VHDL 및 실습

State Machine

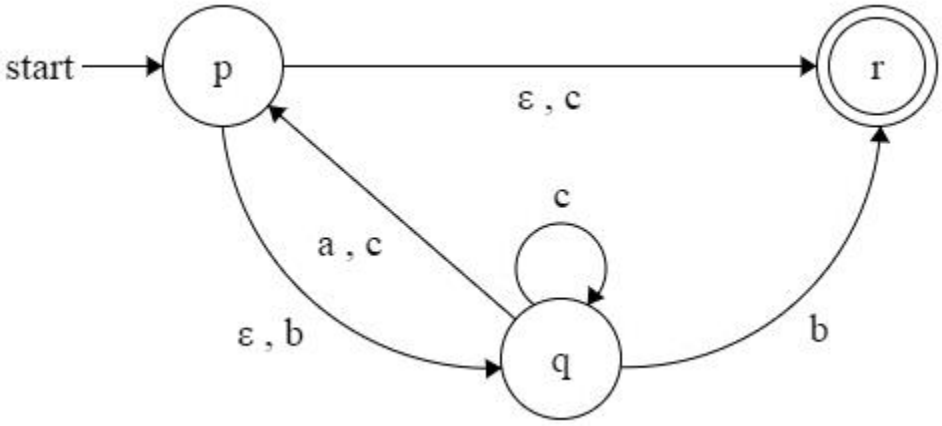
과목	VHDL 및 실습
학과	전자공학과
학번	2011144024
이름	유대성 (오전)
제출일	
담당교수	최종성 교수님

1. 주제 및 배경이론

VHDL 의 계층구조를 이해하고 디지털 시계를 설계해보자.

상태 기계(State Machine)

FSM(Finite State Machine) : 유한 상태 기계, FA(Finite Automaton, 유한 오토마톤)이라고도 불린다. 컴퓨터 사이언스 중 하나인 오토마타로 표현되는 하나의 설계도이다. 컴퓨터 프로그램과 전자 논리 회로를 설계하는 데에 주로 사용되며 간단히 상태 기계라고 부르기도 한다. FSM 은 유한한 개수의 상태를 가질 수 있는 오토마타, 즉 추상 기계라고 할 수 있으며 이러한 기계는 한 번에 오로지 하나의 상태만을 가지게 되며, 현재 상태(Current State)란 임의의 주어진 시간의 상태를 칭한다. 이러한 기계는 어떠한 사건(Event)에 의해 한 상태에서 다른 상태로 변화할 수 있으며 이를 전이(Transition)이라 한다. 특정한 FSM 은 현재 상태로부터 가능한 전이 상태와, 이러한 전이를 유발하는 조건들의 집합으로서 정의한다.



위 그림에서 보이듯 각각의 상태는 원을 통해 표시되며 현재 상태에서 다른 상태로의 전이는 화살표를 통해 나타내어진다. 이를 통해 하나의 상태도가 완성되며 오토마타 이론에 의해 그려진 FSM 이라고 할 수 있다.

Gray Code

이론상의 값과 실제의 값이 일치하는 경우는 극히 드물다. 우리가 사용하는 보드에서도 비슷한 현상이 일어난다. 두 파형 간의 관계에서 글리치가 발생하는 이유처럼 두 파형의 변화는 정확히 일치할 수 없기 때문에 원치 않는 결과를 발생시키게 된다. BCD 코드를 보자. 1 에서 2 로 변할 때 0001 은 0010 으로 두 비트가 동시에 바뀌었다. 하지만 두 과정은 결코 동시에 일어날 수 없기에 1 번 비트가 먼저 변경될 경우 짧은 시간 동안 0011 이라는 파형이 발생될 것이고 0 번 비트가 먼저 변경될 경우엔 0000 이 중간에 발생할 것이다. 이 같은 원치 않는 값을 줄이기 위해 Gray Code 를 사용하는 데 이는 인접한 두 값을 하나의 비트만을 바꾸어 표현하는 것이다.

십진법	BCD 코드	그레이 코드	십진법	BCD 코드	그레이 코드
0	0000	0000	8	1000	1100
1	0001	0001	9	1001	1101
2	0010	0011	10	1010	1111
3	0011	0010	11	1011	1110
4	0100	0110	12	1100	1010
5	0101	0111	13	1101	1011
6	0110	0101	14	1110	1001
7	0111	0100	15	1111	1000

2 진수에서 그레이 코드로 변환하는 방법

① 2 진수의 최상위 비트를 그대로 내려준다.

② 두번째 비트부터는 자신과 그 앞자리에 놓인 비트를 XOR 시킨다

소스코드 및 코드 설명(3, 4 단계 함께 기술)

소스코드를 설명과 함께 기술한다.

2. 시뮬레이션 결과 및 설명(3, 4 단계 함께 기술)

코드 시뮬레이션과 그 결과를 기술한다.

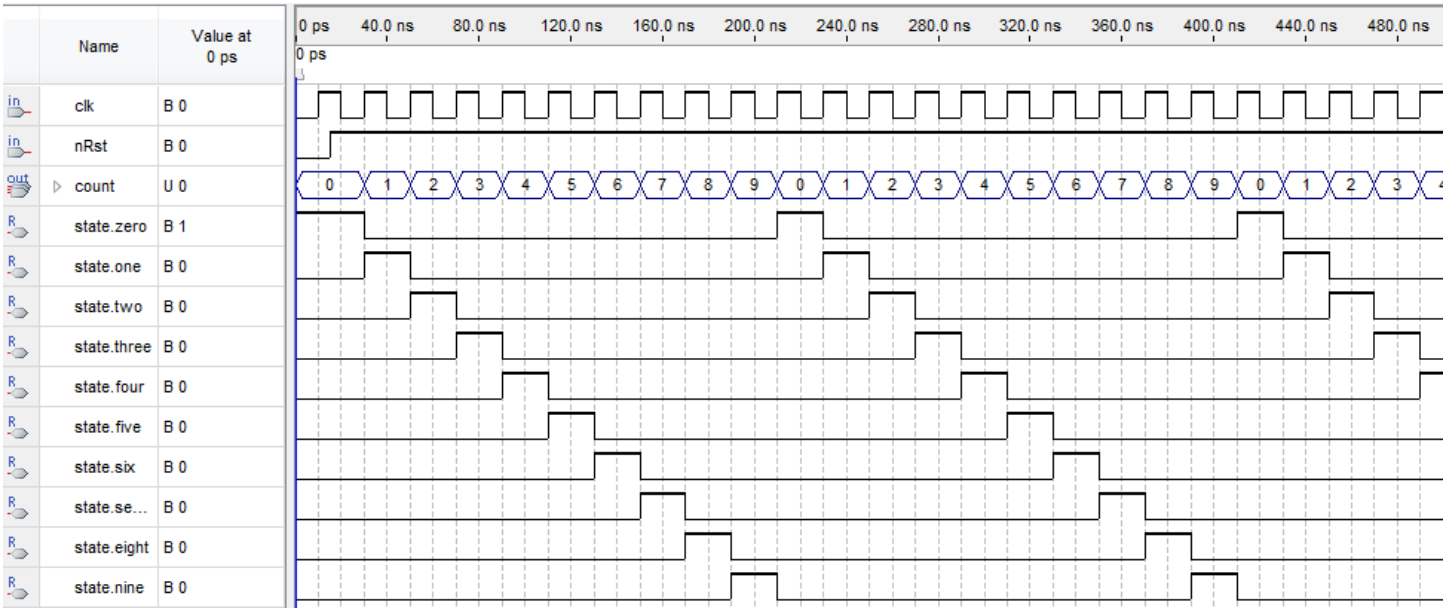
BCD Counter

Counter 를 통해 설계하는 10 진 카운터가 아닌 상태머신에 의해 동작하는 10 진 카운터를 설계해보자.

