Feasibility Test of Protocol Engines for a New Video System in Packet Communication

Hyukje Kwon and Yongseok Choi

Abstract—**The layout between the processor and memory in parallel bus is very complex and difficult to place and route. The expansion of memory capacity and bandwidth is limited. A new memory system using an optical connection is proposed. We designed a serial interface using packet communication, and implemented a protocol engine to be executed on the interface. To test the feasibility of the protocol engine, we implemented a video system using an embedded processor on FPGA. The master and slave protocol engines were on the same FPGA, but used the clock differently. We conducted an experiment on the function of the proposed protocol engine between the video frame buffer and memory using a 2×10-Gbps serial link.**

Index Terms—**Memory channel, protocol, main memory.**

I. Introduction

Recently, due to the development of the internet and personal communication, a large amount of data and information have been produced and circulated. Since a large amount of data requires faster networks and more effective processing, there has been the need for a more computing power and more memory. The processor's memory access is made through a memory channel. In order to transfer a normal data signal using parallel channels on a board, the propagation delay time of its inter-signals should be constant. This is the reason why meticulous P&R is required. If you need more memory capacity and more bandwidth at server-class data centers, you should attach the memory closely to the server processor [1]. In addition, the issues of more required memory bandwidth and capacity are emerging from "multi-core and many-core" that has been evolving recently [2], [3].

In this environment, there are several requirements to improve the performance of memory systems: the growth of memory capacity, the decrease of memory latency, the expansion of memory bandwidth, and the increased speed of the memory I/O operation, among others [4]-[6].

Firstly, the stacked structure has been exploited to increase the memory capacity. There are several memory structures. HMC (hybrid memory cube) [7] has a stacked structure, and now consists of 4 stacks. HBM uses the interposer to expand the bandwidth between DRAM and GPU or CPU, and Wide IO also has a stacked structure, and so on. But it is known that there is a problem with the formation and reliability of the TSV and the micro-bump inside a stacked die [8], [9].

Manuscript received December 2, 2014; revised February 27, 2015. This work was supported by the ICT R&D program of MSIP/IITP, Korea. [10038764, Silicon Nano Photonics Based Next Generation Computer Interface Platform Technology].

Hyukje Kwon and Yongseok Choi are with the Electronics and Telecommunications Research Institute, Korea (e-mail: {heavenwing, shine24}@etri.re.kr).

Secondly, the decrease of memory latency has been studied to reduce the access delay, which is carried out mainly on SDRAM that has a relatively long latency time. The internal cell I/O or external I/O of memory connected by an optical-fibre are being studied [10]. External optical IO is especially converted from electrical IO on a one-to-one basis.

Lastly, to increase the memory bandwidth, while I/O operation speed still has the limits of 200Mbps, I/O pin count is increased to 512 and 1024 similar to Wide IO (2) or HBM. They could be connected directly using external IO connections or the connection between the memory and the processor could be configured on SiP.

DDR SDRAM is generally used as main memory, and the data width between the memory controller in the processor and the memory is generally 32-bit or 64-bit. In addition to the 40-bit address and control information, more than 100 signal lines are required (not including the power lines). In the case of the latest processors, the memory channel (dual) bus occupies about 40% of all package pins [11]. Also, the operating speed of DQs is a 3.2-Gbps/pin in DDR4 [12].

But as the memory I/O speed increases, the stability of the signal between the memory and processor is influenced by both the length and crosstalk. The data lines of the parallel bus between the processor and memory causes difficulty in minimizing the skew of the inter-line. For these reasons, we can see a relationship where the number of DIMMs decrease as the data rate increases [13].

The focus of this paper is related to a memory I/O. Using high-speed serial communication and applying a packet based protocol engine, we designed protocol engines that are able to communicate with each other, a processor and the memory [1], [14]. We implemented a video system to test the feasibility of the protocol engines. This system used a 2×10-Gbps transceiver in each protocol, one used for the control packet, and another used for the data packet.

II. Design of Memory Systems and Protocols

A. Proposed Memory Systems

We proposed a memory system to deal with communication problems between a processor and memory controller or between a memory controller and memory as shown in Fig. 1. Earlier studies had dealt with a memory controller and the processor or a memory controller and the memory. In [14], the memory controller was installed in the in/out side of the processor, and overrode these functional systems.

