논문 2006-43SD-10-6

H.264/AVC를 위한 블록현상 제거필터의 병렬 하드웨어 구조

(A Parallel Hardware Architecture for H.264/AVC Deblocking Filter)

정 용 진*, 김 현 집**

(Yong-Jin Jeong and Hyun-Jip Kim)

요 약

본 논문에서는, H.264/AVC의 블록현상 제거필터의 병렬 하드웨어 구조를 제안한다. 블록현상 제거필터는 H.264/AVC에 있어서 고화질을 보장해주고 있지만, 높은 연산량을 필요로 하기 때문에 임베디드 환경에서는 하드웨어 구현이 필수적이다. 본 논문에서는 실시간 영상 처리를 위해 2개의 1-D 필터를 적용하고, Dual-port SRAM을 사용한 병렬 하드웨어 구조를 적용하였다. 구현된 하드웨어 구조는 Verilog-HDL로 나타내고 Synopsys Design Compiler와 Hynix 0.25um CMOS Cell Library를 이용하여 합성하였다. 구현된 크기는 27.3k의 하드웨어 로직 리소스를 사용하고(내부 SRAM 제외) 최대 동작 주파수는 약 100Mhz가 되었다. 제안한 병렬 구조는 하나의 매크로블록을 처리하는데 258클록이 소요되며, 이는 HD 1080P(1920화소x1080화소) 의 영상을 초당 47.8프레임으로 처리가 가능함을 말한다. 이는 하드웨어 기반의 H.264/AVC 실시간 부/복호화 시스템에 적합한 구조임을 보여준다.

Abstract

In this paper, we proposed a parallel hardware architecture for deblocking filter in H.264/AVC. The deblocking filter has high efficiency in H.264/AVC, but it also has high computational complexity. For real time video processing, we chose a two 1-D parallel filter architecture, and tried to reduce memory access using dual-port SRAM. The proposed architecture has been described in Verilog-HDL and synthesized on Hynix 0.25um CMOS Cell Library using Synopsys Design Compiler. The hardware size was about 27.3K logic gates (without On-chip Memory) and the maximum operating frequency was 100Mhz. It consumes 258 clocks to process one macroblock, witch means it can process 47.8 HD1080P(1920pixel* 1080pixel) frames per second. It seems that it can be used for real time H.264/AVC encoding and decoding of various multimedia applications.

Keywords: H.264/AVC, deblocking filter

I. 서 론

MPEG-2/4, H.263등 블록 기반의 DCT(discrete cosine transform)를 사용하는 영상부호화 방식의 복호화 과정에서는 계단잡음(staircase noise)과 격자잡음(grid noise)같은 블록현상(blocking artifact)이 발생한다[1]. 이러한 블록현상을 가지고 프레임 메모리에 저장된 영상은 움직임예측과 움직임보상의 참조영상으로 사용되어 왜곡된 화질이 그대로 남아 오류가 누적되는 문제점이 있었다. 이 때문에 H.264/AVC 영상부호화 방식에서는 프레임 메모리에 영상이 저장되기 전에 블록현상 제거필터를 추가하여 블록현상을 개선하였고, 영상의 화질 평가 기준인 PSNR값이 평균 9% 성능이 향상하는 것으로 알려져 있다[2].

H.264/AVC 부, 복호화기 에서 블록현상 제거필터는 상당한 화질의 이득을 가져다주지만 코딩 루프에 포함되어 복호화기 전체 연산량의 약1/3을 차지할 정도로 많은 계산량을 필요로 한다. 그림 1의 그래프는 H.264/AVC 복호화기의 연산 복잡도를 분석한 결과로 약1/3의 계산량을 나타내고 있다[3]. 그래서 H.264/AVC 시스템을 실시간으로 처리하기에 어려움이 많아, 계산

* 정회원, ** 학생회원, 광운대학교 전자통신공학과
(Dept. of Electronics and Communication Engineering Kwangwoon University)
※ 본 연구는 2005년도 광운대학교 연구년 지원에 의해 연구되었습니다.
접수일자: 2006년5월17일, 수정완료일: 2006년9월4일

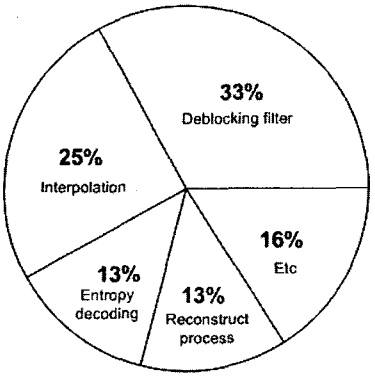

그림 1. 복호화기의 계산량 분석[3]
Fig. 1. Decoder Complexity Analysis[3].

량이 많은 모듈을 하드웨어로 구현하여 성능향상을 이루고자 하는 연구가 진행되고 있다. 그동안 블록현상 제거필터의 하드웨어구현에 관한 연구는 크게 메모리 접근을 줄이는 방법과, 필터 계산속도의 향상을 위해 계산부분의 병렬화로 진행되어 왔다[4][5].

본 논문에서는 블록현상 제거필터를 실시간으로 구현하기 위해 수직, 수평 방향의 알고리즘을 동시에 계산하는 병렬 하드웨어 구조를 제안하고, 필터 모듈의 알고리즘 중복을 줄이고 필터방식을 병렬화 하여 속도를 개선하였다. 이후 본 논문의 구성은, 먼저 Ⅱ장에서는 블록현상 제거필터의 알고리즘을 설명하고 Ⅲ장에서는 제안한 하드웨어 구조를 설명한다. Ⅳ장에서는 구현된 전체 하드웨어 구조의 시뮬레이션 결과 및 성능을 분석하며 마지막으로 Ⅴ장에서 결론을 맺는다.

