Fall 2021

California State University, Northridge

Department of Electrical and Computer Engineering

Homework 0

September 8, 2021

ECE 524

Professor: Shahnam Mirzaei

Written By: Morris Blaustein

1. Design Source

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity barrel_shift is
   Port ( DIN : in STD_LOGIC_VECTOR (15 downto 0);
          SHIFT_AMT : in STD_LOGIC_VECTOR (3 downto 0);
          CLK : in STD_LOGIC;
          ENB : in STD_LOGIC;
          DOUT : out STD_LOGIC_VECTOR (15 downto 0));
end barrel_shift;

architecture Behavioral of barrel_shift is
signal patch1, patch2, patch4 : std_logic_vector(15 downto 0);
signal shift1, shift2, shift4, shift8 : std_logic;

begin

shift1 <= SHIFT_AMT(0);
shift2 <= SHIFT_AMT(1);
shift4 <= SHIFT_AMT(2);
shift8 <= SHIFT_AMT(3);

process (clk)
begin
if (clk'event and clk ='1') then
   if ENB = '1' then
      case shift1 is
         when '0' => patch1 <= DIN;
         when '1' => patch1 <= DIN(14 downto 0) & DIN(15);
         when others => null;
      end case;
      case shift2 is
         when '0' => patch2 <= patch1;
         when '1' => patch2 <= patch1(13 downto 0) & patch1(15 downto 14);
         when others => null;
      end case;
      case shift4 is
         when '0' => patch4 <= patch2;
         when '1' => patch4 <= patch2(11 downto 0) & patch2(15 downto 12);
         when others => null;
      end case;
      case shift8 is
         when '0' => DOUT <= patch4;
```

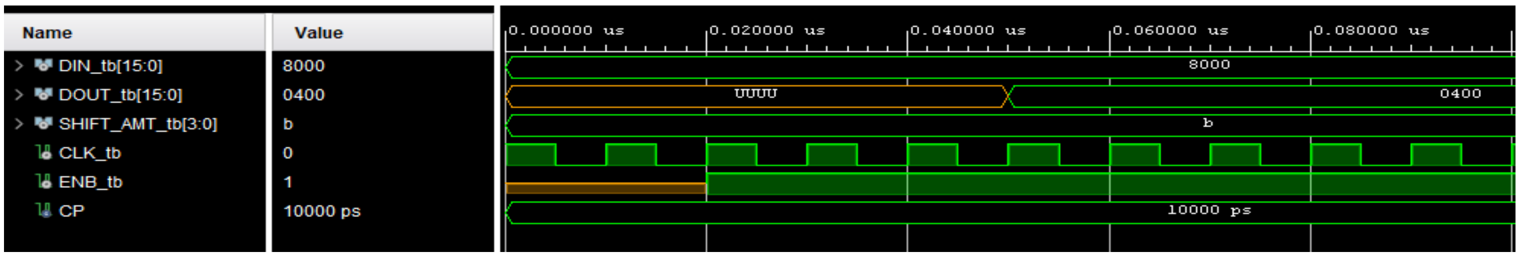
```
            when '1' => DOUT <= patch4(7 downto 0) & patch4(15 downto 8);
            when others => null;
        end case;
    end if;
end if;

end process;
end Behavioral;
```

1. Simulation Source

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity tb_barrel_shift is
-- Port();
end tb_barrel_shift;

architecture Behavioral of tb_barrel_shift is
component barrel_shift is
   Port ( DIN : in STD_LOGIC_VECTOR (15 downto 0);
          SHIFT_AMT : in STD_LOGIC_VECTOR (3 downto 0);
          CLK : in STD_LOGIC;
          ENB : in STD_LOGIC;
          DOUT : out STD_LOGIC_VECTOR (15 downto 0));
end component barrel_shift;

signal DIN_tb, DOUT_tb : std_logic_vector(15 downto 0);
signal SHIFT_AMT_tb : std_logic_vector(3 downto 0);
signal CLK_tb, ENB_tb : std_logic;

constant CP: time := 10ns;

begin
uut : barrel_shift port map( DIN => DIN_tb, SHIFT_AMT => SHIFT_AMT_tb, CLK => CLK_tb, ENB =>
ENB_tb,  DOUT => DOUT_tb );

process
begin
CLK_tb <= '1';
wait for CP/2;
CLK_tb <= '0';
wait for CP/2;
end process;
```

```
process
begin
DIN_tb <= "1000000000000000";
SHIFT_AMT_tb <= "1011";
wait for 20ns;
ENB_tb <= '1';
wait;
end process;
end Behavioral;
```

1. Simulation



Name	Value
DIN_tb[15:0]	8000
DOUT_tb[15:0]	0400
SHIFT_AMT_tb[3:0]	b
CLK_tb	0
ENB_tb	1
CP	10000 ps

2.

