반도체디스플레이기술학회지 제23권 제2호(2024년 6월)
Journal of the Semiconductor & Display Technology, Vol. 23, No. 2. June 2024.

RISC-V 아키텍처 기반 6단계 파이프라인 RV32I프로세서의 설계 및 구현

민경진[*] · 최서진[*] · 황유빈[*] · 김선희[*†]
[*†]상명대학교 시스템반도체공학과

Design and Implementation of a Six-Stage Pipeline RV32I Processor Based on RISC-V Architecture

Kyoungjin Min[*], Seojin Choi[*], Yubeen Hwang[*] and Sunhee Kim[*†]
[*†]Department of System Semiconductor Engineering, Sangmyung University

ABSTRACT

UC Berkeley developed RISC-V, which is an open-source Instruction Set Architecture. This paper proposes a 32-bit 6-stage pipeline architecture based on the RV32I RSIC-V. The performance of the proposed 6-stage pipeline architecture is compared with the existing 32-bit 5-stage pipeline architecture also based on the RV32I processor ISA to determine the impact of the number of pipeline stages on performance. The RISC-V processor is designed in Verilog-HDL and implemented using Quartus Prime 20.1. To compare performance the Dhrystone benchmark is used. Subsequently, peripherals such as GPIO, TIMER, and UART are connected to verify operation through an FPGA. The maximum clock frequency for the 5-stage pipeline processor is 42.02 MHz, while for the 6-stage pipeline processor, it was 49.9MHz, representing an 18.75% increase.

Key Words : RISC-V, RV32I, Verilog-HDL, Pipeline, DMIPS

1. 서 론

RISC-V는 2010년 버클리 대학에서 개발되기 시작한 RISC (Reduced Instruction Set Computer) 계열 ISA (Instruction Set Architecture)이다[1]. 이는 기존의 ARM, x86 등과 같은 ISA와 달리 무료로 개방되어 있는 오픈 소스(open source) ISA로 컴퓨터 아키텍처 연구와 교육 지원을 목적으로 개발되었다. 최근에는 명령어 집합 확장을 통해 클라우드, 사물인터넷, AI 및 머신 러닝 응용 프로그램 등 산업 분야에도 적용되고 있다[2].

RISC-V ISA는 모듈식 명령어 세트로, 다양한 서브셋의 조합을 통해 전력, 면적, 성능 측면에서 다양한 요구 사항에 맞게 조정될 수 있는 유연성을 제공한다[3]. 이러한 RISC-V는 16-bit, 32-bit, 48-bit, 64-bit, 80+x-bit 길이 명령어를 지원하는데[4], 본 논문에서는 32bit 길이 명령어를 제공하는 정수형 명령어 집합 RV32I를 선택하였다. 본 논문에서는 Verilog-HDL을 이용하여 RV32I를 기반으로 하는 6단계 파이프라인 프로세서를 설계한 후 ISA Test, Dhrystone 벤치마크 등 명령어 검증을 위한 테스트를 진행하고, FPGA를 통해 구현하였다. 또한 Quartus Prime 20.1과 Dhrystone 벤치마크를 이용하여 RISC-V 5단계 파이프라인 RV32I 프로세서와 본 논문에서 설계한 RISC-V 6단계 파이프라인 RV32I 프로세서의 성능을 비교하였다. 본 논문은 이를 통해 프로세서의 파이프라인 단계가 증가함에 따라 성능이 향상됨을 확인하고, 최적의 파이프라인 구조를 도출하는 데 목적이 있다.

본 논문의 구성은 다음과 같다. 2장에서는 본 연구에서 제안하는 6단계 파이프라인 프로세서의 구조와, 제안하는 구조에

[†]E-mail: happyshkim@smu.ac.kr

서 고려해야 할 해저드에 대해 설명한다. 3장에서는 설계한 프로세서를 검증 및 구현한 후 5단계 파이프라인 프로세서와 성능을 비교하며 4장에서는 결론에 대해 설명한다.

