사이클 정확도의 재목적화 가능한 마이크로아키텍쳐 시뮬레이션 프레임워크에 관한 연구

양훈모*, 이문기**
㈜코아로직*, 연세대학교**

A study on the Cycle-Accurate Retargetable Micro-Architecture Simulation Framework

Hoonmo Yang*, Moon-Key Lee**
*Corelogic Corp., **Yonsei Univ.
E-mail : *hmyang@corelogic.co.kr, **mklee@yonsei.ac.kr

Abstract

This paper presents CARMA (Cycle-Accurate Retargetable Micro-Architecture) as efficient framework for SoC-centric pipelined instruction-set architectures. It is based on ADL (Architecture Description Language) and provides more concise and manifest semantics to describe behavior of instruction set by mixing efficiency of instruction-set simulators and flexibility of RTL simulators. It exploits new timing model method based on process scheduling so it can support general timing model with cycle accuracy for large-scaled architectures usually used in SoC multimedia chip-set. According to experiments, the proposed framework was shown to be 5.5 times faster than HDL and 2.5 times faster than System-C in simulation speed so it is applicable for complex instruction-set pipelined architectures.

I. 서론

SoC 기술이 보편화되면서 멀티미디어 칩 셋은 점점 복잡해지면서 대규모화 되고 있는 반면 시장의 주기는 더욱 짧아지는 경향을 보이므로 재사용성 및 설계 편이성 증대가 주요 설계 이슈가 되고 있다. 따라서 설계 파라다임이 점차적으로 레지스터 전송 수준에서 명령어 군 수준으로 옮겨가고 있으며 SoC 를 구성하는 IP 는 기존의 무작위 회로에서 응용특화명령어군 프로세서 (application specific instruction set processor 이하 ASIP)로 대체되고 있으며 따라서 명령어군 수준 시뮬레이터가 더욱 중요시되고 있다. 그러나 기존의 명령어 수준 시뮬레이터는 응용 소프트웨어의 동적인 특성을 분석하는데 주로 사용하였기 때문에 하드웨어적인 특성을 정확하게 반영하지 못하고 있다. 반면 대부분의 ASIP 는 파이프라인 구조에 인터럽트 등을 지원하는 복잡한 타이밍 특성을 보이며 다양한 스펙에 적절하게 대응할 수 있는 재목적성이 크게 요구된다. 본 논문은 이를 해결하기 위해 아키텍처 기술 언어에 기반한 새로운 시뮬레이션 프레임 워크인 CARMA 를 설계 제안한다. 기존 ADL 시뮬레이터는 리저베이션 테이블(reservation table)에 기반한 제한된 타이밍 모델을 사용함으로써 일반적인 파이프라인 구조를 완벽히 기술하지 못하는 반면 제안한 구조는 프로세스 스케줄링에 기반하므로 임의의 파이프라인 구조에 대해 사이클 정확도를 가지고 완벽하게 타이밍 모델을 기술할 수 있다. 또한 이와 별도로 이벤트 구동 방식을 지원하여 인터럽트와 같은 파이프라인에 비동기적인 타이밍 모델도 지원한다. 따라서 레지스터 수준 모델의 정확성과 명령어 군 시뮬레이터의 효율성을 효과적으로 절충할 수 있다.

본 논문에서는 2 장에서 제안한 ADL 의 스펙 및 문법, 3 장에서 타이밍 모델 및 파이프라인 기술 방식,

4 장에서 가상 머신의 구조, 5 장에서 본 프레임워크의 성능에 대해 다루고 6 장에서 결론을 도출한다.

II. 아키텍처 기술 언어

제안한 ADL 은 그림 1 과 같은 계층 구조를 갖는다. 이중 prototype 은 최상위 수준 컴포넌트로 인스턴스화가 가능하며 포트를 통하여 다른 prototype 과 통신한다. 각각의 prototype 은 독립적인 파이프라인 구조를 가지며 이러한 파이프라인 구조의 파이프라인 스테이지는. stage 컴포넌트로 기술된다. 본 ADL 은 C 와 유사한 상위 수준 언어로서 명령어군 아키텍처를 톱다운(top-down) 방식으로 기술하며 타이밍 기술에 편리한 스케줄링 명령어 구문을 지원한다.