The following process will run whenever there Is a change to CLK or RST. Thus, it will reset the flip flop to 0 regardless of the state of the clock when RST goes to 1.

```
process (CLK, RST)
begin
if (RST = ‘1’) then
q <= ‘0’;
end if;
if (CLK`event and CLK=`1`) then
q <= d;
end if;
end process;
```

3. Design Source

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity clk_div_2 is
   Port ( clk : in STD_LOGIC;
          div_clk : out STD_LOGIC);
end clk_div_2;

architecture Behavioral of clk_div_2 is
```

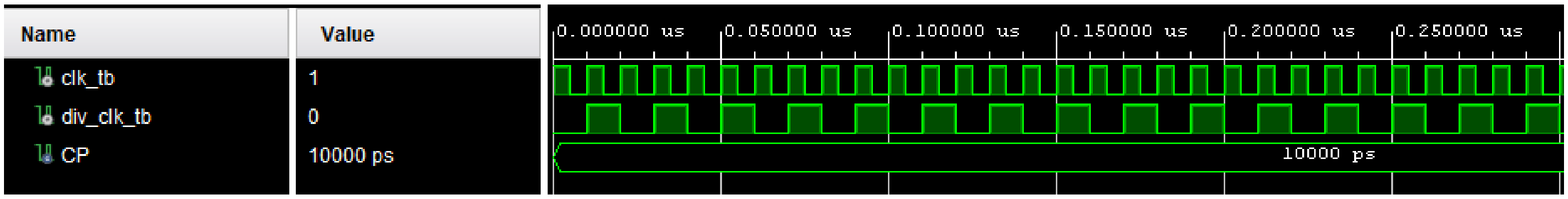
```
signal ct : integer:=0;
signal tmp : std_logic := '0';

begin
process(clk)
begin
if ( clk'event ) then
    ct <= ct + 1;
    if ( ct = 1 ) then
        tmp <= NOT tmp;
        ct <= 0;
    end if;
end if;
div_clk <= tmp;

end process;
end Behavioral;
```

3. Simulation Source

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity tb_clk_div_2 is
-- Port ( );
end tb_clk_div_2;

architecture Behavioral of tb_clk_div_2 is
component clk_div_2 is
    Port ( clk : in STD_LOGIC;
           div_clk : out STD_LOGIC);
end  component clk_div_2;

signal clk_tb, div_clk_tb: STD_LOGIC;

constant CP : time:= 10ns;

begin
uut : clk_div_2 port map(clk => clk_tb, div_clk => div_clk_tb);

process
begin
clk_tb <= '1';
wait for CP/2;
```

```
clk_tb <= '0';
wait for CP/2;
end process;

end Behavioral;
```

