A MULTIPROCESSOR INFRASTRUCTURE FOR PROVIDING FLEXIBLE BANDWIDTH ALLOCATION VIA MULTIPLE INSTANTIATIONS OF SEPARATE DATA BUSES, CONTROL BUSES AND SUPPORT MECHANISMS

CROSS REFERENCE TO RELATED APPLICATIONS

This application claims priority from U.S. Provisional Patent Application Ser. No. 60/315,144 (Attorney Docket No. 10559-579P01), *filed August 27, 2001.*

BACKGROUND

Conventional bus schemes define a bus as an indivisible unit. Although commands and data may be transmitted over separate physical channels to improve concurrency, the bus protocols link the channels. The commands include a fixed number of predefined fields of command information, such as address, length (number of data bytes) and operation type (e.g., read, write, cacheline flush, and so forth).

DESCRIPTION OF DRAWINGS

FIG. 1 is a block diagram of a processing system employing bus arbitration logic to support exchanges between bus masters and bus targets.

FIG. 2 is a detailed block diagram of the processing system (of FIG. 1) showing various buses, including command and push/pull buses, and associated arbiters of the bus

arbitration logic.

FIG. 3 is an illustration of an exemplary format of a command carried over the command bus.

FIG. 4 is a schematic diagram of an exemplary command bus data path of the processing system.

FIG. 5 is a schematic diagram of an exemplary push bus data path of the processing system.

FIG. 6 is a schematic diagram of an exemplary pull bus data path of the processing system.

DETAILED DESCRIPTION

Referring to FIG. 1, a system 10 having multiple first units 12, shown as masters 12a, 12b, 12c, ..., 12j, and multiple second units 14, shown as targets 14a, 14b, 14c, ..., 14k that use bus arbitration logic 16 to control exchanges of information over a bus system 17 is shown. Collectively, the bus system 17 and bus arbitration logic 16 form a bus structure 18. The components of the bus arbitration logic 16 include a command bus arbiter 19, a push bus arbiter 20 and a pull bus arbiter 22. These arbiters 19, 20, 22 are associated with the operation of the various buses in the bus system 17, as will be described with reference to FIGS. 2-6.

The masters 12 can include devices such as processors (e.g., general purpose microprocessors, embedded

microcontrollers) and external bus controllers. The targets 14 can include memory resources (e.g., interfaces to DRAM, SRAM), I/O resources (e.g., interfaces to media, media switch fabric), as well as other types of system resources. A master can also serve as a target. For example, a bus controller that allows the system to communicate with a host or other system via an external bus could perform the functions of both master and target.

For simplicity, only two of the masters, masters 0 and 1, and three of the targets, targets 0-2, are shown in FIGS. 2-6. One of the masters, master 0, is depicted as a device that can serve as both a master and a target.

Referring to FIG. 2, the system 10 includes three separate bus structures, a command bus 30 (indicated in dashed lines), a push bus 32 and a pull bus 34. The respective arbitration units 19, 20 and 24 are associated with and control the use of the command bus 30, the push bus 32 and the pull bus 34 respectively. The command bus 30 includes, on the master side of the arbitration logic, a first command bus 36, and on the target side of the arbiters, a second command bus 37. The master side command bus 36 includes a corresponding command bus for each master, for example, command buses 36a and 36b for master 12a and master 12b, respectively.

The push bus 32 includes a first push bus (target side) 38 and a second push bus (master side) 40. The pull bus includes a first pull bus (target side) 42 and a second pull bus 44 (master side). On the target side, the target push bus 38 and target pull bus 42 are used by the targets to transport commands and data between the targets and the push bus arbiter 20 and pull bus arbiter 22, respectively.

The masters 12a and 12b use the command bus arbiter 19 to arbitrate for use of the command bus 30 and, in particular, the target side command bus 37. The targets 14a, 14b, 14c each use the push bus arbiter 20 and the pull bus arbiter 22 to arbitrate for use of the master side push bus 40 and the pull bus 34, respectively. Control signals related to bus arbitration (not shown) will be discussed later with reference to FIGS. 4-6.

