論文2002-39SD-2-8

저전력 8-비트 마이크로콘트롤러의 설계

(A Design of Low-Power 8-bit Microcontroller)

李 相 宰 *, 丁 恒 根 **

(Sang-Jae Lee and Hang-Geun Jeong)

요 약

본 논문에서는 저전력 8-비트 RISC 마이크로콘트롤러 구조를 제안하였다. 설계된 마이크로콘트롤러는 4 단계 파이프라인 구조를 가지며 기존의 여러 가지 저전력 설계 기법들을 이용하여 구현되었다. 전력 소모는 0.6 μm 공정을 사용했을 때 MIPS당 600 μW 를 소모했으며 0.25 μm 공정을 사용했을 때 MIPS당 70 μW 를 소모했다. RTL 레벨의 설계는 VHDL을 이용해서 수행되었고, 0.6 μm/0.25 μm CMOS IDEC(Integrated Circuit Design Education Center) standard cell library를 이용해서 게이트 레벨에서 기능 검증을 하였다. 합성된 코어는 0.25 μm 공정을 사용했을 때 약 7000개의 NAND 게이트를 0.36 mm^2의 작은 면적에 집적화 시킬 수 있었다. 마지막으로 기존의 상용 마이크로콘트롤러와의 성능 비교를 수행하였다.

Abstract

This paper suggests a 8-bit RISC microcontroller, which has a 4-stage pipeline architecture. Many low-power design techniques that have been proposed by previous works are adopted into it. The proposed microcontroller consumes only 600 μW per MIPS for 0.6 μm CMOS process and even lower power of 70 μW per MIPS for 0.25 μm process. The RTL level design of this microcontroller is carried out using VHDL. The functional verification is thoroughly done at the gate level using 0.6 μm/0.25 μm CMOS IDEC standard cell library. This microcontroller contains 7000 NAND gates on a 0.36 mm^2 die using 0.25 μm process. Finally the comparison of power consumption with other conventional microcontrollers is provided.

I. 서 론

최근에 들어 휴대용 정보기기의 급속한 발달에 힘입어 이에 필수적으로 내장되는 마이크로콘트롤러 설계 기술 또한 점점 더 중요해 지고 있다. 이와 더불어 VLSI 기술이 발전함에 따라 마이크로콘트롤러의 설계에 있어서는 저전력, 고성능, SOC(system on a chip)의 세 가지 항목을 적절히 고려하여 설계해야 할 필요성이 대두되고 있다.

휴대용 기기에 있어서 시스템 전체의 전력 소모에 대해서 마이크로콘트롤러의 소비 전력은 많은 부분을 차지하고 있다. 특히 시스템이 비동작 모드(sleep or idle)에 있을 때는 마이크로콘트롤러가 소모하는 전력이 지배적이다. 때문에 마이크로콘트롤러를 내장한 SOC 설계에 있어서 저전력 설계 기법을 이용한 마이크로콘

* 正會員, 韓國電子通信研究員
(Electronics and Telecommunications Research Institute)
** 正會員, 全北大學校 電子情報工學部
(Chonbuk National University, Division of Electronics and Information)
※ 본 연구는 IDEC의 CAD tool 지원으로 이루어졌음.
接受日字:2001年8月24日, 수정완료일:2002年1月10日

트롤러의 설계 방법은 점점 더 중요하게 고려되고 있다[1].

이에 본 논문에서는 휴대형 기기에 적합한 저전력 8-비트 RISC 마이크로콘트롤러를 설계하였다.

명령어 구조에 있어서는 8-비트 마이크로콘트롤러에서 최적인 레지스터-메모리 구조를 사용하였다[2]. 그리고 일반적인 RISC 구조에서 사용하는 명령어 길이의 절반인 16비트 명령어를 사용함으로써 마이크로콘트롤러에서 가장 많은 전력을 소모하게 되는 메모리 접근시의 전력 소모를 줄일 수 있는 방향으로 설계하였다. 그리고 2가지의 파워 다운 모드와 파워 콘트롤 레지스터를 이용하여 콘트롤러 내부 클럭을 조절함으로써 사용자 수준에서 전력 소모를 줄일 수 있는 방법을 제공하였다.

마이크로아키텍쳐 설계에서는 4단계의 파이프라인 구조를 채택하고 효과적인 데이터 포워딩 구조와 브랜치 유닛을 설계하여 CPI를 1에 가깝게 유지할 수 있도록 하였다. 결과적으로 높은 CPI와 low throughput을 가지는 마이크로콘트롤러에 비해서 저전력으로 동작할 수 있음을 보였다.

회로적으로는 클럭과 시그널 게이팅(signal gating)기법을 사용하여 불필요한 천이를 제거하여 스위칭 커패시턴스에 의한 전력 소모를 줄일 수 있었다[1]. 그리고 선택적인 입력 래칭(latching) 기법을 사용하여 파이프라인 레지스터 사이에 위치해 있는 조합회로의 변화를 선택적으로 막아 줌으로 해서 이로 인해 발생하는 전력 소모를 최소한으로 줄일 수 있었다[1].

