논문 02-27-4C-13 한국통신학회논문지 '02-4 Vol.27 No.4C

고성능 내장형 마이크로프로세서를 위한 SIMD-DSP/FPU의 설계

정회원 정 우 경*, 홍 인 표, 이 용 주, 이 용 석

Design of SIMD-DSP/FPU for a High-Performance Embedded Microprocessor

Woo-Kyeong Jeong*, In-Pyo Hong, Yong-Joo Lee, Yong-Surk Lee ***Regular Members***

요 약

본 논문에서는 고성능 내장형 프로세서에서 멀티미디어 성능을 효과적으로 향상시킬 수 있는 SIMD-DSP/FPU를 설계하였다. 하드웨어 증가를 최소화하기 위해 기존 연산기의 분할 구조를 제시하였고 면적이 작은 연산기를 제안하였다. 연산기의 공유를 통해 FPU의 하드웨어 면적을 크게 줄였다. 제안된 구조는 HDL로 모델링되고 0.35㎛ 표준 셀 공정으로 합성되어, 약 십만 등가 게이트의 면적을 갖는 것으로 보고되었으며 최악조건에서 코어 주파수인 50MHz 이상으로 동작하는 것이 예상된다.

ABSTRACT

We designed a SIMD-DSP/FPU that can efficiently improve multimedia processing performance when integrated into high-performance embedded microprocessors. We proposed partitioned architectures and new schemes for several functional units to reduce chip area. Sharing functional units reduces the area of FPU significantly. The proposed architecture is modeled in HDL and synthesized with a 0.35㎛ standard cell library. The chip area is estimated to be about 100,000 equivalent gates. The designed unit can run at higher than 50MHz clock frequency of CPU core under the worst-case operating conditions.

Ⅰ. 서 론

컴퓨팅 환경에서뿐만 아니라 일상 환경에서도 널리 확산되고 있는 멀티미디어 환경의 일반화는 마이크로프로세서의 성능 발전의 요구를 더욱 증폭시키고 있다. 워크스테이션이나 개인용 컴퓨터에 사용되는 고성능 범용 마이크로프로세서들은 이러한 요구에 맞추어 멀티미디어 처리 성능을 효율적으로 향상시킬 수 있는 확장 SIMD(Single-Instruction-Multiple-Data) 명령어와 이를 수행하는 SIMD 연산기를 구현하고 있다. 이러한 SIMD 가속기능은 일반적으로 하나의 레지스터에 짧은 정수 데이터 여러 개를 저장하여 동시에 병렬 연산하는 방법으로 구현되고 있으며, 짧은 정수 데이터로 이루어진 영상 신호나 저품질의 오디오 처리에 수 배 이상의 성능 향상을 가져온다.[1] 또한, 애플리케이션이 고급화됨에 따라 고품질의 오디오 처리나 정밀도를 요구하는 신호처리를 위해 부동소수점 데이터의 처리량도 증가하게 되었으며, 최근 급격히 증가하고 있는 3D 그래픽 환경의 지원은 단정도 부동소수점 데이터 처리 성능을 크게 요구하고 있다. 이에 따라, 부동소수점 연산기에서도 SIMD 연산기능을 지원하여, 동시에 여러 개의 단정도 부동소수점 연산을 수행할 수 있는 기능이 필수가 되고 있다. 이러한 SIMD 부동소수점 연산 기능은 AMD의 3DNow!를

* 연세대학교 전기전자공학과 프로세서 연구실 (jean@dubiki.yonsei.ac.kr)
논문번호 : 020125-0315, 접수일자 : 2002년 3월 15일
※본 연구는 시스템집적반도체기반기술개발사업 중 (주)에이디칩스의 위탁과제로 수행되었습니다.

필두로 Intel의 SSE, 모토롤라의 AltiVec 등 많은 고성능 마이크로프로세서에서 채택을 하고 있다.[2][3][4]

고성능 범용 마이크로프로세서에 비해 상대적으로 저가의 애플리케이션을 목표로 하는 내장형 마이크로프로세서는 낮은 가격을 위해 보다 저성능으로 개발되어 왔다. 가전기기들이 디지털화되고 모바일 컴퓨팅 환경이 크게 대두됨에 따라 내장형 프로세서들은 그 수요와 응용분야가 크게 확대되고 있으며, 성능 향상의 요구 또한 커지고 있다. 내장형 응용분야에서 성능향상의 요구를 증폭시키고 있는 것은 제어기능보다는 멀티미디어 처리 요구의 증가가 주 요인이 되고 있다. 많은 기능을 단일칩화 하여 시스템 비용을 줄여야 하며, 동시에 다각화되는 멀티미디어 환경에 대응하여야 하는 내장형 마이크로프로세서에서 이러한 미디어 처리 성능과 고성능 DSP 기능은 더욱 필수적인 것이 되어가고 있다. 프로세서의 성능을 높이기 위해서는 동작 주파수를 높이거나 파이프라인의 폭을 넓히는 방법이 일반적으로 사용되고 있으나, 이러한 방법은 소비전력이나 설계의 복잡도를 증가시키면서 멀티미디어 처리 성능은 그다지 효율적으로 향상시키지 못한다. 따라서 내장형 프로세서에서도 SIMD 병렬 처리구조의 연산기를 구현하는 것이 효과적으로 멀티미디어 처리 성능을 향상시키는 방법이 될 수 있다. 그러나, 기존의 고성능 범용 마이크로프로세서에 적용된 SIMD 가속기능을 내장형 프로세서에 그대로 적용하기에는 하드웨어 오버헤드가 너무 크며, 또한 내장형 응용분야에는 적합하지 않은 단점을 가지고 있다. 따라서 본 논문에서는 내장형 프로세서에 적합하도록 적은 하드웨어 비용으로 멀티미디어 처리 성능을 효과적으로 향상시킬 수 있는 SIMD 연산기 구조를 제안하고자 한다.

