DOI: 10.3745/KIPSTA.2010.17A.6.275

CELL 프로세서를 이용한 SEED 블록 암호화 알고리즘의 효율적인 병렬화 기법

김 덕 호[†] · 이 재 영[††] · 노 원 우[†††]

요 약

본 논문에서는 Cell BE 프로세서를 사용한 효율적인 병렬 블록 암호화 알고리즘을 제시한다. 제안하는 알고리즘은, 이종 프로세서인 Cell BE의 특성을 효율적으로 활용하기 위하여 PPE와 SPE에 서로 다른 부호화/복호화 방식을 적용하여 그 성능을 개선하였다. 본 논문에 제시된 구현 방식을 바탕으로 검증된 결과에 따르면, 제안하는 알고리즘은 고성능 네트워크 시스템을 지원할 수 있는 2.59Gbps의 성능을 보여준다. 이는, 다른 다중 코어 프로세서의 병렬 구현 방식과 비교할 때, 1.34배 증가된 성능의 부호화/복호화 속도를 제공한다.

키워드 : SEED 블록 암호 알고리즘, Cell BE 프로세서, 병렬화, 다중 코어 프로세서

An Efficient Parallelized Algorithm of SEED Block Cipher on Cell BE

Deokho Kim[†] · Jaeyoung Yi[††] · Won Woo Ro[†††]

ABSTRACT

In this paper, we discuss and propose an efficiently parallelized block cipher algorithm on the CELL BE processor. With considering the heterogeneous feature of the CELL BE architecture, we apply different encoding/decoding methods to PPE and SPE and improve the throughput. Our implementation was fully tested, with execution results showing achievement of high throughput, capable of supporting as high network speed as 2.59 Gbps. Compared to various parallel implementations on multi-core systems, our approach provides speedup of 1.34 in terms of encoding/decoding speed.

Keywords : SEED block cipher, Cell BE, parallelization, multi-core processor

1. Introduction

As the need for information security increases in our everyday life, the procedure of encoding/decoding data becomes a critical issue in data network systems. Indeed, cryptography has been a major application domain of the diplomatic and military areas. Moreover, the importance of cryptography is continuously growing in our current world of informational society, highly used in electronic commerce, electronic signature, or digital authorization. Cryptosystems must ensure that private information does not leak out to unauthorized users.

To this point, there have been various encryption algorithms such as *DES*(Data Encryption Standard)[1], *AES*(Advanced Encryption Standard)[2], and *FEAL*(Fast data Encipherment ALgorithm)[3] developed. In fact, with high network transmission rate, the process of encryption/decryption is one of the major bottlenecks in contemporary systems[4]. As a result, high-speed encoding is required especially when sending a large amount of important information with high-speed transmission or on Virtual Private Networks(VPN).

The performance requirement of the cryptosystem includes high computational ability, high data throughput, and adaptability to the protocol changes. To address this request, the Cell Broadband Engine(Cell BE) architecture is an attractive match for the cryptosystem implementation. The Cell BE processor provides a multiple number of general purpose programmable cores targeting

※ 본 연구는 2009년 한국산학협동재단의 지원을 받아 수행되었음
† 준 회 원 : 연세대학교 전기전자공학과
†† 정 회 원 : LG전자 CTO System IC 연구원
††† 종신회원 : 연세대학교 전기전자공학과 조교수
논문접수 : 2010년 9월 9일
수 정 일 : 1차 2010년 10월 28일
심사완료 : 2010년 11월 15일

a broad set of workloads, intensive multimedia and scientific processing. Since many cryptographic algorithms consist of a large amount of homogenous computations, using the Cell BE can exploit a high level of parallelism and achieve increased computational performance which will ensure high data throughput.

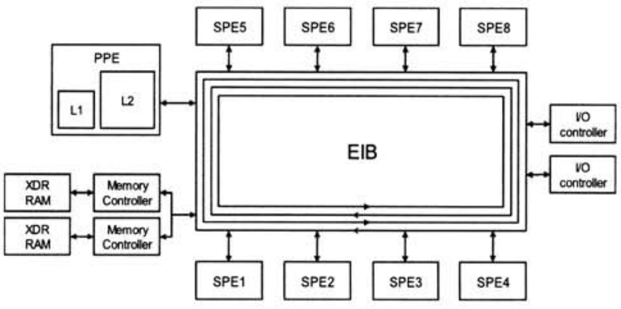
Our contribution in this paper is that we develop an efficient parallel algorithm of the SEED block cipher on the Cell BE architecture with proper task mapping and load balancing. The design is fully functional and we have achieved a 2.59 Gbps throughput. The results are superior to the performance achieved by other parallel implementations on the various multi-core general purpose processor platforms.

