ARM9 호환 Microprocessor의 FPGA 구현

오 민 석, 김 재 우, 남 기 훈, 김 명 환, 이 광 엽
서경대학교 컴퓨터공학과
전화 : 02-940-7240 / 핸드폰 : 011-249-3619

FPGA Implementation of ARM9 Compatible Microprocessor

Min-Seok Oh, Jae-Woo Kim, Ki-Hoon Nam, Myeong-Hwan Kim, Kwang-youb Lee
Dept. of Computer Engineering, Seokyeong University
E-mail : omani21@dreamwiz.com

Abstract

본 논문에서는 로드 명령어 처리와 곱셈기의 구조를 개선한 ARM9 호환 마이크로프로세서를 설계하였으며, ARM9 마이크로프로세서와 비교하여 특정한 로드 명령어 수행 시 1 클록 사이클을 단축하였고, 곱셈 명령어 수행 시 2 클록 사이클 단축하였다. 설계된 ARM9 프로세서는 VHDL로 기술하였으며, 명령어 시뮬레이션 결과 ARM9 마이크로프로세서 시뮬레이터와 실행 결과 값이 동일함을 확인하여 명령어 호환 검증을 하였으며, Xilinx FPGA를 이용하여 66MHz 동작 환경에서 실시간 영상 처리 수행을 검증하였다

I. 서론

최근 Core IP(Intellectual Property) 개발 및 이를 이용한 SoC 설계가 새로운 산업으로 등장하고 있으며, 휴대폰 및 PDA 등의 모바일 제품들의 급격한 수요로 인하여 SoC를 통한 모바일 임베디드 시스템(Embedded System) 설계 시 제한된 배터리 용량에서 보다 긴 사용시간을 갖는 저전력 core IP의 사용이 필수적이다. ARM9TDMI 마이크로프로세서는 타 CPU 제품에 비해 저전력, 저면적 특성으로 인해 현재 SoC 설계 시 CPU core IP로 널리 사용되고 있다. 그러나 연구 목적의 SoC 설계에서 ARM9TDMI 마이크로프로세서 IP를 사용 시 지불해야하는 막대한 비용은 크게 부담이 되므로 CPU core IP의 성능 향상을 위한 연구가 불가능하다.

본 논문에서는 SoC 설계 연구 및 CPU IP 성능 향상 연구를 위해 ARM9TDMI 마이크로프로세서와 같은 명령어를 갖는 ARM9 호환 마이크로프로세서를 설계하였으며 기존 ARM9TDMI 마이크로프로세서보다 성능 향상을 위해 곱셈기의 구조와 Load명령어의 처리를 개선하여 수행 시간을 줄였다. 설계된 마이크로프로세서는 VHDL로 기술하였으며, 시뮬레이션을 통하여 명령어 호환을 검증하였으며, FPGA 검증을 통하여 기술된 VHDL 코드의 합성 가능을 확인하였다.

※ 본 연구는 ETRI와 IT_SoC 사업단 지원으로 수행되었습니다.

II. 본론

2.1 ARM9 프로세서의 구조

ARM9 마이크로프로세서는 ARM7 마이크로프로세서와 같이 ARM architecture V4T 명령어를 사용하고 있으며, 16-bit 압축 명령어인 Thumb 역시 명령어 인출 및 명령어 확장 방법이 동일하다[1]. 그러나 파이프라인 단계가 3 단계인 ARM7과 달리 ARM9는 명령어 인출, 명령어 해석, 명령어 수행, 메모리, 레지스터 쓰기로 5단계 파이프라인으로 구성되어 있다[1][2]. 본 논문의 ARM9 호환 마이크로프로세서는 ARM9와 완벽한 명령어 호환을 위해 기존에 설계된 ARM7 호환 마이크로프로세서의 명령어들을 5단계 파이프라인 구조

에 맞추어 제정의 하였으며, 이로 인해 발생하는 파이프라인 hazard를 해결하기 위해 interlocking 및 forwarding 회로를 추가하였다. 또한 명령어의 수행 사이클 수와 외부 포트 신호 타이밍은 ARM9TDMI Technical Reference Manual(Rev 3)[3]과 같다.

