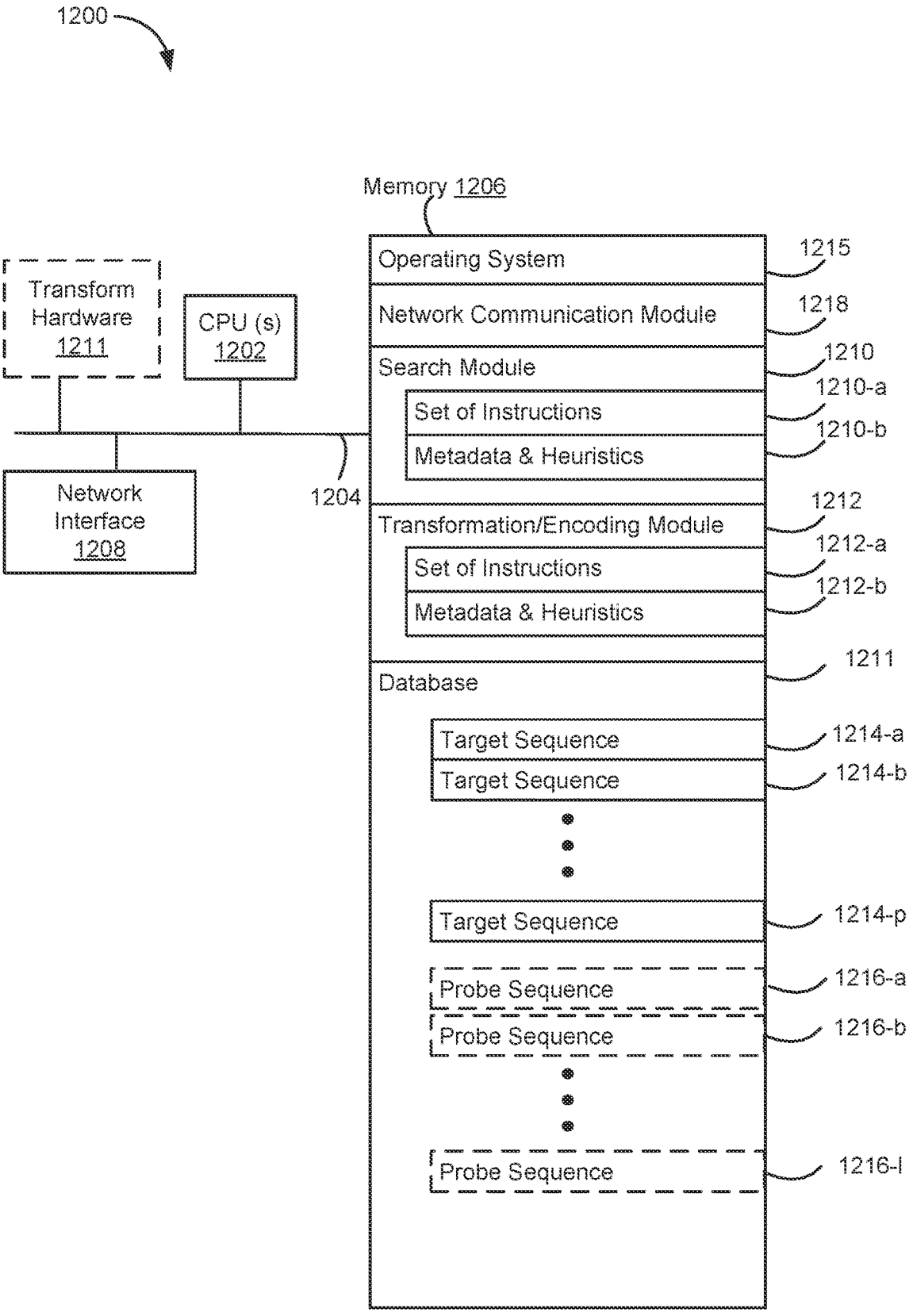

1200
Memory 1206
Operating System
1215
Network Communication Module
1218
Search Module
1210
Set of Instructions
1210-a
Metadata & Heuristics
1210-b
Transformation/Encoding Module
1212
Set of Instructions
1212-a
Metadata & Heuristics
1212-b
Database
1211
Target Sequence
1214-a
Target Sequence
1214-b
Target Sequence
1214-p
Probe Sequence
1216-a
Probe Sequence
1216-b
Probe Sequence
1216-l
Transform Hardware 1211
CPU (s) 1202
Network Interface 1208
1204

Figure 12

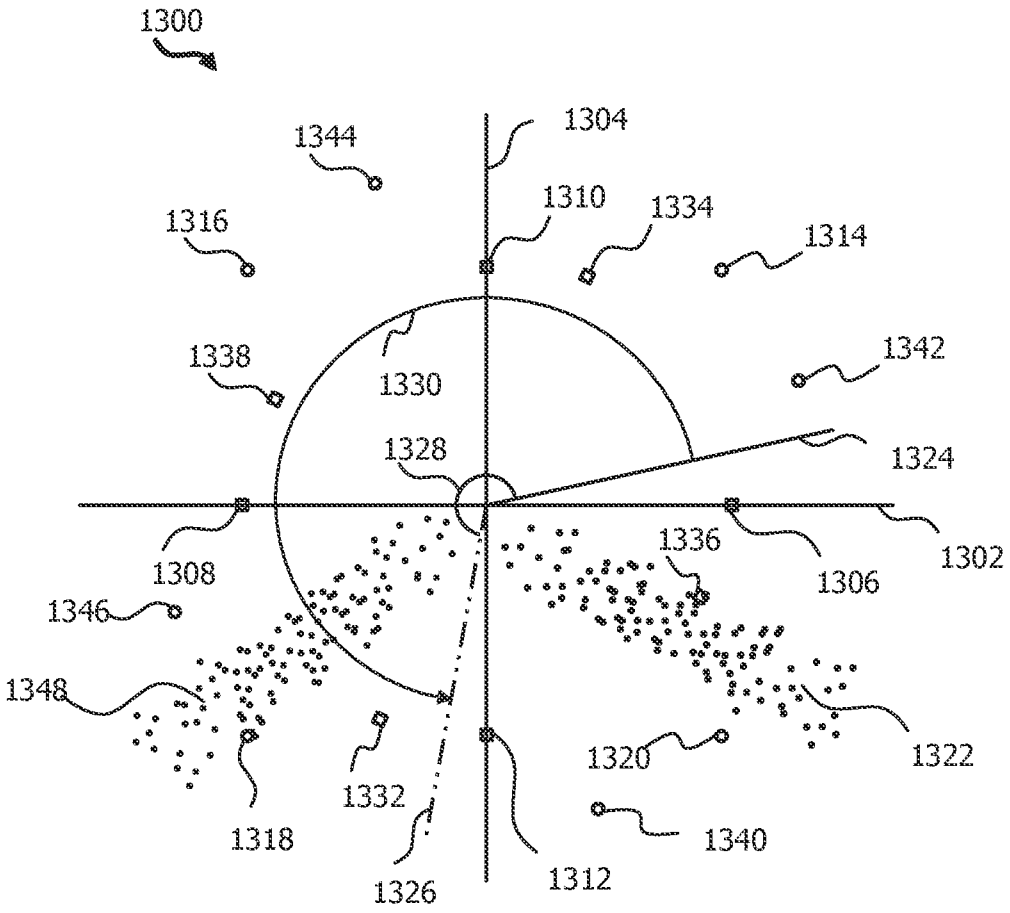

1300
1344
1304
1310
1334
1316
1314
1338
1330
1342
1328
1324
1308
1336
1306
1302
1346
1348
1320
1322
1332
1318
1340
1326
1312

Figure 13

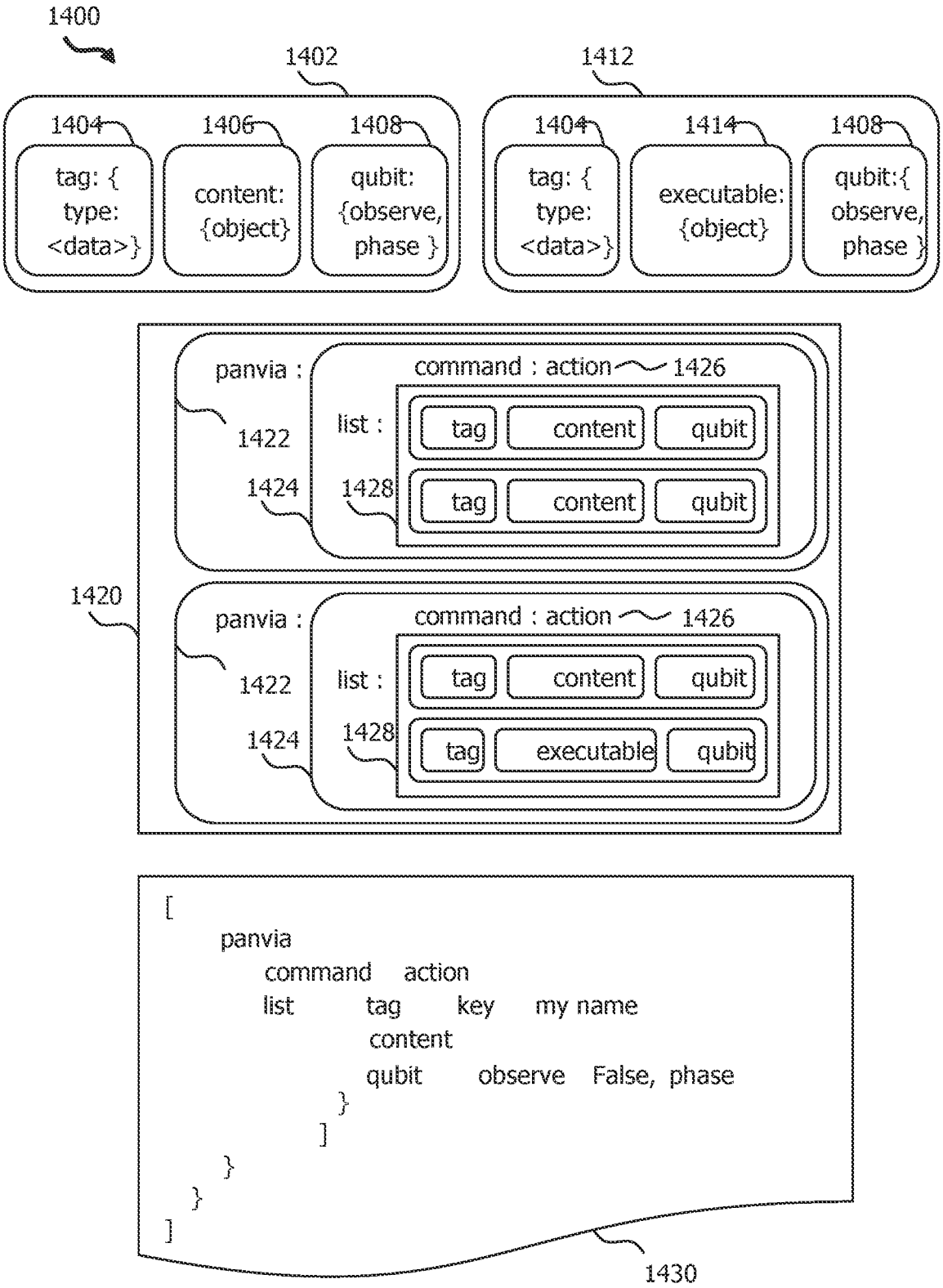

1400
1402
1404
tag: { type: <data>}
1406
content: {object}
1408
qubit: {observe, phase }
1412
1404
tag: { type: <data>}
1414
executable: {object}
1408
qubit:{ observe, phase }
1420
panvia :
command : action
1426
1422
list :
tag
content
qubit
1424
1428
tag
content
qubit
panvia :
command : action
1426
1422
list :
tag
content
qubit
1424
1428
tag
executable
qubit
[
panvia
command action
list tag key my name
content
qubit observe False, phase
}
]
}
}
]
1430

Figure 14

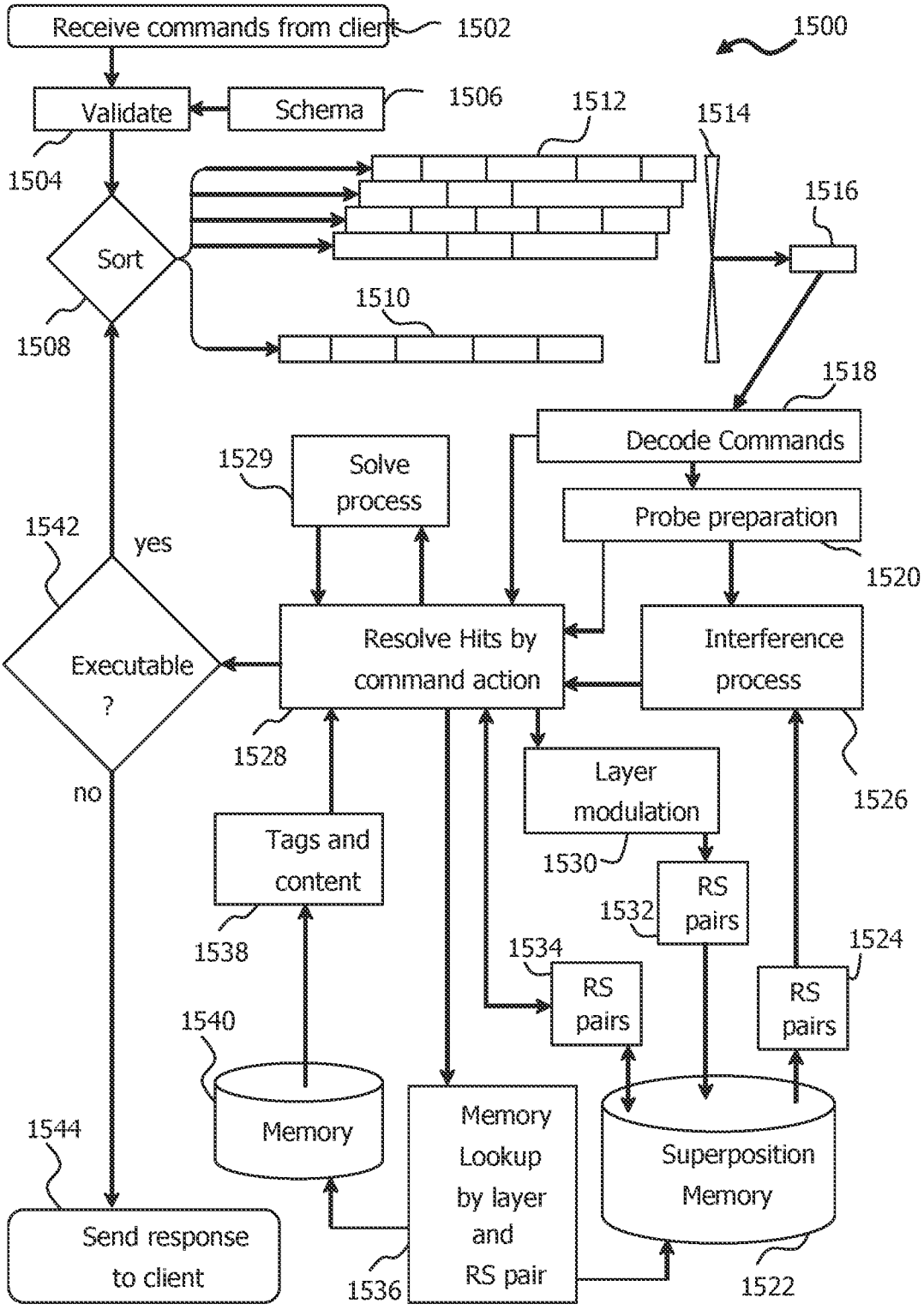

Receive commands from client
1502
1500
Validate
Schema
1506
1512
1514
1504
1516
Sort
1510
1508
1518
Decode Commands
1529
Solve process
Probe preparation
1542
yes
1520
Executable ?
Resolve Hits by command action
Interference process
1528
1526
Layer modulation
no
Tags and content
1530
RS pairs
1532
1538
1534
1524
RS pairs
RS pairs
1540
Memory
Memory Lookup by layer and RS pair
Superposition Memory
1544
Send response to client
1536
1522

Figure 15

1600

action	arguments	function	Description
Store	tag, qubit, content	Ψ +=ϕ @ i, D⇐ x	add tag data layers content, executable
Search	tag, qubit	Ψ.ϕ * ? i,D ⇒ x	retrieve tag layers
Delete	tag, qubit	Ψ.ϕ * ?Ψ -= ϕ @ i	remove tag layers
Inference	unclassified features	P(i) = Ψ.ϕ* ?	Deep Learning Last Layer Evaluation
Dot product	tag, qubit data vectors	R(i) = Ψ.ϕ* ?	Sum of products
Tensor product	tag, qubit data vectors	T = (Ψ.ϕ *?)⊗ (Ψ.θ *?)	Inner products
Correlate	tag, qubit data vectors	r(i) = Ψ.ϕ* ?	correlation by tag and RS layers
Associate	group of (tag, qubit)	Ψ +=ϕ(k) @ i(k) D ⇐ x(k)	Layers in same RS pair
Select	group of (tag, qubit)	∩ Ψ.ϕ(k)*∩ m(k) ? D ⇒ x(k)	Boolean Conditions in same RS pair
Entangle	pairs of (tag, qubit)	Ω =∑ Ψ.ϕ * ?∠ϕ ∀ θ° +=Ω	Rotate dependent θ tag phase
Detangle	pairs of (tag, qubit)	Ω =∑ Ψ.ϕ * ?∠ϕ ∀ θ° -= Ω	- Rotate dependent θ tag phase
Add	pairs of (tag, qubit)	Ψ.ϕ * ? Ψ +=θ @ i	+ dependenθ tag number, size
Subtract	pairs of (tag, qubit)	Ψ.ϕ * ? Ψ -= θ @ i	- dependenθ tag number, size
Trace	tag	Γ =∀ ψ.ϕ*	qubits in registers
Registers	None	Π =∀ ψ	view all registers
Clear	None	Π = {}/∅	clear all registers
Convolve	pairs of (tag, qubit)	Ψ.ϕ * ?ϕ ∉Ψ Δ θ	convolve registers
Transform	pairs of (tag, qubit)	Ψ.ϕ * ? c(k) θ =∑ c(k) .θ(k)	transform probe
Quantum Gate	pairs of (tag, qubit)	Ψt .ϕ * ? α ,β =Ψt .θ* Ψt+1 = W(α ,β)	Quantum Gate on superposition
Solve	pairs of (tag, qubit)	Ψt .ϕ * ? $(J)⇒ Ψt+1	Solve for coupling

Figure 16

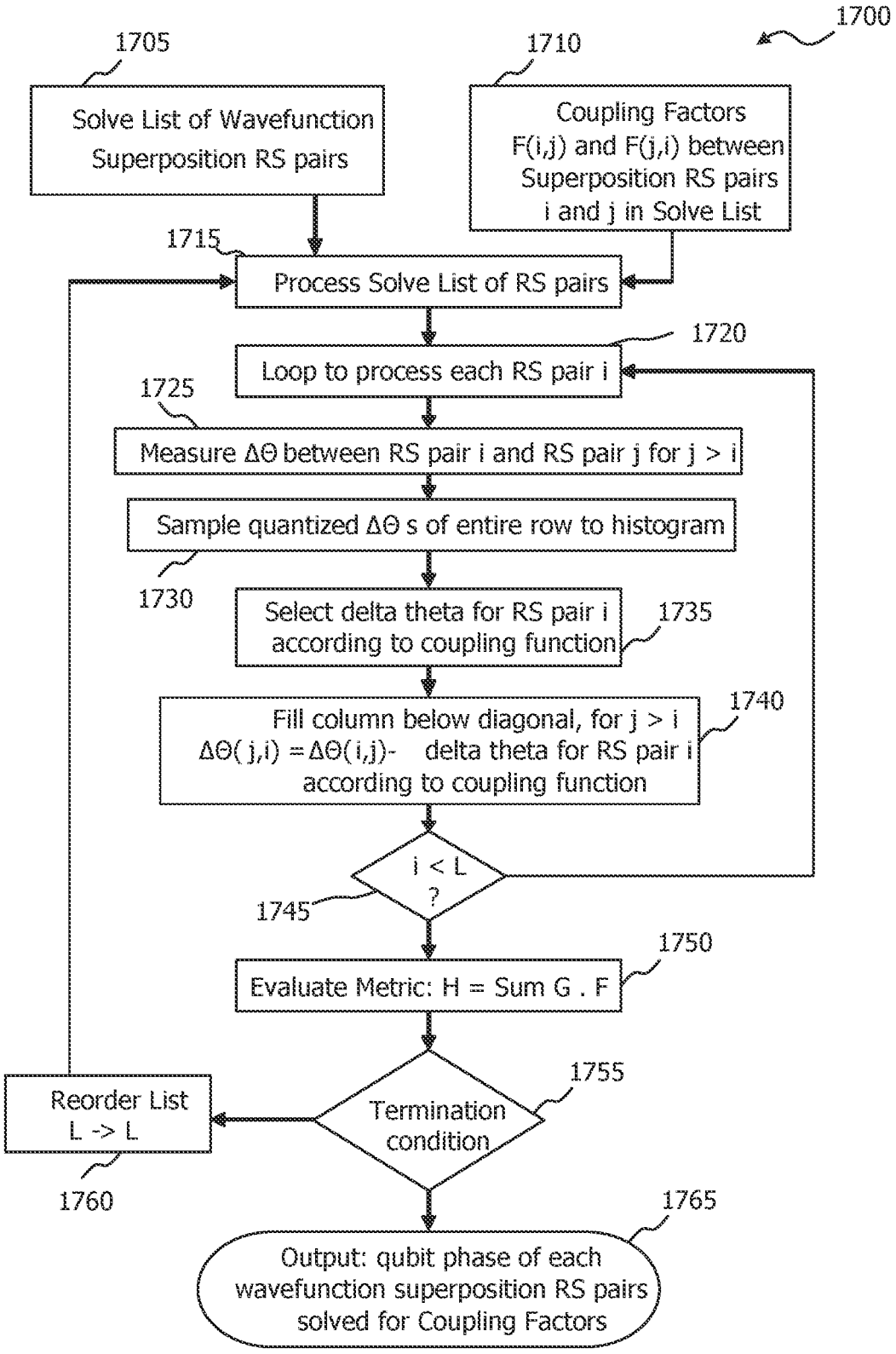
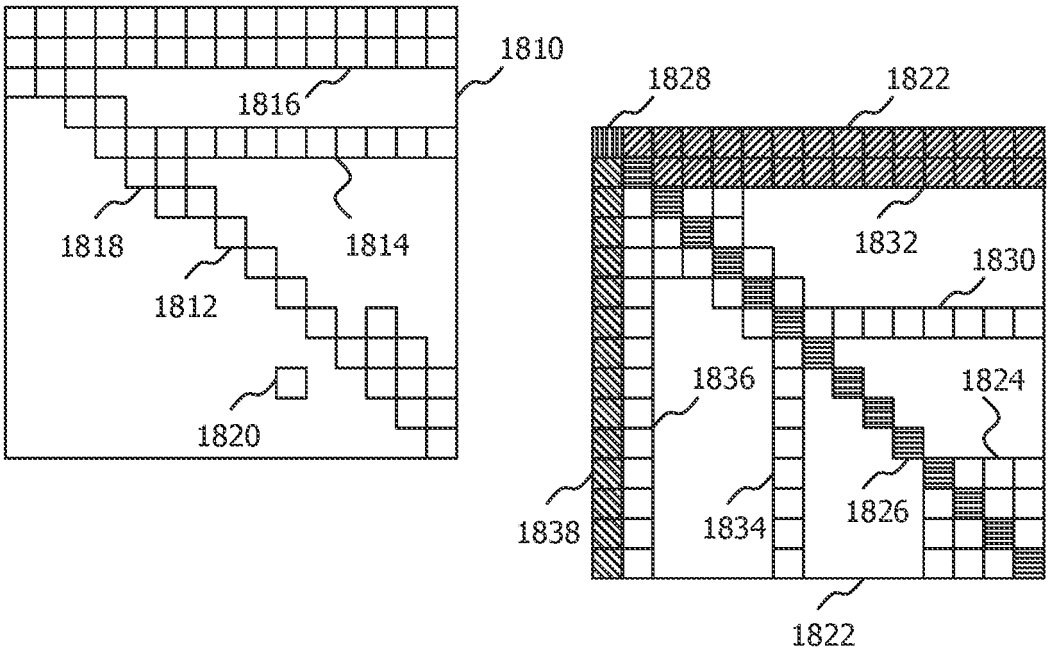

1700
1705
Solve List of Wavefunction Superposition RS pairs
1710
Coupling Factors F(i,j) and F(j,i) between Superposition RS pairs i and j in Solve List
1715
Process Solve List of RS pairs
1720
Loop to process each RS pair i
1725
Measure ΔΘ between RS pair i and RS pair j for j > i
Sample quantized ΔΘ s of entire row to histogram
1730
Select delta theta for RS pair i according to coupling function
1735
Fill column below diagonal, for j > i
ΔΘ(j,i) = ΔΘ(i,j)- delta theta for RS pair i according to coupling function
1740
i < L ?
1745
1750
Evaluate Metric: H = Sum G . F
1755
Termination condition
Reorder List L -> L
1760
1765
Output: qubit phase of each wavefunction superposition RS pairs solved for Coupling Factors

Figure 17


1810
1816
1818
1814
1812
1820
1828
1822
1832
1830
1836
1824
1838
1834
1826
1822

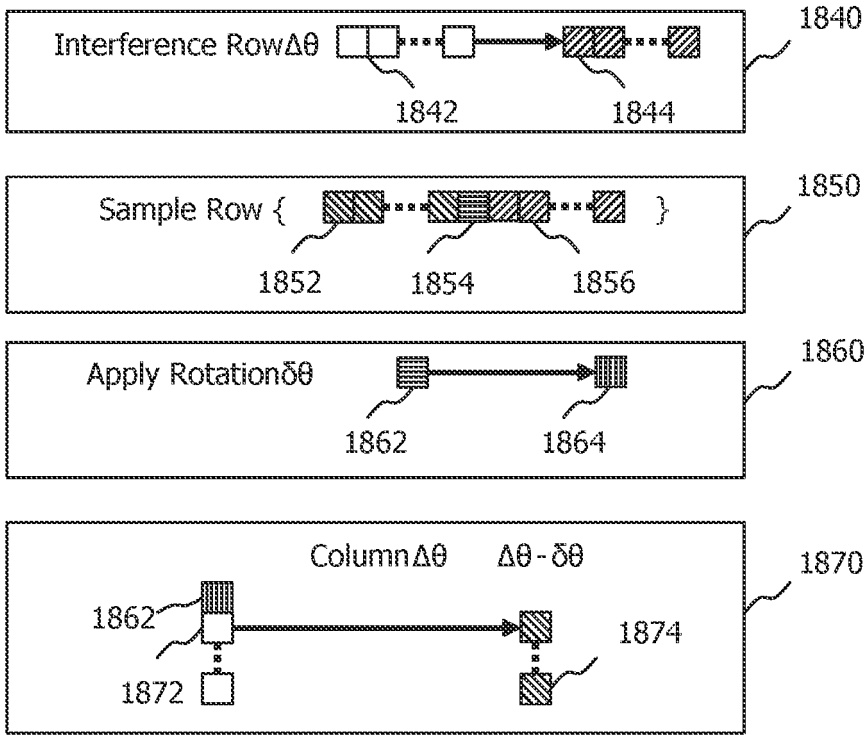

Interference RowΔθ
1842
1844
1840
Sample Row { }
1852
1854
1856
1850
Apply Rotationδθ
1862
1864
1860
ColumnΔθ
Δθ-δθ
1862
1872
1874
1870

Figure 18

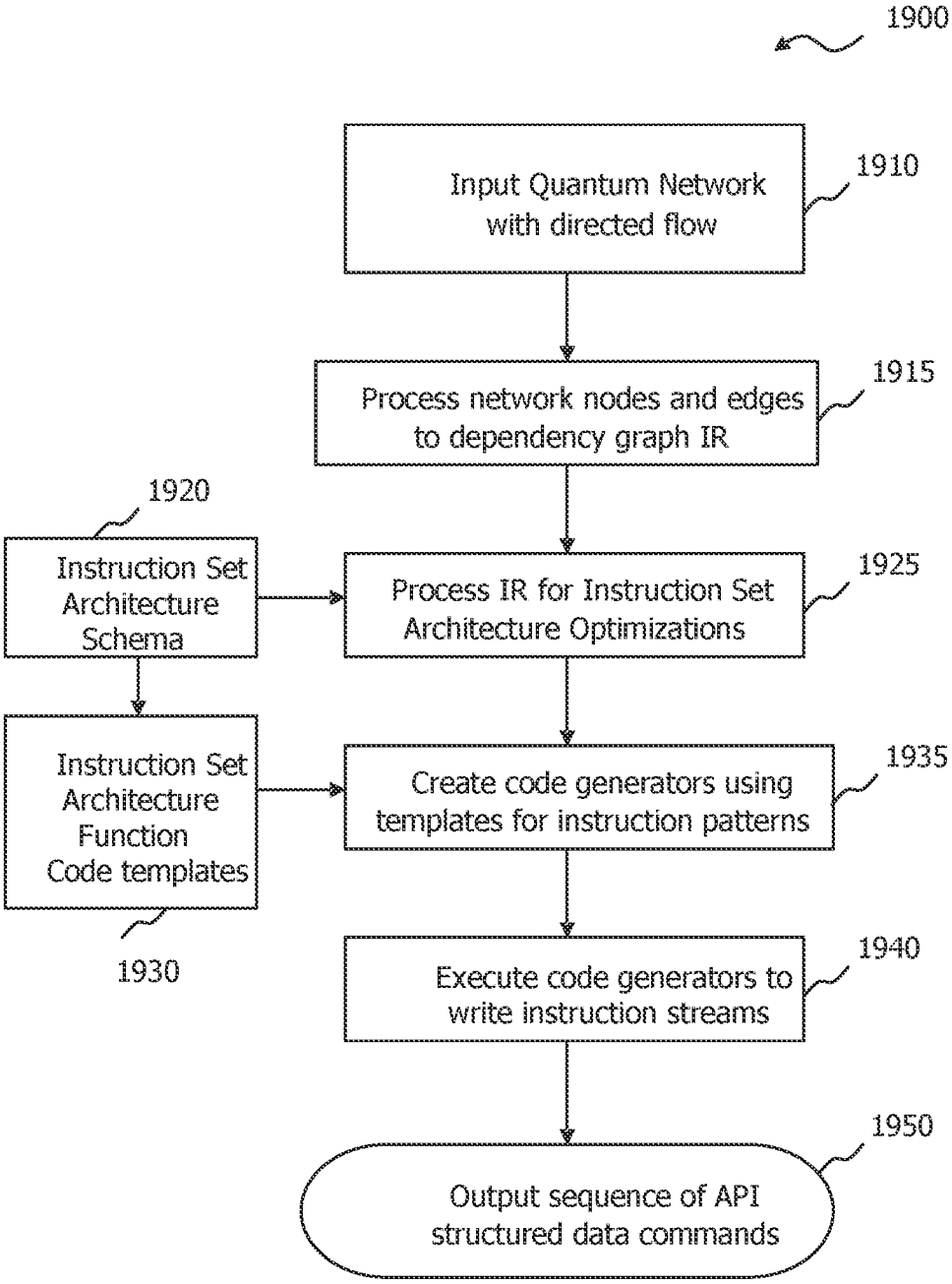

1900
Input Quantum Network with directed flow
1910
Process network nodes and edges to dependency graph IR
1915
1920
Instruction Set Architecture Schema
Process IR for Instruction Set Architecture Optimizations
1925
Instruction Set Architecture Function Code templates
1930
Create code generators using templates for instruction patterns
1935
Execute code generators to write instruction streams
1940
1950
Output sequence of API structured data commands

Figure 19

PROGRAMMABLE QUANTUM COMPUTER

RELATED APPLICATION

This application is a continuation-in-part of U.S. application Ser. No. 16/594,685, filed Oct. 7, 2019 and entitled "Multidimensional Associative Memory And Data Searching," which is a continuation-in-part of U.S. application Ser. No. 14/480,355, filed Sep. 8, 2014 and entitled "Associative Memory and Data Searching System and Method," which is a continuation-in-part of U.S. application Ser. No. 11/914,554, filed Nov. 15, 2007 and entitled "Associative Memory and Data Searching System and Method" which is a national stage filing of PCT/US06/018699, entitled "Associative Memory and Data Searching System and Method" and filed May 15, 2006, which claims the benefit of U.S. Provisional Application Ser. No. 60/681,374, entitled "Associative Memory and Data Searching System and Method" and filed May 16, 2005, each of which is hereby incorporated by reference herein in its entirety.

FIELD

The present disclosure generally relates to a method and system for storing and searching large data sets, and more particularly to a method and system for searching data to locate a match to a query and/or to assess a degree of similarity of a query to information stored in the database.

BACKGROUND

The quest for useful computing devices based on quantum effects can be broken into three separate challenges that must be solved together in order to realize the implementation of a functional quantum computing stack. The first layer of the quantum computing stack is a quantum information state in which the quantum effects of superposition and entanglement are manifest. The second layer of the quantum computing stack is an interfacing method between the quantum information state and a programming language. The third layer of the quantum computer stack is a programming language that allows algorithmic steps to solve problems using quantum parallelism of the quantum superposition and entanglement effects.

All three layers described above are interdependent which in practice means that the nature of the quantum information state necessarily determines what techniques are appropriate for interfacing between the quantum effects native in the quantum information state and a program. The design of the interfacing method is therefore determined by both the quantum information state and the programming language.

The programming language makes the quantum computer stack complete in the sense of being a general-purpose platform that may be adapted for different use cases and applications by means of different expressions of the language. As a result it is important that the programming language is based on an instruction set architecture that is complete in the sense of providing a useful set of functional primitives, and extensible in that more complex procedures can be created by combining primitives.

To date there are several candidate technologies for a viable quantum information state and it is in the nature of each different candidate to have different superposition and entanglement properties and require different interfacing methods. The high cost and low scalability of the current state of the art means that all the candidate technologies are at the point of proving feasibility of the quantum information state rather than developing applications. Since different quantum information states have different needs and requirements of the programming language, there is no accessible low cost development system for quantum computing applications.

Accordingly, it would be desirable to have a solution that overcame the combined challenges of creating and integrating the quantum information state, interfacing method and programming language to provide a programmable quantum computer stack. Furthermore it would be desirable that a programmable quantum computer stack would be low cost and generally accessible in order to enable widely the development of quantum computing applications utilizing data processing gains from the quantum effects of superposition and entanglement.

To date the pursuit of a Quantum Computer device has focused on the development of quantum gates using various candidate technologies for a viable quantum information state. The conditional NOT or C-NOT gate is an example of a quantum gate that is a unitary operator on a pair of qubits. Each qubit is a physical entity with properties represented by a 2 by 2 matrix of four complex numbers. Quantum algorithms such as those by Shor for number factorization and Grover for search use the properties of quantum entanglement and superposition by evolving a quantum state comprising a set of qubits by application of quantum gate operators. The different quantum information state technologies have distinct challenges in establishing an initial quantum state, evolving the quantum state, maintaining coherence of the quantum state and finally reading the solution from the quantum state. While physical quantum information states have noise, the equivalent mathematical model based on the qubit comprising the 2 by 2 matrix of four complex numbers and quantum gate operators as described has the disadvantage of not being scalable to a large number of qubits. The reason for this is that in the conventional mathematical model for the quantum gate operations the output of a two qubit gate is a tensor product operation between the two 2 by 2 matrices of four complex numbers which equals a 4 by 4 matrix of sixteen complex numbers. The data storage and the consequent number computing operations required therefore grows at a rate of two to the power of the number of qubits. In practice this means that conventional computers and even supercomputers can only apply the conventional mathematical model to around forty qubits.

Accordingly, it would be desirable to have a solution that overcame the challenges of scalability of both quantum information state technologies and mathematical representations of the quantum state.

The conventional Quantum Computer model, which represents qubits in C^2, a two-dimensional vector space over the complex numbers and quantum gate operators typically comprising 2 by 2 unitary matrices of four complex numbers, models one type of quantum interaction typified by the spin ½ particle in the Stern-Gerlach experiment conducted in 1922. This quantum phenomenon is modeled by Lie algebra and Special Unitary group two, SU(2), mathematics for rotations in three dimensions. Together these apply to systems with rotational symmetry and the quantization and conservation of angular momentum.

A different physical and quantum phenomenon concerning how energy transfers between a photon and an electron was the basis for The Quantum Theory of the Emissions and Absorption of Radiation by Paul Dirac, also in 1922. According to Dirac "The underlying ideas of the theory are very simple. Consider an atom interacting with a field of

radiation, which we may suppose for definiteness to be confined in an enclosure so as to have only a discrete set of degrees of freedom." The consequence of the enclosure is that the field is quantized and as a result "The number of particles having any specified direction of motion and energy, which can be used as a dynamical variable in the Hamiltonian for the particles, is equal to the number of quanta of energy in the corresponding wave in the Hamiltonian for the waves." According to Quantum Field Theory, in the quantum world energy exists in two orthogonal domains, in one domain it is a particle, in the other it is a wave and as a consequence energy can transition from one state in time to a future one by taking the easiest path, either as a particle or as a wave.

Quantum Field Theory provides an alternative model of quantum effects that may explain far more extended applications of quantum effects in nature. The driving principle of survival is that nature will exploit any mechanism that offers increased survival value. The efficiency of photosynthesis in plants is one example where quantum effects in the chromophore's molecular vibrations assist in energy transfer to raise efficiency to a level of 92% which cannot be explained using classical laws.

Quantum Field Theory also provides an alternative model of quantum effects that may explain how consciousness arises and operates. Cellular scaffolding comprised of proteins self-assembled to microtubules have been suggested to have quantum properties of vibration used by neurons and also in more primitive forms, even by single-celled organisms such as a paramecium to process information without a brain or neurons. The survival value of information processing applies to even the most primitive lifeforms and to date we do not have adequate models to explain how such information processing occurs. The more successful natural mechanisms for being responsive to a hostile environment and predicting the future as a result of past experiences will be retained as a result of natural selection. Although the Penrose-Hameroff model of quantum consciousness based on the microtubule is controversial, it is supported at the biochemical and molecular level by a specific interaction of anaesthetic agents with the microtubules. This result is consistent with the idea that such microtubule quantum consciousness arose early in evolution and was retained in first primitive neural cells specializing in information processing and subsequently retained in the neurons of higher organisms.

Apart from vibrating microtubules there are many possible biological structures that can together interact to exhibit repetitive patterns of activity which may generate cyclostationary signals. These are patterns that arise from combining different frequency components to form distinct signatures. Our understanding of patterns of brain activity is still limited however it is feasible that the underlying activity is processing and storing information as a quantum field represented by patterns of connections and modes of activity. The interesting and non-classical property of a quantum field is that it is complex since different timings and spatial distributions of the brain activity would correspond to different phases of the complex field variables.

Any biological mechanism developed by natural selection for increasing survival chances of the individual by predicting the future and choosing actions as a result of past experiences should have the properties of being able to form deep chains of association of different real-world input stimuli and also the flexibility to learn from a single exposure to an experience or specific stimuli. Current Artificial Intelligence efforts based around Deep Learning in Deep Artificial Neural Networks require training sets of large number of examples, typically thousands of labeled exposures of different training examples. These methods may form deep chains of association however they are not flexible to learn from just a few or even a single exposure to specific stimuli. The unsupervised Deep Learning methods have been shown capable of developing a suite of behavioral responses appropriate to a range of trained scenarios, for example learning winning strategies for playing a particular video game by training on a data set comprising many thousands of recorded human game scenarios. Currently these methods do not respond well to the type of game changing events that occur in the real world which are better tests of survival ability using intelligence to make decisions as to future actions.

Accordingly, it would be desirable to have a solution that allowed trained Deep Artificial Neural Networks to be calculated with a Quantum Computer.

Accordingly, it would also be desirable to have a solution that overcame the separation between Quantum Computing models based on either (i) Quantum Field Theory, or (ii) the quantum gate operator on qubits according to the alternate quantum phenomenon of the conservation of quantized angular momentum by Lie algebra and Special Unitary group two, SU(2), mathematics for rotations in three dimensions, in a manner that combined the features of each model.

Furthermore, it would be also desirable to have a solution that combined the use of Quantum Field Theory and the quantum gate operator on qubits to expand the range of applications of a Quantum Computer to include the modeling of Quantum Consciousness and the ability of a system to derive survival value by predicting the future based on current observations and previous experiences in a way that can both develop deep chains of association between different stimuli and also exhibit the flexibility to modify the learned behavior based on a single experience, in other words a combination of deep and one shot learning.

An associative memory is potentially a quantum information state in which the quantum effects of superposition and entanglement are manifest. An associative memory is therefore possibly the first layer of the quantum computing stack.

The function of an associative memory is commonly used to detect cache "hits" in a computer system by comparing an address word with a memory of address words previously accessed. A "hit" occurs when there are a match between the input address word and an entry in this database. The output of this hit is the cache line where the address was previously read into. An associative memory is therefore in essence a parallel recognition process where a new input is compared with the entire database of prior experiences to detect any match, and in the case of a hit, to output the reference or location.

Parallel recognition processes are conceptually simple, but in actual existing practice grow exponentially in complexity and are unfeasible except for the most limited and small database applications. One potential application of parallel recognition processes is in the searching of DNA/RNA sequences. Such an application of computers to solve information processing problems in the life sciences area is within the general field known as "bioinformatics." However, searches of DNA/RNA sequences typically involve very large databases potentially containing millions to hundreds of millions of bases. These size ranges are inconsistent with the small database sizes suitable for existing parallel recognition processes.

The bioinformatics field, which, in a broad sense, includes any use of computers in solving information problems in the life sciences, and more particularly, the creation and use of extensive electronic databases on genomes, proteomes, etc., is currently in a stage of rapid growth. In order to better appreciate some of the concepts in the bioinformatics field, it is helpful to discuss some of the basic principles of cells.

A cell relies on proteins for a variety of its functions. Producing energy, biosynthesizing all component macromolecules, maintaining cellular architecture, and acting upon intra- and extra-cellular stimuli are all protein-dependent activities. Almost every cell within an organism contains the information necessary to produce the entire repertoire of proteins that the organism can specify. This information is stored as genes within the organism's DNA genome. Different organisms have different numbers of genes to define them. The number of human genes, for example, is estimated to be approximately 25,000.

Genetic information of all life forms is encoded by four basic nucleotides (adenine, thymine, cytosine, and guanine, which are designated by the letters "A", "T", "C", and "G", respectively). The genes are grouped in the base pairs A-T and G-C, and a DNA sequence refers to the ordering or pattern of the nucleotide bases in the gene. The length of a DNA sequence can be very large, and for instance, a DNA sequence may have between 2,000 and two million base pairs. The make-up of all life forms is determined by the sequence of these nucleotides. DNA is the molecule that encodes this sequence of nucleotides.

Each gene typically provides biochemical instructions on how to construct a particular protein. In some cases multiple genes are required to create a single protein, and multiple proteins can be produced through alternative splicing and post-transcriptional modification of a single gene.

Only a portion of the genome is composed of genes, and the set of genes expressed as proteins varies between cell types. Some of the proteins present in a single cell are likely to be present in all cells because they serve functions required in every type of cell. These proteins can be thought of as "housekeeping" proteins. Other proteins serve specialized functions that are only required in particular cell types. Such proteins are generally produced only in limited types of cells. Given that a large part of a cell's specific functionality is determined by the genes that it is expressing, it is logical that transcription, the first step in the process of converting the genetic information stored in an organism's genome into protein, would be highly regulated by the control network that coordinates and directs cellular activity.

There are approximately three billion different DNA base pairs that may be found in humans, and the particular DNA sequences that each person has are located in 23 pairs of chromosomes that contain about 100,000 individual genes. It is significant that faulty genes can be linked to a large variety of human afflictions. An ability to relate an individual gene directly with a particular medical health problem can lead to predictive tests, treatments, and potential cures for a wide variety of medical problems and hereditary ailments.

Currently, about 2,000 human DNA sequences are known and identified, and these DNA sequences are stored in available databases. The number of known and identified human DNA sequences is only a small fraction of the enormous total number of human DNA sequence combinations, and the number of such known and identified DNA sequences is growing rapidly. In addition, the number of DNA sequences of other organisms that have been identified and that are available in databases is also large and likewise growing with time.

The DNA sequence information contained in these growing databases will be a major instrument for basic medical and biological research activities for many years. This information will also be a basis for developing curative techniques for medical and hereditary afflictions. In order to use effectively the information in these enormous and growing databases, it is necessary to provide an efficient means to access that information. In particular, it is necessary to provide an efficient and reliable means to compare a given DNA sequence to the library of known DNA sequences in the databases. Such a comparison is useful to identify, analyze, and interpret that given DNA sequence.

Current procedures for making such comparisons are comparatively slow and impractical. As the amount of stored information increases, current search methods will become unable to function with practical, short processing times, and these methods will have very slow operating speeds. Existing technology is not practical for searching large-scale DNA databases, which may have three billion or more base pair data items.

In addition to the above limitations in searching DNA sequence databases, another of the current limitations on drug discovery research involving the analysis of genome structure and function is the need to perform wet DNA hybridization assays because accurate "in silico" simulations are not available. Further, existing sequence matching tools, such as BLAST, often miss important sequence motifs since they lack the resolution to detect short sequences (e.g., less than 14 bases in length).

In addition to the above limitations in searching DNA sequence databases and drug discovery research, a further limitation is the ability to analyze the biomolecular activity of macromolecules encoded by DNA, whether those are composed of RNA or protein. The analysis of biomolecular activity of macromolecules has been long recognized in the field of biochemistry as being composed of primary structure, secondary structure, tertiary structure and quaternary structure. Primary structure is the one-dimensional sequence of the DNA, RNA or protein. Secondary structure is composed of two-dimensional features comprising two or more primary structure sequences. Examples of secondary structure in RNA are base pairings that form hairpins and stem and loop structures and examples in protein macromolecules are the alpha helix and the beta pleated sheet. In the transformational process of DNA, RNA or protein folding, secondary structure forms first in a process of self-assembly where complementary sequences become proximal and so transition to a lower energy state for the macromolecule. Next, regions of the macromolecule that are differentially hydrophobic and hydrophilic organize to also move to a lower energy state. This involves the hydrophobic regions forming an inner part where water is excluded and the hydrophilic regions assuming an outer part. Complementary three dimensional shapes and charge distributions then fit together to create the tertiary structure that is the resulting three-dimensional shape of the macromolecule. Biological activity often depends on how two or more macromolecules come together to form the quaternary structure, a higher dimensional form that corresponds to a four-dimensional structure. An example is the hemoglobin molecule that self assembles in a pair of pairs structure comprising four separate oxygen binding sites, one in each protein subunit. In this case, the quaternary structure allows the oxygen binding at one heme site to change the oxygen binding

affinity of a separate heme site with the result that cooperative binding occurs. The explanation of how this can happen is given by the Monod-Wyman-Changeux model that proposes two conformational states of the four hemoglobin protein subunits that form a hemoglobin molecule.

From the foregoing it is apparent that the biological activity of proteins and RNA, as interpreted from DNA sequence structure, cannot be fully revealed by a search in one dimension corresponding to the primary structure of the DNA. Instead, methods are required to search for secondary structures, tertiary structures and quaternary structures and combine the search results from each separate type of search.

In order to define and formalize a multidimensional search process such as DNA, RNA and protein folding into multiple conformational states it is desirable to define multidimensional data as comprising values in a multidimensional grid. For example, black and white images comprise a second (vertical) dimension comprising successive rows in a y coordinate of a first (horizontal) one-dimensional sequence of pixels in an x coordinate. This is the direct extension into two dimensions of previously defining one-dimensional data as a position ordered vector containing searchable components. Three-dimensional data is accordingly a sequence of voxels or volume elements in a three dimensional grid addressable in, for example, an x,y,z Cartesian coordinate system. Multidimensional data is likewise a sequence of values in a multidimensional grid, each value addressable as a multidimensional coordinate position with the number of coordinate axes equal to n, the number of dimensions, by definition. Specifically, for an integer n number of dimensions as a general case, the multidimensional data is a sequence of values in a grid of n dimensions, each value addressable as a n-dimensional coordinate position with the number of coordinates equal to n, the number of dimensions. Regardless of how the multidimensional data is physically stored, it is therefore represented as the value of a real or complex function h(k1, . . . , kn) defined over the n-dimensional grid $0 \le k1 \le N1-1, \ldots, 0 \le kn \le Nn-1$, where N1 is the number of grid points in dimension 1 and Nn is the number of grid points in dimension n. In other words, h(k1, . . . , kn) is a real or complex data value at the n-dimensional coordinates of (k1, . . . , kn), where the elision marks between k1 and kn denote coordinates in all the dimension between the first and last dimension. Within this document the words multidimensional and n-dimensional will be used interchangeably to mean any case where the number of dimensions is greater than one and where n-dimensional is a specific case of multidimensional where n is the integer number of dimensions.

Accordingly, it would be desirable to search for two-dimensional sequence data patterns in two-dimensional sequence data. Furthermore, it would be desirable to search for three-dimensional sequence data patterns in three-dimensional sequence data, and to search for four-dimensional data patterns in four-dimensional sequence data. As a general case, it would be desirable to search for multidimensional sequence patterns in multidimensional sequence data.

