International Journal of Electrical and Computer Engineering (IJECE)
Vol. 8, No. 3, June 2018, pp. 1798~1804
ISSN: 2088-8708, DOI: 10.11591/ijece.v8i3.pp1798-1804

Dynamic Frequency Scaling Regarding Memory for Energy Efficiency of Embedded Systems

Junha Kim, Moonju Park
Departement of Computer Science and Engineering, Incheon National University Incheon, Korea

Article Info

Article history:

Received Nov 8, 2017
Revised Jan 11, 2018
Accepted Jan 18, 2018

Keyword:

Embedded systems
Energy efficiency
Frequency scaling
Memory
Power consumption

ABSTRACT

Memory significantly affects the power consumption of embedded systems as well as performance. CPU frequency scaling for power management could fail in optimizing the energy efficiency without considering the memory access. In this paper, we analyze the power consumption and energy efficiency of an embedded system that supports dynamic scaling of frequency for both CPU and memory access. The power consumption of the CPU and the memory is modeled to show that the memory access rate affects the energy efficiency and the CPU frequency selection. Based on the power model, a method for frequency selection is presented to optimize the power efficiency which is measured using Energy-Delay Product (EDP). The proposed method is implemented and tested on a commercial smartphone to achieve about 3.3% - 7.6% enhancement comparing with the power management policy provided by the manufacturer in terms of EDP.

Copyright © 2018 Institute of Advanced Engineering and Science.
All rights reserved.

Corresponding Author:

Moonju Park,
Departement of Computer Science and Engineering,
Incheon National University,
119 Academy Road, Yeonsu-gu, Incheon, Korea 22012.
Email: mpark@inu.ac.kr

1. INTRODUCTION

Dynamic Voltage/Frequency Scaling (DVFS) has been used to reduce power consumption of computing systems. DVFS is a technique that increases or decreases the supply voltage by adjusting the operating frequency of CMOS circuits. CMOS circuits have static and dynamic power dissipation, and dynamic power dissipation is the dominant component in CMOS [1]. Most research on DVFS technique has focused on CPU DVFS [2], [3] because the CPU is the most power-consuming device when a computer system is actively running. Many contemporary OSs support DVFS of CPU. Linux's *cpufreq* [4] subsystem is an example.

The frequency scaling technology is supported in hardware devices other than CPU such as the GPU or the memory bus, in such cases the operating frequency of the device can be managed by the user. For example, Linux system has a subsystem called *devfreq* to support frequency scaling of devices other than CPU [5]. Nexus 6 smartphone is a commercial mobile device supporting device frequency scaling, which allows us to adjust the clock speed of the memory bus that affects the memory bandwidth. Changing the frequency to access memory gives us another option to manage the power consumption of embedded systems.

Attempts to manage the power consumption of memory access have been recently made. In [6], they proposed a DVFS method for DRAM based on memory bandwidth utilization. They devised a bandwidth-based frequency selection policy using their finding in experiments that memory latency is not significantly affected by the memory frequency at low bandwidth. But because memory hardware with DVFS support was not available, they emulated frequency scaling using timing delays. No DVFS is supported by DRAM so far because scaling of IO voltage on DRAM affects the stability and requires significant hardware change, but

DFS (Dynamic Frequency Scaling) is possible. In [6], low-power mode of DRAM by DFS was introduced to achieve energy consumption reduction with limited hardware change. The results of [7] were further extended to consider both CPU and DRAM power consumption in a server [8]. In [9], power management of DRAM using both DFS and low-power states is modeled and studied using simulation. The joint scaling of CPU and DRAM frequencies was also studied in [10] for server systems. Low power design of SRAM can be also considered as in [11]. But in this paper, we focus on DRAM power management.

Previous works require hardware changes for memory frequency scaling to manage DRAM power consumption. In this letter, we propose a power management method by combining DVFS of CPU and DFS of memory bus. We show that CPU and memory are closely related in the view of energy efficiency, which depends on the number of memory access per instruction. From the relationship, we find an optimal frequency ratio between the CPU frequency and the memory frequency.

