Parallelization of Cipher Algorithm on CPU/GPU for Real-time Software-Defined Access Network

Takahiro Suzuki*, Sang-Yuep Kim*, Jun-ichi Kani*, Ken-Ichi Suzuki*, Akihiro Otaka* and Toshihiro Hanawa†
*NTT Access Network Service Systems Laboratories, Kanagawa, Japan
†Information Technology Center, The University of Tokyo, Chiba, Japan
E-mail: suzuki.takahiro@lab.ntt.co.jp Tel/Fax: +81-46-859-2240/5513

***Abstract*—Network Function Virtualization (NFV) and Software Defined Network (SDN) are attracting attention with the goal being enhanced networks efficiency. To enhance flexibility and to meet user requests and conditions, software implementation of communications equipment is pursued as our approach. It becomes possible to implement various optical line terminal (OLT) functions on the common hardware by realizing the functions as software. The proposed approach prevents kinds of systems from increasing and simplifies maintenance. Throughput on lower-performance software comparing with dedicated circuits is a problem. This paper focuses on a cipher algorithm for the access network as typical of the more demanding OLT functions and proposes parallel implementation on CPU/GPU resources that offers real-time processing. This paper targets the CTR and GCM which are utilized on PON systems. For software implementation, this paper proposes a data-parallelization-based algorithm architecture. Evaluation results gathered from a many-core CPU simulator and GPU show that throughput is sufficient for the 5.37-Gbps CTR-AES128 process and the 914 Mbps GCM-AES128 process. The proposed parallel algorithm on a GPU is 141 times faster than serial processing. It is also found that the cipher circuit of 1-Gbps-class PON systems utilizing CTR-AES128 can be replaced with GPU.**

I. INTRODUCTION

Recently, Network Function Virtualization (NFV) and Software Defined Network (SDN) [1] are considered essential concepts for the next generation of networlks. NFV/SDN application to the access network is underway to deal with various user requests and conditions. The Passive Optical Network (PON) is an access system in which most single Optical Line Terminals (OLTs) handle multiple Optical Network Units (ONUs). Each ONU and OLT have quite specific attributes in terms of throughput, delay and distances between ONU and OLT.

Different systems, such as GE-PON [2], 10G-EPON [3] and NG-PON2 [4], have been developed to meet different needs. We expect that more kinds of equipment will be needed to satisfy the expanding demands made by users, for example, more capacity. Accordingly, network flexibility must be increased such that common hardware can implement a wired variety of functions on demand. Our solution is the programmable OLT based on general-purpose CPU or Graphics Processing Unit (GPU). OLT signal processing functions include encipherment, Forward Error Correction (FEC), and MAC framing. This paper focused on cipher algorithm. Various cipher algorithms are proposed in PON systems [5]. To deal with user's requests, performing various cipher algorithms on common hardwares

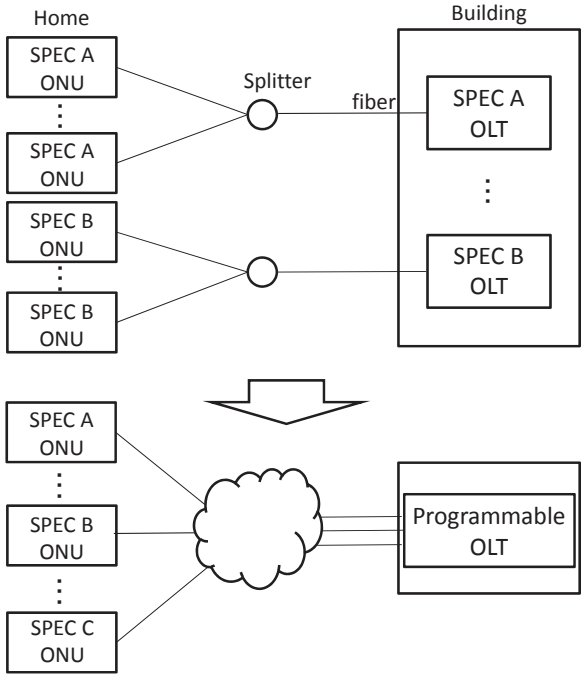

Fig. 1. Programmable OLT

is required. This paper introduces programmable cipher processing that can switch between protocols and throughput, 1 Gbps or 10 Gbps and ON/OFF, to meet user requests while optimizing processor resources, see Fig. 1. It prevents kinds of systems from increasing and simplifies maintenance. It also becomes possible to deal with user's requests quickly.

