실험 강의 (2):
VHDL 설계

이 훈 재

E-mail=hjlee@kyungwoon.ac.kr

Ex1)Half Adder :
-Text Editor

*Menu : Enter Symbol / Set Project to current file

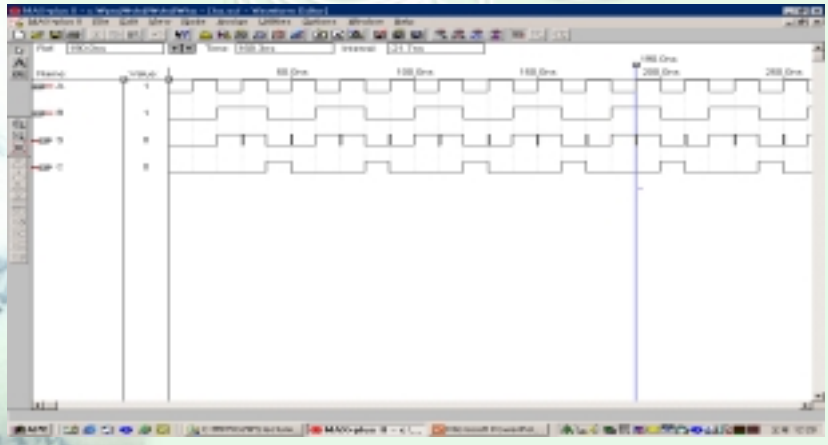
- HA.VHD

```
library IEEE;
use IEEE.std_logic_1164.all;
use
      IEEE.std_logic_unsigned
      .all;
use IEEE.std_logic_arith.all;


entity HA is
port ( A : in std_logic;
         B : in std_logic;
         S : out std_logic;
         C : out std_logic);
end HA;
```

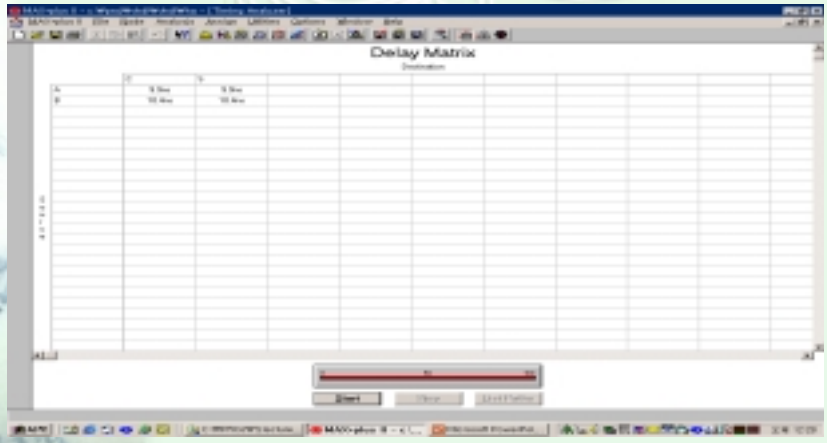
```
architecture EX1 of HA is
begin

 - - half adder
   S <= A xor B;
   C <= A and B;

end EX1;
```

Ex1)Half Adder
- Waveform Editor

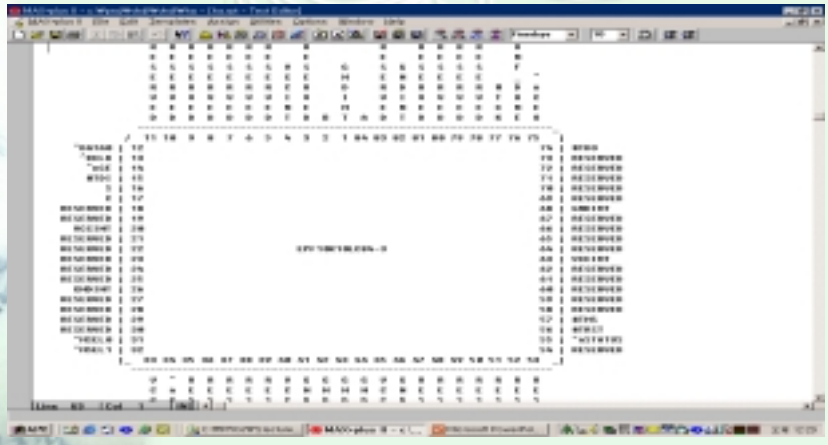
*Menu : Insert Node → List



Ex1)Half Adder
- Timing Analyzer



Ex1)Half Adder
- Hierarchy Display → RPT



Ex2)Full Adder -1
- Text Editor

- FA1.VHD

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;


entity FA1 is
    port ( A      : in std_logic;
           B, Cin : in std_logic;
           S      : out std_logic;
           Cout   : out std_logic);
end FA1;
```

```
use IEEE.std_logic_1164.all;
architecture EX2 of FA1 is
signal HA1_S, HA1_C, HA2_S, HA2_C : std_logic;
begin

 - - half adder
   HA1_S <= B xor Cin;
   HA1_C <= B and Cin;

 - - half adder
   HA2_S <= A xor HA1_S;
   HA2_C <= A and HA1_S;

 - - full adder
   S <= HA2_S;
   Cout <= HA1_C or HA2_C;

end EX2;
```

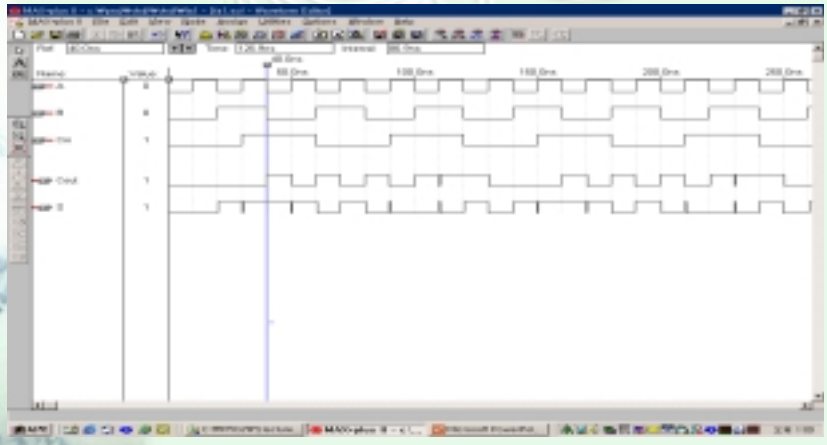
Ex2)Full Adder -1
- Text Editor

- FA2.VHD

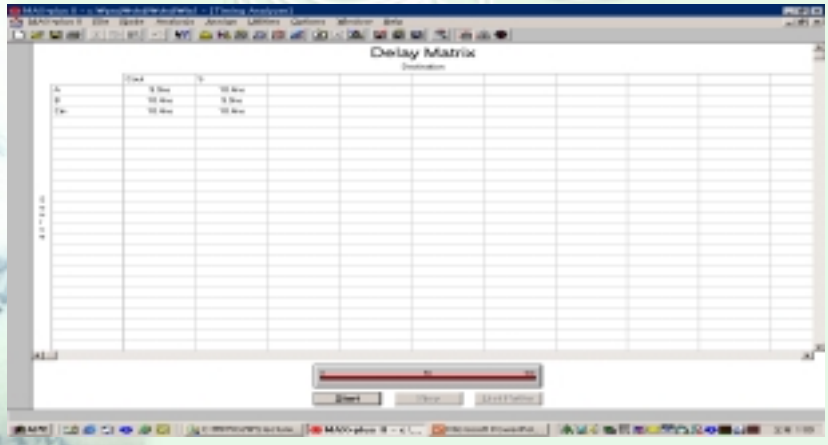
```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;

entity FA2 is
port (
    A   : in  std_logic;
    B   : in  std_logic;
    Cin  : in  std_logic;
    S   : out std_logic;
    Cout : out std_logic
    );
end FA2;
```

