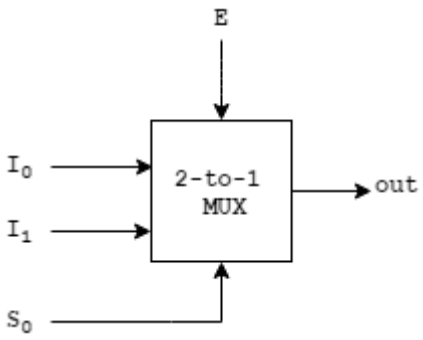
XILINX ISE SIMULATION TUTORIAL

Aim of this tutorial is to show how to simulate your ISE projects in VHDL. ISim Simulator is used to achieve simulation goals. As an example, 2-to-1 multiplexer with enable feature will be used. ISE 14.2 will be used during tutorial. I suppose a project in ISE 14.2 is opened and VHDL code of 2-to-1 multiplexer is already written.

1-) Block diagram and truth table of 2-to-1 multiplexer is given below in **Figure-1**.



E	S0	F
0	0	Z
0	1	Z
1	0	I0
1	1	I1

Figure-1

2-to-1 multiplexer has 4 inputs, two data inputs, one select bit and an enable bit, and one output. Multiplexer works if enable bit is logic-1 otherwise output will be high-Z. According to select bit, one of the data bits can be seen on the output, F.

2-) VHDL project is created with name "TWO_TO_ONE_MUX". In the below **Figure-2**, VHDL code can be seen.

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

entity TWO_TO_ONE_MUX is
    Port ( I0,I1 : in  STD_LOGIC;
           S0,E  : in  STD_LOGIC;
           F : out  STD_LOGIC
         );
end TWO_TO_ONE_MUX;

architecture Behavioral of TWO_TO_ONE_MUX is

begin

       F <= I0 WHEN (S0='0' AND E='1') ELSE
            I1 WHEN (S0='1' AND E='1') ELSE
            'Z';

