The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 21, No. 1, pp.99-105, Feb. 28, 2021. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2021.21.1.99
JIIBC 2021-1-14

완전한 파이프라인 방식의 비순차실행 수퍼스칼라 프로세서의 VHDL 설계

VHDL Design for Out-of-Order Superscalar Processor of A Fully Pipelined Scheme

이종복*

Jongbok Lee*

요 약 오늘날 멀티코어 프로세서, 시스템 반도체, 그래픽처리장치를 막론하고 그것을 구성하는 기본 단위 또는 필수적으로 투입되는 CPU의 기본단위는 수퍼스칼라 프로세서이다. 따라서, 고성능의 비순차실행 수퍼스칼라 프로세서가 채택되어야만 위에서 거론된 시스템의 성능을 극대화할 수 있다. 수퍼스칼라 프로세서는 완전한 파이프라인 방식으로 재배열 버퍼와 예약스테이션을 이용하여 명령어를 동적 스케줄링 함으로써, 매 싸이클 당 복수 개의 명령어를 인출, 발행, 실행 및 기록한다. 본 논문에서는 예측실행 기능이 있는 완전한 파이프라인 방식의 비순차실행 수퍼스칼라 프로세서를 VHDL로 설계하고, GHDL로 검증하였다. 모의실험 결과, ARM 명령어로 구성된 프로그램에 대한 연산을 성공적으로 수행할 수 있었다.

Abstract Today, a superscalar processor is the basic unit or an essential component of a multi-core processor, SoCs, and GPUs. Hence, a high-performance out-of-order superscalar processor must be adopted for these systems to maximize its performance. The superscalar processor fetches, issues, executes, and writes back multiple instructions per cycle by utilizing reorder buffers and reservation stations to dynamically schedule instructions in a pipelined scheme. In this paper, a fully pipelined out-of-order superscalar processor with speculative execution is designed with VHDL and verified with GHDL. As a result of the simulation, the program composed of ARM instructions is successfully performed.

Key Words : fully pipelined, out-of-order, superscalar, VHDL

Ⅰ. 서 론

최근에 이르러 서버, 중형컴퓨터, 데스크탑 컴퓨터와 같은 전통적인 컴퓨터뿐만 아니라 노트북, 태블릿, 스마트폰과 같은 이동형 단말기 그리고 아두이노와 라즈베리파이와 같은 소형컴퓨터의 광범위한 사용으로 인하여, 마이크로 프로세서와 멀티코어 프로세서에 대한 수요가 증대하고 필요성이 강조되고 있다. 2019년까지 이 분야의 전통적인 강자는 Intel과 후발 주자인 AMD였으며, 임베디드 분야에서는 ARM이 석권하고 있었다. 그런데,

*정회원, 한성대학교 기계전자공학부
접수일자 2020년 10월 31일, 수정완료 2021년 1월 3일
게재확정일자 2021년 2월 5일

Received: 31 October, 2020 / Revised: 3 January, 2021 / Accepted: 5 February, 2021
*Corresponding Author: jblee@hansung.ac.kr
School of ME Engineering, Hansung University, Korea

