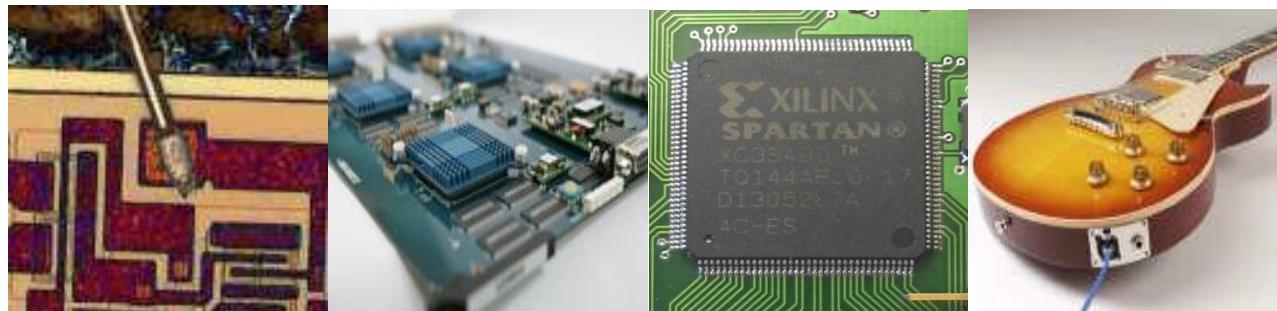




ELE 237

Fundamentals of Digital Systems



The Xilinx Integrated Software
Environment

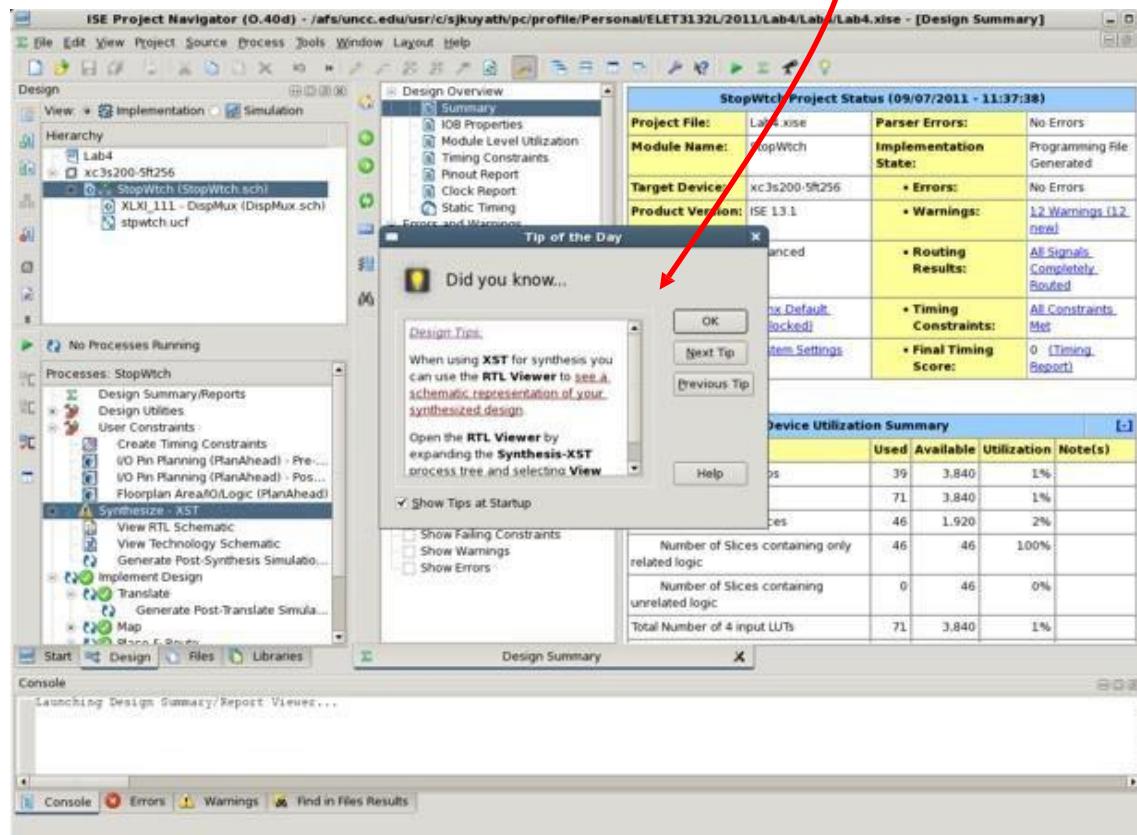
Objectives

- To become competent in using the Xilinx ISE to:
 - Write VHDL code
 - Synthesize VHDL designs
 - Verify the results
 - View:
 - Block Diagram
 - Symbol Diagram
 - Logic Diagram
 - Truth Table
 - K-Maps
 - Simulate the design with ModelSIM
 - Download to the Spartan3 board
 - Verify results



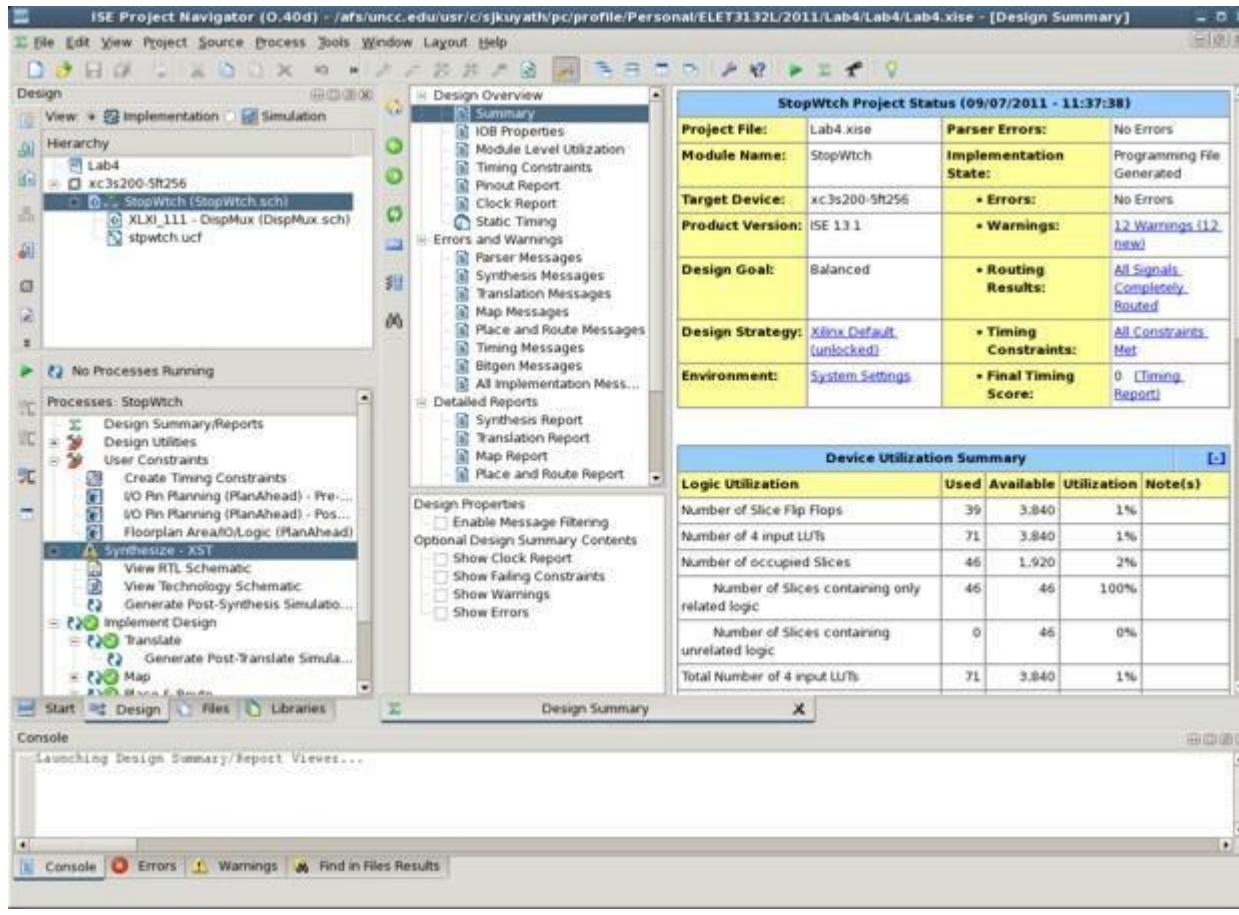
Xilinx Opening Screens

- When you first open the ISE (called the Project Navigator) you get the “Tip of the Day”
 - This can be turned off



