디지탈시스템과 마이크로 프로세서 설계(V)

金 明 恒*

요 약

Bit-slice 마이크로프로세서의 구조를 설명하고, bit-slice 시스템의 설계를 위해 필요한 마이크로 인스트럭션의 구성과 pipelining 기법에 관해 토의한다

1. 서 론

마이크로프로그램 (microprogram) 의 개념은 1951년 M. V. Wilkes 에 의해 처음 시도되었으며 1964년 IBM 360 시스템에 마이크로프로그램의 개념이 도입된 이래 널리 사용되기 시작했다. 이러한 microprogrammed 시스템은 하드웨어 (hardware) 의 복잡성 때문에 사용자가 컴퓨터의 응용을 위한 목적으로의 사용은 불가능하였으나, 오늘날 반도체 기술의 급속한 발전으로 bit slice 프로세서와 같은 집적회로 (IC) 가 개발됨에 따라 마이크로프로그램의 개념이 손쉽게 모든 분야에 도입될 수 있게 되었다.

마이크로프로그램을 이해하기 위해서는 먼저 컴퓨터의 구조와 동작원리를 이해하는 것이 편리하다. 컴퓨터의 하드웨어가 동작하기 위해서는 하드웨어의 각 부분에 많은 제어신호들이 가해져야 한다. 일반적으로 이러한 제어신호들은 수많은 hardwired logic circuit (gates, flip-flop 등) 을 통하여 만들어 지는것이 보통이다. 그러나 microprogrammed 시스템에서는 제어신호들이 hardware logic circuit 대신에 Read Only Memory (ROM) (보통 programmable ROM (PROM)) 에 의해 발생하도록 하는 것이다. 따라서 microprogrammed 시스템은 구성이 간단하고 이해하기 쉬우며 또한 machine 인스트럭션 (instruction) 의 변화에 대해 손쉽게 PROM만을 변경하여 사용할 수 있다는 장점과 함께 가격이나 복잡도의 감소를 기대할 수 있다.

Bit slice 프로세서 및 마이크로프로그램 sequencer 들을 이러한 microprogrammed 시스템에 사용할 수 있도록 만들어진 반도체 소자들이다. 즉 각IC 들을 마이크로프로그램의 기본 시스템 기능을 수행 할 수 있도록 구성되어 있으며 마이크로인스트럭션 (micro instruction) 에 따라 제어 신호를 발생하도록 설계되었다. 예를 들어 AMD (Advanced Micro Devices)사의 2900 family의 경우 AMD 2901과 2903은 4 bit 의 연산논리기능을 수행할 수 있으며 AMD 2909와2910은 micro-sequencer 의 기능을 수행할 수 있도록 설계된 IC 들이다. 따라서 이러한 IC 들을 연결하여 손쉽게 microprogrammed 시스템을 구성할 수 있다.

2. Bit Slice 시스템의 구조

Bit slice 프로세서는 단일 칩 (chip)MOS 마이크로 프로세서와 두가지 면에서 다르다. 첫째 CPU 구조 (architecture)에있어서 MOS 마이크로프로세서는 데이타처리기능과 제어기능 (즉 인스트럭션을 decoding 하는 기능) 이 같은 칩내에 하드웨어로 내장되어 있으나, bit slice 구조는 CPU의 두 기능을 서로 다른 분리된 칩에서 행하게 되어 있다. 둘째, 단일 칩 프로세서는 이미 결정되어 바꿀 수 없는 고정된 단어길이 (word length), 인스트럭션 set 및 하드웨어구조를 가지고 있으나, bit slice 시스템에서는 그와 반대로 사용자가 마음대로 단어길이, 인스트럭션 set 및 하드웨에 구조를 가지고 있으나, bit slice 시스템

* 正會員 : 美國Cornell 大 電氣工學科 敎授 · 工博

에서는 그와 반대로 사용자가 마음대로 단어길이, 인스트럭션 set 및 하드웨어 구조를 선택할 수 있다.

Microprogrammable bit slice 디지탈 소자들을 가장 잘 이용하기 위해서는 control store, sequencer, pipelining, but slice RALU (Registered Arithmetic Logic Unit), microprogramming (horizontal, encoded, Polyphas), 마이크로인스트럭션field 등과 같은 용어와 개념들에 대한 명확한 이해가 필요하다. 또한 firmware 개발을 위해 사용할 수 있는 시스템 support 소프트웨어가 현재까지 상당히 부족하기 때문에 설계자는 bit slice 시스템의 구조와 timing 요구조건에 대해 상세하게 알아야만 한다.

대부분의 microprogrammable bit slice 디지탈 시스템을 그림 1과 같은 기본적인 블럭도로 표시된다. MOS 형 단일칩 CPU와는 달리, microprogrammable CPU는 bipolar Schottky technology를 사용한다. 내부 기능의 복잡성과 bipolar technology 로 현재까지가능한 칩의 복잡도, 핀 (pin) 수, 칩의크기 (size) 가 제한되어 있기 때문에 microprogrammable CPU는 여러개의 칩으로 구현되어야 한다. 따라서 microprogrammable CPU의 제어부 (control section) 와 처리부 (processing section)가 서로 다른 칩에 내장되어 있으며, MOS 형 CPU의 제어부 및 처리부와는상당히 다르게 구성되어 있다.

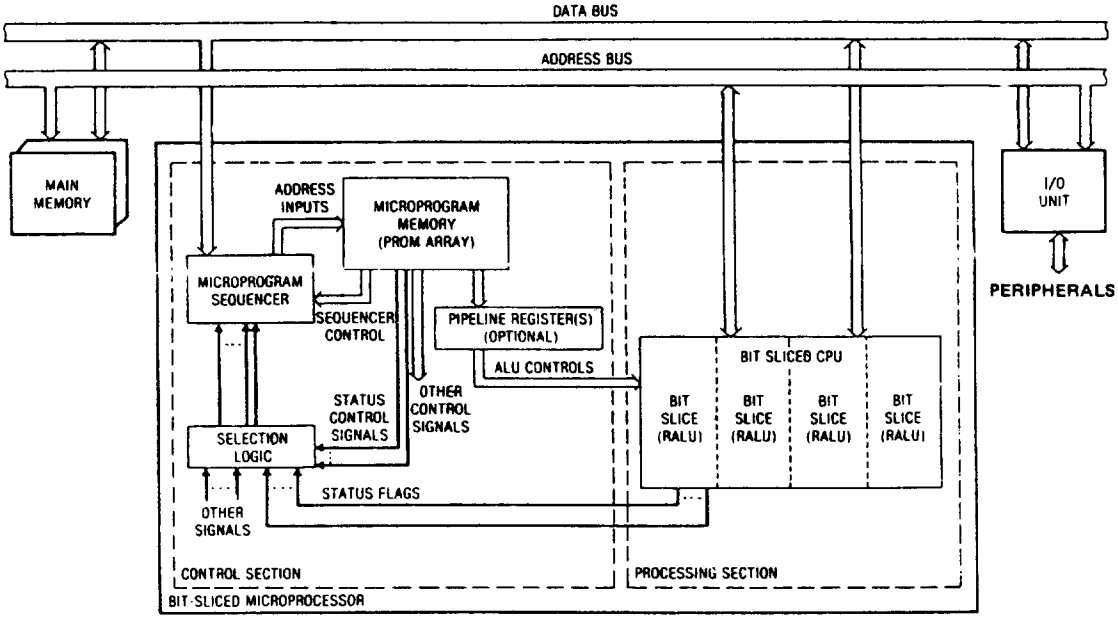


그림 1. A microprogrammable bit-sliced microcomputer

• 제어부 (Control Section)

제어부는 처리부보다 더욱 복잡한 설계가 필요하다. 이부분은 전체 bit slice 시스템에 고유의 인스트럭션 set 을 제공하여 준다.

전형적인 제어부는 마이크로프로그램 메모리 (마이크로프로그램 control memory 또는 control store 라고도 함), 마이크로프로그램 sequencer (마이크로프로그램 control unit 또는 controller라고도 함) 및 selection logic과 pipeline register 등과 같은 기타 주변회로들로 이루어 진다. 이러한 제어부의 구성요소들은 보통 각기 다른 LSI 칩들로 만들어져 있다.

마이크로프로그램 메모리는 bit slice 시스템의 제어순서를 규정하고 처리부내에 있는 RALU (Registered Arithmetic Logic Unit) 의 동작을 제어하는 마이크로인스트럭션을 저장하고 있다. 마이크로프로그램 sequencer 는 마크로인스트럭션 (macroinstruction) decode logic 을 제공하고 마이크로프로그램의 순서를 제어하기 위해 "next microaddress"를 발생시키는 방법을 결정하여 마이크로프로그램 메모리에 다음에 다음에 수행될 마이크로인스트럭션의 주소를 공급한다. 마이크로프로그램 메모리는 ROM이나 PROM으로 구성되며 여러개의 유사한PROM을 접속시킴으로써 내장되는 마이크로프로그램인스트럭션의 수효

나 마이크로인스트럭션의 단어길이를 필요에 따라 확장시킬 수 있다.

• 처리부 (Processing Section)

이 부분에서 모든 연사기능과 논리기능이 수행된다 이 부분은 bit slice 혹은 RALU라고 불리는 여러개의 칩들에 의해 구성된다. 각각의 RALU는 여러개의 범용 혹은 특수용도의 register, accamlator, ALU (Arithmetic Logic Unit) 및 관계되는 status flag 등을 가지고 있다. 오늘날, 대부분의 bipolar microprogrammable 마이크로프로세서들은 2 bit 혹은 4 bit slice 들은 사용하여 구성되고 있다. 이러한 slice들은 여러개 사용하여 접속시킴으로써 기본 slice bit 수의 배수가 되는 어떠한 width의 마이크로프로세서도 만들 수 있다. 따라서 일반적으로 사용하지 않는 단어길이들(24, 32, 48bit 등)도 취급할 수 있으며, 높은 처리효율(throughput rate)을 얻을 수 있다.

3. 제어부의 구조와 동작

Microprogrammed 시스템의 유형은 크게 emulator 형과 controller 형으로 나뉜다. 그림 1의 마이크로 컴퓨터는 emulator 형으로서, bit sliced 마이크로프로세서는 내부에 있는 마이크로프로그램 메모리에서 읽은 마이크로인스트럭션의 지시에 따라 외부에 있는 시스템의 주 (main)메모리에서 마크로인스트럭션(macroinstruction)을 fetch 한다. fetch 된 마크로인스트럭션의 operation code 는 마이크로프로그램 sequencer 에 의해 해석 (즉 마이크로프로그램 메모리 주소로 mapping)되어 일련의 마이크로인스트럭션들에 의해 수행된다.

마크로인스트럭션의 operand code 는 처리부의 RALU에 들어가서 계산에 사용되거나 외부에 있는 주메모리의 주소를 결정하기 위해 사용되어 진다. 따라서 시스템의 주 메모리에는 마크로프로그램 (응용프로그램) 을 저장하고, 제어부의 마이크로프로그램메모리는 구현되는 마이크로 프로세서를 정의하는 마이크로프로그램을 포함하고 있다.

Emulator 형의 microprogrogrammed 디지탈 시스템에는 적어도 두가지 수준 (macro level 과 micro level) 의 프로그래밍과 제어가 필요하게 되며, 설계자는 마크로인스트럭션 set 과 이를 구현하기 위한 마이크로인스트럭션 set 을 정의하고 결정해야 한다. 이에 반하여 controller 형의 microprogrammed 시스템에서는 마이크로 수준의 프로그램과 제어방식만 존재하며, 따라서 모든 프로그램은 마이크로프로그램으로 내부에 있는 마이크로프로그램 메모리 저장되어 있다. 일반적으로 emulator 형의 microprogrammed 시스템은 기존 컴퓨터의 인스트럭션 set 을 emulation 하거나 새로운 컴퓨터 시스템을 설계하고자 할 때 사용되며 bus I/O 형의 구조를 가지고 있다. 또한 controller 형의 microgrammed 시스템은 disk controller, machine control 등과 같이 외부 하드웨어 (hardware) 를 직접 제어하고자 할 때 사용되며 direct I/O형의 구조를 가지고 있다. Emulator 형과 controller 형의 성능을 비교하면 그림 2와 같다. (여기서 화살표 방향은 변수가 증가하는 방향이다)

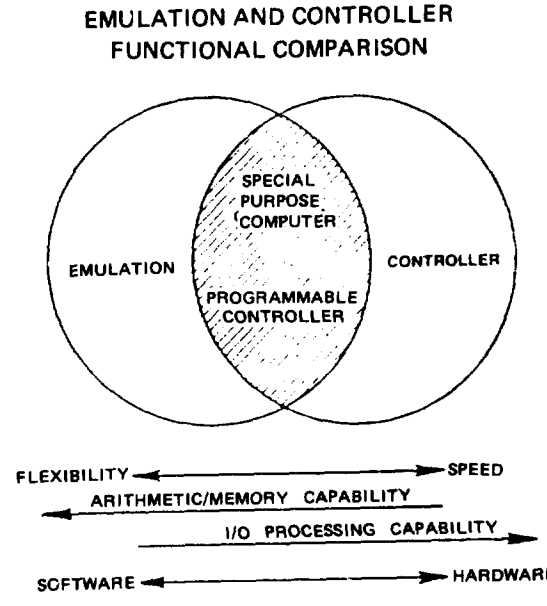


그림 2. Emulator 형과 controller 형의 성능비교

제어부는 크게 두 부분으로 나눌 수 있는데 하나는 마이크로프로그램을 순서적으로 수행시키기 위해 다음 마이크로인스트럭션의 주소를 발생시키는 마이크로프로그램 sequencer 이다. 마이크로프로그램 sequencer 의 기능은 counter, FPLA (Field-Programmble Logic Array) 등과 같은 상용의 MSI 소자들을 이용하여 구현하거나, 상용의 LSI sequencer 를 직접 사용하여 쉽게 구현할 수 있다. 여기서는 상용 LSI sequencer 의 구조에 관해서만 논하고자 한다. 전형적인 상용 sequencer 의 블럭도는 그림 3에 보여져 있다. 그림에서 보는 바와 같이 sequencer 는 내부에 next address logic, 마이크로프로그램주소 register/counter, stack 등을 갖고 있다. 마이크로프로그램 sequ-

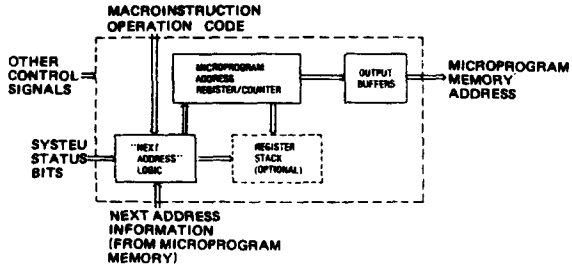


그림 3. General block diagram of the LSI microprogram sequencer

encer 의 주요 기능은 마이크로인스트럭션이 fetch 되고 수행되도록 마이크로프로그램 메모리에 주소를 공급하는 것이다. sequencer 의 next address logic 부분은 특정 address source 를 결정하여 이 address source 의 데이타를 마이크로프로그램 주소 register/counter 에 넣는 역할을 한다. Address source 의 선택은 sequencer 가 받은 다음 주소정보에 관한 bit 들에 의해 이루어지며 이 bit 들은 일반적으로 다음 기능중 하나를 지정할 수 있다. increment, conditional skip next instruction, conditional branch to a microsubroutine, push/pop the register stack 등 대부분의 LSI sequencer 들은 칩 내부에 last-in firstout 방식으로 동작하는 register stack (대개 4 level deep)을 가지고 있다. 이 stack 은 push 와 pop 형 명령에 의해 마이크로프로그램 looping 이나 microsubroutine branching동안 일시적으로 주소를 기억하는데 사용된다. 끝으로 주소출력을 위한 latch 가 마이크로프로그램 sequencer 와 마이크로프로그램 메모리 간의 interface 를 위해 sequencer 내부에 들어 있다.

상용 LSI 마이크로프로그램 sequencer 들은 크게 두 분류로 나눌 수 있다. 하나는 고정된 숫자의 주소까지만 공급할 수 있는 fixed-address sequencer 이고 또 하나는 여러개를 접속 시켜서 원하는 정도의 addressing 능력을 가질 수 있는 bit-slice sequencer 이다. 첫번째 부류에 속하는 것은 Intel 3001 (9bit 주소, 그림 4 (a)), Monolithic Memories 67110 (9bit 주소), Signetics 8x02 (10bit주소, 그림 4 (b)), Fairchil 9408 (10bit 주소), AMD 2910 (12bit 주소) 등이 있다.

두번째 부류인 bit-slice 마이크로프로그램 sequencer 는 보통 4 bit단위로 addressing 이 가능하다. 대표적인예로 AMD 2909, 2911 (그림 4 (c)), 2930/2931 Texas Instrument 74S 482등이 있다. 표 1은 상용 LSI sequencer 들의 목록을 보여주고 있다.

표 1. Commercially available LSI microprogram sequencers

- AMD 2909/2911.2910.2930/2931 (second-sourced by Raytheon, Signetics)
- INTEL 3001 MCU (second-sourced by Signetics)
- SIGNETICS 8X02, 3001 Control Store Sequencer
- MONOLITHIC MEMORIES 6710,67110 Controller
- TEXAS INSTRUMENTS 84S482
- FAIRCHILD 9408 Program Controller
- MOTOROLA 10801 Microprogram Control Device
- NATIONAL CROM

4. 처리부의 구조와 동작

처리부의 가장 중요한 부분은 모든 연산과 논리기능을 수행하는 ALU 이다. 상용의 74181 4-bit 병렬 가산기를 여러개 접속시켜 사용함으로써 원하는 길이의 연산 및 논리기능을 수행하는 modular ALU 를 구성할 수 있다는 사실은 잘 알려져 있다. 물론 ripple-carry 동작이나 74182 소자를 사용한 carry-look ahead 동작을 위해서는 ALU 로 부터 몇 개의 외부 신호가 나와야 한다. 한편 74281 ALU 는 내부에 accumulator 와 shift matrix 를 가지고 있어서 74181 ALU 보다 진보된 기능을 갖는다. 현재의 수직 분할된 bit-sliced CPU 는 표준 RALU 를 접속시킨 것인데, 이 RALU 는 더욱 진보되어 승산 (multiplication) 과 같은 복잡한 앨고리즘을 수행할 때 생기는 임시적인 데이타를 저장하기에 편리한 여러개의 내부 register 를 갖고 있고, 또한 double shift 나 승산시에 일어나는 double length 의 결과를 저장하기 위한 extension register 도 갖고 있다. 상용의 RALU 는 overflow, zero, sign 등과 같은 내부 상태를 검출할 수 있으며, ALU 와 register transfer 를 제어하기 위해 필요한 decoder logic 도 포함하고 있다.

Bit-slice 구조의 특이한 점은 보통의 functional line 을 따른 수평 분할 방식을 채택하지 않고, CPU 의 처리부를 수직분할 했다는 점이다. 이러한 수직분할 방식에 의하여 그림 5 (a) 과 같이 register 와 ALU 는 같을 길이의 서로 같은 기능을 갖는 RALU(혹은 bit-slice) 라고 불리는 부분들로 나뉘어 진다. 이러한 RALU 는 2 bit 혹은 4 bit 데이타를 처리할 수 있으며, 이들을 서로 접속 시킴으로써 더 긴 단어 길이의 데이타를 처리하는 처리부를 구성할 수 있다.

일반적으로 전형적인 RALU 는 ALU, 하나 혹은 두개의 post 를 갖는 multiple (1개에서 16개) word register file, shifter 마이크로인스트럭션 decoding 하

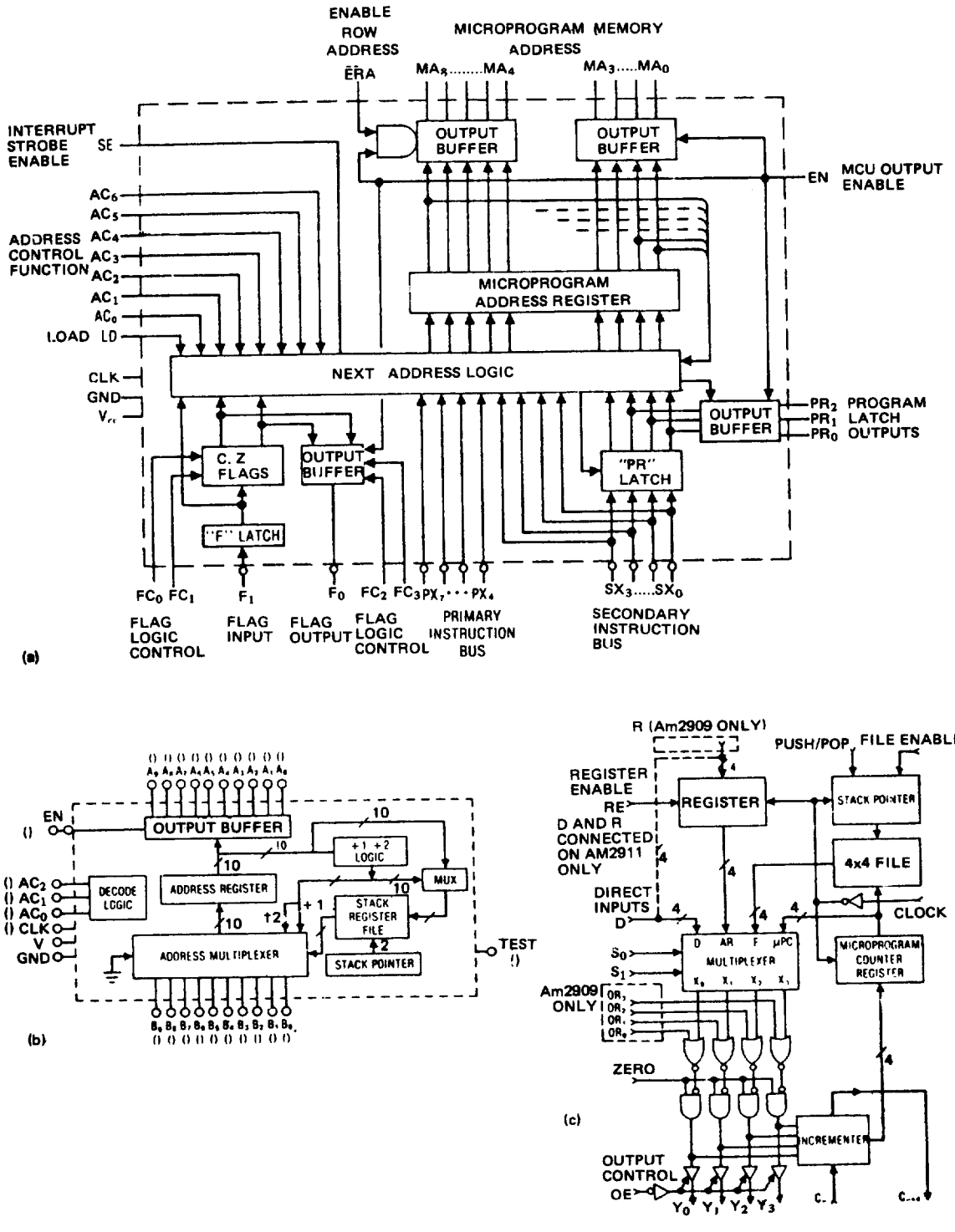


그림 4. Examples of commercial LSI microprogram sequencers : (a) Intel 3001, (b) Signetics 8×02, and (c) Am 2909/2911

기 위한 decoder 등으로 구성되며, 그림 5 (b)에 보여져 있다. bit slice 에 의해 수행되는 모든 논리 및 연산기능은 마이크로프로그램 sequencer 에 의해 지정된 주소에 따라 마이크로프로그램 메모리내의 마이크로인스트럭션이 fetch 되어 bit slice 에 보내진 제어신호에 따라 일어난다. 그림 6은 2 - bit Intel 3002와 4 - bit Am 2901 bit- slice 마이크로프로세서의 블럭도이다. 또한 그림 7은 이 두 slice 를 사용한 microprogrammable bit- slice 컴퓨터 시스템의 블럭도를 보여주고 있다. 표 2에는 현재 시판되로 있는 상용 bit slice 마이크로프로세서의 목록이 수록되어 있다.

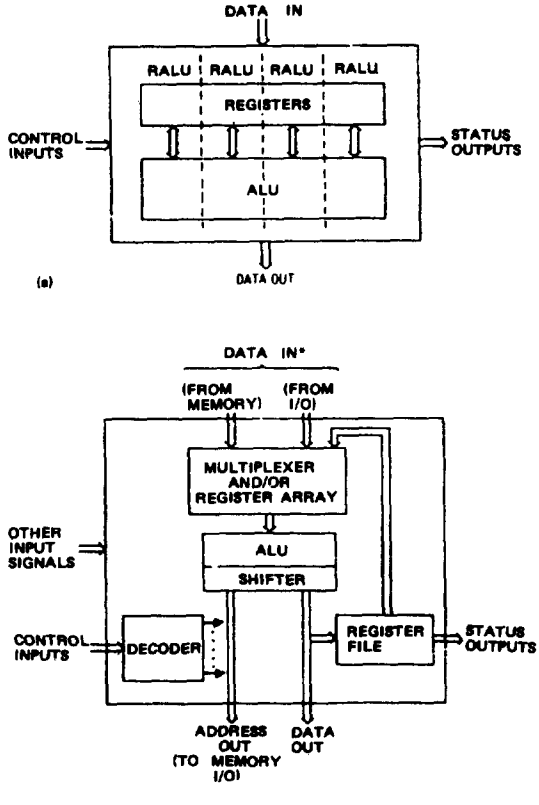
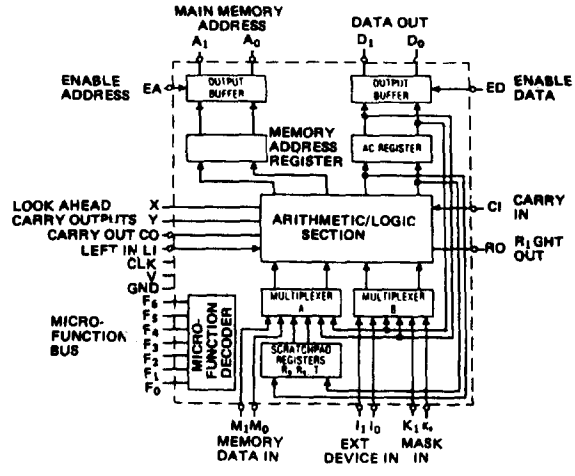


그림 5. (a) A vertically partitioned processing section defining the RALUs and (b) the general block diagram of a RALU.

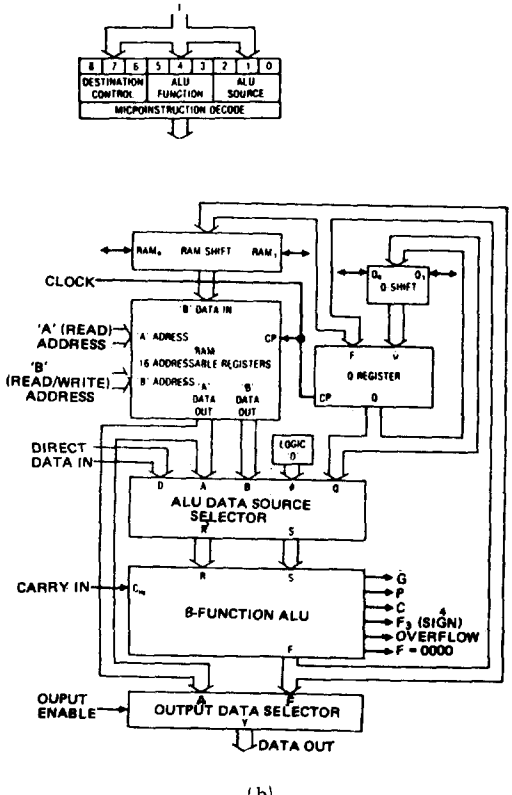
5. 마이크로인스트럭션 설계

5.1 마이크로인스트럭션 format 과 coding

Bit slice 시스템의 마이크로프로그램 메모리는

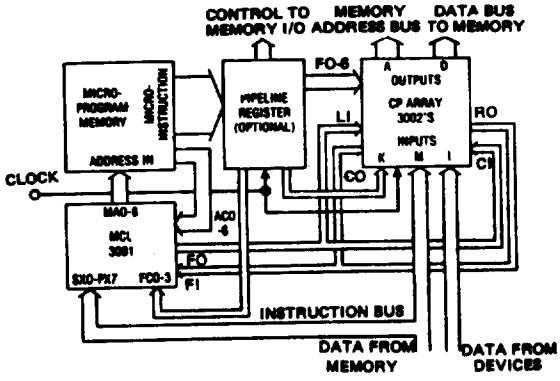


(a)

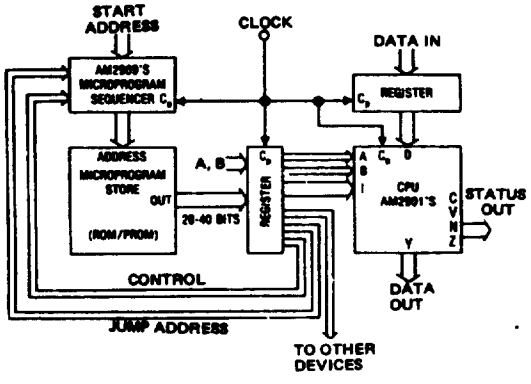


(b)

그림 6. The block diagrams of two commercially available bit-sliced microprocessors : (a) the Intel 3002 2-bit RALU and (b) the Am 2901 4-bit RALU



(a)



(b)

그림 7. Two commercial microprogrammabile bit-sliced microcomputers : (a) the Intel 3000 series and (b) the AMD 2900 serie

표 2. Commercially available LSI bipolar bit-sliced microprocessors

- AMD 2901,2903 (second-sourced by Raytheon, MMI, Signetics) (4-bit slice)
- INTEL 3002 (second-sourced by Signetics) (2-bit slice)
- SIGNETICS 2901-1 (4-bit slice), 3002 (2-bit slice)
- MONOLITHIC MEMORIES 6701 (4-bit slice)
- TEXAS INSTRUMENTS 74S481, SBP-0400 (4-bit slice)
- FAIRCHILD 9405A (4-bit slice)
- MOTOROLA 10800 (4-bit slice)
- RCA 4057 (4-bit slice)
- RAYTHEON 2901/2903, RF 5 (4-bit slice)
- NATIONAL GPC/P (4-bit slic

ROM, PROM, PLA등을 사용, ! 구현된다. 그러나 설계자는 마이크로인스트럭션을 구성할때 매우 다양한 format 과coding 방식 중에서 하나를 선택할 수있다. 마이크로인스트럭션은 보통 서로 다른 기능을 갖는 두개의 부분으로 나눌 수 있다. 그하나는 수행될 microoperation 을 정의하고 제어하는 제어신호 pattern 에 해당하는 bit 들이고 다른 하나는 수행될 다음 인스트럭션의 주소를 제어하고 지정하는 bit 들이다.

일반적으로 마이크로인스트럭션의 format, 단어길이, timing 방법등에는 다양한 여러가지 방법들이 있다. 하지만 bit-slice 시스템에서는 상용RALU (처리부의 slice) 와 상용 마이크로프로그램 sequencer 등을 사용하기 때문에 그 선택폭은 좁아진다. formatting 이란 마이크로인스트럭션의 format 은 고정형식과 가변형식으로 나뉜다. 고정형식에서는 마이크로인스트럭션 대의 각 bit 은 항상 같은 방식으로 해석되나, 가변형식에서는 field 나 bit 의 의미가 특정한 시스템의 상태에 따라 변화한다. (그림 8) 가변 형식을 사용하면 마이크로인스트럭션의 단어길이를 짧게 할 수 있으나 format control 을 위해 부가적인 logic 이 필요하고, 지연시간이 걸리게 된다.

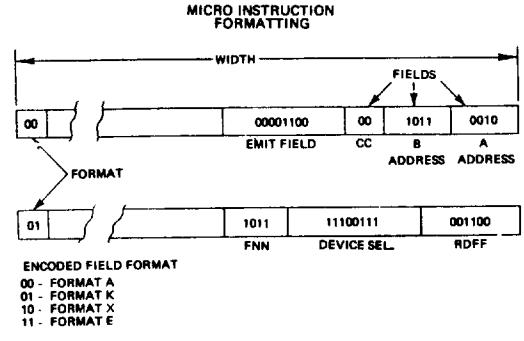


그림 8. 마이크로인스트럭션 가변형식의 예

만일 마이크로인스트럭션의 각 bit 이 제각기 시스템 내의 서로다른 control line 을 제어하기 위하여 사용된다면, 이경우 마이크로인스트럭션이 non-encoded (혹은 unpacked) 형태를 갖는다고 한다. (그림 9 (a)). 이 format 을 사용하면 마이크로인스트럭션의 단어길이가 길어지면 수행속도는 빠르고 동시수행능력 (concurrency)이 크게 된다. 물론 마이크로인스트럭션의 단어길이가 길어지므로 마이크로프로그램메모리의 width 도 커지게 된다. 이와는 다른 방법으로 상호 배타적인 제어 line 들을 모아 하나의 group 을 형성시키고 이 group 을 마이크로인스트럭션내에 있는 하나의 encoded control field 에 의해 제어하는 방식이 있다. 예를들어 2^n개의 상호배타적인 제어 line 들이 있을 경우 마이크로인스트럭션내에 n-bit field 가 필요하며 이 n-bit 은 decoder 에 의해서 마이크로인스트럭션이 수행될때 2^n개의 제어신호를발생시

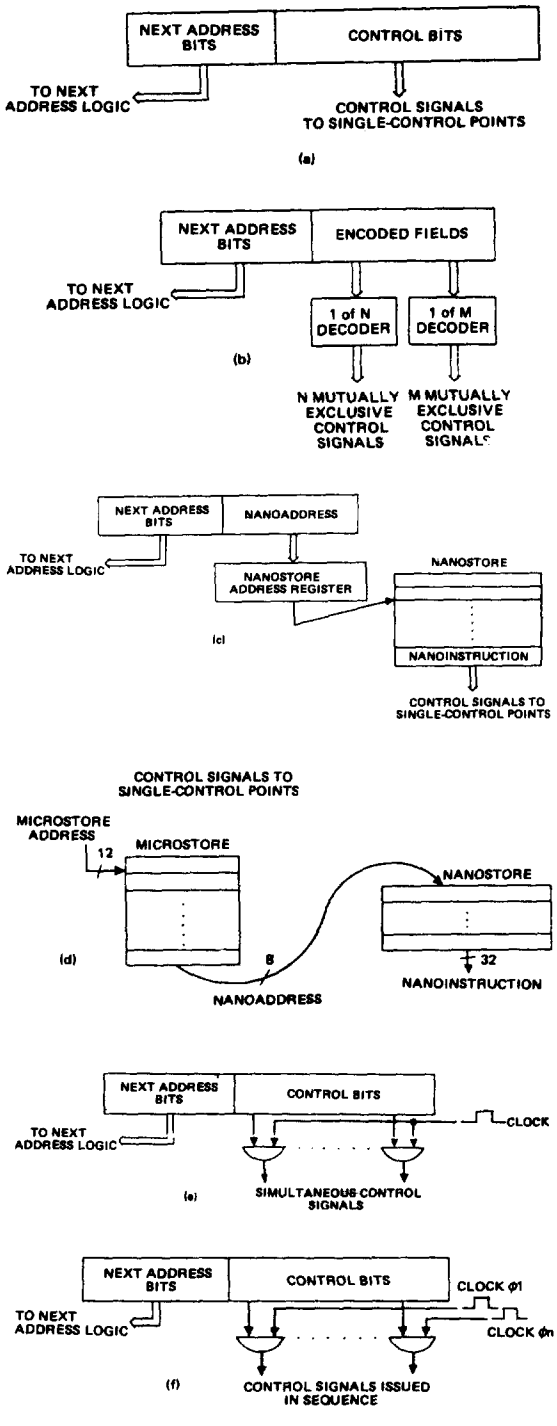


그림 9. Microinstruction formats : (a) non-encoded, (b) encoded, (c) nanoprogrammed, (d) two-level control memory, (e) monophase, and (f) polyphase

킨다. 이러한 마이크로인스트럭션 encoded (혹은 packed 형태를 갖는다고 한다. (그림 9 (b)).

어떤 컴퓨터들은 그림 9 (c)에 보는 바와 같이 제한된 수의 control field를가지고 더 낮은수준의 메모리인 nanoinstruction 을 내장하고 있는 nanostore 를 addressing 하는 nanoprogramed 구조를 가지고 있다. 그림 9 (d)는 많은 수의 horizontal 마이크로인스트럭션이 사용된 두 가지 수준의 메모리를 가진 설계방식을 보여주고 있다. microstore (마이크로프로그램 메모리) 는 짧은 단어길이 (8 - bit) 를 가진 많은 수 (1 K) 의 마이크로인스트럭션을 수록하고 있으며, 반면에 nanostore 는 긴 단어길이 (32-bit) 를 가지고 있으나 적은수 (256) 의 nanoinstruction 을내장하고 있다. 따라서 이 방식을 사용하면 32-bit 단위길이의 4 K word 마이크로프로그램을 단지 4 K words × 32 bits 메모리와 256 word × 32 bitd 메모리 만으로 내장할 수 있다. FPLA를 사용하면 이러한 two-level microstore 방식을 쉽게 구현할 수 있다. 여기서 nanoinstruction 들은 마이크로프로그램에 있어서 "subroutine" 들에 해당한다. 마이크로프로그래밍 방식을 사용한 두 가지 대표적인 예는 nanodata OM-1 과 관계되는 두 가지 용어는 manophase 와 polyphase 마이크로프로그래밍이다. Monophase 마이크로프로그래밍은 그림 9 (e) 와 같이 각각의 마이크로인스트럭션이 단 하나의 clock cycle만 사용하는 손쉬운 방식이다. polyphase 마이크로프래밍은 하나의 마이크로인스럭션을 수행하는데 여러개의 clock cycle이 필요하며 그림 9 (f) 와 같이 마이크로인스트럭션에 의해 발생되는 제어신호들은 여러 clock cycle 동안 순서적으로 전체시스템에 공급된다.

5.2 마이크로인스트럭션

Microprogrammed 시스템에서 가장 유용하고 획기적인 설계개념은 마이크로인스트럭션 pipelining 방식이다. pipelining 이란 보통 약간의 부수적인 소자를 시스템에 삽입하여서 전체 시스템의 cycle시간을 약반 정도로 줄일 수 있는 기법이다. 즉 현재의 인스트럭션이 수행되고 있는 동안 동시에 다음에 수행될 인스트럭션의 주소를 지정해서 fetch 함으로써 제어부 (control section) 와 처리부 (processing section) 가 병렬로 동시에 동작하도록 하는 것이다.

• Nonpipelined design

그림 10은 pipelining 방식을 사용하지 않은 간단한 블럭도이다. 이 경우 control storage에서 나온 현재의 마이크로인스트럭션이 직접 제어부와 처리부의 적

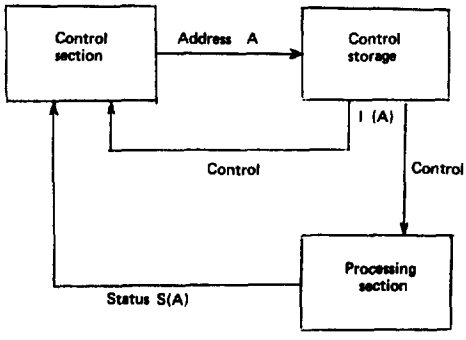


그림 10. Nonpipelined design

당한 지점에 공급되고 있다. 또한 처리부에서 나온 status 출력도 다음 인스트럭션의 주소를 결정하기 위해 제어부에 의해 직접 사용되고 있다. 따라서 전체시스템은 모두 직렬로 연결되어 있고, 그에 따른 timing diagram 은 그림 11과 같다.

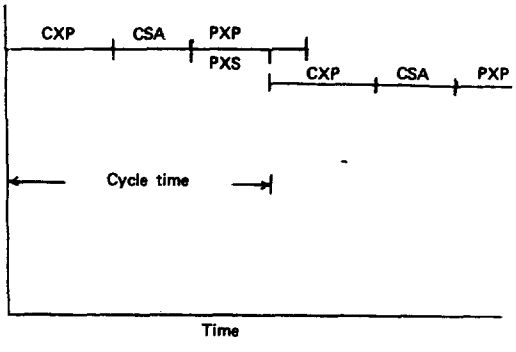


그림 11. System timing for nonpipelined design

여기서

CXP : 제어부의 전달지연 시간

CSA : Control storage 의 acess 시간

PXP : 처리부의 전달지연 시간

PXS : Status 출력이 나올때 까지 소요되는 처리부의 전달지연 시간

이라고 가정할 때, 최소의 machine cycle 시간은 CXP +CSA +PXS 로 결정된다.

c Pipelined design (microinstruction/status pipelining)

가장 보편적으로 사용되는 pipelining 방식은 그림 10에 현재의 마이크로인스트럭션을 저장하는 pipline (혹은 마이크로인스트럭션) register 와 처리부에서 나오는 status 출력을 저장하는 status register 를

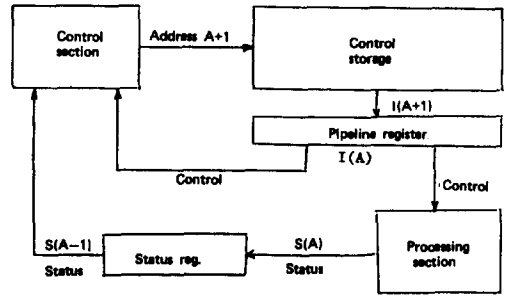


그림 12. Pipelined design (microinstruction/status pipeling)

첨가하여 만드는 것이다. 이 방식을 마이크로인스트럭션 / status pieplining 이라고 하며 그림 12에 보여져 있다. 이 pipelining 개념을 사용하여 시스템을 설계하면 전체 시스템을 크게 두 경로 (path) 로 나눌 수 있다. 첫째 경로는 제어부에서 출발하여 control storage 를 지나 pipeline register 에 이르는 제어경로이다. 두번째 경로는 데이타 경로 로서 pipeline register 에서 처리부를 지나 status register 까지를 말한다. 이 두 경로는 동시에 수행되므로, machine cycle 시간은 두 경로중 가장 시간이 많이 걸리는 쪽의 수행시간과 같게 된다. 그림 13은 제어경로쪽이 데이타경

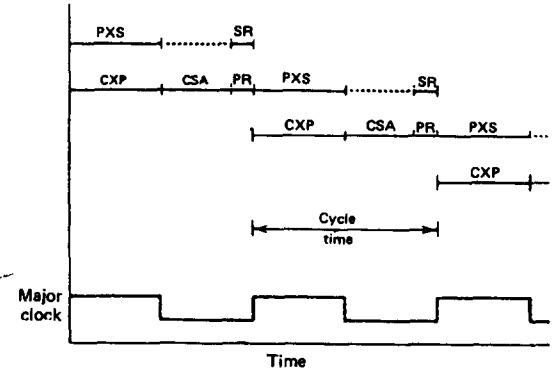


그림 13. Pipelined design timing (control-section time exceede processing-section time)

로쪽보다 시간이 더걸리는 경우의 시스템 timing 을 보여주고 있다. 여기서 cycle 시간은 CXP +CSA +PR 과 같다. 그림 14는 데이타 경로 쪽이 제어경로 쪽보다 시간이 더 걸리는 경우의 시스템 timing 이다. 여기서 cycle 시간은 PXP 가 PXS +SR 보다 짧은 경우 PXP +SR 로 결정된다. 예를 들어 다음과 같은 시간을 가정하자.

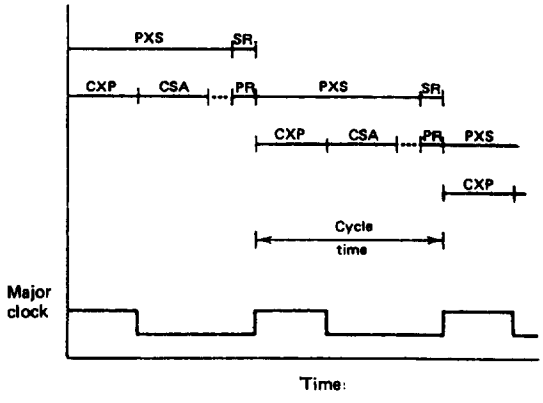


그림 14. Pipelined design timing (processing-section time exceeds control-section time)

CXP : 50ns, CSA : 65ns, PXS : 90ns
PR (set-up time for pipeline register) = 15ns
SR (set-up time for status register) = 15ns

이 경우 CXP +CSA +PR = 130 ns 이고 PXR +SR = 105ns 이다. 따라서 machine cycle 시간은 130ns 가 될 수 있다. 만약 non-pipelined design 이 사용되었다면, machine cycle 시간은 CXP +CSA +PXS 로서 205ns 가 될 것이다. pipelining 이라는 상당히 간단하고 값싼 방법을 사용한 결과 machine cycle 시간이 205ns 에서 130ns 로 줄어 들었음을 알 수 있다.

pipelining 방법을 사용할 할 경우 그 단점으로는 마이크로프로그램이 복잡해 진다는 점을 들 수 있다. 첫째, programmer 는conditional branch 인스트럭션을 사용할 경우 조심해야 한다. 즉 status register 에 기억되어 사용되는 condition 들은 현재의 machine cycle 에서 발생한 것이 아니라 이전 machine cycle 에서 발생한 것들이므로 이 점을 명심해야 한다. (그림 15). 둘째로 pipelined 시스템에서 가끔 발생하는 결점은 programmer가 때때로 한 machine cycle 을 버려야 하는 점이다. 예를 들어 R_2 와 R_3 register 에 들어 있는 값중 어느것이 더 큰지 알기 위하여 condition 을 test 하고, 만족할 경우 branch 하는 프로그램 그림 16 (a) 와 같이 작성하였다. 이 그림에서 한 블럭은 마이크로인스트럭션 하나를 나타낸다. pipelined 시스템에서 comparison 과 conditional branch 는 두개의 마이크로인스트럭션을 필요로 한다. 이때 branch operation 만 있는 인스트럭션에 다음 cycle의 낭비를 막을 수 있다.

이와 같은 예가 그림 16 (b) 에 보여져 있다.

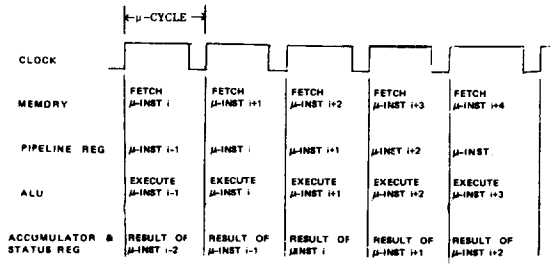


그림 15. Timing Diagram of Microprogram Execution

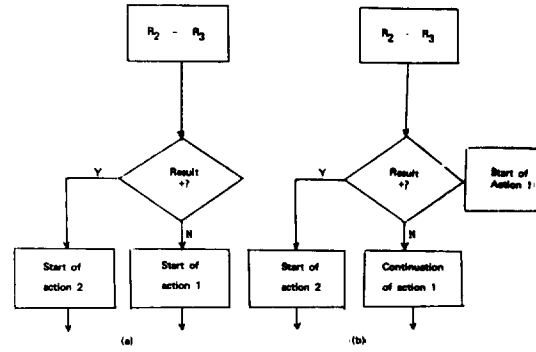


그림 16. Avoiding branch-only cycles in apipelined design

5.3 다른 유형의 Pipelining

앞 절에서는 pipelining 을 위하여 status register와 pipeline register 를 사용하였다. 다음에는 이 방식과는 다른 유형의 pipelining 방법들에 대해 소개하고자 한다.

- Microinstruction/address pipelining

그림 17 과 같이 pipelining 을 위하여 address register 와 pipeline register 를 삽입하였다. 이때 machine cycle 시간은 PXS +CXP +ARI 과 AR2 +CSA

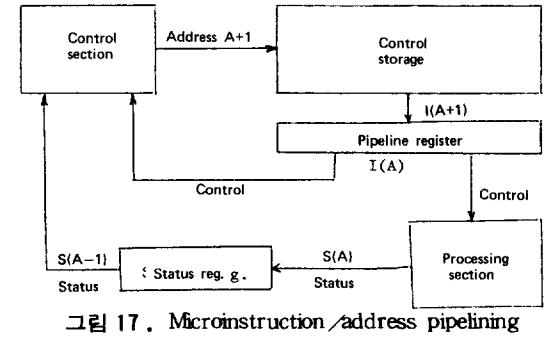


그림 17. Microinstruction/address pipelining

+PR 중 더 큰쪽에 의해서 정해진다. 여기서 다음과 같은 변수들이 더 사용되었다.

AR 1 : set - up time of address register

AR 2 : propagation delay from the output - enable input of address register

만약 AR 1= 5ns, AR 2=15ns 이고 다른 변수들은 5.2 절에서 사용한 값과 같다면 cycle 시간은 145ns 와 95ns 중 큰쪽이므로 145ns 로 결정된다. 따라서 이러한 경우 microinstruction/status pipeline 방법보다 좋지 않다. 하지만 control storage 의 access 시간이 상당히 큰 경우, 예를 들어 CSA=100ns 일 경우는 cycle 시간이 145ns이고, 이 경우 5.2 절에 있는 pipeline 방법을 사용하면 cycle시간이 165ns 이므로 현재의 방법이 더 좋음을 알 수 있다.

