VME-BSA (VME Bus State Analyzer) 개발에 관한 연구

○신 상석, 김 용연, 안 희일, 윤 용호, 오 길록
한국전자통신연구소 컴퓨터개발부

A Study on Development of VME-BSA (VME Bus State Analyzer)

S.S. Shin, Y.Y. Kim, H.I. Ahn, Y.H. Yoon, G.R. Oh
ETRI Computer Technology Department

Abstract

This paper describes of VME-BSA which is a tool for the development, maintenance and repair. In micro/mini computer system using VMEbus as a backplane bus, VME-BSA has some good facilities such as acquiring, storing and analyzering the information (address, data, etc.) on VMEbus according to the various condition which is set by users, and therefore it is easy to isolate and find many complete errors on bus.

1. 개 요

본 고는 VMEbus 상에 발생하는 버스 사이클 정보 (어드레스, 데이타 등등) 를 여러가지 조건으로 저장하고 분석하는 기능을 가지므로써 VMEbus를 백플레인 버스로 사용하는 시스템의 개발, 유지, 보수에 유용하게 이용될 수 있는 tool인 VME-BSA의 개발에 관한 것이다.

백플레인 버스를 사용하는 시스템에서 발생하는 에러의 요인은 각 모듈 자체의 H/W 에러, 프로그램 에러, 버스 인터페이스 에러, 버스 프로토콜 에러, 주위 환경의 갑음에 의한 에러 등 여러가지가 있는데, 시스템의 인테그레이션이나 시스템 사용 중에 발생하는 에러는 대부분 매우 복잡하고 예측하기가 어려워 이의 추출을 위한 적절한 장치의 개발이 필요로 된다. 버스 상의 모듈은 모두 버스를 통하여 모든 정보를 교환하므로 이러한 정보를 저장, 분석할 수 있는 장치가 있다면 위에서 언급한 에러의 발생 위치나 원인에 대하여 자세한 정보를 얻을 수 있다.

VME-BSA는 VMEbus 상에 발생하는 여러가지 정보 (마스터, 어드레스, 데이타, 인터럽트, R/W 등등) 를 사용자가 원하는 다양한 조건으로 저장하고 분석할 수 있는 기능을 가진 장치로서, VMEbus를 사용하는 시스템의 에러 검출 및 발생 원인 규명에 많은 도움을 줄 수 있다.

VME-BSA는 VMEbus 상에 하나의 슬롯을 차지하며, master나 slave의 역할은 전혀 담당하지 않고 단지 observer의 역할만을 수행하기 때문에 실제의 버스 operation에는 영향을 미치지 않는다. 사용자는 이 모듈에 연결된 콘솔을 통하여 여러가지 필요한 명령을 수행할 수 있는데, ASCII 코드를 사용하는 모든 터미날은 이 모듈의 콘솔로 연결될 수 있도록 설계되었으며, 명령 수행 방법은 명령어 입력 방식과 메뉴 선택 방식을 적절히 조합함으로써 사용자에게 편의를 제공하도록 하였다.

2. 특 징

- o VMEbus 전용
- o MC68000 마이크로프로세서 이용에 따른 우수한 성능
- o 다양한 트리거 조건
- o 어드레스와 데이타의 바이트 단위 don't care 트리거 제공
- o 96 채널 2K 버스 사이클의 VMEbus 정보 트레이스 메모리
- o 빠른 버스 사이클 트레이스 (최저 사이클 시간 = 70ns)
- o 버스 에러 검출
- o 일반적인 ASCII 터미날로 콘솔 사용
- o 사용자 인터페이스의 편리 (간단한 명령어, 문답식 처리)

3. 블럭 다이어그램

설계된 VME-BSA의 하드웨어 블럭 다이어그램은 그림 3-1과 같다.

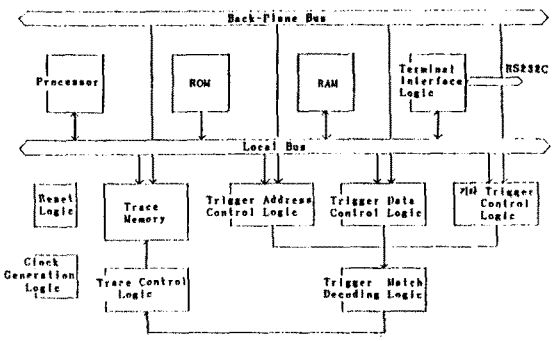

그림 3-1 VME-BSA의 하드웨어 블럭 다이어그램

가. 리소스 (Resource) 분석

여기서는 VME-BSA의 하드웨어 블럭 다이어그램을 각 리소스별로 나누어 기능상의 특성 및 설계 기법에 관해 기술하고자 한다.

