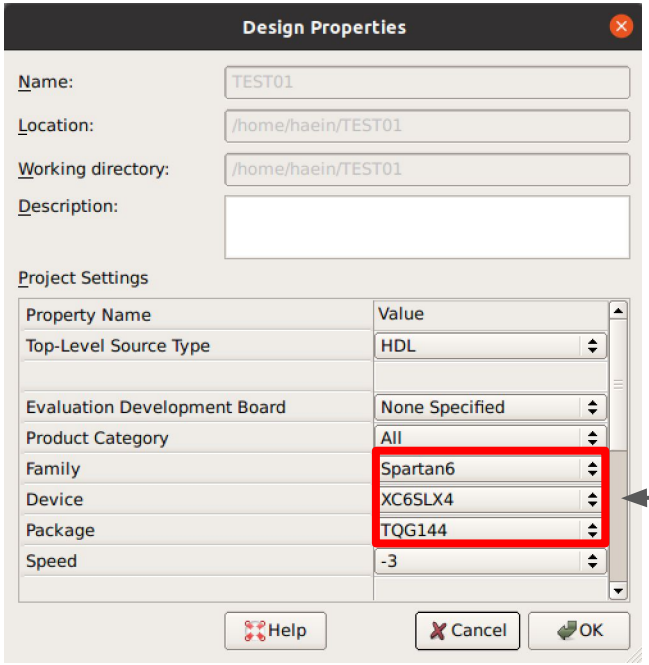
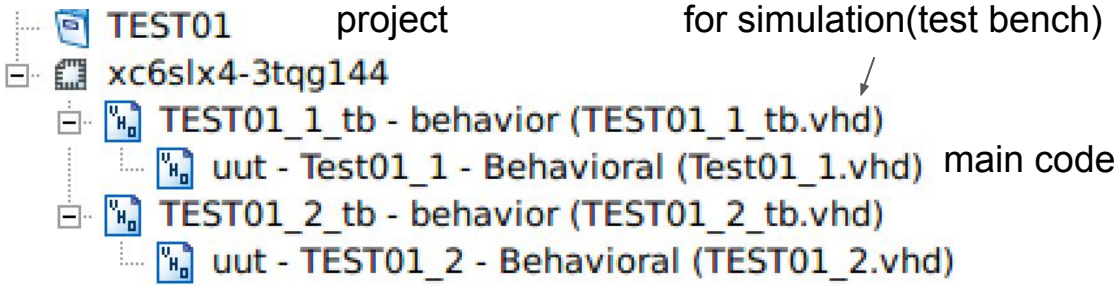
FPGA VHDL