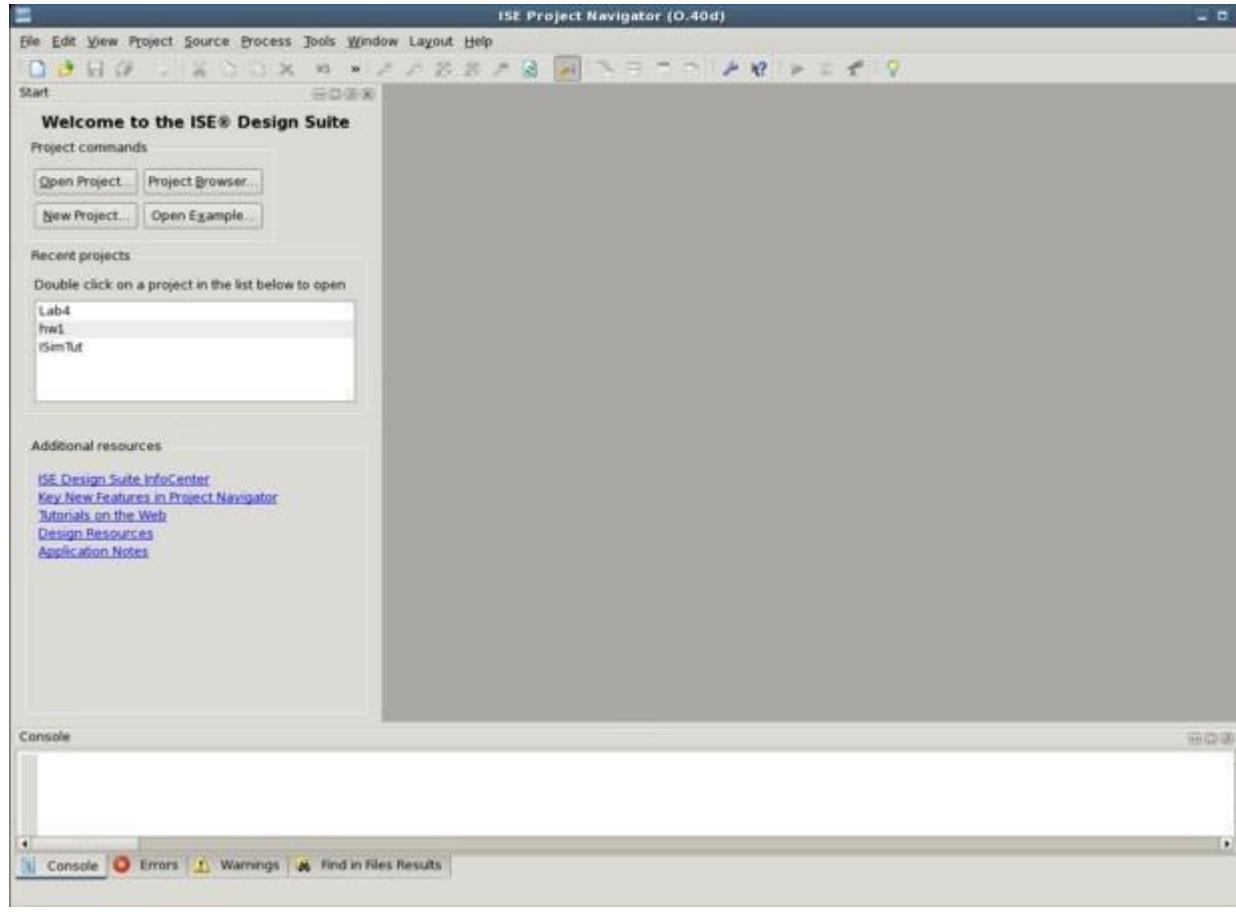
Xilinx Opening Screens

- When you first open the ISE, your most recently saved project will be opened



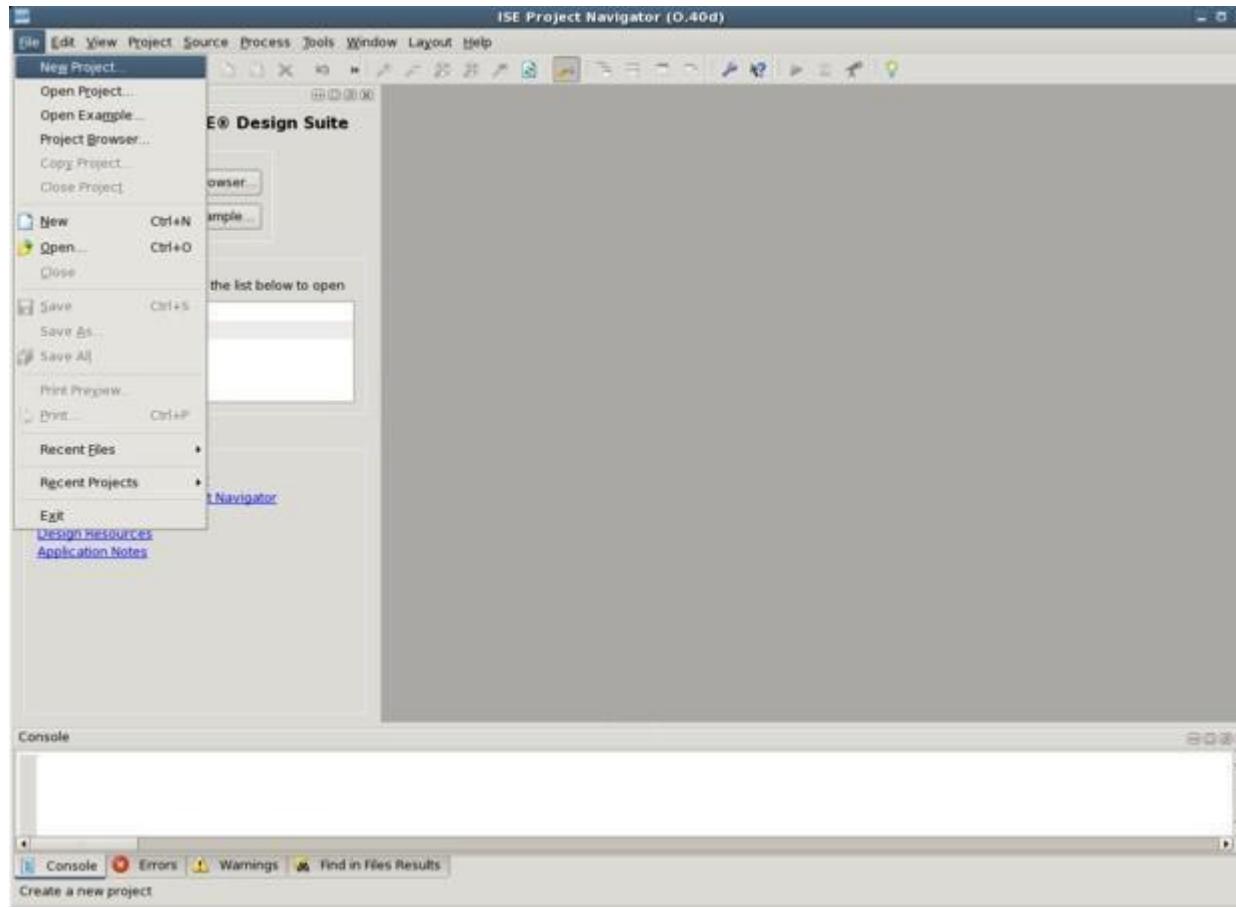
Xilinx Opening Screens

- Unless you have never used the ISE, then you will get a blank screen



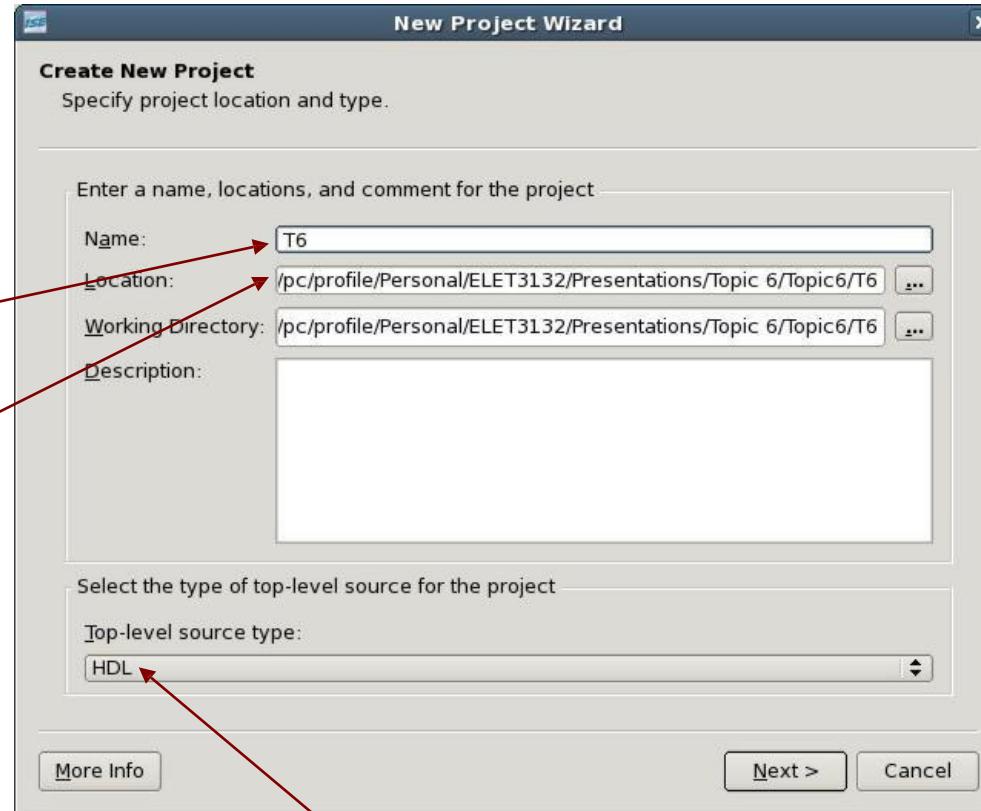
Xilinx Opening Screens

- Either way, you will have to open a New Project
 - Click on the File Menu
 - Click on New Project



Xilinx Opening Screens

- When you click on New Project, the New Project Wizard opens



