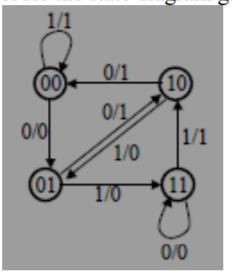
Lab 10 – VHDL

Name: Nhut Quang Tran

Group: NTIVIS14

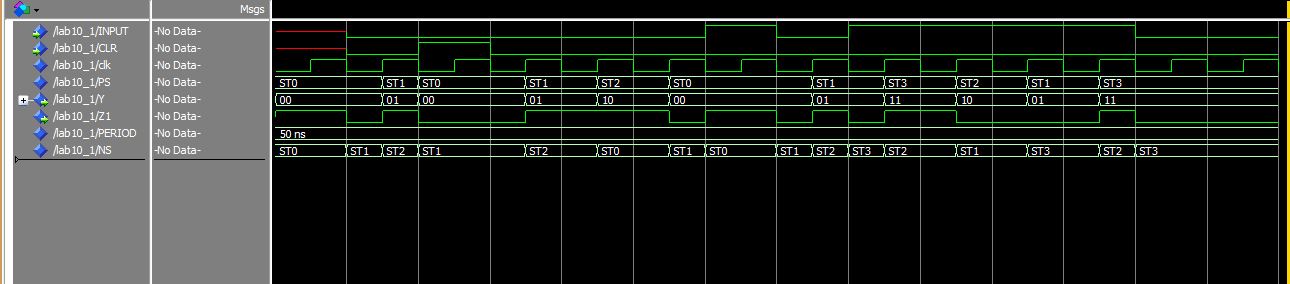
TURKU UNIVERSITY OF APPLIED SCIENCES

**Exercise 1: Make a VHDL model for the state diagram given below.**

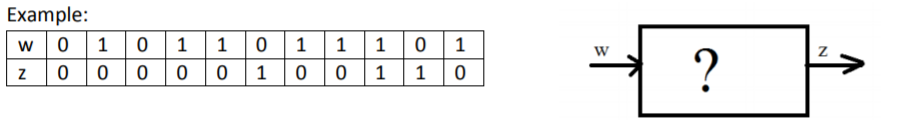
**The code:**

|  |
| --- |
| LIBRARY ieee ;  USE ieee.std\_logic\_1164.all ;  use IEEE.NUMERIC\_STD.all;  use IEEE.std\_logic\_unsigned.all;  ENTITY Lab10\_1 IS  PORT (INPUT, CLR: IN STD\_LOGIC;  Y: OUT STD\_LOGIC\_VECTOR (1 DOWNTO 0);  Z1: OUT STD\_LOGIC);  END Lab10\_1;  ARCHITECTURE FSM OF Lab10\_1 IS  constant PERIOD: time := 50ns;  signal clk : std\_logic := '0';  type state\_type is (ST0,ST1,ST2,ST3);  signal PS,NS : state\_type;  BEGIN  sync\_proc: PROCESS(CLK,CLR,NS)  BEGIN  clk <= not clk after PERIOD/2; -- Generating clock, the period = 50ns.  if(CLR = '1') then PS <= ST0;  elsif (rising\_edge(clk)) then  PS <= NS;  end if;  end process sync\_proc;  comb\_proc: process(PS, INPUT)  begin  case PS is  when ST0 => -- items regarding state ST0    if(INPUT = '0') then  NS <= ST1;  Z1 <= '0'; -- Mealy output  else  NS <= ST0;  Z1 <= '1'; -- Mealy output    end if;  when ST1 => -- items regarding state ST0    if(INPUT = '0') then  NS <= ST2;  Z1 <= '1'; -- Mealy output  else  NS <= ST3;  Z1 <= '0'; -- Mealy output  end if;    when ST2 => -- items regarding state ST0    if(INPUT = '0') then  NS <= ST0;  Z1 <= '1'; -- Mealy output  else  NS <= ST1;  Z1 <= '0'; -- Mealy output  end if;    when ST3 => -- items regarding state ST0    if(INPUT = '0') then  NS <= ST3;  Z1 <= '0'; -- Mealy output  else  NS <= ST2;  Z1 <= '1'; -- Mealy output  end if;  when others => -- the catch-all condition  Z1 <= '1'; -- arbitrary; it should never  Ns <= ST0; -- make it to these two statement  end case;  end process comb\_proc;  with PS select  Y <= "00" when ST0,  "01" when ST1,  "10" when ST2,  "11" when ST3,  "00" when others;  end FSM; |

