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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
23 March 2006 (23.03.2006)

PCT

(10) International Publication Number
WO 2006/031462 A1

(51) International Patent Classification[7]: F02B 3/00, F02F 3/26

(21) International Application Number:
PCT/US2005/031295

(22) International Filing Date:
1 September 2005 (01.09.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

60/609,211	10 September 2004 (10.09.2004)	US
11/024,002	28 December 2004 (28.12.2004)	US

(63) Related by continuation (CON) or continuation-in-part (CIP) to earlier applications:

US	11/024,002 (CON)
Filed on	28 December 2004 (28.12.2004)
US	60/609,211 (CON)
Filed on	10 September 2004 (10.09.2004)

(71) Applicant *(for all designated States except US)*: **CAVIUM NETWORKS** [US/US]; 2610 Augustine Drive, Santa Clara, CA 95054 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **BOUCHARD, Gregg, A.** [US/US]; 8713 Sea Ash Circle, Round Rock, TX 78681 (US). **CARLSON, David, A.** [US/US]; 13909 Bates Aston Road, Haslet, TX 76052 (US). **KESSLER, Richard, E.** [US/US]; 30 Thestland Drive, Shrewsbury, MA 01545 (US). **HUSSAIN, Muhammad, R.** [PK/US]; 3753 Rose Rock Circle, Pleasanton, CA 94588 (US).

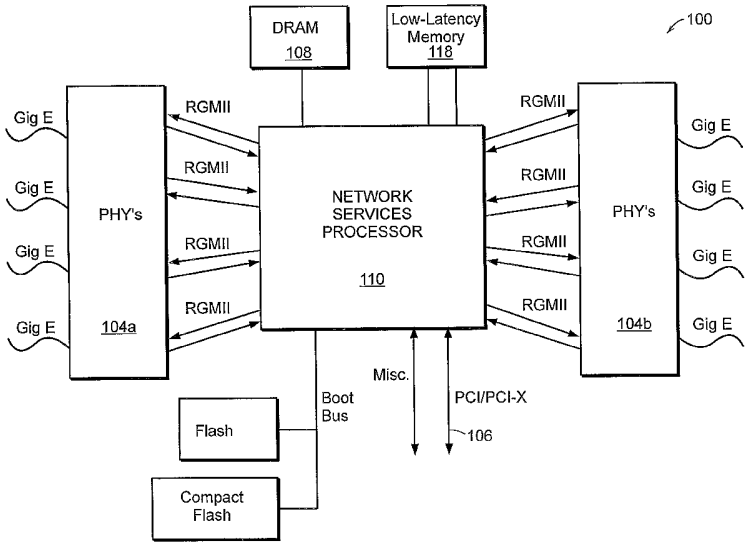
(74) Agents: **MEAGHER, Timothy, J.** et al.; Hamilton, Brook, Smith & Reynolds, P.C., 530 Virginia Road, P.O. Box 9133, Concord, MA 01742-9133 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US (patent), UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI,

[Continued on next page]

(54) Title: DIRECT ACCESS TO LOW-LATENCY MEMORY



(57) Abstract: A content aware application processing system is provided for allowing directed access to data stored in a non-cache memory thereby bypassing cache coherent memory. The processor includes a system interface to cache coherent memory and a low latency memory interface to a non-cache coherent memory. The system interface directs memory access for ordinary load/store instructions executed by the processor to the cache coherent memory. The low latency memory interface directs memory access for non-ordinary load/store instructions executed by the processor to the non-cache memory, thereby bypassing the cache coherent memory. The non-ordinary load/store instruction can be a coprocessor instruction. The memory can be a low-latency type memory. The processor can include a plurality of processor cores.

WO 2006/031462 A1

FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

DIRECT ACCESS TO LOW-LATENCY MEMORY

RELATED APPLICATION

This application is a continuation of U.S. Application No. 11/024,002, filed December 28, 2004, which claims the benefit of U.S. Provisional Application No. 60/609,211, filed on September 10, 2004. The entire teachings of the above application are incorporated herein by reference.

BACKGROUND OF THE INVENTION

The Open Systems Interconnection (OSI) Reference Model defines seven network protocol layers (L1-L7) used to communicate over a transmission medium. The upper layers (L4-L7) represent end-to-end communications and the lower layers (L1-L3) represent local communications.

Networking application aware systems need to process, filter and switch a range of L3 to L7 network protocol layers, for example, L7 network protocol layers such as, HyperText Transfer Protocol (HTTP) and Simple Mail Transfer Protocol (SMTP), and L4 network protocol layers such as Transmission Control Protocol (TCP). In addition to processing the network protocol layers, the networking application aware systems need to simultaneously secure these protocols with access and content based security through L4-L7 network protocol layers including Firewall, Virtual Private Network (VPN), Secure Sockets Layer (SSL), Intrusion Detection System (IDS), Internet Protocol Security (IPSec), Anti-Virus (AV) and Anti-Spam functionality at wire-speed.

Network processors are available for high-throughput L2 and L3 network protocol processing, that is, performing packet processing to forward packets at wire-speed. Typically, a general purpose processor is used to process L4-L7 network protocols that require more intelligent processing. For example, the Transmission Control Protocol (TCP) - an L4 network protocol requires several compute intensive tasks including computing a checksum over the entire payload in the packet, management of TCP segment buffers, and maintaining multiple timers at all times on a per connection basis. Although a general purpose processor can perform the compute intensive tasks, it does not provide sufficient performance to process the data so that it can be forwarded at wire-speed.

Furthermore, content aware applications that examine the content of packets require searching for expressions, which contain both fixed strings and character classes repeated a variable number of times, in a data stream. Several search algorithms are used to perform this task in software. One such algorithm is the Deterministic Finite Automata (DFA). There are limitations when using the DFA search algorithm, such as, exponential growth of graph size and false matches in a data stream with repeated patterns.

Due to these limitations, content processing applications require a significant amount of post processing of the results generated by pattern search. Post processing requires qualifying the matched pattern with other connection state information such as type of connection, and certain values in a protocol header included in the packet. It also requires certain other types of compute intensive qualifications, for example, a pattern match is valid only if it is within a certain position range within data stream, or if it is followed by another pattern and within certain range from the previous pattern or after/at a specific offset from the previous pattern. For example, regular expression matching combines different operators and single characters allowing complex expressions to be constructed.