SIMD 연산은 하나의 명령으로 여러 개의 데이터를 동일하게 처리하는 병렬처리이므로, 기본적으로 데이터의 개수만큼 연산기를 중복 구현하는 것이 필요하다. 그러나, 연산기의 중복은 큰 하드웨어 비용을 필요로 하므로, 가격에 민감한 내장형 프로세서에 적용하기는 어렵다. 본 논문에서는 내장형 프로세서에 적합하도록 하드웨어 증가를 최소화하기 위해, 각 연산기를 8/16/32/64 비트 단위로 연산할 수 있는 분할된 구조를 고안하여 설계하였고, SIMD-FPU에서 SIMD-DSP의 MAC 연산기를 공유함으로써 부동소수점 곱셈기와 나눗셈기의 면적을 크게 줄였다. 또한, 면적이 작은 새로운 스티키 비트 검출기 등의 회로를 설계하여 하드웨어 비용을 최소화한 DSP/FPU를 설계하였다. 본 논문에서 설계한 SIMD-DSP/FPU는 64비트 고성능 내장형 마이크로프로세서인 AE64000 EISC 프로세서와 함께 단일칩으로 구현될 예정이다.

Ⅱ. EISC AE64000 내장형 프로세서[5,6]

EISC는 Extensible Instruction Set Computing의 약자로, 고정 길이의 명령어 셋을 가지고 오퍼랜드 길이를 확장할 수 있는 구조를 뜻한다. AE64000은 EISC 아키텍처를 구현한 64-bit 데이터 버스와 주소 버스, 64-bit ALU를 가진 5단 파이프라인 스칼라 구조의 내장형 64-bit 마이크로프로세서이다. AE64000 코어는 16-bit의 고정 길이 명령어 셋을 가지며, 32-bit EISC 프로세서와 동일한 아키텍처와 호환성을 유지한다.

EISC 아키텍처는 LERI 명령어를 사용하여 즉치 오퍼랜드를 확장할 수 있으며, 특수한 명령어를 추가할 수도 있다. LERI 명령어는 한번 수행될 때마다 12-bit의 즉치 필드값을 확장레지스터에 확장해 두었다가, 확장레지스터를 사용하는 명령을 수행할 때 명령어에 포함된 즉치 오퍼랜드에 확장레지스터에 저장되어 있는 필드를 확장시켜 긴 오퍼랜드를 사용할 수 있도록 해 준다. LERI 명령어는 64-bit 마이크로프로세서에서 16-bit의 고정된 길이의 명령어 셋을 구현하는 것을 가능하게 하고 16-bit, 32-bit, 64-bit 등에서 코드 호환성을 가질 수 있도록 해 주며 프로그램 코드 크기를 크게 줄여주는 장점이 있다. 그러나, LERI 명령을 수행하는 데 소요되는 사이클은 성능을 저하시키는 요인이 될 수 있다. AE64000 마이크로프로세서에서는 이러한 단점을 줄이기 위해 IFU(Instruction Folding Unit)라는 유닛을 구현하였다. IFU는 사전에 페치된 명령어들을 검색하여 LERI 명령의 수행에 파이프라인 사이클을 소비하지 않고 확장 필드들을 연결시켜 주는 역할을 한다.

AE64000 마이크로프로세서는 4개의 보조 프로세서를 지원한다. CP0(Co-Processor 0)는 MMU와 캐쉬 메모리, 온칩 ICE를 내장하고 있다. 캐쉬 메모리는 2KB의 명령어 캐쉬와 4KB의 데이터 캐쉬로 구성되며, 모두 집합연관구조로 되어있다. 본 논문에서 설계한 SIMD-FPU 유닛은 AE64000 마이크로프로세서의 CP1으로 사용된다.

AE64000은 캐쉬 메모리를 제외하고 약 16만 등

가 게이트의 하드웨어 면적을 가지며, 프로세서 코어만으로는 약 5만 게이트의 작고 간단한 하드웨어로 구현된다. AE64000은 0.35㎛ 표준 셀 공정으로 구현하였을 때 최악조건시 약 50MHz의 동작주파수를 가질 것으로 예상된다.

Ⅲ. SIMD-DSP 유닛의 설계

SIMD-DSP 유닛은 AE64000 마이크로프로세서의 실행 유닛으로 포함되며, 121개의 확장 명령어를 수행하기 위해 별도의 명령어 디코더를 갖는다. 연산의 소스 오퍼랜드와 결과는 AE64000의 16개의 64-bit 범용레지스터를 사용해 저장한다. SIMD-DSP 유닛은 AE64000의 64-bit 범용 레지스터 파일에 저장된 여러 개의 짧은 정수 데이터를 병렬로 동일한 연산을 수행한다. SIMD-DSP 유닛에서 처리하는 데이터 형식은 그림 1과 같다. 8개의 8-bit 오퍼랜드, 4개의 16-bit 오퍼랜드, 2개의 32-bit 오퍼랜드를 동시에 처리하는 SIMD 연산 수행이 가능하다. 예를 들면, 오디오 데이터는 일반적으로 16-bit의 정수 형식을 갖지만, MP3, AC-3, AAC 등의 고충실도의 오디오 처리는 20-bit 길이의 데이터를 사용할 것을 권장하고 있다. AE64000에서는 16/20-bit의 오디오 데이터를 32-bit의 정수 데이터로 변환하여 처리한다. Packed 32 형식은 두 개의 32-bit 오디오 데이터를 병렬로 처리하는 것을 가능하게 한다. 이와 같은 SIMD 연산은 연산기의 중복이 필요하나, 본 논문에서는 하드웨어의 증가를 줄이기 위해 64-bit의 데이터 패스를 분할하여 작은 크기의 정수 데이터를 여러 개 처리할 수 있도록 설계하였다.

SIMD-DSP 유닛의 전체 구조는 그림 2와 같다. SIMD-DSP 유닛은 다음과 같은 부분들로 구성되어 있다. 명령어 디코더, SIMD-AU 연산기, SIMD-MAC 연산기, SIMD-쉬프트 연산기, DSP 메모리, 기타 연산기 등이다. 명령어 디코더는 AE64000 페치 유닛에서 보내진 DSP 확장 명령을 디코드한다. SIMD-DSP 유닛이 별도의 명령어 디코더를 가짐으

