논문 2008-45SD-9-11

H.264/AVC 디코더를 위한 Embedded SoC 설계

(Embedded SoC Design for H.264/AVC Decoder)

김 진 욱*, 박 태 근**

(Jin Wook Kim and Tae Geun Park)

요 약

본 논문에서는 H.264/AVC baseline 디코더를 ARM926EJ-S 코어를 탑재한 FPGA(XC4VLX60)기반의 타겟 보드와 임베디드용 Linux Kernel 2.4.26의 개발환경에서 SW/HW 분할을 통해 설계 및 구현하였다. 하드웨어 가속기로는 움직임 보상 모듈, 디블록킹 필터 모듈, YUV2RGB 변환 모듈을 사용하였으며 AMBA 버스 프로토콜을 통하여 소프트웨어와 함께 동작한다. 참조 소프트웨어(JM 11.0)를 OS(Linux)상에서 하드웨어 가속 모듈을 추가하고 메모리 접근 등을 최소화함으로써 성능을 향상시키고자 노력하였다. 설계된 하드웨어 IP와 시스템은 여러 단계로 검증하였으며 시스템의 복호화 속도 개선을 도모하였다. QCIF (176x144) 영상을 24MHz의 클록 주파수의 타겟 보드상에서 약 2 frames/sec의 결과를 얻었으며 타겟 보드의 주파수를 증가시키고 FPGA영역의 IP를 ASIC으로 구현하면 더 좋은 성능을 기대할 수 있다.

Abstract

In this paper, we implement the H.264/AVC baseline decoder by hardware-software partitioning under the embedded Linux Kernel 2.4.26 and the FPGA-based target board with ARM926EJ-S core. We design several IPs for the time-demanding blocks, such as motion compensation, deblocking filter, and YUV-to-RGB and they are communicated with the host through the AMBA bus protocol. We also try to minimize the number of memory accesses between IPs and the reference software (JM 11.0) which is ported in the embedded Linux. The proposed IPs and the system have been designed and verified in several stages. The proposed system decodes the QCIF sample video at 2 frame per second when 24MHz of system clock is running and we expect the better performance if the proposed system is designed with ASIC.

Keywords: H.264/AVC, Embedded SoC 설계, ARM

I. 서 론

멀티미디어의 홍수로 말미암아 정지화상, 음성, 동화상을 비롯한 실시간 멀티미디어 유통 및 서비스가 우리 일상생활에 자연스럽게 섞여 들어오고 있다. 이로 인해 네트워크 유동 정보량이 급증하고 처리해야 할 데이터가 늘어난 만큼 멀티미디어 정보의 압축 및 부호화를 통해 효율적으로 저장하고 전송하기 위한 표준안 제정 및 기술개발이 꾸준히 진행되고 있다.

그러나 압축효과를 높이는 기술은 데이터 압축률이 증가한 만큼 부호화 및 복호화 시스템의 복잡도가 증가하였기 때문에 동영상의 수신과 동시에 디코딩, 인코딩과 동시에 다른 어플리케이션과 연동되는 작업을 수행하는 과정이 실시간으로 이루어지기 어렵다. 연속적으로 빠르게 수행되어야 하는 동화상 프레임의 복호화 및 재생으로 연결되는 과정이 실시간으로 이루어지는 일은 해당 응용프로그램이 상품화 되었을 때 시장에서 요구하고 있는 매우 중요한 요인이다. 따라서 동영상 압축 부호화 알고리즘으로 개발되는 어플리케이션은 상품화와 직결되는 대중화, 이동성 강화, 저전력 소비 및 많은 정보량을 빠르게 처리하는 일들이 저렴한 가격대로 공급되어질 필요가 있다. 아울러 제품 개발에서의 원활한 검증과 테스트를 위해 SoC 설계방법을 이용한다는 것

* 정회원, 지에스인스트루먼트 기반연구소
(R&D Software Team, GS Instruments)
** 정회원, 가톨릭대학교 정보통신전자공학부
(Information, Communications, and Electronics Engineering, The Catholic University of Korea)
※ 본 연구는 2007년도 가톨릭대학교 교비연구비의 지원으로 이루어졌음.
접수일자: 2008년3월4일, 수정완료일: 2008년9월1일

은 매우 효과적인 방법이다.

더욱이 실시간 처리가 필요한 동영상 서비스의 경우는 서비스를 제공받는 클라이언트가 '실시간(Real time)'이라고 인지할 정도의 짧은 시간동안 많은 데이터를 처리하여야 하며, 이러한 처리가 필요한 실시간 영상통화나 DMB(Digital Mutimedia Broadcasting) 서비스를 위한 효율적인 부호화 표준이 H.264/AVC이다.

