ISSN : 1226-7244 (Print)
ISSN : 2288-243X (Online)
논문번호 19-03-27

j.inst.Korean.electr.electron.eng.Vol.23,No.3,925~933,September 2019
http://dx.doi.org/10.7471/ikeee.2019.23.3.925

블록 기반 압축 이미지 및 비디오를 위한 디블로킹 필터의 SoC 구현

SoC Implementation of Deblocking Filter for Block-based Compressed Images and Videos

서 광 석*, 이 주 흥**★

Gwang-Seok Seo*, Joo-Heung Lee**★

Abstract

In this paper, we implement ZYNQ SoC-based post-processing system that utilizes partial reconfiguration to remove blocking artifacts generated by compression algorithm. Hardware implementation of the deblocking filter in a Field Programmable Gate Array (FPGA) provides high computational capability and can be partially reconfigured to process 1080p images in real time. Partially reconfigurable areas in FPGA can be utilized to use hardware more efficiently in highly resource-constrained embedded systems. Experimental results of the proposed system show improvement of visual quality both objectively and subjectively with 0.6dB higher PSNR after deblocking filtering process. The measured power consumption of the deblocking filter during run-time is 68.33mW.

요 약

본 논문에서는 Zynq Soc Platform의 부분 재구성 기능을 사용하여 영상 압축으로 생성된 blocking artifacts를 제거하는 후처리 시스템을 설계한다. 높은 연산량을 제공하고 실시간으로 1080p 영상을 처리하도록 부분 재구성이 가능한 FPGA (Field Programmable Gate Array) 영역에 디블로킹 필터를 구현한다. 또한 부분적으로 재구성 가능한 영역을 활용하여 제한된 환경의 임베디드 시스템에서 하드웨어 리소스를 보다 효율적으로 사용할 수 있다. 제안된 시스템의 실험결과는 디블로킹 필터처리 후 약 0.6dB의 PSNR 향상을 보여준다. Zynq SoC에서 구현된 필터가 동작할 때 68.33mW의 전력을 소모한다.

Key words : FPGA, Deblocking Filter, Zynq SoC, Partial Reconfiguration, High Level Synthesis

Ⅰ. 서론

최근 고해상도 멀티미디어 장치의 발전으로 인해 많은 다양한 영상 압축 기술이 활용되고 있다. 블록기반 압축 알고리즘은 이미지 및 동영상을 처리하는데 널리 사용된다. 예를 들어, JPEG에서 HEVC (High Efficiency Video Coding)에 이르는 다양한 국제표준 압축 알고리즘들이 있다. 표준 압축 알고리즘은 주파수 영역에서의 양자화 과정을 통해 블록 사이에 불연속적인 경계를 생성하는데 이를 blocking artifacts라고 한다. 일반적으로 압축 비율이 높을수록 blocking artifacts의 양은 더 커지게

* Associate Research Engineer, Satrec Initiative
** Professor, Dept. of Electronic and Electrical Engineering, Hongik University
★ Corresponding author
E-mail : joolee@hongik.ac.kr Tel : +82-44-860-2544
Manuscript received Aug. 26, 2019; revised Sep. 19, 2019; accepted Sep. 26, 2019.

된다. Deblocking filter는 블록 간의 불연속을 제거하여 화질 향상에 기여하게 된다[1].

많은 양의 데이터를 필요로 하는 영상 처리의 경우 CPU를 사용하는 소프트웨어는 많은 시간을 소비해야 하므로 실시간 처리에 어려움이 있다. 효과적인 계산을 수행하기 위해, FPGA 및 고속 파이프라이닝과 같은 하드웨어 환경에서 애플리케이션에 최적화된 병렬처리구조를 활용해 실시간으로 영상정보를 처리 할 수 있다. FPGA 구현은 높은 유연성, 낮은 비용 및 낮은 전력소비의 장점을 제공한다[2].

본 논문에서는 계산 복잡도가 높은 deblocking filter를 Xilinx사의 Vivado HLS(High Level Synthesis) Tool을 사용하여 Zynq-7020 SoC 환경에서 구현한다. OpenCV 함수호출은 입출력 영상 값을 얻는데 사용되며, 획득된 영상정보를 사용하여 deblocking 연산이 수행된다. HLS에서 OpenCV 라이브러리로 구현된 deblocking filter는 Programmable Logic의 IP에 통합되어 실시간 비디오 스트림을 처리한다.

본 논문의 구성은 다음과 같다. 2장에서는 Zynq SoC Platform을 소개하고, 3장은 HLS 설계 흐름을 서술한다. 4장에서는 논문에서 구현된 알고리즘을 기술하고, 5장에서는 제안된 시스템의 성능을 평가한다. 6장에서는 연구 결과를 요약한다.