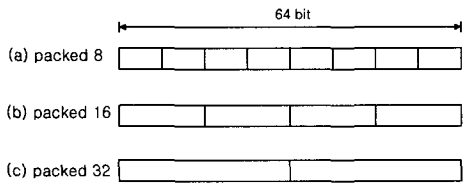

그림 1. SIMD 데이터 형식

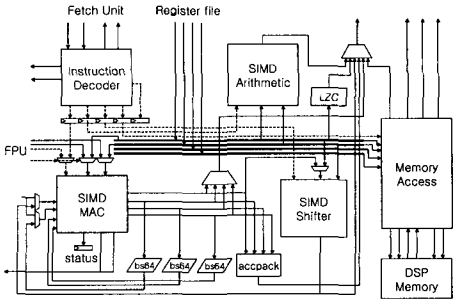

그림 2. SIMD-DSP 유닛의 구조

로써, SIMD-DSP 유닛의 탈부착을 가능하게 하여 내장형 프로세서의 다양한 제품군을 가능하게 할 수 있다. SIMD-AU 연산기는 가감산 등의 일반적인 SIMD 연산을 수행하고, SIMD-MAC 연산기는 DSP 알고리즘의 중심이 되는 MAC 연산을 빠르게 처리하기 위한 연산기이며, SIMD-SFT 연산기는 쉬프트 연산들과 데이터 재배치 명령들을 수행한다. DSP 메모리는 DSP 알고리즘에서 데이터를 빠르게 공급하기 위한 것으로, X/Y 메모리 각각 최대 8MB의 메모리를 내장할 수 있으며, 매 사이클 두 개의 인덱스 주소 계산과 두 개의 64bit 데이터의 액세스가 가능하여, DSP 알고리즘에 필요한 데이터를 높은 대역폭으로 공급할 수 있다. 기타 연산기들은 SIMD-MAC 유닛에 포함된 4개의 64-bit 누적 레지스터들의 쉬프트 연산이나 선행 0 카운팅 등의 명령어를 수행하기 위한 연산 유닛들이다. 각각의 유닛들은 모두 64-bit 데이터패스를 분할하여 하드웨어를 크게 증가시키지 않고 SIMD 연산을 할 수 있도록 구현하였다.

1. SIMD-AU

SIMD-AU는 SIMD 데이터의 기본적인 가감산 연산을 수행하며, 정수 데이터의 DSP 연산에서 유용한 포화 연산을 지원하고 각종 비교 명령과 클램프 명령을 수행한다. 그림 3은 SIMD-AU의 구조를 보여준다. SIMD-AU는 기본적으로 64-bit의 가산기와 비교 결과 처리 및 포화 결과의 생성을 위한 로직으로 구성되어 있다. SIMD 연산 기능은 64-bit 가산기를 8-bit씩 분할하여 캐리의 전달을 제어함으로써 구현할 수 있으며, 이로 인한 하드웨어의 증가는 매우 경미하다. 64-bit 가산기를 8-bit 단위로 CSA(Carry Select Adder)로 구현함으로써 캐리의 제어도 용이할 뿐 아니라 고속의 연산이 가능하다.[7] 그림 4는 SIMD 가감산기의 구조를 보여준다.

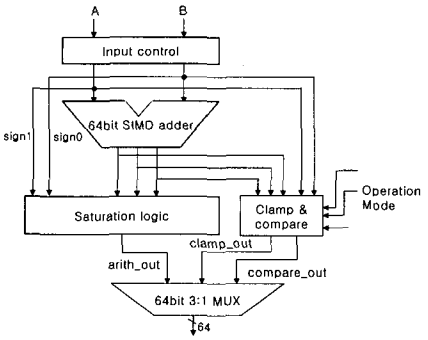

그림 3. SIMD-AU의 구조

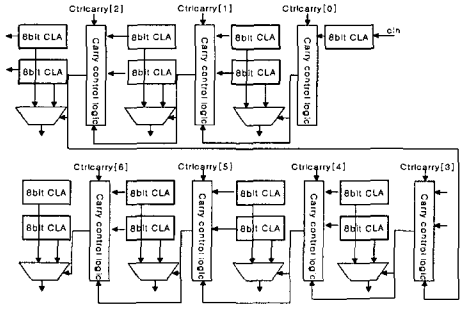

그림 4. SIMD 가감산기의 구조

2. SIMD-MAC[8]

SIMD-MAC 유닛은 2개의 32-bit MAC 유닛과 4개의 64-bit 누적 레지스터로 구성되며, 각 32-bit MAC 유닛은 2개의 16-bit MAC 연산을 수행할 수 있도록 설계되었다. 하나의 32-bit MAC 유닛에 2개의 누적 레지스터가 연결되어 있으며, 4개의 16-bit MAC 연산 수행시 각각의 곱셈 결과와 누적

