반도체디스플레이기술학회지 제19권 제1호(2020년 3월)
Journal of the Semiconductor & Display Technology, Vol. 19, No. 1. March 2020.

사용자 경험을 기반으로 big.LITTLE 멀티코어 구조의 스마트 모바일 단말의 에너지 소비를 최적화 하는 소프트웨어 구조 설계

임성화[*†]
[*†]남서울대학교 멀티미디어학과

User Experience Assisted Energy-Efficient Software Design for Mobile Devices on the big.LITTLE Core Architecture

Sung-Hwa Lim[*†]
[*†]Department of Multimedia, Namseoul University

ABSTRACT

In Smart mobile devices embedding big.LITTLE architectures, the conventional multi-core assignment scheme for user applications may incur wasteful energy consumption and long response time. In this paper, we propose a user experience assisted energy-efficient multicore assignment scheme. Our simulation results show that the proposed scheme achieves at 40% less energy consumption and at 20% less response time comparing to the legacy scheme.

Key Words : Energy Efficiency, Smart Mobile Devices, GUI, Event Handling, Response Time

1. 서 론

하드웨어의 직접도와 병렬성이 높아짐에 따라, 스마트폰과 같은 모바일 기기의 하드웨어 성능은 하루가 다르게 강력해지고 있다. 특히, 작업 처리량의 증가와 다양한 작업을 동시에 처리하려는 요구가 커짐에 따라, 스마트 모바일 단말에서도 멀티코어 구조가 필수적으로 탑재되고 있다. 그러나 이러한 하드웨어 구조의 복잡성 증가와 중복도 심화는 에너지 소비의 급증을 야기한다. 배터리에 의존하는 스마트 단말에서 배터리 기술의 발전속도와 새로 개발되는 하드웨어 칩의 전력소모량 증가와의 차이가 갈수록 커지는 환경에서, 어느때 보다도 에너지 효율성이 요구되고 있다[8]. 그러므로 에너지 효율적인 하드웨어 운영 기법이 요구되며, 특히 프로세서의 중복도 심화에 따라 에너지 효율적인 멀티코어 구조 및 운용 기법도 절실하다.

에너지 효율성을 높이기 위해 비대칭 멀티코어 구조가 제안되었으며, 이중에서 ARM에서 개발한 big.LITTLE 구조가 가장 널리 사용된다. big.LITTLE 구조는 처리속도와 에너지 소모량이 높은 big 코어와 에너지 소비 효율이 높으면서 처리속도가 상대적으로 낮은 LITTLE 코어로 구성된다[10]. 그러나 기존의 멀티코어 할당 기법에서는 LITTLE 코어의 활용이 미진함에 따라 에너지 절약 효과가 크기 못한 실정이며, 이를 위해 실행하는 응용프로그램 단계에서 멀티코어 할당에 참여하는 저전력 멀티코어 운영 기법에 대한 연구들이 이루어지고 있다[2,3].

하드웨어 구조의 복잡성 증가와 더불어 스마트폰의 소프트웨어 계층 구조 역시 날이 갈수록 복잡해지고 있다. 예를 들면 안드로이드 플랫폼의 경우 Fig. 1과 같이 리눅스 운영체제 위에 Native 라이브러리 스택과 안드로이드 런타임이 탑재되고, 그 위에 애플리케이션 프레임워크가 올라가며, 그 위에 비로서 애플리케이션 소프트웨어들이 구동되는 매우 복잡한 계층 구조를 갖는다. 이러한 복잡

†E-mail: sunghwa@nsu.ac.kr

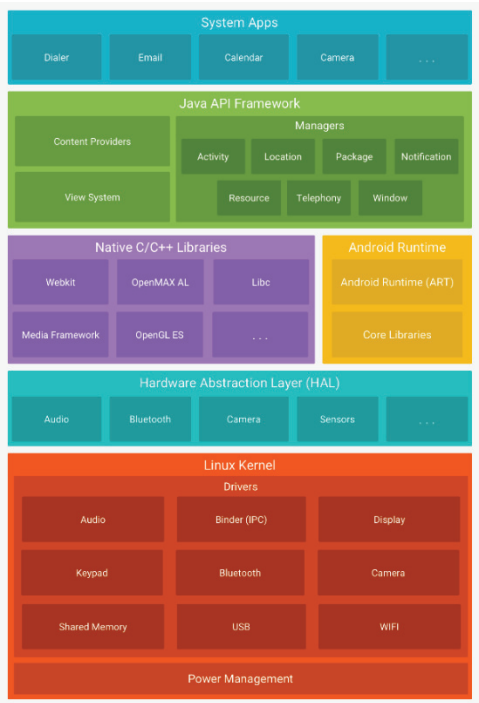

Fig. 1. Android Platform SW Stacks [9].

