**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 | Reference Design Manual<br>Demo Instruction Manual |
| Design File Formats | Encrypted Netlist file |
| Constraints Files | Constraint file example in reference design project |
| Instantiation Templates | VHDL |
| Reference Designs & Application Notes | Vivado Project,<br>See Reference Design Manual |
| Additional Items | Demo on AC701[1], KC705, VC707, ZC706, KCU105 |
| **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 AXI Bridge for PCIe IP from Xilinx 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 AC701[1]/KC705/VC707/ZC706/KCU105 board

**Table 1: Example Implementation Statistics for 7-Series device (PCIe Gen2)**

| Family | Example Device | Fmax (MHz) | Slice Regs | Slice LUTs | Slices[2] | IOB | BRAMTile | PLL | GTX | Design Tools |
|---|---|---|---|---|---|---|---|---|---|---|
| Artix-7 | XC7A200TFFG1156-3E | 125 | 566 | 1317 | 493 | - | - | - | - | Vivado2014.4 |
| Kintex-7 | XC7K325TFFG900-2 | 125 | 568 | 1328 | 478 | - | - | - | - | Vivado2014.4 |
| Virtex-7 | XC7VX485TFFG1761-2 | 125 | 568 | 1329 | 503 | - | - | - | - | Vivado2014.4 |
| Zynq-7000 | XC7Z045FFG900-2 | 125 | 568 | 1359 | 498 | - | - | - | - | Vivado2014.4 |

**Table 2: Example Implementation Statistics for Ultrascale device (PCIe Gen3)**

| Family | Example Device | Fmax (MHz) | CLB Regs | CLB LUTs | CLB | IOB | BRAMTile | PLL | GTH | Design Tools |
|---|---|---|---|---|---|---|---|---|---|---|
| Kintex-Ultrascale | XCKU040FFVA1156-2E | 250 | 568 | 1459 | 272 | - | - | - | - | Vivado2015.4 |

Notes:

1) Despite operations confirmed on the real board, the reference design on AC701 has timing issue inside the Xilinx AXI-PCIe bridge IP. To meet timing requirement in the real implementation, -3 speed grade Artix-7 FPGA should be used.

