제3회 엘썸스쿨 세미나 자료 (2000년 3월 23일~25일)

PLD와 PALASM4 사용법

강사 : 황 민 순
e-mail : minsoon@shmail.hanarotel.co.kr
(엘썸스쿨 객원 강사 /현 SpaceRobotics 근무)

1. PAL이란 무엇인가?

PAL(Programmable Array Logic)이란 PLD(Programmable Logic Device) 중의 하나로서 AMD(현재의 Vantis)社에서 만든 제품의 상표명이다. 현재는 많은 제품과 개발 툴이 지원되고 있다.

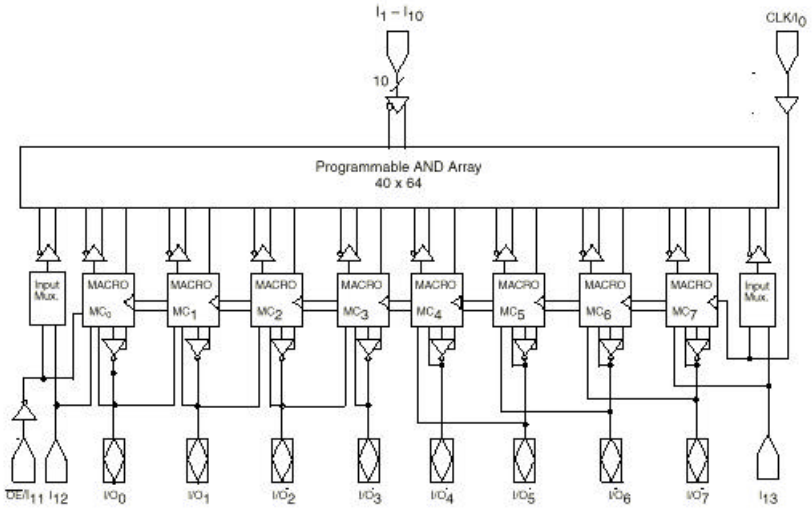
PLD는 AND, OR 게이트 등 논리 조합 회로를 임의로 설계해서 입력시켜서 여러 개의 TTL 칩을 사용하는 것과 같은 효과를 가질 수 있고, 또한 출력 핀으로 지정된 핀에서는 D-FlipFlop이 내장되어 있어서 단순한 조합 논리회로를 사용할 것인지, 시간 개념이 첨가된 플립플롭 회로를 사용할 것인지 결정할 수 있다. 일종의 LATCH처럼 사용할 수 있는 것이다.

다음은 우리가 사용하는 PALCE20V8H-4에 관한 일반적인 설명이다.

PALCE20V8는 저전력, 고속, 전기적인 설정 삭제 가능한 진보된 PAL디바이스이다. 이 칩의 매크로셀은 표준적인 장치 구조이다. PALCE20V8은 GAL20V8과 호환되며, PAL20R8으로도 대체된다.

PALCE20V8은 쉽고 효율적으로 복잡한 논리회로를 구현할 수 있는 친숙한 Sum Of Product (AND-OR)구조를 이용할 수 있다. 조합 논리의 다중 레벨은 sum of product 형태로 단순화될 수 있고, 매우 많은 입력 핀의 이점을 살릴 수 있다. 방정식(Equation)은 전기적으로 삭제 가능한 AND회로 부동 게이트 셀을 통한 장치로 프로그램될 수 있다.

BLOCK DIAGRAM



고정 OR 어레이는 8개의 AND 항을 출력 핀 1개에 설정할 수 있다. 이 AND 항의 OR 합은 출력 매크로셀로 연결된다. 각각의 매크로셀은 Active Low, High 두 가지로 Registered와 Combinatorial 모드로 프로그램 할 수 있다. 출력설정은 각 매크로셀의 2개의 전역 비트와 1개의 개별 비트로 제어되는 4개의 멀티플렉서를 통해서 결정된다.

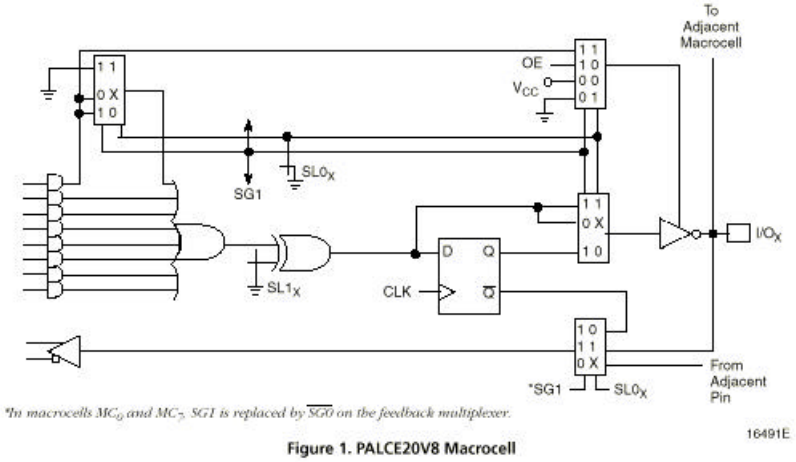
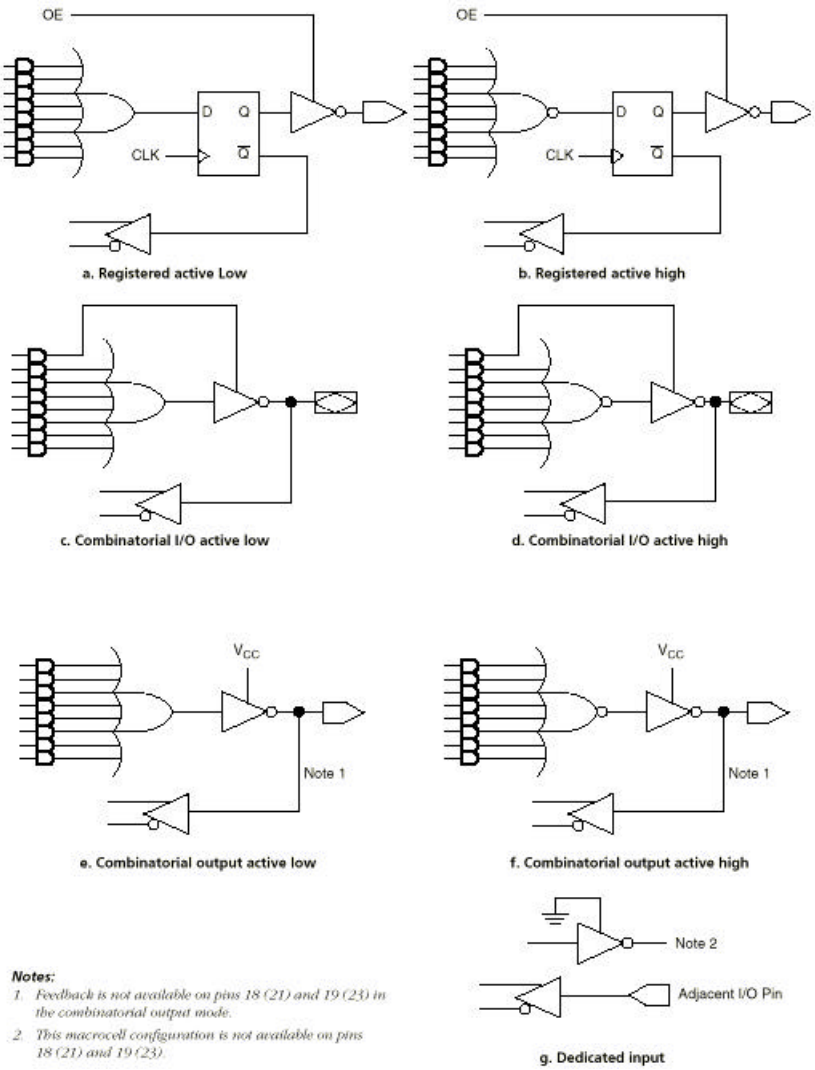


Figure 1. PALCE20V8 Macrocell

각 출력 핀의 모드를 아래와 같이 나타내었다.



