SATA-IP RAIDx4 reference design on ML605 manual

Rev1.3 14-Dec-11

1. Introduction

Serial ATA (SATA) is an evolutionary replacement for the Parallel ATA (PATA) physical storage interface. SATA interface increases speed transfer to be 1.5 Gbps for SATA-I and 3.0 Gbps for SATA-II. To communication by SATA protocol, there are four layers in its architecture, i.e. Application, Transport, Link, and Phy.

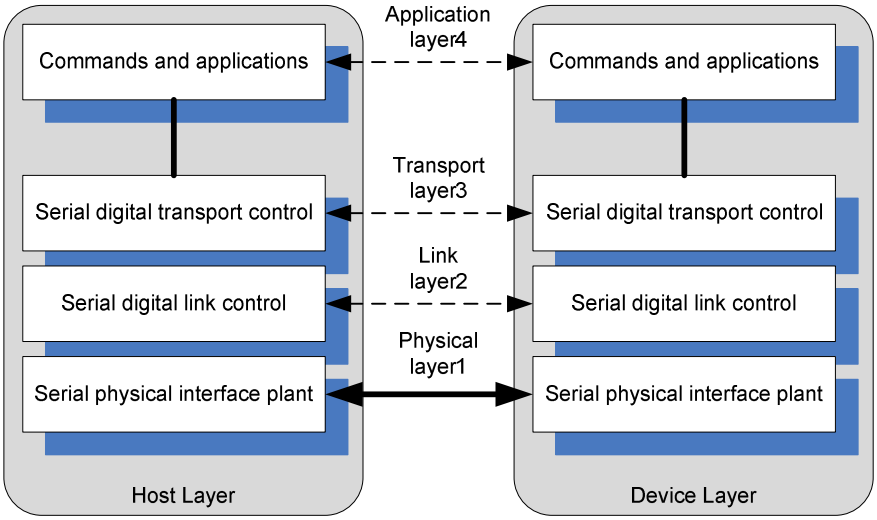

Figure 1 SATA Communication Layer

The Application layer is responsible for overall ATA command execution, including controlling Command Block Register accessed. The Transport layer is responsible for placing control information and data to be transferred between the host and device in a packet/frame, known as a Frame Information Structure (FIS). The Link layer is responsible for taking data from the constructed frames, encoding or decoding each byte using 8b/10b, and inserting control characters such that the 10-bit stream of data may be decoded correctly. The Physical layer is responsible for transmitting and receiving the encoded information as a serial data stream on the wire.

This reference design provides evaluation system which implements all SATA communication layers for Host side. The Application layer in this reference design connects to 4 SATA devices as RAID0 system to improve transfer speed about 4 times of each SATA device speed. In lower layer protocol, SATA-IP with GTX transceiver of the Virtex-6 platform is designed. ML605 is used to implement this RAID design and more details about the design are described as follows.

2. Environment

This reference design is based on the following environment as shown in Figure2.

- ML605 Platform
- ISE 12.3 / EDK 12.3
- FMC2SATA RAID adapter board to connect between FMC connector on ML605 board and SATA Device
  Note: FMC2SATA RAID adapter board can be requested from Design Gateway.
- 4 SATA Peripheral (SATA HDD/SSD) connecting with FMC2SATA RAID adapter board
- 2 USB mini cables for JTAG programming and serial communication. For serial communication, set baud rate=115,200 / data=8bit / Non-Parity / Stop=1bit.