**The simulating picture:**



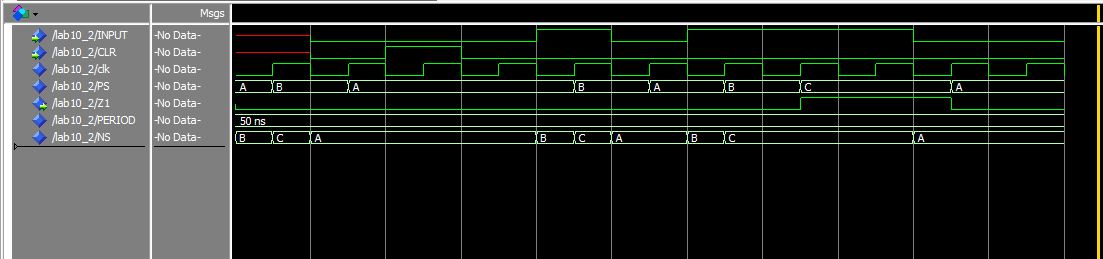
**Exercise 2: Design (VHDL model ) a circuit that: – Has one input (w) and one output (z) – All changes occur on the positive edge of the clock – Output z is equal to 1 if during the two immediately preceding clock cycles the input w was equal to 1. Otherwise z is equal to 0**



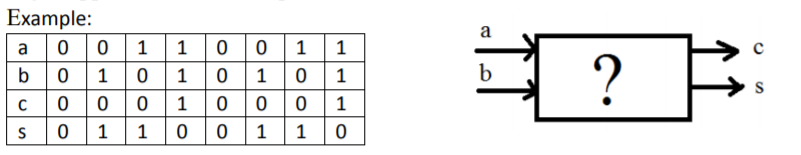
**The code:**

|  |
| --- |
| LIBRARY ieee ;  USE ieee.std\_logic\_1164.all ;  use IEEE.NUMERIC\_STD.all;  use IEEE.std\_logic\_unsigned.all;  ENTITY Lab10\_2 IS  PORT (INPUT, CLR: IN STD\_LOGIC;  Y: OUT STD\_LOGIC\_VECTOR (1 DOWNTO 0);  Z1: OUT STD\_LOGIC);  END Lab10\_2;  ARCHITECTURE FSM OF Lab10\_2 IS  constant PERIOD: time := 50ns;  signal clk : std\_logic := '0';  type state\_type is (A,B,C);  signal PS,NS : state\_type;  BEGIN  sync\_proc: PROCESS(CLK,CLR,NS)  BEGIN  clk <= not clk after PERIOD/2; -- Generating clock, the period = 50ns.  if(CLR = '1') then PS <= A;  elsif (rising\_edge(clk)) then  PS <= NS;  end if;  end process sync\_proc;  comb\_proc: process(PS, INPUT)  begin  case PS is  when A => -- items regarding state ST0  Z1 <= '0'; -- Moore output  if(INPUT = '0') then  NS <= A;  else  NS <= B;  end if;  when B => -- items regarding state ST0  Z1 <= '0'; -- Moore output  if(INPUT = '0') then  NS <= A;    else  NS <= C;  end if;    when C => -- items regarding state ST0  Z1 <= '1'; -- Moore output  if(INPUT = '0') then  NS <= A;  else  NS <= C;  end if;  when others => -- the catch-all condition  Z1 <= '1'; -- arbitrary; it should never  NS <= A; -- make it to these two statement  end case;  end process comb\_proc;  end FSM; |