Notes:

1. *Feedback is not available on pins 18 (21) and 19 (23) in the combinatorial output mode.*
2. *This macrocell configuration is not available on pins 18 (21) and 19 (23).*

다음에 실제 입력 핀에서 출력까지의 Logic Diagram의 일부(계속 비슷한 패턴임)를 나타내었다.

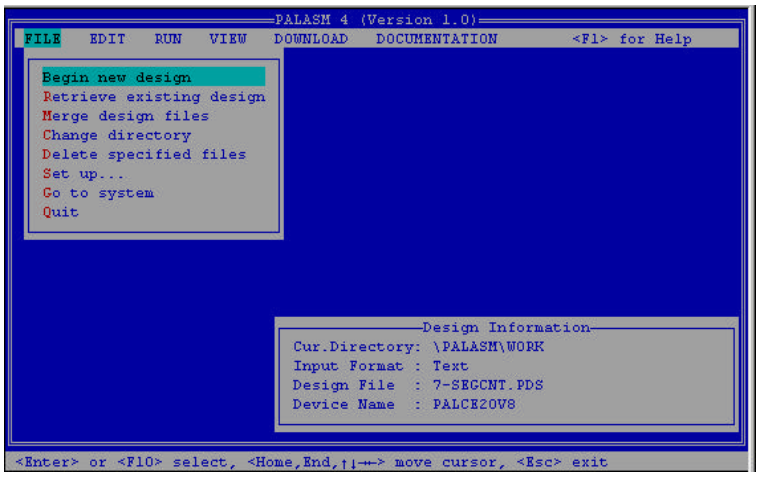
2. PALASM의 사용법

PALASM이란 간단한 논리식으로 AND-OR 게이트 조합을 나타낸 것으로, PAL에 쓸 데이터 파일을 만들기 위해서 사용하는 일종의 프로그램이다. 먼저 이 프로그램을 실행하는 방법을 알아보고 다음으로 그 문법을 간단히 알아본다.

(참고)인스톨과정은 세미나때 설명됩니다.

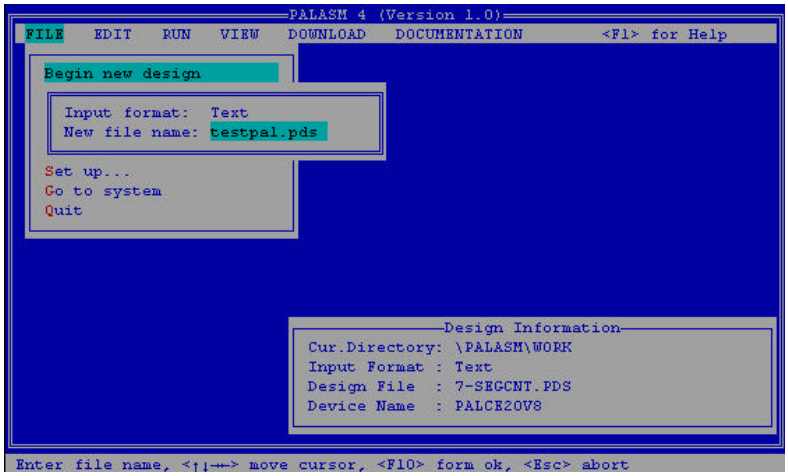
1 c:\palasm\exe\palasm.exe를 실행시키면 다음과 같은 화면이 나오게 된다. 아무키나 누르면 다음으로 넘어간다.

2 먼저 File 메뉴를 설명하면 다음과 같다

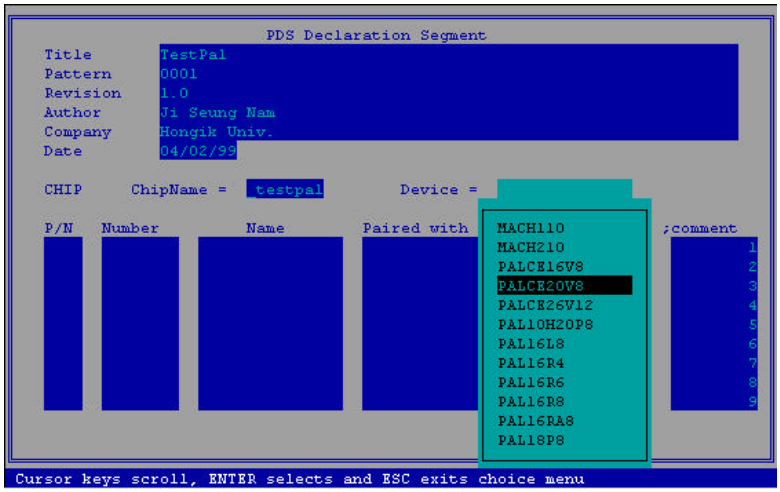


B : 새로운 PAL의 디자인
D : 기존의 디자인 불러오기
M : 디자인 파일 합치기
C : 현재 디렉토리 변경
D : 컴파일시 생성되는 파일들 중 필요없는 것을 지우는 옵션
G : 잠시 Dos로 돌아감

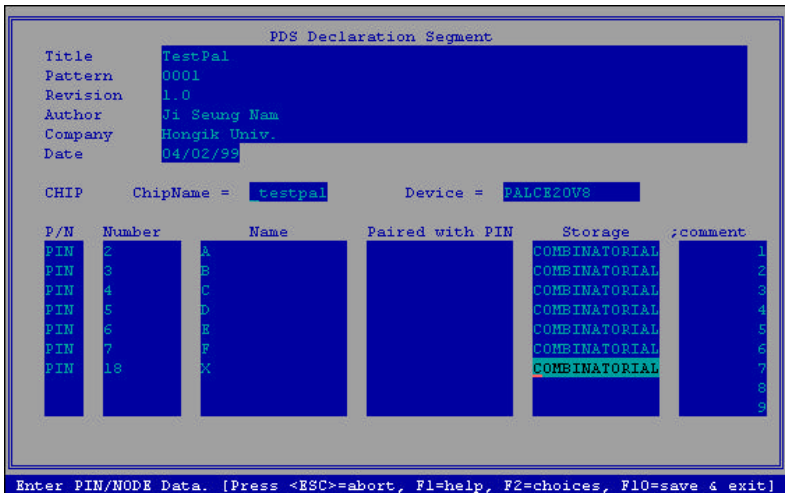
3 소스 파일 이름을 설정한다.



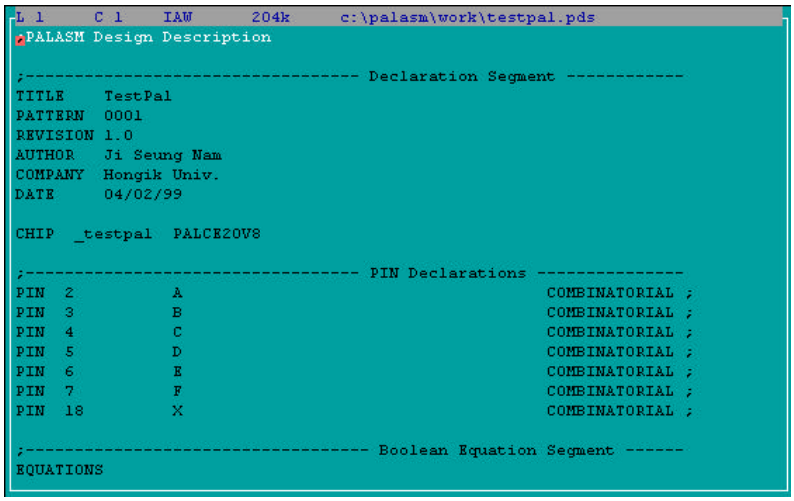
4 아래를 보면 Device를 설정하는 모습이 나와 있다. 우리가 실험에서 사용할 칩은 PALCE20V8이다.



