가변형 레지스터를 가지는 저전력 마이크로컨트롤러

고 규 영
전북대학교 전자정보공학부

이 종 열
전북대학교 전자정보공학부

Ⅰ. 서 론

산업이 발달 할수록 의료, 보안, 스마트 그리드 등 다양한 분야에서 전자기기의 사용은 점점 증가하고 있다. 이러한 전자기기들의 수요 증가로 전력문제가 중요해지고 있다. 이러한 전력문제를 해결하기 위한 방안으로 배터리 용량의 증가 및 초저전력 기기의 개발 및 관련 연구가 행해지고 있다. 이 중 저전력 기기를 개발함에 있어서 기존의 연구들로 RTL(Register Transfer Level) 설계시 다중 전압 설계, 합성시 클럭게이팅 구현, 레지스터 클러스터화 및 분산화, 누설전력 저감, 유효 블록에 전력 증가 등의 분야에서 진행되었다.

전자기기에서 반드시 필요한 부품 중 하나가 마이크로프로세서이다. 그리고 이러한 마이크로프로세서는 모바일 기기, 센서 네트워크 등 장시간 배터리 대기 수명을 요구하는 기기 또는 저전력을 요구하는 분야에서 많이 사용되면서 마이크로 컨트롤러의 소비전력이 중요시되고 있다. 이에 따라 각 회사별로 초저전력 마이크로 프로세서를 개발하고 있다. 다음 〈표 1〉은 각 회사별로 초저전력 마이크로 프로세서 종류와 성능을 나타낸 표이다.

〈표 1〉 회사별 마이크로프로세서 종류와 소비전력

Company	Product	Core	Bit	Sleep Current	Active Current
Texas Instrumets	MSP430	MSP430	16	0.1uA	200uA
Maxim	MAX	MAM RISC	16	0.2uA	2000uA
Siliconlabs	C8051	Siliconlab	8	0.05uA	110uA
Energymicro	EFM32	ARM coretex-M3	32	0.6uA	180uA
STmicroelctronics	STM	STM8	8	0.35uA	150uA
ATmel	ATmega	AVR	16	0.1uA	900uA

〈표 1〉중에서 TI(Texas Instruments)사의 MSP430은 16비트 RISC(Reduced Instruction Set Computer) 마이크로컨트롤러로서 초저전력, 온칩 내장형 연산증폭기, 고성능 12비트 ADC, DAC, 외부 포트 인터럽트, 멀티 클럭 소스, 호환성 높은 통신 기능 등의 뛰어난 성능을 보유하고 있는 마이크로컨트롤러이다[1].

MSP430의 주요 특징은 다음과 같다.

- 0.1uA RAM 유지전류
- 1uA 이하의 RTC 동작전류
- MIPS당 250uA 이하의 동작전류
- 50nA 이하의 핀 누설 전류
- 20kHz 이상의 내부 저전력 오실레이터
- 32.768kHz 의 외부 저주파 크리스털 지원
- 내부 디지털 제어 오실레이터

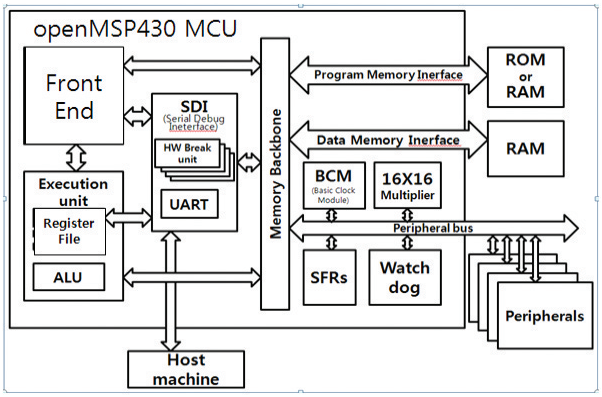
마이크로컨트롤러는 같은 Core를 사용하는 Family 제품이라 하더라도 그 특성상 사용 목적이나 환경, 가격에 따라 성능이나 제품 사양이 크게 달라진다. 하지만 하드웨어 특성상 특정한 환경에 최적화된 제품 개발은 어렵다. 본 논문은 MSP430의 공개 버젼인 openMSP430을 기반으로 마이크로 컨트롤러가 특정한 환경에서 사용되어 질 때 환경에 최적화된 하드웨어를 제공하여 저전력을 달성 할 수 있는 가변형 레지스터를 가진 저전력 마이크로컨트롤러를 기술한다.

