Journal of Korea Multimedia Society Vol. 14, No. 3, March 2011(pp. 462-470)

에너지 소비 모니터링을 위한 재목적 인스트럭션-셋 시뮬레이터

고 광 만†

요 약

재목적성(retargetability)은 아키텍처 정보를 정형화된 형식으로 기술하여 컴파일러, 시뮬레이터와 같은 소프트웨어 개발 도구(SDK)를 생성하는데 이용된다. 시뮬레이터는 임베디드 프로세서의 설계를 하드웨어로 구현하기 전에 아키텍처의 다양한 성능 확인과 개선을 위해 소프트웨어적으로 검증할 수 있는 중요한 하드웨어 및 소프트웨어 개발 도구이다. 이러한 시뮬레이터는 시스템의 기능 검증, 성능 측정, 전력 · 에너지 소비 측정 결과 등을 하드웨어 설계 과정에서 중요하게 활용한다. 이 논문에서는 에너지 소비 측정이 가능한 시뮬레이터를 ADL로부터 생성하기 위해 첫째, 에너지 소비 측정 및 모니터링 요소를 ADL에 표현한다. 둘째, ADL 표현으로부터 에너지 측정 및 모니터링 시뮬레이션 라이브러리를 생성한 후 시뮬레이터인 RenenrgySim을 구축한다. 마지막으로, MiPS R4000에 대한 ADL을 표현을 작성하여 에너지 소비 측정 결과를 제시한다. 이러한 연구는 모바일 임베디드 소프트웨어 개발 분야에서 소프트웨어적인 실험을 통해 효과적인 아키텍처 개발과 신속한 SDK 생성에 활용될 수 있다.

Retargetable Instruction-Set Simulator for Energy Consumption Monitoring

Kwang-Man Ko†

ABSTRACT

Retargetability is typically achieved by providing target machine information, ADL, as input. The ADL are used to specify processor and memory architectures and generate software toolkit including compiler, simulator, etc. Simulator are critical components of the exploration and software design toolkit for the system designer. They can be used to perform diverse tasks such as verifying the functionality and/or timing behavior of the system, and generating quantitative measurements(e.g., power · energy consumption) which can be used to aid the design process. In this paper, we generate the energy consumption estimation simulator through ADL. For this goal, firstly, we describes the energy consumption estimation and monitoring informations on the ADL based on EXPRESSION. Secondly, we generate the energy estimation and monitoring simulation library and then constructs the simulator, RenergySim. Lastly, we represent the energy estimations results for MIPS R4000 ADL description. From this subjects, we contribute to the efficient architecture developments and prompt SDK generation through programmable experiments in the field of mobile software development.

Key words: Retargetability(재목적성), Energy-consumption Estimation(에너지-소비 측정), Simulator (시뮬레이터), Instruction-set Architecture(인스트럭션-셋 아키텍처)

※ 교신저자(Corresponding Author) : 고광만, 주소 : 강원도 원주시 우산동 660번지 상지대학교 컴퓨터정보공학부 (220-702), 전화 : 033)730-0486, FAX : 033)730-0480, E-mail : kkman@sangji.ac.kr
접수일 : 2010년 12월 15일, 수정일 : 2011년 2월 21일
완료일 : 2011년 3월 18일
† 정회원, 상지대학교 컴퓨터정보공학부
※ 이 논문은 2009년도 상지대학교 교내 연구비 지원에 의한 것임.

1. 서 론

특정 응용 분야를 위해 개발된 임베디드 프로세서의 진화 및 새로운 출현에 부응하여 이를 지원할 수 있는 SDK 개발에 관한 연구가 활성화되고 있다. 특정 임베디드 시스템에 적합한 신속하고 효율적인 SDK 개발을 위해 프로세서 및 메모리 구조, 인스트럭션 동작 등에 대한 정보를 정형화된 ADL로 표현한 후 이로부터 컴파일러, 시뮬레이터, 어셈블러, 디버거 등과 같은 SDK를 생성하는 재목적 기술이 적극적으로 응용되고 있다[1]. 시뮬레이터는 임베디드 프로세서의 설계를 하드웨어로 구현하기 전에 아키텍처의 다양한 성능 확인과 개선을 위해 소프트웨어적으로 검증할 수 있는 중요한 하드웨어 및 소프트웨어 개발 도구로서 아키텍처의 기능 검증, 성능 측정, 전력·에너지 소비 측정 결과 등을 하드웨어 설계 과정에서 중요하게 활용하고 있다[2]. 특히, 최근에는 임베디드 시스템에서 실행되는 어플리케이션에 대해 에너지 소비량을 측정하고 모니터링할 수 있는 환경 구축이 컴파일러 분야의 에너지 소비 최적화와 같은 연구와 더불어 필요성이 지속적으로 제기되고 있는 분야이며, 다양한 프로세서에 대한 재목적 시뮬레이터의 개발은 하드웨어와 소프트웨어 통합 설계와 탐색을 강조하는 임베디드 SoC 분야에서 중요한 연구 분야이다[3].