Referring to FIG. 3, an exemplary format of a command 50 carried over the command bus 30 is shown. The command 50 is partitioned into multiple fields 52a, 52b, 52c, ... 52p that carry encoded information. The fields 52 can be of differing widths. The fields 52 include a target ID field 52a and an operation (or command) type field 52b. Each target 14 has a unique, hardwired identification number ("ID"). The target ID field 52a indicates, by ID, which of the targets should accept the command. The masters 12 send

a command in the format of the command 50 to all of the targets 14, which decode the target ID in parallel to determine if the command 50 is meant for them. A set of codes corresponding to target IDs for an exemplary set of targets is shown in TABLE 1:

Code	Target
00 0000	None (no command on the command bus)
00 0001	Media Interface
00 0010	SRAM Interface
00 0011	DRAM Interface
00 1001	Bus Controller
00 0100	Hash Unit

TABLE 1

The operation type field 52b specifies an operation type. Each target interprets the operation type specified by the code in the operation type field 52b to determine the operation to be performed. The value in the operation type field 52b determines how values in other fields are to be interpreted by each target. The operation type can include as many different encodings of information as operation types that most (if not all) target units support, for example, read and write. One or more of the fields 52c - 52p can be used as operation type extensions, based on the value in operation type field, as well. An exemplary

encoding for some commands types supported by the targets shown in TABLE 1 above are shown in TABLE 2:

Code	DRAM I/F	SRAM I/F	Hash Unit	Media I/F	Bus Controller
0000	Read	Read	Hash	Read	Read
0001	Write	Write	Reserved	Write	Write
0010	Receive_Buffer Read	Swap	Reserved	Reserved	Reserved
0011	Transmit_Buffer Write	Set bits	Reserved	Reserved	Reserved
0100	Control Status Registers (CSR) Read	Clear bits	Reserved	Reserved	Reserved
0101	CSR Write	Add	Reserved	Reserved	Reserved

TABLE 2

The remaining fields 52c through 52p (corresponding to fields Field_1, Field_2, …, Field_N) hold other parameters used by the targets 14, such as address and burst count. For example, and as shown in the command format of FIG. 3, Field 3 is used to specify a push/pull ID 53, which identifies a source or destination 54, among other types of information. The push/pull ID 53 is passed to the target 14 via the command bus 30, and provided by the target to the push bus arbiter or the pull bus arbiter when the command is serviced. The push bus arbiter or the pull bus arbiter arbiter, as appropriate in turn, provides the push/pull ID to the appropriate source (in the case of a pull operation) or destination (in the case of a push operation) as an

address. The definition of fields within the push/pull ID field may be dependent on the specified source or destination. In addition, one or more of the Fields 1 through N may be used differently based on operation type. For example, a field can indicate byte masks for a write operation type and priority level for a read operation type.

According to the bus protocol of system 10, each target that receives a command in the form of the command 50 interprets the fields 52 on a per-target basis. This enables the targets to supply a rich set of functions, such as read-modify-write with programmable modify, data transform (e.g., perform a hash on operands and return result) and fast write (use some fields of the command to specify data and others to specify address, eliminating the latency and resource use of data bus).

The masters 12 issue commands to the targets 14 based on instruction execution, or some other appropriate mechanism. For example, if the master is a bus controller, it may issue commands as a result of a DMA operation or when it is being accessed by another device on the bus that it controls.

Referring to FIG. 4, a command bus data path 60 is shown. The command arbiter 19 includes an arbitration unit 62 that receives a command queue full signal 64 as input

from each of the targets 14. The command queue full signal 64 indicates when a threshold fullness of a command queue 66 in each target has been reached. The arbitration unit 62 receives command requests over command request input lines 68 from the masters 12, and provides on grant output lines 70 to the masters 12 grants to the command bus requests by the masters 12.