end Behavioral;
```

Figure-2

3-) We completed the implementation of our multiplexer. Now we need to open a **"VHDL Test Bench"** file to perform simulation. Right click the FPGA model on the **"Design"** window of ISE. Then left click for **"New Source"**, **Figure-3**.

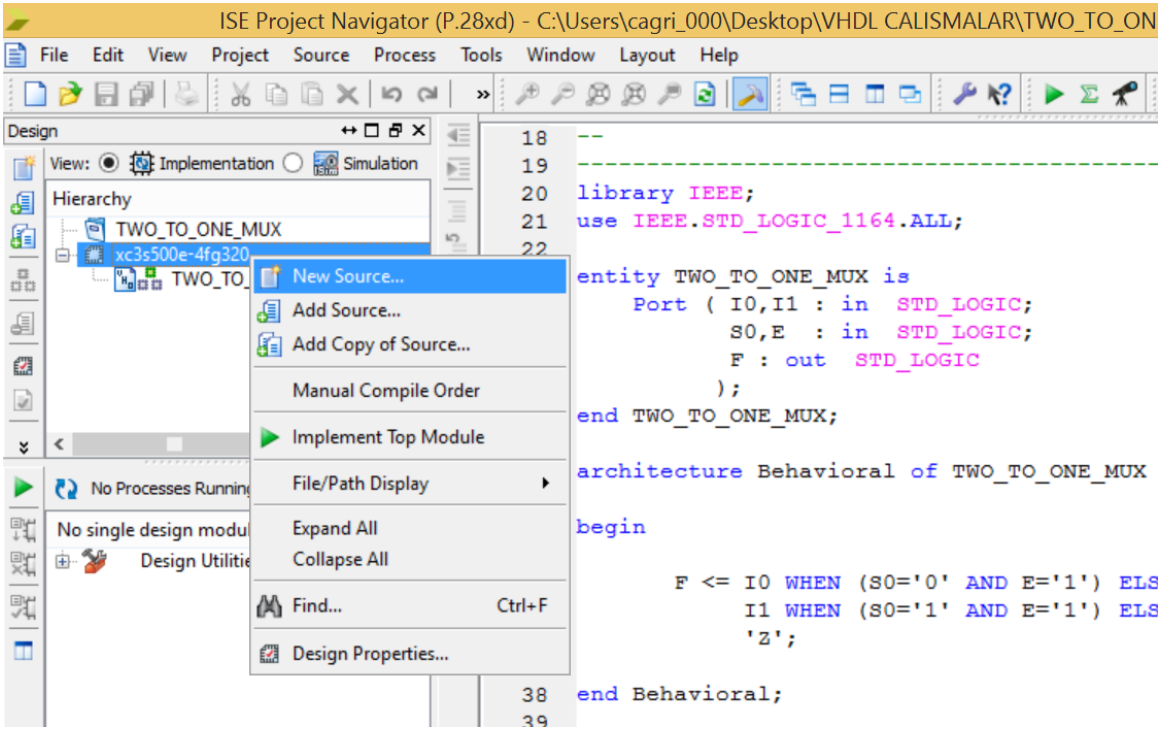


Figure-3

4-) From the **"New Source Wizard"** window choose **"VHDL Test Bench"** and give some name to the file as seen **Figure-4** and then click **"Next".**

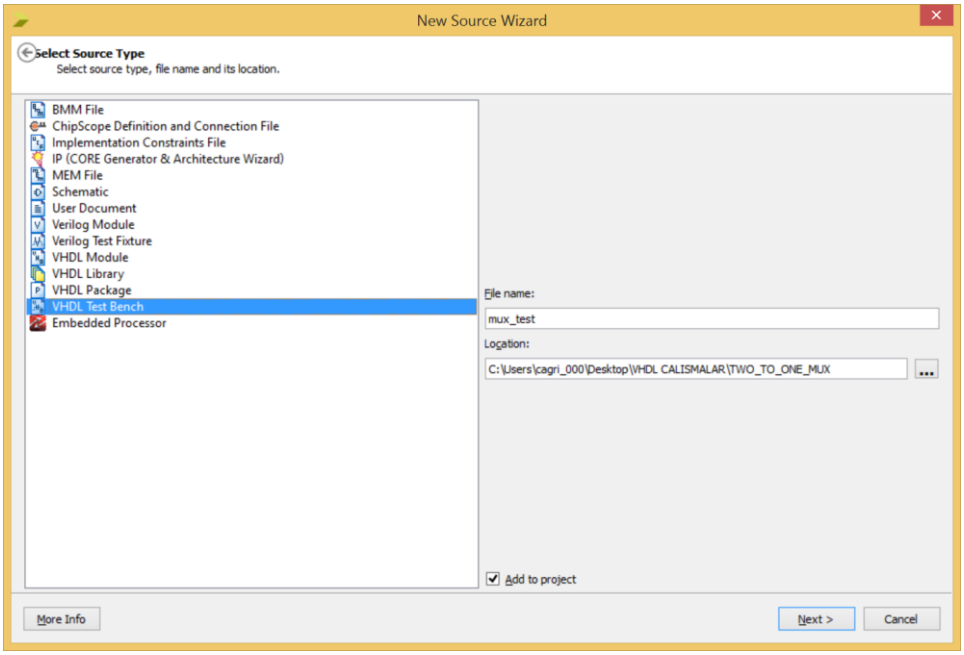


Figure-4

5-) In the next window it shows which VHDL document will be related with your simulation. In our case, "TWO_TO_ONE_MUX" is selected already. Simply click **"Next"** in the following **Figure-5** without changing anything.

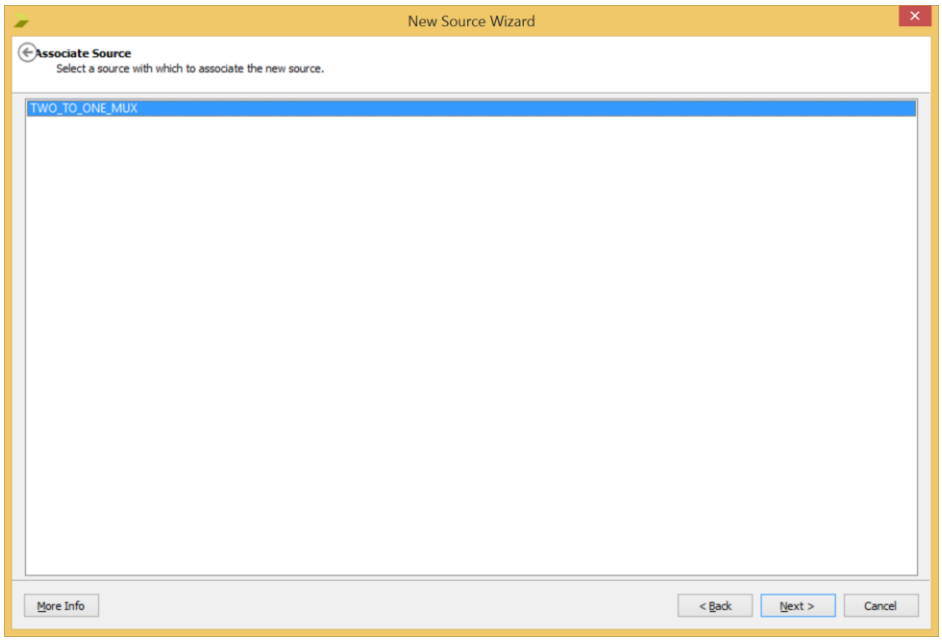


Figure-5

6-) Click **"Finish"** in the next window**, Figure-6**.

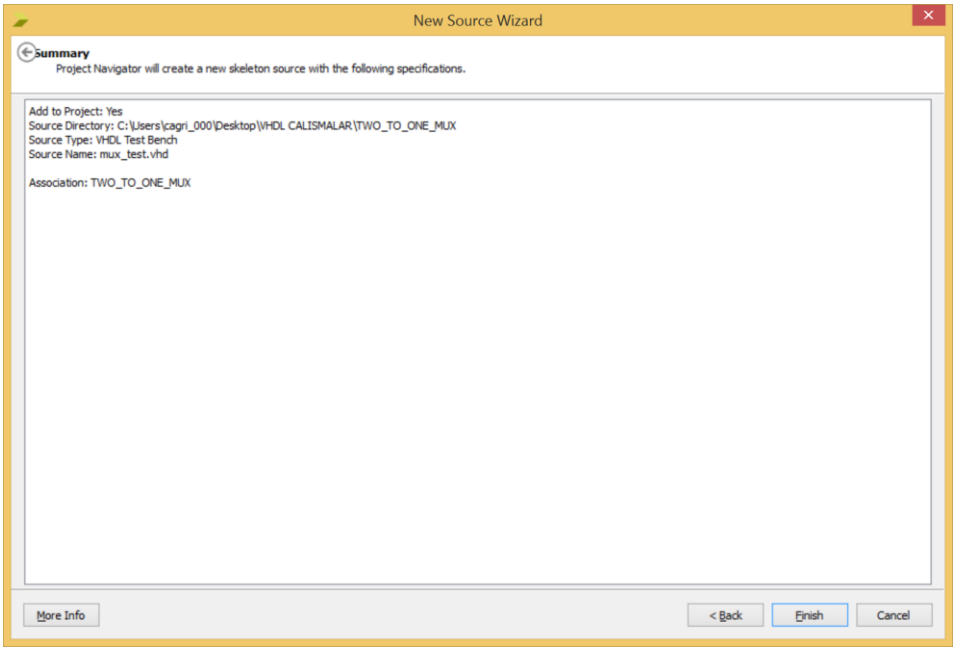


Figure-6

7-) "mux_test.vhd" is opened. Next step is to modify this code according to our scenario. In the below **Figure-7** default test bench code is given. In this example, there is no clock signal. Delete or make comment the shown lines in order to achieve better simulation results.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY mux_test IS
