Lab 12 – VHDL

Name: Nhut Quang Tran

Group: NTIVIS14

TURKU UNIVERSITY OF APPLIED SCIENCES

**Exercise 1:** **Design the VHDL model of a 4- bit Register**

**(a) Serial-in Serial-out**

**(b) Serial in Parallel out**

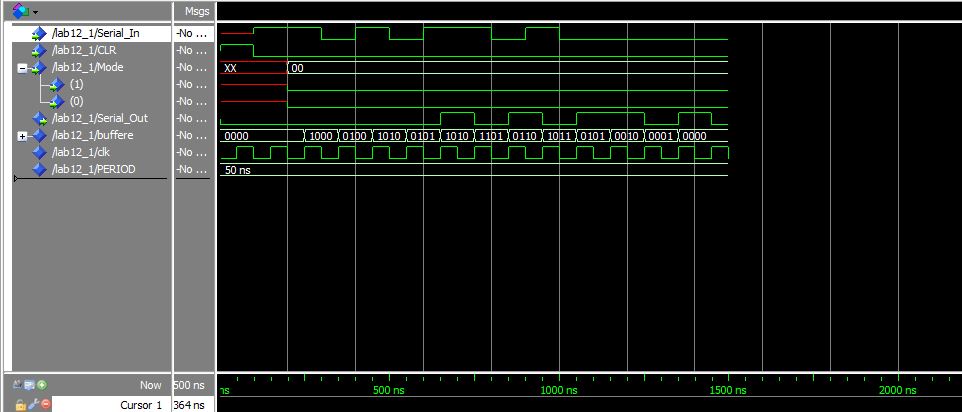
**(c) Parallel in Serial out**

**(d) Parallel in Parallel out**

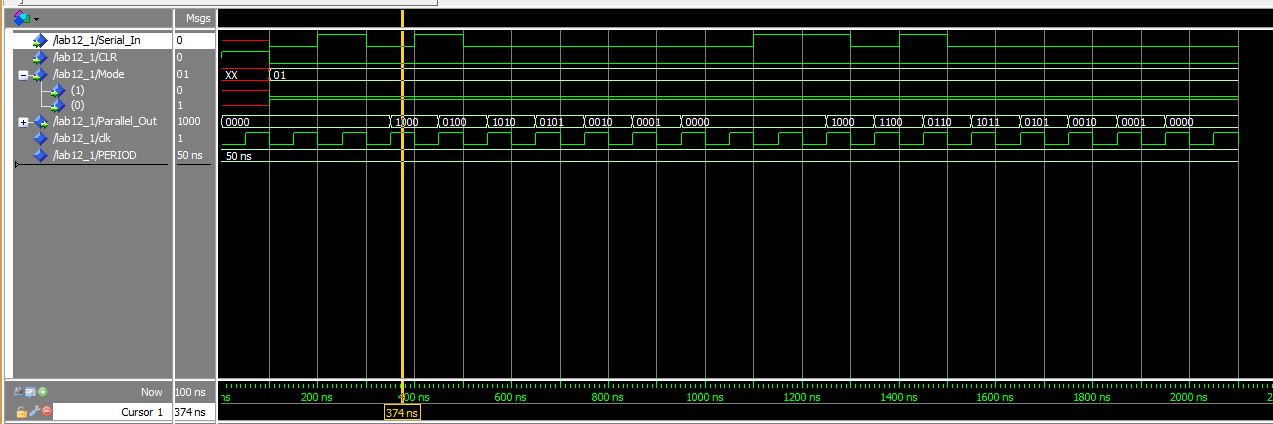
**The code:**

|  |
| --- |
| LIBRARY ieee ;  USE ieee.std\_logic\_1164.all ;  use ieee.numeric\_std.all;  use IEEE.std\_logic\_unsigned.all;  ENTITY Lab12\_1 IS  PORT (Serial\_In: IN STD\_LOGIC ;  Parallel\_In: IN STD\_LOGIC\_VECTOR (3 DOWNTO 0);  Mode: IN STD\_LOGIC\_VECTOR (1 DOWNTO 0);  CLR: IN STD\_LOGIC;  Serial\_Out: OUT STD\_LOGIC;  Parallel\_Out: OUT STD\_LOGIC\_VECTOR (3 DOWNTO 0) );  END Lab12\_1;  ARCHITECTURE bitShifter OF Lab12\_1 IS  signal buffere: unsigned (3 DOWNTO 0) := "0000";  signal n : integer := 0;  constant PERIOD: time := 50ns;  signal clk : std\_logic := '0';  BEGIN  sync\_proc: process(clk)  BEGIN  clk <= not clk after PERIOD;  end process sync\_proc;  Shifter\_proc: process(Serial\_In,Parallel\_In,MODE,buffere,clk,CLR)  BEGIN  case MODE is  when "00" => --Serial in - Serial out  if(CLR = '1') then  Serial\_Out <= '0'; --Reset Serial-Out  elsif (rising\_edge(clk)) then  buffere <= buffere srl 1;  buffere(3) <= Serial\_In;  Serial\_Out <= buffere(0);  end if;  when "01" => --Serial in - Parallel out  if(CLR = '1') then  Parallel\_Out <= "0000"; --Reset Parallel-Out  elsif (rising\_edge(clk)) then  buffere <= buffere srl 1;  buffere(3) <= Serial\_In;  Parallel\_Out <= std\_logic\_vector(buffere);  end if;  when "10" => --Parallel in - Serial out  if(CLR = '1') then  Serial\_Out <= '0'; --Reset Serial-Out  elsif (rising\_edge(clk)) then  n <= n + 1;  buffere <= buffere srl 1;  Serial\_Out <= buffere(0);  end if;  if((n mod 5) = 0) then --Only read the input after 5th period  buffere <= unsigned(Parallel\_In);  end if;  when "11" => --Parallel in - Parallel out  if(CLR = '1') then  Parallel\_Out <= "0000"; --Reset Parallel-Out  elsif (rising\_edge(clk)) then  Parallel\_Out <= Parallel\_In;  end if;  when others =>  Serial\_Out <= '0';  Parallel\_Out <= "0000";  end case;  end process Shifter\_proc;  end bitShifter; |

