Application Report
SPRA479A -September 2001

Interfacing the TMS320C6000 EMIF to a PCI Bus Using the AMCC S5933 PCI Controller

Brian G. Carlson *DNA Enterprises, Inc.*

ABSTRACT

This application report describes the architecture and capabilities of the AMCC S5933 PCI controller and how it can be interfaced to the TMS320C6201 digital signal processor (DSP). The DSP's host port interface (HPI) can be a PCI target, and its external memory interface (EMIF) can be used to support PCI bus mastering. Details on the signals and logic required to implement both PCI slave and master interfaces with the TMS320C6201 DSP are presented. Modifications to the design that allow other TMS320C6000™ series processors in place of the TMS320C6201 are presented. Suggestions are provided on how to improve on the design to achieve greater data throughput.

Contents

List of Figures

List of Tables

List of Examples

1 Introduction

The TMS320C6201 DSP can be interfaced to an off-the-shelf PCI controller to gain access to a PCI bus in adapter card and embedded applications. The AMCC S5933 PCI controller provides a flexible interface to the PCI bus that is compliant with the PCI Local Bus Specification Revision 2.1. The S5933 PCI controller manages all PCI bus transactions and provides a general-purpose, add-on interface. The interface can be used to transfer bi-directional data between a host and the DSP by way of the TMS320C6201's HPI and EMIF. The S5933 interfaces directly to a 5-V, 32-bit PCI bus running up to 33 MHz, supporting a peak bandwidth of 132 Mbytes/s. Its internal registers and FIFOs enable the PCI bus and DSP to operate asynchronously.

The S5933 supports both PCI bus master and slave data transfers to and from the TMS320C6201 DSP memory space. PCI bus master transfers enable the DSP to transfer data to or from the host's or other PCI device's memory space. PCI bus slave transfers enable the host or another PCI device to transfer data to and from the DSP memory space. The use of PCI bus master and slave transfers is application-dependent. One or both of the PCI bus transfer types may be useful in a specific application.

Figure 1 illustrates how the TMS320C6201 DSP interfaces with the AMCC S5933 PCI controller. The S5933 directly interfaces to the PCI bus and handles all PCI bus transactions. A serial EEPROM provides non-volatile storage of PCI configuration information that is automatically downloaded into the S5933 at power-up. Glue logic, which can be implemented in a complex programmable logic device (CPLD), controls the data transfers and interrupts between the S5933 and the DSP.

A 3.3-V CPLD with 5-V-tolerable inputs is required to interface the 5-V PCI controller to the 3.3-V DSP. Voltage translation buffers interface the S5933 5-V signals to the DSP's dedicated HPI data bus. Bus transceivers provide voltage translation and control transmissions between the S5933 and DSP data buses.

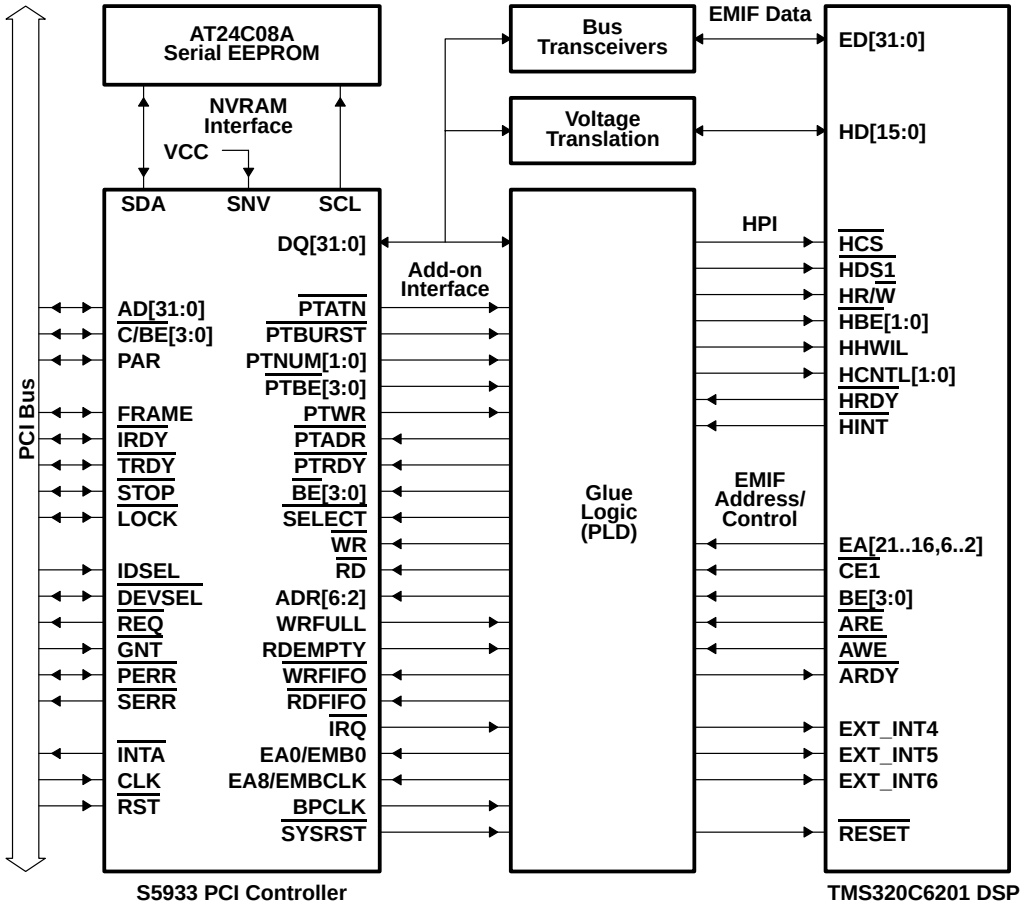

Figure 1. TMS320C6201 Interface to the S5933 PCI Controller

2 S5933 PCI Controller

The S5933 PCI controller is an off-the-shelf device that provides a Revision 2.1-compliant PCI interface. The S5933 supports several modes of operation that allow its general-purpose, add-on interface to be used with different devices, such as the TMS320C6201 DSP. The S5933 provides various ways to communicate between the PCI bus and the DSP, including a bi-directional FIFO interface, mailbox registers, and a pass-thru mode that enables the DSP's HPI to be accessed by way of the PCI bus.

The S5933 PCI controller has three main interfaces, which are the PCI bus, the add-on bus, and the non-volatile memory interface. It also has three sets of registers, which are configuration, PCI operation, and add-on operation registers.

Figure 2 shows a block diagram of the S5933 PCI controller's architecture and interfaces. The following paragraphs summarize the features and interfaces of the S5933. Refer to the AMCC *S5933 PCI Controller Data Book* for detailed information.

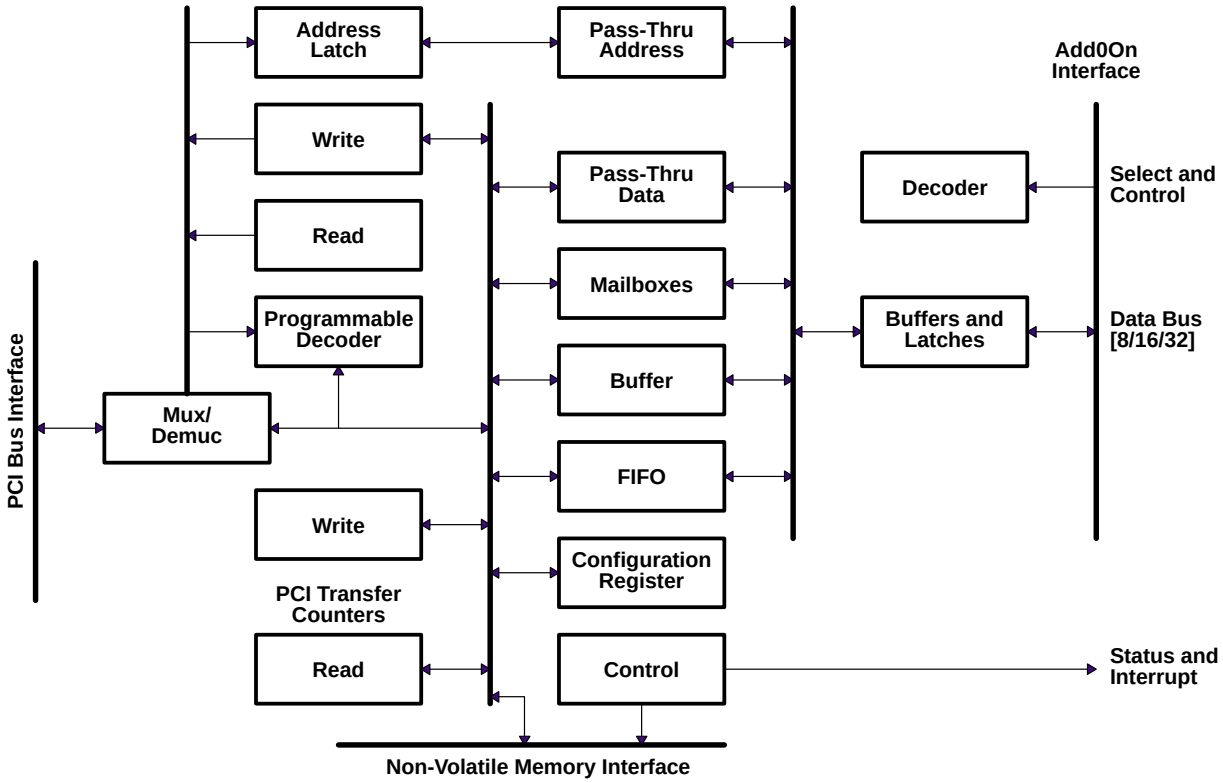

Figure 2. S5933 PCI Controller Block Diagram

2.1 PCI Bus Interface

The S5933 is directly connected to the PCI bus to handle all PCI local bus transactions, including configuration, memory, and I/O accesses. The S5933 device is a PCI bus agent, not a bridge, and is designed to be an endpoint within a given PCI system. This means that it is either the source or destination of a data transfer rather than an intermediate device (bridge) in a PCI data transfer.