SUMMARY OF THE INVENTION

The present invention is directed to increasing the speed at which a processor can perform content processing applications. The processor includes a system interface to cache coherent memory and a low latency memory interface to a non-

cache coherent memory. The system interface directs memory access for ordinary load/store instructions executed by the processor to the cache coherent memory. The low latency memory interface directs memory access for non-ordinary load/store instructions executed by the processor to the non-cache memory, thereby bypassing the cache coherent memory. The non-ordinary load/store instruction can be a coprocessor instruction. The memory can be a low-latency type memory. The processor can include a plurality of processor cores.

In one embodiment, the low latency interface can be a bus coupling the processor to the non-cached memory, the coupling allowing direct access between the processor and the non-cached memory. In another embodiment, data can be stored in a deterministic finite automata (DFA) graph in the memory for performing the content aware application processing.

In another embodiment, the processor can include a plurality of registers for moving the data between the processor core and the memory. The plurality of registers can be located within the processor. The plurality of registers located within the processor can be separate from a main register file located within the processor.

In another embodiment, the low-latency memory can be selected from a group consisting of dynamic random access memory (DRAM), reduced latency dynamic random access memory (RLDRAM), static random access memory (SRAM), and fast cycle random access memory (FCRAM), wherein the processor accesses the RLDRAM with less than or equal to 30 nanosecond latency.

A network services processor integrates network, security and content processing according to the principles of the present invention. The network services processor includes built-in hardware acceleration for content and security processing, along with on-chip co-processor modules for Internet Services acceleration.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other objects, features and advantages of the invention will be apparent from the following more particular description of preferred embodiments of the invention, as illustrated in the accompanying drawings in which

like reference characters refer to the same parts throughout the different views. The drawings are not necessarily to scale, emphasis instead being placed upon illustrating the principles of the invention.

FIG. 1A is a block diagram of a network service processing system including a network services processor according to the principles of the present invention;

FIG. 1B is a block diagram of the network services processor shown in Fig. 1A;

FIG. 2 illustrates a typical DFA graph;

FIG. 3 is a block diagram of a Reduced Instruction Set Computing (RISC) processor according to the principles of the present invention;

FIG. 4 illustrates a LLM load/store instruction format; and

FIG. 5 shows an example use of a load/store operation according to the present invention.

DETAILED DESCRIPTION OF THE INVENTION

A description of preferred embodiments of the invention follows.

FIG. 1A is a block diagram of a security appliance 100 including a network services processor 110 according to the principals of the present invention. The security appliance 100 is a standalone system that can switch packets received at one Ethernet port (Gig E) to another Ethernet port (Gig E) and perform a plurality of security functions on received packets prior to forwarding the packets. For example, the security appliance 100 can be used to perform security processing on packets received on a Wide Area Network prior to forwarding the processed packets to a Local Area Network.

The network services processor 110 includes hardware packet processing, buffering, work scheduling, ordering, synchronization, and cache coherence support to accelerate all packet processing tasks. The network services processor 110 processes Open System Interconnection network L2-L7 layer protocols encapsulated in received packets.

The network services processor 110 receives packets from the Ethernet ports (Gig E) through physical interfaces PHY 104a, 104b, performs L7-L2 network protocol processing on the received packets and forwards processed packets through the physical interfaces 104a, 104b or through a PCI bus 106. The network protocol

processing can include processing of network security protocols such as Firewall, Application Firewall, Virtual Private Network (VPN) including IP Security (IPSEC) and/or Secure Sockets Layer (SSL), Intrusion detection System (IDS) and Anti-virus (AV).

A Dynamic Random Access Memory (DRAM) controller in the network services processor 110 controls access to an external DRAM 108 that is coupled to the network services processor 110. The DRAM 108 is external to the network services processor 110. The DRAM 108 stores data packets received from the PHYs interfaces 104a, 104b or the Peripheral Component Interconnect Extended (PCI-X) interface 106 for processing by the network services processor 110.

A low-latency memory controller in the network services processor 110 controls low-latency memory (LLM) 118. The LLM 118 can be used for Internet Services and Security applications allowing fast lookups, including regular expression matching that may be required for Intrusion Detection System (IDS) or Anti Virus (AV) applications.

Regular expressions are a common way to express string matching patterns. The atomic elements of a regular expression are the single characters to be matched. These are combined with meta-character operators that allow a user to express concatenation, alternation, Kleene-star, etc. Concatenation is used to create multiple character matching patterns from a single charters (or sub-strings) while alternation (|) is used to create patterns that can match any of two or more sub-strings. Kleene-star (*) allows a pattern to match zero (0) or more occurrences of the pattern in a string. Combining different operators and single characters allows complex expressions to be constructed. For example, the expression (th(is|at)*) will match th, this, that, thisis, thisat, thatis, thatat, etc.

FIG. 1B is a block diagram of the network services processor 110 shown in FIG. 1A. The network services processor 110 delivers high application performance using at least one processor core 120 as described in conjunction with FIG. 1A.

A packet is received for processing by any one of the GMX/SPX units 122a, 122b through an SPI-4.2 or RGM II interface. A packet can also be received by a PCI interface 124. The GMX/SPX unit (122a, 122b) performs pre-processing of the received packet by checking various fields in the L2 network protocol header

included in the received packet and then forwards the packet to a packet input unit 126.

The packet input unit 126 performs further pre-processing of network protocol headers (L3 and L4) included in the received packet. The pre-processing includes checksum checks for Transmission Control Protocol (TCP)/User Datagram Protocol (UDP) (L3 network protocols).

A Free Pool Allocator (FPA) 128 maintains pools of pointers to free memory in level 2 cache memory 130 and DRAM 108. The input packet processing unit 126 uses one of the pools of pointers to store received packet data in level 2 cache memory 130 or DRAM 108 and another pool of pointers to allocate work queue entries for the processor cores 120.

The packet input unit 126 then writes packet data into buffers in Level 2 cache 130 or DRAM 108 in a format that is convenient to higher-layer software executed in at least one processor core 120 for further processing of higher level network protocols.

The network services processor 110 also includes application specific co-processors that offload the processor cores 120 so that the network services processor achieves high-throughput. The compression/decompression co- processor 132 is dedicated to performing compression and decompression of received packets. In one embodiment, a deterministic finite automata (DFA) module (not shown) may include dedicated DFA engines to accelerate pattern and signature match necessary for anti-virus (AV), Intrusion Detection Systems (IDS) and other content processing applications at up to 4 Gbps.

An I/O Interface (IOI) 136 manages the overall protocol and arbitration and provides coherent I/O partitioning. The IOI 136 includes an I/O Bridge (IOB) 138 and a Fetch and Add Unit (FAU) 140. Registers in the FAU 140 are used to maintain lengths of the output queues that are used for forwarding processed packets through the packet output unit 126. The IOB 138 includes buffer queues for storing information to be transferred between an I/O Bus 142, a coherent memory bus 144, the packet input unit 126 and the packet output unit 146.