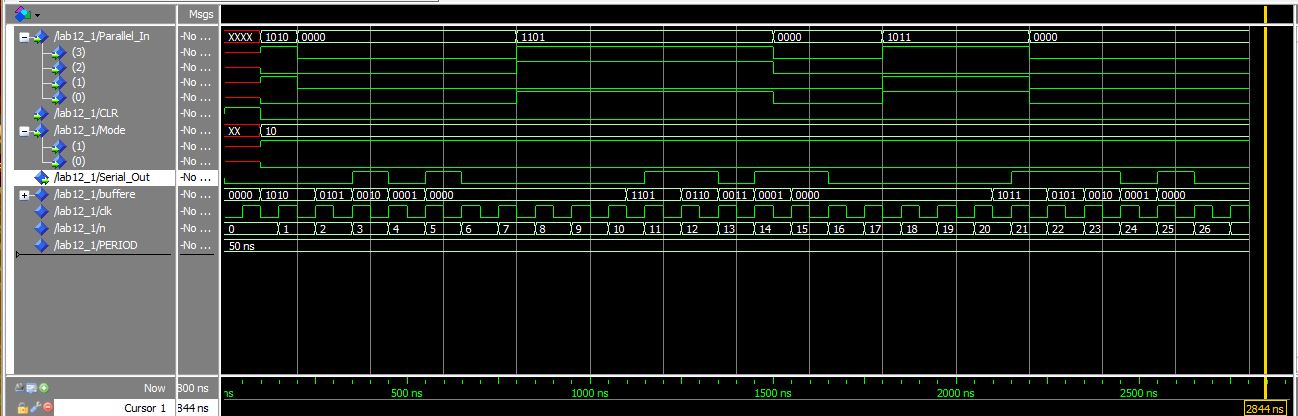
**The simulating picture:**

**(a) Serial-in Serial-out**

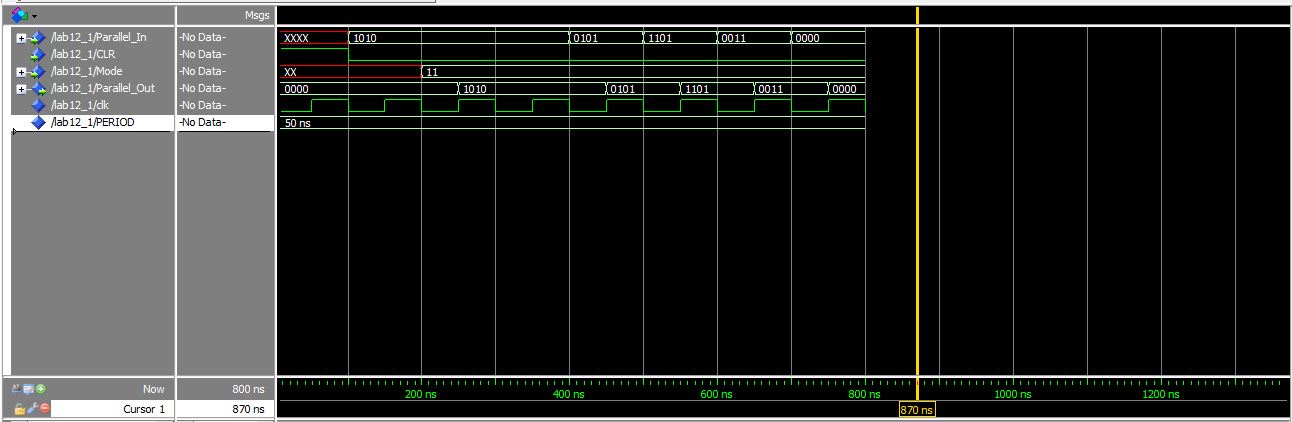
**(b) Serial in Parallel out**

****

**(c) Parallel in Serial out**

****

**(d) Parallel in Parallel out**

****

**Exercise 2: Design the VHDL model of a controllable counter, which has four different sequences. Inputs for this circuit are two control pins (A and B) and one clock. This circuit should produce different patterns to a 7-segment display.**

**The code:**

|  |  |
| --- | --- |
| |  | | --- | | LIBRARY ieee ;  USE ieee.std\_logic\_1164.all ;  use ieee.numeric\_std.all;  use IEEE.std\_logic\_unsigned.all;  ENTITY Lab12\_2 IS  PORT (  Mode: IN STD\_LOGIC\_VECTOR (1 DOWNTO 0);  CLR: IN STD\_LOGIC;  Seven\_Segment: OUT STD\_LOGIC\_VECTOR (6 DOWNTO 0) );  END Lab12\_2;  ARCHITECTURE Seven\_Segment\_Display OF Lab12\_2 IS  signal buffere: unsigned (6 DOWNTO 0) := "0000001";  signal tmp: std\_logic\_vector (3 downto 0) := "0000";  signal Numb\_out: std\_logic\_vector(6 downto 0) := "0000000";  constant PERIOD: time := 50ns;  signal clk : std\_logic := '0';  signal n : integer := 0;  BEGIN  sync\_proc: process(clk)  BEGIN  clk <= not clk after PERIOD;  end process sync\_proc;  Number\_proc: process (tmp)  BEGIN  case tmp is  when "0000" => Numb\_out <= "1000000"; -- '0'  when "0001" => Numb\_out <= "0000110"; -- '1'  when "0010" => Numb\_out <= "1011011"; -- '2'  when "0011" => Numb\_out <= "1001111"; -- '3'  when "0100" => Numb\_out <= "1100110"; -- '4'  when "0101" => Numb\_out <= "1101101"; -- '5'  when "0110" => Numb\_out <= "1111101"; -- '6'  when "0111" => Numb\_out <= "0000111"; -- '7'  when "1000" => Numb\_out <= "1111111"; -- '8'  when "1001" => Numb\_out <= "1101111"; -- '9'  when others => Numb\_out <= "0000000";  end case;  end process Number\_proc;  Display\_proc: process(MODE,buffere,clk,CLR)  BEGIN  case MODE is  when "00" => --Leds are ON one at time in order: A-B-C...