IEICE TRANS. INF. & SYST., VOL.E97–D, NO.6 JUNE 2014

PAPER

Throughput and Power Efficiency Evaluation of Block Ciphers on Kepler and GCN GPUs Using Micro-Benchmark Analysis

Naoki NISHIKAWA[†a)], **Keisuke IWAI**[†], **Hidema TANAKA**[†], *and* **Takakazu KUROKAWA**[†], *Members*

SUMMARY Computer systems with GPUs are expected to become a strong methodology for high-speed encryption processing. Moreover, power consumption has remained a primary deterrent for such processing on devices of all sizes. However, GPU vendors are currently announcing their future roadmaps of GPU architecture development: Nvidia Corp. promotes the Kepler architecture and AMD Corp. emphasizes the GCN architecture. Therefore, we evaluated throughput and power efficiency of three 128-bit block ciphers on GPUs with recent Nvidia Kepler and AMD GCN architectures. From our experiments, whereas the throughput and per-watt throughput of AES-128 on Radeon HD 7970 (2048 cores) with GCN architecture are 205.0 Gbps and 1.3 Gbps/Watt respectively, those on Geforce GTX 680 (1536 cores) with Kepler architecture are, respectively, 63.9 Gbps and 0.43 Gbps/W; an approximately 3.2 times throughput difference occurs between AES-128 on the two GPUs. Next, we investigate the reasons for the throughput difference using our micro-benchmark suites. According to the results, we speculate that to ameliorate Kepler GPUs as co-processor of block ciphers, the arithmetic and logical instructions must be improved in terms of software and hardware.

key words: *throughput, power efficiency, GPU, OpenCL, AES, Camellia, SC2000, Kepler, Graphics Core Next*

1. Introduction

The adoption of computer devices for widespread use has grown explosively. Moreover, the capacity of their data storage has become much greater. At the same time, security countermeasures against these devices have been increasingly sought. Encryption processing is an effective means of protecting these data. Moreover, computer systems are used everywhere, from data centers to handheld devices. Such expansion of the range of use of computers demands additional features related to power consumption, such as battery capacity, heat, and heat-associated factors.

For these reasons, encryption processing on Graphics Processing Units (GPUs) has been examined for encryption of a large amount of data because it benefits from software flexibility and hardware-based computing performance. As a result, it is expected to be applied on secure database backup and full disk encryption. Moreover, some studies have shown that GPUs are a power-efficient device in some fields [1], [2]. However, the evaluation of power efficiency against encryption processing on a GPU is insufficient. For example, only our previous work [3] suggests that the power efficiency of encryption processing on a GPU is excellent compared to that on multicore CPUs.

Recently, Nvidia Corp. and AMD Corp. have respectively released Kepler and Graphics Core Next (GCN) as recent GPU architectures. They are currently revealing their roadmaps, which accentuate Kepler and GCN as the mainstream designs [4], [5]. Therefore, the performance evaluation of block ciphers on GPUs with Kepler and GCN architectures is an important research issue for understanding the potential of high-speed encryption processing on a GPU. However, regarding encryption technology on a GPU, performance only with Nvidia Fermi and AMD VLIW architectures have been reported in previous works [6]–[8].

Therefore, we evaluate the throughput and power efficiency of three 128-bit block ciphers (AES, Camellia, and SC2000) on GPUs with Nvidia Kepler and AMD GCN architectures. First, we evaluate the throughput of three block ciphers on these GPUs ignoring data transfer between the CPU and GPU. Next, we measure the power consumption of the block cipher encryption on a GPU. Then we obtain the power efficiency dividing the throughput by the power consumption value. Moreover, we investigate the difference of values between the throughputs on Kepler and GCN GPUs using micro-benchmark analysis.

2. GPU Architectures

In this work, we used OpenCL language [9], which supports several GPU vendors, because we evaluate and compare performances of both Nvidia and AMD GPUs. The thread block and thread are designated respectively as a work-group and work-item in OpenCL.

GPU architectures comprise hierarchical structures of processor cores, the chip of which has N × Compute Units (CUs). Each CU includes several Processor Elements (PEs), registers, an instruction unit, and a local memory. The programming model in OpenCL is large-scale thread-level parallel processing corresponding to the hierarchical processor cores. Work-items are managed as work-groups of several work-items. A GPU scheduler distributes work-groups evenly to CUs. Then each work-item of a work-group is executed on a PE. However, instructions are issued to a batch grouping a specific number of work-items (32 work-items for Nvidia and 64 work-items for AMD GPUs). Therefore, the number of work-items per work-group is recommended to be set as a multiple of the batch size listed above. Furthermore, the PEs are each deeply pipelined to raise efficiency to execute instructions of work-items in lock-step.

Manuscript received November 1, 2013.
Manuscript revised January 16, 2014.

[†]The authors are with the Department of Computer Science, National Defense Academy of Japan, Yokosuka-shi, 239–8686 Japan.
a) E-mail: fa50327151@yahoo.co.jp

DOI: 10.1587/transinf.E97.D.1506

Recently, Nvidia Corp. and AMD Corp. have been proceeding respectively with their development of Kepler and GCN GPU architectures. Kepler is designed based on Fermi architecture of one generation earlier for better power efficiency. Compared to the Fermi architecture, the number of PE pipeline stages is reduced. Thereby, the size of the clock synchronization circuit decreases and the amount of power consumption necessary for a CU is decreased. Instead of dropping of the core clock frequency, more PEs are packed into a single CU in Kepler architecture. In contrast, AMD Corp. has promoted VLIW-based GPU architecture for graphics applications, but has now changed its development guidelines to a scalar-based GCN architecture. In fact, GCN is reconstructed from VLIW architecture to gain higher performance at general purpose computing.

Currently, CPU and GPU memory spaces are separated exhaustively in current GPU architectures. Then trivial data transfer between CPU and GPU memory spaces is required. However, in this work, we regard encryption throughput ignoring data transfer between CPU and GPU as a throughput value using a GPU. This is true because Nvidia Corp. and AMD Corp. have announced a new technology development program by which memory spaces between the CPU and GPU are integrated. Then the data transfer between both regions is eliminated completely. For example, Nvidia Corp. suggests Unified Virtual Memory technology from the next generation of GPU architecture Maxwell, which is developing based on Kepler [4]. In contrast, AMD Corp. has been developing heterogeneous Uniform Memory Access (hUMA) technology, which is a new technology to integrate both memory spaces [5]. Actually, AMD Corp. has announced that upcoming AMD Accelerated Processing Units (APUs) with the new Kaveri architecture include hUMA technology, and that the interior GPU of the APU is designed based on GCN architecture.