RISC(Reduced Instruction Set Computer)는 CPU 명령어의 개수를 줄여 하드웨어 구조를 좀 더 간단하게 만드는 방식으로, 마이크로프로세서를 설계하는 방법 가운데 하나이며, ARM, MIPS 등의 아키텍처에서 사용된다. RISC에 대되는 개념으로 CISC (Complex Instruction Set Computer)가 있다.

Ⅱ. 본 론

1. openMSP430

openMSP430은 16비트 RISC 마이크로컨트롤러로서 Texas Instrument사의 MSP430을 모델로 opencores에서 개발한 마이크로 컨트롤러이다. 현재



〈그림 1〉 openMSP430 Block Diagram[2]

Verilog HDL로 개발되어 FPGA(Field Programmable Gate Array)나 ASIC(Application Specific Intergrated Circuit)등으로 제작 및 테스트 할 수 있다. openMSP430은 16×16 곱셈기, GPIO(General Parallel I/O), Timer A Watchdog 등의 몇가지 주변장치를 포함하고 있다. 실제 판매되고 있는 MSP430과 동일한 ISA(Instruction Set Architecture)를 가지고 있고 명령어 메모리와 데이터 메모리를 하나의 메모리에서 사용하는 폰 노이만 아키텍처[3]를 사용하는 등 구조 역시 동일하다. 이에 따라 MSP430에 쓰이는 컴파일러나 관련 소프트웨어 툴 등 MSP430 toolchain을 사용할 수 있다 .

〈그림 1〉에서와 같이 openMSP430은 Frontend, Excution unit, Serial Debug Interface, Memory Backbone, SFRs(Special Function Registers) 등으로 이루어져 있다. 먼저 Frontend는 실행해야 할 명령어의 명령어 페치 및 디코드 작업을 수행하는 블록이다. 또한 해당하는 명령어의 작업 실행을 설정하는 Frontend FSM(Finite state machine) 과 Execution unit FSM을 포함하고 있다. 다음으로 Execution unit은 ALU(Arithmetic logic unit)와 Register file로 이루어져 있다. 디코딩 된 명령어를 실행하고 실질적인

연산 및 작업을 실행한다. Serial Debug Interface 블록은 Host와의 통신을 위한 Serial Interface다. Memory Backbone은 Frontend와 Execution unit을 연결하는 블록으로 데이터와 주변장치, 메모리에 접속한다. Basic Clock Module은 openMSP430의 클럭 도메인 생성 및 상황에 따른 저전력 모드를 관리하는 블록이다. SFRs는 특수 레지스터이다. Watchdog은 NMI 인터럽트와 PUC Reset 발생을 감지하기 위한 장치로 소프트웨어에 문제가 일어난 경우, 시스템을 다시 시작하는데 사용된다. 16×16 곱셈기는 16×16, 16×8, 8×16, 8×8비트 곱셈 동작을 지원하며, 부호있는 곱셈 또는 부호 없는 곱셈을 할 수 있다.

openMSP430은 초저전력을 달성하기 위하여 총 3개의 클럭을 사용하고 있다. 3개의 클럭은 MCLK(Master Clock), SMCLK(Sub-Main Clock), Auxiliary Clock(ACLK) 이며 각각의 용도는 다음과 같다.

