Application Report
SPRAAX6–August 2008

TMS320C6474 Common Bus Architecture Throughput

Jon Bradley

ABSTRACT

This application report presents common bus architecture protocols and components as main factors for generic throughput analysis. It provides necessary details on the internal bus structure which enables you to estimate system-on-chip (SoC) performance for a given application.

This document is intended to be used in conjunction with a specific device performance application report, such as the TMS320C6474 Module Throughput (SPRAAW5).

Contents

1 CBA Overview

Texas Instruments developed the common bus architecture (CBA) as a convenient and scalable method of connecting the blocks in a system-on-chip (SoC) without changing the modules themselves.

CBA provides up to 16,000 M-bytes/s (MBps) per interface @ 500 MHz (using a 256-bit data bus) and supports data paths with 8-, 16-, 32-, 64-,128- or 256-bit data, depending on a trade-off between power, area, and performance. The CBA also:

- Allows both pipelined burst read and write operations and non-pipelined transfers
- Provides continuous back-to-back transactions, multiple masters, and background master arbitration
- Supports slave-inserted (but not master-inserted) wait states

2 Protocols

CBA comprises two different protocols that are referred to as VBUSP and VBUSM. The unit of data and the core control handshaking used to transfer this data is shared by both protocols. This commonalty allows the optimal protocol to be used at various places in the system while maintaining a high degree of compatibility and makes bridging between the protocols a straightforward operation.

2.1 Master vs. Slave

In CBA, all data transfers occur between exactly two parties on the interface referred to as the master and the slave.

The master drives all transaction level control information. It also drives the write data and accepts the read data, read status, and write status information.

The slave responds to transactions. For all transactions, this involves accepting and acting on the transaction information. For write transactions, this involves accepting the write data and metering the transfer. For read transactions, this involves driving the read data and metering the transfer.

2.2 VBUSP Protocol

VBUSP is a very simple and easy to implement protocol that is pended such that only a single transaction can be outstanding at any given time. VBUSP protocol is classified as a point-to-point, pended interface protocol.

Transactions are passed between a master and a slave. They are made up of a collection of data phases that are related to one another by address and order and are transferred as a unit across the interface. At the lowest level, all transactions on VBUSP are either reads or writes as specified by the direction signal.

A write status bus is an optional interface for VBUSP peripherals. The intent of the bus is to signal back to the requesting master when write data has truly landed, as opposed to when write data has left the master’s boundary.

The master is responsible for specifying all of the parameters surrounding the transaction including the direction, starting address, address progression, and transfer length. The slave is responsible for sourcing the read data and for responding to the transaction by asserting the appropriate ready signal for reads or for writes.

Each data phase of a VBUSP transaction regardless of if its direction, is required to complete on the interface before the next data phase can be presented. This also necessitates that transactions are completed on VBUSP entirely before the next transaction is presented on the interface.

The relatively good performance and simplicity of the VBUSP protocol makes it highly suitable for use in all but the most demanding areas in a SoC.

2.3 VBUSM Protocol

VBUSM allows for extensive amounts of transaction pipelining by allowing multiple transactions to be outstanding; this can dramatically increase system performance at a cost of higher complexity and more logic.

The VBUSM protocol represents a significant leap in scalability of performance over VBUSP while maintaining transaction level equivalence. VBUSM provides increased performance capabilities by eliminating the blocking behavior between transactions that is inherent in VBUSP. It allows transaction control signals to be pipelined independently from the data.

A write status bus is a required interface for VBUSM peripherals. The intent of the bus is to signal back to the requesting master when write data has truly landed, as opposed to when write data has left the master’s boundary (data could be buffered in intermediate bridges).

The master is responsible for driving all of the command interface signals except the ready signal. The slave is given the capability to meter completion of data phases for both reads and writes using a combination of ready signals. It also has direct control over when the read and write status transfers are initiated.

It is restricted for use as a high speed system bus between CPUs and memory interfaces primarily based on the performance features and cost associated with the VBUSM.

