US 20050166006A1

(19) **United States**

(12) **Patent Application Publication**

(10) **Pub. No.: US 2005/0166006 A1**

(43) **Pub. Date: Jul. 28, 2005**

Talbot et al.

(54) **SYSTEM INCLUDING A HOST CONNECTED SERIALLY IN A CHAIN TO ONE OR MORE MEMORY MODULES THAT INCLUDE A CACHE**

(75) Inventors: **Gerald R. Talbot**, Concord, MA (US); **Frederick D. Weber**, San Jose, CA (US); **Shwetal A. Patel**, San Jose, CA (US)

Correspondence Address:
INGRASSIA FISHER & LORENZ, P.C.
7150 E. CAMELBACK, STE. 325
SCOTTSDALE, AZ 85251 (US)

(73) Assignee: **Advanced Micro Devices, Inc.**

(21) Appl. No.: **10/842,298**

(22) Filed: **May 10, 2004**

Related U.S. Application Data

(60) Provisional application No. 60/470,078, filed on May 13, 2003.

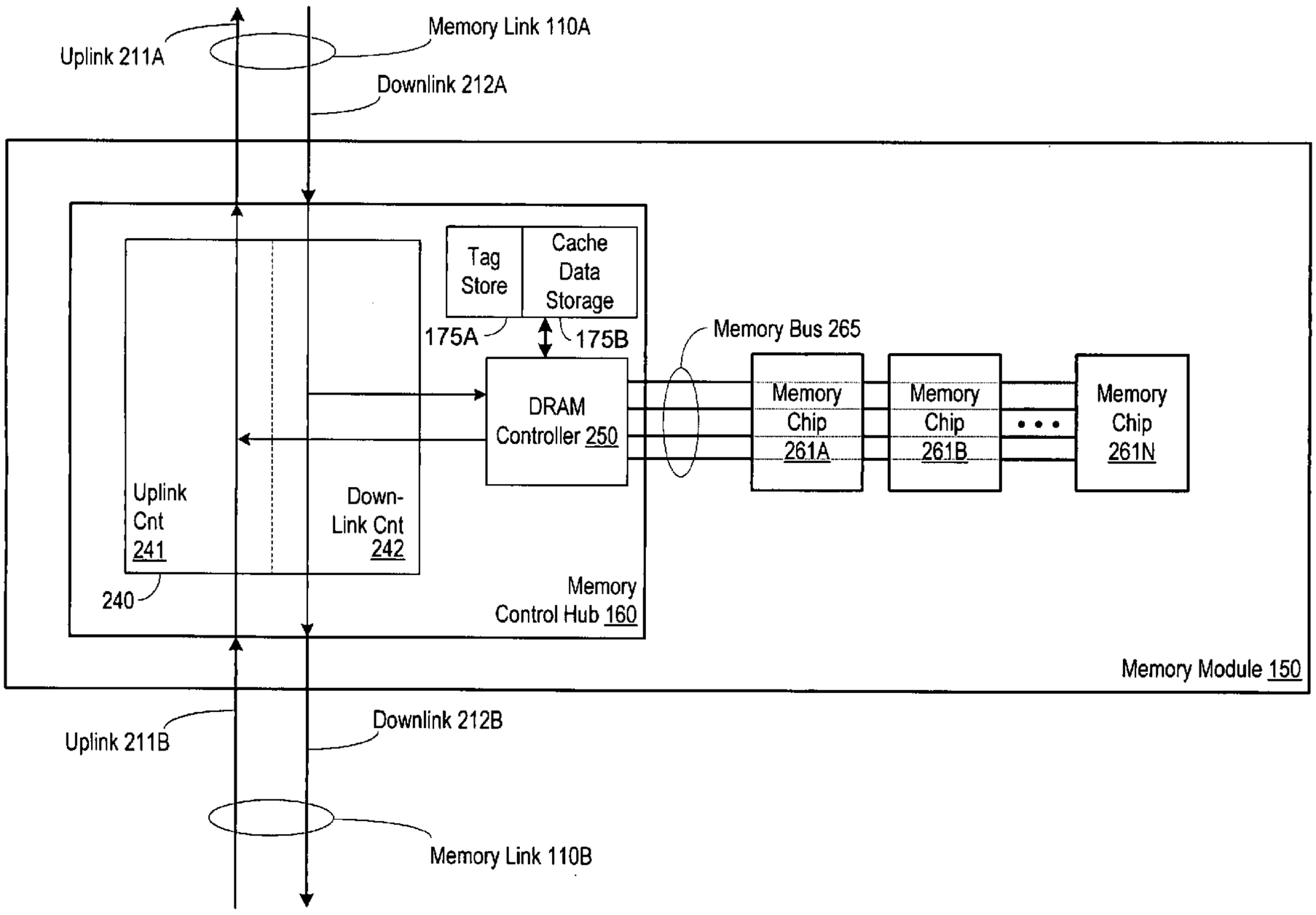
Publication Classification

(51) **Int. Cl.**[7] **G06F 12/00**

(52) **U.S. Cl.** **711/105**; 711/118; 711/154

(57) **ABSTRACT**

A system including a host coupled to a serially connected chain of memory modules. In one embodiment, at least one of the memory modules includes a cache for storing data stored in a system memory.



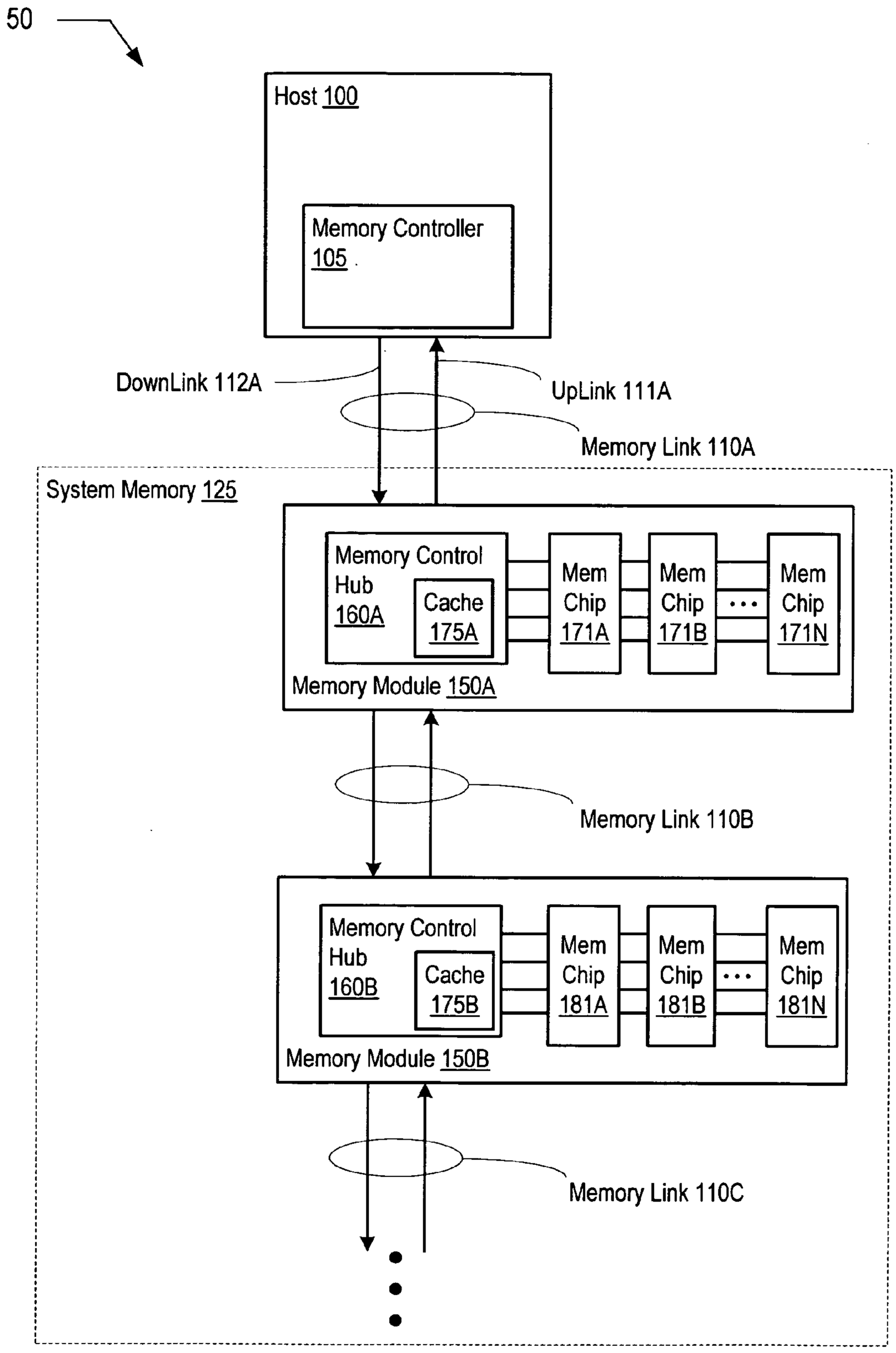

50
Host 100
Memory Controller 105
DownLink 112A
UpLink 111A
Memory Link 110A
System Memory 125
Memory Control Hub 160A
Cache 175A
Mem Chip 171A
Mem Chip 171B
Mem Chip 171N
Memory Module 150A
Memory Link 110B
Memory Control Hub 160B
Cache 175B
Mem Chip 181A
Mem Chip 181B
Mem Chip 181N
Memory Module 150B
Memory Link 110C