Ⅱ. 블록현상 제거필터의 알고리즘

본 장에서는 블록현상 제거필터의 알고리즘을 알아보기 위해 필터가 적용되는 단위, 순서, 필요한 화소의 위치를 알아보고 블록화 현상의 정도를 검색하는 방법을 설명한다.

1. 블록현상 제거필터의 적용

H.264/AVC에서 이산 여현 변환은 4화소x4화소 크기로 이루어진다. 또한 움직임 보상의 가장 작은 단위도 4화소x4화소 크기이다. 이에 따라 블록현상도 4화소x4화소 단위마다 존재하며, 16화소x16화소 단위의 매크로블록 경계 영역에 더욱 심하게 나타난다. 이러한 블록현상의 제거는 매크로블록 단위로 하여 순차주사 방식으로 이루어지게 된다.

그림 2는 영상 한 프레임을 매크로블록 단위로 분리한 그림이다. 그림과 같이 분리된 매크로블록은 프레임 메모리에 저장된 후, 수직방향의 경계에 대해 수평방향으로 필터링을 처리하고, 처리된 영상을 이용하여 다시 수평방향의 경계에 대해 수직방향의 필터 처리를 하게 된다. 그림 3과 그림 4에서는 한 매크로블록 안에서의 필터링 순서와 필터 알고리즘에 사용되는 화소의 위치를 나타내었다. 수평방향과 수직방향의 경계면에

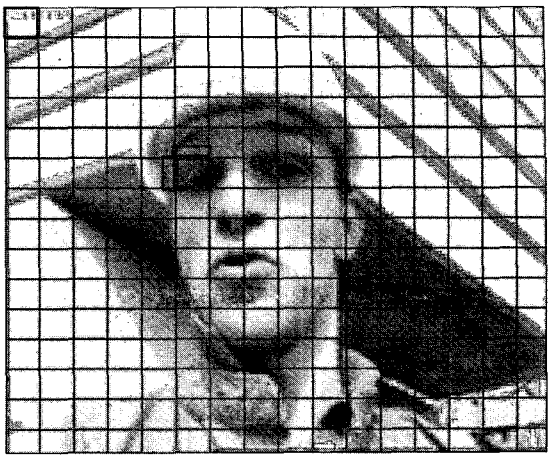

그림 2. 영상 한 프레임을 매크로블록 단위로 나눔
Fig. 2. Division of one image frame into macroblocks.

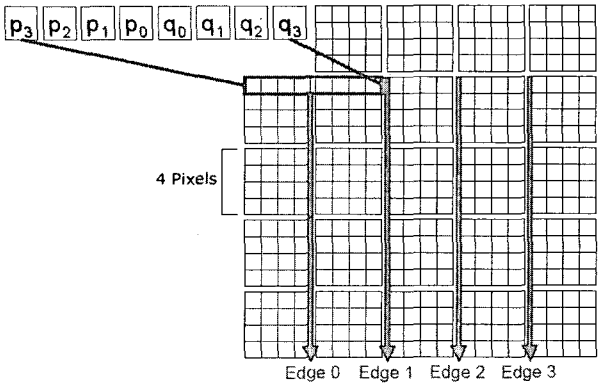

그림 3. 수직 경계면에서 필터의 적용순서
Fig. 3. Processing order for vertical edge.

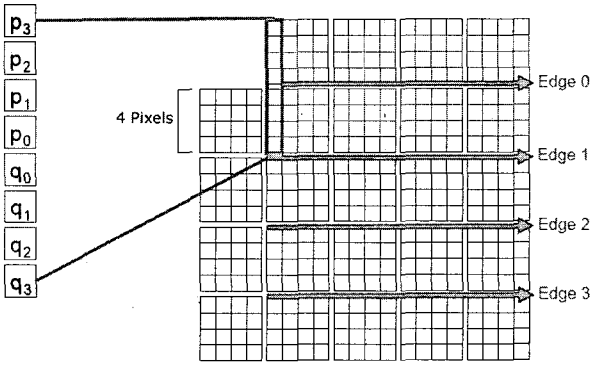

그림 4. 수평 경계면에서 필터의 적용 순서
Fig. 4. Processing order for horizontal edge.

Edge0부터 Edge3 까지 순서로 수행되고, 필터의 수행에는 블록 경계에 인접한 8개의 화소가 사용된다. 이로 인해 현재 필터를 수행하는 매크로블록의 상단과 왼쪽에 위치한 8개의 블록이 더 필요하며 경계면의 왼쪽 화소를 'p', 오른쪽 화소를 'q' 로 표시하여 경계면과의 거리에 따라 숫자를 달리하여 구분한다.

2. 블록현상 검색과정

블록현상 제거필터의 알고리즘이 적용되기 전에 각 블록의 경계마다 필터링의 정도를 구분하여 적용하기 위해 블록화 강도(Boundary strength, Bs)를 결정하게 된다. Bs값의 결정은 표1의 관계에 따라 분류된다.