The proposed memory system does not use parallel communication as the conventional method between the memory controller and the memory bus, and some parts of the functions of the memory controller would be built inside the memory. Instead, the proposed memory system uses the transceiver in serial. The transceiver may use an optical transceiver, if a silicon-nano photonics transceiver is used, and the results should be better. However, in this paper the test system used QSFP as a socket and 40 Gigabit Ethernet as the physical layer of the transceiver.

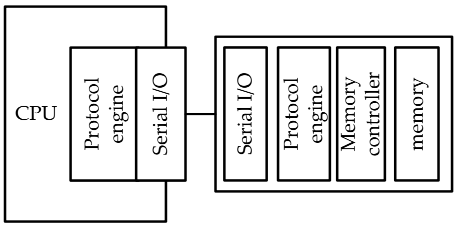


Fig. 1. Proposed memory system including the protocol engine.

We implemented the video system to test the feasibility of our design. As shown in Fig. 2, the master protocol engine connects to the video frame buffer instead of the CPU. And the slave protocol engine operates as the memory. The video frame buffer gets the 65-byte data from the video input module, and it executes the write access to memory. In addition, the video output module requests the data from the video frame buffer. The color depth is 24-bit/pixel. The video frame buffer accesses the read operation of the memory and has a dual clock buffer inside. This procedure always occurs in the video system.

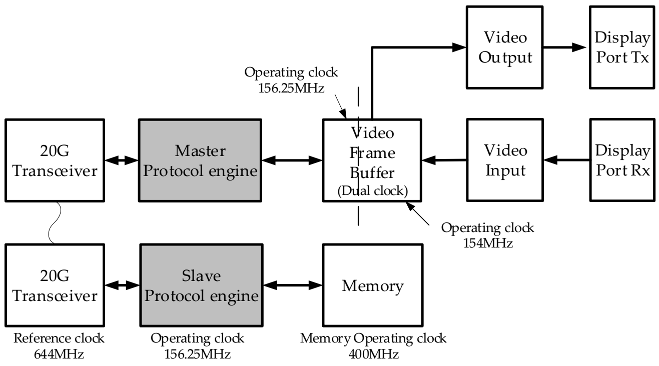


Fig. 2. The block diagram of an implemented system.

B. Protocol's Function and Roles

As shown in Fig. 1, the function of the memory controller can be separated. The master protocol engine of the main control processor is in charge of a function in the memory controller. Its role is a packetizer of the memory address and data generated by the processor. In memory side, there is also a control (slave) protocol engine. The memory system can be configured with master and slave protocol engines.

There are three main roles of the protocol engines. The first is data collection and sorting. The second is data error detection and the retransmission process. The last is the packetizing of the memory command and the data. The master protocol engine is responsible for the packetizing of the memory-related commands, data and addresses generated by the processor. Another main role of the master protocol engine is to manage the detection and correction of errors about the data collected, and if re-transmission is required, it has to retransmit the packet that has been previously transmitted to the memory chip. The slave protocol engine passes the requested data to the internal logic of the memory, generates the corresponding response packet, and then transmits it to the master control protocol engine. Also, if the slave engine has decided that re-transmission is required, it has to retransmit the packet that had previously been transmitted to the master protocol engine.

C. Controlling Buffers

In a protocol engine, the plurality of the buffer may reside. The control modules and buffers in each engine may be built-in. We can express the relationship with the controlling buffers of a protocol engine as shown in Fig. 3 and Fig. 4. A buffer controller (1) controls the input and output of each buffer accessed by the protocol engine, manages the state of each buffer, and is responsible for communicating with the external logic device.

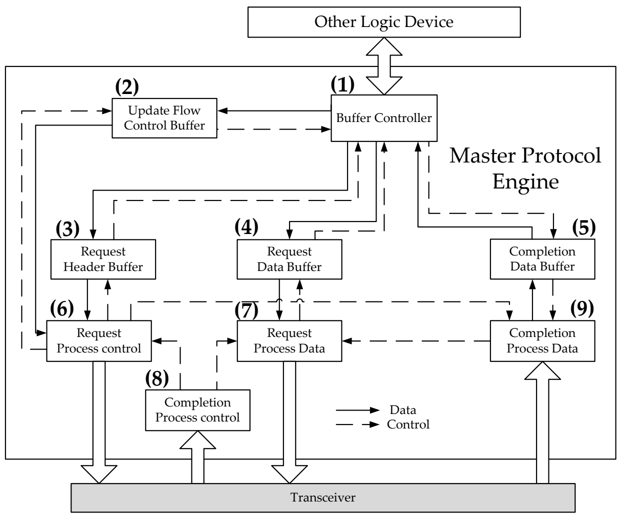


Fig. 3. Control buffer in the master protocol engine.

