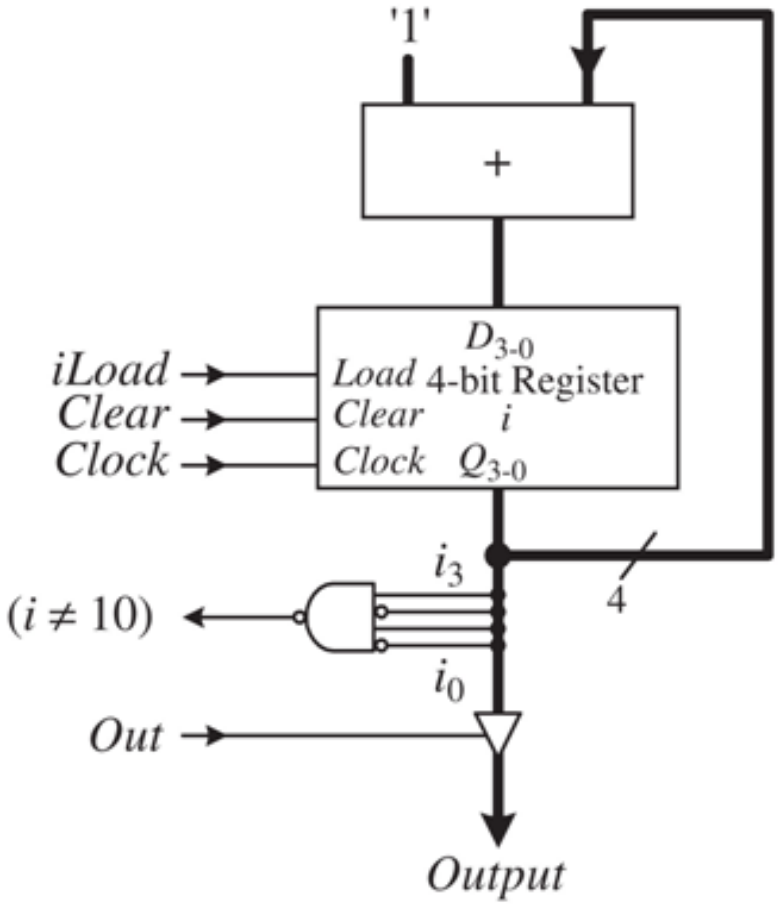
Lab 08. Elementary dedicated microprocessor

201810800 이혜인

Lab 08-1

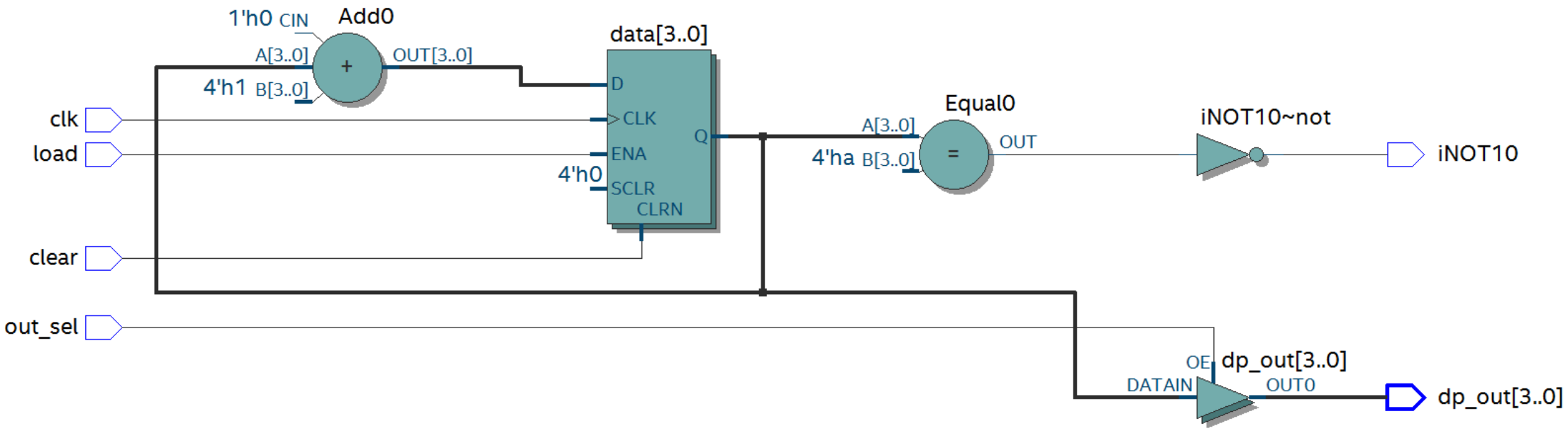
```
1  i = 0
2  WHILE (i ≠ 10) {
3      i = i + 1
4      OUTPUT i
5      }
```

- 다음 페이지에 주어진 code를 이용해서 알고리즘 실행에 필요한 Datapath를 구현하고 간단한 simulation을 통해 검증하라.
  - Variable: i (register 필요)
  - Functional Unit: adder
  - Line 1 (i=0)과 line 3 (i=i+1) 에서 i에 두 가지 입력을 필요로 함.
    - MUX를 사용할 수도 있겠지만 register를 clear (reset) 하는 것으로 충분함.
  - Status signal (i≠10) 을 control unit에 제공해야 함.



Control Word	Instruction	*iLoad*	*Clear*	*Out*
1	$i = 0$	0	1	0
2	$i = i + 1$	1	0	0
3	OUTPUT *i*	0	0	1

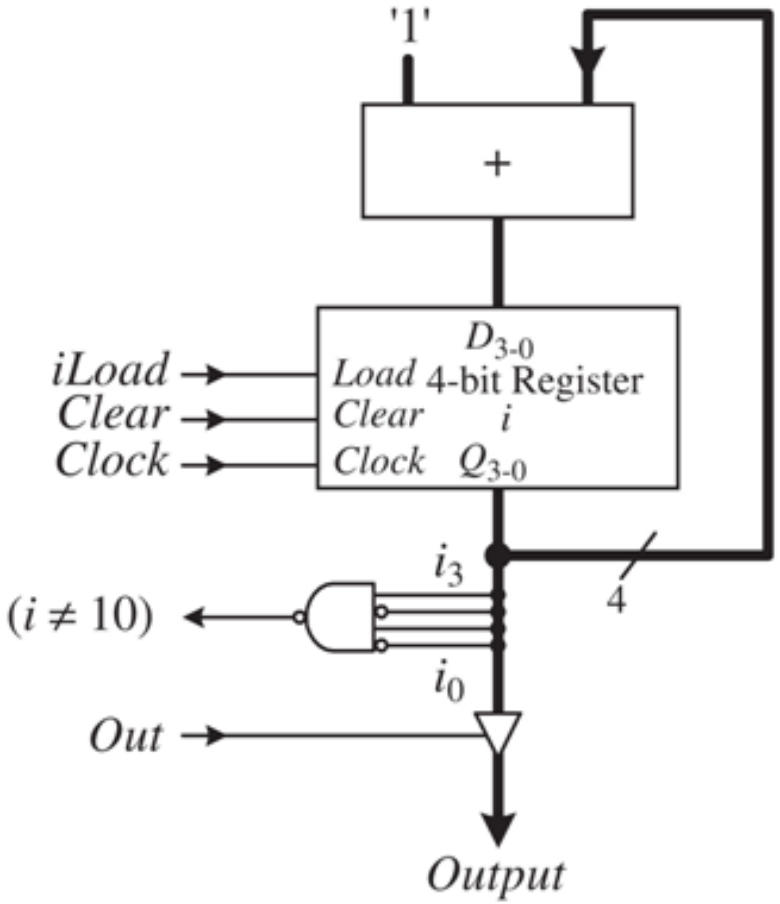
08-1. RTL view example



Lab 08-2

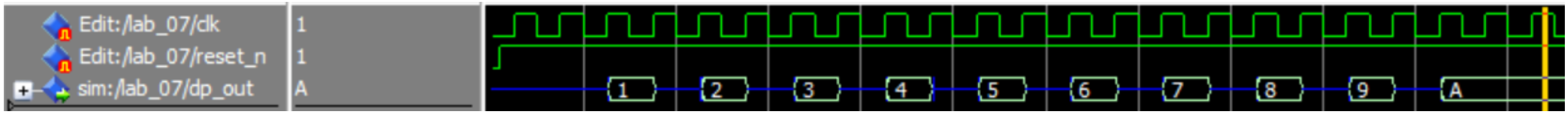
```
1  i = 0
2  WHILE (i ≠ 10) {
3      i = i + 1
4      OUTPUT i
5      }
```

- Create the control unit for the algorithm & combine it with DP.
- 전체 회로를 simulation을 통해 검증하라.



