The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 23, No. 3, pp.153-158, Jun. 30, 2023. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2023.23.3.153
JIIBC 2023-3-21

파이프라인식 비순차실행 수퍼스칼라 프로세서의 FPGA 설계 및 구현

FPGA Design and Implementation of A Pipelined Out-of-Order Superscalar Processor

이종복*

Jongbok Lee*

요 약 국내에서 시스템반도체 설계의 중요성이 대두되고 있으며, 메모리 반도체 설계 기술과의 균형있는 발전을 도모해야 한다. Xilinx에서 제공하는 Vivado 통합 환경 도구를 이용하여 짧은 시간에 큰 비용을 들이지 않고 프로세서를 Xilinx FPGA 반도체 칩에 구현할 수 있다. 본 논문에서는 레코드 자료구조를 지원하여 효율적으로 디지털 시스템을 설계할 수 있는 VHDL을 이용하여 32 비트 ARM 명령어를 실행할 수 있는 파이프라인식 비순차실행 수퍼스칼라 프로세서를 설계하였다. Vivado에서 광범위한 시뮬레이션을 수행한 후에, Xilinx FPGA로 합성, 구현 및 로직아날라이저로 검증하였다. 그 결과, 파이프라인식 비순차실행 수퍼스칼라 프로세서가 FGPA에서 성공적으로 동작하였다.

Abstract Domestically, the importance of system semiconductor design is increasing, and the balanced development with the high-end memory semiconductors should be promoted. Using Xilinx Vivado as a development enivronment tool, it reduces time and cost dramatically in implementing the processor on FPGA. In this paper, the VHDL language which provides record data structure for an efficient digital system design is used for designing a pipelined out-of-order superscalar processor. It has been simulated extensively, synthesized and implemented on FPGA and verified by Integrated Logic Analyzer. As a result, the pipelined out-of-order superscalar processor could be executed successfully.

Key Words : FPGA, Out-of-order, Superscalar, VHDL

Ⅰ. 서 론

최근에 인공반도체가 널리 각광을 받고 있지만, 데스트탑 컴퓨터, 노트북, 스마트폰, 임베디드 시스템에 탑재되는 기존의 범용 프로세서에 대한 국내의 설계 기술 확보가 무엇보다 중요하다. 대한민국은 메모리 반도체 설계 및 공정 기술이 세계 최고의 수준이지만, 시스템반도체 중에서 특히 프로세서 설계는 아직 그렇지 못한 형편이다. 따라서 최근 각광을 받고 있는 인공지능 반도체와 아울러, 멀티코어 프로세서의 기본단위인 범용 수퍼스칼라 프로세서 설계 분야에 대한 연구를 활발히 하고 박차를 가할 필요가 있다.

1993년의 Intel은 수퍼스칼라 프로세서인 Pentium을 개발했으며, 두 개의 명령어를 동시에 실행할 수 있었

*정회원, 한성대학교 전자 및 시스템반도체트랙
접수일자 2023년 5월 27일, 수정완료 2023년 6월 3일
게재확정일자 2023년 6월 9일

Received: 27 May, 2023 / Revised: 3 June, 2023 /
Accepted: 9 June, 2023
*Corresponding Author: jblee@hansung.ac.kr
School of ME Engineering, Hansung University, Korea

다. 1995년의 Pentium Pro는 수퍼스칼라 설계로 싸이클 당 3 개의 명령을 발행하고 실행할 수 있었다. 1997년의 펜티엄 II는 최대 4 개의 명령을 발행하고 실행할 수 있었으며, 더 나은 성능을 위해 분기 예측 향상 기능을 추가했다.