(2)~(5) in Fig. 3 and Fig. 4 are the buffers, and (6)~(9) is the place that the packets are generated and processed for reading or writing information. The update flow control buffer (2) generates a packet for controlling the flow of data. The request header (3) and data (4) buffer generate a packet for a header and data. The completion data buffer (5) receives the data and generates data for completion. It also does (6) and (7) for transmission by using a packet buffer, and the received data may be checked and interpreted in (8) and (9).

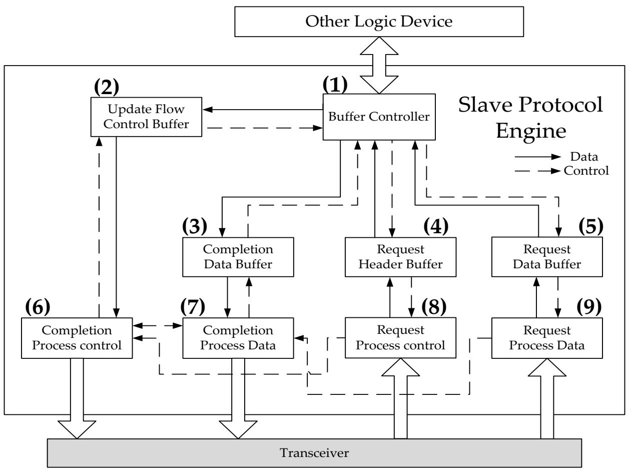


Fig. 4. Control buffer in the slave protocol engine.

The important information for operating the protocol engine's internal buffer is the state of the buffers and whether there is a request for read/write. In order to process read/write access, we should create packets which are stored temporarily in a buffer. When there is a request for a header and data buffer, if the buffer's space is full, the protocol engine waits until it has free space. After generating and transmitting the packet read request, and the engine waits for a response. The received data when responding is written into the completion (master) or request (slave) buffer and then the engine can access the data in the buffers, depending on the state of buffer and the data, and may be transferred to the next level module.

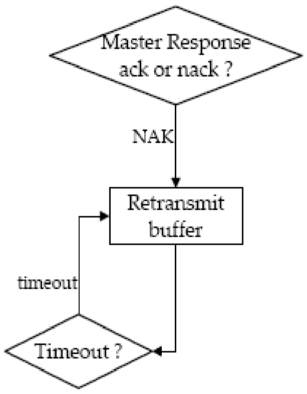


Fig. 5. Flow chart of retry sequence.

D. Packet Re-transmission

The packet retransmission means that the destination device requests for the source device to resend the packet when an error occurs in the packet's communication. For this purpose, both protocol engines should keep the transmitted packets in the buffer. If there is a retransmission request generated by either engine in accordance with error detection, that engine should resend the packet that was transmitted in the order used before. If the normal memory access process is completed without the retransmission request, the data in that buffer is to be deleted or classified as not being used, and the data are then replaced with new data. The retry sequence appears as shown in Fig. 5.

III. Compare with Other Systems

In Table I, we summarize the features of each memory system: pins, data rates, bandwidths, etc. The DIMM was configured as the memory channel in the conventional processor, which uses about 40% of the number of pins in the processor as described above. Through these proposals, if the protocol engine and the memory controller can be mounted on the individual memory chip, the channel interface can be simplified. If the Intel CPU can be configured in memory channel 1 to 3, DDR3/4 DIMM needs 140 pins per channel (excluding the power pins: Micron MT41J128M16HA-125, MT41J128MP8JP-126, 16M×72×8 bank). In the case that three memory channels are used, and the total pin count to be assigned to the CPU was 420 pins.

We show the available channel & pin count of memory systems if the total bandwidth is 320GBps, which is the maximum bandwidth of HMC. If the memory system is configured at OP10, we can remove 90% of DDR3 pins, 83% of DDR4 pins, 50% of HMC pins and 90% of HBM pins. And if the memory system is configured as OP25, we can remove 96% of DDR3 pins, 94% of DDR4 pins, 80% of HMC pins, and 96% of HBM pins.