FIG. 1

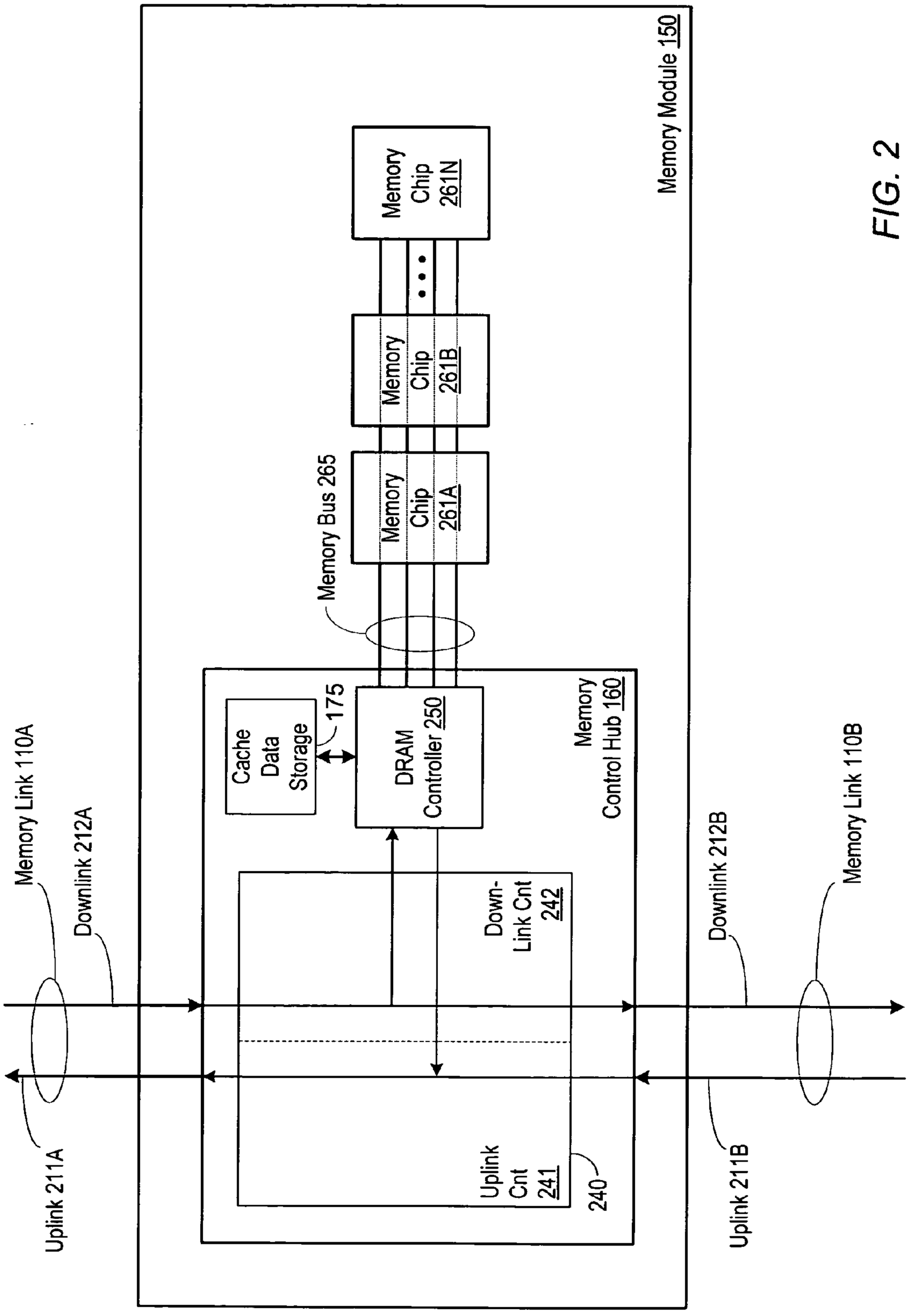

Memory Module 150
Memory Chip 261N
Memory Chip 261B
Memory Chip 261A
Memory Bus 265
Cache Data Storage
175
DRAM Controller 250
Memory Control Hub 160
Down-Link Cnt 242
Uplink Cnt 241
240
Memory Link 110A
Downlink 212A
Uplink 211A
Downlink 212B
Memory Link 110B
Uplink 211B

FIG. 2

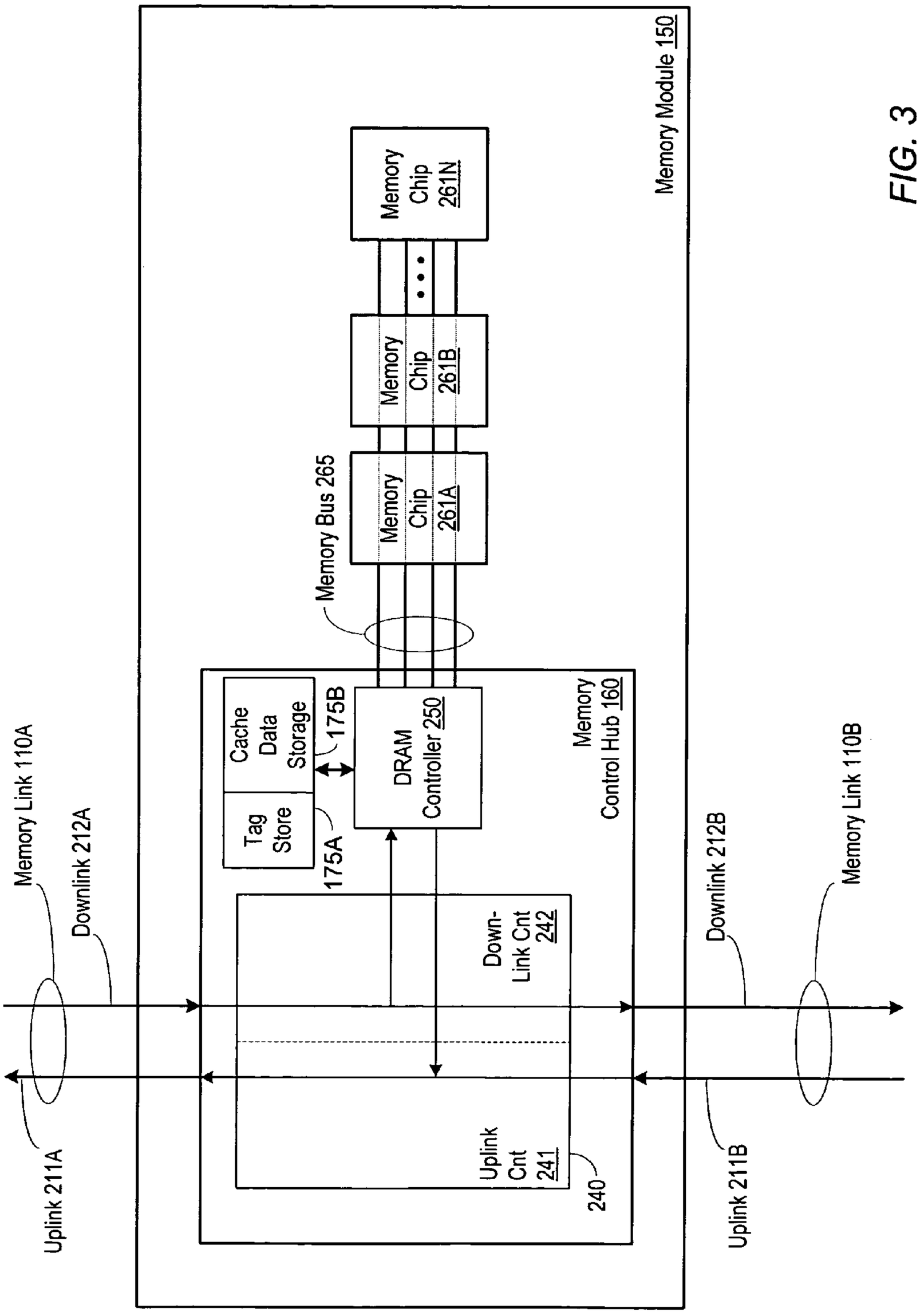

Memory Module 150
Memory Chip 261N
Memory Chip 261B
Memory Chip 261A
Memory Bus 265
Cache Data Storage
175B
Tag Store
175A
DRAM Controller 250
Memory Control Hub 160
Down-Link Cnt 242
Uplink Cnt 241
240
Memory Link 110A
Downlink 212A
Uplink 211A
Downlink 212B
Memory Link 110B
Uplink 211B