2. 6단계 파이프라인 RV32I 프로세서 구조

2.1 5 단계 파이프라인 RV32I 프로세서

정수형 명령어 집합 RV32I RISC-V는 대부분 5단계 파이프라인으로 연구되고 있다[5-8]. IF (Instruction Fetch), ID (Instruction Decode), EX (Execute), MEM (Memory), WB (Write Back)와 같이 총 5단계로 구성된다. IF단계에서 프로세서는 메모리에서 명령어를 읽어온다. ID단계에서 프로세서는 레지스터 파일로부터 피연산자를 읽고 명령어를 해독해 제어 신호를 생성한다. EX단계의 프로세서 ALU (Arithmetic Logic Unit)와 연산을 수행한다. MEM단계의 프로세서는 해당되는 경우 데이터 메모리를 읽거나 쓴다. 마지막으로, WB단계에서 프로세서는 해당되는 경우 레지스터 파일에 결과를 저장한다.

Fig. 1은 기존 5단계 파이프라인 구조에 따라 설계한 프로세서의 블록도이다. 각 단계에서 하나씩, 총 5개의 명령어가 동시에 실행될 수 있다. 각 단계는 전체 로직의 약 1/5로 구성되므로 클럭 주파수는 단일 사이클 프로세서에 비해 명령어 처리율이 약 5배 향상된다. 마이크로프로세서는 초당 수백만 또는 수십억 개의 명령을 실행하기 때문에 지연시간보다 처리율이 더 중요하다. 따라서 본 논문에서는 명령어 처리율을 향상시키기 위하여 6단계 파이프라인 구조를 제안한다.

2.2 제안하는 프로세서 구조

본 연구에서는 32-bit 5단계 파이프라인 프로세서를 Verilog-HDL로 설계한 후 Quartus Prime 20.1을 통해 구현하고 결과를 분석하였다. 각 단계의 data delay를 비교해보았을 때, EX 단계가 27.053ns로 가장 오랜 시간이 걸리는 것을 확인하였다. 특히 EX 단계 내부에 위치한 덧셈기에서 17.624ns로 매우 긴 지연이 발생하였다.

기존 연구에서는 5단계에서 6단계 파이프 라인 프로세서를 구현할 때, MEM 단계에서 가장 긴 지연이 발생하여 데이터 캐시 유닛을 분할하여 파이프라인 단계를 확장했다[9]. 본 논문에서는 기존의 5단계 파이프 라인을 6단계로 확장하면서, data delay 결과를 토대로 가장 오랜 시간이 걸리는 EX 단계를 세분화하기 위해 EX 단계 내부에 파이프라인을 추가하였다. 제안하는 6단계 파이프라인 구조는 IF, ID, EX1(Execute1), EX2(Execute2), MEM, 그리고 WB로 구성된다.

Fig. 2는 제안하는 32bit 6단계 파이프라인 RISC-V 프로세서의 구조를 나타낸다. IF 단계에서 사용할 명령어의 주소를 결정하여 Instruction Memory로부터 명령어를 인출한 후, ID 단계에서 명령어를 해석하고 Control Unit에서 명령어에 따라 Control 신호를 생성한다. 해석된 명령어는 EX1 단계 내부에 위치한 ALU를 통해 연산이 수행되고, EX2 단계를

