□ 技術講座 □

디지탈시스템과 마이크로 프로세서의 설계(Ⅱ)

金 明 恒*

요 약

첫 기술강좌에서 소개된 체계적 설계방법을 써서 calculator설계를 하는데 구조정의 기계조직, 제어순서를 결정했다. 이 강좌에서 데이타 부분과 제어부분 설계를 끝낸다. 제어부분 설계를 끝낸다. 제어부분은 Hardwired제어와 microprogrammed제어방법으로 설계한다.

1. 서 론

첫 강좌〔1〕에서 4개의 operation을 시행하는 calculator 설계를 체계적 설계방법으로 시작했다. 첫째로 calculator의 구조정의(architecture definition)을 하고 이 calculator에는 어떠한 소자(component)가 필요하다는 것을 선언(declare)하며 또 4개의 operation이 무엇이라는 것을 이 소자들을 써서 표시한다. 그리고 기계조직에서 calculator가 제어부분과 Data Flow 부분으로 되어있다는 것을 표시했다. 둘째로는 calculator의 모든 operation을 제어순서(control sequence)로 표시했다. 이 제어순서는 선언한 소자를 써서 clock pulse 하나로 시행되는 microoperation으로 모든 operation을 정확히 표시한다. 이 강좌에서 이 설계를 계속한다.

2.1 Calculator의 Control Sequencer

제어순서가 결정되면 이 제어를 시행하는 논리 시스템을 설계해야 한다. 이 논리시스템을 control sequencer라 하며 각 microoperation을 시행하는데 필요한 timing pulse를 출력으로 낸다. 이 제어시스템설계는 세가지 방법으로 할 수 있다.

1. 한 state에 대해서 한 flip-flop (one flip-flop per state)

2. Register와 Programmable Logic Array (PLA)

3. Microprogramming

첫 세가지 방법은 Hardwired control 이라 한다. Microprogram control은 주로 ROM을 쓰는 것이다. 그렇게 하므로서 제어에 변화가 있을 때는 ROM 내용만 바꾸므로서 기본적 구조는 바뀌지 않아서 좋다. Control sequencer는 그림 1과 같이 회로를 설계할 수 있다.

시작하는 switch를 닫으면 microoperation 1부터 시작한다. 시작하는 switch는 synchronizer를 통해서 microoperation 1을 위한 pulse가 1에서 나오며 microoperation 2는 conditional branch로 별도의 신호가 필요하지 않아서 flip-flop출력 Q_2와 instruction register bits IR_0와 IR_1와 적합하게 AND를 해서 microoperation 3, 9, 14를 위한 pulse를 3, 9, 14에서 낸다. Microoperation을 위한 신호가 3에서 나면 flip-flop출력 Q_3와 IR_1이 AND하므로 microoperation 5와 7을 위한 pulse를 5와 7에서 나온다. 그후 microoperation 5와 7을 시행후는 17 (Halt)로 간다. 다른 microoperation을 위한 pulse는 나머지 회로에서 똑같은 방법으로 설계한다. 이 방법은 설계방법으로 쉬운 것으로 한 state에 대한 flip-flop (one flip-flop per state)방법이라 하며 flip-flop이 제일 많이 써서 회로로서는 비능률적이지만 설계하는데 노력과 시간이 적게드는 장점이 있다. 설계방법 2, 3, 4는 나중에 더 설명하겠다.

