CE 1911

Homework 5

1 – Write behavioral VHDL code for a special 4 bit wrapping up counter with an additional input called mode. If mode is 0 the counter counts normally, if mode is 1 the counter operates as a mod 12 counter. Code, schematic, simulation 50pts

```
----------------------------------------
--
-- counter_sel_mod12_4bit.vhdl
--
-- created 3/27/18
-- tj
--
-- rev 0
----------------------------------------
--
-- 4 bit counter with mod12 mode for hw problem
--
----------------------------------------
--
-- Inputs: rstb, clk, mode
-- Outputs: cnt[3:0]
--
----------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity counter_sel_mod12_4bit is
   port (
      i_clk :  in std_logic;
      i_rstb : in std_logic;
      i_mode : in std_logic;  -- 0 for mod16, 1 for mod 12

      o_cnt :  out std_logic_vector(3 downto 0)
    );
end entity;
```

Approx 17 lines of code

```
architecture behavioral of counter_sel_mod12_4bit is

   --
   -- internal signals
   --
   signal cnt_sig:   unsigned(3 downto 0);
```

```
   --
   -- Output logic
   --
   o_cnt <= std_logic_vector(cnt_sig);

end behavioral;
```

1 – Write behavioral VHDL code for a special 4 bit wrapping up counter with an additional input called mode. If mode is 0 the counter counts normally, if mode is 1 the counter operates as a mod 12 counter. Code, schematic, simulation 50pts

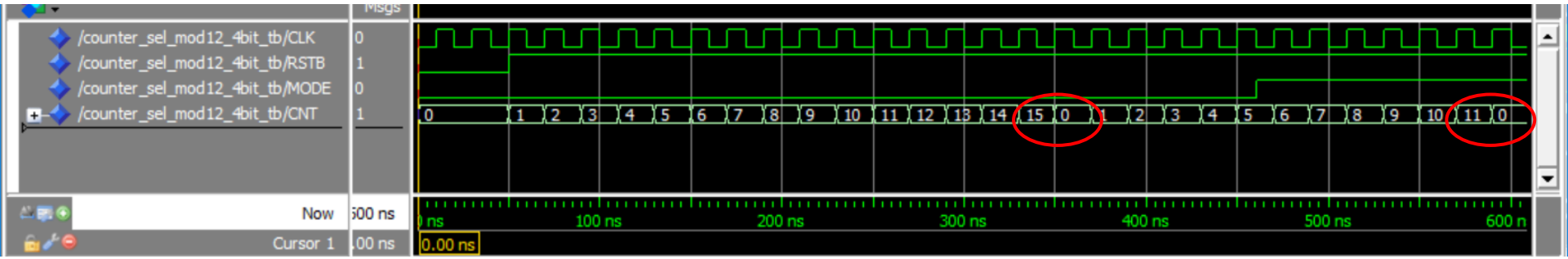
```
--------------------------------
--
-- counter_sel_mod12_4bit_tb.vhdl
--
-- created: 1/26/18
-- by: johnsontimoj
-- rev: 0
--
-- testbench for  bit counter with mod12 mode for hw pro
-- of counter_sel_mod12_4bit.vhdl
--
-- brute force implementation
--
--------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity counter_sel_mod12_4bit_tb is
   -- no entry - testbench
end entity;

architecture testbench of counter_sel_mod12_4bit_tb is
   signal   CLK:     std_logic;
   signal   RSTB:    std_logic;
   signal   MODE:    std_logic;

   signal   CNT:     std_logic_vector(3 downto 0);

   constant PER:     time  := 20 ns;

   ---------------------------------
   -- Component prototype
   ---------------------------------
   COMPONENT counter_sel_mod12_4bit
      port (
            i_clk :  in std_logic;
            i_rstb : in std_logic;
            i_mode : in std_logic;  -- 0 for mod16, 1 fo

            o_cnt  : out std_logic_vector(3 downto 0)
         );
   END COMPONENT;
   ---------------------------------
```