A Packet order/work (POW) module 148 queues and schedules work for the processor cores 120. Work is queued by adding a work queue entry to a queue. For

example, a work queue entry is added by the packet input unit 126 for each packet arrival. A timer unit 150 is used to schedule work for the processor cores.

Processor cores 120 request work from the POW module 148. The POW module 148 selects (i.e. schedules) work for a processor core 120 and returns a pointer to the work queue entry that describes the work to the processor core 120.

The processor core 120 includes instruction cache 152, level 1 (L1) data cache 154 and crypto acceleration 156. In one embodiment, the network services processor 100 (FIG. 1A) includes sixteen superscalar RISC (Reduced Instruction Set Computer)-type processor cores 120. In one embodiment, each superscalar RISC-type processor core 120 is an extension of the MIPS64 version 2 processor core.

Level 2 (L2) cache memory 130 and DRAM 108 is shared by all of the processor cores 120 and I/O co-processor devices. Each processor core 120 is coupled to the Level 2 cache memory 130 by the coherent memory bus 144. The coherent memory bus 144 is a communication channel for all memory and I/O transactions between the processor cores 100, the IOI 136 and the L2 cache memory 130 and a L2 cache memory controller 131. In one embodiment, the coherent memory bus 144 is scalable to 16 processor cores 120, supports fully coherent L1 data caches 154 with write through, is highly buffered and can prioritize I/O.

The L2 cache memory controller 131 maintains memory reference coherence. It returns the latest copy of a block for every fill request, whether the block is stored in L2 cache memory 130, in DRAM 108 or is in-flight. It also stores a duplicate copy of the tags for the data cache 154 in each processor core 120. It compares the addresses of cache block store requests against the data cache tags, and invalidates (both copies) a data cache tag for a processor core 120 whenever a store instruction is from another processor core or from an I/O component via the IOI 136.

A DRAM controller 133 supports up to 16 Mbytes of DRAM. The DRAM controller 133 supports a 64-bit or 128-bit interface to DRAM 108. The DRAM controller 133 supports DDR-I (Double Data Rate) and DDR-II protocols.

After the packet has been processed by the processor cores 120, the packet output unit (PKO) 146 reads the packet data from memory, performs L4 network protocol post-processing (e.g., generates a TCP/UDP checksum), forwards the packet through the GMX/SPC unit 122a, 122b and frees the L2 cache 130/DRAM 108 used by the packet.

A low-latency memory controller 160 manages in-flight transactions (loads/stores) to/from the LLM 118. The low-latency memory (LLM) 118 is shared by all of the processor cores 120. The LLM 118 can be dynamic random access memory (DRAM), reduced latency dynamic random access memory (RLDRAM), synchronous random access memory (SRAM), fast cycle random access memory (FCRAM) or any other type of low-latency memory known in the art. The RLDRAM provides 30 nanosecond memory latency or better; that is, the time taken to satisfy a memory request initiated by the processor 120. Each processor core 120 is directly coupled to the LLM controller 160 by a low-latency memory bus 158. The low-latency memory bus 158 is a communication channel for content aware application processing between the processor cores 120 and the LLM controller 160. The LLM controller 160 is coupled between the processor cores 120 and the LLM 118 for controlling access to the LLM 118.

Content aware application processing utilizes patterns/expressions (data) stored in the LLM 118. The patterns/expressions may be in the form of a deterministic finite automata (DFA). The DFA is a state machine. The input to the DFA state machine is a string of (8-bit) bytes (i.e. the alphabet for the DFA is a byte.). Each input byte causes the state machine to transition from one state to the next. The states and the transition function can be represented by a graph 200 as illustrated in FIG. 2A, where each graph node (210a...210c) is a state and different graph arcs (220a...220d) represent state transitions for different input bytes. The states may contain certain characters related to the state, such as "A...Z, a...z, 0...9," etc. The current state of the state machine is a node identifier that selects a particular graph node. For instance, assume that the input contains the text "Richard". From the initial State 1 (210a), the DFA moves to State 2 (210b) where the "R" is read. For the next five characters, "i", "c", "h", "a", "r" , "d", the DFA continues to loop (220b) to State 2.

FIG. 3 is a block diagram of a Reduced Instruction Set Computing (RISC) processor 120 according to the principles of the present invention. The processor 120 includes an Integer Execution Unit 302, an Instruction Dispatch Unit 304, an Instruction Fetch Unit 306, a Memory Management Unit (MMU) 308, a System Interface 310, a Low-Latency Interface 350, a Load/Store unit 314, a Write Buffer 316, and Security Accelerators 156. The processor core 120 also includes an

EJTAG Interface 330 allowing debug operations to be performed. The system interface 310 controls access to external memory, that is, memory external to the processor 120 such as, external (L2) cache memory 130 or primary/main memory 108.

The Integer Execution unit 302 includes a multiply unit 326, at least one register file (main register file) 328, and two holding registers 330a, 330b. The holding registers 330a, 330b are used to store data to be written to the LLM 118 and data that has been read from the LLM 118 using LLM load/store instructions according to the principles of the present invention. The holding registers 330a, 330b improve the efficiency of the instruction pipeline by allowing two outstanding loads prior to stalling the pipeline. Although two holding registers are shown, one or multiple holding registers may be used. The multiply unit 326 has a 64-bit register-direct multiply. The Instruction fetch unit 306 includes instruction cache (ICache) 152. The load/store unit 314 includes a data cache 154. In one embodiment, the instruction cache 152 is 32K bytes, the data cache 154 is 8K bytes and the write buffer 316 is 2K bytes. The Memory Management Unit 308 includes a Translation Lookaside Buffer (TLB) 340.

In one embodiment, the processor 120 includes a crypto acceleration module (security accelerators) 156 that include cryptography acceleration for Triple Data Encryption standard (3DES), Advanced Encryption Standard (AES), Secure Hash Algorithm (SHA-1), Message Digest Algorithm #5 (MD5). The crypto acceleration module 156 communicates by moves to and from the main register file 328 in the Execution unit 302. RSA and the Diffie-Hellman (DH) algorithm are performed in the multiplier unit 326.

