(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
09 March 2023 (09.03.2023)

WIPO | PCT

(10) International Publication Number
WO 2023/031785 A1

(51) International Patent Classification:
G06N 10/60 (2022.01)

(21) International Application Number:
PCT/IB2022/058111

(22) International Filing Date:
30 August 2022 (30.08.2022)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
63/260,829 01 September 2021 (01.09.2021) US

(71) Applicant: **ENTANGLED NETWORKS LTD.** [CA/CA]; 117 Helendale Av., Toronto, Ontario M4R 1C6 (CA).

(72) Inventors: **THAM, Edwin**; 117 Helendale Av., Toronto, Ontario M4R 1C6 (CA). **KHAIT, Ilia**; 117 Helendale Av., Toronto, Ontario M4R 1C6 (CA). **BRODUTCH, Aharon**; 117 Helendale Av., Toronto,, Ontario M4R 1C6 (CA).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CV, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IQ, IR, IS, IT, JM, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(54) **Title:** SYSTEMS AND METHODS FOR OPERATING AN INTERCONNECTED MULTI-CORE QUANTUM CLUSTER

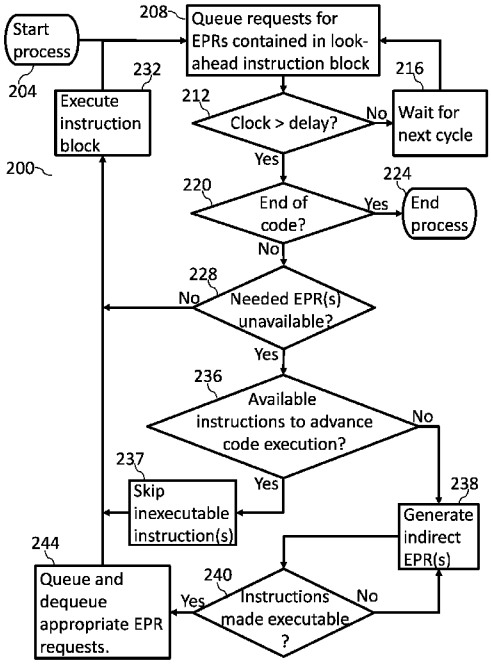

Fig. 2

(57) **Abstract:** A method for speeding up code execution in a multi-core quantum computer having entanglement-based interconnect, comprising: (a). Continuously looking ahead at a sliding look-ahead distance, and upon encountering a needed link, queueing a request for generation thereof. (b) When an instruction cannot be executed due to unavailable links, skipping this instruction and rescheduling it for execution upon detecting that the unavailable links have become available. (c). In the case that step (b) is insufficient to allow further execution advance, attempting to achieve further advance by attempting to replace, for each of at least part of the instructions currently scheduled for execution, its unavailable links by respective indirect links, wherein an indirect link is created by connecting available links in series. For each instruction thus becoming executable: dequeueing a pending generation request for each replaced link thereof and queueing generation requests for each link involved in replacing the unavailable links thereof.

WO 2023/031785 A1

[Continued on next page]

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

SYSTEMS AND METHODS FOR OPERATING AN INTERCONNECTED MULTI-CORE QUANTUM CLUSTER

FIELD OF THE INVENTION

The present invention relates generally to quantum computers, and more particularly to methods for efficiently performing quantum computing on multi-core quantum systems.

BACKGROUND OF THE INVENTION

Modular Quantum Computing architectures have been proposed in the academic and patent literature as a method for scaling quantum computers. For an overview of scientific literature, see for example Awschalom, David, et al. "Development of quantum interconnects (quics) for next-generation information technologies" PRX Quantum 2.1 (2021): 017002. Description of a network of ion-trap quantum computers connected by optical interconnects is provided, for example, by Monroe, C., et al. "Large-scale modular quantum-computer architecture with atomic memory and photonic interconnects." Physical Review A 89.2 (2014): 022317. US Patent No.9858531 discloses further details on that architecture. These modular quantum computers are composed of quantum processing units (QPUs) that need to exchange quantum information using entanglement. Entanglement between QPUs is generated using quantum interconnects and is destroyed as soon as it is used, see e.g. US Patent Application 20210105135. Thus, entanglement can be used as soon as it is created and can also be stored in memory for future use. This leads to the need for scheduling the entanglement generation and managing entanglement allocation.