* 正會員 : 美國 Cornell 大 工大 電氣工學科 教授 · 工博

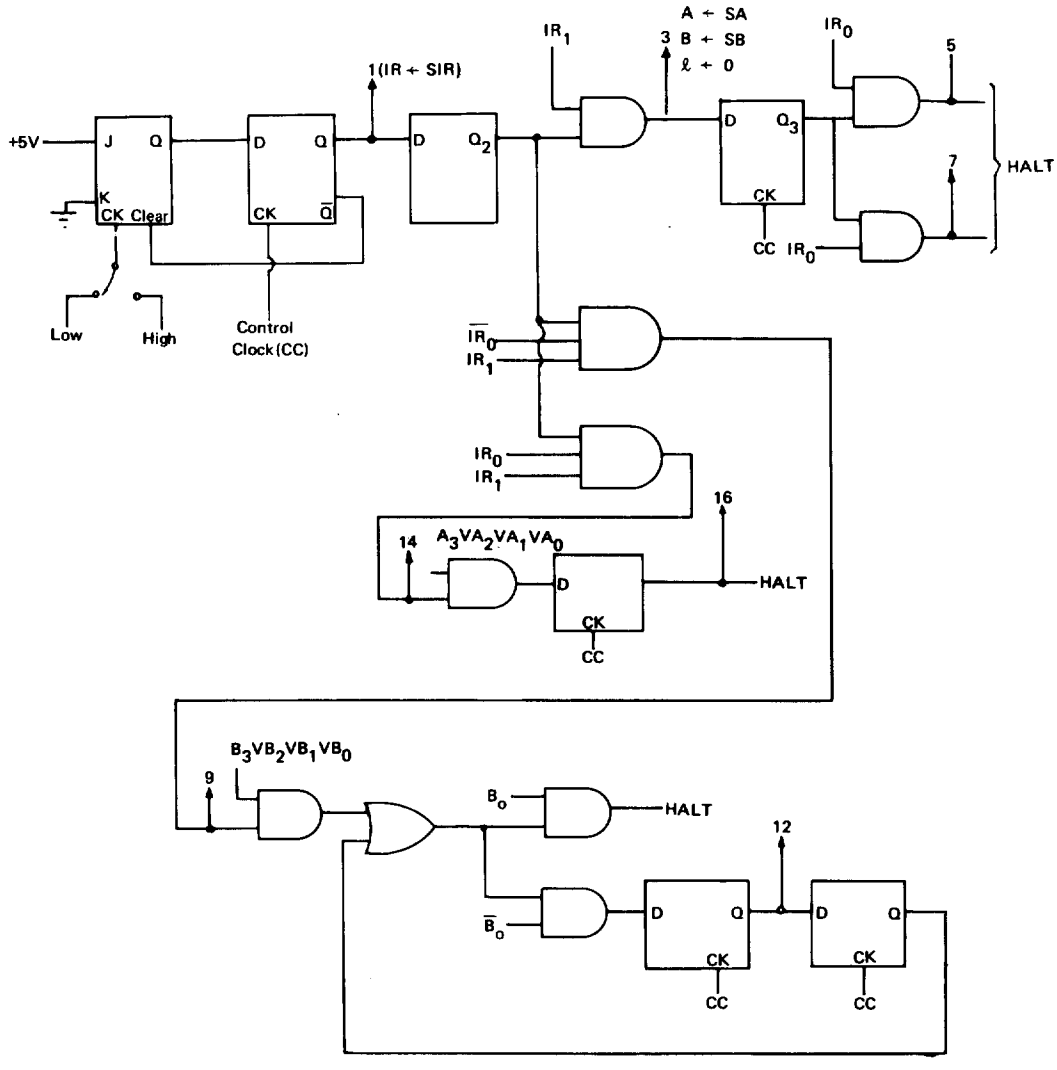

그림 1. Control Sequencer One Flip – Flop per State

2. 2 Data Flow

Data Flow 부분은 register 사이에 연결을 적합히해서 데이타가 정확히 움직이게 하는 부분이다. 각 microoperation 이 시행될 때 제어부분에서 오는 timing pulse 로 인해서 데이타가 정확히 움직여야 하다. 여기서 register A의 4 bit 입력은 microoperation 3과 14에서는 스윗치 SA에서 오고 microoperation 5와 7에서 ALU 출력에서 온다. 그러므로써 multiplexing 이 필요하다. 이러한 데이타의 움직임이 정확하기 위해서는 각 register와 ALU에다 데이타 처리와 움직임을 할 제어 pulse 을 적당히 mode 제어를 위해서 가해 주어야 한다. 이 mode 제어를 register A로 쓰이는 TTL칩 7419 4〔2〕에 어떻다는 것이 다음과 같은 mode 제어에서 알 수 있다.

