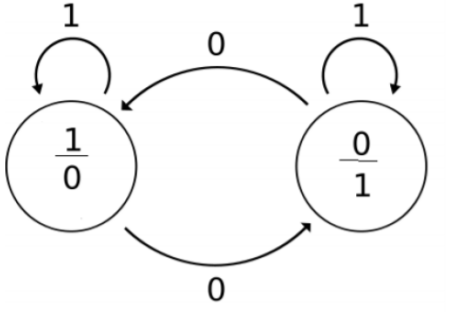
Lab 9 – VHDL

Name: Nhut Quang Tran

Group: NTIVIS14

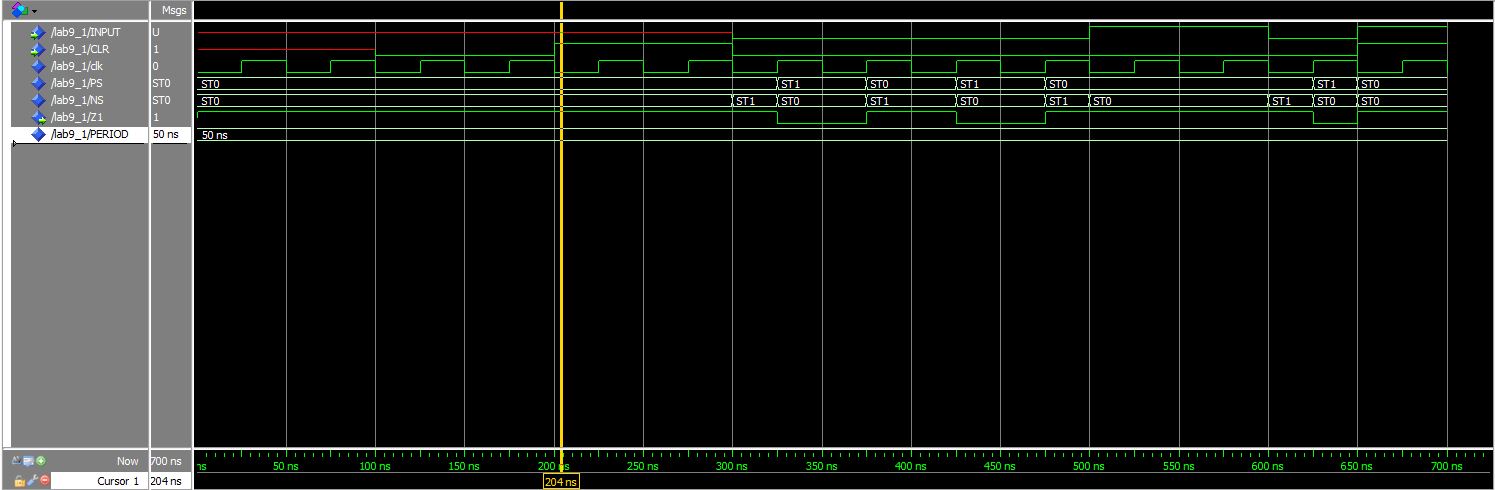
TURKU UNIVERSITY OF APPLIED SCIENCES

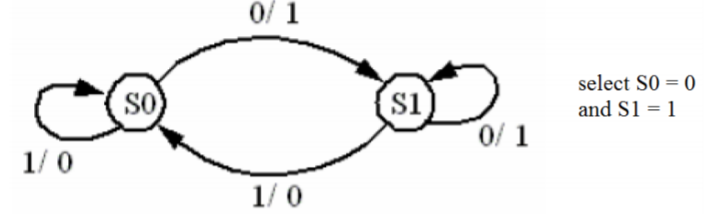
**Exercise 1:** Make a VHDL model for the state diagram given below.

**The code:**

|  |
| --- |
| LIBRARY ieee ;  USE ieee.std\_logic\_1164.all ;  use IEEE.NUMERIC\_STD.all;  use IEEE.std\_logic\_unsigned.all;  ENTITY Lab9\_1 IS  PORT (INPUT, CLR: IN STD\_LOGIC;  Z1,Y: OUT STD\_LOGIC);  END Lab9\_1;  ARCHITECTURE FSM OF Lab9\_1 IS  constant PERIOD: time := 50ns;  signal clk : std\_logic := '0';  type state\_type is (ST0,ST1);  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  Z1 <= '1'; -- Moore output  if(INPUT = '0') then NS <= ST1;  else NS <= ST0;  end if;  when ST1 => -- items regarding state ST0  Z1 <= '0'; -- Moore output  if(INPUT = '0') then NS <= ST0;  else NS <= ST1;  end if;    when others => -- the catch-all condition  Z1 <= '0'; -- arbitrary; it should never  Ns <= ST0; -- make it to these two statement  end case;  end process comb\_proc;  with PS select  Y <= '0' when ST0,  '1' when ST1,  '0' when others;  end FSM; |

