



(51) International Patent Classification:
G06F 9/30 (2018.01)

(21) International Application Number:
PCT/US2022/018060

(22) International Filing Date:
27 February 2022 (27.02.2022)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
17/198,028 10 March 2021 (10.03.2021) US

(71) Applicant: **META PLATFORMS, INC.** [US/US]; c/o Rim-
on Law/Steven Colby, 420 West Main Street, Suite 101B,
Boise, Idaho 83702 (US).

(72) Inventors: **REDDY, Harikrishna Madadi**; 1601 Willow
Road, Menlo Park, California 94025 (US). **ALAPARTHI,
Srikanth**; 1601 Willow Road, Menlo Park, California
94025 (US). **JOHANSSON, Olof**; 1601 Willow Road,
Menlo Park, California 94025 (US). **MASTRO, Alexan-
der**; 1601 Willow Road, Menlo Park, California 94025
(US).

(74) Agent: **COLBY, Steven** et al.; Rimon Law, 420 West Main
Street, Suite 101B, Boise, Idaho 83702 (US).

(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, IT, JM, JO, JP, KE, KG, KH,
KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA,
MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI,
NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU,
RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM,
TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM,
ZW.

(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).

(54) Title: APPARATUS, SYSTEMS, AND METHODS FOR FACILITATING EFFICIENT HARDWARE-FIRMWARE INTER-ACTIONS

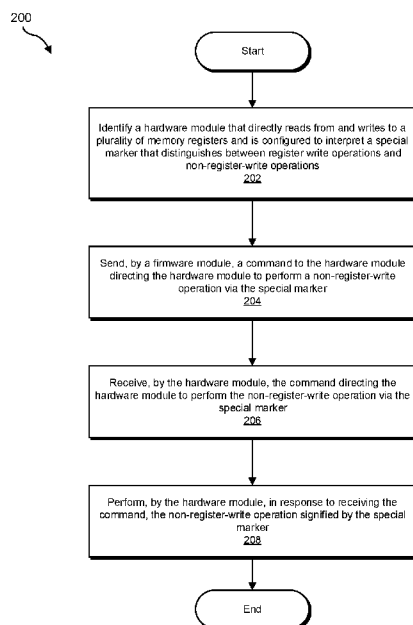


FIG. 2

(57) Abstract: A system for facilitating efficient hardware-firmware interactions may include (i) a plurality of memory registers, (ii) a hardware module that directly reads from and writes to the plurality of memory registers and is configured to interpret a special marker that distinguishes between register write operations and non-register-write operations, and (iii) a firmware module that directs the hardware module to perform operations at least in part by sending the special marker. Various other methods, systems, and computer-readable media are also disclosed.

Published:

- *with international search report (Art. 21(3))*
- *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

1 APPARATUS, SYSTEMS, AND METHODS FOR FACILITATING EFFICIENT

2 HARDWARE-FIRMWARE INTERACTIONS

3 TECHNICAL FIELD

4 [0001] The present disclosure is generally directed to systems and methods for
5 facilitating efficient hardware-firmware interactions.

6 BACKGROUND OF THE DISCLOSURE

7 [0002] In many firmware (FW) controlled system designs, FW prepares the
8 programming sequence and programs hardware (HW) in order to achieve a specific
9 functionality. Preparing this sequence, programming the sequence to HW, waiting for HW
10 completion, and monitoring the HW state for any additional information may involve context
11 switching in FW and cause high latency in the processing time.

12 SUMMARY OF THE DISCLOSURE

13 [0003] In accordance with a first aspect of the present disclosure, there is provided a
14 system comprising: a plurality of memory registers; a hardware module that: directly reads
15 from and writes to the plurality of memory registers; and is configured to interpret a special
16 marker that distinguishes between register write operations and non-register-write
17 operations; and a firmware module that directs the hardware module to perform operations
18 at least in part by sending the special marker.

19 [0004] In some embodiments, the non-register-write operations may comprise at
20 least one of: a register read operation; a wait-for-done operation; or a debug operation.

21 [0005] In some embodiments, the special marker may comprise an address of a
22 predefined special memory register and an operation code.

23 [0006] In some embodiments, the firmware module may prepare a list of commands
24 stored in memory; the firmware module provides at least one address pointer and size for
25 the list of commands to the hardware module; and the hardware module fetches the list of
26 commands via the at least one address pointer and size.

27 [0007] In some embodiments, the firmware module may provide, to the hardware
28 module, a plurality of address pointers that each point to a different segment of a single
29 command in the list of commands.

30 [0008] In some embodiments, the hardware module may store the at least one
31 address pointer to a memory register within the plurality of memory registers.

32 [0009] In some embodiments, the firmware module may provide the at least one

1 address pointer to the hardware module repeatedly during different points in time.

2 **[0010]** In some embodiments, the hardware module may receive a command to
3 perform a wait-for-done operation; the hardware module pauses operating until detecting
4 that a hardware thread has completed; and the hardware module resumes operating in
5 response to detecting that the hardware thread has completed.

6 **[0011]** In some embodiments, the command to perform the wait-for-done operation
7 may comprise a sequence identifier; and the hardware module facilitates cross-thread
8 dependency by pausing operating until detecting that the hardware thread specified by the
9 sequence identifier has completed.

10 **[0012]** In some embodiments, the hardware module may receive a command to
11 perform a terminate operation; and in response, the hardware module: pauses operating
12 until detecting that at least one hardware thread has completed; drains prefetched data;
13 empties a command queue; and confirms a completion of the terminate operation to the
14 firmware module.

15 **[0013]** In some embodiments, the hardware module may receive a command from
16 the firmware to perform a debug operation; and in response, the hardware module writes
17 data to memory that is accessible to the firmware.

18 **[0014]** In some embodiments, the hardware module may store a timeout value that,
19 when reached, prompts the hardware module to: pause operating; and send a timeout
20 message to the firmware module.

21 **[0015]** In some embodiments, the hardware module may store, in at least one
22 memory register within the plurality of memory registers, a current status of the hardware
23 module.

24 **[0016]** In accordance with a further aspect of the present disclosure, there is provided
25 a computer-implemented method comprising: identifying a hardware module that: directly
26 reads from and writes to a plurality of memory registers; and is configured to interpret a
27 special marker that distinguishes between register write operations and non-register-write
28 operations; sending, by a firmware module, a command to the hardware module directing
29 the hardware module to perform a non-register-write operation via the special marker;
30 receiving, by the hardware module, the command directing the hardware module to perform
31 the non-register-write operation via the special marker; and performing, by the hardware
32 module, in response to receiving the command, the non-register-write operation signified by

1 the special marker.

2 **[0017]** In some embodiments, the non-register-write operation may comprise a
3 register read operation; and the hardware module performs the register read operation by
4 reading data from a memory register within the plurality of memory registers.

5 **[0018]** In some embodiments, the non-register-write operation may comprise a wait-
6 for-done operation; the hardware module performs the wait-for-done operation by pausing
7 operating until the hardware module detects that a hardware thread has completed; and the
8 hardware module resumes operating in response to detecting that the hardware thread has
9 completed.

10 **[0019]** In some embodiments, the command to perform the wait-for-done operation
11 may comprise a sequence identifier; and the hardware module facilitates cross-thread
12 dependency by pausing operating until detecting that the hardware thread specified by the
13 sequence identifier has completed.

14 **[0020]** In some embodiments, the non-register-write operation may comprise a
15 debug operation; and the hardware module performs the debug operation by writing data to
16 memory that is accessible to the firmware.

17 **[0021]** In some embodiments, the non-register-write operation may comprise a
18 terminate operation; and the hardware module performs the terminate operation by:
19 pausing operating until detecting that at least one hardware thread has completed; draining
20 prefetched data; emptying a command queue; and confirming a completion of the terminate
21 operation to the firmware module.

22 **[0022]** In accordance with a further aspect of the present disclosure, there is provided
23 an apparatus comprising: a plurality of memory registers; a hardware module that: directly
24 reads from and writes to the plurality of memory registers; and is configured to interpret a
25 special marker that distinguishes between register write operations and non-register-write
26 operations; and a hardware element configured to execute a firmware module that directs
27 the hardware module to perform operations at least in part by sending the special marker.

28 **[0023]** It will be appreciated that any features described herein as being suitable for
29 incorporation into one or more aspects or embodiments of the present disclosure are
30 intended to be generalizable across any and all aspects and embodiments of the present
31 disclosure. Other aspects of the present disclosure can be understood by those skilled in the
32 art in light of the description, the claims, and the drawings of the present disclosure. The

foregoing general description and the following detailed description are exemplary and explanatory only and are not restrictive of the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0024] The accompanying drawings illustrate a number of exemplary embodiments and are a part of the specification. Together with the following description, these drawings demonstrate and explain various principles of the instant disclosure.

[0025] FIG. 1 is a block diagram of an exemplary system for facilitating efficient hardware-firmware interactions.

[0026] FIG. 2 is a flow diagram of an exemplary method for facilitating efficient hardware-firmware interactions.

[0027] FIG. 3 is a block diagram of a command direct memory access module.

[0028] FIG. 4 is a block diagram of a read engine for a command direct memory access module.

[0029] FIG. 5 is a block diagram of a write engine for a command direct memory access module.

[0030] FIG. 6 is a block diagram of a read engine for a command direct memory access module.

[0031] FIG. 7 is a block diagram of a command direct memory access module performing multithreaded operations.

[0032] FIG. 8 is an additional block diagram of a command direct memory access module performing multithreaded operations.

[0033] FIG. 9 is a flow diagram of a method for a command direct memory access module to execute a terminate command.

[0034] Throughout the drawings, identical reference characters and descriptions indicate similar, but not necessarily identical, elements. While the exemplary embodiments described herein are susceptible to various modifications and alternative forms, specific embodiments have been shown by way of example in the drawings and will be described in detail herein. However, the exemplary embodiments described herein are not intended to be limited to the particular forms disclosed. Rather, the instant disclosure covers all modifications, equivalents, and alternatives falling within the scope of the appended claims.

