SAD 연산의 가속을 위한 멀티미디어 코프로세서 구현

조정현*, 정하영**
연세대학교 프로세서 연구실

Implemenation of an ASIP for acceleration SAD operation

Junghyun Jo*, HaYoung Jeong**
Processor Laboratory
Yonsei University
E-mail : *jhjo@dubiki.yonsei.ac.kr, **hyjeong@cnu.ac.kr

Abstract

An H.264 algorithm is commonly used for video compression applications. This algorithm requires a large number of data computations, for example, the sum of absolute difference (SAD) operation. We analyzed H.264 reference encoding workloads. The H.264 encoding program has 8.78% SAD operation. The SAD operation is to sum up 16 difference-values in H.264 4x4 sub-blocks. In order to accelerate SAD operations, we implemented an application specific instruction-set processor (ASIP) that can execute SAD and data transfer instructions. The proposed coprocessor has an absolute value generator and a carry save adder (CSA) unit to sum up 8 difference-values per one clock cycle. We completed SAD operation in 2 clock cycles. Experimental results show that the performance is improved by 34% of total execution time.

I. 서론

최근 인터넷과 이동통신 기술 발전으로 유무선 네트워크를 통한 고화질 동영상 데이터 전송량이 증가하고 있다. 저비용으로 고화질 동영상을 전송하기 위해 중요한 것 중의 하나가 높은 효율의 동영상 압축기술이다. 최신의 영상 압축 기술인 H.264 은 기존의 MPEG-2에 비해 60%의 월등한 압축 효율을 보이고 있으며 인터넷 비디오 스트리밍에서부터 모바일 비디오 통신에 이르기까지 비디오 압축 알고리즘으로 많이 채택되고 있다.

본 논문에서는 H.264 레퍼런스 인코더 프로그램의 실행 시간을 각 루틴별로 측정하여, 가장 많은 연산을 하는 루틴이 움직임 예측 루틴임을밝히고, 이 루틴에서 핵심인 4x4 서브 블록의 SAD(sum of absolute difference) 동작을 가속하기 위한 ARMv5 의 코프로세서로 구현하였다.

II. 본론

H.264 압축기술에서의 압축효율을 높이는데 핵심이 되는 것이 움직임 예측(Motion Estimation)이다. H.264 JM 레퍼런스 인코더를 실행하였을 때의 각 루틴들 별로 수행 시간을 Table 1에 정리하였다. Table 1을 보면, SetupFastFullPelSearch 루틴이 55.9%, SAD 루틴이 8.78%의 수행시간을 갖는다는 것을 알 수 있다. 이 루틴들은 움직임 예측에서 중요한 역할을 수행하는 동작이다.

TABLE I. H.264 JM 레퍼런스 인코더 WORKLOAD

Function Name in program	Time (%)	Dominant Operation
SetupFastFullPelSerch	55.9	ADDS AD
SAD	8.78	SAD
FastFullPelBlockMotionSearch	6.35	-
SubPelBlockMotion Search	5.33	SUB
UnifiedOneForthPix	4.38	MAC
SetupLargerBlocks	4.02	-
UMVLine16Y_11	3.71	-
FastLine16Y_11	1.74	-
dct_luma	1.14	ADD, SUB

SetupFastFullPelSearch 내에서도 SAD 동작이 포함되므로, SAD 연산량이 많다는 것을 알 수 있다. 이를 바탕으로 본 연구에서는 SAD 연산을 가속하기 위한 명령어세트 프로세서를 구현하였다. SAD 연산은 식 (1)에서 보여지듯, 레퍼런스 프레임과 현재 프레임의 차의 절대값을 더하는 연산이다[1].

$$SAD = \sum_{i=0}^{3}\sum_{j=0}^{3} | ref(i,j) - cur(i,j) | \quad (1)$$

H.264 4x4 서브블럭에서는 16개의 차들을 더하는 연산을 하게 된다.

이 SAD 연산을 가속하기 위해 ARMv5의 코프로세서에 예약되어 있는 데이터 전송 명령어 LDC, STC 와 데이터 연산 명령어 포멧을 이용하여 명령어세트를 구현하였다. 또한, 코프로세서내의 레지스터 파일은 64비트 16개의 레지스트러로 정의가 하였다. 이는 레지스터 오퍼런드의 필드가 4비트이기 때문이다[2][3].

코프로세서 데이터 전송 명령어의 L비트가 0이면 STC 명령어, 1이면 LDC명령어가 된다. SAD 명령어는 CDP 내의 CP Opc 4비트 필드를 가지고 SAD 명령어

본 연구는 한국과학재단 특정기초연구(R01-2006-000-10156-0)지원으로 수행되었음.

