다중 프로세서를 이용한 대형 Programmable Controller 구조 및 성능 해석

박 홍 성 . 김 종 일 . 변 대 규 . 권 욱 현
서 울 대 학 교 공 과 대 학 교 제 어 계 측 공 학 과

The Architecture and Performance Evaluation of Large Programmable Controller Using the Multiprocessors

Hong-Seong Park , Jong-Il Kim, Dae -Gyu Byeon and Wook-Hyun Kwon
Dept. of Control and Instrumentation Eng., Seoul National University

ABSTRACT

This thesis investigates the scanning time ; one of the most important performance index of programmable Controller (PC). The multiprocessor architecture of the large PC considered in this thesis are classified as architecture 1 and architecture 2 by the bus control methods. The queuing model of each architecture is developed.

Form the analysis it is observed that in the case of the number of processors less than 3 the best architecture of the large PC is the architecture 2 and in the case of the number of processors greater than 2 the best architecture of the large PC is the architecture 1.

1. 서론

시스템을 생각할 때, 중요한 관심은 그 시스템이 어떤 성능을 가지고 있느냐 하는 것이다. 이를 위해 많이 사용되고 있는 방법은 성능 평가 (Performance evaluation)에 의한 것이다.

시스템의 설계 단계에서 성능 평가를 한다면, 시스템이 없어 실제 성능을 측정할 수 없으므로 모델링 기법이 사용된다.

모델링 기법을 사용하면 원하는 성능을 얻을 수 있는 최선의 하드웨어 구조 및 모니터 프로그램등이 나올 수 있다. 모델링 기법에는 해석적 기법과 수치 해석 혹은 시뮬레이션 기법이 있는데 이들 방법 모두 다 큐잉 모델이 필요하다.[1,2,3] 빠른 스캐닝 시간을 얻기위해 다중 프로세서 구조를 채택하는 PC 시스템에 대해 여러 연구가 진행되고 있다. [4,5,6,7]

본 논문에서는 다중 프로세서 구조를 선택하여 스캐닝 시간에 병목 현상을 일으키는 데이터 전달에 대하여 데이터의 일관성을 유지하며 적은 전달시간이 걸리도록 버스 제어 방법에 대하여 여러 가정을 하여 그 모델들을 분석 . 비교하였다.

본 논문에서는 마이크로프로세서인 MC* 68000을 이용한 다중 프로세서 구조 형태를 취하며 하드웨어적으로 구현하기 쉬운 시간 공유 버스 (time shared bus)를 사용하여, 각 프로세서 보드들에 일정 시간동안 타임 슬롯 (time slot)을 할당하는 방법과 선입선출 (FIFO) 방식의 TTL 큐 (queue)를 사용하는 방법에 대하여 그 성능을 분석하였고, PC의 소프터웨어가 성능 척도에 미치는 영향을 분석하였고 모델링 기법을 제시하였다.

본 논문에서 사용된 시뮬레이션 패키지는 SDL/SIM 이며, 이는 파스칼 언어로 쓰여져 VAX-11/750 혹은 780에서 수행되었다.

2. 본 시스템 (PCSYS)의 구조

2.1 시스템의 개요

본 논문에서 가정한 시스템 (이하 편의상 PCSYS라 칭함)의 구조는 다중 프로세서 구조이며 시스템 버스로는 시간 공유 공통 버스이다. PCSYS 에서 사용될 CPU는 모토롤라 (Motorola) 사의 MC 68000으로 가정하였으며, PCSYS을 구성하는 프로세서 보드들을 기능 별로 다음과 같이 분류한다.

- 시스템 제어기 (System Controller)
- 디스크리트 프로세서 (Discrete Processor) 보드
- 입출력 프로세서 (I/O Processor) 보드

이들의 기능을 간략하게 알아보면, 시스템의 제어기는 PCSYS의 모든 프로세서들을 제어하고 그 동작을 관리하는 기능을 기본으로 한다. I/O 프로세서는 PCSYS와 공정간의 데이터 입.출력을 수행한다.