Control Word	Instruction	*iLoad*	*Clear*	*Out*
1	$i = 0$	0	1	0
2	$i = i + 1$	1	0	0
3	OUTPUT i	0	0	1

08-2. Simulation result example



Lab 08-1

1. VHDL Code - Datapath

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_ARITH.ALL;
4  use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  Entity lab_08_dp is
7
8     port(
9           clk     : IN STD_LOGIC ;
10          load    : IN STD_LOGIC ;
11          clear   : IN STD_LOGIC ;
12          out_sel : IN std_logic ;
13
14          iNOT10  : out std_logic;
15          dp_out  : out std_logic_vector(3 downto 0)
16
17          );
18
19 END lab_08_dp;
20
21 Architecture dataflow of lab_08_dp is
22
23    signal data : std_logic_vector(3 downto 0) := "0000"; --initialize
24    begin
25
26       process(clear, clk) is --register i(clocked process)
27          begin
28
29          if (clear = '1') then
30                   data <= "0000"; --i initialize
31
32          elsif (clk'EVENT AND clk = '1') then
33                      if (load = '1') then
34             data <= data + '1'; --adder
35           end if;
36          end if;
37       end process;
38
39       iNOT10 <= '0' when (data = "1010") else '1'; --combinational logic
40       dp_out <= data when (out_sel = '1') else "ZZZZ";
41
42 end dataflow;
```

→ 다음은 주어진 Datapath의 VHDL Code이다.

→ 해당 Code는 data값이 10이 아니면 1씩 increment해주는 Code이다.

→ Clk, load, Clear, out_sel을 input으로 가지고, iNOT10, dp_out을 Output으로 가진다.

Lab 08-1

1. VHDL Code - Datapath

```
21  Architecture dataflow of lab_08_dp is
22
23      signal data : std_logic_vector(3 downto 0) := "0000"; --initialize
24      begin
25
26          process(clear, clk) is --register i(clocked process)
27              begin
28
29              if (clear = '1') then
30                        data <= "0000"; --i initialize
31
32              elsif (clk'EVENT AND clk = '1') then
33                            if (load = '1') then
34                 data <= data + '1'; --adder
35                end if;
36              end if;
37          end process;
38
39          iNOT10 <= '0' when (data = "1010") else '1'; --combinational logic
40          dp_out <= data when (out_sel = '1') else "ZZZZ";
41
42  end dataflow;
```

→ 먼저 Clear가 1일 때 data를 initialize해준다.

→ Clock의 rising edge에서 load가 1일 때, data값을 1씩 increment해준다.

→ 이때, iNOT10을 통해 data가 1010 즉, 10일 때 '0'으로 만들고, 10이 아닐 때는 '1'을 유지한다.

Lab 08-1

1. VHDL Code - Datapath

```
21  Architecture dataflow of lab_08_dp is
22
23      signal data : std_logic_vector(3 downto 0) := "0000"; --initialize
24      begin
25
26          process(clear, clk) is --register i(clocked process)
27              begin
28
29              if (clear = '1') then
30                        data <= "0000"; --i initialize
31
32              elsif (clk'EVENT AND clk = '1') then
33                            if (load = '1') then
34                 data <= data + '1'; --adder
35                end if;
36              end if;
37          end process;
38
39          iNOT10 <= '0' when (data = "1010") else '1'; --combinational logic
40          dp_out <= data when (out_sel = '1') else "ZZZZ";
41
42  end dataflow;
```

→ 이를 통해 iNOT10이 '0'이면, Output을 출력하고,

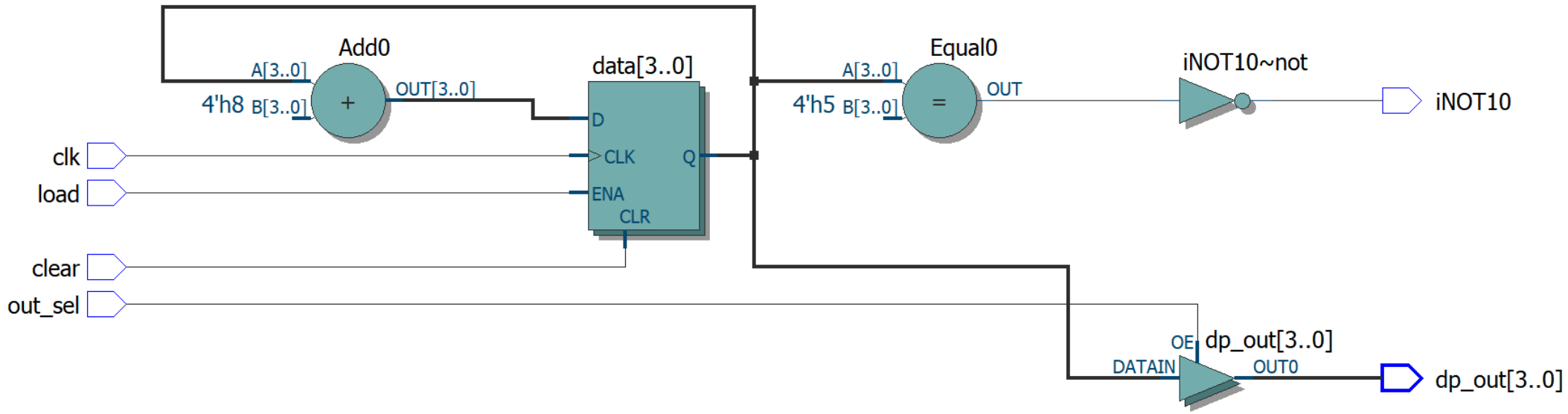
iNOT10이 '1'이면, 1을 increment한다.

→ dp_out은 out_sel이 1일 때만 data를 출력하고,

1이 아닌 경우 Z를 출력한다.

Lab 08-1

2. RTL Viewer

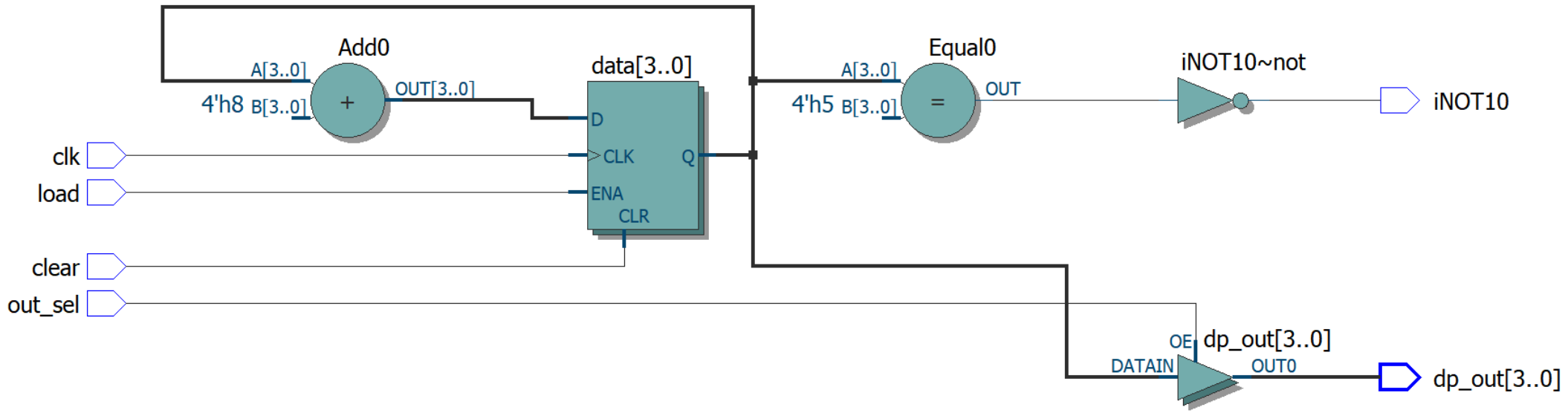


→ RTL Viewer는 다음과 같이 나온다.

→ 이는 교수님이 제시해주신 RTL View Example과 동일한 것을 알 수 있다.

Lab 08-1

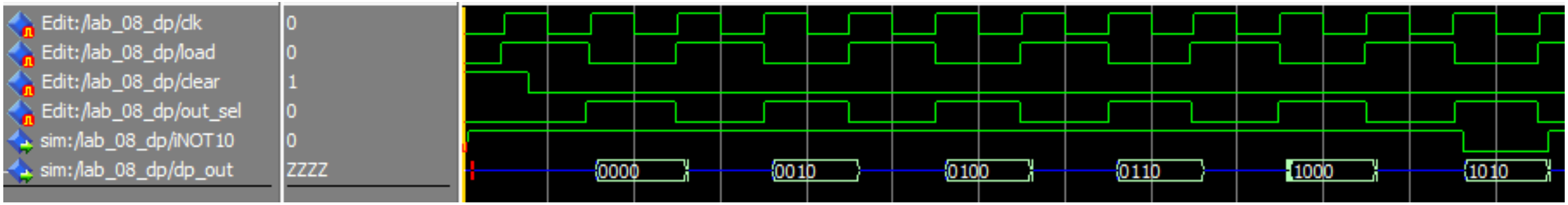
2. RTL Viewer



→ Clock에 동기화되어 ADD가 진행되고 이를 10과 비교하여 10이면 iNOT10이 0이 되어 ADD를 멈추고, 10이 아니면 data를 out하고, 다시 1씩 ADD한다.

Lab 08-1

3. Simulation



→ 해당 Simulation은 다음 Figure9.30과 동일하게 진행하였다.

→ 해당 결과를 확인해보면 1씩 increment가 진행되어 1010에 도달한 것을 볼 수 있다.

→ 즉, Figure9.30과 동일한 결과가 출력되었다.

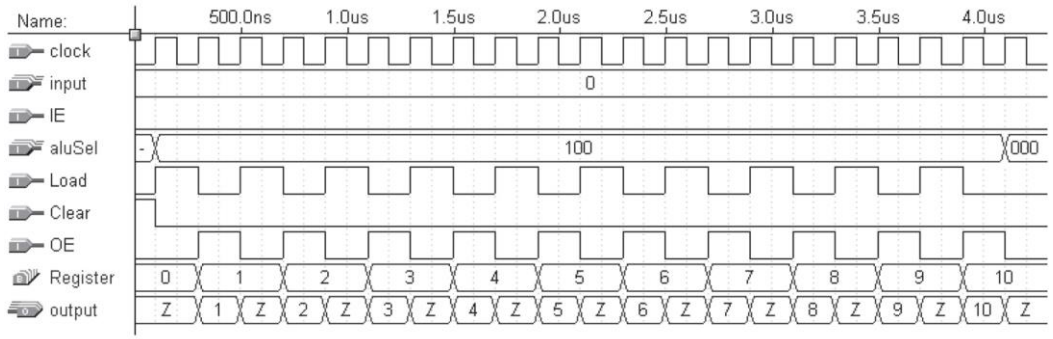


Figure 9.30 Simulation trace for Example 9.8 using the three control words shown in Figure 9.28(b).

Lab 08-1

4. VHDL Code - Datapath

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_ARITH.ALL;
4  use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  Entity lab_08_dp is
7     port(
8           clk     : IN STD_LOGIC ;
9           load    : IN STD_LOGIC ;
10          clear   : IN STD_LOGIC ;
11          out_sel : IN std_logic ;
12
13          iNOT10  : out std_logic;
14          dp_out  : out std_logic_vector(3 downto 0)
15
16          );
17 END lab_08_dp;
18
19 Architecture dataflow of lab_08_dp is
20
21    signal data : std_logic_vector(3 downto 0) := "0000"; --initialize
22    begin
23       process(clear, clk) is --register i(clocked process)
24          begin
25          if (clear = '1') then
26             data <= "0000"; --i initialize
27          elsif (clk'EVENT AND clk = '1') then
28             if (load = '1' AND data /= "1010") then
29                data <= data + '1'; --adder
30                iNOT10 <= '1';
31             else
32                iNOT10 <= '0';
33             end if;
34             IF (out_sel = '1') THEN
35                dp_out <= data; -- output
36             ELSE
37                dp_out <= "ZZZZ";
38             END IF;
39          end if;
40       end process;
41 end dataflow;
```

