Lab 03. One-Counter Design

201810800 이혜인

Design an “one-counter” with the state machine design technique.

- Inputs:
  - A: 8 bit data
  - Load_A (LA)
  - S: Start counting indicator
- Outputs:
  - B: 4 bit data
  - Done
- Function:
  - If ‘Load’ is ‘true’ then receive 8 bit data input A
  - If S is ‘1’then count ‘1’s in A’s bitstream, and output the number to B, and set Done to ‘1’. (example: if A is 11010010 then B=4.)

Refer Pseudo-code, timing diagram, & Shift register code given.

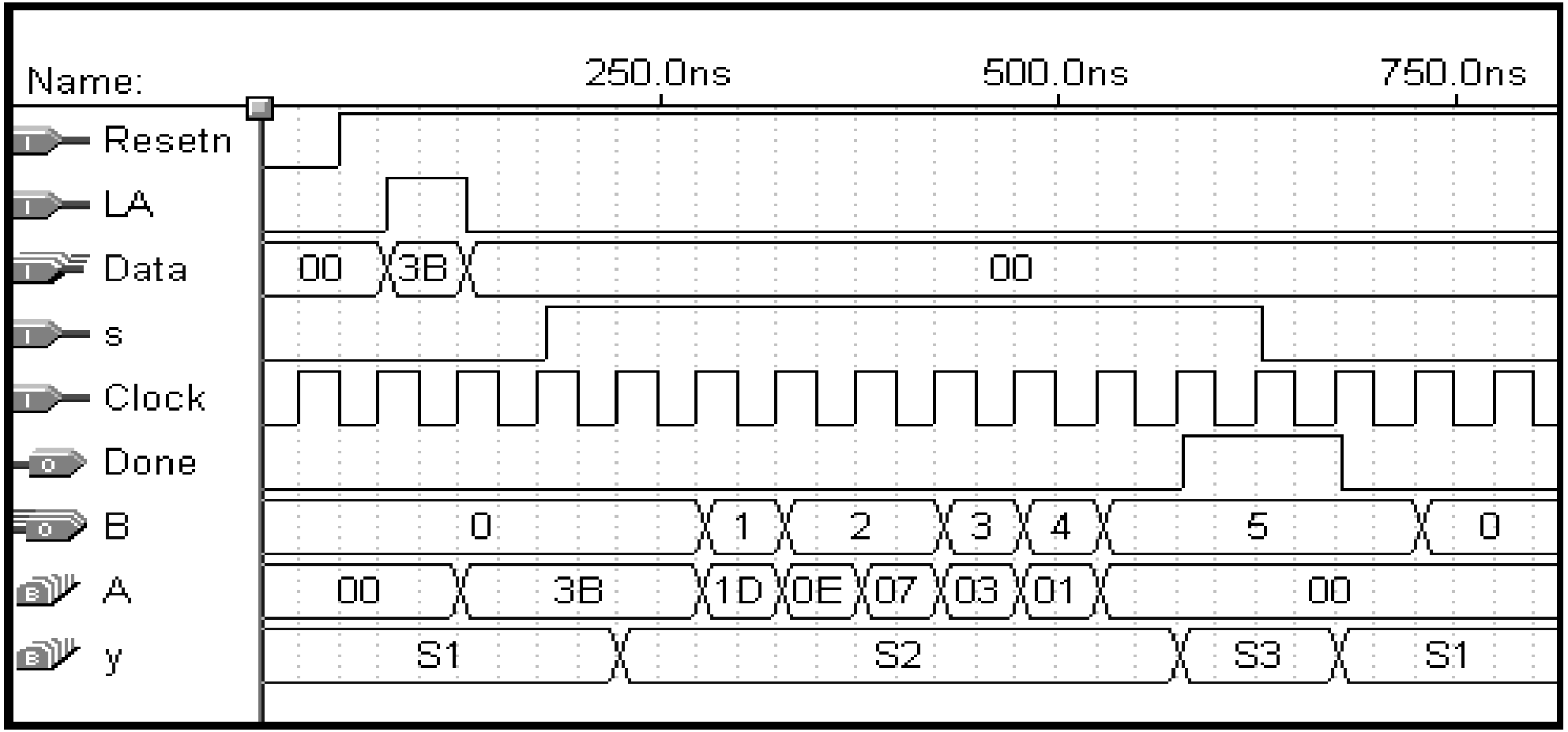
Report should include:

1. VHDL code
2. Simulation capture
3. Problems met during design & Solutions
4. Discussion

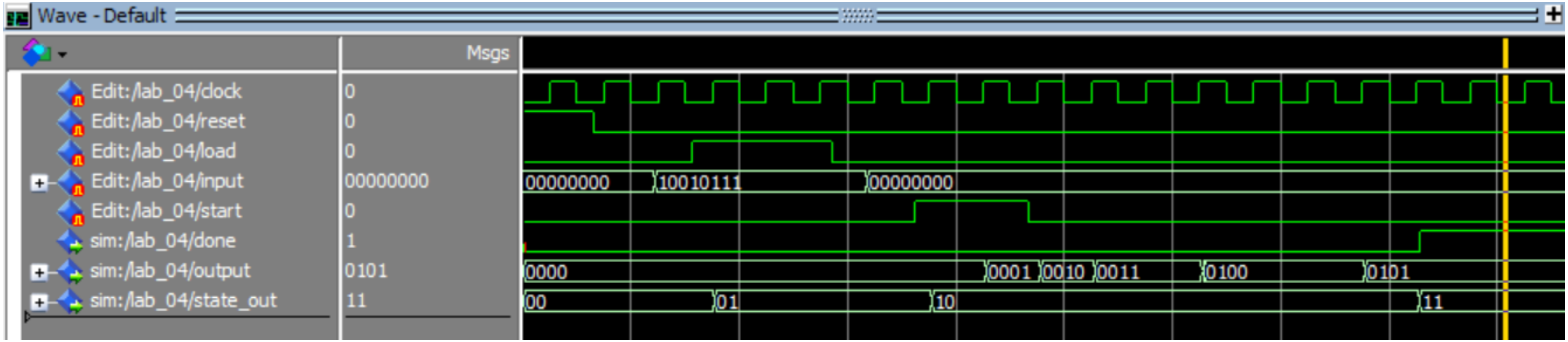
Pseudo-code for the one-counter

```
B = 0;
while A≠0 do
        if a0 = 1 then
                B = B +
1;
        end if;
        Right-shift A;
end while ;
```