2.2 Non-Volatile Memory Interface

At system reset, the S5933 downloads PCI configuration information from external non-volatile memory that is used to support PCI's plug-and-play into its configuration registers. This configuration information includes vendor and device IDs, revision number, class code, memory and I/O space requirements, and interrupt usage. The host system's operating system uses this information for resource allocation and card registration. The S5933 supports downloading from both byte-wide and serial non-volatile memories. If serial memory is used, then additional capabilities, such as external mailbox control, are available because only two pins are required (clock and data) for the interface.

2.3 Add-On Bus Interface

The S5933 provides a general-purpose, 32-bit add-on bus that allows other devices to interface to it. Data transfers to and from the S5933 internal registers are accomplished by way of a chip select decode in conjunction with either a read or write strobe. In addition to control and status register accesses, the add-on bus is used to directly access the S5933's on-chip FIFOs for master PCI transfers and to access its pass-thru address and data registers for slave PCI transfers.

2.4 PCI Configuration Registers

The S5933 contains a 64-byte configuration space region that is required for PCI compliance. These registers, which can only be accessed from the PCI host side, contain the PCI configuration information, including the vendor ID, device ID, revision number, and other information, such as specific device capabilities and the amount of memory required for operation. The configuration registers are intended for use during system initialization and catastrophic error handling.

2.5 FIFOs

The S5933 has two FIFOs that are each 32 bits wide and 8 words deep for transferring data in both directions between the PCI bus to the add-on bus. The FIFOs support PCI bus mastering where the S5933 directly controls the transfer of data on the PCI bus instead of the host. Each FIFO has associated address pointer and transfer count registers to support block transfers. The read and write FIFOs can be directly accessed using the S5933's RDFIFO and WRFIFO strobe pins or can be addressed through their add-on operations register addresses. FIFO flag pins indicate when the read FIFO is empty and the write FIFO is full are available for external logic to control data transfers with the FIFOs. The status of the FIFO flags are also available by reading operation registers.

2.6 Mailbox Registers

The S5933 contains eight 32-bit mailbox registers. Four mailboxes are used to transfer data from the PCI bus to the add-on bus, and four mailboxes are used to transfer data from the add-on bus to the PCI bus. Host and DSP software can detect mailbox empty/full conditions by polling a register or by being interrupted. The mailboxes are useful for transferring data, command, and status information between the host and DSP.

2.7 PCI Operation Registers

The S5933 has 16 32-bit PCI operation registers, accessible only by the host PCI side, that support control and configuration of the interface, provide device status, and implement the PCI to add-on interface mailboxes and FIFOs. The host side also has read/write access to the non-volatile memory via one of the operation registers.

2.8 Add-On Operation Registers

The S5933 has 16 32-bit add-on operation registers, accessible only by the add-on side, that support control and configuration of the interface, provide device status, implement the PCI to add-on interface mailboxes and FIFOs, and present the demultiplexed PCI address and data values associated with pass-thru cycles. The add-on side also has read/write access to the non-volatile memory by way of one of the operation registers.

2.9 Pass-Thru Operation

The S5933 allows PCI bus cycles to be executed in real-time on the add-on interface by providing a simple registered access port to the PCI bus. Using a handshaking protocol with the add-on logic, the PCI bus can read from and write to add-on resources. The S5933 supports up to four individual pass-thru regions in PCI memory or I/O space that are defined by the configuration information stored in the non-volatile RAM. Each pass-thru region's memory width can be individually defined. The pass-thru operation is used for slave transfers where the host controls the data transfers.

The pass-thru logic in the S5933 is comprised of one address register and two data registers (read/write). Status information necessary to define the current pass-thru PCI transaction is provided to the add-on interface on dedicated S5933 pins. Glue logic is required to drive add-on interface and target device control signals to accomplish the data transfers. The S5933 supports both single and burst data transfers in pass-thru operation.

3 TMS320C6201 DSP

The TMS320C6201 DSP can interface to the S5933 PCI controller by way of its HPI and EMIF.

Since the host processor controls HPI data transfers, the HPI can be mapped as a PCI slave device. Since the DSP cannot control HPI data transfers, the HPI cannot be used by the DSP for PCI bus mastering. However, the S5933 PCI controller can be mapped into the DSP's EMIF to support DSP-controlled PCI bus mastering.

The use of the DSP's HPI and EMIF interfaces with the S5933 supports two independent data paths that can be under host and DSP control depending on the needs of an application. Both data paths can be used simultaneously (in a time-sliced fashion as the bus is granted by the PCI arbiter) in some applications to support different types of data transfers.

For example, the PCI bus-mastering interface via the EMIF can be used for full-duplex data streams between the DSP and another processor, while the slave interface is used by another processor to send control and data to the DSP or receive status and data from the DSP. The ability to support two independent PCI data paths provides the flexibility to optimize data flow and simplify software support in some applications.

The TMS320C6201 DSP's HPI allows a PCI bus master to access the complete DSP memory space through a simple 3-register interface that is the front end of a dedicated auxiliary DMA channel. The auxiliary DMA channel connects the HPI to the DSP's memory space, providing direct access to all internal and external memory and memory-mapped peripherals. The HPI includes control, address, and data registers that support single random accesses to the DSP's memory space as well as fast, contiguous block transfers.

The TMS320C6201 DSP's asynchronous operation of its EMIF is compatible with the S5933's add-on bus interface. The S5933's add-on operation registers, including the FIFO and mailbox registers, can be mapped into the DSP's memory space. PCI bus-master transfers can be controlled using DSP DMA channels triggered by the status pins for the S5933's read and write FIFOs or by direct DSP software control using interrupts or polling the FIFO flags.

Figure 3 summarizes the various PCI data paths available to applications when the S5933 PCI controller is interfaced to the TMS320C6201 DSP. The S5933 FIFOs are typically used to support PCI bus mastering as shown but can also be accessed using slave transfers from the host.

The following sections provide detailed descriptions of how the DSP's HPI and EMIF interfaces are used with the S5933 PCI controller for slave and master PCI transfers.

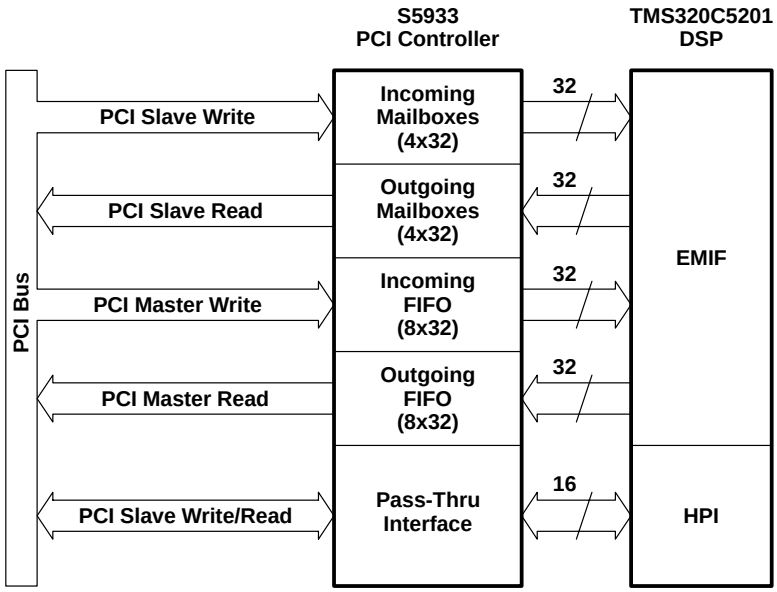

Figure 3. Data Paths of the S5933 PCI Controller/TMS320C6201 DSP Interface

3.1 Host Port Interface (HPI)

The TMS320C6201 DSP's HPI can be interfaced to the S5933 PCI controller for PCI slave transfers, enabling host software read and write access to all of the DSP memory space. Host software can perform both random and sequential accesses using the HPI's registers, which are memory-mapped into the host's memory space.

Before detailing the HPI slave interface implementation, it is important to identify the key issues involved in transferring data between the S5933 and the HPI.

- The HPI presents a 16-bit data interface.
- The S5933 presents a 32-bit data interface that can be configured for an add-on bus width of 16 bits for compatibility with the HPI.
- 32-bit PCI data transfers require two successive 16-bit data transfers.
- Two S5933 base address regions (BARs) can be used to support explicit HPI register addressing for random DSP memory accesses and auto-incremented HPI data addressing for sequential DSP memory accesses.
- Programmable logic is required to monitor the S5933's pass-thru interface and provide the control signals required to transfer data in both directions between the S5933 and the HPI.
- The S5933 is a 5-V device and the DSP requires 3.3-V I/O levels, so voltage translation buffers are required on the S5933 add-on data bus.

Because the HPI is a 16-bit interface, the 32-bit PCI data transfers must be divided into two consecutive 16-bit data transfers. This operation can be supported by configuring the S5933 for a 32-bit add-on interface and its associated HPI BARs for 16-bit pass-thru regions. A 32-bit add-on interface is selected by grounding or floating the S5933's MODE pin. The 16-bit pass-thru regions are selected by initializing bits 31 and 30 of the respective HPI BAR definitions in the NVRAM to [1 0].

In this configuration, the S5933's internal 32-bit data bus can have its byte lanes steered to the lower 16 data bits using the $\overline{BE}$[3:0] byte enable inputs. This action directly supports the requirement to pass the 32-bit PCI data to the 16-bit HPI data interface on a common set of 16 data signals. Since the S5933 is a 5-V device, two Texas Instruments SN74CBTD3384 devices can be used to translate the 16-bit data bus to the 3.3-V DSP.

The HPI PCI slave interface uses the S5933's pass-thru interface, which is comprised of one address register, two data registers, and several control and indicator signals. The S5933 monitors the PCI bus for data transfers that are targeted for BARs defined by the non-volatile memory.

In the case of the HPI, two BARs support random and sequential DSP memory space accesses, respectively. When a PCI master attempts to access either of the two HPI BARs, the S5933 asserts pass-thru indicator signals to notify programmable logic that a pass-thru cycle is initiated. The programmable logic responds to this notification by controlling S5933 and HPI signals to coordinate data transfers.

