2003년도 대한전자공학회 하계종합학술대회 논문집 제26권 제1호

효율적인 멀티미디어데이터 처리를 위한 RISC Processor의 설계

조 태 헌, 남 기 훈, 김 명 환, 이 광 엽
서경대학교 컴퓨터공학과
전화 : 02-940-7240 / 핸드폰 : 011-9972-8198

Design of a RISC Processor with an Efficient Processing Unit for Multimedia Data

Tae-heon CHO, Ki-hun NAM, Myung-hwan KIM, Kwang-youb LEE
Dept. of Computer Engineering, Seokyeong University
E-mail : lletter@home.skuniv.ac.kr

Abstract

본 논문은 멀티미디어 데이터 처리를 위한 효율적인 RISC 프로세서 유닛의 설계를 목표로 Vector 프로세서의 SIMD(Single Instruction Multiple Data) 개념을 바탕으로 고정된 연산기 데이터 비트 수에 비해 상대적으로 작은 비트수의 데이터 연산의 부분 병렬화를 통하여 멀티미디어 데이터 연산의 기본이 되는 곱셈누적(MAC ; Multiply and ACcumulate) 연산의 성능을 향상 시킨다. 또한 기존의 MMX나 VIS 등과 같은 범용 프로세서들의 부분 병렬화를 위해 전 처리 과정의 필요충분조건인 데이터의 연속성을 위해 서로 다른 길이의 데이터 혹은 비트 수가 작은 멀티미디어의 데이터를 하나의 데이터로 재처리 하는 재정렬 혹은 Packing/Unpacking 과정이 성능 전체적인 성능 저하에 작용하게 되므로 본 논문에서는 기존의 프로세서의 연산기 구조를 재이용하여 병렬 곱셈을 위한 연산기 구조를 구현하고 이를 위한 데이터 정렬 연산 구조를 제안한다.

I. 서론

멀티미디어 데이터의 수요는 고성능 컴퓨터의 대중화와 인터넷의 보급과 더불어 더욱 증가하는 추세이며 멀티미디어 데이터를 처리하는 컴퓨팅 능력은 차세대 컴퓨터 평가에 중요한 지표가 될 것으로 받아들여지고 있다[1],[2]. 멀티미디어 데이터는 일반적으로 영상 처리를 위한 8-bit 데이터나 음성정보를 위한 16-bit 단위의 비교적 적은 비트수를 갖는 데이터가 주를 이룬다. 또한 멀티미디어 데이터 연산의 많은 부분은 곱셈누적 연산이 핵심이 된다. 그러나 기존의 연산기들은 일반적으로 고정된 형태의 데이터 버스와 연산부를 가지고 있어 작은 폭을 가진 데이터도 부호 확장이나 비트확장을 통해 고정된 형태의 연산을 처리하기 때문에 연산기 회로의 사용률 저하를 가져온다[3]. 따라서 기존의 연산기 부분을 효율적으로 제어하여 다양한 폭을 갖는 데이터를 하나의 명령어로 동시에 연산할 수 있는 SIMD(Single Instruction Multiple Data) 개념이 제안되었으며 이를 바탕으로 Intel의 MMX[4], MIPS의 MDMX[5], ARM의 ARM11[6], SUN의 VIS[7]과 같은 다양한 벤더의 범용 프로세서들이 개발 되었다.

앞서 설명한 프로세서들은 SIMD 구조를 통하여 기존의 Alpha 코드보다 적게는 1.3 ~ 15 배 가량의 성능 향상을 나타내지만 이를 위해서는 작은 사이즈의 데이터들을 연결하여 연산부 비트에 맞게 재배열하는 과정이 요구 되는데 이때 발생하는 부하가 전체 성능향상에 주요한 손실을 가져온다. 이러한 결점 요소를 제거하기 위해 행렬 단위로 연산을 하는 MOM(Matrix Oriented Multimedia)[8] 그리고 CSI(Complex Streamd instruction set)[9] 가 제안되고 있다.

본 논문에서는 ARM7 과 동등한 명령어 수준을 갖

본 논문은 IDEC 사업의 지원으로 작성되었습니다.

는 RISC 프로세서를 기본으로 하여 SIMD 구조를 위한 Load/Store 명령어를 정의하고 누적곱셈연산 명령어를 추가하여 MPEG2 의 IDCT 연산을 통해 성능을 비교 평가 하였다.