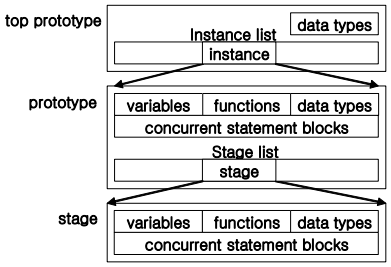

그림 1. 제안한 ADL 의 계층 구조

III. 타이밍 모델

본 프레임워크는 프로세스에 기반한 스케줄러를 이용하여 타이밍 모델을 구축한다. 하나의 프로세서는 그림 2-A 처럼 파이프라인 흐름 중 하나의 쓰레드를 대표한다. 이러한 프로세서는 5 개의 기본 명령어 구문에 의해 제어된다. 표 1 은 이를 정리한 것이다.

표 1. 프로세스 제어 구문

명칭	동작
SPLIT	현재 프로세스로부터 자식 프로세스 생성
STEP	현재 프로세스를 사이클 단위로 지연
ENTER	현재 프로세스를 특정 파이프라인 스테이지에 진입
WAIT	해당 이벤트가 발생할 때까지 현재 프로세스를 대기
EXIT	현재 프로세스 종료

개개의 프로세스는 SPLIT 에 의해 생성되고 파이프라인 스테이지를 거치며 최종적으로 EXIT 에 의해 종료된다. 명령어 페치 유닛에 대한 포괄적인 모델 기술을 위해 본 프레임워크는 SPLIT 문을 이용하여 명령어 페치를 명시적으로 기술한다. 예제 1 은 이러한 일례를 도시하고 있다. 또한 SPLIT 문은 다중 로드/스토어 및 다중 사이클 명령어에 대해 내부 루프 동작을 기술하는데 사용한다. 그림 2-B 는 이러한 예를 도시한다.

예제 1. 파이프라인 흐름의 ADL 기술

```
stage Ife : 1 {
    act  {
        read_code_from_memory( );
        split {
            enter Ife; // 자식 프로세스 생성
        }
        enter Dec; // Dec 스테이지로 진입
    }
};
```

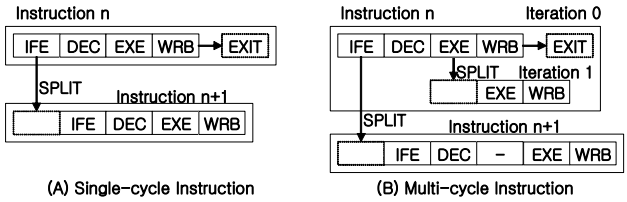

그림 2. 파이프라인 흐름 제어

본 스케줄러는 그림 3-A 와 같은 구조의 시간 바퀴(time wheel)을 이용하여 프로세스를 예약한다. 시간 바퀴는 시간 슬롯으로 구성된 환형 큐 구조로 이루어졌다. 하나의 시간 슬롯은 한 클럭 사이클을 의미하며 시뮬레이션 상 동일 시점의 프로세서는 동일 시간 슬롯에 단일 연결 사슬 구조로 예약된다. 따라서 스케줄러는 현재의 시간 슬롯에 예약된 프로세스를 하나씩 제거하면서 수행하며 더 이상 프로세스 큐에 프로세스가 존재하지 않으면 다음 슬롯으로 이동한다. 예약된 프로세스가 존재하는 시간 슬롯을 구분하기 위해 개개의 시간 슬롯은 활성 상태 비트를 가지고 있다. 이러한 시간 슬롯은 시간 바퀴 상 단일 연결 사슬 구조를 이루며 스케줄러는 시간 슬롯을 일일이 선택하는 대신 연결 사슬을 따라 활성화된 시간 슬롯만을 선택한다. 그림 3-B 는 이러한 프로세스의 상태 천이도를 도시한다.

STEP N 이 실행되면 현재 프로세스는 실행이 중단된다. N 이 시간 바퀴의 전체 시간 슬롯 개수보다

작을 경우 중단된 프로세스는 현재 시간 슬롯으로부터 N 만큼 떨어진 시간 슬롯에 예약된다. N 이 클 경우, 프라이어티 큐에 별도로 저장된다. 스케줄러는 시간 슬롯이 바뀔 때마다 프라이어티 큐를 참조하여 프로세스 예약을 갱신한다. 그림 4 는 STEP 이 실행되었을 경우, 시간 슬롯 상태를 도시한다. 새로 예약된 시간 슬롯이 활성화 상태일 경우 바로 프로세스 큐에 저장한다. 비활성화일 경우, 예약 시간 슬롯을 활성화 시간 슬롯 리스트에 새로 포함시킨다.