표 1. 영상의 블록 경계 강도(Bs)의 정의
Table 1. Determining the boundary strength of block edge.

p, q와 화면내 부호화 관계	Bs값
p또는 q의 블록 중 적어도 한쪽이 화면내(intra) 부호화 매크로블록에 속하고 매크로블록 경계에 위치한다.	4
p또는 q의 블록 중 적어도 한쪽이 화면내 부호화 매크로블록에 속하고 매크로블록 경계에 위치하지 않는다.	3
p와 q의 블록 중 어느 쪽도 화면내 부호화 매크로블록에 속하지 않고 더 나아가 어느 한쪽이 직교변환계수를 가진다.	2
p와 q의 블록 중 어느 쪽도 화면내 부호화 매크로블록에 속하지 않고 어느 쪽도 직교변환계수를 가지지 않는다. 또 참조프레임이 다르거나 참조프레임의 매수가 다르거나 또는 움직임벡터 값이 다르다.	1
p와 q의 블록 중 어느 쪽도 화면내 부호화 매크로블록에 속하지 않고 어느 쪽도 변환계수를 가지지 않고 참조프레임과 움직임 벡터 값도 동일하다.	0

3. 필터링 과정

블록현상 검색과정을 통해 블록화 강도가 정해지면 우선적으로 필터를 수행할 지를 결정하는데 다음 두 가지 조건을 모두 만족하면 수행하게 된다.

$$Bs \neq 0 \tag{1}$$

$$\begin{aligned} &| p0 - q0 | < \alpha \\ &| p1 - p0 | < \beta \\ &| q1 - q0 | < \beta \end{aligned} \tag{2}$$

식(1)의 조건은 블록현상 검색 과정에서 Bs = 0 을 제외한 경우이고, 식(2)는 필터링 해야 하는 경계면이

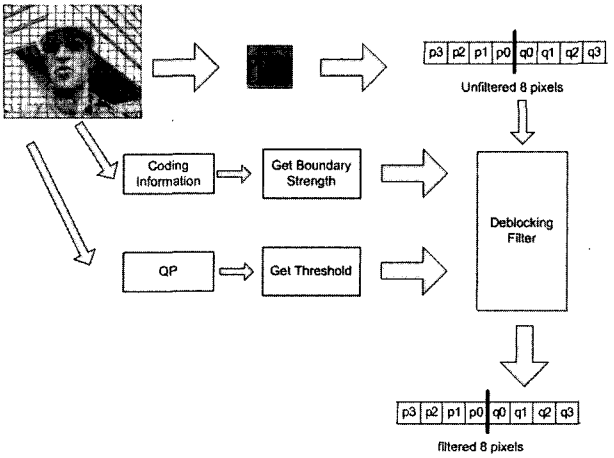

그림 5. 블록현상 제거필터의 흐름
Fig. 5. Flow of deblocking filter.

정말 블록현상에 의한 경계면인지 아니면 실제 영상의 경계면인가를 판단하는 부분이다. 여기서 α, β값은 QP (Quantization Parameter)에 의해 결정되는 Threshold 값이다.

식(1), (2)조건을 만족하면 Bs값에 따라 다른 형태의 필터 알고리즘을 수행하게 된다. 다음의 3가지 경우로 분류하여 필터 알고리즘이 적용된다.

(1) Bs = 0 의 경우

이 경우는 경계면이 블록현상으로 왜곡된 블록이 아니라고 판단하여 필터링을 수행하지 않는다.

(2) Bs = {1, 2, 3} 의 경우

이 경우는 중간 정도의 블록현상이 나타난 경우로 판단하여 4단 FIR (4-tap Finite Impulse Response) 필터를 적용한다.

(3) Bs = 4 의 경우

이 경우는 강하게 블록현상이 나타난 경우로 판단하여 양자화 계수의 값에 따라 3단, 4단, 5단 FIR 필터를 구분하여 적용한다[6].

그림 5는 블록현상 제거필터의 알고리즘을 정리한 그림이다. 영상 한 프레임을 받아서 매크로블록 단위로 분리하고, 분리한 매크로블록에서 8개의 화소와 블록 경계의 강도를 결정하기 위한 Coding Information, 마지막으로 식(2)의 α, β Threshold를 결정하는데 필요한 QP값을 받아 필터 처리된 8개의 화소를 구하게 된다.