II. 본론

1. Zynq-7000 SoC Platform

SoC 개발 Platform이 제공되는 Zynq-7000은 소프트웨어 프로그램 실행을 위한 ARM Cortex A-9을 포함하는 Processing System (PS)과 하드웨어 가속기 설계에 효과적으로 사용될 수 있는 Programmable Logic(PL)이 결합된 이기종 시스템이다. PS는 APU (Application Processor Unit), 캐시, 온칩 메모리, 외부 메모리 인터페이스, GPIO, 기가비트 이더넷, USB, I2C, CAN 및 SPI와 같은 다양한 I/O 주변장치로 구성된다. PL은 다양한 로직을 구현할 수 있는 FPGA로 구성되어 있고, 12-Bit Analog-to-Digital Converters, DSP Blocks, Programmable I/O Block, JTAG, CLB(Configuration Logic Block) 및 Block RAM과 같은 다양한 하드웨어 리소스를 가지고 있다[3]. PL과 PS 간의 상호 작용은 Advanced Microcontroller Bus Architecture(AMBA)에서 지원한다[4]. AMBA 버스 인터페이스에는 AXI(Advanced eXtensible Interface), AHB(Advanced High Performance Bus) 및 APB(Advanced Peripheral Bus)가 포함된다. Zynq SoC는 32 비트 General-Purpose Port(GP) 4개, 32/64 비트 High-Performance Port (HP) 4개, Accelerator Coherency Port(ACP)를 보유하고 있다. ACP는 L1/L2 캐쉬의 CPU 데이터와 PL에 구현된 하드웨어 가속기 간의 캐시 일관성을 위해 PL을 Snoop Control Unit에 연결한다. GP는 각각 2 개의 32 비트 마스터 또는 슬레이브 인터페이스를 가지며 PL과 PS 사이의 모든 AXI 인터페이스에 대한 비동기 클록 주파수 도메인을 지원한다. 마지막으로, HP는 32 비트, 64 비트 데이터 마스터 인터페이스를 지원하는 PL의 고성능 대역폭을 제공하는 포트이며 읽기 및 쓰기가 가능한 1KB 데이터 FIFO를 사용하여 외부 DDR SDRAM 메모리에 데이터를 빠르게 전송할 수 있다[5].

2. Partial Reconfiguration

Xilinx에서 제공하는 Partial Reconfiguration(PR)은 FPGA를 사용함에 있어서 더욱 큰 유연성을 제공하고 있다. FPGA의 Partial Reconfiguration Regions (PRRs)에 사용자가 원하는 새로운 기능을 다른 회로들의 동작을 멈추지 않고 실시간으로 재프로그래밍 할 수 있다. PRRs은 재프로그래밍이 가능하도록 사전에 사용자에 의해 정의된 FPGA의 특정 영역이며, Partial Reconfiguration Modules(PRMs)을 구현하는데 사용된다. 일반적으로 부분 재구성에는 다음과 같은 3가지 이점이 있다. 첫째, 부분재구성은 사용가능한 하드웨어 자원의 일부를 동적으로 다중화 하여 필요한 회로의 설계크기를 줄일 수 있다. 필요에 따라 PRMs을 실시간 로드할 수 있으므로 유휴로직의 양이 줄어들어 추가 공간을 절약할 수 있다. 둘째, PR은 변경 중에 시스템을 종료하지 않고 실시간으로 새로운 기능을 동적으로 삽입할 수 있다. 상호배타적인 기능들의 경우 필요에 따라 동적 재구성이 가능하므로 다양한 기능들을 지원하기 위한 시스템 재설계의 필요성이 감소한다. 마지막으로, 동적으로 다중화한 하드웨어 자원을 이용하여 필요에 따라 다양한 성능의 설계로 전환함으로써 사용자의 변화하는 우선순위(전력소모/실시간 연산량/하드웨어 리소스)를 만족시킬 수 있다[6-7]. PR을 사용하기 위해 Zynq 디바

이스는 비트스트림 다운로드를 위한 2개의 인터페이스를 지원한다. Internal Configuration Access Port(ICAP)를 통해 PL에서 스스로 재구성하거나 Processor Configuration Access Port (PCAP)를 통해 PS가 PL 영역을 재구성할 수 있다. ICAP을 통한 전송 속도는 최대 400MB/s며 100MHz 클럭 주파수에서 동작한다. 본 논문에서 필터의 비트스트림을 부분 재구성하는 작업은 133MHz 클럭으로 최대 145MB/s의 다운로드 처리량을 지원할 수 있는 32bit PCAP 인터페이스를 통해 수행된다. Fig. 1은 장치 구성 흐름도를 보여준다. 시스템이 처음 부팅할 때 외부 SD 카드에 저장한 전체 비트스트림을 PL 영역의 static logic에 다운로드 한다. 그 다음 SD 카드에 저장된 부분 비트스트림을 PS의 메모리 컨트롤러를 통하여 외부 DDR SDRAM 메모리로 전송한다. 부분 비트스트림은 필터의 기능이 필요할 때 PL 영역에 실시간 재구성된다[8].

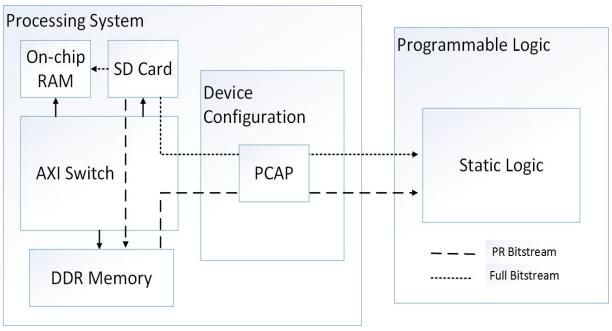


Fig. 1. Device configuration flow.
그림 1. 장치 구성 흐름도

III. High Level Synthesis

C, C++ 및 SystemC와 같은 high level languages에서 HLS(High Level Synthesis)의 사용은 효율적 설계 및 검증을 수행하는데 장점을 제공한다[9]. Vivado HLS에서 하드웨어 설계 Flow는 Fig. 2와 같다. C언어에서 Vivado HLS는 다양한 directives와 pragma를 사용하여 설계된 하드웨어를 최적화하고 병렬처리 및 디자인 인터페이스와 같은 기능을 허용한다[10]. HLS에서 제공하는 C Simulation을 사용하여 functional level에서 시스템을 검증할 수 있다. Functionally simulated design의 합성을 통해 변환된 코드는 C/RTL Co-Simulation을 사용하여 waveform 검증을 수행할 수 있다. 그 후 검증된 디자인은 IP 패키징을 통해 Vivado Design Suite에서 사용할 수 있다[11].