Current cipher algorithms are run on dedicated devices/circuits such as Application Specific Integrated Circuit (ASIC) since software running on CPU/GPU has thought to have insufficient performance. Several papers have considered the implementation of cipher algorithms on Field-Programmable Gate Arrays (FPGAs) [6], [7] and CPUs [8]. However, no practical solution involves the use of contemporary CPU/GPU systems.

This paper proposes the implementation of a parallelized cipher algorithm for real-time software processing on commercial GPUs in access networks. We take as our examples, CTR and GCM, which are utilized by PON systems. In addition, Camellia is evaluated to investigate the feasibility of high

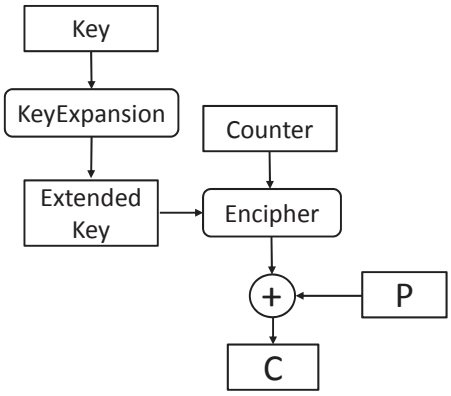

Fig. 2. CTR cipher

throughput processing. On software implementation, this paper proposes data-parallelization-based algorithm architecture. Pre-computable parts are split from the main signal processing parts. Evaluations using a many-core CPU simulator and GPU show acceptable throughput and the feasibility of the software-defined access network.

II. Cipher algorithm in access network

The access network mainly uses two ciphers as follows.

- CTR-AES128: G-PON and XG-PON
- GCM-AES128: SIEPON for GE-PON and 10G-EPON

This section describes each algorithm.

A. CTR cipher

Counter (CTR) mode is utilized on PON systems standardized by ITU. Advanced Encryption Standard (AES) [9] is the cipher algorithm. The process flow is shown in Fig. 2. The key of 128 bits is obtained by key exchange. The counter is the value that is incremented. P is the 128 bit plain text input. C is the enciphered text output. Counter is enciphered and a key stream block is calculated. Plain text and key stream block are calculated by XOR and enciphered text is generated. The cipher algorithm includes the process that creates the extended keys utilized in the main cipher algorithm.

B. GCM cipher

In conformity with IEEE 802.1AE, Galois/Counter Mode (GCM) [10] is utilized on PON systems standardized by IEEE. GCM is cipher mode with authentication. It calculates ciphered text and authentication tag. The process flow is shown in Fig. 3. J_0 is initial vector of 128bit. A is additional authenticated data. $len(X)$ is the bit length of bit string X. Plain text is ciphered similarly in CTR mode. Additional data are added and subjected to the GHASH function. GHASH function is defined as,

$$Y_i = (Y_{i-1} \oplus X_i) \cdot H, \tag{1}$$

where $Y_0 = 0^{128}$, H is the hash subkey and X_i is a sequence of blocks. In this case, $X = A||C||[len(A)]_{64}||[len(C)]_{64}$. MSB_s is a function that outputs a bit string consisting of the s left-most bits of the bit string. After that, the authentication tag is calculated.