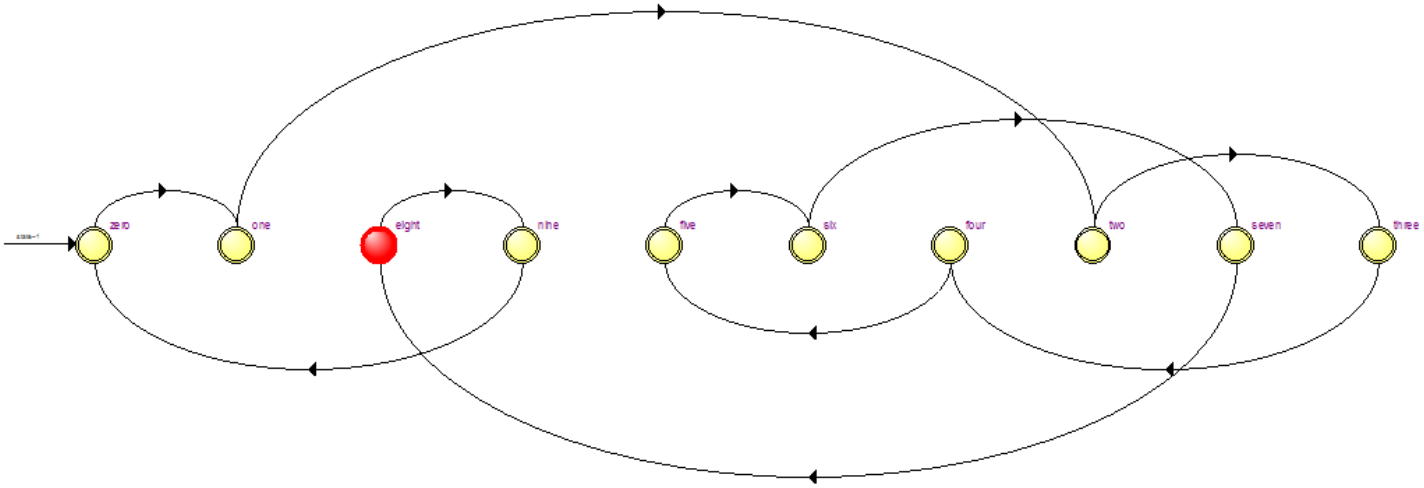
1. VHDL 작성

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

entity bcd_counter is
    port(
        nRst : in std_logic;
        clk : in std_logic;
        count : out std_logic_vector(3 downto 0)
    );
end bcd_counter;

architecture state_machine of bcd_counter is
    type state_type is (zero, one, two, three, four, five, six, seven, eight, nine);
    signal state : state_type;
begin
    state_move : process(nRst, clk)
    begin
        if(nRst = '0') then
            state <= zero;
        elsif rising_edge(clk) then
            case state is
                when zero => state <= one;
                when one => state <= two;
                when two => state <= three;
                when three => state <= four;
                when four => state <= five;
                when five => state <= six;
                when six => state <= seven;
                when seven => state <= eight;
                when eight => state <= nine;
                when nine => state <= zero;
                when others => state <= zero;
            end case;
        end if;
    end process;

    count <= "0000" when state = zero else
             "0001" when state = one else
             "0010" when state = two else
             "0011" when state = three else
             "0100" when state = four else
             "0101" when state = five else
             "0110" when state = six else
             "0111" when state = seven else
             "1000" when state = eight else
             "1001" when state = nine else
             "0000";
end state_machine;
```

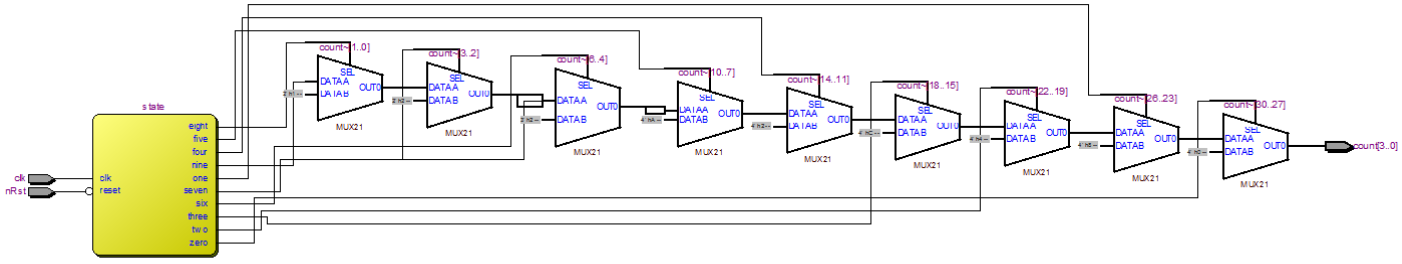
2. Function simulation



3. State diagram



4. RTL Viewer



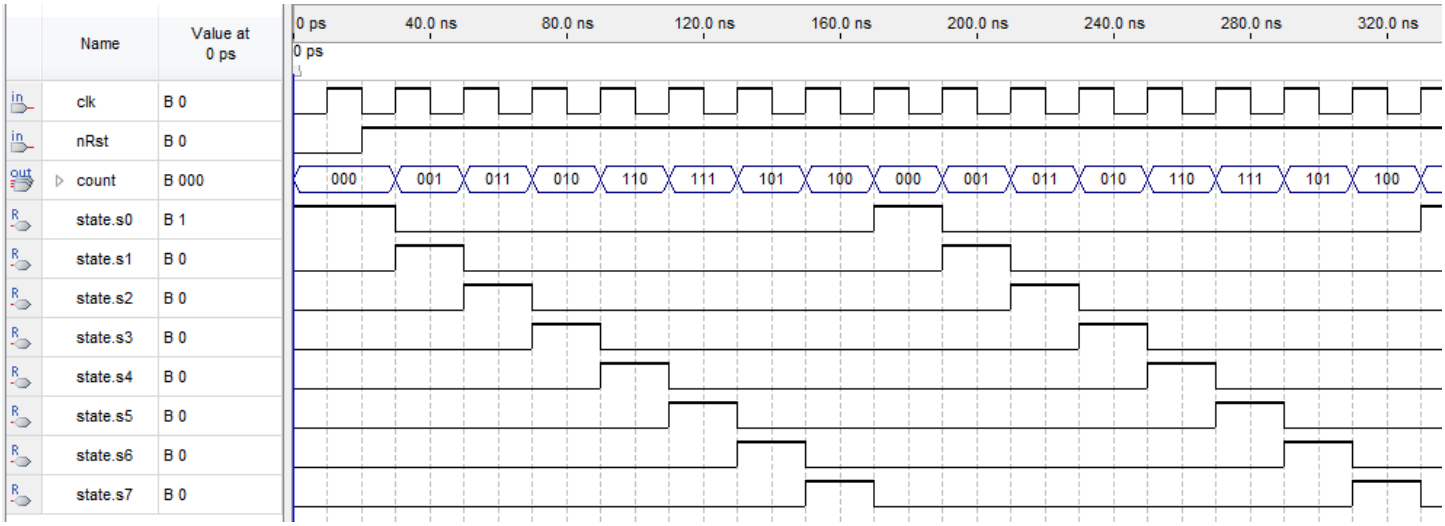
Gray Code Counter

상태 머신에 의해 동작하는 Gray code counter 을 설계해보자.

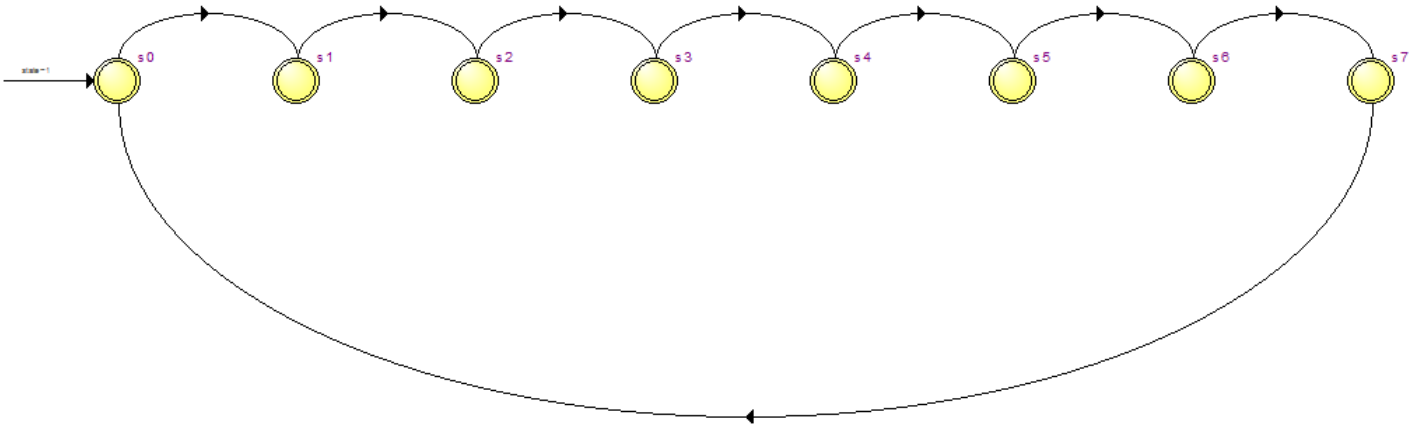
1. VHDL 작성

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

entity gray_counter is
    port(
        nRst : in std_logic;
        clk : in std_logic;
        count : out std_logic_vector(2 downto 0)
    );
end gray_counter;

architecture state_machine of gray_counter is
    type state_type is (s0, s1, s2, s3, s4, s5, s6, s7);
    signal state : state_type;
begin
    state_move : process(nRst, clk)
    begin
        if(nRst = '0') then
            state <= s0;
        elsif rising_edge(clk) then
            case state is
                when s0 => state <= s1;
                when s1 => state <= s2;
                when s2 => state <= s3;
                when s3 => state <= s4;
                when s4 => state <= s5;
                when s5 => state <= s6;
                when s6 => state <= s7;
                when s7 => state <= s0;
                when others => state <= s0;
            end case;
        end if;
    end process;

    count <= "000" when state = s0 else
             "001" when state = s1 else
             "011" when state = s2 else
             "010" when state = s3 else
             "110" when state = s4 else
             "111" when state = s5 else
             "101" when state = s6 else
             "100" when state = s7 else
             "000";
end state_machine;
```

