



- (51) **International Patent Classification:** Not classified
- (21) **International Application Number:** PCT/IB2013/055480
- (22) **International Filing Date:** 4 July 2013 (04.07.2013)
- (25) **Filing Language:** English
- (26) **Publication Language:** English
- (30) **Priority Data:**
2012902870 5 July 2012 (05.07.2012) AU
2012905522 17 December 2012 (17.12.2012) AU
2013902135 9 June 2013 (09.06.2013) AU
- (71) **Applicant:** KELSON, Ron [AU/MT]; Synaptic Laboratories Limited, Po Box 5, Nadur, Gozo, NDR-1000 (MT).
- (71) **Applicant (for all designated States except US):** SYN-APTIC LABORATORIES LIMITED; Synaptic Laboratories Limited, Po Box 5, Nadur, Gozo, NDR-1000 (MT).
- (72) **Inventor:** GITTINS, Benjamin; Synaptic Laboratories Limited, Po Box 5, Nadur, Gozo, NDR-1000 (MT).
- (81) **Designated States** (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM,

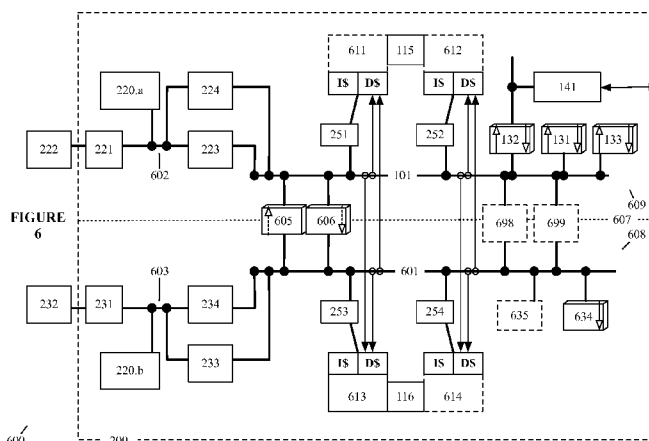
AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

- (84) **Designated States** (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— as to the identity of the inventor (Rule 4.17(i))

[Continued on next page]

(54) **Title:** COMPUTER ARCHITECTURE

(57) **Abstract:** A computing device [600] for performing real-time and mixed criticality tasks has at least one sub-computing device [608, 609]. Each sub-computing device has at least one bus, [101, 601] at least one bus master [605, 606, 611, 612, 613, 614], at least one memory store, [222, 232]. At least one of the at least one sub-computing devices [608, 609] has at least two bus masters [605, 606, 611, 612, 613, 614], and a means [101, 601, 605, 606] to enable or disable at least one of the at least two bus masters [605, 606, 611, 612, 613, 614] from issuing memory transfer requests onto the bus [101, 601] without resetting the at least one of the at least two bus masters [605, 606, 611, 612, 613, 614]. In another aspect, the computing device [600] has at least two cache modules [223, 224, 233, 234] arranged in parallel. The first cache module [223, 224, 233, 234] has an input address space of at least 1 kilobyte in length. The computing device [600] has at least one bus master [611, 612, 613, 614]. A first bus master [611, 612, 613, 614] can perform memory transfer requests with both the first cache module [223, 224, 233, 234] and another cache module [223, 224, 233, 234]. The computing device [600] has at least one memory store [222, 232]. A first contiguous subset of the input address space of at least 1 kilobyte in length of a first memory store [222, 232] is bijectively mapped as cacheable with at least a contiguous subset of the input address space of the first cache module [223, 224, 233, 234], and bijectively mapped as cacheable with at least a subset of the output address

[Continued on next page]

**Published:**

- *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

space of the first bus master [611, 612, 613, 614]. In another aspect a computing device [1200] has $N > 1$ sub-computing devices [1220, 1240, 1260]. Each sub-computing device [1220, 1240, 1260] has at least one bus [1221, 1241, 1261], at least one bus master [1230, 1231, 1291, 1292, 1243, 1263, 1295, 1296] that is a processor [1230, 1231, 1243, 1263], and has the bus slave interface of at least one memory store [1293, 1294, 1281] connected to one of the busses [1221, 1241, 1261]. The computing device has at least one unidirectional bus bridge [1291, 1292, 1295, 1296] that is connected to one of the sub-computing devices [1220, 1240, 1260]. The computing device has at least one memory store [1293, 1294, 1281] that is connected to two of the sub-computing devices. Specifically, X of the N sub-computing devices [1220, 1240, 1260] are directly connected to a common bus [1221] by a corresponding bus bridge [1291, 1292, 1295, 1296] where the value of X is $2 \leq X \leq N$. A first set of two of the sub-computing devices [1220, 1240, 1260] are connected to each other by a first memory store [1293, 1294, 1281].

COMPUTER ARCHITECTURE

Field of the invention

The present invention relates to multi-bus master computer architectures and is particularly applicable to real-time and mixed-criticality computing.

Background of the invention

Throughout this specification, including the claims:

A memory store (e.g. 222 of figure 2) coupled with a memory controller (e.g. 221 of figure 2) may be described at a higher level of abstraction as a memory store;

A computing device is any device that has combinatorial and/or sequential logic, including but not limited to: a general purpose computer, a graphics processing unit, and a network router;

A bus master (e.g. 111 of figure 2) coupled with an address translation unit (e.g. 251 of figure 2) may be described at a higher level of abstraction as a bus master.

Figure 1 is a block schematic diagram illustrating one version of the European Space Agency's Next Generation Microprocessor (NGMP) 100 as described in detail in [1]. The modules illustrated in 100 are described in detail in [4]. The dotted-line rectangle 109 delineates the boundary between modules which are on-chip and modules which are off-chip. For example module 125 is off-chip and module 124 is on-chip.

A 128-bit wide Advanced Microcontroller Bus Architecture (AMBA) Advanced High-performance Bus (AHB) 101 running at 400 MHz is referred to as the processor bus. A 128-bit wide AMBA AHB (also referred to as just AHB in this text) 102 is running at 400 MHz and is referred to as the memory bus. A 32-bit wide AHB 103 running at 400 MHz is referred to as the master I/O bus. A 32-bit wide AHB 104 running at 400 MHz is referred to as the slave I/O bus. A 32-bit wide AMBA Advanced Peripheral Bus (APB) 105 running at 400 MHz is referred to as the APB peripheral bus. A 32-bit wide AHB 106 running at 400 MHz is referred to as the debug bus.

The four Aeroflex Gaisler LEON4T processor cores 111, 112, 113, 114 are connected as bus masters to the processor bus 101. In figure 1 the LEON4T processor cores 111, 112,

113, 114 employ triple modular redundant logic in their implementations (not illustrated) to provide resistance to single event upsets within a processor core.

A bidirectional 128-bit wide AHB to 32-bit wide AHB bridge 131 connects the processor bus 101 and master I/O bus 103. A bidirectional 128-bit wide AHB to 32-bit wide AHB bridge 132 connects the processor bus 101 and slave I/O bus 104. A bidirectional 128-bit wide AHB to 32-bit wide AHB bridge 133 connects the processor bus 101 and debug bus 106. A 128-bit wide AHB to 32-bit wide APB bridge 134 connects the slave I/O bus 104 and peripheral bus 105.

An off-chip 64-bit wide DDR2-800 memory store module 125 is connected to the chip 109. An off-chip 64-bit wide PC-133 memory store module 126 is connected to the chip 109. A forward error-correcting memory controller 124 is capable of switching between two exclusive modes of operation: driving either the DDR2 (125) or PC133 (126) off-chip memory store modules. Only one off-chip memory store module 125 or 126, can be connected to the chip 109 at any one time as the memory control, address and data pins are multiplexed between the two memory controllers in 124. An optional embedded SDRAM module (and tightly coupled forward error correcting memory controller) 123 may be implemented on-chip.

A level 2 cache module 121 is connected as a bus slave to the processor bus 101 and as a bus master to the memory bus 102. A subset of the input address space of the level 2 cache module is mapped against the entire accessible region of the physical address space of the memory store 125 (or memory store 126) in the usual way. The level 2 cache module can be accessed by the four processor cores 111, 112, 113, 114, and by all of the bus master peripherals which are connected (directly or indirectly via bridges 131, 132, 133) to the processor bus 101.

Each processor core 111, 112, 113, 114 has a level 1 instruction cache module (which is not illustrated in figure 3) configured with a 32-byte cache line. Each processor core 111, 112, 113, 114 has a level 1 data cache module (which is not illustrated in figure 3) configured with a 16-byte cache line, operating in write-through mode, with each write operation into the level 1 data cache resulting in a corresponding write operation into the

level 2 data cache 121. A subset of the input address space of each of the level 1 data and instruction cache modules is mapped against the entire accessible region of the physical address space of the memory store 125 (or 126), that is the internally contiguous input address space of the memory store, in the usual way. The level 1 data cache module, when run in *N*-way set associative mode of operation, supports “cache line locking” in which specific address locations can be prevented from being evicted. The floating point units 115 and 116 are shared by processor cores {111, 112} and {113, 114} respectively.

A memory scrubber unit 122 is connected as a bus master to the memory bus 102.

Memory scrubbing is a process that involves detecting and correcting (any) bit errors that occur in computer memory protected by error-correcting codes.

A Peripheral Component Interconnect (PCI) Master module 141, with I/O pins, is connected to the slave I/O bus 104, and to the peripheral bus 105. A universal

asynchronous receiver/transmitter (UART) module 142 with I/O pins, is connected to the peripheral bus 105. A general purpose I/O module 143 with I/O pins is connected to the peripheral bus 105. A timer module 144 is connected to the slave I/O bus 104. Aeroflex Gaisler's combined programmable read only memory (PROM), asynchronous static ram (SRAM), synchronous dynamic ram (SDRAM) and memory mapped I/O devices (I/O) module 145 is connected to the slave I/O bus 104. An external PROM module 146 is connected to the controller 145.

A PCI Direct Memory Access (DMA) controller 151 is connected to the master I/O bus 103 and to the peripheral bus 105. A PCI Target module 152 is connected to the master I/O bus 103 and to the peripheral bus 105 and has I/O pins. Label 153 illustrates a communications module that is compliant with the ESA SpaceWire standard, that is connected to the Master I/O bus 103, the peripheral bus 105 and has I/O pins. Label 154 illustrates a “High Speed Serial Link” (HSSL) module, that is connected to the master I/O bus 103, the peripheral bus 105 and has I/O pins.

An AHB status module 161 is connected to the peripheral bus 105 and monitors the processor bus 101. Various debug and monitoring modules (not illustrated), such as a Joint Test Action Group (JTAG) module, are connected to the debug bus 106.

The proposed NMGP architecture is intended for use in environments that have real-time requirements. However according to the Multicore Benchmarks of NGMP [3] performed by the Barcelona Supercomputing Centre, a prototype of the NGMP running a single task on one core of the quad-core architecture can experience up to 20x delays in the worst case due to the activity of other tasks running on the three other cores. In particular the report states: *“small changes in the other applications in the workload may significantly affect the execution of the application (up to 20x slowdown) This seriously compromise time composability.”*

Summary of the invention

In contrast with the NGMP architecture, in one aspect embodiments of the present invention provide a computing device for performing real-time and mixed criticality tasks comprising:

at least one sub-computing device, each sub-computing device comprising:

at least one bus;

at least one bus master; and

at least one memory store; and

in which at least one of the at least one sub-computing devices comprises:

at least two bus masters; and

a means to enable or disable at least one of the at least two bus masters from issuing memory transfer requests onto the bus without resetting the at least one of the at least two bus masters.

In another aspect embodiments of the present invention provide a computing device comprising:

at least two cache modules, a first cache module of the at least two cache modules having an internally contiguous input address space of at least 1 kilobyte in length, in which:

the output address space of the first cache module is not mapped to the input address space of any other of the at least two cache modules; and the output address space of any of the other at least two cache modules is not mapped to the input address space of first cache module;

at least one bus master, a first bus master having an contiguous output address space of at least 2 kilobyte in length, in which:

at least one bus master can perform memory transfer requests with both the first cache module and another cache module of the at least two cache modules;

at least one memory store comprising an internally contiguous input address space of at least 2 kilobytes, in which: a first contiguous subset of the internally contiguous input address space of at least 1 kilobyte in length of a first memory store is:

bijectionally mapped as cacheable with at least a contiguous subset of the internally contiguous input address space of the first cache module; and
bijectionally mapped as cacheable with at least a subset of the contiguous output address space of the first bus master.

In another aspect embodiments of the present invention provide A computing device for performing real-time and mixed criticality tasks comprising:

N sub-computing devices, where value of N is at least 2, each sub-computing device comprising:

at least one bus;

at least one bus master, in which at least one of the at least one bus masters is a processor core;

the bus slave interface of at least one memory store connected to one of the at least one busses and

at least one unidirectional bus bridge, for each unidirectional bus bridge, each bus interface of that unidirectional bus bridge is connected to the bus of a different one of the sub-computing devices;

at least one memory store, each memory store comprising at least two bus slave interfaces, for each memory store, each bus slave interface is connected to the bus of a different one of the sub-computing devices;

in which:

X of the N sub-computing devices are directly connected to a common bus by a corresponding bus bridge where the value of X is $2 \leq X \leq N$;

a first set of two of the sub-computing devices are connected to each other

by a first memory store comprising at least two bus slave interfaces.

In yet further aspects, embodiments of the present invention provide:

representations in a hardware description language;

5 processes emulating;

signals carrying representations in a hardware description language; and

machine readable substrates carrying representations in a hardware description language,

of each of the preceding aspects of the invention.

10

Further aspects of the present invention are set out in the claims appearing at the end of this specification.

Brief description of the drawings

15 For a better understanding of the invention, and to show how it may be carried into effect, embodiments of it are shown, by way of non-limiting example only, in the accompanying drawings. In the drawings:

figure 1 is a block schematic diagram illustrating one version of the European Space Agency's Next Generation Microprocessor (NGMP);

20 figure 2 is a high-level block schematic diagram partially illustrating an adaption of the NGMP microarchitecture of figure 1 with enhancements according to a preferred embodiment of the present invention;

figure 3 is a hybrid diagram which incorporates a block schematic diagram, a memory partitioning scheme illustrating partitioning of elements of physical memory and a memory partitioning scheme illustrating the partitioning of physical
25 cache module resources;

figure 4 is a hybrid diagram which incorporates a block schematic diagram illustrating four cache modules and two memory stores, a memory partitioning scheme illustrating partitioning of blocks of logical memory, the partitioning of
30 blocks of physical memory, and a memory address translation scheme that maps blocks of logical memory to blocks of physical memory;

figure 5 is a hardware block schematic of an example of a circuit 500 implementing an address translation unit of figures 2, 3 and 4;

figure 6 is a schematic block diagram illustrating an architecture which is an adaption of the NGMP microarchitecture of figure 1 with enhancements according to a preferred embodiment of the present invention;

figure 7 is a block schematic diagram illustrating an AMBA v2.0 AHB controller for a bus with 2 bus masters, and 2 bus slaves illustrated;

figure 8 is a block schematic diagram of the NGMP microarchitecture of figure 1 illustrating the 7 stage integer pipeline, level 1 data cache module, memory management unit (MMU), AHB bus master interface module and a first AHB bus slave interface module of the first LEON4T processor core connected to the processor bus 101;

figure 9 is a block schematic diagram that extends figure 8 and illustrates a portion of the microarchitecture of figure 6 according to a preferred embodiment of the present invention;

figure 10 is a flow-chart illustrating the steps in a dual-snoop process of figure 9 according to a preferred embodiment of the present invention;

figure 11 is a block schematic diagram of an architecture that enhances the microarchitecture of figure 8 to implement four independent processor busses according to a preferred embodiment of the present invention;

figure 12 is a schematic block diagram illustrating an architecture according to a preferred embodiment of the present invention;

figure 13 is a schematic block diagram illustrating an architecture according to a preferred embodiment of the present invention; and

figure 14 is a schematic block diagram illustrating an architecture according to a preferred embodiment of the present invention.

Description of preferred embodiments of the invention

Figure 2 is a high-level block schematic diagram partially illustrating an adaption of the NGMP microarchitecture 100 of figure 1 with enhancements according to a preferred embodiment of the present invention. The dotted-line rectangle 209 delineates the boundary between modules that are on-chip and the modules that are off-chip. Processor bus 101 is as described with reference to figure 1. The bidirectional AHB bridges 131, 132 of figure 1 are both connected to the processor bus 101. The APB bridge 134 of figure 1 is illustrated as connected to AHB Bridge 132. Other modules, such as 103, 104,

105, 106, 345, 346, and so on from figure 1, are employed in chip 209 (but not illustrated).

A first 111 and second 112 processor core are connected to a shared floating point unit module 115. A third 213 and fourth 214 processor core are substituted for the processor
5 cores 113 and 114 of figure 3. Each processor core is illustrated to have a Level 1 instruction (I\$) cache and a Level 1 data (D\$) cache.

The memory bus 102 of figure 1 is replaced with a first memory bus 202, a second memory bus 203 and optionally a third memory bus 204. The first memory bus 202
10 implements a 128-bit wide AHB running at 400 MHz. The second 203 and third 204 memory busses also implement 128-bit wide AHB running at 400 MHz. Each memory bus 202, 203, 204 operates independently of the other memory busses. Each memory bus 202, 203, 204 has a memory controller 221, 231, 241 which is connected to a corresponding SDRAM module 222, 232, 242 respectively. Each memory bus 202, 203,
15 204 is connected to a corresponding memory scrubber 220.

The monolithic level 2 cache module 121 of figure 1 is illustrated as being replaced with 4 or optionally 5 independent level 2 cache modules 223, 224, 233, 234, 243, in which each of the level 2 cache modules is connected as an AHB bus slave to the AHB processor bus
20 101.

The 3 SDRAM memory controllers 221, 231, 241 (when coupled with their corresponding SDRAM modules 222, 232, 242) can each be described as an independent memory store. Each of the 3 memory store 221, 231, 241 has an internally contiguous input address space
25 of at least 2 kilobytes in length. To be clear, the internally contiguous input address space is the address area decoded by the memory store's bus slave interface and which maps to elements of accessible memory in the SDRAM module.

Furthermore each of the 3 memory stores 221, 231, 241 has an internally contiguous input
30 address space of at least 2 kilobytes in length which does not overlap with the internally contiguous input address space of any other memory store. To be clear, a bus master can access elements in any one of the 3 memory stores discretely.

Each of the 5 cache modules 233, 234, 223, 224, 243 has an internally contiguous input address space of at least 1 kilobyte in length which does not overlap with the internally contiguous input address space of any of the other cache modules 233, 234, 223, 224, 243.

- 5 The 5 cache modules (233, 234, 223, 224, 243) are arranged in parallel, not cascaded. Specifically, the output address space of the first cache module 233 is not mapped to the input address space of any other 234, 223, 224, 243 of the at least two cache modules. Furthermore, the output address space of any of the other 234, 223, 224, 243 at least two cache modules is not mapped to the input address space of first cache module 233.

10

A first contiguous subset of the internally contiguous input address space of at least 1 kilobyte in length of the memory store 221 is bijectively mapped as cacheable with at least a contiguous subset of the internally contiguous input address space of the level 2 cache module 223. That subset of the internally contiguous input address space of the level 2
15 cache module 223 being bijectively mapped as cacheable with at least a subset of the contiguous output address space of each of the bus masters 111, 112, 213, 214.

In figure 2, each bus master 111, 112, 213, 214 can perform memory transfer requests with cache modules 223, 224, 234, 233, 243.

20

A second contiguous (non-overlapping) subset of the internally contiguous input address space of at least 1 kilobyte in length of the memory store 221 is bijectively mapped as cacheable with at least a subset of the internally contiguous input address space of the level 2 cache module 224; that subset of the internally contiguous input address space of
25 the level 2 cache module 224 being bijectively mapped as cacheable with at least a subset of the contiguous output address space of each of the bus masters 111, 112, 213, 214.

Note that the contiguous subset of the internally contiguous input address space of the memory store 221 which is mapped as cacheable to the cache module 223 does not
30 overlap with the contiguous subset of the internally contiguous input address space of the memory store 221 which is mapped as cacheable to the second cache module 224.

Also note that the subset of the internally contiguous input address space of the cache

module 223 which is mapped to the bus master 111 does not overlap with the subset of the internally contiguous input address space of the second cache module 224 which is mapped to the first bus master 111.

- 5 In further preferred embodiments of the present invention, in which memory store 221 has an internally contiguous input address space of at least 3 kilobytes in length, a third contiguous (non-overlapping) subset of the internally contiguous address space of the memory store 221 is mapped as an uncacheable region of memory addressable through either cache module 223 or 224.

10

A fourth contiguous (non-overlapping) subset of the internally contiguous input address space of at least 1 kilobyte in length of the memory store 231 is bijectively mapped as cacheable with at least a subset of the internally contiguous input address space of the level 2 cache module 234; that subset of the internally contiguous input address space of the level 2 cache module 234 being bijectively mapped as cacheable with at least a subset of the contiguous output address space of each of the bus masters 111, 112, 213, 214.