3. Simulation



4. Design Source

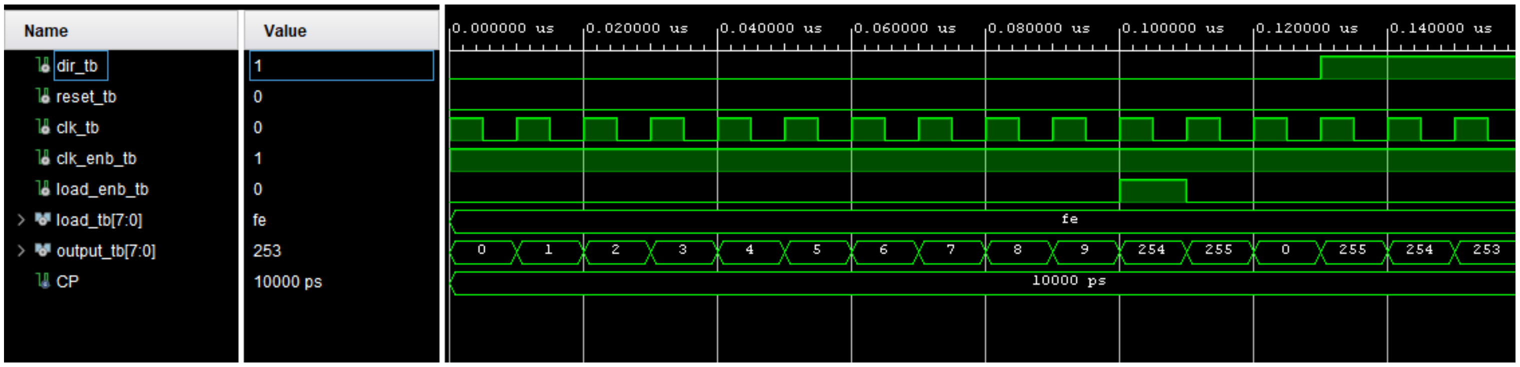
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity up_down_ctr is
    Port ( dir : in STD_LOGIC;
           reset : in STD_LOGIC;
           clk : in STD_LOGIC;
           clk_enb : in STD_LOGIC;
           load : in STD_LOGIC_VECTOR(7 downto 0);
           load_enb : in STD_LOGIC;
           output : out STD_LOGIC_VECTOR(7 downto 0));
end up_down_ctr;

architecture Behavioral of up_down_ctr is

begin
process(clk,reset)
variable count : integer := 0;
begin
    if (reset = '1') then
        count := 0;
    elsif (rising_edge(clk) and clk_enb = '1') then
        if dir = '0' then
            count := count + 1;
            if count > 255 then
                count := 0;
            end if;
        end if;
        if dir = '1' then
            count := count - 1;
            if count < 0 then
                count := 255;
            end if;
        end if;
        if load_enb = '1' then
            count := to_integer(unsigned(load));
        end if;
    end if;
output <= std_logic_vector(to_unsigned(count,output'length));
end process;
end Behavioral;
```

4. Simulation Source

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity tb_up_down_ctr is
--  Port ( );
end tb_up_down_ctr;

architecture Behavioral of tb_up_down_ctr is
component up_down_ctr is
    Port ( dir : in STD_LOGIC;
           reset : in STD_LOGIC;
           clk : in STD_LOGIC;
           clk_enb : in STD_LOGIC;
           load : in STD_LOGIC_VECTOR(7 downto 0);
           load_enb : in STD_LOGIC;
           output : out STD_LOGIC_VECTOR(7 downto 0));
end component up_down_ctr;

signal dir_tb, reset_tb, clk_tb, clk_enb_tb, load_enb_tb : std_logic;
signal load_tb, output_tb : std_logic_vector(7 downto 0);
constant CP : time := 10ns;

begin
uut : up_down_ctr port map ( dir => dir_tb, reset => reset_tb, clk => clk_tb, clk_enb => clk_enb_tb, load
=> load_tb, load_enb => load_enb_tb, output => output_tb );

process
begin
```

```
clk_tb <= '1';
wait for CP/2;
clk_tb <= '0';
wait for CP/2;
end process;

process
begin
dir_tb <= '0';
reset_tb <= '0';
clk_enb_tb <= '1';
load_tb <= "11111110";
load_enb_tb <= '0';
wait for 10*CP;
load_enb_tb <= '1';
wait for CP;
load_enb_tb <= '0';
wait for 2*CP;
dir_tb <= '1';
wait for 10*CP;
wait;
end process;

end Behavioral;
```

4. Simulation



5. Design Source

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity fixed_clk is
   generic (num_cycles : integer := 1);
```

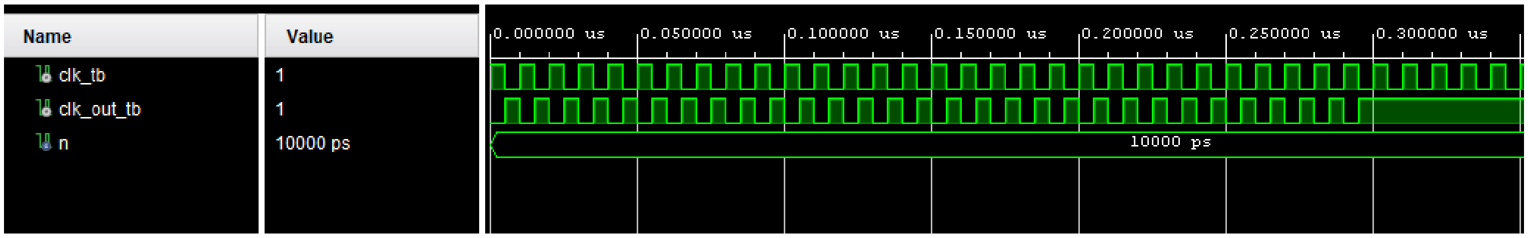
```
    Port (  clk : in STD_LOGIC;
            clk_out : out STD_LOGIC);
end fixed_clk;