It has long been appreciated that the search space grows exponentially when increasing the number of dimensions, which results in a combinatorial explosion that conventional methods cannot adequately scale to meet and therefore perform slowly or unreliably. Around 1960, Richard E. Bellman called this challenge the “curse of dimensionality” because of the exponential impact on the Dynamic Programming algorithm. To the present time the same combinatorial explosion remains an unsolved challenge in many diverse fields.

Real-time three dimensional object detection, object tracking and object recognition are examples of computational challenges for autonomous vehicles where fast and accurate performance is very important for safety, potentially life-saving for the passenger, pedestrian, cyclist or scooter rider. In an object detection task a database containing unknown objects is searched for instances of a known query object. In an object recognition task a query of unknown object type is compared against a database of known objects. An object tracking task is similar to object recognition except that location and movement vectors are output instead of a binary decision on the presence or absence of an object and its location.

Accordingly, it would be desirable to have an improved solution that overcomes the exponentially growing complexities and combinatorial explosion associated with existing parallel recognition processes in bioinformatics, image processing object detection and recognition, real-time three-dimensional object detection, recognition and tracking, Computed Tomography (CT) 3D scanners, Light Detection and Ranging (LIDAR) sensors signal processing, Magnetic Resonance Imaging (MRI) tissue scan data processing, medical image processing, weather and satellite data, terrain and oceanographic maps, and in other technical fields, and that dramatically reduces associative memory search and retrieval effort. It would be further desirable to have systems and methods to perform multidimensional data with convenient database access, high-speed processing, improved resolution, accuracy, and cost efficiency.

SUMMARY

To address the aforementioned problems of data processing with quantum superposition and entanglement, the present disclosure provides methods, computer systems, and non-transitory computer-readable storage media for improved data processing with quantum superposition and entanglement. In particular, in some embodiments, a method is performed by a computer system having one or more processors and memory storing instructions for execution on one or more processors. The method includes receiving from a client connection one or more commands comprising an action to be performed, a qubit comprising a phase angle and a Boolean representing an observer true or false, a tag data and a payload data, validating the commands according to rules provided in a schema, queuing commands according to their action, executing batches of commands with the same action, retrieving results from memory which may include further commands to be executed and when no further commands remain to be executed providing a response to the client. In the method, the tag data is a first probe sequence representation or a first target sequence representation and each is expressed in a first orthogonal domain. The first target sequence representation includes a number of potential probe match sequences. Depending on the action, the first probe sequence representation and the first target sequence representation in the first orthogonal domain are encoded at a complex phase angle determined by the qubit phase, the observer true or false, the action and the probe or target sequence, to produce a first encoded probe sequence representation and a first encoded target sequence representation respectively. The method further includes transforming the first encoded probe sequence representation and the first encoded target sequence representation into a second

orthogonal domain to produce a second probe sequence representation and a second target sequence representation respectively. The second orthogonal domain is expressible using a basis set that is orthogonal to a basis set of the first orthogonal domain. The method further includes encoding the second target sequence with a first plurality of modulation functions in the second orthogonal domain, each of the first plurality of modulation functions having an integer position index corresponding to one of the potential probe match sequences, thereby producing a first plurality of encoded second target sequence representations. The method further includes interfering the first plurality of encoded second target sequence representations with the second probe sequence representation to produce a first set of one or more interfered sequence representations. The method further includes encoding the second target sequence with a second plurality of modulation functions in the second orthogonal domain, each of the second plurality of modulation functions having a negative integer position index corresponding to one of the potential probe match sequences, thereby producing a second plurality of encoded second target sequence representations. The method further includes interfering the second plurality of encoded second target sequence representations with the second probe sequence representation to produce a second set of one or more interfered sequence representations. The method further includes combining each of the first set of interfered sequence representations with the complex conjugate of the corresponding counterpart in the second set of interfered sequence representations to create a combined interfered sequence representation and obtaining an inverse transform result characterizing a respective integer position index from the combined interfered sequence representation. The method further includes determining whether the inverse transform result exceeds a predefined threshold. In accordance with a determination that the inverse transform result exceeds the predefined threshold, information is output indicating that the respective integer position index represents a match between the probe sequence representation and the corresponding one of the potential probe match sequences. On the other hand, in accordance with a determination that the inverse transform result does not exceed the predefined threshold, output of information corresponding to the respective integer position index is forgone.

In the present disclosure the quantum information state of the Programmable Quantum Computer comprises a target to be searched represented as a complementary pair of quantum wavefunction superpositions ΨR and ΨS, in previous disclosures referred to as an RS pair, comprising summations of differentially modulated second target sequence representations according to an integer layer index. The quantum information state of the Programmable Quantum Computer further comprises a probe to be searched for represented in the second orthogonal domain. The output of an interference process between a probe or search wavefunction and a target wavefunction superpositions ΨR and ΨS RS pair is according to the method a correlation between the input probe data and the target data at each layer in the wavefunction superposition of the RS pair. The quantum information state of the Programmable Quantum Computer further comprises conservation of energy in each superposition layer of the ΨR and ΨS RS pair between two orthogonal and complementary variables corresponding to the real and imaginary basis functions in the second target sequence representation. Phase rotation of any individual layer in the RS pair of superpositions changes the probability of an interference process detecting the energy of the layer in either, or both, of two complementary states.

According to the present disclosure, the complementary variables corresponding to the real and imaginary basis functions in the second target sequence representation may be read using the interference process between the RS pair and a probe sequence representation encoded at a complex phase determined by its associated qubit phase. The correlation result of the interference process measures the probability along the two orthogonal real and imaginary axes in proportions according to the phase of the probe qubit. The phase of the probe qubit acts as an observer at that phase if the observe Boolean is true to provide a defined observation frame. In the case that the qubit's observe Boolean is false, the probe detects target match hits at any complex phase represented in the RS pair of wavefunction superpositions.

Furthermore, according to the present disclosure individual layers in an RS pair may be rotated by a phase angle, annihilated by subtraction or augmented by addition. Each of these operations on an RS pair may be accomplished by an appropriate vector complex addition operation to each superposition wavefunction in the RS pair.

In the present disclosure an interfacing method comprising actions and instructions operate on the quantum information state in the RS pair as a Quantum Field Register or QFR in a programming language, and the programming language allows algorithmic sequences of operational steps to be performed on Quantum Field Registers in the Programmable Quantum Computer embodiment described herein.

The Quantum Field Register can hold hundreds of different data as separately encoded layers, so the QFR becomes a quantum superposition of hundreds of data records. Interference processes all the data in the superposition together in parallel, with the interference result detecting which layers were matched. Knowing which layers matched allows the original data to be retrieved from memory. Different data sources can all be represented in the dual domain and as a result the QFR can hold superpositions with combinations of binary, real, complex data and cryptographic hash functions in hundreds of separate items together in one physical entity where they are each distinguishable by their encoded layer. The Programmable Quantum Computer, or PQC, in the present disclosure comprises many stored Quantum Field Registers, an execution unit that performs storage and interference processes associated with the QFRs, and software programmable features of the execution unit that operate on an instruction set which is described below.

As stated above, the tag data is a first probe sequence representation or a first target sequence representation and each is expressed in a first orthogonal domain. The first target sequence representation includes a number of potential probe match sequences. Depending on the action, the first probe sequence representation and the first target sequence representation in the first orthogonal domain are encoded at a complex phase angle determined by the qubit phase, the observer true or false, the action and the probe or target sequence, to produce a first encoded probe sequence representation and a first encoded target sequence representation respectively.

It is a feature of the present method that many different comparison functions may be calculated by the same interference between probe and target wavefunctions in layer encoded superpositions. The default comparison function is correlation between the first probe sequence representation and the first target sequence representation that includes a number of potential probe match sequences. The value of a

perfect correlation is unity and is equal to the normalized dot product of the probe sequence vector aligned to the target sequence vector at the match position. In other words, a correlation sum may be computed by scaling the first encoded probe sequence representation and a first encoded target sequence by separate factors whose product is equal to the reciprocal of the variance of the first encoded probe sequence.

From the foregoing it will be apparent that correlation is just one result that may be calculated according to the present method. It is an additional feature of the current method that a dot product may be calculated between a probe sequence and a target sequence. The dot product is the sum of the products of the corresponding members of aligned vectors. Unlike the correlation sum, which is between positive and negative unity, the dot product sum may have any value. In other words, a dot product sum may be computed by scaling the first encoded probe sequence representation and a first encoded target sequence by separate factors whose product is equal to unity.

Another computational sum performed by the current method is the last layer of a Neural Net where a dot product and soft max function is computed between a feature vector, derived from an unclassified input by means of forward propagation using a set of trained weights, and a last layer trained weight vector for one of many classification categories. The input probe tag data is a feature vector generated by presenting unclassified samples to the first or input layer of a neural net and using the set of trained weights to apply forward propagation for all intermediate layers ending with the input data to the last layer of the neural net. Calculation of a dot product between the feature vector and the last layer weight vector is performed by wavefunction interference and is followed by a soft max function applied to the dot product. The output of an inference command is a classification category and an estimated probability of the classification for any unclassified sample that has been recognized.

According to the present method the last layer computation for forward inference of a trained neural net may be performed by the same wavefunction interference process that was previously described for finding a correlation match between probe and target sequences. In the interference process that computes the last layer of a trained neural net the RS pairs comprise superpositions of weights for different trained classifications. The input probe tag data is generated from unclassified samples ending with the input data to the last layer of the neural net as the first probe sequence representation. The interference process uses tag data probes and RS pair encodings of classification category weights, each expressed in as a second orthogonal domain representation, complex conjugation of the probe sequence second orthogonal domain representation, complex multiplication by the target sequence second orthogonal domain representation, and a unitary orthogonal inverse transform to generate an output data that is a parallel computation of the dot product between the last layer input data and the last layer trained weights for each classification category. The dot product is equal to the sum of multiplication products between the last layer input data values and the last layer trained weights corresponding to each input value and is computed for each classification category.

Since the RS pairs comprise superpositions of multiple weight vectors for different trained classifications, the interference process with a probe wavefunction generated from a last layer feature vector achieves parallel comparison of the unknown feature vector with multiple classification categories simultaneously. As a result of this superposition parallelism, the interference process can perform the last layer inference faster, for example processing a larger number of classification categories in the same time or the same number of categories in a shorter time.

The following describes the programming language for some embodiments of the Programmable Quantum Computer according to the present disclosure. Software for the Programmable Quantum Computer is in a structured data form such as, for example, JavaScript Object Notation (JSON) text which is interoperable with virtually all computer languages The instruction set is defined in a Schema that allows software to be validated before it is sent to the machine. The Schema specifies that the software is an array of instruction objects, each of which comprises a command with an action from a range of allowed actions, a tag data which is one of a range of allowed data types and a content data which may be identified as executable software for the Programmable Quantum Computer. The tag is used to store and search for content that may be either any structured data object or executable software for the Programmable Quantum Computer. Any search hits in layers of a QFR that have executable content cause that content to be executed by the machine on the next cycle. For coherence, all the store commands are executed in a phase of the next cycle that is followed by execution of all the search commands. Any search hits in layers of a QFR that have non-executable content cause that content to be sent as a response from the machine to the sender of the software. Since this response is structured data for any other computer language, the response may be translated to the sender computer language and executed as or by a program on the sender computer. In executing the program, the sender computer may send further software to execute on the Programmable Quantum Computer or on other computers in a network.

It is a feature of the current method that data may be in any positive integer number of dimensions and the same steps apply for processing data represented in any number of dimensions.

To address the aforementioned problems with conventional multidimensional data sequence searching, the present disclosure provides methods, computer systems, and non-transitory computer-readable storage media for improved multidimensional data sequence searching. In particular, in some embodiments, a method is performed by a computer system having one or more processors and memory storing instructions for execution on the one or more processors. The method includes storing a first multidimensional probe sequence representation expressed in a first multidimensional orthogonal domain comprising an integer n number of dimensions. The first multidimensional probe sequence representation is characterized by a regional metadata comprising a power metric of all elements in a bounding region comprising a coordinate grid range in each of the n dimensions. The method further includes storing a first multidimensional target sequence representation expressed in the first multidimensional orthogonal domain comprising the integer n number of dimensions. The first multidimensional target sequence includes a plurality of potential multidimensional probe match sequences each characterized by the regional metadata comprising the power metric of all elements in the bounding region comprising the coordinate grid range in each of the n dimensions. The method further includes transforming the multidimensional probe sequence representation and the multidimensional target sequence representation into a second multidimensional orthogonal domain to produce a second multidimensional probe

sequence representation and a second multidimensional target sequence representation, respectively. The second multidimensional orthogonal domain is expressible using a basis set that is orthogonal to a basis set of the first multidimensional orthogonal domain. The method further includes encoding the second multidimensional target sequence representation with a first plurality of multidimensional modulation functions in the second multidimensional orthogonal domain, each of the first plurality of multidimensional modulation functions having an integer index to a multidimensional coordinate position corresponding to one of the potential probe match sequences, thereby producing a first plurality of encoded second multidimensional target sequence representations. The method further includes interfering the first plurality of encoded second multidimensional target sequence representations with the second multidimensional probe sequence representation to produce a first set of multidimensional interfered sequence representations. The method further includes encoding the second multidimensional target sequence representation with a second plurality of multidimensional modulation functions in the second multidimensional orthogonal domain, each of the second plurality of multidimensional modulation functions having a multidimensional coordinate position that is a negative counterpart of the multidimensional coordinate position of the first plurality of multidimensional modulation functions for the integer layer index corresponding to one of the potential probe match sequences, and thereby producing a second plurality of encoded second multidimensional target sequences. The method further includes interfering the second plurality of encoded second multidimensional target sequences with the second multidimensional probe sequence representation to produce a second set of multidimensional interfered sequence representations. The method further includes combining each of the first set of multidimensional interfered sequence representations with the complex conjugate of the corresponding counterpart in the second set of multidimensional interfered sequence representations to create a combined multidimensional interfered sequence representation and obtaining an inverse multidimensional transform result characterizing a respective integer index of the multidimensional coordinate position from the combined multidimensional interfered sequence representation. The method further includes determining whether the inverse transform result exceeds a predefined threshold. In accordance with a determination that the inverse transform result exceeds the predefined threshold, information is output indicating that the respective integer index of the multidimensional coordinate position represents a match between the multidimensional probe sequence representation and the corresponding one of the potential multidimensional probe match sequences. On the other hand, in accordance with a determination that the inverse multidimensional transform result does not exceed the predefined threshold, output of information corresponding to the respective integer index of the multidimensional coordinate position index is forgone.

In some embodiments, interfering the first plurality of encoded second multidimensional target sequence representations with the second multidimensional probe sequence representation comprises superimposing the first plurality of encoded second multidimensional target sequence representations and interfering the superimposed encoded second multidimensional target sequence representations with the second multidimensional probe sequence representation.

In some embodiments, interfering the first plurality of encoded second multidimensional target sequence representations with the second multidimensional probe sequence representation comprises performing a multidimensional matrix multiply operation between the plurality of encoded second multidimensional target sequence representation and a complex conjugate of the second multidimensional probe sequence representation.

In some embodiments, transforming the multidimensional probe sequence representation and the multidimensional target sequence representation into the second multidimensional orthogonal domain comprises applying a first unitary multidimensional orthogonal domain transform to the multidimensional probe sequence representation and the multidimensional target sequence representation, respectively. In some embodiments, the first unitary multidimensional orthogonal domain transform is a multidimensional Fourier transform.

In some embodiments, obtaining the inverse transform result characterizing a respective integer index of a multidimensional coordinate position includes applying a second unitary multidimensional orthogonal domain transform to the one or more multidimensional interfered sequence representation. The second unitary multidimensional orthogonal domain transform is an inverse of the first unitary multidimensional orthogonal domain transform. In such embodiments, obtaining the inverse transform result also includes selecting, as the inverse transform result, a result of the second unitary multidimensional orthogonal domain transform applied to the one or more multidimensional interfered sequence representation at a multidimensional coordinate position corresponding to the respective integer index.

In some embodiments, the first multidimensional probe sequence representation is a multidimensional matrix of real or complex numbers. In some embodiments, the first multidimensional probe sequence representation comprises a plurality separately searchable component symbols encoded as multidimensional sequential matrices of real or complex numbers.

In some embodiments the first plurality of multidimensional modulation functions having an integer index to a multidimensional coordinate position corresponding to one of the potential probe match sequences are each obtained from a layer delta function comprising a single non-zero data value at the multidimensional coordinate position corresponding to one of the potential probe match sequences.

In another aspect of the present disclosure, some implementations provide a non-volatile computer readable storage medium. The non-volatile computer readable storage medium includes one or more programs storing instructions that when executed by a computer system with one or more processors and memory the computer system to perform any of the methods provided herein.

In another aspect of the present disclosure, some implementations provide a computer system. The computer system includes one or more processors and memory. The memory stores one or more programs that include instructions that, when executed by the one or more processors, cause the computer system to perform any of the methods provided herein.

BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present disclosure, reference is now made to the following figures, wherein like reference numbers refer to similar items throughout the figures:

FIG. **1**A illustrates an example of a genome target used as an input and a probe used as a query in a sequence searching method, in accordance with some embodiments;

FIG. **1**B illustrates an example of a two-dimensional pixel image target used as an input and a two-dimensional pixel image probe used as a query in a two-dimensional data sequence searching method using layer delta functions, in accordance with some embodiments;

FIG. **1**C illustrates an example of a three-dimensional volumetric voxel image target used as an input and a three-dimensional volumetric voxel image probe used as a query in a three-dimensional data sequence searching method using layer delta functions, in accordance with some embodiments;

FIG. **2** illustrates a computer system that may be used to implement the search of the images target of FIGS. **1**A-**1**C;

FIG. **3**A illustrates the transformation of a probe sequence to a wavefunction for use in sequence searching, in accordance with some embodiments;

FIG. **3**B illustrates the transformation of an n-dimensional probe sequence to an n-dimensional wavefunction for use in n-dimensional data sequence searching, in accordance with some embodiments

FIG. **4**A illustrates the transformation of target sequence to superpositions of wavefunctions and layer encoding for use in sequence searching, in accordance with some embodiments;

FIG. **4**B illustrates the transformation of an n-dimensional target sequence to superpositions of n-dimensional wavefunctions and n-dimensional layer encoding using layer delta functions for use in n-dimensional data sequence searching, in accordance with some embodiments for n-dimensional data sequence searching with either continuous or discrete regions;

FIGS. **5**A-**5**B illustrate a flowchart for a method of sequence searching, using interference between the probe and target wavefunctions prepared by the methods of FIGS. **3**A and **4**A and the assessment of hits that are located, in accordance with some embodiments;

FIG. **5**C illustrates a flowchart for a method of sequence searching that is optimized for the special case in which the target sequence has a natural modulus, using interference between the probe and target wavefunctions prepared by the methods of FIGS. **3**A and **4**A and the assessment of hits that are located, in accordance with some embodiments;

FIG. **5**D illustrates a flowchart for a method of sequence searching that is optimized for the special case in which the target sequence has a natural modulus, using interference between the probe and target wavefunctions prepared by the methods of FIGS. **3**A and **4**A and the assessment of hits that are located, in accordance with some embodiments;

FIGS. **5**E-**5**F illustrate a flowchart for a method of n-dimensional data sequence searching with continuous regions, using n-dimensional interference between the n-dimensional probe wavefunctions prepared by the method of FIG. **3**B and n-dimensional target wavefunctions prepared by the continuous variant of method of FIG. **4**B and the continuous regions encoded with a continuous dimension using layer delta functions, and the assessment of hits that are located, in accordance with some embodiments;

FIG. **5**G illustrates a flowchart for a method of n-dimensional sequence searching with discrete regions that is optimized for n-dimensional object recognition in which the n-dimensional target sequence has a natural modulus, using n-dimensional interference between the n-dimensional probe wavefunctions prepared by the method of FIG. **3**B and n-dimensional target wavefunctions having discrete regions encoded with a discrete region method of FIG. **4**B using the discrete layer delta functions and the assessment of hits that are located, in accordance with some embodiments;

FIG. **5**H illustrates a flowchart for a method of n-dimensional data sequence searching with discrete regions that is optimized for n-dimensional object recognition in which the n-dimensional target sequence has a natural modulus, using n-dimensional interference between the n-dimensional probe wavefunctions prepared by the method of FIG. **3**B and n-dimensional target wavefunctions having discrete regions encoded with a discrete region method of FIG. **4**B using the discrete layer delta functions and the assessment of hits that are located, in accordance with some embodiments;

FIG. **5**I illustrates a method of matrix multiplication for computing a plurality of dot products between a row vector and a plurality of column vectors. In addition, FIG. **5**I illustrates a method for tensor product matrix calculation from two input matrices;

FIG. **5**J illustrates a flowchart of a method for computing a plurality of dot products between a row vector as a probe sequence and a plurality of column vectors that comprise the target sequence having a natural modulus, using interference between the probe and target wavefunctions prepared by the methods of FIGS. **3**A and **4**A and the assessment of dot products comprising hits that are located, in accordance with some embodiments;

FIG. **5**K illustrates a flowchart of a method for the interference of two superposition wavefunctions and reading of a plurality of orthogonal layer products between elements of each of the two superposition wavefunctions, in accordance with some embodiments;

FIG. **5**L illustrates a flowchart of a method for differentially encoding elements from two input sequences to two superposition wavefunctions combined with the method shown in FIG. **5**K to calculate the tensor product of the two input sequences, in accordance with some embodiments;

FIG. **5**M illustrates a flowchart of a method for the interference of two superposition wavefunctions and reading and summation of a plurality of orthogonal layer correlations between elements of each of the two superposition wavefunctions, with an output comprising a combined complex correlation between a plurality of orthogonal layers, in accordance with some embodiments;

FIG. **5**N illustrates a performance comparison **500** comprising a table of parameters and quantification of addition and multiplication operations versus other methods and the quantification of a quantum advantage of the current method as the number of addition and multiplication operations required by the current method **222** divided into the number of addition and multiplication operations for the equivalent matrix multiplication computation;

FIG. **6** illustrates a multi-resolution database and the segmentation of a target sequence into multiple sections, and the layer encoding applied to each of these sections for each track of the database, in accordance with some embodiments;

FIG. **7** illustrates the further segmentation into individual frames of the sections of an exemplary track zero of the multi-resolution database of FIG. **6**, in accordance with some embodiments;

FIG. **8** illustrates the preparation of target data, the multi-resolution database of FIG. **6**, and a non-encoded sequence track, in accordance with some embodiments;

FIG. **9** illustrates the selection of a track from the multi-resolution database, the searching for hits in the selected track, and the selection of a next track for searching, in accordance with some embodiments;

FIG. **10** illustrates the use of a single-layer track in the multi-resolution database to confirm a search hit located from the prior searching of one of several multiple-layer tracks in the multi-resolution database according to an exemplary embodiment of the present disclosure;

FIGS. **11**A-**11**D illustrate a flowchart for method of sequence searching, in accordance with some embodiments; and

FIG. **12** illustrates a block diagram of a server system for sequence searching, in accordance with some embodiments.

FIG. **13** depicts different tag data types as first sequence representations in the first orthogonal domain viewed as an x-y plot of data in the complex plane where x is the real axis and y is the imaginary axis according to an exemplary embodiment of the present disclosure. FIG. **13** also shows the rotation by a qubit complex phase angle applied to different tag data types as first sequence representations in the first orthogonal domain viewed as an x-y plot of data in the complex plane where x is the real axis and y is the imaginary axis according to an exemplary embodiment of the present disclosure.

FIG. **14** illustrates an instruction set architecture method comprising commands, actions and lists of combined tag, content and qubit elements, that may be expressed in the form of a structured data document according to an exemplary embodiment of the present disclosure.

FIG. **15** illustrates a method performed by a server computer that receives commands from a client, processes the commands in the manner of a programmable quantum computer using an interference process in a common step for different instructions and sends a response back to the client according to an exemplary embodiment of the present disclosure.

FIG. **16** illustrates the different action types for commands in the instruction set architecture together with their inputs, outputs and functional description according to an exemplary embodiment of the present disclosure.

FIG. **17** illustrates a method to solve the interactions of a plurality of superposition wavefunctions for a designated algorithm and appropriate coupling constraints according to an exemplary embodiment of the present disclosure.

FIG. **18** illustrates a method for progressive solving of superposition wavefunction mutual interactions by incorporating each rotational adjustment of qubit phase in the calculation of subsequent adjustments of qubit phase of other superposition wavefunctions within each pass through a list of items to be solved according to an exemplary embodiment of the present disclosure.

FIG. **19** illustrates a method for compiling instruction streams comprising commands of structured data to execute a quantum processing algorithm presented as input in the form of a directed flow wiring diagram of quantum interactions from an input state to an output state according to an exemplary embodiment of the present disclosure.

The exemplification set out herein illustrates particular embodiments, and such exemplification is not intended to be construed as limiting in any manner.

DETAILED DESCRIPTION

The present disclosure describes methods, computer systems and computer readable storage media for searching, e.g., a long target sequence of numbers for a match to a much shorter probe sequence of numbers. As one specific example, the methods described herein are used to search for a match of a two-dimensional image sequence of pixels (the multidimensional probe sequence) within a two-dimensional image (the multidimensional target sequence) as depicted in FIG. **1**B.

The ability to perform such searches quickly may help to develop new medical tests through which a patient's chromosome is searched for a disease-indicating gene variant. Such methods were also described in U.S. application Ser. No. 11/914,554, filed May 15, 2006 and entitled "Associative Memory and Data Searching System and Method," to which the present disclosure claims priority.

As another specific example, the methods described herein are used to search for a match of a three-dimensional volumetric image sequence of voxels (the multidimensional probe sequence) within a three-dimensional volumetric image (the multidimensional target sequence) as depicted in FIG. **1**C.

As a further specific example, the methods described herein are used to search for a match of an n-dimensional data sequence comprising n-dimensional data elements (the multidimensional probe sequence) within a multidimensional data image comprising multidimensional data elements (the multidimensional target sequence). The ability to perform such multidimensional searches quickly may help to develop new medical tests through which a patient's Magnetic Resonance Image and Computer Aided Tomography scans are analyzed for disease such as the presence of cell abnormalities and tumors. Real-time automated 3D object recognition using Computed Tomography scanner 3D voxel images is another example of the multidimensional data sequence searching method depicted in FIG. **1**C.

Such methods were also described in U.S. application Ser. No. 14/480,355, filed Sep. 14, 2014 and entitled "Associative Memory and Data Searching System and Method," to which the present disclosure claims priority.

Within the broad range of applications described in U.S. application Ser. No. 11/914,554 entitled "Associative Memory and Data Searching System and Method," there are important use cases that are subsets of the general method and which allow specific optimizations of the current method to arrive at the same mathematical result via an alternate but equivalent computational path. The most important use case is for all search keys to be at an aligned cursor instead of the non-aligned position capability in the current method, e.g., the only search hits possible are those where the offset position of the probe is zero relative to the layer encoded target wavefunction.

For example, one difficulty with searching a patient's chromosome is that there is no natural modulus to genes within a chromosome. That is to say, rather than a gene starting every ten, 100, or 1,000 base pairs, genes vary in length and, moreover, are mixed up with non-genetic material. Thus, a gene sequence may start and finish at any point within a chromosomal sequence. But much simpler searching applications exist in which there is a natural modulus, and to which important applications still exist. Consider the example of searching for alphanumeric characters: it is known that each character is contained in a separate or discrete region of pixels. The present disclosure further provides optimized search methods for this special case where discrete regions are searched.

The following description and the drawings illustrate specific embodiments sufficiently to enable those skilled in the art to practice the systems and methods described herein. Other embodiments may incorporate structural, logical, process and other changes. Examples merely typify possible variations. Individual components and functions are optional unless explicitly required, and the sequence of operations

may vary. Portions and features of some embodiments may be included in or substituted for those of others.

The elements that implement the various embodiments of the present system and method are described below, in some cases at an architectural level. Many elements may be configured using well-known structures. The functionality and processes herein are described in such a manner to enable one of ordinary skill in the art to implement the functionality and processes described herein.

The processing described below may be implemented in the form of special purpose hardware and/or in the form of software or firmware being run by a general-purpose or network or other specialized processor. Data handled in such processing or created as a result of such processing can be stored in any type of memory or other computer-readable medium as is conventional in the art. By way of example, such data may be stored in a temporary memory, such as in the random access memory of a given computer system or subsystem. In addition, or in the alternative, such data may be stored in longer-term storage devices, for example, magnetic disks, rewritable optical disks, and so on. For purposes of the disclosure herein, computer-readable media may comprise any form of data storage mechanism, including existing memory technologies as well as hardware or circuit representations of such structures and of such data.

It should also be understood that the techniques of the present system and method might be implemented using a variety of technologies. For example, the methods described herein may be implemented in software running on a programmable microprocessor, or implemented in hardware utilizing either a combination of microprocessors or other specially designed application specific integrated circuits, programmable logic devices, or various combinations thereof. In particular, the methods described herein may be implemented by a series of computer-executable instructions residing on a storage medium such as a carrier wave, disk drive, or other computer-readable medium. In addition to electronic and semiconductor hardware implementations, several important steps may be functionally implemented using non-electronic media and methodologies such as, for example, using optical components. Furthermore, analog media may provide a lower-cost alternative to digital media for storing or transmitting wavefunctions as described in the methods herein. As a result, it is feasible, in some embodiments, that the wavefunction may be processed starting from an analog storage medium, proceed via an analog processing chain, and then the pattern match results output as an analog signal without the need to pass through a step of conventional electronic digital representation, all according to the functional methods described herein.

One field of endeavor in which the present disclosure may be employed is in the field of two-dimensional (2D) image data sequence searching. Another field of endeavor in which the present disclosure may be employed is in the field of three-dimensional (3D) image data sequence searching. Whereas the two-dimensional image data comprises "picture elements" or pixels, the three-dimensional image data comprises "volume elements" or voxels. However, as will be described below, the disclosure is not limited to either two-dimensional image data sequence searching or three-dimensional data sequence searching and may be employed in other fields involving the analysis of multidimensional records.

A further field of endeavor in which the present disclosure may be employed is in the field of multidimensional data sequence searching, for example, where the analysis and interpretation of complex patterns are performed. In multidimensional data sequence searching, data comprises an image spanning an integer n number of dimensions and containing a multidimensional sequence of data elements. The ability to extend the data sequence searching method from one-dimensional sequence data such as DNA to corresponding methods for two-dimensional sequence searching in two-dimensional data such as pixel images, three-dimensional sequence searching in three-dimensional data such as voxel images and multidimensional sequence searching in multidimensional data will now be discussed. In particular, the differences of the multidimensional data sequence searching method will be described in relation to the previously disclosed one-dimensional data sequence searching methods.

In previous disclosures the quantum effect of superposition was created using a pair of quantum field wavefunctions ψR and ψS, referred to as an RS pair, formed by summation of target wavefunctions each differentially modulated according to an integer layer index. The output of an interference process between a probe or search wavefunction and a target ψR and ψS RS pair is according to the method a correlation between the input probe data and the target data at each layer in the wavefunction superposition of the RS pair.

In previous disclosures the input probe and target data was described as a sequence of real or complex numbers or symbols encoded as sequential vectors of real or complex numbers, each in the form of a one-dimensional position ordered vector containing searchable components. In addition, the method described was with the encoding modulation functions of the wavefunctions for the R superposition having an integer position index and the encoding modulation functions of the wavefunctions for the S superposition having a negative integer position index. Furthermore, the method specified that the output of the interference process is an integer position index.

The present disclosure describes how the same process may be extended to process multidimensional data such as pictures, video, 3D LIDAR (light detection and ranging), MRI 3D tissue scans, satellite data such as weather and terrain, geological and terrestrial spatial tomography. The number of dimensions may be any integer and therefore applications involving multidimensional data of several different orders, for example 1D, 2D and 3D data, or a variable number of dimensions and applications using data with a very high number of dimensions can benefit from the extended multidimensional data method presented herein.

In the method of preparation and interference of multidimensional probe data and multidimensional target data the unitary orthogonal transform and the unitary orthogonal inverse transform are replaced by their multidimensional counterparts: a unitary multidimensional orthogonal transform and a unitary multidimensional orthogonal inverse transform. In addition, the encoding of layers using an integer position index and a negative integer position index is replaced with an integer layer index to a multidimensional coordinate position and a multidimensional negative coordinate position. Also, the modulation functions used to encode the target wavefunctions to the R and S wavefunction superpositions are replaced by multidimensional modulation functions used to encode the multidimensional target wavefunctions to the R and S multidimensional wavefunction superpositions. Furthermore, the output of the multidimensional interference process is an integer layer index of the multidimensional coordinate position corresponding to a potential probe sequence match.

The multidimensional space of coordinate positions used to encode layers in an RS pair using multidimensional modulation functions may be generated from an integer layer index and the integer layer index may be recovered from a multidimensional coordinate position index. In this way, the multidimensional space of coordinate position indexes used to encode layers may be addressed as a linear one-dimensional space.

The generation of a space of multidimensional coordinate position indexes used to encode layers in an RS pair from a one-dimensional integer layer index may use any arbitrary pattern of multidimensional translations. Furthermore, according to the present disclosure the n-dimensional data sequence searching methods include a discrete region method and a continuous region method according to different variants of a Regional Metadata **34**. In a continuous region method a continuous dimension in the Regional Metadata defines in which of the n-dimensions the regions are continuous. In the two-dimensional target sequence depicted in FIG. **1**B, for example, a horizontal continuous dimension applies for a horizontal line of alphanumeric characters where the horizontal line is partitioned to a series of consecutive horizontal regions. In this case the integer layer index may apply to a horizontal line partition so that the first line can be layer 1, the second line layer 2 and so forth. In some embodiments, the integer layer index is encoded as a coordinate in the continuous dimension in an n-dimensional coordinate position, for example, with the horizontal continuous dimension the two-dimensional coordinate position (1,0) encodes integer layer index 1, the two-dimensional coordinate position (2,0) encodes integer layer index 2, and so on. In such a case the integer layer index may be recovered in the n-dimensional hit detection process from half the horizontal separation of pairs of hits from differentially modulated wavefunction superpositions.

In some embodiments, the integer layer index may be scaled selectively in each of the coordinate dimensions to generate multidimensional coordinate positions. In a two-dimensional image data sequence search application each integer layer index may designate a 2D image frame in a series of video frames. The transformation of the integer layer index to the multidimensional coordinate position may use the integer layer index as the horizontal coordinate x in a two-dimensional (x,y) coordinate position (e.g. integer layer index 0 is transformed to the 2D coordinate position (0,0), layer index 1 is transformed to the 2D coordinate position (1,0), and so forth). Similarly, in a three-dimensional image data sequence search application each integer layer index may designate a 3D image frame in a series of 3D frames. The integer layer index may be transformed to the x coordinate of a three-dimensional Cartesian coordinate position, for example layer 0 is transformed to the three-dimensional coordinate position (0,0,0), layer 1 is transformed to coordinate position (1,0,0), and so on.

The following description will be limited to examples of a sequence search of two-dimensional image data comprising pixels in a two-dimensional Cartesian coordinate grid of x and y coordinate pairs and a sequence search of three-dimensional volumetric image comprising voxels in a three dimensional Cartesian coordinate grid of x, y and z coordinates for the sake of clarity and brevity. Nonetheless, with respect to multidimensional image data sequence searching, it should be noted that while the following description focuses primarily on two-dimensional image data and three-dimensional image data, the disclosure is not limited to use with either two-dimensional image data or three-dimensional image data but can be utilized with data related to any other integer number of orthogonal dimensions in a multi-dimensional coordinate space. The multidimensional sequence data searching method is therefore generally applicable to a wide variety of high dimensionality pattern recognition applications.

The disclosure of a specific embodiment of a quantum computing multidimensional associative memory system and method is now presented below. This disclosure illustrates several aspects that may be provided by this system and method, which may include one or more of the following features: encoding duality using an orthogonal basis; multidimensional orthogonal transforms and domains; unitary multidimensional operations; quantum field multidimensional wavefunctions; qubit phase selection of quantum field multidimensional wavefunctions; superpositions of multiple quantum field multidimensional wavefunctions and interference with superpositions of multiple quantum field multidimensional wavefunctions.

As used herein, a "match" between a multidimensional target and a multidimensional probe may include a 100% match or a match with a lesser degree of similarity. As used herein, a "hit" means a match located from a search of a multidimensional target using a multidimensional probe.

The multidimensional associative memory feature of the present disclosure is illustrated, for example, in the embodiment below in which location and retrieval from memory is determined by the content of the input instead of by some other proxy or label, such as an address, alphabetic index, hash table or any other external attribute, reference or pointer.

The first of several features to reduce hit detection and retrieval effort described herein is the use of superpositions of multidimensional wavefunctions to process multiple layers of encoded multidimensional data in parallel. A two-dimensional data record, such as an image pixel sequence, is mapped as a plurality of superpositions each comprising multiple layers. A location and retrieval task involves searching the entire database for matches between an input probe or query sequence and the target data sequence. If a match is present, then its location is output. In the present disclosure this location corresponds, for example, to two coordinates: the index of the layer where the matching sequence is located; and the offset within the layer where the match is present. The wavefunction superposition comprising a database therefore may contain a two-dimensional search space in which any sequences in the database that match the probe may be located in this search space by a layer dimension index and an offset dimension index.

Another feature to help reduce hit detection and retrieval effort is the use of layer encoding modulation to create superpositions of wavefunctions such that the layer where a hit is located may be read by means of a layer profile.

In other words, one of the of the features of the present disclosure is the providing of an efficient method for first encoding, then later extracting, the layer index of any matches between the probe sequence and the target database. This implements an associative short-cut that avoids an exhaustive search of each layer in turn. Once the layer index has been identified, the full search space has been narrowed down to a single layer or page. This feature is described as measurement of the layer profile.

The layer profile allows all layers in a superposition to be processed as one layer, compressed into a single record, with the resulting output being a profile of the match correlation between probe and target and the match distance metric for each layer. This provides a faster and more efficient way to evaluate if any match is present between the probe (e.g.,

query or new experience) and the target (e.g., database or sum of past experiences), and if there is, to output the specific index of the layer in the superposition where that match is located.

As is described in greater detail below, the location of the match within the layer selected by the layer profile described above may be extracted by forming an interference viewpoint selected by the layer index. The interference viewpoint may be constructed by using one or more superpositions to interfere with the probe wavefunction. In the case where more two or more superpositions are used, as discussed further below, the process of constructing this viewpoint involves performing an extraction operation to combine the outputs of the first interference process. In this way, the first interference process may be used to generate intermediates (which may be temporarily stored in storage buffers during computations). These are common to both the layer profile process to read the layer index as mentioned above, and also the process to extract the interference viewpoint of that selected layer, which allows reading of a search coordinate equal to the offset position in that layer.

The layer profile process therefore comprises an interference between one or more superpositions and the probe wavefunction. The specific architecture associated with this process has the encoding of the layer index in each of the separate wavefunctions that are combined into superpositions.

It should be noted that the layer profile may be used to process the entire multidimensional target database for any and all matches to the multidimensional input probe: it is a de novo calculation of the multidimensional correlation between the multidimensional probe and the multidimensional target database for all reading frames and sequence alignments within the entire database sequence, such as two-dimensional pixel images, three-dimensional volumetric voxel images, and any positive integer number of dimensions that is generalized as a multidimensional image comprising an integer n dimensional volume containing various data elements. The output of this distance calculation exists as the output of a multidimensional interference process, and may be stored as a multidimensional intermediate wavefunction. For example, as illustrated in FIG. **5**E-**5**F (to be discussed in more detail below), the output of the multidimensional interference process is stored as multidimensional intermediate wavefunctions in storage buffers **125** and **143**. Here, two multidimensional orthogonal superpositions (designated "R" and "S" and provided at steps **111** and **129** of FIGS. **5**E-**5**F) are used to generate the multidimensional intermediate wavefunctions stored in multidimensional storage buffers **125** and **143**, and each multidimensional intermediate wavefunction may be represented as a multidimensional matrix of real or complex numbers.

The specific properties of the layer encoding are such that layer encoding is designed to be orthogonal to position encoding. Because wavefunction superpositions are employed that actually comprise multiple wavefunction layers in a single wavefunction, each stage of the layer profile process is the size of a single dimension of the search space (i.e., a single wavefunction). It is the particular architecture of orthogonal encoding and decoding operations in the present disclosure that allows this processing gain to be realized. In practice this processing gain means that, using a pair of multidimensional data matrixes comprising superpositions of multidimensional wavefunctions of complex numbers equivalent in size to a single dimension of the search space, it is possible to process a match correlation for an entire two-dimensional, three-dimensional and multidimensional search space for any integer n number of dimensions.

Additional aspects of the system and method of this disclosure will now be described in even greater detail for the particular embodiment of two-dimensional image pixel sequence searching. An example of a target two-dimensional image used as an input in image pixel sequence searching is depicted in FIG. **1**B, wherein one or more two-dimensional images are defined as regions **74**, **75**, and **76** of a pixel sequence on the two-dimensional image **71**. The pixels are identified as black and white or opaque and transparent. The two dimensional image **71** has a horizontal x-axis **72** and a vertical y-axis **73** and each pixel in the two-dimensional image is specified by a unique pair of x and y coordinates, for example p(x,y) may designate a pixel value at the coordinate position (x,y) in a two-dimensional Cartesian coordinate space. Target **71** has a bounding region **77** comprising a grid range of pixels in each dimension. In the example shown in FIG. **1**B the bounding region **77** may be defined as pairs of coordinates for opposite corners of the two-dimensional rectangular area contained within it. In general, a two-dimensional image pixel sequence to be searched is referred to as a "pixel target" below, and the pixel target is typically separated into a number of regions of defined length for purposes of processing.

An example of a probe two-dimensional image **81** is also depicted in FIG. **1**B. Probe **81**, in this example, is a two-dimensional image comprising sixty-four pixels arranged as eight vertically consecutive rows, each of eight horizontally consecutive pixels. Each pixel in probe **81** is specified by a unique pair of x and y coordinates according to their position relative to the horizontal x-axis **78** and the vertical y-axis **79**. Each pixel in probe **81** has a value p(x,y), and some p(x,y) values may designate that the pixel appears darker or opaque as shown for pixel **83**, while other p(x,y) values may designate that the pixel appears lighter or transparent as shown for pixel **84**. The value p(x,y) may represent any property or attribute of the pixel, for example density, temperature or velocity represented as a real or complex number. Probe **81** is characterized by a power metric of all the elements in a bounding region **82** comprising a coordinate grid range in each of the two dimensions, which in the example shown is eight pixels in each of the horizontal and vertical image dimensions. In one example, a transparent pixel **84** will have a zero value and a shaded pixel **83** will have a non-zero value. The power metric characteristic of probe **81** is the total of the pixel values squared in the bounding region **82**, which is eight pixel range in each dimension for a total of sixty-four pixels in the example shown.

According to the present disclosure the pixel target in FIG. **1**B may be separated into continuous or discrete regions where, for example, continuous regions may comprise multiple alphanumeric characters in a horizontal line and discrete regions may comprise individual alphanumeric characters. The continuous regions may be used in data sequence searching methods for continuous regions and the discrete regions may be used in data sequence searching methods for discrete regions. Also, according to the present disclosure, multiple pixel targets may be processed, for example, target image **71** may represent a single frame in series of video frames.

Additional aspects of the system and method of this disclosure will now be described in even greater detail for the particular embodiment of two-dimensional image pixel sequence searching. The target two-dimensional image used

as an input in image pixel sequence searching depicted in FIG. **1**B is encoded as a plurality of layers in a two-dimensional wavefunction superposition pair that will be described in greater detail below with reference to FIG. **4**B and a target wavefunction superposition preparation method **105**. The target wavefunction superposition preparation method **105** encodes each layer in the two-dimensional wavefunction superposition pair using a separate delta function for each layer, which is identified by an integer layer index. For one particular integer layer index, a layer delta function **14** comprises a single non-zero value at a two-dimensional coordinate that may be used to encode a layer comprising a line of alphanumeric characters where the line is subdivided into multiple regions as depicted by a bounding region **13** inside the target bounding region **77**. For another particular integer layer index, the layer delta function **18** comprises a single non-zero value at a two-dimensional coordinate within a bounding region **15** which is also inside the target bounding region **77**.

Additional aspects of the system and method of this disclosure will now also be described in even greater detail for the particular embodiment of three-dimensional volumetric image voxel sequence searching. An example of a three-dimensional volumetric target image used as an input in image voxel sequence searching is depicted in FIG. **1**C, wherein one or more three-dimensional volumetric images are defined as regions **86**, **87** and **88** of a voxel sequence in the three-dimensional target image **85**. The voxels are identified as black and white or opaque and transparent. The three-dimensional target image **85** has a horizontal x-axis **89**, a horizontal y-axis **90** and a vertical z-axis **91**. Each voxel in the three-dimensional image target **85** is specified by three spatial coordinates, for example v(x,y,z) may designate a voxel value at coordinate position (x,y,z) in a three-dimensional Cartesian coordinate space. Target **85** has a grid range of voxel elements in each dimension which is depicted as the volume inside a bounding region **92**. In the example shown in FIG. **1**C the bounding region **92** may be defined as pairs of coordinates for opposite corners of the three-dimensional cuboid volume contained within it. In general, a three-dimensional image voxel sequence to be searched is referred to as a "voxel target" below, and the voxel target is typically separated into a number of regions of defined length for purposes of processing.

According to the present disclosure the pixel target in FIG. **1**C may be separated into continuous or discrete regions where, for example, continuous regions may comprise multiple three-dimensional objects in a horizontal section and discrete regions may comprise individual objects. The continuous regions may be used in data sequence searching methods for continuous regions and the discrete regions may be used in data sequence searching methods for discrete regions. Also, according to the present disclosure, multiple voxel targets may be processed, for example, target image **85** may represent a single 3D scan in series of 3D scans.

An example of a three-dimensional probe **93** is also depicted in FIG. **1**C. Probe **93**, in this example, has vertically consecutive horizontal planes of voxels with each horizontal plane comprising sixty-four voxels arranged as eight co-planar consecutive rows. Each voxel in probe **93** is specified by a unique combination of x,y and z coordinates according to their position relative to the x-axis **94**, y-axis **95** and z-axis **96**. Each voxel in probe **93** has a value v(x,y,z) and some v(x,y,z) values may designate that the voxel appears darker or opaque as shown for voxel **98**, while other v(x,y,z) values may designate that the voxel appears lighter or more transparent as shown for voxel **99**. For the purpose of clarity, some of the transparent voxels are not shown in FIG. **1**C. The value v(x,y,z) may represent any property or attribute of the voxel, for example density, temperature or velocity represented as a real or complex number. Probe **93** is characterized by a power metric of all elements in a bounding region **97** comprising a coordinate grid range in each of the three dimensions, which in the example shown is eight voxels in each of the horizontal x and y dimensions and two voxels in the vertical z axis dimension. In one example, a transparent pixel **99** will have a zero value and a shaded pixel **98** will have a non-zero value. The power metric characteristic of probe **94** is the total of the voxel values squared over all the voxels in the bounding region **97**. For the example shown, the voxel range comprises eight voxels in each horizontal dimension and two voxel range in the vertical dimension, for a total of one hundred and twenty-eight voxels as shown with some transparent voxels omitted for clarity.

Additional aspects of the system and method of this disclosure will now be described in even greater detail for the particular embodiment of three-dimensional image voxel sequence searching. The target three-dimensional image used as an input in image voxel sequence searching depicted in FIG. **1**C is encoded as a plurality of layers in a three-dimensional wavefunction superposition pair that will be described in greater detail below with reference to FIG. **4**B and target wavefunction superposition preparation method **105**. The target wavefunction superposition preparation method **105** encodes each layer in the three-dimensional wavefunction superposition pair using a separate delta function for each layer, which is identified by an integer layer index. For one particular integer layer index, the layer delta function **28** comprises a single non-zero value at a three-dimensional coordinate within a bounding region **29** which is inside target bounding region **92**. For another particular integer layer index, the layer delta function **30** comprises a single non-zero value at a three-dimensional coordinate that may be used to encode a layer comprising a section of the target bounding region **92** and the section may be subdivided into multiple regions as depicted by the bounding region **32**.

A block diagram depicting the components of a computer system that may be used with the methods of the present disclosure is provided in FIG. **2**. Computer system **200** comprises an input device **202** for receiving data regarding a target to be searched and a probe (i.e., query) to use in conducting this search. The target may be, for example, two-dimensional sequence of pixels regarding target image **71**, three-dimensional sequence of voxels regarding target image **85**, or an n-dimensional sequence of data for any number of dimensions greater than one. The data may be received from data source **206**, which may be, for example, another computing device coupled to computer system **200** using, for example, the Internet or a local area network. Input device **202** may include multiple ports for receiving data and user input. The user input may be dynamically interactive with computer system **200** and be responsive to information provided to the user on a user display **205** or other user output device. Typically, user input is received from conventional input/output devices such as, for example, a mouse, trackball, keyboard, light pen, etc., but may also be received from other means such as voice or gesture recognition.

