GOVP1199217940

제 2 차년도
최종 보고서

고화질 의료용 영상 시스템에 관한 연구

A Study on the High-resolution Medical Image Processing System

연구기관
한국과학기술원

과 학 기 술 처

제 출 문

과학기술처 장관 귀하

본 보고서를 "고화질 의료용 영상 시스템에 관한 연구" 과제의 2차년도 최종 보고서로 제출합니다.

1992년 7월 20일

연구기관 : 한국과학기술원

전기 및 전자공학과

연구책임자 : 나 종 범

연구원 : 천 강욱

이 우용

이 민섭

조 홍규

임 경원

강 광수

유 한주

조 현덕

요 약 문

I. 제목

고화질 의료용 영상시스템에 관한 연구

II. 연구개발의 목적과 중요성

과학기술의 선진화를 위해 필요한 첨단기술 분야의 하나인 의료용 디지탈 영상시스템의 국내 개발기술은 아직 미진한 상태에 있다.

종전의 의료용 장비들은 대부분이 아날로그(analog)로 되어 있었으며, 환자에 대한 임상적 데이타를 기록해 둘만한 매체가 적당하지 않았는데 영상에 대한 자료는 대부분 화학적 필름으로 보관되었다. 화학적 필름인 경우 필름의 한계 때문에 장기간의 보관이 어렵고, 보관된 영상의 분석을 위한 관리가 환자의 수가 많아지고 데이타량이 많아지면서 번거로우며 신속하지 못한 단점을 지니고 있었다.

국내에서는 의료용 디지탈 영상시스템 개발 실적이 미미하고 국제 경쟁력을 갖추기에는 그 기술적 확보가 크게 뒤떨어져 현재 그 수요가 급격히 증가하고 있는 시점에서 고성능 고화질 의료용 영상 시스템의 개발은 매우 시급한 실정이다. 이러한 맥락에서 볼때 금번 프로젝트에서 수행된 시스템 개발은 의미있는 일이라 할 수 있다.

III. 연구개발의 내용과 범위

본 연구에서는 PC를 Host로 하여, 병렬처리가 가능한 영상처리 및 도시 시스템을 개발하였다. 1차년도에서는 TMS320c30을 이용한 H/W를 설계하였으나 2차년도에서는 이 기술을 바탕으로 c30보다 훨씬 고성능을 가진 i860

을 이용하여 시스템을 설계하고 실제 Prototype을 제작하여 Hardware 시스템을 구성하였다. 이 시스템의 제원을 살펴보면 다음과 같다.

1. Hardware Specifications

1) Digital Signal Processor

Peak performance 80 MFLOPS

12 KBytes Cache Memory

Parallel processing 가능

Dual port memory를 통한 host interface

2) Image and Graphic Display

1024 x 1280, 60 Hz non-interlaced color display

Full color(pixel depth 24 bit)

256 color graphic overlay structure

Cine display

Hardware panning/zooming

Red, Green, Blue separate drive

8 MBytes image memory

3) Parallel Processing

One GSP processor with two DSP processors

2 MBytes shared memory space

Message passing 및 shared memory 구조의 결합

IV. 연구개발 결과 및 활용에 대한 건의

기존의 일반적인 Computer (PC or Workstation)를 사용하며 의료용 영상

을 처리하기에는 처리 속도와 도시 속도가 느리기 때문에 사용자로 하여금 불편함을 느끼게 한다. 그렇다고 해서 값비싼 Mainframe급의 컴퓨터나 Supercomputer를 사용하기에는 코스트(cost) 문제로 시스템 이용자의 부담을 크게 한다. 전용의 저가격 고속 의료용 영상 처리 및 도시 시스템의 개발은 이러한 두가지 문제를 해결할 수 있다. 본 연구과제에서 기존에 개발한 영상처리 및 도시 시스템의 기능을 확장하고 고속의 의료용 영상처리를 위한 병렬처리 시스템의 아키텍처(Architecture)를 설계하였다.

설계한 시스템에서 DSP(Digital Signal Processor)와 GSP(Graphic System Processor)는 메모리를 공유하므로써 DSP에서 처리한 데이타를 GSP와 서로 교환 하도록 하는 공유 메모리(Shared Memory) 방식을 채택하여 처리한 영상을 빠른 속도로 GSP에서 모니터에 도시할 수 있도록 하였다. 또한 병렬처리를 위한 DSP 사이의 정보의 교환은 GSP Module에 있는 Frame Memory를 사이에 두고 메시지를 주고 받을 수 있도록 하는 메세지 패씽(Message Passing) 방식을 채택하였다. 의료용 영상 데이타는 로칼러티(Locality)가 매우 크고 또한 여러장의 영상을 연속적으로 프로세씽할 경우 두 개의 DSP 프로세서에 각각 다른 영상을 처리하게함으로써 DSP들사이에 데이타 교환이 없게되므로 하드웨어 복잡성(Hardware Complexity)을 줄이기 위해 루우즈하게 결합(Loosely Coupled) 시키는 방법을 채택하여 시스템을 설계하였다. 한편 완성될 계획에 있는 시스템에 필요한 운영체제 소프트웨어와 응용 소프트웨어 개발에 착수하였다.

외국의 의료용 영상 시스템의 연구개발은 매우 활발하게 수행되어왔으며 저가의 고속 영상 시스템 개발(Titan, TAAC-1, MC860VS 등.)은 병렬처리 구조를 채택함으로써 단일 프로세서의 프로세싱 파워(Processing Power)를 극복하기 위한 기술을 개발 발전시켜왔다. 본 연구에서도 이러한 기술개발에 있

어서 기존의 연구개발에서 터득한 영상 처리용 위크스테이션(Workstation) 개발기술을 바탕으로 고속의 병렬 영상 처리 시스템을 구축하는 핵심 기술을 국산 개발하므로써 PET, MRI, X-CT 등과 같은 의료용 영상의 고속처리 및 도시, 및 인체 정보의 3차원 도시 또는 의료용 통신 및 데이타 보관을 위한 PACS의 단말 시스템으로의 응용 등 의료용 영상 처리 및 도시 분야에 많은 기여가 예상된다.

SUMMARY

Medical imaging is getting very important in patient diagnostics, and the need for high performance medical image workstation increases drastically to handle the enormous digital image data.

In recent years, the development of VLSI design technology makes low cost high speed microprocessors and large amount of memory devices available in the design of medical image processing workstation. We have developed a high resolution medical image system. It is PC-based low cost, high performance, high resolution image processing workstation, which consists of loosely coupled image processing modules and the high resolution image display module.

For high speed image processing, this system adopts the powerful microprocessor, Intel 80860, which has 80 MFLOPS peak performance. For high resolution (full color) image display, a graphic processor TMS34020, frame buffer, overlay buffer, program memory, and RAMDAC are used.

CONTENTS

목 차

제1장 서론

제1절 연구 개발의 목적과 범위

1. 연구개발의 개요

과학기술의 선진화를 위해 필요한 첨단기술 분야의 하나인 의료용 영상시스템의 국내 개발기술은 아직 미진한 상태에 있다. 근래들어 Computer를 계산을 위한 단순한 이용단계에서 벗어나 각기 특정 용도에 따라 그 기능을 극대화할 수 있는 워크스테이션(Workstation)으로서의 사용범위를 넓혀나가고 있으며, 의료용 분야에서는 X-ray CT (Computerized Tomography), PET(Positron Emittion Tomography), MRI(Magnetic Resonance Imaging), 초음파 진단기 DR(Digital Radiography) 등과 같은 의료 진단용 시스템을 위한 의료용 영상시스템의 필요성이 태동하게 되었다. 의료용 영상시스템이란 여러장비로 부터 얻어진 신호를 해석하여 영상으로 재구성하고 도시하거나 여러장비로 부터 얻어진 신호를 여러가지 처리없이 직접 영상을 화면에 도시하여 환자에 대한 정보를 의사에게 제공할 수 있는 장비이다. 화면에 고화질 또는 고선명도로 도시된 환자에 대한 영상은 전문의의 판독을 거치고 판독된 결과는 도시된 영상과 함께 실려서 적당한 주변기억 장치(secondary storage)에 저장되게 된다. 종전의 의료용 장비들은 대부분이 아날로그(analog)로 되어 있었으며, 환자에 대한 임상적 데이타를 기록해 둘만한 매체가 적당하지 않았으며, 영상에 대한 자료는 대부분 화학적 필름으로 보관되었다. 화학적 필름인 경우 필름의 한계 때문에 장기간의 보관이 어렵고, 보관된 영상의 분석을 위한 관리가 환자의 수가 많아지고 데이타량이 많아지면서 번거로우며 신속하지 못한 단점을 지니고 있었

다. MRI, PET, X-CT, DR 등과 같은 의료용 영상 추출 장비의 발달로 계속해서 더 많은 영상 데이타가 얻어지고 보관되어져야 하기 때문에 병원 관리적인 면에서도 의료용 영상시스템의 필요성이 가중되어가고 있는 실정이다. 국외의 기술개발 현황을 살펴보면, 오래전서부터 Sun Micro System사에서는 Sun Workstation과 함께 TAAC-1의 고속 신호처리 및 도시장비를 개발하였으며, 또한 Pixar, Stellar, IBM 및 HP사에서도 고속연산을 위한 RISC 프로세서들과 이들을 병렬로 사용하여 영상신호 처리를 빠르게 처리하고 또한 도시할 수 있는 새로운 상품들을 계속 새롭게 내놓고 있는데 미국의 중소기업에서 조차도 새로 개발되어져 나오는 DSP칩들(예:TMS 320CXX, i860, RS20X0 etc)을 사용하여 영상처리 시스템을 자체개발하여 상품으로 출시하고 있다. 그러나 국내에서는 그 개발 실적이 미미하고 국제경쟁력을 갖추기에는 그 기술적 확보가 크게 뒤떨어져 현재 그 수요가 급격히 증가하고 있는 시점에서 고성능 고화질 의료용 영상 시스템의 개발은 매우 시급한 실정이다. 이러한 맥락에서 볼때 금번 프로젝트에서 수행될 시스템 개발은 국가 연구기술적 차원에서 의미있는 일이라 할 수 있다.

2. 연구개발의 범위

기존에 본 연구팀에서 개발한 영상처리 및 도시시스템을 금번 프로젝트를 통하여 병렬처리가 가능하게하고 그 성능을 확장시켰다. 1차년도에 Hardware 시스템을 설계하고 2 차년도에 설계된 시스템을 실제 Prototype 제작하며 3차년도에 Software를 추가하여 전체 시스템을 구성하려는 계획인데 이러한 시스템의 제원을 살펴보면 다음과 같다.

가. Hardware Specifications

(1) Digital Signal Processor

Peak performance 80 MFLOPS

12 KBytes Cache Memory

Parallel processing 가능

Dual port memory를 통한 host interface

2) Image and Graphic Display

1024 x 1280, 60 Hz non-interlaced color display

Full color(pixel depth 24 bit)

256 color graphic overlay structure

Cine display

Hardware panning/zooming

Red, Green, Blue separate drive

8 MBytes image memory

3) Parallel Processing

One GSP processor with two DSP processors

2 MBytes shared memory space

Message passing 및 shared memory 구조의 결합

나. Software Specifications

(1) Image and Graphic Software

Multiple image display

Cine Display for Cardiac motion and 3-D volume images

ROI(Region of interest) windowing

3-dimensional display (ex. Volume rendering)

(2) Digital Signal Processing

Complex Floating Point FFT

임의 배의 image scaling

3 x 3, 5 x 5, 7 x 7, 9 x 9 convolution

Multiple window control

Low pass/high pass filtering

Various image processing technique 적용

제2장 하드웨어 시스템의 구조

제1절 DSP(Digital Signal Processor)의 구조

1. Host 접속(Interface)부

Host와 DSP들 사이의 데이타 전송에 있어서 Host측은 DMA를 사용하지 않는다 . Host측은 Programmed I/O를 수행하고 DSP측에서는 DMA를 사용한다.

Host와 DSP 프로세서간의 데이타 전송을 효율적으로 하기 위하여 Dual Port SRAM을 사용하여 보다 빠르게 능률적인 전송이 가능하도록 설계하였다. 데이타 전송은 Dual Port SRAM을 사용한 메세지 패씽(Message Passing) 방식을 채택하였으며, 사용한 Dual Port SRAM은 하드웨어 Semaphore기능, Busy 발생기능 및 인터럽트 발생 기능 등 다양한 기능을 포함하고 있으며 완전히 독립적으로 콘트롤 할 수 있는 두개의 Port를 제공하므로 Host와 DSP가 독립적으로 데이타를 전송할 수 있다.

가 Host to DSP 데이타 전송

Host에서 데이타를 Read 또는 Write하려면 데이타 전송에 필요한 정보와 더불어 데이타를 Dual Port SRAM에 스토어(Store)한 후 DSP쪽에 Dual Port SRAM의 인터럽트 발생 기능을 이용하여 인터럽트를 발생시킨다. DSP에서 인터럽트 서비스 루틴(Interupt Service Routine)을 수행하여 데이타 전송에 필요한 정보를 해독하며 필요한 서비스 루틴을 부르므로써 데이타 전송이 이루어진다.

나 DSP to Host 데이타 전송

DSP에서 Host쪽에 데이터 전달을 행하려면 데이타 전송에 필요한 정보와 실제 데이타를 Dual Port SRAM에 써주고 난후 Host쪽에 Dual Port SRAM의 인터럽트 발생 기능을 이용하여 인터럽트를 Host쪽에 걸어 주는데 Host에서는 이 인터럽트를 받아 데이타 전송에 필요한 정보를 해독하여 이때 필요한 서비스를 수행하므로써 효율적인 데이타 전달이 이루어지도록 설계하였다.

다 DSP와 Host 접속부 설계

Host와 신호처리 프로세서로 사용된 i860과의 접속을 위한 기능을 포함하지 않으므로 접속을 위한 복잡한 인터페이스 로직이 필요하다. 본 연구과제에서는 인터페이스 로직의 복잡성을 피하고 보다 빠른 데이타 전송을 가능하도록 하기 위해 Dual Port SRAM을 사용하여 Host와 i860사이의 데이타 전송이 이루어지도록 설계하였다.

2. EPROM 메모리 접속부

DSP subsystem 은 back-end machine이며, host와의 communication을 담당하고 자체 system을 관장할 monitor프로그램이 필요하다. EPROM은 이 monitor program을 장착시켜 시스템의 초기화를 해줄 뿐만아니라, monitor 루틴을 RAM에 copy하여 정상적인 system의 운동을 담당하도록 하기위해서 필요하다. 사용된 EPROM은 16KB의 크기를 가지며, 8-bit access를 하므로 i860의 CS-8 모드로 동작하게 된다.

3. Local 메모리 접속부

본 시스템의 local memory는 main memory를 말한다. 특별히

local memory라 하는 이유는, 원래 시스템의 설계시에 parallel processing 구조를 고려하여 설계되었는데, 본 시스템이 parallel machine으로 확장될 경우 local memory를 가지는구조로 설계되었기 때문이다. Local memory는 4M DRAM을 사용하며, 지금 현재 16 Mbyte로 구성하였다. Local memory interface는 data access speed를 높이기 위해서 fast page mode access를 지원하며, read시에는 pipelined read가 가능하게 설계되었다. Fast page mode access란 row address는 한번만 주고 column address만을 변화시켜 빠르게 access하는 방법이다. 이렇게하면 average access time이 75 nsec 까지 가능하게 된다. 또 DRAM을 사용하므로 refresh를 위한 control circuit이 필요하며, 부가적으로 timer도 필요하다. Local memory는 program 및 data memory로 사용되므로 error에 치명적이다. 따라서 error를 check하는 one bit parity를 두었다.

4. Timer

Timer는 여러 방면에서 필요하다. 실제로 DRAM refresh signal을 만들어주고, multi-tasking을 위한 clock tick를 만들어 주는데 쓰이며, real-time clock으로도 사용될 수 있다. 여기에 사용된 device는 82C54-2인데, 이것은 내부에 3개의 독립적인 counter를 가지고 있으며, 이들은 여러 유용한 모드로 동작할 수 있다. 더구나 이 모든 모드는 software로 프로그래밍이 가능하다.

5. PIC(Programmable Interrupt Controller)

PIC는 i860 CPU로 들어오는 interrupt를 효과적으로 처리하기 위해서 필요하다. 이것은 intel 82C59A-2를 사용하였다. 이 device는

priority를 갖는 8-level의 interrupt를 받아들여 우선 순위에 따라 선택된 interrupt를 i860의 INT핀에 보낸다. 이때 우선 순위는 software적으로 프로그래밍이 가능하다. 현재 본시스템에서 masking하지 않고 받아들이는 interrupt는 PC interrupt와 Local memory의 parity error interrupt가 있다. 현재 사용되지 않고 reserve되어있는 interrupt는 timer(clock tick generation), frame grabber(real-time image grabbing) interrupt이며, 나중의 system 성능 향상을 위해서 준비해 두었다.

6. Local bus

i860 마이크로 processor, local 메모리, EPROM, dual port SRAM, timer, interrupt controller, 및 D-BUS interface block들은 모두 local bus와 직접 연결되어 있다.

i860이 local bus를 통하여 각 device에 access하려고 할때 local 메모리나 FM-BUS를 통한 frame buffer의 전송 시간을 빠르게 하기 위하여 fast page mode나 pipeline동작을 하고 있는 과정에서 다른 디바이스들과 버스상의 충돌을 제거하기위해 각 block들은 자신의 마지막 작업이 끝나지 않았음을 알려주는 ~ FBSY, ~LBSY 신호를 발생시킨다. ~FBSY는 frame buffer, ~LBSY는 local 메모리가 access 되고 있음을 알려준다. 그림 2.1.6.1 은 i860의 local bus timing diagram을 나타내었다.

7 Local 메모리 의 구성

i860 processor는 프로그램과 데이타 저장을 위하여 local 메모리를 갖는다. local 메모리는 64bit 로 구성되며 최소 8MB 에서 16MB 까지 구성할 수 있다. local 메모리는 2개의 bank를 구성하고