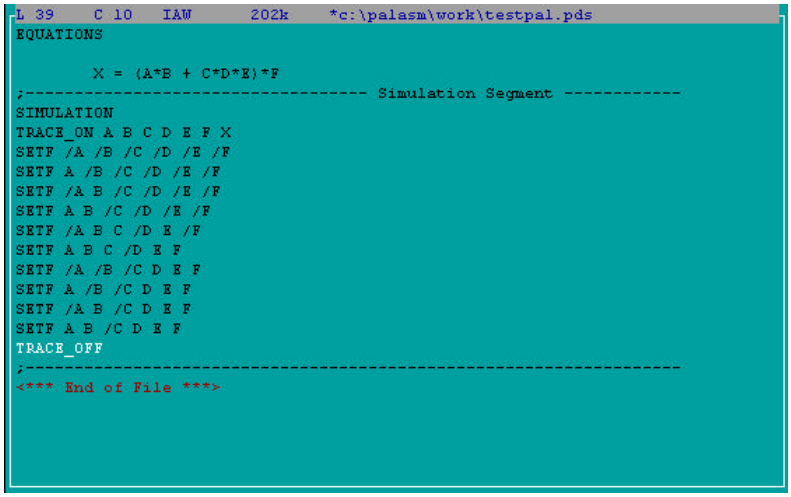
5 아래 그림과 같이 각 PIN을 설정해 준다. Number는 Chip의 핀 번호로 외형도를 참고하면 된다. (주의 : 핀마다 특징이 있다.)
NAME은 Boolean Equation에서 사용할 Signal의 이름이다.
Storage : 시간개념이 없는 조합 논리회로는 Combinatorial, 플립플롭을 사용하는 경우는 Registered로 된다.



6 다 마쳤으면 F10을 누른다. 그러면 다음과 같은 화면이 나타날 것이다.
다음 화면에서는 에디터에서 이미 TITLE에서 PIN 정의까지 위의 작용으로 자동으로 코딩되어 있는 것을 나타내 준다.

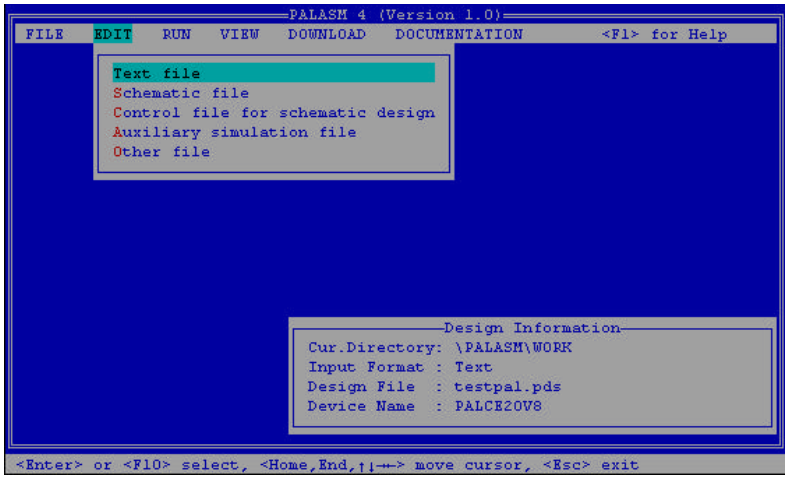


PageDown을 눌러 아래 EQUATION이 있는 곳을 한번 보자.
그리고 빈칸에 쓰여진 대로 하게되면 A, B, C, D, E, F를 입력으로 갖는 AND와 OR의 조합 논리회로의 예제가 된다. 아마 금방 진리표를 그릴 수 있는 이해가 충분히 될 것이라고 생각한다.

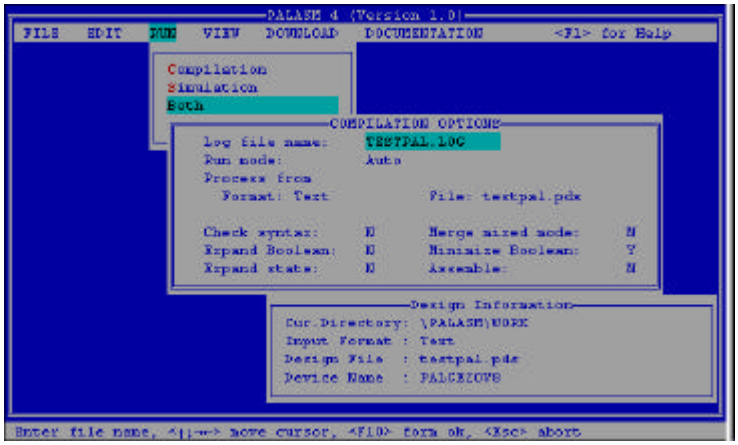


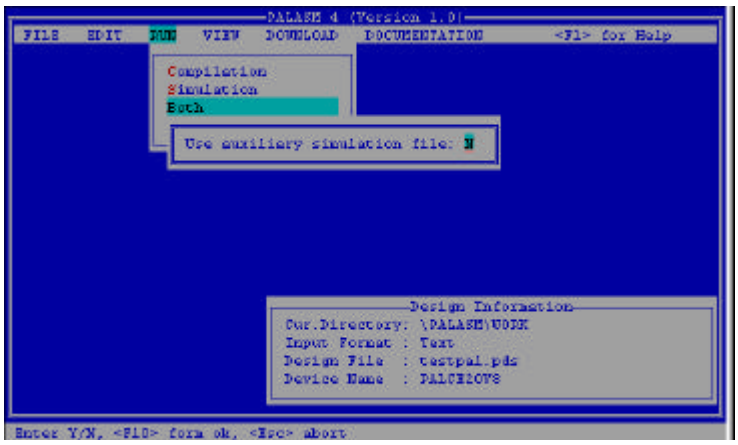
그리고 SIMULATION에는 파악해야 할 SIGNAL을 TRACE_ON으로 설정하고 SETF는 각각의 SIGNAL을 HIGH로 셋팅하는 시뮬레이션 전용 명령이다. 따라서 /A 라고 하면 A에는 LOW가 입력된다.

⑦ EDIT 메뉴는 다음과 같다
T : 텍스트 파일 편집 주로 PDS 디자인파일
S : ORCAD SCHEMATIC 파일
C : 설명 그대로
위에서 모든 것을 틀림 없이 입력하였다면, RUN 메뉴에 가서 컴파일과 시뮬레이션을 실행시켜보자. 컴파일은 PAL 칩에 직접 들어갈 JEDEC 파일을 생성하는 것이고, 시뮬레이션은 이 파일을 바탕으로 예상되는 결과를 LOW, HIGH로 분류해 두는 것이다.



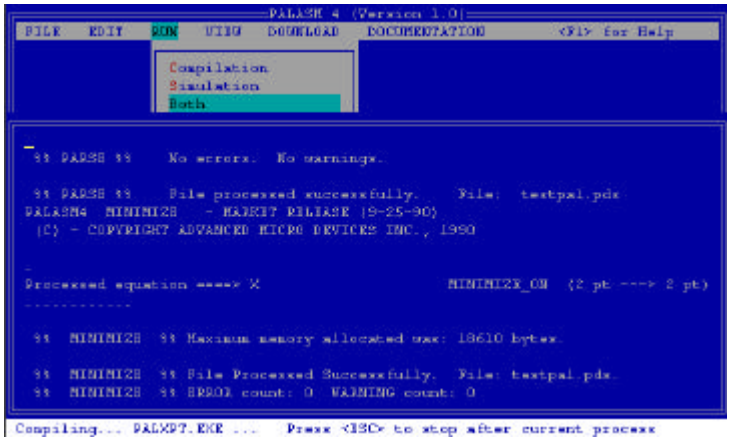
8 메뉴가 아래와 같이 나오면 그대로 F10을눌러 진행한다.