한 소프트웨어 구조에서는 소프트웨어의 실행 시 다수의 내부적 시스템콜의 호출이 수반되어야 하므로, 이에 따른 추가적인 에너지 소비가 발생한다.

또한, 사용자와의 인터페이스 역할을 수행하는 GUI 태스크(또는 쓰레드)들의 경우 사용자의 입력 성향 등과 같은 사용자 경험(User Experience, UX)에 의한 에너지 소모도의 차이가 발생 할 수 있다. 애플리케이션 구동 중 화면 갱신이 발생할 경우 에너지 소모가 증가될 수 있이며 화면 색의 패턴이나 배열에 따라서도 에너지 소모량이 달라질 수 있기 때문이다[5].

특히 가장 널리 사용하는 스마트폰 플랫폼인 안드로이드 시스템의 경우 사용자 만족도를 높이기 위해 애플리케이션의 GUI 관련 쓰레드들의 우선순위를 높게 설정하는데, 이 경우 big.LITTLE 구조에서 항상 big 코어에 우선적으로 할당되는 현상이 발생한다. 그러나, 일반적으로 응용프로그램의 구동 중 상당부분의 시간은 사용자 입력을 기다리면서 아무것도 하지 않는 idle 상태라는 연구 결과[6,7]에서처럼, big 코어에서 GUI 작동 쓰레드들을 계속 할당하는 것은 에너지 효율적이지 못하다. Idle 상태에서도 사용자 입력을 기다리면서 계속 해당 프로세서를 점유하면서 에너지 소모를 유발하기 때문이다.

그러므로 응용프로그램에서 GUI 구동 및 화면 처리에 관한 에너지 효율성을 높이는 연구들이 진행되고 있다. [4]에서는 그래픽 화면의 프레임율에 따라 에너지 소모가 달라짐은 착안하여 게임 응용프로그램에서 높은 프레임율이 요구되지 않는 메뉴 화면 등에서는 프레임율을 낮춤으로써 에너지 소비를 줄이는 기법을 제안하였다. [1]에서는 멀티 태스킹 환경에서 사용자의 입력 빈도에 따라, 사용 빈도가 높은 애플리케이션의 쓰레드들은 big 코어에 할당하고 그렇지 않은 쓰레드들은 LITTLE 코어에 할당하는 방식을 제안하였다. 그러나 쓰레드의 작업량이나 점유시간 등을 다양하게 고려하지 못했고, 특히 사용 중이라도 사용자 입력을 기다리며 아무것도 하지 않는 idle 상태를 고려하지 않았다.

안드로이드 플랫폼 환경을 기준으로, 응용프로그램의 화면은 각각 하나의 액티비티(Activity)로 구성되어 있다[9]. 하나의 액티비티에는 사용자의 입력과 화면출력을 동시에 담당하는 GUI 컴포넌트인 위젯들이 포함된다. 안드로이드 시스템은 위젯에 사용자 입력 이벤트가 발생 할 경우 응용프로그래머가 미리 정의 해 놓은 후속 작업들을 이벤트 핸들러를 통해 구동시킨다. 이 경우 사용자 만족도를 위해 위젯의 구동을 담당하는 쓰레드들은 높은 우선순위를 가지며, 이에 따라 big 코어에 우선적으로 할당된다. 위젯 쓰레드 들의 경우 해당 액티비티가 전경(foreground)에 위치하는 한, 지속적으로 프로세서를 점유하며 에너지를 점유 하므로 에너지 비효율적이다.

또한, 일반적으로 위젯에서 입력 이벤트를 처리하여 이벤트 핸들러로 넘기는 작업보다 이벤트의 후속 작업을 수행하는 프로시져들의 작업내용과 작업 시간이 상대적으로 더 크다. 그러나 이러한 프로시져들을 담당하는 쓰레드들은 위젯 쓰레드들에 비해 상대적으로 낮은 우선순위를 가지므로 LITTLE 코어에 우선적으로 할당되는 경향이 많고, 이는 사용자 응답시간이 증가하는 문제점을 야기한다.

