NEAR EAST UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

COM-473
HARDWARE DESIGN
PROJECT

/w Load, CE And Async Active High Reset

ÖMÜR KUZUCU
20031116

1) Create a Countupd Project:

-Select File -> New Project
-Type countupd in the project name field.
-Enter to location and a countupd subdirectory is created automatically.
-Verify that HDL is selected from top-level source type list.
-Click **Next** to move to the device properties page.
- Fill in the properties in the table as shown below:

- Product Category: **All**
- Family: **Spartan3**
- Device: **XC3S200**
- Package: **FT256**
- Speed Grade: **-4**
- Top-Level Source Type: **HDL**
- Synthesis Tool: **XST (VHDL/Verilog)**
- Simulator: **ISE Simulator (VHDL/Verilog)**
- Preferred Language: **Verilog** (or **VHDL**)
- Verify that **Enable Enhanced Design Summary** is selected.

-Click **Next** to proceed to the Create New Source window in the New Project Wizard. At the end of the next section, your new project will be complete.

2) Create the HDL Source of the Countupd:

-Click the new source button in the new project wizard.
-Select VHDL module as the source type.
-Type in the file name **c**ountupd.
-Verify that the add to project checkbox is selected.
-Declare the ports for the countupd design by filling in the port information as shown below:

Port Name	Direction	Bus	MSB	LSB
clock	in			
reset	in			
clocken	in			
loaden	in			
direction	in			
inp	in		3	0
count	out		3	0

3) HDL Language Template:

-Edit -> Language Templates
-VHDL -> Synthesis Constructs -> Coding Examples -> Binary ->
Up/Down Counters -> /w Load, CE and Async Active High Reset

```
process (<clock>, <reset>)
begin
        if <reset>='1' then
          <count> <= (others => '0');
        elsif <clock>='1' and <clock>'event then
                if <clock_enable>='1' then
                        if <load_enable>='1' then
                                <count> <= <input>;
                        else
                                if <count_direction>='1' then
                                 <count> <= <count> + 1;
                                else
                                 <count> <= <count> - 1;
                                 end if;
                         end if;
                end if;
         end if;
end process;
```

-With -> /w Load, CE and Async Active High Reset selected, select **Edit › Use in File**, or select the **Use Template in File** toolbar button. This step copies the template into the countupd source file.

4) Edit the HDL Code:

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

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity countupd is
 Port ( clock : in  STD_LOGIC;
        reset : in  STD_LOGIC;
        clocken : in  STD_LOGIC;
        loaden : in  STD_LOGIC;
        direction : in  STD_LOGIC;
        inp : in  STD_LOGIC_VECTOR (3 downto 0);
        count : out  STD_LOGIC_VECTOR (3 downto 0));
end countupd;

architecture Behavioral of countupd is
signal input_int : std_logic_vector(3 downto 0) := "0000";
```

```
signal count_int : std_logic_vector(3 downto 0) := "0000";
begin

-- Usage of Asynchronous resets may negatively impact FPGA resources
-- and timing. In general faster and smaller FPGA designs will
-- result from not using Asynchronous Resets. Please refer to
-- the Synthesis and Simulation Design Guide for more information.
process (clock, reset)
begin
if reset='1' then
    count_int <= (others => '0');
  elsif clock='1' and clock'event then
    if clocken='1' then
      if loaden='1' then
        count_int <= input_int;
      else
        if direction='1' then
          count_int <= count_int + 1;
        else
          count_int <= count_int - 1;
       end if;
      end if;
    end if;
  end if;
end process;
count <= count_int;

end Behavioral;
```

5) Syntax Check:

-Verify that Synthesis/Implementation is selected from the drop-drown list in the sources window.
-Click the '+' next to the Synthesize-XST process to expand the process group.
-Double-click the Check Syntax process.

6) create test banch

Project → add new source → test banch
Edit → language temples → VHDL → Clock simulations
And arrange like below.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;

ENTITY countupd_tb_vhd IS
END countupd_tb_vhd;
```

```
ARCHITECTURE behavior OF countupd_tb_vhd IS

    -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT countupd
    PORT(
        clock : IN std_logic;
        reset : IN std_logic;
        clocken : IN std_logic;
        loaden : IN std_logic;
        direction : IN std_logic;
        inp : IN std_logic_vector(3 downto 0);
        count : OUT std_logic_vector(3 downto 0)
        );
    END COMPONENT;

    --Inputs
    SIGNAL clock :  std_logic := '0';
    SIGNAL reset :  std_logic := '1';
    SIGNAL clocken :  std_logic := '0';
    SIGNAL loaden :  std_logic := '0';
    SIGNAL direction :  std_logic := '0';
    SIGNAL inp :  std_logic_vector(3 downto 0) := (others=>'0');

    --Outputs
    SIGNAL count :  std_logic_vector(3 downto 0);

BEGIN

    -- Instantiate the Unit Under Test (UUT)
    uut: countupd PORT MAP(
        clock => clock,
        reset => reset,
        clocken => clocken,
        loaden => loaden,
        direction => direction,
        inp => inp,
        count => count
    );


-- Please ensure that the constant PERIOD is defined prior to the
-- begin statement in the architecture. Refer to the PERIOD Constant Template
-- for more info.
process
begin
clock <= '0';
wait for 5 ns;
clock <= '1';
```

```
wait for 5 ns;
end process;

        tb : PROCESS
        BEGIN

                -- Wait 100 ns for global reset to finish
                wait for 100 ns;

                -- Place stimulus here
inp<="0101";
reset<='0';
clocken<='1';
loaden<='1';
wait for 100 ns;
loaden<='0';
direction<='1';
wait for 100 ns;
loaden<='0';
direction<='0';
                wait; -- will wait forever
        END PROCESS;

END;
```

6) Creating a Verilog Source

Create the top-level Verilog source file for the project as follows:
- Click **New Source** in the New Project dialog box.
- Select **Verilog Module** as the source type in the New Source dialog box.
- Type in the file name **countupd**.
- Verify that the **Add to Project** checkbox is selected.
- Click **Next**.
- Declare the ports for the countupd design by filling in the port information as shown below:

Port Name	Direction	Bus	MSB	LSB
clock	in			
reset	in			
clocken	in			
loaden	in			
direction	in			
inp	in		3	0
count	out		3	0

-Click **Next**, then **Finish** in the New Source Information dialog box to complete the new

source file template.

7) Using Language Templates (Verilog)

- Place the cursor on the line below the output [3:0] COUNT_OUT; statement.
- Open the Language Templates by selecting **Edit □ Language Templates...**
- Using the "**+**" symbol, browse to the following code example:
  Verilog -> Synthesis Constructs -> Coding Examples -> Binary ->
  Up/Down Counters -> /w Load, CE and Async Active High Reset

```
reg [<upper>:0] <reg_name>;

  always @(posedge <clock> or posedge <reset>)
    if (<reset>)
      <reg_name> <= 0;
    else if (<clock_enable>)
      if (<load_enable>)
        <reg_name> <= <load_signal_or_value>;
      else if (<up_down>)
        <reg_name> <= <reg_name> + 1;
      else
        <reg_name> <= <reg_name> - 1;
```

-With /w Load, CE and Async Active High Reset, select **Edit › Use in File**, or select the **Use Template in File** toolbar button. This step copies the template into the countupd source file.

8) Final Editing of the Verilog Source

- To declare and initialize the register that stores the countupd value, modify the declaration statement in the first line of the template as follows:
replace: reg [<upper>:0] <reg_name>;
with: reg [3:0] count_int = 0;

-Add the following line just above the endmodule statement to assign the register value to the output port:
assign COUNT_OUT = count_int;
- Save the file by selecting **File › Save**.
When you are finished, the code for the countupd will look like the following:

```
module countupd(clock, reset, clocken, loaden, direction, inp, count);
   input clock;
   input reset;
   input clocken;
   input loaden;
   input direction;
   input [3:0] inp;
   output [3:0] count;
```

```
// and timing. In general faster and smaller FPGA designs will
// result from not using asynchronous resets. Please refer to
// the Synthesis and Simulation Design Guide for more information.

  reg [3:0] count_int = 0;

  always @(posedge clock or posedge reset)
    if (reset)
      count_int <= 0;
    else if (clocken)
      if (loaden)
        count_int <= 0;
      else if (direction)
        count_int <= count_int + 1;
      else
        count_int <= count_int - 1;
assign COUNT_OUT = count_int;
endmodule
```

-Double-click the **Check Syntax** process.

9) Create the Test Bench of the Countupd HDL Source:

-Select the countupd HDL file in the sources window.
-Create a new test bench source by selecting project -> new source
-In the new source wizard,select test bench waveform as the source type and type countupd_tb in the file name field.

10) Gave Inputs to the Test Bench:

-Clock high time : 20 ns
-Clock low time : 20 ns
-Input setup time : 10 ns
-Offset : 0 ns
-Global signals : GSR(FPGA)
-Initial length of test bench : 1500 ns
-Direction high : 300 ns
-Direction low : 900 ns

11) Control the Inputs:

-Verify that behavioral simulation and countupd_tb are selected in the sources window.
-In the process tab , click the "+" to expand the xilinx ISE simulator process and double-click the simulate behavioral model process.

12) Entering Timing Constraint:

-Select synthesis/implementation from the drop-down list in the sources window.

-Select the countupd HDL source file.

-Click the "+" sign next to the user constraints processes group and double-click the create timing constraint process.

-Select clock in the clock net name field, then select the period toolbar button or double-click the empty period field to display the clock period dialog box.

-Enter 40 ns in the time field.

-Select the pad to setup toolbar button or double-click the empty pad to setup field to display the pad to setup dialog box.

-Enter 10 ns in the offset field to set the input offset constraint.

-Select the clock to pad toolbar button or double-click the empty clock to pad field to display the clock to pad dialog box.

-Enter 10 ns in the offset field to set the output delay constraint.

13) Implemantation:

-Select the countupd source file in the sources window.

-Open the design summary by double-clicking the view design summary process in the processes tab.

-Double-click the implement design process in the processes tab.

14) Pin Location Constraint:

-Verify that countupd is selected in the sources window.

-Double-click the assign package pins process found in the user constraints process group.

-Select the package view tab.

-In the design object list window,enter a pin location for each pin in the loc column using the following information:

*clock input port connects to pin t9

*cont_out<0> output port connects to pin k12

*cont_out<1> output port connects to pin p14

*cont_out<2> output port connects to pin l12

*cont_out<3> output port connects to pin n14

*direction input port connects to pin k13

15) Download Design to the Spartan™-3 Demo Board

- Select **countupd** in the Sources window.
- In the Processes window, click the "+" sign to expand the **Generate Programming File** processes.
- Double-click the **Configure Device (iMPACT)** process.
- The Xilinx WebTalk Dialog box may open during this process. Click **Decline.**
- Select **Disable the collection of device usage statistics for this project only** and click **OK**.
- In the Welcome dialog box, select **Configure devices using Boundary-Scan (JTAG).**
- Verify that **Automatically connect to a cable and identify Boundary-Scan chain** is selected.

- Click **Finish**.
- In the Welcome dialog box, select **Configure devices using Boundary-Scan (JTAG)**.
- Verify that **Automatically connect to a cable and identify Boundary-Scan chain** is selected.
- Click **Finish**.
-Right-click on the xc3s200 device image, and select **Program...** The **Programming Properties** dialog box opens.
- Click **OK** to program the device.

NEAR EAST UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

COM-473
HARDWARE DESIGN
PROJECT

ÖMÜR KUZUCU
20031116

1) Create a Counter Project:

-Select File -> New Project
-Type counter in the project name field.
-Enter to location and a counter subdirectory is created automatically.
-Verify that HDL is selected from top-level source type list.

2) Create the HDL Source of the Counter:

-Click the new source button in the new project wizard.
-Select VHDL module as the source type.
-Type in the file name **c**ounter.
-Verify that the add to project checkbox is selected.

3) HDL Language Template:

-Edit -> Language Templates
-VHDL -> Synthesis Constructs -> Coding Examples -> Binary ->
Up/Down Counters -> Simple Code

```
*process (<clock>)
 begin
        if <clock>='1' and <clock>'event then
                if <count_direction>='1' then
                        <count> <= <count> + 1;
                else
                        <count> <= <count> - 1;
                end if;
        end if;
end process;
```

4) Edit the HDL Code:

```
-library IEEE;
 use IEEE.STD_LOGIC_1164.ALL;
 use IEEE.STD_LOGIC_ARITH.ALL;
 use IEEE.STD_LOGIC_UNSIGNED.ALL;
 entity counter is
        Port (Clock: in STD_LOGIC;
              Direction: in STD_LOGIC;
              Count_Out : out STD_LOGIC_VECTOR (3 down to 0));
 end counter;
 architecture Behavioral of counter is
 signal Count_Int :STD_LOGIC_VECTOR (3 downto 0) := "0000";
 begin
        process (Clock)
  begin
```

```
        if Clock = '1' and Clock'event then
                if Direction = '1' then
                        Count_Int <= Count_Int + 1;
                else
                        Count_Int <= Count_Int – 1;
                end if;
        end if;
end process;
Count_Out <= Count_Int;
 end Behavioral;
```

5) Syntax Check:

-Verify that Synthesis/Implementation is selected from the drop-drown list in the sources window.
-Click the '+' next to the Synthesize-XST process to expand the process group.
-Double-click the Check Syntax process.

6) Create the Test Bench of the Counter HDL Source:

-Select the counter HDL file in the sources window.
-Create a new test bench source by selecting project -> new source
-In the new source wizard,select test bench waveform as the source type and type counter_tb in the file name field.

7) Gave Inputs to the Test Bench:

-Clock high time : 20 ns
-Clock low time : 20 ns
-Input setup time : 10 ns
-Offset : 0 ns
-Global signals : GSR(FPGA)
-Initial length of test bench : 1500 ns
-Direction high : 300 ns
-Direction low : 900 ns

8) Control the Inputs:

-Verify that behavioral simulation and counter_tb are selected in the sources window.
-In the process tab , click the "+" to expand the xilinx ISE simulator process and double-click the simulate behavioral model process.

9) Entering Timing Constraint:

-Select synthesis/implementation from the drop-down list in the sources window.
-Select the counter HDL source file.
-Click the "+" sign next to the user constraints processes group and double-click the

create timing constraint process.

-Select clock in the clock net name field, then select the period toolbar button or double-click the empty period field to display the clock period dialog box.

-Enter 40 ns in the time field.

-Select the pad to setup toolbar button or double-click the empty pad to setup field to display the pad to setup dialog box.

-Enter 10 ns in the offset field to set the input offset constraint.

-Select the clock to pad toolbar button or double-click the empty clock to pad field to display the clock to pad dialog box.

-Enter 10 ns in the offset field to set the output delay constraint.

10) Implemantation:

-Select the counter source file in the sources window.

-Open the design summary by double-clicking the view design summary process in the processes tab.

-Double-click the implement design process in the processes tab.

11) Pin Location Constraint:

-Verify that counter is selected in the sources window.

-Double-click the assign package pins process found in the user constraints process group.

-Select the package view tab.