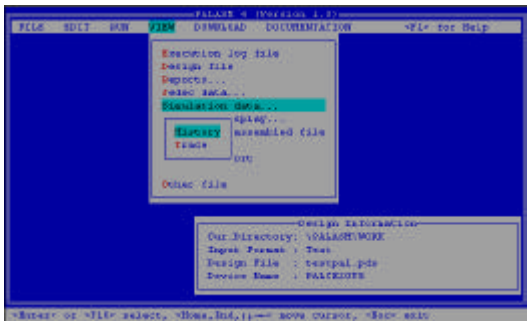


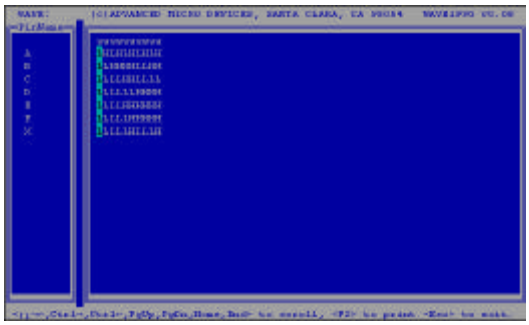
마찬가지로 진행한다.

⑨ 그러면 다음과 같은 결과가 나타날 것이다.



그리고 아래 화면에서 Simulations을 해 보면 아래 두 화면을 금방 실행할 수 있을 것이다.





3. 기본 문법

* OPERATOR (연산자)

Operator	Definition	Example
/	Not	/A
*	And	A * B
+	OR	A + B
:+:	XOR	A :+: B
:*:	XNOR	A :*: B
=	Combinatorial	INPUT1 = A * B
:=	Registered equation	INPUT1 := A * B
:=	State equation	STATE1 := START -> END
*=	Latched	INPUT1 *= A * B
->	State transition	STATE1 := START -> END
+->	Local default	+-> RED -> WAIT
;	Comment	;set low before clocking
,	Literal separator	IN[1,3,4] IN[1..4,6..9]
..	Range	INPUT[0..9]
:	CASE value	0,1:
[]	Term brackets	INPUT[0..9]
()	Expression	IN1 = (A * B) (C * D * F)
{}	Substitute	OUT1 = A * B * C OUT2 = {OUT1} * F
>	Greater than	IF A > 2 THEN...
<	Less than	WHILE A < 2 DO...
<>	Not equal to	IF A <> 2 THEN...
<=	Less than or equal to	WHILE A <= 2 DO...
>=	Greater than or equal	WHILE A >= 2 DO...
%	Don't care	DEFAULT_OUTPUT %OUT1
?	CHECK clash	------??????
' '	String delimiters	STRING INPUT 'A1 + /A2'
#b	Binary radix	#b101000
#d	Decimal radix	#d40
#o	Octal radix	#o50
#h	Hexadecimal radix	#h28B
Space, tab	Separator	PIN 2 IN1 REG

* BOOLEAN EQUATION (대수식 관련해서는 Help 파일 참조)

4. 개념 설명

PLD를 잘 이해해야 여러분들이 나중에 훌륭한 엔지니어가 될 수 있을 것이라고 생각한다. 그 이유는 앞으로 EPLD, FPGA, ASIC등이 발달이 될것이다. 하드웨어 엔지니어가 지금은 소자, CPU, 저항, TR등을 가지고 연결하여 회로를 그리지만 앞으로 몇 년 안에 이 모든것이 하나의 칩에 내장되어 프로그램하여 설계자의 의도대로 각종 게이트, 디바이스, CPU CORE등의 ROUTE를 얼마든지 변경하고, FIX하게 될 것이다. 이것을 주도하는 것이 HDL(HARDWARE DESCRIPTION LANGUAGE)이다. 종류로는 VHDL, AHDL, CUPL, ABEL 등이 있는데 회사마다 다소 차이가 있다. 결국 하드웨어도 프로그램을 잘해야 컴팩트한 회로가 완성된다는 것이다. 그러기 위해 우선 가장 쉽고, 지금 당장쓸 수 있고. 앞으로도 몇십년(?)은 쓸 수 있는 PALASM에 대해서 공부하고자 한다.

논리회로는 크게 두가지 종류가 있다. COMBINATORIAL LOGIC (조합회로)과 SEQUENTIAL LOGIC (순차회로)으로 나뉜다. 조합회로는 입력 상태가 변하면 곧바로 출력이 따라 변하지만 순차회로는 입력 변화에 따라 출력이 바로 변하는 것이 아니라 또 하나의 입력인 기준클럭에 동기해서 모든 출력의 상태가 변하게 된다.

4.1 조합회로 예제

예를 들어서 조합논리는 기본적으로 74139의 반쪽을 구현해보자면

/Y0 = /IN0 * /IN1

/Y1 = /IN0 * IN1

/Y2 = IN0 * /IN1

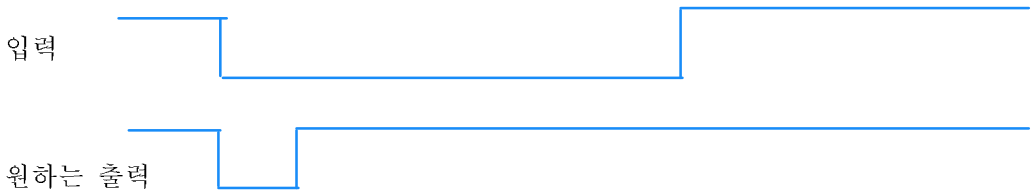
/Y3 = IN0 * IN1

처럼 구현할수 있다.(가장 기본이 되는것이다.)

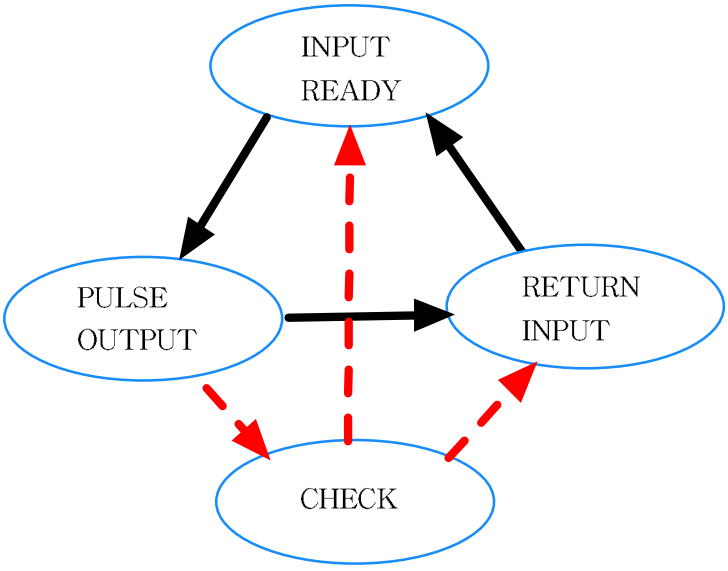
4.2 순차회로 예제 (스테이트머신)

SM(STATE MACHINE, 이하 SM으로 칭함)은 필요한 스테이트를 설정하여야 한다. 직접 설계를 해가면서 설명하기로 한다.

예를들어 ONESHOT PULSE를 만든다고 하자. 주로 인터럽트 로직이나 입력을 레벨이 아니고 에지 트리거 방식으로 요구하는 것들이 있다.



INPUT이 매우 길거나 여러 가지 이유로 내려가지 않는 경우가 발생할 수 있으므로 이것은 조합논리로 도저히 구현할 수가 없다. 그래서 이런것을 주로 스테이트 머신으로 구성한다. 우선 STATE가 몇 가지가 있나 추정해봐야 한다. 우선 입력대기(READY), 출력, 입력복귀의 3가지 스테이트가 만들어진다. 이것을 그림으로 나타내면



INPUT READY --> S0
PULSE OUTPUT --> S1
RETURN INPUT --> S3 라고 가정하자.

S0의 상태를 보면 INPUT이 평상시 LOW이다가 HIGH로 올라가면 S0 -> S1의 상태로 이동하게 된다. INPUT이 LOW면 상태의 이동은 없다. S1도 마찬가지로 정리하자면 INPUT에 무관하게 다음으로 넘어갈 수 있다. S3은 INPUT이 HIGH인 경우 제자리이고 LOW로 떨어지면 S1로 이동한다.