**The simulating picture:**

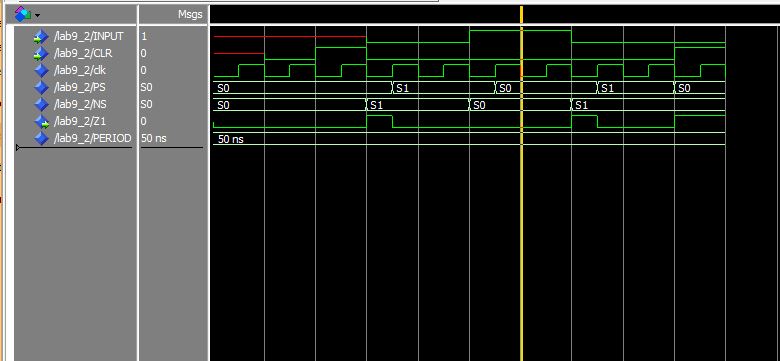
****

**Exercise 2:** Make a VHDL model for the state diagram given below.

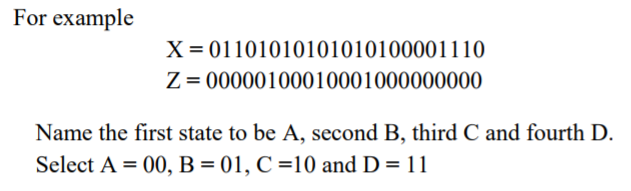
**The code:**

|  |
| --- |
| LIBRARY ieee ;  USE ieee.std\_logic\_1164.all ;  use IEEE.NUMERIC\_STD.all;  use IEEE.std\_logic\_unsigned.all;  ENTITY Lab9\_2 IS  PORT (INPUT, CLR: IN STD\_LOGIC;  Z1: OUT STD\_LOGIC);  END Lab9\_2;  ARCHITECTURE FSM OF Lab9\_2 IS  constant PERIOD: time := 50ns;  signal clk : std\_logic := '0';  type state\_type is (S0,S1);  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 <= S0;  elsif (rising\_edge(clk)) then  PS <= NS;  end if;  end process sync\_proc;  comb\_proc: process(PS, INPUT)  begin  case PS is  when S0 => -- items regarding state ST0    if(INPUT = '0') then  NS <= S1;  Z1 <= '1'; -- Mealy output  else  NS <= S0;  Z1 <= '0'; -- Mealy output    end if;  when S1 => -- items regarding state ST0    if(INPUT = '0') then  NS <= S1;  Z1 <= '0'; -- Mealy output  else  NS <= S0;  Z1 <= '0'; -- Mealy output  end if;    when others => -- the catch-all condition  Z1 <= '1'; -- arbitrary; it should never  Ns <= S0; -- make it to these two statement  end case;  end process comb\_proc;  end FSM; |

**The simulating picture:**



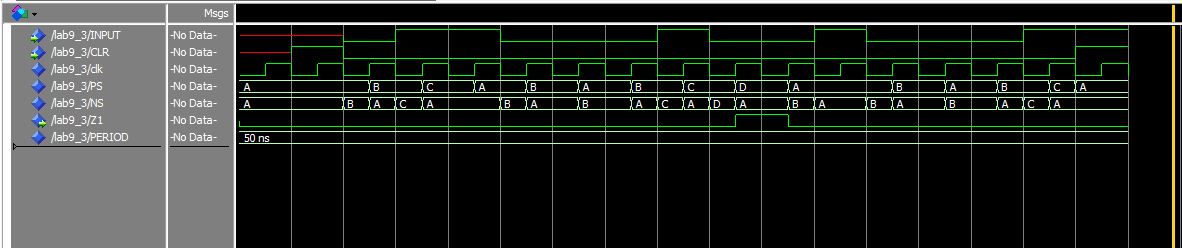
**3. Design with a circuit which detects the input sequence 010 by producing z = 1 as the last 0 occurs. No overlapping allowed. Moore machine only.**