디스크리트 프로세서 보드는 입력 정보에 대한 논리 연산 (Logical Operation)을 담당하여 논리 연산 속도를 향상시키기 위하여 HLS (Hardware Logic Solver, 그림 2)를 가진다. HLS는 한 그룹의 논리 연산 작업을 행 (Column) 단위를 (8개의 명령어)를 1μs에 푸는

하드웨어이다. HLS는 비트 단위의 구조를 가지고 있으며 여러 비트를 하드웨어로 동시에 계산하여 논리 연산을 수행한다. 논리 연산 중에서 AND/OR 연산을 제외한 카운터, 타이머 (timer)등의 응용 명령어는 인터럽트 방식에 의하여 CPU가 수행한다.

2.2 비교 대상 구조

구조 1은 각 디스크리트 프로세서 보드의 내장 프로그램을 완전히 수행한 후 일정 기간 (time slot)을 데이터 전달 기간으로 정하여 버스를 할당받는 구조이며, 구조 2는 HLS에서 랜더 논리를 푼 후 그 결과를 지역 메모리 (local memory)에 쓴다는 점이다. [8,9,10]

(1) 구조 1

이 구조는 여러 프로세서 보드들간의 버스 사용 경쟁 (Contention)을 방지하여 데이터 전송을 오버헤드 시간 (즉 대기 시간) 이 없도록 버스를 시간적으로 분할하여 사용하는 방식이다. (그림3)
이 구조에 사용되는 신호는 다음과 같다.

- TIME/ : BUS의 TADDRI가 유효함을 알려주는 신호
- TADDRI : 각 프로세서 보드에 할당되어 있는 고유 address
- TRDY/ : Multiple Transfer Ready
- LOCK/ : Bus Lock

(2) 구조 2

이 구조는 여러 프로세서 보드들 간의 버스 사용 경쟁을 일으킨다. 따라서 어느정도 대기 시간 (waiting time)이 생기게 된다. 이 구조를 선택한 이유는 HLS에서 부울 논리를 다 푼 후 HLS는 CPU에 인터럽트를 걸어 푼 결과를 데이터 램 (RAM)에 넣는 시간이 있는데 이 시간을 이용하여 다른 프로세서 보드들에게 데이를 전달하면 일정 시간동안 데이터를 전달하는 시간을 절약한다.

각 프로세서 보드들이 버스를 요구하면 프로세서 보드들의 고유 번호가 FIFO 큐에 들어간다. 이미 큐에 버스를 요구하는 신호 즉 번호가 존재하면 기다렸다가 버스를 할당받는다. 또 버스가 사용중이 아닐 때에는 큐에 들어가지 않고 버스를 요구한 프로세서에 즉시 버스를 할당한다. (그림4, 그림5)
이 구조에 사용되는 제어신호는 다음과 같다.

- BUS-REQ : BUS REQUEST
- BUS-BUSY/ : BUS BUSY
- BUS-REL : BUS RELEASE
- Q-ADDR : UNIQUE ADDR. OF Each Processor Board
- BUS-COMP : COmplete BUS-USE

3. 성능 분석

이 성능분석에서 사용되는 변수들을 다음과 같이 정의한다.

- Ck : CPU의 클럭의 한 주기 (8 MHz일 경우 0.125 ns)
- S : 프로그램의 길이 혹은 크기
- Np : 디스크리트 프로세서 (discrete processor) 보드의 갯수
- Ni : 입력점의 갯수
- No : 출력점의 갯수
- Nc : 컨트롤 릴레이의 갯수
- Nn : 프로그램의망 (network)의 수
- Nle : 하나의 망을 구성하는 부울 논리 원소의 갯수
- Tn : 하나의 망을 해석하는데 소요되는 시간
- Tpt : 한 워드를 옮기는데 (transfer) 소요되는 시간
- Wq : 큐에서 서비스 받을 때까지 기다리는 시간
- Wsov : 서비스되는데 소요되는 오버헤드 시간

3.1 구조1의 성능 분석

(1) 가정과 큐잉 모델

가) 구조1의 라이딩 선도에서 TIME/신호는 TRDY/신호와 LOCK/신호를 지연없이 동작 (enable) 시킨다. 즉 버스를 할당할 때 지연 시간 (delay time)은 무시된다.

