내장형 신호처리를 위한 응용분야 전용 프로세서의 설계

이 성 원(李 晟 源), 최 훈(崔 薰), 박 인 철(朴 仁 哲)
한국과학기술원 전기 및 전자 공학과
전화 : (042) 869-4032 / 팩스 : (042) 869-4040

Design of An Application Specific Instruction-set Processor for Embedded DSP Applications

Sung-won Lee, Hoon Choi, In-Cheol Park
Department of Electrical Engineering Korea Advanced Institute of Science and Technology
E-mail : sungwon@fantasia.kaist.ac.kr, hchoi@ieee.org, icpark@ee.kaist. ac.kr

Abstract

This paper describes the design and implementation of an application specific instruction-set processor developed for embedded DSP applications. The instruction-set has an uniform size of 16 bits, and supports 3 types of instructions: Primitive, Complex, and Specific. To reduce code size and cycle count we introduce complex instructions that can be selected according to the application under consideration, which leads to 50% code size reduction maximally. The Processor has two independent data memories to double the data throughput and the address space. The processor is synthesized by 0.6 μm single-poly double-metal technology. Critical path simulation shows that the maximum frequency is 110MHz and total gate count is 132,000.

I. 서론

VLSI 기술의 발달로 인하여 다양한 시스템에서 ASIC(Application Specific Integrated Circuit)이 사용되고 있다. 일반 목적 프로세서(general purpose processors)와 비교하여 ASIC은 성능, 면적, 전력소모 등의 다양한 요구 사항들을 주어진 응용분야에 최적인 구조를 찾아 만족시킨다. 그러나 응용 분야가 복잡해짐에 따라 디자인의 오류와 스펙의 변화에 적절히 대응하기 위하여 더 많은 유연성(flexibility)이 요구되고 있다. ASIC은 하나의 응용목적만을 위하여 만들어지기 때문에 디자인 후기의 여러 문제를 수용하기 어렵다. 이에 반하여 프로그램이 가능한 프로세서는 프로그램의 교체만으로 충분하다. 이런 이유로 최근 ASIP(Application Specific Instruction-set Processor)이 많이 사용되고 있다.

복잡한 명령어(complex instruction)는 단순한 명령어에 비하여 프로세서의 수행 엔진들의 기능을 더 많이 사용할 수 있다는 점이 우수하다. 하지만 일반 목적의 프로세서에서는 복잡한 명령어가 널리 사용되지 않았는데 그 이유는 제공되는 복잡한 명령어가 주어진 응용분야에 효율적이고 강력하게 사용되기에는 너무 범용이기 때문이다. 이에 반해서 ASIP의 경우에는 복잡한 명령어가 ASIP이 대상으로 하고 있는 응용분야에 적합하게 최적화될 수 있기 때문에 유용하게 사용될 수 있다.[1,2]

응용분야에 최적화된 복잡한 명령어는 일반적으로 여러 사이클의 수행을 요구하게 된다. 이러한 다중 사이클의 명령어는 한 사이클의 명령에에 비하여 두 가지 중요한 장점을 가지고 있다. 먼저 프로그램의 크기(code size)를 줄일 수 있는데 이것은 내장형 시스템(embedded system)에서 매우 중요하다. 다음으로 프로그램 수행에 필요한 코드 패치의 수를 줄일 수 있다는 점이다. 즉 줄어든 코드 패치에서 사용되는 전력은 일반 프로세서에서 소모되는 전체 전력의 상당한 부분임이 지적되어 있다.[6]

이 논문에서는 디지털 신호처리 분야에 적용할 수 있는 ASIP의 명령어를 자동으로 합성해 주고, 시뮬레이션 할 수 있는 CAD 환경인 PARTITA의 코어 프로세서의 설계방법을 설명한다.

나머지 부분의 구성은 다음과 같다. 2절에서는 프로세서의 구조에 대해서 설명하고, 3절에서는 하드웨어 검증과 실험 결과에 대해서 다룬다. 마지막으로 4절에서는 결론과 추후의 과제에 대해서 논한다.

2. 프로세서의 구조

PARTITA ASIP 합성 시스템에서 사용되는 프로세서는 16비트 명령어 세트를 가지며, 128킬로 바이트의 코드 메모리, 각각 64킬로 바이트의 X, Y 데이터 메모리, 8개의 범용 레지스터와 8개의 주소 레지스터, 4개의 특수 목적 레지스터와 연산 유닛으로 구성되어 있다. 프로세서의 구조는 그림 1과 같이 파이프라인된 마이크로 프로그램 방식을 따르고 있다.

