ETRI

2012
Electronics and
Telecommunications
Trends

정보통신 미래원천기술 특집

Ⅰ. 플랫폼 기반 설계
Ⅱ. 가상 플랫폼
Ⅲ. 멀티미디어 플랫폼
Ⅳ. 멀티 코어 가상 플랫폼
Ⅴ. 결론

멀티미디어 응용을 위한 멀티 코어 가상 플랫폼 개발

Development of Multi-Core Virtual Platform for Multimedia Applications

장준영 (J.Y. Chang) SW-SoC 융합기획연구팀 팀장
이후성 (H.S. Lee) SW-SoC 융합기획연구팀 선임연구원
손명희 (M.H. Son) SW-SoC 융합기획연구팀 선임연구원
임성호 (S.H. Im) SW-SoC 융합기획연구팀 책임연구원
김성일 (S. Kim) 융합부품기획연구팀 선임연구원
안승호 (S.H. Ahn) 융합부품기획연구팀 팀장
박성수 (S.S. Park) 융합부품 SW-SoC 미래기술연구부 부장

본고에서는 멀티미디어 응용을 위한 멀티 코어 가상 플랫폼 설계 및 검증 방법에 대해서 기술한다. 최근에 멀티미디어 응용인 MPEG-4, H.264, HEVC(High Efficiency Video Coding), 3D 및 홀로그램과 같은 대용량 데이터를 처리하기 위해 다수 개의 코어로 구성된 멀티 코어 플랫폼을 사용한다. 기존의 RTL(Register Transfer Level) 기반의 멀티 코어 플랫폼에서 멀티미디어 응용을 설계하고 검증하는데 시뮬레이션 시간에 의한 제약 사항이 존재한다. 이를 해결하기 위해 시스템 수준에서 하드웨어의 SW 모델로 구성된 가상 플랫폼을 사용한다. 가상 플랫폼은 기존의 RTL 플랫폼보다 100~200배 빠른 고속 시뮬레이션이 가능하므로 멀티미디어 응용에 따른 성능 분석 및 구조 탐색을 통해서 시스템 성능을 향상 시킬 수 있다. 본고에서는 8~32개 멀티 코어 가상 플랫폼에 H.264 디코더를 적용하여 성능 분석하는 방법과 실험 결과에 대해서 기술한다.

© 2012 한국전자통신연구원

Ⅰ. 플랫폼 기반 설계

칩의 복잡도와 SoC(System-on-Chip) 제품의 생산성 차이가 계속적으로 증가함에 따라 현재의 IC 설계 방법으로는 SoC 제품의 성능과 요구의 변화를 만족시킬 수 없다. 칩의 면적을 최소화하고 성능을 최대화하며 게이트 수준의 최적화를 통한 기존의 셀 기반 설계 방법으로는 설계의 생산성 문제를 해결할 수 없다. 이러한 문제를 해결하기 위해 새로운 설계 방법인 IP(Intellectual Property) 재사용을 기반으로 한 플랫폼 기반 설계가 제시되었다. 플랫폼 기반 설계는 SoC 제품을 빠르게 개발하기 위한 응용 기반 플랫폼과 재사용이 가능한 IP들을 이용한 플랫폼을 기반으로 한다. 플랫폼 기반 설계 방법은 90% 이상의 IP 재사용을 통해서 설계 시간을 단축하고, 시스템 수준에서의 최적화를 통해서 제품의 시장 경쟁력(time-to-market)의 문제를 해결하기 위한 방법이다.

1. 기술/응용-기반 플랫폼

플랫폼 기반 설계는 기술-기반 플랫폼(technology-driven platform)과 응용-기반 플랫폼(application-driven platform)으로 구분된다.

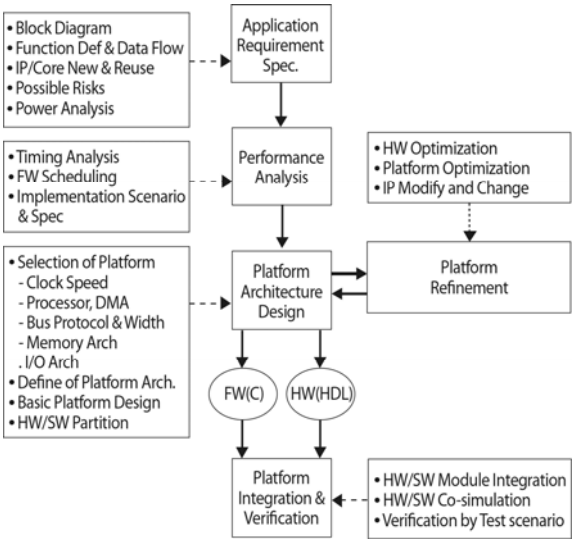
기술-기반 플랫폼은 모듈 설계를 한 후에 성능 분석을 통해서 최적화 및 검증을 실행한 후에 IP 라이브러리를 사용하여 칩을 설계하는 상향식 설계(bottom-up design) 방법이다.