Ⅲ. 블록현상 제거필터의 구조

본 논문에서 제안된 블록현상 제거필터의 하드웨어 구조는 그림 6에 보이는 것처럼 크게 On-chip single-

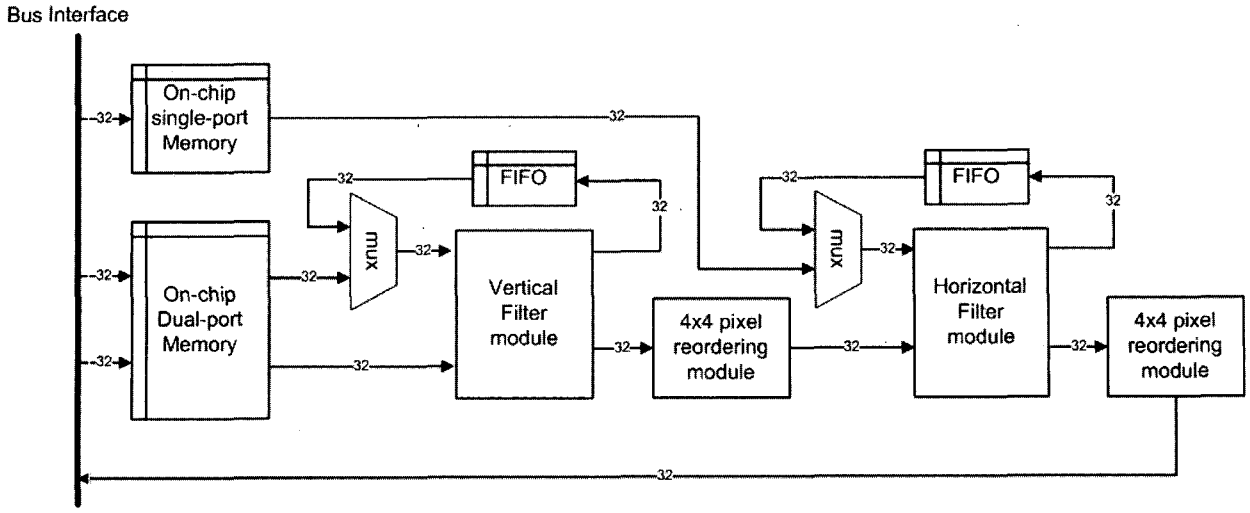

그림 6. 제안한 블록현상 제거필터의 병렬 하드웨어 구조
Fig. 6. Parallel architecture of the proposed deblocking filter.

port Memory, On-chip dual-port Memory, Vertical filter module, Horizontal filter module, 4x4 Pixel Reordering module과 FIFO로 구성되어 있다. 우선 제안된 병렬화 하드웨어 구조의 효율성을 높이기 위한 블록현상 제거필터의 수행 순서를 설명하고, 이에 필요한 내부 메모리 구조를 나타내고, 전체 동작을 설명한다.

1. 블록현상 제거필터의 수행 순서

H.264/AVC에서 블록현상 제거필터의 수행 순서는 휘도성분인 16x16개의 블록에 적용된 후 색차 성분인 2개의 4x4블록에 적용되며 각 성분에 대하여 수직방향 경계면의 필터를 한 후 수평방향 경계면의 필터를 적용한다. 하나의 필터 모듈을 가지고 수행한다면 그림 7, 그림 8에서 나타내듯 수평방향의 경계면을 그림 7의 순서대로 필터링 하고, 다시 방향을 바꾸어 그림 8의 순

서대로 필터링 하여 최종 48번의 블록단위 처리시간이 걸리게 된다. 여기서 필터링하는 시간을 단축하기 위해 두 개의 필터 모듈을 가지고 수행한다면 처리시간이 단축되게 된다.

이를 위해 개선된 필터 순서가 그림 9에 나타나고 있다. 순서를 살펴보면 수직방향의 경계면을 따라 필터링을 2번 순서까지 수행한 후 3번 필터링이 수행되는 시점에서 동시에 수평방향 경계면의 3번 필터링을 수행하게 된다. 이후부터는 수직방향 경계의 필터링과 수평방향의 필터링을 동시에 수행하게 되며, 그림 9에서 같은 숫자로 표시된 수직, 수평방향 경계면이 동시에 수행되고 있음을 나타낸다. 이는 최종 블록단위의 처리시간이 29만큼 소요되어 수직, 수평경계를 따로 처리하여 소요

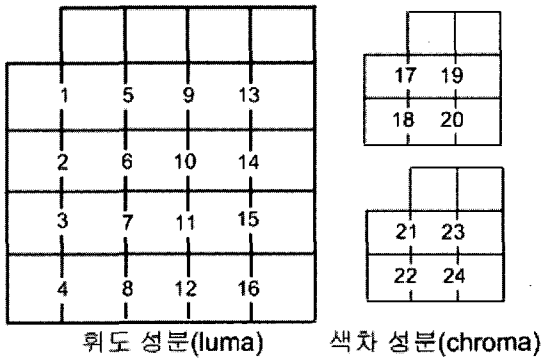

그림 7. 수직방향 경계의 필터링 순서
Fig. 7. Filtering order at verticall edges.

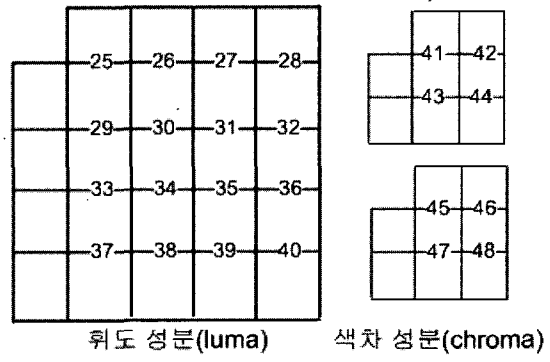

그림 8. 수평방향 경계의 필터링 순서
Fig. 8. Filtering order at horizontal edges.

되는 48번의 블록단위 처리시간보다 빠르다. 이러한 개선된 필터 순서는 2개의 필터링 모듈을 사용하여 수직, 수평방향의 경계를 동시에 수행하기 위해 제안하였고, 또 다른 이득은 1번 경계를 기준으로 왼쪽과 오른쪽 블록이 필요한 필터 알고리즘을 하드웨어로 구현할 경우 오른쪽에 위치한 블록은 다시 2번 경계를 필터링 할 경

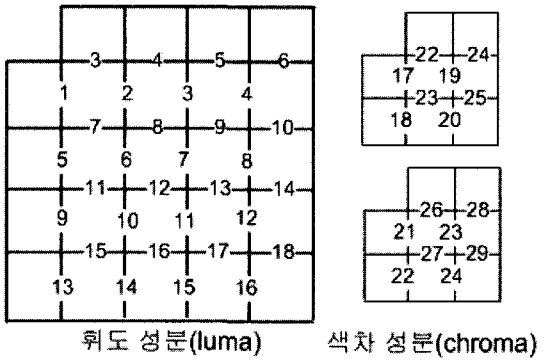

그림 9. 개선된 필터의 수행 순서
Fig. 9. Advanced filtering order of the filter.

