The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 23, No. 5, pp.115-121, Oct. 31, 2023. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2023.23.5.115
JIIBC 2023-5-18

RISC-V 프로세서의 FPGA 구현 및 검증

FPGA Implementation and Verification of RISC-V Processor

이종복*

Jongbok Lee*

요 약 RISC-V는 오픈소스 명령어집합 아키텍처로, 누구나 자유롭게 RISC-V 마이크로프로세서를 설계하고 구현할 수 있다. 본 논문에서는 RISC-V 아키텍처를 설계하고 시뮬레이션한 후, FPGA에 구현 및 합성하고 로직아날라이저(ILA)를 이용하여 검증하였다. RISC-V 코어는 SystemVerilog로 작성되어 효율적인 설계와 높은 재사용성을 나타내며, 다양한 응용 분야에서 사용 가능하다. Vivado를 사용하여 Ultra96-V2 FPGA보드에 합성함으로써 RISC-V 코어를 하드웨어로 구현하였고, 통합로직아날라이저(ILA)를 통해 설계의 정확성과 동작을 검증하였다. 실험 결과, 설계된 RISC-V 코어는 기대한 동작을 수행함을 확인하였으며, 이러한 연구 결과는 RISC-V 기반 시스템 설계와 검증에 중요한 기여를 할 수 있다.

Abstract RISC-V is an open-source instruction set architecture, and anyone can freely design and implement a RISC-V microprocessor. This paper designes and simulates the RISC-V architecture, synthesizing it in FPGA and verifying it using logic analyzer (ILA). RISC-V core is written in SystemVerilog, which has efficient design and high reusability, and can be used in various application fields. The RISC-V core is implemented as hardware by synthesizing it on the Ultra96-V2 FPGA board using Vivado, and the accuracy and operation of the design are verified through Integrated Logic Analyzer(ILA). As a result of the experiment, it is confirmed that the designed RISC-V core performs the expected operation, and these results can contribute to the design and verification of RISC-V based systems.

Key Words : RISC-V, SystemVerilog, FPGA

Ⅰ. 서 론

RISC-V 명령어집합 아키텍처는 2010년도에 최초로 University of California, Berkeley 전기 전자 컴퓨터공학부에서 David Patterson과 Krste Asanovic에 의하여 개발되었고 2015년도에 관련 비영리재단이 설립되었다. RISC-V의 프로세서는 다음과 같은 의미가 있다.

첫째, RISC-V는 오픈 소스 기반의 표준으로서, 아키텍처와 명령어 세트에 대한 정보가 모두 공개되어 있다. 이로 인해 특허 및 라이선스 비용이 없으며, 다양한 기업과 개발자들이 자유롭게 사용하고 발전에 기여할 수

*정회원, 한성대학교 전자 및 시스템반도체트랙
접수일자 2023년 7월 31일, 수정완료 2023년 9월 6일
게재확정일자 2023년 10월 6일

Received: 31 July, 2023 / Revised: 6 September, 2023 / Accepted: 6 October, 2023
*Corresponding Author: jblee@hansung.ac.kr
Electronics & System Semiconductor Track, Hansung University, Korea

있다. 이는 혁신과 개발을 촉진하며 새로운 기술과 제품을 쉽게 도입할 수 있도록 한다. 둘째, RISC-V는 모듈화와 확장성을 고려하여 설계되었다. 사용자는 필요에 따라 아키텍처를 맞춤형으로 또는 특정 어플리케이션에 최적화된 프로세서로 개발할 수 있다. 이로 인하여, 사물인터넷 기기부터 슈퍼컴퓨터까지 다양한 분야에서 사용될 수 있다. 셋째, RISC-V의 간단하고 효율적인 명령어 세트는 에너지 효율적인 프로세서 디자인을 가능하게 한다. 이는 이동통신 기기나 배터리가 제한된 장치에서 특히 중요한 장점이다. 넷째, 오픈 소스 및 교육용 목적으로 개발된 RISC-V는 학생들과 연구자들에게 컴퓨터 아키텍처와 프로세서 디자인에 대한 학습과 연구를 지원한다. 즉. 접근성이 높기 때문에 학습용 플랫폼으로 널리 사용되며, 새로운 아이디어와 혁신을 유발한다. 다섯 번째, RISC-V는 기업들이 새로운 제품과 기술을 개발하고 시장에 빠르게 반응할 수 있도록 돕는다. 특히 스타트업 기업들이 접근하기 쉽고 경쟁력 있는 제품을 개발하는 데 도움이 된다.