나) 다른 디스크리트 프로세서 보드 혹은 CPU 보드에서 필요한 데이터는 프로그램 실행중에는 지역 메모리 (local memory)의 PM과 CM에 저장되어 자신에게 버스 사용권이 오면 다른 프로세서 보드들로 필요한 데이터를 전송한다.

다) 버스의 사용권은 버스 제어기에서 일정 시간동안 각 보드에게 할당한다.

라) 입력점과 출력점의 비율은 1:1이며 컨트롤 릴레이 (Control Relay 이하 CR 이라 칭함)의 수는 입출력 점수의 이다.

마) 스캐닝 시간내에 모든 입력 데이터를 받아들인다.

바) 프로그램은 각 디스크리트 프로세서 보드에 균일하게 분배 (Uniformly distribute) 되어 있다.

사) 응용 명령어는 하나의 망 (network)에 평균 3개씩 poisson 분포를 갖는다. (부울 논리와 혼합)

아) 하나의 망을 구성하는 랜더 논리 원소들의 평균 갯수는 E[Nle] 이다.

자) 데이터 전달은 워드 (Word) 단위로 한다.

(2) 성능평가

평균 갯수 E[Nce]는 NIo/(32*NP)워드이며 E[Noe]는 Nio/32 워드이다.

하나의 망에 응용 연산이 많을 수록 Tn은 훨씬 길어진다. 하나의 망에 포함된 응용 명령어의 평균 갯수를

E[Na]라 하고 하나의 응용 명령어를 푸는데 소요되는 평균소요 시간을 T[Tae] s라 하면 Tn은 10+E[Na]+214 Ck 이다. 그런데 사)의 가정에서 하나의 망에 응용 명령어가 평균 3개씩 들어 있고 구조1은 전체 프로그램을 다 푼 후 데이터를 전송하므로 전체 프로그램으로 보아서는 하나의 망에 평균 3개의 응용 명령어가 포함된다고 생각할 수 있다. E[Tae] = 192 Ck (부록 참조) 이다. 따라서 이를 사용하여 스케닝 시간을 구하면

$$E[t] = E[NnTn/Np]+Wsov+NioTpt/(32Np)$$
$$+ NioTpt/32+[Nio/32+Nio/32+Nio/32]Tpt$$
$$= 170Ck+1.5Ck.Nio+(10+790Ck)S/(E[Nle] Np)$$
$$+ 0.375\ NioCk/Np \quad (Np \geq 2) \qquad (3.1a)$$

Np = 1인 경우는 다음식과 같다.

$$E[T] = 0.75NioCk + (10+790Ck)\ S/E[Nle]$$
$$(Np = 1) \qquad (3.1b)$$

3.2 구조2의 성능분석

(1) 가정과 큐잉 모델

가) 2.2(2)의 그림 4 와 그림 5의 타이밍 선도에서 신호들의 지연시간은 무시된다. TTL급의 서비스시간은 0.5 μs 이다.

나) 다른 디스크리트 프로세서 보드 혹은 I/O보드에서 필요한 데이터는 하나의 망을 완전히 푼 후 그 데이터를 지역 메모리 (local memory)의 PM,CM과 다른 프로세서 보드들이 CM에 전달한다. 즉 버스사용 경쟁을 한다.

다) - 차) 구조1)의 가정 다) - 차)와 같음.

