論文2003-40SD-4-11

CISC 임베디드 컨트롤러를 위한 새로운 비동기 파이프라인 아키텍쳐, A8051

(A New Asynchronous Pipeline Architecture for CISC type Embedded Micro-Controller, A8051)

李 制 勳 *, 趙 慶 錄 **

(Je-Hoon Lee and Kyoung-Rok Cho)

요 약

비동기 설계 기법은 시스템 클럭을 사용하지 않고, 동작이 필요한 모듈만 활성화시켜 전력 및 성능면에서 동기식 설계 기법에 비해 높은 성능을 갖는다. 본 논문은 임베디드 컨트롤러인 Intel 80C51과 완전한 명령어 호환성을 갖고, 비동기식 파이프라인 구조로 최적화된 A8051 아키텍쳐를 제안한다. 다양한 어드레싱 모드와 명령어를 제공하는 CISC 명령어 수행 스킴은 동기식 파이프라인 구조에 적합하지 않고 많은 오버헤드를 유발한다. 본 논문에서는 명령어 실행 사이클을 비동기식 파이프라인 수행에 적합하도록 명령어별로 그룹화하고, 동기화 및 다중 실행 사이클로 인한 오버헤드로 발생된 버블을 제거함으로서 최적화하였다. 또한 적합한 분기 처리 기법 및 가변적인 명령어 길이의 처리 방법을 제시함으로서 명령어 수행시 필요한 상태 수를 최소화하고, 명령어 수행의 병렬성을 증가시켰다. 제안된 A8051 아키텍쳐는 Verilog HDL로 설계하여 0.35㎛ CMOS 공정 표준 셀 라이브러리로 합성하였다. 실험 결과로 A8051은 36MHz 클럭을 사용하는 인텔 80C51과 다른 비동기 80C51에 비해 약 24배의 성능 향상을 얻었다.

Abstract

The asynchronous design methods proved to have the higher performance in power consumption and execution speed than synchronous ones because it just needs to activate the required module without feeding clock in the system. Despite the advantage of CISC machine providing the variable addressing modes and instructions, its execution scheme is hardly suited for a synchronous pipeline architecture and incurs a lot of overhead. This paper proposes a novel asynchronous pipeline architecture, A8051, whose instruction set is fully compatible with that of Intel 80C51, an embedded micro-controller. We classify the instructions into the group keeping the same execution scheme for the asynchronous pipeline and optimize it eliminating the bubble stage that comes from the overhead of the multi-cycle execution. The new methodologies for branch and various instruction lengths are suggested to minimize the number of states required for instructions execution and to increase its parallelism. The proposed A8051 architecture is synthesized with 0.35㎛ CMOS standard cell library. The simulation results show higher speed than that of Intel 80C51 with 36 MHz and other asynchronous counterparts by 24 times.

Keywords : 혼성신호 테스트, BIST, 오실레이션, OTM

* 正會員, 忠北大學校 情報通信工學科 및 컴퓨터 情報通信硏究所

(Department of Computer & Communication Eng. Chungbuk national University)

** 正會員, 忠北大學校 電氣電子工學部

(School of Electrical & Computer Engineering. Chungbuk national University)

接受日字:2002年8月2日, 수정완료일:2003年4月4日

Ⅰ. 서 론

VLSI 기술 발달로 게이트 지연은 감소되나 배선 지연이 상대적으로 증가되어 클럭 신호의 동기화가 어려워지고 클럭 스큐등 많은 문제가 발생된다. 또 동작이 불필요한 모듈에도 클럭을 제공함으로서 시스템 전력의 낭비가 발생한다. 이를 극복하기 위해, 시스템 클럭을 사용하지 않는 비동기 회로 설계 기법에 관한 연구가 활발히 이루어지고 있다[1,2].

시스템 모듈내의 최대 지연 시간을 클럭으로 기준하여 전체 모듈을 동작시키는 동기식 시스템과 달리, 비동기 시스템은 각 모듈의 평균 수행 시간으로 연산이 수행되기에 동기화에 대한 오버헤드가 적어 고속 연산기에 적합하다. 따라서 저전력 비동기 시스템 설계 방법에 관한 연구와 상용 프로세서에 비동기 설계 기법을 적용한 새로운 아키텍쳐를 갖는 프로세서의 개발에 관한 연구가 진행되고 있다[3~7]. 고속 연산을 위해 대부분의 비동기 파이프라인 프로세서는 RISC 아키텍쳐를 채택하였다[4,5,8].

파이프라인 시스템은 각 스테이지의 제어가 외부에서 인가된 시스템 클럭에 의해서인지, 또는 각 스테이지 동작에 따른 이벤트 신호에 대한 핸드쉐이크 프로토콜에 의해서인지에 따라 동기식 및 비동기식 파이프라인으로 구분된다[8].

<그림 1>과 같이, 동기식 파이프라인 시스템의 모든 스테이지는 클럭에 의해 동시에 활성화되며, 클럭 주기는 각 스테이지의 최대 동작 시간으로 결정된다. 따라서 전체 파이프라인 시스템은 가장 긴 지연 시간을 갖는 스테이지에 의해 성능이 제한된다. 또한 동작이 불필요한 모듈에도 클럭이 공급됨으로서 전력 소모가 증가한다.

이에 반해, 비동기 파이프라인 시스템의 각 스테이지는 핸드쉐이크 프로토콜을 사용하여 현재 스테이지의 동작 완료 여부와 다음 스테이지 사용 가능성에 의해 파이프라인이 구동된다. 시스템내의 모든 스테이지가 자기 동기식이며 단지 다음 스테이지의 사용이 불가능할 경우 파이프라인 진행을 기다리는 대기 시간(space time)만이 존재한다. 따라서 각 스테이지는 평균 동작 시간으로 실행되며, 단지 스테이지간 제어 신호의 핸드쉐이크 시간이 추가로 필요하다. 또한 불필요한 스테이지는 동작을 수행하지 않기에 성능 및 전력 측면에서 동기식 시스템에 비해 보다 나은 성능을 얻는다[3]. 일반적으로 N 스테이지 파이프라인 시스템은 파이프라인을 구성하지 않은 동일 시스템에 대해 N 배의 성능향상을 기대한다. 그러나 제어 및 데이터 의존성 발생, 서로 다른 스테이지 수행시간, 동기화를 위한 오버헤드 및 다중 사이클 명령어 실행 등의 이유로 보다 적은 성능 향상을 얻는다. CISC 임베디드 컨트롤러는 효과적인 명령어와 어드레싱 모드를 제공한다. 이는 코드 효율성을 개선하고 더 작은 메모리와 레지스터 사용을 허용하나 다중 명령어 사이클과 같은 복잡한 제어 매커니즘을 요구하고, 이는 파이프라인 시스템 구성시 중요한 성능 감소 요인이 된다.