Expected behavior of the bit counter



Expected behavior of the bit counter



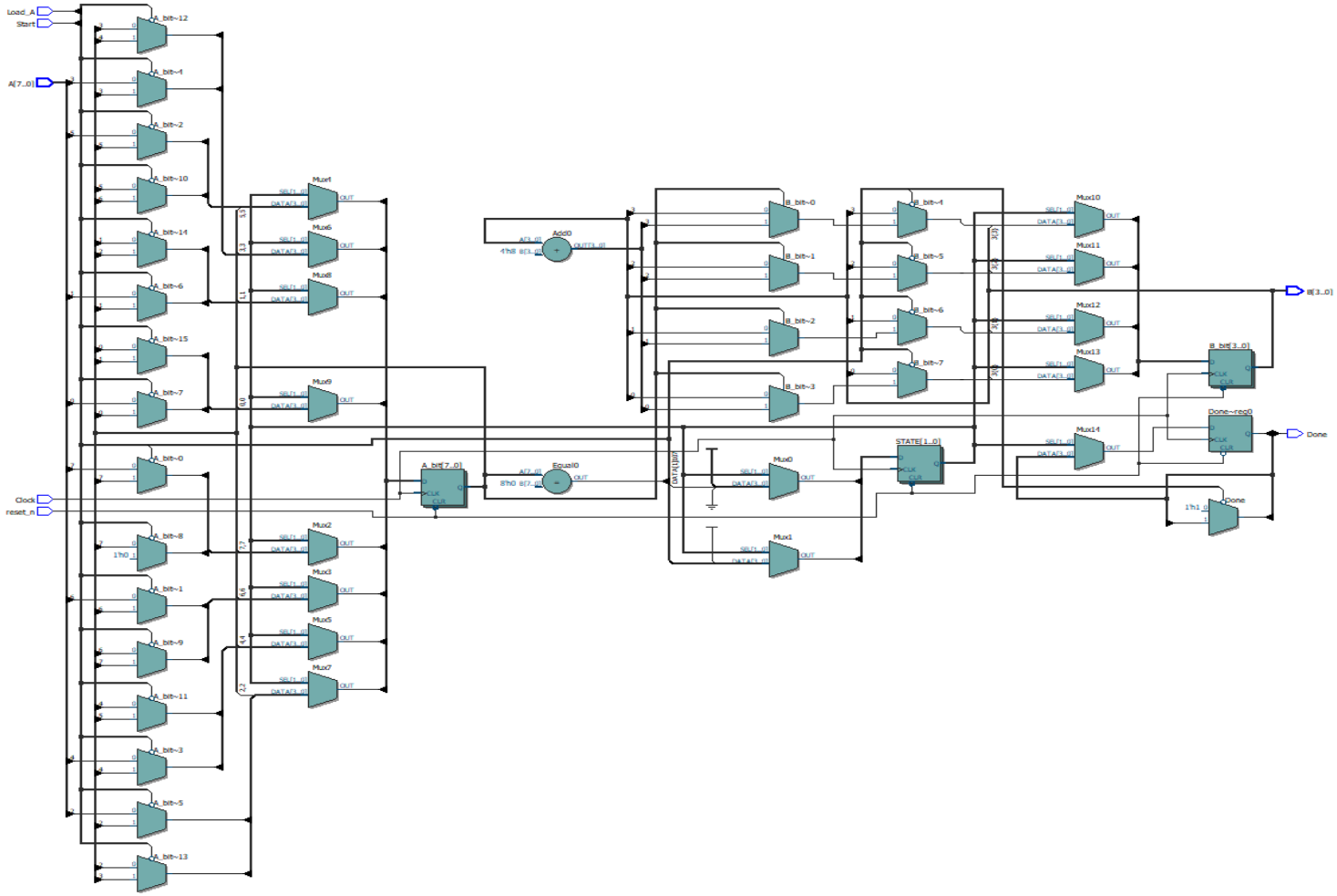
1. VHDL Code

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;
3  USE ieee.std_logic_unsigned.all;
4
5  ENTITY lab_03 IS
6  PORT( Clock       :IN STD_LOGIC ;
7        Load_A,Start:IN STD_LOGIC ;
8        reset_n     :IN STD_LOGIC ;
9        A           :IN STD_LOGIC_VECTOR (7 DOWNTO 0) ;
10       B           :OUT STD_LOGIC_VECTOR (3 DOWNTO 0) ;
11       Done        :OUT STD_LOGIC) ;
12 END lab_03;
13
14 ARCHITECTURE Behavior OF lab_03 IS
15    SIGNAL STATE      :STD_LOGIC_VECTOR (1 DOWNTO 0); --
16    SIGNAL A_bit      :STD_LOGIC_VECTOR (7 DOWNTO 0);
17    SIGNAL B_bit      :STD_LOGIC_VECTOR (3 DOWNTO 0);
18
19 BEGIN
20    PROCESS (Clock, reset_n, A, A_bit)
21    BEGIN
22       IF reset_n='0' THEN
23          STATE<="00"; --reset
24          Done<='0';   --reset
25          B_bit<=(OTHERS=>'0'); --reset
26          A_bit<=(OTHERS=>'0'); --reset
27       ELSIF (Clock'EVENT AND Clock='1') THEN
28          CASE STATE IS
29             WHEN "00"=>
30                if Load_A= '0' then --IF LOAD is False
31                   STATE<= "00";
32                else                --IF LOAD is True
33                   A_bit<=A;        --receive 8 bit data input A
34                   STATE<= "01";
35                end if;
36             WHEN "01"=>
37                if Start= '0' then
38                   STATE<= "01";
39                else
40                   STATE<= "10";
41                end if;
42             WHEN "10"=>