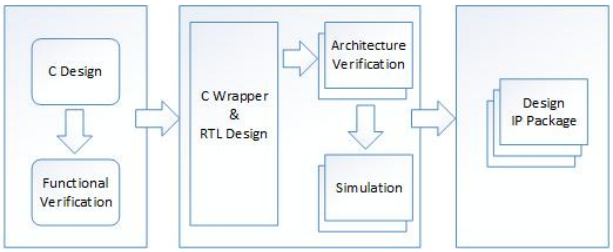


Fig. 2. Vivado HLS flow.
그림 2. Vivado HLS 흐름도

IV. Case Study : deblocking filter

본 논문에서 deblocking filter는 HLS를 사용하여 고정 소수점으로 구현한다. 구현하는 deblocking filter에는 평탄모드와 일반모드가 있다[12]. 필터는 Fig. 3에서 나타내는 블록의 경계에서 압축에 의해 생성된 blocking artifacts를 제거한다. Deblocking filter는 DDR 메모리로부터 수신된 입력영상의 특성을 고려하여 모드를 선택하고, 필터링 후 영상의 결과를 DDR 메모리에 저장한다. 필터의 목표는 블록 사이의 경계를 부드럽게 하는 것이다. 이를 위해 모드 결정, 평탄모드, 일반모드의 세 가지 기능을 구현한다.

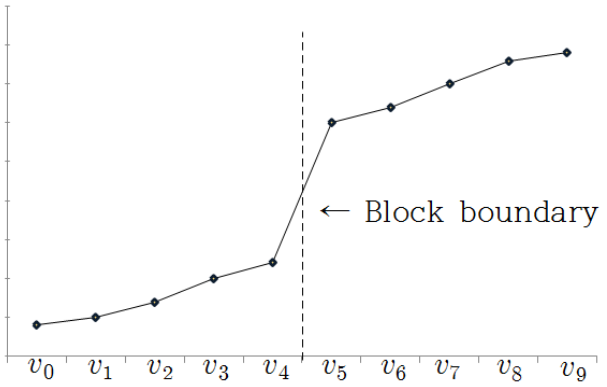


Fig. 3. Block boundary.
그림 3. 블록 경계

첫째, 모드 결정은 블록경계의 평탄도를 사용한다. 평탄도 측정은 다음과 같이 계산한다.

$$F(v) = \sum_{i=0}^{8} \phi(v_i - v_{i+1}) \quad (1)$$

$\phi(X)$는 X의 값이 T_1보다 크면 1의 값을 가지며 그렇지 않으면 0의 값을 갖는다. $F(v)$의 값이 T_2보다 크면 경계가 평탄하지 않다는 것을 의미하고 경계와 경계의 주변 값을 평평하게 하는 평탄모드가 동작한다. 다른 경우에는 오직 경계에서만 평탄화를 수행하는 일반모드에서 작동한다. 이 논문에서 T_1은 2이고, T_2는 6이 사용되었다.

둘째, 일반모드는 복잡한 영역인 블록의 경계에서만 작동한다. 일반모드에서는 다음과 같은 연산이 수행된다.

$$v_4^{'} = v_4 - d,\ v_5^{'} = v_5 - d \tag{2}$$

$$d = \begin{cases} 0 & , \frac{c_2}{c_3} \times (a_{3.1}^{'} - a_{3.1}) \le 0 \\ \frac{c_2}{c_3} \times (a_{3.1}^{'} - a_{3.1}), & otherwise \\ \frac{(v_4 - v_5)}{2} & , \frac{c_2}{c_3} \times (a_{3.1}^{'} - a_{3.1}) \ge \frac{(v_4 - v_5)}{2} \end{cases} \tag{3}$$

$$a_{3.1}^{'} = \begin{cases} a_{3.1} \bullet \frac{MIN(|a_{3.0}|,|a_{3.1}|,|a_{3.2}|)}{|a_{3.1}|}, & |a_{3.1}| \neq 0 \\ 0, & |a_{3.1}| = 0 \end{cases} \tag{4}$$

(2)의 식에서 사용되는 d의 값을 결정하기 위해 블록의 경계를 중심으로 S_0, S_1, S_2 3개의 공간영역으로 분할한다. S_0의 영역은 v_1에서 v_4, S_1은 v_3에서 v_6, S_2는 v_5에서 v_8으로 구성된다. 공간영역의 픽셀 정보를 4-point Discrete Cosine Transform(DCT)를 사용하여 주파수 영역으로 변환한다. S_1의 영역을 4-point DCT를 사용하여 변환된 주파수 영역에서의 값은 $a_{0.1}$, $a_{1.1}$, $a_{2.1}$, $a_{3.1}$로 정의한다. 여기서 S_1영역의 고주파 성분인 $a_{3.1}$은 공간영역에서 blocking artifacts를 생성하는데 직접적인 관련이 있다. 그래서 $a_{3.1}$의 성분을 0과 1사이의 값으로 스케일링하여 고주파 성분의 크기를 줄여 블록간의 불연속을 완화한다. d를 구하기 위해 연산에 c_1= 2, c_2= 5, c_3= 8이 사용된다. 고주파 성분을 이용하여 구한 d 값을 활용해 경계의 픽셀 값에 더하거나 빼서 경계를 부드럽게 할 수 있다.