전력·에너지의 최적화 소비 관리는 전력 공급이 제한적인 모바일 장치를 위한 아키텍처 설계에서 중요한 문제로 대두되고 있다. 따라서 고성능, 저전력 아키텍처의 설계를 위해 소프트웨어적인 실험, 전력·에너지 소비 최적화에 대한 다양한 기술 제안과 평가 방법이 필요하다. 실제로 아키텍처 수준의 전력 측정 도구는 초기 설계 단계에서 에너지 소비를 빠른 속도로 측정하려고 하는 최근의 설계 방법이 복잡해지면서 중요성이 증가되고 있다[4,5]. 하드웨어에서 실행되는 소프트웨어의 에너지 소비는 운영체제, 실행시간 환경, 컴파일러, 어플리케이션 등이 복잡한 상호 작용을 통해 이루어진다. 최근 까지는 에너지-지향적인 고수준 및 저수준의 컴파일러 최적화 동작을 통해 어플리케이션의 에너지 소비를 줄이기 위한 노력이 매우 효과적으로 진행되어 왔다[6].

이 논문에서는 어플리케이션의 에너지 소비 측정 및 모니터링이 가능한 시뮬레이터인 RenergySim을 ADL로부터 재목적 기술을 응용하여 생성한다. 이를 위해 첫째, 에너지 소비 측정 및 모니터링 요소를 ADL에 표현한다. 둘째, ADL 표현 정보를 기반으로 에너지 측정 및 모니터링이 가능한 시뮬레이터를 생성한다. 마지막으로, MiPS R4000에 대한 ADL 표현을 작성하여 실제로 에너지 측정 및 모니터링이 가능한 시뮬레이터를 생성한 후 벤치마킹 결과를 제시한다. 이러한 연구는 모바일 임베디드 시스템 환경에서 실제 하드웨어 개발전에 소프트웨어적인 실험을 통해 효율적인 아키텍처 개발에 응용할 수 있으며, 에너지 소비 효율적인 SDK 개발에 활용한다.

이 논문의 구성은 제2장에서 시뮬레이터 개발 방법에 관한 연구 동향과 어플리케이션의 전력·에너지 소비 측정 기술에 관련된 선행 연구 내용을 소개한다. 제3장에서는 재목적 기술을 응용하여 에너지 소비 모니터링과 측정이 가능한 시뮬레이터를 실제 ADL로부터 생성하는 과정과 벤치마킹 결과를 설명한다. 마지막으로 제4장에서는 결론과 향후 연구 내용에 대해 기술한다.

2. 관련 연구

2.1 시뮬레이터 개발 방법

시뮬레이터의 개발 방법은 추상화 수준에 따라 고수준 추상화는 프로세서의 기능적인 시뮬레이션이 인스트럭션-셋의 모델링을 통해 수행되며 속도가 빠른 장점을 가지고 있다. 저수준 추상화에서는 보다 자세한 타이밍 정보를 생성하는 Cycle-accurate 시뮬레이션 모델링을 통해 수행된다. 시뮬레이터의 실행 엔진 모델은 시뮬레이션의 속도와 융통성에 중요한 영향을 주며 인터프리티브 방식[7], 컴파일 방식[8], 혼용 시뮬레이터[9]로 구분된다.

인터프리티브 시뮬레이션은 아키텍처의 인스트럭션-셋에 대해 인터프리터 처리 방식의 모델을 채택하고 있으며 메모리에 프로세서의 상태를 저장하고 프로그램 메모리에서 인스트럭션 페치, 디코드, 실행을 순차적으로 진행한다. 이 모델의 장점은 구현이 쉬우며 융통성이 뛰어나고 가변적인 프로세서의 상태 정보를 모을 수 있다는 점이다. 하지만 인스트럭션의 페칭, 디코딩, 디스패칭에 따른 오버헤드로 인해 성능 저하의 단점이 제기된다. 현재 상업용로 발표되거나 사용되는 SIMPRESS[10]에서는 인터프