그림 1은 설계된 ARM9 호환 마이크로프로세서의 전체 블록도 이며 데이터 패스는 32 비트 레지스터 뱅크, 32-bit 배럴 쉬프터, ALU(Arithmetic Logic Unit), 곱셈기, 메모리 접근을 위한 address 계산 블록, 데이터 정렬을 위한 aligner, PC, 파이프라인을 위한 파이프라인 레지스터로 구성되어 있다[1].

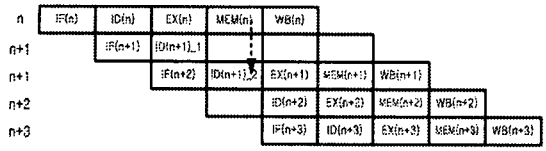

그림 2. 1 사이클 interlock 경우

그림 2는 hazard가 발생한 경우 hazard를 해결하기 위한 interlocking과 forwarding을 나타내고 있다. 그림 2와 같이 데이터 의존성이 발생하게 되면 컨트롤에서는 이를 파악하여 n 번째 명령어의 연산 결과를 interlocking을 통해 forwarding이 가능할 때 까지 명령어를 지연시켜 forwarding 경로를 통하여 n+1 번째 명령어로 n 번째 명령어의 연산 결과를 forwarding하여 n+1 번째 명령어가 n 번째 명령어 다음에 수행 할 수 있도록 해준다.

2.2 성능 향상을 위한 제안된 구조

(가) PC 레지스터를 목적지 레지스터로 갖는 로드 명령어

ARM9 마이크로프로세서는 로드 명령어가 PC 레지스터를 목적지 레지스터로 설정되어있을 경우 데이터 메모리로부터 로드 된 데이터가 WB 단계에서 명령어의 설정에 맞게 정렬되므로 로드 명령어를 뒤따르는 4개의 명령어들은 파이프라인에서 수행이 취소되며 5 클록 사이클 후에 데이터 메모리로부터 로드 된 명령어 메모리 주소의 명령어로 이동하게 된다. 그러나 정렬된 워드 단위를 로드하는 명령어에 의해 로드 된 데이터는 WB 단계에서 데이터 정렬이 필요 없으므로 MEM 단계에서 명령어 주소를 발생할 수 있으며, 이 경우에는 4 클록 사이클 후에 목적 명령어로 이동할 수 있어 1 사이클의 수행 시간을 단축 할 수 있다. 그림 3은 로드한 데이터가 WB 단계에서 데이터 정렬이 필요하지 않을 경우 4 사이클 후에 로드된 명령어 메모리 주소의 목적 명령어로 이동하는 동작을 나타낸다.