2020년에 일부 분야에서 7 nm 공정을 앞세운 AMD의 Ryzen 칩이, 아직까지도 14 nm가 대부분인 Intel의 칩을 능가하기 시작했다. 또한, Apple에서 Mac에 탑재하는 고성능 시스템 반도체인 M1 칩을 자체 설계하여 내놓음으로써 프로세서를 설계 및 생산하는 해외 유수 업체들은 그 어느 때보다 더욱 뜨거운 경쟁을 벌이고 있다. Apple이 Mac용으로 설계한 M1 칩은, ARM을 기반으로 하는 멀티코어 CPU, GPU, DSP 및 뉴럴엔진 등을 단일 칩에 집대성한 시스템 반도체 (SoC) 형태로서, 5 nm 공정에서 160억 개의 트랜지스터로 만들어졌다. 이러한 기술에 힘입어서, M1 칩의 성능은 지금까지 업계의 터줏대감이었던 Intel과 AMD를 능가하여 큰 파장을 불러 일으키고 있다. 또한 전통적으로 그래픽 처리장치를 생산하던 NVIDIA는 최근에 SoftBank로부터 ARM을 인수하여 경쟁사를 긴장하게 만들고 있다. NVIDIA는 CUDA 코어로 구성되는 GTX 계열과, 여기에 Tensor 코어, Ray Tracing 코어를 추가한 RTX 시리즈를 생산하여, 고해상도 그래픽 및 인공지능 연산이 가능한 그래픽 처리장치의 생산과 판매에 박차를 가하고 있다. 이와 같이 멀티코어 프로세서, 시스템반도체, 범용 그래픽 처리장치를 통털어서 수퍼스칼라 프로세서가 기본 구성요소 또는 필수요소이다. 따라서, 국내에서 수퍼스칼라 프로세서에 대한 연구 및 설계기술의 확보가 매우 중요하다.

1966년에 출시된 Seymour Cray의 CDC 6600은 최초의 수퍼스칼라의 설계로 기록되었다. 1967년에 발표된 IBM System/360 Model 91은 또 다른 수퍼스칼라로 구축된 메인 프레임이었다. Motorola MC88100, Intel i960CA 및 AMD 29050 마이크로 프로세서는 최초의 상용 단일 칩 수퍼스칼라 마이크로 프로세서였다. RISC 아키텍처는 CISC 아키텍처에 비하여 적은 개수의 트랜지스터 및 작은 다이 영역을 필요로 하기 때문에, 다중 연산장치를 필요로 하는 수퍼스칼라 실행에 적합하였다. 이러한 이유로, 저전력 애플리케이션, 임베디드 시스템 및 배터리 구동 장치에 사용되는 일부 CPU를 제외하고는, 본질적으로 1998년 이후 개발 된 모든 범용 CPU는 수퍼 스칼라이다.

Intel에서 개발한 P5 Pentium은 최초의 x86 계열의 수퍼스칼라 프로세서였으며, 뒤를 이은 Nx586, P6 Pentium Pro 및 AMD K5는 수퍼스칼라 마이크로 아키텍처에서 실제 실행되기 전에 x86 명령어를 동적 마이크로 코드로 디코딩하는 기술을 이용하여, 더욱 높은 병렬처리를 가능하게 했다.

수퍼스칼라 프로세서는 복수의 명령어를 인출 및 발행하고 다중 연산장치를 통하여 한 싸이클에 복수의 명령어를 처리할 수 있다. 초기 수퍼 스칼라 CPU에는 2개의 ALU와 1 개의 FPU로 구성하는데 그쳤지만, MIPS 또는 PowerPC 같은 최신 프로세서에는 4 개의 ALU, 2 개의 FPU 및 2 개의 SIMD 장치가 포함되도록 하드웨어가 크게 확장되었다[1].

