Motion Estimation을 위한 ASIP 구현

허인구*, 김경원*, 박상현*, 양승준*, 김용주*, 백윤흥*
*서울대학교 전기공학부
e-mail : {igheo, kwkim, shpark, sjyang, yjkim}@optimizer.snu.ac.kr, ypaek@snu.ac.kr

An ASIP Implementation for Motion Estimation

In-Goo HEO*, Kyung-Won Kim*, Sang-Hyun Park*, Seung-Jun Yang*, Yong-Joo Kim*, Yun-Heung Paek*
*Dept of Electrical Engineering, Seoul National University

요 약

기존해 있는 다수의 멀티미디어 아키텍쳐에서 Motion Estimation Block은 ASIC에 의해 구현되어 왔다. 하지만 유연성의 결여와 NRE cost의 증가로 인해 ASIC에 의한 구현 외에 다양한 방법이 소개되고 있는 추세이다. 본 논문에서는 ASIP을 이용하여 Motion Estimation을 효과적으로 구현할 수 있음을 보이고 있다.

1. 서론

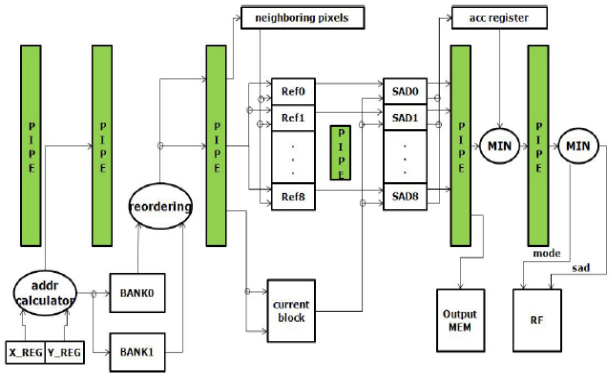
동영상의 품질을 높이고 전송 대역폭에 대한 부담을 줄이기 위해서, 최근 H.264 표준이 각광을 받고 있다. H.264 표준은 물체의 이동을 효과적으로 부호화하기 위해서 Motion Estimation을 사용하고 있는데, 이는 많은 연산을 요구한다.

지금껏 알려진 많은 멀티미디어 아키텍쳐는 이 많은 연산을 효과적으로 처리하기 위해서 여러 가지 방법의 접근을 해 왔다. [1]에서처럼 전통적인 방법으로 ASIC을 사용하기도 했고, [2]에서처럼 VLIW DSP의 SIMD 연산을 이용해 성능향상을 꾀하기도 하였다. ASIC을 이용한 방식의 경우, 성능과 전력 소모 측면에서 매우 뛰어난 모습을 보여주었다. 하지만 유연성의 결여로, 다음 버전의 제품을 만들 때 아키텍쳐 전체를 다시 손봐야 하는 단점이 있었다. 이에 비해 VLIW DSP를 이용할 경우에는, 성능에서 큰 손해를 보지 않으면서도 유연성을 가질 수 있었다. 그러나 DSP의 경우 Motion Estimation 이외의 다양한 멀티미디어 응용에 대한 지원이 이루어져야 하기에 area 측면에서 낭비가 심하였다. General Purpose Processor(GPP)를 이용하여 유연성을 극대화 시키는 방법도 있었지만, 이는 Motion Estimation이 가진 computation-intensive한 특성에 적합하지 못했다.

이런 이유로 ASIP을 이용해 Motion Estimation을 구현하는 접근 방식이 시도되고 있다[3]. ASIP을 이용한 접근은 DSP와 ASIC 사이의 간극을 줄여줄 수 있다. 유연성을 유지하면서도 성능에서 큰 손해를 보지 않는다는 것이다. 더불어 DSP가 가지고 있는 area 측면에서의 낭비 또한 줄일 수 있다.

본 논문에서는 이러한 ASIP을 이용한 접근을 통해 Motion Estimation을 효율적으로 구현할 수 있는 방법에 대해 소개하고자 한다.

2. Proposed Architecture



(그림 1) application specific instruction을 위한 data path

본 장에서는 Motion Estimation을 효과적으로 처리하기 위해 고안된 아키텍쳐에 대해 설명하고자 한다. 이 아키텍쳐는 총 8 stage의 pipeline으로 구성되어 있고 16 bit의 datapath를 가지고 있다. 일반적인 RISC적 명령어들과 함께 application specific instruction들이 지원되어 Motion Estimation과 H.264의 Intra Prediction이 효율적으로 수행될 수 있도록 한다. 2 bank memory를 이용해 읽어온 값들이 reordering 되어 Switch Unit에 제공되고, 여러 개의 Reference Block을 저장하고 있기 위해서 Reference Block register가 9개가 존재한다.

Motion Estimation에서 계산의 복잡도를 가장 증가시키는 것은, 여러 개의 Reference Block과 Current Block 사이의 absolute difference를 계산하는 부분이다. 이 부분을 Software로 구현할 경우 상당한 성능 저하를 가져오지만, 병렬성을 활용할 경우 효율적으로 수행이 가능하다. 이를

위해 이 아키텍처에서는 Sum of Absolute Difference(SAD) 값을 빠르고 효율적으로 계산하기 위한 명령어인 compte_inter_ACC와 compute_inter_SAD를 추가하였다. 9개 이하의 Reference Block data가 Reference Block Register에 load된 상태에서, 이 두 명령어를 이용하면 4x2 block 단위의 SAD값들이 각각 1 cycle 안에 계산될 수 있다.

