IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 19, 2019, accepted February 27, 2019, date of publication March 11, 2019, date of current version March 29, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2904083

Software Implementation of 10G-EPON Downstream Physical-Layer Processing Adopting CPU-GPU Cooperative Computing for Flexible Access Systems

TAKAHIRO SUZUKI, SANG-YUEP KIM, JUN-ICHI KANI, AND JUN TERADA
NTT Access Network Service Systems Laboratories, NTT Corporation, Yokosuka 239-0847, Japan
Corresponding author: Takahiro Suzuki (suzuki.takahiro@lab.ntt.co.jp)

ABSTRACT The application of network function virtualization (NFV) and software-defined networks (SDNs) to optical access systems continues to attract a lot of attention. Their use on general-purpose hardware allows for cost-effective implementation and quick response to functional requirements. This paper demonstrates the softwarization of the complete downstream physical (PHY)-layer functions of the optical line terminal (OLT), including a scrambler, which uses a serial processing algorithm that cannot be parallelized by a general-purpose graphics processing unit (GPU). We propose CPU-GPU cooperative implementation architecture that softwarizes the complete 10G-EPON OLT downstream PHY. We achieve 10.3125-Gbps real-time performance through experiments for the first time.

INDEX TERMS SDN, NFV, access networks, GPU, PHY coding and scrambler.

I. INTRODUCTION

Network function virtualization (NFV) and software defined networks (SDNs) are highly attractive as they support the softwarization of network functions making flexible function replacement and cost-effective implementation of network equipment possible. The application of NFV/SDN to optical access systems is seen as an urgent necessity.

For optical access systems, composing optical line terminals (OLTs) on commercially-available general-purpose hardware and software [1], [2], such as white-box switches and open-source software, has been considered. Our contribution is the flexible access system architecture (FASA) [3]. FASA modularizes OLTs into software functions, a general-purpose server, and external modules. FASA combines these components to realize a wide range of services and can support the standardized 10G-EPON [4], NG-PON2 [5] and XGS-PON [6]. Some related works on FASA have examined the softwarization of PON systems [7], [8]. This approach focuses on the upper-layer processing, and the external module with the dedicated hardware is utilized for PHY processing. Other works also focus on the upper-layer processing, and the utilization of open-specification hardware and open network operating system (ONOS) provides partial flexibility to OLT implementation [9], [10]. However, no study has described the realization of fully flexible OLTs. The final goal of FASA is to fully softwarize OLT functions including physical-layer (PHY) processing as this will enhance the merits of SDN/NFV, which include maximizing system programmability and eliminating the high costs imposed by redesigning and fabricating PHY hardware.

The softwarization of all PHY processing is challenging since PHY processes are computationally demanding and general-purpose hardware is inferior to the application specific integrated circuits (ASICs) used by dedicated hardware. Some other works [11]–[13] try to softwarize PHY processing for wireless transmission. They focus on a single function such as error correction, and do not have external input / output ports for real data streams. Given that their achieved throughput is Mbps class, the results are not valid for access systems requiring more than Gbps throughput. On the other hand, we have demonstrated real-time 10-Gbps throughput for single PHY processing on general-purpose graphic processing units (GPUs), which have large numbers of computational cores and thus well support data parallelism [14], [15]. The software implementation of forward

The associate editor coordinating the review of this manuscript and approving it for publication was Walid Al-Hussaibi.

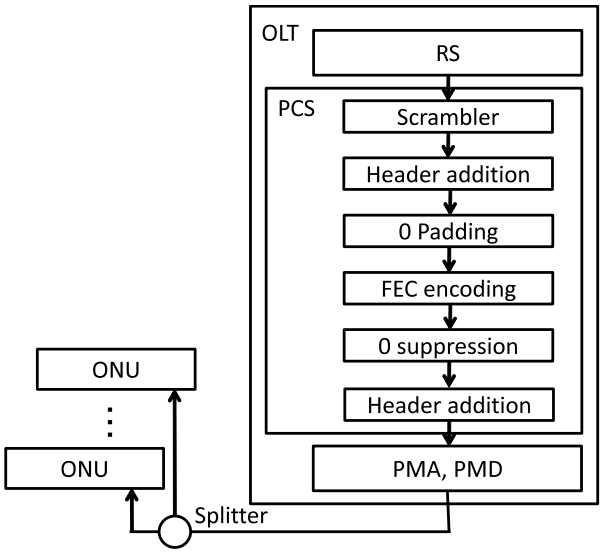


FIGURE 1. Downstream PON system and OLT PHY coding. (RS: reconciliation sublayer, PCS: physical coding sublayer, PMA: physical medium attachment and PMD: physical medium dependent).

error correction (FEC) has been achieved for encoding [16] and decoding [17] respectively. The burst-frame synchronization for upstream signal was demonstrated using a novel low-complexity algorithm [18] and OLT performance was analyzed when our synchronization proposal was combined with FEC [19]. We have demonstrated the softwarization of upstream OLT PHY using these techniques [20]. Given that PON upstream PHY functions are composed of just non-serial processing algorithms, they were implemented on just the GPU. However, PON downstream PHY functions consist of serial and non-serial processing algorithms. In particular, serial processing algorithms significantly hinder the full softwarization of the upstream and downstream PON PHY.

This paper proposes CPU-GPU cooperative computing to softwarize the complete OLT-PHY of downstream 10G-EPON. Since CPUs are faster than GPUs for serial PHY functions in general, our approach is to switch between CPU and GPU depending on the function needed. Our proposal includes a novel scheme to control data transfer and GPU kernel, including a novel implementation method. We softwarize the complete OLT-PHY functions of downstream 10G-EPON for the first time. This paper is organized as follows. Section *II* details downstream PHY functions in PON. Section *III* describes our control scheme and implementation method, which are key advances in the CPU-GPU cooperative architecture. Section *IV* introduces the optimum processor allocation of PHY functions. Section V demonstrates the complete downstream OLT-PHY as softwarized by our proposal. Finally, Section *VI* concludes this paper.

II. PHY PROCESSING IN DOWNSTREAM PON

Figure 1 shows the downstream PON circuit. An OLT is connected to multiple optical network units (ONUs) and continuous signals are sent downstream from OLT to ONUs. The 10G-EPON function layer [4] is divided into physical medium attachment (PMA), physical medium dependent (PMD), physical coding sublayer (PCS), and reconciliation sublayer (RS). PCS, which is our focus, is mainly composed of FEC encoding, 64B66B header addition and scrambling, all of which are PHY coding functions. FASA can use and replace PMA and PMD with the standard pluggable external module.