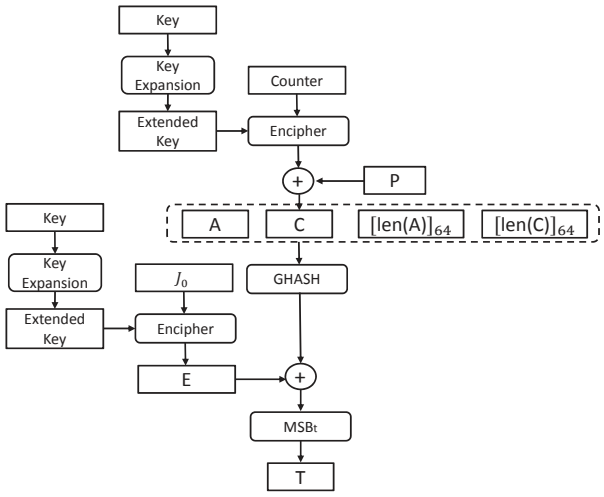

Fig. 3. GCM cipher

III. Parallelization of Cipher Algorithm on CPU/GPU

This paper proposes a parallel implementation of cipher algorithms to enhance throughput on CPU/GPUs. As desired values of throughput, for example, GE-PON is 1 Gbps and 10G-EPON is 10 Gbps. As the cipher, PON systems use CTR and GCM. In hardware approaches, the cipher processes are implemented by combining task-level parallelization and pipelining to minimize resource requirements. Fortunately, the cipher data blocks have clearly independent characteristics so significant speed enhancement is expected with data parallelization. This paper proposes a data-parallelization-based algorithm architecture for software implementation, a key component of which strips out pre-computable data parts that are not related to main signal processing.

A. Parallelization of CTR-AES128

The process flow is shown in Fig. 4. Key expansion parts are pre-computable if the key is not changed. Thus, these parts are removed from main signal parts and pre-computed. The value (Extended key) is sent to memory and utilized by main signal processing. This increases the utilization of memory but reduces the real-time processing load.

Next, our parallelization approach is detailed. The data stream is divided into 128 bit blocks and each is XORed. In the GPU, blocks are allocated to the thread blocks for parallelized processing. Counter is calculated in each block through incrementation and sent for encipherment. Extended key is also entered for sequential encipherment in each thread. The processes of encipherment and XORing between P and enciphered counter can be parallelized because all blocks are independent.

B. Parallelization of GCM-AES128

The process flow is shown in Fig. 5. Key expansion parts are pre-computable when the key is not changed. In addition, E is pre-computable because it depends only on initial values J_0

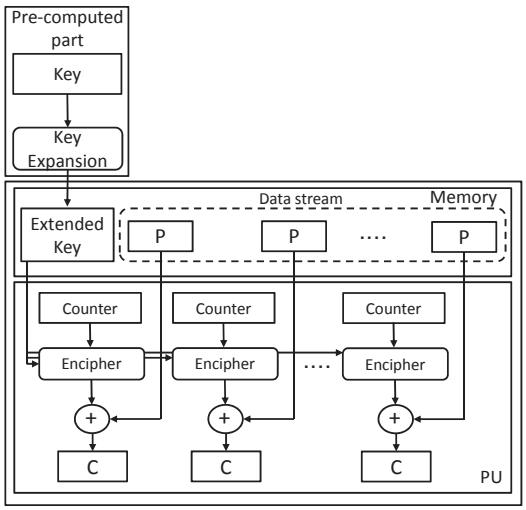

Fig. 4. Parallelization of CTR cipher

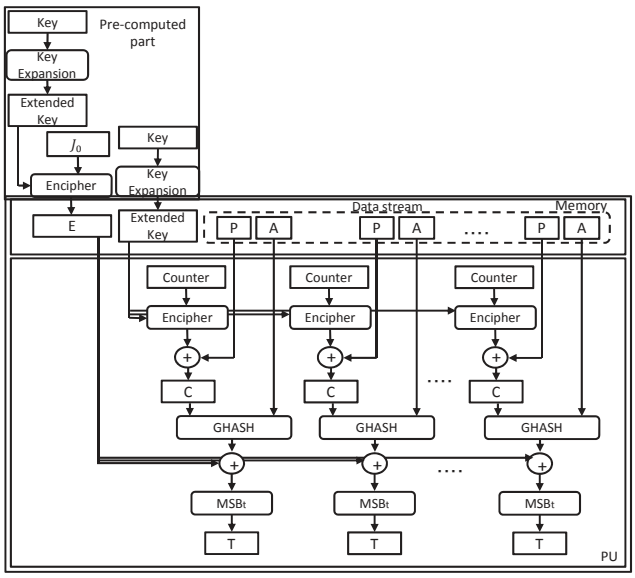

Fig. 5. Parallelization of GCM cipher

and a key. Thus, these parts are removed from the main signal parts and pre-computed. The values (E and Extended Key) are sent to memory and utilized by main signal processing.

Next, our parallelization approach is detailed. Data stream (P and A) is divided into blocks and processed as follows. Data E and extended key in memory are sent for encipherment. Encipher parts, XORing P and enciphered counter, GHASH, and MSB are independent in each block. Thus, data parallelization can be utilized. The architecture enciphers each block simultaneously on the multiple CPU/GPU cores.