마지막으로 서로 다른 동작 전압을 가지는 표준 셀을 사용하여 합성을 수행 해 봄으로써 동작 전압의 변화에 따른 전력 소모의 변화를 비교해 보았다.

Ⅱ. 저전력 설계 기법

1. 저전력 디지털 CMOS회로의 설계

CMOS는 static 전력 소모가 적고 집적도가 높아 저전력 시스템에서 오늘날 거의 대부분 사용되어지고 있는 기술이다. 디지털 CMOS 회로에서의 전력 소모는 간단한 방정식으로 유도할 수 있는데 다음과 같이 3가지의 주요한 항목으로 나눌 수 있다[3].

$$P_{tot} = \alpha C_L V V_{DD} f_{clk} + I_{sc} V_{DD} + I_{leakage} V_{DD} \quad (1)$$

이 중에서 전력 소모의 가장 큰 부분을 차지하는 항이 첫 번째 스위칭 파워 부분이고 나머지 short circuit 전류와 leakage 전류 항은 적은 부분을 차지한다. 결과적으로 식 (1)은 다음과 같이 간략화 되며 저전력 설계는 같은 성능을 유지하면서 $\alpha, C_L, V_{DD}, f_{clk}$ 을 줄이는 문제로 축약된다.

$$P_{tot} = \alpha C_L V_{DD}^2 f_{clk} \quad (2)$$

본 설계에서는 이 중에서 두 가지의 서로 다른 공정을 사용해서 V_{DD} 및 C_L 의 감소가 전체 전력 소모에 미치는 영향을 살펴보았다.

2. 메모리 접근으로 인한 전력 소모 감소 방안

마이크로콘트롤러 서브시스템에서 대부분의 전력 소모는 콘트롤러 자체보다는 메모리와 그 인터페이스에서 발생한다[4]. 외부 메모리 접근 시에는 일단 어드레스가 PCB상의 어드레스 라인을 통해 메모리에 전달되어야 하고, 메모리 자체에서도 전력 소모가 있으며, 원하는 데이터가 다시 PCB상의 데이터 라인을 따라 돌아오게 되는데, 이러한 칩 외부의 데이터패스상의 커패시턴스는 칩 내부의 커패시턴스보다 훨씬 큰 수~수십 nF에 이르게 된다[5]. 때문에 식 (2)에서 C_L 값이 매우 커지게 되므로 전력 소모가 심하게 된다. 본 논문에서는 16레벨의 하드웨어 스택을 설계해서 call이나 exception 발생 시에 복귀 주소를 이곳에 저장시켜 외부 메모리 접근을 방지함으로써 전력 소모를 줄일 수 있도록 했다[6]. 그리고 일반적인 RISC 명령어 길이의 절반인 16bit 길이의 명령어를 사용함으로써 외부 프로그램 메모리 접근시의 전력소모를 감소할 수 있도록 하였다.

3. 파워다운 모드의 사용

일반적으로 휴대용 임베디드 시스템 동작 시에는 전체 시간을 계속해서 동작하기보다는 어떤 시간대에서는 동작을 하고 다른 시간대에서는 동작을 하지 않고 쉬고 있는 경우가 많다. 설계된 저전력 마이크로콘트롤러 시스템에서는 이런 시간 동안에 전력 소모를 최소화하기 위해 그림 1과 같이 다양한 파워다운 모드와 시스템 클럭 조정 기능을 제공한다. 파워다운 모드에서는 주로 시스템 전체나 혹은 코어 내부로 들어가는 클럭 주파수를 막아버리는 방법을 사용한다. 그리고 클럭 제어 레지스터를 조정해서 코어 내부로 들어가는 클럭

주파수를 변경함으로써 정상 동작을 할 경우보다 훨씬 적은 전력만을 소모하도록 하였다.

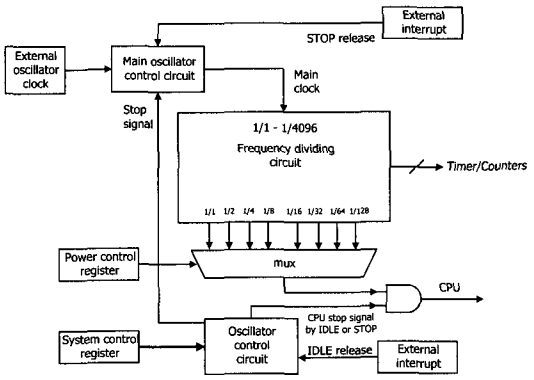


그림 1. 시스템 클럭 제어 블록도
Fig. 1. System clock control block diagram.

