The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 19, No. 1, pp.239-245, Feb. 28, 2019. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2019.19.1.239
JIIBC 2019-1-32

RISC-V 프로세서의 모의실행 및 합성

Simulation and Synthesis of RISC-V Processor

이종복*

Jongbok Lee*

요 약 RISC-V는 프로세서의 혁신을 위하여 개방형 표준 협력을 통하여 개발된 무료이며 개방된 명령어집합 아키텍처 프로세서이다. 산업체와 학계의 협동으로 태동한 RISC-V는 프로세서 구조에 새로운 수준의 하드웨어 및 소프트웨어의 자유를 가져다주면서 확장 가능하기 때문에, 향후 50 년의 컴퓨터 설계와 혁신에 견인차 역할을 할 것으로 기대된다. 본 논문에서는 RISC-V가 개발되고 도입됨에 따라, 산술논리, 메모리, 분기, 제어 및 상태레지스터, 환경호출 및 중단점으로 구성된 명령어 아키텍처를 고찰하고 특징을 살펴보았다. 또한 Verilog를 이용하여 설계된 RISC-V 프로세서를 ModelSim으로 모의실행하고 Quartus-II로 합성한 결과, RISC-V의 38 개 명령어를 성공적으로 수행할 수 있었다.

Abstract RISC-V is a free and open ISA enabling a new era of processor innovation through open standard collaboration. Born in academia and research, RISC-V ISA delivers a new level of free, extensible software and hardware freedom on architecture, paving the way for the next 50 years of computing design and innovation. In this paper, according to the emergence of RISC-V architecture, we describe the RISC-V processor instruction set constituted by arithmetic logic, memory, branch, control, status register, environment call and break point instructions. Using ModelSim and Quartus-II, 38 instructions of RISC-V has been successfully simulated and synthesized.

Key Words : RISC-V, Verilog, ModelSim

Ⅰ. 서 론

RISC-V는 공개적 표준 공동 작업을 통해 프로세서의 혁신을 추구하는 무료 개방형 명령어집합 아키텍처(Insruction Set Architecture, ISA)이다. 비영리 법인인 RISC-V 재단은 2015 년에 설립되었으며, 100 개 이상의 회원 조직으로 발전하여 기술 혁신을 주도하고 있다. 학계와 연구계의 협력으로 탄생한 RISC-V 명령어집합 아키텍처는 향후 50 년간의 컴퓨팅 설계 혁신과 확장 가능한 소프트웨어 및 하드웨어의 자유를 제공할 것으로 기대된다[1].

RISC-V 재단의 회원은 RISC-V 명령어집합 아키텍처 사양 및 관련 하드웨어 및 소프트웨어로 구성되는 개발환경에 접근 및 참여할 수 있다. 본 재단에는 Bluespec, Inc., Google, Microsemi, NVIDIA, NXP, UC Berkely 및 Western Digital의 7 명의 대표로 구성된 이사회가 있다. 매년 RISC-V 재단은 확장 된 개발환경을 통합하여 현재 및 미래의 RISC-V 프로젝트 및 구현에 대해 논의하고 향후 명령 집합 아키텍처의 발전을 추진하기 위하여 글로벌 이벤트를 개최하고 있다. 이벤트 세션은 RISC-V

*정회원, 한성대학교 전자정보공학과
접수일자 : 2019년 1월 2일, 수정완료 : 2019년 2월 2일
게재확정일자 : 2019년 2월 8일

Received: 2 January, 2019 / Revised: 2 February, 2019 / Accepted: 8 February, 2019
*Corresponding Author: jblee@hansung.ac.kr
Dept of Electronics and Information Eng., Hansung University, Korea

아키텍처, 상용 및 오픈 소스 구현, 소프트웨어 및 실리콘, 벡터 및 보안, 애플리케이션 및 가속기, 시뮬레이션 인프라 등을 논의하며 선도적인 기술 기업 및 연구 기관을 그 대상으로 한다. RISC-V 명령어집합 아키텍처는 최초로 University of California, Berkeley 전기 전자 컴퓨터공학 부서에서 개발되었으며, 공개 표준 공동 작업을 통해 새로운 프로세서의 혁신을 함께 이루기 위해 노력을 경주하고 있다.