3. Symmetric Block Ciphers

In this work, we targeted three 128-bit block cipher primitives (AES [10], Camellia [11], and SC2000 [12]), which cover all types of structures of block ciphers. Although 128-bit, 192-bit, and 256-bit key sizes can be selected in the block ciphers, we discussed only 128-bit versions in this paper.

In AES, the structure is an SPN network. The 128-bit key algorithm defines 10-round processes. Each round consists of four transformations: SubBytes, ShiftRows, MixColumns, and AddRoundKey. The final round differs slightly from the other rounds: it does not include MixColumns. In AES, a round process can be combined into a transformation simply using lookup tables called "T-Boxes" and XOR operations [13]. Letting a be the round input, which is divided into four inputs a_0, a_1, a_2, and a_3, each of which consists of 32-bits, the round output e is represented as

$$e_j = T_0[a_{0,j}] \oplus T_1[a_{1,j+1}] \oplus T_2[a_{2,j+2}] \oplus T_3[a_{3,j+3}] \oplus k_j,$$

where T_0, T_1, T_2, and T_3 are lookup tables and k_j is the j-th column of a round key.

In Camellia, the structure is a Feistel network. The 128-bit key algorithm defines 18-round processes. Each round includes an F-function, which includes 8 table substitutions, 2 XOR instructions with two 32-bit keys, and several logical operations. Furthermore, FL and FL^{-1}-functions consisting of logical operations are inserted.

In SC2000, the structure is a hybrid of SPN and Feistel. The 128-bit key algorithm defines seven-round processes. In each round, a plaintext is encrypted through five functions in sequence (I, B, I, R, and R). The I-function is XORs with four 32-bit keys and B-function consists of logical operations. The R-function includes several table substitutions and logical operations. The inputs of these tables are separation of 32-bit and can be selected from multiple options such as (6-bit, 10-bit, 10-bit, and 6-bit) or (11-bit, 10-bit, and 11-bit) depending on the computer memory capacity. In the former, two tables with 6-bit input and two tables with 10-bit are used for substitution. In the latter, two tables are done with 11-bit input and one table with 10-bit input.

Electronic Code Book (ECB), CountTeR (CTR), or Xor-encrypt-xor Tweakable code book mode with ciphertext Stealing (XTS) [14] are known as parallelizable modes in block ciphers. ECB uses a single key applied to all plaintexts, and CTR uses a key stream generated from a secret key and combined to plaintexts. In the CTR mode, the key stream generation is conducted in the same manner as ECB. In the XTS mode, plaintexts are encrypted using two ECB modes. Therefore, we deal only with ECB mode in this study.

4. Related Work

4.1 Throughput of Block Ciphers on GPUs

To date, no report in the relevant literature has described research of the evaluation of block ciphers on GPUs with Nvidia Kepler and AMD GCN architectures. Instead, some works have been done for the evaluation of throughputs on GPUs with Nvidia Fermi and AMD VLIW architectures [6]–[8], [15].

We previously implemented AES-128 (with T-Box), Camellia-128, and SC2000-128 with ECB mode in CUDA environment and respectively achieved about 48.6 Gbps, 50.7 Gbps, and 73.4 Gbps on a Nvidia Tesla C2050 with Fermi architecture, ignoring data transfer between CPU and GPU [15]. Li et al. also implemented AES-128 with T-Box on Nvidia Tesla C2050 using CUDA, giving throughput of 60.0 Gbps ignoring data transfer [6]. However, they reordered the plaintext in host memory before loading it onto device memory to ingenerate coalesced access into global memory.

Gervasi et al. implemented AES-192 and achieved about 38.6 Mbps at 2 KB file size ignoring data transfer using OpenCL on an AMD Firestream 9270 with VLIW architecture [7]. Moreover, the throughput of their AES-192

with data transfer was 31.9 Mbps. Wang et al. implemented AES-256 in XTS mode at 828.8 MB/s using OpenCL on an AMD Radeon HD 5970 with VLIW architecture [8].

4.2 Power Efficiency of Block Ciphers on GPUs

We previously evaluated the throughput and power efficiency of three 128-bit block ciphers (AES, Camellia, and SC2000) on AMD GPUs with VLIW architecture [3]. Results show that the power efficiency on Radeon HD 5450 with VLIW architecture and 80 cores is about nine times higher than that on AMD Phenom II X6 1100T CPU with 6 cores. Aside from our study, no other effort at evaluation of power efficiency of block ciphers on GPUs has been reported.

5. Implementation

For implementation of block ciphers on a GPU, we take advantage of achievements of previous works [3], [16], [17]. First, round keys are generated in the host. Then they are transferred to the GPU's global memory along with tables and plaintexts. Generally in optimized software implementation of block ciphers, the same tables are referred multiple times. Therefore, storage of the tables and round key in local memory with low access latency engender higher performance. For this reason, before encryption starts, work-items deploy them from global memory to local memory in CUs. At the beginning of encryption time, work-items load a 128-bit plaintext to individual registers. During encryption with each block cipher primitive, work-items process 128-bit plaintext blocks independently in parallel and therefore keep hold on the plaintext data in registers. After encryption, each work-item stores a 128-bit ciphertext to global memory.

6. Evaluation

6.1 Evaluation Environment

Evaluation environments are presented in Table 1. We used two machines for the evaluation: one is for Nvidia GPUs; the other is for AMD GPUs. For the evaluation of encryption processing on high-end GPUs aimed at desktop and workstation markets, we used Geforce GTX 680 (Kepler) and Radeon HD 7970 (GCN), of which the core quantities are the most in respective GPU architectures. In contrast, as for the evaluation of low-end GPUs aimed at handheld devices, we used Geforce GTX 650 (Kepler) and Radeon HD 7750 (GCN), of which the quantities of cores are the least in their respective architectures. Moreover, for comparison, we used two GPUs with one-generation-earlier architectures: Geforce GTX 580 with Fermi architecture and Radeon HD 6770 with VLIW architecture. The specifications of these Nvidia and AMD GPUs are presented respectively in Tables 2 and 3. Unlike the other architectures, each PE of Radeon HD 6770 comprises a VLIW type of five cores. Furthermore, for pertinent evaluation of throughput and power consumption, the same memory module and power supply are used as well as the same operating system and host compiler.

In our manual optimization to extract higher throughput, the combinations of the number of work-groups and work-items per work-group were, respectively, (64, 1024), (16, 1024), (128, 512), (256, 256), (64, 256), and (80, 256) for Geforce GTX 680, Geforce GTX 650, Geforce GTX 580, Radeon HD 7970, Radeon HD 7750, and Radeon HD 6770. Those values were common to all three block ciphers.