The study was performed using a real device. The target device used in this study is a commercial smartphone, Nexus 6, which has Snapdragon 805 CPU with 3GB lpDDR3 SDRAM. The CPU frequency can be set to one of 18 levels (300, 422.4, 652.8, 729.6, 883.2, 960, 1036.8, 1190.4, 1267.2, 1497.6, 1574.4, 1728, 1958.4, 2265.6, 2457.6, 2496, 2572.8, 2649.6 MHz) and the memory bus frequency can be set to one of 13 levels (50, 75, 100, 150, 200, 259, 307, 393, 460, 528, 662, 796, 1065 MHz).

This paper is organized as follows. Section 2 explains the power model of the CPU and the memory that is used for the analysis on the relationship between the CPU and the memory frequencies in Section 3. The analysis in Section 3 shows the CPU frequency and the memory frequency are closely related in terms of energy efficiency. Based on the analysis, Section 3 presents a method for frequency selection of both the CPU and the memory. In Section 4, experimental results with a commercial smartphone on which our frequency selection method implemented are presented. Finally, Section 5 concludes our work.

2. POWER MODEL OF CPU AND MEMORY

The power consumption of the CPU in embedded systems is usually divided into dynamic and static power [12]. The power consumption of the CPU can be modeled as:

$$P_{CPU} = P_{dyn} + P_{static} \approx \alpha V^2 F_c + IV \tag{1}$$

where α is a coefficient of the switching activity and the effective capacitance, V is the operating voltage, F_c is the CPU frequency, and I is the leakage current. Reduction in operating voltage decreases the dynamic power consumption, but increases the circuit delay. The relation between the operating voltage and the CPU frequency is given by:

$$F_c \propto \frac{(V_c - V_{th})^2}{V_c} \tag{2}$$

where V_{th} is a threshold voltage which is much smaller than the operating voltage [13], [14]. Equation (2) can be rewritten as $F_c \propto V$, from which we can reformulate Equation (1) as:

$$P_{CPU} = P_{dyn} + P_{static} \approx \beta {F_c}^3 + \gamma F_c \tag{3}$$

where β is a variable depending on the switching activity and γ is a hardware-dependent constant. For multicore CPUs, the power consumption will be given by summing of each core power, that is, $P_{cpu} \approx \sum(\beta_i F_c^2 + \gamma F_c)$ where i represents the core number. Switching activities may differ from each other.

On the other hand, the power consumption of the DRAM system can be divided into operation power and background power [6], [9]. The operation power is the power required to execute memory reads and writes. The background power accounts for all power consumption when there is no memory access. Lowering the frequency to access memory affects the power consumption; it lowers background power linearly [7]. The operation power is not affected by memory frequency, but the energy required for memory access increases because the access time becomes longer. For the DDR-series DRAMs, the background power is a major component in the total DRAM power consumption [6]. So we assume that the operation power can be ignored in our model, and the power consumption of the main memory is modeled as:

$$P_{MEM} \approx \rho F_m \tag{4}$$

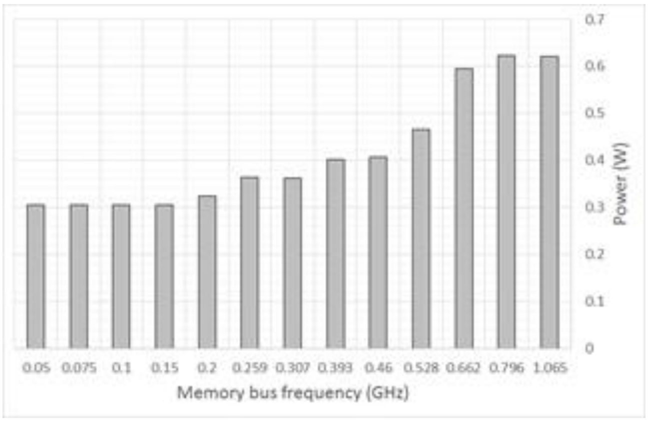


Figure 1. Power consumption of Nexus 6 in idle state with different bus frequencies

where F_m is the memory device frequency, and ρ is a hardware-dependent constant. Combining Equations (3) and (4), we have power consumption estimation:

$$P = P_{CPU} + P_{MEM} \approx \beta {F_c}^3 + \gamma F_c + \rho F_m \qquad (5)$$

assuming the CPU and the memory are the dominating power consumption devices.