IV. Evaluation results

This section shows evaluation results which proposed parallel implementation of cipher algorithm of PON on CPU/GPU.

TABLE I
Specifications of CPU and GPU

	CPU	GPU
Simulator/Processor	gem5 simulator	GTX780Ti
Clock frequency[MHz]	3200	875-928
Memory[GB]	8	3.072
Memory bandwidth[GB/s]	8	336
Environment	OpenMP 4.0	CUDA 6.5

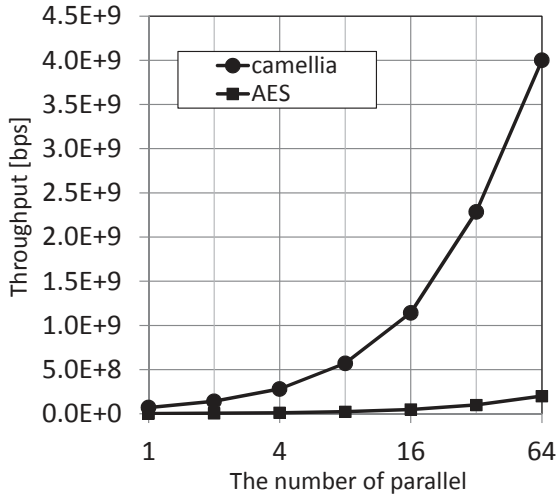

Fig. 6. Throughput of CTR-AES/Camellia128 on many-core CPU simulator

The throughput is defined as,

$$Throughput = \frac{the\ number\ of\ processed\ bits}{processing\ time}. \quad (2)$$

This paper examines the cipher algorithms of AES and Camellia [11]. Camellia is a low complexity algorithm that offers almost the same security as AES but is more efficient. The evaluation results, throughput of each implementation, are shown below.

A. CPU implementation

First, throughput results of CTR-AES/Camellia with CPU implementation are shown, see Fig. 6. We utilized the gem5 simulator [12] which allows the number of cores, caches, and memory to be varied with OpenMP. The resource is the number of cores generated by the simulator. Parameters were set to clock frequency 3.2GHz, memory 8GB, L1 cache 32KB, and L2 cache 256KB.

The results shows the throughput of CTR-AES is lower than 1 Gbps. When utilizing 64 cores, AES throughput is 315 Mbps while that of Camellia is 6.29 Gbps. Throughput of a single core is 3.03 Mbps. Thus, utilizing a 64-core CPU yields a roughly 66 times higher speed than a single core.

B. GPU implementation

Results of GPU Implementations are shown here. The utilized GPU is NVIDIA GeForce GTX780Ti with CUDA.

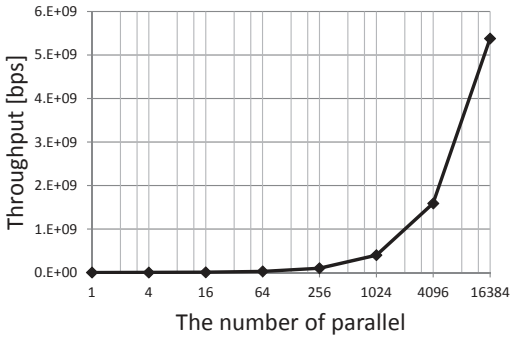

Fig. 7. Throughput of CTR-AES128 on GPU

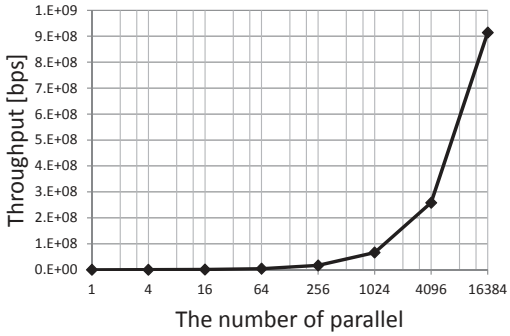

Fig. 8. Throughput of GCM-AES128 on GPU

For the GPU evaluations, transfer time is removed from the processing time.

First, Fig. 7 shows AES-CTR throughput versus the number of functional units in the GPU. It is found that 5.37 Gbps is obtained when 16384 functional units are utilized. This result shows that it satisfies the desired throughput of 1 Gbps. In other words, it shows the feasibility of implementing 1-Gbps-class PON cipher in software. Adequate performance is obtained even when using only 2048 functional units, so it is possible the remaining cores to perform other process. CPU throughput with serial processing is 38 Mbps (Intel core i7-4770). Thus, the 16384 functional units of the GPU yield 141 times higher speeds than serial processing.