An output device **203** is coupled with a processor **201** for providing output through various possible interfaces. Output to a user on user display **205** may be provided on a video display such as a computer screen, but may also be provided

via printers or other means. Output may also be provided to other computer systems, devices or other programs for use therein. For example, computer system **200** may be a Web server used to provide data searching services as an application service provider. For example, search results may be presented on user display **205** as a map showing direct hits and close homologies for a given probe.

Input device **202** may be coupled with processor **201**, which may be, for example, a general-purpose computer processor or a specialized processor designed specifically for use with the present disclosure. Processor **201** may also comprise, for example, multiple processors working in parallel. Processor **201** may be coupled with memory **204** to permit storage of data used in the methods described herein and to store computer programs to be executed by processor **201**. Memory **204** may be implemented as several memories or may be partitioned for storage of different data types. Memory **204** may be used, for example, to provide numerous storage buffers for intermediates that are used in calculations for the searching methods described herein.

As previously mentioned, the terms multidimensional and n-dimensional are used interchangeably and n-dimensional is a specific case of multidimensional where n is the integer number of dimensions. The use of n-dimensional to describe method steps therefore references a specific case n of a general multidimensional method and all steps so described will have the same number n of dimensions as an input and output.

FIG. **3**A illustrates a method **10** for the transformation of a probe sequence (e.g., a DNA sequence) to a wavefunction for use in DNA or RNA sequence searching. The preparation of the DNA or RNA probe sequence by transformation to a wavefunction starts with an input probe or search sequence of DNA or RNA nucleotide bases **16**. This is typically in the format of a text file, known as a "Fasta" file, where each nucleotide base of DNA or RNA is encoded using its initial letter. For example, each line of a Fasta file may contain up to fifty ASCII text characters encoding the same number of DNA or RNA nucleotide bases. For the purpose of the present description the probe or search sequence will be the instance of what is being searched for in a much larger database, such as a chromosome or genome, or any part or combination of these.

The probe sequence may encode any sequence of interest from the very short through to the very long. This may be, for example, a short oligonucleotide fragment such as a small interfering "siRNA" of just 22 bases, or a palindrome sequence, a duplex forming structure or any other possibility. The present method is may be used with short probe lengths such as, for example, one to twelve bases in length. Alternatively, very long DNA and RNA sequences of several thousands of base pairs may be used as the probe sequence.

The input sequence **16** is processed by a unitary encoding of the nucleotide base sequence to a vector in an orthogonal basis **20**. As applied to DNA and RNA sequence searching, the present method achieves an orthogonal and unitary mapping between each of the four nucleotide bases (Adenine, Thymine, Guanine and Cytosine in the case of DNA; or Adenine, Uracil, Guanine and Cytosine in the case of RNA) and the four phase quadrants in a complex number plane (e.g., points around a unit circle). For example, this mapping in a graph in which the x-axis represents a real component and the y-axis an imaginary component may be represented as follows for a DNA sequence: A (1, 0), C (0, j), T (−1, 0), and G (0, −j). The output of step **20** is a data representation of a vector of complex numbers. In this specific embodiment, the length of this complex vector is equal to the length in number of bases of the input probe nucleotide sequence (e.g., seven bases in length).

The unitary encoding to orthogonal basis **20** may receive an additional input of a qubit phase **37** which may be a search parameter that is obtained for the input probe or search sequence **16**. The qubit phase **37** may be represented as a complex number that for example determines the selections in a circular space comprising the complex phase of individual layers in the wavefunction superposition RS pair. The qubit phase **37**, represented as a complex value, may be applied to each of the complex data values using complex multiplication to generate the output of step **20**.

The output of step **20** is then passed to the next step in the process, which is a unitary time frequency anti-aliasing **22**. This step is done to remove aliasing of position of the input sequence due to transform periodicity equal to the complex frame size (2N or 4,096 in the exemplary case discussed herein). Such aliasing may cause ambiguity in the orthogonal transform that would violate the unitary (or 1:1 mapping) principle of the present disclosure, which is desired for symmetry of information flow either backwards or forwards via a complex field equivalent. To preserve the unitary principle, this aliasing must be removed. This may be accomplished by appending to the output of step **20** an equal length complex vector comprising real and imaginary components each having all values of zero. The output of step **22** would therefore be a complex vector at least twice the length of the input probe nucleotide sequence. For this specific embodiment, an additional number of real and imaginary component with zero value are appended to the probe so that the output complex vector from step **22** has a length equal to twice the length of the frame size that will be used for encoding portions of the target sequence, which frame size is discussed further below. In the embodiment described below, the frame size is selected to be 2,048 bases, and so the output of step **22** is a complex vector with a length of 4,096 complex elements (where each complex element has a real and an imaginary component).

The output of step **22** is next processed by a unitary orthogonal transform **24**. This maps the input data representation of a complex vector to an output complex vector in an orthogonal dimension or domain. In addition to the input and output domains being mutually orthogonal, each element of the input complex vector is equivalently represented by a basis function that is orthogonal to each of the other basis functions in that input domain. Symmetrically, each element of the output complex vector is equivalently represented by a basis function that is orthogonal to each of the other basis functions in that output domain.

While many possibilities exist for the unitary orthogonal transform **24** that meet the criteria specified above, for the purposes of the present embodiment the specific instance of the unitary orthogonal transform will be implemented as a discrete Fourier transform. Transforms that may be used in other embodiments include, for example, transforms based on prime number theory, Hadamard transforms, Sine, Cosine, Hartley, as well as many potential wavelet transforms.

Following the unitary orthogonal transform **24**, the probe is now represented by a wavefunction in the form of a complex vector. For the specific example discussed below for a frame size of 2,048 bases, this complex vector contains 4,096 complex elements (where each complex element has a real and imaginary component). The probe is stored in storage buffer **26**, to retain the probe in memory prior to interference with other wavefunctions from the target (discussed below). For purposes of later reference, this stored

wavefunction is designated as φ. Storage buffer **26** may be implemented in memory **204** of computer system **200**, or in another storage device. The stored probe wavefunction may serve as a resource that can be reused if multiple interference processes need to use this wavefunction. Such storage of the probe often may be more efficient and economical compared to serially repeating the wavefunction preparation process. This intermediate may also be used immediately without storage.

FIG. **3**B illustrates a method **103** for multidimensional probe data sequence preparation to a multidimensional probe wavefunction for use in multidimensional data sequence searching where the number of dimensions n is greater than one. The transformation of an n-dimensional probe sequence for example, a two-dimensional pixel image **71**, a three-dimensional voxel image **85**, or multidimensional data in any integer n number of dimensions greater than one, as an n-dimensional sequence of data elements) to an n-dimensional wavefunction for use in n-dimensional data sequence searching. The preparation of the n-dimensional probe sequence by transformation to a n-dimensional wavefunction starts with an input n-dimensional probe or search sequence of data elements **17**. For example in the case where n is two, the input **17** is a two-dimensional image of two-dimensional picture elements or pixels. This is typically in the format of a binary file, such as the type known as a "bmp" file, where each pixel is encoded using its binary value. For example, each pixel of a bmp file may contain eight bits encoding different u-norm values from 0 representing black to 255 representing white. For the purpose of the present description the probe or search sequence will be the instance of what is being searched for in a much larger database, such as a collection of n-dimensional images such as 3D scans, or any part or combination of these.

The n-dimensional probe sequence may encode any sequence of interest from the very short through to the very long. This may be, for example, a short pixel fragment such as a texture patch, or a three-dimensional volumetric concave or convex shape, or any n-dimensional manifold. The present method may be used with short probe lengths such as, for example, sixty-four pixels in length. Alternatively, very long pixel and voxel sequences of several thousands or millions of pixels or voxels may be used as the n-dimensional image probe sequence.

The n-dimensional input sequence **17** is characterized by a bounding region containing pixels. For the purposes of the present method the n-dimensional input sequence **17** is further characterized by a power metric inside the bounding region. To provide this characterization, the n-dimensional input sequence **17** is passed to step **19** which calculates the total power of the n-dimensional input sequence **17** for the purpose of downstream pixel sequence searching. The power metric value output from step **19** is combined with the n-dimensional input sequence **17** by step **21**, a unitary encoding to n-dimensional orthogonal basis. Accordingly, step **21** receives the input pixel sequence within any bounding region **17**, which is also characterized by the power metric **19**. The power metric calculation step **19** is, for example, a sum of the squared pixel values in the bounding region across all n dimensions.

The unitary encoding to n-dimensional orthogonal basis **21** may receive an additional input of a qubit phase **37** which may be a search parameter that is obtained for the input probe or search sequence **17**. The qubit phase **37** may be represented as a complex number that determines the selections in a circular space comprising the complex phase of individual layers in the wavefunction superposition RS pair. The circular phase represented by qubit phase **37** may be used as an extra dimension in computations that is either a discrete or continuous variable. In the discrete variable case the circular space may be divided for example into four orthogonal spaces each comprised of a quadrant in the range plus or minus 45° with respect to each positive and negative real and imaginary axis of the unit circle in the complex plane. In the continuous variable case the circular space may for example represent direction in space or distance between two locations inside a loop.

The n-dimensional input sequence **17** is processed by the unitary encoding of the n-dimensional sequence of data elements to an n-dimensional matrix in an n-dimensional orthogonal basis **21**. As applied to two-dimensional image pixel sequence searching, three-dimensional voxel image sequence searching or multidimensional data sequence searching, the present method achieves an orthogonal and unitary mapping between the real or complex data values at each pixel, voxel or multidimensional data element, which may be for example a color, density, reflectiveness attribute, and a matrix of complex numbers, where the matrix has the same number of dimensions n equal to the number of dimensions in the n-dimensional input sequence **17**. The qubit phase **37**, represented as a complex value, may be applied to the real or complex data values at each pixel, voxel or multidimensional data element using complex multiplication to generate the output of step **21**.

A specific example of unitary encoding to n-dimensional orthogonal basis **21** will now be given where the input probe or search n-dimensional sequence of data elements are real numbers such as a color, density or reflectiveness attribute. The specific example is for a length of N1, . . . , Nn elements in each of the n dimensions. The unitary encoding output of step **21** is a matrix of complex numbers with the same number of dimensions n as the input **17** n-dimensional sequence of real valued data elements. For an input comprising an n-dimensional sequence of real elements in n dimensions having lengths (N1,N2, . . . , Nn) the output of step **21** is a matrix comprising complex data elements in n dimensions having lengths (N1,N2, . . . , Nn).

The output of step **21** is then passed to the next step in the process, which is an n-dimensional unitary time frequency anti-aliasing **23**. This step is done to remove aliasing of position of the input sequence due to multidimensional transform periodicity equal to the complex matrix size ((N1,N2, . . . , Nn) in the n-dimensional data searching exemplary case discussed herein). Such aliasing may cause ambiguity in the multidimensional orthogonal transform that would violate the unitary (or 1:1 mapping) principle of the present disclosure, which is desired for symmetry of information flow either backwards or forwards via a complex field equivalent. To preserve the unitary principle, this aliasing must be removed. This may be accomplished by appending to the output of step **21** in each dimension of the matrix, an equal sized n-dimensional complex matrix comprising real and imaginary components each having all values of zero. The output of step **23** would therefore be a n-dimensional complex matrix at least twice the size of the input n-dimensional probe data sequence in each dimension. For the n-dimensional data searching exemplary case discussed herein the input complex matrix size (N1,N2, . . . , Nn) will generate an output complex matrix size (2N1, 2N2, . . . , 2Nn). For this specific embodiment, an additional number of real and imaginary components with zero value are appended to the probe so that the output complex matrix from step **23** has a length in each dimension equal to twice the length of the matrix size that will be used for encoding

portions of the multidimensional target sequence, which matrix size is discussed further below. In the embodiment described below, the matrix size at step **17** is selected to be (N1,N2, . . . , Nn) real data values, which are converted by step **21** to the n-dimensional complex matrix size (N1,N2, . . . , Nn) and so the output of step **23** is an n-dimensional complex matrix with lengths of (2N1, 2N2, . . . , 2Nn) complex elements (where each complex element has a real and an imaginary component). As a specific example where the number of dimensions n is two, in the case of an input to step **17** of a two-dimensional image with 512 horizontal by 256 vertical real pixels, step **21** generates an output two-dimensional complex matrix of 512 rows by 256 columns. The result of the n-dimensional unitary time frequency anti-aliasing **23** in this case would then be a matrix of 1024×512 complex values.

The output of step **23** is next processed by a unitary n-dimensional orthogonal transform **25**. This maps the input n-dimensional data representation of an n-dimensional complex matrix to an output n-dimensional complex matrix in an n-dimensional orthogonal domain. In addition to the n-dimensional input and output domains being mutually orthogonal, each element of the input n-dimensional complex matrix is equivalently represented by n-dimensional basis functions that are orthogonal to each of the other n-dimensional basis functions in that input domain. Symmetrically, each element of the output n-dimensional complex matrix is equivalently represented by n-dimensional basis functions that are orthogonal to each of the other n-dimensional basis functions in that output domain.

While many possibilities exist for the unitary n-dimensional orthogonal transform **25** that meet the criteria specified above, for the purposes of the present embodiment the specific instance of the unitary n-dimensional orthogonal transform will be implemented as a discrete multidimensional Fourier transform for n dimensions. Transforms that may be used in other embodiments include, for example, multidimensional transforms based on prime number theory, Hadamard transforms, Sine, Cosine, Hartley, as well as many potential wavelet transforms.

In the some embodiments for multidimensional real data input the n-dimensional orthogonal transform is a real transformation commonly used for computing a real discrete Fourier transform (real DFT) using a complex discrete Fourier transform (DFT) of length equal to half the real input dimension.

Following the unitary n-dimensional orthogonal transform **25**, the n-dimensional probe is now represented by an n-dimensional wavefunction in the form of an n-dimensional complex matrix. For the specific example discussed below for a step **17** input matrix size with lengths in each dimension of (N1,N2, . . . , Nn) real data values, this n-dimensional complex matrix contains lengths in each dimension of (2N1, 2N2, . . . , 2Nn) complex elements (where each complex element has a real and imaginary component). The n-dimensional probe wavefunction is stored in an n-dimensional matrix storage buffer **27**, to retain the probe in memory prior to interference with other wavefunctions from the target (discussed below). For purposes of later reference, this stored wavefunction is designated as φM. The n-dimensional storage buffer **27** may be implemented in memory **204** of computer system **200**, or in another storage device. The stored probe wavefunction may serve as a resource that can be reused if multiple n-dimensional interference processes need to use this n-dimensional wavefunction. Such storage of the probe wavefunction often may be more efficient and economical compared to serially repeating the n-dimensional wavefunction preparation process. This intermediate may also be used immediately without storage.

FIG. **4**A illustrates the transformation of a target sequence (e.g., DNA genome **101**) to superpositions of wavefunctions and layer encoding for use in searching the target sequence with a probe. More specifically, FIG. **4**A schematically illustrates DNA/RNA target sequence preparation to a wavefunction form and the creation of orthogonal layer encoding superpositions.

Target sequence (sometimes referred to simply as a "target" herein) preparation begins with the input of a target sequence of DNA or RNA nucleotide bases **36**. This is typically in the format of a text file, for example a "Fasta" file, where each nucleotide base of DNA or RNA is encoded using the initial letter of the base name.

The next four steps (namely, unitary encoding nucleotide bases to orthogonal basis **38**; unitary time frequency anti-aliasing **40**; unitary orthogonal transform **42**; and storage buffer **44**) are similar in functionality to the correspondingly-named steps in method **10** of the probe preparation process described above.

The input sequence **36** is processed by a unitary encoding of nucleotide base sequence to a vector in an orthogonal basis **38**. As mentioned above, the input target sequence, which may be millions of bases in length, is typically broken down into smaller frames for processing using the present method. A frame size that is a power of two is typically selected for ease of use with the Fourier transform, but this is not a requirement of the method described herein. In the example discussed below, the target frame size (N) is selected to be 2,048 bases (N=2,048). Other sizes could also be used such as, for example, 4,096 or 8,192 bases. Hundreds and thousands or more of frames may be each processed as described below in order to encode the full length of a given target as superpositions of wavefunctions, for use in searching as described below.

Similarly as described above for probe preparation, this embodiment as applied to a DNA/RNA sequence search achieves an orthogonal and unitary mapping between each of the four nucleotide bases (Adenine, Thymine, Guanine and Cytosine in DNA; and Adenine, Uracil, Guanine and Cytosine in RNA) and the four phase quadrants in a complex number plane.

The unitary encoding to n-dimensional orthogonal basis **38** may receive an additional input of a qubit phase search parameter **37** that is obtained for the input target sequence **30**. The qubit phase **37** may be represented as a complex value that for example determines the selections in a circular space comprising the complex phase of individual layers in the wavefunction superposition RS pair. The qubit phase **37**, represented as a complex value, may be applied to the complex data values at each data element using complex multiplication to generate the output of step **38**.

The output of step **38** is a data representation of a vector of complex numbers. In the embodiment described here, the length of this complex vector is equal to the length in bases of the given input frame from the target nucleotide sequence (e.g., N=2,048 bases). The output of step **38** is then passed to the next step in the process, which is a unitary time frequency anti-aliasing **40**. Similarly as for the probe above, this step removes aliasing of position in one orthogonal domain with respect to the other orthogonal domain. Such aliasing may cause ambiguity in the orthogonal transform that violates the unitary (or **1**:**1** mapping) principle described above. In order to preserve the unitary principle this aliasing must therefore be removed. This may be accomplished by appending to the vector output of step **38** an equal length

complex vector comprising zero real and imaginary components. The output of step **40** would therefore be a complex vector at least twice the length of the input target nucleotide sequence. In the example of N=2,048 bases, the output of step **40** is a complex vector having 2N or 4,096 complex elements (where each element has a real and an imaginary component).

The output of step **40** is next processed by unitary orthogonal transform **42**. This maps the input data representation of a complex vector to an output complex vector in an orthogonal dimension or domain. In addition to the input and output domains being mutually orthogonal, each element of the input complex vector is equivalently represented by a basis function that is orthogonal to each of the other basis functions in that input domain. Symmetrically, each element of the output complex vector is equivalently represented by a basis function that is orthogonal to each of the other basis functions in that output domain.

While many possibilities exist for the unitary orthogonal transform **42** that meet the criteria specified above, for the purposes of the present embodiment the specific instance of the unitary orthogonal transform will be formalized as a discrete Fourier transform. As previously discussed, other transforms may also be used in other embodiments.

Following the unitary orthogonal transform **42**, the target is represented by a wavefunction represented in the form of a complex vector (e.g., with 2N or 4,096 complex elements in this specific example), which may be stored in storage buffer **44**. Storage buffer **44** stores this target wavefunction in memory for interference with other wavefunctions as part of encoding and preparing the target for later searching by the probe. Storage buffer may be implemented in memory **204** or in another storage device. The storage serves as a resource if multiple interference processes need to use this target wavefunction, and such storage may be more efficient compared to serially repeating the wavefunction generation process. For purposes of later reference, this intermediate wavefunction stored in storage buffer **44** is designated Ψ. This intermediate wavefunction may also be used immediately, for example in computations executed by processor **201**, without storage.

As mentioned more generally above, the target is encoded as one or more superpositions of wavefunctions, and these wavefunctions are encoded with a layer index. The number of layers deep of wavefunctions used in a given superposition may vary, as will be discussed further below. As an example, the superpositions may be formed from 64 wavefunctions each corresponding to a frame from the target, and each of these wavefunctions may be encoded with a layer index, for example, ranging from 0 to 63 (corresponding to a superposition depth of 64 layers). The relationship between layer index and target frames is later discussed in greater detail.

According to the present disclosure, the specific properties of the layer encoding are such that layer encoding is designed to be orthogonal to position encoding. This property is illustrated in the disclosure beginning with steps **46** and **48**. The layer index in this specific embodiment is encoded as a delta function position index **46**. According to this approach, the position of the delta function along a linear position axis will map unitarily to the layer number that is to be encoded in each one of many potential layer instances. For example, for layer zero, the delta function will be at a position zero.

Different embodiments may use any arbitrary unitary mapping between the position of the delta function along a linear position axis and the layer number that is to be encoded in each one of many potential layer instances. For example, the unitary mapping may be an integer scaling of the integer layer index by an integer multiplication factor. The multiplication factor becomes the separation between the position of the delta functions for consecutive encoded layers. This particular type of unitary mapping is referred to as an integer scaled encoding of the integer layer index to delta function position index.

The unitary mapping between the integer layer index and the position of the delta function along a linear position axis may include a bit-reversal of the integer layer index combined with a multiplication factor applied to the separation of the layer index encoded as delta function position index in step **46**. This particular type of unitary mapping is referred to as a scaled bit-reverse encoding of the integer layer index to delta function position index.

The delta function position index **46** is provided as an input to step **48**, which is a unitary orthogonal transform **48**. The delta function for each position corresponding to a layer is converted to a wavefunction using the unitary orthogonal transform. A discrete Fourier transform is used in the DNA sequence example here described, but other transforms may be used in other embodiments, as was mentioned above. The wavefunction in this example is a complex vector having 2N complex elements (e.g., 4,096 complex elements when using a frame size of 2,048 bases). This wavefunction may be stored in storage buffer **52**, and for purposes of later reference is designated as θ. The vector complex conjugate of the wavefunction θ is calculated in step **56**, and is designated as θ^*.

Layer encoding may be accomplished by the combining of the two modulation wavefunctions ϕ and θ^* with the target wavefunction χ. The encoding of the target with a wavefunction of a position index and a complex conjugate of that wavefunction assist with the later decoding of the position index when one or more matches are located in a search.

In the case, as here, where the unitary orthogonal transform **48** is the discrete Fourier transform, the two modulation wavefunctions may be represented as vectors equated to roots of unity: $\theta_k = e^{-j2\pi k\Delta/N}$ in the case of the vector in storage buffer **52**, and $\theta^*_k = e^{j2\pi k\Delta/N}$ in the case of the output of the vector complex conjugate step **56**.

Layer encoding of the target wavefunction χ may be accomplished by the following two operations: (i) a vector complex multiply step **58**, which uses the complex vectors stored in storage buffer **44** and storage buffer **52** and performs a vector complex multiply between the two vectors; and (ii) vector complex multiply step **64**, which uses the complex vectors stored in storage buffer **44** and the output vector of vector complex conjugate step **56** and performs a vector complex multiply between the two vectors.

The output from steps **58** and **64** will each be a complex vector of 2N complex elements in this embodiment (e.g., 4,096 elements for a frame length N of 2,048 bases). Each complex vector from steps **58** and **64** will correspond to a frame in the original target data, which has been layer encoded as two wavefunctions (from the use of an index and a conjugate of that index).

Each of the many vector outputs of vector complex multiply step **58** then pass to vector complex accumulate step **60**. Step **60** calculates a superposition by accumulating several wavefunctions output from step **58**. The number of wavefunctions accumulated will depend on the number (i.e., the layer depth) of wavefunctions that will be represented by each superposition output from step **60**. In other words, the

output of step **60** is a superposition calculated by additively combining multiple wavefunctions output from step **58**, where each wavefunction has a different layer encoding, to create a superposition wavefunction ψR in step **62** after all layers have been accumulated. In equation form, $\psi R=\Sigma\chi^{*}\theta$ for all layers that are used to form the superposition (each layer corresponds to a different wavefunction χ that is output from storage buffer **44** and to an index wavefunction that is output from storage buffer **52**).

In a corresponding manner, several output vectors from vector complex multiply step **64** pass to vector complex accumulate step **66**, which additively combines multiple wavefunctions output from step **64**, each with different layer encoding, to create a superposition wavefunction ψS in step **68** after all layers have been accumulated. In equation form, $\psi S=\Sigma\chi^{*}\theta^{*}$ for all layers that are used to form the superposition (each layer corresponds to a different wavefunction χ that is output from storage buffer **44** and to the conjugate of an index wavefunction that is output from step **56**).

The encoding method described above is repeated as necessary to process all additional frames of input data from a target or database sequence. As additional frames are processed, additional superposition wavefunctions ψR **62** and ψS **68** will be output. It should be noted that as the target is encoded, that superposition wavefunctions are generally provided as superposition R, S pairs (ψR, ψS). Each superposition ψR, ψS in the example described here has 2N (e.g., 4,096) complex elements.

Steps **60** and **66** are performed to provide potentially very many superposition R, S pairs for a given superposition depth (e.g., a depth of 64 wavefunctions). In step **80**, a database may be formed by repeating steps **60** and **66** for other superposition depths (e.g., for layer depths of 32, 16, 8, 4, 2, and 1 wavefunctions).

Also, as an option to the above approach, additional input data frames from the target or database sequence may be processed to extend the length of the output superpositions wavefunctions ψR **62** and ψS **68** instead of only varying the number of layers they contain. As a result, it is possible to create a wide range of output superposition wavefunctions ψR **62** and ψS **68** of various lengths and depths of encoded layers. This flexibility may be used to create multi-resolution wavefunction databases containing a plurality of tracks of varying lengths and depths of encoded layers. The advantage of such a multi-resolution format is that the search process has a range of different superposition depth options in the wavefunction database and may select the most efficient option as appropriate for any given search instance.

The many different possible ways of formatting and combining the output superpositions wavefunctions ψR **62** and ψS **68** so that they may be combined into an encoded target wavefunction database **80** for access during the actual searching process. One specific embodiment of database **80** is the multi-resolution database described below with respect to FIG. **6**. The multi-resolution database may comprise numerous superposition R, S pairs (ψR, ψS) grouped into various tracks with each track having a different layer depth. In this embodiment, typically, the maximum useful layer depth has been found to be 128 layers, with 64 layers being more typical. However, other embodiments of the method disclosed herein may possibly use greater layer depths.

FIG. **4**B illustrates a method **105** for the transformation of a multidimensional target data sequence to a pair of superpositions of multidimensional wavefunctions for use in multidimensional data sequence searching, where the number of dimensions n is greater than one. The transformation of an n-dimensional target data sequence (e.g., two-dimensional image pixel sequence **71**, three-dimensional voxel image sequence **85**, or as a general case for any integer number of dimensions, a multidimensional image containing data elements in a multidimensional coordinate grid) to superpositions of n-dimensional wavefunctions and n-dimensional layer encoding for use in searching the n-dimensional target sequence with an n-dimensional probe. More specifically, FIG. **4**B schematically illustrates multidimensional target sequence preparation to a multidimensional wavefunction form and the creation of multidimensional orthogonal layer encoding superpositions.

Multidimensional target sequence (sometimes referred to simply as a "target" herein) preparation where the number of dimensions is n begins with the input of an n-dimensional target sequence of data elements **31**, for example, two-dimensional pixels, three dimensional voxels or n-dimensional data elements. This is typically in the format of a binary file, for example, in the case of two-dimensional pixels, a "bmp" file, where each pixel is encoded using a single byte representing a value between 0 and 1.0 for the range of values 0 to 255 for each byte.

The n-dimensional target sequence input **31** is next broken into smaller regions by Partition Regions **33**. The output of step **33** comprises sections for processing subdivisions of the entire n-dimensional space. Each region output by step **33** is associated with a Regional Metadata **34** which comprises an integer layer index and a corresponding n-dimensional coordinate position that will be used to generate a Layer encoded as a Delta function at the n-dimensional coordinate position **47**. A region output by Partition Regions **33** also comprises the region's n-dimensional bounds and a power metric for the data within the region's n-dimensional bounds which are passed to both the Regional Metadata **34** and a unitary encoding to n-dimensional orthogonal basis **39**. Partition Regions **33** may sub-divide the input target n-dimensional sequence of data elements **31** according to either a continuous or discrete method and the Regional Metadata will specify which method has been used so that an appropriate sequence searching method may be applied subsequently.

According to the continuous variant of method **105** that prepares target wavefunction superpositions for continuous sequence searching a continuous dimension is defined in Regional Metadata **34**. For example, the two-dimensional target image sequence **71** in FIG. **1**B, may be sub-divided by Partition Regions **33** to four horizontal rectangular bounding regions where each rectangular bounding region contains a line of alphanumeric characters. Partition Regions **33** may use a continuous method of layer encoding where each integer layer index corresponds to a rectangular bounding region containing a line of alphanumeric characters, for example the region containing the first line of characters may be encoded using integer layer index 1, the region containing the second line of characters may be encoded using integer layer index 2, and so forth. The Regional Metadata **34** created by Partition Regions **33** will specify continuous method encoding with the horizontal dimension as the continuous dimension. Partition Regions **33** will sub-divide each rectangular bounding region containing a line of alphanumeric characters to a series of horizontally adjacent non-overlapping regions that are output to step **39**. Each consecutive adjacent non-overlapping region in the continuous dimension is also specified by a consecutive number which may be used to schedule consecutive RS pairs in a continuous n-dimensional sequence searching method. For example, each line may be divided into four horizontally

adjacent non-overlapping regions. In this case, each region output to step **39** will have the following components in Regional Metadata **34**: an integer layer index; a two-dimensional coordinate position corresponding to the integer layer index; two-dimensional region bounds; a power metric; a continuous method designation; a specified continuous dimension; and a consecutive number.

Continuing the example of FIG. **1**B for continuous of layer encoding where each integer layer index corresponds to a rectangular bounding region containing a line of alphanumeric characters, and each line is be divided into four horizontally adjacent non-overlapping regions, Partition Regions **33** executes a pattern that combines the first adjacent non-overlapping regions in each of the four lines as separately encoded layers to an output **70** comprising an R,S pair of wavefunction superpositions with associated Regional Metadata for each of the four separate layers encoded. The method **105** of target preparation will continue in this example with Partition Regions **33** generating the second adjacent non-overlapping regions in each of the four lines that are encoded as separate layers to an second instance of output **70** comprising an R,S pair of wavefunction superpositions with associated Regional Metadata for each of the four separate layers encoded. Similarly, the method **105** of target preparation will continue in this example with Partition Regions **33** generating the third adjacent non-overlapping regions in each of the four lines that are encoded as separate layers to a third instance of output **70**, and fourth adjacent non-overlapping regions in each of the four lines that are encoded as separate layers to a fourth instance of output **70**, comprising an R,S pair of wavefunction superpositions with associated Regional Metadata for each of the four separate layers encoded.

The output of method **105** comprises a Target R,S pair **70** and the Regional Metadata **34** generated by Partition Regions **33** for all layers encoded in the R and S wavefunction superpositions. Subsequently, an n-dimensional data sequence searching method will be selected according to the Regional Metadata associated with R,S pair. A continuous n-dimensional data sequence searching method will use the Regional Metadata associated with the Target R,S pair **70** to specify the continuous dimension and the consecutive number so that n-dimensional interference outputs may be combined from consecutive R·S pairs. The combination of consecutive n-dimensional interference outputs for the same probe wavefunction allows the probe to be detected equally well when a potential probe match is present in two consecutive regions output form partition Regions **33**, for example half in one region and half in the next region, as when it is contained entirely in a single region. In the example described above that divides each horizontal line to four adjacent non-overlapping regions, the continuous method allows each region to be arbitrarily generated, for example the line may be sub-divided into four regions of equal size.

According to the discrete variant of method **105** in FIG. **4**B that prepares target wavefunction superpositions for discrete sequence searching Regional Metadata **34** will specify that regions generated by Partition Regions **33** are discrete instead of continuous and in such cases the continuous dimension component of Regional Metadata **34** is not applicable. For example, the two-dimensional target image sequence **71** in FIG. **1**B, may be sub-divided by Partition Regions **33** to separate regions each containing a single character. In the case of the alphanumeric text in target image sequence **71** Partition Regions **33** may use a variety of different techniques for sub-dividing the bounding region **77** to smaller bounding regions each comprising a single alphanumeric character, for example the presence of vertical whitespace between horizontal lines of characters combined with horizontal whitespace between consecutive characters in a line may be used to delimit the regions so that each contains a discrete character. In this case, each region output to step **39** will have the following components in Regional Metadata **34**: an integer layer index; a two-dimensional coordinate position corresponding to the integer layer index; two-dimensional region bounds; a power metric; a discrete method designation. The two-dimensional coordinate position is unique for each integer layer index since it generates a unique layer delta function and n-dimensional layer modulation wavefunction for the data encoded at that layer. In the methods for searching discrete n-dimensional sequence data a hit is detected at an n-dimensional coordinate position and the layer encoded by that n-dimensional coordinate position has an integer layer index in the Regional Metadata that is associated with the RS wavefunction superpositions where the hit was detected.

According to the discrete variant of method **105** in FIG. **4**B that prepares target wavefunction superpositions for discrete sequence searching Regional Metadata **34** will specify that regions generated by Partition Regions **33** have an integer layer index and an n-dimensional coordinate position used to generate the n-dimensional layer modulation wavefunction. In some embodiments the n-dimensional coordinate position may be contained in the bounding region of the output of Partition Regions **33**, such as depicted in FIG. **1**B for two-dimensional coordinate position **18** and bounding region **15**. In such cases the n-dimensional coordinate position represents a physical location in the target data sequence that is functionally analogous to a physical memory address in a conventional computer system. In other embodiments the n-dimensional coordinate position represents a translated location in the target data sequence that is functionally analogous to a virtual memory address in a conventional computer system.

In embodiments where Partition Regions **33** generates n-dimensional coordinate positions that are translated locations of the integer layer index, any arbitrary pattern of translations may be used, for example, selected scaling of the integer to different coordinate dimensions may be used to translate the integer layer index to an n-dimensional coordinate position.

In the some embodiments where Partition Regions **33** generates n-dimensional coordinate positions that are translated locations, the translation of integer layer index to n-dimensional coordinate positions in Regional Metadata **34** generates a series of unique n-dimensional coordinate positions from the one-dimensional integer layer index by using a combination of a zigzag scan pattern that works by filling an n-dimensional coordinate grid diagonally across the n-dimensional space outwards from the origin, combined with a scaled bit-reversed coordinate transformation comprising bit-reversal of each coordinate of the n-dimensional zigzag generated position coordinates, followed by a scaling of the bit-reversed coordinates. The resulting spatial transformation maps the locally grouped n-dimensional zigzag scan coordinates to a set of n-dimensional coordinate positions at the centers of n-dimensional regions. The n-dimensional coordinate positions are maximally spatially separated and each region has a length in each dimension determined by the scale factor applied to that dimension's coordinate. Spatial separation of the n-dimensional coordinates used to encode distinct layers in the RS pair in this way provides an optimally distributed pattern to maintain layer

signal separation in the RS pair wavefunction superposition layer space while also providing a virtual address space in one-dimension addressable with the integer layer index.

The multidimensional target sequence input **31** is decomposed to sections by Partition Regions **33** by subdividing the target bounding region (e.g., bounding region **77** in FIG. **1**B, bounding region **92** in FIG. **1**C) to smaller separate bounding regions each comprising one section of output from step **33**. In some embodiments the Partition Regions **33** also calculates the power metric in the output section equal to the sum of the squared data values inside the n-dimensional bounding region of the section. The complete n-dimensional section comprising the partitioned section of the n-dimensional target sequence **31** defined by the section bounding region, the n-dimensional grid data values and their total power metric, is passed to step **39** for unitary encoding to n-dimensional orthogonal basis.

The next four steps (namely, unitary encoding to n-dimensional orthogonal basis **39**; unitary n-dimensional time frequency anti-aliasing **41**; unitary n-dimensional orthogonal transform **43**; and n-dimensional matrix storage buffer **45**) are similar in functionality to the correspondingly-named steps in method **103** of the multidimensional probe data preparation process described above.

The n-dimensional input sequence section from step **33** is processed by a unitary encoding of n-dimensional data sequence to an n-dimensional matrix in an n-dimensional orthogonal basis **39**. As mentioned above, the input multidimensional target sequence **31**, which may be millions of pixels or voxels in length, is broken down into smaller sections for processing using Partition Regions step **33** in the present method. A section size that is a power of two is typically selected for ease of use with the Fourier transform, but this is not a requirement of the method described herein. In the example discussed below, the target section size is selected to be 512×256 pixels. Other sizes could also be used such as, for example, 4096×2048 pixels. Hundreds and thousands or more of sections may be each processed as described below in order to encode the full length of a given target as superpositions of wavefunctions, for use in searching as described below.

The unitary encoding to n-dimensional orthogonal basis **39** may receive an additional input of a qubit phase search parameter **37** that is obtained for the input target sequence **31**. The qubit phase **37** may be represented as a complex value that determines the selections in a circular space comprising the complex phase of individual layers in the wavefunction superposition RS pair. The circular phase may be used as an extra dimension in computations that is either a discrete or continuous variable. In the discrete variable case the circular space may be divided for example into four orthogonal spaces each comprised of a quadrant in the range plus or minus 45° with respect to each positive and negative real and imaginary axis of the unit circle in the complex plane. In the continuous variable case the circular space may for example represent direction in space or distance between two locations inside a loop.

Similarly as described above for multidimensional image probe preparation, this embodiment as applied to a two-dimensional image pixel sequence searching, three-dimensional voxel image searching and any multidimensional data searching, achieves an orthogonal and unitary mapping between the real or complex data values at each pixel, voxel or multidimensional data element, which may be for example a color, density, reflectiveness attribute, and a matrix of complex numbers, where the matrix has number of dimensions equal to the number of dimensions n in the multidimensional input sequence **31**. The qubit phase **37**, represented as a complex value, may be applied to the real or complex data values at each pixel, voxel or multidimensional volumetric element using complex multiplication to generate the output of step **39**.

The output of step **39** is a data representation of an n-dimensional matrix of complex numbers. In the embodiment described here, the size of this complex matrix is equal to the dimensions of data value elements of the given input section from the target n-dimensional data sequence (e.g., lengths in each dimensions of (N1, . . . , Nn) for the general preferred embodiment in n dimensions and 512×256 pixels for the specific two-dimensional example described herein). The output of step **39** is then passed to the next step in the process, which is an n-dimensional unitary time frequency anti-aliasing **41**. Similarly as for the probe above, this step removes aliasing of position in one multidimensional orthogonal domain with respect to the other multidimensional orthogonal domain. Such aliasing may causes ambiguity in the multidimensional orthogonal transform that violates the unitary (or **1**:**1** mapping) principle described above. In order to preserve the unitary principle this aliasing must therefore be removed. This may be accomplished by appending to the matrix output of step **39** in each dimension of the matrix an equal sized n-dimensional complex matrix comprising zero real and imaginary components. The output of step **41** would therefore be an n-dimensional complex matrix at least twice the size of the input n-dimensional target data sequence in each dimension. For the n-dimensional data searching exemplary case discussed herein the input complex matrix size (N1,N2, . . . , Nn) will generate an output complex matrix with lengths in each dimension of (2N1, 2N2, . . . , 2Nn). In the example of 512×256 pixels, the output of step **41** is a complex matrix having 1024×512 complex elements (where each element has a real and an imaginary component). As a specific example where the number of dimensions n is two, in the case of an input to step **31** of a two-dimensional image with 512 horizontal by 256 vertical real pixels, step **39** generates an output two-dimensional complex matrix of 256 rows by 512 columns. The result of the n-dimensional unitary time frequency anti-aliasing **41** in this case would then be a matrix of 512×1024 complex values.

The output of step **41** is next processed by unitary n-dimensional orthogonal transform **43**. This maps the input n-dimensional data representation of a complex matrix to an output n-dimensional complex matrix in an n-dimensional orthogonal domain. In addition to the n-dimensional input and output domains being mutually orthogonal, each element of the input n-dimensional complex matrix is equivalently represented by n-dimensional basis functions that are orthogonal to each of the other n-dimensional basis functions in that input domain. Symmetrically, each element of the output n-dimensional complex matrix is equivalently represented by n-dimensional basis functions that are orthogonal to each of the other n-dimensional basis functions in that output domain.

While many possibilities exist for the unitary n-dimensional orthogonal transform **43** that meet the criteria specified above, for the purposes of the present embodiment the specific instance of the unitary n-dimensional orthogonal transform will be formalized as a discrete multidimensional Fourier transform for n dimensions. As previously discussed, other multidimensional transforms may also be used in other embodiments.

In the some embodiments for multidimensional real data input the n-dimensional orthogonal transform is a real

transformation commonly used for computing a real discrete Fourier transform (real DFT) using a complex discrete Fourier transform (DFT) of length equal to half the real input dimension.

Following the unitary n-dimensional orthogonal transform **43**, the target is represented by an n-dimensional wavefunction represented in the form of a n-dimensional complex matrix (e.g., with lengths in each dimension of (2N1, 2N2, . . . 2Nn) complex elements in this specific example), which may be stored in n-dimensional matrix storage buffer **45**. The n-dimensional storage buffer **45** stores this n-dimensional target wavefunction in memory for n-dimensional interference with other n-dimensional wavefunctions as part of encoding and preparing the target for later n-dimensional searching by the n-dimensional probe. The n-dimensional storage buffer **45** may be implemented in memory **204** or in another storage device. The storage serves as a resource if multiple n-dimensional interference processes need to use this n-dimensional target wavefunction, and such storage may be more efficient compared to serially repeating the n-dimensional wavefunction generation process. For purposes of later reference, this n-dimensional intermediate wavefunction stored in n-dimensional storage buffer **45** is designated χM. This n-dimensional intermediate wavefunction may also be used immediately, for example in computations executed by processor **201**, without storage.

As mentioned more generally above, the multidimensional target is encoded as one or more superpositions of multidimensional wavefunctions, and these multidimensional wavefunctions are encoded with an integer layer index. The number of layers deep of wavefunctions used in a given superposition may vary, as will be discussed further below. As an example, the multidimensional superpositions may be formed from 64 multidimensional wavefunctions each corresponding to a section from the multidimensional target, and each of these multidimensional wavefunctions may be encoded with an integer layer index, for example, ranging from 0 to 63 (corresponding to a superposition depth of 64 layers). The relationship between layer index and target sections is later discussed in greater detail.

According to the present disclosure, the specific properties of the layer encoding are such that layer encoding is designed to be orthogonal to position encoding. This property is illustrated in the disclosure beginning with steps **34** and **47**. The integer layer index in this specific embodiment and an n-dimensional coordinate position are output by Partition Regions **33** to Regional Metadata **34** for a corresponding region output from Partition Regions step **33** to step **39**. As described above, any arbitrary pattern of translations such as selected scaling of the integer to different coordinate dimensions may be used to translate the integer layer index to an n-dimensional coordinate position. The output of step **33** is an n-dimensional coordinate position that is encoded as a layer delta function in step **47**. The layer delta function is defined as a null valued n-dimensional range with a single coordinate grid position having a non-zero value. According to this approach, the n-dimensional coordinate position of the delta function will map unitarily to the integer layer index number that is to be encoded in each one of many potential layer instances. For example, for layer zero, the delta function may be at n n-dimensional coordinate position with zero coordinates (0, . . . , 0) known as the origin in any multidimensional coordinate space. In Regional Metadata **34** each integer layer index corresponds to a different n-dimensional coordinate position which will generate a unique layer delta function for the integer layer index in step **47**.

Step **47** uses the n-dimensional coordinate position from Regional Metadata **34** to generate a layer delta function comprising a null valued range with a single non-zero real value at the n-dimensional grid position specified by the n-dimensional coordinates.

The output of step **47** is an n-dimensional complex matrix with a single n-dimensional coordinate position having a non-zero value. In some embodiments of n-dimensional data sequence searching method the complex number value at the single n-dimensional coordinate position having a non-zero value will have a real component of unity and an imaginary component value of zero.

The real valued delta function at the n-dimensional coordinate position **47** is provided as an n-dimension complex matrix comprising a single non-zero value as input to step **49**, which is a unitary n-dimensional orthogonal transform **49**. The delta function for each n-dimensional coordinate position corresponding to an integer layer index is converted to an n-dimensional wavefunction using the unitary n-dimensional orthogonal transform **49**. A discrete multidimensional Fourier transform of n dimensions is used as the unitary n-dimensional orthogonal transform in the multidimensional data sequence searching example here described, but other transforms may be used in other embodiments, as was mentioned above. The n-dimensional wavefunction in this example is a complex matrix having lengths in each dimension of (2N1, 2N2, . . . , 2Nn) complex elements (e.g., 1024×512 complex elements when using a two-dimensional image size of 512×256 pixels). This n-dimensional wavefunction may be stored in n-dimensional matrix storage buffer **53**, and for purposes of later reference is designated as θM. The n-dimensional matrix complex conjugate of the n-dimensional wavefunction θM is calculated in step **57**, and is designated as θM^*.

Layer encoding may be accomplished by the combining of the two n-dimensional modulation wavefunctions ϕM and θM^* with the n-dimensional target wavefunction χM. The encoding of the target with a n-dimensional wavefunction of a n-dimensional coordinate position and a complex conjugate of that n-dimensional wavefunction assist with the later decoding of the n-dimensional coordinate position and the corresponding integer layer index when one or more matches are located in a search.

In the case, as here, where the n-dimensional unitary orthogonal transform **49** is the discrete n-dimensional Fourier transform, the two n-dimensional modulation wavefunctions may be represented as matrices equated to complex roots of unity: such as $\theta m_{(p,q)}=e^{-j2\pi p\Delta x/Nx}.e^{-j2\pi p\Delta x/Ny}=e^{-j2\pi(p\Delta x/Nx+q\Delta y/Ny)}$ in the case of n the number of dimension is two and the two-dimensional coordinates of the delta function are $(\Delta x, \Delta y)$, represented as a matrix in n-dimensional matrix storage buffer **53**, and $\theta M^*_{(p,q)}=e^{-j2\pi p\Delta x/Nx}.e^{-j2\pi p\Delta x/Ny}=e^{-j2\pi(p\Delta x/Nx+q\Delta y/Ny)}$ in the case of the corresponding output of the n-dimensional matrix complex conjugate step **57**, where each is defined over the two-dimensional grid $0\le p\le N1-1, \ldots, 0\le q\le Nn-1$.

The n-dimensional wavefunction in storage buffer **53** and the n-dimensional wavefunction in storage buffer **65** can be represented for the general n-dimensional case where the n-dimensional coordinates of the delta function are $g((\Delta 1, \ldots, \Delta n)$ defined over the n-dimensional grid $0\le\Delta 1\le N1-1, \ldots, 0\le\Delta n\le Nn-1$, where N1 is the number of grid points in dimension 1 and Nn is the number of grid points in dimension n. In other words, $g(\Delta 1, \ldots, \Delta n)$ is a real data value of unity at the n-dimensional coordinates of $(\Delta 1, \ldots, \Delta n)$, where the elision marks between $\Delta 1$ and Δn denote coordinates in all the dimension between the

first and last dimension. Accordingly, the n-dimensional wavefunction in storage buffer **53** is represented as $\theta M_{(k1,k2,\ldots,kn)}=e^{-j2\pi k1\Delta 1/N1}.e^{-j2\pi k2\Delta 2/N2}, \ldots, e^{-j2\pi knkn\Delta ni/Nn}=e^{-j2\pi(k1\Delta 1/N1+k2\Delta 2/N2+\ldots,kn\Delta n\Delta/N)}$, and the n-dimensional wavefunction output by n-dimensional matrix complex conjugate step **57** is represented as $\theta M^*_{(k1,k2,\ldots,kn)}=e^{j2\pi k1\Delta 1/N1}.e^{j2\pi k2\Delta 2/N2}, \ldots, e^{j2\pi knkn\Delta ni/Nn}=e^{j2\pi(k1\Delta 1/N1+k2\Delta 2/N2+\ldots,kn\Delta n\Delta/N)}$ where each is defined over the n-dimensional grid $0\leq k1\leq N1-1, \ldots, 0\leq kn\leq Nn-1$.

Layer encoding of the target wavefunction χM may be accomplished by the following two operations: (i) an n-dimensional matrix complex multiply step **59**, which uses the n-dimension complex matrices stored in storage buffer **45** and storage buffer **53** and performs an n-dimensional matrix complex multiply between the two n-dimensional matrices; and (ii) n-dimensional matrix complex multiply step **65**, which uses the n-dimensional complex matrices stored in storage buffer **45** and the output n-dimensional matrix of n-dimensional matrix complex conjugate step **57** and performs an n-dimensional matrix complex multiply between the two n-dimensional matrices. The n-dimensional matrix complex multiply operation is defined herein as a complex multiplication between values at corresponding coordinates.

The output from steps **59** and **65** will each be an n-dimensional complex matrix with lengths in each dimension of (2N1, 2N2, . . . , 2Nn) complex elements in the preferred n-dimensional data searching embodiment, or for the specific example where the number of dimensions is equal to two (e.g., 2N1×2N2=1024×512 complex elements, for a two-dimensional image dimensions of 512×256 pixel data elements where N1=512 and N2=256). Each n-dimensional complex matrix from steps **59** and **65** will correspond to a region output by Partition Regions **33** comprising a subdivided section of input n-dimensional sequence **31** in the original target data, which has been layer encoded as two n-dimensional wavefunctions (from the use of an integer layer index corresponding to a positive n-dimensional coordinate position and a conjugate position of that integer layer index corresponding to an n-dimensional coordinate position with indices that are negated values of each of the coordinates of the positive position). The n-dimensional coordinate position designated as positive is arbitrary an may include negate or wrap-around coordinate indices in each of the n dimensions. In other words, for the same integer layer index, the n-dimensional coordinate position designated as positive has a conjugate position that is a reflection in the n-dimensional origin of the n-dimensional coordinate position designated as positive.