A data transfer consists of simultaneous register accesses to both the S5933 and the HPI. A register on one side is read with its data presented on the HPI data bus (HD[15:0]). A register on the other side is written with the data. The programmable logic asserts a ready signal back to the S5933 to indicate when each 32-bit PCI data transfer is completed to control the transfer rate with the HPI. Figure 4 shows the HPI interface to the S5933 PCI controller.

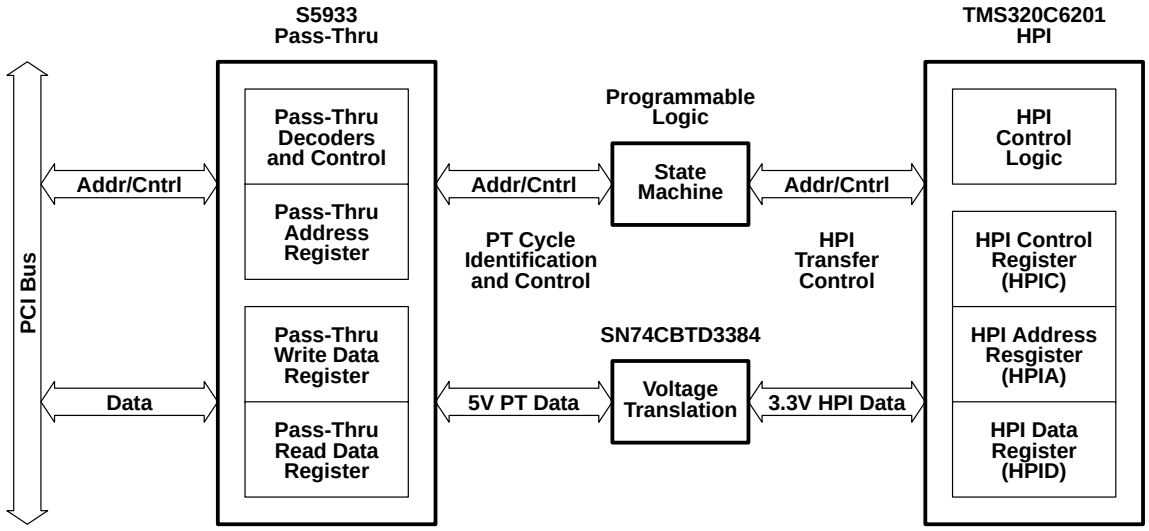

Figure 4. DSP HPI Interface to the S5933 PCI Controller

3.1.1 S5933 Signals

Several S5933 signals are required to interface the PCI controller to the HPI. These signals provide information about the PCI slave data transfers and allow add-on logic to control the transfer of data in both directions between the S5933 and the HPI.

The S5933 pass-thru indicator signals identify the start and type of PCI slave data transfer. These signals are used by the programmable logic to initiate a specific type of data transfer with the HPI. Table 1 identifies the pass-thru indicator signals.

Table 1. S5933 Pass-Thru Indicator Signals

Signal	Function
$\overline{\text{PTATN}}$	Indicates that a pass-thru access is occurring
$\overline{\text{PTBURST}}$	Indicates that a pass-thru access is a PCI burst access
PTNUM[1:]	Indicates which of four pass-thru regions is being accessed
$\overline{\text{PRBE}}$[3:0]	Indicates which data bytes are valid (write) or requested (read)
PTWR	Indicates whether the access is a PCI write or read

The programmable logic state machine must control S5933 pass-thru signals to accomplish the data transfer. These pass-thru control signals allow the state machine to request the target address of the data transfer, access the pass-thru read and write data registers, and indicate to the S5933 when a data transfer is complete. Table 2 identifies the pass-thru control signals.

Table 2. S5933 Pass-Thru Control Signals

Signal	Function
$\overline{\text{PTADR}}$	When asserted, $\overline{\text{PTADR}}$ drives the pass-thru address register contents onto the add-on data bus.
$\overline{\text{BE}}$[3:0]	Individual byte enables used during pass-thru register read and write operations. During reads, they control output drive for each respective byte lane. During writes, they serve as enables to perform the modification of their respective byte lanes.
$\overline{\text{SELECT}}$	Add-on interface select required to read or to write the pass-thru registers
$\overline{\text{WR}}$	Add-on write strobe
$\overline{\text{RD}}$	Add-on read strobe
ADR[6:2]	Add-on address lines to select which of the 32-bit registers within the S5933 is desired for a given read or write cycle. (The add-on pass-thru data register is [0 1 0 1 1].)
$\overline{\text{PTRDY}}$	Indicates that the current pass-thru transfer has been completed by the add-on

The S5933's add-on data bus provides multiplexed address and data information. When the $\overline{\text{PTADR}}$ signal is asserted, the DQ[3:2] signals indicate the target HPI register. The DQ[15:0] signals transfer data to the HPI data bus 16 bits at a time.

The S5933's buffered PCI clock (BPCLK) is used by the programmable logic to clock the state machine that generates the data transfer control signals. This clock has all of the behavioral characteristics of the PCI clock. The state machine is held in the reset state whenever the S5933's system reset ($\overline{\text{SYSRST}}$) output is asserted low. This signal is a buffered form of the PCI bus reset.

3.1.2 HPI Signals

HPI signals are controlled by the programmable logic state machine to read and write HPI registers. These control signals select the HPI register being addressed, the 16-bit word being accessed, the byte lanes that are enabled, the direction of the data transfer, and the timing of the transfers.

Table 3 lists the HPI control signals required for the HPI PCI slave interface. The $\overline{\text{HAS}}$ control signal is not required because the HPI register address is not multiplexed on the HPI data bus. The $\overline{\text{HDS2}}$ control signal is not required because only a single data strobe is used ($\overline{\text{HDS1}}$). These two unused control inputs should be pulled up to 3.3 V.

Table 3. HPI Status Signals

Signal	Function
$\overline{\text{HCS}}$	Chip select must be asserted to enable read and write transfers
$\overline{\text{HDS1}}$	Data strobe that indicates when other control signals and data are valid
HR/$\overline{\text{W}}$	Read/write strobe that is high for reads and low for writes
$\overline{\text{HBE}}$[1:0]	Indicates which bytes in 16-bit word are written. Ignored during reads.
HHWIL	Halfword identification input that is low for first 16-bit word and high for second 16-bit word
HCNTL[1:0]	Access type controls that indicate which internal HPI register is being accessed. (00 = HPIC, 01 = HPIA, 10 = HPID with HPIA inc., 11 = HPID)

The HPI provides two status signals that indicate when the HPI is ready for a data transfer to be performed and when a host interrupt has been asserted. Table 4 lists the HPI status signals.

Table 4. HPI Status Signal

Signal	Function
$\overline{\text{HRDY}}$	HPI is ready for data transfer when asserted low
$\overline{\text{HINT}}$	Interrupt signal to host

3.1.3 Programmable Logic Implementation

The key component of the HPI PCI slave design is the programmable logic state machine that monitors the S5933's pass-thru interface and controls the signals to transfer the data. All data transfers are initiated by a PCI bus master so all information about the transfer is provided by the S5933's pass-thru interface. In response to this information, the state machine generates the corresponding pass-thru and HPI register accesses required for the data transfer. The state machine is clocked by the buffered PCI clock (BPCLK), so it runs at a maximum of 33 MHz (30 ns cycle time).

Three HPI control signals are not directly controlled by the state machine. The HCNTL[1:0] control signals, which select the target HPI register, are decoded from the latched pass-thru address output when the $\overline{\text{PTADR}}$ signal is asserted by the state machine. Since the target address is latched, the HCNTL[1:0] signals remain fixed throughout the HPI register access. The HR/$\overline{\text{W}}$ access-type control signal is the opposite polarity of the S5933's pass-thru PTWR indicator signal. Therefore, the PTWR signal is routed through an inverter to directly generate the HR/$\overline{\text{W}}$ HPI control signal without any state machine interaction. The other five HPI control signals used ($\overline{\text{HCS}}$, $\overline{\text{HDS1}}$, HHWIL and $\overline{\text{HBE}}$[1:0]) are directly controlled by the state machine.

The following four sections describe the implementations of the HPI PCI slave interface for HPI register read, register write, sequential data read and sequential data write accesses.

3.1.3.1 HPI Register Read

A PCI bus master can read the three HPI registers to perform the following functions:

- Determine the selected halfword HPI data transfer order
- Determine the host and DSP interrupt status
- Observe the HPI ready flag
- Read the current HPI address (HPIA) register value
- Read the value from the DSP memory space pointed to by the HPIA register

When an HPI register is read, a simultaneous S5933 register write is required to complete the transfer. The programmable logic state machine performs the requested HPI register read and strobes it into the pass-thru data register for subsequent transfer over the PCI bus to the master.

The programmable logic state machine performs the following steps for an HPI register read. The timing diagram shown in Figure 5 can be referenced for a visual description of these steps.