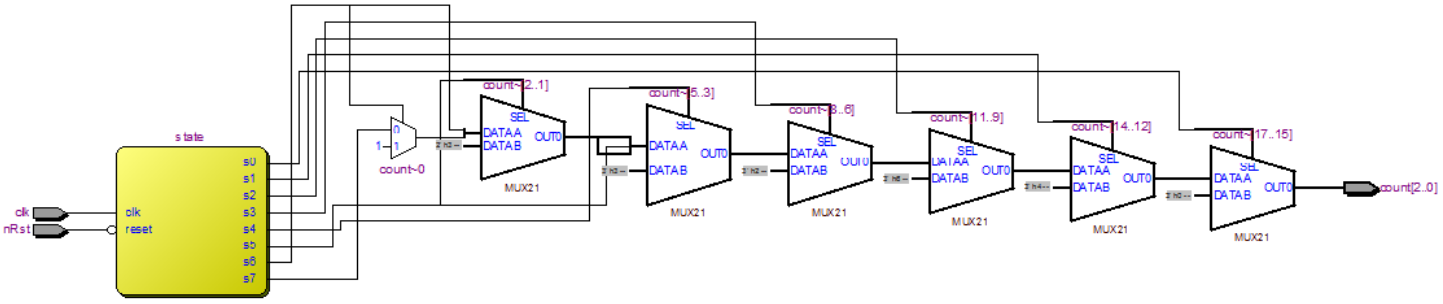
2. Function simulation



3. State diagram



4. RTL Viewer



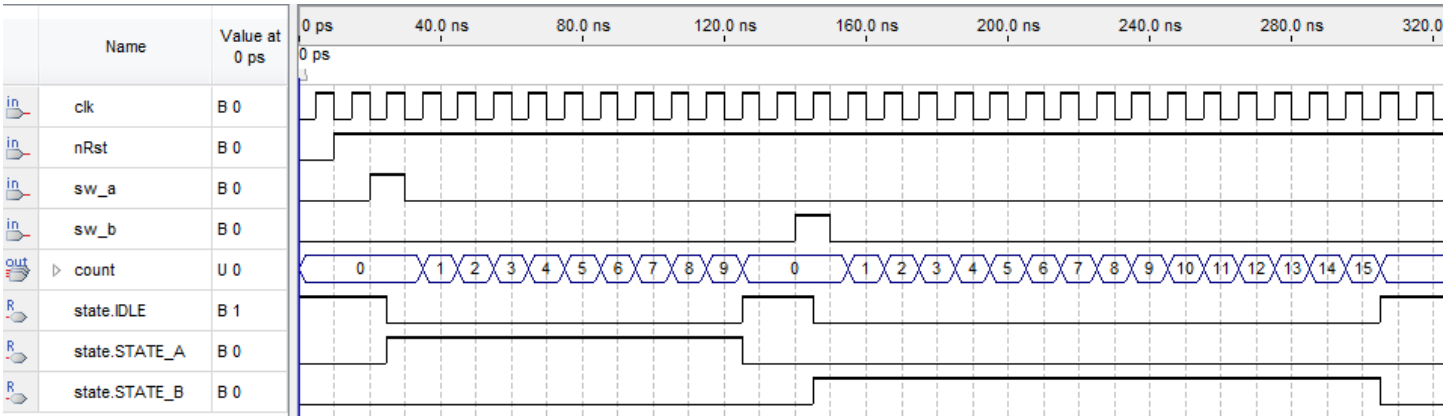
Dual Counter

10 진 기능과 16 진 카운팅 기능이 있는 듀얼 카운터를 상태 머신을 사용하여 구현해보자.

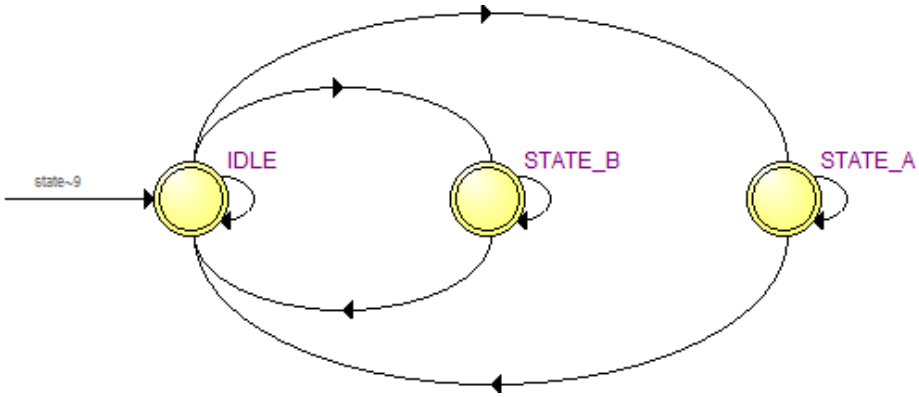
1. VHDL 작성

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

entity dual_counter is
    port(
        nRst : in std_logic;
        clk : in std_logic;
        sw_a : in std_logic;
        sw_b : in std_logic;
        count : out std_logic_vector(3 downto 0)
    );
end dual_counter;

architecture state_machine of dual_counter is
    type state_type is (IDLE, STATE_A, STATE_B);
    signal state : state_type;
    signal cnt : std_logic_vector(3 downto 0);
begin
    process(nRst, clk)
    begin
        if(nRst = '0') then
            state <= IDLE;
            cnt <= (others => '0');
        elsif rising_edge(clk) then
            case state is
                when IDLE =>
                    if(sw_a = '1') then     state <= STATE_A;
                    elsif(sw_b = '1') then state <= STATE_B;
                    else state <= IDLE;
                    end if;
                    cnt <= (others => '0');
                when STATE_A =>
                    if(cnt = 9) then
                        cnt <= (others =>'0');
                        state <= IDLE;
                    else
                        cnt<= cnt+1;
                    end if;
                when STATE_B =>
                    if(cnt = 15) then
                        cnt <= (others =>'0');
                        state <= IDLE;
                    else
                        cnt<= cnt+1;
                    end if;
                when others =>
                    state <= IDLE;
            end case;
        end if;
    end process;
    count <= cnt;
end state_machine;