응용-기반 플랫폼은 응용 분야의 요구 사항에 따라서 다양한 계열로 나누고, 이를 분석하여 적합한 플랫폼의 구조를 결정한 다음, 실제 IP 라이브러리를 적용하여 칩을 설계하는 시스템 수준 설계의 하향식 설계(top-down design) 방법이다. 응용-기반 플랫폼은 상위 수준에서 응용 분야에 따라서 다양한 성능 분석이 가능하므로 개발 실패 부담을 최소화할 수 있다. 미리 정의된 블록이나 모델들을 재사용하므로 개발 시간을 단축할 수 있는 장점이 있다. 플랫폼을 이용한 설계 방법은 높은 초기 플랫폼 설계 비용과 미리 정의된 플랫폼으로 인해서 설계의 유연성이 제한되는 어려움은 있으나 개발된 플랫폼을 이용하여 다양한 응용 분야를 개발하는 데에 필요한 시간과 비용의 감소를 통해서 보완될 수 있다.

2. 플랫폼 기반 설계 절차

플랫폼을 구성하는 단계에서는 먼저 적용하고자 하는 응용 분야의 영역과 제품의 발전 방향을 분석하고 정의한다. 초기 플랫폼을 구성하기 위해서 초기 플랫폼 구조를 구성하는 프로세서와 DSP를 선정하고, 데이터 전송 및 처리를 위한 통신 구조와 내부 및 외부 메모리를 선택한다((그림 1) 참조).

다음으로 응용 분야에 적합한 하드웨어 및 소프트웨어 IP를 결정하고 IP에 필요한 인터페이스를 설계한다. 성능 분석 및 플랫폼 선택 및 구성 과정에서 메모리 대역폭과 전력 소모 및 성능들을 분석 및 평가하여 응용 분야의 요구 사항을 만족하는 플랫폼을 선택하고 구성한다. 플랫폼 재정의(refinement) 단계에서는 응용 분야를 개발하기 위한 플랫폼을 조정 및 최적화하는 과정으로 버스 크기, clock 속도, 데이터 전송 프로토콜에 따라



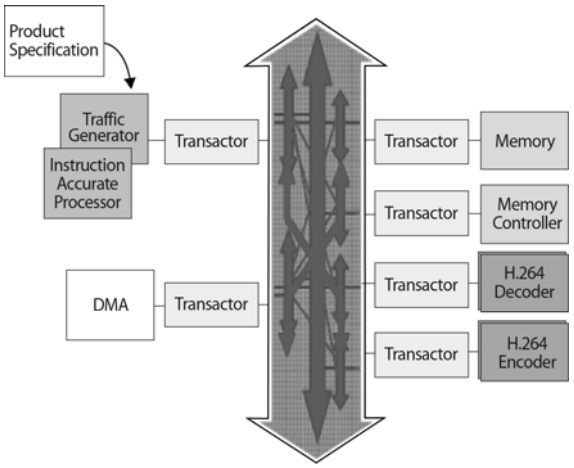
(그림 1) 플랫폼 기반 설계 흐름도

서 각 모듈들을 최적화한다. 하드웨어와 소프트웨어 모듈을 분할하고 소프트웨어의 사이클을 측정하여 태스크 scheduling을 생성한다.

플랫폼 구현과 검증 단계에서 분할된 소프트웨어 모듈은 플랫폼의 프로세서에서 실행되며 하드웨어 모듈은 플랫폼에 연결된다. 다음으로 소프트웨어와 하드웨어 사이의 통합 시뮬레이션을 통해서 구현된 플랫폼을 검증한다. 플랫폼 기반 설계는 플랫폼을 설계하는 과정과 플랫폼을 사용하여 응용 분야 제품을 적용하는 과정으로 구분된다[1]. 플랫폼을 설계하는 과정에서는 응용 분야에 따른 IP나 하드웨어 및 소프트웨어 모듈을 특성화하여 선택하며, 시스템 구조를 결정하고 성능과 전력 소모를 최적화하기 위한 성능 분석 과정을 통해서 플랫폼을 결정하고 선택한다. 플랫폼을 사용하는 과정에서는 플랫폼의 응용 분야에 필요한 IP 및 하드웨어 및 소프트웨어 모듈을 최적화하여 통합하고 검증을 실행한다.

