공유메모리 방식에 의한 병렬컴퓨터의 구현

김 원종
(서울대 공대 제어계측공학과 4년)

차 례

1. 서 론

1970년대에 들어와 반도체 기술의 혁혁한 발전과 마이크로프로세서 구조의 진보로 연산속도 및 데이타처리량에 있어서 괄목할 만한 성장을 보았다. 그러나 이에는 아직도 한계가 있어서 극히 빠른 처리를 요하는 곳(발사체 등의 실시간 제어)과 많은 데이타의 취급(디지틀 신호처리 등)에 이용하기에는 많은 어려운 점이 있다.

따라서 새로운 컴퓨터 구조인 병렬처리 기계(parallel processing machine)에 대한 연구, 더 나아가서는 폰 노이만(Von Neumann)방식을 극복한 차세대 컴퓨터 창조의 시도가 활발해지고 있다. 이에 발맞추어 이번 연구를 통하여 병렬처리 개념에 대한 이해를 높이고, 이를 실제 시스템에 응용할 수 있는 능력을 배양하는 것을 이번연구의 목적으로 하였다.

병렬컴퓨터는 그 구조적인 차이에 따라 파이프라인 컴퓨터(pipelined computer), 어레이프로세서(array processor), 멀티프로세서(multiprocessor)시스템의 세 범주로 크게 분류될 수 있다.[1] 또한 각 병렬처리 모듈간의 결합방식에 따라 굳게 결속된(tightly-coupled)시스템과 느슨히 결속된(loosely-coupled)시스템으로 구분된다. 이 연구의 내용을 위 범주에 의하여 구분한다면 '굳게 결속된 어레이프로세서 시스템'이라고 할 수 있을 것이다.

여기서 어레이프로세서 모듈은 산술 및 논리 연산을 행할 수 있는 처리장치(processing unit)와 이의 명령어들을 저장하는 국부기억장치(local memory), 또 이 처리장치가 배타적으로 관장하는 국부자원(local resource)들을 포함하고 있다.

굳게 결속된 시스템이란, 코프로세서(coprocessor)시스템이나 각 모듈간에 중복된 번지를 갖는 공유기억장치(s-

hared memory)를 이용하여 각 모듈간의 정보전송을 실현하는 시스템을 일컫는다. 느슨히 결속된 시스템은 입출력 포트(port)를 통하여 데이타전송을 하는 것으로 LAN (local area net)도 이 부류에 속한다고 할 수 있다.

이 연구에서는 각 모듈간의 데이타 전송에 보다 적은 시간이 소요되고, 전달방식에 있어서 융통성 있는 구조를 소프트웨어로 구현할 수 있는 장점을 지닌 공유메모리에 의한 굳게 결속된 병렬컴퓨터를 구현했다. 그 위로 병렬 입출력장치(8255 PIO)의 포트 두 개를 확장용으로 남겨놓아, 더 진보적인 구조인 하이퍼큐브(hypercube), 메시지 전달방식(message passing method)등의 다중컴퓨터로의 확장성도 고려하여 설계하였다.

그러나, 이와 같은 굳게 결속된 구조를 택함으로써 공유자원 구조의 구현, 각 모듈간의 데이타, 어드레스 및 제어버스의 충돌(contention)을 방지하고 이들 버스의 중재(arbitration) 를 적절히 해 줄 수 있는 논리회로의 개발이 필연이었으며, 이들 문제의 해결이 역시 최대 난점이 되었다.

다음 장부터 전체시스템의 개요 및 구조와 시스템의 운영을 위한 소프트웨어에 대하여 논하겠다.

2. 시스템의 개요

2.1. CPU 및 주변 장치의 선택

현존하는 16비트 마이크로프로세서는 크게 인텔(Intel)사의 i8086계열, 모토롤라(Motorola)사의 MC68000계열, 자이로그(Zilog)사의 Z8000계열 등이 있다. 이 중 병렬처리에 하드웨어 및 소프트웨어적으로 적합한 구조와 기능을 제공하는 것은 i8086계열과 MC68000계열이다. MC68000 계열은 번지지정(addressing) 가능한 기억용량 및 산술계산 능력면에서 뛰어나다. 하지만, 전세계적으로 개인용컴퓨터 시장을 장악하고 있는 IBM PC가 인텔계열의 칩(chip)들을 채택했기에 이에 빛이 가리워졌다. (i8086, MC68000, Z8000들의 병렬처리 수행에 관한 해석은 ref.8 참조)

이 연구에서는 설계자에게 가장 친숙한 i8086계열의 CPU 및 그 주변장치를 사용했다. 그 이유는 i8086계열은 병렬처리의 핵심이라고 할 버스중재문제 해결에 큰 도움을 줄 수 있는 우수한 구조들을 지니고 있고, IBM PC등 강력한 개발환경을 활용할 수 있기 때문이다.