6.2 Evaluation of Throughput

We run each encryption kernel through the entire code one hundred times and then obtain the average, disregarding the first 10 iterations to avoid cold instruction cache miss. The encryption throughput values ignoring data transfer (T_{enc}) are presented in Fig. 1. Results show that whereas T_{enc} of AES-128 on Radeon HD 7970 was 205.0 Gbps, that on Radeon HD 6770 was 38.5 Gbps; the former value was about 5.3 times higher than the latter one. The reason is that AMD GPU architecture is renovated from VLIW, which was directed at graphic processing, to GCN, which is geared toward general purpose computing. Moreover, the number of processor cores and the core clock frequencies are respectively higher. In contrast, as for Nvidia GPUs, considering that GTX 680 has 0.69 times lower clock frequency but three times the number of processing cores compared to GTX 580, T_{enc} on GTX 680 is naturally expected be higher than that on GTX 580. In reality, however, whereas T_{enc} of AES-128 on GTX 580 ignoring data transfer was 75.8 Gbps, that on GTX 680 was 63.9 Gbps. Moreover, approximately

Table 1 Machine specifications.

	Machine 1	Machine 2
CPU	Intel Core i7-2600K	AMD Phenom II X6 1100T
Motherboard	ASUS P8Z68-V	ASUS M5A99X EVO
Memory	Corsair, CMX8GX3M2A1333C9 DDR3 PC3-10600 32 GB	
OS	CentOS 6.0 (Kernel ver. 2.6.32-71)	
Complier	gcc ver. 4.4.4 (Option -O3)	
Power supply	SilverStone Strider Gold SST-ST1200-G (1200 watts output)	
GPU accelerator	Geforce GTX 680 Geforce GTX 650 Geforce GTX 580	Radeon HD 7970 Radeon HD 7750 Radeon HD 6770
Graphics driver	Nvidia CUDA toolkit ver. 4.2	AMD Catalyst ver. 13.4

Table 2 Specifications of Nvidia GPUs.

	Geforce GTX 680	Geforce GTX 650	Geforce GTX 580
Architecture	Kepler		Fermi
Process technology	28 nm		40 nm
Core clock frequency (At boost)	1006 MHz (1058 MHz)	1058 MHz (–)	1544 MHz (–)
# CUs	8	2	16
# PEs/CU	192		32
# Cores/GPU	$8 \times 192 = 1536$	$2 \times 192 = 384$	$16 \times 32 = 512$
GPU memory capacity	2 GB	1 GB	1.5 GHz
12 V external power	8-pin + 6-pin	6-pin	8-pin + 6-pin

Table 3 Specifications of AMD GPUs.

	Radeon HD 7970	Radeon HD 7750	Radeon HD 6770
Architecture	GCN		VLIW
Process technology	28 nm		40 nm
Core clock frequency	925 MHz	800 MHz	850 MHz
# CUs	32	8	10
# PEs/CU	64		16
# Cores/GPU	$32 \times 64 = 2048$	$8 \times 64 = 512$	$10 \times 16 \times 5 = 800$
GPU memory capacity	3 GB	1 GB	1 GB
12 V external power	8-pin + 6-pin	Not required	6-pin

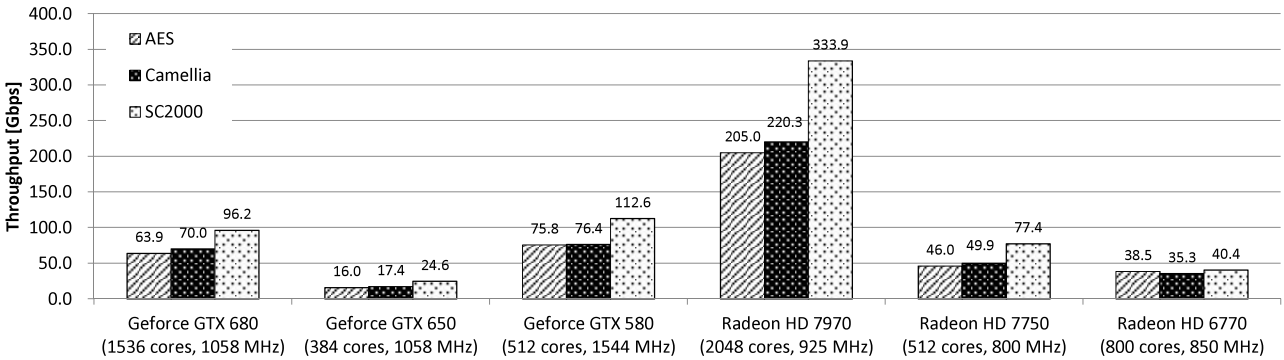


Fig. 1 Comparison of encryption throughputs of block ciphers on a GPU (T_{enc}).

3.2 times the difference of T_{enc} occurs between AES-128 on GTX 680 and on Radeon HD 7970. We discuss and analyze the reason for this measuring result for T_{enc} on Kepler and GCN GPUs through micro-benchmarking methodology described in Sect. 7.

Moreover, as the reference of measurement result on a multicore CPU, we cite the encryption throughput on AMD Phenom II X6 1100T with six cores from our previous work [3], whose machine specification is the same as the one used in this paper. These programs are parallelized and implemented using OpenMP programming interface [18]. The throughput of AES-128, Camellia-128, and SC2000-128 are, respectively, 8.3 Gbps, 6.8 Gbps, and 7.8 Gbps using six threads. These values are, respectively, only 4.0%, 3.1%, and 2.3% of those on AMD Radeon HD 7970 described in this paper.

Incidentally, the `AESEncryptDecrypt` benchmark is provided as an AES cryptography sample program in AMD Accelerated Parallel Processing SDK [19]. However, this program is unfortunately implemented using a non-optimized AES algorithm, the round of which consists of four transformations and which differs considerably from the optimized one, as demonstrated in Sect. 3. Therefore, we do not use this benchmark to verify the results presented above for Kepler and GCN GPUs.

6.3 Evaluation of Power Consumption

The power efficiency of encryption processing has become an important evaluation item. As described herein, we use bps-per-watt as a metric to quantify the power efficiency of encryption processing on a GPU because the performance–power ratio is generally used to quantify the energy efficiency of particular computer architecture or computer hardware. For example, the Green500 project [1] ranks energy-efficient supercomputers by the LINPACK performance per watt. Moreover, the power efficiency of other applications such as radio astronomy is evaluated using a flops-per-watt metric [2].