4. 클럭 게이팅

또 다른 저전력 설계 기법 중의 하나가 클럭 게이팅 방법이다[1]. 명령어의 종류에 따라서 수행하는 작업이 다르기 때문에 동작을 하는 블록 또한 다르다. 이 때 동작을 할 필요가 없는 블록으로 공급되는 클럭을 막아버린다면 레지스터 값의 변화를 막을 수 있고 그에 따른 조합회로들의 변화 역시 막을 수 있다. 이 방법은 설계 전반에 걸쳐서 사용하였고 작은 크기의 로직에서부터 칩 전체에 이르기까지 적용할 수 있었다. 클럭은 최상위의 클럭으로부터 재생성 되어서 각 블록으로 적절한 타이밍에 맞춰 공급되는데 이러한 동적 클럭 관리를 하는 부분은 대부분 명령어의 수행을 결정짓는 부분과도 관련이 있기 때문에 면적이나 복잡도를 크게 증가시키지 않고서도 효율적으로 설계가 가능하였다.

Ⅲ. 명령어 구조

일반적으로 명령어 구조에는 메모리-메모리 구조, 메모리-레지스터 구조, 로드/스토어 구조의 3가지가 있다[2]. 먼저 메모리-메모리 구조는 메모리로부터 오퍼런드를 읽어와서 연산을 한 뒤, 그 결과 값을 메모리에 저장하는 동작을 단일 명령어로 처리할 수 있는 구조이고, 레지스터-메모리 구조는 메모리로부터 오퍼런드를 읽어와서 연산을 하는 동작을 하나의 명령어로 처리하고 메모리에 저장하는 명령을 별도로 수행하는 구조이며, 마지막으로 로드/스토어 구조는 연산에서 오직 레지스터와 레지스터만 사용 가능하고 메모리 접근은 로드와 스토어로만 할 수 있는 구조이다.

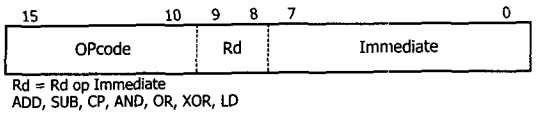
실행되는 명령어의 수는 메모리-메모리 구조가 가장 적고 전통적으로 RISC구조에서 많이 쓰이는 로드/스토어 구조가 가장 많은 명령어를 사용한다. 그러나 단순한 실행 명령어 수뿐만 아니라 CPI(clock cycles per instruction)와 최대 클록 주파수를 고려해야 할 필요가 있다.

일반적인 8비트 마이크로콘트롤러는 1~10 MIPS (million instructions per second)의 성능을 갖는다[2]. CPI는 4~20 정도이므로 1 MIPS의 성능을 얻기 위해서는 4~20 MHz의 클럭이 필요하고 10 MIPS의 성능을 얻기 위해서는 훨씬 빠른 클럭이 필요하게 되며 그에 따른 전력 소모 역시 증가하게 된다. 하지만 CPI가 1이라면 1~10 MIPS의 성능을 얻기 위해 1~10 MHz의 클럭 만을 필요로 하게 된다. 이 정도의 주파수에서는 많은 수의 파이프라인 단계를 사용함으로써 제어가 복잡하고 높은 동작 주파수를 필요로 하는 RISC구조를 사용할 필요가 없이 간단한 파이프라인 구조를 갖는 RISC 구조로도 충분하다. 또한 CPI를 1로 유지하기 위해서 한 명령어 당 두 번의 메모리 접근이 필요한 메모리-메모리 구조의 사용도 적당하지 않다. 즉 8비트 마이크로콘트롤러의 명령어 구조는 레지스터-메모리 구조가 가장 적당함을 알 수 있다.

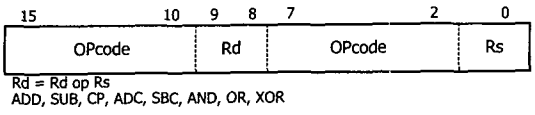
설계된 마이크로콘트롤러는 레지스터-메모리 구조의 간단한 명령어 구조를 갖으며 모든 명령어의 길이는 같다. 그림 2는 명령어 구조의 일부를 나타낸다.

대부분의 명령어에서 Rd의 위치는 고정되어 있어 디코딩을 쉽게 할 수 있도록 하였다. 2개의 오퍼런드를 갖는 명령어는 Rd를 가지고 연산을 수행한 뒤 다시 Rd에 결과 값을 저장하도록 함으로써 효율적으로 명령어를 이용할 수 있으며, 1개의 오퍼런드를 갖는 ALU 명령어는 로직 연산이나 쉬프트 연산에 사용되며 ALUop를 별도로 이용해서 다양한 ALU연산을 수행할 수 있다. 분기 명령은 조건부 분기와 무조건 분기 명령을 모두 지원하도록 하였다.