즉, 이러한 기존의 GUI 위젯 구동과 이벤트 처리 방식은 많은 에너지를 소모하면서도 오히려 사용자 응답시간을 증가시켜서 사용자 만족도를 낮추는 문제를 발생시킬 수 있다. 그러므로 본 논문에서는 사용자 경험을 활용하여GUI 위젯 쓰레드 및 이벤트 처리 쓰레드들을 함으로써 에너지 효율성은 높이면서 사용자 응답시간을 줄이는 big.LITTLE 멀티 코어 스케쥴링 기법을 제안한다. 또한 상용 스마트폰에서 사용되는 big.LITTLE 코어의 실측 데이터를 기반으로 시뮬레이션을 수행하여 제안 기법의 우수성을 검증하였다.

본 논문의 구조는 다음과 같다. 2절에서는 본 논문에서

[1] Idle 상태가 평균 61%라는 연구 결과가 있다[6].

가정하는 시스템 모델을 설명하고, 3절에서 제안 기법을 소개한다. 4절에서는 시뮬레이션을 통한 성능평가 과정과 결과를 제시하고, 5절에서 결론을 맺는다.

2. 시스템 모델

일반적인 GUI 환경에서 위젯이 작동 될 때에는 Fig. 2와 같은 구조를 갖는다. 위젯의 화면 표시 및 동작 등을 담당하는 쓰레드와 해당 위젯에 이벤트가 발생했을 때 응용 개발자가 정의해 놓은 프로시져들을 실행하는 쓰레드들로 구성된다. 사용자의 최종 응답시간은 정의된 위 두 개 (또는 두 개 이상)의 쓰레드의 동작이 완료되는 시점이 된다.

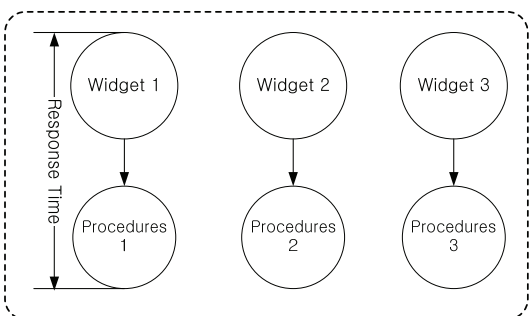

Fig. 2. Threads of GUI processing in an Application.

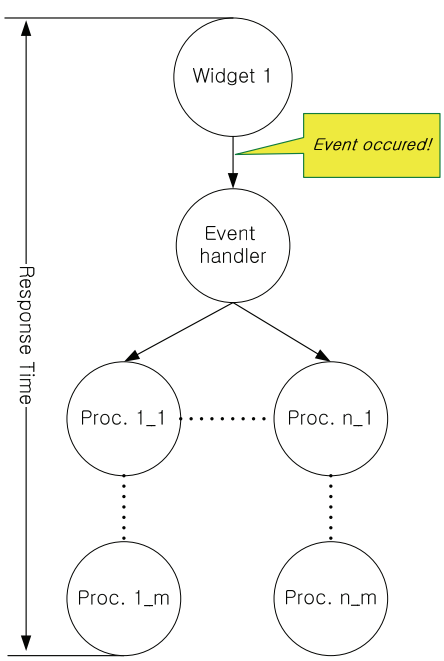

Fig. 3. Threads provoked by an event in a Widget.

하나의 GUI 컴포넌트 (본 논문에서는 위젯으로 표기함)에 대해 좀 더 자세하게 표현한 그림을 Fig. 3에 나타내었다. 해당 위젯에 사용자 이벤트가 발생한 경우 이벤트 핸들러가 동작하여, 해당 애플리케이션 개발자가 등록해 놓은 프로시져들을 구동 시킨다. 이 프로시져들은 한 가지일 수도 있고, Fig. 3과 같이 여러 개가 직렬성과 병렬성을 갖고 구동 될 수도 있다 (그림의 예에서는 n개의 병렬성과 m개의 직렬성을 갖음). 위젯의 화면 동작을 담당하는 쓰레드 (즉, Fig. 3에서 Widget 1)는 사용자 입력을 기다려서 처리 해야 하므로 상시 동작해야 한다. 그러므로 해당 애플리케이션(즉, 액티비티)가 전경(foreground)에 있을 경우, 사용 여부와 상관없이 해당 위젯 담당 쓰레드는 항상 프로세서를 점유한다. 반면에 이벤트 핸들러와 이에 예속된 프로시져들을 담당하는 쓰레드들은 해당 위젯에 이벤트가 발생한 경우 생성 또는 구동되어 프로세서에 할당되며, 처리 완료 후에는 프로세서 할당이 해지된다. 참고로, 본 논문에서는 big.LITTLE 구조를 기반으로 하므로, 프로세서는 big 코어 또는 LITTLE 코어를 의미한다.