국내의 프로세서 반도체 설계 기술은, 세계 최고 수준의 메모리 반도체 설계 및 공정 기술에 비하여 매우 미흡한 형편이다. 원래 반도체 칩은 메모리 외에 CPU, DSP, GPU, SoC 등의 다양한 종류로 구성된다. 그러나, 국내에서는 메모리 반도체에 크게 편중되어 있기 때문에, 그것을 제외한 나머지를 비메모리 반도체라고 부르고 있다. 지금까지 국내의 대학, 연구소 및 기업체에서 단순한 RISC 프로세서를 설계하고 반도체칩으로 구현한 경우는 극히 일부 발견된다. KAIST는 2008년부터 Core-A 프로세서를 야심차게 개발했으나, 비순차실행방식이 아닌 순차실행방식에 그쳤고, 현재로서는 명맥을 잇지 못하고 있다. 삼성전자는 2010년부터 ARM Cortex-A8을 라이센싱하여 Exynos 칩을 직접 생산하였고, 이것과는 별도로 자체의 CPU 코어 개발을 추진하였으나, 그만 2019년에 자체 CPU 코어개발을 중단하기로 하였다. 그 외의 국내의 대학 및 연구기관에서 진행된 비순차실행 수퍼스칼라 프로세서에 대한 연구는 해외에서 개발된 Simplescalar[2]와 같은 모의실험기를 실행하여 오직 그 성능을 측정하는 연구에 그쳤으며, 실제로 프로세서 하드웨어를 설계한 논문은 찾아볼 수가 없다. 최근에 국내에서도 RISC-V를 기반으로 맞춤형 IP 반도체를 생산하는 업체가 생겨나기 시작했으나, 여전히 RISC-V는 국내에서가 아닌, 해외에서 설계한 오픈 하드웨어 CPU이다[3].

위에서 살펴본 바와 같이, 높은 난이도와 복잡성 때문에, 국내에서 완전한 파이프라인 방식의 비순차실행 수퍼스칼라 프로세서 하드웨어를 설계 및 구현한 사례는 거의 찾아볼 수가 없다. 따라서, 다소 늦었더라도 이에 대한 연구를 통하여 설계 노우 하우를 획득하여, 국내의 프로세서 설계 기술을 고도화시키고 메모리 반도체 설계기술과 균형을 이룰 필요가 있다. 본 논문에서는 VHDL을 이용하여 파이프라인 방식을 지원하는 비순차실행 수퍼스칼라 프로세서를 설계했으며, 명령어집합으로 임베디드 마이크로 프로세서로 가장 널리 쓰이는 ARM을 이용하였다[4]. 본 설계의 코딩 및 검증을 위하여 리눅스에서 실행되는 GHDL과 GTKWave를 활용하였다.

본 논문은 다음과 같이 구성된다. 2 장에서 파이프라인 방식의를 이용한 비순차실행 수퍼스칼라 프로세서의

원리에 대하여 살펴보고, 3 장은 본 논문에서 설계한 비순차실행 수퍼스칼라 프로세서의 구조를 VHDL의 설계 관점에서 기술한다. 4 장에서 모의실험 환경과 모의실험 결과를 보이고, 5 장에서 결론을 맺는다.

II. 파이프라인 방식을 이용하는 비순차실행 수퍼스칼라 프로세서의 동작 원리

본 논문에서는 한 싸이클에 2 개의 명령어를 처리할 수 있는 수퍼스칼라의 설계를 목표로 하였다. 수퍼스칼라의 비순차실행이 가능한 동적 스케줄링 기능을 지원하기 위하여, 예약스테이션 (Reservation Station)과 재정렬버퍼(Reorder Buffer)를 이용하였다. 예약스테이션은 연산장치와 결합된 장치로서, 연산을 통하여 피연산자가 준비된 즉시 그 값을 필요로 하는 다른 예약스테이션에 송부한다. 아울러, 아직 피연산자가 준비되지 않은 명령어는 그 값을 공급할 예약스테이션을 가리키도록 함으로써, 예약스테이션을 통하여 동적 스케줄링에 필요한 레지스터 재명명 기능을 수행한다. 이것을 통하여, 수퍼스칼라 프로세서의 해저드(hazard) 감지와 실행제어가 분산되는 장점이 있다.

비순차실행 수퍼스칼라의 운영을 위하여 Tomasulo 알고리즘이 적용되었다[5]. Tomasulo 알고리즘에서 명령어들의 비순차실행 (Out-of-Order Execution)을 허용하면서도 순차완료 (In-Order Completion) 되도록 하려면 재정렬버퍼를 필요로 한다. 재정렬버퍼는 실행이 종료했으나, 아직 완료하지 않은 명령어들을 프로그램의 원순서대로 유지하는 기능을 한다. 재정렬버퍼는 실행을 마쳤으나 아직 레지스터화일에 완료를 하지 않은 연산결과 중에서 이 값을 필요로하는 예약스테이션에 공급하는 역할도 수행한다.