[0035] Features from any of the embodiments described herein may be used in combination with one another in accordance with the general principles described herein.

1 These and other embodiments, features, and advantages will be more fully understood upon
2 reading the following detailed description in conjunction with the accompanying drawings
3 and claims.

4 DETAILED DESCRIPTION OF EXEMPLARY EMBODIMENTS

5 **[0036]** In many firmware (FW) controlled system designs, FW prepares the
6 programming sequence and programs hardware (HW) in order to achieve a specific
7 functionality. Preparing this sequence, programming the sequence to HW, waiting for HW
8 completion, and monitoring the HW state for any additional information may involve context
9 switching in FW and cause high latency in the processing time. The latency may become
10 critical and significant in throughput driven designs where multiple HW threads work in
11 pipeline fashion to achieve a common task, such as a transcoder that decodes a video
12 sequence of a particular format and encodes the video sequence in different formats and
13 resolutions.

14 **[0037]** The present disclosure is generally directed to systems and methods for facilitating
15 efficient hardware-firmware interactions. In order to minimize FW context switching and
16 latency in processing times, the systems described herein offload some of this from FW and
17 implement some of the programming features in HW. In one embodiment, a new HW module,
18 called command direct memory access (CDMA), may be added in the transcoder solution or
19 other hardware configuration. In some examples, a CDMA may support a pointer-to-pointer
20 scheme for basic register programming, a special marker that enables the HW to distinguish
21 between register write operations and special operations (e.g., read, wait, etc.), a wait-for-
22 done command, and/or debug and performance traces. This may enable FW to use dedicated
23 buffers for a programming sequence that is common across frames for a given HW thread. In
24 some embodiments, this system may minimize FW buffer updates (or writes) and/or save
25 command list preparation time.

26 **[0038]** In some embodiments, the systems described herein may improve the functioning
27 of a computing device by increasing the speed at which the computing device performs
28 operations. Additionally, the systems described herein may improve the fields of
29 computational efficiency and/or video transcoding by improving the efficiency at which
30 computing devices can execute certain command sequences, such as the command
31 sequences used in video transcoding.

32 **[0039]** In some embodiments, the systems described herein may facilitate efficient

1 hardware-firmware interaction. FIG. 1 is a block diagram of an exemplary system 100 for
2 facilitating efficient hardware-firmware interaction. In one embodiment, and as will be
3 described in greater detail below, a computing device 102 may be configured with memory
4 registers 104(1) through 104(n). In some embodiments, a hardware module 106 may perform
5 read and/or write operations on memory registers 104(1) through 104(n). In one example,
6 hardware module 106 may be a CDMA that is configured to interpret a special marker that
7 distinguishes between register write operations and non-register-write operations. In one
8 embodiment, computing device 102 may include a firmware module 108 that directs
9 hardware module 106 to perform operations at least in part by sending the special marker. In
10 some embodiments, computing device 102 may include a hardware element configured to
11 execute firmware module 108. Computing device 102 may represent various types of
12 computing devices including but not limited to personal computing devices (e.g., laptops,
13 desktops, smart phones, etc.), servers, embedded computing devices, and/or smart devices.

14 **[0040]** FIG. 2 is a flow diagram of an exemplary method 200 for facilitating hardware-
15 firmware interactions. In some examples, at step 202, the systems described herein may
16 identify a HW module that directly reads from and writes to a plurality of memory registers
17 and is configured to interpret a special marker that distinguishes between register write
18 operations and non-register-write operations. The term “special marker” may generally refer
19 to any string embedded in a message and/or any formatting of a message that is interpreted
20 by a specially configured hardware module (e.g., a CDMA) as a command other than the
21 default command performed by the hardware module (e.g., register write). The special
22 marker may take a variety of forms. For example, the systems described herein may designate
23 a specific register in CDMA control/status register space and use the address of the
24 designated register as the special marker. In some examples, the systems described herein
25 may define specific operation codes (opcodes) that each designate a specific operation, such
26 as wait-for-done, terminate, debug, and so forth. In one embodiment, a 32-bit address field
27 may be split into 28 bits for the address of the designated register and four bits for opcodes.
28 The systems described herein may perform step 202 in a variety of ways. In one example, the
29 systems described herein may identify a CDMA. The term “CDMA” generally refers to any
30 hardware module that is capable of perform register read and write operations and that is
31 configured to interpret a special marker. In some embodiments, a CDMA may manage
32 multiple hardware threads.

1 **[0041]** At step 204, the systems described herein may send, by a FW module, a
2 command to the HW module directing the HW module to perform a non-register-write
3 operation via the special marker. The term “non-register-write operation” may generally refer
4 to any operation performed by hardware that does not exclusively consist of writing data to
5 a memory register. For example, a non-register-write operation may include a register read
6 operation, a wait-for-done operation, a terminate operation, and/or a debug operation. The
7 systems described herein may perform step 204 in a variety of ways. In one example, FW may
8 send a wait-for-done command to the CDMA. In another example, FW may send a terminate
9 command to the CDMA.

10 **[0042]** At step 206, the systems described herein may receive, by the HW module, the
11 command directing the HW module to perform the non-register-write operation via the
12 special marker. The systems described herein may perform step 206 in a variety of ways. For
13 example, the CDMA may read the command from a command queue. In some embodiments,
14 the CDMA may check a designated section of memory for commands from FW.

15 **[0043]** At step 208, the systems describe herein may perform, by the HW module, in
16 response to receiving the command, the non-register-write operation signified by the special
17 marker. For example, the CDMA may read data, wait for a thread to complete, and/or
18 terminate operations. In one example, the CDMA may send debug data to FW. For example,
19 upon receiving a debug command signified by a debug opcode in the special marker, the
20 CDMA may output debugging information into external memory that can be used by FW for
21 performance monitoring, analysis, and/or debugging processes. In another example, the
22 CDMA may receive a wait-for-done command and in response, the CDMA may pause
23 operating until detecting that a hardware thread specified by the wait-for-done command has
24 completed.

25 **[0044]** FIG. 3 is a block diagram of an example CDMA 302. In one embodiment, CDMA
26 302 may include a read/write channel 308 and/or a read/write channel 310. In some
27 examples, read/write channel 308 may use an advanced extensible interface (AXI)
28 communication interface to read data from external memory and/or write data to external
29 memory. In one example, read/write channel 310 may use an advanced microcontroller bus
30 (AHB) interface to read data from memory registers (e.g., the status register) and/or write
31 data to memory registers. In other embodiments, the systems described herein may use other
32 hardware architecture, interfaces, and/or protocols for write channel 308 and/or read

channel 310. In some embodiments, write channel 308 and/or read channel 310 may be capable of executing on multiple threads simultaneously and arbiters 312 and/or 314 may allocate access to write channel 308 and/or read channel 310, respectively. For example, CDMA 302 may be configured to execute CDMA threads 304(1) through 304(n). In one embodiment, CDMA 302 may be configured to execute twelve threads. For example, CDMA 302 may be functioning as a video transcoder and may have twelve threads that each correspond to a different format and/or a different stage of the transcoding process. In some embodiments, CDMA 302 may periodically update a control/status register (CSR) 306 with the current status of CDMA 302. In some embodiments, CDMA 302 may maintain internally the position within each buffer and provide that information at CSR 306. In one example, data in CSR 306 may indicate the address of the current command being executed and/or a pointer in CSR 306 may indicate the position within the current programming sequence buffer. Additionally or alternatively, CDMA 302 may receive instructions from FW via one or more designated registers within CSR 306.

[0045] In some embodiments, the systems described herein may support a programming sequence of a thread that is split across multiple physical buffers in memory. For example, as illustrated in FIG. 4, a CDMA may access memory 402 based on a command queue 404 that issues commands to a CDMA thread 406. Memory 402 may represent various types of memory, including but not limited to double data rate synchronous dynamic random-access memory (DDR SDRAM) and/or any other suitable type of random-access memory. In this example, memory 402 may include three different buffers that are prepared by FW for access by one or more CDMA threads. In some embodiments, this may enable FW to store all the common programming sequences in one buffer to use these sequences across frames for a given thread as well as across threads. For example, the frame width and height for a given video sequence may be unchanged across multiple stages of transcoding. Likewise, the debug programming sequence and/or reset/clear mechanisms may be constant across frames. For all such sequences, FW may store each sequence in one dedicated buffer. By storing reused sequences in buffers in memory, the systems described herein may prevent FW from having to re-program the buffers repeatedly.

[0046] In one embodiment, FW may provide an address pointer and size for the list of commands stored in command queue 404 and the CDMA may fetch the list of commands via the address pointer and size. In some embodiments, the FW may provide the address pointer

1 and size repeatedly, as the buffer may include instructions that are referenced repeatedly,
2 such as the debug programming sequence, clock sequence, reset sequence, and/or interrupt
3 clear sequence.

4 **[0047]** In some examples, once the buffers are ready, FW may provide all the pointers
5 to the CDMA through a CSR. In one example, there may be multiple buffers to process,
6 preventing the CDMA from having explicit information about when to send a CDMA interrupt
7 to FW. In one embodiment, the CDMA may provide control to FW to push an enable-interrupt
8 command into the command queue. When this command is received, the CDMA may
9 generate an interrupt after the processing of the corresponding buffer.

