AHCIPCIeSSD IP Core (APS-IP)

July 14, 2016 Product Specification Rev1.1

Design Gateway Co.,Ltd
54 BB Building 14th Fl., Room No.1402
Sukhumvit 21 Rd. (Asoke), Klongtoey-Nua,
Wattana, Bangkok 10110
Phone: (+66) 02-261-2277
Fax: (+66) 02-261-2290
E-mail: ip-sales@design-gateway.com
URL: www.design-gateway.com

Core Facts

Provided with Core	
Documentation	User Guide, Design Guide
Design File Formats	Encrypted hdl File
Instantiation Templates	VHDL
Reference Designs & Application Notes	QuartusII Project, See Reference Design Manual
Additional Items	Demo on ArriaV GX Starter board/ ArriaV SoC Development board/ Arria10 SoC Development board
Support	
Support Provided by Design Gateway Co., Ltd.	

Features

- Implement application layer to access AHCI PCIe SSD card
- Simple user control I/F and FIFO interface for data port
- Direct connect to Avalon-MM Hard IP for PCI Express from Altera by using 128-bit bus interface
- Small logic utilization and no need for CPU and external memory (DDR)
- Support three ATA commands, i.e. IDENTIFY DEVICE, WRITE DMA EXT, and READ DMA EXT
- Reference design with AB16-PCIeXOVR adapter board available on Arria V GX Starter board, Arria V SoC Development board, and Arria10 SoC Development board

Table 1: Example Implementation Statistics

Family	Example Device	Fmax (MHz)	Logic utilization (ALMs)	Registers[1]	Pin[2]	Block Memory bit	Design Tools
ArriaV GX	5AGXFB3H4F35C4	125	593	790	-	-	QuartusII 15.1
ArriaV ST	5ASTFD5K3F40I3	125	593	793	-	-	QuartusII 15.1
Arria10 SX	10AS066N3F40E2SGE2	250	628	923	-	-	QuartusII 16.0

Notes:

1) Actual logic resource dependent on percentage of unrelated logic

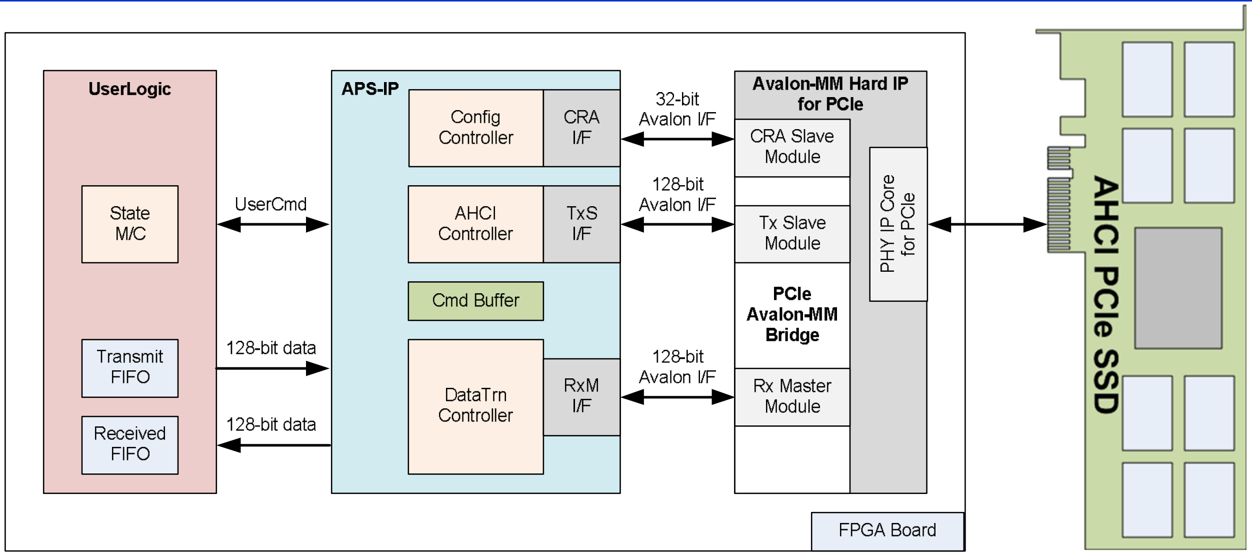

Figure 1: APS IP Block Diagram

Applications

APS IP Core operating with Avalon-MM Hard IP for PCIe from Altera is ideal to access AHCI PCIe SSD without CPU and external memory such as DDR3 requirement. It is recommended to use in the application which require high capacity storage at very high-speed performance. Small size system can be also designed by M.2 storage which uses PCIe protocol standard.