In general, the generation scheduling and allocation of entanglement resources can happen ahead of the computation. However the generation of entanglement is a non-deterministic process. The time for producing entanglement cannot be determined ahead of time. As a result, scheduling generation and allocating entanglement ahead of time can lead to periods where the processor is idle, waiting for the availability of the allocated entanglement resource. As a result program execution times can be significantly longer.

Thus, it would be desirable to provide to the art methods for efficiently managing entanglement resources, in order to alleviate bottlenecks in large quantum computing systems, and thereby to reduce overall execution times.

SUMMARY OF THE INVENTION

In accordance with an embodiment of the present invention, a processor-implemented method is disclosed for minimizing execution time of quantum code (hereinafter "code") running on a multi-core quantum computer. The cores are linked by entanglement-based one-time-use links whose generation time duration varies randomly. The code comprises core local instructions and instructions that involve multiple cores thus needing those entanglement-based inter core links. The disclosed method comprises the following three steps:

(a). Continuously looking ahead at the code, at a predetermined sliding look-ahead time distance from the current execution time, and upon encountering a need for a link, queueing a request for generation thereof;

(b). when an instruction that is currently scheduled for execution cannot be executed due to unavailable one or more links, which is typically due to the random duration of their generation, skipping this inexecutable instruction, and rescheduling the skipped instruction for execution upon detecting that the unavailable one or more links have become available; and

(c). in the case that step (b) above is insufficient to allow further code execution advance, attempting to achieve further code execution advance by attempting to replace, for each of at least part of the instructions currently scheduled for execution, its unavailable one or more links by respective indirect links, wherein an indirect link is created by connecting two or more available links in series. For each instruction thus becoming executable: dequeueing a pending generation request for each replaced link thereof and queueing generation requests for each link involved in replacing the one or more unavailable links thereof. The incentive for dequeueing pending link generation requests is to save redundant workload and over-density in the link generation sub-system of the quantum computer. The incentive for dequeueing generation requests for the replacing links is to avoid unavailability of links that will be needed for future execution scheduled instructions.

In one embodiments, the following two step procedure is performed to select a preferred link combination in the case that more than one combination is available for replacing unavailable one or more links of a currently scheduled for execution instruction:

(a). Assigning to each combination a need-factor calculated as a weighted sum of all the links contained in the combination while employing a predetermined weighting function that gives lower weights to links that will be needed for executing instructions at later times; and

(b). selecting a combination having the minimum need-factor, or the combination having less links in the case of several combinations having the same minimum need-factor. This

criterion of minimum need-factor is used to ensure minimum potential impact on future scheduled instructions.

The above weighting function typically covers the time interval from the current execution time up to the above look-ahead time distance thereof, such that the function's value at the end of the interval is in the range of 0 to 25% of its value at the current execution time. Example weighting functions are exponential, linear and raised cosine.

In one embodiment, the following three step procedure is performed when attempting to replace the unavailable links of several currently scheduled for execution instructions:

(a). Calculating, according to the above two step procedure, the minimum need-factor related to each of the several instructions as if this instruction is the only one that competes for the available links;

(b). selecting the instruction having the smallest minimum need-factor; and

(c). restarting the procedure with the remaining currently scheduled for execution instructions.

In one embodiment, when several currently scheduled for execution instructions contain unavailable links, those instructions are ordered according to the original times of their scheduling for execution, before attempting to replace their unavailable links one by one. This way those instructions that have been already delayed due to lack of available links get priority in replacement, which is related to their delay.

In some system embodiments, code execution advance is conditioning on having a minimum number of executable instructions. This condition actually necessitates the attempt to look for indirect link replacement for unavailable links.

In typical embodiments of the disclosed method, starting the code execution is delayed relative to the looking ahead start time in step (a) above. The incentive for this delay is to start generating the links sufficiently before they are actually needed. The execution delay is typically equal or somewhat greater than the look-ahead time distance.

In typical system embodiments wherein the disclosed method is implemented, at least part of the links are based on Einstein–Podolsky–Rosen (EPR) pairs. In these embodiments, connecting two or more available links in series is based on fusing them. The link generation process in these embodiments approximately behaves according to a Poisson random process. The look-ahead time distance is typically predetermined in the range of $[1/\lambda, 10/\lambda]$, wherein λ is the average rate of link generation when the generation requests queue is not empty.