H.264/AVC는 ITU-T와 ISO/IEC가 함께 표준화 과정을 진행시켜 얻어낸 새로운 비디오 압축 표준으로 비디오 응용의 거의 모든 영역(DVD, digital cinema, low bit rate wireless application 등)에 적용이 가능하며, 이전 비디오 부호화 표준들에 비해 상당 수준의 성능의 개선이 이루어진 새로운 표준안이다.

H.264/AVC의 복호기는 부호기에 비하여 그 복잡도가 상대적으로 낮지만 부호기가 영상데이터 서비스를 보급하는데 사용되는데 반해 이러한 서비스 사용자 및 수신을 통한 부호화된 영상의 복호가 훨씬 더 높은 대중성을 가지기 때문에 오히려 더 많은 조건들이 요구되어 진다. 즉, 다수의 사용자 단말기에서 동작해야 하는 복호기는 증가된 멀티미디어 정보를 처리해야 하는데 사용자의 이동성과 관련하여 네트워크 에러내성, 부호화 효율, 전력 소모량 감소 등의 여러 가지 문제들이 시장 경쟁력과 직접 결부되어 있다.

H.264/AVC 복호기는 높은 압축율을 얻기 위해 가변 블록 크기, 1/4 픽셀 단위의 움직임 보상, 인트라 예측 부호화, 4×4 크기의 정수변환 및 CAVLC 등 다양한 압축 기법으로 인해 연산량이 기존의 영상압축기법에 비해 두 배 이상 증가하였다. 낮은 동작 주파수를 갖는 프로세서를 사용하는 모바일 단말기로는 소프트웨어만으로 H.264/AVC 실시간 복호 처리는 불가능하다. 따라서 H.264복호 알고리즘을 하드웨어로 처리하는 하드웨어 가속 모듈이 개발되고 있다. 플랫폼은 이미 동작이 검증된 SW, HW IP, 프로세서와 운영체제로 원하는 시스템의 기본적인 구성을 갖추고 있어 플랫폼 기반 SoC 개발 방법은 플랫폼의 기본 구성에 필요한 IP를 추가하여 SoC 설계의 여러 가지 문제를 극복하기 위한 해결책으로 제시되고 있다[4].

H.264/AVC 복호 알고리즘의 플랫폼 기반 IP 통합설계를 위해서 HW/SW 분할은 컨텍스트 적응형 가변 길이 디코드(CAVLD), IT/IQ, 움직임보상(MC)/인트라 모드 예측(IP), 디블록킹필터(DF) 블록들을 HW IP와 SW IP로 나누어 구현하는 것이 일반적이다[4~6].

플랫폼 기반 H.264 복호기 구현에 관한 기존 연구들은 듀얼 버스를 사용하여 CAVLD, 움직임 벡터 예측(MVP), IT, Ref_Read, Inter/Intra 예측 그리고 디블록킹 필터 HW IP가 동시에 수행하거나[4], CAVLD와 디블록킹 필터, IT/IQ, Inter/Intra 예측 3개의 IP가 동시에 수행하도록 설계를 하고 있으며[5], 또는 SW와 HW를 병렬처리 디코딩하도록 설계한 사례들이 있다[6]. 각 IP 간 연결은 싱글 공유버스를 사용하여 연결하거나, 전용 버스구조를 사용하여 연결하여 H.264 복호기를 구현하였다[5~6]. 기존 연구들은 플랫폼 기반 설계과정에서 병목현상에 대해서 공유 버스의 개수를 늘리고, IP 간 연결을 전용 버스 구조로 연결하는 등의 방법을 통해 최적화하였다. 본 연구는 참조 소프트웨어에서 계산이 많이 반복되는 부분을 HW IP로 대체하여 여러 가지 단계로 검증하고 시스템의 복호화 속도 개선을 도모하였다.

본 논문은 다음과 같이 구성된다. 먼저 Ⅱ장에서는 설계 및 개발 환경에 대해서 설명하였다. Ⅲ장에서는 소프트웨어와 하드웨어의 분할을 설명한 후 제안한 SoC 부호화기의 구조를 설명하였다. Ⅳ장에서는 추가한 하드웨어 검증 방법을 설명하였으며 데모는 Ⅴ장에서 설명하였다. 마지막으로 결론은 Ⅵ장에 기술하였다.

Ⅱ. H.264/AVC 디코더 및 설계 환경

아래의 그림 1은 H.264/AVC 복호기의 블록도를 나타낸다[1~2]. NAL(Network Abstraction Layer)로 받아온 입력은 엔트로피(Entropy) 복호화를 통해 프레임과 매크로블록의 여러 가지 파라미터와 데이터를 얻어내어, 그 값들을 바탕으로 IQ(Inverse Quantization)/IT(Inverse Transform)을 수행한다. MC(Motion Compensation)은 움직임 벡터(Motoin Vector)를 파라미터로 이용하여 이전 프레임을 보간한 뒤 정수 변환의 역과정(Inverse Integer Transform)에서 얻어진 차이 값과 합하여 uF'n을 얻는다. 마지막으로 루프 내에 존재하는 디블록킹 필터(Deblocking Filter)를 통과하여 F'n을 얻는다.

