마이크로프로세서의 메모리 시스템 설계방법에 관한 고찰 (A Method for the Design of Microprocessor Memory Systems)

양충렬* 소운섭** 이규욱* 김진태***
(C. R. Yang, W. S. So, K. O. Lee, J .T. Kim)

32비트 이하의 마이크로프로세서를 위한 메모리 시스템을 설계하는 방법에 대하여 체계화하여 고찰하였다 이를 위하여 메모리 디바이스 선택을 위한 일반 사항과 메모리 시스템 설계 방법에 대하여 서술하고, 이 설계 방법에 따라 16비트 마이크로프로세서 메모리 시스템 설계 예와 설계된 메모리 시스템의 분석, 메모리 디바이스의 속도결정 등에 대하여 서술하였다

I. 서 론

마이크로프로세서를 위한 메모리 시스템의 설계 방법[1-4]은 보편적으로 잘 알려져 있으나 체계화되어 있지 않고 소형이면서 간단한 프로세서에 국한되고 있다. 메모리 시스템의 설계를 위해서는 먼저 ROM, RAM, 레지스터 블록 및 I/O 디바이스 등으로 구성되는 필요조건을 결정하여야 하고, 다음으로 메모리 맵 테이블의 작성과 칩 선택을 위한 어드레스 디코딩 로직 설계를 거쳐 실제 회로 설계도를 작성한 다음 실제 메모리 테이블을 작성한다. 그리고 디바이스의 속도 결정과 버퍼링을 거쳐 마이크로프로세서에 인터페이스되는 메모리 시스템의 설계를 완료하게 된다. 본 고에서는 이러한 설계 단계를 체계화하였고 TDX-10 교환기 시스템의 MC68000 마이크로프로세서를 채용하고 있는 특정 신호단말 보드를 대상으로한 시범적 설계 예를 제시하였다. 이 방법은 메모리 할당형(memory mapped) I/O 디바이스 및 RAM, ROM에 적용되며, 8, 16 및 32 비트 데이터 버스와 임의 사이즈의 어드레스 버스를 이용하는 다양한 버스 구조에 사용될 수 있다. 또한, 디바이스 메모

* 교환정합연구실 선임연구원
** 교환정합연구실 선임기술원
*** 교환정합연구실 실장

리 공간의 위치를 쉽게 가시화할 수 있으며, 어드레스 디코딩 로직의 불(Boolean)방정식을 쉽게 구할 수 있다. 이 방정식은 디코더나 PLD(programmable logic device) 같은 MSI(medium scale integrated circuit) 디바이스, 게이트 수준의 설계로 수행될 수 있다. 메모리 시스템 설계에 있어서 버스 트러블에 대한 가능성이 쉽게 가시화될 수 있기 때문에 이를 방지할 수도 있으며 기 설계된 메모리 시스템의 확장 및 분석이 가능하다.

II. 메모리 시스템 설계

1. 일반 사항

마이크로프로세서에 이용되는 메모리 디바이스를 선택[5]하기 위한 일반적인 사항과 디코더에 대하여 살펴본다. 메모리의 기본단위는 하나의 비트를 기억하는 회로 즉, 플립플롭이다. 이 플립플롭을 8개 혹은 16개 혹은 32개를 한 묶음으로 한 것이 레지스터이고 이 레지스터를 병렬로 나열해 놓은 것이 RAM(random access memory)이다. RAM은 디코더 즉, 어드레스 선택 회로와 버퍼를 필요로 한다. DRAM (dynamic RAM)과 SRAM(static RAM)이 흔히 이용되고 있다.

SRAM은 플립플롭 회로로 구성된 메모리이고, DRAM은 MOSFET(metal oxide semicon field effect transistor)와 부유 커패시터로 이루어진 메모리로서 간단히 말해서 이들의 차이는 메모리 셀의 형태에 있다. DRAM은 전하의 축적을 이용하므로 2~4ms 이내에 그 내용을 다시 기입하는 리프레시 회로가 필요하며 전원 공급이 중단되면 데이터는 전부 소멸되는 대용량에 적합한 메모리로서 값이 저렴한 이점 때문에 널리 이용되고 있다. 한편, ROM(read only memory)은 TTL(transistor-transistor logic)이나 MOS로 이루어진 것으로 사용자 프로그램이나 데이터를 기억하는 메모리이다. 프로그램을 유지하기 위해 전원을 계속 인가할 필요도 없고 DRAM처럼 리프레시도 필요 없다. 전기적으로 사용자에 의해 프로그램 되는 EPROM과 자외선의 방출이나 전기적 수단에 의해 지울 수 있는 EEPROM(electrically erasable PROM)이 흔히 이용된다.

