Electronic Design Automation

Project 1

Presented to:

Prof. Mohamed Dessouky

By:

Muhammad Tamer Bassiony

muhammadbassiony@gmail.com

17P6044

CESS

Contents

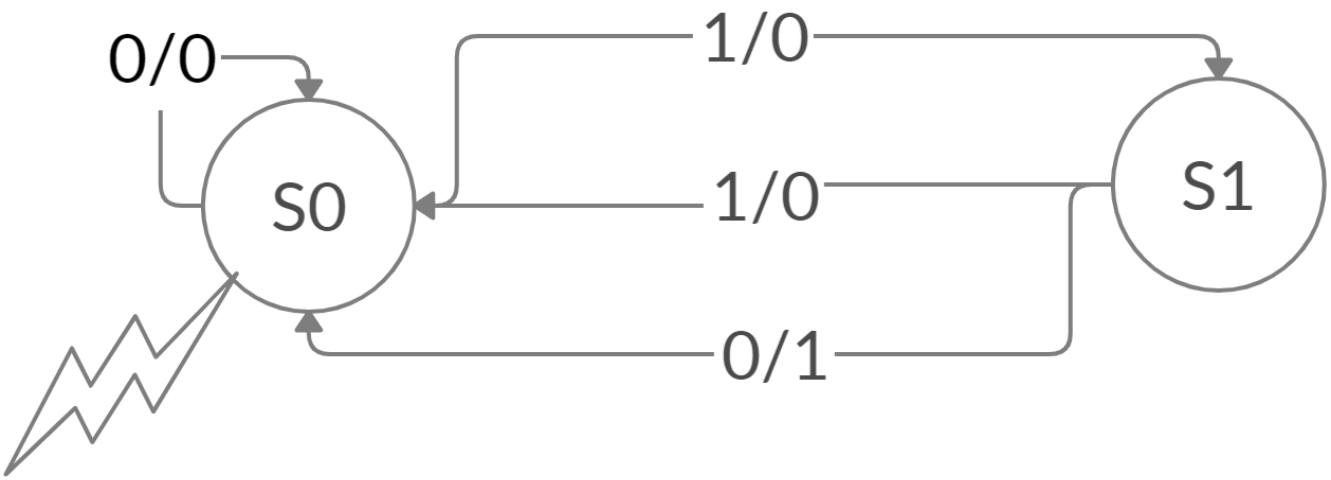
Introduction

This is the first milestone in the Electronic Design Automation class project. This document contains the implementation of a dual sequence detector as a finite state machine implemented in VHDL using ModelSim.

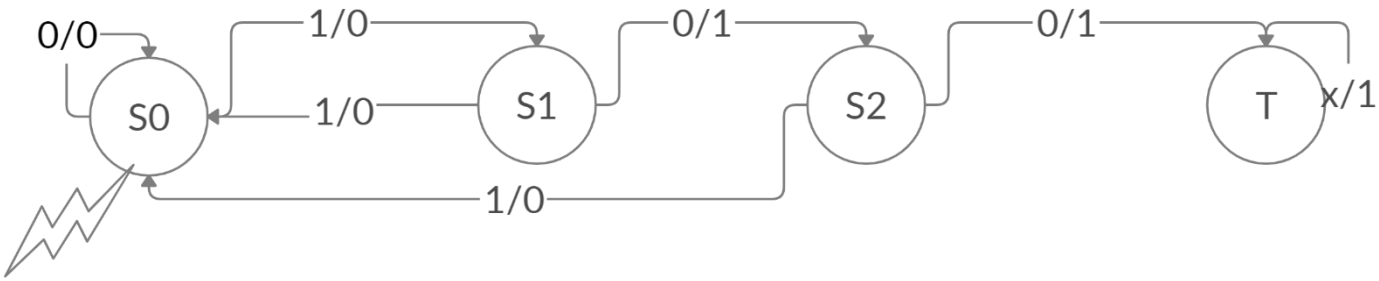
The program is to receive a sequence of bits and output a high when it receives the sequence 010, likewise, it should also output a high on the termination line when it receives the sequence 100. Both sequences are overlapping. When the program receives a reset bit it resets the termination bit and resumes.