→ 다음은 주어진 Datapath의 VHDL Code를 아래 표와 같

이 Instruction을 바꾸어서 다시 코딩한 Code이다.

Control Word	Instruction	*IE* 6	$ALU_2ALU_1ALU_0$ 5–3	*Load* 2	*Clear* 1	*OE* 0
1	$i = 0$	×	× × ×	0	1	0
2	$i = i + 1$	0	100 (add)	1	0	0
3	$i = i + 1$ & OUTPUT i	0	100 (add)	1	0	1

Lab 08-1

4. VHDL Code - Datapath

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_ARITH.ALL;
4  use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  Entity lab_08_dp is
7     port(
8           clk     : IN STD_LOGIC ;
9           load    : IN STD_LOGIC ;
10          clear   : IN STD_LOGIC ;
11          out_sel : IN std_logic ;
12
13          iNOT10  : out std_logic;
14          dp_out  : out std_logic_vector(3 downto 0)
15
16          );
17 END lab_08_dp;
18
19 Architecture dataflow of lab_08_dp is
20
21    signal data : std_logic_vector(3 downto 0) := "0000"; --initialize
22    begin
23       process(clear, clk) is --register i(clocked process)
24          begin
25          if (clear = '1') then
26             data <= "0000"; --i initialize
27          elsif (clk'EVENT AND clk = '1') then
28             if (load = '1' AND data /= "1010") then
29                data <= data + '1'; --adder
30                iNOT10 <= '1';
31             else
32                iNOT10 <= '0';
33             end if;
34             IF (out_sel = '1') THEN
35                dp_out <= data; -- output
36             ELSE
37                dp_out <= "ZZZZ";
38             END IF;
39          end if;
40       end process;
41 end dataflow;
```

Control Word	Instruction	IE 6	$ALU_2ALU_1ALU_0$ 5–3	$Load$ 2	$Clear$ 1	OE 0
1	$i = 0$	×	× × ×	0	1	0
2	$i = i + 1$	0	100 (add)	1	0	0
3	$i = i + 1$ & OUTPUT i	0	100 (add)	1	0	1

→ Input과 Output은 기존 코드와 같다.

→ 다른 부분만 설명하자면, Clock의 rising edge에서 기존에는 load가 1일 때만 data를 1씩 increment하였는데, 해당 코드에서는 data가 10인지를 함께 고려하였다.

Lab 08-1

4. VHDL Code - Datapath

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_ARITH.ALL;
4  use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  Entity lab_08_dp is
7     port(
8           clk     : IN STD_LOGIC ;
9           load    : IN STD_LOGIC ;
10          clear   : IN STD_LOGIC ;
11          out_sel : IN std_logic ;
12
13          iNOT10  : out std_logic;
14          dp_out  : out std_logic_vector(3 downto 0)
15
16          );
17 END lab_08_dp;
18
19 Architecture dataflow of lab_08_dp is
20
21    signal data : std_logic_vector(3 downto 0) := "0000"; --initialize
22    begin
23       process(clear, clk) is --register i(clocked process)
24          begin
25          if (clear = '1') then
26             data <= "0000"; --i initialize
27          elsif (clk'EVENT AND clk = '1') then
28             if (load = '1' AND data /= "1010") then
29                data <= data + '1'; --adder
30                iNOT10 <= '1';
31             else
32                iNOT10 <= '0';
33             end if;
34             IF (out_sel = '1') THEN
35                dp_out <= data; -- output
36             ELSE
37                dp_out <= "ZZZZ";
38             END IF;
39          end if;
40       end process;
41 end dataflow;
```

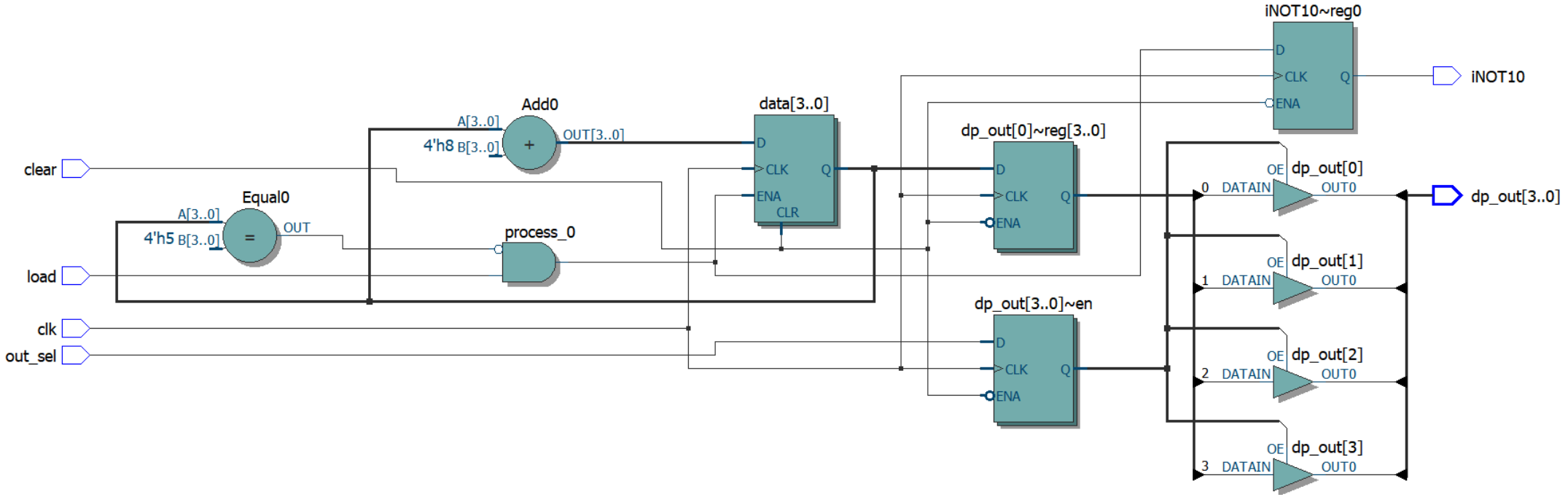
Control Word	Instruction	*IE* 6	$ALU_2ALU_1ALU_0$ 5–3	*Load* 2	*Clear* 1	*OE* 0
1	$i = 0$	×	× × ×	0	1	0
2	$i = i + 1$	0	100 (add)	1	0	0
3	$i = i + 1$ & OUTPUT i	0	100 (add)	1	0	1

