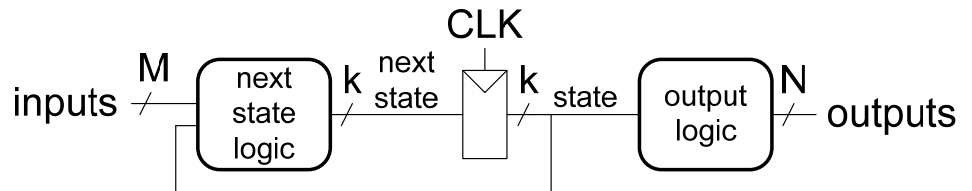
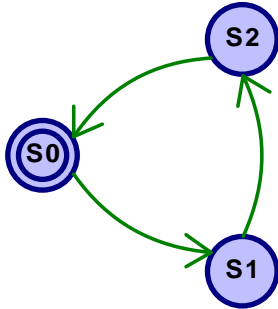


Divide by 3 Finite State Machine



--With concurrent statements

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity divby3FSM is
port (clk, reset: in STD_LOGIC;
      y: out STD_LOGIC);
end divby3FSM;

architecture synth of divby3FSM is
    type statetype is (S0, S1, S2);
    signal state, nextstate: statetype;
begin
    -- state register
    process (clk, reset) begin
        if reset = '1' then state <= S0;
        elsif RISING_EDGE(clk) then
            state <= nextstate;
        end if;
    end process;

    -- next state logic
    nextstate <= S1 when state = S0 else
                S2 when state = S1 else
                S0;

    -- output logic
    y <= '1' when state = S0 else '0';
end synth;
```

```

--with a process

library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity divby3FSM is
port (clk, reset: in STD_LOGIC;
      y: out STD_LOGIC);
end divby3FSM;

architecture synth of divby3FSM is
    type statetype is (S0, S1, S2);
    signal state, nextstate: statetype;
begin
    -- state register
    process (clk, reset) begin
        if reset = '1' then state <= S0;
        elsif RISING_EDGE(clk) then
            state <= nextstate;
        end if;
    end process;

    -- next state logic (combinational)
    process(state) begin
        case state is
            when S0 => nextstate <= S1;
            when S1 => nextstate <= S2;
            when S2 => nextstate <= S0;
            when others => nextstate <= S0;
        end case;
    end process;

    -- output logic (combinational)
    y <= '1' when state = S0 else '0';
end synth;

```