반도체디스플레이기술학회지 제22권 제4호(2023년 12월)
Journal of the Semiconductor & Display Technology, Vol. 22, No. 4. December 2023.

임베디드 환경에서의 32-bit RISC-V RV32IM 파이프라인 프로세서 설계 및 구현

박수빈*·김용우*†
*†상명대학교 시스템반도체공학과

A Design and Implementation of 32-bit RISC-V RV32IM Pipelined Processor in Embedded Systems

Subin Park* and Yongwoo Kim*†
*†Department of System Semiconductor Engineering, Sangmyung University

ABSTRACT

Recently, demand for embedded systems requiring low power and high specifications has been increasing, and RISC-V processors are being widely applied. RISC-V, a RISC-based open instruction set architecture (ISA), has been developed and researched by UC Berkeley and other researchers since 2010. RV32I ISA is sufficient to support integer operations such as addition and subtraction instructions, but M-extension should be defined for multiplication and division instructions. This paper proposes an RV32I, RV32IM processor, and indicates benchmark performance scores compared to an existing processor. Additionally, A non-stalling method was proposed to support a 2-stage pipelined DSP multiplier to the 5-stage pipelined RV32IM processor. Proposed RV32I and RV32IM processors satisfied a maximum operating frequency of 50 MHz on Artix-7 FPGA. The performance of the proposed processors was verified using benchmark programs from Dhrystone and Coremark. As a result, the Coremark benchmark results of the proposed processor showed that it outperformed the existing RV32IM processor by 23.91%.

Key Words : RISC-V, Processor, RV32I, RV32IM, 5-stage pipeline, FPGA

1. 서 론

최근 시스템반도체가 발전함에 따라 저전력, 고사양을 요구하는 임베디드 시스템에 대한 수요가 증가하고 있다. 임베디드 프로세서는 RISC (Reduced Instruction Set Computer) 기반의 ISA (Instruction Set Architecture)를 사용하며 ARM과 RISC-V는 대표적인 RISC 기반의 명령어 체계이다. 이 중 RISC-V는 2010년에 UC 버클리에서 개발되어 현재까지 활발히 연구되고 있는 무료 개방형 ISA이다. RISC-V는 32-bit, 64-bit, 128-bit 기반으로 구분되며 M, C, A, F 등의 확장 명령어 집합을 가지고 있어 사용자의 필요에 따라 맞춤형 프로세서를 설계할 수 있다[1]. 그중 RV32I는 메모리 접근 명령어, 분기 명령어, 정수형 연산 명령어와 같은 필수 명령어를 포함하는 ISA이다. RV32I 이후에 고려해야 하는 확장 옵션은 곱셈과 나눗셈 연산 명령어를 포함하는 M 확장 집합 명령어이다.

본 연구에서는 5단계 파이프라인 기반 RV32I 및 RV32IM 프로세서를 설계하였다. 곱셈 연산을 지원하기 위해 Xilinx DSP IP[2]를 사용하였고 파이프라인 단계에서 곱셈 연산을 최적화하기 위해 non-stalling 기법을 제안하였다. 더불어, Artix-7 시리즈의 Nexys A7-100T FPGA (Field Programmable Gate Array)를 이용하여 설계한 두 프로세서의 벤치마크 성능을 평가하였다. RV32I 및 RV32IM 프로세서의 성능 평가를 위해 DMIPS (Dhrystone Millions of Instructions