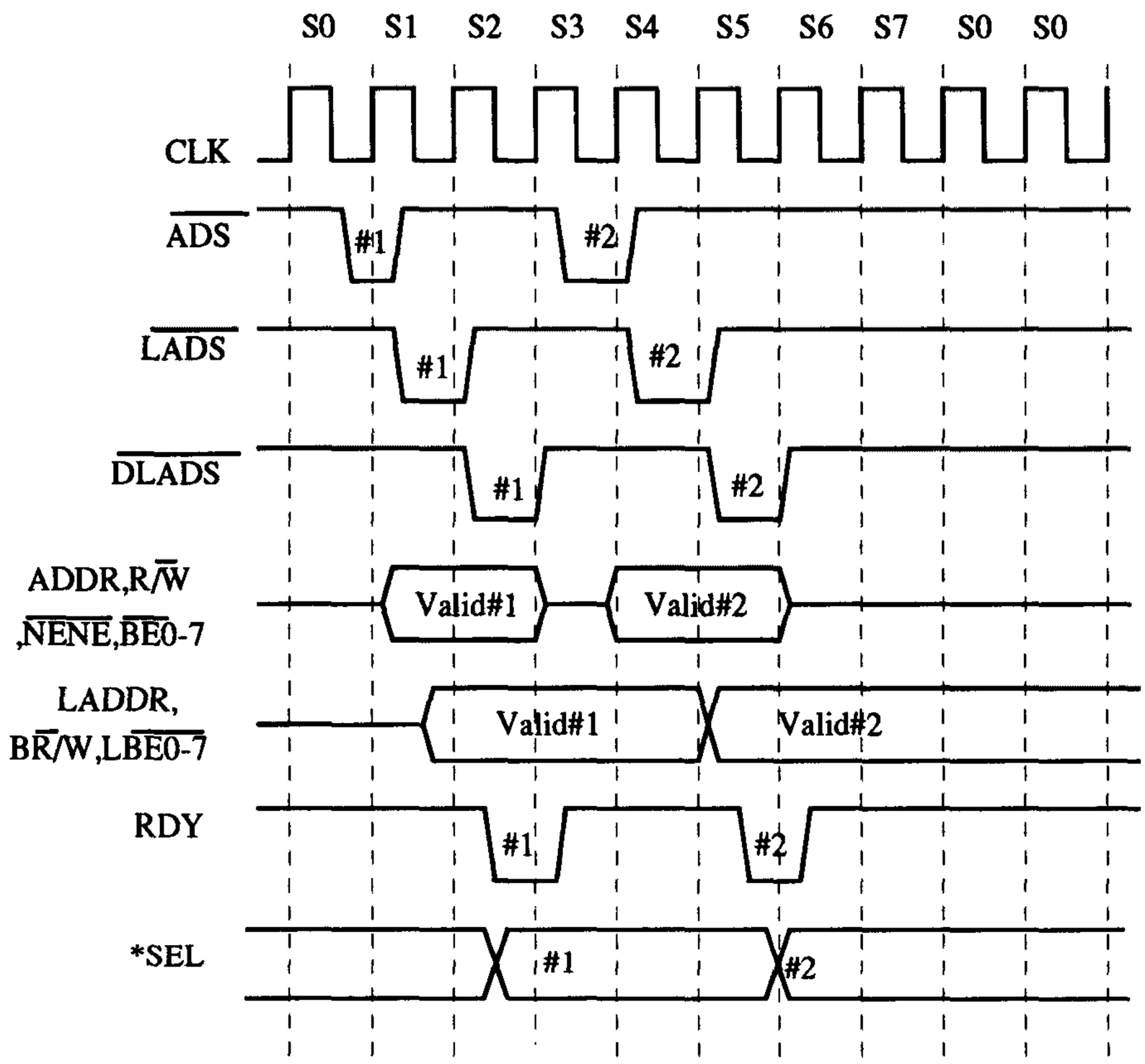
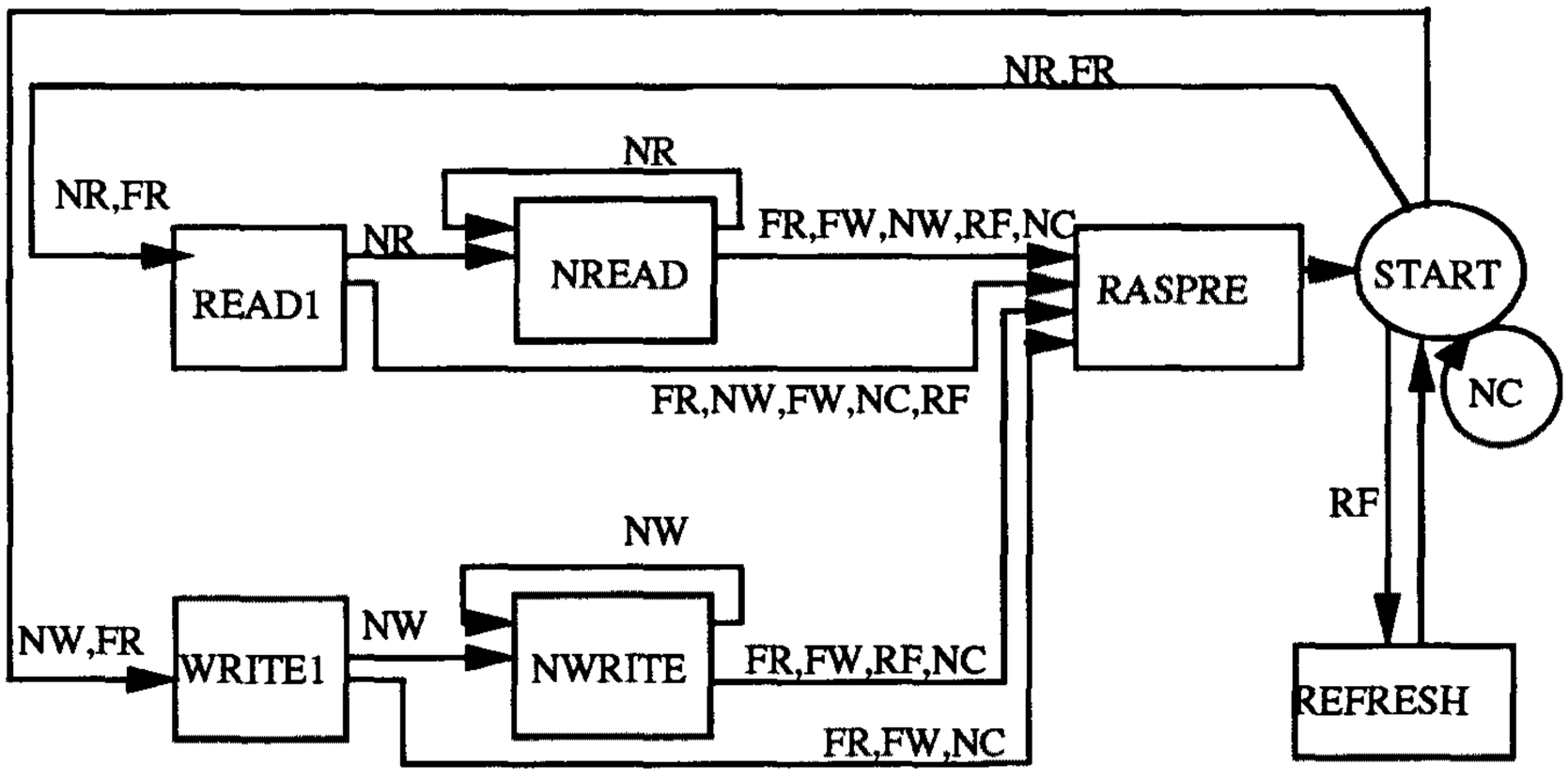


그림2.1.6.1 Local Bus Timing.

있고, 1개의 메모리 bank 의 size 는 1Mword이다. local 메모리의 access cycle은 block mode로 access할 경우 평균 3 clock 걸리며, random mode로 access하는 경우 read cycle의 경우 8 clock, write cycle인 경우 7 clock 걸린다.

Block mode로 local 메모리를 access할 경우 access 속도를 높이기 위하여 fast page mode access 방법과 pipelined access 방식을 결합하여 사용하였다. block mode access는 random access와 구분되어져야 하고 그에 따라 메모리 control도 달라져야 하므로 state machine 을 구성하여 각 state 마다 적절한 시간에 메모리와 버퍼들에 control 신호들을 만들어 주어야 한다. 그림 2.1.7.1에 local 메모리의 state diagram 을 block구조로 나타내었다. 각 block은 read와 write, pipelined access와 nonpipelined access DRAM refresh로 구분되어 나타나 있다. 초기 DRAM이 access 될때의 state machine 은 start block에서 시작되며 Read 또는 Write 신호에 의하여 state 가 분리되어 동작되고 access가 끝난뒤 다시 state block으로 되돌아오게 된다. 이런 sequence가 각 local 메모리 access때마다 반복되어지게 된다. 그림 2.1.7..2 에는 그림 2.1.7.1의 block을 시간별로 세분하여 sequence를 나타내었는데 각각의 sub-block 은 1 clock cycle(25ns)를 나타낸다. 전체 state 는 23개로 구성되며 이러한 state 를 나타내기 위하여는 flip flop 이 5개 필요하게 된다. 그림 2.1.7.3와 그림 2.1.7.4는 block modeaccess의 read와 write 경우에 대하여 어떻게 이루어지는지 여러가지 control 신호들과 함께 timing diagram 으로 나타낸것이다. 여기서 나타난 여러가지 control 신호들의 이름을 설명하면 다음과 같다.

~ADS : Address start 신호



FR : Far Read
NR : Near Read (Fast Page Mode)
FW : Far Write
NW : Near Write (Fast Page Mode)
RF : Refresh
NC : Not cycle

그림 2.1.7.1 State BlockDiagram of Local Memory Access.

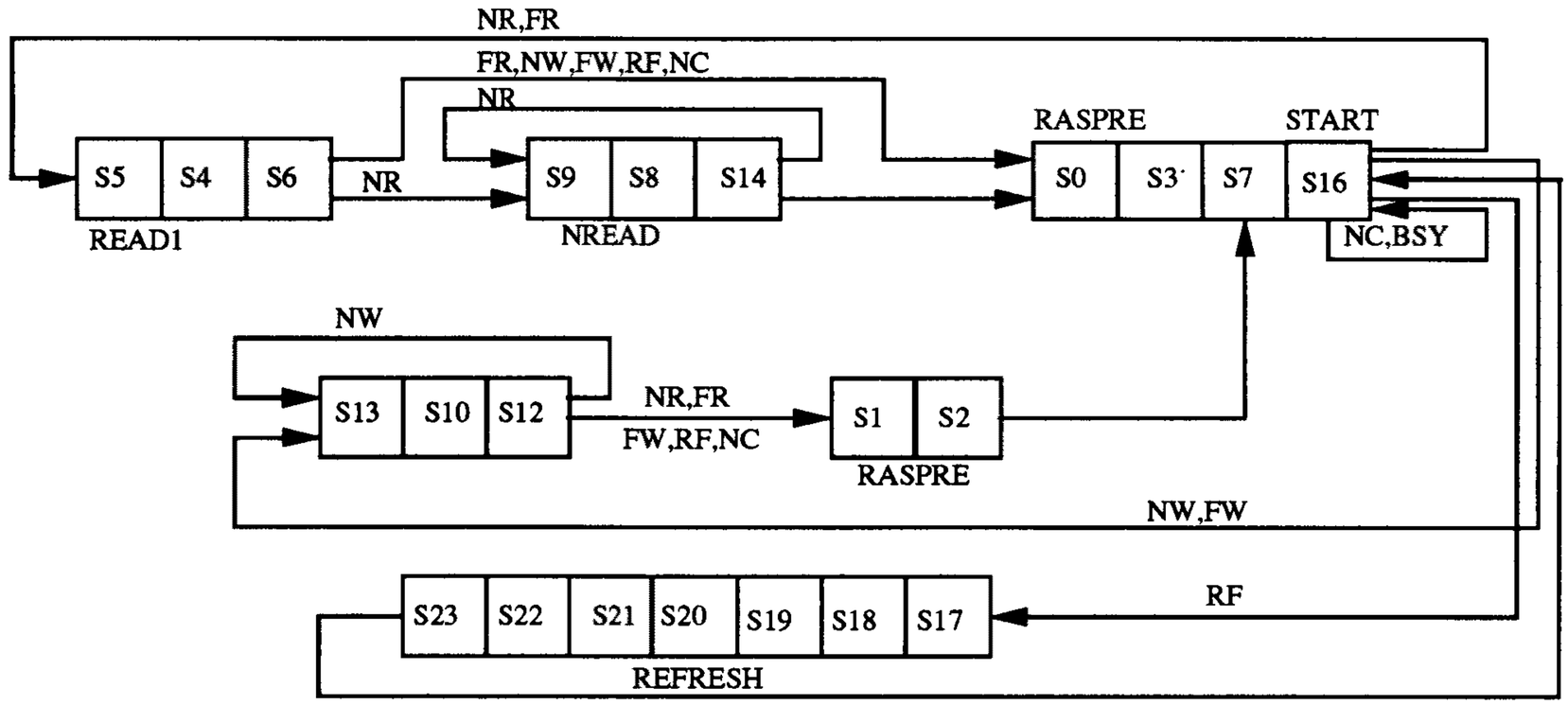


그림 2.1.7.2 State Diagram of Local Memory Access.

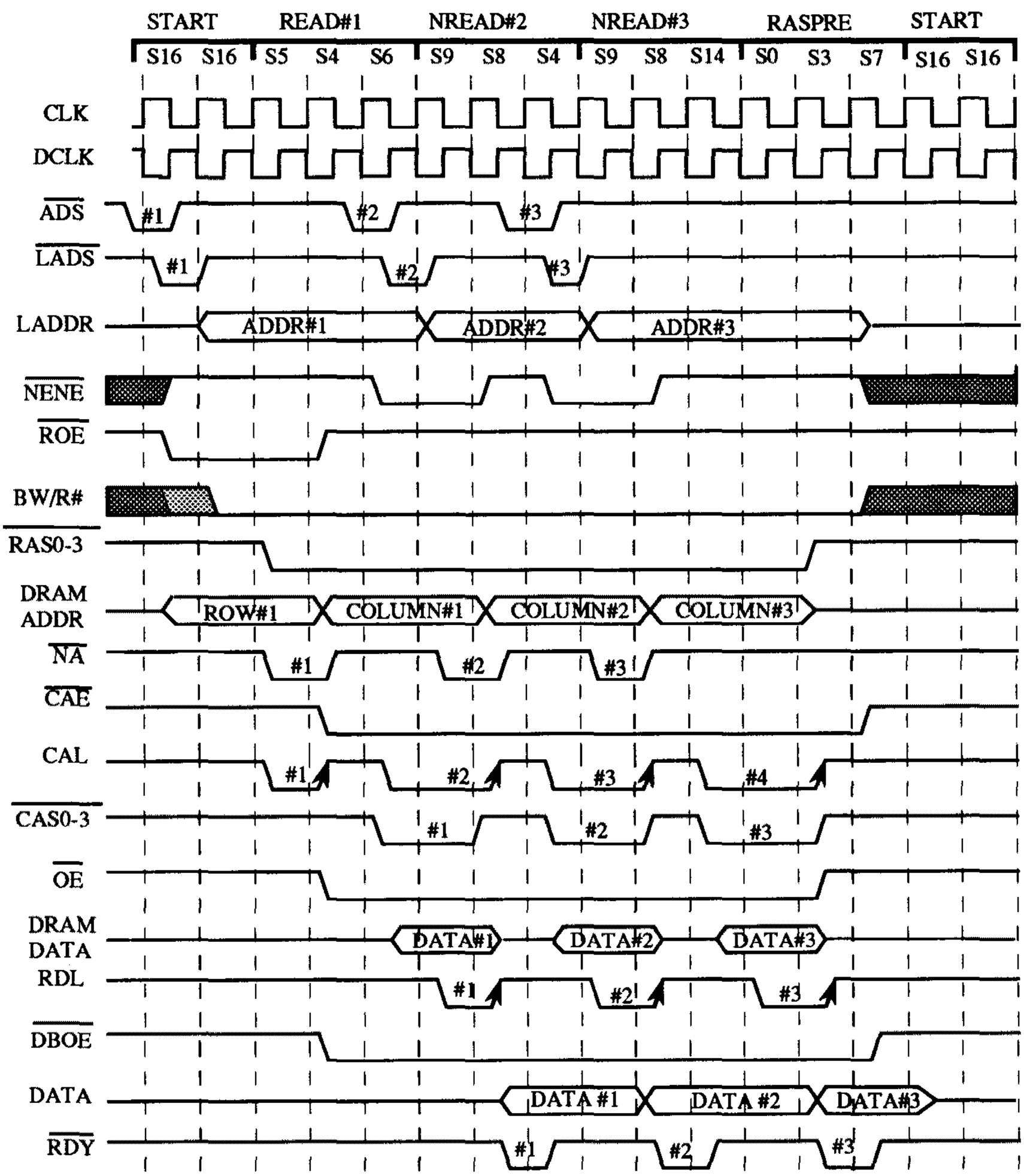


그림 2.1.7.3 Local Memory Near Read Cycle.

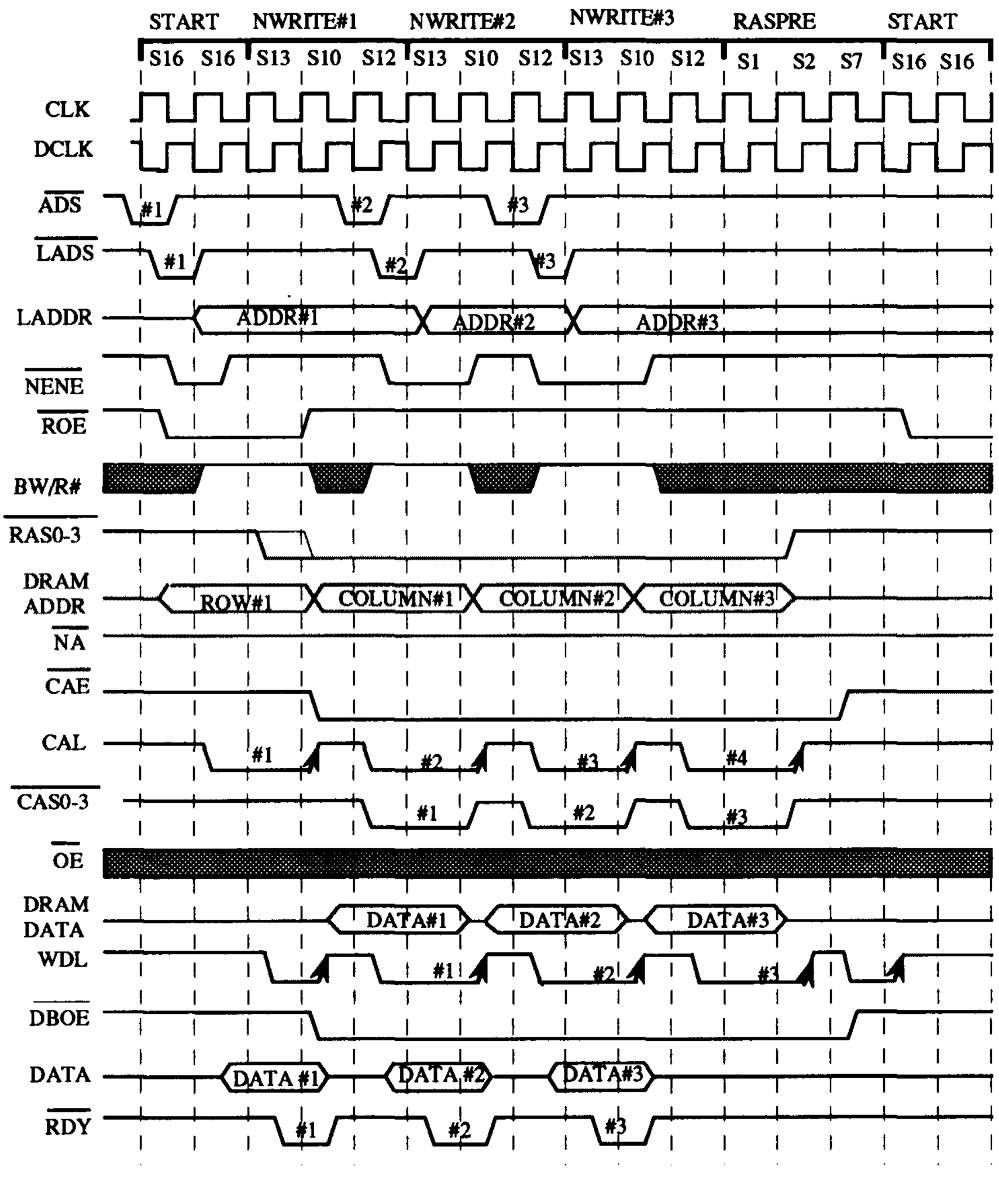


그림 2.1.7.4 Local Memory Near Write Cycle.

~LADS : latched ~ ADS 신호

LADDR : Latched Address 신호

~NENE : Near next address 신호로 local 메모리의 데이타가 앞에서 access한 데이타와 같은 Row address 를 갖는 영역에 존재함 을알린다.

~ROE : Row address output enable 신호로 low level 에서 row address가 어드레스 버퍼에 latch 되어 output enable 되도록 한다.

~COE : Column address output enable 신호로 low level 에서 column address 가 어드레스 버퍼에 latch 되어 output enable 되도록한다. DRAM 은 row address 와 column address가 multiplexing 되어져야 하므로 ~ROE 와 ~COE 는 동시에 low 가 되어서는 안된다.

~R/W : Read or write 신호로 low level 일때 read 이다.

CAL : Column address latch 신호로 rising edge에서 column address buffer에 column address 가 latch 되어지게 한다.

~NA : Next address 요구신호로 pipeline access를 가능하게 한다.

~RAS 0-3 : Row address strobe 신호로 DRAM 에 가해진다.

~CS : Chip seletion 신호로 DRAM device 를 선택한다.

~OE : Output enable 신호로 read sequence의 경우 DRAM 메모리 device의 데이타를 output enable 시킨다.

~RDL : Read data latch 신호로 read sequence의 경우 DRAM 데이타 출력을 데이타 버퍼에 latch 시키는 역할을

한다.

~WDL : Write data latch 신호로 Write sequence의 경우 i860 데이타 출력을 데이타 버퍼에 latch 시키는 역할을 한다. RDL과 WDL은 rising edge 에서 동작한다.

DRAM DATA : DRAM 에서 출력 또는 입력되는 데이타를 나타낸다.

~DATA : i860 processor에 데이타 read/write 가 이루어졌음을 나타내는 신호

그림 2.1.7.3와 그림 2.1.7.4는 block mode로 read 와 write가 일어날 경우 read sequence는 ~NA 신호에 의하여 pipeline 되어 read가 이루어지지만 write sequence는 ~NA 가 항상 "high" 로 되어 pipelined write 가 일어나지 않고 이루어지게 된다. Block mode로 데이타를 read 또는 write 할 경우 평균적으로 걸리는 시간은 3 clock이 된다. (무한히 많은 data 를 access할 경우의 평균 access 시간)

그림 2.1.7.5은 DRAM의 refresh sequence를 나타낸것으로 CAS-before-RAS 방식을 사용하였는데, DRAM내부의 counter 를 사용하여 refresh row 어드레스를 발생시켜 DRAM 을 refresh 하도록 하였다. Refresh 는 16ms 마다 timer 로부터 DRAM controller 로 refresh 요구 (~RFREQ)신호가 인가되어진다. ~RFREQ가 active가 되면 곧바로 DRAM을 refresh 시켜야 하는데 현재 DRAM 을 access를 하고 있으면 현재 진행중인 access를 될수 있는한 단기간에 마치고 refresh 가 이루어지도록해야한다. Pipelined access 가 이루어 지고 있는 경우는 pipeline access를 중단시키고 refresh sequence로 돌아가서 refresh 를 하고 다시 access를 시작하여야 한다.