FIG. 4 illustrates a LLM load/store instruction format 410 that a core 120 uses to reference the LLM 118. These load/store instructions differ from the ordinary load store instructions in a typical general purpose instruction set that load/store data between the main register file 328 and the cache coherent memory system that includes L1 data cache 154, L2 cache memory 130, and DRAM 108 (FIG. 3). In contrast, these new instructions initiate either 64-bit or 36-bit loads/stores directly from/to LLC 118 (FIG. 3) to a core 120. These instructions allow data to be retrieved/stored in LLM memory 118 faster than through the cache coherent memory system. This path provided through the LLM load/store

instructions to low latency memory 118 improves the performance of applications that do not require caching, such as regular expression matching. These load/store instructions are "DMTC2" (double move to co-processor 2) and "DMFC2" (double move from co-processor 2).

Referring to FIG. 4, the COP2 field 412 indicates that the instruction is a co-processor instruction (i.e., not a general purpose instruction). The DMT/DMF field 414 stores the operation code (i.e., indicates the instruction type). The instruction type DMT indicates that data is being moved from low latency memory 118 to a holding register (330a, 330b). The instruction type DMF indicates that data is being moved from a holding register (330a, 330b) to a register in the main register file load. In one embodiment, the low latency memory 118 is 36-bits wide and each DMT/DMF instruction allows 36-bits to be moved. The rt field 416 identifies a register in the main register file. The impl field 418 in conjunction with the operation code identifies the type of coprocessor move instruction and identifies the holding register 330a, 330b.

To load (DMF) the contents of a holding register to a register in the main register file 338, the rt field identifies the register in the register file to which the data stored in the holding register is stored. For example:

GPR[rt] = LLM_DATA0 <63:0>, where

LMM_DATA0 is holding register 330a; and

GPR is the general purpose register.

For a write (DMT) instruction, the rt field (416) identifies the register in the register file that stores the address of location in low latency memory 118. For example:

LLM_DATA0 <63:0> = llmemory[rt], where

LLM_DATA0 is holding register 330a; and

llmemory is low latency memory.

For example the following low latency memory load instruction (DMTC2) can be used to load the holding register 330a with contents of a low-latency memory location instruction, such as:

DMTC2, $5, 0x0400

DMT is the instruction type, i.e. load holding register with data from low latency memory (414);

C2 (COP2) indicates a coprocessor instruction (412);

Register #5 in the main register file 328 (FIG. 3) holds the low-latency memory address location; and

0x0400 identifies the holding register 330 (FIG. 3) (418) (This value is constant and can be a different value in another embodiment).

Similarly, a low latency memory store instruction can be used to move data from the holding register 330 (FIG. 3) into the main register file 328 (FIG. 3). For example, the following low latency memory store instruction can be used:

DMFC2, $6, 0x0402

DMF is the instruction type, i.e. store contents of holding register in register $6 in the main register file 328 (414);

C2 (COP2) indicates a coprocessor instruction (412);

Register #6 in the main register file 328 is the destination register (rt) (416); and

0x0402 identifies the holding register 330 (FIG. 3) (418) (This value is constant and can be a different value in another embodiment).

The instruction format shown above is by way of example and it should be understood by one skilled in the art the instruction format can be any format which allows non-ordinary loads/store instructions.

FIG. 5 shows an example of the use of a non-ordinary load instruction to load data into a register in the register file from an LLM memory location according to the present invention. To load the contents of the LLM address location into register 6 ($6) in the main register file 328 using a non-ordinary load instruction, the following instruction sequence is used:

DMTC2 $5, C1 (C1 is a constant value, such as 0x0400);

DMFC2 $6, C2 (C2 is a constant value, such as 0x0402).

The address (location) in low latency memory 118 from which to load data is first stored in register 5 ($5) in the main register file 328. The "DMTC2 $5, 0x0400" instruction reads the data from the location in LLM 118 identified by the address stored in register 5 into holding register 330a. Then, the "DMFC2 $6, 0x0402" instruction loads the data stored in holding register 330a into $6 in the main

register file 328. These instructions effectively bypass all caches using the LLM Bus 158.

Holding register 330b can be used instead of holding register 330a by changing the values of C1, C2. For example, C1 = 0x0408 can be changed to C2 = 0x040a.

While this invention has been particularly shown and described with references to preferred embodiments thereof, it will be understood by those skilled in the art that various changes in form and details may be made therein without departing from the scope of the invention encompassed by the appended claims.

CLAIMS

What is claimed is:

1. A processor, comprising:

a system interface to cache coherent memory which directs memory access for ordinary load/store instructions executed by the processor to the cache coherent memory; and

a low latency memory interface to a non-cache memory which directs memory access for non-ordinary load/store instructions executed by the processor to the non-cache memory, thereby bypassing cache coherent memory.

2. The processor of Claim 1, wherein the low latency interface is a bus coupling the processor to the non-cache memory, the coupling allowing direct access between the processor and the non-cache memory.

3. The processor of Claim 1, wherein the non-ordinary load/store instruction is a coprocessor instruction.

4. The processor of Claim 1, further including a plurality of registers for moving data between the low-latency memory interface and the non-cache memory.

5. The processor of Claim 4, wherein the plurality of registers are located within an execution unit in the processor.

6. The processor of Claim 5, wherein the plurality of registers are separate from a main register file located within the execution unit in the processor.

7. The processor of Claim 4, wherein the data is stored in a deterministic finite automata (DFA) graph in the non-cache memory for performing content aware application processing.

8. The processor of Claim 1, wherein the non-cache memory is a low-latency memory.

9. The processor of Claim 8, wherein the low latency memory is selected from a group consisting of dynamic random access memory (DRAM), reduced latency dynamic random access memory (RLDRAM), static random access memory (SRAM), and fast cycle random access memory (FCRAM).

10. The processor of Claim 9, wherein the RLDRAM has a latency of less than or equal to 20 nanoseconds.

11. The processor of Claim 1, further including a plurality of processor cores.

12. A method for direct access between a processor and a non-cache memory, comprising:

directing memory access for ordinary load/store instructions executed by the processor to cache coherent memory via a system interface to the cache coherent memory; and

directing memory access for non-ordinary load/store instructions executed by the processor to the non-cache memory via a low latency memory interface to the non-cache cache coherent memory, thereby bypassing the cache coherent memory.

13. The method of Claim 12, wherein the non-ordinary load/store instruction is a coprocessor instruction.

14. The method of Claim 12, further including a plurality of registers for moving the data between the processor and the non-cache memory.

15. The method of Claim 14, wherein the plurality of registers are located within an execution unit within the processor.

16. The method of Claim 15, wherein the plurality of registers located within the processor are separate from a main register file located within the execution unit in the processor.