2. Background

2.1 The Cell BE processor

The Cell Broadband Engine(Cell BE) is a multi-core processor developed by Sony, IBM, and Toshiba in 2000. The Cell BE processor contains one Power Processor Element(PPE), eight Synergistic Processor Elements (SPEs), Direct Memory Access(DMA), and synchronization mechanisms in order to communicate with each element, and the Element Interconnection Bus(EIB). It runs on a clock frequency of 3.2GHz and has features of Single Instruction Multiple Data(SIMD) execution units, high power- and area-efficiency, large memory bandwidth, a large bandwidth on-chip coherent bus, and high-bandwidth flexible I/O[5].

(Figure 1) represents the Cell BE architecture schematically. The Cell BE processor has two channels of 256 MB high-bandwidth DRAM memory. The central EIB is a coherent bus that can transfer up to 96 byte/s. It consists of four 16 byte rings which can transfer data in only one direction, and each ring supports up to three simultaneous data transfers. However, despite of some limitation the Cell BE supports six SPEs for programming on PlayStation 3.



(Fig. 1) The Cell Broadband Engine architecture

The PPE contains a 32 KB instruction memory, 32 KB data L1 cache, and 512 KB L2 cache. The PPE is a dual-threaded, dual-issue(In-order issue), 64-bit Power-architecture processor with AltiVec vector execution unit. The dual-issue design is optimized by interleaving instructions from two computational to maintain maximum efficiency. The PPE can use VMX(Vector Multimedia eXtensions), which was developed for the IBM power PC processors, with AltiVec unit.

The SPE is composed of 256 KB Local store, 128 bit SIMD unit and MFC(Memory Flow Controller). The MFC supports naturally aligned DMA transfer sizes of 1, 2, 4 or 8 bytes, and multiples of 16 Bytes, with a maximum transfer size of 16 KB per transfer with DMA command. On a SPE, it consumes only two cycles for simple fixed point operation, six cycles for single-precision floating point and load instructions. Moreover double-precision has maximum issue rate of one SIMD instruction per seven cycles.

The Cell BE processor uses 256 MB of the Rambus XDR DRAM memory. This memory delivers 12.8 GB/s per 32-bit memory channel and the two channels are supported on the Cell BE processor for a total bandwidth of 25.6 GB/s.

The Cell processor provides a SIMD feature in the vector unit on the PPE and in the SPEs[6]. SIMD units have been demonstrated to be effective in accelerating required computation for multimedia applications.

The PPE has VMX SIMD units called AltiVec. The AltiVec unit supports a floating point and integer SIMD instruction set designed by the Apple, the IBM and the Freescale Semiconductor. The trade name is owned solely by the Freescale Semiconductor and the Apple refers it to as Velocity Engine. It supports for 16-way parallelism for 8-bit, 8-way parallelism for 16-bit and 4-way parallelism for 32-bit signed and unsigned integers and IEEE floating-point numbers with separate 128-bit wide, 32-entry register files.

The SPE has a SIMD unit enhanced than AltiVec units on the PPE. The SPE does not have separated register file but it has same register file for vector and scalar execution units. The register file has 128-entries and 128 bits wide. The SIMD unit supports including what the AltiVec supports and 2-way parallelism for 64-bit signed and unsigned integers and IEEE double precision numbers[7].

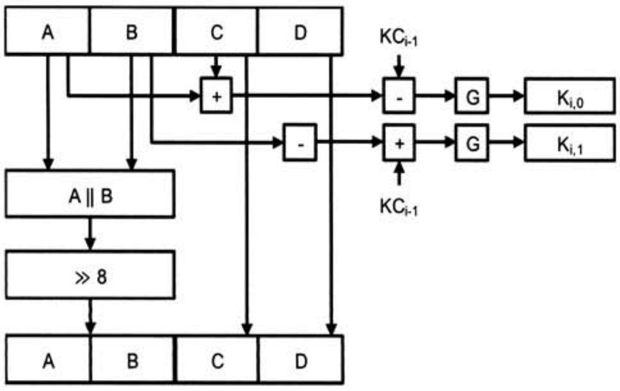
2.2 Overview of SEED algorithm

The SEED block cipher algorithm has been developed by KISA(Korea Information Security Agency) in 1998, by the government's concern on the importance of cipher systems. It has become a national standard since year 2000, and has been adapted to most of the security systems[8].