2) Actual slice count dependent on percentage of unrelated logic

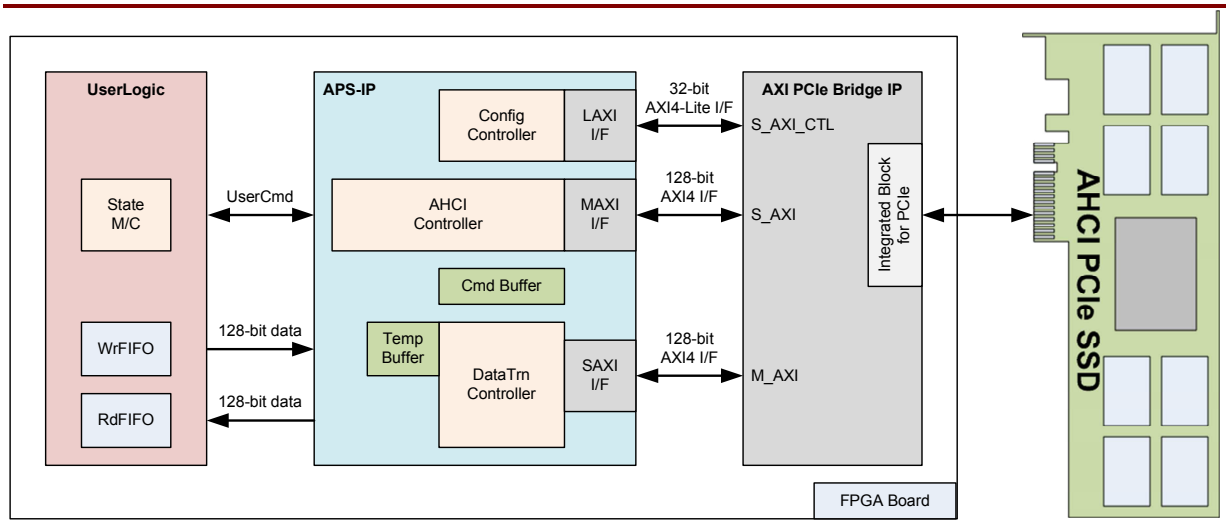


Figure 1: APS IP Block Diagram

## Applications

APS IP Core operating with AXI PCIe Bridge IP from Xilinx 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 AXI PCIe Bridge IP from Xilinx 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 connect 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 AXI PCIe Bridge 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 AC701/KC705/ZC706/VC707/ KCU105 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 designed to interface with each AXI bus of AXI PCIe Bridge 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 sequence to monitor PCIe error and interrupt status. All sequences are controlled through register access.

- **LAXI I/F**

  Configuration register is controlled through 32-bit AXI4-Lite standard. The IP is designed in master mode to write/read configuration data of AXI PCIe Bridge IP, configuration data of PCIe SSD, and internal register of AXI PCIe Bridge IP.

### 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, 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.

- **MAXI I/F**

  AHCI register is mapped to BAR5 of PCIe device which can be accessed through 128-bit AXI4 bus standard. Only 32-bit size with single access is used to access AHCI register through S_AXI port of AXI PCIe Bridge IP. Similar to LAXI 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 UserFIFO and AXI PCIe Bridge IP, and control command/status transfer between Cmd Buffer and AXI PCIe Bridge 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. UserFIFO for the data, Cmd Buffer for command table and command list, and Identify device data.

- **SAXI I/F**

  This is the slave side of 128-bit AXI4 bus. This module cannot support 128-bit cross-boundary access and only single access is supported when bus size is not 128-bit. Burst length of each transaction is defined from the master side within AXI PCIe Bridge IP.

- **Temp Buffer**

  Small RAM to store write data from user to control data flow control with SAXI4 I/F.

## User Logic

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

## AXI PCIe Bridge IP

Xilinx IP to be interface module between AXI4 bus and PCIe. More details are described in “PG055 LogiCore IP AXI Bridge for PCI Express” for PCIe Gen2 or “PG194 AXI Bridge for PCI Express Gen3” for PCIe Gen3.