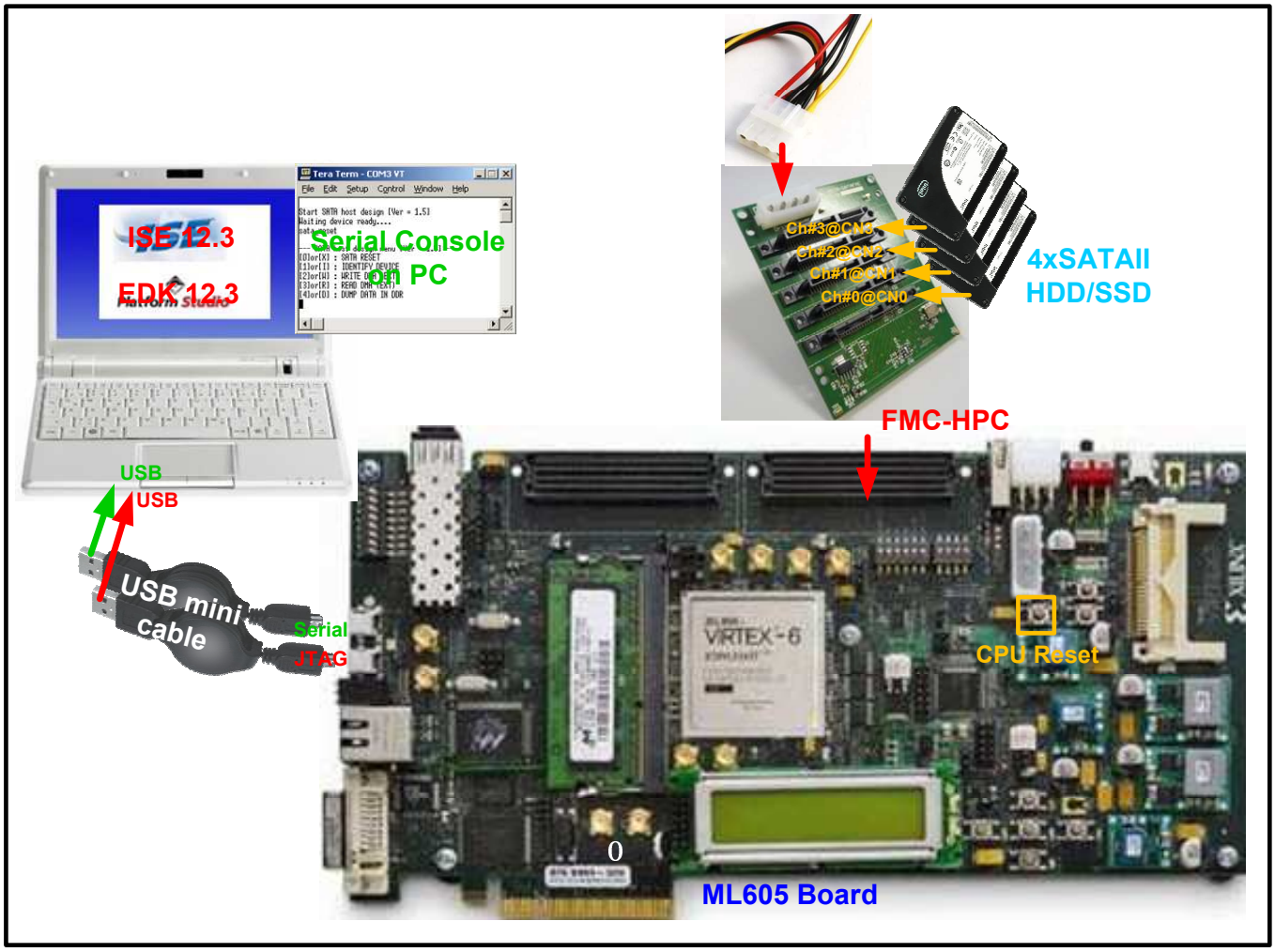

Figure 2 Reference design environment

Refer to “SATA-IP RAIDx4 Demo Instruction on ML605” for operation procedure of this reference design. For evaluation version, the system includes 1-hour limitation to use. After timeout, the system will stop any data transfer.

3. Hardware description

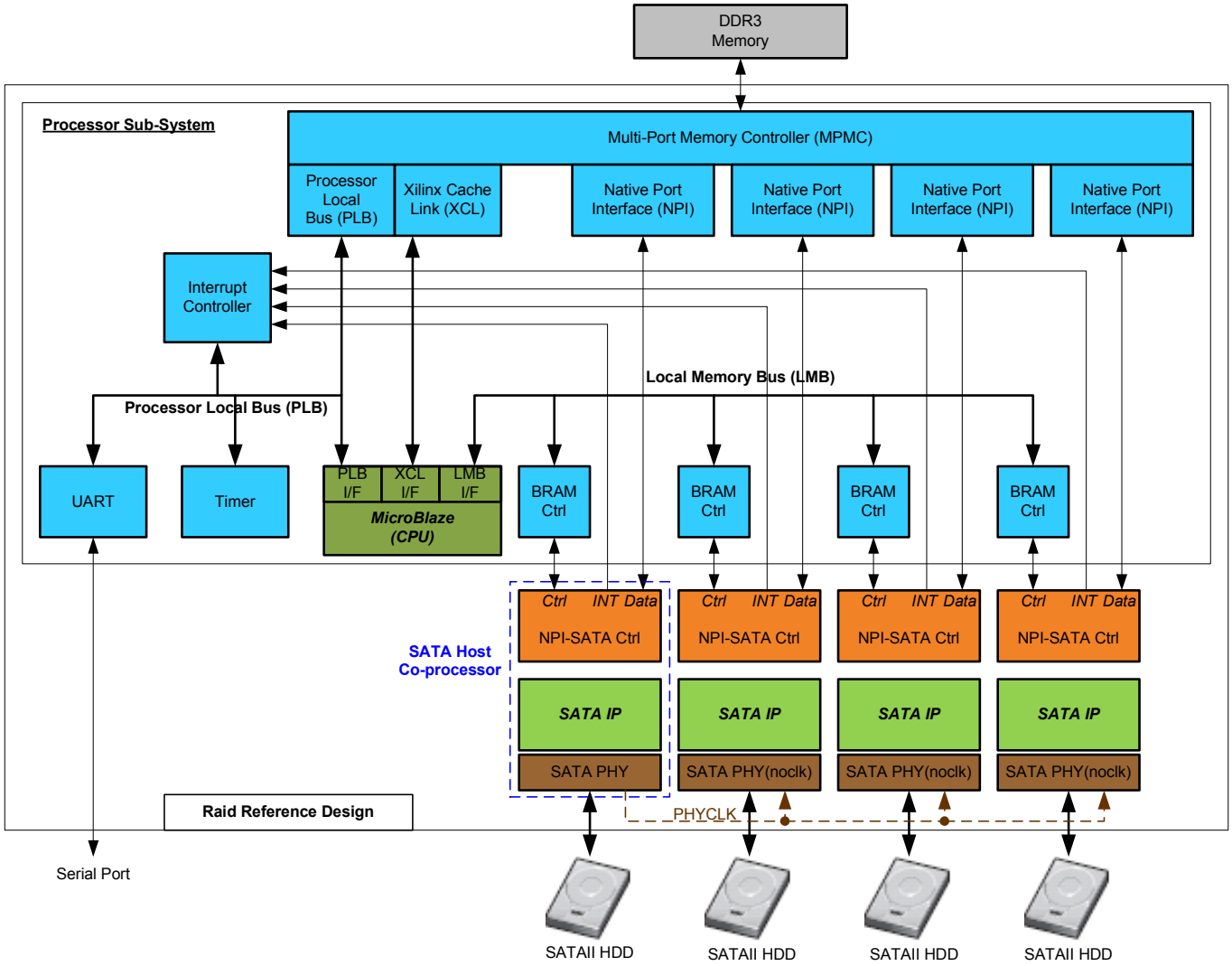

Figure 3 RAIDx4 Reference Design on ML605 Block Diagram

- SATA IP Host RAIDx4 design implementation on Virtex-6 FPGA

As shown in Figure 3, RAID reference design consists of three main modules, i.e. MicroBlaze, MPMC, and 4 SATA Host Co-processors. MicroBlaze creates and read data to DDR3 and then SATA Host Co-processor will read/write it from DDR3 and process to send to each SATA device. So, DDR3 needs to have the controller, called MPMC, to control the request from MicroBlaze and 4 SATA Host Controller which may be sent at the same time. Totally, there are six channels of MPMC to access Main Memory, i.e. 1 PLB for data access by MicroBlaze, 1 XCL for cache in MicroBlaze, and 4 NPI for each SATA channel.

BRAM Ctrl is used to be register access between MicroBlaze and SATA Host Co-processor to send control signals because of simple logic and small resource usage. By using Interrupt routine, Software on MicroBlaze can control both data and control signal for 4 SATA devices to run as RAID0 system with great performance. Serial Port is used to receive command from user.