여기서 펄스의 폭을 조금만 늘이거나 (1 클럭분) 노이즈를 고려해 볼 수 있다. 그러기위해 S2를 S1과 S3사이에 하나 넣자. 그러면 상태 서술이 많이 틀려질 수 있다. 우선 S1은 INPUT이 LOW로 떨어져 있으면 S1 -> S0로 되돌아 갈 수 있다. (이유는 노이즈일 가능성) 그래서 여기에 조건을 넣어 INPUT이 HIGH를 유지해야만 S1 -> S2로 이동하게 만든다. S2는 출력을 종료하는 스테이트이다. (여기에도 경우에 따라서 조건을 넣을 수 있다.)

여기서 상태는 총 4가지이므로 4가지 경우를 표현하기 위해서는 2비트가 있으면 된다. 그 비트 2개를 SB0, SB1이라고 하고 S0 = 00, S1 = 01, S2 = 11, S3 = 10 으로 정하자.
편리하게 0-1-2-3이 아니고 왜 00 -> 01 -> 11 -> 10 으로 정했는가?

여기엔 깊은 뜻이 있다. 우선 GRAY CODE라는 것과 LOGIC의 컴팩트한 구성 때문이다. 그레이 코드라는 것은 무조건 1비트만 변하게 된다. 여러 비트가 동시에 변하면 안되게 된 수 체계이다. 이것은 주로 카운터에 많이 사용한다. 그냥 1-2-3-4 수 체계를 사용하면 3-4로 변할때 동시에 2비트가 변하므로 서로 천이 시간이 틀릴 경우 글리치(Glitch)가 발생하여 오동작과 노이즈의 원인이 된다. 주파수가 높아지면 매우 심각히 고려해야 한다.

```
if( INPUT ) then begin
   if( /SB0 * /SB1 ) then begin
      SB0 := 0
      SB1 := 0
   end
   if( /SB0 * SB1 ) then begin
      SB0 := 0
      SB1 := 0
   end
   if( SB0 * SB1 ) then begin
      SB0 := 0
      SB1 := 0
   end
   if( SB0 * /SB1 ) then begin
      SB0 := 1
      SB1 := 1
   end
end
else begin
   if( /SB0 * /SB1 ) then begin
      SB0 := 1
      SB1 := 0
   end
   if( /SB0 * SB1 ) then begin
      SB0 := 0
      SB1 := 1
   end
   if( SB0 * /SB1 ) then begin
      SB0 := 1
      SB1 := 1
   end
   if( SB0 * SB1 ) then begin
      SB0 := 0
      SB1 := 1
   end
end
```

구성하자면 이렇게 할 수 있다. (위의 것은 입력의 상태에 따라 스테이트 머신 비트의 값을 보고 결정) 반대로 각 스테이트 상태에 따라 입력 상태를 체크할 수도 있다. 결과는 똑같이 나온다. 단지 보는 관점에 따라서 표현될 뿐이다.

```
if 스테이트 then begin
  if INPUT then begin
     SB0 =
     SB1 =
  end
  else begin
     SB0 =
```

```
      SB1 =
  end
end
---위와 같이 나머지 스테이트 3개더 서술....---
```

여기서 출력을 어떻게 뽑을 것인가는 여러가지 방법이 있다. (정의 하기 나름이다.) 우선 노이즈면 S1 -> S0으로 가므로 S1은 출력하지 말아야 한다.

그러므로
OUTPUT = SB0 * SB1 <--- EQUATION
또는
IF SB0 & SB1 THEN BEGIN <--- DESCRIPTION
OUTPUT = VCC
END
으로 표현 할 수 있는데 편한 것을 사용한다.

만약 펄스 폭을 2배로 늘리고 (여기서 2배라 함은 입력 클럭의 1 클럭 기준으로) 노이즈가 없고 상관없다고 하면 S1, S2모두 출력해야 한다.

위의 2가지 방식으로 OUTPUT을 뽑을 수 있지만 스테이트 머신 비트를 잘보면 SB1이 항상 이때 1이다. 그러므로 출력은 그냥 SB1을 사용하면 로직을 더 적게 사용한 것이 된다.

오늘의 설명에선 입력 클록에 관계해서 쓰지는 않았지만 PAL이나 GAL은 입력클록의 상승에지에서 출력이 변하게 된다. 여기서 고려할 사항은 입력 클록의 주기와 INPUT의 펄스폭과 노이즈의 차이를 고려해 결정한다.

이제부터는 실전연습을 해볼까요?.

[실습예제 1] 디코더 구현하기

가장 기본적으로 많이 사용하는 로직인 74LS138을 간단히 구현해 봅시다. 예를 들어 8031 보드를 설계하는 데, 어드레스 공간 64KBytes를 8 공간으로 분할하는 경우, 입력에 어드레스 핀 A15, A14, A13을 입력으로 사용하면 8개의 각 출력신호가 분할된 어드레스 공간의 select 신호가 됩니다.

```
;---------------------------------- PIN Declarations ---------------
PIN  2          X                   COMBINATORIAL           ; INPUT
PIN  3          Y                   COMBINATORIAL           ; INPUT
PIN  4          Z                   COMBINATORIAL           ; INPUT
PIN  10         GND                                         ; INPUT
PIN  12         A                   COMBINATORIAL           ; OUTPUT
PIN  13         B                   COMBINATORIAL           ; OUTPUT
PIN  14         C                   COMBINATORIAL           ; OUTPUT
PIN  15         D                   COMBINATORIAL           ; OUTPUT
PIN  16         E                   COMBINATORIAL           ; OUTPUT
PIN  17         F                   COMBINATORIAL           ; OUTPUT
PIN  18         G                   COMBINATORIAL           ; OUTPUT
PIN  19         H                   COMBINATORIAL           ; OUTPUT
PIN  20         VCC                                         ; INPUT

;---------------------------------- Boolean Equation Segment ------
EQUATIONS
/A = /X * /Y * /Z ; 0000h - 1FFFh
/B = /X * /Y *  Z ; 2000h - 3FFFh
/C = /X *  Y * /Z ; 4000h - 5FFFh
/D = /X *  Y *  Z ; 6000h - 7FFFh
/E =  X * /Y * /Z ; 8000h - 9FFFh
/F =  X * /Y *  Z ; A000h - BFFFh
/G =  X *  Y * /Z ; C000h - DFFFh
/H =  X *  Y *  Z ; E000h - FFFFh

;---------------------------------- Simulation Segment ------------
SIMULATION
TRACE_ON X Y Z A B C D E F G H
SETF  /X /Y /Z
CHECK /A  B  C  D  E  F  G  H
SETF  /X /Y  Z
CHECK  A /B  C  D  E  F  G  H
SETF  /X  Y /Z
CHECK  A  B /C  D  E  F  G  H
SETF  /X  Y  Z
CHECK  A  B  C /D  E  F  G  H
SETF   X /Y /Z
CHECK  A  B  C  D /E  F  G  H
SETF   X /Y  Z
CHECK  A  B  C  D  E /F  G  H
SETF   X  Y /Z
CHECK  A  B  C  D  E  F /G  H
SETF   X  Y  Z
CHECK  A  B  C  D  E  F  G /H
TRACE_OFF
```

