4 Way Traffic Light Controller

ECE 2700: Professor Daniel Llamocca

Sam Narusch, Richard Pinto, Christina Salama, Adrian Sinishtaj

Introduction

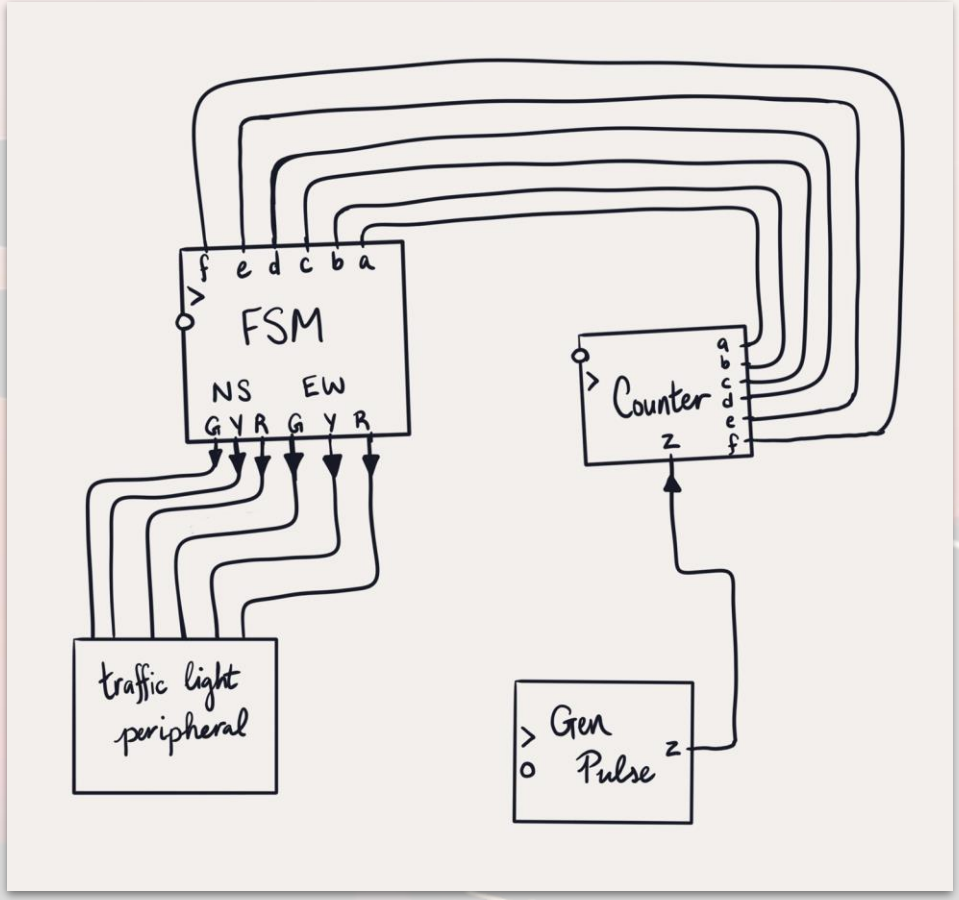
In this project, we designed and implemented a 4 way traffic light controller.

Consists of an FSM (Finite State Machine), a counter, and a gen pulse that display signals on an external board with LED’s.