-A-B-C...  if(CLR = '1') then  Seven\_Segment <= "0000001"; --Reset Seven-Segment  elsif (rising\_edge(clk)) then  buffere <= buffere rol 1;  Seven\_Segment <= std\_logic\_vector(buffere);  end if;  when "01" => --Leds are ON one at time in order: A-B-C...-G-F-E-D...-C-B-A...  if(CLR = '1') then  Seven\_Segment <= "0000001"; --Reset Seven-Segment  n <= 0;  elsif (rising\_edge(clk)) then  n <= n+1;  if(n < 6) then  buffere <= buffere rol 1;  Seven\_Segment <= std\_logic\_vector(buffere);  elsif(n >= 6) then  buffere <= buffere ror 1;  Seven\_Segment <= std\_logic\_vector(buffere);  if(n >= 11) then  n <= 0;  end if;  end if;  end if;  when "10" => --Display ODD numbers: 1-3-5-7-9...  if(CLR = '1') then  Seven\_Segment <= "0000001"; --Reset Seven-Segment  tmp <= "0001";  n <= 0;  elsif (rising\_edge(clk)) then  n <= n + 1;  if(n < 4) then --Because of initial state is started from 0 (5 numbers)  tmp <= tmp + 2;  elsif(n >= 4) then  tmp <= "0001";  n <= 0;  end if;  Seven\_Segment <= Numb\_out;  end if;  when "11" => --Display ODD numbers: 0-2-4-6-8...  if(CLR = '1') then  Seven\_Segment <= "0000000"; --Reset Seven-Segment  tmp <= "0000";  n <= 0;  elsif (rising\_edge(clk)) then  n <= n + 1;  if(n < 4) then  tmp <= tmp + 2;  elsif(n >= 4) then  tmp <= "0000";  n <= 0;  end if;  Seven\_Segment <= Numb\_out;  end if;  when others => Seven\_Segment <= "0000000";  end case;  end process Display\_proc;  end Seven\_Segment\_Display; | |