리티브 방식을 채택하고 있다. 컴파일 시뮬레이션 방식은 인스트럭션을 시뮬레이트되는 기계 상태를 다루는 인스트럭션으로 변환하므로서 실행 시간에 발생되는 오버헤드를 감소시키는 특징을 가지고 있다. 인스트럭션의 변환은 컴파일 시간에 발생되어 페치, 디코드, 디스페칭의 오버헤드를 줄이거나 실행 시간에 발생되어 코드가 반복 실행으로 발생될 수 있는 오버헤드를 줄일 수 있다. 혼용 시뮬레이션 방식은 인터프리티브 시뮬레이션 모델의 장점과 컴파일 방식의 시뮬레이션 모델의 장점을 적절히 혼용하여 사용한다[11]. 두 방식의 장점을 적절히 혼용한 방식으로 JIT-CCS[12], IS-CS[13]가 최근에 발표되었다. JIT-CCS 방식에서는 인스트럭션이 실행 직전인 Just-In-Time에 컴파일되며 추출된 정보는 해당 프로그램 코드의 반복 실행시 재사용을 위해 시뮬레이터 캐쉬에 저장된다. 캐쉬에 저장되어 있던 프로그램 코드에 변경을 인식하게 되면 다시 컴파일하도록 한다. IS-CS 방식에서는 시간 소비가 많은 디코딩을 컴파일 시간에 수행하여 실행 시간 감소 효과를 얻을 수 있다. 실행 시간에 인스트럭션이 변경되는 경우는 실행 직전에 다시 컴파일한다.

시뮬레이션의 속도가 빠르고 재목적성을 반영한 시뮬레이터인 [8]에서는 시뮬레이터가 탑재되는 시스템의 자원을 적극적으로 사용하여 전통적인 정적 컴파일 시뮬레이션 기법의 성능을 개선하였다. 시스템 자원의 사용은 전통적인 접근 방식인 코드 생성 인터페이스로서 C 언어를 사용하기 보다는 ISA를 위한 저수준의 코드 생성 인터페이스를 특별히 정의하여 사용하였다. ADL에 기반을 둔 빠른 재목적 시뮬레이터의 개발에 관한 연구는 LISA[14], EXPRESSION[15]에서 많은 인용과 주목을 받았다. 특히, LISA ADL은 컴파일러, 어셈블러, 링커, 시뮬레이터, 프로파일러로 구성된 소프트웨어 개발 슈트의 자동 생성을 지원하고 짧은 시간에 최적화된 RTL 기술을 통해 아키텍처 설계에 대한 탐색을 지원한다.

2.2 전력 · 에너지 측정 시뮬레이터

에너지 측정 모델은 전력 측정 시점을 기준으로 정적 관리 기술(SPM)과 동적 관리 기술(DPM)로 구분되며, 전력 측정 수준에 따라 사이클 또는 인스트럭션 수준에서 CPU의 전력 소비를 측정하는 저수준 방식과 전체 시스템의 모든 구성 요소에 대해 전력 소비를 측정하는 시스템-수준의 고수준 방식으로 구분되어 연구되고 있다. 또한 CPU-수준에서의 전력 소비는 세부적으로 RTL-수준(Cycle-수준)과 인스트럭션-수준으로 구분되어 분석되고 있다[16]. RTL-수준에서 전력 소비를 분석하는 시뮬레이터는 대표적으로 Simplescalar[17], Simplepower[18], Wattch [19] 등이 집중적으로 주목을 받고 있으며, 현재까지도 많이 인용되고 있다. SimplePower는 transition-sensitive 에너지 모델에 기반을 두고 개발되었으며 실행-드리븐, Cycle-accurate RT 수준의 에너지 시뮬레이터로서 Simplescalar 인스트럭션 셋의 서브셋을 시뮬레이트하고 시뮬레이션 과정에서 C 코드를 Simplepower에서 실행 가능한 형태로 변환하기 위해 Simplescalar의 컴파일러를 이용한다. Wattch는 아키텍처 수준에서 마이크로-프로세서의 전력 소비에 대한 분석과 최적화를 위한 프레임워크로서 CPU의 전력 소비를 측정하는 시뮬레이터이다. 특히, 전력 효율성에 관한 아키텍처와 컴파일 방식의 접근 연구에 대해 전력 소비 측정 모델링 인프라로서 매우 유용하고 중요하게 활용되고 있다. Sim-Panalyzer [20]는 Simplescalar의 "sim-outorder"에 기반을 전력 시뮬레이션 도구로서 최초에는 ARM ISA에 대한 전력 소비 측정과 분석을 목표로 하고 있으며 복잡하고 정확한 전력 측정을 위한 모델을 위해 별도의 라이브러리를 지원하는 특징을 가지고 있다.