```
architecture EX3 of FA2 is
signal TMP_XOR : std_logic := '0';
begin

process(A, B, Cin, TMP_XOR)
begin
    TMP_XOR <= A xor B;
    S       <= TMP_XOR xor Cin;
    Cout    <= (TMP_XOR and Cin) or (A and B);
end process;

end EX3;
```

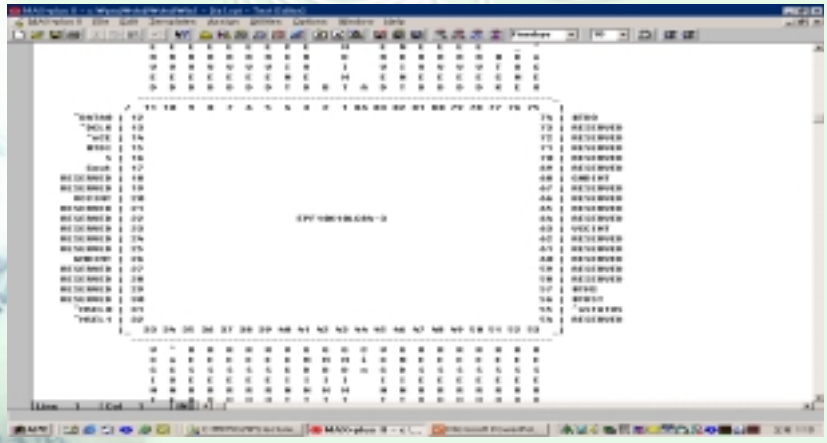
Ex2)Full Adder -1
- Waveform Editor



Ex2)Full Adder -1
- Timing Analyzer



Ex2)Full Adder -1
- Hierarchy Display → RPT



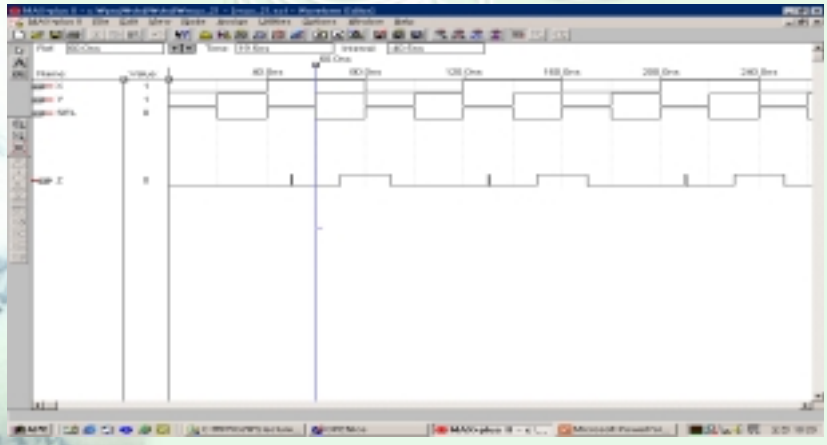
Ex3)MUX_21
- Text Editor

- MUX_21.VHD

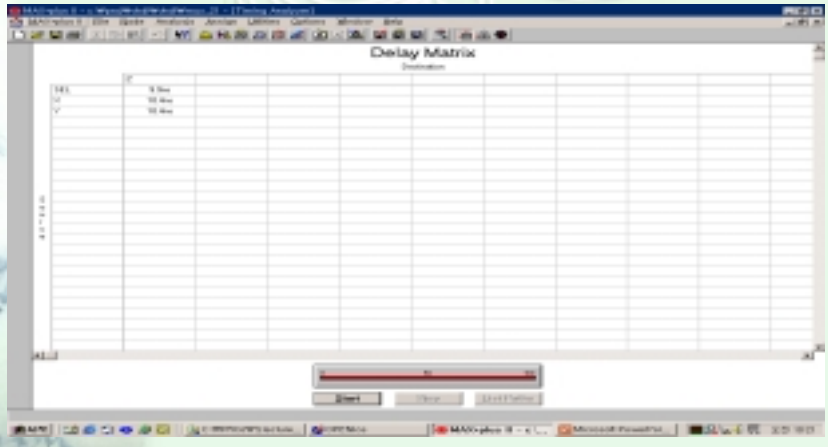
```
library ieee;
use ieee.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;

entity MUX_21 is
      port ( SEL  : in std_logic;
             X, Y : in std_logic;
             Z    : out std_logic);
end MUX_21;
```

```
architecture EX3 of MUX_21 is
begin
            process(X, Y, SEL)
            begin
                        if SEL = '0' then

            Z <= X;
                        else

            Z <= Y;
                        end if;
            end process;
end EX3;
```

Ex3)MUX_21
- Waveform Editor



Ex3)MUX_21
- Timing Analyzer



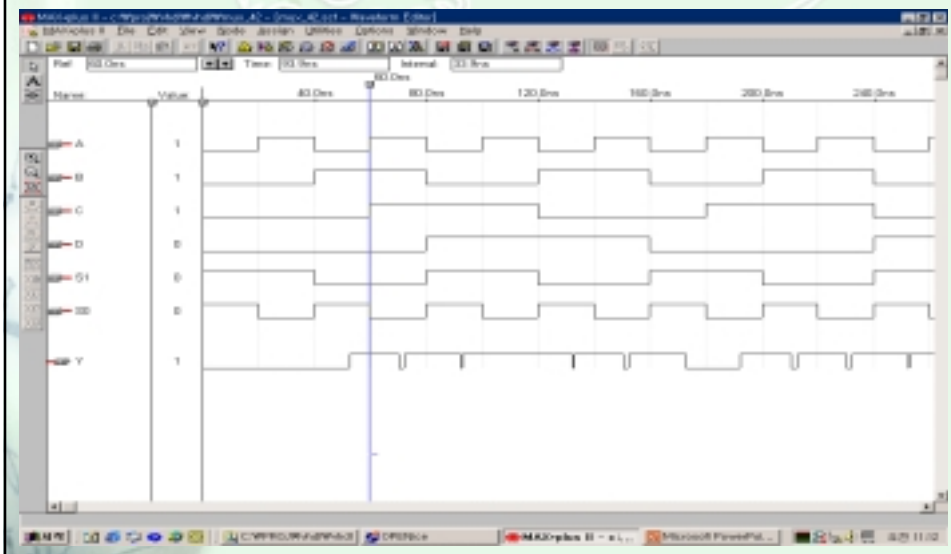
Ex4)MUX_42 : Text Editor

- MUX_42.VHD

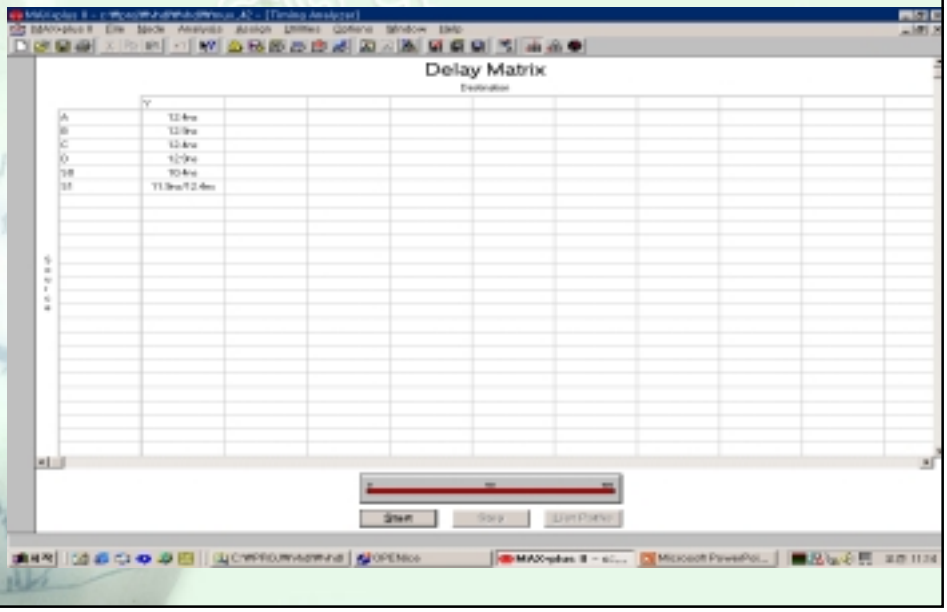
```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;

entity MUX_42 is
port (
   A  : in  std_logic;
   B  : in  std_logic;
   C  : in  std_logic;
   D  : in  std_logic;
   S0 : in  std_logic;
   S1 : in  std_logic;
   Y  : out std_logic
   );
end MUX_42;
```