FIG. 3

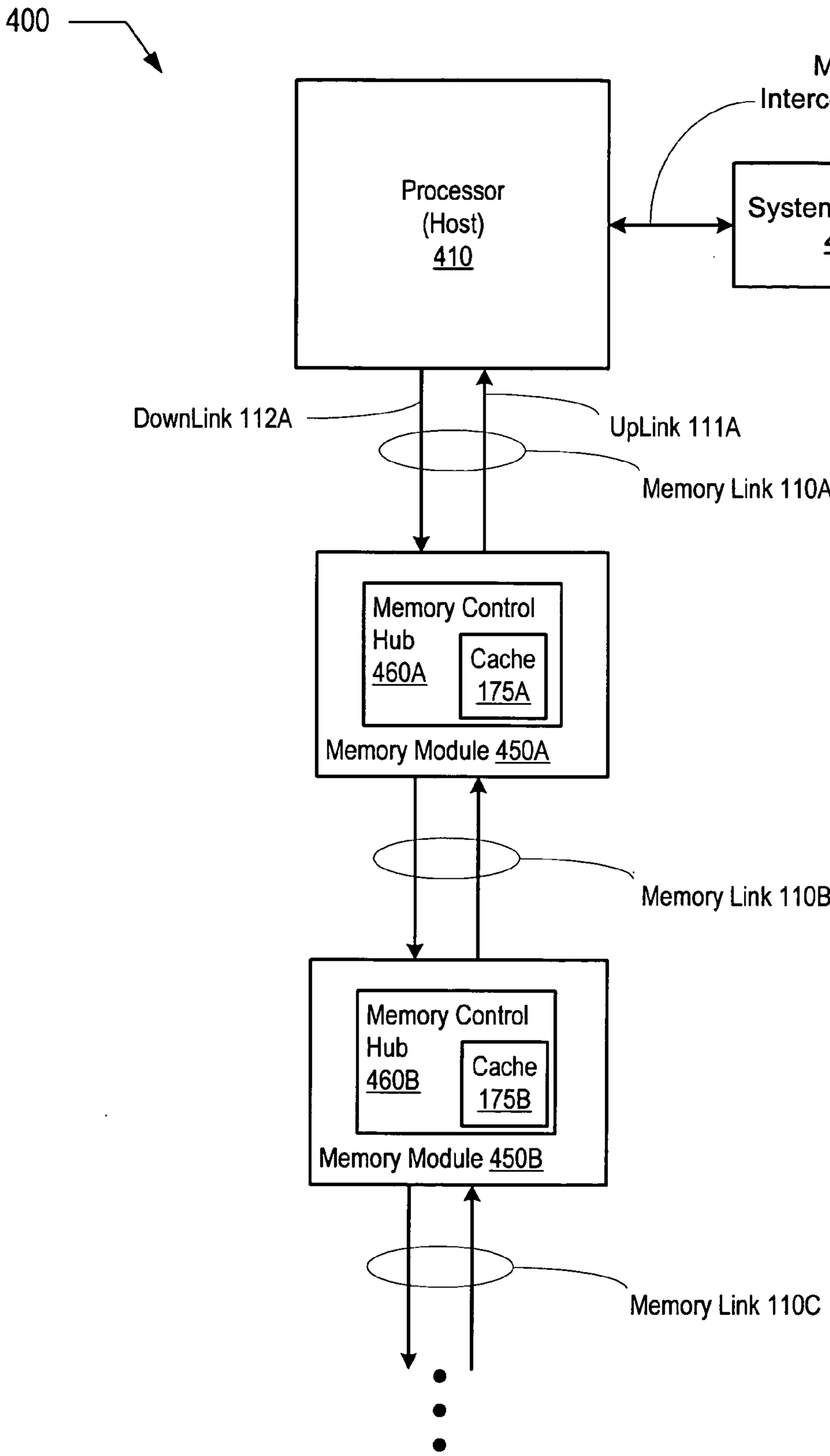

400
Processor
(Host)
410
Memory
Interconnect 430
System Memory
425
DownLink 112A
UpLink 111A
Memory Link 110A
Memory Control
Hub
460A
Cache
175A
Memory Module 450A
Memory Link 110B
Memory Control
Hub
460B
Cache
175B
Memory Module 450B
Memory Link 110C

FIG. 4

Memory Read Packet 525

BIT TIME	15	8	7	6	5	4	3	2	1	0
0		Tag [7:0]		Reserved		Command Code				
1		Address [39:32]		Reserved		Length				
2		Address [31:16]								
3		Address [15:3]						RSV	Priority	
4		CRC [15:0]								
5		CRC [31:16]								

FIG. 5

Cache Read Packet 675

BIT TIME	15	8	7	5	4	0
0		Tag [7:0]		RSV	Command Code	Length
1		Cache [25:16]				
2		Cache [15:0]				
3		CRC [15:0]				
4		CRC [31:16]				

FIG. 6

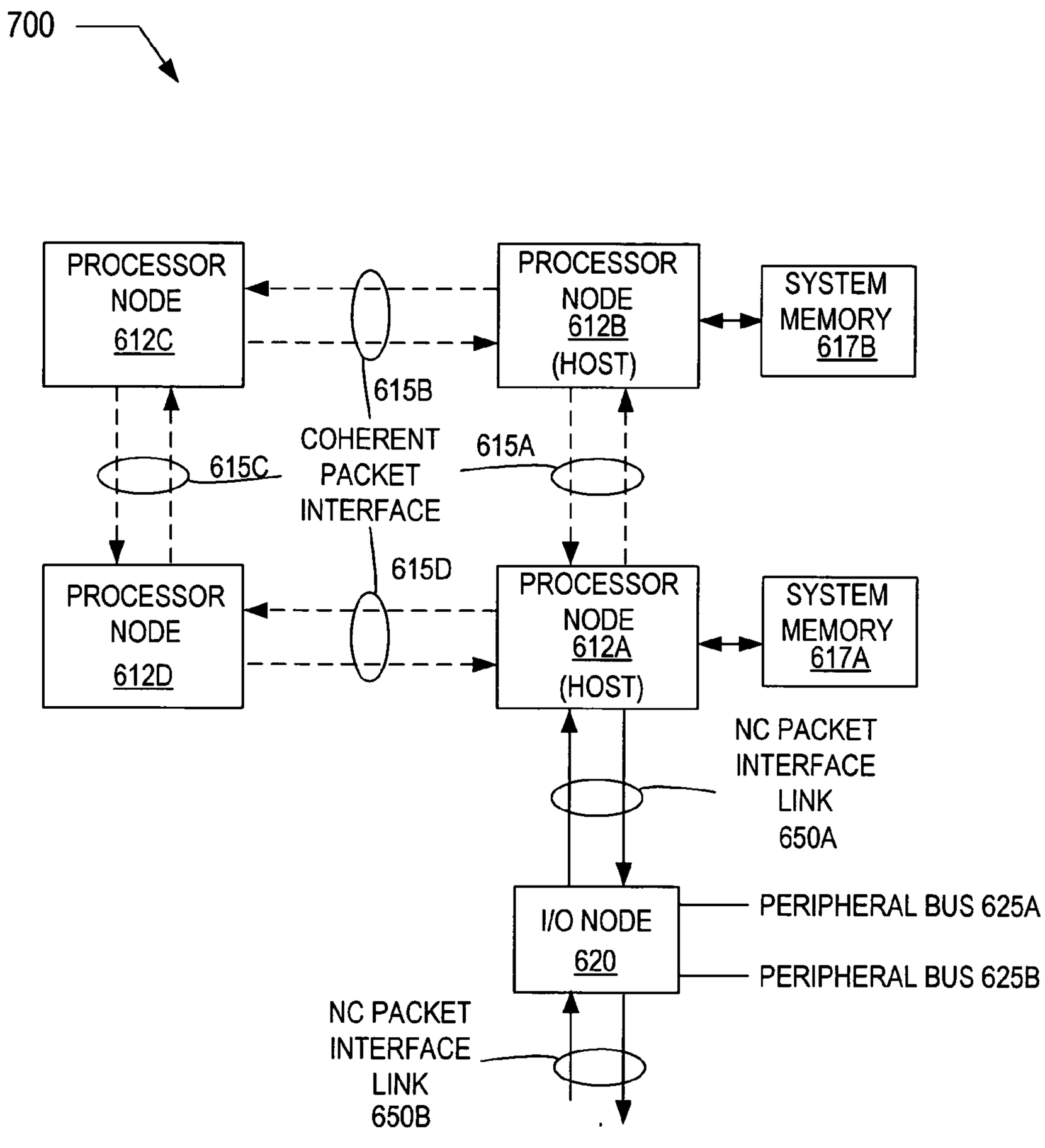

700
PROCESSOR NODE 612C
PROCESSOR NODE 612B (HOST)
SYSTEM MEMORY 617B
615B
COHERENT PACKET INTERFACE
615A
615C
615D
PROCESSOR NODE 612D
PROCESSOR NODE 612A (HOST)
SYSTEM MEMORY 617A
NC PACKET INTERFACE LINK 650A
I/O NODE 620
PERIPHERAL BUS 625A
PERIPHERAL BUS 625B
NC PACKET INTERFACE LINK 650B

FIG. 7

SYSTEM INCLUDING A HOST CONNECTED SERIALLY IN A CHAIN TO ONE OR MORE MEMORY MODULES THAT INCLUDE A CACHE

[0001] This application claims the benefit of U.S. Provisional Application No. 60/470,078 filed May 13, 2003.