General Description

APS IP implements host controller to access PCIe SSD through AHCI standard interface. The IP is designed to connect with Avalon-MM PCIe Hard IP directly to transfer packet with PCIe SSD. Three ATA commands are supported for simple usage, i.e. IDENTIFY DEVICE to check disk capacity, WRITE DMA EXT to record data to SSD, and READ DMA EXT to read data back from SSD. By using pure hardware logic, it can reduce overhead time from bus interface and can achieve disk performance at very high-speed rate. The user interface is simple designed by setting only command, start address, and transfer length to the IP. While data interface can be connected to general FIFO directly. Since there is no asynchronous logic in the IP, clock domain of the IP must use the same clock with output from Avalon-MM PCIe Hard IP. Error signal will be asserted with the error status if IP detects the abnormal condition during packet transferring.

The reference design to test write/read data with AHCI PCIe SSD on ArriaV GX Starter board/ArriaV SoC Development board/Arria10 SoC Development board are available to download for customer evaluate the IP before purchasing.

Functional Description

APS IP Core generates packets and controls the packet sequence to record or read data with AHCI PCIe SSD. Three controllers and small interface modules are desigend to interface with each Avalon I/F of Avalon-MM PCIe Hard IP.

Configuration

After system power-on, PCIe root complex system needs to read and set configuration data with PCIe device following PCIe standard. The sequence to program configuration data to the device is designed within this block. Also, PCIe interrupt signal and status are monitored to confirm that device status still be in normal condition.

- **Config Controller**

  Two operation sequences are designed. First is intialization sequence to check PCIe device class, set BAR address, enable MSI interrupt, and enable master mode. Another is the seqeunce to monitor PCIe error and interrupt status. All sequence are controlled through register access.

- **CRA I/F**

  This module is used for creating TLP packets which contain write/read configuration data of Avalon-MM PCIe Hard IP, configuration data of PCIe SSD, and internal register of Avalon-MM PCIe Hard IP. The IP is designed in master mode and the TLP packet will be flushed to Avalon-MM PCIe Hard IP after completely constructed.

AHCI

This block is used to initialize the drive following AHCI standard. Through AHCI register, SATA LINKUP, SATA error, and SATA interrupt can be monitored. Also, the enable flag to issue command and the memory address to store command/status/data are programmed by this block.

- **AHCI Controller**

  This module is designed to initialize AHCI register only one time after system boot-up. When user sets the new command and parameters to the IP, this module will decode user inputs, prepare command FIS to Cmd buffer, and then set the flag to AHCI register to start device operation. After that, the module will monitor status flag to wait command complete, and check error status.

- **TxS I/F**

  AHCI register is mapped to BAR5 of PCIe device which can be accessed through 128-bit Avalon bus standard. Only 32-bit size with single access is used to access AHCI register through TxS port of Avalon-MM PCIe Hard IP. Similar to CRA I/F, this module is designed in master mode.

- **Cmd Buffer**

  The buffer to store command list and command table in AHCI standard.

Data

This block is designed to control data transfer between Transmit/Received FIFO within user logic and Avalon-MM PCIe Hard IP, and control command/status transfer between Cmd Buffer and Avalon-MM PCIe Hard IP.

- **DataTrn Controller**

  This module decodes the address value of the request to select the memory source/destination. There are three memory interfaces to connect with this module, i.e. Transmit/Received FIFO at user logic for the data, Cmd Buffer for command table and command list, and Identify device data.

- **RxM I/F**

  This is the slave side of 128-bit Avalon bus. This module cannot support 128-bit cross-boundary access and only single access is supported when bus size is not aligned with 128-bit. Burst length of each transaction is defined from the master side within Avalon-MM PCIe Hard IP.

User Logic

Simple logic to send command, address, and size can be designed. Transmit/Received FIFO size is flexible depending on the resource and performance requirement of user system.

Avalon-MM PCIe Hard IP

PCIe Hard IP to be interface module through Avalon-MM bus. More details are described in “ArriaV Avalon-MM Interface for PCIe Solutions” or “Arria10 Avalon-MM Interface for PCIe Solutions”.