이 mode control 표에는 register A가 네가지의 기능을 한다는 것이 표시되어 있다. 첫줄의 function 으로 A←SA이며 이것을 위한 modeterminal 은 Clear $=S_1=S_0=1$ 이며 microoperation 3과

Table 2.2.1 Mode Control Table 1

Register A 74194	Function	Mode Terminals			Microoperation No.
		Clear	S_1	S_0	
	A←SA	H	H	H	3 와 14
	l, A←A+B	H	H	H	5
	l, A←A−B	H	H	H	7
	A←right shift (A)	H	H	L	16
	No operation	H	L	L	나머지 micro-operations

Mode Control Logic for
Register A : Clear = High H = High = 1 L = Low = 0
(74194) $S_1 = 3 + 5 + 7 + 14 + 16$
$S_0 = 3 + 5 + 7 + 14$

14에서 시행된다. clear 와 S_1과 S_0의 mode control 논리는 이 표에서 clear는 항상 1이고 S_0은 microoperation 3과 5와 7과 14일때 1이 되므로 즉 control sequencer [1]의 timing 신호 펄스 3과 5와 7과 14이다. 따라서 $S_0 = 3 + 5 + 7 + 14$이다. 또 S_1은 이 표에서 microoperation 3과 14와 7과 16일때 1이므로 $S_1 = 3 + 5 + 7 + 14 + 16$이다. 다음 mode control 표 2는 microoperation 5와 7을 시행할 때 ALU의 mode terminal 들의 논리식을 표시한다. 똑같은 방법으로 register B와 register 1과 register IR을 위한 mode control 표를 구할 수 있다. 이것으로 제어부분 설계를 한 것이다.

다음은 data flow 부분의 설계를 두 부분으로 나누어서 할 수 있다. 첫째는, Data path Connection 표를 구한다. 이 표는 해당되는 microoperation을 시행하는 데 각 register 나 ALU의 입력이 어디서 온다는 것을 표시한다. Dath Path Connection Table 1이 register A의 입력이 어디서 온다는 것을 표시하는데 첫줄에서 microoperation 3과 14를 시행할 때 입력이 스윗치 SA에서 온다는 것을 표시한다. ALU에 대해서는 Table 2에 있다. 다른 register 을 위해서 똑같은 방법으로 결정한다.

ALU의 mode control table 은 다음과 같다.

Data Path Wiring 은 그림 2와 같이 구할 수 있다. Data Path Witing 은 Data Path Connection 표와 control sequencer 에서 나오는 timing pulse 를 써서 각 TTL chip 의 Data terminal를 적합하게 연결한 것을 표시한다.

Table 2.2.2 Mode Control Table 2

ALU 74181	Function	Mode Terminals						Microoperation No
		M	C_N	S_3	S_2	S_1	S_0	
	add	L	H	H	L	L	H	5
	Subtract	L	L	L	H	H	L	7

Mode Control Logic for ALU : M = L
(74181) $C_N = 5$ $S_3 = 5$ $S_2 = 7$ $S_1 = 7$ $S_0 = 5$

Table 2.2.3 Data Path Connection Table 1

Register A 74194	Function	Inputs to Register A from	Microoperation No
	A←SA	Switch SA	3와 14
	l, A←A+B	F terminals of ALU	5
	l, A←A−B	F terminals of ALU	7

Data Path Connection Tabble 2

ALU 74181	Function	Inputs to ALU from	Microoperation No.
	l, A←A + B	Register A to A terminals Register B to B terminals	5
	l, A←A − B	Register A to A terminals Register B to B terminals	7

2.3 State Diagram

첫강좌[1]에서 이 calculator 의 operation을 microoperation 으로 표시한 제어순서를 State Diagram으로 표시할 수 있다. Microoperation은 한 clock pulse 시간을 시행하는데 필요하다. 제어순서중에서 이러한 operation 을 state 로 assign 하다. 즉 이 calculator 제어순서에서는 No. 1, 3, 5, 7, 9, 12, 14, 16,과 17이며 이것은 T_1, T_3, T_5등으로 정했다. No. 2, 4, 6, 8, 10, 11, 13, 15는 branch operation 으로 펄스가 필요없이 여기에 나타나는 논리 variable 을 순서논리 (sequential logic)

의 입력으로 정하고 state transition에 쓴다. 그림 3과 같은 state diagram을 결정할 수 있다.