II. 가상 플랫폼

가상 플랫폼(virtual platform)이란 SoC 하드웨어를 SystemC로 모델링되어진 SW로 구성된 시스템 레벨 플랫폼을 말하며 (그림 2)와 같다. 시스템 레벨 플랫폼은

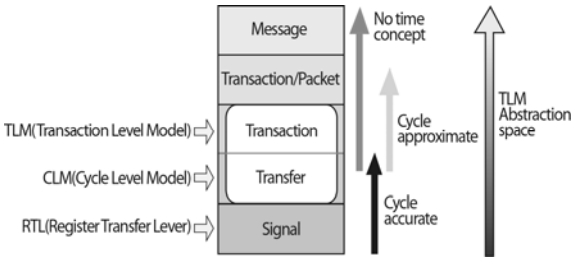


(그림 2) 가상플랫폼

SoC 개발 초기에 기존의 하드웨어 모듈을 재사용하여 플랫폼을 구축하고 시스템 레벨의 구조 탐색과 최적화를 통해서 빠른 SoC 개발이 용이하게 한다.

하드웨어 모듈은 개발 초기에 하드웨어를 모델링하여 플랫폼을 구성한 후 소프트웨어 개발자에게 하드웨어 플랫폼을 제공함으로써 하드웨어와 소프트웨어 개발자가 동시에 개발할 수 있어 개발 시간을 단축할 수 있다. SoC 설계 있어서 시스템 수준에서 하드웨어와 소프트웨어 개발자에게 RTL(Register Transfer Level), CLM (Cycle Level Model), TLM(Transaction Level Model) 등 다양한 형태의 모델을 사용할 수 있게 함으로 유연하고 재사용이 가능한 플랫폼 환경을 제공하고 시스템 수준의 구조 탐색과 최적화를 용이하게 한다. SoC 설계를 위한 가상 플랫폼을 구성하기 TLM 모델의 추상화 레벨은 (그림 3)과 같다. TLM 추상화 레벨 중에서 신호 레벨은 RTL의 신호 수준의 모델로 기존의 RTL로 설계된 IP를 말한다. CLM은 데이터 전송 레벨로 하드웨어 IP의 포트로 전송되는 데이터가 사이클 수준으로 동작되는 모델을 의미한다.

TLM은 트랜잭션으로 데이터 전송을 표현하는 모델이다. TLM 모델로 갈수록 추상화 레벨이 높아지고, time 개념이 없으므로 빠른 시뮬레이션이 가능하다. CLM은 RTL과 같이 사이클 수준의 time 개념으로 동작하므로 기존의 RTL로 구성된 IP를 이용하여 모델링이 가능하다. TLM은 트랜잭션 수준의 데이터 전송이 가능하므로 빠른 시뮬레이션이 가능하고 개발초기에 플랫폼



(그림 3) TLM 추상화 레벨

을 구성하여 소프트웨어 개발자에게 전달하므로 공동 개발이 가능하다.

시스템 수준에서 TLM 플랫폼은 기존의 RTL 수준의 IP도 쉽게 재사용이 가능하고, TLM으로 개발된 IP나 도입된 IP들을 이용하여 쉽게 플랫폼을 구성이 가능하므로 하드웨어 개발자나 소프트웨어 개발 모두에게 유연하고 사용하기 편리한 SoC 개발 환경을 제공한다.

가상 플랫폼을 위한 설계 도구는 Synopsys사의 Platform Architecture[2], Carbon사의 SoC Designer Plus, Mentor의 Vista 등이 있다. 또한 SystemC 모델링 도구로는 CLM을 TLM으로 변환하는 도구로 Carbon사의 Model Studio[3] 등이 있다.

가상 플랫폼 설계 도구인 Platform Architecture는 크게 PCT(Platform Creator), SCSH(SystemC shell), TLM 라이브러리, SCIDE(SystemC Debugger)로 구성된다. PCT는 TLM 라이브러리나 사용자가 설계한 TLM 또는 CLM 모델을 등록하여 플랫폼을 구성하는 도구이다. SCSH는 PCT에서 생성된 가상 플랫폼을 시뮬레이션하는 도구로 플랫폼의 동작 및 에러를 검증하여 디버깅하는 도구이다.