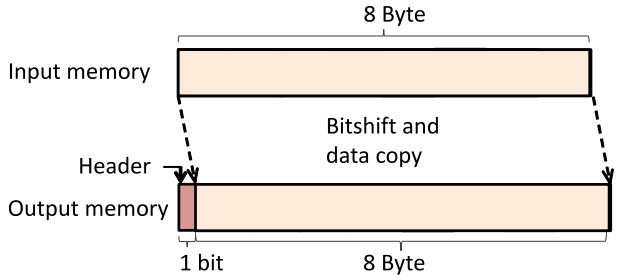


FIGURE 2. 64B66B Header addition.

For FEC encoding, Reed-Solomon (255, 223) [21] is standardized in 10G-EPON. It inserts redundancy into data sequences in the transmitter side to allow errors to be corrected in the receiver side.

64B66B header addition inserts 2 bit headers into the data sequence on an 8-Byte cycle (64 bits). This adds 1-bit header two times as shown in Fig. 1. The 1-bit header addition involves header input, bitshift, and data copy as shown in Fig. 2. The data in input memory is copied to output memory while shifting the data by 1 bit every 8 Bytes and adding a 1-bit header.

Scrambling converts the input sequence into a random sequence for clock recovery. In 10G-EPON, the scrambler polynomial is written as

$$G(x) = 1 + x^{39} + x^{58}. \quad (1)$$

Scrambling is specified in the IEEE 802.3 standard [22], and is executed continuously on all bits. The process is shown in Fig. 3. The algorithm holds 57 bit blocks of the serial input. The exclusive OR (XOR) operation is performed on the 38-bit delayed data, $S38$, and the 57-bit delayed data, $S57$. The calculation is performed serially and repeatedly. Given that this is a serial processing algorithm, its parallel implementation on one or more GPUs is impossible.

In order to softwarize the complete OLT-PHY function including such serial processing algorithms, our approach is to cooperatively use CPU and GPU.

III. CPU-GPU COOPERATIVE IMPLEMENTATION

We propose a CPU-GPU cooperative architecture enabling high-speed execution for complete PON downstream OLT-PHY coding.

A. SERVER ARCHITECTURE WITH UNITARY-OPERATION PROCESSORS

To explain our CPU-GPU cooperative architecture, this subsection reviews the basic architecture for unitary processors.

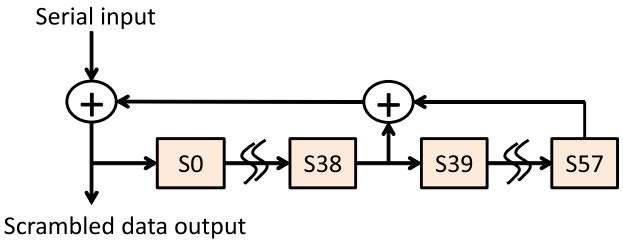


FIGURE 3. Scrambler.

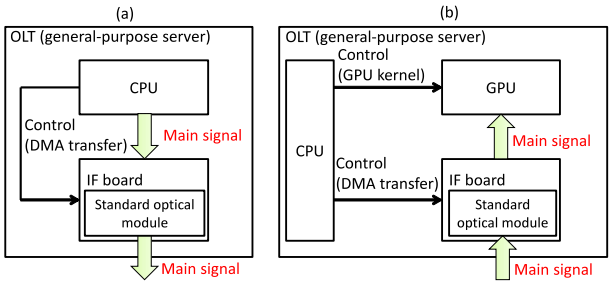


FIGURE 4. Server architecture with unitary-operation processors. (a) CPU-oriented system and (b) GPU-oriented system.

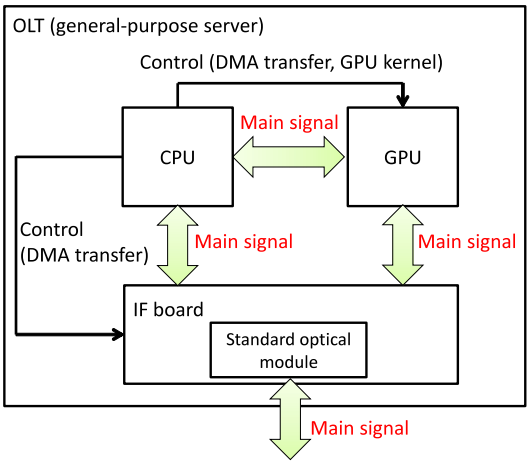


FIGURE 5. Proposed CPU-GPU cooperative architecture to softwarize OLT.

Figure 4 shows two architectures used for the softwarization of FEC encoding and decoding for PON upstream and downstream.

Figure 4 (a) shows the CPU-oriented system [16] in which the CPU is connected to the IF board via PCI express (PCIe). In order to continuously transfer the main signal processed by the CPU to the IF board, the CPU controls and manages the direct memory access (DMA) transfer. Figure 4 (b) shows to the GPU-oriented system of CPU, GPU and IF board [17]. Direct data transfer from IF board to GPU without CPU intervention is essential to fully utilize the GPU's superior calculation performance. Our system uses two kinds of control: DMA transfer and GPU kernel. CPU controls and manages the DMA to transfer the main signal from IF board to GPU, while CPU controls the execution of GPU kernel.

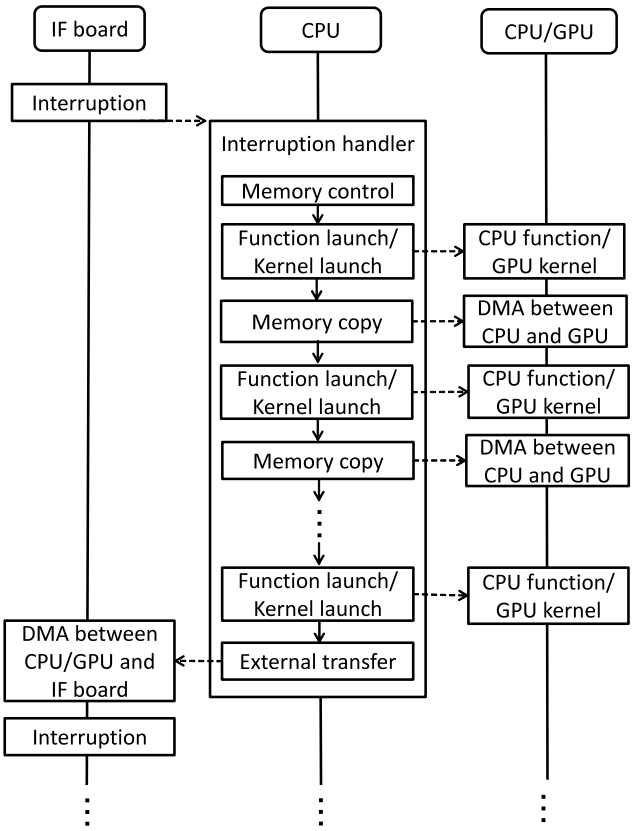


FIGURE 6. Scheme to control data transfer and GPU kernel.