The SEED design consists of the round-key generator, F-function, G-function, and S-boxes. The SEED algorithm processes a block size of 128-bit plain-text using a 128-bit cipher key, producing a 128-bit cipher-text. It is a private key algorithm, meaning that it uses the same key for encryption and decryption, and has a *Feistel* structure[9] with 16 rounds, in order to make it secure. The 128-bit input text stream is divided into two 64-bit blocks, *L0* and *R0*. The first output *R1* is the result of *L0* exclusive-or *F(R0)*. On the other side, *R0* bypasses to *L1* which becomes one of the two inputs to Round 2. After processing the data through 16 rounds, they will result in two 64-bit blocks forming the 128-bit cipher-text output.

The round-keys are created by shift of bits, arithmetic operations, G functions and constants of golden ratio. The input user-key has 128-bit length and is separated into four 32-bit blocks and the input generates couple of round keys for 16 rounds.

(Figure 2) depicts the algorithm that generates round keys $K_{i,0}$ and $K_{i,1}$ for i_{th} round. KC_i means i_{th} golden ration constant and || means concatenation of two data blocks. Two 32-bit blocks, located on *C* and *D*, are not change during round. In the next round the *C* and *D* of previous rounds inserted to location of *A* and *B*. In addition, direction of shift operation changes opposite at every round. In first round, if shift operation shifts concatenation of A and B to right then, shift operator at next round shifts the concatenated blocks to left. The round-keys are generated with the algorithm until it produces keys for 16 rounds



(Fig. 2) Round-key creation algorihtm

The F-function of SEED has a Feistel structure, and provides resistance against differential cryptanalysis, linear analysis, and other known attacks. (Figure 3) represents the structure of F-function. The F-function divides 64-bit block input into two 32-bit blocks and processes them through a mixture of *xors*, additions(mod 32) and G-functions, by inputting 64 bit round keys $K_{i,0}$ and $K_{i,1}$.

The F-function is represented as following equations. Where C and D are upper and lower half of 64 bit inputs respectively and C′ and D′ are result of F-functions.

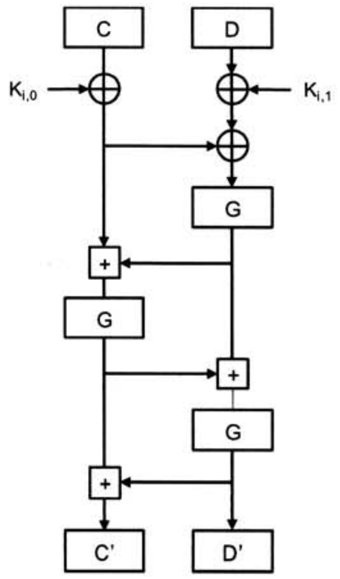
$$C' = G[G[G\{(C \oplus K_{i,0}) \oplus (D \oplus K_{i,1})\} \boxplus (C \oplus K_{i,0})] \boxplus G\{(C \oplus K_{i,0}) \oplus (D \oplus K_{i,1})\}] \boxplus G[G\{(C \oplus K_{i,0}) \oplus (D \oplus K_{i,1})\} \boxplus (C \oplus K_{i,0})]$$

$$D' = G[G[G\{(C \oplus K_{i,0}) \oplus (D \oplus K_{i,1})\} \boxplus (C \oplus K_{i,0})] \boxplus G\{(C \oplus K_{i,0}) \oplus (D \oplus K_{i,1})\}]$$

Where($a \oplus b$: *a* bit-wise exclusive-or *b*, $a \boxplus b$:(a+b) mod 2^{32})

The G-function is the main function used in the F-function and also in the round key generation. A 32-bit input is divided into four 8-bit blocks which are passed through S-boxes. The outputs of S-boxes are processed with bitwise *and*, then *xor* operations to produce the 32-bit output of the G-function. The two S-boxes S_1 and S_2 are represented by two lookup tables[8]. The S-boxes are derived from nonlinear functions, defined as following.