Further disclosed is a processing device implementing the above disclosed method.

Yet further disclosed is multi-core quantum computer that is controlled according to the above disclosed method.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will be more fully understood from the following detailed description of the embodiments thereof, taken together with the drawings in which:

Fig. 1 is a block diagram that schematically illustrates a multi-core quantum computing system, in accordance with an embodiment of the present invention; and

Fig. 2 is a flowchart that schematically illustrates a method for minimizing execution time of quantum code running on a multi-core quantum computing system, in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION OF EMBODIMENTS

Embodiments of the present invention provide quantum computing control methods and systems wherein quantum instructions (hereinafter "instructions") are optimally executed so as to improve the computing efficiency of multi-core quantum computers. A quantum computer core is hereinafter also denoted as quantum processing unit (QPU). The achieved efficiency contributes to faster and more reliable quantum computing. In the following description it is assumed that the underpinning quantum technology is trapped ion. However, the techniques and concepts suggested herein are agnostic to the underpinning quantum technology as well as to the QPUs interconnect technology.

Referring to Fig. 1, there is shown a block diagram of a multi-core quantum computing system 100, in accordance with an embodiment of the present invention. System 100 comprises two principal sub-systems: quantum computer 108 and quantum computer controller 112. These two sub-systems communicate therebetween through EM, readout and addressing input output (I/O) buses 116 and 120 respectively. The dashed line above I/O bus 120 indicates the border between the lower physical (phy) layer and the upper software (SW) layer of system 100. System 100 communicates with the outer world through I/O port 122. The block diagram shown in Fig. 1 focuses on the specific elements of system 100 that are essential for understanding certain features of the disclosed techniques. Conventional elements and connectivity that are not needed for this understanding have been omitted from Fig. 1 for the sake of simplicity, but will be apparent to persons of ordinary skill in the art. For example, no hardware and software components in controller 112 are depicted besides I/O ports and some software functions,

demarcated by block 124, that are particularly relevant to the present invention. Following is a functional description of the aforementioned blocks, wherein all the software blocks within block 124 are denoted for brevity as "blocks".

Controller 112 accepts, through software I/O port 122, quantum code (hereinafter "code") in the form of compiled quantum circuit instructions. The code is typically arranged as instruction blocks for later processing in the rate of one instruction block per clock cycle of quantum computer 108. Each instruction block may contain a single or a plurality of instructions. The instruction blocks are looked ahead in block 128 and processed in block 132, as described in detail with reference to Fig. 2 hereinafter. Block 132 parses the code and schedules the instruction blocks for actual execution. Block 132 transfers the scheduling information to block 128 for code lookahead. Block 132 also gets decisions to queue and dequeue requests for generation of Einstein-Podolsky-Rosen pairs (EPRs) as described below. These EPRs constitute inter-core links (hereinafter "links"), which are needed for executing instructions involving multiple cores. In some embodiments, other means than EPRs may be employed for generating links such as concatenated Greenberger–Horne–Zeilinger states. Block 136 takes care of managing those requests and the resulting EPR reservoir. Block 132 transfers the scheduled quantum instructions to block 140 which adapts the quantum circuit code format to the waveform format that is readable by I/O bus 120. This waveform format includes electrical pulses for inducing electromagnetic (EM) pulses in I/O bus 120, which serve for stimulating quantum activity in quantum computer 108. In various embodiments, the type of the EM pulses can be optical, RF, and/or microwave, however other quantum-suitable EM types may be employed as well. A second source of block 140 is block 136, which transfers thereto requests for EPRs generation as well as requests to cancel generation of previously requested EPRs due to dequeueing requests, which are described hereinafter. Software interface 140 also associates addressing data with the electrical pulses for later directing the EM pulses to the appropriate qubits in quantum computer 108.

The description of controller 112 has so far related to its operation in the downstream (software to physical) direction. In the inverse direction, quantum computer 108 transfers electrical readout pulses, associated with their corresponding source addresses, through I/O basses 116 and 120, to quantum readout

software interface 144. Interface 144 transforms this upstream information to software readable format to allow interpretation of the quantum computing results in block 148. Block 148 then transfers the interpreted results to the outer world through I/O port 122 as well as to block 132 for affecting the code execution control. Block 144 also transfers readout results relating to

successfully generated EPRs to block 136 for EPR reservoir management and for reporting block 132 about newly available EPRs. In some embodiments, controller 112 also outputs, through I/O port 122, logged data related to the quantum processed thereby for later analysis so as to improve future compilations efficiency.