타) 디스크리트 프로세서 보드의 버스 요구 신호는 I/O 프로세서 보드의 버스 신호 요구 보다 우선 순위가 높다.

파) I/O 프로세서 보드의 버스 할당은 타임 슬롯으로 한다.

° 큐잉 모델

앞의 가정을 사용하여 큐잉 모델을 만들면 우선 순위를 가진 모델이 되며 그림7에 나타나 있다.
이 모델은 순환형 큐잉 망이 된다.

따라서 그림7의 큐잉 모델은 I/O 프로세서 보드의 특성을 사용하면 대략적으로 (approximately) 그리면 그림8와 같아진다.

(2) 성능 분석

전체 래더 논리 프로그램이 각 디스크리트 프로세서 보드로 균일하게 분포되어 있으므로 각 디스크리트 프로세서 보드의 프로그램 길이는 S/Np가 된다. 또 망의 갯수로 환산하면

$$E[Nn] = S/(E[Nle] * Np) \qquad (3.2)$$

따라서 래더 논리만을 풀었을 경우의 스케닝 시간을 E[T']라 하면

$$E[T'] = E[Tn] * E[Nn]$$
$$= (Top+Tser+Wq)S/(E[Nle]Np) \qquad (3.3)$$

이다. 스케닝 시간에는 E[T']에 입력 데이터를 각 프로세서 보드에 전달하는 시간 E[Ti]와 CM에서 PM으로 데이터를 전달하는 시간 E[Tcp]를 더해야 한다.

$$E[Ti] = NioTpt/32 \qquad (3.4)$$

$$E[Tc-p] = 3NioTpt/32 \qquad (3.5)$$

(3.3)식, (3.4) 식과 (3.5)식을 더하면

$$E[T] = (Top+Tser+Wq)S/(E[Nle]Np)+NioTpt/8 \qquad (3.6)$$

표 3.1 - 표 3.3에서 구한 평균 대기 시간은 시뮬레이션에서 나온 데이터를 해석하여 나온 결과이다. 즉 작업이 큐에 도착하여 서비스를 완료할 때까지의 시간들을 해석함으로써 추출하였다.

E[T]는 시뮬레이션에 의해서 직접 추출할 수 도 있지만 (3.2) 식 (3.3)식, (3.4)식과 (3.5)식을 이용하여 구할 수 있다.

즉 Np ≥ 2에 대해서

$$E[T] = (Top+Tser+Wq) * S/(E[Nle] * Np)$$
$$+3 * Nio * Tpt/32$$
$$= (10.5+790 * Ck+Wq) * S/(E[Nle] * Np)$$
$$+1.125 * Ck * Nio \qquad (3.6a)$$

Np = 1 인 경우

$$E[T] = (Top+Tser) * S/E[Nle]+Nil * Tpt/32$$
$$= (10.0+790 * Ck) * S/E[Nle]$$
$$+0.375Nio\ Ck \qquad (3.6b)$$

4. 구조1과 구조2의 비교.분석

(3.1)식의 각 항에서 (3.6)식의 각 항을 뺀 후 그 결과를 E[T]라 하면

$$E[T] = \begin{cases} 0.375\ Nio\ Ck & (Np = 1) \quad (4.1a) \\ 170\ Ck + 0.375\ Nio\ Ck/Np & \\ \quad + 0.375Nio\ Ck - Wq.S/(E[Nle].Np) & \\ \quad (Np \geq 2) \quad -------- & (4.1b) \end{cases}$$

(4.1)식을 살펴 보면 E[T]는 Nio, Ck, S, E[Nle]와 Wq의 함수가 된다. 그림9는 Ck = 0.125 s일 때 E[T]의 변화량을 나타낸 것인데 Np ≦ 7일 때 까지는 구조2가 구조1보다 빠른 스케닝 시간을 가졌다. 역시 Nio의 영향에 따라 스케닝 시간이 많이 변함을 알 수 있다.

