SATA-IP Host reference design on 7-Series and KCU105

Rev2.0 20-Jan-16

1. Introduction

Serial ATA (SATA) is an evolutionary replacement for the Parallel ATA (PATA) physical storage interface. SATA interface increases speed transfer to be 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.

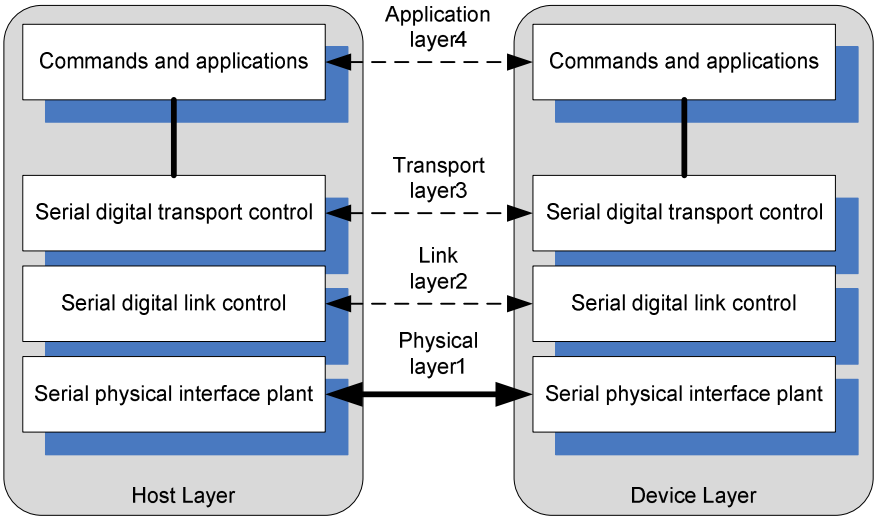

Figure 1-1 SATA Communication Layer

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 to transfer high speed data with SATA-III or SATA-II Device. The SATA-IP core is designed to operate with GTP/GTX/GTH transceiver of 7-Series and Ultrascale device for lower layer protocol and processor for upper layer protocol. More details are described as follows.

2. Environment

This reference design is based on the following environment as shown in Figure 2-1.

- AC701/KC705/ZC706/VC707/VC709/KCU105 Platform
- Vivado/ISE for programming bit file
- FMC SATA RAID board, provided by Design Gateway
- SATA-III/SATA-II Device (HDD/SSD) connecting to SATA connector on FMC SATA RAID board
- USB Micro-B cable for FPGA configuration
- USB Mini-B/Micro-B cable for serial communication. For serial communication, set baud rate=115,200 / data=8bit / Non-Parity / Stop=1bit.