10 **[0048]** FIG. 5 is a block diagram of an example write engine for a CDMA. In some
11 embodiments, the write engine may only be active if the CDMA is executing register read
12 operations. In one example, the CDMA may issue the read request to memory registers
13 through AHB to collect data, combine the {address, data} pairs to match bus width, then write
14 the data out in bursts to DDR through AXI. In one embodiment, a CDMA 504 may write to a
15 memory 502. In some examples, CDMA 504 may write data in a variety of formats. For
16 example, CDMA 504 may be configured to transcode videos into different formats. In these
17 examples, CDMA 504 may include write engines 506(1) through 506(n) that each correspond
18 to a format from formats 512(1) through 512(n). In some embodiments, write engines 506(1)
19 through 506(n) may have resources allocated by an arbiter 510 that controls access to direct
20 memory access 508. In one embodiment, CDMA 504 may use an AHB interface to read the
21 corresponding registers and then write the address and data pair to memory 502 via an AXI
22 interface. In some examples, FW may provide one write address per thread for a given CDMA
23 session and CDMA 504 may continue to write the data in that location. In some embodiments,
24 each CDMA thread may have one associated write address for FW to program. In some
25 examples, once CDMA 504 receives an opcode indicating that the last data has been received,
26 CDMA 504 may flush out any partial data to memory 502 and/or returns write-done interrupt
27 to FW.

28 **[0049]** FIG. 6 is a block diagram of a read engine for a CDMA. In one embodiment, a
29 CDMA 608 may read from a read buffer 604 in memory 602 and/or write to a write buffer 606
30 in memory 602. In some embodiments, CDMA 608 may read from and/or write to a local
31 CDMA buffer 610. For example, CDMA 608 may read and/or process a first set of data at a
32 time 612, a second set of data at a time 614, and/or a third set of data at a time 616. Once

1 local CDMA buffer 610 is full and/or CDMA 608 has received a special marker instructing
2 CDMA 608 to write all data, CDMA 608 may write the data in local CDMA buffer 610 to write
3 buffer 606 in memory 602. In one example, FW may read three registers after each wait-for-
4 done instruction and may finish the session after three frames. In this example, there may be
5 a total of nine register reads and the amount of data written to memory 602 may be 72 bytes.

6 **[0050]** In some embodiments, the systems described herein may use a sequence
7 identifier (ID) inserted in the special marker to facilitate cross thread dependency and/or
8 efficiency within a single thread. In some examples, a sequence ID may be represented as a
9 continuously incrementing eight bit value. For example, as illustrated in FIG. 7, FW may
10 prepare a scalar 704 (e.g., an Xcoder scalar) for processing three frames through a CDMA with
11 a wait-for-done marker between frames. In one example, the scalar thread ID may be three,
12 so the CDMA may use CDMA thread three. In this example, once all the frames are completed,
13 the CDMA may send the final interrupt to FW indicating that processing is done. In some
14 embodiments, each sequence identifier may be incremented for each done command so that
15 other threads may check the sequence identifier to determine the status of the thread. For
16 example, sequence ID 714 for scalar 704 may be initially set to one, increment to two, and
17 then increment to three as each frame is completed. In this example, FW may receive only
18 one interrupt for three frames processed using a CDMA system, reducing the latency
19 compared systems without a CDMA where FW may receive three interrupts, one after each
20 scalar frame is processed.

21 **[0051]** In some examples, the main challenge of cross thread dependency modeling
22 may be the variable processing times of each thread. Some of the threads may finish faster
23 than others, making the synchronization difficult. In order to solve this problem, the systems
24 described herein may use a sequence ID. For example, the systems described herein may
25 store the sequence ID for each scalar in the corresponding CDMA thread. In this example,
26 when a dependent thread is waiting, the thread may compare the thread's own wait-
27 sequence-ID against the stored value from the master thread and may proceed as long as the
28 wait-sequence-ID is greater than or equal to the stored sequence ID.

29 **[0052]** In one example, as illustrated in FIG. 8, a CDMA may handle encode (ENC), bit
30 stream (BS), and/or quality metrics (QM) dependent threads. Each done signal from an ENC
31 thread may trigger the processing of the frame by the BS and QM. For example, at time 812,
32 ENC 802 may finish processing a frame and update a sequence ID 808. Based on this update,

1 at time 814, BS 804 and/or QM 806 may begin processing that frame. Meanwhile, ENC 802
2 may begin processing a new frame. When ENC 802 finishes processing the new frame and
3 updates sequence ID 808 again, at time 816, BS 804 and/or QM 806 may process that frame
4 while ENC 802 moves on to a new frame. In one embodiment, BS 804 may only start
5 processing a new frame if two conditions are met: BS 804 has finished processing the previous
6 frame and ENC 802 has finished processing the new frame. In some examples, the systems
7 described herein may check the current sequence ID for both BS 804 and ENC 802 to
8 determine whether BS 804 is ready to begin processing a new frame.

9 **[0053]** In one example, when the CDMA is processing BS 804, the CDMA may identify
10 the wait-for-done marker for ENC 802. The CDMA may internally compare the stored value
11 from ENC 802 to check if it is greater than or equal to the wait-for-done marker and may wait
12 until that condition is met before programming BS 804. In some examples, ENC 802 may not
13 have to wait at each done message for the done to be sampled by all dependent threads. In
14 this example, each thread with variable processing times may not impact other threads.

15 **[0054]** In some embodiments, a CDMA may terminate processing when certain
16 conditions are met. For example, the CDMA may receive a terminate command from FW. FIG.
17 9 is a flow diagram of an example method for a CDMA to execute a terminate command. After
18 receiving a terminate command, at step 902, the systems described herein may read the
19 CDMA queue status. If the queue is empty, at step 912, the systems described herein may
20 send a new command or end processing. If the queue is not empty, at step 906, the systems
21 described herein may set a terminate bit to equal one. If the terminate is not complete, the
22 systems described herein may wait. For example, the systems described herein may wait for
23 one or more hardware threads to complete. In some embodiments, the systems described
24 herein may drain prefetched data and/or empty the queue. If the terminate is complete, the
25 systems described herein may, at step 910, set the terminate bit equal to zero. In some
26 embodiments, the CDMA may send a message to FW confirming completion of the terminate
27 operation. The systems described herein may then proceed to step 912 and send a new
28 command or end processing.

29 **[0055]** In some embodiments, a CDMA may timeout under certain conditions. In some
30 examples, between the passes or frames, a CDMA may wait for completion from the
31 corresponding HW thread. In order to recover from any hang scenarios, FW may enable
32 timeout behavior and program a timeout value. Upon reaching the timeout value (e.g.,

1 waiting for a hardware thread for an amount of seconds, milliseconds, or other measurement
2 of time that matches the timeout value), a CDMA thread may generate a timeout message to
3 send to FW and wait in the same state until receiving a message from FW. In some examples,
4 FW may continue to wait after receiving the timeout message or may issue a terminate
5 command to the CDMA.

6 **[0056]** As described above, the systems and methods described herein may improve the
7 efficiency of various computing processes, such as video transcoding, by using a special
8 marker to communicate with a CDMA that receives commands from FW and reads and writes
9 to registers. By storing repeatedly accessed information and command sequences in buffers
10 in memory that can be read by the CDMA via the command queue, the systems described
11 herein may eliminate redundant iterations of programming that same information into
12 buffers by FW in between different sequences. The systems described herein may direct the
13 CDMA via a special marker with different opcodes for different operations, such as debug,
14 terminate, and wait-for-done. Using a wait-for-done command with a sequence ID that
15 specifies a thread may enable the systems described herein to facilitate cross-thread
16 dependency by maintaining and transmitting information about the current status of each
17 thread, enabling threads to wait only for relevant other threads to finish processing rather
18 than having to wait for all threads.

19 **[0057] Example Embodiments**

20 **[0058]** Example 1: A system for facilitating efficient hardware-firmware interactions
21 may include (i) a group of memory registers, (ii) a hardware module that directly reads from
22 and writes to the memory registers and is configured to interpret a special marker that
23 distinguishes between register write operations and non-register-write operations, and (iii) a
24 firmware module that directs the hardware module to perform operations at least in part by
25 sending the special marker.

26 **[0059]** Example 2: The system of example 1, where the non-register-write operations
27 include at least one of a register read operation, a wait-for-done operation, and/or a debug
28 operation.

29 **[0060]** Example 3: The system of examples 1-2 may further include an address of a
30 predefined special memory register and an operation code.

31 **[0061]** Example 4: The system of examples 1-3, where the firmware module prepares
32 a list of commands stored in memory, the firmware module provides at least one address

1 pointer and size for the list of commands to the hardware module, and the hardware module
2 fetches the list of commands via the at least one address pointer and size.

3 **[0062]** Example 5: The system of examples 1-4, where the firmware module provides,
4 to the hardware module, a plurality of address pointers that each point to a different segment
5 of a single command in the list of commands.

6 **[0063]** Example 6: The system of examples 1-5, where the hardware module stores
7 the at least one address pointer to a memory register within the plurality of memory registers.

8 **[0064]** Example 7: The system of examples 1-6, where the firmware module provides
9 the at least one address pointer to the hardware module repeatedly during different points
10 in time.

11 **[0065]** Example 8: The system of examples 1-7, where the hardware module receives
12 a command to perform a wait-for-done operation, the hardware module pauses operating
13 until detecting that a hardware thread has completed, and the hardware module resumes
14 operating in response to detecting that the hardware thread has completed.

15 **[0066]** Example 9: The system of examples 1-8, where the command to perform the
16 wait-for-done operation includes a sequence identifier and the hardware module facilitates
17 cross-thread dependency by pausing operating until detecting that the hardware thread
18 specified by the sequence identifier has completed.

19 **[0067]** Example 10: The system of examples 1-9, where the hardware module receives
20 a command to perform a terminate operation and, in response, the hardware module pauses
21 operating until detecting that at least one hardware thread has completed, drains prefetched
22 data, empties a command queue, and confirms a completion of the terminate operation to
23 the firmware module.

24 **[0068]** Example 11: The system of examples 1-10, where the hardware module
25 receives a command from the firmware to perform a debug operation and in response, the
26 hardware module writes data to memory that is accessible to the firmware.

