멀티미디어 응용을 위한 재구성가능 프로세서 설계

박진국, 곽기영, 이범근, 이두영, 정연모
경희대학교 전자공학과

Design of Reconfigurable Processor for Multimedia Application

Jin-Kuk Park, Ki-Yung Kwak, Beom-Geun Lee, Doo-Yung Yi, Yunmo Chung
Dept. of Electronic Eng., Kyunghee University

요 약

본 논문은 다양한 멀티미디어 응용을 위한 재구성가능(reconfigurable) 구조의 프로세서 설계에 대해서 연구하였다. 설계된 프로세서는 RISC 코어 프로세서와 코스-그레인(coarse-grain) 구조의 재구성가능 셀들의 배열로 이루어진 처리 유닛으로 구성되었다. 여기서 사용된 RISC 코어 프로세서는 하드웨어 구조를 간단히 하기 위하여 MIPS 명령어들 중에서 사용빈도가 높은 것들만 고려하였으며, 재구성가능 처리를 위한 별도의 명령어를 추가하였다. 본 논문에서 제시한 재구성가능 프로세서는 VHDL로 모델링하여 실행을 검증하였으며, 하드웨어의 유연성을 증가하여 다양한 멀티미디어 응용에 적용함과 아울러 속도향상에 기여함을 볼 수 있었다.

1. 서론

기존의 범용 프로세서 구조는 다목적용으로 적합하다. 그러나 특수한 목적의 다양한 연산을 필요로 하는 작업들에 대해서는 유연하게 사용되지 못하므로 특정 연산의 응용 부분에서는 적절한 성능을 발휘하지 못한다.

FPGA만을 이용한 재구성가능 시스템은 연산의 수행속도와 응용의 다양성 면에서 보다 효율적으로 사용될 수 있기 때문에 연산의 핵심부분이나 연산처리의 가속화를 위해서 구현되어 왔다. 그러나 코드의 많은 부분을 차지하는 대부분의 특정 기능의 연산은 실행 빈도가 낮으므로 이와 같은 기능을 FPGA 내에 모두 회로화 한다는 것은 비효율적이다. 또한 부동소수점 연산, 다양한 길이의 쉬프트 등과 같은 일반적인 연산분야에서는 프로세서에 내장된 연산부 보다 속도가 떨어진다.

이와 같은 문제점을 해결하기 위해 하나의 시스템 안에 재구성이 가능한 로직과 범용 프로세서를 결합한 프로세서-FPGA 시스템을 설계할 수 있다. 프로세서는 기본적인 알고리즘의 수행을 지원하고, FPGA의 재구성이 가능한 로직은 프로그램의 가장 중요한 연산부분의 수행능력을 향상시키기 위해 사용한다. 그러나 각각의 칩으로 분할 배치된 프로세서-FPGA 시스템은 연결에 따른 병목 현상과 지연을 초래한다. 이러한 문제는 프로세서와 재구성로직을 혼합하여 하나의 프로세서로 구현함으로서 해결할 수 있다[1][2].

본 논문에서는 32비트 RISC 코어 프로세서와 재구성이 가능한 로직을 하나의 프로세서에 구현하는 방식으로써 기존의 RISC 명령어와 호환이 가능하고, 연산과정이 보다 유연하고 고성능을 요구하는 시스템에 적합한 프로세서를 설계하였다.

2. 프로세서의 구조

본 논문에서 설계한 프로세서의 전체 블록은 그림 1과 같다. 32비트 RISC 코어 프로세서와 재구성가능 처리 유닛(reconfigurable processing unit)으로 구성된다. RISC 코어 프로세서는 일반적인 3단 파이프라인 구조를 가진다. 그림 1에서의 음영으로 표시된 재구성가능 처리 유닛은 재구성가능 셀 어레이 유닛(reconfigurable cell array unit)과 RC_control 유닛으로 구성된다. RC_control 유닛은 context 레지스터 파일, 데이터 패스 제어부, 버퍼 메모리로 구성된다.

