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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
02 July 2020 (02.07.2020)

WIPO | PCT

(10) International Publication Number
WO 2020/133391 A1

(51) International Patent Classification:
G05B 19/05 (2006.01) *G06F 16/90* (2019.01)

(21) International Application Number:
PCT/CN2018/125541

(22) International Filing Date:
29 December 2018 (29.12.2018)

(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, CA 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). **SUN, Hua**; 525 Almanor Ave, 4th Floor, Sunnyvale, CA 94085 (US). **LI, Longxiao**; 525 Almanor Ave, 4th Floor, Sunnyvale, CA 94085 (US).

(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, 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))*

(54) Title: SYSTEMS AND METHODS FOR EFFICIENTLY SCANNING A DATABASE USING HARDWARE ACCELERATION

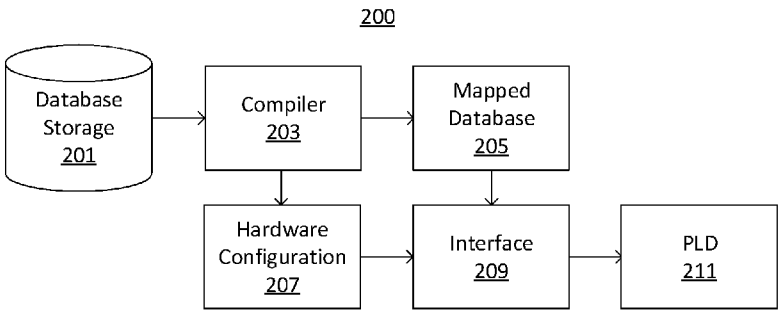

FIG. 2

(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 using a PLD for a database scan operation may include receiving, from a user, a query for execution on a database; configuring hardware configuration instructions based on the query and transmitting the hardware configuration instructions to at least one programmable logic device (PLD) to configure the PLD according to the hardware configuration instructions; sending the query to the configured PLD for execution against the database; in response to the query, receiving results from the PLD; and outputting the results to the user.

WO 2020/133391 A1

SYSTEMS AND METHODS FOR EFFICIENTLY SCANNING A DATABASE USING HARDWARE ACCELERATION

TECHNICAL FIELD

[1] 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 efficiently using programmable logic devices for database scan operations. 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

[2] Field-programmable gate arrays (FPGAs) and other programmable logic device (PLDs) are generally more efficient for operations including comparisons, such as database operations, than conventional processing hardware, such as central processing units (CPUs), graphics processing units (GPUs), or the like. However, FPGAs and other PLDs are usually custom designed for particular functions. Moreover, many PLDs are configured at execution time, reducing the benefits of using PLDs due to long latencies for transferring databases to on- and off-chip memories of the PLDs.

SUMMARY

[3] In some embodiments, a system for using a programmable logic device (PLD) for a database scan operation 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 receiving, from a user, a query

for execution on a database; configuring hardware configuration instructions based on the query; and transmitting the hardware configuration instructions to at least one programmable logic device (PLD) to configure the PLD according to the hardware configuration instructions. The operations may further comprise sending the query to the configured PLD for execution against the database; in response to the query, receiving results from the PLD; and outputting the results to the user.

[4] In some embodiments, a method for using a programmable logic device (PLD) for a database scan operation may comprise receiving, from a user, a query for execution on a database; configuring, using at least one processor, hardware configuration instructions based on the query; and transmitting the hardware configuration instructions to at least one programmable logic device (PLD) to configure the PLD according to the hardware configuration instructions. The method may further comprise sending the query to the configured PLD for execution against the database; in response to the query, receiving results from the PLD; and outputting the results to the user.

[5] 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 using a programmable logic device (PLD) for a database scan operation. The method may comprise receiving, from a user, a query for execution on a database; configuring hardware configuration instructions based on the query; and transmitting the hardware configuration instructions to at least one programmable logic device (PLD) to configure the PLD according to the hardware configuration instructions. The method may further comprise sending the query to the configured PLD for execution against the database; in response to the query, receiving results from the PLD; and outputting the results to the user.

[6] 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.

[7] 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

[8] 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:

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

[10] **FIG. 2** is an exemplary architecture for transferring databases to programmable logic devices (PLDs) and for configuring PLDs to execute queries on transferred databases, according to embodiments of the present disclosure.

[11] **FIG. 3** is a schematic representation of concurrent query type execution in a PLD, according to embodiments of the present disclosure.

[12] **FIG. 4** is a schematic representation of a configuration for regular expression processing in a PLD, according to embodiments of the present disclosure.

[13] **FIG. 5** is a schematic representation of a configuration for string comparison in a PLD, according to embodiments of the present disclosure.

[14] **FIG. 6** is a schematic representation of a configuration for integer comparison in a PLD, according to embodiments of the present disclosure.

[15] **FIG. 7** is a schematic representation of another configuration for string comparison in a PLD, according to embodiments of the present disclosure.

[16] **FIG. 8** is a schematic representation of on-demand query execution by a PLD configured according to embodiments of the present disclosure.

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

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

DETAILED DESCRIPTION

[19] 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.

[20] The disclosed embodiments relate to computer-implemented systems and methods for using a programmable logic device (PLD) for a database scan operation. For example, the operation may include regular expression matching, integer comparisons, string comparisons, or the like. Advantageously, the exemplary embodiments can provide improved efficiency over conventional database scan operations. Embodiments of the present

disclosure can also provide improved on-demand query execution and PLD re-usability across multiple database scan operations.

[21] 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 execute database queries in accordance with the present disclosure.

[22] The embodiments of the present disclosure provide computer-implemented systems and methods for using PLDs to perform database scans. The systems and methods of the present disclosure may provide a technical solution to the technical problem of configuring PLDs for database scan operations without latency at run-time or without having to specially design one or more PLDs for different databases. The systems and methods of the present disclosure may result in efficiency gains as compared with database scan operations on general-purpose processors. Moreover, the systems and methods of the present disclosure may not require customized PLDs or may not result in run-time latencies as existing PLD implementations do.

[23] **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.

[24] 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 perform scan operations on a database. 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.

[25] **FIG. 2** is a schematic representation of a system 200 for transferring databases to PLDs and for configuring PLDs to execute queries 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.

[26] A compiler 203 may comprise one or more instructions executed by at least one processor. For example, compiler 203 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 203 may generate a mapping 205 between database 201 and a programmable logic device (PLD) 211.

For example, compiler 203 may determine a size and spatial location of on- and off-chip memories of PLD 211 and map elements of database 201 to blocks of the on- and off-chip memories. In embodiments where compiler 203 also configures PLD 211 for executing one or more queries on database 201, as described below, compiler 203 may generate mapping 205 such that elements used by one or more configured primitives of PLD 211 are stored in on- or off-chip memories adjacent to such configured primitives.

[27] As further depicted in **FIG. 2**, compiler 203 may generate one or more hardware configuration instructions 207, e.g., as described below in step 903 of method 900 of **FIG. 9**. For example, instructions 207 may configure one or more primitives of PLD 211 to execute one or more queries on database 201 transferred to PLD 211. In some embodiments, instructions 207 may comprise one or more data files in a specification language, such as Verilog, impulse C, or any other hardware description language (HDL).

[28] As further depicted in **FIG. 2**, compiler 203 may transfer database 201, according to mapping 205, or instructions 207 to PLD 211 via an interface 209. For example, interface 209 may comprise a peripheral component interconnect (PCI) bus, a PCI express bus, or the like. Accordingly, interface 209 may facilitate data transfer to and from PLD 211.

[29] **FIG. 3** depicts an exemplary concurrent query type execution 300 in a PLD. In the example of **FIG. 3**, PLD 309 includes three groups of primitives, regular expression primitives 307a, integer comparison primitives 307b, and string comparison primitives 307c. Accordingly, a portion of PLD 309 may execute a regular expression concurrently while another portion of PLD 309 executes an integer comparison or while another portion of PLD 309 may execute a string comparison. As explained above, the expressions may be executed on a database mapped to one or more off-chip memories (not shown) or on-chip memories (not shown) of PLD 309. Accordingly, processor 301 (which may comprise another special-purpose processor or a general-purpose processor) may send queries to PLD 309 via interface

303 for concurrent execution by regular expression primitives 307a, integer comparison primitives 307b, or string comparison primitives 307c.