43                IF A_bit/="00000000" THEN --while A≠0
44                   STATE<="10";
45                   IF(A_bit(0)='1') THEN --A_bit is 1
46                      B_bit<=B_bit+1;    --count+1
47                   END IF;
48                   A_bit<='0'&A_bit(7 DOWNTO 1);  --Change A_bit=0, Right-shift A
49                ELSE
50                   Done<='1';
51                   STATE<="11";
52                END IF;
53             WHEN "11"=>
54                STATE<="11";
55          END CASE;
56       END IF;
57    END PROCESS;
58 B<=B_bit;          --Output
59 END Behavior;      --Finish
```

→ 다음은 VHDL Code이다.

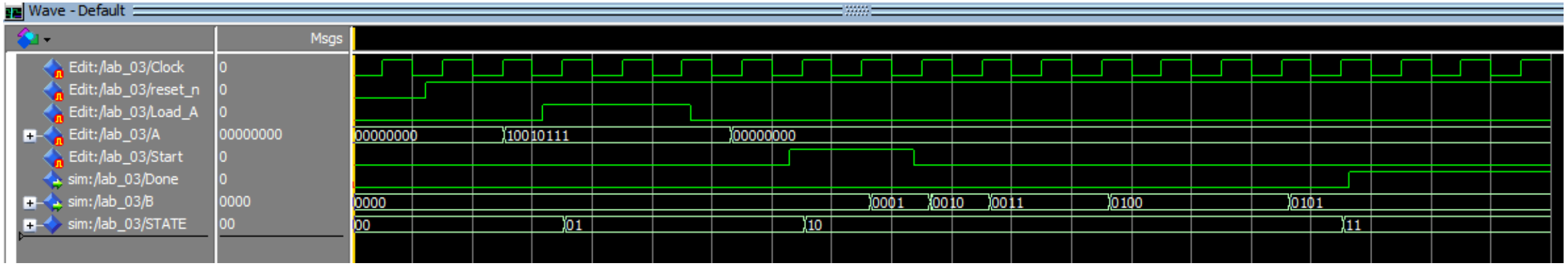
→ 주어진 Input, Output, Function을 기반으로 하여 구현하였다.

1. VHDL Code – RTL Viewer



→ 앞에 작성한 VHDL Code를 RTL Viewer로 표현하면 다음과 같이 나온다.

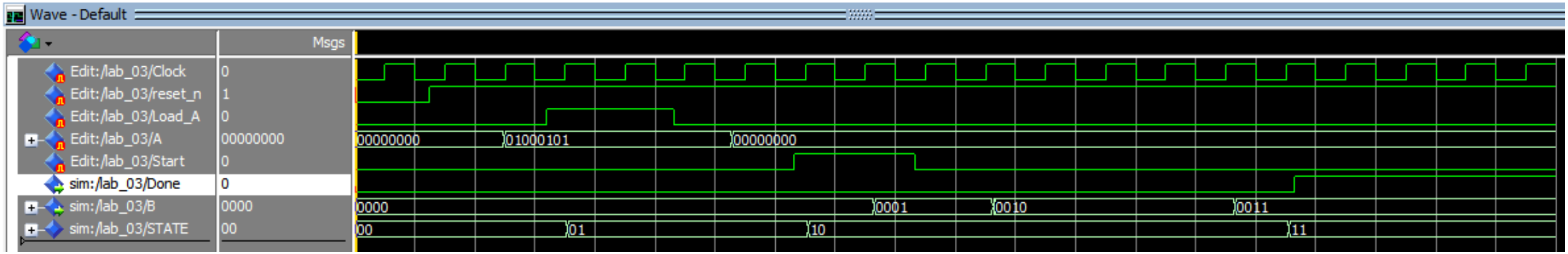
2. Simulation capture



→ 해당 Simulation은 앞에 주어진 자료인 Expected behavior of the bit counter와 동일하게 진행한 것으로 결과가 동일하게 나왔음을 확인할 수 있다.

→ **Load_A가 1이 되면 State가 01로 바뀌고, Start가 1이 되면 State가 01에서 10으로 바뀌면서 A에 있는 '1'을 Count하고 이를 B에 저장한다. 그리고 Count가 완료되면 State가 11로 바뀜과 동시에 Done이 1이 된다. 해당 Output을 Clock에 동기화 되므로 모두 Clock rising edge에서 결과를 확인할 수 있다.**

2. Simulation capture



→ 해당 Simulation은 앞에 Simulation과 다른 값을 넣어 진행하였다.

→ **Input A에 01000101을 입력하였으므로 Output B는 3인 $0011_{(2)}$이 나와야 하므로, 위의 Simulation을 확인하면 올바르게 진행된 것을 알 수 있다.**

→ **또한, Load_A가 1이 되면 State가 01로 바뀌고, Start가 1이 되면 State가 01에서 10으로 바뀌면서 A에 있는 '1'을 Count하고 이를 B에 저장한다. 그리고 Count가 완료되면 State가 11로 바뀜과 동시에 Done이 1이 되는 것을 확인 할 수 있다.**

3. Problems met during design & Solutions