메모리 디바이스 특히 DRAM은 보통 때보다 어드레스시에 대체로 큰 공급 전류를 필요로 하기 때문에 모든 전원과 접지 경로가 메모리 어레이에 비해 넓으며, 공급 전압의 충격을 흡수하기 위해 메모리 디바이스의 핀에 최대한 가깝게 커패시터를 접속[6]한다.

메모리 시스템은 비록 대부분의 응용에서 많은 메모리를 필요로 하지 않아도 마이크로프로세서의 전체 공간을 채우기 위해 설계된다. 외부 어드레스 디코딩 로직은 메모리 칩 선택 입력 외에 메모리 확장을 위해 반드시 필요하며 이 외부 로직은 메모리 디바이스 선택에 필요한 부가 어드레스 비트를 디코드하게 되는데 이를 위해 8비트 이하의 비교적 간단한 메모리 시스템에서는 2, 3 또는 4 입력을 디코드하는 MSI 디코더가 주로 사용되고 그렇지 않은 경우 PAL(programable array logic), GAL(generic array logic), FPGA(field programmable gate array), EPLD(erasable PLD) 등이 주로 사용된다. 메모리 확장에 요구되는 부가 어드레스 선은 디코더 입력에 연결되고 디코더 출력은 메모리 디바이스의 칩 선택 입력을 위한 신호를 공급한다. 보다 큰 디코더는 1개 이상의 인에이

블 입력을 갖는 것도 있다. 어드레스 디코딩 로직은 보다 높은 순서의 어드레스 비트를 디코드하고 하나의 메모리 디바이스의 3-스테이트 출력만을 인에이블 한다.

2. 메모리 시스템 설계 방법

모토롤라의 MC68000 프로세서를 사용하는 보드에서 디코더 회로는 특정 디바이스를 선택하기 위한 회로로서 CPU 또는 DMAC(direct memory access circuit)로 부터 발생된 어드레스 맵에 의해 디코딩되어 RAM이나 ROM 또는 주변 디바이스 및 각종 레지스터들을 선택하거나 ROM 인에이블 신호를 생성한다. 메모리 테이블을 검사하여 메모리의 어드레스 디코딩 로직을 결정할 수 있다. 메모리 테이블은 요구 위치에 각 디바이스의 어드레스 선에 대한 로직 수준을 나타내는 DMT(desired memory table)를 먼저 작성하고 이를 바탕으로 어드레스 디코딩 로직에 의해 요구된 실제 로직 수준을 반영하는 AMT(actual memory table)를 셋업한다. 본 고에서 표기되는 메모리 테이블의 0, 1은 신호의 논리 L(low), H(high) 상태를 나타내고 X는 이 H, L에 대한 무관 조건(don't care condition)을 나타낸다. 점(·)은 각 칩에 있는 어드레스 입력에 어드레스 버스 선이 직접 연결되는 것을 나타내는데 이 어드레스 선은 그 디바이스 내의 특정 위치를 선택하기 위하여 내부적으로 디코드된다. 어드레스 버스의 상위 선들은 특정 칩을 선택하는데 사용하고 하위 선들은 칩내부의 바이트를 선택하는 데 사용한다. I/O 포트 수가 작으면 일부의 어드레스 선이 직접 칩에 연결되고 다수의 어드레스 선은 디코드되지 않을 수도 있다. 어떤 어드레스 선은 디코드되지 않을 수 있기 때문에 테이블 내의 모든 디바이스의 유용한 위치를 결정할 때 X 자리에 0 또는 1의 논리 수준을 지정하여야 한다. X를 지정하는 방법은 시스템 내에서 다른 디바이스와 버스 충돌의 가능성을 피하기 위한 것이기 때문에 주의 깊게 선택하여야 한다. 메모리 시스템의 설계 단계를 다음과 같이 6단계로 구분하였다.

