National University of Singapore

Department of Electrical and Computer Engineering

EE2020: DIGITAL FUNDAMENTALS

A Step-by-Step Guide for VHDL/ Schematic Simulation Using Xilinx ISE WebPACK 13.1

VHDL Design and Simulation Using Xilinx ISE WebPACK 13.1:
This manual illustrates step-by-step instructions on how to use Xilinx's ISE WebPACK 13.1 software for design, simulation and synthesis of digital circuits using VHDL and schematic. Instructions on how to obtain the Xilinx software are detailed in the ***Xilinx 13.1 Installation*** manual.

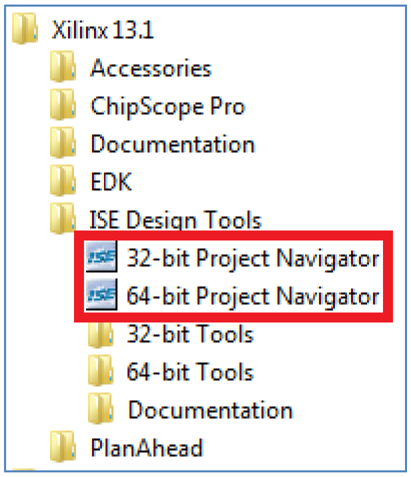
NOTE:
Save all your work in D:\MyWork. Any files saved on the hard drive of the computers in the lab will be cleared on a daily basis.
You should not bring your own storage device for D2 test.

The procedures described in this manual involve six major steps:

1. Creating a new VHDL project using Xilinx 13.1 ISE WebPACK.
2. Using the VHDL Editor for programming and syntax checking.
3. Simulating the design using VHDL Test Bench and ISE Simulator for combinational circuits.
4. Simulating the design using VHDL Test Bench and ISE Simulator for clocked circuits.
5. Simulating a digital logic circuit using schematic source. (might be helpful for D1 project)
6. FPGA implementation of a VHDL design. (Optional)

Steps 1, 2, 3 and 6 describes the VHDL implementation of a 1-bit half adder circuit.
Step 4 describes the VHDL implementation of a D-flip-flop.
Step 5 describes the schematic implementation of a multiplexer-based circuit.

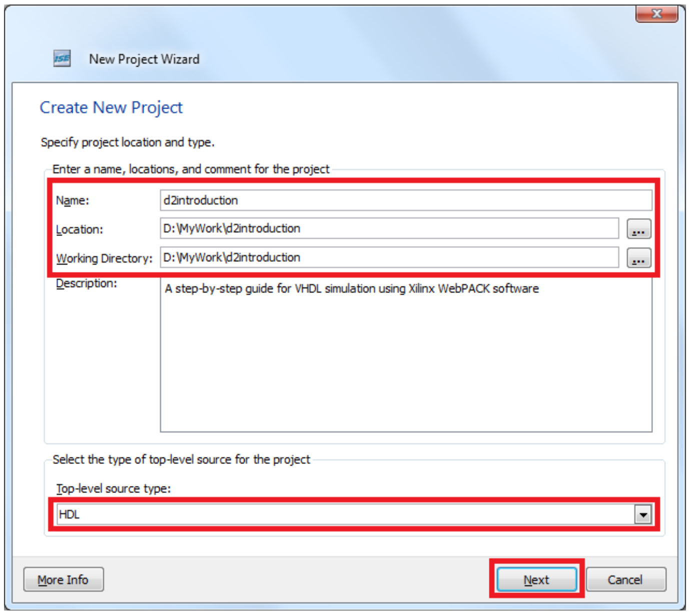
1. Xilinx Project Manager: Creating a New VHDL Project



1. To launch the Xilinx ISE 13.1 software package, either:

   (i) Double click the ***Xilinx 13.1*** icon on your desktop
   or
   (ii) Select ***Start Menu → All Programs → Xilinx ISE Design Suite 13.1 → ISE Design Tools → 32-bit Project Navigator***.

   By default, 64-bit operating systems will use the 64-bit Project Navigator when the desktop icon is double clicked. Either of the 32-bit Project Navigator or 64-bit Project Navigator can be used depending on your preference.


New Project Wizard
Create New Project
Specify project location and type.
Enter a name, locations, and comment for the project

Name: d2introduction
Location: D:\MyWork\d2introduction
Working Directory: D:\MyWork\d2introduction
Description: A step-by-step guide for VHDL simulation using Xilinx WebPACK software

Select the type of top-level source for the project

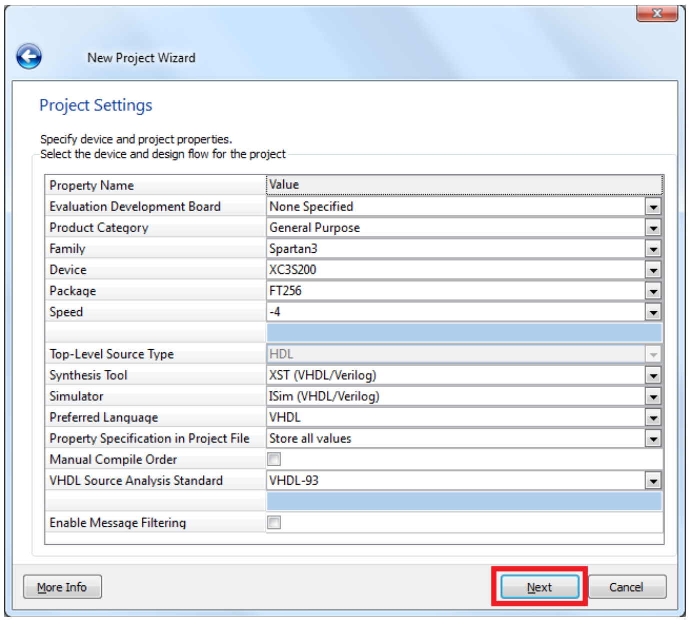
Top-level source type: HDL

2. After launching the ***Project Navigator***, select ***File → New Project***.

   The ***Create New Project*** window will appear.

   (i) Specify a ***Name*** for the project.
   Example: ***d2introduction***
   (ii) Specify a ***Location*** for the project.
   Example: ***D:\MyWork\d2introduction***
   (iii) Specify the ***Working Directory*** of the project.
   Example: ***D:\MyWork\d2introduction***
   (iv) Select ***HDL*** as the Top-level source type.

   Click ***Next*** to continue.



Project Settings

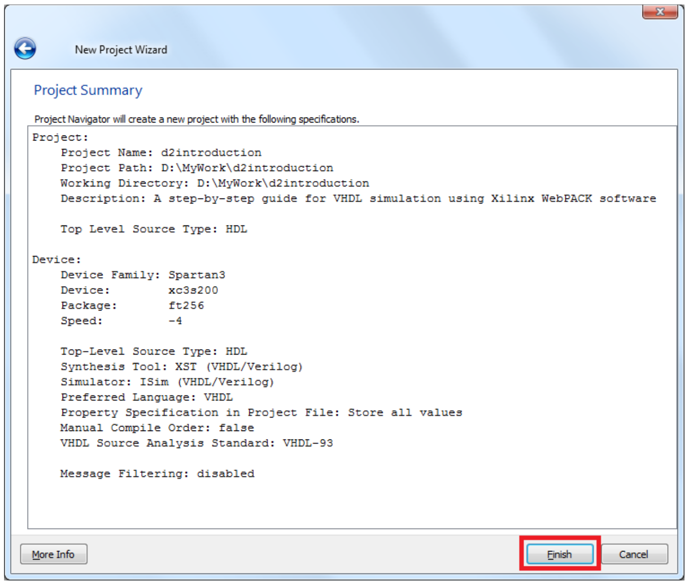
Specify device and project properties.
Select the device and design flow for the project

Property Name	Value
Evaluation Development Board	None Specified
Product Category	General Purpose
Family	Spartan3
Device	XC3S200
Package	FT256
Speed	-4
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	VHDL
Property Specification in Project File	Store all values
Manual Compile Order	☐
VHDL Source Analysis Standard	VHDL-93
Enable Message Filtering	☐

3. The ***Project Settings*** window will appear. Set the following:

(i) Evaluation Development Board: ***None Specified***
(ii) Product Category: ***General Purpose***
(iii) Family: ***Spartan3***
(iv) Device: ***XC3S200***
(v) Package: ***FT256***
(vi) Speed: ***-4***
(vii) Synthesis Tool: ***XST (VHDL/Verilog)***
(viii) Simulator: ***ISim (VHDL/Verilog)***
(ix) Preferred Language: ***VHDL***
(x) Property Specification in Project File: ***Store all values***
(xi) Manual Compile Order: ***Unchecked***
(xii) VHDL Source Analysis Standard: ***VHDL-93***
(xiii) Enable Message Filtering: ***Unchecked***

Click on ***Next***.

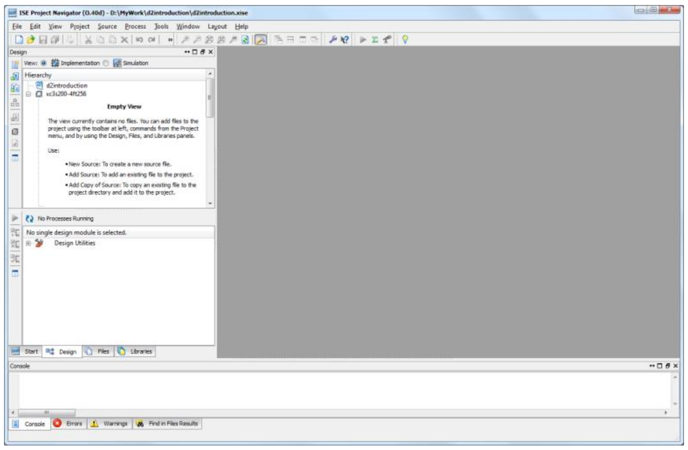


Project Summary

Project Navigator will create a new project with the following specifications.

```
Project:
    Project Name: d2introduction
    Project Path: D:\MyWork\d2introduction
    Working Directory: D:\MyWork\d2introduction
    Description: A step-by-step guide for VHDL simulation using Xilinx WebPACK software

    Top Level Source Type: HDL

Device:
    Device Family: Spartan3
    Device:        xc3s200
    Package:       ft256
    Speed:         -4

    Top-Level Source Type: HDL
    Synthesis Tool: XST (VHDL/Verilog)
    Simulator: ISim (VHDL/Verilog)
    Preferred Language: VHDL
    Property Specification in Project File: Store all values
    Manual Compile Order: false
    VHDL Source Analysis Standard: VHDL-93

    Message Filtering: disabled
```

4. Click on ***Finish*** to continue to the ISE Workspace.



Empty View

The view currently contains no files. You can add files to the project using the toolbar at left, commands from the Project menu, and by using the Design, Files, and Libraries panels.

Use:

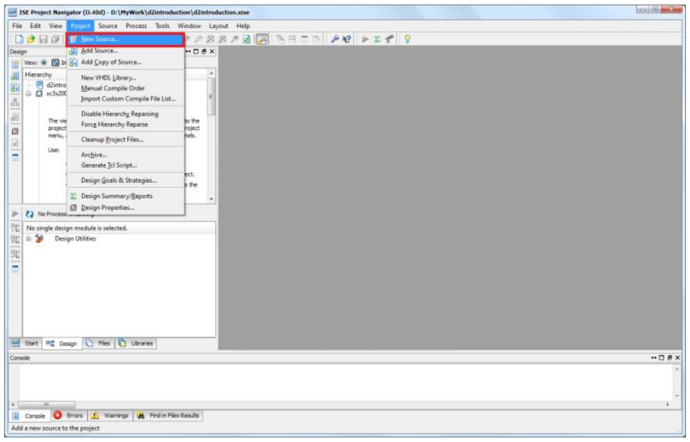
- New Source: To create a new source file.
- Add Source: To add an existing file to the project.
- Add Copy of Source: To copy an existing file to the project directory and add it to the project.

No single design module is selected.

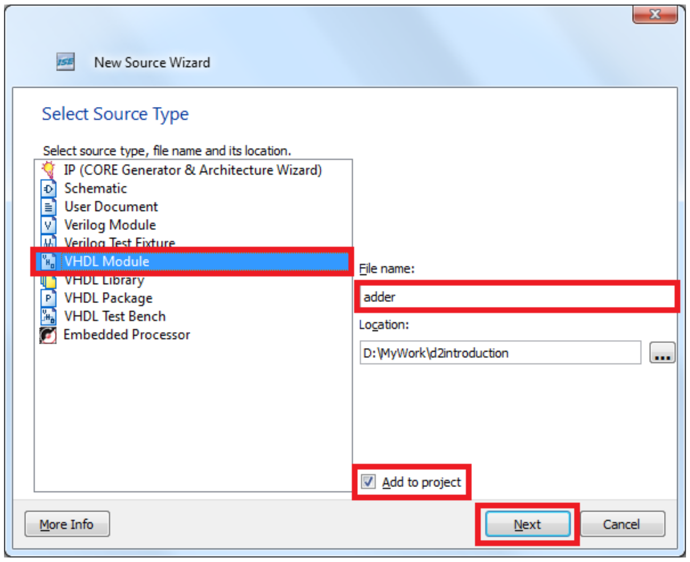
5. The ***ISE Workspace*** window should appear after the new project is created. It is the central place from where all aspects of the design/simulation process are controlled. The ***ISE Workspace*** main window is divided into four sub-windows:

(i) (Upper left) The ***Design*** window displays the files included in the project.
(ii) (Middle left) The ***Processes for Current Source*** window displays available processes.
(iii) (Bottom) The ***Console*** window displays status messages, errors, and warnings and is updated during all project actions.
(iv) (Grey window on the right) The ***VHDL Editor*** window enables editing of source files and access to the VHDL language templates catalogue which provides basic sample codes. The **Language Templates** can be accessed by selecting ***Edit → Language Templates...***

2. Xilinx VHDL Editor: Programming and Syntax Checking


ISE Project Navigator (O.40d) - D:\MyWork\d2introduction\d2introduction.xise
File Edit View Project Source Process Tools Window Layout Help
New Source...
Add Source...
Add Copy of Source...
New VHDL Library...
Manual Compile Order
Import Custom Compile File List...
Disable Hierarchy Reparsing
Force Hierarchy Reparse
Cleanup Project Files...
Archive...
Generate Tcl Script...
Design Goals & Strategies...
Design Summary/Reports
Design Properties...
No single design module is selected.
Design Utilities
Console Errors Warnings Find in Files Results
Add a new source to the project

1. To create a new VHDL program, in ISE Project Navigator, select ***Project → New Source***.



File name: adder

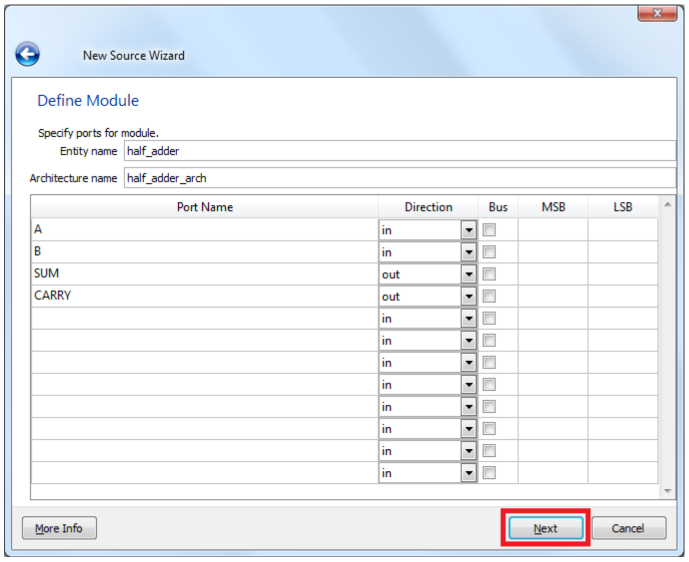
Location: D:\MyWork\d2introduction

☑ Add to project

2. A New Source Wizard window appears. In this ***Select Source Type*** window:

   (i) Select ***VHDL Module***.
   (ii) In the ***File name*** field, type the VHDL file name.
   Example: ***adder***
   (iii) Ensure ***Add to project*** is checked.

   Click ***Next*** to continue.



Entity name: half_adder

Architecture name: half_adder_arch

Port Name	Direction	Bus	MSB	LSB
A	in	☐		
B	in	☐		
SUM	out	☐		
CARRY	out	☐		
	in	☐		
	in	☐		
	in	☐		
	in	☐		
	in	☐		
	in	☐		
	in	☐		
	in	☐		

3. The ports are defined in this ***Define Module*** window

   (i) Enter an ***Entity name***.
   Example: ***half_adder***
   (ii) Enter an ***Architecture name***.
   Example: ***half_adder_arch***

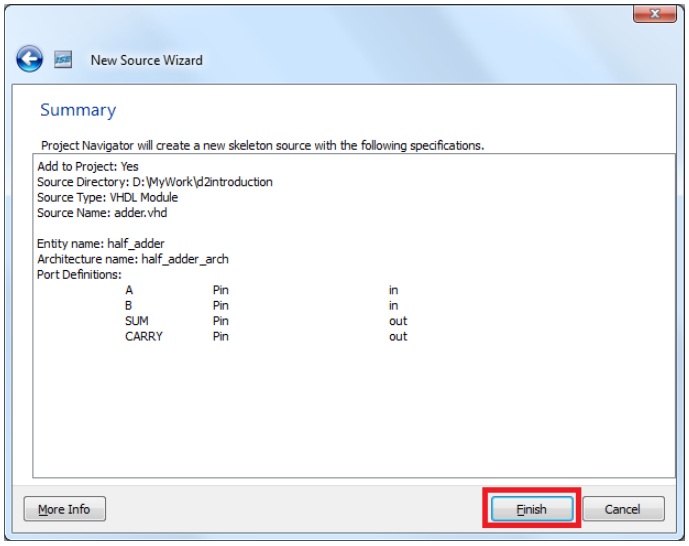
   Click in the ***Port Name*** field to enter a value and select the corresponding direction in the ***Direction*** drop down menu. Enter the following values:

   (iv) Port Name: ***A***, Direction: ***in***.
   (v) Port Name: ***B***, Direction: ***in***.
   (vi) Port Name: ***SUM***, Direction: ***out***.
   (vii)Port Name: ***CARRY***, Direction: ***out***.

   The half adder has two 1-bit inputs and two 1-bit outputs, as defined above.

   Click on ***Next*** to proceed.

4. A description of the module summary is shown. Click ***Finish*** to complete the New Source Wizard.



Summary

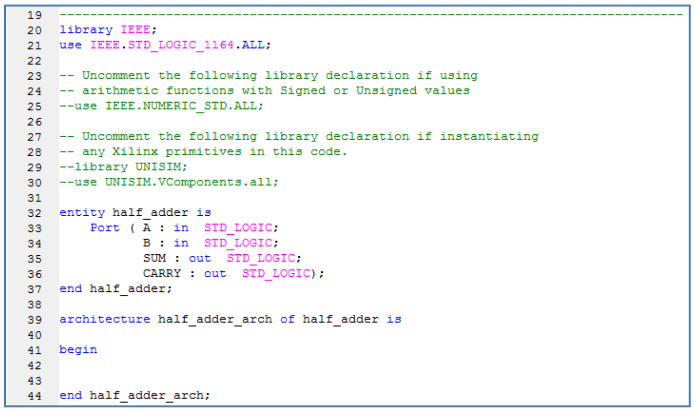
Project Navigator will create a new skeleton source with the following specifications.

