한국산학기술학회논문지
Vol. 10, No. 7, pp. 1480-1483, 2009

ARM-Excalibur를 이용한 H.264/AVC 디코더의 HW/SW 병행 설계

정준모[1*]
[1]군산대학교 전자공학과

HW/SW co-design of H.264/AVC Decoder using ARM-Excalibur

Jun-Mo Jung[1*]
[1]**Depart. of Electronic Engineering, Kunsan National University**

요 약 본 논문에서는 H.264 및 AVC 디코더를 ARM-Excalibur를 이용하여 하드웨어(HW:Hardware)와 소프트웨어(SW:Software)로 병행설계(co-design)하는 방법에 대해서 제안한다. 내장형 프로세서, 메모리, 주변장치 및 논리 회로들을 하나의 칩으로 집적한 SoC(System On-a-Chip)를 하드웨어와 소프트웨어로 분할하여 병행 설계(co-design)하는 방식이 새로운 설계 방법으로 대두되고 있다. 최적화된 분할 방법을 찾는 것이 매우 어렵기 때문에 설계 초기단계에서 빠르게 검증할 필요가 있는데 본 논문에서는 H.264 및 AVC 디코더를 알테라사의 ARM-Excalibur라는 칩을 이용하여 효율적으로 병행 설계하였으며 시스템의 동작속도가 크게 향상되는 것을 확인할 수 있었다.

Abstract In this paper, the hardware(HW) and software(SW) co-design methodology of H.264/AVC decoder using ARM-Excalibur is proposed. The SoC consists of embedded processor, memory, peripheral device and logic circuits. Recently, the co-design method which designs simultaneously HW and SW part is a new paradigm in SoC design. Because the optimization for partitioning the SoC system is very difficult, the verification must be performed earlier in design flow. We designed the H.264 and AVC Decoder using co-design method. It is shown that, for the proposed co-design method, the performance improvements can be obtained.

Key Words : SoC, HW and SW co-design,

1. 서론

최근, 반도체 제조공정 기술의 발달로 집적도가 크게 향상됨에 따라서 내장형 프로세서(embedded processor), 메모리(memory), 주변장치(Peripheral), 그리고 추가적인 로직(logic)들을 하나의 칩으로 묶은 시스템온칩(SoC: System-on-a-Chip)이 각광받고 있다. 하지만, SoC를 실제 ASIC (Application-Specific Integrated Circuit) 으로 구현할 경우, 시스템 설계부터 시작하여 검증, 그 이후 테스트 단계에 이르기까지 많은 시간과 비용이 소모된다. 특히 SoC는 하드웨어(HW)와 소프트웨어(SW) 부분이 동시에 설계되어야 하는데, 설계 공간을 탐색하여 최적화된 분할을 찾는 것은 매우 어려운 문제(NP-Hard: Non-deterministic Polynomial-time Hard)이며, 한번 ASIC으로 구현되고 나면 바꿀 수도 없기 때문에 설계 초기 단계부터 빠르게 검증할 필요성이 있다.

이러한 문제점을 해결하기 위해서, 주요 PLD(Programmable Logic Device) 벤더(Vender)들은 자사의 FPGA(Field Programmable Gate Array)에 프로세서 코어를 내장하여, 사용자가 SoC를 미리 구현해 볼 수 있도록 하고 있다. PLD 벤더들은 이것을 자사의 전략에 따라 서로 다른 이름으로 부르고 있지만, SoC 구현을 위한 최적의 솔루션을 제공한다는 데는 일치한다.

본 논문에서는 Altera[1]의 ARM-Excalibur를 이용하여

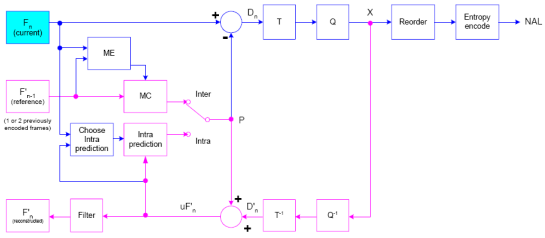
본 연구는 군산대학교 정보통신기술연구소의 부분적인 지원으로 수행되었음.
*교신저자 : 정준모(jmjung@kunsan.ac.kr)
접수일 09년 03월 20일 수정일 (1차 09년 05월 25일, 2차 09년 06월 03일) 게재확정일 09년 07월 22일

빠르게 하드웨어/소프트웨어 병행설계(co-design)하는 방법론을 제시하고, 실제로 H.264/AVC 시스템에 대해서 구현해 보았다. 실험결과는 HW/SW 동시 설계에 따른 이득을 충분히 잘 보여주고 있으며, 본 논문의 연구 결과는 추후 다른 연구에 쉽게 활용할 수 있는 기반을 마련하였다고 할 수 있다.

