스칼라 컴퓨터에 부착 사용가능한 벡터 머신 설계

조진표*, 고영웅, 조영일
한림대학교 정보통신공학부
e-mail:{yicho, jpcho, yuko}@hallym.ac.kr

Conceptual Design of the Vector Machine Attachable to Scalar Machine

Jin-Pyo Cho*, Young-Woong Ko,Young-Il Cho
Division of Information Engineering & Telecommunication,
Hallym University

요 약

데이터 주소의 계수를 위한 하드웨어 설계가 없는 본 노이만(von Neumann) 개념(SISD)의 컴퓨터에서 데이터의 주소지정은 소프트웨어적으로 수행된다. 그러므로 벡터 데이터 요소들의 주소지정은 인덱싱 기법에 의해 그 요소 수만큼 해당 변수들을 만들어서 사용해야 한다. 이것은 데이터 계수기 없이 명령어 계수기, 즉 PC(program counter)만 하드웨어로 설계되기 때문이다. 본 연구에서는 중앙처리장치 외부에 외형적 구조와 크기를 갖는 단위 벡터의 요소를 액세스하는 하드웨어 유닛의 설계를 제안한다. 제안한 방법은 시뮬레이션을 통하여 성능 검증을 하였으며, 실험 결과 동일한 프로세싱 유닛을 가지는 벡터 머신 아키텍쳐보다 12 - 30 % 정도 우수한 성능을 내는 것을 확인하였다.

1. 서론

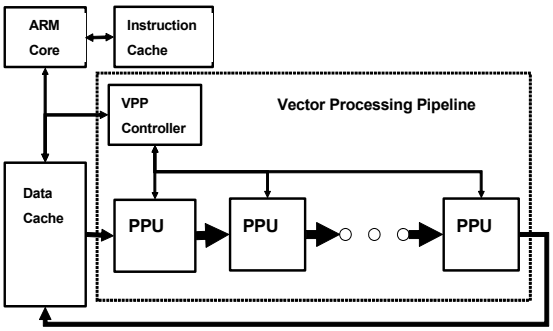
스칼라 컴퓨터에서는 하나의 명령어로 하나의 소스 오퍼랜드를 처리(SISD)하여 단 하나의 결과 값, 즉 하나의 스칼라 결과를 내는 특징을 갖는다. 그러나 벡터 컴퓨터는 하나의 명령어로 여러 개의 단위 벡터를 처리(SIMD)해야 한다. 여기서 벡터는 n(n>2)개의 요소를 가지며, 하나의 구조 내지는 크기를 갖는 구조성 단위이다. 벡터가 처리되기 위해서 그 요소들은 메모리로부터 스트림(stream)으로 이동처리 되어야 하며, 벡터 요소 수만큼의 변수가 필요하고, 그 숫자만큼의 ALU 기능에 의한 주소계산과 반복적 명령어 수행을 위한 시간을 필요로 하게 된다. 스칼라 컴퓨터에서 벡터를 처리하려면 필수 불가결로 발생하는 요소 수만큼의 변수, 주소 계산과 명령어의 반복적 수행은 스칼라 처리에서 장시간을 요구하게 되며, 이런 점이 벡터처리를 위해 구조적 기능 변화를 필요로 하는 이유이다.

벡터처리 전용 컴퓨터인 CRAY에서 이런 문제점을 해결하기 위해 메모리와 데이터 처리부 사이에 벡터 레지스터(8x64 words)를 설계하고 있다[1]. CYBER 205, ETA 10을 제외한 모든 CRAY, Fujitsu VP, Hitachi, NEC SX 계열 등이 모두 레지스터간(register to register) 처리 기법 설계이다[2]. 하지만, 이 방법은 벡터를 레지스터로 옮기는 주소계산시간과 벡터의 처리시간이 직렬로 소요된다는 것에 문제가 있다. 본 논문에서는 보다 더 빠른 벡터처리를 위해 메모리간(memory to memory) 처리 기법 설계를 제안하고 있다.