[30] As used herein, concurrently may include both parallelism (e.g., regular expression primitives 307a executing commands at the same time that integer comparison primitives 307b or string comparison primitives 307c are executing commands) as well as concurrency (e.g., regular expression primitives 307a executing commands along with integer comparison primitives 307b or string comparison primitives 307c such that the commands are executed intermittently during the same time period). In embodiments using concurrency, one or more primitives may be shared between regular expression primitives 307a, integer comparison primitives 307b, or string comparison primitives 307c.

[31] Although depicted using three concurrent groups, PLD 309 may instead include two concurrent groups of primitives or more than three concurrent groups. Moreover, although depicted using regular expression primitives 307a, integer comparison primitives 307b, and string comparison primitives 307c, additional or alternatively primitive groups may be used. For example, PLD 309 may additionally or alternatively include summation primitives, Boolean primitives, or the like.

[32] **FIG. 4** depicts an exemplary configuration 400 for regular expression processing in a PLD. As depicted in **FIG. 4**, an input 401 may comprise the database received by the PLD for comparison with the stored regular expression. In some embodiments, one or more converters 403 of the PLD may convert input 401 (e.g., a string) to a standardized Unicode format for comparison. For example, the PLD may receive a UTF-8 encoded string and converter 403 may standardize such queries to UTF-16. Input 401 may be encoded using encoding schemes, such as American standard code for information interchange (ASCII) or other encoding schemes, or converter 403 may standardize to other encoding scheme, such as International Organization for Standardization/International

Electrotechnical Commissioner (ISO/IEC) 8859 or other encoding schemes. Accordingly, although explained below using Unicode shifters, any other appropriate shifters may be used in the embodiment depicted in **FIG. 4**.

[33] As further depicted in **FIG. 4**, primitives of the PLD may be organized into a plurality of rows of Unicode shifters. Although depicted with two rows, embodiments of the present disclosure may use one row, three row, four rows, or the like. As depicted in **FIG. 4**, each row may include $n+1$ Unicode shifters (e.g., Unicode 405-n to 405-0 and Unicode 413-n to Unicode 413-0).

[34] As further depicted in **FIG. 4**, each row may compare the shifted Unicode using $n+1$ comparators (e.g., comparator 407-n to comparator 407-0 for a first row and comparator 415-n to comparator 415-0 for a second row). By comparing the shifted Unicode to known regular expression patterns, each row may determine whether the corresponding regular expression pattern is included in input 401. Examples of such patterns are shown in Table 1 below.

Wildcard	Description
\d or \D	character classified as digit (\d) or not (\D)
\s or \S	character classified as space, tab, newline, or carriage return (\s) or not (\S)
\w or \W	character classified as word character (\w) or not (\W)
.	any character
[^]	negation of a class (e.g., d, s, or w above)
^	start anchor
$	end anchor
.*	zero or more any character(s)
?	zero or one any character
{n,m}	n through m characters

Table 1

[35] Accordingly, embodiments with additional rows may support additional regular expressions in a query. In some embodiments, fewer comparators may be used in a row, e.g., by transmitting the comparator output to an on- or off-chip memory for storage before inputting to the gate of the row (e.g., gate 409 or gate 417) such that the comparator may be re-used for the comparison with the next shifted Unicode. Although re-use of comparators may reduce temporal efficiency, it may increase spatial efficiency by requiring less primitives from the PLD to implement configuration 400. In some embodiments, regardless of the number of comparators, a gate of each row (e.g., gate 409 or gate 417) may combine outputs from the comparisons, e.g., using an AND logical operation, to determine whether the corresponding regular expression of that row is included in input 401.

[36] As further depicted in **FIG. 4**, additional rows may use the shifted Unicode from the previous row as input, accept input 401, or use a multiplexer (e.g., MUX 411) to select one of the two. A collector 419 may collate results from the gates of the rows (e.g., gate 409 and gate 417) and determine which regular expressions are included in input 401 based on corresponding regular expressions associated with the rows and based on

dependencies therebetween (e.g., established via a multiplexer such as MUX 411). For example, a result from the row may indicate that a negation wildcard is present in input 401, and a result from a row immediately subsequent may indicate that a class wildcard is present in input 401, and collector 419 may use the spatial organization of the rows to determine that a negated class wildcard is present in input 401.

[37] **FIG. 5** depicts an exemplary configuration 500 for string comparison in a PLD. As explained above with respect to **FIG. 3**, in some embodiments, configuration 500 may be used in combination with one or more of configuration 400 of **FIG. 4**, configuration 600 of **FIG. 6**, or configuration 700 of **FIG. 7** on the same PLD. In some embodiments, configuration 500 may be implemented in combination with one or more of configuration 400 of **FIG. 4**, configuration 600 of **FIG. 6**, or configuration 700 of **FIG. 7** using different primitives on the PLD. Alternatively, one or more primitives may be shared between configuration 500 and one or more of configuration 400 of **FIG. 4**, configuration 600 of **FIG. 6**, or configuration 700 of **FIG. 7**. For example, one or more of comparators 505-1, 505-2, . . . , 505-n may also comprise one or more of comparators 407-n . . . 407-0 or 415-n . . . 415-0. Additionally or alternatively, converter 503 may comprise converter 403.

[38] As depicted in **FIG. 5**, input 501 may comprise the string received by the PLD for comparison to the stored database. In some embodiments, one or more converters 503 of the PLD may convert input 501 (e.g., a query) to a standardized format for execution. For example, the PLD may receive a UTF-8 encoded query and converter 503 may standardize such queries to UTF-16. Input 501 may be encoded using encoding schemes, such as American standard code for information interchange (ASCII) or other encoding schemes, or converter 503 may standardize to other encoding schemes, such as International Organization for Standardization/International Electrotechnical Commissioner (ISO/IEC) 8859 or other encoding schemes.

[39] As further depicted in **FIG. 5**, primitives of the PLD may compare the converted input to a plurality of elements in the stored database. For example, configuration 500 may comprise a plurality of comparators (e.g., comparator 505-1, comparator 505-2, . . . , comparator 505-n) each comparing the converted input to stored elements from the database (e.g., Unicode pattern 507-1, Unicode pattern 507-2, . . . , Unicode pattern 507-n, respectively). Although depicted as Unicode patterns, the plurality of comparators may store elements of the database encoded according to any scheme, e.g., the scheme to which input 501 is converted by converter 503.

[40] In some embodiments, fewer comparators may be used than a single comparator for each element in the database, e.g., by transmitting the comparator output to an on- or off-chip memory for storage before input to collector 509 such that the comparator may be re-used for the comparison with another element of the database. Although re-use of comparators may reduce temporal efficiency, it may increase spatial efficiency by requiring less primitives from the PLD to implement configuration 500. In some embodiments, regardless of the number of comparators, collector 509 may combine outputs from the comparisons, e.g., using an OR logical operation, to determine whether the input string 501 is present within the database. Moreover, collector 509 may determine which elements of the database include input 501 based on which comparators sent the corresponding outputs indicating matches (e.g., an output of '1' or another indicator of a true Boolean). For example, a result from comparator 505-1 may indicate that input string 501 is present in element 507-1, a result from comparator 505-2 may indicate that input string 501 is not present in element 507-2, and comparator 505-n may indicate that input string 501 is present in element 507-n; accordingly, collector 509 may determine indices of elements 507-1 and 507-n and output such indices as matches in response to input 501.