Referring now to quantum computer 108, it receives through I/O bus 116 EM pulses addressed to the various cores included in a QPUs array 152. Consequently, the QPUs perform core-local logic operations according to the quantum instructions, as well as inter-core logic operations through interconnect 156. Measurement sub-system 160 reads out of array 152 the results of the logic operations performed thereby, and transfers the readout results, together with the corresponding source addresses, to controller 112 through I/O bus 116. In the same way, EPR generation sub-system 164 reads out, through interconnect 156, successful EPR generation results, as well as successful generation of indirect EPRs, and transfers the readout results, together with their corresponding source addresses, to controller 112 through I/O bus 116. In the above, an indirect EPR is an inter-core linkage achieved by connecting multiple EPRs in series using fusion technique. In some embodiments, other types of link connection techniques may be used.

The block diagram shown in Fig. 1 illustrates an example hardware and software architecture, which was chosen purely for the sake of conceptual clarity. In alternative embodiments, any other suitable architecture can also be used. In one embodiment controller 112 comprises a general purpose processor which runs software for carrying out the functions described above. However, any other suitable control means can be used alternatively or additively such as ASICs and FPGAs.

Referring to Fig. 2, there is shown a flowchart 200 which schematically illustrates a process corresponding to a method for minimizing execution time of quantum code running on a multi-core quantum computer system, such as system 100 above. The cores are connected by entanglement-based one-time-use links, whose generation time duration varies randomly. In typical embodiments, the links are implemented as EPRs and their generation time is exponentially distributed with mean generation time of $1/\lambda$. The links are required for executing inter-core instructions, i.e. instructions involving qubits in different cores.

Flowchart 200 begins with a starting step 204 that starts a clock determining the rate in which instruction blocks are executed in quantum computer 108. In a looking ahead step 208, block 128 looks ahead at a predetermined sliding look-ahead time distance from the current clock cycle, and upon encountering a need for an EPR in an instruction, block 128 transfers a

request for its generation to block 136 through block 132. In various embodiments, the look-ahead time distance may be predetermined in the range of [1/λ, 10/λ] with a typical value of 5/λ. Block 136 manages the EPR generation requests in queues. Once it is informed by block 144 that a requested EPR has been generated in EPR generation sub-system 164, block 136 transfers this EPR from a request queue thereof to an EPR reservoir that it manages. All further logical steps of process 200, besides step 244, are carried out in block 132. Decision step 212 and waiting step 216 that follow actually constitute a clock cycle counter that delays the code execution start cycle by a predetermined delay, during which future needed EPRs can be generated in advance according to step 208. In typical embodiments, this delay is equal to the above look-ahead time distance, however, in some embodiments it may be greater than the look-ahead time distance.

In a decision step 220 block 132 check if the last code instruction is contained in the current instruction block. If the check result is positive, the process reaches its end in step 224; otherwise, the process proceeds to step 228, in which block 132 checks, by querying block 136, whether one or more needed EPRs in the currently scheduled for execution instruction block are unavailable. Obviously, this may happen only in the case of having an inter-core instruction within the current instruction block. If no EPR is unavailable, the current instruction block is executed by quantum computer 108 in step 232. The process now returns to step 208 when a new clock cycle starts and the process continues to loop as above. If one or more needed EPRs are unavailable in one or more instructions in the current instruction block, the process proceeds from step 228 to a decision step 236. According to step 236, if the current instruction block still contains a predetermined minimum number of executable instructions, in the sense of not lacking any needed EPR (which is obvious for core-local instructions), the process proceeds to step 232 through a skipping step 237 and may possibly continue to loop through step 228 and possibly also step 236 as described so far. Each transition through step 232 actually corresponds to an advance in the code execution. In some embodiments the above minimum number of executable instructions may be as small as a single instruction. In some embodiments transition from either step 228 or step 236 to step 232 may comprise advanced execution of previously later scheduled instructions, which is driven by block 132 as part of its scheduling function. Block 132 constantly tracks, through block 136, unavailable EPRs pertaining to inexecutable instructions that were skipped as shown in step 237. Upon detecting that all unavailable EPRs of a previously skipped instruction have become available, block 132 reschedule this instruction for execution.

