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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
16 July 2020 (16.07.2020)

WIPO | PCT

(10) International Publication Number
WO 2020/142971 A1

(51) International Patent Classification:
G06F 12/00 (2006.01)

(21) International Application Number:
PCT/CN2019/071086

(22) International Filing Date:
10 January 2019 (10.01.2019)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **ALIBABA GROUP HOLDING LIMITED** [—/CN]; Fourth Floor, One Capital Place, P.O. Box 847, George Town, Grand Cayman (KY).

(72) Inventors: **GUO, Zhi**; 525 Almanor Ave, 4th Floor, Sunnyvale, California 94085 (US). **SUN, Hua**; 525 Almanor Ave, 4th Floor, Sunnyvale, California 94085 (US). **LI, Longxiao**; 525 Almanor Ave, 4th Floor, Sunnyvale, California 94085 (US). **YAN, Xiaohui**; No. 3331 Keyuan South Road, Nanshan District, Shenzhen, Guangdong 518054 (CN). **WEI, Dongdong**; No. 3331 Keyuan South Road, Nanshan District, Shenzhen, Guangdong 518054 (CN). **YU, Xulin**; No. 3331 Keyuan South Road, Nanshan District, Shenzhen, Guangdong 518054 (CN).

(74) Agent: **BEIJING TSINGYUANHUI INTELLECTUAL PROPERTY LAW FIRM**; Room 362, 3rd Floor, Suzhou Street No.1, Haidian District, Beijing 100080 (CN).

(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, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME,

(54) Title: SYSTEMS AND METHODS FOR PROVIDING DATABASE ACCELERATION USING A PROGRAMMABLE LOGIC DEVICE (PLD)

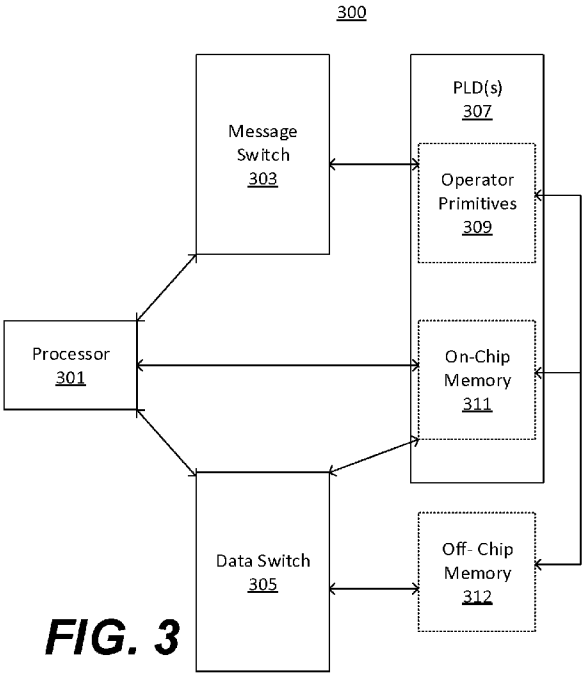

FIG. 3

(57) Abstract: The present disclosure relates to computer-implemented systems and methods for accelerating database operations using programmable logic devices (PLDs). In one implementation, a method for accelerating a database may include transferring a first set of hardware configuration instructions to cause a first portion of at least one programmable logic device (PLD) to execute retrievals of database elements from one or more on-chip memories; transferring a second set of hardware configuration instructions to cause a second portion of the at least one PLD to execute one or more database operations; transferring a third set of hardware configuration instructions to cause a third portion of the at least one PLD to direct an incoming database query to the first portion or the second portion of the at least one PLD based on at least a portion of contents of the query;; transferring a database to the at least one PLD; receiving a query for execution against the database and sending the query to the at least one PLD; and in response to the query, receiving results

WO 2020/142971 A1

[Continued on next page]

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, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(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))*

from the at least one PLD.

SYSTEMS AND METHODS FOR PROVIDING DATABASE ACCELERATION USING A PROGRAMMABLE LOGIC DEVICE (PLD)

TECHNICAL FIELD

[0001] The present disclosure relates generally to the field of database operations and programmable logic devices. More specifically, and without limitation, this disclosure relates to computer-implemented systems and methods for accelerating a database using programmable logic devices. The systems and methods disclosed herein may be used in various applications, such as relational databases (e.g., a structured query language (SQL) database or the like), graphical databases (e.g., an ArangoDB query language (AQL) database, another NoSQL database, or the like) or any other database structures.

BACKGROUND

[0002] Field-programmable gate arrays (FPGAs) and other programmable logic device (PLDs) are generally more efficient for database operations than conventional processing hardware, such as central processing units (CPUs), graphics processing units (GPUs), or the like. However, the use of FPGAs and other PLDs to accelerate a database conventionally includes only particular database operations, limiting the flexibility of the accelerated system. For example, many accelerated systems are only able to process part of complex queries, if at all.

[0003] Moreover, the use of FPGAs and other PLDs to accelerate a database conventionally is designed specifically for a row-oriented database or a column-oriented database. Accordingly, such accelerated systems are unable to be used across different database orientations.

SUMMARY

[0004] In some embodiments, a system for accelerating a database using at least one programmable logic device (PLD) may comprise at least one memory configured to store instructions and at least one processor configured to execute the instructions to cause the system to perform operations. The operations may comprise transferring a first set of hardware configuration instructions to cause a first portion of the at least one PLD to execute retrievals of database elements from one or more on-chip memories; transferring a second set of hardware configuration instructions to cause a second portion of the at least one PLD to execute one or more database operations; transferring a third set of hardware configuration instructions to cause a third portion of the at least one PLD to direct an incoming database query to the first portion or the second portion of the at least one PLD based on at least a portion of contents of the query;; transferring a database to the at least one PLD; receiving a query for execution against the database and sending the query to the at least one PLD for execution against the transferred database; and in response to the query, receiving results from the at least one PLD.

[0005] In some embodiments, a method for accelerating a database using at least one programmable logic device (PLD) may comprise transferring a first set of hardware configuration instructions to cause a first portion of the at least one PLD to execute retrievals of database elements from one or more on-chip memories; transferring a second set of hardware configuration instructions to cause a second portion of the at least one PLD to execute one or more database operations; transferring a third set of hardware configuration instructions to cause a third portion of the at least one PLD to direct an incoming database query to the first portion or the second portion of the at least one PLD based on at least a portion of contents of the query;; transferring a database to the at least one PLD; receiving a query for execution against the database and sending the query to the at least one PLD for

execution against the transferred database; and in response to the query, receiving results from the at least one PLD.

[0006] In some embodiments, a non-transitory computer-readable storage medium may store a set of instructions that is executable by one or more processors to cause the one or more processors to perform a method for accelerating a database using at least one programmable logic device (PLD). The method may comprise transferring a first set of hardware configuration instructions to cause a first portion of the at least one PLD to execute retrievals of database elements from one or more on-chip memories; transferring a second set of hardware configuration instructions to cause a second portion of the at least one PLD to execute one or more database operations; transferring a third set of hardware configuration instructions to cause a third portion of the at least one PLD to direct an incoming database query to the first portion or the second portion of the at least one PLD based on at least a portion of contents of the query;; transferring a database to the at least one PLD; receiving a query for execution against the database and sending the query to the at least one PLD for execution against the transferred database; and in response to the query, receiving results from the at least one PLD.

[0007] Additional objects and advantages of the present disclosure will be set forth in part in the following detailed description, and in part will be obvious from the description, or may be learned by practice of the present disclosure. The objects and advantages of the present disclosure will be realized and attained by means of the elements and combinations particularly pointed out in the appended claims.

[0008] It is to be understood that the foregoing general description and the following detailed description are exemplary and explanatory only, and are not restrictive of the disclosed embodiments.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] The accompanying drawings, which comprise a part of this specification, illustrate several embodiments and, together with the description, serve to explain the principles and features of the disclosed embodiments. In the drawings:

[0010] **FIG. 1** is a schematic representation of primitives in a field-programmable gate array (FPGA), according to embodiments of the present disclosure.

[0011] **FIG. 2** is an exemplary architecture for configuring programmable logic devices (PLDs) to accelerate transferred databases, according to embodiments of the present disclosure.

[0012] **FIG. 3** is a schematic representation of a configuration for database acceleration in a PLD, according to embodiments of the present disclosure.

[0013] **FIG. 4** is a schematic representation of a configuration for a memory array in a PLD, according to embodiments of the present disclosure.

[0014] **FIG. 5** is a schematic representation of a configuration for an operator array in a PLD, according to embodiments of the present disclosure.

[0015] **FIG. 6** is a schematic representation of a configuration for message or data switching in a PLD, according to embodiments of the present disclosure.

[0016] **FIG. 7A** is a graphical representation of a column-oriented database, according to embodiments of the present disclosure.

[0017] **FIG. 7B** is a graphical representation of a row-oriented database, according to embodiments of the present disclosure.

[0018] **FIG. 8** is a schematic representation of configuring a PLD for database acceleration according to embodiments of the present disclosure.

[0019] **FIG. 9** is a flowchart of an exemplary method for accelerating a database using a programmable logic device (PLD), according to embodiments of the present disclosure.

[0020] **FIG. 10** is a depiction of an exemplary computer system for executing methods consistent with the present disclosure.

DETAILED DESCRIPTION

[0021] Reference will now be made in detail to exemplary embodiments, examples of which are illustrated in the accompanying drawings. The following description refers to the accompanying drawings in which the same numbers in different drawings represent the same or similar elements unless otherwise represented. The implementations set forth in the following description of exemplary embodiments do not represent all implementations consistent with the invention. Instead, they are merely examples of apparatuses and methods consistent with aspects related to the invention as recited in the appended claims.

[0022] The disclosed embodiments relate to computer-implemented systems and methods for accelerating a database using a programmable logic device (PLD). For example, the system may be configured for a plurality of operations, such as regular expression matching, integer comparisons, string comparisons, compaction commands, gzip commands, or the like, as well as database retrievals with minor database operations, such as nulling database elements, bit shifting database elements, incrementing database elements, decrementing database elements, or the like. Advantageously, the exemplary embodiments can provide improved flexibility and efficiency over conventional database acceleration systems. Embodiments of the present disclosure can also provide database acceleration systems that are reconfigurable to accept column-oriented databases as well as row-oriented databases.

[0023] Embodiments of the present disclosure may be implemented and used in various programmable logic devices (PLDs). Accordingly, although described in reference to field-programmable gate arrays (FPGAs), other PLDs such as programmable array logics (PALs), programmable logic arrays (PLAs), complex programmable logic devices (CPLDs), and the like may perform database acceleration in accordance with the present disclosure.