3. 재목적 에너지 소비 모니터링 시뮬레이터

3.1 RenergySim 생성 프레임워크

이 논문에서는 EXPRESSION 연구를 기반으로 에너지 소비 측정과 모니터링이 가능한 재목적 시뮬레이터인 RenergySim을 그림 1과 같은 과정을 통해 생성한다. 이를 위해 첫째, EXPRESSION ADL에 에너지 소비 모니터링을 위한 정보를 표현하기 위해 ADL을 확장하였다[21]. 둘째, 확장된 ADL로부터 재목적 컴파일러 생성기인 EXPRESS를 이용하여 재목적 컴파일러를 생성한 후 어플리케이션에 대한 바이너리를 생성하였다. 또한 확장된 ADL로부터 바이너리를 시뮬레이터에 적합한 C++ 코드로 변환하는 인스트럭션 디코더를 자동 생성하였다. 셋째, 시뮬레이터 생성기인 SIMPRESS[10]를 확장하여 시뮬레이터 구성에 필요한 아키텍처 구조 정보 라이브러리,

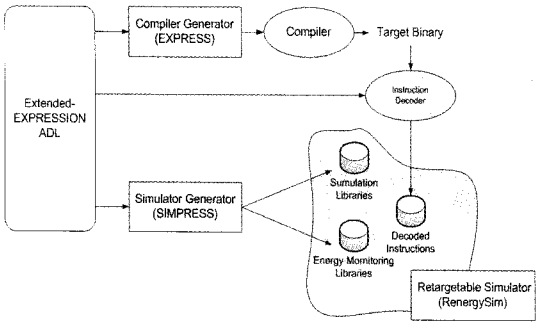

그림 1. ADL을 통한 에너지 측정 시뮬레이터(RenergySim) 생성

시뮬레이션 엔진 구성에 필요한 라이브러리, 에너지 소비 모니터링을 위한 라이브러리를 C++ 형태로 생성하였다. 마지막으로 생성된 인스트럭션 디코딩 정보, 시뮬레이션 라이브러리, 에너지 측정 라이브러리를 링킹한 후 GCC 컴파일러를 이용하여 Renergy-Sim을 생성하였다. 시뮬레이션 엔진은 IS-CS 방식[13]을 기반으로 인터프리티브 시뮬레이터의 장점인 융통성을 보장하면서 단점인 시뮬레이션 속도를 개선하기 위해 인스트럭션 디코딩을 컴파일 시간에 수행하여 변경된 인스트럭션에 대해서만 실행시간에 수행하는 특징을 가지고 있다.

시뮬레이터 구성에 필요한 정보를 생성하는 SIMPRESS는 SimulatorGenerator 클래스의 generatorSimulator() 메소드를 통해 Base Library, Target Component Library, Connectivity Library, Simulator Functionality Library와 확장된 ADL에 기술된 정보를 활용하여 에너지 소비 모니터링을 위한 Energy Monitoring 라이브러리를 그림 2와 같은 과정을 통해 생성한다. Unit, Latch, Storage, Port를 초기화하는 buildSystem() 메소드와 Cache · Sram · Dram을 초기화 하는 initialize() 메소드, Cache · Sram · Dram에 대해 read(), write() 동작이 발생했을 때 에너지 소비량을 측정하는 메소드로 구성된다. initialize() 메소드는 메모리 모듈의 정보가 저장되어 있는 mem.config 파일을 참조를 통해 메모리 모듈과 파라미터 정보(MEM_MODULES), 모듈간 연결 정보(CONNECTIVITY), 주소 맵핑 정보(MEMORY MAP)가 지원된다. 메모리 모듈과 파라미터 정보에는 모듈 번호, 모듈 타입, 모듈 파라미터로 구성되며, 모듈간 연결 정보에는 메모리 모듈 번호를 두 개씩 표시한다. 마지막으로 주소 맵핑 정보는 메모리 모듈이 사용하는 시작 주소 값과 끝 주소 값으로 구성되어 있다.

3.2 에너지 소비 모니터링 정보 및 라이브러리 생성

에너지 소비 모니터링을 위한 라이브러리는 ADL 표현으로부터 생성하는 부분과 독립적인 라이브러리를 구현하여 지원하는 부분으로 구성된다. 에너지 소비를 모니터링 하기 위해 ADL의 "Componenet"