BACKGROUND OF THE INVENTION

[0002] 1. Field of the Invention

[0003] This invention relates to computer system memory and, more particularly, to memory module configurations and the memory subsystem topology.

[0004] 2. Description of the Related Art

[0005] Many computer systems employ a main system memory that may be configured dependent upon the needs of an end user. In such systems, a motherboard or system board may include a number of memory expansion sockets. One or more small circuit boards, referred to as memory modules, may be inserted into the sockets as needed to increase the memory capacity of the computer system. Each of the memory modules typically includes multiple memory devices that provide a given amount of memory capacity. The memory devices are usually implemented using some type of dynamic random access memory (DRAM). Some examples of DRAM types include synchronous DRAM (SDRAM) as well as the various types of double data rate SDRAM (DDR SDRAM).

[0006] In conventional computer systems, the memory modules are connected to a memory/DRAM controller via a memory bus that includes address, control (including clock signals), and data signals. In some computer systems, the address, control and data signals may be multiplexed and thus share the same sets of wires. In other computer systems, the address, control and data signals may use separate wires. In either case, each of the address and control signals are routed to each expansion socket such that the memory modules, when inserted, are connected in parallel to the memory/DRAM controller. There are several drawbacks to parallel bus arrangements. For example, depending on the topology of the interconnect, the maximum memory bus speed may be limited due to transmission line effects such as signal reflections. In some systems, the memory/DRAM controller may reside on the same integrated circuit (IC) chip as the system processor, while in other systems the memory/DRAM controller may reside in one IC (e.g., a Northbridge) of a chipset.

[0007] In addition, many conventional computers systems employ processors that use an external cache memory. In many such systems, the external cache memory interface uses a large number of address and data pins. Furthermore, increasing the size of the external cache may require a system redesign and so the external cache may not be configurable by an end user.

[0008] Although the operating speed of computer system processors continues to increase, the relative performance of the main system memory has not increased at the same rate. This may be due, at least in part, to the incremental improvement in the bandwidth of the memory bus architectures described above.

SUMMARY

[0009] Various embodiments of a system including one or more memory modules are disclosed. In one embodiment, a host is coupled to a serially connected chain of memory modules. At least one memory module includes a cache for storing data stored in a system memory.

[0010] In one specific implementation, the system memory may include a respective plurality of memory chips mounted on each memory module. In addition, each memory module may include a memory control hub including a controller configured to determine whether data associated with a received memory request is stored within the cache.

[0011] In another specific implementation, a given cache may include a storage for storing cache tags corresponding to the data stored within the given cache. In such an implementation, the controller is configured to access the cache in response to determining that the data associated with the received memory request is stored within the cache.

[0012] In another specific implementation, the host includes a memory controller configured to generate memory requests to the memory modules. The memory controller may include a storage for storing cache tags corresponding to data stored within each cache.

[0013] In yet another implementation, the memory control hub includes a controller that may be configured to access the memory chips in response to receiving a memory command having a memory address that matches a memory address associated with the memory chips. In addition, the controller may be configured to access the cache in response to receiving a memory command having a memory address that matches a memory address associated with the cache.

BRIEF DESCRIPTION OF THE DRAWINGS

[0014] **FIG. 1** is a block diagram of one embodiment of a system including a serially connected chain of system memory modules.

[0015] **FIG. 2** is a block diagram of one embodiment of a memory module of **FIG. 1** including one cache implementation.

[0016] **FIG. 3** is a block diagram of another embodiment of a memory module of **FIG. 1** including another cache implementation.

[0017] **FIG. 4** is a block diagram of one embodiment of a system including a serially connected chain of cache memory modules.

[0018] **FIG. 5** is a diagram of one embodiment of a memory read packet.

[0019] **FIG. 6** is a diagram of one embodiment of a cache memory read packet.

[0020] **FIG. 7** is a block diagram of one embodiment of a computer system.

[0021] While the invention is susceptible to various modifications and alternative forms, specific embodiments thereof are shown by way of example in the drawings and will herein be described in detail. It should be understood, however, that the drawings and detailed description thereto are not intended to limit the invention to the particular form disclosed, but on the contrary, the intention is to cover all modifications, equivalents, and alternatives falling within the spirit and scope of the present invention as defined by the

appended claims. Note, the headings are for organizational purposes only and are not meant to be used to limit or interpret the description or claims. Furthermore, note that the word "may" is used throughout this application in a permissive sense (i.e., having the potential to, being able to), not a mandatory sense (i.e., must). The term "include" and derivations thereof mean "including, but not limited to." The term "connected" means "directly or indirectly connected," and the term "coupled" means "directly or indirectly coupled."

DETAILED DESCRIPTION

[**0022**] Turning now to **FIG. 1**, a block diagram of one embodiment of a system including a serially connected chain of memory modules is shown. System **50** includes a host **100** coupled to a system memory **125** via a memory link 1I A. System **50** may be configured to operate as part of a computing device such as a computer system or server system, for example. System memory **125** includes a memory module **150**A coupled to a memory module **150**B via a memory link **110**B. Memory module **150**B is shown coupled to a memory link **110**C, which may be coupled to an additional memory module (not shown) as desired to form a serially connected chain of memory modules that is coupled to host **100**. It is noted that although two memory modules are shown in the chain, it is contemplated that one or more memory modules may be connected in this manner. It is further noted that components including a reference number followed by a reference letter may be referred to generally by the reference number alone. For example, when referring generally to all memory modules, reference may be made to memory module **150**.

[**0023**] In the illustrated embodiment, memory module **150**A includes a memory control hub **160**A, which is coupled to a plurality of memory devices that are designated memory chip **171**A through **171**N, where N may be any number, as desired. In one embodiment, memory control hub **160**A may be coupled to the memory chips via any type of memory interconnect. For example, in one embodiment, the memory interconnect may be a typical address, control and data bus configuration.

[**0024**] Similarly, memory module **150**B includes a memory control hub **160**B, which is coupled to a plurality of memory devices that are designated memory chip **181**A through **181**N, where N may be any number, as desired. In one embodiment, memory control hub **160**B may be coupled to the memory chips via any type of memory interconnect as described above. It is noted that each of memory chips **171**A through **171**N and **181**A through **181**N may be any type of memory device such as a memory device in the DRAM family of memory devices, for example.

[**0025**] In the illustrated embodiment, memory links **110**A-**110**C form a memory interconnect. In one embodiment, each of memory links **110**A-**110**C forms a point-to-point memory interconnect that is implemented as two sets of unidirectional lines. One set of unidirectional lines is referred to as a downlink and is configured to convey transactions away from host **100** in a downstream direction. The other set of unidirectional lines is referred to as an uplink and is configured to convey transactions toward host **100** in an upstream direction. In addition, in one embodiment, each set of unidirectional lines may be implemented using a plurality of differential signal pairs. In one embodiment, each memory link **110** includes an 18-bit downlink and a 16-bit uplink, where each bit is a differential signal pair. As will be described in greater detail below in conjunction with the descriptions of **FIG. 4** and **FIG. 5**, the memory interconnect formed by memory links **110** may be configured to convey packets.

[**0026**] In an alternative embodiment, each of memory links **110** may form a point-to-point memory interconnect that is implemented as one set of bi-directional lines. As such, transactions may flow both upstream and downstream on the set of bi-directional wires. In such an embodiment, the bi-directional lines may be implemented using a plurality of differential signal pairs. It is noted that in other embodiments, other signaling schemes may be used, such as multi-level signaling, for example.