본 논문에서 제안한 A8051 구조는 Intel 8051과 완전한 명령어 호환성을 갖고 비동기 설계 방법을 채택한 파이프라인 구조를 갖는다. 성능 향상을 위해 첫째, 비동기회로의 특징을 이용한 단순한 분기 예측 방법을 통해 분기 명령어로 발생되는 제어 해저드 페널티를 감소시켰다. 둘째, 균등한 데이터패스를 갖는 5 스테이지 파이프라인으로 구성하였다. 셋째, 다중 사이클 명령어 실행시 발생되는 버블을 감소시키기 위해, multi-looping을 허용한 RISC에 가까운 단순한 명령어 실행 스킴으로 명령어별로 그룹화하였다.

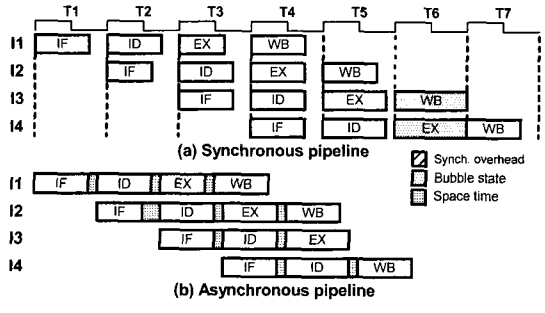


그림 1. 동기식 및 비동기식 파이프라인 비교
Fig. 1. A comparison of synchronous and asynchronous pipeline.

마지막으로 가변 길이의 명령어 처리 방안을 제시하였고, 시스템 구성후 테스트 벤치를 통한 실험으로 이의 결과를 검증하였다.

본 논문은 Ⅱ장에서 Intel 80C51과 A8051의 아키텍쳐 및 명령어 실행 스킴을 비교하였고, Ⅲ장은 제안된 비동기 파이프라인 구조 제어 매커니즘을 기술하고, Ⅳ

장은 실험을 통한 결과를 토대로 성능 평가를 하고 V 장에서 결론을 맺는다.

II. Intel 80C51과 제안된 A8051 구조

Intel 80C51은 8-bit CISC 임베디드 컨트롤러로 <그림 2>와 같이 내부 버스를 SFR등의 레지스터들과 ALU 그리고 내부 프로그램 ROM과 데이터 RAM 및 기타 블록들이 데이터 전송을 위해 공유한다. 또 포트 로직을 통해 외부 메모리가 억세스된다. 이 경우 명령어 수행시 각 스테이지에서 평균적으로 버스 대역폭의 절반의 스위칭이 발생한다[7].

Intel 80C51 명령어 집합은 연산, 논리, 데이터 전송, 부울 연산, 그리고 점프 명령 등 다섯 가지 명령어 클래스로 구분되는 255개의 다양한 명령어를 지원하며 이는 6 가지 주소 지정 모드를 갖고 가변적인 명령어 길이를 갖는다. 각 명령어 실행은 1, 2 그리고 4 머신 사이클동안 수행된다. <그림 3(a)>에서 나타낸 것처럼, 각 머신 사이클은 S1에서 S6의 6 개의 상태로 나뉘며, 각 상태는 미리 할당된 통신 또는 연산 동작을 수행한다[3]. 이러한 동기식 실행 스킴은 CISC 방식의 경우 명령어 수행시 사용되지 않는 많은 리던던시 스테이지를 요구한다. 예를 들면 INC A의 경우, IF, ID 그리고 EX 스테이지 수행만이 요구되고 나머지 상태는 리던던시 상태가 된다. 명령어 실행이 1 머신 사이클의 배수로 실행됨에 따라 동기화를 위한 리던던시 상태가 발생되

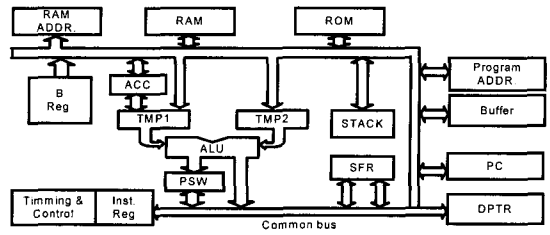


그림 2. Intel 80C51 블록도
Fig. 2. The block diagram of Intel 80C51.

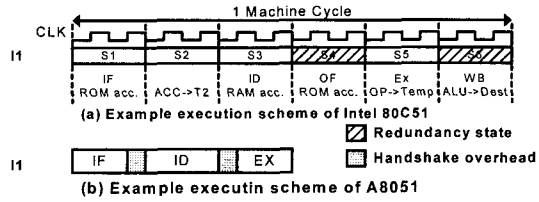


그림 3. INC A 명령어 실행 스킴 비교
Fig. 3. A comparison for INC A instruction execution scheme.

고 이러한 버블 스테이지는 시스템 성능 하락의 원인이 된다.

CISC 파이프라인 구조는 가변적인 명령어 길이와 다중 명령어 사이클과 같은 복잡한 명령어 실행 스킴 때문에 RISC 구조에 비해 복잡한 제어 매커니즘이 요구된다. 또한 다중 사이클을 갖는 명령어 처리시 많은 버블 스테이지가 발생하며 이는 제어 및 데이터 해저드에 따른 페널티를 증가시킨다. 또한 과도한 공용 버스 부하와 각 스테이지의 동기화를 위한 리던던시는

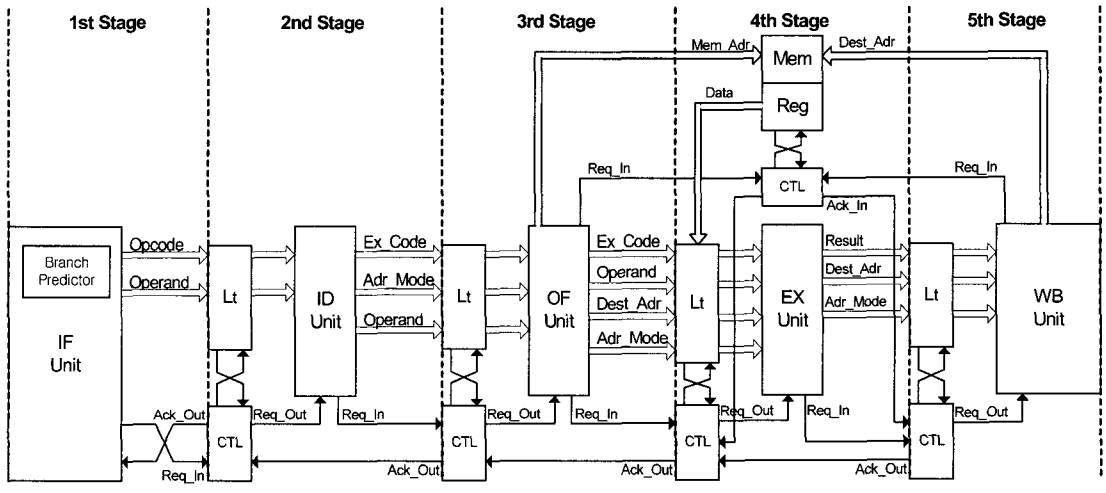


그림 4. 제안된 A8051 아키텍쳐
Fig. 4. The architecture for the proposed A8051.