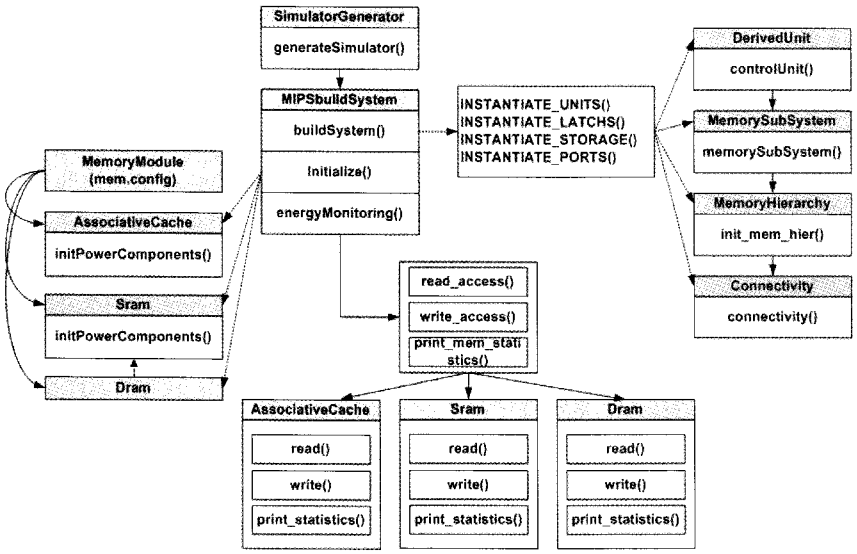

그림 2. Energy Monitoring 라이브러리를 구성하는 핵심 클래스 관계

섹션에 아키텍처 구성 정보가 우선적으로 활용된다. "Componenet" 섹션에서는 파이프라인 유닛, 함수 유닛, 포트, 기억장소, 버스 등과 같은 RT-수준의 컴포넌트와 속성 정보를 SUBCOMPONENTS, LATCHES, PORTS, CONNECTIONS, OPCODES, TIMING, CAPACITY를 이용하여 그림 3과 같이 표현한다.

```
(FetchUnit FETCH
  (CAPACITY 1)
  (INSTR_IN 4)
  (INSTR_OUT 4)
  (TIMING (all 1))
  (OPCODES all)
  (LATCHES (OUT FetDecLatch)
    (OTHER pcLatch))
 )
...
(DecodeUnit DECODE
  (CAPACITY 12)
  (INSTR_IN 4)
  (INSTR_OUT 2)
  (TIMING (all 1))
  (OPCODES all)
  (LATCHES (OUT DecAlu1ReadLatch)
    (OUT DecAlu2ReadLatch)
  (LATCHES (IN FetDecLatch))
 )
```

그림 3. Component 섹션에서 아키텍처 구성 정보 표현

메모리 서브시스템 섹션에서는 레지스터 파일, SRAM, DRAM, CHCHE 등과 같은 다양한 메모리 구성 요소의 자료형과 속성을 표현한다. 실제로 이 연구에서 그림 4와 같이 설계한 메모리 계층 구조에 대한 ADL의 표현은 그림 5와 같다.

```
(STORAGE_SECTION
 (RFA
  (TYPE VirtualRegFile)          //레지스터
  (WIDTH 32) (SIZE 32) (MNEMONIC "R")
 )
 (RFB
  (TYPE VirtualRegFile)
  (WIDTH 64) (SIZE 32) (MNEMONIC "f")
 )
 (L1_Assoc_Dcache
  (TYPE DCACHE)
  (SIZE 64) (LINESIZE 2) (ASSOCIATIVITY 4)
  (ACCESS_TIMES 1) (ADDRESS_RANGE (0 9995904))
 )
 (L2_Assoc_Dcache
  (TYPE DCACHE)
  (SIZE 64)  (LINESIZE 2)  ASSOCIATIVITY 8)
  (NUM_LINES 64)  (ACCESS_TIMES 5)
  (ADDRESS_RANGE (0 9995904))
 )
 (L1_Sram
  (TYPE SRAM)  (ACCESS_TIMES 1)
  (ADDRESS_RANGE (9995905 9999999))
 )
 (MainMem_Dram
  (TYPE DRAM) (ACCESS_TIMES 50)
  (ADDRESS_RANGE (0 9995904))
 )
```

그림 5. 메모리 계층 구조 설계(그림 4)에 대한 ADL 표현

프로세서는 두 개의 레지스터(RFA(32), RFB(64)를 가지고 있으며 MainMem_Dram은 512 비트 크기로서 두 개의 Associative Cache에 의해 접근된다.

ADL에 표현된 아키텍처 및 메모리 정보로부터 어플리케이션의 에너지 소비량을 측정하기 위해 우선적으로 Cache, Ram에 대한 정보를 초기화한 후 Cache, Ram에 대해 읽기(read())와 쓰기(write()) 동작의 횟수를 측정하기 위해 그림 6과 같은 8가지 동작에 대해 정의하였다.