→ 즉, data가 10이면 iNOT10에 1을 대입해 계속 increment를 진행하게 하고, 아니면 0을 대입해 increment를 중지시켰다.

→ 또한, Output을 이를 진행한 바로 다음에 시행하여 위에서 3번 Instruction을 만족시켰다.

Lab 08-1

5. RTL Viewer

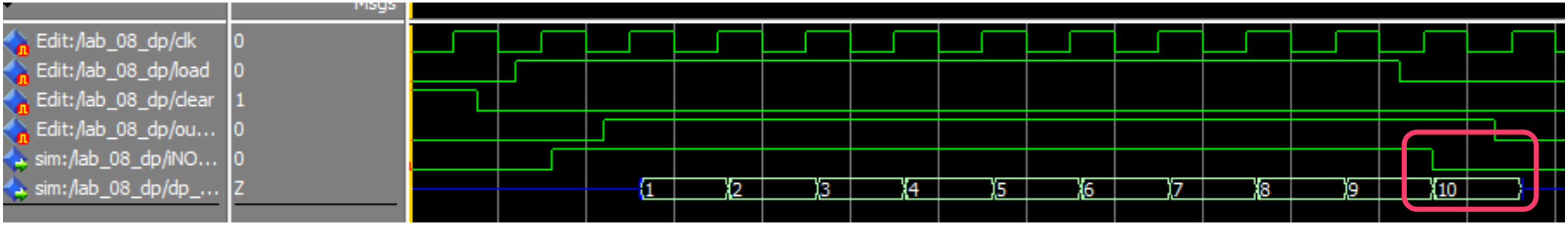


→ 해당 Code는 다음과 같은 RTL Viewer를 가진다.

→ 해당 RTL Viewer는 앞에 기능과 더불어 dp_out과 data increment 를 동시에 시키는 기능을 추가하였다.

Lab 08-1

6. Simulation



→ Clock의 rising edge에서 약간의 propagation delay이 후, Output이 바로 출력되었다.

→ 또한, 10이 나온 수 iNOT10은 0으로 바뀌어 더 이상 increment가 진행되는 것을 막았다.

Figure 9.40 Optimized control words for the counting algorithm using the datapath in Figure 9.27(a).

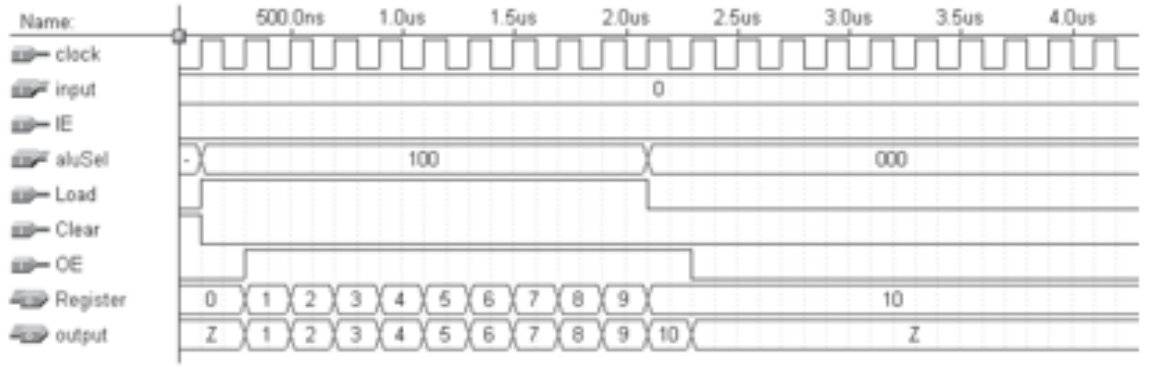
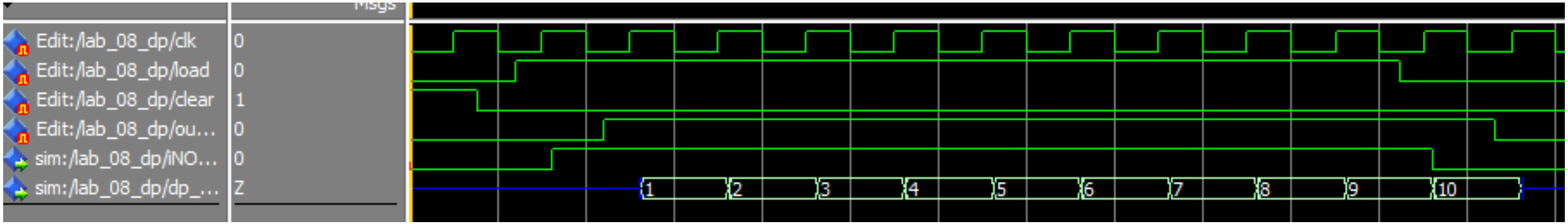


Figure 9.41 Corrected simulation trace for using the three control words from Figure 9.40.

Lab 08-1

6. Simulation



→ 다음 Simulation은 Figure 9.41과 동일하게 진행하였다.

→ 해당 Simulation을 확인하면 이전 Simulation과는 다르게 1이 증가하는 동시에 바로바로 Output이 나오는 것을 확인할 수 있다.

Figure 9.40 Optimized control words for the counting algorithm using the datapath in Figure 9.27(a).

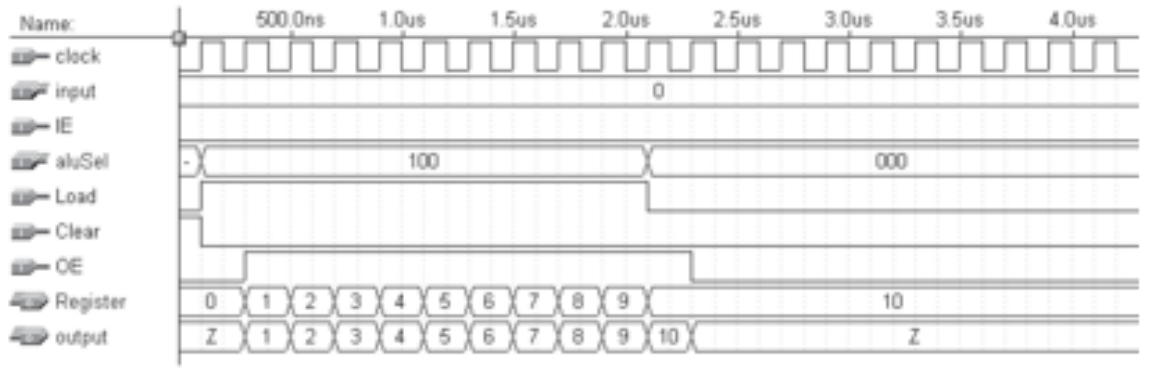


Figure 9.41 Corrected simulation trace for using the three control words from Figure 9.40.

