ISSN: 1226-7244 (Print)
ISSN: 2288-243X (Online)
논문번호 13-04-04
26

Journal of IKEEE.Vol.17,No.4,418~427,December 2013
http://dx.doi.org/10.7471/ikeee.2013.17.4.418

1-D CGRA에서의 H.264/AVC 디블록킹 필터 구현

Implementation of H.264/AVC Deblocking Filter on 1-D CGRA

송 세 현*, 김 기 철*★

Sehyun Song*, Kichul Kim*★

Abstract

In this paper, we propose a parallel deblocking filter algorithm for H.264/AVC video standard. The deblocking filter has different filter processes according to boundary strength (BS) and each filter process requires various conditional calculations. The order of filtering makes it difficult to parallelize deblocking filter calculations. The proposed deblocking filter algorithm is performed on PRAGRAM which is a 1-D coarse grained reconfigurable architecture (CGRA). Each filter calculation is accelerated using uni-directional pipelined architecture of PRAGRAM. The filter selection and the conditional calculations are efficiently performed using dynamic reconfiguration and conditional reconfiguration. The parallel deblocking filter algorithm uses 225 cycles to process a macroblock and it can process a full HD image at 150 MHz.

요 약

본 논문에서는 H.264/AVC 비디오 코덱용 디블록킹 필터의 병렬 알고리즘을 제안한다. 디블록킹 필터는 BS (boundary strength)에 따라 다른 필터 연산을 수행하며, 각 필터 연산은 다양한 조건 연산을 필요로 한다. 또한 각 경계면의 연산 순서가 정해져 있기 때문에 병렬 처리가 쉽지 않다. 본 논문에서 제안하는 디블록킹 필터 알고리즘은 최근에 소개된 1-D CGRA (coarse grained reconfigurable architecture)인 PRAGRAM (pipelined reconfigurable arrays with assistant manager groups)에서 처리된다. 디블록킹 필터 연산은 PRAGRAM의 단방향 파이프라인 PE 배열 구조를 이용하여 각 필터 연산을 고속으로 수행하고, dynamic reconfiguration 및 conditional reconfiguration을 이용하여 필터 선택과 조건 연산을 효율적으로 처리한다. 디블록킹 필터의 병렬 알고리즘은 매크로블록 당 225 사이클을 소요한다. 이는 동작주파수 150 MHz에서 full HD급 영상을 처리할 수 있는 성능이다.

Key words : CGRA, H.264/AVC, PRAGRAM, deblocking filter, conditional reconfiguration

Ⅰ. 서론

휴대기기의 발달은 개인이 비디오 콘텐츠를 생성 및 배포 가능하도록 한다. 이는 휴대기기에서도 비디오 디코딩뿐만 아니라 인코딩도 가능해야 함을 의미한다. 특히 H.264/AVC 비디오 인코더는 현재 가장 많이 사용되는 코덱으로서 화면 내 움직임 예측, 화면 간 움직임 예측, 정수 변환, 양자화, 역 양자화, 디블록킹 필터, 엔트로피 코딩으로 구성된다. 각 연산 과정은 많은 연산량 때문에 일반적으로 개별 IP (intellectual property)로 설계되는데, 휴대기기와 같은 자원이 한정된 곳에서는 모든 연산과정을 개별 IP로 설계할 경우 부담이 될 수 있다. 또한 H.264/AVC 비디오 코덱은 블록단위 영상 압축을 수행하는데, 연산 과정 특성상 현재 압축하는 블록이 종료되기 전까지 다음 블록을 수행할 수 없게 된다[1]. 이는 현재 진행되는 연산 과정을 제외하고는 다른 연산 유닛들은 대

* Dept. of Electrical and Computer Engineering, University of Seoul
shsong02@gmail.com, 02-6490-5690
★ Corresponding author
※ This work was supported by Ministry of Knowledge Economy(MKE) and IDEC Platform center(IPC).
Manuscript received Oct . 16, 2013; revised Nov. 13, 2013 ; accepted Nov. 13. 2013

기 상태에 있어야 함을 의미하며, 하드웨어의 효율성을 떨어뜨릴 수 있다.

디블록킹 필터는 화면 내 움직임 예측 또는 화면 간 움직임 예측을 통해 압축된 블록 영상들 사이의 일그러짐을 제거하는 기능을 한다. 디블록킹 필터의 연산량은 다른 연산과정에 비해 적은 연산량을 요구하지만 연산과정이 복잡하기 때문에 적지 않은 하드웨어 자원을 요구한다[1]. 만약 디블록킹 필터 유닛이 개별 하드웨어로 설계된다면 이 유닛은 인코더 진행 과정에서 연산에 참여하는 시간보다 대기 시간에 있을 경우가 더욱 많게 된다. 이는 자원이 한정된 휴대기기에서는 매우 비효율적일 것이다. 만약 하나의 연산 유닛에서 기능을 변경해 가면서 H.264/AVC 비디오 인코더의 연산과정을 모두 해결할 수 있다면, 위에서 언급한 문제점을 해결할 수 있을 것이다.

CGRA (coarse grained reconfiguration architecture)는 복수개의 PE (processing element)들로 이루어져 있으며, PE들의 기능을 재구성하여 다양한 어플리케이션을 처리 할 수 있다[2]. CGRA는 목표 어플리케이션들에 따라 고려해야할 사항이 많다. PE 내부 구조, PE간의 연결방법, 빠른 재구성 기능, PE들에 지속적인 데이터 공급 방법 등이 이에 해당하며, 고려 사항들에 따라 처리 어플리케이션의 효율성이 달라진다. 기존 많은 CGRA들은 비디오 코덱을 처리하기 위해 노력해 왔다. MorphoSys는 가장 널리 알려진 CGRA로 MPEG-2 비디오 인코딩 및 디코딩을 목표로 한다[3]. MorphoSys는 64개의 PE로 구성되고, 서로 간에 2-D 배열로 연결된다. 특히, MPEG-2 비디오 코덱의 화면 간 움직임 예측을 약 4700 사이클로 처리할 수 있는데, 이는 CIF급 영상 기준으로 초당 30장을 처리할 수 있는 성능이다. 하지만 H.264/AVC 비디오 코덱은 MPEG-2 비디오 코덱 대비 더욱 복잡한 연산을 요구하기 때문에 이 구조는 현재 사용되기에는 한계가 있다. 또 다른 CGRA에는 MORA가 있다[4, 5]. MORA는 64개의 PE로 구성되고, 각 PE는 2단계 2-D 배열로 연결된다. MORA는 DCT, H.264/AVC 정수변환, wavelet 변환에서 높은 효율을 나타내지만 다른 중요한 연산과정에 대해서는 보고된 바가 없다.