**The simulating picture:**

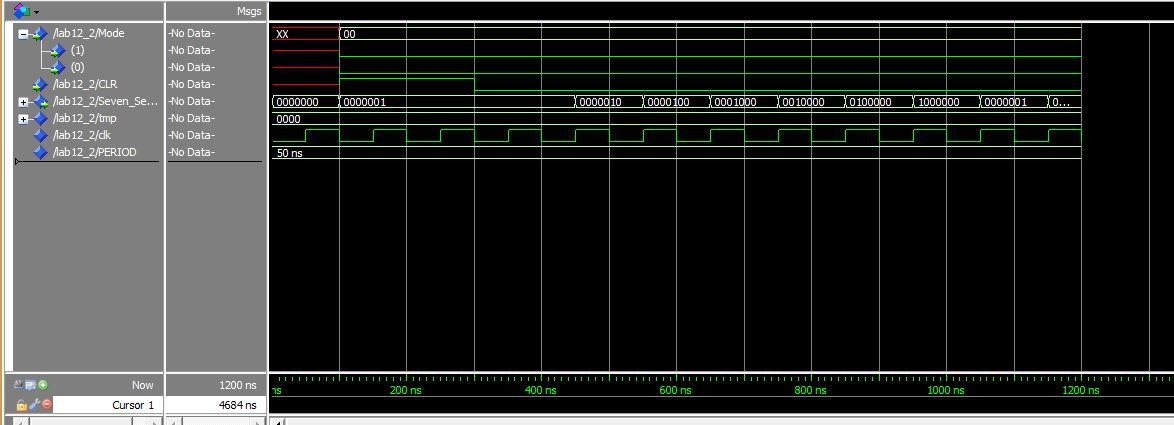
****

Figure 1. LEDs are on at time in order: A-B-C-...-A-B-C..