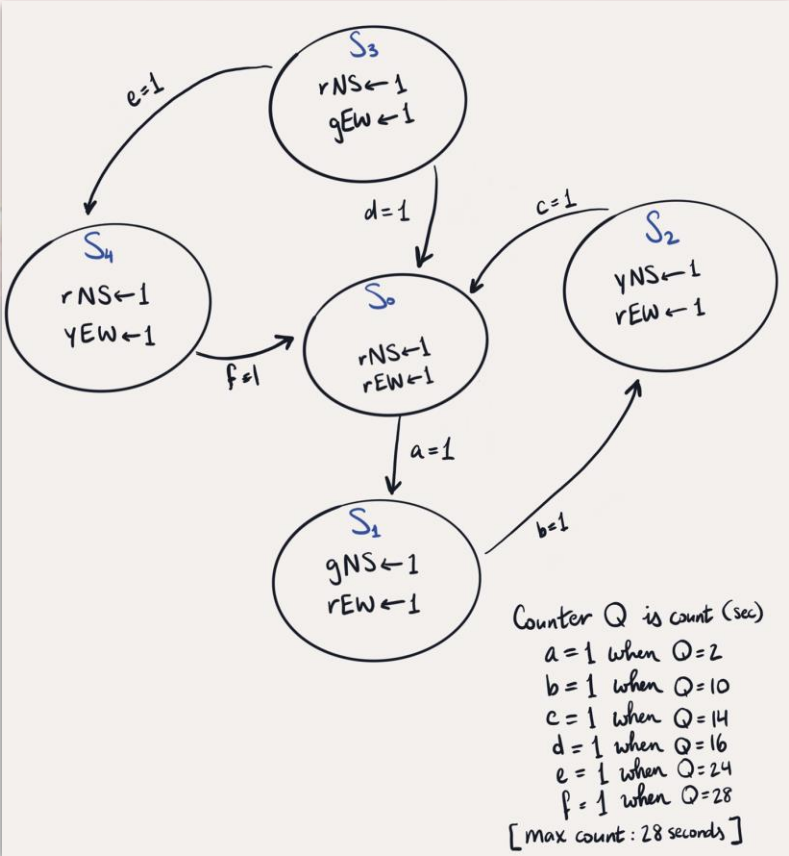
The simulated intersection has 8 seconds with the green light, 4 seconds with yellow light.

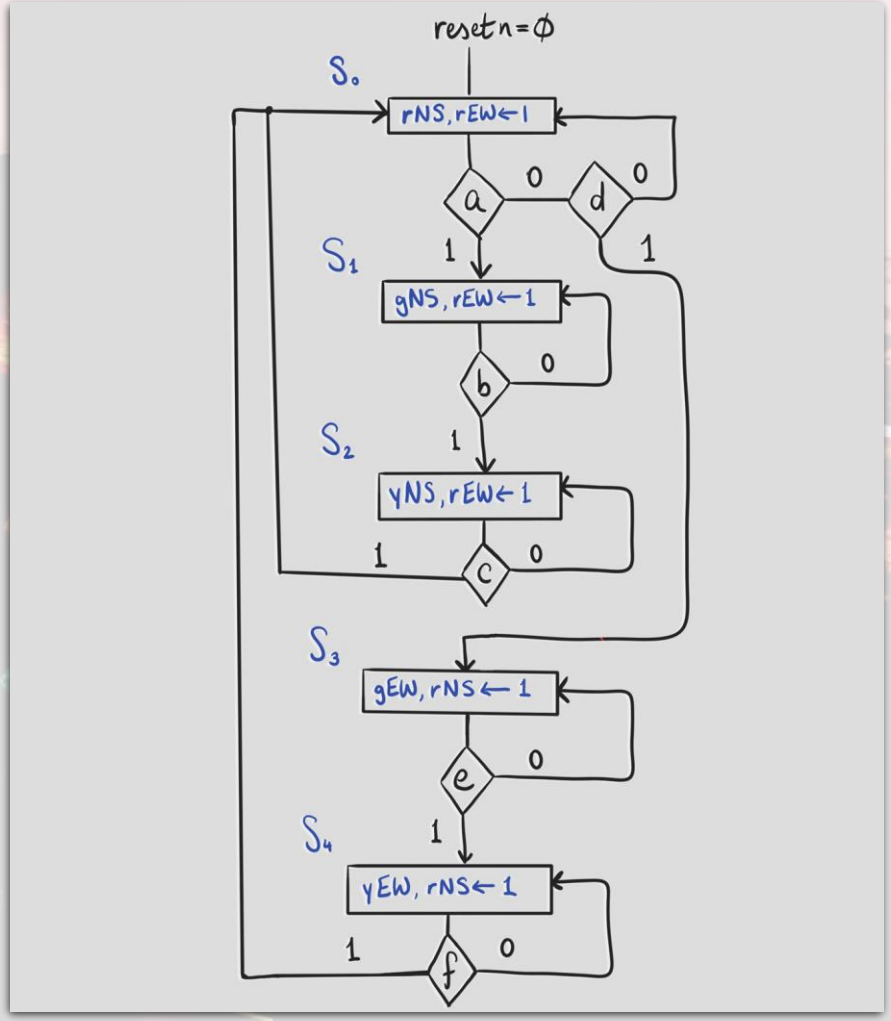
All lights will be red for an additional 2 seconds concurrently for safety purposes.