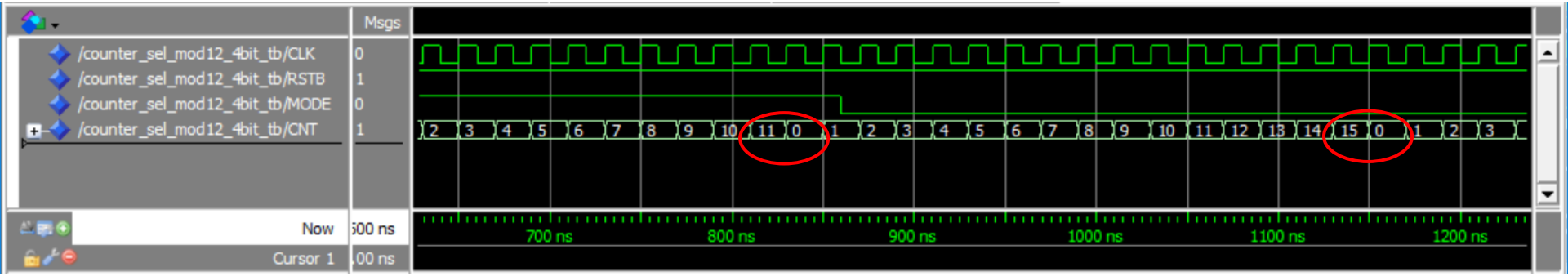
Testbench

```
   -- Run process
   run: process      -- note - no sensitivity
      begin
         -- Initialize  inputs
         MODE <= '0';
         wait for 3*PER;    -- wait for reset
         -- verify mod16
         wait for 20*PER;
         -- verify mod12
         MODE <= '1';
         wait for 20*PER;
         -- reset to mod16
         MODE <= '0';
         wait for 20*PER;
      end process run;

   ---------------------------------
   -- End test processes
   ---------------------------------


end architecture;
```

1 – Write behavioral VHDL code for a special 4 bit up counter with an additional input called mode. If mode is 0 the counter counts normally, if mode is 1 the counter operates as a mod 12 counter. Code, schematic, simulation 50pts

Simulation Results





2 – Modify the behavioral VHDL code for the L/R shift register to add an additional input amt(A). When A is low, the shift register shifts by 1 spot, when A is high the shift register shifts by 2 spots. Instead of shifting in Din, rotate the contents of the register. Reset should put the register in the 1001 0110 state. Provide code and a simulation. There is no Din input! 50pts

```
----------------------------------------
--
-- shiftreg_lr_12_nbit.vhdl
--
-- created 3/27/18
-- tj
--
-- rev 0
----------------------------------------
--
-- n bit 1/2 bit L/R shift register hw problem
--
----------------------------------------
--
-- Inputs: rstb, clk, amt, dir, shift
-- Outputs: reg_out[7:0]
--
----------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity shiftreg_lr_12_nbit is
   generic(
            N:    natural:=   8;
            init: natural:= 150
            );
   port (
      i_clk :     in std_logic;
      i_rstb :    in std_logic;
      i_amt :     in std_logic;   -- 0 for 1, 1 for 2
      i_shift:    in std_logic;   -- 0 for no shift
      i_dir:      in std_logic;   -- 0 for left, 1 for right

      o_reg_out : out std_logic_vector((N - 1) downto 0)
    );
end entity;
```

Approx 8 lines of code change between the example and the solution

2 – Modify the behavioral VHDL code for the L/R shift register to add an additional input amt(A). When A is low, the shift register shifts by 1 spot, when A is high the shift register shifts by 2 spots. Instead of shifting in Din, rotate the contents of the register. Reset should put the register in the 1001 0110 state. Provide code and a