2006년 Intel에서 개발한 Core 2 Duo 프로세서는 2 개의 명령어를 실행하는 수퍼스칼라 코어 2 개를 이용하여 멀티코어프로세서를 구축하기 시작했다. 2011년 Intel의 Sandy Bridge 마이크로아키텍처는 수퍼스칼라 프로세서를 크게 개선했다. 싸이클 당 4 개의 명령어를 실행했으며, 향상된 분기 예측 및 보다 효율적인 비순차적 실행 엔진이 특징으로 쿼드코어를 구성했다. 2013년 Intel의 Haswell 마이크로아키텍처는 수퍼스칼라 프로세서를 더욱 향상시켰다. 싸이클 당 4 개의 명령어를 처리하고 분기 예측 기능이 개선되었으며, 비순차 실행 엔진이 최적화되었다. 2015년 Intel의 Skylake, Kaby Lake, Coffee Lake 등을 포함한 후속 세대 프로세서는 계속해서 수퍼스칼라 설계를 개선해 왔다. 수퍼스칼라 코어 프로세서는 싸이클 당 4 개에서 6 개의 명령어를 실행하였고, 4 개, 6 개, 8 개의 코어를 이용하여 멀티코어프로세서 시스템을 구성했다.

Intel 외 널리 쓰이는 ARM과 Apple사가 개발한 프로세서를 고찰하면 다음과 같다. 2016년도에 개발된 ARM의 Cortex-A72 수퍼스칼라 프로세서는 쿼드코어 형태로 라즈베리파이 4에 장착되어 널리 쓰이고 있다. 이 프로세서는 싸이클 당 3 개의 명령어, 향상된 분기 예측 기능 및 이전 제품에 비해 향상된 성능을 특징으로 한다. 2020년의 ARM 아키텍처를 기반으로 하는 Apple의 M1 칩에 장착된 수퍼 스칼라 프로세서는 싸이클 당 8 개의 명령어 처리가 특징이다. M1 칩은 고급 명령 스케줄링, 비순차적 실행, 개선된 분기 예측 기능을 갖춘 8 개의 수퍼스칼라 코어로 멀티코어 프로세서를 구축했다.

이상 살펴본 바와 같이, 오늘날 서버와 데스크탑 컴퓨터, 스마트폰 및 비롯한 여러 가지 임베디드 장치에 멀티코어프로세서를 활용하고 있으며, 이러한 멀티코어프로세서를 구성하는 기본 프로세서 코어단위는 모두 수퍼스칼라 프로세서이다.

본 논문에서는 VHDL을 이용하여 싸이클 당 2 개의 명령어를 실행하고 6 단계 파이프라인을 갖는 32 비트 비순차실행 수퍼스칼라 프로세서를 설계하였으며, FPGA에 구현하고 검증하였다. 본 프로세서의 설계에서 명령어 집합으로 임베디드 시스템에 널리 쓰이는 ARM을 채택하였다.

본 논문은 다음과 같이 구성된다. 2 장에서 수퍼스칼라 프로세서 구조의 특징과 아울러 명령어 및 파이프라인 단계별 동작을 살펴 고찰한다. 3 장에서 수퍼스칼라 프로세서의 구현 환경 및 FPGA 합성을 자세히 살펴보며, 4 장에서 FPGA에 대한 로직아날라이저 검증 결과를 보이고, 5 장에서 결론을 맺는다.

II. 파이프라인식 비순차실행 수퍼스칼라 프로세서의 구조

본 논문에서는 그림 1과 같이 한 싸이클에 2 개의 명령어를 처리할 수 있는 파이프라인식 비순차실행 수퍼스칼라를 Tomasulo 알고리즘을 적용하여 설계하였다[1-5]. 본 프로세서는 명령어 메모리, 데이터 메모리, 레지스터 화일, 예약스테이션 (Reservation Station), 재배열 버퍼 (Reorder Buffer), 로드와 스토어 장치 각 1 개, 정수형 연산장치 2 개, 정수형 곱셈기 1 개, 분기장치 1 개로 구성된다. 이 때, 예약스테이션은 각 연산장치와 결합되므로 그 개수는 연산장치의 개수와 같고, 재배열버퍼는 최대 명령어 16 개를 할당할 수 있는 크기로 설계했다.

