32-비트 RISC 마이크로 컨트롤러 설계

박 성 일, 최 병 윤
동의대학교 컴퓨터 공학과
02gm104@dongeui.ac.kr

Design for 32-비트 RISC Micro Controller

Sung-il Park, Byeong-yoon Choi
Dept. of Computer Engineering, Dongeui University
E-mail : 02gm104@dongeui.ac.kr

Abstract

This paper presents a 32-비트 RISC Micro-Controller which is useful in the dedicated DSP and communication areas. The designed processor has 5 stages pipeline architecture, and 28 instructions.

This RISC Micro-Controller consist of 22,100 gates and has 5.95 ns data arrival time, and 437 mW total dynamic power.

The RISC Micro-Controller is a IP (Intellectual property) Core module which can implement a number of protocols by and is applicable to DSP and data communication.

I. 서론

범용 마이크로프로세서를 구성하는 요소에는 명령어 세트, 레지스터, 메모리 공간 등이 있는데, 이들 중에 명령어세트(Instruction set)의 복잡도에 따라 RISC (Reduced instruction set computer)와 CISC (Complex instruction set computer)의 2가지로 크게 분류할 수 있다.

RISC 마이크로 컨트롤러는 작은 명령어수와 작은 면적으로 인하여 여러분야에 적용이 가능한데, 특히 통신분야의 Co-Processor나 그래픽 전용 콘트롤러, DSP (Digital Signal Processing) 콘트롤러등에 사용되고 있다[1][2].

본 논문에서는 RISC 프로세서의 설계 철학에 따라 간단한 기능의 명령어들과 모든 명령어들이 32-비트의 같은 크기를 가지도록 하였고, Opcodes와 Operand는 쉽게 디코드 되도록 하였다. 또한 모든 동작들은 레지스터들을 사용하며, 오직 메모리 명령어만 메모리를 접근하며, 파이프라인 특성으로 각 명령어는 단일 클럭 사이클에 수행 완료 될 수 있다.

본 논문에서는 TCP/IP 스택 프로세서에 사용가능하도록 32-비트 RISC 마이크로 콘트롤러를 설계하였고, Verilog HDL로 검증하였다. RISC 명령어 세트는 32-비트의 크기의 명령어로 구성되어지며, 속도와 안전성을 고려하여 5단 파이프라인의 구조를 가지고 있다[4].

II. 32-비트 RISC 마이크로 컨트롤러

2.1 32-비트 RISC 마이크로 컨트롤러의 시스템의 구성

32-비트 RISC 마이크로 컨트롤러는 제어분야 응용

을 고려하여 아래의 그림 1에서처럼 데이터 메모리와 명령어 메모리가 분리된 Harvard 구조로 되었다.

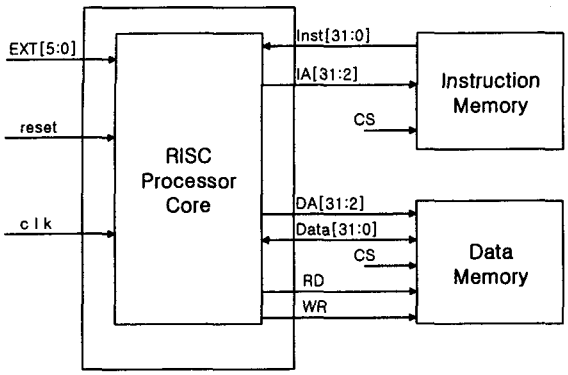

그림 1 32비트 RISC Processor Core

명령어 메모리는 롬 형태로서 프로세서의 사용목적에 따라 다양한 프로그래밍이 가능하다. 외부에서 명령어 세트의 포맷에 맞게 프로그래밍된 롬을 장착함으로써 다양한 동작이 가능하다.

설계된 RISC의 5단계 파이프라인 구조는 그림 2와 같으며 IF 단계에서는 명령어를 인출하고, ID단계에서는 오퍼랜드 인출 및 명령어를 디코딩을 하며, EX단계에서는 명령어 실행 및 Data 메모리 주소를 계산하게 된다. MEM 단계에서는 Memory Access를 하며, 마지막으로 WB 단계는 결과를 저장하는 단계이다.

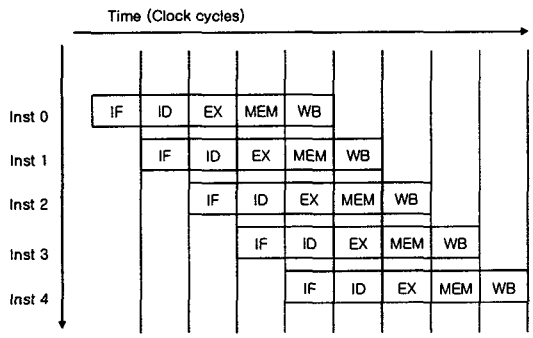

그림 2 RISC의 파이프라인 동작

2.2 32-비트 RISC의 명령어 셋

RISC 마이크로 컨트롤러의 명령어는 총 4개의 명령어 형식을 가진다.

