개선된 Pipeline과 기능 블록을 가진 ARM7 Processor 설계

조현우, 허경철, 박주성
부산대학교 전자공학과
e-mail : hyun9673@pusan.ac.kr, kchua@pusan.ac.kr, juspark@pusan.ac.kr

An ARM7 Processor Design with Improved Pipeline and Function Blocks

Hyun-woo cho, Kyung-Chol Huh, Ju-Sung Park
Dept. of Electronics Engineering
Pusan National University

Abstract

In this paper, we present an improved design of the conventional ARM7 processor. It is based on the flip-flop to improve the pipeline performance of the processor. Also for improving the performance, the optimization of functional blocks and a multiplier is carried out. According to the experimental results, the maximum delay-time of functional blocks and the execution cycle of a multiplier is reduced by 33 % and 2 cycles compared with a conventional design, respectively. Therefore, it leads to improve an operation speed about 30%.

I. 서론

마이크로프로세서와 집적회로의 급속한 발전으로 인해 내장형 시스템의 응용범위는 빠르게 확장되고 있다. 이러한 내장형 시스템의 시장은 매년 급속도로 성장하고 있으며 성능 향상을 위해 다수의 코어를 사용한다 . 그리고 수행 속도를 개선하는 다중 프로세서 SoC (System on Chip) 시스템에 대한 연구가 활발히 진행되고 있다. 이러한 SoC 시스템에 내장 프로세서로 널리 사용되는 ARM사의 ARM7 프로세서는 저전력, 저면적, 고속 동작을 특징으로 하고 있지만 래치 기반 파이프라인(pipeline) 구조로 설계되어 내장형 프로세서 사용에 많은 어려움을 가지고 있다. 본 연구는 기존 ARM7 프로세서의 문제점을 분석하고 이를 바탕으로 설계 개선 대한 내용을 다룬다.

II. 본론

2.1 기존 ARM7 프로세서 취약점 분석

ARM7 프로세서는 3단 파이프라인, 37개의 범용 레지스터, 32bit ALU 및 32 x 8 형태의 MAC(Multiply-Accumulator)의 구조를 가지며 래치(Latch) 기반으로 설계된 내장형 32bit RISC 프로세서이다[1],[2]. 1차적으로 기존 ARM7 프로세서와 동일한 구조의 호환 프로세서를 설계하였으며 삼성 0.18㎛ CMOS 라이브러리를 이용하여 Synopsys사의 design complier을 통해 합성하였다. 설계된 프로세서는 래치 기반의 파이프라인 구조를 가지고 있음으로 인해 래치의 통과 특성과 clock skew등으로 인해 각 기능 블록에 대한 입력 데이터에 대한 timing 분배에 많은 어려움이 나타났다. 또한 칩으로 구현하기 위한 Back-end 설계에 많은 어려움이 있었다. 내장형 프로세서는 다양한 공정에 손쉽게 합성 가능해야 하므로 래치기반의 파이프라인 구조를 변경할 필요성이 나타났다. 동작속도 측면의 분석을 위해 기능 블록별 지연시간을 측정하여 표 1에 나타내었다. 프로세서의 동작에 있어서 데이터 주소 연산과 데이터 이동에 가장 많은 연산 시간이 소요된

다.[1] 설계된 프로세서는 한 사이클에 대한 데이터 순환은 레지스터 뱅크, 베럴 쉬프터, ALU, 레지스터 뱅크 순으로 순환한다. 표 1에서 보듯이 데이터 순환에 가장 많이 사용되고 있는 블록에서 많은 지연이 나타나고 있으며 최적화가 필요하다.

표 1. 기능블록 지연시간

기능블록	지연시간
프로세서 전체	18.21 ns
Control	2.27 ns
Instruction Decoder	4.26 ns
Register Bank	**4.73 ns**
ALU	**4.59 ns**
Barrel shifter	**3.78 ns**
Multiplier	5.04 ns

ARM7에 내장된 곱셈기는 곱셈 연산에 최대 7 사이클이 필요하며 곱셈 연산이 완료될 때까지 프로세서의 다음 명령어는 대기 상태에 있음으로 동작속도 향상을 위해 곱셈기 구조 변경이 필요하다.