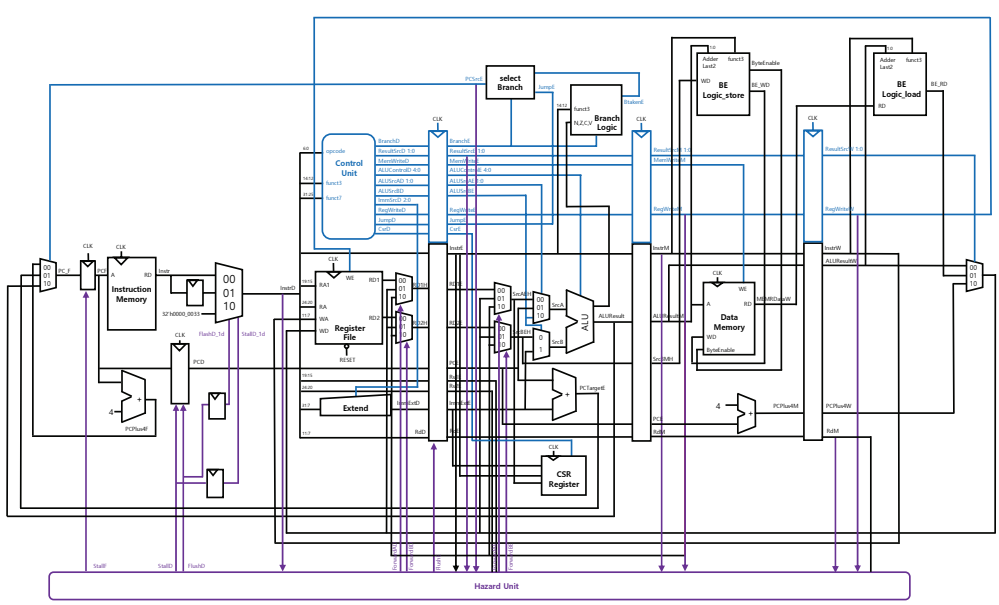


Fig. 1. RISC-V 5-stage pipeline RV32I processor architecture.

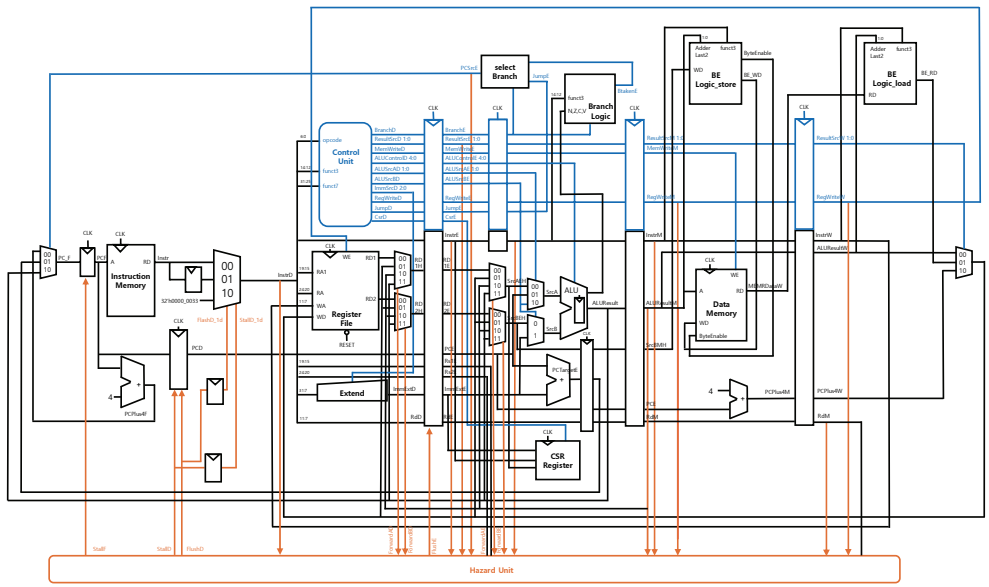


Fig. 2. RISC-V 6-stage pipeline RV32I processor architecture .

거쳐 MEM 단계에서 Data Memory에 접근하여 읽기 또는 쓰기 작업을 수행한다. 마지막 WB 단계에서는 레지스터 파일을 업데이트한다.