15

A fifth contiguous (non-overlapping) subset of the internally contiguous input address space of at least 1 kilobyte in length of the memory store 231 is bijectively mapped as cacheable with at least a subset of the internally contiguous input address space of the level 2 cache module 233; that subset of the internally contiguous input address space of the level 2 cache module 233 being bijectively mapped as cacheable with at least a subset of the contiguous output address space of each of the bus masters 111, 112, 213, 214.

20

25 A memory transfer request initiated by an AHB bus master (111, 112, 213, 214, 131, 132) on processor bus 101 to a discrete element of memory residing in memory store 222, 232 or 241 will be serviced by only one of the independent cache modules 223, 224, 233, 234, 243. In this way we are partitioning a monolithic “level 2 cache” into smaller independent level 2 caches 223, 224, 233, 234, 243, in which each independent cache has been assigned it's own portion of the physical address space which is non-overlapping with the other level 2 caches 223, 224, 233, 234, 243.

30

A read memory transfer request transported over the processor bus 101 to recall data

stored in memory store 222 will not interfere with the selection of cache-lines, or the value of the data in the cache-lines, stored in cache modules 233, 234 and 243. This partitioning configuration permits every processor core access to all elements of the main memory stores (on 222, 232, 242), while providing software developers access to controls that are
5 implemented in the hardware of the microarchitecture to minimize the cache timing interference between memory access operations to different contiguous partitions of at least 1 kilobytes in length of the main memory that have been mapped to different cache modules. With careful management of the assignment of operating system and application memory to different partitions of main memory (as is described below), it is possible to
10 reduce the amount of timing interference between unrelated software tasks, leading to improved timing composability. Generally speaking, a reduction in the amount of unwanted timing interference between tasks tends to tighter WCET estimate bounds.

In some preferred embodiments of the present invention, the processor cores 213 and 214
15 are implemented using LEON4T cores as described in figure 1.

In some preferred embodiments of the present invention, the third processor core 213 implements an entirely different instruction-set, such as the Power instruction set, and the fourth processor core 214, implements an Intel IA64 compatible instruction set.

20 Furthermore, the fourth processor core implements an aggressive out-of-order execution pipeline. These embodiments of the present invention are designed to support applications running on different instruction sets, while efficiently sharing cache and memory resources with improved timing composability.

25 In some further preferred embodiments of the present invention, the third 213 and fourth 214 processor cores implement instruction sets that share a common subset of instructions. For example, the third processor core is a LEON4T processor, and the fourth processor core is a Fujitsu SPARC64 Ix fx core that implements the SPARC v9 instruction set which has binary compatibility for SPARC v8 applications. The LEON4T core 213 is better
30 suited to worst case execution time (WCET) analysis, whereas the SPARC64 Ix fx core 214 is better suited to running computationally expensive tasks as fast as possible in the average case. In this way, this embodiment of figure 2 is optimized for running real-time tasks on cores 1 to 3 (111, 112, 213), and computationally expensive tasks on core 4 (114).

In some preferred embodiments of the present invention, at least one processor bus employs a quality-of-service scheme that enforces an upper and/or lower bound on bandwidth for at least one bus master over that bus. In some preferred embodiments of the present invention, at least one processor bus employs a quality-of-service scheme that ensures an upper and/or lower bound on the jitter of a memory transfer operation issued by at least one bus master.

In preferred embodiments of the present invention, four “address translation units” (ATU) 251, 252, 253, 254, are connected between the shared processor bus 101 and the four processor cores 111, 112, 213, 214 respectively. The functionality of an ATU is described below. In yet further preferred embodiments, there is one ATU for every bus master on the shared processor bus 101 (not illustrated).

Figure 3 is a hybrid diagram which incorporates block schematic diagram 300, a memory partitioning scheme illustrating partitioning of elements of physical memory 330 and a memory partitioning scheme illustrating the partitioning of physical cache module resources 318, 310 and 312.

Figure 3 illustrates the four processor cores 111, 112, 213 and 214 of figure 2, each processor core additionally having a corresponding ATU 251, 252, 253, 254 respectively (as illustrated in figure 2). Figure 3 also illustrates:

four independent level 2 cache modules 223, 224, 233, 234 of figure 2;
a peripheral 305 directly (or indirectly) connected to the processor bus 101 which is capable of issuing memory transfer requests to the four independent level 2 cache modules 223, 224, 233, 234; and
the two SDRAM modules 222 and 232 of figure 2.

In figure 3, each SDRAM module 222 and 232 has equivalent configurations (e.g. the same number of rows, columns, banks and so on).

The four address translation units (ATU) 251, 252, 253, 254 of figure 2 which are connected to processor cores 111, 112, 213, 214 of figure 3 translate the address space of

memory transfer requests initiated on the bus master interface of the processor cores. In figure 3, each ATU employs the same memory address translation scheme in which:

The address space of memory store 222 is logically partitioned into 8 parts 331, 332, 333, 334, 341, 342, 343, 344;

5 The address space of memory store 232 is logically partitioned into 8 parts 351, 352, 353, 354, 361, 362, 363, 364;

The 16 parts (331, 332, 333, 334, 341, 342, 343, 344, 351, 352, 353, 354, 361, 362, 363, 364) are arranged into composite partitions as follows:

10 The parts {331, 332} are concatenated together as a composite partition, so that the next block after the last block of 331 is the first block of 332.

The parts {341, 342} are concatenated together as a composite partition, so that the next block after the last block of 341 is the first block of 342.

The parts {351, 352} are concatenated together as a composite partition, so that the next block after the last block of 351 is the first block of 352.

15 The parts {361, 362} are concatenated together as a composite partition, so that the next block after the last block of 361 is the first block of 362.

The parts {333, 343} are striped together (interleaving blocks of contiguous regions of 333 with blocks of contiguous regions of 343) as a composite partition, in which the size of each block has been set to the largest

20 (instruction or data) cache-line of the four processors 111, 112, 213, 214, which in the case of figure 3 when employing LEON4 cores the maximum cache-line size is the instruction cache line size (32-bytes in length).

The parts {353, 363} are striped together (interleaving blocks of contiguous regions of memory) as a composite partition, in which the size of each block has been set to the largest (instruction or data) cache-line of the four processors 111, 112, 213, 214.

25 The parts {334, 344, 354, 364} are striped together (interleaving blocks of contiguous regions of memory) as a composite partition, in which the size of each block has been set to the largest (instruction or data) cache-line of the four processors 111, 112, 213, 214.

The above-described architecture can be employed in numerous ways. In one exemplary multicore environment, implementing two operating system instances, each operating

system instance running in asymmetric multiprocessing mode on its own core, and a third operating system instance operating in symmetric multiprocessing mode over two cores:

The memory for operating system instance 1 (OS-1) is stored in 331;

The memory for operating system instance 2 (OS-2) is stored in 342;

5 The memory for operating system instance 3 (OS-3) is stored in 351;

Memory for tasks in OS-1 are stored in part 332;

Memory for tasks in OS-2 are stored in part 341; and

Memory for tasks in OS-3 are stored in part 352, 361 and 362.

10 As described above, each processor core 111, 112, 213, 214 may access any memory location (330) offered by the slave interfaces of the SDRAM modules 222 and 232 through their respective memory controllers. Each memory access to 330 will be routed through the appropriate cache module selected from 223, 224, 233, 234. In this example, a first task running on OS-1 (using memory 332) calling the kernel of OS-1 (using
15 memory 331) will access memory stored in 222 through the first level 2 cache 223. A second task running on OS-2 calling the kernel of OS-2 will access memory stored in 222 through the second level 2 cache 224. This illustrates that the first task and second task can call their own respective operating system kernels at the same time without introducing timing interference at the level 2 cache level.

20

In further preferred embodiments of the present invention, each processor core has a memory management unit controlled by the operating system instance running on that core, in which the MMU is configured by the operating system instance to map a subset of the virtual address space to pages of memory which are mapped to the parts of physical
25 memory assigned to that core.

We observe that tasks accessing composite partitions {331, 332}, {341, 342}, {351, 352}, {361, 362} are limited to using only the number of cache-lines present in the corresponding cache module 223, 224, 233, 234.

30

Striping permits two {333, 343}, {353, 363} or more {334, 344, 354, 364} cache modules to be coupled together, permitting a single task to access a contiguous region of memory mapped over two or more cache modules, in this case doubling or quadrupling the number

of cache-lines available to accelerate access to that contiguous region of memory.

For example, an operating system may temporarily halt tasks running on core 213 when running a specific scheduled task on 214 that only accesses memory stored in composite parts {353, 363}, {351, 352}, {361, 362}. In this case that task can take advantage of twice the amount of cache-lines 312 across cache modules 233 and 234 without interference from core 213, and without causing timing interference to memory accesses by processor cores 111 and 112 to caches 223 and 224 and memory store 222. After that task is finished (or that task's timeslot is finished), that task can be swapped out, optionally all cache-lines evicted from the cache modules 223 and 224 for security reasons, tasks on core 213 un-halted, and other tasks permitted to execute on the cores 213 and 214.

In a similar example, a single task running on core 111 may be granted exclusive access to the entire cache and memory subsystem. First all tasks running on cores 112, 213, and 214 are temporarily halted. Then, when required for security reasons, all cache-lines in cache modules 223, 224, 233, 234 are evicted. Then the task running on core 111 can perform memory transfer requests to the composite partition {334, 344, 354, 364} and take full advantage of the cache-lines in all four caches 223, 224, 233, 234 without timing interference from memory transfer requests issued from other cores.

In a further preferred embodiment of the present invention, one or more bus masters peripherals are instructed to temporarily halt issuing memory transfer requests onto one or both processor busses 101, 601.

In a further preferred embodiment of the present invention, the bus arbiter is instructed to temporarily ignore memory transfer requests from selected bus masters connected to that bus.

In principle, any core may be permitted to access any region of memory at any time. For example two tasks on two different cores may share memory on any of the parts of 330. The partitioning architecture in microarchitecture 200 of figure 2 and 300 of figure 3 provides the software developer / integrator with the opportunity to write and schedule tasks in a way that minimizes sharing of cache resources to improve timing composability.

Figure 4 is a hybrid diagram which incorporates a block schematic diagram 400 illustrating four cache modules 223, 224, 233, 234, and two memory stores, a memory partitioning scheme illustrating partitioning of blocks of logical memory 402, the partitioning of blocks of physical memory {331, 332, 333, 334, 341, 342, 343, 344, 354, 364}, and a memory address translation scheme 405 that maps blocks of logical memory 402 to blocks of physical memory {331, 332, 333, 334, 341, 342, 343, 344, 354, 364}.

In figure 4, each part 331, 332, 333, 334, 341, 342, 343, 344, 354, 364 is divided into 4 blocks of 32-bytes in length labeled a, b, c, d. For example, the first (a), second (b), third (c) and fourth (d) blocks of part 331 are referenced in this text as 331.a, 331.b, 331.c, and 331.d respectively.

Figure 4 illustrates the mapping of a contiguous region of the logical address space 402 to the physical address space of the two SDRAM modules 222 (parts 331, 332, 333, 334, 341, 342, 343, 344), and 232 (parts 354, 364), including the address space mapping 405 implemented by the address translation units 251, 252, 253, 254 of figure 2.

Each part 411, 412, 413, 414, 415, 416, 417, 418 of the logical address space 402 is divided into 4 blocks of 32-bytes in length labeled a, b, c, d. For example, the first (a), second (b), third (c) and fourth (d) blocks of part 411 are referenced in this text as 411.a, 411.b, 411.c, and 411.d respectively.

The address space mapping 405 is performed by the address translation units (ATU) 251, 252, 253, 254 of figure 2. An ATU circuit implementing address space mapping 405 is illustrated in figure 5 and described below.

Figure 4 illustrates:

Four cache-lines in partition 331 and 332 being directly mapped to the 4 blocks in 411, and 412 respectively. The last block 411.d of part 411 is followed by the first block 412.a of part 412, creating a composite partition {331, 332} of figure 3 with a single contiguous address space.

Four blocks in part 413 (413.a, 413.b, 413.c, 413.d) being striped as follows:

{413.a, 333.a}, {413.b, 343.a}, {413.c, 333.b}, and {413.d, 343.b}. Four blocks in 417 are striped as follows: {417.a, 333.c}, {417.b, 343.c}, {417.c, 333.d}, and {417.d, 343.d}. Note that various blocks of parts 413 are mapped through one or the other of the two cache modules 223 and 224 of figures 4 and 5.

5 Four blocks in part 414 are striped as follows: {414.a, 334.a}, {414.b, 344.a}, {414.c, 354.a}, {414.d, 364.a}. The 4 blocks in part 418 are striped as follows: {418.a, 334.b}, {418.b, 344.b}, {418.c, 354.b}, {418.d, 364.b}. Note that various blocks of parts 414 are mapped through one or the other of the four cache modules 223, 224, 223, 224 of figures 4 and 5.

10

From the above description of figure 2, figure 3 and figure 4, it can be seen that there is an architecture which has an ATU 251, 252, 253, 254 for each of the four processor cores 111, 112, 213, 214. Each ATU has an internally contiguous input address space on its bus slave interface which is partitioned into at least two portions: a first portion of which is 15 bijectively mapped to within the address space of a first cache module; and another portion of which is striped across 2 cache modules; and in which each stripe is further partitioned into segments of at least one cache-line in length.

For example ATU 251 connected to core 111 has an internally contiguous address space 20 that has 4 parts (411, 412, 413, 414), in which a first part 411 is bijective mapped 405 within the internally contiguous address space of cache module 223 to part 331; and in which blocks of another part 413 is striped 405 across 2 cache modules 223, 224 to parts 333 and 343.

25 It can be seen that the block 331 is exclusively mapped into the internally contiguous address space of cache module 223, and that block 341 is exclusively mapped into the internally contiguous address space of cache module 224.

Figure 5 is a hardware block schematic of an example of a circuit 500 implementing an 30 address translation unit 251, 252, 253, 254 of figures 2, 3 and 4. The circuit 500 implements the address space mapping illustrated in 405 of figure 4. Circuit 500 can be trivially adapted to accommodate any block length, to dynamically increase the size of the total addressable memory, to dynamically vary the size of the partitions and so on.

Bracket 510 identifies an 11-bit wide bus (530, ..., 540) which carries the value of an 11-bit long physical address to the ATU circuit issued by the processor core connected to the ATU circuit. In this example, the 11 bit long value can address a total of 2 kilobytes of memory, clearly insufficient for production systems. Increasing the number of bits in the physical address space increases the total addressable memory in the usual way.

The number of partitions is largely independent of how large the physical address space is. As previously described, the composite partitions are illustrated to employ the use of either 1, 2 or 4 cache modules. However, an embodiment of the present invention could be realized without a composite partition striped over 4 cache modules.

Bracket 560 identifies an 11-bit wide bus (570, ..., 580) which carries the value of an 11-bit long physical address generated as output of the ATU. The number of bits in bus 560 corresponds to the number of bits in bus 510 to permit a bijective mapping. In fact, the circuit 500 does implement a bijective mapping of the physical address space.

The address space of the cache modules, and the elements within each cache module, are mapped to the physical address spaces 510 and 560 in which:

- 20 bracket 521 identifies the highest two bits 540, 539 of the physical address space 510, the values of which are used to select between the four cache modules 223, 224, 223, 224 of figures 4, 5, and 6;
- bracket 511 identifies the first 9 bits {530, ..., 538} of the physical address space 510 that map to the input address space of one of the four cache modules 223, 224, 223, 224 selected by the value on bits 521 and 522; and in which:
 - the value on bits {538, 537} selects one of the four partitions of the selected cache module 223, 224, 223, 224;
 - the value on bits {536, 535} selects one of the four blocks of the selected partition {331, 332, 333, 334, 341, 342, 343, 344, 351, 352, 353, 354, 361, 362, 363, 364}; and
 - the value on bits {534, 533, 532, 531, 530} selects one of the 32 bytes of the selected block.

Permutation module 552 receives a value of 6-bits in length {540, 539, 538, 537, 536, 535} and generates a value of 6-bits in length corresponding to the value on bits {540, 535, 539, 538, 537, 536} respectively. Permutation module 553 receives a value of 6-bits in length {540, 539, 538, 537, 536, 535} and generates a value of 6-bits in length
5 corresponding to the value on bits {536, 535, 540, 539, 538, 537}.

The 6-bit wide 2-to-1 multiplexer 551 generates as output the value generated by module 553 when the value received on wire 537 is zero, otherwise generates as output the value generated by module 552.

10

The 6-bit wide 2-to-1 multiplexer 554, receives the value of the 6-bit wide output of the multiplexer 551 as the first data input and the value of the 6-bits 540 to 535 as the second data input. The multiplexer is controlled from wire 538, generating as output the value received on the first data input when the value on wire 538 is one, otherwise generating as
15 output the value received on the second data input.

The value of the 6-bits of output of multiplexer 554 are mapped to bits 580 to 575 as the output of the ATU module. The value of the 5-bits 534 to 530 are mapped to bits 574 to 570 respectively as the output of the ATU module.

20

In preferred embodiments of the invention (which are not illustrated in the drawings), additional circuitry, such as additional multiplexers, are used to dynamically change the number and/or configuration (non-striped, striped across 2 caches, striped across 4 caches) and or size of the composite partitions.

25

Figure 6 is a schematic block diagram illustrating an architecture 600 which is an adaption of the NGMP microarchitecture 100 of figure 1 with enhancements according to a preferred embodiment of the present invention. Figure 6 illustrates a computing device comprising:

30 at least one sub-computing device 608, 609, each sub-computing device comprising:

at least one bus 101, 601;

at least one bus master 611, 612, 132, 141, 133, 613, 614; and

at least one memory store 222, 232; and

in which at least one of the at least one sub-computing devices 601 comprises:

at least two bus masters 606, 253, 254; and

a means to enable or disable at least one of the at least two bus masters 605

5 from issuing memory transfer requests onto the bus without resetting the at least one of the at least two bus masters; and

in which at least one 608 of the at least two sub-computing devices comprises:

at least one bus master 613, 614, 606; and

at least one unidirectional bus bridge 605, 606,

10 one of the at least one unidirectional bus bridges 606, having:

its bus master interface connected to one 601 of the at least one busses of this sub-computing device; and

its bus slave interface connected to one 101 of the at least one busses of another sub-computing device.

15

In figure 6, an AHB processor bus A 101 is as described in figure 3. The bidirectional bridges 131, 132, 133 of figure 3 are connected to the processor bus A 101. A PCI Master 141 peripheral of figure 1 is connected as a bus master to bidirectional bridge 132. A 128-bit wide AHB 602 running at 400 MHz is referred to as the memory bus A. A first level 2
20 cache module 223 as described in figure 2 is connected as a bus slave to the processor bus A 101 and as a bus master to the memory bus A 602. A second level 2 cache module 224 as described in figure 2 is connected as a bus slave to the processor bus A 101 and as a bus master to the memory bus A 602. A memory scrubber 220.a is connected as a bus master to the memory bus A 602. A SDRAM memory controller 221 as described in figure 2 is
25 connected to off-chip SDRAM module 222.

A 128-bit wide AHB 601 running at 400 MHz is referred to as processor bus B. A 128-bit wide AHB 603 running at 400 MHz is referred to as the memory bus B. A third level 2
cache module 233 as described in figure 2 is disconnected from 101 and 203 and
30 connected as a bus slave to the processor bus B 601 and as a bus master to the memory bus B 603. A fourth level 2 cache module 234 as described in figure 2 is disconnected from 101 and 203 and connected as a bus slave to the processor bus B 601 and as a bus master to the memory bus B 603. A memory scrubber 420.b is connected as a bus master to the

memory bus B 603. An SDRAM memory controller 231 as described in figure 2 is connected to off-chip SDRAM module 232 and as a bus slave to memory bus B 603.

An APB bridge 634 is connected to processor bus 601. One or more bus slave peripherals are connected to the APB bridge (not illustrated). In preferred embodiments of the present invention, peripherals connected to APB bridge 634 are not accessed by bus masters on processor bus A 101. Likewise, bus slave peripherals connected to bridges 132, 131, 133 are not accessed by bus masters on processor bus B.

10 A first 611 and second 612 modified Aeroflex Gaisler LEON4T processor core module share an unmodified floating point unit module 115 that replaces processor core modules 111, 112 of figure 2 respectively. The first 611 and second 612 processor cores are connected to the processor bus A 101.

15 A third 613 and fourth 614 modified Aeroflex Gaisler LEON4T processor core module share an unmodified floating point unit module 116 that replaces processor core modules 213, 214 of figure 2 respectively. The third 613 and fourth 614 processor cores are disconnected from processor bus A 101 and connected to the processor bus B 601.

20 In figure 6, the LEON4T processor cores 611, 612, 613, 614 are modified so that their level 1 data cache has 2 snoop busses, instead of 1 snoop bus as found in their original design. The first and second snoop busses on each processor core are connected to the first 101 and second 601 processor busses respectively as illustrated in figures 6 and 9.

25 Two address translation units (ATUs) 251, 252 sit in between the shared processor bus A 101 and the processor cores 611, 612 respectively. Two ATUs 253, 254 sit in between the shared processor bus B 601 and the processor cores 613, 614 respectively.