```
architecture EX4 of MUX_42 is
begin
   Y <= (A and (not S1) and (not S0))
or
        (B and (not S1) and S0      ) or
        (C and S1 and (not S0)      ) or
        (D and S1 and S0            );

end EX4;
```

Ex4)MUX_42 : Waveform Editor



Ex4)MUX_42 : Timing Analyzer



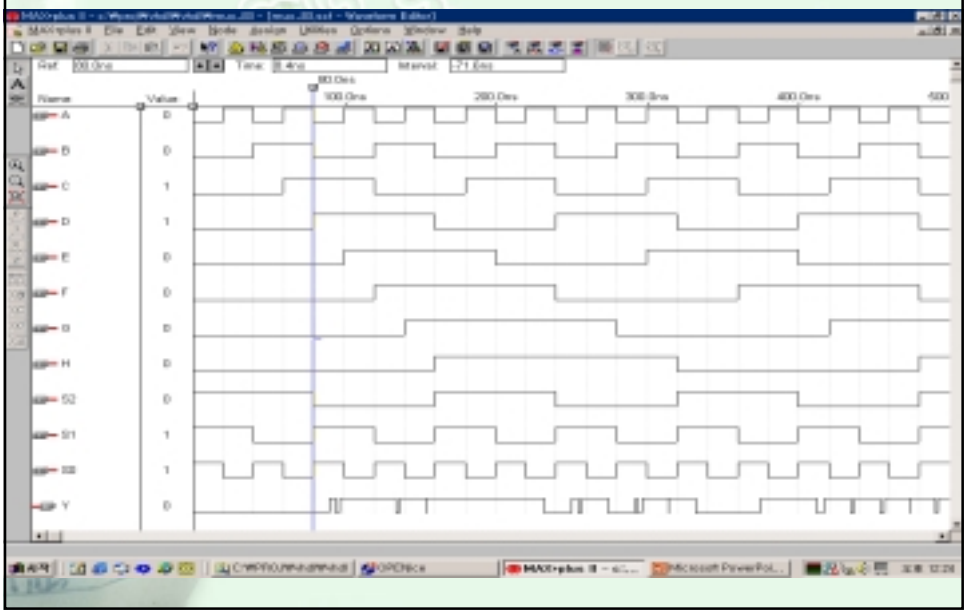
Ex5)MUX_83 : Text Editor

- MUX_83.VHD

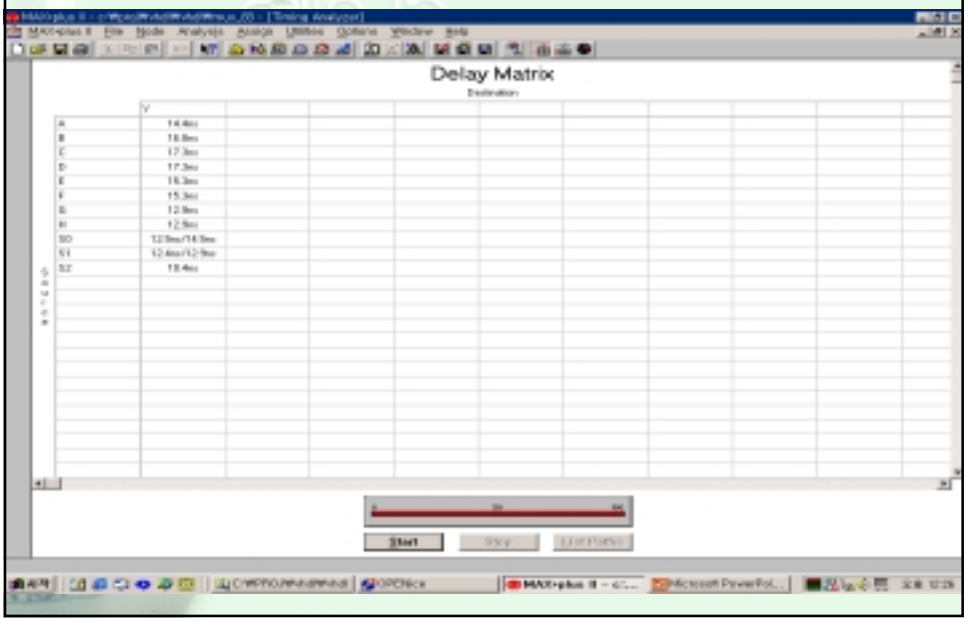
```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;

entity MUX_83 is
   port (A : in std_logic;
         B : in std_logic;
         C : in std_logic;
         D : in std_logic;
         E : in std_logic;
         F : in std_logic;
         G : in std_logic;
         H : in std_logic;
         S0 : in std_logic;
         S1 : in std_logic;
         S2 : in std_logic;
         Y : out std_logic);
end MUX_83;
```

```
architecture EX5 of MUX_83 is
begin
  process(A,B,C,D,E,F,G,H,S0,S1,S2)
  begin
    if (S2 = '0' and S1 = '0' and S0 = '0') then
       Y <= A;
    elsif (S2 = '0' and S1 = '0' and S0 = '1') then
       Y <= B;
    elsif (S2 = '0' and S1 = '1' and S0 = '0') then
       Y <= C;
    elsif (S2 = '0' and S1 = '1' and S0 = '1') then
       Y <= D;
    elsif (S2 = '1' and S1 = '0' and S0 = '0') then
       Y <= E;
    elsif (S2 = '1' and S1 = '0' and S0 = '1') then
       Y <= F;
    elsif (S2 = '1' and S1 = '1' and S0 = '0') then
       Y <= G;
    else
       Y <= H;
    end if;