2.3 해저드

컴퓨터 아키텍처에서 해저드는 구조적 해저드, 데이터 해저드, 제어 해저드로 구분된다. 본 논문에서는 파이프라인을 기존의 5단계에서 6단계로 확장함에 따라 해저드 처리 방식이 변경되었음을 설명한다. 우선, 본 시스템은 하버드 구조를 따르므로 구조적 해저드는 발생하지 않는다. 데이터 해저드는 데이터 처리 명령어와 로드 명령어로 나뉜다. Fig. 3은 데이터 해저드 중 데이터 처리 명령어에 의해 발생하는 해저드와 관련된 그림이다. 데이터 처리 명령어로 인해 발생하는 해저드를 해결을 위해 포워딩 신호를 추가하였다. ForwardAD와 ForwardBD는 ID 단계에서의 소스 레지스터 번호와 EX2 단계의 목적지 레지스터 번호가 일치하고, RegWriteE2 제어신호가 1일 때 EX2 단계의 ALUResultE 값을 가져온다. 또한, ID 단계에서 소스 레지스터 번호와 MEM 단계의 목적지 레지스터 번호가 일치하고, RegWriteM 제어신호가 1일 때 MEM 단계의 ALUResultM 값을 가져오며, ID 단계에서의 소스 레지스터 번호와 WB 단계의 목적지 레지스터 번호가 일치하고, RegWriteW 제어신호가 1일 때 WB 단계의 Result 값을 가져온다. ForwardAE와 ForwardBE와 유사한 ForwardAD와 ForwardBD는 EX1 단계의 소스 레지스터와 목적지 레지스터가 일치했을 때 값을 가져온다.

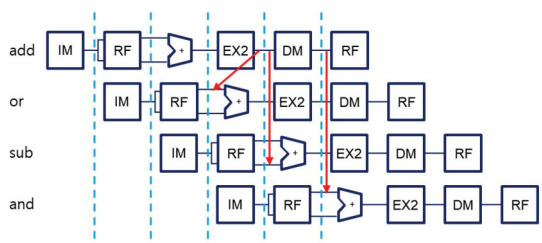


Fig. 3. Data Hazard due to Data Processing Instruction.

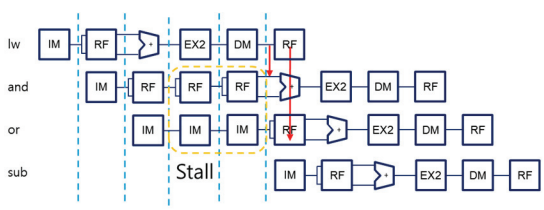


Fig. 4. Data Hazard due to lw Dependency.

Fig. 4는 데이터 해저드 중 로드 명령어에 의해 발생하는 해저드와 관련된 그림이다. 로드 명령어 실행시 데이터 해저드를 해결하기 위해, 로드 명령어가 수행될 때 ResultSrcE의 0번 비트가 1이 되고, ID 단계의 소스 레지스터가 EX1 단계의 목적지 레지스터와 동일하면 IF 단계와 ID 단계를 2번 지연시킨다. 그리고 로드 명령어가 수행될 때 1이 되는 ResultSrcE2의 0번 비트가 1이 되고, ID 단계의 소스 레지스터와 EX2 단계의 목적지 레지스터가 동일하면 IF 단계와 ID 단계를 1번 지연시킨다.

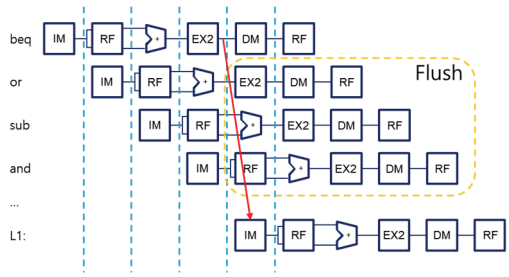


Fig. 5. Control Hazard due to Branch Dependency.

Fig. 5는 제어 해저드와 관련된 그림이다. 제어 해저드는 분기 명령어로 인해 발생한다. 분기 명령어가 실행된 후 EX2 단계에서 분기 신호의 활성화 여부가 결정된다. 분기 신호가 활성화될 때 발생하는 제어 해저드를 해결하기위해 분기 명령어가 실행된 후 분기가 발생하기 이전의 3개의 명령어를 플러시한다.