30 A unidirectional 128-bit wide AHB to 128-bit wide AHB bridge 605 is connected to the processor bus A 101 of first sub-computing device 609 as a bus master and to processor bus B 601 of second sub-computing device 608 as a bus slave. A unidirectional 128-bit wide AHB to 128-bit wide AHB bridge 606 is connected to the processor bus A 101 as a bus slave and to processor bus B 601 as a bus master.

In a preferred embodiment of the present invention, the unidirectional bus bridge 606 is actuable to pass or to refuse to pass memory transfer requests from one, some or all bus masters 611, 612, 132, 131, 133 on processor bus 101 to one some or all bus slaves 223, 224 on processor bus 601. In a preferred embodiment of the present embodiment, refusing to pass a memory transfer request is achieved at run-time by sending an AHB "RETRY" response to the bus master that issued the memory transfer request. In an alternate preferred embodiment of the present embodiment, refusing to pass a memory transfer request is achieved at run-time by sending an AHB "SPLIT" response to the bus master that issued the memory transfer request. In an alternate preferred embodiment, refusing to pass is achieved at run-time by sending an AHB "ERROR" response to the bus master that issued the memory transfer request. In a further preferred embodiment of the present invention, the decision to send a AHB "RETRY", "SPLIT" or "ERROR" response may be adjusted at run-time specifically for one, each or all cores.

In an alternate preferred embodiment of the present invention, the unidirectional bus bridge 606 is actuable to pass or to indefinitely delay memory transfer requests from one, some or all bus masters 611, 612, 132, 131, 133 on processor bus 101 to one, some or all bus slaves 223, 224 on processor bus 601. In a further preferred embodiment of the present invention, after the memory transfer request is delayed a number of clock-cycles, an AHB "RETRY", "SPLIT", or "ERROR" response is sent to the bus master that issued the memory transfer requested that was delayed.

Together 605 and 606 create a bidirectional bridge between processor bus A 101 and processor bus B 601. The bridges 605 and 606 can be used to isolate (see dashed line 607) each processor bus 101 and 601 from some or all unrelated bus activity on the other processor bus 601 and 101 respectively. The bridges 605 and 606 can be used to divide a computing device 209 into two (largely independent) sub-computing devices 608 and 609. When this is done, (and there is at least one processor core present in each sub-computing device 608, 609 as illustrated in figure 6), the performance of each sub-computing device 608 and 609 can be adapted to approximate that of "a single core equivalent" computing device. For example, to enable a single core equivalent context in sub-computing device 608, the unidirectional bridge 606 is actuated to retry or halt all memory transactions (with

or without delays), the memory scrubber 220.b is temporarily halted, the fourth processor core 614 is temporarily halted (by one or any combination of the following means: (a) software executing on core 613 instructing the bus-arbiter 601 to temporarily refuse to grant memory transfer requests issued from processor core 613 to the bus 601; (b)

5 software executing on core 613 modifying the value of a register that temporarily enables or disables execution of instructions on CPU 614; or (c) software executing on core 613 instructing the task scheduler running on processor core 614 to context swap to a task that does not issue memory operations to the bus 601 until instructed otherwise), the third core 613 is then granted exclusive access to the two level 2 caches 223, 224, and SDRAM module 232, all three of which are accessible over processor bus B 601. Note that the task
10 running on processor core 613 can access bus slave peripherals over APB bridge 634.

Optionally, the unidirectional bridge 606 is actuated to delay and/or retry or halt all memory transactions to prevent core 614 from issuing memory transactions that are relayed onto processor bus A 101 as they may receive uncontrolled timing interference
15 from other bus masters (such as 611, 612) connected to processor bus A 101. Interrupts generated by bus slave peripherals attached to processor bus 601 can be masked to prevented unwanted timing interference to the execution of a task.

In this way figure 6 illustrates a sub-computing device 608 that comprises at least two bus
20 masters 606, 613, 614 and a means to grant exclusive access to one 613 of the at least two bus masters to issue memory transfer requests to the bus 601.

Figure 6 also illustrates a computing device further comprising:

at least one (level 1) cache module (in each processor core); and
25 means for maintaining cache coherency (through dual snoop ports in each processor core 611, 612, 613, 614, each dual snoop port connected to each processor bus 101, 601) with at least one other cache module on another sub-computing device when at least one of the unidirectional bus bridge 605, 606 connecting the two sub-computing devices is in a state in which it is not passing
30 memory transfer requests issued on one bus to the other bus.

In a further preferred embodiment a high-precision timer circuit peripheral (not illustrated) is used by the active processor core to trigger shortly after the estimated/calculated “worst

case execution time” for the currently active task running within a single core equivalent context. The output of that currently active task is then released only after the timer circuit is triggered. These capabilities can be used to increase the deterministic behaviour of the system by masking (most if not all) data-dependent execution time variation of certain task operations. Clearly if the estimated/calculated WCET for a task is wrong, then some information may be leaked (either by the task failing to complete, or by the task completing late).

In further preferred embodiments of the present invention, a Direct Memory Access (DMA) module 635 is attached to processor bus B, and that DMA module is used as a time-analyzable accelerator for immediately moving memory under the control of a task running in a single core equivalent context, ensuring that only the processor core, or DMA module, is issuing memory transfer requests onto processor bus B at any given window of time. E.g. A task running within a single core equivalent context issues commands instructing the DMA module residing in the same single core equivalent context to immediately move memory, and that task waits till the DMA module signals completion of memory movement before that task issues any further memory transfer requests. In this way the means to ensure only one bus master is achieved in software. In preferred embodiments of the present invention, means are implemented in hardware to ensure at most one of the {CPU, DMA} modules is capable of issuing memory transfer requests onto the processor bus B.

In further preferred embodiments of the present invention, the memory scrubber 220.b maintains a notion of time and dynamically adjusts the rate of memory scrubbing to ensures deadlines (such as ensuring each row of memory is scrubbed within a given duration of time) are met, permitting the scrubbing operations to be temporarily halted. In further preferred embodiments of the present invention, the memory scrubber is manually controllable, and a portion of time is regularly scheduled by the task scheduler to perform the necessary memory scrubbing.

In preferred embodiments the memory controller 231 maintains a notion of time and dynamically adjusts the rate of SDRAM memory refresh to ensure its refresh deadlines are always met, permitting the memory refresh operations to be temporarily halted. In further

preferred embodiments of the present invention, the SDRAM refresh operations of the memory controller are manually controlled from software, and time is regularly scheduled by the task scheduler to perform memory refresh.

- 5 In further preferred embodiments of the present invention the off-chip SDRAM modules 222 and 232 are implemented using on-chip embedded SDRAM. In further preferred embodiments of the present invention the off-chip SDRAM modules 222 and 232 are implemented using on-chip 3-D IC memory. For example, and without exclusion, the 3-D memories offered by Tezzaron Semiconductor, MonolithIC 3D Inc or Micron. In alternate
10 embodiments of the present invention, the SDRAM modules 222 and 232 are substituted with one or a combination of SRAM, T-RAM, Z-RAM, NRAM, Resistive RAM, TTRAM, Racetrack memory, Millipede memory, NVRAM modules, or other types of computer memory technology.
- 15 It will be seen that the embodiment of figure 6 results in a computing device, with at least two sub-computing devices 608, 609, each sub-computing device comprising:
at least one cache module 223, 224, 234, 233;
means for maintaining cache coherency (through two snoop ports on each
processor core) with at least one other cache module on another sub-computing
20 device when at least one of the unidirectional bus bridge 605, 606 connecting the two sub-computing devices is in a state in which it is not passing memory transfer requests issued on one bus to the other bus.

- In a further preferred embodiment of the present invention, a dual-port SRAM store 698
25 (1x read only port, 1x write only port) is connected to processor bus 601 and 101, in which one-port of the dual-port SRAM 698 is a read-only port connected to bus 601, and the other-port of the dual-port SRAM 698 is a write-only port connected to bus 101. This permits bus masters on sub-computing device 609 to communicate with bus master devices on sub-computing device 608 over SRAM 698, even when both bridges 605 and
30 606 are refusing to pass memory transfer requests.

In a further preferred embodiment of the present invention, a true dual-port SRAM store 699 (each port having read and write capabilities) is connected to processor bus 601 and

101. This permits bus masters on sub-computing device 608 to communicate with bus master devices on sub-computing device 609 over SRAM 699, even when both bridges 605 and 606 are refusing to pass memory transfer requests.

- 5 In a further preferred embodiment of the present invention, all access to dual-port SRAM stores 698, 699 is non-cacheable and thus trivially coherent across all cores 611, 612, 613, 614.

10 In this way figure 6 illustrates A computing device (600) for performing real-time and mixed criticality tasks comprising:

N sub-computing devices (608, 609), where value of N is at least 2, each sub-computing device comprising:

at least one bus (101, 601);

15 at least one bus master (605, 606, 611, 612, 613, 614, 132, 131, 133), in which at least one of the at least one bus masters is a processor core (611, 612, 613, 614);

the bus slave interface of at least one memory store (698, 699) connected to one of the at least one busses (101, 601) and

20 at least one unidirectional bus bridge (605, 606), for each unidirectional bus bridge (605, 606), each bus interface of that unidirectional bus bridge (605, 606) is connected to the bus (101, 601) of a different one of the sub-computing devices;

at least one memory store (698, 699), each memory store (698, 699) comprising at least two bus slave interfaces, for each memory store (698, 699), each bus slave interface is connected to the bus of a different one of the sub-computing devices (101, 601);

25 in which:

X of the N sub-computing devices (608, 609) are directly connected to a common bus (101, 601) by a corresponding bus bridge (605, 606) where the value of X is $2 \leq X \leq N$;

30 a first set of two of the sub-computing devices (608, 609) are connected to each other by a first memory store (698, 699) comprising at least two bus slave interfaces.

Figure 7 is a block schematic diagram illustrating an AMBA v2.0 AHB controller for processor bus A 601 with 2 bus masters (611, 612), and 2 bus slaves (224, 223) illustrated. The AHB controller has an arbiter and decoder module 702, a first multiplexer 703 and a second multiplexer 704. The arbiter and decoder module receive control signals from all bus masters and bus slaves connected to it (not illustrated). After evaluating those control signals in accordance with the AMBA v2.0 AHB specifications, the arbiter selects which input to the multiplexers 703, 704 should be routed to slave and bus masters respectively. The AMB v2.0 AHB standard supports up to 16 masters and 16 slaves. A full description of the AHB arbiter and decoder modules and protocols is available in [2].

Figure 7 is also a block schematic drawing illustrating an AMBA v2.0 AHB controller for processor bus B with 2 bus masters (613, 614), and 2 bus slaves (234, 233) illustrated. The AHB controller has an arbiter and decoder module 712, a first multiplexer 713 and a second multiplexer 714. The arbiter and decoder module receives control signals from all bus masters and bus slaves connected to it (not illustrated). After evaluating those control signals in accordance with the AMBA v2.0 AHB specifications, the arbiter 712 selects which input to the multiplexers 713, 714 should be routed to slave and bus masters respectively.

Figure 8 is a block schematic diagram of the NGMP microarchitecture of figure 1 illustrating the 7 stage integer pipeline 802, level 1 data cache module 810, memory management unit (MMU) 803, AHB bus master interface module 820 and a first AHB bus slave interface module 840 of the first LEON4T processor core 111 connected to the processor bus 101. The LEON4T processor 111 employs a seven stage integer pipeline 802, a level 1 data cache 810, a MMU module 803, and an AHB bus master module 820.

The AHB bus master interface module 820 receives the following signals which are defined in the AMBA specifications [2]:

- input 822 receives 1-bit wide signal HLOCKx1,
- input 823 receives 1-bit wide signal HGRANTx,
- input 831 receives 128-bit wide signal HRDATA [127:0],
- input 832 receives 2-bit wide signal HRESP[1:0],
- input 833 receives 1-bit wide signal HRESETn,

input 834 receives 1-bit wide signal HREADY,
input 835 receives 1-bit wide signal HCLK.

The AHB bus master interface module 820 generates the following signals which are
5 defined in the AMBA specifications [2]:

output 821 generates 1-bit wide signal HBUSREQx1,
output 824 generates 2-bit wide signal HTRANS[1:0],
output 825 generates 31-bit wide signal HADDR[31:0],
output 826 generates 1-bit wide signal HWRITE,
10 output 827 generates 3-bit wide signal HSIZE[2:0],
output 828 generates 3-bit wide signal HBURST[2:0],
output 829 generates 4-bit wide signal HPROT[3:0],
output 830 generates 128-bit wide signal HWDATA [127:0].

15 Figure 8 illustrates the AMBA v2.0 AHB controller 701 of figure 7 with an
arbiter/decoder module 702. The multiplexers within the region bound by the dashed line
701 are the multiplexers 703 and 704 under the control of the arbiter/decoder module 702.

The first AHB bus slave interface module 840 receives the following signals which are
20 defined in the AMBA specifications [2]:

input 841 receives 2-bit wide signal HTRANS[1:0],
input 842 receives 1-bit wide signal HWRITE,
input 843 receives 32-bit wide signal HADDR[31:0],
input 844 receives 3-bit wide signal HSIZE[2:0],
25 input 845 receives 3-bit wide signal HBURST[2:0].

The AHB bus slave interface module 840 receives the output of the AHB bus master
interface module selected by the arbiter/decoder 702 on the processor bus 101 that the
associated processor core 111 is connected to. If the processor core 111 is granted
30 permission to issue memory transfer requests onto the processor bus 101, a feedback loop
will result with the bus slave interface module 840 receiving the value of the control signal
outputs generated by the AHB bus master interface module 820.

The level 1 data cache's finite state machine (FSM) 811 is illustrated in figure 8. The data cache module 812 is a dual-port memory storing the value of cached cache-lines. The address tag module 813 is a dual-port memory storing the value of memory addresses for the corresponding cache-lines stored in the data cache module 812. The status tag module 815 is a dual-port (or optionally triple-port) memory that stores the status (empty, valid, dirty, ...) of the corresponding cache-lines stored in data cache 812, and address tag 813. The status tag FSM 814 updates the state of the status tag module 815.

As described in [4], memory transfer requests to main memory generated by the 7 stage integer pipeline 802 are fed as input to the cache FSM module 811. In response the cache FSM 811 queries the status tag FSM 814 about the contents of the address tag store 813 to determine if the memory being accessed currently resides in one of the cache-lines of the level 1 data cache. Read memory transfer requests that result in a cache-miss must be resolved outside the level 1 data cache 810 and are processed by the MMU module 803 and then by the AHB bus master module 820 where they are then issued to the processor bus A 101. The full behaviour of the level 1 data cache is described in [4].

In parallel with, and somewhat independently of, the memory transfer requests generated by the 7 stage integer pipeline 802, the status FSM 814 of the level 1 data cache module 810 is constantly snooping (monitoring) memory transfer requests issued by any bus master to any other slave connected to the processor bus A 101. In particular, if a bus master connected to the processor bus A 101 issues a memory write transfer transaction to a cacheable region of memory, the status FSM 814 queries the address tag module 813 to determine if that memory location is stored in a cache-line of the level 1 data cache 812. If it is, the corresponding status information in the status tag module 815 is updated to invalidate that cache-line, effectively evicting the offending cache-line entry from the data cache 812.

Figure 9 is a block schematic diagram that extends figure 8 and illustrates a portion of the microarchitecture 600 of figure 6 according to a preferred embodiment 900 of the present invention. The first processor core 111 in figure 8 with one snoop port has been substituted with the first processor core 611 of figure 6 which has been adapted with dual snoop ports. Processor core 611 enhances the design of processor core 111 which has

been previously described in figure 8. In processor core 611, a second AHB bus slave module 940 is illustrated as receiving the following signals which are defined in the AMBA specifications [2]:

- input 941 receives 2-bit wide signal HTRANS[1:0],
- 5 input 942 receives 1-bit wide signal HWRITE,
- input 943 receives 128-bit wide signal HADDR[31:0],
- input 944 receives 3-bit wide signal HSIZE[2:0],
- input 945 receives 3-bit wide signal HBURST[2:0].

- 10 With reference to figure 8, processor bus A 101 and processor bus B 601 have logically independent AHB Bus Controllers (701, 711) as illustrated in figure 7. Recall that figure 7 shows the AHB Bus Controller 701 for processor bus A, and the AHB Bus Controller 711 for processor bus B. In figure 9, the AHB bus slave interface module 940 receives the output of the AHB bus master selected by the arbiter/decoder on the alternate processor
- 15 bus monitored by bus slave interface module 840. In this case, module 940 monitors processor bus B 601.

- In figure 9, the cache module 810 of figure 8 is extended with an additional address tag module 913 that duplicates the functionality of the address tag module 813. The cache
- 20 FSM module 911 adapts FSM module 811 of figure 8 to ensure the address tag modules 813 and 913 hold identical copies of the same information in figure 9 and implements additional capabilities described below.

- The status tag FSM 914 adapts status tag FSM 814 of figure 8 with additional capabilities
- 25 to support the snooping of write memory transfer requests issued on the processor bus B 601 and received on bus slave interface module 940.

Figure 10 is a flow-chart 1050 illustrating the steps in a dual-snoop process of figure 9 according to a preferred embodiment of the present invention.

30

In step 1051: The process is started.

Label 1052 illustrates a node.

In step 1053: The transaction type and transaction address on the processor bus A 101 and

on processor bus B 601 is observed.

In step 1054: If the memory transfer request observed on processor bus A 101 in step 1053 is not a write memory transfer request to a cacheable region of memory, and the memory transfer request observed on processor bus B 601 in step 1053 is not a write
5 memory transfer request to a cacheable region of memory, then go to node 1052, otherwise go to step 1055.

In step 1055: If there is a write memory transfer request observed on processor bus A 101 go to step 1056, otherwise go to node 1060.

In step 1056: If there is an active write memory transfer request on the AHB bus master
10 of processor core 611, the bus has been granted 823 to core 611, and the address of that active memory transfer request matches the address observed on the write memory transfer request on processor bus A 101, then go to node 1060.

In step 1057: The status FSM 914 queries the address tag module 813 to determine if the address of the write transaction is stored in the level 1 data cache.

15 In step 1058: If the query in step 1057 evaluates as true, go to step 1059, otherwise go to node 1060.

In step 1059: The corresponding status information in the status tag module 815 is updated to invalidate that cache-line, effectively evicting that cache-line entry from the data cache 812.

20 Label 1060 illustrates a node.

In step 1061: If there is a write memory transfer request observed on processor bus B 601 then go to step 1062, otherwise go to node 1066.

In step 1062: If there is an active write memory transfer request on the AHB bus master
25 of processor core 611, the bus has been granted 823 to core 611, and the address of the active memory transfer request matches the address observed on the write memory transfer request on processor bus B 601, then go to node 1066.

In step 1063: The status FSM 914 queries the address tag module 913 to determine if the address of the write memory transfer request is stored in the level 1 data cache.

30 In step 1064: If the query in step 1063 evaluates as true, go to node 1066, otherwise go to step 1065.

In step 1065: The corresponding status information in the status tag module 815 is updated to invalidate that cache-line, effectively evicting that cache-line entry from the data cache 812.

Label 1066 illustrates a node. In the next step go to node 1052.

The dual-snoop embodiment of the current invention provides the ability to double the number of memory transfer requests that can occur concurrently in the microarchitecture illustrated in figure 8. By further increasing the number of snoop-ports in this manner it is possible to increase the number of independent processor busses monitored at full memory transfer request rates. Figure 11, which is described below, illustrates an example of a processor core 1111 with four snoop ports employed in a quad processor bus (101, 610, 1101, 1102) configuration.

Write memory transfer requests generated by processor 611 on processor bus A to SDRAM 232 on processor bus B are detected first on the snoop interface 840 then on the snoop interface 940 and must be recognized (1056, 1062) as being one and the same memory transfer request and not result in the corresponding cache-line, if present in the L1 data cache of processor 611, being invalidated.

In an alternate preferred embodiment, some processor cores implementing a single-bus snoop capability can be adapted to provide a dual-bus snoop capacity. A key requirement is to be able to monitor memory transfer requests on both processor busses and invalidate cache-lines fast enough to maintain cache coherency between the processor cores. An observation is that cache-line invalidation occurs only due to write memory transfer requests to cacheable memory, and not read requests. By regulating the flow of write requests to 50% of the theoretical peak rate for processor bus A 101 and processor bus B 601 (where 101 and 601 have the same clock speed) in principle (and depending on the low-level implementation details of the processor) it should be possible for an unmodified processor which is capable of invalidating cache lines at the peak rate of write requests for one processor bus to maintain cache coherency across two write-memory transfer request rate limited busses ($50\% + 50\% = 100\%$). For example, in a two processor bus configuration, if every write operation to a cacheable region of memory is guaranteed to take at least 2 clock cycles, it is possible in principle to evict one cache-line per clock cycle using a single snoop port.

Figure 11 is a block schematic diagram of architecture 1100 that enhances

microarchitecture 800 of figure 8 to implement four independent processor busses according to a preferred embodiment of the present invention. Figure 11 illustrates a computing device in which each sub-computing device is directly connected to every other sub-computing device by a corresponding bus bridge 605, 606, 1105, 1106, 1115, 1116, 1125, 1126, 1135, 1136, 1145, 1146.

Processor bus C 1101 implements a 128-bit wide AHB running at 400 MHz.

