The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 19, No. 6, pp.183-189, Dec. 31, 2019. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2019.19.6.183
JIIBC 2019-6-26

인터럽트 기능을 갖는 ARM 프로세서의 설계 및 모의실행

Design and Simulation of ARM Processor with Interrupts

이종복*

Jongbok Lee*

요 약 ARM은 저가임에도 불구하고, 저전력 소비와 신뢰할만한 성능으로 인하여 스마트폰, 디지털 카메라, 가정용 네트워크 장치, 무선 기술 등에 널리 쓰이고 있다. 국내는 메모리 반도체 설계에 있어서 세계 최고의 수준이나, 프로세서의 설계는 그에 미치지 못하여 메모리와 프로세서의 균형있는 발전을 이루지 못하고 있다. 일반적으로 프로세서를 설계할 때는 반드시 예외처리 및 인터럽트 기능까지 갖춰야하지만 연구단계에서는 이것이 누락되는 경우가 많다. 그러나, 프로세서가 완벽하게 동작하기 위하여 예외처리 및 인터럽트 기능까지 포함되어야 한다. 본 논문에서는 VHDL을 이용하여 예외처리 및 인터럽트 기능을 갖는 32 비트 ARMv4 계열의 프로세서를 설계하고, ModelSim으로 검증하였다. 그 결과, ARM의 예외처리 및 인터럽트 기능을 성공적으로 수행할 수 있었다.

Abstract Despite its low cost, ARM is widely used in smartphones, digital cameras, home network devices, and wireless technologies because of its low power consumption and reliable performance. The domestic memory semiconductor design technology is in the world's highest level, but that of the processor is far less than that, which results in the technology unbalance between the memory and the processor. When designing a processor, exception and interrupt capabilities are requisite, but this is often omitted in the research stage. However, exception processing and interrupts must be included in order for the processor to function fully. In this paper, we design a 32-bit ARMv4 family of processors with exception handling and interrupts using VHDL and verify with ModelSim. As a result, ARM's exception and interrupts are successfully performed.

Key Words : ARM, Exception, Interrupt, VHDL, ModelSim

Ⅰ. 서 론

1980년대에 Acorn Computer Group에서 최초로 개발된 ARM 아키텍처는 Advanced RISC Machine으로 명칭을 바꿔서 오늘에 이르게 되었다. ARM 프로세서는 대부분의 스마트폰, 태블릿, 게임기, 카메라, 로봇, 자동차에 탑재되고 있으며, 현재 100억 개가 매년 판매되고 있다. ARM은 삼성, Altera, 애플, 퀄컴 등의 회사들로 하여금 라이선스를 통하여 시스템 온 칩의 형태로 공급하며, 직접 프로세서를 판매하지 않는 특이한 비즈니스

*정회원, 한성대학교 전자정보공학과
접수일자 : 2019년 11월 1일, 수정완료 : 2019년 12월 1일
게재확정일자 : 2019년 12월 6일

Received: 1 November, 2019 / Revised: 1 December, 2019 / Accepted: 6 December, 2019
*Corresponding Author: jblee@hansung.ac.kr
Dept. of EI Engineering, Hansung University, Korea

모델을 채택하고 있다.

ARM 프로세서는 thumb 및 쉬프트 기능을 강화한 독특한 명령어 특징뿐만이 아니라, 낮은 전력을 사용하도록 설계하여 모바일 및 임베디드 시장에서 큰 강세를 보이고 있다. 오늘날, 데스크탑 컴퓨터나 서버 컴퓨터를 제외한 대부분의 스마트폰 및 임베디드 기기에는 전부 ARM 프로세서가 탑재되어 있다고 해도 과언이 아니다.

