優受論文

8086 프로세서용 인 써키트 에뮬레이터의 제작에 관한 연구

강 중 용
(서울대공대 제어계측공학과4 학년)

Ⅰ. 서 론

1971년 미국의 인텔사에 의해 세계최초의 단일칩 마이크로프로세서인 4004가 개발된 이래 반도체기술의 급속한 향상으로 인해 그 사용범위가 급속히 향상되었다. 이러한 사용기회의 확대는 시스템의 개발자가 마이크로프로세서를 이용한 시스템의 개발과정에서 일반적인 전기전자기기의 제작과정은 달리 소프트웨어와 하드웨어의 결합이라는 과정이 추가되게 된다. 그런데 이 두 부분의 결합과정에서 오류의 검출및 수정은 서로의 특성차이에 의하여 쉽게 이루어질 수 없으며 마이크로프로세서를 이용한 시스템에서의 개발실패사례는 대부분 이러한 소프트웨어와 하드웨어의 결합과정에서 발생하게 된다.[1]

이러한 문제점들이 해결을 용이하게 하여주고 또 하드웨어와 스프트웨어 각각의 개발상의 편의를 위해 범용계측기 제작회사나 마이크로프로세서의 제작회사에서는 마이크로프로세서 개발 시스템(Microprocessor Development System, MDS)을 시판하고 있는데 이는 소프트웨어의 개발을 위한 어셈블러, 크로스어셈블러, 및 여러 종류의 크로스 컴파일러와 링커, 그리고 하드웨어 개발을 위한 로직 어날라이저, 통합과정의 개발 편의를 위한 인써키트 에뮬레이터(In- Circuit Emulator, ICE)를 내장하여 마이크로프로세서를 사용하는 시스템의 구현에 있어서 거의 완벽한 기능을 제공해준다.[2] 이중 ICE는 테스트하려는 시스템과 ICE시스템과의 물리적인 버스전환을 통하여 새로이 개발된 시스템의 실제 조건에서 프로그램의 수행및 시스템의 상태확인이 가능하도록 하는 시스템으로 MDS의 핵심적인 부분이라 할 수 있다.

본 연구에서는 현재 IBM PC등에 사용되는 8088프로세서의 16비트버전인 8086프로세서의 ICE를 설계 제작하였다. 8088프로세서와 8086프로세서는 그 내부 기능이 동일하기때문에 어셈블러나 링커등의 소프트웨어 개발장비들을 IBM PC에서 지원받을 수 있으므로 IBM PC에 연결된 ICE는 전체적으로 하나의 MDS시스템을 구성할 수 있다. 제작된 ICE는 1) 테스트하려는 시스템의 메모리에 대한 읽기 및 쓰기, 2) 테스트프로그램의 실제 조건에서의 수행, 3) 디버깅 기능, 4) ICE의 메모리 영역을 테스트하려는 시스템에서 활용하도록하는 기능 등을 수행할 수 있도록 하였다. 또 8086프로세서는 싱글프로세서모드와 멀티프로세서모드의 두가지 동작모드가 있는데 싱글프로세서모드에서 동작될 수 있도록 설계되었으며 ICE의 기능 수행을 위한 논리 회로의 구성과 이해에 주안점을 두었다.

Ⅱ. ICE의 제작

1. 제작되는 ICE의 기본 구조

ICE는 물리적인 버스 전환을 통하여 테스트하려는

시스템과 에뮬레이터를 번갈아가며 동작되기 때문에 이러한 전환 구조를 어떻게 적절히 설계하느냐에 따라 그 성능이 좌우된다. 이러한 ICE의 버스전환 구조는 여러 사람들에의해 제시되었고 또 ICE의 여러 메이커에서는 자기자신의 특색있는 전환구조를 사용하여 제품을 개발하였다. 이런 구조들을 크게 구분하면 두가지로 나누어지게 되는데 그 하나는 마스터 슬레이브 시스템이고, 다른 하나는 싱글프로세서 시스템이다. [3]

이 두 방식의 개략적인 구조는 그림 1, 2와 같다.
본 연구를 통하여 제작되는 ICE의 구조도는 그림 3과 같다.