[0024] The embodiments of the present disclosure provide computer-implemented systems and methods for providing database acceleration using PLDs. The systems and methods of the present disclosure may provide a technical solution to the technical problem of configuring PLDs for database acceleration with flexibility to process retrievals separately from queries in order to accelerate the former. Moreover, the systems and methods of the present disclosure may provide a technical solution to the technical problem of configuring PLDs for database acceleration with flexibility to process multiple queries in parallel. Because the systems and methods of the present disclosure provide database acceleration, they may also result in efficiency gains as compared with database retrievals or operations executed on general-purpose processors.

[0025] **FIG. 1** is a schematic representation of exemplary portions 100, 150 of an architecture of an FPGA (or other PLD). As depicted in **FIG. 1**, a primitive 105a may connect to a plurality of data buffers, such as off-chip buffers 103a and 103b or on-chip buffers 101a and 101b. As used herein, a primitive refers to a node of the FPGA that performs a basic operation (whether logical, such as AND, OR, XOR, or the like, or arithmetic, such as multiply, add, subtract, max, min, or the like) on one or more inputs to produce one or more outputs. For example, in **FIG. 1**, primitive 105a may accept input from off-chip buffer 103a or on-chip buffer 101a and may output to off-chip buffer 103b or on-chip buffer 101b. As used herein, a buffer refers to any bus used to communicate data, such as a wire, an optical cable, or the like, along with any memory coupled to the bus and

used to store (and thus "buffer") the data or any arbiters or other timing hardware used to manage transfers on the bus.

[0026] Similar to primitive 105a, primitive 105b may accept input from off-chip buffer 103c or on-chip buffer 101b and may output to off-chip buffer 103d or on-chip buffer 101c. Accordingly, in the example of **FIG. 1**, primitive 105a may provide its output as input to primitive 105b using on-chip buffer 101b. Thus, primitive 105a and primitive 105b may be grouped as a subgraph of operations that flow from the operation(s) performed by primitive 105a to the operation(s) performed by primitive 105b. Embodiments of the present disclose may configure primitives (such as primitive 105a and primitive 105b) of an FPGA (or other PLDs) to accelerate database operations. On-chip and off-chip memories (not shown in the example of **FIG. 1**) may store elements of the database that were previously mapped and transferred thereto.

[0027] **FIG. 2** is a schematic representation of a system 200 for transferring databases to PLDs and for configuring PLDs to acceleration operations on transferred databases, consistent with embodiments of the present disclosure. As depicted in **FIG. 2**, a non-transitory storage medium 201 (such as a random access memory (RAM) or a read-only memory (ROM)) may store a database. The database may comprise a relational database, a graphical database, or any other data structure having plurality of elements searchable via at least one index.

[0028] A compiler 207 may comprise one or more instructions executed by at least one processor. For example, compiler 207 may comprise a series of instructions executed by a general-purpose processor (such as a central processing unit (CPU), graphical processing unit (GPU), or the like) or a special-purpose processor (such as an FPGA or other application-specific integrated circuit (ASIC)). As depicted in **FIG. 2**, compiler 207 may configure programmable logic device (PLD) 209 to execute one or more database operations

(e.g., operation 205). Additionally or alternatively, although not depicted in **FIG. 2**, compiler 207 may generate a mapping between database 201 and a programmable logic device (PLD) 209, e.g., by determining a size and spatial location of on- and off-chip memories of PLD 209 and mapping elements of database 201 to blocks of the on- and off-chip memories. In embodiments where compiler 207 also configures PLD 209 for accelerating database 201, as described below, compiler 207 may generate the mapping such that database elements are stored in on-chip memories adjacent to primitives of PLD 209 configured as an intelligent memory array, as explained below with respect to **FIG. 4**. Moreover, as further explained below with respect to **FIGS. 7A and 7B**, compiler 207 may map database 201 to PLD 209 such that the mapping is agnostic as to whether database 201 is row-oriented or column-oriented.

[0029] As further depicted in **FIG. 2**, compiler 207 may generate one or more sets of hardware configuration instructions, e.g., as described below in method 900 of **FIG. 9**. For example, the instructions may configure one or more primitives of PLD 209 to execute database retrievals (e.g., operation 205) on database 201 transferred to PLD 209 as well as to execute database operations on database 201 transferred to PLD 209. In some embodiments, the instructions may comprise one or more data files in a specification language, such as Verilog, impulse C, or any other hardware description language (HDL). Additionally, compiler 207 may configure one or more switches, e.g., switch 203, to arbitrate between queries to database 201 and manage transfer of data to and from PLD 209. As depicted in **FIG. 2**, switch 203 may be implemented on a separate processor (e.g., a separate general-purpose processor such as a CPU, a GPU, or the like, or a separate special-purpose processor such as an FPGA or other ASIC). Additionally or alternatively, switch 203 may be implemented, at least in part, by one or more primitives of PLD 209, e.g., as described below with respect to **FIG. 6**.

[0030] Although not depicted in **FIG. 2**, compiler 207 may transfer database 201, e.g., according to a generated mapping, or one or more sets of hardware configuration instructions to PLD 209 via an interface. For example, the interface may comprise a peripheral component interconnect (PCI) bus, a PCI express bus, or the like. Accordingly, the interface may facilitate data transfer to and from PLD 209.

[0031] **FIG. 3** depicts an exemplary configuration 300 for database acceleration in one or more PLD (e.g., PLD(s) 307). In the example of **FIG. 3**, PLD 307 includes operator primitives 309 and on-chip memory 311 and is in communication with off-chip memory 312. By storing all or at least a portion of a database in on-chip memory 311, PLD 307 may accelerate database retrievals relative to retrievals on conventional systems. Moreover, PLD 307 or one or more processors external to PLD 307 (e.g., a separate general-purpose processor such as a CPU, a GPU, or the like, or a separate special-purpose processor such as an FPGA or other ASIC) may provide a message switch 303 and a data switch 305. Although not shown in **FIG. 3**, on-chip memory 311 may have associated primitives for data transfer in and out. Such primitives may additionally perform minor database operations, such as nulling database elements, bit shifting database elements, incrementing database elements, decrementing database elements, or the like.

[0032] Data switch 305 may queue database retrieval requests such that requests for elements from different portions of on-chip memory 311 or off-chip memory 312 may be concurrently executed without collision. Similarly, message switch 303 may queue operations such that operations may be concurrently execution by different primitives of operator primitives 309 without collision. Data switch 305 may receive requests from processor 301 and message switch 303 may receive operations from processor 301.

[0033] As used herein, concurrently may include both parallelism (e.g., one portion of operator primitives 309 executing commands at the same time that another portion of

operator primitives 309 is executing commands, one portion of primitives associated with on-chip memory 311 executing commands at the same time that another portion of primitives associated with on-chip memory 311 is executing commands, or one portion of primitives associated with off-chip memory 312 executing commands at the same time that another portion of primitives associated with off-chip memory 312 is executing commands) as well as concurrency (e.g., one portion of operator primitives 309 executing commands along with another portion of operator primitives 309 such that the commands are executed intermittently during the same time period, one portion of primitives associated with on-chip memory 311 executing commands along with another portion of primitives associated with on-chip memory 311 such that the commands are executed intermittently during the same time period, or one portion of primitives associated with off-chip memory 312 executing commands along with another portion of primitives associated with off-chip memory 312 such that the commands are executed intermittently during the same time period).

[0034] As further shown in **FIG. 3**, in some embodiments, processor 301 may execute retrievals directly from on-chip memory 311 as well as sending operations to message switch 303 and data switch 305. Other embodiments may prevent such direct retrievals in order to ensure that data switch 305 may prevent collision between concurrent retrievals. Similarly, as shown in **FIG. 3**, in some embodiments, operator primitives 309, primitives performing retrievals from on-chip memories 311, or primitives performing retrievals from off-chip memories 312 may communicate directly, such as, for example, in situations where operator primitives 309 require one or more retrievals to complete a minor operation (e.g., nulling database elements, bit shifting database elements, incrementing database elements, decrementing database elements, or the like). Other embodiments may prevent such direct retrievals such that operator primitives 309 send any requests to data switch 305 (either directly or via message switch 303) in order to ensure that data switch 305

may prevent collision between concurrent retrievals. In such embodiments, message switch 303 may send required requests to data switch 305 before queuing an operation for execution. Accordingly, data switch 305 (or primitives performing retrievals from on-chip memories 311, or primitives performing retrievals from off-chip memories 312) may provide the requested elements to message switch 303 for queuing with the operation or directly to operator primitives 309 assigned to the operation.

[0035] **FIG. 4** depicts an exemplary configuration 400 for implementing an intelligent memory array in a PLD (e.g., PLD(s) 307 of **FIG. 3**). As depicted in **FIG. 4**, an input 401 may comprise a retrieval request received by the PLD for execution in a transferred database (e.g., database 201 of **FIG. 2**). In some embodiments, input 401 may additionally or alternatively comprise a minor operation (e.g., nulling one or more database elements, bit shifting one or more database elements, incrementing one or more database elements, decrementing one or more database elements, or the like) received by the PLD for execution in the transferred database. In some embodiments, one or more arbitors (e.g., arbitor 403) may queue a plurality of inputs to avoid collision. Arbitor 403 may comprise one or more primitives of the PLD configured to perform such queuing. Arbitor 403 may additionally queue results of retrievals or minor database operations for returning as output 409.

[0036] As further depicted in **FIG. 4**, arbitor 403 may also provide, based on content of input 401, one or more portions of input 401 to one or more primitives forming a command executor 405. For example, if input 401 comprises a plurality of database requests, arbitor 403 may divide the requests such that different portions of command executor 405 may execute the requests concurrently. Additionally or alternatively, if input 401 comprises one or more requests with one or more minor database operations, arbitor 403 may divide the request(s) from the operation(s) such that different portions of command executor 405 may execute the request(s) concurrently with the operation(s). In another example, if input 401

comprises one or more requests with one or more database operations, arbitor 403 may divide the request(s) from the operation(s) such that the operation(s) are sent to a message switch of the PLD (or, in some embodiments, directly to a portion of an operator array of the PLD, as described below with respect to **FIG. 5**) and the request(s) are executed by command executor 405. In yet another example, input 401 may comprise one or more requests received from an operator array of the PLD executing one or more operations that require the one or more requests; accordingly, arbitor 403 may provide input 401 to command executor 403 for retrieval and delivery to a message switch of the PLD (or, in some embodiments, directly to the portion of the operator array of the PLD from which input 401 originated).

