More on FSM's (Finite State Machines)

a) Mealy machine

Mealy machine: edge detector of rising edges.

library ieee;
use ieee.std_logic_1164.all;

entity edge_detector is
  port(
    clk, reset : in std_logic;
    level : in std_logic;
    tick : out std_logic
  );
end edge_detector;

architecture mealy_arch of edge_detector is
  type state_type is (s0, s1);
  signal state, state_next : state_type;
begin
  -- state register; process #1
  process (clk, register)
  begin
    if (reset = '1')
      state <= s0;
    else if (clk'event and clk = '1') then
      state <= state_next;
    end if;
  end process;

  -- state next; process #2
  process (clk)
  begin
    if (reset = '1')
      state_next <= s0;
    else if (clk'event and clk = '1') then
      state_next <= state;
    end if;
  end process;
end;
--- next state & output logic ---

```vhdl
process (state, level)
begin
    state'next <= state;
    tick <= '0';
    case state is
    when S0 =>
        if level = '1' then
            state'next <= S1;
            tick <= '1';
        end if;
    when S1 =>
        if level = '0' then
            state'next <= S0;
        end if;
    end case;
end process;
```

end mealy-arch;

--- Moore machine ---

```
S0
\arrow{S1}\quad \text{tick+1}
\arrow{S2}
\arrow{level}

level

S0\quad S1\quad S2
\arrow{level}
\arrow{level}
\arrow{level}
```

```
level

comb. circ. A
\arrow{comb. circ. B}
\arrow{FF}
\arrow{reset}

state

state'next

clk

level

state

tick

---

---

---

---

---
architcture moore_arch of edge_detec is

    type state_type is (S0, S1, S2);
    signal state, state_next : state_type;

begin
    -- state register; process #1
    begin
        if (reset = '1') then
            state <= S0;
        else if (clk = '1' and clk'event) then
            state <= state_next;
        end if;
    end process;

    -- next state and output logic; process #2
    process (state, level)
    begin
        state_next <= state;
        tick <= '0';
        case state is
            when S0 =>
                if (level = '1') then
                    state_next <= S1;
                end if;
            when S1 =>
                tick <= '1';
                if (level = '1') then
                    state_next <= S2;
                else
                    state_next <= S0;
                end if;
            when S2 =>
                if (level = '0') then
                    state_next <= S0;
                end if;
        end case;
    end process;
end moore_arch;