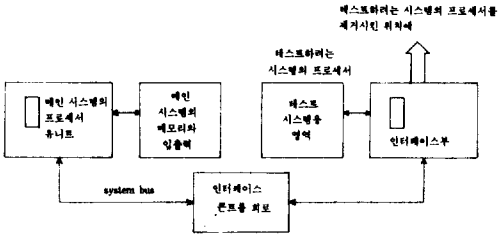

그림 1. 마스터 슬레이브 구조에서의 전환방식

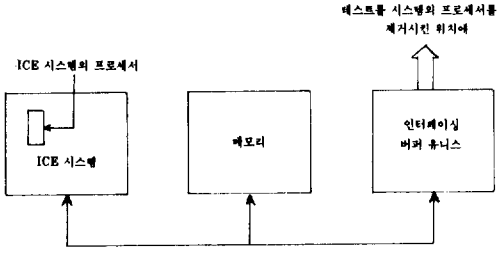

그림 2. 싱글프로세서 구조에서의 전환방식

마스터 슬레이브 방식의 ICE는 두개나 그 이상의 마이크로프로세서에의해 동작되는데 하나는 ICE자신을 위한 것이고 또 하나는 에뮬레이션을 위한 것이다. 이 방식에서는 테스트하려는 시스템의 프로세서 종류에 따라 ICE시스템의 프로세서를 교체할 필요가 없으며 ICE 내부의 버스가 테스트하려는 시스템에 직접 접속되지 않기 때문에 두 시스템과의 분리가 완벽히 이루어질 수 있다는 장점이 있다. 그러나 이러한 구조에서는 그 소프트웨어가 매우 복잡하여지며 콘트롤 버스로직에서 심각한 지연현상을 나타낸다는 문제들이 발생하게 된다. 싱글프로세서 시스템에서는 시스템의 버스가 인터페이스부를 통하여 직접 테스트하려는 시스템에 연결되기 때문에 버스가 공유되어 메모리나 입출력등에서 ICE와 테스트하려는 시스템의 여러 resource가 겹칠수 있다는 단점이 있으나 그 구조의 단순성이나 경제성등에 따른 여러 잇점들이 있게된다.

본 연구에서는 이러한 두 방식의 절충적인 형태를 택하여 하나의 마이크로프로세서를 사용하면서도 적절한 타이밍의 콘트롤신호의 생성에 의하여 버스를 완전히 독립시키는 형태의 구조를 이용하여 ICE를 제작하였다.

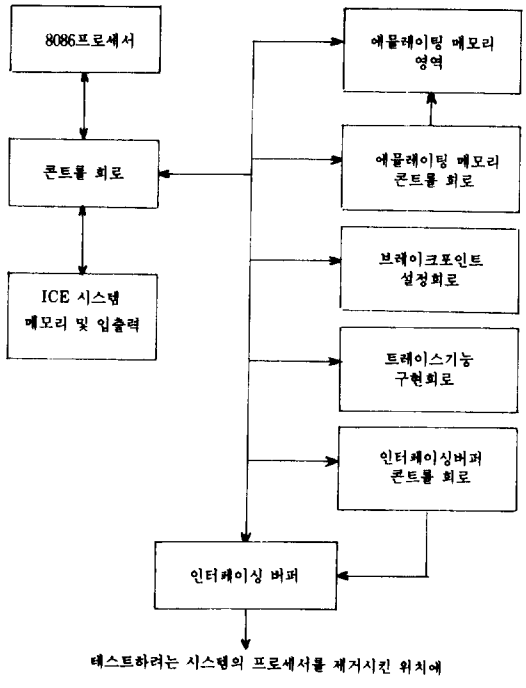

그림 3. 제작되는 ICE의 구조도

2. ICE 하드웨어의 설계

1) 콘트롤 유니트

콘트롤 유니트는 selector의 구실을 하게되는데 이 회로의 출력에 의하여 ICE시스템은 버스의 전환을 행할 수 있게 된다. 이것이 그림 4의 T/$\overline{E}$신호가 되는 것이다. ICE에서 사용되는 8086 프로세서의 특정 신호에 의해 콘트롤 유니트는 테스트하려는 시스템을 선택할 것인지 혹은 ICE자신을 선택할 것인지를 결정하여 준다. ICE가 테스트하려는 시스템과 에뮬레이터 시스템을 번

갈아 오가며 데이타를 전송하고 인스트럭션을 수행하기 때문에 이 부분의 설계는 ICE의 핵심부분이라 할 수 있으며 하드웨어적으로는 마이크로프로세서의 버스 타이밍과 밀접한 관계가 있다. 콘트롤 유니트에서 발생되는 콘트롤 신호에 의해 에뮬레이터는 적절한 순간에 자신의 버스를 모두 사용 불능 상태로 만들고 대신 테스트하려는 시스템의 버스를 사용가능 상태로 만들어 프로세서가 테스트하려는 시스템에서 동작하도록 한다. 그리고 주어진 명령의 조건에 따라 적절한 순간에 다시 버스의 상태를 원래대로 복구시키다.