[0037] Command executor 405 may comprise one or more primitives configured to perform read operations on memory banks 407-1, . . . , 407-n. Memory banks 407-1, . . . , 407-n may comprise on-chip memory banks or off-chip memory banks. Moreover, memory banks 407-1, . . . , 407-n may store a transferred database (e.g., database 201 of **FIG. 2**). In some embodiments, command executor 405 may also be configured to execute minor database operations. For example, command executor 405 may comprise one or more primitives configured to perform write operations (e.g., a nulling operation, a bit shifting operation, an incrementing operation, a decrementing operation, or the like) on memory banks 407-1, . . . , 407-n. Examples of such operations are shown in Table 1 below.

Instruction	Description
Chunk_clear	Clear a memory chunk to all zeroes
Chunk_send	Send a chunk of memory to the data switch
Chunk_deposit	Receive a chunk of memory from the memory switch and deposit the content to a specified destination
Bit_write	Bit-wise read-modify-write

Table 1

[0038] **FIG. 5** depicts an exemplary configuration 500 for implementing an operator array in a PLD (e.g., PLD(s) 505). As depicted in **FIG. 5**, an input 501 may comprise a

database operation received by the PLD for execution in a transferred database (e.g., database 201 of **FIG. 2**). In some embodiments, input 501 may additionally or alternatively comprise a database request (e.g., for one or more elements from the transferred database) for use in the database operation. In some embodiments, one or more arbitors (e.g., arbitor 503) may queue a plurality of inputs to avoid collision. Arbitor 503 may comprise one or more primitives of the PLD configured to perform such queuing. Arbitor 503 may additionally queue results of database operations for returning as output 509.

[0039] As further depicted in **FIG. 5**, arbitor 503 may also provide, based on content of input 501, one or more portions of input 501 to one or more sets of primitives corresponding to an operation included in input 501. In the example of **FIG. 5**, PLD(s) 505 includes regular expression primitives 507a, integer comparison primitives 507b, string comparison primitives 507c, compaction primitives 507d, and GZip primitives 507e. Further examples of possible operations provided by processor array 500 are shown in Table 2 below.

Instruction	Description
HASH1B	Calculate the hash value of 1-byte from selected hash function
HASH2B	Calculate the hash value of 2-byte from selected hash function
HASH4B	Calculate the hash value of 4-byte from selected hash function
SCMPI	Compare a string against a constant string and set a corresponding flag
MSEN	Send a message to another set of primitives through the message switch
CSEN	Send a command (e.g., a minor database operation) to the intelligent memory
FIFOPU	Push a register into a FIFO (first-in, first-out) queue
FIFOPO	Pop top of a FIFO (first-in, first-out) queue into a register
FIFORE	Read top of a FIFO (first-in, first-out) queue without popping

Table 2

[0040] Accordingly, if, for example, input 501 comprises a plurality of database operations, arbitor 503 may divide the requests such that different portions of PLD(s) 505 may execute the operations concurrently. Additionally or alternatively, if input 501 comprises one or more requests with one or more database operations, arbitor 403 may divide the request(s) from the operation(s) such that the request(s) are sent to a message switch of the PLD (or, in some embodiments, directly to a portion of an intelligent memory array of the PLD, as described above with respect to **FIG. 4**) and the operations(s) are executed by portions of PLD(s) 505 (e.g., regular expression primitives 507a, integer comparison primitives 507b, string comparison primitives 507c, compaction primitives 507d, GZip primitives 507e, or the like). In another example, if input 501 comprises one or more minor database operations with one or more other database operations, arbitor 403 may divide the minor operation(s) from the other operation(s) such that the minor operation(s) are sent to a data switch of the PLD (or, in some embodiments, directly to a portion of an intelligent memory array of the PLD, as described above with respect to **FIG. 4**) and the other request(s)

are executed by portions of PLD(s) 505 (e.g., regular expression primitives 507a, integer comparison primitives 507b, string comparison primitives 507c, compaction primitives 507d, GZip primitives 507e, or the like). In yet another example, input 501 may comprise one or more operations received from an intelligent memory array of the PLD executing one or more requests that include the one or more operations; accordingly, arbitor 503 may provide input 501 to portions of PLD(s) 505 (e.g., regular expression primitives 507a, integer comparison primitives 507b, string comparison primitives 507c, compaction primitives 507d, GZip primitives 507e, or the like) for execution and delivery to a data switch of the PLD (or, in some embodiments, directly to the portion of the intelligent memory array of the PLD from which input 501 originated).

[0041] In any of these embodiments, arbitor 503 may further determine if input 501 requires one or more database retrievals for the corresponding operation(s) included in input 501. If so, arbitor 503 may send the required retrieval(s) to a data switch of the PLD (or, in some embodiments, directly to a portion of an intelligent memory array of the PLD, as described above with respect to **FIG. 4**). Accordingly, the intelligent memory array may return the requested elements directly to arbitor 503 or the portion of PLD(s) 505 executing the operation(s) or return the requested elements through the data switch to arbitor 503 or the portion of PLD(s) 505 executing the operation(s). Additionally or alternatively, the portion of PLD(s) 505 executing the operation(s) (such as regular expression primitives 507a, integer comparison primitives 507b, string comparison primitives 507c, compaction primitives 507d, GZip primitives 507e, or the like) may determine the required retrievals and send the same to the data switch (or, in some embodiments, directly to a portion of the intelligent memory array). Accordingly, the intelligent memory array may return the requested elements as described above.

[0042] In some embodiments, one or more multiplexers (e.g., MUX 509) may connect one or more portions of operator array 500 together. In the example of **FIG. 5**, compaction primitives 507d are connected to GZip primitives 507e via multiplexer 509. In this example, PLD(s) 505 may perform a compaction operation and a Gzip operation sequentially without resort to arbitor 503 between the operations. Additional multiplexers may be used to allow for sequential operations without resort to arbitor 503.

[0043] **FIG. 6** depicts an exemplary configuration 600 for data switching or message switching in a PLD. As explained above with respect to **FIGS. 2 and 3**, in some embodiments, configuration 600 may be used in combination with one or more of configuration 400 of **FIG. 4** or configuration 500 of **FIG. 5**.

[0044] As depicted in **FIG. 6**, input 601 may comprise one or more database retrieval(s) or operation(s) received by the PLD for execution on a transferred database (e.g., database 201 of **FIG. 2**). In some embodiments, one or more primitives (not shown) of the PLD may convert input 601 (e.g., a query) to a standardized format for execution. For example, the PLD may receive an 8-bit integer, and the PLD may standardize such an integer to a 64-bit integer or the like.

[0045] As further depicted in **FIG. 6**, one or more input queues (e.g., input queue 603-1, . . . , 603-n) may each provide one or more first-in, first-out storages (e.g., FIFO 603a-1, . . . , 603a-n provided by input queue 603-1). In embodiments where configuration 600 comprises a data switcher, each storage may comprise a fixed-length storage. In embodiments where configuration 600 comprises a message switcher, each storage may comprise a variable-length storage. In other embodiments, configuration 600 may comprise a variable-length storage and be implemented as a data switcher.

[0046] As depicted in **FIG. 6**, each input queue may further provide a scheduler (e.g., scheduler 603b provided by input queue 603-1). The scheduler may extract the queries from

the storages according to one or more scheduling schema (e.g., a weighted round robin technique or the like). Overall scheduler 605 may then select between the schedulers of the input queues (e.g., scheduler 603b of input queue 603-1 and the like) based on one or more scheduling schema, whether the same or difference than the schema implemented by the schedulers of the input queues.

[0047] As further depicted in **FIG. 6**, a multiplexer (e.g., MUX 607) may then direct the queries as scheduled by scheduler 605 to one or more output ports (e.g., output ports 609-1, . . . , 609-n). Each output port may lead to a different portion of the PLD, e.g., an intelligent memory array (e.g., configuration 400 of **FIG. 4**), an operator array (e.g., configuration 500 of **FIG. 5**), or the like.

[0048] As explained above, configuration 600 of **FIG. 6** may be implemented on one or more PLDs implementing intelligent memory array 400 of **FIG. 4** or operator array 500 of **FIG. 5**, at least in part. In such embodiments, the output ports may comprise buses on the one or more PLDs. Additionally or alternatively, configuration 600 of **FIG. 6** may be implemented on one or more processors distinct from the PLDs implementing intelligent memory array 400 of **FIG. 4** or operator array 500 of **FIG. 5**. In such embodiments, the output ports may comprise ports on an interface between the one or more distinct processors and the one or more PLDs.

[0049] **FIG. 7A** depicts a graphical representation of a storage schema 700 for a column-oriented database. As depicted in **FIG. 7A**, column 0 of every row is stored sequentially across memory chunks, then column 1 of every row is stored sequentially thereafter, and the like. **FIG. 7B** depicts a graphical representation of a storage schema 750 for a row-oriented database. Schema 750 is similar to schema 700, but row 0 of every column is stored sequentially across memory chunks, then row 1 of every column is stored sequentially thereafter, and the like.

[0050] By implementing schema 700 and 750, an intelligent memory array of the present disclosure (e.g., array 400 of **FIG. 4**) may have greater flexibility than existing database accelerators. For example, schema 700 and 750 may allow for the same intelligent memory array to be used for row-oriented and column-oriented databases since the storage schema is consistent for both. In one implementation, on-chip memory banks or off-chip memory banks of one or more PLDs may store a transferred database sequentially across chunks according to their orientation, as explained in schema 700 and 750. Thereafter, primitives of the PLDs may agnostically implement database retrievals configured for the orientation of the transferred database.

[0051] **FIG. 8** depicts a data flow 800 of configuring a PLD for database acceleration by a PLD. As depicted in **FIG. 8**, a special- or general-purpose processor receives a database 801 for acceleration. For example, database 801 may comprise a relational database, a graph database, or the like. In embodiments where database 801 comprises a relational database, flow 800 may further include the special- or general-purpose processor determining whether database 801 is row-oriented or column-oriented.

[0052] As further depicted in **FIG. 8**, the special- or general-purpose processor may execute a compiler 803 to transfer database 801 onto the PLD. The special- or general-purpose processor may further execute compiler 803 to generate multiple sets of hardware configuration instructions. For example, a first set of hardware configuration instructions may configure a PLD (e.g., according to configuration 400 or the like) to perform a retrieval from database 801. Additionally or alternatively, another set of hardware configuration instructions may configure a PLD (e.g., according to configuration 500 or the like) to perform a database operation on database 801. Additionally or alternatively, another set of hardware configuration instructions may configure a PLD (e.g., according to configuration 600 or the like) to direct queries to different portions of the PLD (e.g., to one or more portions

configured as an intelligent memory array, to one or more portions configured as an operator array, or the like).

[0053] The special- or general-purpose processor may transmit the hardware configuration instructions from compiler 803 to the PLD, resulting in a PLD configured to function as an accelerated database 805. Accordingly, accelerated database 805 may accept queries for execution on the transferred database by executing the hardware configuration instructions.