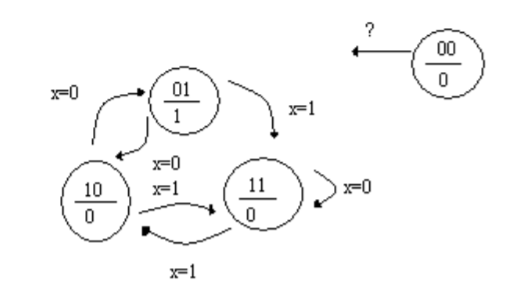
**The code:**

|  |
| --- |
| LIBRARY ieee ;  USE ieee.std\_logic\_1164.all ;  use IEEE.NUMERIC\_STD.all;  use IEEE.std\_logic\_unsigned.all;  ENTITY Lab9\_3 IS  PORT (INPUT, CLR: IN STD\_LOGIC;  Y: OUT STD\_LOGIC\_VECTOR (1 DOWNTO 0);  Z1: OUT STD\_LOGIC);  END Lab9\_3;  ARCHITECTURE FSM OF Lab9\_3 IS  constant PERIOD: time := 50ns;  signal clk : std\_logic := '0';  type state\_type is (A,B,C,D);  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 <= B;  else  NS <= A;  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 <= '0'; -- Moore output  if(INPUT = '0') then  NS <= D;  else  NS <= A;  end if;    when D => -- items regarding state ST0  Z1 <= '1'; -- Moore output  if(INPUT = '0') then  NS <= A;    else  NS <= D;  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;  with PS select  Y <= "00" when A,  "01" when B,  "10" when C,  "11" when D,  "00" when others;  end FSM; |

**The simulating picture:**



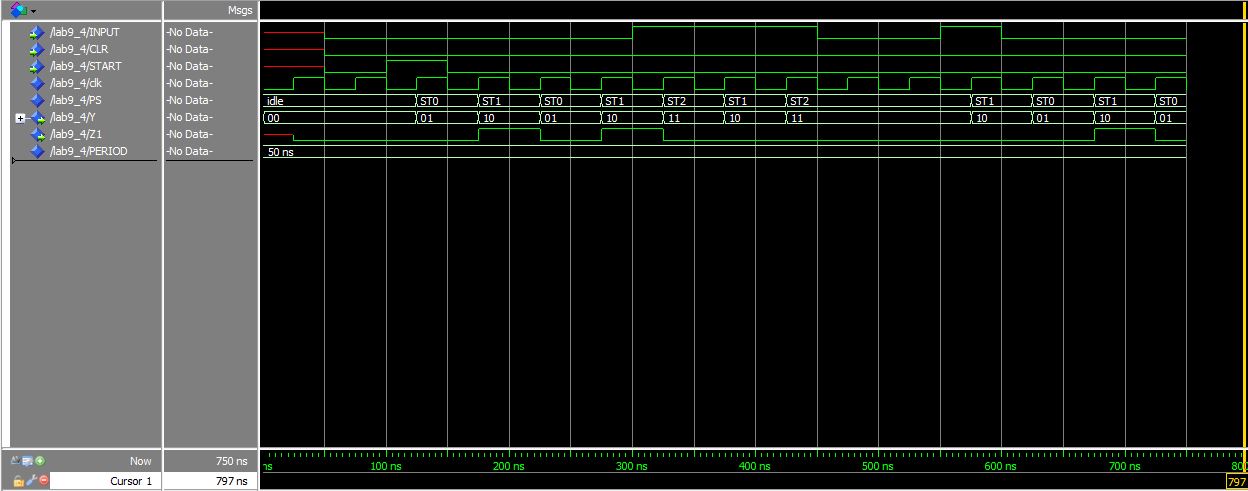
**4. Design a finite state machine which works according following state diagram.**

**Try to find a good solution what should happen with an unspecified state 00.**

**The code:**

|  |
| --- |
| LIBRARY ieee ;  USE ieee.std\_logic\_1164.all ;  use IEEE.NUMERIC\_STD.all;  use IEEE.std\_logic\_unsigned.all;  ENTITY Lab9\_3 IS  PORT (INPUT, CLR: IN STD\_LOGIC;  Y: OUT STD\_LOGIC\_VECTOR (1 DOWNTO 0);  Z1: OUT STD\_LOGIC);  END Lab9\_3;  ARCHITECTURE FSM OF Lab9\_3 IS  constant PERIOD: time := 50ns;  signal clk : std\_logic := '0';  type state\_type is (A,B,C,D);  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 <= B;  else  NS <= A;  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 <= '0'; -- Moore output  if(INPUT = '0') then  NS <= D;  else  NS <= A;  end if;    when D => -- items regarding state ST0  Z1 <= '1'; -- Moore output  if(INPUT = '0') then  NS <= A;    else  NS <= D;  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;  with PS select  Y <= "00" when A,  "01" when B,  "10" when C,  "11" when D,  "00" when others;  end FSM; |

**The simulating picture:**

****