지금까지 개발된 CGRA들에서는 H.264/AVC 비디오 코덱의 디블록킹 필터를 효율적으로 처리한 경우가 없었다. 본 논문에서는 최근에 소개된 새로운 1-D CGRA인 PRAGRAM (pipelined reconfiguration arrays with groups of assistant managers)를 이용한 병렬 디블록킹 필터 알고리즘을 제안하며, 디블록킹 필터 연산을 효율적으로 지원한다. PRAGRAM은 32개의 PE를 하나의 그룹을 형성하고, 각 PE 그룹은 파이프라인 형태를 이룬다[6]. 이와 같은 PE 그룹이 4개 존재하고 경우에 따라 서로 협력하여 복잡한 연산을 수행할 수 있다. 기존 CGRA와 다른 점은 각 PE 그룹마다 이를 보조하는 AM (assistant manager)가 존재하며, 유닛 전체를 담당하는 CM (central manager)가 존재한다는 것이다. 이와 같은 계층적 구조는 PE들이 디블록킹 필터 연산에 집중할 수 있어 빠른 연산이 가능하도록 한다.

본 논문의 구성은 다음과 같다. 본론 1은 새로운 1-D CGRA인 PRAGRAM의 주요 기능을 소개한다. 본론 2는 H.264/AVC 디블록킹 필터 연산 알고리즘을 소개하고, 본론3에서는 PRAGRAM에서의 병렬 디블록킹 필터 알고리즘을 제안한다. 마지막으로 실험 결과를 통해 본 논문의 병렬 디블록킹 필터 알고리즘의 유용함을 증명한다.

II. 본론

1. PRAGRAM 구조

CGRA는 복수 개의 PE들의 협력 연산을 통해 복잡한 연산을 수행할 수 있다. 이때, PE의 내부 구조와 PE들 간의 연결 구조에 의해 처리 어플리케이션

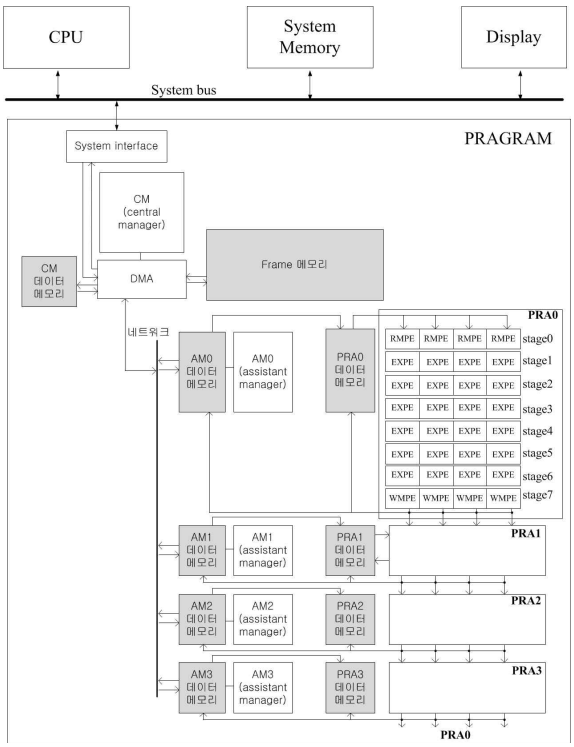


Fig. 1. Overview of PRAGRAM

그림 1. PRAGRAM 개요

의 효율이 달라질 수 있다. 일반적인 CGRA는 2-D 배열로 PE들을 배치하고, 각 PE는 주변 PE들과 데이터를 주고받을 수 있다[2]. 각 PE는 연산기 및 데이터 메모리를 가지고 있으며, 기능을 재구성하기 위한 configuration 메모리를 가지고 있다.

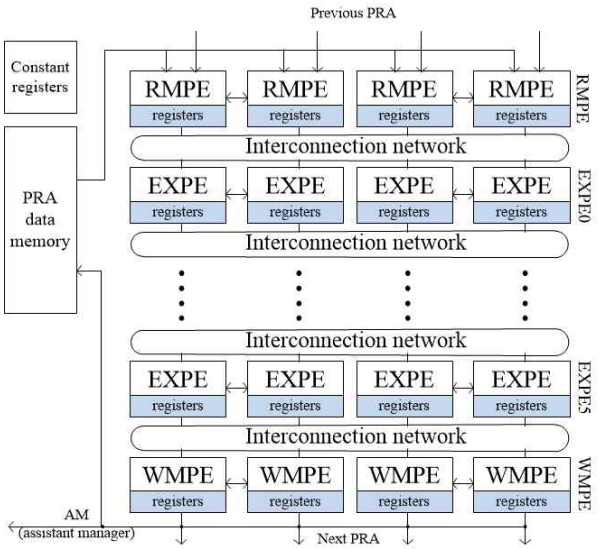


Fig. 2. PRA 및 PE 구조

그림 2. PRA and PE structures

Configuration 메모리에 저장된 configuration 데이터는 PE들이 기능을 변경하기 위해 사용되는 컨트롤 데이터로 처리 어플리케이션에 따라 그 양이 달라진다. 개별 PE들이 많은 연산기능을 지원하고 연결된 주변 PE들의 수가 많을수록 다양한 어플리케이션의 병렬처리가 가능해 질 수 있다. 하지만 PE 수가 증가할수록 필요한 자원이 기하급수적으로 늘어나기 때문에 적정한 타협점이 필요하다.

새로운 1-D CGRA인 PRAGRAM은 H.264/AVC 비디오 코덱을 이용하여 D1급 영상 처리를 목표로 하며, 화면 간 움직임 예측 과정 및 화면 내 움직임 예측 과정에 대해 초당 30장의 D1급 성능을 만족한다[6]. PRAGRAM은 그림 1에서 PRA (pipelined reconfigurable array)로 불리는 PE 배열 4개를 이용하여 비디오 어플리케이션들을 처리한다. PRA는 4개의 PE가 하나의 스테이지를 이루고, 총 8개의 스테이지로 구성된 1-D PE 배열로 설계된다. PE들은 파이프라인 형태로 설계되어 이전 스테이지들의 데이터를 입력 받을 수 있고, 다음 스테이지로 연산 결과를 전달할 수 있다.