기존의 상용 명령어집합 구조를 이용하는 것에 대한 장점으로, 개발도구와 이식된 응용 프로그램을 비롯하여 광범위한 지원 소프트웨어의 개발환경을 연구 및 교육에 활용할 수 있으며, 또한 풍부한 문서와 교육예제를 확보할 수 있다는 점을 들 수 있다. 그러나, 상용 명령어집합을 연구와 교육에 이용할 때 이러한 잇점은 그다지 크지 않으며, 실제로는 단점을 능가하지 못하는 실정이다. 새로운 명령어집합 아키텍처를 개발하는 이유를 살펴보면 다음과 같다.

첫째, Intel, ARM, AMD 등과 같은 상용 명령어집합 아키텍처에는 소유권이 있다. SPARC V8 프로세서를 제외하고, 대부분의 상용 명령어집합 아키텍처의 소유권을 가진 자들은 지적재산권을 지키려고 노력하고 경쟁자가 이것을 무료로 구현하는 것을 환영하지 않는다. 비록 이것이 소프트웨어 시뮬레이터를 이용하는 학계 연구나 교육에서는 큰 문제가 되지 않더라도, 실제로 RTL 구현을 공유하고자 하는 그룹에서는 큰 문제로 대두되고 있다. 이것은 또한 상용 명령어집합 아키텍처 구현물을 신뢰하지 않고 자체적으로 구현하고 싶지만 법적으로 금지된 단체에게도 큰 문제이다.

둘째, 상용 명령어집합 아키텍처들은 특정 시장 분야에만 편중되어있다. 예를 들어 ARM 아키텍처는 아직까지 서버 시장을 제대로 지원하지 못하며, 인텔 X86 아키텍처는 이동 통신 시장에 취약하다. 또 다른 예로는 ARC와 Tensilica로, 확장 코어를 지원하고 있지만 임베디드 시장 분야만을 겨냥하는 한계를 안고 있다.

셋째, 상용 명령어집합 아키텍처는 명멸하고 있다. SPARC나 MIPS처럼 더 이상 수요가 없거나, Alpha처럼 더 이상 생산을 하고 있지 않은 상용 명령어집합 아키텍처가 좋은 예이다.

넷째, 현재 널리 쓰이는 상용 명령어집합 아키텍처는 매우 복잡하다. 인텔 x86과 ARM은 하드웨어로 구현하기에 모두 복잡하며 이것은 성능을 개선시키려는 특성보다는 비효율적인 낡은 명령어집합 설계를 채택한 것으로부터 기인한다. 그 밖에 기존 상용 명령어집합 프로세서들은 응용프로그램을 완전히 지원해주지 못하며, 확장 가능하도록 설계되지 않았고, 낡은 명령어집합을 함께 지니고 가야하는 부담을 항상 내포하고 있다.

새로운 명령어집합 프로세서는 컴퓨터시스템에서 가장 중요한 인터페이스이므로 이것이 소유권에 종속될 이유가 없다. 또한 기존의 상용 명령어집합 아키텍처는 30년 전에 만들어진 것으로 소프트웨어 개발자들은 이제 개방형 표준 하드웨어를 목표로 삼아야 하며, 상용 프로세서 설계자들도 구현의 고도화를 이루어야 한다[2,3].

본 논문은 다음과 같이 구성된다. 2 장에서 RISC-V 프로세서구조의 특징과 아울러 RISC-V 프로세서의 명령어집합 아키텍처를 고찰한다. 3 장에서 RISC-V의 하나인 RV32I 프로세서의 구조를 자세히 살펴보며, 4 장에서 모의실험 환경 및 결과를 보이고, 5 장에서 결론을 맺는다.

II. RISC-V 프로세서의 명령어집합 아키텍처

1. RISC-V 명령어집합 아키텍처의 개요

RISC-V 프로세서는 모든 하드웨어 구현에 반드시 나타나야할 기본 명령어집합 아키텍처와 이것에 대한 부가적인 선택및 확장으로 정의된다. RISC-V의 기본 정수형 명령어집합 아키텍처는 80년대 초기의 RISC 프로세서와 같으나, 분기 지연 슬롯이 없고 가변길이 명령어 암호화를 지원한다는 점이 다르다. 기본 명령어집합 아키텍처는 컴파일러, 어셈블러, 링커, 운영체제에 적합하도록 최소한의 제한된 명령어들로 구성된다.