## Core I/O Signals

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

**Table 3: 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 AXI PCIe Bridge IP to synchronous with AXI4-Lite and AXI4 bus interface.<br>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.<br>Time unit is Clk domain (8 ns for Gen2 or 4 ns for Gen3). |
| UserCmd[1:0] | In | User Command. “00”: Identify device command,<br>“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. Valid from 1 to (LBASize – UserAddr). |
| UserReq | In | Request the new command. Can be asserted only when the IP is Idle (UserBusy=’0’).<br>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.<br>This value will be updated after user sets Identify device command. |
| LinkSpeed[1:0] | Out | PCIe speed.<br>“00”: No linkup, “01”: Gen1 (2.5 Gbps), “10”: Gen2 (5.0 Gbps), “11”: Gen3 (8.0 Gbps). |
| UserError | Out | Error flag. Assert when UserErrorType is not equal to 0.<br>The flag can be cleared by asserting RstB signal. |
| UserErrorType[31:0] | Out | Error status.<br>[0] – Error when reset process within PCIe SSD is not completed until timeout.<br>[1] – Error when the device does not support AHCI protocol.<br>[2] – Error when PHY LINKUP is not detected from all 32 channels following AHCI standard.<br>[3] – Error when Signature FIS is not ATA drive.<br>[4] – Error when 512-byte Identify device data is not returned until timeout.<br>[5] – Error when Portx Interrupt status register detects error conditon.<br>See Portx Interrupt status value from PortIntStatus output signal.<br>[6] – Error when MSI interrupt from PCIe SSD is not received until timeout.<br>[7] – Error when CI register of PCIe SSD is not cleared to ‘0’ which is command end status until timeout.<br>[8] – Error when PCIe interrupt status detects error condition.<br>See Interrupt decode register value from PCIeIntStatus output signal.<br>[31:9] – Reserved<br>Note: Timeout period of bit[0]/[4]/[6]/[7] is set from TimeOutSet input. |
| PCIeIntStatus[31:0] | Out | The latest read value from Interrupt Decode Register (0x138) inside AXI PCIe Bridge IP. Used to monitor when UserErrorType[8] is asserted to ‘1’. More details are described in “PG055 LogiCore IP AXI Bridge for PCI Express/PG194 AXI Bridge for PCI Express Gen3”. |
| 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. |
| UserFifoRdRdy | In | Assert when transmit FIFO has at least 512-byte available data. |
| 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 and IdRamWrAddr. |
| IdRamWrAddr[4:0] | Out | Address of 512-byte Identify Device data in 128-bit unit. Synchronous to IdRamWrEn. |
| IdRamWrData[127:0] | Out | 512-byte stream data output from Identify Device command. Synchronous to IdRamWrEn. |
| PCIeInt | In | Interrupt output from AXI PCIe Bridge IP. |
| Master AXI4-Lite interface | | |
| LAxiwAwAddr[31:0] | Out | Write address. Gives the address of the first transfer in write burst transaction. |
| LAxiAwValid | Out | Write address valid. Indicates that valid write address and control signal are available. |
| LAxiAwReady | In | Write address ready. Indicates that the slave is ready to accept the write address and control signal. |
| LAxiwData[31:0] | Out | Write data. |
| LAxiwStrb[3:0] | Out | Write strobes. Indicates which bytes lanes hold valid data. |
| LAxiwValid | Out | Write valid. Indicates that valid write data and strobes are available. |
| LAxiwReady | In | Write ready. Indicates that the slave can accept the write data. |
| LAxiBValid | In | Write response valid. Indicates valid write response. |
| LAxiBReady | Out | Response ready. Indicates that the master can accept a write response. |
| LAxiArAddr[31:0] | Out | Read adress. Gives the address of the first transfer in a read burst transaction. |
| LAxiArValid | Out | Read address valid. Indicates that valid read address and control signal are available. |
| LAxiArReady | In | Read address ready. Indicates that the slave is ready to accept the read address and control signal. |
| LAxirData[31:0] | In | Read data. |
| LAxirValid | In | Read valid. Indicates that valid read data is available. |
| LAxirReady | Out | Read ready. Indicates that the master can accept the read data and response. |

| Signal | Dir | Description |
|---|---|---|
| Master AXI4 bus interface | | |
| MAxiAwId[3:0] | Out | Write address ID. Always set to 0000b. |
| MAxiAwAddr[31:0] | Out | Write address. Gives the address of the first transfer in write burst transaction. |
| MAxiAwRegion[3:0] | Out | Region identifier. Always set to 0000b. |
| MAxiAwLen[7:0] | Out | Burst length. Gives the exact number of transfers in a burst. Always set to 00h for single access. |
| MAxiAwSize[2:0] | Out | Burst size. Indicates the size of each transfer in the burst. Always set to 010b for 32-bit access. |
| MAxiAwBurst[1:0] | Out | Burst type. Determine how the address for each transfer within the burst is calculated. Always set to 01b (incrementing). |
| MAxiAwValid | Out | Write address valid. Indicates that valid write address and control signal are available. |
| MAxiAwReady | In | Write address ready. Indicates that the slave is ready to accept the write address and control signal. |
| MAxiwData[127:0] | Out | Write data. |
| MAxiwStrb[15:0] | Out | Write strobes. Indicates which bytes lanes hold valid data. |
| MAxiwLast | Out | Write last. Indicates the last transfer in a write burst. |
| MAxiwValid | Out | Write valid. Indicates that valid write data and strobes are available. |
| MAxiwReady | In | Write ready. Indicates that the slave can accept the write data. |
| MAxiBValid | In | Write response valid. Indicates valid write response. |
| MAxiBReady | Out | Response ready. Indicates that the master can accept a write response. |
| MAxiArId[3:0] | Out | Read address ID. Always set to 0000b. |
| MAxiArAddr[31:0] | Out | Read adress. Gives the address of the first transfer in a read burst transaction. |
| MAxiArRegion[3:0] | Out | Region identifier. Always set to 0000b. |
| MAxiArLen[7:0] | Out | Burst length. Gives the exact number of transfers in a burst. Always set to 00h for single access. |
| MAxiArSize[2:0] | Out | Burst size. Indicates the size of each transfer in the burst. Always set to 010b for 32-bit access. |
| MAxiArBurst[1:0] | Out | Burst type. Determine how the address for each transfer within the burst is calculated. Always set to 01b (incrementing). |
| MAxiArValid | Out | Read address valid. Indicates that valid read address and control signal are available. |
| MAxiArReady | In | Read address ready. Indicates that the slave is ready to accept the read address and control signal. |
| MAxirData[127:0] | In | Read data. |
| MAxirValid | In | Read valid. Indicates that valid read data is available. |
| MAxirReady | Out | Read ready. Indicates that the master can accept the read data and response. |