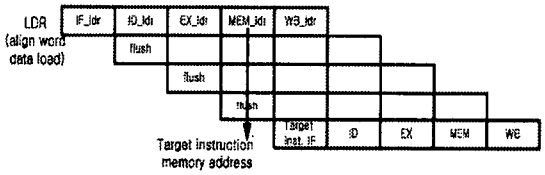

그림 3. 4 cycle 수행 LDR(Rd = PC) 명령어

(나) 32-bit × 12-bit booth 곱셈기

ARM9 마이크로프로세서의 곱셈 명령어는 여러 EX

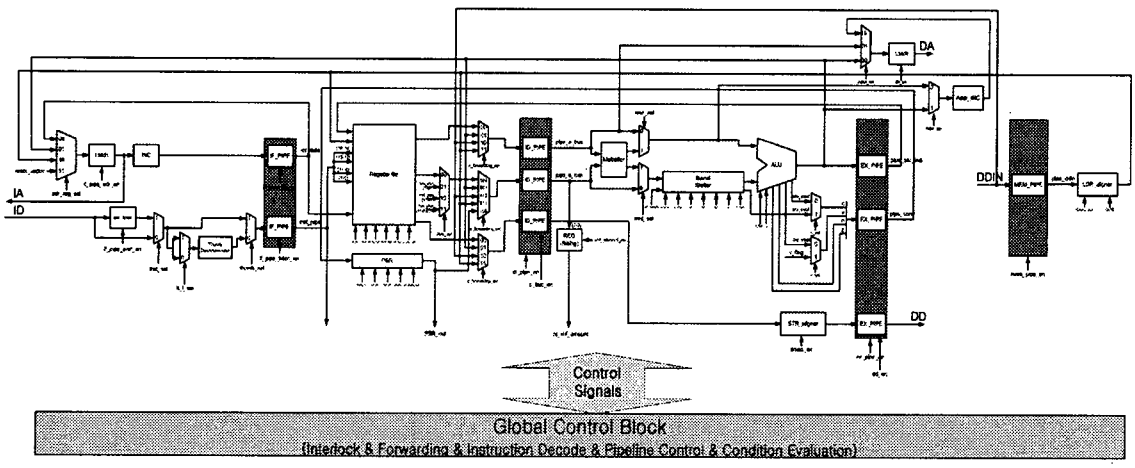

그림 1. ARM9 호환 마이크로프로세서의 구성

단계 과정 갖으며, 초기화, 곱셈기, ALU 사이클 과정

으로 구성된다. 초기화 과정은 1 클록 사이클을 소모하며, 곱셈기의 부분 합, 캐리 레지스터를 초기화하는 것으로 곱셈 후 덧셈 형태의 명령어 경우 내부 레지스터를 덧셈 오퍼랜드로 초기화 시키고, 단순 곱셈 형태의 명령어는 0으로 초기화 시킨다. 초기화 과정을 마친 후 곱셈기 과정은 32-bit × 8-bit 전용 곱셈기를 이용하여 처리된다. 32-bit × 8-bit 전용 곱셈기는 booth 알고리듬을 이용하여 1 클록 사이클 당 32-bit × 8-bit 결과를 발생하며, 곱셈 승수 오퍼랜드에 따라 1 ~ 4 클록 사이클을 반복 수행하여 처리한다. 그림 4는 ARM9 마이크로프로세서가 곱셈 연산 수행 시 1 클록 사이클에 발생하는 부분 곱을 나타낸다.

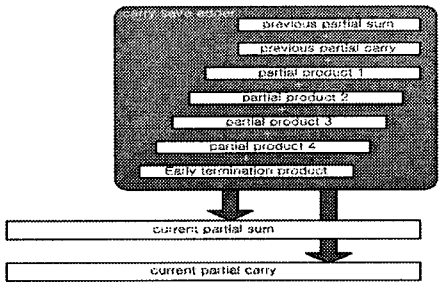

그림 4. ARM9 곱셈기의 부분 곱

그림 4에서 부분 곱 1부터 4까지는 8-bit를 radix-4 booth encoding 과정을 거쳐 발생한 4 개의 부분 곱이며, 누적 부분 합과 캐리는 현재 곱셈 사이클에서 이전 클록 사이클에서 생성된 부분 합과 캐리를 누적시키기 위한 것이다. early termination product는 곱셈기의 수행 사이클이 4 클록 사이클 이전에 종료되는 경우 곱셈기 구조상 발생하는 부호 보정을 위한 것이다. 위에서 언급한 7개의 부분 곱은 CSA(Carry Save Adder)로 더해져 현재 부분 합과 캐리로 저장된다. 곱셈기 사이클 과정 종료 시 ALU로 32-bit 단위로 전달되어 덧셈 연산을 거쳐 목적지 레지스터에 저장되며, 32-bit 결과를 발생하는 곱셈 명령어 경우 1 클록 사이클을 소모하며, 64-bit 결과를 발생하는 곱셈 명령어 경우 2 클록 사이클을 소모한다. 그러므로 32-bit 결과를 발생하는 곱셈 명령어 경우 총 3 ~ 6 클록 사이클을 소모하며, 64-bit 결과를 발생하는 곱셈 명령어 경우 총 4 ~ 7 클록 사이클을 소모한다.