27 **[0069]** Example 12: The system of examples 1-11, where the hardware module stores
28 a timeout value that, when reached, prompts the hardware module to pause operating and
29 send a timeout message to the firmware module.

30 **[0070]** Example 13: The system of examples 1-12, where the hardware module stores,
31 in at least one memory register within the plurality of memory registers, a current status of
32 the hardware module.

1 **[0071]** Example 14: A computer-implemented method for facilitating efficient
2 hardware-firmware interactions may include (i) identifying a hardware module that directly
3 reads from and writes to a plurality of memory registers and is configured to interpret a
4 special marker that distinguishes between register write operations and non-register-write
5 operations, (ii) sending, by a firmware module, a command to the hardware module directing
6 the hardware module to perform a non-register-write operation via the special marker, (iii)
7 receiving, by the hardware module, the command directing the hardware module to perform
8 the non-register-write operation via the special marker, and (iv) performing, by the hardware
9 module, in response to receiving the command, the non-register-write operation signified by
10 the special marker.

11 **[0072]** Example 15: The computer-implemented method of example 14, where the
12 non-register-write operation includes a register read operation and the hardware module
13 performs the register read operation by reading data from a memory register within the
14 plurality of memory registers.

15 **[0073]** Example 16: The computer-implemented method of examples 14-15, where (i)
16 the non-register-write operation includes a wait-for-done operation, (ii) the hardware
17 module performs the wait-for-done operation by pausing operating until the hardware
18 module detects that a hardware thread has completed, and (iii) the hardware module
19 resumes operating in response to detecting that the hardware thread has completed.

20 **[0074]** Example 17: The computer-implemented method of examples 14-16, where
21 the computer-executable instructions cause the physical processor to the command to
22 perform the wait-for-done operation includes a sequence identifier and the hardware module
23 facilitates cross-thread dependency by pausing operating until detecting that the hardware
24 thread specified by the sequence identifier has completed.

25 **[0075]** Example 18: The computer-implemented method of examples 14-17, where
26 the non-register-write operation includes a debug operation and the hardware module
27 performs the debug operation by writing data to memory that is accessible to the firmware.

28 **[0076]** Example 19: The computer-implemented method of examples 14-18, where
29 the non-register-write operation includes a terminate operation and the hardware module
30 performs the terminate operation by (i) pausing operating until detecting that at least one
31 hardware thread has completed, (ii) draining prefetched data, (iii) emptying a command
32 queue, and (iv) confirming a completion of the terminate operation to the firmware module.

1 **[0077]** Example 20: An apparatus may include (i) a plurality of memory registers, (ii) a
2 hardware module that directly reads from and writes to the plurality of memory registers and
3 is configured to interpret a special marker that distinguishes between register write
4 operations and non-register-write operations, and (iii) a hardware element configured to
5 execute a firmware module that directs the hardware module to perform operations at least
6 in part by sending the special marker.

7 **[0078]** As detailed above, the computing devices and systems described and/or illustrated
8 herein broadly represent any type or form of computing device or system capable of
9 executing computer-readable instructions, such as those contained within the modules
10 described herein. In their most basic configuration, these computing device(s) may each
11 include at least one memory device and at least one physical processor.

12 **[0079]** In some examples, the term “memory device” generally refers to any type or form
13 of volatile or non-volatile storage device or medium capable of storing data and/or computer-
14 readable instructions. In one example, a memory device may store, load, and/or maintain one
15 or more of the modules described herein. Examples of memory devices include, without
16 limitation, Random Access Memory (RAM), Read Only Memory (ROM), flash memory, Hard
17 Disk Drives (HDDs), Solid-State Drives (SSDs), optical disk drives, caches, variations or
18 combinations of one or more of the same, or any other suitable storage memory.

19 **[0080]** In some examples, the term “physical processor” generally refers to any type or
20 form of hardware-implemented processing unit capable of interpreting and/or executing
21 computer-readable instructions. In one example, a physical processor may access and/or
22 modify one or more modules stored in the above-described memory device. Examples of
23 physical processors include, without limitation, microprocessors, microcontrollers, Central
24 Processing Units (CPUs), Field-Programmable Gate Arrays (FPGAs) that implement softcore
25 processors, Application-Specific Integrated Circuits (ASICs), portions of one or more of the
26 same, variations or combinations of one or more of the same, or any other suitable physical
27 processor.

28 **[0081]** Although illustrated as separate elements, the modules described and/or
29 illustrated herein may represent portions of a single module or application. In addition, in
30 certain embodiments one or more of these modules may represent one or more software
31 applications or programs that, when executed by a computing device, may cause the
32 computing device to perform one or more tasks. For example, one or more of the modules

1 described and/or illustrated herein may represent modules stored and configured to run on
2 one or more of the computing devices or systems described and/or illustrated herein. One or
3 more of these modules may also represent all or portions of one or more special-purpose
4 computers configured to perform one or more tasks.

5 **[0082]** In addition, one or more of the modules described herein may transform data,
6 physical devices, and/or representations of physical devices from one form to another. For
7 example, one or more of the modules recited herein may receive image data to be
8 transformed, transform the image data into a data structure that stores user characteristic
9 data, output a result of the transformation to select a customized interactive ice breaker
10 widget relevant to the user, use the result of the transformation to present the widget to the
11 user, and store the result of the transformation to create a record of the presented widget.
12 Additionally or alternatively, one or more of the modules recited herein may transform a
13 processor, volatile memory, non-volatile memory, and/or any other portion of a physical
14 computing device from one form to another by executing on the computing device, storing
15 data on the computing device, and/or otherwise interacting with the computing device.

16 **[0083]** In some embodiments, the term “computer-readable medium” generally
17 refers to any form of device, carrier, or medium capable of storing or carrying computer-
18 readable instructions. Examples of computer-readable media include, without limitation,
19 transmission-type media, such as carrier waves, and non-transitory-type media, such as
20 magnetic-storage media (e.g., hard disk drives, tape drives, and floppy disks), optical-storage
21 media (e.g., Compact Disks (CDs), Digital Video Disks (DVDs), and BLU-RAY disks), electronic-
22 storage media (e.g., solid-state drives and flash media), and other distribution systems.

23 **[0084]** The process parameters and sequence of the steps described and/or
24 illustrated herein are given by way of example only and can be varied as desired. For example,
25 while the steps illustrated and/or described herein may be shown or discussed in a particular
26 order, these steps do not necessarily need to be performed in the order illustrated or
27 discussed. The various exemplary methods described and/or illustrated herein may also omit
28 one or more of the steps described or illustrated herein or include additional steps in addition
29 to those disclosed.

30 **[0085]** The preceding description has been provided to enable others skilled in the art
31 to best utilize various aspects of the exemplary embodiments disclosed herein. This
32 exemplary description is not intended to be exhaustive or to be limited to any precise form

1 disclosed. Many modifications and variations are possible without departing from the scope
2 of the claims. The embodiments disclosed herein should be considered in all respects
3 illustrative and not restrictive. Reference should be made to the appended claims and their
4 equivalents in determining the scope of the instant disclosure.

5 **[0086]** Unless otherwise noted, the terms “connected to” and “coupled to” (and their
6 derivatives), as used in the specification and claims, are to be construed as permitting both
7 direct and indirect (i.e., via other elements or components) connection. In addition, the terms
8 “a” or “an,” as used in the specification and claims, are to be construed as meaning “at least
9 one of.” Finally, for ease of use, the terms “including” and “having” (and their derivatives), as
10 used in the specification and claims, are interchangeable with and have the same meaning as
11 the word “comprising.”
12

1 CLAIMS:

2 1. A system comprising:

3 a plurality of memory registers;

4 a hardware module that:

5 directly reads from and writes to the plurality of memory registers; and

6 is configured to interpret a special marker that distinguishes between register

7 write operations and non-register-write operations; and

8 a firmware module that directs the hardware module to perform operations at least

9 in part by sending the special marker.

10 2. The system of claim 1, where the non-register-write operations comprise at

11 least one of:

12 a register read operation;

13 a wait-for-done operation; or

14 a debug operation.

15 3. The system of claim 1 or claim 2, wherein the special marker comprises an

16 address of a predefined special memory register and an operation code.

17 4. The system of claim 1, claim 2 or claim 3, wherein:

18 the firmware module prepares a list of commands stored in memory;

19 the firmware module provides at least one address pointer and size for the list of

20 commands to the hardware module; and

21 the hardware module fetches the list of commands via the at least one address pointer

22 and size.

23 5. The system of claim 4, wherein one or more of:

24 i. the firmware module provides, to the hardware module, a plurality of address
25 pointers that each point to a different segment of a single command in the list of commands;26 ii. the hardware module stores the at least one address pointer to a memory register
27 within the plurality of memory registers; and/or28 iii. the firmware module provides the at least one address pointer to the hardware
29 module repeatedly during different points in time.

30 6. The system of any one of the preceding claims, wherein:

31 the hardware module receives a command to perform a wait-for-done operation;

32 the hardware module pauses operating until detecting that a hardware thread has

1 completed; and

2 the hardware module resumes operating in response to detecting that the hardware
3 thread has completed; and preferably

4 wherein:

5 the command to perform the wait-for-done operation comprises a sequence
6 identifier; and

7 the hardware module facilitates cross-thread dependency by pausing operating until
8 detecting that the hardware thread specified by the sequence identifier has completed.

9 7. The system of any one of the preceding claims, wherein:

10 the hardware module receives a command to perform a terminate operation; and
11 in response, the hardware module:

12 pauses operating until detecting that at least one hardware thread has
13 completed;

14 drains prefetched data;

15 empties a command queue; and

16 confirms a completion of the terminate operation to the firmware module.

17 8. The system of any one of the preceding claims, wherein:

18 the hardware module receives a command from the firmware to perform a debug
19 operation; and

20 in response, the hardware module writes data to memory that is accessible to the
21 firmware.