① Unsigned install

→ 처음에 'USE ieee.std_logic_unsigned.all;'을 install하지 않았다. 이로 인해 operator가 정의되지 않아서 'B<=B+1' 부분에서 연산자 오류가 발생하였다. 이는 Unsigned를 install해줌으로서 해결하였다.

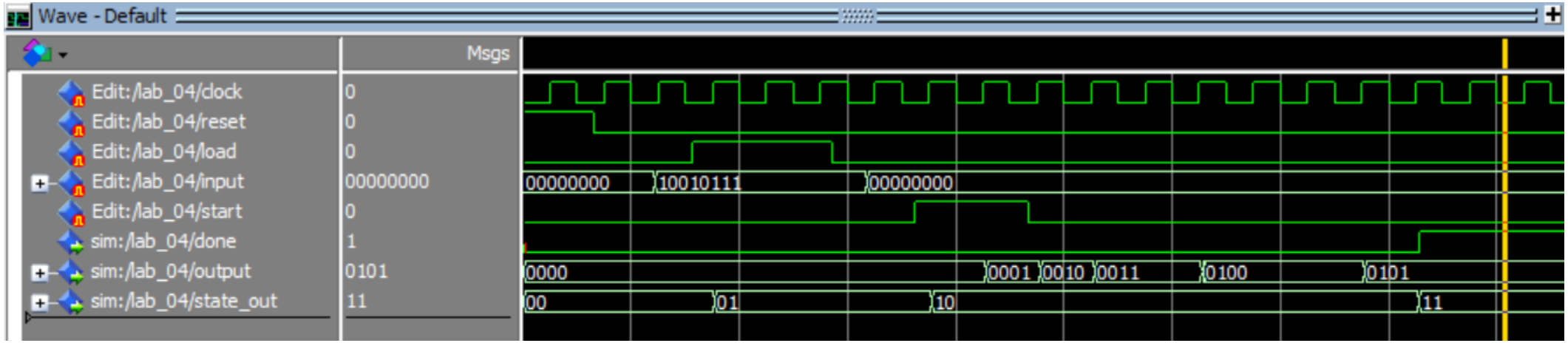
② Case 구문

→ Case 구문에 모든 State의 경우를 다 작성하지 않아서 오류가 발생하였다. 처음에 State '11'상태를 작성해주지 않았는데 이로 인해 'Case Statement choices must cover all possible values of expression' 오류가 발생하였고 State '11'을 추가해줌으로서 이 문제를 해결하였다.

3. Problems met during design & Solutions

③ State

→ State를 변경할 때, 주어진 조건이 없어 어떻게 해야 할 지 고민하게 되었다.



→ 하지만 주어진 Simulation 예시를 보고 Load가 1이 되었을 때는 State를 01로, State가 01일 때 Start가 1이 되면 State를 10으로, 마지막으로 Count가 완료되었을 때, State가 11이 되는 것을 확인하고 이를 이용하여 VHDL 코드를 구현하였다.

3. Problems met during design & Solutions

④ Error (10028): Can't resolve multiple constant drivers for net...

→ "Multiple Constant Drivers" Error Verilog with Quartus Prime이 발생하였다. 이는 처음 작성한 VHDL Code는 여러 PROCESS로 구성하여 이를 연결한 형태였다.

→ 그러다 보니 같은 INPUT, OUTPUT을 여러 번 정의 내려 사용하게 되었다. 그리고 이를 실행할 때, 같은 INPUT, OUPUT들 사이에서 충돌이 일어나 오류가 발생하게 되었다.

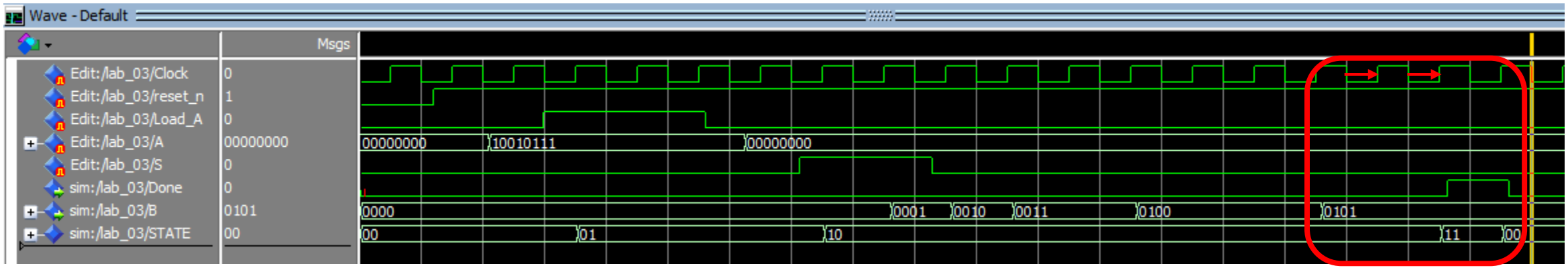
→ 이는 INPUT, OUTPUT을 중복되어 실행되지 않게 꼭 필요한 곳에만 작성하여 해결하였다.

→ (Reset이나 Clock을 너무 여러 번 정의 내림.)

3. Problems met during design & Solutions

⑤ State 11이 2개의 Clock 뒤에 나오는 현상

→ 원래 State는 Clock에 동기화 되므로 Clock의 rising edge에서 결과를 볼 수 있는 것이 맞다. 하지만 초기 코드의 경우 여러 개의 PROCESS를 작성하여서 done을 1로 만드는 z라는 변수로 인해 한 Clock 다음에 Done이 1의 값을 가지게 되고, 동시에 State가 11을 가진다는 것을 알 수 있다. 하지만 이미 한 Clock이 지난 다음에 알게 되었으므로 한 클럭이 더 밀려서 2개의 Clock 뒤에 나오는 현상이 발생하게 된 것이다. **↓다음은 해당 Simulation 결과이다.**



3. Problems met during design & Solutions