**The simulating picture:**

****

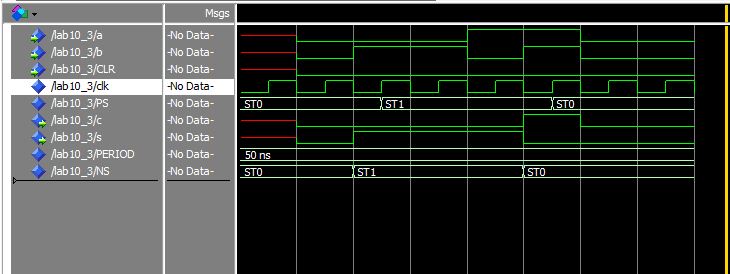
**3. Design (VHDL model) a circuit that: – Has two inputs (a, b) (also a clock) and two outputs (c, s) – All changes occur on the positive edge of the clock – The circuit should work as a serial adder adding bits a and b together every time a rising edge happens in the clock input.**



**The code:**

|  |
| --- |
| LIBRARY ieee ;  USE ieee.std\_logic\_1164.all ;  use IEEE.NUMERIC\_STD.all;  use IEEE.std\_logic\_unsigned.all;  ENTITY Lab10\_3 IS  PORT (a, b: IN STD\_LOGIC ;  CLR: IN STD\_LOGIC;  c, s: OUT STD\_LOGIC);  END Lab10\_3;  ARCHITECTURE FSM OF Lab10\_3 IS  constant PERIOD: time := 50ns;  signal clk : std\_logic := '0';  type state\_type is (ST0,ST1,ST2);  signal PS,NS : state\_type;  BEGIN  sync\_proc: PROCESS(CLK,CLR,NS)  BEGIN  clk <= not clk after PERIOD/2; -- Generating clock, the period = 50ns.  if(CLR = '1') then PS <= ST0;  elsif (rising\_edge(clk)) then  PS <= NS;  end if;  end process sync\_proc;  comb\_proc: process(PS, a, b)  begin  case PS is  when ST0 => -- items regarding state ST0    if(a = '0') and( b = '0') then  NS <= ST0;  c <= '0'; -- Mealy output  s <= '0'; -- Mealy output  elsif(a = '0') and ( b = '1') then  NS <= ST1;  c <= '0'; -- Mealy output  s <= '1'; -- Mealy output  elsif(a = '1') and ( b = '0') then  NS <= ST2;  c <= '0'; -- Mealy output  s <= '1'; -- Mealy output  elsif(a = '1') and ( b = '1') then  NS <= ST0;  c <= '1'; -- Mealy output  s <= '0'; -- Mealy output  end if;  when ST1 => -- items regarding state ST1  if(a = '0') and( b = '0') then  NS <= ST0;  c <= '0'; -- Mealy output  s <= '0'; -- Mealy output  elsif(a = '0') and ( b = '1') then  NS <= ST1;  c <= '0'; -- Mealy output  s <= '1'; -- Mealy output  elsif(a = '1') and ( b = '0') then  NS <= ST2;  c <= '0'; -- Mealy output  s <= '1'; -- Mealy output  elsif(a = '1') and ( b = '1') then  NS <= ST0;  c <= '1'; -- Mealy output  s <= '1'; -- Mealy output  end if;  when ST2 => -- items regarding state ST2  if(a = '0') and( b = '0') then  NS <= ST0;  c <= '0'; -- Mealy output  s <= '0'; -- Mealy output  elsif(a = '0') and ( b = '1') then  NS <= ST1;  c <= '0'; -- Mealy output  s <= '1'; -- Mealy output  elsif(a = '1') and ( b = '0') then  NS <= ST2;  c <= '0'; -- Mealy output  s <= '1'; -- Mealy output  elsif(a = '1') and ( b = '1') then  NS <= ST0;  c <= '1'; -- Mealy output  s <= '0'; -- Mealy output  end if;  when others => -- the catch-all condition  c <= '1'; -- arbitrary; it should never  s <= '1';  NS <= ST0; -- make it to these two statement  end case;  end process comb\_proc;  end FSM; |