기초 정수형 명령어는 정수형 레지스터의 넓이와 이에 해당하는 사용자 주소공간에 의하여 결정된다. 주요 정수형 명령어집합 아키텍처는 RV32I와 RV64I으로 나뉘며, 각각 32 비트와 64 비트의 사용자 수준 주소공간을 나타낸다. 그 밖에 RV32E는 RV32I의 변종으로서 소형 마이크로 컨트롤러를 지원한다. 기본 정수형 명령어집합은 부호를 갖는 정수값을 표현하기 위하여 2의 보수를 이용한다. RISC-V에서 주소공간의 크기로 32 비트를 채택한 이유는, 비록 대형시스템에서 64 비트의 주소공간이 필요하지만, 앞으로도 수십 년간 32 비트 주소공간이 임

베디드 및 클라이언트 장치에서 적절할 것이고, 낮은 메모리 트래픽과 저전력에 유리하기 때문이다. 또한, 32 비트 주소공간은 교육의 목적으로 충분하다. 그러나, 궁극적으로 128 비트 주소공간 역시 필요하기 때문에 RISC-V 명령어집합 아키텍처에 32, 64, 128 비트를 모두 수용 가능하도록 마련하였다.

RISC-V에서 더욱 일반적인 소프트웨어 개발을 지원하기 위하여 기본 정수형은 "I", 정수형 곱셈 및 나눗셈은 "M", 동기화는 "A", 단일 실수형 연산은 "F", 2배 정밀도 실수형 연산은 "D" 표준첨자를 써서 정의하였다.

2. 명령어 길이 부호화

기본 RISC-V 명령어집합 아키텍처는 고정된 32 비트 명령어를 가지며 32 비트 경계에 정렬되어야 한다. 그러나, 표준 RISC-V 부호화 방식은 가변 길이 명령어를 이용하여 명령어집합 아키텍처 확장을 지원하기 위하여 설계되었다. 따라서, 각 명령어는 16 비트의 몇 배가 될 수도 있다. 기본 RISC-V 명령어집합 아키텍처는 리틀 인디언 메모리 시스템을 가지나, 비표준 변형 방식에서는 빅 인디언이나 바이 인디언 메모리시스템도 지원할 수 있다. RISC-V에서 리틀 인디언 메모리 시스템을 채택한 이유는 현재 모든 x86 시스템, 안드로이드, ARM용 윈도우즈 시스템이 리틀 인디언 방식을 사용하기 때문이며, 이것이 하드웨어 설계자들에게도 자연스럽기 때문이다.

3. 예외적 사건, 트랩, 인터럽트

예외적 사건은 RISC-V 쓰레드 내의 명령어와 연관된 실행 시간에 발생한 비정상적인 상황을 의미한다. 트랩은 RISC-V 쓰레드 내에서 예외적인 조건에 의하여 제어가 트랩처리기로 동기적으로 전달된 것을 의미한다. 인터럽트는 RISC-V에 비동기적으로 발생시키는 외부 사건을 의미한다. 예외적 사건이 트랩으로 변환되는 것은 실행환경에 따라서 다르지만, 대부분의 경우 예외사건을 정밀트랩으로 연결시킨다.

III. RISC-V 기본 정수형 명령어집합 및 구조

1. RV32I 명령어집합의 개요

RV32I는 컴파일러와 최신 운영체제 환경을 지원하기 위하여 충분하도록 설계됨과 동시에, 최소 구현이 요구되는 환경에서 하드웨어를 줄이기 위하여 고안되었다. RV32I는 47 개의 명령어를 가지나, SCALL/SBREAK /CSRR과 같은 8 개의 특수명령어를 SYSTEM 명령어로 대체하고, FENCE 관련 2 개의 명령어를 NOP 명령어로 대체하여 38 개로 가능하다. 표 1에 RISC-V의 RV32I에 해당하는 38 개 명령어를 나타냈다.

표 1. RISC-V 프로세서의 명령어집합
Table 1. The instruction set of the Designed RISC-V processor

명령어 유형	명령어
산술 논리	ADD ADDI SUB SLTI SLTU AND ANDI OR ORI XOR XORI SLL SLLI SRL SRLI SRA SRAI LUI AUIPC NOP
메모리	LW LH LHU LB LBU SW SH SB FENCE
분기	JAL JALR BEQ BNE BL BGE
제어 및 상태 레지스터	SYSTEM
환경호출 및 중단점	ECALL BREAK

RISC-V는 32개의 범용레지스터로 구성되며, 레지스터0은 상수 0으로 하드와이어 되어있다. RV32의 경우 레지스터의 넓이는 32 비트이고 RV64의 경우 64 비트이다. 사용자가 이용가능한 또 하나의 레지스터는 프로그램 카운터 PC로서, 현재 명령어의 주소를 갖는다.