We have measured power consumption of the target device, varying the bus frequency when the system is in idle state ($\beta = 0$). Results are shown in Figure 1. Because the operating frequency of the SDRAM ranges from 166MHz to 800MHz, the power consumption is almost unchanged below 200MHz and above 796MHz. Changing the CPU frequency from the lowest level to the highest level in idle state does not affect the power consumption. The power consumption at the lowest bus frequency is about 0.305W and at the highest it is about 0.621W. The difference between the maximum and the minimum is about 0.316W and $\rho \approx 0.498$ when F_m is represented in GHz.

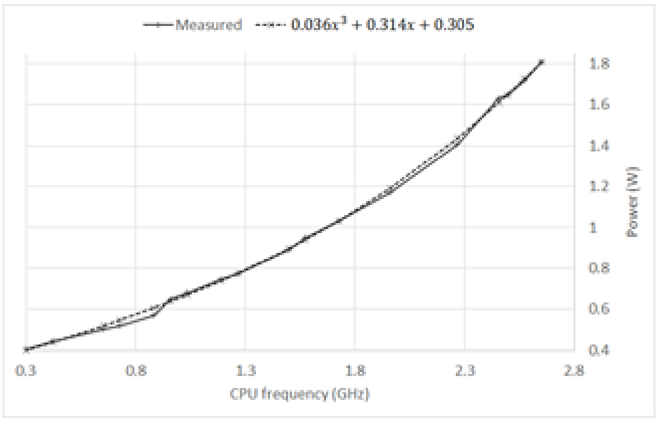


Figure 2. Power consumption for CPU intensive benchmark (one core)

To obtain the values β and γ we used *cpubomb* included in Isolation Benchmark Suite [15] that fully utilizes the CPU and does not access memory. Figure 2 shows the power consumption of *cpubomb* for different CPU frequencies when only one core is used for the benchmark. Memory bus frequency was fixed at the lowest level, so we assume 0.305W is consumed by memory device. Regression analysis gives us $\beta \approx 0.036$ and $\gamma \approx 0.314$, which provide estimation very close to the measured ones. The hardware dependent parameters are used to estimate for a multicore application using the relationship $P_{cpu} \approx \sum(\beta_i F_c^3 + \gamma F_c) = (\sum \beta_i) F_c^3 + n\gamma F_c$ where n is the number of cores used. The comparison between the estimated values and the measured values is shown in Figure 3. The estimation error is accumulated as the number of cores increases: 1.5%, 3.1%, 5.2%, and 7.0% of average error in prediction for 1, 2, 3, and 4 cores respectively. Because the target was overheated when multicores are used with over 2.4GHz of CPU frequency, we could not measure accurate values over 2.4GHz. In this letter, we simply let β represents $\sum \beta_i$.

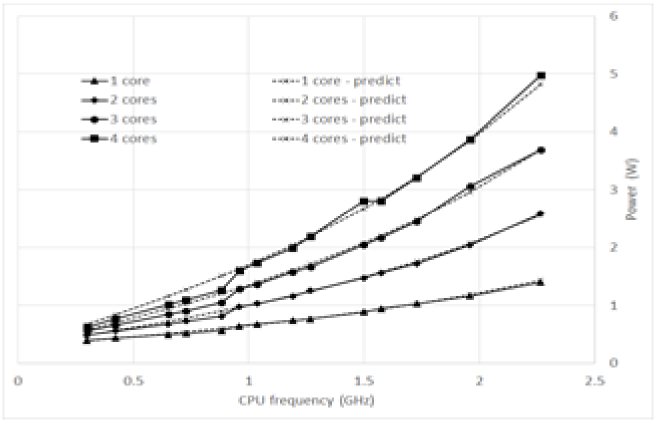


Figure 3. Power estimation for multicore execution (CPU intensive)