TABLE I: Features Compare of the Memory Systems

Feature	DDR3	DDR4	HMC	HBM	OP10	OP25	
Pins/ch.	140	140	64	128	16	8	16
Data rate (Gbps)	0.8~3.0	1.6~3.2	10	1	10	25	25
Ch.BW (GBps)	6.4~17	12.8~25.6	40	16	20	25	50
Channel	1~3	1~3	8	8	1~26	1~52	1~26
Total BW (GBps)	6.4~34	12.8~51.2	320	128	20~520	25~1300	50~1300
Total Pins	140~420	140~420	512	1024	16~416	8~416	16~416
HMC consist of a link using Tx(16 lane) and Rx(16 lane). The lane consists of a differential pair. In this paper, the channel is same as the link of HMC. OP is Optical Fiber, which consist of Tx/Rx signal pair. OP's pin limits to maximum DDR3/4's pin. OP10's data rate is 10Gbps, OP25 is 25Gbps.							

IV. Feasibility Test

A. Test Setup

To test the feasibility of the protocol engine between memory and video, the embedded processor on Altera Stratix V FPGA was emulated. The maximum operating speed of the embedded processor was 200MHz, the width of system bus was 32-bit, and the processor (video) system was configured with inner program memory, DisplayPort logic, general I/O, JTAG UART, and the master protocol engine. In addition, the memory system was a memory controller, memory, and slave protocol engine as is shown in Fig. 6.

We constructed a single FPGA system for ease of testing on a single FPGA. This means the processor (video) system and memory system were on a single FPGA. They were connected by an optical line, but the internal connection was not, and used a QSFP loopback module. Each system used 20Gbps I/O for communication. In the video system the master protocol engines were connected to the video frame buffer and the DisplayPort was connected. The raw maximum bandwidth of DisplayPort was 5.4Gbps/lane, and the DisplayPort was configured to lanes 1, 2, and 4.

B. Memory Access Test

For read access, as shown in Fig. 7, the master protocol engine generates a request packet for the read access (1). The generated packet is transmitted to the slave protocol engine that receives (2) and analyses (3) the read request. And the memory controller executes a memory access using that packet over the DFI interface and the data is transmitted to the master protocol engine (4). When the master protocol engine knows that the read request has been completely transmitted from the read response packet, the response-ACK corresponding to the read response packet is transmitted to the slave protocol engine (5). The memory data is then passed to the processor (6). If the slave protocol engine receives a response-ACK generated by the master protocol engine, both protocol engines complete the read access and then wait for the next access (7).

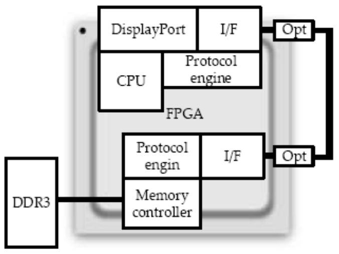
For write access, as shown in Fig. 8, the master protocol engine generates a request packet for the write access (1). The generated packet is transmitted to the slave protocol engine and then delivers a write-ACK packet (slave protocol engine) to the master protocol engine that will inform it that a normal packet has been received (2). The slave protocol engine delivers this data to the memory controller (3). And the memory controller executes a memory access using that data on the DFI interface (4). If the master protocol engine receives a write-ACK generated by the slave protocol engine, both protocol engines complete the write access and then wait for the next access (5).

C. Retransmission of Protocol Engines

We can see the retransmission sequence shown in Fig. 9. The slave protocol engine sends the memory data, corresponding to the read access request of the master protocol engine (1), to the master protocol engine with the sequence numbers, 0xF3, 0xF4, 0xF5, 0xF6 (1)-1. However, if the master protocol engine doesn't receive the memory data, it sends NAK to the slave protocol engine. Then the slave protocol engine receives NAK (2) and re-transmits the transmitted data in the retry buffer (3). We can see the same sequence number as (1) at (4)-1. After that, the slave protocol engine receives ACK from the master protocol engine (4).

D. Video System Test

We illustrated the process of the read request between the master and slave protocol engine based on XGMII clock as shown in Fig. Fig. 10, and we can see the waveforms of read access in the video system as shown in Fig. 11. This waveform is a part of the video operation to read and write the memory and is one horizontal synch in duration. The master protocol engine is requests read access (A). This access comes from a video frame buffer to the output video transceiver. The master protocol engine generates and transmits the packet for read access (B) along with the ACK packets for the previous data packet. The latency of transceiver used in this video system is about 36 clocks. The slave protocol engine receives the packet transmitted by the master protocol engine, and then generates the ACK packet for the read access packets and transmits them (C). In addition, the slave protocol engine accesses the memory from the read data. This occurs before sending the ACK packets. A data packet from the slave protocol engine takes as long as 36 clocks (D). After the master protocol engine receives memory data, ACK packets are sent (E). If the slave protocol engine gets the ACK packets from the master protocol engine, the read request process ends.