B. SCHEME TO CONTROL DMA TRANSFER AND GPU KERNEL IN CPU-GPU COOPERATIVE ARCHITECTURE

The proposed CPU-GPU cooperative server architecture is shown in Fig. 5. The server is mainly composed of CPU, GPU and IF board. Our architecture is realized by extending the two controls mentioned above.

In our proposal, PHY functions involving serial processing algorithms are performed on CPU; the other PHY functions are parallelized on GPU. Thus, it is necessary to transfer the main signal between CPU and GPU as well as between CPU/GPU and IF board. In order to realize this, the proposal sends control signals from CPU to GPU to manage GPU kernel execution, as well as DMA transfer between CPU and GPU. In addition, control signals are sent from CPU to IF board to manage DMA transfer between CPU/GPU and IF board.

Figure 6 shows the scheme that controls DMA transfer and GPU kernel. IF board sends an interrupt signal to CPU when input or output signal is ready. The CPU generates an interrupt handler corresponding to the interrupt signal. The interrupt handler invokes memory control to specify the addresses of where the data is input to/output from the different processors and the data input and output from and to the IF board. The memory addresses for input and output are determined so that the least recently used (LRU) address is selected [17]. The kernel launch or function launch is executed. The kernel launch is for GPU kernel execution

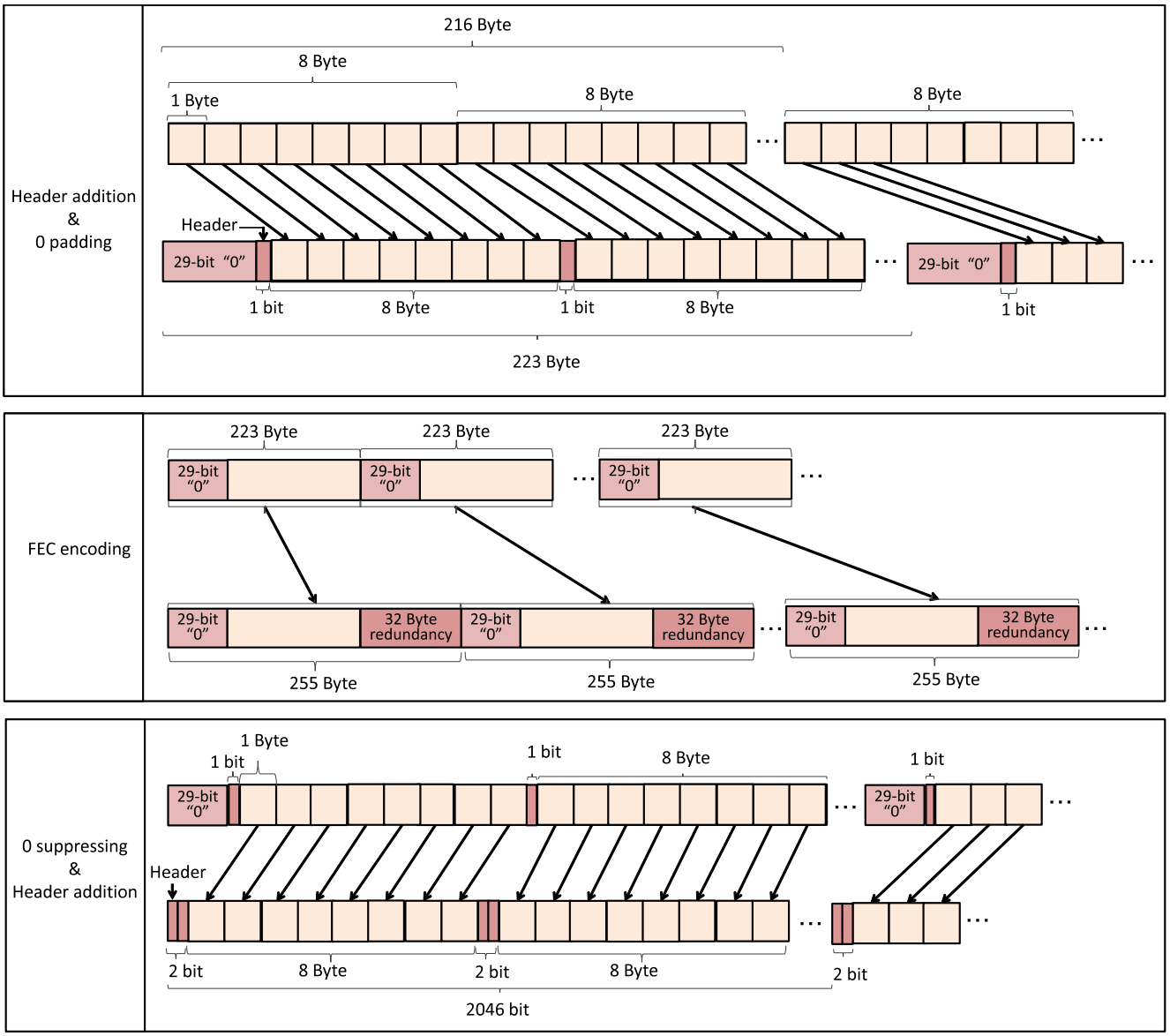


FIGURE 7. Parallel implementation method on GPU.

while the function launch is for CPU execution. Memory copy transfers the data having the specified memory address from CPU to GPU or from GPU to CPU. The function/kernel launch and memory copy are repeated until all PHY processing is completed. The external transfer triggers the startup of the DMA of the IF board; the data is transferred between CPU or GPU and IF board as needed.

C. IMPLEMENTATION METHOD OF 10G-EPON OLT-PHY FUNCTION

As shown in Fig. 7, we utilize data parallelism for functions that have non-serial processing algorithms and execute them on GPU in parallel. In the PCS of 10G-EPON, data is input in multiples of 216 Bytes, and output in multiples of 2046 bits due to the addition of 64B66B header and FEC redundancy. These PHY functions should be run on the CPU and GPU on a data type basis. The minimum unit is 1 Byte (i.e. unsigned character data type).