State diagrams

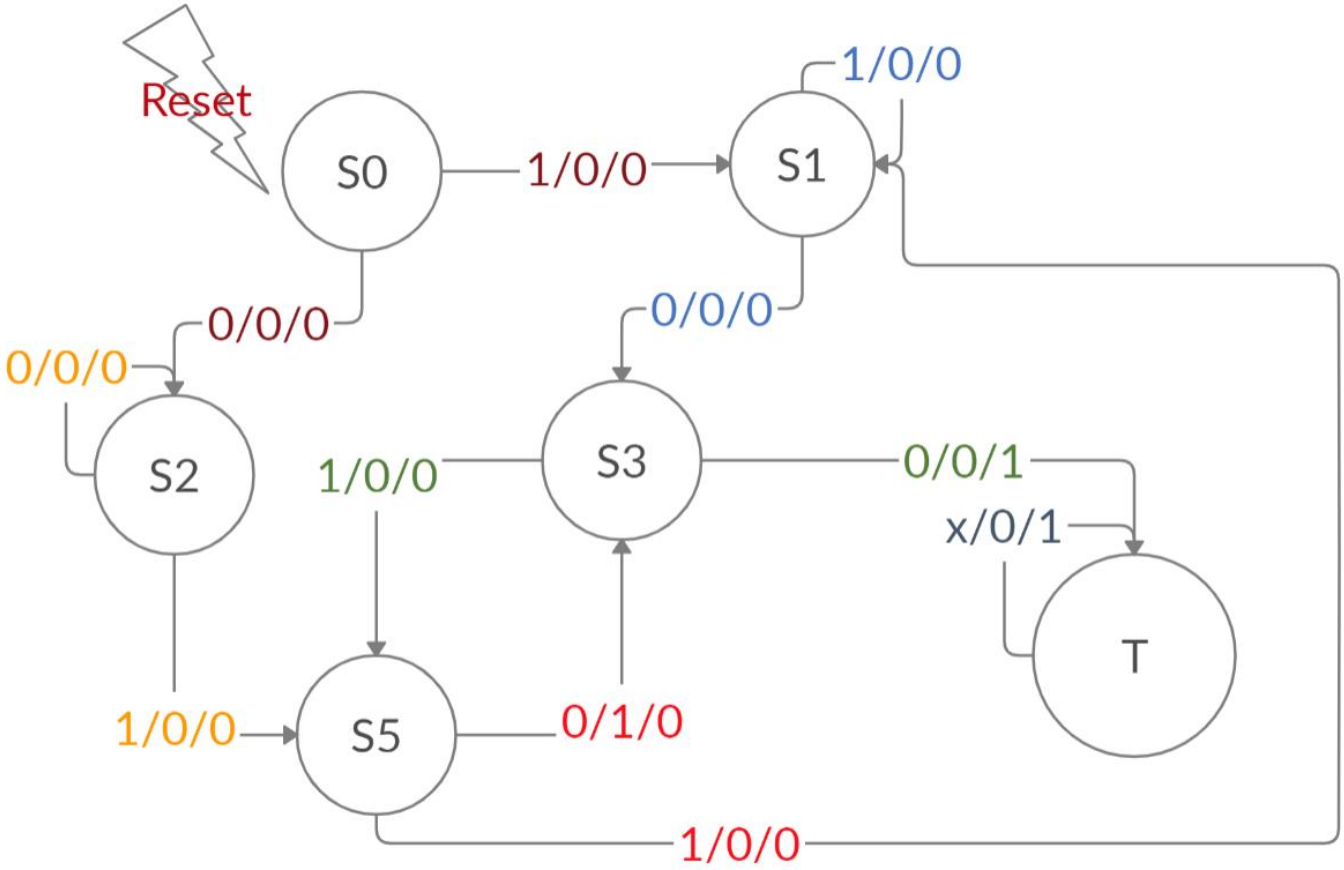
State diagram for the sequence 010



State diagram for the termination sequence 100



Combined state diagram



Simulation results

Test 1

Input: "11011010010"