그림 1에 재정렬버퍼와 예약스테이션을 포함하는 비순차실행 수퍼스칼라 프로세서의 블럭도를 나타냈다. 본 프로세서는 명령어 메모리, 명령어 큐, 데이터 메모리, 레지스터 화일, 예약스테이션, 재배열 버퍼, 로드와 스토어 장치 각 1 개, 정수형 연산장치 3 개, 정수형 곱셈기 1개, 분기장치 1 개로 구성된다. 본 논문에서는 정수형 명령어만을 목표로 설계하였으며, 부동소수점 명령어는 추후에 추가할 예정이다.

각 예약스테이션에는 발행된 명령어들이 삽입되고, 그 명령어들은 연산유닛에서 실행되기를 기다린다. 만일에 어느 명령어의 피연산자의 연산 결과가 준비된 경우에는

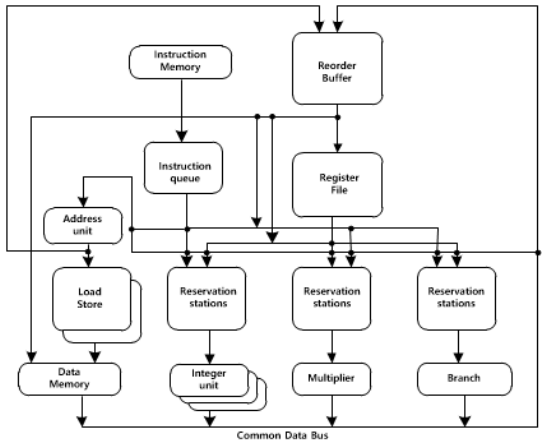

그림 1. 완전한 파이프라인 방식의 비순차실행 수퍼스칼라 프로세서의 블럭도
Fig. 1. The fully pipelined out-of-order superscalar processor

그 결과 역시 예약스테이션의 해당 항목에 삽입된다. 그러나 피연산자가 아직 준비가 되지 않았다면, 그 값을 공급하는 예약스테이션을 가리키도록 한다. 다음은 토마술로 알고리즘의 주요 6 단계를 기술한다.

1. 인출 (fetch)과 발행 (issue) 단계

FIFO 구조로 구성된 명령어큐의 첫 번째와 두 번째 항목에서 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 해저드가 발생한 것으로서, 재정렬버퍼와 예약스테이션이 모두 준비될 때까지 인출과 발행을 멈춘다 (stall).

먼저 동시에 인출된 2 개의 명령어 사이의 데이터 종속 여부를 판단한다. 만일 명령어 2의 원천 레지스터 중의 하나가 명령어 1의 목적 레지스터에 종속이라면, 명령어 2의 피연산자는 명령어 1의 결과를 가리키도록 태그를 부여한다. 반면에 위 명령어 쌍이 서로 종속이 아닌 경우, 인출된 2 개의 명령어는 예약스테이션을 통하여 연산을 실행하는 정상적인 처리단계에 들어간다. 각 명령어가 예약스테이션에서 연산을 하기 위하여 피연산자가 필요하며, 피연산자를 읽기 위하여 다음 세 가지의 경우에 따라서 처리한다.

첫 번째는 예약스테이션이 피연산자를 레지스터로부터 정상적으로 읽어오는 경우이고, 두 번째는 예약스테이

션이 재정렬버퍼에 있는 피연산자를 읽어오는 경우이다. 세 번째는 피연산자가 레지스터는 물론 재정렬버퍼에도 그 값이 존재하지 않는 경우로서, 이때는 그 값을 생산할 명령어의 재정렬버퍼 상의 위치인 태그를 읽는다. 예약스테이션은 추후에 해당 피연산자가 연산이 실행되고 재정렬버퍼에 등록되면 태그를 이용하여 그 값을 가져올 수 있다. 예약스테이션을 이용하는 이 과정에서, 재명명으로 인하여 명령어 간의 반종속 (anti-dependency)과 출력종속 (output-dependency)이 해결된다.