그림9과 그림10를 살펴 보면 Np ≤ 2일 때는 구조2의 스캐닝 시간이 빠르고 3 ≤ Np ≤ 7일 때는 Ck와 Nio에 따라 구조1 혹은 스캐닝 시간이 빠를 수 도 있지만 Np >7 일 경우에는 구조1의 스캐닝 시간이 빠르다.

5. 결론

PCSYS의 버스 제어 방법을 타임 슬롯을 사용하며 데이터를 전달하는 구조1과 선입선출 큐를 사용하여 버스 사용 경쟁을 하며 데이터를 전달하는 구조2에 대하여 각각의 스캐닝 시간을 분석하였고, 또 그들을 비교 분석하였다.

또 구조1과 구조2를 비교 분석하여 얻은 결과는 다음과 같다.

- 스캐닝 시간은 구조1에서는 디스크리트 프로세서의 갯수에 반비례하나 구조2에서는 반비례하다가 NP가 어느 값 이상일 때는 일정하다.
- 입출력 점의 수는 데이터들이 스캐닝 시간에 영향을 별로 안 준다.
- HLS에서는 논리를 해석한 후, 그 결과 데이터를 PM으로 전송하는 알고리즘은 스캐닝 시간에 많은 영향을 준다.
- 3 ≤ Np < 7일때는 입, 출력 점의수와 프로그램의 크기와 CPU의 한 클럭 주기에 따라 구조1 혹은 구조2의 스캐닝 시간이 빠르다.

앞으로 수행되어야 할 과제는 다음과 같은 것이 있다.

- 입, 출력 프로세서에 있어서 외부환경과의 데이터를 주고 받는 등의 입출력 프로세서의 정확한 동작을 모델링 하여 시뮬레이션 혹은 해석적 방법에 의하여 분석
- 하드웨어에 있어서 전달지연 시간을 고려하여 정확한 스캐닝 시간을 얻기위한 모델링

참 고 문 헌

[1] C.H. Sauer and K.M. Chandy, Computer Systems Performance Modeling, Prentice-Hall, New Jersey, 1981.

[2] L. Kleintock, Queueing Systems Volume I : Theory, Wiely, New York, 1975.

[3] IBM, Analysis of Some Queuing Models in Real-Time Systems, IBM, New York, 1971.

[4] GOULD Inc., 584. Microcode Machine Spec., Gould, Jan. 1980.

[5] TI Inc., Texas Instruments Industrial Control Products, Model 560/565 Product Profile, Texas Instrument.

[6] G.A. Tendukar, " A Programmable Controller with Parallel Processors For High Performance Industrial Applications, " Proceedings IECON '84 Volume 2, pp.902-906, 1984.

[7] 박홍성 " 프로그램형 논리 다중 프로세서 구조와 성능평가에 관한 연구, " 서울대 석사학위 논문, 1986.

[8] D.P. BHANDARKAR, " Analysis of Memory Interference in Multiprocessors, " IEEE Trans. on Computers C-24, 9, pp.897-908, Sep. 1975.

[9] J.H. PATEL, " Performance of Processor-Memory Interconnections for Multiprocessor, " IEEE Trans. on Computers C-30, 10, pp.771-780, Oct. 1981.

[10] M.A. Marsan, " Modeling Bus Contention and Memory Interference in a Multiprocessor System, " IEEE Trans. on Computers C-32, 1, pp.50-72, Jan. 1983.