6.3.1 Power Consumption Measurement

Generally, power is supplied to a GPU through a PCIe bus and 12 V external power lines when needed. Some previous studies assessed the voltage or current from the bus and lines. Then the power consumption was calculated [20]–[22]. A riser card was inserted between a GPU and a PCIe connector to measure the voltage or current derived only from the bus to the GPU. This approach is certainly appropriate if only the GPU power consumption is measured. However in reality, not only the GPU but every peripheral device receives power from the motherboard during encryption processing. Specifically, computers receive power through a 24-pin ATX power supply and 8-pin or 6-pin 12 V external power lines.

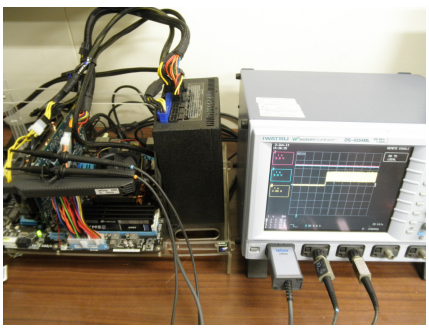
Therefore, we use a measurement approach by which the current and voltage of each power line are measured using current and voltage probes and an oscilloscope shown in Table 4, as depicted in Fig. 2. However, in general, computers consume power not only during processing time but during idle time. Therefore, we examined the power consumption during idle time (P_{idle}) and that during encryption time (P_{enc}). Specifically, we first measured powers through 3.3 V and 5 V, and through 12 V lines of two kinds from a ATX power supply to a motherboard (and one through 12 V external lines to GPU's connectors when required), respectively at idle time (e.g., $P_{idle\_lineof5V}$) and at encryption time (e.g., $P_{enc\_lineof5V}$), as depicted in Fig. 3. For highly accurate measurement, we set the steady state by keeping idle or encryption processing for more than 10 s consecutively. Then we measured the power during 10 s with 25,000/s sampling frequency. Next, we calculated averages of the powers (e.g., $P_{idle\_lineof5V\_avg}$ and $P_{enc\_lineof5V\_avg}$) and summed them up respectively at idle or encryption times as

$$P_{idle} = P_{idle\_lineof5V\_avg} + P_{idle\_lineof3.3V\_avg} + \cdots \quad (1)$$

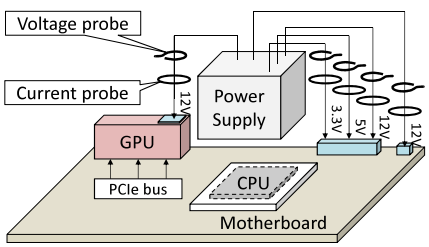
$$P_{enc} = P_{enc\_lineof5V\_avg} + P_{enc\_lineof3.3V\_avg} + \cdots . \quad (2)$$

Table 4 Devices to measure power consumption.

Digital oscilloscope	Iwatsu DS-4354ML 500 MHz 1 GS/S
Current probe	LeCroy CP015 15 ampere 50 MHz
Voltage probe	LeCroy PP006A 12 pF 500 MHz



(a) Overview



(b) Power measurement points

Fig. 2 Power consumption measurement approach.

6.3.2 Measurement Results of Power Consumption

Measurement results of power consumption at idle (P_{idle}) and encryption times (P_{enc}) in each environment are presented in Fig. 4. This figure is represented as a stacked bar chart showing respective power consumption through each power line. Moreover, each power line has a different standard deviation value. Therefore, the range of standard deviation in each environment is presented in Table 5. As described in this paper, we regard the different value (P_{diff}) between power consumption of idle and encryption times (i.e., $P_{diff} = P_{enc} - P_{idle}$) as the energy consumed by encryption processing on a GPU. P_{diff} in each environment is presented in Table 6.

As described in Sect. 2, Kepler is an architecture designed for better power efficiency. Our experimentally obtained results in Fig. 4 show that, for example, the difference between the respective power consumptions of AES encoding and idle times on GTX 680 was 75.6% of that on GTX 580, although GTX 680 has three times the processing cores of GTX 580. In contrast, as for an AMD GPU with GCN architecture, the encryption processing on Radeon HD 7970 consumed about 1.9 times higher power than that on Radeon HD 6770, as shown in Fig. 4. In addition, irrespective of block cipher algorithms, the three key sizes showed no great difference in terms of power consumption.

6.4 Evaluation of Power Efficiency

Power efficiency (P_{eff}) is calculated as throughput values (T_{enc}), as shown in Fig. 1, divided by the difference value between power consumption at idle and encryption times (P_{diff}) in Table 6, which is the throughput value per watt, as presented in Fig. 5.

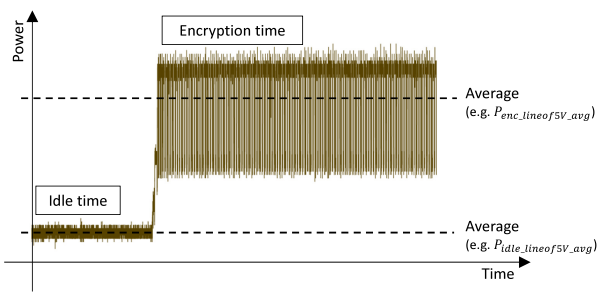


Fig. 3 Measurement concept.

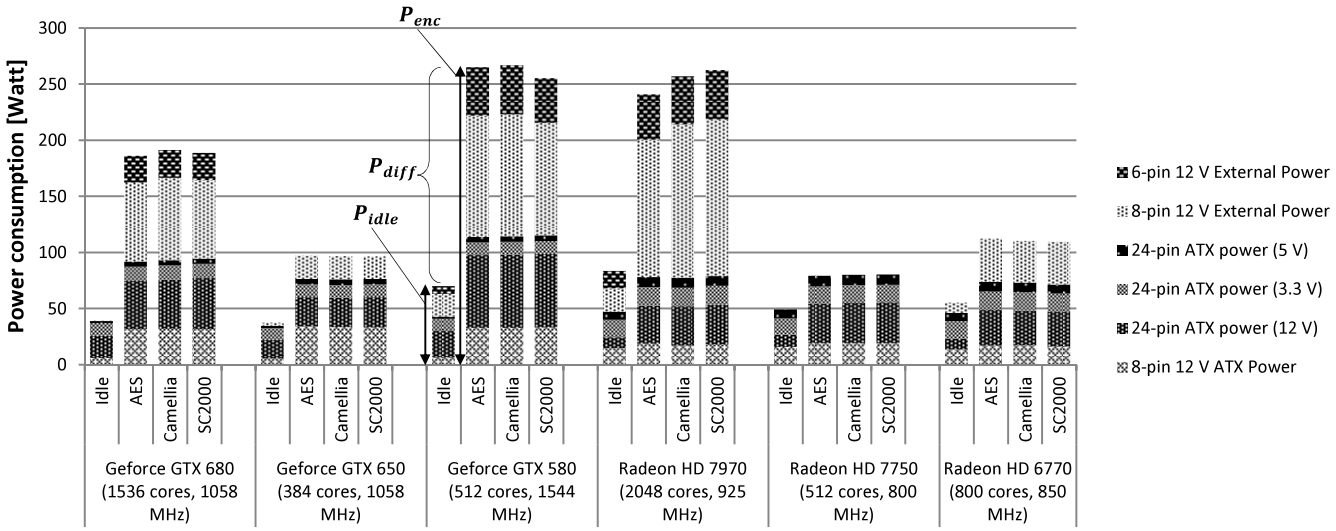


Fig. 4 Power consumption measurement results.