첫째, 마이크로프로세서는 응용에 필요한 RAM, ROM 및 I/O 디바이스의 메모리 필요조건을 결정한다. 그리고 나서 이 설계를 위해 필요한 메모리 량을 계산하여 가능한 실제 디바이스의 필요 칩 수를 결정한다.

둘째, 각 메모리별로 어드레스 범위를 결정하고 초기 메모리 맵을 그린다.

셋째, 시스템에 필요한 메모리 테이블을 작성하고 어드레스 디코딩 로직을 결정한다.

넷째, 위에서 결정된 어드레스 디코딩을 위한 칩 셀렉트 로직으로 실제 설계도를 그린다.

다섯째, 실제 메모리 테이블을 작성하고 가능한 버스 접속을 위한 설계를 검토한다.

여섯째, 위에서 메모리 시스템의 칩 셀렉트 로직의 설계와 오류에 대한 검사가 끝나면 메모리 디바이스의 속도 및 필요한 버퍼링을 결정한다.

가. 16비트 마이크로프로세서 메모리 시스템의 설계 예

본 절에서는 TDX-10 교환기의 신호 단말보드로서 위에서 서술한 설계 방법에 따라 MC68000을 사용한 16비트 마이크로프로세서[7]를 위한 메모리 시스템을 설계하고자 한다. 먼저 메모리 시스템의 필요조건을 결정한다. 이 신호 단말보드를 구성하는

메모리 시스템의 메모리 디바이스는 EVEN, ODD 어드레스로 나뉘어진 256kbyte SRAM(431000) 2개, 128kbyte EPROM(27C512) 2개로 구성되며, I/O 디바이스는 모두 256byte 영역을 갖는 4개의 I/O 디바이스-MPCC, MPCC-AP 등 2개의 SIOC(serial input/output comm.) 칩과 MFP(multi-function peripheral) 및 DMAC를 포함한다. 그리고 디코더는 기능별로 분류하여 5개의 PAL 디바이스[8]로써 구성하였다. 레지스터 디코더는 각각 1개씩의 상태, 제어 및 2개의 식별 등 4개의 I/O 레지스터를 포함하는 레지스터 블록이다. 메모리 시스템은 메모리 공간 내에 적절히 위치한 이들 디바이스를 갖는 프로세서를 위해 설계되어야 한다. MC68000은 메모리 할당형 I/O 방식을 채용하는데 입출력 장치도 메모리의 일부로 보기 때문에 입출력 장치의 어드레스 공간이 따로 없고 입출력 장치에 액세스하기 위해 메모리 명령을 사용한다. CPU 리셋 후 초기 단계에서 CPU는 프로그램 카운터 및 스택 포인터 값을 ROM의 시작 번지에서부터 4워드를 읽어 오는데 이는 초기 ROM 선택 회로에 의해 수행되며, 이 후 이들 값에 의해 정상적인 동작을 하게 된다. 이 보드는 24비트 어드레스를 사용하며 디코더는 회로 기능별로 분리하여 5개의 PAL 칩에 의해 구현하고 있다.

다음에 이 시스템에 필요한 초기 메모리 맵 테이블을 그린다. 〈표 1〉에 메모리 및 I/O 맵을 작성하였다. 메모리의 어드레스 할당은 각 칩에 할당된 메모리 어드레스 테이블에 의해 결정된다. 예를 들면, 32kbyte의 ROM은 15비트(215) 어드레스 영역을 가지며 이는 이진수로 0111 1111 1111 1111 범위이며 16진수로는 $0000-$7FFF로 나타낸다.

셋째, 〈표 1〉과 같이 시스템에 필요한 메모리 테이

〈표 1〉 메모리 및 I/O 맵

$000000	SRAM (256kbyte)
$040000	사용 안함
$AD0000	MFP (256 byte)
$AD0100	MPCC (256 byte)
$AD0200	MPCC-AP (256 byte)
$AD0300	DMAC (256 byte)
$AD0400	사용 안함
$AE0000	REGISTER BLK.
$AF0000	사용 안함
$F00000	EPROM (128kbyte)
$F20000	사용 안함
$FFFFFF	