Each SATA Host Co-processor consists of NPI-SATA Ctrl, SATA IP and SATA PHY which are designed to implement Transport Layer, Link Layer, and PHY Layer in SATA protocol. More details about each module are described as follows.

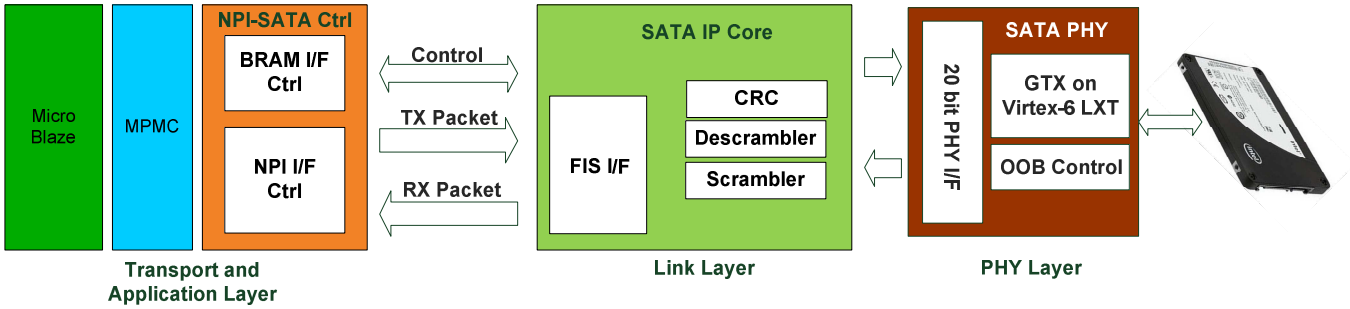

Figure 4 Connection between each Layer in SATA Protocol

- SATA PHY

SATA PHY module in this design is implemented by using built-in high-speed serial circuit of Virtex-6 LXT device (GTX transceiver) which is controlled by OOB (Out-of-Band) Control logic (oob_control.v). Input/Output with SATA PHY is arranged to be 20-bit format (16-bit data and 4-bit control bit). This PHY design can connect only SATA-II devices and speed negotiation or selection feature cannot support to reduce global clock resource in FPGA.

Before building user board, user must read carefully and must follow design guide line described in UG366 (Virtex-6 FPGA GTX Transceivers User Guide).

There are 2 SATA PHY types in this design to share clock resource within all 4 channels of GTX transceiver. Ch#0 which includes MMCM is stored in “pcores/npi_sata_v1_00_a” folder. Ch#1-Ch#3, which has no MMCM, are stored in “pcores/npi_sata_noclk_v1_00_a” folder. PHY source codes are stored in “hdl/vhdl/sata2phy_ml605.vhd” and “hdl/verilog/oob_control.v”. Both PHY source codes will be provided to user after purchasing order is received.

- SATA IP

SATA IP consists of only Link Layer and some part of Transport Layer in SATA protocol. Link Layer design includes CRC, Descramble and Scramble module. FIS I/F is designed to simplify user interface and support to run with user clock frequency. Interface signal of SATA IP core and description are shown in Table 1 and Table 2. These signals are displayed into two groups, i.e. transmit and receive. The waveforms to interface with SATA IP are shown in Figure5-Figure10.

- NPI-SATA Ctrl

This module is the simple example to interface with SATA IP for both data and control paths. The source code is stored in “pcores/npi_sata_(noclk_)v1_00_a/hdl/vhdl/npi_sata(_noclk).vhd”.

As shown in Figure 11, NPI-SATA Ctrl send/receive FIS data with MicroBlaze by using NPI Interface and receive control signal to start or stop by BRAM interface.

To write SATA device, MicroBlaze prepare FIS data in DDR3 and send control signal to start transfer. NPI Command Control will send command request to read data from NPI until total data is equal to set value. Data FIS Header Inserter will monitor NPI_RdFIFO_Empty and read data from NPI to send to SATA IP until no any more data from NPI. Otherwise, Data FIS Header Inserter also inserts FIS header automatically in case of DATA FIS type.

To read SATA device, MicroBlaze sends control signal to start this transfer and then SATA IP will start to send FIS data to user logic. DATA FIS header and Error Code words are removed from FIS data and stored to FIFO16x32 to be system data buffer. Burst size for NPI write request in this reference design is fixed to be 32-words for simple design. So, dummy data will be fed after real data complete if total data size of current FIS packet is not aligned in 32-words. NPI Command Control will send NPI request after each 32-words data are sent to MPMC. Read process will stop when no any more data is transferred from SATA IP.

Clock signals for BRAM interface and NPI interface in this module are different, so asynchronous registers are used to transfer control signals from one clock domain to another clock domain.

Signal	Dir	Clk	Description
Common Interface Signal			
trn_reset	In	trn_clk	Reset SATA IP core. Active high. Assert at least 4 clock period of core_clk for reset SATA-IP.
trn_link_up	Out	trn_clk	Transaction link up is asserted when the core establish the communication with SATA PHY.
trn_clk	In		Clock signal for interface with the Host. This clock frequency is required to be higher than core_clk frequency.
core_clk	In		IP Core operating frequency output (75.00MHz for SATA-II). This clock is generated from SATA PHY.
dev_host_n	In	trn_clk	Device or Host design assignment. '0': ATA Host IP Core, '1': ATA Device IP Core (Use '0' for the host reference design)
Transmit Transaction Interface			
trn_tsof_n	In	trn_clk	Not used now.
trn_teof_n	In	trn_clk	Transmit End-Of-Frame (EOF): Indicate end each SATA FIS packet. Active low.
trn_td[31:0]	In	trn_clk	Transmit Data: SATA FIS packet data to be transmitted.
trn_tsrc_rdy_n	In	trn_clk	Transmit Source Ready: Indicates that trn_td[31:0] from the Host is valid. Active low.
trn_tdst_rdy_n	Out	trn_clk	Transmit Destination Ready: Indicate that the core is ready to accept data on trn_td[31:0]. Active low. trn_tsrc_rdy_n must be de-asserted within 4 period of trn_clk after trn_tdst_rdy_n is de-asserted. So the core can accept 4 DWORD of trn_td[31:0] after trn_tdst_rdy_n is de-asserted.
trn_tsrc_dsc_n	In	trn_clk	Transmit Source Abort: Assert 1 clock period of trn_clk during operation (between tsof and teof) when the Host requires to cancel current write operation. Active low. After asserted, the Core will send SYNC primitive to SATA-PHY for abort the current transfer. The Host needs to wait until trn_tdst_rdy_n ready again before sending next packet. See Figure 8 for more details.
trn_tdst_dsc_n	Out	trn_clk	Transmit Destination Abort: Assert 1 clock period of trn_clk from the Core to cancel current write operation when SYNC primitive is received during data write operation. Active low. See Figure 10 for more details.