그림 4는 제작된 ICE의 콘트롤회로를 보여준다. 콘트롤회로가 그 select신호를 T/$\overline{E}$의 값을 '1' 로 하여 테스트하려는 시스템을 지정하는 경우는 두가지로 구분될 수 있다. 그 하나는 테스트하려는 시스템의 메모리에서 데이타를 읽어오거나 쓰는 경우이고 또 하나는 테스트하려는 시스템에서 프로그램을 실시간으로 수행하는 경우이다. 이 두 경우의 차이는 T/$\overline{E}$신호가 '1'인 값이 지속되는 기간이 하나의 버스싸이클인가 아닌가 하는 것이다. 회로의 74LS74/1-2와 74LS123은 위의 두 경우를 구분시켜주는 레지스터의 구실을 한다. T/$\overline{E}$신호를 '1'로 지정하는 것은 74LS74/1-1의 Q를 '1'로 하여 줌으로써 가능한데 74LS74/1-1의 Q가 '1'이 되는 것은 8086 프로세서가 6 byte의 prefetch queue를 갖고있기 때문이다. 테스트하려는 시스템에로의 전환 과정을 타이밍도 상에서 살펴보면 그림 5와 같다.

2) 디버깅을 위한 회로

하드웨어나 소프트웨어의 디버깅은 ICE의 동작목적이 되는 기능이다. 본 연구에서 제작되는 ICE는 디버깅을 위해 트레이스 기능, 브레이크포인트 설정 기능, 테스트하려는 시스템이 프로그램을 실시간으로 수행하는 기능 등이 제작된다. 이러한 기능들의 실현은 앞서 설명된 T/$\overline{E}$신호 생성회로에 부가적인 기능 수행을 위한 회로들이 결합되어 이루어진다.

트레이스기능은 하나의 명령을 수행시켜가며 프로세서의 내부 상태확인과 외부 하드웨어 상태확인을 통하여 프로그램이나 하드웨어의 이상유무를 확인할 수 있도록 하는 기능이다. 트레이기능의 수행은 8086프로세서 내부의 플래그 레지스터를 사용하여 이루어진다. 8086프로세서는 내부 플래그 레지스터의 8번째 bit를 T-bit라 하여 그 값이 '1'이면 하나의 명령어만을 수행한 뒤 내부에서 인터럽트가 발생하도록 되어있다. 이러한 T-bit의 기능에 의해 테스트하려는 시스템의 stack에 트레이스 기능을 수행하고자하는 어드레스와 플래그레지스터를 save 시켜놓은 뒤 save된 플래그 레지스터의 T-bit