2.4 주변장치

설계한 프로세서를 검증 및 구현하기 위해 설계한 프로세서에 GPIO (General Purpose I/O), TIMER, UART (Universal Asynchronous Receiver/Transmitter)와 같은 다양한 주변 장치를 연결하였다. GPIO는 스 위치(Switch), 버튼(Button), 세그먼트(Segment), LED (Light-Emitting Diode) 등을 포함한 범용 목적 의 입출력을 위해 특화된 입출력 레지스터로, 다양한 외부 장치와의 상호작용을 가능케 한다[10].

설계한 RISC-V 프로세서와 메모리에 주소 디코더(Address Decoder)와 데이터 멀티플렉서(Data Multiplexer) 및 주변 장치를 추가한 블록도이다. 주소 디코더는 프로세서로부터 발생하는 주소 값에 따라 적절한 신호선을 선택하는 모듈이다. 선택된 신호선은 데이터 멀티플렉서의 입력으로 전달된다. 멀티 플렉서는 이 신호선을 통해 선택된 데이터 값을 출력하며, 출력된 데이터 값은 다시 프로세서의 MemRData 신호로 연결된다.

3. 실험 및 결과

3.1 구현 및 검증

본 논문에서는 설계한 프로세서의 명령어 검증과 성능 평가를 위해 ISA 테스트와 Dhrystone 벤치마크를 진행하였다. 합성은Quartus Prime 20.1을 이용하였으며, Altera Cyclone IV-E DE2-115 FPGA보드를 사용하여 구현 및 검증을 수행하였다. 구현 및 검증 과정에서 사용하는 명령어는 컴파일러를 통해 hex 파일을 생성하여 Instruction 메모리에 로드 하였다.

Fig. 6은 명령어 검증을 위해 ISA 테스트를 실행한 결과이다. ISA 테스트는 UC 버클리에서 제공하는 테스트 벤치로, 37개의 명령어와 1개의 C언어 테스트로 구성되어 있다[11]. Fig. 6에서 가운데 줄을 보면 37개의 명령어가 있으며, 각 명령어의 오른쪽에는 시뮬레이션된 경우의 수가 나와있다. 또한 각 명령어의 왼쪽에는 모든 명령어가 통과했음을 보여준다.

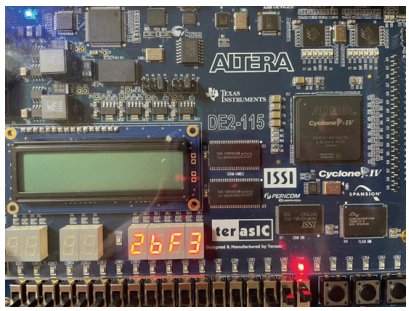
Dhrystone은 프로세서의 성능을 평가하기 위해 설계된 합성 컴퓨팅 벤치마크 프로그램이다[12]. 본 논문에서는 Dhrystone 벤치마크를 통해 프로세서의 성능 측정을 진행하였다. Fig. 7은 Dhrystone 벤치마크를 실행한 결과로, Dhrystones per second 값을 통해 Dhrystone 점수를 확인할 수 있다.

```
[passed] -                    addi.mif in        232 simulation cycles
[passed] -                     add.mif in        482 simulation cycles
[passed] -                    andi.mif in        188 simulation cycles
[passed] -                     and.mif in        502 simulation cycles
[passed] -                   auipc.mif in         37 simulation cycles
[passed] -                     beq.mif in        341 simulation cycles
[passed] -                     bge.mif in        386 simulation cycles
[passed] -                    bgeu.mif in        411 simulation cycles
[passed] -                     blt.mif in        341 simulation cycles
[passed] -                    bltu.mif in        366 simulation cycles
[passed] -                     bne.mif in        347 simulation cycles
[passed] -                     jal.mif in         33 simulation cycles
[passed] -                    jalr.mif in        123 simulation cycles
[passed] -                      lb.mif in        250 simulation cycles
[passed] -                     lbu.mif in        250 simulation cycles
[passed] -                      lh.mif in        258 simulation cycles
[passed] -                     lhu.mif in        263 simulation cycles
[passed] -                     lui.mif in         37 simulation cycles
[passed] -                      lw.mif in        266 simulation cycles
[passed] -                     ori.mif in        195 simulation cycles
[passed] -                      or.mif in        505 simulation cycles
[passed] -                      sb.mif in        462 simulation cycles
[passed] -                      sh.mif in        497 simulation cycles
[passed] -                  simple.mif in         10 simulation cycles
[passed] -                    slli.mif in        231 simulation cycles
[passed] -                     sll.mif in        510 simulation cycles
[passed] -                    slti.mif in        227 simulation cycles
[passed] -                   sltiu.mif in        227 simulation cycles
[passed] -                     slt.mif in        476 simulation cycles
[passed] -                    sltu.mif in        476 simulation cycles
[passed] -                    srai.mif in        246 simulation cycles
[passed] -                     sra.mif in        529 simulation cycles
[passed] -                    srli.mif in        240 simulation cycles
[passed] -                     srl.mif in        523 simulation cycles
[passed] -                     sub.mif in        474 simulation cycles
[passed] -                      sw.mif in        498 simulation cycles
[passed] -                    xori.mif in        197 simulation cycles
[passed] -                     xor.mif in        504 simulation cycles
```