The serial processing algorithm for scrambling is serially implemented on CPU in units of 1 Byte (see. Fig. 3). Through header addition and 0 padding, the 1-bit 64B66B header is added in units of 8 Bytes and 29-bits of "0" are padded to each input 216-Byte block. This uses 216 bitshifts and data copies for the given input 216-Byte data. These bitshifts and data copies are parallelized in units of 1 Byte. FEC encoding is parallelized in the coding unit (i.e. 223 Byte) [16], [17]. In 0 suppression and header addition, the 1-bit 64B66B header is added in units of 8 Bytes and 29-bits of "0" are dropped to yield FEC encoded 248-Byte blocks without padding "0" and header. This uses 248 bitshifts and data copies for the given FEC encoded 248-Byte data. These bitshifts and data copies are also parallelized in units of 1 Byte.

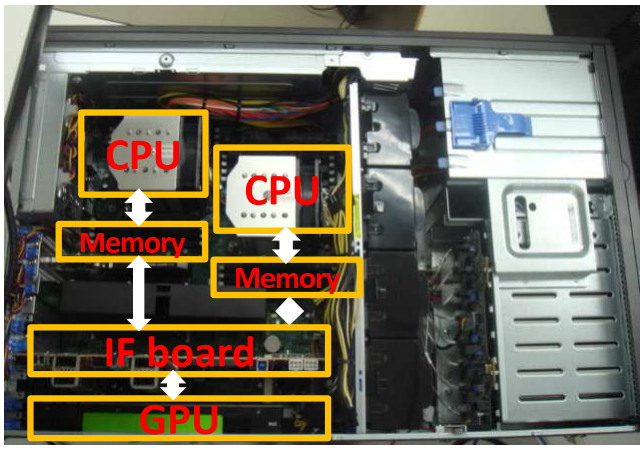


FIGURE 8. Configuration of the server.

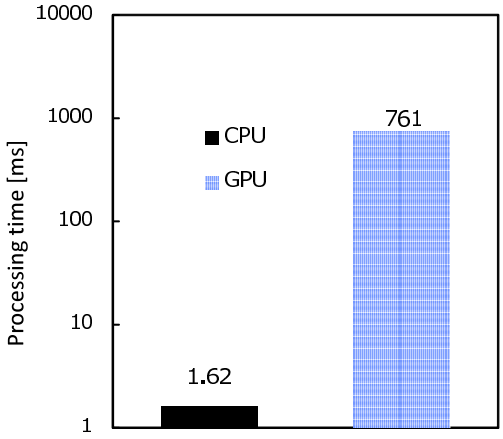


FIGURE 9. Scrambler processing time on CPU and GPU. Scrambling was executed using a single thread.

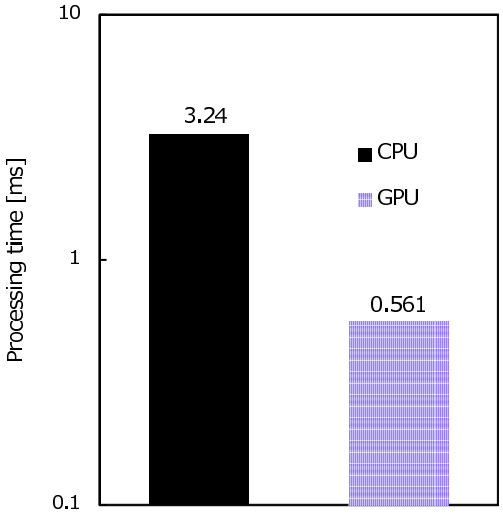


FIGURE 10. The processing times for header addition, 0 padding and 0 suppression on CPU and GPU. All were implemented in parallel on each processor.

IV. OPTIMUM PROCESSOR ALLOCATION OF PHY FUNCTIONS

To allocate 10G-EPON PHY functions to the appropriate processor (i.e. CPU or GPU), we measured the processing times of each function on each processor.

A. EVALUATION ENVIRONMENT

Figure 8 shows the main components of CPU, GPU and IF circuit. The two 2.20 GHz CPUs (Intel Broadwell Xeon E5-2699v4, 22 core, 44 thread) of the server accessed 128 GB of DDR4 memory. GPU (NVIDIA Tesla P100), with 3584 CUDA cores, was run at 1328 MHz using turbo-boost technology. The GPU board included 16 GB of high bandwidth memory (HBM) 2 and was connected to the server via PCIe gen3 x16. As the IF circuit, we utilized Tokyo Electron Device TB-7VX-690T-PCIEXP; it was connected to the server via PCIe gen3 x8. PCIe gen3 x16 throughput is 15.8 GB/s while that of PCIe gen3 x8 is 7.88 GB/s, with the exception for 128B/130B line coding. The small form-factor pluggable (SFP)+ optical module connected to the IF circuit generated 10.3125-Gbps signals. A data stream was generated using a $2^{23} - 1$ pseudo-random bit sequence (PRBS).

B. ANALYSIS OF PROCESSING TIME PERFORMANCE

We measured the processing time of the three PHY functions composing the downstream PCS of 10G-EPON (i.e., scrambling, header addition and FEC encoding) on the CPU and GPU.

Fig. 9 shows scrambler processing times. Given that scrambling involves serial processing, it was processed by a single thread on the CPU and GPU. The processing time was 1.62 ms and 761 ms for CPU and GPU, respectively. The excellent result of CPU originates from the two reasons given; the scrambler involves serial processing and the CPU offers higher processing capability for a single thread than GPU. CPU has higher clock frequency and its architecture is optimized for serial functions, such as branch prediction and super scalar arithmetic.

Next, we evaluated the functions that can be parallelized. In order to realize parallel implementation on CPU, Intel Threading Building Blocks (TBB) was used. As shown in Fig. 10, the processing time for header addition, 0 padding and 0 suppression was 3.24 ms and 0.561 ms for CPU and GPU, respectively. Fig. 11 shows the processing time for FEC, Reed Solomon ($n = 255$, k) encoding in this paper. For all redundancies, defined by $n - k$, GPU was faster than CPU. For Reed Solomon (255, 223), i.e. redundancy of 32, specified in the 10G-EPON standard, the processing time was 6.57 ms and 1.30 ms for CPU and GPU, respectively. These results prove that the GPU's many cores can maximize the processing capability.

