IOP Conf. Series: Materials Science and Engineering **326** (2018) 012001 doi:10.1088/1757-899X/326/1/012001

# A minimal SATA III Host Controller based on FPGA

Hailiang Liu[1,*]

[1]Chengdu Goke Microelectronics Co., Ltd., 5th Floor, Tower B, Oriental Hope Building, 3 Gaopeng Avenue, High-tech Western Zone, Chengdu, Sichuan, China

[*s]hailiangliu@std.uestc.edu.cn

**Abstract.** SATA (Serial Advanced Technology Attachment) is an advanced serial bus which has a outstanding performance in transmitting high speed real-time data applied in Personal Computers, Financial Industry, astronautics and aeronautics, etc. In this express, a minimal SATA III Host Controller based on Xilinx Kintex 7 serial FPGA is designed and implemented. Compared to the state-of-art, registers utilization are reduced 25.3% and LUTs utilization are reduced 65.9%. According to the experimental results, the controller works precisely and steady with the reading bandwidth of up to 536 MB per second and the writing bandwidth of up to 512 MB per second, both of which are close to the maximum bandwidth of the SSD(Solid State Disk) device. The host controller is very suitable for high speed data transmission and mass data storage.

## 1. Introduction

In recent years, the area of memory is developing rapidly. As for its low power consumption, good stability and high speed in storage, SSD(Solid State Disk) is gradually occupied the storage market. At present, many SSD products of the main companies in storage area use SATA (Serial Advanced Technology Attachment) interface to transmit data, such as Intel, Toshiba, Samsung and Seagate.

SATA, with a total of 7 signals included with 3 GND signals, a pair of TX differential signals and a pair of RX differential signals, transmitting and receiving with a pair of differential signals, effectively solves the problem of parallel signal transmission crosstalk in high speed digital bus design.

SATA was first proposed by Intel in 2000 to replace the parallel ATA interface at that time. In 2001, SATA 1.0 standard was released and specified rate of data transmission of 150 MB per second. Just one year later, SATA 2.0 standard was born and the transmission speed was up to 300 MB per second. To meet needs of high speed of data transmission and mass data storage, the Serial ATA International Organization released SATA 3.0 standard [1] in 2009 with a speed of 600 MB per second, which was a double rate of the previous generation.

A lot of related research work has been carried out in recent years. Ling yan Fan et al. [2] put forward a novel hardware architecture with embedded AES hardware engine, which made a flash controller simple and reduce the die size. Jianjun Luo et al. [3] implemented a SSD controller with up to five flash channels, which can be configured as RAID0 or RAID5 mode. Abdul Arfan et al. [4] tried to optimize the performance of SATA disks by devising a new cache algorithm that is aware of disk access time. Young-Jin Kim [5] proposed a novel cache algorithm which combined disk access time and access frequency in a more NCQ-friendly way to enhance the I/O performance. From the public information point of view, there are several commercial and academic SATA host controller cores implemented based on FPGA [6,7,8,9,10]. Wu et al. [6,7] implemented SATA HBA core based on

Xilinx Virtex 5 FPGA and SATA revision 1.x in 2008 with a reading bandwidth of up to 111 MB per second and writing bandwidth of up to 102 MB per second. Woods et al.[8] implemented SATA HBA core based on Xilinx Virtex 5 FPGA and SATA revision 2.x in 2012 with a reading bandwidth of up to 279 MB per second and writing bandwidth of up to 242 MB per second. In 2012, Mendon et al. [9] implemented SATA 2.x HBA and one year later, Lehmann et al. [10] implemented SATA HBA based on SATA revision 3.x. The difference of these host controller cores are shown in table 1.

**Table 1.** Academic SATA cores based on FPGA

| | Wu[7] | Woods[8] | Mendon[9] | Weasel[10] | Liu |
|---|---|---|---|---|---|
| SATA Rev. | 1.x | 2.x | 2.x | 3.x | 3.x |
| NCQ | — | yes | yes | — | — |
| DMA | yes | FP-DMA | yes | yes | yes |
| Interface | PLB | FIFO | PLB | FIFO | FIFO |
| Virtex-5 | X | X | X | | |
| Virtex-6 | | | X | | |
| Kintex-7 | | | | | X |
| Straix II | | | | X | |
| Straix IV | | | | X | |

This paper designs and implements a minimal SATA III Host Controller based on Xilinx Kintex 7 serial FPGA platform, which is shown in section 2 and section 3. The rest of the paper is organized as follows: Section 2 presents the design of the SATA III Host Controller. The process of implementation is described in Section 3. Section 4 analyzes the experimental results and section 5 concludes this paper.