3 Switch Central Resources (SCR)

The SCR is an N:M crossbar that allows N masters to connect to M slaves. It adds no latency and allows seamless arbitration between the masters and slaves (i.e., no dead cycles inserted by the fabric). The SCR is auto-generated and can restrict access to each slave from any number of the masters.

SCR only supports single protocol (VBUSP or VBUSM), frequency, and bus width. Example SCRs are shown in Table 1. In this example, SCR A crossbar is connecting nine input ports to six output ports, has VBUSM protocol, runs at cpu_clk clock domain, and supports 128-bits bus width. Whereas, SCR B is a VBUSP crossbar connecting four input ports to seven output ports, runs at 1/3 the cpu_clk clock domain, and supports 32-bits bus width.

Table 1. Example SCRs

SCR	Type	Clock Domain	Width	Ports (N:M)
A	VBUSM	cpu_clk	128 bits	9:6
B	VBUSP	cpu_clk/3	32 bits	4:7

Any conversion required for one of the three SCR parameters (protocol type, frequency, or bus width) is handled in bridges.

VBUSM and VBUSP SCRs allow for concurrent data traffic between any of the end-points.

As mentioned previously, the VBUSM protocol allows for pipelining of commands to the various end-points. As such, each of the masters can have multiple commands outstanding; however, VBUSP does not allow for pipelining of commands to the various end-points. Each of the masters can have one command outstanding, at most.

Regardless of the SCR protocol, each master-slave connection is completely independent. So, various masters can have commands outstanding and data in-flight at the same time. When communicating with the same slave end-point, the slave is responsible for arbitrating between the requestors based on priority.

Both VBUSP and VBUSM SCRs transport transactions between a scalable number of master and slave interfaces. They have a selectable number of pipeline stages from 0 to 3 allowing zero clock cycle latency. When pipelining is enabled, pipeline stages are inserted to break timing paths between the source and destination. These stages can either be before the master decoder, between the decoders and the arbiters, or after the slave arbiter. While the pipeline stages add registers to ease timing, they only insert a single cycle of latency on writes and two cycles of latency on reads (one forward and one reverse) and continue to achieve unity bandwidth.

4 Bridges

Bridges are major components in CBA based systems. Like SCRs, bridges are considered part of the infrastructure and are not themselves considered initiators or targets. Bridges are not required in all systems but are typically used to:

- Connect bus segments running at different frequencies
- Connect bus segments of different data path widths
- Connect bus segments with different protocols (VBUSP vs. VBUSM)

Bridges are also designed in a way to increase overall throughput. They can be used in a variety of places to connect initiators, targets, central resources, or to connect one bridge to another. Table 2 shows the example bridges.

Table 2. Example Bridges

Bridge	Type (Mstr:Slv)	Width (Mstr:Slv)	Clk (Mstr:Slv)	Read FIFO Ctl	Read FIFO Data	Read FIFO Burst Size	Write FIFO Data	Write FIFO Burst Size	Cmd FIFO Depth	Status FIFO Depth
1	P-M	128:128	1:1	N/A	N/A	N/A	N/A	N/A	N/A	N/A
2	M-M	64:128	1:1	8	8	1	3	2	3	2
3	M-M	32:64	1:2	3	7	1	6	4	2	2
4	M-P	64:32	2:1	2	2	1	3	0	2	2

Table 2 lists some bridge examples. The main parameters for each bridge are listed below:

- Bridge type: This is the protocol conversion provided by the bridge. In Table 2, an “M” indicates VBUSM and a “P” indicates VBUSP. The first letter marks the protocol of the bridge master and the second letter marks the protocol of the bridge slave. For example, bridge #4 connects a master VBUSM bus to a slave VBUSP bus.
- Width: This is the bus width conversion in bits. The first number is the bus width of the bridge master and the second number is the bus width of the bridge slave. For example, bridge #3 has a 32-bit wide master bus and a 64-bit wide slave bus.
- Clock: This is the relative clock ratio of the master clock to the slave clock. For example, bridge #4 could be a bridge between a module at one-third the CPU rate and a module at one-sixth the CPU rate, so the clock bridge parameter is 2:1. Any clock scaling must be global, unless an asynchronous bridge is used between components.
- Read first-in-first-out (FIFO) control: This parameter indicates the number of outstanding read commands that can be present in the bridge. In VBUSP, only a single transaction can be outstanding at any given time; the read FIFO control parameter is only used in bridges where the master has VBUSM protocol.
- Read FIFO data: This parameter indicates the FIFO depth used for read response data. FIFO entries are indicated in terms of the widest port of the bridge (i.e., read FIFO data = 8 for bridge #2 means the FIFO depth is 8 * 128-bit). This parameter is only used in bridges where the master has VBUSM protocol.
- Read FIFO burst size: This parameter controls the bursting attributes of the read response back to the master. The burst size, as with the FIFO size, is in terms of data phases with respect to the wider of the two bridge data buses. The read FIFO burst size parameter is only used in bridges where the master has VBUSM protocol. Data is buffered in the bridge until the read burst size is reached. Time-out mechanism exists to avoid indefinite waiting for small data amounts. A value of 0 for read FIFO burst size means there is no wait.
- Write FIFO data: This parameter is comparable to that for reads. It is the FIFO depth for write data. The write FIFO data parameter is only used in bridges where the master has VBUSM protocol.
- Write FIFO burst size: This parameter is comparable to that for read FIFO burst size. It indicates the write bursting attributes for the bridge. It is only used in bridges where the master has VBUSM protocol. Data is buffered in the bridge until the write burst size is reached. Time-out mechanism exists to avoid indefinite waiting for small data amounts. A value of 0 for write FIFO burst size (as in bridge #4) means there is no wait.

- Command FIFO depth: This parameter indicates the total number of commands that can be accepted by the bridge at any point in time. It is used only in bridges where the master has VBUSM protocol. This parameter is critical in determining the rate of command service of a bridge.
- Status FIFO depth: This indicates the depth of the FIFO used for write status back to the master. It is used only in bridges where the master has VBUSM protocol.

5 Latency Contributors

5.1 Priorities

The priority level of all master peripheral traffic is defined at the SCR boundary. User-programmable priority registers allow the software configuration of the data traffic through the SCR.

The priority of a transfer request across SCRs and bridges is influenced by the nature of the components in the path of a transfer. High priority transfer requests from a master that are expected to be serviced immediately by a slave could be delayed due to bandwidth sharing of the components and the potential blocking of the queue head of line. The high priority command may need to wait for the processing of the current low priority requests before it gets serviced.

Because bridges do not reorder commands, a high-priority request at the master interface of a bridge that happens to be at the tail of a queue is blocked by lower priority commands at the head of the queue. So, at the slave interface of the bridge, the servicing of this high priority request must wait until the lower priority requests get serviced.

Since the SCR allows for concurrent transfers between non-conflicting master/slave pairs, the SCR can support a very high total data rate across any endpoints. If transfers line up such that the source or destination memory is the same, then collisions occur and certain transactions are blocked.

Bridge parameters also affect latency; this is mainly due to the read FIFO burst size and the write FIFO burst size parameters. The assumption is: there is no latency at the peripherals themselves.

5.2 Rate of Service

The command FIFO depth and the servicing scheme (whether the component reorders the queues of commands or not) affect the head of line blocking and consequently the rate of service. As a result:

- A read command initiated from one master to one slave can overtake a write command from another master to the same slave
- A write command from one master to one slave can overtake a write command from another master to the same or different slaves
- A write from one master to one slave can overtake a write from the same master to another slave.

To be certain of correct command ordering, the slave priority sequence is:

- Write commands followed by another write always proceed in order
- Write command followed by a read command to the same slave, the read command is assured to flush the write command (the read returns the previously written data). To avoid wrong data returns, software should perform dummy read from any address in the slave to assure that the current data has landed.

Depending on the application, it is sometimes beneficial to break large transfers into smaller transfers and use self-chaining to prevent starvation of other events in an event queue. In this case, the time between the sets of transfers can be reduced and the overall throughput will increase.

6 References

- TMS320C6474 Module Throughput (SPRAAW5)

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Clocks and Timers	www.ti.com/clocks	Digital Control	www.ti.com/digitalcontrol
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Telephony	www.ti.com/telephony
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless