Stratix4GX SATA3 Host reference design manual

Rev1.0 16-Jan-12

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, 3.0 Gbps for SATA-II, and 6.0 Gbps for SATA-III. To communication by SATA protocol, there are four layers in its architecture, i.e. Application, Transport, Link, and Phy.

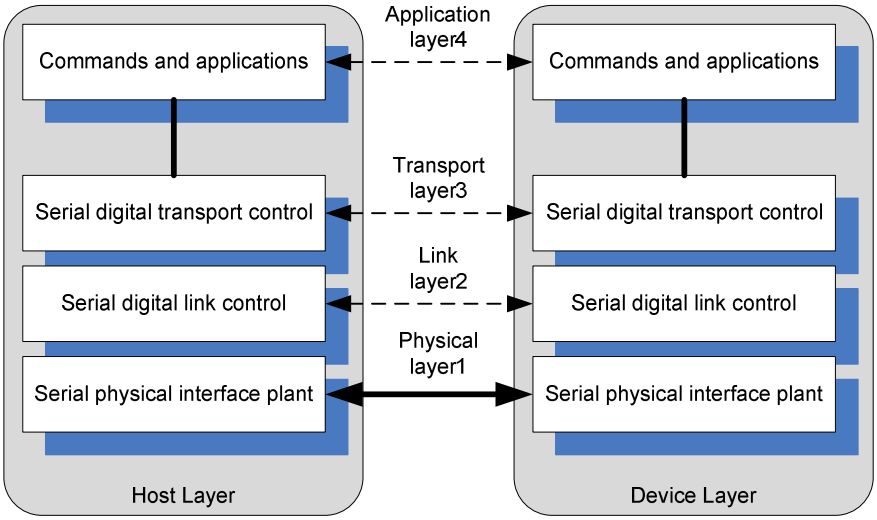

Figure 1 SATA Communitcation Layer

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 to transfer high speed data with SATA-II/SATA-III Device (HDD/SSD). SATA-IP core is designed to operate with GXB Transceiver (Gigabit Transceiver Block) on Stratix IV GX platform and be implemented on Stratix IV GX Development board. More details are described as follows.

2. Hardware description

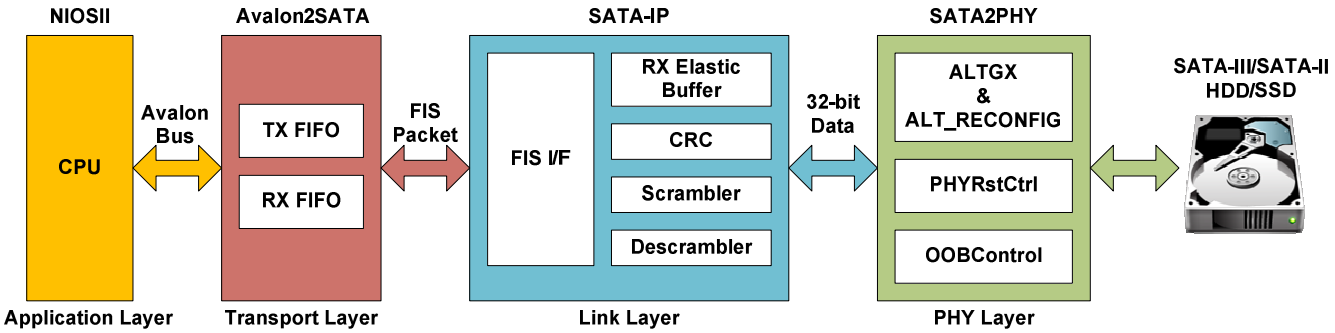

Figure 2 Hardware Architecture in reference design

As shown in Figure 2, hardware architecture can be divided into 4 modules to support each SATA layer protocol. SATA-IP implements Link Layer and some part of Transport Layer, so users need to prepare other Layer such as PHY Layer and Transport Layer by themselves. This reference design shows the example of Transport Layer and PHY Layer example based on Stratix IV GX development board.

- PHY Layer

This layer is designed by using built-in high speed serial component (GXB) and use ALTGX component in Megawizard to set basic parameter on GXB working with internal logic to control OOB (Out-of-Band) signal. State machine within OOBMainCtrl submodule is designed to control OOB sequence of SATA protocol and co-operate with OOBComCtrl submodule which is used to generate/receive COMRESET, COMINIT, and COMWAKE signal. PHYRstCtrl is applied to control reset sequence and select SATA speed interface by re-configuration GXB parameter through ALT_RECONFIG component. In this reference design, user can select speed through DIPSW by setting '1' to run with SATA-III or '0' to run with SATA-II speed.

PHY Layer is stored in "hdl/sata2phy_stratix4.vhd" that also includes "OOBComCtrl.vhd", "PhyRstCtrl.vhd", and "OOBMainCtrl.vhd". All hdl source codes on PHY Layer will be provided in delivery items for customer.

- Link Layer by SATA-IP

Interface signal of SATA IP core and description are shown in Table 1 and Table 2. Transport Layer interface signals are separated into two groups, i.e. transmit and receive. The waveforms to interface with SATA IP are shown in Figure 3 - Figure 8.

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 (150 MHz for SATA-III and 75 MHz for SATA-II).
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: Indicate 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.
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: Indicate 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.

Table 1 SATA-IP Interface Signal Description

Signal	Dir	Clk	Description
SATA PHY Interface			
LINKUP	In	core_clk	Indicate that SATA link communication is established. Active high.
PLLLOCK	In	core_clk	Indicate that PLL of SATA PHY is locked. Active high.
TXDATA[31:0]	Out	core_clk	32-bit transmit data from the core to SATA PHY
TXDATAK[3:0]	Out	core_clk	4-bit Data/Control for the symbols of transmitted data. ("0000": data byte, "0001": control byte, others: undefined).
RECCLK	In		Clock Recovery to synchronous with received data from SATA PHY
RXDATA[31:0]	In	RECCLK	32-bit receive data from the SATA PHY to the core.
RXDATAK[3:0]	In	RECCLK	4-bit Data/Control for the symbols of received data. ("0000": data byte, "0001": control byte, others: undefined)
RXDATAVALID	In	RECCLK	Indicate that RXDATA from SATA PHY is valid.
RXDATAOUT[31:0]	Out	core_clk	RXDATA signal after Elastic buffer and sychronous with core_clk
RXDATAKOUT[3:0]	Out	core_clk	RXDATAK signal after Elastic buffer and sychronous with core_clk
RXDATAVALIDOUT	Out	core_clk	Indicate that RXDATAOUT is valid.

Table1 SATA-IP Interface Signal Description (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. "10": CRC error "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.

Table 2 Error Code Description

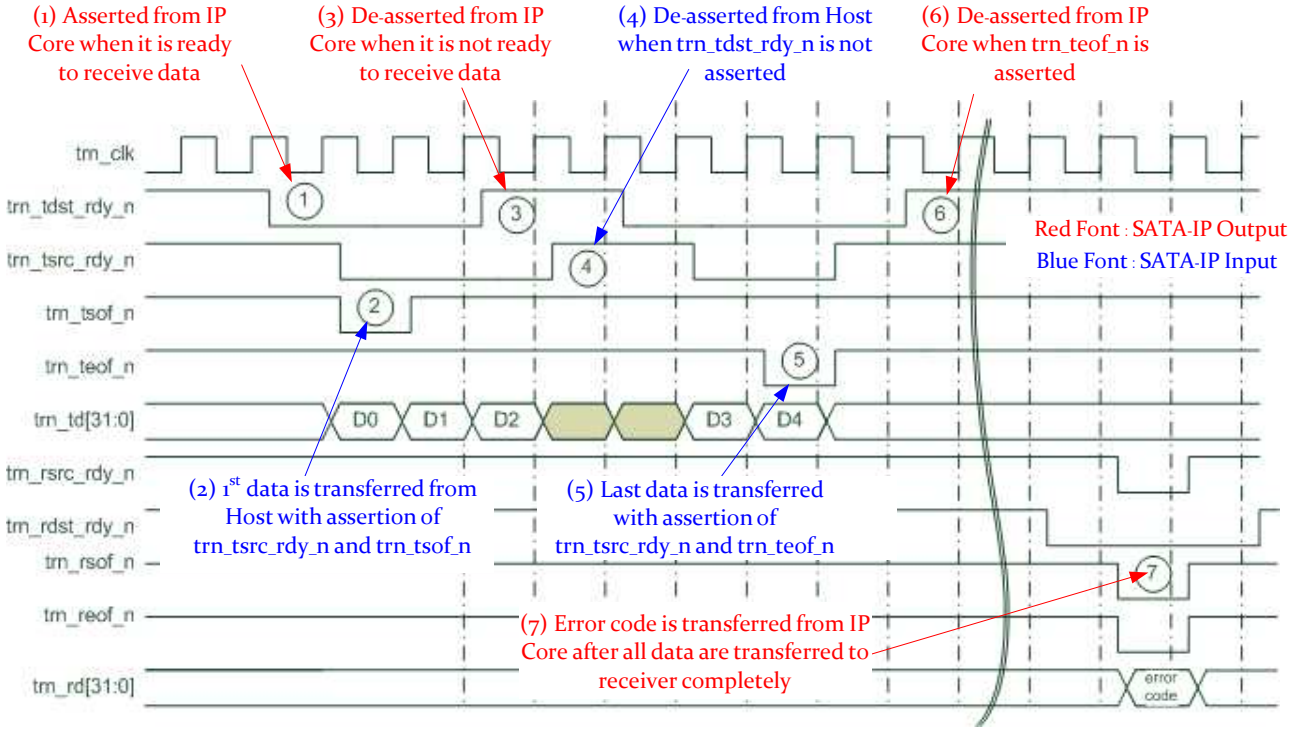

Figure 3 Waveform of data transmit transaction

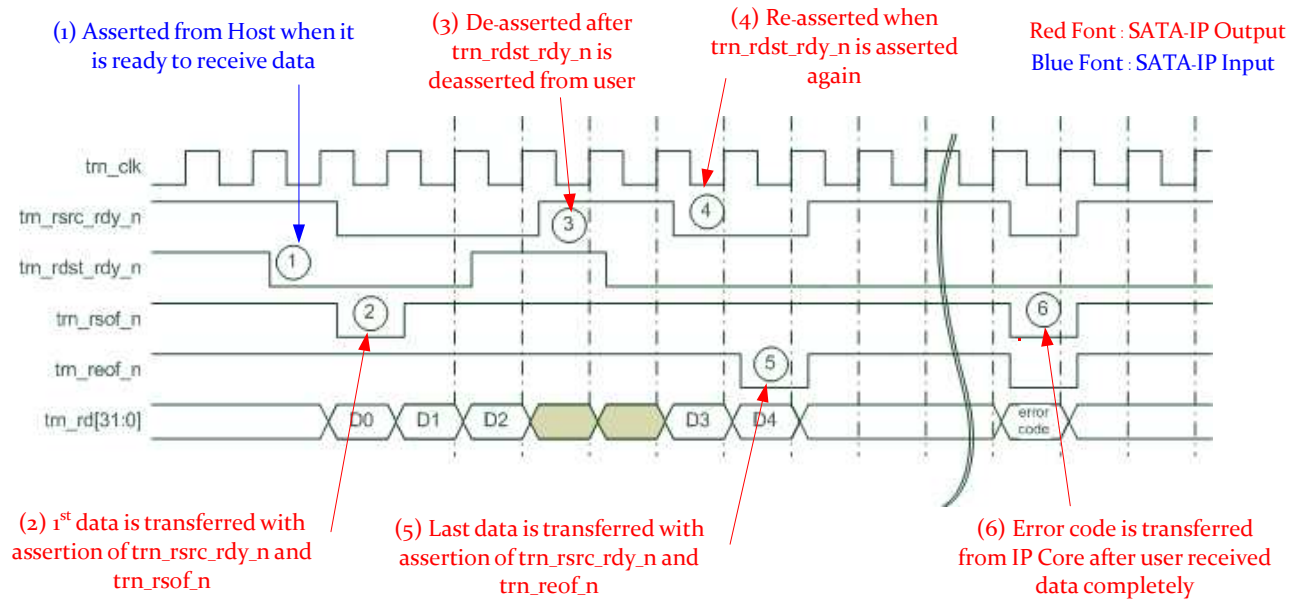

Figure 4 Waveform of data receive transaction

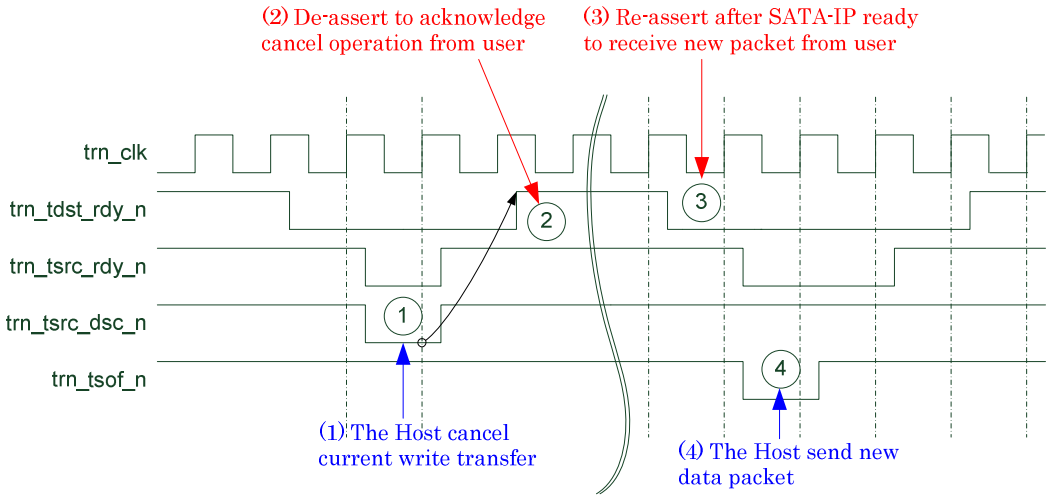

Figure 5 trn_tsrc_dsc_n Timing diagram

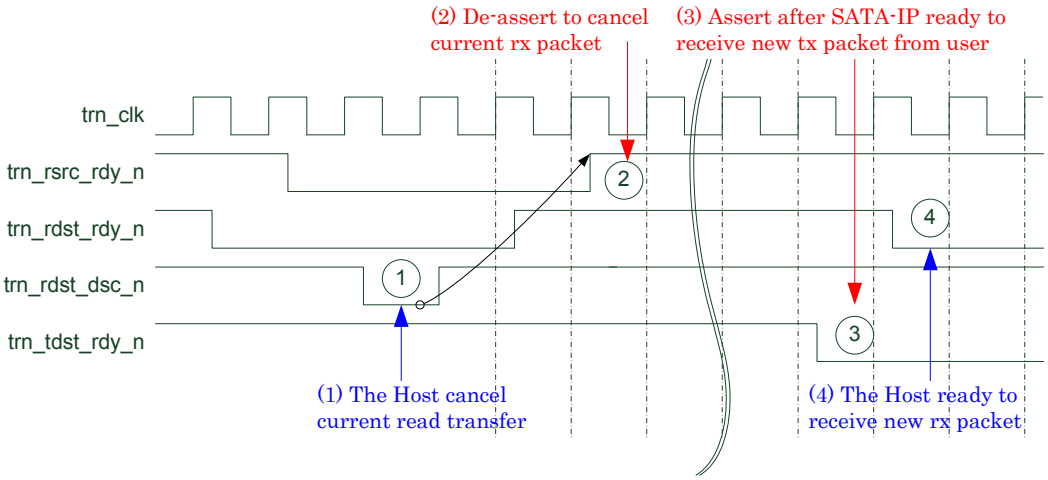

Figure 6 trn_rdst_dsc_n Timing diagram

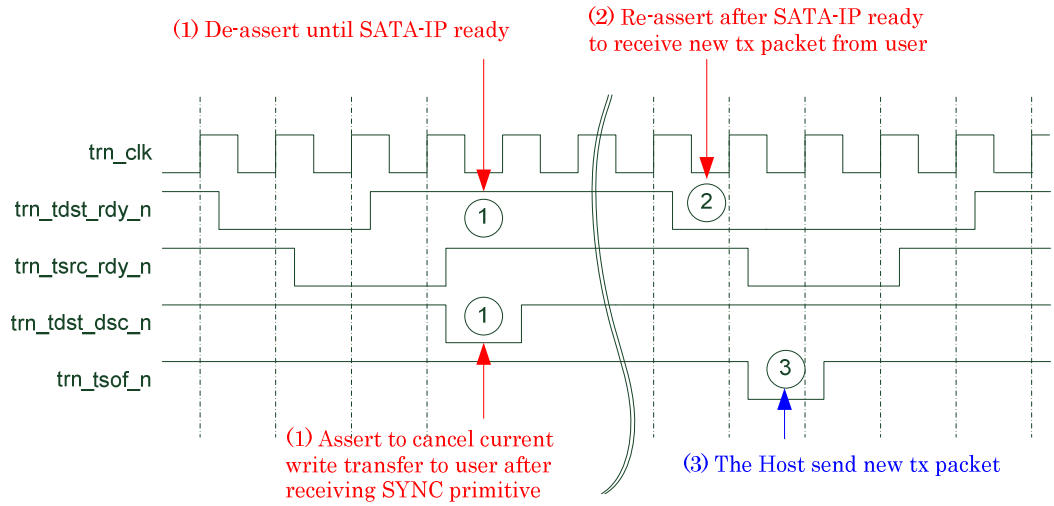

Figure 7 trn_tdst_dsc_n Timing diagram

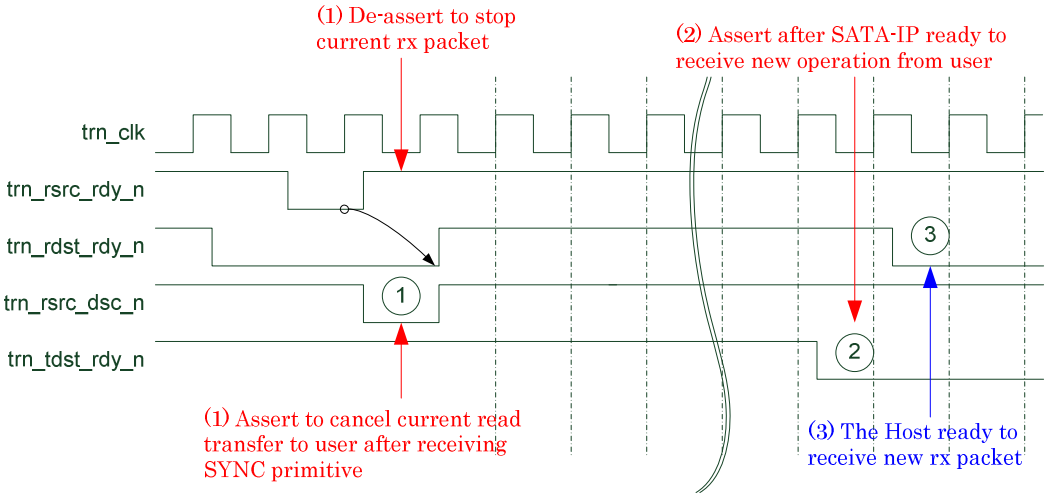

Figure 8 trn_rsrc_dsc_n Timing diagram

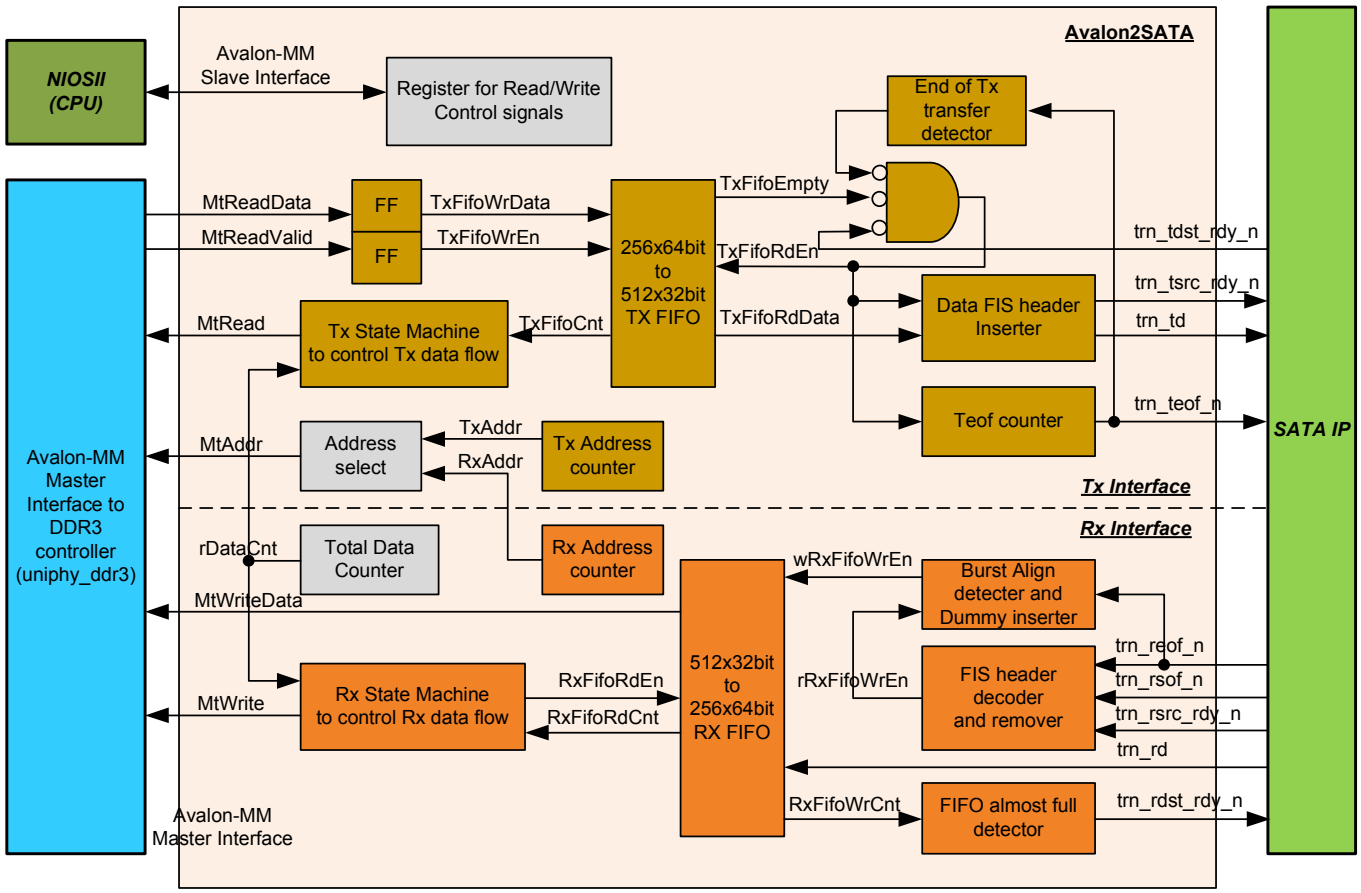

Figure 9 Avalon2SATA block diagram

- Transport Layer

Similar to typical SATA controller, CPU builds FIS data on Main memory (DDR3) and send control signal to lower layer to start DMA mechanism with Main memory. To support DMA data transfer, Avalon-MM Master interface is designed within Avalon2SATA module, and control signal with CPU is designed by Avalon-MM Slave interface. For high performance transfer, Avalon-MM Master interface is designed to fixed 32-burst size with 64-bit interface. So, transfer address need to align with 256-byte unit.

To write FIS packet to SATA-IP, each 32-burst data will be request and store in TXFIFO until Total Data Counter decrement to zero value (all data are transferred). If current transfer is Data FIS packet, Data FIS header will be auto-generated to SATA-IP. If total data size is aligned with 32-burst size (256-byte unit), all data in TXFIFO will be transferred to SATA-IP. For size which is not aligned, only request size will be transferred to SATA-IP. Dummy data from CPU still remain in TXFIFO and will be flush by reset signal.