3. CPU AND MEMORY FREQUENCY SELECTION FOR ENERGY EFFICIENCY

We use EDP (Energy Delay Product) [16] as the measure of energy efficiency to consider both turnaround time and energy consumption. The energy-delay product has been widely used as a metric to measure the energy efficiency coupling both the energy consumption and performance. It is the multiplication of the delay time (execution time) until the end of the program and the energy consumption during the execution of the program. Because the energy consumption of executing an instruction is the multiplication of the power consumption and the execution time of the instruction, the EDP of executing an instruction is modeled as

$$EDP \approx (\beta {F_c}^3 + n\gamma F_c + \rho F_m) \times \left(\frac{CPI}{F_c}\right)^2 = (\beta F_c + \frac{n\gamma}{F_c} + \frac{\rho F_m}{F_c^2}) \times (CPI)^2 \tag{6}$$

where CPI is the Cycles Per Instruction and n is the number cores used. CPI is affected by the memory frequency if there is a memory access. For the RISC CPU such as ARM processor, if we let CPI_0 be the CPI when there is no memory access, CPI can be estimated as

$$CPI \approx (1 + \delta \cdot F_c/F_m)CPI_0 \tag{7}$$

where δ is 1 if there is a memory access, otherwise 0. Assuming CPI_0 is a constant value, minimizing the EDP is equivalent to minimizing the following:

$$\begin{aligned} \mathrm{F}(F_c, F_m) &= (\beta F_c + \frac{n\gamma}{F_c} + \frac{\rho F_m}{F_c^2}) \times \left(1 + \delta \cdot \frac{F_c}{F_m}\right)^2 \\ &= \left(\frac{F_m + \delta F_c}{F_m F_c}\right)^2 \cdot (\beta {F_c}^3 + n\gamma F_c + \rho F_m) \end{aligned} \tag{8}$$

If $\delta = 0$, we have

$$\mathrm{F}(F_c, F_m) = \beta F_c + \frac{n\gamma}{F_c} + \frac{\rho F_m}{F_c^2} \tag{9}$$

and the optimal value of F_c can be obtained when F_m is at its minimum. When $\delta = 1$, because harmonic mean is not larger than geometric mean, we have

$$\mathrm{F}(F_c, F_m) \geq \frac{4}{F_m F_c} \cdot P = 4\left(\frac{\beta F_c^2}{F_m} + \frac{n\gamma}{F_m} + \frac{\rho}{F_c}\right) \tag{10}$$

where P is given in Equation (5). So the optimal value of F_c can be found when F_m is at its maximum. Thus, when α is the rate of memory access per instruction, the expected $\mathrm{F}(F_c, F_m)$ can be found by minimizing the following:

$$\begin{aligned}&(1-\alpha)\left(\beta F_c + \frac{n\gamma}{F_c} + \frac{\rho F_m}{F_c^2}\right) + 4\alpha\left(\frac{\beta F_c^2}{F_m} + \frac{n\gamma}{F_m} + \frac{\rho}{F_c}\right)\\ &=\left(1-\alpha+4\alpha\cdot\frac{F_c}{F_m}\right)\left(\beta F_c + \frac{n\gamma}{F_c} + \frac{\rho F_m}{F_c^2}\right)\end{aligned} \tag{11}$$

By letting $\frac{F_c}{F_m} = R$ in (11), we have

$$(1-\alpha+4\alpha R)\left(\beta F_c + \frac{1}{F_c}\left(n\gamma + \frac{\rho}{R}\right)\right) \tag{12}$$