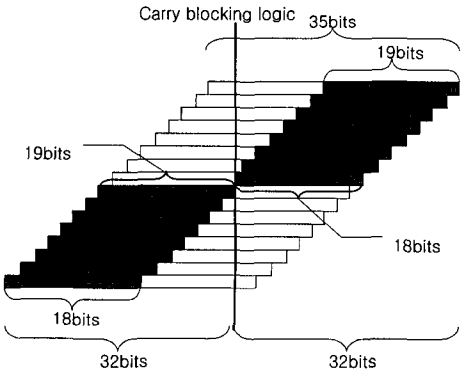

그림 5. SIMD-MAC 유닛의 월레스 트리의 분할 구조

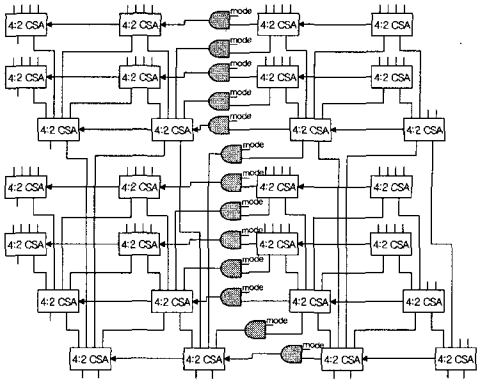

그림 6. 월레스 트리의 분할부

된 결과가 저장된다. 본 논문에서는 SIMD 연산으로 인한 하드웨어 증가를 최소화하기 위해 별도의 16-bit 곱셈기를 추가하지 않고 32-bit 곱셈기의 월레스(Wallace) 트리를 두 부분으로 분할하여 두 개의 독립적인 16-bit 곱셈을 수행할 수 있는 구조를 제안하였다. 그림 5는 월레스 트리의 분할 구조를 보여준다. 2개의 16-bit 곱셈 연산시 그림에서 회색으로 표시된 부분만이 사용되고 흰색 부분은 0으로 채워진다. 그리고 굵은 선으로 표시된 부분에서 캐리가 차단되어 각각의 16-bit 곱셈이 독립적으로 수행된다. 이를 위한 하드웨어의 증가는 그림 6에서와 같이 캐리 전달을 차단하는 AND 게이트 정도로 매우 경미한 것이다. 월레스 트리에서 출력된 캐리와 합 벡터는 누적 레지스터의 출력과 3-to-2 가산기를 통해 다시 두 개의 벡터로 만들어지며, 최종적으로 캐리가 전파되는 CSA 가산기를 통해 결과가 생성된다. 이와 같은 구조로, SIMD-MAC 유닛은 한 사이클에 2개의 32-bit MAC 연산, 또는 4개의 16-bit MAC 연산을 수행할 수 있다. 64-bit의 MAC 연산은 두 개의 32-bit MAC 유닛을 사용하여 세 사이클의 반복연산으로 수행할 수 있다. 이 연산은 부동소수점 연산기의 배정도 부동소수점 데이터의 승산/제산 연산을 위해 사용된다.

3. SIMD-SFT

SIMD-SFT 연산기는 8-bit, 16-bit, 32-bit, 64-bit 각 필드마다 별도의 쉬프트 값으로 쉬프트 연산을 수행하며, 64-bit 레지스터에 여러 개의 데이터를 집어넣거나 위치를 바꾸는 명령들을 수행한다. 이러한 연산은 기본적으로 위치를 바꿔주는 크로스 바 스위치 형태로 구현될 수 있다. 본 논문에서는 하나의 64-bit 쉬프터로 다양한 연산을 수행하기 위해 8개

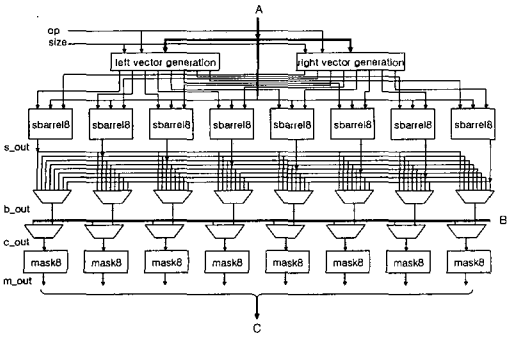

그림 7. SIMD-SFT의 구조

로 나눠질 수 있는 8-bit의 쉬프터와 8-bit 단위의 N by N 크로스 바 스위치로 구성된 구조를 제안하였다. 그림 7은 SIMD-SFT 연산기의 구조를 보여준다. 데이터는 8개의 필드로 나뉘어 각각 독립적인 쉬프트 양으로 좌우 쉬프팅을 할 수 있는 8-bit 배럴 쉬프터로 입력된다. 8-bit 쉬프팅된 결과는 8-bit 단위의 8-to-8 크로스바 네트워크를 통해 임의의 위치로 이동되며, 최종 마스크 회로를 통해 0이나 1의 결과가 되어야 할 부분을 마스킹한다. 이러한 구조로 다양한 크기의 필드를 쉬프팅하고 SIMD 데이터의 순서나 필드크기를 변경하는 다양한 명령의 수행이 가능하다.

4. DSP-MEM

DSP 알고리즘의 성능 향상을 위해서는 데이터를 연산 속도에 맞춰 공급할 수 있는 높은 메모리 대역폭이 필수적이다. 기존의 확장된 SIMD 명령어 셋을 가지는 범용 마이크로프로세서에서는 데이터 캐쉬를 주로 사용하나, 멀티미디어 데이터나 DSP 처리의 특성상 데이터 양이 많고 한번 이용한 데이터를 다시 사용하지 않는 경우가 많아 캐쉬의 효용

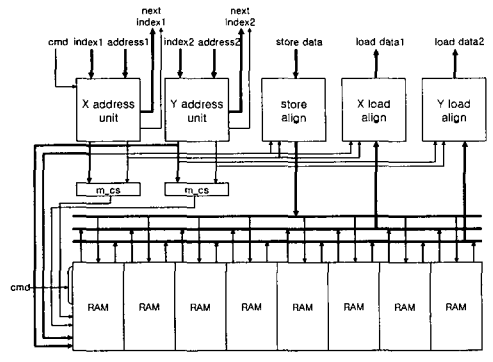

그림 8. SIMD-MEM의 구조

이 크게 떨어진다. 또한 데이터 캐쉬 방식은 실시간 응용에서 동작 시간을 보증할 수 없기 때문에, 내장형 마이크로프로세서에서는 동작을 정확히 예측할 수 있는 DSP 전용 메모리가 유용하다. 또한, 전용의 주소 계산 유닛을 둠으로써, X/Y 메모리의 인덱스를 자동으로 갱신하거나 원형 큐를 구현하는데 용이한 모듈로 주소 모드, FFT 알고리즘에서 유용한 비트 반전 주소 모드 등의 다양한 주소 모드를 지원하도록 구현하였다.

Ⅳ. SIMD-FPU 유닛의 설계

SIMD-FPU 유닛은 IEEE-754 호환의 단정도, 배정도 부동소수점 연산 및 2개의 단정도 부동소수점 데이터가 하나의 레지스터에 패킹된 SIMD-단정도 데이터의 연산을 수행할 수 있는 연산기로, AE64000 마이크로프로세서의 CP1 보조프로세서로 사용된다. SIMD-단정도 형식은 2개의 단정도 부동소수점 수의 연산을 동시에 병렬 수행함으로써, 여러 과학기술용 응용프로그램이나 3D 그래픽, 고품질의 신호처리 성능을 최대 2배까지 향상시킨다. SIMD-FPU는 배정도 부동소수점 연산의 데이터패스를 분할하여 SIMD-단정도 형식의 두 개의 단정도 데이터를 연산할 수 있도록 설계하였다. 부동소수점 연산기는 연산 과정이 복잡하고 많은 하드웨어를 차지한다. 본 논문에서는 부동소수점 연산기의 면적을 줄이기 위해 SIMD-DSP의 MAC유닛을 공유하도록 설계하였다. 실제로 정수 DSP 알고리즘과 부동소수점 알고리즘은 함께 사용되는 일이 드물기 때문에 이로 인한 성능 저하는 거의 없을 것이다.