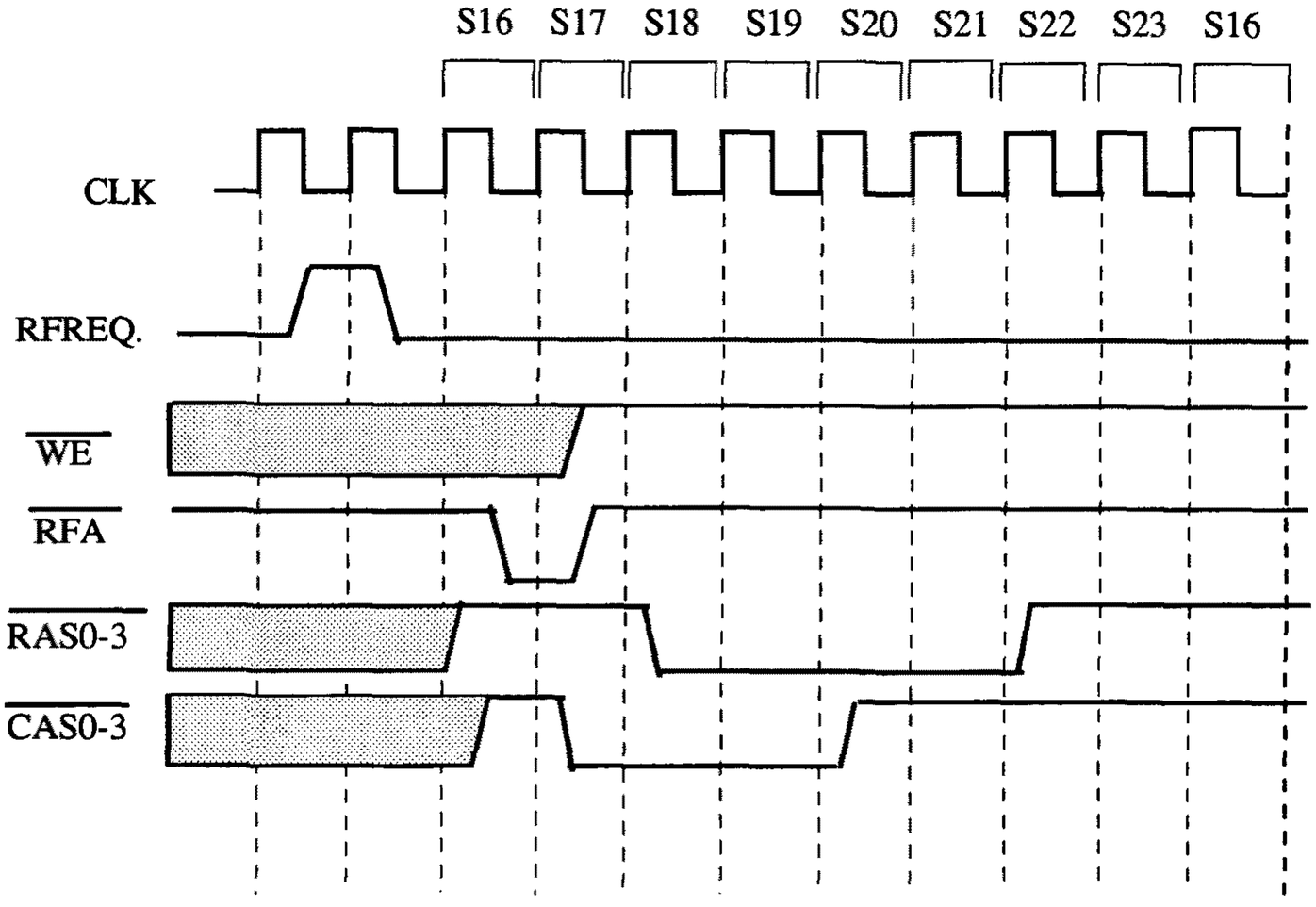


그림 2.1.7.5 Refresh Timing

각 메모리 bank의 크기는 1Mword(8MByte) 로 구성되며 1개의 cluster 마다 1MByte의 parity 메모리가 parity error 검출을 위하여 존재한다. State machine 은 메모리 access cycle의 통제를 위해 있는데 현재의 state 와 입력을 바탕으로 하여 다음 state 를 결정한다. Local BUS controller는 local bus에 연결되어 있는 신호를 만들어준다. Data buffer 는 64bit bidirectional tri-state buffer 로 구성된다. Write enable buffer 는 각 메모리에 write 신호를 만들어주고, column address buffer 와 row address buffer 는 각각 column 과 row address 를 만들어주는데 두가지 address 는 동시에 출력될 수 없고 상호 배제되어야 한다. 검출된 parity error flag은 logic을 통하여 error flag을 setting 하고 i860 processor에 interrupt 를 발생시킨다. i860 processor는 interrupt 를 받은뒤 적절한 조치이후 parity error flag 를 clear 시키게 된다. Local 메모리 controller는 DRAM 메모리 bank에 ~RAS, ~CAS, ~OE 등의 신호를 발생시키며, Local buffer controller는 데이타와 어드레스 버퍼들을 control하는 신호를 만든다. 발생되는 control 신호들은 state machine 에서 발생되는 state 에 따라서 순차적으로 이루어진다.

8. D-BUS interface 설계

칼라모니터에 프로세싱한 데이타 또는 영상을 도시하기 위하여 그래픽/영상 도시부의 frame buffer를 사용하여야 한다.

각 메모리 bank는 64bit 로 구성되어 있으며 512 x 512 의 row x column 공간을 갖는다. frame buffer의 어드레스는 9 bit 로 구성되어 있으며 row 어드레스와 column 어드레스가 multiplex 되어서 VRAM에 가해지게 되므로 메모리의 모든 영역을 random 하게 access 할 수 있다. ~FRAS0-1, ~FCAS0-1 은 각각 frame buffer의 필요한 bank를 선택할 수 있는데 이것은 그림2.1.8.1에 나타나 있는 것

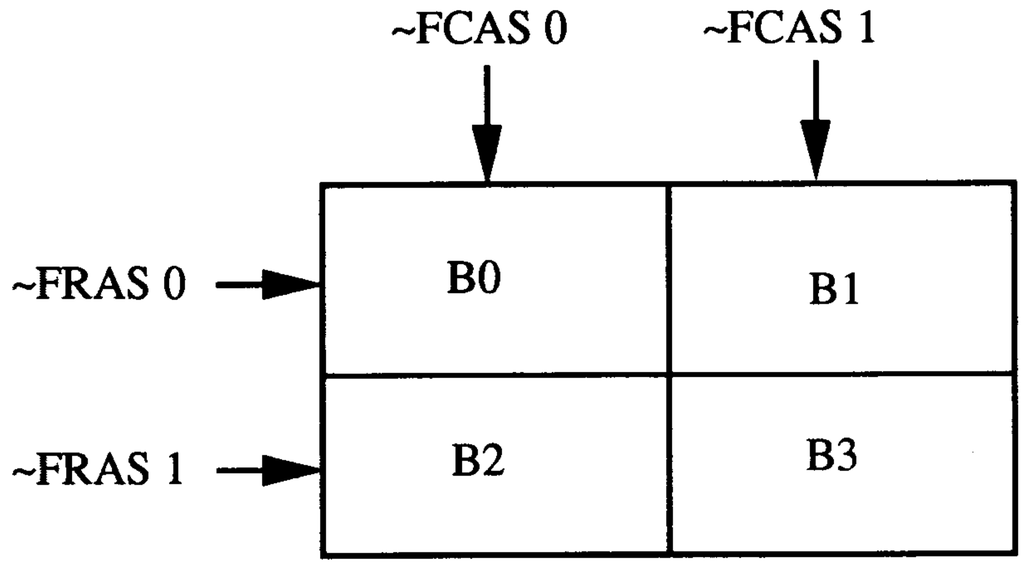
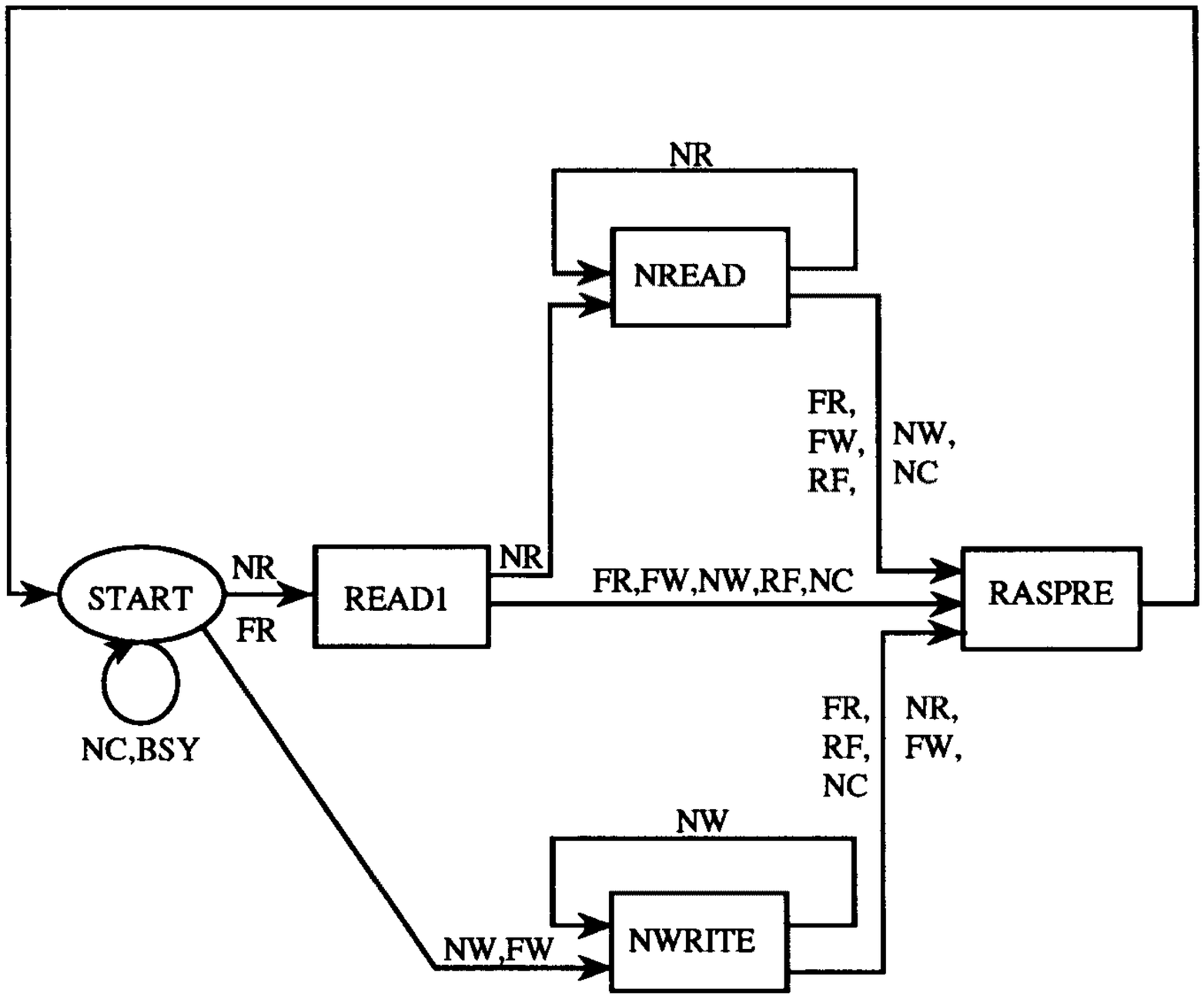


그림 2.1.8.1 Frame buffer의 bank 구성

처럼 bankB0에 ~FRAS0와 ~FCAS0 가 이용되고 B3 에는 ~FRAS1 과 ~FCAS1 이 이용되어 선택된다. Byte write enable(~ FBE0 -7)신호는 byte별 쓰기를 가능하게 한다. Frame buffer의 데이타의 크기는 64 bit 로 표현되는데 64bit 는 2 pixel 을 구성하고 있다. 데이타를 64 bit 로 access하면, 동시에 2 pixel 의 데이타를 access하는 결과가 되므로 1pixel 씩을 access할 수 있는 경우보다 2 배의 전송속도를 높일수 있다. VRAM에 대한 screen refresh또는 DRAM refresh를 수행하기 위하여 ~FRFINT가 영상출력부의 TMS34020에 의하여 정기적으로 발생된다.

Frame buffer access는 fast page mode access 방식과 block access 방식을 같이 사용할 수 있다. Fast page mode 방식은 read인 경우 pipeline 되어 이루어지지만 write일 경우에는 nonpipeline 으로 동작한다. block mode access는 random access와 구분되어져야 하고 그에 따라 control도 달라져야 하므로 별도의 state machine을 구성하여 각 state마다 메모리와 버퍼들에 control 신호들을 구성시켰다. 그림 2.1.8.2에 local 메모리의 state diagram 을 block구조로 나타내었다. 각 block은 read, write, pipelined mode와 nonpipelined 로 구분되어 나타나 있다. State diagram의 sequence는 각 프레임 메모리의 access 때마다 반복되어지게 된다. 전체 state 는 15개로 구성되며 refresh 를 제외한 state는 local memory의 경우와 같으며 이러한 state 를 표현하기 위해서는 최소한 flip flop 이 4개 필요하게 된다. 그림 2.1.8.3와 그림 2.1.8.4는 block mode 및 random mode access가 read 와 write 의 경우에 대하여 어떻게 이루어지는지 여러가지 control 신호들과 함께 timing diagram 으로 나타낸것이다. 여기서 나타난 여러가지 control 신호들은 다음과 같다.



FR : Far Read
NR : Near Read
FW : Far Write
NW : Near Write
RF : Refresh
NC : Not Cycle

그림 2.1.8.2 State Diagram of Frame Memory Access

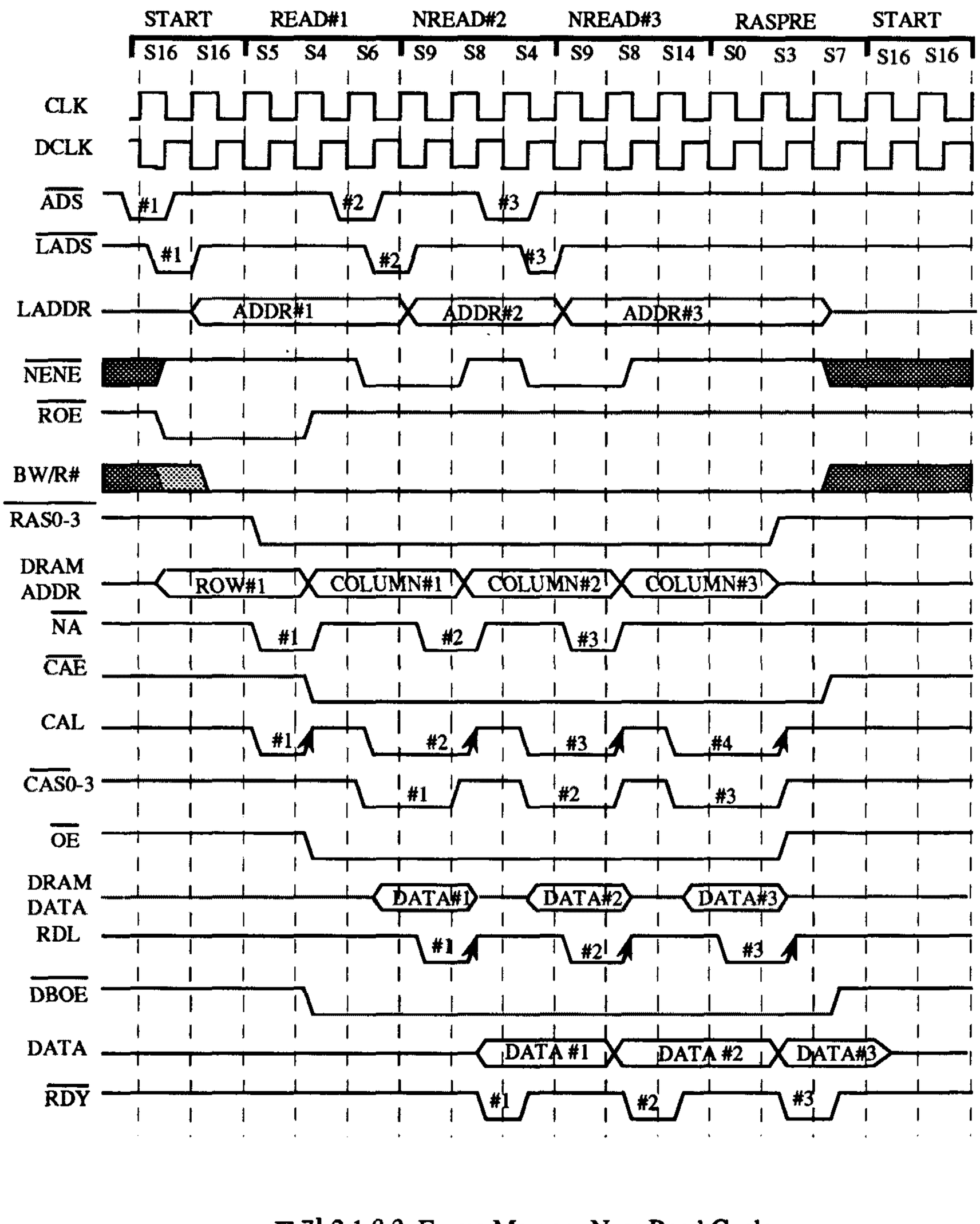


그림 2.1.8.3 Frame Memory Near Read Cycle.

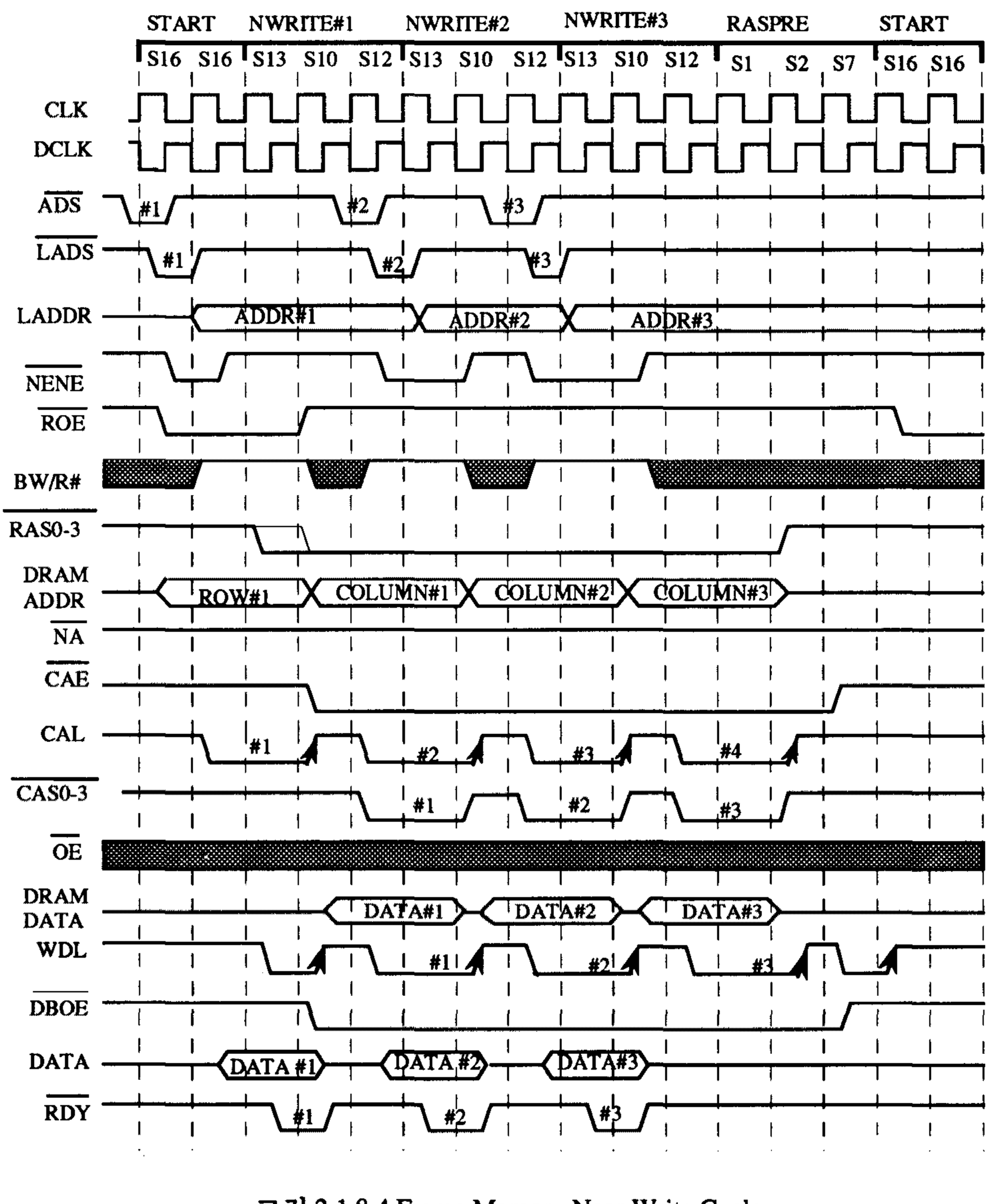


그림 2.1.8.4 Frame Memory Near Write Cycle.

~ADS : Address start 신호

~LADS : Latched ~ADS 신호

LADDR : Latched Address 신호

~NENE : Near next address 신호로 local 메모리의 데이타가 앞에서 access한 데이타와 같은 row address 를 갖는 영역에 존재함을 알린다.

~ROE : Row address output enable 신호로 low level에서 row address가 어드레스 버퍼에 latch 되어 output enable 되도록 한다.

~COE : Column address output enable 신호로 low level에서 column address 가 어드레스 버퍼에 latch 되어 output enable 되도록 한다. VRAM은 row address 와 column address 가 multiplexing 되어져야 하므로 ~ROE 와 ~COE 는 동시에 low 가 되어서는 안된다.

~R/W : Read or write 신호로 low level 일때 read 이다.