                                        Colored by Color Scripter cs
```

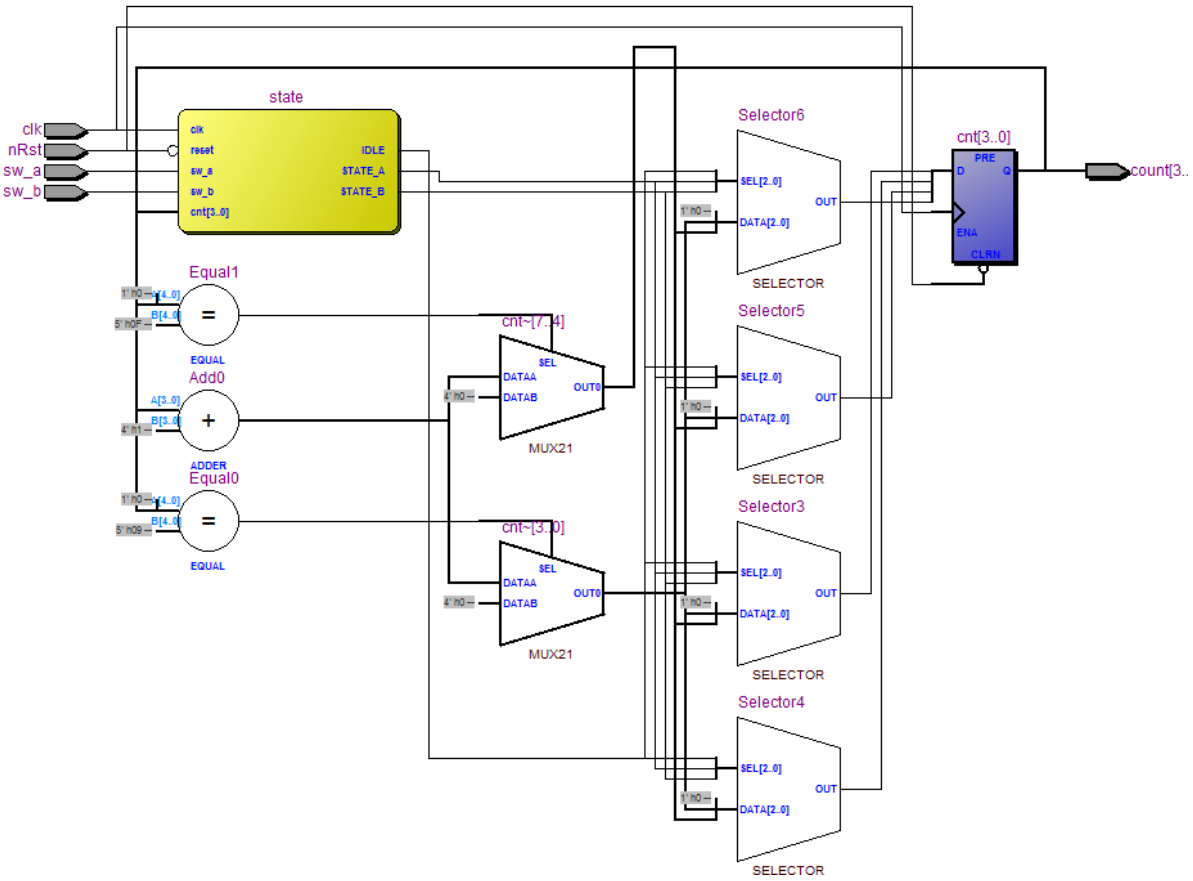
2. Function simulation



3. State diagram



4. RTL Viewer



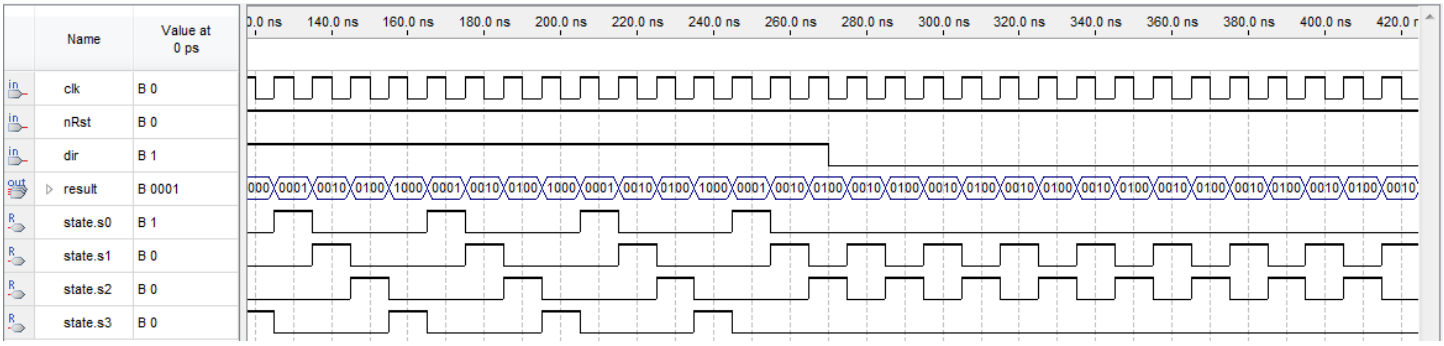
Stepping Motor

Direction 을 지정하여 시계, 반시계 방향으로 회전하는 모터를 만들어 보자.

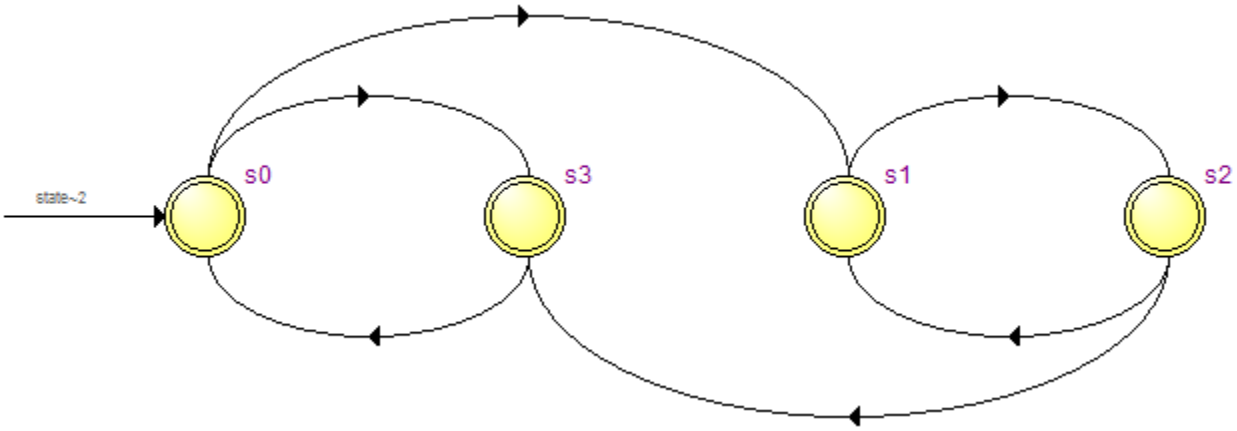
1. VHDL 작성

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

entity stepping_motor is
    port(
        nRst : in std_logic;
        clk : in std_logic;
        dir : in std_logic;
        result : out std_logic_vector(3 downto 0)
    );
end stepping_motor;

architecture state_machine of stepping_motor is
    type state_type is (s0, s1, s2, s3);
    signal state : state_type;
begin
    process(nRst, clk)
    begin
        if(nRst = '0') then
            state <= s0;
        elsif rising_edge(clk) then
            case state is
                when s0 =>
                    if(dir = '1') then state <= s1;
                    else state <= s3;
                    end if;
                when s1 =>
                    if(dir = '1') then state <= s2;
                    else state <= s2;
                    end if;
                when s2 =>
                    if(dir = '1') then state <= s3;
                    else state <= s1;
                    end if;
                when s3 =>
                    if(dir = '1') then state <= s0;
                    else state <= s0;
                    end if;
                when others =>
                    state <= s0;
            end case;
        end if;
    end process;

    result <= "0001" when state = s0 else
              "0010" when state = s1 else
              "0100" when state = s2 else
              "1000" when state = s3 else
    "0000";

end state_machine;                                  cs
```