마지막으로, 평탄모드는 블록의 내부 및 경계에서 필터링을 수행한다. 평탄모드는 다음과 같이 수행한다.

$$v_n^{'} = \frac{1}{16} \sum_{k=-4}^{4} b_k \times p_{n+k},\ 1 \le n \le 8 \tag{5}$$

$$p_{n+k} = \begin{cases} v_0, \text{if } |v_1 - v_0| < QP\ when n+k < 1 \\ v_1, \text{if } |v_1 - v_0| \ge QP\ when n+k < 1 \\ v_n, \qquad\qquad when 1 \le n+k \le 8 \\ v_8, \text{if } |v_8 - v_9| < QP\ when\ n+k > 8 \\ v_9, \text{if } |v_1 - v_0| < QP\ when\ n+k > 8 \end{cases} \tag{6}$$

$$b_k = [1,1,2,2,4,2,2,1,1],\ -4 \le k \le 4 \tag{7}$$

여기서 $v_n^{'}$은 후처리된 픽셀 값이다. 경계에서 픽셀 값을 부드럽게 늘리거나 줄임으로써 blocking artifacts를 제거한다. 그러나 영상의 엣지가 평활화 되는 것을 방지하기 위해, v의 최댓값과 최솟값의 차이가 2QP보다 큰 경우, 영상의 엣지로 간주하여 필터링 되지 않는다. 여기서, QP(Quantization Parameter)는 블록의 양자화 파라미터를 사용한다.

V. Experimental Results

그림 4는 RTL 시뮬레이션 결과를 파형으로 검증한 결과이다. HLS에서 C/RTL co-simulation 기능을 활용하여 시뮬레이션을 수행하였고, 각 함수별 입출력을 C로 구현한 결과와 비교 평가하여 검증하였다.

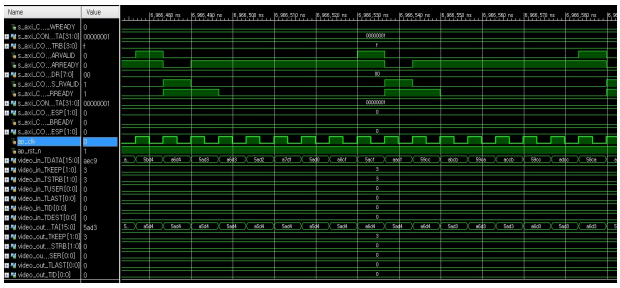
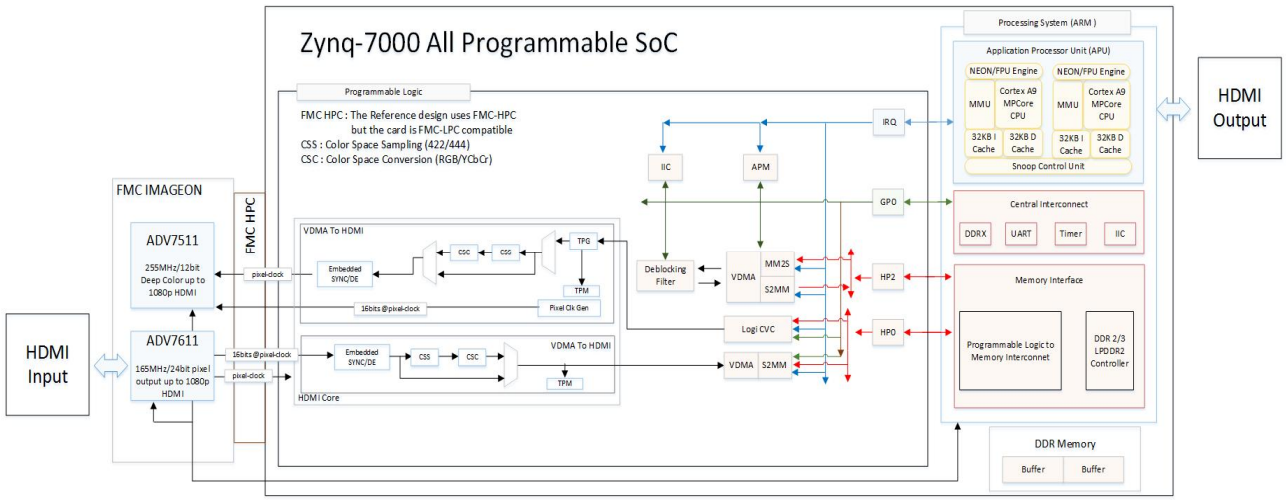