Each of the many n-dimensional matrix outputs of n-dimensional matrix complex multiply step **59** then pass to n-dimensional matrix complex accumulate step **61**. Step **61** calculates an n-dimensional superposition by accumulating several n-dimensional wavefunctions output from step **59**. The number of n-dimensional wavefunctions accumulated will depend on the number (i.e., the layer depth) of n-dimensional wavefunctions that will be represented by each n-dimensional wavefunction superposition output from step **61**. In other words, the output of step **61** is a target n-dimensional wavefunction superposition R **63**, calculated by additively combining multiple n-dimensional wavefunctions output from step **59**, where each wavefunction has a different layer encoding, to create a target n-dimensional wavefunction superposition ψRM in step **63** after all layers have been accumulated. In equation form, $\psi RM=\Sigma\chi M^{\bullet}\theta M$ for all layers that are used to form the n-dimensional superposition (each layer corresponds to a different n-dimensional wavefunction χM that is output from n-dimensional storage buffer **45** and to the R differential n-dimensional layer modulation wavefunction θM (generated from an n-dimensional layer delta function coordinate position) that is output from multidimensional storage buffer **53**).

In a corresponding manner, several output n-dimensional matrices from n-dimensional matrix complex multiply step **65** pass to n-dimensional matrix complex accumulate step **67**, which additively combines multiple n-dimensional wavefunctions output from step **65**, each with different layer encoding, to create an n-dimensional wavefunction superposition ψSM in step **69** after all layers have been accumulated. In equation form, $\psi SM=\Sigma\chi M^{\bullet}\theta M^*$ for all layers that are used to form the n-dimensional superposition (each layer corresponds to a different n-dimensional wavefunction χM that is output from n-dimensional storage buffer **45** and to the S differential n-dimensional layer modulation wavefunction **57** designated θM* that is the complex conjugate of the n-dimensional coordinate position layer modulation wavefunction θM that is output from step **53**).

The encoding method described above is repeated as necessary to process all additional subdivisions of multidimensional input data from a target or database sequence. As additional groups of multiple input sections comprising n-dimensional bounded regions inside an input target n-dimensional sequence **31** are processed, additional superposition wavefunctions ψRM **63** and ψSM **69** will be output. It should be noted that as the target is encoded, that superposition wavefunctions are generally provided as n-dimensional superposition R, S pairs (ψRM, ψSM) and their associated Regional Metadata **34** for all layers. Each superposition ψRM, ψSM in the example described here may be represented as an n-dimensional matrix with lengths in each dimension of (2N1, 2N2, . . . , 2Nn) complex elements. The Regional Metadata for each R,S pair **70** includes the Regional Metadata **34** for each layer that has been encoded in the R,S pair. According to the present disclosure each integer layer index is unique for each layer contained in R,S pair **70** and is uniquely associated with an n-dimensional coordinate position, n-dimensional bounds of the region and a power metric of the data elements contained inside the bounded n-dimensional region. The Regional Metadata **34** associated with each R,S pair **70** includes applicable parameters that were employed by Partition Regions **33** to generate the layer encoded sub-divided section of the input target or database n-dimensional sequence of data elements. In the case that Partition Regions **33** generated consecutive regions by stepping in a selected continuous dimension the Regional Metadata **34** comprises the specification of continuous encoding, the continuous dimension for the step and a consecutive number for scheduling consecutive R,S pairs in the continuous method. For example, the two-dimensional target image **71** of FIG. **1**B may be sub-divided by Partition Regions **33** to four horizontal rectangular bounding regions where each rectangular bounding region contains a line of alphanumeric characters. The Regional Metadata **34** created by Partition Regions **33** will specify continuous method encoding with the horizontal dimension as the continuous dimension. Partition Regions **33** will sub-divide the rectangular bounding region containing a line of alphanumeric characters to horizontally adjacent non-overlapping regions that are output to step **39** each with a consecutive number in the Regional Metadata **34**. A subsequent n-dimensional data sequence searching method will be selected according to the Regional Metadata associated with the R,S pair. A continuous n-dimensional data sequence searching method will use the Regional Metadata associated with the R,S pair to specify the continuous dimension so that n-dimensional

interference outputs may be combined from consecutive R. S pairs having consecutive numbers in the associated Regional Metadata. The combination of consecutive n-dimensional interference outputs for the same probe wavefunction allows the probe to be detected equally well when a potential probe match is present in two consecutive regions output form partition Regions **33**, for example half in one region and half in the next region, compared to when it is contained entirely in a single region.

Steps **61** and **67** are performed to provide potentially very many n-dimensional superposition R, S pairs **70** for a given superposition depth (e.g., a depth of 64 wavefunctions). Also, as an option to the above approach, additional input data sections as regions from the target or database sequence may be processed to extend the length of the output n-dimensional superpositions wavefunctions ψRM **63** and ψSM **69** instead of only varying the number of layers they contain. As a result, it is possible to create a wide range of output n-dimensional superposition wavefunctions ψRM **63** and ψSM **69** of various lengths and depths of encoded layers.

FIGS. **5**A-**5**B illustrate a sequence searching method **100**, using interference between the probe and target wavefunctions prepared by the methods of FIGS. **3**A and **4**A and including the assessment of hits that are located. More specifically, method **100** involves the search for and detection of any pattern matches in the target wavefunction superpositions and the reading of the position offset plus the layer index of any detected matches.

According to the present disclosure, the pattern matching search process performs an interference between the probe wavefunction and one or more superpositions of target wavefunctions that have been modulated according to their particular layer index, as was discussed above. In the diagram of method **100**, two separate superposition wavefunctions are input: target wavefunction superposition R **110** and target wavefunction superposition S **128**. Superpositions **110** and **128** may be a superposition R, S pair (ψR, ψS) to be processed next from among the many such R, S pairs that correspond to a given track (the preparation of which was discussed above).

Method **100** illustrates the parallel searching of many such corresponding superposition R, S pairs. The given track has been selected, for example, from the multi-resolution database for performing a search. Criteria that may be used for selecting the track to use are discussed later below.

The input probe wavefunction φ **146** is prepared by method **10** of FIG. **3**A. The complex conjugate of φ is calculated in step **148** and is designated φ*. The complex conjugate φ* is then interfered separately with each of ψR and ψS by performing a vector complex multiply operation in each of steps **112** and **130**, respectively. The output from each step **112** and **120** is a complex vector having 2N or 4,096 complex elements in the specific example of an input frame length of N or 2,048 bases.

The outputs of vector complex multiply steps **112** and **130** are next each processed with a unitary orthogonal inverse transform in steps **114** and **132**, respectively. These inverse transforms are the inverse of the transform that was previously used to prepare the probe and target wavefunctions above. While many possibilities exist for the unitary orthogonal inverse transforms **114** and **132** as mentioned earlier, for the purposes of the present embodiment the unitary orthogonal transform will be an inverse discrete Fourier transform.

As will be next discussed, overlapping portions selected from the incoming inverse transform vector outputs from steps **114** and **132** will be separately added to overlap buffer **116** and overlap buffer **134** respectively, to provide vector complex addition results at each of steps **118** and **136**. These results are designated for later reference as VCA_R and VCA_S, respectively. After the processing described below, each of the vector results VCA_R and VCA_S will be a complex vector of length N or 2,048 complex elements for the exemplary input target frame size of N or 2,048 bases. It should be noted that a benefit of the overlapping of successive frames is to create VCA_R and VCA_S data that is independent of the probe sequence occurrences in the target sequence, even when the probe sequence is contained in two separate but consecutive wavefunction frames input to steps **110** and **128**.

More specifically, each of the complex vectors output from step **114** (each such complex vector is designated as "frame h" for purposes of discussion) is processed to overlap a portion of a given frame h with a portion of its predecessor frame using an overlap buffer **116** and a vector complex add operation at step **118**. For discussion purposes, each predecessor and successor complex vector output from step **114** is designated as "frame h−1" and "frame h+1", respectively. For the example of an initial input target frame length of N or 2,048 bases, each frame h contains 2N or 4,096 elements. For purposes of discussion, the elements for frame h may be considered as divided into a first half of elements 0 to N−1, and a second half of elements N to 2N−1 (each half contains N elements). For each frame output from step **114**, the vector elements 0 to N−1 from the first half of the frame are stored in an overlap buffer at step **116**. Overlap buffers **116** and **134** may be any suitable form of memory or other storage.

Next, in step **118** each of vector elements N to 2N−1 in the second half of a new frame h output from step **114** is added using vector complex add step **118** to the corresponding elements 0 to N−1 in the first half of the preceding output frame h−1, which was previously stored in overlap buffer **116**. In other words, element 0 is added to element N, element 1 is added to element N+1, and so forth, and element N−1 is added to element 2N−1. The output from step **118** (VCA_R) is a set of N complex correlations 0 to N−1 for each incoming frame h. The total number of vectors VCA_R output from step **118** is one for each frame input in steps **110** and **128**. During loop processing, the total number of vectors VCA_R output from step **118** is equal to the number of wavefunction superposition frames input in steps **110** and **128**.

The first half (elements 0 to N−1) of the same new output frame h from the inverse transform **114** is stored in overlap buffer **116** until it is brought together with the second half (elements N to 2N−1) of the next output frame h+1 from inverse transform **114**. As a result of this repeated overlap and vector addition processing, vector complex add **118** outputs a vector VCA_R with a length that is half that of the output vectors from inverse transform **114**.

In a similar manner to that described immediately above, each of many complex vector outputs from inverse transform step **132** is processed to overlap each frame with its predecessor and successor frames using overlap buffer **134** and vector complex add **136**. As a result of this repeated overlap and vector addition process, vector complex add **136** outputs a vector VCA_S with a length that is half that of the output vectors from inverse transform **132**. The output from step **136** (VCA_S) is a set of N complex correlations 0 to N−1 for each incoming frame h.

The next steps involve processing the output vector VCA_R of vector complex add step **118** via a complex inflation step **120**, and processing the output vector VCA_S of vector complex add step **136** via a complex inflation step **138**. For

each complex inflation step **120** and **138**, each element of the vector VCA_R and VCA_S, respectively, is raised to the third power using vector complex multiplication. In other embodiments, it should be noted that other powers or other manners of inflation may used. Complex inflation may assist in separating background noise in vectors VCA_R and VCA_S from peak signals therein that correspond to potential matches between the probe and the target database.

The outputs from complex inflation steps **120** and **138** may be processed using optional phase filtering steps **121** and **139**. Typically, this is designed to select direct correlates and exclude phase rotated correlates since these are of greater interest in representing a pattern match. The outputs from this phase filtering are provided to modulus and normalization steps **122** and **140**, respectively. Each of these steps **121** and **139** involves phase scaling by multiplying each complex element of each of the inflated vectors by a phase scale factor. The phase scale factor may be determined by the phase of the complex element. For example, the phase scale factor F may be cosine (arctangent (imaginary/real)) for the interval where real is positive. In the case where the real is less than or equal to zero, the phase scale factor F is zero. The phase filtered complex value output is (F*real, F*imag). The phase scaling may eliminate correlations that are out of phase. For example, there may be a match that is not a direct phase match to a probe, but instead is a rotation of that phase (e.g., the same sequence rotated by 90°).

The two parallel processing paths continue with the providing of the output of complex inflation step **120** to modulus and normalization step **122**, and providing of the output of complex inflation step **138** to modulus and normalization step **140**. Each of the modulus and normalization steps **122** and **140** involves calculation of the real modulus of each input complex element followed by a multiplication by a normalization scale factor that is constant for all elements in the input vector to step **122** or **140**. The normalization scale factor may be determined using the theoretical maximum correlation value corresponding to the vector outputs from steps **114** and **132**. The normalization scale factor may be calculated as the desired output peak height divided by a total input energy calculated according to the energy in the encoding of a single base, squared to reflect 100% correlation, times the number of bases raised to the third power to account for inflation, times the gain of the unitary orthogonal transform, times the gain of the unitary orthogonal inverse transform. In the case where each base is encoded by a complex value of magnitude **1,000**, and using discrete Fourier transforms of 4,096 complex values, for a normalized peak of 687 the scale factor is $1e^{26}$. One aspect of the foregoing is that the normalization scale factor may be proportional to the reciprocal of the cube of the count of effective bases in the probe.

The output of each step **122** and **140** is a vector of scalar numbers (each vector is designated as RV_R and RVs, respectively) representing the scaled magnitude of the corresponding input complex vector elements. For example, for an exemplary input frame length of N or 2,048 bases, RV_R and RVs contain N or 2,048 real numbers. Each of the vectors RV_R and RVs is a correlation at a particular base position between the probe sequence and the target sequence. It should be noted that this correlation may be measured at every base position in the target sequence.

The outputs of steps **122** and **140** may be temporarily saved to storage buffers **124** and **142**, respectively. These storage buffers may be used to hold these normalized results of the superposition R and S wavefunction interferences with the probe wavefunction so that the results may be further analyzed to determine if any hits or significant similarities are present. In this specific context, a "hit" refers to a peak value in the normalized data output from step **122** or **140**.

In typical use, there is a stream of real vectors RV_R and RVs, with corresponding RV_R and RVs vectors stored in storage buffer **124** or **142** for each superposition wavefunction R, S pair (ψR, ψS) that entered method **100** at steps **110** and **128**. The number of RV_R, RVs vector pairs will depend, for example, on the resolution track from the multi-resolution database that was selected for searching. The real vectors typically abut one another directly (i.e., this process is like the assembly of one long real vector, with a length depending, for example, the number of layers being processed in parallel). The RV_R and RVs vectors are examined for peaks that correspond to high correlations between the probe and the target.

Hit Detection Process

In an attempt to capture all potentially significant matches between the probe and target sequences, the hit detection process described below contains several steps. These steps are designed to identify all potential candidates, then to perform a more rigorous qualification of these candidates to attempt to eliminate false detection events. The hit detection process outputs qualified events comprising a pair of hits, one each from the parallel R and S wavefunction interference processing paths described above. However, as part of the hit detection process, the real vectors RV_R and RVs are initially each screened for hits independently. This approach improves the likelihood that all significant matches between the probe and target sequences will be found since two detection opportunities are thus created for each potential match.

The hit detection process begins with a threshold calculation at step **126**, which is applied to data from real vector RV_R obtained from storage buffer **124** and a threshold calculation at step **144**, which is applied to data from real vector RVs obtained from storage buffer **142**. In each case, the normalized magnitude data are analyzed using statistical methods to calculate the mean and standard deviation of each set of data. In threshold calculation **126** two threshold values are calculated for the R wavefunction interference data, and in threshold calculation **144** two thresholds are calculated for the S wavefunction interference data.

Each pair of thresholds consists of a primary threshold and a secondary threshold, with the primary threshold being greater than the corresponding secondary threshold. These thresholds are calculated by adding the mean value to a multiple of the standard deviation value of the respective data set. For example, the multiples may be sixteen times in the case of the primary threshold and six times in the case of the secondary threshold. For the primary threshold a larger multiple is applied to the standard deviation value, and for the secondary threshold a smaller multiple is applied to the standard deviation value. In addition, a tertiary threshold is calculated for each data set using the combined means from steps **126** and **144** plus a multiple of the combined standard deviations from steps **126** and **144**. This multiple is, for example, sixteen times the combined standard deviations. The tertiary threshold is used to test the combined maximum values in a pair of R, S hits. The use of the primary, secondary and tertiary thresholds is described below.

The next steps in the detection process are the determination of a maximum in a window in a step **150**, which is

applied to the data from the R wavefunction interference in storage buffer **124**, and a maximum in a window in a step **170**, which is applied to the data from the S wavefunction interference in storage buffer **142**. The window length is in each case equal to the effective length of the probe sequence, where the effective length refers to the number of coding bases in the probe sequence, (but not counting gaps or bases with a zero input weighting—for example, a value γ_k of zero as discussed below). Using a window size determined in this manner attempts to account for a hypothetical worst case scenario in which the probe sequence repeats itself as two adjacent sequences in the overall target sequence.

The window is advanced across the full length of each real vector RV_R and RVs. The window is advanced, after each selection of a maximum value, by a number of index positions in vector RV_R or RVs equal to the total window length so that no given sequence position of the real vector is examined in step **150** more than once. A maximum value is selected from within the window for each vector RV_R or RVs at each position in its advance along the corresponding vector.

The output of maximum in window step **150** is the index position (selected from positions 0 to N−1 in this example) and magnitude of the maximum value within the real vector RV_R from storage buffer **124**. A value and index data pair is output for each advance of the window along RV_R. Similarly, the output of maximum in window step **170** is the index position and magnitude of the maximum value within the real vector RVs from storage buffer **142**. A value and index data pair is output for each advance of the window along RVs.

The compare maximum to thresholds step **152** first tests whether the maximum value output from maximum in window step **150** meets or exceeds the primary threshold value derived in threshold calculation step **126**. If it does, the result is considered to be a "Potential R Hit" **154** that is processed further as described below. Each Potential R Hit corresponds to a position and magnitude in RV_R.

Similarly, compare maximum to thresholds step **172** first tests whether the maximum value output from maximum in window step **170** meets or exceeds the primary threshold value derived in threshold calculation **144**. If it does, the result is a "Potential S Hit" **174** that is processed further as described below. Each Potential S Hit corresponds to a position and magnitude in RVs.

Alternatively, if the maximum value output from maximum in window step **150** is below the primary threshold value derived in threshold calculation step **126**, and the maximum value output from maximum in window step **170** is below the primary threshold value derived in threshold calculation step **144**, processing will continue by returning to maximum in window step **150** and maximum in window step **170**, advancing the windows along each vector RV_R or RVs to generate new maxima for testing in the compare maximum to thresholds step **152** and compare maximum to thresholds step **172** as described above.

If a Potential R Hit **154** exists, a search is performed in step **176** for the existence of a corresponding or dual hit in an attempt to make an "R, S Hit Pair". It should be noted that if a real correlation between the probe and target exists, then two significant corresponding correlations that were calculated from the superposition R and S wavefunctions (as was discussed earlier above) should also exist. An R, S Hit Pair would correspond to these two correlations and consist of an R hit and an S hit. The method described below intelligently searches for a corresponding or dual R hit or S hit in a range in which such a hit is expected to be if it corresponds to a truly significant correlation between the probe and the target. In general, the positional separation will be two times the layer number that the signal causing the correlation peaks to appear was earlier encoded into. For example, as will become more clear from the discussion below, if the hit is in a portion of the target sequence that was encoded into layer 3, the positions of the R hit and S hit will be separated by six positions in the position index used for the elements of real vectors RV_R or RVs.

Step **176** first tests, based upon the position of the Potential R Hit within the window used in step **150**, if the entire range of possible dual S hit positions (i.e., corresponding to the Potential R Hit) was already included within the window used for the obtaining the S maximum from vector RVs in step **170**. The entire range of possible dual S hit positions is defined relative to the Potential R Hit position by the largest separation delta generated by layer encoding. For any target track **902** (discussed further below) being processed, it is known how many layers have been combined (i.e., the depth of the superposition wavefunctions used). According to the exemplary multi-resolution database **900** (illustrated in FIG. **6**), the largest separation delta generated by layer encoding is equal to twice the number of layers in that superposition.

If the entire range of possible dual S hit positions was included in the window of step **170**, then the S maximum from step **170** may be used as the possible dual S hit corresponding to the Potential R Hit **154**. Alternatively, if the entire range of possible dual S hit positions was not included in the window of step **170**, then a search of the vector RVs is done to select the maximum value from within the entire range of potential dual S hit positions in RVs for use as the S maximum.

Next, the separation delta of the R and S maxima is calculated. The "separation delta" is equal to the position of the R maximum minus the position of the S maximum. The separation delta is then tested to see if it meets the following three conditions: (i) the separation delta is non-negative, (ii) the separation delta is even-numbered, and (iii) the separation delta is within the range of position separation generated by layer encoding, which as mentioned above, is equal to twice the number of layers in that superposition.

If the separation delta satisfies all of the above three conditions, the magnitude of the S maximum is compared against the secondary threshold previously calculated in threshold calculation step **144**. If the magnitude of the S maximum meets or exceeds the secondary threshold, then the sum of the magnitudes of the R and S maxima is compared against the tertiary threshold previously calculated from steps **126** and **144**. If the sum of the magnitudes of the R and S maxima is greater than or equal to the tertiary threshold, then the R and S maxima are passed as a potential dual R hit and S hit pair (or simply "R, S hit pair") to step **178**.

Step **178** calculates the offset and separation of the potential dual R, S hit pair. Specifically, step **178** finds the position midpoint between the R and S maxima and records this as the offset of the R, S hit pair. In equation form: $\text{position midpoint}=(\text{position}_{R\ maximum}+\text{position}_{S\ maximum})/2$. The separation delta of the two maxima is also determined. The separation delta is equal to the position of the R maximum minus the position of the S maximum.

Similarly as was described above for the existence of a Potential R Hit, if a Potential S Hit **174** exists, a search is performed in step **156** for the existence of a dual hit in an attempt to make an R, S Hit Pair. Step **156** first tests, based upon the position of the Potential S Hit within the window

used in step **170**, if the entire range of possible dual R hit positions (i.e., corresponding to the Potential S Hit) was already included within the window used for the obtaining the R maximum from vector RV_R in step **150**. The entire range of possible dual R hit positions is defined relative to the Potential S Hit position by the largest separation delta generated by layer encoding.

If the entire range of possible dual R hit positions was included in the window of step **150**, then the R maximum from step **150** may be used as the possible dual R hit corresponding to the Potential S Hit **174**. Alternatively, if the entire range of possible dual R hit positions was not included in the window of step **150**, then a search of the vector RV_R is done to select the maximum value from within the entire range of potential dual R hit positions in RV_R for use as the R maximum.

Next, following step **156**, is step **158**, in which the separation delta of the R and S maxima is calculated. The "separation delta" is equal to the position of the R maximum minus the position of the S maximum. The separation delta is then tested to see if it meets the following three conditions: (i) the separation delta is non-negative, (ii) the separation delta is even-numbered, and (iii) the separation delta is within the range of position separation generated by layer encoding.

If the separation delta satisfies all of the above three conditions, the magnitude of the R maximum is compared against the secondary threshold previously calculated in threshold calculation step **126**. If the magnitude of the R maximum meets or exceeds the secondary threshold, then the sum of the magnitudes of the R and S maxima is compared against the tertiary threshold previously calculated from steps **126** and **144**. If the sum of the magnitudes of the R and S maxima is greater than or equal to the tertiary threshold, then the R and S maxima are passed as a potential dual R,S hit pair to step **178**.

Step **158** calculates the offset and separation of the potential dual R,S hit pair. Specifically, step **158** finds the position midpoint between the R and S maxima and records this as the offset of the R,S hit pair. In equation form: $\text{position midpoint}=(\text{position}_{R\ maximum}+\text{positions}_{maximum})/2$. The separation delta of the two maxima is also determined. The separation delta is equal to the position of the R maximum minus the position of the S maximum.

As mentioned above, the processing steps **150**, **152**, **154**, **156** and **158** provide a substantially parallel path to the processing steps **170**, **172**, **174**, **176** and **178**, assisting in ensuring that the R and S normalized interference data are initially each screened for hits independently. This approach attempts to ensure that all significant matches between the probe and target sequences will be found since two detection opportunities are created for each match between the probe and the target. The results from these two parallel paths converge in step **180**, which qualifies the R,S Hit Pairs. More specifically, step **180** eliminates R,S Hit Pair duplicates from these results (e.g., a duplicate R,S Hit Pair may have arisen as a result of using the parallel detection processes discussed earlier).

As will be discussed in more detail below, when originally preparing the target for searching, the target may be divided into a number of sections (e.g., 128 sections, designated as $S_0 \ldots S_{127}$) with each section encoded to one of the layer indices used for forming the layer-encoded superposition wavefunctions. In step **182**, data for each of the R,S Hit Pairs detected in the preceding steps is used to generate the following data for each R,S Hit Pair: (i) a position offset, and (ii) a target section number. The target section number may be used with the position offset to identify the location in the target sequence that matches the probe sequence for the given R,S Hit Pair. The position offset is calculated from the offset of the R,S Hit Pair, which is equal to the position midpoint between the R and S maxima plus an offset of the frame size N for each frame of the target input in steps **110** and **128**.

The superposition layer index is obtained by halving the layer separation delta from steps **158** and **178** (equal to the position of the R maximum minus the position of the S maximum). In the exemplary multi-resolution database **900** shown in FIG. **6**, the number of sections in each layer varies from two for track 0 (64 deep) to sixty-four for track 5 (2 deep). Accordingly, the superposition layer index is multiplied by the number of sections per layer in the track **902** being processed to obtain the target section number. For example, in the case of a layer separation delta for a R,S dual hit of 16, the superposition layer index of this hit pair would be 8. If track 0 (64 deep) of database **900** in FIG. **6** were being processed, the target section number would be obtained by multiplying 8 by 2, where two is the number of sections in one layer.

The separation delta of the two maxima (as mentioned above, the separation delta is equal to the position of the R maximum minus the position of the S maximum), is divided by two to give the superposition layer number in which the hit was detected. The superposition layer number is then multiplied by the number of target sections originally included in each layer in order to yield the target section number (e.g., a number selected from the range 0 to 127 in the specific example discussed herein) in which the matching sequence is located.

FIGS. **5**E-**5**F illustrate a multidimensional continuous sequence searching method **107**, using interference between the n-dimensional probe wavefunctions prepared by the method **103** of FIG. **3**B and RS pairs of n-dimensional target wavefunctions prepared by the continuous variant of method **105** for target preparation depicted in FIG. **4**B and including the assessment of hits that are located. More specifically, method **107** involves the search for and detection of any pattern matches in the target wavefunction superpositions and the reading of the n-dimensional coordinate position offset plus the layer index of any detected matches. An essential feature of method **107** is that the potential probe sequence match contained in the target sequence may be represented by two separate RS pairs of target wavefunction superpositions. This situation occurs when the potential probe match sequence contained in the target sequence is represented in two separate regions output from Partition Regions **33**. To handle this situation the method **107** uses overlap buffers **117** and **135** to combine the interference results from two separate RS pairs. As a result, the correlation result for the entire potential probe match sequence is obtained from the overlap and add of the correlation result of one portion of the potential probe match sequence generated from one RS pair with the correlation result of other portion of the potential probe match sequence generated from another RS pair.

The method **107** illustrated in FIGS. **5**E-**5**F is especially suited to multidimensional object recognition applications it situations such as a conveyor belt of luggage moving past a Computerized Tomography (CT) scanner or a street view of a self-driving vehicle. In these cases a continuous time sequence of target data corresponding to successive CT images or street views. An object recognition task involves searching for a probe or query data sequence in the continuous time sequence of target data and outputting the

identity and location of any match. The continuous time sequence of target data is processed in sections and as a result the potential probe match contained in the target data may be processed to separate RS wavefunction superposition pairs. In method **107** the overlap and add of the interference results from separate RS pairs generates a continuous correlation result for any probe sequence regardless of where the target sequence data was partitioned for processing by Partition Regions **33** in FIG. **4**B operating according to the continuous dimension layer encoding method.

According to the present disclosure, the pattern matching search process performs an interference between the n-dimensional probe wavefunction **147** and one or more n-dimensional superpositions of n-dimensional target wavefunctions that have been n-dimensionally modulated according to their particular layer index, as was discussed above. In the diagram of method **107**, two separate n-dimensional superposition wavefunctions are input: n-dimensional target wavefunction superposition R matrix **111** and target wavefunction superposition S matrix **129**. Superpositions **111** and **129** may be a n-dimensional superposition R, S pair (ψRM, ψSM) to be processed next from among the many such n-dimensional R, S pairs that have been scheduled for having consecutive numbers in the Regional Metadata associated with each R·S pair.

Method **107** illustrates the parallel searching of many such corresponding n-dimensional superposition R, S pairs. In some embodiments a separate program instance processes a unique combination of probe n-dimensional wavefunction **147** and the R, S pair comprising target n-dimensional wavefunction superposition R **111** and target n-dimensional wavefunction superposition S **129**. Many program instances running concurrently provide a way to efficiently search a very large database comprised of R,S pairs for a large number of search probe queries.

The input n-dimensional probe wavefunction φM **147** is prepared by method **103** of FIG. **3**B. The complex conjugate of φM is calculated in step **149** and is designated φM*. The complex conjugate φM* is then interfered separately with each of ψRM and ψSM by performing an n-dimensional matrix complex multiply operation in each of steps **113** and **131**, respectively. The output from each step **113** and **131** is an n-dimensional complex matrix.

The outputs of n-dimensional matrix complex multiply steps **113** and **131** are next each processed with a unitary n-dimensional orthogonal inverse transform in steps **115** and **133**, respectively. These n-dimensional inverse transforms are the inverse of the n-dimensional transform that was previously used to prepare the n-dimensional probe and target wavefunctions above. While many possibilities exist for the unitary n-dimensional orthogonal inverse transforms **115** and **133** as mentioned earlier, for the purposes of the present embodiment the unitary n-dimensional orthogonal transform will be an inverse discrete multidimensional Fourier transform of n dimensions.

As will be next discussed, overlapping portions selected from the incoming n-dimensional inverse transform matrix outputs from steps **115** and **133** will be separately added to overlap buffer **117** and overlap buffer **135** respectively, to provide n-dimensional matrix complex addition results at each of steps **119** and **137**. These results are designated for later reference as $VCAM_R$ and $VCAM_S$, respectively. After the processing described below, each of the n-dimensional matrix results $VCAM_R$ and $VCAM_S$ will be an n-dimensional complex matrix of length (N1,N2, . . . , Nn) complex elements for the exemplary input n-dimensional target sequence size of (N1,N2, . . . , Nn) data elements. It should be noted that a benefit of the overlapping of successive n-dimensional transforms is to create $VCAM_R$ and $VCAM_S$ data that is independent of the n-dimensional probe sequence occurrences in the target n-dimensional sequence, even when the n-dimensional probe sequence is contained in two separate but consecutive n-dimensional wavefunction superposition R, S pairs comprising target wavefunction superposition R and S input to steps **111** and **129** respectively.

More specifically, each of the complex matrices output from step **115** (each such complex matrix is designated as "frame h" for purposes of discussion) is processed to overlap a portion of a given frame h with a portion of its predecessor frame using an overlap buffer **117** and a matrix complex add operation at step **119**. For discussion purposes, each predecessor and successor complex matrix output from step **115** is designated as "frame h−1" and "frame h+1", respectively. For the example of an initial input target n-dimensional sequence with lengths in each dimension of (N1,N2, . . . , Nn), each frame h contains (2N1, 2N2, . . . , 2Nn) elements. For purposes of discussion, the elements for frame h may be considered as divided, according to the continuous dimension specified in the Regional Metadata associated with the input **70**, into a first half of an n-dimensional range of elements with coordinates 0 to N−1 in the continuous dimension, and a second half of an n-dimensional range of elements with coordinates N to 2N−1 in the continuous dimension (each half contains a range of N coordinates in the continuous dimension). For each frame output from step **115**, the matrix elements 0 to N−1 in the continuous dimension from the first half of the frame are stored in an n-dimensional overlap buffer at step **117**. N-dimensional overlap buffers **117** and **135** may be any suitable form of memory or other storage.

Next, in step **119** each of n-dimensional matrix elements N to 2N−1 in the continuous dimension in the second half of a new frame h output from step **115** comprising negative lag correlations of frame h is added using n-dimensional matrix complex addition step **119** to the corresponding elements 0 to N−1 in the continuous dimension in the first half of the preceding output frame h−1 comprising positive lag correlations of frame h−1, which was previously stored in overlap buffer **117**. In other words, element 0 in the continuous dimension is added to element N in the continuous dimension, element 1 is added to element N+1, and so forth, and element N−1 is added to element 2N−1. The output from step **119** ($VCAM_R$) is a set of n-dimensional complex correlations in the range 0 to N−1 in the continuous dimension for each incoming frame h. The total number of matrices $VCAM_R$ output from step **119** is one for each frame input **70** RS pair processed in steps **111** and **129**. During loop processing, the total number of matrices $VCAM_R$ output from step **119** is equal to the number of target n-dimensional wavefunction superpositions input in **70** Target RS pair that are passed to steps **111** and **129**.

The first half (elements with coordinates 0 to N−1 in the continuous dimension) comprising positive lag correlations of the same new output frame h from the unitary n-dimensional orthogonal inverse transform **115** is stored in overlap buffer **117** until it is brought together with the second half (elements with coordinates N to 2N−1 in the continuous dimension) comprising negative lag correlations of the next output frame h+1 from inverse transform **115**. As a result of this repeated overlap and matrix complex addition processing, matrix complex addition **119** outputs a matrix $VCAM_R$

with a length that is half that of the output matrices from unitary n-dimensional orthogonal inverse transform **115**.

In a similar manner to that described immediately above, each of many complex matrix outputs from unitary n-dimensional inverse transform step **133** is processed to overlap each frame with its predecessor and successor frames using overlap buffer **135** and n-dimensional matrix complex addition **137**. As a result of this repeated overlap and matrix addition process, n-dimensional matrix complex addition **137** outputs a matrix $VCAM_S$ with a length that is half that of the output matrix from unitary n-dimensional inverse transform **133**. The output from step **137** ($VCAM_S$) is a set of N n-dimensional complex correlations with coordinates 0 to N−1 in the continuous dimension for each incoming frame h.

The next steps involve processing the output matrix $VCAM_R$ of n-dimensional matrix complex addition step **119** via a complex inflation step **190**, and processing the output matrix $VCAM_S$ of n-dimensional matrix complex addition step **137** via a complex inflation step **198**. For each complex inflation step **190** and **198**, each element of the matrix $VCAM_R$ and $VCAM_S$, respectively, is raised to the third power using complex multiplication. In other embodiments, it should be noted that other powers or other manners of inflation may be used. Complex inflation may assist in separating background noise in matrices $VCAM_R$ and $VCAM_S$ from peak signals therein that correspond to potential matches between the n-dimensional probe and the n-dimensional target database.

The outputs from complex inflation steps **190** and **198** may be processed using optional phase filtering steps **191** and **199**. Typically, this is designed to select direct correlates and exclude phase rotated correlates since these are of greater interest in representing a pattern match. The outputs from this phase filtering are provided to modulus and normalization steps **123** and **141**, respectively. Each of these steps **191** and **199** involves phase scaling by multiplying each complex element of each of the inflated matrices by a phase scale factor. The phase scale factor may be determined by the phase of the complex element. For example, the phase scale factor F may be cosine (arctangent (imaginary/real)) for the interval where real is positive. In the case where the real is less than or equal to zero, the phase scale factor F is zero. The phase filtered complex value output is (F*real, F*imag). The phase scaling may eliminate correlations that are out of phase. For example, there may be a match that is not a direct phase match to a probe, but instead is a rotation of that phase (e.g., the same sequence rotated by 90°).

The two parallel processing paths continue with the providing of the output of complex inflation step **190** to modulus and normalization step **123**, and providing of the output of complex inflation step **198** to modulus and normalization step **141**. Each of the modulus and normalization steps **123** and **141** involves calculation of the real modulus of each input complex element followed by a multiplication by a normalization scale factor that is constant for all elements in the input n-dimensional matrix to step **123** or **141**. The normalization scale factor may be determined using the theoretical maximum correlation value corresponding to the n-dimensional matrix outputs from steps **115** and **133**. The normalization scale factor may be calculated as the desired output peak height divided by a total input energy calculated according to the energy, or power metric, of the input probe n-dimensional wavefunction **147**. In the some embodiments the probe or search n-dimensional wavefunction prepared by method **103** is encoded by step **21** to generate an interference output of unity for a potential probe sequence that is 100% similar.

In other embodiments the same principles of power conservation will apply with different intermediate scale factors. For example, in the method **103** the n-dimension probe sequence **17** is transformed to an n-dimension wavefunction **27** via steps **21** and **25**, each of which may apply predetermined scaling factors to the total power of probe sequence **17**. As a result the total power of probe n-dimensional wavefunction **147** input to method **107** would be a predetermined scale factor of the total power of the n-dimensional probe **17**.

The n-dimensional interference process **107** calculates any correlation between the n-dimensional probe data sequence and the n-dimensional target data sequence represented as an RS pair of n-dimensional wavefunction superpositions, by n-dimensional matrix complex multiplication in steps **113** and **131** of the target n-dimensional wavefunctions **111** and **129** with the complex conjugate of the probe n-dimensional wavefunction **147** complex matrix, followed by unitary n-dimensional orthogonal inverse transforms **115** and **133**.

The output of each step **123** and **141** is a matrix of scalar numbers (each matrix is designated as RVM_R and RVM_S, respectively) representing the scaled magnitude of the corresponding input complex matrix elements. For example, for an exemplary input frame length of N1, . . . , Nn elements in each of then dimensions, RVM_R and RVM_S contain N1, . . . , Nn real numbers in each of then dimensions. Each of the matrices RVM_R and RVM_S is a correlation at a particular n-dimensional grid position between the probe n-dimensional sequence and the target n-dimensional sequence. It should be noted that this correlation may be measured at every grid position in the target n-dimensional sequence of data elements.

The outputs of steps **123** and **141** may be temporarily saved to n-dimensional storage buffers **125** and **143**, respectively. These n-dimensional storage buffers may be used to hold these normalized results of the n-dimensional superposition R and S wavefunction interferences with the n-dimensional probe wavefunction so that the results may be further analyzed to determine if any hits or significant similarities are present. In this specific context, a "hit" refers to a peak value in the normalized data output from step **123** or **141**.

In typical use, there is a stream of n-dimensional real matrices RVM_R and RVM_S, with corresponding RVM_R and RVM_S matrices stored in n-dimension storage buffer **125** or **143** for each n-dimensional superposition wavefunction R, S pair (ψRM, ψSM) that entered method **107** at steps **111** and **129**. The number of RVM_R, RVM_S matrix pairs will depend, for example, on the resolution of an n-dimensional image in the database that was selected for searching. The real matrices typically abut one another directly (i.e., this process is like the assembly of one long real matrix, with a length depending, for example, the number of layers being processed in parallel). The RVM_R and RVM_S n-dimensional matrices are examined for peaks that correspond to high correlations between the n-dimensional probe and the n-dimensional target.

Hit Detection Process in Multidimensional Search

In an attempt to capture all potentially significant matches between the n-dimensional probe and n-dimensional target sequences, the hit detection process described below contains several steps. These steps are designed to identify all

potential candidates, then to perform a more rigorous qualification of these candidates to attempt to eliminate false detection events. The hit detection process outputs qualified events comprising a pair of hits, one each from the parallel R and S n-dimensional wavefunction interference processing paths described above. However, as part of the hit detection process, the real n-dimensional matrices RVM_R and RVM_S are initially each screened for hits independently. This approach improves the likelihood that all significant matches between the n-dimensional probe and n-dimensional target sequences will be found since two detection opportunities are thus created for each potential match.

The hit detection process begins with a threshold calculation at step **127**, which is applied to data from real n-dimensional matrix RVM_R obtained from n-dimensional matrix storage buffer **125** and a threshold calculation at step **145**, which is applied to data from real n-dimensional matrix RVM_S obtained from n-dimensional matrix storage buffer **143**. In each case, the normalized magnitude data are analyzed using statistical methods to calculate the mean and standard deviation of each set of data. In threshold calculation **127** two threshold values are calculated for the n-dimensional R wavefunction interference data, and in threshold calculation **145** two thresholds are calculated for the n-dimensional S wavefunction interference data.

Each pair of thresholds consists of a primary threshold and a secondary threshold, with the primary threshold being greater than the corresponding secondary threshold. These thresholds are calculated by adding the mean value to a multiple of the standard deviation value of the respective data set. For example, the multiples may be sixteen times in the case of the primary threshold and six times in the case of the secondary threshold. For the primary threshold a larger multiple is applied to the standard deviation value, and for the secondary threshold a smaller multiple is applied to the standard deviation value. In addition, a tertiary threshold is calculated for each data set using the combined means from steps **127** and **145** plus a multiple of the combined standard deviations from steps **127** and **145**. This multiple is, for example, sixteen times the combined standard deviations. The tertiary threshold is used to test the combined maximum values in a pair of R, S hits. The use of the primary, secondary and tertiary thresholds is described below.

The next steps in the detection process are the determination of a maximum in a window in a step **151**, which is applied to the data from the n-dimensional R wavefunction interference in storage buffer **125**, and a maximum in a window in a step **171**, which is applied to the data from the n-dimensional S wavefunction interference in storage buffer **143**. The window length is in each case equal to the effective length of the n-dimensional probe sequence, where the effective length refers to the n-dimensional bounding region of the probe n-dimensional sequence. Using a window size determined in this manner attempts to account for a hypothetical worst case scenario in which the probe sequence repeats itself as two adjacent sequences in the overall target sequence.

The window is advanced across the full length of each real n-dimensional matrix RVM_R and RVM_S. The window is advanced, after each selection of a maximum value, by a number of index positions in matrix RVM_R or RVM_S equal to the total window length so that no given sequence position of the real matrix is examined in step **151** more than once. A maximum value is selected from within the window for each matrix RVM_R or RVM_S at each position in its advance along the corresponding n-dimensional matrix. The output of maximum in window step **151** is the n-dimensional coordinate position (selected from positions with coordinates in the range 0 to N1−1, 0 to N2−1, . . . , 0 to Nn−1 in this example) and magnitude of the maximum value within the real n-dimensional matrix RVM_R from storage buffer **125**. A value and n-dimensional coordinate data pair is output for each advance of the window along RVM_R. Similarly, the output of maximum in window step **171** is the n-dimensional coordinate position and magnitude of the maximum value within the real n-dimensional matrix RVM_S from storage buffer **143**. A maximum value and n-dimensional coordinate position data pair is output for each advance of the window in RVM_S.

The compare maximum to thresholds step **153** first tests whether the maximum value output from maximum in window step **151** meets or exceeds the primary threshold value derived in threshold calculation step **127**. If it does, the result is considered to be a "Potential R Hit" **155** that is processed further as described below. Each Potential R Hit corresponds to a n-dimensional coordinate position and magnitude in RVM_R.

Similarly, compare maximum to thresholds step **173** first tests whether the maximum value output from maximum in window step **171** meets or exceeds the primary threshold value derived in threshold calculation **145**. If it does, the result is a "Potential S Hit" **175** that is processed further as described below. Each Potential S Hit corresponds to a n-dimensional coordinate position and magnitude in RVM_S.

Alternatively, if the maximum value output from maximum in window step **151** is below the primary threshold value derived in threshold calculation step **127**, and the maximum value output from maximum in window step **171** is below the primary threshold value derived in threshold calculation step **145**, processing will continue by returning to maximum in window step **151** and maximum in window step **171**, advancing the windows along each n-dimensional matrix RVM_R or RVM_S to generate new maxima for testing in the compare maximum to thresholds step **153** and compare maximum to thresholds step **173** as described above.

If a Potential R Hit **155** exists, a search is performed in step **177** for the existence of a corresponding or dual hit in an attempt to make an "R, S Hit Pair". It should be noted that if a real correlation between the n-dimensional probe and n-dimensional target exists, then two significant corresponding correlations that were calculated from the superposition R and S wavefunctions (as was discussed earlier above) should also exist at two n-dimensional coordinate positions. An R, S Hit Pair would correspond to these two correlations and consist of an R hit and an S hit. The method described below intelligently searches for a corresponding or dual R hit or S hit in a range in which such a hit is expected to be if it corresponds to a truly significant correlation between the probe and the target. In general, the n-dimensional coordinate positional separation will be two times the n-dimensional coordinate position of the layer number that the signal causing the correlation peaks to appear was earlier encoded into. For example, as will become more clear from the discussion below, if the hit is in a portion of the target sequence that was encoded by a layer delta function generated from n-dimensional coordinate of (3, 0, . . . , 0) for layer 3, the positions of the R hit and S hit will be separated by six positions in the first dimension of n-dimensional coordinate positions used for the elements of real matrices RVM_R or RVM_S.

Step **177** first tests, based upon the n-dimensional position of the Potential R Hit within the window used in step **151**, if the entire range of possible dual S hit positions (i.e.,

corresponding to the Potential R Hit) was already included within the window used for the obtaining the S maximum from matrix RVM_S in step **171**. The entire range of possible dual S hit positions is defined relative to the Potential R Hit position by the largest separation delta generated by layer encoding. For any target n-dimensional sequence being processed, it is known how many layers have been combined (i.e., the depth of the superposition wavefunctions used) and the range of n-dimensional positions used to generate layer delta functions, hence the largest separation delta generated by layer encoding is also known.

If the entire range of possible dual S hit positions was included in the window of step **171**, then the S maximum from step **171** may be used as the possible dual S hit corresponding to the Potential R Hit **155**. Alternatively, if the entire range of possible dual S hit positions was not included in the window of step **171**, then a search of the n-dimensional matrix RVM_S is done to select the maximum value from within the entire range of potential dual S hit positions in RVM_S for use as the S maximum.

Next, the n-dimensional coordinate separation delta of the R and S maxima is calculated. The "n-dimensional coordinate separation delta" is equal to the n-dimensional coordinate position of the R maximum minus the n-dimensional coordinate position of the S maximum. The n-dimensional coordinate separation delta is then tested to see if it meets the following three conditions: (i) the n-dimensional coordinate separation delta is non-negative, (ii) the n-dimensional coordinate separation delta is even-numbered, and (iii) the n-dimensional coordinate separation delta is within the range of position separation generated by n-dimensional coordinate layer encoding.

If the n-dimensional coordinate separation delta satisfies all of the above three conditions, the magnitude of the S maximum is compared against the secondary threshold previously calculated in threshold calculation step **145**. If the magnitude of the S maximum meets or exceeds the secondary threshold, then the sum of the magnitudes of the R and S maxima is compared against the tertiary threshold previously calculated from steps **127** and **145**. If the sum of the magnitudes of the R and S maxima is greater than or equal to the tertiary threshold, then the R and S maxima are passed as a potential dual R hit and S hit pair (or simply "R, S hit pair") to step **179**.

Step **179** calculates the n-dimensional coordinate offset and n-dimensional coordinate separation of the potential dual R, S hit pair. Specifically, step **179** finds the n-dimensional coordinate position midpoint between the R and S maxima and records this as the n-dimensional coordinate offset of the R, S hit pair. In equation form: position midpoint(m)=(position(m)$_R$ maximum$^+$position(m)$_S$ maximum)/2, where m is each of the n-dimensional coordinates. The n-dimensional coordinate separation delta of the two maxima is also determined. The n-dimensional coordinate separation delta is equal to the n-dimensional coordinate position of the R maximum minus the n-dimensional coordinate position of the S maximum.

Similarly as was described above for the existence of a Potential R Hit, if a Potential S Hit **175** exists, a search is performed in step **157** for the existence of a dual hit in an attempt to make an R, S Hit Pair. Step **157** first tests, based upon the position of the Potential S Hit within the window used in step **171**, if the entire range of possible dual R hit positions (i.e., corresponding to the Potential S Hit) was already included within the window used for the obtaining the R maximum from matrix RVM_R in step **151**. The entire range of possible dual R hit positions is defined relative to the Potential S Hit position by the largest n-dimensional coordinate separation delta generated by n-dimensional coordinate layer encoding.

If the entire range of possible dual R hit positions was included in the window of step **151**, then the R maximum from step **151** may be used as the possible dual R hit corresponding to the Potential S Hit **175**. Alternatively, if the entire range of possible dual R hit positions was not included in the window of step **151**, then a search of the matrix RVM_R is done to select the maximum value from within the entire range of potential dual R hit positions in RVM_R for use as the R maximum.

Next, following step **157**, is step **159**, in which the n-dimensional coordinate separation delta of the R and S maxima is calculated. The "n-dimensional coordinate separation delta" is equal to the n-dimensional coordinate position of the R maximum minus the n-dimensional coordinate position of the S maximum. The separation delta is then tested to see if it meets the following three conditions: (i) the n-dimensional coordinate separation delta is non-negative, (ii) the n-dimensional coordinate separation delta is even-numbered, and (iii) the n-dimensional coordinate separation delta is within the range of position separation generated by n-dimensional coordinate layer encoding.

If the n-dimensional coordinate separation delta satisfies all of the above three conditions, the magnitude of the R maximum is compared against the secondary threshold previously calculated in threshold calculation step **127**. If the magnitude of the R maximum meets or exceeds the secondary threshold, then the sum of the magnitudes of the R and S maxima is compared against the tertiary threshold previously calculated from steps **127** and **145**. If the sum of the magnitudes of the R and S maxima is greater than or equal to the tertiary threshold, then the R and S maxima are passed as a potential dual R,S hit pair to step **179**.

Step **159** calculates the n-dimensional coordinate offset and n-dimensional coordinate separation of the potential dual R,S hit pair. Specifically, step **159** finds the n-dimensional coordinate position midpoint between the R and S maxima and records this as the n-dimensional coordinate offset of the R,S hit pair. In equation form: position midpoint (m)=(position(m)$_{R\ maximum}$+position(m)$_{S\ maximum}$)/2, where m is each of the n-dimensional coordinates. The n-dimensional coordinate separation delta of the two maxima is also determined. The n-dimensional coordinate separation delta is equal to the n-dimensional coordinate position of the R maximum minus the n-dimensional coordinate position of the S maximum.

As mentioned above, the processing steps **151**, **153**, **155**, **157** and **159** provide a substantially parallel path to the processing steps **171**, **173**, **175**, **177** and **179**, assisting in ensuring that the R and S normalized interference data are initially each screened for hits independently. This approach attempts to ensure that all significant matches between the n-dimensional probe and n-dimensional target sequences will be found since two detection opportunities are created for each match between the n-dimensional probe and the n-dimensional target. The results from these two parallel paths converge in step **181**, which qualifies the R,S Hit Pairs. More specifically, step **181** eliminates R,S Hit Pair duplicates from these results (e.g., a duplicate R,S Hit Pair may have arisen as a result of using the parallel detection processes discussed earlier).