먼저 2장에서는 H.264/AVC 디코더에 대해서 설명하며 3장에서는 H.264/AVC 디코더를 병행설계 하는 방법에 대해서 서술하고, 4장은 실험결과, 5장은 결론을 설명하였다.

2. H.264/AVC 시스템의 구조

ITU와 MPEG이 공동으로 개발한 멀티미디어 압축 기술인 H.264/AVC는 기존 MPEG-2에 비하여 2배 이상의 압축 효율을 가지고 있기 때문에 오히려 MPEG-2를 밀어내고 차세대 비디오 표준의 핵심 코덱(CODEC)으로 간주되고 있다. 논문 [2]에서도 언급되었듯이, 특히 HD 방송에서 오히려 후발 주자라고 할 수 있는 유럽 각국에서는 HD 방송을 위한 비디오 표준으로 H.264/AVC 기술을 사실상 지정해 놓고 있는 실정이다. H.264/AVC의 부호화기(encoder)의 기본적인 구조는 그림 1과 같으며, 가능한 높은 압축률과 고화질을 위해 많은 기술들을 표준으로 채택하여 부호화(encoding) 및 복호화(decoding)에 있어 매우 높은 복잡도(complexity)를 지니고 있다.



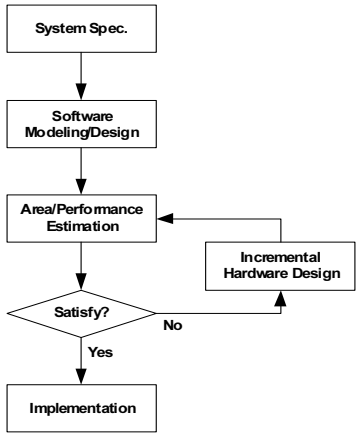
[그림 1] H.264/AVC 복호화기의 전형적인 구조

따라서 H.264/AVC 코덱의 실제 구현이 광범위하게 시도되었음에도 불구하고, ASIC 하드웨어 혹은 대규모 병렬 DSP 방식 외에는 부호화 및 복호화가 모두 어려울 것으로 예상하고 있다. 실제로 소프트웨어적인 접근보다는 주로 ASIC에 의한 하드웨어 접근[3, 4,9,10]과 DSP를 사용한 접근이 주류를 이루고 있다.

하지만, H.264/AVC 코덱의 하드웨어적인 접근방법들은 소프트웨어에 비해서 확장성(extensibility)이 떨어지며, 하드웨어 설계에 많은 시간과 비용이 든다는 단점이 있다. 따라서, 양쪽의 장점을 동시에 가질 수 있는 HW/SW 병행설계가 반드시 필요하다고 할 수 있다.

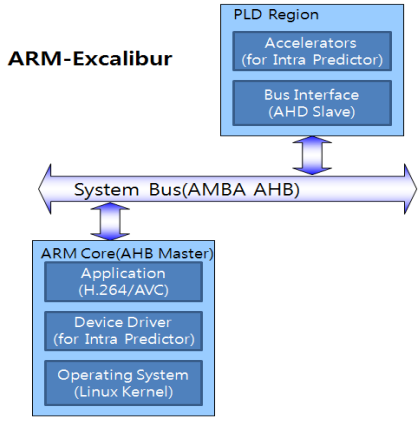
3. 하드웨어/소프트웨어 병행설계(Co-design)

본 논문에서 제시하는 개략적인 HW/SW 병행설계방법론은 그림 2와 같다. 먼저 소프트웨어로 전체 시스템을 모델링(modeling)한 다음, 시스템의 성능을 예측(estimation)하고, 가장 연산(computation)양이 많은 부분을 점진적으로 하드웨어로 설계하면서 시스템의 전체적인 성능을 맞추어 나가는 것이다.



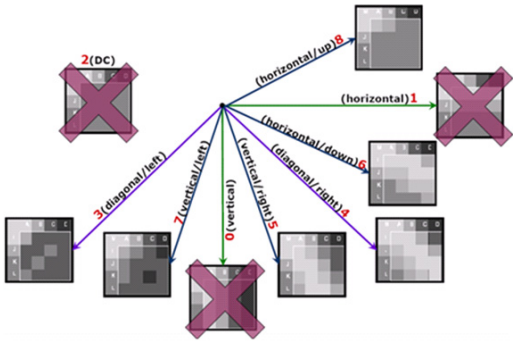
[그림 2] 제안하는 HW/SW 병행설계 방법론

본 논문에서는 H.264/AVC 코덱에서 많은 연산 시간을 소모하는 부분 중 하나인 4x4 모드를 위한 프레임간 예측기(intra-frame predictor)을 하드웨어로 설계하였고, 전체적인 시스템 구조는 그림 3과 같다.