우 바로 이용이 가능하기 때문에 메모리 접근을 줄여 성능을 향상시키고, 저전력 하드웨어 구조의 구현이 가능하다.

2. 내부 메모리의 구조

그림 9에서 제안한 필터의 수행순서를 구현하기 위해서 On-chip dual-port Memory에는 수직방향 경계의 필터에 사용되는 화소로 현재 수행중인 매크로블록과 부가적으로 필요한 왼쪽에 위치한 4개의 블록을 가지고 있고, On-chip single-port Memory에는 수평방향 경계의 필터를 위해 나머지 상단에 위치한 4개의 블록을 가지고 있다. 그림 10은 이러한 내부 메모리의 구성을 보여주고 있다. 각 메모리의 크기는 On-chip dual-port Memory가 128x32비트, On-chip single-port Memory는 32x32비트 크기이다. 그리고 모든 내부 메모리의 Word size는 32bit로 4개의 화소를 가지고 있다. 예를 들면 그림 10에서 {1, 2, 3, 4}의 화소가 1개의 Word에

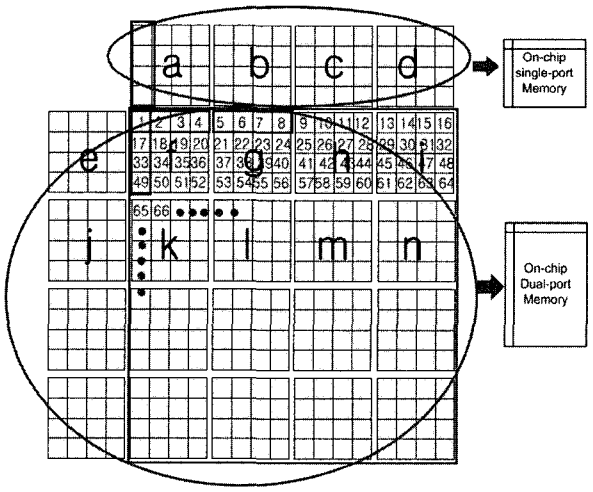

그림 10. 내부 메모리 구조
Fig. 10. Organization of the on-chip memory.

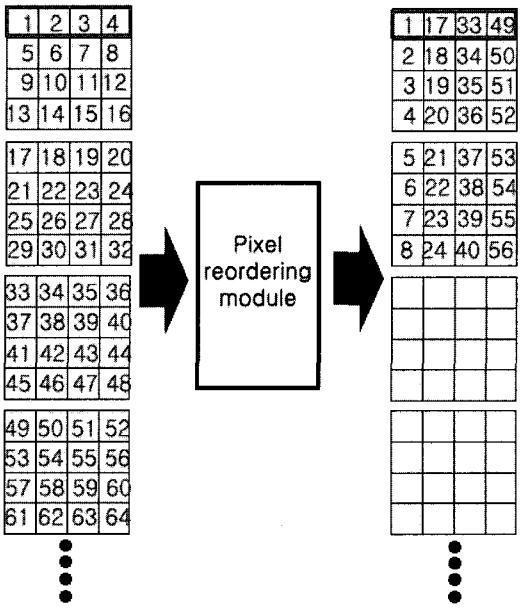

그림 11. 4x4 Pixel reordering module의 기능
Fig. 11. Function of the 4x4 Pixel reordering module.

저장되게 된다.

3. 병렬 하드웨어 구조를 위한 기능

수직, 수평 경계면을 동시에 처리하기 위한 병렬 처리 하드웨어 구조는 그림 9의 개선된 필터링 순서를 수행하기 위한 구조로, 수직 방향의 1, 2번 경계까지 처리한 후 2번 경계의 왼쪽 블록을 바로 수평 방향의 필터링 3번에 사용하면서 수직 방향의 3번 필터링도 동시에 수행하는 구조이다. 여기서 문제점은 내부 메모리의 word size는 32bit로 그림 10의 메모리 구조와 같이 {1, 2, 3, 4}, {17, 18, 19, 20}, {33, 34, 35, 36}, {49, 50, 51, 52}와 같은 단위로 읽어오게 되고 이를 필터링한 결과도 {1, 2, 3, 4}, {17, 18, 19, 20}, {33, 34, 35, 36}, {49, 50, 51, 52}와 같이 출력되게 된다는 점이다. 이 결과를 바로 수평 방향 경계인 그림 9의 3번 필터링에 사용되기 위해서는 한 개의 메모리 word의 화소 값은 {1, 17, 33, 49}, {2, 18, 34, 50}, {2, 19, 35, 51}, {4, 20, 36, 52}처럼 받아들여야 한다. 우리는 이런 변환 과정을 수행하는 4x4 pixel reordering module을 추가하여 그림 11와 같이 왼쪽의 순서와 같이 입력되는 수직 경계면을 필터링한 결과를 블록 단위로 받아들여 오른쪽의 결과와 같은 구성으로 화소 값의 위치를 다시 정렬하게 된다.

4. 필터 모듈의 구조

블록현상 제거필터의 필터 모듈을 하드웨어로 구성함에 있어 모든 선택에 의한 지연을 줄이기 위하여 그

림 12와 같은 구조를 구성하였다. 필터 모듈로 입력되는 화소 데이터는 레지스터로 받은 후 각각 By_pass, Normal filter, Strong filter, Threshold decider로 입력된다.