big.LITTLE 구조 기반의 스마트 모바일 단말에서는, 빠른 사용자 응답성을 제공하기 위해 GUI 처리 쓰레드들을 big 코어에 우선적으로 할당한다[2]. Fig. 2의 경우 Widget1, Widget2, 그리고 Widget3 쓰레드는 big 코어에 우선적으로 할당된다. GUI 위젯 쓰레드들은 해당 액티비티가 전경(foreground)에 있을 경우 사용 여부와 상관 없이 프로세서를 계속 점유하므로, 해당 예의 경우 사용 여부와 상관없이 최소 3개의 big 코어가 계속 동작해야 하므로 에너지 효율성이 떨어진다. 또한 사용자가 입력 이벤트를 발생 시킨 후 응답시간은 위젯 처리 쓰레드와 이벤트 처리를 위해 등록된 프로시져들을 처리하는 쓰레드들이 모두 완료할 때 까지의 시간이며, 이 경우 위젯 처리 쓰레드보다는 이벤트 처리 프로시져들의 구동에 상대적으로 월등히 오랜 시간이 소요될 수 있다.

3. 제안 기법

앞장의 설명과 같이 기존의 big.LITTLE 할당방식에서는 에너지 비효율성과 응답시간의 지연문제가 발생하므로, 이를 개선하기 위해 GUI환경에서 사용자 경험을 활용한 저전력 비대칭 멀티코어 할당 기법을 제안한다. 제안 기법의 주요내용은 다음과 같다. 사용자 경험 정보를 활용하여 사용 빈도가 높지 않은 위젯들의 처리 쓰레드들은 LITTE 코어에 우선적으로 할당하면 에너지 소비를 줄일 수 있다. 반면, 사용빈도가 높은 위젯들의 이벤xm처 처리 쓰레드들을 big 코어에 우선적으로 할당하면, 응답시간을 단축함으로써 사용자 만족성을 높일 수 있다.

Fig. 4는 본 논문에서 제안하는 기법을 나타낸다. 제안 기법은 애플리케이션의 액티비티가 생성된 경우(1~7라인)와 위젯에 사용자 입력 이벤트가 발생한 경우(8~14라인)의 처리 과정으로 구성된다.

```
- widget[i] : i-the widget of the application activity
- num_widget: the number of widgets in the application
- num_Proc[i]: the number of event handling procedures of
widget[i]
- widget[i].usage: usage value of widget[i]
- widget[i].proc[k]: k-th event handling procedure of widget[i]
- LITTLE/big core first policy: assign the target thread to
LITTLE/big core firstly. If all of LITTLE/big cores are
unavailable, then assign it to any available core.
- USAGE_THRSHOLD : pre-defined minimum usage value for
a frequently used widget

On the activity of the application is Created,
1 FOR (i=0; i< num_widget; ++i)
2     IF (widget[i].usage <USAGE_THRSHOLD)
3         assign widget[i] to a core with the LITTLE core first
          policy
4     ELSE
5         assign widget[i] to a core with the ig core first policy
6     ENDIF
7 END

On a user event for widget[i] is occurred
8 IF (widget[i].usage >USAGE_THRESH)
9    FOR (k=0; k< num_Proc[i]; ++k)
10        assign widget[i].proc[k] to a core with the big core first
          policy
11     END
12 ELSE
13     FOR (k=0; k< num_Proc[i]; ++k)
14         assign widget[i].proc[k] to a core with the LITTLE core
           first policy
15     END
16 ENDIF
```

Fig. 4. Proposed Algorithm.

액티비티가 생성된 1)의 경우 액티비티에 등록된 위젯들을 검사하여 사용빈도가 낮은 경우 (즉, USAGE_THRSHOLD 보다 작은 경우) 해당 위젯의 구동 쓰레드를 LITTLE 코어 우선으로 할당하고 그렇지 않은 경우 big 코어 우선으로 할당한다.

특정 위젯의 입력이벤트가 발생된 2)의 경우 해당 위젯이 사용빈도가 높은 경우 그 위젯의 이벤트 핸들러에 등록된 프로시져들을 구동하는 쓰레드들을 big 코어 우선으로 할당하고, 그렇지 않은 경우 LITTLE 코어 우선으로 할당한다.