1. Wait for $\overline{\text{PTATN}}$ low and PTNUM[1:0], indicating a PCI transfer to the HPI registers' pass-thru region.
2. Assert $\overline{\text{PTADR}}$ low to output the target address (HPI register indication). Assert the S5933 pass-thru control signals to prepare for a write to the pass-thru data register ($\overline{\text{SELECT}}$ and ADR[6:2]).
3. Deassert $\overline{\text{PTADR}}$ to end the address phase. Assert the $\overline{\text{WR}}$ pass-thru control signal to begin the data write phase to the pass-thru data register. Assert $\overline{\text{HCS}}$ to enable the HPI.
4. Set HHWIL low to indicate the first halfword of the HPI transfer. Assert the HCNTL[1:0] control signals to select the HPI access cycle. The timing diagram in Figure 5 shows a write the HPID register since HCNTL[1:0] = [1 1].
5. Assert the $\overline{\text{HDS1}}$ signal, which causes the internal HPI logic to latch the other HPI control signals (HR/$\overline{\text{W}}$, HHWIL, HCNTL[1:0]). Assert the $\overline{\text{BE}}$[1:0] signals to enable the write to the lower 16 bits of the pass-thru data register. The DSP should set the $\overline{\text{HRDY}}$ signal high to indicate that the internal DMA operation is in progress. When the data is available to be read from the HPI data register, the DSP will assert the $\overline{\text{HRDY}}$ signal low.
6. Wait for $\overline{\text{HRDY}}$ to be asserted, indicating that the data is available to be read. Since the DSP is asynchronous the PCI controller, the $\overline{\text{HRDY}}$ signal should be synchronized to the PCI clock (BPCLK) before being sampled by the state machine to address metastability. The synchronization and sampling of the $\overline{\text{HRDY}}$ signal requires two PCI clock cycles.
7. Deassert the $\overline{\text{HDS1}}$ signal to end the first halfword transfer. Set the HHWIL signal high to indicate the second halfword of the HPI data transfer. Deassert the pass-thru $\overline{\text{BE}}$[1:0] signals to disable the further writes to lower 16 bits of pass-thru data register. Assert the $\overline{\text{BE}}$[3:2] signals to enable the write to the upper 16 bits of the pass-thru data register. The lower 16 bits of the value being read are clocked into the pass-thru data register on the rising BPCLK edge of this step.
8. Assert the $\overline{\text{HDS1}}$ signal to latch other HPI control signals and begin the second halfword read. Note that the second halfword is always available, so $\overline{\text{HRDY}}$ does not have to be checked.
9. Deassert the $\overline{\text{HDS1}}$ signal to end the second halfword transfer. Deassert the pass-thru $\overline{\text{WR}}$ signal to end pass-thru data register write cycle. Deassert $\overline{\text{BE}}$[3:2] signals to disable further writes to the upper 16-bits of the pass-thru data register. Assert $\overline{\text{PTRDY}}$ to indicate that the data transfer was performed. The upper 16 bits of the value being read are clocked into the pass-thru data register on the rising BPCLK edge of this step.
10. Deassert the $\overline{\text{HCS}}$ signal to disable access to the HPI. Deassert the pass-thru $\overline{\text{SELECT}}$ and ADR[6:2] signals to disable the add-on interface. Deassert $\overline{\text{PTRDY}}$ to end data transfer. Return to step 1) to handle subsequent HPI data transfers.

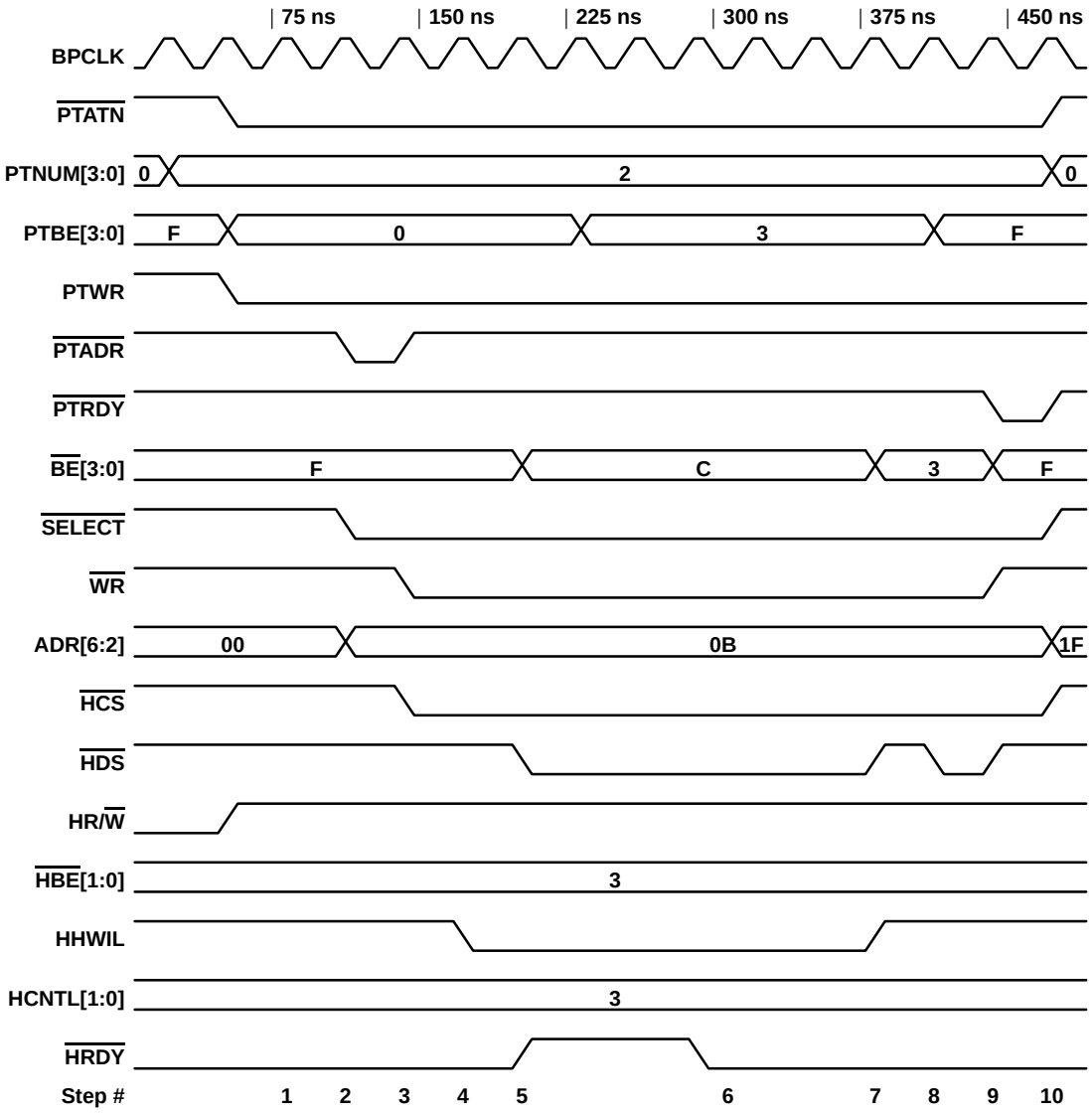

Figure 5. HPI Register Read Control Timing

3.1.3.2 HPI Register Write

A PCI bus master can write to the three HPI registers to perform the following functions:

- Select the halfword HPI data transfer order.
- Assert the DSPINT interrupt to the DSP and clear the HINT host interrupt.
- Request a HPI data fetch under software control.
- Write DSP memory address to the HPI address (HPIA) register.
- Write data into the DSP memory space pointed to by the HPIA register.

When an HPI register is written, a simultaneous S5933 register read is required to access the source PCI data to complete the transfer. The programmable logic state machine performs a read of the pass-thru data register and strobes the data into the HPI data register. When the pass-thru data register is read, and the ready signal is returned to the S5933, the PCI controller can accept more data from a master on the PCI bus.

The programmable logic state machine performs the following steps for an HPI register write. The timing diagram shown in Figure 6 can be referenced for a visual description of these steps.

1. Wait for $\overline{\text{PTATN}}$ low and PTNUM[1:0], indicating a PCI transfer to the HPI registers' pass-thru region.
2. Assert $\overline{\text{PTADR}}$ low to output the target address (HPI register indication). Assert the S5933 pass-thru control signals to prepare for a write to the pass-thru data register ($\overline{\text{SELECT}}$ and ADR[6:2]).
3. Deassert $\overline{\text{PTADR}}$ to end the address phase. Assert the $\overline{\text{BE}}$[1:0] signals to enable a read from the lower 16 bits of the pass-thru data register. Assert the $\overline{\text{RD}}$ pass-thru control signal to begin the data read phase from the pass-thru data register. Assert $\overline{\text{HCS}}$ to enable the HPI.
4. Set HHWIL low to indicate the first halfword of the HPI transfer. Set the $\overline{\text{HBE}}$[1:0] byte enables corresponding to the values of $\overline{\text{PTBE}}$[1:0] for the first halfword transfer. Assert the HCNTL[1:0] control signals to select the HPI access cycle. The timing diagram shows a write the HPID register since HCNTL[1:0] = [1 1].
5. Assert the $\overline{\text{HDS1}}$ signal which causes the internal HPI logic to latch the other HPI control signals (HR/$\overline{\text{W}}$, HHWIL, HCNTL[1:0]).
6. Wait for $\overline{\text{HRDY}}$ to be asserted, indicating that the HPI is ready to accept another transfer. In most cases, $\overline{\text{HRDY}}$ should be asserted immediately, but there is a chance that a previous HPI write or prefetch read may not have completed yet. Since the DSP is asynchronous the PCI controller, the $\overline{\text{HRDY}}$ signal should be synchronized to the PCI clock (BPCLK) before being sampled by the state machine to address metastability. This adds an extra PCI clock delay before the state machine can respond to it.
7. Deassert the $\overline{\text{HDS1}}$ signal to end first halfword transfer. Set the HHWIL signal high to indicate the second halfword of the HPI data transfer. Deassert the pass-thru $\overline{\text{BE}}$[1:0] signals to disable the output of the lower 16 bits of pass-thru data register. Assert the $\overline{\text{BE}}$[3:2] signals to enable the output of the upper 16 bits of the pass-thru data register.
8. Assert the $\overline{\text{HDS1}}$ signal to latch other HPI control signals. Note that the second halfword can always be immediately written, so $\overline{\text{HRDY}}$ does not have to be checked.
9. Deassert the $\overline{\text{HDS1}}$ signal to end second halfword transfer. The HCNTL[1:0] control signals can be optionally set to a default state of [1 1]. When $\overline{\text{HDS1}}$ is brought back high, the auxiliary DMA controller initiates the write into the DSP memory space. The $\overline{\text{HRDY}}$ signal is deasserted during this operation.
10. Deassert the $\overline{\text{HCS}}$ signal to disable access to the HPI. When $\overline{\text{HCS}}$ is not asserted, the $\overline{\text{HRDY}}$ signal returns to its default asserted state. Deassert the pass-thru $\overline{\text{RD}}$ signal to end pass-thru data register read cycle. Deassert $\overline{\text{BE}}$[3:2] signals to disable the data from the upper 16-bits of the pass-thru data register. Assert $\overline{\text{PTRDY}}$ to indicate that the data transfer was performed.
11. Deassert the pass-thru $\overline{\text{SELECT}}$ and ADR[6:2] signals to disable the add-on interface. Deassert $\overline{\text{PTRDY}}$ to end data transfer. Return to step 1) to handle subsequent HPI data transfers.