```
1   LIBRARY ieee;
2   USE ieee.std_logic_1164.all;
3   USE ieee.std_logic_unsigned.all;
4
5   ENTITY lab_03 IS
6   PORT( Clock       :IN STD_LOGIC ;
7         Load_A,Start:IN STD_LOGIC ;
8         reset_n     :IN STD_LOGIC ;
9         A           :IN STD_LOGIC_VECTOR (7 DOWNTO 0) ;
10        B           :OUT STD_LOGIC_VECTOR (3 DOWNTO 0) ;
11        Done        :OUT STD_LOGIC) ;
12  END lab_03;
13
14  ARCHITECTURE Behavior OF lab_03 IS
15     SIGNAL STATE      :STD_LOGIC_VECTOR (1 DOWNTO 0);
16     SIGNAL A_bit      :STD_LOGIC_VECTOR (7 DOWNTO 0);
17     SIGNAL B_bit      :STD_LOGIC_VECTOR (3 DOWNTO 0);
18     SIGNAL z,RA,CB    :STD_LOGIC; -- z:make done 1, RA:st
19
20  BEGIN
21  State_Transitions:
22     PROCESS (Clock, reset_n)
23     BEGIN
24        IF reset_n='0' THEN
25           STATE<="00";
26        ELSIF (Clock'EVENT AND Clock='1') THEN
27           CASE STATE IS
28              WHEN "00"=>
29                 if Load_A= '0' then
30                    STATE<= "00";
31                 else
32                    STATE<= "01";
33                 end if;
34              WHEN "01"=>
35                 if Start= '0' then
36                    STATE<= "01";
37                 else
38                    STATE<= "10";
39                 end if;
40              WHEN "10"=>
41                 if z= '0' then
42                    STATE<= "10";
43                 else
44                    STATE<= "11";
45                 end if;
46              WHEN "11"=>
47                 STATE<="11";
48           END CASE;
49        END IF;
50     END PROCESS State_Transitions;
51
52  Control_Outputs: --RA:start read A, CB:start count b->b+1
53     PROCESS (STATE,Start,A_bit(0))
54     BEGIN
55        Done<='0';
56        IF STATE="00" THEN
57           RA<='0';
58           CB<='0';
59
60        ELSIF STATE="10" THEN
61           RA<='1'; --start read bit
62           IF A_bit(0)='1' THEN
63              CB<='1';
64           ELSE
65              CB<='0';
66           END IF;
67        ELSIF STATE="11" THEN
68           RA<= '0';
69           CB<= '0';
70           Done<='1';
71        END IF;
72     END PROCESS Control_Outputs;
73
74  Datapath:
75     PROCESS (Clock)
76     BEGIN
77        IF (Clock'EVENT AND Clock='1') THEN
78           IF STATE="00" THEN
79              B_bit<="0000";
80           ELSE
81              if CB= '1'              --if bit is one
82                 then B_bit<=B_bit+1; --count
83              end if;
84           END IF;
85        END IF;
86     END PROCESS Datapath;
87
88  ShiftA:
89     PROCESS(Clock,A,Load_A,Start,Clock) --Load_A->onebit
90     BEGIN
91        IF (Clock'EVENT AND Clock='1') THEN
92           IF (Load_A='1') THEN
```