그림 9는 SIMD-FPU의 구조이다. SIMD-FPU는 명령어 디코더, 레지스터 파일, FP-AU, FP-MUL, FP-DIV 유닛으로 구성된다. AE64000에서 페치된 명령어는 표준적인 보조프로세서 인터페이스를 통해 SIMD-FPU의 명령어 디코더로 전달되며, 15개의 64-bit 부동소수점 레지스터 파일에 저장된 오퍼랜드로 연산을 수행한다. FP-AU는 가감산, 형식 변환 등의 기본적인 연산들을 수행하고 FP-MUL은 SIMD-DSP 유닛의 MAC 유닛을 이용하여 곱셈 연산을 수행한다. SIMD-DIV 유닛도 SIMD-MAC 유닛을 공유하며, Goldschmidt 알고리즘을 사용하여 나눗셈을 수행한다.[9]

특히, SIMD-DIV 연산기는 단정도 부동소수점 나눗셈을 6 사이클의 짧은 지연시간으로 수행할 수 있는 고성능 나눗셈기이다.

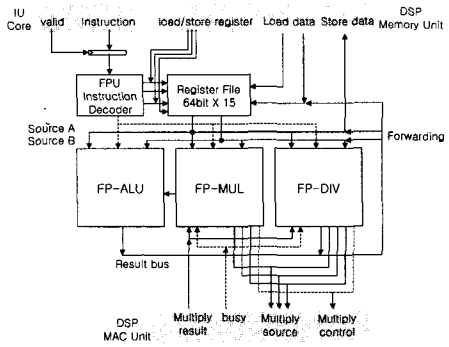

그림 9. SIMD-FPU의 구조

1. FP-AU

그림 10은 FP-AU의 구조를 보여준다. FP-AU는 3단의 파이프라인으로 구성되어 있으며, 매 사이클마다 연산 결과를 출력한다. 첫 번째 단에서는 지수의 감산과 가수의 정렬, 두 번째 단에서는 가수의 가감산과 선행 0 검출, 세 번째 단에서는 라운딩 및 정규화와 지수의 가산을 수행한다. FP-AU는 IEEE-754 배정도 부동소수점 데이터 연산을 위한 데이터 패스를 두 부분으로 분할하여 두 개의 단정도 부동소수점 연산을 동시에 수행할 수 있도록 설계하였다. 부동소수점의 가감산에서는 지수부와 가수부를 따로 계산하여야 한다. 가수부는 배정도 데이터의 53-bit의 데이터 패스를 분할하여 두 개의 단정도 데이터의 가수를 처리할 수 있으나, 지수부는 두 개의 단정도 데이터의 지수를 처리하기 위해 별도의 지수 연산부가 추가되어야 한다. 그러나 지수 연산부는 단정도 데이터의 경우 8-bit의 작은 데이터이므로 가수부에 비해 매우 크기가 작은 하드웨어이며 중복에 의한 하드웨어 증가는 그다지 크지 않다. 가수를 처리하는 연산부는 첫 번째 단의 정렬기, 두 번째 단의 가수 가산기, 세 번째 단의 라운딩 유닛과 정규화기가 있으며, 이 부분들이 FP-AU의 대부분의 하드웨어를 차지하고 있다.

두 번째 단의 가수 가산기와 세 번째 단의 라운딩 유닛은 중간에서 전달되는 캐리 입력을 제어함으로써, 53-bit 배정도 데이터 패스로 두 개의 단정도 가수를 처리하도록 설계하였다. 하드웨어 오버헤드는 거의 없으며, 지연시간에도 거의 영향을 미치지 않으면서 SIMD 연산 구현이 가능하다.

첫 번째 단의 정렬기와 세 번째 단의 정규화기는 배럴 쉬프터로 구현된다. 본 논문에서는 배정도 가수를 처리할 수 있는 배럴 쉬프터로 두 개의 단정도 가수를 독립적으로 쉬프트하기 위해 그림 11과 같이 쉬프터의 중앙이 분리된 쉬프터 구조를 제안하였다. 제어 신호에 따라 오른쪽 쉬프터로 입력되는 값이 왼쪽 쉬프터에서 쉬프팅 된 값으로 되거나 0으로 차단되어 왼쪽과 오른쪽 쉬프터가 서로 별도의 값으로 쉬프팅 될 수 있다. 쉬프터는 2-to-1 멀티플렉서로 구현하여 합성이 용이하고 지수 감산기의 지연시간을 숨길 수 있는 구조로 되어있다. 리플 캐리 가산기로 구현된 지수 감산기의 결과가 하위 비트부터 연속적인 지연시간을 가지고 출력되는 것을 이용하여 한 비트의 감산 결과가 나옴에 따라 2-to-1 멀티플렉서 한 단을 거치게 함으로써 지수 감산기의 지연시간을 완전히 숨길 수 있다. 일반적으로 8-to-1 멀티플렉서를 2단으로 구성하는 방법보다 하드웨어 면적도 작고 팬-아웃도 작기 때문에 지연시간도 짧은 장점이 있다.

가수 정렬기에서 오른쪽으로 쉬프팅되어 나온 비트열은 실제 가수의 가감산에는 모두 필요하지는 않다. 정확한 라운딩을 위해 G, R 두 비트와 나머지 비트를 모두 OR 시킨 S 비트만 있으면 된다. 기존에 S 비트를 구하는 방법은 TZC(Trailing Zero

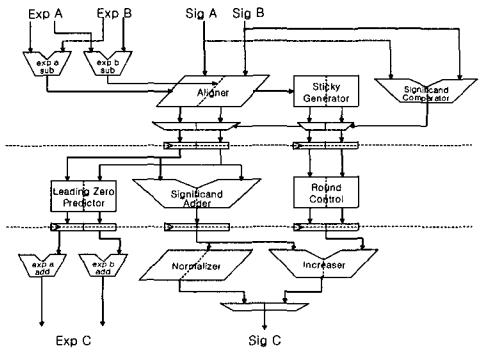

그림 10. FP-AU의 구조

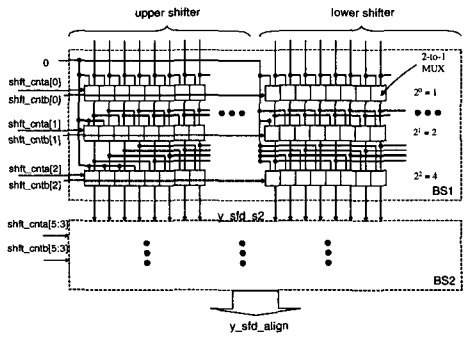

그림 11. SIMD 배럴 쉬프터의 구조

Counter)를 이용하는 방법과 마스크 방법이 있다. TZC 방법은 LSB로부터 연속된 0의 개수를 센 다음 지수 감산의 결과와 비교하여 S 비트를 결정하는 방법이다.[10] 마스크 방법은 지수 감산의 결과를 가지고 가수 각 비트에 해당하는 마스크를 발생시켜 가수 전체와 AND 연산을 함으로써 어떤 비트가 S 비트에 영향을 미치는지를 결정하여 S 비트를 구하는 방법이다.[11] 두 방법 모두 정렬기와 병렬로 동작하여 지연시간을 감출 수 있으나, TZC 회로나 53-bit의 마스크 발생 회로가 상당한 하드웨어 면적을 차지한다.

