반도체디스플레이기술학회지 제21권 제4호(2022년 12월)
Journal of the Semiconductor & Display Technology, Vol. 21, No. 4. December 2022.

FPGA를 이용한 32-bit RISC-V 5단계 파이프라인 프로세서 설계 및 구현

조상운[*] · 이종환[*] · 김용우[*†]
[*†]상명대학교 시스템반도체공학과

A Design and Implementation of 32-bit Five-Stage RISC-V Processor Using FPGA

Sangun Jo[*], Jonghwan Lee[*] and Yongwoo Kim[*†]
[*†]Department of System Semiconductor Engineering, Sangmyung University

ABSTRACT

RISC-V is an open instruction set architecture (ISA) developed in 2010 at UC Berkeley, and active research is being conducted as a processor to compete with ARM. In this paper, we propose an SoC system including an RV32I ISA-based 32-bit 5-stage pipeline processor and AHB bus master. The proposed RISC-V processor supports 37 instructions, excluding FENCE, ECALL, and EBREAK instructions, out of a total of 40 instructions based on RV32I ISA. In addition, the RISC-V processor can be connected to peripheral devices such as BRAM, UART, and TIMER using the AHB-lite bus protocol through the proposed AHB bus master. The proposed SoC system was implemented in Arty A7-35T FPGA with 1,959 LUTs and 1,982 flip-flops. Furthermore, the proposed hardware has a maximum operating frequency of 50 MHz. In the Dhrystone benchmark, the proposed processor performance was confirmed to be 0.48 DMIPS.

Key Words : RISC-V, Processor, AMBA BUS, AHB-lite, FPGA

1. 서 론

RISC-V는 2010년부터 미국의 UC 버클리 대학에서 개발 중인 RISC (Reduced Instruction Set Computer) 계열 ISA (Instruction Set Architecture) 이며, 여기서 V는 1981년 이후 버클리 대학에서 개발된 RISC 아키텍처의 세대 수를 의미한다[1,2]. RISC-V는 RISC-V International 재단을 통해 소프트웨어 및 하드웨어 설계를 위한 ISA의 무제한 사용을 허용하는 오픈 소스 ISA를 지원해 특정 기업이 소유하지 않는 구조로 인수 시도 등 경영권에 영향을 받지 않고 라이선스 비용이 없다는 특징이 있어서 시스템 반도체 IP 시장에서 높은 점유율을 가진 ARM의 대안으로 부상하고 있으며, 구글, IBM, 퀄컴, 마이크로소프트 등 여러 기업과 단체에서 RISC-V 연구를 후원하고 있다[3]. RISC-V는 명령어 집합 확장이나 임의의 명령어 집합을 직접 추가하여 연구 및 교육만 아니라 스마트폰이나 임베디드 장치의 CPU, 클라우드 컴퓨터, AI 등 실제 산업계에서도 광범위하게 사용할 수 있다.

본 논문에서는 RV32I를 지원하는 5단계 파이프라인 프로세서를 설계하고, 이를 FPGA로 구현한다. 본 논문의 주요 기여는 다음과 같다.

1) RISC-V ISA 중 기본 32-bit 정수형 명령어 집합인 RV32I를 기반으로 32-bit 5단계 파이프라인 CPU를 구현하여 SoC(System on Chip) 시스템을 구성하였다.
2) AMBA 표준의 AHB-lite 버스 마스터를 설계하여 네

[†]E-mail: yongwoo.kim@smu.ac.kr

이티브 버스를 AHB-lite 버스로 변환하였다. 변환된 AHB-lite 버스에 UART(Universal asynchronous receiver and transmitter), GPIO(General Purpose Input/Output), Timer 등의 주변장치 모듈을 연결해 SoC 시스템을 구현하였다. 네이티브 버스 대신 산업계에서 표준으로 쓰이는 AMBA AHB-lite 버스를 사용하면 기존에 설계되어 있는 많은 IP를 다시 설계해야 하는 까다로운 단계를 거치지 않고 쉽게 연결할 수 있는 장점이 있다.

3) Arty A7-35T FPGA를 이용하여 Dhrystone 벤치마크[4]를 실행해 SoC 시스템의 동작을 검증하고 DMIPS (Dhrystone Millions of Instructions Per Second)를 측정하여 다른 프로세서들과 성능 비교를 수행하였다.