architecture Behavioral of fixed_clk is

begin
process(clk)
variable temp : integer := 0;
variable temp2: std_logic := '0';
begin
if ( clk'event and clk='1' ) then
    temp := temp + 1;
end if;
if ( temp <= num_cycles ) then
    temp2 := not temp2;
end if;
clk_out <= temp2;
end process;

end Behavioral;
```

5. Simulation Source

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity tb_fixed_clk is
--  Port ( );
end tb_fixed_clk;

architecture Behavioral of tb_fixed_clk is
component fixed_clk is
    generic (num_cycles : integer);
    Port (  clk : in STD_LOGIC;
            clk_out : out STD_LOGIC);
end component fixed_clk;

signal clk_tb, clk_out_tb : STD_LOGIC;

constant n : time := 10ns;

begin
uut : fixed_clk  generic map ( num_cycles =>30 )
                port map ( clk => clk_tb, clk_out => clk_out_tb );
```

```
process
begin
clk_tb <= '1';
wait for n/2;
clk_tb <= '0';
wait for n/2;
end process;

end Behavioral;
```

5. Simulation



6. Design Source

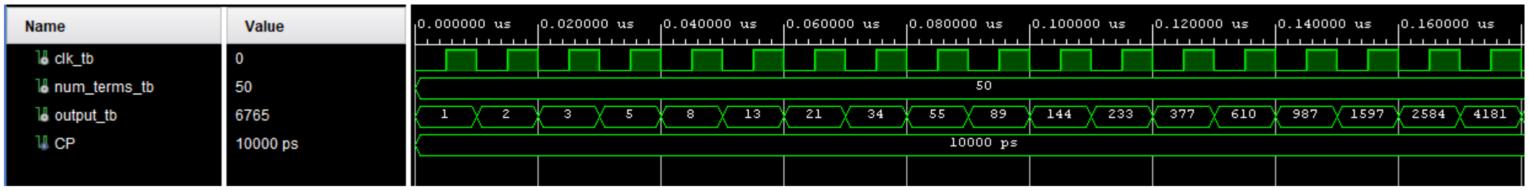
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity fib is
   Port ( clk : in STD_LOGIC;
          num_terms : in integer;
          output : out integer);
end fib;

architecture Behavioral of fib is

signal x : integer;
signal y : integer := 1;
signal z : integer := 0;

begin
process (clk)
variable count : integer := 0;
begin
x <= y + z;
if (rising_edge(clk) and count < num_terms) then
   z <= y;
   y <= x;
   count := count + 1;
```

```
end if;
end process;

output <= x;

end Behavioral;
```

6. Simulation Source

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity tb_fib is
--  Port ( );
end tb_fib;

architecture Behavioral of tb_fib is
component fib is
   Port ( clk : in STD_LOGIC;
          num_terms : in integer;
          output : out integer);
end component fib;

signal clk_tb : std_logic;
signal num_terms_tb, output_tb : integer;
constant CP : time := 10ns;

begin
uut : fib port map(clk=>clk_tb,num_terms=>num_terms_tb,output=>output_tb);

process
begin
clk_tb <= '0';
wait for CP/2;
clk_tb <= '1';
wait for CP/2;
end process;

process
begin
num_terms_tb <= 50;
wait;
end process;

end Behavioral;
```