| Signal | Dir | Description |
|---|---|---|
| Slave AXI4 bus interface | | |
| SAxiAwAddr[31:0] | In | Write address. Gives the address of the first transfer in write burst transaction. |
| SAxiAwLen[7:0] | In | Burst length. Gives the exact number of transfers in a burst. |
| SAxiAwValid | In | Write address valid. Indicates that valid write address and control signal are available. |
| SAxiAwReady | Out | Write address ready. Indicates that the slave is ready to accept the write address and control signal. |
| SAxiwData[127:0] | In | Write data. |
| SAxiwStrb[15:0] | In | Write strobes. Indicates which bytes lanes hold valid data. |
| SAxiwLast | In | Write last. Indicates the last transfer in a write burst. |
| SAxiwValid | In | Write valid. Indicates that valid write data and strobes are available. |
| SAxiwReady | Out | Write ready. Indicates that the slave can accept the write data. |
| SAxiBResp[1:0] | Out | Write response. Indicates the status of the write transaction. Always set to 00b (OK). |
| SAxiBValid | Out | Write response valid. Indicates valid write response. |
| SAxiBReady | In | Response ready. Indicates that the master can accept a write response. |
| SAxiArAddr[31:0] | In | Read adress. Gives the address of the first transfer in a read burst transaction. |
| SAxiArLen[7:0] | In | Burst length. Gives the exact number of transfers in a burst. |
| SAxiArValid | In | Read address valid. Indicates that valid read address and control signal are available. |
| SAxiArReady | Out | Read address ready. Indicates that the slave is ready to accept the read address and control signal. |
| SAxirData[127:0] | Out | Read data. |
| SAxirResp[1:0] | Out | Read response. Indicates the status of the read transfer. Always set to 00b (OK). |
| SAxirLast | Out | Read last. Indicates the last transfer in a read burst. |
| SAxirValid | Out | Read valid. Indicates that valid read data is available. |
| SAxirReady | In | Read ready. Indicates that the master can accept the read data and response. |

Note:

1) For Slave AXI4 interface, the IP cannot support narrow transaction for burst transfer (AwLen/ArLen is not 0).
2) For Slave AXI4 interface, the IP cannot support 128-bit cross-boundary, so AwAddr/ArAddr[3:0] must be equal to 0000b in case of burst transfer.

## Timing Diagram

Clk signal input of the IP is generated from MMCM inside AXI PCIe Bridge IP, so RstB must be released after Clk is stable which can be monitored by MMCM_Lock output signal from AXI PCIe Bridge IP as shown in Figure 2.
After that, 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.