블을 완성한 다음, 〈표 2〉의 DMT와 같은 어드레스 디코딩 로직을 결정한다. RAM 어드레스와 ROM 어드레스 사이의 구별은 다른 버스 선으로 하며 이 칩 선택은 사용되지 않는 버스 선을 이용한다. 마이크로프로세서는 인터페이스 장치와도 통신하기 때문에 각 인터페이스에도 어드레스를 할당할 필요가 있다. 인터페이스에 할당된 어드레스의 감지는 일반적으로 외부에 AND 게이트를 삽입하여 수행한다. RAM과 ROM 디바이스는 액티브 L 칩 셀렉트만 갖고, 대부분의 I/O 디바이스와 마찬가지로 I/O 디바이스는 액티브 H와 액티브 L 칩 셀렉트를 갖는다. I/O 디바이스의 위치가 특별히 정의되지 않아도 이들 디바이스를 선택적으로 활성화하기 위해 액티브 H 칩 셀렉트를 사용한다. 필요한 메모리 테이블에 따라 칩 선택 로직의 부울 방정식을 결정할 수 있다. 이 방정식의 개수와 로직상태에 따라 1×4, 1×8, 3×8 등의 디코더가 선택적으로 사용된다. 이 방정식은 여러 형태의 논리 결합으로 수행되는데 예를 들어 7개의 방정식과 액티브 L 출력을 갖는다면 1 to 8 디코더

인 74138이 적합하다. 대부분의 칩 셀렉트 로직이 디코더에 어드레스 선을 직접 연결함으로써 바로 수행된다. 이 메모리 맵 테이블의 검사에 의해 A23-A16의 2진 값으로 어서트된 조건이 각 디바이스들을 구별해 주는 것을 알 수 있다. MC68000에서 23개의 어드레스 선은 A0에 상당하는 UDS(upper data strobe), LDS(lower data strobe)까지 합하여 24비트의 어드레스 지정을 하여 16Mbyte의 메모리 공간에 직접 액세스한다. ROM, RAM은 액티브 L 칩 셀렉트를 갖기 때문에 〈표 2〉의 DMT에 대한 어드레스 디코딩 로직 방정식은 다음과 같이 4개로 최소화되어 나타난다. MC68000은 데이터 버스가 16비트 워드이므로 워드 동작을 한다. 따라서 SRAM 및 EPROM을 구분하기 위해서는 CPU에서 공급되는 UDS, LDS 신호를 ROMSEL 및 RAMSEL 신호와 조합시켜 메모리 디코더에서 디코딩해 준다. 4개의 I/O 디바이스를 구분하기 위해서는 IOSEL과 A8-A11 신호를 조합하여 I/O 디코더에서 256 바이트씩 디코딩하여 각각의 I/O 디바이스를 선택하도록 한다[9]. 그리고 4개의 레지스터를 선택하기 위해 REGSEL과 A1-A3 신호를 조합하여 레지스터 디코더에서 디코딩한다.

!SRAM = !(!A23 · !A22 · !A21 · !A20 · !A19 · !A18)

!I/O device = !(A23 · !A22 · A21 · !A20 · A19 · A18 · !A17 · A16)

!REGISTER = !(A23 · !A22 · A21 · A20 · A19 · A18 · A17 · !A16)

!EPROM1 = !(A23 · A22 · A21 · A20 · !A19 · !A18 · !A17)