Fig. 4. RTL simulation results.
그림 4. RTL의 시뮬레이션 결과

본 논문에서는 HDMI input/output을 지원하는 ZC702 evaluation board 및 FMC 모듈을 이용하여 제안된 재구성 시스템을 구현한다. Fig. 5는 디블로킹 필터가 구현된 시스템의 원형을 도시한다. 작동중인 ZC702 evaluation board는 PC 환경의 UART 터미널에 의해 제어되며 Trace-32를 통해 실시간 동작 검증을 수행한다. 1920x1080 해상도의 비디오 영상은 FMC IMAGEON의 HDMI 인터페이스를 통해 입력되고 ZC702 evaluation board의 HDMI 인터페이스를 통해 후처리된 최종 영상 결과가 출력된다. 입력 영상은 Color Space Sampling 및



Processor	ARM Cortex-A9
OS	Peta-Linux
Technology	Artix-7 FPGA
Throughput	Full-HD (1080p) @ 35fps
Frequency	150MHz

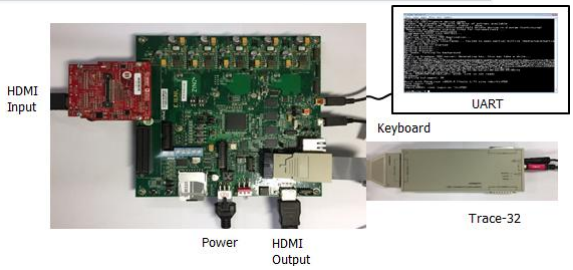


Fig. 5. Prototype of the system.
그림 5. 시스템 프로토타입

Color Space Conversion을 거쳐 AXI Video DMA (VDMA)를 통해 Zynq DDR 메모리에 저장된다. VDMA Core는 시스템 메모리와 AXI4-Stream 기반 비디오 IP Core 간에 고속 데이터 이동을 제공한다. AXI VDMA Core는 AXI Interconnect를 사용하여 고속 데이터 이동이 가능한 64비트 HP Interface에 연결된다. Processing Pipeline은 HP2 읽기/쓰기포트에 연결되며 HLS 영상 필터와 VDMA로 구성된다. HLS Image Filter가 활성화되어 있지 않으면 입력영상은 디스플레이로 우회하게 된다. Image Filter가 활성화 될 경우 VDMA는 메모리에서 비디오 프레임을 읽고 이를 HLS 영상 필터로 보낸다. 비디오 프레임이 처리 된 후, VDMA write channel은 결과를 다시 메모리에 기록한다. HLS 영상 필터는 이 디자인에서 실시간 재구성되는 하드웨어 블록이다. 시스템은 PR을 사용하여 PCAP 인터페이스를 통해 런타임에 동적으로 필터 모듈을 불러온다. 출력 비디오 스트림은 HDMI를 통해 모니터로 전송된다.

Deblocking filter의 비트스트림은 SD 카드에 저장하고 PR을 사용하여 작동한다. 부분 비트스트림은 전체 비트스트림보다 bit파일 크기가 작으므로 메모리 용량과 구성 시간을 줄일 수 있다. 전체 비트스트림의 크기는 4,045,564 Bytes이고 부분 비트스트림의 크기는 753,848 Bytes이기 때문에 약 5.4배이며 로직 구성 시간은 전체 비트스트림은 83ms, 부분 비트스트림은 22ms를 소모한다. 로직 구성 시간은 약 3.7 배 빠른 것을 확인할 수 있다.

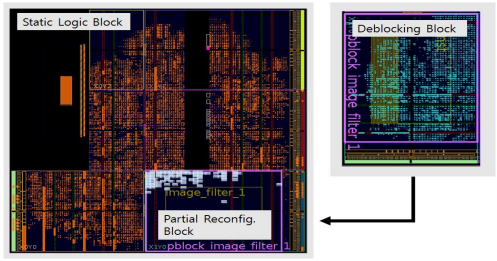


Fig. 6. Floorplanning.
그림 6. 시스템 플로어플래닝

Fig. 6은 SoC 내부 PL 영역에서의 Static Logic 및 deblocking filter 블록을 도시한다. 시스템이 처음 실행될 때 static logic 전체 비트스트림이 PL 영역에 다운로드 된다. PR 영역에 필터를 구현하여 작동시키며, 필터가 더 이상 필요하지 않으면 PR 영역에 blank 비트스트림을 다운로드하여 전력 소비를 줄이는 데 활용한다.

Table 1. Resources required for Partial Reconfiguration Module
표 1. 부분재구성 모듈 자원

Resource	PRR	PRM (Deblocking)	
	Available	Utilization	
LUT	12000	4666	38.88%
SLICE	2200	1396	63.45%
RAMB18	60	17	28.33%
DSPs	40	0	0%

표 1은 PRM인 deblocking filter 구현에 필요한 FPGA 하드웨어 자원을 나타낸다. 플로어 플래닝 시 30% 이상 재구성 가능 하드웨어 자원들의 여분이 남을 수 있도록 설계하여 P&R 과정이 쉽게 이루어 질 수 있도록 하였다.