As will be discussed in more detail below, when originally preparing the target for searching, the target may be divided into a number of regions with each region encoded to one of the layer indices used for forming the layer-encoded super-

position wavefunctions. Each region is also represented by Regional Metadata **34** which was associated with the target RS pair of layer encoded superposition wavefunctions **70** by the target preparation method **105**. In step **193**, data for each of the R,S Hit Pairs detected in the preceding steps is used with the Regional Metadata to identify the location in the target sequence that matches the probe sequence for the given R,S Hit Pair. The location in the target that matches the probe sequence is found by first identifying which n-dimensional coordinate position the matching target sequence was encoded by, which is next used to recover the Regional Metadata of the matching target sequence. The location in the target sequence of the matching probe is then calculated by combining the bounding region of the encoded target sequence in the recovered Regional Metadata with the n-dimensional coordinate offset of the R,S Hit Pair, which is equal to the n-dimensional coordinate position midpoint between the R and S maxima.

The superposition integer layer index for the R,S Hit pair is obtained by first calculating the n-dimensional coordinate position of the encoding delta function by halving the layer n-dimensional coordinate separation delta from steps **159** and **179** (equal to the n-dimensional coordinate position of the R maximum minus the n-dimensional coordinate position of the S maximum). The n-dimensional coordinate position in the Regional Metadata of all layers encoded in the target RS pair **70** is then compared with the calculated n-dimensional coordinate position of the encoding delta function and the Regional Metadata in which they are equal is selected. The selected Regional Metadata comprises the integer layer index and the region n-dimensional bounds of the n-dimensional target sequence in which the probe sequence match was detected as the RS Hit pair.

The n-dimensional coordinate separation delta of the two maxima (as mentioned above, the n-dimensional coordinate separation delta is equal to the n-dimensional coordinate position of the R maximum minus the n-dimensional coordinate position of the S maximum), is divided by two to give the n-dimensional coordinate position used to encode the superposition layer in which the hit was detected. In other words, for each of the n dimensions the coordinate of the encoding delta function is equal to half the difference between the coordinate of the R maximum minus the coordinate of the S maximum. The halving of the separation delta of the two maxima is the final step of a unitary decoding by the wavefunction interference method **107** of the layer encoded target RS pair **70**. The unitary decoding by method **107** is the inverse of a unitary encoding by target preparation method **105** in that any n-dimensional coordinate position used to encode a layer in method **105** may be recovered for a matching probe sequence. In target preparation method **105** the target wavefunction in n-dimensional matrix storage buffer **45** is multiplied by a modulation wavefunction in n-dimensional matrix storage buffer **53** to generate a layer in the target n-dimensional wavefunction superposition R matrix. As a result, the positions of potential probe matches will be translated by the n-dimensional coordinate position used to generate the layer encoded as a delta function **47**. Similarly, multiplication of the target wavefunction by the complex conjugate of the same modulation wavefunction to generate a layer in the target n-dimensional wavefunction superposition S matrix corresponds to translation of the positions of potential probe matches by the negative of the n-dimensional coordinate position used to generate the layer encoded as a delta function **47**. As a result of the equal positive and negative translations in each dimension, a separation is created in each dimension that is twice the positive translation in that dimension. Consequently, the positive translation in each dimension can be recovered by halving the separation in each dimension. The superposition layer number is in order to yield the target in which the matching sequence is located.

FIG. **5**C illustrates a flowchart for a method of sequence searching **300** that is optimized for the special case in which the target sequence has a natural modulus, using interference between the probe and target wavefunctions prepared by the methods of FIGS. **3**A and **4**A, in accordance with some embodiments. This is also known as the zero-offset use case, for reasons that are explained elsewhere in this document. Many of the steps in method **300** shown in FIG. **5**C are analogous to those shown in FIGS. **5**A-**5**B and are thus referred to by the same reference numeral. In addition, the descriptions of these steps are not repeated for the sake of brevity. However, the need for certain outputs of certain steps (e.g., step **114**'s output to overlap buffer **116**) are obviated in accordance with the optimizations in light of the zero-offset use case. Further details about these optimizations are described with reference to FIGS. **11**A-**11**D.

The difference between FIG. **5**C and FIGS. **5**A-**5**B is that, after the unitary orthogonal inverse transform step **132**, the resulting data is reflected around lag 0 in step **183**. The output of step **183** is data that aligns with the resulting data from unitary orthogonal inverse transform step **114** (as will be clear from FIG. **5**D and FIG. **5**J, there are at least two other implementations through which to make this alignment of the data). The two sets of data, now aligned, are added in a vector complex addition step **184**, in effect, increasing the signal-to-noise ratio of either individual unitary orthogonal inverse transform (i.e., step **114** or step **132**). The method proceeds through the numbered steps as described with reference to FIGS. **5**A-**5**B until step **185**. At step **185**, in some embodiments, the result of step **140** is compared to a predefined threshold, which in some embodiments is predefined based on a calculation done in threshold calculation step **144** based on the parameters of the data stored in the target sequence (e.g., the length of entries, the number of entries, but without regard to further information in the storage buffers **124/142**, FIG. **5**A).

Stated another way, in FIG. **5**A, the position coordinate variable is maintained between frames using Overlap Buffers in steps **116** and **134**. The first consequence of the zero position offset alignment use case is that the Overlap Buffers in steps **116** and **134** are not required (as shown in FIG. **5**C) since there is no position overlap between frames. In addition, another consequence of the zero position offset alignment use case is that the zero position offset implies a reflection symmetry around zero lag between the output results of unitary orthogonal inverse transforms in steps **114** and **132**. This symmetry property provides a computational path to combine the R and S Hit deltas by negating the index of the S encoded target wavefunction output of step **132** to map on top of the positive index of the R encoded target wavefunction from step **114**. Negating the index of the S encoded target wavefunction output of step **132** is achieved by reflect data around lag 0 step **183**.

The combination of R and S Hit deltas in step **184** of FIG. **5**C has two advantages, benefitting both performance and efficiency. Firstly, it makes full use of the orthogonality of the error component of the output of step **114** versus the error component of step **132**: complex vector addition in step **184** allows R and S errors to either cancel each other or otherwise not reinforce each other thereby increasing the signal to noise and discrimination of the hit detection process. Secondly, it has the advantage of reducing two

downstream paths to a single downstream computational path that processes the R,S hit data more efficiently. A contribution to this greater efficiency is that a dual R and S hit detection process in steps **154**, **156**, **158**, **175**, **176**, **178** and **180** of FIG. **5**B has been replaced by a single, more discriminating, detection process that selects mirror encoded R and S hits in FIG. **5**C.

FIG. **5**D illustrates a flowchart for another method of sequence searching **301** that is optimized for the special case in which the target sequence has a natural modulus, using interference between the probe and target wavefunctions prepared by the methods of FIGS. **3**A and **4**A and the assessment of hits that are located, in accordance with some embodiments. Many steps in the method shown in FIG. **5**C are analogous to those shown in FIGS. **5**A-**5**B and are thus referred to by the same reference numeral. In addition, the description of these steps is not repeated for the sake of brevity. However, the need for certain outputs of certain steps (e.g., step **114**'s output to overlap buffer **116**) are obviated in accordance with the optimized method for the special case in which the target sequence has a natural modulus. Further details about these optimizations are described with reference to FIGS. **11**A-**11**D.

The difference between FIG. **5**D and FIG. **5**C is that, rather than reflecting the resulting data from step **132** around lag zero to create the alignment with data from step **114**, the alignment is performed prior to applying a unitary orthogonal inverse transform. To that end, the alignment is performed (e.g., in wavefunction space, sometimes called a frequency space) by taking the complex conjugate of the result of the vector complex multiply step **130** prior to obtaining a unitary orthogonal inverse transform in step **114**. In addition, in FIG. **5**D, the vector complex multiplied data from step **112** is added to the complex conjugated data **186** in a vector complex addition step **187** before the unitary orthogonal inverse transform step **114**. This has the effect of increasing the signal-to-noise in the wavefunction space prior to the inverse transform. Another effect is that the need for two unitary orthogonal inverse transform steps is obviated, and hence step **132** is omitted from FIG. **5**D.

The computational effect achieved by step **183** in FIG. **5**C by reflecting data around lag 0 is also known by the term "time reversal," since each data point at a positive time index moves to its corresponding negative time index and vice-versa. The time reversal operation in step **183** of FIG. **5**C has a dual domain equivalent in the orthogonal domain that is the operation of phase conjugation in step **186** of FIG. **5**D. The same performance and efficiency advantages of combining orthogonal error components of outputs of steps **114** and **132** in FIG. **5**C are equivalently realized by combining orthogonal error components of outputs of steps **112** and **130** in FIG. **5**D. By combining R,S Hit data by complex addition in step **187**, only a single Unitary Orthogonal inverse transform step **114** is needed in FIG. **5**D versus two separate Unitary Orthogonal inverse transforms in steps **114** and **132** in FIG. **5**A and/or FIG. **5**C.

An additional optimization that may be applied in the zero position offset use case is to select the unitary orthogonal inverse transform to have the same computational properties but partitioned optimally between the probe or search wavefunction encoding method **10** in FIG. **3**A and the sequence searching method **100** in FIG. **5**A, method **301** in FIG. **5**C, method **300** in FIG. **5**D and method **222** in FIG. **5**J.

Different unitary orthogonal forward/inverse transform pairs may have asymmetric loading between their forward and inverse transform counterparts. Since search requests benefit from lower latency more than storage encoding requests, and also predominate over storage encoding requests in frequency of usage, it is more optimal to have the unitary orthogonal inverse transform asymmetrically partitioned to be less computational intensive than the unitary orthogonal forward transform.

The unitary orthogonal inverse transform **114** in FIGS. **5**A, **5**C and **5**D can be made less computational intensive than the unitary orthogonal forward transform **24** in FIG. **3**A and **42** in FIG. **4**A by only calculating output data corresponding to the zero position offset and the encoding superposition layer. Using such a modified unitary orthogonal inverse transform step **114**, the number of output data calculated will be a fraction of the unmodified unitary orthogonal inverse transform step **114**. When the encoded delta function indices of consecutive layers are separated by a multiplication factor the number of output values calculated may be reduced by the same multiplication factor, since the values not at the spatially scaled delta functions are not required. The reduced number of output values for a modified unitary orthogonal inverse transform **114** result from a multiplication factor applied to the separation of the layer index encoded as delta function position index in step **46**. The multiplication factor has two advantages of: (a) decreasing the signal overlap between different layer encoded representations so allowing more useable layers; (b) decreasing the number of output values calculated and the total computation operations required.

FIG. **5**G illustrates a flowchart for a method of n-dimensional discrete sequence searching **303** that is optimized for the special case in which the n-dimensional target sequence has a natural modulus, using interference between the n-dimensional probe wavefunctions prepared by the method **103** of FIG. **3**B and RS pairs of n-dimensional target wavefunctions prepared by the discrete variant of method **105** for target preparation depicted in FIG. **4**B, in accordance with some embodiments. This is also known as the zero-offset use case, for reasons that are explained elsewhere in this document. Many of the steps in method **303** shown in FIG. **5**G are analogous to those shown in FIGS. **5**E-**5**F and are thus referred to by the same reference numeral. In addition, the descriptions of these steps are not repeated for the sake of brevity. However, the need for certain outputs of certain steps (e.g., step **115**'s output to overlap buffer **117**) are obviated in accordance with the optimizations in light of the zero-offset use case.

The primary difference between FIG. **5**G and FIGS. **5**E-**5**F is that, after the unitary n-dimensional orthogonal inverse transform step **133**, the resulting data is reflected in the n-dimensional coordinate origin in step **188**. The output of step **188** is data that aligns with the resulting data from unitary n-dimensional orthogonal inverse transform step **115** (as will be clear from FIG. **5**H, there is at least one other implementation through which to make this alignment of the data). The two sets of n-dimensional data, now aligned, are added in a n-dimensional matrix complex addition step **189**, in effect, increasing the signal-to-noise ratio of either individual unitary n-dimensional orthogonal inverse transform (i.e., step **115** or step **133**). The method proceeds through the numbered steps as described with reference to FIGS. **5**E-**5**F until step **194**.

At step **194**, in some embodiments, a maximum in n-dimensional window region is calculated on the real normalized correlation output of step **141**. In the case that the probe n-dimensional sequence data has a variable n-dimensional offset relative to the potential probe matches in the n-dimensional target data sequence, finding the maximum cor-

relation in a region will compensate for such n-dimensional coordinate position variations.

At Compare Maximum to Thresholds step **173**, in some embodiments, the result of step **194** is compared to a predefined threshold, which in some embodiments is predefined based on a calculation done in threshold calculation step **145** based on the parameters of the data stored in the target sequence (e.g., the lengths of entries, the number of entries, but without regard to further information.

At step **194**, in some embodiments, a maximum in n-dimensional region is calculated on the real normalized correlation output of step **141** for each integer layer index in a procedure that is described below that iterates over the integer layer indexes and for each one obtains, for example, the scaled and bit-reversed n-dimensional zigzag pattern coordinates described in Partition Regions **33** which was passed via Regional Metadata **34** to step **47**, layer encode as delta functional at n-dimensional coordinate position. With the delta function n-dimensional coordinate positions, step **194** also obtains the dimensions of the n-dimensional region that has the single non-zero delta function at its center. As a result step **194** can define limits of an n-dimensional window with which to calculate the maximum in region output correlation.

At step **173**, in some embodiments, the result of step **194** comprising a maximum correlation value is compared to a predefined threshold, which in some embodiments is predefined based on a calculation done in threshold calculation step **145** based on the parameters of the data stored in the target sequence (e.g., the length of entries, the number of entries, but without regard to further information).

The output of step **195** comprising the R, S Hit pair in which the hit correlation value at an n-dimensional coordinate offset exceeded the predetermined threshold, the layer in the R, S Hit pair in which the hit exceeded the predetermined threshold is obtained as an integer layer index in the Regional Metadata associated with input **70** which has an n-dimensional coordinate position equal to the n-dimensional coordinate position offset of the hit correlation value that exceeded the predetermined threshold. The integer layer index is passed to the output of method **303**.

Stated another way, in FIG. **5**E, the n-dimensional position coordinate variable is maintained between frames using Overlap Buffers in steps **117** and **135**. The first consequence of the zero position offset alignment use case is that the Overlap Buffers in steps **117** and **135** are not required (as shown in FIG. **5**G) since there is no n-dimensional coordinate position overlap between frames. In addition, another consequence of the zero position offset alignment use case is that the zero position offset implies a reflection symmetry around zero lag between the output results of unitary n-dimensional orthogonal inverse transforms in steps **115** and **133**. This symmetry property provides a computational path to combine the R and S Hit deltas by negating the n-dimensional coordinate indexes of the S encoded target wavefunction output of step **133** to map on top of the positive n-dimensional coordinate indexes of the R encoded target wavefunction from step **115**. Negating the n-dimensional coordinate indexes of the S encoded target wavefunction output of step **133** is achieved by reflect data in the n-dimensional coordinate origin step **188**.

The combination of R and S Hit deltas in step **189** of FIG. **5**G has two advantages, benefiting both performance and efficiency. Firstly, it makes full use of the orthogonality of the error component of the output of step **115** versus the error component of step **133**: n-dimensional complex matrix addition in step **189** allows R and S errors to either cancel each other or otherwise not reinforce each other thereby increasing the signal to noise and discrimination of the Hit detection process. Secondly, it has the advantage of reducing two downstream paths to a single downstream computational path that processes the R,S Hit data more efficiently. A contribution to this greater efficiency is that a dual R and S hit detection process in steps **155**, **157**, **159**, **175**, **177**, **179** and **181** of FIG. **5**F has been replaced by a single, more discriminating, detection process that selects mirror encoded R and S hits in FIG. **5**G

FIG. **5**H illustrates a flowchart for another method of multidimensional sequence searching **304** that is optimized for the special case in which the target sequence has a natural modulus, using n-dimensional interference between the n-dimensional probe prepared by the method of FIG. **3**B and n-dimensional target wavefunctions prepared by the discrete variant of method **105** depicted in FIG. **4**B and the assessment of hits that are located, in accordance with some embodiments. Many steps in the method shown in FIG. **5**H are analogous to those shown in FIG. **5**G and FIGS. **5**E-**5**F and are thus referred to by the same reference numeral. In addition, the description of these steps is not repeated for the sake of brevity. However, the need for certain outputs of certain steps (e.g., step **115**'s output to overlap buffer **116**) are obviated in accordance with the optimized method for the special case in which the target sequence has a natural modulus.

The difference between FIG. **5**H and FIG. **5**G is that, rather than reflecting the resulting data from step **133** in the n-dimensional coordinate origin **188** to create the alignment with n-dimensional data from step **115**, the alignment is performed prior to applying unitary n-dimensional orthogonal inverse transform **115**. To that end, the alignment is performed (e.g., in wavefunction space, sometimes called a n-dimensional frequency space) by taking the complex conjugate **196** of the result of the n-dimensional matrix complex multiply step **131** prior to obtaining a unitary n-dimensional orthogonal inverse transform in step **115**. In addition, in FIG. **5**H, the matrix complex multiplied data from step **113** is added to the complex conjugated n-dimensional complex matrix data from step **196** in a matrix complex addition step **197** before the unitary orthogonal inverse transform step **115**. This has the effect of increasing the signal-to-noise in the n-dimensional wavefunction space prior to the n-dimensional inverse transform. Another effect is that the need for two unitary n-dimensional orthogonal inverse transform steps is obviated, and hence step **133** is omitted.

The computational effect achieved by step **188** in FIG. **5**G by reflecting data in the n-dimensional coordinate origin is also known by the term "time reversal," since each data point at a positive time index moves to its corresponding negative time index and vice-versa. The time reversal operation in step **188** of FIG. **5**G has a dual domain equivalent in the orthogonal domain that is the operation of phase conjugation in step **196** of FIG. **5**H. For any number of dimensions, the dual domain symmetry applies to a pair of orthogonal n-dimensional dual domains: the complex conjugation of the n-dimensional matrix data before n-dimensional orthogonal inverse transformation is equivalent to a negation of the coordinates, or a reflection in the n-dimensional coordinate origin, of the n-dimensional matrix data in the orthogonal n-dimensional domain following the transformation. The same performance and efficiency advantages of combining orthogonal error components of outputs of steps **115** and **133** in FIG. **5**G are equivalently realized by combining orthogonal error components of outputs of steps

113 and 131 in FIG. 5H. By combining R,S Hit data by n-dimensional matrix complex addition in step 197, only a single Unitary n-dimensional Orthogonal inverse transform step 115 is needed in FIG. 5H versus two separate Unitary n-dimensional Orthogonal inverse transforms in steps 115 and 133 in FIG. 5F and/or FIG. 5G.

An additional optimization that may be applied in the zero position offset use case is to select the unitary n-dimensional orthogonal inverse transform to have the same computational properties but partitioned optimally between the n-dimensional probe or search wavefunction encoding method 103 in FIG. 3B and the sequence searching method 107 in FIG. 5E-5F, method 303 in FIG. 5G and method 304 in FIG. 5H.

Different unitary orthogonal forward/inverse transform pairs may have asymmetric loading between their forward and inverse transform counterparts. Since search requests benefit from lower latency than storage and encoding request, and also predominate over storage encoding requests in frequency of usage, it is more optimal to have the unitary orthogonal inverse transform asymmetrically partitioned to be less computational intensive than the unitary orthogonal forward transform. Less computational intensive is fewer or simpler arithmetic operations to achieve an equivalent result. The more efficient transforms especially have fewer multiplication operations since multiplication is more computationally intensive than addition or subtraction. The unitary n-dimensional orthogonal inverse transform may also be made more efficient by using the particular requirements of certain applications as will be described below.

The unitary n-dimensional orthogonal inverse transform 115 in FIGS. 5E, 5G and 5H can be made less computational intensive than the unitary n-dimensional orthogonal forward transform 25 in FIG. 3B and 43 in FIG. 4B by only calculating output data corresponding to the zero position offset and the encoding superposition layer. Using such a modified unitary n-dimensional orthogonal inverse transform step 115, the number of output data calculated will be a fraction of the unmodified unitary orthogonal inverse transform step 115. When the encoded delta function indices of consecutive layers are separated by a multiplication factor the number of output values calculated may be reduced by the same multiplication factor, since the values not at the spatially scaled delta functions are not required. The reduced number of output values for a modified unitary n-dimensional orthogonal inverse transform 115 result from a multiplication factor applied to the separation of the layer index encoded as delta function position index in step 47. The multiplication factor has two advantages of: (a) decreasing the signal overlap between different layer encoded representations so allowing more useable layers; (b) decreasing the number of output values calculated and the total computation operations required.

It is a feature of the current method that it may be used to compute different sums such as correlation and dot product where each is computed by an unitary orthogonal inverse transform of the complex multiplication product of the R target wavefunction with the complex conjugate of the probe wavefunction combined with the complex conjugate of the complex multiplication product of the S target wavefunction with the complex conjugate of the probe wavefunction.

FIG. 5I illustrates a method of matrix multiplication for computing a plurality of dot products between a row vector and a plurality of column vectors. A row vector 212 comprising an integer P real or complex numbers is combined with column vectors comprising P real or complex numbers in a matrix 216 using the dot operation 214. The dot product is defined as a scalar or complex value equal to the sum of the products of corresponding values in each of two vectors. The output is a scalar or complex value for each column of matrix 216 in an ordered vector 218. The total count of multiply-accumulate operations is P, equal to the length in values of the row and column vectors (which are equal), multiplied by M, equal to the number of column vectors in matrix 216. Equivalently, the vector dot product is P multiplication and P−1 addition operations. There are consequently M·P multiplication operations and M·(P−1) addition operations required to compute a (ROW×COLUMN) matrix multiplication sum of (1×P)·(P×M)=(1×M) by other methods.

FIG. 5I also illustrates a method of tensor product of vector spaces where an outer vector space 213 is exemplified by a 2×2 matrix with elements ‘a’, ‘b’, ‘c’ and ‘d’ as a first input to tensor product operator 215 and an inner vector space 217 comprising a 2×2 matrix with elements ‘e’, ‘f’, ‘g’ and ‘h’ exemplifies a second input to tensor product operator 215. The tensor product 219 is exemplified by a 4×4 matrix with elements ‘ae’, ‘af’, ‘be’, ‘bf’, ‘ag’, ‘ah’, ‘bg’, ‘bh’, ‘ce’, ‘cf’, ‘de’, ‘df’, ‘eg’, ‘ch’, ‘dg’ and ‘dh’. According to the conventional nomenclature, a tensor product output element, ‘af’ denotes the product of ‘a’ in the outer vector space and ‘f’ in the inner vector space, and similarly for each element of 219.

FIG. 5J illustrates a flowchart for a method for computing a plurality of dot products between a row vector as a probe sequence and a plurality of column vectors that comprise the target sequence having a natural modulus, using interference between the probe and target wavefunctions prepared by the methods of FIGS. 3A and 4A and the assessment of dot products comprising hits that are located, in accordance with some embodiments.

The method that is the subject of the present disclosure is able to compute real and complex matrix multiplication products according to the variations of how the probe wavefunction and RS pair wavefunctions superposition are prepared. To compute real matrix multiplication: the method 10 input probe or search sequence of P data elements in step 16 comprises real numbers that are a copy of the row vector 212 shown in FIG. 5I; and the target or database sequence 36 comprises real numbers that are a copy of the column vector of matrix 216 shown in FIG. 5I.

To compute complex matrix multiplication: the method 10 input probe or search sequence of P data elements in step 16 comprises two sequence partitions that are mutually orthogonal according to the current method since they are spatially non-overlapping in the zero offset alignment frame. The two mutually orthogonal partitions are used for the real and imaginary components if the input row vector 212 is complex data. Similarly, the target preparation comprises two sequence partitions that are mutually orthogonal according to the current method since they also are spatially non-overlapping in the zero offset alignment frame.

The two mutually orthogonal partitions are used for the real and imaginary components of the input column vector of matrix 216 complex data. The real component of the complex multiplication is achieved by interference of the spatially separate real and imaginary sequences in the probe wavefunction interfering with corresponding alignment of the real component of the column vector of matrix 216 and the negative of the imaginary component of the column vector of matrix 216. The real component of complex multiplication is consequently computed equal to the probe real component times the target real component plus the

probe imaginary component times the negative target imaginary component. It will be apparent that alternatively, the real component of complex multiplication may be computed from the probe real component times the target real component plus the probe negative imaginary component times the target imaginary component.

The imaginary component of the complex multiplication is achieved by interference of the spatially separate real and imaginary sequences in the probe wavefunction interfering with corresponding alignment of the imaginary component of the column vector of matrix **216** and the real component of the column vector of matrix **216**. The imaginary component of the complex product is consequently computed equal to the probe real component times the target imaginary component plus the probe imaginary component times the target real component.

In the method **222** shown in FIG. **5**J the probe wavefunction **146** comprises the output of method **10** where the input probe sequence of P data elements in step **16** is a copy of the row vector **212** shown in FIG. **5**I. The target wavefunction superposition R **110** and target wavefunction superposition S **128** comprise the output of the method of FIG. **4**A on a target sequence comprising the column vectors of matrix **216**. In this case the natural modulus of the target sequence is equal to P, the length of the column vector. In other words, the matrix **216** of M columns each of P real or complex values comprises a target sequence of M times P real or complex values with a natural modulus of P.

For the purpose of comparison a (P×M) matrix **216** of M columns of P real or complex values shown in FIG. **5**I is represented by an RS pair generated by the method of target wavefunction preparation of FIG. **4**A and shown as target wavefunction superposition R **110** and target wavefunction superposition S **128** shown in FIG. **5**J. Since the target sequence of length M·P real or complex values has a natural modulus of P, it has been encoded to M separate layers in the wavefunction superposition RS pair. The RS pair comprises two complex vectors each of length an integer Q complex values. In the case where the matrix **216** is comprised of columns of the integer P complex values, the integer Q has a minimum size of two times the integer P. This is similar to the case of DNA or RNA nucleotide bases encoded to complex symbols where a sequence of length N nucleotide bases is encoded to a complex frame of length 2N complex values as described elsewhere herein.

In the case where the matrix **216** is comprised of columns of the integer P real values, the integer Q has a minimum size equal to the integer P. In both the real and complex vector cases, the number of layers in an RS pair is equal to M, the number of column vectors in matrix **216**.

There are many potential combinations of the integers P, Q and M. According to the present example of matrix multiplication all the target vectors are the same length P. In this case, for a given length of P, increasing the number of layers M requires a larger Q, the RS pair wavefunction complex frame size.

Different applications will have data with different measures of entropy and other signal properties. It is to be generally expected that data that is more sparse may be encoded with a larger integer M, the number of layers in a superposition.

According to method **222** shown in FIG. **5**J the target wavefunction superposition R **110** and target wavefunction superposition S **128** pass to vector complex add step **224** to produce the complex sum as target wavefunction superposition R+S **228**. In vector complex subtract step **226** complex subtraction of the target wavefunction superposition S **128** from target wavefunction superposition R **110** produces target wavefunction superposition R−S **230**. The combination of vector complex add **224** and vector complex subtract **226** perform a Hadamard transform on the target wavefunction superposition R **110** and target wavefunction superposition S **128**. This Hadamard transform is an information preserving unitary mapping of complex vectors **110** and **128** to complex vectors **228** and **230** so that downstream processing is minimized according to method **222**. In particular the Hadamard transform may be precomputed in which case the cost in addition and multiplication operations will be divided by the times the data is reused. For example, if the matrix **216** is reused for many row vector **212** instances, the RS pair superpositions may be formed and then processed by the Hadamard transform comprising steps **224** and **226** to form target wavefunction superposition R+S **228** and target wavefunction superposition R−S **230** which may be reused for many instances of probe wavefunction **146**. In this way the operations cost of target preparation is amortized across the number of times the data is reused. For the purpose of comparison the amortization of preparation cost will be to zero, so it will not be included in a total of the real-time operations count of method **222**.

In the method **222** an interference butterfly step **232** combines the probe wavefunction **146** with target wavefunction superposition R+S **228** and target wavefunction superposition R-S **230** to generate a complex vector input to decimation step **270**. In particular target wavefunction superposition R+S **228** is combined with probe wavefunction **146** to calculate the real component **256** of a complex input to decimation step **270**. The target wavefunction superposition R-S **230** is combined with probe wavefunction **146** to calculate the imaginary component **268** of a complex input to decimation step **270**. Decimation step **270** and unitary orthogonal inverse transform **284** together form an optimized replacement of unitary orthogonal inverse transform **114** in FIGS. **5**A, **5**C and **5**D in some embodiments.

In some embodiments decimation step **270** and unitary orthogonal inverse transform **284** together form an optimized replacement of unitary orthogonal inverse transform **114** in FIGS. **5**A, **5**C and **5**D that only calculates output data corresponding to the zero position offset and the encoding superposition layer. The multiplication factor applied in integer scaled and scaled bit-reverse encoding unitary mappings applied in layer encoded as a delta function position index step **46** may be removed by decimation step **270** using a decimation factor equal to the multiplication factor applied in step **46**. Decimation step **270** is an inverse mapping of the integer scaling of the integer layer index by the integer multiplication factor. The multiplication factor separation between the position of the delta functions for consecutive encoded layers is removed by decimation step **270** reducing the output values to the number of encoded layers.

Interference butterfly step **232** together with the Hadamard transform of steps **224** and **226** perform the same interference method comprising the combination of steps in method **300** shown in FIG. **5**D of vector complex conjugate **148**, vector complex multiply **112**, vector complex multiply **130**, vector complex conjugate **186** and vector complex add **187**. According to the method **300** these steps of two complex multiply and one complex add require two times four multiplication and two add/subtract operations per complex multiplication plus two add/subtract operations per complex add for a total of eight multiply and six add/subtract operations for each probe wavefunction complex value.

The interference butterfly **232** comprises four multiply operations, one in each of steps **248**, **250**, **262** and **264** plus

two add/subtract operations in steps **252** and **266**. As a result, four less multiply and four less add/subtract operations are required using the interference butterfly **232** on Hadamard transformed input target wavefunctions compared with the interference method **300**. Use of the interference butterfly **232** therefore produces a reduction of multiplication operations of 50% and reduction in add/subtract operations of 66%.

The interference butterfly **232** processes target wavefunction superposition R+S **228**, target wavefunction superposition R-S **230** and probe wavefunction **146** to real values **256** and imaginary values **268** of a complex vector output that is passed to decimation step **270**. The input complex vector **272** of length Q complex pairs, equal to a decimation factor d times M, is transformed to the output complex vector **274** of length M with a series of complex additions of periodically spaced values. The decimation can be generally expressed as summation of values with the same index modulo-M (% M) for input vector x to output vector y. The same function may also be expressed as $y(m)=\Sigma x(m+a\cdot M)$ with sum over a, where a is an integer between zero and the decimation factor d minus one, and m is an integer between zero and M minus one.

The complex additions in decimation step **270** may be arranged in a number of stages where the decimation factor d is equal to two to the power of the number of stages, d=2ˆstages. In this case the length of each stage progressively halves, to end equal to the output length M. As depicted in FIG. **5**J an input complex vector **272** which in this example is eight times M in length is first reduced by stage **276** comprising four times 2·M additions to a complex vector of four times M in length. This is next reduced by stage **278** comprising two times 2·M additions to a complex vector of two times M in length and finally by stage **280** comprising 2·M additions to a complex vector **274** of M in length.

For a decimation factor d and an output complex vector length of M complex values, the minimum number of addition operations in decimation step **270** is equal to $2\cdot(d-1)\cdot M$, since d−1 additions are required to sum d values, which is done for M values of reals and imaginaries. In the case where the decimation factor d is equal to two to the power of the number of stages, d=2ˆstages, in the progressive reduction shown in FIG. **5**J steps **276**, **278** and **280**, there are two M times the sum of {2ˆ(stages−1), . . . , 1} additions which equals the minimum number of $2\cdot(d-1)\cdot M$, showing that progressive reduction is one solution that is optimal when d is a power of two.

It will be apparent that decimation step **270** uses addition of real and imaginary components of complex values in separate paths to create an output of length equal to the number of wavefunction superposition layers that have been encoded into target wavefunction superposition R and target wavefunction superposition S. In other words there is one complex value output by decimation step **270** for each of the M input vectors of length P that were encoded to the target wavefunction superpositions R and S.

The output of decimation step **270** comprising a complex vector of length M complex values is passed to unitary orthogonal inverse transform step **284**. As previously described, decimation step **270** and unitary orthogonal inverse transform **284** together form an optimized replacement of unitary orthogonal inverse transform **114** in FIGS. **5**A, **5**C and **5**D in some embodiments. Also as previously described, while many possibilities exist for the unitary orthogonal inverse transforms **114** and **132** as mentioned earlier, in the preferred embodiment the unitary orthogonal inverse transform is an inverse discrete Fourier transform. Two possible forms for unitary orthogonal inverse transform **284** exist, comprising one form with in order outputs and another form with bit-reverse ordered outputs. It is a feature of the current method that the output ordering of vector **290** in monotonically increasing integer layer index may be formed either by: (a) using integer scaling layer encoding in step **46** and unitary orthogonal inverse transform **284** with in order outputs; or (b) using scaled bit reverse layer encoding in step **46** and unitary orthogonal inverse transform **284** with bit-reverse ordered outputs. As a result, re-ordering of the outputs of unitary orthogonal inverse transform **284** in vector **290** can be accomplished by a change to the unitary mapping used for encoding layers in the RS pair of target wavefunction superpositions applied in step **46**.

The output of unitary orthogonal inverse transform step **284** is a vector **290** of M real or complex values, depending on the application. In the case where the M input vectors each of length P samples are real, the output vector **290** may be M real values equal to the dot product of the probe input sequence vector and each of the target sequence vectors. In the case where the M input vectors each of length P samples are complex, the output vector **290** may be M complex values equal to the dot product of the probe input sequence vector and each of the target sequence vectors.

From the foregoing description of the method **222** shown in FIG. **5**J, the total number of addition and multiplication operations required by the method **222** may be calculated and compared to the number of operations required by the method of matrix multiplication shown in FIG. **5**I. For the purposes of comparison in FIG. **5**N, a matrix **216** in FIG. **5**I will be encoded in a single RS pair of wavefunction superpositions, **110** and **128** in FIG. **5**J. Accordingly, each column vector in matrix **216**, with total of line **515** parameter M columns, is encoded as a separate layer in the RS pair, with total of line **525** parameter M layers. The length of the row vector **212**, line **505** parameter P, is equal to the length of the column vector in matrix **216**, line **510** parameter P. Each of the two wavefunction superpositions in the RS pair is a separate complex vector of length line **520** parameter Q complex pairs.

FIG. **5**K illustrates a flowchart of a method **223** for the interference of two superposition wavefunctions and reading of a plurality of orthogonal layer products between elements of each of the two superposition wavefunctions. The specific combination of products is determined by the arrangement of layers encoded in each superposition wavefunction, such that the output location of each of the products is a function of the layer locations of each input element multiplicand comprising the product. The properties of method **223** may be used to calculate combinatorial functions such as calculating a product score for different combinations of elements. An important use of method **223** is in the calculation of a tensor product of two matrices and is described below with reference to FIG. **5**L.

According to the method **223** depicted in FIG. **5**K a first wavefunction superposition RS pair designated 'i' and shown as comprising superposition R complex vector **231**, and superposition S complex vector **233**, is interfered with a second wavefunction superposition RS pair designated 'j' and shown as comprising superposition R complex vector **235**, and superposition S complex vector **237** in a process that starts with complex conjugation of the RS pair designated 'j' by steps **239** and **241**. The RS pair 'i' R wavefunction superposition complex vector is multiplied by the complex conjugate of its RS pair 'j' R wavefunction superposition counterpart by vector complex multiply step **243**.

Similarly, the RS pair 'i' S wavefunction superposition complex vector is multiplied by the complex conjugate of its RS pair 'j' S wavefunction superposition counterpart by vector complex multiply step **245**.

Since the S superpositions are layer encoded with the negative delta of the same pair's corresponding R superposition layers, the S interference product complex vector output from step **245** is complex conjugated by step **247** and is then combined by vector complex add **249** with the R interference product complex vector output from step **243**.

The output of step **249** passes as a complex vector to decimation step **270** which may be the same as the previously described step **270** used in method **222** depicted in FIG. **5**J.

As described above decimation step **270** uses addition of real and imaginary components of complex values in separate paths to create an output of length equal to the number of wavefunction superposition layers that have been encoded into target wavefunction superposition R and target wavefunction superposition S. In other words there is one complex value output by decimation step **270** for each of the M input vectors of length P that were encoded to the target wavefunction superpositions R and S.

The complex additions in decimation step **270** may be arranged in a number of stages where the decimation factor d is equal to two to the power of the number of stages, d=2^stages. In this case the length of each stage progressively halves, to end equal to the output length M. As depicted in FIG. **5**K an input complex vector **272** which in this example is eight times M in length is first reduced by stage **276** comprising four times 2·M additions to a complex vector of four times M in length. This is next reduced by stage **278** comprising two times 2·M additions to a complex vector of two times M in length and finally by stage **280** comprising 2·M additions to a complex vector **274** of M in length.

The output of decimation step **270** comprising a complex vector of length M complex values is passed to unitary orthogonal inverse transform step **284**. As previously described, decimation step **270** and unitary orthogonal inverse transform **284** together form an optimized replacement of unitary orthogonal inverse transform **114** in FIGS. **5**A, **5**C and **5**D in some embodiments. Also as previously described, while many possibilities exist for the unitary orthogonal inverse transforms **114** and **132** as mentioned earlier, in the preferred embodiment the unitary orthogonal inverse transform is an inverse discrete Fourier transform.

The output of step **284** is the real or complex vector **292** comprising the interference between the first wavefunction superposition RS pair T and the second wavefunction superposition RS pair 'j' by method **223**.

FIG. **5**L illustrates a flowchart of a method **293** for differentially encoding elements from two input sequences to two superposition wavefunctions, combined with the method shown in FIG. **5**K to calculate the tensor product of the two input sequences.

A first input sequence of elements is depicted as an input **227** comprising a plurality of sequence elements arranged in a plurality of layers shown in FIG. **5**L in which each layer is graphically represented as a separate box. Shown here for the purpose of illustration, the input **227** comprises an occupied layer with the sequence elements 'a' followed by three empty layers, followed by an occupied layer with the sequence elements 'b' followed by three empty layers, followed by an occupied layer with the sequence elements 'c' followed by three empty layers, followed by an occupied layer with the sequence elements 'd' followed by four empty layers.

The input **227** passes to an encode target superposition step **294** which comprises the method of FIG. **4**A for encoding a target or database sequence of elements to wavefunction superposition RS pairs comprising an R superposition wavefunction with positive delta layer encoding and an S superposition with corresponding layers with negative delta layer encoding. The output of step **294** is the RS pair(i) **295**.

A second input sequence of elements is depicted as an input **229** comprising a plurality of sequence elements arranged in a plurality of layers shown in FIG. **5**L in which each layer is graphically represented as a separate box. Shown here for the purpose of illustration, the input **229** comprises four occupied layers with the sequence, elements 'e', 'g', 'h' followed by a sequence of empty layers. The sequence of empty layers in input **229** should at least as long as the position of highest occupied layer of input **227**. Accordingly, the input **229** will be sufficiently long for all necessary combinations of occupied layers in each of input **227** and input **229**.

The highest occupied layer of input **227** plus the highest occupied layer in input **229** determine the position of the highest combination and therefore the length of the interference output **292** of method **293**. As illustrated in FIG. **5**L the highest layer in input **227** is the sequence element 'd' which is 12 layers above the lowest occupied layer with sequence element 'a'. In input **229** the highest layer is the sequence element 'h' which is 3 layers above the lowest occupied layer with sequence element 'e'. Consequently, in this illustration the highest combination is 15=12+3 and therefore the length of the interference output **292** is 16=15+1 to accommodate all combinations layers from 0 to 15. In FIG. **5**L the combination of sequence element 'a' with sequence element 'e' is shown in the lowest position output interference result 'ae' in output **292**, and the combination of sequence element 'd' with sequence element 'h' is shown in the highest position output interference result 'dh' in output **292**

The input **229** passes to an encode target superposition step **296** which comprises the same method of FIG. **4**A for encoding a target or database sequence of elements to wavefunction superposition RS pairs comprising an R superposition wavefunction with positive delta layer encoding and an S superposition with corresponding layers with negative delta layer encoding. The output of step **296** is the RS pair(j) **297**.

The first prepared wavefunction superposition RS pair(i) **295** is interfered with the second prepared wavefunction superposition RS pair(i) **297** by step **298** comprising the previously described method **223** depicted in FIG. **5**K.

The output of step **298** is a real or complex vector **292** comprising a pattern of different combinations of the input **227** sequence of elements with the input **229** sequence of elements.

For the purpose of illustration the method depicted in FIG. **5**L shows an arrangement that embodies a tensor product calculation between a first 2×2 real or complex matrix with elements 'a', 'b', 'c', 'd' and a second 2×2 real or complex matrix with elements 'e', 'g', 'h' to produce an output 4×4 real or complex matrix with elements 'ae', 'ag', 'ah', 'be', 'bf', 'bg', 'bh', 'ce', 'cf', 'cg', 'de', 'dg', 'dh'.

The real or complex vector **292** comprising the pattern of different combinations of the input **227** sequence of elements with the input **229** sequence of elements is presented

as the output tensor product **299**. For the purpose of illustration in FIG. **5**L this is shown as the **16** different combinations being mapped to the elements of a 4×4 real or complex matrix.

To provide a comparison of the novel method **293** for tensor product calculation versus the method shown in FIG. **5**I, input **227** represents the input matrix **213** depicted in FIG. **5**I; input **229** represents the input matrix **217** also depicted in FIG. **5**I; and output **299** corresponds to output **219** depicted in FIG. **5**I.

FIG. **5**M illustrates a flowchart of a method **225** for the interference of two superposition wavefunctions and reading and summation of a plurality of orthogonal layer correlations between elements of each of the two superposition wavefunctions, with an output comprising a combined complex correlation between a plurality of orthogonal layers,

As previously described with reference to wavefunction encoding in FIG. **4**A, the qubit phase **37** is a complex phase rotation applied to an RS pair comprising an R wavefunction superposition and an S wavefunction superposition. The qubit phase may also be subsequently changed to a new value by the same delta phase rotation being applied to the R wavefunction superposition and the S wavefunction superposition by a complex multiplication of each complex vector element by a complex number equal to the qubits phase change angle, eg. complex rotation=$(\cos(\theta), i\cdot\sin(\theta))$.

The complex correlation between two RS pairs allows two distinct wavefunction superpositions to be compared based on whether the layer locations of each RS pair share common elements and so may be correlated. It is a novel feature of the present method that a plurality of orthogonal dimensions may be evaluated in parallel depending on the layer data contents of each RS pair. There are two conditions that apply such that (a) if layer locations correspond in each RS pairs and (b) the layer data contents match or are similar, then (c) their mutual correlation metric contributes to the output. Equally, there is no systematic contribution to the output if the layers are different or data are orthogonal. In summary, the output **291** complex number of method **225** is a sum total of all the correlations at layers that are equal in each RS pair. As a result the output of this method provides a combined match metric for a plurality of data elements encoded in a plurality of layers, as both magnitude and phase represented by the complex number **291** for the sum total of correlation.

Qubit phase or 'spin' alignment is an important aspect of many quantum algorithms and therefore measurement of the qubit phase alignment between RS pairs is an essential step for such algorithms. The phase component of the complex number output **291** of method **225** is equal to arctangent (imaginary/real) and the magnitude is equal to the square root of the sum of real squared and imaginary squared. The magnitude associated with output **291** may be used as a factor in the quantum spin algorithm or as a qualification for application of the phase component. A phase component is generally not significant if the magnitude is below a non-zero threshold with the precise threshold value appropriate for the usage case.

An important qubit phase or 'spin' alignment algorithm is a generalized solve process described below as a preferred embodiment in method **1700** and shown in FIG. **17**. The generalized solve problem has problem constraints determined by a set of coupling factors **1710** between RS pairs of wavefunction superpositions **1705**. The solve process evaluates the RS pair interaction versus the coupling factor constraints and evolves the set of RS pairs towards an optimal solution by rotation of the qubit phase of selected RS pairs. The interference of RS pairs to measure their qubit phase difference by method **225** is used in an input to a separate sampling step that selects the phase rotation applied to each RS pair according to the particular solve algorithm being employed.

In the case of a generalized 'spin glass' quantum spin alignment algorithm the coupling factors determine the desired or lowest energy qubit phase relationship between each RS pair. For example a +1 coupling factor between two RS pairs may indicate qubit phase alignment, or a delta qubit phase of zero equal to arccosine(+1), while a −1 coupling factor between two RS pairs may indicate qubit phase opposite-alignment, or a delta qubit phase of 180° or π radians equal to arccosine(−1).

In the case of the generalized 'four color' problem that has been formulated as a quantum spin alignment algorithm, the present method has the novel feature of using a mapping of four arbitrary colors to four separated qubit phases such as 0°, 90°, 180°, 270°. the coupling factor between RS pairs may indicate whether the two 2D map regions, each represented by one of two RS pairs, are connected in which case the coupling factor may be −1, indicating the regions must not have the same color, or equivalently in this algorithm, the same qubit phase. Alternatively, if the two 2D map regions, each represented by one of two RS pairs, are not connected the coupling factor may be 0, indicating the regions may have the same color, or qubit phase.

According to the method **225** depicted in FIG. **5**M a first wavefunction superposition RS pair designated 'i' and shown as comprising superposition R complex vector **231**, and superposition S complex vector **233**, is interfered with a second wavefunction superposition RS pair designated 'j' and shown as comprising superposition R complex vector **235**, and superposition S complex vector **237** in a process that starts with complex conjugation of the RS pair designated 'j' by steps **239** and **241**. The RS pair 'i' R wavefunction superposition complex vector is multiplied by the complex conjugate of its RS pair 'j' R wavefunction superposition counterpart by vector complex multiply step **243**. Similarly, the RS pair 'i' S wavefunction superposition complex vector is multiplied by the complex conjugate of its RS pair 'j' S wavefunction superposition counterpart by vector complex multiply step **245**.

According to the present method a wavefunction superposition RS pair may be rotated by a qubit phase that is applied as a complex rotation equally to the R and S superpositions in the RS pair. As a result the complex correlations of the R(i) and R(j) are both independent of, but also in phase with, the complex correlations of the S(i) and S(j). As a result the R interference product complex vector output of step **243** and the S interference product complex vector output of step **245** are combined directly by vector complex add step **249**.

The complex correlations of the R(i) and R(j) wavefunction superpositions are combined by step **249** with the complex correlations of the S(i) and S(j) wavefunction superpositions and pass as a complex vector to step **251**. The method **225** calculates the angular difference of the qubit phase of RS pair(i) minus the qubit phase of the RS pair(j) as the output complex correlation value **291**. In the case of method **225** the DC or zero frequency component of a unitary orthogonal inverse transform is required which according to the preferred embodiment is a vector complex sum **251** comprising the complex addition sum of all elements in the interference output complex vector.

FIG. **5**N shows a performance comparison **500** comprising a table of parameters and formulae for quantification of

addition and multiplication operations for the matrix multiplication method shown in FIG. **5**I and for the steps of method **222** shown in FIG. **5**J comprising the interference butterfly **232**, the decimation step **270** and unitary orthogonal inverse transform step **284**. The interference butterfly **232** has line **540** formula of 2Q addition and 4Q multiplication operations as previously described. The decimation step **270** has line **545** formula of 2Q addition operations as previously described. Unitary orthogonal inverse transform step **284** has line **550** formula of less than or equal to 3M·log 2(M) addition plus less than or equal to 2M·log 2(M) multiplication operations. The line **550** formula for the number of operations is an upper bound based on a radix 2 Discrete Fast Fourier Transform algorithm comprising log 2(M) stages each with M/2 butterfly operations each comprising two complex additions and one complex multiplication for a total of six addition and four multiplication operations per butterfly. More efficient Fast Fourier Transforms exist with fewer total operations than the line **550** formula, for example radix 4 algorithms.

The performance of the method **222** may be compared with matrix multiplication shown in FIG. **5**I by dividing the operations required by matrix multiplication for real line **530**, and complex line **535** data, by the total operations line **555** required by method **222** to calculate a quantum advantage factor for real matrix multiplication line **560** and complex matrix multiplication line **570** respectively. Where the method **222** is faster than the matrix multiplication shown in FIG. **5**I, the quantum advantage factor will be greater than unity, with the quantum advantage factor representing a speed up provided by method **222** compared to matrix multiplication shown in FIG. **5**I.

FIG. **5**N includes a Quantum Breakeven metric for Real Matrix Multiplication line **565** and Complex Matrix Multiplication Line **575**, each in units of wavefunction superposition layers M, such that the Quantum Breakeven M number is the minimum depth of superpositions required to breakeven with conventional matrix multiplication methods in terms of the computational cost in multiplication and addition operations. Each factor of two in the actual value of M, the number of layers in the RS pair superpositions, compared to the Quantum Breakeven value of M is equivalent to a factor two increase in performance relative to conventional methods for calculating matrix multiplication. For example, if the Quantum Breakeven is at M=4 and acceptable results are achieved at M=16 layers of superposition, then the Quantum advantage will be 16/4=4 times speed up. The quantum breakeven is calculated for the maximum size of real or complex vector in order to remove size as a factor. As previously described, a vector of P real values may be encoded to a layer in a wavefunction of P=Q complex values. A vector of P complex values may be encoded to a layer in a wavefunction of **2**P=Q complex values. The useable number of layers M in a wavefunction superposition RS pair will increase from the breakeven M for shorter real and complex vectors.