Add to Project: Yes
Source Directory: D:\MyWork\d2introduction
Source Type: VHDL Module
Source Name: adder.vhd

Entity name: half_adder
Architecture name: half_adder_arch
Port Definitions:

A	Pin	in
B	Pin	in
SUM	Pin	out
CARRY	Pin	out

5. A VHDL file template is created in the ***VHDL Editor*** window.

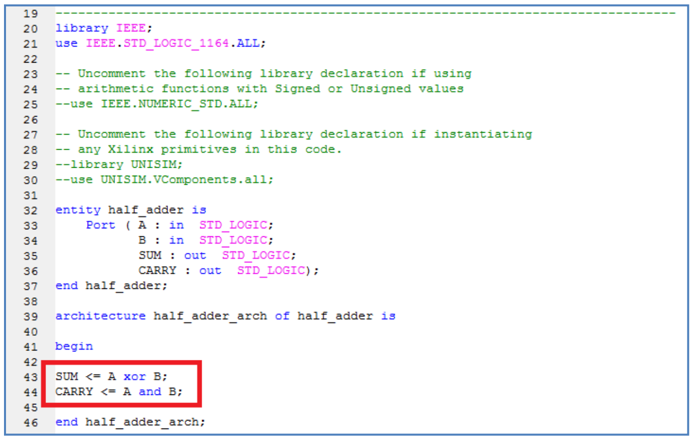


```
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

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

entity half_adder is
    Port ( A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           SUM : out  STD_LOGIC;
           CARRY : out  STD_LOGIC);
end half_adder;

architecture half_adder_arch of half_adder is

begin


end half_adder_arch;
```

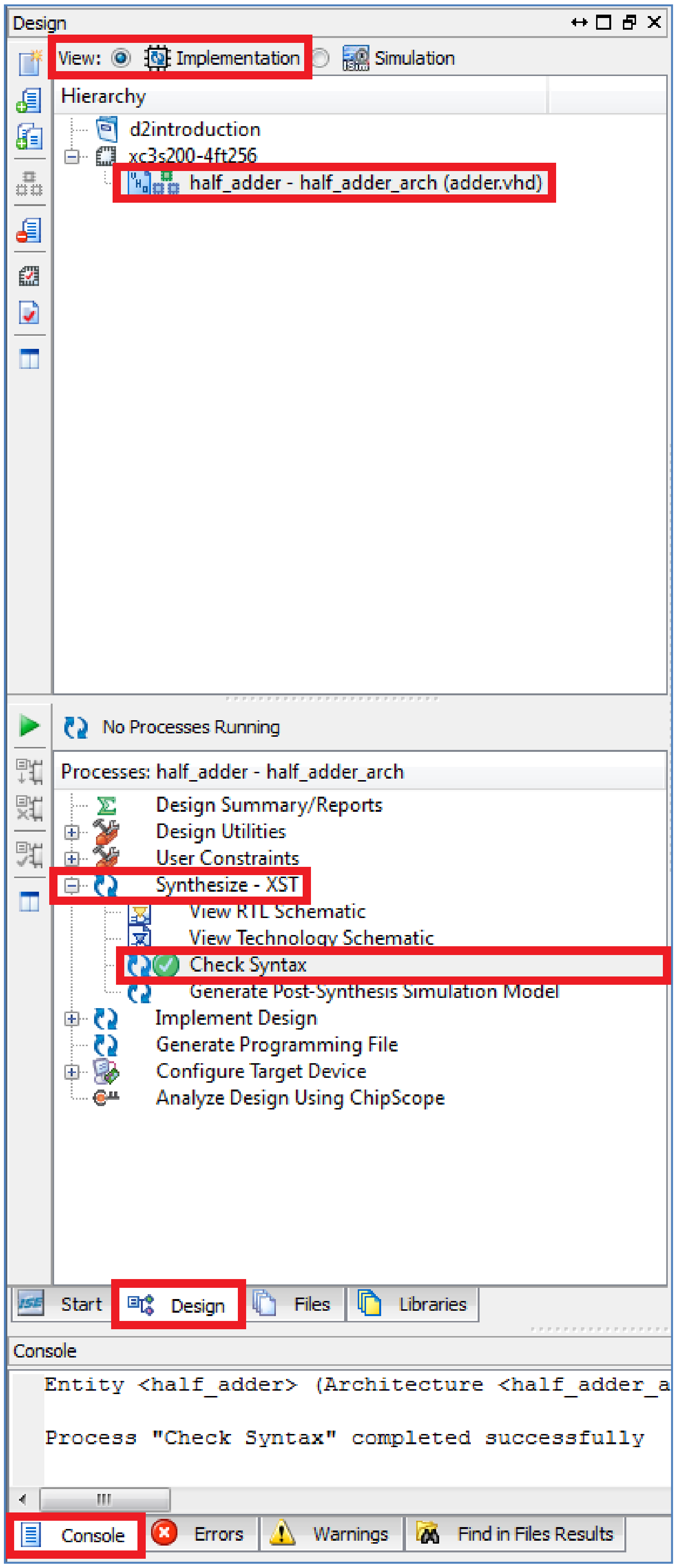
6. Add the following lines within the architecture:

SUM <= A xor B;
CARRY <= A and B;

The VHDL code is now complete and functional.



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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

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

entity half_adder is
    Port ( A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           SUM : out  STD_LOGIC;
           CARRY : out  STD_LOGIC);
end half_adder;

architecture half_adder_arch of half_adder is

begin

SUM <= A xor B;
CARRY <= A and B;

end half_adder_arch;
```



```
Entity <half_adder> (Architecture <half_adder_a

Process "Check Syntax" completed successfully
```

7. Save the project by selecting ***File → Save***.

Ensure the following:

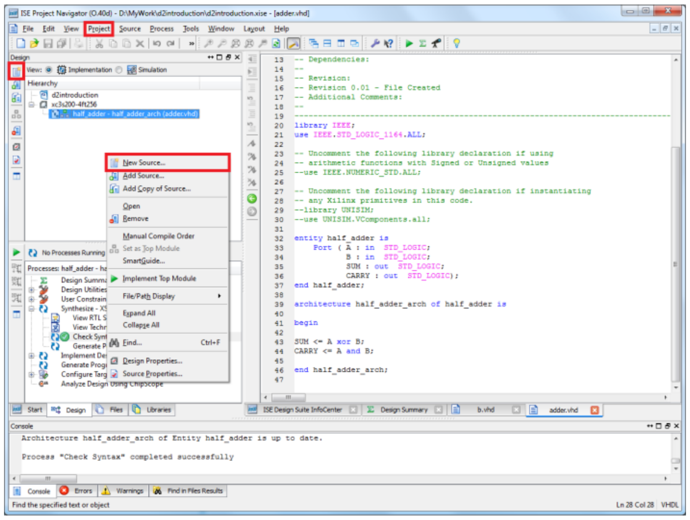
(i) The ***Design*** tab is selected.
(ii) ***View*** is set to ***Implementation***.
(iii) ***half_adder - half_adder_arch (adder.vhd)*** is selected.
(iv) ***Synthesize - XST*** is expanded.

Double click ***Check Syntax*** to verify the VHDL code syntax.

A tick in a green circle next to ***Check Syntax*** indicates that there are no syntax errors and that the VDHL program is ready for functional simulation.

3. Behavioural Simulation: Using VHDL Test Bench for Combinational Circuits

Behavioural simulation is performed before actual design implementation on hardware to verify that the logic created is correct.

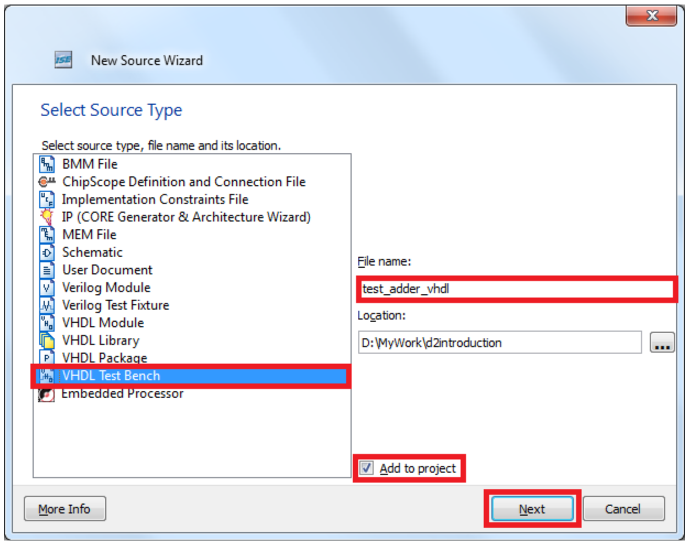

ISE Project Navigator (O.40d) - D:\MyWork\d2introduction\d2introduction.xise - [adder.vhd]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
d2introduction
xc3s200-4ft256
half_adder - half_adder_arch (adder.vhd)
New Source...
Add Source...
Add Copy of Source...
Open
Remove
Manual Compile Order
Set as Top Module
SmartGuide...
Implement Top Module
File/Path Display
Expand All
Collapse All
Find... Ctrl+F
Design Properties...
Source Properties...
No Processes Running
Processes: half_adder - half_adder_arch
Design Summary
Design Utilities
User Constraints
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation Model
Implement Design
Generate Programming File
Configure Target Device
Analyze Design Using ChipScope
Start Design Files Libraries
ISE Design Suite InfoCenter Design Summary b.vhd adder.vhd
Console Errors Warnings Find in Files Results
Find the specified text or object
Ln 28 Col 28 VHDL

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

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

entity half_adder is
    Port ( A : in  STD_LOGIC;
           B : in  STD_LOGIC;
           SUM : out  STD_LOGIC;
           CARRY : out  STD_LOGIC);
end half_adder;

architecture half_adder_arch of half_adder is

begin

SUM <= A xor B;
CARRY <= A and B;

end half_adder_arch;
```

```
Architecture half_adder_arch of Entity half_adder is up to date.

Process "Check Syntax" completed successfully
```

1. Add a ***New Source*** to the project. This can be done by:

   (i) Right clicking in the design window and selecting ***New Source*** or
   (ii) Selecting ***Project → New Source*** or
   (iii) Clicking on the ***New Source*** shortcut icon.



Select source type, file name and its location.

File name: test_adder_vhdl

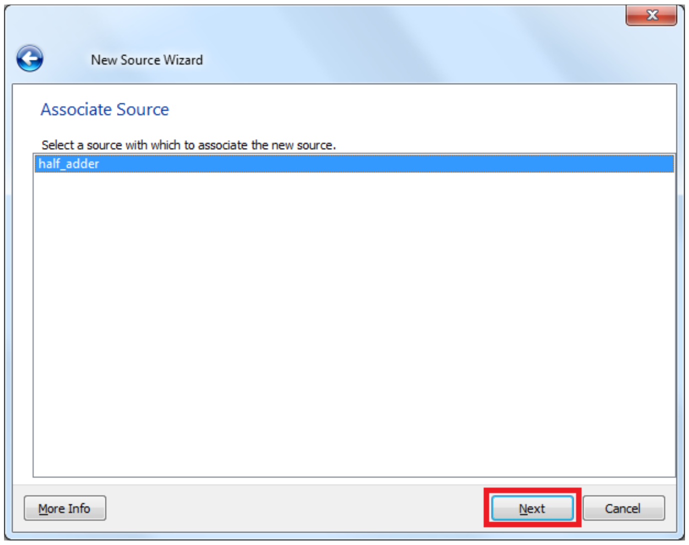
Location: D:\MyWork\d2introduction

☑ Add to project

2. A New Source Wizard window appears. In this ***Select Source Type*** window

   (i) Select ***VHDL Test Bench***.
   (ii) In the ***File name*** field, type the file name. Example: ***test_adder_vhdl***
   (iii) Ensure ***Add to project*** is checked.

   Click ***Next*** to continue.

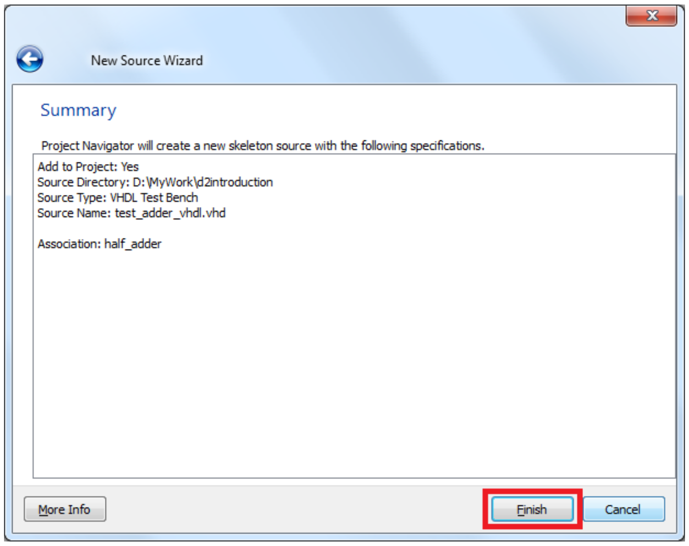


Select a source with which to associate the new source.

half_adder

3. Make sure that it is associated with the source that needs to be tested. In this case, ***half_adder*** as defined in step 3 of section 2, is to be tested.

   Click ***Next*** to proceed.



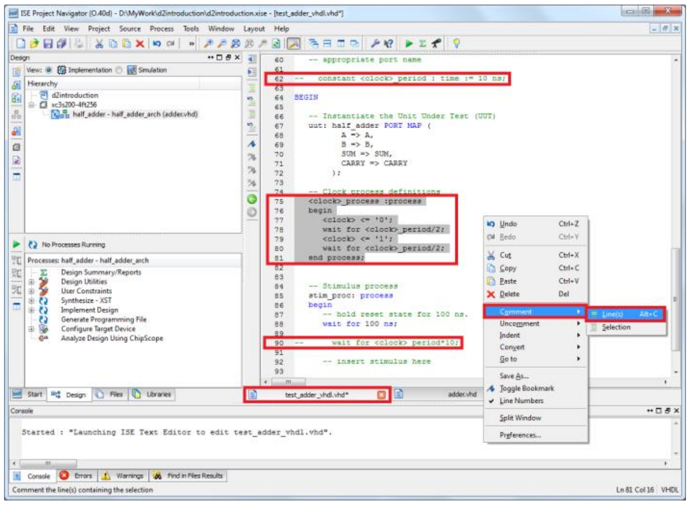
Summary

Project Navigator will create a new skeleton source with the following specifications.

Add to Project: Yes
Source Directory: D:\MyWork\d2introduction
Source Type: VHDL Test Bench
Source Name: test_adder_vhdl.vhd

Association: half_adder

4. Click ***Finish*** after reviewing the Summary window.



```
   -- appropriate port name

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

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: half_adder PORT MAP (
          A => A,
          B => B,
          SUM => SUM,
          CARRY => CARRY
        );

   -- 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
```

```
Started : "Launching ISE Text Editor to edit test_adder_vhdl.vhd".
```

Comment the line(s) containing the selection

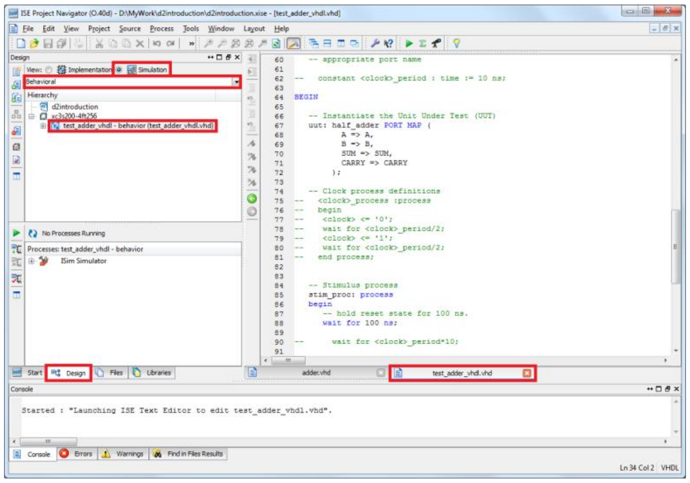
5. The test bench source opens up in a new tab. If a ***combinational circuit*** is being tested (like half_adder, which does not have a clock), comment out all lines which mention about the clock. Otherwise, syntax errors will occur.

   For this case, the lines to be commented are:

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

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

   wait for <clock>_period*10;

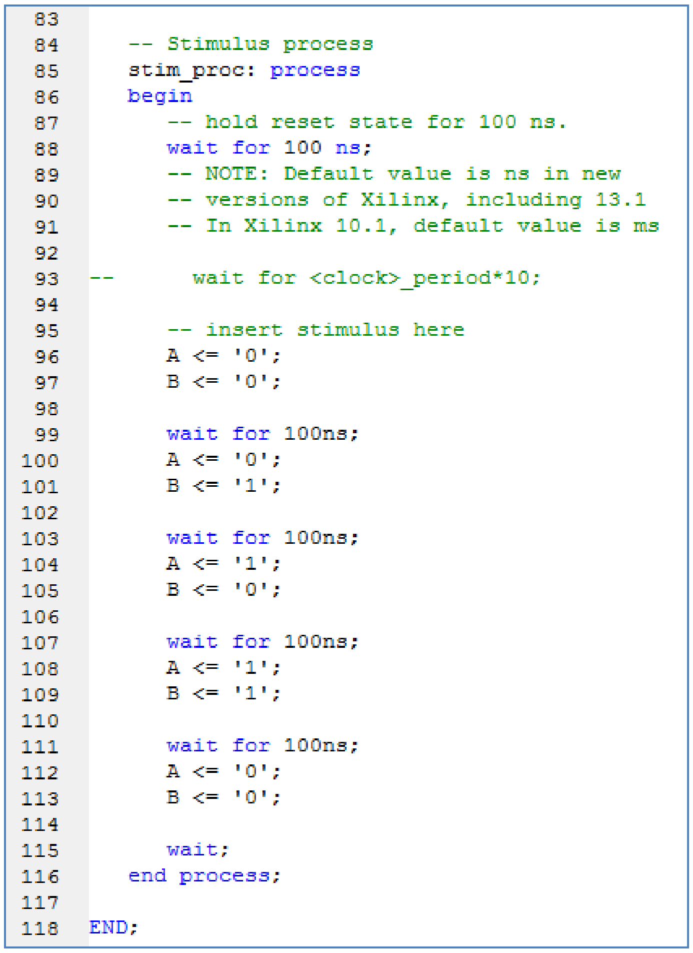

ISE Project Navigator (O.40d) - D:\MyWork\d2introduction\d2introduction.xise - [test_adder_vhdl.vhd]
View: Implementation Simulation
Behavioral
Hierarchy
d2introduction
xc3s200-4ft256
test_adder_vhdl - behavior (test_adder_vhdl.vhd)
No Processes Running
Processes: test_adder_vhdl - behavior
ISim Simulator
Start Design Files Libraries
adder.vhd
test_adder_vhdl.vhd

```
   -- appropriate port name

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

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: half_adder PORT MAP (
          A => A,
          B => B,
          SUM => SUM,
          CARRY => CARRY
        );

   -- 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;
```

```
Started : "Launching ISE Text Editor to edit test_adder_vhdl.vhd".
```

6. This step is an additional note to the previous step. If the test bench source has been created but closed, perform the following to open the test bench source again:

   (i) Click on the ***Design*** tab.
   (ii) In View, select ***Simulation***.
   (iii) Double click on
   test_adder_vhdl - behaviour (test_adder_vhdl.vhd).
   (iv) Select the ***test_adder_vhdl.vhd*** tab.



```
    -- Stimulus process
    stim_proc: process
    begin
       -- hold reset state for 100 ns.
       wait for 100 ns;
       -- NOTE: Default value is ns in new
       -- versions of Xilinx, including 13.1
       -- In Xilinx 10.1, default value is ms

--        wait for <clock>_period*10;

       -- insert stimulus here
       A <= '0';
       B <= '0';

       wait for 100ns;
       A <= '0';
       B <= '1';

       wait for 100ns;
       A <= '1';
       B <= '0';

       wait for 100ns;
       A <= '1';
       B <= '1';

       wait for 100ns;
       A <= '0';
       B <= '0';

       wait;
    end process;

END;
```

7. The Stimulus process indicates the input to the design. The process should begin with a ***wait for 100ns***, end with a ***wait***, and having appropriate stimuli in between.

   Note that for Xilinx 10.1, the default value of ***ms*** should be changed to ***ns***. In newer versions, including Xilinx 13.1, there is no need to change the time to ***ns*** as the default value is already in ***ns***.

