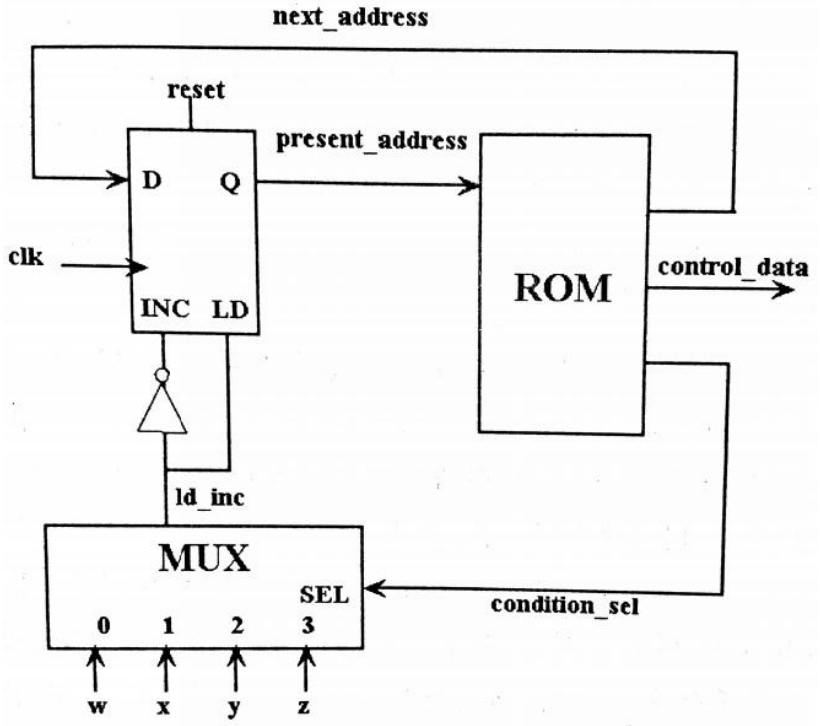
Lab 05.

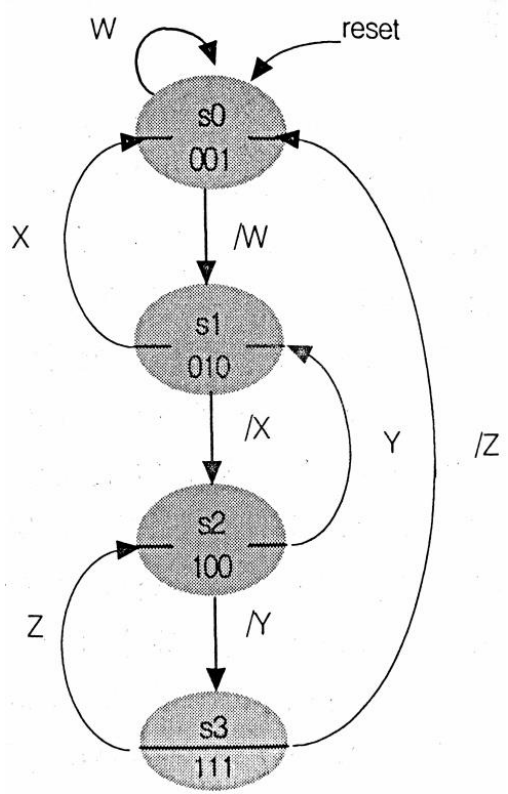
Microprogramming Logic Design

201810800 이혜인

▶오른쪽 아래의 Moore machine은 적당한 조건에서 적절한 control data를 내보내기 위한 state machine이다. 이것과 동일한 동작을 수행하는 Microprogramming logic을 설계하라.

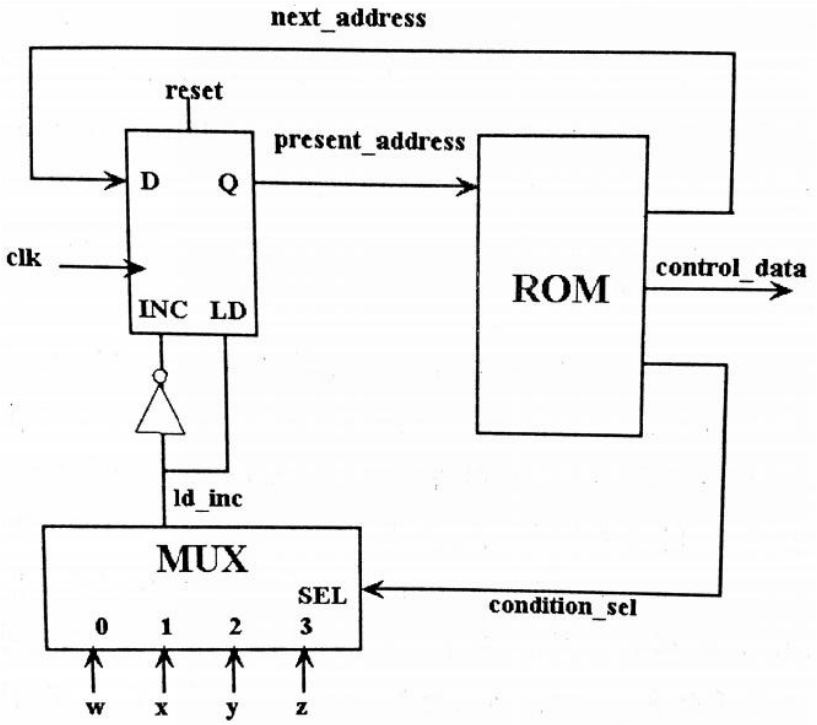
▶아래의 기본 회로를 이용하라.