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.

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

본 논문은 다음과 같이 구성된다. 2 장에서 RISC-V 프로세서의 명령어집합과 프로세서 구조를 자세히 고찰한다. 3 장에서 모의실험 환경과 결과를 살펴보며, 4 장에서 FPGA 구현 및 ILA 결과를 보이고, 5 장에서 결론을 맺는다.

II. RISC-V 프로세서의 명령어집합 및 프로세서의 구조

1. RISC-V 명령어집합 아키텍처의 개요

RISC-V의 기본 정수형 명령어집합 아키텍처는 80년대 초기의 RISC 프로세서와 같으나, 분기 지연 방식을 채택하지 않고 가변길이 명령어 부호화를 지원한다는 점이 다르다.

주요 정수형 명령어집합 아키텍처는 RV32I와 RV64I으로 나뉘며, 각각 32 비트와 64 비트의 사용자 수준 주소공간을 나타낸다. 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 임베디드 및 클라이언트 장치에서 적절할 것이고, 낮은 메모리 트래픽과 저전력에 유리하기 때문이다. 또한, 32 비트 주소공간은 교육의 목적으로 충분하다.

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 정의하고 분류하였다.

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 배수로 확장 가능하다. 현재 모든 x86 시스템, 안드로이드, ARM용 윈도우즈 시스템이 리틀 인디언 방식을 사용하기 때문에 RISC-V에서도 리틀 인디언 메모리 시스템을 채택하였으며, 이것이 하드웨어 설계자들에게도 자연스럽다.

2. RISC-V 프로세서 아키텍처의 블럭도

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 쓰기작업을 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.

3. RISC-V 명령어집합의 개요

표 1에 RISC-V의 RV32I에 해당하는 37 개 명령어

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

를 나타냈다[1]. RISC-V는 32개의 범용레지스터로 구성되며, 레지스터0은 예외적으로 상수 0에 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. 이것은 각각 레지스터, 즉시값, 스토어, 무조건분기, 상위즉시값, 조건분기에 해당한다. 모든 명령어는 고정된 32 비트의 넓이를 가지며 메모리의 4 바이트의 경계에 정렬되어야 한다. 명령어 주소 비정렬 예외사건은 분기나 무조건 점프명령어에서 타겟 주소가 4 바이트에 정렬되어있지 않을 때 발생한다.

RISC-V 명령어집합 아키텍처는 원천 레지스터 Rs1, Rs2와 결과 레지스터 Rd를 모든 포맷에서 동일한 위치에 유지하여 명령어 해독을 용이하게 한다. 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 분리하여 모든 레지스터들을 동일한 위치에 오게 하였다. 즉시 피연산자들은 모두 부호확장되는데, 이것은 MIPS에서처럼 0을 확장하는 것에 뚜렷한 잇점을 발견하지 못했기 때문이다.

RISC-V의 6 가지 기본 명령어 포맷에서, R 포맷과 I 포맷은 서로 다르지만, S 포맷과 B 포맷, U 포맷과 J 포맷끼리 서로 유사하다. S 포맷에서 상위 7비트와 중간 5비트를 합쳐서 즉시피연산자 12 비트를 만들어내는 반면에, B 포맷에서는 상위 7비트와 중간 5비트를 뒤섞고 끝에 0을 추가하여 13 비트 오프셋을 만든다. 마찬가지로, U 포맷에서는 상위 20 비트를 즉시피연산자로 이용하는 반면에, J 포맷에서는 하위 20비트의 순서를 바꾸고 끝에 0을 추가하여 21 비트의 오프셋으로 이용한다.

4. 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 레지스터에 씌여진다.