파이프라인 성능을 크게 감소시킨다. 따라서 대부분의 고성능 파이프라인 프로세서는 RISC 구조를 채택하였다.

본 논문에서 제안한 A8051은 비동기 파이프라인 구조를 상용 CISC 임베디드 컨트롤러에 적용하여, 앞에서 기술한 단점을 보완하고 비동기식 파이프라인에 적합한 아키텍처를 갖도록 여러 해결 방안을 제시하였다. 이를 통해 단순한 제어 흐름, 높은 성능 향상 및 저전력 이득을 얻도록 하였다.

우선 RISC에 가까운 파이프라인 성능을 얻기 위해, A8051의 명령어 실행 스킴을 개선하였다. 동기화를 위한 버블 상태를 제거한 후 다중 사이클이 필요한 명령어 실행을 위해 OF 및 EX 스테이지만의 다중 루핑(multi-looping)을 허용하는 데이터 패스를 구성하였다. 이는 버블 스테이지 제거 뿐만 아니라, 해저드에 의해 발생되는 페널티를 크게 감소시킨다. 비동기식 파이프라인 시스템에서, 모든 명령어가 각 스테이지를 모두 사용할 필요는 없으며, 동작이 필요한 스테이지만을 활성화하고 동작이 불필요한 스테이지들은 수행되지 않고 단지 데이터만을 다음 스테이지로 전송한다. 이렇게 동작이 불필요한 스테이지를 버블(bubble)이라 부르며, 동기식 파이프라인의 경우 외부 메모리 억세스와 같은 이유로 리던던시 스테이지로 유지하나 비동기 회로는 필요할 경우에만 억세스를 허용하므로 이러한 버블이 불필요하다. 단지 외부 메모리로부터 데이터가 수신되기까지 대기하는 space time만이 필요하다. 따라서 위에 언급된 INC A 명령어의 경우 <그림 3(b)>에서 보여진 것처럼 외부 메모리 억세스나 WB 스테이지가 불필요하기 때문에 이러한 버블 스테이지를 수행하지 않고 IF, ID, 그리고 EX 스테이지만을 수행한 후 종료된다. 또한 A8051의 각 모듈의 파이프라인 구성을 위해 각 모듈간의 통신은 point-to-point 방식을 사용하였다. 이는 파이프라인 진행을 단일 제어 유닛에 집중시키는 대신 각 비동기 모듈로 분산시켜 내부 버스의 로드를 줄이고 각 스테이지의 동작에 따라 필요한 제어만을 갖도록 개선하였다. <그림 4>와 같이 A8051 구조는 IF, ID, OF, EX, 그리고 WB의 다섯 스테이지를 갖는 파이프라인 구조를 갖는다. 각 스테이지 모듈은 4-phase 비동기 핸드쉐이크 방식을 통해 데이터를 전송한다. 따라서 각 스테이지의 명령어 종료 시간은 클럭에 의해 결정되지 않고, 각 스테이지의 평균 수행 시간으로 연산 후 결과를 다음 스테이지로 전송한다. 각 스테이지의 수행 기능은 다음과 같다.

(a) 첫 번째 스테이지는 IF (Instruction Fetch) 유닛으로 프로그램 메모리로부터의 명령어 페치를 담당한다. 또한 제어 해저드를 감소시키기 위해 프리디코딩을 통해 페치된 명령어의 분기 명령어 여부를 판단하고, 분기 명령어일 경우 분기 예측을 통해 다음에 수행할 명령어의 주소를 결정한다. (b) 두 번째 스테이지는 ID (Instruction Decode) 유닛으로 페치된 명령어 opcode를 이의 마이크로명령어로 매핑하고 이전 명령어와의 데이터 의존성을 검사 후, 만일 데이터 해저드가 발생할 경우 해소될 때까지 파이프라인을 스톨시킨다. (c) 세 번째 스테이지는 OF (Operand Fetch) 유닛으로 명령어의 operand를 메모리로부터 페치한다. (d) 네 번째 스테이지는 EX (Execution) 유닛으로 마이크로명령어와 오퍼랜드를 수신 후, 마이크로명령어에 따라 연산을 수행하고 결과를 누산기에 저장한다. (e) 마지막으로 연산 결과를 레지스터나 데이터 메모리로의 저장은 WB (Write Back) 스테이지가 담당한다.

A8051의 지연 모델은 DI (Delay Insensitive) 모델을 기반하여 설계하였고, 각 블럭간 데이터를 위해 듀얼 레일 인코딩된 데이터의 4-phase 비동기 핸드쉐이크 프로토콜을 사용하였다. <그림 5(b)>는 <그림 5(a)>의 블록간 데이터 전송을 위한 핸드쉐이크 타이밍도를 나타내며, <그림 5(c)>는 조합회로 연산 종료후 데이터의 유효함을 다음 블럭에 지시하는 완료 신호의 생성을 위한 회로를 나타낸다.

Ⅲ. A8051 파이프라인 구조와 제어 매커니즘

동기식 Intel 80C51 구조에 비동기식 파이프라인 구조를 적용시, 앞 장에서 설명한 것과 같이 많은 문제점들이 발생한다. 본 장에서는 A8051 아키텍처에서 이들 문제점들을 해결하기 위해 사용한 여러 해결 방안에 대해 기술한다.