To read FIS packet from SATA-IP, FIS header will be decoded and will be auto-removed for data FIS header, so only pure data will be stored to RX FIFO. If total data from SATA-IP is not aligned with 32-burst size, dummy data will be auto-added to align burst size. Then, data with dummy will be dumped to DDR3. In Avalon2SATA module, two state machines are designed to control each transfer direction independently.

Transport Layer source code is stored in “MMMt2SATA64.vhd” that also includes SATA IP Core and PHY Layer instance.

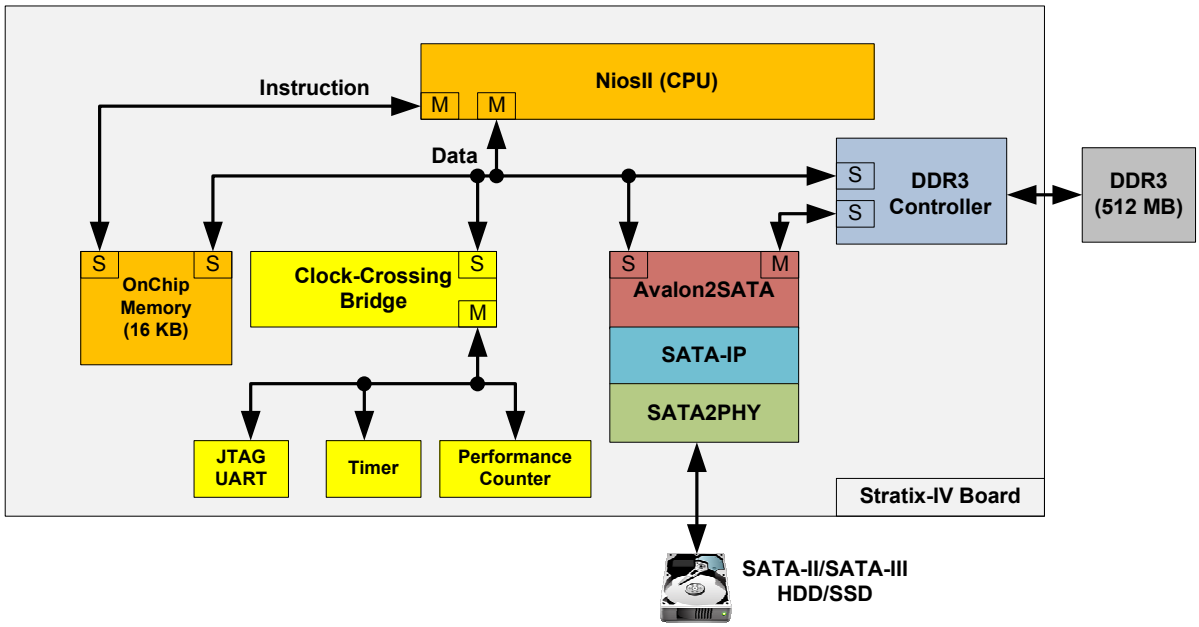

Figure 10 Block diagram of the reference design

- Application Layer by NiosII system

This reference design uses NiosII as a host processor, and uses DDR3 to be main memory which can be accessed by both NiosII and user logic connection for SATA-IP. DDR3 memory is split into four areas to store DATA FIS and other FIS of write and read operation, as shown in Figure 11. Program and data memory of NiosII is designed by using OnChip memory and user can send command and monitor result with NiosII by using JTAG UART. Since test result on test application shows transfer performance of each SATA device, Timer is required in NIOSII system.

Register map of control signals for this reference design is shown in Table 3. Software on NiosII along with DMA mechanism will transmit FIS data on the main memory to the Link Layer of SATA IP, or will receive data from the Link Layer to the main memory. Memory address and transfer size are controlled by Software. Data FIS header is added and removed by Avalon2SATA module, so Software do not need to mind about header information control for data management. After ending of each transfer, Finish flag will be set to let NiosII know and start next data transfer until complete. More details about Software are described in next topic.