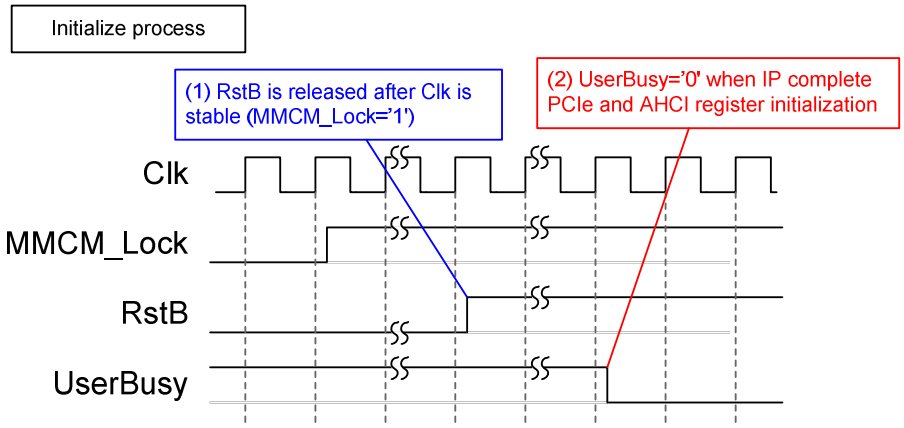


Figure 2: RstB and UserBusy after system boot-up

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.

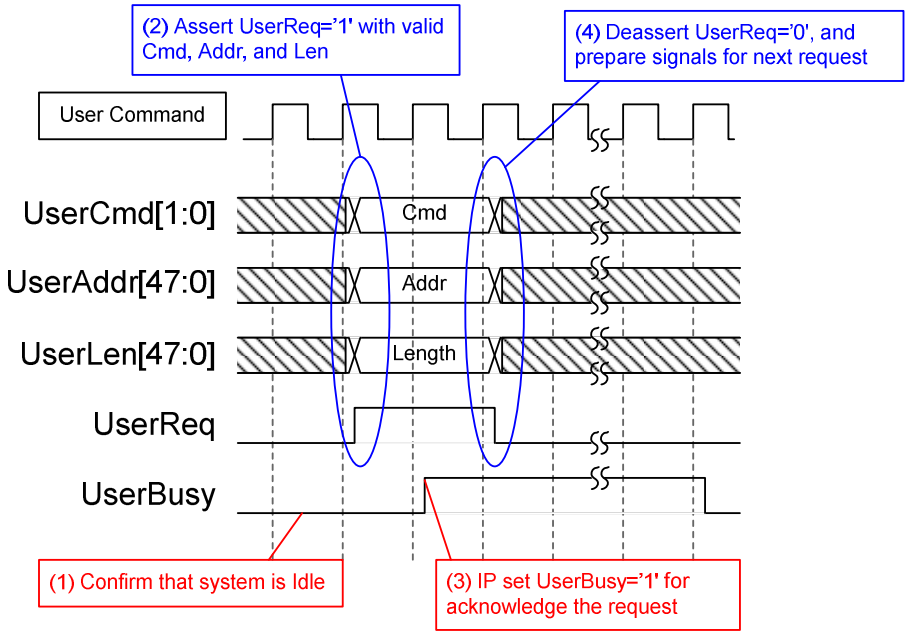


Figure 3: User Command Interface Timing diagram

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.