RISC-V의 명령어 포맷은 6 가지 유형인 R, I, S, B, U, J로 구성된다. 모든 명령어는 고정된 32 비트의 넓이를 가지며 메모리의 4 바이트 경계에 정렬되어야 한다. 명령어 주소 비정렬 예외사건은 타겟 주소가 4 바이트에 정렬되어있지 않을 때 취해진 분기나 무조건 점프에서 발생한다. RISC-V 명령어집합 아키텍처는 근원 레지스터 Rs1, Rs2와 결과 레지스터 Rd를 모든 포맷에서 동일한 위치에 유지하여 명령어 해독을 용이하게 한다. CSR 명령어에서의 5 비트 즉시피연산자를 제외하고, 즉시피연자들은 항상 부호확장되고 가능한한 가장 왼쪽으로 밀착하여 하드웨어 복잡도를 줄이도록 한다. 또한, 모든 즉시피연산자의 부호비트는 명령어의 31번째 비트에 위치하여 부호확장 회로가 고속으로 동작할 수 있게 한다. 레지스터를 해독하는 것은 구현시 임계경로에 해당하므로, 즉시피연산자 비트를 포맷에 따라서 분리하면서까지 모든 레지스터들을 동일한 위치에 오게 하였다. 즉시 피연

산자들은 모두 부호확장되는데, 이것은 MIPS에서처럼 0을 확장하는 것에 뚜렷한 잇점을 발견하지 못했기 때문이다.

피연산자를 취급하는 방법에 따라, B와 J의 명령어 포맷 변형 두 가지가 있다. S와 B 포맷의 유일한 차잇점은, B 포맷에서 즉시피연산자 값 영역 12 비트를 분기 오프셋을 부호화하는데 쓴다는 점이다. 하드웨어로 1 비트를 왼쪽으로 자리이동하는 기존의 방식과는 달리, 중간 비트와 부호비트를 고정해놓고 S 포맷의 최하비트로 B 포맷의 높은 자리의 비트를 부호화한다. 마찬가지로, U와 J 포맷의 유일한 차이점은 20 비트의 즉시피연산자를 왼쪽으로 12 비트 자리이동하여 U 즉시피연산자를 형성하고, 1 비트 자리이동하여 J 즉시피연산자를 형성한다는 것이다. U와 J 포맷에서의 명령어 비트의 위치는 서로 다른 포맷과 상호간의 겹침을 최대화하기 위하여 결정되었다.

2. 정수형 연산 명령어

정수형 연산 명령어들은 표 2에 나타낸 것과 같이 R-타입, I-타입, S-타입, B-타입, U-타입, J-타입으로 나뉘어지는데, I-타입 포맷은 레지스터 즉시 연산을 이용하고, R-타입 포맷은 레지스터 대 레지스터 연산을 이용한다. 연산의 결과는 두 포맷 모두 Rd 결과 레지스터이다. 정수형 연산 명령어는 예외사건을 일으키지 않는다.

표 2. RISC-V 기본 명령어 포맷
Table 2. RISC-V base instruction format

	31-25	24-20	19-15	14-12	11-7	6-0
R-type	funct7	Rs2	Rs1	funct3	Rd	Opcode
I-type	Imm		Rs1	funct3	Rd	Opcode
S-type	Imm	Rs2	Rs1	funct3	Imm	Opcode
B-type	Imm	Rs2	Rs1	funct3	Imm	Opcode
U-type	Imm				Rd	Opcode
J-type	Imm				Rd	Opcode