22 9. The system of any one of the preceding claims, wherein the hardware module
23 stores a timeout value that, when reached, prompts the hardware module to:

24 pause operating; and

25 send a timeout message to the firmware module; and/or preferably

26 wherein the hardware module stores, in at least one memory register within the
27 plurality of memory registers, a current status of the hardware module.

28 10. A computer-implemented method comprising:

29 identifying a hardware module that:

30 directly reads from and writes to a plurality of memory registers; and

31 is configured to interpret a special marker that distinguishes between register
32 write operations and non-register-write operations;

1 sending, by a firmware module, a command to the hardware module directing the
2 hardware module to perform a non-register-write operation via the special marker;
3 receiving, by the hardware module, the command directing the hardware module to
4 perform the non-register-write operation via the special marker; and
5 performing, by the hardware module, in response to receiving the command, the non-
6 register-write operation signified by the special marker.

7 11. The computer-implemented method of claim 10, wherein:
8 the non-register-write operation comprises a register read operation; and
9 the hardware module performs the register read operation by reading data from a
10 memory register within the plurality of memory registers.

11 12. The computer-implemented method of claim 10 or claim 11, wherein:
12 the non-register-write operation comprises a wait-for-done operation;
13 the hardware module performs the wait-for-done operation by pausing operating
14 until the hardware module detects that a hardware thread has completed; and
15 the hardware module resumes operating in response to detecting that the hardware
16 thread has completed; and preferably

17 wherein:
18 the command to perform the wait-for-done operation comprises a sequence
19 identifier; and
20 the hardware module facilitates cross-thread dependency by pausing operating until
21 detecting that the hardware thread specified by the sequence identifier has completed.

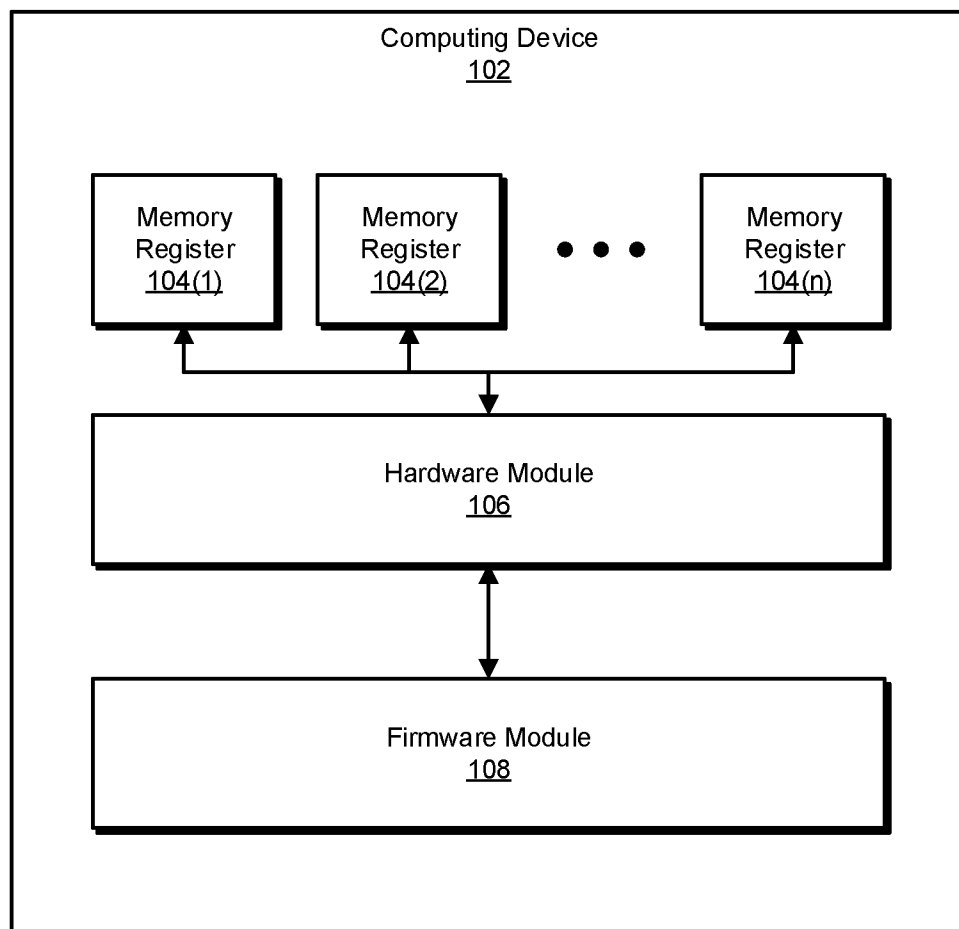
22 13. The computer-implemented method of claim 10, claim 11 or claim 12,
23 wherein:
24 the non-register-write operation comprises a debug operation; and
25 the hardware module performs the debug operation by writing data to memory that
26 is accessible to the firmware.

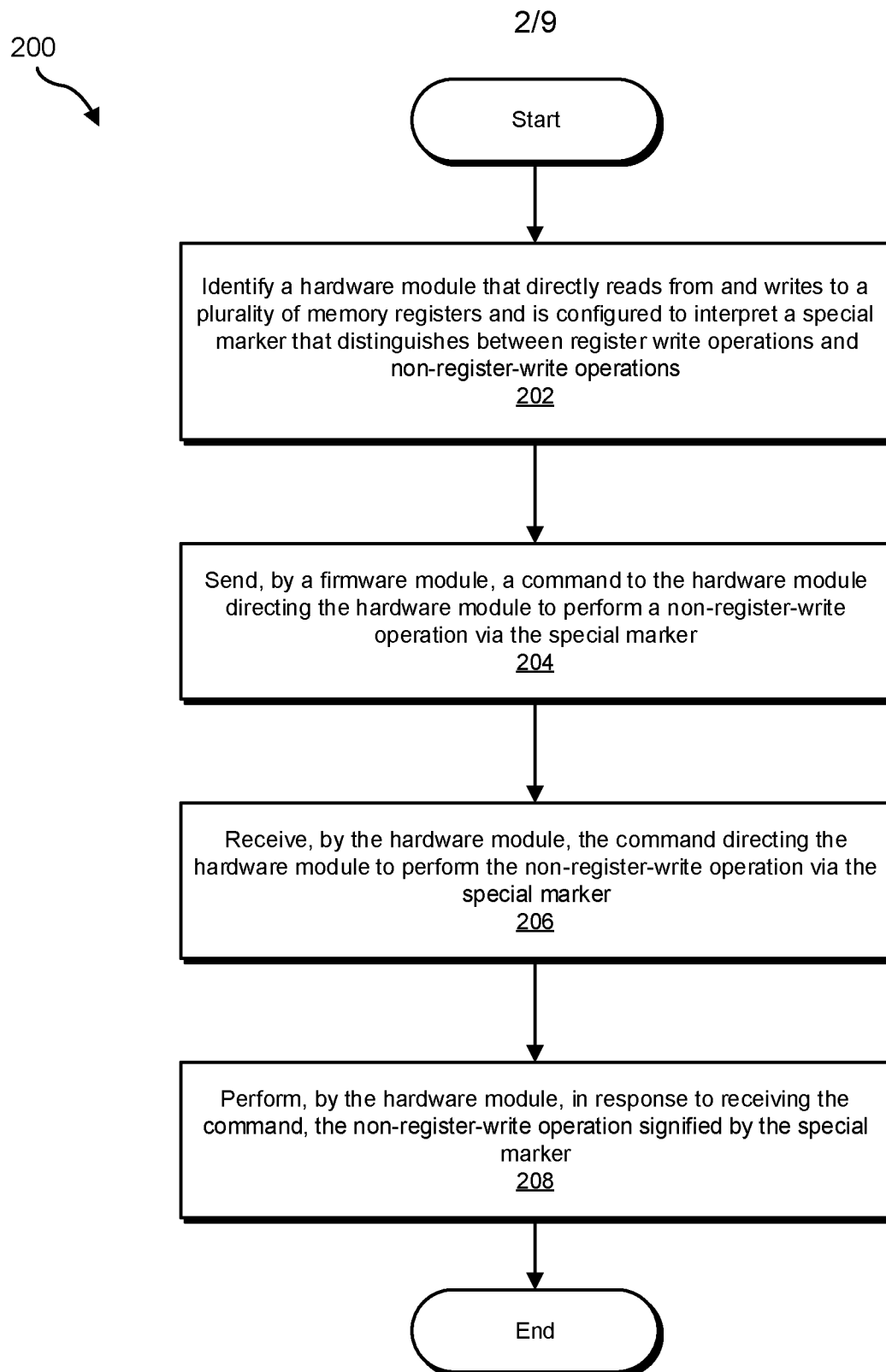
27 14. The computer-implemented method of any one of claims 10 to 13, wherein:
28 the non-register-write operation comprises a terminate operation; and
29 the hardware module performs the terminate operation by:
30 pausing operating until detecting that at least one hardware thread has
31 completed;
32 draining prefetched data;

1 emptying a command queue; and
2 confirming a completion of the terminate operation to the firmware module.
3 15. An apparatus comprising:
4 a plurality of memory registers;
5 a hardware module that:
6 directly reads from and writes to the plurality of memory registers; and
7 is configured to interpret a special marker that distinguishes between register
8 write operations and non-register-write operations; and
9 a hardware element configured to execute a firmware module that directs the
10 hardware module to perform operations at least in part by sending the special marker.