Enter a name for your project, and make sure the project location is where you would like it to be (this can be changed later, but it is a little awkward doing so).

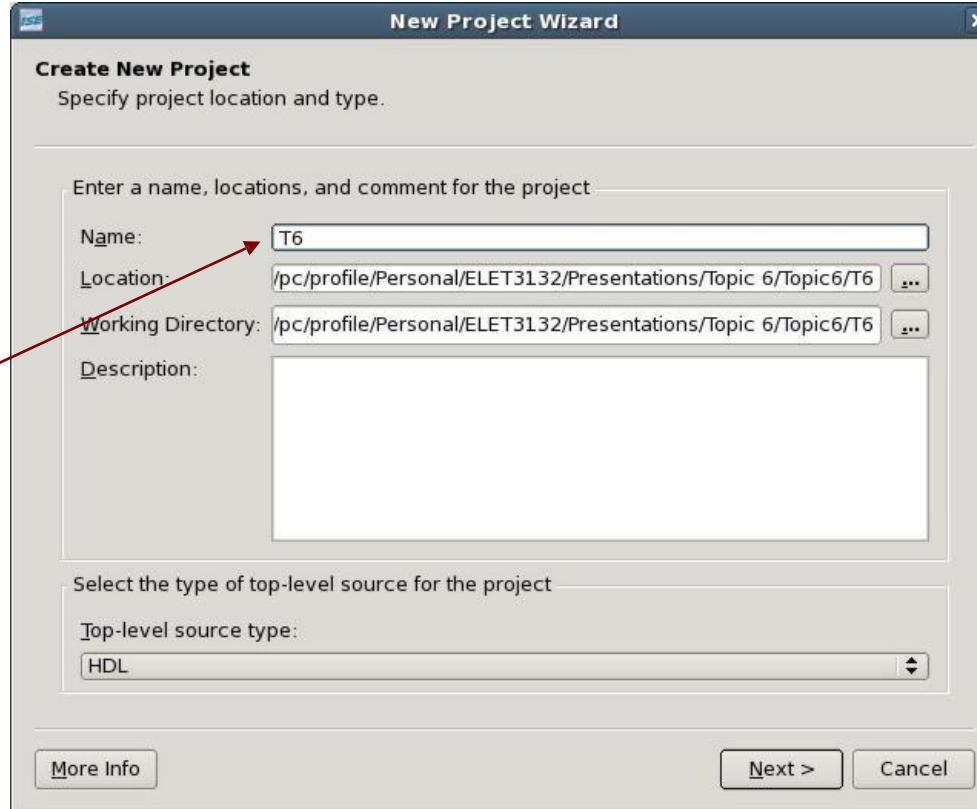
Make sure you have HDL as the Top-Level Module when you are writing VHDL code.



Xilinx Opening Screens

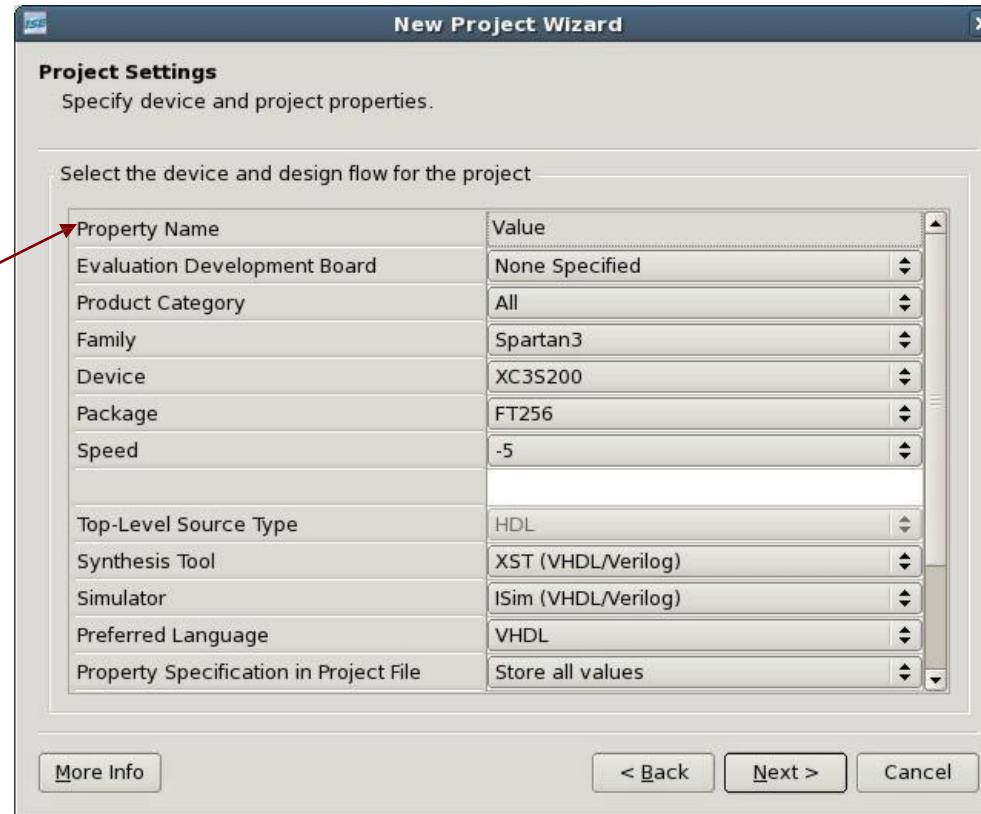
In version 13.1 the file name may be able to have special characters and be longer than 8 characters - IDK

When you have typed in the name, the [NEXT] button is highlighted: Click it!