가. 정수형 레지스터-즉시피연산자 명령어

정수형 레지스터 즉시피연산자 명령어에는 ADDI, SLTI, ANDI, ORI, XORI가 있다. ADDI는 레지스터 Rs1에 12 비트 즉시피연산자를 더하는데, 이 때 연산 오버플로우는 무시한다. ADDI Rd, Rs1, 0 명령어는 MV Rd, Rs1 어셈블러 의사명령어를 구현하기 위하여 이용된다. SLTI 명령어는 레지스터 Rs1이 부호확장된 즉시피연산자보다 작을 때 레지스터 Rd에 값 1을 기록하고 그렇지 않은 경우에는 값 0을 기록한다. SLTIU는 이것과 유사하지만 부호가 없는 값들을 이용한다는 점이 다르다. ANDI, ORI, XORI는 레지스터 Rs1과 부호확장된 12 비트 즉시피연산자 사이의 비트별 논리연산을 수행하여 그 결과를 레지스터 Rd에 수록한다. XORI Rd, Rs1, -1 명령에 의하여 레지스터 Rs1에 대한 비트별 논리 역전을 수행할 수 있다.

상수값에 의한 자리이동 연산은 I-타입의 특별한 처리로 부호화한다. 자리이동의 대상인 피연산자는 Rs1에 두고, 자리이동하는 양은 I-즉시영역의 하위 5비트로 부호화한다. 오른쪽 자리이동은 I-즉시영역의 상위비트로 부호화한다. SLLI는 논리왼쪽이동으로 하위 비트에 0을 채우며, SRLI는 논리오른쪽이동으로 상위비트에 0을 채운다. SRAI는 산술오른쪽이동으로서, 부호비트를 복사하여 상위비트에 채운다.

LUI는 32 비트 상수를 만들기 위하여 이용되며 U-타입 포맷을 이용한다. LUI는 결과레지스터 Rd 상위 20 비트에 U-즉시 값을 넣고 하위 12 비트를 0으로 채운다. AUIPC는 프로그램 카운터 상대 주소를 만들기 위하여 이용하며 U-타입 포맷을 이용한다. AUIPC는 20 비트 U-즉시값으로 32 비트 오프셋을 생성하며 하위 12 비트를 0으로 채우고, 이 오프셋을 프로그램 카운터에 더하여 결과를 Rd에 기록한다. AUIPC 명령어는 제어흐름 전달과 데이터 접근을 위하여 PC를 통한 임의의 오프셋 접근을 위한 2 개의 명령어를 지원한다. AUIPC와 JALR의 12 비트 즉시값의 조합으로 임의의 32 비트 프로그램 카운터 상대 주소로 제어를 전달할 수 있으며, 정규 로드와 스토어 명령어 내의 12 비트 즉시값으로 임의의 32 비트 프로그램 카운터 상대 데이터 주소를 접근할 수 있다.

나. 정수형 레지스터 대 레지스터 연산

RV32I는 다양한 산술 R-타입 연산을 정의한다. 모든 명령어들은 피연산자로 Rs1과 Rs2 레지스터를 읽으며 그 결과를 레지스터 Rd에 기록한다. 이 때, funct7과 funct3 영역이 연산의 형태를 선택한다. ADD와 SUB은 덧셈과 뺄셈을 수행하는데, 이 때 오버플로우는 무시되며 결과의 하위 비트들이 최종 결과로 씌여진다. SLT와 SLTU는 부호가 있거나 없는 수에 대한 비교를 수행하며, Rs1이 Rs2보다 작을 때 Rd에 1을 기록하고 그렇지 않을 때 0을 기록한다. AND, OR, XOR는 비트별 논리연산을 수행한다.

SLL, SRL, SRA는 레지스터 Rs1에 대하여 레지스터 Rs2의 하위 5 비트 값만큼 논리 오른쪽, 왼쪽 그리고 산술 자리이동을 수행한다. NOP 명령어는 아무 상태 변화를 일으키지 않고 단지 프로그램 카운터값을 전진시킨다.

다. 제어전달 명령어

RV32I는 무조건 분기와 조건 분기 두 가지 제어전달 명령어를 제공하며, 지연슬롯을 갖지 않는다. 무조건 분기의 하나인 JAL 명령어는 J-타입 포맷을 이용하는데, J-즉시피연산자는 2 바이트의 배수로 오프셋을 부호화한다. 이 오프셋은 부호확장되고 프로그램 카운터와 더하여 분기 타겟 주소를 생성한다. 따라서 JAL은 ±1MB 범위를 아우르며, 분기 다음의 명령어의 주소인 PC+4 값을 레지스터 Rd에 저장한다. 간접 무조건 분기 명령어인 JALR은 I-타입 부호화 방식을 이용한다. 이 때, 타겟 주소는 레지스터 Rs1에 12 비트 부호화된 I-즉시피연산자를 더하여 구하며, 분기 다음의 명령어의 주소인 PC+4 값을 레지스터 Rd에 저장하는 것은 JAL과 같다.