본 논문의 마이크로프로세서는 현재 ARM9 마이크로프로세서가 곱셈 명령어가 수행하기 위해 소모되는 클록 사이클 수를 줄이기 위하여 초기화 과정과 곱셈기의 구조를 개선하였다. 초기화 과정의 경우 레지스터 파일에서 곱셈 오퍼랜드와 덧셈 오퍼랜드를 동시에 공급하여 별도의 초기화 과정을 위해 소모되는 클록 사이클을 줄였다. 그림 5에서 곱셈기의 첫 번째 클록 사이클에서 CSA에 공급하는 초기화된 누적 부분 합, 캐리 대신 0 또는 레지스터 C 포트를 통해 전달되는 덧셈 오퍼랜드를 직접 공급하여 곱셈기의 첫 번째 사이클과 초기화 사이클을 통합하였다.

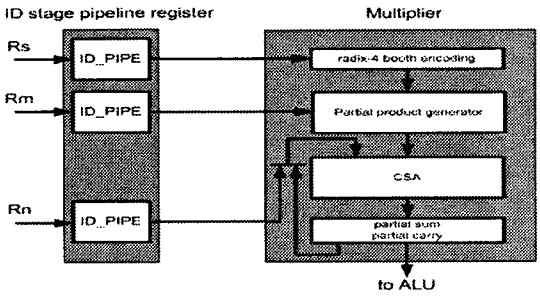

그림 5. 곱셈기 데이터 패스

제안하는 곱셈기의 구조는 1 클록 사이클 당 32-bit × 12-bit booth 곱셈을 수행하며, 곱셈 승수 오퍼랜드에 따라 1 ~ 3 클록 사이클을 반복 수행한다. ARM9 마이크로프로세서의 32-bit × 8-bit booth에 비하여 1 클록 사이클 당 2개의 부분 곱이 증가하여 부분 곱을 더하는 CSA가 9개의 부분 곱을 처리해야하는 부담이 있으나, 전체 곱셈기 수행 사이클은 1 클록 사이클 감소한다. 제안하는 곱셈기는 ARM9 마이크로프로세서와 같이 early termination 기능이 있으며, 표 1은 제안한 곱셈기의 수행 사이클을 나타내며, 표 2는 제안한 마이크로프로세서의 곱셈 명령어 수행 사이클 수이다.

표 1. 제안하는 곱셈기 수행 사이클 수

명령어	곱셈 승수 : Rs	곱셈기 수행
signed	[31:12] = all 0 or 1	1 사이클
	[31:24] = all 0 or 1	2 사이클
unsigned	[31:12] = all 0	1 사이클
	[31:24] = all 0	2 사이클
-	그 외의 경우	3 사이클

표 2. 곱셈 명령어 수행 사이클 수

명령어	초기화	곱셈기	ALU	합계
MUL, MLA	0	1 ~ 3	1	2 ~ 4
SMULL, UMULL	0	1 ~ 3	2	3 ~ 5
SMLAL UMLAL	1	1 ~ 3	2	4 ~ 6

Ⅲ. 실험 및 결과

3.1 제안하는 곱셈기의 실험

ARM9 호환 마이크로프로세서의 성능을 개선하기 위해 사용된 곱셈기의 Xilinx FPGA 합성 결과 및 처리 속도와 수행 사이클 수를 표 3에 정리하였다.