When the code execution cannot advance from step 236 to step 232 due to unavailability of links required for advance in the code execution as described above, process 200 proceeds to a

generation step 238, in which block 132 attempts to turn inexecutable instructions to be executable so as to schedule them for execution. This is done by attempting to replace the unavailable EPRs with indirect EPRs, which can be generated as explained above with reference sub-system 164. Block 132 is aware of available EPRs that are candidates for this sake by querying the EPR reservoir through block 136. In a decision step 240 that follows step 238, block 132 schedules for execution instructions that have become executable following step 238. For those instructions, the process proceeds to execution step 232 through a queueing and dequeueing step 244. In step 244 block 132 takes care of dequeueing a pending, i.e. already queued, generation request for each replaced link, so as to save redundant workload and over-density from sub-system 164. Block 132 also takes care, in step 244, of queueing a generation request for each link involved in replacing unavailable links in step 238, so as to avoid unavailability of links that will be needed for future scheduled instructions. For the instructions that failed to become executable in step 238 process 200 returns to step 238 for repeated attempts to make them executable by means of indirect links while concurrently waiting for the appropriate direct unavailable EPRs to become available.

There are various options related to replacing unavailable EPRs that may be employed by block 132 in embodiments of the present invention. In the case that more than one EPR combination is available for replacing the unavailable one or more EPRs of an instruction, selecting a preferred combination thereof is performed in one embodiment by the following two steps example procedure:

(a) Assigning to each combination a need-factor calculated as a weighted sum of all the EPRs contained in the combination while employing a predetermined weighting function that gives lower weights to EPRs that will be needed for executing instructions at later times; and

(b) selecting a combination having a minimum need-factor.

In the case of a plurality of combinations have the same minimum need-factor, selecting the one thereof having less EPRs.

In typical embodiments, the weighting function covers the time interval from the current execution time up to the look-ahead time distance thereof, such that its value at the end of the interval is in the range of 0 to 25% of its value at the current execution time. Example weighting functions are exponential, linear and raised cosine.

Block 132 may also attempt, in step 238, to replace the unavailable EPRs of multiple instructions simultaneously, typically those currently scheduled for execution. This can be done by the following example three step procedure:

(a). Calculating, according to the above two step procedure, the minimum need-factor related to each instruction as if this instruction is the only one that competes for the available links;

(b). selecting the instruction having the smallest minimum need-factor; and

(c). restarting the procedure with the remaining instructions.

In one embodiment, replacing the unavailable EPRs of multiple instructions is performed in the order respective to their original scheduled execution times.

Flowchart 200 is an example flowchart, which was chosen purely for the sake of conceptual clarity. In alternative embodiments, any other suitable flowchart can also be used for illustrating the disclosed method. Method steps that are not mandatory for understanding the disclosed techniques were omitted from Fig. 2 for the sake of simplicity.

It will thus be appreciated that the embodiments described above are cited by way of example, and that the present invention is not limited to what has been particularly shown and described hereinabove. Rather, the scope of the present invention includes both combinations and sub-combinations of the various features described hereinabove, as well as variations and modifications thereof which would occur to persons skilled in the art upon reading the foregoing description and which are not disclosed in the prior art.

CLAIMS

1. A processor-implemented method for minimizing execution time of quantum code (hereinafter "code") running on a multi-core quantum computer, wherein the cores are linked by entanglement-based one-time-use links whose generation time duration varies randomly, the code comprising core-local instructions and instructions that involve multiple cores thus needing inter-core links, the method comprising the following steps:

(1a). continuously looking ahead at the code, at a predetermined sliding look-ahead time distance from the current execution time, and upon encountering a need for a link, queueing a request for generation thereof;

(1b). when an instruction that is currently scheduled for execution cannot be executed due to unavailable one or more links, skipping this inexecutable instruction, and rescheduling the skipped instruction for execution upon detecting that the unavailable one or more links have become available; and

(1c). in the case that step (1b) above is insufficient to allow further code execution advance, attempting to achieve further code execution advance by attempting to replace, for each of at least part of the instructions currently scheduled for execution, its unavailable one or more links by respective indirect links, wherein an indirect link is created by connecting two or more available links in series, and for each instruction thus becoming executable: dequeueing a pending generation request for each replaced link thereof and queueing generation requests for each link involved in replacing the one or more unavailable links thereof.