Xilinx Opening Screens

- When you click on the [NEXT] button, the screen below appears



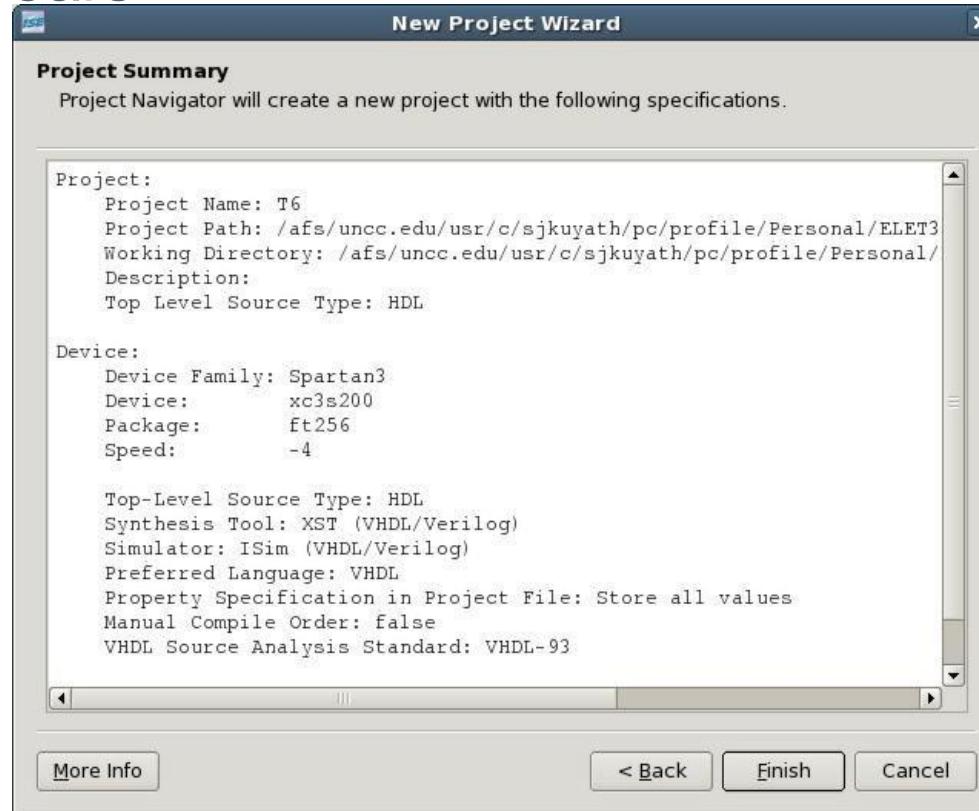
This is mainly a summary page, but make sure the information is the same as it appears here. It specifies the Spartan 3 boards that we will use for projects.

When everything is as it should be, click [NEXT]



Xilinx Opening Screens

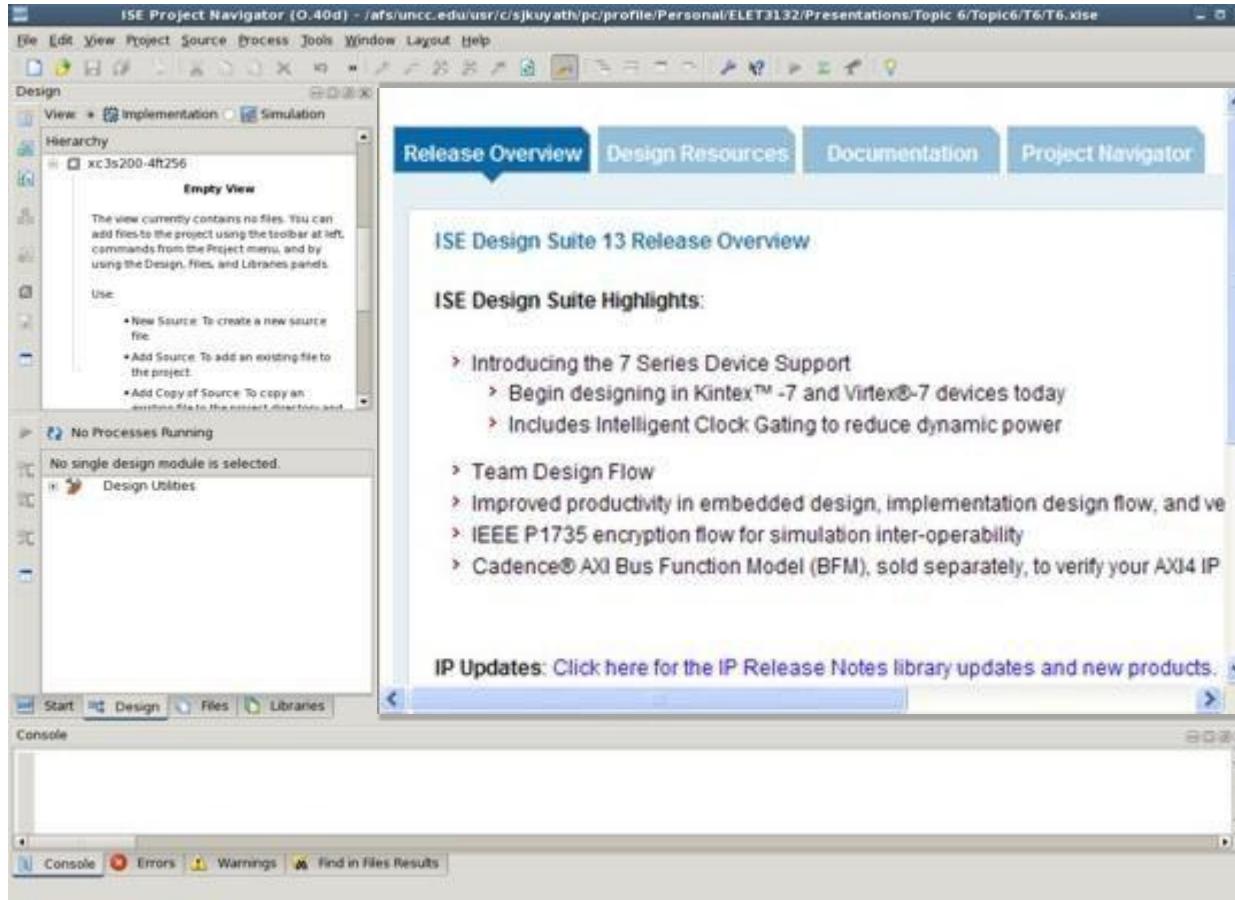
- When you click on the [NEXT] button, the summary screen below appears



If something is incorrect, click on the back button and fix it.
When everything is as it should be, click [NEXT]