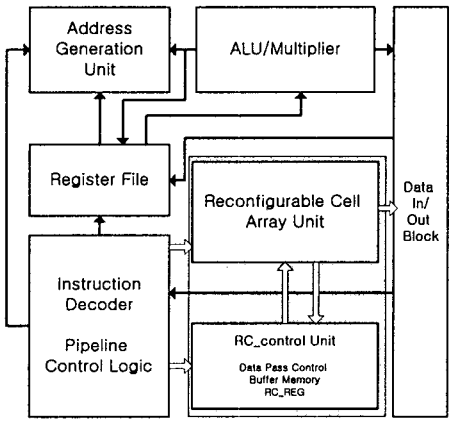

그림 1. 전체 블록도

재구성가능 처리 유닛은 RISC 프로세서의 실행 유닛처럼 동작하도록 함으로서 코프로세서를 이용한 방식보다 프로세서 및 메모리와 커뮤니케이션 오버헤드를 줄일 수 있다. 또한 재구성가능 프로세서를 위한 컴파일러는 기존의 RISC 컴파일러에 약간의 수정을 하여 구현할 수 있다. 하지만 이러한 방식은 코프로세서 방식에 비해 제한된 하드웨어의 크기를 가지는 단점이 있다.

2.1 RISC 코어 프로세서

RISC 코어는 페치, 디코더/실행, Write Back의 3단 파이프라인을 구조를 가진다. 본 논문에서는 RISC 코어 프로세서의 하드웨어 구조를 간단히 하기 위하여 MIPS 기반의 명령어 내에서 출현 빈도가 높은 명령어들을 사용하였다. 또한 재구성가능 처리 유닛의 제어를 위하여 재구성 명령어를 추가하였다. RISC 코어 프로세서의 내부 구조는 32비트 데이터 레지스터 뱅크, 곱셈기, ALU, 배럴쉬프터, 명령어 디코더, 파이프라인 제어로직 등으로 구성된다. 명령어 디코더는 전형적인 RISC 명령어 디코더와 재구성 명령어를 해석하고 재구성가능 처리 유닛을 제어하기 위한 디코더로 구성하였다.

2. 2 재구성가능 처리 유닛 구조

본 논문에서 사용된 재구성가능 처리 유닛의 구조는 재구성가능 셀 어레이 유닛과 이를 제어하는 RC_Control 유닛으로 구성되었다. RC_Control 유닛은 외부 메모리와 재구성가능 셀 어레이 사이의 데이터 패스를 제어해주는 데이터패스 유닛, 재구성가능 셀 어레이의 데이터를 저장하는 버퍼 메모리로 구성된다.

전통적으로 재구성가능 셀은 파인-그레인(fine-grain)과 코스-그레인 구조로 나누어진다[3,4,5]. MorphoSys에서는 코스-그레인 구조를 사용하여 완전한 ALU 형태의 재구성가능 셀을 제안하였다[6,7].

본 논문에서는 위의 방법을 참조하여 재구성가능 셀과 배열을 수정하여 기능을 향상하도록 설계하였다. 그림 2와 같이 서로 다른 기능을 가진 두 개의 셀을 코스-그레인 구조로 연결하여 하나의 재구성가능 셀로 재설계하였다. 왼쪽 셀을 L구조, 오른쪽 셀을 R구조라고 부르겠으며 각각의 특징은 다음과 같다. L구조는 오퍼랜드 A, B와 상수를 포함한 세 가지의 입력의 사용이 가능하다. ALU, 곱셈기, 쉬프터, 출력 레지스터, 플래그 레지스터, 결과 값을 임시 저장하는 레지스터 R0, R1로 구성된다. R구조는 L구조보다 간단한 구조이며 오퍼랜드 B와 상수 값을 입력으로 사용한다. ALU, 플래그 레지스터, 출력 레지스터, 결과 값을 행이나 열로 선택적으로 출력할 수 있는 3상태 버퍼로 구성된다.

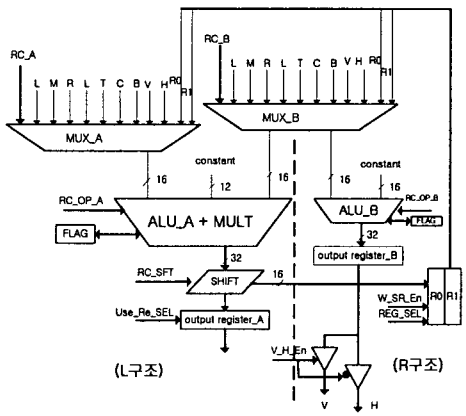

그림 2. 재구성가능 셀 구조

재구성가능 셀의 재구성 정보는 context word라고 칭하며 RC_Control 유닛 내에 있는 RC_REG 레지스터 파일에 32비트로 저장된다. context word는 그림 3 (a), (b)와 같이 L구조와 R구조를 위해 두 가지 형태로 저장된다. 각각의 비트별 정의는 다음과 같다. RC_OP_A는 L구조의 ALU_A+MULT의 함수, RC_OP_B는 R구조는 ALU_B의 함수를 설정한다. MUX_A와 MUX_B는 재구성가능 셀의 입력 오퍼랜드 선택을 위한 멀티플렉서의 제어 비트이다. RS_LS와 ALU_SFT는 L구조의 ALU_A+MULT에서 나온 결과 값을 쉬프트하기 위하여 쉬프트 방향과 쉬프트 양을 설정한다. REG_SEL