2. The method of claim 1, wherein in the case that more than one link combination (hereinafter "combination") is available for replacing the unavailable one or more links of a currently scheduled for execution instruction, selecting a preferred combination thereof by the following steps:

(2a). assigning to each combination a need-factor calculated as a weighted sum of all the links contained in the combination while employing a predetermined weighting function that gives lower weights to links that will be needed for executing instructions at later times, and

(2b). selecting a combination having a minimum need-factor.

3. The method of claim 2, wherein in the case of a plurality of combinations having the same minimum need-factor, selecting the one thereof having less links.

4 . The method of claim 2, wherein the predetermined weighting function covers the time interval from the current execution time up to the look-ahead time distance thereof, such that its value at the end of the time interval is in the range of 0 to 25% of its value at the current execution time.

5 . The method of claim 2, wherein the predetermined weighting function is selected from the group of functions consisting of exponential, linear and raised cosine.

6 . The method of claim 2, while attempting to replace the unavailable links of a plurality of currently scheduled for execution instructions by the following procedure steps:

(6a). calculating the minimum need-factor related to each of said plurality of currently scheduled for execution instructions;

(6b). selecting the instruction having the smallest minimum need-factor; and

(6c). restarting the procedure with the remaining currently scheduled for execution instructions.

7 . The method of claim 1, while attempting to replace the unavailable links of a plurality of currently scheduled for execution instructions in the order respective to their original scheduled execution times.

8 . The method of claim 1, while conditioning code execution advance on having a minimum number of executable instructions.

9 . The method of claim 1 performed while starting the code execution after starting step 1a with a predetermined delay which is greater than or equal to the look-ahead time distance.

10 . The method of claim 1, wherein at least part of the links are based on Einstein–Podolsky–Rosen (EPR) pairs.

11 . The method of claim 1, wherein connecting in series two or more available links is based on fusing thereof.

12 . The method of claim 1, wherein the look-ahead time distance is predetermined in the range of $[1/\lambda, 10/\lambda]$, wherein λ is the average rate of link generation when the generation requests queue is not empty.

13 . A processing device implementing the method of claim 1.

14 . A multi-core quantum computer, which is controlled according to the method of claim 1.