[41] **FIG. 6** depicts an exemplary configuration 600 for integer comparison in a PLD. As explained above with respect to **FIG. 3**, in some embodiments, configuration 600 may be used in combination with one or more of configuration 400 of **FIG. 4**, configuration 500 of **FIG. 5**, or configuration 700 of **FIG. 7** on the same PLD. In some embodiments, configuration 600 may be implemented in combination with one or more of configuration 400 of **FIG. 4**, configuration 500 of **FIG. 5**, or configuration 700 of **FIG. 7** using different primitives on the PLD. Alternatively, one or more primitives may be shared between configuration 600 and one or more of configuration 400 of **FIG. 4**, configuration 500 of **FIG. 5**, or configuration 700 of **FIG. 7**. For example, one or more of comparators 603-1, 603-2, . . . , 603-n may also comprise one or more of comparators 505-1, 505-2, . . . , 505-n. Additionally or alternatively, gate 605 may comprise collector 509.

[42] As depicted in **FIG. 6**, input 601 may comprise the integer received by the PLD for comparison to the stored database. 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 a 8-bit integer, and the PLD may standardize such integer to a 64-bit integer or the like.

[43] As further depicted in **FIG. 6**, primitives of the PLD may compare the converted input to a plurality of elements in the stored database. For example, configuration 600 may comprise a plurality of comparators (e.g., comparator 603-1, comparator 603-2, . . . , comparator 603-n), each comparing the converted input to stored elements from the database. The plurality of comparators may store elements of the database encoded according to any scheme, e.g., the scheme to which input 601 is standardized by the PLD.

[44] In some embodiments, fewer comparators may be used than a single comparator for each element in the database, e.g., by transmitting the comparator output to an on- or off-chip memory for storage before input to gate 605 such that the comparator may be

re-used for the comparison with another element of the database. Although re-use of comparators may reduce temporal efficiency, it may increase spatial efficiency by requiring less primitives from the PLD to implement configuration 600. In some embodiments, regardless of the number of comparators, gate 605 may combine outputs from the comparisons, e.g., using an OR logical operation, to determine whether the input integer 601 is present within the database. Although depicted as a gate, in some embodiments, element 605 may further determine which elements of the database include input 601 based on which comparators sent the corresponding outputs indicating matches (e.g., an output of '1' or another indicator of a true Boolean). For example, a result from comparator 603-1 may indicate that input integer 601 is present in a first element, a result from comparator 603-2 may indicate that input integer 601 is also present in a second element, and comparator 603-n may indicate that input integer 601 is not present in a third element; accordingly, element 605 may determine indices of the first and second elements and output such indices as matches in response to input 601.

[45] **FIG. 7** depicts another exemplary configuration 700 for string or integer comparison in a PLD. As explained above with respect to **FIG. 3**, in some embodiments, configuration 700 may be used in combination with one or more of configuration 400 of **FIG. 4**, configuration 500 of **FIG. 5**, or configuration 600 of **FIG. 6** on the same PLD. In some embodiments, configuration 700 may be implemented in combination with one or more of configuration 400 of **FIG. 4**, configuration 500 of **FIG. 5**, or configuration 600 of **FIG. 6** using different primitives on the PLD. Alternatively, one or more primitives may be shared between configuration 700 and one or more of configuration 400 of **FIG. 4**, configuration 500 of **FIG. 5**, or configuration 600 of **FIG. 6**. For example, multiplexer 705 may comprise gate 605, collector 509, or collector 419.

[46] Configuration 700 is similar to configuration 500 and configuration 600 but uses hashes to match input 701 to elements of the database rather than direct comparison. For example, one or more primitives of the PLD (e.g., hashers 703-1, 703-2, . . . , 703-n) may be configured according to different hashing functions applied to both elements of the database and input 701.

[47] As depicted in **FIG. 7**, input 701 may comprise the string or integer received by the PLD for comparison to the stored database. In some embodiments, one or more primitives (not shown) of the PLD may convert input 701 (e.g., a query) to a standardized format for hashing. For example, the PLD may receive a UTF-8 encoded string, and the PLD may standardize such queries to UTF-16. Input 701 may be encoded using encoding schemes, such as American standard code for information interchange (ASCII) or other encoding schemes, or may be standardized to other encoding scheme, such as International Organization for Standardization/International Electrotechnical Commissioner (ISO/IEC) 8859 or other encoding schemes.

[48] In some embodiments, one or more of hashers 703-1, 703-2, . . . , 703-n may be selected based on which hashing algorithms do not cause collision. For example, if at least two elements of the database have the same hash code, the corresponding hasher may be deactivated from use. Multiplexer 705 may perform the selection.

[49] As further depicted in **FIG. 7**, the PLD may search one of more hash tables (e.g., tables 707 and 709) for hashed elements matching one or more results of input 701 as output by one or more of hashers 703-1, 703-2, . . . , 703-n. The use of hashing tables may be faster than direct comparisons, e.g., because the hashes are all the same length. Additionally, the use of hashing tables may allow for searching secured databases because the elements are encoded using one or more of hashers 703-1, 703-2, . . . , 703-n rather than being directly accessible by the PLD.

[50] **FIG. 8** depicts a data flow 800 of an on-demand query execution by a PLD. As depicted in **FIG. 8**, a special- or general-purpose processor receives a query 801 from a user. For example, query 801 may comprise a natural language query or a database language query. In embodiments where query 801 includes a natural language query, flow 800 may further include natural language processing performed by the special- or general-purpose processor.

[51] As further depicted in **FIG. 8**, the special- or general-purpose processor may execute a compiler 803 to transform query 801 into hardware configuration instructions. For example, the hardware configuration instructions may configure a PLD (e.g., according to configuration 400, 500, 600, 700, or the like) to execute a type of query corresponding to a type of query 801. Alternatively, the hardware configuration instructions may comprise a command to execute query 801 according to the PLD as already configured (e.g., according to configuration 400, 500, 600, 700, or the like) by a previous set of hardware configuration instructions such that configured PLD 801 may accept queries (such as query 801) for execution.

[52] The special- or general-purpose processor may transmit the hardware configuration instructions from compiler 803 to the PLD (e.g., configured PLD 805). Accordingly, PLD 801 may execute query 801 by executing the hardware configuration instructions.

[53] In any of the embodiments described above, PLD 805 may store a transferred database upon which query 801 is to be executed. For example, the database may have been mapped to on- or off-chip memories of PLD 801.

[54] **FIG. 9** is a flowchart of an exemplary method 900 for using a programmable logic device (PLD) for a database scan operation. 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.

[55] At step 901, the at least one processor may receive, from a user, a query for execution on a database. For example, the query may comprise a command 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 a natural language command. 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.

[56] At step 903, the at least one processor may configure hardware configuration instructions based on the query. For example, the hardware configuration instructions may comprise Verilog, impulse C, or any other HDL. The hardware configuration instructions may configure one or more primitives of at least one programmable logic device (PLD) such that the at least one PLD may execute the received query. For example, the hardware configuration instructions may configure primitives in accordance with configuration 400, 500, 600, 700, or the like.

[57] For example, the query may comprise at least one of integer comparison, string comparison, or regular expression matching. The at least one processor may therefore generate the hardware configuration instructions such that the at least one PLD may execute the corresponding query type. In some embodiments, the regular expression match may include at least one of negative matching, anchored matching, or Unicode-character matching, as explained above with respect to Table 1.

[58] Furthermore, at step 903, the at least one processor may transmit the hardware configuration instructions to at least one programmable logic device (PLD) to configure the PLD according to the hardware configuration instructions. For example, the at least one

processor may use one or more interfaces (such as interface 209 of **FIG. 2**) to transmit the hardware configuration instructions.

[59] At step 905, the at least one processor may transfer the database to the configured PLD. For example, the at least one processor may use one or more interfaces (such as interface 209 of **FIG. 2**) to transmit the hardware configuration instructions. In some embodiments, step 905 may be omitted such that the configured PLD executes queries on a database stored remotely from the PLD.

[60] At step 907, the at least one processor may send the query to the configured PLD for execution against the database (whether transferred or not). For example, the at least one processor may use one or more interfaces (such as interface 209 of **FIG. 2**) 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.

[61] At step 909, in response to the query, the at least one processor may receive results from the PLD. For example, the at least one processor may receive the results over one or more interfaces (such as interface 209 of **FIG. 2**). As explained above with respect to **FIG. 4, FIG. 5, FIG. 6, and FIG. 7**, the results may include a simple Boolean (e.g., indicating whether the query is satisfied by the database) or a list of indices of elements satisfying the query.