메모리 타입 (Memory type), 분기타입 (Branch type), 연산 타입 (Compute type), 쉬프트타입 (Shift type) 으로 구성되며, 총 28개의 명령어로 구성된다.[3]

명령어의 형식은 그림 3 과 같다.

Memory type

31	29	26	25	20	15	0
00	OP	x	Src1	Dest	Offset(16)	

Branch type

31	29	26	25	20	15	0
00	cond	sq	Src1	Src2	Offset(16)	

Compute type

31	29	26	25	20	15	10	0
00	OP	I	Src1	Dest	Offset(16)		
				Src2	Dest	xxx	

Shift type

31	29	26	25	20	15	4	0
00	OP	x	Src1	Dest	xxx	shf(5)	

그림 3. 32-비트 RISC의 명령어 타입

명령어 세트는 설계 기준은 MIPS RISC를 참조하여 설계하였고, 워드 데이터(word data)만을 사용하며, MIPS RISC에서와 같이 내부 레지스터[0]은 항상 0의 값을 가지도록 하였으며, 분기 명령시 compute and branch 기법을 사용하였다. 호출명령(Call)과 복귀명령의 경우 내부 레지스터 RF[31]을 사용하게 된다. 분기 명령에서는 Squashed branch를 지원하도록 하였다[1][2].

각각의 명령어 타입별 어셈블리 문장(Assembly syntax)을 살펴보면 표 1과 같다.

표 1. 명령어 세트

명령어	어셈블리 표현
LD	ld rdest,offset(rs1)
ST	st rdest,offset(rs1)
BEQ	beq_sp rs1,rs2,label
BNE	bne_sq rs1,rs2,label
CALL	call label
RET	ret
BNZ0	bnz0_sq label
BNZ1	bnz1_sq label
BNZ2	bnz2_sq label
BA	ba label
ADD	add rdest,rs1,rs2
ADDI	addi rdest rs1,imm
SUB	sub rdest, rs1,rs2
SUBI	subi rdest rs1,imm
LUI	lui rdest, imm
SLT	slt rdest,rs1,rs2

SLTI	slti rdest,rs1,imm
AND	and rdest, rs1, rs2
ANDI	andi rdest,rs1,imm
ORI	ori rdest,rs1,imm
XOR	xor rdest rs1, rs2
XORI	xor rdest,rs1,imm
NOT	not rdest, rs1
LSR	lsr rs1,dest, shf
LSL	lsl rs1,dest, shf
ASR	asr rs1,dest, shf
ROR	rsr rs1,dest, shf

III. 32-비트 RISC의 설계

3.1 ALU 와 Shifter 의 설계

RISC의 기본적인 논리 및 산술 연산을 수행하기 위하여 ALU를 설계하였다. 또한 ALU에 사용된 덧셈기는 빠른 속도를 얻기 위하여 캐리 미리보기 덧셈기(Carry Look Ahead Adder)를 사용하였다.

설계에 사용된 베럴 Shifter는 Funnel 구조로서 1사이클에 multi-비트 shift가 가능하고 (signed/unsigned) left/right shift, rotate가 가능하다. ALU와 Shifter의 결합은 그림 4 와 같다.

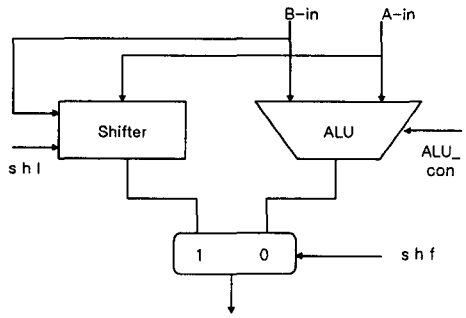

그림 4. ALU와 shifter의 분리 구조

ALU 내부의 구조는 는 또한 그림 5와 같은 구조를 가진다. ALU_con[3:0]의 신호에 따라서 Carry look ahead Adder의 내부에 있는 GP_gen 모듈의 동작을 결정하고 ALU_con[2] 신호에 따라서 산술 연산과 논리 연산을 수행하게 된다.

쉬프터의 구조는 그림 6에서 64 비트 입력을 받아 32비트 출력을 생성하는 32비트 Funnel Shifter 이다. Shf[4:0]에 정의된 크기 만큼 이동하며, OP 필드 값에 따라 LSR, LSL, ASR, ROR 동작을 하게 된다.