†E-mail: yongwoo.kim@smu.ac.kr

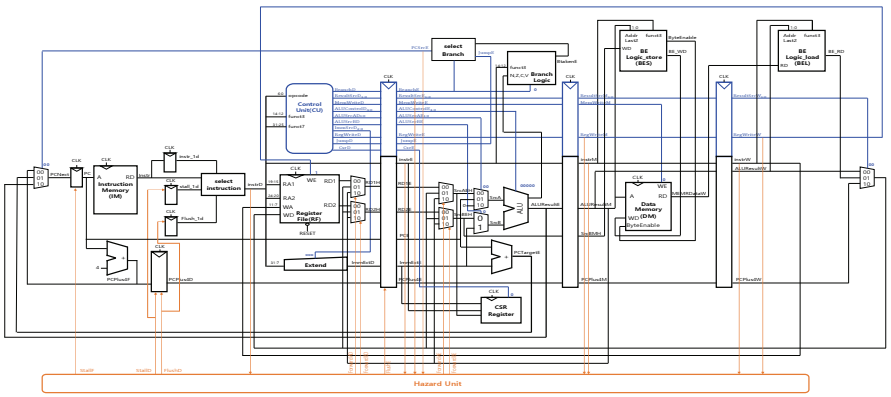

Fig. 1. A block diagram of five stage pipelined RV32I processor.

Per Second) 및 Coremark의 벤치마크 점수를 확인하였고 선행 연구[3,4,5]된 프로세서와 성능을 비교하였다.

본 연구의 구성은 다음과 같다. 2장에서는 본 연구와 관련된 연구에 대해 설명한다. 3장에서는 제안하는 RV32IM 프로세서 구조를 설명하며 제안 기법 유무에 따른 신호 흐름의 변화 및 SoC (System on Chip) 시스템을 나타낸다. 4장에서는 프로세서의 성능을 나타내고 기존 연구와 비교하며 5장에서는 결론 및 추후 연구에 대해 설명한다.

2. 관련 연구

2.1 RISC-V 프로세서

RISC-V ISA가 공개됨에 따라, 엔비디아에서 제안한 딥러닝 가속기를 RISC-V 기반 SoC에 접목시켜 실시간 임베디드 시스템을 위한 연구가 진행되었다[6]. 또한, 국내외에서 여러 프로그래밍 언어를 사용하여 RISC-V 기반 프로세서 개발을 진행하고 있다. ARM의 Cortex-M4[7]는 FPGA에서 사용 가능한 프로세서이지만 무료 개방형 소스가 아니라는 단점을 갖는다. RISC-V ISA를 개발한 UC-Berkely는 합성 가능한 RTL을 생성하는 Rocket chip generator을 개발하였다. Rocket chip generator가 제공하는 코어는 Chisel 언어로 구성된 ROCKET 과 BOOM 코어로 구분된다[8]. Chisel은 스칼라 언어를 사용하고 있지만, VerilogHDL 및 SystemVerilog 언어와 비교하여 확장성이 낮다는 단점을 갖는다.

3. 제안 기법

본 장에서는 5단계 파이프라인 기반 RV32I 및 RV32IM 프로세서의 구조와 곱셈 명령어를 지원하기 위해 제안하는 non-stalling 기법을 설명한다. 또한, 사용되는 곱셈기 및 나눗셈기 구조 및 주변장치를 연결한 SoC 시스템을 설명한다.

3.1 RV32I 프로세서 대비 RV32IM 프로세서 구조 변화

Fig.1은 5단계 RV32I 프로세서[9]를 기반으로 재설계한 RV32I 프로세서 구조를 나타내며 FPGA에서의 구현을 위해 동기 메모리를 사용한 구조를 보인다. 프로세서는 37개의 명령어를 지원하며 RV32I ISA에서 FENCE, ECALL, EBREAK 명령어를 제외하였다. 5단계 파이프라인 구조는 다음과 같다. IF (Instruction Fetch) 단계에서 메모리로부터 명령어를 가져오고, ID (Instruction Decode) 단계에서 명령어를 해석한다. 해석된 명령어로부터 컨트롤 유닛을 통해 제어 신호가 생성되고 EXE (Execute) 단계에서 ALU (Arithmetic Logic Unit)를 통해 연산이 수행된다. MEM (Memory) 단계에서 데이터 메모리에 접근하여 데이터를 쓰거나 읽고 WB (WriteBack) 단계에서 레지스터 파일을 업데이트한다. 프로세서의 정상 동작 검증을 위해 특권 명령어 집합에 해당하는 CSR (Control Status Register) 명령어를 추가적으로 구현하였다[10].