Table 5 Range of standard deviation for power consumption through power lines in each environment. [Watt]

	Geforce GTX 680	Geforce GTX 650	Geforce GTX 580	Radeon HD 7970	Radeon HD 7750	Radeon HD 6770
Idle	0.4–5.8	0.2–5.5	0.4–5.6	0.5– 1.8	0.8–2.3	0.5–1.8
AES	0.4–2.6	0.2–1.0	0.3–4.5	0.6–29.8	0.7–7.0	0.6–2.5
Camellia	0.4–2.7	0.2–1.3	0.4–4.1	0.6–31.7	0.7–7.5	0.6–2.5
SC2000	0.4–2.4	0.2–1.3	0.4–3.6	0.6–29.6	0.7–7.1	0.7–2.5

Table 6 Difference of power consumption at idle and at encryption times (P_{diff}). [Watt]

	Geforce GTX 680	Geforce GTX 650	Geforce GTX 580	Radeon HD 7970	Radeon HD 7750	Radeon HD 6770
AES	147.7	59.4	194.9	157.5	30.1	57.0
Camellia	153.1	59.2	196.9	173.5	31.1	54.8
SC2000	150.3	58.9	185.3	179.0	31.3	53.9

Results showed that, whereas P_{eff} of AES-128 on GTX 680 and GTX 650 were, respectively, 0.43 Gbps/W and 0.27 Gbps/W, that on GTX 580 was 0.39 Gbps/W. Kepler architecture is designed for better power efficiency. However, in reality, P_{eff} on GTX 680 and GTX 650 were almost as much as that on GTX 580. The main reason is that the power efficiency of the Kepler GPUs was hampered by the low throughput shown in Fig. 1. In contrast, whereas P_{eff} of AES-128 on Radeon HD 7970 was 1.3 Gbps/W, that on Radeon HD 6770 was 0.68 Gbps/W. The former value was about 1.9 times higher than the latter one. The power efficiency increased drastically with increasing throughput. Results show that P_{eff} on Radeon HD 7970 was about 3.0 times higher than that on GTX 680. Similarly, encryption processing on Radeon HD 7750 with the fewest cores in GCN architecture exploits higher power efficiency than that on GTX 650. Therefore, GCN GPUs are expected to be considerably strong co-processors for use with encryption processing.

Like the evaluation of encryption throughput described in Sect. 6.2, we cite the power efficiency on AMD Phenom II X6 1100T from our previous work [3]. Implementation methodology and measurement environment are the same as the one described in Sect. 6.2. The power efficiency of AES-128, Camellia-128, and SC2000-128 are 0.07 Gbps/W, 0.06 Gbps/W, and 0.07 Gbps/W, respectively. These values are, respectively, only 5.4%, 4.7%, 3.7% of those on AMD Radeon HD 7970 described in this paper.

7. Micro-Benchmark Analysis

As described in Sect. 6.2, encryption throughput on GTX 680 was much lower than that on Radeon HD 7970 with GCN architecture as well as that on GTX 580 with Fermi architecture of one generation earlier.

Here, as described in Sect. 3, block cipher algorithms comprise accesses to substitution tables and round keys in addition to arithmetic and logical instructions. However, the numbers of accesses to substitution tables and round keys are much smaller than that of arithmetic and logical instructions. For example, in SC2000 block cipher, the 128-bit block is encrypted by 72 times of substitution tables, 56 times of 32-bit round keys, and 506 times of arithmetic and logical instructions. Therefore, encryption throughput value is largely affected by arithmetic and logical instructions such as XOR. For this reason, we investigate the throughput of

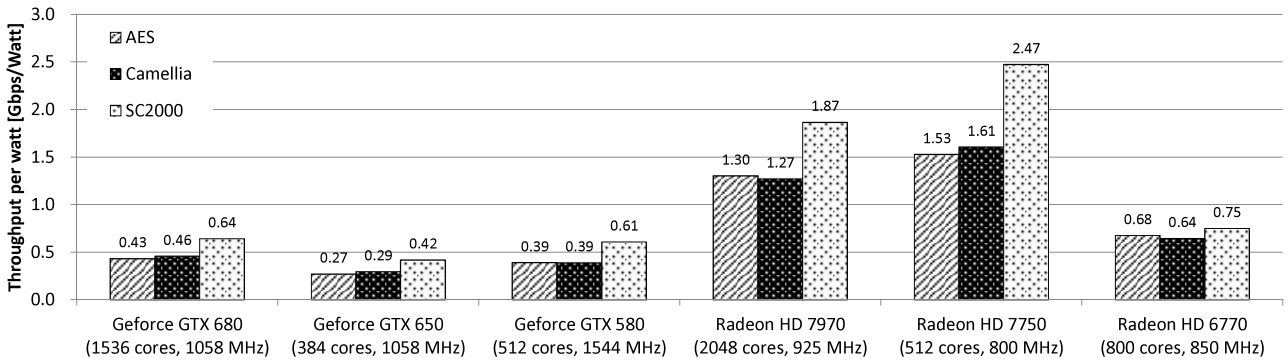


Fig. 5 Comparison of power efficiencies of block ciphers on a GPU (P_{eff}).

arithmetic and logical instructions of GPUs using micro-benchmarks.

7.1 Experimental Setup

Unfortunately in OpenCL, no built-in function to measure work-item execution cycles inside of a kernel such as `clock()` is available. Therefore, we first run an empty kernel eliminating a sequence of instructions as well as the original micro-benchmark kernel, measuring both the elapsed times using `clGetProgramInfo` API, thereby obtaining the difference between the two values as instruction latency. Then we obtained elapsed cycles by dividing the elapsed time by the reciprocal of the GPU core clock frequency.