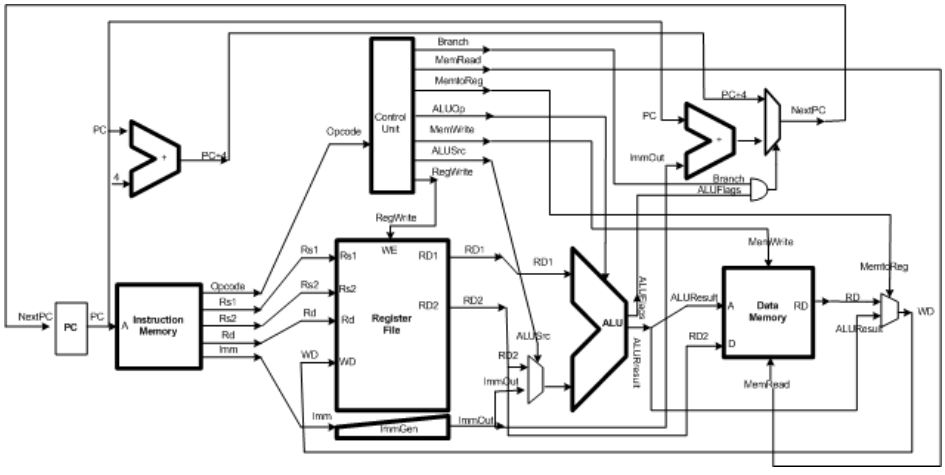

그림 1. RISC-V 아키텍처의 블럭도
Fig. 1. The RISC-V architecture block diagram

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 더한다. 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 논리 왼쪽 자리이동으로 하위비트에 0을 채우며, SRLI는 논리 오른쪽 자리이동으로 상위비트에 0을 채운다. SRAI는 산술 오른쪽 자리이동으로서, 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 읽어서 그 결과를 레지스터 Rd에 기록한다. 이 때, funct7과 funct3 영역이 연산의 형태를 선택한다. ADD와 SUB은 덧셈과 뺄셈을 수행한다. 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, 논리 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 담당할 수 있으며, 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 만들므로, 조건 분기 명령어는 ±4KB 범위를 담당 가능하다. BEQ와 BNE 분기 명령어에서는 각각 두 개의 레지스터를 비교하는데, Rs1과 Rs2가 같

거나 다를 때 각각 분기가 된다. BLT 명령어는 Rs1이 Rs2보다 작을 때 분기를 택하며, BGE 명령어는 Rs1이 Rs2보다 크거나 같을 때 분기를 택한다.

라. 로드 및 스토어 명령어

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

LW 명령어는 메모리로부터 32 비트의 데이터 값을 읽어서 Rd에 적재한다. LH 명령어는 메모리로부터 16 비트 값을 적재하여 32 비트로 부호확장한 다음에 Rd에 저장한다. LB와 LBU 명령어는 8 비트 값에 대하여 위와 동일한 기능을 각각 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 통신을 이용하여 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.

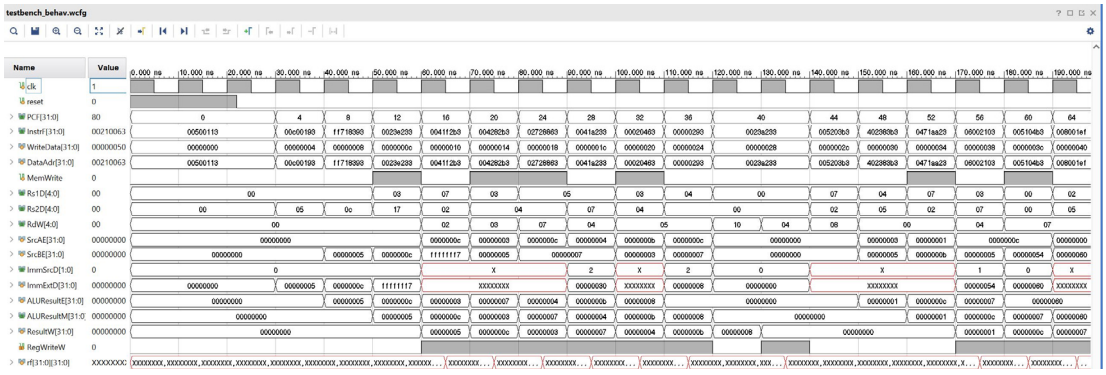

그림 2. RISC-V의 Vivado 모의실행 결과
Fig. 2. The Vivado simulation resutls of RISC-V processor