6. Simulation

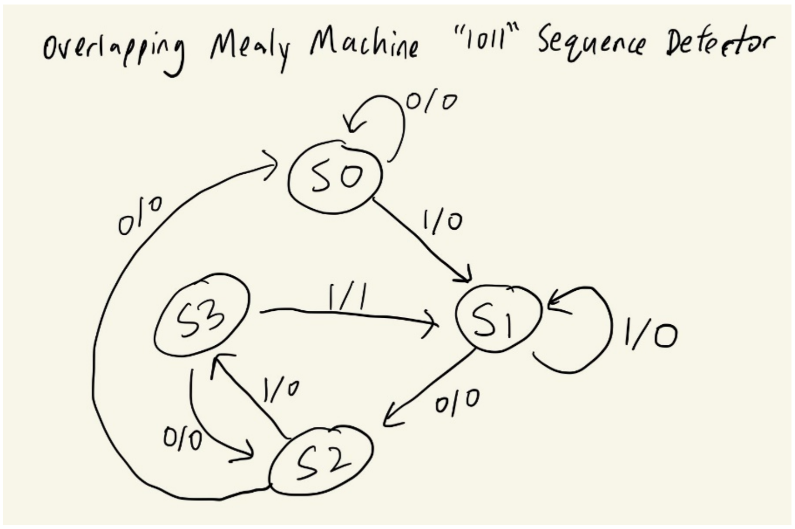


7. Design Source

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity seq_det is
   Port ( input : in STD_LOGIC;
          clk : in STD_LOGIC;
          reset : in STD_LOGIC;
          output : out STD_LOGIC);
end seq_det;

architecture Behavioral of seq_det is

type state_type is (s0,s1,s2,s3);
signal state, next_state : state_type;

begin
process (clk,reset)
begin
if (reset ='1') then
   state <= s0;
elsif (rising_edge(clk)) then
   state <= next_state;
end if;
end process;

process (state, input)
begin
case state is
   when s0 =>
      output <= '0';
      if (input = '0') then
         next_state <= s0;
      else
```

```
            next_state <= s1;
        end if;
    when s1 =>
        output <= '0';
        if (input = '0') then
            next_state <= s2;
        end if;
    when s2 =>
        output <= '0';
        if (input = '0') then
            next_state <= s0;
        else
            next_state <= s3;
        end if;
    when s3 =>
        if (input = '0') then
            next_state <= s2;
            output <= '0';
        else
            next_state <= s1;
            output <= '1';
        end if;
end case;
end process;
end Behavioral;
```

7.

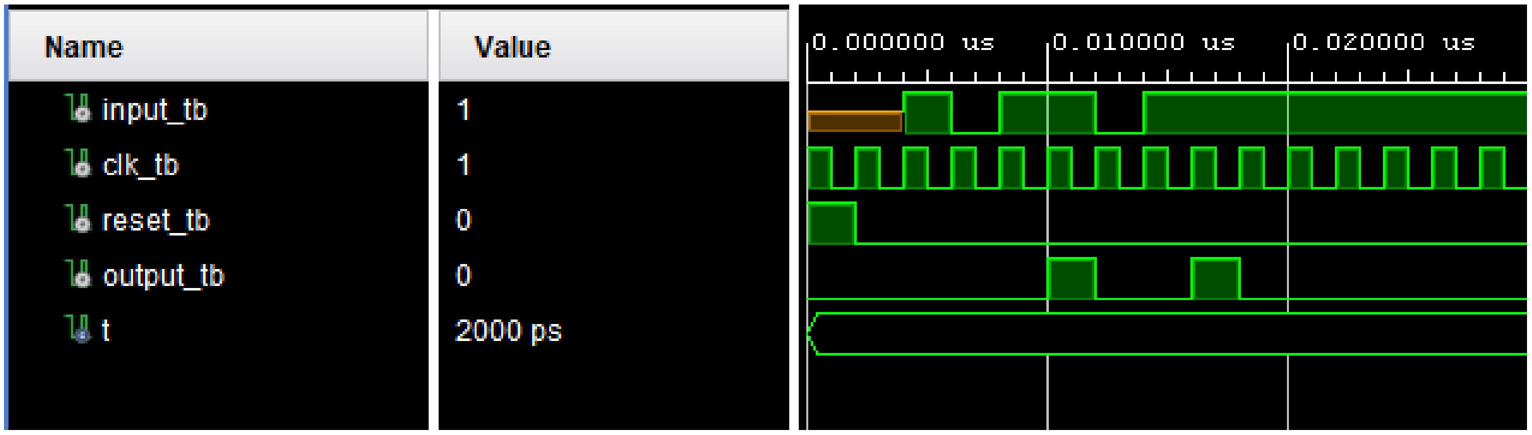


7. Simulation Source

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity tb_seq_det is
--  Port ( );
end tb_seq_det;

architecture Behavioral of tb_seq_det is
component seq_det is
    Port ( input : in STD_LOGIC;
           clk : in STD_LOGIC;
           reset : in STD_LOGIC;
           output : out STD_LOGIC);
end component seq_det;

signal input_tb, clk_tb, reset_tb, output_tb : STD_LOGIC;\

constant t : time := 2ns;

begin
uut : seq_det port map ( input => input_tb, clk => clk_tb, reset => reset_tb, output => output_tb );

process
begin
clk_tb <= '1';
wait for t/2;
clk_tb <= '0';
wait for t/2;
end process;

process
begin
reset_tb <='1';
wait for t;
reset_tb <='0';
wait for t;
input_tb <= '1';
wait for t;
input_tb <= '0';
wait for t;
input_tb <= '1';
wait for t;
input_tb <= '1';
```

```
wait for t;
input_tb <= '0';
wait for t;
input_tb <= '1';
wait for t;
input_tb <= '1';
wait;
end process;
end Behavioral;
```