1/9

100

**FIG. 1**

**FIG. 2**

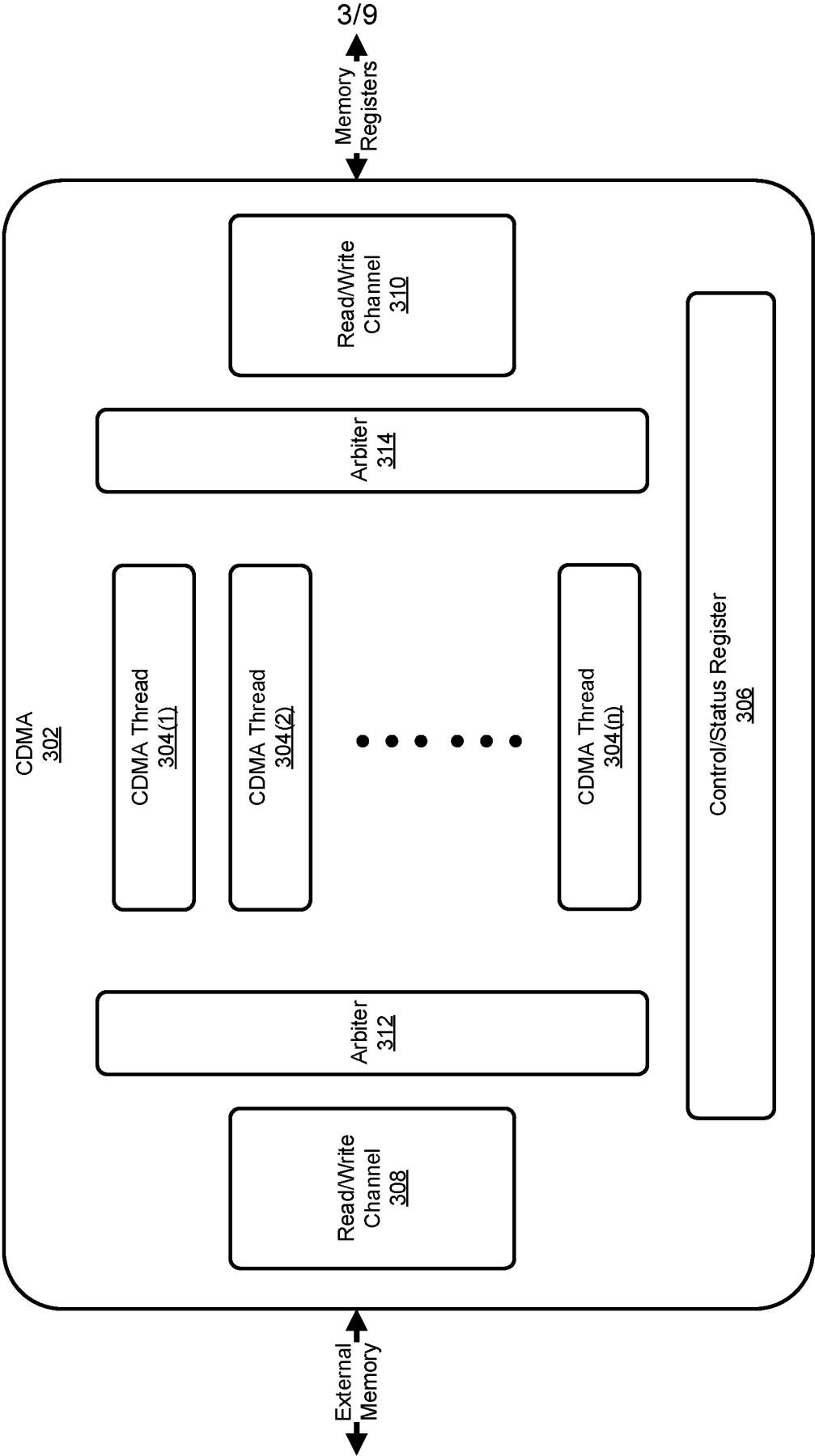


FIG. 3

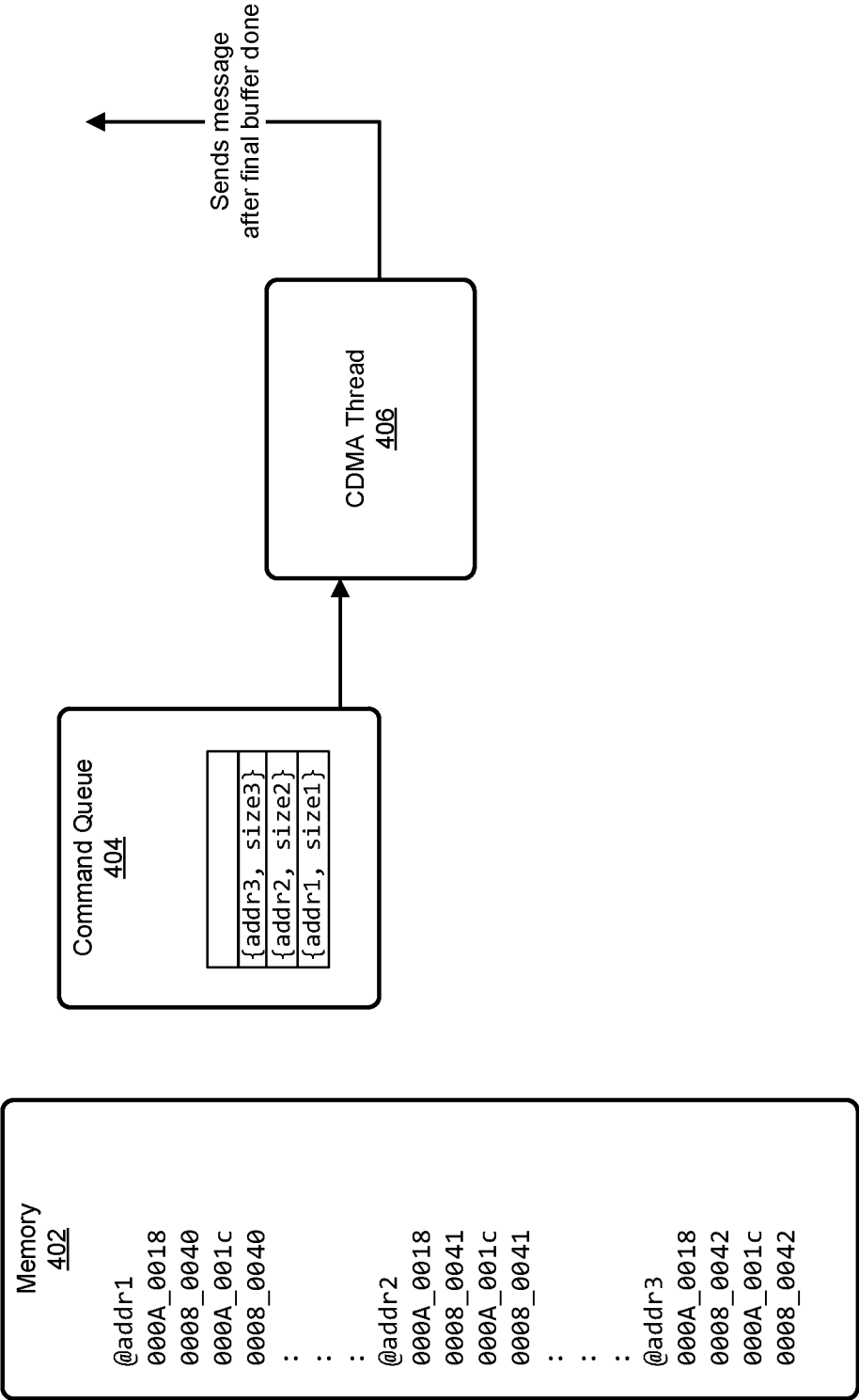


FIG. 4

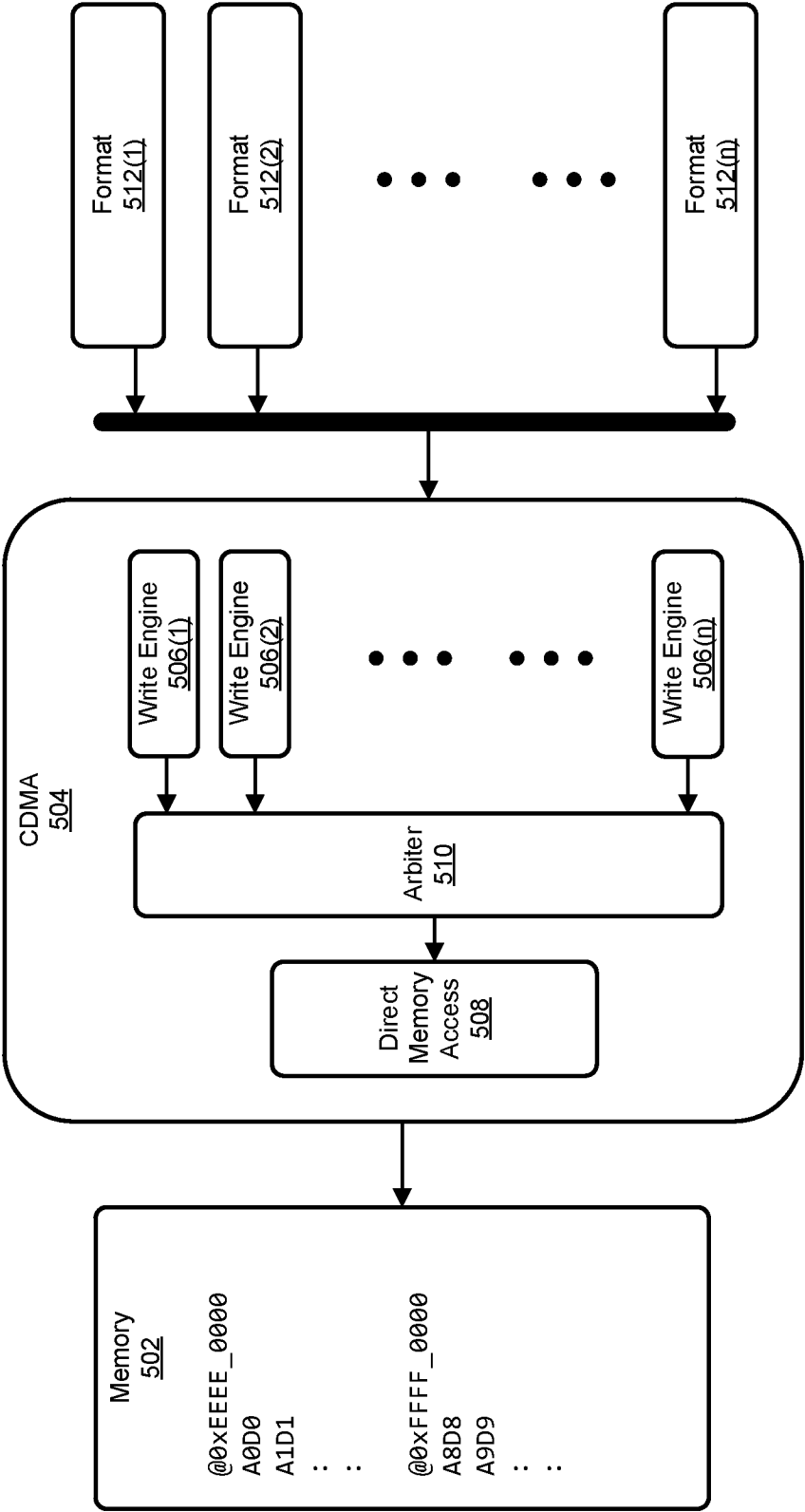


FIG. 5

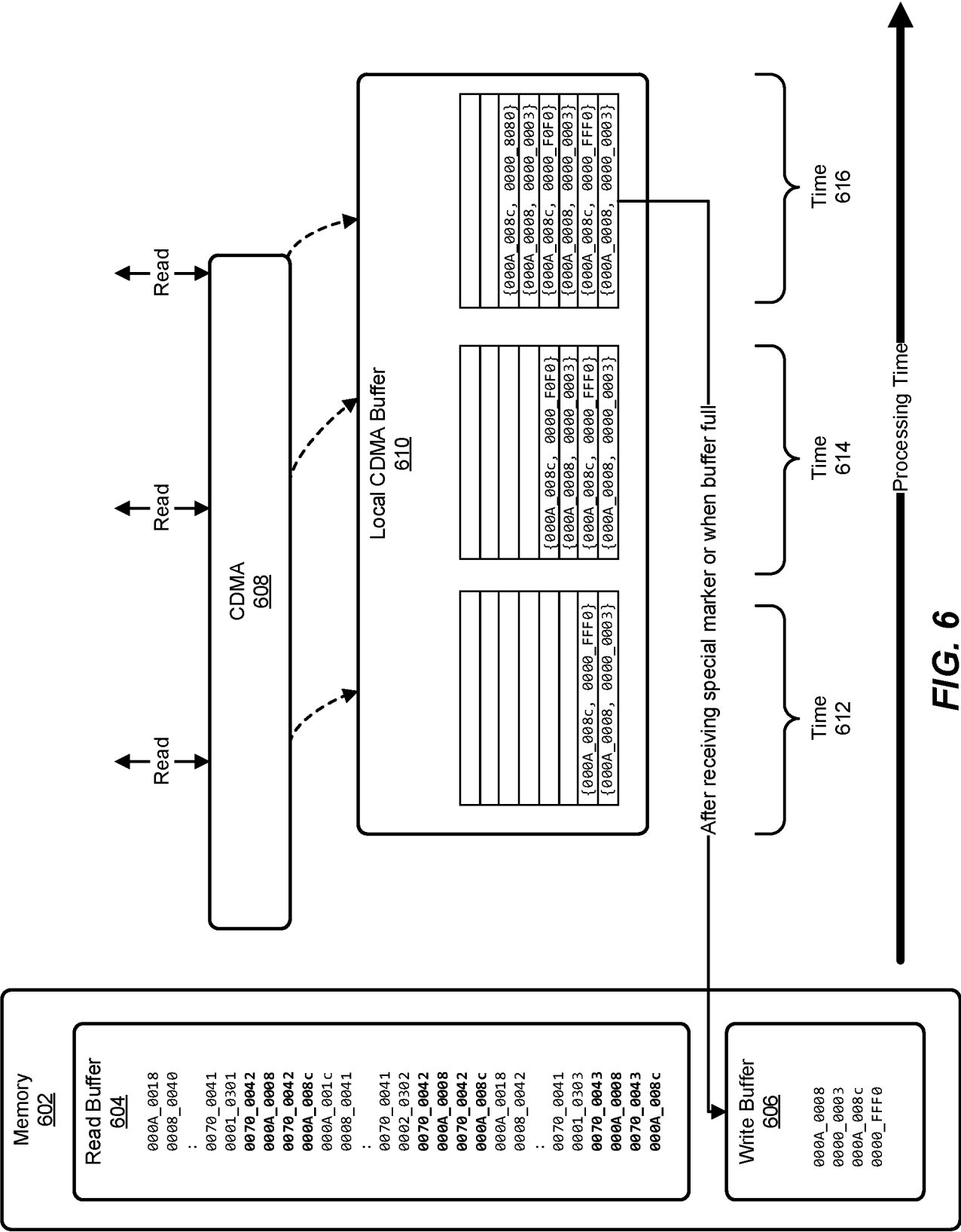


FIG. 6

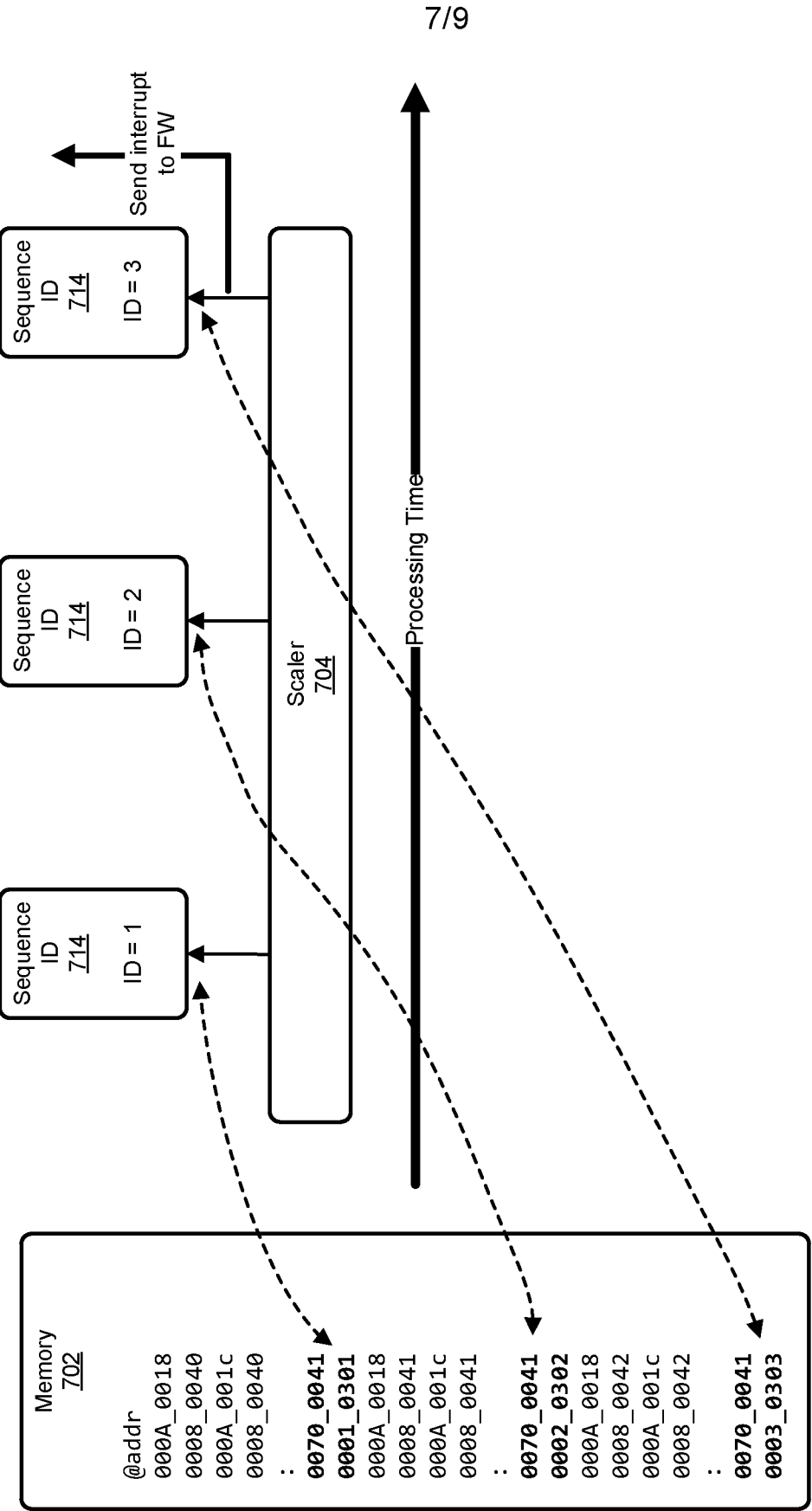


FIG. 7

8/9

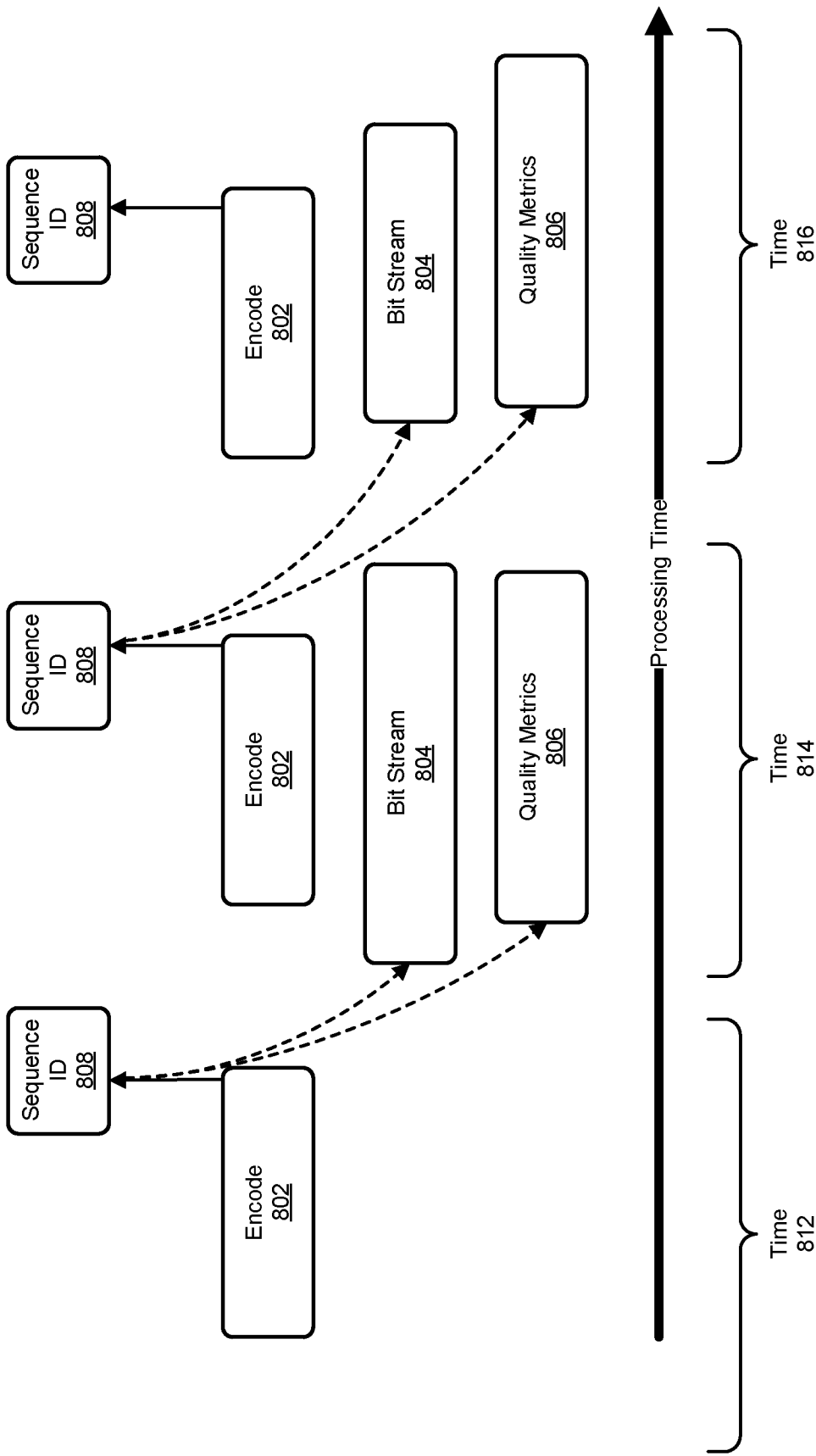
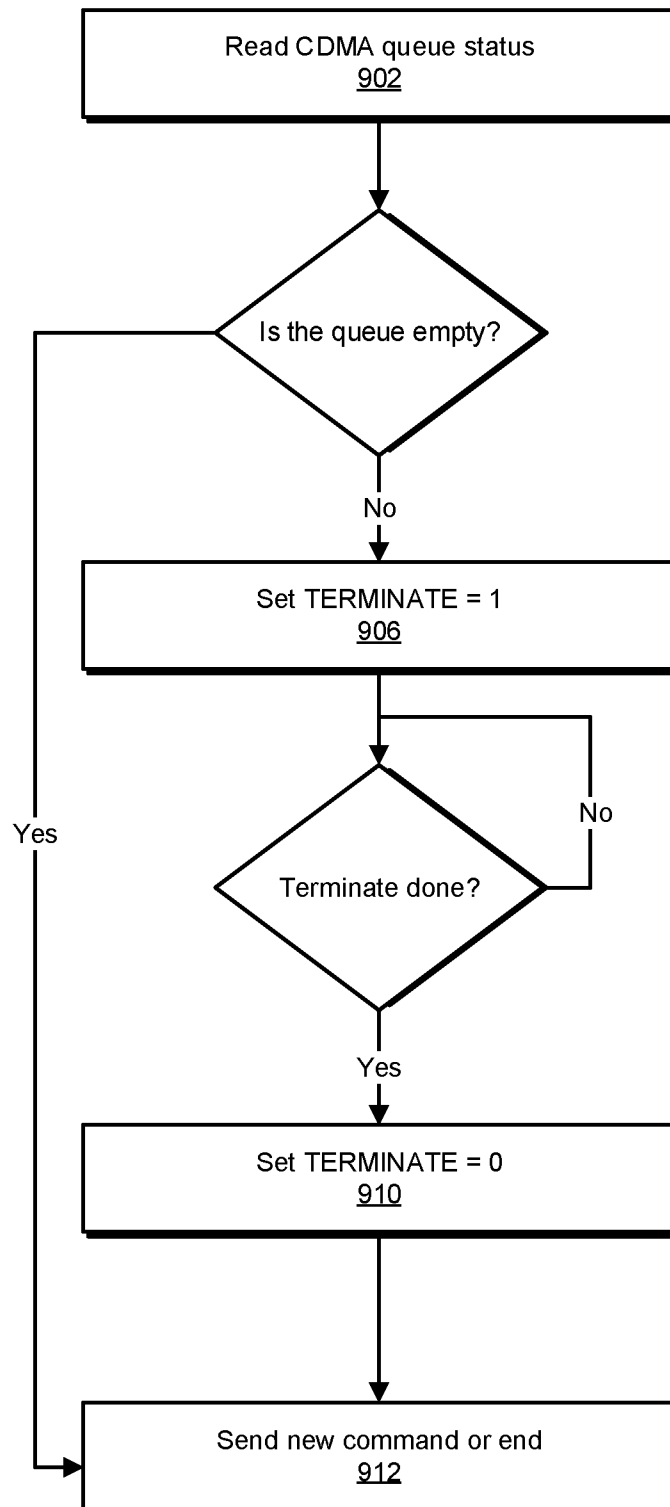


FIG. 8

9/9

**FIG. 9**

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2022/018060

A. CLASSIFICATION OF SUBJECT MATTER

INV. G06F9/30

ADD.

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

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)

EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 4 807 113 A (MATSUMOTO HIDEKAZU [JP] ET AL) 21 February 1989 (1989-02-21) column 2, line 37 - column 6, line 49; claims 1-11; figures 1-7 -----	1-15
X	US 2006/248391 A1 (GLOVER JEFFREY C [US] ET AL) 2 November 2006 (2006-11-02) paragraphs [0009] - [0012], [0018] - [0088]; figures 1-4 -----	1-15