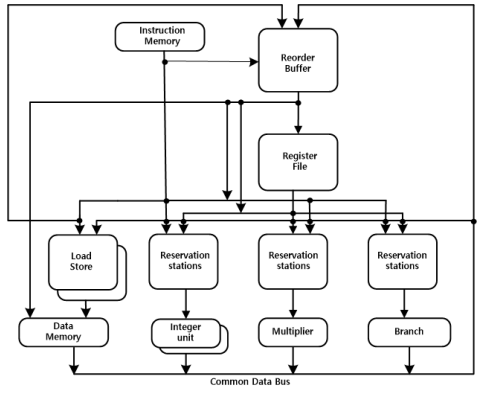

그림 1. 파이프라인식 비순차실행 수퍼스칼라 프로세서의 블럭도
Fig. 1. Block diagram of the pipelined out-of-order superscalar processor

예약스테이션은 연산장치와 결합된 장치로서, 예약스테이션에 할당된 명령어의 피연산자가 준비되면 연산을 수행하고, 그 값을 필요로 하는 다른 예약스테이션의 명령어에 송부한다. 또한, 아직 피연산자가 준비되지 않은 명령어는 그 값을 공급할 예약스테이션을 가리키도록 한다. 따라서, 예약스테이션을 통하여 동적 스케줄링에 필

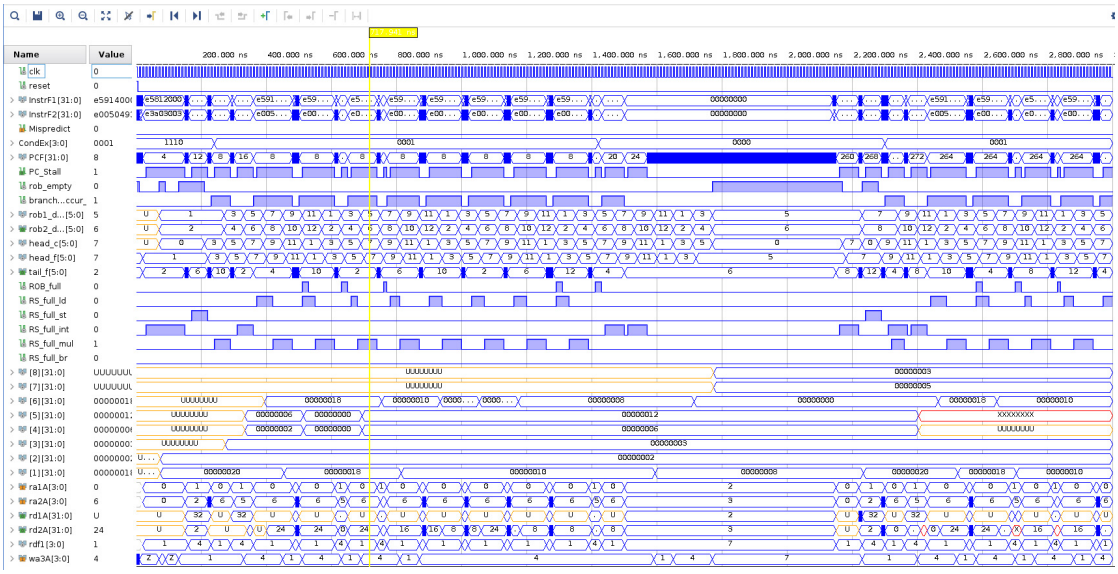

그림 2. 수퍼스칼라 프로세서의 시뮬레이션 결과
Fig. 2. Simulation result of the superscalar processor

요한 레지스터 재명명 기능이 수행된다.

재정렬버퍼는 명령어들의 비순차실행 (Out-of-Order Execution)을 허용하면서도 순차완료 (In-Order Completion) 되도록 하기 위하여 필요하다. 이것을 위하여, 재정렬버퍼는 실행이 종료했으나, 아직 완료하지 않은 명령어들을 프로그램의 원순서대로 유지하고 레지스터화일에 기록하면서 삭제한다. 따라서, 아직 레지스터화일에 완료를 하지 않은 명령어의 연산결과는, 이 값을 필요로하는 예약스테이션의 명령어의 피연산자로 공급된다.