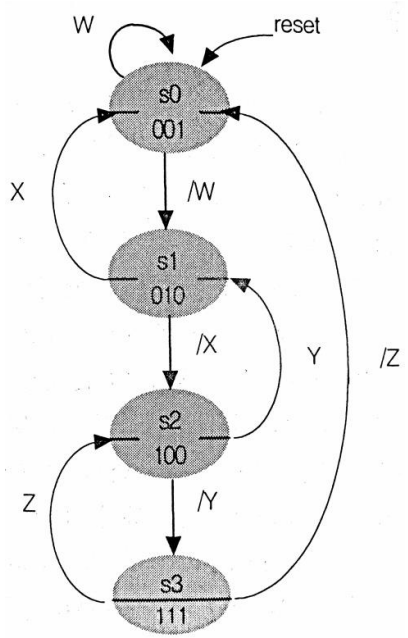




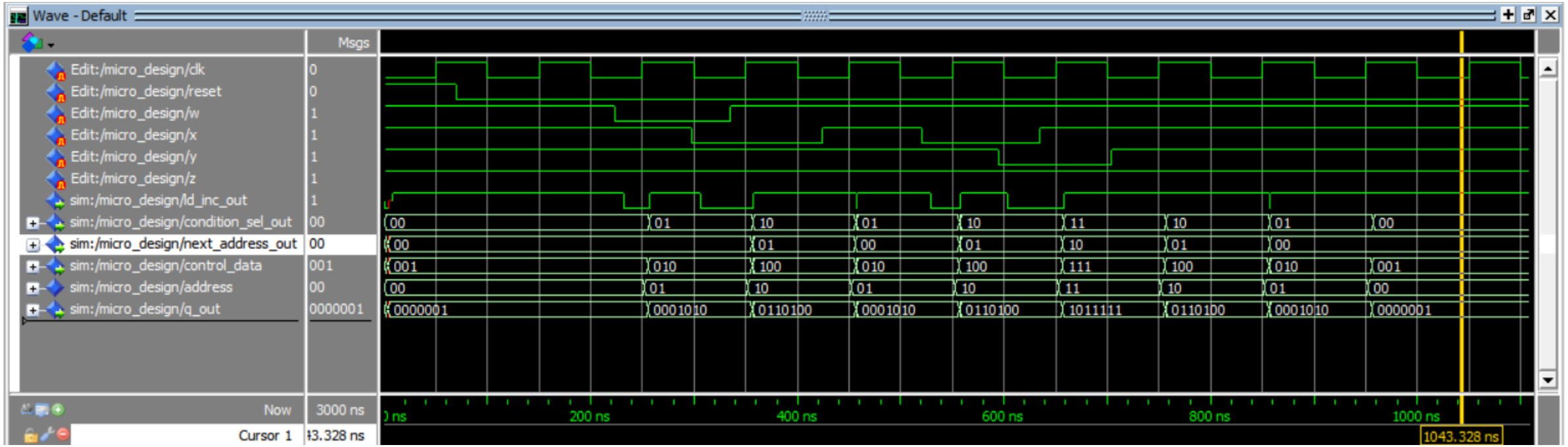
▶ROM content

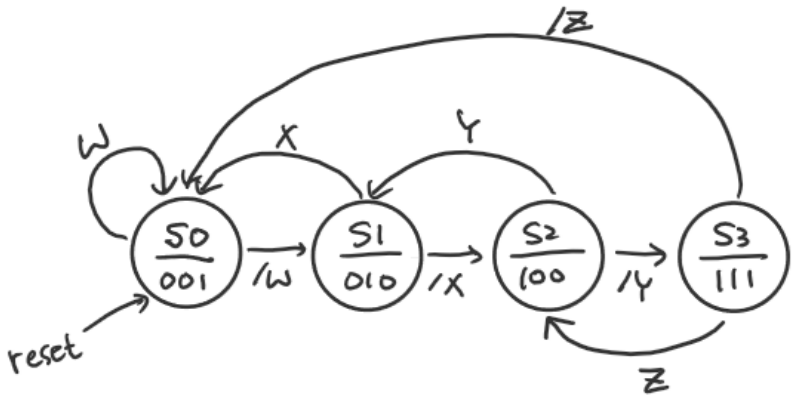
ROM의 주소	ROM의 데이터		
present_address	next_address	condition_sel	control_data
00	00	00	001
01	00	01	010
10	01	10	100
11	10	11	111