2. 관련 연구

본 논문에서 제안하는 본 노이만 구조의 스칼라 프로세서에 벡터 프로세서를 부착하는 방법과 유사한 연구로는 VPP(Vector Processing Pipeline) 프로그래밍 모델[3]과 벡터 방식의 지능형 램(VIRAM: Vector Intelligent RAM)[4,5,6,7]이 대표적이다. VPP는 계산 집약적인 무선 프로토콜을 처리하기 위한 유연한 구조로 제안되었으며, 프로그래밍 가능한 특성을 가지는 벡터 코프로세서(vector coprocessor)이다. 설계의 핵심은 데이터 레벨의 고수준의 병렬성을 효율적으로 이용하는 메모리 스트리밍 방식의

매크로 파이프라인(macro-pipelined) 벡터 구조를 이용하고 있는 것이다. VPP가 처리하는 무선 프로토콜의 경우 생산자 소비자 모델에 기반하고 있으며, 데이터가 병렬적으로 처리될 수 있는 구조를 취하고 있다. 즉 데이터가 스트림으로 입력되면 다양한 시그널 프로세싱 유닛에 의해서 처리된 후에 스트림으로 출력된다.



[그림 1] Vector Processing Pipeline 구조도

VPP 연구는 특정한 용도(무선 프로토콜과 같은 분야)에 적합하게 사용될 수 있으나, 데이터가 생산자 소비자 모델을 따르지 않는 응용에 대해서는 효과를 낼 수 없다는 단점이 있다.

VIRAM(Vector Intelligent RAM)은 멀티미디어를 처리하는 다양한 임베디드 장치를 지원해주기 위한 연구로 개발되었다. VIRAM의 주요 컴포넌트는 스칼라 코어(scalar core), 벡터 제어 유닛(vector control unit), 벡터 레인(vector lanes) 그리고 내장형 DRAM(embedded DRAM)으로 구성되어 있다. VIRAM에서는 고정 DSP 유형 오퍼레이션(fixed-point DSP-type operations)과 RGB 픽셀 처리를 위한 메모리 오퍼레이션(load sequential, every other, every third memory element), 그리고 이미지 및 비디오 처리에서 종종 사용하는 FFT(fast Fourier transforms)의 성능을 향상 시킬 수 있는 감축(reduction) 오퍼레이션을 지원해준다. 하지만 VIRAM의 경우도 앞에서 살펴본 VPP 기법과 마찬가지로 특수 목적의 응용에 적합한 형태로 설계되었으며 범용적인 벡터 처리를 지원하기에는 부족하다.

3. 벡터연산을 위한 Vector Processing Unit 설계

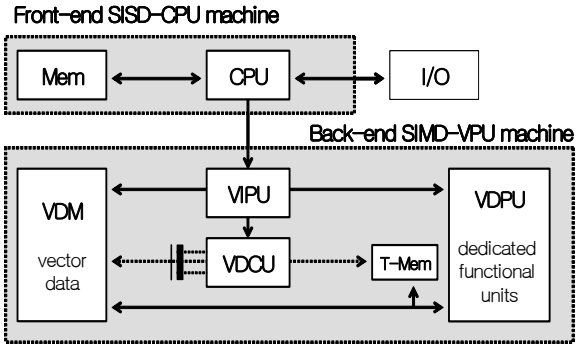
컴퓨터 처리 기법에서 하나의 연산자(○)로 여러 개의 데이터 요소를 처리하는 것(C[i]=○A[i], C[i]=A[i]○B[i], i=0,1,2,3,⋯,n-1)을 벡터처리라 한다. 여기서 벡터 요소들은 하나의 외형적 구조를 가지며, i는 각각의 요소들이 벡터구조 내에서 갖는 고유의 순서 내지는 인덱스라 한다. 초기 본 노이만(von Neumann) 컴퓨터에서 드럼 메모리의 기계식 회전속성이 명령어 액세스를 위한 명령어 계수기(PC)로 전환 설계되어 중앙처리장치(CPU)안으로 들어왔으나, 데이터는 외형적 구조 크기가 없는 단일 요소의 스칼라, 즉 SISD 개념의 스칼라 처리를 위한 설계이므로 수행 중에 오퍼랜드의 순서를 계수하는 데이터 계수의 필요성을 느끼지 못했었다. 이러한 개념(SISD)의 컴퓨터에서 n(n>2)개의 벡터 요소들이 처리되기 위해서는 n번의 SISD 수행이 필요한 것이다. 데이터 요소가 n개라는 이유로 같은 명령어가 n번 반복되어야 하고, 여기에 벡터 요소들의 주소 계산도 데이터 처리부(ALU)에서 주소 처리(addition, increment)가 필연적으로 n번 수행되어야 한다. 이런 처리기법은 데이터 처리부의 원래 목적인 데이터의 수·논리 처리 외에 (n-1)번의 메모리 액세스와 데이터 주소계산의 부담까지 추가되는 것이다. 다시 말해, 본 노이만 개념(SISD)은 하드웨어적 오퍼랜드 계수 개념 설계가 없기 때문이다. 이렇게 주소 계산 처리와 데이터 처리를 하는 데이터 처리부의 수행시간(clock period) 부담은 SIMD 개념에 의한 데이터 처리부 외부에 데이터 계수기의 설계로 완화 될 수 있는 것이다. 그러므로 메모리에 집단적 연속성의 벡터 요소를 계수하여 데이터 처리부에 공급할 수 있는 데이터 계수장치(data counter)를 중앙처리장치(CPU) 또는 데이터 처리부 외부에 하드웨어로 설계해야 된다.