Core I/O Signals

Descriptions of all signal I/O are provided in Table 2.

Table 2: Core I/O Signals

Signal	Dir	Description
User Interface		
RstB	In	Reset signal. Active low. Release this signal when Clk signal input is stable.
Clk	In	Clock output from Avalon-MM PCIe Hard IP to synchronous with Avalon bus interface. 125 MHz for PCIe Gen2 and 250 MHz for PCIe Gen3.
TestPin[31:0]	Out	Reserved to be IP Test point.
TimeOutSet[31:0]	In	Timeout value to wait completion from SSD. Time unit is Clk domain (8 ns for Gen2 or 4 ns for Gen3).
UserCmd[1:0]	In	User Command. “00”: Identify device command, “10”: Write PCIe SSD, ”11”: Read PCIe SSD.
UserAddr[47:0]	In	Start address of PCIe SSD to write/read in sector unit (512 byte)
UserLen[47:0]	In	Total transfer size in the request. Must not set equal to 0.
UserReq	In	Request the new command. Can be asserted only when the IP is Idle (UserBusy=’0’). Asserted with valid value on UserCmd/UserAddr/UserLen signals.
UserBusy	Out	IP Busy status. New request will not be allowed if this signal is asserted to ‘1’.
LBASize[47:0]	Out	Total capacity of PCIe SSD in sector unit (512 byte). Default value is 0. This value will be updated after user sets Identify device command.
UserError	Out	Error flag. Assert when UserErrorType is not equal to 0. The flag can be cleared by asserting RstB signal.
UserErrorType[31:0]	Out	Error status. [0] – Error when reset process within PCIe SSD is not completed before timeout. [1] – Error when the device does not support AHCI protocol. [2] – Error when PHY LINKUP is not detected from all 32 channels following AHCI standard. [3] – Error when Signature FIS is not ATA drive. [4] – Error when 512-byte Identify device data is not returned completely before timeout. [5] – Error when Portx Interrupt status register detects error conditon. See Portx Interrupt status value from PortIntStatus output signal. [6] – Error when MSI interrupt from PCIe SSD is not received before timeout. [7] – Error when CI register of PCIe SSD is not cleared to ‘0’ to confirm that command end before timeout. [31:8] – Reserved Note: Timeout period of bit[0]/[4]/[6]/[7] is set from TimeOutSet input.
PortIntStatus[31:0]	Out	The latest read value from Port Interrupt Status (0x10) of AHCI register map. Used to monitor when UserErrorType[5] is asserted to ‘1’. More details are described in “Serial ATA AHCI Specification”.

Signal	Dir	Description
FIFO Interface		
UserFifoWrRdy	In	Assert when received FIFO has at least 512-byte available space area.
UserFifoWrEn	Out	Write data valid of received FIFO.
UserFifoWrData[127:0]	Out	Write data bus of received FIFO. Synchronous to UserFifoWrEn.
UserFifoEmpty	In	Assert when transmit FIFO is empty.
UserFifoRdEn	Out	Read valid of transmit FIFO.
UserFifoRdData[127:0]	In	Read data returned from transmit FIFO. Valid after UserFifoRdEn asserted about one clock period.
Other Interface		
IdRamWrEn	Out	Valid signal of IdRamWrData.
IdRamWrData[127:0]	Out	512-byte stream data output from Identify Device command. Synchronous to IdRamWrEn.
PCIeLinkup	In	Interrupt output from Avalon-MM PCIe Hard IP.
CRA I/F		
CraChipSel	Out	Assert to select CRA port.
CraAddress[13:0]	Out	Write/Read byte address. Bit[1:0] is always equal to 00b for 32-bit access.
CraByteEnable[3:0]	Out	Byte enable.
CraRead	Out	Read enable. Indicate that valid read address is available.
CraReadData[31:0]	In	Read data.
CraWrite	Out	Write request. Inidicate that valid write address, write data, and byte enable are available.
CraWriteData[31:0]	Out	Write data.
CraWtRequest	In	Wait Request to hold off more requests.
TxS I/F		
TxSChipselect	Out	Assert to select TX Slave port.
TxSByteEnable[15:0]	Out	Byte enable for write data.
TxSReadData[127:0]	In	Read data return from PCIe Hard IP.
TxSWriteData[127:0]	Out	Write data to TX Slave port.
TxSRead	Out	Issue read request.
TxSWrite	Out	Issue write request.
TxSBurstCount[5:0]	Out	Indicate the amount of data requested. Always set to 000001b for single access.
TxSReadDataValid	In	Indicate that valid read data is available.
TxSWaitRequest	In	Hold off read or write data.
TxSAddress[28:0]	Out	Write/Read address.