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

발행단계에서 살펴본 바와 같이, 만일 피연산자가 준비되어있지 않으면, 공통데이터버스를 통하여 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 싸이클당 2 개의 연산을 실행할 수 있으므로 수퍼스칼라 비순차실행이 가능하다. 이와 같이 피연산자가 준비될 때까지 명령어의 연산을 유예시킴으로서, 명령어 간의 실제종속 (true -dependency)를 해결할 수 있다.

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

3. 쓰기 (write back) 및 완료 (commit) 단계

각 연산장치에서 명령어의 실행이 완료되어 결과가 준비되었을 때, 그 결과를 기다리는 재정렬버퍼와 예약스테이션에 공통데이터버스를 통하여 각각 전송한다. 스토어의 경우 메모리에 저장하는 값을 재정렬버퍼의 값 영역에 기록한다. 만일에 저장해야할 값이 아직 준비가 되지 않았다면, 공통데이터버스에 그 값이 준비가 될 때까지 대기한다.

완료는 명령어를 마무리하는 마지막 단계로서, 마무리하는 명령어가 분기어, 스토어, 로드를 포함한 일반 명령어 중에 어느 것 인가에 따라서 처리가 다르다. 일반 명령어가 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 도달하는 경우, 이것은 예측이 틀렸다는 것을 의미한다. 이 때는 재정렬버퍼를 일소 (squash) 하고 미리 저장해 두었던 분기어의 올바른 후속 명령어부터 실행을 재개한다. 반면에, 만일 분기어가 올바르게 예측되었다면 정상적으로 완료된다[6-8].

본 연구에서 설계한 수퍼스칼라 프로세서에서 비순차실행을 위한 핵심 장치인 재배열버퍼와 예약스테이션은 매 싸이클마다 각 단계에서 데이터를 저장했다가 다음 단계로 데이터를 전달하여 완전한 파이프라인 동작을 할 수 있도록 설계되었다.

III. 파이프라인 방식의 비순차실행 수퍼스칼라 프로세서의 VHDL 설계

본 연구에서 파이프라인 방식을 이용하는 비순차실행 수퍼스칼라 프로세서를 설계하기 위하여 VHDL을 이용하였다. 특히, 재정렬버퍼와 예약스테이션을 VHDL의 레코드형 배열 자료구조로 정의하면 높은 추상도를 가짐과 동시에 설계에 편리하다. 또한, 재정렬버퍼와 예약스테이션을 레코드형 배열로 선언한 후에, 조건과 부합하는 레코드형을 찾을 때는 VHDL에서 지원하는 순환 반복문인 for 루프를 이용함으로써 비순차실행 수퍼스칼라 프로세서를 간편하고 효율적으로 설계할 수 있다.

VHDL에서 레코드를 자료구조로 갖는 신호를 모든 모듈에서 이용하기 위하여, 레코드 자료구조를 패키지문에 선언하고, 모든 VHDL 모듈에서 use문을 이용하여 참조가 가능하게 할 수 있다. 그림 2는 본 연구에서 설계에 이용한 VHDL의 레코드 자료구조를 패키지문으로 선언한 것이다. 예약스테이션의 단일 레코드형을 TyReserv, 재정렬버퍼의 단일 레코드형을 TyROB로 선언한 다음에, 다시 TyArrayReserv와 TyArrayROB를 각각 32 개와 10 개의 레코드형 배열로 선언하였다.