  end process;
end EX5;
```

Ex5)MUX_83 : Waveform Editor



Ex5)MUX_83 : Timing Analyzer

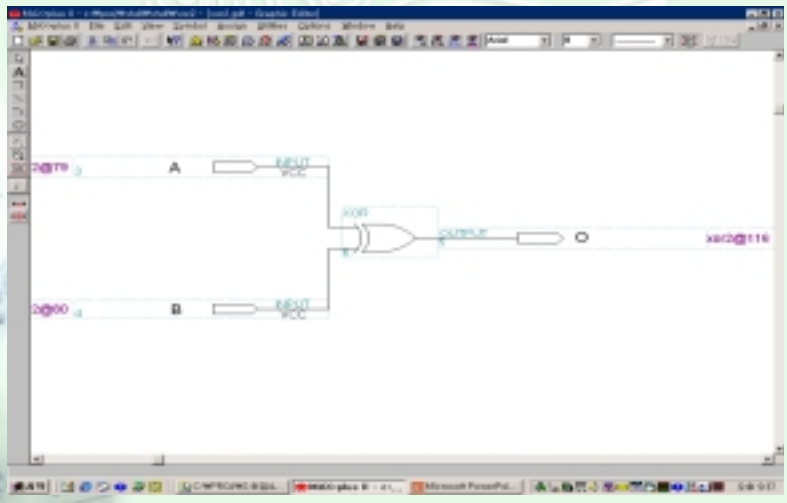


HDB-DTK-240 KIT 사용법

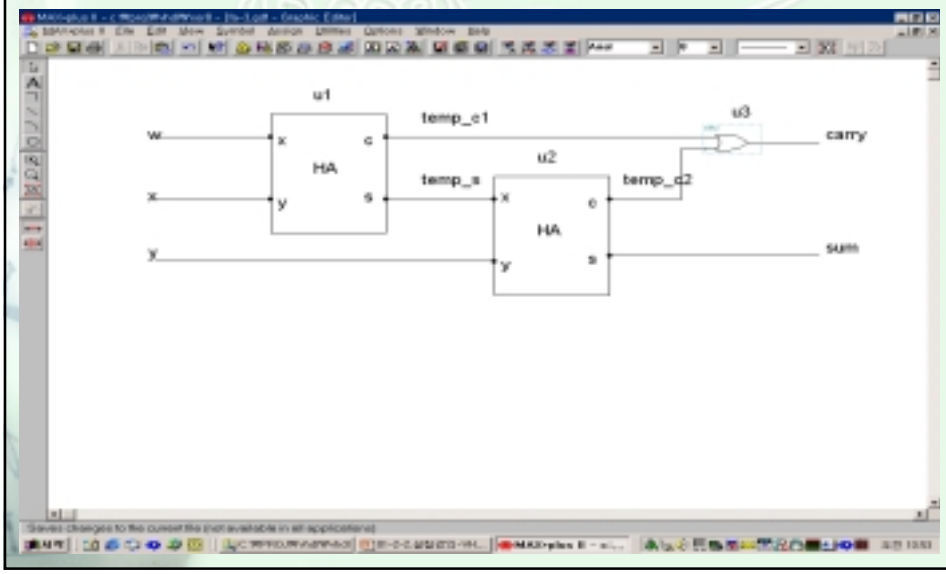
- Training Kit 사용법
  - 한백전자 트레이닝 키트
  - CPLD device = Flex10K family
    EPF10K20RC240-3
  - Input Device : Keypad, SW
  - Output Device : 7-segment display, LCD,LED
  - Connection : PC=Parallel Port - - RJ-45=KIT
  - Mode = ByteBlaster(Download=configure)

HDB-DTK-240 KIT 사용법(2)

- XOR 회로 테스트(Key_1=79;Key_2=80;LED0=116)



Ex3) 2-bit Real Adder : block



Ex3) 2 -bit Real Adder : 설계 방법

- Emulation 확인 방법

- 1) HA.VHD → design & compile
- 2) OR_2.VHD → design & compile
- 3) FA3.VHD → set project
- 4) FA3.VHD → design & compile
- 5) FA3.VHD → pin assignment & compile
- w=pin79 → KEY_0;
- x=pin 80 → KEY_1;
- y=pin 81 → KEY_2;
- carry=pin116 → LED0;
- sum=pin 117 → LED1;
- 6) FA3.VHD → waveform editor
- 7) FA3.VHD → simulate
- 8) FA3.VHD → programming (configure)
- 9) emulation
- 키 한 개 누를 때 LED0=OFF, LED1=ON (01)
- 키 두 개 누를 때 LED0=ON , LED1=OFF(10)
- 키 세 개 누를 때 LED0=ON , LED1=ON(11)

Ex3) 2 -bit Real Adder : FA3.VHD(1)

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;

entity FA3 is
      port(w,x,y     : in std_logic;
           carry,sum : out std_logic);
end FA3;
```

```
architecture EX3 of FA3 is
      signal temp_c1, temp_c2, temp_s : std_logic;
      component HA port(x,y          : in std_logic;
                        c,s          : out std_logic);
      end component;

      component OR_2 port(a,b        : in std_logic;
                          c          : out std_logic);
      end component;

begin
      u1: HA port map(w,x,temp_c1,temp_s);
      u2: HA port map(temp_s,y,temp_c2,sum);
entity FA3 is
      port(w,x,y                     : in std_logic;
           carry,sum                 : out std_logic);
end FA3;
```

Ex3) 2 -bit Real Adder : HA.VHD(2)

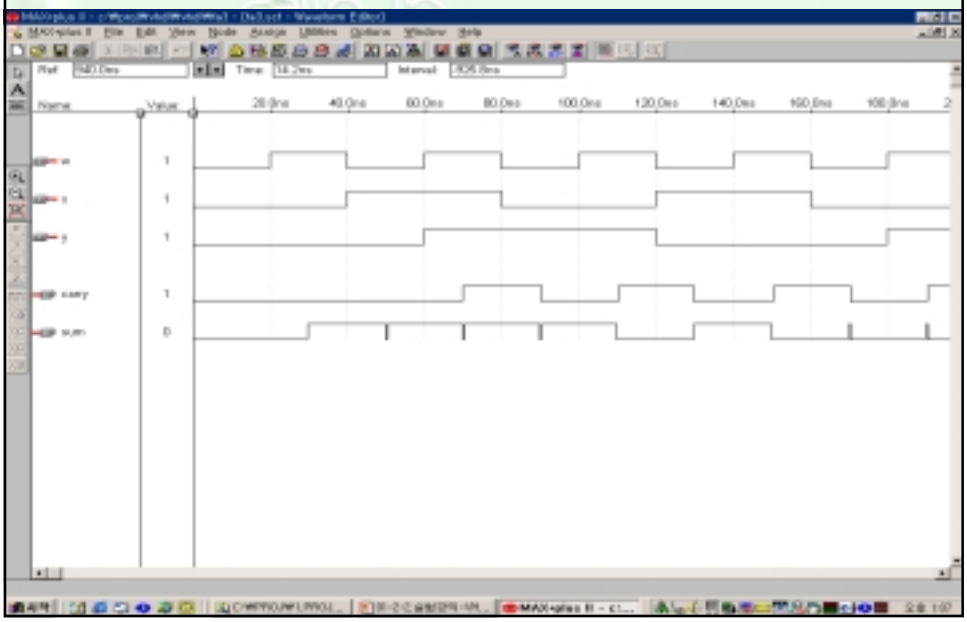
```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;

entity HA is
        port(x,y        :in std_logic;
             c,s        :out std_logic);
end HA;
```

```
architecture EX3 of HA is
begin
    p1: process(x,y)
    begin
        if(x='1') and (y='1') then
            c<='1';
        else
            c<='0';
        end if;
    end process;

    p2: process(x,y)
    begin
        if(x=y) then
            s<='0';
        else
            s<='1';
        end if;
    end process;
end EX3;
```