[0054] **FIG. 9** is a flowchart of an exemplary method 900 for accelerating a database using a programmable logic device (PLD). Method 900 may be performed by at least one processor (e.g., processor 1001 of system 1000 of **FIG. 10**). Method 900 may apply to any programmable logic device (PLD), such as an FPGA, a PAL, a PLA, a CPLD, or the like.

[0055] At step 901, the at least one processor may configure a first set of hardware configuration instructions to cause performance of retrieving database elements from one or more on-chip memories of at least one programmable logic device. For example, the retrievals may comprise commands in a database query language (such as structured query language (SQL), ArangoDB query language (AQL), or the like). Additionally or alternatively, the retrievals may comprise natural language commands. In such embodiments, the at least one processor may further perform natural language processing on the retrievals to transform the retrievalfrom natural language to a database query language or configure the first set of hardware configuration instructions to perform the same.

[0056] In any of the embodiments described above, the hardware configuration instructions may comprise Verilog, impulse C, or any other hardware description language (HDL). The hardware configuration instructions may configure one or more primitives of the at least one programmable logic device (PLD) such that the at least one PLD may perform the

retrieval. For example, the hardware configuration instructions may configure primitives in accordance with configuration 400 of **FIG. 4**, or the like.

[0057] In some embodiments, the first set of hardware configuration instructions may further cause performance of minor database operations on database elements from one or more on-chip memories of the at least one programmable logic device. For example, the minor database operations may comprise at least one of nulling a database element, bit shifting a database element, incrementing a database element, or decrementing a database element. Additionally or alternatively, the minor database operations may include operations explained above with respect to Table 1. In such embodiments, the first set of hardware configuration instructions may configure one or more primitives in communication with the one or more on-chip memories to perform the one or more minor database operations. For example, the hardware configuration instructions may configure primitives in accordance with configuration 400 of **FIG. 4**, or the like.

[0058] In any of the embodiments described above, the first set of hardware configuration instructions may comprise instructions for configuring one or more primitives as at least one arbitor (e.g., arbitor 403 of **FIG. 4**) configured to direct a query for a database element to at least one primitive of the PLD closer to an on-chip memory storing the database element than the arbitor.

[0059] At step 903, the at least one processor may configure a second set of hardware configuration instructions to cause performance of one or more database operations on the at least one PLD. For example, the operations may comprise commands in a database query language (such as structured query language (SQL), ArangoDB query language (AQL), or the like). Additionally or alternatively, the operations may comprise natural language commands. In such embodiments, the at least one processor may further perform natural language processing on the operations to transform the operations from natural language to a database

query language or configure the second set of hardware configuration instructions to cause performance of the same.

[0060] In any of the embodiments described above, the hardware configuration instructions may comprise Verilog, impulse C, or any other hardware description language (HDL). The hardware configuration instructions may configure one or more primitives of the at least one programmable logic device (PLD) such that the at least one PLD may perform the operations. For example, the hardware configuration instructions may configure primitives in accordance with configuration 500 of **FIG. 5**, or the like. Accordingly, the second set of hardware configuration instructions may comprise instructions for configuring one or more primitives as at least one arbitor (e.g., arbitor 503 of **FIG. 5**) configured to direct a query to one of a plurality of sets of primitives based on a type of the query.

[0061] In some embodiments, the database operations may comprise at least one of integer comparison, string comparison, regular expression matching, a compaction command, or a gzip command. Additionally or alternatively, the minor database operations may include operations explained above with respect to Table 2.

[0062] At step 905, the at least one processor may configure a third set of hardware configuration instructions to cause direction of a query to a portion of the at least one PLD based on at least a portion of contents of the query. For example, the query may comprise one or more commands in a database query language (such as structured query language (SQL), ArangoDB query language (AQL), or the like). Additionally or alternatively, the query may comprise one or more natural language commands. In such embodiments, the at least one processor may further perform natural language processing on the query to transform the query from natural language to a database query language or configure the third set of hardware configuration instructions to cause performance of the same. In some embodiments, the query may include retrievals for which the first set of hardware configuration instructions

are to cause to be processed and operations for which the second set of hardware configuration instructions are to cause to be processed.

[0063] In any of the embodiments described above, the hardware configuration instructions may comprise Verilog, impulse C, or any other hardware description language (HDL). The hardware configuration instructions may configure one or more primitives of the at least one programmable logic device (PLD) such that the at least one PLD may queue and direct the query. For example, the hardware configuration instructions may configure primitives in accordance with configuration 600 of **FIG. 6**, or the like. Accordingly, the third set of the hardware configuration instructions may comprise instructions for configuring one or more primitives as one or more virtual output queues. In such embodiments, the third set of the hardware configuration instructions may comprise instructions for configuring a first set of primitives as a first set of virtual output queues for database retrievals and a second set of primitives as a second set of virtual output queues for database operations.

[0064] At step 907, the at least one processor may transfer the first set of hardware configuration instructions, the second set of hardware configuration instructions, and the third set of hardware configuration instructions to the at least one PLD to configure the at least one PLD accordingly. For example, the at least one processor may use one or more interfaces to transmit the hardware configuration instructions.

[0065] In some embodiments, the instructions of steps 901, 903, or 905 may be preconfigured rather than configured by the at least one processor. For example, the at least one processor may retrieve the instructions from at least one storage or receive the instructions over at least one network.

[0066] At step 909, the at least one processor may transfer a database to the at least one PLD. For example, the at least one processor may use one or more interfaces to transmit the hardware configuration instructions. In some embodiments, transferring the database may

comprise copying database elements of the database to the one or more on-chip memories. In such embodiments, copying the database elements may comprise copying the database elements from at least one memory storing the database from which the database elements are copied. The at least one memory storing the database may be, at least in part, the same memory storing instructions the at least one processor executes to perform method 900. Alternatively, the at least one memory storing the database may be distinct from one or more memories storing instructions the at least one processor executes to perform method 900.

[0067] At step 911, the at least one processor may receive a query for execution against the database and send the query to the at least one PLD for execution against the transferred database. For example, the at least one processor may use one or more interfaces to transmit the query. In some embodiments, the at least one processor may parse the query from a natural language command or a database query language (such as structured query language (SQL), ArangoDB query language (AQL), or the like) command to Verilog, impulse C, or any other HDL before transmitting to the configured PLD. Additionally or alternatively, one or more primitives of the PLD may perform natural language processing or other parsing before executing the query.

[0068] Furthermore, at step 911, in response to the query, the at least one processor may receive results from the at least one PLD. For example, the at least one processor may receive the results over one or more interfaces. The results may include a simple Boolean expression (e.g., indicating whether the query is satisfied by the database or whether a desired modification to the database was successfully executed), a list of indices of elements satisfying the query, the actual elements satisfying the query, or the like.

[0069] Furthermore, at step 911, the at least one processor may output the results to the user. For example, the at least one processor may store a file including the results, transmit the results using one or more packets over one or more computer networks, or

display the results to the user (e.g., using text or one or more graphical user interfaces (GUIs)).

[0070] Method 900 may allow for concurrent execution of a plurality of queries, as explained above with respect to **FIG. 3**. For example, the sets of hardware configuration instructions may be generated to configure the PLD to execute a plurality of retrievals and operators within a query. In such embodiments, the at least one processor may receive a plurality of queries and sending the plurality of queries to the configured PLD for queuing and concurrent execution.

[0071] Consistent with the present disclosure, the example method 900 may include additional steps. For example, in some embodiments, method 900 may include constructing a mapping between a memory storing the database and one or more on-chip memories and one or more off-chip memories of the PLD and transferring the database according to the mapping. The database may be stored locally or remotely. Accordingly, the memory storing the database may comprise at least one memory storing instructions for method 900. Additionally or alternatively, one or more external memories accessible by the at least one processor over one or more computer networks may comprise the memory storing the database.

[0072] Although described above with respect to one or more on-chip memories, the at least one PLD may additionally or alternatively store at least part of the transferred database in one or more off-chip memories. In such embodiments, the first set of hardware configuration instructions may further cause performance of retrieving database elements from the one or more off-chip memories.

[0073] **FIG. 10** is a depiction of an example system 1000 for accelerating a database using a programmable logic device (PLD), consistent with embodiments of the present disclosure. Although depicted as a server in **FIG. 10**, system 1000 may comprise any

computer, such as a desktop computer, a laptop computer, a tablet, or the like, configured to execute, for example, method 900 of **FIG. 9**.

[0074] As depicted in **FIG. 10**, server 1000 may have a processor 1001. Processor 1001 may comprise a single processor or a plurality of processors. For example, processor 1001 may comprise a CPU, a GPU, a reconfigurable array (e.g., an FPGA or other ASIC), or the like.

[0075] Processor 1001 may be in operable connection with a memory 1003, an input/output module 1005, and a network interface controller (NIC) 1007. Memory 1003 may comprise a single memory or a plurality of memories. In addition, memory 1003 may comprise volatile memory, non-volatile memory, or a combination thereof. As depicted in **FIG. 10**, memory 1003 may store one or more operating systems 1009, and a compiler 1011. For example, compiler 1011 may include instructions to generate one or more sets hardware configuration instructions for configuring one or more PLDs for database acceleration (e.g., as explained in steps 901, 902, and 903 of method 900 of **FIG. 9**). Therefore, compiler 1011 may cooperate with the one or more PLDs to perform method 900 of **FIG. 9**.

[0076] Input/output module 1005 may store and retrieve data from one or more databases 1015. For example, database(s) 1015 may include elements for mapping to the one or more PLDs for accelerating database(s) 1015 using the one or more PLDs, as described above.

[0077] NIC 1007 may connect server 1000 to one or more computer networks. In the example of **FIG. 10**, NIC 1007 connects server 1000 to the Internet. Server 1000 may receive data and instructions over a network using NIC 1007 and may transmit data and instructions over a network using NIC 1007. Moreover, server 1000 may transmit data and commands to and from the one or more PLDs using NIC 1007 or another interface, as described above.

[0078] The foregoing description has been presented for purposes of illustration. It is not exhaustive and is not limited to precise forms or embodiments disclosed. Modifications and adaptations of the embodiments will be apparent from consideration of the specification and practice of the disclosed embodiments. For example, the described implementations include hardware, but systems and methods consistent with the present disclosure can be implemented with hardware and software. In addition, while certain components have been described as being coupled to one another, such components may be integrated with one another or distributed in any suitable fashion.