17. The method of Claim 14, wherein the data is stored in a deterministic finite automata (DFA) graph in the non-cache memory for performing content aware application processing.

18. The method of Claim 12, wherein the non-cache memory is a low-latency memory.

19. The method of Claim 18, wherein the low-latency memory is selected from a group consisting of dynamic random access memory (DRAM), reduced latency dynamic random access memory (RLDRAM), static random access memory (SRAM), and fast cycle random access memory (FCRAM).

20. The method of Claim 19, wherein the processor accesses the RLDRAM with less than or equal to 20 nanosecond latency.

21. The method of Claim 12, wherein the method includes a plurality of processor cores.

22. A system for increasing processor speed for content aware application processing, comprising:

a processor, the processor including an interface to a non-cache memory; and

non-ordinary load/store instructions included in a general purpose instruction set for moving data between the processor and the non-cache memory, thereby bypassing cache coherent memory.

23. A method for direct access between a processor and a non-cache memory, comprising:

means for storing data in the non-cache memory; and

means for loading the data from the non-cache memory into the processor bypassing a coherent cache memory.

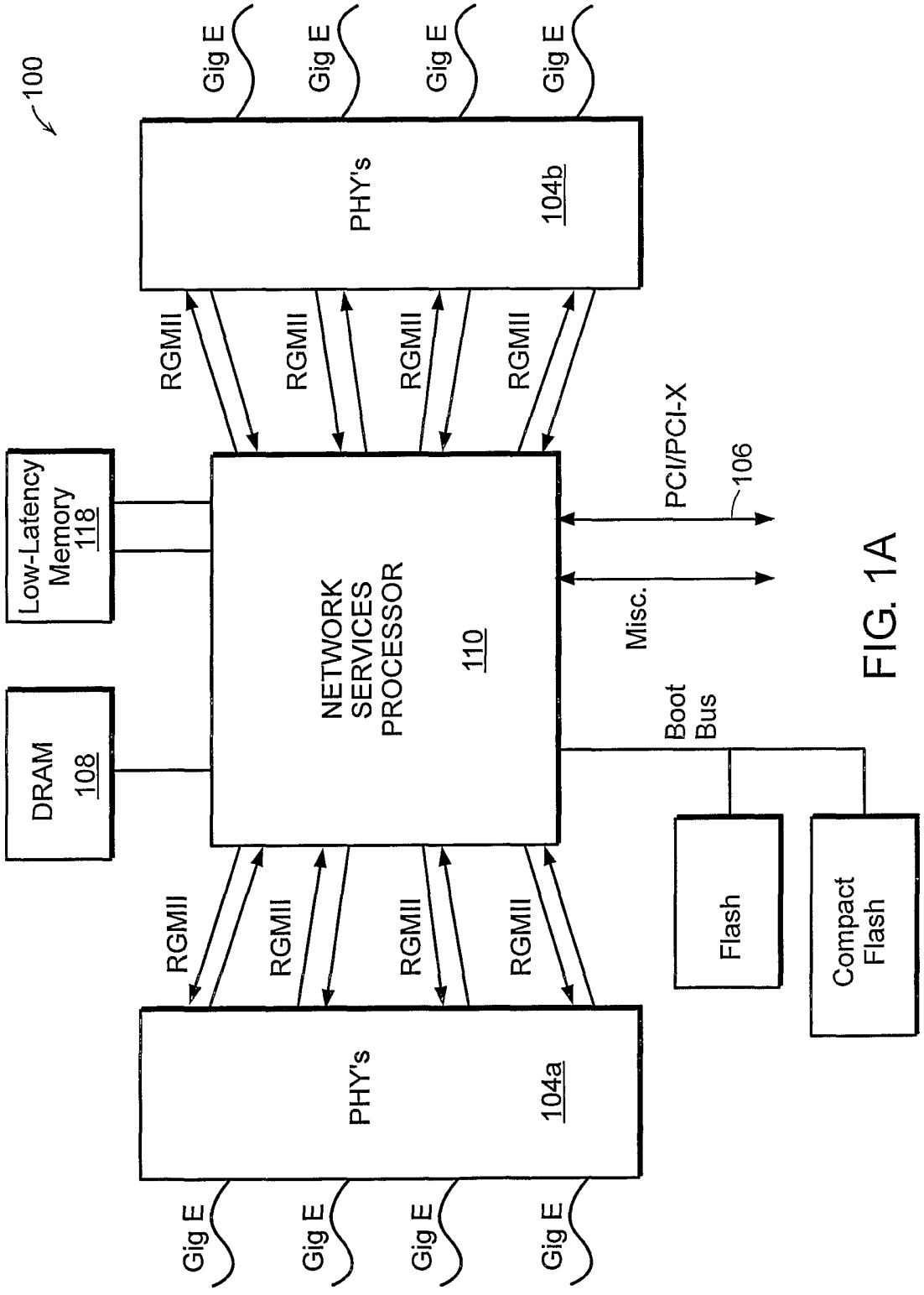

100
Gig E
Gig E
Gig E
Gig E
PHY's
104b
RGMII
RGMII
RGMII
RGMII
Low-Latency Memory 118
DRAM 108
NETWORK SERVICES PROCESSOR 110
PCI/PCI-X
106
Misc.
Boot Bus
Flash
Compact Flash
RGMII
RGMII
RGMII
RGMII
PHY's
104a
Gig E
Gig E
Gig E
Gig E