Platform Architecture는 ARM 프로세서와 AMBA (Advanced Microprocessor Bus Architecture) 버스 라이브러리인 AHB(Advanced High performance Bus), AXI(Advanced eXtensibleInterface)[4]들과 주변장치 라이브러리와 ARM, DMAC(Direct Memory Access Controller), MPMC(Multi-Port Memory Controller) 등의 TLM 라이브러리를 제공한다. SCIDE는 사용자가 설계한 SystemC 코드를 디버깅하는 데 사용한다.

본고에서는 플랫폼 설계 도구인 Platform Architecture를 이용하여 가상 RTL, CLM, TLM 플랫폼을 구성하고 각각의 가상 플랫폼을 이용하여 멀티미디어 응용인 H.264 디코더를 설계하고 검증하는 방법에 대해서 설명한다.

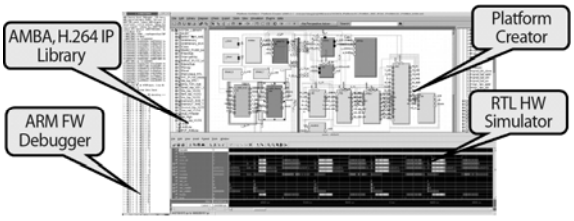
Ⅲ. 멀티미디어 플랫폼

1. 가상 RTL 플랫폼

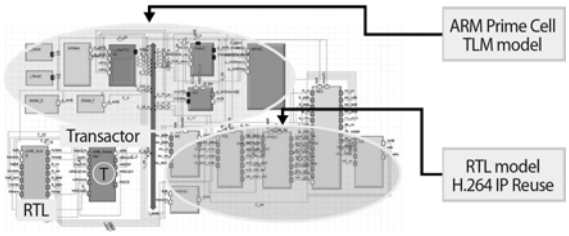
가상 RTL 플랫폼은 기본 시스템인 ARM 프로세서, AMBA, DMA, MPMC, SDRAM은 SystemC TLM 라이브러리를 이용하고 H.264 디코더 IP들은 RTL로 모델링되어 transactor를 통해서 연결되는 플랫폼이다.

가상 RTL 플랫폼을 이용함으로 H.264 디코더 RTL IP를 재사용하고, TLM과 RTL의 통합 시뮬레이션이 가능하고 각 RTL 모듈의 기능을 검증하다. RTL 모듈을 socket 방식으로 구성이 가능하므로 유연한 가상 플랫폼이 가능하다. AHB 버스에 RTL slave 모듈을 연결하기 위해 AHB slave의 wrapper인 transactor를 이용한다. H.264 디코더의 마스터 모듈인 ARM 프로세서와 데이터 전송 마스터인 DMA와 메모리 컨트롤러인 MPMC 등은 TLM 라이브러리를 사용함으로 RTL로 구성된 플랫폼보다 고속 시뮬레이션이 가능하다.

Platform Architecture를 이용한 H.264 디코더를 구현하기 위한 platform의 구성 요소는 (그림 4)와 같다. Platform creator에 사용하기 위한 IP들을 라이브러리로 등록하고, 등록된 라이브러리를 이용하여 H.264 디코더를 구성한다. 시뮬레이션을 통해서 H.264 디코더의 firmware는 ARM 컴파일러로 컴파일된 후 ARM 프로세서에서 실행되어 실행 결과가 ARM 디버거를 통해서 터미널 화면으로 출력되고, 하드웨어 모듈은 HDL 시뮬레이터인 Modelsim을 통해서 wave 형태로 출력된다. 따라서 출력된 wave 형태를 보고 시스템의 동작을



(그림 4) 가상 RTL 플랫폼



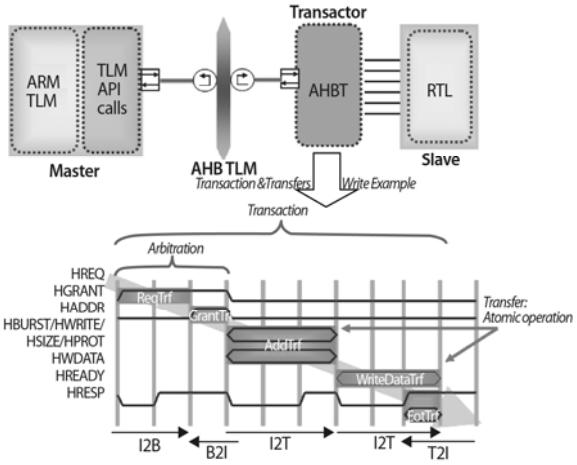
(그림 5) Transactor를 이용한 가상 RTL 플랫폼

검증한다.