Processor bus D 1102 implements a 128-bit wide AHB running at 400 MHz.

The modified LEON4T processor core 612 with 2 snoop ports of figure 6 is substituted with a LEON4T processor 1111 that has 4 snoop ports that monitor processor busses 101, 601, 1101, 1102 and that maintains cache coherency across the 4 processor busses. Each of the other processor cores 612, 613, 614 are each adapted to have 4 snoop ports and to monitor all four processor busses (not illustrated). In further preferred embodiments of the present invention additional processor cores with 4 snoop ports can be added to processor bus C 1101 and processor bus D 1102.

Each of unidirectional bridges 1105, 1106, 1115, 1116, 1125, 1126, 1135, 1136, 1145, 1146 are connected to two of the processor busses 101, 601, 1101, 1102 as illustrated, permitting a bus master on any one of the four processor busses 101, 601, 1101, 1102 to perform memory transfer requests with bus slaves connected to any of those four processor busses.

In this way figures 6, 7, 8, 9, 10, 11 illustrate an inventive architecture that can: (a) run real-time tasks rapidly in a time analyzable way with tight bounds; and (b) run non real-time tasks efficiently; in a multi-core system that shares resources. The ability to dynamically enable and disable bus masters and bus bridges permits the computing device to be dynamically adjusted to minimize or eliminate timing interference between bus masters when running one or more tasks with deadlines, or to permit multiple bus masters to efficiently compete for access to all shared resources when running one or more non real-time tasks.

Figure 12 is a schematic block diagram illustrating an architecture 1200 according to a

preferred embodiment of the present invention. Figure 12 illustrates a computing device comprising:

three sub-computing device 1220, 1240, 1260, each sub-computing device comprising:

- 5 at least one bus 1261, 1241, 1221;
- at least one bus master 1230, 1231, 1235, 1291, 1292, 1243, 1245, 1295, 1296, 1263, 1265; and
- at least one memory store 1224, 1227, 1293, 1294, 1281; and

in which at least one of the at least one sub-computing devices 1260 comprises:

- 10 at least two bus masters 1263, 1265, 1295; and
- a means to enable or disable at least one of the at least two bus masters 1295, 1265 from issuing memory transfer requests onto the bus 1261.

in which the first sub-computing device 1220 comprises:

- at least one memory store 1224; and
- 15 none of the at least one memory stores 1294, 1281 of the second sub-computing device 1240 has as many elements as that memory store 1224;

in which at least one memory store 1294 that has two bus slave interfaces (is implemented using a true dual port SRAM), in which:

- a first bus slave interface of that memory store 1294 is connected to a first
- 20 bus 1221 of a first sub-computing device 1220; and
- a second bus slave interface of that memory store 1294 is connected to a second bus 1241 of a second sub-computing device 1240; and
- the execution time of memory transfer requests on each bus slave interface is not influenced by the memory transfer activity on any of the other at least
- 25 two bus slave interfaces.

In figure 12, the dotted-line rectangle 1201 delineates the boundary between modules which are on-chip and modules which are off-chip. For example module 1224 is off-chip and module 1225 is on-chip.

30

The first sub-computing device 1220 comprises:

a 128-bit wide AHB bus 1221 being adapted to execute a policy selected from the group comprising:

enabling all bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus;

enabling some of the bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus;

5 enabling at most one of the bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus; and

a 32-bit wide AHB bus 1223;

an optional processor core 1230 with L1 instruction cache and L1 data cache which is adapted to snoop on memory transfer requests on bus 1221;

10 a processor core 1231 with L1 instruction cache and L1 data cache which is adapted to snoop on memory transfer requests on bus 1221;

a bidirectional bridge 1235 connected to bus 1221 and bus 1223 being actuable to enable or disable issuing memory transfer requests across that bridge in one or both directions;

15 three peripherals 1236, 1237, 1238 which are connected to bus 1223;

a multiprocessor interrupt controller 1233, in which interrupts from peripherals 1236, 1237, 1238 are individually routed to one or both of processor cores 1230, 1231;

a DMA module 1229;

20 an optional L2 cache module 1228;

an optional FLASH memory store 1227;

an optional memory scrubber 1226;

an optional SDRAM memory controller 1225; and

an optional SDRAM store 1224.

25

The second sub-computing device 1240 comprises:

a 32-bit wide AHB bus 1241 being adapted to execute a policy selected from the group comprising:

enabling all bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus;

30

enabling some of the bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus;

enabling at most one of the bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus; and
a 32-bit wide AHB bus 1242;
a processor core 1243 with L1 instruction cache and L1 data cache which is
5 adapted to snoop on memory transfer requests on bus 1221;
a bidirectional bridge 1245 connected to bus 1241 and bus 1242 being actuable to enable or disable issuing memory transfer requests across that bridge in one or both directions;
three peripherals 1246, 1247, 1248 which are connected to bus 1242; and
10 an interrupt controller 1244, in which interrupts from peripherals 1246, 1247, 1248 are routed to processor core 1243.

The third sub-computing device 1260 comprises:

a 32-bit wide AHB bus 1261 being adapted to execute a policy selected from the
15 group comprising:

enabling all bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus;

enabling some of the bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus;

20 enabling at most one of the bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus; and

a 32-bit wide AHB bus 1262;

a processor core 1263 with L1 instruction cache and L1 data cache which is adapted to snoop on memory transfer requests on bus 1221;

25 a bidirectional bridge 1265 connected to bus 1261 and bus 1262 being actuable to enable or disable issuing memory transfer requests across that bridge in one or both directions;

three peripherals 1266, 1267, 1268 which are connected to bus 1262; and

an interrupt controller 1264, in which interrupts from peripherals 1266, 1267, 1268
30 are routed to processor core 1263.

The first sub-computing device 1220 and second sub-computing device 1240 are connected by:

an optional unidirectional bridge 1291 connected to bus 1221 and bus 1241 being actuable to enable issuing memory transfer requests received from bus masters on 1221 to bus 1241;

a unidirectional bridge 1292 connected to bus 1221 and bus 1241 being actuable to enable issuing memory transfer requests received from bus masters on 1241 to bus 1221; and

a true dual port SRAM store 1294 connected to bus 1221 and bus 1241.

The first sub-computing device 1220 and third sub-computing device 1260 are connected by:

an optional unidirectional bridge 1295 connected to bus 1221 and bus 1261 being actuable to enable issuing memory transfer requests received from bus masters on 1221 to bus 1261;

a unidirectional bridge 1296 connected to bus 1221 and bus 1261 being actuable to enable issuing memory transfer requests received from bus masters on 1261 to bus 1221; and

a true dual port SRAM store 1293 connected to bus 1221 and bus 1261.

The second sub-computing device 1240 and third sub-computing device 1260 are connected by a true dual port SRAM store 1281 connected to bus 1241 and bus 1261.

Figure 12 illustrates at least one memory store 1294 comprising two bus slave interfaces, in which:

a first bus slave interface of that memory store is connected to a first bus 1221 of the first sub-computing device 1220;

a second bus slave interface of that memory store is connected to a second bus 1241 of the second sub-computing device 1240; and

the execution time of memory transfer requests on each bus slave interface is not influenced by the memory transfer activity on any of the other at least two bus slave interfaces.

In a further preferred embodiment of the present invention, each processor core 1230, 1231, 1243, 1263 can issue a maskable interrupt, or respond to an interrupt, from every

other processor core (not illustrated).

In a further preferred embodiment of the present invention, all access to SRAM stores 1281, 1293, 1294 is non-cacheable and thus trivially coherent across all cores. All
5 accesses to the level 2 cache 1228 or the memory stores 1227 and 1224 behind the level 2 cache 1228 maintain cache-coherency because all such memory transfer requests must pass over bus 1221, and each processor core snoops bus 1221.

In a further preferred embodiment, the memory store 1281 has sufficient elements to store
10 one to two RTOS instances, run-time memory for a RTOS instance, executable code and run-time memory for the executables.

In one preferred embodiment a symmetric multi processor RTOS executable code and its run-time data is stored in 1281, and run on cores 1263 and 1243. Processor affinity is used
15 to ensure that tasks that access one or more of the peripherals from the set {1266, 1267, 1268} or {1246, 1247, 1248} are run on the appropriate core 1243 or 1263 respectively. When running time-analyzable tasks, the executable code and run-time data for those tasks is stored in 1281 and the unidirectional bridges 1291, 1292, 1295, 1296 are disabled from
20 issuing memory transfer requests over them to enable single-core equivalent contexts for 1241 or 1261 respectively. If processor core 1230 is not present or if present is temporarily disabled, figure 12 can be configured at run time to create 3 single-core-equivalent contexts, each of which can access its own peripherals at the same time.

Advantageously tasks that do not access peripherals can be scheduled to run on either one
25 of the processor cores 1243 and 1263 without limitation, each of which task may be permitted to share the level 2 cache 1228 and memory stores 1228 and 1224.

Furthermore an independent RTOS executable and instance is stored in 1224 and run on core 1231 and optionally core 1230. Advantageously, both RTOS instances (stored in
30 1281 and 1226) can send messages to each other through the shared memory stores 1293, 1294.

In this way figure 12 illustrates an inventive architecture that can: (a) run real-time tasks

rapidly in a time analyzable way with tight bounds; and (b) run non real-time tasks efficiently; in a multi-core system that shares resources. The ability to dynamically enable and disable bus masters and bus bridges permits the system to be adjusted at run-time to reduce or eliminate the timing interference between bus masters when running one or
5 more tasks that have deadlines, or to efficiently share all resources when running one or more non real-time tasks. When bus bridges 1295 and 1291 are disabled or absent the potential sources of unwanted timing interference from bus masters in the first, second and third sub-computing devices against the second and third sub-computing devices are greatly reduced. When bus bridges 1292 and 1296 are disabled, then it is possible to
10 prevent real-time tasks being delayed by memory transfer requests issued on to bus 1221.

These isolation controls in the hardware increase assurance levels with regard to correct operation of the device, and simplify safety and/or security analysis and certification costs.

15 The ability to enable memory transfer requests over any one or both of bridge 1292 and 1296 permits non-real time tasks to efficiently access memory stores attached to the first sub-computing device 1223 with low latency. As all memory transfer requests to memory stores 1224 and 1227 must be issued onto processor bus 1221 it is possible to ensure cache-coherency for the processor cores 1263 and 1243 even if one of those processor
20 cores is temporarily prevented from issuing memory transfer requests onto that bus 1221.

Compared to the worst case slow down of a real-time task by ~20x of the NGMP architecture of figure 1 when all processor cores are actively processing tasks [3], the architecture in figure 12 can achieve a ~2x increase (over real-time computing work
25 performed on a single core) when all processor cores are actively processing tasks. If all real-time tasks must run at full speed, NGMP must disable all but one core resulting in a single-core system, where as in figure 12 only one core needs to be disabled resulting in ~3x increase in real-time performance.

30 Furthermore, figure 12 can achieve similar levels of non real-time performance when running 1, 2, 3 or 4 concurrent non real-time tasks on cores 1230, 1231, 1342, 1263.

Advantageously, unlike real-time multi-core solutions that use Time Triggered Protocol

interconnects with static schedulers between the processor cores and main-memory (SDRAM), figure 12 can dynamically optimize its memory subsystem in response to the requirements of the system at any instant to maximize system performance.

- 5 Figure 13 is a schematic block diagram illustrating an architecture 1300 according to a preferred embodiment of the present invention. Figure 13 illustrates a computing device comprising:

five sub-computing devices 1310, 1320, 1330, 1350, 1360 each sub-computing device comprising:

- 10 at least one bus 1311, 1321, 1331, 1351, 1361;
 at least one bus master 1316, 1318, 1323, 1324, 1325, 1326, 1338, 1339,
 1340, 1341, 1342, 1343, 1352, 1353, 1354, 1355, 1362, 1363; and
 at least one memory store 1314, 1315, 1322, 1335, 1337, 1334, 1336, 1345,
 1357;

- 15 in which at least one of the at least one sub-computing devices 1310 comprises:

 at least two bus masters 1316, 1318; and
 a means to enable or disable at least one 1318 of the at least two bus
 masters from issuing memory transfer requests to the bus 1311;

 in which the third sub-computing device 1330 comprises:

- 20 at least one memory store 1346; and
 none of the at least one memory stores 1314, 1315, 1322, 1335, 1337, 1334,
 1336, 1345, 1357 of the other sub-computing device 1310, 1320, 1350,
 1360 has as many elements as that memory store 1331; and

 in which at least one memory store 1371 that has two bus slave interfaces is

- 25 implemented using a true dual port SRAM, in which:

 a first bus slave interface of that memory store 1357 is connected to a first
 bus 1351 of a fourth sub-computing device 1350;

 a second bus slave interface of that memory store 1357 is connected to a
 fifth bus 1361 of a second sub-computing device 1360; and

- 30 the execution time of memory transfer requests on each bus slave interface
 is not influenced by the memory transfer activity on any of the other at least
 two bus slave interfaces.

The dotted-line rectangle 1301 delineates the boundary between modules which are on-chip and modules which are off-chip. For example SDRAM module 1370 is off-chip and SDRAM memory controller module 1371 is on-chip.

5 The first sub-computing device 1310 comprises:

a 32-bit wide AHB bus 1311 being adapted to execute a policy selected from the group comprising:

enabling all bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus;

10 enabling some of the bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus;

enabling at most one of the bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus; and

a 128-bit wide AHB bus 1312;

15 a 32-bit wide AHB bus 1391;

a bidirectional bridge 1318 connected to bus 1331 and bus 1391 being actuable to enable or disable issuing memory transfer requests across that bridge in one or both directions;

20 a unidirectional bridge 1311 connected to bus 1312 and bus 1345 being actuable to enable or disable issuing memory transfer requests across that bridge to bus 1345;

a processor core 1316 with L1 instruction cache and L1 data cache which is adapted to snoop on memory transfer requests on bus 1331;

a single port SRAM store 1315 connected to bus 1311;

a peripheral 1319 connected to bus 1391;

25 a true dual port SRAM store 1314 connected to bus 1311 and connected to a 128-bit wide bus connected to DMA module 1313; and

a DMA module with a bus slave interface connected to bus 1311, a first bus master interface connected to the bus connected to SRAM 1314 and a second bus master interface connected to bus 1312.

30

The second sub-computing device 1320 comprises:

a 32-bit wide AHB bus 1321 being adapted to execute a policy selected from the group comprising:

enabling all bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus;

enabling some of the bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus;

5 enabling at most one of the bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus; and

a 32-bit wide AHB bus 1392;

a bidirectional bridge 1326 connected to bus 1321 and bus 1392 being actuable to enable or disable issuing memory transfer requests across that bridge in one or both
10 directions;

a processor core 1325 with L1 instruction cache and L1 data cache which is adapted to snoop on memory transfer requests on bus 1331;

a peripheral 1327 connected to bus 1392.

15 The third sub-computing device 1330 comprises:

a 128-bit wide AHB bus 1331 being adapted to execute a policy selected from the group comprising:

enabling all bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus;

20 enabling some of the bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus;

enabling at most one of the bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus; and

a 32-bit wide AHB bus 1393;

25 a bidirectional bridge 1343 connected to bus 1331 and bus 1393 being actuable to enable or disable issuing memory transfer requests across that bridge in one or both directions;

a processor core 1342 with L1 instruction cache and L1 data cache which is adapted to snoop on memory transfer requests on bus 1331;

30 a peripheral 1344 connected to bus 1393;

optionally a level 2 cache module 1333 with separate instruction and data partitions connected to bus 1331 and 1345; and

an optional FLASH memory store 1346 that has more elements of memory than the

SRAM memory stores 1314, 1315, 1322, 1335, 1337, 1334, 1336, 1345, 1357;
an optional SDRAM memory controller 1371 with memory store 1370.

The fourth sub-computing device 1350 comprises:

- 5 a 32-bit wide AHB bus 1351 being adapted to execute a policy selected from the group comprising:
 - enabling all bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus;
 - enabling some of the bus masters connected to that bus of that sub-
 - 10 computing device to issue memory transfer requests onto that bus;
 - enabling at most one of the bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus; and
- a 32-bit wide AHB bus 1394;
- a bidirectional bridge 1355 connected to bus 1351 and bus 1394 being actuable to
- 15 enable or disable issuing memory transfer requests across that bridge in one or both directions;
- a processor core 1354 with L1 instruction cache and L1 data cache which is adapted to snoop on memory transfer requests on bus 1331; and
- a peripheral 1356 connected to bus 1394.

20

The fifth sub-computing device 1360 comprises:

- a 32-bit wide AHB bus 1361 being adapted to execute a policy selected from the group comprising:
 - enabling all bus masters connected to that bus of that sub-computing device
 - 25 to issue memory transfer requests onto that bus;
 - enabling some of the bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus;
 - enabling at most one of the bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus; and
- 30 a 32-bit wide AHB bus 1395;
- a bidirectional bridge 1363 connected to bus 1361 and bus 1395 being actuable to enable or disable issuing memory transfer requests across that bridge in one or both directions;

a processor core 1362 with L1 instruction cache and L1 data cache which is adapted to snoop on memory transfer requests on bus 1331; and
a peripheral 1364 connected to bus 1395.

- 5 The first sub-computing device 1310 and second sub-computing device 1320 are connected by a true dual port SRAM store 1322 connected to bus 1311 and bus 1321.

The first sub-computing device 1310 and third sub-computing device 1330 are connected by:

- 10 a true multi-port SRAM store 1337 connected to bus 1311 and bus 1331;
a unidirectional bridge 1323 connected to bus 1311 and bus 1331 being actuable to enable issuing memory transfer requests received from bus masters on 1331 to bus 1311;
a unidirectional bridge 1324 connected to bus 1311 and bus 1331 being actuable to
15 enable issuing memory transfer requests received from bus masters on 1311 to bus 1331.

The first sub-computing device 1310 and sixth sub-computing device 1330 are optionally connected by a true multi-port SRAM store 1337 connected to bus 1311 and bus 1361.

20

The second sub-computing device 1320 and third sub-computing device 1330 are connected by:

- a true dual port SRAM store 1335 connected to bus 1321 and bus 1331;
a unidirectional bridge 1338 connected to bus 1321 and bus 1331 being actuable to
25 enable issuing memory transfer requests received from bus masters on 1331 to bus 1321; and
a unidirectional bridge 1339 connected to bus 1321 and bus 1331 being actuable to enable issuing memory transfer requests received from bus masters on 1321 to bus 1331.

30

The second sub-computing device 1320 and fourth sub-computing device 1350 are connected by a true dual port SRAM store 1334 connected to bus 1321 and bus 1351.

The third sub-computing device 1330 and fourth sub-computing device 1350 are connected by:

- a true dual port SRAM store 1345 connected to bus 1331 and bus 1351;
- a unidirectional bridge 1340 connected to bus 1331 and bus 1351 being actuable to enable issuing memory transfer requests received from bus masters on 1351 to bus 1331; and
- a unidirectional bridge 1341 connected to bus 1331 and bus 1351 being actuable to enable issuing memory transfer requests received from bus masters on 1331 to bus 1351.

The third sub-computing device 1330 and fifth sub-computing device 1360 are connected by:

- a true dual port SRAM store 1345 connected to bus 1331 and bus 1361;
- a unidirectional bridge 1352 connected to bus 1331 and bus 1361 being actuable to enable issuing memory transfer requests received from bus masters on 1361 to bus 1331; and
- a unidirectional bridge 1353 connected to bus 1331 and bus 1361 being actuable to enable issuing memory transfer requests received from bus masters on 1331 to bus 1361.

The fourth sub-computing device 1350 and fifth sub-computing device 1360 are connected by a true dual port SRAM store 1357 connected to bus 1351 and bus 1361.

A space-wire peripheral 1399 has a first port with its own unique network address connected to bus 1392, and a second port with its own unique network address connected to bus 1394.

In figure 13, no multi-port SRAM is connected to every bus 1311, 1321, 1331, 1341, 1351, 1361. Figure 13 illustrates:

- each sub-computing device is connected to a multi-port memory store;
- each of the multi-port memory stores is connected to at least two of the sub-computing devices; and

each sub-computing device is connected to every other sub-computing device by at least one of the multi-port memory stores; and
none of the multi-port memory stores is connected to every sub-computing device.

- 5 In preferred embodiments of the present invention, DMA 1313 is used to copy executable code (and/or run-time data) for a function stored in main store 1370 into SRAM 1314 while the first sub-computing device 1310 operates in single-core equivalent mode of operation. Specifically, the DMA 1313 memory transfer requests to memory store 1370 do not influence the execution time of memory transfer requests issued onto bus 1311 by
10 other bus masters 1316, 1318.

In preferred embodiments of the present invention, all access to SRAM stores 1314, 1315, 1322, 1335, 1337, 1334, 1336, 1345, 1357 are non-cacheable and thus trivially coherent across all cores. All accesses to the level 2 cache 1333, or the memory stores 1370 and
15 1346 behind the level 2 cache 1333, by processor cores 1316, 1325, 1342, 1354, 1362 maintains cache-coherency because all such memory transfer requests must pass over bus 1331, and each of those processor cores snoop the bus 1331.