**The simulating picture:**

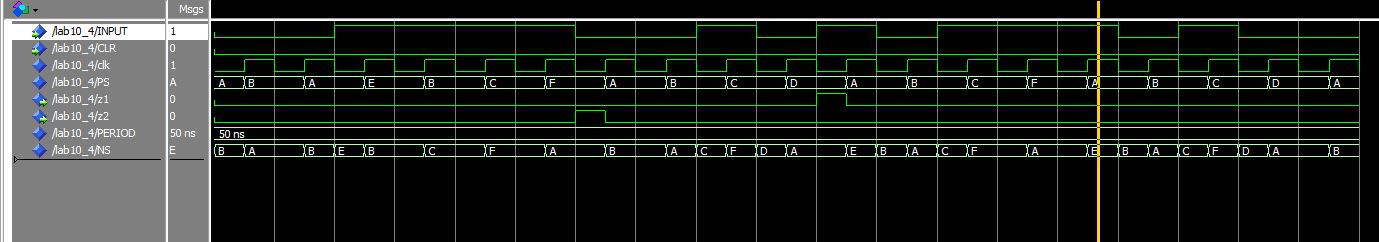
****

**4. Design (VHDL model) a circuit that recognizes and distinguishes the occurrence of two specific sequences in its input stream. The quiescent output pattern is z1z2 = 0. If the sequence x1 = 0101 is applied to the input line x, the output should become 01 when the last bit of x1 appears, and the circuit should return to its reset state A. If a second sequence x2 = 1110 is applied to x, the output should become 10 when the last bit of x2 appears, and again the circuit should return to state A.**

**The code:**

|  |
| --- |
| LIBRARY ieee ;  USE ieee.std\_logic\_1164.all ;  use IEEE.NUMERIC\_STD.all;  use IEEE.std\_logic\_unsigned.all;  ENTITY Lab10\_4 IS  PORT (INPUT, CLR: IN STD\_LOGIC;  z1,z2: OUT STD\_LOGIC);  END Lab10\_4;  ARCHITECTURE FSM OF Lab10\_4 IS  constant PERIOD: time := 50ns;  signal clk : std\_logic := '0';  type state\_type is (A,B,C,D,E,F);  signal PS,NS : state\_type;  BEGIN  sync\_proc: PROCESS(CLK,CLR,NS)  BEGIN  clk <= not clk after PERIOD/2; -- Generating clock, the period = 50ns.  if(CLR = '1') then PS <= A;  elsif (rising\_edge(clk)) then  PS <= NS;  end if;  end process sync\_proc;  comb\_proc: process(PS, INPUT)  begin  case PS is  when A => -- items regarding state ST0    if(INPUT = '0') then  NS <= B;  z1 <= '0'; -- Mealy output  z2 <= '0'; -- Mealy output  else  NS <= E;  z1 <= '0'; -- Mealy output  z2 <= '0'; -- Mealy output    end if;  when B => -- items regarding state ST0    if(INPUT = '0') then  NS <= A;  z1 <= '0'; -- Mealy output  z2 <= '0'; -- Mealy output  else  NS <= C;  z1 <= '0'; -- Mealy output  z2 <= '0'; -- Mealy output  end if;    when C => -- items regarding state ST0    if(INPUT = '0') then  NS <= D;  z1 <= '0'; -- Mealy output  z2 <= '0'; -- Mealy output  else  NS <= F;  z1 <= '0'; -- Mealy output  z2 <= '0'; -- Mealy output  end if;    when D => -- items regarding state ST0    if(INPUT = '0') then  NS <= A;  z1 <= '0'; -- Mealy output  z2 <= '0'; -- Mealy output  else  NS <= A;  z1 <= '1'; -- Mealy output  z2 <= '0'; -- Mealy output  end if;    when E => -- items regarding state ST0    if(INPUT = '0') then  NS <= A;  z1 <= '0'; -- Mealy output  z2 <= '0'; -- Mealy output  else  NS <= B;  z1 <= '0'; -- Mealy output  z2 <= '0'; -- Mealy output  end if;    when F => -- items regarding state ST0    if(INPUT = '0') then  NS <= A;  z1 <= '0'; -- Mealy output  z2 <= '1'; -- Mealy output  else  NS <= A;  z1 <= '0'; -- Mealy output  z2 <= '0'; -- Mealy output  end if;    when others => -- the catch-all condition  z1 <= '1'; -- arbitrary; it should never  z2 <= '1';  Ns <= A; -- make it to these two statement  end case;  end process comb\_proc;  end FSM; |

**The simulating picture:**

****