국내의 프로세서 설계 분야는, 세계 최고 수준의 메모리 반도체 설계 및 공정 기술에 비하여 매우 미흡한 형편이다. 따라서 프로세서 설계 분야에 대한 연구를 활발히 하고 박차를 가하여 균형을 이룰 필요가 있다. 일반적으로 프로세서 아키텍처를 설계할 때는 반드시 예외처리(exception) 및 인터럽트(interrupt) 기능이 포함되어야 하지만, 가장 어렵고 까다로운 설계 부분이기 때문에 연구단계에서는 누락되는 경우가 많다. 이하 본 논문에서는 인터럽트라는 용어로 통일하고, VHDL을 이용하여 인터럽트 기능을 포함하는 5 단계 파이프라인을 갖는 ARM 프로세서를 설계하였으며, 코딩 및 검증을 위하여 ModelSim을 활용하였다. 본 연구에서 채택한 ARM의 아키텍처 계열은 ARMv4로서, 후속 버전인 ARM8 및 Strong ARM에서 채택한 구조이다.

본 논문은 다음과 같이 구성된다. 2 장에서 ARM 프로세서의 인터럽트를 살펴보고, 3 장에서 ARM 프로세서의 명령어 및 파이프라인 단계별 동작과 본 논문에서 설계한 ARM 프로세서의 구조를 기술한다, 4 장에서 모의실험 환경과 모의실험 결과를 보이고, 5 장에서 결론을 맺는다.

II. ARM 프로세서의 인터럽트 기능

인터럽트를 적절히 다루는 것은 마이크로 프로세서의 꽃이라고 할 수 있다. 인터럽트를 효율적으로 사용하여 마이크로프로세서와 외부시스템과의 상호작용을 관리함으로써 시스템의 효율과 처리자원의 사용을 극적으로 개선시킬 수 있다. 인터럽트는 인터럽트 처리 루틴으로의 분기 및 인터럽트 처리를 완료한 후에 다시 인터럽트로부터 정상 프로그램으로의 복귀로 나눌 수 있다. 이 때, 인터럽트 벡터 어드레스로의 1 차 분기는 프로세서 회로 설계로 구현하고, 인터럽트 처리 루틴으로의 2 차 분기 및 인터럽트로부터의 복귀는 프로그램 명령어, 즉 소프트웨어 방식을 이용한다. 즉, 인터럽트 처리는 하드웨어와 소프트웨어의 조화로운 협업에 의하여 구현되는 아름다운 기술이다.

ARM에서 인터럽트는 프로세스 상태 레지스터인 CPSR (Current Process Status Register)의 비트를 설정함으로써 활성화 또는 비활성화된다. 이 때, CPSR의 제 6 비트가 FIQ(Fast Interrupt Request), 제 7 비트가 IRQ(Interupt Request)에 해당한다. CPSR은 또한 시스템 모드나 사용자 모드 등의 프로세서 모드를 제어하는 기능을 수행하며, CPSR의 상위 4 비트는 NZCV의 조건부 실행 플래그를 나타낸다.

인터럽트가 CPSR에서 활성화되어 인터럽트가 발생하면, ARM 프로세서는 인터럽트를 처리하기 전에 파이프라인에 이미 삽입된 명령어들은 실행을 중단하지 않고 계속한다. 또한 프로세서는 CPSR의 값을 SPSR에 저장하고, 현재 PC의 값을 LR 레지스터에 저장하여 복귀할 수 있도록 하며, PC의 값에 각 인터럽트별 벡터 어드레스를 적재하여 분기한다.

표 1. ARM의 인터럽트 유형, 순위 및 벡터어드레스
Table 1. The type, priority and vector address of ARM processor interrupts

인터럽트 유형	우선순위	벡터어드레스
Reset	1	0x00000000
Data Abort	2	0x00000010
FIQ	3	0x0000001C
IRQ	4	0x00000018
Prefetch Abort	5	0x0000000C
SWI	6	0x00000008
Undefined	6	0x00000004

표 1에 ARM 마이크로 프로세서의 7 가지 인터럽트 유형과 우선순위 및 벡터 어드레스를 나타냈다. 인터럽트 사건은 동시에 발생할 수 있으므로, 우선 순위가 설정되어야 한다. 리셋이 가장 우선순위가 높은데, 그 이유는 ARM 마이크로 프로세서가 켜졌을 때 즉시 발생하기 때문이다. 리셋 다음으로는 데이터 중단 (Data Abort) 인터럽트가 두 번째로 높은 우선순위를 갖는다.