가상 RTL 플랫폼의 구성 요소는 ARM 프로세서, ARMBA 버스와 DMAC, MPMC TLM 모델을 이용한다. H.264 디코더 RTL IP들은 RTL로 모델링되어 AHB slave transactor를 통해서 연결된다((그림 5) 참조).

AHB slave transactor는 RTL의 signal level transfer를 TLM의 transaction level transfer로 변환하는 장치로서 Platform Architecture의 API(Application Program Interface)로 구현된다. ARM TLM 모듈은 RTL salve 모듈과 AHB slave transactor를 통해서 연결된다. AHBT는 AHB slave transactor로 TLM 신호를 signal 수준의 CLM으로 변환하는 장치이다.

(그림 6)은 AHB TLM 마스터 모듈에서 RTL slave 모듈에 데이터를 write하는 transaction과 데이터 전송 예를 나타낸다. 이 transaction은 Platform Architecture에서 제공하는 TLM API이다.

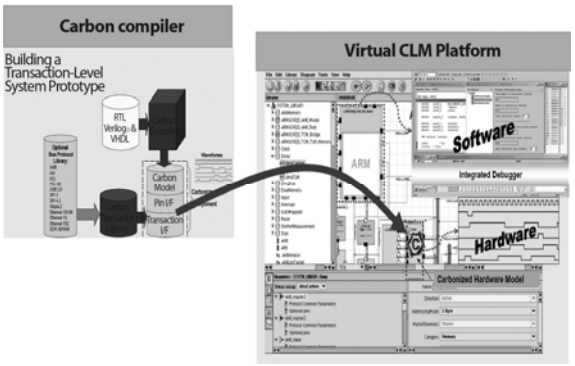


(그림 6) AHB Slave Transactor

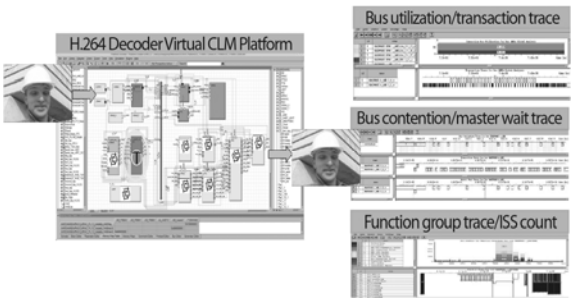
2. 가상 CLM 플랫폼

가상 CLM 플랫폼은 ARM 프로세서, AMBA, DMA, MPMC, SDRAM은 SystemC TLM이고 H.264 디코더 IP들은 SystemC CLM으로 모델링되어 transactor를 통해서 연결되는 플랫폼이다.

H.264 디코더 응용 IP들은 SystemC CLM으로 모델링되어 AHB slave transactor를 통해서 연결된다. H.264 디코더의 RTL IP들을 SW 모듈로 컴파일해주는 도구인 carbon compiler[3]를 사용하여 CLM 수준의 목적 코드를 생성한 후 가상 플랫폼에 적용한다. 가상 CLM 플랫폼은 RTL을 목적 코드 형태의 CLM 모델로 변환함으로 시뮬레이션 속도를 향상시킬 수 있다. (그림 7)은 CLM 모델로 구성된 가상 CLM 플랫폼으로 가상 RTL 플랫폼보다 시뮬레이션 속도가 빠르면서 사이클 수준의 검증이 가능하다. (그림 8)은 H.264 디코더 구현을 위한 가상 CLM 플랫폼과 시뮬레이션 결과를 보여준다.



(그림 7) 가상 CLM 플랫폼



(그림 8) 가상 CLM 플랫폼에 의한 H.264 디코더 구현

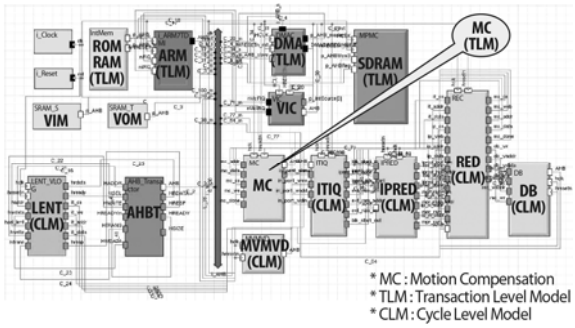
가상 CLM 플랫폼에 의한 H.264 디코더 실행 결과를 통해서 H.264 하드웨어 모듈의 기능 검증이 가능하고 사이클 수준의 성능 분석이 가능하다.