Lab 08-1

7. VHDL Code – Datapath(오류)

```
1   LIBRARY IEEE;
2   USE IEEE.STD_LOGIC_1164.ALL;
3   use IEEE.STD_LOGIC_ARITH.ALL;
4   use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6   Entity lab_08_dp is
7      port(
8            clk     : IN STD_LOGIC ;
9            load    : IN STD_LOGIC ;
10           clear   : IN STD_LOGIC ;
11           out_sel : IN std_logic ;
12
13           iNOT10  : out std_logic;
14           dp_out  : out std_logic_vector(3 downto 0)
15
16           );
17  END lab_08_dp;
18
19  Architecture dataflow of lab_08_dp is
20
21     signal data : std_logic_vector(3 downto 0) := "0000"; --initialize
22     begin
23
24        process(clear, clk) is --register i(clocked process)
25           begin
26
27           if (clear = '1') then
28              data <= "0000"; --i initialize
29
30           elsif (clk'EVENT AND clk = '1') then
31              if (load = '1') then
32              data <= data + '1'; --adder
33                 if (out_sel = '1') then
34                    dp_out <= data; -- output
35                 else
36                    dp_out <= "ZZZZ";
37              end if;
38             end if;
39           end if;
40        end process;
41        iNOT10 <= '0' when (data = "1010") else '1'; --combinational logic
42  end dataflow;
```

→ 다음은 주어진 Datapath의 VHDL Code를 아래 표와 같이 Instruction을 바꾸어서 코딩하는 과정에서 발생한 오류 코드이다.

Control Word	Instruction	IE 6	$ALU_2ALU_1ALU_0$ 5–3	$Load$ 2	$Clear$ 1	OE 0
1	$i = 0$	×	× × ×	0	1	0
2	$i = i + 1$	0	100 (add)	1	0	0
3	$i = i + 1$ & OUTPUT i	0	100 (add)	1	0	1

Lab 08-1

7. VHDL Code – Datapath(오류)

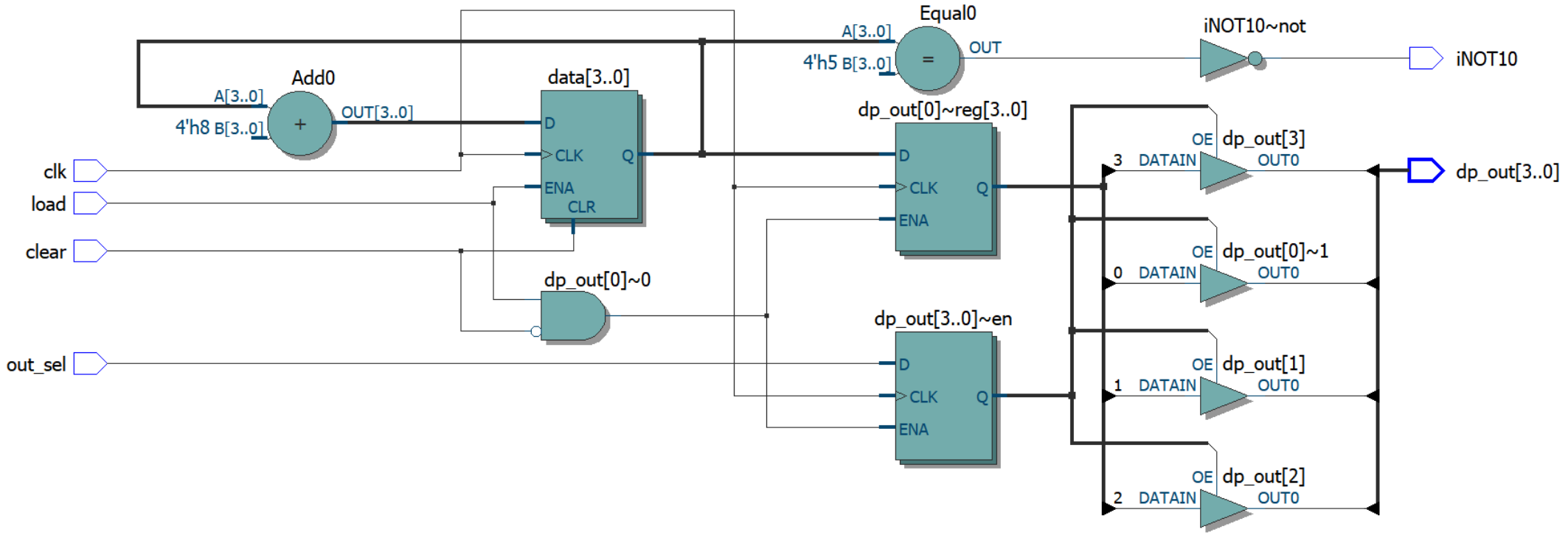
```
1   LIBRARY IEEE;
2   USE IEEE.STD_LOGIC_1164.ALL;
3   use IEEE.STD_LOGIC_ARITH.ALL;
4   use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6   Entity lab_08_dp is
7      port(
8            clk     : IN STD_LOGIC ;
9            load    : IN STD_LOGIC ;
10           clear   : IN STD_LOGIC ;
11           out_sel : IN std_logic ;
12
13           iNOT10  : out std_logic;
14           dp_out  : out std_logic_vector(3 downto 0)
15
16           );
17  END lab_08_dp;
18
19  Architecture dataflow of lab_08_dp is
20
21     signal data : std_logic_vector(3 downto 0) := "0000"; --initialize
22     begin
23
24        process(clear, clk) is --register i(clocked process)
25           begin
26
27           if (clear = '1') then
28              data <= "0000"; --i initialize
29
30           elsif (clk'EVENT AND clk = '1') then
31              if (load = '1') then
32              data <= data + '1'; --adder
33                 if (out_sel = '1') then
34                    dp_out <= data; -- output
35                 else
36                    dp_out <= "ZZZZ";
37              end if;
38             end if;
39           end if;
40        end process;
41        iNOT10 <= '0' when (data = "1010") else '1'; --combinational logic
42  end dataflow;
```

→ 해당 코드는 이전과 다르게 increment와 output을 동시에 시켜주는 과정에서 iNOT10을 고려하지 않고 코딩을 진행하였다.

Control Word	Instruction	*IE* 6	$ALU_2ALU_1ALU_0$ 5–3	*Load* 2	*Clear* 1	*OE* 0
1	$i = 0$	×	× × ×	0	1	0
2	$i = i + 1$	0	100 (add)	1	0	0
3	$i = i + 1$ & OUTPUT i	0	100 (add)	1	0	1

Lab 08-1

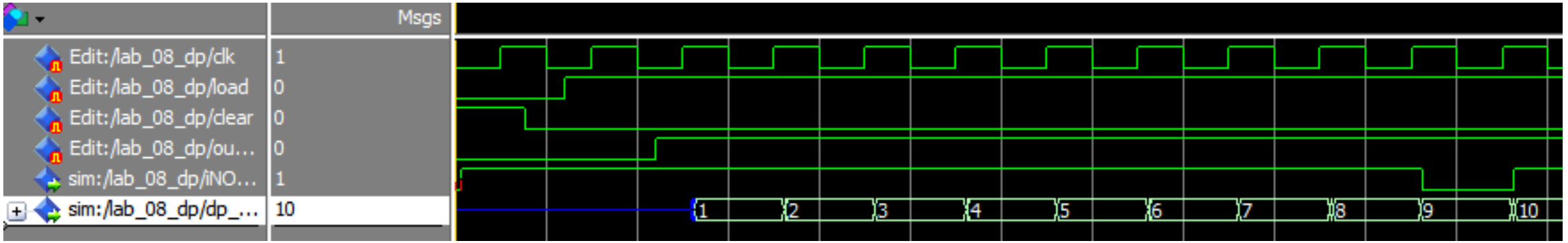
4. RTL Viewer(오류)



→ 위의 코드의 RTL Viewer는 다음과 같이 나온다.