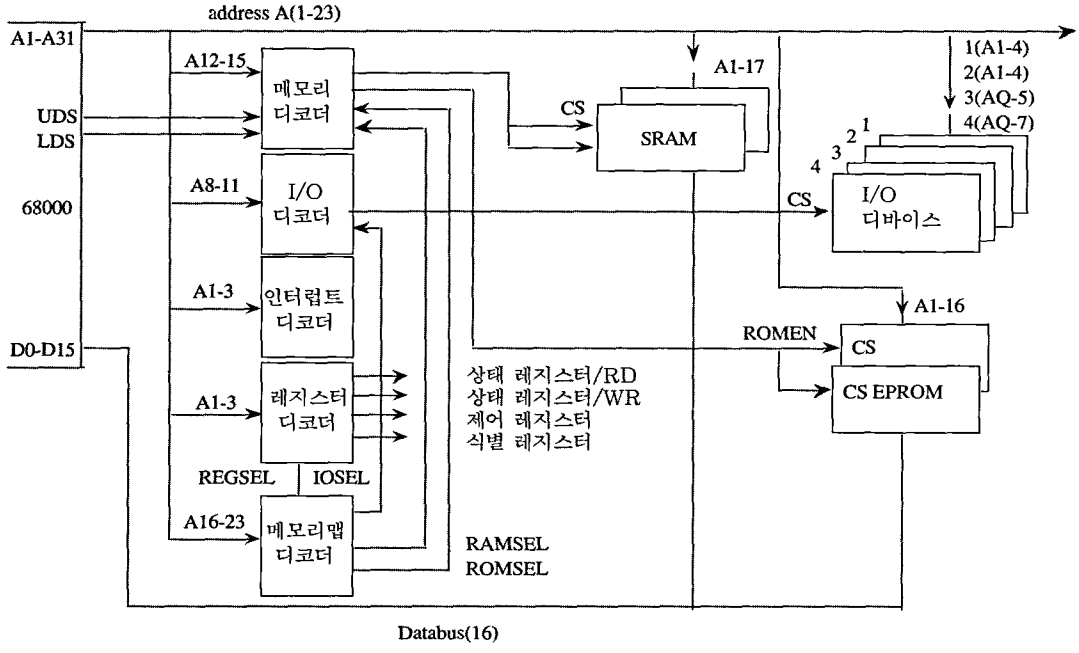
이 때 주의할 점은 어드레스 디코딩 로직은 보다 높은 순서의 어드레스 비트 및 제어 선을 그 입력으로 가지며 버스 충돌이 없도록 한번에 한 디바이스가 인에이블되거나 선택되어야 한다. 블록 사이즈를 결정하기 위한 대안은 낮은 순서의 어드레스 비트를 디코더에 연결하지 않으면 된다.

넷째, 〈표 2〉에서 결정된 어드레스 디코딩을 위한 칩 셀렉트 로직으로 (그림 1)과 같은 실제 회로 설계도를 그린다. 간단히 하기 위해서 출력 인에이블과 쓰기 인에이블을 위한 타이밍 도는 생략하였다.

다섯째, 〈표 3〉의 실제 메모리 테이블을 작성하고 가능한 버스 접속을 위한 설계를 검토한다. 이 때 앞에서도 언급하였듯이 디바이스와 버스 충돌의 가능성을 피하기 위하여 무관 조건인 X를 지정할 때 유의하여야 한다. 그리고 버스 접속을 위한 설계가 적절한지 검토한다. 16비트 또는 32비트 마이크로프로세서를 채용하는 메모리 시스템에서는 사용되는 I/O 디바이스의 수가 대체로 많고 어드레스 영역도 여유가 있으므로 〈표 2〉와 같은 방법으로 구성하는 것이 좋고, 8비트 마이크로프로세서를 채용하는 간단한 메모리 시스템은 대체로 I/O 디바이스의 수가 적고 어드레스 영역이 그리 여유가 없는 편이므로 〈표 4〉에 나타낸 바와 같이 하위 어드레스 선을 I/O 디바이

〈표 2〉 DMT

ADDRESS / DEVICE	A A A A 23 22 21 20	A A A A 19 18 17 16	A A A A 15 14 13 12	A A A A 11 10 09 08	A A A A 07 06 05 04	A A A A 03 02 01 00	Usable Location
SRAM	0 0 0 0	0 0 · ·	· · · ·	· · · ·	· · · ·	· · · ·	$000000-$03FFFF
I/O Device	1 0 1 0	1 1 0 1	· · · ·	· · · ·	· · · ·	· · · ·	$AD0000-$ADFFFF
Register Blk	1 0 1 1	1 1 1 0	· · · ·	· · · ·	· · · ·	· · · ·	$AE0000-$AEFFFF
EPROM	1 1 1 1	0 0 0 ·	· · · ·	· · · ·	· · · ·	· · · ·	$F00000-$F1FFFF