[0079] Moreover, while illustrative embodiments have been described herein, the scope includes any and all embodiments having equivalent elements, modifications, omissions, combinations (e.g., of aspects across various embodiments), adaptations or alterations based on the present disclosure. The elements in the claims are to be interpreted broadly based on the language employed in the claims and not limited to examples described in the present specification or during the prosecution of the application, which examples are to be construed as nonexclusive. Further, the steps of the disclosed methods can be modified in any manner, including reordering steps or inserting or deleting steps.

[0080] The features and advantages of the disclosure are apparent from the detailed specification, and thus, it is intended that the appended claims cover all systems and methods falling within the true spirit and scope of the disclosure. As used herein, the indefinite articles “a” and “an” mean “one or more.” Similarly, the use of a plural term does not necessarily denote a plurality unless it is unambiguous in the given context. Further, since numerous modifications and variations will readily occur from studying the present disclosure, it is not desired to limit the disclosure to the exact construction and operation illustrated and described, and accordingly, all suitable modifications and equivalents may be resorted to, falling within the scope of the disclosure.

[0081] As used herein, unless specifically stated otherwise, the term "or" encompasses all possible combinations, except where infeasible. For example, if it is stated that a database may include A or B, then, unless specifically stated otherwise or infeasible, the database may include A, or B, or A and B. As a second example, if it is stated that a database may include A, B, or C, then, unless specifically stated otherwise or infeasible, the database may include A, or B, or C, or A and B, or A and C, or B and C, or A and B and C.

[0082] Other embodiments will be apparent from consideration of the specification and practice of the embodiments disclosed herein. It is intended that the specification and examples be considered as example only, with a true scope and spirit of the disclosed embodiments being indicated by the following claims.

WHAT IS CLAIMED IS:

1. A system for accelerating a database using at least one programmable logic device (PLD), comprising:

at least one memory configured to store instructions; and

at least one processor configured to execute the instructions to cause the system to perform operations comprising:

transferring a first set of hardware configuration instructions to cause a first portion of the at least one PLD to execute retrievals of database elements from one or more on-chip memories;

transferring a second set of hardware configuration instructions to cause a second portion of the at least one PLD to execute one or more database operations;

transferring a third set of hardware configuration instructions to cause a third portion of the at least one PLD to direct an incoming database query to the first portion or the second portion of the at least one PLD based on at least a portion of contents of the query;

transferring a database to the at least one PLD;

receiving a query for execution against the database and sending the query to the at least one PLD for execution against the transferred database; and

in response to the query, receiving results from the at least one PLD.

2. The system of claim 1, wherein the one or more database operations comprise at least one of integer comparison, string comparison, regular expression matching, a compaction command, or a gzip command.

3. The system of claim 1 or 2, wherein the at least one PLD is further configured to store at least part of the transferred database in one or more off-chip memories.

4. The system of claim 3, wherein the first set of hardware configuration instructions further cause the first portion of the at least one PLD to execute retrievals of database elements from the one or more off-chip memories.

5. The system of any one of claims 1-4, wherein the first set of hardware configuration instructions further cause one or more primitives of the first portion of the at least one PLD to function as at least one arbitor configured to direct a query for a database element to at least one primitive of the first portion of the at least one PLD closer to an on-chip memory storing the database element than the arbitor.

6. The system of any one of claims 1-5, wherein the second set of hardware configuration instructions further cause one or more primitives of the second portion of the at least one PLD to function as at least one arbitor configured to direct a query to one of a plurality of sets of primitives of the second portion of the at least one PLD based on a type of the query.

7. The system of any one of claims 1-6, wherein the first set of hardware configuration instructions further cause one or more primitives of the first portion of the at least one PLD in communication with the one or more on-chip memories to perform one or more minor database operations.

8. The system of claim 7, wherein the one or more minor database operations comprise at least one of nulling a database element, bit shifting a database element, incrementing a database element, or decrementing a database element.

9. The system of any one of claims 1-8, wherein transferring the database comprises copying database elements of the database to the one or more on-chip memories.

10. The system of claim 9, wherein copying the database elements comprises copying the database elements from at least one memory storing the database from which the database elements are copied.

11. The system of claim 10, wherein the at least one memory comprises the memory storing the database from which the database elements are copied.

12. The system of any one of claims 1-11, wherein the third set of hardware configuration instructions further cause one or more primitives of the third portion of the at least one PLD to function as one or more virtual output queues.

13. The system of claim 12, wherein the third set of hardware configuration instructions cause a first set of primitives of the third portion of the at least one PLD to function as a first set of virtual output queues for database retrievals and a second set of primitives of the third portion of the at least one PLD to function as a second set of virtual output queues for database operations.

14. The system of any one of claims 1-13, wherein the at least one PLD comprises a field-programmable gate array (FPGA).

15. The system of any one of claims 1-14, wherein at least one of the first set of hardware configuration instructions, the second set of hardware configuration instructions, or the third set of hardware configuration instructions comprise preconfigured instructions.

16. The system of any one of claims 1-14, wherein the operations further comprise configuring at least one of the first set of hardware configuration instructions, the second set of hardware configuration instructions, or the third set of hardware configuration instructions comprise preconfigured instructions.

17. A method for accelerating a database using at least one programmable logic device (PLD), comprising:

transferring a first set of hardware configuration instructions to cause a first portion of the at least one PLD to execute retrievals of database elements from one or more on-chip memories;

transferring a second set of hardware configuration instructions to cause a second portion of the at least one PLD to execute one or more database operations;

transferring a third set of hardware configuration instructions to cause a third portion of the at least one PLD to direct an incoming database query to the first portion or the second portion of the at least one PLD based on at least a portion of contents of the query;

transferring a database to the at least one PLD;

receiving a query for execution against the database and sending the query to the at least one PLD for execution against the transferred database; and

in response to the query, receiving results from the at least one PLD.

18. The method of claim 17, wherein the one or more database operations comprise at least one of integer comparison, string comparison, regular expression matching, a compaction command, or a gzip command.

19. The method of claim 17 or 18, wherein the at least one PLD is further configured to store at least part of the transferred database in one or more off-chip memories.

20. The method of claim 19, wherein the first set of hardware configuration instructions furthe cause the first portion of the at least one PLD to execute retrievals of database elements from the one or more off-chip memories.

21. The method of any one of claims 17-20, wherein the first set of hardware configuration instructions further cause one or more primitives of the first portion of the at least one PLD to function as at least one arbitor configured to direct a query for a database element to at least one primitive of the first portion of the at least one PLD closer to an on-chip memory storing the database element than the arbitor.

22. The method of any one of claims 17-21, wherein the second set of hardware configuration instructions further cause one or more primitives of the second portion of the at least one PLD to function as at least one arbitor configured to direct a query to one of a plurality of sets of primitives of the second portion of the at least one PLD based on a type of the query.

23. The method of any one of claims 17-22, wherein the first set of hardware configuration instructions further cause one or more primitives of the first portion of the at least one PLD in communication with the one or more on-chip memories to perform one or more minor database operations.

24. The method of claim 23, wherein the one or more minor database operations comprise at least one of nulling a database element, bit shifting a database element, incrementing a database element, or decrementing a database element.

25. The method of any one of claims 17-24, wherein transferring the database comprises copying database elements of the database to the one or more on-chip memories.

26. The method of claim 25, wherein copying the database elements comprises copying the database elements from at least one memory storing the database from which the database elements are copied.

27. The method of claim 26, wherein the at least one memory comprises the memory storing the database from which the database elements are copied.

28. The method of any one of claims 17-27, wherein the third set of hardware configuration instructions further cause one or more primitives of the third portion of the at least one PLD to function as one or more virtual output queues.

29. The method of claim 28, wherein the third set of hardware configuration instructions cause a first set of primitives of the third portion of the at least one PLD to function as a first

set of virtual output queues for database retrievals and a second set of primitives of the third portion of the at least one PLD to function as a second set of virtual output queues for database operations.

30. The method of any one of claims 17-29, wherein the at least one PLD comprises a field-programmable gate array (FPGA).

31. The method of any one of claims 17-30, wherein at least one of the first set of hardware configuration instructions, the second set of hardware configuration instructions, or the third set of hardware configuration instructions comprise preconfigured instructions.

32. The system of any one of claims 17-30, wherein the method further comprises configuring at least one of the first set of hardware configuration instructions, the second set of hardware configuration instructions, or the third set of hardware configuration instructions comprise preconfigured instructions.

33. A non-transitory computer-readable storage medium storing a set of instructions that is executable by one or more processors to cause the one or more processors to perform a method for accelerating a database using at least one programmable logic device (PLD), the method comprising:

transferring a first set of hardware configuration instructions to cause a first portion of the at least one PLD to execute retrievals of database elements from one or more on-chip memories;

transferring a second set of hardware configuration instructions to cause a second portion of the at least one PLD to execute one or more database operations;

transferring a third set of hardware configuration instructions to cause a third portion of the at least one PLD to direct an incoming database query to the first portion or the second portion of the at least one PLD based on at least a portion of contents of the query;

transferring a database to the at least one PLD;

receiving a query for execution against the database and sending the query to the at least one PLD for execution against the transferred database; and

in response to the query, receiving results from the at least one PLD.