3. 가상 TLM 플랫폼

가상 TLM 플랫폼은 ARM 프로세서, ARMBA, DMAC, MPMC 등을 이용하고, H.264 디코더 IP들은 SystemC TLM으로 모델링되어 AHB TLM 버스에 직접 연결된 가상 플랫폼으로 (그림 9)와 같다. H.264 디코더 IP들은 SystemC TLM으로 모델링하여 AHB 버스에 직접 연결한다. 가상 TLM 플랫폼은 높은 추상화 레벨을 통해서 시뮬레이션 실행 시간이 가장 빠르다. 가상 TLM 플랫폼은 상위 수준에서 트랜잭션 레벨의 모델을 사용하여 시뮬레이션을 함으로 빠르게 기능 검증 및 성능 분석이 가능하나, 사이클 수준의 성능 분석은 어렵다.

4. 성능 분석 및 실험 결과

가상 RTL, CLM, TLM 플랫폼 설계 및 검증 환경은 PC(3.2GHz)의 Linux 2.6.9 환경하에서 시스템 수준 설계 도구인 Platform Architecture v2011을 이용하여 구현되었다. 가상 플랫폼을 검증하기 위한 데이터로는 CIF(352×288)급 foreman 동영상을 사용하여 I, P, P 프레임을 시뮬레이션한 결과는 (그림 10)과 같다.



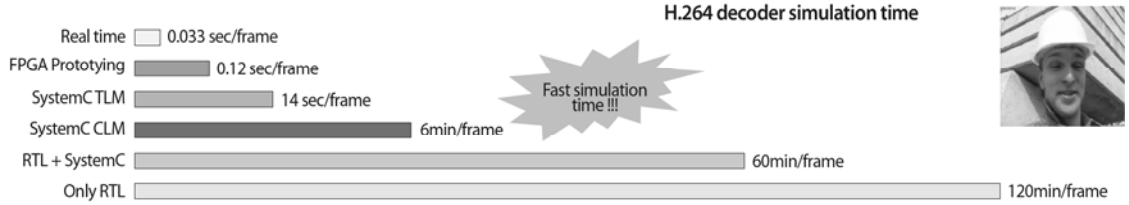
(그림 9) 가상 TLM 플랫폼에 의한 H.264 디코더 구현

가상 RTL 플랫폼에서는 1 프레임 처리하는 데 2시간의 시뮬레이션 시간이 사용되었다. 가상 RTL 플랫폼(RTL+SystemC)에서는 프레임당 1시간이 소요되었다. 가상 CLM 플랫폼에서는 프레임당 6분의 시뮬레이션 시간이 소요되었다. RTL보다 CLM 플랫폼이 약 8~10배 정도의 빠른 시뮬레이션 속도를 보인다. 모든 H.264 IP들을 SystemC TLM으로 모델링된 경우는 frame당 14초로 CLM에 비해 100~200배 이상의 빠른 시뮬레이션 속도를 보인다. 하드웨어의 추상화 레벨이 높아짐에 따라서 빠른 시뮬레이션 속도가 보장되며, 기존의 RTL IP들을 이용하여 TLM 모듈과 통합 시뮬레이션이 가능하게 되어 가상 플랫폼의 유연성이 높아진다.

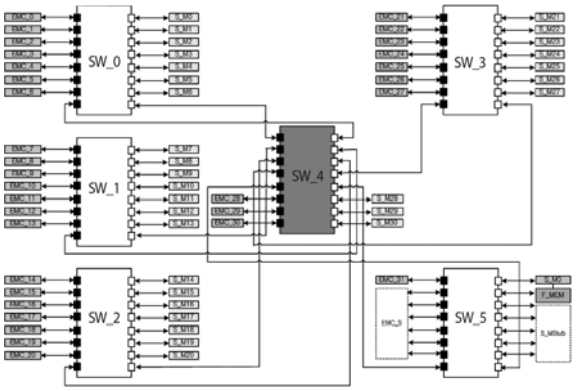
또한 상위 수준에서 가상 플랫폼을 이용한 H.264 디코더의 성능 분석 결과[6] 전체 버스 사용률이 80%를 사용하고 ARM 프로세서가 47.2%, DMAC가 33.4%를 차지함으로 Multi-layer Bus Matrix, Multi-Channel AXI 버스 구조나 최대의 병렬성을 제공하는 NOC (Network-On-Chip)[5]를 가진 가상 플랫폼이 필요함을 알게 되었다.