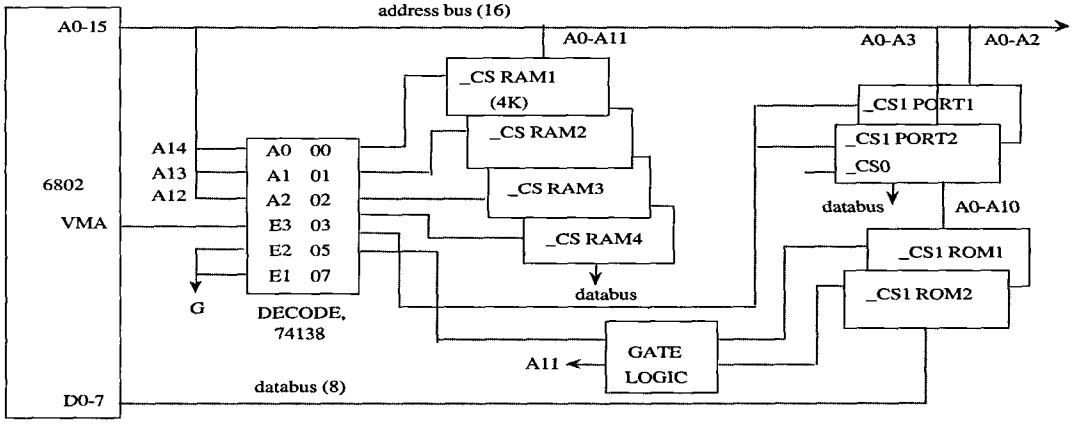
(그림 1) 간략화한 68000 마이크로프로세서의 메모리 시스템 설계도 예

〈표 3〉 AMT

ADDRESS / DEVICE	A A A A 23 22 21 20	A A A A 19 18 17 16	A A A A 15 14 13 12	A A A A 11 10 09 08	A A A A 07 06 05 04	A A A A 03 02 01 00	Usable Location
SRAM	0 0 0 0	0 0 · ·	· · · ·	· · · ·	· · · ·	· · · ·	$000000-$03FFFF
I/O Device	1 0 1 0	1 1 0 1	X X X X	0 0 · ·	· · · ·	· · · ·	$AD0000-$AD03FF
Register Blk	1 0 1 1	1 1 1 0	X X X X	X X X X	X X X X	· · · ·	$AE0000-$AE000F
EPROM	1 1 1 1	0 0 0 ·	· · · ·	· · · ·	· · · ·	· · · ·	$F00000-$F1FFFF

스 선택에 이용하는 메모리 테이블이 권고된다.

(그림 2)는 8비트 마이크로프로세서를 채용한 전형적인 메모리 시스템 설계도이다. 4kbyte의 소용량 RAM과 2kbyte ROM을 사용하고 하나의 디코더로 간단히 구현한 경우이다. A12, A13 및 A14를 A0, A1, A2에 직접 연결함으로써 대부분의 칩 선택 로직이 바로 수행되며, 디코더와 AND 게이트로 ROM과 RAM 디바이스를 분리한다. A11은 ROM1, ROM2를 구별하는 데 사용되고 A15는 디코드되지 않는다. 대부분의 경우 메모리 디바이스를 위한 신호를 생성하기 위한 타임 신호는 Read/Write가 AND된다. 마이크로프로세서에 의해 발생되는 액티브 H 신호인 VMA(valid memory address)는 디코더의 액티브 H 칩 인에이블에 사용된다. 〈표 4〉에 위 (그림 2)의 AMT를 나타내었다. 여기서 액티브 L 칩 셀렉트에 연결된 로직과 조합되는 I/O 포트의 어드레스 선 가운데 A4, A5중 하나의 H 수준이 2개의 포트를 구별하기 위해 사용되고 있다. I/O 디바이스를 위해 할당된 어드레



(그림 2) 8비트 마이크로프로세서 메모리 시스템 설계도

〈표 4〉 8비트 마이크로프로세서를 위한 메모리 시스템 AMT

DEVICE	A15 A14 A13 A12	A11 A10 A09 A08	A07 A06 A05 A04	A03 A02 A01 A00	Usable location
RAM1	X 0 0 0	· · · ·	· · · ·	· · · ·	S0000 - S0FFF
RAM2	X 0 0 1	· · · ·	· · · ·	· · · ·	S1000 - S1FFF
RAM3	X 0 1 0	· · · ·	· · · ·	· · · ·	$2000 - $2FFF
RAM4	X 0 1 1	· · · ·	· · · ·	· · · ·	$3000 - $3FFF
PORT1	X 1 0 0	X X X X	X X X 1	X · · ·	$4010 - $4017
PORT2	X 1 0 0	X X X X	X X 1 X	· · · ·	$4020 - $402F
ROM1	X 1 1 1	0 · · ·	· · · ·	· · · ·	$F000 - $F7FF
ROM2	X 1 1 1	1 · · ·	· · · ·	· · · ·	$F800 - $FFFF