Signal	Dir	Description
RxM I/F		
RxMAddress[31:0]	In	Write/Read address to access.
RxMBurstCount[7:0]	In	Gives the exact number of the burst count for write/read request.
RxMByteEnable[15:0]	In	Byte enable for write data.
RxMWrite	In	Write request sent from PCIe Hard IP.
RxMRead	In	Read request sent from PCIe Hard IP
RxMWaitRequest	Out	Assert to hold data transfer.
RxMWriteData[127:0]	In	Write data.
RxMReadData[127:0]	Out	Read data.
RxMReadDataValid	Out	Indicate that read data is valid.
RxMAddress[31:0]	In	Write/Read address. Gives the address of the first transfer in write/read burst transaction.

Note:

1) For Slave Avalon RxM interface, the IP cannot support burst transfer (AvBurstCount is not 1) which is not aligned to 128-bit.
2) For Slave Avalon RxM interface, the IP cannot support 128-bit cross-boundary, so AvAddress[3:0] must be equal to 0000b in case of burst transfer.

Timing Diagram

Clk and RstB input signal of the IP are generated from Avalon-MM PCIe Hard IP. After RstB is released, the IP will initialize PCIe configuration register and AHCI register of PCIe SSD. UserBusy is de-asserted to ‘0’ after both initialization sequences run completely as shown in Figure 2.

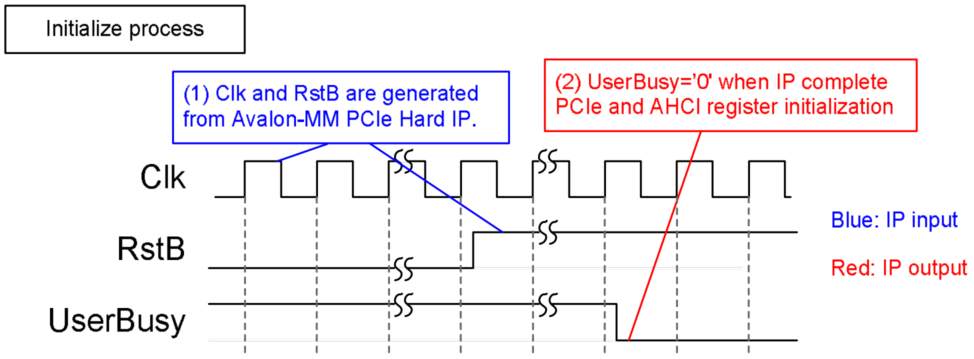
Initialize process
(1) Clk and RstB are generated from Avalon-MM PCIe Hard IP.
(2) UserBusy='0' when IP complete PCIe and AHCI register initialization
Clk
RstB
UserBusy
Blue: IP input
Red: IP output

Figure 2: RstB and UserBusy after system boot-up

Before sending new write/read command to IP, UserBusy must be always monitored to confirm that IP is Idle. UserCmd, UserAddr, and UserLen must be valid and latched during asserting UserReq=’1’, as shown in Figure 3. UserBusy is asserted to 1’ for acknowledge the IP that current command has been received and in processing. After that, UserReq can be cleared and user logic can prepare the next command to the command bus.