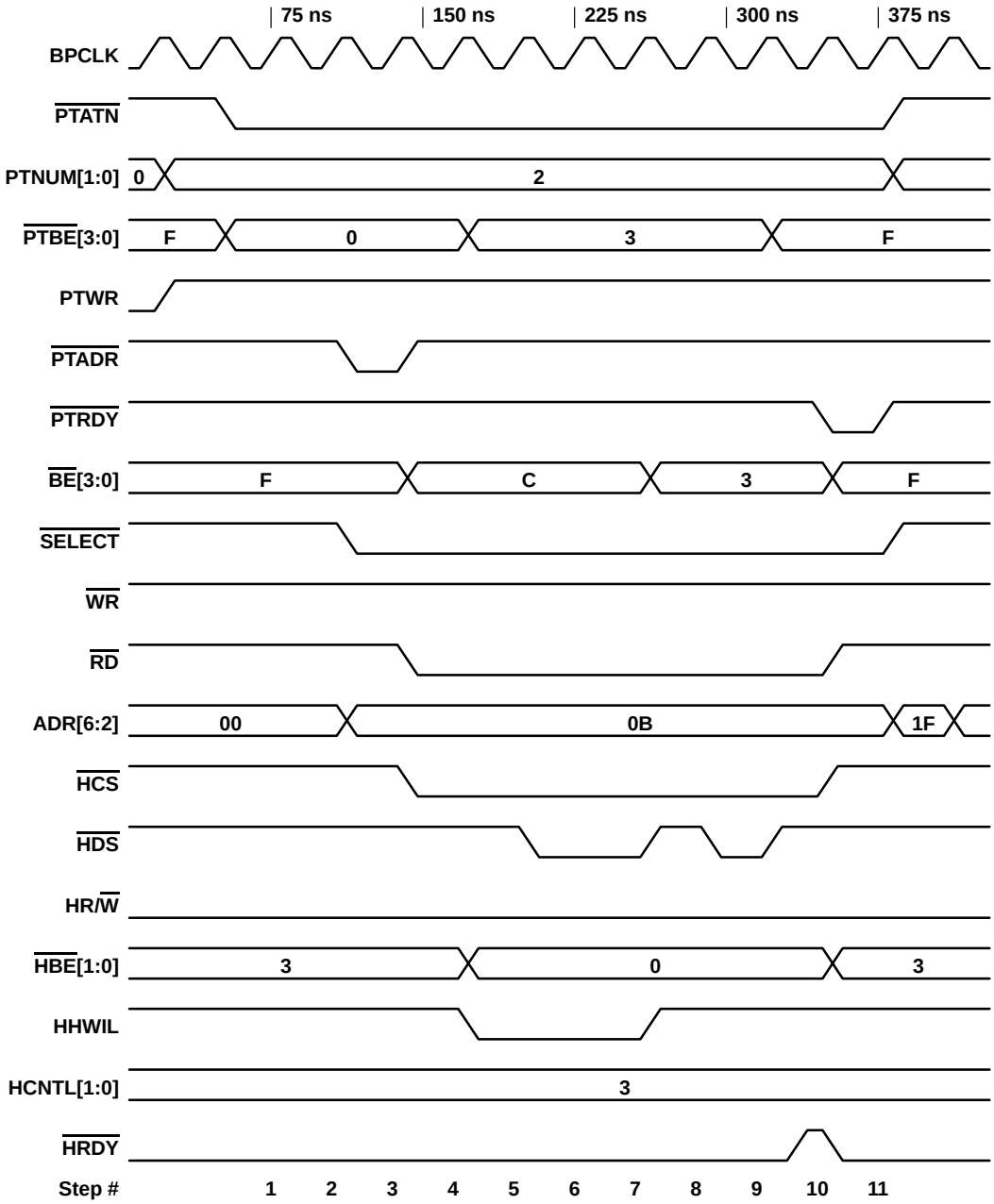

Figure 6. HPI Register Write Control Timing

3.1.3.3 HPI Sequential Data Reads

A PCI bus master can take advantage of the HPI's sequential memory access capability to efficiently support block read transfers. The HPI's address auto-increment feature facilitates reading from sequential memory locations with data prefetching to reduce the latency on subsequent host read requests. In this mode of HPI operation, the HPI address register (HPIA) is initialized to the DSP memory source address for the block read transfer. When the HCNTL[1:0] control signals are configured for [1 0], all subsequent reads of the HPI data register (HPID) increment the HPIA automatically and prefetch the data at the next address in anticipation of the next HPI data read.

This capability simplifies data transfer coordination on the host side because the source address only has to be written once and the data can be read sequentially thereafter without writes to other registers. This allows the host software to implement efficient block data transfers using tight loops that may result in efficient PCI bus data bursts. PCI data bursts increase data throughput because they eliminate the repetitive bus arbitration and address phase overhead involved in single-word transfers.

A separate PCI pass-thru memory region defined by one of the S5933 PCI controller's BARs can be used for dedicated, sequential memory accesses. Since all data transfers to this dedicated memory region are targeted for the HPID register, the detailed address information provided on the multiplexed PCI bus is not needed. Therefore, the address phase of each PCI data transfer to this region can be eliminated, allowing data to be transferred from the HPI immediately. This means that the S5933's PTADR control signal is not required to support HPI block reads.

An additional benefit of using a dedicated memory region for sequential memory accesses is that the HPI HCNTL[1:0] control signals can be forced to [1 0] to select the HPID with address auto-incrementing for all of its accesses. This provides a very simple decoding mechanism for supporting block transfers.

From the host software, the dedicated memory region for block transfers can be viewed as a linear or circular buffer. The key point is that the data transfer must not extend past the memory region's defined memory space defined in the PCI configuration NVRAM. Memory accesses beyond the end of this memory space are not accepted by the S5933 and can cause system problems. Therefore, for proper operation, it is important to ensure that all host memory accesses are maintained in the memory region's address space.

The control timing for HPI sequential data reads is the same as individual HPI register reads with the following exceptions:

- The pass-thru address phase is eliminated, so the steps involving the assertion and deassertion of the $\overline{\text{PTADR}}$ control signal are not required.
- The HCNTL[1:0] control signals are forced to [1 0] for all accesses to the dedicated memory region, rather than being decoded from the latched pass-thru address register value.
- At the end of the each 32-bit data transfer, the $\overline{\text{PTATN}}$ and $\overline{\text{PTBURST}}$ indicator signals should be sampled to determine if it is a burst data transfer with subsequent data being requested and an appropriate action should be taken. Table 5 summarizes the four combinations of these signal states and the corresponding actions to take.

Table 5. PCI HPI Burst End-of-Transfer Actions

PTATN	PTBURST	Status	Action
0	0	Not end of burst. S5933 is ready for next read.	Read HPID to get data from next memory address.
0	1	One more read is left in burst.	Read HPID to get last data in burst from next memory address.
1	0	Not end of burst, S5933 is not ready for next read.	Wait for PTATN to be asserted before reading HPID to get data from next memory address.
1	1	End of burst	Return to idle state awaiting next pass-thru transfer indicated by PTATN being asserted.

3.1.3.4 HPI Sequential Data Writes

A PCI bus master can also take advantage of the HPI's sequential memory access capability to efficiently support block write transfers. The HPI's address auto-increment feature facilitates writing to sequential memory locations after initializing the HPIA register with the block's destination address in the DSP memory space. When the HCNTL[1:0] control signals are configured for [1 0], all subsequent writes to the HPI data register (HPID) increment the HPIA automatically.

The same dedicated pass-thru memory region used for HPI sequential data reads should be used for data writes. The state machine should check the PTWR indicator signal to determine whether a read or write access is being made to the HPID register.

The control timing for HPI sequential data writes is the same as individual HPI register writes, with the same exceptions identified for HPI sequential data reads. The pass-thru PTATN and PTBURST indicator signals must similarly be sampled at the end of each data word transfer to determine the corresponding action to take, as summarized in Table 5.

3.1.3.5 HPI PCI Slave Performance

The required HPI transfer protocol defines the maximum transfer rate between the host and the DSP to transfer two 16-bit words, the 30-ns state machine clock period, and the response time of the HPI's auxiliary DMA controller. The HPI is a 16-bit wide asynchronous interface, so overhead is required to interface to the 32-bit wide synchronous interface of the PCI bus. Because the state machine must be synchronous to the 33-MHz PCI clock, there is a minimum of 30 ns between control signal transitions. The HPI response (HRDY) time typically varies between 11 and 29 CPU clocks[4] but can be longer depending on bus conflicts. Assuming that a 32-bit data transfer requires 9 states at 30 ns plus 11 CPU clocks at 5 ns, an HPI PCI slave throughput rate of about 12 Mbytes/s can be achieved.

3.2 External Memory Interface (EMIF)

The TMS320C6201 DSP's asynchronous EMIF can be interfaced to the S5933 PCI controller to access the S5933's add-on registers, including control, status, mailbox, and FIFO registers. DSP access to the add-on FIFO registers enables support for PCI bus mastering where the DSP resource controls data transfers on the PCI bus rather than the host.

Before detailing the EMIF interface implementation, it is important to identify the key issues involved.

- Both the DSP's EMIF and the S5933 add-on data buses are 32 bits wide.
- Programmable logic is required to decode the DSP's EMIF control and address signals, enable data bus transceivers, and generate S5933 control signals for add-on accesses.
- Programmable logic is required to monitor S5933 FIFO flags and generate interrupts to the DSP for PCI bus mastering data transfer control.
- The S5933 generates an interrupt to the DSP to indicate defined mailbox states, end-of-transfers, and error conditions.
- Because the S5933 is a 5-V device and the DSP requires 3.3-V I/O levels, voltage translation buffers are required on the S5933 add-on data bus.