- MCLK : CPU에 공급되어지는 마스터 클럭
- SMCLK : 주변장치에 공급되어지는 클럭
- ACLK : 저전력 주변장치를 위한 클럭

또한 openMSP430은 다양한 클럭을 활용하여 하나의 활성화 모드(Active Mode)와 5가지의 저전력 모드(Low Power Mode)를 제공한다. 〈표 2〉는 openMSP430의 활성화 모드와 5가지의 저전력 모드에서의 CPU 및 클럭 상태를 나타낸 것이다. 저전력 모드는 SR(Status Register)를 이용하여 저전력 모드를 선택하면, 선택된 저전력 모드 상태로 프로세서가 동작된다. 저전력 모드로 들어간 경우, 모든 I/O 포트 핀과 RAM/레지스터의 상태는 변하지 않게 된다. 인터럽트 요청이 있으면 저전력 모드를 벗어나 인터럽트 처리를 한 후, 다시 선택된 저전력 모드로 들어가게 된다. 인터럽트 처리 루틴으로 들어가면, PC(Program Counter)와 SR은 스택에 자동으로 저장되고 CPUOFF, SOG1 및 OSCOFF는 자동으로 리셋된다[4].

〈표 2〉 openMSP430 실행 모드[1]

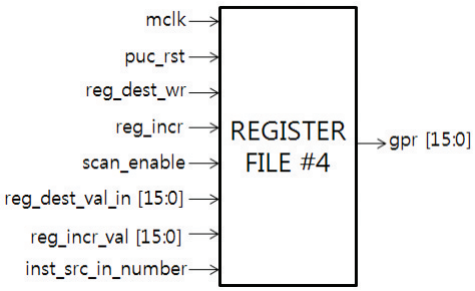
모드	CPU 및 클럭 상태
Active Mode	CPU, MCLK, SMCLK, ACLK 동작
LPM0	CPU, MCLK 동작 불가능 SMCLK, ACLK 동작
LPM1	CPU, MCLK, DCO 발진기, DC 발생기 동작 불가능 SMCLK, ACLK 동작
LPM2	CPU, MCLK, SMCLK, DCO 발진기 동작 불가능 DC 발생기, ACLK 동작
LPM3	CPU, MCLK, SMCLK, DCO 발진기, DC 발생기 동작 불가능 ACLK 동작
LPM4	CPU, MCLK, SMCLK, ACLK 동작 불가능

2. Signal Gating

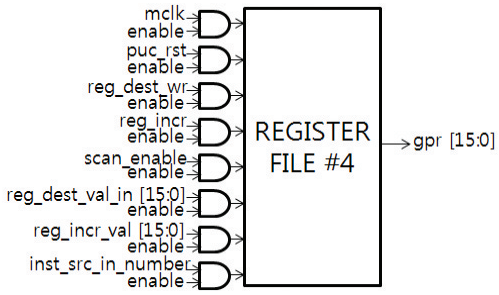
마이크로컨트롤러는 사용 환경이나 목적에 따라 요구되어지는 성능이 달라진다. 이에 따라 사용되어지는 하드웨어도 제한적이게 된다. 하지만 가격적인 측면상 각각 그 요구에 맞추어 설계를 할 수 없고 일반화된 제품을 사용하게 된다. 이러할 경우 전력부분에 있어서는 불필요한 하드웨어 동작으로 전력을 소모하게 된다.

MSP430과 같은 마이크로 컨트롤러들은 대부분의 경우 가스센서, 온도센서, 압력센서, 유량 측정기기, 일부의료기기 등은 단순히 저성능의 간단한 동작만을 수행하는 것을 필요로 한다. MSP430의 경우 16비트 16개의 레지스터를 가진 CPU이다. 레지스터의 개수는 고정적이지만 이러한 마이크로컨트롤러가 사용되는 특정한 환경에서는 실질적으로 이 16개의 레지스터 중 일부만 사용된다. 뿐만 아니라 일부분의 레지스터 파일은 지속적으로 사용하지 않아도 시스템 동작에 별문제가 없다. 하지만 하드웨어 특성상 사용되어지지 않는 레지스터에도 신호가 들어가게 되고 이에 따른 전력 소비가 발생하게 된다. 다음 〈그림 2〉는 연산 및

폰 노이만 구조는 존 폰 노이만이 고안한 내장 메모리 순차처리 방식이다. 데이터 메모리와 프로그램 메모리가 구분되어 있지 않고 하나의 버스를 가지고 있는 구조를 말한다.