Table1 SATA IP interface signal definition

Signal	Dir	Clk	Description
Receive Transaction Interface			
trn_rsof_n	Out	trn_clk	Receive Start-Of-Frame (SOF): Indicate start each SATA FIS packet. Active low.
trn_reof_n	Out	trn_clk	Receive End-Of-Frame (EOF): Indicate end each SATA FIS packet. Active low.
trn_rd[31:0]	Out	trn_clk	Receive Data: SATA FIS packet data to be transmitted.
trn_rsrc_rdy_n	Out	trn_clk	Receive Source Ready: Indicates that trn_rd[31:0] from the core is valid. Active low.
trn_rdst_rdy_n	In	trn_clk	Receive Destination Ready: Indicate that the Host is ready to accept data on trn_rd[31:0]. Active low. trn_rsrc_rdy_n will be de-asserted within 4 period of trn_clk after trn_rdst_rdy_n is de-asserted. So Host should be supported to accept 4 DWORD of trn_rd[31:0] after trn_rdst_rdy_n is de-asserted.
trn_rsrc_dsc_n	Out	trn_clk	Receive Source Abort: Assert 1 clock period of trn_clk from the Core to cancel current read operation when SYNC primitive is received during data read operation. Active low. See Figure 11 for more details.
trn_rdst_dsc_n	In	trn_clk	Receive Destination Abort: Assert 1 clock period of trn_clk during read operation (between rsof and reof) when the Host requires to cancel current read operation. Active low. After asserted, the core will send SYNC primitive to SATA-PHY for abort the current transfer. The Host needs to wait until trn_tdst_rdy_n ready again before sending next packet. See Figure 9 for more details.
SATA PHY Interface			
PHYRESET	In	PHYCLK	Not used now.
PHYCLK	In		Reference Clock for 16-bit SATA PHY (150MHz for SATA-II) This clock is generated from GTX module and synchronous with 16-bit transmit data to SATA PHY.
LINKUP	In	PHYCLK	Indicates that SATA link communication is established. Active high.
PLLLOCK	In	PHYCLK	Indicates that PLL of SATA PHY is locked. Active high.
TXDATA[15:0]	Out	PHYCLK	16-bit transmit data from the core to the SATA PHY
TXDATAK[1:0]	Out	PHYCLK	2-bit Data/Control for the symbols of transmitted data. ("00": data byte, "01": control byte, "1X": undefined).
RXDATA[15:0]	In	PHYCLK	16-bit receive data from the SATA PHY to the core.
RXDATAK[1:0]	In	PHYCLK	2-bit Data/Control for the symbols of received data. ("00": data byte, "01": control byte, "1X": undefined)
RXDATAVALID	In	PHYCLK	Not used now

Table1 SATA IP interface signal definition (Cont'd)

Bit	Signal Name	Description
[31:27]	Reserved	Always zero
[26]	Dir	Current transfer direction flag. '0': From the Host to SATA IP, '1': From SATA IP to the Host
[25:24]	Error	Error code flag. "00": No error "01": Bad/Unknown SATA FIS packet. WTRM primitive is received during read operation or R_ERR primitive is received at the end of write operation. Please check data packet is correct format or not when this error detected. "10": CRC error. Please check SATA signal quality when this error detected. "11": Reserved
[23:8]	Reserved	Always zero
[7:0]	FIS Type	This byte indicates the header of error code packet. "0xEF" is defined to be different from other SATA FIS.