모든 분기명령어들은 B-타입 명령어 포맷을 이용한다. 12 비트 B-즉시피연산자를 이용하여 2의 배수로 오프셋을 부호화하고 프로그램 카운터의 현재 값에 더하여 목표 주소를 만들며, 조건 분기의 범위는 ±4KB 범위를 아우른다. 분기 명령어는 두 개의 레지스터를 비교하는데, BEQ와 BNE는 Rs1과 Rs2가 같거나 다를 때 각각 분기가 된다. BLT 명령어는 Rs1이 Rs2보다 작을 때 부호화되지 않은 비교를 통하여 분기를 택하며, BGE 명령어는 Rs1이 Rs2보다 크거나 같을 때 부호화되지 않은 비교를 통하여 분기를 택한다.

라. 로드 및 스토어 명령어

RV32I는 로스 스토어 구조로서, 로드 스토어 명령어들만 메모리를 접근하고 산술연산 명령어들은 CPU 레지스터에만 작용한다. RV32I는 32 비트 사용자 주소 공간을 제공하고 바이트 단위이며 리틀 인디언 방식을 쓴다. 이 때, 실행환경에 의하여 적법하게 접근할 수 있는 주소 공간의 영역이 정의된다. 로드와 스토어 명령어는 레지스터와 메모리 간에 데이터를 전달하는 기능을 수행한다. 로드 명령어는 I-타입 포맷으로 부호화되고 스토어 명령어는 S-타입 포맷으로 부호화된다. 유효바이트 주소는 레지스터 Rs1에 부호확장된 12 비트 오프셋을 더하여 얻는다. 로드 명령어는 메모리의 데이터 값을 레지스터 Rd에 복사하는 기능을, 스토어 명령어는 Rs2의 데이터 값을 메모리에 복사하는 기능을 수행한다.

LW 명령어는 메모리로부터 32 비트의 데이터 값을 읽어서 Rd에 적재한다. LH는 메모리로부터 16 비트 값을 적재하여 32 비트로 부호확장한 다음에 Rd에 저장한다. LB와 LBU 명령어는 8 비트 값에 대하여 위와 동일한 기능을 수행한다. SW, SH, SB 명령어는 레지스터 Rs2의 하위비트에서 각각 32 비트, 16 비트, 8 비트 값들을 메모리로 저장한다.

마. 메모리 모델

기본 RISC-V 명령어집합 아키텍처는 단일한 사용자 주소공간에서 실행하는 다수의 병행 쓰레드를 지원한다. 각 RISC-V 하드웨어 쓰레드는 사용자 레지스터 상태와 프로그램 카운터를 갖고 독립적인 순차 명령어 흐름을 실행한다. RISC-V 쓰레드들은 실행환경에 대한 호출이나 공유메모리 시스템을 이용한 통신으로 서로 동기화한다. RISC-V 쓰레드들은 또한 입출력 장치들과 직접적으로 상호작용하거나 입출력장치에 할당된 주소공간에 대한 로드나 스토어 명령어를 통하여 간접적으로 상호작용한다.

기본 RISC-V 명령어집합 아키텍처에서 각 쓰레드는 프로그램의 순서대로 순차실행할 때 각 메모리 연산을 관찰한다. 쓰레드 간의 메모리모델은 FENCE 명령어를 이용하여 서로 다른 쓰레드 간의 메모리 연산에 대한 순서를 보장한다.

바. 기타 명령어

SYSTEM 명령어를 이용하여 특수접근을 요구하는 시스템 접근 기능을 수행하며 I-타입 명령어 포맷을 이용하여 부호화한다. 이것은 두 가지 유형으로 분류되는데, 제어 및 상태 레지스터 (CSR)를 읽고 수정하고 쓰는 것과 기타 특수명령어를 일컫는다.