본 논문의 구성은 다음과 같다. 2장에서는 RISC-V ISA 기반 프로세서 및 AHB-lite 버스 프로토콜 관련 연구를 소개하고, 3장에서는 제안하는 5단계 파이프라인 RISC-V 프로세서 및 SoC 시스템에 대하여 설명한다. 4장에서는 실험 및 결과에 대하여 설명한다. 마지막 장에서는 결론을 짓는다.

2. 관련 연구

이번 절에서는 RISC-V ISA 기반 프로세서와 주변장치를 연결하는 AMBA AHB-lite 시스템 버스에 대한 연구 동향을 서술한다.

2.1 RISC-V 프로세서 기반 관련 연구 동향

딥러닝과 관련된 연구가 많아지면서 CPU나 GPU에 관련된 연구가 많이 진행되고 있다.[5][6] RISC-V-ISA 또한ISA가 공개되고 난 후 다양한 프로세서들이 설계되어 왔다. RISC-V를 개발하기 시작한 버클리 대학은 자체 개발한 스칼라 기반 HDL 언어인 Chisel을 이용하여 RISC-V Rocket Chip SoC Generator 프로젝트를 진행하였다[7]. Rocket Chip generator의 대표적인 CPU 코어는 Rocket 코어[8]와 Berkeley Out-of-Order Machine (BOOM) 코어[9]가 있다. Rocket Core는 RV32G ISA와 RV64G ISA를 구현하는 5단계 파이프라인 순차 실행 프로세서로, 일부 확장 명령어 집합인 (M, A, F, D)를 지원한다. 또한, 페이지 기반 가상메모리, 캐시, 분기 예측 기능이 있는 프론트 엔드를 지원하는 MMU(Memory Management Unit)가 존재한다. BOOM 코어는 RV64G ISA를 지원하는 6단계 파이프라인 비순차 실행 슈퍼스칼라 프로세서이다. BOOM 코어는 비 순차 실행 마이크로 아키텍처를 교육, 연구 및 산업을 위한 구현의 기준선 역할을 하는 목적으로 개발되었다.

VexRiscv[10]는 SpinalHDL으로 구현한 FPGA에 친화적인 5단계 파이프라인 프로세서로, RV32I[M][A][F[D]][C] ISA를 지원한다. AXI4, Avalon, Wishbone 버스를 사용할 수 있으며, 선택적으로 플러그인을 추가하여 명령어 및 데이터 캐시와 MMU, CSR (Control Status Register) 레지스터 등을 CPU에 추가할 수 있다.

국내에서도 RV32I ISA를 기반으로 2단계 파이프라인 프로세서를 설계한 사례가 있다[11]. 40개의 ISA 중 FENCE와 EBREAK 명령어를 제외한 38개의 ISA를 지원하고, M모드에서 트랩처리를 위한 특권 명령어 집합 (Privileged ISA) 및 CSR을 구현하였고, PULP 프로젝트[12]의 PULPissimo SoC[13]를 레퍼런스로 주변장치를 설계하여 시스템을 구성하였다.

2.2 AMBA AHB-lite 버스 프로토콜

SoC 시스템을 구성하기 위해 내부 연결 버스로 AMBA(Advanced Microcontroller Bus Architecture)버스 표준 프로토콜이 많이 사용된다. [14]. AMBA 버스의 기본적인 기능으로는 어느 버스와 동일하게 주소, 데이터, 제어신호를 전송하는 기능을 가지고 있다. 본 논문에서 설계한 AMBA AHB-lite 버스 마스터는 제안된 RISC-V 프로세서와 ARM Cortex-M0 DesignStart[15]에 포함된 주변장치 IP와 SoC 칩 내부 연결을 위해 제안하였다.

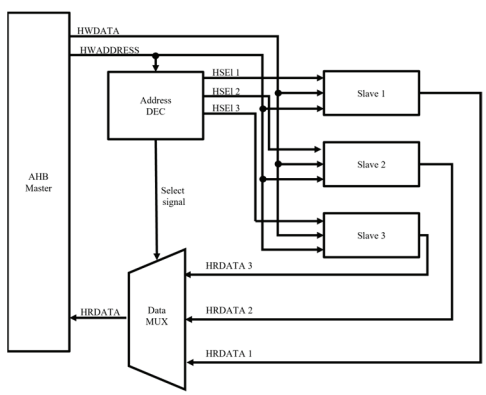

Fig. 1. A block diagram of read and write data on the AHB-lite bus.