   As shown in the figure, add the following stimuli to the VHDL test bench code:

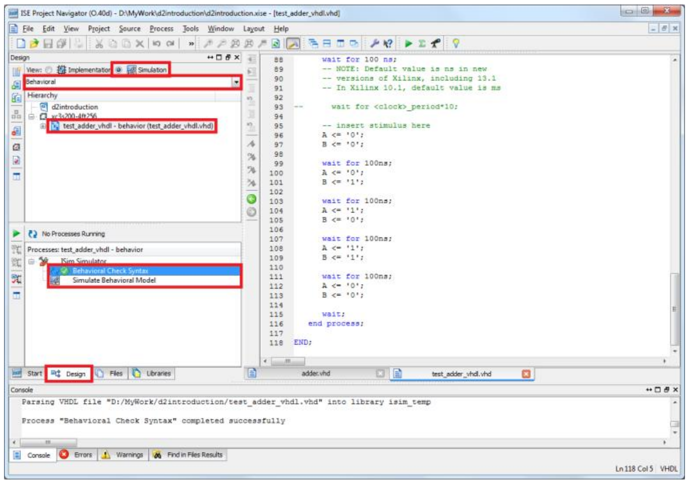
   A <= '0';
   B <= '0';

   wait for 100ns;
   A <= '0';
   B <= '1';

   wait for 100ns;
   A <= '1';
   B <= '0';

   wait for 100ns;
   A <= '1';
   B <= '1';

   wait for 100ns;
   A <= '0';
   B <= '0';



ISE Project Navigator (O.40d) - D:\MyWork\d2introduction\d2introduction.xise - [test_adder_vhdl.vhd]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Behavioral
Hierarchy
d2introduction
xc3s200-4ft256
test_adder_vhdl - behavior (test_adder_vhdl.vhd)
No Processes Running
Processes: test_adder_vhdl - behavior
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model
Start Design Files Libraries
adder.vhd
test_adder_vhdl.vhd
Console
Console Errors Warnings Find in Files Results
Ln 118 Col 5 VHDL

```
       wait for 100 ns;
       -- NOTE: Default value is ns in new
       -- versions of Xilinx, including 13.1
       -- In Xilinx 10.1, default value is ms

--        wait for <clock>_period*10;

       -- insert stimulus here
       A <= '0';
       B <= '0';

       wait for 100ns;
       A <= '0';
       B <= '1';

       wait for 100ns;
       A <= '1';
       B <= '0';

       wait for 100ns;
       A <= '1';
       B <= '1';

       wait for 100ns;
       A <= '0';
       B <= '0';

       wait;
    end process;

END;
```

```
Parsing VHDL file "D:/MyWork/d2introduction/test_adder_vhdl.vhd" into library isim_temp

Process "Behavioral Check Syntax" completed successfully
```

8. Save the project by selecting ***File → Save***.

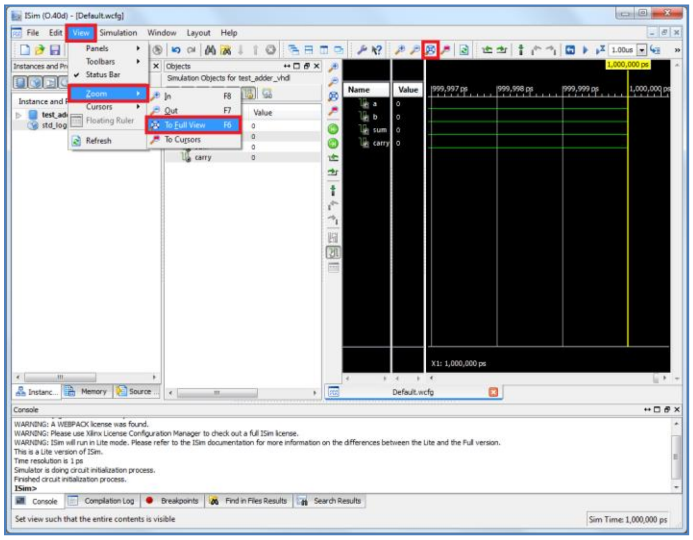
   Ensure the following:

   (i) The ***Design*** tab is selected.
   (ii) ***View*** is set to ***Simulation***.
   (iii) ***test_adder_vhdl - behaviour (test_adder_vhdl.vhd)*** is selected.
   (iv) ***ISim Simulator*** is expanded.

   Double click ***Behavioral Check Syntax*** to verify the VHDL code syntax

   A tick in a green circle next to ***Check Syntax*** indicates that there are no syntax errors and that the VDHL program is ready for functional simulation.

   Double click S***imulate Behavioral Model*** if the Behavioral Check Syntax is correct, else verify the VHDL codes if not.



```
WARNING: A WEBPACK license was found.
WARNING: Please use Xilinx License Configuration Manager to check out a full ISim license.
WARNING: ISim will run in Lite mode. Please refer to the ISim documentation for more information on the differences between the Lite and the Full version.
This is a Lite version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
```

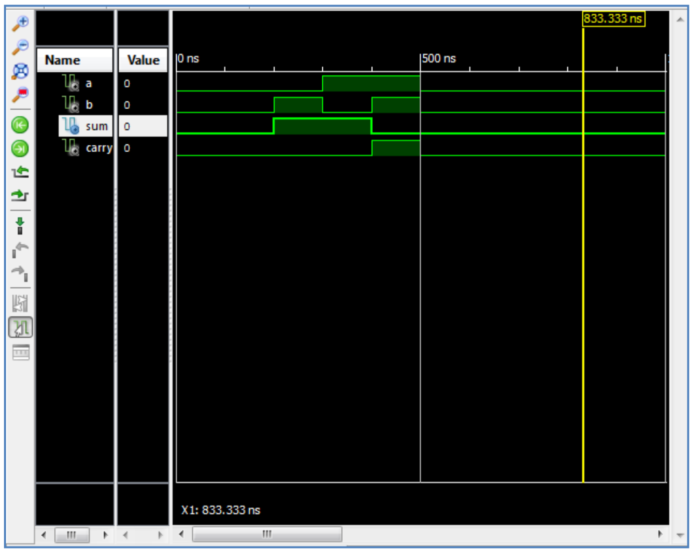
Set view such that the entire contents is visible

Sim Time: 1,000,000 ps

9. The simulation window will open up. The time resolution might be too high. In this case, the resolution is in ***ps***.

   To adjust the view ***To Full View***, do one of the following:
   (i) Select ***View → Zoom → To Full View*** or
   (ii) Click on the ***Zoom to Full View*** icon or
   (iii) Press the keyboard shortcut button ***F6***.

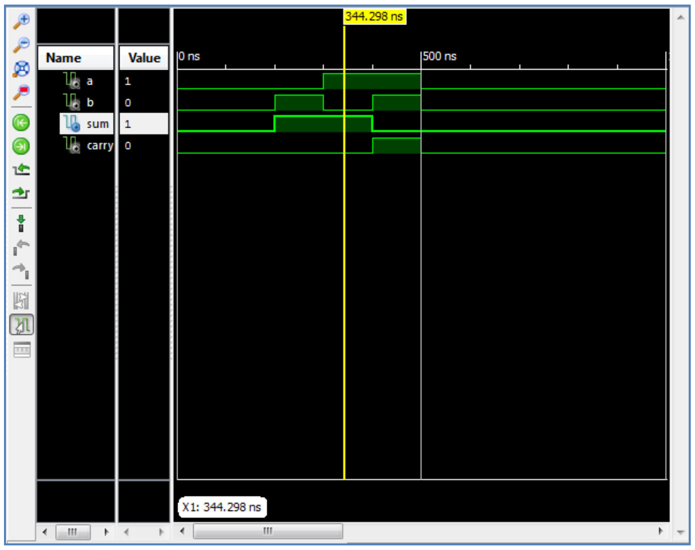
   If the ***Zoom to Full View*** option is not available, click anywhere in the black portion of the timescale window and try again.



10. In ***Full View***, the simulation window shows the full view of the simulation.

    a and ***b*** are the stimuli value, in other words, the input.
    sum and ***carry*** are the results of the stimuli, in other words, the output.

    All these four names and their behaviors were previously defined in steps 3 to 6 of section 2.



11. The yellow vertical line can be clicked on and moved around to get the ***Value*** of the ***Names*** at a particular time instant.

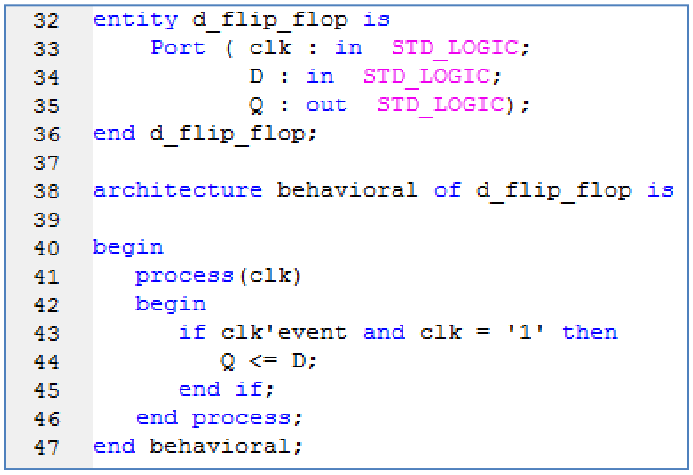
    Here, the window shows that at time instant 344.298 ns, the values of the ***Names*** are:

    a = 1
    b = 0
    sum = 1
    carry = 0

    Exit the application, or save if required. If the results are not as expected, modify the stimuli process and perform the simulation again.

4. Behavioural Simulation: Using VHDL Test Bench for Clocked Circuits

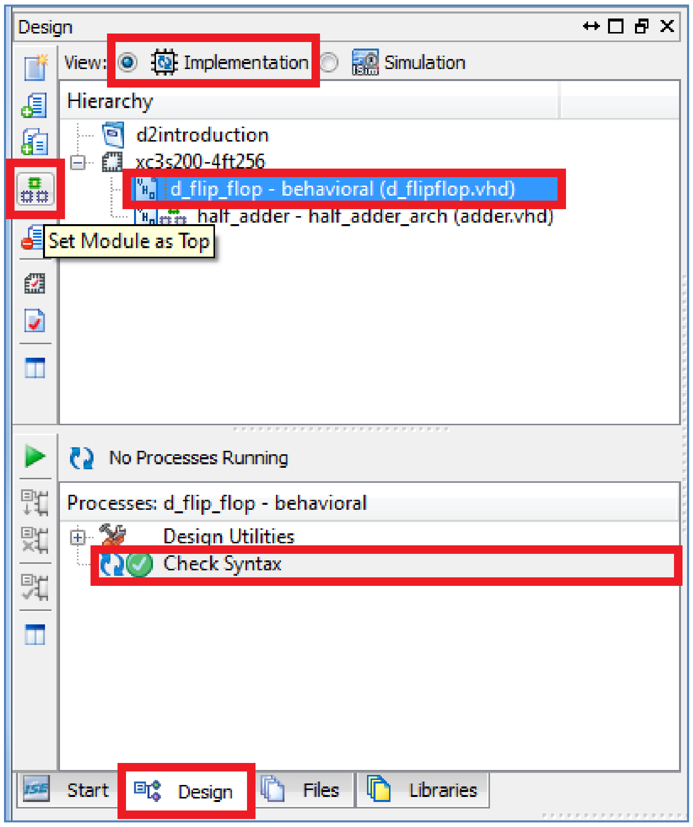
This section will show how to write a VHDL Test Bench for a clocked circuit. It involves a positive edge triggered D-flip-flop.



```
entity d_flip_flop is
    Port ( clk : in  STD_LOGIC;
           D : in  STD_LOGIC;
           Q : out  STD_LOGIC);
end d_flip_flop;

architecture behavioral of d_flip_flop is

begin
   process(clk)
   begin
      if clk'event and clk = '1' then
         Q <= D;
      end if;
   end process;
end behavioral;
```

1. Similar to step 1 to 6 of section 2, create the VHDL code for a D-flip-flop. The VHDL codes will look like:

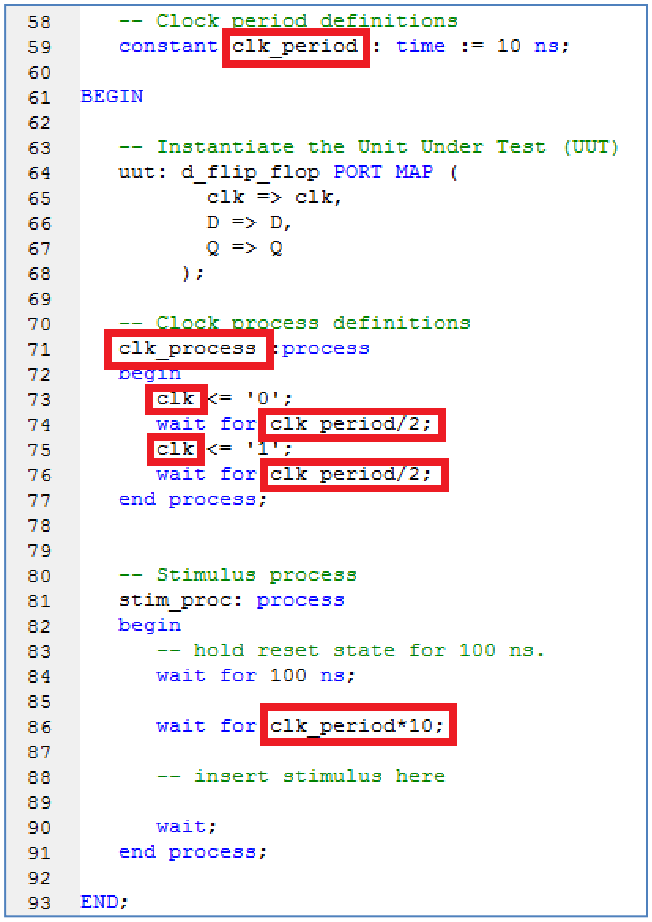
```
entity d_flip_flop is
   Port ( clk : in  STD_LOGIC;
          D : in  STD_LOGIC;
          Q : out  STD_LOGIC);
end d_flip_flop;

architecture behavioral of d_flip_flop is

begin
  process(clk)
  begin
    if clk'event and clk = '1' then
      Q <= D;
    end if;
  end process;
end behavioral;
```



2. Do not forget to save and then ***Check for Syntax*** while the ***d_flip_flop - behavioral (d_flipflop.vhd)*** is selected.

   Notice that the ***Processes for Current Source*** window is different from that of step 7 of section 2. This is still ok because syntax can still be checked.

   If there is a need to implement the d_flip_flop design on an FPGA, a ***Processes for Current Source*** window similar to that of step 7 in section 2 will be required. To have such a window, select the ***d_flip_flop - behavioral (d_flipflop.vhd)*** and click on the ***Set Module as Top*** icon. Accept the reset when prompted. This ***Set Module as Top*** is not required if FPGA implementation is not needed.



```
   -- Clock period definitions
   constant clk_period : time := 10 ns;

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: d_flip_flop PORT MAP (
          clk => clk,
          D => D,
          Q => Q
        );

   -- Clock process definitions
   clk_process :process
   begin
      clk <= '0';
      wait for clk_period/2;
      clk <= '1';
      wait for clk_period/2;
   end process;


   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
      wait for 100 ns;

      wait for clk_period*10;

      -- insert stimulus here

      wait;
   end process;

END;
```

3. After the syntax has been determined to be correct, create a VHDL Test Bench for this ***d_flip_flop***. The steps are similar to steps 1 to 4 of section 3.

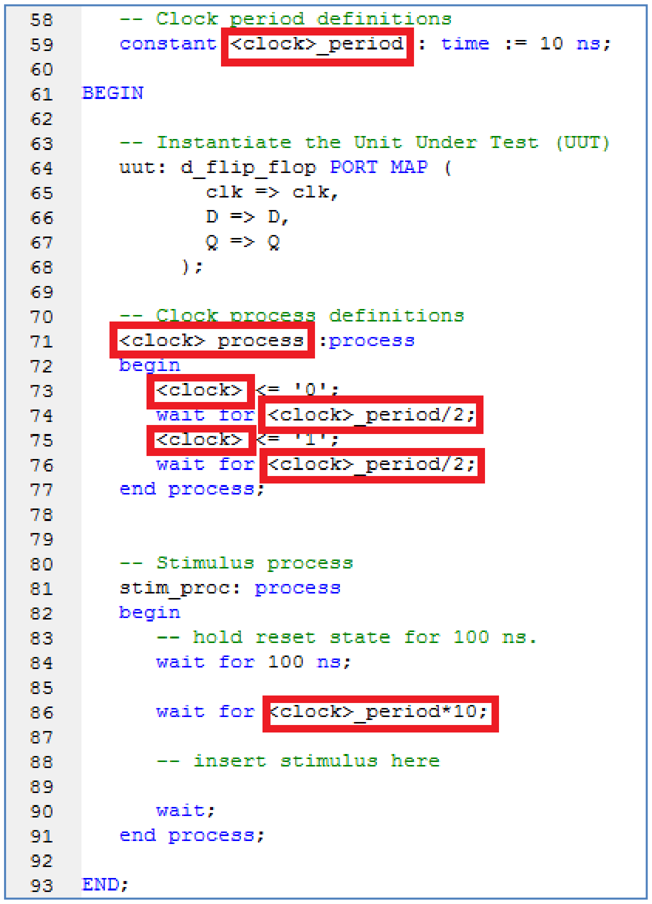
   Xilinx ISE WebPACK 13.1 usually does a good job of identifying the clock and creating a good skeleton code for the test bench accordingly. The VHDL codes used in the VHDL module help Xilinx ISE WebPACK 13.1 in determining the clock. For example:

   clk : in STD_LOGIC

   if clk'event and clk = '1' then

   The above codes indicated that ***clk*** is most likely the clock signal.

   In this figure, Xilinx ISE WebPACK 13.1 has properly identified ***clk*** as being the clock, and the correct skeleton code for the test bench has been created.



```
   -- Clock period definitions
   constant <clock>_period : time := 10 ns;

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: d_flip_flop PORT MAP (
          clk => clk,
          D => D,
          Q => Q
        );

   -- 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;
```

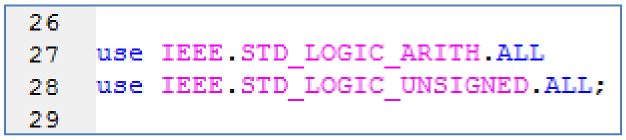
4. If clock is not identified properly, this figure might appear. Replace <clock> with the name of the port being used for clock. For example, "***clk***".

   It is recommended that the VHDL codes of the VHDL module be revised logically when Xilinx ISE WebPACK 13.1 is not able to determine the clock automatically. VHDL codes with correct clock syntax should enable Xilinx ISE WebPACK 13.1 to automatically determine the clock signal, as indicated in step 3 of this section.