Fig. 2는 제안하는 5단계 파이프라인 기반 RV32IM 프로세서 구조를 나타내며 Fig. 1의 RV32I 대비 RV32IM 프로세서를 설계하기 위해 추가한 모듈을 강조한다. 제안하는 프로세서 구조의 단순화를 위해 CSR 레지스터와 같이 곱셈 및 나눗셈 연산에 직접적으로 영향을 미치지 않는 모듈은 배제하고 도식화하여 Fig. 2를 나타내었다. RV32IM은 RV32I의 37개 명령어에서 MUL, MULH, MULHSU, MULHU, DIV, DIVU, REM, REMU 명령어를 추가 지원하여 총 45개를 지원한다.

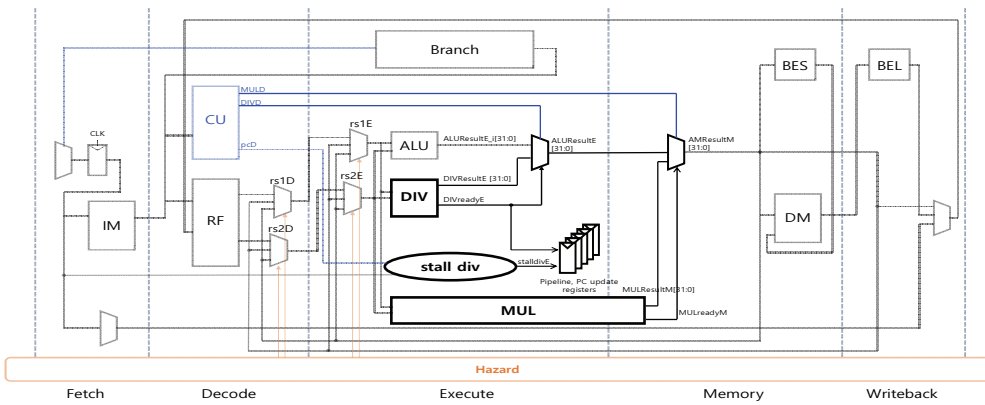

Fig. 2. Proposed RV32IM processor block diagram highlighting additional blocks compared to RV32I processor.

곱셈 연산을 지원하기 위해 두 사이클 (two cycle) 기반의 곱셈기를 사용하였다. Fig. 3은 두 사이클 기반 곱셈 연산을 파이프라인 구조에서 지원하였을 때 EXE 단계의 지연 유무에 따라 소요되는 클럭 사이클을 나타낸다. Fig. 3(a)의 구조는 곱셈 연산이 시작되는 EXE 단계를 연산이 종료될 때까지 유지하여 파이프라인 구조를 유지한다. 이는 곱셈 연산이 진행될 때마다 지연이 필요한 구조로 전체 성능 저하에 영향을 미치기 때문에 최적화가 필요하다. 따라서 Fig. 3(b)와 같이 곱셈 연산이 진행될 때 EXE 단계를 지연하지 않고 MEM 단계가 실행되는 non-stalling 기법을 제안한다.

Fig. 2는 곱셈 연산을 위해 non-stalling 기법, 나눗셈 연산을 위해 복원 알고리즘의 나눗셈기가 사용된 프로세서 구조를 보여준다. EXE 단계에서 연산이 시작 및 종료되는 ALU와 나눗셈기를 통해 ALUResultE 값을 결정한다. 곱셈 연산은 MEM 단계에서 종료되므로 ALUResultM 값과 MULResultM 값을 비교하여 최종 연산 결과에 해당하는 AMResultM 값을 결정한 후 메모리에 접근한다. 두 사이클 기반 곱셈기를 지원하며 발생하는 해저드를 해결하기 위해 세 가지 연산 결과가 모두 반영된 AMResultM 값과 AMResultW 값을 ID 및 EXE 단계로 포워딩 하였다.