## 2. Design phase

In design process, to achieve a minimal SATA III host controller, asynchronous transmitting FIFO and receiving FIFO interface is proposed. Compared to PLB (Processor Local Bus) interface adopted in paper [7, 9], asynchronous FIFO interface is simple to design and flexible to use. What's more, the FIFO interface also cuts down the logic expenditures during implementation phase. Compared to paper [10], which places FIFO before 8b/10b encode and decode, at this time the transmitting data are packaged into FIS (Frame Information Structure) and added with SOF (Start Of Frame), CRC(Cyclic Redundancy Check), EOF(End Of Frame), so our transmitting FIFO could have a relatively shallow depth(the receiving FIFO has a similar explanation), which also saves RAM resource.

In our design, the data width is 32 bits, and the reference clock is 150 MHz. So after 8b/10b encoded and parallel-to-serial, the speed of SerDes is up to 6 Gb per second. Then we choose Xilinx XC7K-325T FPGA as our hardware implementation platform.

In physical layer, the initial of host controller is the key to implement the design. Here figure 1 descr-ibes the initial flow of host controller. When power on or hot plug, host will have a globe reset, then host will send COMRESET, one of three (COMRESET, COMINIT, COMWAKE) OOB (Out Of Band)signals. In the following, host will wait for receiving the COMINIT signal from device. When receiving COMINIT signal, host will send COMWAKE signal to device and wait for receiving COMWAKE signal from device. After receiving COMWAKE for 64 clock period, host will reset Rocket I/O. The reason we do this is to avoid k code error if Rocket I/O doesn't have a reset after host receiving COMWAKE signal. Then host will send D10.2 signal and ALIGNp primitives at 6 Gb per second to device. In order to reduce logical expense, our minimal SATA host controller only supports SATA 3.0 protocol. Device will also send ALIGNp primitives at the same speed to host and the initial of physical layer is completed. Then the link up signal will go high to indicate that the physical link is established, which will be transmitted layer by layer until application layer.

In application layer, in order to verify the performance of Host Controller, one hand we design a random number generator which will write data to SSD device randomly. Correspondingly, an error

counter is designed to count the error data reading from SSD device which is not equal to the writing data.

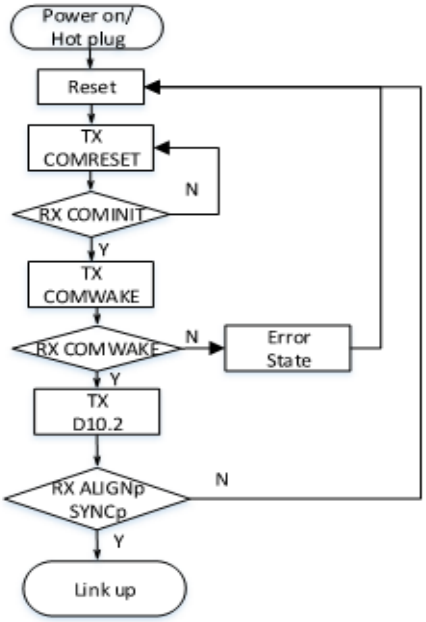


**Figure 1.** The initial flow of host controller

## 3. Implementation

Here figure 2 gives the simplest structure of SATA III Host Controller. We can see that the structure could mainly divided into two parts: the writing path and the reading path and each path has four layer: application layer, transport layer, link layer and physical layer. As a minimal SATA III Host Controller structure, only the basic function of each layer are implemented.