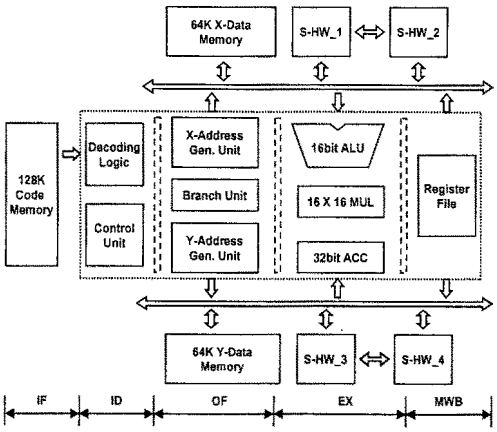

그림1. 프로세서 블록 다이어그램

전체 파이프라인은 코드 메모리로부터 수행할 명령어를 가져오는 IF 스테이지와 명령어를 디코딩하여 마이크로 롬으로부터 마이크로 코드를 가져오는 ID 스테이지, X, Y 데이터 메모리와 레지스터 파일로부터 오퍼랜드를 가져오는 OF 스테이지, 그리고 연산을 수행하는 EX 스테이지, 메모리와 레지스터 파일에 값을 기록하는 MWB 스테이지로 나뉘어져 있다. 코드 메모리와 데이터 메모리는 온 다이 되어 있는 메모리를 대상으로 하여, 한 사이클에 데이터를 읽고 쓸 수 있다고 가정하였다.

프로세서의 구조적인 특징은 다음과 같다. 첫 번째로 X와 Y 두개의 데이터 메모리를 두어서 별도의 복잡한 방법을 동원하지 않고서도 두 배의 어드레서블 공간을 확보하였으며, 데이터 오퍼랜드를 가져 올 때 두 배의 속도를 제공한다. 둘째, 제어 명령어가 실행될 때, 컴파일러가 항상 2개의 Delay slot을 제공하여 스톨이 발생하는 것을 방지한다. 셋째, 오퍼랜드를 패치 해 오는 과정에서 DSP 응용분야의 특성인 순차적인 데이터 접근을 효과적으로 수행하기 위해서 주소 생성 유닛(AGU)에서 선택적으로 값을 증가시킬 수 있다. 이는 3개의 명령어를 하나의 명령어로 대신하는 효과가 있다. 넷째, ALU와 독립적으로 곱셈기(Multiplier)와 누산기(Accumualtor)가 있어서 디지털 필터 구현에 많이 쓰이는 MAC연산을 한 사이클에 수행할 수 있다. 다섯째, 32bit의 연산 결과를 갖는 곱셈기와 누산기는 MPEG Layer3(MP3) 오디오나 H.261과 같은 멀티미디어 응용 분야의 다이나믹 레인지를 충분히 만족시킬 수 있다. 여섯째, 프로세서의 실시간 응용분야 적용을 위하여 하드웨어 인터럽트 핀을 제공하며, 인터럽트가 마스킹 되어 있지 않은 상황에서는 최장 두 사이클의 지연 후에 인터럽트 서비스 루틴을 실행한다.

프로세서는 모두 세 가지 클래스의 명령어를 수행할 수 있다. 먼저 P 클래스 명령어는 모든 응용분야에서 반드시 필요한 기본적인 명령어들로 구성되어 있다. ADD, SUB, AND, OR 등의 단순한 연산 명령어들과 JUMP, CALL 등의 제어 명령어가 포함된다. P 클래스 명령어 만을 수행하는 경우, 일반적인 RISC 프로세서와 같이 동작한다. 다음으로 C 클래스 명령어는 P클래스 보다 복잡하고 긴 명령어들의 집합이다. 이들은 복수의 P 클래스 명령어에 해당하며, 마이크로 코드의 적당한 조합으로 P클래스 명령어로는 불가능한 기능을 수행할 수도 있다.