데이터종속이 없을 경우, 한 싸이클당 최대 2 개의 명령어가 예약스테이션을 통하여 연산유닛에서 비순차실행된다. 다음은 토마술로 알고리즘의 주요 6 단계를 기술한다.

1. 인출 (fetch)과 발행 (issue) 단계

수퍼스칼라 프로세서는 명령어 메모리로부터 다음 2 개의 명령어를 인출 받는다. 만일에 부합하는 예약스테이션과 재정렬버퍼가 모두 비어있으면, 2 개의 명령어를 예약스테이션과 재정렬버퍼에 동시에 삽입한다. 만일에 예약스테이션 또는 재정렬버퍼가 비어있지 않으면 구조적 해저드 (structural hazard)가 발생한 것으로서, 새로운 명령어의 인출과 발행을 멈춘다 (stall). 그러나, 실행단계 이하의 파이프라인은 계속 동작하여 이전에 인출한 명령어들을 처리함으로써 예약스테이션과 재정렬버퍼를 비울 수 있도록 한다.

수퍼스칼라가 한 싸이클에 동시에 인출한 명령어 2 개를 각각 명령어 1 과 명령어 2 라고 할 때, 명령어 1과 명령어 2 사이의 데이터 종속 여부를 판단해야 한다. 만일 명령어 2의 원천 레지스터 중의 하나가 명령어 1의 목적 레지스터에 종속이라면, 명령어 2의 피연산자는 명령어 1의 결과를 가리키도록 태그를 설정한다. 그러나, 인출된 2 개의 명령어가 서로 독립인 경우, 예약스테이션에 할당하여 연산이 실행된다.

각 명령어가 예약스테이션에서 연산을 하기 위하여 필요한 피연산자를 읽기 위하여, 다음 세 가지의 경우에 따라서 처리한다. 첫 번째는 예약스테이션에서 처리하는 명령어의 연산에 필요한 피연산자가 레지스터화일에 있으므로 그 값을 정상적으로 읽어오는 경우로서, 이것은 즉각 실행된다. 두 번째는 예약스테이션에서 처리하는 명령어의 연산에 필요한 피연산자가 아직 레지스터화일에 그 결과가 기록되지 않고, 재정렬버퍼에 있는 명령어에 있기 때문에 그 결과로부터 읽어오는 경우이다. 세 번째는 예약스테이션에서 처리하는 명령어가 필요한 피연산자가 레지스터는 물론 재정렬버퍼에도 그 값이 아직 준비되지 않는 경우이다. 이때는 필요한 피연산자를 생산할 명령어의 재정렬버퍼 내의 위치인 태그를 설정받았다가, 추후에 연산이 실행되고 해당 피연산자의 결과값이 재정렬버퍼에 등록되면 가지고 있던 태그를 이용하여 그 값을 읽어온다. 예약스테이션을 이용하는 이 과정에

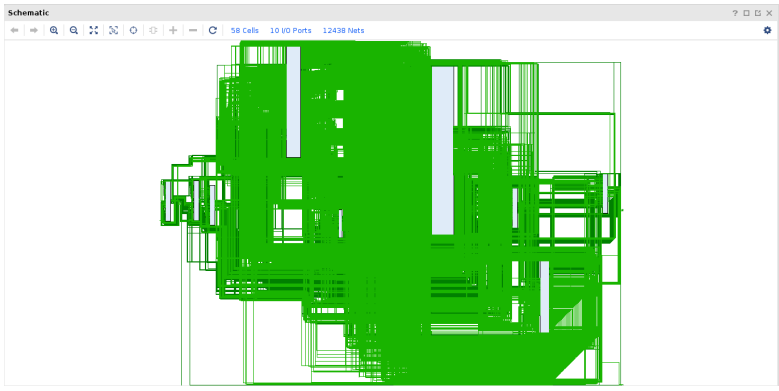

그림 3. 합성된 수퍼스칼라 프로세서
Fig. 3. Synthesized superscalar processor

서, 레지스터가 재명명되어 명령어 간의 종속이 해결된다.