o 프로세서

VME-BSA의 프로세서로는 24 비트 어드레스, 16 비트 데이타를 갖는 MC68000-12 프로세서를 사용하였다.

o 롬 & 램 (ROM & RAM)

벡터 테이블과 모니터 프로그램을 내장하기 위한 롬으로는 32K x 8 비트 EPROM (MBL27256) 2개를 사용하여 64K 바이트의 용량을 가지도록 하였으며, 시스템 스택 (system stack) 과 정보 저장 버퍼 기능을 갖는 램으로는 2K x 8 비트 CMOS SRAM (V61C16-55) 2개를 사용하여 4K 바이트의 용량을 가지도록 하였다.

o 트레이스 메모리

버스 트레이스 정보를 저장하기 위한 트레이스 메모리는 2K x 8 비트 CMOS SRAM (V61C16-55) 12개를 사용하여 2K x 96 비트의 용량을 가지도록 하였다. read/write 모드는 프로세서 모드와 버스 모드로 구분되며 리드 사이클인 프로세서 모드는 프로세서 리드 타이밍에 그리고 라이트 사이클인 버스 모드는 데이타 store time (DTACK*) 에 의존하도록 설계하였다. VMEbus spec. 상에 가능한 최소 블럭 사이클은 약 70nsec의 사이클 타임을 만족해야 하므로 버스 모드시 라이트 사이클 타이밍은 32MHz 클럭을 사용하여 설계하였다.

o 콤페어 로직 (Compare Logic)

콤페어 로직은 레지스터 (트리거 어드레스 레지스터, 트리거 데이타 레지스터, 어드레스 모디파이어 레지스터, 트리거 콘트롤 레지스터, 채너럴 콘트롤 레지스터, 카운터/스테이터스 레지스터, LED 레지스터) 들과 콤퍼레이터 그리고 매치 대코딩 PAL로 구성되어 있다.

- 트리거 어드레스 & 데이타 레지스터
  트리거 어드레스 레지스터와 데이타 레지스터는 32 비트 write only 레지스터로 8 비트 콤퍼레이터를 트리거 콘트롤 레지스터와 연결하여 boundary value 측정이 가능하도록 설계하였다.

- 트리거 어드레스 모디파이어 레지스터 (AM* Register)
  트리거 어드레스 모디파이어 레지스터는 8 비트 write only 레지스터이며 콤퍼레이터와의 매치 콘디션 테이블 (match condition table) 은 어드레스 레지스터와 동일하다.

- 트리거 콘트롤 레지스터
  트리거 콘트롤 레지스터는 8 비트 write only 레지스터이며 어드레스와 데이타의 매치 콘디션을 세트하기 위한 레지스터이다. 각 비트들에 대한 정의는 표 3-1과 같으며, high enable이다.

표 3-1 트리거 콘트롤 레지스터 테이블

- 비트 7 : Data Enable 3
- 비트 6 : Data Enable 2
- 비트 5 : Data enable 1
- 비트 4 : Data enable 0
- 비트 3 : Address Enable 3
- 비트 2 : Address Enable 2
- 비트 1 : Address Enable 1
- 비트 0 : Address Enable 0

- 제너럴 콘트롤 레지스터
  제너럴 콘트롤 레지스터는 16 비트 write only 레지스터이며, 시스템 동작의 매치 콘디션과 기능들을 세트하기 위한 레지스터로서 각 비트에 대한 정의는 표 3-2와 같다.

표 3-2 제너럴 콘트롤 레지스터 테이블

- 비트 15 : Bus Grant Enable
- 비트 14 : Data Type Selection 1
- 비트 13 : Data Type Selection 0
- 비트 12 : Mode Selection
- 비트 11 : Trace Type Selection 1
- 비트 10 : Trace Type Selection 0
- 비트 9 : Interrupt Acknowledge Enable 1
- 비트 8 : Interrupt Acknowledge Enable 0
- 비트 7 : Master Reset
- 비트 6 : Write Enable 1
- 비트 5 : Write Enable 0
- 비트 4 : Bus Grant Enable 3
- 비트 3 : Bus Grant Enable 2
- 비트 2 : Bus Grant Enable 1
- 비트 1 : Bus Grant Enable 0
- 비트 0 : Address Modifier Enable

- 카운터 & 스테이터스 레지스터
  카운터 레지스터는 write 모드 동작 종료시 어드레스 카운터의 위치를 지정하여 주기위한 레지스터이며, 스테이터스 레지스터는 write 동작 종료와 버스 에러 (BERR*) 의 여부를 저장하기 위한 레지스터이다. 또한 각 레지스터 비트에 대한 정의는 표 3-3과 같다.