II. Multimedia Data 의 연산

2.1 곱셈누적 연산

멀티미디어 데이터 연산의 성능을 향상시키는 데 있어 가장 핵심적인 명령은 곱셈누적(MAC) 연산이다. 곱셈 누적 연산기는 누적 저장기(AC), 곱셈기의 두 입력을 a, b 라고 할 때 (1)과 같은 연산을 수행한다.

$$AC = AC \pm a \times b \quad (1)$$

단, a와 b가 N 비트의 데이터 폭을 갖게 되면 AC의 데이터 비트의 폭은 2N을 만족해야 한다.
영상이나 음성처리에 많이 사용되는 FFT(2)와 디지털 필터(3)를 살펴보면 다음과 같다.

$$F(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)\, W_N^{ux} \quad , W_N = \exp[-2j\pi/N] \quad (2)$$

$$y(n) = \sum_{k=0}^{L-1} h(k)x(n-k) \quad (3)$$

위의 두 수식을 보면 두 수를 곱한 후 그 결과를 다시 더하는 연산을 N-1 만큼 수행하는 것을 확인 할 수 있는데 이는 (1)과 유사함을 확인 할 수 있다[10]. 이는 곱셈누적 연산의 성능 향상이 멀티미디어 데이터 처리에 효율적이라는 점을 시사한다.
일반적으로 곱셈누적기는 고속의 연산을 필요로 하기 때문에 어레이 곱셈기나 트리 곱셈기등이 사용된다. 하지만 본 본문에서는 기존 연산기의 추가 없이 내장형 프로세서의 적합한 형태의 곱셈누적기의 설계를 목표로 한다.

2.2 SIMD

SIMD(Single Instruction Multiple Data)는 하나의 제어유닛 하부의 다수의 프로세서 요소(PE, Processor Elements)를 포함하는 큰 규모의 정형적인 벡터머신에 사용되는 기술 이었으나 최근에는 다수개의 서로 다른 데이터들을 하나의 명령으로 기존의 연산부에서 부분 병렬화를 이용하여 연산하는 것을 의미하기도 한다[8].
예를 들면 기존 연산기를 통해 멀티미디어(영상) 데이터를 처리하는 결과를 살펴보면 32X32 의 곱셈이 가능한 곱셈기와 결과 값을 저장 할 수 있는 64비트 곱셈레지스터가 존재하면 8비트 영상의 곱셈결과는 2n으로 표현 되므로 16비트의 결과 값이 산출되므로 전체 연산기 회로의 25%만 사용하는 결과를 낳게 되어 연산기의 이용도가 떨어지게 된다.

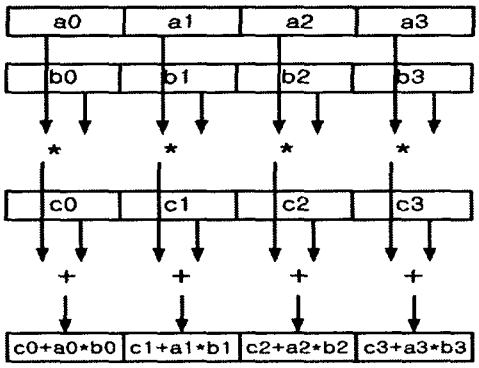

그림 1 SMID의 MAC 연산

이러한 문제점을 개선하기 위한 SIMD 연산기는 [그림 1]와 같이 32비트의 연산기에서 8비트씩 4부분으로 분할하여 각각의 데이터에 대하여 부분 병렬화를 보여주고 있다. 부분 병렬화를 통해 8비트 덧셈이나 곱셈 4개를 동시에 수행 가능 하게하여 입력되는 데이터들이 연속적으로 인가되면 2 ~ 4 배 정도의 성능 향상을 기대할 수 있으며 이를 위해 다양한 형태의 비트 수를 갖는 데이터 처리가 자유로운 형태의 부분 병렬 연산기를 제안한다.

2.3 SIMD 연산을 위한 데이터 Packing/Unpacking

부분 병렬화를 통해 성능을 향상시키기 위해서는 입력 데이터들의 연속적이어야 하는데 멀티미디어 데이터의 경우에는 커다란 메트릭스안에 서브블럭 단위로 연산되기 때문에 데이터가 연속성을 가지기 위해서는 packing 작업이 필요로 되며 이러한 전 처리 동작은 성능 저하 요인으로 작용된다. 이런 문제점을 해결하기 위한 대안으로 MOM은 영상처리 과정이 매트릭스 단위로 진행되는 점을 착안하여 전형적인 Vector ISA 의 특징과 SIMD 구조의 특징을 결합하여 메트릭스 단위의 연산을 수행하여 보다 나은 수행 결과를 얻었었지만 64-bit 워드를 갖는 16개의 매트릭스 레지스터가 요구 된다. 또한, CSI 의 경우에는 멀티미디어 데이터 처리를 위한 연산부가 추가적으로 요구되기 때문에 내

장형 프로세서는 크기면 에서 부담으로 작용한다.