CAL : Column address latch 신호로 rising edge 에서 column address buffer 에 column address 가 latch 되어지게 한다.

~NA : Next address 요구신호로 pipeline access를 가능하게 한다.

~RAS 0-3 : Row address strobe 신호로 VRAM 에 가해진다.

~CAS0-3 :Column address strobe 신호로 VRAM device 를 선택 한다.

~OE : Output enable 신호로 read sequence의 경우 VRAM

메모리 device의 데이타를 output enable 시킨다.

RDL : Read data latch 신호로 read sequence의 경우 DRAM 데이타 출력을 데이타 버퍼에 latch 시키는 역할을 한다.

WDL : Write data latch 신호로 write sequence의 경우 i860 데이타 출력을 데이타 버퍼에 latch 시키는 역할을 한다. RDL 과 WDL 은 rising edge 에서 동 작한다.

VRAM DATA : VRAM 에서 출력 또는 입력되어지는 데이타를 나타냄.

DATA : i860 processor에 데이타 read/write 가 이루어졌음을 나타내는 신호로 active low 임.

Static column mode 방식과 fast page mode 사이의 차이점은 ~CS와 ~CAS 의 차이로 나타나는데 read 일 경우 ~CS 는 "low" 로 고정된 상태에서 어드레스만을 바꾸는 반면에 ~CAS는 어드레스를 바꾸면서 ~CAS신호도 같이 toggling 되어져야 한다는 것이다.

제2절 Graphic System Processor(GSP)의 구조

1. Graphic System Processor의 목적 및 기능

본 연구과제에서 개발한 고화질 의료용 영상 처리 시스템의 구성에 있어, 두개의 고속 Processor로 구성된 영상 처리부에서 처리된 의료용 영상 데이타를 고해상도의 real color monitor에 실시간으로 도시하는 기능을 담당하는 역활을 GSP에서 수행한다. 이를 위해서는 DSP와 GSP간의 고속 FM-BUS의 구성을 필요로 하며, 이를 통한 원활한 데이타 전송이 이루어지도록 설계가 진행되었다. 또한 DSP와 독립적으로 동작할 수 있도록 GSP 또한 host와 접속되어 있으며, 이 경우를 위해 8 bit의 의료용 영상을 도시할 수 있도록 구성되었다.

2. Graphic System Processor의 제원 및 구조

두 개의 고속 Processor를 사용한 DSP부에서 처리된 영상을 도시하고, GSP 자체적으로 8 bit의 영상을 도시하기 위하여 GSP에서 요구되는 제반요소를 살펴보면 다음과 같다.

-. Host : PC/AT급이상

-. Main Processor : TMS34020(32MHz)

-. Frame Memory : 1024x2048x24bits(R(8bit),G(8bit),B(8bit))

-. Overlay Memory : 1024x2048x8bits

-. Display look up table과 Video DAC-Bt463 RAMDAC

-. DSP Interface

이러한 구성요소들을 결합하여 GSP부가 구성되며, 그림 2.2.1에 GSP부의 구성상태 block diagram으로 나타내었다.

그림 2.2.1에서 GSP부의 processor로 TMS34020을 선택하였다. TMS34020의 경우 Host interface를 위한 pin들이 따로 제공되며, screen refresh와 DRAM refresh가 자동적으로 수행된다. 또한 DRAM refresh의 경우 외부 device 또는 processor에 의해 수행이 pending될 수 있다는 특징을 가지고 있다. 그외에도 일반적인 그래픽 processor에 비하여 처리속도가 빠르다. 이러한 특징들에 의하여 Host interface를 손쉽게 할 수 있으며 frame buffer와 Overlay buffer에 대한 screen refresh에 신경을 쓸 필요가 없어지게 된다. 또한 DRAM refresh의 pending은 DSP부와의 interface에서, DSP부의 메모리 cycle을 파괴함이 없이 수행될 수 있다는 것을 보장할 수 있다.

- Host Interface부

Host로 사용된 PC-bus로 부터 Program 또는 영상의 전송 등 Host와의 원할한 통신을 위하여 Host 인터페이스를 필요로 하며 memory-mapped I/O 방식과 I/O-mapped I/O를 혼합한 방식을 취하고 있다. GSP board에서 채용하고 있는 Processor인 TMS34020의 경우 Host interface를 원할히 하기 위하여 많은 pin들이 제공되고 있어, 비교적 간단한 interface logic을 꾸밀 수 있다.

-. FM-BUS의 Interface 설계

GSP부는 기본적으로 여러개의 DSP부에서 처리된 의료용 영상을 real color monitor에 도시하기 위한 부분들로 구성된다. DSP부에서 처리된 영상들은 DSP부의 FM-BUS를 통해서 GSP부의 frame buffer로 옮겨지게 된다. GSP부 processor(TMS34020)와 DSP부간의 handshaking 신호을 교환함으로써 상호간의 메모리 cycle이 파괴되는 것을 방지

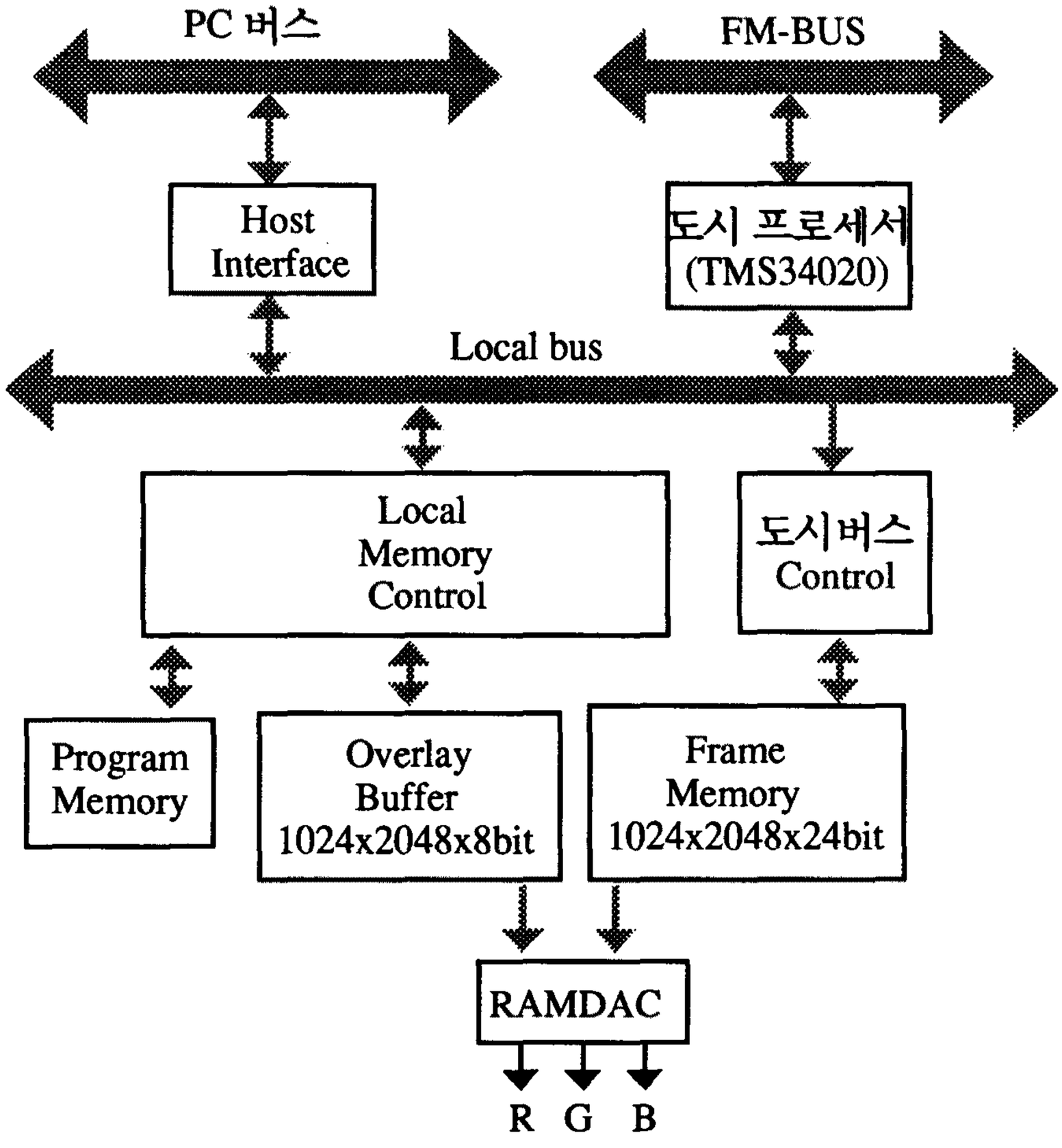


그림 2.2.1 그래픽/영상 도시부의 블럭 다이아그램

하도록 설계하였다.

-. GSP부의 local memory 및 interface부

GSP부 processor(TMS34020)는 여러 종류의 local memory들을 가지게 된다. 이들 frame buffer , overlay buffer, program memory, RAMDAC등 이다. Overlay buffer는 GSP부만으로 독립하여 사용할 경우 frame buffer로서의 역할을 하게 된다. Program memory는 TMS34020이 수행할 프로그램이 저장된다.

Video timing interface부

GSP부 processor(TMS34020)은 그래픽용 processor로서 video timing에 관계되는 여러개의 신호 핀들을 가지고 있다. 이들은 analog video신호의 sync및 blank등에 관계되는 신호들로써 이들에 의해 video timing에 대한 interface가 원활히 이루어질 수 있게 된다. 이들 pin들은 TMS34020내부의 control register들의 내용을 프로그램 해줌으로써 여러가지의 동작mode를 가질 수 있으며 입력, 또는 출력으로 사용될 수 있다.

이상에서 GSP의 각 구성 block들에 대해 간략하게 살펴 보았으며 다음 절에서 이들 블럭의 설계방식 및 동작에 대해 자세하게 살펴보도록 한다.

가. Host Interface부의 구성

GSP의 local 메모리 또는 도시 processor의 control register등과 host 사이의 데이타, command 및 status 정보등을 교환하기 위해서 host 인터페이스가 필요하며, host로 사용된 PC/AT버스와 GSP의

main processor인 TMS34020에서 제공하고 있는 host interface부의 신호를 유기적으로 결합함으로써 인터페이스부를 구성할 수 있다.

표 2.2.1과 표 2.2.2는 각각 PC/AT bus의 신호구성과 TMS34020의 host interface부의 신호구성을 나타낸 것으로, PC/AT bus는 크게 데이타버스, 어드레스버스, 데이타전송 control 신호, 인터럽트 신호 및 유틸리티 신호들로 구성되어 있으며, TMS34020의 host이스 신호 그룹은 데이타 버스, 어드레스 버스 및 데이타전송 control신호로 구성되어 있다.

PC/AT버스는 기본적으로 16 bit address공간(direct addressing)과 16 bit의 데이타 버스가 제공되므로, TMS34020 32bit adrrss 영역의 access및 32bit의 데이타 전송을 위해서는 메모리 mapped I/O와 I/O mapped I/O 방식을 혼합한 interface방식이 요구된다.

그림 2.2.2는 host인터페이스부의 블럭 diagram을 나타낸 것으로 크게 Address decoding 및 control 블럭, 양방향 데이타 transceiver 블럭, 그리고 I/O mapped I/O 를 위한 address lactch 블럭 및 Interrupt control 블럭으로 구성되어 있다.

어드레스 디코딩 블럭은 PC/AT bus의 address 및 데이타 전송 control신호를 입력으로 받아 address를 디코딩하여 TMS34020의 data transfer control신호들을 생성한다. 그림 2.2.3에는 어드레스 디코딩을 위한 Host의 어드레스 map을 나타낸 것으로 080000h부터 09FFFFh 사이의 I/O channel 영영중 하나를 선택하여 I/O mapped I/O transfer를 하며, 0C0000h ~ 0DFFFFh 영역중 32 KByte segment 내에서 memory mapped I/O를 위한 영역이 할당되어 있다.

양방향 트랜시버 블럭의 구성은 PC/AT의 data bus와 GSP의 데이타 버스의 접속을 위한 부분으로 PC/AT의 16 비트 데이타 버스를

Signal Name	I/O	Description
HA5-HA31	I	27 host-address input signals. A host can access a long-word by placing the address on these lines. HA5-HA31 correspond to the LAD5-LAD31 signals that output the address to the local memory.
HBSO-HBS4	I	4 host byte selects. The byte selects identify which bytes within the long-word are being selected.
~HCS	I	Host chip select. A host drives this signal low to latch the current host address present on HA5-HA31 and the host byte selects on HBSO-HBS3. This signal also enables host access cycles to the TMS34020 I/O registers or local memory. During the low-to-high transition of ~RESET, the level on the HCS input determines whether the TMS34020 is halted (~HCS is high for host-present mode) or whether it begins executing its reset service routine(~HCS is low for self-bootstrap mode).
HDST	O	Host data strobe. The rising edge of this signal latches data from the TMS34020 local address space to the external transceivers on host read accesses. It can be used in conjunction with HRDY to indicate that data is valid in the external transceivers.
~HINT	O	Host interrupt. This signal allows the TMS34020 to interrupt a host by setting the INTOUT bit in the HSTCTLL I/O register. This signal can also be used to interrupt the host if a BUSFLT or RETRY occurs due to a host access cycle.
~HOE	O	Host-data output enable. This signal enables data from the external transceivers to the TMS34020 local address space on host write cycles. HOE can be used in conjunction with HPDY to indicate data has been written to memory from the external transceivers.
HRDY	O	Host ready. This signal is normally low and goes high to indicate that the T MS34020 is ready to complete a host-initiated read or write cycle. A host can use HRDY logically combined with HDST and ~HOE to determine when the local bus access cycles have completed.
~HREAD	I	Host read strobe. This signal is driven low during a read request from a host processor. This notifies the TMS34020 that the host is requesting access to local memory or to the I/O registers. ~HREAFD should not be asserted at the same time that ~HWRITE is asserted.
~HWRITE	I	Host write strobe. This signal is driven low to indicate a write request by a host processor. This notifies the TMS34030 that a write request is pending. The rising edge of ~HWRITE is used to indicate that the data provided by the host in the external data transceivers can be written. ~HWRITE should not be asserted at the same time ~HREAD is asserted.

표 2.2.1. TMS34020의 host interface신호 그룹

Signal Name	I/O	Description
SA0 ~SA19	O	20 Output address signal: These 20 lines allow access of up to 1MB of memory by direct addressing SA0~SA19 one gated on the system bus when BALE signal is high and one latched on the falling edge of BALE
LA17~LA23	O	These signals(unlatched)one used to address memory and I/O devices. They give the system up to 16MB of addressability.
SD0~SD15	I/O	16I/O data signal: D0 is the least-significant bit and D15 is the most significant bit
CLK	O	System clock signal: It is synchronous microprocessor cycle clock
RESET_DRV	O	Reset Drive signal
BALE	O	Buffered Address Latch Enable signal : Address SA0~SD19 one latched with the falling edge of BALE
~I/O_CH_CK	I	I/O channel check signal: This signal provide the system board with parity(error) information about memory or devices on the I/O channel.
I/O_CH_RDY	I	I/O channel Ready signal: It is pulled low (not ready) by a memory or I/O device to lengthen I/O or memory cycle.
IRQ3~IRQ7 IRQ9~IRQ12 IRQ14,IRQ15	I	Interrupt Request signals: These are used to signal the micro processor that an I/O device needs attention. The interrupt Request are prioritized, ITQ9 is highest and IRQ7 is lowest.
~IOR	I/O	I/O Read signal: It instructs an I/O device to drive its data onto the data bus
~I/OW	I/O	I/O Write Signal instructs an I/O device to read the data off the data bus.
~SMEMR ~MEMR	O I/O	These signal instruct the memory device to drive data onto the data bus. ~SMEMR is active only when the memory decode is within the low 1 M of memory space. ~MEMR is active on all memory read cycle.
~SMEMW ~MEMW	O I/O	These signals instruct the memory devices to store the data present on the data bus.
DRQ0~DRQ3 DRQ5~DRQ7	I	DMA Request signal: These signals are asynchronous channel requests used by peripheral devices and a microprocessor to gain BMA service.
~DACK0--~DACK3 ~DACK5--~DACK7	O	DMA Acknowledge Signals one used to acknowledge DMA reguests.
AEN	O	The "Address Enable" Signal is used to degrate the microprocessor and other device from the I/O channel to allow DMA transfers to take place.
~REFRESH	I/O	This signal is used to indicate a refresh cycle.
T/C	O	The 'terminal count' signal provides a high pulse when the terminal count for any DMA channel is reached.

SBME	I/O	The 'system Bus High Enable' signal indicates a transfer of data on the upper byte of data bus.
~MASTER	I	This signal is used with a DRQ line to gain control of the system
~MEM CS16	I	The 'Memory 16-bit chip select 'signal indicates to the system that the present data transfer is a 1 wait-state cycle.
~I/O CS16	I	The 'I/O 16 bit chip select' signal indicates to the system that the present data transfer is a 16 set, 1 wait-state, I/O cycle.
OSC	O	The 'Oscillator' signal is a high-speed clock with a 70-nanosecond period
OWS	I	The 'Zero Wait State' signal tells the microprocessor that it can complete the present bus cycle without any additional wait cycles.

표 2.2.2 PC/AT 버스의 신호 구성

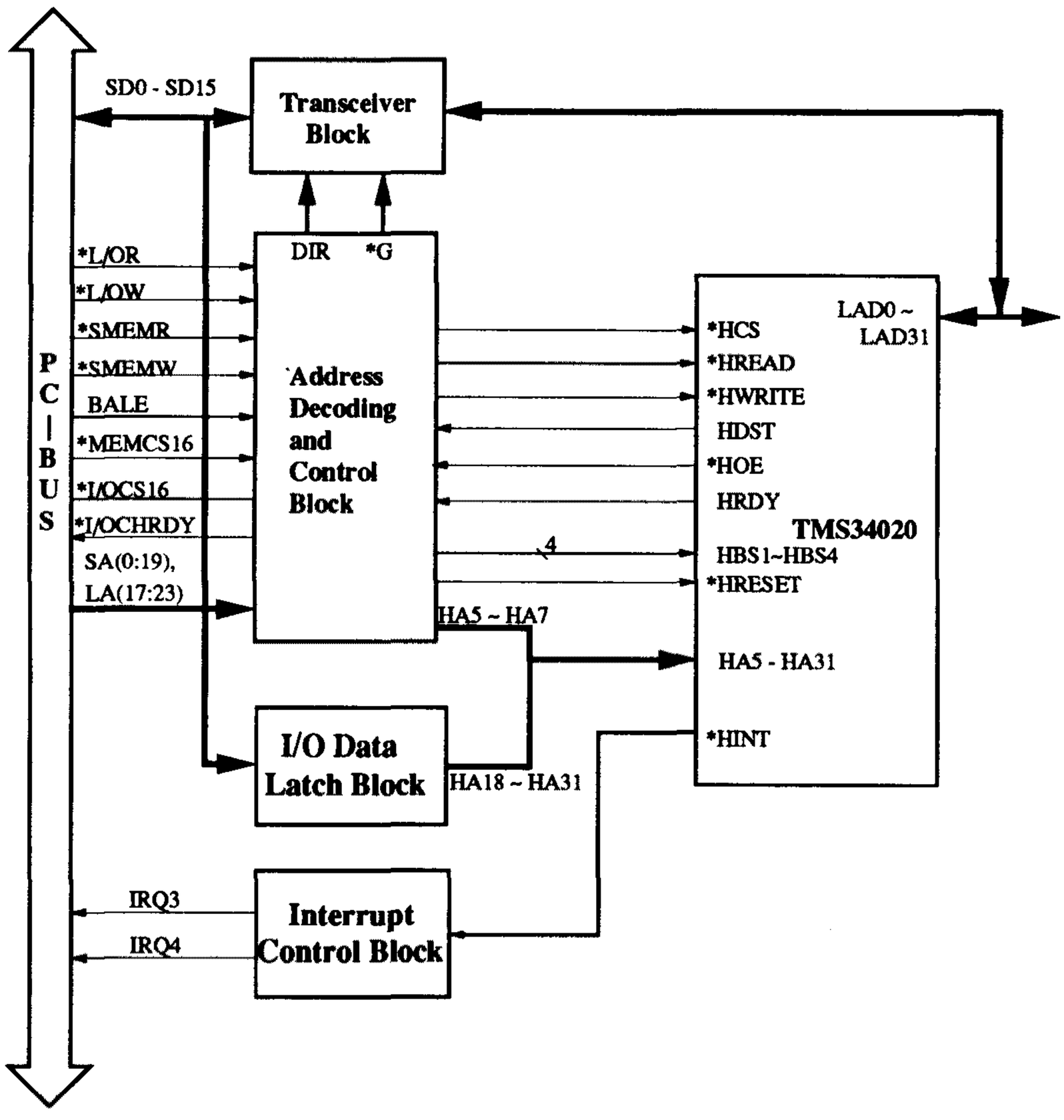


그림 2.2.2 Host Interface부의 Block Diagram

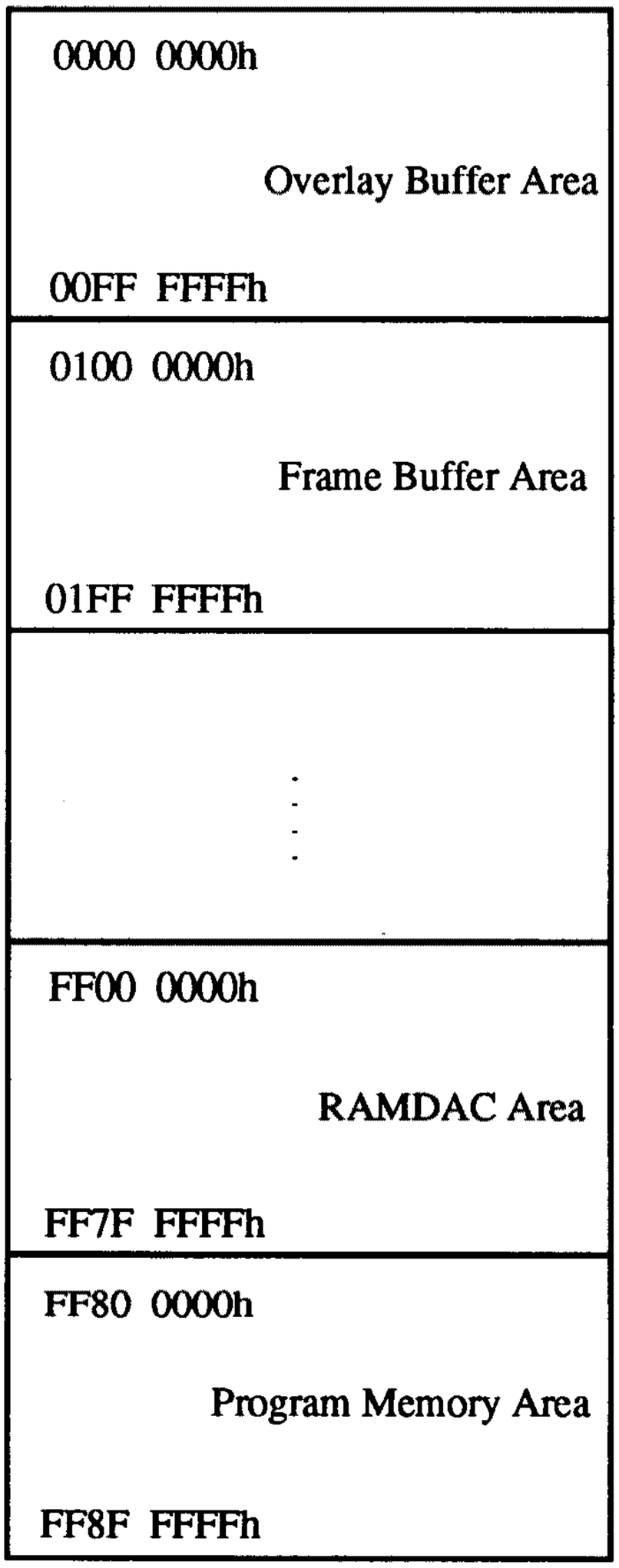


그림 2.2.3 화상 도시부의 local memory map

32 비트 크기의 GSP data버스에 interleaving함으로써 32 비트 data transfer를 수행한다.