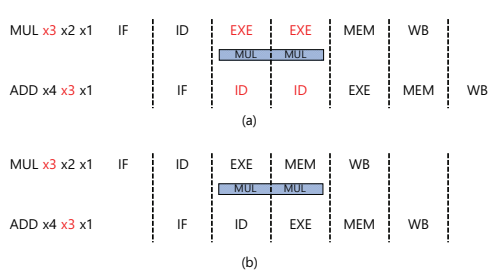

Fig. 3. Pipeline operation flow of a) stalling method used b) non-stalling method used in multiplication operation.

3.2 제안하는 곱셈기 및 나눗셈기 구조

본 연구에서는 두 사이클 기반 32-bit 곱셈 연산을 위해 Xilinx의 1 단계 파이프라인 DSP IP[2]를 사용하였고 나눗셈 연산을 위해서는 복원 알고리즘[11]이 적용된 나눗셈기를 사용하였다. Fig. 4와 Fig. 5는 각각 제안하는 곱셈기와 나눗셈기 구조를 나타낸다. 연산 시작을 위한 시작 (start) 신호의 생성은 컨트롤 신호 생성기 (control signal generator)에서 진행되며, 시작 신호 생성을 위해 pcE 값을 한 클럭 지연하여 pcE_before 값을 생성하였다. 또한, opcode가 R-type을 나타내고 funct3E의 최상위 비트 값이 0인 경우 현재 실행 명령어가 곱셈에 관한 명령어임을 나타내므로 MULE 신호를 활성화하였고 1인 경우 DIVE 신호를 활성화하였다. MULE 혹은 DIVE 신호가 활성화되었을 때, pcE 값과 pcE_before 값이 다를 경우에 시작 신호를 활성화하였다.

시작 신호가 활성화되면 funct3E, 연산자 rs1E, rs2E 값에 따라 부호가 없는 두 연산자가 결정된다. 곱셈기의 경우, 두 연산자가 DSP를 통해 연산 될 때 funct3E, rs1E, rs2E 값을 한 사이클 지연시켰고 해당 값에 따라 DSP로부터 얻는 64-bit 곱셈 값에서 상위 혹은 하위 32-bit와 부호 활성화 여부를 결정하여 최종 연산 값을 도출하였다. 두 사이클 기반의 곱셈기를 사용하므로 시작 신호를 한 사이클 지연시켜 MULreadyM 신호를 생성하였다. 나눗셈 연산의 경우, 복원 나눗셈 알고리즘을 사용하므로 32 사이클이 소요된다. 복원 나눗셈기로부터 발생하는 64-bit 연산 값에서 곱셈기와 마찬가지로 32-bit 값의 최종 연산 값을 나타내야 하므로 funct3E, rs1E, rs2E 값을 DIVreadyE 신호가 활성화되기 전까지 지연시켰다. DIVreadyE 신호는 시작 신호가 활성화된 후 32 사이클 뒤에 활성화되는 카운터를 통해 생성하였다.

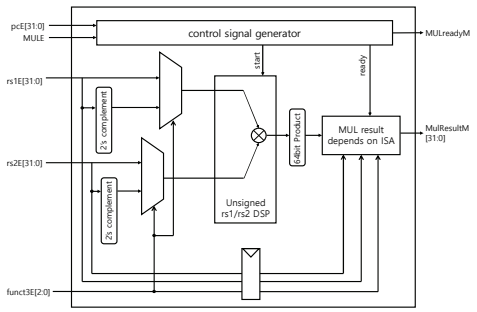

Fig. 4. Proposed DSP IP-based multiplier structure.