In further preferred embodiments of the present invention, each pair of sub-computing
20 devices {1310, 1320}, {1310, 1330}, {1310, 1350}, {1310, 1360}, {1320, 1330}, {1320, 1350}, {1320, 1360}, {1330, 1350}, {1330, 1360}, {1350, 1360} is connected by a different true dual-port memory (not illustrated) permitting low-latency any-to-any communication between sub-computing devices with zero timing interference.

- 25 Figure 14 is a schematic block diagram illustrating an architecture 1400 according to a preferred embodiment of the present invention. Figure 14 illustrates a scalable real-time multi-core computing device comprising:

N sub-computing devices 1410, 1420, 1430, 1440, 1450, 1460, where the value of N is at least 3, each sub-computing device comprising:

- 30 at least one bus 1492, 1493, 1494, 1495, 1496, 1497;
at least one bus master 1411, 1412, 1421, 1422, 1431, 1441, 1442, 1451, 1452, 1461, 1462, 1467, 1468, 1469, 1470, 1471;

the bus slave interface of at least one memory store 1424, 1425, 1434, 1435, 1436, 1437, 1438, 1444, 1445 connected to one of the at least one busses; and

5 at least two unidirectional bus bridges for each unidirectional bus bridge 1467, 1468, 1469, 1470, 1471, each bus interface of that unidirectional bus bridge is connected to the bus 1492, 1493, 1494, 1495, 1496, 1497 of a different one of the sub-computing devices;

10 at least two memory stores, each memory store comprising at least two bus slave interfaces 1424, 1425, 1433, 1434, 1435, 1436, 1437, 1438, 1444, 1445, for each memory store, each bus slave interface is connected to the bus 1492, 1493, 1494, 1495, 1496, 1497 of a different one of the sub-computing devices;

in which:

15 X of the N sub-computing devices 1410, 1420, 1430, 1440, 1450, 1460, 1470 are directly connected to a common bus 1491 by a corresponding bus bridge 1481, 1482, 1483, 1484, 1485, 1486, where the value of X is $2 \leq X \leq N$;

20 two of the sub-computing devices 1430, 1420 are connected to each other by a first memory store 1434 comprising at least two bus slave interfaces; another two sub-computing devices 1430, 1440 are connected to each other by at least a second memory store 1444 comprising at least two bus slave interfaces;

the unordered set of sub-computing devices {1430, 1420} connected to the first memory store is different to the unordered set of sub-computing devices {1430, 1440} connected to the second memory store; and

25 in which each sub-computing device 1410, 1420, 1430, 1440, 1450, 1460, 1470 further comprising:

at least one cache module in each processor core 1411, 1421, 1431, 1441, 1451, 1461; and

30 means for maintaining cache coherency with at least one other cache module on another sub-computing device using bus snooping; and

in which for each memory store 1424, 1425, 1433, 1434, 1435, 1436, 1437, 1438, 1444, 1445 comprising at least two bus slave interfaces, the execution time of memory transfer requests on each bus slave interface is not influenced by the

- memory transfer activity on any of the other at least two bus slave interfaces;
in which one sub-computing device 1460 is directly connected to every other sub-computing device 1410, 1420, 1430, 1440, 1450 by a corresponding bus bridge 1467, 1468, 1469, 1470, 1471;
- 5 in which one sub-computing device 1460 further comprises a memory store 1464, 1465, that sub-computing device being directly connected to every other sub-computing device by a corresponding bus bridge 1467, 1468, 1469, 1470, 1471 to permit memory transfer requests to be issued from each sub-computing device to that memory store 1464, 1465; and
- 10 in which no bus master of any of the sub-computing devices 1410, 1420, 1430, 1440, 1450 can issue memory transfer requests onto any of the at least one busses 1492, 1493, 1494, 1495, 1496, 1497 of any of the other sub-computing devices 1410, 1420, 1430, 1440, 1450.
- 15 The dotted-line rectangle 1401 delineates the boundary between modules which are on-chip and modules which are off-chip. For example module 1464 is off-chip and module 1465 is on-chip.
- The first sub-computing device 1410 comprises:
- 20 a 32-bit wide AHB bus 1492 being adapted to execute a policy selected from the group comprising:
- enabling all bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus;
- enabling some of the bus masters connected to that bus of that sub-
- 25 computing device to issue memory transfer requests onto that bus;
- enabling at most one of the bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus; and
- a 32-bit wide AHB bus 1402;
- a unidirectional bridge 1481 connected to bus 1492 and bus 1491 being actuable to
- 30 enable or disable issuing memory transfer requests across that bridge to bus 1491;
- a processor core 1411 with L1 instruction cache (not illustrated) and L1 data cache (not illustrated) which is adapted to snoop on memory transfer requests (illustrated by a line with a solid arrow on one end, and a open circle on the other end) on bus

1497 and optionally to also snoop on bus 1491 (not illustrated);
a peripheral 1413 which is connected to bus 1402; and
an interrupt controller (not illustrated), in which interrupts from the peripherals
1413 of that sub-computing device are routed to the processor core 1411.

5

The second sub-computing device 1420 comprises:

a 32-bit wide AHB bus 1493 being adapted to execute a policy selected from the
group comprising:

10

enabling all bus masters connected to that bus of that sub-computing device
to issue memory transfer requests onto that bus;

enabling some of the bus masters connected to that bus of that sub-
computing device to issue memory transfer requests onto that bus;

enabling at most one of the bus masters connected to that bus of that sub-
computing device to issue memory transfer requests onto that bus; and

15

a 32-bit wide AHB bus 1403;

a unidirectional bridge 1482 connected to bus 1493 and bus 1491 being actuable to
enable or disable issuing memory transfer requests across that bridge to bus 1491;

a processor core 1421 with L1 instruction cache (not illustrated) and L1 data cache
(not illustrated) which is adapted to snoop on memory transfer requests (illustrated

20

by a line with a solid arrow on one end, and a open circle on the other end) on bus
1497 and optionally to also snoop on bus 1491 (not illustrated);

a peripheral 1423 which is connected to bus 1403; and

an interrupt controller (not illustrated), in which interrupts from the peripherals
1423 of that sub-computing device are routed to the processor core 1422.

25

The third sub-computing device 1430 comprises:

a 32-bit wide AHB bus 1494 being adapted to execute a policy selected from the
group comprising:

30

enabling all bus masters connected to that bus of that sub-computing device
to issue memory transfer requests onto that bus;

enabling some of the bus masters connected to that bus of that sub-
computing device to issue memory transfer requests onto that bus;

enabling at most one of the bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus; and
a unidirectional bridge 1483 connected to bus 1494 and bus 1491 being actuable to enable or disable issuing memory transfer requests across that bridge to bus 1491;
5 a processor core 1431 with L1 instruction cache (not illustrated) and L1 data cache (not illustrated) which is adapted to snoop on memory transfer requests (illustrated by a line with a solid arrow on one end, and a open circle on the other end) on bus 1497 and optionally to also snoop on bus 1491 (not illustrated);
an interrupt controller (not illustrated);
10 a SRAM store 1433 connected to bus 1494; and
an optional FLASH memory store 1432.

The fourth sub-computing device 1440 comprises:

a 32-bit wide AHB bus 1495 being adapted to execute a policy selected from the
15 group comprising:
enabling all bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus;
enabling some of the bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus;
20 enabling at most one of the bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus; and
a 32-bit wide AHB bus 1404;
a unidirectional bridge 1484 connected to bus 1495 and bus 1491 being actuable to enable or disable issuing memory transfer requests across that bridge to bus 1491;
25 a processor core 1441 with L1 instruction cache (not illustrated) and L1 data cache (not illustrated) which is adapted to snoop on memory transfer requests (illustrated by a line with a solid arrow on one end, and a open circle on the other end) on bus 1497 and optionally to also snoop on bus 1491 (not illustrated);
a peripheral 1443 which is connected to bus 1404; and
30 an interrupt controller (not illustrated), in which interrupts from the peripherals 1443 of that sub-computing device are routed to the processor core 1441.

The fifth sub-computing device 1450 comprises:

a 32-bit wide AHB bus 1496 being adapted to execute a policy selected from the group comprising:

enabling all bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus;

5 enabling some of the bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus;

enabling at most one of the bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus; and

a 32-bit wide AHB bus 1405;

10 a unidirectional bridge 1485 connected to bus 1496 and bus 1491 being actuable to enable or disable issuing memory transfer requests across that bridge to bus 1491;

a processor core 1451 with L1 instruction cache (not illustrated) and L1 data cache (not illustrated) which is adapted to snoop on memory transfer requests (illustrated by a line with a solid arrow on one end, and a open circle on the other end) on bus

15 1497 and optionally to also snoop on bus 1491 (not illustrated);

a peripheral 1453 which is connected to bus 1405; and

an interrupt controller (not illustrated), in which interrupts from the peripherals 1453 of that sub-computing device are routed to the processor core 1451.

20 The sixth sub-computing device 1460 comprises:

a 128-bit wide AHB bus 1497 being adapted to execute a policy selected from the group comprising:

enabling all bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus;

25 enabling some of the bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus;

enabling at most one of the bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus; and

a 32-bit wide AHB bus 1406;

30 a unidirectional bridge 1486 connected to bus 1497 and bus 1491 being actuable to enable or disable issuing memory transfer requests across that bridge to bus 1491;

a processor core 1461 with L1 instruction cache (not illustrated) and L1 data cache (not illustrated) which is adapted to snoop on memory transfer requests (illustrated

by a line with a solid arrow on one end, and a open circle on the other end) on bus 1497 and optionally to also snoop on bus 1491 (not illustrated);
a peripheral 1463 which is connected to bus 1406;
an interrupt controller (not illustrated), in which interrupts from the peripherals
5 1463 of that sub-computing device are routed to the processor core 1461;
an optional L2 cache module 1466;
an optional FLASH memory store 1465;
a SDRAM memory controller 1472; and
a SDRAM store 1464.

10

A memory subsystem 1480 comprises:

a 128-bit wide AHB bus 1491 being adapted to execute a policy selected from the group comprising:

15

enabling all bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus;

enabling some of the bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus;

enabling at most one of the bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus; and

20

a SDRAM memory controller 1473; and

a SDRAM store 1490.

The first sub-computing device 1410 and second sub-computing device 1420 are connected by a true dual port SRAM store 1424 connected to bus 1492 and bus 1493.

25

The first sub-computing device 1410 and third sub-computing device 1430 are connected by a true dual port SRAM store 1435 connected to bus 1492 and bus 1494.

The first sub-computing device 1410 and fourth sub-computing device 1440 are connected by a true dual port SRAM store 1425 connected to bus 1492 and bus 1495.

30

The first sub-computing device 1410 and fifth sub-computing device 1450 are connected by a true dual port SRAM store 1480 connected to bus 1492 and bus 1496.

The first sub-computing device 1410 and sixth sub-computing device 1460 are connected by a bidirectional bridge 1467 connected to bus 1492 and bus 1497 being actuable to enable or disable issuing memory transfer requests across that bridge in one or both directions.

5

The second sub-computing device 1420 and third sub-computing device 1430 are connected by a true dual port SRAM store 1434 connected to bus 1493 and bus 1494.

10 The second sub-computing device 1420 and fourth sub-computing device 1440 are connected by a true dual port SRAM store 1439 connected to bus 1493 and bus 1495.

The second sub-computing device 1420 and fifth sub-computing device 1450 are connected by a true dual port SRAM store 1445 connected to bus 1493 and bus 1495.

15 The second sub-computing device 1420 and sixth sub-computing device 1460 are connected by a bidirectional bridge 1468 connected to bus 1493 and bus 1497 being actuable to enable or disable issuing memory transfer requests across that bridge in one or both directions.

20 The third sub-computing device 1430 and fourth sub-computing device 1440 are connected by a true dual port SRAM store 1436 connected to bus 1494 and bus 1495.

The third sub-computing device 1430 and fifth sub-computing device 1450 are connected by a true dual port SRAM store 1437 connected to bus 1494 and bus 1496.

25

The third sub-computing device 1430 and sixth sub-computing device 1460 are connected by:

a true dual port SRAM store 1438 connected to bus 1494 and bus 1497; and
a bidirectional bridge 1469 connected to bus 1494 and bus 1497 being actuable to
30 enable or disable issuing memory transfer requests across that bridge in one or both directions.

The fourth sub-computing device 1440 and fifth sub-computing device 1450 are connected

by a true dual port SRAM store 1444 connected to bus 1495 and bus 1496.

The fourth sub-computing device 1440 and sixth sub-computing device 1460 are connected by a bidirectional bridge 1470 connected to bus 1495 and bus 1497 being
5 actuatable to enable or disable issuing memory transfer requests across that bridge in one or both directions.

The fifth sub-computing device 1450 and sixth sub-computing device 1460 are connected by a bidirectional bridge 1471 connected to bus 1496 and bus 1497 being actuatable to
10 enable or disable issuing memory transfer requests across that bridge in one or both directions.

Figure 14 illustrates that communication is possible between every sub-computing device over the bus bridges 1467, 1468, 1469, 1470, 1471.

15 Figure 14 illustrates that the first, second, third, fourth and fifth sub-computing devices are each connected to each other by a true dual-port SRAM module 1424, 1425, 1434, 1435, 1436, 1437, 1444, 1445. This connectivity makes those sub-computing devices particularly well suited for running real-time tasks. The third and sixth sub-computing
20 module are also connected to each other by a true dual-port SRAM module 1438 to permit tasks running on core 1461 to communicate with tasks running on core 1431 at any time without unwanted timing interference.

Memory store 1464 can be accessed by all 6 sub-computing devices. In further preferred
25 embodiments of the present invention, the memory store 1464 is optimized for non real-time memory access (such as open page mode of operation for the memory controller 1472). Likewise the processor core 1461 and its cache and the level 2 cache 1466 can all be optimized for non real-time tasks.

30 Memory store 1490 can also be accessed by all 6 sub-computing devices. In further preferred embodiments of the present invention the memory store 1490 is optimized for real-time memory access (such as by using a time-analyzable closed-page mode of operation for the memory controller 1473). Advantageously memory subsystem 1480 can

be adapted at run time to control which of the sub-computing systems may access the memory store 1490. For example, when running time and space partitioning operating systems, a first time-slot may grant exclusive access to the memory store 1490 to sub-computing device 3, and a second time-slot may grant shared access to the memory store 1490 to sub-computing devices 2 and 4. In this way the worst case execution time performance of the memory transfer requests to the time-analyzable memory subsystem 1480 can be adjusted to meet the needs of the active tasks in any given time-slot.

Figure 14 illustrates that high speed, high-bandwidth, true dual-port SRAM modules can be adapted to permit high speed time-analyzable communications between selected sub-computing device in a way that delivers exceptional system performance while controlling hardware manufacture costs. With regard to controlling cost, figure 14 does not implement full connectivity of dual-port SRAM across all 6 sub-computing devices. In preferred embodiments of the present invention high-frequency real-time sensitive peripherals are assigned to the first and fourth sub-computing devices 1410 and 1440, lower-frequency bulk-data real-time sensitive peripherals are assigned to sub-computing devices 1420 and 1450, high-level management of at least some real time peripherals are assigned to the third sub-computing device and non real-time peripherals are assigned to the sixth sub-computing device.

20

In further preferred embodiments of the present invention, the first and fourth sub-computing devices are adapted to access the same RTOS executable code stored in memory store 1425 to reduce manufacturing costs and to permit SMP RTOS instances to be run, that RTOS executable code being optimized for high-frequency priority driven event scheduling. The second and fifth sub-computing devices 1420 and 1450 are adapted to access the same RTOS executable code stored in memory store, that RTOS executable code being optimized for time and space scheduling. High-level control and feedback communications between the real-time peripherals and the third sub-computing device can occur rapidly in a time-analyzable way over dual-port SRAM stores 1434, 1435, 1436, 1437. Furthermore, a real-time task running on processor core 1421 in the second sub-computing device can rapidly access last known sensor data received on high-frequency peripherals 1413 and 1443 over memory stores 1424 and 1425 and communicate actuator data back over those memory stores back to those high frequency peripherals.

30

Advantageously the absence of bus bridges between the first, second, third, fourth and fifth sub-computing devices:

- 5 reduces manufacture cost (by reducing the amount of circuitry that must be implemented); and
- simplifies multi-core worst case execution timing analysis by eliminating potential sources of unwanted timing interference between those five sub-computing devices.

- 10 Advantageously, the processor cores 1411, 1421, 1431, 1441, 1451 can run mixed-criticality real-time and non real-time software, exploiting time-analyzable memory store 1490 and high bandwidth memory store 1464 as appropriate for the currently active task without constraining the choice of real-time or non-real-time task on the other cores, thereby simplifying scheduling and increasing overall system performance.

15

In an alternate single processor core 613 variation of the present invention illustrated in figure 6, the processor cores 611, 612, and 614 of figure 6 are removed, the bus master peripherals remain connected on processor bus A 101, and all bus slave peripherals previously connected (directly or indirectly) to processor bus A are disconnected from processor bus A and connected within sub-computing device 608. Figure 6 then illustrates a computing device comprising:

20

- at least one sub-computing 608 device, each sub-computing device comprising:

- at least one bus 601;

- at least one bus master 601; and

25

- at least one memory store 233; and

- in which at least one of the at least one sub-computing devices comprises:

- at least two bus masters 613, 606; and

- a means to enable or disable at least one 606 of the at least two bus masters from issuing memory transfer requests to the bus 601.

30

In further preferred embodiments of the present invention, the unidirectional AHB bridge 606 is removed, ensuring bus master peripherals within sub-computing device 609 can only access SDRAM 222, where as the processor cores and bus masters on processor bus

B can access both SDRAM 222 and 232.

In further preferred embodiments of the present invention the bus masters on processor bus B are temporarily halted while a real-time task on the processor core 613 accesses
5 SDRAM 222 to ensure time-analysability of those memory accesses.

In an alternate embodiment of figure 14, the 5 bidirectional bus bridges 1467, 1468, 1469, 1470, 1471 are replaced with unidirectional bus bridges, with the bus master interface of each of the 5 bus bridges attached to the common bus 1497. Advantageously, no bus
10 master of any of the 5 sub-computing devices 1410, 1420, 1430, 1440, 1450 can issue memory transfer requests to any of the at least one bus of any of the 4 other sub-computing devices 1410, 1420, 1430, 1440, 1450, further simplifying worst case execution time analysis of those 5 sub-computing systems, while still permitting each bus master to access shared memory stores 1464 and 1490.

15

Various embodiments of the invention may be embodied in many different forms, including computer program logic for use with a processor (eg., a microprocessor, microcontroller, digital signal processor, or general purpose computer), programmable logic for use with a programmable logic device (eg., a field programmable gate array
20 (FPGA) or other PLD), discrete components, integrated circuitry (eg., an application specific integrated circuit (ASIC)), or any other means including any combination thereof. In an exemplary embodiment of the present invention, predominantly all of the communication between users and the server is implemented as a set of computer program instructions that is converted into a computer executable form, stored as such in a
25 computer readable medium, and executed by a microprocessor under the control of an operating system.

Computer program logic implementing all or part of the functionality where described herein may be embodied in various forms, including a source code form, a computer
30 executable form, and various intermediate forms (e.g., forms generated by an assembler, compiler, linker, or locator). Source code may include a series of computer program instructions implemented in any of various programming languages (e.g., an object code, an assembly language, or a high-level language such as ADA SPARK, Fortran, C, C++,

JAVA, Ruby, or HTML) for use with various operating systems or operating environments. The source code may define and use various data structures and communication messages. The source code may be in a computer executable form (e.g., via an interpreter), or the source code may be converted (e.g., via a translator, assembler, or compiler) into a computer executable form.

The computer program may be fixed in any form (e.g., source code form, computer executable form, or an intermediate form) either permanently or transitorily in a tangible storage medium, such as a semiconductor memory device (e.g., a RAM, ROM, PROM, EEPROM, or Flash-Programmable RAM), a magnetic memory device (e.g., a diskette or fixed disk), an optical memory device (e.g., a CD-ROM or DVD-ROM), a PC card (e.g., PCMCIA card), or other memory device. The computer program may be fixed in any form in a signal that is transmittable to a computer using any of various communication technologies, including, but in no way limited to, analog technologies, digital technologies, optical technologies, wireless technologies (e.g., Bluetooth), networking technologies, and inter-networking technologies. The computer program may be distributed in any form as a removable storage medium with accompanying printed or electronic documentation (e.g., shrink wrapped software), preloaded with a computer system (e.g., on system ROM or fixed disk), or distributed from a server or electronic bulletin board over the communication system (e.g., the internet or world wide web).

Hardware logic (including programmable logic for use with a programmable logic device) implementing all or part of the functionality where described herein may be designed using traditional manual methods, or may be designed, captured, simulated, or documented electronically using various tools, such as computer aided design (CAD), a hardware description language (e.g., VHDL or AHDL), or a PLD programming language (e.g., PALASM, ABEL, or CUPL).