1. 명령어 실행 스킴에 따른 명령어 그룹화

Ⅱ장에서 설명된 것처럼, Intel 80C51은 명령어 수행시 동기화를 위해 버블 즉 리던던시 상태를 갖는다. 그러나 A8051 명령어는 클럭에 의한 스테이지 제어를 갖지 않는 대신 각 스테이지가 활성화된 후 유효 출력의 발생 여부와 다음 스테이지의 사용 가능 여부에 따라

파이프라인 진행을 결정한다. 즉 이와 같은 데이터 의존한 제어로 파이프라인이 동작되고 불필요한 스테이지를 사용하지 않기 때문에 버블 스테이지가 필요한 동기식 파이프라인보다 단순한 데이터 패스를 갖는다.

일례로, 분기 명령어 처리시 무조건 분기의 경우 분기 예측을 통해 다음 수행할 명령어의 주소 결정을 IF 스테이지내에서 끝낼 경우 다음 스테이지로 진행할 필요 없이 현재 수행중인 명령어를 종료하고, 결정된 주소로부터의 명령어 페치를 연속적으로 수행할 수 있다. 또 이는 클럭에 의해 제한을 받지 않기 때문에 오히려 무조건 분기 명령어 실행은 다음 명령어 실행과 오버랩되어, IF 스테이지 동작 시간이 긴 하나의 명령을 수행되어 오히려 페널티대신 명령어 실행이 가속되는 효과를 얻는다.

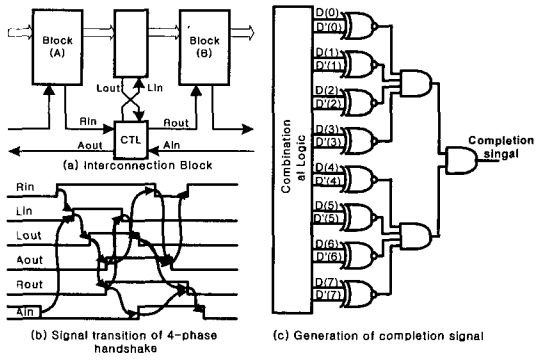


그림 5. A8051 handshake model
Fig. 5. The handshake model of the A8051.

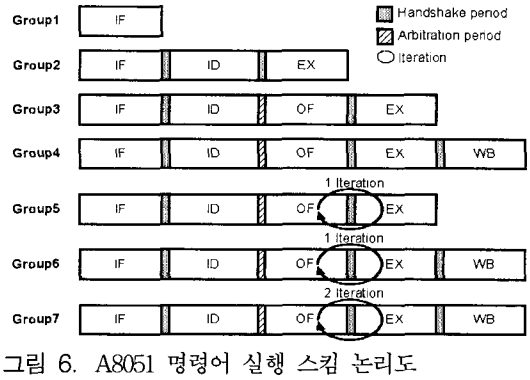


그림 6. A8051 명령어 실행 스킴 논리도
Fig. 6. The logical layout for execution scheme in an A8051.

이러한 이유로 A8051은 필요한 스테이지만을 동작하도록 명령어 실행 스킴을 재정의하여 리던던시 상태를 갖는 동기식과 비교하여 유휴 시간이 많이 감소되며 입력 레이턴시도 감소되므로 시스템 성능은 크게 증가한다. 따라서 A8051은 <그림 6>에 보인 것처럼 총 7개의 명령어 실행 스킴을 갖도록 재정의하였다. 즉 무조건 분기 명령어와 NOP 명령어의 경우 IF 스테이지만의 동작이 필요하므로 다음 스테이지로 진행하지 않고 분기 주소만 결정한 후 명령어 수행을 종료한다. 또 누산기 연산의 경우도 누산기값의 페치와 연산 결과값 저장은 EX 스테이지 동작내에 포함되므로 OF와 WB 스테이지 수행은 불필요하다. 이와 같은 리던던시 스테이지를 제외하고 동작시킴으로서 효율적인 명령어 수행이 가능하다. 이와 같이, 연속된 명령어 실행들간 지연이 적을수록 더욱 높은 성능을 얻을 수 있기 때문에, A8051은 복잡한 CISC 명령어의 실행 스킴을 최적화하여 단순한 선형 파이프라인 실행과 같이 동작되도록 구성하였고 이는 불필요한 스테이지의 생략과 필요시 스테이지의 반복을 허용한다. <표 1>은 A8051의 모든 명령어가 어떤 명령어 실행 스킴을 갖는가를 나타낸다. 파이프라인은 크게 IF, ID, OF, EX 그리고 WB의 5개의 스테이지로 구성되며 각 명령어 그룹은 필요한 스테이지 연산만을 수행하고, 필요시 스테이지들의 반복도 허용한다. 즉 그룹 5, 6 그리고 7은 OF와 EX 스테이지가 반복되는 명령어 실행 스킴을 갖는다. A8051에서 스테이지간의 반복을 허용하기에, OF 스테이지가 앞 명령어의 피드백된 요구와 명령어 디코더 유닛으로부터 전송되는 요구가 동시에 발생될 수 있고, 먼저 수행된 명령어가 OF 스테이지로부터 전송한 요구를 높은 우선순위로 중재된다. 따라서 이를 위해 우선순위를 부여하는 중재 블록과 추가적인 제어 신호가 필요하다.

표 1. 재정의된 명령어 실행 스킴
Table 1. The re-defined instruction execution scheme.

Group	Scheme	Num	Example
1	IF	9	Unconditional branch, NOP
2	IF-ID-EX	13	Function A(or C)
3	IF-ID-OF-EX	43	Function A,Rn(or @Ri, dir, #data) Function C, bit MOVX A, @Ri(or @DPTR) MOVX @DPTR, A MUL, DIV, POP
4	IF-ID-OF-EX-WB	24	Function Rn(or @Ri, dir), A Function bit addr MOV Rn(or Ri), dir(or #data) MOV dir, Rn (or @Ri) MOVX @Ri, A, PUSH
5	IF-ID-OF-EX-OF-EX	4	Function A, @A+DPTR (or PC) INC DPTR MOV DPTR, #data 16
6	IF-ID-OF-EX-OF-EX-WB	5	Function dir, #data MOV dir, dir
7	IF-ID-OF-EX-OF-EX-OF-EX	13	Conditional branch

이러한 경우 중재 방법은 다음과 같이 간단히 처리된다. 한 명령어가 디코딩된 후 A8051은 어떤 명령어 그룹에 속하는 지 결정한다. 동시에 OF-EX 스테이지의 반복 횟수가 비동기 다운 카운터로 세트가 되고, 이 카운터 값은 OF 유닛을 반복 사용될 때마다 감소된다. 카운터 출력이 0인 경우 카운터는 ID 유닛에서 전송된 명령어를 수행하고 그렇지 않은 경우 ID 유닛으로부터의 데이터 전송에 Ack 신호를 전송하지 않음으로서 ID 유닛이 대기하도록 한다. 그리고 EX 유닛에서 전송된 명령어 수행을 먼저 처리한다. 위와 같이 가장 단순한 명령어 실행 스킴으로 명령어를 수행함으로서, 최적화된 성능을 얻는다.