스 범위 안에서 연속적으로 A4 및 A5를 어서트할 가능성 때문에 규정된 위치(usable location)를 사용하지 않으면 일시적으로 I/O 디바이스들간에 충돌이 일어날 수 있다. 예를 들면 또 다른 포트가 액티브 H 칩 셀렉트를 구동하기 위해 A9를 사용할 수 있고 사용 가능한 위치를 적절히 선택하면 버스 충돌은 일어나지 않는다. 비록 간단한 메모리 시스템일지라도 흔히 메모리 어드레싱이 문제가 되는 경우가 흔히 발생되기 때문에 유의하여야 한다. 이에 관해서는 3절에 설명한다. 이 방법을 이용하면 시스템의 I/O 확장을 대단히 쉽게 할 수 있다. 끝으로 여섯째, 메모리 디바이스의 속도 결정 및 버퍼링에 대하여는 4절에 이를 설명한다.

3. 설계된 메모리 시스템의 분석

메모리 시스템의 확장 또는 재설계할 필요가 있을 때 더 많은 메모리 또는 I/O 디바이스를 추가적으로 필요로 하는 경우 및 기존 설계를 해석해야 할 경우

가 흔히 생기게 된다. 이 경우 설계도에서 바로 AMT를 그리는 단계 즉, 첫 번째에서 세 번째 단계는 간단히 해석된다. 〈표 3〉에서 I/O 디바이스가 추가될 경우 디코딩은 A9와 A8을 사용하는 어드레스 영역은 모두 사용되었으므로 A11과 A10을 추가 포함하여 디코딩하면 256바이트 용량의 I/O 디바이스를 최대 16개까지 추가할 수 있다. 또, RAM과 ROM을 확장할 경우에는 A19, A18, A17을 이용한다면 256kbyte RAM은 최대 4개까지 128kbyte ROM은 최대 8개까지 추가할 수 있다. 레지스터가 추가될 경우에는 A3 신호를 디코딩하면 되므로 메모리 확장이 가능하도록 되어 있으며 상호 겹치는 어드레스 공간이 존재하지 않으므로 버스 문제점이 발생하지 않고 따라서 이 시스템은 정상 동작하게 된다[10]. 메모리(워드 길이) 확장은 어드레스 및 칩 선택 신호를 분담하도록 병렬로 몇 개의 메모리 디바이스를 연결하는 방식으로 한다.

4. 메모리 디바이스의 속도 결정 및 버퍼링

앞에서 메모리 시스템의 설계 방법에 따라 AMT를 설계, 칩 셀렉트 로직의 설계와 오류에 대한 검사가 완료되면 메모리 디바이스의 속도를 결정하여야 한다. 속도에 따른 메모리 디바이스의 종류는 ROM, RAM같은 랜덤 액세스 메모리와 시프트 레지스터 같은 시퀀셜 액세스 메모리의 두 가지로 분류된다. (그림 2)는 68000에 인터페이스된 메모리 시스템을 나타낸다. 메모리 디바이스의 속도 결정을 위한 목표는 시스템 타이밍을 제어하는 마이크로프로세서 명령이 수행될 때 WAIT 상태가 필요 없도록 이들 디바이스에 요구되는 속도를 결정하는 것이다. 메모리 디바이스의 지연뿐만 아니라 어드레스 디코딩 로직이 메모리 시스템의 속도를 더 느리게 할 수 있기 때문에 이 로직을 통한 신호의 전파 지연이 최소가 되도록 최소의 어드레스 비트로 구성되는 어드레스 디코딩 로직을 설계할 필요가 있다.