AMBA 버스는 크게 AHB(Advanced High-performance Bus), APB(Advanced Peripheral Bus), AXI(Advanced eXtensible Interface) 3가지로 구성할 수 있다. AHB 버스는 고성능의 버스로, 시스템 버스 중 메인을 구성한다. APB 버스는 AHB 버스 대비 느린 주변장치의 제어를 위한 버스이다. AHB 버스와 APB 버스를 분리함으로써 전력 소모를 제어하며, 레이턴시를 감소시킨다. AXI 버스는 AHB 버스와 유사하나,

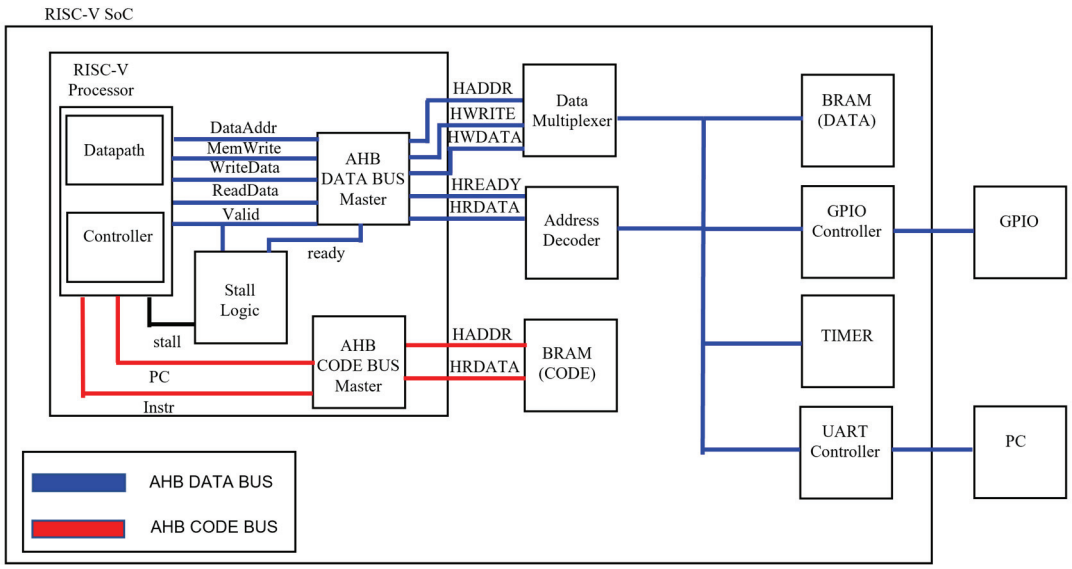

Fig. 2. A block diagram of our proposed SoC system.

채널의 개념이 도입되어, 독립적 데이터 전송을 가능하게 하여, 지연 시간을 줄인 버스이다. 본 논문은 AMBA3 기반의 AHB-lite를 시스템 버스로 사용한다[16]. AHB-lite 버스는 AHB 버스 대비 중재 신호가 일부 삭제된 버스이다. Fig. 1은 AHB-lite 버스가 데이터를 읽고 쓰는 과정을 표현한다.

3. 제안하는 RISC-V 기반 SoC 시스템

Fig. 2는 설계한 32-bit 5단계 파이프라인 RV32I ISA를 지원하는 RISC-V 프로세서를 기반으로 하여 구현한 SoC 시스템의 블록도를 나타내고 있다. 본 논문에서는 AHB-lite 버스 마스터를 통해 네이티브 버스를AHB-lite 버스 프로토콜로 변환해주어AHB-lite 버스를 사용하는 ARM 사의 Cortex-M0 DesignStart에 포함된 AHB-lite 버스 기반 내부 SRAM 메모리와 UART 등의 주변장치들과 연결해 주었다. UART는 Dhrystone 벤치마크 결과를 시리얼 터미널에서 확인하기 위한 용도로 사용한다. 또한, 제안한 AHB-lite 버스 마스터 모듈에서는 AHB-lite 데이터 버스에 연결된 주변 장치들에 레이턴시가 있을 경우 데이터 메모리에 접근하는 명령어가 실행되면 valid 신호를 활성화하여, stall 신호를 발생시키고, ready 신호가 활성화되면 stall 신호를 비활성화 시켜주는 logic을 추가로 설계하였다. Fig. 3은 제안한 SoC 시스템의 메모리맵이다. 다음 절에서는 네이티브 방식의 버스를 AHB-lite 버스 프로토콜로 변환시키는 AHB 버스 마스터와 설계한32-bit 5단계 파이프라인 RISC-V 프로세서 구조에 대하여 서술한다.