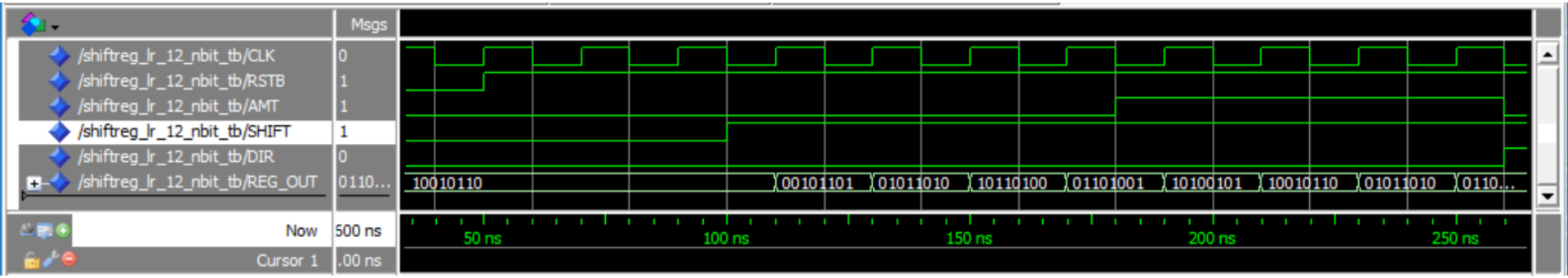
Testbench

```
--------------------------------
--
-- shiftreg_lr_12_nbit_tb.vhdl
--
-- created: 1/26/18
-- by: johnsontimoj
-- rev: 0
--
-- testbench for n bit lr, 1/2 bit shift register
-- of shiftreg_lr_12_nbit.vhdl
--
-- Using the PERIOD construct
--
-- brute force implementation
--
--------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity shiftreg_lr_12_nbit_tb is
   generic(
         N: natural := 8;
         INIT: natural := 150
   );
   -- no entry - testbench
end entity;

architecture testbench of shiftreg_lr_12_nbit_tb is
   signal   CLK:      std_logic;
   signal   RSTB:     std_logic;
   signal   AMT:      std_logic;
   signal   SHIFT:    std_logic;
   signal   DIR:      std_logic;

   signal   REG_OUT: std_logic_vector((N-1) downto 0);

   constant PER:      time  := 20 ns;
```

```
   -- Run process
   run: process      -- note - no sensitivity
      begin
         ------------------------
         -- Initialize  inputs
         AMT <= '0';
         SHIFT <= '0';
         DIR <= '0';
         wait for 3*PER;   -- wait for reset
         -- verify no shift
         wait for PER*2;
         -- verify shift lt 1
         SHIFT <= '1';
         AMT <= '0';
         DIR <= '0';
            wait for PER*4;
         -- verify shift lt 2
         SHIFT <= '1';
         AMT <= '1';
         DIR <= '0';
         wait for PER*4;
         -- verify shift rt 1
         SHIFT <= '1';
         AMT <= '0';
         DIR <= '1';
         wait for PER*4;
         -- verify shift rt 2
         SHIFT <= '1';
         AMT <= '1';
         DIR <= '1';
         wait for PER*4;
      end process run;

   -------------------------------------
   -- End test processes
   -------------------------------------


end architecture;
```

2 – Modify the behavioral VHDL code for the L/R shift register to add an additional input amt(A). When A is low, the shift register shifts by 1 spot, when A is high the shift register shifts by 2 spots. Instead of shifting in Din, rotate the contents of the register. Reset should put the register in the 1001 0110 state. Provide code and a simulation 50pts

Simulation Results

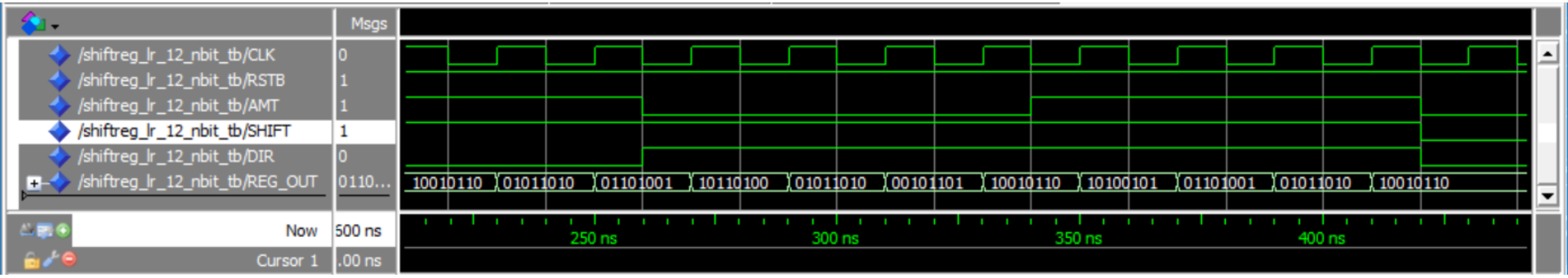


1001 0110 start

no shift

shift left by 1

shift left by 2



shift right by 1

shift right by 2

no shift