Lab 08-1

3. Simulation(오류)



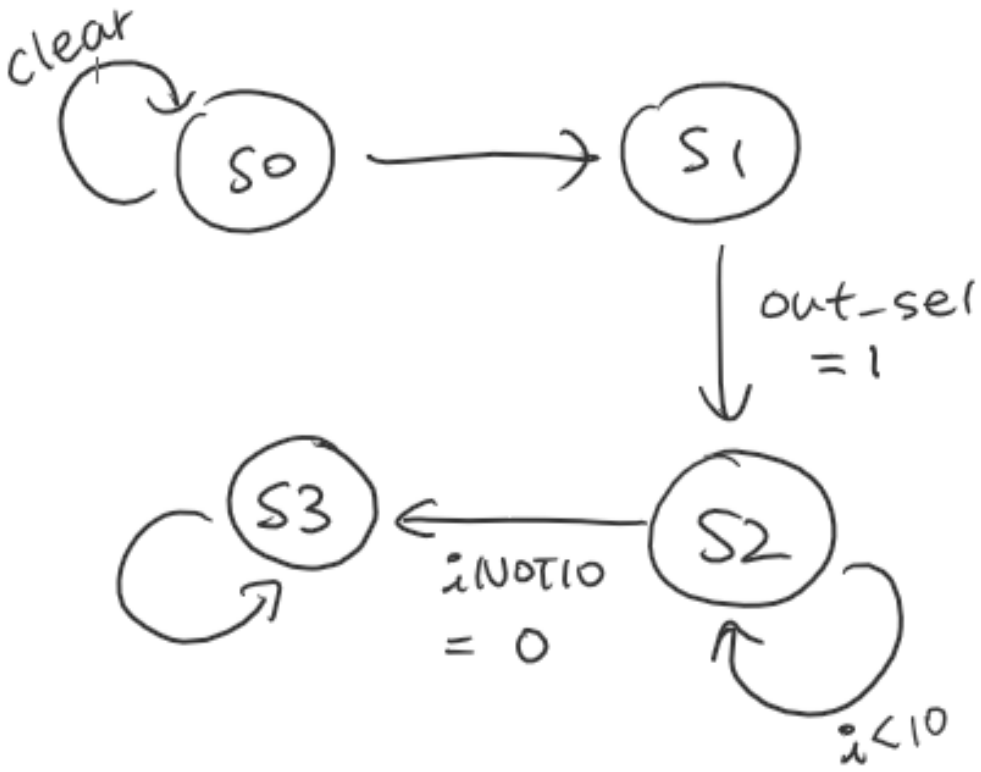
→ 해당 Code의 Simulation은 다음과 같이 진행되었다.

→ 이는 Clock밖에서 iNOT10이 출력되어 10이 아닌 9에서 0값으로 바뀐 것을 확인할 수 있었다.

→ 이를 통해 iNOT10을 Clock안에 넣어서 코딩을 해주었다.

Lab 08-2

1. State Diagram



→ 다음은 주어진 Code에 대한 State Diagram이다.

S0: i=0 (Clear or reset)
load=1

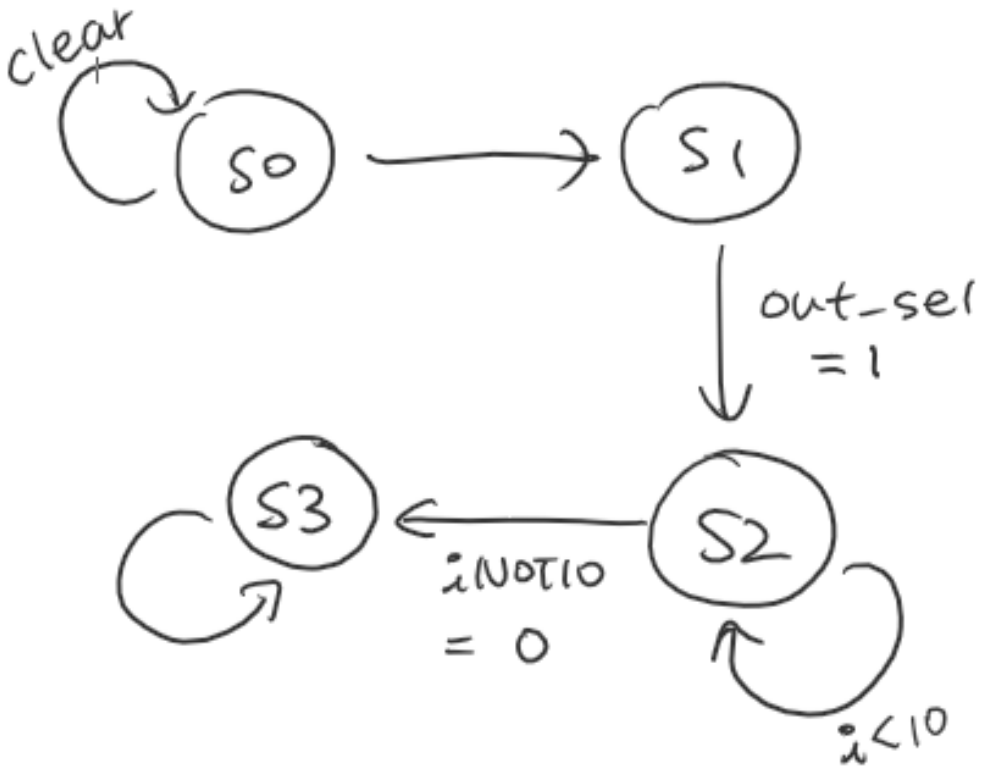
S1: i=i+1 (load=1)

S2: i=i+1 & OUTPUT i, iNOT10=1

S3: i=10, iNOT10=0

Lab 08-2

1. State Diagram



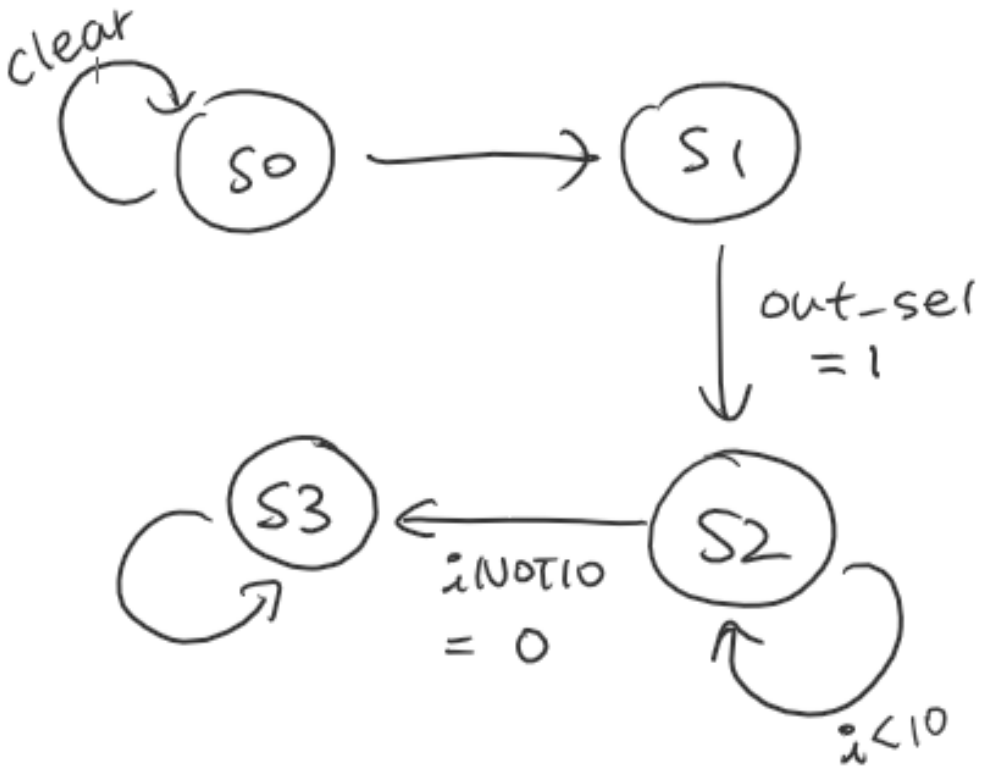
→ State가 s0일 때, clear를 해준 상태에서 load값을 1로 해준다.

→ State가 s1일 때, load 1을 유지하면서 i를 1씩 increment 시켜준다.

S0: i=0 (clear or reset)
load=1
S1: i=i+1 (load=1)
S2: i=i+1 & OUTPUT i, iNOT10=1
S3: i=10, iNOT10=0

Lab 08-2

1. State Diagram



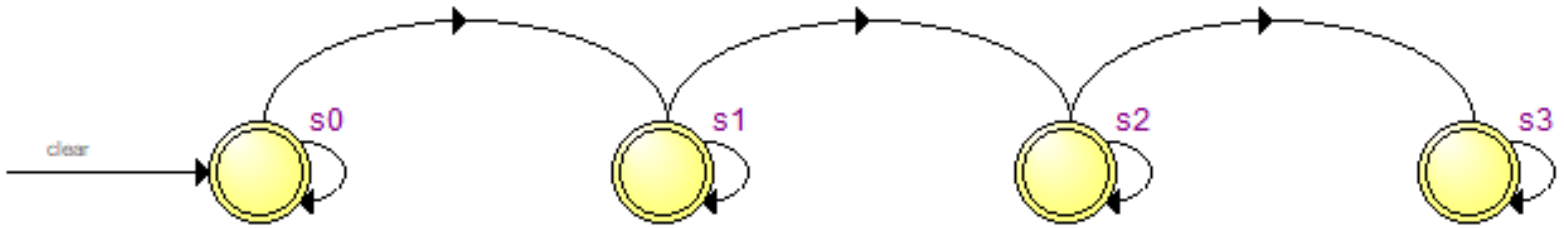
→ State가 s2일 때, i를 1씩 increment해주고 output을 출력해준다. iNOT10은 1인 상태이다.