[그림 3] ARM-Excalibur을 이용한 시스템 구조

H.264/AVC 코덱에서 4x4 블록(block) 프레임간 예측기는 그림 4와 같이 크게 9가지 방향에 대해서 예측을 수행한다. 9가지의 방향은 다음과 같다. 우선 DC 성분이 있으며, Horizontal, Horizontal Down, Horizontal Up, Vertical, Vertical Right, Vertical Left, Diagonal Left, Diagonal Right 등이 있다. 각 방향에 대해서 예측기를 수행한다. 따라서 소프트웨어적으로 전 방향에 대해서 연산을 수행하면, 너무 많은 연산시간이 소모되고, 매우 빈번하게 사용되는 부분이므로 전력 소모도 크다고 할 수 있다.



[그림 4] 4x4 block intra prediction directions

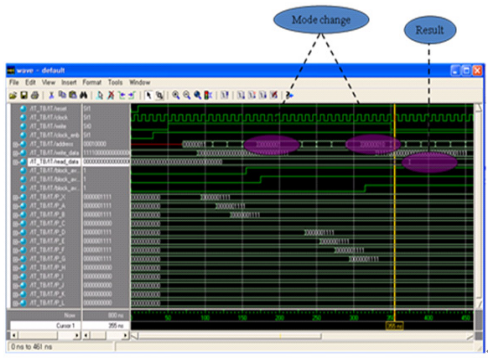
실제 구현상에서는 9가지 방향에 대해서 전부 하드웨어적인 연산으로 동작하도록 구현하지 않고, 0, 1, 2 (Vertical, Horizontal, DC) 모드는 제외하였다. 제외한 이유는 해당 3가지 모드는 굳이 하드웨어 연산 없이 소프트웨어적으로도 매우 쉽게 가능하기 때문이다.

우리는 나머지 방향에 대해서 표 1과 같이 모드를 구분하고, 메모리 어드레스(address)를 할당하였다. 연산할 4x4 블록을 레지스터로 전송하고, 해당 메모리 어드레스를 조정해서 연산 동작 모드를 바꾸면서, 그 결과를 다시 받는 구조로 설계하였다.

[표 1] Direction mode에 따른 어드레스 할당

Direction Mode	Register Name	Address [7:0]
3, 7	block_available_up	00000000
8	block_available_left	00000001
4, 5, 6	block_available_up_left	00000010

그림 5는 ModelSim[5]을 이용한 시뮬레이션 결과를 보여주고 있다.



[그림 5] ModelSim 시뮬레이션 결과

그림 5의 시뮬레이션은 방향 모드(direction mode)에 따라서 4x4 블록에 대한 프레임간 예측기의 결과를 1 cycle 후에 레지스터로 저장하고 있는 모습이다. 모드의 변화(mode change)와 결과를 나타내고 있다.

4. 실험 결과

실험을 위한 ARM-Excalibur 장비는 한백전자[6]의 Hanback Entry II를 사용하였으며, H.264/AVC의 소프트웨어는 JM94[7]를 이용하였다. 그리고 H.264/AVC을 전부 하드웨어로 구현한 참조 모델은 nova[8]를 사용하였다.

리눅스 커널 (Linux kernel), 디바이스 드라이버와 해당 부분이 수정된 JM94 소스 코드는 ARM 코어를 위한 GNU GCC 크로스 컴파일러를 이용하여 컴파일 하였다. 그리고프레임간 예측기는 Verilog HDL을 이용해서 설계하였으며, AMBA AHB BUS 인터페이스(interface)를 지원하도록 하였으며 Synplify을 이용하여 합성하였다.

QuartusII[1]을 사용해서, ARM- Excalibur의 PLD상에 H.264/AVC을 위한 프레임간 예측기를 설계하였으며, 해당 설계를 보드에 다운로드(download)하고, 리눅스 커널과 설계한 디바이스 드라이버를 부팅해서 전체 시스템을 동작시켰다.

QCIF resolution을 가지는 비디오 시퀀스 (video sequence)를 이용하여 테스트를 수행하였고, 레퍼런스 이미지와 동일한 결과를 얻을 수 있었다. 표 2는 전체가 소프트웨어 동작했을 때, 전체가 하드웨어로 동작했을 때, 그리고, 제안한 구조를 가질 때의 성능을 비교하였다.