Programmable logic may be fixed either permanently or transitorily in a tangible storage medium, such as a semiconductor memory device (e.g., a RAM, ROM, PROM, EEPROM, or Flash-Programmable RAM), a magnetic memory device (e.g., a diskette or fixed disk), an optical memory device (e.g., a CD-ROM or DVD-ROM), or other memory device. The programmable logic may be fixed in a signal that is transmittable to a computer using any

of various communication technologies, including, but in no way limited to, analog technologies, digital technologies, optical technologies, wireless technologies (e.g., Bluetooth), networking technologies, and internetworking technologies. The programmable logic may be distributed as a removable storage medium with
5 accompanying printed or electronic documentation (e.g., shrink wrapped software), preloaded with a computer system (e.g., on system ROM or fixed disk), or distributed from a server or electronic bulletin board over the communication system (e.g., the internet or world wide web).

10 Throughout this specification, the words “comprise”, “comprised”, “comprising” and “comprises” are to be taken to specify the presence of stated features, integers, steps or components but does not preclude the presence or addition of one or more other features, integers, steps, components or groups thereof.

15 REFERENCES

[1] Aeroflex Gaisler. NGMP Specification, Next Generation Multi-Purpose Microprocessor. Report, European Space Agency, Feb 2010. Contract 22279/09/NL/JK.
<http://microelectronics.esa.int/ngmp/NGMP-SPEC-0001-i1r4.pdf>

[2] ARM AMBA Specification (Rev 2.0), 1999. ARM IHI 0011A

20 [3] F. J. Cazorla, R. Gioiosa, M. Fernandez, E. Quinones, M. Zulianello, and L. Fossati. Multicore OS Benchmark (for NGMP). Final report, Barcelona Supercomputing Centre, 2012. Under contract RFQ-3-13153/10/NL/JK.
http://microelectronics.esa.int/ngmp/MulticoreOSBenchmark-FinalReport_v7.pdf

[4] Aeroflex Gaisler, GRLIB IP Core User’s Manual Version 1.1.0 - B4108, June

25 2011

[5] Altera. Arria V Device Handbook, Volume 1: Device Overview and Datasheet. Altera, Feb 2012.

Claims

1. A computing device for performing real-time and mixed criticality tasks comprising:
 - at least one sub-computing device, each sub-computing device comprising:
 - at least one bus;
 - at least one bus master;
 - at least one memory store; and
 - in which at least one of the at least one sub-computing devices comprises:
 - at least two bus masters; and
 - a means to enable or disable at least one of the at least two bus masters from issuing memory transfer requests onto the bus without resetting the at least one of the at least two bus masters.
2. A computing device as claimed in claim 1, in which at least one of the at least one sub-computing devices that comprises at least two bus masters further comprises a means to grant exclusive access to one of the at least two bus masters to issue memory transfer requests to onto the bus.
3. A computing device as claimed in claim 1 or claim 2 comprising:
 - at least two sub-computing devices,
 - in which at least one of the at least one two sub-computing devices comprises:
 - at least one bus master; and
 - at least one unidirectional bus bridge,
 - one of the at least one unidirectional bus bridges, having:
 - its bus master interface connected to one of the at least one busses of this sub-computing device; and
 - its bus slave interface connected to one of the at least one busses of another sub-computing device.
4. A computing device as claimed in claim 3 in which at least one of the at least one two sub-computing devices comprises:
 - at least one bus master; and
 - at least one unidirectional bus bridge,

one of the at least one unidirectional bus bridges:

having its bus master interface connected to one of the at least one
busses of this sub-computing device;

5 having its bus slave interface connected to one of the at least one
busses of another sub-computing device;

being actuable to enable or disable issuing memory transfer requests
to the bus by refusing to pass memory transfer requests across the
bridge; and

10 in which refusing to pass is implemented by sending to the bus
master that issued the memory transfer request one of:

a RETRY response;

a SPLIT response; and

an ERROR response.

15 5. A computing device as claimed in claim 3 in which the refusing to pass is
implemented by the unidirectional bus bridge performing a single action selected
from the group comprising:

delaying the memory transfer request for an indefinite period of time;

20 delaying the memory transfer request for at least 1 clock cycle then sending
a RETRY response to the bus master that issued the memory transfer
request;

delaying the memory transfer request for at least 1 clock cycle then sending
a SPLIT response to the bus master that issued the memory transfer
request; and

25 delaying the memory transfer request for at least 1 clock cycle then sending
an error response to the bus master that issued the memory transfer request.

6. A computing device as claimed in any one of claims 3 to 5, in which each sub-
computing device further comprises:

30 at least one cache module; and

means for maintaining cache coherency with at least one other cache
module on another sub-computing device when at least one of the
unidirectional bus bridge connecting the two sub-computing devices is in a

state in which it is not passing memory transfer requests issued on one bus to the other bus.

7. A computing device as claimed in any one of claims 3 to 6 in which:

5 the first sub-computing device comprises at least one memory store; and none of the at least one memory stores of the second sub-computing device has as many elements as that memory store of the first sub-computing device.

8. A computing device as claimed in any one of claims 3 to 7 further comprising:

10 at least one memory store comprising two bus slave interfaces, in which:

a first bus slave interface of that memory store is connected to a first bus of a first sub-computing device;

a second bus slave interface of that memory store is connected to a second bus of a second sub-computing device; and

15 the execution time of memory transfer requests on each bus slave interface is not influenced by the memory transfer activity on any of the other at least two bus slave interfaces.

9. A computing device as claimed in any one of claims 3 to 7 further comprising:

20 at least N sub-computing devices, where N is larger than 2;

at least one memory store comprising at least X bus slave interfaces where $2 < X < N$, in which:

a first bus slave interface of a memory store is connected to a first bus of a first sub-computing device;

25 a second bus slave interface of that memory store is connected to a second bus of a second sub-computing device; and

the execution time of memory transfer requests on each bus slave interface is not influenced by the memory transfer activity on any of the other at least two bus slave interfaces.

30

10. A computing device as claimed in any one of claims 1 to 9 further comprising:

at least two cache modules, a first cache module of the at least two cache modules having an internally contiguous input address space of at least 1 kilobyte in length,

in which:

the output address space of the first cache module is not mapped to the input address space of any other of the at least two cache modules; and the output address space of any of the other at least two cache modules is not mapped to the input address space of first cache module;

at least one bus master, a first bus master having an contiguous output address space of at least 2 kilobyte in length, in which:

at least one bus master can perform memory transfer requests with both the first cache module and another cache module of the at least two cache modules;

at least one memory store comprising an internally contiguous input address space of at least 2 kilobytes, in which: a first contiguous subset of the internally contiguous input address space of at least 1 kilobyte in length of a first memory store is:

bijectionally mapped as cacheable with at least a contiguous subset of the internally contiguous input address space of the first cache module; and bijectionally mapped as cacheable with at least a subset of the contiguous output address space of the first bus master.

11. A computing device as claimed in any claim 10, in which:

a second cache module of the at least two cache modules comprises an internally contiguous input address space of at least 1 kilobyte in length;

a second contiguous subset of the internally contiguous input address space of at least 1 kilobyte in length of the first memory store is:

bijectionally mapped as cacheable with at least a subset of the internally contiguous input address space of the second cache module; and bijectionally mapped as cacheable with at least a subset of the contiguous output address space of the first bus master; and

in which:

the contiguous subset of the internally contiguous input address space of the first memory store which is mapped as cacheable to the first cache module does not overlap with the contiguous subset of the internally contiguous input address space of the first memory store which is mapped as cacheable

to the second cache module; and
the subset of the internally contiguous input address space of the first cache module which is mapped to the first bus master does not overlap with the subset of the internally contiguous input address space of the second cache module which is mapped to the first bus master.

12. A computing device as claimed in claim 10 or claim 11 comprising at least two memory stores, each of which comprises an internally contiguous input address space of at least 2 kilobytes in length which does not overlap with the internally contiguous input address space of any other memory store, in which:

a first contiguous subset of the internally contiguous input address space of at least 1 kilobyte in length of a second memory store is:

bijectionally mapped as cacheable with at least a contiguous subset of the internally contiguous input address space of a third cache module; and

bijectionally mapped as cacheable with at least a subset of the contiguous output address space of the first bus master; and

at least one bus master can perform memory transfer requests with the first, second and third cache module.

13. A computing device as claimed in any one of claims 10 to 12, further comprising at least one address translation unit, each address translation unit:

having an internally contiguous input address space which is partitioned into at least two portions, comprising:

a first portion A of which is bijectionally mapped to within the input address space of the first cache module;

another portion B of which is striped across X of the at least 2 cache modules, where $X > 1$; and

in which:

the portion B of address space is partitioned into several consecutive strips, each stripe further partitioned into blocks of at least one cache-line in length; and

each of the X cache modules has one of the blocks

exclusively mapped in its input address space.

14. A computing device as claimed in claim 13, in which one of the at least one address translation units is adapted to perform a bijective mapping of its input address space to its output address space.

15. A method of controlling a sub-computing device of a computing device as claimed in any one of claims 1 to 14 in which the method is adapted to execute a policy selected from the group comprising:

10 enabling all bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus;
enabling some of the bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus; and
enabling at most one of the bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus.

16. A process of controlling a sub-computing device of a computing device as claimed in any one of claims 1 to 15 in which the process is adapted to execute a policy selected from the group comprising:

20 enabling all bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus;
enabling some of the bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus; and
enabling at most one of the bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus.

17. A computing device comprising:
at least two cache modules, a first cache module of the at least two cache modules having an internally contiguous input address space of at least 1 kilobyte in length,
in which:

30 the output address space of the first cache module is not mapped to the input address space of any other of the at least two cache modules; and
the output address space of any of the other at least two cache modules is

not mapped to the input address space of first cache module;
at least one bus master, a first bus master having an contiguous output address
space of at least 2 kilobyte in length, in which:

at least one bus master can perform memory transfer requests with both the
first cache module and another cache module of the at least two cache
modules;

at least one memory store comprising an internally contiguous input address space
of at least 2 kilobytes, in which: a first contiguous subset of the internally
contiguous input address space of at least 1 kilobyte in length of a first memory
store is:

bijectionally mapped as cacheable with at least a contiguous subset of the
internally contiguous input address space of the first cache module; and
bijectionally mapped as cacheable with at least a subset of the contiguous
output address space of the first bus master.

18. A computing device as claimed in claim 17, in which:

a second cache module of the at least two cache modules comprises an internally
contiguous input address space of at least 1 kilobyte in length;

a second contiguous subset of the internally contiguous input address space of at
least 1 kilobyte in length of the first memory store is:

bijectionally mapped as cacheable with at least a subset of the internally
contiguous input address space of the second cache module; and
bijectionally mapped as cacheable with at least a subset of the contiguous
output address space of the first bus master; and

in which:

the contiguous subset of the internally contiguous input address space of the
first memory store which is mapped as cacheable to the first cache module
does not overlap with the contiguous subset of the internally contiguous
input address space of the first memory store which is mapped as cacheable
to the second cache module; and

the subset of the internally contiguous input address space of the first cache
module which is mapped to the first bus master does not overlap with the
subset of the internally contiguous input address space of the second cache

module which is mapped to the first bus master.

19. A computing device as claimed in claim 17 or claim 18, comprising at least two memory stores, each of which comprises an internally contiguous input address space of at least 2 kilobytes in length which does not overlap with the internally contiguous input address space of any other memory store, in which:

a first contiguous subset of the internally contiguous input address space of at least 1 kilobyte in length of a second memory store is:

bijectionally mapped as cacheable with at least a contiguous subset of the internally contiguous input address space of a third cache module; and

bijectionally mapped as cacheable with at least a subset of the contiguous output address space of the first bus master; and

at least one bus master can perform memory transfer requests with the first, second and third cache module.

20. A computing device as claimed in any one of claims 17 to 19, further comprising at least one address translation unit, each address translation unit:

having an internally contiguous input address space which is partitioned into at least two portions, comprising:

a first portion A of which is bijectionally mapped to within the input address space of the first cache module;

another portion B of which is striped across X of the at least 2 cache modules, where $X > 1$; and

in which:

the portion B of address space is partitioned into several consecutive strips, each stripe further partitioned into blocks of at least one cache-line in length; and

each of the X cache modules has one of the blocks exclusively mapped in its input address space.

21. A computing device as claimed in claim 20, in which one of the at least one address translation units is adapted to perform a bijective mapping of its input

address space to its output address space.

22. A representation in a hardware description language of a computing device as claimed in any one of claims 1 to 14.

5

23. A process emulating a computing device as claimed in any one of claims 1 to 14.

24. A signal carrying a representation in a hardware description language of a computing device as claimed in any one of claims 1 to 14.

10

25. A machine readable substrate carrying a representation in a hardware description language of a computing device as claimed in any one of claims 1 to 14.

15

26. A representation in a hardware description language of a computing device as claimed in any one of claims 17 to 21.

27. A process emulating a computing device as claimed in any one of claims 17 to 21.

20

28. A signal carrying a representation in a hardware description language of a computing device as claimed in any one of claims 17 to 21.

29. A machine readable substrate carrying a representation in a hardware description language of a computing device as claimed in any one of claims 17 to 21.

25

30. A computing device for performing real-time and mixed criticality tasks comprising:

N sub-computing devices, where value of N is at least 2, each sub-computing device comprising:

at least one bus;

30

at least one bus master, in which at least one of the at least one bus masters is a processor core;

the bus slave interface of at least one memory store connected to one of the at least one busses and

at least one unidirectional bus bridge, for each unidirectional bus bridge, each bus interface of that unidirectional bus bridge is connected to the bus of a different one of the sub-computing devices;

at least one memory store, each memory store comprising at least two bus slave interfaces, for each memory store, each bus slave interface is connected to the bus of a different one of the sub-computing devices;

in which:

X of the N sub-computing devices are directly connected to a common bus by a corresponding bus bridge where the value of X is $2 \leq X \leq N$;

a first set of two of the sub-computing devices are connected to each other by a first memory store comprising at least two bus slave interfaces.

31. A computing device as claimed in claim 30 comprising:

N sub-computing devices, where value of N is at least 3, each sub-computing device comprising:

at least one bus;

at least one bus master, in which at least one of the at least one bus masters is a processor core;

the bus slave interface of at least one memory store connected to one of the at least one busses and

at least two unidirectional bus bridges, for each unidirectional bus bridge, each bus interface of that unidirectional bus bridge is connected to the bus of a different one of the sub-computing devices;

at least two memory stores, each memory store comprising at least two bus slave interfaces, for each memory store, each bus slave interface is connected to the bus of a different one of the sub-computing devices;

in which:

X of the N sub-computing devices are directly connected to a common bus by a corresponding bus bridge where the value of X is $2 \leq X \leq N$;

a first set of two of the sub-computing devices are connected to each

other by a first memory store comprising at least two bus slave interfaces;

a second set of two sub-computing devices are connected to each other by a second memory store comprising at least two bus slave interfaces; and

the first set of sub-computing devices is different from the second set of sub-computing devices.

32. A computing device as claimed in claim 30 or claim 31, in which each corresponding bus bridge being a unidirectional bus bridge that has its bus master connected to the common bus.

33. A computing device as claimed in any one of claims 30 to 32, in which each sub-computing device further comprises:

at least one cache module; and

means for maintaining cache coherency with at least one other cache module on another sub-computing device.

34. A computing device as claimed in any one of claims 30 to 33, in which for each memory store, the execution time of memory transfer requests on each bus slave interface is not influenced by the memory transfer activity on any of the other at least two bus slave interfaces.

35. A computing device as claimed in any one of claims 31 to 34 in which: a bus of at least one bus of a first sub-computing device and a bus of at least one bus of a second sub-computing device are connected by:

at least one unidirectional bus bridge; and

at least one memory store comprising at least two bus slave interfaces;

a bus of at least one bus of a first sub-computing device and a bus of at least one bus of a third sub-computing device are connected by:

at least one unidirectional bus bridge; and

at least one memory store comprising at least two bus slave interfaces; and

a bus of at least one bus of a second sub-computing device and a bus of at least one

bus of a third sub-computing device are connected by:

at least one unidirectional bus bridge; and

at least one memory store comprising at least two bus slave interfaces.

- 5 36. A computing device as claimed in any of claims 31 to 35 in which:
each sub-computing device is connected to a multi-port memory store;
each of the multi-port memory stores is connected to at least two of the sub-
computing devices; and
each sub-computing device is connected to every other sub-computing device by at
10 least one of the multi-port memory stores; and
none of the multi-port memory stores is connected to every sub-computing device.

37. A computing device as claimed in any one of claims 30 to 36 in which one sub-
computing device is directly connected to every other sub-computing device by a
15 corresponding bus bridge.

38. A computing device as claimed in any one of claims 30 to 37 in which each sub-
computing device is directly connected to every other sub-computing device by a
20 corresponding bus bridge.

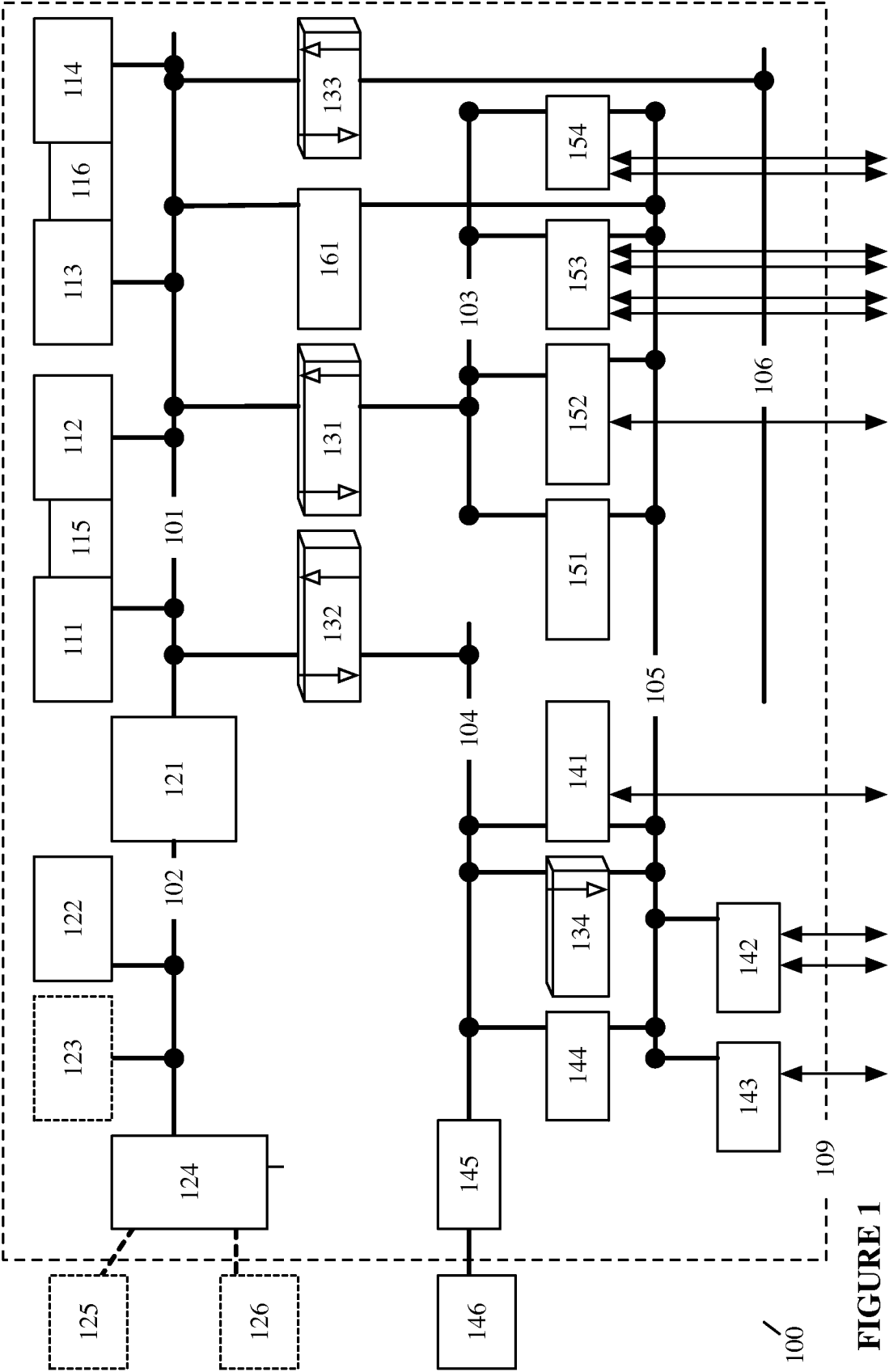
39. A computing device as claimed in any one of claims 30 to 38 in which one sub-
computing device further comprises an additional memory store, that sub-
computing device being directly connected to every other sub-computing device by
a corresponding bus bridge to permit memory transfer requests to be issued from
25 each sub-computing device to that additional memory store.

40. A method of selectively controlling a sub-computing device of a computing device
as claimed in any one of claims 30 to 39 in which the method is adapted to comply
with a policy selected from the group comprising:

- 30 permitting all bus masters connected to that bus of that sub-computing
device to issue memory transfer requests onto that bus;
permitting some of the bus masters connected to that bus of that sub-
computing device to issue memory transfer requests onto that bus; and

permitting at most one of the bus masters connected to that bus of that sub-computing device to issue memory transfer requests onto that bus.