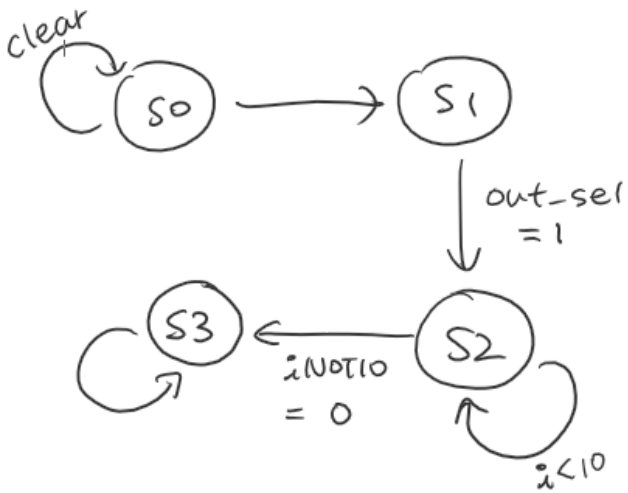
→ State가 s3일 때, data 가 10인 상태이므로 iNOT10을 0으로 한다.

S0: i=0 (Clear or reset)
load=1
S1: i=i+1 (load=1)
S2: i=i+1 & OUTPUT i, iNOT10=1
S3: i=10, iNOT10=0

Lab 08-2

1. State Diagram





→ 다음은 Code를 돌려 State Diagram을 출력한 결과이다. 이를 통해 앞에 State를 정확히 구현한 것을 알 수 있다.

Lab 08-2

1. VHDL Code – Control Unit

```
1   LIBRARY IEEE;
2   USE IEEE.STD_LOGIC_1164.ALL;
3   use IEEE.STD_LOGIC_ARITH.ALL;
4   use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6   Entity lab_08_2 is
7      port(
8            clk      : IN STD_LOGIC ;
9            load     : IN STD_LOGIC ;
10           clear    : IN STD_LOGIC ;
11           out_sel  : IN std_logic ;
12           iNOT10   : out std_logic;
13           dp_out   : out std_logic_vector(3 downto 0)
14           );
15  END lab_08_2;
16
17  Architecture dataflow of lab_08_2 is
18     TYPE state_type is (s0, s1, s2, s3);
19     signal state : state_type;
20     signal data : std_logic_vector(3 downto 0) := "0000"; --initialize
21     begin
22        process(clear, clk) is --register i(clocked process)
23           begin
24           if clear = '1' then
25              state <= s0;
26           elsif (clk'event AND clk = '1') then
27              CASE state is
28                 WHEN s0 =>
29                    if (load = '1') then
30                       data <= "0000"; --i initialize
31                       state<=s1;
32                    end if;
33                 WHEN s1 =>
34                    if (load = '1') then
35                       data <= data + '1'; --adder
36                       dp_out<=data;
37                       state<=s2;
38                    elsif (out_sel='0') then
39                       dp_out <= "ZZZZ";
40                    end if;
41                 WHEN s2 =>
42                    if (load = '1' AND out_sel = '1' AND data/="1010") then
43                       data <= data + '1'; --adder
44                       dp_out <= data;
45                    end if;
46                    if(data = "1010") then
47                       dp_out <= data;
48                       state<=s3;
49                       iNOT10 <= '0';
50                    else
51                       iNOT10 <= '1';
52                    end if;
53                 WHEN s3 =>
54                    if (out_sel = '0') then
55                       dp_out <= "ZZZZ";
56                    state<= s3;
57                    end if;
58           END CASE;
59        end if;
60     end process;
61  end dataflow;
```

→ 다음은 State를 추가한 Code이다.

→ Control Unit을 구현하였다.

Lab 08-2

1. VHDL Code – Control Unit

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

Entity lab_08_2 is
   port(
         clk      : IN STD_LOGIC ;
         load     : IN STD_LOGIC ;
         clear    : IN STD_LOGIC ;
         out_sel  : IN std_logic ;
         iNOT10   : out std_logic;
         dp_out   : out std_logic_vector(3 downto 0)
         );
END lab_08_2;

Architecture dataflow of lab_08_2 is
   TYPE state_type is (s0, s1, s2, s3);
   signal state : state_type;
   signal data : std_logic_vector(3 downto 0) := "0000"; --initialize
   begin
      process(clear, clk) is --register i(clocked process)
         begin
         if clear = '1' then
            state <= s0;
         elsif (clk'event AND clk = '1') then
            CASE state is
               WHEN s0 =>
                  if (load = '1') then
                     data <= "0000"; --i initialize
                     state<=s1;
                  end if;
               WHEN s1 =>
                  if (load = '1') then
                     data <= data + '1'; --adder
                     dp_out<=data;
                     state<=s2;
                  elsif (out_sel='0') then
                     dp_out <= "ZZZZ";
                  end if;
```

→ 다음 Input과 Output은 앞에 코드와 동일하게 작성하였다.

→ State는 s0, s1, s2, s3로 4가지이다.

→ 먼저 clear를 통해 state를 s0으로 reset해준다.

→ 다음 코드는 clock의 rising edge에서 진행된다.

Lab 08-2

1. VHDL Code – Control Unit

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

Entity lab_08_2 is
   port(
         clk      : IN STD_LOGIC ;
         load     : IN STD_LOGIC ;
         clear    : IN STD_LOGIC ;
         out_sel  : IN std_logic ;
         iNOT10   : out std_logic;
         dp_out   : out std_logic_vector(3 downto 0)
         );
END lab_08_2;

Architecture dataflow of lab_08_2 is
   TYPE state_type is (s0, s1, s2, s3);
   signal state : state_type;
   signal data : std_logic_vector(3 downto 0) := "0000"; --initialize
   begin
      process(clear, clk) is --register i(clocked process)
         begin
         if clear = '1' then
            state <= s0;
         elsif (clk'event AND clk = '1') then
            CASE state is
               WHEN s0 =>
                  if (load = '1') then
                     data <= "0000"; --i initialize
                     state<=s1;
                  end if;
               WHEN s1 =>
                  if (load = '1') then
                     data <= data + '1'; --adder
                     dp_out<=data;
                     state<=s2;
                  elsif (out_sel='0') then
                     dp_out <= "ZZZZ";
                  end if;
```

→ So일 때 load가 1이면 data는 초기값인 0000을 가지고 다음 state인 s1으로 넘어간다.

→ S1일 때 load가 1이면 data를 1 increment해주고 data_out에 data를 반영해준다. 그리고 다음 State를 s2로 넘어간다.

→ 만약, out_sel이 0이면 dp_out은 ZZZZ를 출력한다.

Lab 08-2

1. VHDL Code – Control Unit

```
41              WHEN s2 =>
42                 if (load = '1' AND out_sel = '1' AND data/="1010") then
43                    data <= data + '1'; --adder
44                    dp_out <= data;
45                 end if;
46                 if(data = "1010") then
47                    dp_out <= data;
48                    state<=s3;
49                    iNOT10 <= '0';
50                 else
51                    iNOT10 <= '1';
52                 end if;
53              WHEN s3 =>
54                 if (out_sel = '0') then
55                    dp_out <= "ZZZZ";
56                 state<= s3;
57                 end if;
58           END CASE;
59        end if;
60     end process;
61  end dataflow;
```

→ S2일 때, load가 1이고, out_sel이 1이고 data가 1010이 아닌 경우, 1을 증가해주고, output에 data를 반영해준다.

→ 그리고 data가 1010이면 10인 값을 출력해주고, iNOT10은 0으로 바뀌고, state는 s3로 넘어간다.

→ 만약 모두 해당이 안되면, iNOT10은 1이다.

Lab 08-2

1. VHDL Code – Control Unit

```
41              WHEN s2 =>
42                 if (load = '1' AND out_sel = '1' AND data/="1010") then
43                    data <= data + '1'; --adder
44                    dp_out <= data;
45                 end if;
46                 if(data = "1010") then
47                    dp_out <= data;
48                    state<=s3;
49                    iNOT10 <= '0';
50                 else
51                    iNOT10 <= '1';
52                 end if;
53              WHEN s3 =>
54                 if (out_sel = '0') then
55                    dp_out <= "ZZZZ";
56                 state<= s3;
57                 end if;
58           END CASE;
59        end if;
60     end process;
61  end dataflow;
```

→ S3일 때, out_sel이 0이면, dp_out은 ZZZZ를 출력해주고, state는 s3에 머문다.