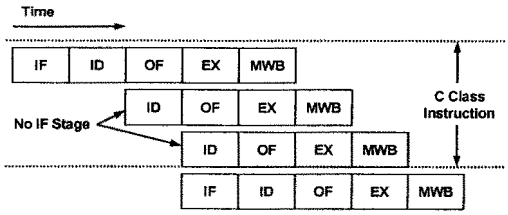

그림 2. C명령어의 파이프라인 진행

C 클래스 명령어는 16비트라는 명령어 길이의 제한으로 여러 사이클에 걸쳐서, 미리 정해져 있는 오퍼랜드만으로 수행을 해야만 하는 문제가 발생한다. 이 문제는 명령어에서 오퍼랜드 필드의 값을 부호화하고, 마이크로 롬에 해당하는 오퍼랜드를 저장하고, 디코딩 유닛에서 이 값을 구해내는 약간의 오버헤드 만으로 여러 개의 오퍼랜드를 제한된 수의 비트로 표현하였다.

P 클래스 명령어는 코드 상에서는 16 비트로 되어있으므로, 명령어 패치 회수와 코드 길이를 축소하는 효과가 있으며, 실제 프로세서 상에서는 그림 2와 같이 여러 개의 마이크로 코드로 디코딩 되어서 실행되어 멀티 사이클로 실행된다.

이러한 C 클래스 명령어는 PARTITA 시스템 내의 C 명령어 생성기 [3, 4, 5] 에 의해서 해당응용분야의 코드를 분석하여 최적화된 형태로 찾아지며, 프로세서는 C 명령어 생성기에서 넘겨주는 마이크로 코드의 정보와 부호화 정보를 이용해서 자동으로 디코딩 유닛을 재합성하고, 마이크로 롬의 내용을 변경하여 새로운 명령어를 수행하게 된다.

마지막으로 S 클래스는 특수한 하드웨어 가속 블록에 의하여 수행되는 명령어이다. 이 명령어는 하나 혹은 둘 이상의 하드웨어 블록으로 이루어지며, 프로세서는 이러한 블록에서 데이터가 처리되고 있는 동안 다른 명령어들을 처리하게 되므로, 전체 수행 속도나 코드 길이 면에서 최대의 이득을 얻게 된다. 특수 하드웨어와 프로세서는 그림 1과 같이 연결되고, 서로 데이터를 주고받기 위해서 특정 데이터 메모리 번지를 IO영역으로 할당하는 Memory mapped IO방식을 이용한다. 특수 하드웨어 사이의 데이터 교환이 필요한 경우에는 PARTITA 시스템에서 제공하는 동기식, 혹은 비동기식의 인터페이스 합성기가 제공하는 인터페이스를 이용한다.

3. 하드웨어 검증과 실험 결과

프로세서는 Verilog HDL을 이용해서 RTL 수준으로 기술하였다. 마이크로 롬은 편의상 PLA를 이용하여 구현하였다. 일차적으로 Cadence의 Verilog XL[7]을 이용하여 RTL Simulation을 하였고, C로 기술한 ISS(Instruction Set Simulator)의 결과와 비교하였다.

현대 0.6m 셀 라이브러리로 합성한 결과, 총 13,200 게이트, 동작 속도는 시뮬레이션 결과 110MHz로 측정되었다.

프로세서의 동작을 검증하기 위해서, Altera FLEX10K100ARC240-3 FPGA를 이용하였다. PC와 인터페이스 하기 위해서, 원래의 디자인을 약간 수정하였다. 즉 프로세서가 모드 0에서는 ISA 버스를 통해서 코드 메모리에 프로그램을 다운로드 하게 되고, PC에 의해서 모드 1이 되면, 프로그램을 수행하기 시작한다. 프로그램이 종료하면 모드 2로 전환되며, 이때 PC에 테스트 프로그램 종료를 알려주게 된다. 그러면 PC에서는 프로세서를 모드 0으로 바꾸고, 데이터 메모리의 값을 읽어오게 된다. 이 방법으로 C 컴파일러가 생성한 프로그램을 수행하였다.