Haein Lee

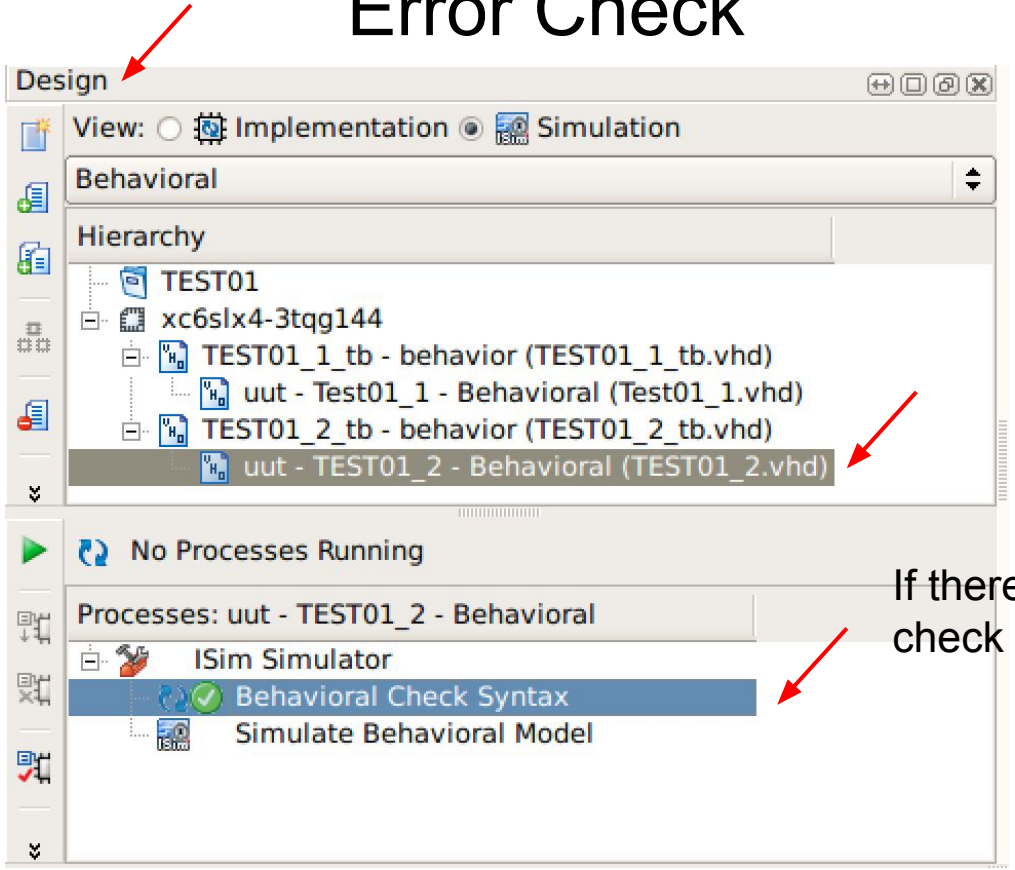
Basic Structure

New project





Error Check



If there is no error, the green check mark will appear.

Example 1 - Multiplexer

Main code

```
2  library ieee;
3  use ieee.std_logic_1164.all;
4  use ieee.std_logic_arith.all;
5  use ieee.std_logic_unsigned.all;
6
7  entity TEST01_2 is
8     port(INPUT : in STD_LOGIC_VECTOR(3 downto 0);
9          SEL : in STD_LOGIC_VECTOR(1 downto 0);
10         OUTPUT : out STD_LOGIC);
11 end TEST01_2;
12
13 architecture Behavioral of TEST01_2 is
14 begin
15    OUTPUT <= INPUT(0) when SEL = "00"
16    else INPUT(1) when SEL = "01"
17    else INPUT(2) when SEL = "10"
18    else INPUT(3);
19 end Behavioral;
```

대문자 소문자 구분 X

Example 1 - Multiplexer

Test Bench

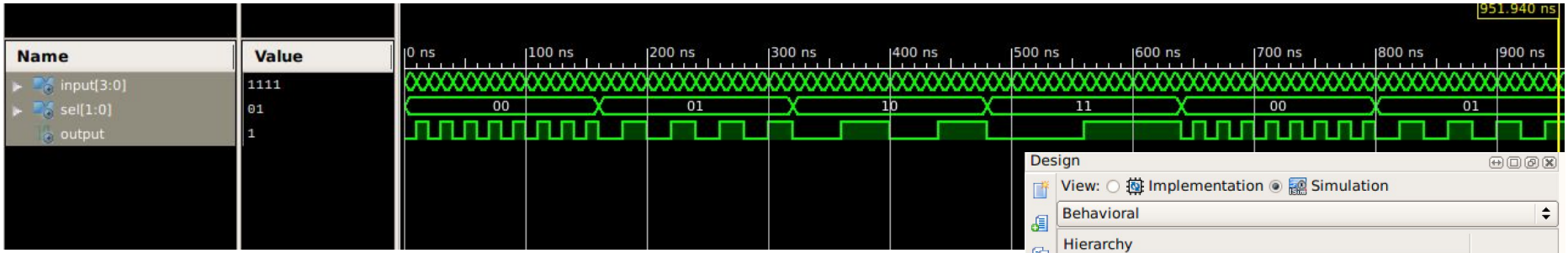
```
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.ALL;
4  use ieee.std_logic_arith.all;
5  use ieee.std_logic_unsigned.all;
6
7
8  ENTITY TEST01_2_tb IS
9  END TEST01_2_tb;
10
11 ARCHITECTURE behavior OF TEST01_2_tb IS
12     COMPONENT TEST01_2
13     PORT(
14          INPUT : IN  std_logic_vector(3 downto 0);
15          SEL : IN  std_logic_vector(1 downto 0);
16          OUTPUT : OUT  std_logic
17         );
18     END COMPONENT;
19
20    signal INPUT : std_logic_vector(3 downto 0) := (others => '0');
21    signal SEL : std_logic_vector(1 downto 0) := (others => '0');
22    signal OUTPUT : std_logic;
23
24 BEGIN
25    uut: TEST01_2 PORT MAP (
26           INPUT => INPUT,
27           SEL => SEL,
28           OUTPUT => OUTPUT
29         );
30
```

```
31    INPUT0:process is
32    begin
33       INPUT(0)<='0';
34       wait for 10 ns;
35       INPUT(0)<='1';
36       wait for 10 ns;
37    end process;
38
39    INPUT1:process is
40    begin
41       INPUT(1)<='0';
42       wait for 20 ns;
43       INPUT(1)<='1';
44       wait for 20 ns;
45    end process;
46
47    INPUT2:process is
48    begin
49       INPUT(2)<='0';
50       wait for 40 ns;
51       INPUT(2)<='1';
52       wait for 40 ns;
53    end process;
54
55    INPUT3:process is
56    begin
57       INPUT(3)<='0';
58       wait for 80 ns;
59       INPUT(3)<='1';
60       wait for 80 ns;
61    end process;
```