본 논문에서는 부분병렬화 연산을 위한 데이터의 효율적인 Packing/Unpacking 을 위해서 기존의 64 비트의 추가적인 레지스터와 해당 연산 과정을 제어하는 상태 레지스터 그리고 기존의 crossbar switch 로 구성된 고속의 배럴시프터를 이용하여 다양한 비트 수를 갖는 데이터를 추출하고 하나의 데이터 포맷으로 융합이 가능한 블록을 하드웨어적으로 모델링한다.

III. Processor Unit 설계

3.1 곱셈누적기

본 논문에서는 사용되는 곱셈누적기는 32X32의 곱셈연산을 하여 64 비트의 연산결과물을 얻는다. SIMD 연산을 수행해야 함으로 기존의 Booth Encoder 에서 [그림 2] 와 같이 2개의 입력을 가지는 MUX 회로를 4개 추가하여 재설정 가능한 형태의 Booth Encoder 와 ALU 내부의 CLA로 구성하였다.

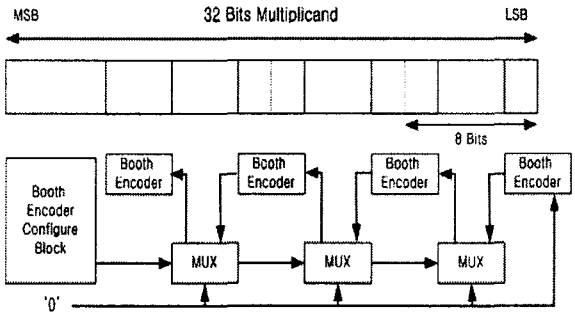

그림 2 Configurable Booth Encoder

그림에서 Booth Encoder 를 제어하는 Booth Encoder Configure는 [표 1] 과 같은 진리표를 갖는다.

Value	Operation
000	All 32-bit operation
001	All 8-bit operation
010	All 16-bit operation
011	8:8:16 operation
100	8:16:8 operation
101	16:8:8 operation

표 1 Booth Encoder Configure 의 동작

이러한 구현은 서로 다른 포맷의 데이터의 연산을 가능하게 한다. 또한 연산과 함께 Overflow 검색을 통해서 Saturation 비트가 설정 되어 있을 경우 연산 결과를 2^{n-1}의 값으로 설정하여 오류 누적을 최소화 하였다 [11]. 곱셈누적의 연산결과는 General Register File 로 전달되며 연산 결과를 순차적으로 저장하기 위해서 연산 종료 후 선택 된 레지스터에 저장된다.

3.2 Packing/Unpacking 블록

부분 병렬화의 효율적인 성능 향상을 위해서는 연속적인 데이터의 입력이 매우 중요하다. 이를 위해서 MMX나 VIS와 같은 확장 ISA들은 별도의 재정렬 혹은 Packing/Unpacking 명령어를 새로 정의하여 사용하고 있다.

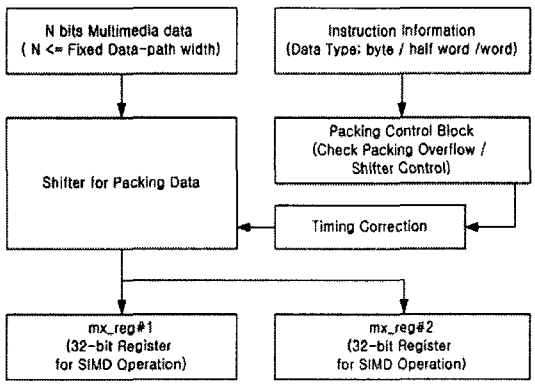

그림 3 SIMD 연산을 위한 Data Packing Block

본 논문에서는 기존의 Load/Store 명령어를 이용하여 데이터의 Load 혹은 Store 연산과 동시에 [그림 3] 과 같이 쉬프터 레지스터를 통해 연산기에 정의된 비트 수에 맞게 재정의 되어 연산에 사용된다. 이때 효율적인 Packing을 위해 레지스터화일에 바로 저장되지 않고 별도의 레지스터를 두어 저장하여 Packing하게 함으로 레지스터화일에서 인출하는 수행 사이클을 제거하였다. 또한 코드 밀집도가 높아짐으로서 상대적으로 제한적인 메모리 사용의 효율을 높일 수 있다.