$$S_i : Z_{2^8} \rightarrow Z_{2^8}, S_i(x) = A^i \cdot x^{n_i} \oplus b_i$$



(Fig. 3) Structure of F-function

where $n_1 = 247$, $n_2 = 251$, $b_1 = 169$, $b_2 = 56$

However the non-linear transformation has large time to calculate on programming language code. The G function is reconstructed as four SS-boxes. The SS-boxes can be formulated as following equations

$$SS_3 = S_2(X_3)\&m_2 \parallel S_2(X_3)\&m_1 \parallel S_2(X_3)\&m_0 \parallel S_2(X_3)\&m_3$$
$$SS_2 = S_1(X_2)\&m_1 \parallel S_1(X_2)\&m_0 \parallel S_1(X_2)\&m_3 \parallel S_1(X_2)\&m_2$$
$$SS_1 = S_2(X_1)\&m_0 \parallel S_2(X_1)\&m_3 \parallel S_2(X_1)\&m_2 \parallel S_2(X_1)\&m_1$$
$$SS_0 = S_1(X_0)\&m_3 \parallel S_1(X_0)\&m_2 \parallel S_1(X_0)\&m_1 \parallel S_1(X_0)\&m_0$$

where $\parallel$ is concatenation and a & b: a bit-wise and b

The result of G-function is derived from following equation.

$$Z = SS_3(X_3) \oplus SS_2(X_2) \oplus SS_1(X_1) \oplus SS_0(X_0)$$

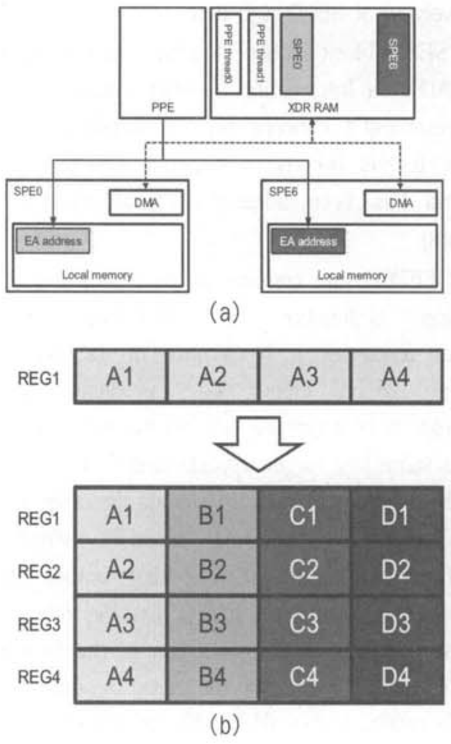
where $\oplus$ is a bit-wise exclusive-or.

3. SEED implementation on Cell BE

The Cell BE architecture is very efficient and is suitable for exploiting parallelism of the SEED algorithm. In fact, our SEED algorithm is implemented on the Cell BE as a manner of a multi-threaded program. The encryption component of the algorithm which is distributed in 16 rounds occupies a significant percentage of the total execution time. Consequently, this step is implemented to run in parallel on the six SPEs available in the programming level on the Cell BE platform.

Our design uses the Cell BE's heterogeneous structure appropriately with considering the load balancing between the PPE and SPEs. Both PPE and SPEs execute the 16 rounds of the algorithm, with computation of F-functions and G-functions. In addition, the PPE creates the round keys and manages the SPEs transporting data and signals to the SPEs with DMA commands.

The detailed operation for data transfer is described in (Figure 4)-(a). The data is transferred from the main memory(XDR RAM in the (Figure 4)-(a)) to the SPE's local memory with main memory address called *effective address*. The PPE first sets up environment variables for each of the SPE threads. Then, it assigns each of the threads to the dedicated SPE. Each SPE checks the effective address whether it is null or not, then SPE starts DMA command to transfer the necessary data with the 64 bit effective address. After waiting data transfer, SPE processes 16 round operations of SEED.



(Fig. 4) Issues in implementation

Memory alignment is also taken into consideration when developing the parallelized SEED algorithm as shown in (Figure 4)-(b). The 128-bit input in the algorithm is divided into four 32-bit parts, and then processed with bitwise operations and shift operations. In order to take advantage of the SIMD structure of the SPEs, data are aligned in the transfer process from the main memory to the SPE's local memory. Instead of receiving the 128-bit input data on one 128-bit register, the data is broken up into four parts to go into four registers. Three other 128-bit input data are decomposed and distributed into the four registers in the same way. This allows the SIMD units to simultaneously execute the same instruction on four 32-bit data. Accordingly, the encoding/decoding throughput is increased by four times

As for the S-box function, all of the possible output values for input 0 to 255 are pre-calculated, and a lookup table is constructed for each box. Consequently, the processing time of the S-box is insignificant; each output value can be accessed directly from the input address. Because of this lookup method, we cannot exploit the SIMD unit for the S-box operations. The SIMD unit can only process the same single instruction for different data, so it cannot access different indices of the lookup table at one time.