2. 가변적인 길이를 갖는 명령어의 페치

Intel 80C51은 오퍼랜드 수에 따라 가변적인 명령어 길이를 갖고 이의 실행을 위해 리던던시 스테이트를 갖는 최대 4 머신 사이클의 복잡한 실행 스킴과 제어 회로가 필요하다. 이는 심각한 시스템 성능 하락을 유발하며, 이를 해결하기 위해 A8051은 IF 스테이지에서 프리디코더에서 명령어 길이를 미리 결정하고 결정된 길이에 따라 명령어를 페치하여 IR 레지스터에 저장한다. IF 스테이지는 <그림 7>과 같이 구성된다. 크게 PC 및 IR 래치와 프로그램 메모리, 프리디코더 그리고 NPC (Next PC value Calculator) 블록으로 구성된다. 프리디코더 블록은 프로그램 메모리로부터 페치된 명령어의 Opcode를 미리 저장된 정보와 완전 연관 검색을 통해 명령어 길이를 결정하고 필요한 제어 신호를 전송한다. 이를 위해 각 opcode에 따라 분기 명령어 여부 및 명령어 길이 정보를 미리 저장해야 한다. 또한 페치된 명령어가 분기 명령어인 경우 무조건 분기의 경우 NPC 블록으로 제어 신호를 전송하여 다음에 수행할 명령어의 길이를 즉시 결정하여 프로그램 카운터(PC)를 갱신하고, 조건 분기의 경우 "true"로 예측된 주소를 PC로 전송한다.

만일 명령어의 길이가 1바이트가 아닌 경우 프리디코더에서 결정된 명령어 길이를 Mux 내부의 다운 카운터의 초기값으로 설정한다. Mux는 프로그램 메모리로부터 1바이트씩 페치될 때마다 IR 레지스터에 저장 후 1 씩 감소된다. Mux 출력은 IR 레지스터의 저장 위치를 결정하며 Mux 내부의 카운터 값이 0인 경우 End^+ 신호를 전송하여 명령어 전체를 페치하였음을 알리고, ID 스테이지로 명령어를 전송하기 위해 이벤트 신호로 $Rout^+$ 신호를 전송한다. IR 레지스터는 최대 3 바이트의 저장 용량을 갖는다. 프로그램 메모리로부터 페치된 데이터를 래치할 때마다 $Aout^+$ 신호를 전송하여 1 바이트를 페치했음을 알린다. 한 명령어의 다음 1 바이트를 페치하고 MUX로부터 End^+ 신호 수신시 ID 스테이지와 핸드쉐이크하여 페치된 명령어를 전송하고 전송이 종료되면 다음 명령어 수신을 위해 $Next_inst^+$를 PC 래치로 전송한다.

NPC (Next PC value Calculator) 블록은 다음 페치할 명령어 주소를 결정한다. 또한 무조건 분기명령어의 경우 다음 수행될 명령어의 주소를 결정하여 PC 래치로 전송한다. 조건 분기 명령어의 경우 임시 레지스터에 조건이 fail일 경우 수행할 명령어 주소를 저장하고 조건이 true로 예측하고 다음 실행할 명령어 위치를 PC 래치로 전송한다. 그리고 PC 래치와 IR 래치는 각각 핸드쉐이크 신호를 통해 PC와 명령어를 저장하고 <그림 5>에 보인 4-페이즈 핸드쉐이크 모델을 통해 데이터를 통신한다.

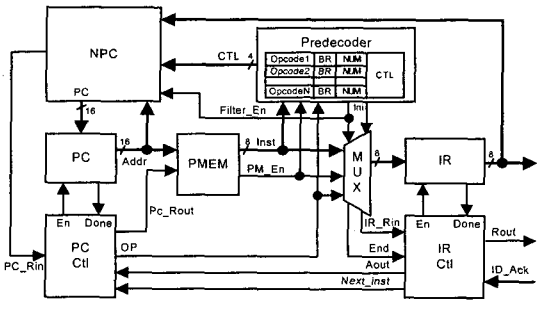


그림 7. 명령어 길이 결정 및 페치 유닛 블록도
Fig. 7. The instruction length decision and fetch unit block diagram.

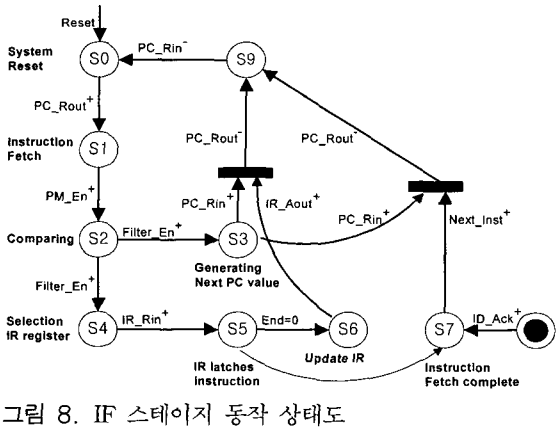


그림 8. IF 스테이지 동작 상태도
Fig. 8. The state diagram for IF stage implementation.

가변 길이 명령어 처리를 위한 IF 스테이지의 동작은 <그림 8>의 상태도로 나타내었다. $\overline{\text{Reset}}$이 high일 때, S1 상태로 진입하여 모든 래치를 초기화하고 PC 래치 입력은 프로그램 메모리의 처음 실행할 주소로 초기화한다. $\overline{\text{Reset}}$이 low로 천이시 S0 상태로 진입하여 초기화된 PC 값을 전송하고, 이의 완료 검출 후 프로그램 메모리 인에이블 입력으로 PC_Rout+ 신호를 전송한다. 또한 처음 페치될 명령어는 opcode이므로 PC_OP 신호는 high로 세트한다. S1 상태는 PC가 지정하는 위치의 명령어 1 바이트를 페치한다. PC 래치 출력의 완료 신호 검출 후 PC_Rout+ 신호를 전송하여 S2 상태로 천이한다. S2 상태에서는 opcode 수신시 미리 저장된 정보와 완전 연관 검색을 수행하여 명령어 길이에 따라 Mux의 초기값을 IF_Ini 버스로 전송하고 다음 페치할 주소 결정을 위한 제어 신호는 IF_Ctl 버스를 통해 전송한다. 이의 완료 신호 검출은 Filter_En을 high로 천이시킨다.