[실습 예제2] REGISTERED 출력 이해

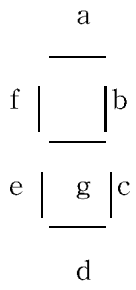
combinatorial 출력과 registered 출력의 차이를 이해한다.

```
;+------------ *------------- *------ *------ *------------- *-------------+
;|                                                                          |
;|                 C & R Output                                             |
;|                                                                          |
;|                                 Written By MinSoon.Hwang                 |
;|                                                 Thu  12-28-95            |
;+------------ *------------- *------ *------ *------------- *-------------+
;------------- * ------------  Declaration Segment  -------------------------+
TITLE           APPLICATION
PATTERN         A
REVISION        1.0
AUTHOR          Minsoon.Hwang [HiTEL ID:makeTTs]
COMPANY         Elsome
DATE            Thu  12-28-95

chip    One_Shot  palce16v8

;------ ** ---- * ------------  pin declarations -------------------------------+
PIN  1          CLK                 COMBINATORIAL           ; INPUT
PIN  2          X                   COMBINATORIAL           ; INPUT
PIN  3          Y                   COMBINATORIAL           ; INPUT
PIN  4          Z                   COMBINATORIAL           ; INPUT
PIN  10         GND                                         ; INPUT
PIN  12         A                   COMBINATORIAL           ; OUTPUT
PIN  13         B                   COMBINATORIAL           ; OUTPUT
PIN  14         C                   COMBINATORIAL           ; OUTPUT
PIN  15         D                   COMBINATORIAL           ; OUTPUT
PIN  16         E                   COMBINATORIAL           ; OUTPUT
PIN  17         F                   COMBINATORIAL           ; OUTPUT
PIN  18         G                   COMBINATORIAL           ; OUTPUT
PIN  19         H                   COMBINATORIAL           ; OUTPUT
PIN  20         VCC                                         ; INPUT

;---------------------------------- Boolean Equation Segment ------
EQUATIONS
/A := /X * /Y * /Z        ; 0000h - 1FFFh
/B := /X * /Y *  Z        ; 2000h - 3FFFh
/C := /X *  Y * /Z        ; 4000h - 5FFFh
/D := /X *  Y *  Z        ; 6000h - 7FFFh
/E :=  X * /Y * /Z        ; 8000h - 9FFFh
/F :=  X * /Y *  Z        ; A000h - BFFFh
/G :=  X *  Y * /Z        ; C000h - DFFFh
/H :=  X *  Y *  Z        ; E000h - FFFFh

;---------------------------------- Simulation Segment ------------
SIMULATION
TRACE_ON X Y Z A B C D E F G H
SETF  /X /Y /Z
CHECK /A  B  C  D  E  F  G  H
SETF  /X /Y  Z
CHECK  A /B  C  D  E  F  G  H
SETF  /X  Y /Z
CHECK  A  B /C  D  E  F  G  H
SETF  /X  Y  Z
```

```
CHECK  A  B  C /D  E  F  G  H
SETF   X /Y /Z
CHECK  A  B  C  D /E  F  G  H
SETF   X /Y  Z
CHECK  A  B  C  D  E /F  G  H
SETF   X  Y /Z
CHECK  A  B  C  D  E  F /G  H
SETF   X  Y  Z
CHECK  A  B  C  D  E  F  G /H
TRACE_OFF
```

[실습 예제3] 7세그먼트 LED 디코더

7-Segment는 아래와 같이 되어 있다.



7 SEGMENT LED는 COMMON-ANODE (공통 Vcc) 형을 사용한다로 가정하고 설계를 해봅시다. 숫자의 표현은 0-9까지로 제한하면 총 10가지수를 가지므로 입력은 최소 4개가 필요해집니다. 그럼 설계해 보실까요.

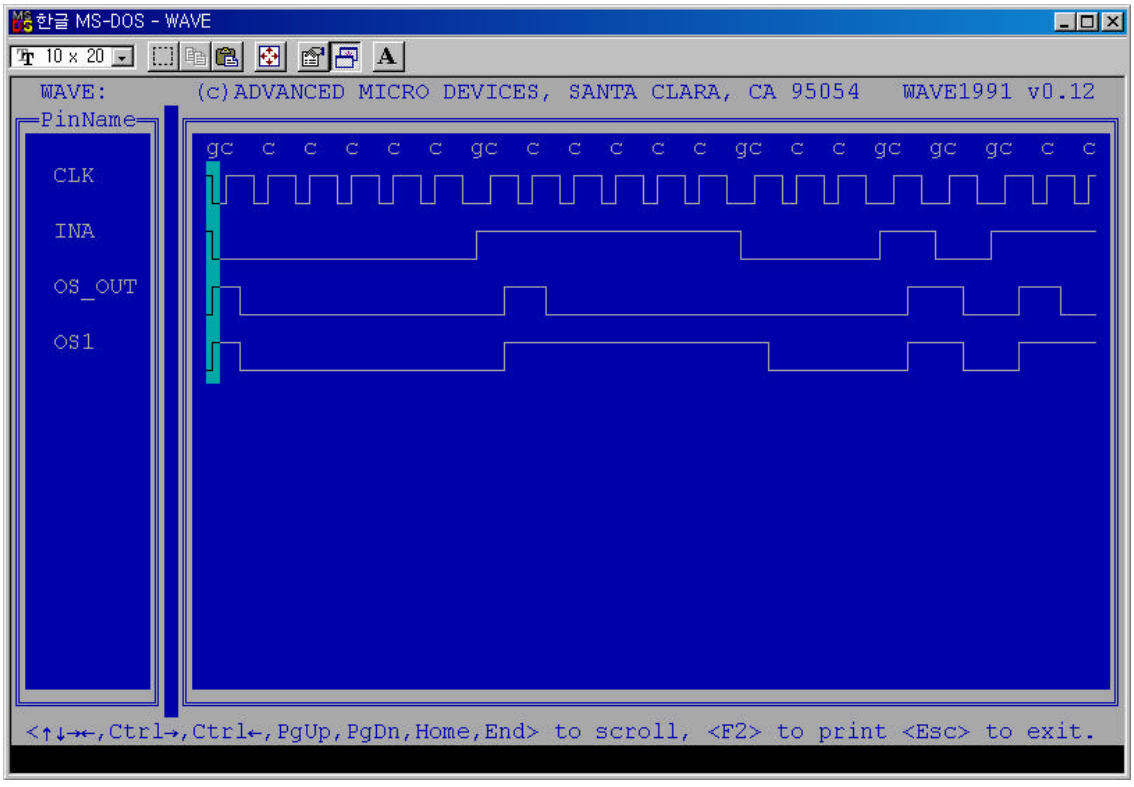
```
SIMULATION
TRACE_ON A0 A1 A2 A3 A B C D E F G
SETF     /A0 /A1 /A2 /A3
SETF     A0 /A1 /A2 /A3
SETF     /A0 A1 /A2 /A3
SETF     A0 A1 /A2 /A3
SETF     /A0 /A1 A2 /A3
SETF     A0 /A1 A2 /A3
SETF     /A0 A1 A2 /A3
SETF     A0 A1 A2 /A3
SETF     /A0 /A1 /A2 A3
SETF     A0 /A1 /A2 A3
TRACE_OFF
```

[실습 예제4] OneShot Trigger 만들기

이번엔 표현을 HDL을 사용하여 구현해봅시다. Low Pulse의 길이를 조절해 봅시다.

```
;PALASM Design Description
;        > Make one shot trigger using Language
;---------------------------------- Declaration Segment ------------
TITLE     APPLICATION
PATTERN   A
REVISION 1.0
AUTHOR    Minsoon.Hwang [HiTEL ID:makeTTs]
COMPANY   Elsome.
DATE      08/01/95

chip  name        palce16v8

;---------------------------------- pin declarations ---------------
pin  1            clk                combinatorial ; input
pin  2            ina                combinatorial ; input

pin  10  gnd
pin  20  vcc

pin  19  os_out             registered      ; output
pin  18  os1                registered      ; output

;---------------------------------- boolean equation segment ------
equations

if( /ina ) then begin
   os_out := 0;
   os1 := 0;
end
else begin
   if( /os1 ) then begin
          os_out := 1
          os1 := 1;
   end
   else begin
          os_out := 0;
          os1 := 1
   end
end
;---------------------------------- simulation segment ------------
simulation
trace_on  clk ina os_out os1

setf /ina /clk

clockf clk
clockf clk

for i := 1 to 100 do begin
   clockf clk

   if( i = 4 ) then begin
          setf   ina
   end
```

```
    if( i = 10 ) then begin
            setf /ina
    end

    if( i = 13 ) then begin
            setf ina
    end
    if( i = 14 ) then begin
            setf /ina
    end

    if( i = 15 ) then begin
            setf ina
    end
    if( i = 30 ) then begin
            setf /ina
    end
end
trace_off
;-----------------------------------------------------------------
```