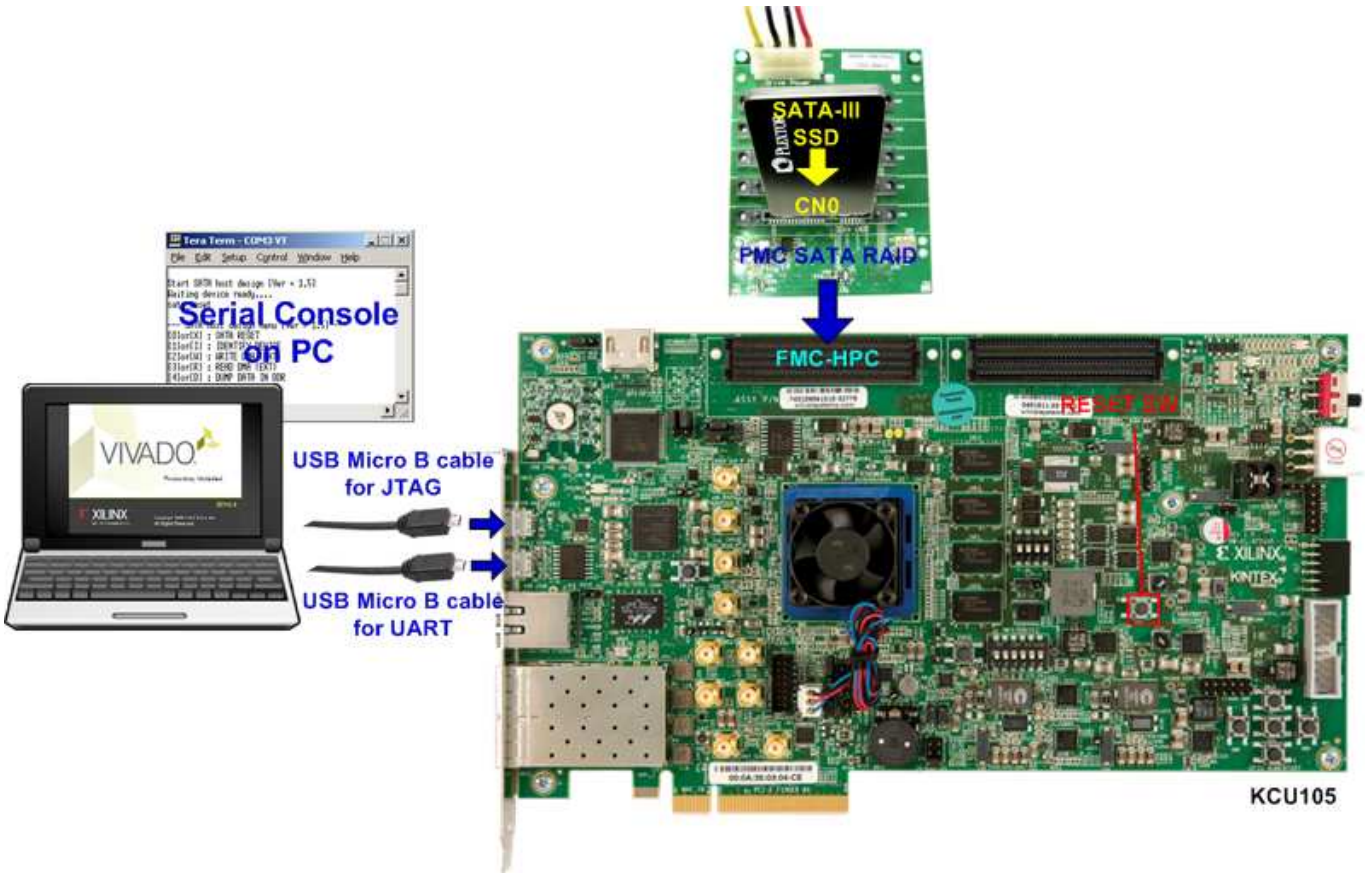

Figure 2-1 Reference design environment

3. Hardware description