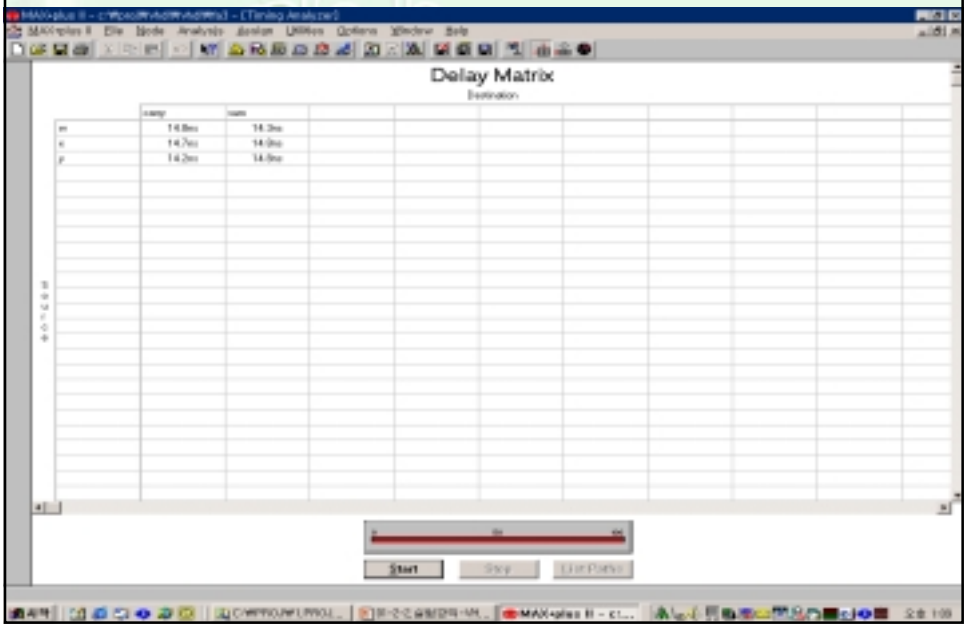
Ex3) 2 -bit Real Adder: OR_2.VHD(3)

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;

entity OR_2 is
        port(a,b        :in std_logic;
             c          :out std_logic);
end OR_2;

architecture EX3 of OR_2 is

begin
    c <= a or b;
end EX3;
```

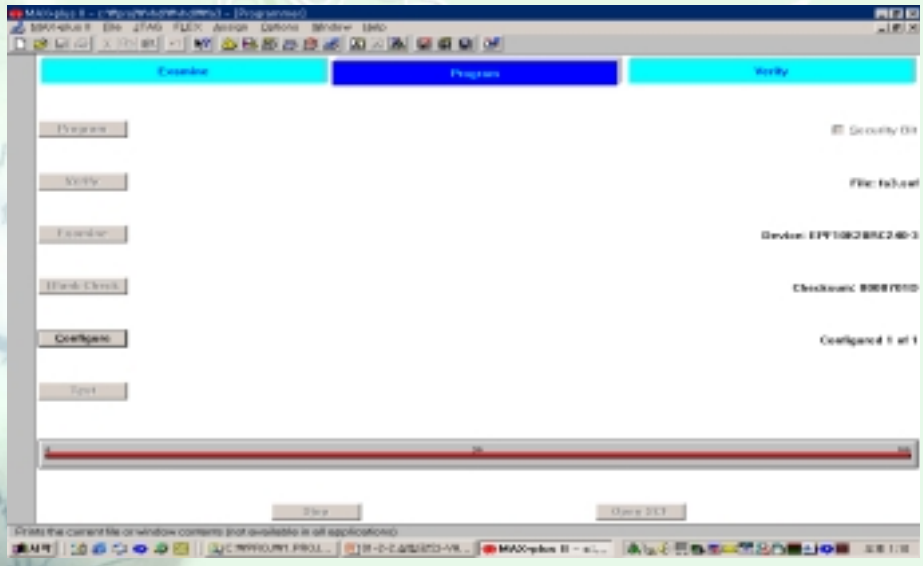
Ex3) 2 -bit Real Adder : Simulation



Ex3) 2 -bit Real Adder :Timing Analyzer

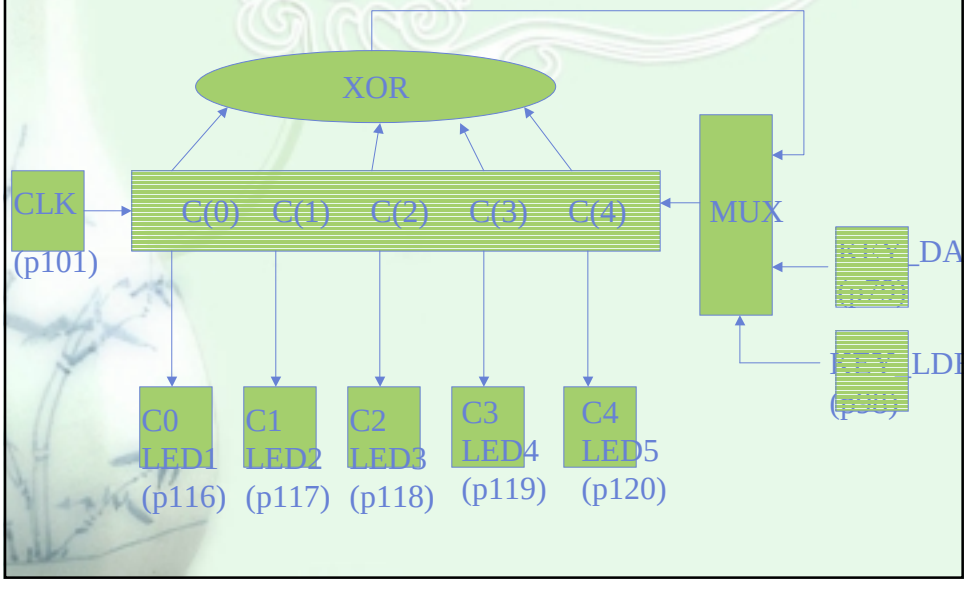


Ex3) 2 -bit Real Adder : download



Ex4) 5 -bit LFSR5 : design

- LFSR5 is $p(x) = x^5 + x^4 + x^3 + x^2 + 1$



Ex4) 5 -bit LFSR5 :LFSR5.VHD(1)

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;

entity LFSR5 is
port (
    CLK       : in std_logic;
    KEY_DATA  : in std_logic;
    KEY_LDEN  : in std_logic;
    C0        : out std_logic;
    C1        : out std_logic;
    C2        : out std_logic;
    C3        : out std_logic;
    C4        : out std_logic
    );
end LFSR5 ;
```

```
architecture ARCH_LFSR5  of LFSR5  is

    signal C        : unsigned(4 downto 0);
    signal TMP_FEED : std_logic := '0';
    signal FEED_IN  : std_logic := '0';
begin

C0 <= C(0);
C1 <= C(1);
C2 <= C(2);
C3 <= C(3);
C4 <= C(4);

FEED_IN <= C(0) xor C(2) xor C(3) xor C(4);
TMP_FEED <= (KEY_LDEN and KEY_DATA) or (not(KEY_LDEN) and FEED_IN);
```

Ex4) 5 -bit LFSR5 :LFSR5.VHD(2)

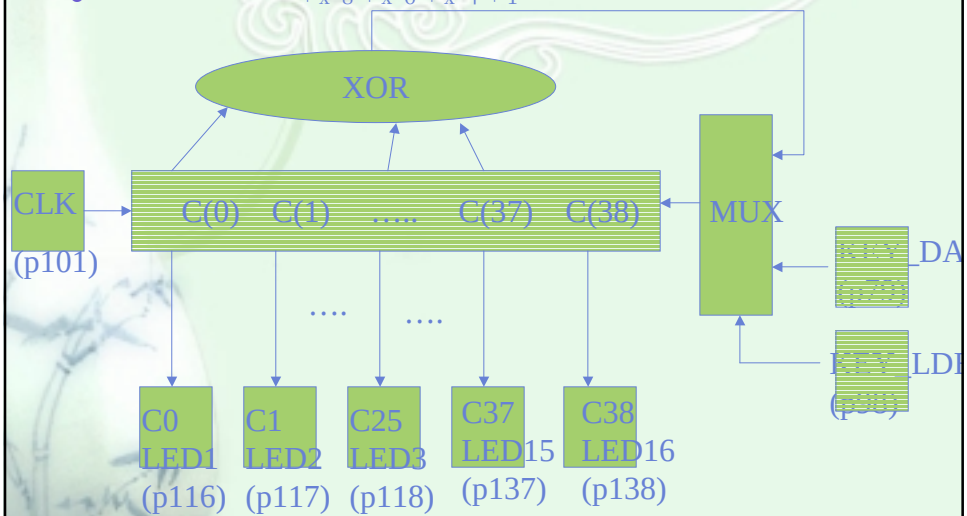
```
process(CLK)
begin
    if CLK'event and CLK = '1' then
--        C <= TMP_FEED & C(4 downto 1);
        C(3 downto 0) <= C(4 downto 1);
C(4) <= TMP_FEED;
--        C(0)<=C(1); C(1)<=C(2); C(2) <=
C(3); C(3) <= C(4);
--        C(4) <= TMP_FEED;
    end if;
end process;
end ARCH_LFSR5 ;
```