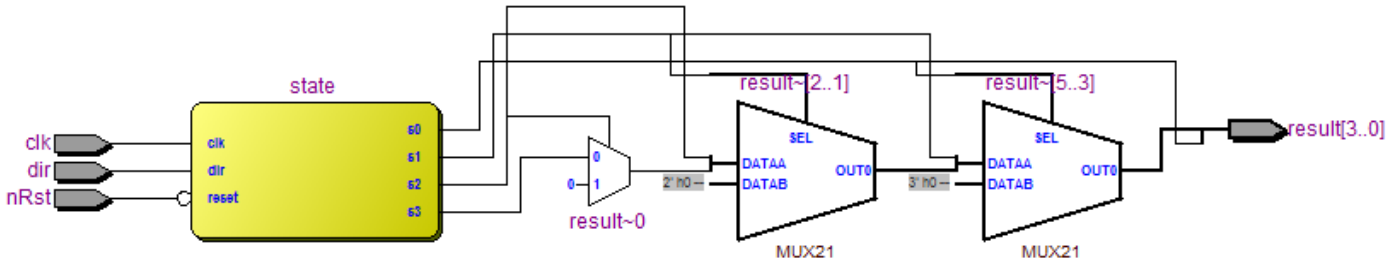
2. Function simulation



3. State diagram



4. RTL Viewer



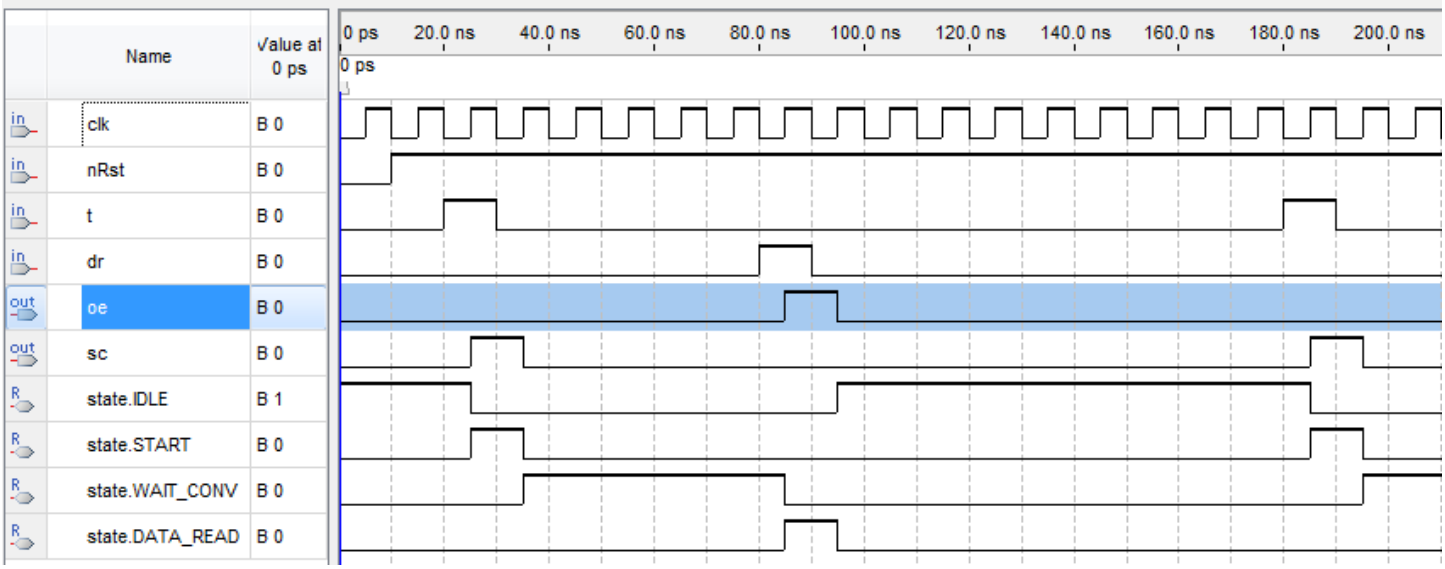
ADC Controller

앞에서 구상한 ADC Controller 를 직접 설계하여 보자.

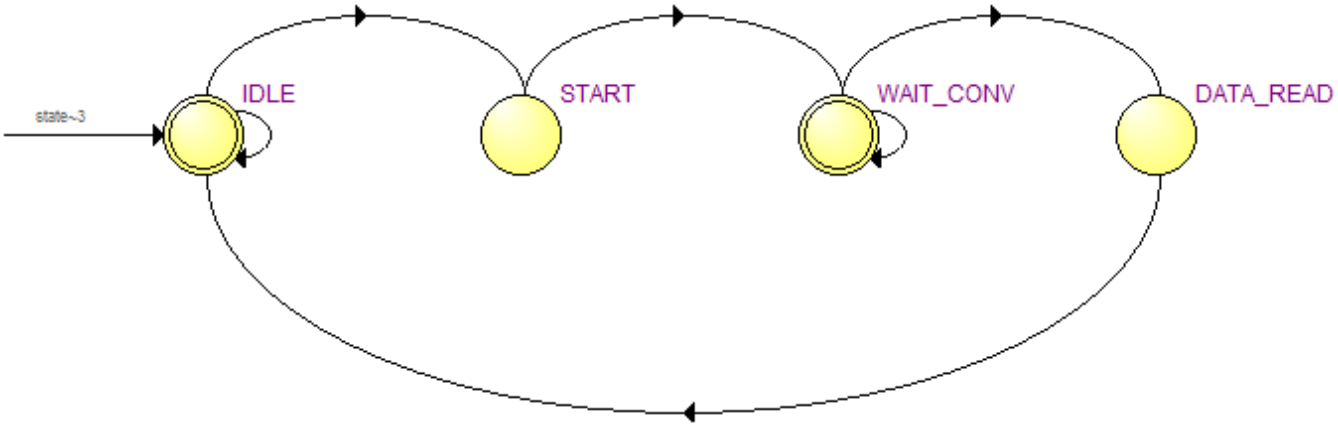
1. VHDL 작성

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

entity adc_control is
    port(
        nRst : in std_logic;
        clk : in std_logic;
        t : in std_logic;
        dr : in std_logic;
        sc : out std_logic;
        oe : out std_logic
    );
end adc_control;

architecture BEH of adc_control is
    type state_type is (IDLE, START, WAIT_CONV, DATA_READ);
    signal state : state_type;
begin
    process(nRst, clk)
    begin
        if(nRst = '0') then
            state <= IDLE;
            sc <= '0';
            oe <= '0';
        elsif rising_edge(clk) then
            case state is
                when IDLE =>
                    if(t = '1') then
                        state <= START;
                        sc <= '1';
                        oe <= '0';
                    else
                        state <= IDLE;
                        sc <= '0';
                        oe <= '0';
                    end if;
                when START =>
                    state <= WAIT_CONV;
                    sc <= '0';
                    oe <= '0';
                when WAIT_CONV =>
                    if(dr = '1') then
                        state <= DATA_READ;
                        sc <= '0';
                        oe <= '1';
                    else
                        state <= WAIT_CONV;
                        sc <= '0';
                        oe <= '0';
                    end if;
                when DATA_READ =>
                    state <= IDLE;
                    sc <= '0';
                    oe <= '0';
                when others =>
                    state <= IDLE;
            end case;
        end if;
    end process;