[62] Furthermore, at step 909, 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)).

[63] Method 900 may allow for execution of comparison queries (such as string comparisons, integer comparisons, or the like). Accordingly, in some embodiments, method 900 may further include consolidating a plurality of results from a plurality of comparison primitives implemented on the PLD for outputting to the user. In such embodiments, the consolidating may comprise configuring additional hardware configuration instructions and transmitting the additional hardware configuration instructions to the at least one PLD to configure the PLD to perform the consolidating. For example, the additional hardware configuration instructions may configure one or more primitives of the at least one PLD to function as collector 419 of **FIG. 4**, collector 509 of **FIG. 5**, gate 605 of **FIG. 6**, or the like.

[64] Additionally or alternatively, consolidating the plurality of results may comprise receiving the plurality of results from the PLD and performing the consolidation using the at least one processor. For example, the at least one processor may receive a plurality of results from comparators of the PLD (as explained with respect to **FIG. 5** and **FIG. 6**) or from gates of the PLD (as explained with respect to **FIG. 4**) and determine indices of matching elements of the database based on the comparators or gates from which the results were received.

[65] Method 900 may also allow for concurrent execution of a plurality of queries, as explained above with respect to **FIG. 3**. For example, the hardware configuration instructions may be generated to configure the PLD to execute a plurality of query types. In such embodiments, the at least one processor may receive a second query of a different type than the query and sending the second query to the configured PLD for execution concurrently with the first query. Additionally or alternatively, the hardware configuration instructions may be generated to configure multiple groups of primitives of the PLD to execute the same type of query. In such embodiments, the at least one processor may send

two queries of the same type to different spatial locations of the PLD for concurrent execution.

[66] 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.

[67] **FIG. 10** is a depiction of an example system 1000 for executing database queries on 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**.

[68] 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.

[69] 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, a database mapper

1011a, and a compiler 1011b. For example, mapper 1011a may include instructions to map database elements to one or more PLDs (e.g., as explained above with respect to **FIG. 2**), and compiler 1011b may include instructions to generate hardware configuration instructions for execution of a query on the one or more PLDs (e.g., as explained in step 903 of method 900 of **FIG. 9**). Therefore, mapper 1011a and compiler 1011b may cooperate with the one or more PLDs to perform method 900 of **FIG. 9**.

[70] 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 execution of a query on database(s) 1015, as described above.

[71] 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.

[72] 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.

[73] 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.

[74] 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.

[75] 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.

[76] 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 using a programmable logic device (PLD) for a database scan operation, 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:

receiving, from a user, a query for execution on a database;

configuring hardware configuration instructions based on the query and transmitting the hardware configuration instructions to at least one programmable logic device (PLD) to configure the PLD according to the hardware configuration instructions;

sending the query to the configured PLD for execution against the database;

in response to the query, receiving results from the PLD; and

outputting the results to the user.

2. The system of claim 1, wherein the query comprises at least one of integer comparison, string comparison, or regular expression matching.

3. The system of claim 1, wherein the query comprises a comparison, and the operations further comprise consolidating a plurality of results from a plurality of comparison primitives implemented on the PLD for outputting to the user.

4. The system of claim 3, wherein consolidating the plurality of results comprises configuring additional hardware configuration instructions and transmitting the additional hardware configuration instructions to the at least one PLD to configure the PLD to perform the consolidating.

5. The system of claim 3, wherein consolidating the plurality of results comprises receiving the plurality of results from the PLD and performing the consolidation using the at least one processor.

6. The system of claim 1, wherein the query comprises a regular expression match, and the regular expression match includes at least one of negative matching, anchored matching, or Unicode-character matching.

7. The system of any one of claims 1-6, further comprising an interface configured to communicate data and instructions to and from the PLD.

8. The system of claim 7, wherein the interface comprises a peripheral component interconnect (PCI) bus.

9. The system of any one of claims 1-8, wherein the operations further comprise transferring the database to the configured PLD.

10. The system of claim 9, wherein transferring the database comprises 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.

11. The system of claim 10, wherein the at least one memory comprises the memory storing the database.

12. The system of claim 10, wherein one or more external memories accessible over one or more computer networks comprise the memory storing the database.

13. The system of any one of claims 1-12, wherein the hardware configuration instructions are further generated to configure the PLD to execute a plurality of query types.

14. The system of claim 13, wherein the operations further comprise receiving a second query of a different type from the query and sending the second query to the configured PLD for execution concurrently with the query.

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

16. A method for using a programmable logic device (PLD) for a database scan operation, comprising:

receiving, from a user, a query for execution on a database;

configuring, using at least one processor, hardware configuration instructions based on the query and transmitting the hardware configuration instructions to at least one programmable logic device (PLD) to configure the PLD according to the hardware configuration instructions;

sending the query to the configured PLD for execution against the transferred database;

in response to the query, receiving results from the PLD; and

outputting the results to the user.

17. The method of claim 16, wherein the query comprises at least one of integer comparison, string comparison, or regular expression matching.

18. The method of claim 16, wherein the query comprises a comparison, and the method further comprises consolidating a plurality of results from a plurality of comparison primitives implemented on the PLD for outputting to the user.

19. The method of claim 18, wherein consolidating the plurality of results comprises configuring additional hardware configuration instructions and transmitting the additional

hardware configuration instructions to the at least one PLD to configure the PLD to perform the consolidating.

20. The method of claim 18, wherein consolidating the plurality of results comprises receiving the plurality of results from the PLD and performing the consolidation using the at least one processor.

21. The method of claim 16, wherein the query comprises a regular expression match, and the regular expression match includes at least one of negative matching, anchored matching, or Unicode-character matching.

22. The method of any one of claims 16-21, further comprising using an interface to send the query to the PLD, and receive the results from the PLD.

23. The method of claim 22, wherein the interface comprises a peripheral component interconnect (PCI) bus.

24. The method of any one of claims 16-23, further comprising transferring the database to the configured PLD.

25. The method of claim 24, wherein transferring the database comprises 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.

26. The method of claim 25, wherein the at least one memory comprises the memory storing the database.

27. The method of claim 25, wherein one or more external memories accessible over one or more computer networks comprise the memory storing the database.

28. The method of any one of claims 16-27, wherein the hardware configuration instructions are further generated to configure the PLD to execute a plurality of query types.

29. The method of claim 28, further comprising receiving a second query of a different type than the query and sending the second query to the configured PLD for execution concurrently with the first query.

30. The method of any one of claims 16-29, wherein the PLD comprises a field-programmable gate array (FPGA).

31. 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 using a programmable logic device (PLD) for a database scan operation, the method comprising:

receiving, from a user, a query for execution on a database;

configuring hardware configuration instructions based on the query and transmitting the hardware configuration instructions to at least one programmable logic device (PLD) to configure the PLD according to the hardware configuration instructions;

sending the query to the configured PLD for execution against the database;

in response to the query, receiving results from the PLD; and

outputting the results to the user.