나눗셈 연산 시 32 사이클 지연은 파이프라인 구조에도 적용되어야 하므로 Fig. 2에서 보는 바와 같이 stall div 모듈을 제안하였다. stalldivE 신호는 앞서 설명된 시작 신호 발생 조건과 동일하게 생성된다. stalldivE 신호가 발생하면 5단계 파이프라인 단계를 구분 짓는 파이프라인 레지스터 및 PC 업데이트 레지스터 값을 DIVreadyE 신호가 활성화되기 전까지 유지하였다. 나눗셈 연산을 위해 32 사이클이 소요되고 DIVreadyE 신호가 활성화된 후 EXE 단계가 종료되므로 EXE 단계는 총 33 사이클이 소요된다. RISC-V 나눗셈 연산에는 두 가지 조건에 따른 예외[1] 처리가 요구되므로 나눗셈 연산 이후 32-bit 결과값을 나타낼 때 반영하였다.

3.3 제안하는 SoC 구조

제안하는 프로세서는 두 개의 메모리 중 하나를 선택하여 사용한다. 첫 번째 메모리는 사용자 프로그램에 접근하는 메모리이며 또 다른 메모리는 BIOS (Basic Input/Output System) 프로그램을 위한 메모리이다. BIOS는 UART를 이용하여 사용자 프로그램을 다운을 수 있도록 하는 소프트웨어를 의미한다. Fig. 6의 SoC 시스템에는 RV32IM 파이프라인 프로세서와 사용자 메모리, BIOS

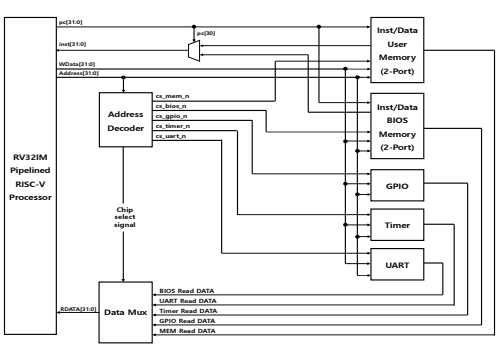

Fig. 6. The block diagram of proposed RV32IM pipelined RISC-V processor and SoC system.

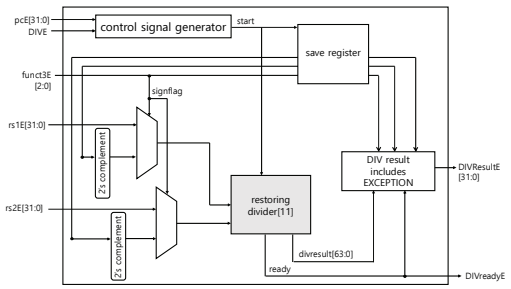

Fig. 5. Proposed block diagram of divider.

메모리가 있고 주소 디코더와 데이터 멀티플렉서가 이용되어 GPIO (General Purpose Input/Output), UART (Universal asynchronous receiver and transmitter), 타이머 등 주변장치가 연결된다. BIOS 및 모든 사용자 프로그램은 GCC 컴파일러를 통해 빌드를 수행하고 기계어를 생성하였다.

4. 성능 평가

본 장에서는 제안하는 RV32I 및 RV32IM 프로세서의 성능을 확인하고 선행 연구된 RISC-V 프로세서와 벤치마크 점수를 비교한다. 성능 측정을 위해 Vivado 2019.02 버전을 이용하여 합성 및 구현을 진행하였다. 최적화를 위해 RV32IM 프로세서의 합성 (Synthesis) 및 구현 (Implementation) 옵션을 Flow Perfoptimized high, Performance ExtraTimingOpt으로 설정하였다.

Table 1은 RV32I 및 RV32IM 프로세서의 LUT(Lookup Table), FF(FlipFlop), BRAM(Block RAM), DSP 등 하드웨어 구현 시 사용되는 로직 사용량과 최대 동작 주파수 및 전력 사용량을 보여준다. Table 1의 로직 사용량 중 LUT의 경우, 유효한 63,400개에서 RV32I는 2,083개, RV32IM은 3,143개의 LUT가 사용되어 구현되었다. RV32IM 프로세서는 RV32I 대비 M 집합 확장 명령어 구현을 위한 로직이 추가되므로 RV32I 대비 로직 사용량 및 전력 사용량이 증가한다. 더불어, RV32I와 RV32IM 프로세서의 최대 동작 주파수가 각각 53.38 MHz 및 50.03 MHz임을 확인하였다.