한편, 인출된 명령어가 분기명령어인 경우에는 2 단계 적응형 분기예측방식 (two-level adaptive branch prediction)을 활용하여 다음 명령어를 예측하여 인출한다.

2. 실행 (execute) 및 메모리 접근(memory access) 단계

발행단계에서 살펴본 바와 같이, 만일 피연산자가 준비되어있지 않으면, 필요한 피연산자 값이 공급될 때까지 대기한다. 만일 피연산자가 준비되었으면, 그것을 기다리는 모든 예약스테이션에 공급한다. 모든 피연산자가 준비된 명령어는 순서와 상관없이 각 예약스테이션과 결합된 연산유닛에서 매 싸이클 당 2 개의 연산을 실행할 수 있으므로 수퍼스칼라 비순차실행이 가능하다.

로드 명령어는 실행을 위하여 2단계가 필요하다. 첫 번째는 실행단계로서 유효주소의 계산이며, 이것은 로드버퍼에 삽입된다. 두 번째는 메모리 접근 단계로서 실제로 데이터메모리로부터 데이터를 읽는 단계이다. 스토어 명령어는 이미 데이터메모리에 기록할 데이터를 확보한 상태이므로, 실행단계에서 유효주소의 계산을 한다.

3. 쓰기 (write-back) 및 완료 (commit) 단계

각 연산장치에서 명령어의 실행이 완료되어 결과가 준비되었을 때, 그 결과를 기다리는 재정렬버퍼와 예약스테이션에 각각 전송한다. 스토어의 경우 데이터메모리에 저장할 값이 준비되었으면 재정렬버퍼의 값 영역에 이것을 저장한다. 그러나, 만일에 데이터메모리에 저장해야 할 값이 아직 준비가 되지 않았다면, 그 값이 준비가 될 때까지 대기한다.

완료는 명령어를 마무리하는 마지막 단계로서, 마무리하는 명령어가 일반 명령어, 분기어, 스토어, 로드 중에 어느 것 인가에 따라서 처리가 달라진다. 일반 명령어 또는 로드명령어가 재정렬버퍼의 헤드에 도달하고 그 결과가 준비되었을 때, 그 결과는 최종적으로 레지스터에 기록되고 명령어를 재정렬버퍼에서 삭제함으로써 정상적으로 완료된다. 스토어 명령어를 완료하는 것은 이것과 유사하지만, 단지 데이터가 레지스터가 아닌 메모리에 기록된다는 점이 다르다.

예측오류가 난 분기어가 재정렬버퍼의 헤드에 도달했을 때, 재정렬버퍼 내 분기 이후의 명령어를 모두 삭제하고, 오류가 발생할 것을 대비하여 미리 저장해두었던 분기어의 올바른 후속 명령어부터 인출 및 실행을 다시 재개한다. 반면에, 만일 분기어가 올바르게 예측되었다면 정상적으로 완료된다.

표 1에 본 논문에서 설계에 이용한 ARM 명령어 집합을 나타냈다. ARM 명령어는 크게 산술 논리, 곱셈, 메모리, 분기의 4 가지 유형으로 구성된다. 산술 논리 명령어

표 1. 설계된 수퍼스칼라 프로세서의 명령어 집합
Table 1. The instruction set of the designed superscalar processor

명령어 유형	명령어
산술 논리	AND EOR SUB RSB ADD ADC SBC RSC TST TEQ CMP CMN ORR MOV
곱셈	MUL
메모리	STR LDR
분기	B BL

는 AND, EOR, SUB 등의 14 개 명령어로 구성되며. 곱셈 명령어는 1 개의 명령어로 이루어진다. 메모리 명령어는 STR, LDR과 같이 워드단위의 스토어 및 로드 명령어로 구성되며, 분기 명령어는 B와 BL로 구성된다.