Because the S5933 add-on and DSP EMIF data buses are both 32 bits, data transfers involve single 32-bit data transfers. The S5933 is mapped into one of the DSP's memory spaces (CE0-CE3) that has its corresponding memory space control register set for 32-bit asynchronous memory. Figure 7 shows the interface between the EMIF and the S5933 PCI controller.

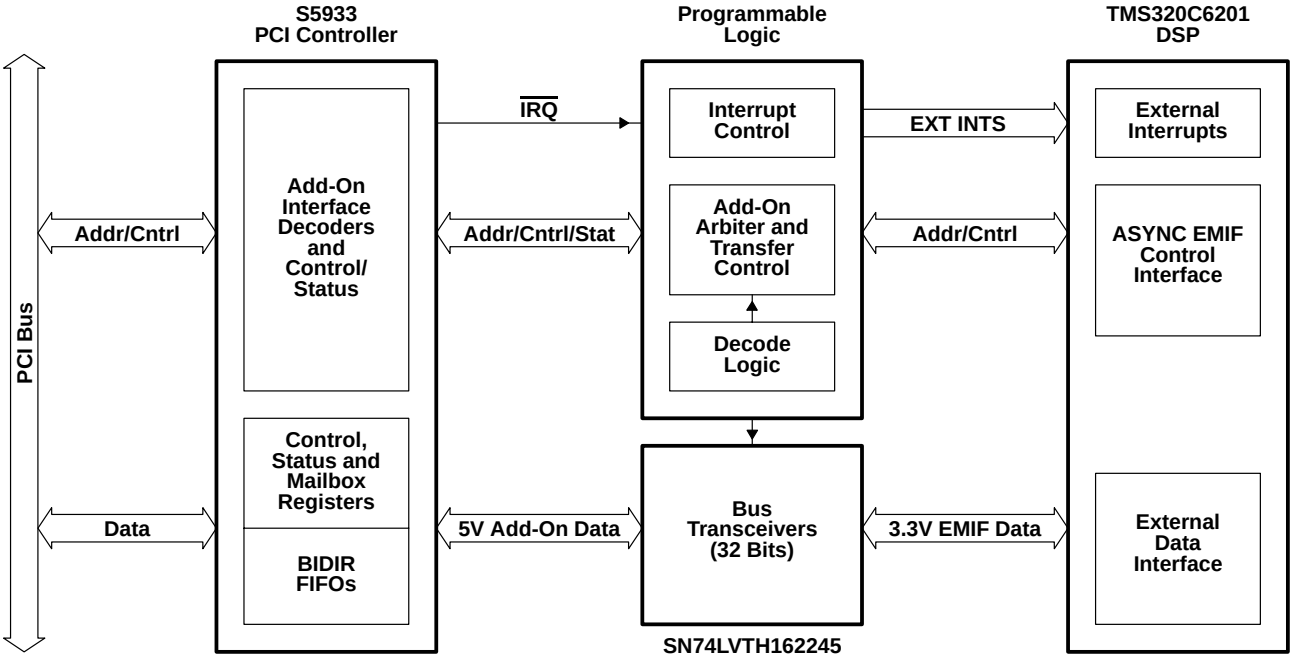

Figure 7. DSP EMIF Interface to the S5933 PCI Controller

The S5933 add-on interface presents 18 operation registers mapped on 32-bit word boundaries. Two of these registers are used for pass-thru support, so only 16 are actually used by the EMIF. The add-on operation registers are accessed by asserting the add-on bus chip select pin ($\overline{\text{SELEC}}$) and address pins in conjunction with either the read ($\overline{\text{R}}$) or write ($\overline{\text{W}}$) control strobe. Access to the bi-directional FIFOs can also be achieved using the dedicated $\overline{\text{RDFIFO}}$ and $\overline{\text{WRFIF}}$ pins. Table 6 lists the add-on operation registers that provide the communication interface between the DSP and the S5933.

Table 6. S5933 Add-On Registers

Address Offset	Register Name	Description	Access Type
0x00	AIMB1	Add-on Incoming Mailbox Register #1	Read Only
0x04	AIMB2	Add-on Incoming Mailbox Register #2	Read Only
0x08	AIMB3	Add-on Incoming Mailbox Register #3	Read Only
0x0C	AIMB4	Add-on Incoming Mailbox Register #4	Read Only
0x10	AOMB1	Add-on Outgoing Mailbox Register #1	Read/Write
0x14	AOMB2	Add-on Outgoing Mailbox Register #2	Read/Write
0x18	AOMB3	Add-on Outgoing Mailbox Register #3	Read/Write
0x1C	AOMB4	Add-on Outgoing Mailbox Register #4	Read/Write
0x20	AFIFO	Add-on FIFO Port	Read/Write
0x24	MWAR	Bus Master Write Address Register	Read/Write
0x28	APTA	Add-on Pass-Thru Address Register	Read Only
0x2C	APTD	Add-on Pass-Thru Data Register	Read/Write
0x30	MRAR	Buss Master Read Address Register	Read/Write
0x34	AMBEF	Add-on Mailbox Empty/Full Status	Read Only
0x38	AINT	Add-on Interrupt Control Register	Read/Write
0x3C	AGCSTS	Add-on General Control/Status Register	Read/Write
0x58	MWTC	Bus Master Write Transfer Count	Read/Write
0x5C	MRTC	Bus Master Read Transfer Count	Read/Write

Programmable logic performs memory decoding, S5933 access arbitration, direct control of the S5933 add-on interface, and interrupt generation. Logic determines when the DSP is attempting to access the S5933 add-on bus by decoding the EMIF address and control signals. A PCI bus master can simultaneously access the DSP's HPI interface as a slave, so the add-on interface may not be available to the EMIF.

Arbitration logic must be included to allow a PCI bus master or the DSP access to the S5933 add-on bus at one time. If the add-on bus is not available to the DSP, the pending asynchronous access is held off by not asserting the EMIF ARDY signal. When the EMIF is granted access to the add-on bus, data bus transceivers, such as the TI SN74LVTH162245, are enabled to connect the EMIF and add-on data buses and provide the required voltage translation. A state controls the S5933 chip select, address, and read/write strobes to achieve the data transfer as well as the DSP's ARDY signal to indicate the end of the data transfer.

External DSP interrupts support the interface between the S5933 and the TMS320C6201. One interrupt is dedicated for general PCI controller notifications, such as mailbox status, PCI transfer status, and error conditions. The second interrupt notifies the DSP when data can be written to the S5933's write FIFO. The third interrupt notifies the DSP when data can be read from the S5933's read FIFO. The FIFO interrupts can be used to directly control the synchronization of DSP DMA controllers to move data in the background.

3.2.1 S5933 Signals

Several S5933 signals are required to interface the PCI controller to the EMIF for DSP access to the S5933's add-on registers and bi-directional FIFOs.

The S5933 add-on registers are accessed by asserting the device's chip select, selecting the register offset and desired byte enables, and strobing the data with either the write or read strobe. Table 7 lists the signals required to provide the DSP access to the S5933's add-on registers.

Table 7. S5933 Add-On Register Access Signals

Signal	Function
$\overline{\text{SELECT}}$	Add-on interface chip select
ADR[6:2]	Add-on address lines to select which of the 32-bit registers within the S5933 is desired for a given read or write cycle
$\overline{\text{WR}}$	Add-on write strobe
$\overline{\text{RD}}$	Add-on read strobe
$\overline{\text{BE}}$[3:0]	Individual byte enables used during register read/write accesses. During reads, they control the output drive for each respective byte lane. During writes, they serve as enables to perform the modification of their respective byte lanes.

The S5933 provides dedicated signals that indicate the status of the read and write FIFOs and allow direct access to the FIFOs without requiring the use of the chip select and read/write strobes. Table 8 lists the signals related to the FIFOs.

Table 8. S5933 FIFO Access Signals

Signal	Function
WRFULL	Write FIFO is full when asserted.
RDEMPTY	Read FIFO is empty when asserted.
$\overline{\text{WRFIFO}}$	Automatically writes add-on data bus information into FIFO
$\overline{\text{RDFIFO}}$	Automatically drives FIFO data onto add-on data bus

The S5933 also provides an $\overline{\text{IRQ}}$ signal that is the source of the general PCI controller interrupt to the DSP. The S5933's buffered PCI clock (BPCLK) is used by the programmable logic to clock that state machine that generates the data transfer control signals. The state machine is held in the reset state whenever the S5933's system reset ($\overline{\text{SYSRST}}$) output is asserted low.

3.2.2 EMIF Signals

The S5933 is accessed by the DSP, as are other asynchronous devices. It is memory-mapped to a certain DSP memory space configured for 32-bit asynchronous memory accesses. (The CE1 memory space is used in this application report.)

The external address (EA) signals decode memory accesses to the S5933 or an EMIF request to the S5933's add-on bus. The upper address bits EA[21:16] are used in this application report to determine when the S5933 is being accessed and the lower address bits EA[6:2] are used to access one of the specific add-on registers.

Table 9 lists the DSP's asynchronous EMIF signals used to interface to the S5933 PCI controller.

Table 9. Asynchronous EMIF Signals

Signal	Function
ED[31:0]	External data bus
EA[21:16]	External address signals used to decode accesses to S5933 PCI controller
EA[6:2]	External address signals used to access S5933 add-on registers
$\overline{\text{CE1}}$	Indicates when selected external memory space (configured for 32-bit asynchronous accesses) is being accessed
$\overline{\text{BE}}$[3:0]	Individual byte lane enables indicating which bytes are selected for both read and write accesses
$\overline{\text{ARE}}$	Asynchronous read strobe
$\overline{\text{AWE}}$	Asynchronous write strobe
ARDY	Asynchronous ready input used to assert wait states as needed

Three external DSP interrupts support the DSP interface to the S5933. One is used for general PCI notifications and the other two synchronize the transfer of PCI bus mastering data. Table 10 lists the DSP external interrupts used in this application report.