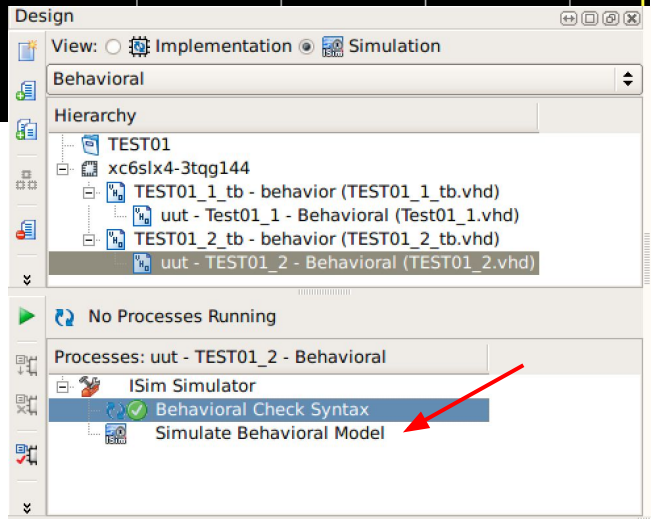
```
63    SEL0: process is
64    begin
65       SEL(0)<='0';
66       wait for 160 ns;
67       SEL(0)<='1';
68       wait for 160 ns;
69    end process;
70
71    SEL1: process is
72    begin
73       SEL(1)<='0';
74       wait for 320 ns;
75       SEL(1)<='1';
76       wait for 320 ns;
77    end process;
78
79
80 END;
```

Example 1 - Multiplexer

Simulation Result



입력	선택단자		출력
	SEL(1)	SEL(0)	OUTPUT
INPUT(0)	0	0	INPUT(0)
INPUT(1)	0	1	INPUT(1)
INPUT(2)	1	0	INPUT(2)
INPUT(3)	1	1	INPUT(3)



Example 2 - Demultiplexer

SEL0	SEL1	OUT0	OUT1	OUT2	OUT3
0	0	INPUT	0	0	0
1	0	0	INPUT	0	0
0	1	0	0	INPUT	0
1	1	0	0	0	INPUT

$$O0 = I\overline{S_0}\overline{S_1},\ O1 = IS_0\overline{S_1},\ O2 = \overline{IS_0}S_1,\ O3 = IS_0S_1$$

논리 연산 표시 : A and B = AB
A or B = A+B
not A = $\bar{A}$

Example 2 - Demultiplexer

Main code

```
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4
5  entity TEST01_3 is
6     Port(INPUT : in STD_LOGIC;
7          SEL : in STD_LOGIC_VECTOR(1 downto 0);
8          OUTPUT : out STD_LOGIC_VECTOR(3 downto 0));
9  end TEST01_3;
10
11 architecture Behavioral of TEST01_3 is
12 begin
13       OUTPUT(0) <= INPUT and not SEL(0) and not SEL(1);
14       OUTPUT(1) <= INPUT and SEL(0) and not SEL(1);
15       OUTPUT(2) <= INPUT and not SEL(0) and SEL(1);
16       OUTPUT(3) <= INPUT and SEL(0) and SEL(1);
17
18 end Behavioral;
19
20
```