Simulation example





Index

1. VHDL Code

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;

entity keyl is
  port( clk, reset            : in std_logic;
        w, x, y, z            : in std_logic;
        ld_inc_out            : out std_logic;
        condition_sel_out     : out std_logic_vector(1 downto 0);
        next_address_out      : out std_logic_vector(1 downto 0);
        q_out                 : out std_logic_vector(6 downto 0);
        control_data          : out std_logic_vector(2 downto 0));
end keyl;

architecture rtl of keyl is
  signal address        : std_logic_vector(1 downto 0);
  signal next_address   : std_logic_vector(1 downto 0);
  signal condition_sel  : std_logic_vector(1 downto 0);

  signal ld_inc   : std_logic;
  signal q        : std_logic_vector(6 downto 0);

  type ROM is array(0 to 3) of std_logic_vector(6 downto 0);
  constant rom_7x4: ROM := (
  "0000001",
  "0001010",
  "0110100",
  "1011111"
  );
begin
  reg : process (clk, reset)
  begin
    if reset = '1' then
      address <= "00";
    elsif clk'event and clk='1' then
      if ld_inc ='1' then
        address <= next_address;
      else
        address <= address + 1;
      end if;
    end if;
  end process;

  con_mux: process(condition_sel, w, x, y, z)
   begin
     case condition_sel is
       when "00" => ld_inc <= w;
       when "01" => ld_inc <= x;
       when "10" => ld_inc <= y;
       when "11" => ld_inc <= z;
       when others => ld_inc <= '1';
     end case;
  end process;

  q <= rom_7x4(to_integer(unsigned(address)));

  ld_inc_out <= ld_inc;
  condition_sel_out <= condition_sel;
  next_address_out <= next_address;
  next_address <= q(6 downto 5);
  condition_sel <= q(4 downto 3);
  control_data<=q(2 downto 0);
  q_out <= q;
end rtl;
```

→ 주어진 코드를 이용하여 앞에 명시된 Rom Content에 맞게 작성한 VHDL Code이다.

1. VHDL Code

```
1   library ieee;
2   use ieee.std_logic_1164.all;
3   use ieee.std_logic_unsigned.all;
4   use ieee.numeric_std.all;
5
6   entity keyl is
7     port( clk, reset            : in std_logic;
8           w, x, y, z            : in std_logic;
9           ld_inc_out            : out std_logic;
10          condition_sel_out     : out std_logic_vector(1 downto 0);
11          next_address_out      : out std_logic_vector(1 downto 0);
12          q_out                 : out std_logic_vector(6 downto 0);
13          control_data          : out std_logic_vector(2 downto 0));
14  end keyl;
15
16  architecture rtl of keyl is
17    signal address          : std_logic_vector(1 downto 0);
18    signal next_address     : std_logic_vector(1 downto 0);
19    signal condition_sel    : std_logic_vector(1 downto 0);
20
21    signal ld_inc    : std_logic;
22    signal q         : std_logic_vector(6 downto 0);
```

→ 다음 VHDL은 clk(Clock)와 reset을 input으로 정의하였고, address를 바꿔 줄 input인 w, x, y, z를 정의했다.

→ 또한, ROM주소와 Data를 내부 Signal로 각각 address, next_address, condition_sel를 정의하였고, address를 1씩 up-count시킬 id_inc와 ROM Data를 한번에 나타낸 q를 정의하였다.

1. VHDL Code