표 3-3 카운터 & 스테이터스 레지스터 테이블

- 비트 13 : Bus Error
- 비트 12 : Stop Bit
- 비트 11-0 : Address Counter 0-11

- 매치 PAL 로직 (Match PAL Logic)
  매치 PAL 로직은 비교해야할 각 소스들에 대하여 사용자가 원하는 각 경우의 수를 만족시키기 위해 프로그램된 로직 어레이 (logic array) 로서 PAL-20L8B를 사용하여 설계하였다.

o 카운터 로직
  카운터 로직은 트레이스 카운터와 어드레스 카운터 그리고 먹스 (mux) 로 구성되어 있다. 여기서 트레이스 카운터는 버스 모드시 트레이스 타입을 결정하기 위한 역할을 하며, 어드레스 카운터는 트레이스 메모리의 어드레스를 지정하는 역할을 하도록 설계되었다.

- 트레이스 카운터
  트레이스 카운터는 버스상의 데이타와 레지스터 상의 데이타가 매치된 후 트레이스 타입 (스타트 모드, 미들 모드, 엔드 모드)를 결정하기 위한 로직이다. 여기서 한번 매치가 이루어 지면 트레이스 타입이 완료될 때까지는 다른 매치에 영향을 받지 않도록 설계되었으며, 타입 선택은 제너럴 콘트롤 레지스터에 의하여 직접 조절된다.

- 어드레스 카운터
  어드레스 카운터는 트레이스 메모리의 어드레스를 지정하여 주기위한 카운터로서 트레이스 타입이 완료되면 현재의 어드레스 포인터 (address pointer) 의 위치를 알려주기 위한 카운터 레지스터를 포함하고 있다. 또한 카운터의 어드레스는 store time에 의하여 증가된다.

o 어드레스 PAL (Address PAL)
  어드레스 PAL은 VME-BSA 내의 모든 리소스들을 어드레스 맵 (address map) 에 알맞게 제공할 수 있도록 프로그램된 로직이다. 또한 어드레스 데코딩 PAL은 프로세서 모드에서만 동작할 수 있도록 PAL16L8A로 설계하였으며 DTACK* 생성 로직을 포함하고 있다. 각 소스들에 대한 상세한 어드레스 맵은 그림 3-2와 같다.

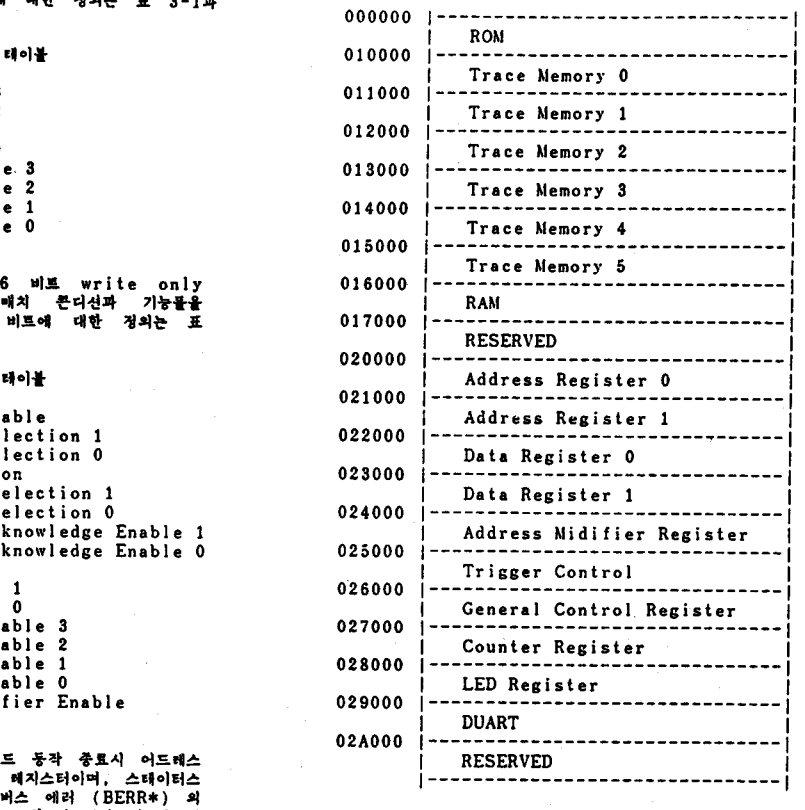

그림 3-2 어드레스 맵

o 콘트롤 로직
  콘트롤 로직의 대부분은 제너럴 콘트롤 레지스터와 트리거 콘트롤 레지스터에 의하여 제공되며 주된 기능은 아래와 같다.