Expected output: "00000001000"

Expected termination: "00000000111"

Testbench Code:

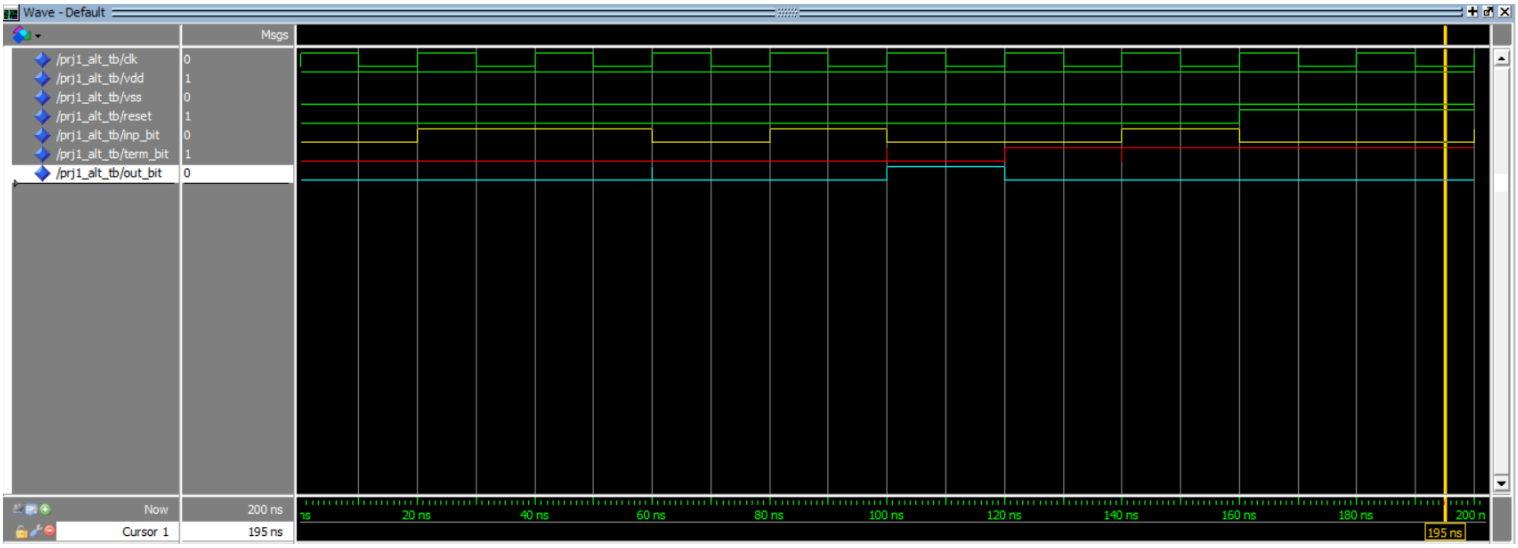
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use ieee.numeric_std.all;
use ieee.std_logic_arith.all;



ENTITY prj1_alt_tb IS
END ENTITY prj1_alt_tb;

ARCHITECTURE arc OF prj1_alt_tb IS

component seq_detector is
        port(
        input   :       in      bit;
        vdd     :       in      bit;
        vss     :       in      bit;
        clk     :       in      bit;
        reset   :       in      bit;
        output  :       out     bit;
        termin  :       out     bit
        );
end component seq_detector;

for dut : seq_detector use entity work.seq_detector(fsm);
```

```
-- signals declaration

        SIGNAL clk              : bit := '0';
        SIGNAL vdd              : bit := '1';
        SIGNAL vss              : bit := '0';
        SIGNAL reset            : bit := '0';