```
 6  entity key1 is
 7    port( clk, reset            : in std_logic;
 8          w, x, y, z            : in std_logic;
 9          ld_inc_out            : out std_logic;
10          condition_sel_out     : out std_logic_vector(1 downto 0);
11          next_address_out      : out std_logic_vector(1 downto 0);
12          q_out                 : out std_logic_vector(6 downto 0);
13          control_data          : out std_logic_vector(2 downto 0));
14  end key1;
15
16  architecture rtl of key1 is
17    signal address         : std_logic_vector(1 downto 0);
18    signal next_address    : std_logic_vector(1 downto 0);
19    signal condition_sel   : std_logic_vector(1 downto 0);
20
21    signal ld_inc    : std_logic;
22    signal q         : std_logic_vector(6 downto 0);
23
24    type ROM is array(0 to 3) of std_logic_vector(6 downto 0);
25    constant rom_7x4: ROM := (
26    "0000001",
27    "0001010",
28    "0110100",
29    "1011111"
30    );
```

→ 이 모든 것을 외부로 출력해줄 out std_logic_vector들을 PORT에 정의하였다.

→ 그리고 7*4 ROM(7 bits가 4개)을 constant를 통해 다음과 같이 정의하였다. 이를 정의할 때, ROM Data를 이용하였다.

ROM의 주소	ROM의 데이터		
present_address	next_address	condition_sel	control_data
00	00	00	001
01	00	01	010
10	01	10	100
11	10	11	111

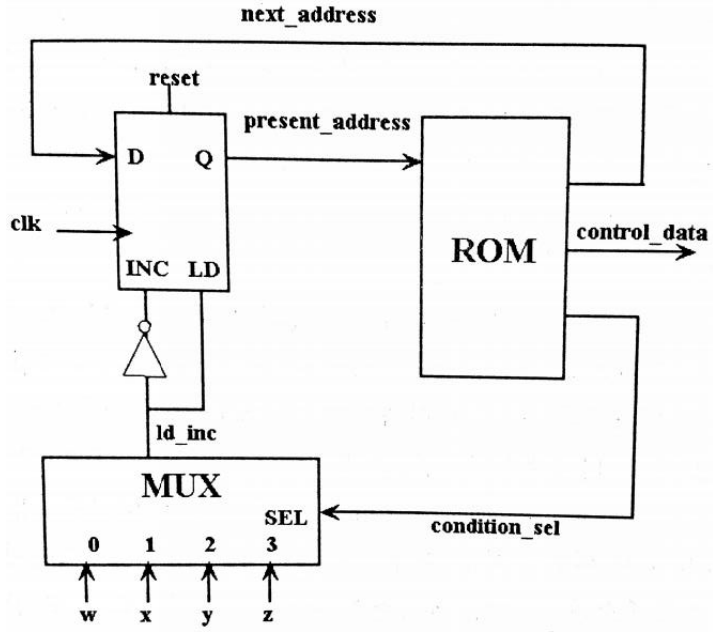
1. VHDL Code

→ Reset이 1이면 address를 "00"으로 초기화한다.

→ 그리고 reset이 1이 아니고 Clock의 rising edge인데 id_inc가 1인 경우(회로도를 참고해보면 이는 Load를 시키는 경우이다.) next_address를 현재 address에 Load한다.

→ id_inc가 1이 아닌 경우(회로도를 참고하면 이는 INC+1을 시킨다.) 현재 address에 +1을 시켜 다음 address로 넘어가게 한다.

```
32  begin
33    reg : process (clk, reset)
34    begin
35      if reset = '1' then
36        address <= "00";
37      elsif clk'event and clk='1' then
38        if ld_inc ='1' then
39          address <= next_address;
40        else
41          address <= address + 1;
42        end if;
43      end if;
44    end process;
```

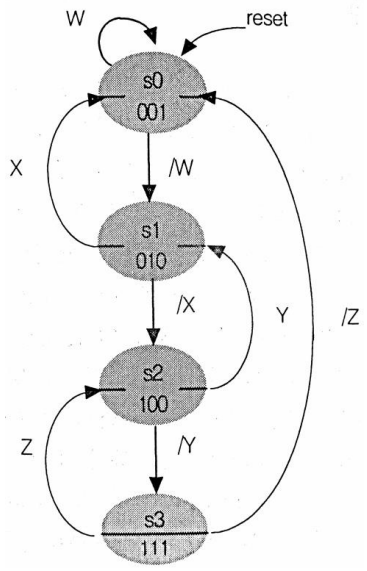


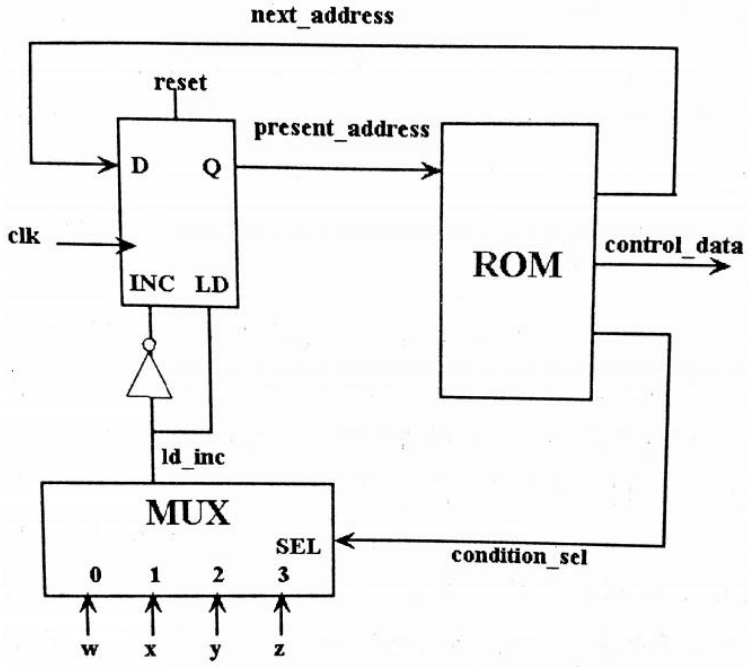
1. VHDL Code

→ Condition_sel이 00일 때, id_inc는 w가 들어오고, 01일 때는 x, 10일 때는 y, 11일 때는 z가 들어오게 된다. 이 때, id_inc는 0이 되어 각각 현재 address에 +1을 시켜 다음 address로 넘어가게 한다.

→ 나머지 경우에는 id_inc가 1이므로 Load가 1이 되어 next_address를 현재 address에 Load한다.