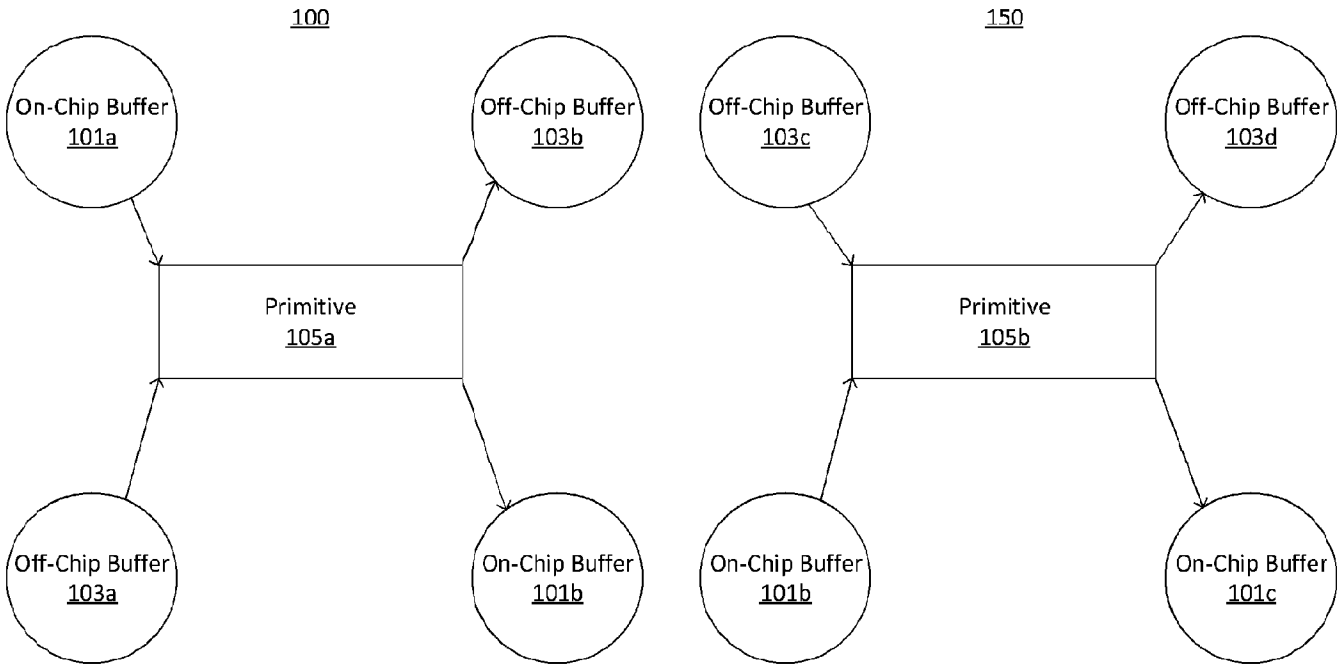
100
On-Chip Buffer
101a
Off-Chip Buffer
103b
Primitive
105a
Off-Chip Buffer
103a
On-Chip Buffer
101b
150
Off-Chip Buffer
103c
Off-Chip Buffer
103d
Primitive
105b
On-Chip Buffer
101b
On-Chip Buffer
101c

FIG. 1

200

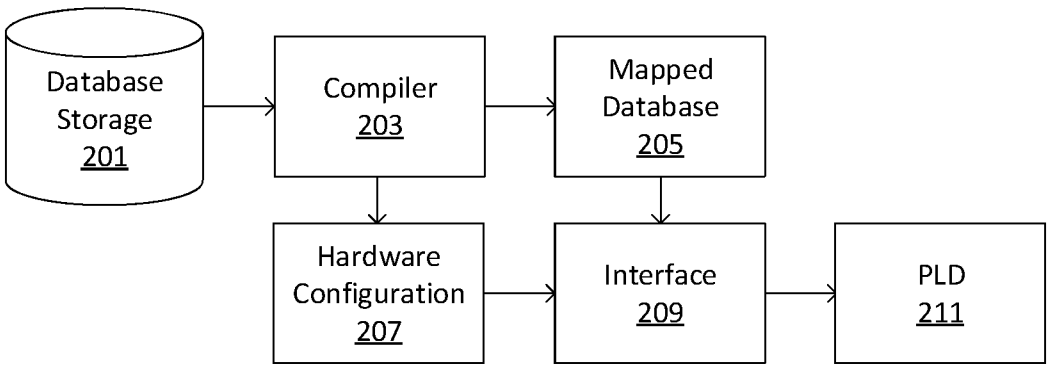
Database
Storage
201
Compiler
203
Mapped
Database
205
Hardware
Configuration
207
Interface
209
PLD
211

FIG. 2

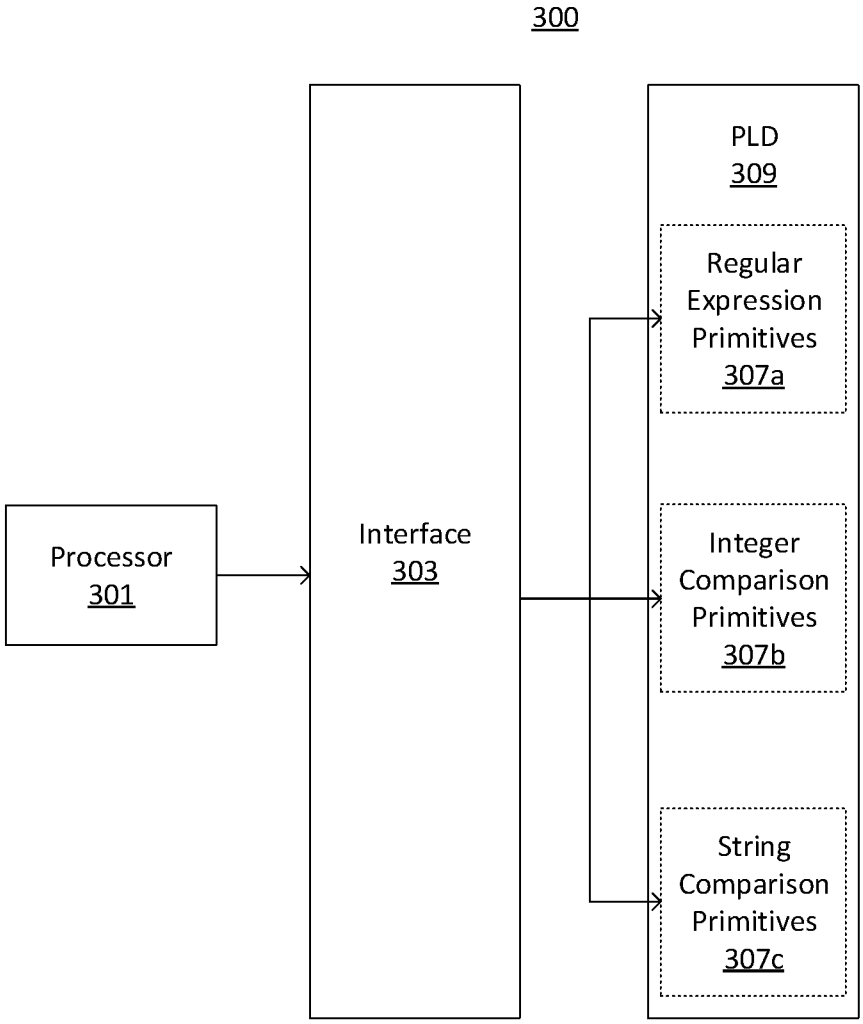
300
Processor
301
Interface
303
PLD
309
Regular
Expression
Primitives
307a
Integer
Comparison
Primitives
307b
String
Comparison
Primitives
307c