7.2 Design of the Micro-Benchmark

As a kernel of micro-benchmark of arithmetic and logical instructions we designed, that for XOR instruction is shown in Fig. 6. This is true because XOR instruction is often used in block cipher algorithms, for example for combination of plaintext with round key. Regarding other instructions, only operators of the fifth line in Fig. 6 are changed. Our micro-benchmark for arithmetic and logical instructions is designed to run a kernel of a sequence of their dependent arithmetic and logical instructions in an unrolled loop, based on some previous works [23], [24]. Instruction latency is adopted as the average value. We set the number of work-groups to six times the number of CUs in each GPU to keep the CUs busy.

Moreover, to validate our micro-benchmark, we obtain a disassembled code of the micro-benchmark kernel of Fig. 6. In Nvidia OpenCL, we first use `clGetProgramInfo` API to dump the ptx code [25] for Kepler architecture. Next, we use Nvidia's nvcc compiler [26] to compile the ptx code to the binary file for the architecture using the flag "`-cubin -arch=sm_30`". Then we disassemble the binary file to the code in Fig. 7 using Nvidia's cuobjdump [27]. In contrast, in AMD OpenCL, the disassembled code of Fig. 8 are obtained to set the environment variable `AMD_OCL_BUILD_OPTIONS_APPEND=-save-temps` and run the kernel.

```
1  // Kernel for instructions (XOR)
2  __kernel void inst_latency(__global uint out, uint
     p1, uint p2, int its){
3    uint lid=get_local_id(0);
4    unsigned int a=p1, b=p2;
5    for(int i=0; i<its; i++) repeat256(a^=b; b^=a)
6    out[lid]=a+b;
7  }
```

Fig. 6 Kernel of our micro-benchmark (for XOR instruction).

```
 1  code for sm_30
 2  Function : inst_latency
 3  MOV R1, c [0x0] [0x44];
 4  MOV R4, RZ;
 5  MOV R2, c [0x0] [0x144];
 6  MOV R3, c [0x0] [0x148];
 7  MOV R0, RZ;
 8  CAL 0x13b0;
 9  ISETP.LT.AND P0, pt, RZ, c [0x0] [0x14c], pt;
10  @!P0 BRA 0x12b0;
11  LOP.XOR R2, R2, R3;
12  LOP.XOR R3, R3, R2;
13  LOP.XOR R2, R2, R3;
14  LOP.XOR R3, R3, R2;
15  LOP.XOR R2, R2, R3;
16  LOP.XOR R3, R3, R2;
17  ...
```

Fig. 7 Disassembled code of micro-benchmark in Fig. 6 (for GTX 680 in Nvidia OpenCL environment).

7.3 Micro-Benchmarking Result

7.3.1 Disassembled Micro-Benchmark Code

The disassembled codes for Geforce GTX 680 in Nvidia OpenCL and for Radeon HD 7970 in AMD OpenCL environment are displayed respectively in Figs. 7 and 8. In both Nvidia and AMD OpenCL environments, the assembly code is obtainable by disassembling the micro-benchmark. The 11th and subsequent lines in Fig. 7 and the 29th and subsequent lines in Fig. 8 indicate that the micro-benchmark of Fig. 6 is compiled correctly to a series of instructions respectively in Nvidia and AMD OpenCL environments. We confirm that the number of instructions in the respectively disassembled code coincides with the number of iterations of the instructions in the micro-benchmark.

```
 1  ; -------- Disassembly --------------------
 2  shader main
 3   asic(SI_ASIC)
 4  type(CS)
 5
 6  s_buffer_load_dword  s0, s[8:11], 0x04
 7  s_buffer_load_dword  s1, s[8:11], 0x18
 8  s_waitcnt     lgkmcnt(0)
 9  s_min_u32     s0, s0, 0x0000ffff
10  v_mov_b32     v1, s0
11  v_mul_i32_i24  v1, s16, v1
12  v_add_i32     v0, vcc, v0, v1
13  s_buffer_load_dword  s0, s[12:15], 0x00
14  s_buffer_load_dword  s2, s[12:15], 0x04
15  s_buffer_load_dword  s3, s[12:15], 0x08
16  s_buffer_load_dword  s8, s[12:15], 0x0c
17  v_add_i32     v0, vcc, s1, v0
18  s_movk_i32    s1, 0x0000
19  label_000E:
20  s_waitcnt     lgkmcnt(0)
21  s_cmp_le_i32  s8, s1
22  s_cbranch_scc0  label_0016
23  s_branch      label_101C
24  s_nop         0x0000
25  s_nop         0x0000
26  s_nop         0x0000
27  s_nop         0x0000
28  label_0016:
29  s_xor_b32     s2, s2, s3
30  s_xor_b32     s3, s3, s2
31  s_xor_b32     s2, s2, s3
32  s_xor_b32     s3, s3, s2
33  s_xor_b32     s2, s2, s3
34  s_xor_b32     s3, s3, s2
35  ...
```

Fig. 8 Disassembled code of micro-benchmark in Fig. 6 (for Radeon HD 7970 in AMD OpenCL environment).

7.3.2 Measurement Result

The measurement results of throughput of arithmetic and logical instructions of Nvidia and AMD GPUs are presented respectively in Tables 7 and 8. In the both tables, the instructions indicated by boldface are included in block cipher algorithms and used in the performance evaluation on a GPU in Sect. 6. In particular, the values in shaded cells are the instruction throughputs of Kepler architecture, which are dealt with as main analysis subject in this section. Furthermore, considering that GPUs are generally recognized as powerful floating-point architecture, we also measure and list the throughputs of float and double instructions in these tables for reference. The values are all the throughput obtained through the above kernel program divided by the number of work-groups. In addition, each work-group is assigned to each CU as described in Sect. 2. Therefore, these values in both Tables represent the throughput per CU. In addition, instruction throughputs measured by assembly level benchmarking researched by Lai et al. in CUDA platform [28] are listed in Table 9, for comparison with those of Kepler architecture.

As shown by boldface in Tables 7 and 8, the throughputs of major arithmetic and logical instructions for block cipher algorithms, for example XOR instruction, of Radeon HD 7970 and of Geforce GTX 580 saturate respectively at about 64 ops/cycle and about 31 ops/cycle, which values roughly coincide with the numbers of the cores per CU in the respective architectures. This fact indicates that such instructions are processed ideally in parallel through PE pipelines in these GPUs. Incidentally, the measurement results of instruction throughputs of GTX 580 roughly coincide with those of GTX 480 with Fermi architecture, as reported previously in [23].