예약스테이션 레코드의 주요 구성은 Busy, Opcode, V_1, V_2, Q_1, Q_2, Tag, Addrs, Value 등으로 구성된다. Busy는 예약스테이션의 항목이 현재 비어있는지의 여부를 나타내는데 쓰이고, Opcode는 해당 명령어의 유형을 기록한다. V_1, V_2는명령어의 준비된 제1피연산자와 제2피연산자를 각각 나타내며, 이 때 Q_1, Q_2의 값은 모두 0

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
package poop_pkg is
type TyReserv is record
       Busy : std_logic;
       Opcode : std_logic_vector(1 downto 0);
       V1 : std_logic_vector(31 downto 0);
       V2 : std_logic_vector(31 downto 0);
       Q1 : natural;
       Q2 : natural;
       Tag : natural;
       Addrs : std_logic_vector(31 downto 0);
       Value : std_logic_vector(31 downto 0);
       Executed : std_logic;
end record TyReserv;
type TyArrayReserv is array (9 downto 0) of TyReserv;
type TyROB is record
       Busy : std_logic;
       Opcode : std_logic_vector(1 downto 0);
       Ready : std_logic;
       State : std_logic_vector(2 downto 0);
       Dest : std_logic_vector(3 downto 0);
       Value : std_logic_vector(31 downto 0);
       Addrs : std_logic_vector(31 downto 0);
       Mispredict : std_logic;
       Commit : std_logic;
       Squash : std_logic;
       Speculative : std_logic;
       Link : natural;
end record TyROB;
type TyArrayROB is array (15 downto 0) of TyROB;
```

그림 2. 예약스테이션과 재정렬버퍼의 VHDL 레코드형 자료구조
Fig. 2. The VHDL record data structures of Reservation Stations and ROBs

이다. 만일 피연산자들이 아직 준비되지 않았을 경우에는, Q_1과 Q_2는 각각 첫 번째와 두 번째 피연산자를 공급해줄 명령어의 재정렬버퍼 항목번호를 가리키게 된다. Tag는 현 명령어의 목적 레지스터 값을 계산하여 공급하는 명령어의 재정렬버퍼 상의 항목번호이고, Addrs는 로드와 스토어의 유효주고 계산에 쓰이며, Values는 명령어의 연산결과를 수록한다. 재정렬버퍼의 레코드는 Busy, Opcode, Ready, Dest, Values, Addrs 등으로 구성된다. Ready는 명령어가 완료되었는지의 여부를 나타내며, Dest는 목적 레지스터의 번호를 기록한다. 그 외의 재정렬버퍼의 레코드 항목은 예약스테이션과 같다.

각 VHDL 모듈에서 엔티티문의 포트로 레코드형 배열을 입력 또는 출력으로 선언하여, 레코드형 배열 단위로 서로 다른 모듈 간에 효율적으로 정보를 주고받을 수 있다. 이 때, 완전한 파이프라인 방식의 운영을 위하여 매 싸이클마다 서로 다른 VHDL 모듈에서 공통의 재정렬버퍼나 예약스테이션에 쓰기작업이 필요하다. 그러나, 하드웨어 기술언어에서 이것은 다중 원천 신호의 구동에 해당되어 금지된다. 따라서, 이 문제를 해결하기 위하여 본 논문에서는 예약스테이션과 재배열버퍼의 자료구조를 파이프라인의 각 6 단계로 구성하고, 이전 단계의 데이터를 다음 단계의 데이터로 매 싸이클마다 전달하도록 하였다. 이때, 마지막 완료단계는 최초의 인출단계로 데이터가 다시 되먹임되도록 하여, 위 자료구조 데이터의 항상성을 유지하였다.

IV. 모의실험

본 논문의 모의실험은 운영체제 Fedora 33에서 3.1 GHz로 동작하는 Intel Core i5-2400 데스크탑 컴퓨터에서 시행하였다. VHDL 2008 버전을 컴파일하기 위한 도구로 GHDL 0.37 버전을, 모의실험 파형을 관찰하는 도구로 GTKWave 3.3.101 버전을 이용하였다. 총 25개의 독립적인 VHDL 프로그램 모듈을 코딩하여 비순차실행 수퍼스칼라 프로세서를 설계하는데 이용하였다.

그림 3은 모의실행에서 비순차실행 수퍼스칼라 프로세서의 입력으로 이용한 ARM 어셈블리 프로그램과 기계어를 나란히 표현한 것이다. 이 때, 로드와 스토어, 정수형 덧셈, 뺄셈, 곱셈, 분기 명령어는 각각 1 싸이클이 소요된다.

그림 4에 완전한 파이프라인 방식의 비순차실행 수퍼스칼라 프로세서를 GHDL로 컴파일 후에 모의실행되는 결과를 GTKWave의 결과 파형으로 나타냈다. 설계한 수퍼스칼라 프로세서가 명령어 간의 종속이 없을 때는, 매 싸이클 마다 명령어 2 개를 인출하여 재정렬버퍼와 예약스테이션에 삽입하며, 실행 완료시에 명령어 2 개의 결과를 레지스터 화일에 쓰는 것을 확인할 수 있다. 순차실행 프로세서인 경우에는, 세 번째 명령어인 STR 연산이 완료되지 않았을 때, 그 이하의 명령어들은 발행 및 실행을 할 수가 없다. 그러나, 파이프라인 방식을 이용하는 비순차실행 수퍼스칼라 프로세서의 기능으로 인하여, 네 번째 명령어인 SUB의 발행 및 실행이 되었다. 그리고, 분기 여부가 결정되기 전에 BNEQ 분기어를 예측실행하였다. 모의실험을 수행하고 수퍼스칼라 프로세서의 레지스터 화일과 데이터메모리에 기록된 값을 확인하니