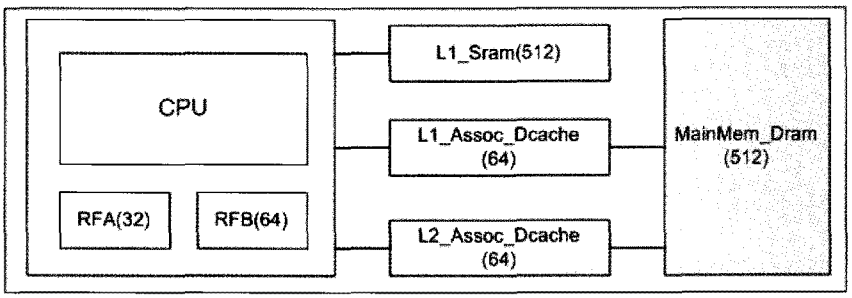

그림 4. 메모리 계층 구조 설계

① Cache에서 Read한다는 것은 Cache에 접근.
② Cache에서 Read가 성공했다는 것은 Cache에서 읽기 성공.
③ Cache에서 Read가 실패하였다는 것은 Memory의 접근이 필요.
④ Cache에서 Write한다는 것은 Cache에 접근 성공.
⑤ Cache에서 Write가 성공했다는 것은 Cache에 저장.
⑥ Cache에서 Write가 실패하였다는 것은 다른 공간에 데이터 이동이 필요.
⑦ Ram에서 Read했다는 것은 Ram에서 읽기 성공.
⑧ Ram에서 Write했다는 것은 Ram에 저장 성공.

그림 6. Cache, Ram에 대한 8가지 동작 정의

이러한 정보를 저장하기 위해 MemoryModule을 상속받은 AssociativeCache, SRAM, DRAM 클래스를 초기화 시켜준다. MemoryModule에는 Cache나 Ram이 read/write를 했을 때의 횟수를 저장해 두는 read_hits, read_misses, write_hits, write_misses와 Cache를 접근한 횟수를 저장해 두는 cache_accesses를 정의하고 있다. 그 외에 read/write 할 때 발생하는 지연시간을 저장하기 위해 _latency를 정의하고 있으며, 에너지 측정에 필요한 요소들인 _decoder_power, _wordline_power, _bitline_power, _senseamp_power, _tagarray_power가 정의되어 있고, 마지막으로 read/write의 횟수를 얻기 위한 함수들이 정의되어 있다. 이러한 MemoryModule 클래스를 상속받은 AssociativeCache, SRAM, DRAM 클래스에서는 실제적인 에너지 측정을 위해 initPowerComponents() 메소드를 이용하여 MemoryModule 클래스에 선언된 변수들을 초기화하고 함수들을 재정의 한다. 실질적인 에너지 소비 측정은 SimMain 클래스에서 singleStep() 메소드가 사이클 횟수만큼 실행하면서 측정한다. MemorySubsystem 클래스의 doStep() 메소드에서 Cache에 대한 read/write 동작과 Ram에 대한 read/write 동작을 결정한다. Cache와 Ram에 대한 모든 read/write 동작이 끝나게 되면 실제 사용된 에너지 소비량은 print_statistics() 메소드를 통해서 계산 및 출력한다.

```
//AssocaitiveCache 클래스의 print_statistics 메소드,
에너지 소비량 측정
void print_statistics(FILE *statPtr) {
  float rhit_ratio, whit_ratio, hit_ratio;
  if (cache_accesses == 0) {
     rhit_ratio=0.0;  whit_ratio=0.0;  hit_ratio=0.0;
  }
  else {
     hit_ratio=float(read_hits +
        write_hits)/float(cache_accesses);
     if ((read_hits + read_misses) == 0)
        rhit_ratio = 0.0;
     else  rhit_ratio = float(read_hits)/float(read_
        hits + read_misses);
     if ((write_hits + write_misses) == 0) whit_
        ratio = 0.0;
     else whit_ratio =
float(write_hits)/float(write_hits + write_misses);
  }
...
}
```

그림 7. Associative_Dcache에 대한 에너지 소비량 측정

실제 에너지 소비량을 계산 및 출력해 주는 print_statistics() 메소드의 구현은 그림 8과 같이 Cache 접근에 대한 에너지 소비량은 cache_accesses 횟수가 결정적인 요소이며 Sram은 read_hits와 write_hits 횟수가 에너지 소비량을 결정한다. Dram은 accessPower 값을 구할 때 read_hist와 write_hits 횟수를 적용하여 에너지 소비량을 측정한다.

3.3 실험 및 결과

이 논문에서는 최초에 의도한대로 EXPRESSION ADL에 에너지 소비 정보를 표현한 후 이로부터 기존 성능 측정과 더불어 에너지 소비 모니터링과 측정이 가능한 RenergySim 시뮬레이터를 생성한 후 EXPRESSION에서 컴파일러와 시뮬레이터를 검증에 적용되었던 어플리케이션(LL1.c~LL24.c)에 대한 에너지 소비량을 측정하였다. 실험 환경은 Window

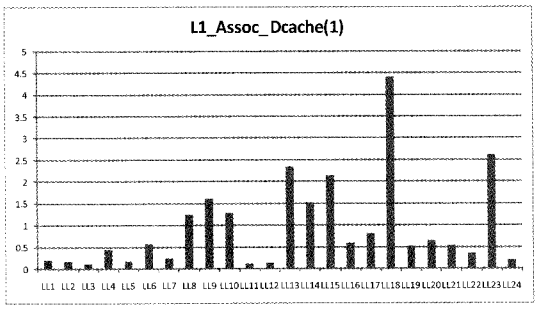

그림 8. L1_Assoc_Dcache(1) 접근에 대한 에너지 소비량 (uJ)

7, VC++ 2010, Intel Core 프로세서(3.60 GHz, 4.00G RAM) 환경에서 측정하였다. ADL 표현은 MIPS R4000 아키텍처에 대해 그림 4에서 설계한 메모리 구조를 표현하여 그림 8, 9와 같이 Cache 접근에 대한 에너지 소비량을 측정하였다.