부록. 성능 평가를 위한 알고리즘들의 수행 주기 계산

1. CPU 보드에 있어서 망 (network)의 처리 시간

1.1 HLS의 해석

하드웨어 로직 솔버는 비트 슬라이스 (bit slice) 개념을 사용하여 구성되어 있는데 그 블록 선도는 그림 2에 있다. 이 하드웨어 로직 솔버로써 부울 논리를 풀 때는 1MHz 클럭을 사용하는데 하나의 열 (colunn)을 푸는 데 1μs가 소요되므로 10개의 열을 풀 때는 10μs가 소요된다. 그리고 푼 결과를 데이터램 (data RAM)과 공용 메모리에 써야 (write) 한다. 이 때 소요되는 시간은 다음의 알고리즘들로부터 구할 수 있으며, 이 소요 시간에 10μs를 더한 값이 하나의 망을 처리하는 시간이 된다.

1.2 Data 전달 알고리즘

8개의 버퍼에 BUFFER에서 BUFFER+7 까지의 어드레스를 할당한다. 8개의 결과 비트를

저장하여 두는 레치의 첫번째 어드레스를 LATCH 라 하며 마지막 어드레스는 LATCH+7이다. 이러한 어드레스들을 사용하여 결과 비트들을 DATA RAM에 저장하는 루틴은 다음과 같다. 이러한 루틴을 사용하였을 때, 데이터를 전달하는데 소요 주기는 214 주기이며 8MHz의 클럭을 사용할 경우는 하나의 망을 처리하는데 필요한 소요 시간은 26.75μs가 된다.

Interrupt걸렸을때		44	cycles
MOVEM.W	A0-A1, (A6)-	18	cycles
MOVE.W	# BUFFER, A0	8	cycles
MOVE.W	# LATCH, A1	8	cycles
MOVE.B	(A1)+, (A0)+	12	cycles
MOVE.B	(A1)+, (A0)+	12	cycles
MOVE.B	(A1)+, (A0)+	12	cycles
MOVE.B	(A1)+, (A0)+	12	cycles
MOVE.B	(A1)+, (A0)+	12	cycles
MOVE.B	(A1)+, (A0)+	12	cycles
MOVE.B	(A1)+, (A0)+	12	cycles
MOVE.B	(A1)+, (A0)+	12	cycles
MOVEM.W	(A6)+, A0-A1	20	cycles
RTE		20	cycles

1.3 부울 논리와 응용 연산이 혼합되어 구성

응용 연산과 부울 논리가 혼합되어 사용될 경우는, 하드웨어 로직 솔버로써는 응용 연산을 해석하지 못하므로 68000 CPU를 사용하여 그들을 푼다. 부울 논리를 풀다가 응용 연산이 나오면 인터럽트를 CPU에 걸어 그 연산을 처리하게 되므로 위의 부울 논리로만 구성된 망 (network)을 처리한 시간에 응용 연산을 풀기 위해 소요된 시간을 더하면 혼합되어 구성된 망의 처리 시간이 된다. 응용 연산을 풀기위한 알고리즘은 다음과 같다. 이 알고리즘에서 사용되는 변수들을 생각하면, INTLAT는 블록 선도의 INTERRUPT LATCH의 어드레스를 나타내고 그 LATCH의 내용은 떤 행에서 인터럽트가 일어 났음을 표시한다. 비트 1이 1이 되어 있으면 첫번째 행에 응용 연산이 있다는 의미가 된다. INST는 INSTRUCTION RAM의 어드레스를 나타내기 위한 마스크 (mask) 수이다. 이렇게 하여 구한 어드레스 A1은 실제적인 응용 연산을 푸는 루틴의 어드레스가 된다. 응용 연산을 풀기 위한 오보헤드 시간은 다음의 주기를 모두 더한 값 144 주기가 된다.