다음은 각 인터럽트 유형에 대한 설명이다. 리셋(Reset)은 프로세서에 전원이 켜질 때 발생한다. 리셋이 걸리면 CPSR이 SVC 모드에 진입하고 IRQ와 FIQ가 모두 1로 설정되어 더 이상의 인터럽트 발생을 금지한다. 리셋 처리기는 스택을 포함한 메모리와 캐쉬를 설정하고 초기화하며, 소프트웨어 인터럽트, 정의되지 않은 명령어 및 메모리 접근을 회피하여 중간에 인터럽트나 예외사항이 발생하지 않도록 주의해야한다.

데이터 중단(Data Abort)은 유효하지 않은 메모리 어드레스에 대한 접근이 발생했다는 것을 메모리제어기 또는 메모리관리장치(MMU)가 프로세서에게 알릴 때 발생한다. 이 때 유효하지 않은 메모리 어드레스에 대한 접근은 해당 어드레스에 대한 메모리가 아예 존재하지 않거나, 프로세서가 해당 메모리 지역에 대한 접근권한이 없을 때를 의미한다.

고속 인터럽트(FIQ)는 외부장치가 FIQ 핀을 구동했을 때 발생한다. FIQ 인터럽트가 시작되면 IRQ와 추가의 FIQ 인터럽트를 금지시킨다. 인터럽트를 효율적으로 처리하기 위하여 FIQ 처리기를 주의하여 설계해야 한다. 일반 인터럽트 (IRQ) 역시 외부장치가 IRQ 핀을 구동했을 때 발생한다. IRQ는 데이터 중단이나 고속인터럽트보다 우선순위가 낮다.

선인출 중단 (Prefetch Abort)은 명령어를 적재하는 과정에서 메모리 결함이 일어날 때 발생한다. 이 인터럽트는 명령어가 파이프라인의 실행단계에 진입하고 이것보다 우선순위가 더 높은 인터럽트가 제기되지 않았을 때만 발생한다. 소프트웨어 인터럽트(SWI)는 ARM의 SWI 명령어가 성공적으로 인출 및 해독되고 여타의 우선순위가 높은 인터럽트가 걸리지 않았을 때 발생하며, CPSR이 SVC 모드로 진입한다. 마지막으로, 정의되지 않은 명령어 (Undefined) 사건은, ARM의 명령어집합에 속하지 않은 명령어가 인출 및 해독되고 여타의 인터럽트가 걸리지 않았을 때 발생한다.

ARM 프로세서가 각 인터럽트에 대한 벡터 어드레스로 분기했을 때, 메모리의 벡터 어드레스들의 간격이 협소하므로 각 위치에 인터럽트 처리 루틴까지는 모두 수록할 수가 없다. 따라서, 각 벡터 어드레스에는 LDR이나 B 명령어를 이용하여 해당 인터럽트 서비스 루틴으로 다시 한 번 분기하도록 한다. 따라서, 인터럽트가 걸렸을 때 처음에 회로 설계에 의하여 벡터 어드레스로 1차 분기하고, 다시 프로그램 명령어를 이용한 인터럽트 서비스 루틴으로 2 차 분기하여 도합 2 회의 분기를 하게 된다.

인터럽트 처리를 완료한 뒤에 인터럽트로부터 복귀할 때, 먼저 SPSR을 다시 CPSR로 적재하여 프로세어의 원상태를 복구한다. 이 때, 연결 레지스터(LR)에 저장된 복귀주소는 오프셋을 포함하며 인터럽트의 종류에 따라서 다르므로 그 값만큼 빼야 한다. 표 2에 각 인터럽트 사건에 대한 오프셋을 나타냈다.