Table2 Error code description

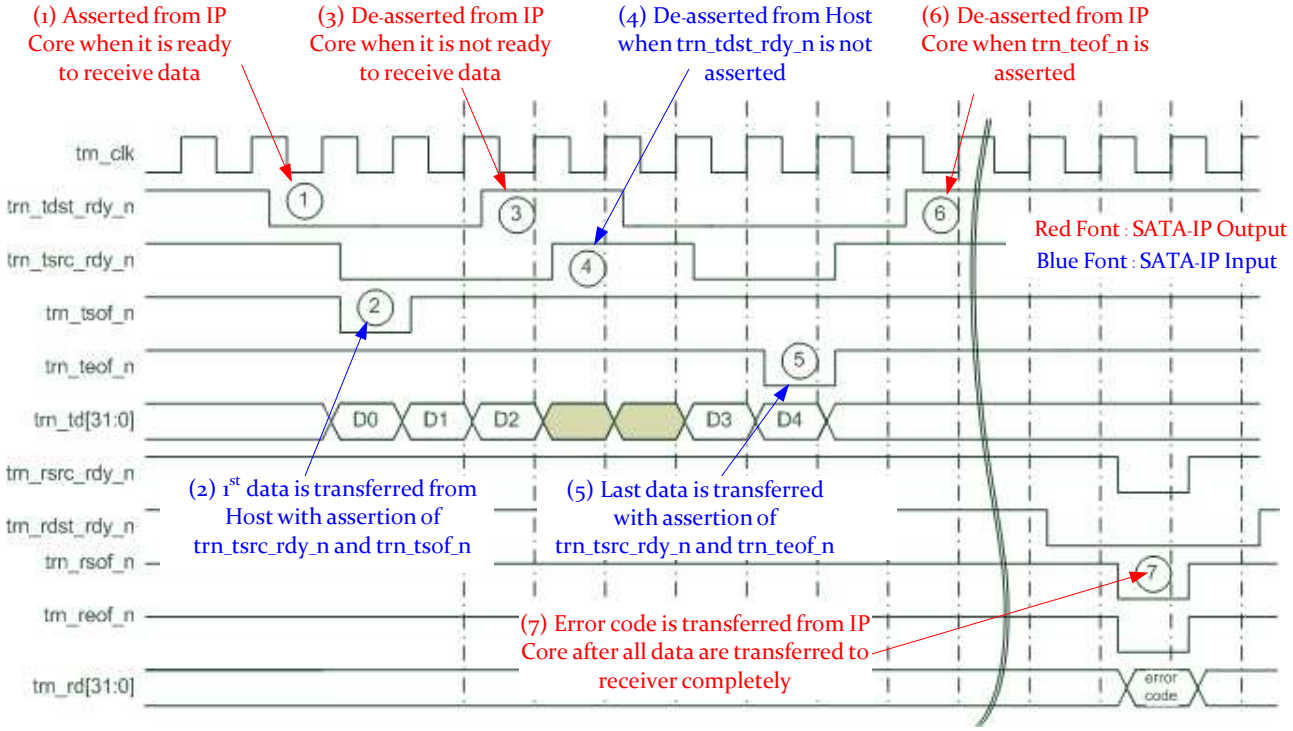

Figure 5 Waveform of data transmit transaction

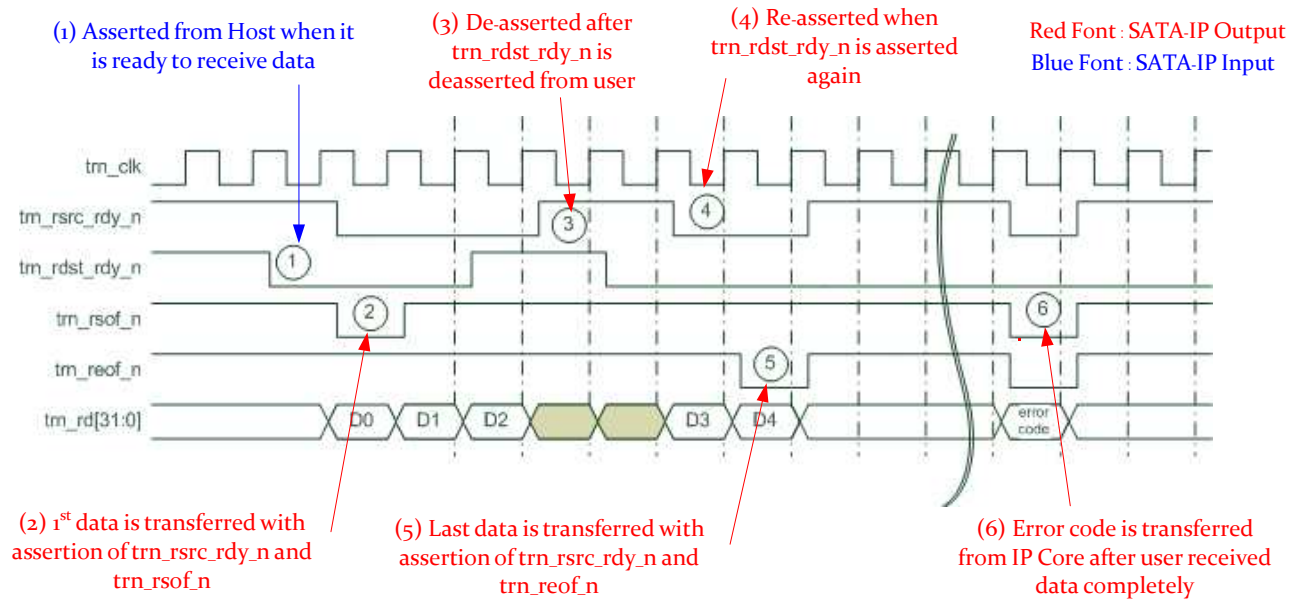

Figure 6 Waveform of data receive transaction

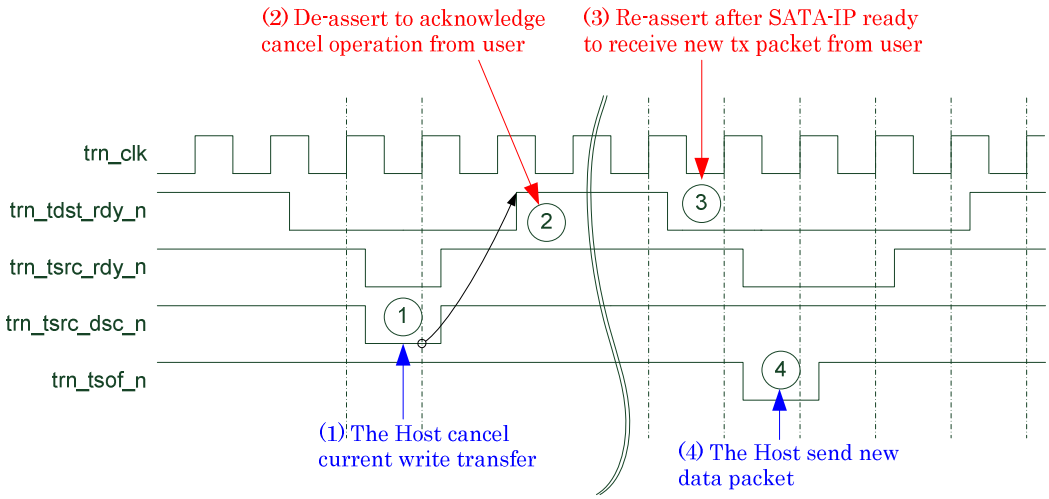

Figure 7 trn_tsrc_dsc_n Timing diagram

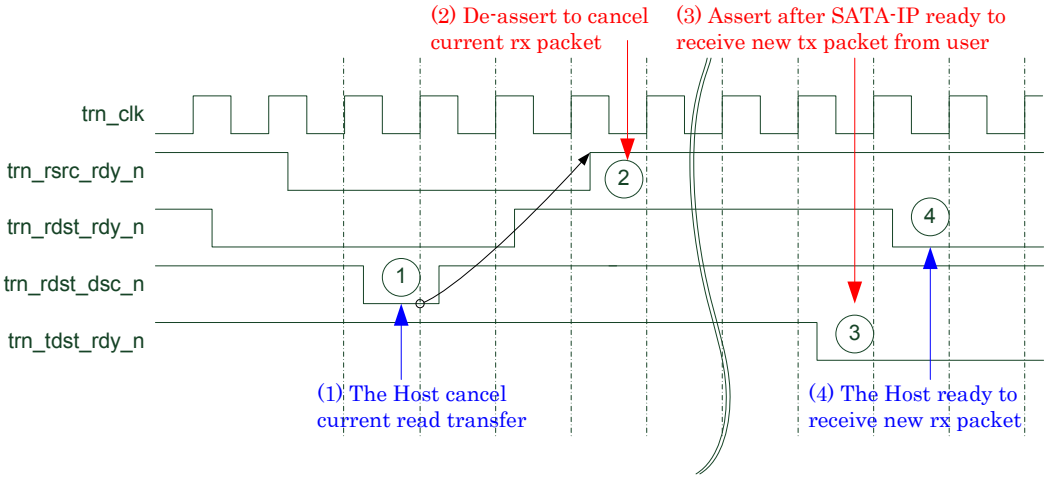

Figure 8 trn_rdst_dsc_n Timing diagram

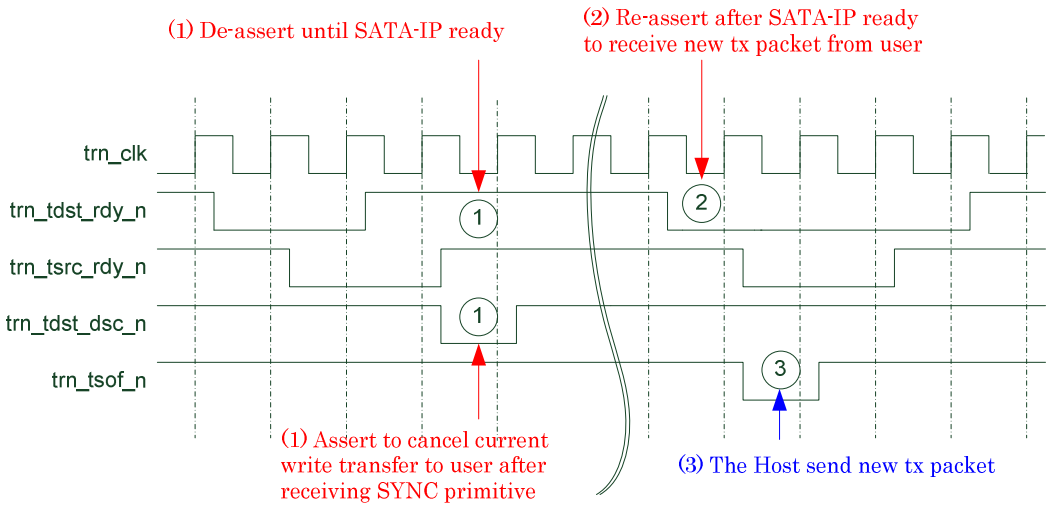

Figure 9 trn_tdst_dsc_n Timing diagram

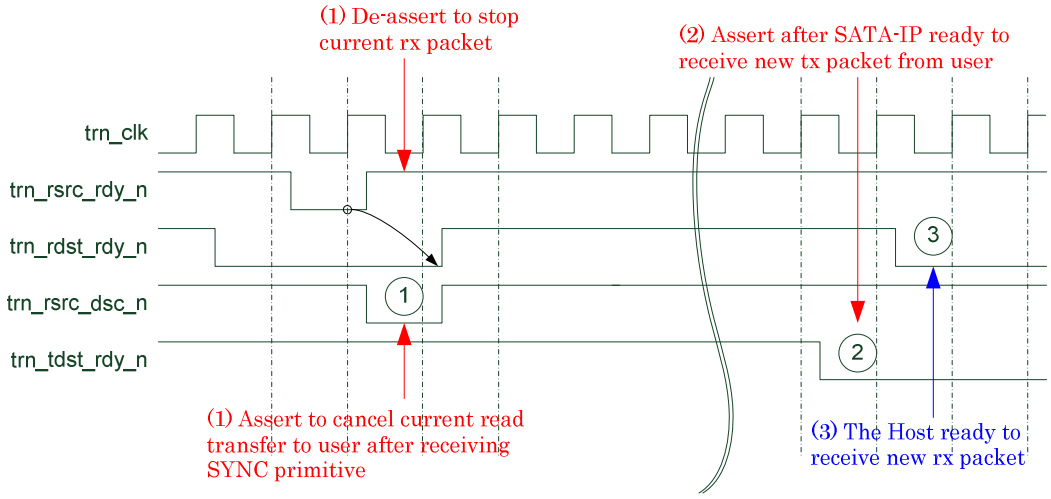

Figure 10 trn_rsrc_dsc_n Timing diagram

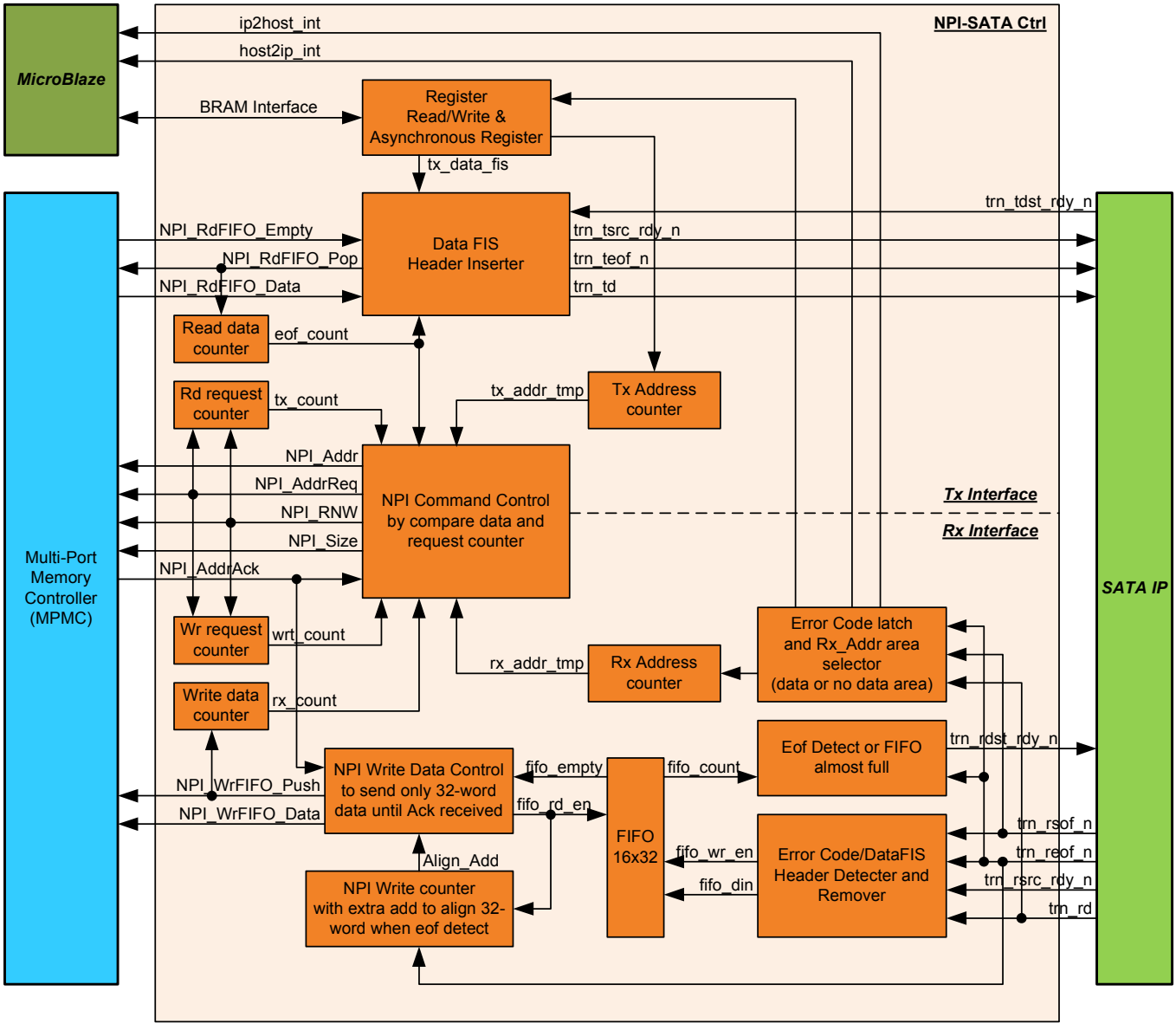

Figure 11 NPI-SATA Ctrl Block Diagram

- MicroBlaze

Software on MicroBlaze along with DMA mechanism will transmit FIS data on the main memory to the Link Layer of SATA IP, or will received data from the Link Layer to the main memory. During receiving data, Data FIS will be stored in different address space of Main Memory from other FIS types which is controlled by FIS header decoder in NPI-SATA Ctrl. For transmitting data, every FIS will use same address space, but Data FIS header will be added to packet or not depending on “Data FIS” flag value in Control Reg. More details about Register mapping is shown in Table3.

Address Rd/Wr	Register Name (Label in the “sata_host.c”)	Description (Bit order is little endian)
BA+0x00 Rd	Status Reg. (STATUS)	[8] : MPMC Ready [7:4] : OOB State Machine [3] : PLL in GTP lock [1] : Fixed to ‘1’ for SATA-II device [0] : SATA IP Link up
BA+0x04 Rd	Error Code Reg. (ERROR_CODE)	SATA IP Error code after transmit/receive completion to detect CRC or FIS error.