인터럽트 발생		44	cycles
MOVEM.W	A0/A1,(A6)+	20	cycles
MOVE.W	CNT,D1	12	cycles
MOVE.W	INTLAT,D0	12	cycles
OR.W	D1, D0	4	cycles
MOVE.W	D0, D1	4	cycles
AND.W	# INST, D1	12	cycles
MOVE.W	D1, A0	4	cycles
MOVE.W	(A0), A1	8	cycles
JMP	(A1)	8	cycles
.....			
.....	실제 응용 연산 루틴(부록3참조)		
.....			
RTE		20	cycles

1.4 Tpt의 계산

하나의 CPU 보드에서 다른 CPU 보드로 데이터를 전달할 때 68000 어셈블리 명령어인 move.w (source), (destination)을 사용하여 16 비트씩 전달한다. 그런데 이것의 수행 주기 (cycle)은 12주기이므로 클락을 8MHz로 하였을 경우 1.5μs가 소요된다.

1.5 Wsov 의 계산

SYSCNTR 이 각 CPU보드에 버스 사용권을 주기 위해 인터럽트를 걸면 인터럽트를 받은 CPU 보드는 일정 시간 동안 버스를 사용한 후 사용권을 다시 SYSCNTR에 주어 다른 CPU보드에 버스 사용권을 주도록 한다. 여기서의 Wsov는 각 CPU보드에서 데이터 전달 시, 데이터 전달 이외에 필요한 오보헤드 (overhead) 시간이다. 따라서 이를 assembly로 program 하여 계산하면 Wsov는 130 주기이며 8MHz 클럭을 사용할 경우 16.25μs이다.

1.6 응용 명령어의 사용 빈도의 가정

응용 명령어들이 균일하게 (uniformly) 사용된다고 가정하면 응용 명령어의 평균 수행 주기는 약 48주기가 된다. 성능 분석을 할 경우 이 가정도 필요하지만 해석적 모델을 간단하게 하기 위해서 응용 명령어의 수행 주기는 일정하게 (deterministic) 결정 되었다고 가정한다.

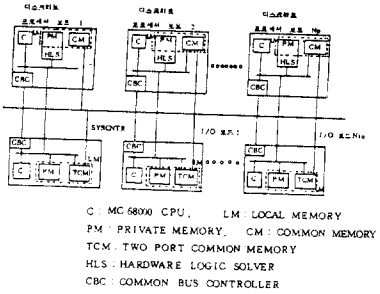

그림 1. PC SYS의 전체블록선도

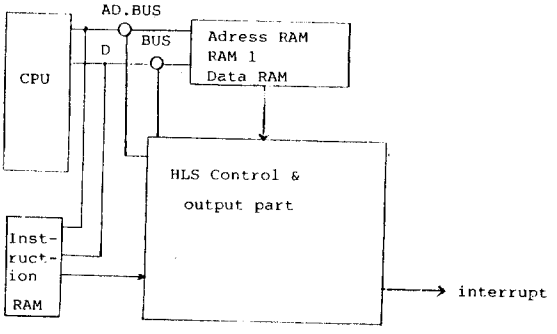

그림2. HLS의 블록선도

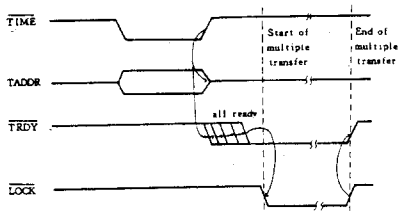

그림 3. 구조 1의 버스 제어 타이밍 선도

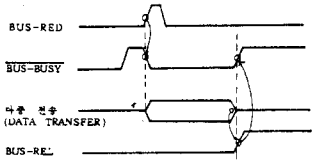

그림 4 버스가 사용 중이 아닐 때

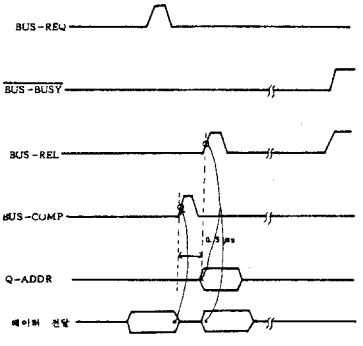

그림 5 류에 내용이 존재할 경우

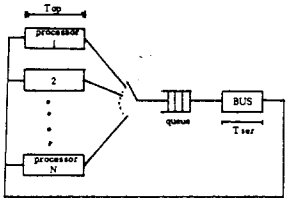

그림 6 파일출롯을 이용한 류잉 모델 (N = Np + Niop)