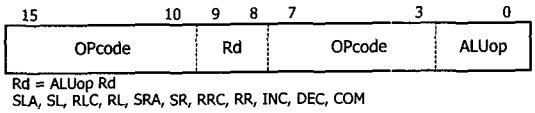
메모리 접근은 직접 주소 지정 방식과 인덱스 주소 지정 방식을 사용한다. 레지스터-메모리 구조를 사용하기 때문에 메모리 접근은 로드나 스토어 명령어로 제한하지 않고 일반적인 연산 명령어들도 메모리로부터 오퍼런드를 읽어와 연산이 가능하다. 단 메모리 쓰기는 스토어 명령으로만 가능하다.



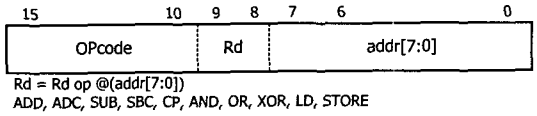
(a) Immediate 오퍼런드를 갖는 명령어
(a) Immediate operand instructions.



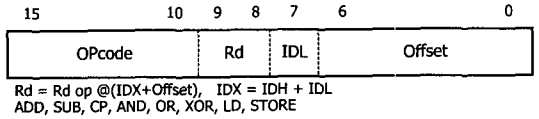
(b) 2개의 오퍼런드를 갖는 명령어
(b) 2 operand instructions.



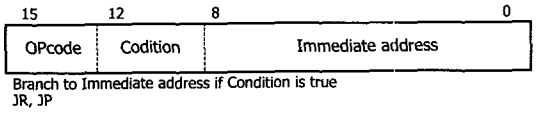
(c) 1개의 오퍼런드를 갖는 ALU 명령어
(c) 1 operand ALU instructions.



(d) 직접 메모리 방식의 명령어
(d) Direct memory access instructions.



(e) 인덱스 메모리 방식의 명령어
(e) Index memory access instructions.



(f) 조건부 브랜치 명령어
(f) Conditional branch instructions.

그림 2. 명령어 구조
Fig. 2. Instruction architecture.

직접 메모리 주소는 상위 인덱스 레지스터 IDH와 명령어 디코드 블록으로부터 추출된 addr[7:0]를 결합해서 생성한다. 반면 간접 메모리 주소는 IDH와 IDL0나 IDL1에 offset[4:0]을 더한 값을 결합해서 생성한다.

저전력 동작을 위한 명령어로 내부의 클럭을 조정하는 명령이 있다. 소프트웨어적으로 CPU 내부로 입력되는 클럭을 1 - 126배로 분주할 수 있다. IDLE 명령은 CPU 내부로 입력되는 클럭을 막지만 주변 장치들로 입력되는 클럭은 생성이 된다. STOP 명령은 주변 장치로 입력되는 클럭 또한 막아 버리는 역할을 한다. 콘트롤러가 일단 STOP이나 IDLE 모드로 들어가게 되면 인터럽트가 발생할 때까지 이 상태를 계속 유지함으로써 저전력으로 동작할 수 있다.

Ⅳ. 인스트럭션 파이프라인의 설계

1. 파이프라인의 구조

일반적인 RISC 파이프라인은 IF(instruction fetch), ID(instruction decoding), EX(execution), MEM (memory access), WB(write back)의 5단계를 갖으며 대부분의 명령어가 한 사이클에 실행 되도록 한다[6].

표 1. 파이프라인 동작 흐름도
Table 1. Pipeline operation flow.

Stage	Operations
IF	Instruction address bus ⇐ PC Instruction register ⇐ Instruction bus
ID / MEM	Instruction decoding, bypass condition check Register file read (second phase of the clock) PC ⇐ displacement address, exception vector Data address bus ⇐ result register Data bus ⇐ store register, stack value Stack ⇐ result register
EX	ALU operation Result register ⇐ ALU result
WB	Register file write (first phase of the clock)

저전력 마이크로콘트롤러는 보통 이 5단계 파이프라인이나 그 이하의 파이프라인을 사용하게 된다.

설계된 마이크로콘트롤러는 10 MHz이하에서 동작했을 때 10MIPS 정도의 성능을 요구하도록 설계되었기 때문에 5단계 이상의 깊은 파이프라인 구조를 사용할 필요가 없다. 때문에 일반적으로 사용하는 5단계 파이프라인에서 한 단계 줄인 4단계 파이프라인을 사용함으로써 사용되는 하드웨어를 줄여서 전력 감소에 도움을 주도록 하였다. 즉 ID단계와 MEM단계를 결합함으로써 4단계 파이프라인을 구성했으며 전체적인 파이프

라인의 흐름은 표 1에 나와있다.

2. Hazard의 방지

일반적인 RISC 구조에서는 MEM 단계가 4번째 세그먼트에 있기 때문에 그 다음 명령의 EX 단계가 시간적으로 앞서 있게 되어 그림 3과 같이 hazard가 발생하게 된다. 그러나 그림 4에서 볼 수 있듯이 설계된 구조에서는 MEM 단계가 ID 단계와 같은 2번째 세그먼트에 있으므로 그 다음 EX 단계에서 이 데이터를 사용할 수가 있게 되어 hazard를 방지할 수가 있다.