        signal  inp_bit  :      bit;
        signal  term_bit :      bit;
        signal  out_bit         :       bit;

        constant inputSig1      : BIT_VECTOR (10 downto 0) := "11011010010";
        constant outputSig1     : BIT_VECTOR (10 downto 0) := "00000001000";
        constant termSig1       : BIT_VECTOR (10 downto 0) := "00000000111";

        constant inputSig2      : BIT_VECTOR (10 downto 0) := "00101010010";
        constant outputSig2     : BIT_VECTOR (10 downto 0) := "00010101000";
        constant termSig2       : BIT_VECTOR (10 downto 0) := "00000000111";


        constant clkdur    : time := 20 ns;

-- beginning of architecture
begin

--dut instantiation
dut: seq_detector PORT MAP (inp_bit, vdd, vss, clk, reset, out_bit, term_bit);


clk_process : process
begin
clk <= '1';
wait for clkdur/2 ;
clk <= '0';
wait for clkdur/2 ;
```

```
end process;



stim: process is

variable termin_flag : boolean := false;

begin

        reset <= '1' after 8*clkdur;

        for i in (inputsig1'length-3) downto 0 loop

                assert false report "loop no.: "&integer'image(i) severity note;
                inp_bit <= inputsig1(i);
                wait for 20 ns;

                if (inputsig1(i) = '0') then

                        if(inputsig1(i+1) = '0' and inputsig1(i+2)='1') then

                                termin_flag := true;

                                assert term_bit='1' report "@@ @@TERM NOT EQUAL  "&integer'image(i) severity error;
                                --assert termsig1(i)='1' report " ### TERM NOT EQUAL  "&integer'image(i) severity error;

                                assert out_bit='0' report "@@ @@OUT NOT EQUAL  "&integer'image(i) severity error;
                                --assert outputsig1(i)='0' report " ### OUT NOT EQUAL  "&integer'image(i) severity error;


                        elsif(inputsig1(i+1) = '1' and inputsig1(i+2)='0' and termin_flag = false) then

                                assert out_bit='1' report "@@ @@OUT NOT EQUAL  "&integer'image(i) severity error;
```

```
                --assert outputsig1(i)='1' report " ### OUT NOT EQUAL  "&integer'image(i) severity error;

                end if;
            end if;
        end loop;

end process;