벡터처리에서는 고속처리를 위해 벡터 요소들이 스트림으로 공급되어야 하기 때문에 그 기능이 없는 스칼라 컴퓨터에서는 벡터처리의 효율이 거의 없다. 즉 데이터의 스트림 공급과 처리를 위한 전용기능의 데이터 처리부는 단항벡터처리, 2-항 벡터처리, 연계벡터처리로 분류되며 파이프라인으로 설계되어야 한다. 벡터를 고속 처리하는 슈퍼컴퓨터 CRAY는 메모리와 데이터처리부 사이에 레지스터(8x64 words)를 이용하는 레지스터간 처리순서를 갖는다. 그러나 벡터 단위로 고속처리하기 위해서는 메모리간 처리순서를 가져야 하며, 고속 스트림 처리를 위해 메모리는 병렬 모듈로 설계되어야 한다[8][9]. 이러한 문제점들을 해결하기 위하여 본 연구에서는 기존의 스칼라 컴퓨터(SISD)에 부착사용을 전제로 벡터를 고속처리 하는 벡터 처리전용 프로세서(VPU: vector processing unit)를 제안한다.

1. 벡터처리장치(VPU)는 벡터 명령처리부(VIPU: vector instruction processing unit), 벡터 계수장치(VDCU: vector data counting unit), 벡터 데이터 처리부(VDPU: vector data processing unit), 벡터 데이터 메모리(VDM: vector data memory)로 구성된다.
2. 벡터는 하나의 벡터단위(vector unit)를 위해 문자변수를 사용하고, 그 벡터변수는 벡터의 구조 크기와 메모리주소를 갖는 변수 기술어(Vd:variable descriptor)를 지시한다.
3. 벡터 요소들의 주소계산을 위해 벡터 처리장치 외부에 벡터 데이터 계수장치(VDCU)를 설계하여

벡터 요소들만 계수하도록 한다.
4. 벡터 처리장치는 벡터 오퍼레이션에 따라서 전용 기능을 가지는 벡터 데이터 파이프라인 유닛(VDPU)으로 설계 한다.
5. 벡터를 제외한 모든 처리는 기존의 SISD 중앙처리장치(CPU)로 하고, 벡터 처리는 SIMD 원리에 의한 벡터 처리장치(VPU)로 한다.

제안하는 기법에서 변수기술어(Vd)는 벡터 단위와 벡터처리에 사용되는 스칼라를 위해 사용되고, SISD 개념의 중앙처리장치(CPU)와 SIMD개념의 벡터처리장치(VPU)는 Front-Back 관계를 가지며, VPU는 CPU의 지시를 받아 벡터만 처리하는 구조를 취한다. 전반적인 개념도는 그림 2와 같다.

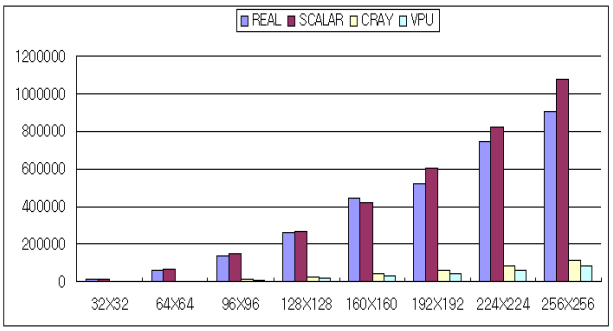


[그림 2] SISD-SIMD의 Front-Back 개념도

4. 시뮬레이션 및 성능 분석

본 연구에서는 같은 크기의 벡터를 스칼라 컴퓨터 처리기법과 CRAY 처리기법, 본 연구에서 제안하는 VPU 처리기법을 구조성 데이터의 2-차원적 액세스 처리기법에 대해 각각 시뮬레이션하고 수행 처리시간을 측정하여 성능을 비교 검토하여 평가 하였다. 시뮬레이션은 벡터 동일좌표 처리와 벡터 내적 처리에 대해서 수행 시간을 측정 하였으며 벡터 요소 수(64의 배수)로 하여 벡터의 동일좌표 처리 및 내적 연산에 대해서 성능을 측정하였다. 실험에 사용된 컴퓨터 사양은 Intel Pentium 4 2.8 GHz, 512 MByte RAM, IDE 방식의 80GByte HDD이며, 운영체제는 Linux kernel 2.6 상에서 수행되었다. 실험은 다음과 같이 4가지 방식에 대해서 결과를 구하였다.