However, as shown by boldface in shaded cells in Table 7, the throughput of these instructions on GTX 680 does not achieve 192 ops/cycle, the value of which is disproportionate to the number of processor cores per CU in a Kepler GPU. Consequently, there are two considerations involved with the software and hardware. First, an important fact is that the instruction throughputs of Kepler GPU fluctuate greatly depending on the source register's allocation. Assembly level benchmarking by Lai et al. uncovered the fact that the registers in Kepler GPU are interleaved like the shared memory [28]. In their work, they also indicate that careful allocation of the source registers can avoid the register bank conflict. In fact, the arithmetic and float instruction throughputs we measured in Table 7 coincide with the result in Table 9, which are described in the best or worst cases of Lai's work. Therefore, we speculate that completely eliminating the register bank conflict would be difficult because determining the register's allocation at the compilation phase to avoid a bank conflict. Second, the fact is that even though Nvidia's compiler is improved in the future, the instruction throughputs in Table 7 do not surpass about 132 ops/clk described in Table 9, of which the value is optimized at the assembly level in the Kepler GPU, which indicates that only 132 work-item instructions per clock cycle at most are issued to a CU. The value is much lower than the PE's processing throughput (192 ops/clk). Therefore, encryption throughputs of cryptographic applications deteriorate because of the heavy use of arithmetic and logical instructions.

In addition, as an example of throughputs of arithmetic and logical instructions at various batch sizes, measurement results of XOR throughput on Geforce GTX 680 and Radeon HD 7970 are presented respectively in Fig. 9. As described in Sect. 2, a unit of each work-item batch differs in the two GPU architecture, with 32 work-items for Kepler and 64 work-items for GCN. Results show that to extract high throughput of arithmetic and logical instructions on a GPU, it is necessary that the number of work-item batches be activated sufficiently to keep the pipeline of the PEs full.

7.3.3 Example Cases and Scenarios

This paper explores the causes of performance difference between Kepler and GCN architectures and points out the bottleneck in Kepler architecture for arithmetic and logical instructions throughputs. Thus, we expect that these analysis results would be useful for those who try to implement these instructions bound applications (i.e., hash functions and error-correcting codes) on GPUs.

Table 7 List of throughputs of arithmetic and logical instructions on Nvidia GPUs. [ops/cycle]

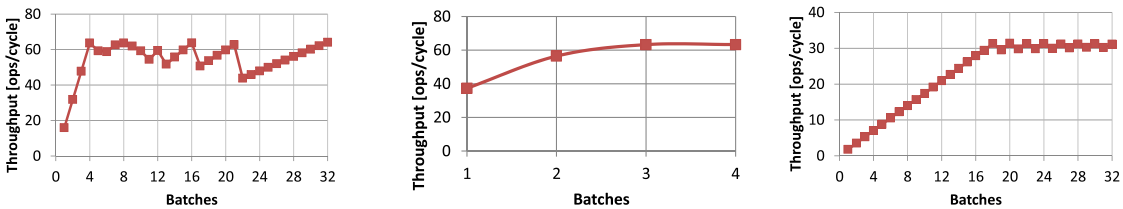
GPU				Geforce GTX 680	Geforce GTX 650	Geforce GTX 580
Architecture, # Cores				Kepler, 192 cores/CU		Fermi, 32 cores/CU
OpenCL micro-benchmark in Fig. 6	**Arithmetic**	**ADD, SUB**	**For use in block ciphers**	**63.7**	**63.8**	**31.1**
		MUL	**For use in block ciphers**	**32.3**	**32.0**	**15.9**
		MAD	**For use in block ciphers**	**32.3**	**32.0**	**15.9**
	Logical	**XOR, AND, OR**	**For use in block ciphers**	**63.7**	**63.8**	**31.1**
		SHL, SHR	**For use in block ciphers**	**31.3**	**31.0**	**15.6**
	Float	ADD	For comparison	64.6	63.8	31.8
		MUL	For comparison	64.6	63.8	31.8
	Double	ADD	For comparison	8.0	8.0	4.0
		MUL	For comparison	8.0	8.0	4.0

Table 8 List of throughputs of arithmetic and logical instructions on AMD GPUs. [ops/cycle]

GPU				Radeon HD 7970	Radeon HD 7750
Architecture, # Cores				GCN, 64 cores/CU	
OpenCL micro-benchmark in Fig. 6	**Arithmetic**	**ADD, SUB**	**For use in block ciphers**	**63.3**	**64.0**
		MUL	**For use in block ciphers**	**63.3**	**64.0**
		MAD	**For use in block ciphers**	**31.6**	**31.9**
	Logical	**XOR, AND, OR**	**For use in block ciphers**	**63.3**	**64.0**
		SHL, SHR	**For use in block ciphers**	**63.3**	**64.0**
	Float	ADD	For comparison	63.1	61.2
		MUL	For comparison	63.1	62.1
	Double	ADD	For comparison	31.6	8.0
		MUL	For comparison	15.8	4.0

Table 9 List of instruction throughputs on Geforce GTX 680 cited from [28]. [ops/cycle]

GPU			Geforce GTX 680
Architecture, # Cores			Kepler, 192 cores/CU
Assembly level benchmark in CUDA platform	Arithmetic	ADD	66.2–132.4
		MUL	33.2
		MAD	26.5–33.2
	Float	ADD	66.2–132.0
		MUL	66.2–132.0



(a) Geforce GTX 680 (Kepler, 192 cores/CU) (b) Radeon HD 7970 (GCN, 64 cores/CU) (c) Geforce 580 (Fermi, 32 cores/CU)

Fig. 9 Throughput of XOR instructions on Kepler and GCN GPUs.

8. Conclusion

We evaluated the throughput and the power efficiency of three 128-bit block ciphers (AES, Camellia, and SC2000) on GPUs with Kepler and GCN architectures. Moreover, for comparison against these GPUs, we used two GPUs with one-generation-earlier architecture. From the experimentally obtained results, GCN GPUs designed for general purpose computing engender extremely high throughput and power efficiency of encryption processing. In contrast, as for Kepler GPUs developed for better power efficiency, the throughput values of arithmetic and logical instructions comprising a large part of block cipher are not commensurate with the number of the processor cores. As a result, for example, the throughput of AES-128 on the Geforce GTX 680 was 31.2% of that on Radeon HD 7970. Power consumption of encryption processing on Kepler GPUs is certainly lower than that on GTX 580 with Fermi architecture, but the power efficiency was hampered by the low throughput. Thereby, the power efficiency on Kepler GPUs was lower than that on Radeon HD 7970. Consequently, at this time, we recommend GCN GPUs as a strong co-processor of block ciphers. To ameliorate Kepler GPUs as a co-processor of block ciphers, the arithmetic and logical instructions must be improved from both sides of software and hardware.