- 트레이스 모드 (write/read) 결정
- 트레이스 타입 결정
- master reset 결정
- trigger qualifier 결정
- 어드레스, 데이타, AM* 매치 콘디션 결정

o 기타 주변 로직

- reset 로직
  VME-BSA 의 reset 로직은 power on reset과 local reset의 두가지로 구성되어 있다. power on reset은 power on과 동시에 자동적으로 시스템이 reset되며 보드내의 모든 리소스들 (resources) 이 clear 된다. 또한 local reset은 제너럴 콘트롤 레지스터의 master reset 비트를 세트함에 의하여 자체 보드만을 reset 할 수있다.

- 클럭 생성 로직
  VME-BSA 에서는 32MHz의 크리스탈 (crystal) 을 분주하여 사용하며 프로세서로는 16MHz의 클럭을 공급하고, 버스 모드에서의 write 타이밍 생성 로직에 32MHz의 클럭을 이용한다.

o I/O 인터페이스 로직

I/O 인터페이스 로직은 DUART (MC68681), 터미널 인터페이스 버퍼, 포트 (port) 등으로 구성되어 있다. DUART(dual asynchronous receiver & transmitter) 는 MC68000 프로세서와 직접 인터페이스할 수 있는 칩이다. 여기서는 2개의 I/O 포트를 제공하며 디바이스는 RS232C 프로토콜를 통해 데이타가 전송된다.

나. 중요 하드웨어에 대한 상세한 설명

1) 빠른 버스 사이클 정보 저장을 위한 더블 쉬프터 로직

설계된 더블 쉬프터 로직에 power가 공급되면 플리플롭과 두개의 쉬프터가 clear되고, 이때 버스 상의 정보가 유용한 정보임을 나타내는 버스 어크놀리지 (bus acknowledge) 신호가 플리플롭에 세트되면 두개의 쉬프터 로직이 동작하기 시작한다. 쉬프터1과 쉬프터2는 서로 반대의 클럭 에지 (clock edge) 즉, 32MHz 클럭은 쉬프터1으로 인버팅된 클럭은 쉬프터2로 각각 공급되어 쉬프터1에 비해 32 메가 쉬프터된 신호가 쉬프터2에서 발생한다.

이렇게 발생된 신호를 이용하여 트레이스 메모리 write enable 신호와 트레이스 메모리의 칩 선택 신호를 만들면, 로직이 단순해질 뿐아니라 1개의 쉬프터를 사용할 때보다는 정보 저장이 가능한 최소 버스 사이클 시간 (minimum bus cycle time) 을 클럭 사이클의 최대 1/2에 해당하는 시간 만큼 줄일 수 있다.

2) 래지스터와 PLA를 사용하여 단순화한 트리거 콘트롤 로직

설계된 트리거 콘트롤 로직에 power가 공급되면 채너럴 콘트롤 래지스터가 clear되고, 이때 다양한 트리거 조건을 채너럴 콘트롤 래지스터에 세팅하여 그 값과 실제 버스 정보를 약간의 콤비네이션 로직을 포함한 PLA로 구성하면 다양하면서도 단순한 트리거 콘트롤 로직을 구현할 수 있다. PLA의 입력 중 어드레스/데이타 매치 신호들은 don't care condition을 포함한 어드레스/데이타 콤페어 로직으로 부터 발생한 신호이며, 어드레스 모드 화이어 아울 (AM* out) 신호는 다양한 어드레스 타입을 결정하기 위한 로직으로 부터 발생한 신호이다. 또한 매치 시그널을 인버트한 시그널은 일단 매치가 발생하면 그 타입이 완료될 때까지 다른 외부의 영향을 받지 않고 매치 상태를 유지시켜주는 시그널이고, 어드레스 카운터 디스에이블 시그널은 이러한 매치 상태를 풀어주는 시그널이다.

VME-BSA 보드에서는 다양한 조건과 경우를 만족하기 위해 복잡해지기 쉬운 트리거 콘트롤 로직을 트리거 콘트롤 PLA 를 이용하여 프로그램 하므로서 더욱 기능을 강화하였을 뿐아니라 회로를 만순화 하였다.