end BEH;
```

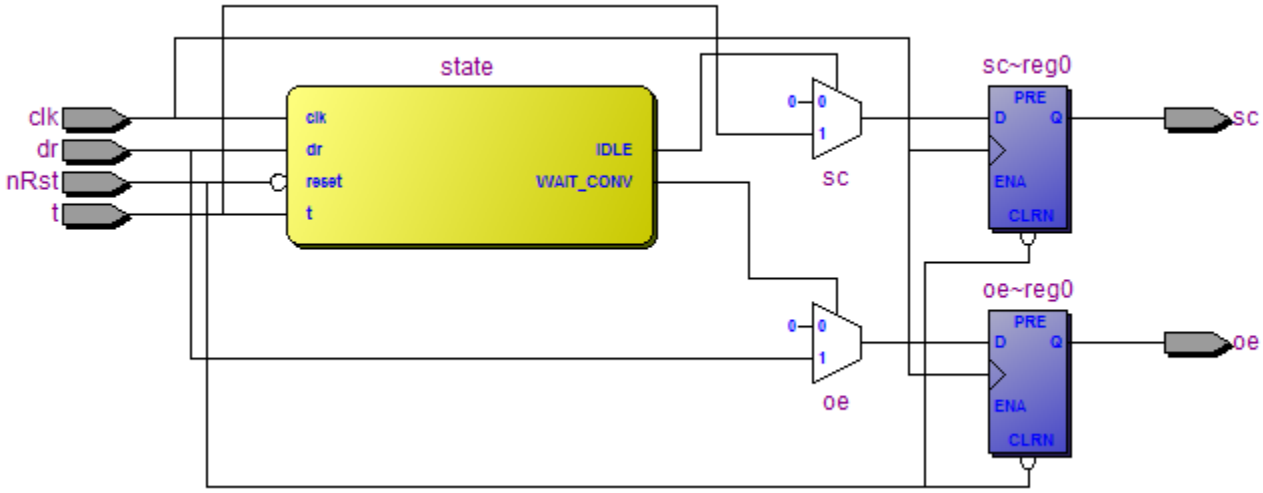
2. Function simulation



3. State diagram



4. RTL Viewer



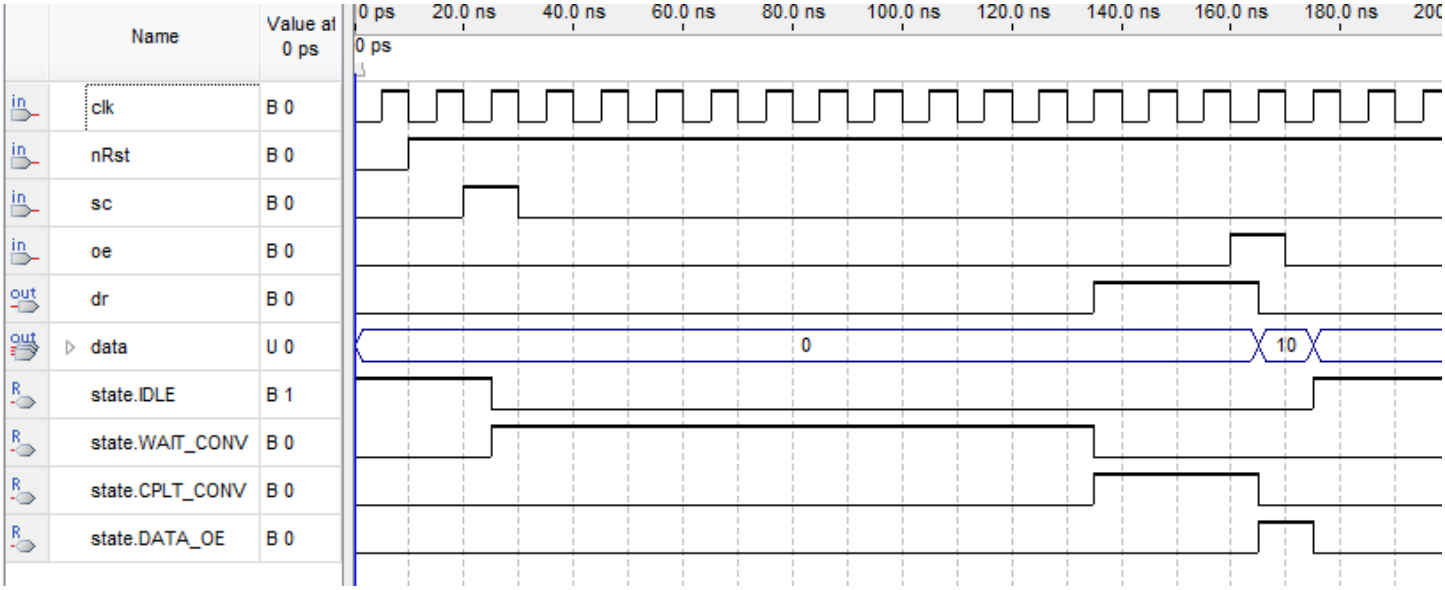
ADC Model

앞에서 설계한 ADC Controller 시뮬레이션을 위한 ADC Model 을 설계해 보자.

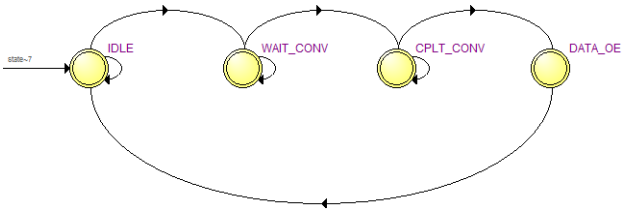
1. VHDL 작성

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

entity adc is
    port(
        nRst : in std_logic;
        clk : in std_logic;
        dr : out std_logic;
        sc : in std_logic;
        oe : in std_logic;
        data : out std_logic_vector(3 downto 0)
    );
end adc;

architecture BEH of adc is
    type state_type is (IDLE, WAIT_CONV, CPLT_CONV, DATA_OE);
    signal state : state_type;
    signal cnt : std_logic_vector(3 downto 0);
begin
    process(nRst, clk)
    begin
        if(nRst = '0') then
            state <= IDLE;
            dr <= '0';
            cnt <= (others => '0');
        elsif rising_edge(clk) then
            case state is
                when IDLE =>
                    if(sc = '1') then
                        state <= WAIT_CONV;
                        dr <= '0';
                    else
                        state <= IDLE;
                        dr <= '0';
                    end if;
                when WAIT_CONV =>
                    if(cnt = 10) then
                        state <= CPLT_CONV;
                        dr <= '1';
                    else
                        cnt <= cnt+1;
                    end if;
                when CPLT_CONV =>
                    if(oe = '1') then
                        state <= DATA_OE;
                        dr <= '0';
                    else
                        state <= CPLT_CONV;
                        dr <= '1';
                    end if;
                when DATA_OE =>
                    state <= IDLE;
                    dr <= '0';
                    cnt <= (others => '0');
                when others =>
                    state <= IDLE;
                    dr <= '0';
                    cnt <= (others => '0');
            end case;
        end if;
    end process;

    data <= cnt when state = DATA_OE else "0000";

end BEH;

                                                    Colored by Color Scripter cs
```