2.4 Register와 Programmable Logic Array (PLA)

이 제어방법은 synchronous 순서논리를 써서 timing pulse를 출력으로 내는 것이다. 그러므로 위의 state diagram을 써서 state table을 구하고 나서 출력과 flip-flop 입력함수를 PLA〔3〕로 시행하는 것이다. D flip-flop을 써서 이러한 방법으로 control sequencer를 설계하면 그림 4의 회로가 된다. 그림 3의 state diagram을 state table와 D flip-flop은 채택한 입력을 포함한 표를 Table 2.4.1에 표시했다. 이 표에서 첫줄에서 Pre-

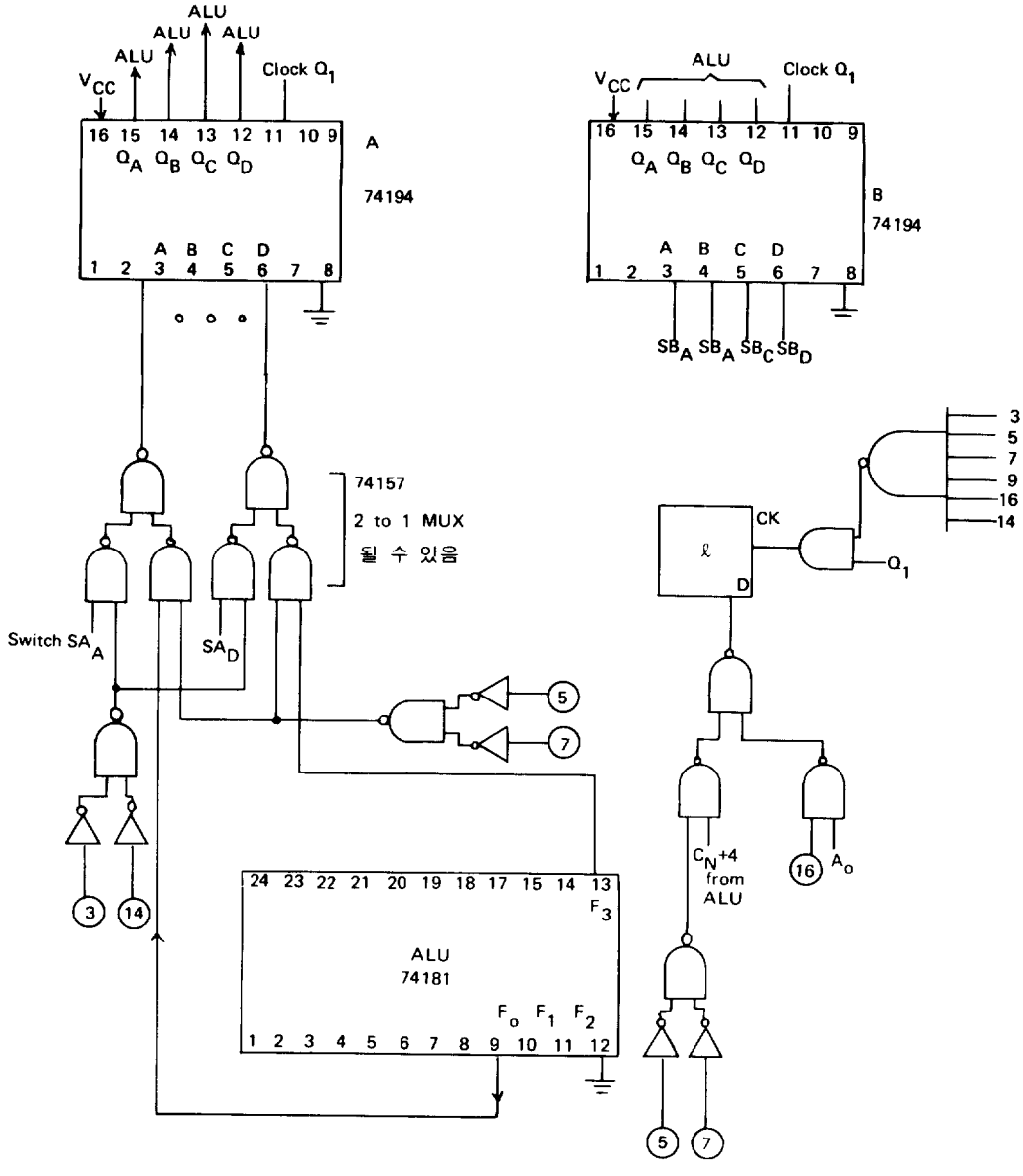

그림 2. Data Path Wiring

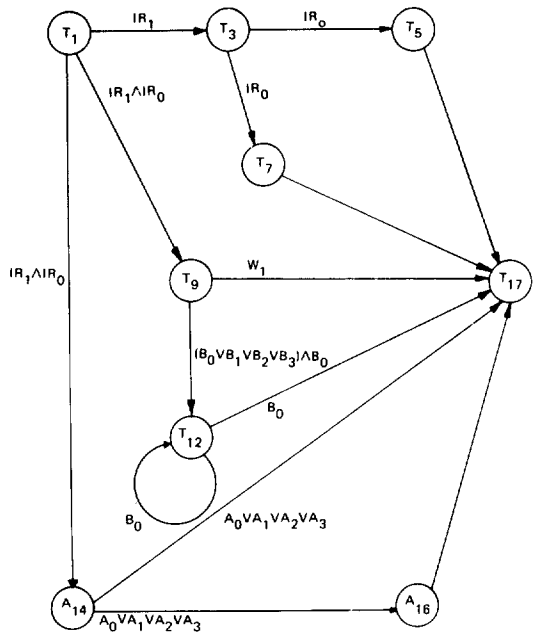

그림 3. Calculator의 State Diagram

sent state가 0000 (T_1 state)이며 $IR_1 = 0$ 일때 next state가 0001 (T_3 state)로 가며 다른 입력 IR_0, B_0와 PA($=A_0VA_1VA_2VA_3$)와 PB($=B_0VB_1VB_2VB_3$)은 관계되지 않으므로 don't care condition이다. 그림 4에 표시되어 있는 D flip-flop의 입력과 각 state를 출력이 state table를 써서 결정하여야 한다. 여기서 PLA를 써서 combinational circuit를 설계하는 데 이 PLA의 입력은 순서논리회로의 입력인 IR_1, IR_0, PA, PB, B_0와 D flip-flop의 출력인 y_0, y_1, y_2, y_3이다. PLA의 출력은 T_1, T_3, T_5, T_7, T_9, T_{12}, T_{14}, T_{16} T_{17}과 D flip-flop의 입력 Y_0, Y_1, Y_2, Y_3이다. 따라서 Table 2,4,2의 PLA Table을 구할 수있다