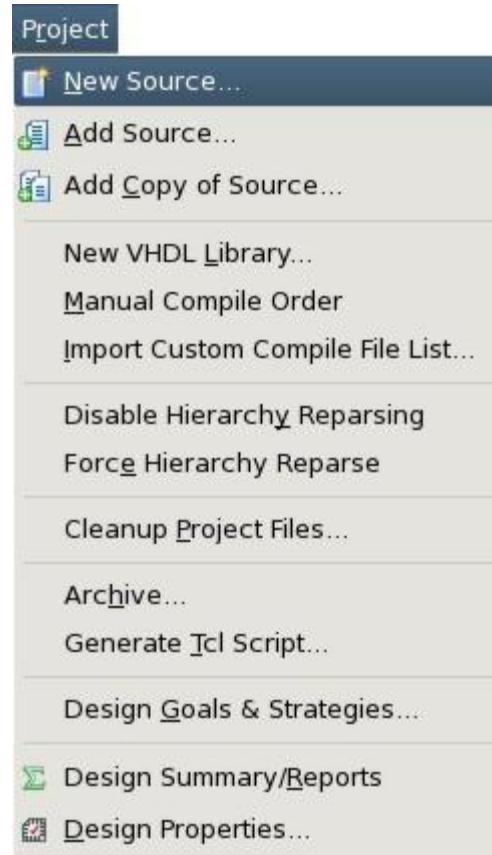
Xilinx Opening Screens

- When you click on the [NEXT] button, the Empty View appears



Xilinx Opening Screens

- We want to create a new source, so click on Project ➔ New Source

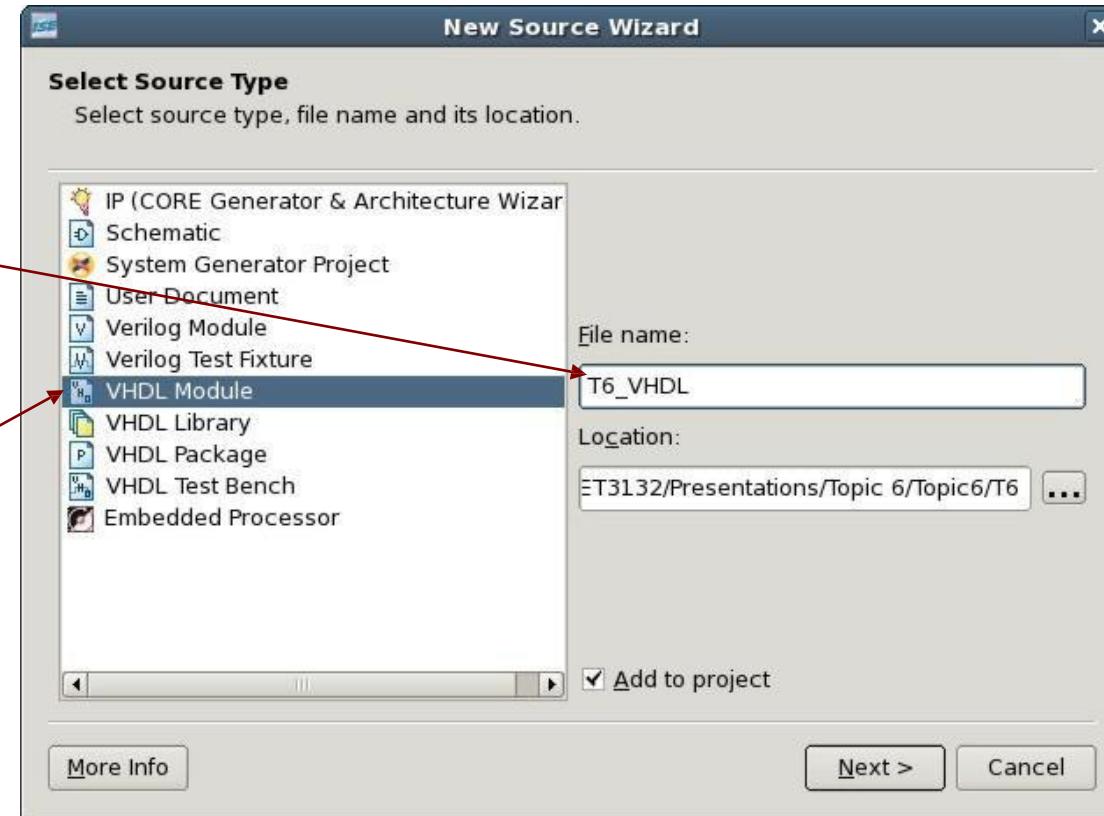


Xilinx Opening Screens

- Provide the Source Filename and Source Type (VHDL Module)

Give the source file a name (no Spaces)

And highlight VHDL Module

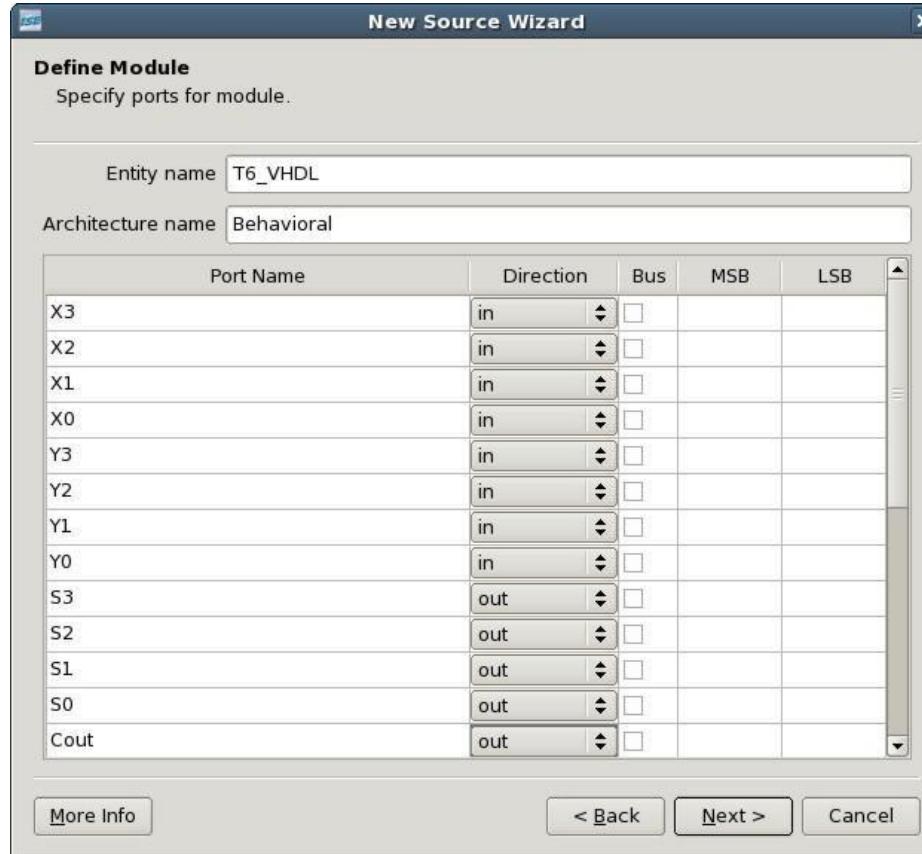


Click the [Next] button



Xilinx Opening Screens

- In this box, you specify the Entity name, the Architecture name, and all the inputs and outputs