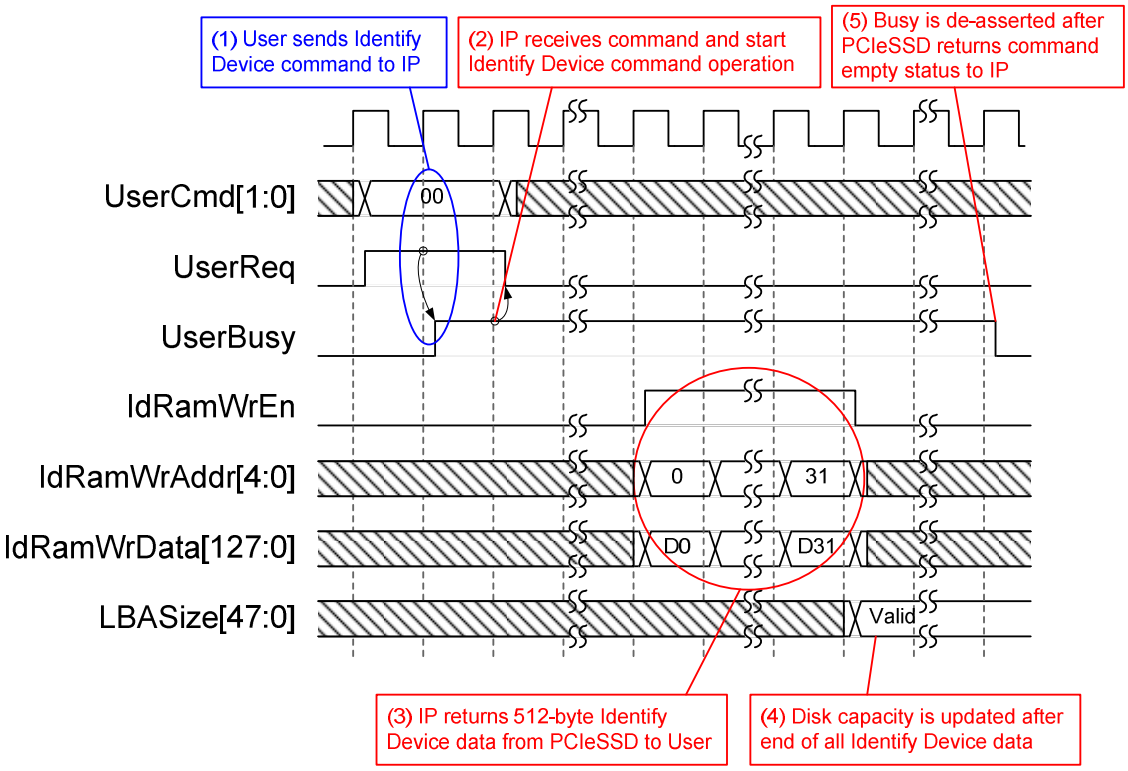


Figure 4: LBASize update after Identify Device command

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 AXI PCIe Bridge IP through Slave-AXI4 bus interface. If burst size from AXI PCIe Bridge IP is more than 512-byte, the IP will split into multiple 512-byte transactions instead. UserFifoRdRdy will be monitored to confirm that at least 512-byte is available in Transmit FIFO before starting burst transfer. Similar to typical FIFO, UserFifoRdData is valid after UserFifoRdEn is asserted about 1 clock period, as shown in Figure 5.

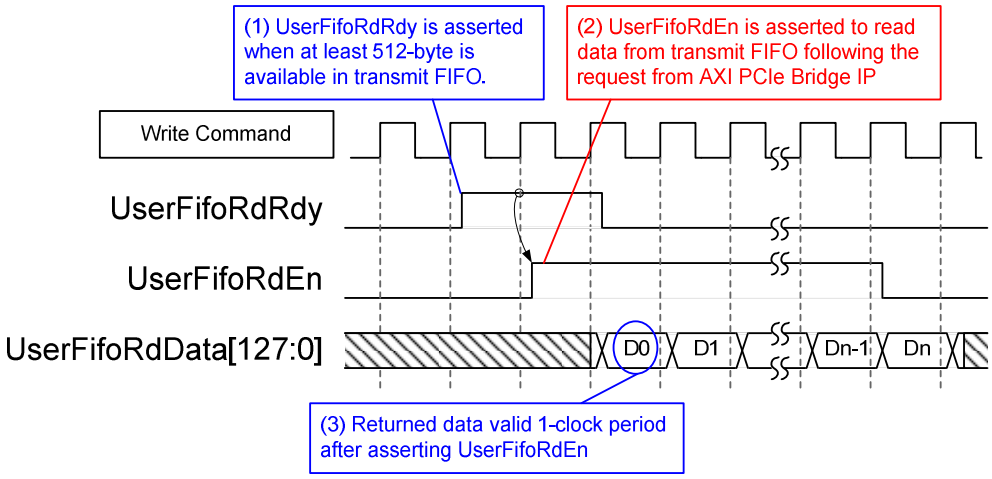


Figure 5: Transmit FIFO Interface for Write command

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 AXI PCIe Bridge IP. UserFifoWrRdy is monitored to check that at least 512-byte space area is available in Received FIFO. Similar to Write command, if burst request size is more than 512-byte, the IP will split into multiple 512-byte transactions instead.