3) Don't Care Condition 을 포함한 어드레스,데이타 콤페어 로직

어드레스 콤페어 로직에서는 이미 저장된 어드레스 래지스터 값과 버스상의 실제 어드레스 값을 비교하여 트리거 콘트롤 로직으로 보내므로서 원하는 정보를 얻을 수 있다. 이때 각 콤퍼레이터는 어드레스를 바이트 단위로 비교하므로서 다양한 타입 (short, standard, extended) 으로 각각의 어드레스 값을 비교할 수 있을 뿐아니라, 각 타입에서 바이트 단위의 don't care condition 비교가 가능하다. 데이타 콤페어 로직에서는 어드레스 콤페어 로직에서 제공하는 기능외에 실제 버스상의 데이타 타입 (Byte, Word, Lword) 을 단순화된 PLA 로직으로 처리하므로서 더욱 다양한 기능을 제공할 수 있다. 데이타 콤페어 로직의 PAL 입력중 데이타 타입 0-1 은 사용자가 원하는 버스상의 데이타 타입을 결정하기 위한 신호이고, Longword, DS0-1, Address 1 은 실제 버스상의 데이타 타입을 나타내는 신호이다.

또한 데이타 enable 0-3 은 버스상의 데이타 타입에 무관하게 사용자가 비교하고저 하는 데이타 타입을 결정하기 위한 신호이다. 이러한 신호들이 데이타 콤페어 타입 인에이블 PAL을 통하여 프로그램된 후,' don't care condition을 제공하기 위하여 각기 다른 난드 게이트 (NAND gate) 로 보내진다. 이렇게 보내진 신호들은 두개씩 앤드 게이트 (AND gate) 로 조합되어 트리거 콘트롤 PAL의 입력으로 보내지므로서 사용자가 이와같이 다양한 조건들을 적절히 조합하면 원하는 거의 모든 타입의 어드레스 및 데이타 조건을 얻을 수 있다.

4) 두개의 카운터를 이용한 트레이스 타입 및 어드레스 결정 로직

트레이스 타입 및 어드레스 결정 로직에 파우어가 공급되면 이벤트 카운터와 어드레스 카운터가 clear된다. 여기서 시스템 동작중 트리거 콘트롤 로직으로 부터 매치가 발생하면 매치 아울 (match out) 입력으로 매치 신호가 들어오게 되어 이벤트 카운터가 동작을 시작한다. 동작을 시작한 이벤트 카운터는 매치 후 0, 1K, 2K 의 세가지 경우 (start, middle, end type) 의 신호를 타입결정 먹스로 내보낸다. 타입결정 먹스에서는 채너럴 콘트롤 래지스터로 부터 제공된 트레이스 타입 선택 비트에 의해 트레이스 타입이 결정되는데, 이는 트리거 조건을 만족하는 버스 사이클의 전후 사이클 범위를 자유롭게 선택할 수 있는 모드 선택 비트와 ORing되어 버스 트레이스에 필요한 로직을 disable시키는 한편, 어드레스 카운터의 동작을 정지시킨다.

이렇게 정지된 어드레스 카운터는 버스 모드일 때에만 버퍼를 통과하여, 트레이스 메모리의 어드레스 카운터는 마지막으로 트레이스된 트레이스 메모리 어드레스 값을 갖게된다. 위와 같은 정상적인 절차에 의하여 버스 트레이스가 멈추는 경우외에, 버스 트레이스 중에 사용자가 임의로 동작을 멈출 수 있는 기능을 제공하기 위하여 콘트롤 래지스터의 모드 선택 비트를 포함시켰다.

다. 전체적인 기능 설명

사용자가 본 발명 장치 보드에 연결된 터미널을 통하여 트리거 조건을 입력하면 보드 상의 프로세서는 롬에 내장되어 있는 모니터 프로그램을 수행하여, 필요로 되는 래지스터에 알맞은 트리거 조건을 저장한다. (버스 상의 정보 중 사용자가 필요로 하는 기준 버스 사이클의 정보를 트리거 조건이라 한다 : 어드레스, 데이타, 데이타 사이즈 타입 등등) 이 조건 중 어드레스와 데이타는 각 바이트 마다 그 값을 무시할 수 있도록 약간의 하드웨어가 어드레스/데이타 콤페어 로직에 첨가되어 있다.