FIG. 1A

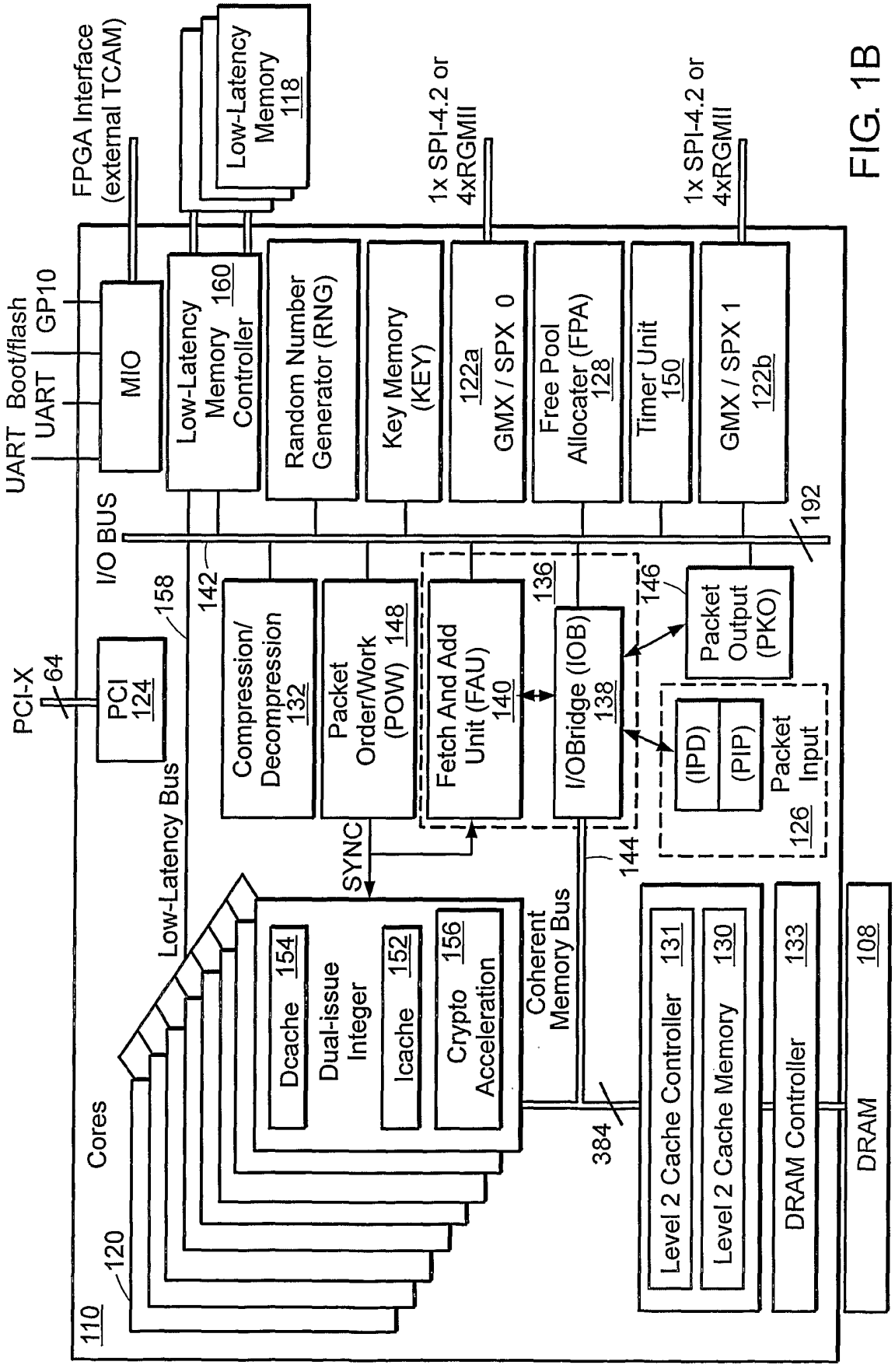

FPGA Interface (external TCAM)
Low-Latency Memory 118
1x SPI-4.2 or 4xRGMII
1x SPI-4.2 or 4xRGMII
UART Boot/flash
UART
GP10
MIO
Low-Latency Memory Controller 160
Random Number Generator (RNG)
Key Memory (KEY)
122a GMX / SPX 0
Free Pool Allocater (FPA) 128
Timer Unit 150
GMX / SPX 1 122b
I/O BUS
192
142
158
PCI-X
64
PCI 124
Compression/ Decompression 132
Packet Order/Work (POW) 148
Fetch And Add Unit (FAU) 140
136
I/OBridge (IOB) 138
146
Packet Output (PKO)
(IPD)
(PIP)
Packet Input
126
Low-Latency Bus
SYNC
144
Coherent Memory Bus
Dcache 154
Dual-issue Integer
Icache 152
Crypto Acceleration 156
Cores
120
110
384
Level 2 Cache Controller 131
Level 2 Cache Memory 130
DRAM Controller 133
DRAM 108