<표 1> Motion Estimation을 위해 추가된 명령어

Instruction	역할
compute_inter_ACC	상위 4x2 block의 SAD 값을 계산하여 저장
compute_inter_SAD	하위 4x2 block의 SAD 값을 계산하여 상위 4x2 block의 SAD 값과 합친 뒤 저장, , 최소 SAD값과 최소 SAD값이 나온 Reference Block register의 number를 계산

3. Motion Estimation 구현

<표 2> Motion Estimation code의 예

```
add_reg_to_x_y r7, r1
nop
load_inter_ref 7
add_reg_to_x_y r7, r1
nop
load_inter_ref 8
compute_inter_SAD 8
```

구성된 아키텍처를 이용해 간단한 Motion Estimation을 구현해 보았다. 표2의 코드는 9개의 Reference Block data를 Reference Block register에 load한 뒤 SAD값을 한꺼번에 계산하는 부분이다. compute_iner_SAD에서 operand 8은 0번부터 8번까지의 Reference Block으로부터 모두 SAD를 계산하라는 의미이다. add_reg_to_x_y는 Reference Block을 읽어올 좌표를 기존의 좌표에서 register의 값만큼 더해서 갱신하는 명령어이다. load_inter_ref는 구해진 좌표로부터 data를 읽어와 operand로 주어진 Reference Block register에 data를 쓴다. Pipeline의 구조상 forwarding이 불가능한 1 cycle 때문에 nop 명령어가 사이에 들어간다.

이러한 방식으로 4x4 block을 Three Step Search(TSS)를 이용해 Motion Estimation 하는데 340 cycle이 걸린다. 16x16 block을 역시 TSS를 이용해 수행하면 4840 cycle이 걸린다. 하지만 현재의 coding은 Instruction Memory size를 최소화하는 방식이기 때문에 수행 시간을 줄이기 위한 최적화를 하면 실행 시간이 20% 정도 감소한다. ASIC의 수행시간에 비하면 느리다고 볼 수 있지만 일반적인 GPP를 이용한 경우보다는 매우 빠르다고 할 수 있겠다. 이에 더해, 본 논문에서는 TSS를 이용해 Motion Estimation Algorithm을 구현했지만, 주어진 명령어를 이용해 다른 Motion Estimation Algoritm을 사용하는 것이 가능하다. 이는 앞서 밝혔던 유연성 면에서 ASIC을 앞서는 것이라 하겠다.

4. 결론

ASIC을 이용한 접근은 ASIC과 GPP의 간극을 줄인다는 측면에서 매우 매력적인 방식이다. GPP의 유연성을 잃지 않으면서 ASIC에 크게 뒤처지지 않는 성능을 유지할 수 있기 때문이다. 제품의 수명과 개발 시간이 갈수록 단축되고 있는 추세 속에서, Software 적으로 Programming이 가능한 ASIP은 NRE cost의 감소라는 측면에서도 강점이 있다.

H.264 등의 최신 코덱에서 사용되는 Motion Estimation이 이러한 ASIP 방식으로 구현될 경우, 주로 수행되는 동영상의 특성에 따라 알고리즘을 쉽게 다변화할 수 있다는 것은 큰 장점이 될 것이다. 더욱 효율적인 아키텍처 설계를 통해 성능적인 측면에서도 ASIC에 근접할 수 있다면 ASIP을 이용한 구현은 더욱 각광받게 될 것이다.

참고문헌

[1] Swee Yeow Yap; McCanny, J.V., "A VLSI architecture for variable block size video motion estimation," Circuits and Systems II: Express Briefs, IEEE Transactions on , vol.51, no.7, pp. 384-389, July 2004

[2] Hyuchang Im; Wonchul Lee; Wonyong Sung, "Implementation of an H.264 motion estimation algorithm on a VLIW programmable digital signal processor," Signal Processing Systems Design and Implementation, 2005. IEEE Workshop on , vol., no., pp. 302-306, 2-4 Nov. 2005

[3] Peters, H.; Sethuraman, R.; Beric, A.; Meuwissen, P.; Balakrishnan, S.; Pinto, C.A.A.; Kruijtzer, W.; Ernst, F.; Alkadi, G.; van Meerbergen, J.; de Haan, G., "Application specific instruction-set processor template for motion estimation in video applications," Circuits and Systems for Video Technology, IEEE Transactions on , vol.15, no.4, pp. 508-527, April 2005

[4] L.-P. Chau and X. Jing, ""Efficient Three-Step Search Algorithm for Block Motion Estimation in Video Coding,"" Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing 2003 (ICASSP '03), pp. 421-424, Apr. 2003.

ACKNOWLEDGE

본 연구는 교육과학기술부/한국과학재단 우수연구센터육성사업(R11-2008-007-01001-0), 지식경제부 출연금으로 ETRI, SoC산업진흥센터에서 수행한 ITSoC 핵심설계인력양성사업, 서울시 산학연 협력사업(10560), 2009년도 정부(교육과학기술부)의 재원으로 한국과학재단의 국가지정연구실사업(No.2009-0083190)의 지원을 받아 수행되었습니다.