END mux_test;

ARCHITECTURE behavior OF mux_test IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT TWO_TO_ONE_MUX
    PORT(
         I0 : IN  std_logic;
         I1 : IN  std_logic;
         S0 : IN  std_logic;
         E : IN  std_logic;
         F : OUT  std_logic
        );
    END COMPONENT;

   --Inputs
   signal I0 : std_logic := '0';
   signal I1 : std_logic := '0';
   signal S0 : std_logic := '0';
   signal E : std_logic := '0';


    --Outputs
   signal F : std_logic;
   -- No clocks detected in port list. Replace <clock> below with
   -- appropriate port name

   constant <clock>_period : time := 10 ns;

BEGIN

    -- Instantiate the Unit Under Test (UUT)
   uut: TWO_TO_ONE_MUX PORT MAP (
          I0 => I0,
          I1 => I1,
          S0 => S0,
          E => E,
          F => F
        );

   -- Clock process definitions
   <clock>_process :process
   begin
        <clock> <= '0';
        wait for <clock>_period/2;
        <clock> <= '1';
        wait for <clock>_period/2;
   end process;

   -- Stimulus process
   stim_proc: process
```

```
   begin
      -- hold reset state for 100 ns.
      wait for 100 ns;
      wait for <clock>_period*10;
      -- insert stimulus here
      wait;
   end process;

END;
```