(a) Testing system diagram.

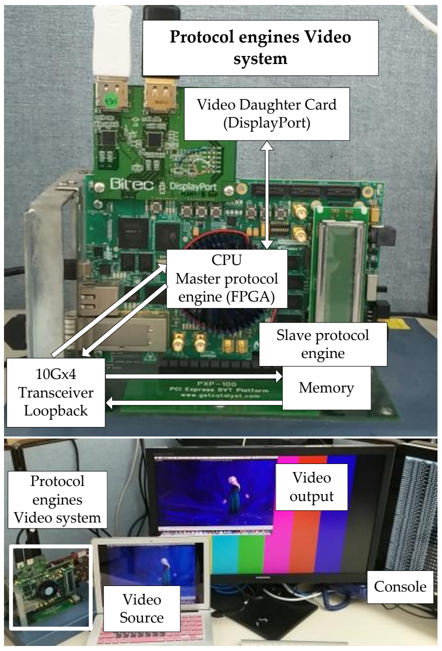
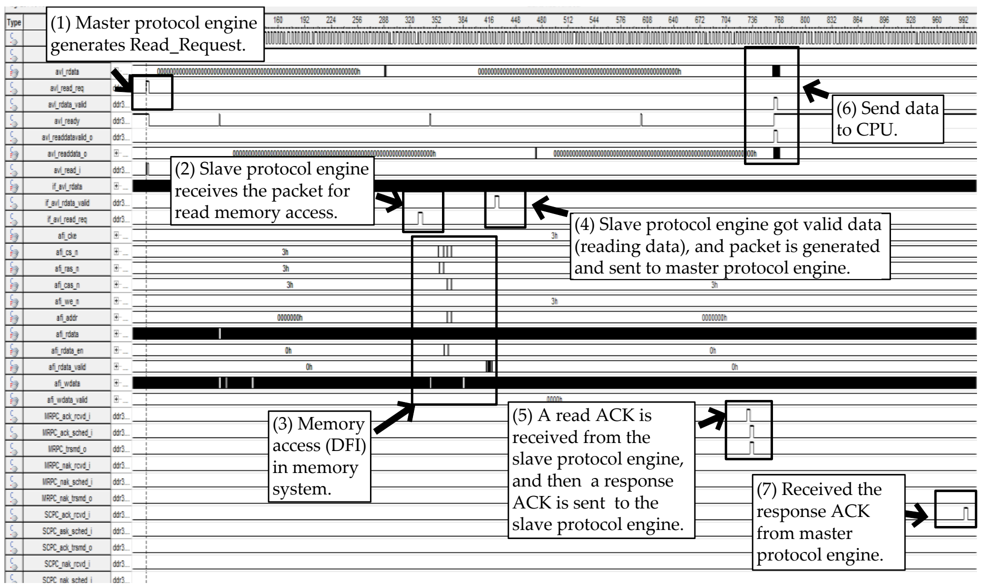

Protocol engines Video system
Video Daughter Card (DisplayPort)
CPU Master protocol engine (FPGA)
Slave protocol engine
10Gx4 Transceiver Loopback
Memory
Protocol engines Video system
Video output
Video Source
Console
(b) Testing board setup.

Fig. 6. Test system diagram and board setup.


(1) Master protocol engine generates Read_Request.
(2) Slave protocol engine receives the packet for read memory access.
(3) Memory access (DFI) in memory system.
(4) Slave protocol engine got valid data (reading data), and packet is generated and sent to master protocol engine.
(5) A read ACK is received from the slave protocol engine, and then a response ACK is sent to the slave protocol engine.
(6) Send data to CPU.
(7) Received the response ACK from master protocol engine.
Fig. 7. Captured read access waveforms of Signal-Tap.