PRA는 비디오 코덱과 같은 스트림 데이터 처리에 적합하도록 PE들의 기능을 구분하였다. PE들은 RMPE (read memory processing element), EXPE (execution processing element), WMPE (write memory processing element)로 구분된다. PRA의 첫 번째 스테이지는 4개의 RMPE로 구성된다. 그림 2 (a)의 RMPE는 메모리 읽기에 집중하며, 다른 스테이지에 데이터를 공급하는 역할을 담당한다. RMPE는 매 사이클마다 다른 주소의 데이터를 읽어 올 수 있으며, 읽어온 데이터를 내부 레지스터 파일에 저장 및 재배치하여 다음 스테이지에 데이터를 공급해 준다. 1-D 배열의 PRA는 내부 PE들에게 데이터를 전송하기 위해서 파이프라인 방법을 사용한다. 4개의 RMPE는 매 사이클마다 최대 24개의 파이프라인 레지스터에 전송될 데이터를 저장시킨다. 두 번째 스테이지는 RMPE들에서 입력받은 데이터를 이용 및 가공하여 다음 스테이지로 전달한다. 또한 여분의 파이프라인 레지스터를 이용하여 가공되지 않은 데이터를 전달시킨다. 세 번째 스테이지는 두 번째 스테이지에서 연산된 데이터와 파이프라인 레지스터를 통해 전달된 가공되지 않은 데이터로 연산을 수행한다. 이와 같은 데이터 전송 방법은 마지막 스테이지까지 반복된다.

실질적인 연산은 EXPE와 WMPE가 담당한다. 두 번째 스테이지부터 일곱 번째 스테이지는 EXPE로 구성된다. EXPE는 두 개의 16 비트 덧셈/뺄셈기로 동시에 두 가지 연산을 할 수 있으며, configuration data를 변경하여 다른 연산기능으로 사용될 수 있다. 첫 번째 연산기능은 3-입력 덧셈/뺄셈기이다. EXPE

는 그림 2 (c)의 ADD/SUB1에서 두 개의 입력 데이터를 연산하고 결과값을 ADD/SUB0에 보내어 새로운 데이터와 함께 연산된다. 두 번째 연산기능은 4-입력 덧셈/뺄셈기이다. 해당 EXPE의 ADD/SUB1과 이웃한 EXPE의 ADD/SUB1에서 각각 2개의 입력 데이터를 연산한다. 연산된 결과값은 해당 EXPE의 ADD/SUB0에서 연산되어 결과값을 출력한다. 세 번째 연산기능은 32 비트 덧셈/뺄셈기이다. ADD/SUB1의 출력 캐리 (output carry)가 ADD/SUB0의 입력 캐리 (input carry)로 사용되어 32비트 유닛으로 확장한다. 이는 L-shifter00/ L-shifter01/ R-shifter0와 L-shifter10/ L-shifter11/ R-shifter1에도 동일하게 적용된다. 앞에서 설명한 연산기능들은 모두 1 사이클 안에 재구성될 수 있어 다양한 연산에 대처할 수 있다. EXPE는 재구성을 통한 다양한 데이터 전송이 가능하다. EXPE의 switch box는 다음 스테이지로 전송될 데이터를 선택하는 역할을 수행하고, 파이프라인 레지스터는 데이터를 전송하는 역할 뿐만 아니라 내부 레지스터 용도로도 사용될 수 있다. 마지막 스테이지는 4개의 WMPE (write memory processing element)로 구성되어 메모리 쓰기를 담당한다. WMPE는 EXPE와 유사한 구조를 가지고 있으며, 메모리 컨트롤러 부분이 추가되었다.

CGRA에서 효율적인 비디오 어플리케이션을 처리하기 위해서는 빠른 configuration 데이터 변환이 가능해야 한다. Configuration 데이터는 PE의 기능을 재구성하기 위해 사용되는 컨트롤 데이터이다. PRA는 연산중에도 configuration 데이터를 변경할 수 있는 dynamic reconfiguration 기능을 지원한다. 또한 conditional reconfiguration을 이용하여 PE들 간의 조건 연산을 수행할 수 있다. Conditional reconfiguration은 주변 PE들의 연산 결과를 이용하여 실시간으로 configuration 데이터를 변경하는 방법이다. 기존의 CGRA들이 H.264/AVC 비디오 코덱을 완벽하게 지원하지 못한 이유는 복잡한 조건연산이 많아 연산 효율이 떨어지기 때문이었는데, PRA은 conditional reconfiguration으로 문제를 해결한다[6].

그림 3의 AM (assistant manager) 유닛은 PRA가 병렬 연산에 집중할 수 있도록 관리자 역할을 한다. PE 배열의 효율적인 병렬 연산을 위해서는 빠른 데이터 공급 및 재배치가 가능해야 하고, 복잡한 조건연산을 수행할 보조 연산기가 필요하다. AM은 일반적인 프로세서 형태로 연산에 필요한 파라미터들을 생성하고, PE 배열에서 연산하기 힘든 조건 연산들을 계산한다. 또한 PRA의 WMPE에서 출력된 연산 결과값들은 AM의 레지스터 파일에 바로 입력되어 AM에 의해 재배치된다. 재배치된 데이터는 PRA의 메모리로 전달되어 RMPE로 보내진다. 이와 같은 구조는 PRA가 데이터 재배치에 시간을 보낼 필요가 없이 연산에만 집중하도록 한다. 비디오 어플리케이션을 처리하기 위해서는 대량의 비디오 데이터를 지속적으로 PE 배열에 공급해줄 수 있어야하는데, AM은 내부의 고속데이터 전송기를 이용하여 하나의 명령어로 최대 128개의 데이터를 연속적으로 PRA로 전달시킨다.

AM들은 담당해야 할 PRA를 하나씩 가지고 있으며, PRAGRAM은 총 4개의 AM-PRA를 갖는다. AM-PRA는 이웃한 AM-PRA들과 협력처리를 하거나 독립적인 연산을 수행할 수 있다. 만약, 처리해야 할 연산이 복잡할 경우 AM-PRA는 그림 3과 같이 그룹을 이뤄 협력 처리한다. 첫 번째 방법은 하나의 AM과 하나의 PRA만을 이용하는 기본적인 협력 모드이다. AM은 PRA에 데이터를 공급 및 재배치를 담당하여 PRA가 연산에만 집중할 수 있도록 돕는다. 이때, 나머지 AM-PRA는 다른 연산을 수행한다. 두 번째 방법은 하나의 AM과 하나의 PRA의 협력모드를 복수 개 사용하는 방법이다. 이는 연산 과정이 여러 하위 연산과정으로 구성될 때 유용하다. PRA1는 PRA0의 결과값과 AM1에서 입력받은 새로운 데이터로 두 번째 하위 연산과정을 수행한다. 다른 하위 연산과정들도 나머지 PRA들에 할당되어 PRA 단위의 병렬 처리를 수행한다. 세 번째 방법은 PRA들의 협력 방법이다. PRA는 파이프라인 구조로 설계되어 있는데, 복잡한 연산을 수행할 경우에는 파이프라인 스테이지가 부족할 수 있다. 이 때, PRA들은 서로 연결되어 파이프라인 스테이지를 확장시킬 수 있으며, AM0를 제외한 AM들은 AM0에 데이터를 전송하는 역할만을 수행한다.