ARM 코어 및 설계한 가속 모듈은 50MHz로 동작시켰으며, Nova 하드웨어 모델은 1.5MHz 동작시켰다. 하드웨어 크기를 비교하기 위한 게이트 수(gate counts)를 구하

기 위해서, Hynix 0.18um standard cell library를 사용하였고, Synopsys Design Compiler을 이용하여 합성하였다

표 2에 나타낸바와 같이, 게이트수와 디코딩 시간을 기준으로 각 성능을 비교하였다. 비교 대상회로는 시스템을 모두 SW로 구현한 경우(JM94), 하드웨어로만 구현한 경우(Nova) 및 제안한 시스템이다. 우선 게이트 수를 이용한 하드웨어의 크기를 비교 결과이다.

JM94는 0 게이트, Nova는 2.5K byte SRAM을 포함하여 169K의 게이트가 사용된 반면에 제안한 방법은 1K 정도의 게이트만 사용되었으므로 하드웨어의 크기는 Nova에 비하여 상당히 감소되었다. 또한 디코딩 시간은 Silent/30QP 와 Container/30QP에 대해서 비교하였다. Silent/30QP의 경우, JM94의 경우 디코딩 시간이 2m 15sec인 반면에 제안한 방법은 1m 24s로서 매우 빠른 디코딩 시간을 나타내고 있다. 또한 Container/30QP의 경우에도 상대적으로 매우 빠른 디코딩 시간을 보여주고 있다.

[표 2] 각 구현에 따른 성능 비교

	All SW (JM94)	All HW (Nova)	제안한 방법
게이트 수 (0.18um)	0	169K with 2.5K byte SRAM	1K
디코딩 시간 (Silent, 30 QP)	2m 15s	30 fps Real-time	1m 24s
디코딩 시간 (Container, 30 QP)	1m 32s	30 fps Real-time	54s

실제로 프레임간 예측기만을 하드웨어로 구현해서는 실시간(real-time)으로 동작하는 결과를 얻을 수 없었지만, 하나의 많은 연산 시간을 차지하는 부분을 하드웨어로 구현함에 따라서 시스템의 전체 속도가 크게 향상되는 것을 확인할 수 있었다.

5. 결론

본 논문에서는 ARM-Excalibur를 이용하여 빠르게 HW/SW 병행설계(co-design)를 수행하고, 실제로 H.264/AVC 시스템을 구현해 보았다. 실험결과는 HW/SW 병행설계에 따른 이득을 충분히 잘 보여주고 있으며, 본 논문의 연구 결과는 추후 다른 연구에 쉽게 활용할 수 있을 것으로 기대한다.

참 고 문 헌

[1] Altera, Inc., http://www.altera.com

[2] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra, "Overview of the H.264/AVC Video Coding Standard," IEEE Trans. on Circuits and Systems for Video Technology, Vol.13, No.7, pp.560-576, July 2003.

[3] Kelvin Xu, "Power-efficient Design Methodology for Video Decoding," PhD thesis, 2007.

[4] Kelvin Xu and et al., "A 5-stage Pipeline, 204 Cycles/MB, Single-port SRAM Based Deblcoking Filter for H.264/AVC," IEEE Trans. on Circuits and Systems for Video Technology, Vol. 38, issue 3, pp 363-374, 2008.

[5] Modelsim, http://www.model.com

[6] 한백전자, http://www.hanback.co.kr

[7] JM94, http://iphome.hhi.de/suehring/tml/

[8] Kelvin Xu, Nova project, Opencores, http://www.opencores.org

[9] Mustafa Parlark and Ilker Hamzaoglu, " Low Power H.264 Deblocking Filter Hardware Implementations," IEEE Trans. on Consumer Electronics, Vol. 54, No. 2, May 2008.

[10] Yun Cheng and et al., "Research On intra Modes for Inter-Frame coding in H.264," The 9th Int. Conference on Computer Supported Cooperative Work in Design Processings, Vol. 2, 24-26, May 2005.

정 준 모(Jun-Mo Jung) **[종신회원]**

- 1987년 2월 : 한양대학교 전자공학과 졸업
- 1989년 2월 : 한양대학교 전자공학과 석사
- 2004년 2월 : 한양대학교 전자공학과 박사
- 2005년 3월 ~ 현재 : 군산대학교 전자공학과 부교수

<관심분야>
VLSI 설계, ASIC Design, 무선 통신