5. If mathematical operations are being used for giving stimuli in an automated fashion, the following two libraries might have to be included as well:

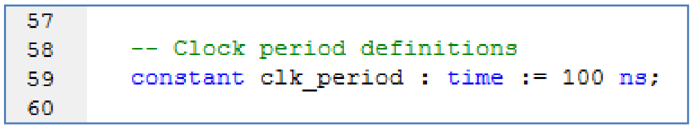
use IEEE.STD_LOGIC_ARITH.ALL
use IEEE.STD_LOGIC_UNSIGNED.ALL;

In the example described in this section, the stimuli given does not require the use of the above two libraries.



```

use IEEE.STD_LOGIC_ARITH.ALL
use IEEE.STD_LOGIC_UNSIGNED.ALL;

```

6. Continuing from step 3 of this section, change the ***clk_period*** to ***100 ns*** instead of the default ***10 ns***



```

   -- Clock period definitions
   constant clk_period : time := 100 ns;

```

7. Comment out the following line:

wait for clk_period*10;

Insert the following stimuli in the VHDL test bench:

wait for 25 ns;
D <= '1';

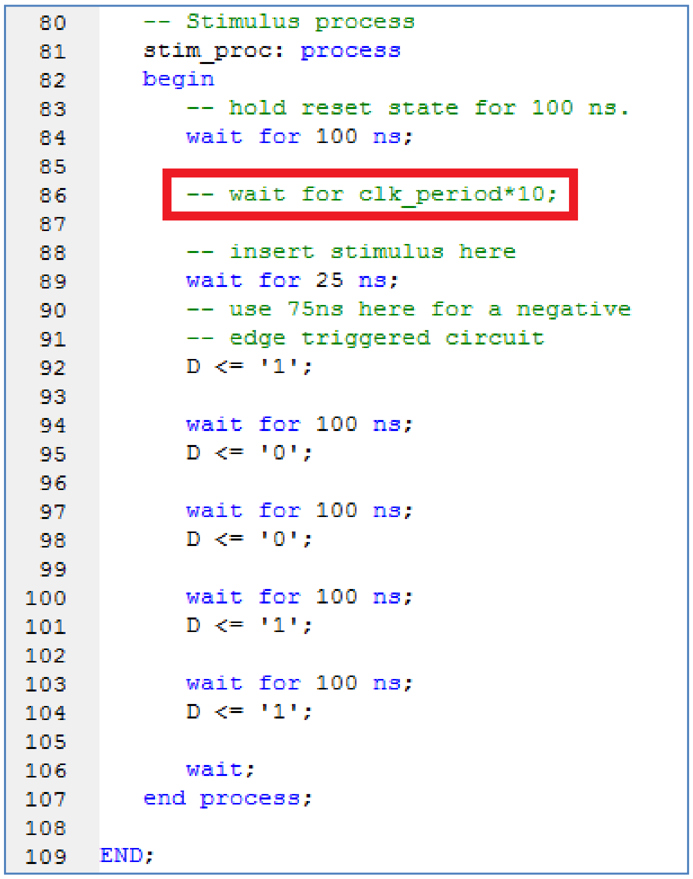
wait for 100 ns;
D <= '0';

wait for 100 ns;
D <= '0';

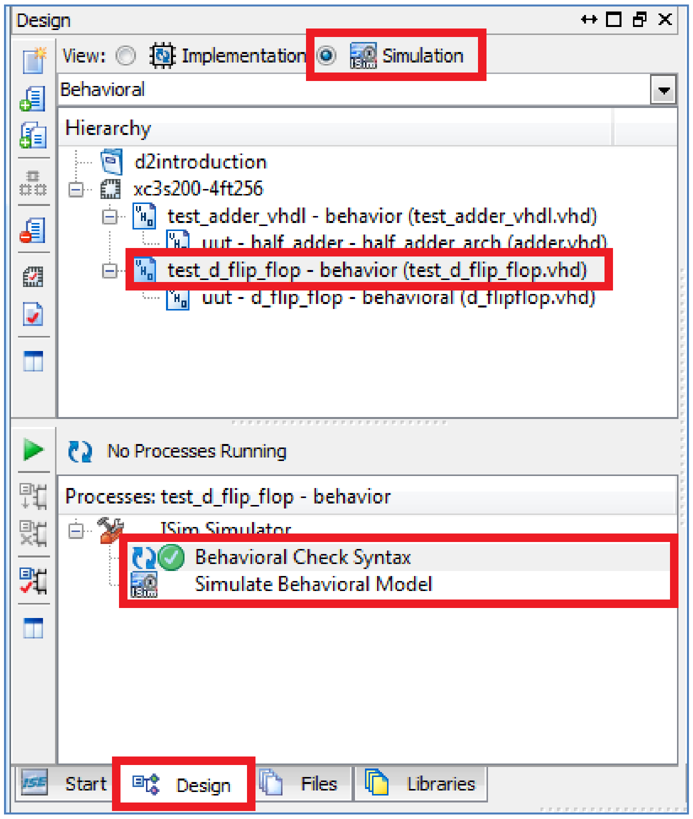
wait for 100 ns;
D <= '1';

wait for 100 ns;
D <= '1';

Note that the ***wait for 25 ns*** should preferably be ***wait for 75 ns*** if negative edge triggered circuits are used. If there is more than one input, these inputs can be changed simultaneously in between the stimuli. In this example, D is the only input that is being changed in between stimuli.



```
   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
      wait for 100 ns;

      -- wait for clk_period*10;

      -- insert stimulus here
      wait for 25 ns;
      -- use 75ns here for a negative
      -- edge triggered circuit
      D <= '1';

      wait for 100 ns;
      D <= '0';

      wait for 100 ns;
      D <= '0';

      wait for 100 ns;
      D <= '1';

      wait for 100 ns;
      D <= '1';

      wait;
   end process;

END;
```


Design
View: Implementation Simulation
Behavioral
Hierarchy
d2introduction
xc3s200-4ft256
test_adder_vhdl - behavior (test_adder_vhdl.vhd)
uut - half_adder - half_adder_arch (adder.vhd)
test_d_flip_flop - behavior (test_d_flip_flop.vhd)
uut - d_flip_flop - behavioral (d_flipflop.vhd)
No Processes Running
Processes: test_d_flip_flop - behavior
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model
Start
Design
Files
Libraries

8. Save the project by selecting ***File → Save***

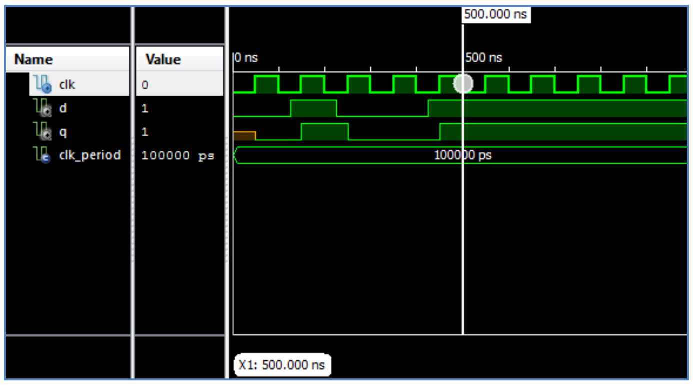
Ensure the following:

(i) The ***Design*** tab is selected.
(ii) View is set to ***Simulation***.
(iii) ***test_d_flip_flop - behaviour (test_d_flip_flop.vhd)*** is selected.
(iv) ***ISim Simulator*** is expanded.

Double click ***Behavioral Check Syntax*** to verify the VHDL code syntax.

A tick in a green circle next to ***Check Syntax*** indicates that there are no syntax errors and that the VDHL program is ready for functional simulation.

Double click S***imulate Behavioral Model*** if the Behavioral Check Syntax is correct, else verify the VHDL codes if not.

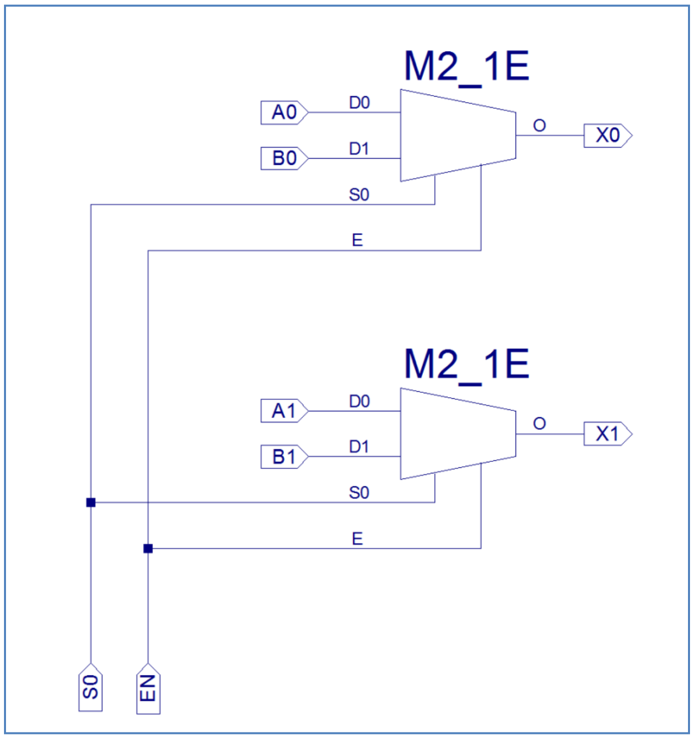


9. Similar to step 9 to 11 of section 3, the simulation results will be presented.

In this figure, the D-flip-flop is triggered at every positive clock cycle.

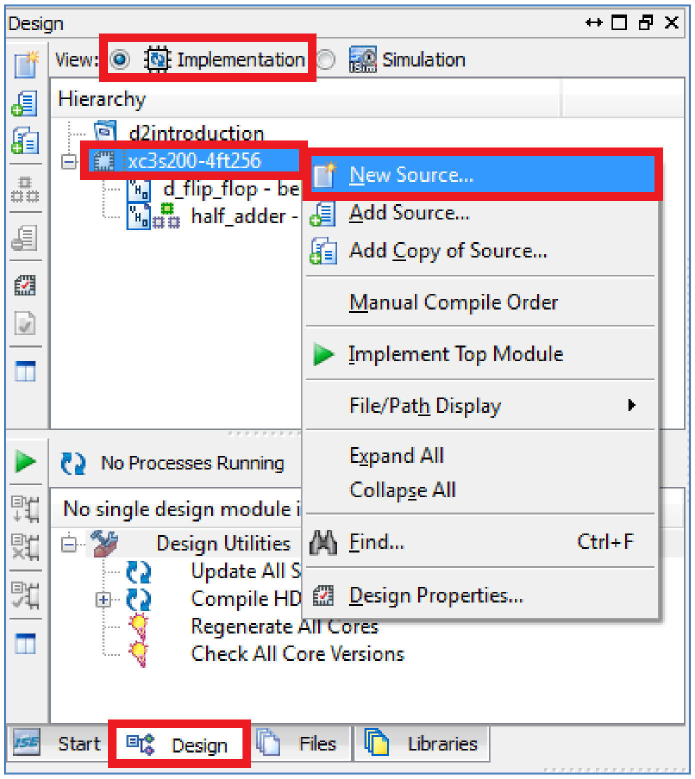
5. Schematic: Simulating a Digital Logic Circuit Using a Schematic Diagram

This section describes how to create and simulate a digital logic circuit based on a schematic diagram in Xilinx ISE WebPACK.

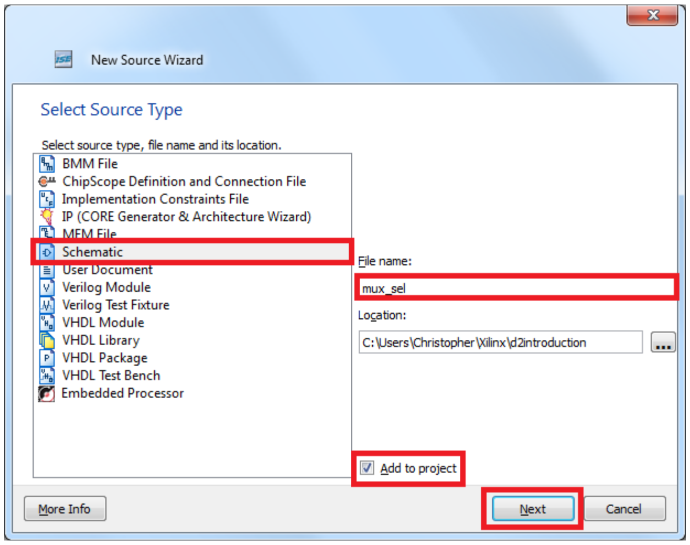


1. The circuit to be simulated represents a multiplexer-based circuit to connect one of two 2-bit words A or B to another device whose inputs are marked X.

   Important: Before the lab session, try to understand the operation of the circuit. This understanding is important for being able to verify the system functionality through simulation.



2. Add a ***New Source*** to the project.


New Source Wizard

Select Source Type

Select source type, file name and its location.

- BMM File
- ChipScope Definition and Connection File
- Implementation Constraints File
- IP (CORE Generator & Architecture Wizard)
- MEM File
- Schematic
- User Document
- Verilog Module
- Verilog Test Fixture
- VHDL Module
- VHDL Library
- VHDL Package
- VHDL Test Bench
- Embedded Processor

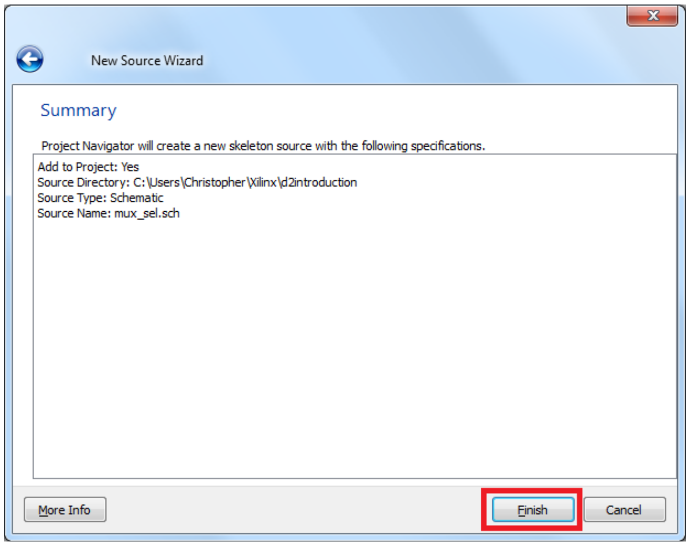
File name: mux_sel

Location: C:\Users\Christopher\Xilinx\d2introduction

☑ Add to project

3. A New Source Wizard window appears. In this Select ***Source Type*** window:
   (i) Select ***Schematic***.
   (ii) In the ***File name*** field, type the file name.
   Example: ***mux_sel***
   (iii) Ensure ***Add to project*** is checked.

   Click ***Next*** to continue.

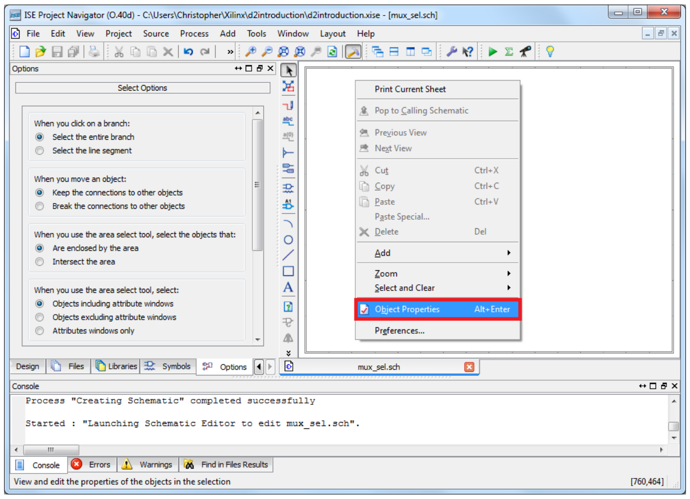

New Source Wizard

Summary

Project Navigator will create a new skeleton source with the following specifications.

Add to Project: Yes
Source Directory: C:\Users\Christopher\Xilinx\d2introduction
Source Type: Schematic
Source Name: mux_sel.sch

4. Review the Summary and close the window by clicking on ***Finish***.


ISE Project Navigator (O.40d) - C:\Users\Christopher\Xilinx\d2introduction\d2introduction.xise - [mux_sel.sch]

Options — Select Options

When you click on a branch:
- (●) Select the entire branch
- ( ) Select the line segment

When you move an object:
- (●) Keep the connections to other objects
- ( ) Break the connections to other objects

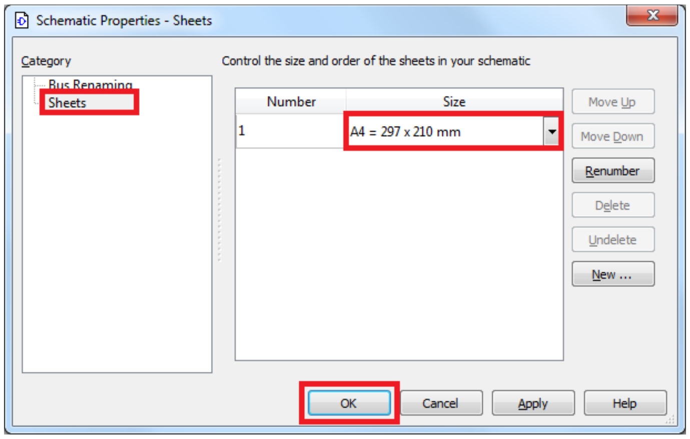
When you use the area select tool, select the objects that:
- (●) Are enclosed by the area
- ( ) Intersect the area

When you use the area select tool, select:
- (●) Objects including attribute windows
- ( ) Objects excluding attribute windows
- ( ) Attributes windows only

Console
```
Process "Creating Schematic" completed successfully

Started : "Launching Schematic Editor to edit mux_sel.sch".
```
View and edit the properties of the objects in the selection [760,464]

5. Schematic editor is very similar to any other graphic editing software. Prior to starting the schematic drawing, we may want to setup the page layout to A4 paper size.

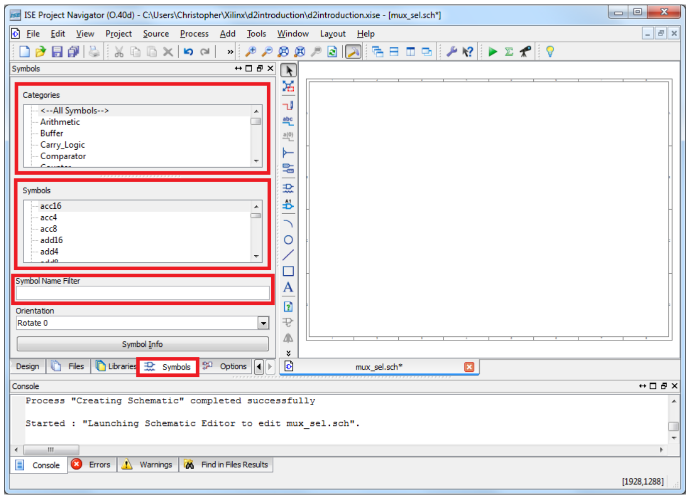
   Right click on schematic drawing space and select ***Object Properties***.



Control the size and order of the sheets in your schematic

Number	Size
1	A4 = 297 x 210 mm

6. In the window that appears, select the ***Size A4 = 297 x 210 mm*** and click on ***OK***.



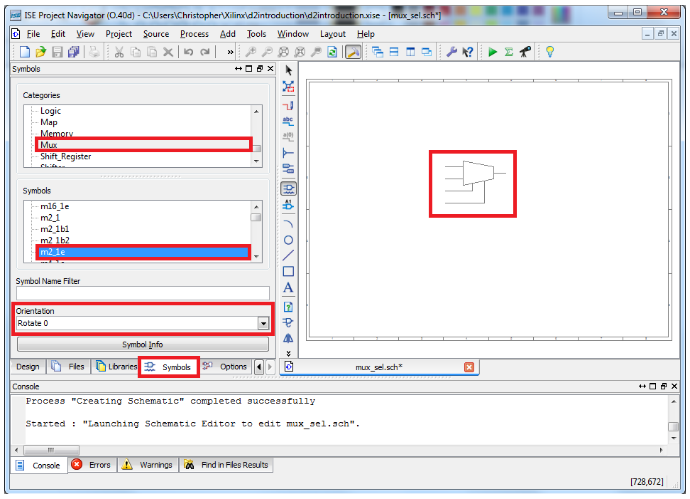
```
Process "Creating Schematic" completed successfully

Started : "Launching Schematic Editor to edit mux_sel.sch".
```

7. Back in the schematic editor, select the ***Symbols*** tab. If the ***Symbols*** tab is not available, from the menu, select ***Add → Symbol***.

The Symbols window on the left has two main sections: The ***Categories*** section and the ***Symbols*** section. The ***Categories*** section is selected first, followed by the ***Symbols*** section.

For example, we want to add a 2-bit multiplexer with enable input. First, select ***Mux*** from Categories, followed by ***m2_1e*** from symbols.

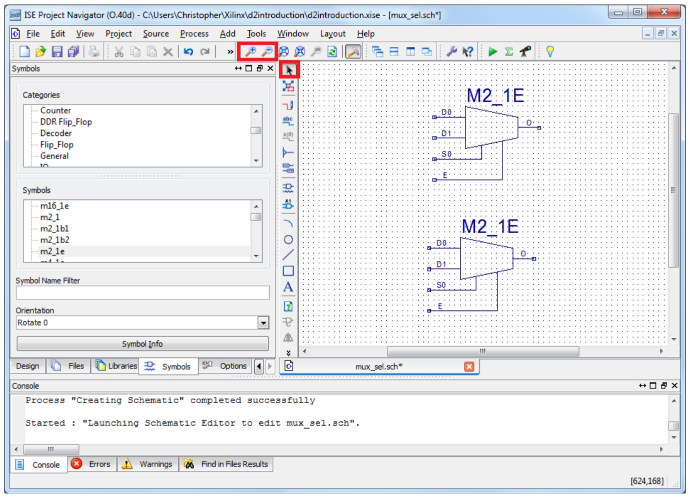
Alternately, select ***<--All Symbols-->*** from ***Categories***, and type ***m2_1e*** in the Symbol Name Filter.



```
Process "Creating Schematic" completed successfully

Started : "Launching Schematic Editor to edit mux_sel.sch".
```

8. Move the mouse back into the schematic window. The cursor should have changed to a symbol outline representing the m2_1e symbol. Move the symbol outline to a specified location and click the left mouse button to place the object.

You can rotate new components being added to a schematic by selecting CTRL+R while the component is being placed (Holding right click button) or by selecting from the ***Orientation*** drop down menu.