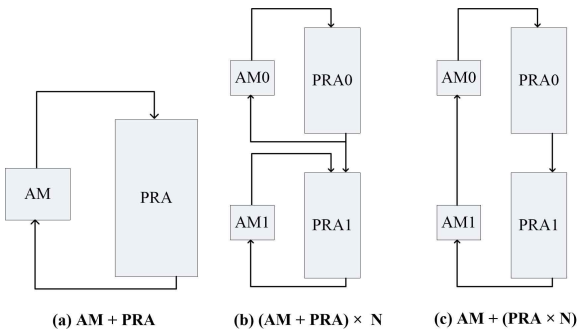


Fig. 3. AM-PRA extension methods according to task characteristics

그림 3. 처리 연산에 따른 AM-PRA 확장 방법

CM (central manager)는 PRAGRAM의 전반적인 시스템 제어를 담당한다. AM들에게 프로그램을 할당

하고, 진행상황을 체크하여 다음 연산이 딜레이 없이 수행되도록 돕는다. 또한 외부로부터 입력받은 데이터를 프레임 메모리 및 CM 메모리에 저장하고 있다가 AM들이 필요로 할 때 데이터를 제공한다. AM들에서 연산된 결과값들은 CM에서 취합하여 외부로 전송한다. PRAGRAM은 SoC에서 비디오 어플리케이션을 처리하기 위한 보조 프로세서 역할을 담당하며, 시스템 버스에 연결되어 메인 프로세서로부터 연산에 필요한 다양한 정보 및 데이터를 제공받는다. 메인 프로세서는 CM과 AM에서 실행되는 프로그램과 PRA에서 실행될 configuration 데이터를 전송하고, 처리하고자 하는 멀티미디어 어플리케이션 정보를 전달한다. 멀티미디어 어플리케이션 정보에는 비디오 코덱 종류, 목표 해상도 등이 해당된다. 제공된 정보들로 시스템을 세팅되면, 미리 약속된 통신 방식에 따라 멀티미디어 데이터를 제공 받는다. 연산된 결과값은 시스템 버스를 통해 외부 메모리 또는 디스플레이로 출력된다.

2. PRAGRAM에서의 H.264/AVC 디블록킹 필터 병렬 알고리즘

H.264/AVC 비디오 코덱은 블록 단위 영상 압축 기법을 사용한다.

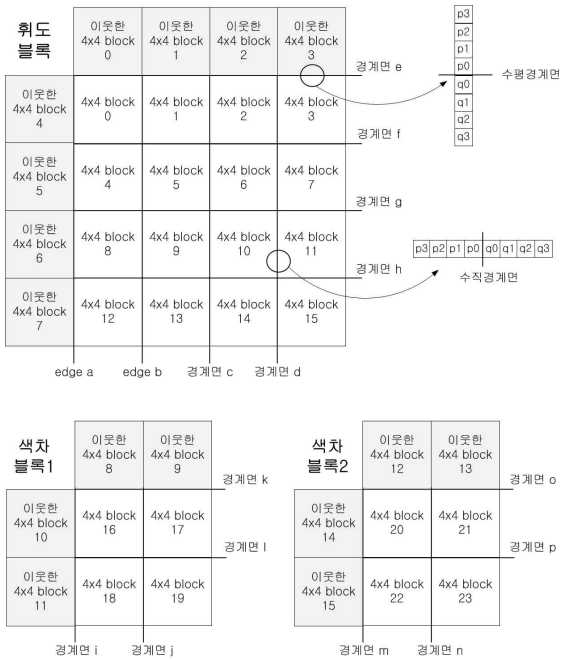


Fig. 4. The filter orders for deblocking filter

그림 4. 디블록킹 필터의 연산 순서

디블록킹 필터 연산은 화면 간 움직임 예측과 화면 내 움직임 예측을 통해 압축된 블록 영상들 사이 경계면에서의 일그러짐을 제거하는 기능을 한다. H.264/AVC 비디오 코덱은 4×4 블록을 기반으로 압축을 수행하기 때문에 디블록킹 필터 연산 시 휘도 블록과 두 개의 색차 블록에서 4×4 블록 경계마다 연산을 수행해야 한다.

그림 4는 이와 같은 디블록킹 필터의 연산 순서를 나타낸 것이다. 베이스라인 4:2:0 비디오 포맷의 영상을 처리할 경우, 휘도 블록은 16x16개의 픽셀을 가지며, 수직 경계 (vertical boundary) a, b, c, d 순으로 필터 연산을 수행하고 수평 경계 (horizontal boundary) e, f, g, h 순으로 필터 연산을 수행한다. 색차 블록은 각각 8×8개의 픽셀을 가지며, 첫 번째 색차 블록은 수직 경계 i, j 그리고 수평 경계 k, l 순으로 연산한다. 두 번째 색차 블록도 유사하게 m, n, o, p 순으로 연산한다. 각 필터 연산은 경계면 기준으로 양쪽의 4개의 픽셀을 참조 픽셀로 사용한다. 수직 경계에 대한 필터 연산은 경계면으로부터 좌측 방향으로 p0, p1, p2, p3 픽셀을 참조하고, 우측 방향으로 q0, q1, q2, q3 픽셀을 참조하여 필터 연산을 수행한다. 수평 경계에 대한 필터 연산은 경계면으로부터 위쪽방향으로 p0, p1, p2, p3 픽셀을 참조하고, 아래쪽 방향으로 q0, q1, q2, q3 픽셀을 참조한다.

디블록킹 필터 연산이 CGRA에서 수행되기 어려운 이유 중 하나는 경계면마다 다른 필터 연산을 수행해야하기 때문이다. H.264/AVC 디블록킹 필터 연산은 BS (boundary strength)에 따라 다른 필터 연산을 수행하는데, BS는 QP (quantization parameter)와 매크로 블록의 압축 방법에 의해 결정된다. BS는 0부터 4까지 존재하고, 블록 왜곡이 가장 심할 경우 BS=4가 된다. 필터 연산 방법은 크게 BS=4인 경우와 BS<4인 경우로 구분된다. 즉, 디블록킹 필터 연산을 CGRA에서 효율적으로 처리하기 위해서는 경계면마다 BS 조건을 확인하고 연산될 필터 과정을 선택할 수 있어야 한다. 필터 변환 과정이 빠르게 진행될 수 있을 때, 불필요한 연산 딜레이를 방지할 수 있게 된다.