Application layer provides a FIFO interface to transport layer and responses to answer all interface commands, analyze the device operation commands and do corresponding operation according the protocol. Besides, we add the corresponding logic to monitor the status of writing and reading proce-ss. Based on the Phy initialization status, the host adapter knows whether a device is attached to each of the two ports used in a Master/Slave emulation configuration. Device Control register writes, that have the SRST bit set to one, shall result in the associated Shadow Device Control register being writt-en for each port to which a device is attached. The frame transmission protocol for each associated port executed, results in a Register-Host to Device FIS being transmitted to each attached device. Transport layer presides over serial digital transporting control, mainly including packing control inf-ormation and data into FIS type and transporting to link layer. The another work of transport layer is to manage the conflict between reading and writing, unpack FIS when reading data from device. A FIS is a group of Dwords that convey information between host and device. Primitives are used to define the boundaries of the FIS and may be inserted to control the rate of the information flow.

In link layer, the FISs will comprise a frame with SOF, CRC, EOF. In the process of the transmitting, there will be added ALIGNp, SYNCp, HOLDp, HOLDAp, R OKp, R IPp and so on primitives to con-trol the flow the transmitting. Then the data flow will be scrambled and transmit to physical layer. The Link layer transmits and receives frames, transmits primitives based on control signals from the Trans-port layer, and receives primitives from the Physical layer which are converted to control signals to the Transport layer. The coding scheme used by Serial ATA translates unencoded data and control bytes to characters. The encoded characters are then transmitted by the Physical layer

over the serial line where they are received from the Physical layer and decoded into the corresponding byte and control value.

In physical layer, data flow will be encoded in 8b/10b algorithm and transmit to storage device thro-ugh Rocket I/O transceiver with 6 Gb/s. Physical layer provides specified OOB signal detection and transmission, uses OOB signal protocol for initializing the Serial ATA interface, then uses this OOB sequence to execute a pre-defined speed negotiation function. Perform proper power-on sequencing and speed negotiation. So far, all jobs of the writing path are done and the reading path is designed of converse job compared to the writing path.

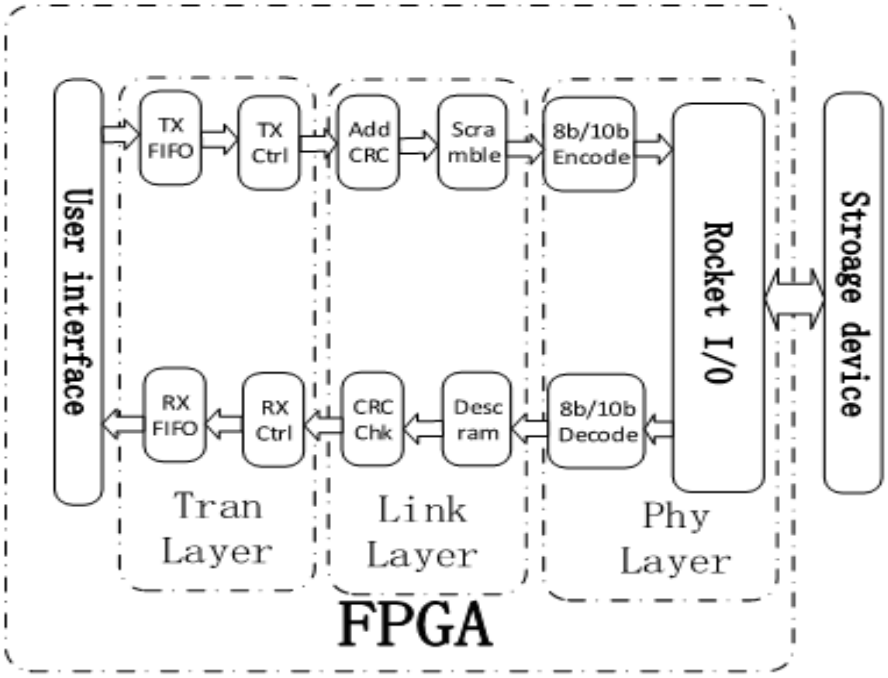


**Figure 2.** The structure of minimal SATA III Host Controller

## 4. Results and analysis

As we described before, we implement our SATA III host controller based on Xilinx Kintex XC7K325T FPGA. We choose Samsung Pro 850(MZ-7KE512B) as our storage device, whose reading bandwidth is up to 540 MB per second and writing bandwidth is up to 520 MB per second under ideal condition according to the vendor . As papers[6,7,8,9] are design implementation of SATA I or SATA II HBAs(Host Bus Adapters), the reading bandwidth and writing bandwidth of SATA I, SATA II and SATA III are lack of comparability. Figure 3 is the describe of reading bandwidth and writing band-width of different HBAs[6].