본 논문에서는 그림 12와 같은 2단 마스크 스티키 발생 회로를 제안하였다. 본 논문에서 제안한 2단 마스크 방법은 지수 감산기에서 출력된 6 비트의 쉬프트 값을 상위 3비트, 하위 3비트로 나누어 각각에 대한 마스크 비트를 구하는 것으로, 기존의 마스크 방식에 비해 약 1/5 정도의 하드웨어로 구현하는 것이 가능하다. 하위 3비트는 0부터 7까지의 쉬프트를 의미하는데, G, R 비트를 빼고 쉬프트 0을 고려하면 실제로 가수의 하위 5비트에 대한 마스크 제어 신호만을 발생시켜서 AND-OR를 적용하면 된다. 이러한 동작은 하위 3비트 쉬프트 값에 의해 배럴 쉬프터에서 가수가 정렬되는 것과 병렬로 수행된다. 배럴 쉬프터의 중간값은 다시 8의 배수로 쉬프트를 수행하는데, 따라서 오른쪽으로 쉬프트되어 나가는 비트열도 8비트 단위로 처리된다. 이것은 상위 3비트 쉬프트 값에 의해 결정되므로, 8비트 단위의 필드를 OR한 신호를 상위 3비트 쉬프트 카운터로 마스킹 방법을 적용할 수 있다. 최대 쉬프트는

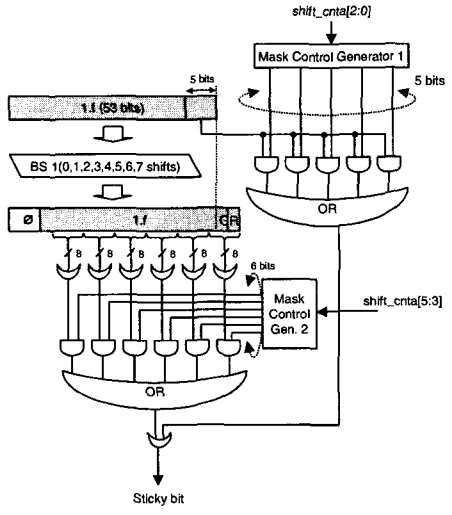

그림 12. 2단 스티키 발생 회로

배정도 연산시 53비트 이므로, 8비트씩 묶을 경우 6개의 신호가 만들어지고 따라서 6비트의 마스크 신호를 만들어 AND-OR를 적용하면 된다. 최종적으로, 하위 3비트 쉬프트 카운트로 얻어진 S 비트와 상위 3비트 쉬프트 카운트로 얻어진 S 비트를 OR 연산하면 최종 S 비트를 얻을 수 있다. 기존의 마스크 방법에서는 53비트의 마스크 발생회로가 필요한 반면, 제안된 2단 스티키 발생 회로는 하위 5비트, 상위 6비트의 마스크 발생회로만 있으면 되므로 약 1/5의 하드웨어로 구현할 수 있다.

2. FP-MUL

FP-MUL은 단정도와 배정도 부동소수점 데이터의 곱셈, 또는 두 개의 단정도 부동소수점 데이터의 곱셈을 수행한다. 부동소수점 데이터의 곱셈 연산에서는 지수의 가산과 가수의 곱셈이 필요한데, 가수의 곱셈기는 매우 큰 하드웨어를 차지한다. 본 논문에서는 SIMD-DSP의 MAC 유닛을 공유하도록 설계함으로써 가수 곱셈기의 하드웨어를 제거하였다. 또한 지수의 가산과 최종 라운딩은 FP-AU에서 처리하도록 함으로써, FP-MUL 유닛은 거의 하드웨어 면적을 차지하지 않으며, 전체 설계에서 하드웨어 면적을 크게 줄여주는 요인이 된다. 정수의 DSP 알고리즘과 부동소수점 데이터 연산 알고리즘은 보통 함께 사용되지 않기 때문에 MAC 유닛의 공유로 인한 성능 저하는 거의 없다고 할 수 있다.

3. FP-DIV

본 논문에서 설계한 부동소수점 나눗셈기는 Goldschmidt 알고리즘을 사용하여 단정도 부동소수점 수의 나눗셈의 경우 6 사이클이라는 짧은 지연시간을 갖는 고성능 나눗셈기이다. Goldschmidt 알고리즘을 구현하기 위해서는 곱셈기가 필요한데, 본 논문에서는 하드웨어를 줄이기 위해 SIMD-DSP의 MAC 유닛을 공유하였다. SIMD-MAC 유닛은 2개의 32-bit 곱셈을 매 사이클 수행할 수 있기 때문에, 두 번의 독립적인 곱셈의 반복 수행으로 몫을 구할 수 있는 Goldschmidt 알고리즘의 구현에 용이하다.