```
46  con_mux: process(condition_sel, w, x, y, z)
47   begin
48     case condition_sel is
49       when "00" => ld_inc <= w;
50       when "01" => ld_inc <= x;
51       when "10" => ld_inc <= y;
52       when "11" => ld_inc <= z;
53       when others => ld_inc <= '1';
54     end case;
55  end process;
```





1. VHDL Code

→ Q는 rom_7*4를 integer 형태로 받는다.

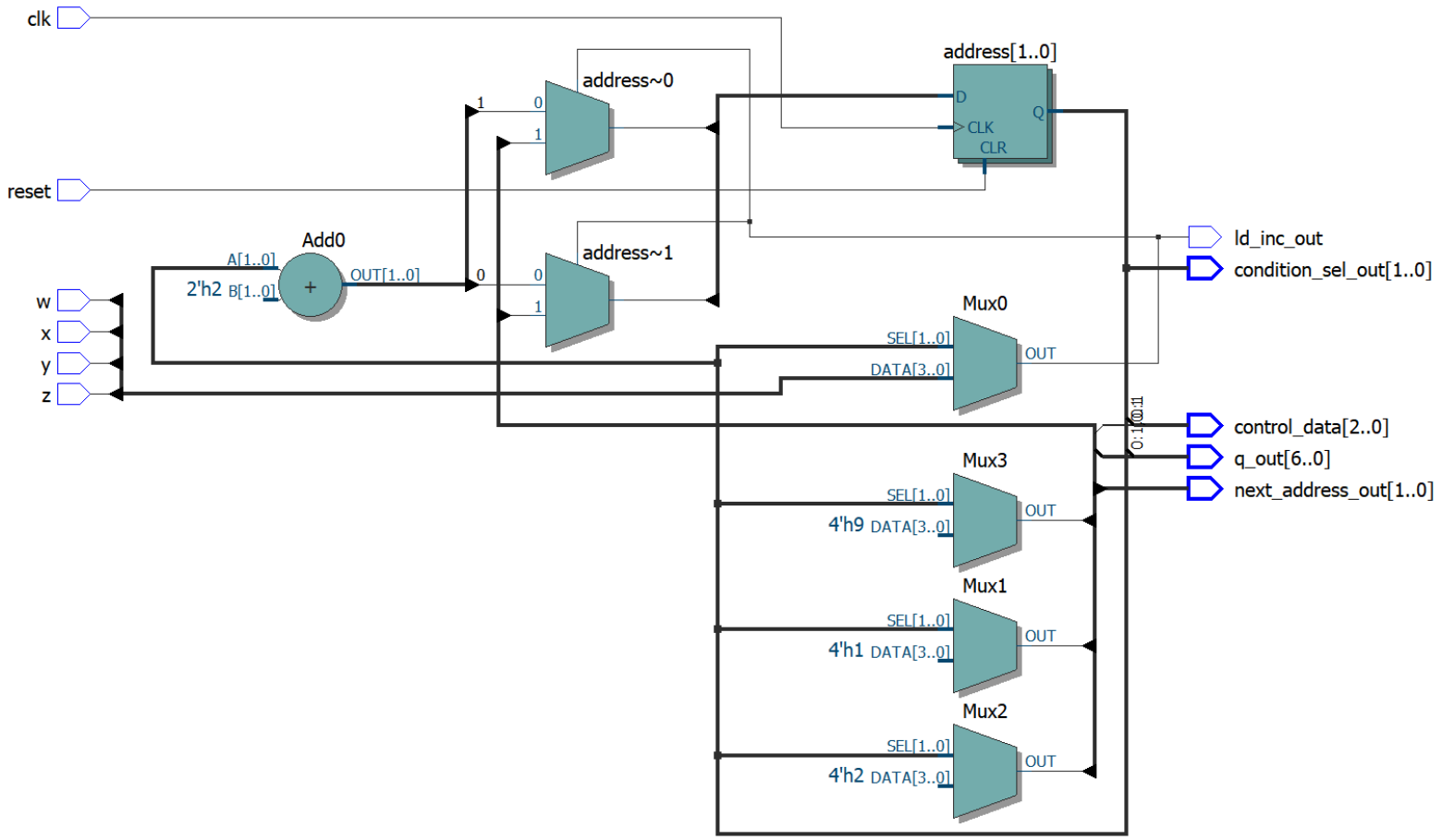
→ 각각 내부 output을 외부로 출력할 수 있

도록 외부 Output과 연결한다.