III. 수퍼스칼라 프로세서의 구현 환경 및 FPGA 합성

본 논문연구의 전 구현 과정은 3.07 GHz로 동작하는 Intel Core i7-950 데스크탑 컴퓨터에 CentOS 8.4 운영체제를 설치하여 시행하였다. 설계도구로 Xilinx사의 Vivado 2022.2 버전을, VHDL 컴파일러는 2008 버전을 이용하였다. 예약스테이션과 재정렬버퍼를 효율적으로 구현하기 위하여 VHDL의 레코드 자료구조를 활용하였다. Vivado에서 VHDL로 프로그래밍한 파이프라인 방식의 32 비트 수퍼스칼라 프로세서에 대한 광범위한 시뮬레이션을 수행한 결과를 그림 2에 나타냈다. 끝난 후에 RTL 분석, 합성, 구현, FPGA 프로그램의 순서로 연구를 진행하였다[4,5].

그림 3에 합성된 수퍼스칼라 프로세서의 회로도를 나타냈다. 최상위 레벨 회로도에서 명령어 메모리와 데이터 메모리는 Vivado의 IP-Catalog에서 제공하는 Distributed Memory Generator를 통해 각각 ROM과 Single Port RAM으로 구성했다.

명령어 메모리는 검증을 위한 수퍼스칼라 프로세서 프로그램을 64 비트 16 진수 기계어로 작성하여 COE 파일로 적재하여 초기화했다. 그리고 로직아날라이저를 통한 검증을 위하여 ILA 코어를 포함시켰다.

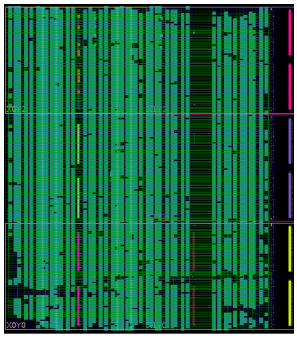

그림 4. FPGA에 구현된 파이프라인식 비순차실행 수퍼스칼라 프로세서
Fig. 4. FPGA implemented pipelined out-of-order superscalar processor

RTL 분석 결과, 본 프로세서는 58 개의 Cell, 10 개의 I/O Port, 12,438 개의 Nets로 구성됐다. 합성에 4 분 31초, 구현에 10분 7초가 소요되었다. 그림 4에 FPGA로 구현된 수퍼스칼라 프로세서를 나타내고 있다. 합성 결과, LUT 53%, LUTRAM 3%, FF 9%, BRAM 3%, DSP 1%, IO 12%, BUFG 12%가 소요되었다. 전력 분석 결과, 총 0.283 W가 소모되었다. 타이밍 분석 결과, 최장 셋업 시간이 14.2ns, 홀드 타임이 0.035ns, 펄스폭이 24.4 ns로 측정되었다.

IV. 구현된 수퍼스칼라 프로세서의 ILA 검증

그림 5에 본 연구의 FPGA 구현 및 실행에 이용된 보드인 국내 휴인스전자의 easySoC PYNQ를 나타냈다. 이 보드에 탑재된 FPGA 보드는 Ultra96-V2이고 FPGA 소자는 xczu3eg-svba484-1-i이다.

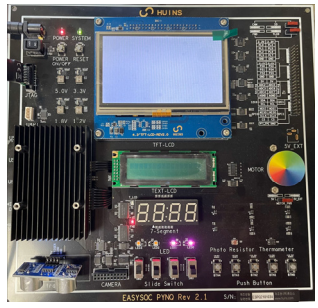

그림 5. 휴인스전자의 easySoC PYNQ 보드
Fig. 5. easySoC PYNQ board of Huins Inc.

일반적으로 FPGA 검증을 위하여 IP-Catalog에서 ILA 코어를 설계에 포함시키는 방법과, 합성된 후에 관찰하고 싶은 노드를 찾아서 찍어보는 방법이 있다. 그러나, 합성 후에는 신호들의 이름이 바뀌거나 버스 신호가 분리되기 때문에 두 번째 방법이 적절치가 않다. 따라서, 본 논문에서는 ILA 코어를 설계에 포함시키는 첫 번째 방법을 채택하였다.