With CPU frequency F_c given, we can calculate the frequency ratio R minimizing Equation (12) as

$$R = \sqrt{\frac{(1-\alpha)\rho}{4\alpha(\beta F_c^2 + n\gamma)}} \tag{13}$$

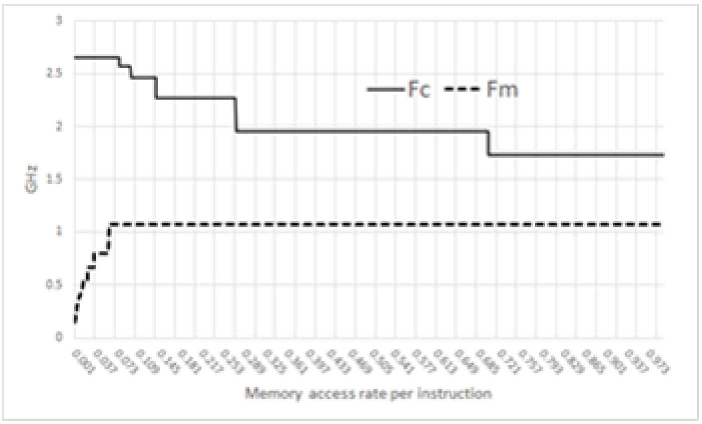


Figure 4. Frequency pair for optimal EDP

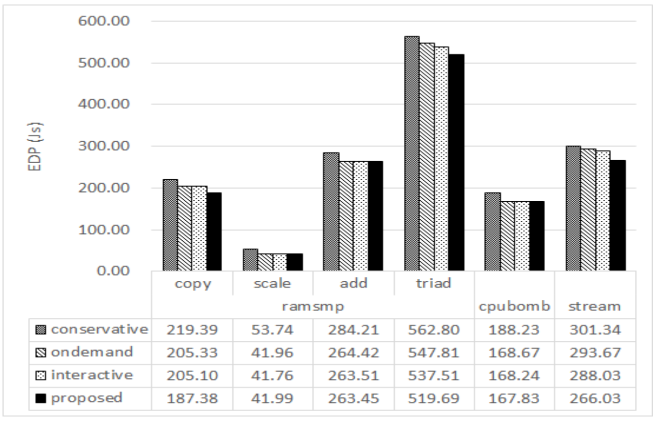
If $\alpha = 0$, $R \to \infty$ so F_m will be set to the minimum value. If $\alpha > 0$, the value of R can be calculated with a given F_c, then we get the corresponding value of F_m. With limited number of CPU frequency levels, we can calculate the value of F_m for each F_c with given utilization and the average memory access rate per instruction. Then we compare the corresponding energy consumption using Equation (11) to determine the pair of F_c and F_m that give the minimum value. As an example, the values of F_c and F_m obtained for a single core application are shown in Figure 4 (α is in 0.1%-99.9%, increased by 0.1%). The highest memory bus frequency is used as a bound to indicate that the optimal F_m is higher than 800MHz. The results show that if the memory access rate is less than 0.3% we do not need to raise the memory access frequency from its lowest level. With less than 3.5% of the memory access rate, the frequency should be maintained below its highest level.

4. APPLICATION TO A REAL TARGET

We measured energy consumption and performance of applications on a real target (Nexus 6) to validate our analysis. To measure the power and energy consumption, we disassembled the battery parts of Nexus 6 and connected the charging port to a digital power meter (ODROID Smart Power was used) which supports 10Hz sampling rate. We tested three benchmarks: *cpubomb*, *ramsmp* [17], and *STREAM* [18]. *ramsmp* and *STREAM* have 4 kinds of operations: *copy*, *scale*, *add*, and *triad*. *Copy* moves data in an array to another. *Scale* multiplies a value to data from an array and stores it to another. *Add* adds data from two arrays then stores the sum to the other array. *Triad* combines *scale* and *add*. Operations of *ramsmp* were tested separately, but those of *STREAM* were tested all together for comparison. The ranges of memory access rate

α for benchmarks are: 0.276-0.702 for copy, 0.016-0.023 for scale, 0.06~0.07 for add, 0.016-0.023 for triad of *ramsmp*, 0.001 for *cpubomb*, and 0.14-0.70 for *STREAM*. The CPU utilization is 100% for all benchmarks.