end architecture arc;
```

Simulation



Observation:

No errors or notes appeared in the log and program executes perfectly.

Test 2

Input: "00101010010"

Expected output: "00010101000"

Expected termination: "00000000111"

TestBench Code:

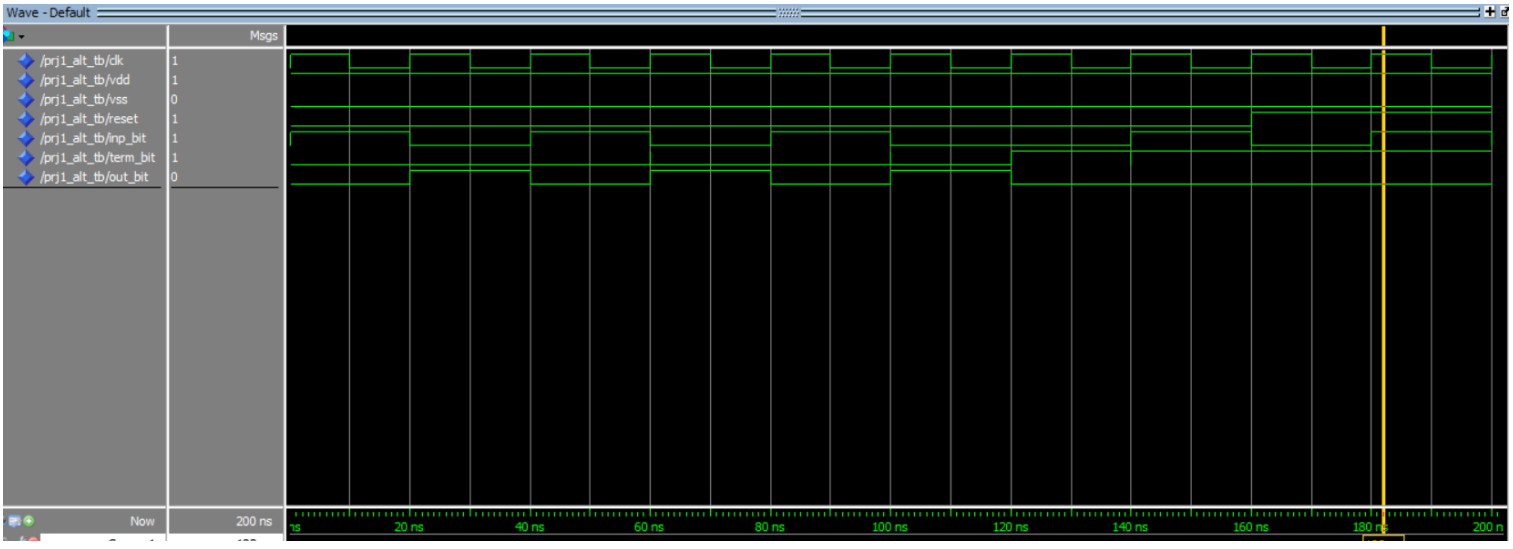
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use ieee.numeric_std.all;
use ieee.std_logic_arith.all;



ENTITY prj1_alt_tb IS
END ENTITY prj1_alt_tb;

ARCHITECTURE arc OF prj1_alt_tb IS

component seq_detector is
        port(
        input   :       in      bit;
        vdd     :       in      bit;
        vss     :       in      bit;
        clk     :       in      bit;
        reset   :       in      bit;
        output  :       out     bit;
        termin  :       out     bit
        );
end component seq_detector;

for dut : seq_detector use entity work.seq_detector(fsm);
```

```
-- signals declaration

        SIGNAL clk              : bit := '0';
        SIGNAL vdd              : bit := '1';
        SIGNAL vss              : bit := '0';
        SIGNAL reset            : bit := '0';

        signal  inp_bit  :      bit;
        signal  term_bit :      bit;
        signal  out_bit         :       bit;

        constant inputSig1      : BIT_VECTOR (10 downto 0) := "11011010010";
        constant outputSig1     : BIT_VECTOR (10 downto 0) := "00000001000";
        constant termSig1       : BIT_VECTOR (10 downto 0) := "00000000111";

        constant inputSig2      : BIT_VECTOR (10 downto 0) := "00101010010";
        constant outputSig2     : BIT_VECTOR (10 downto 0) := "00010101000";
        constant termSig2       : BIT_VECTOR (10 downto 0) := "00000000111";


        constant clkdur    : time := 20 ns;

-- beginning of architecture
begin

--dut instantiation
dut: seq_detector PORT MAP (inp_bit, vdd, vss, clk, reset, out_bit, term_bit);


clk_process : process
begin
clk <= '1';
wait for clkdur/2 ;
clk <= '0';
wait for clkdur/2 ;
```

```
end process;



stim: process is

variable termin_flag : boolean := false;

begin

        reset <= '1' after 8*clkdur;

        for i in (inputsig2'length-3) downto 0 loop

                assert false report "loop no.: "&integer'image(i) severity note;
                inp_bit <= inputsig2(i);
                wait for 20 ns;

                if (inputsig2(i) = '0') then

                        if(inputsig2(i+1) = '0' and inputsig2(i+2)='1') then

                                termin_flag := true;

                                assert term_bit='1' report "@@ @@TERM NOT EQUAL  "&integer'image(i) severity error;
                                assert termsig2(i)='1' report " ### TERM NOT EQUAL  "&integer'image(i) severity error;

                                assert out_bit='0' report "@@ @@OUT NOT EQUAL  "&integer'image(i) severity error;
                                assert outputsig2(i)='0' report " ### OUT NOT EQUAL  "&integer'image(i) severity error;


                        elsif(inputsig2(i+1) = '1' and inputsig2(i+2)='0' and termin_flag = false) then

                                assert out_bit='1' report "@@ @@OUT NOT EQUAL  "&integer'image(i) severity error;
```

```
                    assert outputsig2(i)='1' report " ### OUT NOT EQUAL  "&integer'image(i) severity error;

                end if;

            end if;

        end loop;

end process;

end architecture arc;
```

Simulation:



Observation:

The program does not display any errors or notes and executes perfectly.