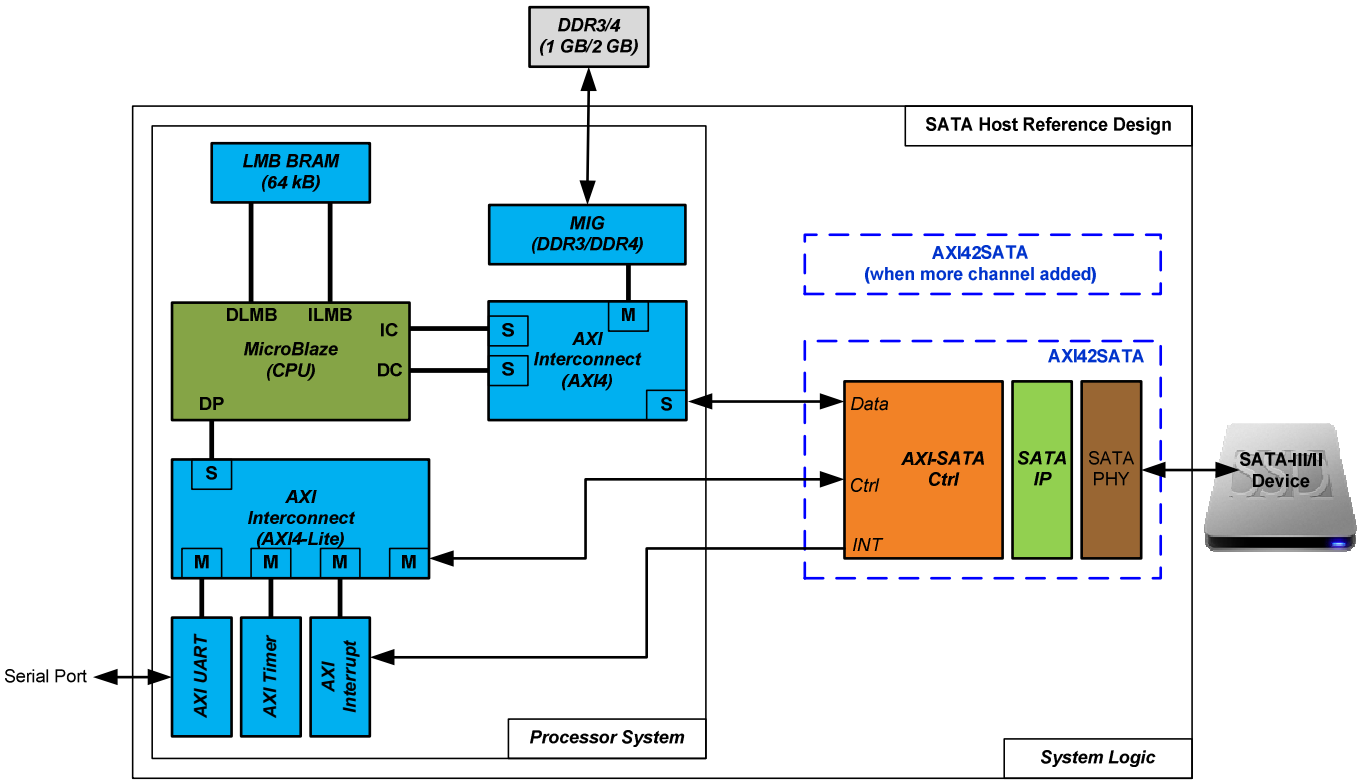

Figure 3-1 Block diagram of the reference design on FPGA board except ZC706 board