Example 2 - Demultiplexer

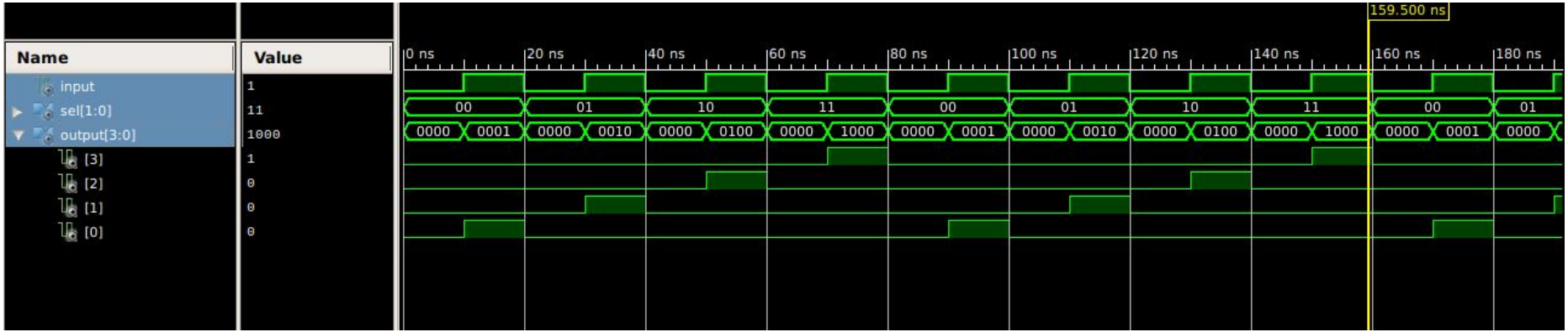
Test Bench

```
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.ALL;
4
5
6  ENTITY TEST01_3_tb IS
7  END TEST01_3_tb;
8
9  ARCHITECTURE behavior OF TEST01_3_tb IS
10     COMPONENT TEST01_3
11     PORT(
12          INPUT : IN  std_logic;
13          SEL : IN  std_logic_vector(1 downto 0);
14          OUTPUT : OUT  std_logic_vector(3 downto 0)
15        );
16     END COMPONENT;
17
18    signal INPUT : std_logic := '0';
19    signal SEL : std_logic_vector(1 downto 0) := (others => '0');
20    signal OUTPUT : std_logic_vector(3 downto 0);
21
22 BEGIN
23    uut: TEST01_3 PORT MAP (
24           INPUT => INPUT,
25           SEL => SEL,
26           OUTPUT => OUTPUT
27         );
```

```
28    INPUT_SIG : process is
29    begin
30       INPUT <= '0';
31       wait for 10 ns;
32       INPUT <= '1';
33       wait for 10 ns;
34    end process;
35
36    SEL0 : process is
37    begin
38       SEL(0) <= '0';
39       wait for 20 ns;
40       SEL(0) <= '1';
41       wait for 20 ns;
42    end process;
43
44    SEL1 : process is
45    begin
46       SEL(1) <= '0';
47       wait for 40 ns;
48       SEL(1) <= '1';
49       wait for 40 ns;
50    end process;
51
52
53 END;
```