이와같이 LR 레지스터에 저장된 주소의 값에서 오프셋을 뺀 값을 PC에 적재하여 복귀함으로써 인터럽트가 걸리기 전에 실행이 중단되었던 프로그램 명령어를 정상 수행한다.

표 2. ARM의 인터럽트 복귀주 오프셋 및 복귀명령어
Table 2. The return address offset and return instruction from the interrupts of ARM processor

인터럽트 유형	오프셋	복귀명령어
Reset	없음	없음
Data Abort	-8	SUB LR,#8 MOVS PC, LR
FIQ	-4	SUB LR,#4 MOVS PC, LR
IRQ	-4	SUB LR,#4 MOVS PC, LR
Prefetch Abort	-4	SUB LR,#4 MOVS PC, LR
SWI	0	MOVS PC, LR
Undefined	0	MOVS PC, LR

III. ARM 프로세서의 구조

표 3에 본 논문에서 설계한 ARM 명령어 집합을 나타냈다. ARM 명령어는 크게 산술 논리, 곱셈, 메모리, 분기의 4 가지 유형으로 구성된다[1]. 산술 논리 명령어는 AND, EOR, SUB 등의 21 개 명령어로 구성된다. 곱셈 명령어는 6 개의 명령어로 이루어지며, 결과가 32 비트인 것과 64 비트인 것으로 나뉜다. 메모리 명령어는 STR, LDR 등의 스토어 및 로드 명령어로 구성되며 워드 단위 이외에 바이트 단위 또는 하프워드 단위로 수행한다. 마지막으로, 분기 명령어는 B와 BL로 구성된다.

그림 1은 ARM 프로세서의 5 단계 파이프라인이다. 이것은 인출(Fetch), 해독(Decode), 메모리 접근(Memory), 실행(Execute), 되쓰기(Write Back)의 5 단계로 나뉜다. 각 단계의 사이마다 파이프라인의 모든 신호를 래치시키는 D 플립플롭이 설치된다.

표 3. 설계된 ARM 프로세서의 명령어 집합
Table 3. The instruction set of the Designed ARM processor

명령어 유형	명령어
산술 논리	AND EOR SUB RSB ADD ADC SBC RSC TST TEQ CMP CMN ORR MOV LSL LSR ASR RRX ROR BIC MVN
곱셈	MUL MLA UMULL UMLAL SMULL SMLAL
메모리	STR LDR STRB LDRB STRH LDRH LDRSB LDRSH
분기	B BL

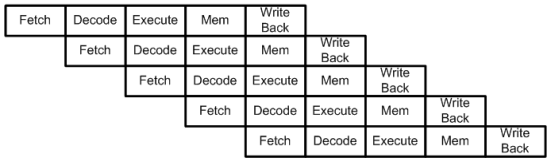

그림 1. ARM 프로세서의 5 단계 파이프라인
Fig. 1. The 5 stage pipeline of ARM processor

명령어 파이프라인에서 명령어 간에 쓰기 후 읽기 (RAW) 종속이 발생하는 경우를 해저드라고 하며, 해저드 유닛에서 처리한다. 인접한 명령어 간에 발생하는 해저드는 메모리단계의 결과를 포워딩시키고, 한 개의 명령어만큼 떨어진 명령어 간에 발생하는 해저드는 되쓰기 단계의 결과를 포워딩 시켜서 해결할 수 있다. 마지막으로, 두 개의 명령어만큼 떨어진 명령어 간에 발생하는 해저드는 레지스터 화일의 내부 포워딩에 의하여 해소된다[2,3].