From Figure 3,we can see that the steady reading bandwidth of S2GX:WD Caviar Blue 500G is about 120 MB per second[10], the steady writing bandwidth of S2G:OCZ Vertex3 120G and K7: Samsung Pro 500G is about 535 MB per second, which is close to the maximum value of the storage device. Meanwhile, the reading bandwidth fluctuations of S2G:OCZ Vertex3 120G is a little bigger than K7: Samsung Pro 500G device.

Figure 4 indicates that our writing bandwidth is up to about 510 MB per second, which is close to the maximum value 520 MB per second of the device. How about the FPGA resource utilization? Com-pared to paper[10], we get table 2. It is easy to find that, compared to Weasel on S2GX, our LUTs utilization are reduced 35% and RAMs utilization are reduced from 71 to 2. Compared to Weasel on S4GX, our registers utilization are reduced 25.3% and LUTs utilization are reduced 65.9%.

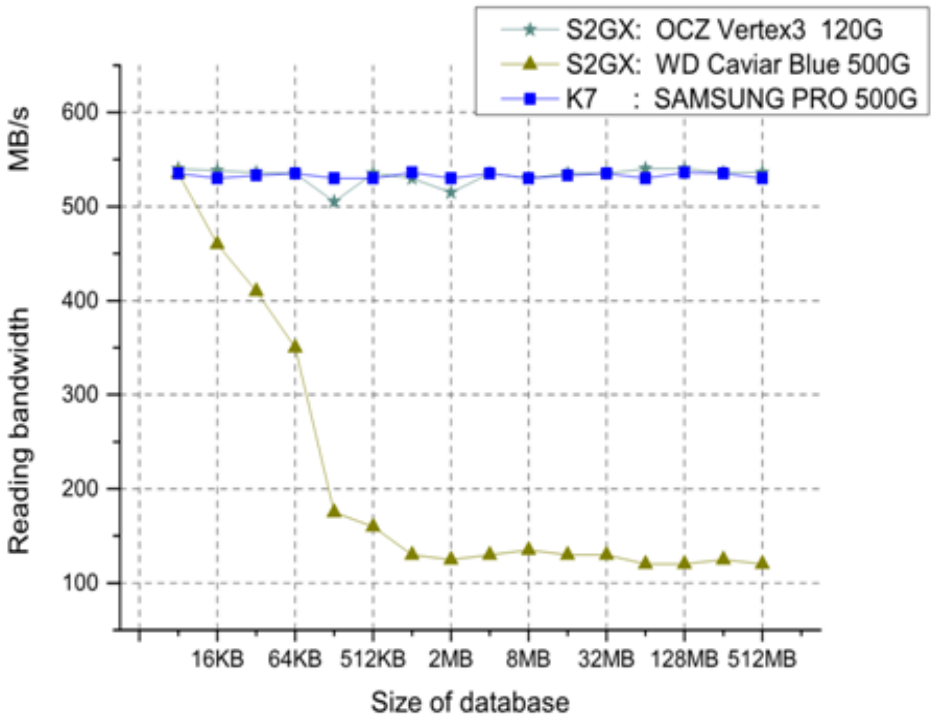


**Figure 3.** The reading bandwidth of the host controller

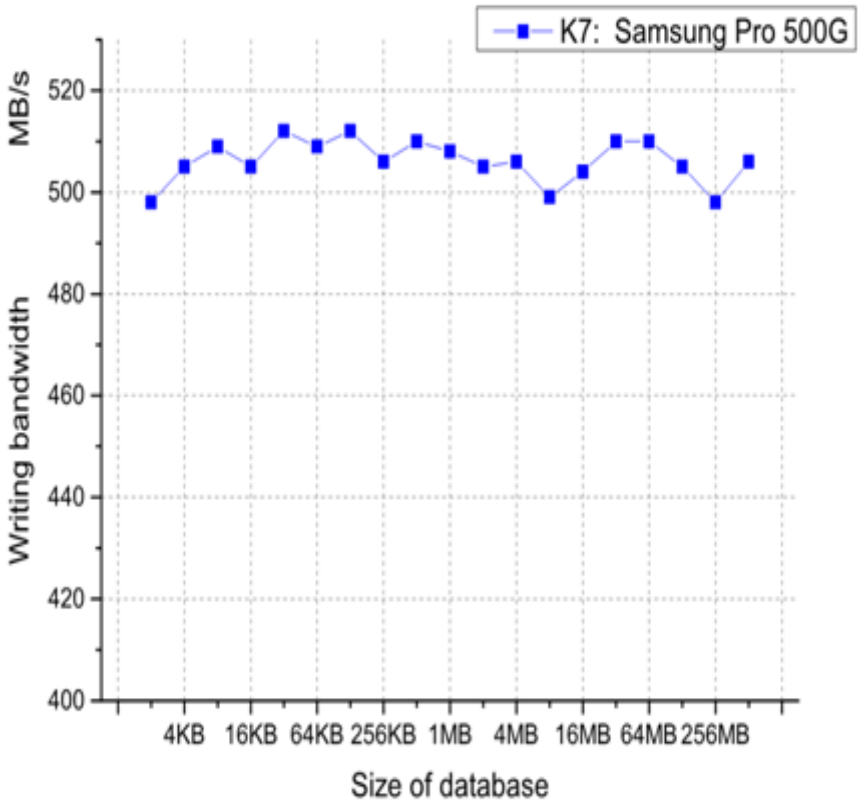


**Figure 4.** The writing bandwidth of the host controller

**Table 2.** FPGA resource utilization compare among different papers

| | Reg. | LUTs | RAM | SATA Rev |
|---|---|---|---|---|
| Weasel on S2GX[10] | 1884 | 2630 | 71 | 3.x |
| Weasel on S4GX[10] | 2027 | 3219 | 2 | 3.x |
| Liu on K7 | 1617 | 1940 | 2 | 3.x |

## 5. Conclusion

In this express, we design and implement a minimal SATA III host controller based on Xilinx XC7K-325T FPGA and the storage device we choose is Samsung MZ-7KE512B, which is a new kind of high performance SSD. According to the results, the controller works precisely and steady with the reading bandwidth of up to 536M/s and the writing