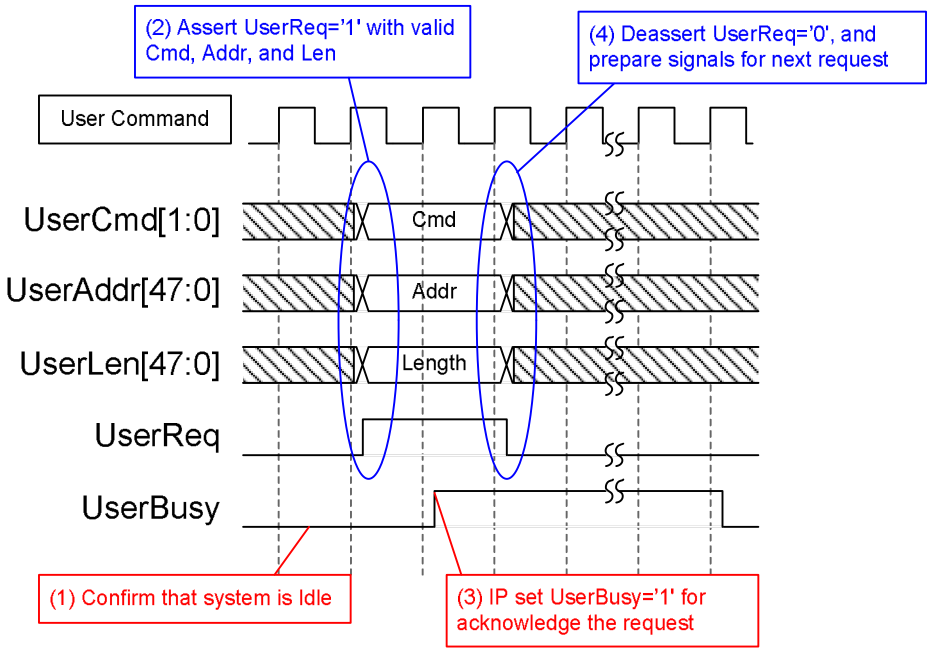
(2) Assert UserReq='1' with valid Cmd, Addr, and Len
(4) Deassert UserReq='0', and prepare signals for next request
User Command
UserCmd[1:0]
Cmd
UserAddr[47:0]
Addr
UserLen[47:0]
Length
UserReq
UserBusy
(1) Confirm that system is Idle
(3) IP set UserBusy='1' for acknowledge the request

Figure 3: User Command Interface Timing diagram

Before sending write or read command to IP, user should send Identify Device command firstly to update LBASize output. LBASize value is used in User Logic to confirm that the sum of address and length in write/read command is not oversize.

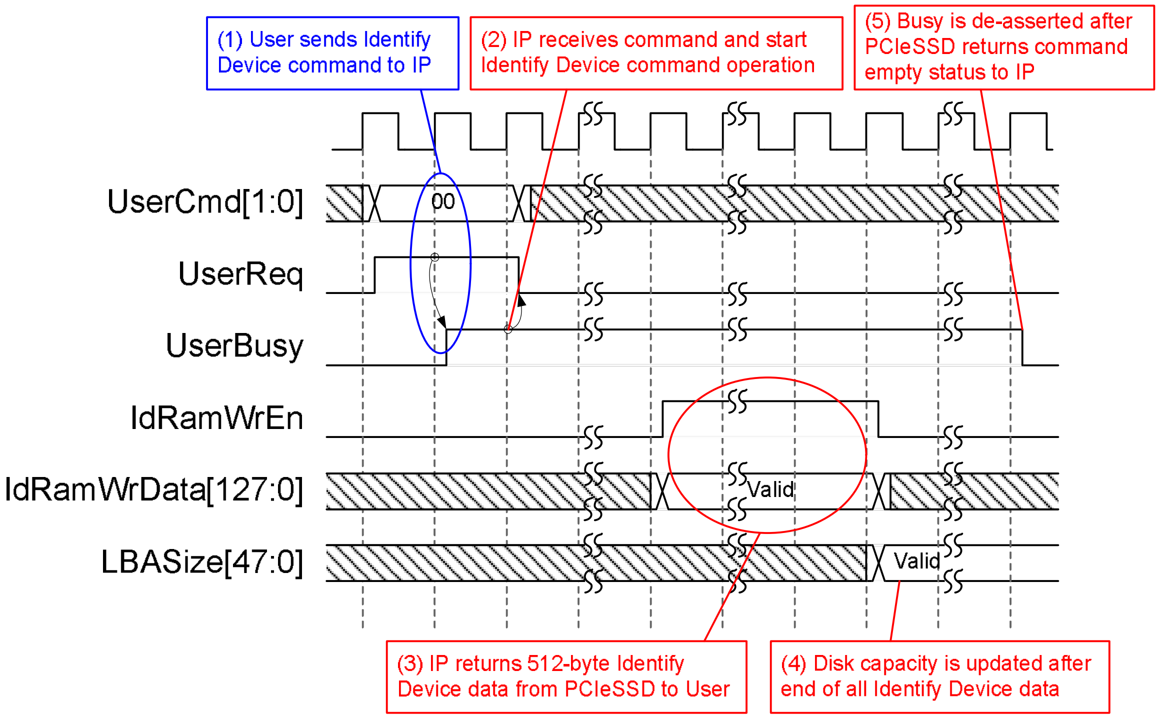
(1) User sends Identify Device command to IP
(2) IP receives command and start Identify Device command operation
(5) Busy is de-asserted after PCIeSSD returns command empty status to IP
UserCmd[1:0]
00
UserReq
UserBusy
IdRamWrEn
IdRamWrData[127:0]
Valid
LBASize[47:0]
Valid
(3) IP returns 512-byte Identify Device data from PCIeSSD to User
(4) Disk capacity is updated after end of all Identify Device data