H.264에는 특정한 기능을 지원하는 3 개의 프로파일이 정의 되어 있는데, 각각의 프로파일과 호환되기 위해서 부호기와 복호기에 요구되는 사항들이 정의되어 있다. Baseline 프로파일인 I, P Picture를 사용하는 인트라 코딩 및 인터 코딩, 그리고 컨텍스트 적응형 가변 길이 코드(context-adaptive variable-length codes, CAVLC)을 사용하는 엔트로피 코딩을 지원한다.

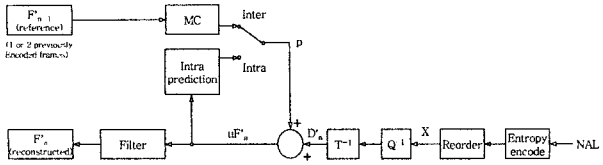

그림 1. H.264/AVC 복호기
Fig. 1. H.264/AVC decoder.

Baseline 프로파일은 B Picture를 사용하지 않고 실시간으로 인코딩과 디코딩이 이루어 져야 하는 화상전화, 화상회의 및 무선 통신에 유용하다.

H.264/AVC 복호기의 설계 환경은 서울대학교 SoC 설계기술 사업단이 ETRI와 공동연구개발, 구축한 SoCBase 1.0을 기반으로 한다[3]. SoCBase 1.0은 단일 ARM 프로세서를 기반으로 하는 SoC 플랫폼으로서 AMBA 2.0 버스 구조를 바탕으로 30여개의 컴포넌트 IP(intellectual property)와 기능 검증, 시스템 라이브러리 등을 포함하여 구성되어 있다.

본 연구에서 타겟 보드(SoCMaster3)에 탑재된 SoCBase 1.0 구조를 바탕으로 AMBA 버스 프로토콜과 FPGA, ARM 프로세서, SoC 컴포넌트 라이브러리 모듈(TFT-LCD, UART, LAN, 등)을 사용하여 구현하고자 하는 H.264/AVC 복호기를 최적으로 구성할 수 있다.

타겟 보드에는 ARM926 프로세서가 내장되어 있으며, 설정신호와 다중 AHB 버스는 보드의 커넥터에 연결되어 있으며 IP들과 구성 블록들은 AMBA 버스 프로토콜을 이용하여 통신이 이루어진다. 타겟 보드로 포팅(porting)할 리눅스는 2.4.20 커널을 기반으로 하고 있으며, 호스트(host) 시스템에 사용할 리눅스는 레드햇 9 배포판을 사용하였다. 타겟 보드는 임베디드 시스템을 구동하기 위해서 부트로더, 커널, 파일 시스템 등을 생성하여 적재하였다. 우리는 이러한 설계 환경을 플랫폼에 맞게 구성하고 이에 따라 H.264/AVC Joint Video Team의 소프트웨어 결과물인 JM 소프트웨어를 참조하였다[7].

III. 제안한 구조

1. 복호기 복잡도와 HW/SW 분할

그림 2는 H.264/AVC 각 모듈의 계산 복잡도를 분석한 것이다. 각 복잡도를 결정하는 기준은 JM Software에서의 P프레임 복호의 함수 수행 시간을 기준으로 하며, 분석 대상이 되는 영상마다 값은 조금씩 달라지지만, 움직임 보상(Motion Compensation)와 디블록킹 필터(Deblocking Filter), 그리고 CAVLD(Variable Length Decoding)에서 가장 많은 시간이 소요된다는 것을 알 수 있다.

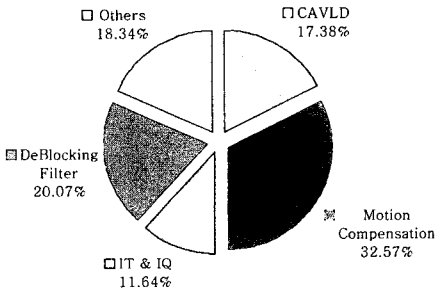

그림 2. 계산 복잡도 분석
Fig. 2. Computational complexity analysis.

2. 시스템 구조

가. 플랫폼의 구성

SoC 복호기 플랫폼은 ARM926, SDRAM, TFT-LCD, UART 등 주변장치로 구분할 수 있다. 하드웨어 구성을 위해서 AMBA 버스, 메모리 컨트롤러 등이 필요하며, 주변장치들의 속도에 따라서 AMBA 버스의 AHB, APB를 통하여 각각의 주변 장치들이 동작한다. AHB와 APB를 이용해 주변장치는 버스 래퍼(bus wrapper)와 레지스터 파일과 같은 모듈을 통해 통신한다.