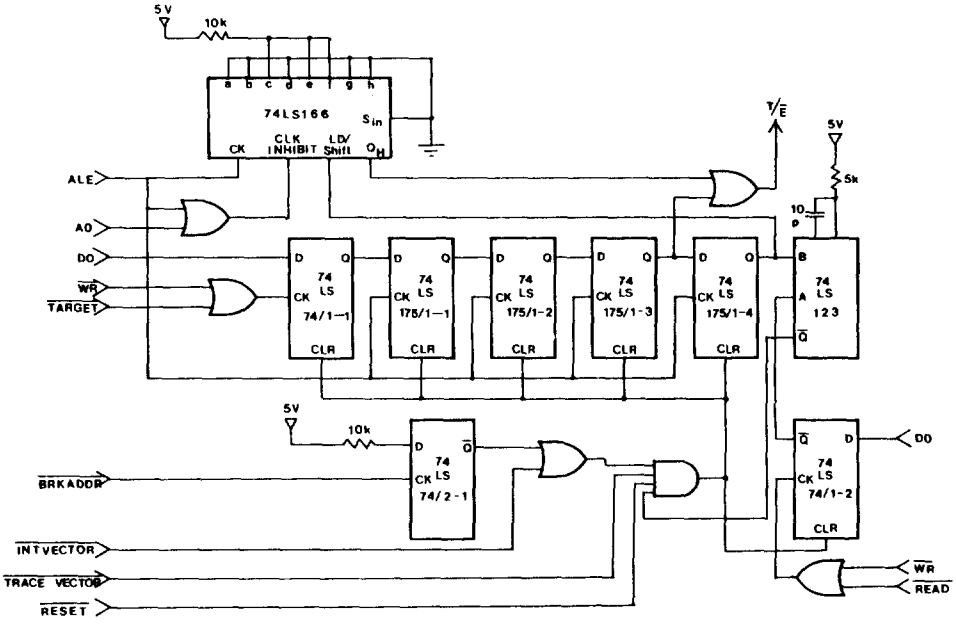

그림 4. 콘트롤 신호 생성회로

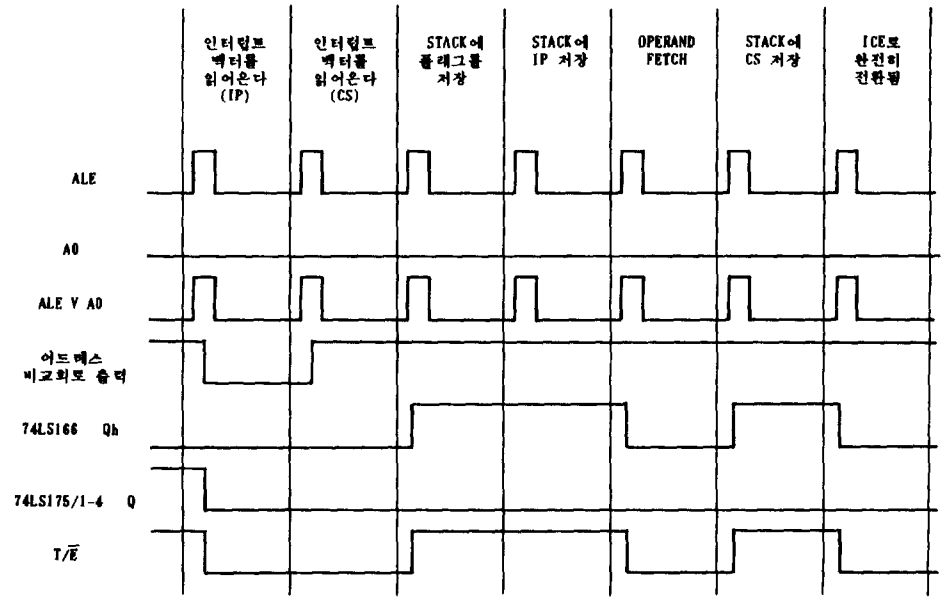

그림 5. 한 버스싸이클동안 테스트프로그램의 메모리를 읽고 쓰는 동작이 일어날 때의 타이밍

를 '1' 로 만들어준 다음 T/$\overline{E}$신호가 '1' 이 되도록 콘트를 레지스터의 값을 조정하고 IRET명령을 수행하게되면 테스트하려는 시스템으로 전환되어 하나의 명령어만을 수행하도록 되는 것이다.

테스트하려는 시스템에서 ICE시스템으로의 전환은 T-bit가 '1' 인 경우에 발생하는 인터럽트의 벡터가 00004H 이고 테스트하려는 시스템의 프로그램을 수행하고 있는 경우에 TRACE VECTOR 출력을 '0'이 되도록 하여 그림 4에서 발생되는 T/$\overline{E}$ 신호가 '0'이 되도록 한다. 그런데 8086프로세서는 인터럽트의 발생시 그 처리가 일반적인 프로세서의 경우와는 다른 형태를 취한다.[5] 즉 일반적인 프로세서의 경우 현재의 플래그 레지스터와 프로그램카운터를 stack에 save한 뒤 그 처리루틴을 지정된 어드레스에서 fetch하여오는 흐름을 갖고 있으나 8086프로세서라 다음과 같이 조금은 복잡한 처리과정을 거친다.

i) 처리루틴의 IP를 fetch
ii) 처리루틴의 CS를 fetch
iii) 프로세서의 PSW를 stack에 save
iv) 프로세서의 CS를 stack에 save
v) 처리루틴의 어드레스에서 수행할 명령어 fetch
vi) 프로세서의 IP를 stack에 save

위의 과정중에서 save하는 과정은 테스트하려는 시스템 내부에서 이루어져야 하고 fetch하는 과정은 ICE 내부에서 이루어져야 한다. 그러므로 위의 과정중 i, ii, v 가 이루어지는 버스 싸이클 동안은 T/$\overline{E}$ 신호가 '0'이어야 하고 그 이외의 경우는 '1', 그리고 과정 vi 이후는 다시 '0'이 되어야 한다. 이러한 일련의 콘트롤 흐름을 위해 그림 4의 74LS166이 사용되었다. 위의 과정들에 대한 여러 콘트롤 신호들의 변화과정은 그림 6에 나타나 있다.

브레이크포인트 설정기능은 특정한 어드레스를 브레이크포인트로 지정한 상태에서 테스트하려는 시스템의 프로그램을 수행시켜 프로그램의 흐름이 미리 지정된 어드레스가 되는 경우 프로그램의 수행을 정지하고 ICE 시스템으로 전환되는 기능을 의미한다. 이 기능이 ICE에서 사용될때 브레이크포인트를 설정하는 어드레스가 RAM 영역인 경우 이외에도 동작되어야 하고 또 prefetch queue의 동작에 의해 어드레스는 브레이크포인트를 가리키고 있더라도 그것은 단순히 명령어를 fetch하는 동작이므로 실제 그 이전의 모든 명령어들이 수행되기까지는 시간적인 차이가 발생하게 된다. 그러므로 본 연구에서는 어드레스 비교방식과 데이타버스의 조작을 통하여 브레이크포인트 기능을 구현하였다. 그림 7은 브레