Block Diagram

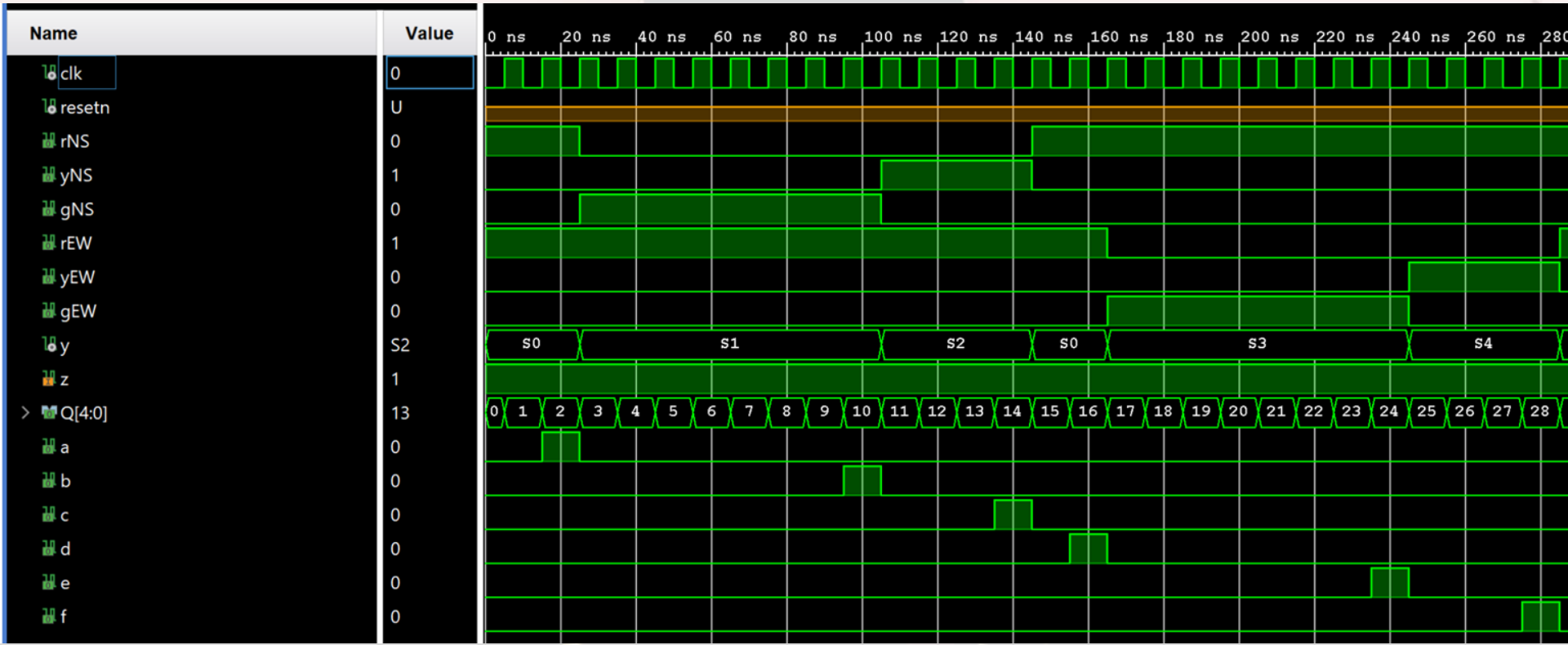


FSM Diagram

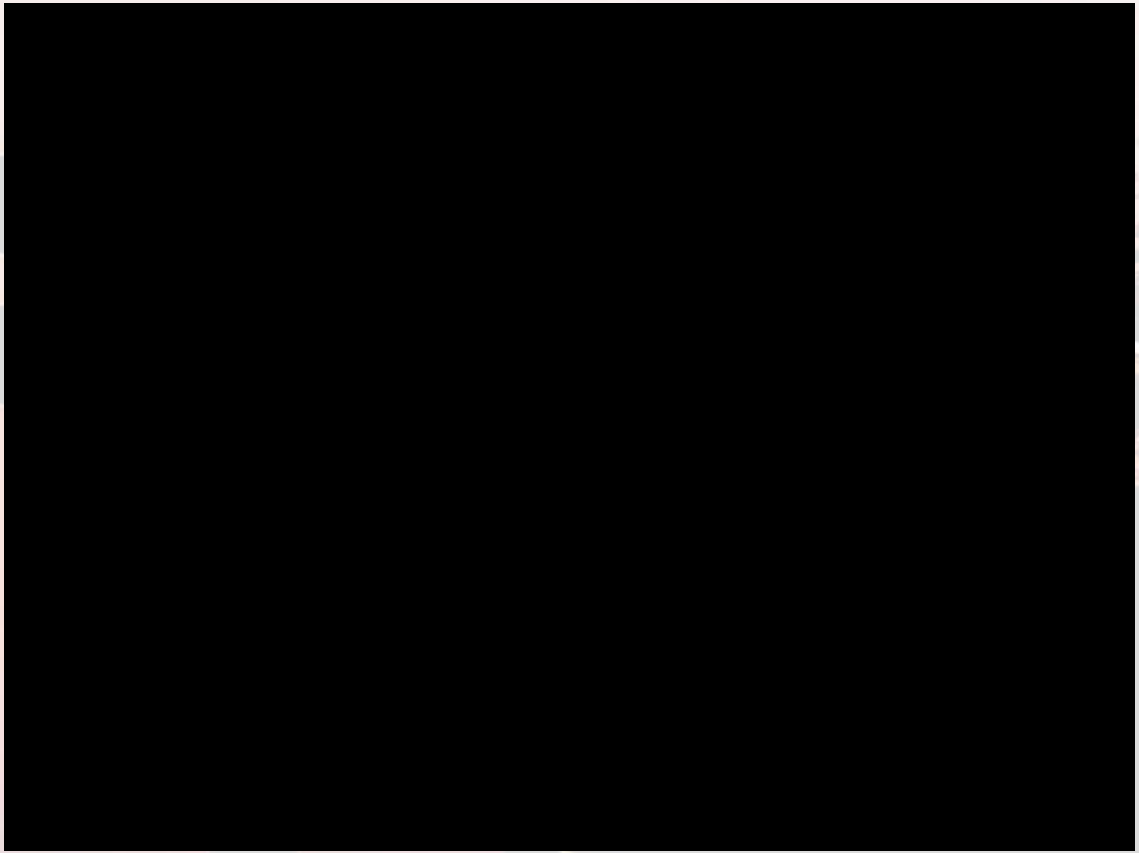

S3
rNS←1
gEW←1
e=1
d=1
c=1
S4
rNS←1
yEW←1
S0
rNS←1
rEW←1
S2
yNS←1
rEW←1
f=1
a=1
S1
gNS←1
rEW←1
b=1
Counter Q is count (sec)
a = 1 when Q = 2
b = 1 when Q = 10
c = 1 when Q = 14
d = 1 when Q = 16
e = 1 when Q = 24
f = 1 when Q = 28
[max count : 28 seconds]


reset n = ϕ
S0
rNS, rEW←1
a
0
d
0
1
1
S1
gNS, rEW←1
b
0
1
S2
yNS, rEW←1
1
c
0
S3
gEW, rNS←1
e
0
1
S4
yEW, rNS←1
1
f
0