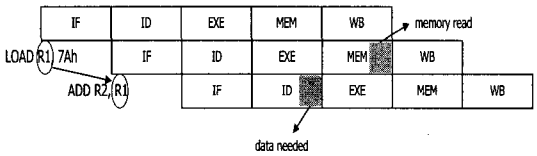


그림 3. 일반적인 RISC에서의 Load hazard
Fig. 3. Load hazard in conventional RISC.

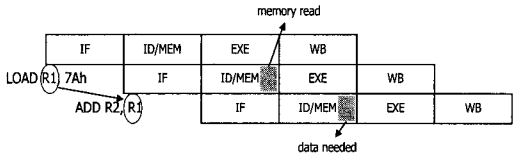


그림 4. 설계된 RISC에서의 load hazard 해결
Fig. 4. Solving load hazard in proposed RISC.

또 다른 데이터 hazard 문제로는 아직 WB 단계가 끝나지 않은 데이터를 다음 명령에서 source 오퍼런드로 사용하고자 할 때 그림 5에서처럼hazard가 발생한다.

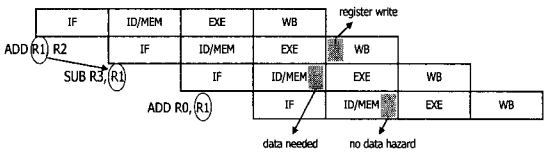


그림 5. 설계된 RISC에서의 data hazard 해결
Fig. 5. Solving data hazard in proposed RISC.

설계된 콘트롤러에서는 이 문제를 해결하기 위해 인터널 포워딩 방법을 사용한다[6]. Hazard를 검출할 수 있는 블록을 설계해서 ID 단계에 있는 명령의 source 오퍼런드 중의 하나가 EX 단계에 있는 명령의 destination 오퍼런드와 같은 지를 검사한다. 만약 같다면 다음 EX 단계의 source 오퍼런드가 bypass bus를 통해서 포워딩된다.

그리고 레지스터 쓰기는 WB 단계의 첫 번째 phase에서 수행되고 레지스터 읽기는 두 번째 phase에서 수행되기 때문에 데이터 hazard는 발생하지 않는다. 그림 6는 인터널 포워딩에 대한 블록도이다.

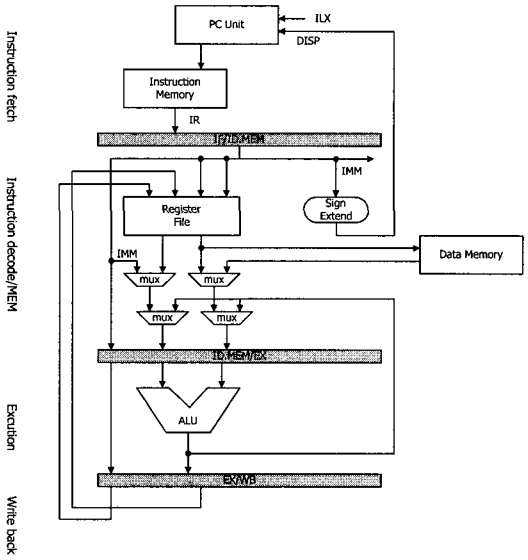


그림 6. 인터널 포워딩 블록도
Fig. 6. Internal forwarding block diagram.

일반적인 파이프라인에 의해 생기는 또 다른 문제인 콘트롤 hazard는 콘트롤 플로우 명령인 브랜치나 점프 명령에서 점프할 주소의 계산이 EX 단계의 끝에 가서야 결정되기 때문에 일어나는 문제이다. 브랜치 명령과 target 번지의 명령어 사이에 브랜치 슬롯이 생기게 되면 여기에는 no operation이나 optimizing compiler에 의해 적절한 명령을 채우게 된다. 하지만 채울 수 있는 적절한 명령어가 없다면 브랜치 슬롯은 no operation으로 채울 수밖에 없고, 그렇게 되면 명령어 사이클의 손실이 발생하게 된다.

설계된 마이크로콘트롤러는 이러한 콘트롤 hazard를 없애기 위해 별도로 ID단계에서 주소를 생성할 수 있는 블록을 설계했다. 즉 콘디션 코드가 EX 단계에서 생성이 되면 branch나 jump명령의 ID 단계에서 그 콘디션 코드를 참조해서 만약 브랜치가 발생해야 한다면 target 주소가 첫 번째 phase에서 생성이 된다. 그러면 그 주소를 이용해서 그 다음 IF 단계의 두 번째 phase에서 그 주소를 이용해 target 명령어를 가져오게 된다. 만약 브랜치가 발생하지 않아도 되는 경우라면 정상적

으로 증가되어 있는 PC(program counter)의 값을 이용해 명령어를 가져오게 된다. 이러한 분기 주소 생성에 대한 타이밍도가 그림 7에 나와있다.