계속해서 버스 저장 정보를 명령하는 입력이 들어 오면 버스상의 정보를 트레이스 메모리에 저장하기 시작하는데, 이때 트레이스 메모리를 적절히 제어하는 부분이 트레이스 메모리 콘트롤 로직이다. 버스상의 어드레스와 데이타 등은 어드레스/데이타 비교 로직으로, 이를 거친 결과와 버스상의 다른 정보인 read/write, 버스 마스터, 데이타 사이즈 타입 등등은 이미 지정된 트리거 조건과 비교하기 위하여 트리거 콘트롤 로직을 거치게 되고, 여기에서 주어진 조건이 만족되면 매치 신호가 발생하여 매치 플리플롭을 세트하게 된다.

이 신호가 발생하면 이벤트 카운터가 동작하기 시작하여, 트리거 조건 만족 사이클로 부터 연속되는 버스 사이클을 카운트하게 되고, 알맞은 수의 카운트 값이 타입 선택 로직을 거쳐 선택되면 이것이 버스로 부터의 데이타 저장을 멈추게 하는 신호가 된다. 이때까지 저장된 트레이스 메모리 정보는 다시 프로세서의 모니터 프로그램에 의하여 분석되어 그 결과가 사용자 터미날로 보내지게 되는데, 그 다음은 초기 상태와 같다.

4. 모니터 프로그램 설계

VME-BSA 모니터 프로그램은 C 언어로 짜여진 후 68000 어셈블리 언어로 컴파일되었으며, 그 크기는 오브젝트 코드로 25 Kbyte 정도이다.

가. 프로그램의 구성

기능상 크게 6부분으로 나누어 설계하였으며 그 구성 및 전체적인 흐름도는 그림 4-1과 같다. VME-BSA가 처음으로 power on 상태가 될 때, 또는 power on 상태에서 reset-push-botton이 눌러지면 초기화 프로그램 (Initialization Routine) 의 수행이 시작되고, 이것이 끝나면 메인 프로그램 (Main Routine) 이 수행된다.

메인 프로그램 (Main Routine) 은 loop를 돌면서, 터미날로부터의 입력이 있는지의 여부 및 이전에 버스로부터의 정보 저장 명령이 있었던 경우, 정보 저장이 끝났는지의 여부를 조사하여 전자의 경우가 발생하였을 때는 입력 처리 프로그램 (Input Routine), 후자의 경우는 트레이스 처리 프로그램 (Trace Routine) 을 수행한다. 위의 수행이 끝나면 다시 메인 프로그램 (Main Routine) 으로 돌아가 앞에서 설명한 기능을 반복 수행한다. 앞에서 언급된 모든 프로그램은 터미날 입출력을 위하여 궁통적으로 입출력 프로그램 (I/O routine) 을 필요에 따라 수행하며, 한편 모든 프로그램의 수행시 VME-BSA 자체에서 발생한 에러는 에러 처리 프로그램 (Error Routine) 에서 처리한다.

- 초기화 프로그램 (Initialization Routine)

먼저 터미날 인터페이스를 위하여 DUART를 알맞게 초기화하고 시작 메세지를 내보낸 후, ROM read 테스트 및 RAM read/write 테스트를 실시한다. 에러 발생 시는 에러 발생 어드레스를, 그렇지 않은 경우는 테스트 완료 메세지를 터미날에 나타낸 후 여러가지 필요로 되는 초기값을 세트하고 prompt를 나타낸다. 수행 후 메인 프로그램으로 넘어 간다.

- 메인 프로그램 (Main Routine)

터미날로부터의 입력이 있는지의 여부를 조사하고, 입력이 있으면 입력 처리 프로그램을 수행한 후 다시 돌아온다. 입력이 없는 경우나 입력 처리가 끝나면 계속해서 현재의 모드를 조사하고, 트레이스 모드가 아닌 경우 또는 트레이스 모드이나 트레이스가 아직 끝나지 않은 경우는 다시 처음부터 메인 프로그램을 수행하고, 그렇지 않은 경우는 트레이스 처리 프로그램을 수행한 후 메인 프로그램의 처음으로 간다.