[**0027**] Generally speaking, all transactions from host **100** flow downstream through all memory modules **150** on the downlink and all response transactions flow upstream from the responding memory module **150** through each upstream memory module **150** on the uplink. More particularly, in one embodiment, host **100** may request to retrieve or store data within system memory **125**. In response to host **100** making a request, memory controller **105** initiates a corresponding transaction such as a memory read transaction or a memory write transaction, for example. Memory controller **105** transmits the transaction to system memory **125** via memory link **110**A. In the illustrated embodiment, the transaction is received by memory control hub **160**A of memory module **150**A.

[**0028**] In response to receiving the transaction, memory control hub **160**A is configured to transmit the received transaction to memory module **150**B via memory link **110**B without decoding or modifying the transaction. This is referred to as forwarding the transaction downstream. Thus, each transaction received on a downlink by a given memory control hub **160** of a given memory module **150** is forwarded to the next memory module **150** in the chain that is coupled to the downlink without decoding the transaction. In one embodiment, decoding of the transaction may occur in parallel with the forwarding of the transaction. In other embodiments, the decoding of the transaction may occur after the transaction has been forwarded.

[**0029**] Likewise, if memory controller **105** initiates a read request transaction, for example, the memory module **150** having the memory location corresponding to the address in the request will respond with the requested data. The response will be transmitted on the memory module's uplink toward host **100**. If there are any intervening memory modules between the sending memory module and host **100**, the intervening memory module will forward the response transaction on its uplink to either host **100** or the next memory module in the chain in an upstream direction. In addition, when the responding memory module is ready to send the response, it may inject the response into a sequence of transactions that are being forwarded upstream on the uplink.

[**0030**] In one embodiment, memory controller **105** may be configured to make requests to system memory **125** without knowledge of which of memory modules **150**A and **150**B a particular address is associated. For example, each of memory modules **150** may be assigned a range of memory

addresses during a system configuration sequence. Each memory control hub **160** may include logic (not shown in **FIG. 1**) that may decode the address of an incoming request. Thus, a memory control hub **160** of a given memory module **150** may initiate a memory read cycle or memory write cycle to the memory chips on the given memory module **150** in response to decoding a memory request having an address that is in the address range assigned to the given memory module **150**. As will be described in greater detail below in conjunction with the descriptions of **FIG. 2** and **FIG. 3**, each memory control hub **160** may include a DRAM controller (not shown in **FIG. 1**) for initiating memory cycles to the memory chips to which it is connected.

[**0031**] In addition, in one embodiment, memory controller **105** may initiate a subsequent memory access request prior to receiving a response to a previous memory access request. In such an embodiment, memory controller **105** may keep track of outstanding requests and may thus process the responses in a different order than they were sent.

[**0032**] Further, in the illustrated embodiment, memory control hubs **160**A and **160**B include a cache memory designated **175**A and **175**B, respectively. Cache memories **175**A-B may each serve as a cache memory for data stored elsewhere in the computing system. For example, as will be described in greater detail below, cache memories **175**A-B may each serve as a cache memory for data stored within the respective memory chips of each memory module. Alternatively, cache memories **175**A-B may each serve as a cache memory for data stored within other memory modules in the chain that may be further from the host and thus have longer latencies. Likewise, in computing systems having multiple processor nodes, cache memories **175**A-B may serve as a cache memory for data stored in a remote processor node. In the embodiment described in conjunction with the description of **FIG. 3**, cache **175** includes storage for cache tags. However, in the embodiment described in conjunction with the description of **FIG. 2**, cache **175** does not include storage for cache tags. In such an embodiment, memory controller **105** may include cache tag storage (not shown). It is noted that although cache memories **175** are shown as part of memory control hub **160** (e.g., on the same device), it is contemplated that in other embodiments, cache memories **175** may be implemented on different devices than memory control hub **160**.

[**0033**] The Memory Interconnect

[**0034**] The memory interconnect includes one or more high-speed point-to-point memory links such as memory links **110**A-**110**C each including an uplink such as uplink **111**A and a downlink such as downlink **112**A, for example. As noted above, in one embodiment downlinks may be 18-bit links while uplinks may be 16-bit links. As such, an 18-bit downlink may include 16 control, address and data (CAD) signals, a busy signal and a Control (CTL) signal. A given uplink may include 16 control, address and data (CAD) signals. It is contemplated however, that in an alternative embodiment, an uplink such as uplink **211**A may also include a CTL signal.

[**0035**] In addition to the high-speed links, other signals may be provided to each memory module **150**. For example, in one embodiment, a reset signal, a power OK signal and a reference clock may be provided to each memory module **150** from host **100**. Further, other signals may be provided between each memory module. For example, as described above, a next memory module present signal may be provided between memory modules.

[**0036**] Generally speaking, the types of transactions conveyed on memory links **110** may be categorized into configuration and control transactions and memory transactions. In one embodiment, configuration and control transactions may be used to configure memory control hub **160**. For example, configuration and control transactions may be used to access configuration registers, assign a memory address range to a memory module or to assign a hub address to a memory control hub. Memory transactions may be used to access the memory locations within the memory chips (e.g., **171**A-**171**N . . . **181**A-**181**N). In addition, as described further below, certain memory transactions may be used to directly access cache **175**.

[**0037**] Accordingly, in one embodiment, there are two types of addressing supported: hub addressing and memory addressing. Using hub addressing, eight hub bits identify the specific memory control hub being accessed. In one embodiment, a hub address of FFh may be indicative of a broadcast to all memory control hubs. Using memory addressing, each hub decodes the upper portion of the address bits to determine which hub should accept the request and the lower portion to determine the memory location to be accessed. In one embodiment, there are 40 address bits, although it is contemplated that other numbers of address bits may be used as desired.

[**0038**] As described further below, in certain embodiments, there may be an additional memory addressing type supported. The additional memory addressing type may be used to specifically access a cache memory located on a given memory module. For example, the last four entries of table 1, below, illustrate some exemplary cache access command codes.

[**0039**] In one embodiment, each of the memory links is configured to convey the transactions using one or more packets. The packets include control and configuration packets and memory access packets, each of which may include a data payload depending on the type of command the packet carries. As such, the sets of wires that make up memory links **110** may be used to convey control, address and data.

[**0040**] The packets may be generally characterized by the following: Each packet includes a number of bit positions which convey a single bit of information. Each packet is divided into several bit times and during a given bit time, all of the bit positions of the packet are sampled. As such, the control information and data share the same wires of a given link (e.g., CAD wires). As will be described in greater detail below, in one embodiment, packets are multiples of bit pairs and the first bit-time of every packet is sampled at an even bit-time. Packets begin with a control header that may be either one or two bit-pairs in length. In one embodiment, the first five bits of the control header is the command code. Table 1 below illustrates the various types of packets and their associated command codes. It is noted however, that the actual codes shown in column one are for illustrative purposes and that other codes may be used for each given command. It is further noted that other commands and encodings may be used.

TABLE 1

Packet types and command codes

Code	Header Length (bit-times)	Command	Description	Direction	Normal Response	Address Type
00h	—	NOP	Null Operation/Idle State	Both	—	—
04h	2	AddrSet	Address Set	Down	AddrAck	Hub
05h	2	AddrAck	Address Acknowledge	Up	—	—
06h	2	Ack	Acknowledge	Up	—	—
07h	2	Nak	Not Acknowledge/Error	Up	—	—
08h	2	SRdResp	Short Read Response	Up	—	—
09h	2	LRdResp	Long Read Response	Up	—	—
0Ah	2	ConfigRd	Configuration Read	Down	RdResp	Hub
0Ch	2	ConfigWr	Configuration Write	Down	Ack	Hub
0Eh	2	DIMMCt1	DIMM Control	Down	Ack	Hub
10h	4	SMemRd	Short Memory Read	Down	RdResp/Ack	Memory
11h	4	LMemRd	Long Memory Read	Down	RdResp	Memory
12h	4	BlkMemWr	Block Memory Write	Down	Ack	Memory
13h	4	SbytMemWr	Short Byte Memory Write	Down	Ack	Memory
14h	4	LbytMemWr	Long Byte Memory Write	Down	Ack	Memory
15h	4	CacheDataRd	Cache Read	Down	RdResp	M/Cache
16h	4	CacheDataWr	Cache Write	Down	Ack	M/Cache
17h	4	CacheTagRd	Cache Tag Read	Down	RdResp	M/Cache
18h	4	CacheTagWr	Cache Tag Write	Down	Ack	M/Cache

[**0041**] Further, in one embodiment, packets (except NOP packets) are transmitted with an error detecting code (EDC). It is noted that in one embodiment, the EDC is a 32-bit cyclic redundancy code (CRC), although other embodiments may employ other EDC's as desired. Additionally, addresses are sent most significant bit-time first to speed decode within memory control hub **160** while data is sent least significant byte first. It is noted however, that other embodiments are contemplated in which the addresses may be sent least significant bit-time first and data my be sent most significant byte first. Packets may carry a payload of byte enables and/or data. Packets with no payload are referred to as header-only packets. In one embodiment, the size of the data short reads may be up to one half of a programmed cache line size. In addition, the size of the data for long reads and block writes may be up to the programmed cache line size. Further, the size of the data for byte writes may be a maximum of 64 bytes regardless of the cache line size setting.