ISE Project Navigator (O.40d) - C:\Users\Christopher\Xilinx\d2introduction\d2introduction.xise - [mux_sel.sch*]
File Edit View Project Source Process Add Tools Window Layout Help
Symbols
Categories
Counter
DDR Flip_Flop
Decoder
Flip_Flop
General
Symbols
m16_1e
m2_1
m2_1b1
m2_1b2
m2_1e
Symbol Name Filter
Orientation
Rotate 0
Symbol Info
Design Files Libraries Symbols Options
M2_1E
D0
D1
S0
E
O
mux_sel.sch*
Console
Console Errors Warnings Find in Files Results

```
Process "Creating Schematic" completed successfully

Started : "Launching Schematic Editor to edit mux_sel.sch".
```

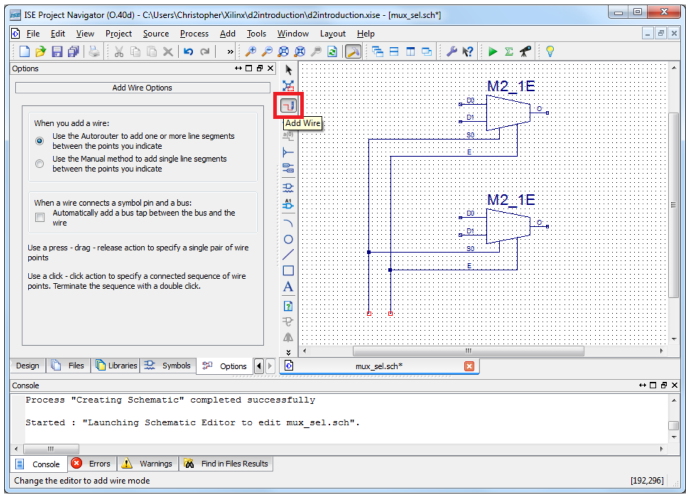
[624,168]

9. Place two ***m2_1e*** symbols in the schematic window. You can use the ***Zoom In*** and ***Zoom Out*** button, as well as the ***Select*** button to move the schematic view comfortable for you.

   To exit the Symbols Mode, press the Esc key on the keyboard.

   To move the component, click the component and drag it to the desired location.

   To delete a placed component, click the component and press the ***Delete*** key on the keyboard or right click the component and select ***Delete***.


ISE Project Navigator (O.40d) - C:\Users\Christopher\Xilinx\d2introduction\d2introduction.xise - [mux_sel.sch*]
File Edit View Project Source Process Add Tools Window Layout Help
Options
Add Wire
M2_1E
D0
D1
S0
E
O
Design Files Libraries Symbols Options
mux_sel.sch*
Console
Console Errors Warnings Find in Files Results

Add Wire Options

When you add a wire:
[✓ Use the Autorouter to add one or more line segments between the points you indicate]
Use the Manual method to add single line segments between the points you indicate

When a wire connects a symbol pin and a bus:
Automatically add a bus tap between the bus and the wire

Use a press - drag - release action to specify a single pair of wire points

Use a click - click action to specify a connected sequence of wire points. Terminate the sequence with a double click.

```
Process "Creating Schematic" completed successfully

Started : "Launching Schematic Editor to edit mux_sel.sch".
```

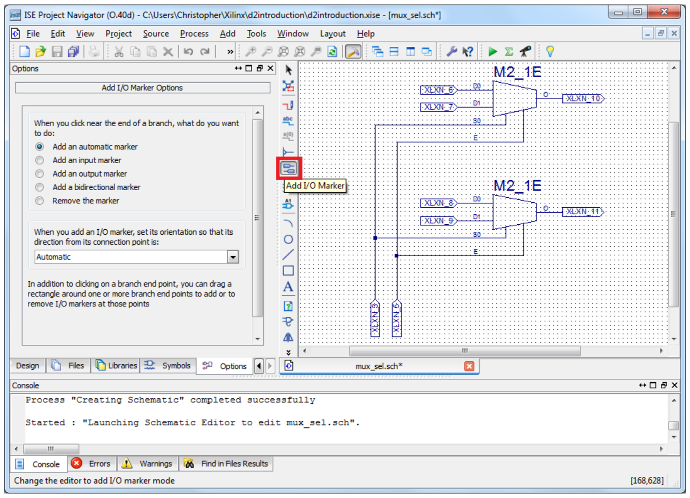
Change the editor to add wire mode [192,296]

10. Use the ***Add Wire*** icon from the tools toolbar, or select ***Add → Wire*** from the menu to draw wires (also called nets) to connect the components or extend the wires.

    To draw a wire connection, first click the cursor pointer on the terminal of the source component, usually indicated by a small square, and then move the pointer to the destination component terminal and click again.

    If the connection was successful, you will see a blue wire (net) connecting the two terminals. For long wires between distant components, and with the default options, the schematic editor will auto-route the connection so that it does not overlap with other symbols or wires. However, if you like a wire to follow a specific path you may do so by clicking on each corner of the desired path prior to clicking on the destination terminal.

    Create the wire connections as shown in this figure.


ISE Project Navigator (O.40d) - C:\Users\Christopher\Xilinx\d2introduction\d2introduction.xise - [mux_sel.sch*]
File Edit View Project Source Process Add Tools Window Layout Help
Options
Add I/O Marker
M2_1E
XLXN_6
XLXN_7
XLXN_10
XLXN_8
XLXN_9
XLXN_11
XLXN_3
XLXN_5
D0
D1
S0
E
O
Design Files Libraries Symbols Options
mux_sel.sch*
Console
Console Errors Warnings Find in Files Results

Add I/O Marker Options

When you click near the end of a branch, what do you want to do:
[✓ Add an automatic marker]
Add an input marker
Add an output marker
Add a bidirectional marker
Remove the marker

When you add an I/O marker, set its orientation so that its direction from its connection point is:
Automatic

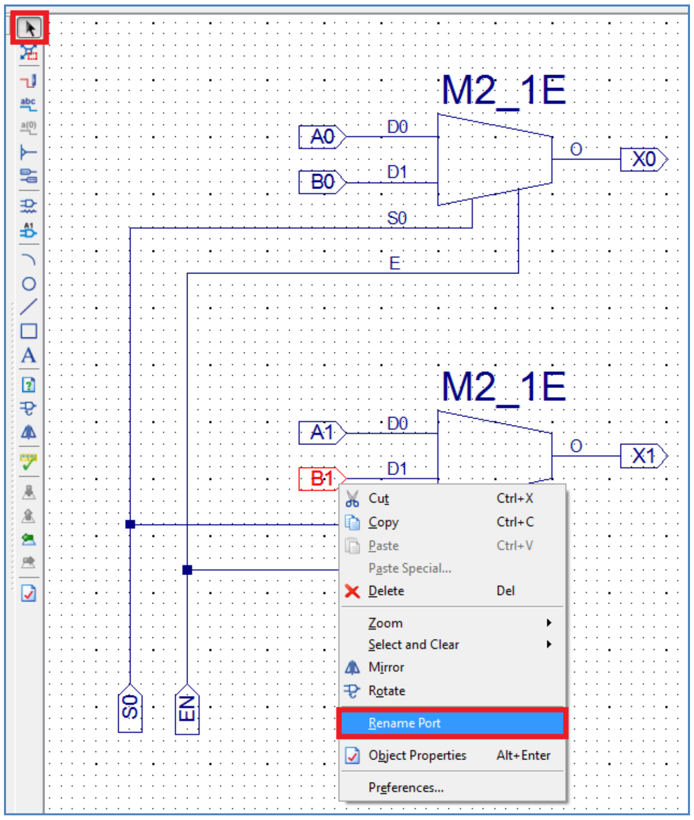
In addition to clicking on a branch end point, you can drag a rectangle around one or more branch end points to add or to remove I/O markers at those points

```
Process "Creating Schematic" completed successfully

Started : "Launching Schematic Editor to edit mux_sel.sch".
```

Change the editor to add I/O marker mode [168,628]

11. I/O markers are used to define the I/O the ports on a top level schematic. To add the I/O markers, use the ***Add I/O Marker*** icon from the tools toolbar, or select Add → ***I/O Marker*** from the menu.

    Then click and drag a box around each of the input/output nets (terminals of the devices and where the wires end, represented by small squares)

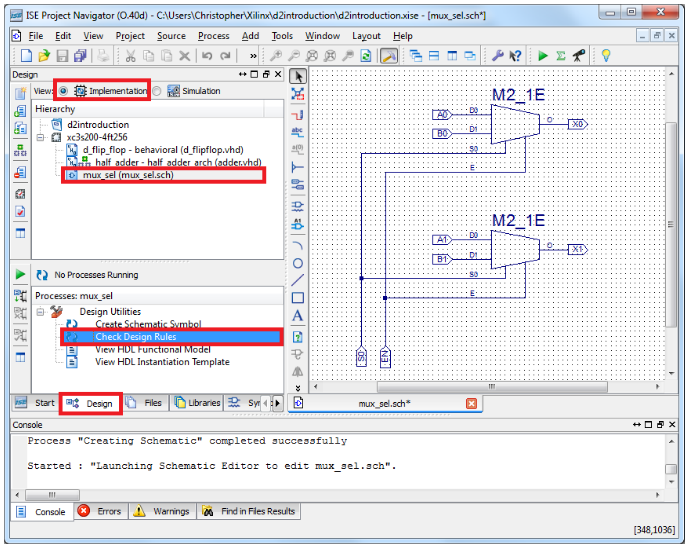


12. Ensuring that the ***Select*** icon is selected from the toolbar, right click on one of the I/O markers and select ***Rename Port***.

    Change the names to those shown in this figure.

    For the upper M2_1E device:
    A0 → D0
    B0 → D1
    S0 → S0
    EN → E
    O → ***XO***

    For the lower M2_1E device:
    A1 → D0
    B1 → D1
    S0 → S0
    EN → E
    O → ***X1***

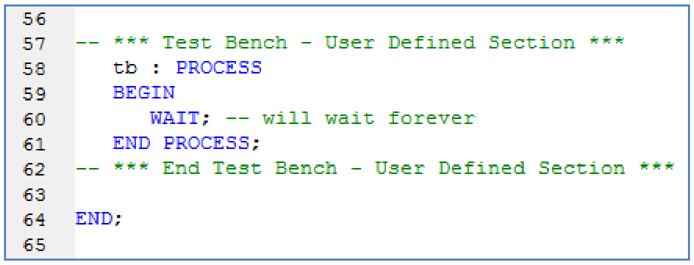

ISE Project Navigator (O.40d) - C:\Users\Christopher\Xilinx\d2introduction\d2introduction.xise - [mux_sel.sch*]
File Edit View Project Source Process Add Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
d2introduction
xc3s200-4ft256
d_flip_flop - behavioral (d_flipflop.vhd)
half_adder - half_adder_arch (adder.vhd)
mux_sel (mux_sel.sch)
No Processes Running
Processes: mux_sel
Design Utilities
Create Schematic Symbol
Check Design Rules
View HDL Functional Model
View HDL Instantiation Template
Start Design Files Libraries Syr
mux_sel.sch*
M2_1E
A0
B0
X0
M2_1E
A1
B1
X1
S0
EN
Console
Console Errors Warnings Find in Files Results
[348,1036]

```
Process "Creating Schematic" completed successfully

Started : "Launching Schematic Editor to edit mux_sel.sch".
```

13. Ensure the following:
    (i) ***Design*** tab is selected.
    (ii) ***View*** is set to ***Implementation***.
    (iii) ***mux_sel (mux_sel.sch)*** is selected.
    (iv) ***Design Utilities*** is expanded.

    Double click on ***Check Design Rules*** to perform DRC. Save the file if prompted.

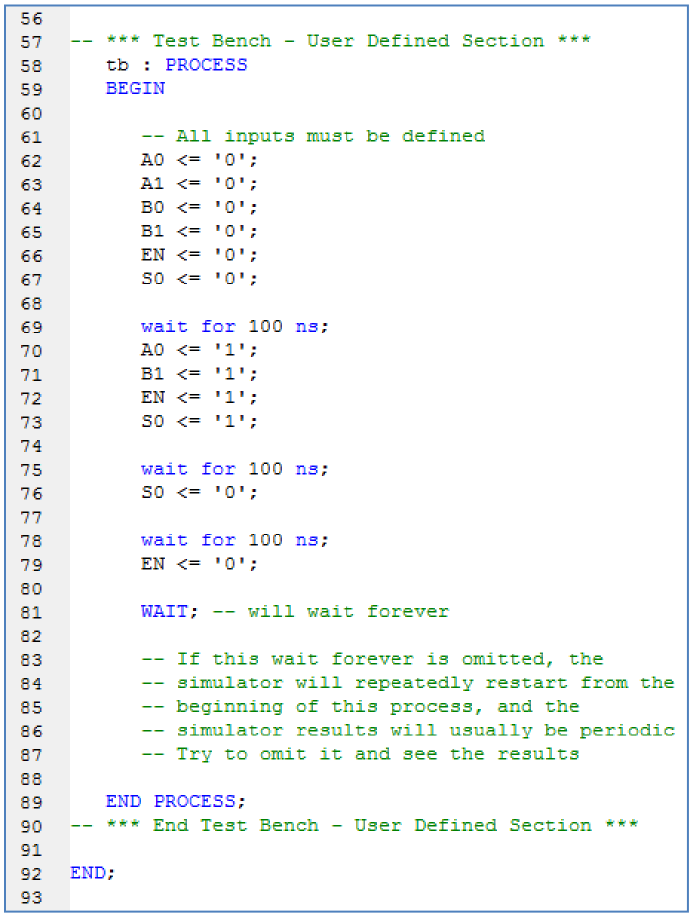
    Design Rule Checks (DRC's) are a set of rules that simply check whether you have left any input unconnected, connected more than one output to a single wire and if there are any errors like short-circuits etc. in your design. However they do not check whether your connections are logically and functionally correct.



```

-- *** Test Bench - User Defined Section ***
   tb : PROCESS
   BEGIN
      WAIT; -- will wait forever
   END PROCESS;
-- *** End Test Bench - User Defined Section ***

END;

```

14. Create a VHDL Test Bench, as explained in section 3. The file name can be called ***test_mux_sel***. This file must be associated with ***mux_sel***.

    The skeleton code created by Xilinx does have any clock in this case, as the circuit is combinational.



```
-- *** Test Bench - User Defined Section ***
   tb : PROCESS
   BEGIN

      -- All inputs must be defined
      A0 <= '0';
      A1 <= '0';
      B0 <= '0';
      B1 <= '0';
      EN <= '0';
      S0 <= '0';

      wait for 100 ns;
      A0 <= '1';
      B1 <= '1';
      EN <= '1';
      S0 <= '1';

      wait for 100 ns;
      S0 <= '0';

      wait for 100 ns;
      EN <= '0';

      WAIT; -- will wait forever

      -- If this wait forever is omitted, the
      -- simulator will repeatedly restart from the
      -- beginning of this process, and the
      -- simulator results will usually be periodic
      -- Try to omit it and see the results

   END PROCESS;
-- *** End Test Bench - User Defined Section ***

END;
```

15. Present the following stimuli to the test bench:

A0 <= '0';
A1 <= '0';
B0 <= '0';
B1 <= '0';
EN <= '0';
S0 <= '0';

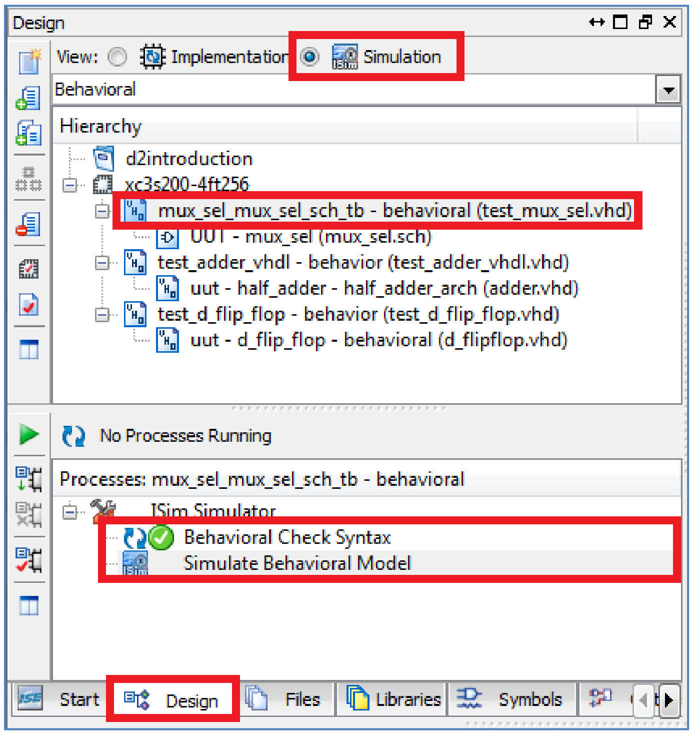
wait for 100 ns;
A0 <= '1';
B1 <= '1';
EN <= '1';
S0 <= '1';

wait for 100 ns;
S0 <= '0';

wait for 100 ns;
EN <= '0';

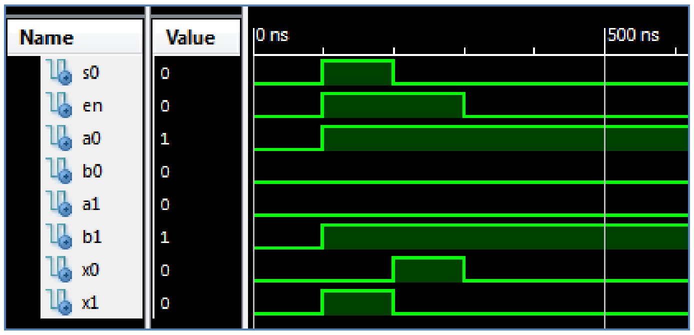
Do not forget to define all inputs at the beginning of the process, even if it has a value of '0'.

Also, take note that if the ***WAIT; --will wait forever*** is omitted, the simulator will repeatedly restart from the beginning of this process, and the simulator results will usually be periodic. Try to omit it later on and see the results.

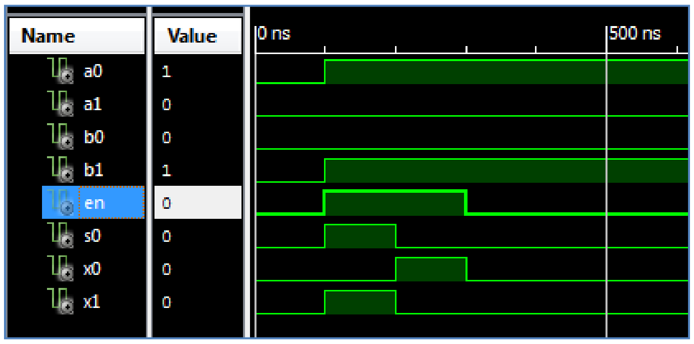

Design
View: Implementation Simulation
Behavioral
Hierarchy
d2introduction
xc3s200-4ft256
mux_sel_mux_sel_sch_tb - behavioral (test_mux_sel.vhd)
UUT - mux_sel (mux_sel.sch)
test_adder_vhdl - behavior (test_adder_vhdl.vhd)
uut - half_adder - half_adder_arch (adder.vhd)
test_d_flip_flop - behavior (test_d_flip_flop.vhd)
uut - d_flip_flop - behavioral (d_flipflop.vhd)
No Processes Running
Processes: mux_sel_mux_sel_sch_tb - behavioral
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model
Start
Design
Files
Libraries
Symbols

16. While the ***Design*** tab is selected, with ***View*** set to ***Simulation*** and ***mux_sel_mux_sel_sch_tb - behavioral(test_mux_sel.vhd)*** selected, expand ***ISim Simulator***.

Perform a ***Behavioral Check Syntax***, and if successful, double click on ***Simulate Behavioral Model***.