4. 성능평가

제안 기법의 성능을 검증하기 위해 시뮬레이션을 통해 성능평가를 하였다. 실제 환경에 대한 모사를 위하여 다음과 같은 가정을 하였다. 가정과 실험 파라메터들은 기존 관련 연구 [2,3]의 설정 값을 참조하였다.

- 시뮬레이션 대상 스마트 모바일 단말은 4개의 big 코어와 4개의 LITTLE 코어로 구성 된다.
- big코어는 1.8GHz으로 동작하며 초당 0.57μJ의 에너지를 소모한다. LITTLE 코어는 1.4GHz로 동작하며 초당 0.24μJ의 에너지를 소모한다.
- 각각의 GUI 위젯들은 각각 별도의 쓰레드에서 동작한다. GUI 위젯 쓰레드는 상시 동작 하므로 해당 앱이 foreground인 경우 지속적인 CPU 점유가 발생한다.
- 각GUI 위젯에 사용자 이벤트가 발생 할 경우 이벤트 핸들러과 등록된 프로시져들의 동작을 위한 쓰레드가 추가로 구동되는데, 본 논문에서는 위젯 별로 한 개의 쓰레드가 추가되서 담당한다.
- 위젯 이벤트 처리 프로시져들은 각 위젯별로 한 개의 쓰레드에서 담당한다. 또한, 이들의 작업량은 해당 위젯 쓰레드의 배수 배로 가정한다.
- 사용자 이벤트의 발생은 도착률이 λ인 포아송 프로세스 (Poisson process)를 따른다.
- 한번의 실험은 10,000초간 진행 되었고, 총 100번 반복하여 평균을 구했다.
- 측정하는 에너지 소비량은 시뮬레이션 시간(즉, 10,000초) 동안 소비된 총 에너지를 mJ 단위로 표현한다.

성능평가를 위하여 현재 안드로이드 플랫폼에서 주로 사용되고 있는 기존의 기법(Legacy)와 본 논문에서 제안하는 기법(Proposed)의 에너지 소비량과 대기 시간을 비교하였다. 각 기법에 대한 설명은 다음과 같다.

- *Legacy*: 위젯의 화면 출력과 동작을 담당하는 쓰레드는 big 코어 우선으로 할당된다. 위젯에 사용자 입력 이벤트 발생 시 미리 등록된 이벤트 핸들러와 프로시져들을 담당하는 쓰레드는 우선순위와 작업의 양에 따라 big/LITTLE 코어 우선이 결정된다. 단, 본 논문에서는 LITTLE 코어 우선으로 가정하였다.
- *Proposed*: 본 논문에서 제안하는 알고리즘 (Fig. 4)에 따라 big/LITTLE 코어에 할당 된다.

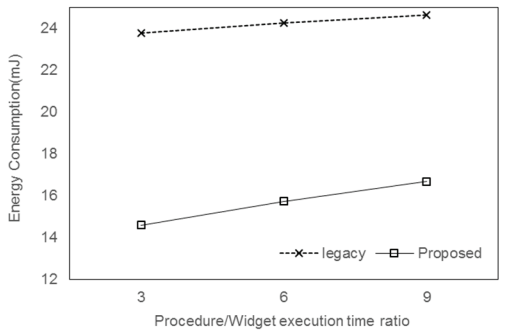

Fig. 5. Energy consumption by varying procedure/widget execution time ratio.

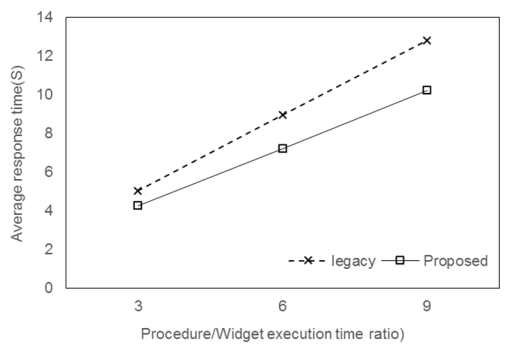

Fig. 6. Average response time by varying procedure/widget execution time ratio.