그림 2에 본 논문에서 설계한 ARM 프로세서의 전체 블럭도를 보였다. 인출단계, 해독단계, 실행단계, 메모리 단계, 되쓰기단계의 각 사이마다 파이프라인 레지스터가 삽입되어 명령어 파이프라인 체계를 지원한다. 복잡성을 피하기 위하여, 제어신호는 선을 직접 연결하지 않고 동일한 제어신호 이름으로 연결 상태를 알 수 있게 하였다.

인출단계는 명령어 메모리와 3 개의 멀티플렉서, 덧셈기로 구성된다. 제 1 멀티플렉서는 PC의 순차 증가 어드레스 값과 PC 값을 업데이트하는 명령어의 결과값 중에서 하나를 선택한다. 제 2 멀티플렉서는 위의 값과, 채택된 분기의 ALU 연산 결과인 분기 어드레스 중에서 선택한다. 마지막으로 제 3 멀티플렉서는 위의 값과 인터럽트 벡터 어드레스 중에서 선택하는 기능을 수행한다. 명령어 메모리에는 ARM 프로세서가 실행할 명령어들이 기억되어있으며, 프로그램 카운터 (PC) 레지스터에 수록된 명령어 주소를 공급받아 명령어를 출력하는 기능을 수행한다. 인터럽트가 걸렸을 때 벡터 어드레스가 강제로 입력될 수 있도록 멀티플렉서를 설치하였다.

해독단계에서 레지스터 화일은 동시에 3 개의 피연산자를 읽고, 2 개의 결과값을 쓸 수 있도록 설계되었다. 레지스터 화일의 제 1 피연산자 입력은 R_n, R_{15}, R_{14}, R_m 중에서 선택한다. R_n은 일반 데이터처리 명령어에서 제 1 소오스 레지스터이며 명령어 필드의 19:16에 해당한다. R_{15}는 프로그램 카운터로 이용되는 레지스터이고, R_{14}는 인터럽트가 걸렸을 때의 복귀주소 또는 BL 분기 명령어에서의 복귀주소를 저장한다. R_m은 일반 데이터처리 명령어에서 제 2 소오스 레지스터이나, 곱셈 명령어에서는 제 1 레지스터에 해당되므로 위 단자로 공급된다. 레지스터 화일의 제 2 피연산자 입력은 R_m과 R_d를 멀티플렉서로 선택한다. R_m은 일반 데이터처리 명령어의 제 2 레지스터이며, 명령어 필드의 3:0에 해당한다.

실행단계는 ALU, 곱셈기, 쉬프터, 멀티플렉서, 조건플래그 단위 및 해독단계에서 파이프라인 래치를 통하여 넘어온 각종 제어신호 등으로 구성된다. ALU의 상단에 곱셈기와 쉬프터를 설치해서 산술논리연산 외에 곱셈과 쉬프팅 동작을 지원한다. ALU의 제 1 입력은 제 1 소오