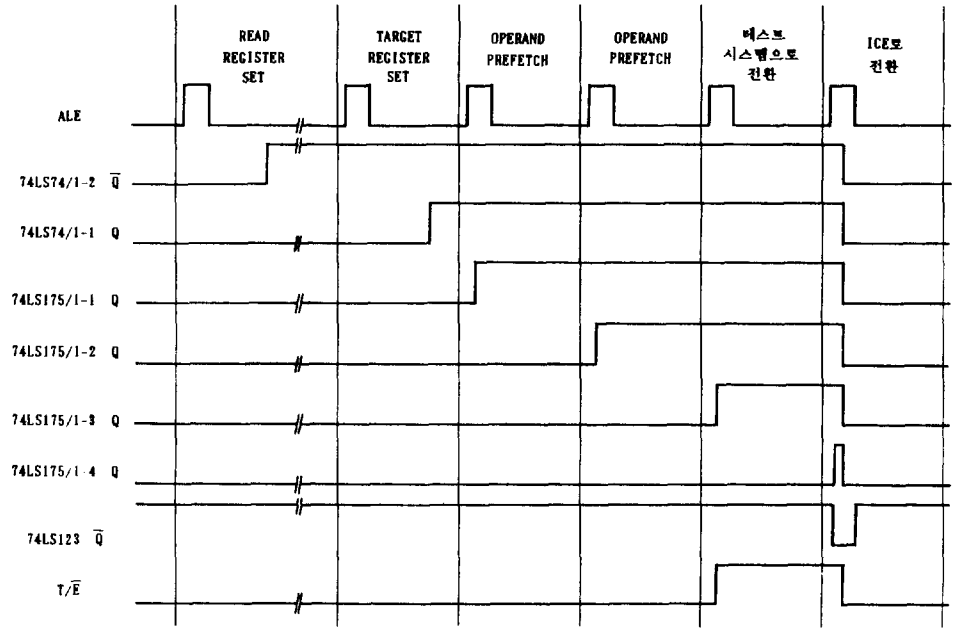

그림 6. 트레이스기능을 수행한 뒤 인터럽트에 의해 ICE로 전환될 때의 타이밍

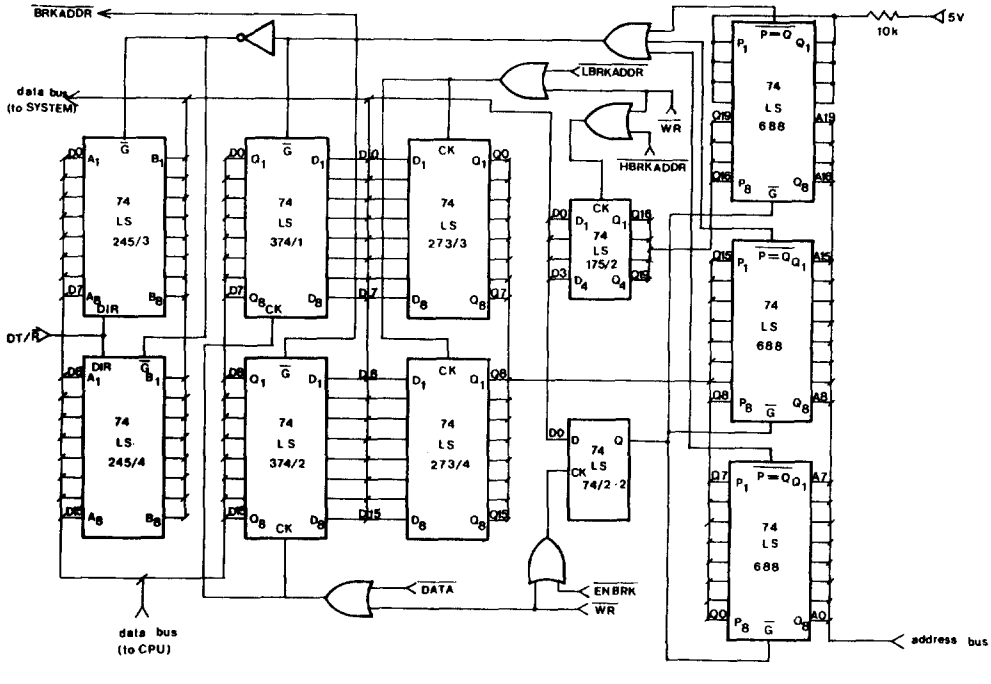

그림 7. 브레이크포인트 설정회로

이크포인트 설정을 위한 회로를 보여준다.