그림 13은 FP-DIV 유닛의 구조를 보여준다. SIMD-MAC 유닛을 제외하고는 초기 근사치 발생을 위한 ROM 테이블과 최종 라운딩을 위한 덧셈기 정도의 작은 하드웨어만을 가진다. 단정도 부동소수점 수의 나눗셈이 수행되는 과정은 다음과 같다. 첫 번째 사이클에서는 ROM 테이블을 읽어서 역수의 초기값을 구한다. 본 논문에서는 11Kbit

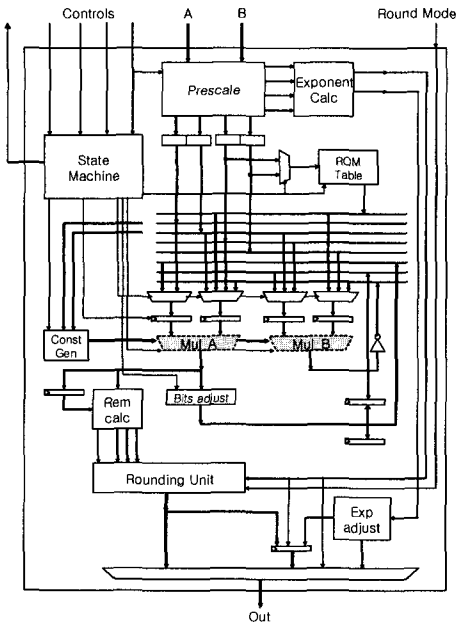

그림 13. FP-DIV 유닛의 구조

ROM 테이블에서 12 비트의 정밀도를 갖는 초기 근사치를 얻을 수 있도록 설계하였다. 2번째와 3번째 사이클은 원하는 정밀도를 얻을 때까지 곱셈 반복연산을 수행하는 과정이다. 4번째 사이클에서 원하는 정밀도의 몫이 구해지며, 5번째 사이클에서 나머지를 구해 6번째 사이클에 정확한 라운딩을 수행하게 된다. 배정도 나눗셈의 경우 더 많은 곱셈 반복연산이 필요하고 한번의 곱셈에 3사이클이 소요되며 두 번의 곱셈을 한꺼번에 수행할 수 없기 때문에 26사이클이 소요된다. 또한, SIMD-단정도 데이터의 나눗셈에는 9 사이클이 소요된다. 두 번의 단정도 나눗셈이 차례로 수행되는데, 첫 번째 나눗셈이 수행되는 동안 두 번째 나눗셈의 ROM 테이블을 읽을 수 있고, 하나의 곱셈만이 수행되는 몫 곱셈과 나머지 구하는 사이클에서 남는 곱셈기를 사용하여 두 번째 나눗셈의 반복 곱셈을 겹쳐 수행함으로써 사이클 수를 줄일 수 있다. 이러한 제어는 나눗셈기를 제어하는 상태천이머신에서 수행하며, 몇 개의 임시 저장 레지스터만을 추가함으로써 이러한 SIMD-단정도 데이터의 나눗셈 수행을 가능하도록 설계하였다.

V. 검증 및 합성 결과

본 논문에서 설계한 SIMD-DSP/FPU 구조는 Verilog HDL로 모델링 되었다. C 프로그램으로 만든 랜덤 벡터와 UCB FPU 테스트 셋을 사용하여 동작을 검증하였다. 검증된 SIMD-DSP/FPU HDL 모델은 AE64000 코어의 HDL 모델과 합쳐서 명령어 수준의 동작검증을 수행하였다. 검증된 HDL 모델은 0.35㎛ 표준 셀 공정을 사용하여 합성되었다.

표 1은 SIMD-DSP 유닛의 합성 결과이다. 면적은 Synopsys에서 보고된 배선 면적을 포함한 등가 게이트 수이며, 지연시간은 최악동작조건에서의 임계 경로의 딜레이를 ns 단위로 나타낸 것이다. 최악동작조건은 3.0V, 최악공정, 85℃로 설정하였다. 합성 결과, 가장 큰 하드웨어 면적을 차지하는 것은 SIMD-MAC 유닛으로, SIMD-DSP 유닛의 75%를 차지하여, 지연시간도 가장 긴 18.09ns를 나타낸다.

표 2는 SIMD-FPU 유닛의 합성결과이다. 마찬가지로 하드웨어 면적과 최악동작조건에서의 임계 경로의 지연시간을 나타내었다. FP-DIV의 합성 결과에서 괄호 안에 나타낸 숫자는 ROM 테이블의 크기를 로직과 구분하여 기록한 것이다. FP-AU가 32.2%로 가장 큰 면적을 차지하고, FP-REG가 28.0%로 다음으로 큰 면적을 차지한다. FP-DIV 유닛은 6 사이클의 짧은 지연시간과 SIMD 나눗셈 기능에도 불구하고 1만 2천 게이트 정도의 작은 면적으로 구현되었으며, FP-MUL 유닛은 SIMD-MAC

표 1. SIMD-DSP 유닛의 합성결과

	Area	Delay	(%)
Decoder	350	4.94ns	0.5
Arithmetic	2,957	15ns	4.3
MAC	51,105	18.09ns	75.0
Shifter	4,709	8.99ns	6.9
Misc.	5,941	7.46ns	8.7
Memory	3,096	14ns	4.5
Total	68,158	18.09ns	100

표 2. SIMD-FPU 유닛의 합성 결과

	Area	Delay	%
FP-ALU	12,860	8.88ns	32.2
FP-MUL	1,087	18.95ns	2.7
FP-DIV	12,252 (8,181+4,071)	19.27ns	20.7
Decoder	137	1.73ns	0.3
FP-REG	11,174	3.24ns	28.0
Total	39,928 (35,857+4,071)	19.27ns	100

유닛과 FP-AU 유닛의 공유로 1천 게이트만을 차지하였다. 가장 긴 지연시간을 가지는 것은 FP-DIV 유닛으로, SIMD-MAC 유닛의 최대 지연시간 18.09ns에 약간의 지연시간이 더 추가된 19.27ns를 나타내었다. 이것은 SIMD-DSP/FPU 유닛의 가장 긴 임계지연경로로서, 50MHz에서 동작이 예상되는 코어의 임계지연경로에 영향을 미치지 않는다.