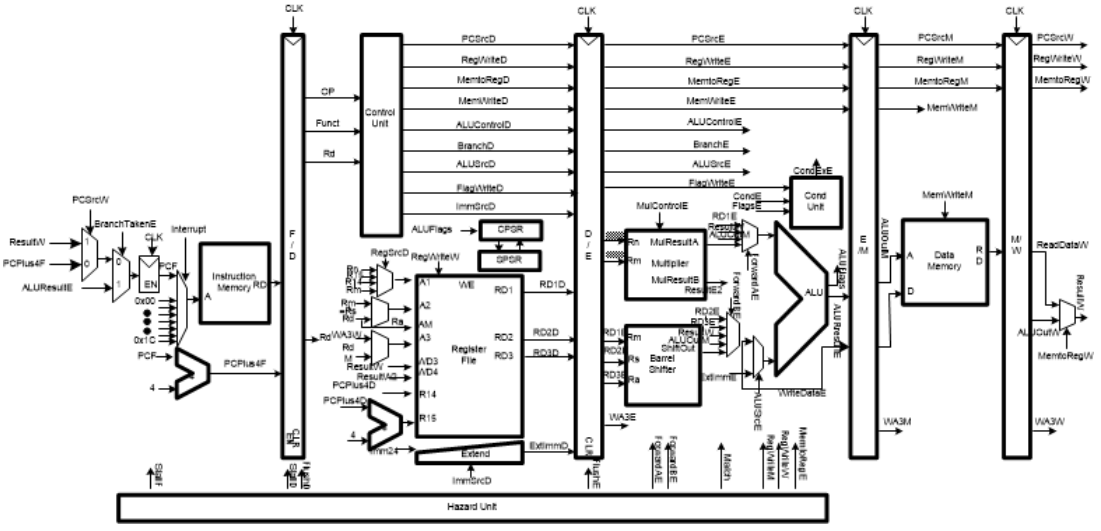

그림 2. ARM 프로세서의 전체 블럭도
Fig. 2. The block diagram of ARM processor

스 레지스터, 메모리 단계에서 포워딩되는 값, 되쓰기 단계에서 포워딩되는 값, 곱셈기에서 출력되는 값 중에서 멀티플렉서로 선택한다. ALU의 제 2 입력은 제 2 소오스 레지스터, 메모리단계에서 포워딩되는 값, 되쓰기 단계에서 포워딩되는 값, 부호확장기로부터 공급되는 값, 그리고 쉬프터의 출력 중에서 선택한다.

메모리단계는 데이터메모리와 실행단계에서 파이프라인 래치를 통하여 전달된 각종 제어신호들로 구성된다. 명령어 메모리의 주소 입력은 실행단계의 ALU 출력과 연결되며, 데이터 입력은 레지스터화일의 제 2 출력으로부터 공급된 데이터와 연결된다. 명령어 메모리의 출력은 되쓰기 단계의 파이프라인 레지스터로 전달된다.

되쓰기단계는 명령어 파이프라인 제 2단계인 해독단계에서 데이터를 읽어들이는데 이용하는 레지스터 화일에 반대로 데이터를 기록하는 과정이다. 파이프라인에서 레지스터 화일은 유일하게 읽기와 쓰기를 위하여 한 싸이클에 두 번 접근되는 장치이다. 로드인 경우에 명령어 메모리로부터 읽은 값과, ALU 연산인 경우에는 그 결과를 레지스터 화일에 역시 써야 한다.

IV. 모의실험

본 논문의 모의실험은 운영체제 Windows 10에서 3.1GHz로 동작하는 Intel Core i5-2400에서 시행하였다. ModelSim은 Intel 10.6d 버전을, VHDL 컴파일러는 1076-2002 버전을 이용하였다. 총 31 개의 VHDL 프로그램이 ARM 프로세서를 설계하는데 이용되었다. 표 4는 인터럽트 모의실험에 사용된 ARM 벡터테이블이다. 본 모의실험은 인터럽트 복귀시 오프셋이 0, -4, -8 인 3 가지 대표적 경우에 한하여 기술한다. 0x00 번지는 리셋이 걸렸을 때 명령어가 무조건 실행하여 0xA0번지로 분기하며, 0xA0 번지에는 사용자 프로그램이 시작하도록 설정하였다.

표 4. 인터럽트 모의실험에 이용된 벡터테이블
Table 4. The Vector Table used for interrupt simulation

주소	유형	기계어	어셈블리어
0x00000000	RESET	EA000026	B 0x26
0x00000004	UDEF	EA00000D	B 0xD
0x00000008	SWI	EA000011	B 0x11
0x0000000C	PABT	EA000014	B 0x14
0x00000010	DABT	EA000015	B 0x15
0x00000014	reserved	reserved	reserved
0x00000018	IRQ	EA000018	B 0x18
0x0000001C	FIQ	〈ISR begins〉	〈ISR begins〉
		E24FF004	SUBS LR,LR,#4
		E1B0F00E	MOVS PC,LR

모의실험을 위하여, 본 ARM 마이크로프로세서를 0xA0, 0xA4, 0xA8, 0xAC 번지의 명령어로 정상적으로 실행시키다가 0xB0에서 고속인터럽트(FIQ)를 인가했다.