Our future work will include evaluation of encryption processing on mobile-type GPUs such as Intel Iris Graphics and ARM Mali GPUs.

Acknowledgment

This work was supported by JSPS KAKENHI Grant-in-Aid for Young Scientists (B) Number 25871223.

References

[1] The Green500 List, http://www.green500.org/lists/green201306
[2] R. van Nieuwpoort and J.W. Romein, "Correlating radio astronomy signals with many-core hardware," Int. J. Parallel Programming, vol.39, no.1, pp.88–114, 2011.
[3] N. Nishikawa, K. Iwai, and T. Kurokawa, "Power efficiency evaluation of block ciphers on GPU-integrated multicore processor," Proc. 12th International Conference on Algorithms and Architectures for Parallel Processing – Volume Part I, ICA3PP'12, Berlin, Heidelberg, pp.347–361, Springer-Verlag, 2012.
[4] GTC Keynote with J.-H. Juang, "GPU technology conference 2013."
[5] Press conference, "Computex Taipei 2013."
[6] Q. Li, C. Zhong, K. Zhao, X. Mei, and X. Chu, "Implementation and analysis of AES encryption on GPU," Proc. 2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE Ninth International Conference on Embedded Software and Systems, HPCC'12, Washington, DC, USA, pp.843–848, IEEE Computer Society, 2012.
[7] O. Gervasi, D. Russo, and F. Vella, "The AES implantation based on OpenCL for multi/many core architecture," 2010 International Conference on Computational Science and Its Applications, pp.129–134, 2010.
[8] X. Wang, X. Li, M. Zou, and J. Zhou, "AES finalists implementation for GPU and multi-core CPU based on OpenCL," Proc. 2011 IEEE International Conference on Anti-Counterfeiting, Security and Identification, ASID 2011, pp.38–42, 2011.
[9] Khronos Group, "Open compute language."
[10] National Institute of Standards and Technology (NIST), FIPS-197 Advanced Encryption Standard (AES), 2001.
[11] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, and T. Tokita, "Camellia: A 128-bit block cipher suitable for multiple platforms – Design and analysis," Selected Areas in Cryptography, pp.39–56, 2000.
[12] T. Shimoyama, H. Yanami, K. Yokoyama, M. Takenaka, K. Itoh, J. Yajima, N. Torii, and H. Tanaka, "The block cipher SC2000," Revised manuscripts from the Eighth International Workshop on Fast Software Encryption, FSE'01, London, UK, pp.312–327, Springer-Verlag, 2002.
[13] J. Daemen and V. Rijmen, The Design of Rijndael: AES – The Advanced Encryption Standard, Springer Verlag, Berlin, Heidelberg, New York, 2002.
[14] The IEEE Security in Storage Working Group, "XTS block cipher-based mode (XEX-based tweaked-codebook mode with ciphertext stealing)," http://siswg.net/
[15] N. Nishikawa, K. Iwai, and T. Kurokawa, "High-performance symmetric block ciphers on multicore CPU and GPUs," IJNC, vol.2, no.2, pp.251–268, 2012.
[16] D.A. Osvik, J.W. Bos, D. Stefan, and D. Canright, "Fast software AES encryption," FSE, pp.75–93, 2010.
[17] A.D. Biagio, A. Barenghi, G. Agosta, and G. Pelosi, "Design of a parallel AES for graphics hardware using the CUDA framework," Parallel and Distributed Processing Symposium, International, pp.1–8, 2009.
[18] OpenMP Architecture Review Board, "OpenMP application program interface."
[19] AMD Corp., Accelerated Parallel Processing SDK v2.8, 2013.
[20] X. Ma, M. Dong, L. Zhong, and Z. Deng, "Statistical power consumption analysis and modeling for GPU-based computing," Proc. ACM SOSP Workshop on Power Aware Computing and Systems HotPower, 2009.
[21] H. Nagasaka, N. Maruyama, A. Nukada, T. Endo, and S. Matsuoka, "Statistical power modeling of GPU kernels using performance counters," Proc. International Green Computing Conference, 2010.
[22] Y. Zhang, Y. Hu, B. Li, and L. Peng, "Performance and power analysis of ATI GPU: A statistical approach," Proc. 2011 IEEE Sixth International Conference on Networking, Architecture, and Storage, NAS'11, pp.149–158, 2011.
[23] A. Resios and V. Holdermans, GPU performance prediction using parameterized models, Master Thesis of Utrecht University, 2009.
[24] H. Wong, M.M. Papadopoulou, M. Sadooghi-Alvandi, and A. Moshovos, "Demystifying GPU microarchitecture through microbenchmarking," 2010 IEEE International Symposium on Performance Analysis of Systems Software, ISPASS, pp.235–246, 2010.
[25] NVIDIA Corp., Parallel Thread Execution ISA v3.2, 2013.
[26] NVIDIA Corp., NVIDIA CUDA Programming Guide 4.2, 2012.
[27] NVIDIA Corp., cuobjdump Application Note version 03, 2011.
[28] J. Lai and A. Seznec, "Performance upper bound analysis and optimization of SGEMM on Fermi and Kepler GPUs," CGO, pp.1–10, 2013.

Naoki Nishikawa received B.E. and M.E. degrees from National Defense Academy of Japan in 2005 and 2010, respectively. From 2010 to 2011, he was affiliated with the Technical Research and Development Institute. He is now a Ph.D. student at the National Defense Academy of Japan. His research interests include cryptography and parallel computing. He won the IPSJ Computer Science Research Award for Young Scientists in 2010.

Keisuke Iwai completed the doctoral program in the Department of C.S. at Keio University in 1998 and fulfilled credit requirements in 2001. He holds a D.Eng. degree. He is now a lecturer in the Department of C.S., National Defense Academy of Japan. He is engaged in research on automatic parallel compilers and multi-processor systems.

Hidema Tanaka received B.E., M.E, and Ph.D. degrees in electrical engineering from the Science University of Tokyo, in 1995, 1997, and 2000, respectively. He was a senior researcher of Security Fundamentals Group at the National Institute of Information and Communications Technology until 2011. Currently, he is an Associate Professor in the Department of C.S., National Defense Academy of Japan. He was awarded one of the SCIS2003 paper prizes.

Takakazu Kurokawa received his B.S. degree from the Department of E.E, Keio University, in 1983 and completed the doctoral program in 1988. He holds a D.Eng. degree. He is now a Professor in the Department of C.S., National Defense Academy of Japan. He is engaged in research on dedicated computers and cryptography.