BA+0x0C Rd	Receive Word Count Reg. (RX_COUNT)	Total Received word count of FIS data. Auto clear when next transfer start (CONTROL Reg is written).
BA+0x00 Wr	Transmit Data Address Reg. (TX_ADDR)	Set DDR3 start address of transmit FIS data area
BA+0x04 Wr	Received Data Address1 Reg. (RX_ADDR)	Set DDR3 start address of received other FIS area (except DATA FIS type). Bit[7:0] of this value needs to be equal to 0.
BA+0x08 Wr	Control Reg. (CONTROL)	[31] : Hardware Reset [30] : Start Transmit data [29] : FIS type (‘1’: Data, ‘0’: Others) [15:0] : Total Transmit word count. RX_COUNT register is cleared by write operation to this register
BA+0x0C Wr	Received Data Address2 Reg. (RX2_ADDR)	Set DDR3 start address of received DATA FIS area
BA+0x10 Wr	Interrupt Clear Reg (INT_CLEAR)	[31] : Set this bit to clear ip2host interrupt [30] : Set this bit to clear host2ip interrupt

Table3 Register mapping from CPU side

Note: BA: Base Address will be different for each SATA device number.
- BA = 0x10000000 for device number#0
- BA = 0x20000000 for device number#1
- BA = 0x30000000 for device number#2
- BA = 0x40000000 for device number#3

4. Software description

- SATA Device access via FIS

Communication between the Host and the Device via SATA is done by FIS (Frame Information Structure) data structure. MicroBlaze in the Host design will build FIS data structure on its main memory space, and will send it to the Device by DMA controller that operates bus master. And FIS data sent from the Device is also transferred on the main memory by DMA controller.

Thus, MicroBlaze will execute access to the SATA Device by the following sequence.
(1) Build FIS Data structure (First FIS command should be RegH2D FIS)