```
57      q <= rom_7x4(to_integer(unsigned(address)));
58
59      ld_inc_out <= ld_inc;
60      condition_sel_out <= condition_sel;
61      next_address_out <= next_address;
62      next_address <= q(6 downto 5);
63      condition_sel <= q(4 downto 3);
64      control_data<=q(2 downto 0);
65      q_out <= q;
66  end rtl;
```

ROM의 주소	ROM의 데이터		
present_address	next_address	condition_sel	control_data
00	00	00	001
01	00	01	010
10	01	10	100
11	10	11	111

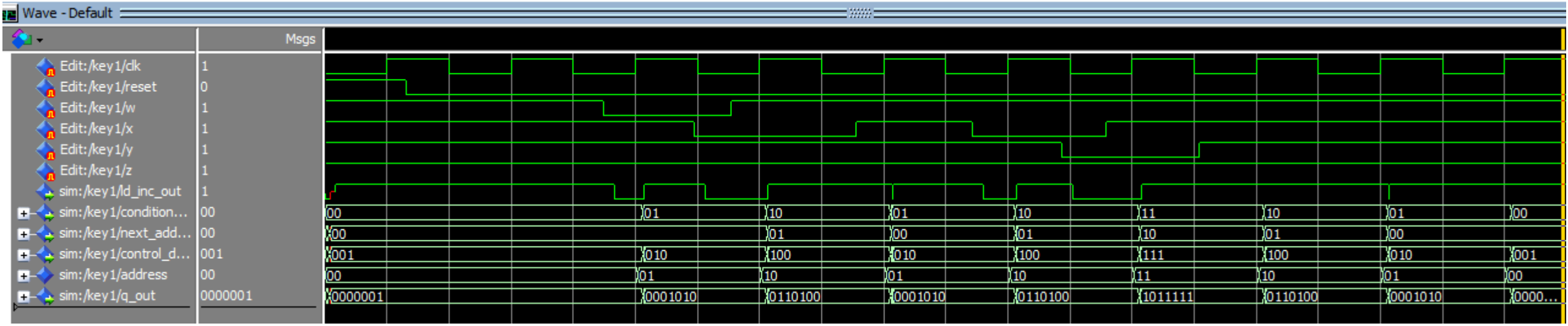
1. VHDL Code - RLT Viewer



→ 앞선 VHDL Code를 RTL Viewer를 통해 나타내면 다음과 같다.

→ w, x, y, z를 각각의 input으로 받고, control_data, condition_sel, q, next_address를 각각 외부로 output을 만들어서 이를 출력한다.

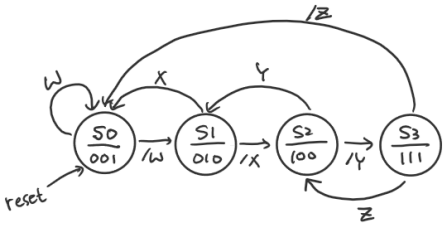
3. Simulation capture



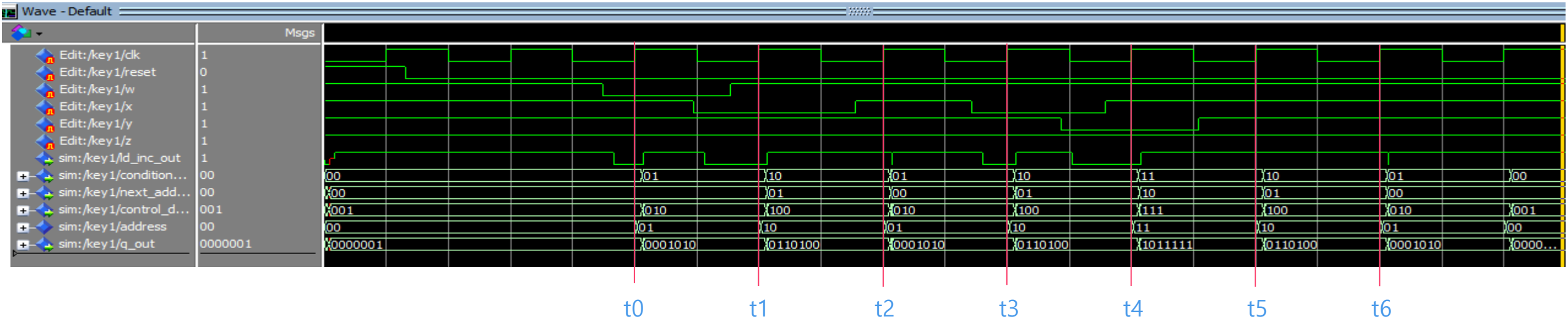
→ 해당 Simulation은 앞에 주어진 자료인 Simulation Example과 동일하게 진행한 것으로 결과가 동일하게 나왔음을 확인할 수 있다.

→ **w가 0이 되면 address가 01이 되고, 01일 때, x가 0이 되면 address가 10이 된다. 10일 때 y가 1이면 다시 01이 되고, y가 0이면 address가 11이 되는데, 모두 올바르게 변하는 것을 확인할 수 있다.**

3. Simulation capture



ROM의 주소	ROM의 데이터		
present_address	next_address	condition_sel	control_data
00	00	00	001
01	00	01	010
10	01	10	100
11	10	11	111



→ t0 : address가 00일 때, w가 0이므로 다음 address는 01이 된다. 이 때, control_data는 010이고, conditional_sel은 01, 이 모두를 합친 q는 0001010이다.

→ t1 : address가 01일 때, x가 0이므로 다음 address는 10이 된다. 이 때, control_data는 100이고, conditional_sel은 10, 이 모두를 합친 q는 0110100이다.