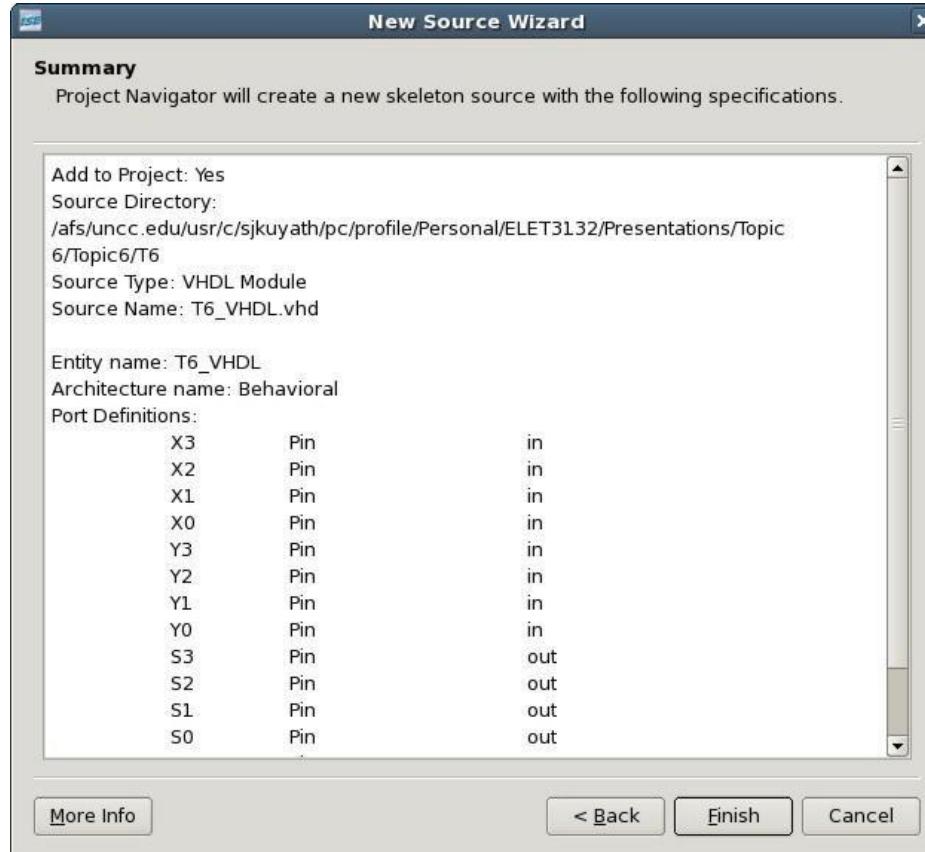


When finished, verify your info and click the [Next] button



Xilinx Opening Screens

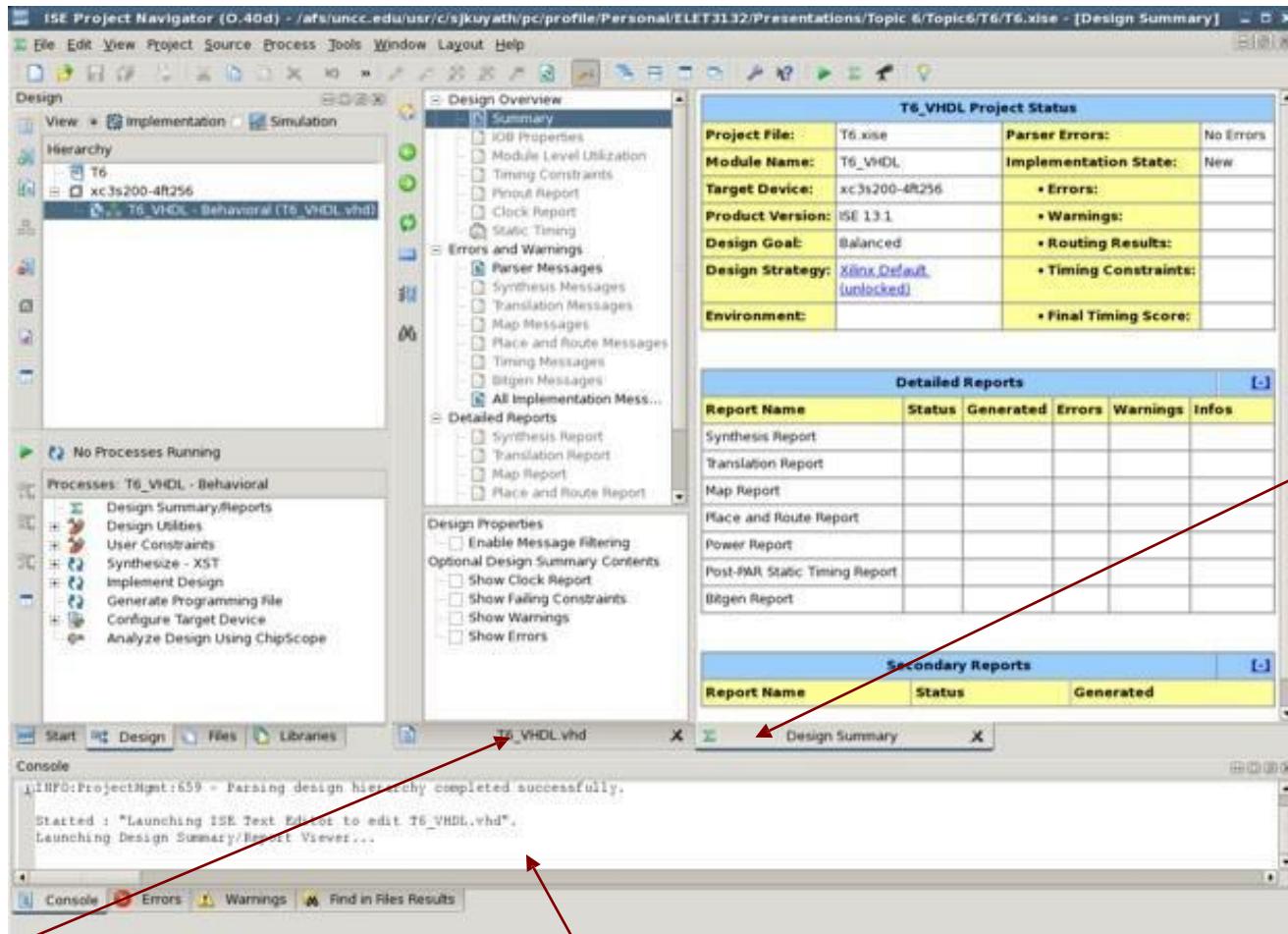
- In this box, you verify the information you just entered



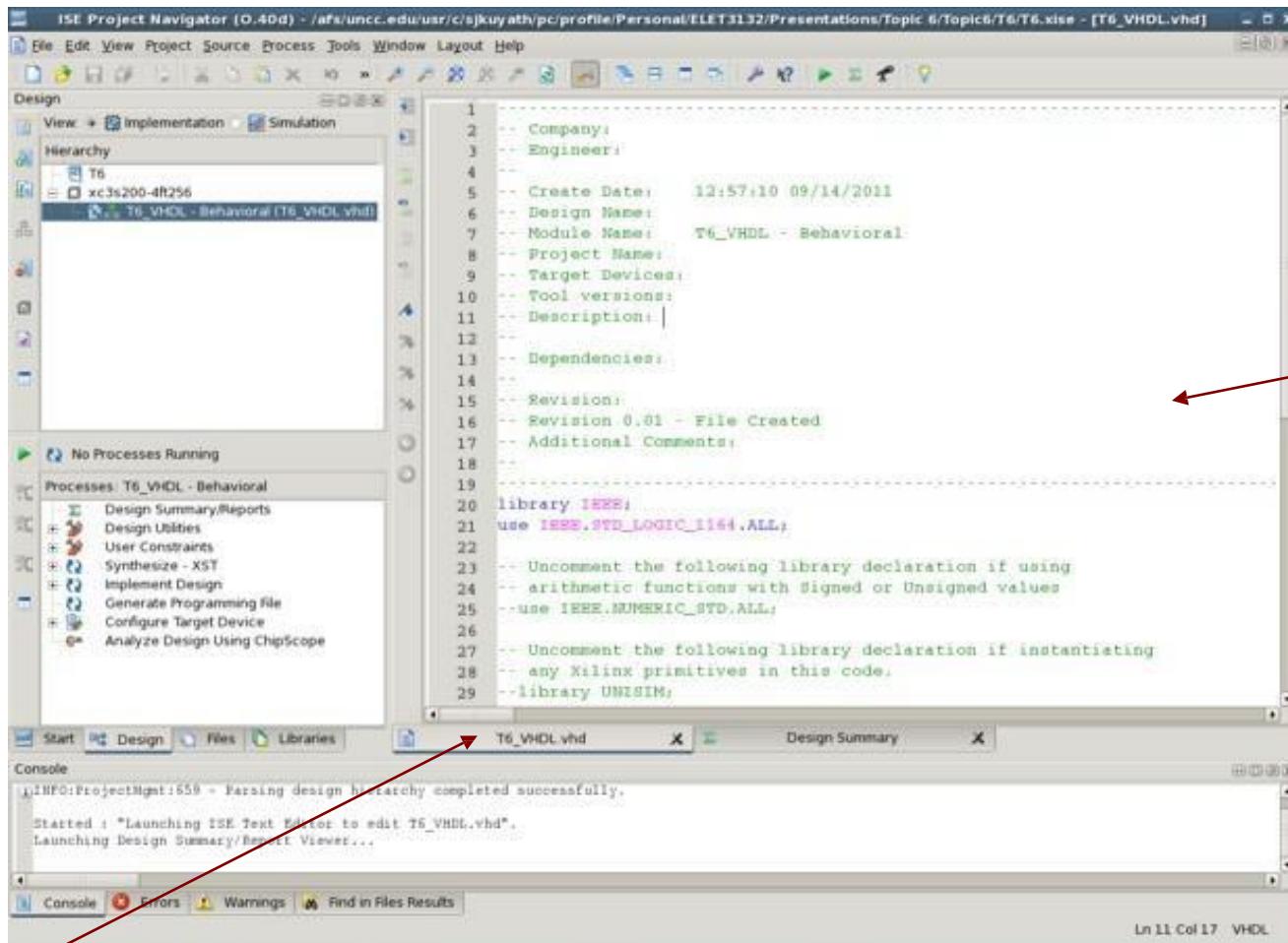
When verified click the [Finish] button



Xilinx Initial Setup



Xilinx Initial Setup



The screenshot shows the Xilinx ISE Project Navigator interface. The main window displays a VHDL file named "T6_VHDL.vhd". The code editor tab shows the beginning of a VHDL file with library declarations and comments. The left sidebar shows the project hierarchy with a top-level module "T6" and a device "xc3s200-4R256". The bottom-left pane shows a "Processes" tree with various design steps like Synthesize-XST, Implement Design, and Analyze Design Using ChipScope. The bottom-right pane is a "Console" showing command-line output related to the project's build process.

```
1 -- Company:
2 -- Engineer:
3 
4 -- Create Date: 12:57:10 09/14/2011
5 Design Name:
6 Module Name: T6_VHDL - Behavioral
7 Project Name:
8 Target Devices:
9 Tool versions:
10 Description:
11 
12 Dependencies:
13 
14 Revision:
15 Revision 0.01 - File Created
16 Additional Comments:
17 
18 
19 library IEEE;
20 use IEEE.STD_LOGIC_1164.ALL;
21 
22 -- Uncomment the following library declaration if using
23 -- arithmetic functions with Signed or Unsigned values
24 --use IEEE.NUMERIC_STD.ALL;
25 
26 -- Uncomment the following library declaration if instantiating
27 -- any Xilinx primitives in this code.
28 --library UNISIM;
29 
```

The VHDL file template is on this tab.