Figure-7

8-) "mux_test.vhd" is modified. Simulation scenario covers all the possible outcomes, of the truth table of 2-to-1 multiplexer with enable feature implementation, **Figure-8**. 100 ns delays are added at each time a change has done in between inputs, for better observation. It should be noticed that simulation steps are covered in the "stim_proc" process.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY mux_test IS
END mux_test;

ARCHITECTURE behavior OF mux_test IS
    -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT TWO_TO_ONE_MUX
    PORT(
         I0 : IN  std_logic;
         I1 : IN  std_logic;
         S0 : IN  std_logic;
         E : IN  std_logic;
         F : OUT  std_logic
        );
    END COMPONENT;
   --Inputs
   signal I0 : std_logic := '0';
   signal I1 : std_logic := '0';
   signal S0 : std_logic := '0';
   signal E : std_logic := '0';
    --Outputs
   signal F : std_logic;

BEGIN
    -- Instantiate the Unit Under Test (UUT)
   uut: TWO_TO_ONE_MUX PORT MAP (
          I0 => I0,
          I1 => I1,
          S0 => S0,
          E => E,
          F => F
        );
   -- Stimulus process
   stim_proc: process
   begin
        I0 <='1'; I1 <='0';
        S0 <='0'; E <= '0';
      wait for 100 ns;
        S0 <='1'; E <= '0';
      wait for 100 ns;
```

```
        S0 <='0'; E <= '1';
      wait for 100 ns;
        S0 <='1'; E <= '1';
      wait for 100 ns;
   end process;