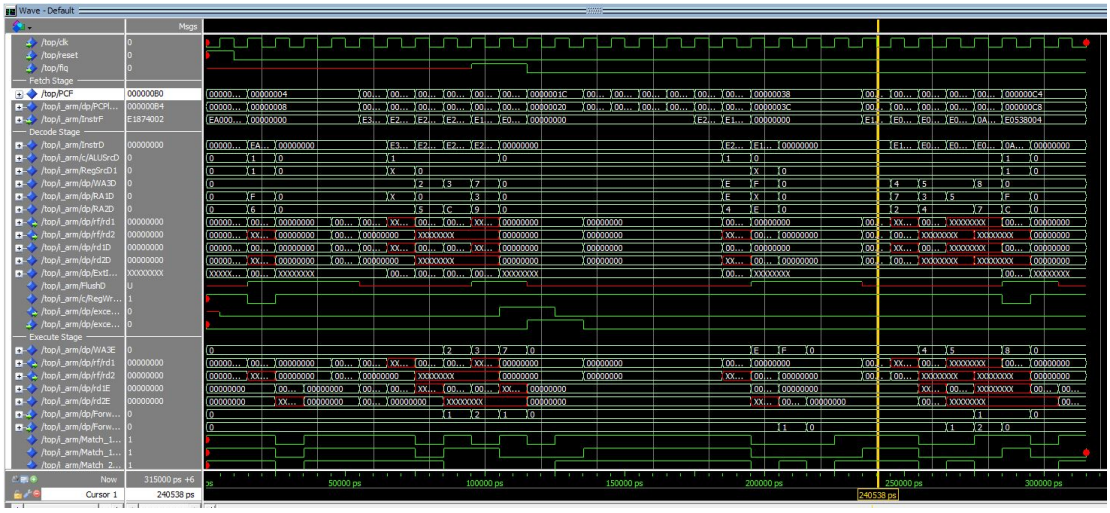

그림 3. ARM 프로세서의 인터럽트 ModelSim 실행결과
Fig. 3. ModelSim simulation waves of ARM processor with interrupts

이 때, LR 레지스터에 0xB0의 다음 주소인 0xB4 번지를 저장하고, 프로그램 카운터에 고속인터럽트의 벡터주소인 0x1C번지가 적재되어 해당 번지로 분기한다. FIQ 예외처리를 제외한 다른 예외처리 명령어들은 벡터 주소가 연속되어있어서 다시 한번 해당 인터럽트 서비스 처리 루틴으로 분기를 해야만 한다. 그러나, FIQ 예외처리의 경우 그 이하의 주소가 다른 예외처리 벡터주소로 사용되고 있지 않다. 따라서, FIQ는 다시 한번 다른 주소로 분기하지 않고 인접한 다음 주소에 인터럽트 서비스 루틴 코드를 적재 및 실행할 수 있다. FIQ의 인터럽트 서비스가 완료되면 SUB LR, LR, #4 명령어를 이용하여 복귀주소를 0xB0로 계산하고, MOVS PC, LR 명령어를 실행하여 PC에 이 값을 업데이트함으로써 인터럽트가 걸렸던 당시의 명령어로 복귀했다.

데이터중단(DABT) 인터럽트가 0xB0번지에서 발생하는 경우, 역시 LR 레지스터에 0xB4번지를 저장하며, 인터럽트 처리를 마친 후에 FIQ 인터럽트와는 달리 SUB LR, LR, #8 명령어를 실행하여 0xAC번지로 복귀했다. 마지막으로, 같은 주소에서 소프트웨어(SWI) 인터럽트가 발생한 경우, LR 레지스터에 0xB4 번지를 기록하고, 편차가 없으므로 LR 주소의 편차계산을 하지 않고 동일한 0xB4번지로 복귀하는 것을 확인했다. 그림 3에 ARM 프로세서에 고속인터럽트(FIQ)가 인가되었을 때, 실행되는 ModelSim의 결과 파형을 나타냈다. 그 결과, ARM 프로세서가 위에서 설명한 바와 같이 인터럽트 처리를 올바르게 실행하는 것을 확인했다.