〈그림 2〉 openMSP430 레지스터 파일 #4



〈그림 3〉 Signal Gating을 적용한 레지스터 파일#4

처리로 사용되어지는 16개의 General Purpose Register File 중 4번 레지스터 파일 하나의 입력 신호와 출력신호를 나타낸 그림이다.

각 신호에 대한 설명은 다음과 같다.

- mclk : 메인 시스템 클럭
- puc_rst : 메인 시스템 리셋
- reg_dest_wr : 타겟 레지스터 쓰기
- ret_incr : 소스 레지스터 증가
- scan_enable : 레지스터의 사용 여부
- reg_dest_val : 타겟 레지스터의 값
- reg_incr_val : 소스 레지스터 증가 값
- inst_dest_number : 타겟 레지스터 선택
- inst_src_in_number : 소스 레지스터 선택
- gpr : 레지스터 파일 출력 신호

해당 레지스터 파일이 사용되어지지 않더라도 mclk, reg_dest_val, reg_incr_val 같은 신호에는 지속적으로 신호가 들어가게 되고 이에 따라 불필요한 스위칭 전력이 소비 된다. 이러한 동적 전력 소비를 줄이고자 Signal Gating을 구현했다. Signal Gating을 구현한 설계는 〈그림 3〉에 제시되었다. Signal Gating은 입력 신호들에 AND 게이트를 추가하여 사용되어지지 않는 레지스터 파일에 enable 신호를 보내 입력 신호를 차단한다. 컴파일 단계에서 사용되어지지 않는 레지스터 파일을 검색하여 관련 enable 신호를 이용 입력 신호를 차단하는 Signal Gating 하게 되면 레지스터 파일에서 발생하는 불필요한 동적 전력 소비를 줄일 수 있다. 또한 마이크로 컨트롤러의 사용 목적이나 환경에 맞게 사용되어지는 레지스터 파일의 개수를 임의로 정할 수 있어 좀 더 최적화된 하드웨어를 제공하게 되고 이는 저전력 효과를 가져오게 된다.

general purpose register는 프로세서 레지스터 중의 하나로 범용 레지스터라고도 불리며, 주소와 데이터를 모두 저장할 수 있는 레지스터이다. 프로세서 레지스터는 컴퓨터의 프로세서 내에서 자료를 보관하는 아주 빠른 기억 장소이다. 일반적으로 현재 계산을 수행중인 값을 저장하는 데 사용된다. 대부분의 현대 프로세서는 메인 메모리에서 레지스터로 데이터를 옮겨와 데이터를 처리한 후 그 내용을 다시 레지스터에서 메인 메모리로 저장하는 로드-스토어 설계를 사용하고 있다.

3. 레지스터 파일 구조

openMSP430에는 16개의 레지스터가 있다. 이중 R0~R3까지는 Special Purpose Register로서 고유한 기능을 가지고 R4~R15까지는 General Purpose Register로서 일반적인 목적으로 사용된다.

R0레지스터는 PC(Program Counter)레지스터로서 실행될 다음 명령어를 가리킨다. R1 레지스터는 SP(Stack Pointer) 레지스터로서 인터럽트 또는 서브루틴을 처리한 후 되돌아갈 리턴 어드레스를 임시적으로 저장하는 목적으로 사용되는 레지스터이다. R2는 SR(Status Register)로서 CPU 내부의 ALU(Arithmetic Logic Unit)의 연산결과를 반영하는 상태비트(V, N, Z, C)와 시스템의 클럭을 제어하는

〈표 3〉 상태 레지스터 SR의 비트 정의[1]

비 트	설 명
V	오버플로우 비트
N	네거티브 비트
Z	제로 비트
C	캐리 비트
SCG0	시스템 클럭 발생기0
SCG1	시스템 클럭 발생기1
OSCOFF	오실레이터 OFF
CPUOFF	CPU OFF
GIE	광역 인터럽트 활성화

제어비트 (SCG1, SCG0, OSCOFF), 광역 인터럽트 제어 비트 (GIE)로 구성된다[4]. 〈표 3〉은 상태 레지스터의 상태 비트 및 제어비트들을 요약한 표이다.