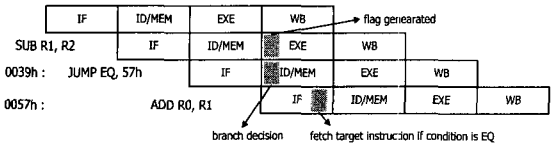


그림 7. 분기 주소의 생성
Fig. 7. Branch target address generation.

3. 파이프라인에서의 Exception처리

파이프라인 설계가 어려운 이유 중의 하나는 exception을 처리하기가 복잡하기 때문이다. 파이프라인에서는 명령어가 중첩되어 실행되기 때문에 명령어가 안전하게 콘트롤러의 상태를 바꾸고 새로운 작업을 수행하도록 하는 어려운 문제가 있다. 또한 명령어들은 여러 개의 세그먼트로 분리되어 실행되고 여러 클럭 사이클 동안 수행되며 또 다른 명령어가 exception을 유발시킬 수 있는 가능성도 있다. exception이 발생하면 파이프라인이 새로 채워지게 된다. 파이프라인 콘트롤러는 이 때 파이프라인 안에 있던 무의미한 데이터에 의해 부적절한 동작을 하는 것을 막는 역할을 한다. 그림 8에서 2번 명령어가 MEM 단계에 있을 때 exception이 발생해서 interrupt vector 주소로 뛰었다고 가정할 때 빗금 쳐 있는 부분에 있는 명령은 무의미한 명령어가 되며 이 명령어에 의해서 부적절한 동작을 하지 않게 설계되었다. 즉 이 명령들에 의해서 콘트롤 플로우나, 메모리 쓰기, 또는 레지스터 쓰기가 일어나지 않으며, 또 이 명령들과의 data dependency에 의해서 인터널 포워딩도 발생하지 않는다.

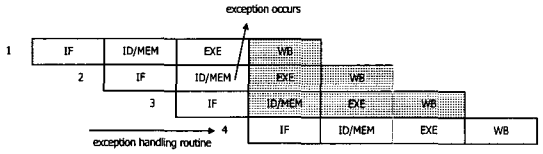


그림 8. 파이프라인 내에 있는 명령의 무효화
Fig. 8. Squashing instructions within pipeline.

V. 기능 블록의 설계

일반적으로 실시간 시스템에서는 주어진 시간 안에 정해진 태스크를 수행해야 하는데, CPI가 높아서 정해진 시간 안에 작업을 완료할 수 없을 경우에는 클럭 주파수를 높여야 하며, 전력소모는 늘어나게 된다. 하지만 CPI가 낮다면 정해진 시간 안에 같은 태스크를 수행하는 데 걸리는 시간이 상대적으로 짧게 되며, 보다 적은 전력을 소모하게 된다. 설계된 마이크로콘트롤러는 실시간 시스템에 적합한 응용분야에서 사용될 수 있도록 낮은 CPI를 유지할 수 있는 구조로 구현되었다.

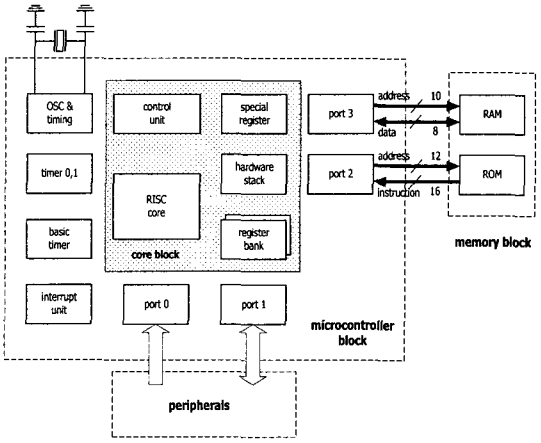


그림 9. 전체 블록도
Fig. 9. Top level block diagram.

그림 9는 설계된 마이크로콘트롤러의 전체 구성도를 나타낸다. 메모리 구조는 데이터 메모리와 프로그램 메모리가 분리된 Harvard 구조를 사용했다. 명령어는 약 40여 개의 명령어로 이루어졌으며, 모든 명령어는 1 clock cycle에 수행이 가능하도록 설계되었다. 이렇게 낮은 CPI 때문에 일반적으로 높은 CPI를 가지는 마이크로콘트롤러에 비해서 주어진 태스크에 대해 적은 수의 클럭만을 사용하게 된다.

코어 블록은 데이터패스 부분과 제어 블록, 범용 레지스터, 스페셜 레지스터, 16레벨의 하드웨어 스택으로 구성된다. 그리고 타이머나 인터럽트 콘트롤러와 같은 외부 블록을 별도로 설계하였다.

VI. 합성결과 및 성능분석

지금까지 살펴 본 각각의 기능 블록들은 VHDL을 이용해서 RTL 레벨로 설계되고 검증되었다. 그리고 정확한 타이밍 검증을 위해서IDEC에서 제공하는 IDEC C-631 표준 셀 라이브러리와 IDEC C-221 표준 셀 라이

브러리를 이용해서 게이트 레벨 시뮬레이션을 수행하였다[7-8].