VHDL Template File

```
-- Company:  
-- Engineer:  
--  
-- Create Date: 17:56:19 08/24/07  
-- Design Name:  
-- Module Name: adder4 - Structure  
-- Project Name:  
-- Target Device:  
-- 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;
```

This is the file automatically created
It reserves place for documentation,

```
entity adder4 is  
Port ( Cin : in std_logic;  
X3 : in std_logic;  
X2 : in std_logic;  
X1 : in std_logic;  
X0 : in std_logic;  
Y3 : in std_logic;  
Y2 : in std_logic;  
Y1 : in std_logic;  
Y0 : in std_logic;  
S3 : out std_logic;  
S2 : out std_logic;  
S1 : out std_logic;  
S0 : out std_logic;  
Cout : out std_logic);  
end adder4;  
  
architecture Structure of adder4 is  
  
begin  
  
end Structure;
```



VHDL Template File

```
-- Company:  
-- Engineer:  
--  
-- Create Date: 17:56:19 08/24/07  
-- Design Name:  
-- Module Name: adder4 - Structure  
-- Project Name:  
-- Target Device:  
-- 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;
```

It automatically puts in the most used library and use files

```
entity adder4 is  
Port ( Cin : in std_logic;  
X3 : in std_logic;  
X2 : in std_logic;  
X1 : in std_logic;  
X0 : in std_logic;  
Y3 : in std_logic;  
Y2 : in std_logic;  
Y1 : in std_logic;  
Y0 : in std_logic;  
S3 : out std_logic;  
S2 : out std_logic;  
S1 : out std_logic;  
S0 : out std_logic;  
Cout : out std_logic);  
end adder4;  
  
architecture Structure of adder4 is  
  
begin  
  
end Structure;
```



VHDL Template File

```
-- Company:  
-- Engineer:  
--  
-- Create Date: 17:56:19 08/24/07  
-- Design Name:  
-- Module Name: adder4 - Structure  
-- Project Name:  
-- Target Device:  
-- 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;
```

It automatically creates the entity section; including all the inputs and outputs that were specified and using the name specified

```
entity adder4 is  
Port ( Cin : in  
      std_logic; X3 : in  
      std_logic; X2 : in  
      std_logic; X1 : in  
      std_logic; X0 : in  
      std_logic; Y3 : in  
      std_logic; Y2 : in  
      std_logic; Y1 : in  
      std_logic; Y0 : in  
      std_logic; S3 :  
      out std_logic; S2  
      : out std_logic;  
      S1 : out  
      std_logic; S0 :  
      std_logic;  
      Cout : out std_logic);
```

**architecture Structure of adder4
is**

```
end Structure;
```



VHDL Template File

```
-- Company:  
-- Engineer:  
--  
-- Create Date: 17:56:19 08/24/07  
-- Design Name:  
-- Module Name: adder4 - Structure  
-- Project Name:  
-- Target Device:  
-- 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;
```

It provides the **template** for the Architecture section. This is where the digital design engineer enters all of the logic information.

```
entity adder4 is  
Port ( Cin : in std_logic;  
      X3 : in std_logic;  
      X2 : in std_logic;  
      X1 : in std_logic;  
      X0 : in std_logic;  
      Y3 : in std_logic;  
      Y2 : in std_logic;  
      Y1 : in std_logic;  
      Y0 : in std_logic;  
      S3 : out std_logic;  
      S2 : out std_logic;  
      S1 : out std_logic;  
      S0 : out std_logic;  
      Cout : out std_logic);  
end adder4;  
  
architecture Structure of adder4 is  
  
begin  
  
end Structure;
```



VHDL Template File

```
-- Company:  
-- Engineer:  
--  
-- Create Date: 17:56:19 08/24/07  
-- Design Name:  
-- Module Name: adder4 - Structure  
-- Project Name:  
-- Target Device:  
-- 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;
```

It reserves space for the logic

It reserves space for declarations.

```
entity adder4 is  
Port ( Cin : in std_logic;  
X3 : in std_logic;  
X2 : in std_logic;  
X1 : in std_logic;  
X0 : in std_logic;  
Y3 : in std_logic;  
Y2 : in std_logic;  
Y1 : in std_logic;  
Y0 : in std_logic;  
S3 : out std_logic;  
S2 : out std_logic;  
S1 : out std_logic;  
S0 : out std_logic;  
Cout : out std_logic);  
end adder4;
```

```
architecture Structure of adder4 is
```

```
begin
```

```
end Structure;
```



VHDL Template File

```
entity adder4 is
  Port ( Cin : in
         std_logic; X3 : in
         std_logic; X2 : in
         std_logic; X1 : in
         std_logic; X0 : in
         std_logic; Y3 : in
         std_logic; Y2 : in
         std_logic; Y1 : in
         std_logic; Y0 : in
         std_logic; S3 :
         out std_logic; S2
         : out std_logic;
         S1 : out
         std_logic; S0 :
         out std_logic;
         Cout : out std_logic);
end adder4;

architecture Structure of adder4 is
  SIGNAL c1, c2, c3: STD_LOGIC
  ;
  COMPONENT fulladd
    PORT ( Cin, x, y: IN STD_LOGIC ;
           s, Cout: OUT STD_LOGIC );
  END COMPONENT ;
begin
  stage0: fulladd PORT MAP ( Cin, x0, y0, s0, c1
  ); stage1: fulladd PORT MAP ( c1, x1, y1, s1,
  c2 );
  stage2: fulladd PORT MAP ( c2, x2, y2, s2, c3 )
  ; stage3: fulladd PORT MAP (
    Cin => c3, Cout => Cout, x => x3, y => y3, s => s3 )
  ;
end Structure;
```

Xilinx color codes the text in the file: all keywords are in blue, data types in magenta and variable names in black.



VHDL Template File

```
entity adder4 is
  Port ( Cin : in
         std_logic; X3 : in
         std_logic; X2 : in
         std_logic; X1 : in
         std_logic; X0 : in
         std_logic; Y3 : in
         std_logic; Y2 : in
         std_logic; Y1 : in
         std_logic; Y0 : in
         std_logic; S3 :
         out std_logic; S2
         : out std_logic;
         S1 : out
         std_logic; S0 :
         out std_logic;
         Cout : out std_logic);
end adder4;

architecture Structure of adder4 is
  SIGNAL c1, c2, c3: STD_LOGIC
  ;
  COMPONENT fulladd
    PORT ( Cin, x, y: IN STD_LOGIC ;
           s, Cout: OUT STD_LOGIC );
  END COMPONENT ;
begin
  stage0: fulladd PORT MAP ( Cin, x0, y0, s0, c1
  ); stage1: fulladd PORT MAP ( c1, x1, y1, s1,
  c2 );
  stage2: fulladd PORT MAP ( c2, x2, y2, s2, c3 )
  ; stage3: fulladd PORT MAP (
    Cin => c3, Cout => Cout, x => x3, y => y3, s => s3 )
  ;
end Structure;
```

The entity section is similar to previously seen entity sections (except that each Port Name is on a separate line)



VHDL Template File

```
entity adder4 is
  Port ( Cin : in
         std_logic; X3 : in
         std_logic; X2 : in
         std_logic; X1 : in
         std_logic; X0 : in
         std_logic; Y3 : in
         std_logic; Y2 : in
         std_logic; Y1 : in
         std_logic; Y0 : in
         std_logic; S3 :
         out std_logic; S2
         : out std_logic;
         S1 : out
         std_logic; S0 :
         out std_logic;
         Cout : out std_logic);
end adder4;

architecture Structure of adder4 is
  SIGNAL c1, c2, c3: STD_LOGIC
  ;
  COMPONENT fulladd
    PORT ( Cin, x, y: IN STD_LOGIC ;
           s, Cout: OUT STD_LOGIC );
  END COMPONENT ;
begin
  stage0: fulladd PORT MAP ( Cin, x0, y0, s0, c1
  ); stage1: fulladd PORT MAP ( c1, x1, y1, s1,
  c2 );
  stage2: fulladd PORT MAP ( c2, x2, y2, s2, c3 )
  ; stage3: fulladd PORT MAP (
    Cin => c3, Cout => Cout, x => x3, y => y3, s => s3 )
  ;
end Structure;
```

This architecture section has some differences:

It is making use of some declarations, namely:
Signal and Component (the component structure is the same as the entity structure) and is considered a **sub-circuit – a sub-VHDL file**



VHDL Template File

```
entity adder4 is
  Port ( Cin : in std_logic;
         X3 : in std_logic;
         X2 : in std_logic;
         X1 : in std_logic;
         X0 : in std_logic;
         Y3 : in std_logic;
         Y2 : in std_logic;
         Y1 : in std_logic;
         Y0 : in std_logic;
         S3 : out std_logic;
         S2 : out std_logic;
         S1 : out std_logic;
         S0 : out std_logic;
         Cout : out std_logic);
end adder4;

architecture Structure of adder4 is
  SIGNAL c1, c2, c3: STD_LOGIC ;
  COMPONENT fulladd
    PORT ( Cin, x, y: IN STD_LOGIC ;
           s, Cout: OUT STD_LOGIC );
  END COMPONENT ;
begin
  stage0: fulladd PORT MAP ( Cin, x0, y0, s0, c1 ) ;
  stage1: fulladd PORT MAP ( c1, x1, y1, s1, c2 ) ;
  stage2: fulladd PORT MAP ( c2, x2, y2, s2, c3 ) ;
  stage3: fulladd PORT MAP (
    Cin => c3, Cout => Cout, x => x3, y => y3, s => s3 ) ;
end Structure;
```

“Signal” is declaring 3 signals that will be used in this architecture, but were not declared in the entity or the component.