Test 3

Input: "00101010010"

Expected output: "00010101000"

Expected termination: "00000000111"

Strategy:

Reset the program at the 5th iteration

TestBench Code:

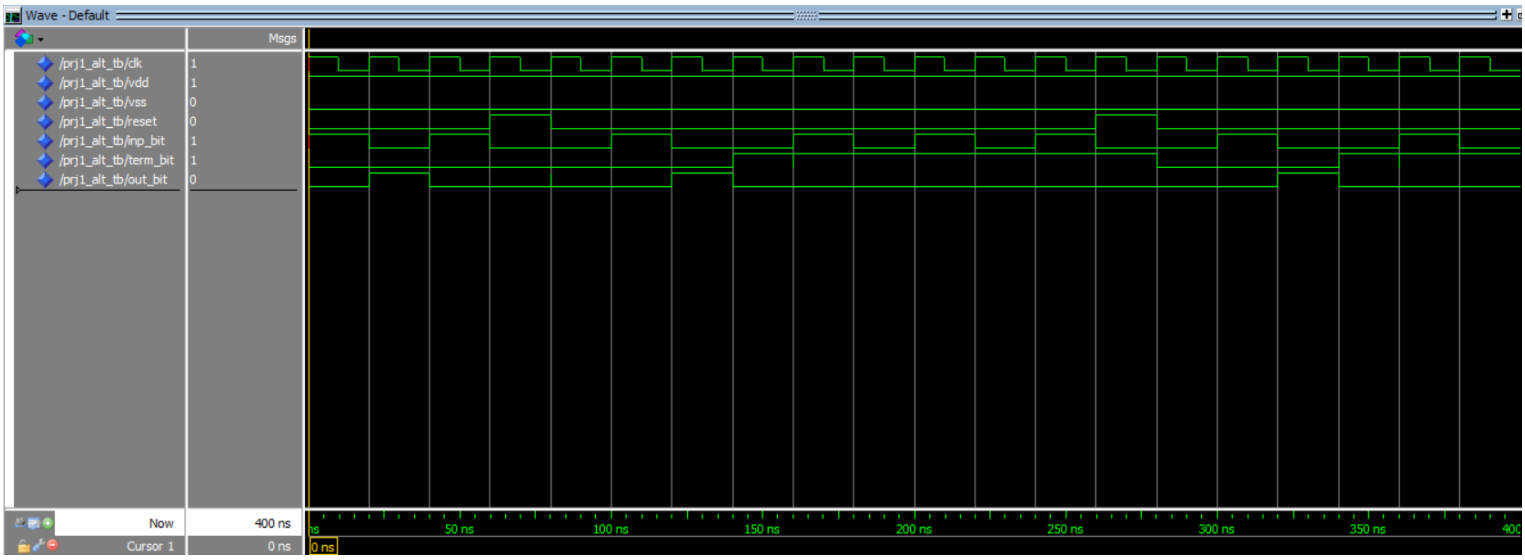
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use ieee.numeric_std.all;
use ieee.std_logic_arith.all;



ENTITY prj1_alt_tb IS
END ENTITY prj1_alt_tb;