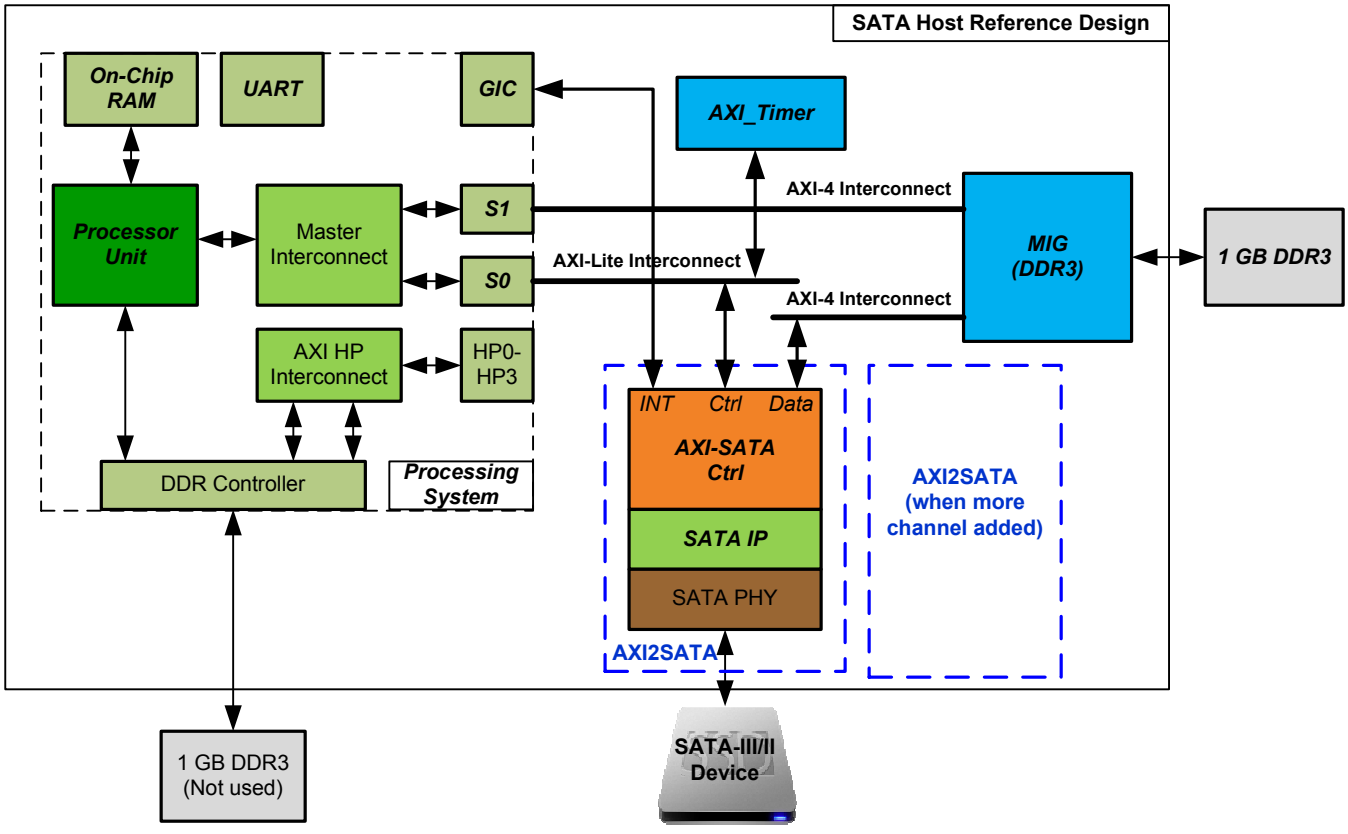

Figure 3-2 Block diagram of the reference design on ZC706 board

- SATA IP Host design implementation on FPGA

As shown in Figure 3-1 and Figure 3-2, the demo system consists of two parts, i.e. Processor system which includes processor and its I/O interface, and AXI42SATA module which is hardware logic.

Processor system includes the basic components such as UART, Timer, Interrupt controller, DDR3/4 controller, and Block RAM for storing instruction/data of the processor. AXI4-Lite and AXI4 interface in Processor system are exported to connect the hardware logic on top level.

System logic is HDL code which includes system wrapper of Processor system, and AXI42SATA logic connecting together through two bus interfaces. AXI4-Lite bus is the interface for processor (MicroBlaze/ARM) controlling AXI42SATA operation. AXI4 bus is the interface for DDR3/4 which is used for data buffer between processor and AXI42SATA. User can add more channels by duplicating AXI42SATA module on system logic and exports more ports of AXI4-Lite and AXI4 bus to connect to Processor system. The details of SATA-IP host demo are described as follows.

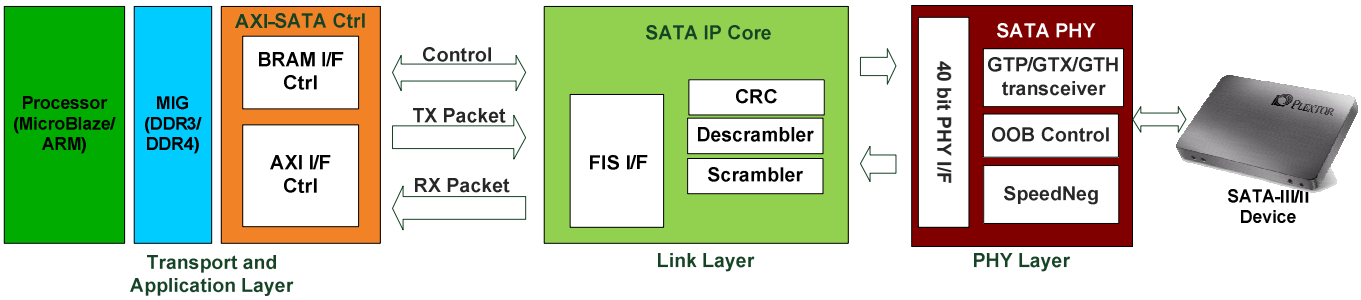

Figure 3-3 SATA-IP Host Demo Block Diagram

- PHY Layer

This layer is designed by using GTP/GTX/GTH module (built-in high speed serial circuit) of 7-Series/Ultrascale device, operating with logic control to generate OOB (Out-of-band) sequence and initialization sequence of physical layer. State machine to control OOB sequence is designed in “oob_control.vhd” which is sub-module of “sata2phy_xx.vhd” (xx is FPGA model), the top layer of PHY source code. To support both SATA-II/SATA-III speed, PHY on AC701/KC705/ZC706/ VC707 board also includes “speed_neg_control.vhd” module to run speed auto-negotiation function for selecting SATA speed to be SATA3 or SATA2.

The reference design follows PLL and transceiver reset sequence which has described in “Reset and Initialization” section within Transceivers User Guide. For AC701 board, reset sequence is designed within “gtp_rxreset_seq.vhd” while other devices include in “sata2phy_xx.vhd”.