어드레스 레치 블럭은 PC버스의 16 비트 address 영역을 GSP의 32 비트 영역으로 확창시키기 위한 블럭으로서, I/O channel로부터 data를 전송받아서 이를 GSP의 Host address의 상위 어드레스로 사용한다.

끝으로 인터럽트 control블럭은 GSP로부터 host로 전달되는 interupt 신호를 handling하는 부분으로써 IRQ3 또는 IRQ4 중 하나를 interrupt request 로 사용한다.

나. GSP의 local memory및 인터페이스 부의 구성

화상도시부 프로세서인 TMS34020은 여러 종류의 local memory들을 가지게 된다. 이들은 frame buffer, overlay buffer, program memory, RAMDAC등이다. Overlay buffer 화상도시부만으로 독립하여 사용할 경우 frame buffer로서의 역할을 하게 된다. Program memory는 TMS34020이 수행할 프로그램이 저장된다. RAMDAC는 Brooktree사의 Bt463모델로서 X-window를 지원할 수 있는 H/W구조를 가지고 있으며 32 bit의 true color pixel입력을 처리하게 된다. 그림 2.2.3은 화상도시부 프로세서의 local 메모리에 대한 메모리 map을 보이고 있다. 또한 그림 2.2.4 는 이들 memory map에 따라 각각의 메모리 device들에 대한 선택을 하기 위한 address decoding에 대하여 보이고 있다.

위에서 언급한 각각의 메모리들에 대한 즉 메모리 사이클은 기본적으로 TMS34020의 address를 decoding함으로써 선택된 메모리에 대하여 이루어지게 되며 각각의 메모리에 대한 control 신호은 위의 메모리 선택신호와의 조합들에 의하여 만들어지게 된다. 지금부터는

LAD25	LAD24	LAD23	Selection signal
L	L	X	overlay buffer selection
L	H	X	frame buffer seclection
H	H	L	RAMDAC selection
H	H	H	program memory selection

그림 2.2.4 Local 메모리 선택을 위한 address decoding

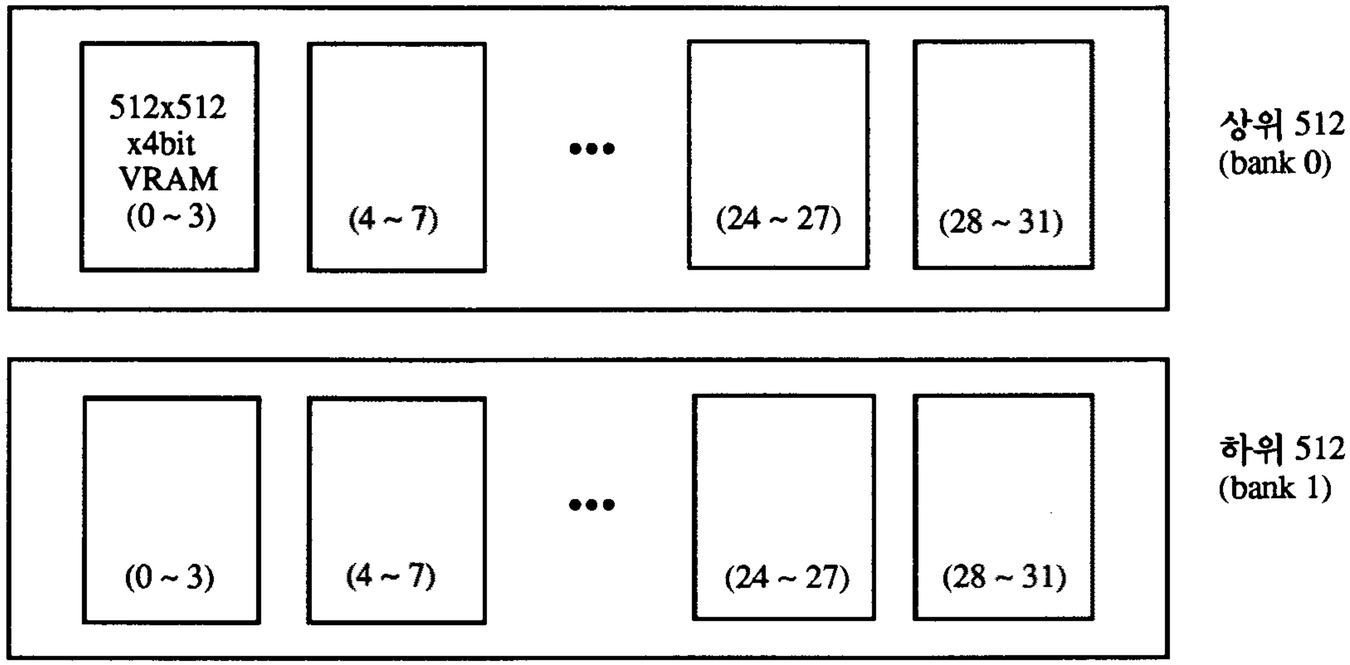
각각의 메모리들에 대한 control신호들의 생성에 대하여 알아보겠다.

(1) Overlay buffer인터페이스 설계

Overlay buffer는 화상도시부 자체의 frame buffer이며 신호처리부와 같이 연결되었을때에는 overlay plane으로서의 역할을 하게 된다. overlay buffer는 1024 line 2048 pixel의 크기를 가지며 한 pixel의 depth는 8 bit이다. 따라서 1 Mbit의 VRAM으로 text buffer를 구성할 경우 그림 2.2.5와 같은 구성이 된다. 이 그림에서 알수 있듯이 overlay buffer의 윗쪽 512line및 아래쪽 512 line에 대하여 다시 선택할 수 있어야만 화상도시부 프로세서의 text buffer에 대한 메모리 사이클을 수행할 수가 있는 것이다. 화상도시부 프로세서인 TMS34020은 32 bit의 local BUS를 가지고 있다. 따라서 TMS34020의 overlay buffer에 대한 access는 한번에 4 개의 pixel씩 이루어 질 수가 있다. 이렇게 text buffer를 bank로 나누어서 생각할 때 각각의 bank에 대한 control 신호들을 정리하였다.

각각의 bank 에 대한 선택신호를 *bank0, *bank1으로 할 경우 text buffer에 대한 선택신호인 *csoverlay 및 * bank0, *bank1이 모든 control 신호의 active 여부를 결정하게 된다. Text buffer를 control 하기 위해서 필요한 신호들은 다음과 같다.

- Row address strobe for bank0, bank1 (*ORAS0, *ORAS1)
- Column address stroke for all banks (*OCAS)
- Transfer/output enable for all banks (*OTR/QE)
- Write enable for all banks (*OWE)



bank 0, bank1 : 각각 8 개의 1 Mbit VRAM으로 구성됨.

0 ~ 7 bit : pixel 1
8 ~ 15 bit : pixel 2
16 ~ 23 bit : pixel 3
24 ~ 31 bit : pixel 4

그림 2.2.5 Text buffer bank division

이들 신호들은 overlay buffer의 해당되는 bank에 대한 메모리 사이클이 수행될 때 유효(active)하게 되며, 그외에 overlay buffer가 frame buffer의 용도 로 사용됨으로 인하여 screen refresh cycle 일때는 그 사이클에 맞는 동작을 해주어야 한다. 또한 DARM refresh 사이클을 수행할 수 있도록 동작하여야만 한다.

이러한 사항들을 고려하여 위의 control신호들을 만들어주면 다음과 같다.

*ORAS0 = (*csoverlay & *DRFSH & *SRF)||(*bank0 & *DRFSH)
|| *RAS

*ORAS1 = (*csoverlay & *DRFSH & *SRF)||(*bank 1 & *
DRFSH) || *RAS

*OCAS = (*csoverlay & *DRFSH & *SRF)||*CAS

*OTR/OE = *TR/QE

*OWE = *WE

여기서 *DRFSH는 DRAM refresh 사이클임을 알려주는 flag이며 *SRF는 screen refresh임을 알려주는 flag, 또한 *RAS, *CAS, *TR/QE, *WE은 TMS34020으로 부터 나오는 control 신호들이다.

(2) Frame Buffer Interface

화상도시부 프로세서인 TMS34020에 의하여 frame buffer에 대한 메모리 사이클이 수행되는 경우는 없다. 그러나 이것이 가능하다고 생각하고서 frame buffer에 대한 TMS34020의 메모리 access에 필요한 control 신호들을 정리할 필요가 생긴다. 실제로는 frame buffer에 대

한 TMS34020의 access는 screen refresh cycle 또는 DRAM refresh cycle의 경우로 국한되게 되므로 앞에서 가정한 사실은 필요없게 될 수도 있다. 그러나 화상도시부로 독립시켰을 경우의 데이타 line의 확장성을 고려한다면 위의 가정이 반드시 필요하게 된다.

Overlay buffer에서의 경우와 마찬가지로 frame buffer의 경우도 자신에 대한 메모리 사이클임을 판단할 수 있는 선택신호 및 bank 선택신호가 필요하게 되며, 이들 신호와 TMS34020으로부터 나오는 control 신호들의 조합에 의하여 frame buffer에 대한 control 신호들을 만들어 주게 된다. DRAM refresh나 screen refresh 사이클이 아닌 경우는 앞의 설명과 같은 형태에 의해서 만들어진 control신호들에 의해서 frame buffer에 대한 메모리 access가 이루어질 수 있으나 DRAM refresh나 screen refresh를 고려한다면 이들 control신호들에 대한 수정이 필요하게 된다. 즉, DRAM refresh의 경우 어떤 특정한 하나의 메모리 소자에 대한 사이클이 아니라, 모든 local메모리 (DRAM 소자를 사용하는) 소자들에 대하여 동일한 신호가 공급되어야 한다는 제한조건이 들어가게 되고, screen refresh 의 경우는 frame buffer와 text buffer에 대하여 동시에 작용하여야 하므로 이러한 사항들이 앞서 말한 control신호들의 수정에 필요한 판단근거를 제공하게 되는 것이다.

따라서, 이러한 사항들이 고려된 상태에서의 frame buffer control 신호들을 만들어 주어야만 한다. 이 신호들은 다음과 같다.

- Row address strobe 신호들

*vRAS 0 = (*csframe & * DRFSH & *SRF) // (*bank 0 & * DRFSH)//*

RAS

*vRAS1 = (*csframe & * DRFSH & *SRF) // (*bank 1 & *

DRFSH)//*RAS

- Column address strobe신호

*vCAS0 = *vCAS1 = (*csframe & *DRFSH & *SRF) // *CAS

- Transfer and Output enable신호

*vTR/QE = *TR/QE

- Write enable 신호

*vWE = *WE

(3) 프로그램 메모리 인터페이스 설계

프로그램 메모리는 화상도시부 프로세서는 TMS34020의 local 프로그램 메모리로서의 역활을 하게되며 host인 VME system으로부터 이곳으로 프로그램을 download받아 이것을 수행하게 된다. 프로그램 메모리는 256 Kwords(32 bit/word)의 용량을 가지며 앞의 두 메모리들과 마찬가지로 TMS34020의 local address line을 decoding함으로써 선택되어진다.

프로그램 메모리의 경우도 앞의 두 경우와 같이 메모리 control 신호들을 만들어주는데 제한적인 요소가 있다. 그것은 DRAM refresh 사이클의 경우로서 이 경우에도 앞의 것들과 같은 방법으로, control 신호에 대한 수정을 해주어야만 한다.

이렇게 하여 생성된 프로그램 메모리에 대한 control신호들은 다음과 같다.

-Row address strobe신호

*PRAS = (*csprogram & *DRFSH) // *RAS

-Column address strobe신호

*PCAS = (*csprogram & *DRFSH) // *CAS

- Write enable신호

*PWE = *WE

(4) RAMDAC 인터페이스 설계

Frame buffer및 text buffer의 pixel데이타들을 모니터상에 도시하기 위해서는 이들 pixel데이타들의 형태로 디지탈 신호에서 아날로그 비디오 신호로 바꾸어 주어야 한다. 이과정에서 pixel데이타 자체가 DAC되어 비디오 신호를 만들어 줄수도 있겠지만, pixel데이타를 address로 하여 lookup table을 거쳐서 DAC를 할 수 있다면 pixel데이타의 모니터 상의 도시는 여러가지의 가능성을 가질 수 있게 될 것이다. 본 연구에서 이러한 점을 고려하여 비디오 DAC와 lookup table이 같이 들어있는 소자를 이용하여 frame buffer및 overlay buffer에 들어있는 화상의 도시를 할 수 있게 하였다.

이렇게 할 경우 이소자 (RAMDAC, Bt463)의 내부에 있는 lookup table및 이들에 대한 control register들에 대한 access가 필요하게 되므로 RAMDAC도 하나의 메모리 소자로서 간주하여야 하게 된다. Bt463 RAMDAC의 경우 메모리 소자로 간주하여도 동작에 지장이 없도록 control신호들을 만들어 줄 수 있으며 이들 신호들은

address의 decoding에서 만들어 주는 신호들 뿐 아니라 비디오 DAC를 할 때의 비디오 clock등도 필요하게 된다. 따라서 RAMDAC에 대해 두가지 종류의 control신호들을 모두 정의하여야만 한다.

- Address decoding으로부터 만들어지는 신호

C0,C1 : RAMDAC의 address위치 지정

R/W : Read mode or write mode

*CE : RAMDAC enable신호

- 비디오 control신호

dotclk, *dotclk : 비디오 clock

*LD : pixel data load clock

VREF : 비디오 DAC의 기준 전압 입력

*SYNC, *BLANK : 비디오 신호의 sync, blank timing입력 신호

위에서 비디오 control신호들은 timing설계부분에서 다룰 것이며 여기서는 생략하기로 한다. RAMDAC에 대한 메모리로서의 동작은 *CE신호의 high에서 low신호로의 천이 순간에서의 C0,C1및 R/W신호들의 값이 어떤 상태이냐에 따라서 결정되게 된다. 따라서 RAMDAC에 대한 read, write동작에서 R/w신호는 stroke신호로서의 동작이 아닌 flag신호로서 사용되게 된다. 이러한 사항들을 고려하여 만든 control신호들은 다음과 같다.

C0,C1 : 2MSB5 of colum address

R/W = *CSRAMDAC ‖ *WE

*CE = *CSRAMDAC ‖ *CAS

다. 화상도시부 timing설계(비디오 timing인터페이스 설계)

화상도시부 프로세서인 TMS34020은 그래픽용 프로세서로서 비디오 timing에 관계되는 여러개의 신호핀들을 가지고 있다. 이들은 아날로그 비디오 신호의 sync및 blank등에 관계되는 신호들로서 이들에 의해 비디오 timing에 대한 인터페이스가 원활히 이루어 질수 있게 된다.

이들 pin들은 TMS34020내부의 control register들의 내용을 프로그램 해줌으로써 여러가지의 동작 모드를 가질수 있으며 입력, 또는 출력으로 사용될 수 있다.

화상도시부에서 비디오 신호(아날로그)의 생성에 관계되는 부분들은 화상도시부 프로세서인 frame buffer와 overlay buffer, 그리고 RAMDAC 및 이들을 인터페이스 시켜주는 부분들이다. 이들 사이의 관계는 그림 2.2.6 에 나타나 있다.

그림 2.2.6에서 볼 수 있는 바와 같이 TMS34020으로부터 나오는 비디오 timing신호들은 video 인터페이스부에서 몇가지의 다른 신호들과 결합되어 frame buffer 및 text buffer 에 대한 pixel shift clock및 output enable신호, 또한 RAMDAC에 대한 timing신호들로 만들어지게 된다.

이들 신호들에 대하여 좀 더 말하자면, pixel shift clock은 VRAM의 특성상 DRAM patr에서 shift register로 옮겨진 (screen refresh)pixel data들을 한 pixel씩 shift out 시켜주기 위하여 필요하며, output enable신호는 bank 0, bank 1들중의 하나의 bank에서만 pixel data를 출력시켜주기 위하여 필요하게 된다. 즉 output enable이 active 인

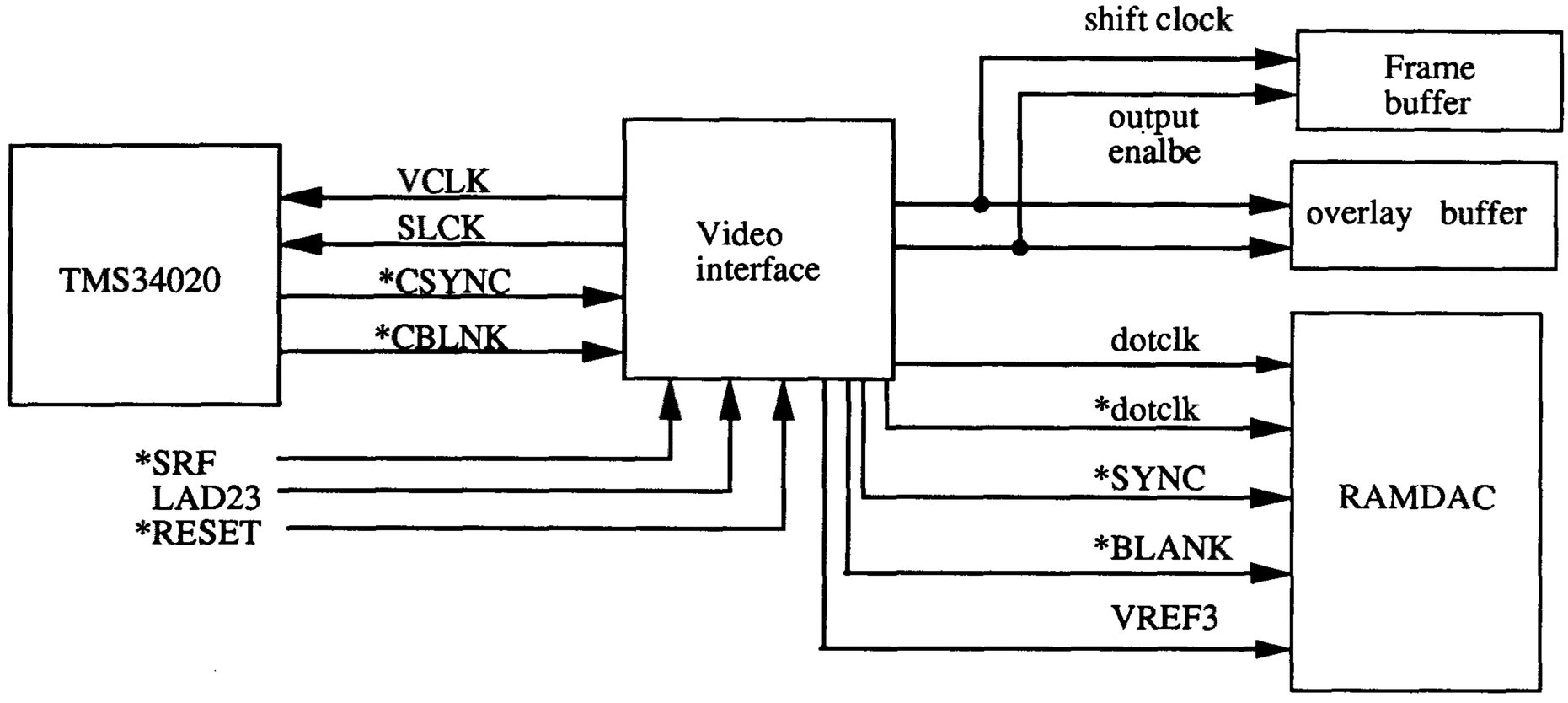


그림 2.2.6 Block diagram of video timing circuit

bank에 pixel shift clock 이 들어가면 VRAM내부의 shift register로부터 pixel data들이 하나씩 나와 이것이 RAMDAC을 거쳐 비디오 모니터상에 도시되는 것이다. RAMDAC에 대한 timing 신호들은 dotclk, *dotclk, *LD, *SYNC, *BLANK등이 있다. 우선 dotclk와 *dotclk은 1280 x 1024크기의 모니터에 50Hz 순차주사방식의 도시를 위하여 106 MHzMH주파수를 갖는 clock 발진기를 통하여 만들어주게 된다. 그리고 *LD신호는 DAC하기 위한 pixel data를 latch 하기 위한 clock으로서, VRAM의 shift register의 출력 속도가 106MHz를 따라가지 못하기 때문에 4개의 pixel을 한꺼번에 latch하여 이를 multiplexing함으로써 106 MHz속도의 도시를 할 수 있게 된다. 따라서 *LD clock은 dotclk을 4분주하여 만들어 주게 된다. 그리고 *SYNC신호와 *BLANK신호는 *LD clock의 Low에서 High로의 천이 순간에 RAMDAC로 읽혀들어가 DAC된 아날로그 비디오 신호의 SYNC와 BLANK로서 작용하게 된다. SYNC와 BLANK의 timing이 흐트러질 경우 모니터상에 도시되는 화상이 흐르거나, 깜박거릴수 있으므로 *SYNC, *BLANK신호는 *LD신호에 동기시켜서 사용하는 것이 바람직한 사용법이 된다.