-In the design object list window,enter a pin location for each pin in the loc column using the following information:

*clock input port connects to pin t9
*cont_out<0> output port connects to pin k12
*cont_out<1> output port connects to pin p14
*cont_out<2> output port connects to pin l12
*cont_out<3> output port connects to pin n14
*direction input port connects to pin k13

FULLADDER

DESING STEPS

DESING DESCRIPTION

TOOLS USED

CREATE THE NEW PROJECT FULLADDER

CREATE AN HDL HALFADD

CREATE TEST BENCH HALFADD

CREATE AN HDL ORGATE

CREATE TEST BENCH ORGATE

CREATE AN HDL FULLADD

CREATE TEST BENCH FULLADD

SIMILATION

DESING DESCRIPTION

This project name FULLADDER.

```
Entity FULLADD is
Port ( A : in  STD_LOGIC;
B : in  STD_LOGIC;
cin : in  STD_LOGIC;
sum : out  STD_LOGIC;
carry : out  STD_LOGIC);

 Entity halfadd is
Port{  A: in STD_LOGIC
B: in STD_LOGIC
Sum: out STD_LOGIC
carry: out STD_LOGIC

 Entity orgate is
Port{  a: in STD_LOGIC
b: in STD_LOGIC
z: out STD_LOGIC
```

The project family name Spartan3

Dvice XC3S200

Package FT256

Top-lwvwl source type HDL

TOOLS USED

HARDWARE

Family name Spartan3

Dvice XC3S200

Package FT256

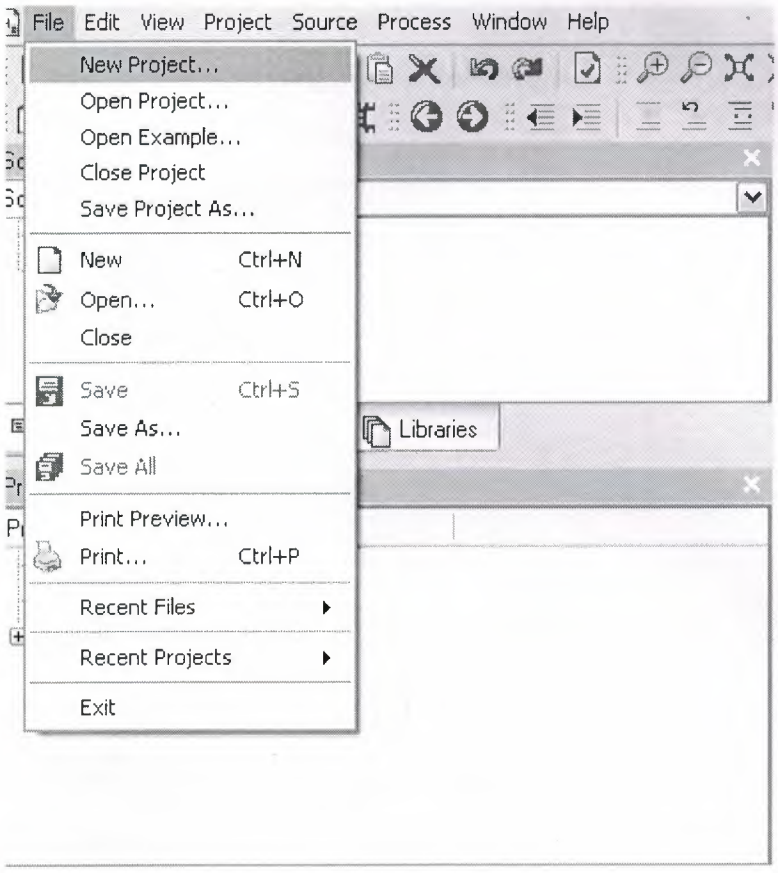
Speed -4

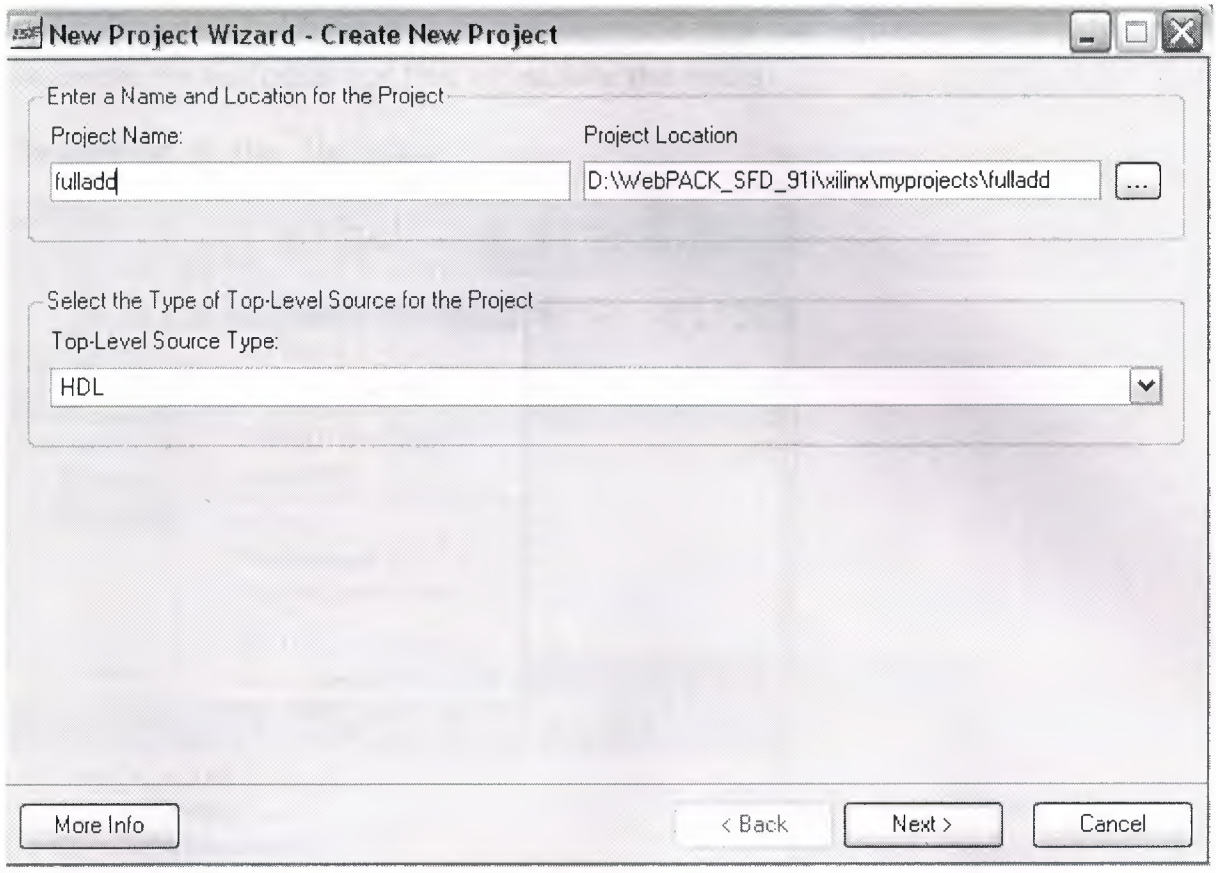
SOFTWARE

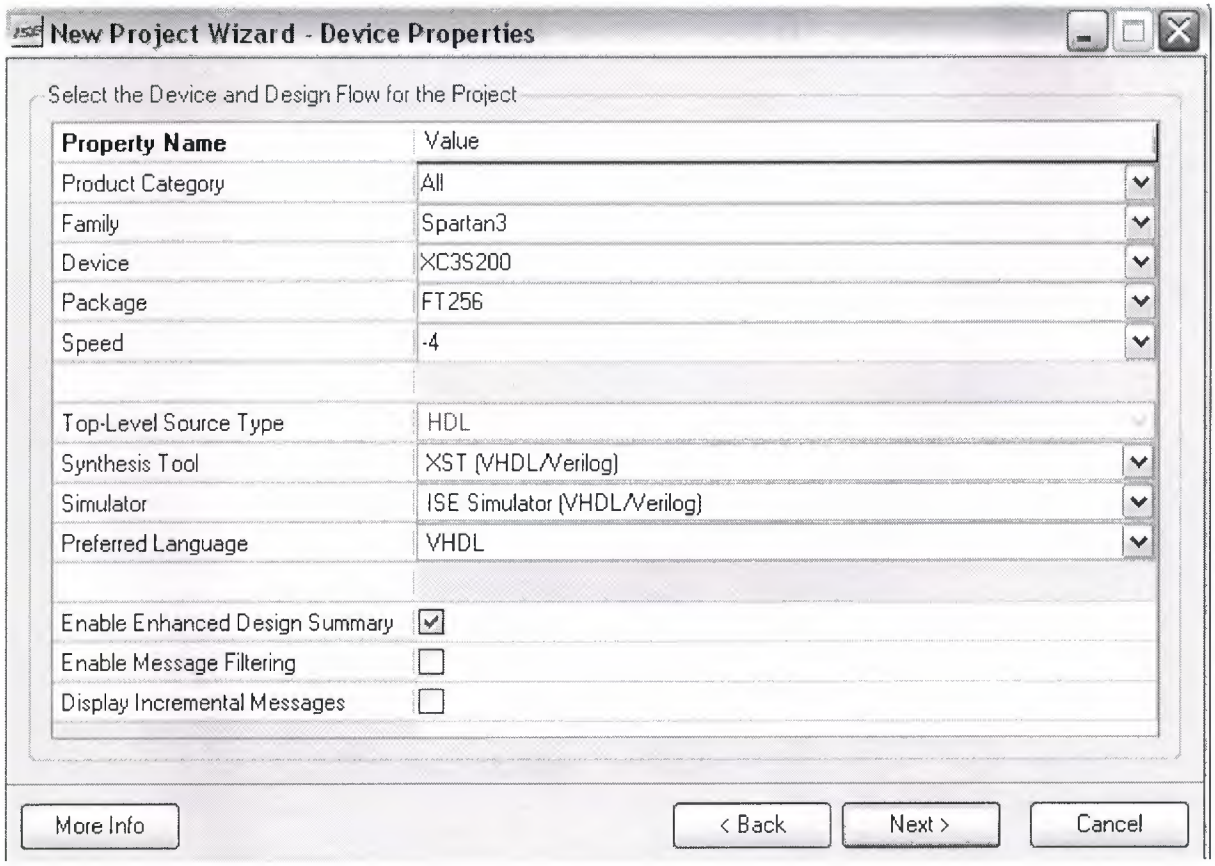
XILINX-ISE 9.1 i

Top Level source type	VHDL
Synthesis tool	XST (VHDL/verilog)
Similator	ISE Similator (VHDL/verilog)
Language	VHDL

1.Design Steps Of FULLADDER

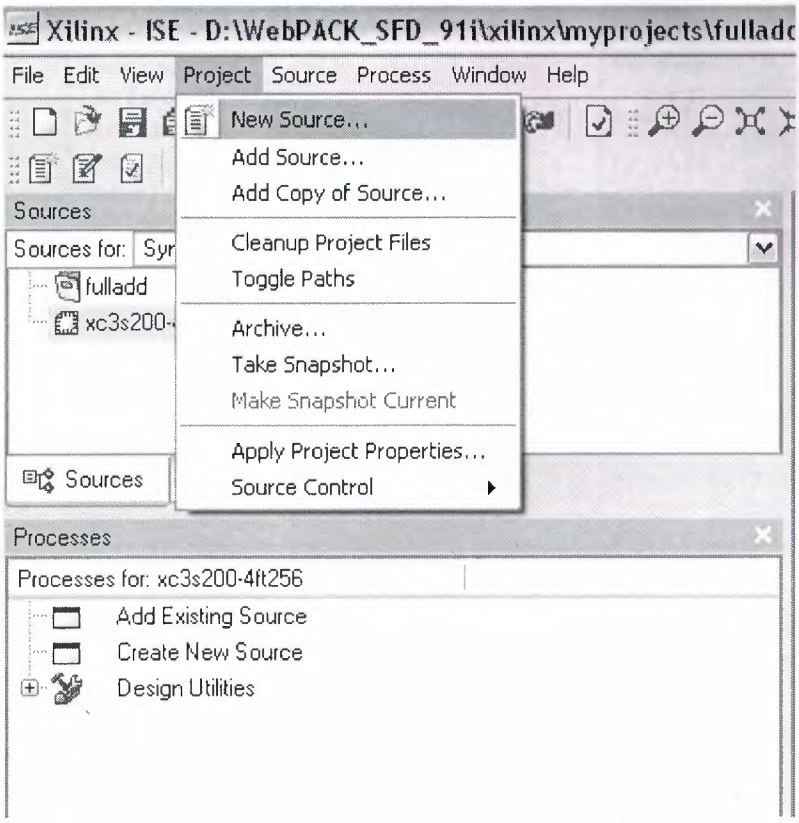


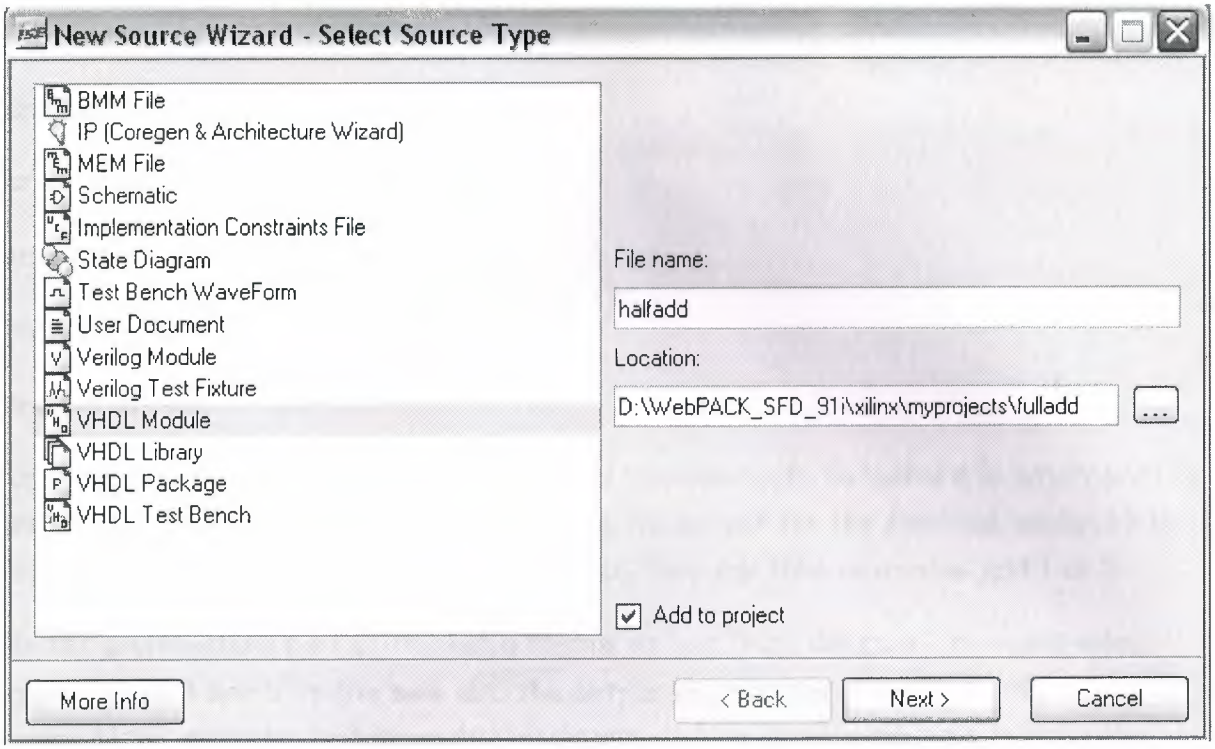

New Project Wizard - Create New Project
Enter a Name and Location for the Project
Project Name:
fulladd
Project Location
D:\WebPACK_SFD_91i\xilinx\myprojects\fulladd
...
Select the Type of Top-Level Source for the Project
Top-Level Source Type:
HDL
More Info
< Back
Next >
Cancel

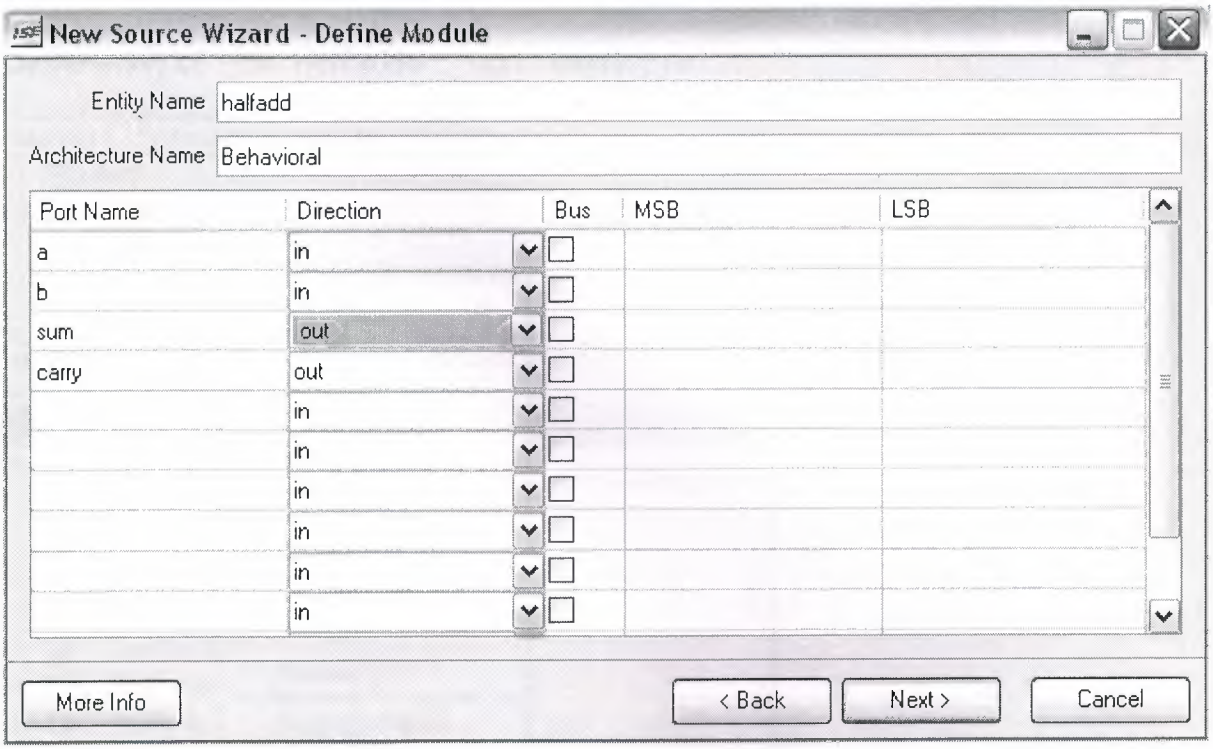

New Project Wizard - Device Properties
Select the Device and Design Flow for the Project
Property Name
Value
Product Category
All
Family
Spartan3
Device
XC3S200
Package
FT256
Speed
-4
Top-Level Source Type
HDL
Synthesis Tool
XST (VHDL/Verilog)
Simulator
ISE Simulator (VHDL/Verilog)
Preferred Language
VHDL
Enable Enhanced Design Summary
Enable Message Filtering
Display Incremental Messages
More Info
< Back
Next >
Cancel

Before create a fulladder in VHDL file we create firsr two halfadder and orgate.First we create the halfadder for that we declare the inputs .