8086프로세서는 74LS175/2, 74LS273/3, 4에 브레이크 어드레스를 기억시켜 그 출력이 8 bit comparator인 S688/ 1, 2, 3의 비교입력 P에 입력되어 어드레스의 값이 브레이크 어드레스와 같은 경우 fetch되는 명령어를 74LS245/3, 4 를 불능상태로 만들어 차단시키고 74LS374에 미리 저장시킨 브레이크포이트 처리 명령어를 프로세서가 대신 읽도록 한다. 대신 프로세서에 읽혀지는 명령어는 1 byte의 길이를 갖으며 소프트웨어 인터럽트의 기능을 하는 'CC' 이다.[6] 브레이크포이트의 어드레스가 even이면 'CC'값은 74LS374/1에 저장되어지고 odd인 경우에는 74LS374/1에는 브레이크포인트 이전에 있는 명령어의 op code나 operand가 저장되고 74LS374/ 2에 'CC'값이 저장된다. 소프트웨어 인터럽트가 발생되면 트레이스기능의 수행때 발생하는 인터럽트와 마찬가지의 수행 과정을 거치게 되는데 이런 과정을 통하여 T/$\overline{E}$신호가 '0'이 되어 ICE시스템으로 전환된다. 단 이 경우 인터럽트벡터의 값이 0000CH 이므로 트레이스기능의 수행과는 별도의 어드레스 비교회로를 갖게된다.

3) 메모리 운용 유니트

메모리 운용 유니트는 ICE자신에 목적으로하는 시스템상에 있는 것과 같은 기능을 하는 64kbyte RAM 영역의 관리및 운용을 담당한다. 이부분의 메모리 영역은 목적으로 하는 시스템에 메모리 영역이 내장되어있지 못한 상황에서 테스트를 하게될 경우 또는 그 영역이 충분하지 못한 경우등에 이용된다. 또 목적으로하는 시스템의 ROM 영역에 내장되어진 프로그램을 수정하여 목적으로 하는 시스템에서 실행시킬 필요가 있는 경우와 ICE에 의한 간단한 테스트용 프로그램의 작성및 그

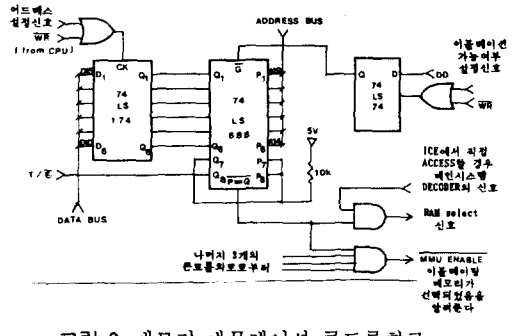

그림 8. 메모리 에뮬레이션 콘트롤회로

수행시에 이용된다. 이 메모리 운용 유니트는 각기 독립된 16kbyte의 영역을 각각 관리한다. 그래서 목적으로 하는 시스템의 각기 다른 4개의 메모리 영역으로 동작할 수 있다. 그림 8은 각각의 에뮬레이팅 메모리 영역에 대해 동작되는 회로를 보여준다.

3. 소프트웨어의 제작

본 연구에서 제작된 ICE가 수행할 수 있는 다음과 같다.

- 트레이스
- 브레이크포인트 설정
- 레지스터와 메모리의 내용검색
- 레지스터와 메모리의 내용수정
- 테스트하려는 시스템에서의 계속적인 명령 수행
- 호스트 시스템으로부터 테스트하려는 시스템을 위한 intellec hex format의 프로그램을 전송받아 지정된 영역에 저장
- 메모리 운용 유니트를 원하는 메모리 영역으로 할당
- 디스어셈블링

위와 같은 기능의 수행을 위하여 소프트웨어적으로 입력된 명령을 해석하고 해석된 명령에 따라 적절한 위치로 분기되도록 하며 또 처리된 결과를 호스트 시스템으로 전송하는 등의 기능을 하는 간단한 형식의 모니터가 작성되었다. 분기된 위치에는 각 기능을 수행을 위한 처리루틴들이 있게 되는데 디스어셈블러를 제외한 과정은 콘트롤회로의 여러 레지스터의 값들을 변환시키는 과정이 주된 처리내용이 된다.