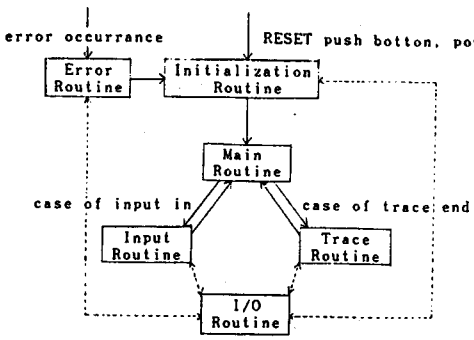

그림 4-1 모니터 프로그램의 구성 및 흐름도

- 입력 처리 프로그램 (Input Routine)
  터미날로부터의 입력을 처리하는 부분으로 입력 당시의 모드에 따라 각각에 알맞은 조치를 취하고 메인 프로그램으로 돌아간다. 전체 프로그램의 대부분을 차지하며 일반적인 처리 법칙은 다음과 같다.
  - o 현재의 모드에서 유용한 입력은 저장하면서 그 의미를 터미날에 나타내고, 유용하지 않은 입력은 무시하면서 벨을 울린다.
  - o <ESC> 키가 입력되면 현재의 모드에 관계없이 진행되는 수행을 멈추고, prompt를 내보내면서 초기 상태로 돌아간다.
  - o <CR> 키가 입력되면 현재까지 저장된 정보를 수행한다.
  - o <BS> 키가 입력되면 바로 전에 입력된 정보가 무시되고, 바로 전 상태로 돌아간다.

- 트레이스 처리 프로그램 (Trace Routine)
  트레이스 타입과 트리거 포인트 및 그동안에 저장된 VMEbus의 정보를 포맷에 맞게 터미날에 나타낸다. 처음 나타내는 정보는 트리거 포인트를 포함한 16 사이클이고 그 이전이나 이후의 사이클에 관한 정보는 사용자의 입력에 따라 원하는 장소의 16 사이클을 나타내 준다.

- 입출력 프로그램 (I/O Routine)
  터미날의 입출력을 담당하는 프로그램으로 한개의 입출력이나, 연속된 여러개의 출력을 처리한다.

- 에러 처리 프로그램 (Error Routine)
  동작 중에 나타나는 에러를 처리하여 주는 프로그램으로 에러가 발생하였음을 터미날에 나타내고 초기 상태로 돌아간다.

나. 초기 상태

default로 지정되어 있는 트리거 조건의 초기 상태는 아래와 같다.

- o trigger address enable
- o trigger address value = 0xffffffff
- o trigger address type = extended
- o trigger data disable
- o trigger AM disable
- o trigger master disable
- o trigger R/W type = any access
- o trigger cycle type = any cycle
- o trigger trace type = end type

다. 명령어

prompt가 나타난 상태에서 콘솔로부터 입력된 데이타는 명령어로 처리된다. 사용 가능한 명령어는 모두 5개로 각각 1개의 데이타로 이루어져 있으며, 이 명령어를 입력하면 사용자의 편의를 위하여 명령어의 전체 이름을 모니터 프로그램에서 터미날로 디스 플레이하여 준다. 설계된 명령어에 대한 설명을 아래에 나타내었다.

o <H> : help for VME-BSA commands

처음 시작하는 사용자의 편의를 위하여 사용 가능한 명령어의 종류 및 간단한 설명을 디스 플레이하여 주는 명령어이다.

o <T> : trigger condition set

트레이스의 기준점이 되는 트리거 조건을 정하는 명령어이다. VME-BSA는 <G> 명령어 수행시 트리거 조건에 의해 한정된 버스 데이타를 비교하며, 조건이 맞으면 트리거된다. 사용자가 정할 수 있는 여러가지 트리거 조건은 변화의 여지가 많은 조건부터 하나씩 문답식으로 결정하는 방법을 택하였으며 여기서 결정되는 트리거 조건은 아래와 같다.

- AM 비교 여부 및 해당 값
- 어드레스 비교 여부, 타입 (short, standard, extended) 및 해당 값
- 데이타 비교 여부, 사이클 타입 (byte, word, longword, don't care) 및 해당 값
- R/W 타입 (read, write, any)
- 마스터 비교 여부 및 해당 마스터 번호
- 사이클 타입 (normal, interrupt, any)
- 트레이스 타입 (start, middle, end)

o <S> : status 디스 플레이

현재의 트리거 조건의 상태를 나타내는 명령어이다. 디스 플레이 되는 값 중, disable 되어 있는 트리거 조건은 "don't care"의 메세지로 나타난다.