Decleration of the Halfadder







Entity halfadd is

Port{ A: in STD_LOGIC

B: in STD_LOGIC

Sum: out STD_LOGIC

carry: out STD_LOGIC

end halfadd;

architecture behaviroal of halfadd is

sum<=A xor B

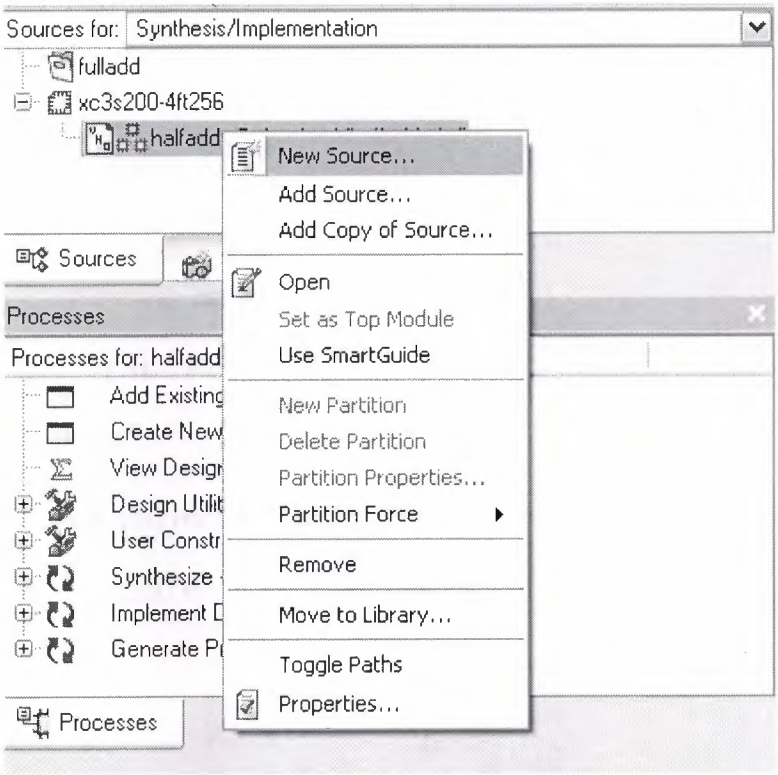
carry<=A and B

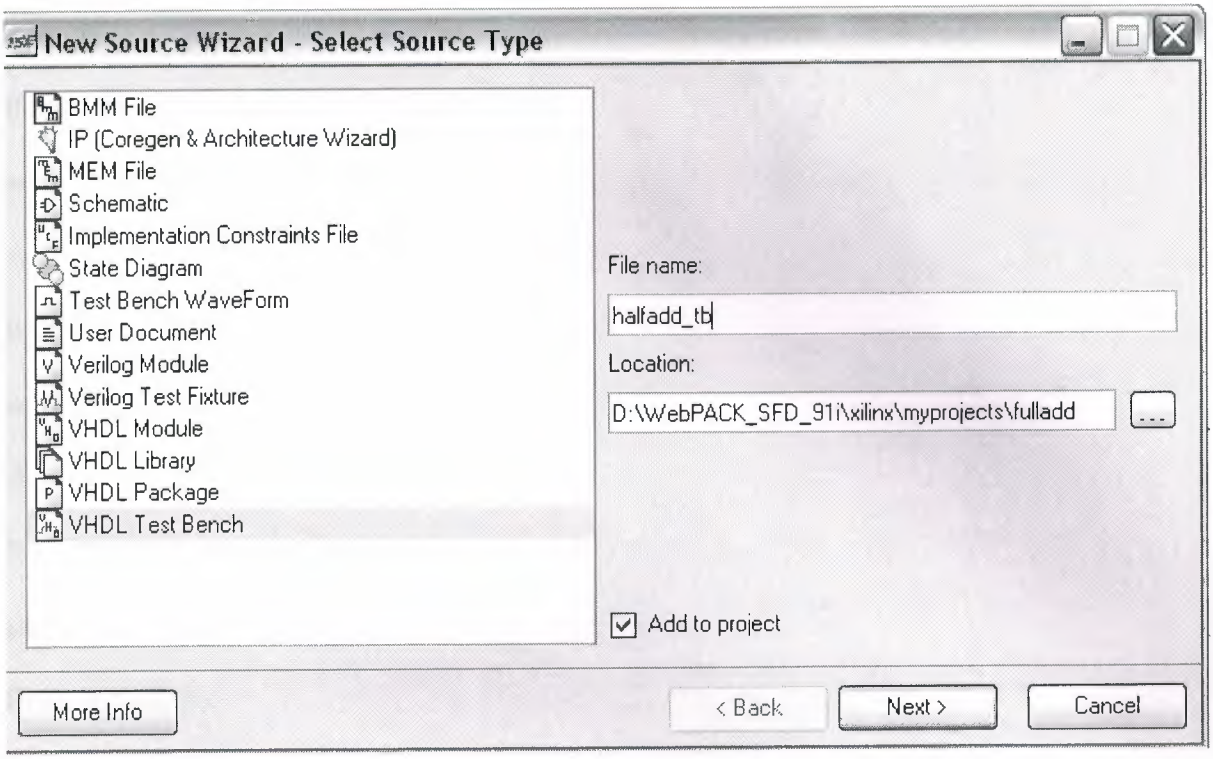
end behaviroal;

In the entity part of the halfadd we declera the inputs of the halfadd.In entity part as we see A,B are the inputs and sum, carry are the output for the first halfadder.All the inputs and the outputs types are STD_LOGIC.They can take as a value just 1 or 0.

By the architecture part of the halfadder as we see from the code.The ouput value comes from' A xor B' to the **sum** also the output value comes' A and B' to the carry.After cretating halfadder file by double clicking check syntax we compile the page to able to create the test bench file.

Decleration of the Halfadder Test Bench File



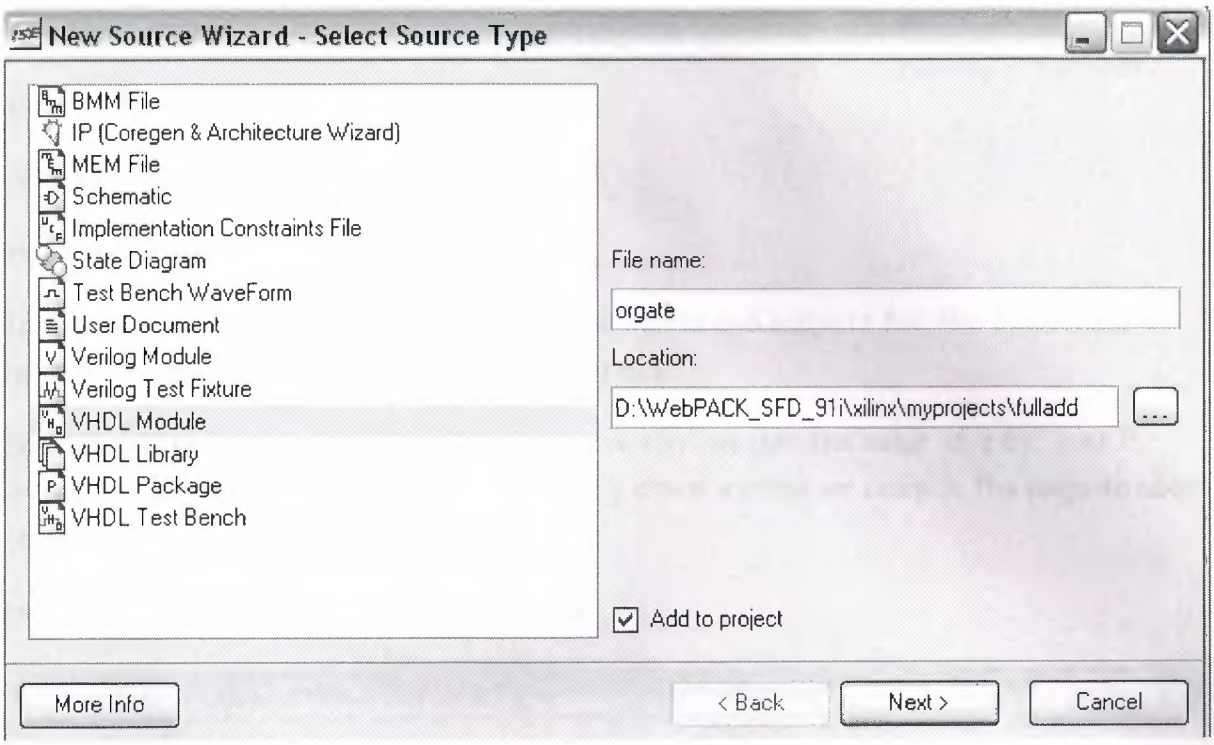


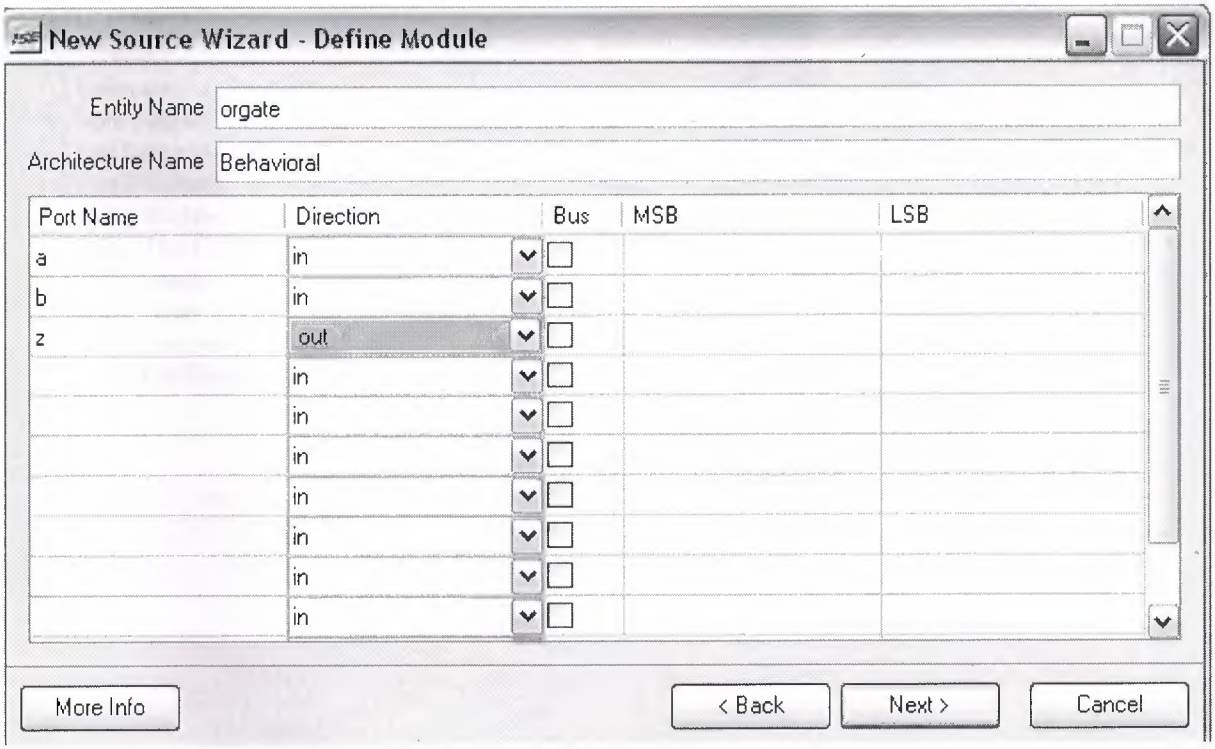

```
BEGIN

uut: halfadd PORT MAP(

                A => A,

                B => B,

                sum => sum,

                carry => carry

        ); tb : PROCESS

BEGIN

-- Wait 100 ns for global reset to finish

wait for 100 ns;

A <= '0';

B <= '0';

-- Place stimulus here

wait for 100 ns;
```

```
A <= '0';

B <= '1';

wait for 100 ns;

A <= '1';

B <= '0';

wait for 100 ns;

A <= '1';

B <= '1';

wait; -- will wait forever

END PROCESS;

END;
```

At the port map part we just declera again the inputs and the outputs.Where the alues will come from. For the inputs A,B we assign the values to able to see the results in the simulation part .Then by double clicking simulate behaviroal model we can see the results.As shown below:

Decleration of Orgate





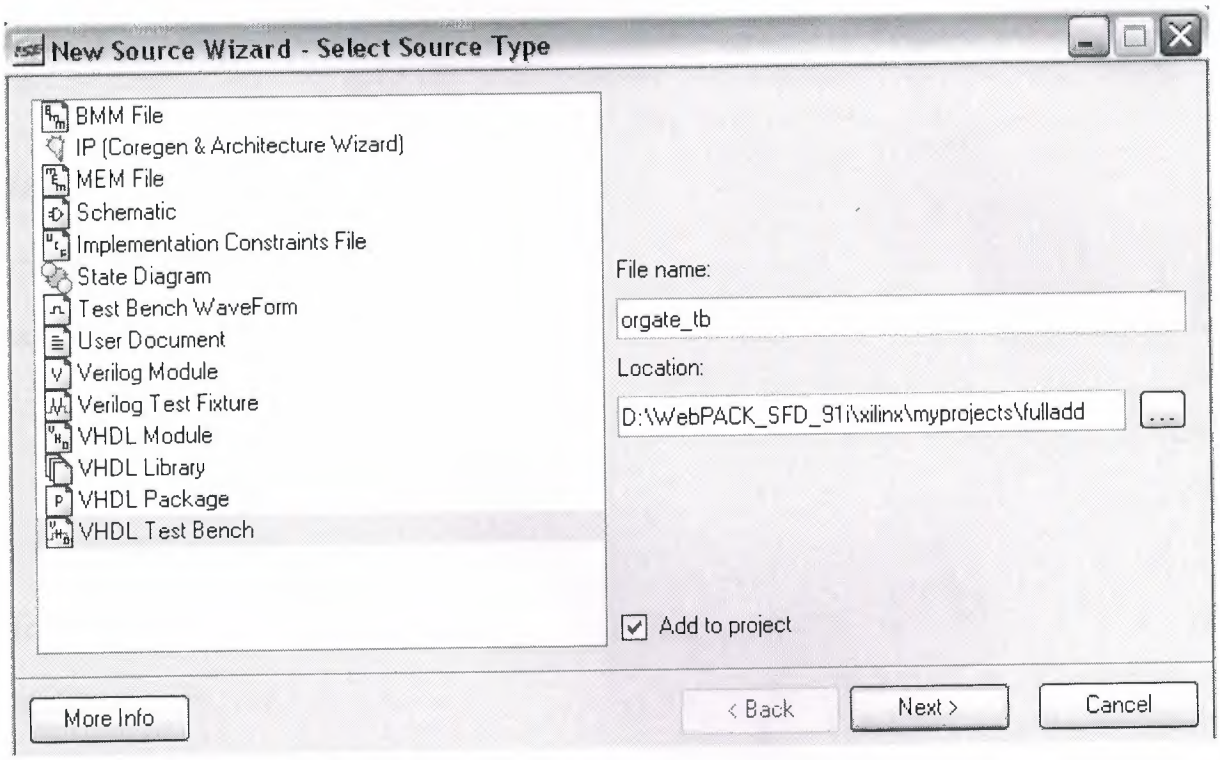

```
Entity orgate is

Port{  a: in STD_LOGIC

       b: in STD_LOGIC

       z: out STD_LOGIC
```