그림 3과 같이 제안한 H.264/AVC 복호기는 고속 이미지 처리를 위한 하드웨어 가속기(움직임 보상, 디블록킹 필터, YUV2RGB)들은 AHB 버스 프로토콜을 이용하여 연결되어 있으며 UART는 APB를 이용한다. 우리는 그림 4와 같이 기본적으로 4×10 Matrix 2개와 AHB2AHB 브리지(bridge) 1개, AHB2APB 브리지(bridge) 2개와 각종 주변기기들의 버스 래퍼와 컨트롤

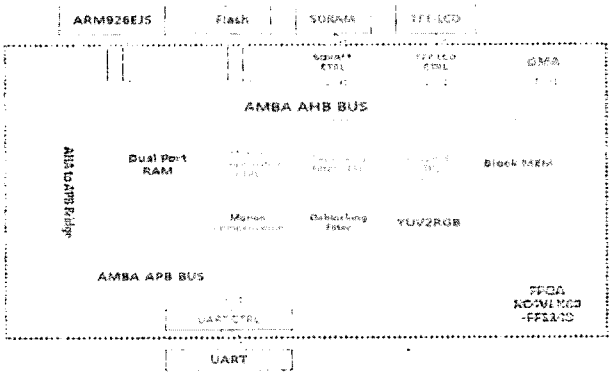

그림 3. 제안한 H.264/AVC 복호기의 구조
Fig. 3. Proposed architecture of H.264/AVC decoder.

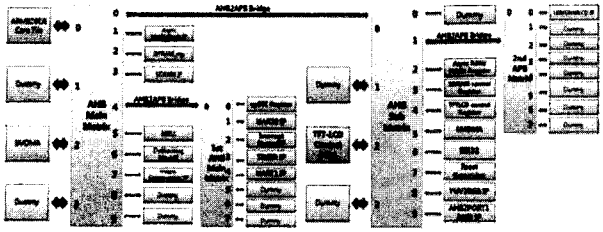

그림 4. SoC 시스템 구성
Fig. 4. SoC system configuration.

러들로 구성하였다.

추가된 하드웨어 모듈은 IP로 설계되고 기본 구조에 병합하여 구현 및 검증하였다. DPRAM(Dual port RAM)은 Xilinx ISE 8.1i 버전의 코어젠을 이용하여 설계하였으며 DPRAM의 A 포트는 AHB 버스와 B 포트는 움직임 보상 모듈과 연결되어 움직임 보상 처리 시에 DPRAM의 B 포트를 이용하여 픽셀 값들을 공급받는다. 디블록킹 필터 모듈은 Main Matrix AHB Slave 7에 연결되어 있으며, YUV2RGB IP 역시 AHB 버스인 Sub Matrix AHB Slave 8에 연결되어 원하는 IP를 매트릭스와 브리지를 통하여 플랫폼을 구성하였다.

나. 메모리 맵

전체 메모리맵은 표 1과 같으며 추가한 모듈인 움직임보상 CTRL, DPRAM, 디블록킹 필터, YUV2RGB 모듈들을 할당하였다.

표 1. SoC 시스템 메모리 맵
Table 1. Memory map of SoC system.

Address	Peripheral Devices
0x71FF_FFFF 0x7100_0000	EBI(EtherNet) DATA
[illegible]	YUV2RGB 모듈
0x7000_5FFF 0x7000_5000	EBI(EtherNet) Reg.
0x7000_4FFF 0x7000_4000	SDMA CTRL Reg.
0x7000_3FFF 0x7000_3000	TFT-LCD CTRL Reg.
0x7000_2FFF 0x7000_2000	SDRAM CTRL Reg.
0x7000_1FFF 0x7000_1000	Async. SRAM CTRL Reg.
[illegible]	움직임 보상 CTRL Reg.
[illegible]	DPRAM(Control)
0x6000_10FF 0x6000_1000	UART
0x5FFF_FFFF 0x5000_0000	SDRAM
0x4FFF_FFFF 0x4000_0000	DPRAM
0x3FFF_FFFF 0x3000_0000	디블록킹 필터 모듈
0x07FF_FFFF 0x0000_0000	FLASH (8MB)

3. IP 설계 및 구현

가. 움직임 보상

두 개의 정수샘플사이의 1/2 픽셀은 6-tap FIR필터와 가중치(1/32, -5/32, 5/8, 5/8, -5/32, 1/32)를 사용하여 보간 된다. 그림 5에서 1/2 픽셀인 b는 E, F, G, H, I, J를 필터링하여 보간된다. 수평과 수직 방향으로 정수 샘플과 인접한 모든 샘플이 계산되면 선형 보간에 의해 1/4 픽셀 위치의 샘플이 생성된다.

그림 6과 같이 움직임 보상 제어 모듈은 움직임 보상 모듈에 필요한 움직임 벡터와 사용하는 픽셀 데이터들을 공급해 주며 움직임 보상 모듈의 메모리 접근량이 많이 때문에 DPRAM을 이용하여 메모리 접근을 최적화 하였다.