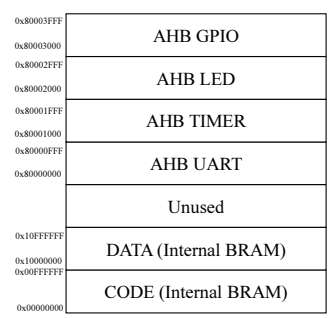

Fig. 3. A memory map of proposed SoC system.

3.1 제안하는 32-bit 5 단계 파이프라인 RISC-V 프로세서 구조

Fig. 4는 해리스 제안 기법[17]을 기반으로 설계한 32-bit 5단계 파이프라인 RISC-V 프로세서의 블록도를 보여주고 있다. 해리스 제안 기법[17]의 경우 RISC-V 프로세서 설계 과정을 상세하게 보여주고 있지만, RV32I 명령어 전체 중 일부 명령어를 기반으로 설계를 진행한다. 본 논문에서 설계한 프로세서는 RV32I ISA의 40개의 명령어 중 FENCE, ECALL, EBREAK 명령어를 제외한 37개의 명령어를 지원한다.

설계한 5단계 파이프라인 구조는 코드 메모리에서 명령어를 가져오는 IF (Instruction fetch) 단계, 명령어를 해독하여 컨트롤 유닛에서 제어 신호들을 생성하고, 필요한 레지스터들을 불러오는 ID (Instruction decode) 단계, ALU (Arithmetic logic unit)를 이용하여 연산을 수행하거나 주소

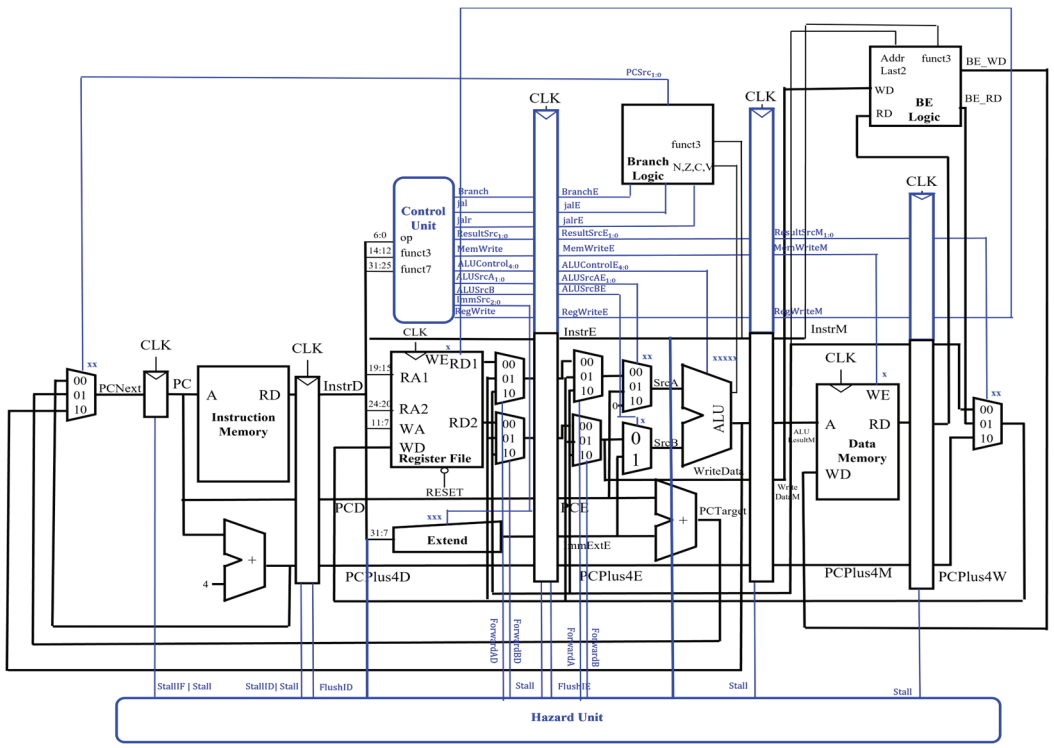

Fig. 4. A block diagram of our proposed 32-bit 5 stage pipeline RISC-V processor.