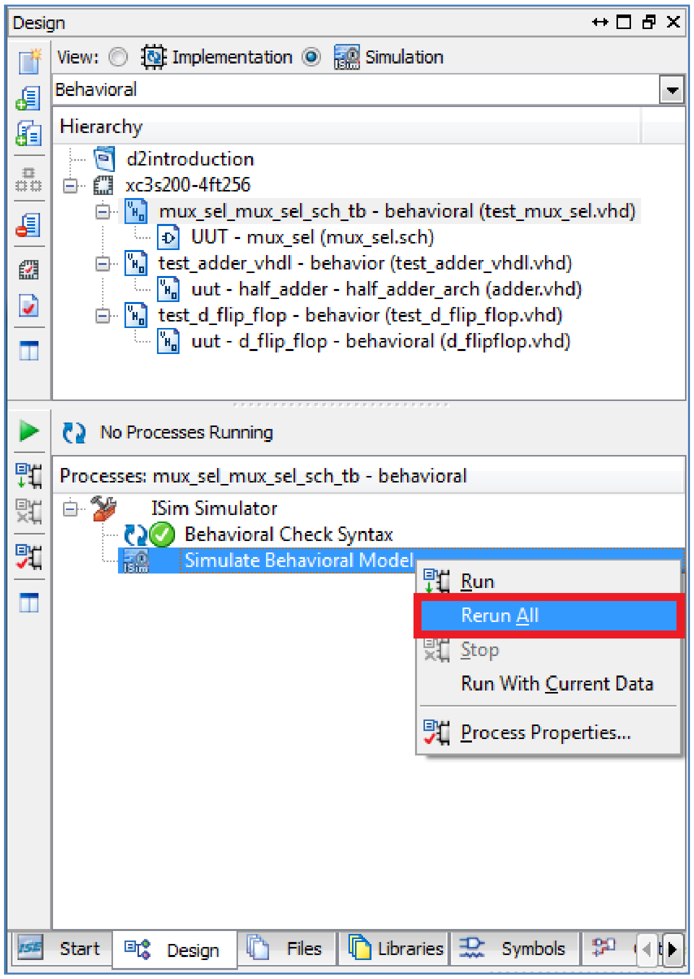


17. The simulation results will be presented. Do not forget to ***Zoom to Full View*** to have a clearer view of the results.

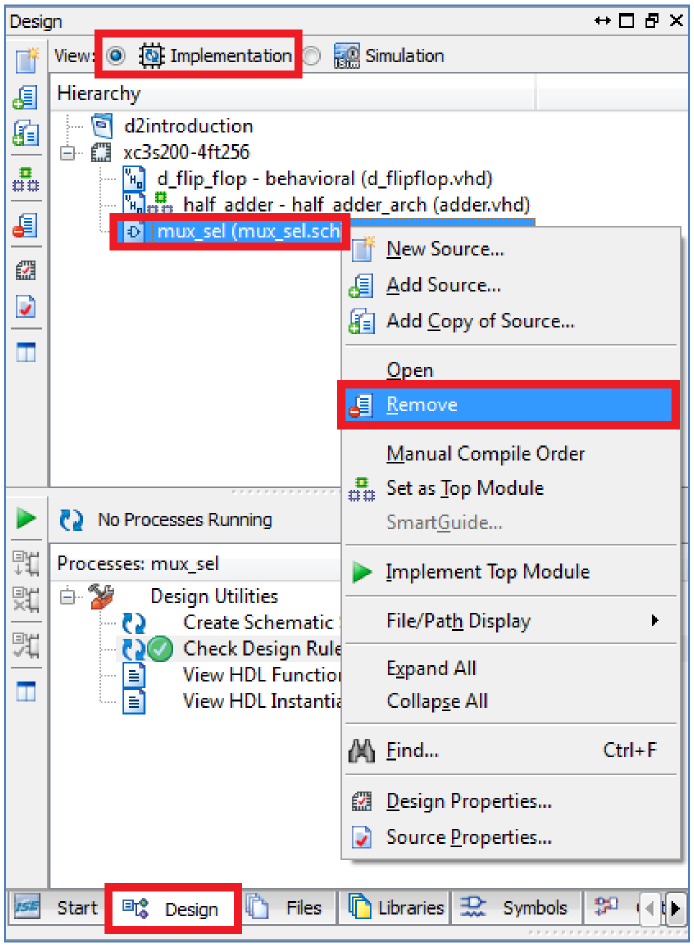


18. You can also click on one of the ***Name*** and drag them up or down. This depends on your preference for the order of the results. the order can make it easier for us to understand the simulation results.

    Exit the simulation window, and the results can be saved if desired.


Design
View: Implementation Simulation
Behavioral
Hierarchy
d2introduction
xc3s200-4ft256
mux_sel_mux_sel_sch_tb - behavioral (test_mux_sel.vhd)
UUT - mux_sel (mux_sel.sch)
test_adder_vhdl - behavior (test_adder_vhdl.vhd)
uut - half_adder - half_adder_arch (adder.vhd)
test_d_flip_flop - behavior (test_d_flip_flop.vhd)
uut - d_flip_flop - behavioral (d_flipflop.vhd)
No Processes Running
Processes: mux_sel_mux_sel_sch_tb - behavioral
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model
Run
Rerun All
Stop
Run With Current Data
Process Properties...
Start
Design
Files
Libraries
Symbols

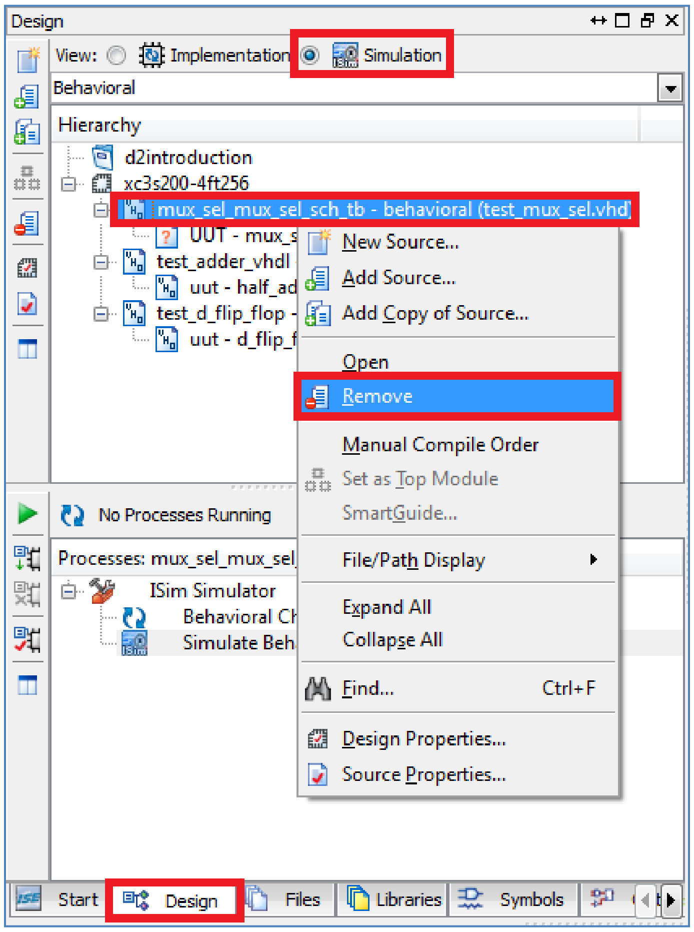
19. Note that only one simulation window can run at a time. Sometimes, you will not be allowed to ***Simulate Behavioral Model*** because of some existing processes. To solve this issue, ensure that the existing simulation result window is closed, right click on ***Simulate Behavioral Model*** and select ***Rerun All***.



20. This step and the next step is not essential. However, if you want your screenshots for section 6 to be similar to the screenshots in this manual, perform the following:

(i) Ensure the ***Design*** tab is selected.
(ii) ***View*** should be set to ***Implementation***.
(iii) Right click on ***mux_sel(mux_sel.sch)*** and choose ***Remove***.
(iv) Confirm the removal in the window that appears by clicking on ***Yes***.

If you can cope with slightly dissimilar screenshots, there is no need to perform this step and the next one.



21. Similarly:

(i) Ensure the ***Design*** tab is selected.
(ii) ***View*** should be set to ***Simulation***.
(iii) Right click on ***mux_sel_mux_sel_sch_tb - behavioral (test_mux_sel.vhd)*** and choose ***Remove***.
(iv) Confirm the removal in the window that appears by clicking on ***Yes***.

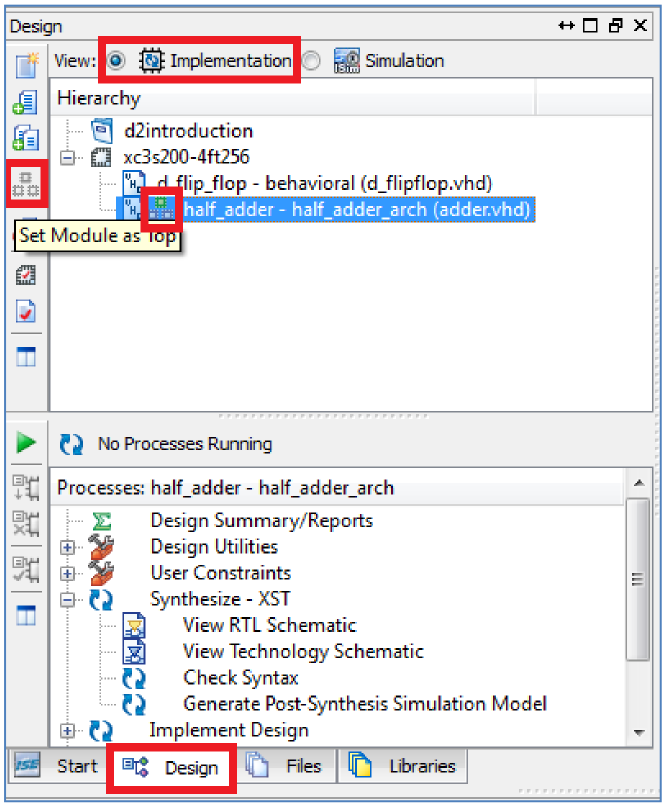
Note that the files that have been removed from the project are not deleted. They are still present in your Xilinx project folder, but are not being used in the current project window.

If they needs to be added later on, ***Add Source*** instead of the ***New Source*** option can be used.

Hence, your screen should now be similar to the screenshots of section 6. The ***Symbols*** tab and ***Options*** can also be closed to further improve similarity.

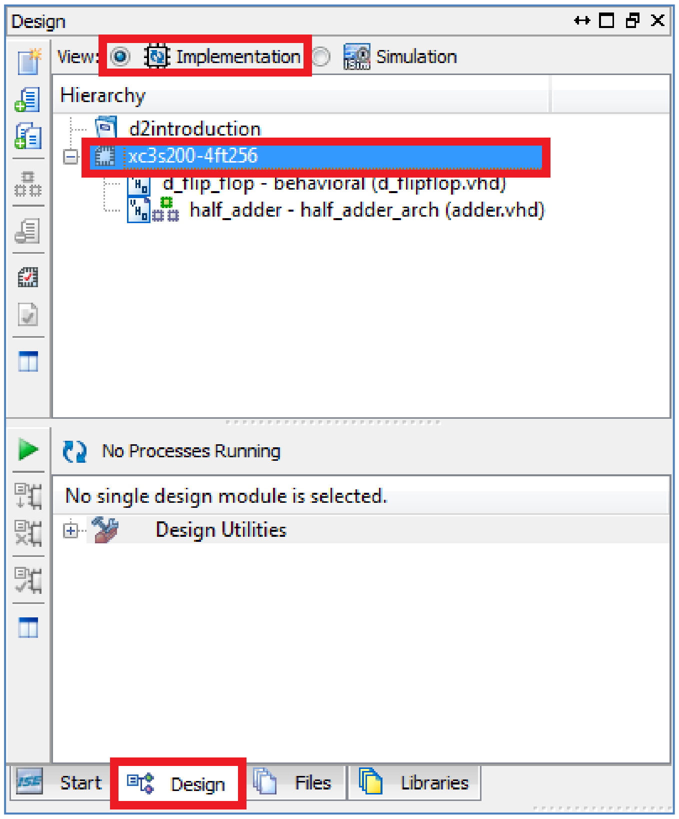
6. FPGA Implementation: 1-Bit Half-Adder

This section describes how to implement the design made in section 1 to 3 on an actual FPGA hardware.

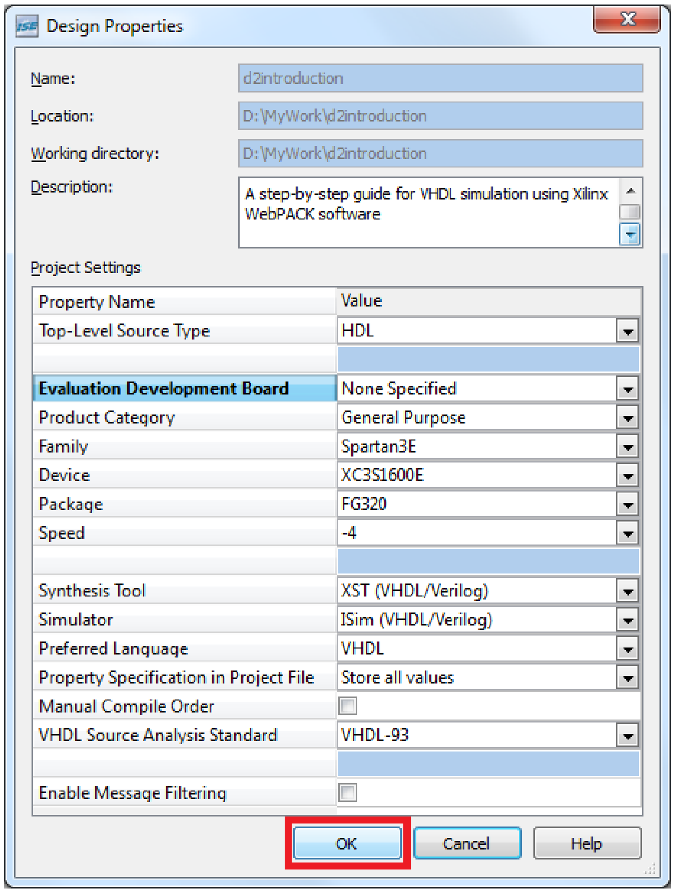


1. Set the ***half_adder - half_adder_arch (adder.vhd)*** as ***Top Module*** if not already a ***Top Module***. To this by selecting ***half_adder - half_adder_arch (adder.vhd)*** and clicking on the ***Set Module as Top*** icon

   If the file is already a ***Top Module***, an icon will appear to the left of the file name as shown in the figure, and the ***Set Module as Top*** icon will be disabled


Design
View: Implementation Simulation
Hierarchy
d2introduction
xc3s200-4ft256
d_flip_flop - behavioral (d_flipflop.vhd)
half_adder - half_adder_arch (adder.vhd)
No Processes Running
No single design module is selected.
Design Utilities
Start
Design
Files
Libraries

2. Double click on ***xcs200-4ft256***


Design Properties

Name: d2introduction

Location: D:\MyWork\d2introduction

Working directory: D:\MyWork\d2introduction

Description: A step-by-step guide for VHDL simulation using Xilinx WebPACK software

Project Settings

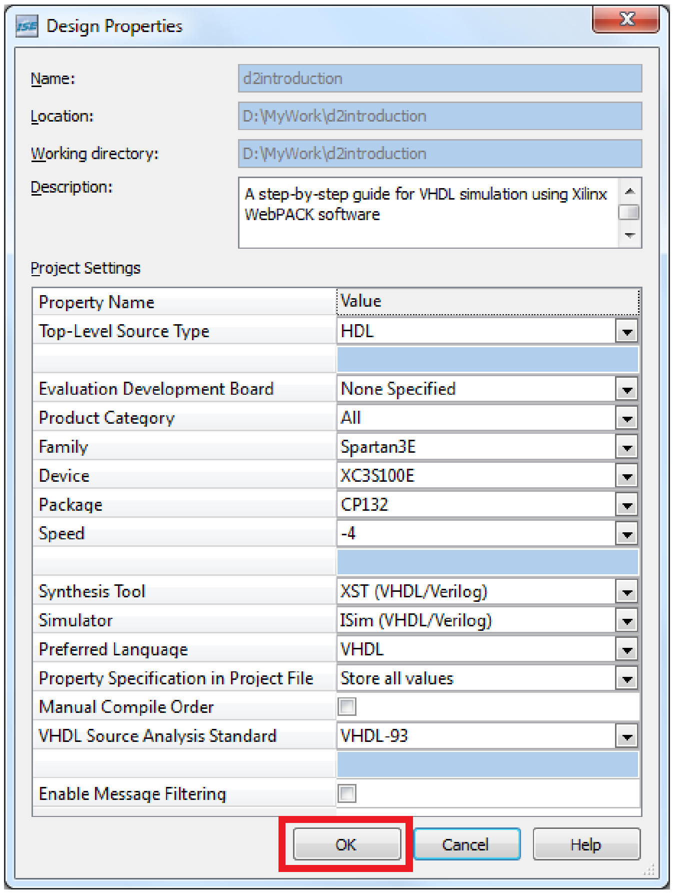
Property Name	Value
Top-Level Source Type	HDL
Evaluation Development Board	None Specified
Product Category	General Purpose
Family	Spartan3E
Device	XC3S1600E
Package	FG320
Speed	-4
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	VHDL
Property Specification in Project File	Store all values
Manual Compile Order	☐
VHDL Source Analysis Standard	VHDL-93
Enable Message Filtering	☐

3. Set the following:

For ***Spartan 3E 1600E Board*** (Bigger, green board)

(i) Top-Level Source Type: ***HDL***
(ii) Evaluation Development Board: ***None Specified***
(iii) Product Category: ***General Purpose***
(iv) Family: ***Spartan3E***
(v) Device: ***XC3S1600E***
(vi) Package: ***FG320***
(vii) Speed: ***-4***
(viii) Synthesis Tool: ***XST (VHDL/Verilog)***
(ix) Simulator: ***ISim (VHDL/Verilog)***
(x) Preferred Language: ***VHDL***
(xi) Property Specification in Project File: ***Store all values***
(xii) Manual Compile Order: ***Unchecked***
(xiii) VHDL Source Analysis Standard: ***VHDL-93***
(xiv) Enable Message Filtering: ***Unchecked***

Click on ***OK***.


Design Properties

Name: d2introduction

Location: D:\MyWork\d2introduction

Working directory: D:\MyWork\d2introduction

Description: A step-by-step guide for VHDL simulation using Xilinx WebPACK software

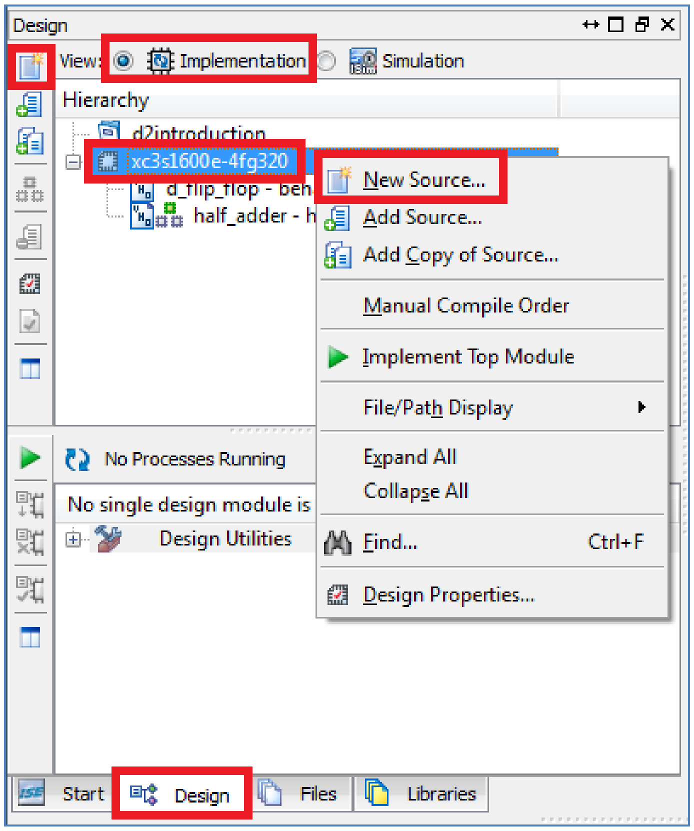
Project Settings

Property Name	Value
Top-Level Source Type	HDL
Evaluation Development Board	None Specified
Product Category	All
Family	Spartan3E
Device	XC3S100E
Package	CP132
Speed	-4
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	VHDL
Property Specification in Project File	Store all values
Manual Compile Order	☐
VHDL Source Analysis Standard	VHDL-93
Enable Message Filtering	☐

For ***Spartan 3E 100E Board*** (BASYS2)

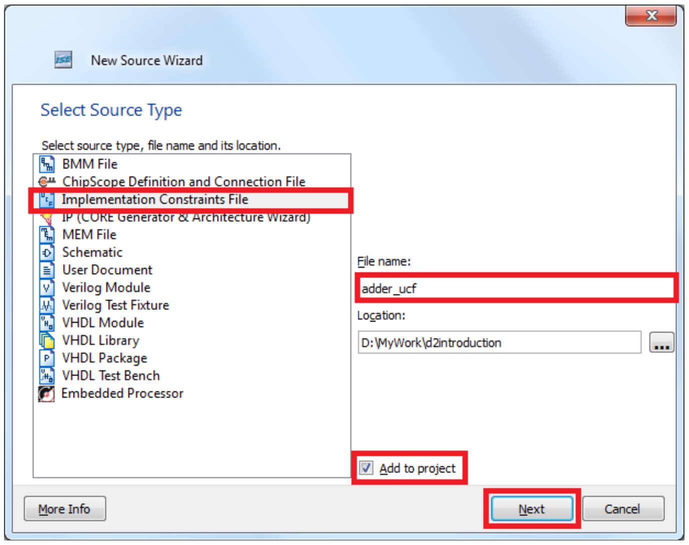
(i) Top-Level Source Type: ***HDL***
(ii) Evaluation Development Board: ***None Specified***
(iii) Product Category: ***General Purpose***
(iv) Family: ***Spartan3E***
(v) Device: ***XC3S100E***
(vi) Package: ***CP132***
(vii) Speed: ***-4***
(viii) Synthesis Tool: ***XST (VHDL/Verilog)***
(ix) Simulator: ***ISim (VHDL/Verilog)***
(x) Preferred Language: ***VHDL***
(xi) Property Specification in Project File: ***Store all values***
(xii) Manual Compile Order: ***Unchecked***
(xiii) VHDL Source Analysis Standard: ***VHDL-93***
(xiv) Enable Message Filtering: ***Unchecked***

Click on ***OK***.