사용자가 위젯에 입력 이벤트를 발생 시켰을 때, 위젯 쓰레드에서 수행하는 시간과 위젯의 이벤트 처리를 위해 등록된 프로시져들을 수행하는 시간의 비율(즉, P/W Ratio)을 1:3 에서 1:9까지 변화시켜 가면서 측정한 에너지 소모량을 Fig. 5에서 나타내었고, 동일한 상황에서 측정한 평균 사용자 응답시간을 Fig. 6에서 나타내었다. λ는 0.1, 위젯의 개수는 4개이고 이중 한 개의 위젯이 전체 사용 빈도 중 50%를 차지하도록 설정하였다.

Fig. 5에서와 같이 제안 기법(*Proposed*)이 전반적으로 적은 에너지 소모량을 나타내었는데, 최대 40%의 에너지 절약 효과를 보였다. 단, P/W Ratio가 커 질수록 *Proposed* 와 *Legacy* 모두 에너지 소모량이 증가하였는데, *Legacy*의 증가 폭이 적은 이유는 구조상의 최대치(즉 big 코어로만 수행한경우)에 가까워 졌기 때문으로 추정된다.

Fig. 6에서와 같이 평균 응답시간도 *Proposed*가 전반적으로 더 작은 응답시간을 나타내었는데, *Legacy*보다 최대 20%의 단축효과를 보였다. P/W Ratio가 증가할수록 *Legacy*와 *Proposed*의 차이가 커짐을 볼 수 있는데, 이는 *Proposed* 에서 위젯 쓰레드 보다 이벤트 처리 프로시져 쓰레드를 big 코어에 할당하는 경향이 크기 때문이다.

5. 결 론

기존의 big.LITTLE 할당방식에서는 에너지 비효율성과 응답시간의 지연문제가 발생하므로, 이를 개선하기 위해 본 논문에서는 GUI환경에서 사용자 경험을 활용한 저전력 비대칭 멀티코어 할당 기법을 제안하였다. 실제 환경을 반영한 시뮬레이션을 통해 제안 기법이 에너지 효율면에서 기존 기법 대비 최대 40%를 향상시켰고, 응답시간 단축면에서도 최대 20%를 개선 시켰음을 보였다.

감사의 글

이 논문은 2018년도 남서울대학교 학술연구비 지원에 의해 연구되었음.

참고문헌

1. Pi-Cheng Hsiu, Po-Hsien Tseng, Wei-Ming Chen, Chin-Chiang Pan, and Tei-Wei Kuo, “User-centric scheduling and governing on mobile devices with big.little processors,” ACM Transactions on Embedded Computing Systems, vol. 15, no. 1, pp.17-22, 2016.
2. D. H. Bui, Y. Liu, H. Kim, I. Shin and F. Zhao, “Rethinking Energy-Performance Trade-Off in MObile Web Page Loading,” Mobicom 2015, Paris, France, pp.14-26, Sep. 2015.
3. DongHoon Kim, Young-Bae Ko, and Sung-Hwa Lim, “Power-efficient big.LITTLE core assignment scheme for real-time smartphone applications, In The 4th International Symposium on Mobile Internet Security (MobiSec), Oct. 2019.
4. C. Hwang, S. Pushp, C. Koh, & J. Yoon, Y. Liu, S. Choi, J. Song, “RAVEN: Perception-aware Optimization of Power Consumption for Mobile Games”, Mobicom17, Utah, USA, pp. 422-434, Oct. 2017.
5. L. Zhong and N.K. Jha, “Graphical User Interface Energy Characterization for Handheld Computers,” Proc. Int’l Conf. Compilers, Architecture, and Synthesis for Embedded Systems, pp. 232-242, Oct. 2003.
6. D. Li, S. Hao, J. Gui, W. G.J. Halford, “An Empirical Study of the Energy Consumption of Android Applications,” Proc. 2014 IEEE International Conference of Software Maintenance and Evolution,

Canada, Oct. 2014.
7. K. S. Vallerio, Lin Zhong and N. K. Jha, “Energy-efficient graphical user interface design,” in IEEE Transactions on Mobile Computing, vol. 5, no. 7, pp. 846-859, July 2006.
8. J. A. Paradiso and T. Starner, “Energy scavenging for mobile and wireless electronics,” IEEE Pervasive computing, vol. 1, pp. 18–27, 2005.
9. Android official site, developers.android.com.
10. “big.LITTLE Technology: The Future of Mobile,” white paper, ARM technology. Available: https://www.arm.com/files/pdf/big_LITTLE_Technology_the_Futue_of_Mobile.pdf

접수일: 2020년 2월 25일, 심사일: 2020년 3월 11일,
게재확정일: 2020년 3월 18일