Simulation


Name
Value
clk
0
resetn
U
rNS
0
yNS
1
gNS
0
rEW
1
yEW
0
gEW
0
y
S2
z
1
Q[4:0]
13
a
0
b
0
c
0
d
0
e
0
f
0
0 ns
20 ns
40 ns
60 ns
80 ns
100 ns
120 ns
140 ns
160 ns
180 ns
200 ns
220 ns
240 ns
260 ns
S0
S1
S2
S0
S3
S4
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28

Implementation



VHDL Code for Finite State Machine

```
21  entity fsm is
22  port (clock,resetn: in std_logic;
23        a, b, c, d, e, f : in std_logic;
24        rNS, yNS, gNS, rEW, yEW, gEW: out std_logic);
25  --      e,d,resetc: out std_logic);
26        --done: out std_logic);
27  end fsm;
28
29  --struct is architecture name and can call it whatever you want as long as its the same for end 'struct'
30  architecture struct of fsm is
31      type state is (S0,S1,S2,S3,S4);
32      signal y:state;
33
34  begin
35  --VHDL is concurrent, so order of f, x, and y do not matter
36      Transitions: process (resetn,clock, a, b, c, d, e, f)
37      begin
38          if resetn = '0' then
39              y<=S0; --initial state
40         elsif (clock'event and clock='1') then
41      case y is
42          when S0 => if a = '1' then y <= S1;
43          elsif d = '1' then y <= S3; else y <= S0;
44         end if;
45          when S1 => if b = '1' then y <= S2; else y <= S1; end if;
46          when S2 => if c = '1' then y <= S0; else y <= S2; end if;
47          when S3 => if e = '1' then y <= S4; else y <= S3; end if;
48          when S4 => if f = '1' then y <= S0; else y <= S4; end if;
49      end case;
```

```
50  end if;
51  end process;
52      Outputs: process (y)
53  begin
54          --resetC<='0'; d<='0';; --default values
55          rNS <= '0'; yNS <= '0'; gNS <= '0'; rEW <= '0' ;yEW <= '0'; gEW <= '0';
56          case y is
57              when S0 => rNS <= '1'; rEW <= '1'; --if RL = '1' then resetC <= '1';
58              when S1 => gNS <= '1'; rEW <= '1'; --if GL = '1' then resetC <= '1';
59              when S2 => yNS <= '1'; rEW <= '1'; --e <= '1'; d <= '1'; if YL = '1'
60              when S3 => gEW <= '1'; rNS <= '1'; --if GL = '1' then resetC <= '1';
61              when S4 => yEW <= '1'; rNS <= '1'; --e <= '1'; d <= '0'; if YL ='1' t
62  end case;
63  end process;
64
65  end struct;
```

VHDL Code for Counter

```
2  use IEEE.STD_LOGIC_1164.ALL;
3  use ieee.std_logic_arith.all;
4
5  entity counter is
6      port ( clock, resetn, z: in std_logic;
7               Q: out std_logic_vector (3 downto 0);
8               a, b, c, d, e, f: out std_logic);
9  end counter;
10
11 architecture Behavioral of counter is
12     signal Qt: integer range 0 to 28;
13 begin
14
15     process (resetn, clock, z)
16     begin
17         if resetn = '0' then
18             Qt <= 0;
19         elsif (clock'event and clock = '1') and z = '1' then
```

```
21                 if Qt = 28 then
22                     Qt <= 0;
23                 else
24                     Qt <= Qt + 1;
25                 end if;
26             end if;
27     end process;
28     Q <= conv_std_logic_vector(Qt,4);
29
30     a <= '1' when Qt = 2 else '0';
31     b <= '1' when Qt = 10 else '0';
32     c <= '1' when Qt = 14 else '0';
33     d <= '1' when Qt = 16 else '0';
34     e <= '1' when Qt = 24 else '0';
35     f <= '1' when Qt = 28 else '0';
36
37 end Behavioral;
```