시뮬레이션을 통하여 기능이 검증된 각각의 기능 블록들을 다시 표준 셀 라이브러리를 이용하여 합성하였다. 사용된 라이브러리는 각각의 기본 셀에 대한 입력 커패시턴스 정보를 가지고 있기 때문에 이를 이용하여 다이나믹 전력 소모를 구할 수 있다. 이미 나와있는 다른 8비트 마이크로콘트롤러와의 비교를 위해서 코어부분의 기능 블록에 대한 전력소모를 구한 값이 표 2에 나와있다.

0.6 μm/3.3V 공정을 이용했을 때 전체적으로 약 0.64 mW 를 소모한 반면에 0.25 μm/2.5V 공정을 사용했을 때는 0.07 mW 의 적은 전력을 소모한 결과를 볼 수 있다. 0.25 μm 공정에 사용된 셀들은 동작전압 뿐만 아니라 로드 커패시턴스도 0.6 μm 공정에서 사용된 셀들보다 작기 때문에 전력 소모를 현저히 낮출 수 있었다.

표 2. 합성된 코어 블록의 전력 소모
Table 2. Power consumption of synthesized core block. - μW / MIPS.

공정 / 기능 블록	0.6 μm/ 3.3V 소모 전력	0.25 μm/ 2.5V 소모 전력
Data Mem. Add. generation	30.5	3.1
Program counter	101.0	11.6
Special purpose register	76.8	8.4
ALU	112.3	12.4
Control part	55.1	7.0
Register file	92.6	9.7
Stack	169.8	18.3
Total	638.1	70.4

또한 0.25 μm 공정에서는 0.6 μm 공정보다 다양한 셀들을 사용했는데 이로 인해서 같은 기능을 구현할 때에도 상대적으로 게이트수를 적게 사용할 수 있었다. 결과적으로 표 3에서 볼 수 있듯이 0.25 μm 공정을 사용했을 때 NAND게이트로 환산한 전체 게이트수에 있어서도 약 30% 줄어든 게이트만을 사용해서 합성이 가능한 것을 볼 수 있다.

전체적인 구조에 대한 전력소모를 이미 나와있는 마이크로콘트롤러와 비교한 결과가 그림 10에 나와있다[1,2]. 정확한 비교를 위해서 1 MIPS에 공급전압을 3.0V로 하였으나 설계된 마이크로콘트롤러가 사용한

표 3. 합성된 코어 블록의 게이트 수
Table 3. Gate counts of synthesized core block - 2-input NAND gate 기준

공정 / 기능 블록	0.6 μm 게이트 수	0.25 μm 게이트 수
Data Mem. Add. generation	260	188
Program counter	1118	770
Special purpose register	815	541
ALU	854	642
Control part	1465	1120
Register file	1200	856
Stack	4387	2892
Total	10100	7014

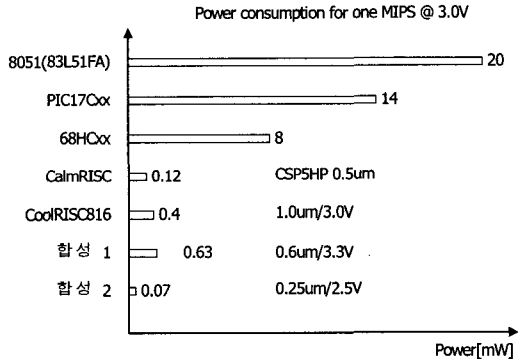


그림 10. 성능의 비교
Fig. 10. Comparison of performance..

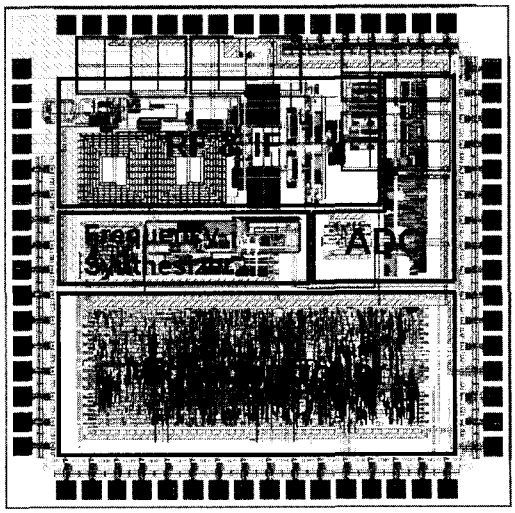


그림 11. 임베디드 core로 사용된 예
Fig. 11. Embedded core example.