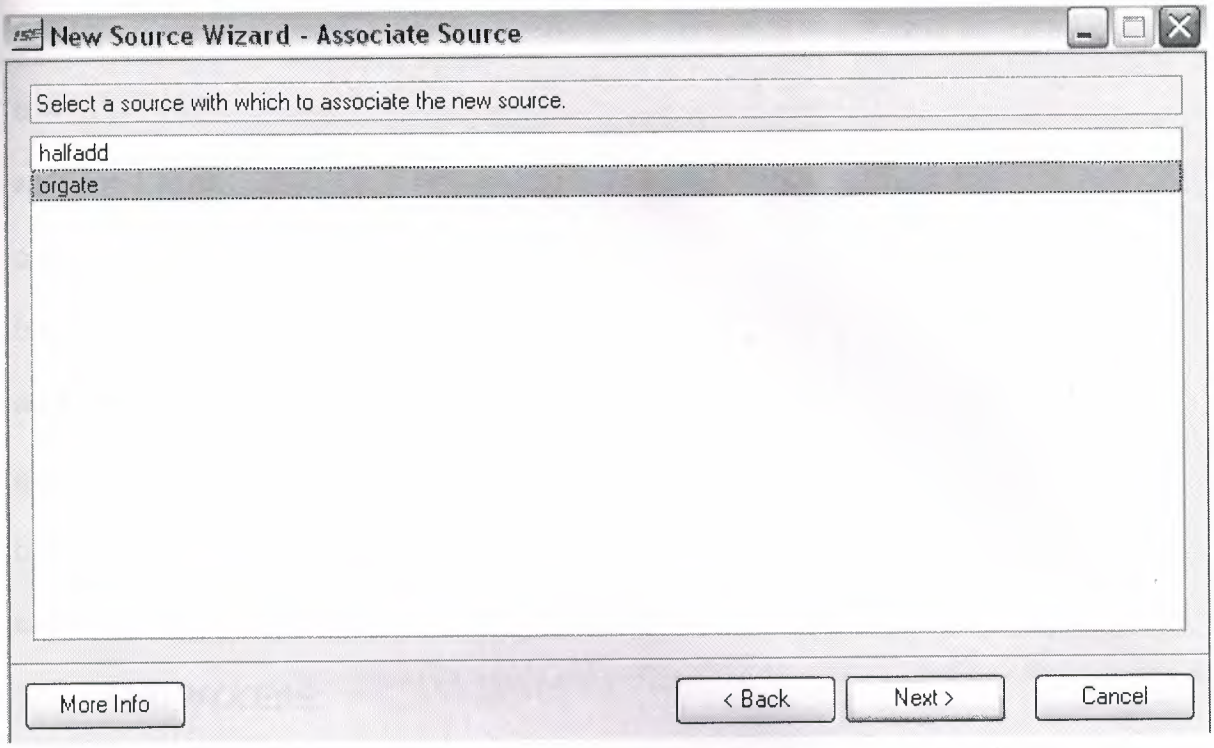
```
    end orgate;

architecture  behaviroal of orgate is

z<= a or b;

end behaviroal;
```

In the entity part of orgate we declared the inputs and outputs for the fulladder.In here again A,B are the inputs and the output is z.

In the architecture part of orgate we can see that we get the value of z by 'a or b'. After cretating orgate file by double clicking check syntax we compile the page to able to create the test bench file.

Decleration of the Orgate Test Bench File



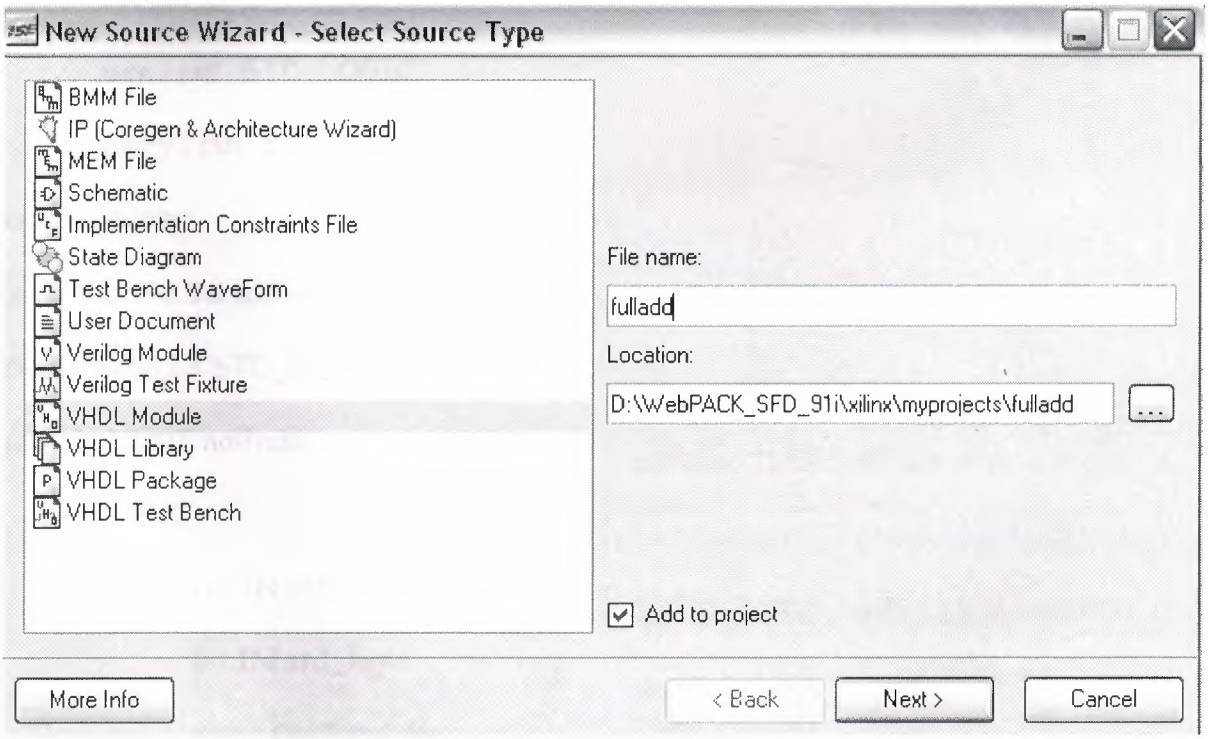


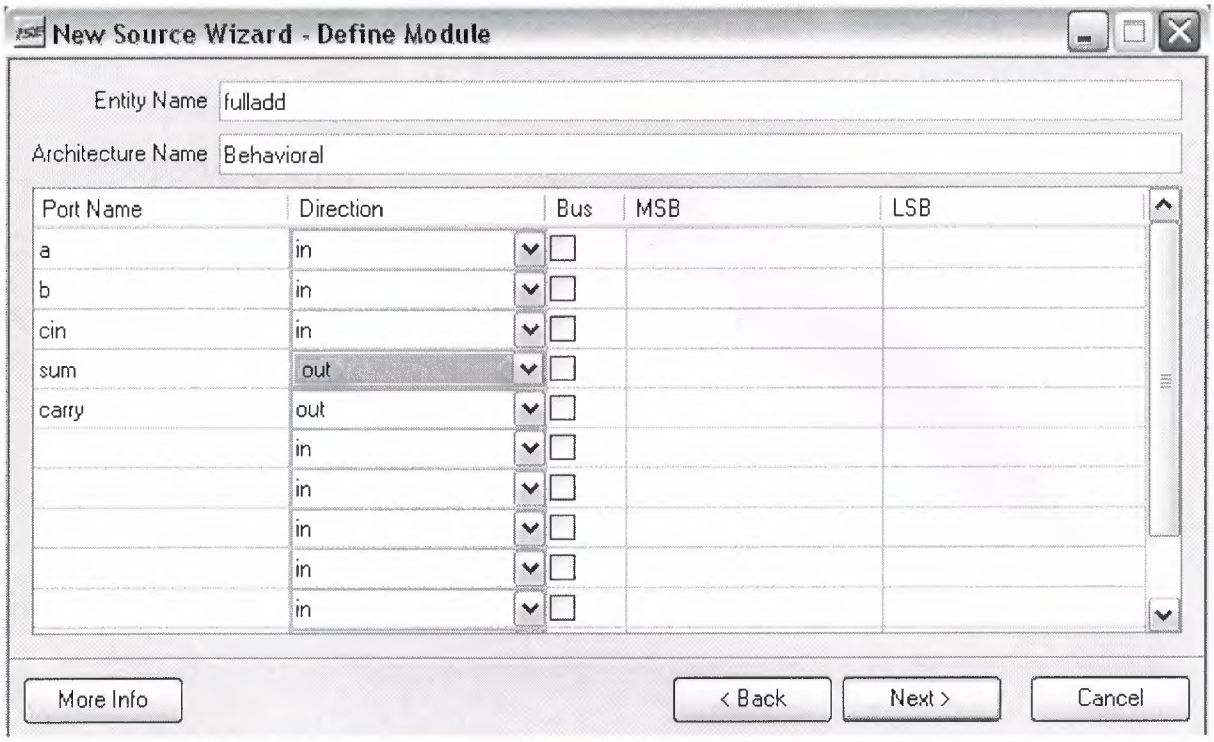

```
BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: orgate PORT MAP(

                a => a,

                b => b,

                z => z);

tb : PROCESS

BEGIN

-- Wait 100 ns for global reset to finish

        wait for 100 ns;

a <= '0';

b <= '0';

-- Place stimulus here

wait for 100 ns;
```

```
a <= '0';

b <= '1';

wait for 100 ns;

a <= '1';

b <= '0';

wait for 100 ns;

a <= '1';

b <= '1';

wait; -- will wait forever

        END PROCESS;
```

It is the same with halfadder test bench file again in port map part we defined the inputs and outputs.Then able to see in the simulation we give values for a and b , the output values assigned to the **z** by **a or b.**

Decleration of Fulladder






```
entity fulladder is
   Port ( A : in  STD_LOGIC;
          B : in  STD_LOGIC;
          cin : in  STD_LOGIC;
          sum : out  STD_LOGIC;
          carry : out  STD_LOGIC);
end fulladder;
architecture Behavioral of fulladder is
signal I1,I2,I3:STD_LOGIC;
COMPONENT halfadd
          PORT(
                    A : IN std_logic;
                    B : IN std_logic;
                    sum : OUT std_logic;
```

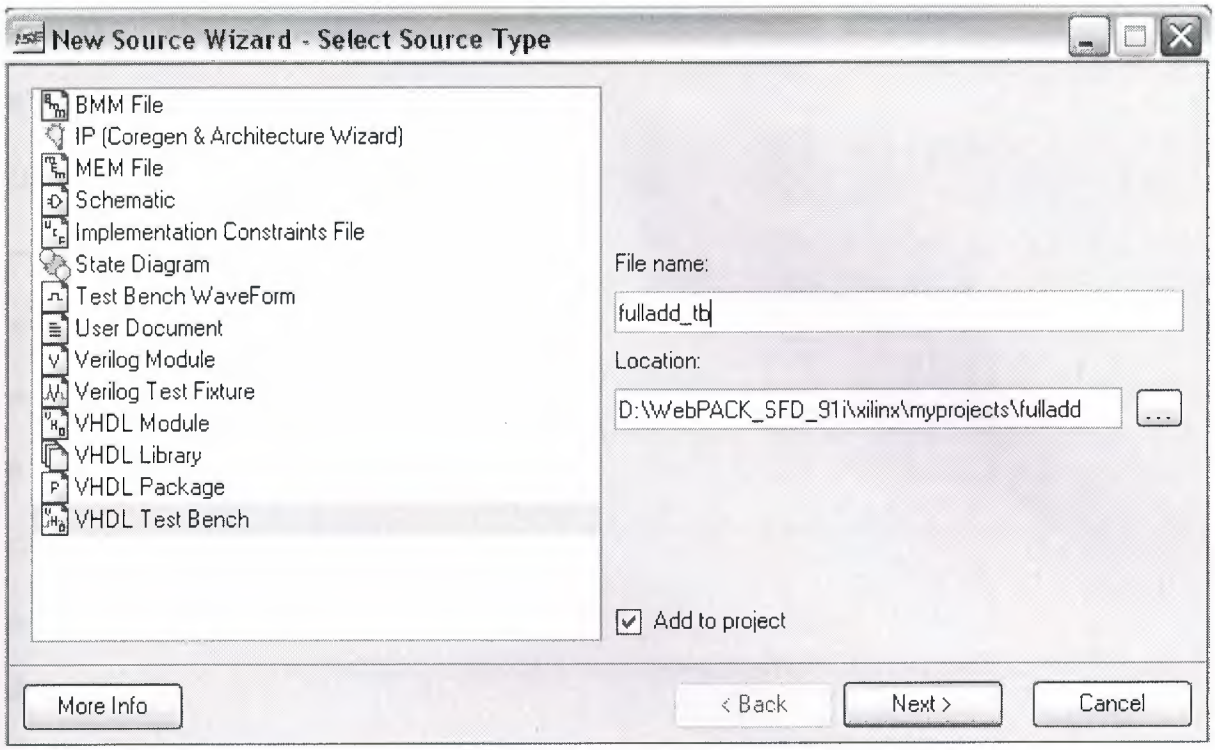
```
            carry : OUT std_logic
            );
    END COMPONENT;
COMPONENT orgate
PORT(       a : IN std_logic;
            b : IN std_logic;
            z : OUT std_logic
            );
END COMPONENT;
begin
u1: halfadd port map (a,b,I1,I2);
u2: halfadd port map (I1,cin,sum,I3);
u3: orgate  port map (I3,I2,carry);
end Behavioral;
```