Ex4) LFSR5: HDB-DTK-240 KIT

- Input pin
  - KEY_LDEN → KEYPAD(0)=pin98
  - KEY_DATA → KEYPAD(1)=pin79
  - CLK → KEYPAD(D)=pin101
- Output pin
  - C0 → LED1 = pin116
  - C1 → LED2 = pin117
  - C2 → LED3 = pin118
  - C3 → LED4 = pin119
  - C4 → LED5 = pin120
- Emulation test
  - If KEYPAD(01)= 11 → KEYPAD(D) 1 strike = CLK
  - LED12345 = 00001
  - If KEYPAD(01)= 11 → KEYPAD(D) 5 strikes = CLK
  - LED12345 = 11111
  - if KEYPAD(01)= 00 → KEYPAD(D) 1 strike = CLK
  - LED12345 = 11110 (1-bit left shifted)
  - if KEYPAD(01)= 00 → KEYPAD(D) 31 strikes = CLK
  - LED12345 = 11111 (31-bit left shifted: period=2^5 -1)

Ex5) 39-bit LFSR39 : design

- LFSR5 is p(x) = x^39 + x^37 + x^25 + x^24 + x^22
- + x^8 + x^6 + x^4 + 1



Ex5)39 -bit LFSR39 :LFSR39.VHD(1)

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;

entity LFSR39 is
port (
    CLK      : in std_logic;
    KEY_DATA  : in std_logic;
    KEY_LDEN  : in std_logic;
    C0       : out std_logic;
    C1       : out std_logic;
    C25      : out std_logic;
    C26      : out std_logic;
    C27      : out std_logic;
    C28      : out std_logic;
    C29      : out std_logic;
    C30      : out std_logic;
    C31      : out std_logic;
    C32      : out std_logic;
    C33      : out std_logic;
    C34      : out std_logic;
    C35      : out std_logic;
    C36      : out std_logic;
    C37      : out std_logic;
    C38      : out std_logic);
end LFSR39 ;
```

```
architecture ARCH_LFSR39  of LFSR39  is
  signal C        : unsigned(38 downto 0);
  signal TMP_FEED : std_logic := '0';
  signal FEED_IN  : std_logic := '0';
  begin
    C0  <= C(0);
    C1  <= C(1);
    C25 <= C(25);
    C26 <= C(26);
    C27 <= C(27);
    C28 <= C(28);
    C29 <= C(29);
    C30 <= C(30);
    C31 <= C(31);
    C32 <= C(32);
    C33 <= C(33);
    C34 <= C(34);
    C35 <= C(35);
    C36 <= C(36);
    C37 <= C(37);
    C38 <= C(38);
```

Ex5)39 -bit LFSR39 :LFSR39.VHD(2)

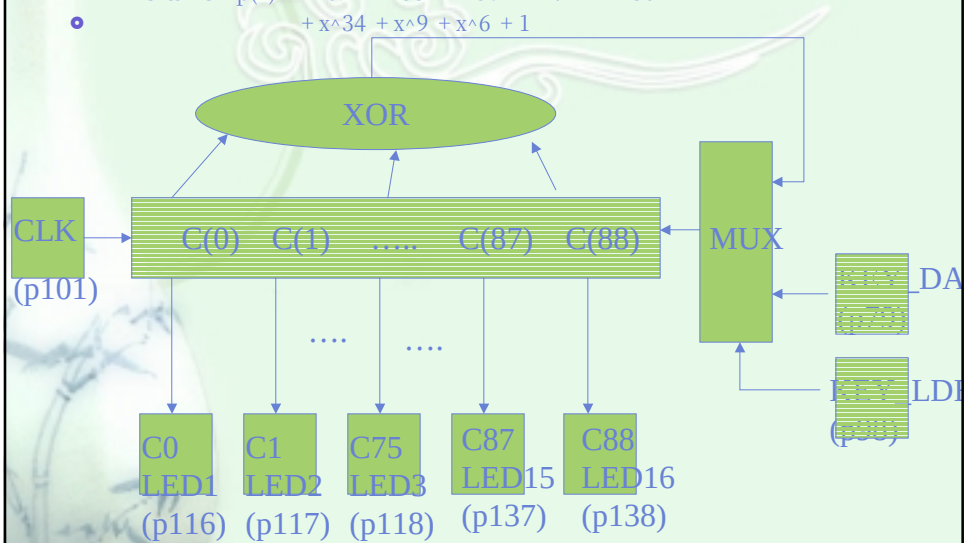
```
FEED_IN <= C(0) xor C(4) xor C(6) xor
C(8) xor C(22) xor C(24) xor C(25) xor
C(37);
    TMP_FEED <= (KEY_LDEN and
KEY_DATA) or (not(KEY_LDEN) and
FEED_IN);

    process(CLK)
    begin
      if CLK'event and CLK = '1' then
        C(37 downto 0) <= C(38 downto 1);
        C(38) <= TMP_FEED;
      end if;
    end process;
end ARCH_LFSR39;
```

Ex5) LFSR39: HDB-DTK-240 KIT

- Input pin
  - KEY_LDEN → KEYPAD(0)=pin98
  - KEY_DATA → KEYPAD(1)=pin79
  - CLK → KEYPAD(D)=pin101
- Output pin
  - C0 → LED1 = pin116
  - C1 → LED2 = pin117
  - C25 → LED3 = pin118
  - C26 → LED4 = pin119
  - (pin120, p126, p127, p128, p129, p131, p132, p133
  - p134, p136, p137)
  - C38 → LED16 = pin138
- Emulation test
  - If KEYPAD(01)= 11 → KEYPAD(D) 1 strike = CLK
  - LED12345..16 = 00000....1
  - If KEYPAD(01)= 11 → KEYPAD(D) 38 strikes = CLK
  - LED12345..16 = 11111....1
  - if KEYPAD(01)= 00 → KEYPAD(D) 1 strike = CLK
  - LED12345..16 = 11111....10 (1-bit left shifted)
  - if KEYPAD(01)= 00 → KEYPAD(D) 31 strikes = CLK

Ex6) 89-bit LFSR89 : design

- LFSR5 is p(x) = x^89 + x^88 + x^50 + x^47 + x^36
- + x^34 + x^9 + x^6 + 1



Ex6)89 -bit LFSR89 :LFSR89.VHD(1)

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;

entity LFSR89 is
port (
    CLK      : in std_logic;
    KEY_DATA  : in std_logic;
    KEY_LDEN  : in std_logic;
    C0       : out std_logic;
    C1       : out std_logic;
    C75      : out std_logic;
    C76      : out std_logic;
    C77      : out std_logic;
    C78      : out std_logic;
    C79      : out std_logic;
    C80      : out std_logic;
    C81      : out std_logic;
    C82      : out std_logic;
    C83      : out std_logic;
    C84      : out std_logic;
    C85      : out std_logic;
    C86      : out std_logic;
    C87      : out std_logic;
    C88      : out std_logic);
end LFSR89 ;
```

```
architecture ARCH_LFSR89  of LFSR89  is
  signal C        : unsigned(88 downto 0);
  signal TMP_FEED : std_logic := '0';
  signal FEED_IN  : std_logic := '0';
  begin
    C0  <= C(0);
    C1  <= C(1);
    C75 <= C(75);
    C76 <= C(76);
    C77 <= C(77);
    C78 <= C(78);
    C79 <= C(79);
    C80 <= C(80);
    C81 <= C(81);
    C82 <= C(82);
    C83 <= C(83);
    C84 <= C(84);
    C85 <= C(85);
    C86 <= C(86);
    C87 <= C(87);
    C88 <= C(88);
```