Before building user board, user must read carefully and must follow design guide line described in “Board Design Guidelines” chapter within Transceivers User Guide.

Note: Transceivers User Guide of each device is follows.

- UG482: 7 Series FPGAs GTP Transceivers User Guide (AC701)
- UG476: 7 Series FPGAs GTX/GTH Transceivers User Guide (KC705/ZC706/VC707/VC709)
- UG576: Ultrascale GTH Transceivers User Guide (KCU105)

- Link Layer by SATA-IP

Link Layer and some part of Transport layer are implemented by SATA-IP. FIS packet format is converted to lower layer by including CRC and scramble processing. Also, packet from physical layer is decoded and arrange to FIS packet format to interface with user. More details about SATA-IP interface are described in “dg_sata_ip_datasheet_7series_en” document.

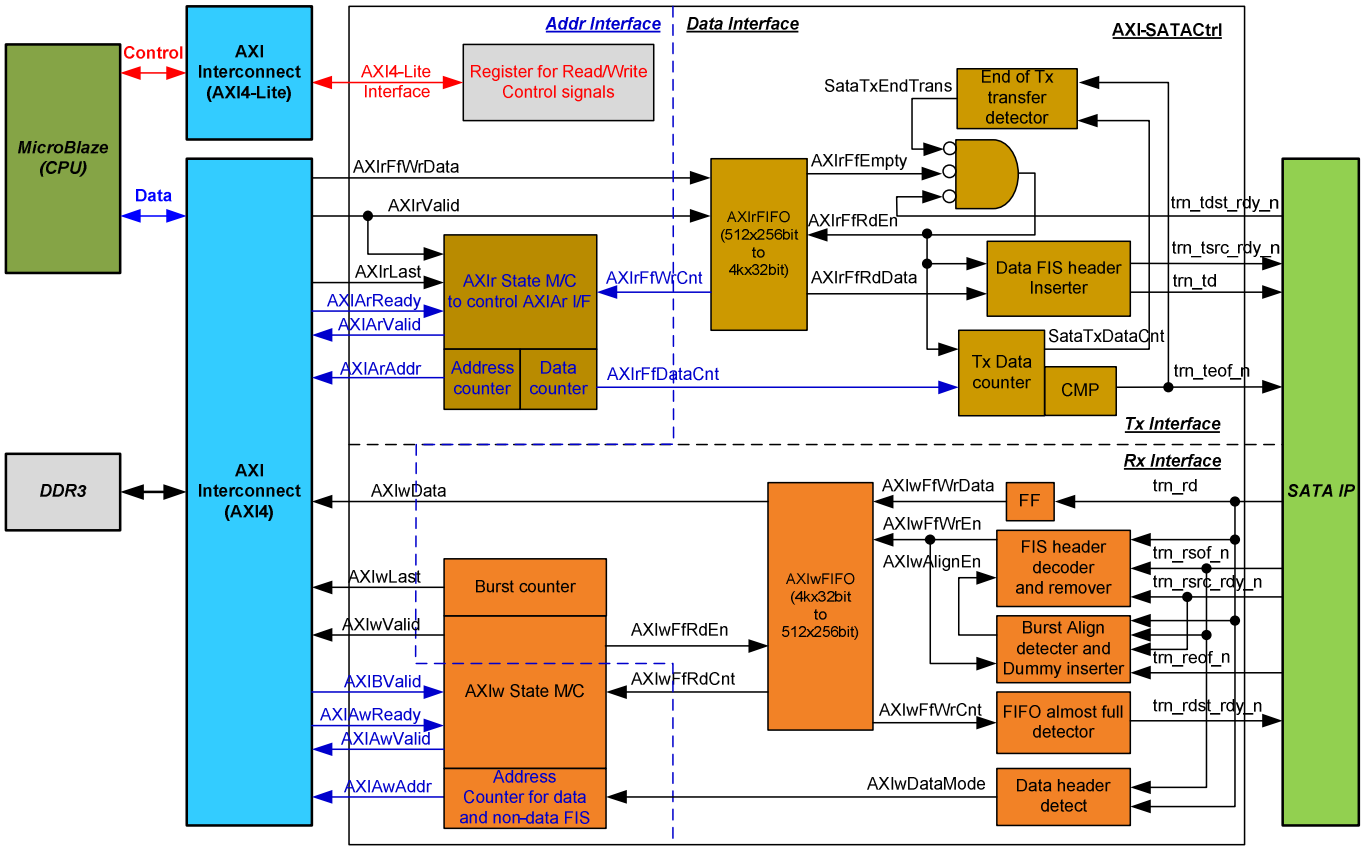

Figure 3-4 Block diagram of AXI-SATACtrl

- Transport Layer by AXI-SATACtrl

AXI-SATACtrl is the logic design to connect SATA-IP interface to standard interface of Processor system, i.e. AXI4-Lite interface for control signal and AXI4 for data signal.