2.2 프로세서 구조개선

본 논문에서는 기존 ARM7 구조에서 다음 3가지를 개선한다. 첫 번째로 래치 기반의 파이프라인 구조를 Flip-Flop (F/F) 파이프라인으로 변경하였다. 파이프라인 구조 변경을 위해 기능 블록 단위로 timing을 분석하여 래치를 F/F 구조로 변경하였다. 모든 래치와 F/F이 일대일로 전환 되는 것이 아니며 추가적인 조합회로가 더해지기도 했다. 둘째로 곱셈기의 성능개선을 위해 5가지의 곱셈기를 설계하였고 프로세서에 장착한 결과를 표 2.에 나타내었다. 최대지연시간은 모두 동일하므로 면적과 사이클 수가 성능을 나타내는 변수가 되었다. Radix-4 32×8 구조와 대비하여 27% 정도의 면적 증가로 최대 연산 사이클을 5사이클로 줄여 30%의 속도 향상을 나타낸 radix-4 32x16 구조의 곱셈기를 선택하였다.

표 2. 곱셈기 성능 비교

Multiply Method	Area(A) gate	Delay time/ Cycle(T/C)	Cycle (C)	A×T (Normalized)
Radix-4 32×8	7536	14.12ns	7	1.27
Radix-4 32×16	8327	14.12ns	5	1
Radix-4 32×32	10396	14.12ns	4	1.001
Radix-8 32×16	9147	14.12ns	5	1.1
Radix-8 32×32	10732	14.12ns	4	1.03

마지막으로 동작 속도 향상을 위해 레지스터 뱅크, 베럴 쉬프터, ALU를 중심으로 기능블록 별 최적화를 수행하였다. 기능블록 최적화와 곱셈기 구조 변경을 통해 지연시간을 줄일 수 있었으며 표 3에서 결과를 나타내었다.

표 3. 설계된 프로세서 지연시간 비교

기능블럭	래치기반 설계 지연시간 (L)	F/F 기반 설계 지연시간 (F)
프로세서 전체	18.21 ns	12.1 ns
Control	2.27 ns	3.02 ns
Instruction Decoder	4.26 ns	**4.02 ns**
Register Bank	4.73 ns	**3.63 ns**
ALU	4.59 ns	**4.7 ns**
Barrel shifter	3.78 ns	**2.75 ns**
Multiplier	5.04 ns	**4.09 ns**

2.3 설계 검증

설계된 프로세서의 테스트를 위해 레지스터 번호와 레지스터 초기값과 피연산자로 사용되는 상수 값 등에 다양한 값을 적용하기 위해 명령어를 그룹별로 임의추출하고 무작위로 생성된 피연산자를 결합하여 462개의 테스트 벡터를 생성하였다[4]. 설계된 코어에 ADPCM, MP3 디코더등의 응용프로그램을 수행시켜 명령어 조합에 의한 오류를 검증하였다. 모든 명령어들의 시뮬레이션 결과는 ARM7 상용 시뮬레이터의 결과와 비교 하였다.

Ⅲ. 결론

본 연구는 ARM7과 호환되는 프로세서를 설계하여 기존 프로세서의 문제점을 분석하고 성능을 개선하였다. 다양한 공정에 손쉽게 합성 가능하도록 파이프라인 구조를 F/F 기반으로 변경하였고 기능 블록 최적화와 곱셈기의 성능을 개선하였다. 기능 블록 최적화로 인해 최대 지연시간은 18.21 ns에서 12.1ns로 감소하였으며 곱셈기 성능 개선으로 인해 곱셈 사이클을 5 사이클로 줄여 전체 동작속도의 30% 향상을 가져왔다.

참고문헌

[1] Steve Furber, ARM System Architecture, Addison-Wesley, 1996

[2] ARM Ltd, ARM7TDMI Data Sheet (ARM DDI0029E), Advanced RISC Machines Ltd, 1995

[3] Dave Jagger, ARM Architectual Reference Manual, Prentice Hall, London

[4] Ta-Chung Chang, "A Biased Random Instruction Generation Environment for Architectural Verification of Pipelined Processor", in Journal of Electronic Testing : Theory and Application 16, pp.13~27, 2000