Ex6) 89 -bit LFSR89 :LFSR89.VHD(2)

```
FEED_IN <= C(0) xor C(6) xor C(9) xor
C(34) xor C(36) xor C(47) xor C(50) xor
C(88);
    TMP_FEED <= (KEY_LDEN and
KEY_DATA) or (not(KEY_LDEN) and
FEED_IN);

    process(CLK)
    begin
      if CLK'event and CLK = '1' then
        C(87 downto 0) <= C(88 downto 1);
        C(88) <= TMP_FEED;
      end if;
    end process;

end ARCH_LFSR89;
```

Ex6) LFSR89: HDB-DTK-240 KIT

- Input pin
  - KEY_LDEN → KEYPAD(0)=pin98
  - KEY_DATA → KEYPAD(1)=pin79
  - CLK → KEYPAD(D)=pin101
- Output pin
  - C0 → LED1 = pin116
  - C1 → LED2 = pin117
  - C75 → LED3 = pin118
  - C76 → LED4 = pin119
  - (pin120, p126, p127, p128, p129, p131, p132, p133
  - p134, p136, p137)
  - C88 → LED16 = pin138
- Emulation test
  - If KEYPAD(01)= 11 → KEYPAD(D) 1 strike = CLK
  - LED12345..16 = 00000....1
  - If KEYPAD(01)= 11 → KEYPAD(D) 38 strikes = CLK
  - LED12345..16 = 11111....1
  - if KEYPAD(01)= 00 → KEYPAD(D) 1 strike = CLK
  - LED12345..16 = 11111....10 (1-bit left shifted)
  - if KEYPAD(01)= 00 → KEYPAD(D) 31 strikes = CLK

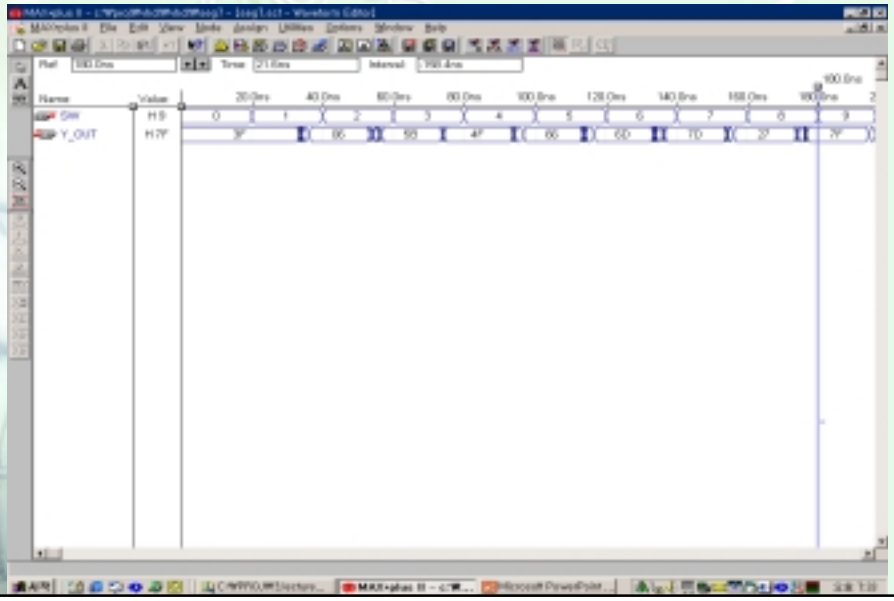
Ex7) SW to 7-segment Display (1)

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;

entity SEG7 is
port ( SW      : in std_logic_vector(3 downto 0);
       Y_OUT   : out std_logic_vector(6 downto 0) );
end SEG7;
```

Ex7) SW to 7-segment Display (2)

```
architecture ARCH_SEG7 of SEG7 is
function dis_seg(x : std_logic_vector(3 downto 0)) return std_logic_vector is
        variable seg_decode : std_logic_vector(6 downto 0);
        begin
            case x is
                    when "0000" => seg_decode := "0111111";
                    when "0001" => seg_decode := "0000110";
                    when "0010" => seg_decode := "1011011";
                    when "0011" => seg_decode := "1001111";
                    when "0100" => seg_decode := "1100110";
                    when "0101" => seg_decode := "1101101";
                    when "0110" => seg_decode := "1111101";
                    when "0111" => seg_decode := "0100111";
                    when "1000" => seg_decode := "1111111";
                    when "1001" => seg_decode := "1100111";
                    when others => seg_decode := "0000000";
            end case;
            return (seg_decode);
        end dis_seg;

begin
        Y_OUT <= dis_seg(SW);
end ARCH_SEG7;
```

Ex7)SEG7 : Simulation Waveform



Ex7)SEG7 : HDB-DTK-240 KIT

Emulation test

- If KEYPAD(123A)=0000 → 7-SEG="00000000"
- If KEYPAD(123A)=0001 → 7-SEG="11111111"
- If KEYPAD(123A)=0010 → 7-SEG="22222222"
- If KEYPAD(123A)=0011 → 7-SEG="33333333"
- If KEYPAD(123A)=0100 → 7-SEG="44444444"
- If KEYPAD(123A)=0101 → 7-SEG="55555555"
- If KEYPAD(123A)=0110 → 7-SEG="66666666"
- If KEYPAD(123A)=0111 → 7-SEG="77777777"
- If KEYPAD(123A)=1000 → 7-SEG="88888888"
- If KEYPAD(123A)=1001 → 7-SEG="99999999"

Ex8) KEYPAD to 7-SEG : SW설정

<SW 설정>

- SW(15 14 13 ... 0) = 10000000000...000
- SW(15 14 13 ... 0) = 01000000000...000
- SW(15 14 13 ... 0) = 00100000000...000
- .....
- SW(15 14 13 ... 0) = 00000000000...001
- SW(15) = pin98 (KEYPAD=0)
- SW(14) = pin79 (KEYPAD=1)
- SW(13) = pin80 (KEYPAD=2)
- SW(12) = pin81 (KEYPAD=3)
- SW(11) = pin83 (KEYPAD=4)
- SW(10..0) = pin84/85/86/88/94/95/82/87/97
- /101/100/99(KEYPAD=56...EF)

Ex8) KEYPAD to 7 -seg Display (1)

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;

entity SEG_0F is
      port ( sw : in std_logic_vector(15 downto 0);
                  y_out : out std_logic_vector(6 downto 0));
end SEG_0F;
architecture ARCH_SEG of SEG_0F is
function dis_seg(x : std_logic_vector(15 downto 0)) return std_logic_vector is
    variable seg_decode : std_logic_vector(6 downto 0);
    begin
            case x is
                    when "1000000000000000" => seg_decode := "0111111"; --0
                    when "0100000000000000" => seg_decode := "0000110"; --1
                    when "0010000000000000" => seg_decode := "1011011"; --2
                    when "0001000000000000" => seg_decode := "1001111"; --3
                    when "0000100000000000" => seg_decode := "1100110"; --4
                    when "0000010000000000" => seg_decode := "1101101"; --5
```