은 쉬프트된 데이터를 임시 저장 레지스터인 R0, R1의 어드레스 비트이고, W_SR_En은 임시 저장 레지스터의 쓰기 가능을 결정하는 비트이다. Use_Re_SEL은 L구조의 출력 레지스터의 제어 비트이고, V_H_En은 R구조의 출력 레지스터의 데이터를 V 또는 H로 출력할지를 결정한다. Constant는 임의의 상수 값을 연산하기 위하여 사용되어지며 L구조에는 12비트, R구조에서는 16비트의 크기를 가진다.

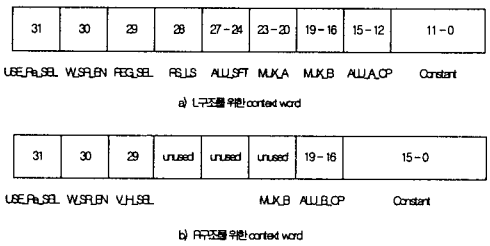

그림 3. RC_Reg Context Word

RC_REG 레지스터 파일은 그림 4와 같이 두 개의 파일 뱅크를 가진다. 뱅크 0은 L구조, 뱅크 1은 R구조의 context word를 저장한다. 각각의 뱅크는 context word를 저장하기 위해 16개의 레지스터로 구성된다. 모든 재구성가능 셀은 하나의 레지스터 파일을 가지며 RC_REG 레지스터 파일을 가지며 재구성 처리 유닛은 총 16개의 RC_REG 레지스터 파일을 가진다.

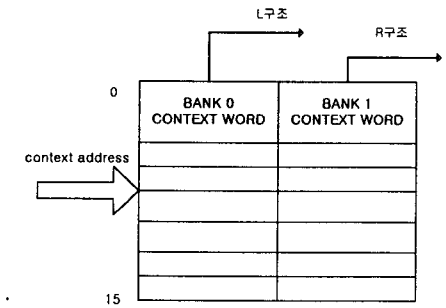

그림 4 RC_REG 레지스터 파일

표 1과 같이 context word의 ALU_OP_A, ALU_OP_B 값에 따라 L구조의 ALU_A+MULT, R구조의ALU_B에서 사용이 가능한 함수를 정의할 수 있다.

제안한 프로세서는 기존의 RISC구조의 명령어를 기본으로 하며 재구성가능 처리 유닛의 효율적인 실행을 위하여 새로운 재구성 명령어를 추가하였다. 재구성 명령어의 형식은 표 2와 같다. 추가된 재구성 명령어는 context word를 RC_REG 레지스터 파일에 저장하는 명령어, 일반 레지스터의 데이터를 오퍼랜드로 로드하는 명령어, 저장된 context word에 의해 재구성가능 셀 어레이들을 활성화시키는 명령어, 재구성가능 셀들의 결과 값을 메모리에 저장시키는 명령어 형식들로 정의할 수 있다.

표 1. 재구성 셀에서 이용 가능한 함수

RC_OP(A, B)	Function(A)	Function(B)
0000	Load Constant(C)	Load Constant(C)
0001	A or C	B or C
0010	A and C	B and C
0011	A xor C	B xor C
0100	A + C	B - C
0101	A - C	B + C
0111	A * C	not B
1000	Load A	Load B
1001	A or B	reserve
1010	A and B	reserve
1011	A xor B	reserve
1100	A + B	reserve
1101	A - B	reserve
1110	A * B	reserve
1111	Reset	Reset

표 2. 추가된 재구성 명령어 형식

명령어	동작
LDBCXT	context word를 RC_REG 레지스터 파일에 저장한다.
LDXMR	일반 레지스터 값을 재구성가능 셀의 오퍼랜드로 사용한다. .
STRXM	연산결과를 메모리에 저장한다.
RUNRC	재구성가능 셀 어레이를 활성화시킨다.