In addition to the six SPEs, we also use PPE to encrypt data in order to fully utilize the computing power of Cell BE. Since the PPE controls execution of the SPEs as well as has different computing power compared to the SPEs, we need to adjust and balance the portion of workload assigned to the PPE.

As the PPE has a dual-threaded and dual-issue architecture, the PPE runs two threads whereas the SPEs runs one thread on each core. To consider the workload balancing, we have set workload ratio as the amount on a PPE thread over the amount on a SPE thread.

To utilize the workload balancing, we fully consider the architectural design of the PPE. Although the PPE has a SIMD execution unit, it is physically near to the main memory and has cache different from the SPEs. As the existence of the cache, the encoding/decoding with the scalar execution unit can provide performance gain.

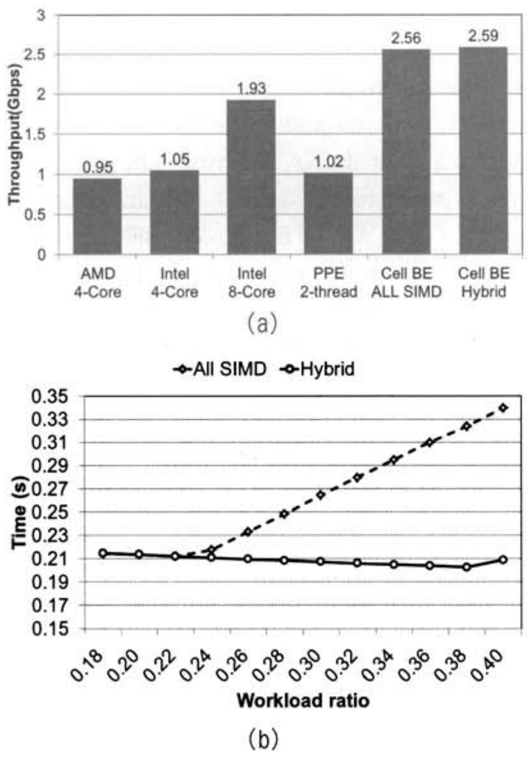
Since the SIMD execution unit needs a memory alignment overhead and has large granularity, which is 4 × 128 bits for every operation, rather than that of scalar execution unit which is 128 bits for every operation, the workload ratio can be controlled more in details with using scalar execution unit. Therefore, we propose a new method which uses different execution unit for the PPE and the SPEs.

We labeled an approach that uses the SIMD units on both cores as ALL SIMD and the method that does not use the SIMD unit on PPE as Hybrid. The Hybrid implementation provide efficient workload balancing and it achieves high-performance for the encoding/decoding speed.

4. Experimental results and performance analysis

To demonstrate correctness of the proposed idea, we have tested it thoroughly on Cell BE with diverse test vectors. With -O3 optimizations applied on the GCC compiler, we achieved a throughput of 2.59 Gbps for both SEED encoding and decoding operations. We achieved the peak performance with the workload ratio of 0.38 and the ratio is obtained in a heuristic method.

In (Figure 5)-(a), we have parallelized and compared the performance on the Cell BE to the results implemented on various other multi-core platforms. The parallelization on the homogeneous processors equally divides entire works to the each core on the processor and we have fully parallelized it by the Pthread library. As seen in the (Figure 5)-(a), the experiments were performed on various desktop computing environments, including an AMD quad-core *phenom-X4 9550*, an Intel quad-core *core 2 Quad Q9400* and 8-core *Xeon E5440* × 2 system. Furthermore, the PPE 2-*thread* configuration



(Fig. 5) Performance results and platform information

has been tested and compared, which only uses the PPE for the encoding/decoding operations. Several data were input and encrypted in parallel to exploit the multi-core environment, using all the available cores to their fullest. The Cell BE implementation, which is depicted as Cell BE Hybrid, shows approximately speed up of 1.34 and 2.54 in total computation performance compared to the Intel 8-core and PPE core system, respectively.

(Figure 5)-(b) is a result about adjusting workload distributed to PPE and adopting encoding/decoding methods. We achieve higher performance and higher workload ratio by using selectively applying encoding / decoding methods in the Hybrid approach.