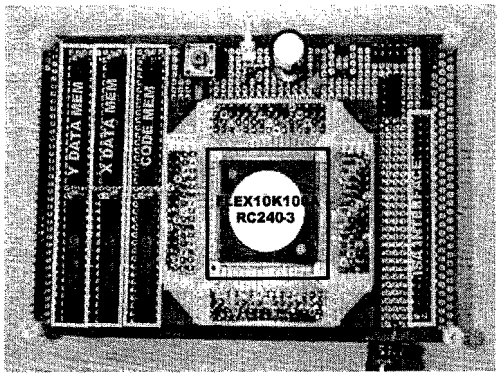

그림 3. FPGA 검증용 보드

프로세서는 기본적인 P 클래서 명령어와 C 클래스 명령어만으로 동작을 시켰으며, 응용분야의 C 프로그램에 적합한 C 클래스 명령어를 C 명령어 생성기로 생성하고, 이 정보를 이용하여, 앞서의 방법으로 프로세서의 구조를 변경하여 동작을 살펴보았으며, ISS와 Sun 워크스테이션 상에서의 실행결과와 비교하여 정확함을 확인하였다.
그림 4에서 보듯이 상용 DSP 프로세서와 비교하여, 사이클 수에서는 최고 60% 감소효과가 있었으며, 코드 사이즈 측면에서도 최고 50%의 감소효과가 나타났다.

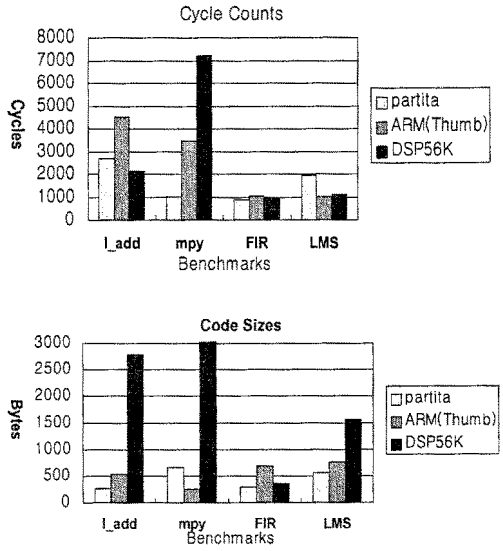

그림 4. 실험 결과

4. 결론 및 추후 과제

응용분야에 최적화된 명령어 세트를 갖는 프로세서인 ASIP을 디자인하였다. 여기에서는, 기존의 범용 프로세서 상에서 복잡한 명령어가 갖는 한계를 극복하고, 하드웨어를 응용분야가 바뀔 때마다 재 합성하는 방법을 통해서 유용한 C 클래스 명령어를 수행 가능하게 함으로써 사이클 수와 코드 길이측면에서 얻는 이득을 확인하였다. 이 점은 내장형 신호처리 부분에서는 제한된 코드 사이즈가 중요한 제약이라는 점을 생각해 볼 때 큰 장점이 된다. 또 명령어 패치의 회수 자체가 줄어들면서 얻는 전력 절약 효과도 덧붙여 얻을 수 있다.
추후 과제로는 특수 하드웨어를 적용하는 명령어인 S 클래스 명령어에 대해서 실험하고 H.261이나 MPEG 오디오 시스템에 대한 적용해 보는 일이 있겠다.

참고 문헌

[1] W. Zhao and C.A. Papachristou, “An Evolution Programming Approach on Multiple Behaviors for the Design of Application Specific Programmable Processors,” *European Design & Test Conference*, pp. 144-150, 1996.

[2] K. Kim, R. Karri and M. Potkonjak, “Synthesis of Application Specific Programmable Processors,” *34th Design Automation Conference*, pp. 353-358, 1997.

[3] J.H. Yi, H. Choi, I.C. Park, S.H. Hwang and C.M. Kyung, “Multiple Behavior Module Synthesis Based on Selective Groupings,”*Design, Automation and Test in Europe*, pp.384-388, 1998.

[4] H. Choi, I.C. Park, S.H. Hwang and C.M. Kyung, “Synthesis of Application Specific Instructions for Embedded DSP Software,” *International Conference on Computer-Aided Design*, pp.665-671, 1998.

[5] H. Choi, J.S. Kim, C.W. Yoon, I.C. Park, S.H. Hwang, and C.M. Kyung, “Synthesis of Application Specific Instructions for Embedded DSP Software,” *IEEE Trans. on Computers*, pp.603-614, Jun. 1999.

[6] M. T.-C. Lee, V. Tiwary, S. Malik and M. Fujita, “Power analysis and Minimization Techniques for Embedded DSP Software,” *IEEE Trans. on VLSI Systems*, pp. 123-135, Mar. 1997.

[7] Cadence Design Systems, “Verilog-XL Reference Mamual Vol. 1, 2,” *Cadence Design Systems*, Mar. 1991