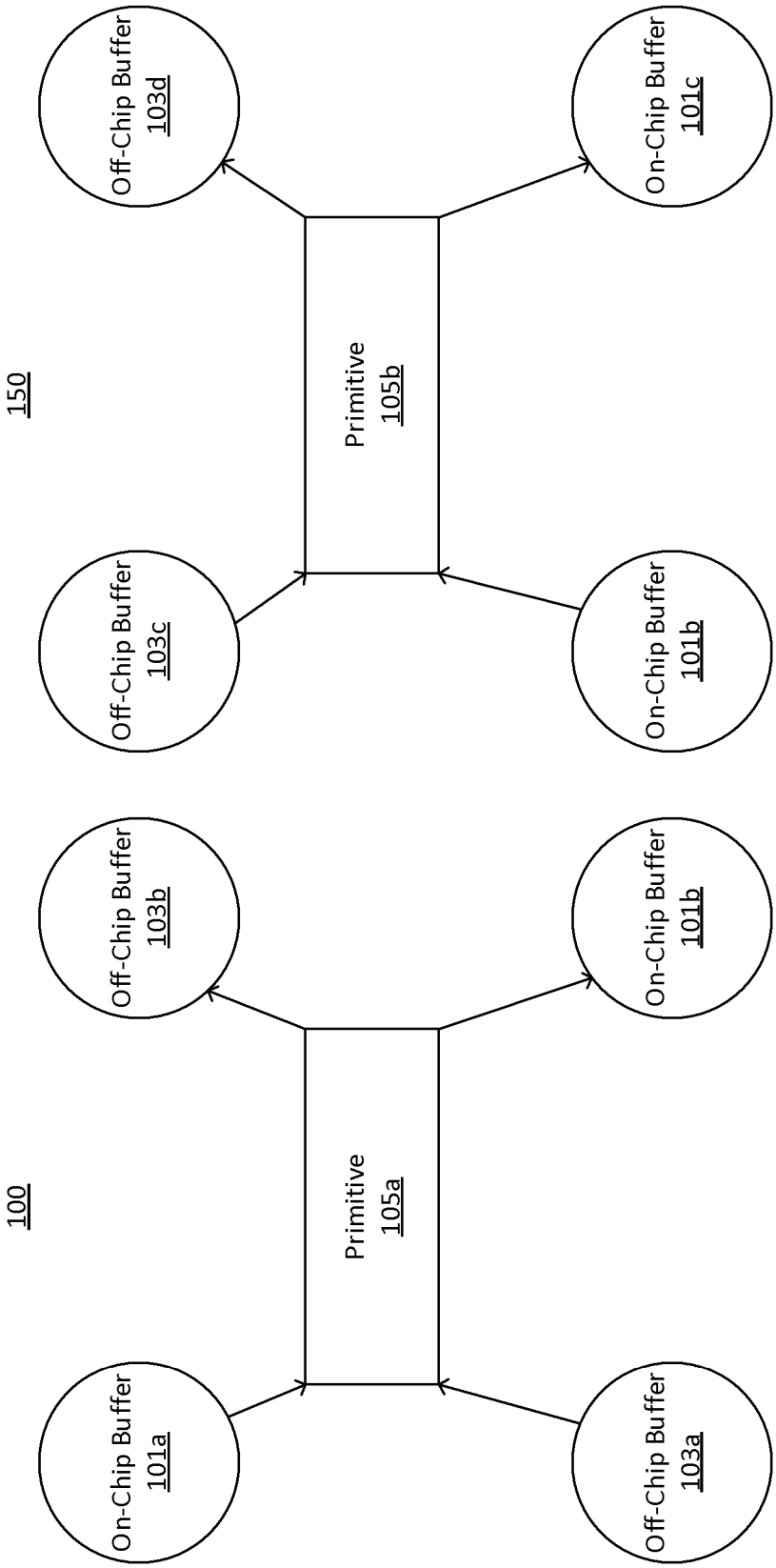
100
On-Chip Buffer 101a
Off-Chip Buffer 103b
Primitive 105a
On-Chip Buffer 101b
Off-Chip Buffer 103a
150
Off-Chip Buffer 103d
Off-Chip Buffer 103c
Primitive 105b
On-Chip Buffer 101c
On-Chip Buffer 101b

FIG. 1

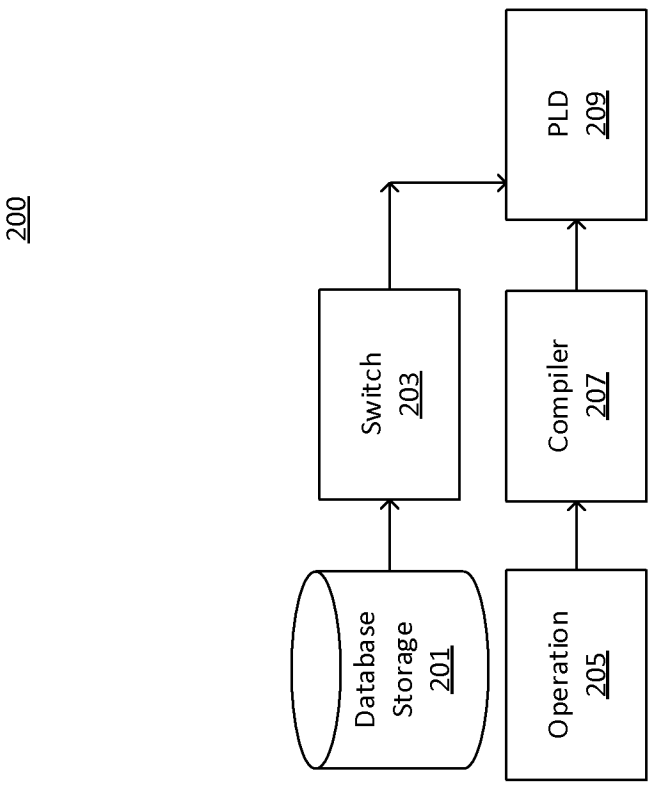
200
Database Storage 201
Switch 203
Operation 205
Compiler 207
PLD 209

FIG. 2

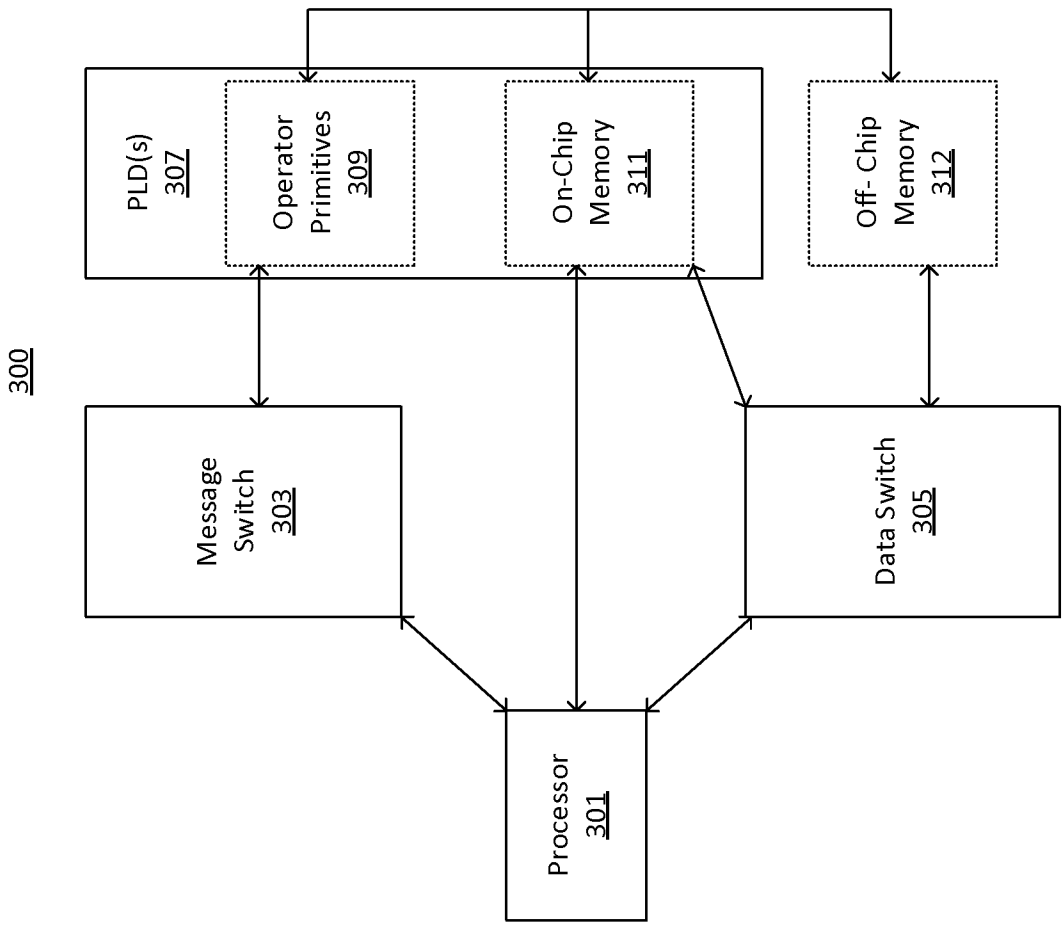
300
PLD(s)
307
Operator
Primitives
309
On-Chip
Memory
311
Off- Chip
Memory
312
Message
Switch
303
Data Switch
305
Processor
301

FIG. 3

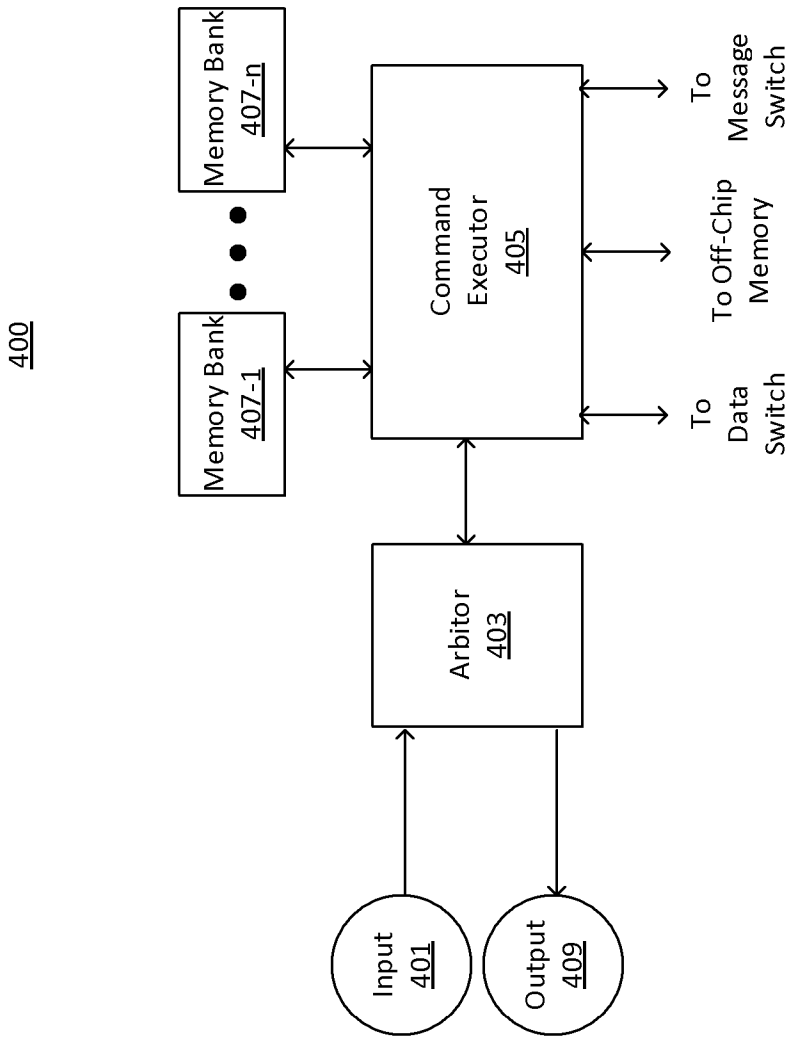
400
Memory Bank 407-n
Memory Bank 407-1
Command Executor 405
To Message Switch
To Off-Chip Memory
To Data Switch
Arbitor 403
Input 401
Output 409