Each bus master 12 that wishes to drive a command on the command bus 30 first sends a request to the arbitration unit 60 on a respective one of the input lines 68. This request indicates the ID of the target with which the master wishes to communicate. As mentioned earlier, the arbitration unit 62 also receives signals from each target 14 indicating whether that target's command queue 66 is full or not. The arbitration unit 62 uses the command queue full information to inhibit grants to the command requesters if the command queue of the target they need is full, while permitting requesters to other targets to obtain grants. The arbitration policy of the arbitration unit 62 may be, for example, a round robin policy, and may employ a programmable priority for certain masters. The arbitration scheme of the arbitration unit 62 opportunistically grants access to another bus master when one of two conditions occur: a master is scheduled to be granted access but does

not have a request pending; or a master is scheduled to be granted access but the requested target's command queue is full.

When the arbitration unit 62 has granted a bus request of a master, that master transmits a command on a corresponding one of the master command buses 36, to the command bus arbiter 19. The command bus arbiter 19 also includes a multiplexor (MUX) 72 that receives a command from a master and is enabled by the arbitration unit 62 to control the transmission of the command over the command bus 37 to the targets.

FIG. 5 shows details of a push bus datapath 80. The push bus arbiter 20 includes multiple push command queues 82a, 82b, 82c, 82d, one for each supported target, coupled to a multiplexor (MUX) 84 and a push bus arbitration unit 86. The push command queues 82a, 82b, 82c, 82d are coupled to corresponding targets 14a, 14b, 14c and target "master 12a" (for operations in which master 0 functions as a target), respectively, by sets of push data buses 88 and push command buses 90, one set for each target in the system, that is, buses 88a and 90a for target 14a, buses 88b and 90b for target 14, buses 88c and 90c for target 14c and buses 88d and 90d for master 12a (as a target). The target side bus 38 also includes a push command queue full signal

(Push_CmdQ_Full) 91 to indicate to a target that its corresponding push command queue has reached a predetermined fullness level. The master side push bus 40 (from FIG. 2) includes a push ID bus 92 and a push data bus 94.

A target, having received and decoded a command that was intended for it, sends requested data to a corresponding one of the queues 82 on the corresponding push data bus 88. Thus, the push command queues store a combination of push data and push commands. The push commands include push IDs, which are provided to the arbitration units 86.

Data stored in entries of the push command queues 82 are provided to the MUX 84. The arbitration unit 86 selects one of the push command queues 82. The arbitration unit 86 thus provides a select signal 96 to the MUX 84 to enable MUX 84 to provide as output (for transmission onto the push data bus 94) data from the selected push command queue and transmits the push ID from the selected push command queue onto the push ID bus 92.

Referring to FIG. 6, a pull bus data path 100 is shown. On the target side of the pull bus arbiter 22, the targets 14 are coupled to the pull bus arbiter 22 by the pull bus 42 (from FIG. 2). The masters 12 are coupled to the pull bus arbiter 22 by the master side pull bus 44 (from FIG. 2). The targets 14 are end points and the masters 12 are sources

for a pull operation.

The pull bus arbiter 22 includes a multiplexor (MUX) 102, an arbitration unit 104 and pull command queues 106a, 106b, 106c, 106d, one for each supported target, coupled to the pull bus arbitration unit 104. The pull command queues 106a, 106b, 106c, 106d are coupled to corresponding targets 14a, 14b, 14c and target "master 12a" (for operations in which master 0 functions as a target), respectively, by pull data buses 108a, 108b, 108c and 108d, respectively, and pull command buses 110a, 110b, 110c and 110d, respectively. Collectively, the buses 108, 110 make up the target side pull bus 42. The arbiter 22 also provides a pull command queue full signal (Pull_CmdQ_Full) 111 to a target to indicate to the target that the corresponding pull command queue 106 is almost full, and a signal Take_Data 112 to indicate to a target that pull data has been transferred to that target.

The master side pull bus 44 includes a pull data bus 112a for master 12a and a pull data bus 113b for master 12b. The pull data bus 113 is used by the masters to send pull data to the arbiter 22, or more specifically, to the multiplexor 102. The pull bus 44 also includes a pull ID bus 114 and pull done signal (Pull_Sig_Done) 116 that allow the arbiter 22 to provide information to the masters during

a pull (or write) operation.