In order to softwarize the complete PCS of 10G-EPON, we must consider the total of the processing times required for the aforementioned PCS functions. The total value should be within the time constraint, which is a primary factor limiting softwarization. The time constraint is given by 3.25 ms; for the throughput target of 10.3125 Gbps, the buffer data of 4.19 MByte, determined by our hardware setup, should

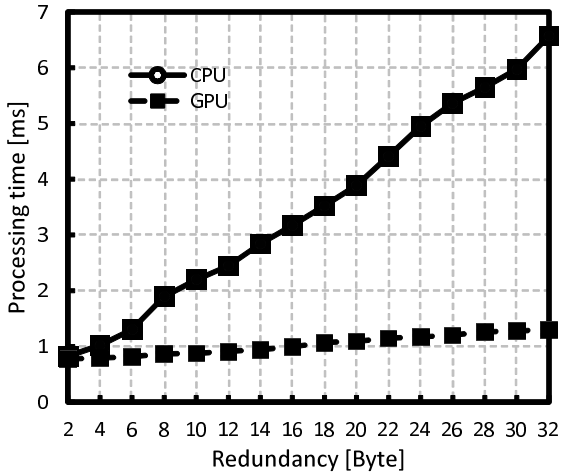


FIGURE 11. The processing times for FEC encoding on CPU and GPU. This function was implemented in parallel on each processor.

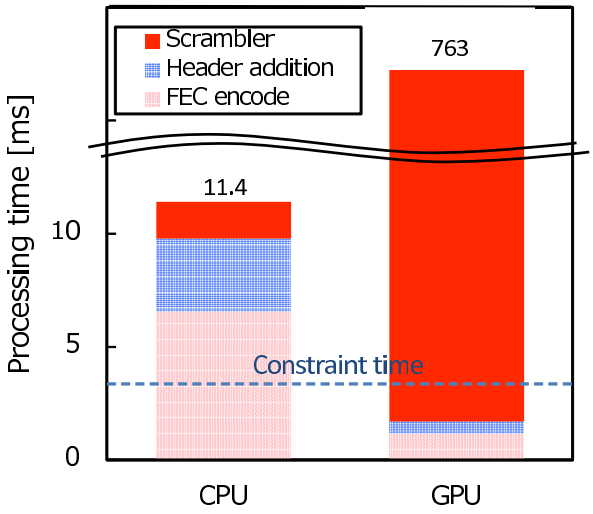


FIGURE 12. Summed processing time of PCS function achieved by a single type processor.

be processed within the time constraint. Fig. 12 shows the summation of each processing time achieved by a single type processor, i.e. CPU and GPU (see Fig. 9, 10 and 11). We can clearly see that the use of just a single type of processor cannot match the time constraint, i.e. 11.4 ms (i.e. 1.62 + 3.24 + 6.57) and 763 ms (i.e. 761 + 0.561 + 1.30) for CPU and GPU, respectively. For the CPU, the processing times of FEC encode of RS(255,223) and header addition exceeds the constraint time individually, while the scrambler processing time exceeds the constraint time for the GPU case. In our related work [20], the upstream PCS of PON (i.e. FEC decoding, PON frame synchronization, descrambler and header suppression) was implemented on just the GPU. Given its total processing time was within 2.55 ms, the softwarization of downstream PCS is significantly more challenging than that of upstream PCS.

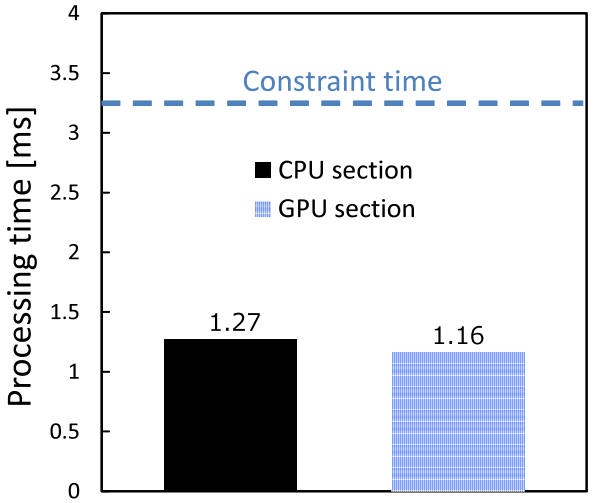


FIGURE 13. Processing time measured when all functions of 10 G-EPON is optimally allocated.

Figure 12 also shows that each PCS function has different processing time for processor type; the scrambler has smaller processing time on CPU than GPU. Header addition and FEC encode have smaller processing times on GPU than CPU. Our approach is to utilize CPU and GPU cooperatively, and we allocate scrambler to CPU while allocating the header addition, 0 padding, 0 suppression and FEC encoding to GPU. This is the best design to handle all PCS functions.

C. PROCESSOR ALLOCATION OF 10G-EPON DOWNSTREAM PHY FUNCTIONS

When all 10G-EPON PHY functions were optimally allocated and implemented on the proposed CPU-GPU cooperative architecture, Fig. 13 shows the processing time measured for the CPU section and the GPU section, and includes the data transfer section. This result shows that with cooperation the processing times meet the constraint time for 10.3125 Gbps; the processing time was 1.27 ms and 1.16 ms for scrambler and FEC/header addition, respectively. The results show slightly better performance, compared to that of the single PHY softwarization mentioned in subsection B (i.e. 1.62 ms for scrambler on CPU and 1.86 ms (0.561 + 1.30) for FEC/header addition on GPU). The reason may be because the cache hit ratio is improved by repeating the same processing in the given experimental setup where the signal flow is continuous.

All 10G-EPON PHY functions were optimally allocated and implemented on the proposed CPU-GPU cooperative architecture as shown in Fig. 14. The input (i.e. d_input) and output GPU memory regions (i.e. d_output) were allocated in advance to support the continuous input and output of the main signal. Memory assignment created N regions, each was L Bytes long. First, memory control is performed in interruption handler. Function launch activates the function that includes scrambler for the main signal which is held in CPU h_output. After that, the scrambled signal is transferred to GPU memory by memory copy. When the GPU kernel is