By_pass는 Bs값이 '0'인 경우 필터링과정을 안하는 경우의 모듈이고, Normal filter는 Bs값이 '1~3'인 경우, Strong filter는 Bs값이 '4'인 경우 각각 필터링 알고리즘에 따라 FIR필터가 적용되는 모듈이다. Threshold decider는 입력되는 QP(Quantization Parameter)값에 따라 필터링 계산에 필요한 값을 생성해주는 ROM table을 가진 모듈이다.

Bs값에 따른 각각의 필터링 모드를 결정하는 Selector는 Threshold decider로부터 필터링 계산에 필요한 값을 Normal, Strong filter모듈에 전달해주고, 출력되는 MUX의 결과를 선택하는 역할도 한다. 이는 Bs값에 따라 필터링 모드가 결정된 후 연산이 이루어지는 지연을 막기 위한 병렬화 구조로, 레지스터로 계산이 필요한 화소값을 받으면 우선적으로 모든 Bs의 경우를 계산하고 원하는 결과의 출력을 MUX로 선택해주는 구조이다[5].

필터 모듈 내부의 연산이 이루어지는 Normal filter와 Strong filter의 알고리즘 구조는 4tap부터 6tap 까지 덧셈으로 이루어져 있다. 이를 CLA(Carry Look ahead Adder)와 여러 오퍼랜드의 덧셈을 빠르게 구하기 위해 CSA(Carry Save Adder)를 이용하여 구현하였다.

5. 제안된 블록현상 제거필터의 동작

앞의 Ⅲ.1절에서 제안한 블록현상 제거필터의 수행순서를 2개의 필터링 모듈을 가지고 수직, 수평 경계면을 동시에 필터링하는 병렬 하드웨어 구조인 그림 9와 그림 10의 메모리 구조를 통해 전체 동작을 살펴본다.

우선 왼쪽의 Bus Interface를 통해 필터를 수행해야 하는 1개의 매크로블록을 받아들인다. 각 Single-port memory와 Dual-port memory에는 그림 10과 같은 구성으로 매크로블록을 가지고 있다.

다음으로 그림 10에서 표시하는 블록 e와 f의 경계의 필터를 시작으로 동작을 보면, Dual-port memory의 32bit인 2개의 출력은 Vertical Filter module로 입력된다. 결과로 필터 처리된 e, f블록 중 f블록은 다음 경계면인 f와 g의 필터링을 위해 FIFO에 쌓이게 되고 Dual-port memory에서 g블록을 불러옴으로 f와 g의 필터가 수행된다. 다시 g블록은 h블록과의 필터를 위해 FIFO로 쌓이게 된다.

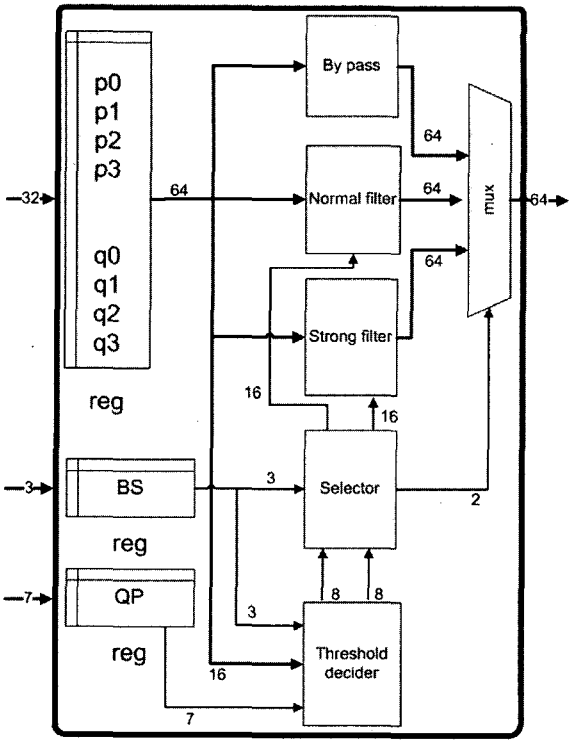

그림 12. 필터 모듈의 구조
Fig. 12. Architecture of the filter module.

블록 g와 h의 필터가 수행하는 순간부터 이전에 수직방향 경계에 대해 필터가 완료된 f블록은 a블록과 수평방향 필터를 위해 4x4 pixel reordering module을 통과하여 그림 11처럼 화소의 위치가 변경된다. 이는 다시 Horizontal Filter module에 입력되어 블록 a와 같이 수평방향 필터를 수행하고 f블록은 다시 Horizontal Filterd에 연결된 FIFO에 쌓이게 되는데 이는 아래에 있는 블록 k와의 필터에 사용되기 위함이다. 위와 같은 과정을 통해 수직, 수평 방향의 경계면에 대해 모두 필터된 화소는 마지막의 4x4 Pixel reordering module을 통과하여 그림 11의 왼쪽 메모리 구조와 같은 원래 위치로 변환되어 출력된다.

Ⅳ. 시뮬레이션 결과 및 성능 분석

본 장에서는 제안된 병렬 하드웨어 구조로 구현한 전체 모듈들에 대하여 설계 검증 및 성능 분석을 한다. 구현된 블록현상 제거필터는 Verilog-HDL을 사용하여 설계하였고, 구현된 구조의 검증을 위해 두 가지 방법을 사용하여 결과 값을 분석하였다.

먼저 Altera(사)의 Excalibur Device를 이용하여 AMBA 인터페이스를 구성하고 FPGA상에서 구현된 구조의 동작을 검증하였다. 그림 13은 Excalibur를 이용

표 2. 기존 구조들과의 성능비교
Table 2. Performance comparison with existing architectures.