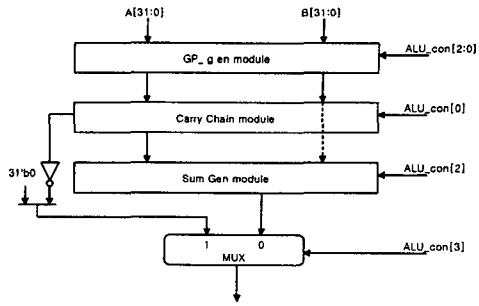

그림 5 ALU의 내부 구조

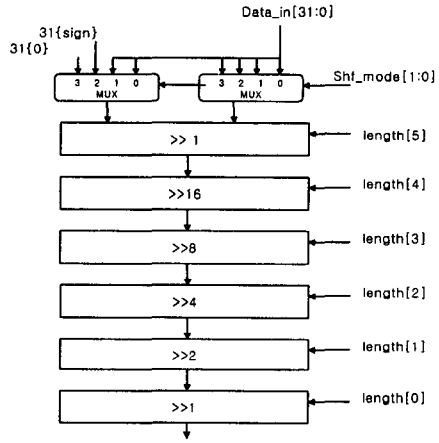

그림 6 Funnel Shifter의 구조

3.2 비교기 설계

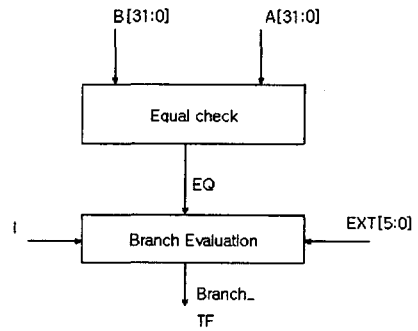

그림 7 분기 명령을 위한 비교기와 분기 평가 구조

분기 명령을 수행하기위하여 비교기를 이용한다.

그림 7에서와 같이 분기 명령은 rs1과 rs2의 값을 비교하여 외부 신호를 이용하여 분기 값의 참과 거짓을 판단하게 된다.

Ⅳ. RISC의 검증 및 성능 분석

32-비트 RISC 마이크로 컨트롤러의 설계와 검증은 Verilog HDL을 이용하여 수행하였으며 시뮬레이션 파형은 그림 8과 같다.

시뮬레이션은 NC Verilog의 Simvision를 이용하여 컴파일 및 검증 하였으며, 검증에 사용된 프로그램은 명령어 세트 형식에 맞추어 프로그래밍된 명령어를 명령어 메모리에 로드 후 수행하였다.

그림 8에서 ⓓ로 표시된 것은 데이터 메모리에서 데이터를 입출력하는 파형이며 ⓘ로 표시된 것은 명령어 메모리로부터 명령어를 받아 디코딩하기 위하여 명령어 메모리의 [31:26] 비트를 IR_OP에 할당한 파형이다.

파형에서와 같이 각각의 명령어의 수행은 매 클럭 사이클에 수행됨을 알 수 있다.

본 논문에 설계된 32-비트 RISC 마이크로 콘트롤러를 삼성 0.35um 표준셀과 Synopsys tool로 합성한 전기적 특성은 표 2.와 같다.

표 2. 설계된 RISC 마이크로 콘트롤러의 전기적특성

공정	삼성 0.35um
Cell library	std90
면적	22,100 gate
최대 지연시간	5.98ns (167MHz)
동작 전압	3.3V
소비전력	437mW

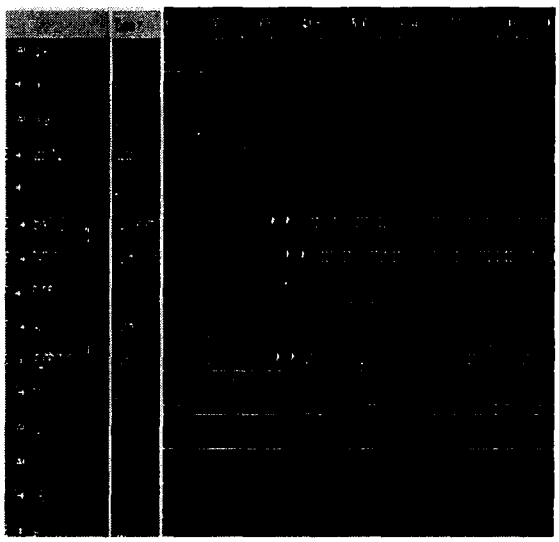

그림 8. 시뮬레이션 파형

Ⅴ. 결론

본 논문에서 설계된 RISC IP core는 내부 프로그램 메모리의 다양한 프로그래밍을 통해 다양한 분야에 적용 가능하다.

차후 연구 계획으로는 설계된 RISC 프로세서의 명령어를 생성하는 어셈블러를 컴파일러의 제작을 통하여 보다 쉽게 프로그래밍을 가능하도록 할 예정이다.

컴파일러는 명령어 세트의 형식에 맞추어 각각의 필드를 명령어 메모리에 입력하기 쉽게 비트형태로 변환시켜주게 되며, 각각의 필드를 유효성을 검사하도록 설계 할 예정이다.

감사의 글

본 논문은 회로 구현에 IDEC CAD software를 사용하였습니다.

참고문헌

[1] David A. Patterson and John L. Hennessy, "Computer Organization & Design" 1998. Morgan Kaufmann Publishers,

[2] 최 병 윤, "RISC 프로세서 구조",부산대학교 IDEC, 2001. 1.

[3] Mohamed Elbeshti "A RISC-based ATM Network Interface : processing, architecture, scalability and performance",2000. Computer Science, Acadia University.

[4] David Craig and Mark Pontius "The Zot1 Microprocessor A pipilined RISC processor implemented on an FPGA", 1994, UC Irvine.

[5] Samsung Corp. CMOS Standard Cell Library std90 datasheet, 2002.4.4, http://www.samsung.com