Each of the targets uses a data buffer (not shown) to store pull (or write) data. The target receives the command 50 over the command bus 37 (shown in FIGS. 2 and 4) and determines from the command's operation type field 52b (FIG. 3) that the operation is a pull operation. When the target has room in the data buffer for the amount of data specified in the command, it arbitrates for the use of the pull data bus 34. The information specifying the location of the data (that is, which master and which storage location being used by the master for the data) was presented in the pull ID field 53 of the command 50. Because that information is in a command field, the unit identified as the source of the pull data does not need to be the master that sent the command. The source can be any addressable unit that has a path to the pull data bus 108.

Each target sends the full Pull ID and length (derived from the command 50) for information it would like to pull to the target. The target must have buffer space available for the pull data when it asserts the Pull ID via the corresponding pull command bus 110.

The Pull ID is enqueued in a corresponding one of the pull command queues 106 in the pull bus arbiter 22 unless the Pull_CmdQ_Full signal 111 is asserted for that pull

command queue. The assertion of the Pull_CmdQ_Full signal 111 indicates that the pull command queue 106 for that specific target has reached a predetermined fullness threshold.

The arbitration unit 104 arbitrates among the currently valid pull IDs enqueued in the pull command queues 106 to select a target, or more specifically, a pull ID enqueued by that target. The arbitration policy can be one of a number of well known schemes, for example, round robin, or a priority based scheme. The arbitration unit 104 sends the selected pull ID to the corresponding source over the pull ID bus 114. The pull bus arbiter 22 asserts the Take_Data signal 112 to the selected target. The source provides the pull data to the MUX 102, which is enabled to send the pull data onto the pull data bus 108 by the arbitration unit 104 via control signal 119. The arbitration unit 104 asserts the pull done signal 116 to the source.

Thus, write data transport is under the control of the target. The target of a write operation pulls the write or pull data over the pull bus 34 when it needs it, rather than having it sent at the same time as the write operation type. Therefore the target can allocate internal buffers for the pull data based on when the target needs the data and has available buffer space. Also, as mentioned earlier, the

target can get write data from a source other than the command initiator, as directed by information in the command.

The architecture of system 10 provides for flexible bandwidth allocation via multiple instantiations of the various buses and arbiters. Because there are separate buses for commands, push data, and pull data, the buses can be added incrementally as needed.

Referring back to FIG. 4, if the control bandwidth for an application is insufficient, one or more additional copies of the command bus 37 and arbiter 19 could be added to the system. Each arbiter 19 could support a subset of the masters (for example, in a four-master implementation in which two arbiters are used, each arbiter could support a different pair of the masters), but the targets would receive commands from all of the masters.

For increased data bus bandwidth, copies of the push bus and pull bus, along with the appropriate arbitration logic, could be added. For example, referring back to FIG. 5, additional master-side buses 40 and arbiters 20 could be added to the system. Each target's push data bus 88 and command bus 90 would be coupled to each of the arbiters so there would be no need for a target to drive more than one set of push data/command buses. Each arbiter would have to

be aware of the masters (destinations) to which it is wired, and enqueue the data and IDs accordingly. Each arbiter 20 and associated bus 40 could be connected to a subset of the sources.

Referring to FIG. 6, copies of the target side pull bus 42 and associated arbitration logic could be added to the system. Each arbiter 22 and bus 42 would be connected to each of the targets. Each arbiter 22 and associated bus 44 could support a subset of the sources. The pull bus arrangement would be similar to the push bus arrangement in that each target only needs to drive one copy of the pull ID to all of the arbiters. However, unique copies of the pull data are needed, as it is possible that the pull buses would have valid data on them on the same cycle. In the case of both push and pull buses, all of the arbiters would be connected to all targets, allowing data to be moved between any master and any target.

Other embodiments are within the scope of the following claims.