END;
```

Figure-8

9-) Simulation code is ready. Now we need to check syntax of the simulation code. First choose **"Simulation"** from the design menu as shown in **Figure-9**.

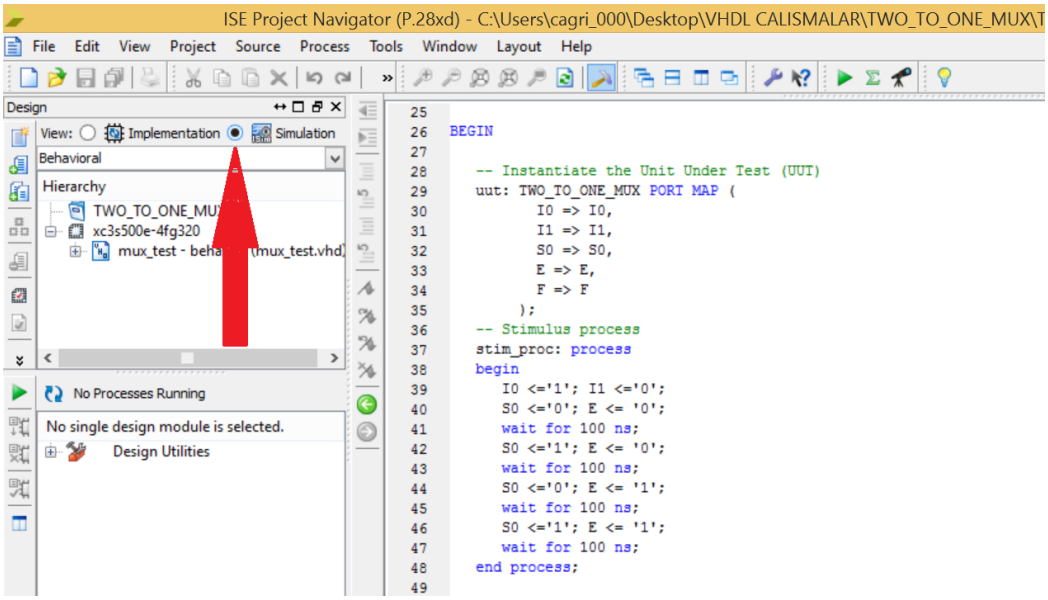


Figure-9

10-) Double click **"Behavioral Check Syntax"** from the Processors menu, **Figure-10.**

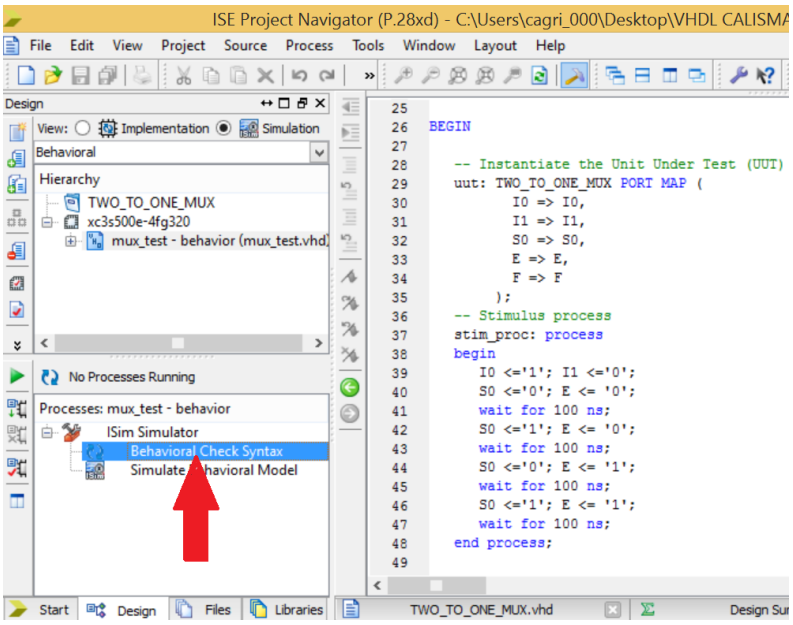


Figure-10

11-) "Behavioral Check Syntax" is completed. We have no error. Now we can start the simulation. Double click **"Simulate Behavioral Model"** from the **Processes** menu under the **"ISim Simulator", Figure-11.**

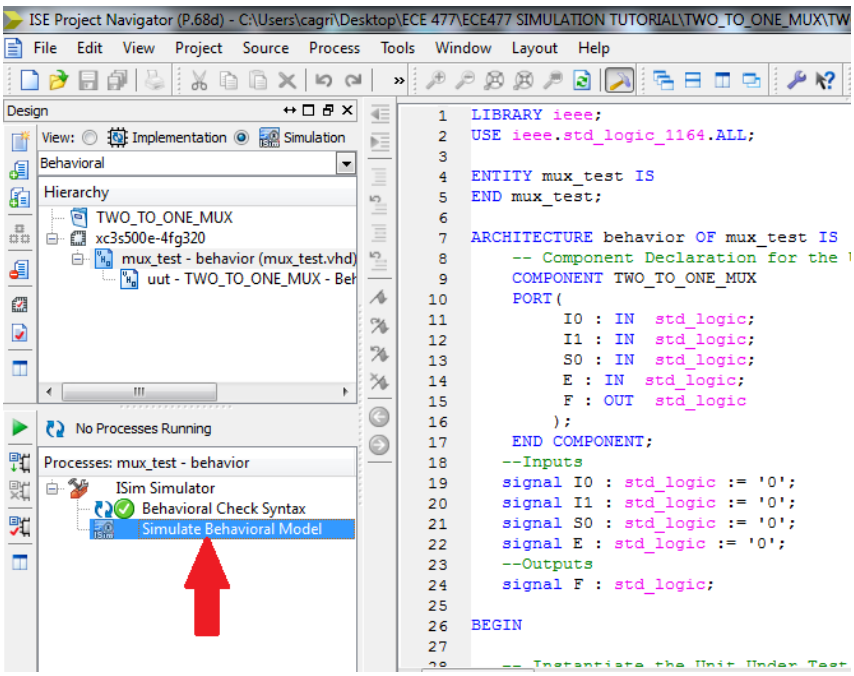


Figure-11

12-) Simulation window is opened. In our case, if we run our simulation for 1 us, it will be sufficient to observe all