를 정의하였다. SAD 명령어는 16개의 차를 더하는 연산이므로, 16개의 차를 읽어와서 누적하는 과정이 필요하다. 4x4 서브블록에서의 차는 8비트 값이다. LDC명령어를 통해 16개의 차값을 읽어오면, 2개의 코프로세서 레지스터에 저장을 할 수 있다. 16개의 값을 한꺼번에 누적하는 연산기를 설계를 하게 되면, 한 사이클 내에 결과가 나오기가 힘들므로, 2 사이클에 나누어서, 첫 번째 사이클에서 8개의 차값을 합하고, 두 번째 사이클에서 나머지 8개의 차값과 첫 번째 사이클에서의 결과값을 더하여 SAD 연산을 하게 된다[4].

정의된 명령어세트를 Simple-Scalar 시뮬레이터를 사용하여 시뮬레이션 하였다[5]. 제안된 SAD 명령어를 적용하지 않은 SAD 연산을 하는 테스트 프로그램에서는 7649개의 명령어가 수행이 되고, 제안된 SAD 명령어가 적용되었을 때는 5061개의 명령어가 수행이 되었다. 이 데이터를 통해 제안된 명령어 세트를 통해 34%의 성능개선이 있음을 알 수 있다.

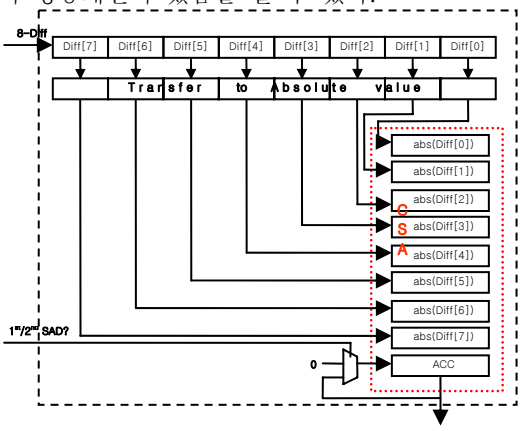

그림 1. SAD 연산기 구조

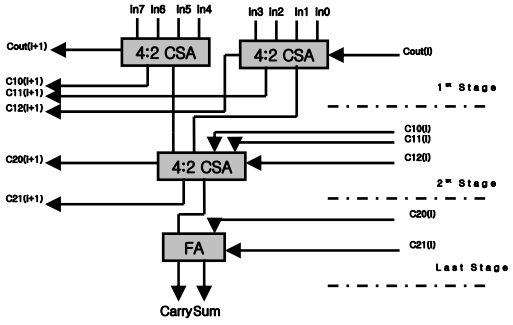

그림 2. CSA 구조

설계된 SAD 연산기 구조는 그림1에 나타나있다. 8개의 절대값을 합산하는 SAD연산기 구조이다. 8개의 입력은 먼저 절대값 생성기를 통해서 절대값으로 변환이 되고, 이 절대값이 케리 저장 덧셈기(CSA)에서 더해지게 된다. CSA덧셈기 구조는 그림 2에 나타난 바와 같이 케리 전파 시간을 줄일 수 있도록 설계되었다. 더해진 결과는 축적기에 저장에 되고, 2번째 사이클의 입력이 되어 나머지 8개의 입력과 더해지게 된다. 설계된 하드웨어 구조의 합성결과는 최대 동작 주파수 60MHz, 면적은 1187.02 게이트(2-input NAND 게이트 기준)로 임베디드 시스템에서 사용하기엔 적합하다.

IV. 결론 및 향후 연구 방향

본 논문에서는 16 개의 레퍼런스 프레임과 현재프레임의 차이값을 입력받아, 그 차이값을 절대값을 취한 후, 16 개를 한꺼번에 더해주는 SAD 연산을 가속할 수 있는 연산기를 구현하였다. 시뮬레이션을 통해 제안된 명령어세트를 사용하지 않고, 기존의 ARM ISA v.5 명령어세트를 사용했을 때와 비교했을 때보다 34%의 가속 성능의 개선이 있었다는 것을 알 수 있다. 시뮬레이션에서 가속된 연산은 SATD 내의 sum-up 연산만이 가속되었으므로, 차후 컴파일러의 지원을 받아 JM 레퍼런스 인코딩 프로그램내의 더 많은 SAD 연산을 하는 루틴들을 찾아 이 명령어세트를 적용시키게 된다면, H.264 인코딩 프로그램의 더 높은 가속 성능을 기대할 수 있을 것이다.

참고문헌

[1] Sinan Yalcin, Hasan F. Ates, Ilker Hamzaoglu: *A High Performance Hardware Architecture for an SAD Reuse based Hierarchical Motion Estimation Algorithm for H.264 Video Coding.* FPL 2005: 509-514

[2] *MOVE™ Coprocessor Technical Reference Manual*, Advanced RISC Machines Ltd., 2001 and 2002.

[3] Tse-Chen Yeh, *A Multimedia Coprocessor For ARM7 Microprocessors*, Proceedings of the 15th VLSI Design/CAD Symposium, Taiwan, Aug. 2004

[4] Yong Surk Lee, *A 4 Clock Cycle 64 x 64 Multiplier with 60 MHz clock Frequency*, KITE Journal of Electronics Enginneerign, December, 1991

[5] Doug Burger & Todd M. Austin, *The SimpleScalar Tool Set*