7. Simulation



8. Design Source

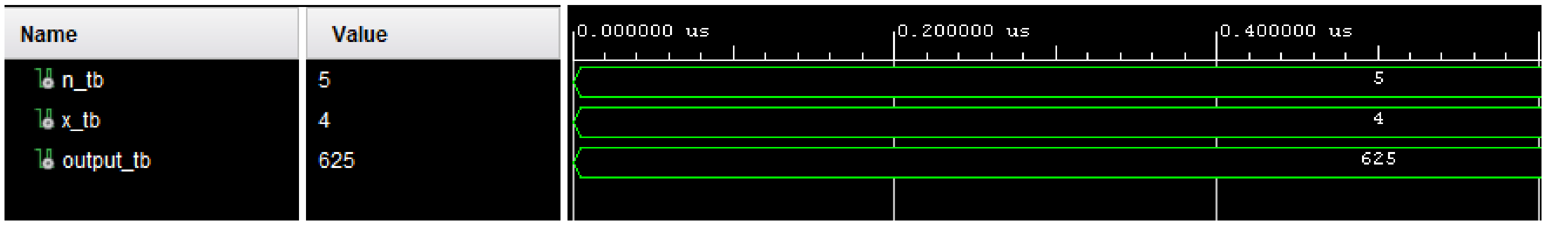
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity exp is
   Port ( n : in integer;
         x : in integer;
         output : out integer);
end exp;

architecture Behavioral of exp is

function NEXP( n : integer; x : integer )
   return integer is
variable temp : integer := 1;
begin
for I in 1 to 100 loop
   if I <= x then
      temp := temp*n;
```

```
    end if;
end loop;
return temp;
end NEXP;

begin
output <= NEXP(n,x);

end Behavioral;
```

8. Simulation Source

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity exp_tb is
--  Port ( );
end exp_tb;

architecture Behavioral of exp_tb is
component exp is
    Port ( n : in integer;
           x : in integer;
           output : out integer);
end component exp;

signal n_tb,x_tb,output_tb : integer;

begin
uut : exp port map ( n => n_tb, x => x_tb, output => output_tb );

process
begin
n_tb <= 5;
x_tb <= 4;
wait for 10ns;
wait;
end process;

end Behavioral;
```

8. Simulation



```
n_tb       5
x_tb       4
output_tb  625
```

9.

```
function COUNT_ZEROES( WORD : bit_vector(15 downto 0) )
    return integer is
variable count : integer := 0;
begin
for I in 0 to 15 loop
    if WORD(I) = '0' then
        count := count + 1;
    end if;
end loop;
return count;
end COUNT_ZEROES;
```

10.

```
function MAX(  a : integer; b : integer )
    return integer is
begin
if a > b then
    return a;
else
    return b;
end if;
end MAX;
```