situations. Click the **"run for the time specified on the tool bar"** button as shown in **Figure-12**. It is also noticed that colors of the waveform window is different than the default one. If you want to change the colors follow the path on ISim Simulator window **"Edit" -> "Preferences"** and click **"New"** for the new color scheme. Then adjust the colors according to your desire.

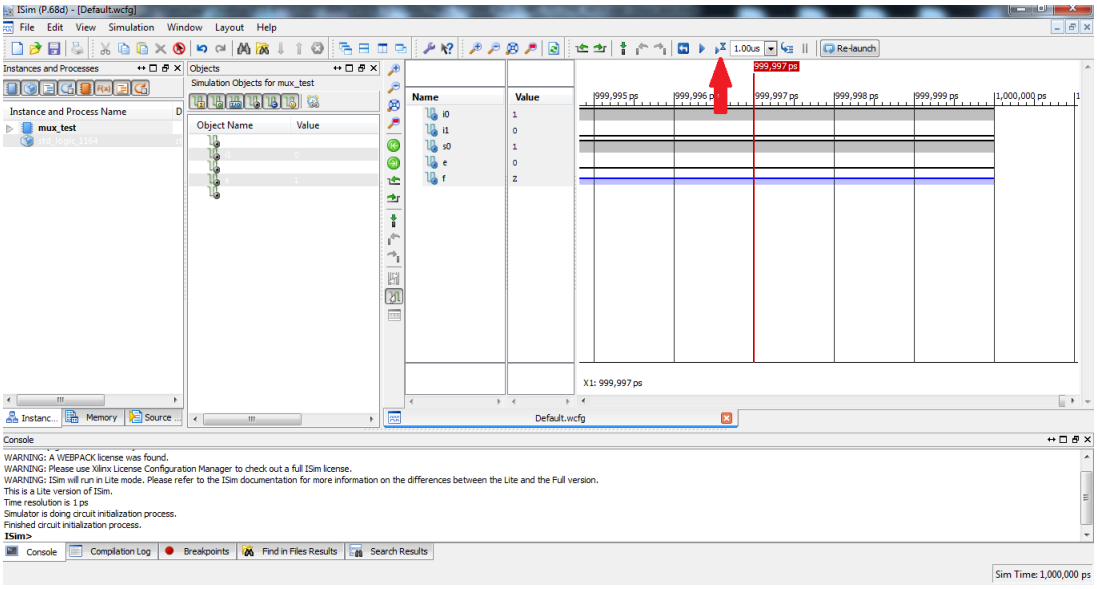


Figure-12

13-) Figure-13 shows the result of our simulation. If you follow the waveforms, you can see that we cover all the situations. It is important to say that red vertical line is cursor and the blue waveform components are the high-z values.

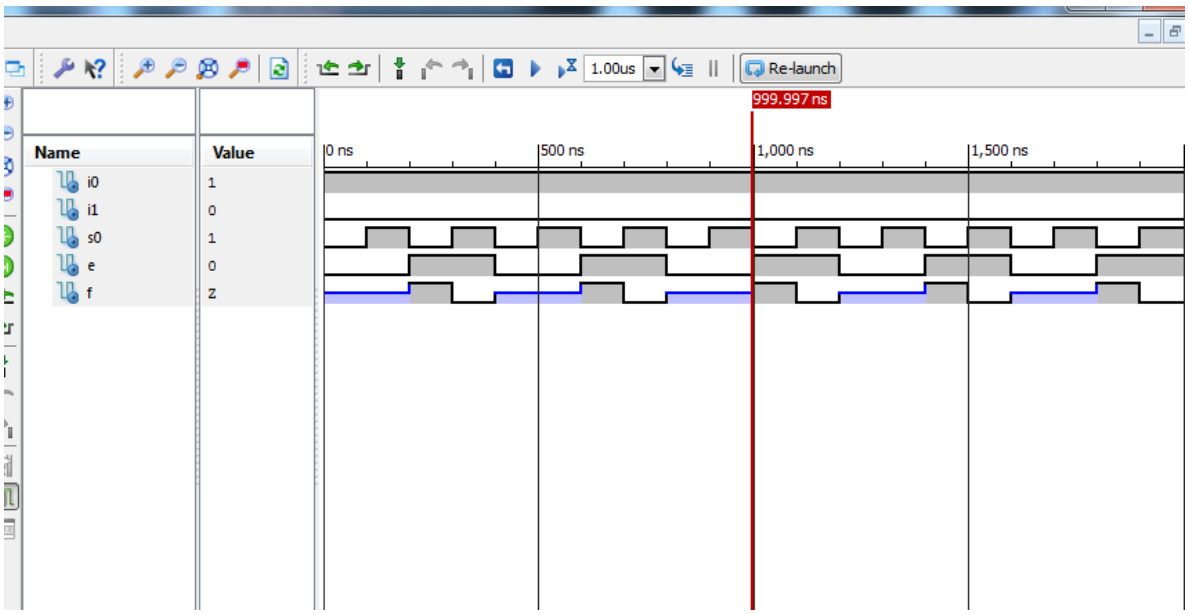


Figure-13

Prepared By

A.Çağrı Arlı