41. A computing device as claimed in claim 30 and claim 31, in which no bus master
5 of any of the N sub-computing devices can issue memory transfer requests onto any of the at least one busses of any of the (N-1) other sub-computing devices.
42. A representation in a hardware description language of a computing device as
10 claimed in any one of claims 30 to 39 or 41.
42. A process emulating a computing device as claimed in any one of claims 30 to 39
or 41.
43. A signal carrying a representation in a hardware description language of a
15 computing device as claimed in any one of claims 30 to 39 or 41.
44. A machine readable substrate carrying a representation in a hardware description
language of a computing device as claimed in any one of claims 30 to 39 or 41.



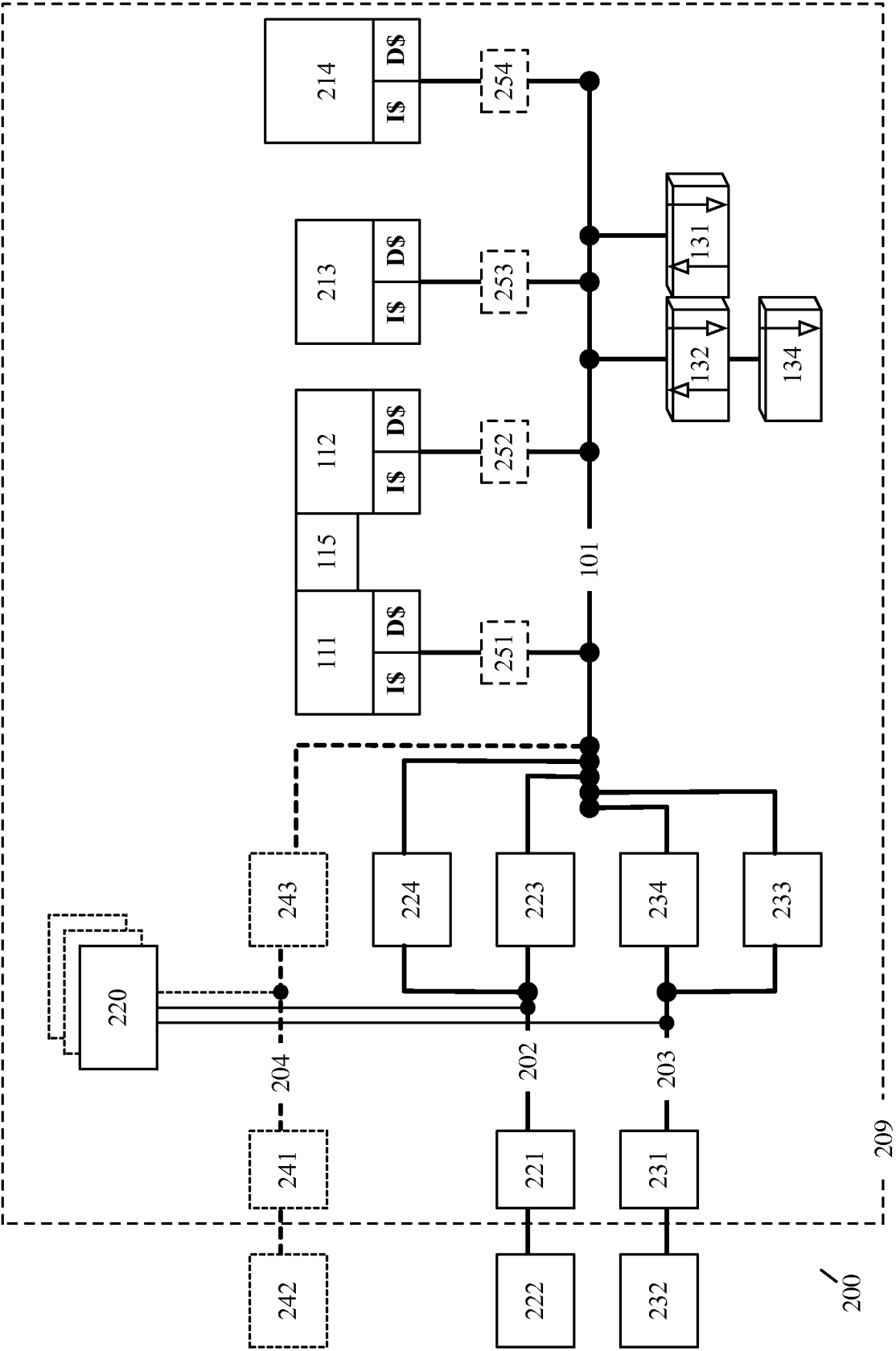


FIGURE 2

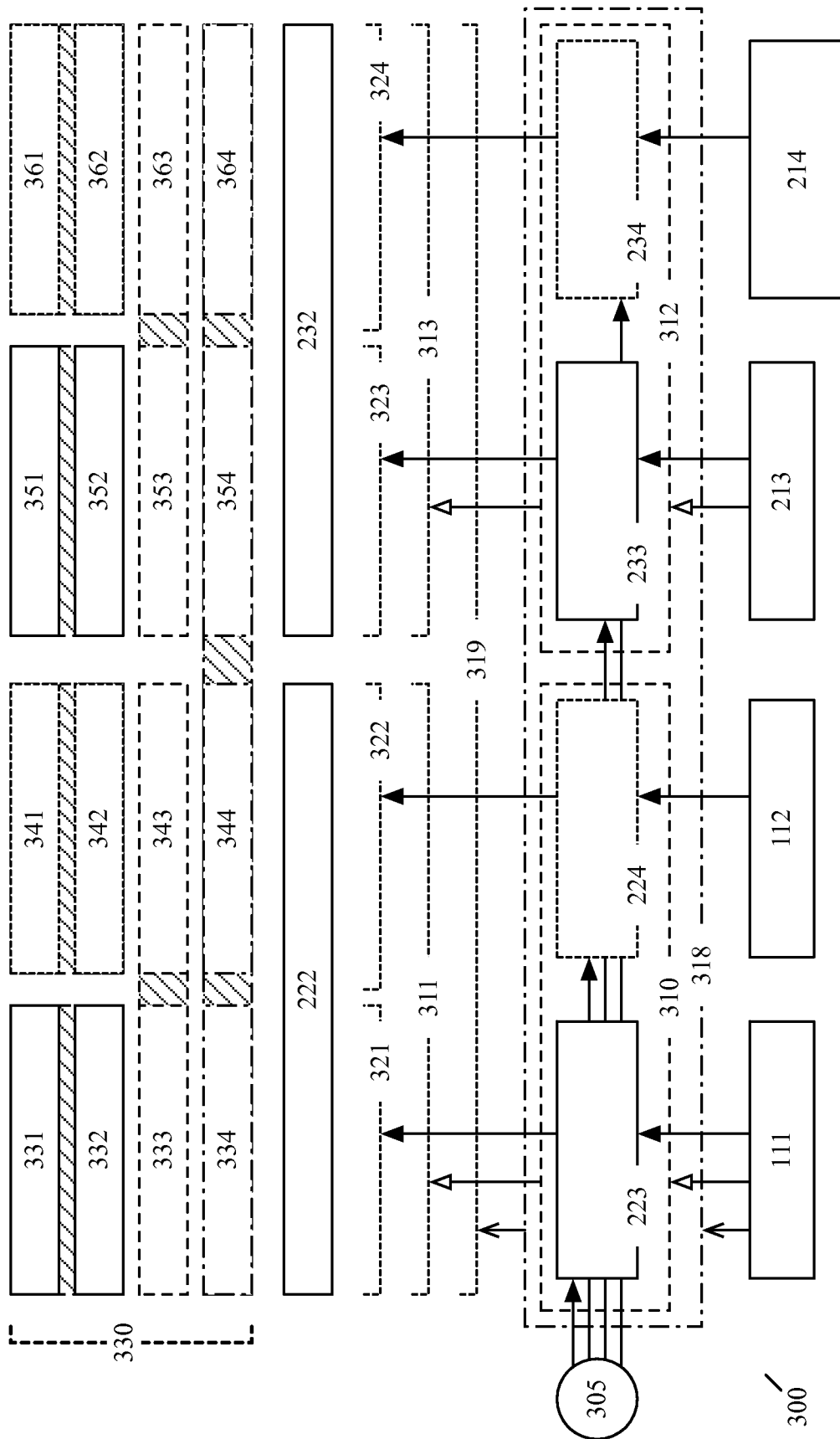


FIGURE 3

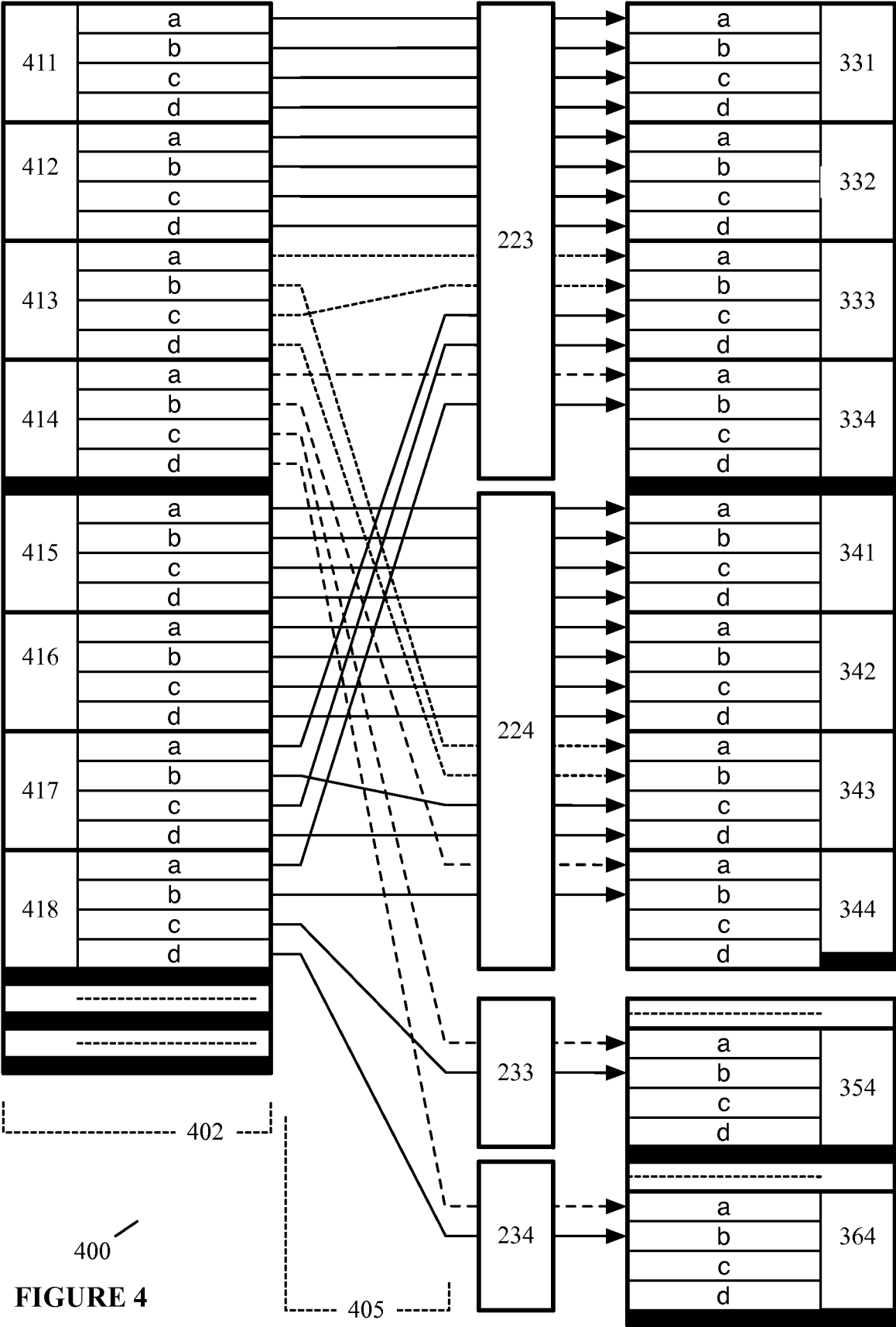
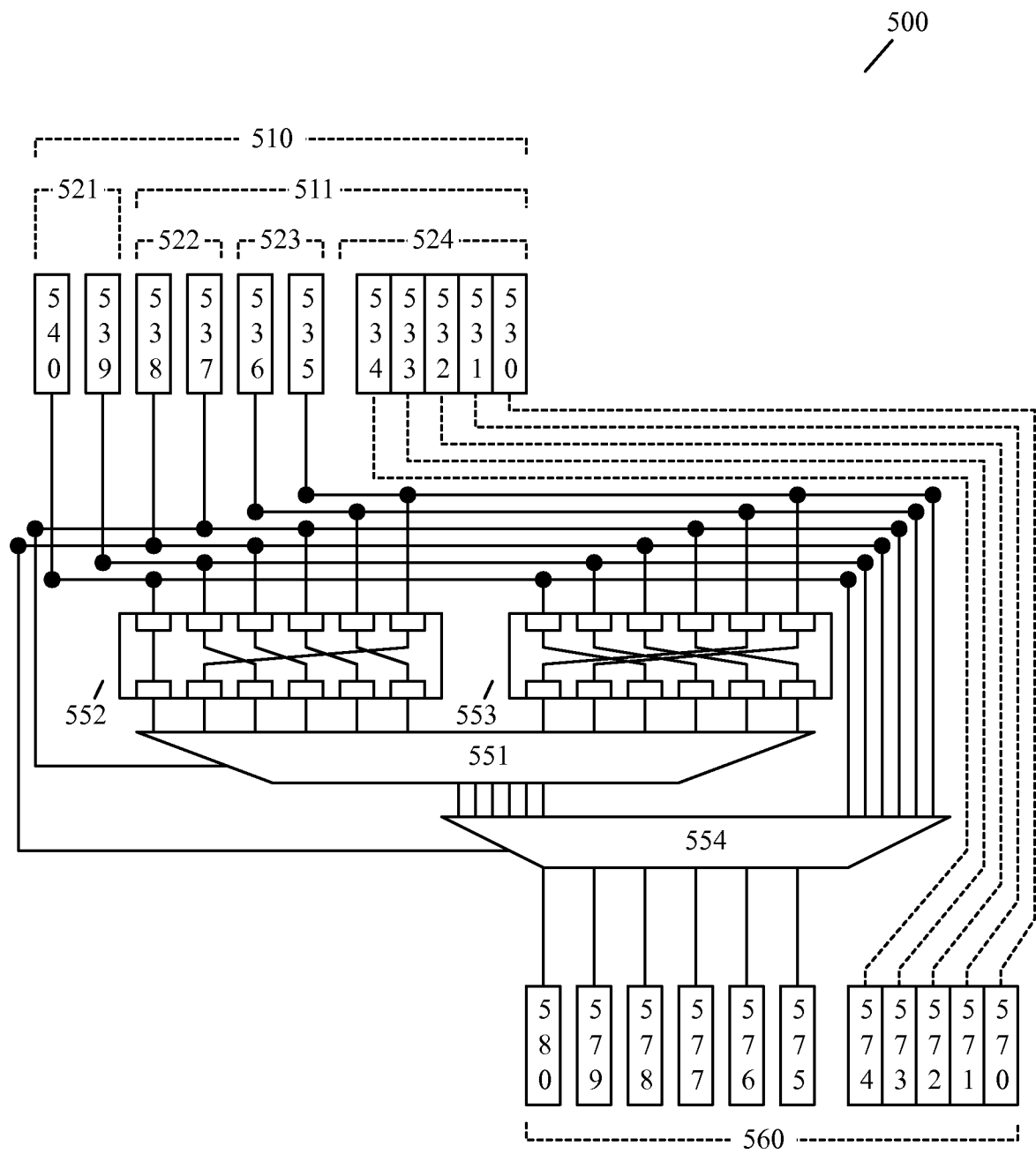