– Address /status pepelining (그림 18)

Machine cycle 시간은 AR 2 +CSA +PXS +SR 과 AR 2 +CSA +CXP +AR 1 중에서 더 큰 쪽에 의해 결

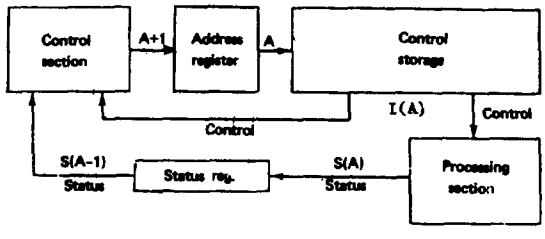


그림 18 . Address /status pipelining

정된다. 이 경우 control - storage access 시간이 두 경로 (제어경로와 데이타 경로) 에 모두 들어가게 된다. 앞서 사용한 표본 값들을 이용하면 최소 cycle 시간은 185ns와 135ns 중 큰 것에 의해 결정되므로 185ns 이다.

일반적으로 이 방법은 5.2 절에서 언급한 첫번째 방법보다 항상 속도가 느리다. 이 방법의 유일한 장점은 address register 가 마이크로인스트럭션(pipeline) register 보다 보통 작기 때문에 약간 가격이 싸다는 점이다.

Microinstruction /address /status pipelining (그림19)

이 방법은 앞서 연구한 모든 방법을 조합하여 이루어졌다고 생각할 수 있다. 즉 마이크로인스트럭션 register, address register, status register 를 사용하여 세 개의 병렬경로를 만들어 준다. 따라서 이 방법을 double pipelining (혹은 two level pipelining) 이라고부른다. 최소 machine cycle 시간은 PXS +SR, CXP +AR

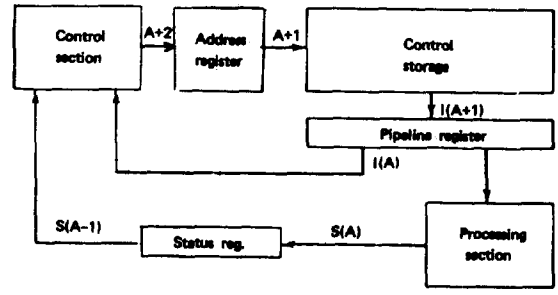


그림 19 . Microinstruction /address /status pipelining

1 과 AR 2 +CSA +PR의 세가지 값중 가장 큰 값에 의해 정해진다. 앞서 언급한 표본 값을 각 변수에 대입하면 cycle 시간은 105ns, 95ns 중에서 가장 큰 값인 105ns 로 결정된다. 따라서 이러한 유형의 pipeline방식은 처리부, 제어부 및 control storage 의 속도를 분리 시킴으로써 빠른 속도의 시스템을 제공해준다. 그러나 다른 방법보다 많은 logic 이 필요하게 되어 시스템 cost 가 높아진다. 더우기 이러한 시스템에서 마이크로프로그램을 구현하는 작업은 branching operation 이 일어나는 cycle 주변에서의 혼란때문에 매우 어렵게 된다. 따라서 이 방법은 거의 사용하지 않으며 여러가지 pipeline 방식중에서 microinstuction / status pipeline 방식이 대부분의 시스템설계에 가장 좋은 것으로 생각된다.

5.4 Microorder encoding

마이크로인스트럭션에서 encoding 을 많이하면 할수록 control storage age 의 크기를 줄일 수 있으나 decoding logic 이 복잡하여 가격이 높아지고 decoding logic 에 의한 부수적인 지연 (delay) 때문에 cycle

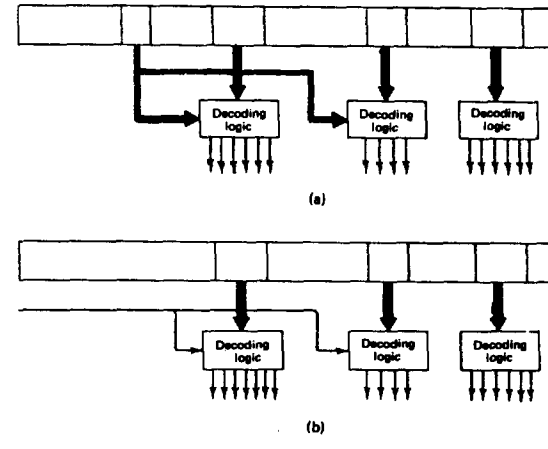


그림 20 . Two forms of indirect encoding

시간이 느려질 우려가 있다. encoding 의 유형은 보통의 직접 encoding (혹은 sigle-level encoding) 과 간접 encoding (혹은 multiple-level encoding)으로 나뉘어 진다. 간접 encoding 의 경우는 그림 20과 같이 특정한 field 가 decoder 를 제어한다.

6. 결 론

지금까지 언급한 LSI bit-slice 소자들을 사용하기 위하여 시스템 설계자는 고려하는 각각의 응용에 대해 복잡한 tradeoff 들을 평가하여야 한다.

Microprogrammable bit-slice 디지탈 프로세서는 주로 다음과 같은 경제적인 응용에 사용되어 진다. 첫째, 복잡한 SSI / MSI 수준의 hardwired logic 을 LSI microprogrammed logic 으로 대치할 경우, 둘째, 유연성 (flexible)이 있고 더욱 빠른 새로운 microprocessor-based 디지탈 시스템과 controller 를 설계할 경우, 셋째, 기존 컴퓨터의 operational software를 사용하기 위하여 기존 컴퓨터를 emulation 할 경우이다.

Bit-slice 시스템을 사용할 경우 많은 장점이 있으나, 또한 몇가지 단점도 있다. 예를 들면 일반적으로 마이크로프로그램에 의한 제어방식은 하드웨어에 의한 제어방식보다 속도가 느리며 bit-slice 시스템을 개발하기 위한 support tool 이 부족하다는 점이다. 또 여러개의 칩을 사용하게 되어 연결부위가 많아지므로 한개의 칩으로 구성된 마이크로 프로세서 보다 신뢰도가 떨어진다는 점이다. 이러한 단점이 있음에도 불구하고, bit-slice 마이크로프로세서 구조는 많은 수의 고속도 소자가 요구되는 real-time 처리가 필요한 응용에 매우 적합하다.

參 考 文 獻

[1] Nikitas A. Alexandridis; "Special moncgraph: Bit-sliced micro-processor architecture," IEEE Computer Mag., June, 1978.

[2] Glenford J. Myers; Digital system design with LSI bit-slice logic, John Wiley & Sons, Inc., 1980.

[3] Agrawala, A.K. and T.G. Rauscher; Foundation of microprogramming, New York: ACM Monograph Series, Academic Press, 1976.

[4] John Mick and James Brick; Bit-slice microprocessor design, McGraw-Hill, Inc., 1980.

[5] The Am 2900 Family Data Book, Advanced Micro Devices; 1979.