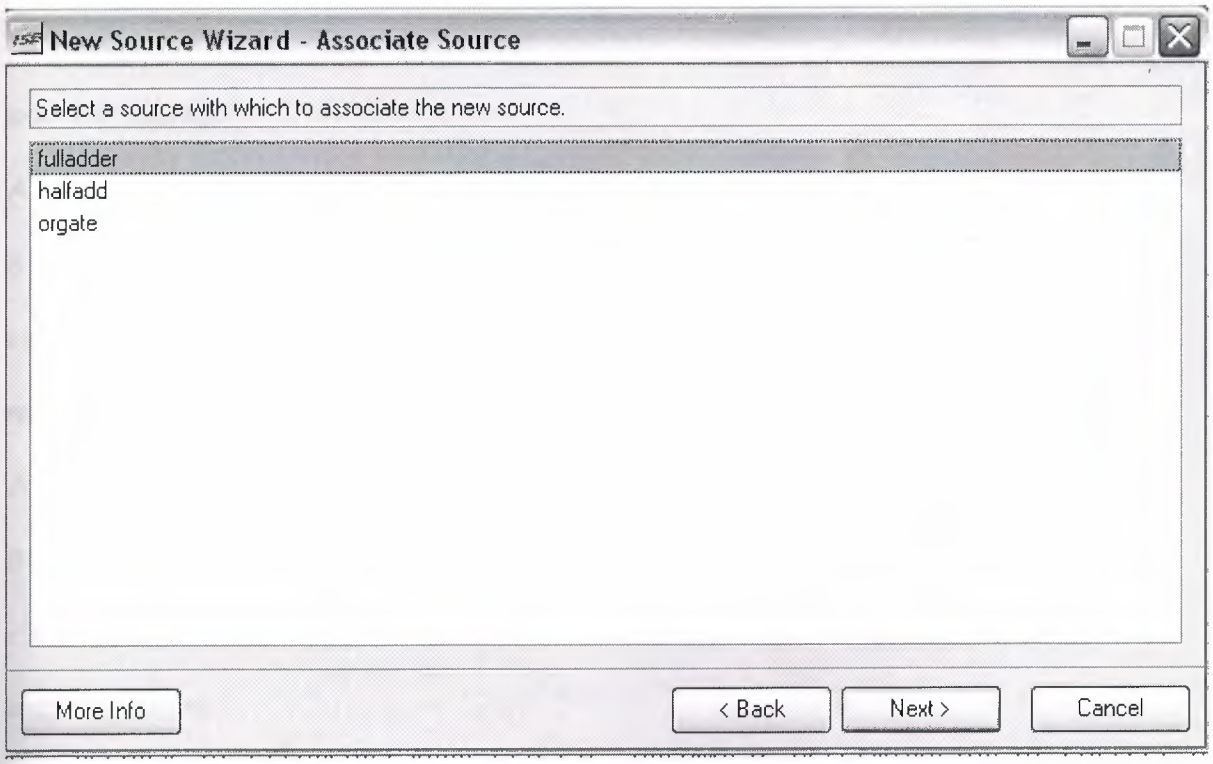
In fulladder differently we declare signals they are 1,I2,I3,I4.Those signals type also STD_LOGIC they can get as value just 1or 0.By this part of code

```
        begin
u1: halfadd port map (a,b,I1,I2);
u2: halfadd port map (I1,cin,sum,I3);
u3: orgate  port map (I3,I2,carry);
    end Behavioral;
```

We assign the outputs of first halfadder I1,I2 , assign for second halfadder first input I1,second output I3,assign the inputs of orgate I3,I4.After finishing those assigments wesave the file and double clicking check syntax we compile the file and then we can create the test bench file.

Decleration of Fulladder Test Bench File



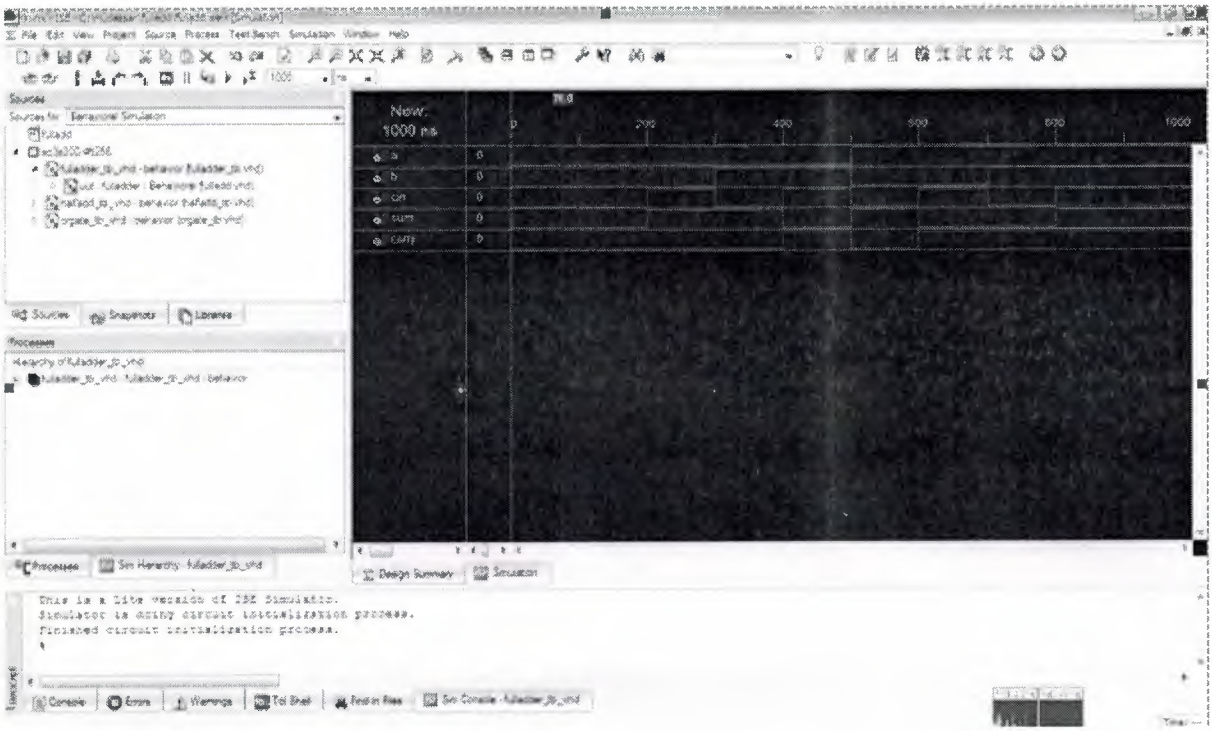


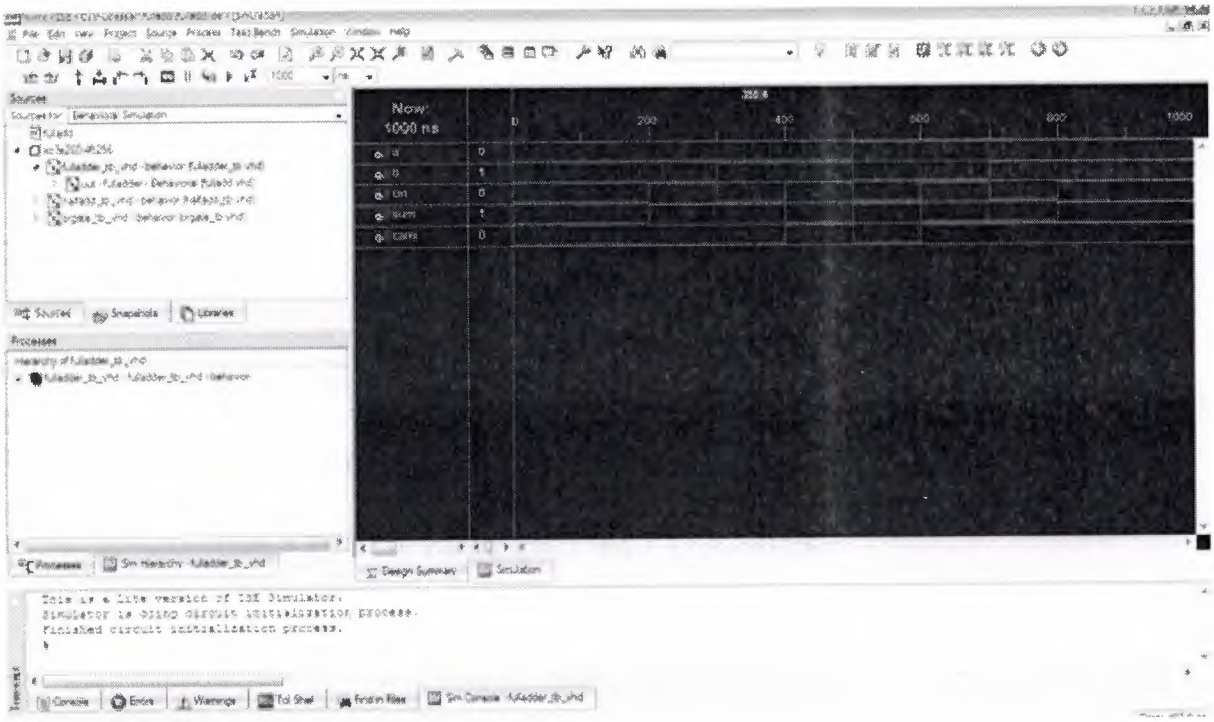

```
ARCHITECTURE behavior OF fulladder_tb_vhd IS

BEGIN

-- Wait 100 ns for global reset to finish

wait for 100 ns;
```

```
A <= '0';
B <= '0';
cin <= '0';
-- Place stimulus here
wait for 100 ns;
A <= '0';
B <= '0'
cin <= '1';
wait for 100 ns;
A <= '0';
B <= '1';
cin <= '0';
wait for 100 ns;
A <= '0';
B <= '1';
cin <= '1';
wait for 100 ns;
A <= '1';
B <= '0';
cin <= '0';
wait for 100 ns;
A <= '1';
B <= '0';
cin <= '1';
wait for 100 ns;
```

```
A <= '1';

B <= '1';

cin <= '0';

wait for 100 ns;

A <= '1';

B <= '1';

cin <= '1';

wait; -- will wait forever

END PROCESS;

END;
```

In the test bench file we assign values for the inputs A,B,Cin for the simulation design as we can see from the pictures in the different values of inputs it gives different outputs.In the first picture all the values are 0 and outputs are zero .In the second picture a=0,b=1 so cin =0,sum=1,carry=0.As we see from the project 1+0=0 with co carry.


Now:
1000 ns
a
b
cin
sum
cout
Design Summary
Simulation
This is a Lite version of ISE Simulator.
Simulator is doing circuit initialization process.
Finished circuit initialization process.
Console
Errors
Warnings
Tcl Shell
Find in Files


Now:
1000 ns
0
200
400
600
800
1000
Sources
Processes
Design Summary
Simulation
This is a Lite version of XSIM Simulator.
Simulator is doing circuit initialization process.
Finished circuit initialization process.
Console
Errors
Warnings
Tcl Shell
Find in Files

DESING STEPS

DESING DESCRIPTION

TOOLS USED

CREATE THE NEW PROJECT COUNTER

CREATE AN HDL SOURCE

DESING SIMILATION

SIMILATION DESING FUNCTIONALITY

CREATE VHDL TEST BENCH

DESING DESCRIPTION

This project name STAT_TWO.
The project have tree ports.

A	: in STD_LOGIC.
CLOCK	: in STD_LOGIC.
RESET	: in STD_LOGIC.
X	: out STD_LOGIC.

The project family name Spartan3
Dvice XC3S200
Package FT256
Top-lwvwl source type HDL

TOOLS USED

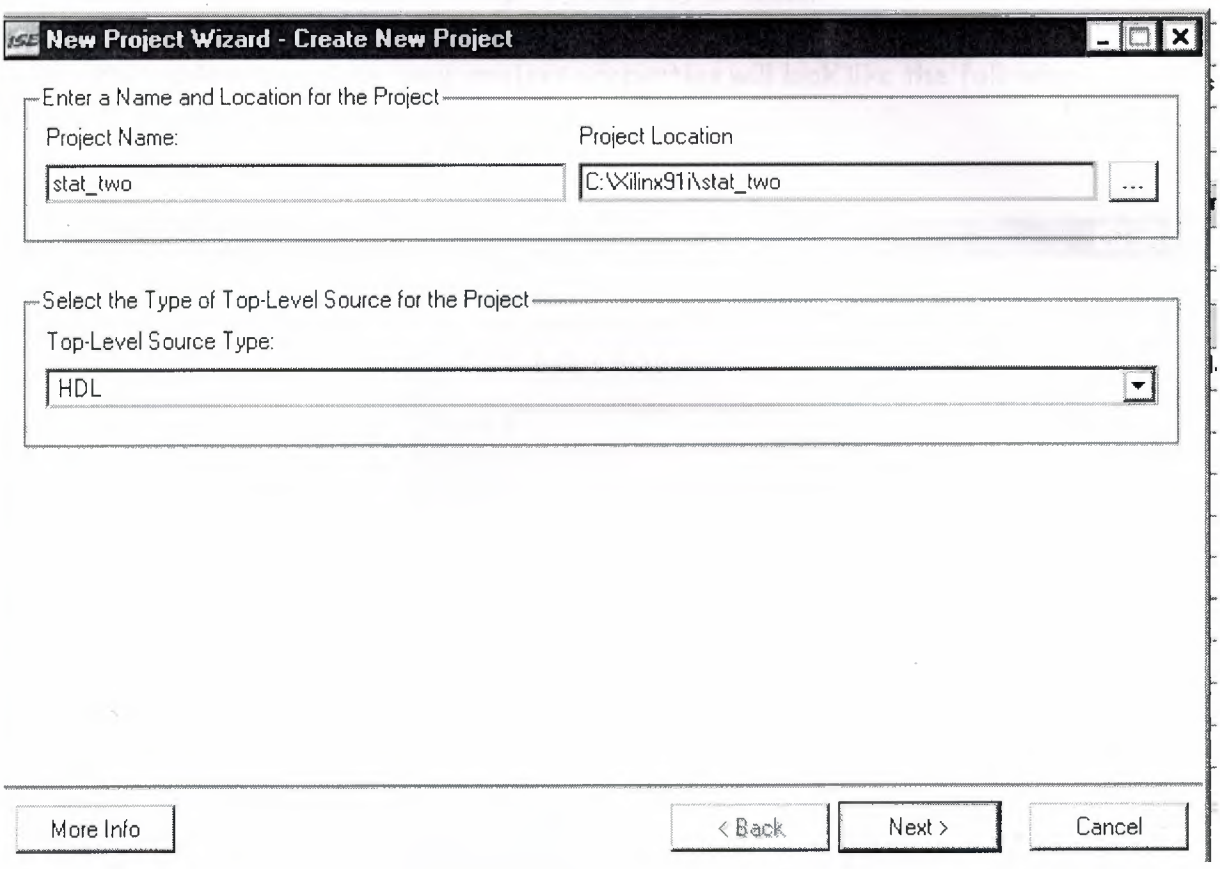
HARDWARE
Family name Spartan3
Dvice XC3S200
Package FT256
Speed -4

SOFTWARE
XILINX-ISE 9.1 i

Top Level source type	HDL
Synthesis tool	XST (VHDL/verilog)
Similator	ISE Similator (VHDL/verilog)
Language	VHTL