Figure 4: LBASize update after Identify Device command

As shown in Figure 4, UserCmd and UserReq are set when UserBusy='0'. UserAddr and UserLen input are not required for Identify Device command. After that, 512-byte Identify Device data will be sent out through IdRam output and LBASize will be valid. UserBusy is de-asserted when PCIe SSD returns valid status to the IP at the end of command operation.

For write command, data from Transmit FIFO will be forwarded to Avalon-MM PCIe Hard IP through RxM interface. UserFifoRdEn will be asserted if the Transmit FIFO is not empty, monitored by UserFifoEmpty. Similar to typical FIFO, UserFifoRdData is valid after UserFifoRdEn is asserted about 1 clock period, as shown in Figure 5.

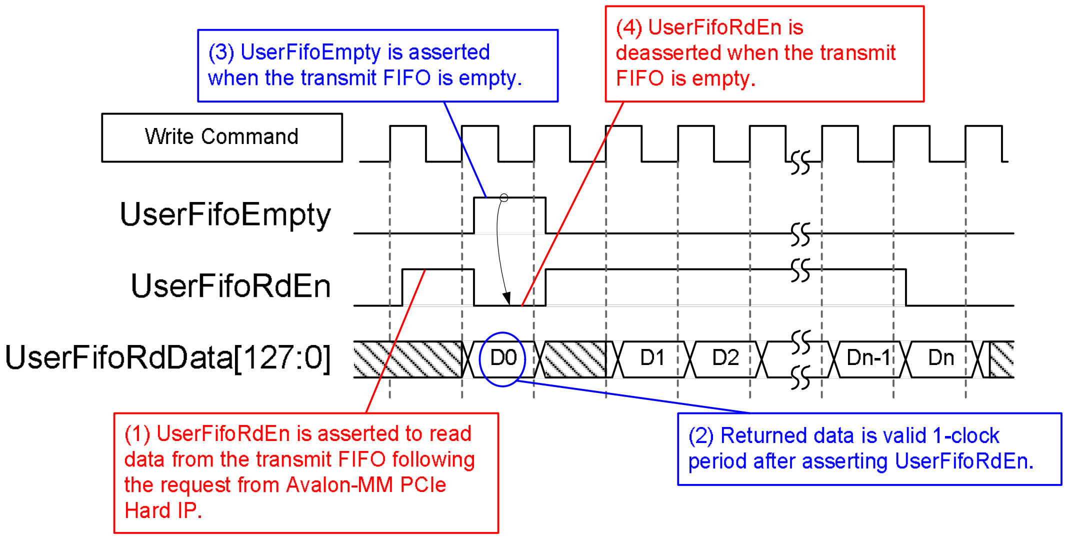
(3) UserFifoEmpty is asserted when the transmit FIFO is empty.
(4) UserFifoRdEn is deasserted when the transmit FIFO is empty.
Write Command
UserFifoEmpty
UserFifoRdEn
UserFifoRdData[127:0]
D0
D1
D2
Dn-1
Dn
(1) UserFifoRdEn is asserted to read data from the transmit FIFO following the request from Avalon-MM PCIe Hard IP.
(2) Returned data is valid 1-clock period after asserting UserFifoRdEn.

Figure 5: Transmit FIFO Interface for Write command

For read command, UserFifoWrEn will be asserted with the valid value of UserFifoWrData to store received data in Received FIFO until total numbers of data in that burst transfer equal to the request size from Avalon-MM PCIe Hard IP. Before asserting UserFifOWrEn, UserFifoWrRdy is monitored to check that at least 512-byte space area is available in Received FIFO. So, if request size is more than 512-byte, IP will split into many 512-byte burst transfers instead.