a) REAL : PC 상에서 랜덤(random)하게 생성한 벡터 데이터에 대해서 벡터 동일 좌표 연산 및 벡터 내적에 대한 처리 속도를 구하였다.
b) SCALAR : 스칼라 방식으로 동작되는 프로세서 아키텍쳐에 대한 모델링을 하였으며, 메모리 접근 시간, 프로세서의 데이터 처리 시간, 캐쉬 히트율(cache hit ratio) 등을 고려해서 수행 시간을 측정하였다.
c) CRAY : 64개 단위로 벡터 연산을 수행하는 CRAY 아키텍쳐를 모델링 하였으며, 벡터 프로세서 유닛이 5개 있다고 가정을 하였으며, 메모리 접근 시간은 스칼라 방식과 동일한 값을 사용하였다.
d) VPU : 본 논문에서 제안하는 방식으로 Front-end SISD-CPU 모듈에서 VDM으로 전송된 벡터 데이터를 CRAY와 동일하게 5개의 벡터 프로세싱 유닛이 있다고 가정을 하여 수행 결과를 구하였다. 이때 T-Mem에 접근하는 속도는 일반 메모리에 접근하는 속도와 동일한 값을 사용하였다.

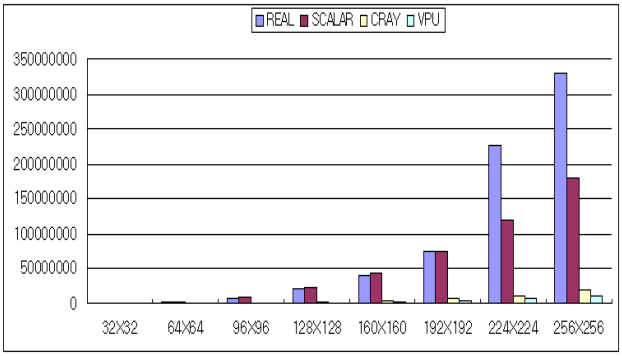


[그림 3] 동일 좌표 처리 결과 비교

그림 3에서는 REAL, SCALAR, CRAY, VPU 방식으로 동일 좌표 처리 시간을 측정하였으며, X좌표는 벡터 연산에 사용된 배열의 크기를 의미한다. 즉 32X32는 행과 열이 각각 32개의 정수로 이루어진 배열이다. Y좌표는 수행 시간이며 단위는 나노세컨드(nano second)이다. 실험 결과에서 보면 실제 배열 데이터를 가지고 PC 상에서 수행된 REAL과 PC 구조를 모델링한 SCALAR가 거의 유사한 값을 보이고 있음을 알 수 있다. CRAY 방식과 VPU 방식의 처리 결과는 VPU 방식이 처리 속도를 줄이고 있음을 보여주며, 벡터의 행과 열의 값이 커질수록 성능 차이가 크게 나타나고 있음을 알 수 있다. 본 논문에서 제안하는 VPU의 동일 좌표 연산에 대한 성능 향상 비율은 CRAY와 비교하여 27 - 30% 향상되었다.

그림 4는 REAL, SCALAR, CRAY, VPU 방식으로 내적 처리 시간을 측정하였으며, X좌표와 Y좌표는 동일 좌표 처리 결과와 동일하다. 동일 좌표 처리 결과에 비해서 REAL과 SCALAR의 수행 시간이 배열의 크기가 커질수록 차이가 벌어지고 있으며, 이것은 프로세서 내부의 캐쉬 히트율이 떨어지기 때문에 발생하는 것으로 추론된다. 왜냐하면 작은 크기의 벡터 연산에 대해서는 캐쉬에 가져온 데이터를 이용해서 메모리에 접근하는 시간을 줄일 수 있지만, 벡터 연산에 사용되는 배열의 크기가 커질수록 캐쉬를 통한 이득이 점차 떨어지기 때문이다. 동일 좌표 연산과 마찬가지로 내적 연산의 경우도 큰 크기의 벡터 연산에 대해서 VPU가 높은 성능을 보여주고 있음을 알 수 있다. 본 논문에서 제안하는 VPU의 내적 연산에 대한 성능 향상 비율은 CRAY

와 비교하여 12 - 40% 향상되었다.