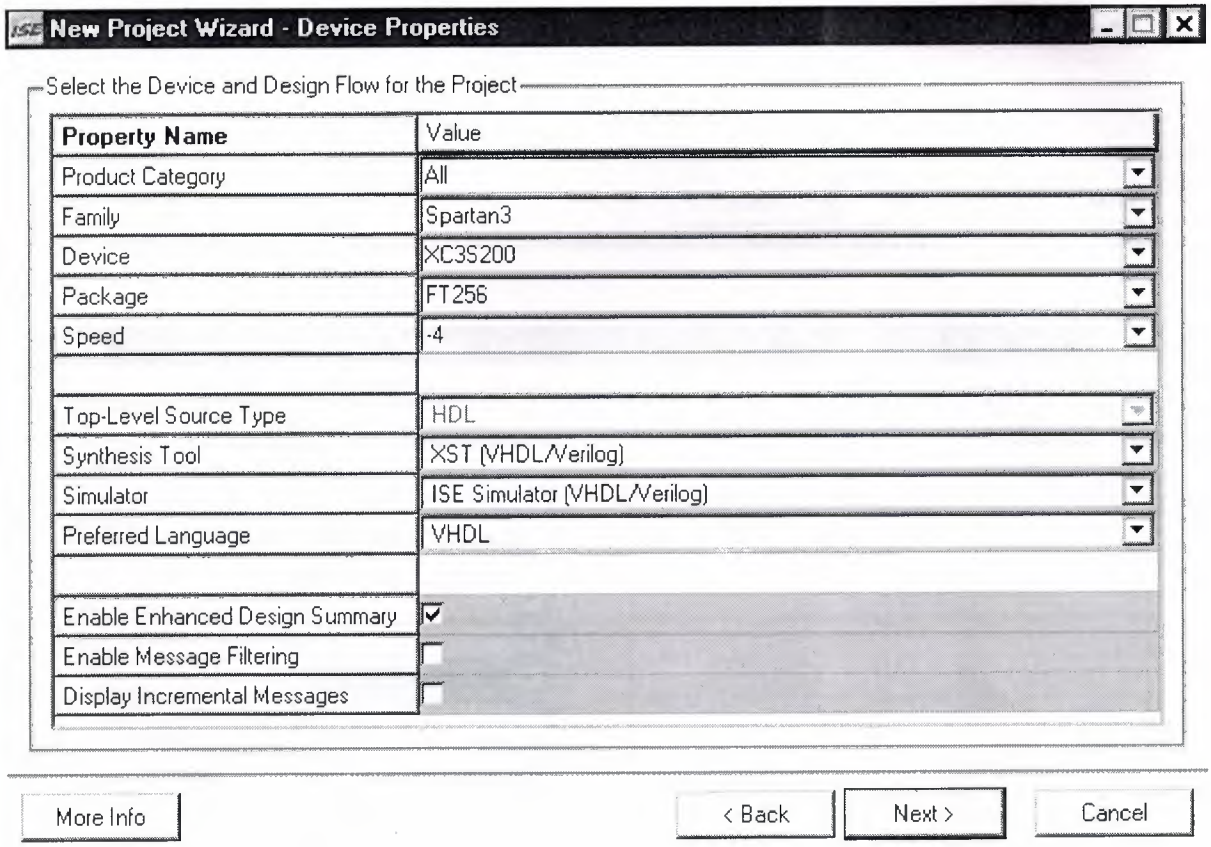
STAT_TWO

Create a New Project Counter



Create a new ISE project which will target the FPGA device on the Spartan-3 Startup Kit
demo board.
To create a new project:
1. Select File □□New Project... The New Project Wizard appears.
2. Type stat_two in the Project Name field.
3. Enter or browse to a location (directory path) for the new project. A tutorial subdirectory is created automatically.
4. Verify that HDL is selected from the Top-Level Source Type list.
5. Click Next to move to the device properties page.
6. Fill in the properties in the table as shown below:
□□Product Category: All
□□Family: Spartan3
□□Device: XC3S200

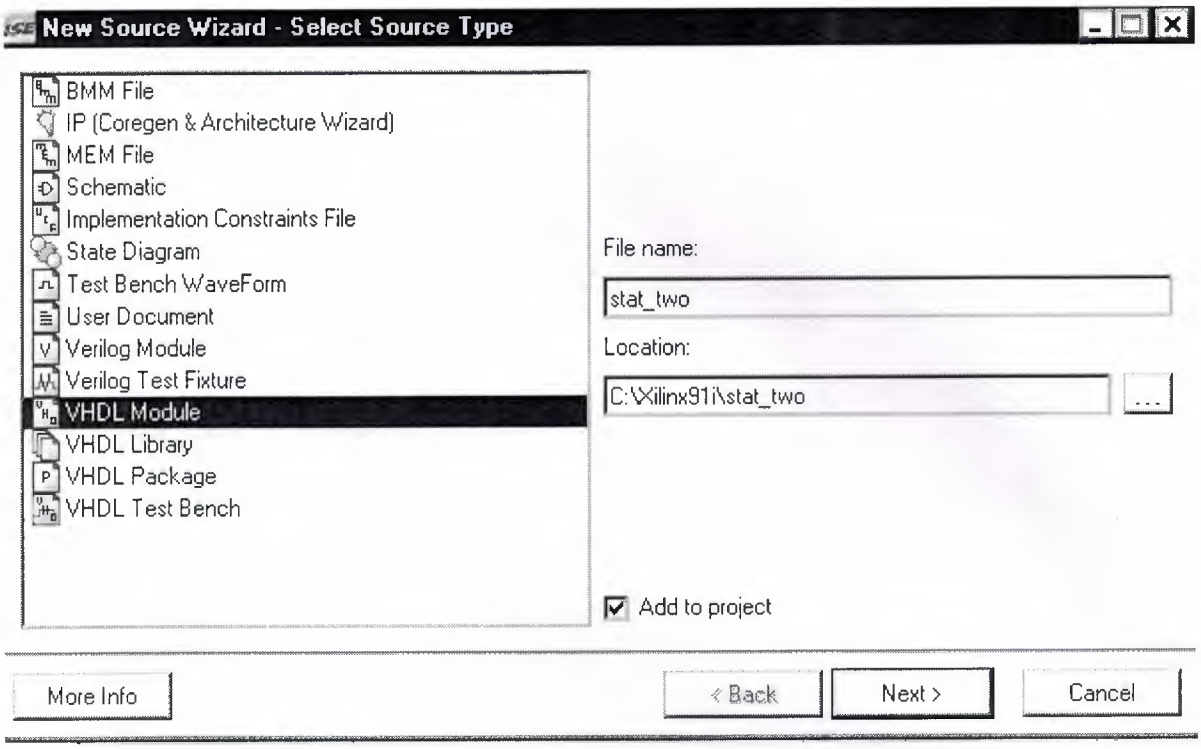
Package: FT256
Speed Grade: -4
Top-Level Source Type: HDL
Synthesis Tool: XST (VHDL/Verilog)
Simulator: ISE Simulator (VHDL/Verilog)
Preferred Language: Verilog (or VHDL)
Verify that Enable Enhanced Design Summary is selected.
Leave the default values in the remaining fields.
When the table is complete, your project properties will look like the following:
Figure 2: Project Device Properties

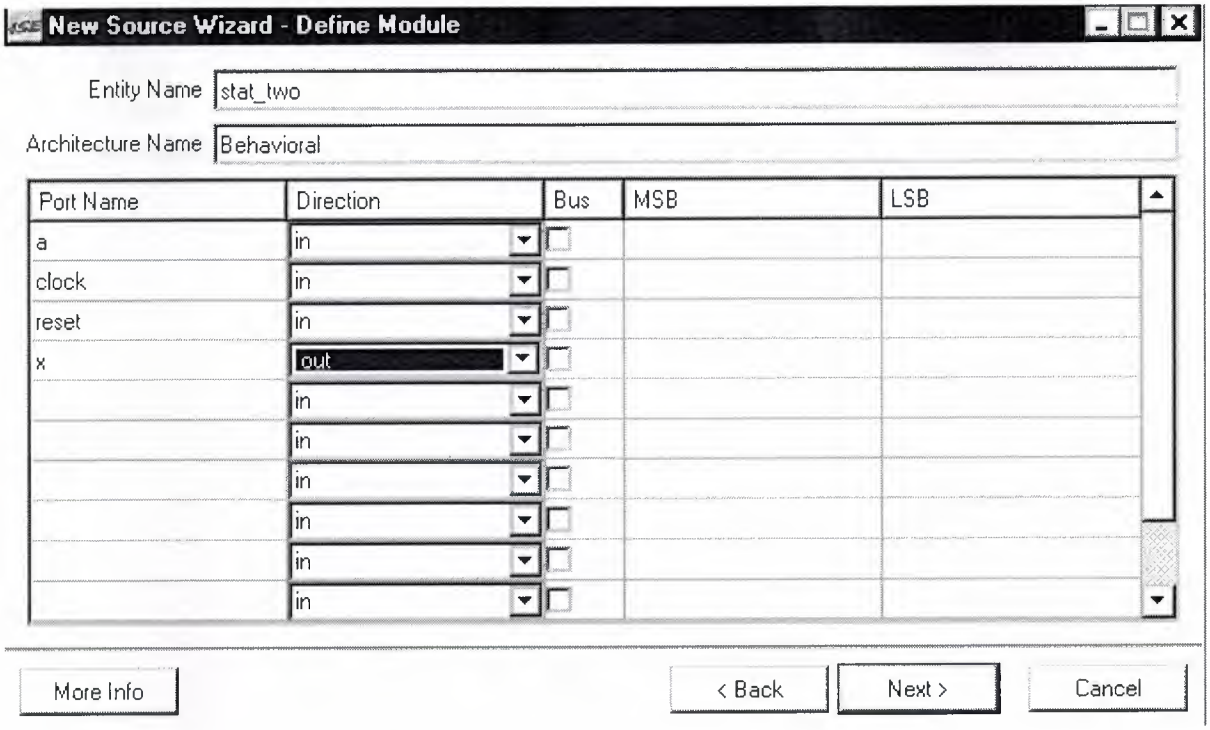


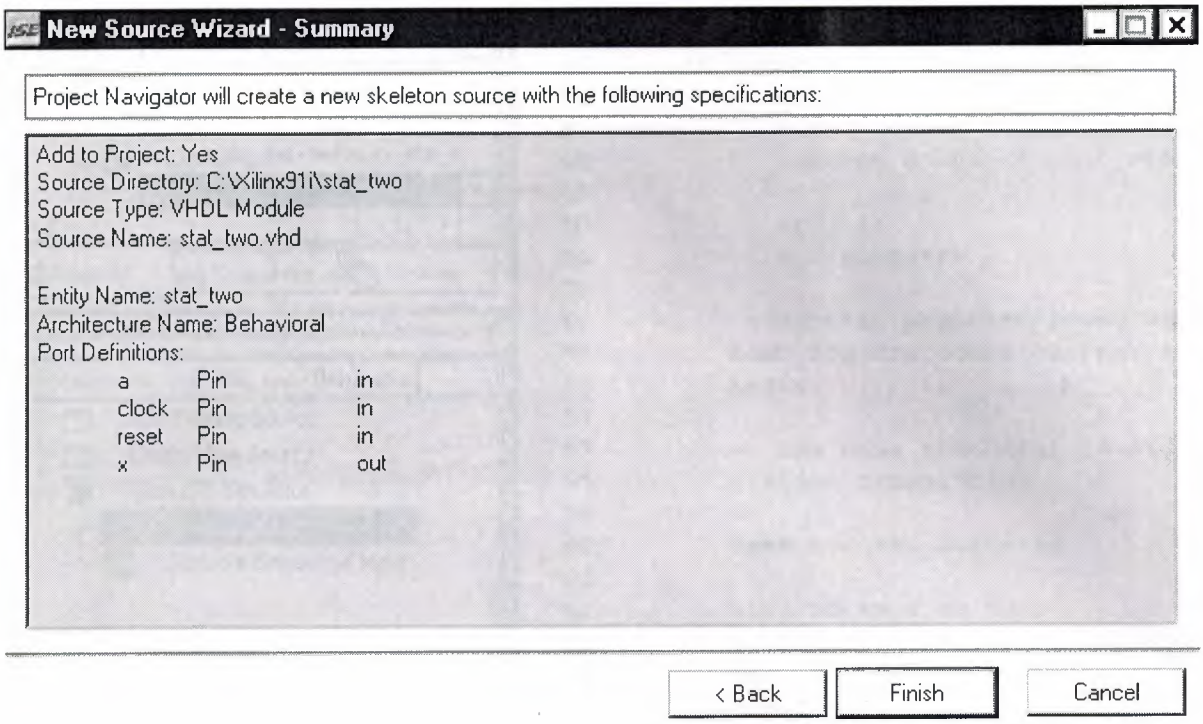
Create an HDL Source

Create a VHDL source file for the project as follows:

1. Click the New Source button in the New Project Wizard.
2. Select VHDL Module as the source type.
3. Type in the file name stat_two.
4. Verify that the Add to project checkbox is selected.
5. Click Next.
6. Declare the ports for the counter design by filling in the port information as shown below:

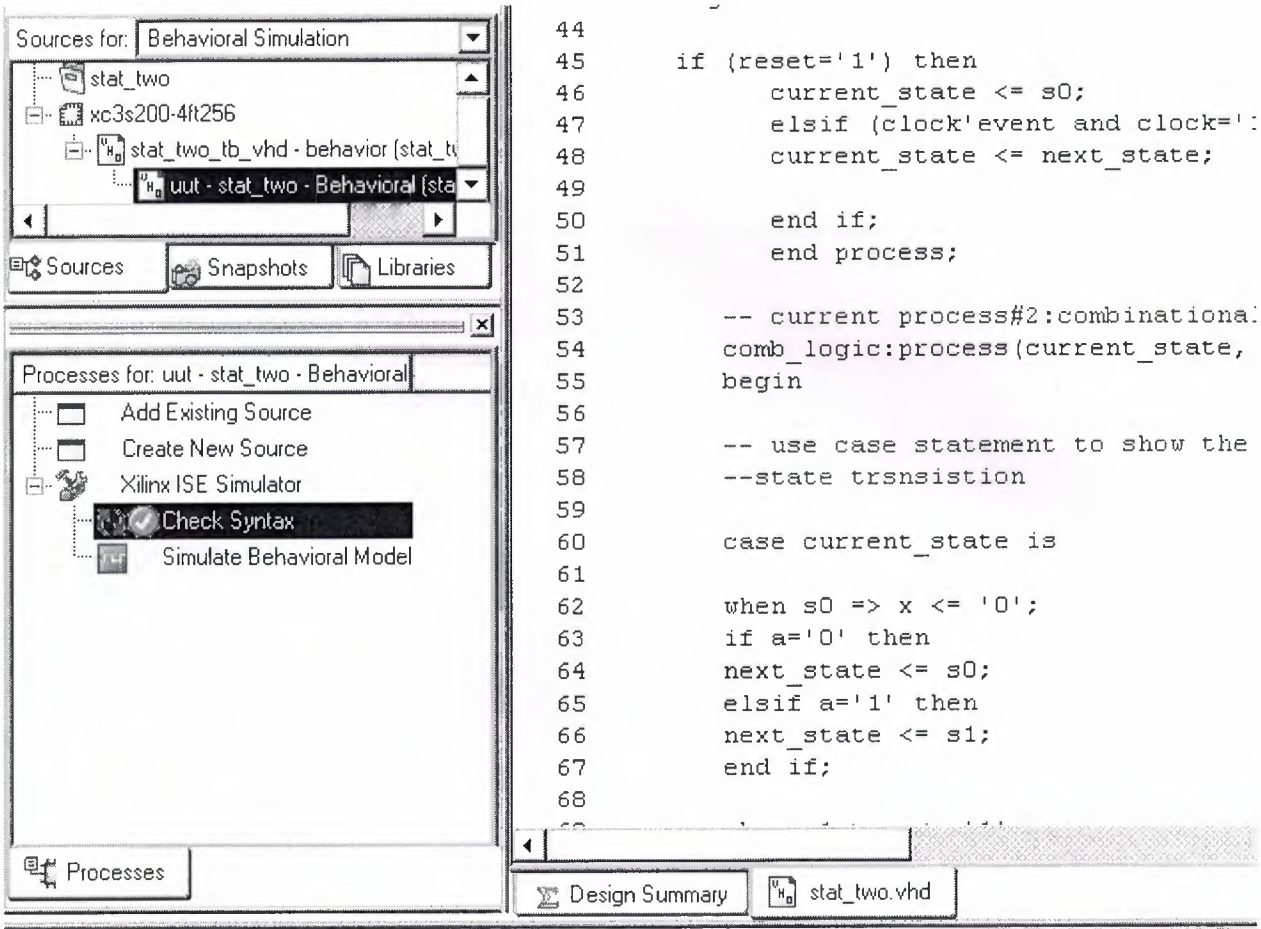

New Source Wizard - Select Source Type
BMM File
IP (Coregen & Architecture Wizard)
MEM File
Schematic
Implementation Constraints File
State Diagram
Test Bench WaveForm
User Document
Verilog Module
Verilog Test Fixture
VHDL Module
VHDL Library
VHDL Package
VHDL Test Bench
File name:
stat_two
Location:
C:\Xilinx91i\stat_two
...
Add to project
More Info
< Back
Next >
Cancel


New Source Wizard - Define Module
Entity Name
stat_two
Architecture Name
Behavioral
Port Name
Direction
Bus
MSB
LSB
a
in
clock
in
reset
in
x
out
in
in
in
in
in
in
More Info
< Back
Next >
Cancel



7. Click Next, then Finish in the New Source Wizard - Summary dialog box to complete the new source file template.
8. Click Next, then Next, then Finish.