라. 도시버스와의 인터페이스 설계

화상도시부는 기본적으로, 신호처리부에서 처리된 화상을 모니터에 도시하기 위한 부분들로 구성된다. 신호처리부에서 처리된 화상들은 신호처리부의 D-Bus를 통하여 화상도시부의 frame buffer로 옮겨지게 된다. 도시버스(D-Bus)로 부터 나오는 신호들은 그림 2.2.7과 같다.

이들은 각각 frame buffer에 대한 address와 화상데이타(64 bit), 또

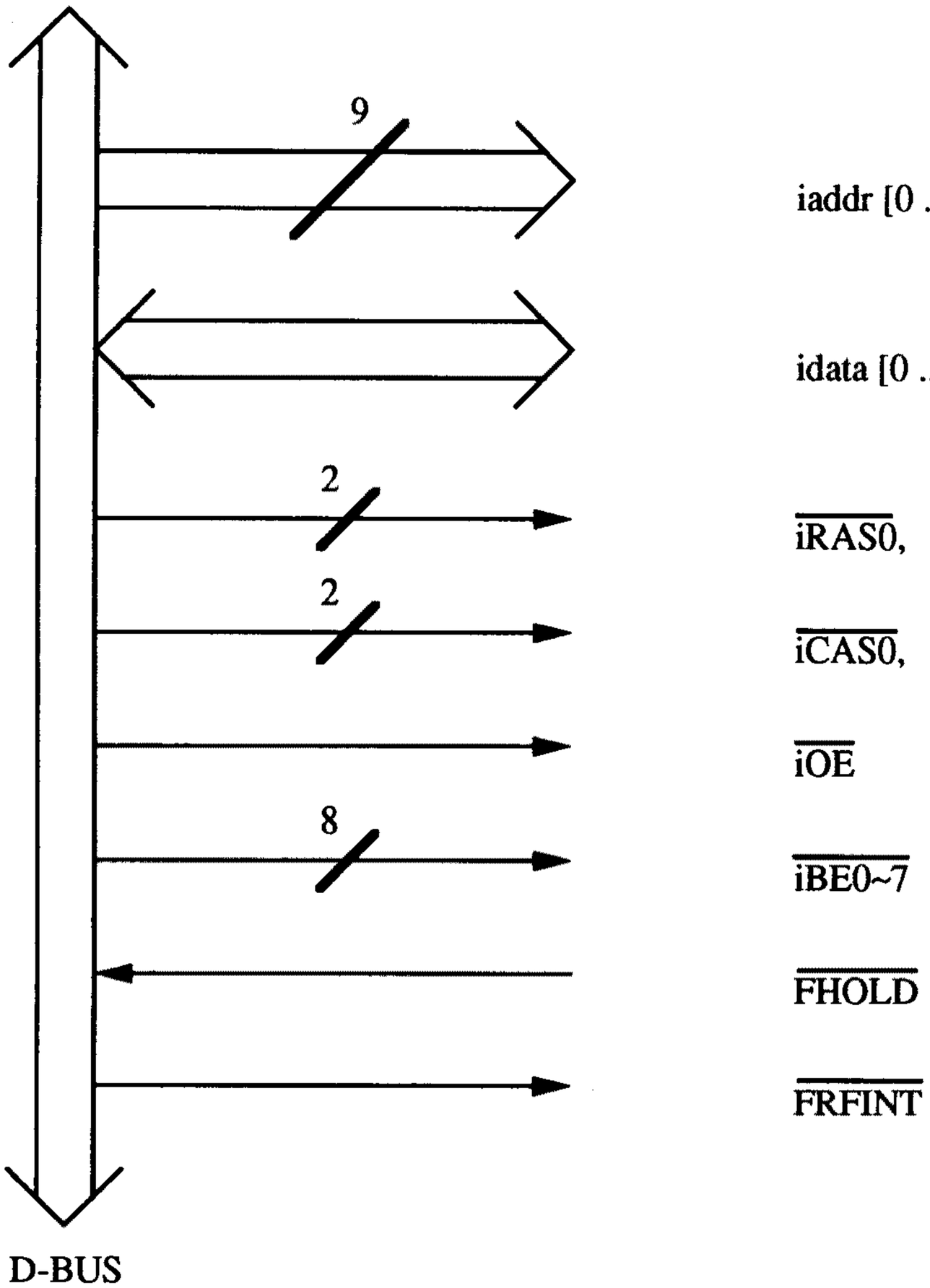


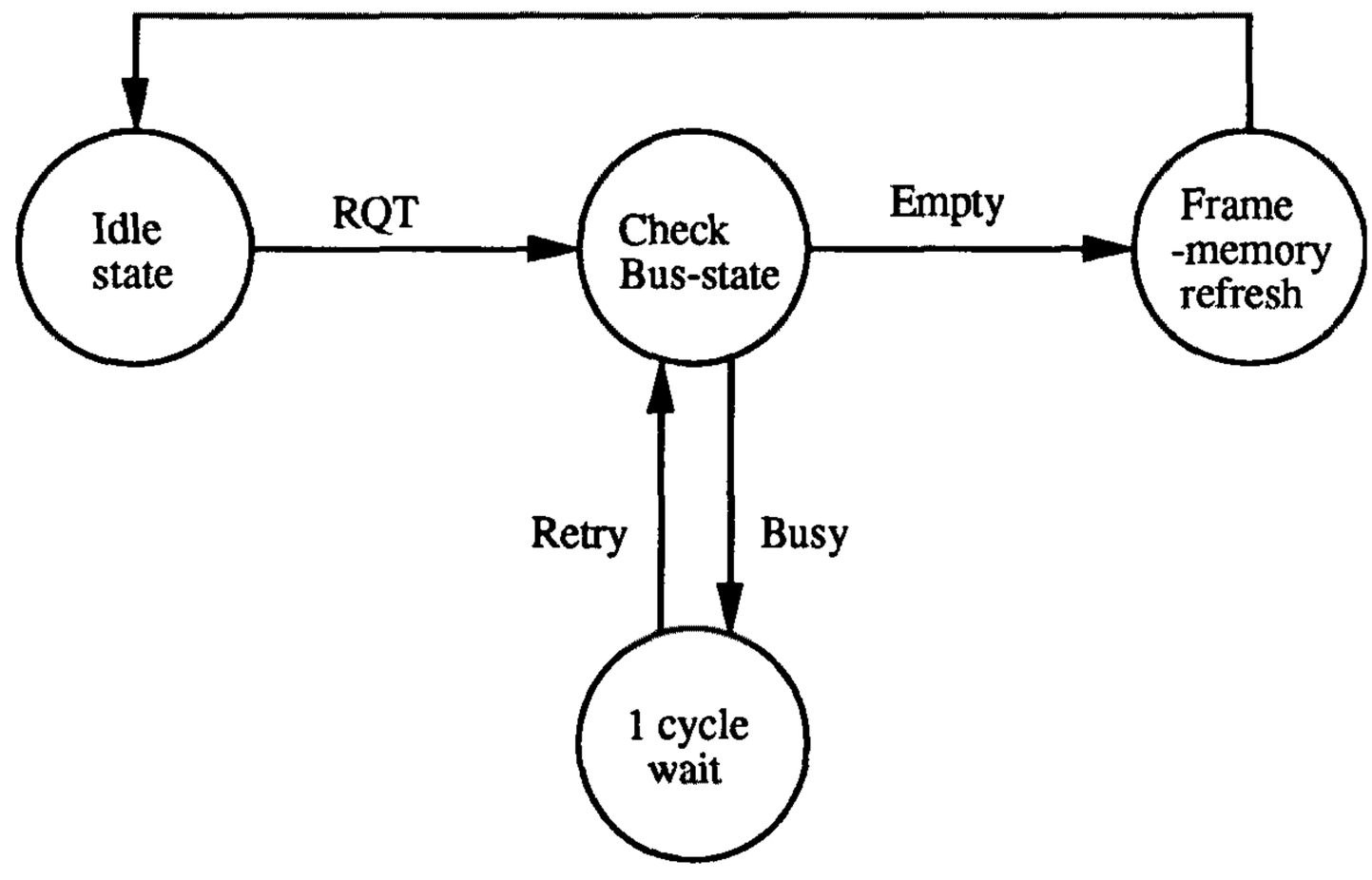
그림 2.2.7 도시버스에서 나오는 신호들

한 frame buffer의 bank를 선택하기 위한 address strobe신호 및 output enable과 write enable신호들이다. 또한 여기에 첨가하여 화상도시부 프로세서와 신호처리부간의 hand shaking신호들이 포함되게 된다.

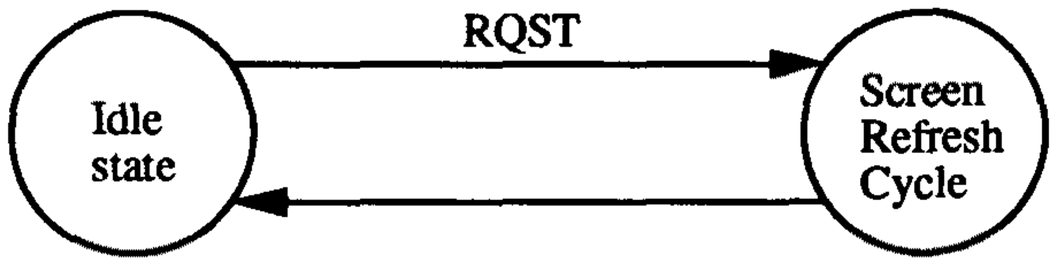
화상도시부 프로세서 (TMS34020)와 신호처리부간의 hand-shaking 신호의 교환은 상호간의 메모리 사이클이 파괴되는 것을 방지한다는 의미에서 중요하다. 즉 frame buffer에 대한 화상도시부 프로세서의 access는 주기적인 화상의 도시 (screen refresh) 및 DRAM refresh 사이클 이외에는 없다. 그러나 screen refresh와 DRAM refresh의 경우 시간적인 주기성이 매우 중요한 요소로 작용하게 되므로 이들 메모리 사이클이 시작되기 직전에는 신호처리부 프로세서의 frame buffer에 대한 데이타 전송의 한 사이클이 끝나야만 한다. 따라서 화상도시부로 부터의 request (~FRFINT)에 의하여 신호처리부에서는 현재 수행중인 메모리 사이클의 수행을 끝을 내고 그 결과를 화상도시부에 알려주어야 한다 (~FHOLD).

그림 2.2.8 에는 화상도시부와 신호처리부간의 hand shaking을 나타낸 것으로 DRAM Refresh의 경우는 control 신호의 조정에 의해 refresh cycle이 pending될수 있으므로 wait_state를 포함한 cycle이 수행되며, screen refresh의 경우는 wait_state를 포함하지 않으므로 실제 screen refresh time보다 앞당겨 refresh를 request하여야 한다.

앞에서도 잠깐 언급하였듯이, 화상도시부 프로세서의 신호처리부에 대한 access는 screen refresh와 DRAM refresh의 경우 이외에는 존재하지 않는다. 따라서 frame buffer에 대한 address및 데이타 line의 연결 및 기타 control신호들의 연결도 이러한 사실을 고려하여야 한다. 즉, frame buffer 에 일상적으로 연결되어야 할 신호선들은 도시버스로 부터 나오는 신호들이 되어야 하고, Screen refresh와 DRAM



(a) DRAM Refresh Cycle



(b) Screen Rfresh Cycle

그림 2.2.8 DSP 와 GSP 사이의 Handshaking

refresh주기에는 화상도시부 프로세서로 부터 나오는 신호들이 frame buffer에 연결되어야만 한다. 또한 screen refresh와 DRAM refresh의 경우 데이타 line의 신호는 그 메모리 사이클과는 관계가 없으므로 도시버스로 부터 나오는 데이타 line의 신호를 그대로 frame buffer의 데이타 line으로 묶어주어도 관계가 없다.

제3절 병렬처리 시스템 구성

두 개의 DSP(i860)와 GSP(TMS 34020)사이에 데이타를 전송하기 위해서 두 개의 DRAM 메모리 모듈을 억세스하기 위한 인터페이스부를 설계하여 병렬처리용 컴퓨터 시스템을 구성하였다.

DSP에서 처리한 데이타는 GSP와 Frame 메모리 모듈을 공유하므로써 서로 교환하는 방식으로 타이틀하게 카플(Tightly Coupled)되어 있고 DSP들 끼리의 정보교환은 Frame 메모리를 하나의 Mail-box로 보고 Daisy chain방식으로 데이타를 주고 받을 수 있도록 루우즈하게 카플(Loosely Coupled)되어 있는 하이브리드(Hybrid) 형태의 멀티프로세서(Multiprocessor) 시스템을 구성하였다.

의료용의 영상 데이타는 여러개의 이미지 프레임(Image frame)으로 구성되어 있으므로 두개의 DSP 프로세서를 사용하며 Concurrent한 프로세싱이 가능하므로 단일 프로세서를 사용할때 보다 두 배에 가까운 프로세싱 파워(Processing Power)를 가질 것이다. 두 개의 DRAM 모듈은 각각의 DSP 프로세서에게 로칼(Local)하게 메모리가 공급되므로 한쪽의 DSP에 있는 DRAM 모듈에 있는 DSP가 이미 프로세싱한 영상 데이타를 CRT 모니터에 도시하는 동안 다른 쪽의 DSP 프로세서는 또 다른 영상 데이타를 프로세싱할 수 있으므로 거의 연속적으로 영상 화면을 도시할 수 있게 된다.

쉽게 구할 수 있는 저가의 일반적인 DSP, GSP 프로세서와 로직칩들을 사용하므로써 코스트 이펙티브(Cost Effective)한 하이브리드 형태의 멀티프로세서 시스템을 구성하였으며 의료용 영상 데이타를 처리하는데 연속적으로 2 장의 영상을 동시에 처리할 수 있을 뿐 아니라 GSP 프로세서는 연속하여 두개의 DSP에서 처리한 영상을

CRT 모니터에 도시 할 수 있다는 장점을 지닌다. 전체적인 병렬처리 컴퓨터 아키텍처는 다음 그림 2.3.1과 같다.

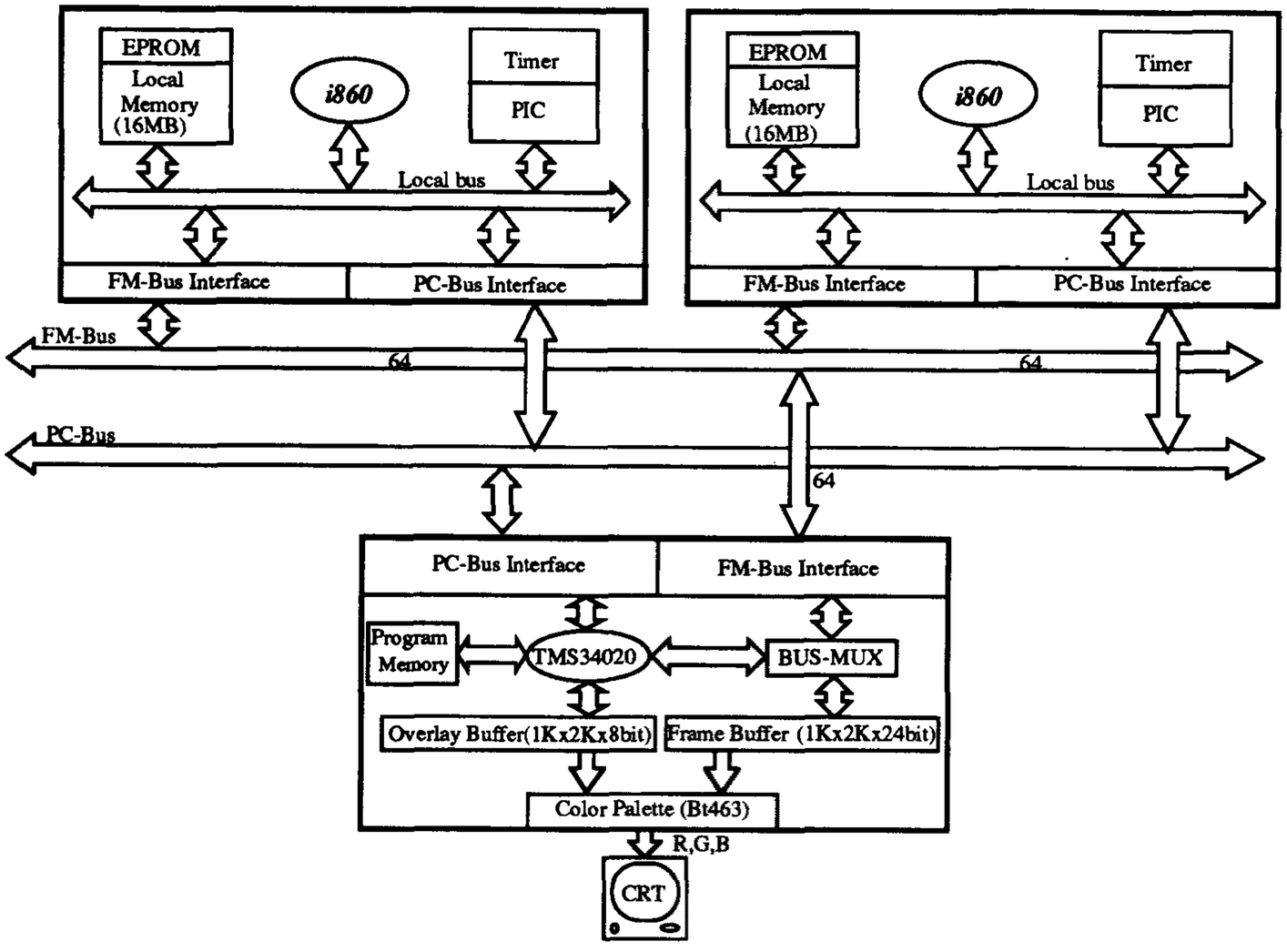


그림2.3.1 병렬처리 컴퓨터 시스템 블럭다이아그램

제3장 영상처리 시스템 소프트웨어 구조

본 시스템의 소프트웨어의 구성은 크게 두부분으로 되어있다. 이것은 호스트인 IBM-PC와 영상처리 보드에서의 구성인데, IBM-PC에서 실행되는 프로그램들은 크로스 컴파일러와 IBM-PC의 시스템 콜을 이용하여 프로그램된 디바이스 드라이버(Device Driver) 루틴들(DSPs, GSP), 그리고 각종 명령이나 명령의 전처리 작업을 해주는 유틸리티로 구성되어 있으며, i860에서 실행되는 것은 모니터 프로그램으로서 영상처리 보드에서의 프로그램 실행과 호스트 컴퓨터가 요구하는 인터럽트에 대한 서비스를 담당한다. 전체적인 구성도는 그림3.0.1에 나타나 있으며, 다음절에서 자세히 살펴 보겠다.