Next, Fig. 8 shows AES-GCM throughput versus GPU resource number. The GCM Tag is 64 bits long. With 16384 functional units, the throughput is 914 Mbps. It is lower speed comparing with AES-CTR. More proposals are required to achieve the 10 Gbps demanded by AES-GCM.

Finally, Fig. 9 shows GCM-Camellia throughput versus GPU resource number. The throughput of 5.42 Gbps is achieved with 16384 functional units. It is found that the throughput is enhanced by replacing GCM-AES with GCM-Camellia.

V. Conclusion

To allow the access network to offer the flexibility needed to satisfy user requests and demands, software implementation of communications equipment is a good solution. This paper

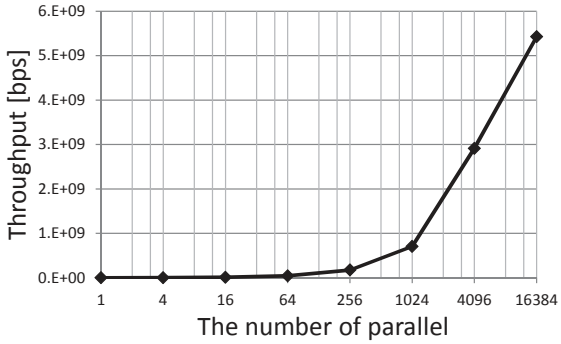

Fig. 9. Throughput of GCM-Camellia128 on GPU

focused on the cipher algorithms employed in the Optical Line Terminals (OLTs) and proposed a parallel implementation architecture that utilizes GPU resources for real-time software processing. We examined and implemented in software paralleized versions of the CTR and GCM ciphers utilized in PON systems. Simulator results shows that 64-core CPU of roughly 66 times greater speed than a single core CPU. The propose parallel algorithm on a commercial GPU is 141 times faster than serial processing. The GPU achieved high AES-CTR throughput, 5.37 Gbps, which is much faster than CPU implementation. In addition, this paper shows the cipher circuit of 1-Gbps-class PON systems utilizing CTR-AES128 can be implemented on a GPU. Future plan include raising the speed of AES-GCM algorithm processing.

References

[1] European Telecommunications Standards Institute (ETSI) GS NFV 001, Use Cases, 2013.
[2] "IEEE Standard for Information technology Telecommunications and information exchange between systems Local and metropolitan area networks Specific requirements", IEEE, 2004.
[3] F. J. Effenberger, J. Kani, and Y. Maeda, "Standardization Trends and Prospective Views on the Next Generation of Broadband Optical Access Systems", IEEE Journal on Selected Areas In Communications, Vol. 28, No. 6, August 2010.
[4] "40-Gigabit-capable passive optical networks (NG-PON2): General requirements," ITU-T, 2013.
[5] M. Hajduczenia, "On EPON Security Issues", IEEE Commun. Surveys and Tutorials, vol. 9, no. 1, pp.68 -83, 2007.
[6] Elmoghany, Mohamed, et al, "FPGA implementation of high speed XTS-AES for data storage devices", Internet Technology and Secured Transactions (ICITST), 2011 International Conference for. IEEE, 2011.
[7] A. Hodjat and I. Verbauwhede, "A 21.54 Gbits/s fully pipelined processor on FPGA", Proc. IEEE 12th Annu. Symp. Field-Programm. Custom Comput. Mach., pp.308 -309 2004.
[8] B. Liu and B. Baas. "Parallel AES encryption engines for many-core processor arrays. Computers", IEEE Transactions on, 62(3):536-547, 2013.
[9] National Institute of Standards and Technology (NIST), "Advanced Encryption Standard (AES)", FIPS PUB-197, 2001.
[10] National Institute of Standards and Technology (NIST), "Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", November 2007.
[11] K. Aoki, T. Ichikawa, etl.. "Camellia: A 128-Bit Block Cipher Suitable for Multiple Platforms - Design and Analysis. SAC 2000", LNCS 1281, pp. 39-56, Springer-Verlag, 2000.
[12] Nathan Binkert, et.al, "The gem5 simulator", ACM SIGARCH Computer Architecture News archive, Vol.39, No.2,pp.1-7, May 2011.