Example 2 - Demultiplexer

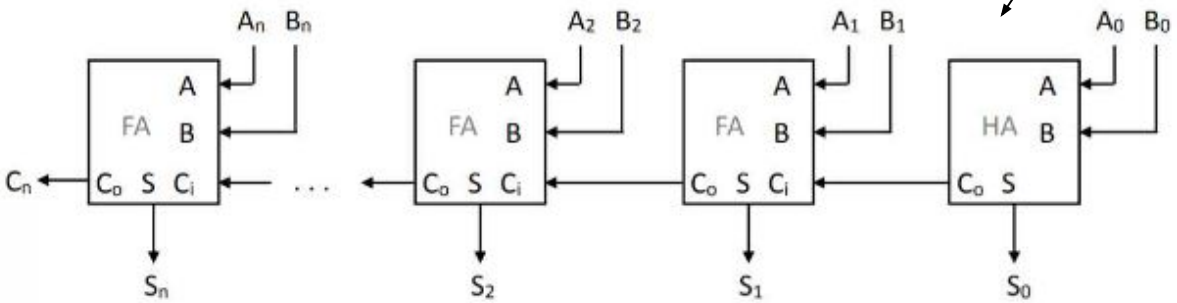
Simulation Result

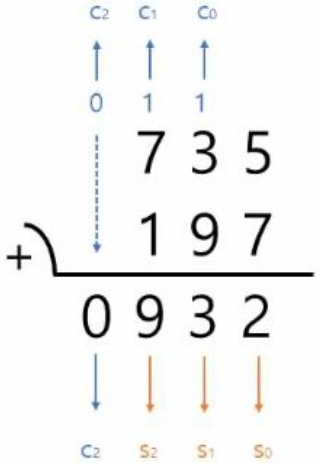


Example 3 - Full Adder

Code for adding two binary numbers

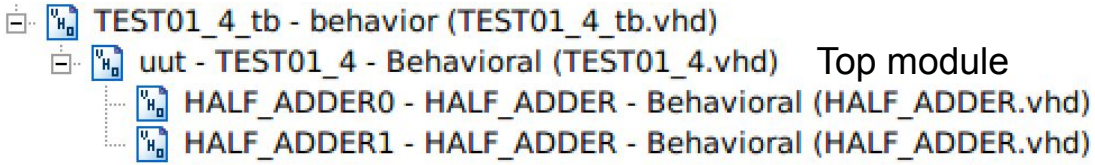
Combination of half adder and full adders





In decimal notation

Example 3 - Full Adder



Top module

Component module

Top module

```
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4
5  entity TEST01_4 is
6     port(INPUT : in std_logic_vector(1 downto 0);
7          CARRY_INPUT : in std_logic;
8          SUM : out std_logic;
9          CARRY_OUTPUT : out std_logic);
10 end TEST01_4;
11
12 architecture Behavioral of TEST01_4 is
13    component HALF_ADDER
14       port(INPUT : in std_logic_vector(1 downto 0);
15            SUM : out std_logic;
16            CARRY : out std_logic);
17    end component;
18
19    signal SUM_TEMP : std_logic;
20    signal CARRY_TEMP : std_logic;
21    signal CARRY_RESULT : std_logic;
22
23 begin
24    HALF_ADDER0 : HALF_ADDER port map(INPUT(0)=>INPUT(0), INPUT(1)=>INPUT(1),
25    SUM=>SUM_TEMP, CARRY=>CARRY_TEMP);
26    HALF_ADDER1 : HALF_ADDER port map(INPUT(0)=>SUM_TEMP, INPUT(1)=>CARRY_INPUT,
27    SUM=>SUM, CARRY=>CARRY_RESULT);
28
29    process(CARRY_TEMP, CARRY_RESULT) is
30    begin
31       CARRY_OUTPUT <= CARRY_TEMP or CARRY_RESULT;
32    end process;
33 end Behavioral;
```

Example 3 - Full Adder

Component module

```
 2  library IEEE;
 3  use IEEE.STD_LOGIC_1164.ALL;
 4
 5  entity HALF_ADDER is
 6     port(INPUT : in std_logic_vector(1 downto 0);
 7          SUM : out std_logic;
 8          CARRY : out std_logic);
 9  end HALF_ADDER;
10
11  architecture Behavioral of HALF_ADDER is
12
13  begin
14     process(INPUT)
15     begin
16        SUM<=INPUT(0) xor INPUT(1);
17        CARRY<=INPUT(0) and INPUT(1);
18     end process;
19  end Behavioral;
20
21
```