Ⅳ. 멀티 코어 가상 플랫폼

최근에 멀티미디어 응용인 MPEG-4, H.264, HEVC (High Efficiency Video Coding), 3D 및 홀로그램과 같



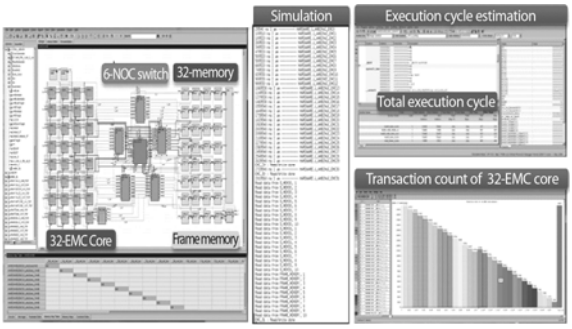
(그림 10) 가상 플랫폼 성능 분석 결과



(그림 11) 멀티 코어 구성도

은 대용량 데이터를 처리하기 위해 다수 개의 동질 코어 또는 CPU, DSP, GPU, 및 ASIP(Application Specific Instruction Processor)와 같은 이질 코어들로 구성된 멀티 코어 플랫폼이 사용된다[7]. 본고에서는 2~32개 이상의 프로세서 코어를 탑재하고 각 프로세서 코어 간 온칩 버스 및 NOC 구조를 가진 멀티미디어 응용을 위한 멀티 코어 가상 플랫폼에 대해 설명한다.

멀티 코어를 이용하여 멀티미디어 응용 소프트웨어로 처리하기 위해서는 시스템 수준의 가상 플랫폼을 이용한 고속의 시뮬레이션 환경 및 설계 및 검증 환경이 필요하다. H.264 디코더를 위한 멀티 코어 구성도는 (그림 11)과 같다. 멀티 코어 플랫폼은 멀티미디어 프로세서 32개와 공용 메모리 32개, 1개의 프레임 메모리와 8M×8S 스위치 6개로 NOC 구조로 연결되었다. 공유 메모리와 프레임 메모리는 NOC를 통해서 병렬로 데이터를 전송한다. NOC는 AHB와 같은 온칩 버스의 공유로 인한 지연 시간을 제거할 수 있다. 이와 같이 32개의 멀티미디어 프로세서로 구성된 가상 플랫폼을 통해서 고속 기능 검증 및 성능 분석을 통해서 멀티미디어 응용 분야의 요구 사항에 따라 시스템 수준의 구조적 탐색이 가능하다. H.264 디코더를 위한 멀티 코어 가상 플랫폼은 다음 (그림 12)와 같다. 멀티미디어 프로세서와 NOC 구조의 RTL은 SystemC wrapper를 사용하여 라이브러리로 등록하고, 공유 메모리와 프레임 메모리는 TLM

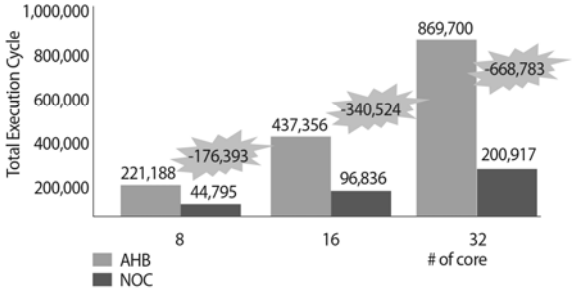


(그림 12) 멀티 코어 가상 플랫폼

라이브러리를 사용하여 가상 플랫폼을 구성하며, 고속 시뮬레이션을 통해서 기능을 검증하고 성능 분석한다. 프로세서 코어들 간의 통신 구조는 NOC를 사용하여 구성하였다. H.264 디코더는 소프트웨어 모듈로 32개의 코어에서 병렬로 실행된다.

8, 16, 32개의 멀티미디어 코어를 온칩 버스와 NOC를 사용하여 멀티미디어 응용 프로그램에서 10, 100, 500, 1000개의 transaction을 동시에 발생시켜 전체 실행 시간을 측정한 결과는 (그림 13)과 같다. 실험 결과, 8개의 코어를 사용하는 경우 NOC 구조가 온칩 버스를 사용한 구조보다 약 5배의 성능이 향상 되고, 32개의 코어를 사용하는 경우 약 4배의 성능 개선이 있다.