값을 계산하는 EX (Execute) 단계, 데이터 메모리에 접근하여 데이터를 쓰거나 읽는 MEM (Memory) 접근 단계, 데이터를 레지스터에 쓰는 WB (Writeback) 단계로 구분된다.

파이프라인 설계 시에 발생하는 파이프라인 해저드를 해결하기 위해 해저드 유닛을 설계하였다. 데이터 해저드를 해결하기 위해 포워딩 신호를 생성하였다. ForwardAD와 ForwardBD는 각각 ID 단계에서의 불러오는 소스 레지스터 번호가 MEM 단계의 목적지 레지스터 번호와 일치하고, RegWriteM 제어신호가 활성화되었을 때 MEM 단계의 ALUResultM으로 포워딩 해 주고, ID 단계에서의 불러오는 소스 레지스터의 번호가 WB 단계의 목적지 레지스터 번호와 일치하고, RegWriteW 제어 신호가 활성화되었을 때 WB 단계의 Result 신호를 포워딩 해준다. ForwardAE와 ForwardBE는 ForwardAD, ForwardBD와 유사하지만 ID 단계의 소스 레지스터가 아닌 IE 단계의 소스 레지스터가 목적지 레지스터와 일치했을 때 포워딩을 수행한다.

로드(load) 명령어 실행 시 데이터 의존성을 해결하기 위해 ID 단계의 소스 레지스터와 EX 단계의 목적지 레지스터가 일치하고, 로드 명령어가 수행될 때 활성화되는 ResultSrcE의 0번 비트가 활성화되면 IF 단계와 ID 단계를 지연해 준다. 또한, 분기 명령어가 fetch 된 후 EX 단계가 되어야 분기가 결정되기 때문에, 분기가 발생하면 분기 명령어가 fetch 된 후 분기가 발생하기 이전의 2개의 명령어를 flush 해준다. 데이터 버스에 연결된 주변장치의 지연으로 발생하는 stall 신호 또한 데이터가 준비될 때까지 명령어를 실행하지 않고 기다려주게 하였다.

3.2 제안하는 AHB BUS Master

본 논문에서 제안한 5단계 파이프라인 구조의 RISC-V 프로세서는 하버드 구조를 사용하기 때문에 데이터 버스와 명령어 버스가 분리되어 있다. 따라서, 본 논문에서는 AHB-lite 데이터 버스 마스터와 AHB-lite 코드 버스 마스터를 각각 설계하여 각각 데이터 버스와 코드 버스로 변환하여 연결하였다. CPU의 DataAddr (Data Address), MemWrite (Memory Write), WriteData, ReadData 신호는 각각 데이터 버스의 HADDR, HWRITE, HWDATA, HRDATA 신호와 연결되며, HTRANS 신호는 네이티브 버스의 valid신호에 따라 1로 설정되어있으면 NONSEQ신호로 변환해주고, 0으로 설정되어 있으면 IDLE 신호로 변환해준다. 또한, 네이티브 버스의 WriteData신호를 한 클럭 지연 후에 HWDATA로 전달해준다. 마지막으로 HSIZE 신호는 네이티브 버스의 ByteEnable 신호에 따라 word, halfword, byte 단위로 데이터의

크기를 결정하여 신호를 변환해준다.

Fig. 5는 기존의 버스를 AHB-lite 데이터 버스로 변환했을 때의 타이밍 도이다. Fig. 5(a)는 쓰기 동작, Fig. 5(b)는 읽기 동작의 타이밍 도이다. HWDATA와 HRDATA가 한 클럭씩 지연되어 연결되는 것은 AHB-lite 버스가 주소와 제어 신호를 주고받는 address phase와 데이터를 주고받는 data phase로 구분되어 있기 때문이다. CPU의 PC (Program Counter) 신호는 코드 버스의 HADDR로 변환되고, Instr (Instruction) 신호는 코드 버스의 HRDATA와 연결된다.