FIG. 1B

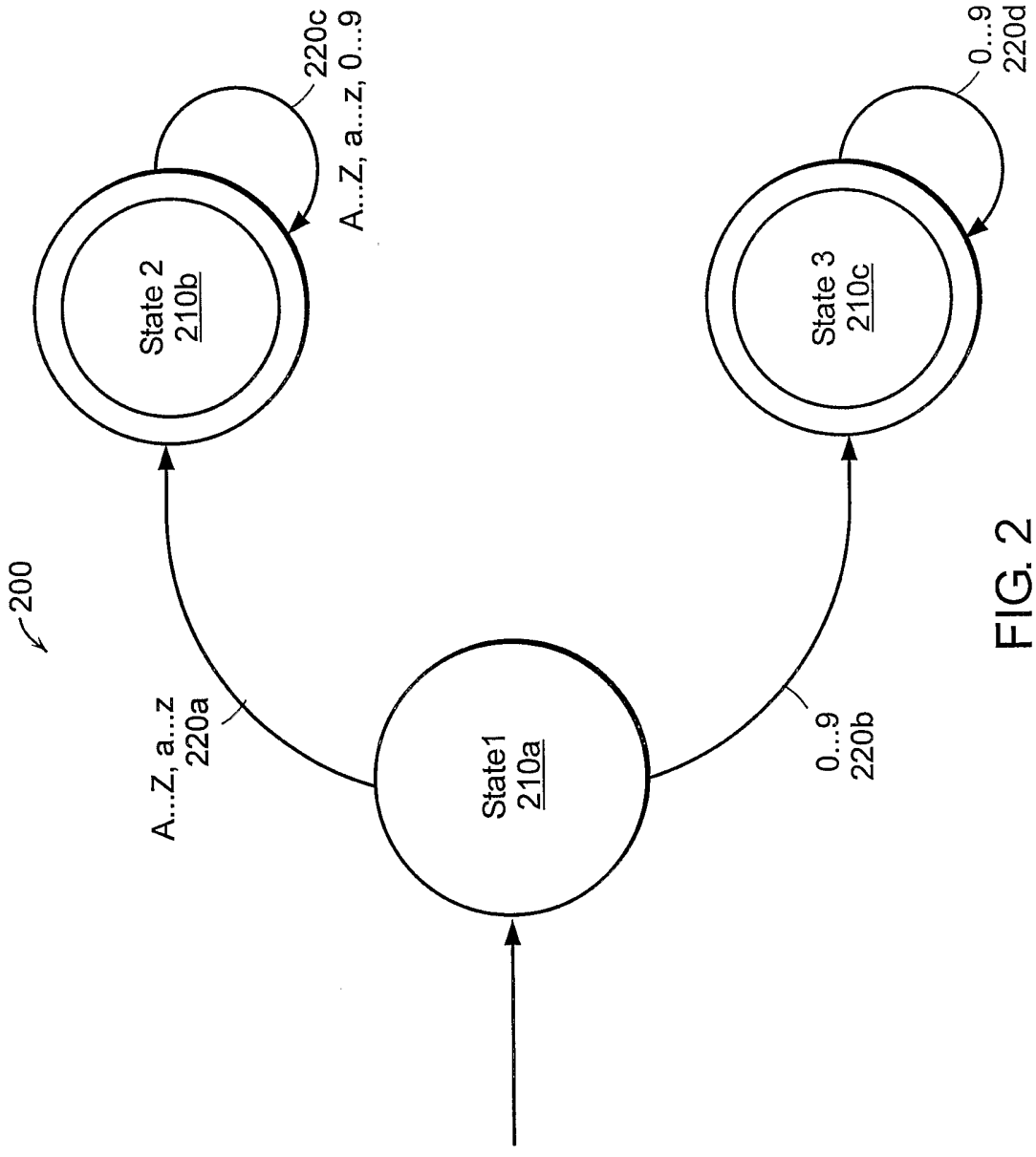

200
State1
210a
State 2
210b
State 3
210c
A...Z, a...z
220a
0...9
220b
A...Z, a...z, 0...9
220c
0...9
220d

FIG. 2

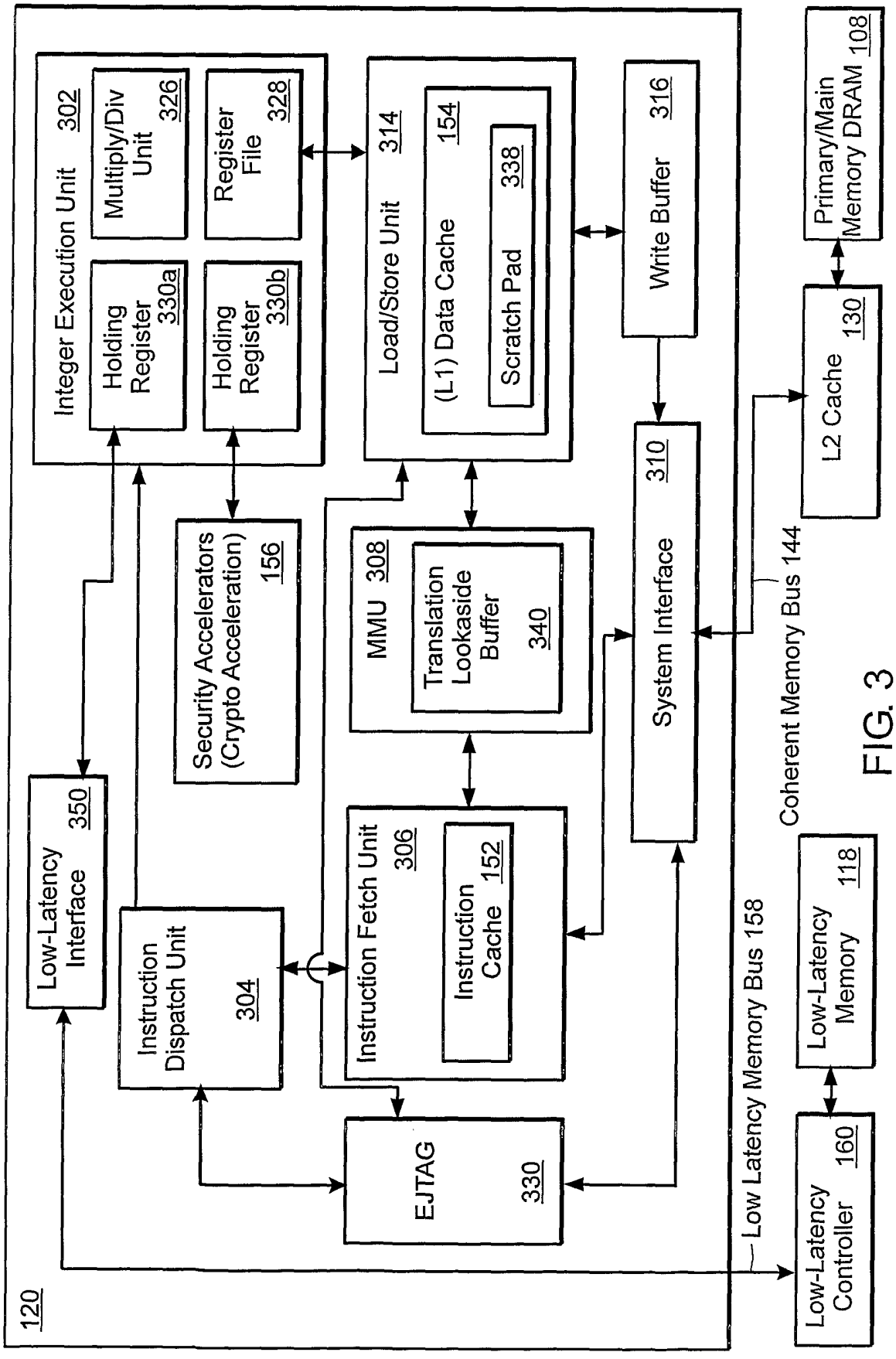
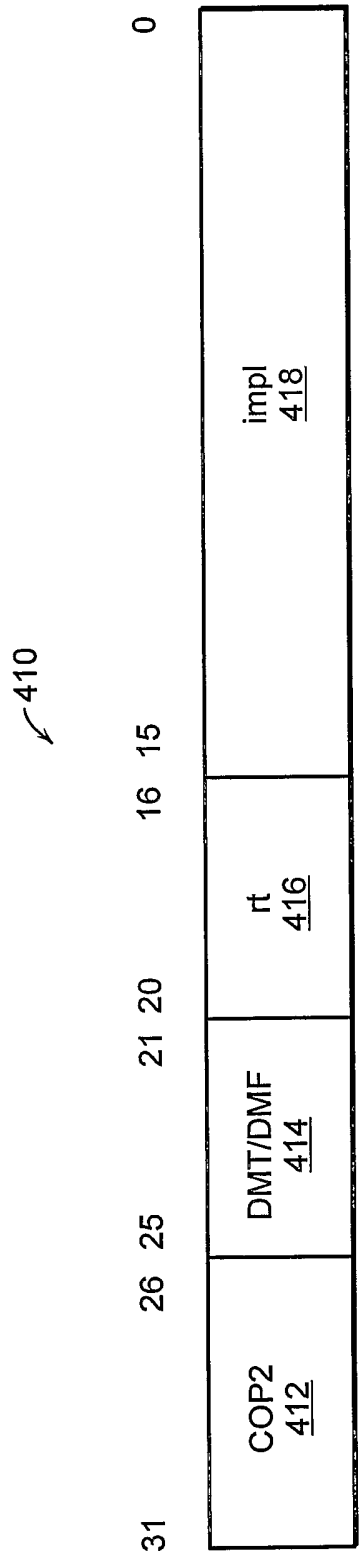