→ t2 : address가 10일 때, y가 1이므로 다음 address는 01이 된다. 이 때, control_data는 010이고, conditional_sel은 01, 이 모두를 합친 q는 0001010이다.

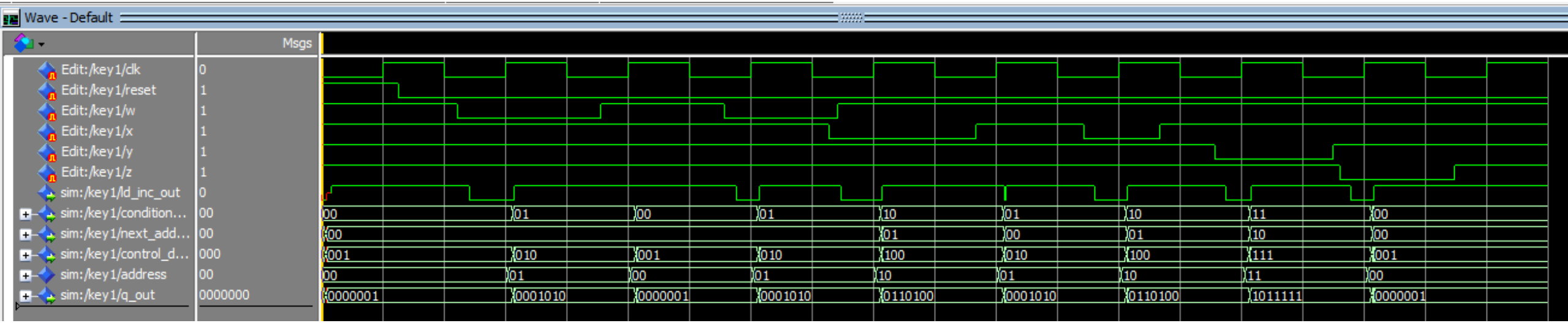
→ t3 : address가 01일 때, x가 0이므로 다음 address는 10이 된다. 이 때, control_data는 100이고, conditional_sel은 10, 이 모두를 합친 q는 0110100이다.

→ t4 : address가 10일 때, y가 0이므로 다음 address는 11이 된다. 이 때, control_data는 111이고, conditional_sel은 11, 이 모두를 합친 q는 1011111이다.

→ t5 : address가 11일 때, z가 1이므로 다음 address는 10이 된다. 이 때, control_data는 100이고, conditional_sel은 10, 이 모두를 합친 q는 0110100이다.

→ t6 : address가 10일 때, y가 1이므로 다음 address는 01이 된다. 이 때, control_data는 010이고, conditional_sel은 01, 이 모두를 합친 q는 0001010이다.

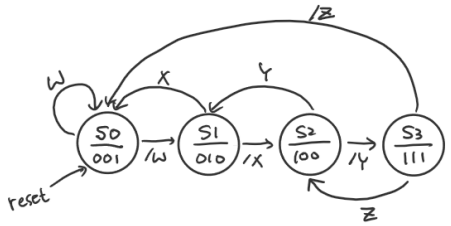
3. Simulation capture



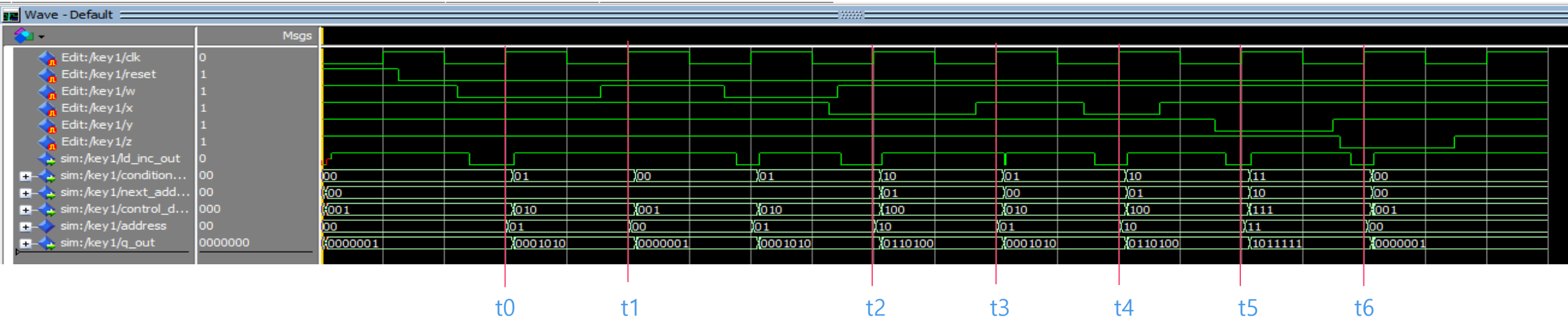
→ 해당 Simulation은 임의로 진행한 Simulation으로 결과가 올바르게 나온 것을 확인할 수 있다.

→ **W가 0이 되면 address가 01이 되고, 01일 때, X가 1이 되면 address가 00이 되고, X가 0이 되면 address가 10이 된다. 10일 때 Y가 1이면 다시 01이 되고, Y가 0이면 address가 11이 된다. 또한 address가 11일 때 z가 0이 되면 address가 00이 된다. 다음 Simulation은 모두 올바르게 변하는 것을 확인할 수 있다.**

3. Simulation capture



ROM의 주소	ROM의 데이터		
present_address	next_address	condition_sel	control_data
00	00	00	001
01	00	01	010
10	01	10	100
11	10	11	111



→ t0 : address가 00일 때, w가 0이므로 다음 address는 01이 된다. 이 때, control_data는 010이고, conditional_sel은 01, 이 모두를 합친 q는 0001010이다.