또한 R2와 R3는 CGR(Constant Generator Register 1, 2)로서 자주 사용되는 6개의 상수를 추가적인 코드 없이 자동으로 발생시킨다. 발생되는 상수값은 다음의 〈표 4〉에서와 같이 소스 레지스터의 어드레싱 모드에 의하여 결정된다.

R2, R3와 같은 상수 발생기의 장점은 다음과 같다[5].

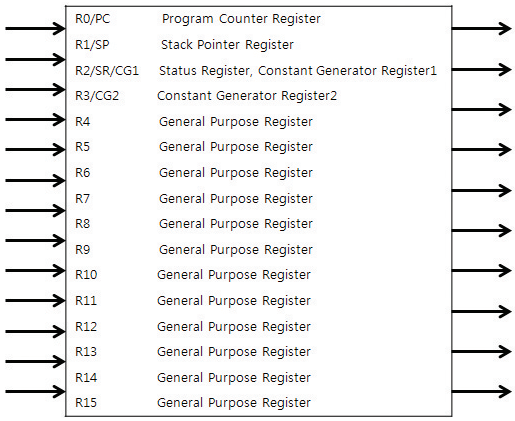
- 특별한 명령이 필요하지 않다.
- 추가적인 코드 워드가 필요치 않다.
- 상수 추출을 위한 메모리 엑세스가 필요하지 않다.

R4~R15는 일반적인 목적으로 사용되는 레지스터로서 데이터 레지스터, 어드레스 포인터, 인덱스 값으로 바이트 또는 워드 명령어를 통하여 접근 가능하다.

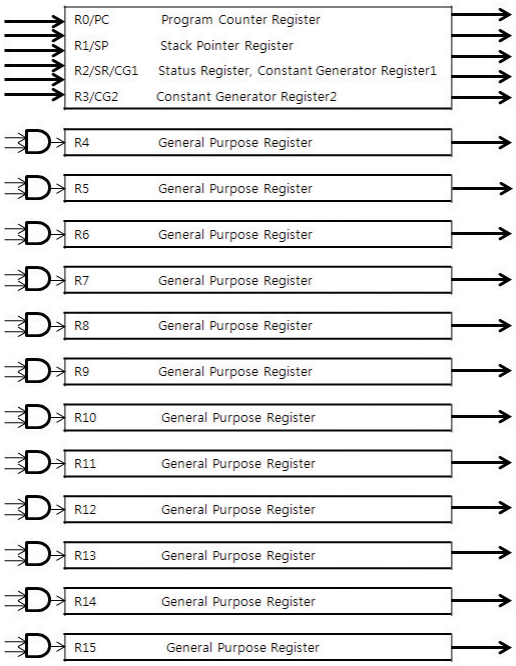
〈표 4〉 상수 발생 레지스터 CG1, 2의 값[1]

레지스터	As	상 수	비 고
R2	00	----	레지스터 모드
R2	01	(0)	절대 어드레스 모드
R2	10	00004h	+4, 비트 프로세싱
R2	11	00008h	+8, 비트 프로세싱
R3	00	00000h	0, 워드 프로세싱
R3	01	00001h	+1
R3	10	00002h	+2, 비트 프로세싱
R3	11	0FFFFh	-1, 워드 프로세싱

MSP430의 레지스터 파일의 블록 다이어그램은 다음 〈그림 4〉와 같다. 레지스터 파일 16개가 하나의 레지스터 파일 로 구성되어 있고 다수의 입력신호와 출력신호를 가지게 된다. Signal Gating은 각각의 레지스터 파일의 신호를 차단하는 방법이므로 이를 적용하기 위해 레지스터를 종류에 따라 분리, 변경하였다. 변경된 레지스터 파일 구조는 〈그림 5〉에 나와 있다. R4~R15까지의 General Purpose Register는 범용