Stat_two displays in the Source tab, as shown below:

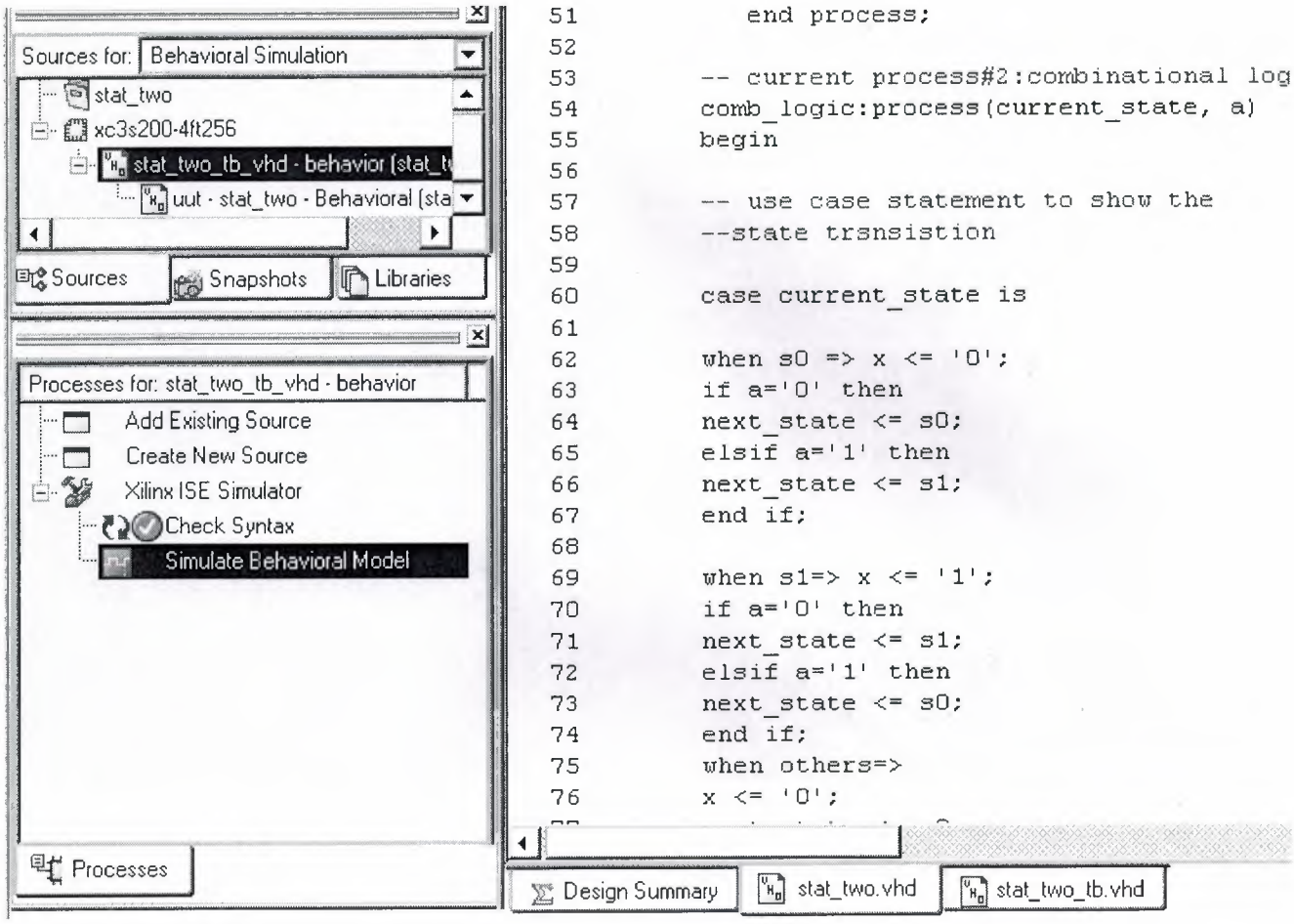


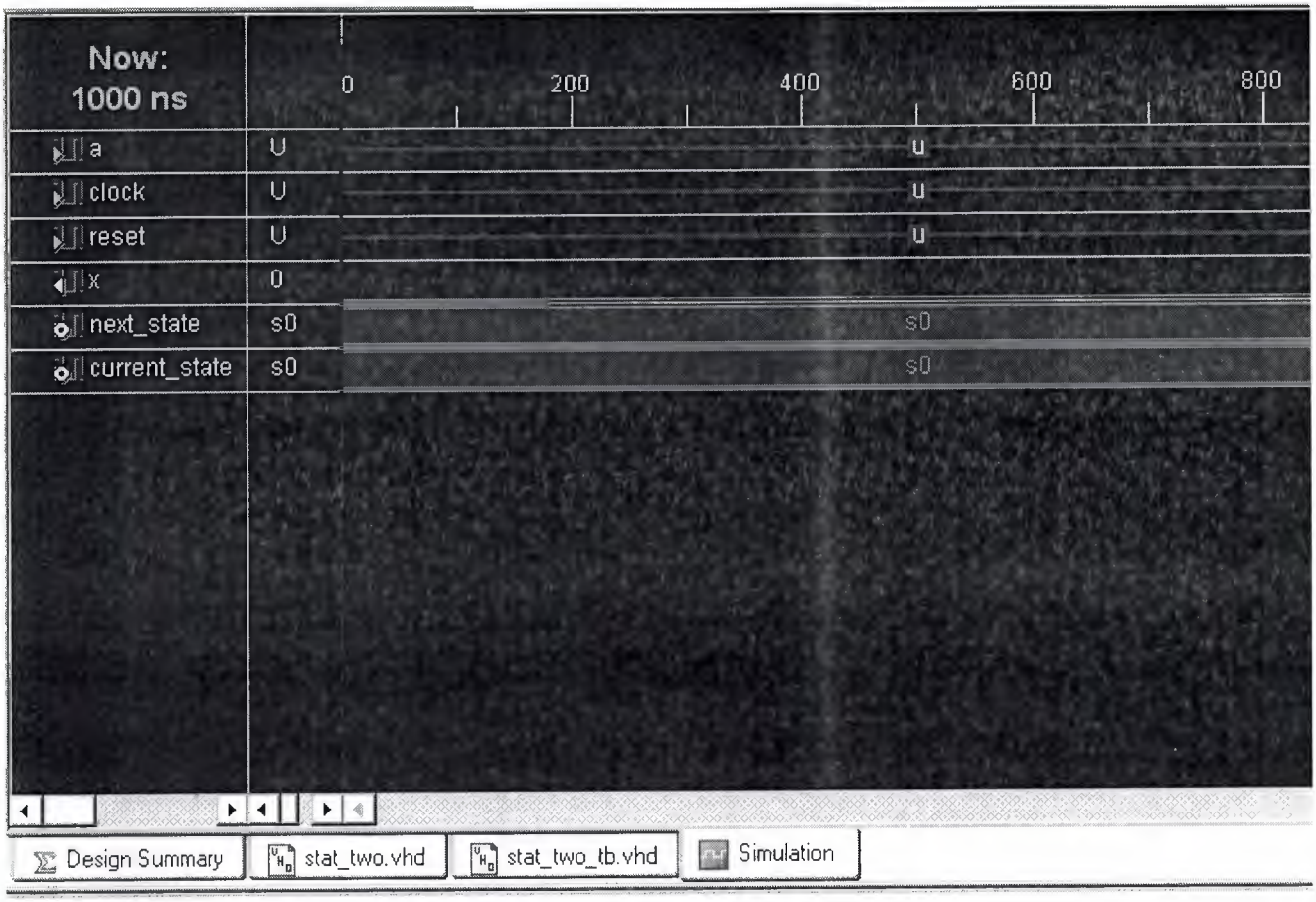

```
----------------------------------------------------------------------------------
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:    13:39:12 08/14/2008
-- Design Name:
-- Module Name:    stat_two - Behavioral
-- Project Name:
-- Target Devices:
-- Tool versions:
-- Description:
--
-- Dependencies:
```

```
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity stat_two is
   Port ( a : in  STD_LOGIC;
          clock : in  STD_LOGIC;
          reset : in  STD_LOGIC;
          x : out  STD_LOGIC);
end stat_two;

architecture Behavioral of stat_two is

type state_type is (s0,s1);
signal next_state, current_state: state_type;
begin
        state_reg: process (clock, reset)
        begin

        if (reset='1') then
                        current_state <= s0;
                        elsif (clock'event and clock='1') then
                        current_state <= next_state;

                        end if;
                        end process;

                -- current process#2:combinational logic
                comb_logic:process(current_state, a)
                begin

                -- use case statement to show the
                --state trsnsistion
```

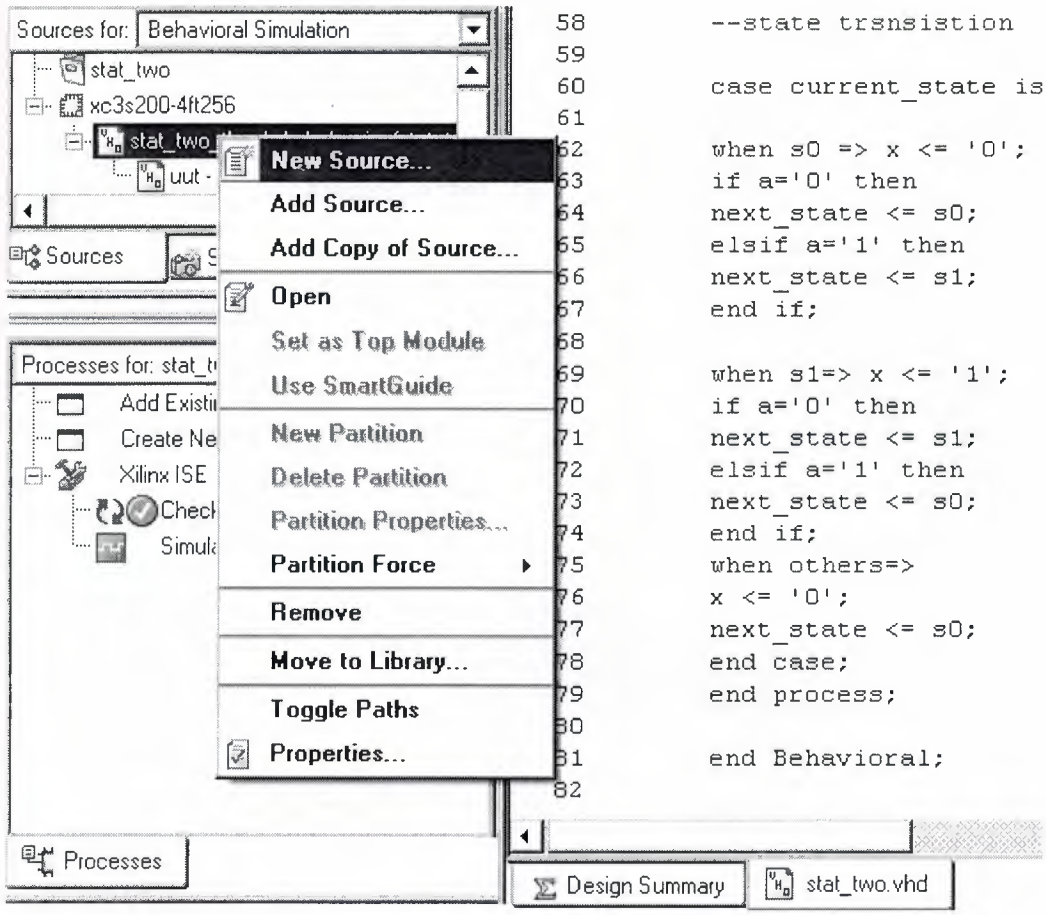
```
case current_state is

when s0 => x <= '0';
if a='0' then
next_state <= s0;
elsif a='1' then
next_state <= s1;
end if;

when s1=> x <= '1';
if a='0' then
next_state <= s1;
elsif a='1' then
next_state <= s0;
end if;
when others=>
x <= '0';
next_state <= s0;
end case;
end process;

end Behavioral;Synthesis Report:
```

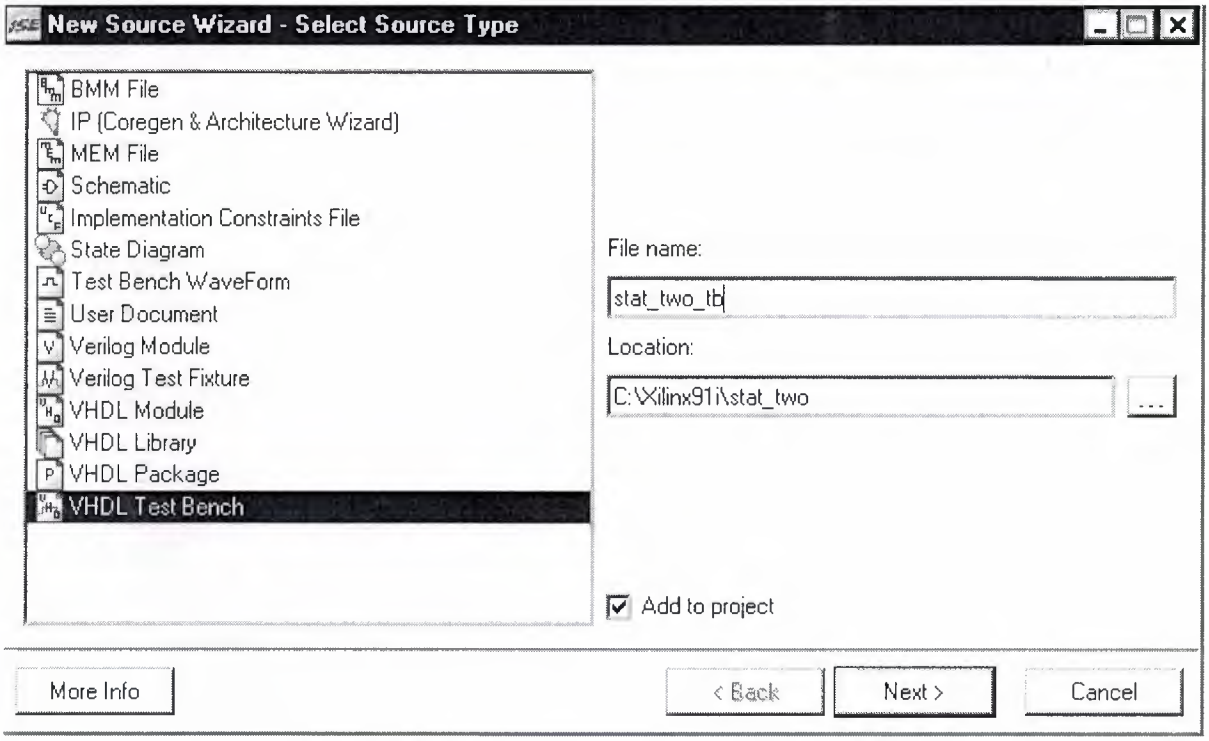
Design Simulation


Sources for: Behavioral Simulation
stat_two
xc3s200-4ft256
stat_two_tb_vhd - behavior (stat_t
uut - stat_two - Behavioral (sta
Sources
Snapshots
Libraries
Processes for: stat_two_tb_vhd - behavior
Add Existing Source
Create New Source
Xilinx ISE Simulator
Check Syntax
Simulate Behavioral Model
Processes
end process;
-- current process#2:combinational log
comb_logic:process(current_state, a)
begin
-- use case statement to show the
--state trsnsistion
case current_state is
when s0 => x <= '0';
if a='0' then
next_state <= s0;
elsif a='1' then
next_state <= s1;
end if;
when s1=> x <= '1';
if a='0' then
next_state <= s1;
elsif a='1' then
next_state <= s0;
end if;
when others=>
x <= '0';
Design Summary
stat_two.vhd
stat_two_tb.vhd