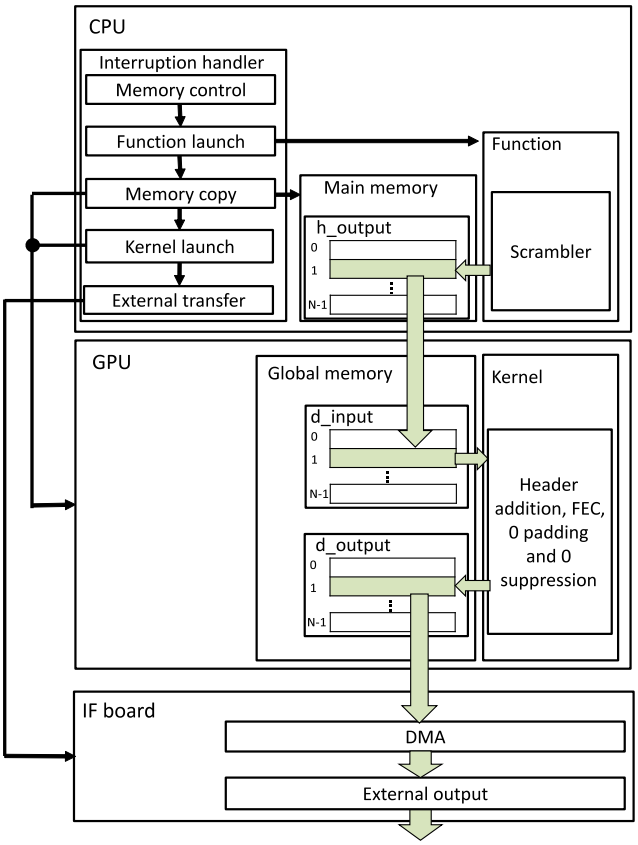


FIGURE 14. Complete 10G-EPON PHY implemented on CPU-GPU cooperative architecture. Thick arrows show the main signal and thin arrows show the control flows.

launched, header addition, FEC, 0 padding and 0 suppression are executed for d_input and the encoded data is written into d_output. Finally, IF board directly receives the encoded data from GPU and outputs the data to the SFP+ optical module.

Algorithm 1 Control Scheme of the Architecture Shown in Fig. 14

```
// Initialization
i ← 0
// Interruption handler
if DMA of output signal is ready then
    i ← (i + 1) mod N
    Function(h_output[i])
    Memcpy(d_input[i], h_output[i], L)
    GPU_kernel(d_output[i], d_input[i])
    DMA(d_output[i])
end if
```

Algorithm 1 shows details of our implemented control scheme. First, index i for assigning GPU memory addresses is initialized. Index i is incremented up to the maximum number (i.e. N) supported by the buffers, and used by memory control to access the LRU memory addresses in proper order. Function(·) executes scrambling on CPU for h_output[i]. After that, memory copy (i.e. Memcpy) from h_output[i] on CPU to d_input[i] on GPU is executed. For the received d_input[i], GPU kernel is activated and the calculated data is written into d_output[i]. Finally, the external transfer instruction is sent to IF board (i.e. DMA and external output) and executed for d_output[i]. The IF board hosts a DMA engine that executes DMA transfer and its DMA controller. The CPU sequentially assigns the source memory address on the GPU to the DMA controller, and the CPU then sends the instruction, which starts the DMA transfer, from the CPU to the IF board. The DMA transfer copies an area of a constant size of the GPU memory to the memory on the IF board via the PCIe bus. Finally, the data stored in the memory on the IF board is serialized and output to the outside external port. These processes are repeated for each data output timing of the IF board.

V. DEMONSTRATION

To demonstrate the 10.3125-Gbps performance of the implementation proposal shown in Section III and the processor allocation shown in Section IV, we implemented the complete 10G-EPON PHY on a general-purpose server and its real-time performance was measured. In addition, the bit error rate (BER) of the processed signal output by the server was measured.

A. TRANSFER PERFORMANCE

First, we evaluated the transfer performance between each device in the proposed implementation. Transfer throughput is defined by

$$Throughput = \frac{Transfer\ size}{Transfer\ time}. \tag{2}$$

Transfer size is the length transferred by DMA. Transfer time is the time taken to transfer the data having the transfer size. For CPU to GPU transfer, time is estimated from the processing time of "cudaMemcpyAsync" from the CUDA application programming interface (API). For GPU to IF board transfer, time is calculated by counting the number of clocks taken for the transfer at the given PCIe operating frequency in our utilized DMA IP (SYSTEC SYPCIE).

Figure 15 shows the performance of the CPU-GPU cooperative architecture for two transfer routes: "CPU to GPU" and "GPU to IF board". For both routes, the proposal achieved the 10.3125-Gbps throughput required by 10G-EPON PHY. This means that the signals processed in and output the CPU and GPU could be transferred at 10.3125 Gbps with no throughput degradation from CPU to GPU and from GPU to IF board, respectively. For the case of data transfer from GPU to IF board, although the maximum PCIe (gen3x8) bandwidth is 64 Gbps including line coding [23], the achieved throughput of fixed 10.3125 Gbps is reasonable since our DMA controller was designed for the 10.3125-Gbps limit.

B. PROCESSING PERFORMANCE

Figure 16 shows CPU and GPU throughputs confirming complete 10G-EPON PHY softwarization. The throughput is

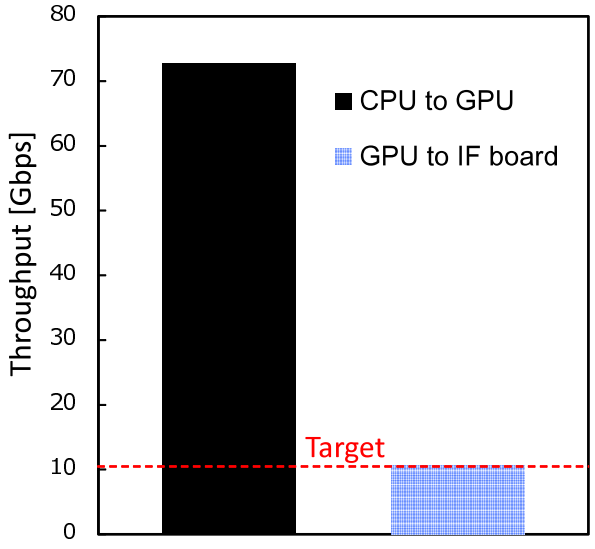


FIGURE 15. Transfer performance for "CPU to GPU" and "GPU to IF board".

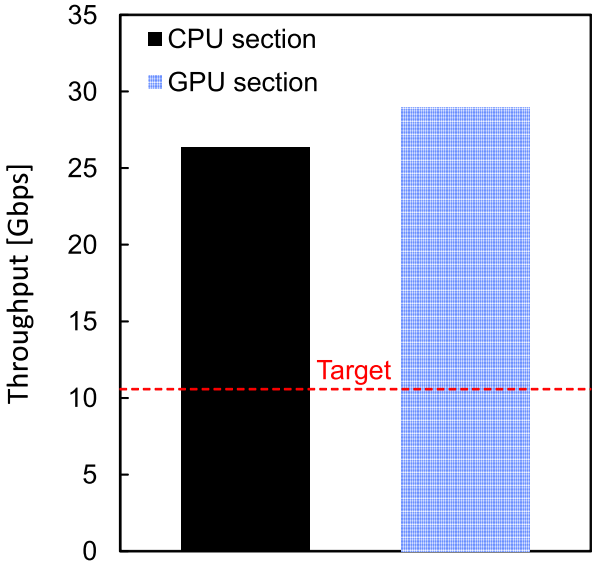


FIGURE 16. Throughput performance of CPU and GPU section when running the complete 10G-EPON PHY.

defined by

$$Throughput = \frac{Processing\ data\ size}{Processing\ time}. \tag{3}$$

When running the complete 10G-EPON PHY, we measured the processing times of scrambling in CPU, and of FEC encoding and header addition, 0 padding and 0 suppression in GPU. As both CPU and GPU achieved the throughput of 10.3125 Gbps, the whole system is capable of processing a 10.3125 Gbps signal.