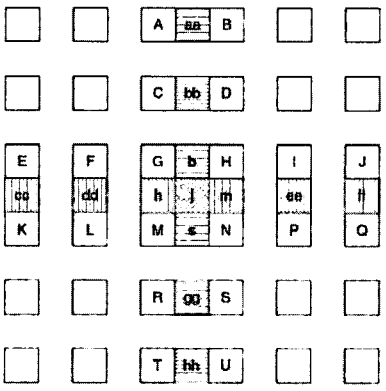

그림 5. 1/2 휘도 샘플의 보간
Fig. 5. Interpolation of half pixels.

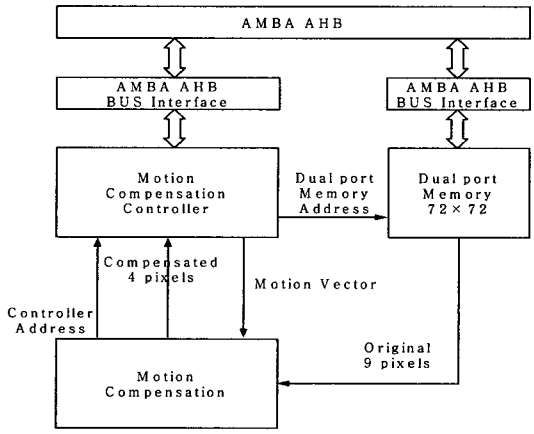

그림 6. 움직임 보상 모듈의 전체구조
Fig. 6. Architecture of motion compensation.

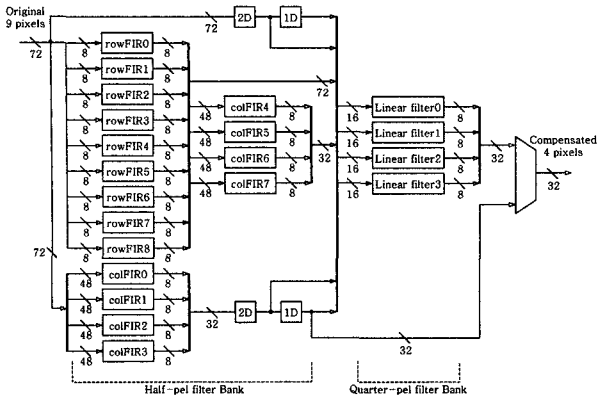

그림 7. 제안한 움직임 보상 모듈의 구조
Fig. 7. Proposed architecture of the motion compensator.

그림 7과 같이 움직임 보상은 두 종류의 6-tap FIR 필터를 사용한다(rowFIR, colFIR). rowFIR은 5단 파이프라인 FIR필터이다[8~9]. rowFIR 모듈은 1 클록 사이클 마다 1개의 정수 샘플이 입력되어 총 개의 정수 샘플을 입력받아 4개의 1/2 샘플을 생성한다. rowFIR은 총 9개가 사용되었다. colFIR은 6개의 정수 샘플이 입력되어 바로 1/2 샘플을 생성한다. colFIR은 8개가 사용되었으며 4개는 정수입력을 받아서 1/2 샘플을 생성하며 나머지 4개는 rowFIR의 출력을 입력받아서 그림 5의 'J'샘플을 생성한다.

움직임 보상 모듈은 크게 1/2 픽셀(half-pel) 필터뱅크와 1/4 픽셀 필터뱅크로 나뉜다. 1/4 샘플을 계산하기 위해서 1/2 샘플이 필요하므로 먼저 1/2 픽셀 필터뱅크에서 1/2 샘플들을 모두 계산하며 다음 클록 사이클에서 1/4 픽셀 샘플을 계산한다. 만일 1/4 샘플을 필요로 하지 않을 경우에는 1/2 샘플만 계산한 뒤 1/4 샘플은 진행하지 않고 출력한다.

나. 디블록킹 필터

그림 8에서와 같이 양자화 파라미터로 구해지는 α, β와 블록정보에 따라 주어지는 Bs로 인해 각 샘플경계에서 디블록킹 필터의 적용여부가 결정된다. 경계주위의 샘플 값의 차분 값이 크면 DCT로 인한 블록킹 현상일 확률이 크므로 강한 필터를 적용하고 그렇지 않으면 약한 필터를 적용하거나 필터를 사용하지 않는다.

그림 9와 같이 디블록킹 필터는 총 8개의 필터를 사용하였고, 2개의 블록경계를 동시에 필터링하게 된다. 같은 블록경계에서는 필터의 파라미터가 동일하게 적용되므로 2개의 파라미터만 입력받아 사용한다.