ARCHITECTURE arc OF prj1_alt_tb IS

component seq_detector is
        port(
        input   :       in      bit;
        vdd     :       in      bit;
        vss     :       in      bit;
        clk     :       in      bit;
        reset   :       in      bit;
        output  :       out     bit;
        termin  :       out     bit
        );
end component seq_detector;

for dut : seq_detector use entity work.seq_detector(fsm);
```

```
-- signals declaration

        SIGNAL clk              : bit := '0';
        SIGNAL vdd              : bit := '1';
        SIGNAL vss              : bit := '0';
        SIGNAL reset            : bit := '0';

        signal  inp_bit  :      bit;
        signal  term_bit :      bit;
        signal  out_bit         :       bit;

        constant inputSig1      : BIT_VECTOR (10 downto 0) := "11011010010";
        constant outputSig1     : BIT_VECTOR (10 downto 0) := "00000001000";
        constant termSig1       : BIT_VECTOR (10 downto 0) := "00000000111";

        constant inputSig2      : BIT_VECTOR (10 downto 0) := "00101010010";
        constant outputSig2     : BIT_VECTOR (10 downto 0) := "00010101000";
        constant termSig2       : BIT_VECTOR (10 downto 0) := "00000000111";


        constant clkdur    : time := 20 ns;

-- beginning of architecture
begin

--dut instantiation
dut: seq_detector PORT MAP (inp_bit, vdd, vss, clk, reset, out_bit, term_bit);


clk_process : process
begin
clk <= '1';
wait for clkdur/2 ;
clk <= '0';
wait for clkdur/2 ;
```

```
end process;



stim: process is

variable termin_flag : boolean := false;

begin

        reset <= '1' after 8*clkdur;

        for i in (inputsig2'length-3) downto 0 loop

                assert false report "loop no.: "&integer'image(i) severity note;
                inp_bit <= inputsig2(i);

                if(i = 5) then
                        reset <= '1';
                        wait for 20 ns;
                        reset <= '0';
                end if;

                wait for 20 ns;


                if (inputsig2(i) = '0') then

                        if(inputsig2(i+1) = '0' and inputsig2(i+2)='1') then

                                termin_flag := true;

                                assert term_bit='1' report "@@ @@TERM NOT EQUAL  "&integer'image(i) severity error;
                                assert termsig2(i)='1' report " ### TERM NOT EQUAL  "&integer'image(i) severity error;
```

```
                    assert out_bit='0' report "@@ @@OUT NOT EQUAL  "&integer'image(i) severity error;
                    assert outputsig2(i)='0' report " ### OUT NOT EQUAL  "&integer'image(i) severity error;


                elsif(inputsig2(i+1) = '1' and inputsig2(i+2)='0' and termin_flag = false) then

                    assert out_bit='1' report "@@ @@OUT NOT EQUAL  "&integer'image(i) severity error;
                    assert outputsig2(i)='1' report " ### OUT NOT EQUAL  "&integer'image(i) severity error;

                end if;



            end if;



        end loop;

end process;

end architecture arc;
```

Simulation:



Observation:

The program resets and returns to state S0

Appendix

VHDL code

This Finite state machine is MEALY and NOT Moore as it depends on both the current state and the inputs.

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

entity seq_detector is
        port(
        input   :       in      bit;
        vdd     :       in      bit;
        vss     :       in      bit;
        clk     :       in      bit;
        reset   :       in      bit;
        output  :       out     bit;
        termin  :       out     bit
        );

end seq_detector;

architecture fsm of seq_detector is

  type STATE_TYPE is (S0, S1, S2, S3, S5, T);
  signal NS, CS : STATE_TYPE;

begin

process (input, cs, reset, clk)
  begin
        if (reset = '1') then
          --resetting to S0
          ns <= S0;
```

```
else


case cs is
when S0 =>
        if (input = '1') then
                ns <= S1;
                output <= '0';
                termin <= '0';
else
                ns <= S2;
                output <= '0';
                termin <= '0';
        end if;
when S1 =>
        if (input = '1') then
                ns <= S1;
                output <= '0';
                termin <= '0';
else
                ns <= S3;
                output <= '0';
                termin <= '0';
        end if;
when S2 =>
        if (input = '1') then
                ns <= S5;
                output <= '0';
                termin <= '0';
else
                ns <= S2;
                output <= '0';
                termin <= '0';
        end if;
when S3 =>
        if (input = '1') then
```

```
                ns <= S5;
                output <= '0';
                termin <= '0';
        else
                ns <= T;
                output <= '0';
                termin <= '1';
            end if;
        when S5 =>
            if (input = '1') then
                ns <= S1;
                output <= '0';
                termin <= '0';
        else
                ns <= S3;
                output <= '1';
                termin <= '0';
            end if;
        when T =>
                ns <= T;
                output <= '0';
                termin <= '1';



        end case;
        end if;
end process;



process(clk)

begin
        if(clk = '1' AND NOT clk'stable)then
             cs <= ns;
        end if;
```

```
end process;


end fsm;
```