표 3. 곱셈기의 성능 비교

	ARM9 곱셈기	개선된 곱셈기
수행 속도	91.327 MHz	70.235 MHz
면적 (XCV1000E)	Slice : 695개 FF : 112개 LUT : 1307개	Slice : 903개 FF : 112개 LUT : 1512개
수행 사이클	1~4 cycle	1~3 cycle

본 논문에서 곱셈 명령어 수행 사이클 단축을 위해 제안한 32-bit × 12-bit booth 곱셈기는 표 3에서와 같이 ARM9의 32-bit × 8-bit booth 곱셈기에 비해 수행 시간과 면적이 늘어났지만 기존 곱셈기에 비해 수행 사이클의 수를 1 사이클 단축시킬 수 있어 전체적으로 수행 시간을 단축시킬 수 있다.

3.2 제안하는 ARM9 호환 마이크로프로세서 FPGA 검증

설계된 ARM9 마이크로프로세서는 PCI 버스 타입 Xilinx FPGA 보드인 iProve(XCV1000E-iPROVE)를 이용하여 66MHz 동작 환경에서 수행 검증을 하였으며, 3405개의 slice(27%), 2416개의 register(9%), 5981개의 LUT(24%)로 구성되었다. 개선된 ARM9 호환 마이크로프로세서의 검증 환경은 PCI 버스를 통해 호스트 PC의 가상 명령어 메모리로부터 채도 극대화 프로그램의 명령어들을 전달 받으며, 호스트 PC의 실시간 8 bit 영상 데이터를 전달받아, 채도 극대화 처리된 영상 데이터를 메모리로 전달한다. 그림 7은 ARM9 호환 마이크로프로세서 검증 환경의 GUI(Graphic User Interface)로 왼쪽 위 부분이 PC 카메라로부터 받은 영상이며, 오른쪽 위부분이 ARM9 호환 마이크로프로세서가 채도 극대화 알고리듬을 수행하여 출력하는 영상이다. 아래쪽은 ARM9 호환 마이크로프로세서가 수행하는 채도 극대화 알고리듬의 ARM9 명령어들이다. ARM9 호환 마이크로프로세서는 320*240 실시간 영상의 채도 극대화 연산을 초당 25~30 프레임 속도로 처리함을 확인하였다.

그림 7. FPGA 검증 GUI

Ⅳ. 결론

본 논문에서는 로드 명령어 처리와 곱셈기의 구조를 개선한 ARM9 호환 마이크로프로세서를 설계하였으며, ARM9 마이크로프로세서와 비교하여 특정한 로드 명령어 수행 시 1 클록 사이클을 단축하였고, 곱셈 명령어 수행 시 2 클록 사이클 단축하였다. 설계된 ARM9 프로세서는 VHDL로 기술하였으며, 명령어 시뮬레이션 결과 ARM9 마이크로프로세서 시뮬레이터와 실행 결과 값이 동일함을 확인하여 명령어 호환 검증을 하였으며, Xilinx FPGA를 이용하여 66MHz 동작 환경에서 실시간 영상 처리 수행을 검증하였다. 본 논문에서 설계된 ARM9 호환 마이크로프로세서는 합성 가능한 VHDL로 설계되어있어 SoC 설계 시 SoC 성능 사양에 맞게 개량할 수 있으며, 성능 향상을 위해 여러 알고리즘을 적용할 수 있을 것이라 사료된다.

참고문헌

[1] Steve furber, *ARM system-on-chip architecture second edition*, Addison wesley, 2000.

[2] David seal, *ARM Architecture Reference Manual second edition*, Addision wesley, 2000.

[3] 서보익, 배영돈, 박인철, *"Synthesizable ARM9 호환 CPU의 설계"*, 대한전자공학회, 추계종합학술대회 논문집, 제23권, 제2편, 2000.

[4] Advanced RISC Machines, *ARM9TDMI Techinical Reference Manual*, Rev 3, 2000.