bandwidth of up to 512 MB per second, both of which are close to the max bandwidth of the SSD. Meanwhile, compared to Weasel on S2GX[10], our FPGA resource LUTs utilization are reduced 35% and RAMs utilization are reduced from 71 to 2. Compared to Weasel on S4GX[10], our registers utilization are reduced 25.3% and LUTs utilization are reduced 65.9%. The controller is very suitable for high speed data transmission and huge data storage in SOCs, embedded system or as a SATA III host controller core integrated in FPGA.

## Reference

[1] Dell C,Hitachi G, Intel C, Maxim I, Seagate T and Weatern D 2009 *Serial ATA International Organization* vol 3 chapter 7-12 pp 183–535
[2] Lingyan F, Jianjun L, Hailuan Liu and Xuan G 2014 *J. IEICE Electron Express* **11** 20140535
[3] Jianjun L, Lingyan F, Chris T and Xuan G 2014 *J. IEICE Electron Express* **11** 20140419
[4] Abdul A, Young-Jin K and Jin B 2014 *J. IEICE Electron Express* **9** 1707-13
[5] Young-Jin K 2014 J. IEICE Electron Express 11 20140363
[6] Wei W, Hai-Bing S and Qin-Zhang W 2008 Proc. Int. Conf. on Computer Science and Software Engineering (Wuhan) vol 6 (IEEE Press) p 18
[7] Wei W, Hai-Bing S and Qin-Zhang W 2009 Proc. Int. Conf. on Computational Intelligence and Design (Changsha) vol 2 (IEEE Press) p 124
[8] Louis W and Ken E 2012 Proc. Int. Sym. on Field-Programmable Custom Computing Machines (Toronto) vol 20 (IEEE Press) p 220
[9] Ashwin A. M, Bin H and Ron S 2012 Proc. Int. Conf. on Field-Programmable Logic and Applications (Oslo) vol 22 (IEEE Press) p 421
[10] Patrick L, Thomas F, Oliver K, Steffen K, Thomas B. P and Rainer G. S 2013 Proc. Int. Sym. on Field-Programmable Custom Computing Machines (Porto) vol 23 (IEEE Press) p 1