FIG. 3

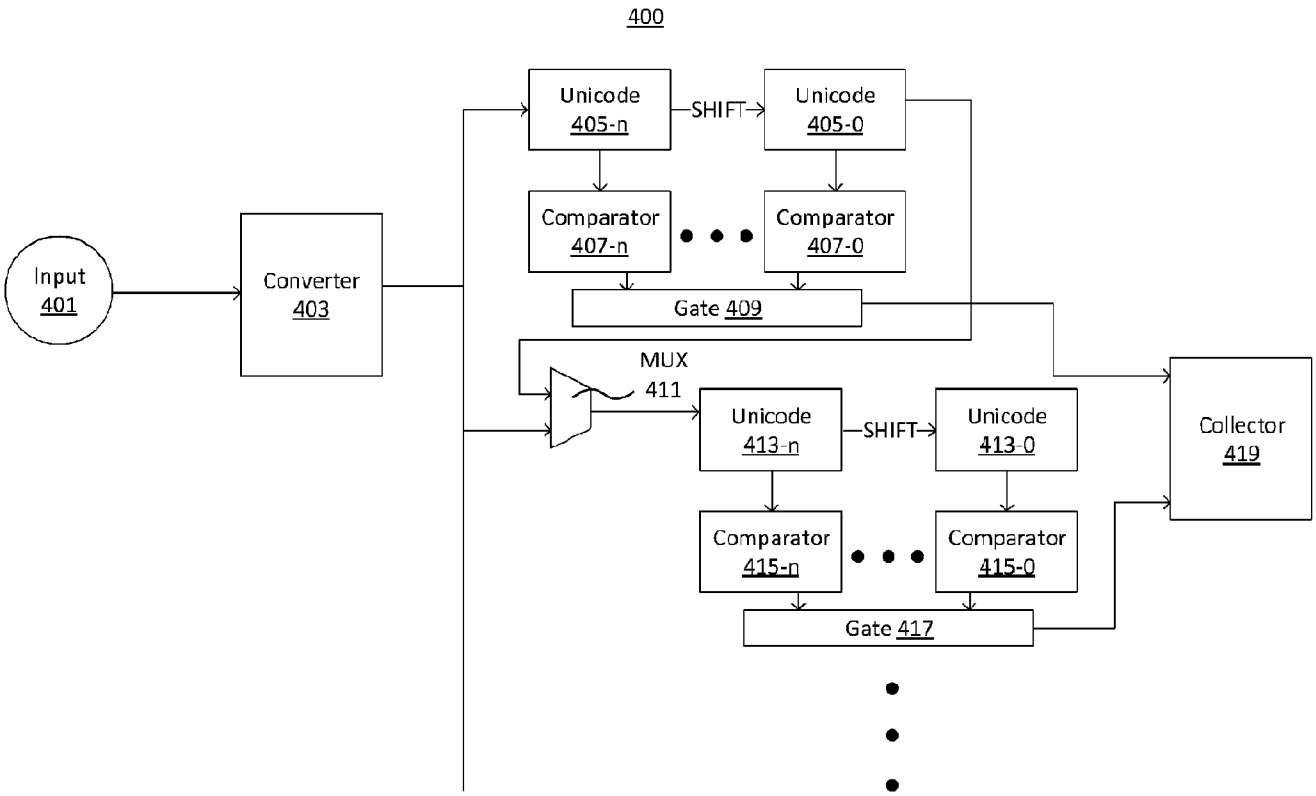
400
Input
401
Converter
403
Unicode
405-n
SHIFT
Unicode
405-0
Comparator
407-n
Comparator
407-0
Gate 409
MUX
411
Unicode
413-n
SHIFT
Unicode
413-0
Comparator
415-n
Comparator
415-0
Gate 417
Collector
419

FIG. 4

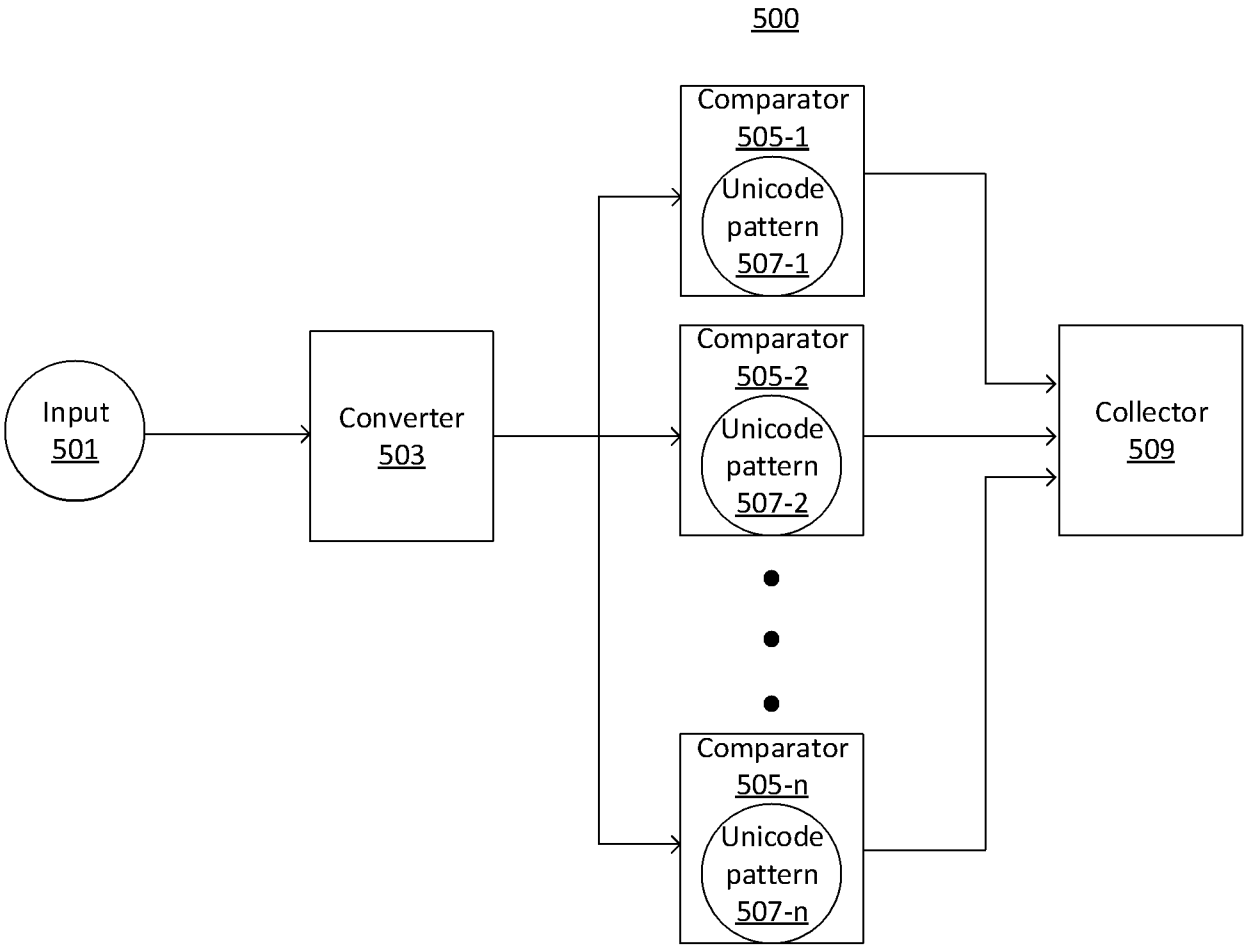
500
Comparator
505-1
Unicode
pattern
507-1
Comparator
505-2
Unicode
pattern
507-2
Comparator
505-n
Unicode
pattern
507-n
Input
501
Converter
503
Collector
509

FIG. 5

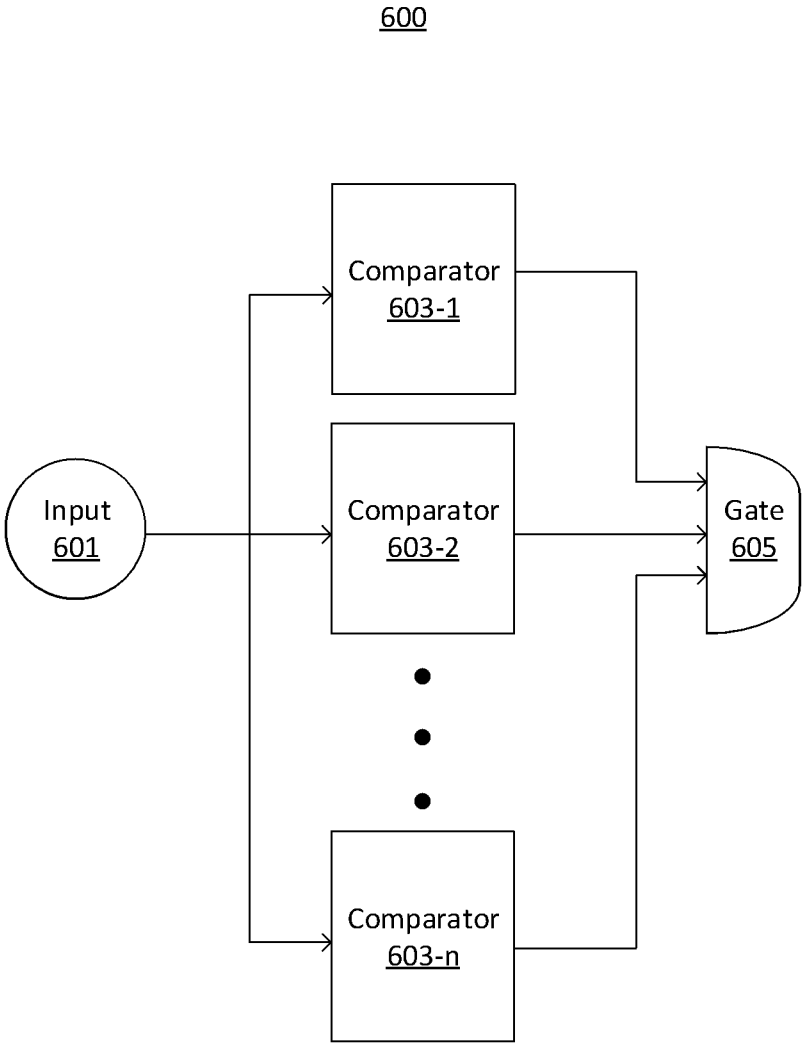
600
Comparator
603-1
Input
601
Comparator
603-2
Gate
605
Comparator
603-n