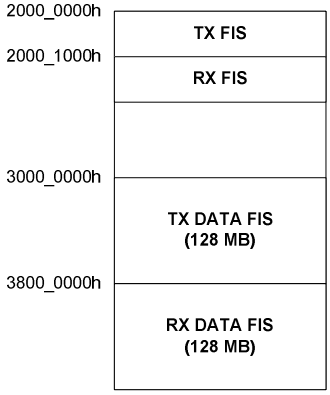

Figure 11 Memory map of Main Memory (DDR3)

Address Rd/Wr	Register Name	Description
BA+0x00 Rd/Wr	Transmit Data Address Reg (TX_ADDR)	Wr: Set start address of transmit data storage area. Rd: Return current address of transmit data storage.
BA+0x04 Rd/Wr	Received Data Address Reg (RX_ADDR)	Wr: Set start address of received data storage area. Rd: Return current address of received data storage.
BA+0x08 Rd/Wr	Control Reg (CONTROL)	Wr: [31] SATA Reset [30] Transmit direction (‘1’:TX, ‘0’:RX) [29] Transmit Data FIS [28] Start flag [23:0] Transmit/Received word count Rd: Return remaining transmit/received word count
BA+0x0C Rd/Wr	Status Reg (TRN_STATUS)	Wr: [31] Clear transmit finish flag [30] Clear received finish flag [29] Clear error flag Rd: [31] Transmit finish flag [30] Received finish flag [29] Error flag
BA+0x10 Rd	PHY Status Reg (PHY_STATUS)	[0]: SATA PHY LINKUP [1]: SATA Speed from DIPSW (‘1’:SATA-III, ‘0’:SATA-II) [23:16] Transmit FIFO counter [31:24] Received FIFO counter
BA+0x04 Rd	Error Code Reg. (ERROR_CODE)	SATA IP Error code. Set when transmit/receive completion. User can detect CRC or FIS error.

Table 3 Register Map of CPU

(BA : Base Address = 0x01000000)

3. 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. NiosII 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, NioII 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, and WRITE DMA EXT. This reference design can support both 48-bit and 28-bit LBA (Logical Block Address) 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 1	R 0	R 0	R 0	PM Port 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 instead 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 instead 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 “software/Sata_host/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.

Figure 12 shows reference design operation result on serial terminal screen.

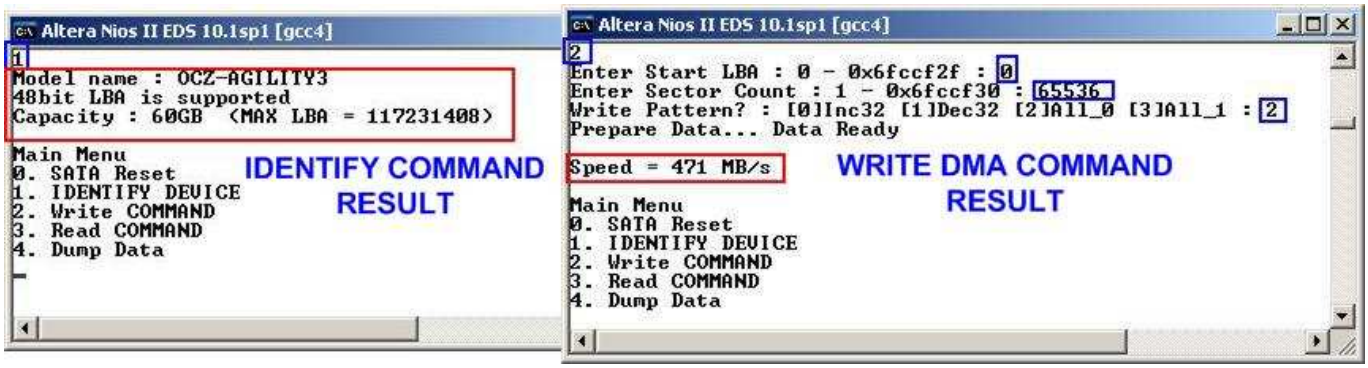

Figure 12 Operation result sample screen

4. Revision History

Revision	Date	Description
1.0	16-Jan-12	Initial Release