FIG. 4

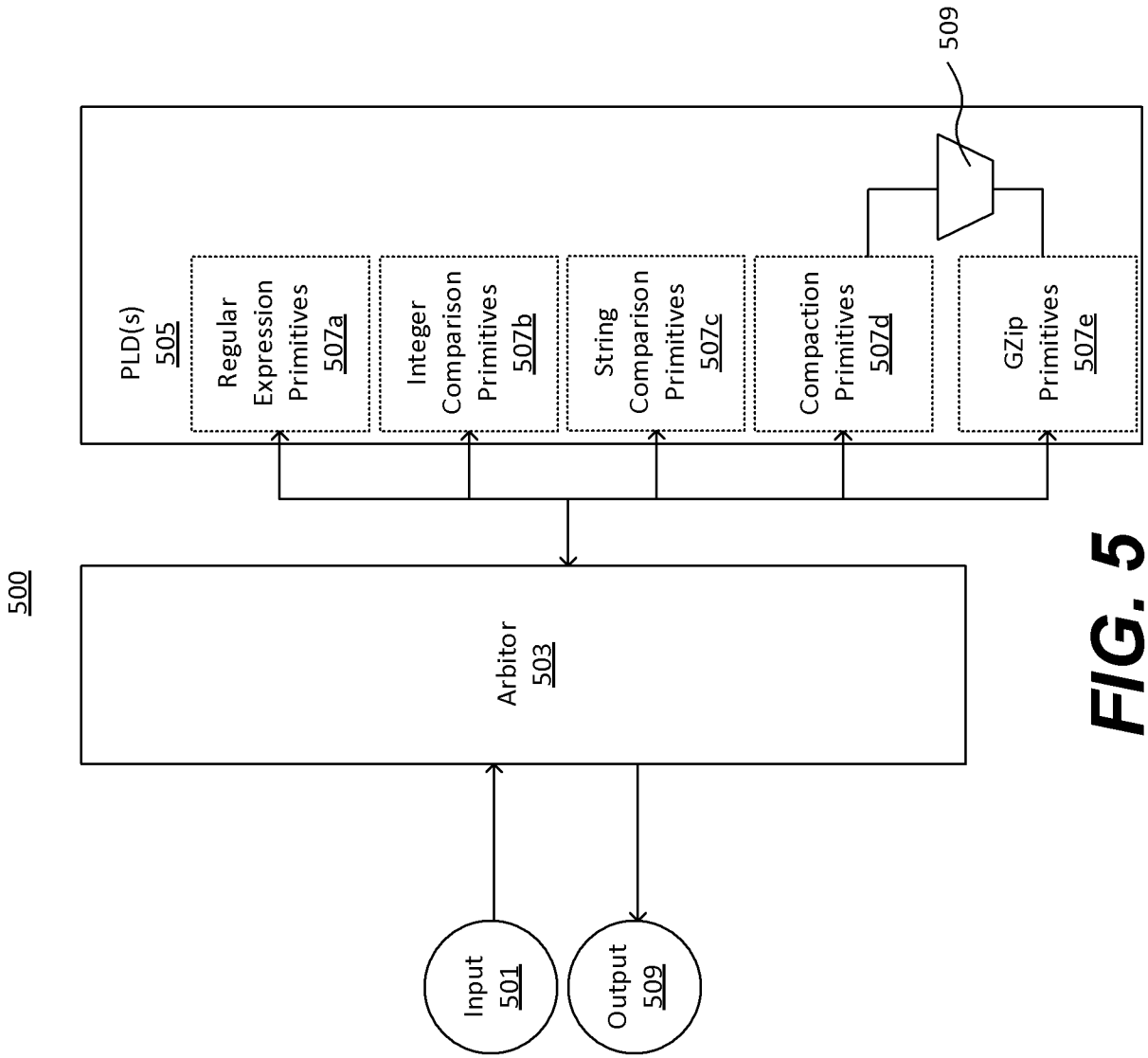
500
Input 501
Output 509
Arbitor 503
PLD(s) 505
Regular Expression Primitives 507a
Integer Comparison Primitives 507b
String Comparison Primitives 507c
Compaction Primitives 507d
GZip Primitives 507e
509

FIG. 5

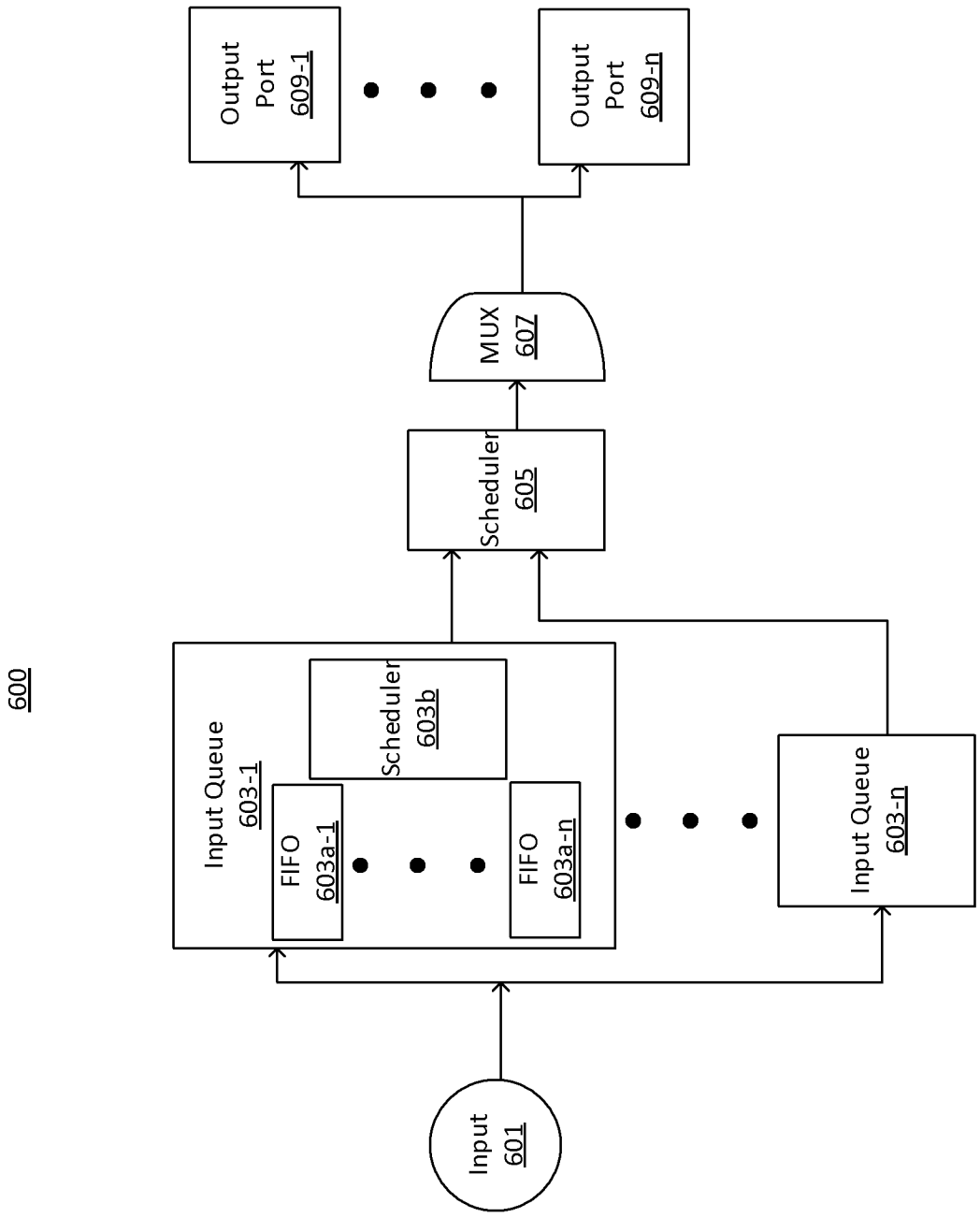
600
Input
601
Input Queue
603-1
FIFO
603a-1
FIFO
603a-n
Scheduler
603b
Input Queue
603-n
Scheduler
605
MUX
607
Output
Port
609-1
Output
Port
609-n

FIG. 6

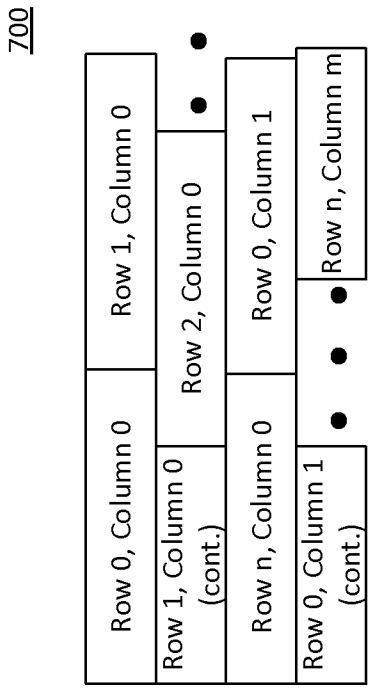
700
Row 0, Column 0
Row 1, Column 0
Row 1, Column 0 (cont.)
Row 2, Column 0
Row n, Column 0
Row 0, Column 1
Row 0, Column 1 (cont.)
Row n, Column m

FIG. 7A

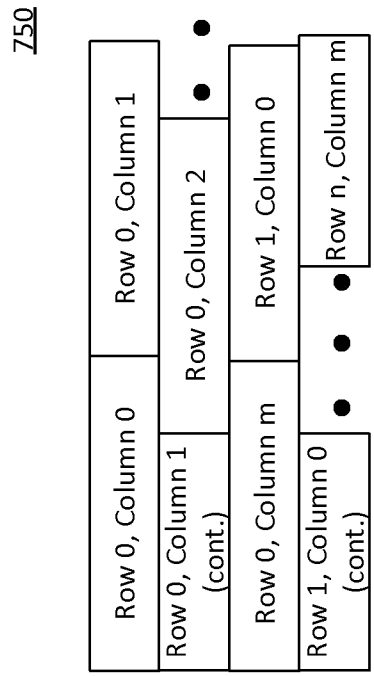
750
Row 0, Column 0
Row 0, Column 1
Row 0, Column 1 (cont.)
Row 0, Column 2
Row 0, Column m
Row 1, Column 0
Row 1, Column 0 (cont.)
Row n, Column m