```
MAIN : LDR R2, [R1, #0]   E5912000
       MUL R4, R2, R3     E0040392
       STR R4, [R0, #0]   E5804000
       SUB R1, R1, #8     E2411008
       SUBS R5, R1, R3    E0515003
       BNEQ MAIN          1AFFFFF0
       ADD R6,R2,R3       E0826003
```

그림 3. 모의실행의 입력에 이용된 ARM 프로그램과 기계어
Fig. 3. The input ARM program and machine code used for simulation

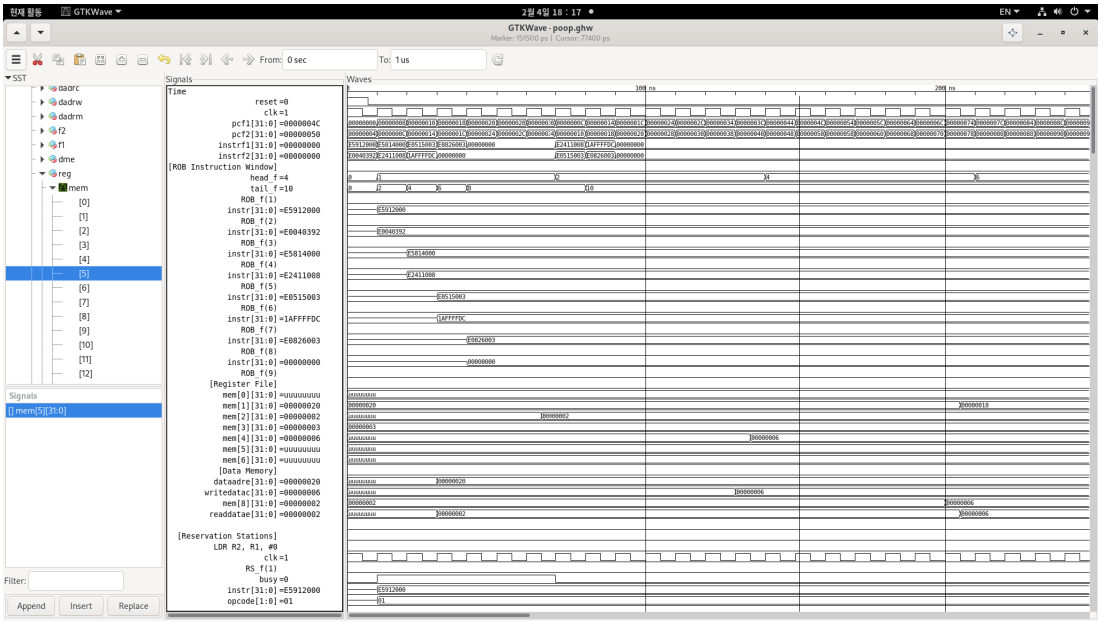

그림 4. 완전한 파이프라인 방식의 비순차실행 수퍼스칼라 프로세서의 모의실행 결과
Fig. 4. The simulation results of the fully pipelined out-of-order superscalar processor

올바른 연산결과가 기록되었다. 따라서, 완전한 파이프라인 방식의 비순차실행 수퍼스칼라 프로세서가 올바르게 실행하는 것을 확인할 수 있었다.

V. 결 론

본 논문에서는 VHDL을 이용하여, ARM 명령어를 기반으로 하는 완전한 파이프라인 방식의 비순차실행 수퍼스칼라 프로세서를 설계하였다. 특히, 레코드형 자료구조를 이용하여 재정렬버퍼와 예약스테이션을 배열로 설계함으로써 프로그램 언어상으로 높은 수준의 추상도, 편리함 및 간결성을 확보하였다. 주어진 ARM 어셈블리 프로그램을 입력으로 모의실행 시킨 결과, 완전한 파이프라인 방식에 따라 비순차실행 수퍼스칼라 프로세서가 올바르게 동작하는 것을 확인할 수 있었다.

추후의 연구로, 한 싸이클에 명령어 2 개를 처리하는 수퍼스칼라 프로세서를 명령어 4 개를 처리할 수 있도록 구조를 개선시키는 것이다. 또한, 수퍼스칼라 프로세서에 부동소수점 명령어와, 높은 정확도를 갖는 분기예측기 하드웨어를 추가할 수 있다. 나아가 설계된 수퍼스칼라 프로세서를 Xilinx사의 Vivado 또는 Altera사의 Quartus II를 이용하여 합성한 후에, 정적 시간 분석(STA)과 합성 후 모의실험 (Post synthesis simulation)을 거쳐 최종적인 동작을 검증하고 FPGA로 구현할 예정이다.