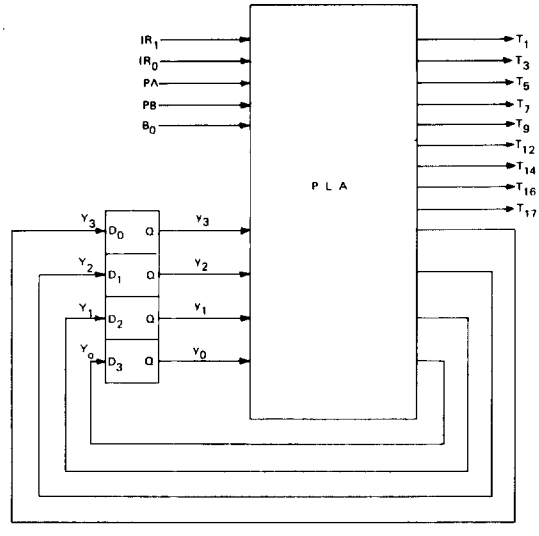

그림 4. Calculator의 Register의 PLA Control Sequencer

Table 2.4.1 Calculator의 State Table과 Flip-flop Inputs

Present State				INPUT					Next State				Flip-flop			
y_3	y_2	y_1	y_0	IR_1	IR_0	PA	PB	B_0	Y_3	Y_2	Y_1	Y_0	D_3	D_2	D_1	D_0
0	0	0	0 (T_1)	0	d	d	d	d	0	0	0	1 (T_3)	0	0	0	1
0	0	0	0 (T_1)	1	0	d	d	d	0	1	0	0 (T_9)	0	1	0	0
0	0	0	0 (T_1)	1	1	d	d	d	0	1	1	0 (T_{14})	0	1	1	0
0	0	0	1 (T_3)	d	0	d	d	d	0	0	1	0 (T_5)	0	0	1	0
0	0	0	1 (T_3)	d	1	d	d	d	0	0	1	1 (T_7)	0	0	1	1
0	0	1	0 (T_5)	d	d	d	d	d	1	0	0	0 (T_{17})	1	0	0	0
0	0	1	1 (T_7)	d	d	d	d	d	1	0	0	0 (T_{17})	1	0	0	0
0	1	0	0 (T_9)	d	d	d	1	1	1	0	0	0 (T_{17})	1	0	0	0
0	1	0	0 (T_9)	d	d	d	1	0	0	1	0	1 (T_{12})	0	1	0	1
0	1	0	1 (T_{12})	d	d	d	d	0	0	1	0	1 (T_{12})	0	1	0	1
0	1	0	1 (T_{12})	d	d	d	d	1	1	0	0	0 (T_{17})	1	0	0	0

0	1	1	0 (T_{14})	d	d	d	d	d	1	0	0	0 (T_{17})	1	0	0	0
0	1	1	0 (T_{14})	d	d	1	d	d	0	1	1	1 (T_{16})	0	1	1	1
0	1	1	1 (T_{16})	d	d	d	d	d	1	0	0	0 (T_{17})	1	0	0	0

$T_1 = 0000$ $T_{12} = 0101$ $PA = A_0\,VA_1\,VA_2\,VA_3$

$T_3 = 0001$ $T_{14} = 0110$ $PB = B_0\,VB_1\,VB_2\,VB_3$

$T_5 = 0010$ $T_{16} = 0111$ d = don t care

$T_7 = 0011$ $T_{17} = 1000$

$T_9 = 0100$

Table 2.4.2 Calculator의 PLA Table

Product term	INPUTS									OUTPUT												
	y_3	y_2	y_1	y_0	IR_1	IR_0	PA	PB	B_0	Y_3	Y_2	Y_1	Y_0	T_1	T_3	T_5	T_7	T_9	T_{12}	T_{14}	T_{16}	T_{17}
1	0	0	0	0	0	X	X	X	X	0	0	0	1	1	0	0	0	0	0	0	0	0
2	0	0	0	0	1	0	X	X	X	0	1	0	0	1	0	0	0	0	0	0	0	0
3	0	0	0	0	1	1	X	X	X	0	1	1	0	1	0	0	0	0	0	0	0	0
4	0	0	0	1	X	0	X	X	X	0	0	1	0	0	1	0	0	0	0	0	0	0
5	0	0	0	1	X	1	X	X	X	0	0	1	1	0	1	0	0	0	0	0	0	0
6	0	0	1	0	X	X	X	X	X	1	0	0	0	0	0	1	0	0	0	0	0	0
7	0	0	1	1	X	X	X	X	X	1	0	0	0	0	0	0	1	0	0	0	0	0
8	0	1	0	0	X	X	X	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0
9	0	1	1	0	X	X	X	1	0	0	1	0	1	0	0	0	0	1	0	0	0	0
10	0	1	0	1	X	X	X	X	0	0	1	0	1	0	0	0	0	0	1	0	0	0
11	0	1	0	1	X	X	X	X	1	1	0	0	0	0	0	0	0	0	1	0	0	0
12	0	1	1	0	X	X	0	X	X	1	0	0	0	0	0	0	0	0	0	1	0	0
13	0	1	1	0	X	X	1	X	X	0	1	1	1	0	0	0	0	0	0	1	0	0
14	0	1	1	1	X	X	X	X	X	1	0	0	0	0	0	0	0	0	0	0	1	0

2. 5 Microprogram 제어

제어를 위해서 timing signal을 flip-flop 를 쓰지않고 저장용(memory) register 를 쓸 수 있다. Memory에 기억된 word 를 제어에 쓰는 데 한 bit 이 1이면 이것을 제어 timing signal 로 쓰며 다음 timing signal 은 다음 memory word 에서 얻는다. 이렇게 쓰이는 memory control word 를 microinstruction 이라 하며 이 제어방법을 micro-program 제어라 한다. Microprogram 제어는 모든 제어 WORD를 ROM에 넣어서 계속 읽는(Read) operation을 하여 timing signal 을 낼 수 있다.

Microprogram 제어는 일반적으로 다음과 같은 구조를 갖는다.