Table 1. Hardware implementation results of proposed RV32I and RV32IM processor

Resource	Available	Utilization	
		RV32I	RV32IM
LUT	63,400	2,083(3.29%)	3,143(4.96%)
FF	126,800	2,094(1.65%)	2,369(1.87%)
BRAM	135	68(50.37%)	68(50.37%)
DSP	240	0(0%)	4(1.67%)
Max Freq.	-	53.38MHz	50.03MHz
Power(W)	-	0.225	0.206

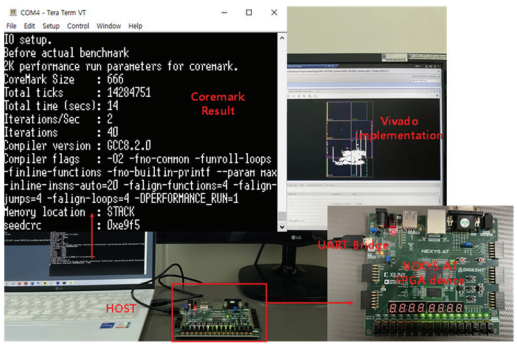

Fig. 7. FPGA verification of SoC system.

Table 2. Benchmark results of proposed RISC-V RV32I processor and other RV32I processors

Benchmark	ISA	DMIPS	CoreMark/MHz
RVCoreP[4]	RV32I	1.03	0.84
VexRiscv (small) [3]	RV32I	0.52	-
Proposed RV32I	RV32I	0.58	1.00

본 연구에서는 제안한 프로세서의 성능 확인을 위해 벤치마크 프로그램을 적용하였다. Fig. 7은 Xilinx의 Artix-7 Nexys A7-100T FPGA 보드를 사용하여 RV32IM 프로세서의 Coremark 프로그램을 통한 성능 측정 결과를 UART를 통해 나타내는 과정을 보인다. GCC 8.2.0 버전에 O2 최적화 옵션을 적용하였으며 반복 횟수는 40번으로 지정하였다. Coremark 점수는 반복 횟수인 40회에서 프로그램 처리시간인 14를 나누어 2.85임을 확인하였다. Dhrystone의 경우, 2.2 버전에 O3 최적화 옵션을 적용하였고 반복 횟수는 1000번으로 지정하였다.

Table 3. Benchmark results of proposed RISC-V RV32IM processor and other RV32IM processors

Benchmark	ISA	DMIPS	CoreMark/MHz
RVCoreP (radix-4) [5]	RV32IM	1.25	1.69
VexRiscv (full) [3]	RV32IM	1.21	2.30
Proposed RV32IM	RV32IM	0.62	2.85