셀 라이브러리가 3.3V와 2.5V를 각각 사용하기 때문에 성능비교에 있어서 차이는 있다. 파이프라인 구조를 사용하지 않은 8051은 상당히 큰 전력을 소모한 반면에 설계된 마이크로콘트롤러는 현저히 적은 전력을 소모하는 것으로 나타났다. 즉 공정뿐만 아니라 설계 구조면에서도 저전력으로 설계되었음을 알 수 있다.

그림 11은 0.25 μm 공정을 이용해 설계된 마이크로콘트롤러가 내장된 칩의 도면을 나타낸다. 설계된 칩은 RF&IF단, 아날로그-디지털 변환기, 주파수 합성기, 마이크로콘트롤러 부분으로 나뉜다. 마이크로콘트롤러는 주파수합성기를 제어하고 아날로그-디지털 변환기로부터 데이터를 입력받아 처리하는 역할을 담당한다. 약 0.36 mm^2의 작은 면적을 차지하면서 저전력 임베디드 마이크로콘트롤러로 사용되는 예를 보여주고 있다.

Ⅶ. 결 론

본 논문에서는 RISC 구조를 갖는 8비트 저전력 마이크로콘트롤러를 설계하였다. 일반적인 RISC 명령어 길이의 절반인 16비트 명령어를 채택함으로써 전력 소모의 감소에 도움이 되도록 하였다. 명령어 구조에 있어서는 레지스터-메모리 구조를 사용하여 필요한 명령어 집합을 선정하였다. 그리고 이러한 명령어 집합들을 기초로 하여 전력 소모와 코어의 면적을 최소로 하는 것을 목표로 각각의 기능 블록을 설계하였다.

저전력 설계를 위해 여러 가지 설계 기법들이 적용되었다. 외부 메모리 접근으로 인한 전력 소모를 감소하기 위해서 하드웨어 스택을 설계하였고 다양한 파워다운 모드를 통해서 저전력 동작이 가능하게 하였다. 그리고 시스템 전반에 걸쳐 클럭 게이팅과 시그널 게이팅 기법을 적용하였고 4단계 파이프라인 구조를 채택해서 CPI를 1에 가깝게 유지함으로써 저전력 마이크로콘트롤러를 구현할 수 있었다. 또한 3.3V와 2.5V의 공급전압 변화가 전력 소모에 미치는 영향을 알아보았다.

설계된 블록들은 submicron 표준 셀 라이브러리를 사용하여 semi-custom형식으로 합성되었다. 합성된 코어는 0.25 μm 공정을 사용하여 0.36 mm^2의 면적에 집적화 시킬 수 있었고, 약 70 μW 의 적은 전력을 소모하는 것으로 나타났다.

참 고 문 헌

[1] Kyoung-Mook Lim et al. "CalmRISCTM : A Low Power Microcontroller with Efficient Coprocessor Interface" Proc. of IEEE International Conference on Computer Design : VLSI in Computers and Processors, pp.299-302, 1999.

[2] C. Piguet et al. "Low-Power Design of 8-b Embedded CoolRisc Microcontroller Cores." IEEE J. Solid-State Circuits, vol. 32, pp. 1067-1077, July. 1997.

[3] A. P. Chandrakasan, S, Sheng, and R. W. Brodersen, "Low-Power CMOS Digital Design," IEEE J. Solid-State Circuits, vol. 27, pp. 473-484, Apr, 1992.

[4] A. P. Chandrakasan, A. Burstein, and R. W. Brodersen, "A Low Power Chipset for Portable Multimedia Applications," IEEE J. Solid-State Circuits, vol. 29, pp. 111-131, Dec, 1994.

[5] L.D. Smith, "Decoupling capacitor calculations for CMOS circuits", Electrical Performance of Electronic packaging, IEEE 3rd Topical Meeting on, pp. 101-105, 1994.

[6] J. L. Hennessy, D. A. Patterson, Computer Architecture : A Quantitative Approach, 2nd Edition. Morgan Kaufmann Publishers, San Mateo, CA, 1996.

[7] IDEC-C631. IC Design Education Center, 1998.

[8] IDEC Cell Library Data Book : IDEC-C221. IC Design Education Center, 2000.

저 자 소 개

李 相 宰(正會員)

1999년 : 전북대학교 전자공학과. 2001년 : 전북대학교 대학원. 2001년~현재 : ETRI 네트워크연구소 홈네트워킹팀 연구원. <관심분야> VLSI, Microprocessor, 홈네트워킹 기술

丁 恒 根(正會員) 第37卷 第2號 SC扁 參照

1977년 2월 : 서울대학교 전자공학과(공학사). 1979년 2월 : KAIST 전기 및 전자공학과(공학석사). 1989년 12월 : University of Florida 전기공학과(공학박사). 1979년~1982년 : ETRI 연구원. 1989년~1991년 : Motorola 연구원. 1991~현재 : 전북대학교 전자정보공학부 전임강사, 조교수, 부교수, 전북대학교부설 공학연구원 전자정보신기술연구센터 연구원. <주관심분야> RF IC 설계, 광연결용 고속 CMOS 집적회로 설계