Complete Testbench code

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



ENTITY prj1_alt_tb IS
END ENTITY prj1_alt_tb;

ARCHITECTURE arc OF prj1_alt_tb IS

component seq_detector is
        port(
        input   :       in      bit;
        vdd     :       in      bit;
        vss     :       in      bit;
        clk     :       in      bit;
        reset   :       in      bit;
        output  :       out     bit;
        termin  :       out     bit
        );
end component seq_detector;

for dut : seq_detector use entity work.seq_detector(fsm);

-- signals declaration

        SIGNAL clk              : bit := '0';
        SIGNAL vdd              : bit := '1';
        SIGNAL vss              : bit := '0';
        SIGNAL reset            : bit := '0';

        signal  inp_bit :       bit;
```

```
        signal    term_bit :        bit;
        signal    out_bit          :        bit;

        constant inputSig1        : BIT_VECTOR (10 downto 0) := "11011010010";
        constant outputSig1       : BIT_VECTOR (10 downto 0) := "00000001000";
        constant termSig1         : BIT_VECTOR (10 downto 0) := "00000000111";

        constant inputSig2        : BIT_VECTOR (10 downto 0) := "00101010010";
        constant outputSig2       : BIT_VECTOR (10 downto 0) := "00010101000";
        constant termSig2         : BIT_VECTOR (10 downto 0) := "00000000111";


        constant clkdur    : time := 20 ns;

-- beginning of architecture
begin

--dut instantiation
dut: seq_detector PORT MAP (inp_bit, vdd, vss, clk, reset, out_bit, term_bit);


clk_process : process
begin
clk <= '1';
wait for clkdur/2 ;
clk <= '0';
wait for clkdur/2 ;
end process;



stim: process is

variable termin_flag : boolean := false;
```

```
begin

        reset <= '1' after 8*clkdur;

        for i in (inputsig2'length-3) downto 0 loop

                assert false report "loop no.: "&integer'image(i) severity note;
                inp_bit <= inputsig2(i);

                if(i = 5) then
                        reset <= '1';
                        wait for 20 ns;
                        reset <= '0';
                end if;

                wait for 20 ns;


                if (inputsig2(i) = '0') then

                        if(inputsig2(i+1) = '0' and inputsig2(i+2)='1') then

                                termin_flag := true;

                                assert term_bit='1' report "@@ @@TERM NOT EQUAL  "&integer'image(i) severity error;
                                assert termsig2(i)='1' report " ### TERM NOT EQUAL  "&integer'image(i) severity error;

                                assert out_bit='0' report "@@ @@OUT NOT EQUAL  "&integer'image(i) severity error;
                                assert outputsig2(i)='0' report " ### OUT NOT EQUAL  "&integer'image(i) severity error;


                        elsif(inputsig2(i+1) = '1' and inputsig2(i+2)='0' and termin_flag = false) then

                                assert out_bit='1' report "@@ @@OUT NOT EQUAL  "&integer'image(i) severity error;
```

```
                    assert outputsig2(i)='1' report " ### OUT NOT EQUAL  "&integer'image(i) severity error;

                end if;



            end if;



        end loop;

end process;

end architecture arc;
```

References

- Lecture Notes
- Various Stack Overflow entries