Filter_En+ 신호 전송은 각각 IR (Instruction Register) 래치와 NPC (Next PC value Calculation) 블록을 활성화시킨다. S4 상태에서 Mux 내부의 카운터는 Inst_Filter로부터 수신된 ini 신호값에 의해 초기화되고, 프로그램 메모리로부터의 PM_En 신호의 상승에지에서 다운 카운트한다. 이 출력은 Mux의 출력으로 연결되며, 이는 IR 래치내의 저장 위치를 결정한다. 이의 결정후 완료 신호 검출은 IR_Rin 신호를 high로 세트하고 지정된 위치로 페치된 명령어를 저장하고 Mux의 End 신호값에 따라 0인 경우 S6 상태로 천이하여 명령어 페치가 완료되지 않았기에 IR_Aout을 전송하여 현 명령어의 다음 바이트를 IR 레지스터에 저장하기 위한 동작을 요구한다. End 신호가 1인 경우 S7 상태로 천이하여 ID 스테이지로부터 ID_Ack+ 신호 수신시 현재 명령어 전체가 페치되었음을 알리고 이의 확인을 위해 Next_Inst 신호를 통해 핸드쉐이크한다. S6 및 S7 상태에서 IR 레지스터로부터 Aout+ 혹은 Next_Inst+ 신호의 수신을 대기한다.

Aout+ 신호의 수신은 명령어의 다음 바이트를 페치해야 함을 의미하여, Next_Inst+ 신호 수신은 현재 명령어가 모두 페치되어 ID 스테이지로 전송되었고 새로운 명령어를 페치해야 함을 의미한다. 이 대기 상태에서 locking이 이루어지며, 두 신호 모두 high로 천이시 S8 상태로 천이하여 회로를 비활성시키기 위해 PC_Rout- 신호를 전송한다. 이는 각 블록의 출력을 무효화하며, 마지막으로 PC_Rin 신호를 low로 천이시킨다. 이와 IR 래치와의 핸드쉐이크 종료시 S0 상태로 천이하여 PC 래치에 저장된 주소의 데이터를 페치하여 IR 레지스터에 저장하기 위해 PC_Rout 신호가 high로 다시 천이하고 다음 명령어 페치를 시작한다.

3. A8051의 제어 해저드 처리 방안

파이프라인 동작시 분기 명령어의 수행으로 인한 제어 해저드는 분기 페널티를 발생시킨다. 분기 명령어 실행시, 분기 검출 시점과 분기 주소 결정 시점이 중요한 성능 감소 요인이 된다. ID 스테이지에서 현재 페치된 명령어가 분기 명령어임을 검출할 경우, IF 스테이지에서의 페치 동작은 플러쉬되어져야 한다. 또 EX 스테이지에서 분기 주소를 결정한 경우 그 이전 스테이지까지의 동작은 분기 페널티가 된다. 따라서, 제어 해저드 발생은 전체 파이프라인 시스템의 중요한 성능 감소 요인이 된다.

제안된 A8051 구조에서는 이에 따르는 성능 손실을 줄이기 위해, IF 스테이지에서 분기 명령어임을 검출하는 pre-decoding을 수행한다. 또한 무조건 분기의 경우 다음 타겟 주소의 명령어를 즉시 결정한 후 이에 따라 PC를 갱신하여 연속적으로 페치하도록 구성하였다. 따라서, 다음 명령어의 IF 스테이지와 오버랩되며, 두 명령어 실행이 하나의 긴 IF 스테이지 동작만을 요구한다. 이에 따라 IF 스테이지 동작 시간이 증가하나 1 스테이지 스톨에 비해 성능 하락을 크게 감소된다. 조건 분기의 경우, 무조건 "true"로 예측하는 정적 분기 예측 기법을 도입하였다. 이를 위한 동작은 <그림 9>와 같이 나타난다.

프로그램 메모리로부터 페치된 opcode는 IR 레지스터로 저장된 후, ID 스테이지로 전달하여야 하나, 분기 명령어의 경우 타겟 명령어를 IR 레지스터에 저장해야 한다. 이를 위해 페치된 분기 명령어는 프리디코더에 저장된 정보와 비교후 결과를 각 블록으로 출력한다. br은 연산에 필요한 주소의 길이를 명시하고, num[0:1]은 명령어 길이, state[0:1]는 저장될 BTA 레지스터 위치를 가리킨다. select[0:1] 신호는 mux 입력중 하나의 주소를 선택하고, 이를 통해 결정된 주소는 addr_latch에 저장된 후 핸드쉐이크를 통해 프로그램 메모리로 전달되고 다음 타겟 명령어가 페치시 IR 레지스터로 저장한다. 따라서 무조건 분기 명령어의 경우 다음 타겟 명령어의 IF 스테이지와 오버랩되므로 연산 시간이

길어지나 추가적인 분기 페널티는 발생하지 않고 오히려 타겟 명령어 페치까지 필요한 시간이 분기 명령어와 이의 타겟 명령어를 개별적으로 페치할 때보다 걸리는 시간보다 적다. 정적 분기 예측을 통한 조건 분기의 경우에만 조건 실행후 비순차실행인 경우 3 스테이지 스톨이 발생한다. 이 경우 각 파이프라인 스테이지의 연산 결과를 모두 리셋한다. 따라서 조건 분기 명령어의 경우 분기 예측이 틀릴 경우에만 스톨이 발생하기 때문에 전체적으로 제어 해저드에 따른 시스템 성능 하락의 폭을 상당히 낮출 수 있다. 조건 분기 명령어의 분기 예측은 이와 같은 데이터 의존한 제어를 통해 향후 더욱 높은 효율의 분기 예측 기법의 적용도 가능하다.

4. 파이프라인 스테이지의 수

파이프라인은 흔히 동일 혹은 비슷한 수행 시간을 갖는 각 스테이지로 구성시 스테이지간의 동기화를 위한 오버헤드를 줄일 수 있기에 최적화된다[10]. 그러나 각 수행시 걸리는 시간이 절대적으로 동일하기는 불가능하고, 동기식의 경우 각 스테이지의 최대의 지연 시간을 클럭의 기준으로 삼는 반면 비동기식 파이프라인 구조는 평균 지연 시간으로 각 스테이지가 동작되기에 성능상 효율이 높다. 따라서 제반사항을 고려하여 스테이지의 수를 결정해야 하며, A8051은 각 스테이지에 부여된 작업의 균형성과 해저드 문제를 고려하여 총 5개의 스테이지로 구성하였다.