Motion JPEG은 IP Camera, 디지털 카메라, HDTV 미디어 플레이어 등 다양한 응용 분야에서 영상 압축 방식으로 사용되고 있다. 본 논문에서 구현한 후처리 필터의 성능을 측정하기 위해 약 25.5:1의 비율로 압축률을 높여 테스트 입력 영상 Elephants Dream에 blocking artifacts를 생성한다. 후처리 성능의 객관적 비교를 위하여 PSNR(Peak Signal-to-Noise Ratio)을 사용한다. 본 논문에서 구현한 필터를 적용하였을 경우 평균적으로 PSNR이 0.6dB 개선되는 것을 볼 수 있다. Fig. 7의 (a), (c)는 압축된 영상이며, (b), (d)는 deblocking filter를 사용하여 후처리한 결과이다. 주관적 평가를 통해서도 시각적 화질이 개선되었음을 확인할 수 있다.

Fig. 8은 #637 프레임의 135번째 행의 처리 전후의 화소 값을 나타낸다. (a)는 행의 전체 화소 값을 보여준다. (b)는 600번째에서 660번째 열의 영역이고 압축으로 인해 blocking artifacts가 생긴 경계를 나타내며 deblocking 처리 후 픽셀의 기울기가 완만해지는 것을 볼 수 있다. (c)는 980번째에서 1040번째 열의 영역을 확대한 영상이고 이 영역은 영상의 실제 엣지로 deblocking 처리를 하지 않고 입력이 그대로 출력되어 영상의 흐려짐을 방지하고 있다.

본 논문에서는 시스템의 성능을 평가하기 위해

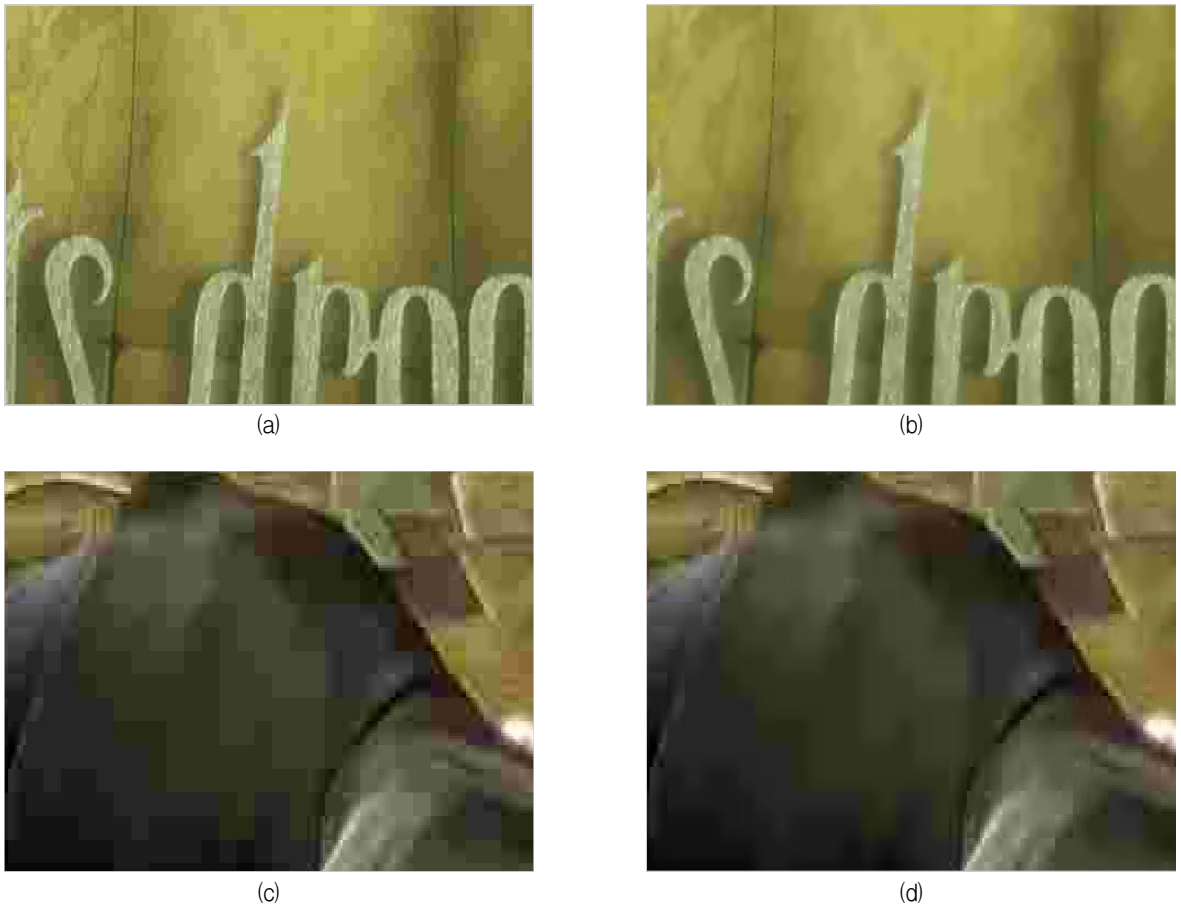
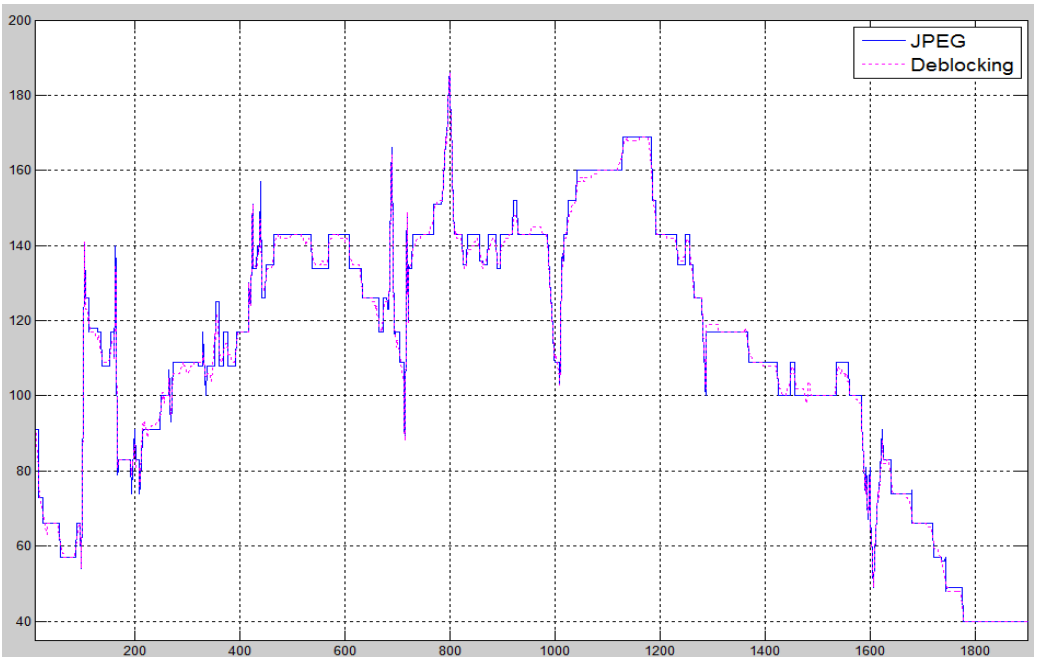
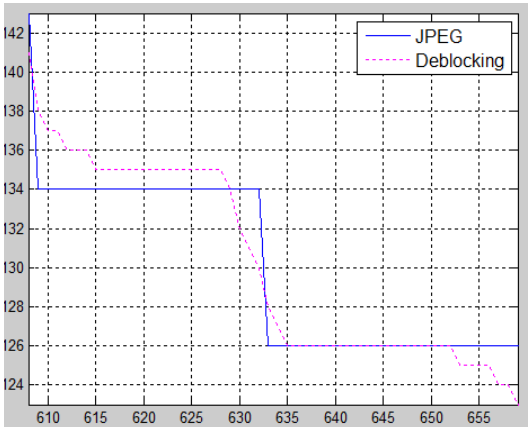