다. YUV2RGB

SoC Master3의 TFT-LCD는 16비트 RGB의 픽셀 값을 지원하지만 참조 소프트웨어는 YUV형 픽셀로 출력되므로 YUV 픽셀을 16비트 RGB픽셀로 변형하는 모듈을 설계하였다. 이 때 변환식은 실수연산이므로 효율적인 구현을 위해 아래와 같이 정수연산으로 변형하였다.

$$B = (76284 \times (Y-16) + 132252 \times (U-128)) >>16$$
$$G = (76284 \times (Y-16) - 53281 \times (V-128) - 25625 \times (U-128)) >>16$$
$$R = (76284 \times (Y-16) + 104595 \times (V-128)) >>16$$

또한 참조 소프트웨어에서 4:2:0 YUV로 출력하여 U

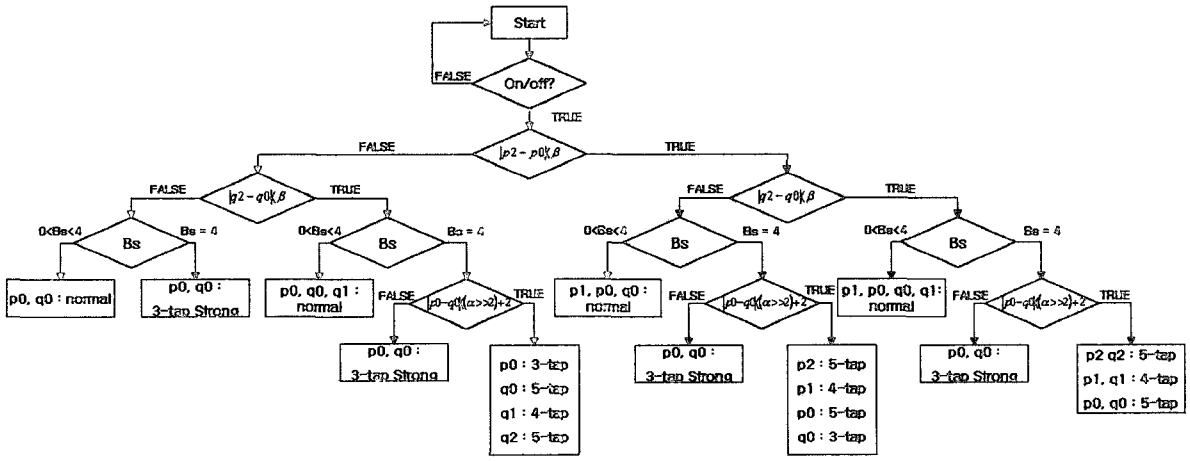

그림 8. 디블록킹 필터의 흐름도
Fig. 8. Deblocking filter flow.

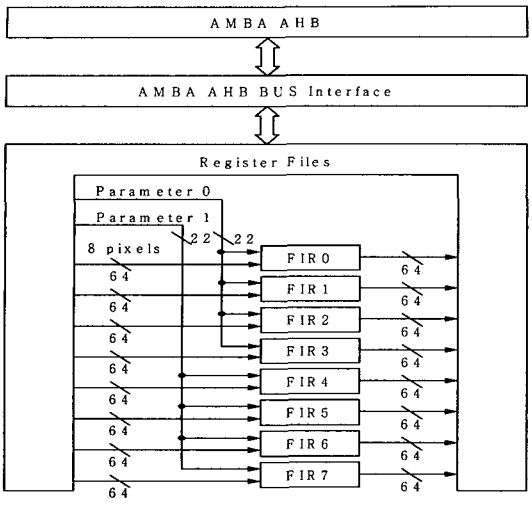

그림 9. 제안한 디블로킹 필터의 구조
Fig. 9. Proposed architecture of the deblocking filter.

와 V는 4번 사용된다. 제안한 YUV2RGB 구조는 Y 4개, U, V 각각 1개씩을 변환의 최소 단위로 처리하여 반복되는 메모리 접근을 최소화하였다. YUV2RGB 모듈은 $Y_{0\sim3}$ 4개와 U, V 각각 1개씩 입력되며 24비트 RGB를 연산한 후 16비트 RGB로 절삭하여 4개의 16비트 $RGB_{0\sim3}$를 출력한다.

Ⅳ. 검증 및 구현

1. 검증

시스템의 검증은 그림 10을 바탕으로 여러 단계(A~E)로 검증하였으며 추가할 IP의 기능을 파형으로 검증(A)후 검증된 HW IP를 AMBA 버스와의 인터페이스 부분을 추가하여 검증(B)하고, 물리주소를 사용하여 검증(C)한 뒤, 가상주소에서 검증(D)한다. 마지막으로 원하는 전체 시스템에서 검증(E)하였다.

A : IP 동작 검증

추가하는 하드웨어 모듈을 검증하기 위해 Modelsim을 이용하여 동작을 검증하였다. 테스트 입력은 하드웨어 모듈에서 필요한 데이터와 파라미터이며 이에 따른 출력이 옳은지를 참조 소프트웨어의 값과 비교하였다.