그림 6에 수퍼스칼라 프로세서가 실행되는 ILA (Integrated Logic Analyzer) 결과 파형을 나타냈다. 에뮬레이션을 위하여 easySoC PYNQ에서 제공되는 100MHz 클럭을 이용하였다. 검증에 필요한 노드의 신호들로 프로그램 카운터, 32 비트 명령어 2 개, 메모리에서 읽은 32 비트 데이터, 메모리에 기록하는 32 비트 데이터, ALU에서

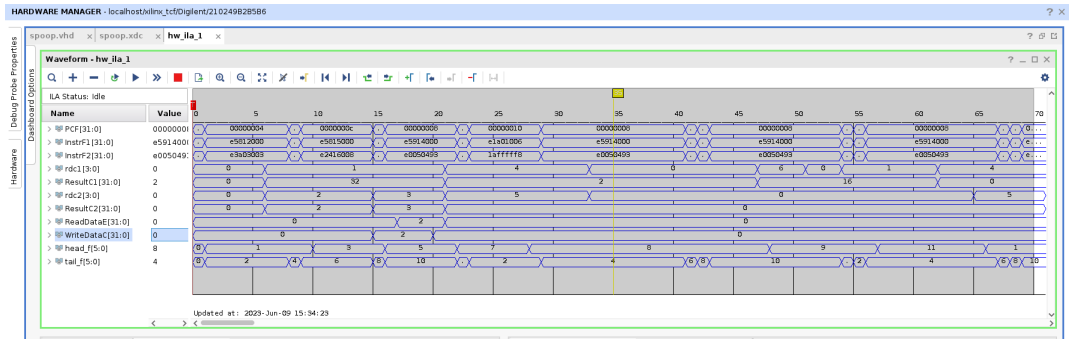

그림 6. 수퍼스칼라 프로세서의 Vivado Integrated Logic Analyzer의 파형
Fig. 6. Superscalar processor waveform of Vivado Integrated Logic Analyzer

계산한 32 비트 값 2 개, 재배열버퍼의 헤드포인터와 테일포인터를 선택하였다.

이 값들을 관찰한 결과, 본 연구에서 설계한 파이프라인 방식의 32 비트 파이프라인식 비순차실행 수퍼스칼라 프로세서가 프로그램을 올바르게 실행하는 것을 확인하였다.

V. 결 론

본 논문에서는 Vivado 환경에서 VHDL을 이용하여 19 개의 명령어를 실행할 수 있는 파이프라인식 수퍼스칼라 프로세서를 설계, FPGA로 구현 및 검증하였다. 주어진 ARM 프로그램을 실행시키고 로직 아날라이저로 측정한 결과, 올바르게 동작하는 것을 확인할 수 있었다.

추후로, FPGA로 검증된 본 프로세서를 Synopsys사의 Design Compiler를 비롯한 Front-End와 Back-End 도구를 이용하여, 국내의 반도체 교육기관인 IDEC을 통하여 ASIC 칩으로 구현할 예정이다.

References

[1] ARM Architecure Reference Manual, http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.subset.architecture.reference/index.html

[2] J. L. Hennessy, and D. A. Patterson, “Computer Architecture A Quantitative Approach”, 6th Edition; 2018. ISBN:978-012-811905-1

[3] S. L. Harris, and D. M. Harris, "Digital Design and Computer Architecture ARM Edition", Elsevier Korea LLC, 2016. DOI:https://doi.org/10.1016/C2018-O-14352-8

[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] R. M. Tomasulo, "An Efficient Algorithm for Exploiting Multiple Arithmetic Units", IBM Journal of Research and Development, Vol. 11, Issue. 1, Jan 1967, pp. 25-33. DOI:https://doi.org/10.1147/rd.111.0025

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, 멀티코어 프로세서, 인공지능 프로세서

※ 본 연구는 한성대학교 학술연구비 지원과제임.