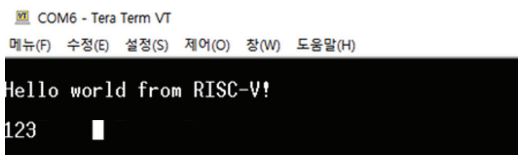
Fig. 6. A result of ISA Test for the 6-stage pipeline processor.

```
TBMAN: Str_1_Loc:           DHRYSTONE PROGRAM, 1'ST STRING

TBMAN:          should be:  DHRYSTONE PROGRAM, 1'ST STRING

TBMAN: Str_2_Loc:           DHRYSTONE PROGRAM, 2'ND STRING

TBMAN:          should be:  DHRYSTONE PROGRAM, 2'ND STRING

TBMAN:

TBMAN: Microseconds for one run through Dhrystone: 1090

TBMAN: Dhrystones per Second:                       917

TBMAN: CPU requested termination, exit code        123
Design ran for      436829 cycles
$finish called from file "../../src/rtl/myCPU/rev12_project/tbman/
$finish at simulation time           8736810000
           V C S   S i m u l a t i o n   R e p o r t
Time: 8736810000 ps
CPU Time:      8.640 seconds;       Data structure size:   0.3Mb
Thu May 16 19:11:08 2024
```

Fig. 7. A result of Dhrystone benchmark for the 6-stage pipeline processor.



(a)



(b)

Fig. 8. (a) Verification of FPGA operation, (b) Verification of UART operation using the Tera Term.

Fig. 8은 FPGA를 통해 프로세서에 연결된 주변 장치들이 작동하고 있는 것을 보여준다. UART 는 시리얼 케이블을 연결하여 통신 인터페이스인 Tera Term을 통해 프로세서와 통신하는 동작을 확인하였다. GPIO는 FPGA의 LED, 세그먼트, 스위치, 버튼을 사용하여 Peripheral 동작을 확인하였고, 타이머는 세그먼트에 표시되는 수가 클럭에 따라 점 점 증가하는 것을 통해 타이머 기능이 정상적으로 동작하는 것을 확인하였다.

3.2 성능 비교

기존의 5단계 파이프라인 프로세서와 본 논문에서 설계한 6단계 파이프라인 프로세서의 성능 비교를 위해 Dhrystone 벤치마크와 Quartus Prime 20.1를 사용하였다.