Register map of control signals to interface with processor (MicroBlaze/ARM) is shown in Table 3-1. Main operation of these registers is to define DDR3/4 address for transferring FIS data, transfer length, and FIS type (data or non-data). The operation status can be monitored by processor accessing these registers.

Address Rd/Wr	Register Name (Label in the "sata_host.c")	Description (Bit order is little endian)
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)	[31] : Received FIS type in this interrupt ('1': Non-Data FIS, '0': Data FIS). This bit is cleared by writing bit[29] of INT_CLEAR = '1'. [23:0] : 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 DDR3/4 start address of transmit FIS data area Bit[8:0] of this value needs to be equal to 0 for sector alignment.
BA+0x04 Wr	Received Data Address1 Reg. (RX_ADDR)	Set DDR3/4 start address of received other FIS area (except DATA FIS type). Bit[8:0] of this value needs to be equal to 0 for sector alignment.
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 DDR3/4 start address of received DATA FIS area Bit[8:0] of this value needs to be equal to 0.
BA+0x10 Wr	Interrupt Clear Reg (INT_CLEAR)	[31] : Set this bit to clear ip2host interrupt [30] : Set this bit to clear host2ip interrupt [29] : Set this bit to clear received FIS type

Table 3-1 Register mapping from Processor

(BA : Base Address)

For data transfer with DDR3/4, AXI4 bus can be divided into four groups, i.e. AXIAr for read command request, AXIr for read data transferring, AXIAw for write command request, and AXIw for write data transferring. The requests for both read and write command are controlled by state machine. Data bus size of AXI4 is 256-bit while data bus size of SATA-IP is 32-bit, so the FIFO is used for data bus size converting with flow control. The sequence of read and write operation are follows.

For read operation from AXI to SATA, the operation is started by processor. After processor prepares FIS or data at DDR3/4, it will set CONTROL register to start data or non-data FIS transferring to SATA with setting the packet size. If data FIS is sent, Data FIS header will be auto-added by internal logic and then followed by the data from DDR3/4. AXIrFIFO is applied to convert data bus size and for data flow control. If FIFO is almost full, state machine will pause to request next data from DDR3/4 and wait FIFO space available enough. Also, the logic at SATA side will monitor empty flag of FIFO to read and send data out to SATA-IP. By above sequence, processor can simply create any FIS type with specified FIS size to SATA-IP.

For write operation from SATA to AXI, the operation is started by SATA. DDR3/4 address to store FIS from SATA is pre-defined by processor. Two different DDR3/4 areas are defined for storing Data FIS and non-DATA FIS packet, as shown in Figure 3-5. Data FIS is stored to RX_DATA_ADDR area while non-DATA FIS is stored to RX_FIS_ADDR following the operation mode. AXIwFIFO is used for data flow control and data bus size converting. If FIFO is almost full, SATA packet transferring from SATA-IP will be paused. If total data size in SATA packet from SATA-IP is too small and not aligned word (less than 256-bit), the dummy word will be filled to FIFO by internal logic. At the read side, state machine will be always monitored FIFO count to check that the data is much enough, and then send request and forward data out to DDR3/4 through AXI4 bus. The FIS header of every SATA packet is decoded to check FIS type that is non-data or data format, and then select the correct DDR3/4 address to store the SATA FIS.

Data transactions in both directions are size-fixed to 256-bitx16 beat (512 byte) for simple design logic and high performance transfer. AXI-SATACtrl logic operating with SATA-IP and PHY layer code are stored in “AXI42SATA.vhd”, provided to user as delivery item.