FIG. 7B

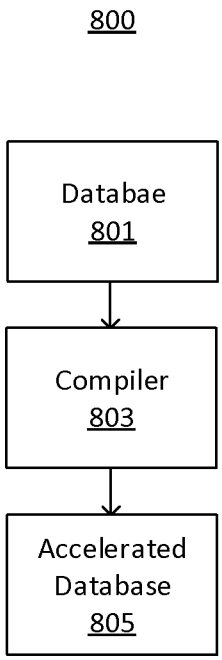

FIG. 8

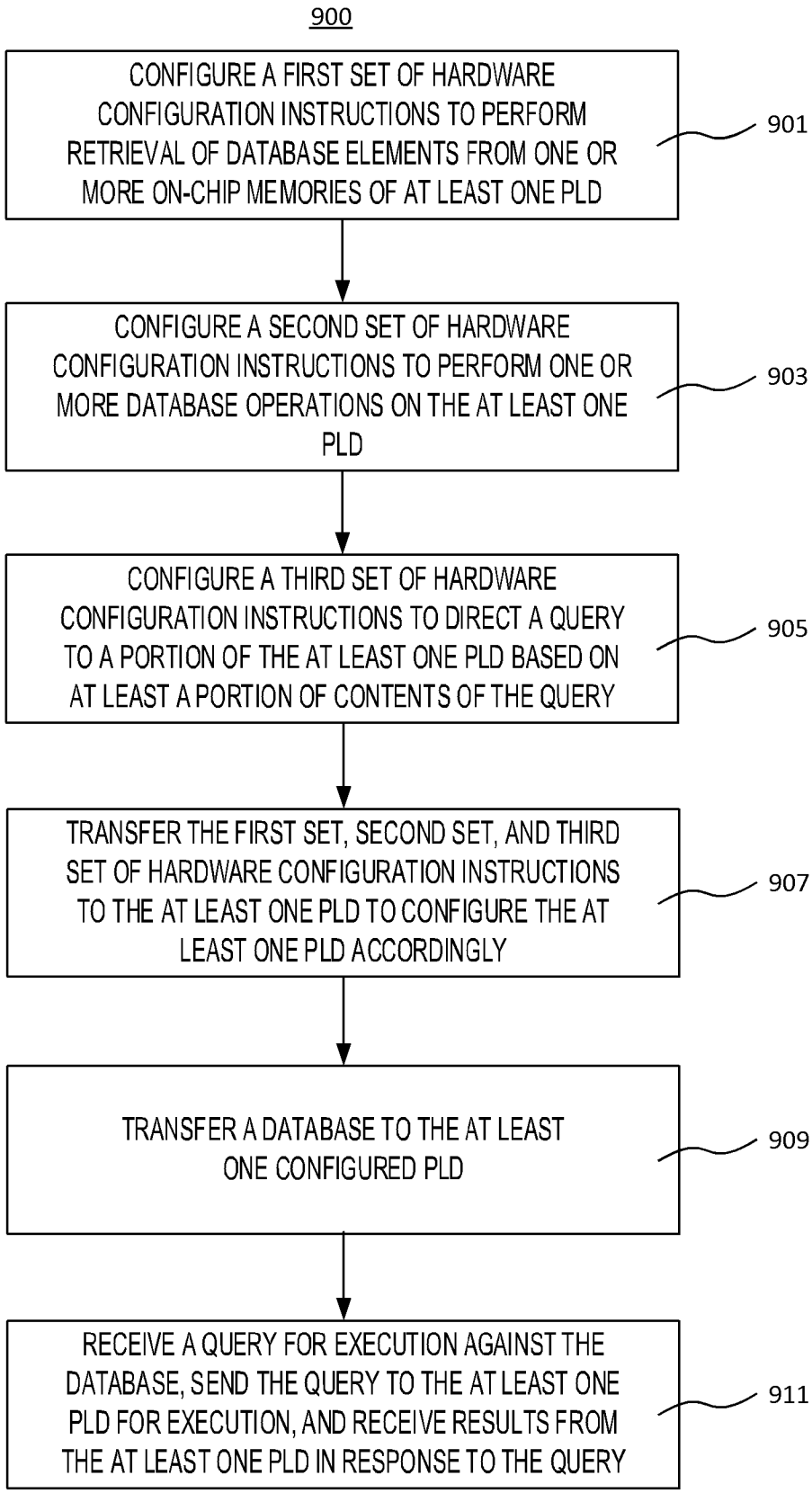
900
CONFIGURE A FIRST SET OF HARDWARE CONFIGURATION INSTRUCTIONS TO PERFORM RETRIEVAL OF DATABASE ELEMENTS FROM ONE OR MORE ON-CHIP MEMORIES OF AT LEAST ONE PLD
901
CONFIGURE A SECOND SET OF HARDWARE CONFIGURATION INSTRUCTIONS TO PERFORM ONE OR MORE DATABASE OPERATIONS ON THE AT LEAST ONE PLD
903
CONFIGURE A THIRD SET OF HARDWARE CONFIGURATION INSTRUCTIONS TO DIRECT A QUERY TO A PORTION OF THE AT LEAST ONE PLD BASED ON AT LEAST A PORTION OF CONTENTS OF THE QUERY
905
TRANSFER THE FIRST SET, SECOND SET, AND THIRD SET OF HARDWARE CONFIGURATION INSTRUCTIONS TO THE AT LEAST ONE PLD TO CONFIGURE THE AT LEAST ONE PLD ACCORDINGLY
907
TRANSFER A DATABASE TO THE AT LEAST ONE CONFIGURED PLD
909
RECEIVE A QUERY FOR EXECUTION AGAINST THE DATABASE, SEND THE QUERY TO THE AT LEAST ONE PLD FOR EXECUTION, AND RECEIVE RESULTS FROM THE AT LEAST ONE PLD IN RESPONSE TO THE QUERY
911

FIG. 9

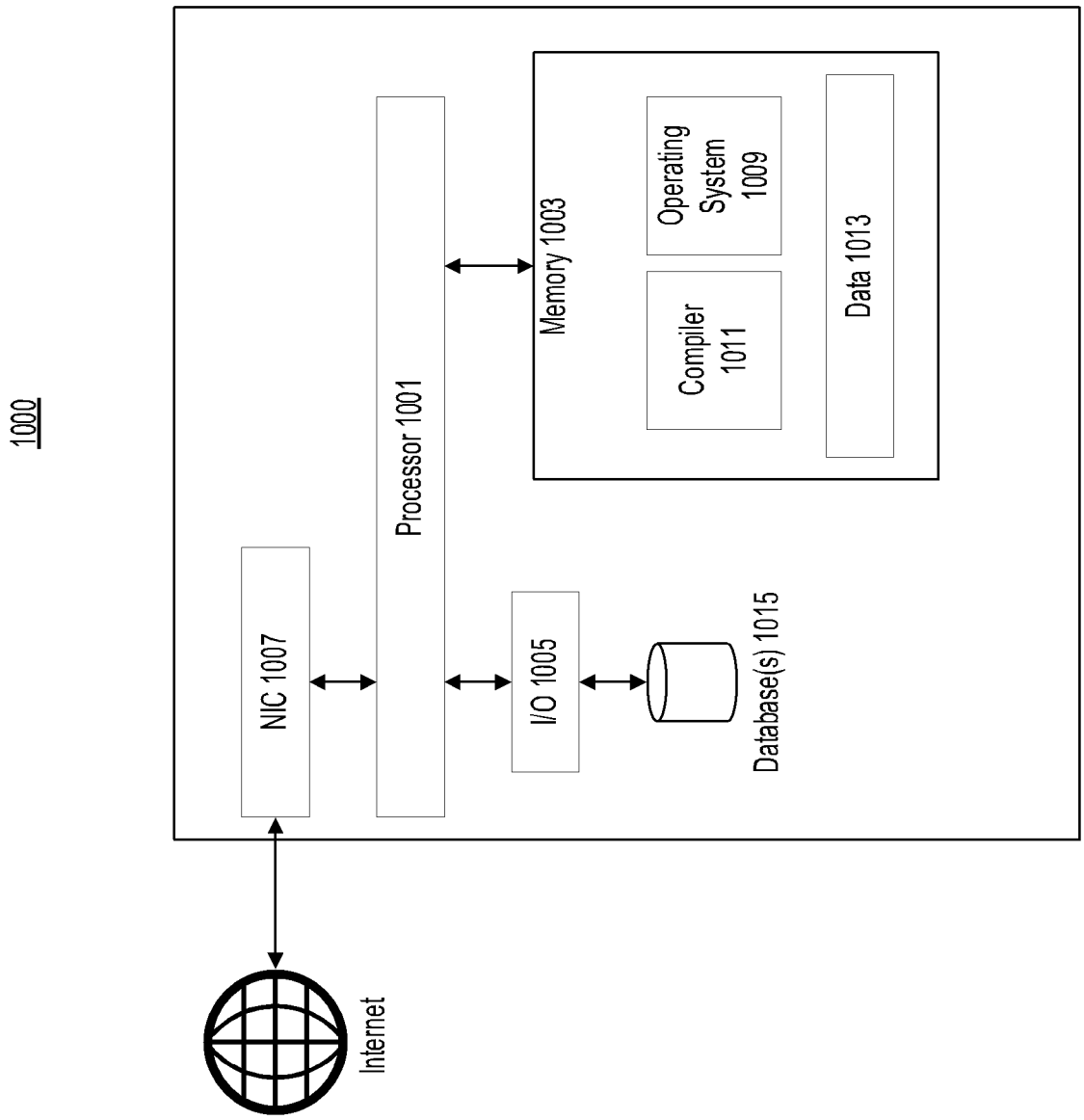
1000
Internet
NIC 1007
Processor 1001
I/O 1005
Database(s) 1015
Memory 1003
Compiler 1011
Operating System 1009
Data 1013

FIG. 10

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2019/071086

A. CLASSIFICATION OF SUBJECT MATTER

G06F 12/00(2006.01)i

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)

G06F; G06K

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

CNABS, DWPI, CNKI, SIPOABS; database, accelerate, FPGA, PLD, query, search,

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 7464088 B1 (SAGE-N RES INC) 09 December 2008 (2008-12-09) the whole document	1-33
A	CN 105589938 A (THIRD RES INST MIN PUBLIC SECURITY) 18 May 2016 (2016-05-18) the whole document	1-33
A	CN 108846364 A (SHENZHEN SURFILTER TECHNOLOGY DEV CO LTD, ETC.) 20 November 2018 (2018-11-20) the whole document	1-33

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

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"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

"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

"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

"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

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
15 August 2019	**29 August 2019**
Name and mailing address of the ISA/CN **National Intellectual Property Administration, PRC** **6, Xitucheng Rd., Jimen Bridge, Haidian District, Beijing 100088** **China** Facsimile No. **(86-10)62019451**	Authorized officer **MENG,Tiange** Telephone No. **86-10-62411653**

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2019/071086

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
US 7464088 B1	09 December 2008	None	
CN 105589938 A	18 May 2016	None	
CN 108846364 A	20 November 2018	None	