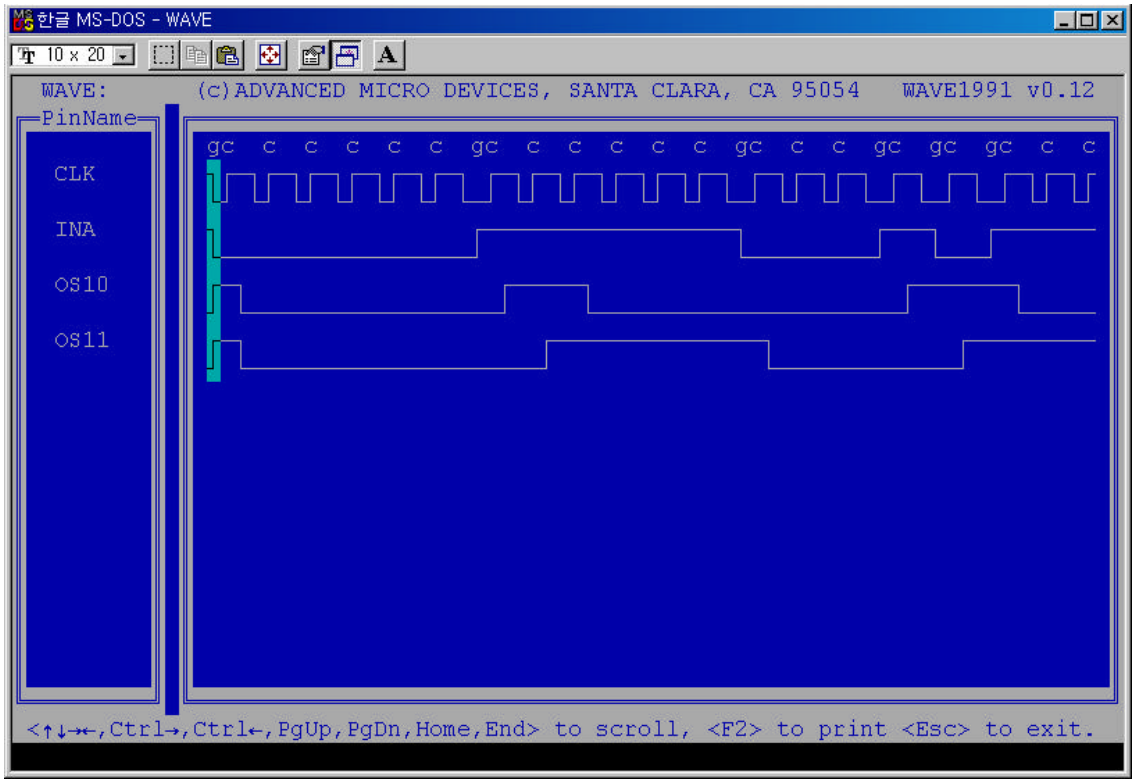
[실습 예제5] 인터럽트 펄스 구현하기

앞의 예제에서 한 것을 기초로 하여 인터럽트 펄스를 구현해 봅시다. 입력신호의 Rising edge를 Detect하여 High One shot Pulse를 발생하는 것입니다.

```
;PALASM Design Description
;       -> Make 2 cycle one shot trigger using Language for c30 & c31
;---------------------------------- Declaration Segment ------------
TITLE     APPLICATION
PATTERN   A
REVISION 1.0
AUTHOR    Minsoon.Hwang [HiTEL ID:makeTTs]
COMPANY   Elsome.
DATE      08/01/95

chip  name        palce16v8
;---------------------------------- pin declarations ---------------
pin  1            clk              combinatorial ; input
pin  2            ina              combinatorial ; input
pin  3            pin3             combinatorial ; input

pin  10  gnd
pin  20  vcc

pin  19  os10             registered       ; output
pin  18  os11             registered       ; output

;---------------------------------- boolean equation segment ------
equations

if( /ina ) then begin
   if( /os10 * /os11 ) then begin
          os10 := 0
          os11 := 0
   end
   if( /os10 * os11 ) then begin
          os10 := 0
          os11 := 0
   end
   if( os10 * os11 ) then begin
          os10 := 0
          os11 := 0
   end
   if( os10 * /os11 ) then begin
          os10 := 1
          os11 := 1
   end
end
else begin
   if( /os10 * /os11 ) then begin
          os10 := 1
          os11 := 0
   end
   if( /os10 * os11 ) then begin
          os10 := 0
          os11 := 1
   end
   if( os10 * /os11 ) then begin
```

```
          os10 := 1
          os11 := 1
   end
   if( os10 * os11 ) then begin
          os10 := 0
          os11 := 1
   end
end
;---------------------------------- simulation segment ------------
simulation
trace_on  clk ina os10 os11

setf /ina /clk

clockf clk
clockf clk

for i := 1 to 100 do begin
   clockf clk
;
   if( i = 4 ) then begin
          setf   ina
   end
   if( i = 10 ) then begin
          setf /ina
   end
;
   if( i = 13 ) then begin
          setf ina
   end
   if( i = 14 ) then begin
          setf /ina
   end
;
   if( i = 15 ) then begin
          setf ina
   end
   if( i = 30 ) then begin
          setf /ina
   end
;
   if( i = 31 ) then begin
          setf ina
   end
   if( i = 32 ) then begin
          setf /ina
   end
;
   if( i = 33 ) then begin
          setf ina
   end
   if( i = 34 ) then begin
          setf /ina
   end
;
   if( i = 35 ) then begin
          setf ina
   end
   if( i = 36 ) then begin
          setf /ina
```

```
   end
;
   if( i = 37 ) then begin
          setf ina
   end
   if( i = 38 ) then begin
          setf /ina
   end
;
   if( i = 39 ) then begin
          setf ina
   end
   if( i = 40 ) then begin
          setf /ina
   end
end
trace_off
;-----------------------------------------------------------------
```



[실습 예제6] 스텝모터 구동신호 발생

1) 모터 구동 회로

이번에는 Full Step Stepping Motor Controller를 PAL20V8에 넣는 것을 알아본다. 먼저 COMBINATORIAL모드와 REGISTERED모드 두 가지를 알아보자. 먼저 전자는 CLOCK에 상관없이 바로바로 입력의 연산에 의해 출력 핀의 상태가 결정되는 모드이다. 그리고 후자는 CLOCK이 LOW에서 HIGH로 변할 때, POSITIVE EDGE TRIG에서 출력 핀의 다음 상태가 결정되는 모드이다. PAL20V8은 이 두 가지를 다 지원하고, 각각 사용이 가능하며 혼용도 가능하다. 전자는 앞 시간에 한 Memory Decoder에서 CLOCK등에 상관없이 A15 A14 A13 A12 A11 A10 A9 A8 의 변화에 따라 바로 바로 변하는 것을 예로 들 수 있다. 그리고 REGISTERED모드는 프로그램 구현 방법이 다양하다. 이번에 하는 방식이 가장 단순한 방식이다. 그리고 다른 방식은DIGITAL에서 있듯이 밀리머신, 무어머신 등을 들 수 있겠다.

PAL20V8의 1번 핀은, 출력 핀에 연결된 D-FF의 Clock핀에 연결될 수 있다. 프로그램을 보면 알겠지만 "PIN 1 CLOCK COMBINATORIAL"이다.
기본은 COMBINATORIAL이므로 위의 것은 "PIN 1 CLOCK" 이라고 써도 된다. 그리고 Clock은 내부적으로 1번 핀 하나만 연결할 수 있다. 다시 말해 출력 핀 A의 Clock은 CLKA에, 출력 핀 B의 Clock은 CLKB에 연결을 하지 못한다. 기본으로 Clock핀은 1번 핀에만 붙을 수 있는 것이다. 주의할 사항이다. 물론 CLKA와 CLKB의 PIN 번호를 "?"로 하면 시뮬레이션은 되지만 PAL에 구울 수 있는 JEDEC파일은 만들 수 없다. DIRA는 말 그대로 방향이다. 그리고 GND와 VCC는 안 적어 줘도 알아서 처리해준다. A0..3은 실제 파형이 나가는 출력단자이다.