FIG. **5**N includes a comparison factor based on memory data bandwidth of the method **222** versus the conventional matrix multiply. Quantum Advantage Data Size line **580** is the size of the conventional (P×M) matrix as real or complex data values required for matrix multiplication, relative to the size of the wavefunction superposition RS pair. Each wavefunction superposition comprises Q complex values. In the performance comparison **500**, the RS pair is an encoded wavefunction representation of line **515** the M column vectors of the matrix **216** depicted in FIG. **5**I for matrix multiplication. Wavefunction superposition of M layers line **525** achieves a data compression by combining M column vectors in matrix **216** in two complex vectors each of length Q complex pairs, line **520**, equal to four Q reals. The Quantum Advantage Data Size line **580** shows an advantage of M times P divided by four Q for real matrix multiplication and an advantage of M times P divided by two Q for complex matrix multiplication.

The overall quantum advantage comprises the quantum advantage in operations count for addition and multiplication lines **560**, **570** and reduced memory bandwidth represented by the quantum advantage data size line **580**. It is a feature of the current method that comparable Quantum Advantage speed up factors apply to either the real matrix multiply operations line **560** or the complex matrix multiply line **570** versus the data size line **580**, so providing a speed up to all systems that is independent of their relative compute versus memory performance. In general, compute and memory bandwidth advantages can factor together to synergistically increase performance.

Multi-Resolution Database

FIG. **6** illustrates a multi-resolution database **900** and the segmentation of a target sequence into multiple sections (designated as $S_0 \ldots S_{127}$), as was mentioned above. Multi-resolution database **900** may be optionally used with the searching method discussed above. However, in other embodiments, different forms of data structure or databases may be used to store the encoded probe and/or target used with the searching methods earlier discussed. For purposes of illustration, genome **101** is shown here as divided into 128 sections. In other embodiments, other numbers of sections may be used.

As discussed above, the multi-resolution database **900** may be organized in a number of different tracks **902** of varying resolutions. Each track **902** corresponds to a different superposition layer depth. For example, Track 1 corresponds to a preparation of all sections using superpositions that are 32 layers of data deep (as was discussed above).

A layer encoding index **904** may be applied to each of these sections for each track **902** of the database as illustrated. For example, the superpositions created for data in sections S_0 and S_1 of Track 0 are encoded with layer index 0, and sections S_2 and S_3 are encoded with layer index 1. For Track 6, since it is only one layer deep, it should be noted that only one set of superpositions R is needed, since the superpositions S will be identical for one-layer depth.

FIG. **7** illustrates the exemplary further segmentation into individual frames (e.g., $f_0, f_1, f_2, \ldots$) of the sections of Track 0 of multi-resolution database **900**. Each section S from database **900** may be organized as a number of frames. The number of frames in each section may be determined, for example, by dividing the total base length of the target by the number of sections (e.g., 128) multiplied by the frame base length (e.g., 2,048 bases).

For example, for a target length of 25,165,824 bases, a total section number selected to be 128 sections, and a frame length selected to be 2,048 bases, the number of frames per section is 96. Thus, frame group **950** corresponding to section So will have 96 frames, and frame group **958** corresponding to section S_1 will have 96 frames.

In one approach, a superposition wavefunction R, S pair (ψR, ψS) is calculated using a frame selected from the same relative position for each layer encoding index **904**. For example, ψR and ψS are calculated using a superposition of 64 wavefunctions, with each wavefunction corresponding to frame f_0 selected from the frame group **950**, **954**, . . . , **956**

as indicated by the dashed lines at **952**. A (ψR, ψS) pair is next calculated for each frame f_1, and so on for each corresponding frame position for Sections S_0, S_2, . . . , S_{126}. The same approach is also applied to frame groups **958**, **960**, . . . , **962** for Sections S_1, S_3, . . . , S_{127}.

To maintain database continuity, it is typically necessary and desirable to remove boundary effects or discontinuities, which may occur as explained below. To accommodate this, there will be an additional overlap frame that is present in each track of more than one-layer deep. The result of the calculations in the simple exemplary case illustrated above will be a total of 1+96+96=193 (ψR, ψS) pairs for Track 0. These calculations may then be done for each track **902** so that database **900** in general contains a set of (ψR, ψS) pairs for each track **902**, which can be selected and used for searching with a probe as earlier discussed.

As referred to above, overlapping of successive frames creates VCA_R and VCA_S data that is independent of the probe sequence occurrences in the target sequence, even when the probe sequence is contained in two separate, but consecutive wavefunction frames input to steps **110** and **128**. The special case where the probe sequence crosses a section boundary is generally dealt with in database **900** by making each track comprise successively ordered sections: the last frame of section 0 is followed by the first frame of section 1, and within all tracks **902** shown in the exemplary case section 0 is followed by section 1.

There is, however, a separate worst case where the probe sequence may cross a section boundary that lies at the end of a track layer. An example of such a case would be a probe that had its first half as the last three bases in section **63** and its second half as the first three bases of section **64**. This worst case can be made equivalent to a more normal case by a formatting of database **900** so that duplicates of the last frame of the endmost sections of each track are encoded as an extra frame at the beginning of the next layer of the superposition. For the first section 0, the place of this overlap is filled by a frame of zero data for the purposes of the frame-aligned summation that creates the superposition of layers. Therefore, in database **900** in the exemplary case, the successive layers are arranged so that the section at the end of the track in one layer is followed in the original input information sequence by the first section of the next layer in the superposition. This allows the section boundary to be made transparent as far as output position index such that the same sequence position is represented by the following information: (start of next layer−x) and (end of previous layer+N−x), where x is less than N and the start and end positions are the chromosomal target position indices of the first base in each of the corresponding frames.

In the worst case example where a probe sequence is present in the target with its first half as the last three bases in section **63** and its second half as the first three bases of section **64**, there would be a hit detection at a position equal to the starting position of section **64** minus 3 bases. This hit would be generated by VCA_R and VCA_S data that combines the overlap frame from the end of section **63** as a first frame, with the first frame of section **64** as a second frame. As indicated previously, vector elements N to 2N−1 in the second half of a second frame output from step **114** are added using vector complex add step **118** to the corresponding elements 0 to N−1 in the first half of the first output frame, which was previously stored in overlap buffer **116**.

In one example of the above, the database is created to maintain continuity across more than 10,000 frames of the frames in the target sequence.

Searching Process and Interaction with Multi-Resolution Database

FIG. **8** illustrates the preparation of target data and certain interactions that may occur with multi-resolution database **900**. A target sequence is read in step **210**. A map table may be created that records continuous runs of zeroes and gaps in the target sequence. The runs of zeroes and gaps may be identified for the purpose of removing them from the data prior to preparing database **900**. The locations may be later used when interpreting the search results and determining match positions in the target sequence. A unit size is calculated to partition the entire target sequence (for example, the number of frames per section discussed above) for forming database **900**.

In step **220**, the target may be encoded to intermediates as illustrated in FIG. **4**A. In step **240**, the intermediates may be encoded to form multi-resolution database **900**. Different tracks may be output from database **900** with one or more channels. An R superposition plus an S superposition is an example of two channels of wavefunction frame data, and the final track 6 of database **900** is an example with one channel of wavefunction frame data. The different tracks correspond to different depths of superposition as discussed above.

It is typically desired to want to exploit the maximum number of layers of superposition can be searched in parallel in any one particular instance in order to have the maximum search efficiency. The layer depth of superpositions that can be processed typically depends on the number of defined bases in the probe sequence. The longer the probe, the greater layer depth of superpositions that can be used in doing the search because of the greater uniqueness of the probe.

For purposes of explanation, the layer depth limit is related to the signal-to-noise ratio (SNR). A longer probe will have a higher SNR than a shorter probe. For each length of probe, there is typically a superposition layer depth that can be effectively processed. It is desired to have a database that has different layered depths in it that may be selected as optimal for different probes. For example, the database will have greater layer depths for longer probes, shorter layer depths for shorter probes, and for the shortest probe, it may have a depth of only a single layer.

In step **260**, a non-encoded target sequence track may be created. This track is separate from, but may be synchronized on a frame basis, with the wavefunction frames in other tracks. As a result, this non-encoded target sequence track may be more efficiently accessed for any wavefunction frame by using the corresponding frame counter or index. This non-encoded track may include position dependent annotations at various positions along the target sequence and/or hyperlinks that may present additional information to a user or permit a user to interactively link to additional information about various positions on the target while interacting with and interpreting search results (e.g., while viewing results on user display **205**).

FIG. **9** illustrates the selection of a track **902** from multi-resolution database **900**, the searching for hits in the selected track, and the selection of a next track **902** at a different resolution for continued searching. A DNA/RNA probe sequence is prepared in step **305** by transforming to wavefunction form as discussed above. Alternatively, in step **310**, a previously prepared probe sequence wavefunction may be obtained from storage (e.g., memory) for use, or combinations of probe sequences may be used to prepare customized wavefunctions for particular applications.

The probe sequence selected for use is passed to step **320**, in which the expressed length of the probe is calculated. The expressed probe length is used to calculate the normalization scale factor applied during the searching process discussed above in modulus and normalization steps **122** and **140**. Normalization is desired to attempt to make the output correlation peaks independent of the probe length.

In step **330**, the maximum superposition layer depth is selected based on the expressed length of probe from step **320**. In step **340**, a track **902** is selected from multi-resolution database **900** corresponding to the maximum superposition layer depth that may be efficiently processed for the effective length of the probe. Then, a search as described above in FIG. **5** for method **100** is performed.

In step **350**, a search hit (e.g., an R, S Hit Pair) from executing method **100** is output with its track position and the layer index of the hit. Processing of the track continues in step **360** to determine if additional search hits will be found. For example, anywhere from one to millions of hits may be found from the search of the first selected track **902**.

In step **370**, if the process is at the end of the selected track and at least one search hit has been found, then the process ends in step **390**. If no search hits were found, then in step **380** another superposition layer depth is selected (e.g., half the layer depth just used in the failed search), and the corresponding track **902** is obtained from database **900** for additional processing using method **100** at step **340**.

FIG. **10** illustrates the use of a single-layer track in the multi-resolution database to confirm one or more search hits located from one of several multiple-layer tracks **902** in the multi-resolution database **900**. A probe input is selected in steps **420** and **425** are similar to steps **305** and **310** discussed above, and a probe length is determined in step **430** similarly as discussed above in step **320**. In step **410**, the one or more search hits from step **350** are used as an input to step **440**.

In step **440**, the track position and layer index of each incoming search hit are converted to a target sequence offset. In other words, the track position can be converted to a sequence offset of the hit within the original input sequence. As discussed above, in step **182** the layer index may be converted to a target section number. The track position may then be calculated as the count of frames from the track start plus the length of a single target section times the target section number.

In step **450**, the process determines a position in the target sequence that is upstream of the search hit's offset in the single-layer track **902** (e.g., Track 6 of FIG. **6**) of the multi-resolution database **900**. Step **450** is performed to attempt to confirm that a hit actually exists in the original target location identified from the search of method **100**.

Simply stated, a separate process is started for each potential search hit that is generated from method **100** of FIG. **5**. A search hit is processed again over a range (referred to below as a "window") bounded by a small positional distance upstream from the search hit's position in the target sequence to a small positional distance downstream (e.g., this may be just a couple of frames). Each search hit may be confirmed by checking the single-layer track of database **900** (note that there is no superposition in the single-layer track). It should be noted that in the single-layer track of database **900**, dual R, S data is not needed. This is so because, if the separation between the R and S data is twice the layer index and layer index zero is being used, then the R and S data are identical and only the R or S data is needed.

In step **460**, the single-layer track is processed over the window that is determined by the primary frame that contains the position of the hit, plus an additional downstream (successor) frame if the position within the frame is such that part of the matching sequence is in that downstream frame. In addition it may be desirable to start processing with the frame that is immediately upstream (preceding) the frame that contains the hit in order to provide an overlap frame at steps **116** and **134** that will be combined via steps **118** and **136** with the primary frame that contains the position of the hit. The prepared probe sequence from step **420** or **425** is used here again, in addition to the search that generated the search hit, because the interference process is basically being repeated on the single-layer track as a confirmation of the search hit.

In step **470**, if the hit position determined from the single-layer track is equal to the input search hit position, then the process ends at step **480**. If the hit position is not equal to the input position, then in step **475** the single-layer track is processed further. If the hit position is upstream (preceding) the input position, processing will continue until the input position has been reached and passed. The processing of a single wavefunction frame will yield a full N potential match alignments, so even the minimum window equal to a single frame will detect all the potentially multiple match alignments that may be present in the target sequence information inside this window. In this way, all of the multiple hits will be detected based even a single hit occurring an a superposition layer interference as in method **100**. Processing of the single-layer track continues until the end of the window being processed is reached at step **490**.

FIGS. **11**A-**11**D illustrate a flowchart of method **1100** for sequence searching, in accordance with some implementations. In particular, some implementations of method **700** illustrate optimizations to any of the previously described methods for the special case when the probe has a fixed length and the target includes a plurality of potential matches all having the same fixed length. This situation is described elsewhere in this document as a target having a natural modulus. It is also described as a "zero-offset use case," because a priori knowledge of the length of the potential matches allows the target data to be structured in such a way as to guarantee a zero-offset result within a given matching layer. For example, consider a list of United States phone numbers. Each phone number, including area code, has 10 digits. So, in this example, the natural modulus is 10. Moreover, when searching for a match to a specific phone number within that list (e.g., a probe phone number), a match that starts in the middle of one phone number and overlaps onto the next phone number, in this situation, is not a match at all. Thus, the offset in such a use case is always zero because only matches that begin at the beginning of a phone number are considered. Such a priori knowledge obviates the need for certain steps and operations described elsewhere in this document. In particular, such a prior knowledge obviates the need for a storage buffer and leads to optimizations in how the target sequence is stored, as described below.

It should be understood that the operations in method **1100** may be performed by a single computer system or may be divided among several computer systems. For ease of explanation, method **1100** is described as being performed by a single computer system (e.g., computer system **1200**, FIG. **12**).

The computer system stores (**1102**) a first probe sequence representation expressed in a first orthogonal domain. The first probe sequence representation is characterized by a length (e.g., a natural or predefined length). In some embodiments, the first probe sequence representation is (**1104**) a vector of real or complex numbers. Considering the example

illustrated above, a natural or predefined length for a United States phone number, including area code, is 10 real numbers. Thus, in some embodiments, method **1100** can be used for lookup in a reverse phone book.

In some embodiments, the first probe sequence representation comprises a plurality separately searchable component symbols encoded as sequential vectors of real or complex numbers. For example, considering a sequence of playing cards: spades could be encoded with a value of "1"; hearts could be encoded with a value of "−1"; diamonds with a value of "j" (where j is the imaginary unit); and clubs with a value of "−j." In this sense, component symbols are encoded as real or complex numbers, and an encoded sequence of such component symbols comprises a sequence of real or complex numbers. The domain in which the probe sequence representation is initially expressed in (i.e., the first orthogonal domain) is sometimes referred to as the spatial domain. But in many applications, this designation is purely arbitrary.

The computer system stores (**1108**) a first target sequence representation expressed in the first orthogonal domain. The first target sequence includes a plurality of potential probe match sequences each characterized by the length. In some embodiments, the process of encoding the potential probe match sequences into the first target sequence representation is analogous to the process described with reference to FIG. **4**A, with the difference being that the sequence does not necessarily comprise DNA or RNA nucleotide bases.

The computer system transforms (**1110**) the probe sequence representation and the target sequence representation into a second orthogonal domain to produce a second probe sequence representation and a second target sequence representation, respectively. The second orthogonal domain is expressible using a basis set that is orthogonal to a basis set of the first orthogonal domain. In some embodiments, transforming (**1112**) the probe sequence representation and the target sequence representation into the second orthogonal domain comprises applying a first orthogonal domain unitary transform to the probe sequence representation and the target sequence representation, respectively. In some embodiments, the first orthogonal domain unitary transform is (**1114**) a Fourier transform (e.g., a computer-implemented Fourier transform and/or a discrete Fourier transform). Such unitary orthogonal transforms are described further with reference to step **24** (FIG. **3**A).

In some circumstances, representations expressed in the second orthogonal domain are referred to herein as "wavefunctions." Such wavefunctions can be superimposed (e.g., by vector addition). The result of a superposition of one or more wavefunctions (i.e., representations in the second orthogonal domain) is variously referred to as: "superpositions"; "superposition wavefunctions"; or "superimposed wavefunctions."

The computer system encodes (**1116**) the second target sequence with a first plurality of modulation functions in the second orthogonal domain, each of the first plurality of modulation functions having an integer position index corresponding to one of the potential probe match sequences, thereby producing a first plurality of encoded second target sequence representations. Each of the first plurality of encoded second target sequence representations is described herein as a "layer." For example, considering a list of 10 phone numbers, each phone number is transformed into the second orthogonal domain, and modulated (e.g., encoded) with a modulation function (e.g., a transformed delta function), to form a layer. The layers are added together to form a superposition, which represents the list of 10 phone numbers. In some circumstances, this encoding is considered a convolution of each of the potential matches with a unique delta function in the first orthogonal domain. The convolution, however, is performed in the second orthogonal domain.

As noted above, in some embodiments, the modulation functions are delta functions. Thus operation **1116** proceeds similarly to operation **46** (FIG. **4**A). However, several optimizations are optionally applied for the zero-offset use case, in accordance with some embodiments. For example, in some embodiments, the number of useable encoding layers may be increased when a multiplication factor is applied to the separation of the layer index encoded as delta function position index step **46** of FIG. **4**A. This increase in the number of useable encoding layers means that deeper superpositions comprising more layer-encoded representations (i.e., "wavefunctions") can be used and so more target representations can be searched in parallel, thereby increasing performance. The increase in the number of useable encoding layers occurs because the multiplication factor applied to the separation of the layer index encoded as delta function position index step **46** of FIG. **4**A decreases the signal overlap between different layer encoded representations.

Continuing with method **1100**, the computer system interferes (**1118**) the first plurality of encoded second target sequence representations with the second probe sequence representation to produce one or more interfered sequence representations. This operation proceeds in an analogous fashion to steps **112** and **130** (FIGS. **5**A-**5**D). To that end, in some embodiments interfering the first plurality of encoded second target sequence representations with the second probe sequence representation comprises (**1120**) superimposing the first plurality of encoded second target sequence representations (i.e., to generate the wavefunction) and interfering the superimposed encoded second target sequence representations (i.e., the wavefunction) with the second probe sequence representation. In some circumstances, this operation is considered a convolution of the probe sequence and the first plurality of encoded second target sequence representations (which is stored as a wavefunction superposition). The convolution is carried out in the second orthogonal domain, e.g., by performing (**1122**) a vector multiply operation between the plurality of encoded second target sequence representations and a complex conjugate of the second probe sequence representation.

In some embodiments, the encoding is a dual encoding, meaning that in addition to the encoding described with reference to operation **1116**, the computer system encodes (**1124**) the second target sequence with a second plurality of modulation functions in the second orthogonal domain, thereby producing a second plurality of encoded second target sequence representations. Each modulation function in the first plurality of modulation functions has a positive integer position index and corresponds to a modulation function in the second plurality of modulation functions that has a negative integer position index with the same magnitude as the positive integer position index. This aspect of the present disclosure is discussed with reference to the preparation of target wavefunction superpositions R and S, respectively (e.g., step **110** and **128**, FIG. **5**C). Likewise, the computer system interferes the second plurality of encoded second target sequence representations with the second probe sequence representation to produce one or more corresponding interfered sequence representations (which is described with reference to steps **112** and **132**, FIG. **5**C). The computer system further combines (**1128**) each interfered

sequence representation with a conjugate of the corresponding interfered sequence representation, as shown with reference to step **186** (FIG. **5**D). An alternative to the conjugate approach is the reflection of data about zero lag, as described with reference to step **183** (FIG. **5**C).

The computer system obtains (**1130**) an inverse transform result characterizing a respective integer position index from a respective interfered sequence representation. In some embodiments, the inverse transform result is (**1132**) obtained from the combination of the interfered sequence representation and the corresponding conjugate interfered sequence representation (e.g., because they have been combined per step **1128** to enhance the signal-to-noise). In some embodiments, obtaining the inverse transform includes applying (**1134**) a second orthogonal domain unitary transform to the one or more interfered sequence representations. The second orthogonal domain unitary transform is an inverse of the first orthogonal domain unitary transform (e.g., an inverse discrete Fourier transform). The result is a convolution in the first orthogonal domain. This operation is analogous to step **114** (FIG. **5**A) except that step **114** can be optimized when the encoded delta function indices of consecutive layers are separated by a multiplication factor (as described above). Namely, the unitary orthogonal inverse transform step **114** can be modified to only calculate output data corresponding to the zero position offset and the encoding superposition layer. Using such a modified unitary orthogonal inverse transform step **114**, the number of output data calculated will be a fraction of the unmodified unitary orthogonal inverse transform step **114** output, where this fraction is the reciprocal of the multiplication factor applied to the separation of the layer index encoded as delta function position index step **46** of FIG. **4**A. Thus, in some embodiments, step **114** in FIGS. **5**C-**5**D and FIG. **5**J should be understood to be the modified version of this step.

When the convolution represents a hit, it will have a strong delta signal in the first orthogonal domain. On the contrary, a convolution between a probe and a non-hit will not produce a delta signal. Thus, in some embodiments, the computer system selects (**1136**), as the inverse transform result, a result of the second orthogonal domain unitary transform applied to the one or more interfered sequence representations at a position corresponding to the respective integer position index.

To be sure that the selected result represents a true hit, the computer system determines (**1138**) whether the inverse transform result exceeds a predefined threshold. In accordance with a determination that the inverse transform result exceeds the predefined threshold, the computer system outputs (**1140**) information indicating that the respective integer position index represents a match between the probe sequence representation and the corresponding one of the potential probe match sequences. In accordance with a determination that the inverse transform result does not exceed the predefined threshold, the computer system forgoes (**1142**) output of information corresponding to the respective integer position index.

FIG. **12** is a block diagram illustrating an example of a computer system **1200** for sequence search, in accordance with some embodiments. In some embodiments, computer system **1200** is analogous to, or shares any of the features of, computer system **200** (FIG. **2**). While certain specific features are illustrated, those skilled in the art will appreciate from the present disclosure that various other features have not been illustrated for the sake of brevity and so as not to obscure more pertinent aspects of the implementations disclosed herein. To that end, computer system **1200** includes one or more processing units or cores (CPUs) **1202**, one or more network or other communications interfaces **1208**, memory **1206**, optional transform hardware **1210** and one or more communication buses **1204** for interconnecting these and various other components. Communication buses **1204** may include circuitry (sometimes called a chipset) that interconnects and controls communications between system components. Memory **1206** includes high-speed random access memory, such as DRAM, SRAM, DDR RAM or other random access solid state memory devices; and may include non-volatile memory, such as one or more magnetic disk storage devices, optical disk storage devices, flash memory devices, or other non-volatile solid state storage devices. Memory **1206** may optionally include one or more storage devices remotely located from CPU(s) **1202**. Memory **1206**, including the non-volatile and volatile memory device(s) within memory **1206**, comprises a non-transitory computer readable storage medium.

In some implementations, memory **1206** or the non-transitory computer readable storage medium of memory **1206** stores the following programs, modules and data structures, or a subset thereof including operating system **1216**, network communication module **1218**, search module **1210**, transformation/encoding module **1212**, and database **1211**.

Operating system **1216** includes procedures for handling various basic system services and for performing hardware dependent tasks.

Network communication module **1218** facilitates communication with other devices (e.g., other server systems and/or client devices) via one or more network interfaces **1208** (wired or wireless) and one or more communication networks, such as the Internet, other wide area networks, local area networks, metropolitan area networks, and so on.

Search module **1210** is configured to perform various tasks described with reference to FIG. **11**A-**11**D as well as elsewhere in this document. For example, in some embodiments, search module **1210** encodes target sequences **1214** (e.g., target sequence **1214**-*a* through **1214**-*p*, which are stored in database **1211**) with modulation functions to produce a plurality of encoded target sequences. In some embodiments, search module **1210** interferes encodes target sequences **1214** with probe sequences **1216** (e.g., probe sequence **1216**-*a* through **1216**-*l*, which are optionally stored in database **1211**). To this end, search module **1210** includes a set of instructions **1210**-*a* and, optionally, metadata and heuristics **1210**-*b*.

In some embodiments, search module **1210** calls transformation/encoding module **1212** to perform various transformations, such as the unitary orthogonal transformations described elsewhere in this document (other modules, not shown, may also call transformation/encoding module **1212** as necessary). In turn, in some embodiments, transformation/encoding module **1212** optionally utilized specialized hardware, such as transform hardware **1210** to perform calculations necessary to perform the unitary orthogonal transformations. In some embodiments, transform hardware **1210** is or is similar to a video card. In some embodiments, transform hardware **1210** includes its own firmware for performing such calculations. To perform these and other tasks, transform/encoding module **1210** includes a set of instructions **1212**-*a* and, optionally, metadata and heuristics **1212**-*b*.

FIG. **13** illustrates a method **1300** for encoding different tag data types as first sequence representations in the first orthogonal domain, viewed as an x-y plot of data in the complex plane where x is the real axis **1302** and y is the

imaginary axis **1304**. It is a feature of the present method that many different data types may be represented, for example binary sequences, real and complex sequences, symbol sequences and cryptographic hashing codes as unique identifiers, with data in one or any number of dimensions. It is an additional feature of the present method that the different data types are mutually incompatible and independent so that they comprise different mathematical signal spaces in the first and second orthogonal domains. According to the present method, probe or search data and stored target data are each processed by encoding to first sequence representations in the first orthogonal domain. Different data types for probe and target are encoded by different symbols to create mutually incompatible and independent signal spaces in the first orthogonal domain. For example, binary data may be encoded as symbols at locations in the complex plane using coordinates defined according to the real and imaginary axes. Two bits may be encoded by the real and imaginary coordinates where a positive coordinate represents bit value '1' and an negative coordinate a bit value '0'. Accordingly, if the real coordinate encodes the lower bit and the imaginary coordinate encodes the upper bit in a bit pair, in FIG. **13**, symbol **1314** may encode a pair of bits with value '11' or 3, symbol **1316** may encode a pair of bits with value '10' or 2, symbol **1318** may encode a pair of bits with value '00' or 0, and symbol **1320** may encode a pair of bits with value '01' or 1, according to one embodiment. In such an embodiment it is possible to match each bit separately using symbols encoded in the positive real axis **1306** and negative real axis **1308** to test the lower bit and symbols encoded in the positive imaginary axis **1310** and negative imaginary axis **1312** to test the upper bit.

While testing individual bits is useful for binary data, it is less useful for a tag data comprising a cryptographic hashing code since individual bit differences are randomized. In the case of tag data comprising a cryptographic hashing code the binary data may be advantageously encoded as a sequence of complex symbols each representing a pair of binary bits. In such an encoding scheme a single bit error will cause mismatch in both the real and imaginary dimensions and therefore generate a more robust error metric than if a single bit error generated a mismatch in only one of either the real or imaginary dimensions.

Symbol **1306** may encode a complex value with a positive real value and zero imaginary value and symbol **1308** may encode a complex value with a negative real value and zero imaginary value. Similarly, Symbol **1310** may encode a complex value with a zero real value and positive imaginary value and symbol **1312** may encode a complex value with a zero real value and negative imaginary value. According to the present method data sequences may be encoded as complex symbols, for example the A,T,C,G nucleotide bases in DNA sequences may be represented by the symbols **1306**, **1308**, **1310** and **1312** respectively as shown in FIG. **13**. The previously described unitary encoding of the nucleotide base sequence to a vector in an orthogonal basis step **20** in FIG. **3**A was for such an example of a DNA sequence encoded as: A (1, 0), C (0, j), T (−1, 0), and G (0, −j).

According to the present method the type of tag data may include cryptographic hashing codes as unique identifiers. These hashing codes are generated as a signature or digest of a data record and are designed to not match if the data record contains any bit differences. According to one embodiment of the present invention tag data comprising cryptographic hashing codes is encoded as a sequence of complex number symbols. For example, symbol **1306** may encode a pair of bits with value '11', symbol **1310** may encode a pair of bits with value '01', symbol **1308** may encode a pair of bits with value '00', and symbol **1312** may encode a pair of bits with value '10'.

For many applications the tag data type is real or complex number sequences in one or more dimensions. In the case of complex number sequences each complex number is represented as a point relative to the real and imaginary axes in FIG. **13**. Real number sequences may be represented by a complex number sequence of half the length, wherein the real component is equal to even number indexed values and the imaginary component is equal to the odd number indexed values, which is generally known as an even extension. In FIG. **13** an example instance of the tag data type that comprises real or complex numbers is shown as a scatter or x-y plot so that the data is represented by a point cloud of coordinates 1322.

FIG. **13** also shows a rotation **1330** in the direction of the arrow by a qubit complex phase angle **1328** between a starting radial marker **1324** and an ending radial marker **1326**. The rotation **1330** is a polar coordinate transform comprising addition to the angular component or equivalently in Cartesian coordinates, complex multiplication by a point on the unit circle. According to the present method a qubit complex phase angle determines such a complex rotation and is applied to different tag data types as first sequence representations in the first orthogonal domain. As viewed as an x-y plot of data in the complex plane where x is the real axis **1302** and y is the imaginary axis **1304**, the previously encoded first sequence representations in the first orthogonal domain are encoded at a complex phase angle determined by the qubit phase. Accordingly, symbol **1306** is rotated by qubit with phase angle **1328** to the position of symbol **1332**, symbol **1308** is mapped to the position of symbol **1334**, symbol **1310** is mapped to the position of symbol **1336**, and symbol **1312** is mapped to the position of symbol **1338**. Similarly, symbol **1314** is rotated by qubit with phase angle **1328** to the position of symbol **1340**, symbol **1316** is mapped to the position of symbol **1342**, symbol **1318** is mapped to the position of symbol **1344**, and symbol **1320** is mapped to the position of symbol **1346**. The example encoding of complex and real sequences to point cloud **1322** will be transformed by rotation defined by a qubit with phase angle **1328** to new positions shown as the point cloud **1348**.

According to the present invention the rotation **1330** shown in the direction of the arrow of and determined by a qubit with phase angle **1328** may be applied to the first probe sequence representation and to the first target sequence representation in the first orthogonal domain. It is a feature of the unitary orthogonal transform used in current method that the complex phase of the first probe sequence representation and the first target sequence representation in the first orthogonal domain will be directly mapped without information loss to their representations in the second orthogonal domain. As a result the qubit complex phase angle may be used as a common way to modulate the storage and search for the different types of tag data that may be encoded by the present method.

It is a further feature of the current method that the complex phase rotation of angle **1328** shown as arrow **1330** may be applied directly to second target sequence representations in the second orthogonal domain, as described below.

FIG. **14** illustrates a method **1400** for generating a program **1420** of panvia items **1422** comprising commands **1424**, each with an action **1426** and a list **1428** of objects **1402** with combined tag data **1404**, content data **1406** and

qubit **1408** elements, that may be expressed in the form of a structured data document **1430** according to an exemplary embodiment of the present disclosure. In the present disclosure tag and tag data are used interchangeably. Furthermore, tag data **1404** comprises a type that specifies the data moiety for the data so that an extendable set of different data types may be supported. The data types described herein include binary sequences, real and complex sequences, symbol sequences and cryptographic hashing codes as unique identifiers, with data in one or any number of dimensions. In addition, the type of tag data **1404** may specify the function to be applied to the data so that an extendable set of different functions may be supported. In addition to general-purpose search using sequences of symbols and unique identifiers, the current method as described herein includes the functions of correlation, dot product, and neural net inference. Also, in the present disclosure, content and content data are used interchangeably. According to the method described herein, content may comprise structured data for any object. Furthermore, objects **1412** may comprise executable **1414** which is structured data for an object that comprises an instance of the structured data document **1430**. As a result of executable **1414** the method **1400** has recursive and self-referencing properties.

FIG. **15** illustrates a method **1500** performed by a server computer that receives commands from a client **1502**, processes the commands in the manner of a programmable quantum computer using an interference process **1526** in a common step for different commands and sends a response back to the client **1544** according to an exemplary embodiment of the present disclosure. The operating method of the programmable quantum computer is different from a conventional computer in that the working memory is in the quantum field domain, represented by the wavefunctions ψR and ψS, referred to as an RS pair, each comprising superpositions of layer encoded wavefunctions prepared from target sequence data in the form of a tag data. The RS pairs **1524** are retrieved from memory **1522** and combined by the interference process **1526** with probe wavefunctions sequence data prepared in step **1520** from probe sequence data in the form of a tag data contained in commands **1516** in command queues **1512**.

The commands received from client **1502** are processed by a Validate step **1504** that uses a schema **1506** as the rules for validation. In the preferred embodiment the commands are received in the form of Javascipt Object Notation (JSON) structured data via an Internet connection and Hypertext Transfer Protocol (HTTP). In the preferred embodiment schema **1506** is a JSON schema that contains rules that specify how the input JSON should conform to the constraints of the instruction set architecture. The Validate step **1504** will either allow the input commands to pass to step **1508** in the case that the conditions set by the rules of the schema **1506** were met, or alternatively step **1504** will reject the input commands **1502** as invalid if the schema **1506** rules are not satisfied. Validate **1504** adds a timestamp to each validated command for use in downstream scheduling and as part of a mechanism to prevent futile recursions.

The validated commands pass to Sort step **1508** that transfers each command to a queue of commands for the same action **1510**, that is one of a collection of many command queues **1512**. Each queue for an action is a first-in first-out (FIFO) buffer of commands with the same action, each command having a timestamp applied by step **1504**.

The collection of command queues **1512** with different actions is used by method **1500** to process an execution cycle comprising an ordered sequence of sub-cycles each for a different command queue and command action.

In one execution sub-cycle, a queue output selector **1514** selects one of the queues in the collection of queues **1512** and outputs a batch of commands **1516** with the same action to Decode commands **1518**. For the selected command queue, all commands with the same timestamp will be transferred from command queues **1512** to Decode commands **1518** to schedule execution of a batch of commands with the same action and timestamp.

There are several reasons why commands are scheduled for execution according to a common action. Firstly, each command in each batch has the same priority and they are run in the same cohort, therefore allowing a deterministic behavior for a set of commands of the same action. Secondly, each command action is processed in turn according to a predetermined order in the execution cycle. As a result, a sequence of commands **1502** with different actions will also be processed in a deterministic way according to the batch processing of different actions in a predetermined order. Thirdly, another sequence of commands coming from Resolve hits by command action **1528**, via the executable content decision step **1542**, into sort **1508** with different command actions and the same timestamp applied by step **1528**, will also be sorted into command queues **1512** and then executed in a predetermined sequence of batches of commands according to the execution cycle.

The batch of commands with the same action is processed by Decode commands **1518** to a batch of instructions with the same action. Commands are comprised of three types: (i) the actions that are associated with lists of items comprising a single tag data and qubit combination; (ii) the actions that are associated with lists of items comprising groups of tag data and qubit combinations; (iii) the actions that are associated with lists of pairs of tag data and qubit combinations. Instructions comprise a single tag data and action. Decode commands **1518** converts each command in an input batch of commands to a sequence of one or more instructions each comprising a tag data and qubit combination, and an action. Each instruction generated from the same command is given the same command index.

For commands with actions comprising a single tag data and qubit combination, Decode commands **1518** creates an instruction for each tag data **1404**. Each instruction in the batch will have a unique command index. The instruction batch index and command batch index will be used by the downstream Resolve hits by command action **1528** to identify the corresponding probe wavefunction in the batch of probe wavefunctions created by **1520**.

For commands comprising groups of tag data and qubit combinations, Decode commands **1518** creates a separate instruction for each tag data **1404** in the group of tags. Consecutive instructions are each given a monotonic group index starting at zero and a total equal to the number of tags in the group of tags. The command index, group index and group total will be used by the downstream Resolve hits by command action **1528** to identify corresponding probe wavefunctions in the batch of probe wavefunctions created by probe preparation step **1520**.

For commands comprising pairs of tag and qubit combinations, Decode commands **1518** creates a separate instruction for each tag **1404** in the pair of tags. Each pair of tags comprises a dependent tag and an independent tag. Decode commands will create a dependent tag type instruction and an independent tag type instruction accordingly and each type in the pair of instructions will share the same command index. The two instruction types with a common command

index will be used by the downstream Resolve hits by command action **1528** to identify corresponding probe wavefunctions in the batch of probe wavefunctions created by probe preparation step **1520**.

The output of Decode commands **1518** is the input to probe preparation step **1520** which encodes the tag data according to the previously described method **10** shown in FIG. **3**A. The tag data comprises probe or search sequence of data elements **16** is encoded to a first probe sequence representation by unitary encoding to orthogonal basis **20**, using a method determined by the tag data type. In the case that the qubit associated with the tag has observe true, the probe first sequence representation is rotated by the qubit phase angle **37**. The first probe sequence representation is then transformed to a second probe sequence representation by a unitary orthogonal transform **24** after anti-aliasing in step **22**. The output of the unitary orthogonal transform **24** is the second probe sequence representation and passes as an input to the interference process step **1526**.

A second input to the interference process step **1526** is supplied in the form of RS pairs **1524** that are retrieved from superposition memory **1522**. The interference process step **1526** processes the second probe sequence representation, or probe wavefunction, and the RS pair inputs according to the methods **100**, **301**, **300**, **222** as previously described for the interference of a probe wavefunction with an RS pair. The output of interference process step **1526** is correlation values and hits comprising correlation values that exceed a predetermined threshold which pass to step **1528**, Resolve hits by command action.

The previous steps are common to all commands that have actions on tags. There are several advantages of having a common processing path for different actions beyond efficiency. It is a practical solution for the common dependency of different actions as to whether or how many similar tags are already present in the memory. For example, a command with store action for a tag and its associated content will first generate a probe wavefunction to search for the tag. Depending on whether a similar tag is found or not, a new layer is created or an existing layer tag count or amplitude may be modified.

The common path for different command actions includes step **1528** which resolves hits and correlations produced by the interference process **1526** by different methods according to the action type of the command. The command actions are supplied from Decode commands **1518** to resolve hits by command action **1528**. In the case of a particular command action solve described below Resolve Hits by command action **1528** sends selected RS pairs and solving constraints to solve process **1529**. Following execution of the solve process **1529** results are passed back to Resolve Hits by command action **1528**. The solve process **1529** is described below in detail with reference to FIG. **17** and FIG. **18**.

Instruction Set Architecture

The operation of the preferred embodiment of the present disclosure will now be described for each instruction's action according to an instruction set architecture **1600** depicted in FIG. **16**. For each action listed in the first column of FIG. **16** a symbolic notation will be used to encapsulate the essence of the function with a minimal syntax, as shown in the third column of FIG. **16**. In the interests of being economical, readable and condense, several features of the present method are implicitly incorporated into the symbolic notation and syntax.

Specifically, it is an axiom of the current disclosure that tag data is represented in a transformed representation known as a wavefunction and further that superpositions of wavefunctions comprise a plurality of layer encoded wavefunctions, hence representing a plurality of tags. Accordingly the function notation expresses the wavefunction operations using ϕ and θ to represent two separate wavefunctions for two separate tags, and using Ψ to represent the wavefunction superposition comprising a plurality of layer encoded wavefunctions. In this minimal notation transformation of tags to wavefunctions is implicit in use of ϕ and θ to represent the wavefunctions. The present disclosure describes a system processing a large number of tags which may be collectively represented by a single ϕ or θ symbol.

According to the present disclosure Ψ is a representational notation for the RS pair of wavefunction superpositions. The notation of +=denotes adding or increasing a wavefunction layer in a superposition of wavefunctions. Similarly, the notation of −=denotes removing or decreasing a wavefunction layer from a superposition of wavefunctions.

The process of wavefunction interference is denoted by $\Psi\cdot\phi^*?$ which is employed in multiple different actions producing different results according to the action. The present disclosure describes a system processing a large number of RS pairs which may be collectively represented by a single Ψ symbol. This applies to the interference process that compares an input wavefunction ϕ with all RS pairs and is denoted by $\Psi\cdot\phi^*?$. According to the preferred embodiment of the present disclosure described herein, interference process **1526** performs the wavefunction interference process denoted by $\Psi\cdot\phi^*?$.

It is a further axiom of the current disclosure that data in the form of content which may be executable is stored in a database in correspondence to a wavefunction of its corresponding tag. Consequently the data content may be retrieved by the interference process **1526** that uses the wavefunction of its corresponding tag. According to the present disclosure, the notation D represents a database comprising data in the form of content that is stored and retrieved. The syntax D⇐x syntax denotes content x being retrievably stored in database D. Similarly, the syntax D⇒x syntax denotes content x being retrieved from database D.

Store Action

In the case of a command with action store **1605**, the parameters are tags with associated qubits and optional content which may comprise executable content and non-executable content. The tag data may be any vector of real or complex values that encodes an arbitrary symbol sequence. The tag data may also be any vector of real or complex values that encodes application specific data processing elements, such as the weight vectors of the last layer of a trained neural network. The store action's function shown in FIG. **16** has the formula:

$$\Psi{+}{=}\phi@i,D{\Leftarrow}x$$

where Ψ represents an RS pair wavefunction superposition, ϕ represents the tag data wavefunction and i represents the integer layer position index used to encode tag data wavefunction ϕ in RS pair wavefunction superposition Ψ. In this formula D represents a database and x represents content. The D⇐x syntax denotes content x being retrievably stored in database D using references to wavefunction superposition Ψ and integer layer position index i.

In the case of a command with action store **1605**, step **1528** will detect if any hits exist in the interference process

1526 output. The position of a hit in the interference process output represents a specific combination of probe and RS pair input to interference process **1526**. Accordingly, the position of a hit in the interference process output is decoded to a probe index, a layer index and an RS pair index. The probe index is equal to Decode commands **1518** output instruction batch index, so it is used to locate the command including the tag, contents and qubit.

Continuing the above store command example in Resolve hits by command action **1528**, in the case when there is no hit detected for the probe index the resolution of the store action is to create a new layer in an RS pair stored in superposition memory **1522** and using the layer index and RS pair index, create a lookup entry in memory lookup by layer and RS pair **1536** for the address of a location in memory **1540** where the store tag is saved and a location in memory **1540** where the content is saved.

To store a new layer for a probe index the corresponding probe wavefunction used in by interference process **1526** is supplied from step **1520** to step **1528**. The input probe wavefunction is converted to a target data wavefunction for the new layer. This conversion between probe wavefunction and target wavefunction comprises only a scaling factor since the steps in probe wavefunction preparation are common to the steps in target wavefunction preparation and will generate identical wavefunctions for the same tag data and qubit. The probe wavefunction is therefore converted and passed as a target wavefunction to layer modulation step **1530** with a layer index and an RS pair destination. Step **1530** uses the supplied layer index to select layer modulation functions that are applied to the target wavefunction. A pair of layer encoded wavefunctions are generated by differentially modulating the target wavefunction for the R and S wavefunction superpositions in the RS pair. The R modulated target wavefunction is then combined with the R wavefunction superposition by vector complex addition. The S modulated target wavefunction is likewise combined with the S wavefunction superposition by vector complex addition. The output of layer modulation step **1530** is RS pairs **1532** each comprising a modulated target data wavefunction derived from the probe wavefunction for the same tag data. The RS pairs **1532** are saved to superposition memory **1522**.

Resolve hits by command action **1528** is also connected to superposition memory **1522** via a step of memory lookup by layer and RS pair **1536**, which is in addition connected to memory **1540**. While superposition memory **1522** holds RS pairs, memory **1540** holds tags, content and executable content. The function of memory lookup by layer and RS pair **1536** is to allow a tag to exist in a conventional data form and in a quantum field form as a superposition layer in an RS pair, each at corresponding locations in separate memories. Step **1536** maintains correspondence between superposition memory **1522** and memory **1540** so that a location of a layer index and RS pair index in superposition memory **1522** are converted by step **1536** to the address of a location in memory **1540** where the tag that was encoded to the layer wavefunctions in the RS pair is stored. In addition to providing the address of a location in memory **1540** where the tag is stored, step **1536** will provide the address of a location in memory **1540** where the content and executable content associated with the tag is stored.

Search Action

In the case of a command with action search **1610**, the parameters are tags with associated qubits. The tag data may be any vector of real or complex values that encodes an arbitrary symbol sequence. The search action's function shown in FIG. **16** has the formula:

$$\Psi \cdot \phi^* ?_{i,D\ i,s} \Rightarrow x$$

where Ψ represents a plurality of RS pairs each comprising a wavefunction superposition, ϕ^* represents the tag data conjugate wavefunction, ? denotes the interference process and i represents the recovered integer layer position index of the tag data wavefunction ϕ in RS pair wavefunction superposition Ψ. As previous mentioned Ψ represents a collection of RS pairs for a parallel search according to the present disclosure. In this formula D represents a database and x represents content. The $D_{i,s} \Rightarrow x$ syntax denotes content x being conditionally retrieved from database D using references to wavefunction superposition Ψ denoted by an integer s and integer layer position index denoted by i.

In the case of a command with action search **1610**, Resolve hits by command action step **1528** will detect if any hits exist in the interference process **1526** output. The position of a hit in the interference process output represents a specific combination of probe and RS pair input to interference process **1526**. Accordingly, the position of a hit in the interference process output is decoded to a probe index, a layer index and an RS pair index. The layer index and RS pair index are passed from step **1528** to memory lookup by layer and RS pair **1536**, which will convert these inputs to the address of a location in memory **1540** where the tag is stored and the address of a location in memory **1540** where the content and executable content associated with the tag is stored. Resolve hits by command action step **1528** uses the addresses provided by memory lookup by layer and RS pair **1536** to access memory **1540** to retrieve tag, content and executable content **1538**, which it then passes to the executable decision step **1542**.

The executable decision step **1542** processes the content and executable content input from step **1528** to detect executable content that was retrieved by Resolve hits by command action step **1528** from memory **1540**. All executable content detected by step **1542** is in the form of commands comprising an action and a list of tag, qubit and content combinations. To mark a fresh execution cycle cohort of commands in the method **1500**, each command is given a timestamp updated to be distinct for the start of a new execution cycle. In this way all commands that are passed from step **1542** to sort **1508** and will form a new timestamp cohort of commands in separate action command queues **1512**.

Executable content is directed to sort **1508** and will initiate another execution cycle of the method **1500**. If there is any executable content no response is sent to the client pending the execution of the further execution cycle. The executable decision step **1542** can also create a record of all content and executable content comprising multiple successive execution cycles that can be attached to the eventual response to client **1544**.

Delete Action

In the case of a command with action search **1610**, the parameters are tags with associated qubits. The delete action's function shown in FIG. **16** has the formula:

$$\Psi \cdot \phi^* ?\Psi - = \phi @ i$$

where Ψ represents a plurality of RS pair wavefunction superpositions, ϕ^* represents one or more tag data conjugate wavefunctions, ? denotes the interference process and i

represents recovered the integer layer position index for each of a plurality of tag data wavefunctions ϕ in a plurality of RS pair wavefunction superpositions Ψ. The expression following the inference ? is conditional on the inference detection of ϕ as a hit in a layer of in RS pair superposition Ψ. The conditionally executed action denoted by Ψ−=ϕ@i is to remove tag data wavefunction ϕ at layer i from RS pair wavefunction superposition Ψ.

In the case of a command with action delete **1615**, Resolve hits by command action step **1528** will detect if any hits exist in the interference process **1526** output. The position of a hit in the interference process output represents a specific combination of probe and RS pair input to interference process **1526**. Accordingly, the position of a hit in the interference process output is decoded to a probe index, a layer index and an RS pair index. The layer index and RS pair index are passed from step **1528** to memory lookup by layer and RS pair **1536**, which will convert these inputs to the address of a location in memory **1540** where the tag is stored and the address of a location in memory **1540** where the content and executable content associated with the tag is stored. Resolve hits by command action step **1528** uses the addresses provided by memory lookup by layer and RS pair **1536** to access memory **1540** to retrieve the tag **1538** used to encode the target wavefunction at the hit layer in the RS pair. The type of tag will determine how to confirm the deletion and according to the applicable method the probe tag is compared with the target layer tag retrieved from memory **1540**. If the probe and target tag comparison method confirms the match detected as an output of interference process **1526**, the layer responsible for the hit will be removed from the RS pair and the tag, content and executable content stored in memory **1540** will be deleted from memory **1540**.

To remove a layer from an RS pair, Resolve hits by command action **1528** passes the layer index and the RS pair index to the memory lookup by layer and RS pair **1536**, which converts these inputs to a location in superposition memory **1522** and retrieves the RS pair **1534** containing the hit layer. Resolve hits by command action also uses the probe index to retrieve the probe wavefunction that was supplied to step **1528** by step **1520**. The following steps share some of the steps that were used for the store action resolution to add a new layer. The probe wavefunction is converted to a target wavefunction as described for store however the real and imaginary components of each element of the target wavefunction complex vector are each negated. The negated target wavefunction is modulated by the layer index in the same way as for store to generate an R modulated negated target wavefunction and an S target modulated negated wavefunction and a vector complex addition step is also used to combine these with the R and S superpositions respectively in an RS pair. The tag wavefunctions at the hit layer in the RS pair will be annihilated by the R modulated negated target wavefunction and an S modulated negated target wavefunction.