→ t1 : address가 01일 때, x가 1이므로 다음 address는 00이 된다. 이 때, control_data는 001이고, conditional_sel은 00, 이 모두를 합친 q는 0000001이다.

→ t2 : address가 01일 때, x가 0이므로 다음 address는 10이 된다. 이 때, control_data는 100이고, conditional_sel은 10, 이 모두를 합친 q는 0110100이다.

→ t3 : address가 10일 때, y가 1이므로 다음 address는 01이 된다. 이 때, control_data는 010이고, conditional_sel은 01, 이 모두를 합친 q는 0001010이다.

→ t4 : address가 01일 때, x가 0이므로 다음 address는 10이 된다. 이 때, control_data는 100이고, conditional_sel은 10, 이 모두를 합친 q는 0110100이다.

→ t5 : address가 10일 때, y가 0이므로 다음 address는 11이 된다. 이 때, control_data는 111이고, conditional_sel은 11, 이 모두를 합친 q는 1011111이다.

→ t6 : address가 11일 때, z가 0이므로 다음 address는 00이 된다. 이 때, control_data는 001이고, conditional_sel은 00, 이 모두를 합친 q는 0000001이다.

4. Discussion

→ 이번 Lab을 진행하기 위해서 10강을 이해하는 데 많은 시간이 소요되었다. 이해를 했다고 생각이 되다가, 어느 때에 INC가 1이 되고 0이 되는지, Load는 어느때에 일어나는지 등 헷갈리는 부분이 종종 있어서 시간이 오래 걸린 것 같았다. 하지만, 한 번 이해한 다음부터는 주어진 Lab에 맞게 VHDL Code를 수정하는 데는 시간이 많이 소요되지 않았다. 처음에 Simulation된 것만 먼저 보고 많은 Input과 Output의 개수에 당황하였다. 하지만 각각의 기능과 필요한 Input, Output이 무엇인지, 어느 때에 어떤 address를 가지는지 이해한 다음 차근차근 진행해보니 생각보다 Lab이 수월하게 진행되었다.

4. Discussion

→ 처음 보는 Logic Design이어서 다소 생소함을 느끼기도 했지만, 하나하나 천천히 보고, 00001다음에는 어떤 address가 나오는지 꼼꼼히 살펴보니까 생각했던 것만큼 어렵지 않았다. 또한, 이해했다고 생각하지 말고 다시 보라는 교수님의 말씀이 너무나 공감되었다. 이해됐다고 생각하고 Lab을 진행하다 보니 헷갈리는 부분이 나오게 되었고 이로 인해 다시 한번 강의자료를 복습하게 되었다.

→ 이번 실습을 통해서는 하나하나 정확하게 이해하는 것의 중요성을 배우게 된 것 같다. 또한, Microprogramming Logic이 무엇인지에 대해서도 자세하게 알 수 있던 실습이었다.