Ex8) KEYPAD to 7 -seg Display (2)

```
                    when "0000001000000000" => seg_decode := "1111101"; --6
                    when "0000000100000000" => seg_decode := "0100111"; --7
                    when "0000000010000000" => seg_decode := "1111111"; --8
                    when "0000000001000000" => seg_decode := "1100111"; --9
                    when "0000000000100000" => seg_decode := "1110111"; --A
                    when "0000000000010000" => seg_decode := "1111100"; --b
                    when "0000000000001000" => seg_decode := "0111001"; --C
                    when "0000000000000100" => seg_decode := "1011110"; --d
                    when "0000000000000010" => seg_decode := "1111001"; --E
                    when "0000000000000001" => seg_decode := "1110001"; --F
                    when others => seg_decode := "0000000"; --0
            end case;
            return (seg_decode);
    end dis_seg;

    begin
            y_out <= dis_seg(sw);
end ARCH_SEG;
```

Ex8)SEG7 : HDB-DTK-240 KIT

Emulation test

- If KEYPAD(0) → 7-SEG="00000000"
- If KEYPAD(1) → 7-SEG="11111111"
- If KEYPAD(2) → 7-SEG="22222222"
- If KEYPAD(3) → 7-SEG="33333333"
- If KEYPAD(4) → 7-SEG="44444444"
- If KEYPAD(5) → 7-SEG="55555555"
- If KEYPAD(6) → 7-SEG="66666666"
- If KEYPAD(7) → 7-SEG="77777777"
- If KEYPAD(8) → 7-SEG="88888888"
- If KEYPAD(9) → 7-SEG="99999999"
- If KEYPAD(A) → 7-SEG="AAAAAAAA"
- If KEYPAD(B) → 7-SEG="bbbbbbbb"
- If KEYPAD(C) → 7-SEG="CCCCCCCC"
- If KEYPAD(D) → 7-SEG="dddddddd"
- If KEYPAD(E) → 7-SEG="EEEEEEEE"

Ex9) KEY0 – Z to 7-SEG: SW설정

- SW(15 14 13 ... 0) = 10000000000...000 → 0
- SW(15 14 13 ... 0) = 01000000000...000 → 1
- SW(15 14 13 ... 0) = 00100000000...000 → 2
- .....
- SW(15 14 13 ... 0) = 00000000000...001 → F
- SW(15 14 13 ... 0) = 11000000000...000 → g
- SW(15 14 13 ... 0) = 01100000000...000 → H
- SW(15 14 13 ... 0) = 00110000000...000 → i
- .....
- SW(15 14 13 ... 0) = 00000000000...011 → y

- SW(15) = pin98 (KEYPAD=0)
- SW(14) = pin79 (KEYPAD=1)
- SW(13) = pin80 (KEYPAD=2)
- SW(12) = pin81 (KEYPAD=3)
- SW(11) = pin83 (KEYPAD=4)
- SW(10..0) = pin84/85/86/88/94/95/82/87/97
  /101/100/99(KEYPAD=56...EF)

Ex9) KEY0 – Z to 7 -SEG Display (1)

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;

entity SEG_0Z is
      port ( sw : in std_logic_vector(15 downto 0);
                  y_out : out std_logic_vector(6 downto 0));
end SEG_0Z;
architecture ARCH_SEG of SEG_0Z is
function dis_seg(x : std_logic_vector(15 downto 0)) return std_logic_vector is
    variable seg_decode : std_logic_vector(6 downto 0);
    begin
            case x is
                  when "1000000000000000" => seg_decode := "0111111"; --0
                  when "0100000000000000" => seg_decode := "0000110"; --1
                  when "0010000000000000" => seg_decode := "1011011"; --2
                  when "0001000000000000" => seg_decode := "1001111"; --3
                  when "0000100000000000" => seg_decode := "1100110"; --4
                  when "0000010000000000" => seg_decode := "1101101"; --5
```

Ex9) KEY0 – Z to 7 -SEG Display (2)

```
                  when "0000001000000000" => seg_decode := "1111101"; --6
                  when "0000000100000000" => seg_decode := "0100111"; --7
                  when "0000000010000000" => seg_decode := "1111111"; --8
                  when "0000000001000000" => seg_decode := "1100111"; --9
                  when "0000000000100000" => seg_decode := "1110111"; --A
                  when "0000000000010000" => seg_decode := "1111100"; --b
                  when "0000000000001000" => seg_decode := "0111001"; --C
                  when "0000000000000100" => seg_decode := "1011110"; --d
                  when "0000000000000010" => seg_decode := "1111001"; --E
                  when "0000000000000001" => seg_decode := "1110000"; --F
                  when "0110000000000000"=> seg_decode := "1101111";  --g
                  when "0011000000000000"=> seg_decode := "1110110";  --H
                  when "0001100000000000"=> seg_decode := "0000110";  --i
                  when "0000110000000000"=> seg_decode := "0001110";  --j
                  when "0000011000000000"=> seg_decode := "1111010";  --k
                  when "0000001100000000"=> seg_decode := "0111000";  --L
                  when "0000000110000000"=> seg_decode := "1010100";  --n
```

Ex9) KEY0 – Z to 7 -SEG Display (3)

```
                    when "0000000011000000"=> seg_decode := "1011100";  --o
                    when "0000000001100000"=> seg_decode := "1110011";  --P
                    when "0000000000110000"=> seg_decode := "1010000";  --r
                    when "0000000000011000"=> seg_decode := "1101101";  --S
                    when "0000000000001100"=> seg_decode := "1111000";  --t
                    when "0000000000000110"=> seg_decode := "0011100";  --u
                    when "0000000000000011"=> seg_decode := "1101110";  --y
                    when others => seg_decode := "0000000";  --0
            end case;
            return (seg_decode);
    end dis_seg;

    begin
            y_out <= dis_seg(sw);
end ARCH_SEG;
```

Ex9)SEG7 : HDB -DTK -240 KIT(1)

Emulation test

- If KEYPAD(0) → 7 -SEG=“00000000”
- If KEYPAD(1) → 7 -SEG=“11111111”
- If KEYPAD(2) → 7 -SEG=“22222222”
- If KEYPAD(3) → 7 -SEG=“33333333”
- If KEYPAD(4) → 7 -SEG=“44444444”
- If KEYPAD(5) → 7 -SEG=“55555555”
- If KEYPAD(6) → 7 -SEG=“66666666”
- If KEYPAD(7) → 7 -SEG=“77777777”
- If KEYPAD(8) → 7 -SEG=“88888888”
- If KEYPAD(9) → 7 -SEG=“99999999”
- If KEYPAD(A) → 7 -SEG=“AAAAAAAA”
- If KEYPAD(B) → 7 -SEG=“bbbbbbbb”
- If KEYPAD(C) → 7 -SEG=“CCCCCCCC”
- If KEYPAD(D) → 7 -SEG=“dddddddd”
- If KEYPAD(E) → 7 -SEG=“EEEEEEEE”

Ex9)SEG7 : HDB-DTK-240 KIT(2)

Emulation test

- If KEYPAD(12) → 7-SEG=“gggggggg”
- If KEYPAD(23) → 7-SEG=“HHHHHHHH”
- If KEYPAD(34) → 7-SEG=“iiiiiiii”
- If KEYPAD(45) → 7-SEG=“jjjjjjjj”
- If KEYPAD(56) → 7-SEG=“kkkkkkkk”
- If KEYPAD(67) → 7-SEG=“LLLLLLLL”
- If KEYPAD(78) → 7-SEG=“nnnnnnnn”
- If KEYPAD(89) → 7-SEG=“oooooooo”
- If KEYPAD(9A) → 7-SEG=“PPPPPPPP”
- If KEYPAD(AB) → 7-SEG=“rrrrrrrr”
- If KEYPAD(BC) → 7-SEG=“SSSSSSSS”
- If KEYPAD(CD) → 7-SEG=“tttttttt”
- If KEYPAD(DE) → 7-SEG=“uuuuuuuu”