FIG. 6

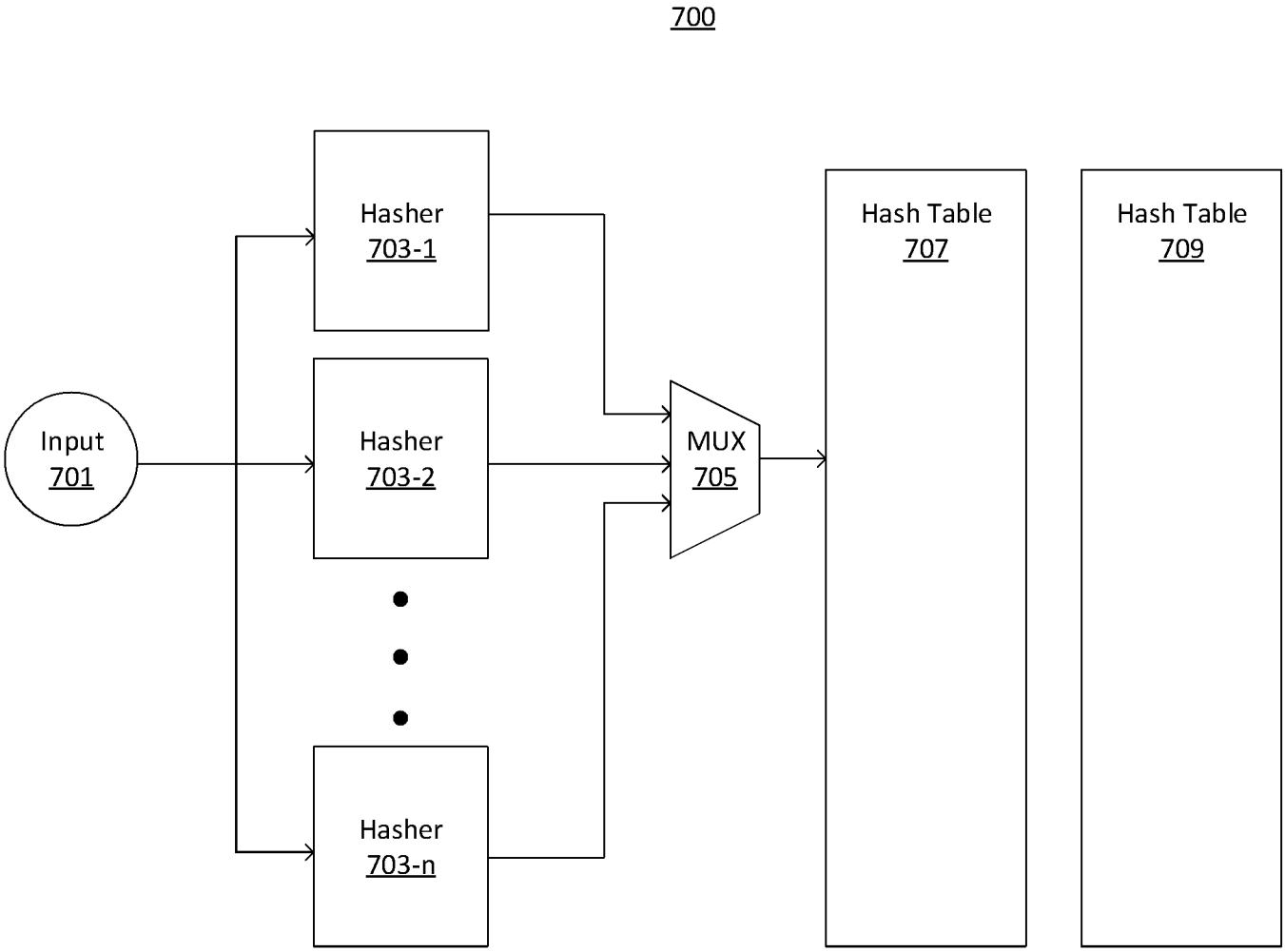
700
Input
701
Hasher
703-1
Hasher
703-2
Hasher
703-n
MUX
705
Hash Table
707
Hash Table
709