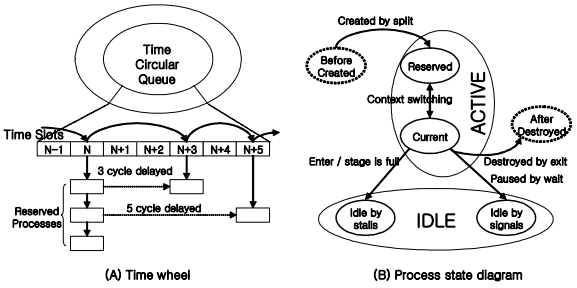

그림 3. 시간 바퀴 구조 및 프로세스 상태 천이도

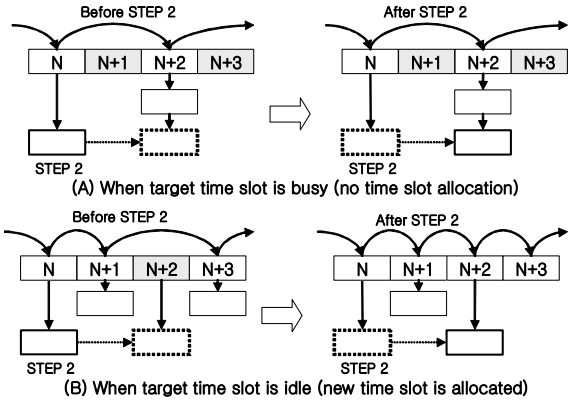

그림 4. STEP 실행 시 시간 슬롯 상태

개개의 파이프라인 스테이지는 동시에 수용할 수 있는 프로세스의 개수를 명시할 수 있다. 개개의 파이프라인 스테이지는 카운트 변수를 가지며 이 카운트 변수는 최대 수용 개수보다 크거나 작다. ENTER X 명령어가 수행될 경우, 현재 프로세스가 현재 시간 슬롯으로부터 제외되면서 카운트가 1 만큼 감소한다. 이후 스케줄러는 현재 파이프라인 스테이지에서 진입하지 못하고 대기 중인 프로세스의 유무를 검사하고 존재할 경우, 현재 시간 슬롯으로 하나의 프로세스를 영입하면서 카운트는 다시 1 만큼 증가한다. 제외된 현재 프로세스의 소속 파이프라인 스테이지는 X 로 천이되며 스케줄러는 X 스테이지의 카운트를 검사한다. 만약 카운트가 최대 값보다 클 경우, 프로세스는 1 사이클 지연된 시간 슬롯에 예약되고 카운트가 1 만큼 증가한다. 카운트가 최대값과 동일할 경우, 이는 파이프라인 스테이지가 완전히 찼다는 의미가 되므로 해당 프로세스는 X 파이프라인 스테이지의 대기 큐에 예약되며 카운트는 변화 없다. 이러한 메커니즘에 따라 파이프라인 스톨 및 전진에 대한 완벽한 기술이 가능하다. 그림 5 는 파이프라인 스톨이 발생하였을 경우의 파이프라인 흐름을 도시한다.

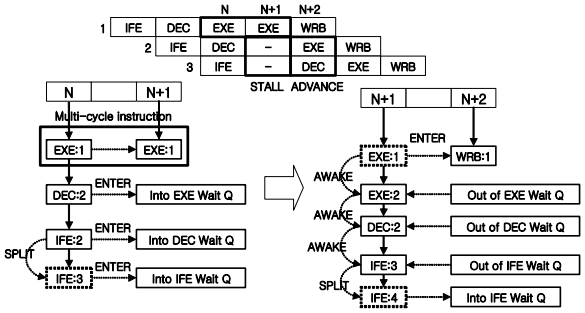

그림 5. 스톨 발생 시, 파이프라인 흐름 처리 절차

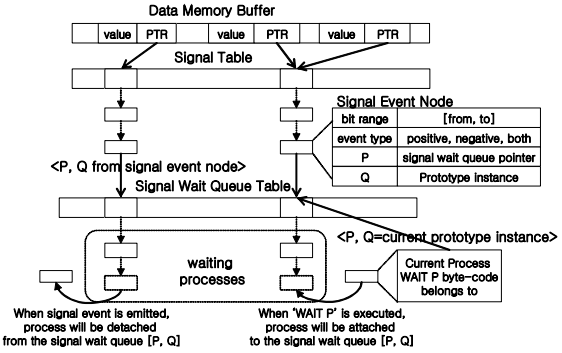

그림 6. 신호 테이블의 구조

본 프레임워크는 파이프라인 구조와 비동기적으로 발생하는 인터럽트 처리 및 코프로세서 인터페이스 등을 처리하기 위해 하드웨어 기술 언어(HDL)와 유사하게 변수에 기반한 이벤트 구동 방식도 추가로 지원한다. WAIT 명령어가 수행될 경우, 현재 프로세스는 실행이 중단되고 신호 대기 큐에 예약 대기된다. 신호 변수는 자체 값 외에 신호 테이블 상 원소를 가리키는 포인터를 지니고 있으며 여기에는 신호 이벤트 리스트가 존재한다. 신호 변수 값이 갱신될 때마다 스케줄러는 신호 이벤트 리스트를 탐색하여 신호 이벤트 발생 여부를 결정한다. 신호

이벤트가 발생하였을 경우, 신호 이벤트에 연관된 신호 대기 큐에 예약 대기 중인 프로세스가 모두 현재의 시간 슬롯으로 복귀된다. 그림 6 은 이러한 신호 테이블의 구조를 나타낸다.