추가적으로, 필터 연산 과정에서도 다양한 조건 연산을 필요로 한다. 필터 연산은 “BS=4 필터 연산“과 ”BS<4 필터 연산“으로 구분된다. 그림 5는 두 개의 필터 연산 과정을 나타낸 것이다. BS=4 필터 연산은 |p2-p0|가 β보다 작고 |p0-q0|가 α보다 작을 때, p0, p1, p2에 대해 필터 연산을 수행한다. 아닐 경우에는 p0에 대해서만 필터 연산을 수행한다. 또한 |q2-q0|가 β보다 작고 |p0-q0|가 α보다 작을 때, q0, q1, q2에 대해 필터 연산을 수행하며, 아닐 경우에는 q0에 대

해서만 필터 연산을 수행한다. α, β는 필터 수행 여부를 위한 문턱값에 해당하며, QP값에 의해 결정되는 상수 값이다.

● Bs = 4 filter operation

If ($|p_2-p_0| < \beta$ && $|p_0-q_0| < ((\alpha >> 2) +2)$)
P0' = ($p_2 + 2p_1 + 2p_0 + 2q_0 + q_1 + 4$)>>3
P1' = ($p_2 + p_1 + p_0 + q_0 + 2$)>>2
P2' = ($2p_3 + 3p_2 + p_1 + p_0 + q_0 + 4$)>>3
else
P0'' = ($2p_1 + p_0 + q_1 + 2$)>>2

If ($|q_2 - q_0| < \beta$ && $|p_0 - q_0| < ((\alpha >> 2) +2)$)
Q0' = ($p_1 + 2p_0 + 2q_0 + 2q_1 + q_2 + 4$)>>3
Q1' = ($p_0 + q_0 + q_1 + q_2 + 2$)>>2
Q2' = ($2q_3 + 3q_2 + q_1 + q_0 + p_0 + 4$)>>3
else
Q0'' = ($2q_1 + q_0 + p_1 + 2$)>>2

(a) BS=4 필터 연산

● Bs<4 filter operation

$t_C = t_{C0} + ((|p_2 - p_0| < \beta)?1:0) + ((|q_2 - q_0| < \beta)?1:0)$
$\Delta = Clip_3(-t_C, t_C, (((q_0 - p_0) << 2 + (p_1 - q_1) + 4) >> 3))$

$P0 = clip_3(0, 255, p_0 + \Delta)$
$Q0 = clip_3(0, 255, q_0 + \Delta)$

If(|p2 - p0| < β)
P1' = $p_1 + Clip_3(-t_{C0}, t_{C0}, (p_2+((p_0+q_0+1)>>1) - (p_1<<1))>>1)$
else P1'' = p_1

If(|q2 - q0| < β)
Q1' = $q_1 + Clip_3(-t_{C0}, t_{C0}, (q_2+((p_0+q_0+1)>>1) - (q_1<<1))>>1)$
else Q1'' = q_1

(b) BS<4 필터 연산

Fig. 5. The filter operations with BS
그림 5. BS에 따른 필터 연산

BS<4 필터 연산은 상대적으로 많은 조건 연산을 필요로 한다. 필터 연산을 수행하기 전에 t_c와 Δ을 먼저 구해줘야 한다. 이 때, t_{c0}는 α, β와 마찬가지로 QP와 BS에서 결정되는 문턱값 상수이다. 연산과정 중에 clip3(A,B,C)는 C가 A보다 작으면 C는 A로 대체하고, B보다 클 경우에는 C가 B로 대체됨을 의미한다. BS<4 필터 연산에서 p0, q0는 참조 픽셀들에 상관없이 필터 연산을 수행하며, p1과 q1는 각각 |p2-p0|가 β보다 작고 |q2-q0|가 β보다 작을 경우에 필터 연산을 수행한다.

CGRA에서의 H.264/AVC 디블록킹 필터의 병렬 알고리즘은 다음과 같은 기능을 수행할 수 있을 때, 높은 성능을 나타낼 수 있다. 첫째로, 그림 4와 같은 필터 연산 순서를 유지하면서 경계면 연산 사이의 데이터 재배치 시간을 최소화할 수 있어야 한다. 데이터 재배치 시간 동안에는 PE들이 연산을 수행할 수 없기 때문에 대기 상태에 놓이게 될 것이다. 이는 CGRA의 장점을 살릴 수 없음을 의미한다. PRAGRAM에서의 디블록킹 필터의 병렬 알고리즘은 task-level 병렬화를 통해 고속 연산을 수행하며, 연산이 진행됨과 동시에 데이터를 재배치하기 때문에 PE들이 대기 상태에 놓일 필요가 없게 된다. 둘째로, 병렬 처리 시 경계면마다 다른 필터 연산을 수행할 수 있어야 한다. 디블록킹 필터의 병렬 알고리즘은 CM, AM, PRA로 이뤄진 계층적 구조를 이용하여 PRA마다 다른 필터 연산을 수행한다. PRA가 연산하는 중에도 필터 연산을 변경할 수 있기 때문에 기능 변경에 딜레이 시간이 발생하지 않도록 구현한다. 마지막으로, PE 간의 조건 연산이 가능해야 한다. BS=4 필터 연산과 BS<4 필터 연산은 참조 픽셀 값에 따라 생성되는 픽셀 값이 달라진다. H.264/AVC 디블록킹 필터의 병렬 알고리즘은 PRA의 conditional reconfiguration을 사용하여 PE들 사이의 실시간 조건 연산을 수행한다.