(2) Transmit FIS Data
(3) Wait to receive FIS Data
(4) Read received FIS Data
(5) Additional FIS data transmit/receive if necessary.

FIS transmit and receive counts are different according to the protocol type, but the brief sequence should be as above.

- Software of reference design

Software of this reference design implements three popular commands, i.e. IDENTIFY DEVICE, READ DMA EXT/READ DMA, and WRITE DMA EXT/WRITE DMA. This reference design can support both 48-bit LBA (Logical Block Address) mode and 28-bit LBA mode to transfer data with SATA Device.
When SATA Device is powered on, it always sends Register – Device to Host FIS, so Host must wait this FIS from SATA Device before issue first command.

- IDENTIFY DEVICE

Table4 shows FIS structure of IDENTIFY COMMAND that gets device information from SATA Device. This command requires parameter settings for its Command Opcode (ECH) and device number that is typically set to zero in SATA device. Device register value will be A0H because obsolete bit#7 and bit#5 are recommended to set. “C” bit should be set whenever SATA Host sends command to the SATA Device, and it is also the same for all other commands issue.
After finish parameter settings to Register – Host to Device FIS, Host sends it to Link Layer. SATA Device will firstly send PIO Setup FIS, and then send Data FIS that includes device information.
For detailed device information, please refer to the ATA Standard document that can be obtained from http://www.t13.org/. This reference design only shows device model number, 48bit LBA support information, and disk capacity information.