Ⅵ. 결 론

본 논문에서는 고성능 내장형 마이크로프로세서 EISC AE64000에서 멀티미디어 및 DSP 처리 성능을 크게 향상시킬 수 있는 SIMD 병렬 연산 기능을 가진 DSP와 FPU를 설계하였다. SIMD-DSP/FPU를 구현하는 데 있어 하드웨어 면적의 증가를 최소로 하기 위해 본 논문에서는 기존의 데이터 패스를 분할하여 작은 데이터 여러 개를 효과적으로 처리할 수 있는 연산기 구조를 제시하였으며, DSP와 FPU 간에 연산기를 공유함으로써 FPU의 구현에서 하드웨어 면적을 크게 줄일 수 있었다. SIMD-DSP에서는 SIMD-AU, SIMD-MAC, SIMD-SFT 등의 연산 유닛의 분할 구조를 제시하였다. SIMD-FPU에서는 FP-AU의 분할 구조를 제시하고, 면적이 작은 새로운 스티키 비트 발생기 구조를 제안하였다. 또한, FP-MUL과 FP-DIV에서 SIMD-DSP의 MAC 유닛을 공유함으로써 하드웨어 면적을 크게 절약하였다. FP-DIV에는 Goldschmidt 알고리즘을 적용하여 단정도 데이터 나눗셈에서 6 사이클, SIMD-단정도의 나눗셈에서 9 사이클이라는 짧은 대기시간을 갖는 고성능 부동소수점 나눗셈기를 제안하였다.

제안한 구조를 HDL로 구현하고 0.35㎛ 표준 셀 공정으로 합성한 결과, SIMD-DSP는 68,158 등가 게이트, SIMD-FPU는 39,928 등가 게이트의 면적을 차지할 것으로 예상되며, 총 108,086 등가 게이트의 하드웨어 면적을 갖는 것으로 보고되었다. 회로의 임계 경로는 최악조건에서 19.27ns로, 코어의 동작 예상 주파수 50MHz에서 잘 동작할 것으로 예상된다.

본 연구에는 다음과 같은 추후 연구가 필요하다. 현재 SIMD-DSP의 MAC 유닛에 많은 기능이 집중되어, 매우 복잡한 설계가 되어 큰 하드웨어 면적과 긴 임계경로를 갖고 있다. 따라서, SIMD-MAC 유닛의 구조를 더욱 최적화하는 연구가 필요하며, 다른 유닛들의 지연시간과도 균형을 맞추는 노력이 필요하다. SIMD-MAC 유닛이 더욱 최적화 될 경우 더 작은 하드웨어와 짧은 임계 경로를 얻을 수 있을 것으로 예상된다. DSP 전용 메모리는 DSP 알고리즘 수행에는 유리하나, 일반 애플리케이션에서는 사용되지 않아 낭비되므로, 캐쉬 동작 모드에 따라 DSP 알고리즘을 수행하지 않을 때는 캐쉬 메모리로 활용하는 것도 좋은 방법이다. 동작 검증이 완료된 AE64000 코어와 SIMD-DSP/FPU는 동작 주파수를 향상시키기 위해 0.18㎛ 공정으로 재 합성한 후, 각종 주변장치들과 함께 SoC(System-on-Chip) 칩으로 구현될 예정이다.

참 고 문 헌

[1] P. Ranganathan, S. Adve, N. P. Jouppi, "Performance of Image and Video Processing with General-Purpose Processors and Media ISA Extensions," *Proc. of the 26th Ann. Int. Symp. on Comp. Arch.*, 27(2), pp. 124-135, May 1999.

[2] S. Oberman, G. Favor, F. Weber, "AMD 3DNow! Technology: Architecture and Implementations," *IEEE Micro*, 19(2), pp. 37-48, Mar. 1999.

[3] K. Diefendorff, "Pentium III = Pentium II + SSE," *Microprocessor Report*, pp. 6-11, Mar. 1999.

[4] M. S. Schmookler et al. "A Low-power, High-speed Implementation of a PowerPC™ Microprocessor Vector Extension," *Proc. of 14th IEEE Symp. on Comp. Arith.*, Apr. 1999.

[5] *EISC 64bit Microprocessor AE64000*, (주)에이디칩스, Jan. 20001.

[6] *EISC 64bit Microprocessor AE64000 Core Manual*, (주)에이디칩스, Jun. 2000.

[7] Israel Koren, *Computer Arithmetic Algorithms, Prentice Hall*, 1993.

[8] 홍인표, 정우경, 정재원, 이용석, "멀티미디어 데이터 처리에 적합한 SIMD MAC 연산기의 설계", *대한전자공학회 논문지 SD*, 38(12), pp. 890-901, 2001.

[9] R. E. Goldschmidt, "Applications of Division by Convergence," *MS thesis, Dept. of EE.*, Massachusetts Inst. of Technology, Cambridge, Mass., June 1964

[10] J. H. Edmondson et al, "Internal Organization

of the Alpha 21164, a 300-MHz 64-bit Quad-issue CMOS RISC Microprocessor," *Digital Technical Journal*, 7(1), pp. 119-135, Jan. 1995.

[11] N. Quach, M. Flynn, "Design and Implementation of the SNAP Floating-Point Adder," *Technical Report CSL-TR-91-501*, Comp. Sys. Lab., Stanford Univ., Dec. 1991.

정 우 경(Woo-kyeong Jeong) 정회원

1996년 2월: 연세대학교 전자공학과 졸업
1998년 2월: 연세대학교 전자공학과 석사
1998년 3월~현재: 연세대학교 전기전자공학과 박사과정

<주관심 분야> 반도체 설계, 마이크로프로세서, FPU, 멀티미디어

홍 인 표(In-pyo Hong) 정회원

1999년 2월: 연세대학교 전자공학과 졸업
2001년 2월: 연세대학교 전기전자공학과 석사
2001년 3월~현재: 연세대학교 전기전자공학과 박사과정

<주관심 분야> 반도체 설계, 마이크로프로세서, FPU, 멀티미디어

이 용 주(Yong-joo Lee) 정회원

1999년 8월 : 연세대학교 전자공학과 졸업
2001년 8월 : 연세대학교 전기전자공학과 석사
2001년 9월 ~ 현재 : 연세대학교 전기전자공학과 박사과정

<주관심 분야> 반도체 설계, 마이크로프로세서, FPU, 멀티미디어

이 용 석(Yong-surk Lee) 정회원

1973년: 연세대학교 전기공학과 졸업
1977년: University of Michigan, 전기공학과 석사
1981년: University of Michigan, 전기공학과 박사
1982년 ~ 1984년: Sperry-Univac Computer, 미국
1984년~1986년: Hyundai Electronics America, 미국
1986년~1988년: National Semiconductor, 미국
1988년~1989년: Performance Semiconductor, 미국
1989년~1992년: Intel, 미국
1993년~현재: 연세대학교 전기전자공학과 교수

<주관심 분야> 반도체 설계, 마이크로프로세서, FPU, 멀티미디어, 암호화 프로세서, 네트워크 프로세서