11. Design Source

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity piso is
   Port ( mode : in STD_LOGIC_VECTOR (1 downto 0);
          input : in STD_LOGIC_VECTOR (7 downto 0);
          clk : in STD_LOGIC;
          rst : in STD_LOGIC;
```

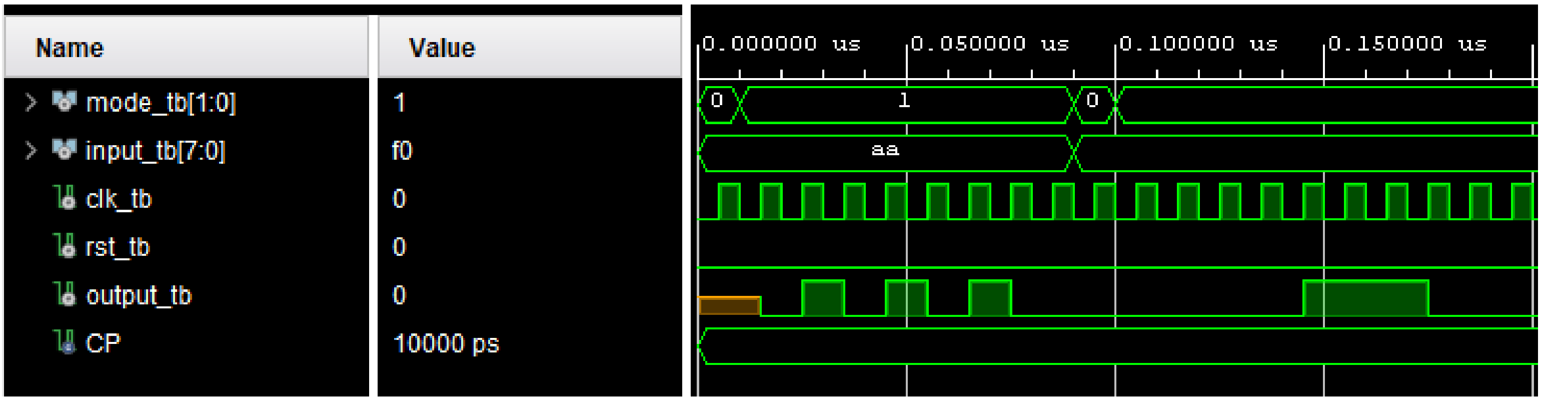
```
        output : out STD_LOGIC);
end piso;

architecture Behavioral of piso is

signal reg : STD_LOGIC_VECTOR(7 downto 0);

begin
process(clk,rst)
variable index : integer := 0;
begin
if rising_edge(clk) then
   if ( rst = '1' ) then
      reg <= ( others => '0' );
      output <= '0';
      index := 0;
   else
      case mode is
         when "00" =>
            reg <= input;
            index := 0;
         when "01" =>
            if index < 8 then
               output <= reg(index);
               index := index + 1;
            end if;
         when others => null;
      end case;
   end if;
   if index = 8 then
      output <= '0';
   end if;
end if;
end process;
end Behavioral;
```

11. Simulation Source

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity piso_tb is
--  Port ( );
end piso_tb;
```

```
architecture Behavioral of piso_tb is
component piso is
   Port ( mode : in STD_LOGIC_VECTOR (1 downto 0);
          input : in STD_LOGIC_VECTOR (7 downto 0);
          clk : in STD_LOGIC;
          rst : in STD_LOGIC;
          output : out STD_LOGIC);
end component piso;

signal mode_tb : std_logic_vector(1 downto 0);
signal input_tb : std_logic_vector(7 downto 0);
signal clk_tb, rst_tb, output_tb : std_logic;

constant CP : time := 10ns;
begin
uut : piso port map( mode => mode_tb, input => input_tb, clk => clk_tb, rst => rst_tb, output =>
output_tb);

process
begin
clk_tb <= '0';
wait for CP/2;
clk_tb <= '1';
wait for CP/2;
end process;

process
begin
rst_tb <= '0';
input_tb <= "10101010";
mode_tb <= "00";
wait for CP;
mode_tb <= "01";
wait for 8*CP;
input_tb <= "11110000";
mode_tb <= "00";
wait for CP;
mode_tb <= "01";
wait for 8*CP;
wait;
end process;

end Behavioral;
```

11. Simulation



12.

```
constant PERIOD : time := 10ns;
constant DUTY_CYCLE : time := 2ns;
process
begin
clk_tb <= ‘0’;
wait for (PERIOD – DUTY_CYCLE);
clk_tb <= ‘1’;
wait for DUTY_CYCLE;
end process;
```

13.

```
function PARITY( a: std_logic_vector(31 downto 0) )
    return std_logic us
variable parity : std_logic := ‘0’;
begin
for I in a’range loop
    parity := parity xor a(i);
end loop
return parity;   --returns 1 if a has even parity
end PARITY;
```