3.3 추가된 명령어

추가된 명령어는 기본적으로 ARM의 명령어세트에서 벗어나지 않게 인코딩 하였으며 이를 위해 실제로는 사용되지 않는 조건비트 정보 중에서 NV(Never)상태를 이용하여 mLDR, mSTR, mMUL, mMLA과 같은 명령어를 새로 정의하였으며 기존의 명령어 셋과 같은 조건실행 연산을 기본으로 지원한다. 또한 Byte, Half Word, Word 단위의 Load/Store 가 가능하며 Base Address를 이용한 인덱싱 모드를 지원한다.

3.4 시뮬레이션 환경

본 논문에서 제안하는 RISC 프로세서는 VHDL을 기반으로 Behavior Level의 시뮬레이션을 통해 검증되었으며 Mentor사의 modelsim과 Synopsys 사의 Vhdl-analyzer를 이용하여 [그림 4]과 같이 시뮬레이션 하였다.

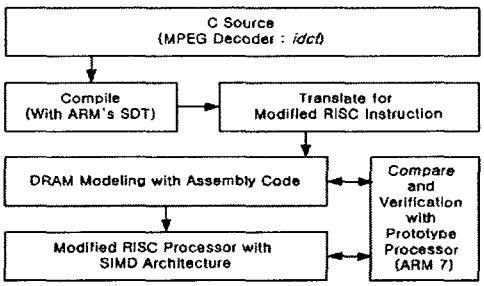

그림 4 Simulation Flow Chart

V. 결론

본 논문에서는 SIMD 구조를 갖는 RISC 프로세서의 설계를 통하여 새로 정의된 명령어들의 수행 사이클은 [표 2]와 같으며 N은 Non-sequential S는 Sequential, I는 Internal cycle을 나타낸다. [그림 5]는 데이터

Instruction Cycles(Modified RISC Processor)			
mSTR	2N	mMUL	1S + (m+3)I
mLDR	1S + 1N + 1I	mMLA	1S + (m+3)I

표 2 Instruction Cycles

Packing과정을 보여 주고 있으며 [그림 6]은 부분병렬화 곱셈 연산을 보여준다. MPEG Decoder의 IDCT 연산을 수행한 결과 [표 3]과 같은 결과를 얻었으며 기존의 프로세서와 SIMD 구조를 갖는 프로세서를 비교하였을 때 기존의 프로세서의 비해 명령어 개수와 실

그림 5 SIMD MAC Operation

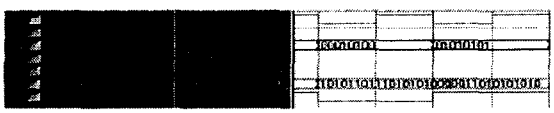

그림 6 Multimedia Data Packing Simulation

행 사이클 수가 각각 27% 그리고 34% 정도의 성능향상을 확인할 수 있어 멀티미디어 데이터 처리에 있어서 기존의 SISD(Single Instruction Single Data) 구조보다 개선됨을 검증 하였다.

Instruction \ Processor	ARM7	Modified RISC
Add/Sub	16	16
Mult	22	13
MAC	14	9
Instruction	52	38
Execution Cycle	732	484

표 3 IDCT 연산 비교

참고문헌

[1] K. Deifendorff and P. Dubey. How multimedia Workloads Will Change Processor Design. IEEE Computer, 1997

[2] R.Lee and M. Smith. Media Processing: A New Design target. IEEE Micro, 1996

[3] Corinna G. Lee and Derek J. DeVries Initial results on the performance and cost of vector microprocessors. December 1-3, 1997, IEEE Computer Society TC-MICRO and ACM SIGMICRO.

[4] Alex Peleg and Uri Weiser. MMX Technology Extension to the Intel Architecture. IEEE Micro,pages 42-50,Agust 1996.

[5] Mips extension for digital media with 3d. Technical Report http://www.mips.com, MIPS technologies.Inc., 1997.

[6] David Cromie, The ARM11 Microarchitecture, http://www.arm.com, ARM Ltd, April 2002.

[7] Marc Tremblay, J. Michael O'Connor, Venkatesh Narayanan, and Liang HE. VIS Speeds New media Processing. IEEE Micro,pages 10-20, Agust,1996.

[8] J. Corbal, M. Valero, and R. Espasa. Exploiting a New Level of DLP in Multimedia Applications. In Micro 32, 1999.

[9] J. Benm, T. Dmitri, V. Stamatis and W. harry, Implementation and Evaluation of the Complex Streamd Instruction Set

[10] 홍인표, 멀티미디어 데이터 처리에 적합한 SIMD 곱셈누적 연산기의 설계

[11] Michael J, Schulte, Pablo I, Balzola, Ahmet Akkas, and Robert W. Brocato, Interger Multiplication with Overflow Detection or Saturation, IEEE, July 2000.