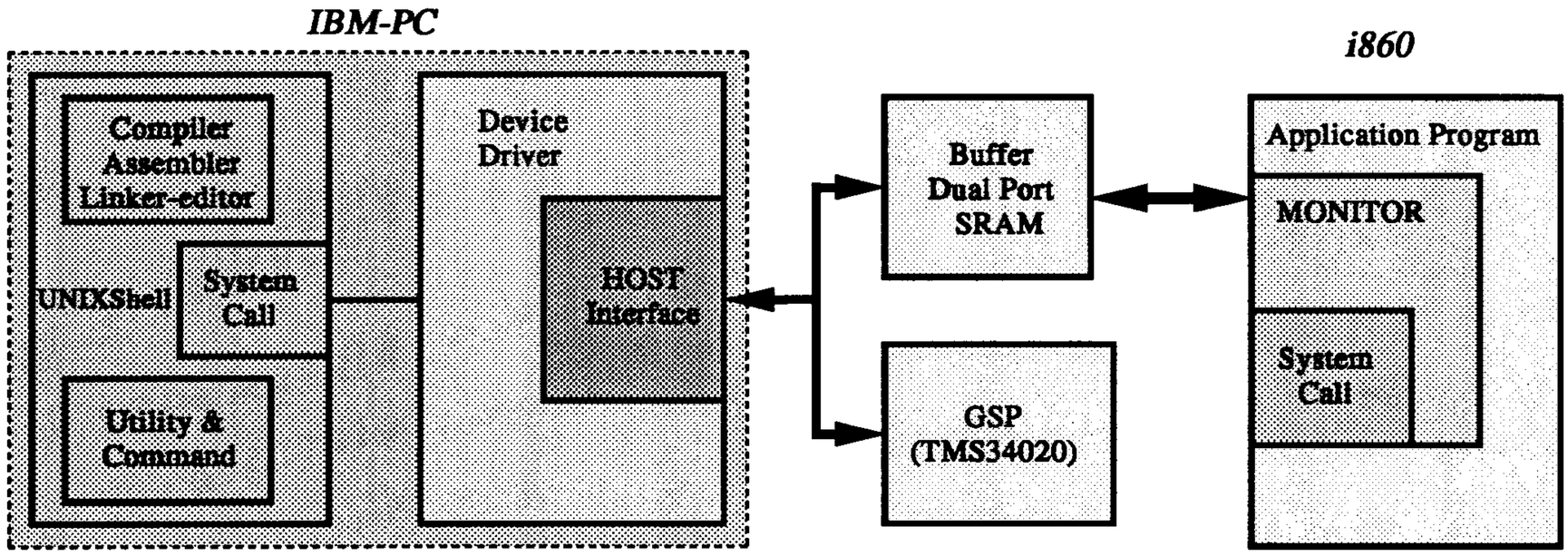


그림3.0.1 전체적인 S/W 구성도

제1절 IBM-PC Host 에서의 소프트웨어 구조

1. 디바이스 드라이버

디바이스 드라이버는 호스트와 영상처리 보드간의 데이타 전송을 담당하고 여러 명령을 영상처리 보드에 보내어, 이를 실행하게 된다. 영상처리 보드의 호스트 인터페이스는 IBM-PC 메모리 공간내에 dual-port SRAM을 버퍼로 사용하며, 동시에 i860 CPU가 access할 수 있도록 하는 shared memory 방식을 사용한다. 따라서 IBM-PC는 이 메모리를 직접 억세스할 수 있으며, i860도 마찬가지이다. 디바이스 드라이버에서 데이터 전송 프로토콜은 기본적으로 인터럽트를 주고 받는 형태로 구성되며 디바이스 드라이버의 동작에 대해 살펴보면 다음과 같다. 사용자 프로세스(응용 프로그램)가 디바이스를 볼 때 정규 파일처럼 열거나 닫고, 쓰고 읽을 수 있는 특수 파일로 본다는 것이다. 사용자 프로세스는 시스템 호출을 이용하여 보통 파일처럼 디바이스를 운용할 수 있다.

특수 파일이 존재하는지를 알고 있는 UNIX 커널에 시스템 호출을 하면, 커널이 실제 파일과 디바이스 특수 파일을 구분하여 해당되는 디바이스 드라이버를 억세스 한다. 이 드라이버가 디바이스의 모든 동작을 제어하며, 디바이스는 드라이버가 요구하기 전에는 아무런 동작도 하지 않는다.

시스템 호출이 사용자 프로세스와 UNIX간의 인터페이스를 담당하고, 디바이스 드라이버는 커널과 주변 디바이스와의 인터페이스를 담당한다. 그러므로 디바이스 드라이버는 UNIX의 디바이스 독립성을 이루는데 결정적으로 중대한 역할을 하는 것이다. 디바이스 드라

이버만이 시스템의 주변 디바이스를 억세스할 수 있다. 그림 3.1.1은 디바이스 드라이버의 역할에 대하여 보여 주고 있다. 사용자 프로세스가 시스템 호출을 하면 커널로 수행이 옮겨진다. 이때, I/O에 관련된 시스템 호출인 경우에는 커널이 정규 파일(디스크와 같은 블럭 디바이스에 있는 파일들)과 다른 I/O 디바이스(단말기나 프린터등)에 관련된 특수 파일을 구분한다. 이러한 디바이스들은 /dev 디렉토리를 있는 특수 파일들로 보여진다. 특수 파일은 보통 방법으로는 생성되지 않는다. 특수 파일이 가지고 있는 inode 정보(파일 상태를 나타내는 시스템 구조)는 정규 파일에 있는 것과는 달라서, 특수 파일은 mknod 라는 시스템 관리 명령에 의해서만 생성될 수 있다. 정규 파일이 갖고 있는 정보는 디스크 상에 존재하고 있는 주소를 나타내는데 반해, 특수 파일(디바이스)의 inode는 디바이스 드라이버(주 디바이스 번호(Major Device Number)), 디바이스 분류(블럭 또는 캐릭터), 부 디바이스 번호(Minor Device Number)등을 결정하는 정보를 가지고 있다.

어떤 타입의 파일이 억세스 되든지, 커널은 그 호출에 응답해야 하는 디바이스 드라이버를 결정해야 한다. 이 결정을 하려면, 그 파일에 관련된 디바이스의 이름을 알고 있어야 하는데, 그것을 node의 주/부 디바이스 번호로부터 알아낼 수 있다. 디바이스 이름과 주 디바이스 번호와의 연관은 /dev 디렉토리에 있는 디바이스 엔트리에 의해 이루어진다. 주 디바이스 번호는 두가지의 디바이스 스위치중에서 하나를 결정하는데에 쓰인다. 이들 스위치는 bdevsw(블럭 디바이스 스위치)와 cdevsw(캐릭터 다바이스 스위치)인데 실제는 스트럭쳐(structure)의 배열로 되어있다. 주 디바이스 번호가 이들 스트럭쳐 중 하나를 가리킴으로써 드라이버를 선택하게 된다. 부 디바이스 번

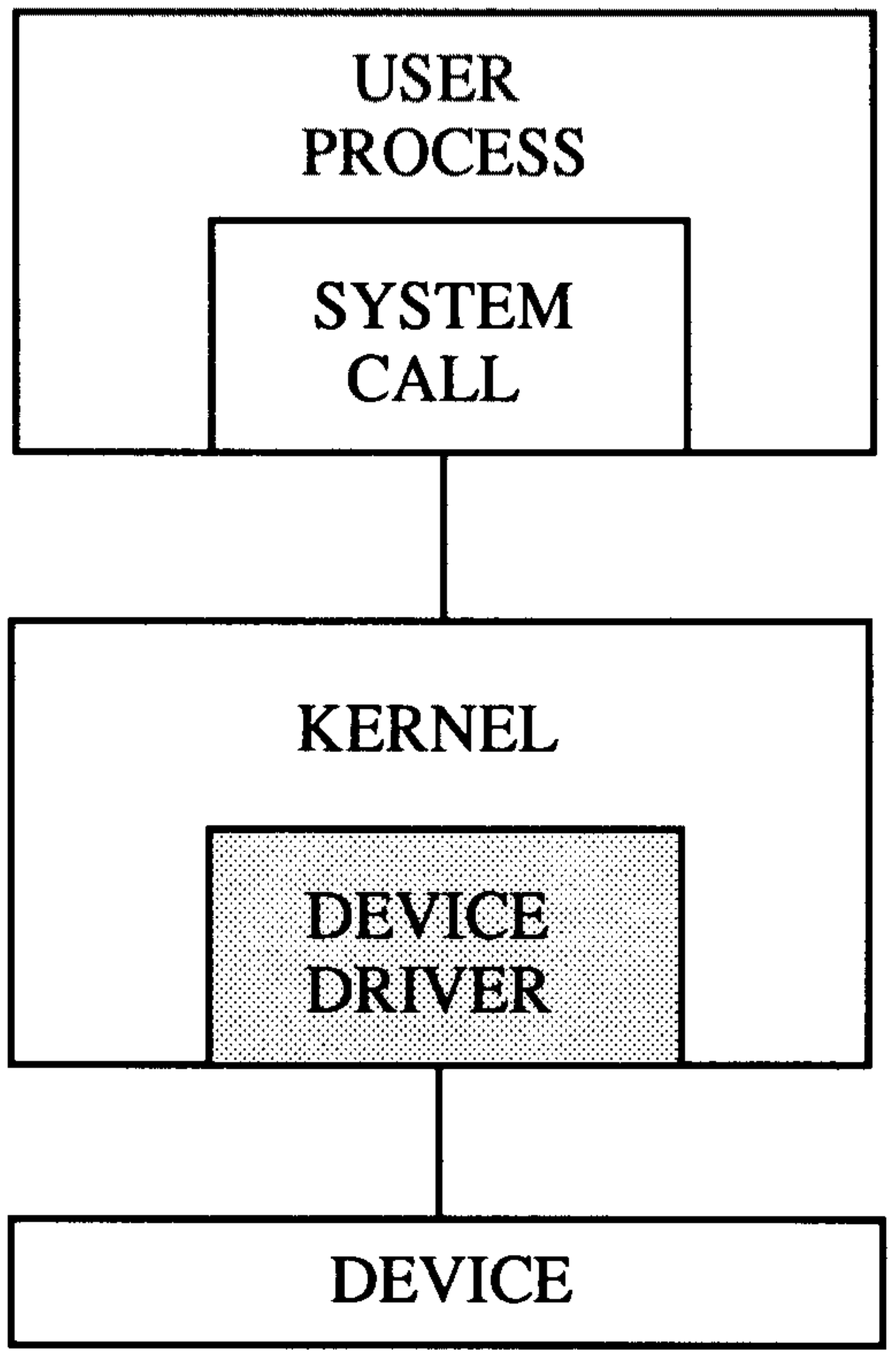


그림 3.1.1 디바이스 드라이버의 역활

호는 선택된 드라이버 내에서 국소적인 지시에 쓰인다. 새로운 디바이스 드라이버를 추가할 때에는 두 디바이스 스위치중 하나에 필요한 엔트리를 추가해야 한다.

응용 프로그램이 파일 인터페이스에 관련된 동작을 하기 위해서 OS에 요청을 한다. 그러면 OS는 그 요청을 특정 파일에 연결된 디바이스 드라이버에 옮긴다. 특정 파일 동작과 그에 해당하는 다바이스 드라이버 엔트리 간의 인터페이스는 bdevsw와 cdevsw를 통해 이루어 진다. bdevsw나 cdevsw의 각 엔트리는 드라이버의 엔트리 함수를 가리키는 포인터를 갖고 있다. 스트럭쳐 내의 엔트리 위치는 디바이스에 할당된 주 디바이스 번호에 해당한다. 부 디바이스 번호는 디바이스 드라이버에 아규먼트로 전해진다. 일반적으로, 부 디바이스 번호는 여러개의 동일한 실제 디바이스 중에서 하나를 억세스하거나, 여러 개의 부번호가 서로 다른 모드에서 동작하는 같은 디바이스를 가리키도록 하는데에 쓰인다.

2. IBM-PC와 i860 사이의 데이타 교환(Downloader)

Downloader의 동작은 기본적으로 interrupt를 base로 하여 각각의 processor들이 semaphore와 endflag를 check하면서 이루어 지는데, 그 전체적인 블럭다이아그램은 그림3.1.2.1에 나타난 것과 같다.

Downloading은 먼저 PC에서 interrupt를 i860에 가하여 수행하게 된다. Interrupt를 주기전에 먼저 semaphore와 endflag를 set해 주고, 전송할 데이타를 버퍼에 쓴다. 그리고 semaphore를 i860에게 넘겨 준다. 이렇게 하면 i860은 PC의 interrupt를 판별하고, PC-BUS interrupt이면 어떤 command인지를 다시 check하여, downloading 명령이면 먼

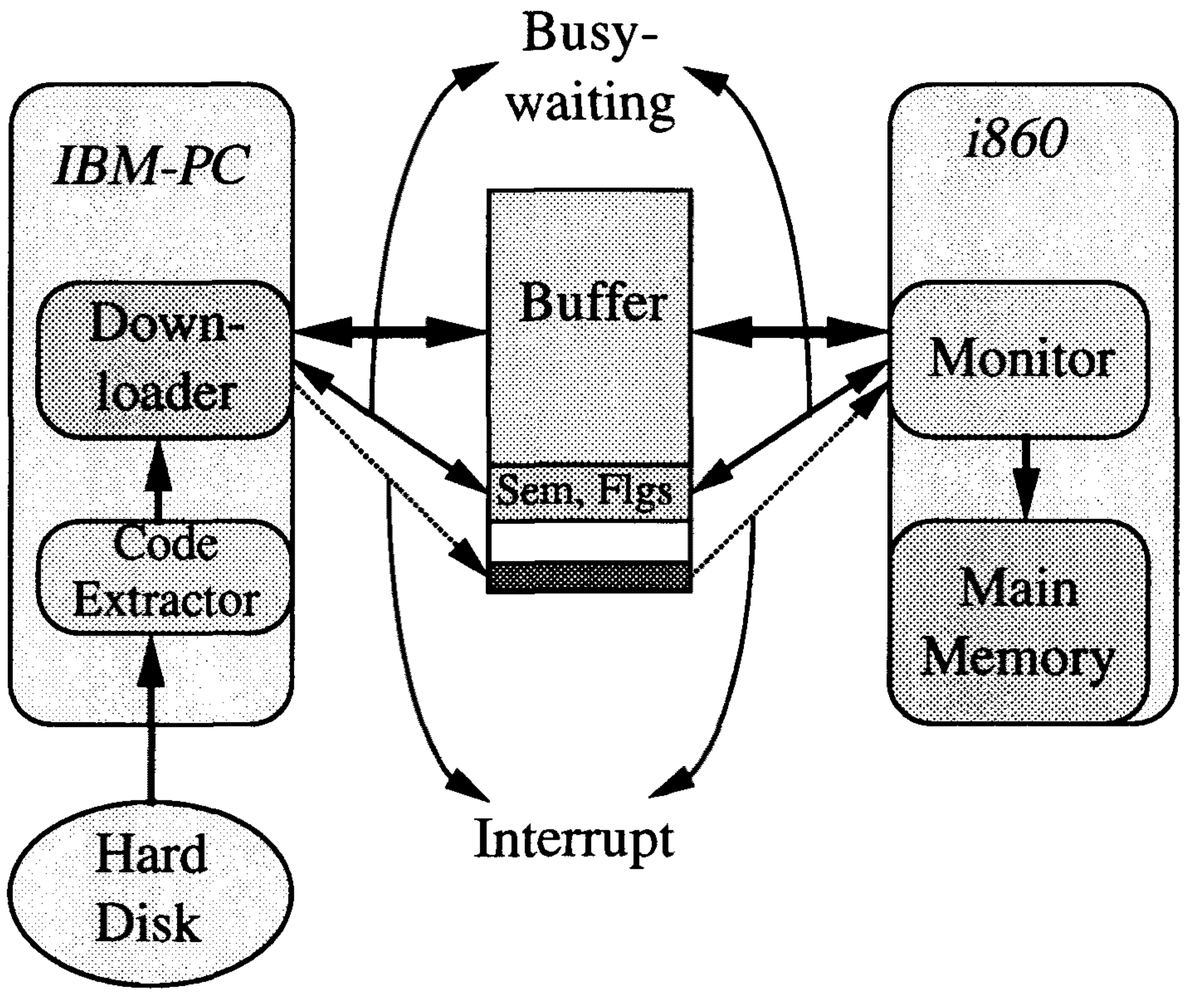


그림3.1.2.1 IBMPC - i860 Communication

저 semaphore를 check하여 i860에게 semadhore가 있으면, buffer로부터 data를 가져와 지정된 downloading address에 저장한다. 그리고 semaphore를 PC에게 되돌린다. 그 후 endflag를 check하여 communication 종료여부를 판단하는데, 종료가 아니면, 다시 semaphore를 check하면서 busy-waiting하게 된다. 이때 busy-waiting 상태에 있던 PC가 semaphore를 받아 다시 데이타를 buffer에 쓰고 처음과 같은 동작을 반복한다. 그런데 이때 한가지 다른 점은, 이제는 interrupt를 가하지 않는다는 점이다. Interrupt는 communication을 유발하기 위해서 한번만 가해 주면 그만이다. 데이타를 모두 전송하고, 더이상 전송할 데이타가 없으면, PC는 endflag를 set하여 communication 종료를 i860에게 알린다. 그러면 i860은 마지막 블럭을 가져간 후 endflag를 check하여 communication을 끝내고 PC에게 endflag를 reply해 준다. 그림3.1.2.2와 그림3.1.2.3에 PC측과 i860측의 downloading routine의 flowchart를 나타냈다.