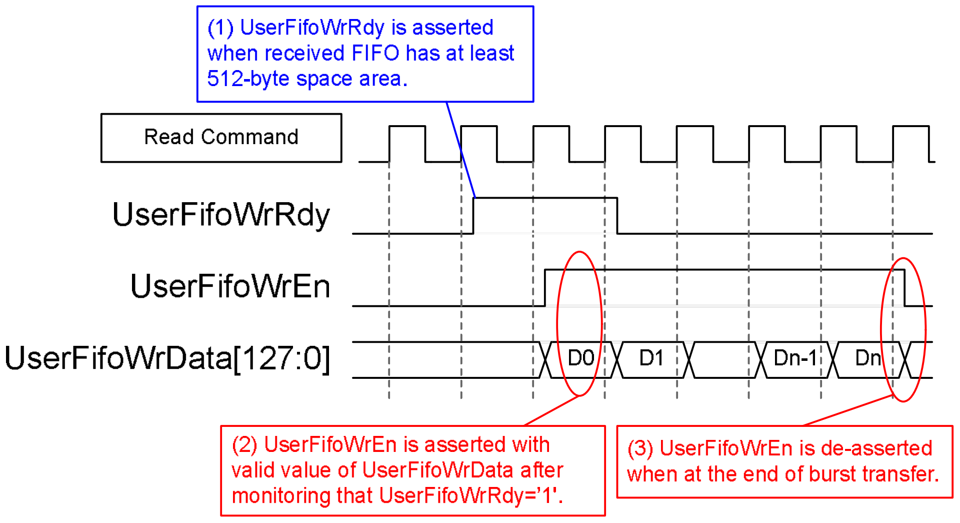
(1) UserFifoWrRdy is asserted when received FIFO has at least 512-byte space area.
Read Command
UserFifoWrRdy
UserFifoWrEn
UserFifoWrData[127:0]
D0
D1
Dn-1
Dn
(2) UserFifoWrEn is asserted with valid value of UserFifoWrData after monitoring that UserFifoWrRdy='1'.
(3) UserFifoWrEn is de-asserted when at the end of burst transfer.

Figure 6: Received FIFO Interface for Read command

During normal operation, UserError and all bits of UserErrorType signal will be always 0. UserError is generated by OR condition of each-bit of UserErrorType. If any bit of UserErrorType is set to ‘1’, UserError will be asserted and latched until RstB is asserted to ‘0’, as shown in Figure 7.

If PortIntStatus value has error condition, UserErrorType bit[5] will be set. So, user can see more details of the error cause by reading PortIntStatus.

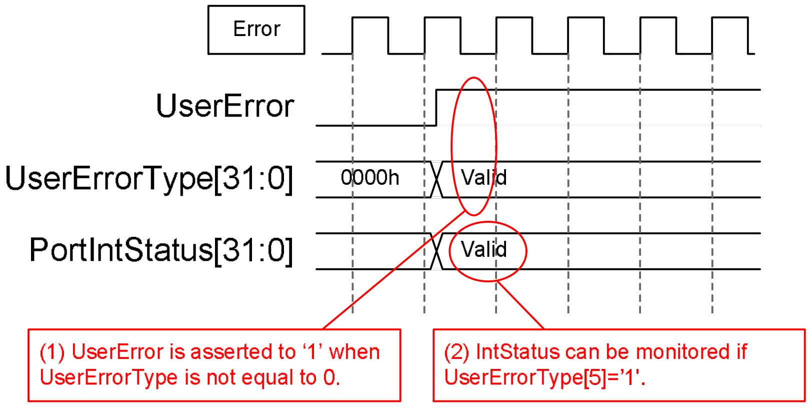
Error
UserError
UserErrorType[31:0]
0000h
Valid
PortIntStatus[31:0]
Valid
(1) UserError is asserted to ‘1’ when UserErrorType is not equal to 0.
(2) IntStatus can be monitored if UserErrorType[5]=’1’.

Figure 7: Error flag Timing diagram

Verification Methods

The APS IP Core functionality was verified by simulation and also proved on real board design by using ArriaV GX Starter board/ArriaV SoC Development board/Arria10 SoC Development board.

Recommended Design Experience

Experience design engineers with a knowledge of QuartusII Tools should easily integrate this IP into their design.

Ordering Information

This product is available directly from Design Gateway Co., Ltd. Please contact Design Gatway Co., Ltd. for pricing and additional information about this product using the contact information on the front page of this datasheet.

Revision History

Revision	Date	Description
1.0	Jun-26-2016	New release
1.1	Jul-14-2016	Add Gen3 support on Arria10 SoC