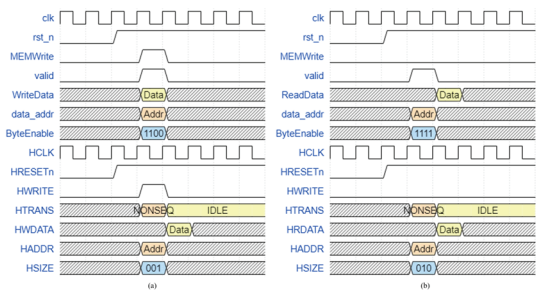

Fig. 5. A timing diagram of Native BUS covert into AHB-lite data BUS. (a) A timing diagram of write operation. (b) A timing diagram of read operation.

4. 실험 및 결과

본 논문에서는 프로세서의 동작을 검증하고 성능을 측정하기 위해 Arty A7-35T 기반 FPGA 보드를 사용하였다. 동작 속도는 50MHz 사용하여 Xilinx Vivado 툴을 이용하여 합성을 진행하였다. 검증과 성능 측정을 위한 소프트웨어 코드는 데이터 메모리와 코드 메모리를 위한 두 개의 hex 파일로 변환하여 코드를 수행하였다. 합성 결과, 최대 동작 주파수는 53MHz까지 동작하는 것을 확인하였다. Table 1은 제안한 프로세서 및 SoC 하드웨어의 로직 사용량 및 파워 소모를 보여준다.

Table 1. Hardware and Power Consumption of In-house Processor

Resource	Available	Used
LUT	20,800	1,959(9.4%)
FF	41,600	1,928(4.6%)
BRAM	50	47(94.0%)
DSP	90	3(3.3%)
Maximum Frequency	53MHz	-
Power(W)	-	0.248

또한, 본 논문에서는 설계된 프로세서의 성능을 측정하기 위해 Dhrystone 벤치마크를 진행하였다. 반복횟수는 500회고, 컴파일 옵션은 -O3 -fno-inline를 이용하였다. Fig. 6은 데이터 버스에 연결된 주변장치의 레이턴시(latency)가 0일 때의 Dhrystone 결과를 UART를 이용하여 PC와 통신하여 시리얼 터미널에 출력한 모습을 보여주고 있다. Dhrystone per Second가 850이므로 이를 VAX 11/780의 Dhrystone 점수인 1757로 나누어 주면 DMIPS가 0.48이 나오는 것을 알 수 있다. Table 2는 데이터 버스에 연결된 주변장치의 레이턴시에 따른 DMIPS와 다른 프로세서의 DMIPS를 비교한 표를 보여주고 있다.

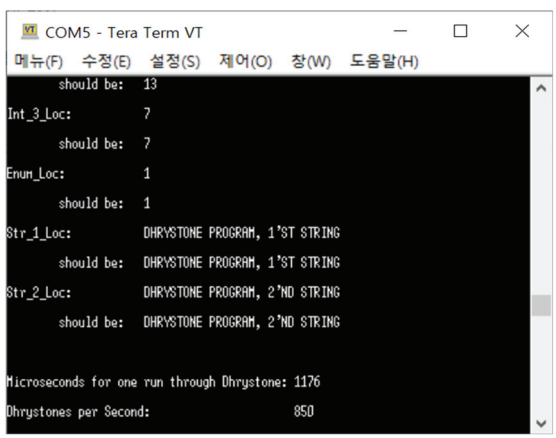

Fig. 6. A result of Dhrystone benchmark

Table 2. Comparison of DMIPS with In-house Processor and Other Processors

Processor	ISA	Pipeline stage	DMIPS
In-house (Latency: 0)	RV32I	5	0.48
In-house (Latency: 1)	RV32I	5	0.38
In-house (Latency: 2)	RV32I	5	0.31
VexRiscv full [8]	RV32IM	5	1.21
VexRiscv small [8]	RV32I	5	0.52
Cortex-M3 [16]	Armv7-M	3	1.24

Table 2에서 보는 바와 같이 데이터 메모리를 포함하고 있는 주변장치의 레이턴시가 증가할수록 DMIPS가 감소하는 것을 볼 수 있다. 이는 추후 데이터 캐시를 구현하여 속도를 향상할 수 있다. VexRiscV[8]의 프로세서 중 곱

셈을 지원하지 않는 경우 DMIPS는 0.52로 In-house 프로세서와 유사하지만, 곱셈을 지원하는 M 확장이 포함된 경우 DMIPS는 1.21로 더 높은 것을 확인할 수 있다. 또한, Cortex-M3 프로세서[18]의 DMIPS에 비해 본 논문에서 제안한 In-house 프로세서의 DMIPS 성능이 더 높은 것으로 보아 정수형 곱셈 나눗셈 기능을 지원하는 M 확장 ISA를 지원하지 않기 때문으로 판단된다. 따라서, 정수형 곱셈 나눗셈을 지원하는 M 확장을 추가로 구현한다면, DMIPS 성능을 개선할 수 있을 것으로 예상된다.