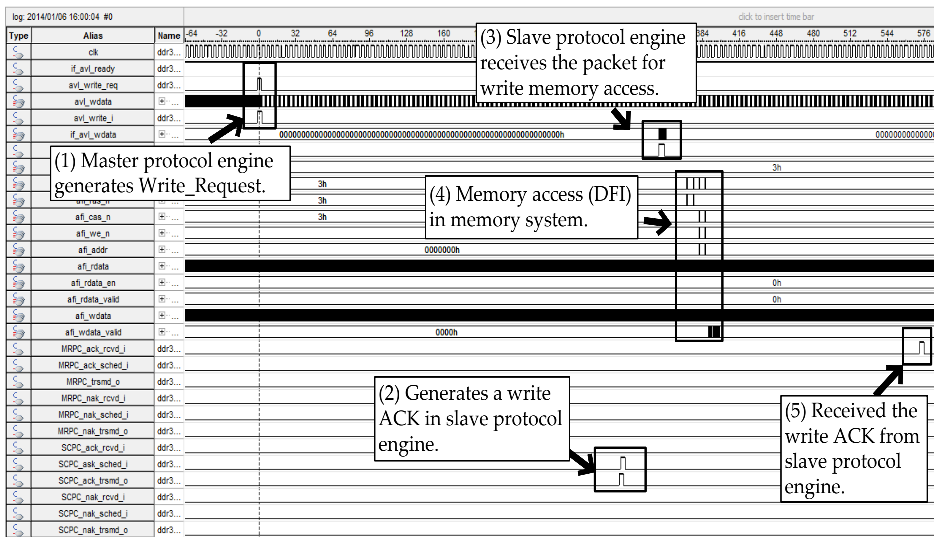

(3) Slave protocol engine receives the packet for write memory access.
(1) Master protocol engine generates Write_Request.
(4) Memory access (DFI) in memory system.
(2) Generates a write ACK in slave protocol engine.
(5) Received the write ACK from slave protocol engine.

Fig. 8. Captured write access waveforms of Signal-Tap.

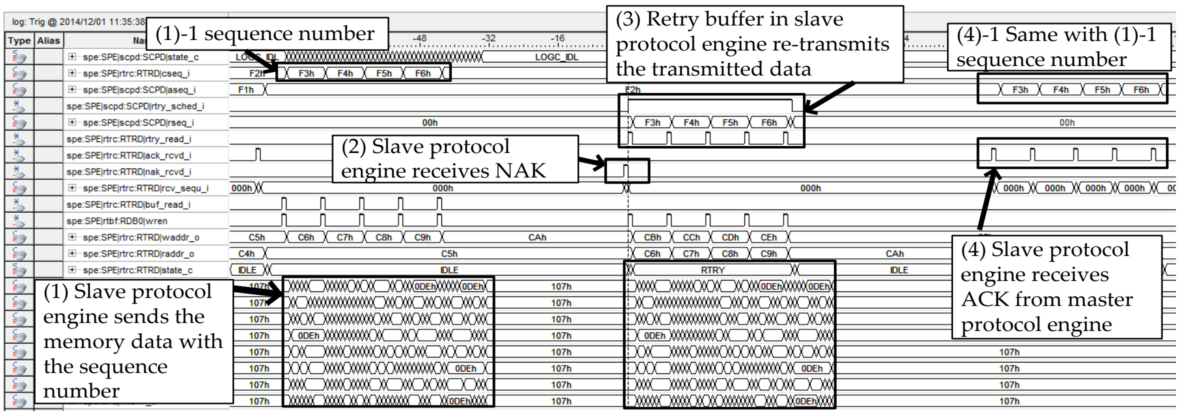

(1)-1 sequence number
(3) Retry buffer in slave protocol engine re-transmits the transmitted data
(4)-1 Same with (1)-1 sequence number
(2) Slave protocol engine receives NAK
(4) Slave protocol engine receives ACK from master protocol engine
(1) Slave protocol engine sends the memory data with the sequence number

Fig. 9. Captured retransmission waveforms in slave protocol engine.

As shown in Fig. 10, we can see more detailed timing in the video system. Furthermore, the memory reading process time for one horizontal synch is approximately 10μs in total, and the operating clock frequency is 156.25MHz, 6.4ns. This system reads 520-byte from memory. We can also ideally read about 1577-byte for 10us as in Eq. (1). In this case, the utilization of accessing time for data can be estimated as similar to Eq. (2), at about 33%. TAB is total of bytes in read access time. ORT is one reading access time. OBT is one byte time. The total bandwidth of one read access is not the bandwidth of the memory but that of the transceiver.

$$\mathrm{TBA} = \frac{\mathrm{ORT}}{\mathrm{OBT}} = \frac{10.09\mu\mathrm{s}}{6.4\mathrm{ns/byte}} \cong 1577\ \mathrm{byte} \tag{1}$$

$$\mathrm{AU} = \frac{\mathrm{TB}}{\mathrm{TAB}} = 520\ \mathrm{byte}/1577\ \mathrm{byte} \cong 0.3297 \tag{2}$$