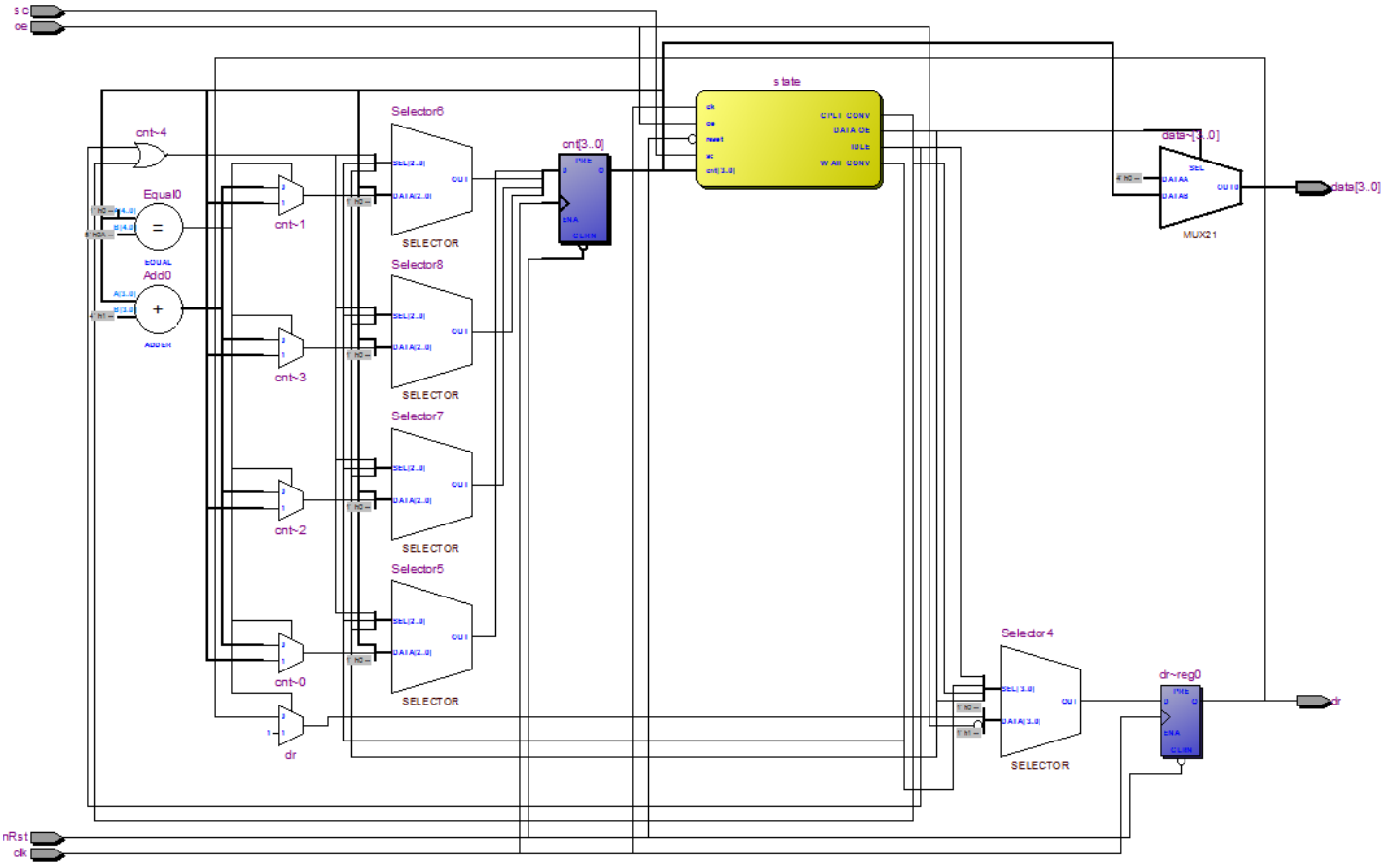
2. Function simulation



3. State diagram



4. RTL Viewer



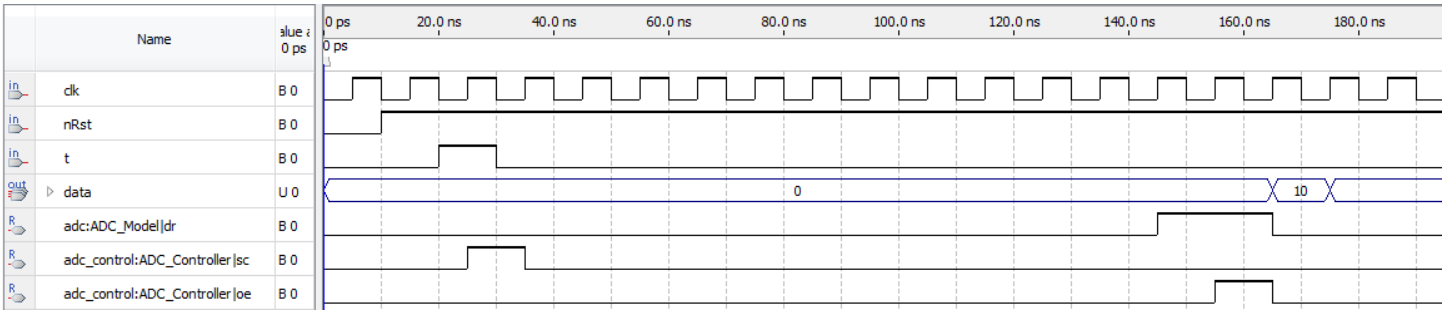
ADC Converter (Top Entity)

이제 필요한 아키텍처를 모두 설계했으니 두 가지 컴포넌트를 사용해 시뮬레이션을 진행해보자.

1. VHDL 작성

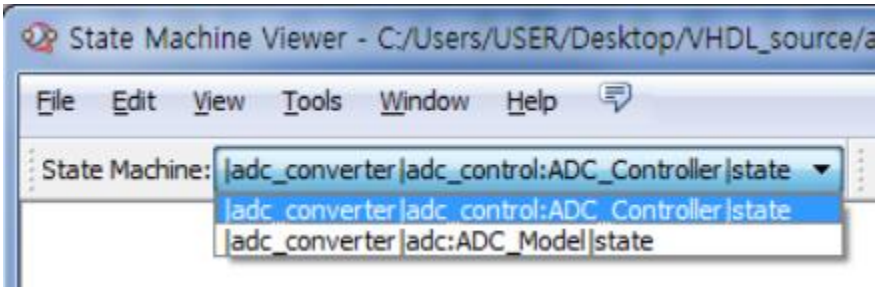
```
library ieee;
    use ieee.std_logic_1164.all;
    use ieee.std_logic_arith.all;
    use ieee.std_logic_unsigned.all;

entity adc_converter is
    port(
        nRst : in std_logic;
        clk : in std_logic;
        t : in std_logic;
        data : out std_logic_vector (3 downto 0)
    );
end adc_converter;

architecture BEH of adc_converter is

    component adc_control
        port(
            nRst : in std_logic;
            clk : in std_logic;
            t : in std_logic;
            dr : in std_logic;
            sc : out std_logic;
            oe : out std_logic
        );
    end component;

    component adc
        port (
            nRst : in std_logic;
            clk : in std_logic;
            dr : out std_logic;
            sc : in std_logic;
            oe : in std_logic;
            data : out std_logic_vector(3 downto 0)
        );
    end component;

    signal dr : std_logic;
    signal sc : std_logic;
    signal oe : std_logic;

    begin
        ADC_Controller : adc_control
        port map(
            nRst => nRst,
            clk => clk,
            t => t,
            dr => dr,
            sc => sc,
            oe => oe
        );

        ADC_Model : adc
        port map(
            nRst => nRst,
            clk => clk,
            dr => dr,
            sc => sc,
            oe => oe,
            data => data
        );

end BEH;
```