What is claimed is:

1. A method comprising:

transferring a command and a target identification between one or more bus masters and bus targets over a bus structure, the command including information that is interpreted differently by one of the bus targets based on the target identification.

2. The method of claim 1 wherein the target identification is a field in the command.

3. The method of claim 1 wherein the bus structure includes a separate command bus, push bus and pull bus.

4. The method of claim 1 wherein transferring includes sending the command to all of the bus targets over the command bus so that each bus target can determine if the command is intended for such target.

5. The method of claim 3, wherein the bus target identified in the command controls the transfer of information over the push bus to one of the bus masters for

a push operation.

6. The method of claim 5 wherein the one of the bus masters is identified in the command by an identifier in a destination field in the command.

7. The method of claim 3 wherein the bus target identified in the command is operable to control transfer of information over the pull bus from one of the master buses to the bus target for a pull operation.

8. The method of claim 7 wherein the one of the bus masters is identified in the command by an identifier in a source field in the command.

9. The method of claim 1 wherein the command is formatted to specify one of a plurality of operation types, and the command includes at least one field that is interpreted according to the which of the plurality of operation types is specified in the command.

10. The method of claim 2 wherein at least one of the units acts as one of the bus masters at times and acts as one of the bus targets at other times.

11. The method of claim 3 wherein the bus structure further includes arbiters, the arbiters including a command arbiter associated with the command bus, a push arbiter associated with the push bus and a pull arbiter associated with the pull bus

12. The method of claim 11 wherein the bus masters arbitrate for use of the command using the command arbiter, and the bus targets arbitrate for use of the push and pull buses using the respective push and pull arbiters.

13. An article comprising:

a storage medium having stored thereon instructions that when executed by a machine result in the following:

transferring a command over a bus to bus targets, the command being formatted to identify one of the bus targets and include information that is interpreted differently based on which one of the bus targets is identified.

14. The article of claim 13 wherein the bus structure includes a separate command bus, push bus and pull bus.

15. An apparatus comprising:

a plurality of units;

a bus structure to enable communication exchanges between the units connected to the bus structure, with one or more of the units being bus masters and others of the units being bus targets, the bus masters operable to send a command to bus targets over the bus structure, the command formatted to identify one of the bus targets and having information that is interpreted differently based on which one of the bus targets is identified.

16. The apparatus of claim 15 wherein the bus structure includes a separate command bus, push bus and pull bus.

17. The apparatus of claim 16, wherein the command is sent to all of the bus targets over the command bus to all of the bus targets and each bus target determines if the command is intended for such target.

18. The apparatus of claim 16, wherein the bus target identified in the command is operable to control transfer of information over the push bus to one of the bus masters for a push operation.

19. The apparatus of claim 18, wherein the one of the bus masters is identified in the command by an identifier in a destination field in the command.

20. The apparatus of claim 16, wherein the bus target identified in the command is operable to control transfer of information over the pull bus from one of the bus masters to the bus target for a pull operation.

21. The apparatus of claim 20, wherein the one of the bus masters is identified in the command by an identifier in a source field in the command.

22. The apparatus of claim 15 wherein the command is formatted to specify one of a plurality of operation types, and the command includes at least one field that is interpreted according to the which of the plurality of operation types is specified in the command.

23. A apparatus comprising:

a bus master operable to send a command to bus targets over a bus structure, the command being formatted to identify one of the bus targets and including information

that is interpreted differently based on which one of the bus targets is identified.

24. The apparatus of claim 23 wherein the command is formatted to specify one of a plurality of operation types, and the command includes at least one field that is interpreted according to the which of the plurality of operation types is specified in the command.

25. An apparatus comprising:

a bus target operable to receive a command from a bus master over a bus structure, the command being formatted to identify the bus target that receives the command; and

logic in the bus target to interpret information that is received by the bus target identified.

26. The apparatus of claim 25 wherein logic processes the command that is formatted to specify one of a plurality of operation types, and the command includes at least one field interpreted according to the which of the plurality of operation types is specified in the command.