Table 10. External Interrupt Signals

Signal	Function
EXT_INT4	PCI controller interrupt (mailbox status, transfer done, errors)
EXT_INT5	PCI bus master reads DMA synchronization
EXT_INT6	PCI bus master writes DMA synchronization

3.2.3 Programmable Logic Implementation

The following four areas of programmable logic are associated with the EMIF interface to the S5933 PCI controller:

- Decode logic
- Add-on arbiter
- Data transfer control
- Interrupt control

3.2.3.1 Decode Logic

Decode logic monitors the EMIF address signals to determine when the S5933 is being accessed. In this application report, the upper address signals (EA[21:16]) are used to decode the S5933's operation registers and FIFOs. Two separate memory regions are used, with one mapped for access to the operation register bank and the other mapped for direct FIFO access. The FIFOs can be accessed at offset 0x20, but for design flexibility a separate memory region is also provided to take advantage of the direct FIFO access capability and support the potential use of external FIFOs to increase the depth of the on-chip FIFOs. The output of the decode logic is a flag indicating that an access to the S5933 is being made by the DSP. Example 1 illustrates the decoding logic.

Example 1. Decode Logic Showing Access to the S5933

```
"Constants
EA =   [EA21, EA20, EA19, EA18, EA17, EA16];
PCI_REG_ADDR   =  [1, 1, 0, 0, 0, 0];
PCI_FIFO_ADDR  =  [1, 1, 0, 0, 0, 1];

"S5933 Decode (EMIF Request)
emif_req = !CE1_ &
                   ( (EA = PCI_REG_ADDR)  (# EA = PCI_FIFO_ADDR));
```

3.2.3.2 Add-On Arbiter

A state machine is required to perform arbitration of the add-on bus and to control the EMIF accesses to the S5933. This is the same state machine used to handle the HPI PCI slave transfers discussed previously. The state machine is expanded to handle the EMIF accesses of the S5933. Add-on bus arbitration is inherent in the state machine design since the state machine only services one request at a time. If the EMIF requests an access to the S5933 while an HPI transfer is taking place, it is held off (ARDY is not asserted) until the HPI transfer has completed. Likewise, an HPI access does not begin until the current EMIF access has completed.

The state machine sets the *emif_ack* flag when the EMIF is granted access to the add-on bus. This flag is used in the generation of the ARDY signal back to the DSP.

```
ARDY =  emif_req & emif_ack & (!ARE_ # !AWE_)
```

ARDY is asserted when the EMIF requests and is granted the add-on bus, and the read/write strobe ($\overline{\text{ARE}}$/$\overline{\text{AWE}}$) is asserted. ARDY is deasserted when the DSP deasserts the read/write strobe.

3.2.3.3 Data Transfer Control

There are four types of data transfers between the EMIF and the S5933:

- Add-on operation register write
- Add-on operation register read
- Add-on FIFO write
- Add-on FIFO read

The type of data transfer is determined by the DSP's external address (EA), $\overline{\text{ARE}}$, and $\overline{\text{AWE}}$ signals. Based on these values, the state machine branches to states that handle the particular type of data transfer. For an S5933 add-on operation register access, the state machine controls the S5933 signals defined in Table 7.

For an S5933 add-on FIFO access, the state machine only controls the $\overline{\text{RDFIFO}}$ signal for reads and the $\overline{\text{WRFIFO}}$ signal for writes. The $\overline{\text{RDFIFO}}$ and $\overline{\text{WRFIFO}}$ strobes are synchronized to the BPCLK, but the S5933 is configured for asynchronous FIFO accesses (bits 5 and 6 are set at offset 0x45 of the configuration NVRAM). Asynchronous FIFO accesses are directly compatible with the DSP's asynchronous EMIF.

The DSP's memory space control register value must select a minimum of 3 CLKOUT periods for the read and write strobes to support the ARDY logic.

Figure 8 shows the control signal timing for a write to an S5933 add-on register. Figure 9 shows the control signal timing for a read from the S5933 FIFO.

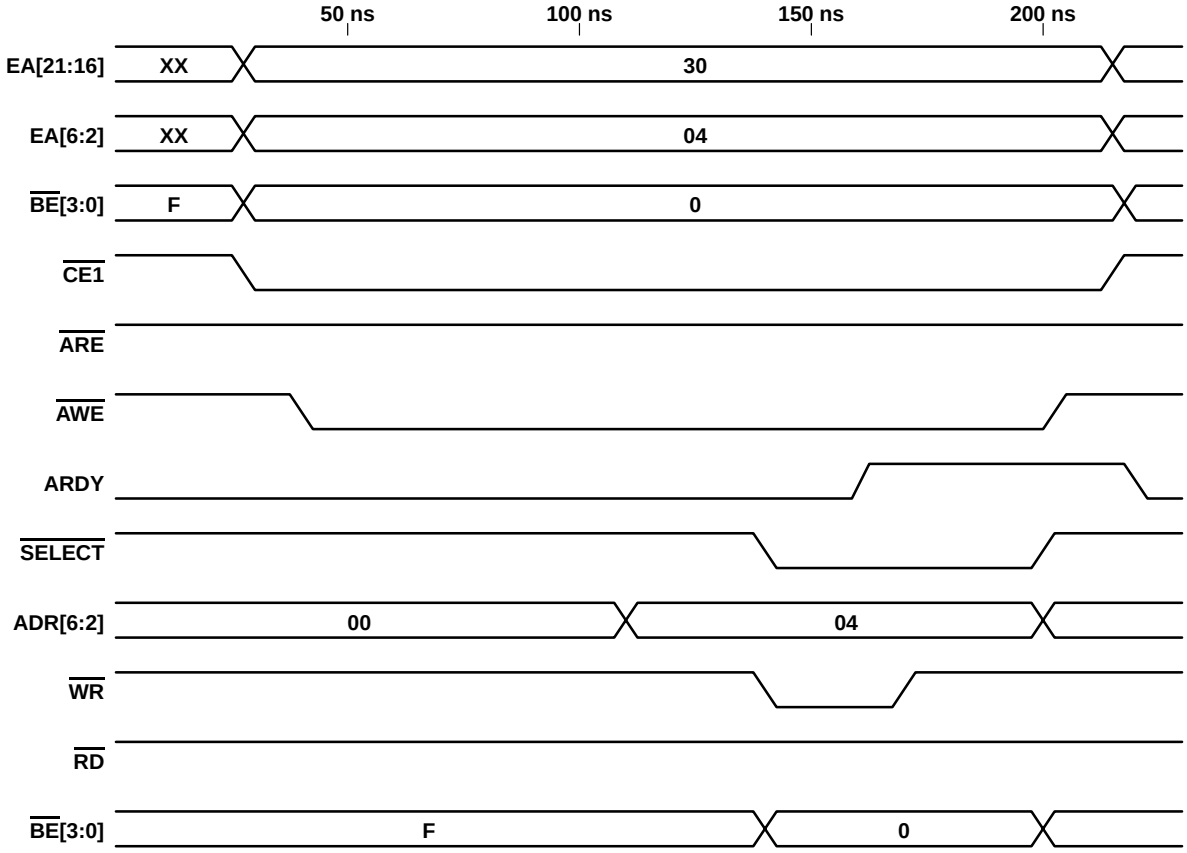

Figure 8. EMIF Write to S5933 Add-On Register

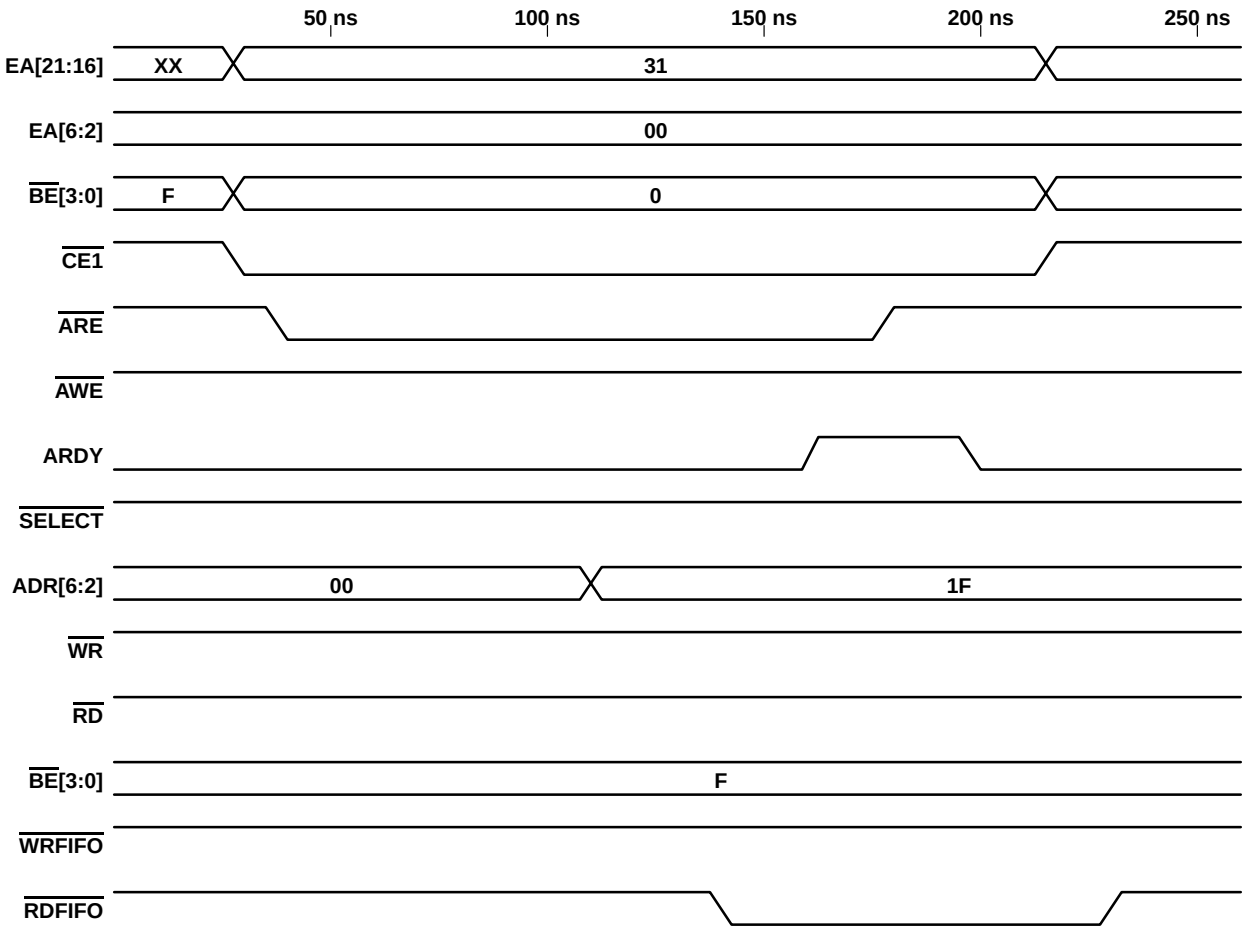

Figure 9. EMIF Read From S5933 Add-On FIFO

3.2.3.4 Interrupt Control

Two types of interrupt control are associated with the EMIF interface to the S5933. The first type of interrupt control consists of the generation of the EXT_INT4 interrupt, which indicates a general PCI controller event, such as mailbox, end-of-transfer, or error condition. This interrupt can be routed directly to the EXT_INT4 pin, but its polarity (falling edge) is the opposite of the DSP's default polarity. For consistency with other system interrupt polarities, the S5933's $\overline{IRQ}$ output can be inverted so that EXT_INT4 is asserted on a rising edge. Since each interrupt's polarity can be uniquely programmed, the hardware inversion is optional.