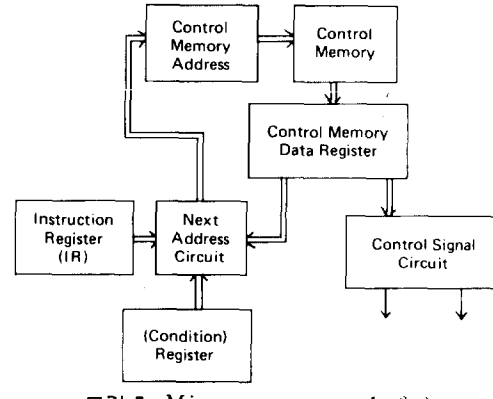

그림 5. Microprogrammed 제어

제어 word 로 쓰이는 microinstruction 은 format 이 다음과 같다.

Control Signal	Next Address

Microinstruction 이 디지탈 시스템 소자들을 위한 microoperation 을 지시하며 이 microoperation 을 시행하는 데 쓰이는 제어 timing signal 을 Control Signal Field에서 낸다. 해당되는 microoperation을 시행하면 Next Address Field 와 instruction register 와 다른 해당되는 register 에 있는 데이타를 써서 Next Address Circuit 에서 다음 control memory 의 Next Address 를 결정한다. 다음 Microinstruction 이 현재 Address 다음이거나 그렇지 않으면 Address 에 있을 수도 있다.

Calculator 의 microprogrammed control sequencer 는 그림 6 과 Table 2.5.1에 표시되어 있으며 ROM에 있는 microinstruction 은 Table 2.5.1에 표시되어 있다. ROM Address 0000 에는 state T_1 의 timing signal 을 위해서 bit 가 1 이며 Next Address Field 에는 0000 이며 select 는 001 이다. Table 2.5.1에 보면 select 가 001 이면 control Address Register (CAR) 의 입력을 기억한다. 따라서 $IR_1 = 0$ 이면 Next Address 가 0001 이며 $IR_1 \wedge \overline{IR_0} = 1$ 이면 0100

Table 2.5.1 Multiplexer 의 기능

ROM bits 13	14	15	Multiplexer Selet
0	0	0	Increment CAR
0	0	1	입력을 CAR에 기억
0	1	0	$\overline{IR_0} = 1$ 이면 입력을 CAR, $\overline{IR_0} = 0$ 이면 increment CAR
0	1	1	$PB \wedge B_0 = 1$ 이면 입력을 CAR, $PB \wedge B_0 = 0$ 이면 increment CAR
1	0	0	$\overline{B_0} = 1$ 이면 입력을 CAR, $\overline{B_0} = 0$ 이면 increment CAR
1	0	1	$\overline{PA} = 1$ 이면 입력을 CAR, $\overline{PA} = 0$ 이면 increment CAR