5. 결 론

본 논문에서는 RV32I ISA를 기반으로 32-bit 5단계 파이프라인 프로세서를 구현하고, AHB-lite 버스 마스터를 설계하여AHB-lite 버스를 사용하는 UART, GPIO, Timer 등 주변장치를 연결하여 SoC 시스템을 구성하였다. Arty A7-35T FPGA 보드를 사용해 합성하여 하드웨어 사용량 및 소모 전력을 확인하였고, Dhrystone 벤치마크를 통해 제안된 프로세서의 성능을 검증하였다. 추후 연구로 확장 명령어 집합 M을 구현하여 프로세서의 벤치마크 성능을 향상하고, 캐시를 구현하여 데이터 버스의 레이턴시에 의한 성능저하를 개선하고자 한다. 또한, CSR과 특권 명령어 집합을 추가로 구현하여 인터럽트나 RTOS 등과 같은 기능을 지원하도록 할 예정이다.

감사의 글

본 연구는 2022학년도 상명대학교 교내연구비를 지원받아 수행하였음.

참고문헌

1. A. Waterman, K. Asanovi, and RISC-V International, “The RISC-V instruction set manual, Volume I: User-Level ISA, document version 20191213”, 2019.
2. A. Waterman, K. Asanovi, and RISC-V International, “The RISC-V instruction set manual, Volume II: Privileged Architecture, document version 20211203”, 2021.
3. Asanović, Krste, and David A. Patterson, “Instruction sets should be free: The case for risc-v,” EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS- 2014-146, 2014.
4. Reinhold P. Weicker, “Dhrystone: A synthetic systems programming benchmark,” Communications of the ACM, Vol.27, No.10, pp.1013-1030, 1984.
5. S. Park, C. Park, “Implementation of GPU Acceleration of Object Detection Application with Drone Video”, Journal of the Semiconductor & Display Technology, vol. 20, no. 3, pp. 117–119, 2021.
6. C. Park, “Performance Analysis of DNN inference using OpenCV Built in CPU and GPU Functions”, Journal of the Semiconductor & Display Technology, vol. 21, no. 1, pp. 75–78, 2022.
7. K. Asanovic, et al., “The rocket chip generator,” EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17, 2016.
8. Asanović, Krste; et al., rocket-chip. Retrieved October, 5, 2022, from https://github.com/chipsalliance/rocket-chip
9. Celio, Christopher., riscv-boom. Retrieved October, 5, 2022, from https://github.com/riscv-boom/riscv-boom
10. SpainHDL, VexRiscv. Retrieved October, 5, 2022, from https://github.com/SpinalHDL/VexRiscv
11. S. Jang, S. Park, G. Kwon, T. Suh, “Design and Evaluation of 32-Bit RISC-V Processor Using FPGA”, KIPS Transactions on Computer and Communication Systems, vol. 11, no. 1, pp. 1–8, Jan. 2022.
12. PULP Platform, Retrieved October, 5, 2022, from https://pulp-platform.org
13. Pulpino. Retrieved October, 5, 2022, from https://github.com/pulp-platform/pulpino
14. ARM IHI 0011A, “AMBA Specification (Rev 2.0)”, 1999.
15. ARM, DesignStart. Retrieved October, 5, 2022, from http://www.arm.com/products/processors/designstartprocessor-ip/index.php
16. ARM IHI 0033A, “AMBA 3 AHB-Lite Protocol v1.0 specification”, 2006.
17. Harris, S.L. and Harris, D.M., Digital Design and Computer Architecture RISC-V Edition, Morgan Kaufmann, pp. 393-497, (2021).
18. Arm Limited Arm Cortex-M Processor Comparison Table. Retrieved October, 5, 2022, from https://developer.arm.com/documentation/102787/0100.

접수일: 2022년 11월 1일, 심사일: 2022년 12월 8일, 게재확정일: 2022년 12월 13일