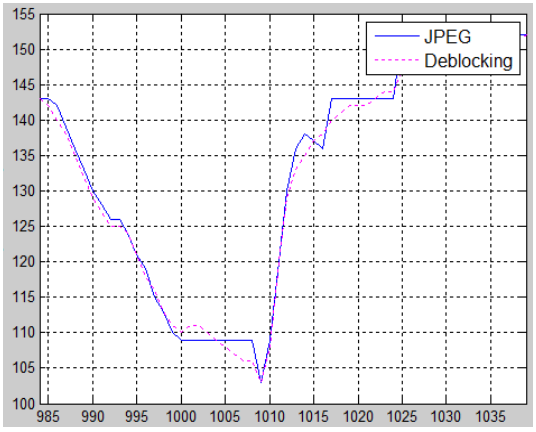
Fig. 7. Deblocking results of image sequence(Elephants Dream).
그림 7. 이미지 시퀀스에 대한 디블로킹 후처리 결과



(a)



(b)



(c)

Fig. 8. Comparison of pixel values before and after processing.
그림 8. 처리 전 후 화소 값 비교

AXI Performance Monitor(APM)을 사용한다. APM Core는 performance를 측정하고 연결된 AXI 인터페이스 처리량의 실시간 성능을 기록한다[13]. 본 디자인은 HP0 및 HP2 포트를 사용하는데, HP0은 FMC를 통해 입력된 영상을 외부 DDR메모리에 기록하기 위해 사용되고, HP2는 메모리의 입력 영상을 읽고, 필터 처리를 한 후 다시 메모리에 저장하는데 사용된다. 본 실험에서 측정된 HP0 / HP2의 영상 데이터 처리율은 각각 246.64Mb/s의 성능을 보인다.

Table. 2. Performance Comparison.
표 2. 성능 비교

Technology	Frequency	Throughput	Power
Zynq PS (ARM-Cortex A9 - SW only)	1GHz	0.59 fps [Resolution : 1920x1080]	89.63mW
Zynq PL (FPGA - SW/HW codesign)	150MHz	35 fps [Resolution : 1920x1080]	68.33mW

ZC702 Board는 전원 레귤레이터와 PMBus 호환 시스템 컨트롤러를 사용하여 보드에 전원을 공급한다[14]. Zynq7000은 PMBus Protocol을 사용하여 Zynq Processing System의 I2C 인터페이스에 연결된 버스를 통해 TI사의 디지털 파워 컨트롤러와 통신한다. 레귤레이터의 출력 전압은 파워 컨트롤러에 의해 모니터링 및 제어된다. 파워 컨트롤러는 deblocking filter가 처리되는 동안 내부 전력 소모를 측정하는데 이용된다.