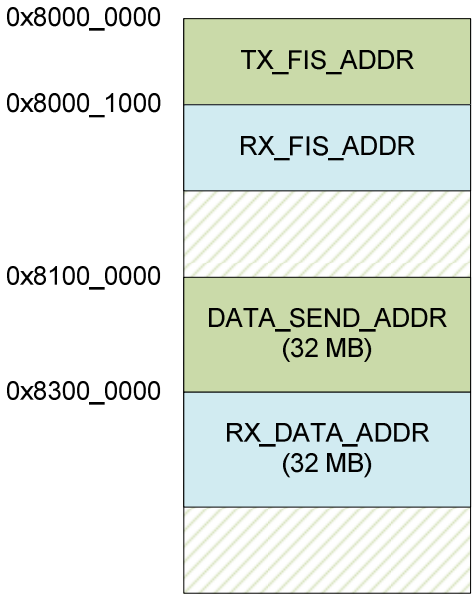

Figure 3-5 DDR3/4 Memory map in the demo

- TX_FIS_ADDR for storing transmit non-DATA FIS from processor to SATA device
- RX_FIS_ADDR for storing received non-DATA FIS from SATA device to processor
- DATA_SEND_ADDR for storing transmit DATA FIS from processor to SATA device
- RX_DATA_ADDR for storing received DATA FIS from SATA device to processor

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. Processor 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, processor 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

Table 4-1 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)

Table 4-1 IDENTIFY COMMAND FIS structure

- READ DMA EXT

Table 4-2 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)

Table 4-2 DMA READ EXT FIS structure

- WRITE DMA EXT

Table 4-3 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)

Table 4-3 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.

Figure 4-1 shows reference design operation result on serial terminal screen.

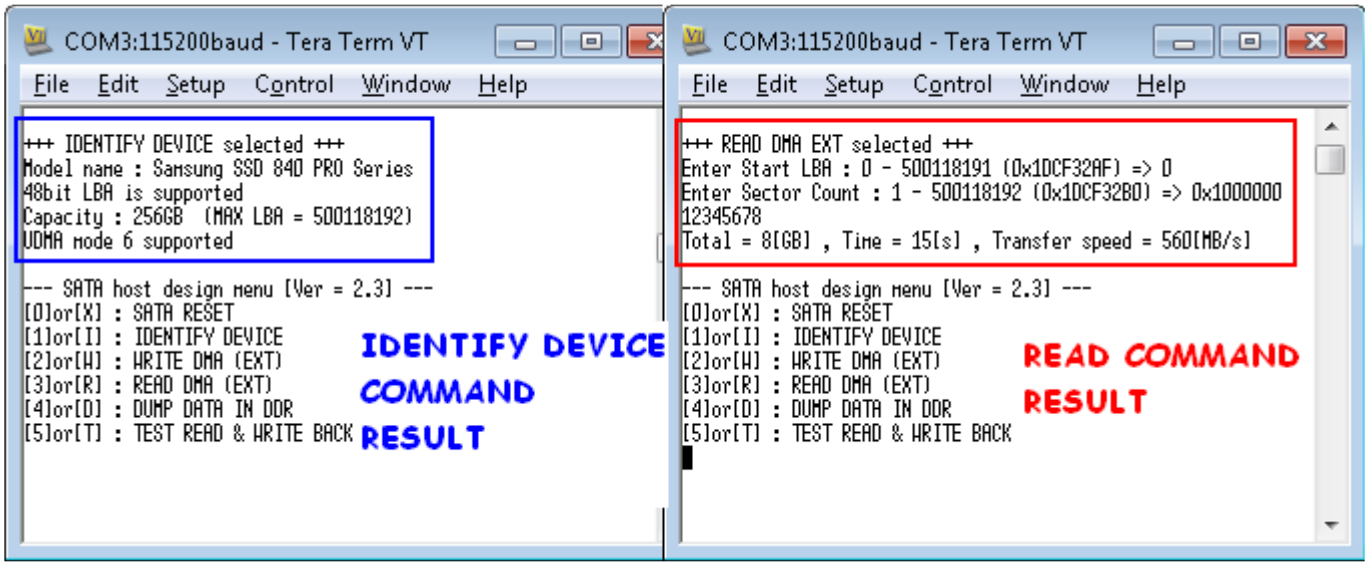

Figure 4-1 Operation result sample screen

5. Revision History

Revision	Date	Description
1.0	21-Apr-14	Initial release
1.1	1-Jul-14	Update to Vivado2014.2 version
2.0	20-Jan-16	Support 7-series and KCU105 board