The second type of interrupt control consists of the generation of the DSP's EXT_INT5 and EXT_INT6 interrupts, which indicate when it is time to read and write the S5933 FIFOs, respectively. These interrupts synchronize read and write DSP DMA data transfers between the DSP and the S5933. The sources of these interrupts are the status of the S5933's RDEMPTY and WRFULL signals, respectively. Two independent state machines generate the EXT_INT5 (read) and EXT_INT6 (write) interrupts. They both work the same, so only the EXT_INT5 interrupt generation is described.

In this example, a register bit named *pcimren*, located in the programmable logic, enables the interrupt generation. The *pcimren* bit, which can be directly controlled by DSP software, is a flag used to enable the interrupt generation state machine. The following four states are required to implement the state machine:

- State MRD_IDLE: Set EXT_INT5 low (no interrupt yet). When (pcimren = 1) and (RDEMPTY = 0) goto MRD_INT, else MRD_IDLE. This state waits for the RDEMPTY flag to be 0 because this indicates that the read FIFO is not empty, so data can be read from it. This is the default state at reset.
- State MRD_INT: Set EXT_INT5 high to interrupt DSP. Goto MRD_WAIT. This state causes an external interrupt that triggers a DMA transfer that reads the S5933 FIFO.
- State MRD_WAIT: Keep EXT_INT5 high. If (pcimren = 0) or (RDFIFO = 0), then goto MRD_DONE, else MRD_WAIT. This state waits for the state machine to be disabled or the start of a FIFO read access to begin.
- State MRD_DONE: Keep EXT_INT5 high. If (pcimren = 0) or (RDFIFO = 1), then goto MRD_IDLE, else MRD_DONE. This state waits for the state machine to be disabled or the FIFO read access to complete. It is important to wait for the FIFO access to complete and the FIFO flags to be updated before they are sampled again to prevent multiple interrupts.

3.2.4 EMIF PCI Bus Master Performance

PCI bus data throughput is optimized when burst transfers are used to achieve the peak bandwidth of 132 Mbytes/s. Several factors can influence the ability to maintain burst transfers, including the PCI components in the system. The S5933 supports burst transfers and can achieve peaks of 132 Mbytes/s. However, a limiting factor is how fast the add-on device can provide the data and if it can keep the pipeline full to remain the bus master and achieve long bursts of data.

The S5933 provides an internal, 8-word deep FIFO that is not sufficient to maintain long bursts, if the add-on device cannot provide data fast enough to keep the internal FIFO full. This can be addressed using external, high-speed dual-port SRAM or FIFOs that can be filled by the add-on device before the PCI bus is requested, resulting in long bursts of data at the peak rate.

The design described in this application report provides a simple, low-cost EMIF interface to the S5333 PCI controller that provides about 12 Mbytes/s average throughput, which is comparable to the HPI slave interface throughput. The EMIF bus mastering has an advantage over HPI slave transfers because bus transfers are handled in hardware without requiring host software to explicitly transfer the data. This means that bus mastering with the EMIF requires less processing time on the host side to move the data, which may be important in some applications.

The design described in this application report uses the TMS320C6x evaluation module (EVM) and had to meet the following constraints: the design had to be simple, low-cost, and able to support at least 10 Mbytes/s throughput. All of these constraints are satisfied with the design. For additional information, including programming, refer to the *TMS320C66201/6701 Evaluation Module User's Guide* (SPRU269).

Improved throughput can be realized using external memories to increase the size of the PCI bursts and to decouple the DSP from the PCI controller's limited FIFO depth and full/empty indicator flags. If external memory is used, the DSP can burst blocks of data to and from it on each interrupt assertion instead of incurring the additional latency and overhead associated with one word transferred per interrupt. On the PCI side, the S5933 can maintain the bus longer as it moves data between the external memory and its internal FIFO without interruption. It is possible to increase the data throughput to an average of about 55-60 Mbytes/s to host memory using external memories.

4 Other TMS320C6000 Devices

This interface between the AMCC S5933 PCI controller and the TMS320C6201 requires only minor modifications to work with other TMS320C6000 devices. To use this setup, only DSPs that include a Host Processor Interface (HPI) can be used. This includes the following DSPs: TMS320C620x, TMS320C621x/C671x and TMS320C64x™.

4.1 HPI Differences

The HPI peripheral of a TMS320C6000 DSP is either 16-bits or 32-bits wide. The 16-bit HPI is comparable to the HPI of the TMS320C6201 and has only a few variations between the different DSPs. The 32-bit HPI is somewhat different, but simplifies the design by sending 32-bit words rather than pairs of 16-bit halfwords.

4.1.1 TMS320C621x/C671x and TMS320C64x 16HPI

The C621x/C671x and C64x™ 16HPI pin interface is similar to the C620x HPI interface, except that byte enables ($\overline{\text{HBE}}$[1:0] in C620x) are not supported. Thus, all access through the 16-bit data bus have to be in pairs. This requires modification of the state machine to automatically transfer the second 16-bit halfword without using the HPI byte enable pins.

Unlike the C620x HPI interface, which used the DMA auxiliary channel to perform accesses, the C621x/C671x HPI ties directly into internal address generation hardware. No specific EDMA channel is used for performing C621x/C671x HPI accesses. Instead the internal address generation hardware, which is not visible to users, handles the read/write requests and accesses. Additionally, an 8-deep read and an 8-deep write buffer has been added to the C621x/C671x HPI and a 16-deep read and a 32-deep write buffer has been added to the C64x. This should improve the PCI burst transfer performance and transmission rates.

4.1.2 TMS320C64x 32HPI

The TMS320C64x 32HPI has a 32-bit HPI. This simplifies the interface between the S5933 and the DSP, allowing each transfer to complete with one 32-bit transfer rather than two 16-bit halfword transfers. Transferring an entire word at a time eliminates the need for halfword selection, so the HHWIL and HBE[1:0] pins are absent in this mode. The 32-bit HPI adds 16 extra pins (HD[31:16]) to the interface and requires two more Texas Instruments SN74CBTD3384 devices to translate the voltage of the 5-V S5933 to the 3.3-V DSP.

The S5933 needs to be configured to have a 32-bit pass-thru region to support 32-bit HPI transfers. This can be accomplished by initializing bits 31 and 30 of the respective HPI BAR definitions in the NVRAM to [1:1]. The Add-on bus keeps the same setting of 32-bits that it had for the original interface. This is accomplished by grounding or floating the MODE pin on the S5933.

TMS320C64x and C64x are trademarks of Texas Instruments.

In 32-bit mode, the S5933's $\overline{BE}$[3:0] byte-enable-inputs pins keep the same setting for the duration of the transfer. This simplifies the state machine design by eliminating the need for some of the states that were only involved in the second halfword transfer. The 32-bit interface should offer immediate performance improvements by lowering the number of clock cycles required for each transfer. Additionally, PCI burst transfers should improve due to the write and read buffers on the C64x series of DSPs.

4.2 EMIF Differences

The EMIF peripheral is included in on all TMS320C62x™ generation DSPs. There are only a few differences between them that should be noted. The first change involves the addition of an EMIF clock input (ECLKIN) to the C621x/C671x and C64x EMIF. This input is required for the C621x/C671x and is optional for the C64x. In this interface this pin should be hooked to the BPCLK output of the S5933.

On the C621x/C671x and C64x EMIF the $\overline{ARE}$ and $\overline{AWE}$ output pins are multiplexed with other outputs that are not used in this interface. To set up the pins to work as $\overline{ARE}$ and $\overline{AWE}$, the space control registers must be set for an asynchronous interface. This is accomplished by setting the MTYPE field of the CExCTL register to 0010b (32-bit-wide asynchronous interface).

The C64x EMIF peripheral is different than the C620x EMIF, but it can be configured to work in this interface. The C64x has two EMIFs, the 64-bit EMIFA and the 16-bit EMIFB. For this interface, the EMIFA is used in 32-bit mode and uses the same data pins as the C620x. This is accomplished by setting the MTYPE field of the CExCTL register to 0010b (32-bit-wide asynchronous interface).

This application report should serve as a guide to the use of other C6000 DSPs in this interface. By making the modifications suggested here, any of the C6000™ DSPs with an HPI as a peripheral should be able to use the S5933 to connect to a PCI bus.

5 References

1. *TMS320C6201 Digital Signal Processor* (SPRS051).
2. *TMS320C6000 Peripherals Reference Guide* (SPRU190).
3. *S5933 PCI Controller Data Book*, Applied Micro Circuits Corporation, 1997.
4. *TMS320C6201/6701 DSP Host Port Interface (HPI) Performance* (SPRA449).
5. *TMS320C66201/6701 Evaluation Module User's Guide* (SPRU269).

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third–party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265