FIG. 7

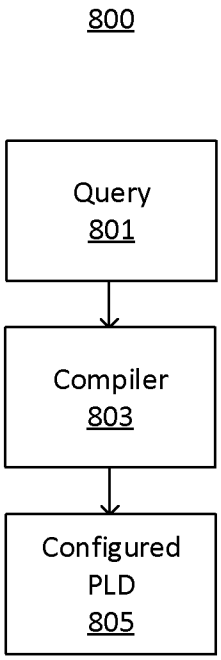

FIG. 8

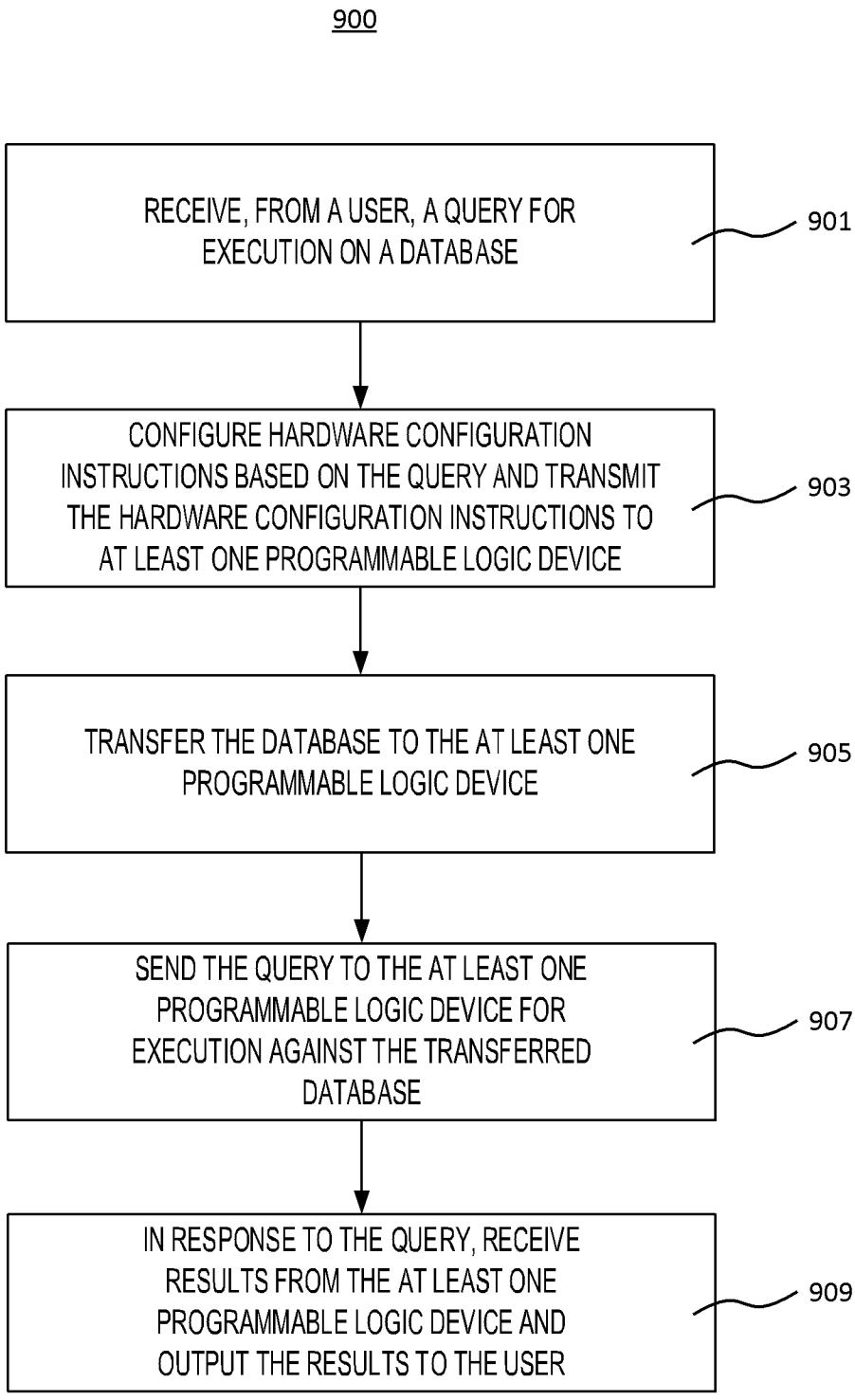
900
RECEIVE, FROM A USER, A QUERY FOR EXECUTION ON A DATABASE
901
CONFIGURE HARDWARE CONFIGURATION INSTRUCTIONS BASED ON THE QUERY AND TRANSMIT THE HARDWARE CONFIGURATION INSTRUCTIONS TO AT LEAST ONE PROGRAMMABLE LOGIC DEVICE
903
TRANSFER THE DATABASE TO THE AT LEAST ONE PROGRAMMABLE LOGIC DEVICE
905
SEND THE QUERY TO THE AT LEAST ONE PROGRAMMABLE LOGIC DEVICE FOR EXECUTION AGAINST THE TRANSFERRED DATABASE
907
IN RESPONSE TO THE QUERY, RECEIVE RESULTS FROM THE AT LEAST ONE PROGRAMMABLE LOGIC DEVICE AND OUTPUT THE RESULTS TO THE USER
909

FIG. 9

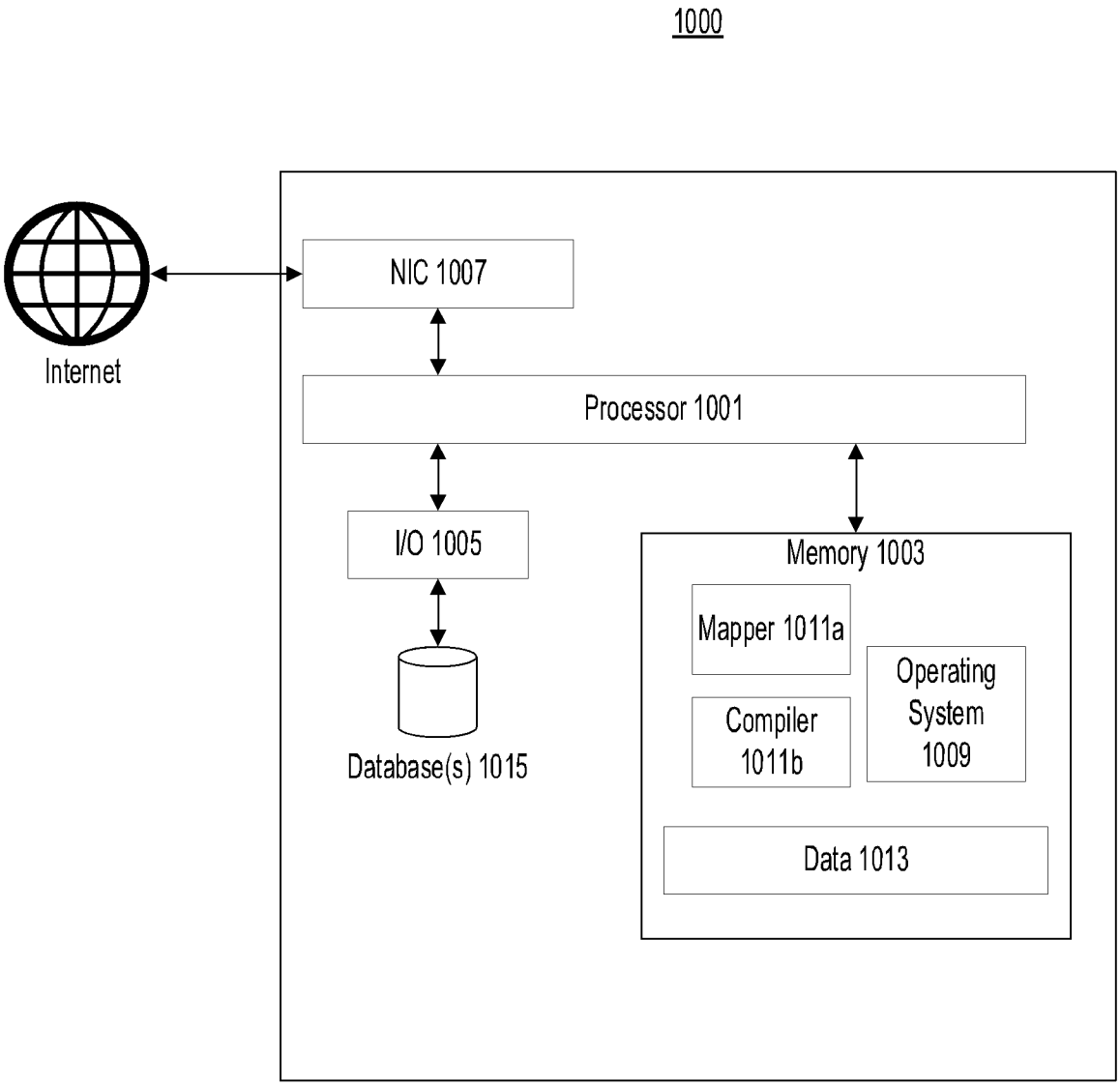
1000
Internet
NIC 1007
Processor 1001
I/O 1005
Database(s) 1015
Memory 1003
Mapper 1011a
Compiler 1011b
Operating System 1009
Data 1013

FIG. 10

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2018/125541

A. CLASSIFICATION OF SUBJECT MATTER

G05B 19/05(2006.01)i; G06F 16/90(2019.01)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;G05B

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,SIPOABS,DWPI,CNKI:database;scan;programmable;logic;instruction;configuration;query

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 2017027179 A1 (SIEMENS AKTIENGESELLSCHAFT) 16 February 2017 (2017-02-16) description, paragraphes 19-52	1-31
A	US 2014067845 A1 (INTERNATIONAL BUSINESS MACHINES CORPORATION) 06 March 2014 (2014-03-06) the whole document	1-31
A	US 2008091646 A1 (HEWLETT PACKARD COMPANY) 17 April 2008 (2008-04-17) the whole document	1-31

☐ 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
19 September 2019	**26 September 2019**

Name and mailing address of the ISA/CN	Authorized officer
National Intellectual Property Administration, PRC **6, Xitucheng Rd., Jimen Bridge, Haidian District, Beijing 100088** **China**	**ZHANG,Yan**
Facsimile No. **(86-10)62019451**	Telephone No. **86-(10)-62412178**

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2018/125541

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
WO 2017027179 A1	16 February 2017	US 20171038754 A1	09 February 2017
		EP 3332296 A1	13 June 2018
		CN 107850882 B	21 June 2019
		CN 107850882 A	27 March 2018
		BR 112018001045 A2	11 September 2018
		RU 2688451 C1	21 May 2019
US 2014067845 A1	06 March 2014	US 8977637 B2	10 March 2015
US 2008091646 A1	17 April 2008	US 7743053 B2	22 June 2010