VHDL Code for Genpulse

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
use ieee.math_real.log2;
use ieee.math_real.ceil;

entity my_genpulse is
    generic (COUNT: INTEGER:= (10**8)/2); -- (10**8)/2 cycles of T = 10 ns --> 0.5 s
    port (clock, resetn, EN: in std_logic;
            Q: out std_logic_vector ( integer(ceil(log2(real(COUNT)))) - 1 downto 0);
            z: out std_logic);
end my_genpulse;

architecture Behavioral of my_genpulse is
    constant nbits: INTEGER:= integer(ceil(log2(real(COUNT))));
    signal Qt: std_logic_vector (nbits -1 downto 0);
begin

    process (resetn, clock)
    begin
        if resetn = '0' then
            Qt <= (others => '0');
        elsif (clock'event and clock = '1') then
            if EN = '1' then
                if Qt = conv_std_logic_vector (COUNT-1,nbits) then
                    Qt <= (others => '0');
                else
                    Qt <= Qt + conv_std_logic_vector (1,nbits);
                end if;
            end if;
        end if;
    end process;

    z <= '1' when Qt = conv_std_logic_vector (COUNT-1,nbits) else '0';
   Q <= Qt;

end Behavioral;
```

VHDL Code for Top File

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use ieee.math_real.log2;
use ieee.math_real.ceil;

entity top is
    port( resetn, clk : in std_logic;
          gNS, gEW, yNS, yEW, rNS, rEW : out std_logic);
end top;

architecture struct of top is


component fsm
port (clock,resetn: in std_logic;
      a, b, c, d, e, f : in std_logic;
      rNS, yNS, gNS, rEW, yEW, gEW: out std_logic);
end component;

component counter
    port ( clock, resetn, z: in std_logic;
             Q: out std_logic_vector (3 downto 0);
             a, b, c, d, e, f: out std_logic);
end component;
```

```vhdl
component my_genpulse
generic (COUNT: INTEGER:= (10**8)/2); -- (10**8)/2 cycles of T = 10 ns --> 0.5 s
    port (clock, resetn, EN: in std_logic;
            Q: out std_logic_vector ( integer(ceil(log2(real(COUNT)))) - 1 downto 0);
            z: out std_logic);
            end component;
signal a, b, c, d, e, f, z : std_logic;

begin

a1: fsm port map (clock => clk, resetn=>resetn,
                  rNS => rNS, yNS => yNS, gNS => gNS, rEW => rEW,
                  yEW => yEW, gEW => gEW,
                  a => a, b => b, c => c, d => d, e => e, f => f);
a2: counter port map (clock => clk, resetn => resetn, a => a, b => b, c => c, d => d, e => e, f => f, q => open, z => z);
--z <= '1'; -- only for simulations
a3: my_genpulse port map (clock => clk, resetn => resetn, EN => '1', Q => open, z => z);

END STRUCT;
```

VHDL Code for Testbench

```
9  ENTITY top_tb IS
10 END top_tb;
11
12 ARCHITECTURE behavior OF top_tb IS
13
14     -- Component Declaration for the Unit Under Test (UUT)
15
16     COMPONENT top
17     Port (resetn, clk : in std_logic;
18           gNS, gEW, yNS, yEW, rNS, rEW : out std_logic
19          );
20     END COMPONENT;
21
22 signal clk, resetn : std_logic;
23 signal gNS, gEW, yNS, yEW, rNS, rEW : std_logic;
24
25 BEGIN
26
27     -- Instantiate the Unit Under Test (UUT)
28    uut: top PORT MAP (
29           clk => clk,
30           resetn => resetn,
31           gNS => gNS,
32           gEW => gEW,
33           yNS => yNS,
34           yEW => yEW,
35           rNS => rNS,
36           rEW => rEW);
```

```
38 clock_process: process
39     begin
40     clk <= '0'; wait for 5 ns;
41     clk <= '1'; wait for 5 ns;
42     end process;
43
44     -- Stimulus process
45    stim_proc: process
46    begin
47       -- hold reset state for 100 ns.
48       wait for 100 ns;
49       --resetn <= '1';
50       -- insert stimulus here
51     wait;
52     end process;
53 END;
```

Thank you!

You're Welcome :)