Inference Action

In the case of a command with action inference **1620**, the parameters are tags with associated qubits. The inference action's function shown in FIG. **16** has the formula:

$$P(i)=\Psi\cdot\phi^{*}?$$

where Ψ represents an RS pair wavefunction superposition, ϕ* represents the tag data conjugate wavefunction, ? denotes the interference process and P(i) represents the probability of the tag data wavefunction ϕ associated with each integer layer position index i in RS pair wavefunction superposition Ψ. The tag data may be any vector of real or complex values that encodes application specific data processing elements, such as the feature vectors of the last layer of a trained neural network. According to an exemplary embodiment of the present disclosure, the tag data for wavefunction ϕ comprises feature vectors of the last layer of a trained neural network and RS pair superposition Ψ comprises weight vectors of the last layer of a trained neural network. In this case the inference action denoted Ψ·ϕ*? performs the equivalent calculation of the dot product of the tag last layer feature vector and the last layer weight vector for each classification category in a trained neural network.

In the case of a command with action inference **1620** the list **1428** comprises tag data **1404** in which the type of data is the input to the last layer of a neural net. The input tag data is generated by presenting unclassified samples to the first or input layer of a neural net and using trained weights to apply forward propagation for all intermediate layers ending with the input data to the last layer of the neural net. The output of the command with action inference is a classification category for any unclassified sample that has been recognized within a certain probability. According to the present method the last layer computation of a trained neural net may be performed by interference process **1526** where the RS pairs **1524** comprise superpositions of weights for different trained classifications. Interference process **1526** uses tag data probes and RS pair classification category weights, each expressed in as a second orthogonal domain representation, complex conjugation of the probe wavefunction, complex multiplication and a unitary orthogonal inverse transform to generate an output data that is a parallel computation of the dot product between the last layer input data and the last layer trained weights for each classification category. The dot product is equal to the sum of multiplication products between last layer input data values and the last layer trained weights corresponding to each value and is computed for each classification category. For commands with action inference **1620**, as for those with action search **1610**, the interference process step **1526** processes the second probe sequence representation, or probe wavefunction, and the RS pair inputs according to the methods **100** in FIG. **5**A, method **301** in FIG. **5**C, method **300** in FIG. **5**D and method **222** in FIG. **5**J as previously described for the interference of a probe wavefunction with an RS pair.

In the case of a command with action inference **1620**, Resolve Hits by command action step **1528** will detect if any classification of probe tags may be inferred from the interference process **1526** output. Since the target superposition layers are weights for different classification categories, the layer with the largest interference value corresponds to the most likely classification category, since the interference values are the dot product between the last layer input data and the last layer trained weights for each classification category. The degree of confidence of the classification is also important to estimate in order to inform decision-making. To estimate the probability that the classification is correct one soft max method frequently used is to sum the negative exponential function of the maximum dot product value of a probe tag versus all classification categories minus the dot product value of the probe tag for each classification category. In the case that the estimated probability for a classification category exceeds a predetermined threshold the output of Resolve hits by command action **1528** is the classification category for the probe tag. Alternatively, if the estimated probability for a classification

category is less than a predetermined threshold no classification is made for the probe tag. The position of interference values in the interference process output represents a specific combination of probe and RS pair input to interference process **1526**. Accordingly, the interference values and their positions in the interference process output are decoded to classification probabilities for a probe index to belong to a category represented by a layer index and an RS pair index.

Dot Product Action

In the case of a command with action dot product **1625**, the parameters are tags with associated qubits. The dot product action's function shown in FIG. **16** has the formula:

$$R(i)=\Psi\cdot\phi^{*}?$$

where Ψ represents an RS pair wavefunction superposition, ϕ* represents the tag data conjugate wavefunction, ? denotes the interference process and R(i) represents the dot product of the tag data row vector and the column vector associated with each integer layer position index i in RS pair wavefunction superposition Ψ. The dot product may be a real or a complex number.

In the case of a command with action dot product **1625**, Resolve Hits by command action step **1528** will format the dot product values output by interference process **1526** according to the probe, RS pair and layer in the RS pair. The formatted dot product values will be sent to the client in the ultimate response by step **1544**.

In the case of a command with action tensor product **1627**, the parameters are a pairs of tags with associated qubits where each pair comprises an independent tag and a dependent tag. According to the instructions set architecture **1600** of the present disclosure, independent tags provide selective functionality that is conditioned on the presence of the independent tag wavefunction ϕ as a layer in a quantum register represented by an RS pair of wavefunction superpositions. The tag data may be any vector of real or complex values that encodes an arbitrary symbol sequence. The independent tag wavefunction ϕ represents the selection function for the first or outer vector space of the tensor product calculation.

The dependent tag similarly selects the second or inner vector space of the tensor product calculation conditioned on the presence of the dependent tag wavefunction θ as a layer in a quantum register represented by an RS pair of wavefunction superpositions. The tensor product action's function shown in FIG. **16** has the formula:

$$T=(\Psi\cdot\phi^{*}?)\otimes(\Psi\cdot\theta^{*}?)$$

where Ψ represents an RS pair wavefunction superposition, ϕ* represents the independent tag data conjugate wavefunction, ? denotes the interference process, θ* represents the dependent tag data conjugate wavefunction, and T represents the tensor product from the operation denoted ⊗ on the outer vector space comprising the quantum register represented by an RS pair containing the independent tag ϕ as a layer and the inner vector space comprising the quantum register represented by an RS pair containing the dependent tag θ as a layer.

Correlate Action

In the case of a command with action correlate **1630**, the parameters are tags with associated qubits. The correlate action's function shown in FIG. **16** has the formula:

$$r(i)=\Psi\cdot\phi^{*}?$$

where Ψ represents an RS pair wavefunction superposition, ϕ* represents the tag data conjugate wavefunction, ? denotes the interference process and r(i) represents the correlation coefficient, also known as the product moment, between the tag data row vector and the column vector associated with each integer layer position index i in RS pair wavefunction superposition Ψ. According to the present disclosure the wavefunction interference process **1526** that is used for the dot product action may also be used for correlation coefficient calculation by scaling wavefunctions to generate an interference product range of −1 to 1 for a correlation coefficient, where: 1 represents 100% correlation of identical vectors; 0 represents 0% correlation of orthogonal vectors and −1 represents negative 100% correlation of oppositely signed identical vectors.

In the case of a command with action correlate **1630**, Resolve Hits by command action step **1528** will format the correlation values output by interference process **1526** according to the probe, RS pair and layer in the RS pair. The formatted correlation values will be sent to the client in the ultimate response by step **1544**.

Associate Action

In the case of a command with action associate **1635**, the parameters are tags with associated qubits. The associate action's function shown in FIG. **16** has the formula:

$$\Psi+=\phi_k@i_k$$

$$D\Leftarrow x_k$$

where Ψ represents an RS pair wavefunction superposition, ϕ_k represents the k'th element in a plurality of tag data wavefunctions and i_k represents the k'th element in a plurality of integer layer position indeces used to encode tag data wavefunction ϕ_k in RS pair wavefunction superposition Ψ. In this formula D represents a database and x represents content. The $D\Leftarrow x_k$ syntax denotes content x_k representing the k'th element in a plurality of content data being retrievably stored in database D using references to wavefunction superposition Ψ and integer layer position index i_k.

In the case of a command with action associate **1635**, Decode commands **1518** creates a separate instruction for each tag **1404** in the group of tags in items in a list **1428** in a command with action **1426** equal to associate. Consecutive instructions in a group of tags are each given a monotonic group index starting at zero and a total equal to the number of tags in the group of tags in the list. The group index and group total will be used by the downstream Resolve hits by command action **1528** to identify corresponding probe wavefunctions in the batch of probe wavefunctions created by Decode commands **1518**, each for a tag in the group of the tags in the associate command.

The associate commands and instructions are supplied by Decode commands step **1518** to Resolve hits by command action **1528** so that this step can process all the tags in a group in an item in a list **1428** contained in a command with action **1426** equal to associate. The resolution of an associate action is an RS pair comprising all the tags in the group. Each probe wavefunction supplied by probe preparation step **1520** is converted to a target wavefunction as described for the store action. Target wavefunctions are passed to layer modulation step **1530** with a layer index for the destination layer in the new RS pair. When the target wavefunctions for

all tags in the group have been modulated to layers the RS pairs **1532** are saved to superposition memory **1522**.

Select Action

In the case of a command with action select **1640**, the parameters are tags with associated qubits. The select action's function shown in FIG. **16** has the formula:

$$\cap \Psi \cdot \phi_k^* \cap m_k?$$

$$D_{ik,s} \Rightarrow x_k$$

where $\cap$ represents the intersection at the individual quantum register of the following expression, meaning that all logical elements of the expression are true for an individual RS pair that represents a quantum register; Ψ represents an RS pair wavefunction superposition, ϕ_k^* represents the k'th wavefunction in a plurality of tag data conjugate wavefunctions and m_k represents the k'th element in a plurality of binary condition mask comprising 'included' and 'not included' logical conditions to apply to the interference result of each corresponding tag data wavefunction θ_k using the (second) intersection operator $\cap$. In the event that all logical conditions determined by the condition mask nu, are true the selection action triggers a retrieval from the database D. In the event that the logical condition of each nu, are not all satisfied there is no retrieval from the database D.

The $D_{ik,s} \Rightarrow$xk syntax denotes content xk representing the k'th element in a plurality of content data being conditionally retrieved from database D using references to wavefunction superposition Ψ denoted by integer s and integer layer position index denoted by ik.

In the case of a command with action select **1640**, Decode commands **1518** creates a separate instruction for each tag in the group of tags in an item in a list **1428** in a command **1424** with action **1426** equal to select. The tags within a group each have an include qualifier that is either true or false. Resolve hits by command action step **1528** will detect if hits exist in the interference process **1526** output for each of the include true and include false tags in the group of tags in a command with action select. For each RS pair, if all include true tags are present as hits and none of the include false tags in the same group are present as hits, then the select action is successful for that RS pair. A successful select action results in the content and executable content of each tag with include true being retrieved from memory **1540** and sent to the execution decision step **1542**. An unsuccessful select action will not retrieve any content from memory.

The tag in a group in a command with action select **1640** can also be associated with an executable content **1414** and a qubit **1408** as shown in object **1412**. In the event that the select action is successful, the executable content **1538** associated with the tag is retrieved from memory **1540** by Resolve hits by command action **1528** passing a layer index and an RS pair index of a detected hit to Memory lookup by layer and RS pair **1536**. The executable content is passed by step **1528** to the executable decision **1542** step which will direct it to the sort **1508**.

The position of a hit in the interference process **1526** output represents a specific combination of probe and RS pair input to interference process **1526**. Resolve hits by command action **1528** will process the interference process **1526** output according to different actions in the instruction set architecture **1600**. For example, in the preferred embodiment, for some commands such as select that comprise groups of tags, the interference process **1526** output comprising correlation values at positions that are addressable by a probe index, a layer index and an RS pair index, is processed with an outer loop on RS pairs and an inner loop on probe index and inside loop on layer. The advantage of this processing pattern is that a list of RS pair hits has a grouped monotonic set of probe index hits for each RS pair. Resolve hits by command action **1528** uses the processing pattern in a first processing step to create an intermediate hit resolution list. The intermediate hit resolution list is input to a second processing step that applies the select command tag's include true or false selection modifier. The second processing step receives a consecutive list of all probe hits in ascending probe index order, so the combined logic for all include true or false tags can be evaluated by comparing the include condition versus the presence or absence of a hit in the RS pair for a probe wavefunction corresponding to a tag in the select command.

The tag data **1404** in a command **1424** with action select may include binary sequences, real and complex sequences, symbol sequences and cryptographic hashing codes as unique identifiers. In addition, the type of tag data **1404** may specify the function to be applied to the data which includes the functions of correlation, dot product, and neural net inference in addition to general-purpose search using sequences of symbols and unique identifiers.

In the preferred embodiment interference process **1526** is a common step that outputs a function which may be symbol match, real correlation, dot product or inference values between the probe data comprising a tag data in a command with actions of search, select, correlate, inference or dot product, and a target data that is represented in superposition memory **1522** in the form of a layer in an RS pair of wavefunction superpositions. In this way the RS pair is able to hold different representations of real world objects, categories and concepts in a single combined record. For example, the RS pair may comprise layers that are noun or verb synonyms in ASCII binary text form, sounds in real digital samples, and inference last layer weights of a trained neural net. A selection may match the word with the sound and a feature vector versus the trained weights for multi-domain comparison references to the category named by the word and sound.

The instruction set architecture **1600** includes commands with actions that evaluate the complex phase angle of layers in RS pairs. To accommodate reading the complex phase angle of layers in RS pairs, Decode commands **1518** creates four instructions for each such tag, each with complex phases of 0°, 90°, 180° and 270°. Resolve hits by command action will evaluate the pair of two highest correlations to select a pair of adjacent complex phase angles, then measure the complex phase angle of the layer for that tag in the RS pair. For example if the two highest correlations are at 0° and 90°, the complex phase angle ω is estimated using the formula ω=arctangent(correlation at 90°/correlation at 0°).

Entangle Action

In the case of a command with action entangle **1645**, the parameters are pairs of tags with associated qubits where each pair comprises an independent tag and a dependent tag. According to the instructions set architecture **1600** of the present disclosure, pairs of dependent and independent tags provide selective functionality that is conditioned on the presence of the independent tag wavefunction ϕ as a layer in a quantum register represented by an RS pair of wavefunction superpositions. The tag data may be any vector of real or complex values that encodes an arbitrary symbol sequence. The independent tag wavefunction ϕ represents

the selection function that makes a conditional action by the dependent tag wavefunction θ.

For a command with action entangle **1645** the action's function shown in FIG. **16** has the formula:

$$\Omega=\Sigma\Psi\cdot\phi^{*}?\angle\phi$$

$$\forall\theta^{\circ}+=\Omega$$

where Ω represents a qubit phase angle that is applied to the qubit phase angle of all dependent tag wavefunctions ϕ. In the entangle command action function the dependent wavefunction θ is not required to be in the same RS pair embodiment of a quantum register as the independent tag wavefunction θ. The θ°+=Ω notation indicates that the qubit phase, represented by the degree symbol ° of the dependent wavefunction θ, is incremented by the phase angle Ω. For example, if Ω is π radians or 180° each dependent will be rotated to have a new qubit phase angle that is π radians, 180° different from their previous qubit phase, which is also the wavefunction of the negative tag data when the tag data is real or complex, and also the binary complement when the tag data is binary.

The condition for the action entangle is denoted by $\Omega=\Sigma\Psi\cdot\phi^{*}?\angle\phi$ where Ω is a phase angle that equates to a summation Σ of angles modulo 2π radians, 360°, applied to the qubit phase ∠ of the independent tag wavefunction ϕ. As previously described $\Psi\cdot\phi^{*}$ denotes the interference process between a plurality of RS pair wavefunction superpositions Ψ and the independent tag wavefunction ϕ such that the entangle action accumulates the qubit phase of each independent tag wavefunction ϕ detected by the interference process.

From the preceding description of the command with action entangle it will be apparent that the accumulated qubit phase of the independent tag wavefunctions applied to dependent tag wavefunctions independently in separate quantum field registers provides a functionality of both polling and broadcasting information in a plurality of quantum field registers.

In the case of a command with action entangle **1645**, there are four different operators depending on the combination of the observe Boolean in the qubit **1408** associated with the independent tag and the observe Boolean in the qubit **1408** associated with the dependent tag. In the case that the qubit associated with the independent tag has the observe Boolean true Decode commands **1518** creates one instruction for each independent tag **1404** with a complex phase equal to the qubit **1408** phase. In the case that the qubit associated with the independent tag has the observe Boolean false Decode commands **1518** creates four separate instructions for each independent tag, each with complex phases of 0°, 90°, 180° and 270°. Decode commands **1518** also creates four separate instructions for each dependent tag, each with complex phases of 0°, 90°, 180° and 270°, for both the case that the qubit associated with the dependent tag has observe true and the case that has observe false. An entangle command comprising a dependent tag and independent tag as a pair therefore generates either five or eight instructions for two tags depending on the observe Boolean of the qubit associated with the independent tag. The instructions pass from Decode commands **1518** to probe preparation **1520** and Resolve hits by command action **1528**, which also receives the entangle commands.

Following the interference process **1526** between the probe wavefunctions output from probe preparation **1520** and RS pairs **1524** that were retrieved from superposition memory **1522**, Resolve hits by command action **1528** processes entangle commands in the order of the independent tags followed by the dependent tags. Since the output of interference process **1526** comprises different combinations of probe wavefunction and RS pair, it is a range of correlation values addressable by probe index, RS pair index and layer index. In the case of action entangle all independent tags are processed with probe index as the outer loop, RS pair as the inner loop and layer as the inside loop. The complex phase of each layer in the collection of RS pairs matching the independent tag is accumulated to a total sum of complex phase angle of all matching independent tag layers. In the case where the observe Boolean is false in the qubit associated with the independent tag four separate instructions were generated each with complex phases of 0°, 90°, 180° and 270°. As a result, the independent tag will be matched at any angle it is present as a layer in the RS pairs. In the case that the observe Boolean is true in the qubit associated with the independent tag, a single instruction was generated at the qubit complex phase and consequently the independent tag will only be matched if it is present at a similar phase as a layer in the RS pairs. The observe Boolean in the qubit associated with the independent tag therefore determines if the independent tag is selectively matched by phase or non-selectively matched for all phases.

Resolve hits by command **1528** next evaluates the complex phase angle of each layer in the collection of RS pairs matching the dependent tag. In the case where the qubit associated with the dependent tag observe Boolean is true, the total complex phase angle of the independent tag layers is applied as a rotation of the complex phase angle for the layers matching the dependent tag. This rotation is achieved by first generating a pair of R and S modulated target wavefunctions for the dependent tag layer in the RS pair at the estimated complex phase angle as a first modifying component. Next, generating a pair of R and S modulated target wavefunctions for the dependent tag layer in the RS pair at the sum of the estimated complex phase angle and the total complex phase angle of the matching independent tag layers as a second modifying component. The R and S wavefunctions in the first modifying component are subtracted from the R and S wavefunctions respectively in the second component by vector complex addition to generate modifying R and S wavefunctions. The modifying R wavefunction is then combined with the R wavefunction in the dependent tag RS pair by vector complex addition. The modifying S wavefunction is the combined with the S wavefunction in the dependent tag RS pair by vector complex addition. The effect of the preceding steps is to replace the layer in the RS pair for the dependent tag in an entangle action pair with the same layer at a rotated complex phase angle.

In the case of an entangle command where the qubit associated with the dependent tag observe Boolean is false a new layer is added for each matched dependent tag layer. The new dependent tag layer will have the same complex phase as a modified dependent tag layer in the case where the observe Boolean was true. As a result, the effect of the dependent tag qubit having observe false is to create a second layer which is a phase rotated copy of the matched dependent tag layer. As in the observe Boolean true case, the total complex phase angle of the independent tag layers is applied as a rotation of the complex phase angle for the layers matching the dependent tag.

Detangle Action

In the case of a command with action detangle **1650**, the parameters are pairs of tags with associated qubits where

each pair comprises an independent tag and a dependent tag. According to the instructions set architecture **1600** of the present disclosure, pairs of dependent and independent tags provide selective functionality that is conditioned on the presence of the independent tag wavefunction ϕ as a layer in a quantum register represented by an RS pair of wavefunction superpositions. The tag data may be any vector of real or complex values that encodes an arbitrary symbol sequence. The independent tag wavefunction ϕ represents the selection function that makes a conditional action by the dependent tag wavefunction θ.

For a command with action detangle **1650** the action's function shown in FIG. **16** has the formula:

$$\Omega=\Sigma\Psi\cdot\phi^{*}?\angle\phi$$

$$\forall\theta^{\circ}-=\Omega$$

where Ω represents a qubit phase angle that is applied to the qubit phase angle of all dependent tag wavefunctions ϕ. In the detangle command action function the dependent wavefunction θ is not required to be in the same RS pair embodiment of a quantum register as the independent tag wavefunction θ. The $\theta^{\circ}-=\Omega$ notation indicates that the qubit phase, represented by the degree symbol ° of the dependent wavefunction θ, is decremented by the phase angle Ω. For example, if Ω is π radians or 180° each dependent will be rotated to have a new qubit phase angle that is π radians, 180° different from their previous qubit phase, which is also the wavefunction of the negative tag data when the tag data is real or complex, and also the binary complement when the tag data is binary.

The condition for the action detangle is denoted by $Q=E\ \Psi\cdot\phi^{*}?\angle\phi$ where Ω is a phase angle that equates to a summation Σ of angles modulo 2π radians, 360°, applied to the qubit phase ∠ of the independent tag wavefunction θ. As previously described $\Psi\cdot\phi^{*}$ denotes the interference process between a plurality of RS pair wavefunction superpositions Ψ and the independent tag wavefunction ϕ such that the entangle action accumulates the qubit phase of each independent tag wavefunction ϕ detected by the interference process.

From the preceding description of the command with action detangle it will be apparent that the accumulated qubit phase of the independent tag wavefunctions applied to dependent tag wavefunctions independently in separate quantum field registers provides a functionality of both polling and broadcasting information in a plurality of quantum field registers.

In the case of a command with action detangle **1650**, there are four different operators depending on the combination of the observe Boolean in the qubit associated with the independent tag and the observe Boolean in the qubit associated with the dependent tag. In the case that the qubit associated with the independent tag has the observe Boolean true Decode commands **1518** creates one instruction for each independent tag with a complex phase equal to the qubit phase. In the case that the qubit associated with the independent tag has the observe Boolean false Decode commands **1518** creates four separate instructions for each tag, each with complex phases of 0°, 90°, 180° and 270°. Decode commands **1518** also creates four separate instructions for each dependent tag, each with complex phases of 0°, 90°, 180° and 270°, for both the case that the qubit associated with the dependent tag has observe true and the case that has observe false. A detangle command comprising a dependent tag and independent tag as a pair, therefore generates either five or eight instructions for two tags depending on the observe Boolean of the qubit associated with the independent tag. The instructions pass from Decode commands **1518** to probe preparation **1520** and resolve hits by command action **1528**, which also receives the detangle commands. The detangle instructions from step **1518** are processed by probe preparation **1520** to create probe wavefunctions that are identical to the probe wavefunctions created by entangle instructions.

Following the interference process **1526** between the probe wavefunctions output from probe preparation **1520** and RS pairs **1524** that were retrieved from superposition memory **1522**, Resolve hits by command action **1528** processes detangle commands in the order of the independent tags followed by the dependent tags. The processing of the output of interference process **1526** is identical to the steps applied in processing entangle commands except that a rotation by the negative of the total complex phase angle of the matching independent tag layers is applied to each matching dependent tag layer instead of a rotation by the total complex phase angle of the matching independent tag layers.

Action Add

In the case of a command with action add **1655**, the parameters are pairs of tags with associated qubits where each pair comprises an independent tag and a dependent tag. According to the instructions set architecture **1600** of the present disclosure, pairs of dependent and independent tags provide selective functionality that is conditioned on the presence of the independent tag wavefunction as a layer in a quantum register represented by an RS pair of wavefunction superpositions. The tag data may be any vector of real or complex values that encodes an arbitrary symbol sequence. The independent tag wavefunction ϕ represents the selection function that makes a conditional action by the dependent tag wavefunction θ.

For a command with action add **1655** the action's function shown in FIG. **16** has the formula:

$$\Psi\cdot\phi^{*}?\Psi+=\theta@i$$

where $\Psi\cdot\phi^{*}$ denotes the interference process between a plurality of RS pair wavefunction superpositions Ψ and a plurality of independent tag wavefunctions ϕ. In the event that the independent tag wavefunctions is detected as a layer in an RS pair of wavefunction superpositions embodiment of a quantum register, the dependent tag wavefunction θ is added as a layer to the same RS pair of wavefunction superpositions embodiment of a quantum register.

In the case of a command with action add **1655**, Decode commands **1518** creates instructions for the independent tag **1404** and qubit **1408** combination and instructions for the dependent tag **1404** and qubit **1408** combination. In the case the qubit associated with the independent tag has observe true, a single instruction is created by Decode commands **1518** and passed to probe preparation **1520** which will rotate the probe wavefunction by the phase angle of the qubit **1408** associated with the independent tag **1404**. The commands and instructions are also passed from Decode commands **1518** to Resolve hits by command action **1528**.

In a command with action add, for the case the qubit **1408** associated with the independent tag **1404** has observe false, four instructions are created by Decode commands **1518** and passed to probe preparation **1520** which will rotate each of the probe wavefunctions created by complex phase angles of 0°, 90°, 180° and 270°. In this way a panel of probe wavefunctions is created for the same independent tag. The

commands and instructions are also passed from Decode commands **1518** to Resolve hits by command action **1528**.

For the dependent tag and qubit combination in a command with action add, a single instruction is created by Decode commands **1518** and passed to probe preparation **1520** which will rotate the probe wavefunction by the phase angle of the qubit **1408** associated with the dependent tag **1404**. The processing of the dependent tag and qubit combination by steps **1518** and **1520** is the same for either qubit observe true or qubit observe false scenario. The commands and instructions are also passed from Decode commands **1518** to Resolve hits by command action **1528**. The qubit associated with the dependent tag applies the observe true or false property in Resolve hits by command action **1528** to select either a discrete addition for observe true with the creation of an additional layer, or continuous addition for observe false with the addition of layer encoded wavefunction to the RS pair.

Following the interference process **1526** between the probe wavefunctions output from probe preparation **1520** and RS pairs **1524** that were retrieved from superposition memory **1522**, resolve hits by command action **1528** processes add commands in the pairs of the independent tags and the dependent tags. For example, in the preferred embodiment, as described above Decode commands **1518** creates instructions for the independent tag and qubit combination and instructions for the dependent tag and qubit combination, and the commands and instructions are supplied by step **1520** to step **1528**. The probe wavefunctions for the independent tag and the dependent tag consequently form a group of two or five consecutive instructions, depending on observe true or false of the qubit associated with the independent tag. As a result, resolve hits by command action **1528** can identify dependent and independent tag probe wavefunctions as a range of probe wavefunction indexes generated in a batch by probe preparation step **1520**.

Since the output of interference process **1526** comprises different combinations of probe wavefunction and RS pair, it is range of correlation values addressable by probe index, RS pair index and layer index. In the case of action add the addressable range is processed with the outer loop as RS pairs, probe index as the inner loop and layer index as the inside loop. The advantage of this processing pattern is that a list of RS pair hits has a grouped monotonic set of probe index hits for each RS pair. Resolve hits by command action **1528** uses the processing pattern in a first processing step to create an intermediate hit resolution add list comprising RS pairs with both the independent tag hit and the dependent tag hit in a pair, for each item in the list.

The intermediate hit resolution add list is next processed by Resolve hits by command action **1528** to apply the dependent tag to the matching layers in the RS pair that also contains the independent tag as one or more layers in the same RS pair. In this way the independent tag provides for selective action of the add function on only those RS pairs comprising the independent tag as a layer. As described above, if the qubit associated with the independent tag has observe true then a single probe wavefunction is created by probe preparation step **1520**, so the add function will only be applied to RS pairs comprising a layer matching the independent tag at or around the qubit phase angle. Alternatively, if the qubit associated with the independent tag has observe false, a panel of four probe wavefunctions is created by probe preparation step **1520** at complex phase angles of 0°, 90°, 180° and 270°, so the add function will be applied to RS pairs comprising a layer matching the independent tag at any qubit phase angle.

There are two processing paths for add commands in Resolve hits by command action **1528** depending on the qubit associated with the dependent tag having observe true or false.

In the scenario that the qubit associated with the dependent tag has observe true, a new layer is created in the RS pair. To store a new layer for the dependent tag, the probe index of the dependent tag in the intermediate hit resolution identifies the corresponding probe wavefunction used in by interference process **1526** which is also supplied from step **1520** to step **1528**. The input probe wavefunction is converted to a target data wavefunction for the new layer. This conversion between probe and target wavefunction comprises only a scaling factor since the steps in probe wavefunction preparation are common to the steps in target wavefunction preparation and will generate identical wavefunctions for the same tag data and qubit. The probe wavefunction is therefore converted and passed as a target wavefunction to layer modulation step **1530** with a layer index and an RS pair destination. Step **1530** uses the supplied layer index to select layer modulation functions that are applied to the target wavefunction using the previously described target preparation method shown in FIG. **4**A. A pair of layer encoded wavefunctions are generated by differentially modulating the target wavefunction for the R and S wavefunction superpositions in the RS pair. The R modulated target wavefunction is then combined with the R wavefunction superposition by vector complex addition. The S modulated target wavefunction is likewise combined with the S wavefunction superposition by vector complex addition. The output of layer modulation step **1530** is RS pairs **1532** each comprising a modulated target data wavefunction derived from the probe wavefunction for the same tag data. The RS pairs **1532** are saved to superposition memory **1522**.

In the scenario that the qubit associated with the dependent tag has observe false, one or more existing layers in the RS pair are modified by adding the layer modulated wavefunction of the dependent tag to the matching layer. This is achieved by converting the probe wavefunction corresponding to the dependent tag supplied by probe preparation **1520** to a target wavefunction by multiplication with a positive scaling factor. Next, the layer and RS pair index of the dependent tag hit are passed to memory lookup by layer and RS pair **1536** to access superposition memory **1522** for the RS pair **1532** that is to be modified by the add action resolution. Resolve hits by command action **1528** next modifies the wavefunction superpositions in the RS pair by first generating an R modulation of the target wavefunction for the layer of the dependent tag hit, and an S modulation of the target wavefunction for the layer of the dependent tag hit using the previously described target preparation method shown in FIG. **4**A. The R modulation of the target wavefunction is added to the R wavefunction superposition in the RS pair, and the S modulation of the target wavefunction is added to the S wavefunction superposition in the RS pair using vector complex addition. The output of the vector complex addition is a modified RS pair **1534** that is written back to superposition memory **1522**.

Action Subtract

In the case of a command with action subtract **1660**, the parameters are pairs of tags with associated qubits where

each pair comprises an independent tag and a dependent tag. According to the instructions set architecture **1600** of the present disclosure, pairs of dependent and independent tags provide selective functionality that is conditioned on the presence of the independent tag wavefunction ϕ as a layer in a quantum register represented by an RS pair of wavefunction superpositions. The tag data may be any vector of real or complex values that encodes an arbitrary symbol sequence. The independent tag wavefunction ϕ represents the selection function that makes a conditional action by the dependent tag wavefunction θ.

For a command with action subtract **1660** the action's function shown in FIG. **16** has the formula:

$$\Psi \cdot \phi^* ? \Psi = \theta @ i$$

where Ψ·ϕ* denotes the interference process between a plurality of RS pair wavefunction superpositions Ψ and a plurality of independent tag wavefunctions ϕ. In the event that the independent tag wavefunctions ϕ is detected as a layer in an RS pair of wavefunction superpositions embodiment of a quantum register, the dependent tag wavefunction θ is subtracted as a layer from the same RS pair of wavefunction superpositions embodiment of a quantum register

In the case of a command with action subtract **1660**, Decode commands **1518** creates instructions for the independent tag and qubit combination and instructions for the dependent tag and qubit combination. In the case the qubit associated with the independent tag has observe true, a single instruction is created by Decode commands **1518** and passed to probe preparation **1520** which will rotate the probe wavefunction by the phase angle of the qubit **1408** associated with the independent tag **1404**. The commands and instructions are also passed from Decode commands **1518** to Resolve hits by command action **1528**.

In the case the qubit associated with the independent tag has observe false, four instructions are created by Decode commands **1518** and passed to probe preparation **1520** which will rotate each of the probe wavefunctions created by complex phase angles of 0°, 90°, 180° and 270°. In this way a panel of probe wavefunctions is created for the same independent tag. The commands and instructions are also passed from Decode commands **1518** to Resolve hits by command action **1528**.

For the dependent tag and qubit combination, a single instruction is created by Decode commands **1518** and passed to probe preparation **1520** which will rotate the probe wavefunction by the phase angle of the qubit associated with the dependent tag. The processing of the dependent tag and qubit combination by steps **1518** and **1520** is the same for either qubit true or qubit false scenario. The commands and instructions are also passed from Decode commands **1518** to resolve hits by command action **1528**. The dependent tag associated qubit applies the observe true or false property in resolve hits by command action **1528** to select either a discrete subtraction for observe true with the removal of an existing layer, or continuous subtraction for observe false with the subtraction of layer encoded wavefunction from the RS pair.

Following the interference process **1526** between the probe wavefunctions output from probe preparation **1520** and RS pairs **1524** that were retrieved from superposition memory **1522**, resolve hits by command action **1528** processes subtract commands in the pairs of the independent tags and the dependent tags. For example, in the preferred embodiment, as described above Decode commands **1518** creates instructions for the independent tag and qubit combination and instructions for the dependent tag and qubit combination, and the commands and instructions are supplied by step **1520** to step **1528**. The probe wavefunctions for the independent tag and the dependent tag consequently form a group of two or five consecutive instructions, depending on observe true or false of the qubit associated with the independent tag. As a result, resolve hits by command action **1528** can identify dependent and independent tag probe wavefunctions as a range of probe wavefunction indexes generated in a batch by probe preparation step **1520**.

Since the output of interference process **1526** comprises different combinations of probe wavefunction and RS pair, it is range of correlation values addressable by probe index, RS pair index and layer index. In the case of action subtract the addressable range is processed with the outer loop as RS pairs, probe index as the inner loop and layer index as the inside loop. The advantage of this processing pattern is that a list of RS pair hits has a grouped monotonic set of probe index hits for each RS pair. Resolve hits by command action **1528** uses the processing pattern in a first processing step to create an intermediate hit resolution subtract list comprising RS pairs with both the independent tag hit and the dependent tag hit in a pair, for each item in the list in a command with action subtract.

The intermediate hit resolution subtract list is next processed by Resolve hits by action **1528** to apply the dependent tag to the matching layers in the RS pair that also contains the independent tag as one or more layers in the same RS pair. In this way the independent tag provides for selective action of the subtract function on only those RS pairs comprising the independent tag as a layer. As described above, if the qubit associated with the independent tag has observe true a single probe wavefunction is created by probe preparation step **1520** so the subtract function will only be applied to RS pairs comprising a layer matching the independent tag at or around the qubit phase angle. Alternatively, if the qubit associated with the independent tag has observe false, a panel of four probe wavefunctions at is created by probe preparation step **1520** at complex phase angles of 0°, 90°, 180° and 270°, so the subtract function will be applied to RS pairs comprising a layer matching the independent tag at any qubit phase angle.

There are two processing paths for subtract commands in Resolve hits by command action **1528** depending on the qubit associated with the dependent tag having observe true or false.

In the scenario that qubit associated with the dependent tag has observe true, a new layer is created in the RS pair. To store a new layer for the dependent tag, probe index the corresponding probe wavefunction used in by interference process **1526** is supplied from step **1520** to step **1528**. The input probe wavefunction is converted to a target data wavefunction for the new layer. This conversion between probe and target wavefunction comprises multiplying real and imaginary components by a negative scale factor. Since the steps in probe wavefunction preparation are common to the steps in target wavefunction preparation and will generate identical wavefunctions for the same tag data and qubit, a negative unity scaling for example, creates a target wavefunction with the same root mean squared power that is at a 180° complex phase angle. The converted target wavefunction is passed to layer modulation step **1530** with a layer index and an RS pair destination. Step **1530** uses the supplied layer index to select layer modulation functions that are applied to the target wavefunction using the previously described target preparation method shown in FIG. **4**A. A pair of layer encoded wavefunctions are generated by

differentially modulating the target wavefunction for the R and S wavefunction superpositions in the RS pair. The R modulated target wavefunction is then combined with the R wavefunction superposition by vector complex addition. The S modulated target wavefunction is likewise combined with the S wavefunction superposition by vector complex addition. The output of layer modulation step **1530** is RS pairs **1532** each comprising a modulated target data wavefunction derived from the probe wavefunction for the same tag data. The RS pairs **1532** are saved to superposition memory **1522**.

In the scenario that qubit associated with the dependent tag has observe false, one or more existing layers in the RS pair are modified by subtracting the layer modulated wavefunction of the dependent tag from the matching layer. This is achieved by converting the probe wavefunction corresponding to the dependent tag supplied by probe preparation **1520** to a target wavefunction by multiplication with a negative scaling factor applied to the real and imaginary components of the complex target wavefunction vector. Next, the layer and RS pair index of the dependent tag hit are passed to memory lookup by layer and RS pair **1536** to access superposition memory **1522** for the RS pair **1532** that is to be modified by the subtract action resolution. Resolve hits by command action **1528** next modifies the wavefunction superpositions in the RS pair by first generating an R modulation of the previously negative scaled target wavefunction for the layer of the dependent tag hit, and an S modulation of the previously negative scaled target wavefunction for the layer of the dependent tag hit using the previously described target preparation method shown in FIG. **4**A. The R modulation of the target wavefunction is added to the R wavefunction superposition in the RS pair, and the S modulation of the target wavefunction is added to the S wavefunction superposition in the RS pair using vector complex addition. The output of the vector complex addition is a modified RS pair **1532** that is written back to superposition memory **1522**.

Action Trace

In the case of a command with action trace **1665** the parameters are tags with associated qubits. The trace action's function shown in FIG. **16** has the formula:

$$\Gamma=\forall\psi\cdot\phi^*$$

where $\Psi\cdot\phi^*$ denotes the interference process between a plurality of RS pair wavefunction superpositions Ψ and a plurality of independent tag wavefunctions ϕ. The symbol $\forall$ denotes that the interference process detects the tag wavefunction ϕ any qubit phase that it may have. As a result, the output of the action trace denoted Γ is the presence and qubit phase angle of all instances of the tag wavefunction ϕ in the plurality of RS wavefunction superpositions denoted by Ψ.

In the case of a command with action trace **1665**, Decode commands **1518** creates four instructions that are passed to probe preparation **1520** which will rotate each of the probe wavefunctions created by complex phase angles of 0°, 90°, 180° and 270°. In this way a panel of probe wavefunctions is created for the same trace tag. The commands and instructions are also passed from Decode commands **1518** to Resolve hits by command action **1528**.

Following the interference process **1526** between the probe wavefunctions output from probe preparation **1520** and RS pairs **1524** that were retrieved from superposition memory **1522**, Resolve hits by command action **1528** processes trace commands in the output of step **1526** in groups of the four probe wavefunctions at complex phase angles of 0°, 90°, 180° and 270° in the panel for each trace command tag.

Resolve hits by command **1528** next evaluates the complex phase angle of each layer in the collection of RS pairs matching the trace tag. Resolve hits by command action will process a group of four probe wavefunction indexes corresponding to the four probe wavefunctions at complex phase angles of 0°, 90°, 180° and 270° in the panel to evaluate the pair of two highest correlations to select a pair of adjacent complex phase angles, then measure the complex phase angle of the layer for that tag in the RS pair. For example if the two highest correlations are at 0° and 90°, the complex phase angle ω is estimated using the formula ω=arctangent (correlation at 90°/correlation at 0°).

Since the output of interference process **1526** comprises different combinations of probe wavefunction and RS pair, it is range of correlation values addressable by probe index, RS pair index and layer index. In the case of action trace all tags are processed with probe index as the outer loop and RS pair as the inner loop and layer as the inside loop. The complex phase of each layer in the collection of RS pairs matching the trace tag is recorded together with the tag, so that a response to the client can be formatted comprising a layer occupancy of each RS pair. The layer occupancy of an RS pair comprises a manifest of the occupied layer indexes with tags and the associated qubit complex phase angle of the wavefunction encoded from the tag.

Action Registers

In the case of a command with action registers **1670** there are no parameters. The register action's function shown in FIG. **16** has the formula:

$$\Pi=\forall\psi$$

where the symbol $\forall$ denotes all occupied layers in a plurality of RS pair wavefunction superpositions Ψ, detects all tag wavefunction layers at whatever qubit phase they may have. As a result, the output of the action registers denoted Π is the presence and qubit phase angle of all instances of all tag wavefunctions in the plurality of RS wavefunction superpositions denoted by Ψ.

In the case of a command with action registers **1670**, Decode commands **1518** receives no tags and creates a single instruction with action registers. The command and instruction are passed from Decode commands **1518** to Resolve hits by command action **1528**. No instructions are passed from Decode commands **1518** to probe preparation **1520**. No RS pairs are retrieved from superposition memory **1522** and the interference process **1526** does not produce any output for resolve hits by command action **1528**.

Resolve hits by command action **1528** processes the command and instruction with action registers by sending an exhaustive list of RS pairs and layers in RS pairs to memory lookup by layer and RS pair **1536**, which will access all occupied layers of all RS pairs to retrieve the corresponding tags and contents **1538** from memory **1540**. Resolve hits by command action **1528** formats a response to the client that presents each RS pair as a quantum field register comprising occupied and unoccupied layers. Each occupied layer is detailed in the response by its index, a tag and an associated qubit complex phase angle of the wavefunction encoded from the tag.

Action Clear

In the case of a command with action clear **1675** there are no parameters. The register action's function shown in FIG. **16** has the formula:

$$\Pi=\{\ \}/\emptyset$$

where { }/Ø denotes that both the wavefunction superposition space embodiment in RS pairs is reset to the empty wavefunction space Ø and the content stored in the database is also cleared to the empty space comprising no content { }. As a result, the output of the action clear denoted Π is the that all instances of all tag wavefunctions in the plurality of RS wavefunction superpositions have been removed entirely, together with all content data associated with the tag.

In the case of a command with action clear **1675**, Decode commands **1518** receives no tags and creates a single instruction with action clear. The command and instruction are passed from Decode commands **1518** to Resolve hits by command action **1528**. No instructions are passed from Decode commands **1518** to probe preparation **1520**. No RS pairs are retrieved from superposition memory **1522** and the interference process **1526** does not produce any output for resolve hits by command action **1528**.

Resolve hits by command action **1528** processes the command and instruction with action clear by sending a reset signal to superposition memory **1522**, memory **1540** and memory lookup by layer and RS pair **1536**. The reset signal is processed by each of steps **1522**, **1540** and **1536** to clear all memory contents, returning the programmable quantum computer to the same initial or ground state following a reset and power up.

Action Convolve

In the case of a command with action convolve **1680**, the parameters are pairs of tags with associated qubits where each pair comprises an independent tag and a dependent tag. According to the instructions set architecture **1600** of the present disclosure, pairs of dependent and independent tags provide selective functionality that is conditioned on the presence of the independent tag wavefunction ϕ as a layer in a quantum register represented by an RS pair of wavefunction superpositions. The tag data may be any vector of real or complex values that encodes an arbitrary symbol sequence. The independent tag wavefunction ϕ represents the selection function that makes a conditional action by the dependent tag wavefunction θ.

For a command with action convolve **1680** the action's function shown in FIG. **16** has the formula:

$$\Psi\cdot\phi^*?\notin\Psi\Delta\theta$$

where $\Psi\cdot\phi^*$ denotes the interference process between a plurality of RS pair wavefunction superpositions Ψ and a plurality of independent tag wavefunctions ϕ. In the event that the independent tag wavefunctions ϕ is detected as a layer in an RS pair of wavefunction superpositions embodiment of a quantum register, $\phi\notin\Psi$ denotes that the independent tag wavefunctions ϕ is removed from the RS pair wavefunction superposition embodiment of the quantum register followed by the operation denoted $\Delta\theta$ in which the dependent tag wavefunction θ is multiplied as a complex vector against each of the R and S wavefunction superpositions comprising the same RS pair, to generate a new pair of R and S wavefunction superpositions comprising the vector complex multiplication product of the input RS pair and the dependent tag wavefunction θ. In this manner the wavefunction superpositions embodiment of selected quantum registers are each multiplied by the tag wavefunction. The convolve action therefore implements a scaling and phase rotation applied in the wavefunction domain to each and every wavefunction comprising the RS pair of wavefunction superpositions, except the layer that is the independent tag. As a result, the convolve action selectively makes the independent tag layer invariant. This is useful since the same conditional operator in the form of an independent tag may be used for multiple convolve actions on the same RS pair. The independent tag wavefunction is restored intact following the convolution of the other layer wavefunctions in the same RS pair.

In the case of a command with action convolve **1680**, Decode commands **1518** creates instructions for the independent tag **1404** and qubit **1408** combination and instructions for the dependent tag **1404** and qubit **1408** combination. In the case the qubit associated with the independent tag has observe true, a single instruction is created by Decode commands **1518** and passed to probe preparation **1520** which will rotate the probe wavefunction by the phase angle of the qubit **1408** associated with the independent tag **1404**. The commands and instructions are also passed from Decode commands **1518** to Resolve hits by command action **1528**.

For commands with action convolve **1680**, in the case the qubit **1408** associated with the independent tag **1404** has observe false, four instructions are created by Decode commands **1518** and passed to probe preparation **1520** which will rotate each of the probe wavefunctions created by complex phase angles of 0°, 90°, 180° and 270°. In this way a panel of probe wavefunctions is created for the same independent tag. The commands and instructions are also passed from Decode commands **1518** to Resolve hits by command action **1528**.

For the dependent tag and qubit combination in a command with action convolve, a single instruction is created by Decode commands **1518** and passed to probe preparation **1520** which will rotate the probe wavefunction by the phase angle of the qubit **1408** associated with the dependent tag **1404**. The processing of the dependent tag and qubit combination by steps **1518** and **1520** is the same for either qubit observe true or qubit observe false scenario. In each case the commands and instructions are passed from Decode commands **1518** to probe preparation **1520**. Resolve hits by command action **1528** receives the commands and instructions from **1518** and also probe wavefunctions from **1520**. The qubit associated with the dependent tag applies the observe true or false property in Resolve hits by command action **1528** to select if the convolve action preserves a copy of the RS pair selected by the independent tag wavefunction in addition to the new convolve action transformed RS pair. The dependent tag wavefunctions for action convolve produced in step **1520** do not pass to the interference process **1526**, instead passing directly to Resolve hits by command action **1528**.

Following the interference process **1526** between the independent tag probe wavefunctions output from probe preparation **1520** and RS pairs **1524** that were retrieved from superposition memory **1522**, Resolve hits by command action **1528** processes convolve commands in the order of the independent tags followed by the dependent tags. Since the output of interference process **1526** comprises different combinations of probe wavefunction and RS pair, it is a range of correlation values addressable by probe index, RS pair index and layer index. In the case of action convolve all

independent tags are processed with probe index as the outer loop, RS pair as the inner loop and layer as the inside loop. All of the RS pairs matching the independent tag will be selected for convolve action by the dependent tag in the same convolve command.

Following selection of RS pairs by detection of the independent tag wavefunction, Resolve hits by command action **1528** creates an intermediate hit resolution convolve list comprising RS pairs with both the independent tag hit and the dependent tag in a pair. For each item in the intermediate hit resolution convolve list Resolve hits by command action **1528** performs a sequence of operations on the list item's RS pair comprising: (a) retrieving the RS pair; (b) removing the independent tag wavefunction from the RS pair superpositions to generate a modified RS pair; (c) vector complex multiplication of the modified RS pair's R and S wavefunction superpositions by the dependent tag wavefunction; (d) restoring the independent tag wavefunction to the modified RS pair of superpositions to generate a final output RS pair for the specified convolve action. The final output RS pair for the specified convolve action is output by Resolve hits by command action **1528** as RS pairs **1524** that are saved to superposition memory **1522**.

In the operation of retrieving the RS pair Resolve hits by command action **1528** passes the layer index and the RS pair index to the memory lookup by layer and RS pair **1536**, which converts these inputs to a location in superposition memory **1522** and retrieves the RS pair **1534** containing the hit layer for the independent tag.

The step of removing the independent tag wavefunction from the RS pair superpositions to generate a modified RS pair shares some of the steps that were used for the store action resolution to add a new layer. The independent tag probe wavefunction is converted to a target wavefunction as described for store however the real and imaginary components of each element of the target wavefunction complex vector are each negated. The negated target wavefunction is modulated by the layer index in the same way as for store to generate an R modulated target wavefunction and an S target wavefunction and a vector complex addition step is also used to combine these with the R and S superpositions respectively in an RS pair. The tag wavefunctions at the hit layer in the RS pair will be annihilated by the R modulated negated target wavefunction and an S modulated negated target wavefunction.

The step of restoring the independent tag wavefunction from the RS pair superpositions to generate a modified RS pair also shares some of the steps that were used for the store action resolution to add a new layer. The independent tag probe wavefunction is converted to a target wavefunction as described for store with the real and imaginary components of each element of the target wavefunction complex vector. The target wavefunction is modulated by the layer index in the same way as for store to generate an R modulated target wavefunction and an S target wavefunction and a vector complex addition step is also used to combine these with the R and S superpositions respectively in an RS pair. The tag wavefunctions at the hit layer in the RS pair will be restored by the R modulated target wavefunction and an S modulated target wavefunction.