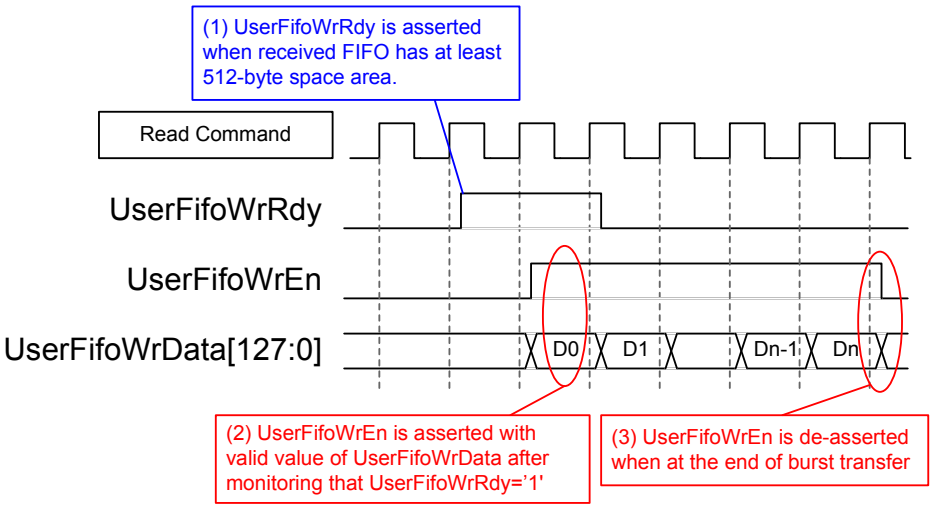


Figure 6: Received FIFO Interface for Read command

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 PCIeIntStatus or PortIntStatus value has error condition, UserErrorType bit[5]/[8] will be set. So, user can see more details of the error cause by reading PCIeIntStatus and PortIntStatus.

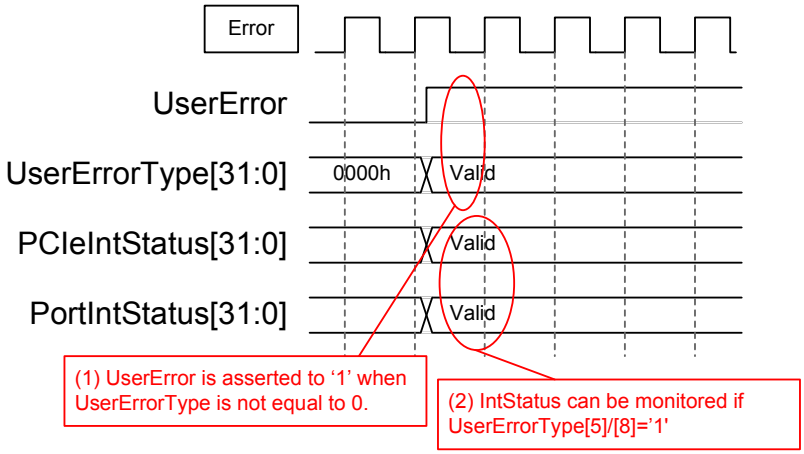


Figure 7: Error flag Timing diagram

## Verification Methods

The APS IP Core functionality was verified by simulation and also proved on real board design by using AC701/KC705/ZC706/VC707/KCU105 evaluation board.

## Recommended Design Experience

Experience design engineers with a knowledge of Vivado 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 |
|---|---|---|
| 0.1 | Aug-27-2015 | New release |
| 0.2 | Aug-31-2015 | Fixed some expression |
| 1.0 | Sep-11-2015 | 1st Release |
| 1.1 | Feb-17-2016 | Add KCU105 support and TimeOutSet input |
| 1.2 | Mar-1-2016 | Add LinkSpeed output |
| 1.3 | Oct-20-2016 | Add Artix-7 support |