However, the latency of the transceiver using this system is about 36 clocks (based on XGMII clocks) and the protocol also has redundancy elements like acknowledgment packets. If there are not any redundancies in reading access from memory, the bandwidth of the data can be calculated as Eq. (3), at 8.125Gbps. Therefore, the real bandwidth of the data is 2.68Gbps as in Eq. (4). AU is accessing utilization, TB is total of bytes reading from memory. TBR is total bandwidth of one reading access, TbitM is total of bits reading from memory, FRQ is operating clock frequency. SB is the support bandwidth.

$$\mathrm{TBR} = \frac{\mathrm{TbitM}}{1/\mathrm{FRQ}} = 520\ \mathrm{bit}\,/\left(\frac{1}{156.25\mathrm{MHz}}\right) = 81{,}250\mathrm{Mbps} = 8.125\mathrm{Gbps} \tag{3}$$

$$\mathrm{SB}=\mathrm{TBR}\times\mathrm{AU}=8.125\mathrm{Gbps}\times 0.3297 \cong 2.6788\mathrm{Gbps} \tag{4}$$

To support the maximum bandwidth of the DisplayPort, 21.6Gbps (5.2Gbps/lane × 4 lane), this system has to reduce the latency of the transceiver and to increase accessing utilization through more bandwidth from the transceiver. If this system uses a transceiver that doesn't have a media access control layer (MAC), and instead its uses a 40Gbps transceiver to access memory data, these factors could be reduced and increased. The latency of the transceiver using this system is about 36 clocks (based on a XGMII clock). If the MAC layer is omitted, the latency is about 10 clocks (based on a XGMII clock). As shown in Fig. 10, we can reduce the duration time until the read valid signal is 56 clocks (base on a XGMII clock).

$$\mathrm{TBA} = \frac{\mathrm{ORT}}{\mathrm{OBT}} = \frac{4.6592\mu\mathrm{s}}{6.4\mathrm{ns/byte}} \cong 728\ \mathrm{byte} \tag{5}$$

$$\mathrm{AU} = \frac{\mathrm{TB}}{\mathrm{TAB}} = 4 \times 520\ \mathrm{byte}/728\ \mathrm{byte} \cong 2.857 \tag{6}$$

$$\mathrm{SB} = \mathrm{TBR} \times \mathrm{AU} = 8.125\mathrm{Gbps} \times 2.857 \cong 23.2143\mathrm{Gbps} \tag{7}$$

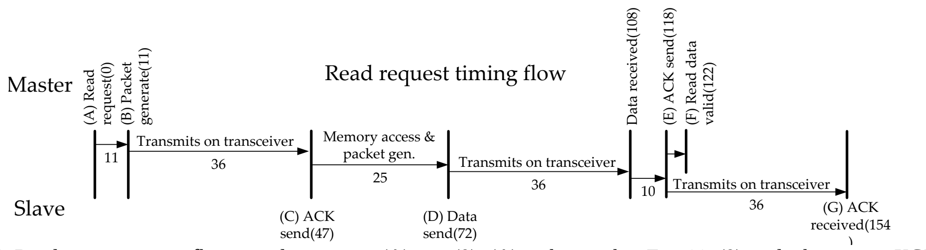


Fig. 10. Read request timing flow in video system, (A) xxx (0): (A) is the number Fig. 11, (0) is clock count in XGMII.