일반적으로 메모리 디바이스의 속도는 사용자 매뉴얼에 잘 나타나 있기 때문에 이러한 속도를 마이크로프로세서의 속도에 맞도록 적절히 결정하면 된다. 메모리나 대부분 소자에 필요한 마이크로프로세서 비동기 제어 신호인 DTACK(data acknowledge) 신호 발생용 타이밍 회로는 74F164 시프트 레지스터를 사용하여 설계되며 ROM 어드레스가 선택되고 이 ROM의 액세스 타임이 120ns, 74F164의 기준 클럭이 시스템 기준클럭인 10MHz(100ns)로서 CPU 클럭과 같다고 할 때 지연 시간은 120/100=1.2, 즉 2클럭의 지연이 필요하다. 이렇게 하면 DTACK 신호는 어드레스 디코더에 의해 특정 디바이스가 선택된 다음 디바이스의 액세스 시간만큼 시간이 경과한 후에 발생하게 된다. 즉, 칩 자체의 액세스 타임에 따라 RAMACK, ROMACK가 어서트되는 시간을 옵션적으로 가변할 수 있다.

주변 디바이스를 보편적으로 많이 채용되는 메모리 시스템에는 어드레스, 데이터 및 제어 버스를 위한 메모리 버퍼가 필요한데 앞에서도 언급하였듯이 여기서는 버스를 사용하는 주변 디바이스가 많지 않고 DMAC를 채용하고 있다. CPU가 버스 마스터 일 때에는 AS*, UDS*, LDS*, R/W*가 입력이 되고 DTACK*은 출력으로 동작하여 CPU는 DMAC의 내부 레지스터를 읽고 쓰고 할 수 있지만, 반대로 DMAC가 버스 마스터일 때에는 신호 방향은 반대가

되고 일련의 버스 사이클을 DMAC가 관리한다. 따라서 DMAC를 위한 버퍼가 필수적이다[11]. 결론적으로 말해서 메모리와 I/O 디바이스를 위한 속도 필요조건을 계산할 때 그 시스템에 채용되는 버퍼의 전파 지연도 반드시 포함되도록 고려하여야 한다.

III. 결 론

본 고는 메모리, I/O 칩 또는 레지스터 블록의 선택 로직으로서 PLD를 사용하는 16비트 마이크로프로세서의 메모리 시스템을 설계하였다. 이 방법은 32비트 이하의 마이크로프로세서의 메모리 시스템을 설계하는 데에 적용된다. 전형적이고 간단한 설계 방법을 응용하여 실제적이고 효율적으로 32비트 이하의 마이크로프로세서에 대하여 메모리 시스템을 설계할 수 있도록 설계 방법을 체계화하여 제시하였고 이 방법을 통하여 메모리 시스템의 평가를 위한 분석 및 메모리 시스템 확장 방법을 고찰함으로써 메모리 시스템 설계를 위한 해석툴로서도 유용하게 활용할 수 있도록 하였다.

참 고 문 헌

[1] Jerry J Cupal, "A Technique for the Design of Microprocessor Memory Systems," *IEEE Transactions on Education*, vol. 37, no.3, pp. 237~242, Aug 1994.

[2] Rulph Chassaing, "A Course in Microprocessors Based on the 16/32 Bit 68000 μP," *IEEE Transactions on Education*, vol. E-30, no 3, pp. 194~197, Aug 1987.

[3] John D, Carpinelli, "Bridging the Gap Between Digital Circuits and Microprocessors," *IEEE Transactions on Education*, vol. 36, no.3, pp. 334~339, Aug. 1993.

[4] Kenneth L. Short, *Microprocessors and Programmed Logic*, Englewood Cliffs, NJ: Prentice-Hall International Inc., pp 49-58, 1987.

[5] William D, Simpson, MSEE, *Microprocessor/Microcomputers/ System Design*, TI Electronics, QA76. S T49, pp. 4-30~4-31, 1980.

[6] 황희륭, 디지털 논리와 컴퓨터 설계, 동일 출판사, 서울 1991 2, pp. 11-54.

[7] Motorola Inc , *M68000 8-/16-/32 Bit Microprocessors User's Manual*, Englewood Cliffs, NJ: Prentice-Hall, 1989.

[8] Lattice, *GAL Handbook*, 1986

[9] 진달복, 16/32비트 마이크로프로세서, 1992 4, pp. 11-54

[10] *Programmable Peripheral Design and Applications Handbook*, WaferScale Integrate, Inc., 1992, pp. 2-391~2-402

[11] Motorola Inc., *MC68450 Advanced Information DMAC*, 1986, pp. 2-3.