		Ref [4] Advanced	Ref [4] Parallel	Ref [5]	Ref [7]	제안된 구조
1개의 MB 처리에 소요되는 클록수		814 clk	614 clk	600 clk	566 clk	258clk
HD 1920x1080 영상 (frame per second at 100Mhz)		15 FPS	20 FPS	20.5 FPS	21.8 FPS	47.8 FPS
SRAM의 종류 와 크기		dual-port (160x32bit)	two-port (96x32bit) + two-port (64x32bit)	two-port (Frame size)	8 * dual-port par allel (80x8bit)	Dual-port (128x32bit) + Single-port (32x32bit)
필터 모듈의 수		1	1	1	1	2
메모리 접근	수직 필터 수행 시	read and write	read and write	read and write	read and write	read only
	수평 필터 수행 시	read and write	read and write	read and write	read and write	write only
Process (um)		0.25um	0.25um	n/a	0.18um	0.25um
Gate count (without SRAM)		18.91k	20.66k	n/a	9.57k	27.3k

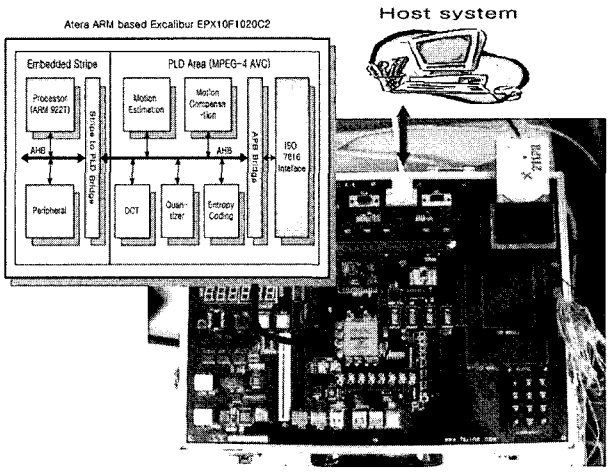

그림 13. FPGA 검증 플랫폼
Fig. 13. FPGA verification platform.

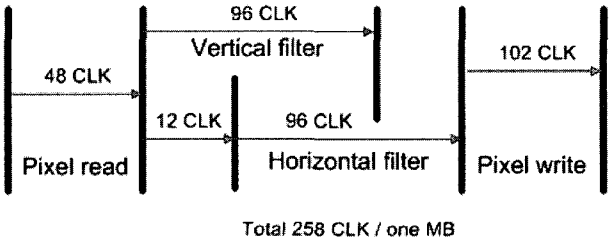

그림 14. 소요된 클록의 분포도
Fig. 14. Distribution chart of consumed clocks.

한 FPGA검증시스템의 구성을 보여준다.

FPGA검증 후 Synopsys Design-compiler와 Hynix Hsm222a 0.25um CMOS Cell Library로 합성하여 칩으로 구현하였을 때의 성능을 예측하였다. 구현된 전체 모듈을 ASIC 라이브러리와의 합성 결과, 측정된 최장 지연 경로(Critical Path)는 10.279ns이며 최대 동작 주파수는 약 100Mhz가 된다. 이러한 동작 속도에 따라서 구현된 하드웨어 로직 리소스는 내부의 SRAM을 제외하고 약 27.31k 게이트가 사용되었다.

측정된 모듈의 성능에 의해 블록현상 제거필터의 성능은, 먼저 구현된 전체 모듈의 동작 클록수를 그림 14에서 살펴보면, 외부의 Memory Interface를 통해 내부 SRAM에 값을 읽어오는데 48클록이 소요되고, Vertical Filter와 Horizontal Filter를 수행하는데 각각 96클록이 소요된다. 그리고 Horizontal Filter의 수행은 Vertical Filter의 출력을 기다리는 동안의 12클록 후부터 시작되게 된다. 최종 결과를 다시 외부 Memory Interface로 출력하는데 102클록이 소요되어 최종 1개의 매크로블록을 필터링 하는데 258클록 소요되게 된다.

구현된 블록현상 제거필터의 처리 속도를 살펴보게 되면 CIF(352화소 x 288화소) 크기의 영상을 기준으로

했을 경우 한 프레임은 330개의 매크로블록으로 이루어져 있으므로 총 85,140클록이 소요되며 동작속도 100Mhz로 보면 초당 1,174프레임의 영상 데이터를 필터링 할 수 있는 성능이다. 이 결과를 차세대 HD서비스를 위한 비디오 코덱에 적용시켜 보면 HD 720P(1280화소x720화소)의 영상 크기와 HD 1080P(1920화소x1080화소)의 고해상도 영상에 대해서 각각 초당 107프레임과 47.8프레임의 처리가 가능한 성능이 나오며 이는 실시간으로 처리가 가능함을 말해준다.

기존 블록현상 제거필터의 구조와 본 논문에서 제안한 구조를 표 2에서 비교해 보았다. 1개의 매크로블록을 처리하는데 소요되는 클록, 칩으로 구현하였을 때 100Mhz의 클록 속도로 영상을 처리하는 성능 결과와 하드웨어 구조에서의 메모리 크기와 접근정도, 그리고 구현된 구조의 크기를 비교하였다. 기존의 구조를 살펴보면 1개의 필터 모듈을 사용하면서 내부 메모리에 저장된 매크로블록을 읽어와 처리하는데 필터 시 메모리에서 읽어오고 이를 다시 업데이트하는 동작으로 메모리접근을 많이 하는 반면, 본 논문에서 제안한 구조에서는 수직방향 경계의 수행을 위한 데이터를 메모리에서 읽어온 후 이를 메모리에 업데이트하지 않고 수평방향 필터를 수행하도록 하여 메모리 접근을 최소화하였다.