4. Add a ***New Source*** to the project.

   Note: The device shown in the ***Design*** window will depend on the board that is being used, but the Xilinx ISE WebPACK 13.1 procedure for creating the codes for the FPGA are the same. The figures in this manual will be based on the ***xc3s1600e-4fg320***.

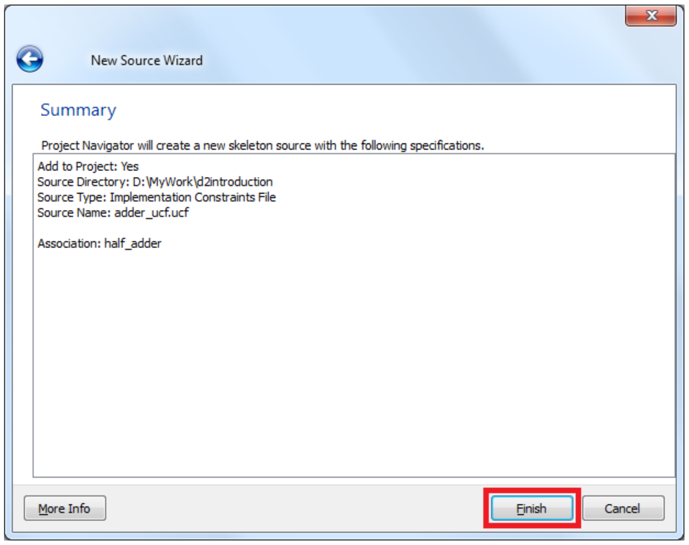


File name: adder_ucf
Location: D:\MyWork\d2introduction

5. Select ***Implementation Constraints File*** in the ***Select Source Type*** window.

   Enter an appropriate ***File Name***.
   Example: ***adder_ucf***

   Check ***Add to Project*** and click on ***Next*** to continue.

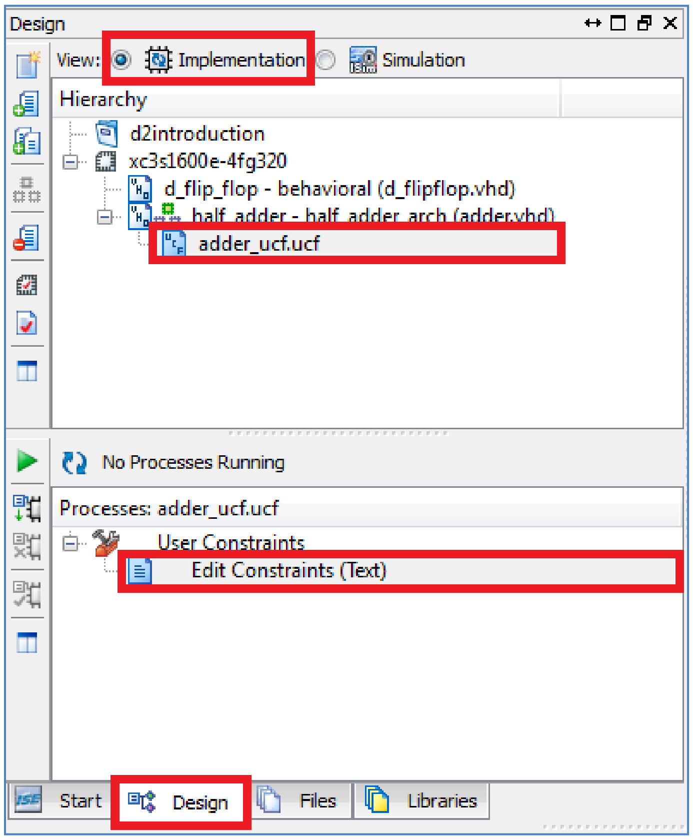


Project Navigator will create a new skeleton source with the following specifications.

Add to Project: Yes
Source Directory: D:\MyWork\d2introduction
Source Type: Implementation Constraints File
Source Name: adder_ucf.ucf

Association: half_adder

6. Click on ***Finish*** to close the New Source Wizard window.



7. The ***.ucf*** file needs to be edited. To do this,

(i) Ensure the ***Design*** tab is selected.
(ii) ***View*** is set to ***Implementation***.
(iii) In the Design window, expand ***half_adder - half-adder_arch (adder.vhd)***.
(iv) Click once on ***adder_ucf.ucf***.
(v) In the Processes for Current Source window, expand ***User Constraints***.
(vi) Double click on ***Edit Constraints (Text)***.

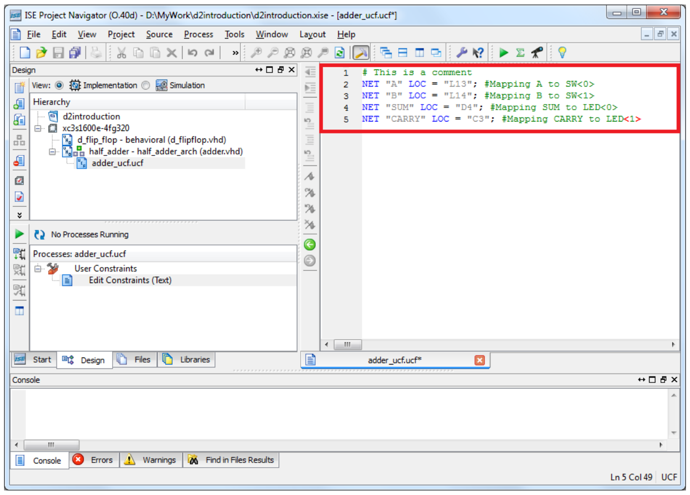
8. In the text file opening up in the ***Editor*** window, specify the mapping of inputs and outputs to FPGA pins, depending on the board being used.

   Create the UCF file mapping input/output signals (in your code) with pins. Text following '#' is a comment.

   For ***Spartan 3E 1600E Board*** (Bigger, green board), type:

   NET "A" LOC = "L13"; #Mapping A to SW<0>
   NET "B" LOC = "L14"; #Mapping B to SW<1>
   NET "SUM" LOC = "D4"; #Mapping SUM to LED<0>
   NET "CARRY" LOC = "C3"; #Mapping CARRY to LED<1>

   Save the .ucf file by clicking on ***File → Save***

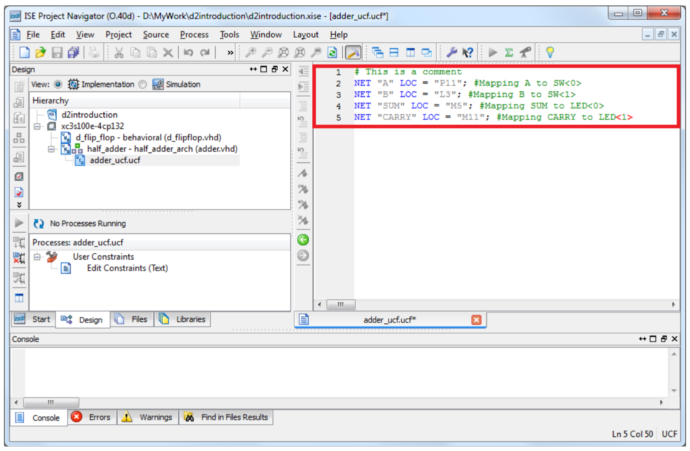

ISE Project Navigator (O.40d) - D:\MyWork\d2introduction\d2introduction.xise - [adder_ucf.ucf*]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
d2introduction
xc3s1600e-4fg320
d_flip_flop - behavioral (d_flipflop.vhd)
half_adder - half_adder_arch (adder.vhd)
adder_ucf.ucf
No Processes Running
Processes: adder_ucf.ucf
User Constraints
Edit Constraints (Text)
Start Design Files Libraries
adder_ucf.ucf*
Console
Console Errors Warnings Find in Files Results
Ln 5 Col 49 UCF

```
# This is a comment
NET "A" LOC = "L13"; #Mapping A to SW<0>
NET "B" LOC = "L14"; #Mapping B to SW<1>
NET "SUM" LOC = "D4"; #Mapping SUM to LED<0>
NET "CARRY" LOC = "C3"; #Mapping CARRY to LED<1>
```

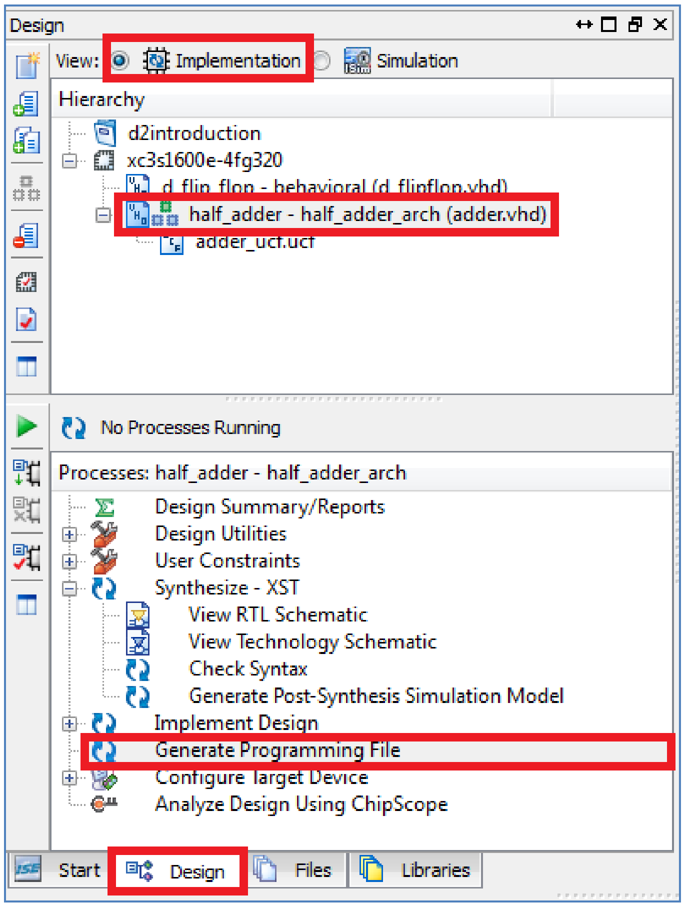
For ***Spartan 3E 100E Board*** (BASYS2), type:

NET "A" LOC = "P11"; #Mapping A to SW<0>
NET "B" LOC = "L3"; #Mapping B to SW<1>
NET "SUM" LOC = "M5"; #Mapping SUM to LED<0>
NET "CARRY" LOC = "M11"; #Mapping CARRY to LED<1>

Save the .ucf file by clicking on ***File → Save***

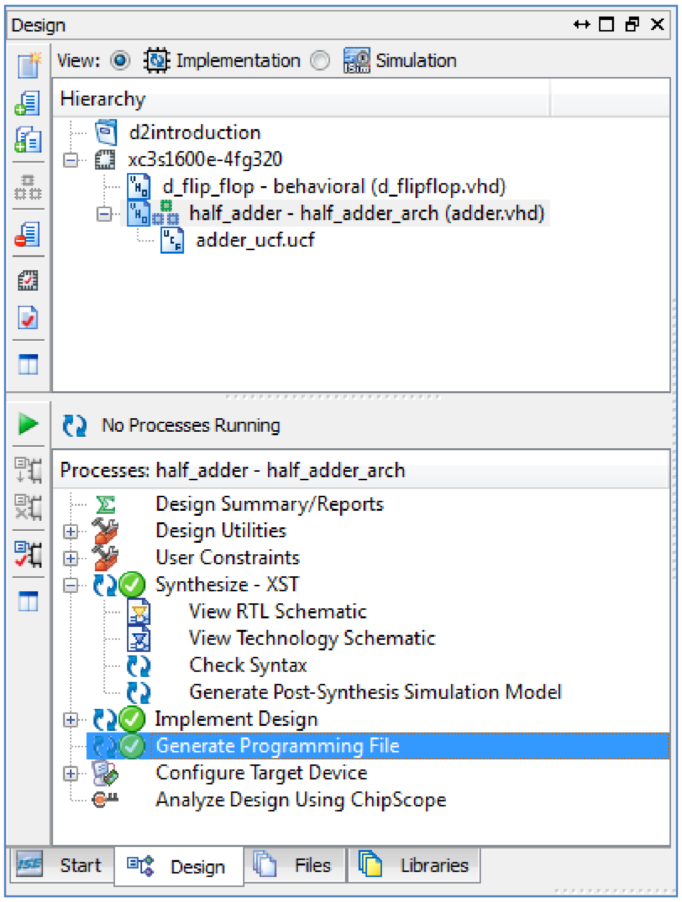

ISE Project Navigator (O.40d) - D:\MyWork\d2introduction\d2introduction.xise - [adder_ucf.ucf*]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
d2introduction
xc3s100e-4cp132
d_flip_flop - behavioral (d_flipflop.vhd)
half_adder - half_adder_arch (adder.vhd)
adder_ucf.ucf
No Processes Running
Processes: adder_ucf.ucf
User Constraints
Edit Constraints (Text)
Start Design Files Libraries
adder_ucf.ucf*
Console
Console Errors Warnings Find in Files Results
Ln 5 Col 50 UCF

```
# This is a comment
NET "A" LOC = "P11"; #Mapping A to SW<0>
NET "B" LOC = "L3"; #Mapping B to SW<1>
NET "SUM" LOC = "M5"; #Mapping SUM to LED<0>
NET "CARRY" LOC = "M11"; #Mapping CARRY to LED<1>
```

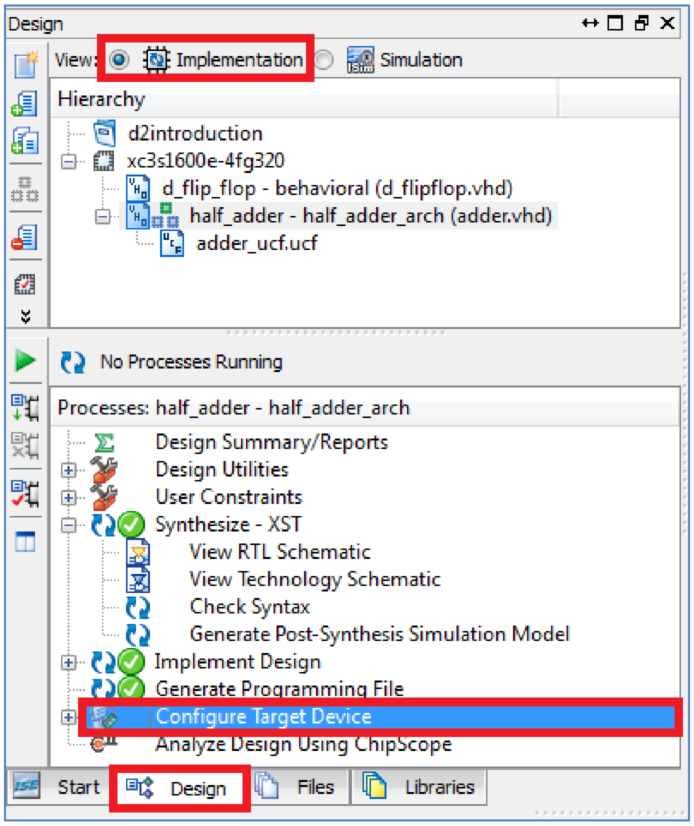

Design
View: Implementation Simulation
Hierarchy
d2introduction
xc3s1600e-4fg320
d_flip_flop - behavioral (d_flipflop.vhd)
half_adder - half_adder_arch (adder.vhd)
adder_ucf.ucf
No Processes Running
Processes: half_adder - half_adder_arch
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation Model
Implement Design
Generate Programming File
Configure Target Device
Analyze Design Using ChipScope
Start
Design
Files
Libraries

9. In the ***Design*** window, highlight the VHDL source ***half_adder - half_adder_arch (adder.vhd)***, and double click on ***Generate Programming File*** in the ***Process for Current Source*** window.

   This process will take some time.


Design
View: Implementation Simulation
Hierarchy
d2introduction
xc3s1600e-4fg320
d_flip_flop - behavioral (d_flipflop.vhd)
half_adder - half_adder_arch (adder.vhd)
adder_ucf.ucf
No Processes Running
Processes: half_adder - half_adder_arch
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation Model
Implement Design
Generate Programming File
Configure Target Device
Analyze Design Using ChipScope
Start
Design
Files
Libraries

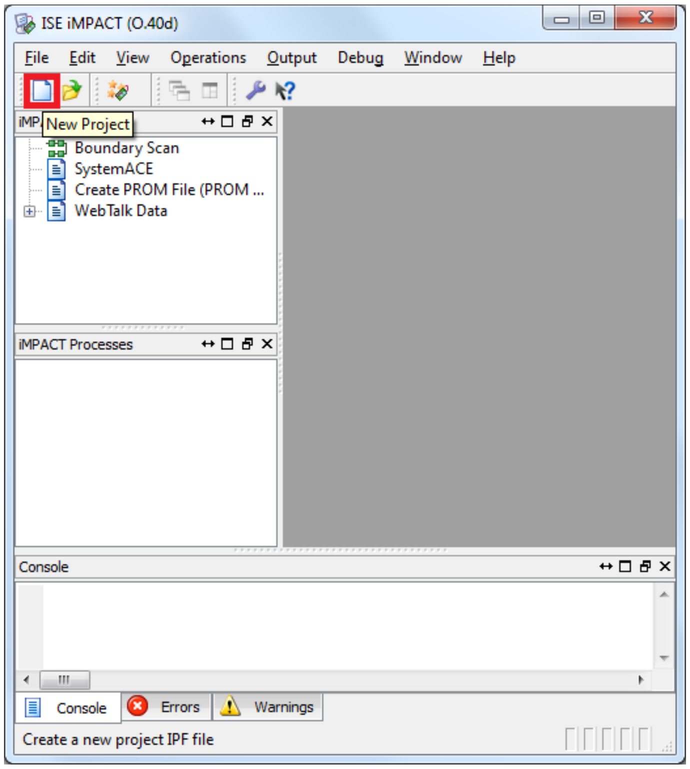
10. Once the synthesis is complete and syntax is correct, tick marks in green circles will be shown to the left of ***Synthesize - XST***, ***Implement Design*** and ***Generate Programming File***.


Design
View: Implementation Simulation
Hierarchy
d2introduction
xc3s1600e-4fg320
d_flip_flop - behavioral (d_flipflop.vhd)
half_adder - half_adder_arch (adder.vhd)
adder_ucf.ucf
No Processes Running
Processes: half_adder - half_adder_arch
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation Model
Implement Design
Generate Programming File
Configure Target Device
Analyze Design Using ChipScope
Start
Design
Files
Libraries

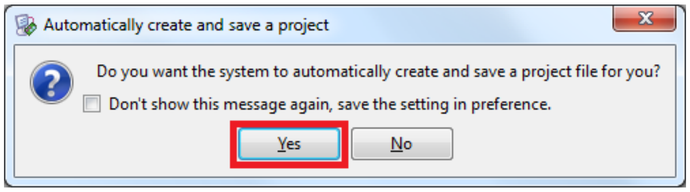
Steps 11A to 21A are applicable only if Spartan 3E 1600E board is being used.