〈그림 4〉 openMSP430 레지스터 파일 구조



〈그림 5〉 제안된 레지스터 파일 구조

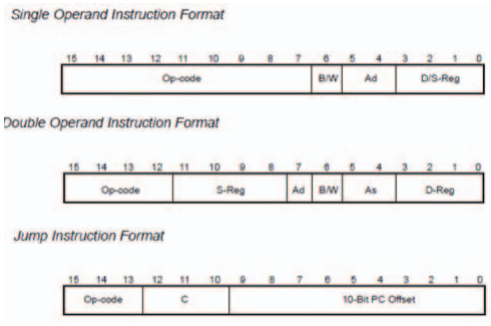
레지스터로서 프로그램 또는 환경에 따라 사용되어지지 않는 레지스터가 있으므로 General Purpose Register에만 Signal Gating을 적용하였다. 이를 위해 각각의 레지스터 파일을 따로 분리해 총 13개의 레지스터 파일을 만들었다. R0~R3까지의 Special Purpose Register는 특정한 기능을 가지는 레지스터로서 시스템의 동작을 위하여 Signal Gating을 적용하지 않고 이 4개의 레지스터를 따로 분리했다. 이와 같은 레지스터 파일 구조에 AND 게이트를 R4~R15까지의 12개 레지스터 파일의 입력신호에 추가하여 Signal Gating을 구현 하였다. 이러한 구조의 이점은 사용되지 않는 레지스터의 경우 입력신호가 완전히 차단되기 때문에 불필요한 스위칭 전력 소비를 줄일 수 있고 각각의 레지스터 파일이 독립적으로 신호를 받으므로 실제 시스템 동작에 문제 없이 실행 될 수 있다.

어드레싱 모드는 메모리에 저장된 데이터의 주소를 표시하는 방법이다.

4. 명령어 추가

openMSP430의 ISA(Instruction Set Architecture)는 27개의 코어 명령어와 24개의 추가 명령어 즉 총 51개의 명령어로 이루어져 있다. 코어 명령어는 CPU에 의하여 고유하게 해석되는 동작 코드이며, 추가 명령어는 자체 동작 코드를 가지지는 않지만 코드를 읽고 쓰는데 편리하도록 만들기 위하여 어셈블러에 의하여 같은 기능을 하는 코어 명령어로 자동으로 대치 되는 명령어를 의미한다. 각각의 명령어들은 〈그림 6〉에 나타낸 것과 같이 3개의 명령어 포맷으로 구분된다[4].

Single Operand Instruction의 경우 15~13비트는 Single Operand Instruction Format을 표시하는데 사용되고 12~7비트는 명령어를 결정한다. 6비트는 bit/word 단위, 5~4비트는 어드레싱 모드를 결정한다. 마지막으로 3~0비트는 Destination/Source Register를 결정한다. Double Operand Instruction의 15~12비트는 Double Operand Instruction Format 및 명령어를 결정하고 11~8비트는 Source Register



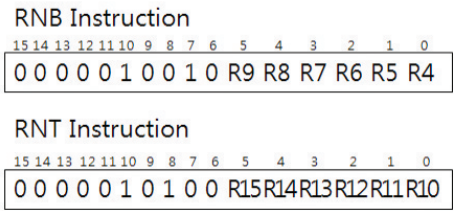
〈그림 6〉 openMSP430 명령어 포맷[2]

7, 5~4비트는 어드레싱 모드, 6비트는 bit/word, 단위 3~0비트는 Destination Register를 결정한다. Jump Instruction Format은 15~13비트가 Jump Instruction Format을 12~10비트가 명령어를 그리고 9~0비트가 PC Offset 값을 나타낸다.