ECALL 명령어는 운영체제에 대한 요청을 할 때 이용된다. 이것을 위한 ABI(Advanced Binary Interface)는 어떻게 환경요청에 대한 파라미터를 전달시키는지 정의하며, 이것은 대개 정수형 레지스터화일에 존재한다. EBREAK 명령어는 디버거에 의하여 실행을 중단시키고 디버깅 환경으로 제어를 전달하기 위하여 이용된다[4-7].

3. RISC-V 프로세서 아키텍처의 블럭도

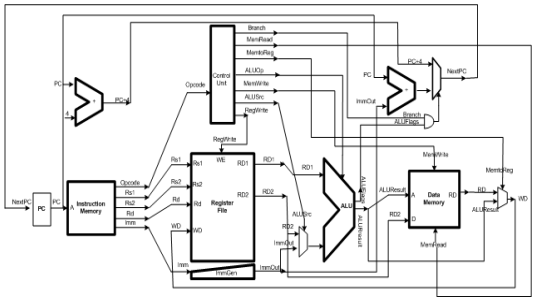

그림 1. RISC-V 아키텍처의 블럭도
Fig. 1. The RISC-V architecture block diagram

그림 1은 본 논문에서 채택한 RISC-V 프로세서 아키텍처의 블럭도이다. 제어부에 대한 입력은 명령어의 7 비트 Opcode이며, 제어부의 출력은 2 개의 멀티플렉서를 제어하기 위한 ALUSrc와 MemtoReg 제어신호, 레지스터화일과 데이터메모리에서의 읽기와 쓰기를 위한 RegWrite, MemRead, MemWrite 3 개의 제어신호로 구성된다. 또한 분기여부를 결정하는 1 비트인 Branch 제어신호와 ALU를 제어하기 위한 2 비트인 ALUOp가 이용되는데, Branch 제어신호와 ALU의 Zero 출력을 AND 게이트를 이용하여 다음 PC를 결정한다.

IV. 모의실험 환경 및 결과

본 논문의 모의실험은 운영체제 Fedora25에서 3.1GHz로 동작하는 Intel Core i5-2400에서 시행하였다. ModelSim은 Altera Starter Edition 10.3d 버전을 이용하였으며, Quartus-II는 13.0 버전을 이용하였다. RISC-V 중에서 최적화된 크기로 설계된 PicoRV32를 채택하였으며, 이것은 총 8 개의 독립적인 Verilog 프로그램 소오스 모듈로 구성되었다. 표 3은 모의실험에 사용된 RISC-V 어셈블리어 코드 프로그램이다.

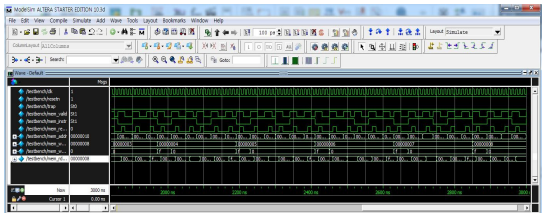

그림 2 RISC-V의 ModelSim 모의실행 파형 결과
Fig. 2. The ModelSim waves of RISC-V processor

그림 2에 RISC-V 프로세서가 모의실행되는 ModelSim의 결과 파형을 나타냈다. 본 모의실행에 이용한 RISC-V 어셈블리 프로그램은 LI, SW, LW, ADDI, J를 이용한 간단한 프로그램이다. 그 결과, RISC-V 프로세서가 제대로 동작하는 것을 확인할 수 있었다.

그림 3에 RISC-V를 Altera Quartus-II를 이용하여 합성한 결과를 보였다. 본 합성에 이용된 Altera FPGA는 Cyclone II의 EP2C50F672C8 소자이다. 합성 결과, 논리요소 1631 개, 조합회로 1593 개, 논리 레지스터가 601 개, 핀 409 개, 메모리 비트 2048 개가 할당되었다. 총 논리요소의 수는 1638 개로, FPGA 자원의 3 %만을 활용하므로, 하드웨어가 매우 간결함을 알 수가 있다.