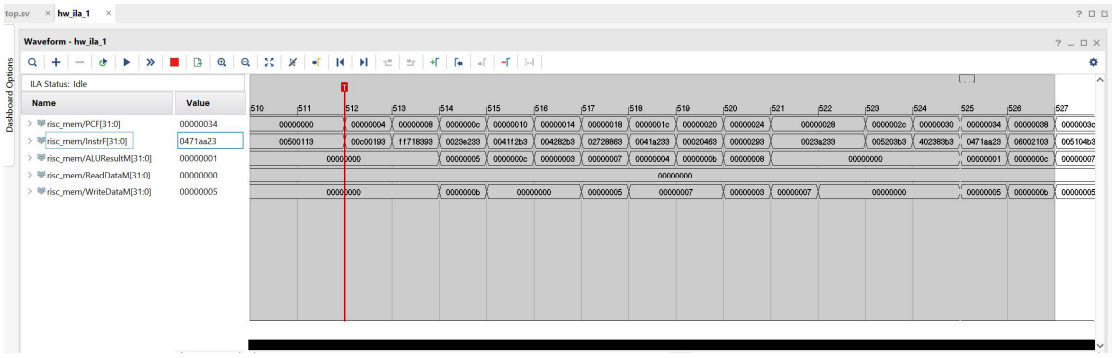

그림 3. FPGA로 구현된 RISC-V의 ILA 검증 결과
Fig. 3. The ILA verification of FPGA implemented RISC-V Processor

III. 모의실험 환경 및 검증 결과

본 논문의 구현 및 검증은 3.07 GHz로 동작하는 Intel Core i7-950 데스크탑 컴퓨터에 Windows 10 운영체제를 설치하여 시행하였다. 설계도구로 Xilinx사의 Vivado 2022.2 버전을, 하드웨어 기술언어로는 SystemVerilog가 이용되었다[2-6]. 2002년도에 개발된 SystemVerilog는 Verilog HDL을 확장하고 개선한 것으로서, 논리, 정수, 실수와 같은 기본 데이터 타입뿐만이 아니라, 구조체, 열거형, 연결 리스트 등 복잡한 데이터 타입을 지원함으로써 설계의 추상도를 고도화하여, 가독성과 유지 보수성을 향상시킬 수 있다.

그림 2에 Vivado에서 RISC-V 프로세서가 모의실행되는 결과의 파형을 나타냈다. RISC-V 명령어 집합의 주요 로드, 스토어, 레지스터 연산, 분기 명령어로 구성된 기계어 프로그램이 명령어 메모리에 입력되었다. 그 결과, RISC-V 프로세서가 제대로 동작하는 것을 확인할 수 있었다.

IV. FPGA 구현 및 ILA 검증 결과

그림 3에 RISC-V를 Ultra96-V2 FPGA 보드에 합성한 결과를 보였다. RTL 분석 결과, 본 프로세서의 합성에 2분 15초, 구현에 5분 47초가 소요되었다. 합성 결과, LUT 2336개, LUTRAM 318개, FF 3151개, BRAM 4.5개, IO 10개, BUFG 2개가 소요되었다. 전력 분석 결과, 총 0.229 W가 소모되었으며, 타이밍 분석 결과, 최장 셋업 시간이 15.5ns, 홀드 타임이 0.026ns, 펄스폭이 24.4 ns로 측정되었다.