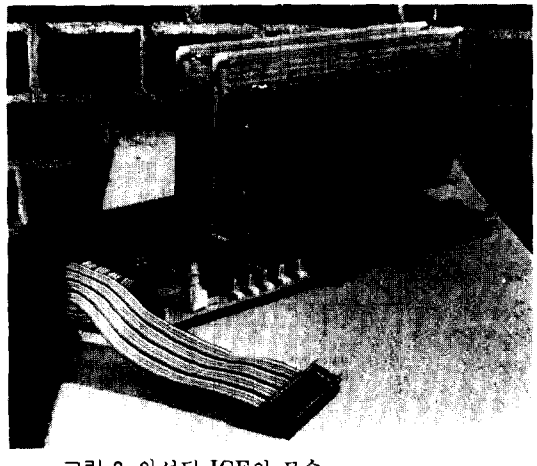

그림 9. 완성된 ICE의 모습

(표 1) 제작된 ICE의 명령 일람표

입 력	기 능
R(CR)	현재의 레지스터상태를 표시
R xx=yyyy(CR)	레지스터 xx에 yyyy를 입력
S xxxx(CR)	현재의 CS에 xxxx의 offset에 있는 테스트 시스템 메모리의 내용을 byte 단위로 바꾼다.
T(CR)	현재의 IP,CS상태에서 테스트 프로그램의 트레이스기능 수행
T xxxx(CR)	현재의 CS에 xxxx를 offset으로하여 테스트 프로그램의 트레이스기능 수행
T xxxx, yyyy(CR)	현재의 CS에 xxxx를 offset으로하여 테스트 프로그램의 트레이스기능을 yyyy번 수행
B xxxxx(CR)	어드레스 xxxxx를 브레이크포인트로 설정
BD(CR)	설정된 브레이크포인트를 해제
D xxxx, yyyy(CR)	현재의 CS에 xxxx를 start offset, yyyy를 end offset으로하여 테스트 내부 메모리의 내용을 화면에 표시
E xxxxx(CR)	에뮬레이팅 메모리의 전 영역에 테스트시스템의 메모리 영역 xxxxx부터 데이타를 복사
F(CR)	현재의 CS에 0000을 start offset으로하여 intellec hex format의 테스트 프로그램을 테스트시스템에 다운로드
F xxxx(CR)	현재의 CS에 xxxx를 start offset으로 하여 intellec hex format의 테스트 프로그램을 테스트시스템에 다운로드
G(CR)	현재의 IP, CS상태하에서 테스트프로그램 수행
G xxxx(CR)	현재의 CS상태에서 xxxx를 offset으로하여 테스트 프로그램 수행
L(CR)	현재의 CS, IP가 지정하는 어드레스부터 10개의 instruction을 디스어셈블링
L xxxx(CR)	현재의 CS에 xxxx를 offset으로하는 어드레스부터 10개의 instruction를 디스어셈블링
L xxxx, yyyy(CR)	현재의 CS에 xxxx를 start offset, yyyy를 end offset으로하여 디스어셈블링
M(CR)	에뮬레이팅 메모리가 설정되어진 상태를 표시
M x, yyyyy(CR)	메모리 에뮬레이션 콘트롤유니트x 를 테스트 시스템의 메모리 공간 yyyy부터(yyyy+16kbyte)까지로 설정
MD x(CR)	메모리 에뮬레이션 콘트롤유니트 x의 설정 해제
MODE(CR)	ICE 모드와 자체 시뮬레이션 모드와의 전환