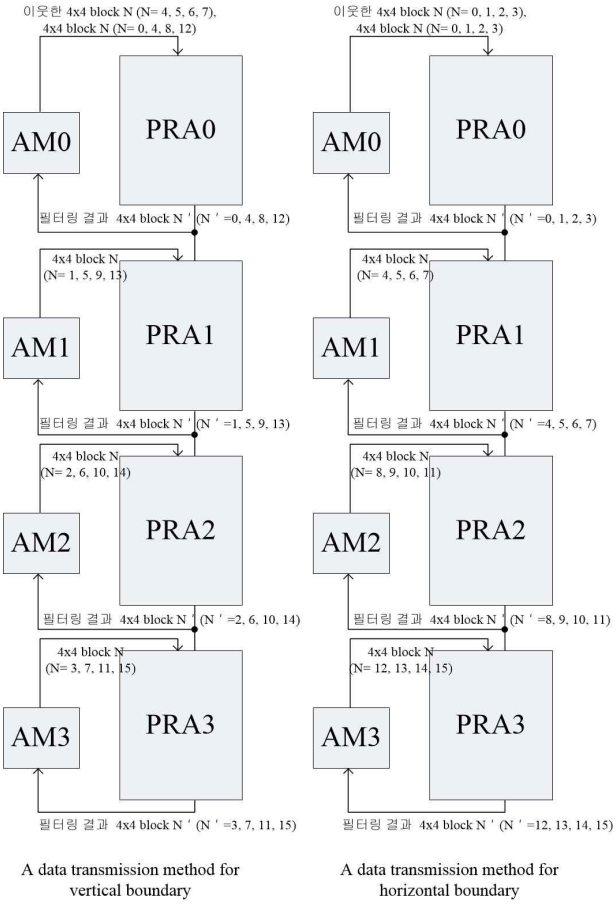


Fig. 6. The dataflow for parallel deblocking filter
그림 6. 병렬 디블록킹 필터의 데이터 처리 과정

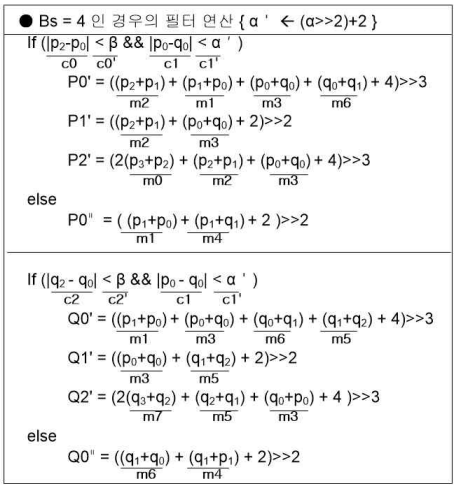
그림 6은 4개의 AM-PRA에서 필터 연산을 수행하는 방법이다. 디블록킹 필터의 병렬 알고리즘은 4개

의 PRA에서 task-level 병렬화로 구현된다. 각 PRA는 서로 다른 경계면 연산을 담당하고 연산 데이터만을 PRA들 사이로 전달하여 병렬 연산을 수행한다. 예를 들어, 수직 경계면 연산 시 PRA0부터 PRA3까지는 차례대로 경계면 a, b, c, d를 담당한다. AM0는 PRA0으로 경계면 a 연산에 필요한 이웃하고 있는 4x4 블록 4, 5, 6, 7과 4x4 블록 0, 4, 8, 12를 전송한다. PRA0는 필터 연산에 필요한 픽셀 p0, p1, p2, p3, q0, q1, q2, q3를 각 PE에 전달해 가면서 필터 연산을 수행한다. 픽셀들은 8 스테이지의 파이프라인을 지나는 동안 필터링된다. 경계면 a를 처리하기 위해서 PRA는 총 16번의 필터링을 수행하는데, 파이프라인 버블 없이 연속적으로 수행되기 때문에 초기 파이프라인 딜레이를 제외하고 16 사이클 만에 경계면 a를 처리할 수 있게 된다. AM0-PRA0에서 첫 번째 필터 연산이 종료될 때, AM1에서 경계면 b 연산에 필요한 4x4 블록 1, 5, 9, 13을 추가적으로 PRA1에 전송한다. PRA1은 PRA0의 결과값인 필터링된 4x4 블록 0, 4, 8, 12와 새롭게 입력받은 4x4 블록 1, 5, 9, 13을 이용하여 16 사이클 동안 경계면 b를 처리한다. PRA2와 PRA3에서 이와 동일한 방법으로 각각 경계면 c와 경계면 d를 처리한다. 휘도 블록의 수직 경계면 연산은 PRA0부터 PRA3까지 확장된 파이프라인에서 연속적으로 처리되며, 경계면 사이에 발생할 수 있는 데이터 재배치 시간을 0으로 만든다. PRA3의 결과값인 수직 경계면 연산의 최종값은 다시 PRA0로 보내져서 수평 경계면 연산을 수행하게 된다. PRA0부터 PRA3은 각각 경계면 e, f, g, h를 수행한다. 필터 연산 과정은 수직 경계면 연산과 동일한 방법을 사용한다. 색차 블록은 휘도 블록에 비해 절반 이하의 연산량을 요구하기 때문에 PRA0와 PRA1에서 첫 번째 색차 블록을 처리하고, PRA2와 PRA3에서 두 번째 색차 블록을 처리한다. 두 개의 색차블록들은 동시에 처리되며, 휘도 블록과 마찬가지로 task-level 병렬화로 진행된다.

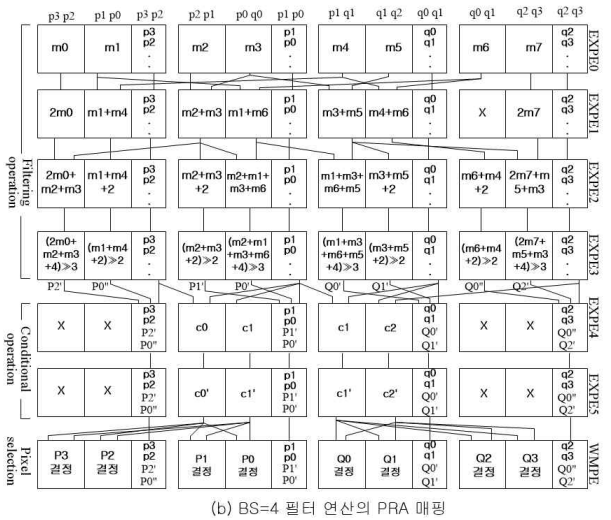
디블록킹 필터의 병렬 알고리즘은 그림 7과 같이 PRAGRAM의 계층적 구조를 이용하여 PRA마다 다른 필터 연산을 수행한다. PRAGRAM의 CM은 처리해야할 픽셀들을 AM들에게 배치해 줄 뿐만 아니라, 경계면의 세기 조건들을 AM들에게 동시에 입력해 줄 수 있다. AM들은 입력받은 픽셀 값들을 자신에게 할당된 PRA에게 전달해 주며, 처리해야 할 경계면 조건들을 분석하여 PRA의 필터연산을 결정해 준다. 이때, AM들은 개별적으로 동작되어 PRA들이 서로 다른 연산을 수행할 수 있도록 한다. PRA들은 연산 중에 생성된 필터링 결과를 해당 AM에게 전달해 주며, AM들은 CM으로 전달하여 결과값을 취합한다.

Fig. 7. A deblocking filter multitasking on PRAGRAM

그림 7. PRAGRAM에서의 디블록킹 필터 멀티태스킹



(a) BS=4 필터 연산의 병렬화 과정



(b) BS=4 필터 연산의 PRA 매핑

Fig. 8. A PRA mapping for BS=4 filter operation

그림 8. BS=4 필터 연산을 위한 PRA 매핑

CM, AM, PRA로 이뤄진 계층적 구조는 서로 다른 역할에 집중하여 연산 효율을 극대화시킨다.

디블록킹 필터 연산은 BS=4 필터 연산과 BS<4 필터 연산으로 구성된다. 그림 8은 BS=4 필터 연산의 병렬화 과정이다.