11A. Make sure that the FPGA board is powered on and connected to the PC via USB cable.

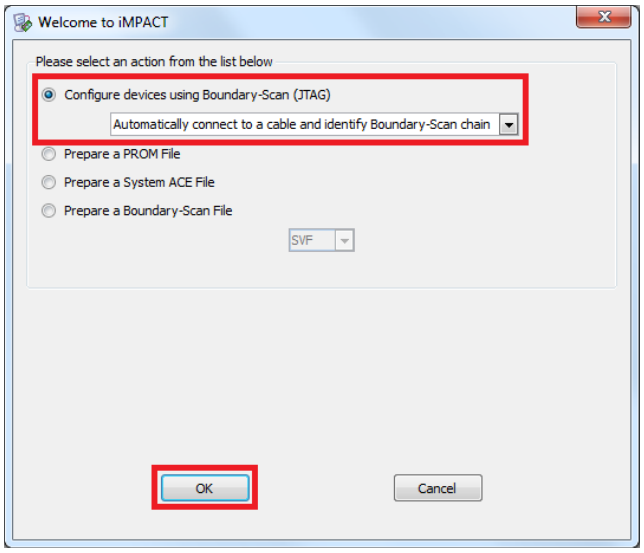
Then double click on ***Configure Target Device*** in the Xilinx ***Process for Current Source*** window. Dismiss the iMPACT project message box that pops up by clicking ***OK***.



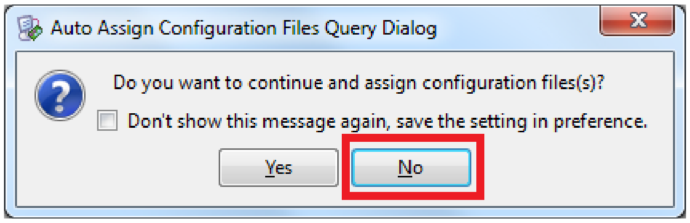
12A. A window as shown will appear. Create a New Project by clicking on the New Project icon or by selecting ***File* → *New Project***.


Automatically create and save a project
Do you want the system to automatically create and save a project file for you?
Don't show this message again, save the setting in preference.
Yes
No

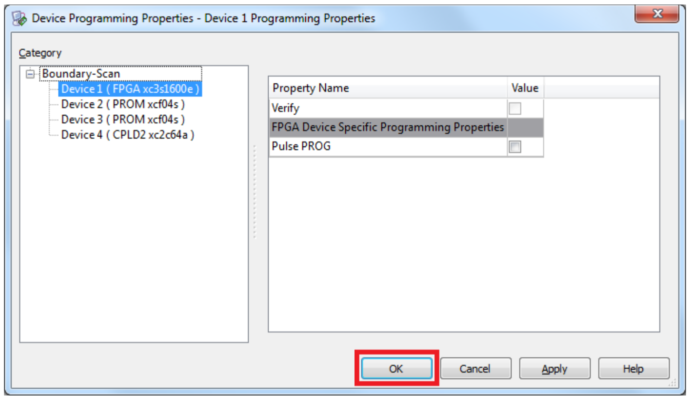
13A. Click ***Yes*** if shown with this window.


Welcome to iMPACT
Please select an action from the list below
Configure devices using Boundary-Scan (JTAG)
Automatically connect to a cable and identify Boundary-Scan chain
Prepare a PROM File
Prepare a System ACE File
Prepare a Boundary-Scan File
SVF
OK
Cancel

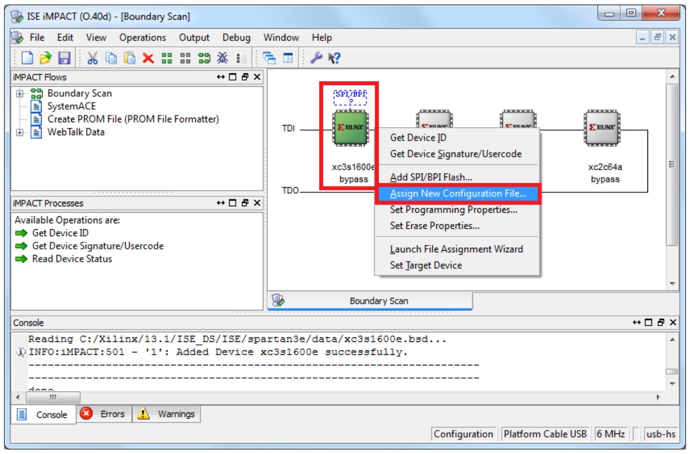
14A. Select ***Configure devices using Boundary-Scan (JTAG), Automatically connect to a cable and identify Boundary-Scan chain***. Click on ***OK*** to proceed.


Auto Assign Configuration Files Query Dialog
Do you want to continue and assign configuration files(s)?
Don't show this message again, save the setting in preference.
Yes
No

15A. Click ***No*** if shown with this window.

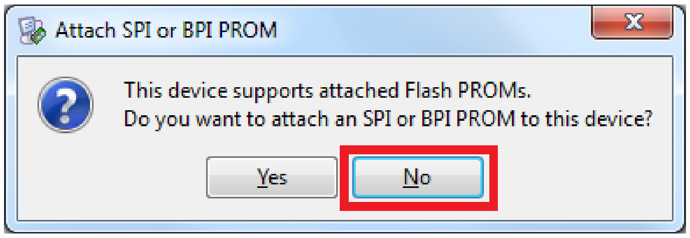

Device Programming Properties - Device 1 Programming Properties
Category
Boundary-Scan
Device 1 (FPGA xc3s1600e)
Device 2 (PROM xcf04s)
Device 3 (PROM xcf04s)
Device 4 (CPLD2 xc2c64a)
Property Name
Value
Verify
FPGA Device Specific Programming Properties
Pulse PROG
OK
Cancel
Apply
Help

16A. Click on ***OK***.



```
Reading C:/Xilinx/13.1/ISE_DS/ISE/spartan3e/data/xc3s1600e.bsd...
INFO:iMPACT:501 - '1': Added Device xc3s1600e successfully.
----------------------------------------------------------------------
----------------------------------------------------------------------
```

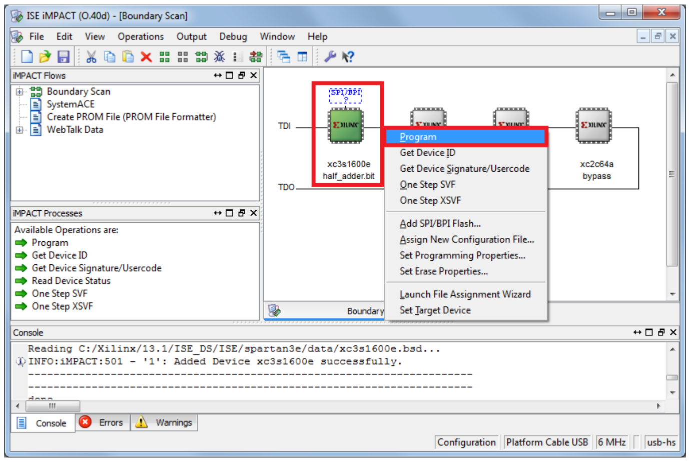
17A. Right click on ***xc3s1600e*** and select ***Assign New Configuration File...*** Then select the .bit file that has been created previously and located in the Xilinx project folder. In this case, the bit file was ***half_adder.bit***.


Attach SPI or BPI PROM

This device supports attached Flash PROMs.
Do you want to attach an SPI or BPI PROM to this device?

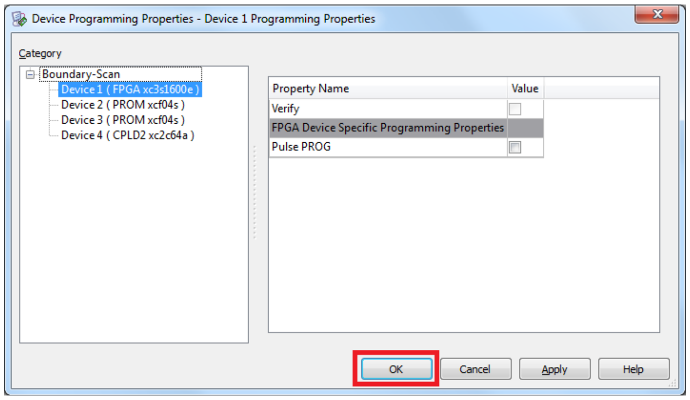
Yes [✓ No]

18A. Click ***No*** if prompted with this window.



```
Reading C:/Xilinx/13.1/ISE_DS/ISE/spartan3e/data/xc3s1600e.bsd...
INFO:iMPACT:501 - '1': Added Device xc3s1600e successfully.
----------------------------------------------------------------------
----------------------------------------------------------------------
done.
```

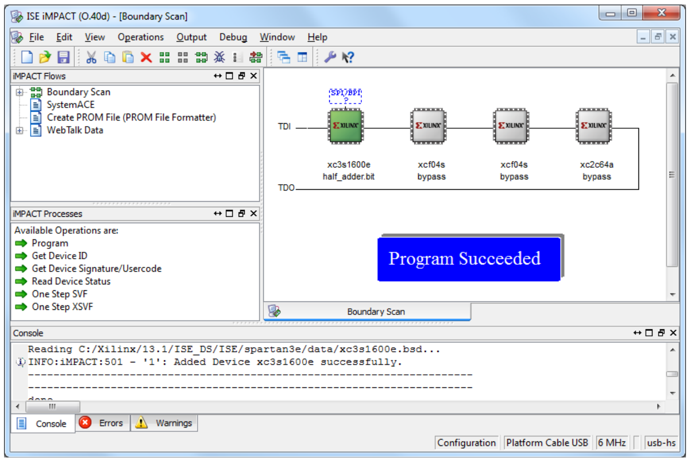
19A. Right click on ***xc3s1600e*** and select ***Program***.


Device Programming Properties - Device 1 Programming Properties

Category
- Boundary-Scan
  - Device 1 (FPGA xc3s1600e)
  - Device 2 (PROM xcf04s)
  - Device 3 (PROM xcf04s)
  - Device 4 (CPLD2 xc2c64a)

Property Name	Value
Verify	☐
FPGA Device Specific Programming Properties	
Pulse PROG	☐

20A. Click on ***OK.***

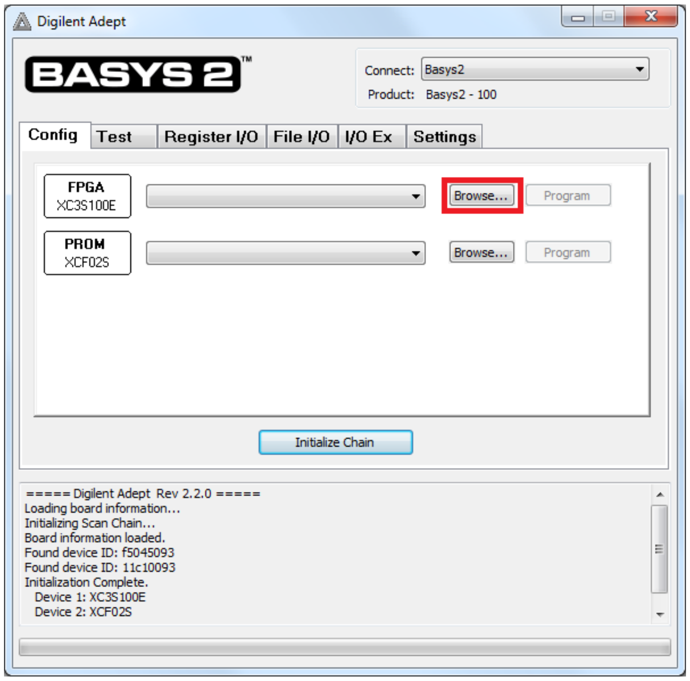


Program Succeeded

```
Reading C:/Xilinx/13.1/ISE_DS/ISE/spartan3e/data/xc3s1600e.bsd...
INFO:iMPACT:501 - '1': Added Device xc3s1600e successfully.
----------------------------------------------------------------------
----------------------------------------------------------------------
done.
```

21A. If the programming succeeds, a message showing "***Program Succeeded***" will be shown. Now the functionality of the design on the FPGA board can be tested.

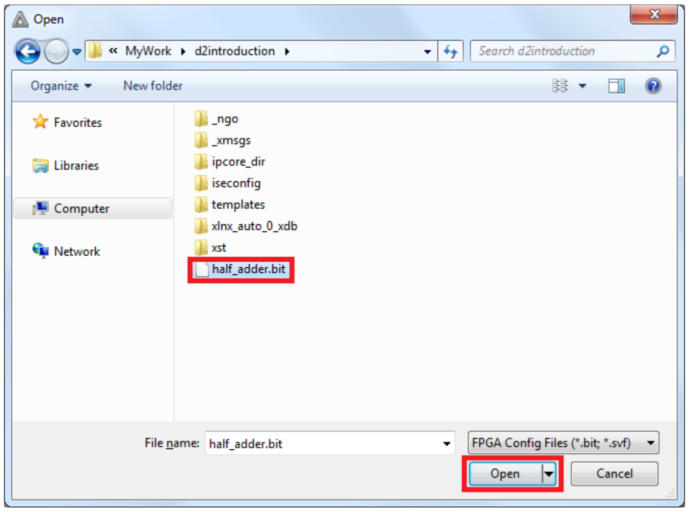
If there is a need to program the FPGA using another .bit file, repeat step 17A to 20A.



```
===== Digilent Adept Rev 2.2.0 =====
Loading board information...
Initializing Scan Chain...
Board information loaded.
Found device ID: f5045093
Found device ID: 11c10093
Initialization Complete.
  Device 1: XC3S100E
  Device 2: XCF02S
```

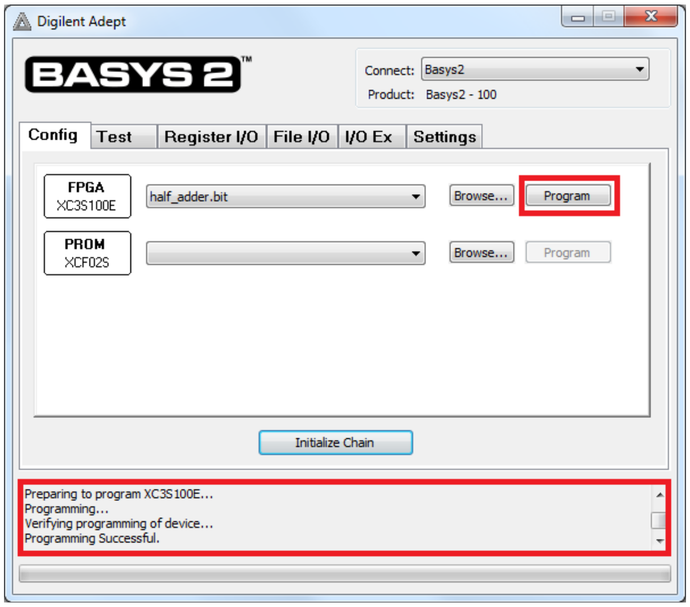
Steps 11B to 13B are applicable only if Spartan 3E (BASYS 2) 100E board is being used.

11B. Open the ***Digilent Adept*** Program. It should detect BASYS2 (Spartan 3E 100E) board and a window similar to the one shown should appear. If the device is not detected, make sure it is connected and powered on. If it is still not detected, force a detection by pressing ***Initialize Chain***.



12B. Click on the ***Browse*** button next to the FPGA device as shown in step 11B. Select the .bit file corresponding to our program and click ***Open***. Example: ***half_adder.bit*** located in the folder where your Xilinx VHDL project is.

Click ***Yes*** if it a warning message about the Startup clock is shown.



```
Preparing to program XC3S100E...
Programming...
Verifying programming of device...
Programming Successful.
```

13B. Now click ***Program***. Dismiss warnings, if any, by clicking ***Yes***.

If the programming succeeds, a message showing "***Programming Successful***" will be shown. Now the functionality of the design on the FPGA board can be tested.

Note for Clocked Circuits

The clock given by the Xilinx development board is 50MHz. To obtain a clock in the order of 1Hz, we will need to divide it using a process as shown below, and clk_div can be used as the clock of the system we are designing

Hint : You might want to do the behavioral simulation without the clock divider, and change the clock name in the main process after introducing the clock divider

```
architecture some_sequential_arch of some_sequential is
signal clk_div :std_logic;  --divided clock
begin

--process to divide clock
clk_divider : process(clk)  --clk is the clock port
variable clk_count : std_logic_vector(25 downto 0) := (others => '0');
begin
   if clk'event and clk = '1' then
      clk_count := clk_count+1;
      clk_div <= clk_count(25);
   end if;
end process;

--main process
main : process(clk_div)
begin
   if clk_div'event and clk_div = '1' then
    ....
    --main code goes here.
   end if;
end process;

end some_sequential_arch;
```

Location Constraints for Spartan 3E 1600E Board

```
#Spartan-3E 1600E Location Constraints
##### Important ##### : Some pins on the Spartan-3E 1600E board are mislablled.
##### Please use the pin locations given in this manual

#Replace SW<0>, BTN_WEST, LED<0>, CLK_50MHZ etc with port names.
#For example, if you have a 3 bit vector "count" to be shown on LEDs, the constraints would be
#NET "count<2>" LOC = "D6" ; count(2)
#NET "count<1>" LOC = "C3" ; count(1)
#NET "count<0>" LOC = "D4" ; count(0)

#DIP/SLIDE Switches
NET "SW<0>" LOC = "L13" ;
NET "SW<1>" LOC = "L14" ;
NET "SW<2>" LOC = "H18" ;
NET "SW<3>" LOC = "N17" ;

#Push-buttons
NET "BTN_EAST" LOC = "H13" ;
NET "BTN_NORTH" LOC = "V4" ;
NET "BTN_SOUTH" LOC = "K17" ;
NET "BTN_WEST" LOC = "D18" ;

#LEDs
NET "LED<7>" LOC = "A8" ;
NET "LED<6>" LOC = "G9" ;
NET "LED<5>" LOC = "A7" ;
NET "LED<4>" LOC = "D13" ;
NET "LED<3>" LOC = "E6" ;
NET "LED<2>" LOC = "D6" ;
NET "LED<1>" LOC = "C3" ;
NET "LED<0>" LOC = "D4" ;

#50MHz Clock
NET "CLK_50MHZ" LOC = "C9" ;
```

Location Constraints for Spartan 3E 100E (BASYS2) Board

```
#Spartan-3E 100E (Basys 2) Location Constraints
#Replace seg<0>, an<0>, Led<0>, CLK_50MHZ etc with port names.
#For example, if you have a 3 bit vector "count" to be shown on LEDs, the constraints would be
#NET "count<2>" LOC = "P7" ; # count(2)
#NET "count<1>" LOC = "M11" ; # count(1)
#NET "count<0>" LOC = "M5” ; # count(0)

# 7seg display (active low)
NET "seg<0>" LOC = "L14";
NET "seg<1>" LOC = "H12";
NET "seg<2>" LOC = "N14";
NET "seg<3>" LOC = "N11";
NET "seg<4>" LOC = "P12";
NET "seg<5>" LOC = "L13";
NET "seg<6>" LOC = "M12";
NET "dp" LOC = "N13";

# Enables of 7seg display (active low)
NET "an<3>" LOC = "K14";
NET "an<2>" LOC = "M13";
NET "an<1>" LOC = "J12";
NET "an<0>" LOC = "F12";

# LEDs
NET "Led<7>" LOC = "G1" ;
NET "Led<6>" LOC = "P4" ;
NET "Led<5>" LOC = "N4" ;
NET "Led<4>" LOC = "N5" ;
NET "Led<3>" LOC = "P6" ;
NET "Led<2>" LOC = "P7" ;
NET "Led<1>" LOC = "M11" ;
NET "Led<0>" LOC = "M5" ;

# DIP/SLIDE switches
NET "sw<7>" LOC = "N3";
NET "sw<6>" LOC = "E2";
NET "sw<5>" LOC = "F3";
NET "sw<4>" LOC = "G3";
NET "sw<3>" LOC = "B4";
NET "sw<2>" LOC = "K3";
NET "sw<1>" LOC = "L3";
NET "sw<0>" LOC = "P11";

#Push-buttons
NET "btn<3>" LOC = "A7";
NET "btn<2>" LOC = "M4";
NET "btn<1>" LOC = "C11";
NET "btn<0>" LOC = "G12";

#clock
NET "CLK_50MHZ" LOC = "B8";
NET "CLK_50MHZ" CLOCK_DEDICATED_ROUTE = FALSE;
```