References

[1] K. C. Yeager, "The Mips R10000 superscalar microprocessor", IEEE Micro, Vol. 16, No. 2, pp. 28-41, Apr 1996. DOI:https://doi.org/ 10.1109/40.491460.

[2] T. Austin, E. Larson, and D. Ernest, "SimpleScalar : An Infrastructure for Computer System Modeling", Computer, Vol. 35, No. 2, pp. 59-67, Feb 2002. DOI:https://doi.org/10.1109/2.982917.

[3] "Design of the RISC-V Instruction Set Architecture," Andrew Waterman, "University of California, Berkeley Techincal Report No. UCB/EECS-2016-1", Jan 2016. http://www2.eecs.berkely.edu/ Pubs/TechRpts/2016/EECS-2016-1.pdf

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

[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

[6] J. L. Hennessy, and D. A. Patterson, "Computer Architecture A Quantitative Approach", 2018.

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

[8] 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.

저 자 소 개

이 종 복(정회원)

- 1964년 8월 20일생.
- 1988년 서울대 컴퓨터공학과 졸업.
- 1998년 동 대학 전기공학부 졸업 (공박).
- 1998~2000 LG반도체선임연구원.
- 2000년~현재 한성대 기계전자공학부 교수

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

※ 본 연구는 한성대학교 교내학술연구비 지원과제임.