Each of the steps above are performed for action convolve to create a new RS pair for any RS pair quantum register that contained a detectable independent tag wavefunction according to the qubit associated with the independent tag. In the case that the qubit associated with the dependent tag has Boolean observe true the new convolved RS pair embodiment of a quantum register replaces the RS pair that contained the detectable independent tag wavefunction. In the alternate case that the qubit associated with the dependent tag has Boolean observe false the new convolved RS pair embodiment of a quantum register exists henceforth alongside the RS pair that contained the detectable independent tag wavefunction.

Action Transform

In the case of a command with action transform **1685**, the parameters are pairs of tags with associated qubits where each pair comprises an independent tag and a dependent tag. According to the instructions set architecture **1600** of the present disclosure, pairs of dependent and independent tags provide selective functionality that is conditioned on the presence of the independent tag wavefunction ϕ as a layer in a quantum register represented by an RS pair of wavefunction superpositions. The tag data may be any vector of real or complex values that encodes an arbitrary symbol sequence. The independent tag wavefunction ϕ represents the selection function that makes a conditional action by the dependent tag wavefunction θ.

For a command with action transform **1685** the action's function shown in FIG. **16** has the formula:

$$\{\phi_{t+1},\theta_{t+1},F_{t+1}\}=\Psi\cdot\phi_t^*?$$

$$c_i=\Psi\cdot\theta_t^*?$$

$$\theta_{t+1}=\Sigma c_i\cdot\theta_{i,t+1}$$

$$F_{t+1}(\phi_{t+1},\theta_{t+1})$$

where $\Psi\cdot\phi^*$ denotes the interference process between a plurality of RS pair wavefunction superpositions Ψ and a plurality of independent tag wavefunctions ϕ. The subscript t in ϕt is used to denote independent tag wavefunctions ϕ at a particular time slot corresponding to one cycle of the method **1500** shown in FIG. **15**. Following cycle t is cycle t+1 so the subscript t+1 in $\phi t+1$ is used to denote wavefunctions in cycle t+1. In the event that the independent tag wavefunctions ϕ are detected as a hit in a layer in an RS pair of wavefunction superpositions embodiment of a quantum register, the payload at that hit layer is retrieved from memory **1540** using references to the RS pair and the hit layer. The payload retrieved from memory is denoted as $\{\phi_{t+1}, \theta_{t+1}, F_{t+1}\}$ and comprises (a) an independent tag ϕ_{t+1}, (b) a dependent tag θ_{t+1} and (c) a function descriptor F_{t+1} which are applied to create a command for the next cycle of method **1500** shown in FIG. **15**.

The complex coefficient c_i equal to the interference of the dependent tag wavefunctions θ_t^* with RS pair wavefunction superpositions Ψ is applied as a transformation of the next cycle's dependent tag probe wavefunction $\theta_{i,t+1}$ such that separate elements θ_i denoted by the index i comprising the dependent tag wavefunctions $\theta_{i,t+1}$ are the product of complex multiplication by coefficient c_i, followed by summation for all i as in the formula $\theta=\Sigma c_i\cdot\theta_{i,t+1}$. The transform action therefore implements a scaling and phase rotation c_i applied to each of the separate elements $\theta_{i,t+1}$ denoted by the index i comprising the dependent tag wavefunction θ for cycle t+1.

One output of action transform **1685** is $\theta_{i,t+1}$ the transformed dependent tag probe wavefunction θ for cycle t+1 comprising the separate elements denoted by the index i transformed by the complex coefficient c_i that is derived from a layer interference complex value ck denoting the complex interference value at layer k. The mapping between layer index k and separate element index i may be any predefined function.

A second output of action transform **1685** is function descriptor F_{t+1} which is applied to create an output comprising a command for the next cycle of method **1500** shown in FIG. **15** where the independent tag ϕ_{t+1} and dependent tag θ_{t+1} are arguments for a command with action specified by function descriptor F_{t+1}.

It is a feature of the method in the present disclosure that the interference value of non-aligned sequences comprising tag elements is zero and the interference value of aligned sequences comprising matching tag elements is unity. It is a further feature of the method which is the subject of the present disclosure that any sequence comprises a plurality of non-overlapping partitions that may be generated by any number of partitioning functions. Consequently each of the non-overlapping partitions of orthogonal elements functionally correspond to independent variables in a computational model.

In the simplest example the independent tag wavefunction ϕ_t comprises the second orthogonal domain representation of a first single tag element and the dependent tag probe wavefunction θ_t comprises the second orthogonal domain representation of a second single tag element. Also, for the purpose this simplest example, let the RS pair wavefunction superpositions Ψ comprise a layer of the first single tag element matching the independent tag probe wavefunction ϕ_t. The layer of the first single tag element matching the independent tag probe wavefunction ϕ_t is furthermore associated with a payload that was stored in memory **1540** and which comprises a next cycle independent tag wavefunction ϕ_{t+1}, a next cycle dependent tag wavefunction θ_{t+1} and a next cycle function descriptor F_{t+1} specifying a command action **1426**. In this case, the command with action transform **1685**, where the independent tag is the first tag element and the dependent tag is the second tag element, will generate a unity modulated or identical next cycle tag probe wavefunction θ_{t+1} representing the next cycle dependent second tag element as a result of the unity interference value of c_i, the interference of the dependent tag wavefunctions θ_t and the RS pair wavefunction superpositions W. The unity value of c_i is applied to the dependent second tag element so that in this simplest case the output probe wavefunction θ_{t+1} is the second orthogonal domain representation of the dependent second tag element θ_t in the command with action transform. So, in this example the dependent tag probe θ_t is transformed to the next cycle dependent tag probe θ_{t+1} which becomes input for the next cycle command with action specified by the next cycle function descriptor F_{t+1}. It will be apparent that the transform actions are determined by the both the selection of independent and dependent tags in the transform command and state of the RS pair wavefunction superpositions at the time of the action.

In the case of a command with action transform **1685**, Decode commands **1518** creates instructions for the independent tag **1404** and qubit **1408** combination and instructions for the dependent tag **1404** and qubit **1408** combination. In the case the qubit associated with the independent tag has observe true, a single instruction is created by Decode commands **1518** and passed to probe preparation **1520** which will rotate the probe wavefunction by the phase angle of the qubit **1408** associated with the independent tag **1404**. The commands and instructions are also passed from Decode commands **1518** to Resolve hits by command action **1528**.

For commands with action transform **1685**, in the case the qubit **1408** associated with the independent tag **1404** has observe false, four instructions are created by Decode commands **1518** and passed to probe preparation **1520** which will rotate each of the probe wavefunctions created by complex phase angles of 0°, 90°, 180° and 270°. In this way a panel of probe wavefunctions is created for the same independent tag. The commands and instructions are also passed from Decode commands **1518** to Resolve hits by command action **1528**.

For the dependent tag and qubit combination in a command with action transform, a single instruction is created by Decode commands **1518** and passed to probe preparation **1520** which will rotate the probe wavefunction by the phase angle of the qubit **1408** associated with the dependent tag **1404**. The processing of the dependent tag and qubit combination by steps **1518** and **1520** is the same for either qubit observe true or qubit observe false scenario. In each case the commands and instructions are passed from Decode commands **1518** to probe preparation **1520**. Resolve hits by command action **1528** receives the commands and instructions from **1518** and also probe wavefunctions from **1520**. The qubit associated with the dependent tag applies the observe true or false property in Resolve hits by command action **1528** to select if the transformed dependent probe output of Resolve hits comprises a first orthogonal domain representation in the form of a tag **1404** and associated qubit **1408** in the case the observe is true, or a second orthogonal domain representation comprising a probe wavefunction representation of the tag element in the case the observe property of the qubit associated with the dependent tag is false. The dependent tag wavefunctions for action transform produced in step **1520** do not pass to the interference process **1526**, instead passing directly to Resolve hits by command action **1528**.

Following the interference process **1526** between the independent tag probe wavefunctions output from probe preparation **1520** and RS pairs **1524** that were retrieved from superposition memory **1522**, Resolve hits by command action **1528** processes transform commands in the order of the independent tags followed by the dependent tags. Since the output of interference process **1526** comprises different combinations of probe wavefunction and RS pair, it is a range of correlation values addressable by probe index, RS pair index and layer index. In the case of action transform all independent tags are processed with probe index as the outer loop, RS pair as the inner loop and layer as the inside loop. All of the RS pairs matching the independent tag will be selected for transform action by the dependent tag in the same transform command.

Following selection of RS pairs by detection of the independent tag wavefunction, Resolve hits by command action **1528** creates an intermediate hit resolution transform list comprising RS pairs with both the independent tag hit and the dependent tag in a pair. For each item in the intermediate hit resolution transform list Resolve hits by command action **1528** performs a sequence of operations: (a) for the independent tag hit obtaining the payload associated with the hit layer comprising the next cycle dependent tag, independent tag and function descriptor; (b) for the dependent tag θ_t obtaining the interference values ck denoting the real or complex interference value at layer k; (b) from the ck values deriving complex coefficient c_i by any predefined mapping or functional equation; (c) vector complex multiplication transformation of the next cycle dependent tag probe wavefunction θ_{t+1} such that separate elements θ_i denoted by the index i comprising the dependent tag wavefunctions θ_{t+1} are the product of complex multiplication by coefficient c_i, followed by summation for all i as in the formula $\theta_{t+1}=\Sigma c_i \cdot \theta_{t+1,i}$.

The next cycle function descriptor F t+1 obtained from the independent tag hit is used by Resolve hits by command action **1528** to synthesize a new command for the next cycle of method **1500** that comprises (a) an independent tag ϕ_{t+1} and (b) a dependent tag θ_{t+1}.

As described above, the dependent tag θ_{t+1} retrieved from memory **1540** when the independent tag wavefunction ϕ is detected as a hit in a layer in an RS pair of wavefunction superpositions embodiment of a quantum register has an associated qubit **1408** with the Boolean observe property that is either true or false. In the case the observe property associated with the dependent tag is false the dependent tag θ_{t+1} comprises the second orthogonal domain representation of a probe wavefunction representation of the tag element. Conversely, if the observe property associated with the dependent tag is true the dependent tag θ_{t+1} comprises the first orthogonal domain representation of the tag element.

According to the method of the present disclosure the transformation of the next cycle dependent tag wavefunction may in some cases be efficiently achieved downstream by means of the qubit **1408** associated with a tag **1404** by making use of the complex phase rotation by qubit **1408** that will be applied to tag **1404** by steps Decode Commands **1518** and Probe preparation **1520** in the next cycle of method **1500**.

In the case that the command action specified in the next cycle function descriptor F_{t+1} has a single wavefunction argument, either the next cycle independent tag or next cycle dependent tag may be omitted. The new command comprising the next cycle independent tag and associated qubit, the next cycle dependent tag and associated qubit and the command action **1426** specified by the next cycle function descriptor is marked as executable and passed to step **1542** which sends the command to Sort **1508** to be processed by method **1500** in the next cycle.

Quantum Computing Applications Using Lie Algebra

The present disclosure describes methods to related to a second representation of data as a plurality of wavefunctions in wavefunction superpositions where each wavefunction has an orthogonal layer encoding and is also characterized by a complex phase determined by the associated qubit phase. As stated above, the qubit phase **37** may be represented as a complex number that for example determines the selections in a circular space comprising the complex phase of individual layers in the wavefunction superposition RS pair. Interference of a probe wavefunction with the target wavefunction superposition as described previously calculates correlations of the probe tag and the target tag that are complex numbers.

In some embodiments the wavefunction comprising a second representation of a data tag **1404** may be encoded to pairs of layers in the same or different wavefunction superpositions, such that the combined magnitude of both layers is equal to unity. For example, a complex number 'α' may be applied as qubit phase **37** to encode the tag to a wavefunction in the first layer of the pair of layers, and a second complex number 'β' may be applied as qubit phase **37** to encode the tag to a wavefunction in the second layer of the pair of layers, where the combined magnitude of the complex numbers 'α' and 'β' is equal to unity, so that $\alpha^2+\beta^2=1$.

Quantum computing applications often use the Lie algebra of the SU(2) Special Unitary Group of 2×2 matrices to operate on C^2 qubits in a two-dimensional vector space over the complex numbers. A C^2 qubit is a vector of two complex numbers (α,β) where the combined magnitude of the complex numbers 'α' and 'β' is equal to unity, so that $\alpha^2+\beta^2=1$.

From the foregoing it will be apparent that a C^2 qubit as used with the SU(2) matrices in quantum computing applications may be implemented as a pair of layers in a wavefunction superposition, the first encoded with a qubit phase **37** represented by the complex number 'α' and the second encoded with a qubit phase **37** represented by the complex number 'β' where the combined magnitude of the complex numbers 'α' and 'β' is equal to unity, so that $\alpha^2+\beta^2=1$.

Action Quantum Gate

In the case of a command with action quantum gate **1690**, the parameters comprise a plurality of tags with associated qubits where each plurality comprises: an independent tag; one or more dependent tags; and a pair of complex vectors associated with each dependent tag. The pair of complex vectors associated with each dependent tag each comprises one row of a quantum gate matrix such that each complex vector in the pair of vectors contains two complex numbers for each of the dependent tags. As a result the quantum gate matrix has a total dimension of rows and of columns equal to twice the number of dependent tags. Each combination of two dependent tags is therefore associated with a 2×2 complex matrix cell in the quantum gate matrix. According to the instructions set architecture **1600** of the present disclosure, a plurality of dependent and independent tags provide selective functionality that is conditioned on the presence of the independent tag wavefunction ϕ as a layer in a quantum register represented by an RS pair of wavefunction superpositions. The tag data may be any vector of real or complex values that encodes an arbitrary symbol sequence. The independent tag wavefunction ϕ represents the selection function that makes a conditional action by the dependent tag wavefunctions θ.

For a command with action quantum gate **1690**, the action's function shown in FIG. **16** has the formula:

$$\Psi_t \cdot \phi^* ?\alpha,\beta = \Psi_t \cdot \theta^*$$

$$\Psi_{t+1} = W(\alpha,\beta)$$

where $\Psi_t \cdot \phi^*$ denotes the interference process between a plurality of RS pair wavefunction superpositions Ψ_t and a plurality of independent tag wavefunctions ϕ. In the event that the independent tag wavefunctions ϕ is detected as a layer in an RS pair of wavefunction superpositions embodiment of a quantum register, the dependent tag wavefunction θ is interfered with the RS pair of wavefunction superpositions to read a plurality of pairs of complex correlation coefficients denoted 'α' and 'β', each pair equal to the qubit phase angle of the dependent tag in two separate layers of the RS pair of wavefunction superposition. In the formula above for action quantum gate Ψ denotes the quantum gate matrix and $W(\alpha,\beta)$ represents a column vector complex product of the quantum gate matrix and the plurality of 'α' and 'β' pairs that are the qubit phase angles of each dependent tag in two separate layers of the RS pair of wavefunction superposition. As previously described, quantum computing algorithms using SU(2) Lie algebra operate on C^2 qubits comprising a two-dimensional vector space over the complex numbers, eg. (α,β) where $\alpha^2+\beta^2=1$, and may be represented by wavefunctions located in a pair of layers in a wavefunction superposition. The subscript t in Ψ_t is used to denote the collective state of the plurality of RS pair wavefunction superpositions at a particular time slot 't' corresponding to one cycle of the method **1500** shown in

FIG. **15**. Following cycle t is cycle t+1 so the subscript t+1 in Ψ_{t+1} is used to denote the modified state of the plurality of RS pair wavefunction superpositions in cycle t+1 following the quantum gate operation.

In the case of a command with action quantum gate **1690**, Resolve hits by command action step **1528** will detect if any hits exist in the interference process **1526** output. As a first step the hits associated with the dependent tags serve to select the RS pairs to be further processed by the quantum gate. As a second step, pairs of hit layers of the dependent tag in the selected RS pairs are read as complex number correlations to give a two-dimensional complex vector (α,β) for each dependent tag, where $\alpha^2+\beta^2=1$.

The position of a hit in the interference process output represents a specific combination of probe and RS pair input to interference process **1526**. Accordingly, the position of a hit in the interference process output is decoded to a probe index, a layer index and an RS pair index. The layer index and RS pair index are passed from step **1528** to memory lookup by layer and RS pair **1536**. The RS pair index is passed to Superposition Memory **1522** which retrieves the designated wavefunction superposition data comprising the returns the selected RS pair **1534** to Resolve Hits by command action **1528**.

In the case of command with action quantum gate Resolve Hits by command action **1528** also uses the probe index of dependent tag hits to obtain the pair of complex vectors associated with each dependent tag each that comprises one row of a quantum gate matrix. The complete quantum gate matrix for all dependent tags previously denoted Ψ can therefore be formed by Resolve Hits by command action **1528** from the pair of complex vectors associated with each dependent tag.

The C^2 qubits comprising the two-dimensional complex vector (α,β) for each dependent tag are next transformed by the quantum gate matrix previously denoted Ψ to $W(\alpha,\beta)$ comprising a column vector complex product of the quantum gate matrix and the plurality of 'α' and 'β' pairs that are the qubit phase angles of each dependent tag in two separate layers of the RS pair of wavefunction superposition.

The RS pairs selected by each dependent tag in the command with action quantum gate and retrieved from memory are now transformed to represent a new set of C^2 qubit states by Resolve Hits by command action **1528**. Each pair of layers for each dependent tag is modified to represent the new two-dimensional complex vector (α,β) that is one row in the column vector $W(\alpha,\beta)$ complex product of the quantum gate matrix and the previous state.

The process of modifying pairs of dependent tag layers in the retrieved RS pair is similar to the modification of individual layers previously described for other command actions. Layer modification is achieved by first generating a pair of R and S modulated target wavefunctions for the dependent tag layer in the RS pair at the current estimated complex qubit phase **37** as a first modifying component. Next, a pair of R and S modulated target wavefunctions are generated for the dependent tag layer in the RS pair at the new complex qubit phase angle **37** as a second modifying component. The R and S wavefunctions in the first modifying component are subtracted from the R and S wavefunctions respectively in the second component by vector complex addition to generate modifying R and S wavefunctions. The modifying R wavefunction is then combined with the R wavefunction in the dependent tag RS pair by vector complex addition. The modifying S wavefunction is the combined with the S wavefunction in the dependent tag RS pair by vector complex addition. The effect of the preceding steps is to replace the layer in the RS pair for the dependent tag with a starting qubit phase **37** with the same layer having a final qubit phase **37**.

Following modification of the pairs of layers of selected dependent tag wavefunctions in the retrieved RS pairs, Resolve Hits by command action **1528** writes the modified RS pairs **1534** to superposition memory **1522** to complete the operation of action quantum gate.

Action Solve

In the case of a command with action solve **1695**, the parameters comprise: a plurality of tags with associated qubits; an algorithm type to be applied; and a plurality of coupling factors relative to each other tag in the plurality of tags. As shown in FIG. **14** the command action **1426** for solve comprises the list **1428** of a plurality of items **1402**. In the case of a command with action solve action a vector of real or complex values provides the plurality of coupling factors of the tag **1404** in item **1402** relative to each other tag in the plurality of tags. The tag data may be any vector of real or complex values that encodes an arbitrary symbol sequence. The solve action's function shown in FIG. **16** has the formula:

$$\Psi_t \cdot \phi^* ? \$(\mathrm{J}) \Rightarrow \omega_{t+1}$$

where Ψ_t represents a plurality of RS pairs each comprising a wavefunction superposition, ϕ^* represents the tag data conjugate wavefunction, ? denotes the interference process and \$(J) represents the solve process **1529** application of coupling factor constraints J to the RS pairs selected by the interference process. As previously mentioned Ψ_t represents a collection of RS pairs of wavefunction superpositions at a particular time slot 't' corresponding to one cycle of the method **1500** shown in FIG. **15**. Following cycle t is cycle t+1 so the subscript t+1 in Ψ_{t+1} is used to denote the modified state of the plurality of RS pair wavefunction superpositions in cycle t+1 following the solve operation. In this formula represents the solve process **1529** in method **1500** and Ψ_{t+1} denotes the final solved state which achieves the lowest mutual conflict of the selected RS pairs for the coupling factor constraints J provided in the command with action solve.

A particular tag in the command with action solve may select any number of matching RS pairs including none. The coupling factors of each tag versus the other tags will be applied collectively to all of the RS pairs matching each respective tag. Consequently the solve process **1529** described below may have a plurality of RS pairs that are included for each input tag and its associated coupling factors.

In the case of a command with action solve **1695**, Resolve hits by command action step **1528** will detect if any hits exist in the interference process **1526** output. The position of a hit in the interference process output represents a specific combination of probe and RS pair input to interference process **1526**. Accordingly, the position of a hit in the interference process output is decoded to a probe index, a layer index and an RS pair index. The layer index and RS pair index are passed from step **1528** to memory lookup by layer and RS pair **1536**. The RS pair index is passed to Superposition Memory **1522** which retrieves the designated wavefunction superposition data comprising the returns the selected RS pair **1534** to Resolve Hits by command action **1528**.

The RS pairs selected by each tag in the command with action solve are associated with the coupling factor of that

tag relative to all other tags. The total number of tags is expanded to include the total number of RS pairs selected by the total number of tags in the solve command or commands. The total number of RS pairs selected is passed by Resolve Hits by command action **1528** to solve process **1529**.

In the case of commands with action solve the coupling factor constraints are passed from decode commands **1518** to Resolve Hits by commands action **1528**. As a result the probe index from each decoded hit may be used to obtain the coupling factor constraints of the probe tag relative to all other tags in the solve command or commands. Accordingly, Resolve Hits by command action also passes the coupling factor constraints as an input to solve process **1529**. In addition, the particular solve algorithm for which the coupling factors apply is also passed from the command with action solve via decode commands **1518** to Resolve Hits by command action **1528** and to solve process **1529**.

In the preferred embodiment of the present invention the solve process **1529** performs the method **1700** depicted in FIG. **17** which is described below in detail.

The output of solve process **1529** is a rotated version of the set of wavefunction superposition RS pairs that were input to the process. The rotation of the qubit phase of each RS pair is a response to the solve algorithm type and the collective coupling factors between RS pairs. Method **1700** described below resolves the collective constraints to progressively approach a lowest energy state determined by the minimum conflict of the qubit phase of the superposition wavefunction RS pairs according to the algorithm type and the collective coupling factors between respective RS pairs.

The solve process described in the present disclosure has the novel feature of resolving a plurality of wavefunction superposition layers in each RS pair combination. Each RS pair has hundreds of available layers according to the present invention, and furthermore only similar data on equivalent layers will contribute to the interference between RS pairs by method **225** shown in FIG. **5**M. Conversely, layers that are empty or have a different data type that have been encoded by an orthogonal basis will not interact in method **225**. Specifically, method **225** performs a reading and summation of matching corresponding layers to arrive at a singe complex number **291** representing a complex correlation of two RS pairs and consequently provides an angular difference of the first RS pair's qubit phase angle relative to the second RS pair's qubit phase angle.

The output of solve process **1529** comprising a rotated version of the set of wavefunction superposition RS pairs input to **1529** and represents a potentially new superposition wavefunction register state of the method **1500**. The successful completion of solve process **1529** may be evaluated by the reduction of the energy as defined by the collective qubit phase of each RS pair and their respective coupling factors. Either a reduction to an absolute low energy pass threshold or a significant proportional reduction relative to an initial energy state may qualify as a successful completion of solve process **1529**.

The lower energy state as defined by the collective energy of the rotated qubit phases of each RS pair with respect to their mutual coupling factors that comprises the output of solve process **1529** may be saved back to the superposition memory **1522** by Resolve Hits by command action **1528** via RS pairs **1534**.

According to the present method Resolve Hits by command action **1528** outputs the results of the commands with action solve as an update to the qubit phase of all the interference matched RS pairs and also optionally generates a report in the form of structured data for the modified qubit phase of the affected RS pairs that is forwarded to send response to client **1544** and hence to the client originating the commands.

FIG. **17** illustrates a flowchart of a method **1700** depicting a general-purpose qubit phase 'spin' alignment solve process that may be applied to different combinatorial problems comprising a plurality of items in a solve list **1705**, each with relative constraints with respect to other items. The relative constraints are represented by coupling factors **1710** between RS pairs that represent the data of each item in the list of items to solve in the form of wavefunction superpositions. The solve list **1705** and the coupling factors **1710** are input to an iterative process starting with step **1715** comprising an outer loop for successive iterations of an inner loop method. The inner loop method starts with step **1720** and calculates a qubit phase rotation for each item in the solve list.

The steps comprising the inner loop of method **1700** are depicted in greater detail by the collected graphical representations of inputs, steps and intermediate states shown in FIG. **18**. The coupling factors **1710** between RS pairs in FIG. **17** is depicted graphically in FIG. **18** as a coupling grid **1810**. The items in solve list **1705** in FIG. **17** are depicted graphically as the diagonal elements from top left to bottom right of a data grid **1822** in FIG. **18**. The inner loop of method **1700** proceeds along the diagonal of the data grid **1822** in a specific pattern of processing the current row then filling the current column. This pattern has the feature of progressively incorporating the rotation of each item in the processing of the next item in the solve list. By incorporating its history, method **1700** has the desirable properties associated with a 'progressive solve' method.

Depending on the specific algorithm an energy metric may be evaluated for the qubit phases of each item in the set of RS pairs relative to the coupling factor specified between different items. In the present example the coupling factors are represented by real or complex numbers in a coupling matrix according to the specific algorithm that is being applied.

An important qubit phase or 'spin' alignment algorithm is a generalized solve process described below. The generalized solve problem has problem constraints determined by a set of coupling factors **1710** between RS pairs of wavefunction superpositions **1705**. The solve process evaluates the RS pair interaction versus the coupling factor constraints and evolves the set of RS pairs towards an optimal solution by rotation of the qubit phase of selected RS pairs. Step **1725** performs the interference of RS pairs to measure their qubit phase difference by method **225** shown in FIG. **5**M. The qubit phase differences measured in step **1725** are between the current item T in the solve list and each item 'j' above it in the list, denoted 'j>i'. The measured qubit phase differences for 'j>i' from step **1725** are input to step **1730** together with the qubit phase differences for 'j<i' comprising previously calculated qubit phase differences that have been adjusted for relative rotation of the items below and before the current T.

Following step **1725**, step **1730** processes the qubit phase differences of all items versus the current item 'i', comprising the 'j<i' and the 'j>i' qubit phase differences, by first quantizing the angle within a plurality of angular ranges of the unit circle. A histogram is then generated by calculating the relative frequency of quantized qubit phase differences for each quantization bin in the plurality of angular ranges of the unit circle.

The histogram calculated in step **1730** then passes to step **1735** that selects the phase rotation to apply to the current RS pair 'i' according to the particular solve algorithm being employed.

The phase rotation calculated in step **1735** is applied to the current RS pair according to the above description of qubit phase **37** with reference to FIG. **4**A: the qubit phase may also be subsequently changed to a new value by the same delta phase rotation being applied to the R wavefunction superposition and the S wavefunction superposition by a complex multiplication of each complex vector element by a complex number equal to the qubits phase change angle, eg. complex rotation=(cos(θ), i·sin(θ)).

Following the rotation of the current RS pair 'i' in step **1735**, the phase rotation calculated in step **1735** passes to step **1740** where the qubit phase differences of the yet to be processed items in the solve list of RS pairs are adjusted for the rotation applied to the current RS pair 'i' in step **1735**.

According to the preferred embodiment depicted graphically in FIG. **18**, a data grid **1822** comprises a matrix of qubit phase differences between different RS pairs in the solve list. For a solve list of 'L' items, data grid **1822** will comprise L rows and L columns. The grid diagonal **1826** is traversed from top left to bottom right as items in the solve list are processed by the inner loop of method **1700**. An individual element in the data grid is designated by row and column coordinates. According to the present description the inner loop of method **1700** comprising the steps **1720** through **1745**, each iteration processes a row 'i' of the data grid **1822**. Each row of the data grid **1822** comprises 'L' columns, where the column index 'j' may be either greater than 'i' for the right half of the row above the diagonal, or the column index 'j' may be less than 'i' for the left half of the row below the diagonal.

The interference between RS pair 'i' and RS pair 'j', where j>i measured the qubit phase of RS pair 'i' minus the qubit phase of RS pair 'j' in step **1725**, is used in step **1740** to derive transpose elements in the data grid **1822**. The transpose elements fill the column below the current RS pair on the diagonal by transposing the above the diagonal row processed in step **1725**. In addition, the rotation of the current RS pair 'i' calculated in step **1735** is applied to each of the transposed elements. From the calculation of the qubit phase difference in step **1725** it follows that $\Delta\Theta(j,i)=-\Delta\Theta(i,j)$. The phase rotation **60**(*i*) applied to RS pair 'i' can be represented in polar form as an adjusted qubit phase $\Psi\theta(i)'=\Psi\theta(i)+\delta\theta(i)$. In step **1740** $\Delta\Theta(j,i)=\Psi\theta(j)-\Psi\theta(i)$ therefore the adjusted value denoted by $\Delta\Theta(j,i)=\delta\theta(j)-\Psi\theta(i)'=\Psi\theta(j)-\Psi\theta(i)-\delta\theta(i)$. Consequently the transpose elements in the grid **1822** may be calculated by negating the qubit phase difference calculated by interference in step **1725** and subtracting the phase rotation $\delta\theta(i)$ applied to the current RS pair 'i' in step **1735**.

Following step **1740** is the inner loop iteration test of the row index 'i' being less than the number of items 'L' in the solve list: if it is less than the total, the next value of 'i' is processed by returning to step **1720**.

After processing all solve items in the list step **1745** passes to step **1750** which calculates an energy metric H based on the collective state of the qubit phases of the RS pairs in the solve list, denoted G, with relative qubit phases of each evaluated according to a specified solve algorithm and the mutual coupling factors between qubit phases, denoted F, that were supplied as constraints to the solve process.

In the case where the specified solve algorithm is a generalized 'spin glass' quantum spin alignment algorithm the coupling factors determine the desired or lowest energy qubit phase relationship between each RS pair. For example a +1 coupling factor between two RS pairs may indicate qubit phase alignment, or a delta qubit phase of zero equal to arccos(+1), while a −1 coupling factor between two RS pairs may indicate qubit phase opposite-alignment, or a delta qubit phase of 180° or π radians equal to arccos(−1).

In the case where the specified solve algorithm is the generalized 'four color' problem that has been formulated as a quantum spin alignment algorithm, the present method has the novel feature of using a mapping of four arbitrary colors to four separated qubit phases such as 0°, 90°, 180°, 270°. the coupling factor between RS pairs may indicate whether the two 2D map regions, each represented by one of two RS pairs, are connected in which case the coupling factor may be −1, indicating the regions must not have the same color, or equivalently in this algorithm, the same qubit phase. Alternatively, if the two 2D map regions, each represented by one of two RS pairs, are not connected the coupling factor may be 0, indicating the regions may have the same color, or qubit phase.

The energy metric H based on the collective state of the qubit phases of the RS pairs in the solve list, denoted G, and specified solve algorithm and coupling factors F calculated in Evaluate Metric block **1750** passes to step termination condition **1755** at the bottom of the outer loop of method **1700**. The outer loop will repeat if the energy metric indicates that the solve process is incomplete. Various absolute and relative measures may be applied to the energy metric, for example being above a minimal threshold value or reaching a plateau where the relative improvement in energy metric is below a certain threshold. In addition the number of outer loop iterations and the trajectory of the energy metric may be incorporated in the termination condition **1755**.

When termination condition **1755** indicates that the solve process is incomplete another iteration of the outer loop commences with step **1760** that performs a reordering of the solve list. The precise pattern of reordering may not be important in many cases since many different reorderings will achieve the same effect of presenting the solve list in a new order for the next outer loop iteration. As a result random reordering is one possible variation for step **1760** that may be used in certain cases. In other cases a solve algorithm may employ a rank based reordering as another possible variation for step **1760**.

When termination condition **1755** indicates that the solve process is complete the outer loop exits to step **1765** where the adjusted qubit phase of each RS pair in the solve list input **1705** is output together with the energy metric H of the solve quantum register state according to the mutual coupling factors and the specified solve algorithm.

FIG. **18** illustrates a method for progressive solving of superposition wavefunction mutual interactions by incorporating each rotational adjustment of qubit phase in the calculation of subsequent adjustments of qubit phase of other superposition wavefunctions within each pass through a list of items to be solved according to an exemplary embodiment of the present disclosure. The steps comprising the inner loop of method **1700** are depicted in greater detail by the collected graphical representations of inputs, steps and intermediate states in FIG. **18**. Data grid **1810** represents the coupling factors **1710** input to method **1700** as a square matrix of dimension L times L where L is the number of items in solve list **1705**. The row and column coordinates of each element in data grid **1810** designate the pair if indices in the solve list to which the coupling factor applies. As

described above different solving algorithms use different coupling factors including real and complex numbers. The diagonal elements **1812** of data grid **1810** correspond to a self-coupling factor that may instead be interpreted as a constant value or bias value of that element. A row of coupling factors **1816** may be supplied as the coupling factors between the solve list item with row index 'i' and each of the other solve list items. In this way the data grid **1810** may be formed by a vector of coupling factors associated with each solve item as part of the structured data command with action solve. Another row of coupling factors **1814** may also be supplied as the coupling factors between a different item in solve list item and its coupled items.

The pattern of coupling factors in data grid **1810** represents the constraints between items in the solve list according to the specified solve algorithm. When there is no interaction between two items in the solve list the coupling factor will be zero or void, as depicted by the empty cells in data grid **1810**. Coupling factors **1818** located below the diagonal elements **1812** may be transposes or negative values of corresponding elements located above the diagonal, or not, depending on the specified algorithm. The coupling factors may also be sparsely distributed within data grid **1810**, as depicted by element **1820**.

According to the preferred embodiment depicted graphically in FIG. **18**, a data grid **1822** comprises a matrix of qubit phase differences between different RS pairs in the solve list. For a solve list of 'L' items, data grid **1822** will comprise L rows and L columns. The grid diagonal **1826** is traversed from top left to bottom right as items in the solve list are processed by the inner loop of method **1700**.

The progression of method **1700** with reference to data grid **1822** is illustrated by a sequence of steps shown in FIG. **18** that correspond to the inner loop steps of method **1700**. The interference of RS pairs to measure their qubit phase difference in step **1725** is illustrated graphically as step **1840** in FIG. **18**. As graphically depicted in **1840**, the elements **1842** of an unprocessed row **1830** comprising empty cells in the grid **1822** are transformed by step **1725** of method **1700** to the measured angular difference of RS pair 'i' qubit phase minus RS pair 'j' qubit phase. Following step **1725** the qubit phase difference measured for each RS pair is depicted by element **1844**. The complete output of step **1725** is illustrated as an example row **1832** with the same hash pattern.

Similarly, sample row step **1850** corresponds to the previously described step sample quantized delta theta step **1730**. The full row comprises elements **1852** below the diagonal, the diagonal element **1854** and elements above the diagonal **1856**.

In a similar correspondence, sample row step **1860** illustrates to the previously described step select delta theta step **1735**. In step **1860** the currently processed RS pair 'i' located on the diagonal of data grid **1822** with qubit phase **1862** is rotated by addition of delta theta rotation from step **1735** to the new diagonal element **1864**.

Similarly, fill column step **1870** corresponds to the previously described step fill column below diagonal step **1740**. In step **1870** the currently processed RS pair 'i' located on the diagonal of data grid **1822** with qubit phase **1862** is applied to an empty column **1872** corresponding to an empty column such as **1834** in data grid **1822**. Following step **1740** the column below the current RS pair 'i' located on the diagonal of data grid **1822** is transformed from an empty column as depicted by column **1834** to become a filled column as depicted by column **1838**. As the method **1700** outer loop progresses down the diagonal **1826** the remaining unprocessed rows shown as **1824** are transformed to measured row and filled column until the last item in the solve list has been processed.

FIG. **19** illustrates a flowchart of a method **1900** for compiling instruction streams comprising commands of structured data to execute a quantum processing algorithm. Input is presented in the form of a directed flow wiring diagram of quantum interactions from an input state to an output state. The method described below is an exemplary embodiment of a general-purpose automated compiler program flow incorporating various elements previously described according to the present disclosure.

The method **1900** is configured by an Instruction Set Architecture Schema **1920** comprising a structured data document that describes the valid commands and instructions for an implementation of a programmable quantum computer using the method **1500** as performed by a server computer that receives commands from a client. Schema **1920** corresponds to the schema **1506** previously described as a component of method **1500** which allows validation of commands sent from the client. It will be apparent that the equivalent schemae **1506** and **1920** comprising structured data descriptions of the Instruction Set Architecture constitutes a program syntax definition that may be used to define an instruction opcode space for compilations in addition to being used to validate commands after they have been submitted.

Instruction Set Architecture Schema **1920** is an input to step **1930** that generates code templates comprising subroutine functions for an Application Programming Interface or API defined by the schema **1920** as the instruction opcode space for compilations according to the present disclosure. Step **1930** may be automated based on previously written code fragments and patterns and the common reuse of input data types by different opcodes.

Method **1900** starts with an Input Quantum Network with directed flow **1910** comprising a wiring diagram of quantum interactions between an input quantum state and an output quantum state. The input quantum state comprises a plurality of input qubits which are connected by the specific topology of a quantum algorithm via a plurality of nodes to a plurality of output qubits. The nodes between input qubits and output qubits may be defined as quantum gates such as conditional NOT, CNOT, which will change the quantum state of one input qubit depending on the quantum state of the other input qubit. A feature of the present method is the great variety of different quantum gates and qubit interactions are included in the Instruction Set Architecture described herein compared with other systems that are constrained by limited types of quantum gates with few inputs. The flexibility of the present method overcomes common hardware physical limitations such as gate type and proximity that contribute to implementation complexity and when combined with noise have previously made scaling quantum networks impractical in many cases.

According to method **1900** the input quantum network **1910** may be in any computer readable data format that conforms to a quantum algorithm description language for a directed flow graph or wiring diagram. Step **1915** performs the function of transforming the input quantum network flow graph data to an Intermediate Representation or IR. Different forms of input **1910** may be transformed by step **1915** to a common output IR.

The Intermediate Representation captures the specified topology of the input quantum network **1910** to a graph of nodes connected by edges. The input quantum state corresponds to input nodes to the graph representing a plurality of

qubits, and the output quantum state corresponds to the output nodes of the graph also representing a plurality of qubits. The network flow graph as commonly depicted comprises wires in a wiring diagram that represent connections between nodes in the quantum network graph. A plurality of qubit interaction nodes are located in the quantum network between the input nodes and the output nodes. The quantum interaction nodes may comprise quantum gates of different types and other elements such as negation, conjugation, rotation and quantization of qubits.

The intermediate representation generated by step **1915** passes next to step **1925** where the Instruction Set Architecture of the intended implementation, in the form of a schema comprising a structured data representation of the data types and commands, is selected as an input from step **1920**. The nodes in the quantum network graph are mapped by step **1925** to commands in the Instruction set architecture. For example, many different quantum gates may be implemented by the command with action quantum gate as described above with reference to method **1500**. Step **1925** may also perform various optimizations of the intermediate representation according to the features of the Instruction Set Architecture.

One potential optimization performed by step **1925** comprise coalescing separate quantum gates with low numbers of input and output qubits in the input quantum network **1910** to a single monolithic quantum gate that combines their function.

Another potential optimization performed by step **1925** is the replacement of quantum gates by tensor products. For example, a conditional NOT, CNOT, quantum gate as commonly described in a C^2 complex (2×2) vector space basis may be replaced by a tensor product between two qubits in the C^2 complex (2×2) vector space basis generating a C^2 complex (4×4) vector space basis tensor product.

The output of step **1925** comprises an optimized intermediate representation that has been decorated according to the commands and operations available in the Instruction Set Architecture. This passes to step **1935** which process the decorated IR to invoke the selected commands and operations as API calls to a plurality of function code templates from step **1930**. The function code templates automate passing input arguments for specific operations in the Instruction Set Architecture. The sequence of API calls to selected code templates comprise code generators for each node or combination of nodes in the quantum network.

In the next step **1940**, the code generators obtained in step **1935** are executed as a program in a high-level language to generate one or more instruction streams comprising structured data commands and operations **1950**.

Method **1900** for automated compilation of commands and data as described above may be implemented in any programming language with the necessary capabilities and may use various libraries and tools that are available for data translation, graph analysis and data manipulation.

What is claimed is:

1. A method, performed by a computer system having one or more processors and memory storing instructions for execution on the one or more processors, for parallel searching of probe data in a database of target data, the method comprising:

storing, as first data at a location within the memory, a first probe data representation expressed in a first orthogonal domain, wherein the first probe data representation is characterized by a length;

storing, as second data in the memory, a first target data representation expressed in the first orthogonal domain, wherein the first target data includes a plurality of potential probe match data each characterized by the length;

transforming the first probe data representation and the first target data representation into a second orthogonal domain to produce a second probe data representation and a second target data representation, respectively, wherein transforming the first probe data representation and the first target data representation into the second orthogonal domain comprises applying a first orthogonal domain unitary transform to the first probe data representation and the first target data representation, respectively, wherein the second orthogonal domain is expressible using a basis set that is orthogonal to a basis set of the first orthogonal domain;

storing the second probe data representation in the memory;

encoding the second target data representation with a first plurality of modulation functions in the second orthogonal domain, each of the first plurality of modulation functions having an integer position index corresponding to one of the potential probe match data and a respective phase rotation, thereby producing a first plurality of encoded second target data representations encoded at different phase rotations;

superimposing the first plurality of encoded second target data representations to produce a superposition of the first plurality of encoded second target data representations;

storing the superposition of the first plurality of encoded second target data representations in the memory;

encoding the second target data representation with a second plurality of modulation functions in the second orthogonal domain, thereby producing a second plurality of encoded second target data representations, wherein each modulation function in the first plurality of modulation functions has a positive integer position index and corresponds to a modulation function in the second plurality of modulation functions that has a negative integer position index with the same magnitude as the positive integer position index;

superimposing the second plurality of encoded second target data representations to produce a superposition of the second plurality of encoded second target data representations;

storing the superposition of the second plurality of encoded second target data representations in memory;

creating a plurality of program instances, each of which retrieves from memory a different combination of the stored second probe representation and the pair comprising the stored superposition of the first plurality of encoded second target data representations and the stored superposition of the second plurality of encoded second target data representations;

interfering the superposition of the first plurality of encoded second target data representations with the second probe data representation to produce a first set of one or more interfered data representations;

interfering the superposition of the second plurality of encoded second target data representations with the second probe data representation to produce a second set of one or more corresponding interfered data representations;

combining each interfered data representation in the first set with a conjugate of the corresponding interfered data representation in the second set;

obtaining an inverse transform result characterizing a respective integer position index from a respective interfered data representation, wherein the inverse transform result is obtained from the combination of the interfered data representation in the first set and the corresponding conjugate interfered data representation in the second set;

determining whether the inverse transform result exceeds a predefined threshold; and

in accordance with a determination that the inverse transform result exceeds the predefined threshold, outputting, as the location in the memory where the first probe data is stored, the respective integer position index.

2. The method of claim **1**, wherein interfering the first plurality of encoded second target data representations with the second probe data representation comprises performing a vector multiply operation between the plurality of encoded second target data representations and a complex conjugate of the second probe data representation.

3. The method of claim **1**, wherein the first orthogonal domain unitary transform is a Fourier transform.

4. The method of claim **1**, wherein obtaining the inverse transform result characterizing a respective integer position index comprises:

applying a second orthogonal domain unitary transform to the one or more interfered data representations, wherein the second orthogonal domain unitary transform is an inverse of the first orthogonal domain unitary transform; and

selecting, as the inverse transform result, a result of the second orthogonal domain unitary transform applied to the one or more interfered data representations at a position corresponding to the respective integer position index.

5. The method of claim **1**, wherein the first probe data representation is a vector of real or complex numbers.

6. The method of claim **1**, wherein the first probe data representation comprises a plurality separately searchable component symbols encoded as sequential vectors of real or complex numbers.

7. The method of claim **1**, wherein the first data and the second data each comprise multi-dimensional data.

8. A computer system for parallel searching of probe data in a database of target data, comprising:

one or more processors; and

memory storing one or more programs for execution on the one or more processors, the one or more programs comprising instructions for:

storing, as first data at a location within the memory, a first probe data representation expressed in a first orthogonal domain, wherein the first probe data representation is characterized by a length;

storing, as second data in the memory, a first target data representation expressed in the first orthogonal domain, wherein the first target data includes a plurality of potential probe match data each characterized by the length;

transforming the first probe data representation and the first target data representation into a second orthogonal domain to produce a second probe data representation and a second target data representation, respectively, wherein transforming the first probe data representation and the first target data representation into the second orthogonal domain comprises applying a first orthogonal domain unitary transform to the first probe data representation and the first target data representation, respectively, wherein the second orthogonal domain is expressible using a basis set that is orthogonal to a basis set of the first orthogonal domain;

storing the second probe data representation in the memory;

encoding the second target data representation with a first plurality of modulation functions in the second orthogonal domain, each of the first plurality of modulation functions having an integer position index corresponding to one of the potential probe match data and a respective phase rotation, thereby producing a first plurality of encoded second target data representations encoded at different phase rotations;

superimposing the first plurality of encoded second target data representations to produce a superposition of the first plurality of encoded second target data representations;

storing the superposition of the first plurality of encoded second target data representations in the memory;

encoding the second target data representation with a second plurality of modulation functions in the second orthogonal domain, thereby producing a second plurality of encoded second target data representations, wherein each modulation function in the first plurality of modulation functions has a positive integer position index and corresponds to a modulation function in the second plurality of modulation functions that has a negative integer position index with the same magnitude as the positive integer position index;

superimposing the second plurality of encoded second target data representations to produce a superposition of the second plurality of encoded second target data representations;

storing the superposition of the second plurality of encoded second target data representations in memory;

creating a plurality of program instances, each of which retrieves from memory a different combination of the stored second probe representation and the pair comprising the stored superposition of the first plurality of encoded second target data representations and the stored superposition of the second plurality of encoded second target data representations;

interfering the superposition of the first plurality of encoded second target data representations with the second probe data representation to produce a first set of one or more interfered data representations;

interfering the superposition of the second plurality of encoded second target data representations with the second probe data representation to produce a second set of one or more corresponding interfered data representations;

combining each interfered data representation in the first set with a conjugate of the corresponding interfered data representation in the second set;

obtaining an inverse transform result characterizing a respective integer position index from a respective interfered data representation, wherein the inverse transform result is obtained from the combination of the interfered data representation in the first set and the corresponding conjugate interfered data representation in the second set;

determining whether the inverse transform result exceeds a predefined threshold; and

in accordance with a determination that the inverse transform result exceeds the predefined threshold, outputting, as the location in the memory where the first probe data is stored, the respective integer position index.

9. The computer system of claim 8, wherein interfering the first plurality of encoded second target data representations with the second probe data representation comprises performing a vector multiply operation between the plurality of encoded second target data representations and a complex conjugate of the second probe data representation.

10. A non-transitory computer readable storage medium, comprising:

one or more programs including instructions for execution by a computer system including one or more processors and memory, the one or more programs including instructions for:

storing, as first data at a location within the memory, a first probe data representation expressed in a first orthogonal domain, wherein the first probe data representation is characterized by a length;

storing, as second data in the memory, a first target data representation expressed in the first orthogonal domain, wherein the first target data includes a plurality of potential probe match data each characterized by the length;

transforming the first probe data representation and the first target data representation into a second orthogonal domain to produce a second probe data representation and a second target data representation, respectively, wherein transforming the first probe data representation and the first target data representation into the second orthogonal domain comprises applying a first orthogonal domain unitary transform to the first probe data representation and the first target data representation, respectively, wherein the second orthogonal domain is expressible using a basis set that is orthogonal to a basis set of the first orthogonal domain;

storing the second probe data representation in the memory;

encoding the second target data representation with a first plurality of modulation functions in the second orthogonal domain, each of the first plurality of modulation functions having an integer position index corresponding to one of the potential probe match data, thereby producing a first plurality of encoded second target data representations;

superimposing the first plurality of encoded second target data representations to produce a superposition of the first plurality of encoded second target data representations;

storing the superposition of the first plurality of encoded second target data representations in the memory;

encoding the second target data representation with a first plurality of modulation functions in the second orthogonal domain, each of the first plurality of modulation functions having an integer position index corresponding to one of the potential probe match data and a respective phase rotation, thereby producing a first plurality of encoded second target data representations encoded at different phase rotations;

superimposing the second plurality of encoded second target data representations to produce a superposition of the second plurality of encoded second target data representations;

storing the superposition of the second plurality of encoded second target data representations in memory;

creating a plurality of program instances, each of which retrieves from memory a different combination of the stored second probe representation and the pair comprising the stored superposition of the first plurality of encoded second target data representations and the stored superposition of the second plurality of encoded second target data representations;

interfering the superposition of the first plurality of encoded second target data representations with the second probe data representation to produce a first set of one or more interfered data representations;

interfering the superposition of the second plurality of encoded second target data representations with the second probe data representation to produce a second set of one or more corresponding interfered data representations;

combining each interfered data representation in the first set with a conjugate of the corresponding interfered data representation in the second set;

obtaining an inverse transform result characterizing a respective integer position index from a respective interfered data representation, wherein the inverse transform result is obtained from the combination of the interfered data representation in the first set and the corresponding conjugate interfered data representation in the second set;

determining whether the inverse transform result exceeds a predefined threshold; and

in accordance with a determination that the inverse transform result exceeds the predefined threshold, outputting, as the location in the memory where the first probe data is stored, the respective integer position index.

* * * * *