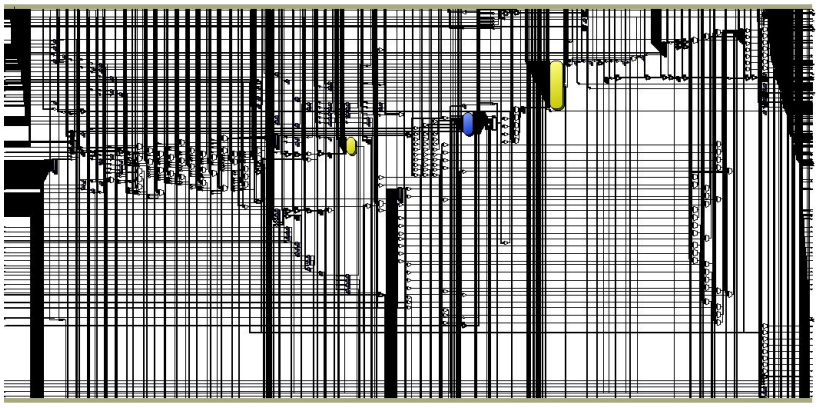

그림 3. RISC-V 프로세서의 합성 결과
Fig. 3. The Synthesized RISC-V Processor

V. 결론

본 논문에서는 ModelSim 환경에서 Verilog를 이용하여 38개의 명령어를 실행할 수 있는 RISC-V 프로세서를 모의실행시킨 결과, 올바르게 동작하는 것을 확인할 수 있었으며 Altera Quartus-II로 합성을 하였다.

추후로, RISC-V에서 제공하나 본 설계에서 누락된 파이프라인을 추가하고 인터럽트와 관련된 기타 명령어를 포함시키며, 설계된 RISC-V 프로세서를 타이밍 시뮬레이션을 거친 후에, 정적 시간 분석 (STA)과 합성 후 모의실험(Post synthesis simulation)을 거쳐 최종적인 동작을 검증하고 FPGA로 프로그래밍하여 동작을 검증할 예정이다. FPGA로 검증을 완료한 후에는, Synopsis로 합성하여 국내 기관인 IDEC을 통하여 ASIC 칩으로 구현할 예정이다.

References

[1] "The RISC-V Instruction Set Manual, Volume I: Base User-Level ISA," Andrew Waterman, Yunsup Lee, David Patterson, Krste Asanovic, Technical Report UCB/EECS-2011-62, EECS Department, University of California, Berkeley, May 2011.

[2] "The RISC-V Instruction Set," Andrew Waterman, Yunsup Lee, Rimas Avizienis, Henry Cook, David Patterson, Krste Asanovic, Poster at the Symposium on High Performance Chips, Stanford, CA, August 2013.

[3] "The RISC-V Instruction Set Manual, Volume I: User-Level ISA Version 2.0," Andrew Waterman, Yunsup Lee, David Patterson, Krste Asanovic, Technical Report UCB/EECS-2014-52, EECS Department, University of California, Berkeley, May 2014.

[4] "Instruction Sets Should Be Free: The Case For RISC-V," Krste Asanovic, David Patterson, Technical Report UCB/EECS-2014-146, EECS Department, University of California, Berkeley, August 2014.

[5] "A 45nm 1.3GHz 16.7 Double-Precision GFLOPS/W RISC-V Processor with Vector Accelerators," Yunsup Lee, Andrew Waterman, Rimas Avizienis, Henry Cook, Chen Sun, Vladimir Stojanovc, Krste Asanovic, European Solid-State Circuits Conference (ESSCIRC-2014), Venice, Italy, September 2014.

[6] "The Berkeley Out-of-Order Machine (BOOM): An Industry-Competitive, Synthesizable, Parameterized RISC-V Processor," Celio, Christopher and Patterson, David A. and Asanović, Krste, Technical Report No. UCB/EECS-2015-167, EECS Department, University of California, Berkeley, June 2015.

[7] "RISC-V Out-of-Order Data Conversion Co-Processor," A. Raveendran, V. Patil, V. Desalphine, P. M. Sobha, A. David Selvakumar, VLSI Design and Test (VDAT), 2015 19th International Symposium, Ahmedabad, India, June 2015.

저자 소개

이 종 복(정회원)

- 1964년 8월 20일생.
- 1988년 서울대 컴퓨터공학과 졸업.
- 1998년 동 대학 전기공학부 졸업 (공박).
- 1998~2000 LG반도체 선임연구원.
- 2000년~현재 한성대 전자정보공학과 교수

- Tel : 02-760-449
- Fax : 02-760-4435
- E-mail : jblee@hansung.ac.kr
- 관심분야 : 마이크로 프로세서, 멀티코어 프로세서, 텐서 프로세서 유닛

※ 본 연구는 한성대학교 교내학술연구비 지원과제임.