Table 2와 Table 3은 제안하는 프로세서와 기존 연구된 RISC-V 프로세서의 벤치마크 점수를 보여주며 각각 RV32I 및 RV32IM ISA를 지원하는 프로세서를 나타낸다. 지원하는 ISA에 의해 명령어 개수의 차이가 발생하므로 RV32I 및 RV32IM 프로세서는 성능 차이를 갖는다. Table 2를 통해 RV32I 프로세서의 DMIPS는 0.58로 small 옵션의 VexRiscv[3] DMIPS에 비해 약 11.53% 높지만 RVCoreP[4]에 비해서는 약 43.68% 낮은 성능을 보이는 것을 확인하였다. 이는 세 프로세서 중 RVcoreP 프로세서만 분기 예측기를 지원하기 때문으로 예상된다. Table 3은 제안하는 RV32IM 프로세서의 Coremark 점수는 RVCoreP[5] 대비 약 68.63%, full 옵션의 VexRiscv[3] 대비 약 23.91% 높음을 보인다. 두 명령어 사이의 명령어 결과를 다음 명령어에서 사용할 때 의존성이 발생하며, 앞선 명령어와 의존성이 있을 경우, VexRiscv는 곱셈과 나눗셈 지연에 각각 2, 34 사이클을 갖고, RVCoreP는 18, 35 사이클을 갖는다. 반면에 제안하는 RV32IM 프로세서는 곱셈에 의해 지연되는 시간이 없고 나눗셈에는 총 32 사이클이 소요되었기에, 이 사이클의 차이로 Coremark의 점수가 기존 연구 대비 향상된 것으로 예상할 수 있다. RV32IM ISA의 VexRiscv 및 RVCoreP의 DMIPS는 제안하는 RV32IM 프로세서의 DMIPS에 비해 높은데 두 프로세서 모두 분기 예측기를 지원하기 때문이라고 추측한다. 추후 연구에서 제안하는 프로세서에 분기 예측기를 추가 지원하면 Dhrystone 벤치마크 성능을 향상할 수 있을 것으로 기대한다.

5. 결 론

본 연구에서는 RV32I 프로세서 및 RV32I에 기반하여 M 집합 확장 명령어를 추가적으로 지원하는 RV32IM 프로세서를 설계하였고 5단계 파이프라인 구조에 곱셈 연산을 최적화하기 위해 non-stalling 기법을 적용하였다. 제안한 RV32IM RISC-V 프로세서와 주변장치를 연결하여 SoC 시스템을 구성하였고 Nexys A7-100T FPGA를 이용하여 합성 및 구현을 진행하였다. Coremark 벤치마크 점수를 통해 선행 연구된 RV32IM 프로세서 대비 제안하는 RV32IM 프로세서의 성능 향상을 확인하였다. 추후 연구로 분기 예측기를 추가 구현하여 프로세서의 성능 개선을 위한 연구를 수행할 예정이다.

감사의 글

본 연구는 2023학년도 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. Vivado Design Suite, “Multiplier v12.0 LogiCORE IP Product Guide”, 2015.
3. SpainHDL, VexRiscv. Retrieved November, 17, 2023, from https://github.com/SpinalHDL/VexRiscv.
4. H. Miyazaki, T. Kanamori, M. Ashraful, K. KISE, “RVCoreP: An Optimized RISC-V Soft Processor of Five-Stage Pipelining”, Retrieved November, 17, 2023, from https://arxiv.org/abs/2002.03568.
5. M. Ashraful, H. Miyazaki, K. KISE, “RVCoreP-32IM: An effective architecture to implement mul/div instructions for five stage RISC-V soft processors”, Retrieved November, 17, 2023, from https://arxiv.org/abs/2010.16171.
6. F. Farshchi, Q. Huang and H. Yun, "Integrating NVIDIA Deep Learning Accelerator (NVDLA) with RISC-V SoC on FireSim," 2019 2nd Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications (EMC2), Washington, DC, USA, pp. 21-25, 2019.
7. ARM DDI 0439B, “Cortex-M4 Revision r0p0 Technical Reference Manual”, 2010.
8. K. Asanovic, et al., “The rocket chip generator,” EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17, 2016.
9. S. Jo, J. Lee, Y. Kim, “A Design and Implementation of 32-bit Five-Stage RISC-V Processor Using FPGA”, Journal of the Semiconductor & Display Technology, vol. 21, no. 4, pp. 27–32, 2022.
10. A. Menon, S. Kim, Y. Chen, “EECS151/251A Spring 2022 Final Project Specification RISCV151, document version 1.0”, 2022.
11. U. S. Patankar, A. Koel, "Review of basic classes of dividers based on division algorithm", IEEE Access, vol. 9, pp. 23035 - 23069, 2021.

접수일: 2023년 11월 23일, 심사일: 2023년 12월 12일, 게재확정일: 2023년 12월 14일