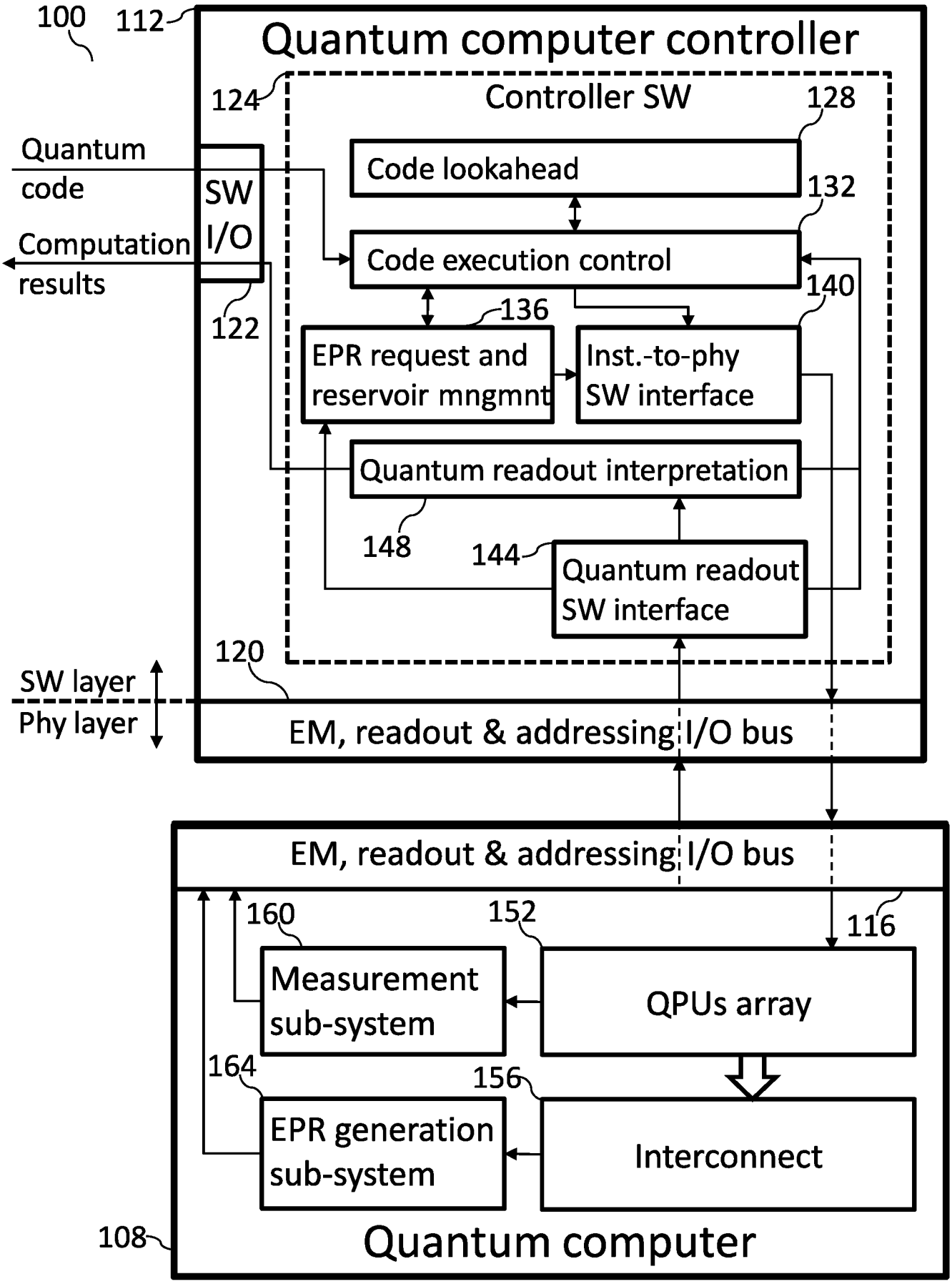
100
112
Quantum computer controller
124
Controller SW
128
Quantum code
SW I/O
Code lookahead
132
Computation results
Code execution control
140
122
136
EPR request and reservoir mngmnt
Inst.-to-phy SW interface
Quantum readout interpretation
148
144
Quantum readout SW interface
120
SW layer
Phy layer
EM, readout & addressing I/O bus
EM, readout & addressing I/O bus
160
152
116
Measurement sub-system
QPUs array
164
156
EPR generation sub-system
Interconnect
108
Quantum computer