V. 결 론

본 논문에서는 ModelSim 환경에서 VHDL을 이용하여 37 개의 명령어를 실행할 수 있고, 인터럽트 기능을 지원하는 ARM 프로세서를 설계 및 개발하였다. 주어진 ARM 어셈블리 프로그램을 모의실행시킨 결과, 7 가지 인터럽트가 모두 올바르게 처리되는 것을 확인할 수 있었다.

추후로, 실제의 ARM 프로세서에서 제공하고 있지만 본 설계에서 누락된 블록 데이터 명령어와 부동소수점 연산 명령어를 포함시키는 것이 목표이다. 블록 데이터 명령어는 로드나 스토어 명령어 한 개로 여러 개의 블록을 레지스터와 메모리 간에 전송시키며, 스택을 이용하는 인터럽트 처리에도 필요하다. 부동 소수점 연산유닛은 Cortex-M4와 같은 일부 ARM 계열의 프로세서에만 내장되어있다. 본 프로세서에 부동 소수점 연산유닛을 추가로 설계한다면 실수형 덧셈, 뺄셈, 곱셈, 나눗셈 등의 실수형 연산을 지원할 수 있다.

설계된 ARM 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 프로그래밍하여 동작을 검증할 예정이다[4-11]. FPGA로 검증을 완료한 후에는, Synopsis의 DC(Design Compiler) 또는 유럽의 Alliance로 합성하여 국내 기관인 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.

[3] S. L. Harris, and D. M. Harris, "Digital Design and Computer Architecture ARM Edition", Elsevier Korea LLC, 2016.

[4] F.J. Jurado Carmona, J. Tombs, M.A. Aguirre Echanove and A. Torralba, "Implementation of a fully pipelined ARM compatible microprocessor core," XVII Design on Circuits and Integrated Systems Conference, 2002, pp. 559-563.

[5] J. S. Pastor, I. Gonzalez, J. Lopez, F.G. Arribas, J. Martinez, "A Remote Laboratory for Debugging FPGA-Based Microprocessor Prototypes," Proceedings of the IEEE International Conference on Advanced Learning Technologies, 2004.

[6] A.A. Morgan, M.E. Allam, M.A. Salama, and H.A.K Mansourm "Implementation of an ARM Compatible Processor Core for SOC Designs," 2005 International Conference on Information and Communication Technology, Dec. 2005.

[7] M. M. Kinage and D.G. Khairnar, "Design and Implementation of FPGA Soft Core Processor for Low Power Multicore Embedded System using VHDL," Sep. 2016.

[8] J. Davidson, "FPGA Implementation of a Reconfigurable Microprocessor," IEEE Custom Integrated Circuits Conference, 1993, pp.3.2.1-3.2.4.

[9] E. Alaer, A. Tangel, M. Yakut, "MIB-16 FPGA based Design and Implementation of a 16 Bit Microprocessor for Educational Use," 6th WSEAS International Conference on Circuits, Systems,

Electronics, Control & Signal Processog, 2007. pp.284-288.

[10] H.S. Herman, C. Srihari, M. Matthew, "Pipeline Reconfigurable FPGAs," Journal of VLSI Signal Processing Systems,", 2000, Vol. 24, pp. 129-146

[11] 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.

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 전자정보공학과 교수
- Tel : 02-760-4497
- Fax : 02-760-4435
- E-mail : jblee@hansung.ac.kr
- 관심분야 : 마이크로 프로세서, 멀티코어 프로세서, 텐서 프로세서 유닛.

※ 본 연구는 한성대학교 교내학술연구비 지원과제임.