B : AMBA를 통한 버스 인터페이스 검증

위의 방법으로 검증된 하드웨어 모듈을 시스템에 추가하기 위해서 AMBA 버스 프로토콜에 맞는 버스래퍼와 레지스터 파일이 요구된다. 각각의 하드웨어 모듈의 특성에 알맞게 입력 레지스터의 길이와 개수를 정의하

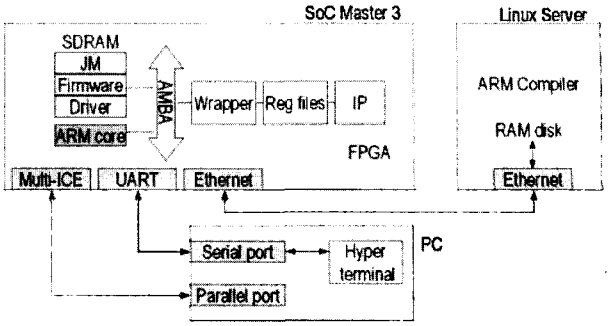

그림 10. 검증 플랫폼
Fig. 10. Verification Platform.

고 하드웨어 모듈과 맞물려서 AMBA 버스 프로토콜과의 인터페이스에 문제가 없는지를 파형으로 분석한다. 이때 입력은 AMBA 버스 프로토콜에 따른 AHB Slave 입력 값이며 이에 따른 AHB Slave 출력 값과 레지스터 파일 내부 신호들이 옳게 하드웨어 모듈로 전달되며 또 하드웨어 모듈에서의 출력역시 옳게 레지스터 파일로 저장이 되어 AMBA 버스 프로토콜에 맞게 출력되는지를 확인한다.

C : 물리주소에서의 검증

추가하는 하드웨어 모듈을 AHB 버스에 연결하여 하드웨어 이미지를 생성 후 FPGA에 다운로드한 뒤, 펌웨어 단계에서 확인한다. 원하는 물리 주소에 하드웨어 모듈이 연결되었으며 데이터의 입출력을 Multi-ICE를 통하여 검증한다.

D : 가상주소에서의 검증(Device Driver 검증)

앞 단계의 검증으로 하드웨어 모듈이 옳게 동작을 하며 AMBA 버스 프로토콜을 통해 원하는 주소(물리주소)에 연결이 되었으므로, 디바이스 드라이버를 생성한 후, 검증소스를 작성하여 추가하는 하드웨어 모듈에 대한 디바이스 드라이버를 통한 접근을 확인한다.

E : 응용 소프트웨어에서의 검증

위의 과정으로 운영체제(Embedded Linux)상에서 추가한 하드웨어 모듈을 사용할 수 있으며 올바르게 동작하므로 응용 프로그램에 포팅한 뒤 검증한다.

2. 구현

H.264/AVC baseline 복호기를 ARM926과 FPGA기반의 플랫폼에서 구현하여 QCIF(176×144) 크기 영상을 약 2fps의 성능으로 복호하였다. 표 2에서는 각각의

표 2. FPGA 리소스 사용분석
Table 2. Analysis of FPGA resource usage.

	Slice F/F	Slice	4 input LUT	Total equivalent gate count
Top	8174	13198	20536	231886
deblocking filter	1410	4382	7236	75416
motion compensation	952	1318	2155	32347
TFTLCD ctrl	1426	1209	1245	20629
yuv2rgb	269	967	1547	18753
SDRAM ctrl	496	929	1661	15149
EBI ctrl	420	760	1313	12444
DMA	425	600	905	9713
matrix, Line buffer등등	92	653	1195	8737

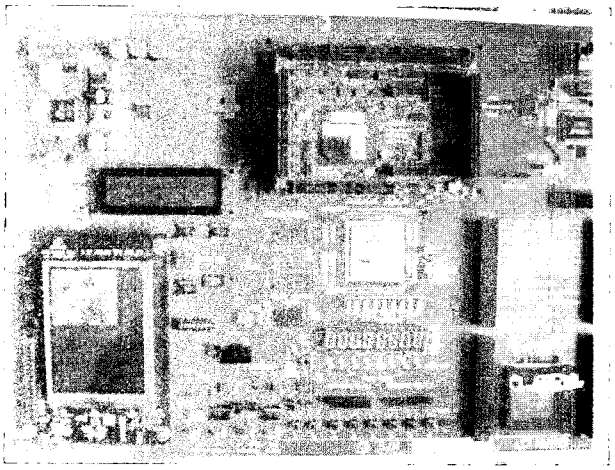

그림 11. 테스트 동영상 시연
Fig. 11. Test Sequence Demonstration.