Main_Mem_Dram 접근에 대한 에너지 소비량은 그림 10과 같다.

어플리케이션(LL1.c~LL24.c)에 대한 에너지 소비량을 측정은 기존 연구 방법과 차별적으로 ADL에 에너지 소비량을 측정할 수 있는 정보를 기술하여 이로 부터 생성된 시뮬레이터가 에너지 소비량을 측정하고 결과를 제시하였다. 이를 통해, 새로운 아키텍처의 설계 또는 변경에 대해 단지 ADL 표현의 변경만으로 에너지 소비 측정이 가능한 시뮬레이터를 생성한다. 이러한 연구 결과는 모바일 임베디드 시스템 분야에 적합한 아키텍처 설계와 소프트웨어를 개발하는데 활용할 수 있다.

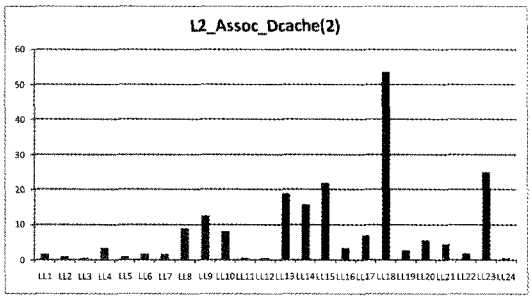

그림 9. L1_Assoc_Dcache(2) 접근에 대한 에너지 소비량 (uJ)

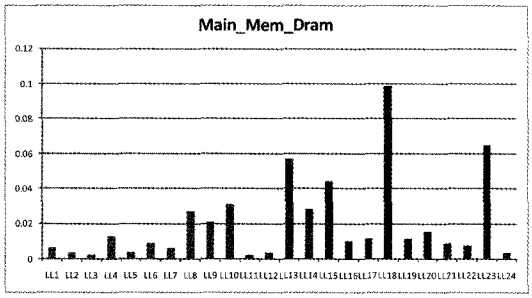

그림 10. Main_Mem_Dram 접근에 대한 에너지 소비량 (uJ)

4. 결 론

시뮬레이터는 임베디드 프로세서의 설계를 하드웨어로 구현하기 전에 아키텍처의 다양한 성능 확인과 개선을 위해 소프트웨어적으로 검증할 수 있는 중요한 하드웨어 및 소프트웨어 개발 도구로서, 아키텍처의 기능 검증, 성능 측정, 전력·에너지 소비 측정 결과 등을 하드웨어 설계 과정에서 중요하게 활용하고 있다. 최근에는 임베디드 시스템에서 실행되는 어플리케이션에 대해 에너지 소비량을 측정하고 모니터링할 수 있는 환경 구축의 필요성이 지속적으로 제기되고 있으며, 다양한 프로세서에 대한 시뮬레이터를 개발하기 위해 재목적 기술의 적용은 개발 비용과 시간을 줄일 수 있는 방안으로 제시되고 있다.

이 논문에서는 어플리케이션의 에너지 소비 측정 및 모니터링이 가능한 시뮬레이터인 RenergySim을 ADL로부터 재목적 기술을 적용하여 생성하였다. 이 연구를 통해 첫째, 에너지 소비 측정 및 모니터링 요소을 위해 아키텍처 구조 정보와 메모리 구조를 ADL에 표현하였다. 둘째, ADL 표현 정보로부터 에너지 측정 및 모니터링에 필요한 라이브러리를 생성한 후 실제 시뮬레이터 개발에 활용하였다. 마지막으로, RenergySim의 동작과 에너지 소비 측정을 검증하기 위해 MiPS R4000에 대한 ADL을 표현한 후 에너지 소비량 측정 결과를 제시하였다. 이러한 연구는 모바일 임베디드 시스템 환경에서 실제 하드웨어 개발전에 소프트웨어적인 실험을 통해 효율적인 아키텍처 개발에 응용할 수 있으며 신속하고 에너지 소비 효율적인 SDK 개발에 활용할 수 있다. 현재보다 세분화된 에너지 소비량 측정 정보를 ADL에 표현하여 아키텍처의 다양한 부분에서 에너지 소비량 측정이 가능하도록 보완 연구중이며 시뮬레이션 과정에서 에너지 소비 요소를 추적할 수 있는 디버거를 개발할 예정이다.