A8051의 각 스테이지의 평균 지연 시간은 0.35㎛ 표준 CMOS 공정을 사용하여, 합성시 <표 2>와 같이 나타난다. WB 스테이지 수행 시간이 상대적으로 타 스테이지에 비해 크지만 명령어의 실행 빈도가 타 스테이지보다 적기 때문에 전체 시스템 성능에 미치는 영향은 작다.

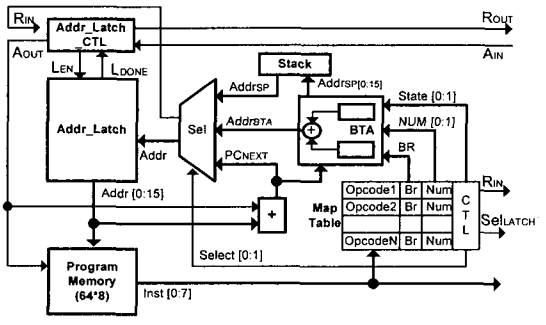


그림 9. 타겟 주소 연산 유닛 블럭도

Fig. 9. The block diagram of target address calculation unit

표 2. 각 스테이지별 평균 수행 시간

Table 2. The average execution time of each stages.

Stage	IF	ID	OF	EX	WB
평균수행시간	5.9 ns	4.8 ns	5.0 ns	5.1 ns	6.6 ns

Ⅳ. 시뮬레이션 결과 및 성능

본 논문에서 제안한 A8051은 Intel 80C51과 완전히 명령어 호환되며 주요 데이터 패스는 IF, ID, OF, EX 그리고 WB 유닛으로 구성되는 파이프라인 구조를 갖고 각 스테이지 데이터 전송은 4-페이즈 핸드쉐이크 프로토콜을 이용한다. 또 회로 합성을 위해 Verilog HDL로 설계 후 0.35㎛ CMOS 표준 셀 라이브러리를 이용하여 합성하였다.

성능 평가를 위해 우선 80C51의 명령어를 순차적으로 실행하여 평균 동작 시간을 측정하였고, 테스트 벤치 프로그램을 사용하여 약 3,200 개의 명령어를 수행하여 성능 평가를 수행하였다. 파이프라인 시스템 성능은 분기 명령어의 발생 빈도에 따라 높은 편차가 발생한다. 일반적으로 공학용 프로그램의 경우 분기 명령어는 평균 5% 그리고 일반 프로그램의 경우 평균적으로 20% 정도의 분기 명령어 수행 빈도를 갖는다[10]. 이에 따라 테스트 벤치는 약 20%의 분기 명령어를 갖도록 구성하였다. 성능 평가를 위해 동기식 Intel 80C51과 파이프라인 구조를 갖는 A8051과 파이프라인 구조를 갖지 않는 A8051 그리고 Gegeldonk의 Asynch80c51[7]의 성능을 비교하였다.

255개의 명령어를 순차 수행시 각각의 성능은 <표 3>과 같이 나타난다. 즉 본 논문에서 제안한 비동기 설계 기법을 적용 후, 명령어 실행 스킴을 최적화하여 리던던시 상태 수행을 제거한 경우, 35.8 MIPS의 성능을 나타내고, 여기에 파이프라인 구조를 적용한 경우 평균적으로 75.5 MIPS의 성능을 갖는다. 명령어의 순차 수행시 비동기 파이프라인 구조를 갖는 A8051은 Intel 80C51에 비해 약 23.6 배의 성능 향상을 갖고 Asynch80C51에 비해 약 18.9 배의 성능 향상을 갖는다. 따라서 A8051은 비동기 파이프라인 구조의 최적화를 통해 높은 성능을 얻었음을 나타낸다.

표 3. 명령어 순차 실행 결과
Table 3. The result of sequential execution.

Ver.	Intel 8051 (36MHz)	Async80C51 (0.6㎛ CMOS)	A8051(proposed) non-pipeline version (0.35㎛ CMOS)	A8051(proposed) pipeline version (0.35㎛ CMOS)
MIPS	3.2 MIPS	4 MIPS	35.8 MIPS	75.5 MIPS

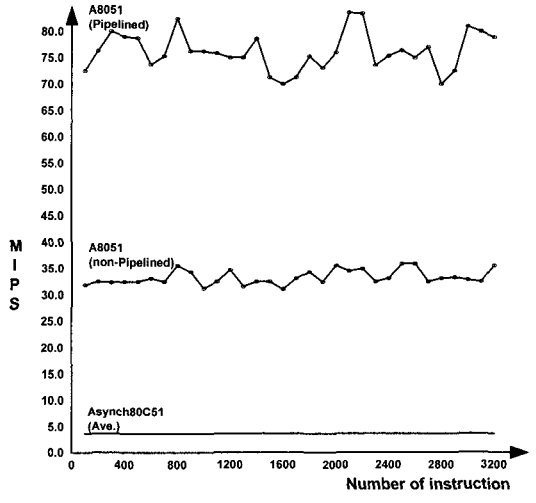


그림 10. 명령어 실행 수에 따른 성능 비교 결과
Fig. 10. The performance comparison to the number of instruction.

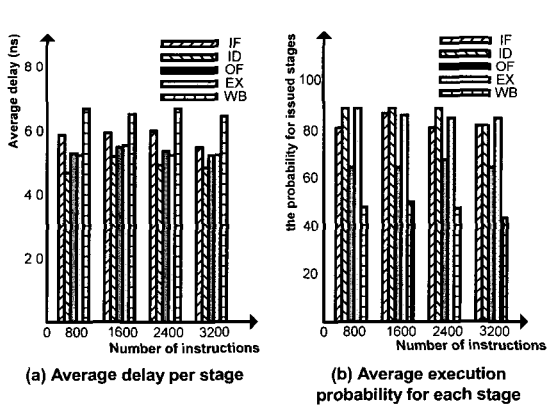


그림 11. 스테이지별 평균 동작 시간 및 수행 빈도
Fig. 11. The average execution time and the frequency of operation.

<그림 10>은 테스트 벤치 프로그램 코드를 사용한 실행 결과를 나타낸다. 테스트 벤치 프로그램은 약 3,200개의 명령어를 수행하며 평균적으로 20%의 분기 명령어를 갖는다. 파이프라인 구조를 갖지 않는 A8051의 경우 평균적으로 33 MIPS를 그리고 파이프라인 구조를 갖는 A8051의 경우 평균적으로 76 MIPS의 성능을 보여준다. 따라서 파이프라인 구조를 갖는 A8051의 경우 파이프라인 구조를 갖는 동일 구조에 비해 2.3 배의 성능 향상을 얻었다. 또 분기 명령어의 실행 빈도에 따라 15 MIPS의 큰 성능차가 남을 알 수 있다.