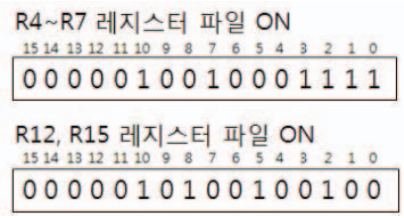
openMSP430은 소스 오퍼랜드용으로 7개의 어드레싱 모드가 있고, 타겟 오퍼랜드용으로 4개의 어드레싱 모드가 있다. 이들 어드레싱 모드는 전체 메모리 공간을 예외없이 접근할 수 있다. 〈표 5〉는 소스/타겟 오퍼랜드 어드레싱 모드를 나타낸다. As는 Addressing

〈표 5〉 어드레싱 모드의 종류[1]

As/Ad	어드레싱 모드	문 법	설 명
00/0	레지스터 모드	Rn	레지스터 내용이 오퍼랜드가 된다.
01/1	인덱스 모드	X(Rn)	(Rn+X)가 오퍼랜드를 가리킨다. X는 그 다음 워드에 저장된다.
01/1	심볼릭 모드	ADDR	(PC+X)가 오퍼랜드를 가리킨다. 인덱스모드로 X(PC)가 사용된다.
01/1	절대 모드	&ADDR	이 명령어 뒤에 오는 워드는 절대 어드레스를 의미한다.
10/–	간접 레지스터 모드	@Rn	Rn이 오퍼랜드를 가리키는 포인터로 사용된다.
11/–	간접 자동증가 모드	@Rn+	Rn이 오퍼랜드를 가리키는 포인터로 사용된다. Rn은 바이트 명령의 경우는 1씩, 워드 명령의 경우는 2씩 증가한다.
11/–	즉시 모드	#N	이 명령어 뒤에 오는 워드는 상수 N을 의미한다.



〈그림 7〉 RNB, RNT 명령어 형식



〈그림 8〉 RNB, RNT 명령어의 예

Source 이고, Ad는 Addressing Destination이다[5].

openMSP430의 레지스터 파일은 총 16개로 이 중 R0~R3는 Special Purpose Register로서 시스템 동작에 관하여 밀접한 연관을 가지고 있어 Signal Gating을 사용하지 않는다. R4~R15는 General Purpose Register로서 범용적인 용도에 사용되는 이 12개의 레지스터 파일에 Signal Gating을 사용했다.

Signal Gating을 실행하기 위하여 레지스터의 enable 신호를 보낼 수 있도록 새로운 명령어의 추가를 필요로 한다. 〈그림 6〉에 보는바와 같이 Single Operand Instruction의 Op-code 총 8비트로서 이중 Single Operand Instruction Format을 표시하는데 필요한 15~13 비트 값을 제외한 12~7비트 까지가 명령어의 종류를 나타내는데 사용된다. 즉 Single Operand Instruction은 총 64개의 명령어를 가질 수 있으나 openMSP430에서는 8개의 Single Operand Instruction 가지고 있어 56개의 명령어를 추가할 수 있다. 이 중 RNT(Register Number Top), RNB(Register Number Bottom) 두 개의 명령어를 추가하여 각각의 레지스터 파일의 신호를 차단 및 허용을 하도록 했다. RNT, RNB 명령어의 형식은 〈그림 7〉에 나와 있다.

Op-code에서 10~7비트 까지는 RNT, RNB 명령어를 선택하도록 했으며 5~0비트 까지는 각각의 레지스터파일의 Signal Gating의 enable 신호를 결정하는데 쓰이도록 했다. 이중 레지스터 파일 R4~R9 까지는 RNB명령어의 5~0비트가, 레지스터 파일 R10~R15까지는 RNT명령어의 5~0비트가 enable 신호다. 예를 들어 레지스터 파일 R4~R7까지의 레지스터 그리고 레지스터 파일 R12, R15가 사용되면 Op-code는 〈그림 8〉과 같이 된다. 이렇게 함으로써 각각의 레지스터 파일을 독립적으로 신호를 차단 및 허용할 수 있다.

5. 전력 측정 결과

openMSP430에 제안된 Signal Gating 구조를 Verilog HDL로 구현하여 시뮬레이션 및 테스트를 진행하였다. 시뮬레이션은 Mentor Graphics 사의 ModelSim을 이용하여 실행했으며, 클럭은 50Mhz, 동작 전압은 1.62V, MagnaChip사의 0.18um CMOS 공정으로 Synopsys사의 Design Compiler을 활용하여 합성하였다. ModelSim에서 시뮬레이션 수행 후 생성된 vcd(vector change dump)파일을 가지고 Synopsys사의 PrimeTime PX 모드에서 openMSP430의 소비 전력을 측정했다. 측정 결과는 〈표 6〉에 나와있다.