Figure 17 shows a time profile of GPU processing as measured by the NVIDIA Visual Profiler. All functions are processed within one interrupt cycle of 3.25 ms for 4.19-MByte data, which confirms the realization of 10.3125-Gbps real-time processing (i.e. 8 × 4.19 MByte/3.25 ms).

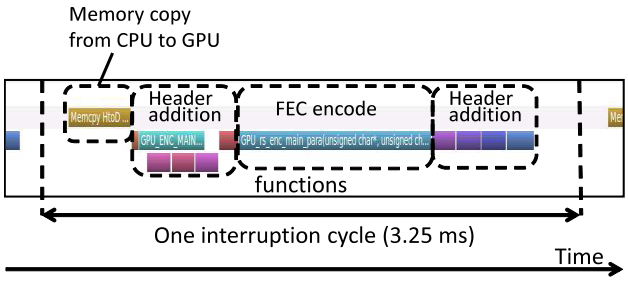


FIGURE 17. Time profile for GPU processing.

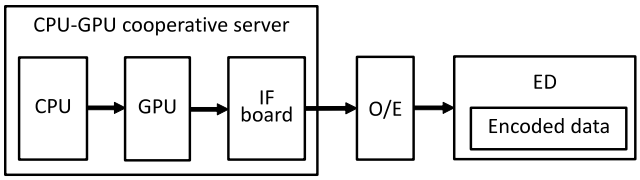


FIGURE 18. Experimental setup for demonstrating downstream 10G-EPON PHY. (O/E: optical/electrical converter).

In order to verify the softwarization of complete 10G-EPON OLT-PHY, real-time BER measurements were performed by connecting IF board output to an error detector (ED), as shown in Fig. 18. To check for bit errors, the ED (Agilent technologies serial BERT N4906B) held the 10G-EPON OLT-PHY encoded pattern. This experiment confirmed that no bit error occurred in processing the 10^{14}-bit received data. This means our proposal was successfully implemented and yielded the throughput of 10.3125 Gbps.

Although our implementation achieves the target throughput, there are two major challenges posed by commercial deployment. The first challenge is delay performance. The implemented 10G-EPON downstream has a large delay of 3.25 ms due to data buffering in our server configuration. The delay can be reduced by shortening the amount of buffer, which increases the load of CPU due to the increased number of interrupts for data input/output. Given that the interrupts trigger context switching with large loads, it is necessary to utilize a new technique for data input/output, such as the polling method. The second challenge is power consumption. The total power consumption of the server we used (see Section *IV*, Subsection A) is 153 W (as measured), while the power consumption of commercial ASIC-based products, which were developed for large-scale deployments, is 20 W per PON port [24]. For the given system throughput of 10.3125-Gbps, the energy per bit of our platform (current state) is 14.8 (nJ) whereas the ASIC-based products is 1.9 (nJ). Therefore, more power efficient versions are needed to encourage commercialization. There have been significant advances in semiconductor technology allowing CPUs and GPUs to be widely deployed to data centers and Internet of Things (IoT) devices. Promising technologies include the dynamic frequency and voltage optimization of processors tailored to programs [25] and power consumption control

technologies such as power gating [26]. These technologies have reduced CPU power consumption at the rate of about 23% per year, and thus it is expected that our system can, in 5 years, be realized with energy per bit values that will permit large-scale deployment.

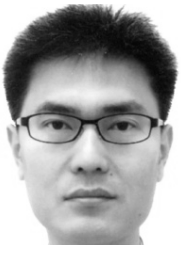
VI. CONCLUSION

Our aim is to fully realize the merits of SDN/NFV as applied to optical access systems. This paper proposed CPU-GPU cooperative implementation architecture to softwarize the complete OLT-PHY for downstream 10G-EPON functionality. Our contributions also included a scheme to control DMA data transfer and GPU kernel for the CPU-GPU cooperative computing. We performed the analysis on the processing time of each downstream function composing the complete downstream OLT-PHY, having serial and non-serial processing algorithms. All OLT-PHY functions were successfully implemented on our CPU-GPU cooperative architecture. In experiments, real-time BER measurements showed that no error occurred in processing 10G-EPON-compliant signals of 10.3125-Gbps. These results confirmed that the proposed CPU-GPU cooperative implementation can achieve the throughput of 10.3125 Gbps.