120
Integer Execution Unit 302
Multiply/Div Unit 326
Register File 328
Holding Register 330a
Holding Register 330b
Load/Store Unit 314
(L1) Data Cache 154
Scratch Pad 338
Write Buffer 316
Primary/Main Memory DRAM 108
L2 Cache 130
Security Accelerators (Crypto Acceleration) 156
MMU 308
Translation Lookaside Buffer 340
System Interface 310
Low-Latency Interface 350
Instruction Dispatch Unit 304
Instruction Fetch Unit 306
Instruction Cache 152
EJTAG 330
Coherent Memory Bus 144
Low-Latency Memory 118
Low Latency Memory Bus 158
Low-Latency Controller 160

FIG. 3



LLM LOAD/STORE INSTRUCTION FORMAT

FIG. 4

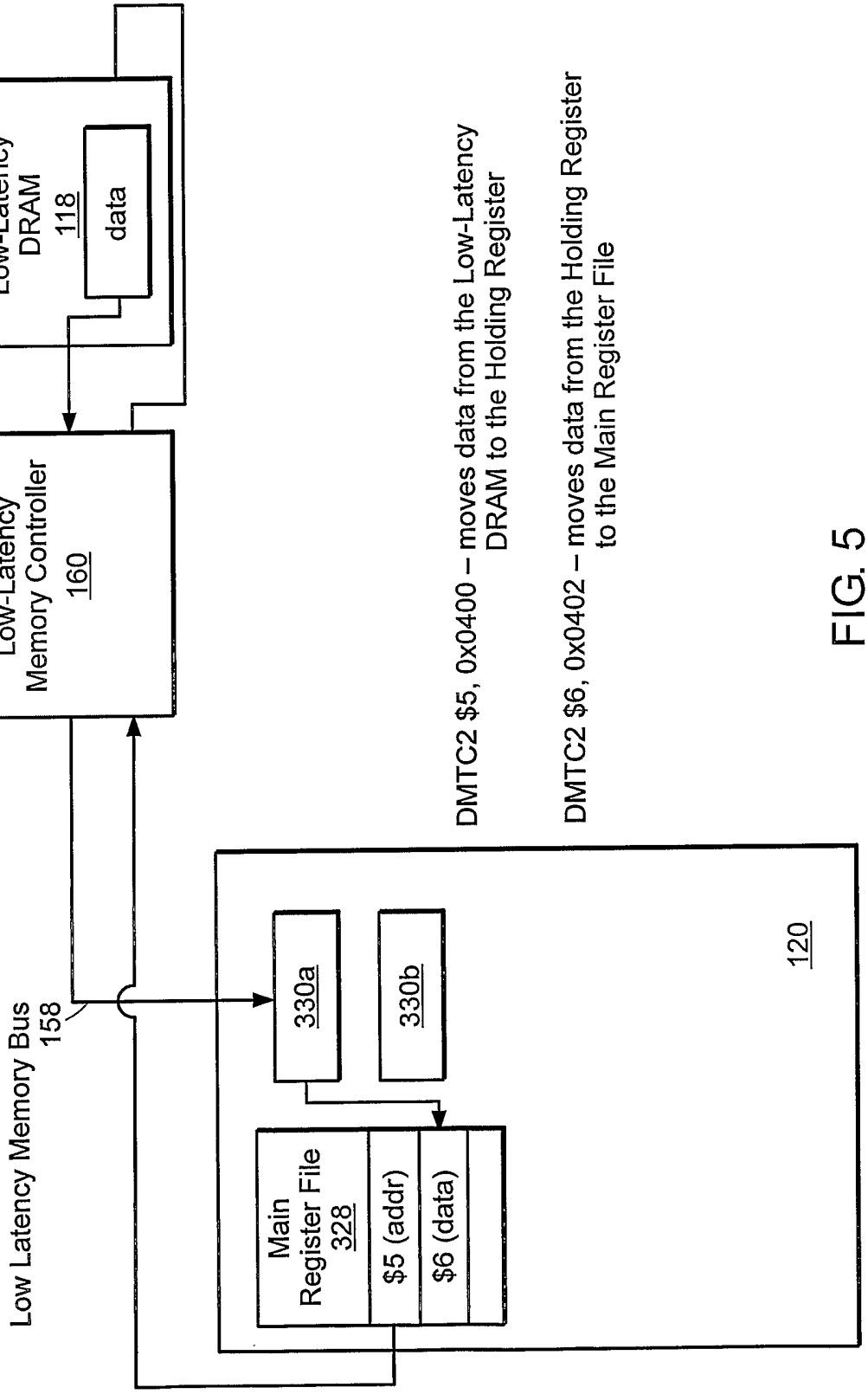

Low-Latency DRAM 118
data
Low-Latency Memory Controller 160
Low Latency Memory Bus 158
330a
330b
Main Register File 328
$5 (addr)
$6 (data)
120
DMTC2 $5, 0x0400 – moves data from the Low-Latency DRAM to the Holding Register
DMTC2 $6, 0x0402 – moves data from the Holding Register to the Main Register File

FIG. 5

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US05/23593

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) : F02B 3/00; F02F 3/26

US CL : 123/305, 193.6, 276, 661

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)

U.S. : 123/305, 193.6, 276, 661, 260, 261, 262, 279, 280, 285, 294, 295, 298

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

NONE

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

NONE

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 6,588,396 B1 (CLEARY et al) 08 July 2003 (08.09.2003).	1-7
A	US 6,494,178 B1 (CLEARY et al) 17 December 2002 (17.12.2002).	1-7
A	US 6,386,177 B2 (URUSHIHARA et al) 14 May 2002 (14.05.2002).	1-7

☐ 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

"B" earlier application or patent 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

22 September 2005 (22.09.2005)

Date of mailing of the international search report

17 NOV 2005

Name and mailing address of the ISA/US

Mail Stop PCT, Attn: ISA/US
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450

Facsimile No. (571) 273-3201

Authorized officer

For T. M. Argenbright

Telephone No. 571-272-3700