시뮬레이션은 General Purpose Register File의 사용 개수를 3개씩 증가 시키면서 수행했고 그에 따라서

〈표 6〉 Signal Gating을 적용한 MSP430 소비 전력

	Total power (mW)	Percent (%)	Dynamic power (mW)	Net Switching power (mW)	Cell internal power (mW)	Cell leakage power (mW)
Register Input signal 3	0.931	80	0.929	0.418	0.511	0.0018
Register Input signal 6	1.007	86	1.004	0.440	0.558	0.0018
Register Input signal 9	1.083	93	1.080	0.474	0.606	0.0018
Register Input signal 12	1.162	10	1.160	0.504	0.656	0.0018

관련 시뮬레이션 파형을 기초로 openMSP430 소비전력을 측정했다. General Purpose Register File을 3개만 사용했을 경우의 소비 전력은 0.93mW이고, 12개 전부 사용했을 경우의 소비 전력은 1.16mW가 나와서 약 20%정도의 전력 절감 효과가 있는 것으로 나타났다. 평균적으로 레지스터 파일 3개를 사용하지 않을 경우 약 7%의 전력 절감 효과가 나타나는 것을 알 수 있다.

Ⅲ. 향후 연구 및 결론

현재 레지스터 파일의 사용 개수에 따라 입력 신호를 차단하는 Signal Gating의 하드웨어 구현까지 완료가 되었다. 하지만 아직 Signal Gating을 사용하기 위해 어셈블러에 레지스터 파일을 선택하는 역할을 하는 명령어를 추가하는 작업은 아직 연구 중에 있다. 따라서 향후 연구에는 컴파일 단계에서 사용되지 않는 레지스터 파일을 검색하여 이 레지스터 파일의 Signal Gating을 실행시키기 위해 관련 명령어를 어셈블러에 추가하는 작업이 필요하다.

휴대용 기기의 확산으로 저전력 마이크로프로세서는 앞으로 그 중요성이 더욱 커질 전망이다. 본 논문에서는 현재 저전력 마이크로컨트롤러로 많이 각광 받고 있는 MSP430에 마이크로컨트롤러의 특성에 초점을 맞추어 상황에 따라 레지스터 파일의 개수를 선택할 수 있는 가변형 레지스터 파일 구조를 가진 마이크로컨트롤러를 설계하였다. 이를 바탕으로 기존의 저전력 마이크로프로세서에서 추가적으로 전력 절감 효과를 달성할 수 있었고 기존의 고정적인 하드웨어 구조에서 환경과 용도에 따라 최적화된 하드웨어 구조를 제공할 수 있도록 하였다.

참 고 문 헌

[1] http://www.ti.com

[2] http://opencores.org

[3] David A. Patterson, John L. Hennessy, "Computer Organization & Design", Elsevier Publishing Company, 2012

[4] 이정환, 김경호, "마이크로컨트롤러 MSP430의 활용", 문운당, 2012.

[5] 이정욱 외, "MSP430 마이크로프로세서", 그린, 2010.

고 규 영

2011년 2월 전북대학교 전자공학과 학사
2012년 3월~현재 전북대학교 전자정보공학부 석사과정

〈관심분야〉
프로세서 설계, SoC (System-on-Chip), DSP (Digital Signal Processing)

이 종 열

1993년 2月 한국과학기술원 공학사 (전자공학)
1996년 2月 한국과학기술원 공학석사 (전자공학)
2002년 8月 한국과학기술원 공학박사 (전자공학)
2002년 9월~2002년 9월 하이닉스반도체 선임연구원
2003년 10월~2004년 2월 한국과학기술원 BK 초빙교수
2004년 3월~2006년 3월 전북대학교 전임강사
2006년 4월~2010년 3월 전북대학교 조교수
2010년 4월~현재 전북대학교 부교수

〈관심분야〉
SoC 설계 방법론, 컴파일러, Digital Hardware