[**0042**] In addition to the control header and command code information included within a packet, the CTL signal may be used to convey information about each packet. As illustrated in Table 2 below, some exemplary CTL encodings are shown.

TABLE 2

CTL encodings for downstream use

Even	Odd	Content of CAD
0	0	Data or Byte Enable Payload
1	1	Control Header
0	1	CRC for a Packet with Payload
1	0	CRC for a Header-Only Packet

[**0043**] Different values of CTL for the header and payload portions of a packet may provide enough information to allow header-only packets to be inserted within the payload of another packet. This may be useful for reducing the latency of read commands by allowing them to issue while a write packet is still being sent on the link. Table 3 illustrates an exemplary packet including a payload in tabular format. The packet in table 3 also shows that a header-only packet is inserted in the payload during bit times **4-7**. It is noted however, that other packet encodings are possible and contemplated. For example, in other embodiments, a portion of the CRC bits may be transmitted during each bit time.

TABLE 3

Packet with payload and header-only packet inserted within payload

Bit-time	CTL	CAD
0	1	Headerl bits [15:0]
1	1	Headeri bits [31:16]
2	0	Databits [15:0]
3	0	Databits [31:16]
4	1	Header2 bits [15:0]
5	1	Header2 bits [31:16]
6	1	CRC2 bits [15:0]
7	0	CRC2 bits [31:16]
8	0	Data bits [47:32]
9	0	Data bits [64:48]
10	0	CRC1 bits [15:0]
11	1	CRC1 bits [31:16]

[**0044**] Referring to **FIG. 2**, a block diagram of one embodiment of a memory module such as the memory module illustrated in **FIG. 1** is shown. Components that correspond to those shown in **FIG. 1** are numbered identically for clarity and simplicity. Memory module **150** includes a memory control hub **160** coupled to memory chips **261**A through **261**N via z memory bus **265**.

[**0045**] Memory control hub **160** includes a control unit **240** coupled to a DRAM controller **250**. DRAM controller **250** is coupled to memory chips **261**A-**261**N and to a cache memory **175**. Control unit **240** includes an uplink control

241 and a downlink control **242**. As noted above, memory bus **265** may be any type of memory interconnect. In the illustrated embodiment, memory control hub **160** is coupled to a memory link **110**A in an upstream direction and a memory link **110**B in a downstream direction. It is further noted that the frequency of operation of memory bus **265** may be independent of the frequency of operation of memory links **110**. It is noted that although cache memory **175** is shown as part of memory control hub **160**, in other embodiments, cache memory **175** may be separate from memory control hub **160**, but may still be included on the same memory module.

[**0046**] In the illustrated embodiment, uplink control unit **241** may be configured to receive and forward packets received from another memory module downstream. The receiving and forwarding of the upstream packets creates an upstream transaction sequence. In addition, uplink control unit **241** may be configured to inject packets that originate within memory module **150** into the transaction stream.

[**0047**] In the illustrated embodiment, downlink control unit **242** may be configured to receive packets that originate at the host and if a memory module is connected downstream, to forward those packets to the downstream memory module. In addition, downlink control unit **242** may be configured to copy and decode the packets. In one embodiment, if the packets include an address that is within the range of addresses assigned to memory module **150** and the packet is a memory access request, downlink control unit **242** may pass the command associated with the packet to DRAM controller **250**. However, if the packet is not a memory request, but is instead a configuration packet, downlink control unit **242** may pass the configuration command associated with the packet to the core logic of control unit **240** (not shown) for processing. It is noted that in one embodiment, if the packet does not include an address that is within the range of addresses assigned to memory module **150**, memory control hub **160** may drop or discard the packet if memory module **150** is the last memory module in the chain.

[**0048**] In one embodiment, memory control hub **160** is configured to receive a module present signal (not shown), which when activated by a downstream memory module, indicates to an upstream memory module that there is a downstream memory module present. In such an embodiment, if memory control hub **160** receives a transaction and no downstream memory module is determined to be present, memory control hub **160** may drop the transaction. In addition, if no downstream memory module is determined to be present, a memory control hub **160** may power down the downstream transmit and receive circuits; thereby reducing power consumption and possibly radiated emissions.

[**0049**] As mentioned above, in one implementation, cache memory **175** of **FIG. 2** is configured to cache frequently accessed data, whether that data is stored within memory chips **261**A-N or some other place within the system. Thus, depending on whether the requested data is stored within cache memory **175**, DRAM controller **250** is configured to initiate memory cycles to either to cache memory **175** or to memory chips **261**A-**261**N in response to memory commands received by memory control hub **160**. Alternatively, DRAM controller **250** may respond with a cache miss which may initiate a request to another memory in response to memory commands received by memory control hub **160**. In one embodiment, cache memory **175** may be implemented using memory devices that are typically used for cache memory in a processor. For example, the memory devices may be in the static RAM (SRAM) or fast SRAM (FSRAM) family of devices.

[**0050**] In the illustrated embodiment, cache memory **175** is configured to store cache data while memory controller **105** includes storage for cache tags corresponding to the cache data stored within cache memory **175**. In one implementation, memory controller **105** is configured to perform a tag lookup within a tag storage (not shown) prior to initiating a memory access request to system memory **125**. In doing so, memory controller **105** determines whether the data is located in cache memory **175** or not. In either case, the access times associated with the request may be planned for and the responses scheduled accordingly.

[**0051**] As described above, each memory module may be associated with a particular address space. Thus, when a memory request is received that has an address associated with memory module **150**, DRAM controller **250** is configured to generate read or write cycles to either cache **175** or memory chips **261**A-N. In one embodiment, the address space associated with memory chips **261**A-N may be different than the address space associated with cache memory **175**. For example, the address space associated with memory chips **261**A-N of all memory modules may be 00000000h through FFFFFFFFh, while the address space associated with of all cache memories may be 00000h through FFFFFh. To access the address space associated with memory chips **261**A-N a memory access command code is used and to access the address space associated with cache memory **175**, a cache access command code is used. Exemplary memory and cache read packets are described below in conjunction with the descriptions of **FIG. 5** and **FIG. 6**, respectively. Since the memory controller **105** has determined whether the data resides in cache memory **175** or in DRAM chips **261**A-N, memory controller **105** sends the access request using the correct addressing type.

[**0052**] In an alternative embodiment, the memory space associated with memory module **150** may include addresses associated with memory chips **261**A-N as well as addresses associated with cache memory **175**. For example, if the address space associated with memory module **150** includes addresses in the range 00000000h through 3FFFFFFF, the address space associated with cache memory **175** may be allocated to addresses 00000000h through 000FFFFFh and the remaining addresses may be allocated to memory chips **261**A-N. Thus, if a memory request having an address in the DRAM range is received, DRAM controller **250** may access memory chips **261**A-N. Conversely, if the received memory request has an address in the cache address range, DRAM controller **250** may access cache **175**. In such an alternative embodiment, the type of packet may be a standard memory read or write packet including the requested address.

[**0053**] In one embodiment, DRAM controller **250** may include memory control logic (not shown) that may provide support for ensuring that cached data is written back to memory chips **261**A-N. For example, DRAM controller **250** may provide a write back buffer and/or an eviction/victim buffer and support logic (not shown) for cache memory **175**. In such an embodiment, DRAM controller **250** may imple-

ment an eviction algorithm such as a least recently used (LRU) algorithm, for example, for evicting data from cache memory **175**. In one implementation, memory controller **105** may provide explicit write back instructions to DRAM controller **250**.

[**0054**] As noted above in the description of **FIG. 1**, cache memory **175** of **FIG. 2** may also be configured to cache frequently accessed data stored within a remote processor node (not shown in **FIG. 2**) or the memory chips of another downstream memory module. In such implementations, memory controller **105** may explicitly write data to cache memory **175** rather than writing to cache memory **175** as an artifact of writing to memory chips **261**A-N.

[**0055**] Turning to **FIG. 3**, a block diagram of another embodiment of a memory module such as the memory module of **FIG. 1** is shown. Components that correspond to those shown in **FIG. 1** are numbered identically for clarity and simplicity. Memory module **150** of **FIG. 3** includes a memory control hub **160** coupled to memory chips **261**A through **261**N via a memory bus **265**.