0	Features 00h	command ECh	C R R R PM Port 1 0 0 0 0h	FIS Type (27h)
1	Device A0h	LBA High 00h	LBA Mid 00h	LBA Low 00h
2	Features (exp) 00h	LBA High (exp) 00h	LBA Mid (exp) 00h	LBA Low (exp) 00h
3	Control 00h	Reserved (0)	sector Count (exp) 00h	Sector Count 00h
4	Reserved (0)	Reserved (0)	Reserved (0)	Reserved (0)

Table4 IDENTIFY COMMAND FIS structure

- READ DMA EXT

Table 5 shows FIS structure of READ DMA EXT that reads data from SATA Device in 48-bit LBA mode. READ DMA command will be used to read data in 28-bit LBA mode. There are two data transfer types, i.e. PIO and DMA, but their difference is insignificant for SATA case. In SATA Device, speed performance of PIO transfer and DMA transfer are also not so different. Because DMA Read process is easier than PIO, this reference design selects DMA transfer.

Host will set Opcode to 25H for 48-bit mode or C8H for 28-bit mode, LBA bit (bit#6) of Device register, LBA address, and read sector count to the Register – Host to Device FIS, and then transmit to SATA Device. Device will send read data equal with read sector count setting in Data FIS, and then send Register – Device to Host FIS to finish this command.

0	Features 00h	command 25h	C R R R PM Port 1 0 0 0 0h	FIS Type (27h)
1	Device E0h	LBA High LBA[23:16]	LBA Mid LBA[15:8]	LBA Low LBA[7:0]
2	Features (exp) 00h	LBA High (exp) LBA[47:40]	LBA Mid (exp) LBA[39:32]	LBA Low (exp) LBA[31:24]
3	Control 00h	Reserved (0)	sector Count (exp) sector_count[15:8]	Sector Count sector_count[7:0]
4	Reserved (0)	Reserved (0)	Reserved (0)	Reserved (0)

Table5 DMA READ EXT FIS structure

- WRITE DMA EXT

Table 6 shows FIS structure of WRITE DMA EXT that writes data to SATA Device in 48-bit LBA mode. WRITE DMA command will be used to write data in 28-bit LBA mode. FIS structure is almost identical to that of READ DMA EXT. Host will set Opcode to 35H for 48-bit mode or CAH for 28-bit mode, LBA bit, LBA address, write sector count. After sending this Host to Device FIS, Device will send DMA Activate FIS to the Host. Then, Host sends first Data FIS to the Device.

Host will repeat sending Data FIS to the Device until all the data transfer is completed. Finally, Device sends Register- Device to Host FIS to finish this command.

0	Features 00h	command 35h	C R R R PM Port 1 0 0 0 0h	FIS Type (27h)
1	Device E0h	LBA High LBA[23:16]	LBA Mid LBA[15:8]	LBA Low LBA[7:0]
2	Features (exp) 00h	LBA High (exp) LBA[47:40]	LBA Mid (exp) LBA[39:32]	LBA Low (exp) LBA[31:24]
3	Control 00h	Reserved (0)	sector Count (exp) sector_count[15:8]	Sector Count sector_count[7:0]
4	Reserved (0)	Reserved (0)	Reserved (0)	Reserved (0)

Table6 DMA WRITE EXT FIS structure

- Necessary consideration

Host software source code of this design is stored in “sata_host.c”. Note that this reference design does not include error check or recovery from illegal/unexpected behavior. So user needs to add such consideration that software should check status or error check when Register – Device to Host FIS is received from the Device.

Figure12 shows reference design operation result on serial terminal screen.

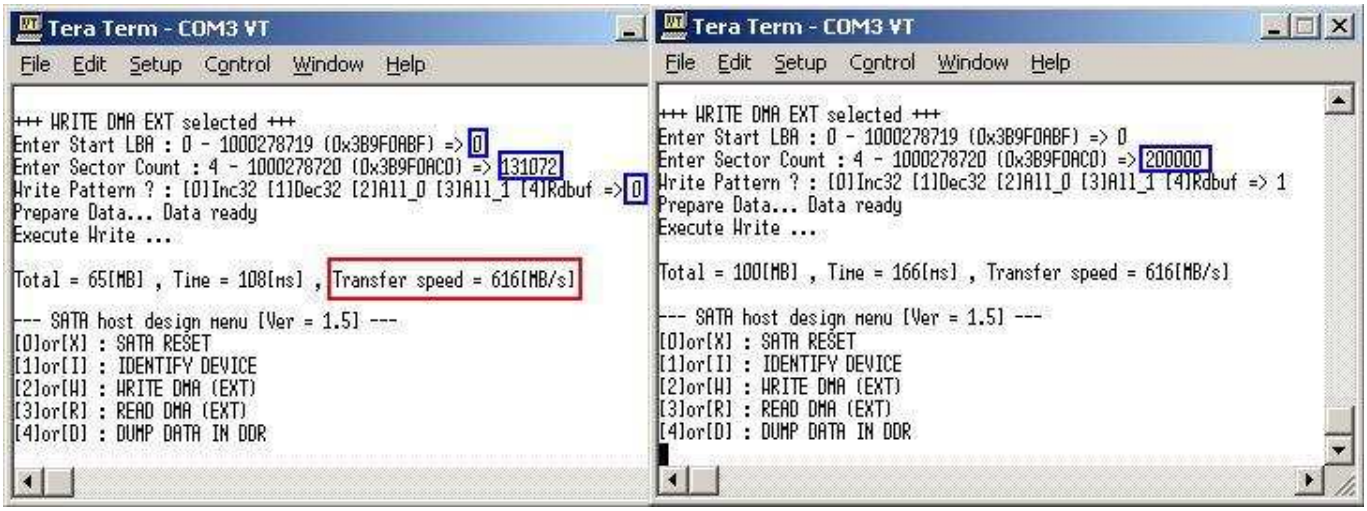

Figure 12 Operation result sample screen

5. Revision History

Revision	Date	Description
1.0	24-Feb-10	Initial release
1.1	26-Apr-10	Update ISE version, SATA PHY description, and Figure 11 and 12.
1.2	04-Nov-10	Update Signal name and description
1.3	14-Dec-11	Update for FMC2SATA RAID adapter board