The encoding/decoding time increases in (Figure 5)-(b) is caused by the unbalanced workload distribution. If the workload is unbalanced, the core that has more computation needs to wait until the other cores finish their works. Therefore, the unbalanced workload distribution causes severe performance degradation. As a result, the ALL SIMD method shows decreased performance after the optimal workload ratio and the Hybrid method also shows increase of encoding/decoding time after its optimal workload ratio.

For the ALL SIMD and the Hybrid methods, we can achieve the workload ratio up to 0.38 with the Hybrid implementation while ALL SIMD achieves 0.23. Performance without adjusting workload of ALL SIMD and Hybrid shows 0.74 Gbps and 1.21 Gbps respectively. With workload balancing, we can achieve more than twice better performance.

In the general purpose processor system, the cache contributes a great deal in achieving high performance, as it reduces the memory accesses which require a long access time. The six SPEs in Cell BE have no cache nevertheless show a speed up of 1.34 in encoding performance compared to the best 8-core system. In addition, it shows speed up of 2.54 compared to the result of using only the PPE. This indicates that the architecture of Cell BE is more suitable for such computational encryption processes than general multi-core systems. Accordingly, using the Cell BE implementation can fulfill the needs of fast encryption speed required in high-performance network systems, where current available desktop can only provide limited performance.

5. Conclusion

In this paper, we have shown a high-performance parallelized implementation of the SEED block cipher algorithm on the Cell BE processor. The proposed design is fully parallelized and provides 2.59 Gbps performance.

This is a sufficient performance rate to prevent the SEED block cipher from being a bottleneck in high-performance network systems, where the encoding and decoding speed of network security algorithms is crucial. As SEED is a widely used algorithm in Korea, a nation where high-speed network transmission rates are widely provided, we are confident to claim that our Cell BE implementation would be of great use.

References

[1] NBS, "Data Encryption Standard," FIPS, pub. 46, U.S, DEPARTMENT OF COMMERCE/National Institute of Standards and Technology, Jan. 1997.

[2] NBS, "Announcing the ADVANCED ENCRYPTION STANDARD(AES), FIPS, pub. 197, U.S, DEPARTMENT OF COMMERCE/National Institute of Standards and Technology, Nov. 2001.

[3] S. Miyaguchi, "The FEAL cipher family," *proceedings of the 10 th Annual International Crpytology Conference on Advances in Crpytology*, p.627-638, August 11-15, 1990.

[4] H. Xie, L. Zhou, and L. Bhuyan, "Architectural Analysis of Cryptographic Applications for Network Processors," *IEEE First Workshop on Network Processors*, 2002.

[5] J.A. Kahle, M.N. et al. "Introduction to the Cell multiprocessor." *IBM J. RES. & DEV*, Vol.49, No.4/5, pp.589-604, 2005.

[6] Dac C. pham. et al. "Overview of the Architecture, Circuit Design, and Physical Implementation of a First-Generation Cell Processor." *Solid-State Circuits*, Vol.41, No.1, pp.179-196, 2006.

[7] A. Arevalo, R. M. Matinata, M. R. Pandian, E. Peri, K. Ruby, F. Thomas, and C. Almond, *Programming the Cell Broadband Engine Architecture: Examples and Best Practic*, Vervante, 2008.

[8] Korea Information Security Agency, A Design and Analysis of 128-bit Symmetric Block Cipher(SEED), 1999. 4.

[9] Bruce Schneier, Applied Cryptography, Wiley, 1996.

김 덕 호

e-mail : nautes87@yonsei.ac.kr
2010년 연세대학교 전기전자공학과(학사)
2010년 연세대학교 전기전자공학과(석사과정)
관심분야 : 컴퓨터 시스템, 병렬 프로세싱, Cell 프로세서 등

이 재 영

e-mail : jaeyoung.yi@lge.com
2008년 연세대학교 수학과/컴퓨터과학과(학사)
2010년 연세대학교 전기전자공학과(석사)
현재 LG전자 CTO System IC 연구원
관심분야 : 임베디드 시스템, 컴퓨터 아키텍처 등

노 원 우

e-mail: wro@yonsei.ac.kr
1996년 연세대학교 전기공학과(학사)
1999년 University of Southern California (석사)
2004년 University of Southern California (공학박사)
2003년~2004년 Apple Computer Inc. 인턴 연구원
2004년~2007년 California State University 전기 및 컴퓨터공학과 조교수
2006년~2007년 ARM Inc. 소프트웨어 엔지니어
2007년~현 재 연세대학교 전기전자공학과 조교수
관심분야 : 고성능 마이크로프로세서 디자인, 컴파일러 최적화, 임베디드 시스템 디자인 등