[**0056**] Memory control hub **160** includes a control unit **240** coupled to a DRAM controller **250**. DRAM controller **250** is coupled to memory chips **261**A-**261**N and to a cache memory **175**. The operation of the memory interconnect and aspects of the operation of the DRAM controller **250** of memory module **150** of **FIG. 3** are similar to the operation of memory module **150** Of **FIG. 2**. However, in contrast to memory control hub **160** of **FIG. 2**, memory control hub **160** of **FIG. 3** includes a cache memory **175** that includes a cache tag storage **175**A and a cache data storage **175**B. Differences in functionality are described further below.

[**0057**] In the illustrated embodiment, cache data storage **175**B is configured to cache frequently accessed data stored within memory chips **261**A-N. Likewise, cache tag storage **175**A is configured to store address tags corresponding to the data stored within cache data storage **175**B. Accordingly, data written to and read from memory chips **261**A-N may be stored within cache data storage **175**B. Thus, depending on whether the requested data is stored within cache memory **175**, DRAM controller **250** is configured to initiate memory cycles to either memory chips **261**A-**261**N or to cache data storage **175**B in response to memory commands received by memory control hub **160**. In one embodiment, cache memory **175**A and **175**B may be implemented using memory devices that are typically used for cache memory in a processor. For example, the memory devices may be in the static RAM (SRAM) or fast SRAM (FSRAM) family of devices. It is noted that although cache memory **175**A-B is shown as part of memory control hub **160** (e.g., on the same device), it is contemplated that in other embodiments, cache memory **175**A-B may be implemented as a separate device (e.g., on a different IC).

[**0058**] In one implementation, memory controller **105** is configured to initiate a memory access request to system memory **125**. As described above, each memory module may be associated with a particular address space. Thus, when a memory request is received that has an address associated with memory module **150**, using the tags stored within cache tag storage **175**A, DRAM controller **250** is configured to determine whether the requested data is stored within cache data storage **175**B. In the case of a read request, if the data is stored within cache data storage **175**B (cache hit), DRAM controller **250** is configured to generate read cycles to cache data storage **175**B. If, on the other hand, if the data is not stored within cache data storage **175**B (cache miss), DRAM controller **250** is configured to generate read cycles to memory chips **261**A-N.

[**0059**] Since the tags are stored within cache tag storage **175**A, memory controller **105** may not have any a priori knowledge of whether a given read request will hit in the cache or not. Since read response latencies for cache hits and misses is typically different, the latency of a given read request is unknown to memory controller **105**. To handle the unknown read response latency, memory controller **105** may include logic (not shown) that handles out-of-order read responses by tracking outstanding read responses.

[**0060**] It is noted that depending on the implementation, it may be necessary to ensure the correct ordering of other transactions. For example, it is important to ensure that write data (e.g., data that may be posted to cache data storage **175**B) followed by a read of that data is not overwritten by a subsequent write to the same location prior to the read. In one embodiment, this functionality may be implemented in memory controller **105** while in other embodiments this functionality may be implemented in DRAM controller **250**.

[**0061**] There are various ways to implement cache policies with respect to writing data. Thus, assuming that such transaction ordering protection is in place, irrespective of the location, if a write request is received that has an address associated with memory module **150**, using the tags stored within cache tag storage **175**A, DRAM controller **250** is configured to determine whether the requested data is stored within cache data storage **175**B. In one embodiment, if the data is not stored within cache data storage **175**B (cache miss), DRAM controller **250** may allocate a location within cache data storage **175**B for the write data and possibly evict data already present. As described above, DRAM controller may include logic (not shown), such as write back buffers or eviction/victim buffers to support operation of cache data storage**175**A and tag storage **175**B. DRAM controller **250** may write the received data into cache data storage **175**B. The evicted data and the newly written data may be written back to memory chips **261**A-N as determined by DRAM controller **250**. If the write data is stored within cache data storage **175**B (cache hit) and has not been flushed to memory chips **261**A-N, DRAM controller **250** may simply overwrite the data within cache data storage **175**B.

[**0062**] However, if the write ordering protection is not in place, and if the write data is stored within cache data storage **175**B (cache hit) and has not been flushed to memory chips **261**A-N, DRAM controller **250** may write the data back to memory chips **261**A-N. DRAM controller **250** may then write the received data into the locations within cache data storage **250**. Alternatively, DRAM controller **250** may move the dirty data already in cache data storage **175**B into a write back buffer to be written back to memory chips **261**A-N at some later point in time.

[**0063**] It is noted that to access the address space associated with memory chips **261**A-N a memory access command code is used. An exemplary memory read packet is described below in conjunction with the description of **FIG. 5**.

[**0064**] Turning now to **FIG. 4**, a block diagram of one embodiment of a system including a serially connected

chain of cache memory modules is shown. Components that correspond to those shown in **FIG. 1** are numbered identically for clarity and simplicity. System **400** includes a host processor **410** coupled to a cache memory module **450**A and a cache memory module **450**B via a memory link **110**A and a memory link **110**B, respectively. Host processor **410** is also coupled to a system memory **425** via a memory interconnect **430**. Cache memory module **450**B is shown coupled to a memory link **110**C, which may be coupled to an additional cache memory module (not shown) as desired to form a serially connected chain of cache memory modules that is coupled to processor host **410**. It is noted that although two cache memory modules are shown in the chain, it is contemplated that any number of cache memory modules may be connected in this manner. It is further noted that components including a reference number followed by a reference letter may be referred to generally by the reference number alone. For example, when referring generally to all cache memory modules, reference may be made to cache memory module **450**.

[**0065**] In the illustrated embodiment, memory links **110**A-B and cache memory modules **450**A-B provide any level of external cache capability to processor **410**. For example, cache memory modules **45**GA-B may be an external L3 or L4 cache and may provide a means of providing a very large external cache memory. Each cache memory module includes a cache memory control hub **460** which includes a cache memory **475**. Processor **410** may include cache control logic (not shown) configured to generate cache transactions and to maintain cache coherency. Thus, the external cache memory provided by cache memory modules **450**A-B may be used to cache frequently used data. That data may be stored within system memory **425** or any other storage. It is noted that system memory **425** may be representative of any type of system memory such as the system memory including the memory modules described above in conjunction with the descriptions of **FIG. 1-3**, for example.

[**0066**] It is further noted that additional memory modules (not shown in **FIG. 4**) may be connected downstream from memory module **450**B. In such an implementation, the additional memory modules may be representative of the memory modules illustrated in **FIG. 1** through **FIG. 3** and may include memory chips. In one implementation, the additional memory modules may not include cache memories.

[**0067**] Cache memory transactions that are conveyed upon memory links **110**A-C may be representative of the transactions described above in conjunction with the description of **FIG. 1**, above.

[**0068**] **FIG. 5** and **FIG. 6** illustrate exemplary memory access packets that may be conveyed on memory links **110**A through **110**C of **FIG. 1**. Turning now to **FIG. 5**, a diagram of one embodiment of a memory read packet is shown. In the illustrated embodiment, memory read packet **525** is 16 bits wide and includes six bit times or three bit-pairs. During bit time zero, the five-bit command code (e.g., 10h or 11h) is conveyed in bit positions **0-4**. Bit positions **5-7** are reserved. An eight-bit tag is conveyed in bit positions **8-15**.

[**0069**] During bit time one, the length of the data that should be returned conveyed in bit positions **0-5**. In one embodiment, a value of 00h indicates no data, a value of 01h indicates two bit-pairs of data, a value of 02h indicates four bit-pairs of data, and so on. A zero length read results in an acknowledge packet (Ack) being returned to the requester. In one embodiment, a read of a half cache line or less may result in a short RdResp and a read of more than a half cache line may result in either a single long RdResp or two short RdResp. The cache line size may be programmed by software into the configuration registers of host **100** and each memory control hub**160**. Bit positions **6-7** are reserved. Address bits **39-32** of the requested location in DRAM are conveyed in bit positions **8-15**.

[**0070**] During bit time two, the address bits **31-16** of the requested location in DRAM are conveyed in bit positions **0-15** and during bit time **3**, the address bits **3-15** of the requested location in DRAM are conveyed in bit positions **3-15**. Also during bit time **3**, the packet priority is conveyed in bit positions **0-1**. In one embodiment, the priority may be indicative of the priority of the packet relative to other requests. For example, one priority may be to delay all requests with lower priority even if they are already in progress and to execute this request ahead of them. Bit position **2** is reserved. During bit times four and five, bits **0-15** and **16-31**, respectively, of a CRC are conveyed in bit positions **0-15**.

[**0071**] Referring to **FIG. 6**, a diagram of one embodiment of a cache read packet is shown. In the illustrated embodiment, cache read packet **525** is 16 bits wide and includes 4 bit times or two bit-pairs. During bit time zero, the five-bit command code (e.g., 15h or 16h) is conveyed in bit positions **0-4**. Bit positions **5-7** are reserved. An eight-bit tag is conveyed in bit positions **8-15**.