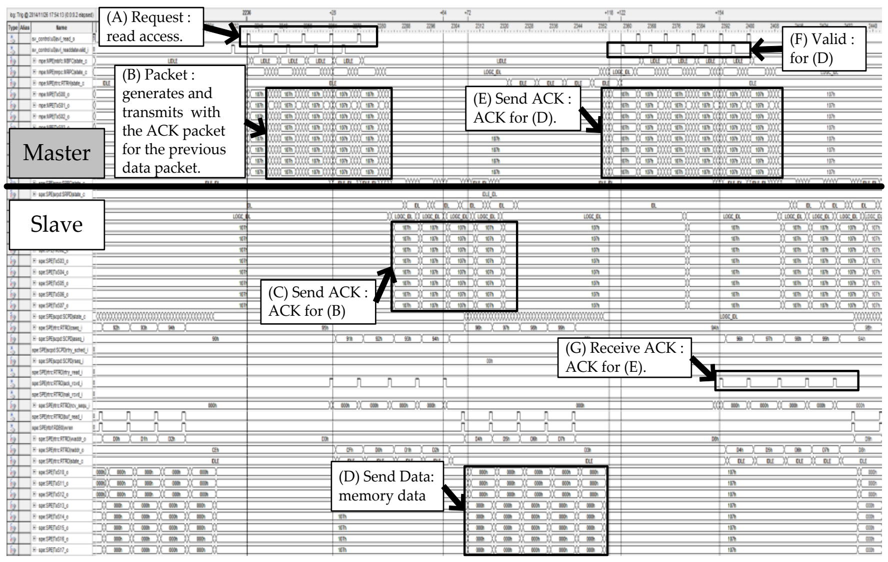

(A) Request : read access.
(B) Packet : generates and transmits with the ACK packet for the previous data packet.
Master
Slave
(C) Send ACK : ACK for (B)
(D) Send Data: memory data
(E) Send ACK : ACK for (D).
(F) Valid : for (D)
(G) Receive ACK : ACK for (E).

Fig. 11. Captured read request waveform in video system of Signal-Tap, sampled by XGMII clock, 156.2 MHz.

V. Conclusion

In this paper, we designed and tested the protocol in the proposed structure, and configured a video frame buffer and memory system connected by optical fibers. The video system included the master protocol engine and the video buffer system including the processor, and the memory system was composed of the slave protocol engine and the memory controller. We adapted to the video system to test the feasibility of the protocol engine's function. The transceiver used in this paper had a MAC layer. The latency may be caused by the MAC layer for processing data but this system didn't need. Because the protocol engines also played a role of MAC. If we use other transceiver without MAC, the accessing utilization is higher.

Acknowledgment

This work was supported by the ICT R&D program of MSIP/IITP, Korea. [10038764, Silicon Nano Photonics Based Next Generation Computer Interface Platform Technology].

References

[1] A. Okazaki and Y. Katayama, "Optical interconnect opportunities for future server memory systems," in *Proc. HPCA*, 2007, pp. 46-50.

[2] L. Li, S. Y. Liu, M. Y. Chen, and J. P. Fan, "Grid memory service architecture for high perfomance computing," in *Proc. Seventh International Conference on Grid and Cooperative Computing*, 2008, pp. 24-26.

[3] C. Batten, A. Joshi, J. Orcutt, A. Khilo *et al*., "Building many-core processor-to-dram networks with monolithic cmos silicon photonics," in *Proc. HOTI'08*, 2009, pp. 21-30.

[4] Y. W. Yin, R. Proietti, X. H. Ye, S. J. B. Yoo, and V. Akella, "Experimental demonstration of optical processor-memory interconnection," *Advanced Intelligence and Awarenss Internet*, pp. 23-25, 2010.

[5] A. Hadke, T. Benavides, R. Amirtharajah, M. Farrens, and V. Akella, "Desing and evaluation of an optical CPU-DRAM interconnect," in *Proc. ICCD*, 2008, pp. 492-497.

[6] C. P. Lai, C. Ware, and D. Brunina, "Building data centers with optically connected memory," *Journal of Optical Society of America*, vol. 3, issue 8, pp. A40-A48, 2011.

[7] H. M. C. Consortium, "Hybrid memory cube specificatiion 1.0," 2013.

[8] U. Kang, "8Gb 3D DDR3 DRAM using through-Silicon-Via technology," in *Proc. ISSCC*, 2009.

[9] W. S. Lee, "A study on the effectiveness of underfill in the high bandwidth memory with TSV," in *Proc. IMAPS*, 2013.

[10] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," *IEEE Transl. J. Magn. Japan*, vol. 2, pp. 740-741, August 1987.

[11] Bit-tech. Intel sandy bridge: Details of the next gen. [Online]. Available: http://www.bit-tech.net/hardware/cpus/2010/04/21/intel-sandy-bridge-details-of-the-next-gen/1

[12] JEDEC, *DDR4 SDRAM JESD79-4*, September 2012.

[13] Rambus, "Challenges and solutions for future main memory," White paper, May 26, 2009.

[14] H. T. Jun, B. Chelepalli, N. Xue, and B. C. Lee, "Disintegrated cControl for energy-efficient and heterogeneous memory systems," in *Proc. IEEE 19th International Symposium on High Performance Computer Architecture*, 2013.

Hyukje Kwon received his MS degree in electronics engineering in Chonbuk National Univerisity, Korea in 1997. And in 2008, he received PhD degree in the same university. In 2012, he joined the Server Platform Research team at the Electronics and Telecommunications Research Institute (ETRI), Rep. of Korea.

Yongseok Choi joined the Server Platform Research team at the Electronics and Telecommunications Research Institute (ETRI), Rep. of Korea in 2000.