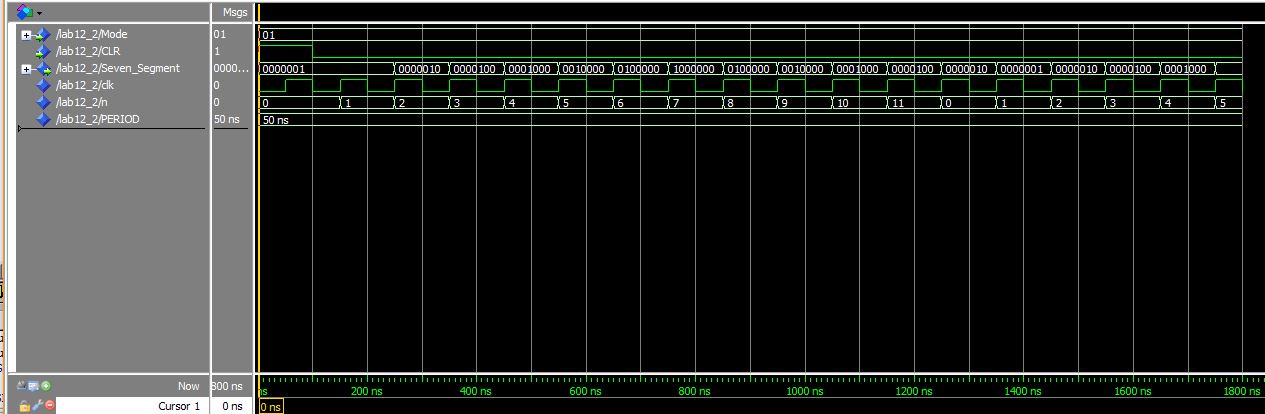
****

Figure 2. LEDs are on at time in order: A-B-C...-F-G-F-E-...

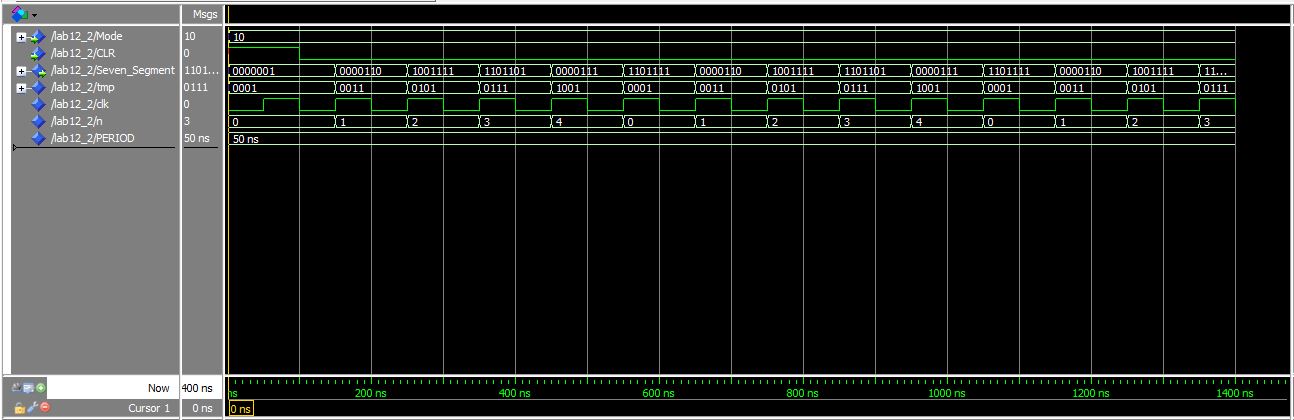
****

Figure 3. Display ODD numbers

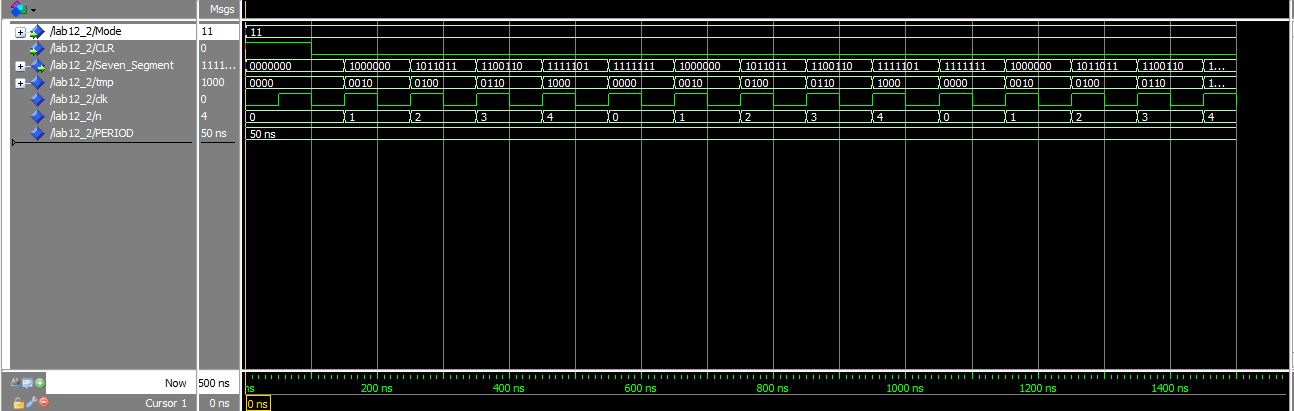
****

Figure 4. Display EVEN numbers