Example 3 - Full Adder

Test Bench

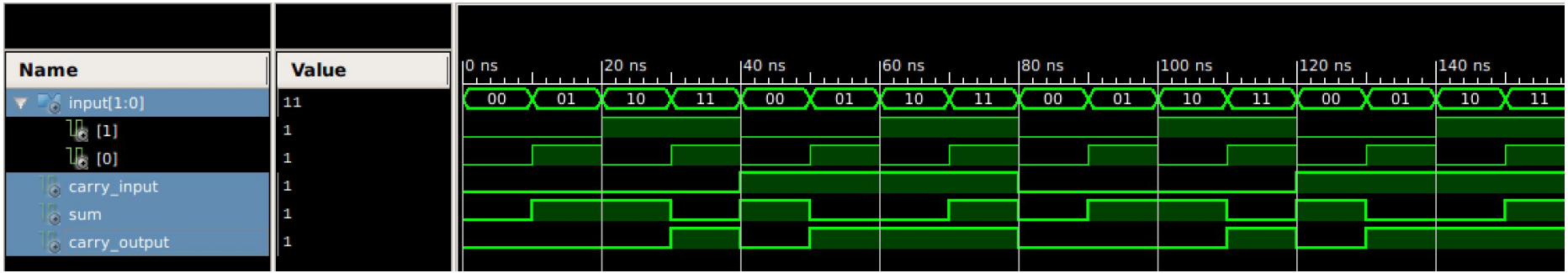
```
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.ALL;
4
5  ENTITY TEST01_4_tb IS
6  END TEST01_4_tb;
7
8  ARCHITECTURE behavior OF TEST01_4_tb IS
9      COMPONENT TEST01_4
10     PORT(
11          INPUT : IN  std_logic_vector(1 downto 0);
12          CARRY_INPUT : IN  std_logic;
13          SUM : OUT  std_logic;
14          CARRY_OUTPUT : OUT  std_logic
15         );
16     END COMPONENT;
17
18    signal INPUT : std_logic_vector(1 downto 0) := (others => '0');
19    signal CARRY_INPUT : std_logic := '0';
20    signal SUM : std_logic;
21    signal CARRY_OUTPUT : std_logic;
22
```

```
23 BEGIN
24    uut: TEST01_4 PORT MAP (
25            INPUT => INPUT,
26            CARRY_INPUT => CARRY_INPUT,
27            SUM => SUM,
28            CARRY_OUTPUT => CARRY_OUTPUT
29          );
30
31    INPUT0 : process is
32    begin
33       INPUT(0) <= '0';
34       wait for 10 ns;
35       INPUT(0) <= '1';
36       wait for 10 ns;
37    end process;
38
39    INPUT1 : process is
40    begin
41       INPUT(1) <= '0';
42       wait for 20 ns;
43       INPUT(1) <= '1';
44       wait for 20 ns;
```

```
45    end process;
46
47    CARRY_IN : process is
48    begin
49       CARRY_INPUT <= '0';
50       wait for 40 ns;
51       CARRY_INPUT <= '1';
52       wait for 40 ns;
53    end process;
54 END;
```

Example 3 - Full Adder

Simulation Result



Sequential Logic(순차 논리 회로)

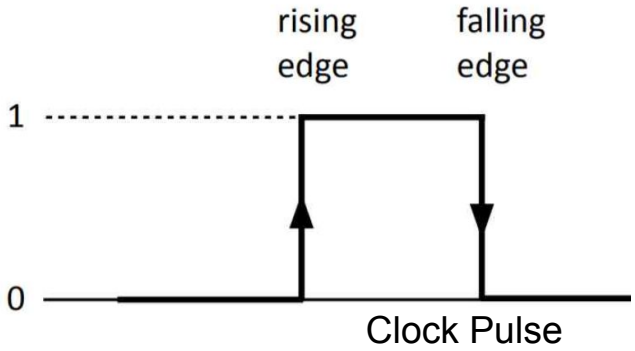
- 이전의 상태에 대한 정보를 가지고 있기에, 이 정보에 따라 출력이 결정된다. (같은 입력에도 다른 출력이 나올 수 있음)

⟺ 조합 논리 회로 : 오직 입력신호에 의해 출력이 결정됨(이전 예시들이 모두 조합 회로임)

ex) RS Flip-Flop, JK Flip-Flop, T Flip-Flop, D Flip-Flop

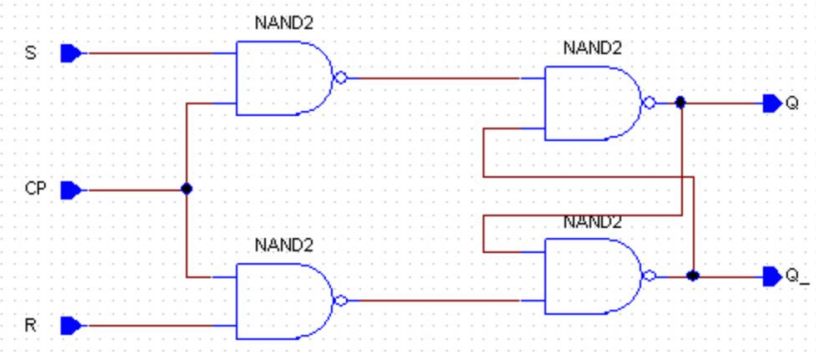
정보 저장, 기억 소자(1bit 저장 가능)

Flip-Flop에서 보통 Clock의 rising edge 이용



RS Flip-Flop

Clock Pulse



- Set, Reset 모두 0이면 이전 값 유지
- Set만 1이면 출력 1
- Reset만 1이면 출력 0
- Set, Reset 모두 1이면 출력값 모름

		입력 전	입력 후
Set	Reset	Q_t	Q_{t+1}
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	?
0	0	1	1
0	1	1	0
1	0	1	1
1	1	1	?