EQUATIONS 블럭을 보면 /ENABLE이 되면 A0..3은 모두 LOW가 된다. 그리고 이 상태에서 ENABLE로 되면 /A0 * /A1 * /A2 * /A3 에 의해 A0 와 A2가 HIGH로 A1과 A3는 /A0, /A2인 LOW로 된다. 초기화가 자동으로 되는 것이다. 단계 단계 넘어가는 것은 시뮬레이션을 돌려 파형을 보면서 해보라. 그리고 조금의 팁을 말하자면 PIN정의에서 출력핀의 MODE를 정하지 않으면 EQUATIONS블럭에서의 연산자에 의해 자동으로 그 MODE가 결정된다.

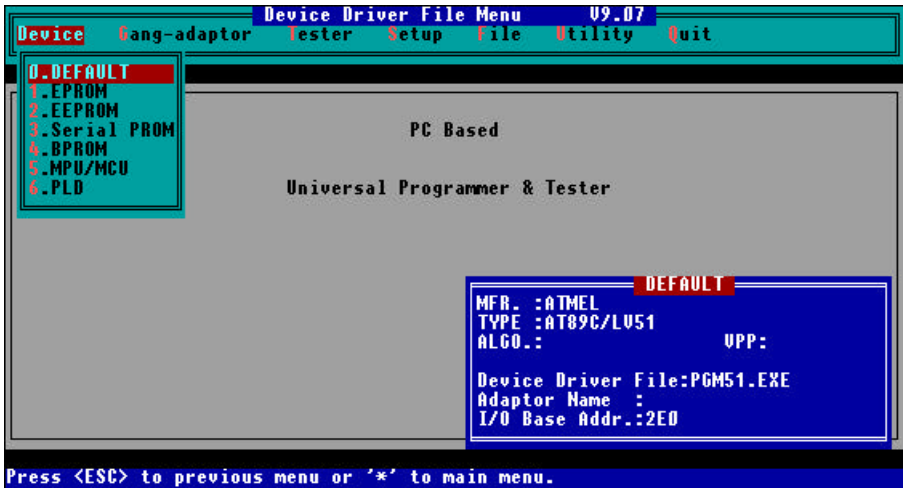
```
TITLE    Stepper Controller
PATTERN  XTAL297
REVISION 1.0
AUTHOR   ChoiYH73 ( Choi Yong Hee 92 Ajou Univ. )
COMPANY  X-TAL CONTROL ENG. AJOU UNIV.
DATE     11/29/94

CHIP XTAL297 PALCE20V8

;/////////////////////////////////////////////////////////////
;                DEFINE PIN NAME
;/////////////////////////////////////////////////////////////
PIN  1          CLOCK           COMBINATORIAL
PIN  4          DIRA            COMBINATORIAL
PIN  6         /ENABLE          COMBINATORIAL
PIN  12         GND
PIN  19         A0              REGISTERED
```

```
PIN  20         A1              REGISTERED
PIN  21         A2              REGISTERED
PIN  22         A3              REGISTERED
PIN  24         VCC

;/////////////////////////////////////////////////////////////
;                  EQUATIONS
;/////////////////////////////////////////////////////////////
EQUATIONS

    A0 :=
                        A2       *  DIRA * ENABLE
          +                    A3 * /DIRA * ENABLE
          + /A0 * /A1 * /A2 * /A3         * ENABLE
    A1 :=
                             A3 *  DIRA * ENABLE
          +              A2       * /DIRA * ENABLE
    A2 :=
                  A1             *  DIRA * ENABLE
          +  A0                  * /DIRA * ENABLE
          + /A0 * /A1 * /A2 * /A3         * ENABLE
    A3 :=
             A0                  *  DIRA * ENABLE
          +        A1             * /DIRA * ENABLE

;/////////////////////////////////////////////////////////////
;                 SIMULATION
;/////////////////////////////////////////////////////////////
SIMULATION

TRACE_ON ENABLE DIRA A0 A1 A2 A3

SETF /ENABLE DIRA
CLOCKF CLOCK
SETF ENABLE
CLOCKF CLOCK

FOR I := 1 TO 10 DO
BEGIN
    CLOCKF CLOCK
END
SETF /DIRA
FOR I := 1 TO 10 DO
BEGIN
    CLOCKF CLOCK
END
SETF DIRA /ENABLE
FOR I := 1 TO 10 DO
BEGIN
    CLOCKF CLOCK
END
SETF /DIRA
FOR I := 1 TO 10 DO
BEGIN
    CLOCKF CLOCK
END

TRACE_OFF
```

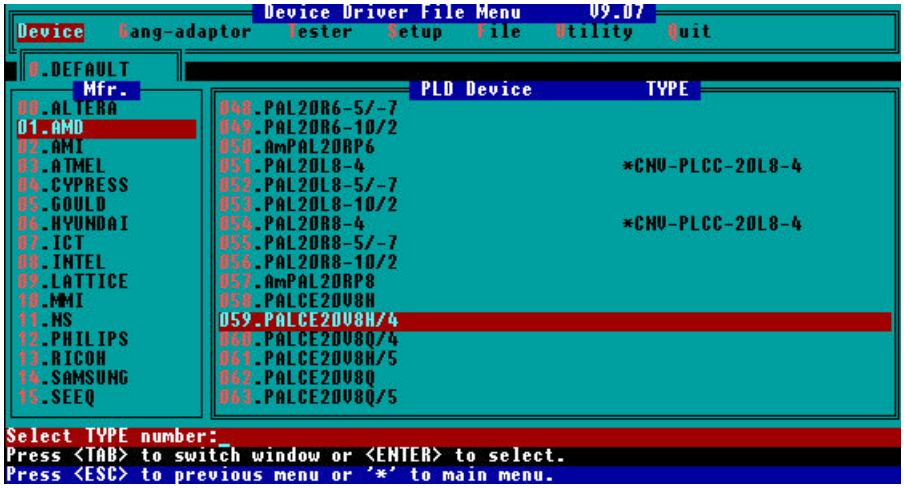
[참고] PLD 프로그래밍 - UNI돌이(ALL-03A 호환) 사용법

1 c:\uni돌이 폴더에서 Access.exe 프로그램을 실행시키면 다음과 같은 초기화면이 뜬다.

2 Device 메뉴를 선택하면 다음 화면이 나타난다.



3 6. PLD를 선택하고 Mfr.은 1번 AMD를 Device Type은 59번 PALCE20V8H/4를 선택한다.



4 그러면 다음과 같은 화면이 나오게 된다. 메뉴선택은 해당 숫자나 영문자를 누르면 된다. 먼저 위에 숫자로 메뉴가 나오는 것 중에서 2. Load JEDEC File 메뉴로 들어 간다.
여기서 우리가 전에 작업했던 PALASM\WORK\ 에 있는 *.JED 파일을 불러온다.

```
-GAL20V8H Universal   Programmer
MODEL: PC Based
GAL2
-----------  Main Menu  -------------
1. DOS SHELL
2. Load JEDEC file to buffer
3. Save buffer to disk
4. Edit buffer    7. Display buffer
5. Change I/O base address

T. Type select    M. Mfr. select

B. Blank check    D. Display
P. Program        A. Auto(B&P&V&S)
R. Read           V. Verify
E. Erase          S. Security fuse blow
Q. Quit
---------------------------------------
Select function ?
```

5 간단히 오토(A를 누른다.)를 실행한다. A. Auto(B&P&V&S)를 실행하면 Blank Check -> Program -> Verify -> Security fuse blow 까지 알아서 척척 해준다. 블랭크 검사해서 내용이 들어 있으면 알아서 지우니까 편리하다.

6 Q. Quit 하여 ROM Writer 프로그램에서 빠져나가면 사용이 끝난다.