4. 제작 및 성능 평가

그림 9는 완성된 ICE의 모습을 보여준다. 제작에 사용된 8086프로세서는 8086-2로 최대 8MHz에서 동작될 수 있는 것이다. 하드웨어의 설계에서 언급되었던 대로 모든 기능이 이 8MHz의 조건에 맞도록 제작되어야 하므로 일반적인 74LS 시리즈의 TTL IC들로는 사용이 불가능한 부분들이 많아 거의 모든 콘트롤 신호 관계부분에서 74S 시리즈를 사용하여 지연 시간이 1/3로 감소될 수 있도록 하였다. 단 어드레스비교회로 에서는 74S 시리즈의 비교회로를 제작하는 것 보다는 74LS688이 더 작은 값이 지연시간을 가지므로 유용히 사용되었다. 로직의 단계의 줄이는 과정에서 버퍼링을 필요로 하는 부분들이 많아 fan out 값을 고려한 최소의 버퍼링으로 동작될 수 있도록 하였다. 제작된 ICE에 의한 동작예는 그림 10과 같다. 동작은 6. 67MHz상에서 이루어졌는데 모든 기능이 완벽하게 이루어졌다. 제작된 ICE는 싱글 프로세서모드에서 동작되도록 설계가 되어있어서 멀티 프로세서모드로 설계된 시스템에 대해서는 동작이 이루어질 수 없다. 이점이 현재 제작된 ICE의 가장 큰 제한 조건이 될 것이다. 그러나 싱글 프로세서의 경우에서는 현재의 기능만 가지고도 충분히 개발하려는 시스템의 에뮬레이션을 행할 수 있다고 본다. 현재 시판되고 있는 ICE의 기능과 제작된 ICE를 비교하여볼 때 싱글프로세서모드에서의 차이는 어셈블러, history기능에서만 차이가 발생하고 그 이외의 경우는 제작된 ICE가 모두 구비하고 있기 때문이다. 본 연구에서 제시된 콘트롤회로의 기법이나 트레이스기능, 브레이크포인트 기능들을 이용하여 CPU보드의 설계를 멀티프로세서용으로 전환시킨다면 완벽한 성능의 ICE가 될 수 있으리라고 본다.

```
i8086 IN-CIRCUIT EMULATOR
DEPT. OF CONTROL & INSTRUMENTATION S. N. U.
VERSION 1. 10 1987

MO>F

ready to receive file
start address is 0000 : 0000

receive OK. end address is 0000 : 00EF

>MODE
MO>M 0, 00000

MMU0 : ALLOCATED 00000
MMU1 : ALLOCATED nothing
MMU2 : ALLOCATED nothing
MMU3 : ALLOCATED nothing

MO>M 1, 00000

MMU0 : ALLOCATED 00000
MMU1 : ALLOCATED 10000
MMU2 : ALLOCATED nothing
MMU3 : ALLOCATED nothing

MO>R

AX=0000 BX=0000 CX=0000 DX=0000 BP=0000 SP=0000 SI=0000
DI=0000 CS=0000 DS=0000 ES=0000 SS=0000 PC=0000 SW=0000

MO>L 0050

0000 : 0050 B80010 :          MOV     AX, 1000
0000 : 0053 8ED8              MOV     DS, AX
0000 : 0055 8ED0              MOV     SS, AX
0000 : 0057 C70600019090      MOV     [0100], 9090
0000 : 005D BC0003            MOV     SP, 0300
0000 : 0060 9C                PUSHF
0000 : 0061 58                POP     AX
0000 : 0062 0D0001            OR      AX, 0100
0000 : 0065 50                PUSH    AX
0000 : 0066 B80000            MOV     AX, 0000

MO>R SP=1000

AX=0000 BX=0000 CX=0000 DX=0000 BP=0000 SP=1000 SI=0000
DI=0000 CS=0000 DS=0000 ES=0000 SS=0000 PC=0000 SW=0000

MO>T 0050

AX=1000 BX=0000 CX=0000 DX=0000 BP=0000 SP=1000 SI=0000
DI=0000 CS=0000 DS=0000 ES=0000 SS=0000 PC=0053 SW=F002

MO>B

  BREAK ADDRESS : nothing

MO>B 00060

  BREAK ADDRESS : 00060

MO>R CS=1000

AX=1000 BX=0000 CX=0000 DX=0000 BP=0000 SP=000 SW=0000
DI=0000 CS=1000 DS=0000 ES=0000 SS=0000 PC=0053 SI=F002

MO>D 0100, 0110

1000 : 0100 F6 38 F0 21 2D 85 54 8D 46 06 70 40 06 00 D0 68
1000 : 0110 05 05

MO>R CS=0000

AX=1000 BX=0000 CX=0000 DX=0000 BP=0000 SP=1000 SI=0000
DI=0000 CS=0000 DS=0000 ES=0000 SS=0000 PC=0053 SW=F002

MO>G 0053

AX=1000 BX=0000 CX=0000 DX=0000 BP=0000 SP=0300 SI=0000
DI=0000 CS=0000 DS=1000 ES=0000 SS=1000 PC=0060 SW=F002

MO>R CS=1000

AX=1000 BX=0000 CX=0000 DX=0000 BP=0000 SP=0300 SI=0000
DI=0000 CS=1000 DS=1000 ES=0000 SS=1000 PC=0060 SW=F002
```

그림 10. ICE의 동작예

Ⅲ. 결 론

8086프로세서용 ICE를 제작함으로써 ICE에서 사용되는 기본적인 콘트롤방식과 그 사용에 필요로하는 기능들을 이해할 수 있었다. 이는 다른 종류의 프로세서에 적용될 수 있는 것이 대부분이어서 반도체 제작기술의 발달로 인하여 점점 다양화되고 있는 각각의 프로세서들을 위한 ICE의 제작을 용이하게 해줄 수 있으리라 믿는다. 또 이 8086프로세서용 ICE를 제작함으로써 IBM PC와 결합하여 간이형 MDS를 구성할 수 있기 때문에 MDS의 필요성은 인식하면서도 시스템의 개발자가 쉽게 접할 수 없었던 점을 극복하고 8086프로세서를 베이스로 하는 시스템의 개발에 널리 이용될 수 있는 개발시스템을 제시하여준 데 본 연구의 의미가 있다고 하겠다.

참 고 문 헌

1) Duglas Lundin & Michael Crovitz, "Triple- threat instrument debugs up- based systems", Electronic Dlesign vol. 32 no. 3, p 127,Feb 1984

2) Jonah McLeod, "μC development have the hardware, software 16-bit μPs needs.", Electronic Design vol. 27 no. 19, p 94, Sep 1979

3) Bruce E. Goldstone, "Comparing Microcomputer Development System Capabilities", Computer Design, vol. 18 no. 2, pp. 87-90, Feb 1979

4) John Uffenbeck, The 8086 / 8088 Family, Prentice Hall, pp. 29-32 1987

5) Russell Rector & George Alevy, The 8086 Book, OSBORNE / McGraw-Hill, p 8. 36 1980,

6) James W. Coffron, Programming the 8086 / 8088, Sybex, pp.203-205 1983,