CREATE THE TEST BENCH

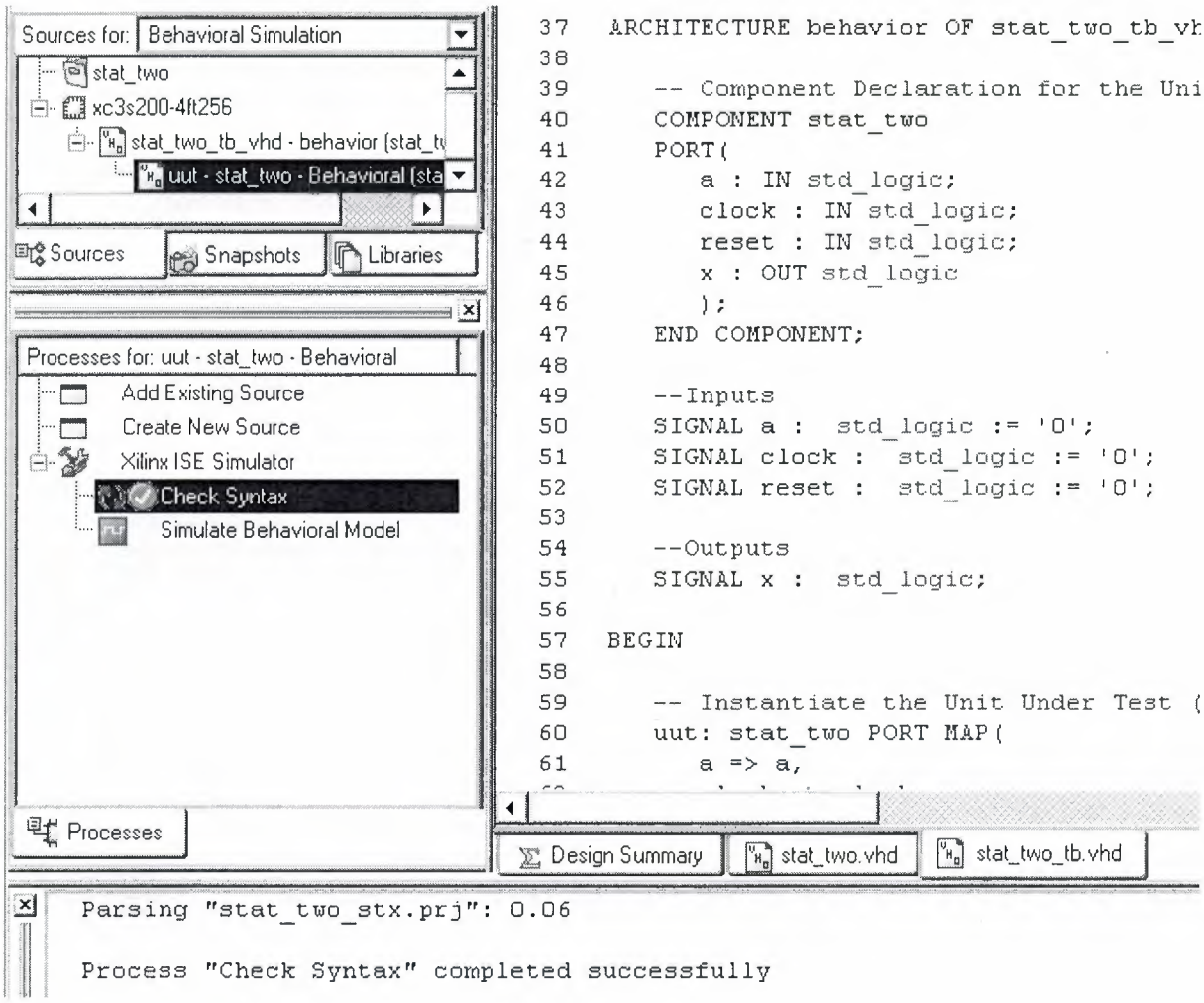

Sources for: Behavioral Simulation
stat_two
xc3s200-4ft256
uut -
New Source...
Add Source...
Add Copy of Source...
Open
Set as Top Module
Use SmartGuide
New Partition
Delete Partition
Partition Properties...
Partition Force
Remove
Move to Library...
Toggle Paths
Properties...
Sources
Processes for: stat_t
Add Existi
Create Ne
Xilinx ISE
Check
Simula
Processes
58 --state trsnsistion
59
60 case current_state is
61
62 when s0 => x <= '0';
63 if a='0' then
64 next_state <= s0;
65 elsif a='1' then
66 next_state <= s1;
67 end if;
68
69 when s1=> x <= '1';
70 if a='0' then
71 next_state <= s1;
72 elsif a='1' then
73 next_state <= s0;
74 end if;
75 when others=>
76 x <= '0';
77 next_state <= s0;
78 end case;
79 end process;
80
81 end Behavioral;
82
Design Summary
stat_two.vhd

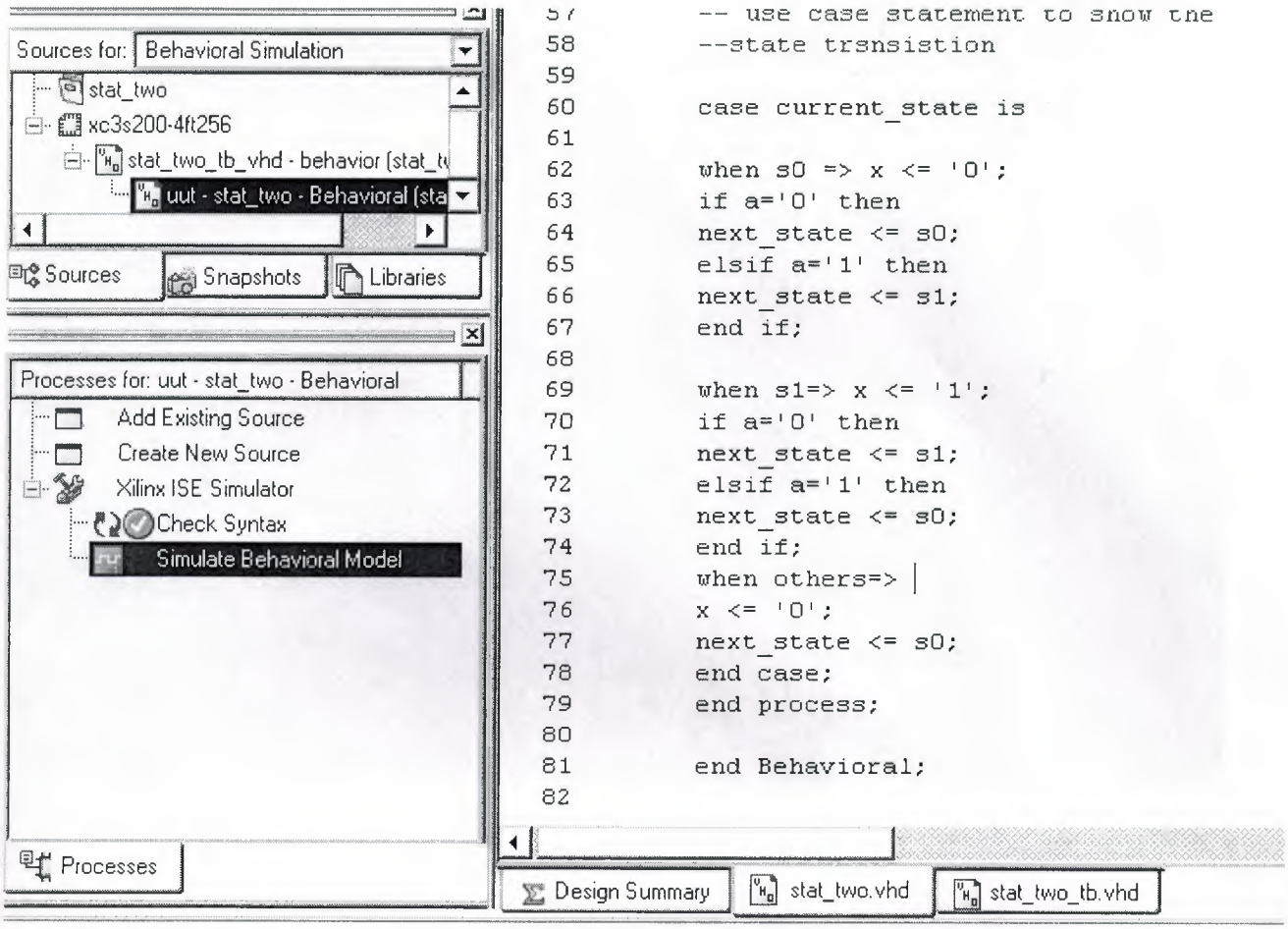

New Source Wizard - Select Source Type
BMM File
IP (Coregen & Architecture Wizard)
MEM File
Schematic
Implementation Constraints File
State Diagram
Test Bench WaveForm
User Document
Verilog Module
Verilog Test Fixture
VHDL Module
VHDL Library
VHDL Package
VHDL Test Bench
File name:
stat_two_tb
Location:
C:\Xilinx91i\stat_two
...
Add to project
More Info
< Back
Next >
Cancel

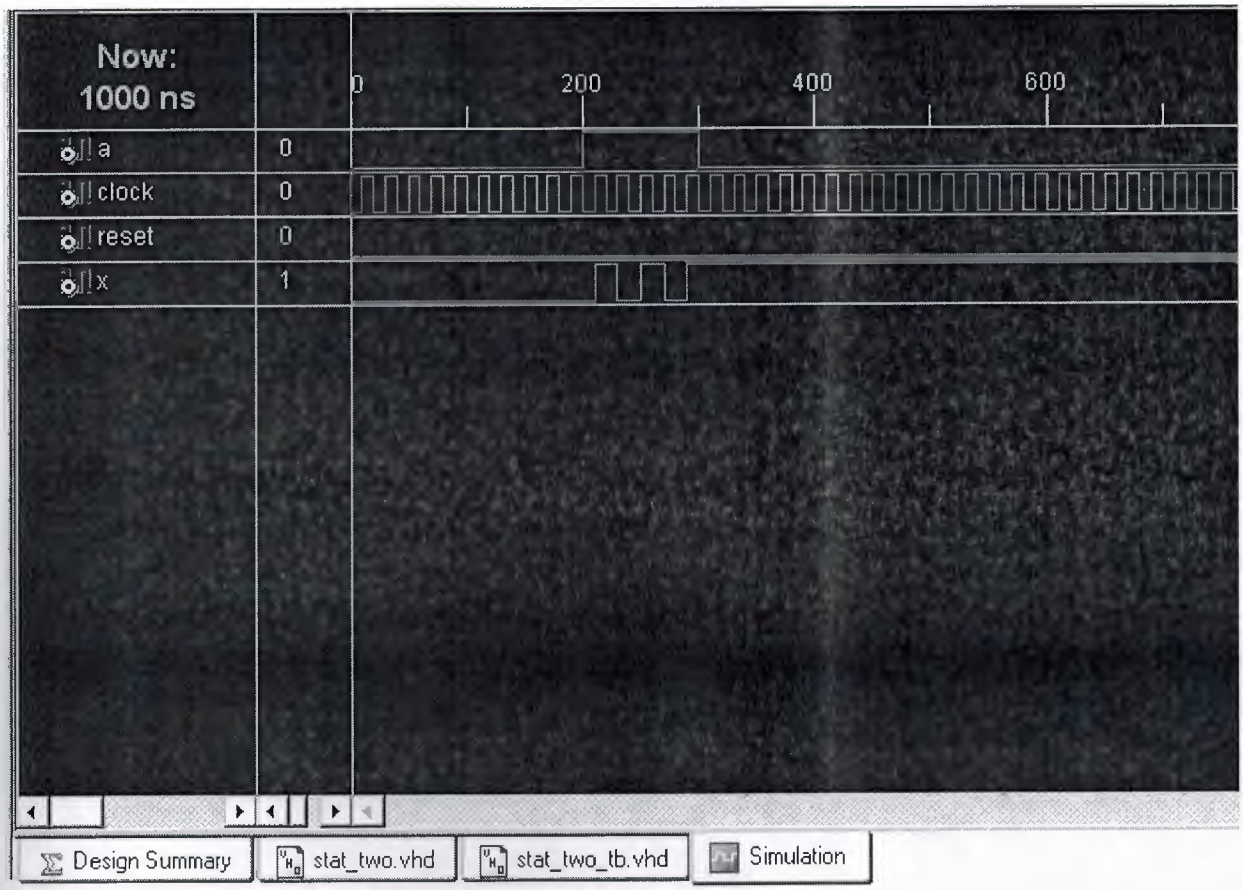
-- Company:

```
-- Engineer:
--
-- Create Date:   01:17:18 08/10/2008
-- Design Name:   stat_two
-- Module Name:   C:/Xilinx91i/stat_two/stat_two_tb.vhd
-- Project Name:  stat_two
-- Target Device:
-- Tool versions:
-- Description:
--
-- VHDL Test Bench Created by ISE for module: stat_two
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
-- Notes:
-- This testbench has been automatically generated using types std_logic and
-- std_logic_vector for the ports of the unit under test.  Xilinx recommends
-- that these types always be used for the top-level I/O of a design in order
-- to guarantee that the testbench will bind correctly to the post-implementation
-- simulation model.
--------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;

ENTITY stat_two_tb_vhd IS
END stat_two_tb_vhd;

ARCHITECTURE behavior OF stat_two_tb_vhd IS

        -- Component Declaration for the Unit Under Test (UUT)
        COMPONENT stat_two
        PORT(
                a : IN std_logic;
                clock : IN std_logic;
                reset : IN std_logic;
                x : OUT std_logic
                );
        END COMPONENT;
```

```
        --Inputs
        SIGNAL a :  std_logic := '0';
        SIGNAL clock :  std_logic := '0';
        SIGNAL reset :  std_logic := '0';

        --Outputs
        SIGNAL x :  std_logic;

BEGIN

        -- Instantiate the Unit Under Test (UUT)
        uut: stat_two PORT MAP(
                a => a,
                clock => clock,
                reset => reset,
                x => x
        );
clkp: process
  begin
        clock <= '0';
        wait for 10 ns;
        clock <= '1';
        wait for 10 ns;
        end process;


        tb : PROCESS
        BEGIN

                -- Wait 100 ns for global reset to finish
                wait for 100 ns;
                reset <='0';
                a<='0';
                wait for 100 ns;
                a<='1';
                wait for 100 ns;
                a<='0';

                -- Place stimulus here

                wait; -- will wait forever
        END PROCESS;

END;
```


Sources for: Behavioral Simulation
stat_two
xc3s200-4ft256
stat_two_tb_vhd - behavior (stat_t
uut - stat_two - Behavioral (sta
Sources
Snapshots
Libraries
Processes for: uut - stat_two - Behavioral
Add Existing Source
Create New Source
Xilinx ISE Simulator
Check Syntax
Simulate Behavioral Model
Processes
37 ARCHITECTURE behavior OF stat_two_tb_vh
38
39 -- Component Declaration for the Uni
40 COMPONENT stat_two
41 PORT(
42 a : IN std_logic;
43 clock : IN std_logic;
44 reset : IN std_logic;
45 x : OUT std_logic
46);
47 END COMPONENT;
48
49 --Inputs
50 SIGNAL a : std_logic := '0';
51 SIGNAL clock : std_logic := '0';
52 SIGNAL reset : std_logic := '0';
53
54 --Outputs
55 SIGNAL x : std_logic;
56
57 BEGIN
58
59 -- Instantiate the Unit Under Test (
60 uut: stat_two PORT MAP(
61 a => a,
Design Summary
stat_two.vhd
stat_two_tb.vhd
Parsing "stat_two_stx.prj": 0.06
Process "Check Syntax" completed successfully


Sources for: Behavioral Simulation
stat_two
xc3s200-4ft256
stat_two_tb_vhd - behavior (stat_t
uut - stat_two - Behavioral (sta
Sources
Snapshots
Libraries
Processes for: uut - stat_two - Behavioral
Add Existing Source
Create New Source
Xilinx ISE Simulator
Check Syntax
Simulate Behavioral Model
Processes
-- use case statement to show the
--state trsnsistion
case current_state is
when s0 => x <= '0';
if a='0' then
next_state <= s0;
elsif a='1' then
next_state <= s1;
end if;
when s1=> x <= '1';
if a='0' then
next_state <= s1;
elsif a='1' then
next_state <= s0;
end if;
when others=>
x <= '0';
next_state <= s0;
end case;
end process;
end Behavioral;
Design Summary
stat_two.vhd
stat_two_tb.vhd


Now:
1000 ns
0
200
400
600
a
0
clock
0
reset
0
x
1
Design Summary
stat_two.vhd
stat_two_tb.vhd
Simulation