A	US 2011/219208 A1 (ASAAD SAMEH [US] ET AL) 8 September 2011 (2011-09-08) paragraphs [0013] - [0030], [1834] ----- -/-	1-15



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

6 July 2022

Date of mailing of the international search report

19/07/2022

Name and mailing address of the ISA/

European Patent Office, P.B. 5818 Patentlaan 2

NL - 2280 HV Rijswijk

Tel. (+31-70) 340-2040,

Fax: (+31-70) 340-3016

Authorized officer

Freitas, Arthur

INTERNATIONAL SEARCH REPORT

International application No

PCT/US2022/018060

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	PINTO S ET AL: "Exploiting Template Metaprogramming to customize an object-oriented operating system", INDUSTRIAL ELECTRONICS (ISIE), 2013 IEEE INTERNATIONAL SYMPOSIUM ON, IEEE, 28 May 2013 (2013-05-28), pages 1-6, XP032439400, ISSN: 2163-5137, DOI: 10.1109/ISIE.2013.6563724 ISBN: 978-1-4673-5194-2 page 2, right-hand column -----	1-15
A	US 2002/099532 A1 (TRAUT ERIC P [US]) 25 July 2002 (2002-07-25) paragraph [0029] -----	1-15
A	GILBERT D A ET AL: "A result forwarding mechanism for asynchronous pipelined systems", ADVANCED RESEARCH IN ASYNCHRONOUS CIRCUITS AND SYSTEMS, 1997. PROCEEDINGS., THIRD INTERNATIONAL SYMPOSIUM ON EINDHOVEN, NETHERLANDS 7-10 APRIL 1997, LOS ALAMITOS, CA, USA, IEEE COMPUT. SOC, US, 7 April 1997 (1997-04-07), pages 2-11, XP010223488, DOI: 10.1109/ASYNC.1997.587137 ISBN: 978-0-8186-7922-3 page 6, right-hand column -----	1-15

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No

PCT/US2022/018060

Patent document cited in search report		Publication date	Patent family member(s)	Publication date
US 4807113	A	21-02-1989	EP 0087785 A2	07-09-1983
			JP S6232503 B2	15-07-1987
			JP S58146941 A	01-09-1983
			US 4807113 A	21-02-1989

US 2006248391	A1	02-11-2006	NONE	

US 2011219208	A1	08-09-2011	US 2011219208 A1	08-09-2011
			US 2016011996 A1	14-01-2016

US 2002099532	A1	25-07-2002	AT 456087 T	15-02-2010
			ES 2336892 T3	19-04-2010
			US 2002082823 A1	27-06-2002
			US 2002099532 A1	25-07-2002
			US 2005055192 A1	10-03-2005
			US 2005091029 A1	28-04-2005