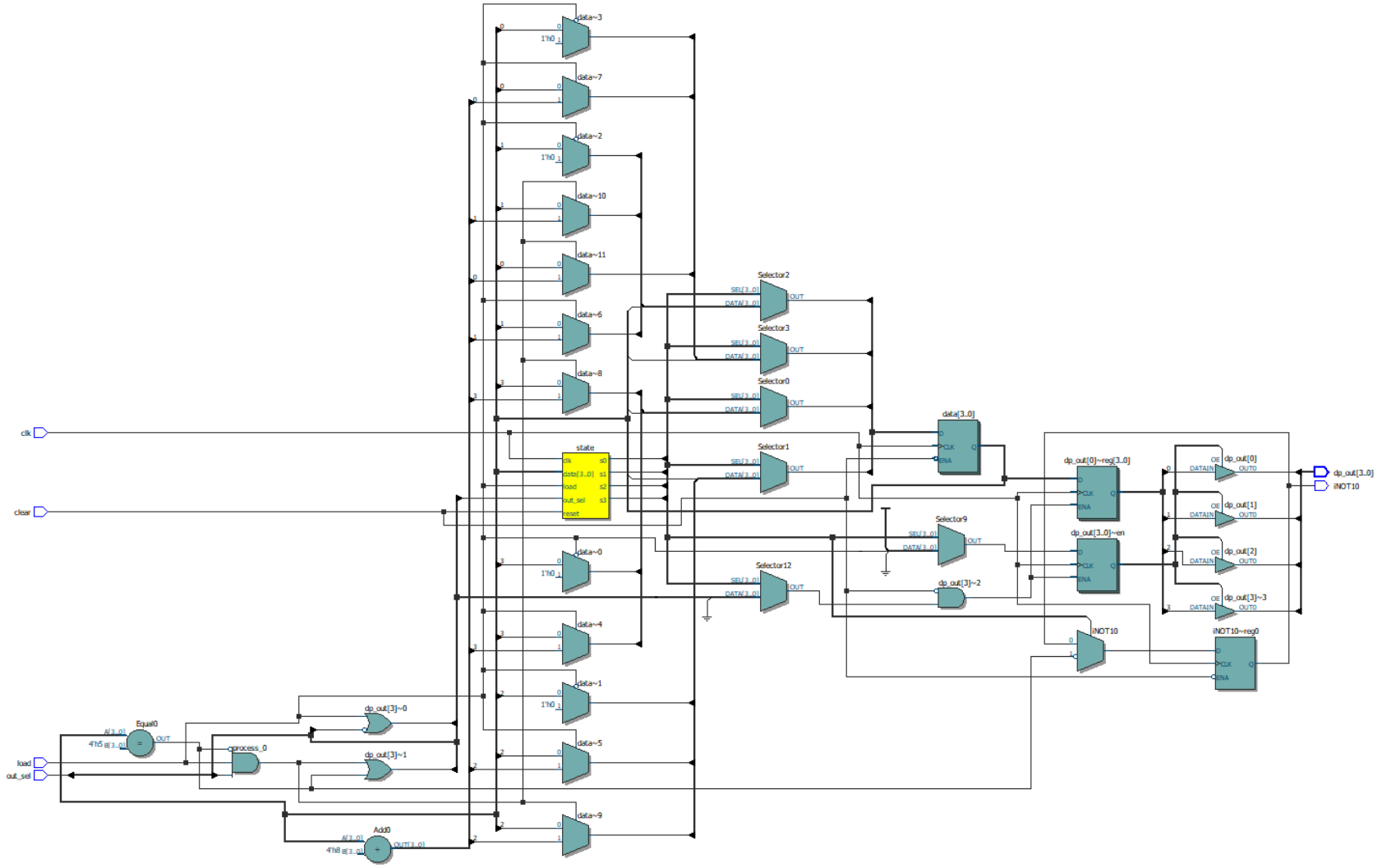
→ 이와 같은 state로 다음 Control Unit을 진행한다.

Lab 08-2

2. RTL Viewer

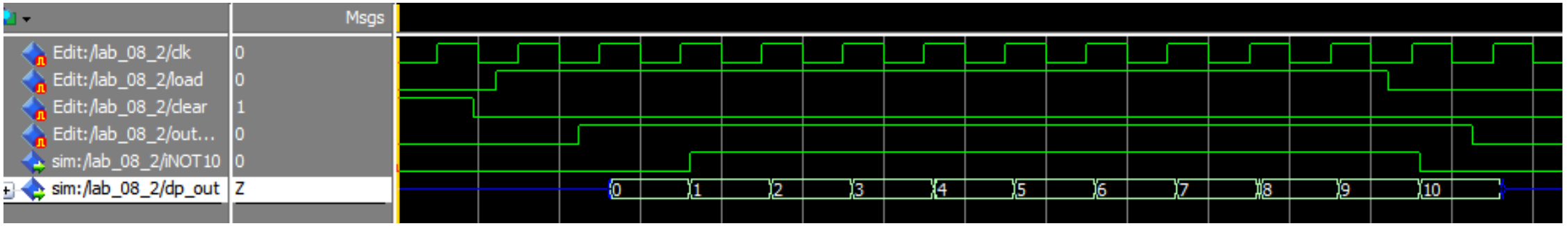
→ 다음은 해당 Code의

RTL Viewer이다.



Lab 08-2

3. Simulation



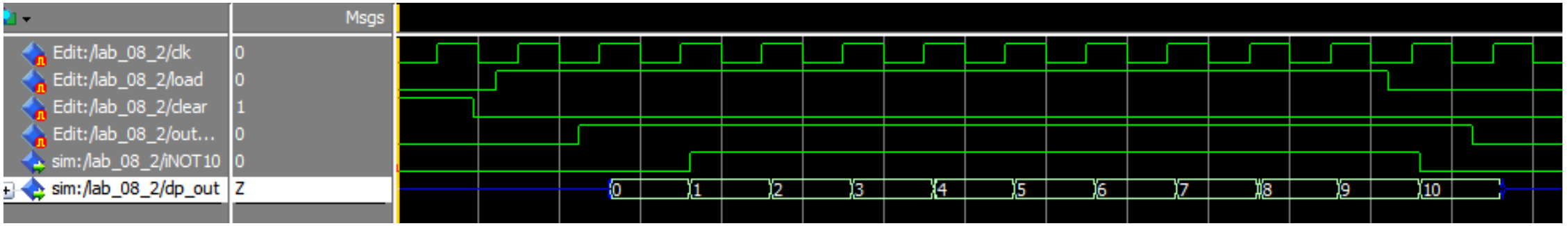
→ 다음은 위의 Code를 Simulation한 결과이다.

→ 해당 Simulation은 위의 8-1 Simulation과 동일하게 진행하였다.

→ 결과가 동일한 것을 확인할 수 있다.

Lab 08-2

3. Simulation



→ 다음 output은 1씩 증가하였으며, 그때 iNOT10은 1로 출력되었다.

→ 또한, output이 10이 나온 다음에는 iNOT10은 0으로 제대로 바뀐 것을 확인할 수 있다.

→ 하지만 앞에 코드와는 다르게 0값이 출력되었는데, 이 이유를 정확히 파악할 수는 없었다. 예상으로는 s1에서 output을 출력함으로 인해 1이 증가되지 않은 상태에서 나오지 않았나 생각해보지만, 순차적으로 진행되는 과정이므로 1부터 나올 것으로 예상되어 이는 오류가 아니라고 생각했다.

4. Discussion

→ 이번 Simulation은 코드가 주어져 비교적 쉽게 코딩을 하였지만, 사소한 부분 하나하나로 인해 많은 시간이 소요되었다.

→ 이를 통해 IF문의 구조를 체계적으로 써야 한다는 것과, Clock이 진행되는 상황에 맞게 하나하나 Simulation을 진행해야 한다는 것을 알 수 있었다.

→ 또한, State를 하는 과정에서 여러 if문 구조를 통해 많은 오류도 있었고, 아예 output이 나오지 않는 경우도 있었지만, 차근차근 State를 만들어가면서 이를 해결하였다.

4. Discussion

→ 또한, 마지막 Simulation에서 알 수 없이 0 값이 나오게 되어 다방면으로 코드를 수정해 보았지만, 이를 해결하지 못하였다.

→ 하지만, 이를 통해 많은 과정을 거쳐보았고 State에 따른 구조에 대해 더 잘 알 수 있었다.

→ 이번 실습은 State에 대해 이해하고, ALU에 대해서 더 잘 이해할 수 있던 실습이었다. 다음에 더 많은 시간이 주어진다면 마지막 0에 대해서 더 고찰할 수 있는 시간이 필요할 것 같다고 생각된다.