Performance Analysis(total execution cycle)						
#of Core	8		16		32	
#of Trans.	AHB	NOC	AHB	NOC	AHB	NOC
10	2,398	504	4,726	1,053	9,390	2,174
100	22,277	4,520	44,045	9,761	87,581	19,668
500	110,715	22,422	218,931	48,471	435,364	97,367
1,000	221,188	44,795	437,356	96,836	869,700	200,917



(그림 13) 멀티 코어 가상 플랫폼 성능 분석

V. 결론

본고에서는 상위 수준의 멀티미디어 설계 및 검증 환경을 구축하기 위해서 SystemC 수준의 모델을 라이브러리로 등록하고 플랫폼을 생성한 후에 성능 분석 및 구조 탐색이 가능한 가상 플랫폼을 설계하고 검증하는 방법에 대해서 설명하였다. 다양한 형태의 멀티미디어 응용을 위한 IP를 재사용하여 RTL, CLM 및 TLM 플랫폼을 설계하고 이를 이용하여 H.264 디코더를 검증하였다. 각 플랫폼에서 H.264 디코더 IP들을 모델링하여 적용한 결과, RTL IP들로 구성된 가상 RTL 플랫폼보다 CLM 플랫폼이 약 10배 빠른 시뮬레이션 속도를 보이며, TLM 플랫폼에서는 약 100~200배의 빠른 시뮬레이션 속도를 보인다. 8~32개로 구성된 멀티 코어 가상 플랫폼을 이용하여 실험한 결과, NOC 구조가 AHB 버스 구조보다 4~5배의 성능이 향상 되는 것을 알 수 있다.

용어해설

SoC(System-on-Chip) 일정량 이상의 복잡도를 가지는 다수의 칩의 기능을 하나의 칩으로 구현하는 것

플랫폼 기술(Platform Technology) 어떤 작업 또는 기술 구현이 이루어질 수 있는 '공통 · 공용의 표준화된 하드웨어 및 소프트웨어 틀(환경)'을 말하는데, 플랫폼의 최대 이점은 SoC 플랫폼 내 기능 블록, 입력과 출력 등이 이미 '최적화되고 검증'되어 있으며, 검증된 기본블록 및 소프트웨어들을 응용 분야에 따라 기능을 추가하거나 삭제하여 '파생제품을 신속하게 설계' 가능한 장점이 있음.

NOC(Network-on-Chip) 다양한 통신 인터페이스를 가진 멀티 코어들이 네트워크 인터페이스를 통해서 스위치에 연결되는 구조

약어 정리

AHB	Advanced High performance Bus
AMBA	Advanced Microprocessor Bus Architecture
API	Application Program Interface
ASIP	Application Specific Instruction Processor
AXI	Advanced eXtensibleInterface
CLM	Cycle Level Model
DMAC	Direct Memory Access Controller
HEVC	High Efficiency Video Coding
IP	Intellectual Property
MPMC	Multi-Port Memory Controller
NOC	Network-On-Chip
RTL	Register Transfer Level
SoC	System-on-Chip
TLM	Transaction Level Model

참고문헌

[1] H. Chang et al., *Surviving the SOC Revolution: A Guide to Platform-Based Design*, Kluwer Academic Publishers, Nov. 1999, ARM Ltd.

[2] Synopsys Inc. Platform Architecture. http://www.synopsys.com/Systems/ArchitectureDesign/pages/PlatformArchitect.aspx

[3] Carbon Design Systems. http://carbondesignsystems.com

[4] "AMBA™ Specification Rev 2.0," ARM IHI 0011A, 1999.

[5] L. Benini and G. Micheli, "Networks on Chips: A New SoC Paradigm," *Comput.*, vol. 35, no. 1, Jan. 2002, pp. 70-78.

[6] J.-Y. Chang et al., "On-Chip Network Based Virtual Platform for Multimedia Applications," *Int. Tech. Conf. Circuits/Syst. Comput. Commun.*, vol. 2, July 2007, pp. 649-650.

[7] J.-Y. Lee, J.-J. Lee, and S.M. Park, "Multi-core Platform for an Efficient H.264 and VC-1 Video Decoding Based on Macroblock Row-Level Parallelism," *IET Circuits Devices Syst.*, 2010, vol. 4, no. 2, pp. 147-158.