성능 비교를 위하여 Dhrystone 벤치마크를 통해 DMIPS (Dhrystone Millions of Instructions Per Second) 값도 비교하였다. DMIPS는 Dhrystone 점수를 VAX 11/780 컴퓨터가 1초 동안에 수행한 Dhrystone 루프의 수인 1757로 나눈 값이다 [13,14]. 이때 Dhrystone 점수는 Dhrystones per second 값으로 Dhrystone 벤치마크를 실행시키면 확인할 수 있다. Fig. 9는 기존의 5단계 파이프라인 프로세서의 Dhrystone 벤치마크를 진행한 결과다. 그 결과 Dhrystones per second가 1035으로, DMIPS가 약 0.59임을 알 수 있다. 본 논문에서 설계한 6단계 파이프라인의 프로세서의 Dhrystones per second는 917로, DMIPS는 약 0.52가 나오는 것을 알 수 있다. 따라서 기존의 5단계 파이프라인 프로세서의 DMIPS보다 본 논문에서 설계한 6단계 파이프라인 프로세서의DMIPS가 약 0.067 감소하였다. 기존 연구된 RISC-V 6단계 파이프라인 RV32I 프로세서의 DMIPS는 1.9이다. 본 논문에서 제안한 프로세서와 비교했을 때 더 높은 성능을 보이는데, 이는 기존 프로세서가 가하기 때문으로 추정된다[15].

Quartus Prime 20.1에서 합성을 진행한 후, 성능을 비교하였다. Table 1은 Quartus Prime 20.1에서 확인한 5단계 파이프라인 프로세서와 6단계 파이프라인 프로세서의 Total logic elements, Total register, 클럭 최대 주파수 Fmax 값을 나타낸다. Total logic elements의 경우, 유효한 114,480개에서 5단계 파이프라인 프로세서는 3,711개, 6단계 파이프라인 프로세서는 4,013개가 사용되었다. Total registers의 경우, 5단계 파이프라인 프로세서는 1,856개, 6단계 파이프라인 프로세서는 1,995개가 사용되었다. 파이프라인이 1개가 추가됨에 따라 Total logic elements와 Total register는 증가한 것을 확인할 수 있다. 더불어, 5단계 파이프라인 프로세서와 6단계 파이프라인 프로세서의 Fmax 값은 각각 42.02 MHz, 49.9 MHz로, Fmax가 약 18.75% 증가하였다. Fmax는 프로세서의 데이터 처리 속도와 밀접한 관련이 있고, Fmax 값이 높을수록 클럭 최대 주파수가 높아지고, 데이터 처리 속도가 빨라지게 된다. 따라서 5단계 파이프라인 프로세서를 6단계 파이프라인 프로세서로 확장하여 설계한 결과, Fmax 값이 증가하여 실행 속도가 향상된 것을 확인할 수 있다.

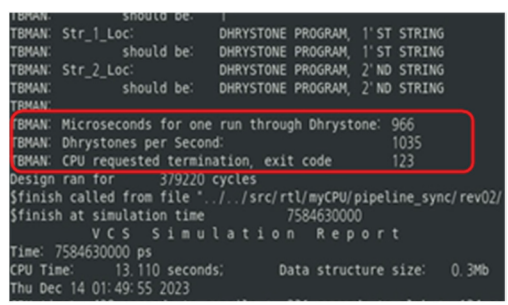


Fig. 9. A result of Dhrystone benchmark for the 5-stage.

Table 1. A result of Quartus

Resource	Utilization	
	5-stage	6-stage
Total logic elements	3,711	4,013
Total registers	1,856	1,995
Fmax	42.02MHz	49.9 MHz

4. 결 론

본 논문에서는 RISC-V ISA 중 32-bit 명령어를 제공하는 정수형 명령어 집합 RV32I ISA를 기반으로 6단계 파이프라인 RV32I 프로세서를 구현하였다. 구현한 프로세서에