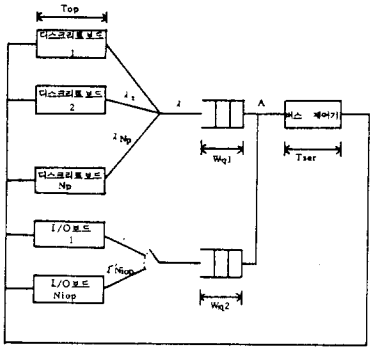

그림 7 구조 2의 류잉 모델

λi : 도착율(arrival rate)

W_{q1} : 류에서의 대기 시간

Tser : 버스 제어기에서 서비스 시간

A : 우선 순위에 의한 버스 제어기 할당

Top : 하나의 망을 통해 걸리는 시간

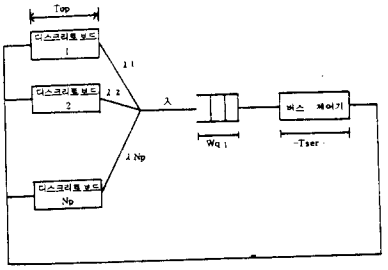

그림 8 구조 2의 근사화된 류잉 모델

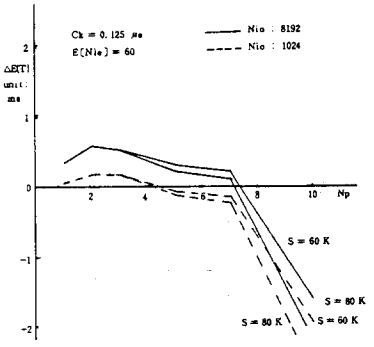

그림 9 응용 명령어 해석 시 ΔE(T)의 변화 (Ck = 0.125 μs)

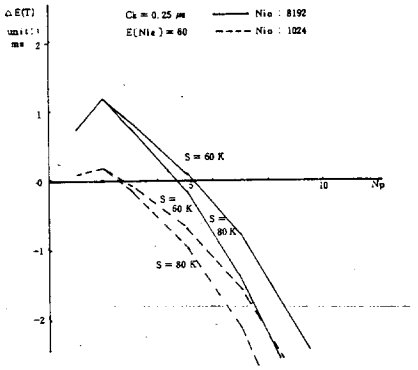

그림10. 응용 명령어 해석 시 ΔE(T)의 변화 (Ck = 0.25 μs)

Np	평균대기시간($W_{q_{avg}}$) μs	E(Ts) μs	E(T) ms
2	0	100.25	10.797
3	0	100.25	13.302
5	0.09	100.09	8.665
7	1.66	101.66	8.301
10	38.57	138.82	5.807

표 1 Ck = 0.125 μs, Nio = 8192, E(Nie) = 60 일때의 평균 대기 시간과 소해닝 시간 (S = 80 K logic)

Np	평균대기시간 μs	E(Ts) μs	E(T) ms
2	0	200	17.600
3	0.08	200.08	20.043
5	4.18	233.18	16.788
7	11.99	235.99	13.639
10	37.05	265.05	11.103

표 2 Ck = 0.25 μs, Nio = 8192, E(Nie) = 60 일때의 평균 대기 시간과 소해닝 시간 (S = 80 K logic)

Np	평균대기시간 μs	E(Ts) μs	E(T) ms
2	0	400.5	75.813
3	1.15	401.65	59.873
5	7.48	407.9	35.795
7	21.20	421.7	25.434
10	60.03	460.53	22.710

표 3 Ck = 0.5 μs, Nio = 8192, E(Nie) = 60 일때의 평균 대기 시간과 소해닝 시간 (S = 80 K logic)