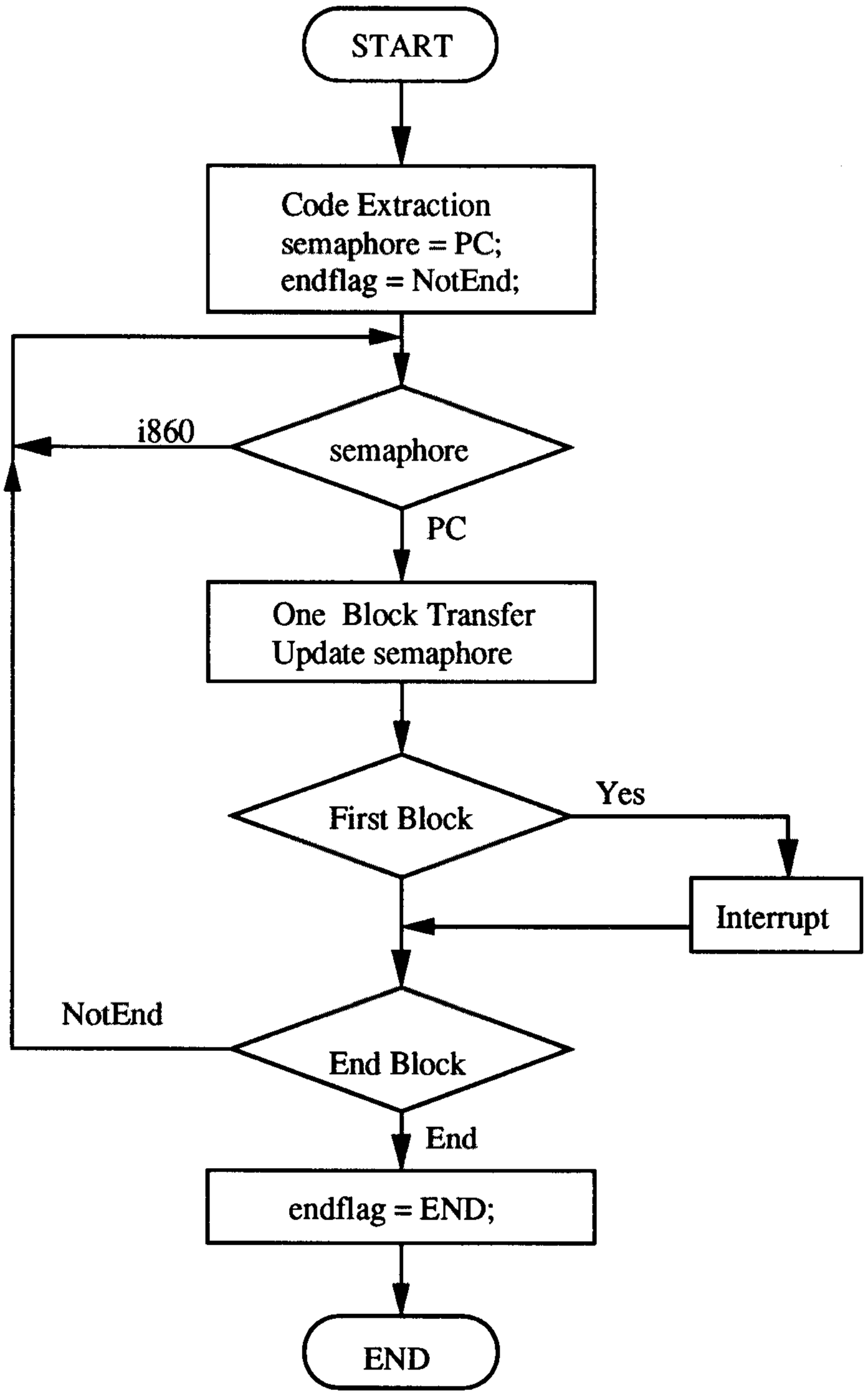


그림 3.1.2.2 PC Downloading Flow Chart

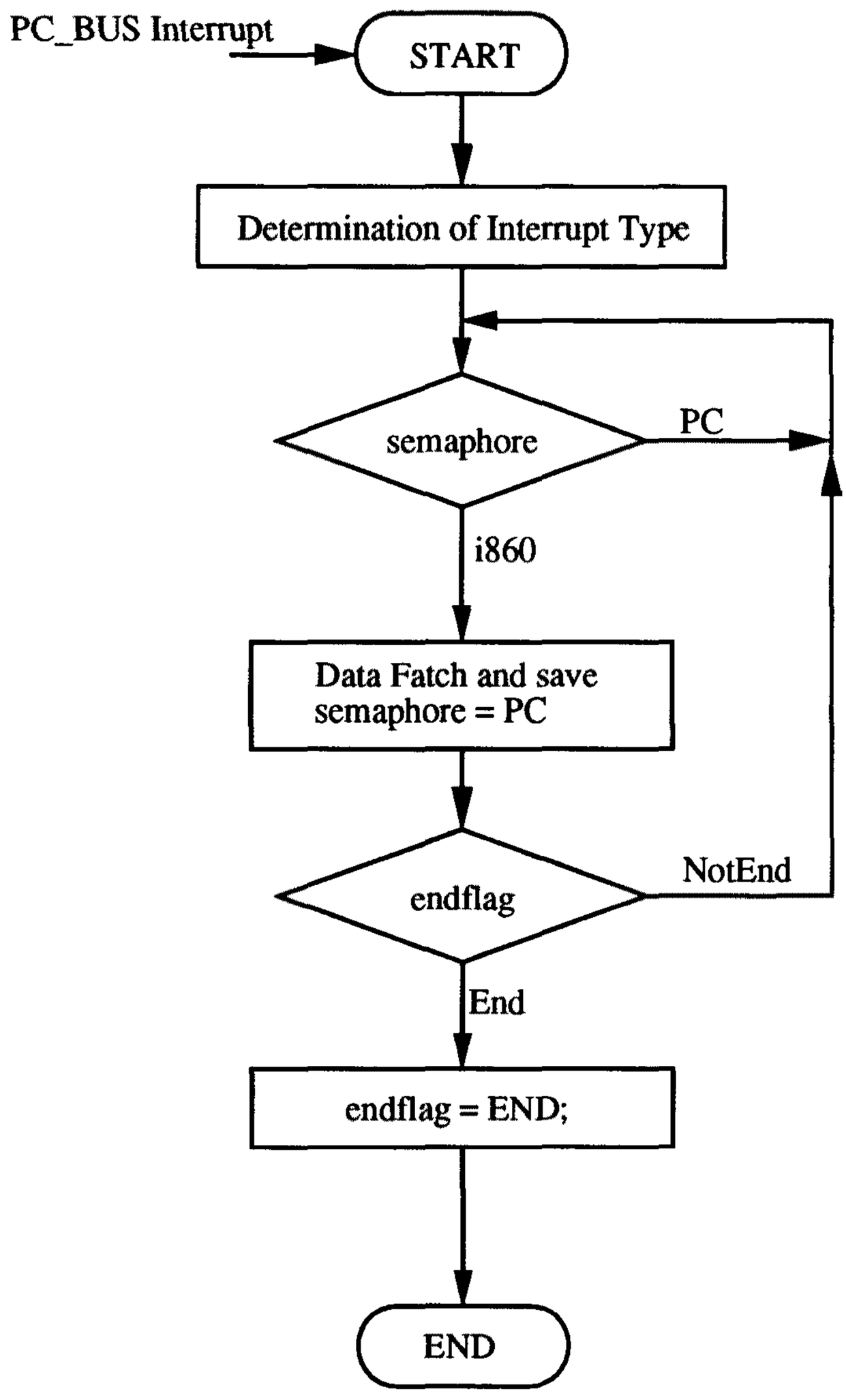


그림 3.1.2.3 i860 Downloading Flow Chart

제2절 i860 영상처리 보드에서의 소프트웨어 구조

i860 software는 주로 ROM program을 말하는 것으로, system initialization을 담당하는 것과 초기화 후에 system을 관리하는 monitor로 이루어져 있다. 이들은 4개의 어셈블리 프로그램으로 구성되어 있다. 그 구성은 그림3.2.0에 나타나 있다.

1. Reset routine

i860이 하드웨어 reset되었을때, 프로그램 카운터는 0xFFFFFF00번지로 가게되며, 이때는 ROM access mode(CS8 mode)로 동작하게 된다. 그러므로 시스템이 동작하는 첫번지가 0xFFFFFF00이며, 여기에 reset routine이 오게된다. 이것의 동작은 단순히 copyrom routine으로 분기하는 것 뿐이다.

2. Copyrom Routine

이 루틴은 시스템의 ROM mode에서 실질적으로 실행되는 루틴이다. 주요 동작은 system initialization과 monitor code를 RAM에 복사하여, 실행할 수 있게 해 준다. 이 두 동작을 살펴보자.

가. System Initialization

Reset 후에 시스템의 상태는 invalid하다. 따라서 적절한 초기화를 통하여 정상적으로 동작하게 해 주어야 한다. 다음과 같은 초기화 작업을 해준다.

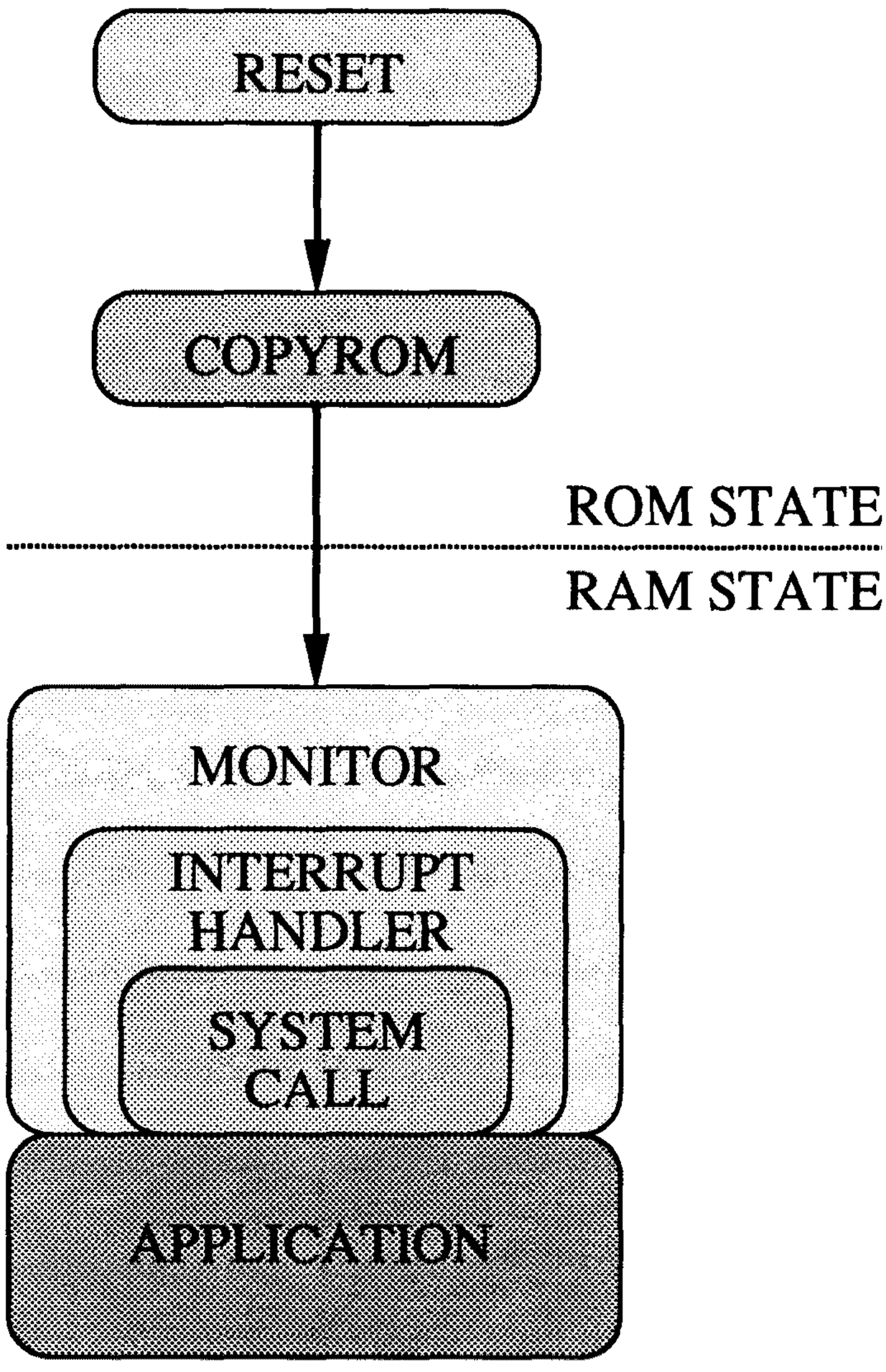


그림3.2.0 DSP Module S/W 구조

- Register initialization

- Pipeline initialization

- Cache initialization

- Timer initialization

- PIC initialization

- Token set

- Memory test and initialization

- Parity bit clear

Register initialization은 control register를 set하고 fir(Fault Instruction Register)를 clear해 준다. Pipeline의 초기화는 register f0를 사용한 pipeline instruction을 사용하여 각 plpeline의 상태를 clear할 수 있다. Timer initialization은 두가지 목적을 갖는다. 하나는 RAM refresh 신호를 만들어 주기 위해서 timer 내부에 있는 counter를 16 microsec cycle을 갖도록 해주고, 다른 하나는 multi-tasking용 clock tick을 만들어 주는데 쓰인다. 두번째의 경우는 지금 필요하지 않으므로 PIC에서 masking하여 reserve해 둔다. 이 때 timer를 set하는 방법은 다음과 같다. 먼저 timer를 선택하기 위한 decoding 신호를 만들기 위해서 지정된 memory map을 선택하여, timer의 control word를 set한다. 이때 control word에는 다음에 set될 counter를 지정하고, 그 mode를 지정한 정보가 들어 있다. 그리고 나서 counter에 지정된 값을 쓴다. PIC 초기화는 외부에서 들어오는 interrupt의 우선순위를 결정하거나, 이들을 선택적으로 masking하여 받아들이기 위함이다. 이것은 timer의 경우와 비슷하게 initialization command word들과

operation command word들의 mode를 선택하게 해준다. Token 은 parallel processing에 대비하여 해주는 작업인데, 하나의 board에만 해준다. 이것은 daisy-chain을 구성하여 처음의 bus사용권을 하나의 processor에게만 주고, 요구에 따라 이것을 다른 프로세서들이 취할 수 있게 해준다.

Memory initialization은 Frame memory와 SRAM을 clear 시키고, local memory를 '1', '0'에 대해서 test를 먼저 한 후에 clear시킨다. 그리고 나서 parity bit을 clear시켜 parity error를 check할 수 있게 준비한다.

나. ROM 코드의 RAM으로 복사.

ROM에 들어 있는 코드는 직접 실행 가능한 code(reset, copyrom)가 있고, RAM으로 복사되기 위한 특별한 format을 가진 code(monitor)로 구성되어 있다. RAM으로 복사할때 double copy load method를 사용하는데, i860의 구조상 CS8 mode(ROM mode)에서 ROM code를 직접 read할 수 없기 때문에 ROM의 address를 다른 address space로 변환시켜 access하게 된다. 이 때 복사를 위한 특별한 format이란, ROM을 access할때 8 bits로 access하는데, 이것은 바이트 enable 신호(BE#0~BE#2)를 조합하여 가능하지만, BE#2~BE#0의 값이 '001', '010' 일때는 address가 invalid하기 때문에 이 address에는 garbage code를 넣어 원래의 코드가 잘못 복사되는 것을 막아주는 것을 말한다. 이렇게 하여 copy가 끝나면 copyrom 루틴은 어드레스를 remap하여 정상적인 동작 mode를 RAM address space에 오게 한다. 이렇게 한 후에 모니터 프로그램의 wait 루틴으로 분기한다.

3. Monitor

Monitor는 대기(wait)루틴, 인터럽트 핸들러(interrupt handler), 그리고 실행시의 library를 지원하기 위한 system call로 구성되어 있다. Wait routine은 단순히 외부의 interrupt를 기다리는 routine이다. Interrupt handler는 외부 인터럽트나 trap이 발생했을때, 이를 처리해 주며, system call은 dynamic memory allocation을 위해서 필요하다. 각각에 대해서 자세히 알아 보겠다.

가. Interrupt Handler

우선 인터럽트가 들어 왔을때 처리 순서를 그림3.2.3.1에 나타냈다. 그림에서 보는 바와 같이 trap의 시작은 psr(processor state register)의 IT(Instruction Trap), IN(external Interrupt), IAT(Instruction Access Trap), DAT(Data Access Trap), FT(Floating-point Trap) bit들이 set되면 일어난다. 이런 경우가 발생하면, handler는 바로 환경(context)을 stack에 저장하여 trap service후에 원래 상태를 복구할 수 있게 한다. 그리고 psr을 검사하여 trap의 type을 결정하여, trap(여기서 trap은 외부 인터럽트를 제외한 것) 서비스 루틴으로 가거나, 외부의 인터럽트 서비스 루틴으로 간다. 인터럽트 서비스 루틴은 다운로딩 루틴과 프로그램을 실행시키는 루틴으로 구성된다. 다운로딩 루틴은 앞 절에서 설명한 것과 같으며, 프로그램 실행 루틴은 실행될 프로그램을 위한 stack과 환경을 설정해 주고 실행 어드레스 entry point로 프로그램 카운터를 옮긴다. 그리고 실행이 끝나면 다시 원래의 상태를 회복하게 해준다.

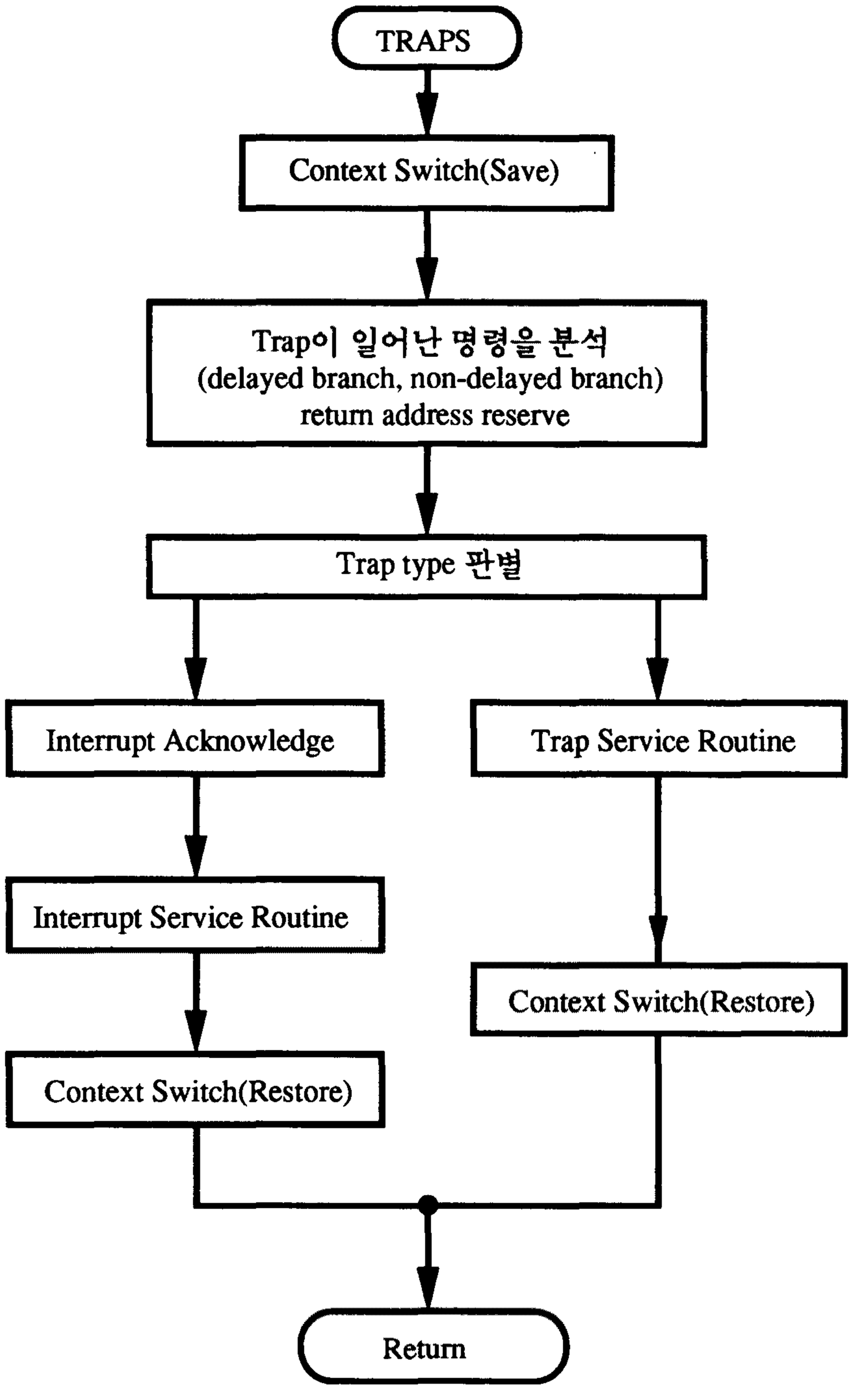


그림3.2.3.1 인터럽트 처리

나. System call

System call 이란 사용자가 직접 프로그램상에서 다룰 수 없는 문제나 실행 도중에 처리되어야 하는 경우에 프로그램상에서 불리어져 응용 프로그램을 더욱 유용하게 해 준다. 본 시스템의 monitor에서 지원하는 system call은 실행시에 dynamic memory allocation을 해주기 위한 것으로서 brk(), sbrk()를 지원한다. 여기서 brk(addr)는 data section의 break value를 addr에 assign해주는 system call이고, sbrk(size)는 size만큼의 메모리를 할당하고, 이것을 초기화 한다. 그리고 할당된 메모리의 pointer를 return한다. 이들은 malloc(), calloc()를 만들어 주는데 쓰인다. 그림3.2.3.2에 brk(), sbrk()동작 설명과 run-time시에 실행되는 과정을 보였다.

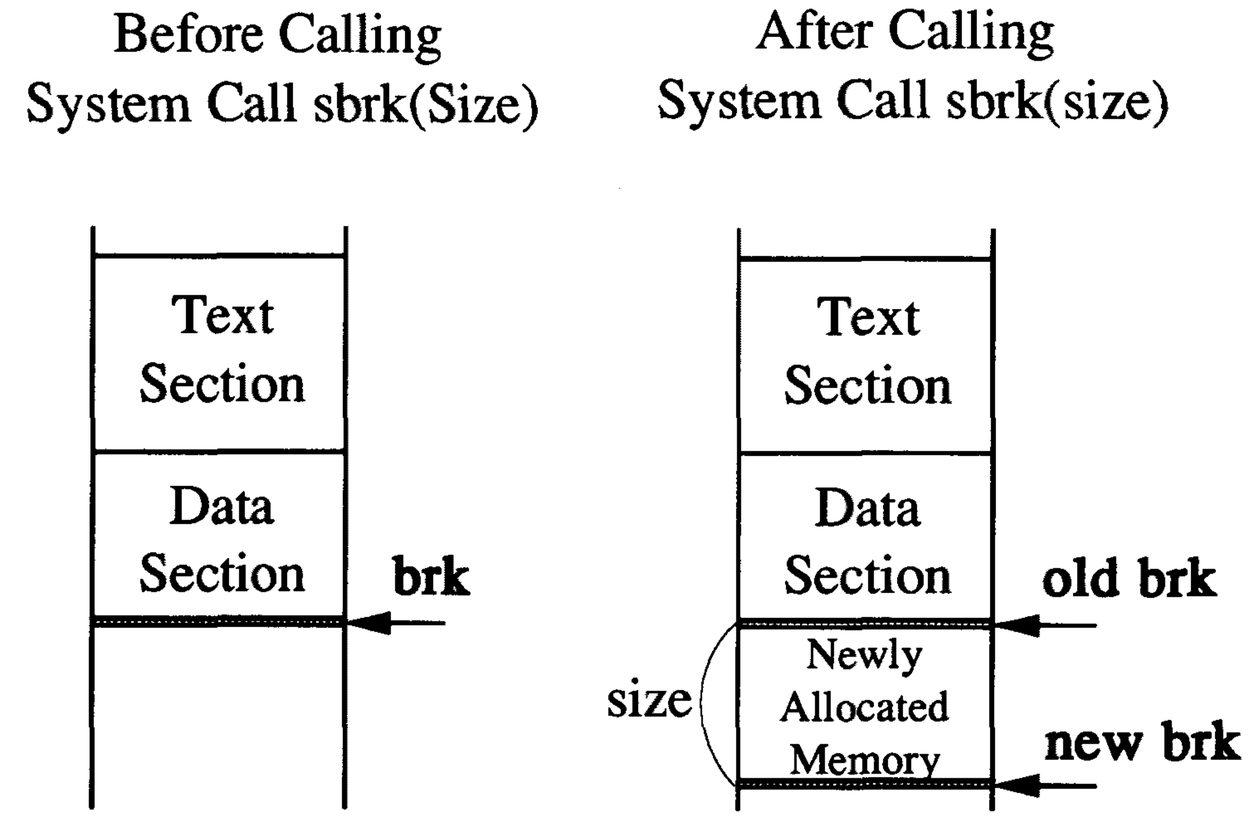


그림3.2.3.2 sbrk()의 동작