그림 8 (a)의 BS=4 필터 연산은 m0~m7으로 연산 과정을 분해할 수 있으며, 필터링 결과인 P0'~P3', P0'', Q0'~Q3, Q''의 모든 과정을 m0~m7로 표현할 수 있다. RMPE는 m0~m7 생성에 필요한 픽셀들을 배치하여 EXPE0 스테이지로 전송하고, 이후 스테이지에서 사용될 원본 픽셀 p0~p3, q0~q3을 파이프라인 레지스터로 전달한다. 그림 8 (b)의 각 PE는 3 등분되어 표시되며, 처음 두 번째는 ADD/SUB0와 ADD/SUB1로 연산 과정을 나타내며, 나머지는 파이프라인 레지스터로 데이터 전송 내용을 나타낸다. PRA의 EXPE0 스테이지부터 EXPE3 스테이지까지는 EXPE0에서 생성된 m0~m7을 조합하여 P0'~P3' ,P0'', Q0'~Q3 ,Q''을 생성한다. 그림 8 (a)의 c0 ~ c2, c0'~c2'는 필터링된 픽셀들을 선택하기 위한 조건 연산 과정이다. EXPE4에서는 조건 연산의 c0, c1, c2를 구해주고, EXPE5에서 조건절 c0', c1', c2' 값을 계산한다. WMPE 스테이지에서는 conditional reconfiguration 방법을 이용하여 최종 필터링 결과를 선택한다. WMPE들은 c0', c1', c2' 값에 따라 PE의 입력 MUX의 컨트롤 비트를 실시간으로 변경하는 방식으로 픽셀을 선택한다.

그림 9는 BS<4 필터 연산의 병렬화 과정으로 BS=4 필터 연산보다 복잡한 조건 연산을 필요로 한다. 특히 clip3 연산은 두 번의 조건 과정을 필요로 한다. 예를 들어 Δ를 구하기 위해서는 Δ^1, Δ^2를 필요로 한다. Δ^1는 $-t_c$와 m2''의 크기를 비교한 것이며, Δ^2는 t_c와 m''의 크기를 비교한 것이다. (CR:c2)은 conditional reconfiguration를 표시한 것으로 Δ를 구하기 위하여 c2 조건과정을 수행한다. PRA 매핑 과정에서 회색으로 표시된 PE들은 tc, Δ 연산과정과 이를 이용한 P0, Q0 연산 과정을 나타낸 것이다. 나머지 PE들은 if문을 계산하고 P1', Q1'을 구하게 된다. 결과적으로 디블록킹 필터의 병렬 알고리즘은 BS=4 필터 연산과 BS<4 필터 연산은 완벽하게 지원하며, 파이프라인 버블 없이 연속적인 필터링 연산이 가능하다.

PRAGRAM에서의 병렬 디블록킹 필터 알고리즘은 고속 디블록킹 필터 연산을 위한 3가지 조건을 모두 만족한다. 이는 데이터 재배치 시간을 최소화하고 task-level 병렬화를 이용하여 서로 다른 필터연산을 수행한다. 각 경계면 연산은 16 사이클에 처리되며, 4개의 PRA를 이용하여 48 사이클 만에 휘도의 수직 경계면을 처리할 정도로 빠른 연산 처리가 가능하다.

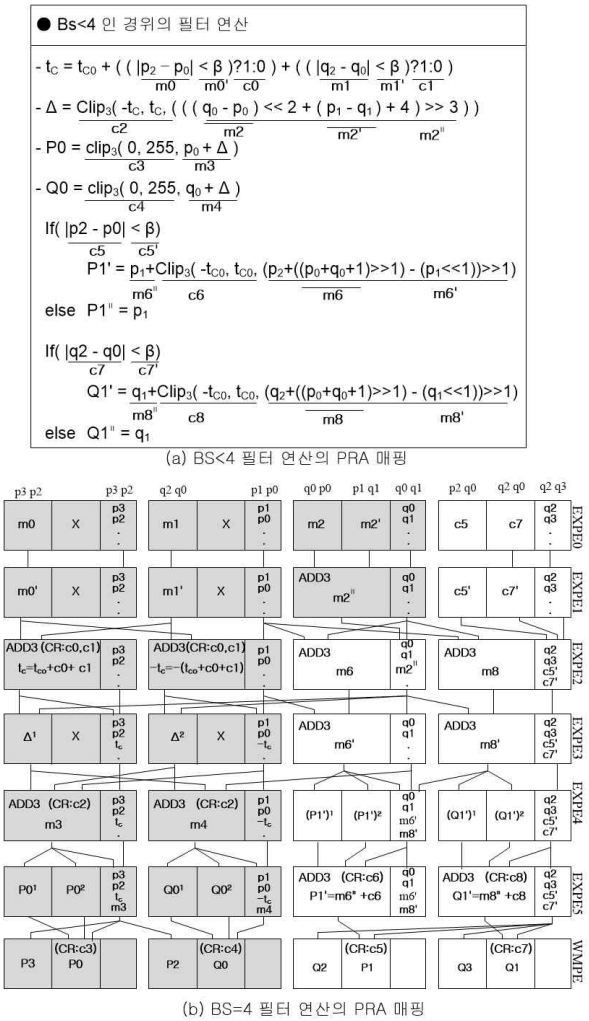


Fig. 9. A PRA mapping for BS<4 filter operation

그림 9. BS<4 필터 연산을 위한 PRA 매핑

III. 실험결과

본 논문에는 디블록킹 필터의 병렬 알고리즘을 테스트베드로 테스트하였다. 테스트베드는 호스트 PC, 임베디드 보드, 두 개의 FPGA 보드로 구성되었다. 임베디드 보드는 SoC의 메인 프로세서 역할을 담당하며 디블록킹 필터 연산에 필요한 정보 및 비디오 데이터를 FPGA 보드에 전송한다. 임베디드 보드는 S5PV310 Cortex-A9 듀얼 프로세서를 사용하며, 구동 OS는 안드로이드 2.3 버전이다. 두개의 FPGA 보드에 설계된 PRAGRAM은 32비트 GPIO 통신으로 임베디드 보드로부터 데이터들을 전송 받으며, 디블록킹 필터의 연산 결과를 임베디드 보드로 출력한다. FPGA 보드는 Xilinx사의 Virtex-5 LX330을 사용하였다. 호스트 PC는 임베디드로 출력되는 결과값을 모니터링

하는데 사용되었다. 디블록킹 필터의 병렬 알고리즘은 매크로블록 단위로 테스트하였으며, 미리 계산된 연산결과를 호스트 PC에서 비교하여 처리 여부를 확인하였다.