HW IP가 전체 모듈인 Top에서의 Slice, Slice F/F, 4 input LUT와 이를 gate단위로 환산했을 경우의 표이다. 추가한 HW IP 중 8개의 FIR필터를 병렬로 사용한 디블록킹 필터의 사용량이 가장 많음을 보여준다.

그림 11과 같이 TFT-LCD 모듈의 드라이버를 설치한 후, NAL 소스를 복호화하여 아래와 같이 동작을 검증할 수 있었다. TFT-LCD에서 시험 동영상 중 하나인 수지(suzie) 영상이 출력되고 있다.

VI. 결 론

임베디드용 Linux와 ARM926EJ-S 코어를 탑재한 FPGA(XC4VLX60)기반의 개발환경에서 SW/HW 분할을 통해 수정된 JM 소프트웨어는 GNU를 이용해 컴파일되었으며, 설계된 하드웨어 모듈은 움직임 보상 모듈, 디블록킹 필터 모듈, YUV2RGB 모듈이다. 설계된 각 하드웨어 모듈은 소프트웨어에서 복잡도가 높은 블록을 선정하여 효율적으로 계산량을 감소시켰으며 하드웨어 모듈은 소프트웨어와 함께 맞물려 동작하기 위해 AMBA 버스 프로토콜을 준수한다. 이러한 과정을 거쳐 H.264/AVC 복호기를 구현하였다.

본 연구는 참조 소프트웨어에서 계산이 많이 반복되는 부분을 HW IP로 대체하여 여러 가지 단계로 검증하고 시스템의 복호화 속도 개선을 도모하였다. 본 연구에서 사용한 타겟보드는 검증을 주목표로 25MHz 클록 주파수를 사용한 ARM926과 FPGA에 QCIF (176×144) 영상을 약 2 frames/sec의 결과를 얻었다. 추후에 타겟 보드의 주파수 증가 및 참조 소프트웨어를 최적화하고 FPGA영역의 IP들을 ASIC으로 구현하면 실시간처리를 기대할 수 있다.

감사의 글

저자들은 본 연구를 위하여 설계 환경을 제공하여 준 IDEC(IC Design Education Center)에 감사드린다.

참 고 문 헌

[1] ITU-T Recommendation H.264: Advanced video coding, ITU, 2004.
[2] Iain E. G. Richardson, "H.264 and MPEG-4 Video Compression Video Coding for Next-generation multimedia," John Willey & Sons, 2003.
[3] 채수익, 박상규, "SoCBase 플랫폼 아키텍쳐 설명서," 서울대학교 SoC 설계기술사업단, 2004.
[4] Hae-Yong Kang, Kyung-Ah Jeong, Jung-Yang Bae, Young-Su Lee and Seung-Ho Lee. "MPEG4 AVC/H.264 Decoder with Scalable Bus Architecture and Dual Memory Controller," IEEE International Symposium on Circuits and Systems, vol.2, pp.145-148, 2004.
[5] Seongmo Park, Hanjin Cho, Heebum Jung and Dukdong Lee, "An Implemented of H.264 Video Decoder using Hardware and Software," Proceedings of the IEEE 2005 Custom Integrated Circuits Conference, pp.271-275. 2005.
[6] Yang Kun, Zhang Chun, Du Guoze, Xie Jiangxiang and Wang Zhihua, "A Hardware-Software Co-design for H.264/AVC Decoder," IEEE Asian Solid-State Circuits Conference, pp.119-122, 2006.
[7] ITU, H.264/AVC Reference Software, http://iphome.hhi.de/suehring/tml, ver.JM11.0
[8] Yang Song, Zhenyu Liu, Goto Satoshi and Ikenaga Takeshi, "A VLSI architecture for

motion compensation interpolation in H.264/AVC," IEEE 6th International Conference On ASIC(ASICON 2005), vol.1, pp.279-282, 2005.
[9] Dajiang Zhou and Peilin Liu, "A Hardware-Efficient Dual-Standard VLSI Architecture for MC Interpolation in AVS and H.264," IEEE International Symposium on Circuits and Systems, pp.2910-2913, 2007.

저 자 소 개

김 진 욱(정회원)
2005년 가톨릭대학교 정보통신공학과 졸업.
2008년 가톨릭대학교 컴퓨터공학과 석사 졸업.
2008년~현재 GSI 기반연구소 연구원.
<주관심분야 : SoC, VLSI 설계, 신호처리, Motion estimation>

박 태 근(정회원)-교신저자
1985년 연세대학교 전자공학과 졸업.
1988년 Syracuse Univ. Computer 공학석사 졸업.
1993년 Syracuse Univ. Computer 공학박사 졸업.
1991년~1993년 Coherent Research Inc. VLSI 설계 엔지니어.
1994년~1998년 현대전자 System IC 연구소 책임연구원.
1998년~현재 가톨릭대학교 정보통신전자공학부 교수.
<주관심분야 : VLSI 설계, CAD, 병렬처리>