표 2의 메모리 접근 항목을 보면 수직 필터와 수평필터를 수행하는 경우로 구분하여 접근의 정도를 보여준다. 내부 메모리를 제외한 하드웨어 로직 리소스의 크기를 비교해 보면 본 논문의 병렬 구조는 2개의 필터 모듈을 사용하여 [4]의 구조에 비해 하드웨어 로직 리소스의 크기는 약 35%증가 하였고, [7]의 구조와 비교하면 2배 이상 증가하였다. 그러나 [7]의 구조는 중간과정에서 필요한 버퍼레지스터를 사용하지 않고 작은 크기의 내부 메모리를 8개 사용하여 로직을 줄이고, 0.18um의 공정 라이브러리를 사용하여 적은 리소스를 차지하였다. 본 논문의 구조는 2개의 필터를 병렬로 동작하도록 하여 1개의 매크로블록을 처리하는데 100Mhz 클록 속도의 칩으로 구현하였을 경우 가장 적은 클록을 소요하게 되며 HD 1080급 고행상도 영상의 실시간 처리가 가능함을 보여준다.

V. 결 론

본 논문에서는 H.264/AVC를 위한 블록현상 제거필터의 알고리즘을 기술하고 효율적인 병렬 하드웨어 구조를 제안하였다. 블록현상 제거필터의 적용이 1개의 매크로블록을 기준으로 보면 수직방향의 경계를 모두 수행하고 이를 다시 수평방향의 경계를 수행해야 하므로 많은 메모리 접근이 불가피 하였다. 이를 개선하여 하드웨어 구조의 전체 성능을 향상시키기 위하여 2개의 1-D filter를 구성하였고, 병렬로 수직방향과 수평방향의 경계면을 보다 효율적으로 필터링 하기위한 순서를 제안하였다.

또한 위 구조를 효율적으로 수행하기 위해 Dual-port Memory의 적용으로 보다 빠른 데이터의 전달을 가능하도록 하였다. ASIC 라이브러리와 합성한 결과를 바탕으로 성능을 분석하면 HD 720P(1280화소x720화소)의 영상 크기와 HD 1080P(1920화소x1080화소)의 고해상도 영상에 대해서 각각 초당 107프레임과 47.8프레임의 처리가 가능하였다. 이는 H.264/AVC Base-line 영상 시스템에 적용은 물론 차세대 HD급의 고해상도 영상 시스템에서도 실시간 처리가 가능한 성능으로 적당한 구조임을 보여준다. 현재는 전체 H.264/AVC부호화기의 부분 중에서 인트라 예측 모듈의 설계와 인터 예측모듈의 최적화 과정에 있으며, 향후 차세대 고해상도 HD시스템에 적합한 저전력 및 실시간 처리 구조의 부호화기 하드웨어를 구현할 예정이다.

참 고 문 헌

[1] Ramamurthi, B., Gersho, A., "Nonlinear space-variant postprocessing of block coded images", Acoustics, Speech, and Signal Processing, IEEE Transactions on , Volume: 34 , Issue: 5 , pp. 1258 - 1268, Oct 1986.

[2] P. List, A. Joch, J. Lainema, G. Bjøntegaard, and M. Karczewicz, Adaptive deblocking filter," IEEE Trans, Circuits System for Video Technology, vol. 13, no. 7, pp. 614-619, Jul 2003.

[3] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro, "H.264/AVC Baseline Profile Decoder Complexity Analysis", IEEE Transaction on Circuits and System for Video Technology, Vol.13, No. 7, pp. 704-716, July 2003.

[4] Yuwen Huang, Towei Chen, "Architecture Design for Deblocking Filter in H.264/AVC", IEEE International Conference on Multimedia and Expo 2003, pp.693-696, Baltimore, Maryland, USA, July 2003.

[5] Miao Sima, Yuanhua Zhou and Wei Zhang, ""an Efficient Architecture for Adaptive Deblocking Filter of H.264/AVC Video Coding" IEEE

Transactions on Consumer Electronics, Vol. 50, Issue 1, pp. 292-296, Feb 2004.
[6] 정제창 "H.264/AVC 비디오 압축 표준", 홍릉과학 출판사, 162-167쪽, 2005.
[7] Lingfeng Li, Goto S, and Ikenaga, T "A Highly Parallel Architecture Deblocking Filter in H.264/AVC", IEICE Trans. INF. and SYST., vol. E88-D, No.7 July 2005.

저 자 소 개

정 용 진(정회원)
1983년 서울대학교 제어계측 공학과 학사 졸업.
1983년 3월~1989년 8월 한국전자 통신연구원
1995년 2월 미국 UMASS 전자 전산공학과 박사
1995년 4월~1999년 2월 삼성 전자 반도체 수석 연구원
1999년 3월 광운대학교 전자공학부 부교수
<주관심분야 : 무선 통신, 정보보호, SoC 설계, 영상처리 및 인식, 임베디드 시스템>

김 현 집(학생회원)
2005년 2월 광운대학교 정보제어 공학과 학사 졸업.
2005년 3월~현재 광운대학교 전자통신공학과 석사과정
<주관심분야 : SoC 설계, 영상처리 및 압축코덱, 임베디드 시스템>