FIGURE 5



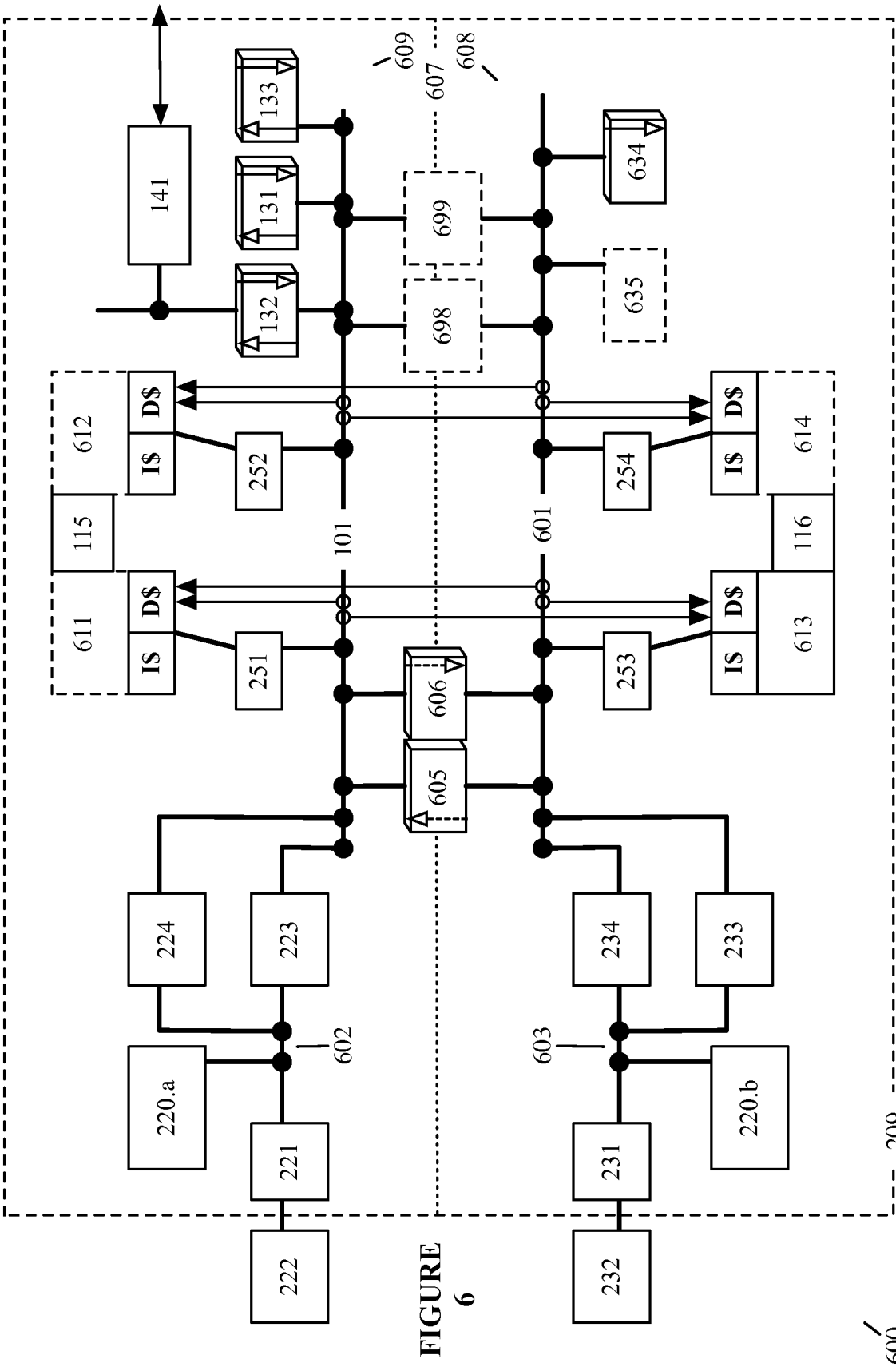
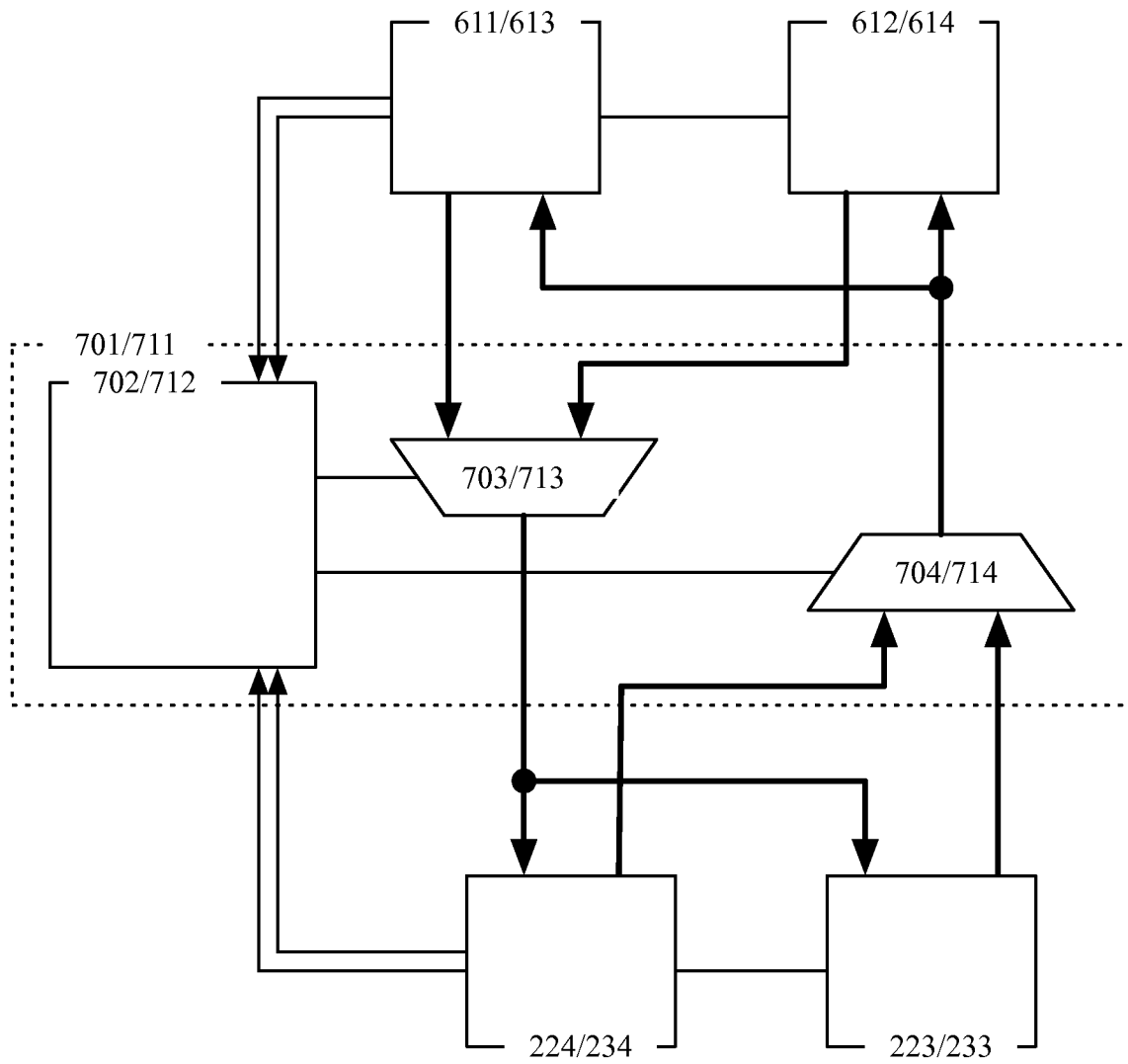
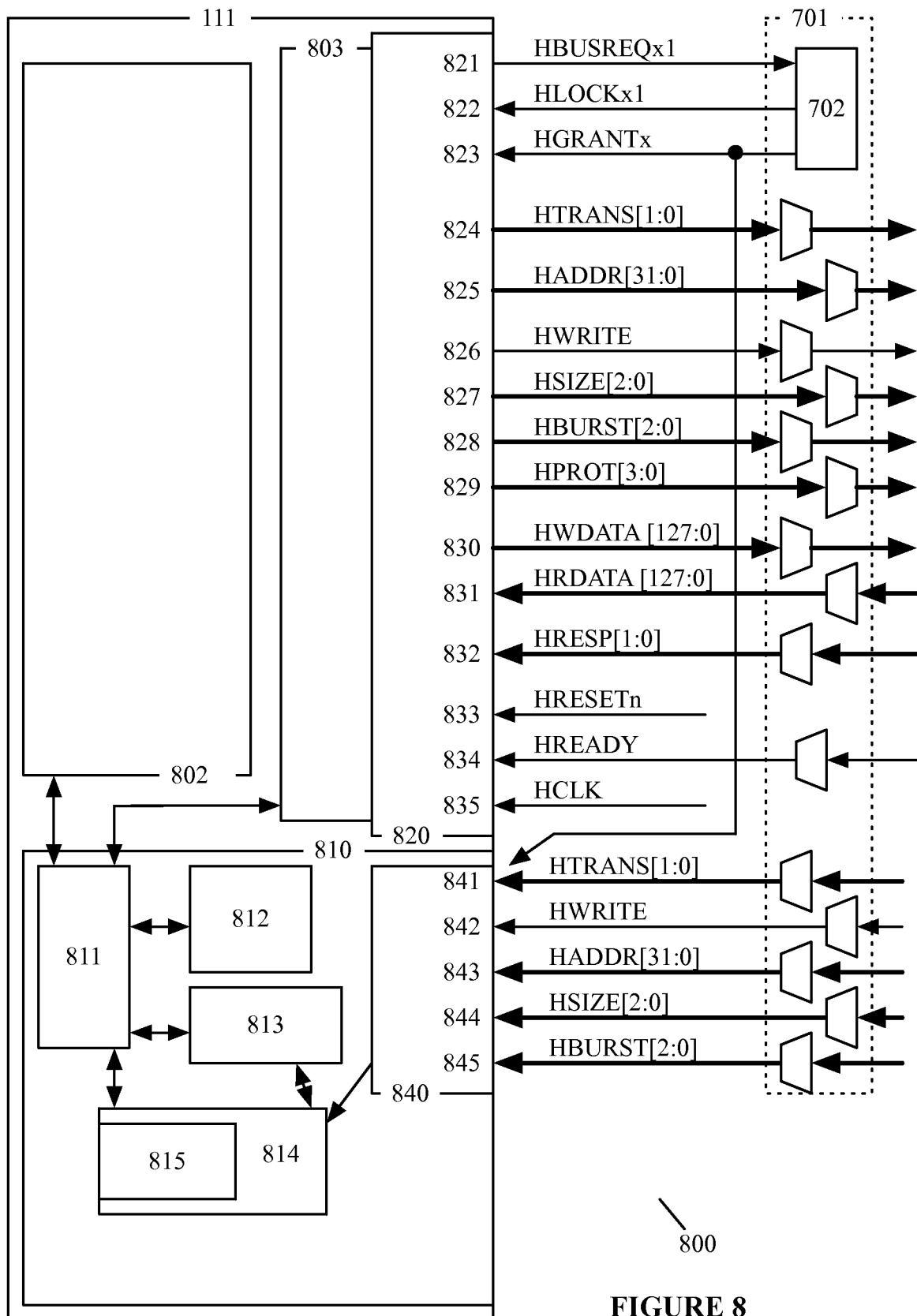


FIGURE 7





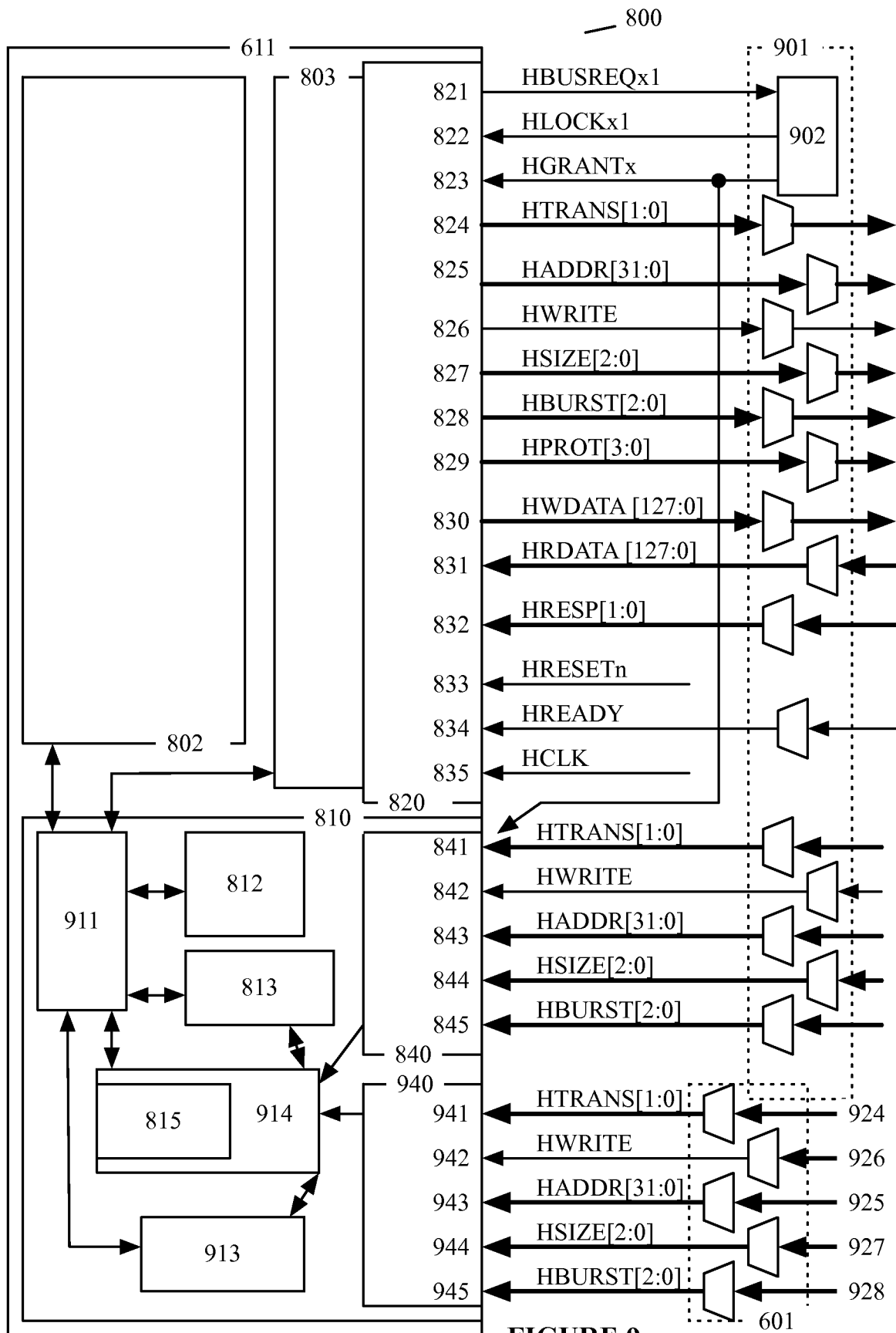
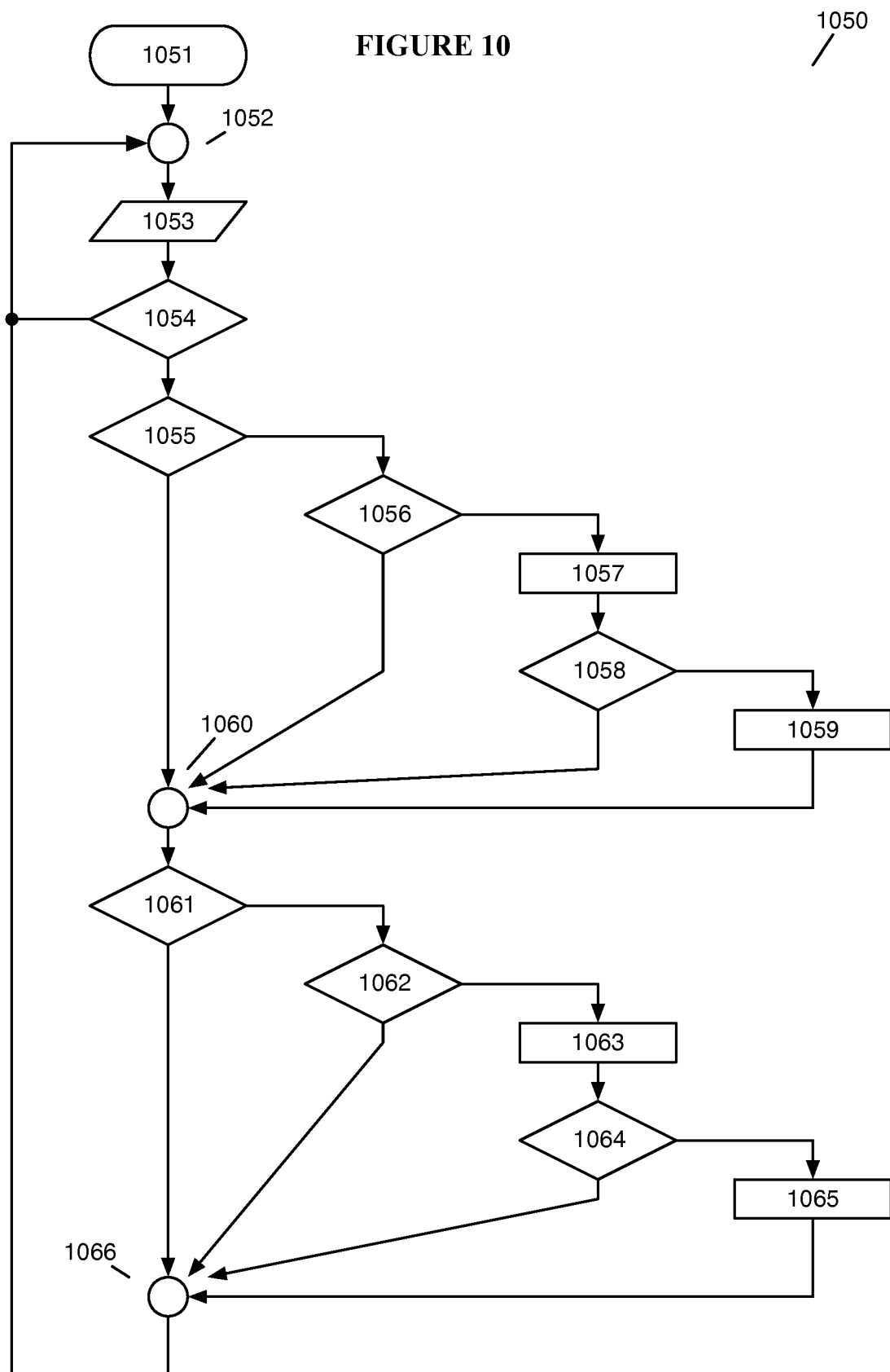
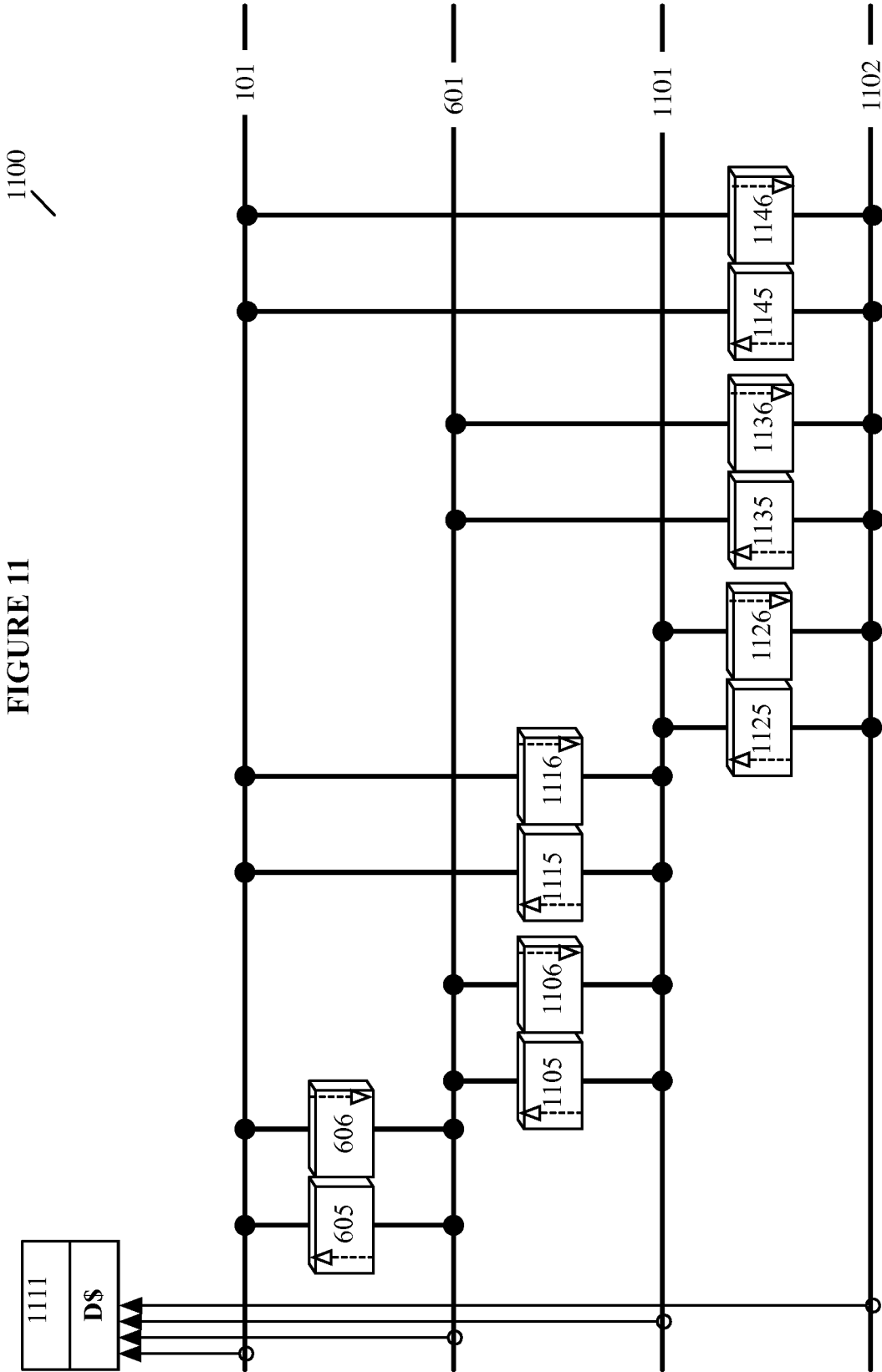


FIGURE 9

FIGURE 10





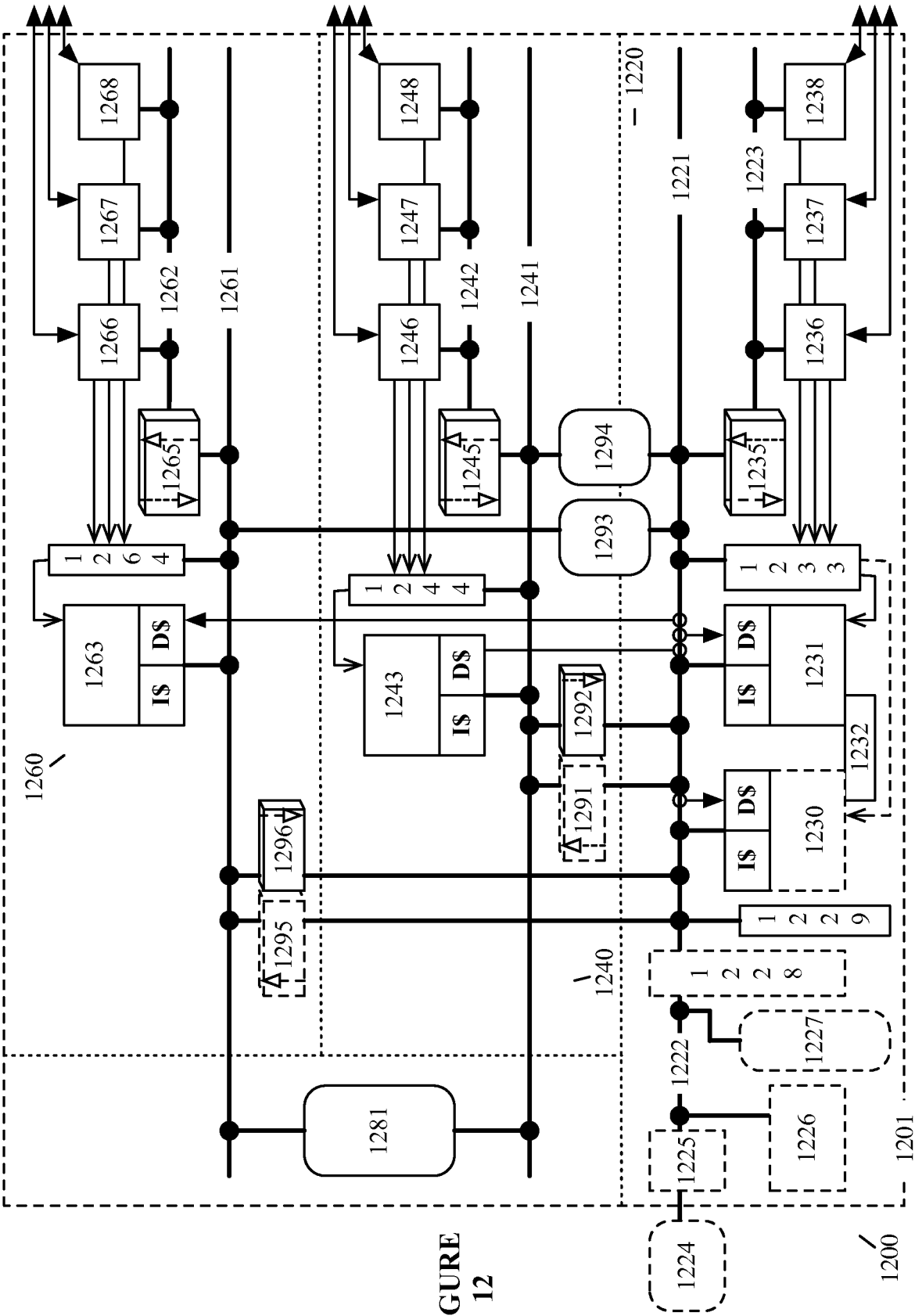


FIGURE 13