<그림 11>은 각 스테이지별 평균 동작 시간 및 수행 빈도를 나타낸다. <그림 11(a)>는 명령어 실행에 따른 각 스테이지의 평균 지연 시간을 나타낸다. 이는 <표 2>의 평균 지연 시간에 따른 지연이 발생됨을 나타낸다. WB 스테이지 지연이 상대적으로 약간 크나 <그림 11(b)>에서 알 수 있듯이 파이프라인 수행중 WB 스테이지의 수행 빈도가 타 스테이지에 비해 적기 때문에, 이에 따른 성능 감소는 크지 않음을 알 수 있다.

메모리 억세스를 요구하는 OF 스테이지와 WB 스테이지의 사용 빈도가 낮기 때문에 동시에 메모리 억세스 요구시 발생되는 중재에 필요한 오버헤드가 적음을 알 수 있다. 5 스테이지 파이프라인 구조를 선택했음에도 불구하고 다중 사이클 파이프라인 채택, 빈번한 점프 및 분기 명령어 실행 그리고 각 스테이지간 핸드쉐이크 오버헤드로 인해 약 2.3 배 정도의 성능 향상을 얻었다.

A8051의 성능 개선에 가장 중요한 요소는 분기 명령어로 인한 제어 해저드이며, <그림 9>에서 보여지듯이 최고 성능을 갖는 구간과 최악의 성능을 갖는 구간간에 11 MIPS의 속도차를 보인다.

V. 결 론

본 논문은 완전한 비동기식 제어 스킴 파이프라인 구조를 갖는 임베디드 컨트롤러 A8051 아키텍처를 제안한다. 이는 DI 지연 모델을 기반으로한 비동기식 데이터 패스를 갖고 5 스테이지 파이프라인으로 구성되었다. 스테이지간 데이터 전송은 2선식 코드로 전송하며, 제어를 위해 4-phase 핸드쉐이크 프로토콜을 사용한다. A8051은 CISC 파이프라인 구조에 적합한 다중 사이클 명령어 실행 스킴을 갖고 이에 적합하도록 명령어 실행 스킴을 재정의하였다. 또한 제어 해저드 처리 및 가변적인 명령어 길이를 갖는 문제에 대한 처리 방안을 제시하였다.

제안된 A8051은 다음과 같은 몇 가지 장점을 갖는다. 첫째, 단순화된 제어를 갖는 다중 명령어 실행 스킴을

적용하였다. 둘째, 분기명령어 수행시 발생되는 페널티를 감소하기 위해, IF 스테이지에서 pre-decoding을 통해, 분기 명령어를 검출하고, 무조건 분기 명령어와 타겟 명령어의 IF 스테이지를 오버랩시켜 space time은 증가하나, 전체 성능 하락은 감소되었다. 또한 조건 분기의 경우 정적 분기 예측을 통해 제어 해저드에 따른 성능 감소를 해결하였다. 마지막으로 가변적인 길이를 갖는 명령어 페치를 위한 단순한 제어 방법을 제안하였다. 0.35㎛ 표준 CMOS 공정에 따라 구현된 A8051 아키텍쳐는 36MHz 클럭으로 동작하는 Intel 80C51과 다른 비동기식 80C51에 비해 높은 성능을 나타내었다. 실험을 통해 확인된 A8051의 평균 처리 속도는 약 76 MIPS로 이는 동기식 Intel 80C51 및 다른 비동기 80C51에 비해 약 24배의 높은 처리속도로 명령어를 처리한다.

참 고 문 헌

[1] S. Hauck, "Asynchronous design methodologes : an overview," *Proc. the IEEE*, Vol. 83, No 1, pp. 69~93, Jan, 1995.

[2] M. B. Josephs, S. M. Nowick, C. H. Van Berkel, "Modeling and Design of Asynchronous Circuits," *Proc. the IEEE*, Vol. 87, pp. 234~242, Feb, 1999.

[3] K. R. Cho, K. Okura, K. Asada, "Design of a 32-bit Fully Asynchronous Microprocessor (FAM)," *Proc. 35th Midwest Symp. on Circuits and Systems*, Vol. 2, pp. 1500~1503, 1992.

[4] T. Nanya et al., "TITAC-2 : an asynchronous 32-bit microprocessor based on scalable-delay-insensitive model," *Proc. ICCD'97*, pp. 288~294, 1997.

[5] S. B. Furber, J. D. Garside, D. A. Gilbert, "AMULET3 : a high-performance *self-timed ARM microprocessor,"*, Proc. ICCD'98, pp. 247~252, 1998.

[6] Jamin M. C. Tse and Daniel P. K. Lun, "ASYNMPU : A Fully Asynchronous CISC Micro-processor," *ISCAS 1997*, pp. 1816~1819, 1997.

[7] H. van Gageldonk, D. Baumann, K van Berkel, D. Gloor, A. Peeters, and G. Stegmann. "An asynchronous low-power 80C51 microcontroller," *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 96~107, 1998.

[8] I. E. Sutherland, "Micropipelines," *Communication of the ACM*, Vol. 32, No. 6, pp. 720~739, 1989.

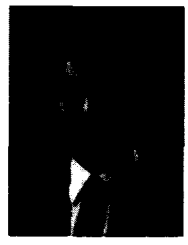
[9] Intel, "*Microprocessor and Peripheral Hand book*," 1997.

[10] D. Sima, T. Fountain and P. Kacsuk, "*Advanced Computer Architecture : A Design Space Approach*," Addison-wesley, 1997.

저 자 소 개

李 制 勳(正會員)

1998년 2월 : 충북대학교 정보통신공학과 학사. 2001년 2월 : 충북대학교 정보통신공학과 석사. 2003년 2월~충북대학교 정보통신공학과 박사과정수료. <주관심분야 : 홈 네트워킹, 통신용 ASIC 설계. 고속 마이크로프로세서 설계>



趙 慶 錄(正會員)

1977년 : 경북대학교 전자공학과 공학사. 1989년 : 일본 동경대학교 전자공학과 공학석사. 1992년 : 일본 동경대학교 전자공학과 공학박사. 1979년~1986년 :(주)금성사 TV연구소 선임연구원. 1999년 1월~2000년 1월 Oregon State University 객원교수. 1992년~현 : 충북대학교 정보통신공학과 교수. <주관심분야 : VLSI 시스템설계, 통신 시스템용 LSI 개발, 고속 마이크로프로세서 설계>