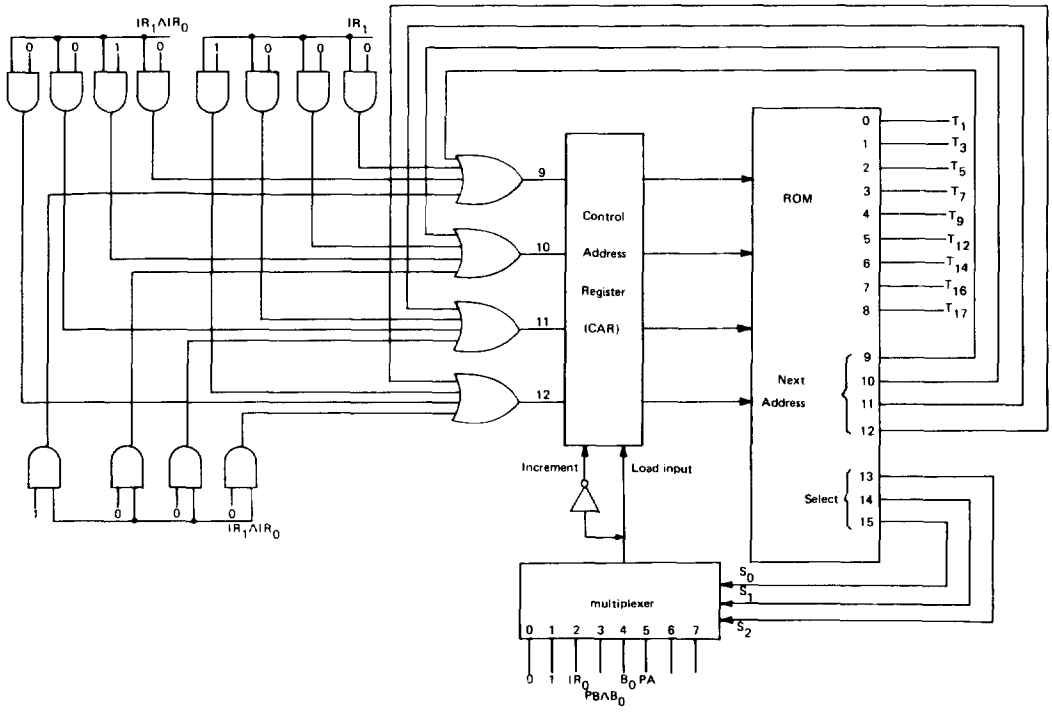

그림 6. Calculator 의 microprogrammed Control Sequencer

Table 2.5.2 Calculator의 제어 Microprogram

ROM Address	ROM OUTPUTS															
	T_1	T_3	T_5	T_7	T_9	T_{12}	T_{14}	T_{16}	T_{17}	Next Address				Select		
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0000	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0001	0	1	0	0	0	0	0	0	0	0	0	1	1	0	1	0
0010	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	1
0011	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	1
0100	0	0	0	0	1	0	0	0	0	0	1	1	0	0	1	1
0101	0	0	0	0	0	1	0	0	0	0	1	0	1	1	0	0
0110	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1
1000	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1
1001	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	1

이며 $IR_1 \wedge IR_0 = 1$이면 1000이다. 다음 줄인 ROM Address 0001에는 state T_3가 1이며 Next Address는 0011이다. 그러나 select는 010이므로서 (Table 2.5.1) $\overline{IR_0} = 1$이면 Next Address Field에 있는 T_7을 위한 0011을 CAR에 기억하며 그렇지 않고 $\overline{IR_0} = 1$이면 increment하므로 Next Address가 T_5을 위한 0010이라는 것이다.

2.6. 결 론

이 논문에서 체계적 설계방법을 써서 calculator를 설계했다. 이러한 방법으로 설계된 디지탈 시스템은 제어부분과 데이타 부분으로 되어 있으며 제어부분 설계를 위해서 제어순서를 항상 결정하여야 한다. 이 제어순서의 microoperation을 state diagram으로 표시할 수 있다. 제어부분 설계는 one flip - flop per state 방법을 쓰거나 state diagram을 써서 state table을 구해서 synchronous sequential circuit로 실현하므로서 combination circuit 부분을 PLA로 설계할 수 있다. 이 두 가지 방법으로 설계된 제어를 Hardwired 제어라고 한다. Microprogramming을 써서 제어부분을 설계한 것을 microprogrammed 제어라 한다. 이 세가지 방법을 다써서 calculator를 설계했다.

참 고 문 헌

[1] 김명환, "디지탈 시스템과 마이크로 프로세서 설계 : Ⅰ, 체계적 설계 - 1", 전기공학회지, 제 31권, 1982. 7월.

[2] The TTL Data Book for Design Engineers Texas: Texas Instrument, Inc..

[3] F. Hill and G. Peterson; Introduction to Switching Theory and Logic Design, 3rd ed. New York: John Wiley and Sons, Inc., 1980.

[4] F. Hill and G. Peterson; "DIGITAL SYSTEM: Hardware Organization and Design", 2nd Ed., John Wiley & Sons, Inc., 1978, pp. 121-420.

[5] M. Mano; "Digital logic and computer design", Prentice-Hall, Inc., Englewood Clifts, N.J., 1979, Chapter 8.

[6] Mano, M.M.; 'Computer system architecture", Prentice Hall, Inc., Englewood Clifts, N.J., 1976.

[7] Kline, M.M.; "Digital computer design", prentice-Hall, Inc., Englewood Clifts, N.J., 1977.

[8] Chu, Y; "Computer organization and microprogramming", Prentice-Hall Inc., Englewood Clifts, N.J., 1972.