We compared the energy efficiency of different governors of Linux with the presented method. Our method was implemented using the governor interface of Linux and the sampling rate of our policy is the same as other governors. Linux supports 3 dynamic policies for CPU DVFS: *conservative*, *ondemand*, and *interactive*. The default CPU DVFS policy for Nexus 6 device is the *interactive* governor, which is typical for Android devices. The governor for the memory bus is *cpubw_hwmon*; it monitors the memory reads and writes and adjust the bus frequency according to the memory bandwidth. Note that our method has an integrated governor that performs both CPU DVFS and control of memory bus frequency simultaneously. Figure 5 compares the EDP of benchmarks.



	copy	scale	add	triad	cpubomb	stream
	ramsmp					
▩ conservative	219.39	53.74	284.21	562.80	188.23	301.34
▧ ondemand	205.33	41.96	264.42	547.81	168.67	293.67
▨ interactive	205.10	41.76	263.51	537.51	168.24	288.03
■ proposed	187.38	41.99	263.45	519.69	167.83	266.03

Figure 5. EDP of benchmarks

With *ramsmp*, frequency scaling of CPU and memory based on our analysis shows lowest EDP value in this experiment. EDP value is enhanced about 8.6% for *copy* operation and about 3.3 % for *triad* operation over the default governor. The energy efficiency was enhanced about 3.4% over *interactive* governor and 9.6% over *conservative* governor in total operations of *ramsmp*. Test with *STREAM* benchmark shows similar result: enhanced 7.6% over *interactive* and 11.7% over *conservative* governor. If memory is barely accessed, the proposed method does not degrade performance as in the results with *cpubomb*.

5. CONCLUSION

Although the CPU is the most power-consuming device in a computer system, memory also has the significant effects on power consumption as well as performance. Because of its impact on the performance, the memory is important especially in terms of energy efficiency. Thus frequency selection of CPU without considering the memory access could fail in optimizing the energy efficiency of the system. In this paper, we have analyzed the relationship between CPU and memory frequency in the view of energy efficiency. For CPU-intensive applications, lowering memory access frequency can reduce the power consumption of the system. For applications with considerable memory access, proper selection of CPU and memory frequency is needed. We presented a model for selection, and it was tested on a real target (Nexus 6 smartphone). The results show frequency assignments based on our analysis enhances energy efficiency.

ACKNOWLEDGEMENTS

This research was supported by Incheon National University Research Grant in 2016.

REFERENCES

[1] Burd TD, Brodersen RW, “Energy efficient CMOS microprocessor design”, *28th Annual Hawaii International Conference on System Sciences*, 1995, pp. 288-297.
[2] Weiser S, Welch B, Demers A, Shenker S, “Scheduling for reduced CPU energy”, *1st USENIX Conference on Operating Systems Design and Implementation*, 1996, pp. 13-23.
[3] Awadalla M, “Processor speed control for power reduction of real-time systems heuristically”, *International Journal of Electrical and Computer Engineering*, 2015, vol. 5, no. 4, pp. 701-719.