```
90     BEGIN
91        IF (Clock'EVENT AND Clock='1') THEN
92           IF (Load_A='1') THEN
93           A_bit<=A;
94           ELSE
95              IF RA='1' THEN
96                 A_bit<='0'&A_bit(7 DOWNTO 1);
97                 IF A_bit="00000000" THEN
98                    z<='1'; --make done:1
99                 ELSE
100                   z<='0'; --still repeat
101                END IF;
102             END IF;
103          END IF;
104       END IF;
105    END PROCESS ShiftA;
106
107 B<=B_bit;       --Output
108 END Behavior;    --Finish
```

→ 앞서 말한 PROCESS가 여러 개인 VHDL Code이다.

3. Problems met during design & Solutions

→ **각 프로세스는 다음과 같다.**

→ State_Transition : State 바꾸기

→ Control_Outputs : A를 읽어 1이면 CB를 1로 바꿔 Count할 수 있는 조건을 만듦(a_0 = 1)

→ Datapath : Count 할 수 있는 조건이 되면 +1을 함(B<=B+1)

→ ShiftA : A를 Shift하여 A가 00000000이 아닌지 확인하고 00000000이 아니면 오른쪽으로 Shift함(Right-shift A)

4. Discussion

→ Counter를 이해하느라 생각보다 많은 시간이 소요되었지만, 이해하고 나니 코드는 보다 수월하게 작성할 수 있었다. 다만, 여러 PROCESS로 나눠서 작성하다 보니 불필요한 변수가 많이 필요하게 되었고 이 과정에서 중복되는 변수로 인한 ERROR와 Delay가 한 번 더 일어나는 등 많은 시행착오를 겪게 되었다.

→ 하지만, 이로 인해 VHDL Code와 Counter의 구조에 대해서는 좀 더 상세하게 알 수 있는 계기가 되었다. 그리고 많은 ERROR를 수정하는 요령도 좀 더 많이 생겨나게 된 것 같았다.

4. Discussion

→ 또한, 여러 PROCESS 기능 별로 나눠서 작성하는 것이 좋지만은 않다는 것도 알게 되었다. 처음에는 기능별로 나눠서 쓰는 것이 더 알아보기 쉽고 간편할 것이라고 생각하였는데, 이로 인해 많은 불필요한 변수가 생기고 코드 자체가 길어져서 한눈에 알아보기도 어려워졌다.

→ 이번 실습을 통해서는 문제를 정확히 파악하고 이를 어떻게 해결하는 것이 좋은 방법인지에 대해 배운 것 같았다. 또한, VHDL Code를 작성하는 법에도 조금 더 가까워진 것 같다는 느낌이 든다.