PS영역에 리눅스 운영체제를 포팅하고, 1 GHz에서 동작하는 Cortex-A9 CPU에서 S/W를 이용하여 디블록킹 필터를 구현하였다. SW 구현 성능의 경우, 1920x1080 고해상도 영상의 실시간 연산이 불가능한 0.59 fps(frames per second)의 낮은 처리량을 보여주며, 이때 89.63mW의 전력을 소모하였다. 제한된 임베디드 환경 하에서 실시간 화질 향상 기능을 구현하기 위하여 SW/HW codesign 방법을 활용하였다. Zynq SoC의 PL 영역에 디블록킹 필터를 구현하였을 경우, 150 MHz의 동작 주파수에서 35 fps의 실시간 처리가 가능하였으며, 68.33mW의 전력을 소모함을 확인하였다. 임베디드 환경에서 실시간 연산처리량이 높은 deblocking filter를 부분 재구성이 가능한 하드웨어에 맵핑하여 구현함으로서 낮은 전력에서 더 높은 성능 향상을 이룰 수 있었다. 성능 측정 및 비교 결과는 표 2에 표시하였다.

Ⅵ. Conclusion

본 논문에서는 Zynq SoC 환경에서 실시간 화질 향상을 위한 후처리용 디블록킹 필터를 PL 영역에 구현하여 평균적으로 약 0.6dB의 PSNR이 향상되었으며, 주관적으로도 뚜렷한 화질 개선을 보여주었다. 소프트웨어로 구현하였을 경우, 1920x1080 고해상도 영상의 실시간 처리가 불가능하였으나, 재구성 가능 하드웨어를 사용하여 지원을 할 경우 실시간 최대 35 fps를 처리함으로서 필터의 성능과 전력소모에서 더욱 향상된 결과를 얻을 수 있었다. 또한 동적 부분 재구성 기능을 활용하여 후처리용 필터를 사용하지 않을 경우 블랭크 비트스트림을 다운로드함으로서 전력소모를 줄일 수 있다.

References

[1] N. C. Francisco, N. M. M. Rodrigues, E. A. B. da Silva, S. M. M. de Faria, “A generic post-deblocking filter for block based image compression algorithms,” *Signal Process. Image Commun.*, vol.27, no.9, pp.985-997, 2012.
DOI: 10.1016/j.image.2012.05.005

[2] Gwangseok Seo, Jungwon Cho, Jooheung Lee, “Implementation of Deblocking Filter Using Partial Reconfiguration on FPGA,” *KOREA INFORMATION SCIENCE SOCIETY*, pp.1010-1012, 2017.
DOI: 10.1109/ICASSP.2010.5495525

[3] DS190(v1.11), “Zynq-7000 All Programmable SoC Data Sheet : Overview”, *Xilinx*, June 2017.

[4] João Silva, Valery Sklyarov, and Iouliia Skliarova, “Comparison of On-chip Communications in Zynq-7000 All Programmable Systems-on-Chip,” *IEEE Embedded Systems Letters*, vol.7, issue: 1, 2015. DOI: 10.1109/LES.2015.2399656

[5] UG585(v1.11), “Zynq-7000 All Programmable SoC Technical Reference Manual,” *Xilinx,* 2016.

[6] WP374(v1.2) “Partial Reconfiguration of Xilinx FPGAs Using ISE Design Suite,” *Xilinx,* 2012.

[7] UG909(v2015.1), “Vivado Design Suite User Guide Partial Reconfiguration,” *Xilinx*, 2015.

[8] XAPP1231(v1.1), “Partial Reconfiguration of a Hardware Accelerator with Vivado Design Suite for Zynq-7000 AP SoC Processor,” *Xilinx*, 2015.

[9] WP416(v1.1), “Vivado Design Suite,” *Xilinx*, 2012.

[10] A. Cortes, I. Velez and A. Irizar, “High level synthesis using Vivado HLS for Zynq SoC: Image processing case studies,” *Design of Circuits and Integrated Systems (DCIS)*, 2016.
DOI: 10.1109/DCIS.2016.7845376

[11] M. Fingeroff and T. Bollaert, “High-Level Synthesis Blue Book.” *Mentor Graphics Corp.*, 2010.

[12] S. D. Kim, J. Yi, H. M. Kim, J. B. Ra, “A deblocking filter with two separate modes in block-based video coding,” *IEEE Trans. Circuits Syst. Video Technol.*, vol.9, pp.156-160, 1999.

DOI: 10.1109/76.744282
[13] PG307, “AXI Performance Monitor v5.0,” *Xilinx*, 2016.
[14] UCD9248, “Digital PWM System Controller,” *Texas Instruments*, 2012.

BIOGRAPHY

Joo-Heung Lee (Member)

1996 : BS degree in Electronic Engineering, Inha University.
1998 : MS degree in Electronic Communication Engineering, Hanyang University.
2006 : PhD degree in Electrical Engineering, The Pennsylvania State University.
1998~2000 : Researcher, R&D Complex of LG Electronics.
2006~2011 : Assistant Professor, University of Central Florida.
2011~ : Professor, Electronic and Electrical Engineering, Hongik University.

Gwang-Seok Seo (Member)

2016 : BS degree in Dept. of Electronic and Electrical Engineering, Hongik University.
2018 : MS degree in Dept. of Electronics and Computer Engineering, Hongik University.
2018~ : Associate Research Engineer, Satrec Initiative.