Fig. 1

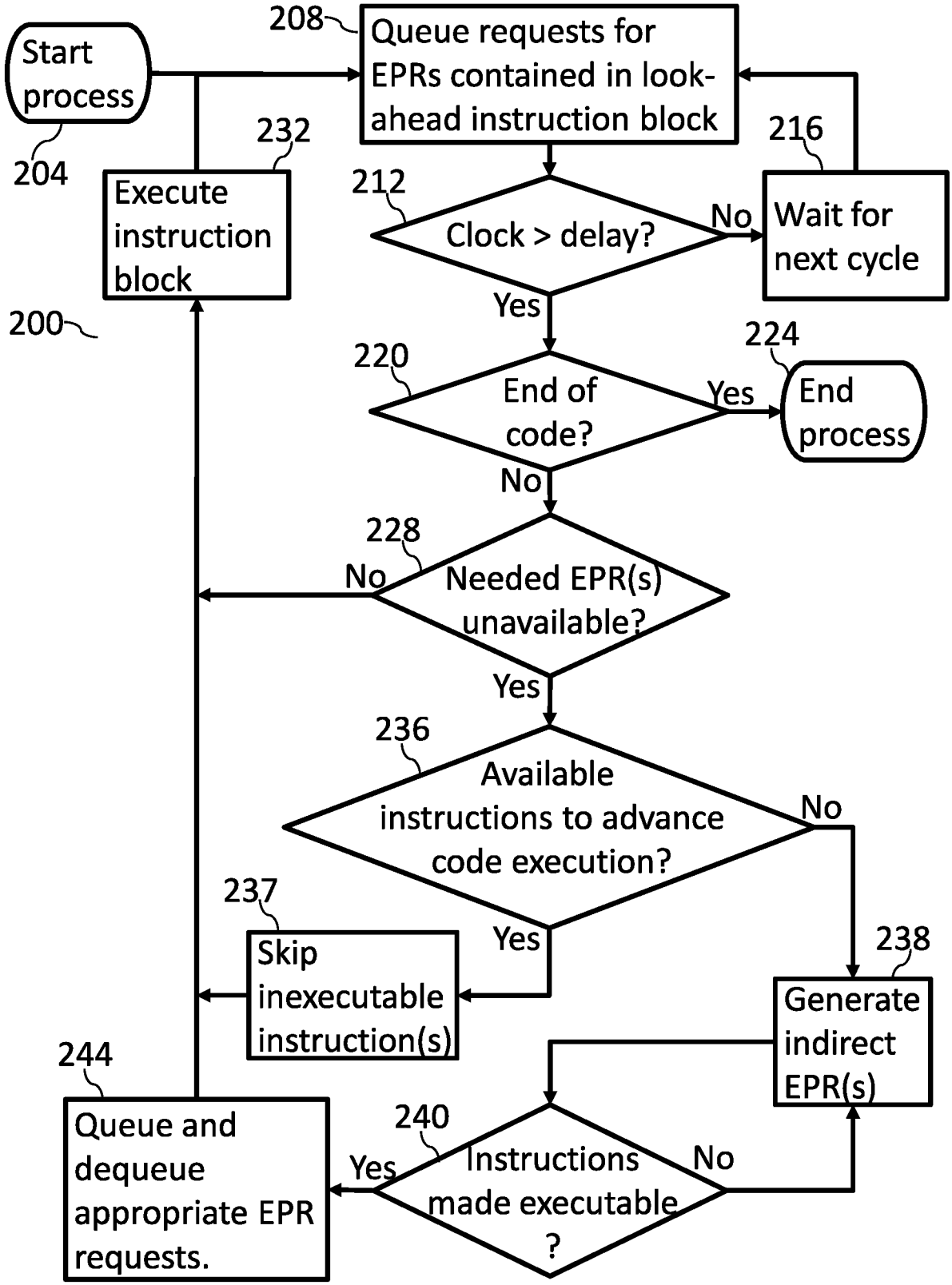
Start process
204
208
Queue requests for EPRs contained in look-ahead instruction block
216
Wait for next cycle
212
Clock > delay?
No
Yes
232
Execute instruction block
200
220
End of code?
Yes
224
End process
No
228
Needed EPR(s) unavailable?
No
Yes
236
Available instructions to advance code execution?
No
Yes
237
Skip inexecutable instruction(s)
238
Generate indirect EPR(s)
244
Queue and dequeue appropriate EPR requests.
240
Instructions made executable ?
Yes
No

Fig. 2

INTERNATIONAL SEARCH REPORT

International application No.
PCT/IB2022/058111

A. CLASSIFICATION OF SUBJECT MATTER

IPC: ***G06N 10/60*** (2022.01)

CPC: **G06N 10/60** (2022.01)

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
Searched across all classifications.

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
None

Electronic database(s) consulted during the international search (name of database(s) and, where practicable, search terms used)
Orbit.com and Google/scholar: quantum computer, code, execution, link, multi-core, one-time-use links, inter-core links, request, generation of links, look ahead, skip execution, indirect links, replaced link, interconnect.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	Rodrigo, Santiago, et al. "Will Quantum Computers Scale Without Inter-Chip Comms? A Structured Design Exploration to the Monolithic vs Distributed Architectures Quest." 2020 XXXV Conference on Design of Circuits and Integrated Systems (DCIS). IEEE, 2020. *pages 2-3*	1
A	Chen et al., "Contention Minimization in Emerging SMART NoC via Direct and Indirect Routes", IEEE TRANSACTIONS ON COMPUTERS, VOL. 71, NO. 8, AUGUST 2022, 1874-1888 *abstract*	1

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"D"	document cited by the applicant in the international application	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"E"	earlier application or patent but published on or after the international filing date	"&"	document member of the same patent family
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)		
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search 12 October 2022 (12-10-2022)	Date of mailing of the international search report 18 November 2022 (18-11-2022)
Name and mailing address of the ISA/CA Canadian Intellectual Property Office Place du Portage I, C114 - 1st Floor, Box PCT 50 Victoria Street Gatineau, Quebec K1A 0C9 Facsimile No.: 819-953-2476	Authorized officer Wei Wang (819) 639-8340