[**0072**] During bit time one, the length of the data that should be returned is conveyed in bit positions **0-5**. In one embodiment, a value of 00h indicates no data, a value of 01h indicates two bit-pairs of data, a value of 02h indicates four bit-pairs of data, and so on. A zero length read results in an acknowledge packet (Ack) being returned to the requestor. In one embodiment, a read of a half cache line or less may result in a short RdResp and a read of more than a half cache line may result in either a single long RdResp or two short RdResp. The cache line size may be programmed by software into the configuration registers of host **100** and each memory control hub**160**. Address bits **25-16** of the requested location in cache are conveyed in bit positions **6-15**.

[**0073**] During bit time two, the address bits **15-0** of the requested location in cache **175** are conveyed in bit positions **7-16**. During bit times three and four, bits **0-15** and **16-31**, respectively, of a CRC are conveyed in bit positions **0-15**.

[**0074**] It is noted that although only two types of packets were shown, other types of packets, which may correspond to the command codes listed in table 3 are contemplated. It is further noted that the illustrated packets are for discussion purposes only and that although the various fields of the exemplary packets are shown having a particular number of bits, it is contemplated that in other embodiments, the various fields of the each packet may include other numbers of bits as desired.

[**0075**] **FIG. 7** is a block diagram of one embodiment of a computer system. Computer system **700** includes processor nodes **612**A-**612**D each interconnected by coherent packet interface links **615**A-D. Each link of coherent packet interface **615** may form a high-speed point-to-point link. Pro-

cessor nodes **612**A-D may each include one or more processors. Computer system **700** also includes an I/O node **620** which is coupled to processor node **612**A via a non-coherent packet interface **650**A. I/O node **620** may be connected to another I/O node (not shown) in a chain topology for example, by non-coherent packet interface **650**B. Processor nodes **612**A is illustrated as a host node and may include a host bridge for communicating with I/O node **620** via NC packet interface **650**A. Processor nodes **612**B-D may also include host bridges for communication with other I/O nodes (not shown). The non-coherent packet interface links formed by NC packet interface **650**A-B may also be referred to as point-to-point links. I/O node **620** is connected to a pair of peripheral buses **625**A-B.

[**0076**] **FIG. 7** further illustrates respective system memories (e.g., **617**A and **617**B) coupled to processor nodes **612**A and **612**B. In the illustrated embodiment, processor node **612**A and **612**B are each illustrative of a host as shown in **FIG. 1**, and each system memory **617** may be implemented in the configuration described in conjunction with the descriptions of **FIG. 2** and **FIG. 3** above. Further, the interconnects between each of processor nodes **612**A and **612**B and their respective system memories **617** may be reflective of the memory interconnect including memory link **110**C described above in **FIG. 1** through **FIG. 6**. It is noted that in other embodiments, other numbers of processor nodes may be used. Further, it is contemplated that each of processor nodes **612**C and **612**D may be similarly connected to a respective system memory such as system memory **617**, for example. As such, a cache memory of a memory control hub located within system memory **617**A may cache data stored within either of system memories **617**A or **617**B, for example.

[**0077**] In the illustrated embodiment, each link of coherent packet interface **615** is implemented as sets of unidirectional lines (e.g. lines **615**A are used to transmit packets from processing node **612**A to processing node **612**B and lines **615**B are used to transmit packets from processing node **612**B to processing node **612**C). Other sets of lines **615**C-D are used to transmit packets between other processing nodes as illustrated in **FIG. 7**. The coherent packet interface **615** may be operated in a cache coherent fashion for communication between processing nodes ("the coherent link"). Further, non-coherent packet interface **650** may be operated in a non-coherent fashion for communication between I/O nodes and between I/O nodes and a host bridge such as the host bridge of processor node **612**A ("the non-coherent link"). The interconnection of two or more nodes via coherent links may be referred to as a "coherent fabric". Similarly, the interconnection of two or more nodes via non-coherent links may be referred to as a "non-coherent fabric". It is noted that a packet to be transmitted from one processing node to another may pass through one or more intermediate nodes. For example, a packet transmitted by processing node **612**A to processing node **612**C may pass through either processing node **612**B or processing node **612**D as shown in **FIG. 7**. Any suitable routing algorithm may be used. Other embodiments of computer system **700** may include more or fewer processing nodes than the embodiment shown in **FIG. 6**.

[**0078**] One example of a packet interface such as non-coherent packet interface **650** may be compatible with HyperTransport™ technology. Peripheral buses **625**A and **625**B are illustrative of a common peripheral bus such as a peripheral component interconnect (PCI) bus. It is understood, however, that other types of buses may be used.

[**0079**] It is further noted that other computer system configurations are possible and contemplated. For example, it is contemplated that the system memory configuration described above may be used in conjunction with a computer system employing a processor chipset that includes a Northbridge. In such an embodiment, a memory controller within the Northbridge may serve as the host.

[**0080**] Although the embodiments above have been described in considerable detail, numerous variations and modifications will become apparent to those skilled in the art once the above disclosure is fully appreciated. It is intended that the following claims be interpreted to embrace all such variations and modifications.

What is claimed is:

1. A system comprising:

a host; and

one or more memory modules coupled serially in a chain to said host;

wherein at least one of said memory modules includes a cache for storing data stored in a system memory.

2. The system as recited in claim 1, wherein said system memory includes a respective plurality of memory chips mounted on given memory modules of said one or more memory modules.

3. The system as recited in claim 2, wherein each memory module includes a memory control hub including a controller configured to determine whether data associated with a received memory request is stored within said cache.

4. The system as recited in claim 3, wherein a given cache includes a storage for storing cache tags corresponding to said data stored within said given cache.

5. The system as recited in claim 4, wherein said controller is configured to access said cache in response to determining that said data associated with a received memory request is stored within said cache.

6. The system as recited in claim 2, wherein said host includes a memory controller configured to generate memory requests to said one or more memory modules.

7. The system as recited in claim 6, wherein said memory controller includes a storage for storing cache tags corresponding to said data stored within each cache.

8. The system as recited in claim 7, wherein said memory control hub includes a controller configured to access said plurality of memory chips in response to receiving a memory command having a memory address that matches a memory address associated with said plurality of memory chips.

9. The system as recited in claim 8, wherein said controller is further configured to access said cache in response to receiving a memory command having a memory address that matches a memory address associated with said cache.

10. The system as recited in claim 6, wherein said memory controller is further configured to issue a memory read request transaction prior to receiving a response to a previous memory read request transaction.

11. The system as recited in claim 1, wherein said one or more memory modules is coupled serially in a chain to said host via a plurality of memory links, wherein each memory

link includes an uplink for conveying transactions toward said host and a downlink for conveying transactions originating at said host to a next memory module in said chain.

12. The system as recited in claim 11, wherein said uplink and said downlink are each a unidirectional link including a plurality of signals configured to convey transactions using packets that include control and configuration packets and memory access packets, wherein at least a portion of packets include control, address and data information, and wherein said control, address and data information share the same wires of a given link.

13. The system as recited in claim 12, wherein each of said plurality of signals of each of said unidirectional links uses differential signaling.

14. The system as recited in claim 13, wherein said packets are conveyed on said uplink and said downlink using **16** bit positions to convey said control, address and data information.

15. A memory module comprising:

a cache configured to store data stored in a system memory;

a memory control hub coupled to control access to said storage;

wherein said memory control hub is configured to receive memory transactions originating at a processor via a first downlink and to transmit said memory transactions via a second downlink to another memory module independent of decoding said transactions.

16. The memory module as recited in claim 15, wherein said system memory includes a respective plurality of memory chips mounted said memory module.

17. The memory module as recited in claim 16, wherein said memory control hub includes a controller configured to determine whether data associated with a received memory request is stored within said cache.

18. The memory module as recited in claim 17, wherein said cache is further configured to store cache tags corresponding to said data stored within said cache.

19. The memory module as recited in claim 18, wherein said controller is configured to access said cache in response to determining that said data associated with a received memory request is stored within said cache.

20. The memory module as recited in claim 16, wherein said processor includes a memory controller configured to generate memory requests to said memory module.

21. The memory module as recited in claim 20, wherein said memory controller includes a storage for storing cache tags corresponding to data stored within said cache.

22. The memory module as recited in claim 21, wherein said memory control hub includes a controller configured to access said plurality of memory chips in response to receiving a memory command having a memory address that matches a memory address associated with said plurality of memory chips.

23. The memory module as recited in claim 22, wherein said controller is further configured to access said cache in response to receiving a memory command having a memory address that matches a memory address associated with said cache.

* * * * *