IV. 가상 머신

본 프레임워크는 시뮬레이션 구현 방식으로 가상 머신 방식을 채택하였다. 본 가상 머신은 바이트 코드 처리 루틴, 프로세스 스케줄러, 바이트 코드 용 메모리 관리기로 구성되어 있으며 그림 7 과 같은 구조로 되어 있다. 프로세스 스케줄링은 선점형 (preempt) 방식으로 발생하며 프로세스 스케줄링 명령어가 수행될 경우에만 이루어진다.

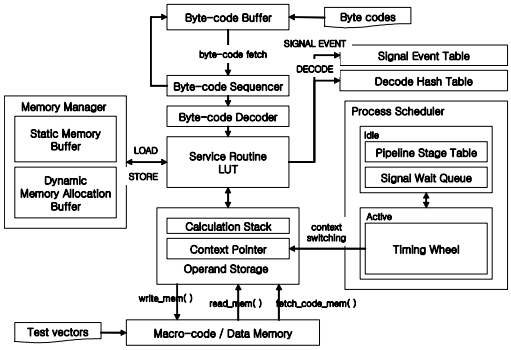

그림 7. 가상 머신의 전체 블록도

V. 시뮬레이션 결과

본 프레임워크의 디자인 플로우는 다음과 같다. 먼저 ADL 의 문법을 정의하고 이를 바탕으로 바이트코드 컴파일러 및 가상 머신을 C++을 이용하여 구현하였다. 본 프레임워크의 성능을 실험하기 위해 ARM7TDMI, 연세대학교에서 개발한 SPARC 및 CALMRISC32 를 선택하였다. 성능 평가 및 비교를 위해 동일 프로세서 군에 대해 각각 순수 C 모델, VerilogTM HDL 모델 및 System-C 모델을 적용하고 SUN SPARC60TM 워크스테이션 상 시뮬레이션을 수행하였다. 수행한 워크로드는 랜덤 벡터를 이용하였다. 시뮬레이션 수행 결과는 표 2 와 같다.

표 2. 제안한 프레임워크의 성능 비교

(# of codes/sec)

	ARM7TDMI	SPARC	CALMRISC32
C	19,000	20,000	22,000
HDL	90	120	150
System-C	260	330	370
제안한 프레임워크	520	630	719

제안한 프레임워크는 시뮬레이션 속도가 HDL 모델보다 대략 5.5 배, System-C 보다 2.5 배 더 빠름을 알 수 있다. 비록 C 모델에 비하면 시뮬레이션 속도가 느리나 명령어 군 수준에서 복잡한 모델의 추상화에 요구되는 재목적성을 고려할 경우, 시뮬레이션 속도 저하는 충분히 만회 가능하다.

VI. 결론

본 논문이 제안한 시뮬레이션 프레임워크는 ADL 에 기반하여 명령어 수준 시뮬레이터의 효율성과 레지스터 전송 수준 언어의 융통성을 절충하며 레지스터 전송 수준 언어보다 간소하고 명료하게 명령어 군을 표현할 수 있는 구문을 제공하며 프로세스 스케줄링에 기반한 새로운 타이밍 모델을 사용함으로써 파이프라인 구조에 대한 사이클 수준의 완벽한 타이밍 모델을 지원한다. 실험 결과, 제안한 구조는 HDL 모델에 비해 약 5.5 배, System-C 모델에 비해 약 2.5 배 시뮬레이션 속도가 더 빠르므로 SoC 멀티미디어 칩셋의 근간을 이루는 응용 특화 명령어 군 프로세서 (ASIP)의 모델링에 효과적으로 적용될 수 있다.

참조문헌

[1] Z.V., et al: LISA – Machine Description Language and Generic Machine Model for HW/SW Co-design, IEEE Workshop on VLSI Signal Processing (1996)

[2] Cho, S., et al: CALMRISC32: A 32bit Low-power MCU Core. IEEE Proceeding of AP-ASIC2000, Cheju, Korea (2000)