설계된 프로세서는 실행 유닛에 재구성 연산부를 설계하였기 때문에 제한된 하드웨어 크기를 가진다. 이를 고려하여 MorphoSys의 8×8 매트릭스를 기본 구조로 하여 데이터 버스와 오퍼랜드 버스를 수정하여 4×4 매트릭스 구조로 재설계하였다. 그림 5는 설계된 4×4 매트릭스의 연결구조이다. RC_0에서 RC_F까지의 셀 내부의 모든 L구조는 행과 열이 다른 모든 셀과 입·출력의 데이터 교환이 가능하며 R구조는 다른 모든 셀의 출력에 대한 입력은 가능하지만 출력은 같은 행이나 열로 출력이 가능하다.

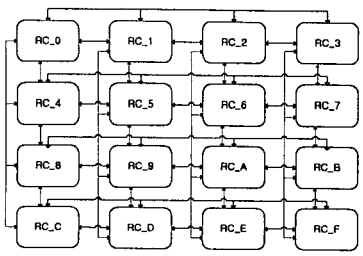

그림 5. 4×4 매트릭스 연결도

본 논문에서 사용한 4×4 매트릭스 구조는 두 개의 재구성가능 셀을 병렬로 연결하여 하나의 셀로 만들었으므로 실질적으로는 8×8 매트릭스와 같은 기능을 한다. 단, 재구성가능 셀 어레이 유닛의 크기를 고려하여 R구조를 ALU의 기본적인 함수만을 동작하도록 설계하였으므로 특정 응용 분야에서는 8×8 매트릭스보다는 유연성 및 처리 속도가 저하될 수 있으나 임베디드 사양을 고려할 경우에는 프로세서의 크기를 줄일 수 있는 장점이 있다.

3 구현 및 성능평가

VHDL를 사용하여 재구성가능 프로세서를 설계하여 합성 및 시뮬레이션을 하였다. 기존의 Morphosys의 8×8 매트릭스는 RISC 코어 프로세서의 11배정도의 크기를 필요로 하였다. 그러나 본 논문에서 사용된 4×4 매트릭스는 Synopsys 환경에서 Standard Library를 사용하여 합성한 결과는 RISC 코어 프로세서의 3배정도 크기만을 요구하였다. 동작검증은 프로그램 메모리, 데이터 메모리, 클럭 발생기 등이 구현된 가상의 시스템 상에서 수행이 하였다. RISC 명령어 수준의 호환성과 추가된 재구성 명령어의 동작 검증을 위해서 컴파일러와 어셈블러를 병행하여 기계어를 생성하였고, 생성된 기계어 코드를 프로그램 메모리에서 시뮬레이션을 하였다.

4 결론

본 논문에서는 RISC 코어 프로세서와 재구성가능 처리 유닛을 단일 프로세서 내부에 설계하였다. RISC 코어 프로세서는 하드웨어 구조를 단순화하기 위하여 사용빈도가 높은 명령어들만 고려하여 설계하였다. 재구성가능 처리 유닛은 매트릭스의 배열을 작게 하여 전체적인 하드웨어의 크기는 줄였다. 또한 두 가지 구조의 재구성가능 셀을 연결하여 하나의 기본단위의 재구성가능 셀로 사용함으로써 유연하게 연산 방법을 확장할 수 있었다. 설계한 프로세서는 일반목적 프로세서인 범용성과 특정응용 시스템에서 보다 유연한 시스템 설계가 가능하다. 특히, 재구성 프로세서의 사용은 이미지 처리, 패턴인식, 암호화/복호화 시스템과 같은 계산 집약적인 멀티미디어 응용 부분에서 보다 향상된 성능을 보일 것으로 사료된다.

[참고문헌]

[1] S. Hauck, Multi-FPGA Systems, Ph.D.Thesis, University of Washington, Dept. of Computer Science & Engineering, 1995.

[2] Darren C. Cronquist et al., "Mapping application to the rapid configurable architecture", *In Field-Programmable Custom Computing Machines (FCCM-97)*, 1997

[3] Sawitzki, S., Gratz, A., Spallek, R.G, " CoMPARE: A Simple Reconfigurable Processor Architecture Exploiting Instruction Level Parallelism", *Proceedings of the 5th Australasian Conference on Parallel and Real-Time Systems (PART98)*, pp.213-224, 1998.

[4] Elliot Waingold, et. al., "Baring it a Reconfigurable machines," *IEEE Computer.*

[5] S.C. Goldstein, et. al., "PipeRench : A Reconfigurable Architecture and Compiler," pp.70-77, 2000

[6] H.Singh, et. al., "Morphosys: Case study of a reconfigurable Architecture," *NATO Symposium on Systems Concepts and Integration*, Monterey, CA, 1998

[7] H. Singh, et. al., "Morphosys: A Parallel Reconfigurable System," *Euro-Par*, Toulouse, France, 1999.