REFERENCES

[1] L. Peterson, "Cord: Central office re-architected as a datacenter," *IEEE Softw. Defined Netw. Newslett.*, Nov. 2015.
[2] T. Anschutz, L. Peterson, S. Das, and A. Al-Shabibi, "CORD: Central office re-architected as a datacenter," ONS Inspire! Webinars, Nov. 2015.
[3] J.-I. Kani *et al.*, "Flexible access system architecture (FASA) to support diverse requirements and agile service creation," *J. Lightw. Technol.*, vol. 36, no. 8, pp. 1510–1515, Apr. 15, 2018.
[4] *IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks—Specific Requirements—Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*, IEEE Standard 802.3av, 2009.
[5] *40-Gigabit-Capable Passive Optical Networks (NG-PON2)*, document ITU-T Rec. G.989, 2015.
[6] *10-Gigabit-Capable Symmetric Passive Optical Network (XGS-PON)*, document ITU-T Rec. G.9807.1, Jun. 2016.
[7] K. Nishimoto, M. Tadokoro, T. Mochida, A. Takeda, T. Tanaka, and T. Inoue, "Virtualization of EPON OLT functions and collision suppression techniques for multi-point MAC control," in *Proc. Opt. Fiber Commun. Conf. (OFC)*, Mar. 2016, pp. 1–3.
[8] M. Tadokoro *et al.*, "Design of softwarized EPON OLT and its transmission jitter suppression techniques over MPCP," in *Proc. Opt. Fiber Commun. Conf. (OFC)*, Mar. 2017, pp. 1–2.
[9] T. Anschutz, "Open GPON OLT—An open disaggregated broadband access device," OCP, San Jose, CA, USA, Tech. Rep., Mar. 2016.
[10] *Introducing ONOS—A SDN Network Operating System for Service Providers*, ON. LAB, Nov. 2014.
[11] C.-C. Chang, Y.-L. Chang, M.-Y. Huang, and B. Huang, "Accelerating regular LDPC code decoders on GPUs," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 4, no. 3, pp. 653–659, Sep. 2011.
[12] Y. Lin and W. Niu, "High throughput LDPC decoder on GPU," *IEEE Commun. Lett.*, vol. 18, no. 2, pp. 344–347, Feb. 2014.
[13] R. Li, J. Zhou, Y. Dou, S. Guo, D. Zou, and S. Wang, "A multi-standard efficient column-layered LDPC decoder for software defined radio on GPUs," in *Proc. IEEE 14th Workshop Signal Process. Adv. Wireless Commun.*, Jun. 2013, pp. 724–728.
[14] T. Suzuki, S. Kim, J. Kani, K. I. Suzuki, A. Otaka, and T. Hanawa, "Parallelization of cipher algorithm on CPU/GPU for real-time software-defined access network," in *Proc. Asia–Pacific Signal Inf. Process. Assoc. Annu. Summit Conf. (APSIPA)*, Dec. 2015, pp. 484–487.
[15] B. Shinde and S. T. Singh, "Data parallelism for distributed streaming applications," in *Proc. Int. Conf. Comput. Commun. Control Automat. (ICCUBEA)*, Aug. 2016, pp. 1–4.
[16] T. Suzuki, S.-Y. Kim, J.-I. Kani, K.-I. Suzuki, and A. Otaka, "Real-time demonstration of PHY processing on CPU for programmable optical access systems," in *Proc. Global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–6.
[17] T. Suzuki, S.-Y. Kim, J.-I. Kani, T. Hanawa, K.-I. Suzuki, and A. Otaka, "Demonstration of 10-Gbps real-time Reed–Solomon decoding using GPU direct transfer and kernel scheduling for flexible access systems," *J. Lightw. Technol.*, vol. 36, no. 10, pp. 1875–1881, May 15, 2018.
[18] T. Suzuki *et al.*, "10-Gbps real-time burst-frame synchronization using dual-stage detection for full-software optical access systems," in *Proc. Opt. Fiber Commun. Conf. (OFC)*, Mar. 2018, pp. 1–3.
[19] T. Suzuki, S. Kim, J. Kani, A. Otaka, and T. Hanawa, "10-Gb/s software implementation of burst-frame synchronization using array-access bitshift and dual-stage detection for flexible access systems," *J. Lightw. Technol.*, vol. 36, no. 23, pp. 5656–5662, Dec. 1, 2018. doi: 10.1109/JLT.2018.2870912.
[20] T. Suzuki, S.-Y. Kim, J.-I. Kani, and J. Terada, "Software implementation of 10G-EPON upstream physical-layer processing for flexible access systems," *J. Lightw. Technol.*, to be published. doi: 10.1109/JLT.2018.2883912.
[21] C. K. P. Clarke, "Reed–Solomon error correction," BBC Res. Develop., Salford, U.K., White Paper WHP 031, Jul. 2002.
[22] *IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks—Specific Requirements—Part 3: Carrier Sense Multiple Access With Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*, IEEE Standard 802.3, 2005.
[23] *PCI Express Base Specification Revision 3.0*, PCI-SIG, Beaverton, OR, USA, Nov. 2010.
[24] S. Oogushi, S. Satou, and N. Saeki, "Development of 10G-EPON to better handle increased traffic," *NEC Tech. J.*, vol. 10, no. 3, pp. 58–61, 2015.
[25] X. Mei, L. S. Yung, K. Zhao, and X. Chu, "A measurement study of GPU DVFS on energy conservation," in *Proc. Workshop Power-Aware Comput. Syst.*, 2013, pp. 1–10.
[26] S. Kaxiras, Z. Hu, and M. Martonosi, "Cache decay: Exploiting generational behavior to reduce cache leakage power," in *Proc. 28th Annu. Int. Symp. Comput. Archit.*, Jun./Jul. 2001, pp. 240–251.

TAKAHIRO SUZUKI received the B.E., M.E., and Ph.D. degrees in engineering from Waseda University, Tokyo, Japan, in 2012, 2014, and 2017, respectively. In 2014, he joined NTT Access Network Service Systems Laboratories, NTT Corporation, Kanagawa, Japan. His research interests include signal processing for optical communication systems and image/video systems. He is a member of the Institute of Electronics, Information and Communication Engineers (IEICE), and serving as an IEICE technical committee member on smart info-media systems (SIS). He received the Best Paper Awards at GLOBECOM 2016 from IEEE ComSoc TAOS TC, and at IARIA MMEDIA 2014.



SANG-YUEP KIM received the Ph.D. degree in electronics engineering from Kwangwoon University, Seoul, South Korea, in 2004. From 2004 to 2007, he was with the University of Tokyo, Japan, under a Postdoctoral Foreign Researcher Fellowship. In 2008, he joined NTT Access Network Service Systems Laboratories, NTT Corporation, Chiba, Japan, where he is currently researching DSP technologies for future optical access systems.

JUN-ICHI KANI received the B.E., M.E., and Ph.D. degrees from Waseda University, Tokyo, Japan, in 1994, 1996, and 2005, respectively, all in applied physics. In 1996, he joined NTT Optical Network Systems Laboratories, where he is involved in researching optical multiplexing and transmission technologies. Since 2003, he has been with NTT Access Network Service Systems Laboratories, where he is involved in the research and development of optical communication systems for metro and access applications, and currently heads the Access Systems Technology Group. He has been participating in ITU-T and the Full Service Access Network initiative (FSAN), since 2003.

JUN TERADA received the B.E. degree in science and engineering and the M.E. degree in computer science from Keio University, Kanagawa, Japan, in 1993 and 1995, respectively. In 1995, he joined NTT LSI Laboratories, where he was involved in the research and development of low-voltage analog circuits, especially, A/D and D/A converters. From 1999, he was involved in developing small and low-power wireless systems for sensor networks. From 2006, he was involved in high-speed front-end circuits for optical transceivers. He is currently a Senior Research Engineer and a Supervisor with NTT Access Network Service Systems Laboratories, where he is responsible for Research and Development management of optical access networks, including fixed-wireless convergence and virtualization technology. He is a Senior Member of the Institute of Electronics, Information and Communication Engineers (IEICE) of Japan, and serving as a Technical Committee Member on Asian Solid- State Circuits Conference (A-SSCC). He is serving as the Vice Chair of IEICE Technical Committee on Communication Systems (CS).