RS Flip-Flop

Main Code

```
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4  use ieee.std_logic_arith.all;
5  use ieee.std_logic_unsigned.all;
6
7  entity TEST02_1 is
8     port(set,reset,clock,rst : in std_logic;
9          Q,Qbar : out std_logic);
10 end TEST02_1;
11
12 architecture Behavioral of TEST02_1 is
13 begin
14    process(clock,set,reset,rst)
15    begin
16       if(rst = '1') then Q<='0';
17       elsif(Rising_edge(clock)) then
18          if(set /= reset)then
19          Q <=set;
20          Qbar <= reset;
21          elsif(set='1' and reset='1')then
22          Q<='Z';
23          Qbar<='Z';
24          end if;
25       end if;
26    end process;
27 end Behavioral;
28
```

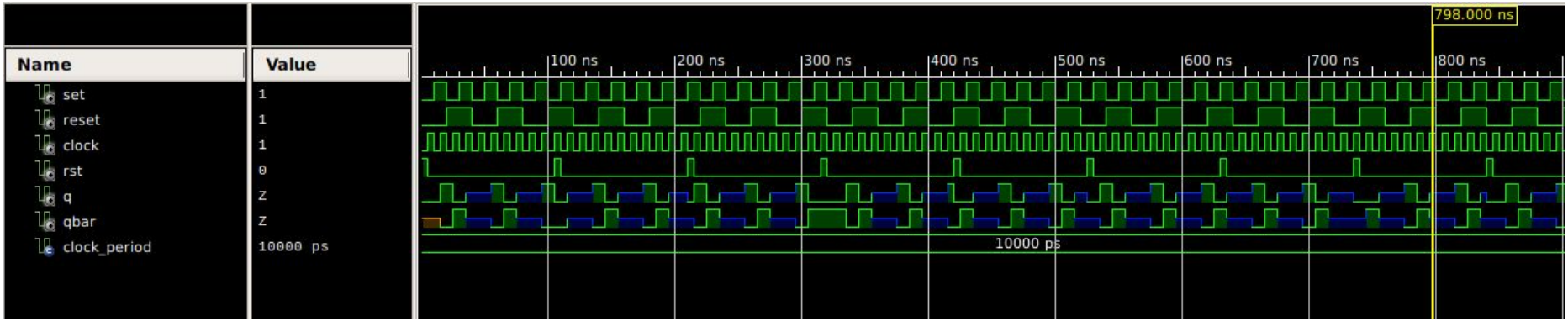
RS Flip-Flop

Test Bench

```
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.ALL;
4  use ieee.std_logic_arith.all;
5  use ieee.std_logic_unsigned.all;
6
7  ENTITY TEST02_1_tb IS
8  END TEST02_1_tb;
9
10 ARCHITECTURE behavior OF TEST02_1_tb IS
11     COMPONENT TEST02_1
12     PORT(
13          set : IN  std_logic;
14          reset : IN  std_logic;
15          clock : IN  std_logic;
16          rst : in std_logic;
17          Q : OUT  std_logic;
18          Qbar : OUT  std_logic
19         );
20     END COMPONENT;
21
22    signal set : std_logic := '0';
23    signal reset : std_logic := '0';
24    signal clock : std_logic := '0';
25    signal rst : std_logic := '0';
26    signal Q : std_logic;
27    signal Qbar : std_logic;
28
29    constant clock_period : time := 10 ns
```

```
31 BEGIN
32    uut: TEST02_1 PORT MAP (
33           set => set,
34           reset => reset,
35           clock => clock,
36           rst => rst,
37           Q => Q,
38           Qbar => Qbar
39         );
40
41    clock_process :process
42    begin
43       clock <= '0';
44       wait for clock_period/2;
45       clock <= '1';
46       wait for clock_period/2;
47    end process;
48
49    set0 : process
50    begin
51       set <= '0';
52       wait for clock_period;
53       set <= '1';
54       wait for clock_period;
55    end process;
56
```

```
57    reset0 : process
58    begin
59       reset <= '0';
60       wait for 2*clock_period;
61       reset <= '1';
62       wait for 2*clock_period;
63    end process;
64
65    -- Stimulus process
66    stim_proc: process
67    begin
68       rst<='1';
69       wait for 5 ns;
70       rst<='0';
71       wait for 100 ns;
72
73    end process;
74
75 END;
```

RS Flip-Flop

Simulation Result