o <G> : go & start trace

VMEbus로부터 버스 사이클을 catch하여 트레이스 정보로 저장하기 시작하는 명령어이다. 지정된 트리거 조건과 트레이스 조건이 만족되면 트레이스를 멈추게 된다. 한편, 트레이스 중에 사용자가 <ESC>를 입력했을 때는 즉시 트레이스를 멈춘다. 트레이스가 멈추면, 트레이스 타입과 트리거 포인트 및 그동안에 저장된 VMEbus의 정보를 포맷에 맞게 터미날에 나타낸다. 처음 나타내는 정보는 트리거 포인트를 포함한 16 사이클이고 그 이전이나 이후의 사이클에 관한 정보는 사용자의 입력에 따라 원하는 장소의 16 사이클을 나타내 준다. 트레이스되는 정보에 관한 설명은 다음과 같다.

```
NUM    : 트레이스 사이클의 순서 (0x7ff가 항상 마지막
         트레이스 사이클)
MST    : 현재 사이클의 master (BG3* - BG0*,
         0,1,2,3 중에 하나) 두개 이상 발생하거나
         하나도 없으면 "?"가 디스 플레이 됨
CYC    : 사이클의 타입 (IACK*, I = interrupt,
         N = normal cycle)
R/W    : R/W 타입 (WRITE*, R = read,
         W = write)
AM     : AM 값 (AM5 - AM0)
ADDR   : 어드레스 값 (A31 - A00,
         A00는 LD*, DS1*, DS0*의 조합 값)
DATA   : 데이타 값 (D31 - D00)
BRQ*   : bus request (BRQ3* - BRQ0*)
IRQ*   : interrupt request (IRQ7* - IRQ1*)
DTYPE  : 데이타 transfer 사이클 타입
         (LWORD*, A1, DS1*, DS0*)
         LONG  = longword cycle
         WORD  = word cycle
         BYTE  = byte cycle
         UA0-2 = unaligned 0-2 byte cycle
         UA1-3 = unaligned 1-3 byte cycle
         UA1-2 = unaligned 1-2 byte cycle
         ?0011 ?0101 ?0110 ?0111 ?1011 ?1111
               = undefined cycle
BERR   : bus error (BERR*, YES = error,
         NO = no error)
```

o <D> : dump trace memory

<G> 명령어 수행 중이 아닌 상태에서 트레이스 정보를 다시 보고자 할 때 사용하는 명령어로, 마지막 <G> 명령어 수행이 끝난 상태의 트레이스 정보를 디스 플레이하여 준다.

5. 결론

이제까지 VMEbus 상의 정보 저장 및 분석의 기능을 갖는 VME-BSA의 특성과 구성, 기능 등에 관하여 자세히 살펴보았다. 컴퓨터 시스템의 다양성과 복잡성, 호환성의 제문제를 해결하기 위하여 대부분의 마이크로 또는 수퍼마이크로 컴퓨터 시스템은 표준화된 백플레인버스를 사용하는 것이 보편화 되었으며, 이러한 시스템의 구성 시에 발생하는 여러가지 에러를 발견하고 수정하기 위하여 BSA와 같은 버스 상의 tool의 필요성은 점점 중요시되고 있다.

현재까지 개발된 BSA의 기능을 갖는 tool은 2가지 형태로, 첫째는 하나의 중앙제어 모듈과 여러가지의 버스 및 프로세서 모듈을 가지므로서 다양성을 강조한 경우인데 이는 특정 모듈을 위한 기능의 구현이 [illegible]하고, 둘째는 특정 버스를 위한 모듈을 제공하는 경우인데 이는 단순한 기능만을 제공하는 약점이 있다. VME-BSA는 VMEbus의 경우 위의 두가지의 약점을 보완하고 더 많은 편의를 제공하는데, 특징적인 것만을 요약하면 버스 상에 삽입되는 하나의 모듈 내에 전체의 기능을 구현한 점 (*),

일반적인 ASCII 터미날을 콘솔로 이용하여 제약을 없앤점, 매우 다양한 트리거 조건을 구현한 점 (**) 등이다. VME-BSA의 개발 기술은 다른 백플레인 버스에 알맞는 BSA의 구현 시에도 쉽게 적용될 수 있으며, 다른 로직 설계에도 응용될 수 있다. 앞으로의 연구 방향은 더욱 다양한 기능 (연속 트리거 조건 제공, 트리거 조건 counting, 선택된 조건만의 트레이스, real time 트레이스 등등) 의 구현 및 좀 더 편리한 사용자 인터페이스의 제공 등이 될 것이다.

주 : (*)와 (**)에 관한 부분은 현재 특허 출원 중

참고 문헌

[1] TEX8540IU System User Manual, Textronix, Oct 1983

[2] VMEbus Monitor Board User Manual, Signetics, 1983

[3] Realtime Bus State Analyzer User's Manual, Motorola, May 1983

[4] VMEbus Specification Manual, Motorola, Feb 1985

[5] Fairchild Advanced Schotty TTL Data Book, Fairchild, 1984/1985

[6] Mitsubishi Semiconductor Bipolar Digital IC LSTTL Data Book, Mitsubishi Electronic, 1985