[4] Pallipadi V, Starikovskiy A, "The ondemand governor", *Linux Symposium*, 2006, vol. 2, pp. 215-230.
[5] Rao K, Yalamanchili S, Wardi Y, Wang J, Ye H, "Application-specific performance-aware energy optimization on Android mobile devices", *IEEE International Symposium on High Performance Computer Architecture*, 2017, pp. 169-180.
[6] David H, Fallin C, Gorbatov E, Hanebutte UR, Mutlu O, "Memory power management via dynamic voltage/frequency scaling", *8th ACM International Conference on Autonomic Computing*, 2011, pp. 31-40.
[7] Deng Q, Meisner D, Ramos L, Wenisch TF, Bianchini R, "MemScale: Active low-power modes for main memory", *16th International Conference on Architectural Support for Programming Language and Operating Systems*, 2011, pp. 225-238.
[8] Deng Q, Meisner D, Bhattacharjee A, Wenisch TF, Bianchini R, "CoScale: Coordinating CPU and memory system DVFS in server systems", *45th Annual ACM/IEEE International Symposium on Microarchitecture*, 2012, pp. 143-154.
[9] Lu Y, He B, Tang X, Guo M, "Synergy of dynamic frequency scaling and demotion on DRAM power management: Models and optimizations", *IEEE Transactions on Computers*, 2015, vol. 64, no. 8, pp. 2367-2381.
[10] Sundriyal V, Sosonkina M, "Joint frequency scaling of processor and DRAM", *Journal of Supercomputing*, 2016, vol. 72, no. 4, pp. 1549-1569.
[11] Rahman LF, Amir MFB, Reaz MBI, Marufuzzaman M, Husain H, "Advances on low power designs for SRAM cell", *TELKOMNIKA (Telecommunication Computing, Electronics and Control)*, 2014, vol. 12, no. 8, pp. 6063-6082.
[12] Mittal S, "A survey of techniques for improving energy efficiency in embedded computing systems", *International Journal of Computer Aided Engineering and Technology*, 2014, vol. 6, no. 4, pp. 440-459.
[13] Hong I, Kirovki D, Qu G, Potkonjak M, Srivastava M, "Power optimization of variable voltage core-based systems", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 1999, vol. 18, no. 12, pp. 1702-1714.
[14] Goyal S, "Design of low leakage multi-threshold (Vth) CMOS level shifter", *International Journal of Electrical and Computer Engineering*, 2013, vol. 3, no. 5, pp. 584-592.
[15] Matthews JN, Hu W, Hapuarachchi M, Deshane T, Dimatos D, Hamilton G, McCabe M, Owens J, "Quantifying the performance isolation properties of virtualization systems", *Workshop on Experimental Computer Science*, 2007, Article no. 6.
[16] Gonzalez R, Horowitz M, "Energy dissipation in general purpose microprocessors", *IEEE Journal of Solid-State Circuit*, 1996, vol. 31, no. 9, pp. 1277-1284.
[17] Bell C, Clark S, Radcliff R, Designing a memory benchmark. Whitepaper, Deopli Co. 2012; [Online] Available: http://www.deopli.com/docs/whitepapers/Designing_a_Memory_Benchmark.pdf. Accessed on: Jun. 27, 2017.
[18] McCalpin JD. Stream Benchmark. [Online] Available: https://www.cs.virginia.edu/stream/, Accessed on: Sept. 26, 2017.

BIOGRAPHIES OF AUTHORS

Junha Kim received his B.S. and M.S. degree from Dept. of Computer Science and Engineering, Incheon National University in 2015 and 2017 respectively. His research interests include embedded systems, operating systems, and power management systems.

Moonju Park received the Ph.D. degree in the School of Electrical and Computer Engineering, the M.E. degree in the Dept. of Computer Engineering, and the B.E. degree in the Dept. of Naval Architecture and Ocean Engineering from Seoul National University in 1995, 1998, and 2002 respectively. He has been with the Dept. of Computer Science and Engineering, Incheon National University since 2007, where he is currently an Associate Professor and the Director of the Computing and Information Center at INU from 2016. He was the chairman of PG601 in TC4 of Korea ICT Standardization Committee from 2010 to 2013. He served as Program Committee member of IEEE ISORC in 2014 and 2015, ACM RACS in 2011, 2012, and 2013. He worked at IBM Ubiquious Computing Lab. as an Advisory SW Engineer from 2006 to 2007. He was with LG Electronics as a Chief Research Engineer at Mobile Communications Lab. from 2002 to 2006. His research interests include real-time systems, embedded systems, operating systems, and scheduling problems.