2. Function simulation

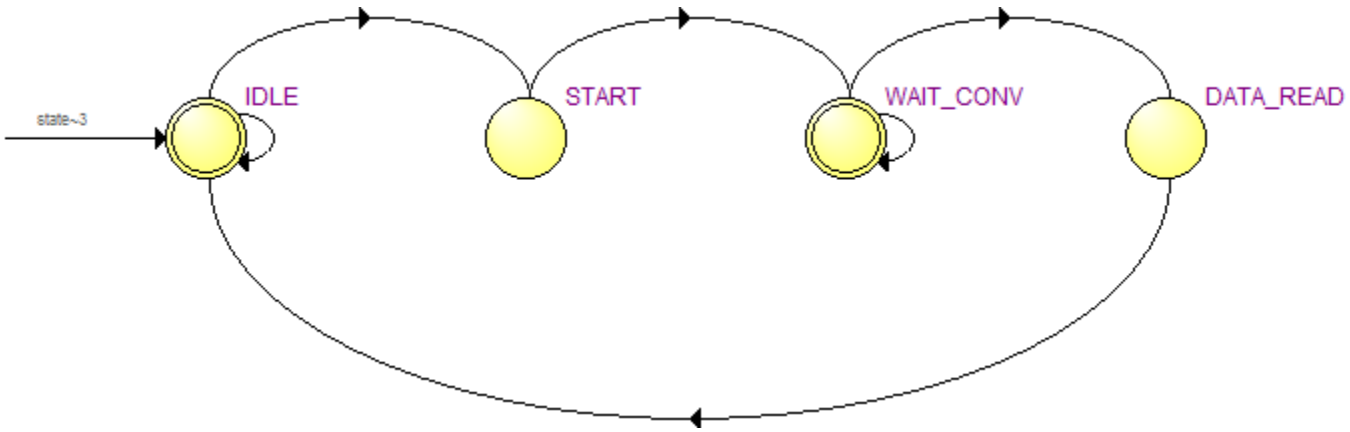


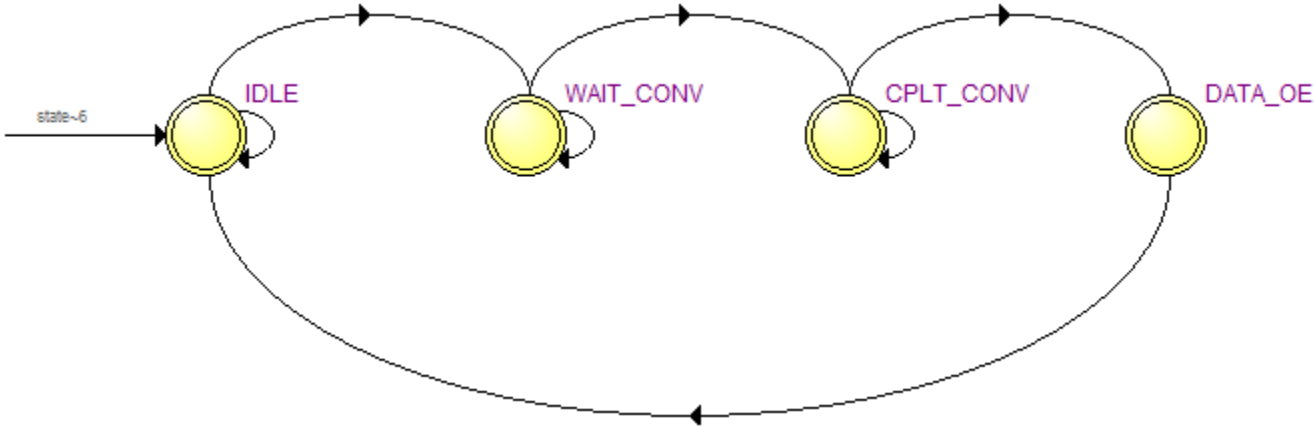
3. State diagram

- 상태도가 다음과 같이 두 가지로 나타났다. 상태도를 합치면 경우에 수가 많아져 별도로 생성되는 듯 하다.

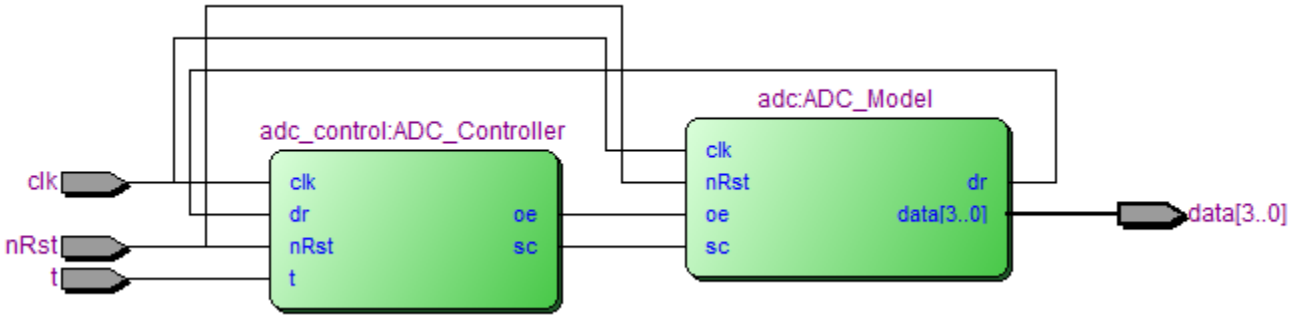


- 각각의 상태도는 개별 컴포넌트를 설계하였을 때와 동일했다. 이로써 하위 구조의 상태도는 별도로 합쳐지는 과정 없이 표시된다는 것을 알 수 있었다.





4. RTL Viewer



3. 실습보드 적용 결과

소스를 보드에 다운로드하여 예상한 결과에 맞게 동작하는 지 테스트하시오

ADC Converter (Top Entity)

보드 테스트를 위해 1 초 생성기를 추가하여 테스트 하였습니다.

1) 수정된 VHDL

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

entity adc_converter is
    port(
        nRst : in std_logic;
        clk : in std_logic;
        t : in std_logic;
        data : out std_logic_vector (3 downto 0)
    );
end adc_converter;

architecture BEH of adc_converter is

    component sec_gen
        port(
            nRst        : in std_logic;
            clk         : in std_logic;
            sec_sig     : out std_logic
        );
    end component;

    component adc_control
        port(
            nRst : in std_logic;
            clk : in std_logic;
            t : in std_logic;
            dr : in std_logic;
            sc : out std_logic;
            oe : out std_logic
        );
    end component;

    component adc
        port (
            nRst : in std_logic;
            clk : in std_logic;
```

```
40          dr : out std_logic;
41          sc : in std_logic;
42          oe : in std_logic;
43          data : out std_logic_vector(3 downto 0)
44      );
45  end component;
46
47  signal sec_clk : std_logic;
48  signal dr : std_logic;
49  signal sc : std_logic;
50  signal oe : std_logic;
51
52  begin
53      sec_generator : sec_gen
54      port map(
55          nRst => nRst,
56          clk => clk,
57          sec_sig => sec_clk
58      );
59
60      ADC_Controller : adc_control
61      port map(
62          nRst => nRst,
63          clk => sec_clk,
64          t => t,
65          dr => dr,
66          sc => sc,
67          oe => oe
68      );
69
70      ADC_Model : adc
71      port map(
72          nRst => nRst,
73          clk => sec_clk,
74          dr => dr,
75          sc => sc,
76          oe => oe,
77          data => data
78      );
79
80  end BEH;
```

Colored by Color Scripter

2) Pin Planner

Node Name	Direction	Location
clk	Input	PIN_N2
data[3]	Output	PIN_AC22
data[2]	Output	PIN_AB21
data[1]	Output	PIN_AF23
data[0]	Output	PIN_AE23
nRst	Input	PIN_G26
t	Input	PIN_N23

3) 보드 테스트

4. 실습소감

실습을 통해 경험한 것을 자유롭게 서술하시오.

금주 실습에서 이전 프로그래밍 수업시간에 배운 오토마타와 비슷한 개념의 지식을 배워 어느 정도 이해하는 데 도움이 되었다. 매주 수업마다 기존에 내가 알던 프로그래밍과 비교하며 수업을 듣게 되는데 이런 과정 자체가 VHDL 이라는 과목을 이해하는 데 많은 도움이 되고 있다. 오늘 같은 경우에 생각보다 실습하는 데 오래걸리지 않을 뿐더러 어렵지도 않았다. 2~3 주차까지에 비해 훨씬 어려운 것을 배우고 있지만 마음은 한결 편하다. 실력이 늘어뿌듯하지만 오늘 수업 중에 교수님이 말씀하신 대로 실전에서 사용할 수 있는 수준인가라는 질문에는 답하기 어려울 듯 하다. 아직 갈 길이 멀다는 생각은 들지만 자신감이 생겨서 잘 해결해 나갈 수 있을 것 같다.

5. 문의 사항

실습하다 겪은 어려웠던 점을 기술하시오.

이번 실습중 ADC Model 에 대한 RTL Viewer 의 회로도가 엄청 복잡하게 나왔는데 제 설계에 문제가 있었던 것인지 궁금합니다. 회로의 동작만 보면 ADC Control 과 비슷한 동작을 하지만 구성도의 크기가 너무 차이가 나서 당황했습니다. 동작은 구현하였지만 회로 합성과정에서 불필요한 요소가 들어간 것 같은데 효율적으로 구성한다면 어느 부분을 손봐야 할 지 피드백 부탁드립니다.