[그림 4] 내적 처리 결과 비교

5. 결론

본 논문에서는 기존 벡터 처리 방식의 문제점을 CRAY 아키텍처를 중심으로 찾아보았으며 이를 해결하는 방안을 제시하고 있다. 벡터를 고속 처리하는 슈퍼컴퓨터 CRAY는 메모리와 데이터 처리부 사이에 레지스터를 이용하는 레지스터간 처리순서를 갖고 있으며, 이러한 방식은 벡터 단위로 고속처리하기 위해서 성능상의 병목지점이 될 수 있기 때문에 메모리간 처리순서를 제공해 주어야 한다. 본 연구에서는 기존의 스칼라 컴퓨터(SISD)에 부착사용이 가능한 벡터 처리전용 프로세서(VPU)를 제안하고 있다. 제안하는 기법에서 변수기술어는 벡터 단위와 벡터처리에 사용되는 스칼라를 위해 사용되고, SISD 개념의 중앙처리장치(CPU)와 SIMD개념의 벡터처리장치(VPU)는 Front-Back 관계를 가지며, VPU는 CPU의 지시를 받아 벡터만 처리하는 구조를 취한다. VPU 아키텍처의 유용성을 증명하기 위해서 SIMD 스칼라 아키텍처, CRAY 아키텍처 그리고 VPU 아키텍처를 모델링하고 시뮬레이션을 수행시켰으며, 수행 결과 동일 좌표 벡터 연산 및 벡터 내적과 같은 고속의 벡터 처리 문제에 대해서 CRAY 아키텍처보다 우수한 처리 속도를 보여주었다.

근래의 컴퓨터 발전 동향을 보면 범용적인 시스템 환경에서 다양한 멀티미디어 응용 프로그램들이 수행되고 있으며 주어진 작업을 처리하기 위해서 대용량의 프로세싱 파워를 요구 하고 있다. 이러한 주변 환경을 고려할 때, 본 논문에서 제안하는 방법은 기존 프로세서에 VPU를 추가하는 방법을 통해서 높은 프로세싱 파워를 제공해줄 수 있으므로 고가의 장비를 대체할 수 있는 방법으로 사용될 수 있다. 향후 멀티프로세서 환경에서 부착 사용할 수 있는 벡터 처리 전용 프로세서에 대한 연구를 수행할 예정이며 이를 통해서 병렬 처리 시스템이나 실시간 시스템 등에서 활용할 수 있는 모델을 제시할 계획이다.

참고문헌

[1] R.M.Russell, "The CRAY-I Computer System," Cray Research, 1986.

[2] K.L.Chung, W.M.Yan, "Fast Vectorization for Calculating a Moving Sum," IEEE Trans. on Comput., vol.44. Nov. 1995.

[3] Yuan Lin, Nadav Baron, Hyunseok Lee, Scott Mahlke, and Trevor Mudge,"A Programmable Vector Coprocessor Architecture for Wireless Applications", Proc. 3rd Workshop on Application Specific Processors, Sep. 2004.

[4] C. Kozyrakis, J. Gebis, D Martin, et al. ,"VIRAM: A Media-oriented Vector Processor with Embedded DRAM," In the Conference Record of the Hot Chips XII Symposium, Palo Alto, CA, August 2000.

[5] C. Kozyrakis. "A Media-enhanced Vector Architecture for Embedded Memory Systems," Technical Report CSD-99-1059, Computer Science Division, University of California at Berkeley, 1999.

[6] C. Kozyrakis. Scalable Vector Media-processors for Embedded Systems. PhD thesis, University of California at Berkeley, 2002.

[7] C. Kozyrakis and D. Patterson. "Vector vs. superscalar and vliw architectures for embedded multimedia benchmarks," In MICRO, Nov. 2002.

[8] D.H.Bailey, "Vector Computer Memory Bank Contention," IEEE Trans. on Comput., vol. c-36, No.3, Mar.1987. pp.293-297.

[9] A.L.Decegama, The Technology of Parallel Processing, Parallel Processing Architectures and VLSI Hardware vol. 1, Prentice-Hall Int.Editions 1989. pp.71-77, 166-177.