GPIO, TIMER, UART와 같은 주변 장치를 연결하여 FPGA를 통해 동작을 검증하였다. 또한, Quartus Prime 20.1과 Dhrystone 벤치마크를 통해 성능을 검증하였고, 기존의 RISC-V 5단계 파이프라인 프로세서와 비교하여 성능이 향상됨을 보였다. 추후 연구로 6단계 파이프라인 프로세서에서 파이프라인 단계를 더 증가시키는 것을 고려하고 있다. 파이프라인 단계를 확장하는 과정에서 새로운 종류의 해저드를 처리해야 하지만, 전체적인 성능이 더욱 향상될 것으로 기대된다. 추가적으로, 본 연구에서 설계한 RV32I 프로세서에 기반하여 확장 명령어를 추가하여 RV32IM 프로세서를 설계하고자 한다. 또한 캐시를 구현하여 성능 향상을 위한 연구를 수행하고자 한다.

감사의 글

다음의 성과는 과학기술정보통신부와 연구개발특구진흥재단이 지원하는 과학벨트지원사업으로 수행된 연구결과입니다.

참고문헌

1. A. Waterman, K. Asanovi, and RISC-V Foundation, “The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Document Version 20191213”, 2019.
2. A. Waterman, K.Asanovi, and RISC-V Foundation, “The RISC-V Instruction Set Manual, Volume II: Privileged Architecture, Document Version 20190608-Priv-MSU-Ratified”, 2019.
3. C. Chen et al., "Xuantie-910: A Commercial Multi-Core 12-Stage Pipeline Out-of-Order 64-bit High Performance RISC-V Processor with Vector Extension : Industrial Product," 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA), pp. 52-64, 2020.
4. Jon Gabay, RISC-V, An open-source processor architecture that has become an international standard, November, 28, 2022, from https://www.epnc.co.kr/news/articleView.html?idxno=230169
5. Davide Harris&Sarah Harris,「Digital Design and Computer Architecture, Risc-V Edition」, Morgan Kaufmann, 2021
6. Park, Su-bin, et al. "A Design and Implementation of 32-bit Five-Stage RISC-V Processor Using FPGA." Journal of The Korean Society of Semiconductor & Display Technology, Vol. 22, pp. 81-86, 2023.
7. A. E. Phangestu, I. T. Mujiono, M. I. Kom and S. Ahmad Zaini, "Five-Stage Pipelined 32-Bit RISC-V Base Integer Instruction Set Architecture Soft Microprocessor Core in VHDL," 2022 International Seminar on Intelligent Technology and Its Applications (ISITIA), Surabaya, Indonesia, pp. 304-309, 2022.
8. Anmol Singh, Arpit Kumar, Abhishek Singh, R. Anirudh Reddy, and K. N. Pushpalatha, "Design and Implementation of RISC-V ISA (RV32IM) on FPGA," SSRG International Journal of VLSI & Signal Processing, vol. 10, no. 2, pp. 17-21, 2023.
9. W.J. Teng, “Design of a 6-Stage pipeline RISC processor.” Universiti Tunku Abdul Rahman, Faculty of Information and Communication Technology. UTAR Eprints (4281) - 2021.
10. KTword, 2021, from http://www.ktword.co.kr/test/view/view.php?m_temp1=5385
11. L. Wren, RISCBoy Documentation, from https://github.com/Wren6991/RISCBoy
12. Reinhold P. Weicker, “Dhrystone: A synthetic systems programming benchmark,” Communications of the ACM, Vol.27, No.10, pp.1013-1030, 1984.
13. Clive Maxfield, MIPS introduces new Aptiv generation of processor cores, May, 10, 2012, from https://www.eetimes.com/mips-introduces-new-aptiv-generation-of-processor-cores/
14. Jo, Sang-un, et al. "A Design and Implementation of 32-bit Five-Stage RISC-V Processor Using FPGA." Journal of The Korean Society of Semiconductor & Display Technology, Vol. 21, pp. 27-32, 2022.
15. Ultraembedded, biriscv. Retrieved September, 19, 2021, from https://github.com/ultraembedded/biriscv.

접수일: 2024년 5월 30일, 심사일: 2024년 6월 12일, 게재확정일: 2024년 6월 21일