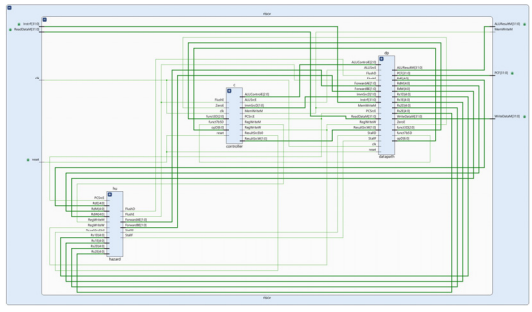

그림 4. RISC-V FPGA 합성 결과
Fig. 4. FPGA implemented RISC-V Processor

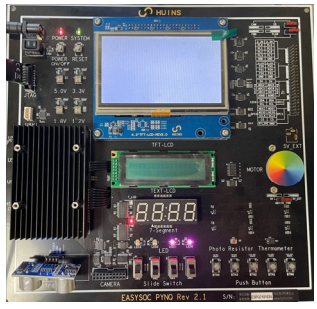

그림 5. 휴인스전자의 easySoC PYNQ 보드
Fig. 5. easySoC PYNQ board of Huins Inc.

그림 4에 본 연구의 FPGA 구현 및 실행에 이용된 장비인 휴인스전자의 easySoC PYNQ를 나타냈다. 이 장비에 Ultra96-V2 FPGA 보드가 탑재되었고, FPGA 소자는 xczu3eg-svba484-1-i이다.

그림 5에 ILA로 검증한 결과를 나타냈다. PC 어드레스 값을 트리거 기준으로 하여 0번지부터 순차적으로 프로세서의 내부 값들을 관찰하였다. 그 결과, ALU의 연산결과 값을 나타내는 ALUResultM의 값이 시뮬레이

션의 값과 정확히 일치하는 것을 확인하였다. 따라서, RISC-V가 올바르게 설계된 것을 알 수 있었다.

V. 결 론

본 논문에서는 Vivado 환경에서 SystemVerilog를 이용하여, RISC-V 프로세서를 설계하였다. 설계된 RISC-V 프로세서는 Ultra96-V2 FPGA 보드에 구현되었다. 시뮬레이션 결과와 ILA로 검증한 결과가 정확히 일치하여, 올바르게 동작하는 것을 확인할 수 있었다.

추후로, 구현되지 않은 나머지 명령어와 인터럽트와 관련된 기타 명령어를 추가시킨 후에, 설계된 RISC-V 프로세서를 Synopsys의 FrontEnd 및 BackEnd 설계 도구를 이용하여 국내 기관인 IDEC을 통하여 ASIC 칩으로 구현할 예정이다.

References

[1] "The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Version 2.3," A. Waterman, K. Asanović, D. Patterson, May 2017.
DOI:https//doi.org/10.5281/zenodo.835125

[2] S. L. Harris and D. M. Harris, "Digital Design and Computer Architecture RISC-V Edition", Morgan Kaufmann LLC, 2022.
DOI: https//doi.org/10.1016/B978-0-12-820064-3

[3] J. L. Hennessy, and D. A. Patterson, “Computer Architecture A Quantitative Approach”, 6th Edition, 2018.
DOI:https//doi.org/10.5555/1999263.

[4] J. Lee, “Design and Simulation of ARM Processor using VHDL”, Journal of The Institute of Internet, Broadcasting and Communication, Vol. 18, No. 5, pp. 229-235, Oct 2018.
DOI:https://doi.org/10.7236/JIIBC.2018.18.5.229

[5] J. Lee, “Simulation and Synthesis of RISC-V Processor”, Journal of The Institute of Internet, Broadcasting and Communication, Vol. 19, No. 1, pp. 239-245, Feb 2019.
DOI:https://doi.org/10.7236/JIIBC.2019.19.1.239

[6] J. Lee, “FPGA Implementation and Verification of A 32-bit Pipelined ARM Processor”, Journal of The Institute of Internet, Broadcasting and Communication, Vol. 22, No. 5, pp. 105-110, Aug 2022.
DOI:https://doi.org/10.7236/JIIBC.2022.5.105

저 자 소 개

이 종 복(정회원)

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

- Tel : 02-760-4497
- Fax : 02-760-4435
- E-mail : jblee@hansung.ac.kr
- 관심분야 : 마이크로 프로세서, 인공지능 프로세서

※ 본 연구는 한성대학교 학술연구비 지원과제임.