4개의 PRA에서의 수직 경계면 연산은 32사이클의 파이프라인 딜레이와 16사이클의 연산시간을 필요로 한다. 수직방향 경계면 연산 결과값은 그림 6과 같이 각 PRA의 WMPE로부터 AM으로 전달되며, AM들은 AM0 데이터 메모리로 전달하여 수평 경계면 연산을 준비한다. 이와 같은 데이터 재배치 과정은 PRA의 연산 과정 중에 수행되기 때문에 많은 시간을 절약할 수 있다. 재배치에 소요되는 시간은 약 20 사이클이 소요된다. 수평 경계면 연산 역시 파이프라인 딜레이를 포함하여 48 사이클이 소요된다. 16x16 휘도 블록의 디블록킹 연산은 초기 configuration 데이터 세팅 시간을 포함하여 약 150 사이클이 소요된다. 2개의 8x8 색차 블록의 디블록킹 필터 연산은 병렬로 처리된다. PRA0와 PRA1에서 첫 번째 색차 블록을 처리하고 PRA2와 PRA3에서 두 번째 색차 블록을 처리한다. 색차블록 연산 시간은 약 75 사이클이 소요되며, 매크로 블록에 대한 디블록킹 필터 연산은 총 225 사이클이 소요된다.

저자들이 조사한 바에 의하면 기존 CGRA들은 H.264/AVC 디블록킹 필터 연산을 만족할 만한 성능으로 지원하지 못했다. 그러므로 CGRA에서 수행된 디블록킹 필터의 성능을 비교할 수 없었으며, ASIC으로 설계된 전용 디블록킹 필터 유닛과 성능을 비교했다. 표1은 성능 비교표이다. 멀티미디어 어플리케이션에서의 디블록킹 필터 연산은 다른 핵심 어플리케이션에 비해 상대적으로 적은 연산량을 필요로 한다. 그러므로 다양한 어플리케이션 처리가 가능한 CGRA와 디블록킹 필터 연산만을 처리할 수 있는 ASIC 유닛의 자원비교는 무의미하여 비교 대상에서 제외하였다. 본 논문의 병렬 디블록킹 필터 알고리즘은 성능 면에서 전용 하드웨어로 설계된 디블록킹 유닛과 유사한 것을 확인할 수 있다. 매크로블록 당 처리 속도가 225 사이클일 경우 초당 30장의 full HD급 영상을 처리 할 수 있으므로 디블록킹 필터 연산에서 약간의 성능 차이는 무의미하다.

표 1. 디블록킹 필터의 성능 비교표

Table 1. Comparison of deblocking filters

	[7]	[8]	[9]	Proposed
CLK (MHz)	180	77	200	150
cycles/ MB(16×16)	192	230	192	225
Target resolution	FHD	FHD	FHD	FHD
Target frame	30 fps	30 fps	30 fps	30 fps
multi-function 지원	X	X	X	O

Ⅳ. 결론

본 논문에서 제안하는 H.264/AVC 디블록킹 필터의 병렬 알고리즘은 최근에 소개된 1-D CGRA인 PRAGRAM을 이용하여 효율적으로 처리된다. 디블록킹 필터는 동시에 복수개의 경계면을 연산하며, 경계면마다 다른 필터 연산을 처리한다. 또한 경계면 연산 사이의 데이터 재배치 시간을 줄여 불필요한 연산 딜레이를 감소 시켰다. BS=4 필터 연산과 BS<4 필터 연산은 다양한 조건 연산들을 필요로 하는데, PRAGRAM의 conditional reconfiguration 방법으로 이를 해결하였다. PRAGRAM에서의 디블록킹 필터는 연속적인 파이프라인 연산으로 48 사이클 만에 휘도 블록의 수직경계면을 처리한다. 디블록킹 필터의 최종 결과값인 하나의 16×16 휘도 블록과 두 개의 8×8 색차 블록은 총 225 사이클 만에 처리된다. 이는 동작주파수 150MHz에서 full HD급 비디오를 처리할 수 있는 성능이다.

References

[1] Joint Video Team of ITU-T and ISO/IEC JTC 1, “Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC),” Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, JVT-G050, March 2003.

[2] Hartenstein, R., "A decade of reconfigurable computing: a visionary retrospective," Design, Automation and Test in Europe, 2001. Conference and Exhibition 2001. Proceedings, vol., no., pp.642-649, 2001.

[3] Singh, H.; Ming-Hau Lee; Guangming Lu; Kurdahi, F.J.; Bagherzadeh, N.; Chaves Filho, E.M., "MorphoSys: an integrated reconfigurable system for data-parallel and computation-intensive applications," Computers, IEEE Transactions on, vol.49, no.5, pp.465-481, May 2000.

[4] Chalamalasetti, S.R.; Purohit, S.; Margala, M.;

Vanderbauwhede, W., "MORA - An Architecture and Programming Model for a Resource Efficient Coarse Grained Reconfigurable Processor," Adaptive Hardware and Systems, 2009. AHS 2009. NASA/ESA Conference on, vol., no., pp.389,396, July 29 2009-Aug. 1 2009.
[5] Purohit, S.; Rahul, S.; Margala, M.; Vanderbauwhede, W., "Throughput/Resource-Efficient Reconfigurable Processor for Multimedia Applications," Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, vol.PP, no.99, pp.1,1, 0.
[6] S. H. Song, S. H. Yoo, K. H. Kim, "PRAGRAM : Pipelined Reconfigurable Array with Groups of Assistant Managers," manuscript, 2013
[7] Y. J. Jung, K. G. Ryoo, "Optimized Hardware Design of Deblocking Filter for H.264/AVC," Journal of IEEK SD, vol. 47, no. 1, pp. 20-27, 2010.
[8] Y. H. Yu, C. H. Lee, "Design of a Pipelined Deblocking Filter with efficient memory management for high performance H.264 decoders," " Journal of IEEK SD, vol. 45, no. 1, pp. 64-70, 2008.
[9] G. Khurana and A. A.Kassim, "A Pipelined Hardware Implementation of In-loop Deblocking Filter in H.264/AVC," IEEE Transactions On Consumer Electronics, Vol. 52, No. 2, pp. 536-540, May 2006.

BIOGRAPHY

Sehyun Song (Student Member)

2006 : BS degree in Electrical and Computer Engineering, University of Seoul.
2008 : MS degree in Electrical and Computer Engineering, University of Seoul.
2008~ : Working toward the PhD degree in Electrical and Computer Engineering, University of Seoul.

Kichul Kim (Member)

1982 : BS degree in Electrical Engineering, Seoul National University.
1984 : MS degree in Electrical Engineering, Seoul National University.
1991 : PhD degree in Electrical Engineering, University of Southern California.
1984~1994 : Senior Research Engineer, Electronics and Telecommunications Research Institute (ETRI).
1994~ : Professor, in Electrical and Computer Engineering, University of Seoul.