2.2. 시스템의 기본 구성

병렬컴퓨터 시스템의 주요 구성요소는 다음과 같다.

a. 클록부

각 모듈에 독립된 8284 클록발생기가 쓰인다. 또한 이 두 8284를 동기화 시키기 위한 마스터클록발생기로 한 개의 8284가 쓰인다. 두 개의 8284는 또 CPU의 동작을 제어(READY, RESET등으로)하는 신호를 발생시킨다.

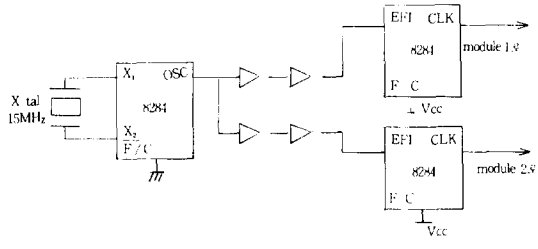

그림 1. 여러 개의 시스템을 위한 외부 주파수원

b. 기억장치부

기본적으로 0FC000H~0FFFFFh 번지에 모니터 ROM (16KB), 000000h~00FFFFh에 사용자 프로그램용 RAM (64KB), 010000h~013FFFh에 공유RAM(16KB)을 설치했다. 이정도 크기의 기억용량은 어셈블리어 수준의 프로그래밍에는 충분하며 고수준 언어(high-level language)의 이식에 대비하여 총 320KB를 기판상에서 확장가능하도록 설계했다.

c. 버스중재부

CPU의 잠금(lock) 신호, 8255출력, 다른 모듈의 버스 요구 상태를 입력으로하여 버스중재를 해결하는 8289버스중재기가 이 부분에서 핵심되는 요소이다. 또, 데이타 및 어드레스버스의 개·폐를 직접 제어하는 버스제어장치와 래치, 송수신기가 여기 포함된다.

d. 국부자원부

CPU와는 독립적으로 작동하는 하드웨어 계수기(8253)와 버스중재기에 대하여 CPU의 제어신호를 발생시키는 8255병렬입출력장치가 이에 속한다.

e. 공유자원부

IBM PC로부터 사용자 프로그램을 전송받고 시스템의 상태를 모니터링하는데 쓰이는 직렬 입출력장치 및 보드율(baud rate)발생기, 각 모듈간 전송될 데이타가 저장될

공유메모리등이 이에 속한다. PC와의 접속은 RS232C 직렬포트를 통한다.

2.3 시스템의 특징—제어신호의 궤환

버스중재 문제는 뒤의 제 3장에서 자세히 언급하겠지만, 이 연구의 어려운 점은 각 모듈이 일종의 궤환(feed-back) 시스템이라는 점이었다.

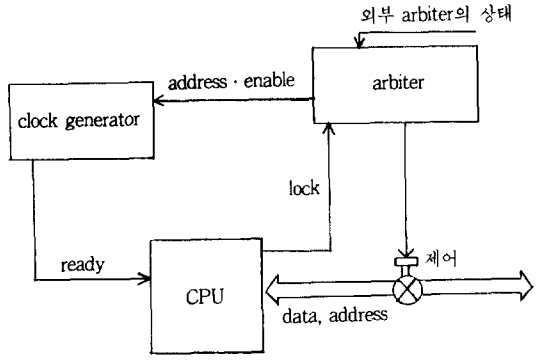

그림 2. 제어신호의 궤환

즉, 그림 2에서 알 수 있는 바와 같이 버스중재기(arbiter)는 상대편 중재기, CPU의 잠금(lock) 신호 등의 정보들을 이용하여 클록발생기에 어드레스 가능 신호가 비활성 상태일 때, 즉 상대편 모듈이 공유버스를 장악하고 있을 때 READY신호를 비활성상태로 만들어 CPU가 기다리게끔 한다. 이 루우프의 모든 요소가 동시에 완벽히 동작해야 하므로 제작초기에 하드웨어 수정하는데 큰 어려움을 겪었다.

2.4. 메모리 맵된 공유 I/O (Memory mapped shared I/O)

Intel계열의 프로세서는 I/O에 어드레스를 지정해 주는 방식으로써 'I/O mapped I/O'방식을 택하고 있다. 이는 I/O장치를 메모리와는 다른 것으로 간주하여 I/O READ, WRITE신호를 따로 발생시킨다. 이는 I/O장치와 기억장치를 같은 종류의 주변장치로 인식하여 I/O 번지를 연속인 메모리번지 중 하나로 지정하는 'memory mapped I/O'방식과 구별된다.

한편 Intel 데이타북에는[4] 아무런 문제없이도 공유자원으로 기억장치 및 입출력장치를 같이 쓸 수 있는 것처럼 기술해 놓았다. 그러나, 8251직렬입출력장치의 디코우딩 코드(decoding code)와 같은 메모리번지의 내용에 접근(access)하려하면 시스템버스쪽의 래치도 열려버려 버스충돌이 일어나게 된다.

위와 같은 사실을 해결하기 위하여 하드웨어적으로 여러 대책을 강구했으나 그리 효율적이지 못하였다. 그래서 이의 해결에 Intel 계열에서는 파격적이라할 'memory maped I/O'방식을 사용했다. 즉 직렬 입출력장치의 데이타 레지스터를 메모리 01FFF0h번지에, 제어 레지스터를 01FFF0h에 지정하여 이의 입출력을 메모리 'MOV' 명령으로 수행하였다.

이 방법이 가능하게 된 것은 014000h번지 이후는 사용하지 않는 메모리 영역이고, i8086 최대모우드(maximum mode)에서의 메모리 입출력과 I/O입출력 신호의 파형이 완전히 일치하기 때문이다.

```
char-in : mov al, es : [siocon]
          and al, 00000010b
          jz char-in
          mov al, es : [siodata]
          call char out
          ret
char-out : mov al, es : [siocon]
          and al, 00000001b
          jz char out
          mov al, ah
          mov es : [siodata], al
          ret
```

그림 3. memory mapped I/O에 의한 입출력 루우틴

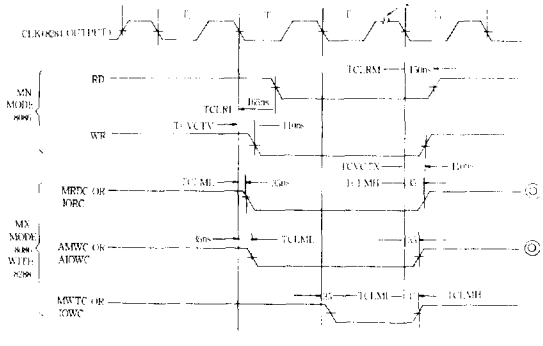

그림 4. 8086 명령 타이밍

3. 병렬컴퓨터의 구현

3.1. 리세트 신호

리세트 신호의 지속은 그림 5에서의 RC시상수에 의해서 결정된다. Intel데이타북에는 이 시간이 162μs이면 충분하다고 되어있으나(ref. 4 p.3-304 참조), 이는 스위치에서 발생하는 채터링(chattering)의 영향을 무시한 값으로, 이대로 저항과 캐패시터를 달아보았더니 리세트 성공률이 20%도 채 안되었다. 이 문제는 R=270kΩ, C=4.7μF로 시상수를 크게 늘려서 1.27s가 되게 함으로써 해결 가능했다.

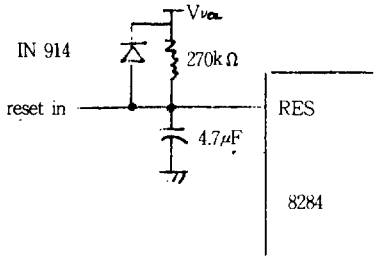

그림5. 리세트 회로

3.2. 8284 리세트출력의 채터링 문제의 해결

두번째 모듈의 제작시에 CPU의 리세트 입력에 들어가는 선에서 원인 불명의 채터링이 발생하는 문제가 생겼다. 이의 해결을 위해 여러 실험을 한 끝에 그림 6과 같은 회로를 고안했다.

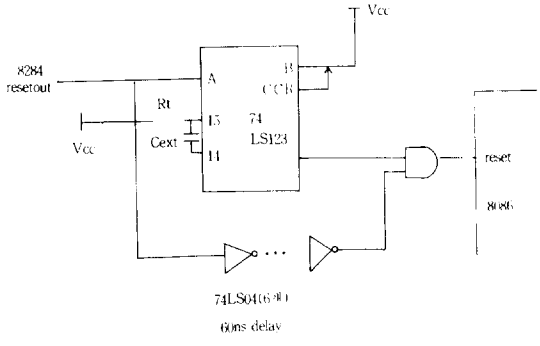

그림 6. 8284 RESETOUT 채터링의 해결

8284 RESETOUT 단에서 발생하는 채터링의 지속시간을 오실로스코우프로 측정해 보니 수 십ms였다. 74LS123 단안정 발진기의 지속시간 결정식은 아래와 같다.[6]

$$tw=0.45\times R_T\times Cext$$

여기서 R_T=200kΩ, Cext=4.7μF로 하여 tw=0.423s가 되게 하였다.

이 회로는 리세트 출력이 '0'상태가 되는 최초의 에지(edge)에 단안정 발진기가 동작하여 8086의 리세트 입력을 비활성화 시킨다. 즉, 0.423s 동안 발생하는 채터링의 영향을 완전히 제거할 수 있었다.

3.3. 잠금(LOCK) 신호의 발생

버스중재기의 잠금 입력은 CPU의 $\overline{\text{lock}}$출력을 인지하여 버스를 절대로 상대편 중재기로 방출하지 않게 하는 역할을 한다. 모니터 프로그램 하에서 버스를 한 모듈에 대해서만 배타적으로 사용할 때에는 외부에서 영속적으로 버스 중재기의 잠금입력을 활성화시켜야 한다. 이는 병렬입출력장치의 한 비트를 CPU의 $\overline{\text{lock}}$ 신호와 AND 게이트로 연결해서 구현했다. 이에 따른 버스 잠금 및 방출 루우틴은 그림 7과 같다.

```
lock : in al, [pioc]
       and al, 01111111b
       out [pioc], al
       ret


release : in al, [pioc]
          or al, 10000000b
          out [pioc], al
          ret
```

그림 7. 버스 잠금 및 방출 루우틴

3.4. 공유자원의 번지지정

앞서 언급한 바와 같이 직렬입출력장치의 번지지정(addressing) 은 메모리 매핑방식으로 하였다. 이 방식으로 번지를 지정하기 이전에 고찰한 하드웨어적인 해결방법을 열거하면 다음과 같다.

(1). 8251의 $\overline{\text{CS}}$신호를 CPU의 S2신호와 OR게이트로 묶는다.

(2). 8288의 제어신호($\overline{\text{IORD}}$, $\overline{\text{AIOWC}}$)를 S2와 OR게이트로 묶는다.

(3). 국부자원의 $\overline{\text{CS}}$신호의 발생시에는 강제로 어드레

스 및 데이타 래치의 출력을 고임피던스(high-impedance) 상태로 만든다.

위의 (1), (2)항은 실제로 실험해 보았으나 전파지연때문에 타이밍이 맞지 않아서 실패했다. 제 (3)항은 버스제어기의 어드레스 래치 가능 신호를 직접 제어하는 것으로써 게이트를 추가로 달아야 하는 등 번거롭다. 따라서 메모리 매핑(memory mapping)에 의한 번지지정 방식을 채택하게 되었다.

3.5. 전파지연(propagation delay)에 대한 고찰

이 병렬컴퓨터에는 제어 칩들이 많이 쓰이고 8086→8289→8284→8086의 궤환 루우프도 포함되어 있어서 전파지연에 대한 타이밍 문제를 철저히 연구했다. 가장 복잡하고 까다로왔던 공유자원을 액세스(access)할 때까지의 시간을 계산해 보았더니 최악의 경우에 287ns가 걸렸다. 하지만, 8086 신호 시간표(ref. 2의 부록 C-7참조)에 의하면 위 지연의 최대 허용치는 350ns로 약 60ns의 여유가 있음을 확인했다. 다른 모든 시간 문제도 위와 같이 확인한 다음 설계했다.

3.6. 버스 장악의 우선순위

여러 프로세서가 동시에 시스템버스를 요구할 때를 대비한 버스중재기끼리의 우선순위 문제를 해결하는 방법에는, 병렬 우선순위 해결기법(parallel priority resolving technique)과 직렬 우선순위 해결기법(serial priority resolving technique)이 있다. 이 연구에서는 후자의 한 방법인 데이지 체인(daisy chain)기법을 썼다.

4. 시스템 및 실용 프로그램

4.1. 각 모듈의 시스템 버스 장악

이 병렬컴퓨터의 모니터 프로그램이나 다운로드 (download) 명령 상의 커서(cursor)는 '1)', '2)'로 표시했다. 이는 각기 모듈이 배타적으로 버스를 사용하고 있고 키보드로부터의 모니터 명령을 받아들일 준비가 되어있다는 표시이다. 따라서 이는 단순한 커서 이상의 의미를 갖는다. 물론 사용자 프로그램의 수행상태에 있어서는 각 모듈이 필요에 따라서만 버스를 장악하지만, 모니터 상태에서는 배타적으로 버스를 사용할 수 있는 권리를 각 모듈에 주어야만 명령에 대한 혼동을 방지할 수 있다.

모니터 상에서 버스의 제어상황을 변경하는 흐름도를 그림 8에 나타내었다. 잠금(lock)및 방출(release)신호는 그림7의 루우틴과 같이 병렬입출력 포트를 사용한다.

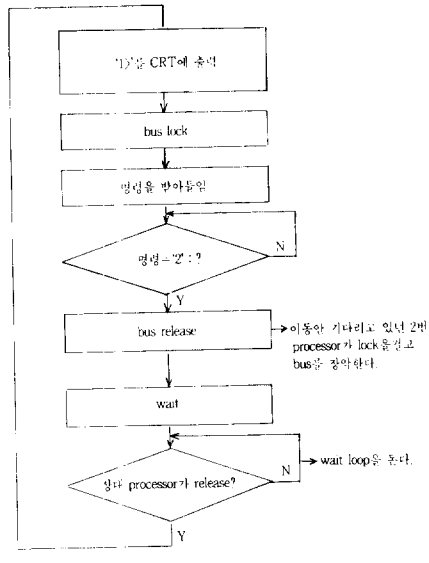

그림 8. 모니터 상에서 버스 장악 모듈의 변경

4.2. 모듈간의 데이타 전송

공유메모리를 사용하는 것이 입출력 포트를 사용하는 것보다 더 효율적이고 융통성있는 데이타 전송이 가능하다는 것은 전술한 바와 같다. 입출력 포트에 의하여서는 한번에 기껏해야 한 바이트씩 밖에 주고받을 수 밖에 없고, 받는 쪽의 모듈이 데이타를 가져갈 때까지 주는 쪽이 기다려야 한다는 치명적인 약점이 있다. 따라서, 이는 많은 데이타의 연속전송과 같은 국한된 용도 이외에는 비효율적이다. 물론 입출력 포트에 의한 방식은 공유메모리 방식보다 하드웨어 설계상의 간편함이라는 잇점은 있다.

이에 반하여 공유 메모리 방식은 공유메모리에 전송 데이타의 모든 데이타 구조(벡터, 다차원 배열, 레코오드, 포인터 등)를 구현할 수 있고, 데이타 수형(type) [문자형, 정수, 고정소숫점(fixed point) 실수, 부동소숫점(floating point)실수]의 의미 단위씩의 전송이 가능하다. 즉 고도의 효율성과 소프트웨어 상의 융통성을 발휘한다.

또한, 공유메모리 방식의 큰 잇점 중의 하나는 데이타를 주고받는 동작간에 시간의 차이가 허용된다는 것이다. 그러나 이런 잇점에 부가하여 공유메모리로 옮겨진 데이

타의 확인을 위하여 새로운 장치가 요구된다. 이를 위한 우수한 방법 중 하나는 공유메모리 내에 세마포어(semaphore) 를 설정하는 것이다.[5]즉, 한 모듈이 공유메모리를 사용하기 위하여서는 세마포어에 '사용중'이라는 표시를 한 다음에 사용하도록 하는 것이다. 이의 가장 간단한 구조인 단일 세마포어에 의한 데이타의 전송에 대한 흐름도를 그림9에 나타내었다.

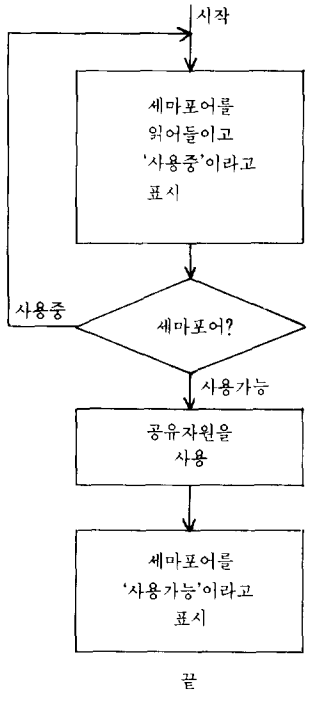

그림 9. 단일 세마포어 루우틴

본 병렬컴퓨터에서는 이중 세마포어를 사용하고 있다. 즉, 다른 모듈로 데이타를 보내는 채널, 받는 채널을 설정하여 두 개의 세마포어를 두 모듈이 공동으로 운영하는 것이다.

4.3. 다운로드(down load) 프로그램

이 병렬컴퓨터의 특장중 하나가 사용자의 프로그램을 ROM에 옮기는 번거로움 없이 IBM PC를 통하여 플라피 디스크(floppy disk)에서 직접 시스템 RAM으로 다운로드해서 동작시켜 볼 수 있다는 것이다.

모든 사용자 프로그램은 인텔 헥사포맷(Intel Hexa Format) 으로 변환되어진 다음 다운로드된다. 인텔 헥사 포맷으로 변환된 파일은 파일의 크기, 체크 섬(check sum), 파일의 끝과 같은 유용한 정보를 포함하고 있다.

다운로드되는 초기 번지는 000I00h로서 이는 변경 가능하다.

4.4 모니터 프로그램

시스템의 상태를 한 눈에 알 수 있고 여러 시스템 함수(system function)를 이용할 수 있게 하는 모니터 프로그램은 새로운 시스템 개발에 필수 요소이다. 이 병렬컴퓨터는 다음과 같은 명령어를 포함하고 있다.

1. '2'(또는 '1') : 이는 커서를 다른 모듈의 것으로 바꾸는 명령이다. 즉, 현재의 모듈로 하여금 공유버스를 방출하게끔 한다.
2. 'r' : 이는 모든 CPU레지스터의 내용을 나타내는 명령이다. IP(instruction pointer)레지스터는 자유롭게 내용을 볼 수 없기 때문에, 그림 10과 같이 CALL 명령 수행시 스택 맨 위에 IP가 저장되는 점을 이용했다.

```
reg-content   MACRO reg
              mov ax, reg
              call reg-out
              mov ah, out
              call char-out
              ENDM
              call next : IP print routine
next :        pop ax
              sub ax, 3h
              reg-content ax
```

그림 10. IP의 내용을 인쇄하는 루우틴

3. 'l' : 이 명령을 받으면 시스템이 다운로드되는 데이타를 받을 준비가 되었다는 메시지를 보낸다. Alt-T 키이를 누르면 전송할 파일을 묻고, 파일을 전송받은 다음 다운로드가 성공했다는 메시지를 보낸다.
4. 'c', 's' : 이는 계수기 시동 명령 및 중지 명령이다. 경과시간을 10진수 10자리로 나타낸다. 계수단위는 1㎲이다.
5. 'x' : 이는 설정된 NMI(non-maskable interrupt)벡터를 취소하는 기능을 한다. 즉 이 명령 후에 NMI 키이를 누르면 모니터 상태로 돌아간다.
6. 'g' : 이 명령을 보내면 사용자 프로그램으로 뛸 준비

가 된다. 이후 NMI키이를 누르면 두 개의 모듈이 동시에 동작하게 된다.

4.5. 부동 소숫점 연산의 실현

병렬컴퓨터가 의미있는 동작을 하려면 부동 소숫점(floating point) 연산 기능이 완벽해야 한다. 따라서, 이 연구에서 부동소숫점 루우틴의 이식은 중요한 과제였다. 그러나 방대한 부동 소숫점 연산 루우틴을 모두 새로 작성한다는 것은 무리였으므로 IBM PC용으로 발표된 기존 부프로그램들[12]을 하드웨어 의존성을 고려하여 병렬컴퓨터에 이식하는 접근 방식을 택했다. 이 부동 소숫점 숫자의 유효숫자는 7자리이고 $\pm 10^{-64} \sim 10^{64}$의 실수를 나타낼 수 있다.

4.6. 수행시간의 측정

병렬 알고리듬의 검증에는 무엇보다도 수행시간의 측정이 요구된다. 따라서, CPU와 독립적으로 작동하는 계수기를 병렬컴퓨터에 포함시켰다. 측정 단위는 1μs로 2^{48} μs 즉, 약 9년까지 CPU타임을 표시할 수 있도록 하였다.

부록의 프로그램과 같이 사용자 프로그램 수행 직전에 시동(start)부프로그램을 실행시켜 계수기의 클록 게이트를 열어준 다음 수행이 끝나고 중지(stop)부프로그램을 실행시켜 이를 닫는다. 시스템에 포함된 계수기는 감소계수기(down counter)이므로 보수(complement)를 취하면 수행 시간을 얻을 수 있다.

5. 결 론

이번 연구를 수행하면서 독립된 CPU를 두 개 이상 같이 동작시키는 것, 특히 버스 및 주변장치를 공유하게 하는 것은 매우 고도의 기술이라는 점을 체험할 수 있었다. 설계자 자신이 두 개의 모듈을 독립적으로는 완벽히 완성하고도 오랜 시간을 버스중재와 직렬입출력 장치의 공유 문제를 해결하는데 소모했다. 그러나 병렬컴퓨터를 설계하고 아무데서도 원인을 찾을 수 없을 것 같은 많은 문제에 봉착하고 또 이를 극복해 나가며 자신의 문제 해결력을 크게 키울 수 있었다.

이번 연구의 가장 큰 의의는 범용 프로세서를 사용한 국내 최초의 병렬컴퓨터를 구현했다는 점에서 찾을 수 있다. 최근에 Intel사, TI사 등에서 어레이 프로세싱 전용의 IC들을 발표하고 있다. 그러나 이에는 개발환경이 거의 없을뿐더러 8086어셈블리어로 짜여진 많은 우수한 응용프로그램을 대치할 만한 것이 전무한 상태이다.

최근의 학술지에는 64000개의 프로세서를 연결한 병렬컴퓨터도 실험적으로 설계·제작되었음이 발표되었다. 그러나, 이번 연구와 같이 마이크로프로세서와 TTL (transistor- transistor logic)수준의 설계에 의한 노우하우 축적이 없이는 많은 수의 프로세서를 사용하는 병렬컴퓨터의 구현은 불가능한 일이고, 더 나아가 뉴럴 네트웍(neural network)등의 VLSI화는 생각해 볼 수도 없다.

또, 여러 병렬처리 알고리듬이 개발되고 있으나 국내에서는 이를 검증할 만한 하드웨어가 거의 없기 때문에 이론적인 증명에만 의존하고 있다. 그러나, 이로써는 각 처리모듈간의 통신 중 발생할 수 있는 문제점들을 모두 잡아낼 수는 없다. 이와 같은 상황에서 완벽한 부동소숫점 연산 능력을 갖추고, 벡터·행렬의 계산도 병렬처리가 가능하며, 병렬 알고리듬의 실제 측정시간의 검증도 가능하게 된 이번 연구의 의의는 참으로 크다 하겠다.

끝으로, 연구 동기를 부여해 주시고 유·무형의 지원을 아끼지 않으신 서울대학교 제어계측공학과의 이 장규 교수님께 깊은 사의를 표하고, 박사학위 논문에 이 연구 결과를 포함해 주신 김형중 선배님, 프로그램 작성에 큰 도움을 주신 김 용준 선배께 감사드립니다.

참 고 문 헌

1) Hwang, K., et al., "Computer Architecture and Parallel Processing" Mc-Graw Hill, 1984
2) Rector, R., et al., "The 8086 Book", Mc-Graw Hill, 1980
3) Microsoft co., "Microsoft Macro Assembler Manual", Microsoft co., 1985
4) Intel co., "Component Databook", Intel, 1984
5) Intel co., "iAPX 86/88, 186/188 User's Manual", Intel.
6) Texas Instruments, "The Bipolar Digital Integrated Circuits Data Book", TI, 1985
7) Liu, Y-C., et al. "Microcomputer Systems : The 8086 / 88 Family", Prentice-Hall, 1984
8) Enslow, F.H., "Multiprocessors and Parallel Processing", Wiley, 1974

9) Bowen, B.A., "The logical Design of Multiple-microprocessor Systems", Prentice-Hall, 1980
10) Reed, D.A., "Multiprocessing Networks Massage-Based Parallel Processing", The MIT Press, 1987
11) Karp, A., "Programing for Parallelism", IEEE Computer, 1987
12) Morgan, C.L., "Bluebook of Assembly Routines for the IBM PC & XT", Waite