참 고 문 헌

[1] Prabhat Mishra and Nikil Dutt, *Processor Description Languages*, Morgan Kaufmann, 2008.

[2] Rainer Leupers, "Code Generation for Embedded Processor," ISSS;00: 13th International System Synthesis Symposium, pp. 173-178, 2000.

[3] A. Hoffman and H. Meyr, R. Leupers, Architecture Exploration for Embedded Processors

with LISA, Kluwer Academic Publishers (ISBN: 1-4020-7338-0), Dec. 2002.

[4] Uli Kremer, "Compilers for Power and Energy Management," PLDI'03: ACM SIGPLAN 2003 conference on Programming Language Design and Implementation Tutorial, 2003.

[5] Uli Kremer, "Low Power/Power Compiler Optimizations," in Low-Power Electronics Design(Editor: Christian Piguet), CRC Press, 2005.

[6] A. Parikh, Soontae Kim, M. Kandemir, N. Vijaykrishnan, and M. J. Irwin, "Instruction Scheduling for Low Power," *Journal of VLSI Signal Processing*, Vol.37, pp. 129-149, 2004.

[7] Eric C. Schnarr, Mark D. Hill, and James R. Larus, "Facile: A Language and Compiler for High-Performance Processor Simulators," PLDI'99: Proceedings of the ACM SIGPLAN 1999 conference on Programming Language Design and Implementation, pp. 1-11, 1999.

[8] Mehrdad Reshadi, Nikhil Bansal, Prabhat Mishra, and Nikill Dutt, "An Efficient Retargetable Framework for Instruction-Set Simulation," CODES+ISSS'03: Proceedings of the International Symposium on Hardware/Software Codesign and System Synthesis, pp. 13-18, 2003.

[9] R. Leupers, J. Elste, and B. Landwehr, "Generation of Interpretive and Compiled Instruction Set Simulators," ASP-DAC'99: Proceeding of the Asia South Pacific Design Automation Conference 1999, 1999.

[10] Alex Nicolau, et al., "V-SAT: A Visual Specification and Analysis Tool for System-on-Chip Exploration," *Journal of System Architecture*, Vol.47, pp. 263~275, 2001.

[11] Jianwen Zhu and Daniel D. Gasski, "A Retargetable, Ultra-fast Instruction Set Simulator," DATE'99: Proceedings of the Design Automation and Test conference in Europe, p. 1999.

[12] A. Nohl, et al., "A Universal Technique for Fast and Flexible Instruction-set Architecture Simulation," In Proc. of DAC, 2002.

[13] M. Reshadi, et al., "Instruction Set Compiled Simulation: A Technique for Fast and Flexible Instruction-set Simulation," In Proc. of DAC, 2003.

[14] Anupam Chattopadhyay, Heinrich Meyr, and Rainer Leupers, "LISA: A Uniform for Embedded Processor Modeling, Implementation, and Software Toolsuite Generation," Processor Description Languages: Chap. 5, Morgan Kauffman, 2008.

[15] Prabhat Mishra, Aviral Shrivastava, and NiKill Dutt, "Architecture Description Language-driven Software Toolkit Generation for Architectural Exploration of Programmable SOCs," *ACM Transactions on Design Automation of Electronics Systems*, Vol.11, No.2, pp. 626-658, 2006.

[16] Wissam Chedid and Chansu Yu, "Survey on Power Management Techniques for Energy Efficient Computer Systems," Technical Report: CSU-ECE-TR-02-01, Cleveland State University, 2002.

[17] SimpleScalar: MASE. http://www.simplescalar.com

[18] W. Ye, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, "The Design and Use of SimplePower: A Cycle Accurate Energy Estimation Tool," DAC'00: Proceedings of the 37th Design Automation Conference, pp. 340-345, 2000.

[19] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations," In Proc. of International Symposium on Computer Architecture, pp. 83-94, 2000.

[20] Sim-Panalyzer(ver 2.0.3) reference Manual. http://www.eecs.umich.edu/~panalyzer/.

[21] Kwangman Ko, "Design and Verification of the Class-based Architecture Description Language," Journal of Korea Multimedia Society, Vol.13, No.7, July 2010.

고 광 만

1991년 2월 원광대학교 컴퓨터공학과(공학사)
1993년 2월 동국대학교 컴퓨터공학과(공학석사)
1998년 2월 동국대학교 컴퓨터공학과(공학박사)
1998년 3월~2001년 8월 광주여자대학교 컴퓨터과학과 전임강사
방문연구 : QUT(2003, 호주), UQAM(2008, 캐나다), UC Irvine(2010, 미국)
2001년 9월~현재 상지대학교 컴퓨터정보공학부 교수
관심분야 : 프로그래밍언어론 및 컴파일러