VHDL Template File

```
entity adder4 is
  Port ( Cin : in std_logic;
         X3 : in std_logic;
         X2 : in std_logic;
         X1 : in std_logic;
         X0 : in std_logic;
         Y3 : in std_logic;
         Y2 : in std_logic;
         Y1 : in std_logic;
         Y0 : in std_logic;
         S3 : out std_logic;
         S2 : out std_logic;
         S1 : out std_logic;
         S0 : out std_logic;
         Cout : out std_logic);
end adder4;

architecture Structure of adder4 is
  SIGNAL c1, c2, c3: STD_LOGIC ;
  COMPONENT fulladd
    PORT ( Cin, x, y: IN STD_LOGIC ;
           s, Cout: OUT STD_LOGIC );
  END COMPONENT ;
begin
  stage0: fulladd PORT MAP ( Cin, x0, y0, s0, c1 ) ;
  stage1: fulladd PORT MAP ( c1, x1, y1, s1, c2 ) ;
  stage2: fulladd PORT MAP ( c2, x2, y2, s2, c3 ) ;
  stage3: fulladd PORT MAP (
    Cin => c3, Cout => Cout, x => x3, y => y3, s => s3 ) ;
end Structure;
```

“Component” is declaring a structure similar to “Entity”, except for the fact that there is no entity called “fulladd” (Remember a Component is really a **SUB-CIRCUIT** and should be defined elsewhere).

“fulladd” is declared as a component, but has no “end” as the entity does.

This is because “fulladd” was declared as an entity in a different file.



VHDL Template File

The ISE is indicating a problem (there is no entity called fulladd) by showing a question mark by a sub-file that should be called fulladd but cannot be found. It is also showing 4 instances of fulladd.

Note also that the ISE realizes it should be a sub-circuit, because it is shown as a sub-section of the adder4 structure

```
33 Port ( X3 : in STD_LOGIC;
34 X2 : in STD_LOGIC;
35 X1 : in STD_LOGIC;
36 X0 : in STD_LOGIC;
37 Y3 : in STD_LOGIC;
38 Y2 : in STD_LOGIC;
39 Y1 : in STD_LOGIC;
40 Y0 : in STD_LOGIC;
41 S3 : out STD_LOGIC;
42 S2 : out STD_LOGIC;
43 S1 : out STD_LOGIC;
44 S0 : out STD_LOGIC;
45 Cout : out STD_LOGIC);
46 end adder4;
47
48 architecture Structure of adder4 is
49 SIGNAL c1, c2, c3: STD_LOGIC;
50 COMPONENT fulladd
51 PORT ( Cin, x, y: IN STD_LOGIC ;
52 s, Cout: OUT STD_LOGIC );
53 END COMPONENT ;
54 begin
55 stage0: fulladd PORT MAP ( Cin, x0, y0, s0, c1 ) ;
56 stage1: fulladd PORT MAP ( c1, x1, y1, s1, c2 ) ;
57 stage2: fulladd PORT MAP ( c2, x2, y2, s2, c3 ) ;
58 stage3: fulladd PORT MAP (
59 Cin => c3, Cout => Cout, x => x3, y => y3, s => s3 ) ;
60 end Structure;
```



VHDL Template File

The ISE is indicating a problem (there is no entity called fulladd) by showing a question mark by a sub-file that should be called fulladd but cannot be found. It is also showing 4 instances of fulladd.

Note also that the ISE realizes it should be a sub-circuit, because it is shown as a sub-section of the adder4 structure

Double click on fulladd

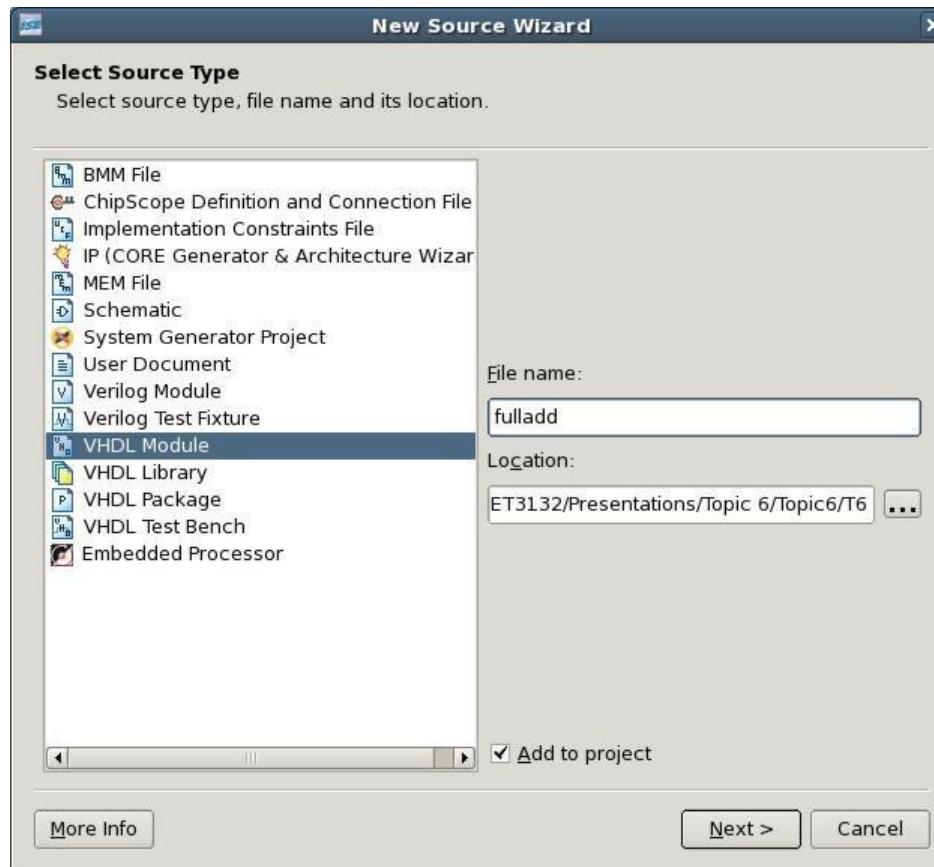
```
33  Port ( X3 : in STD_LOGIC;
34    X2 : in STD_LOGIC;
35    X1 : in STD_LOGIC;
36    X0 : in STD_LOGIC;
37    Y3 : in STD_LOGIC;
38    Y2 : in STD_LOGIC;
39    Y1 : in STD_LOGIC;
40    Y0 : in STD_LOGIC;
41    S3 : out STD_LOGIC;
42    S2 : out STD_LOGIC;
43    S1 : out STD_LOGIC;
44    S0 : out STD_LOGIC;
45    Cout : out STD_LOGIC);
46 end adder4;
47
48 architecture Structure of adder4 is
49   SIGNAL c1, c2, c3: STD_LOGIC;
50   COMPONENT fulladd
51     PORT ( Cin, x, y: IN STD_LOGIC ;
52            s, Cout: OUT STD_LOGIC );
53   END COMPONENT ;
54 begin
55   stage0: fulladd PORT MAP ( Cin, x0, y0, s0, c1 ) ;
56   stage1: fulladd PORT MAP ( c1, x1, y1, s1, c2 ) ;
57   stage2: fulladd PORT MAP ( c2, x2, y2, s2, c3 ) ;
58   stage3: fulladd PORT MAP (
59     Cin => c3, Cout => Cout, x => x3, y => y3, s => s3 ) ;
60 end Structure;
61
```



VHDL Template File

When you double click on fulladd, the New Source window pops up with the filename already filled in.

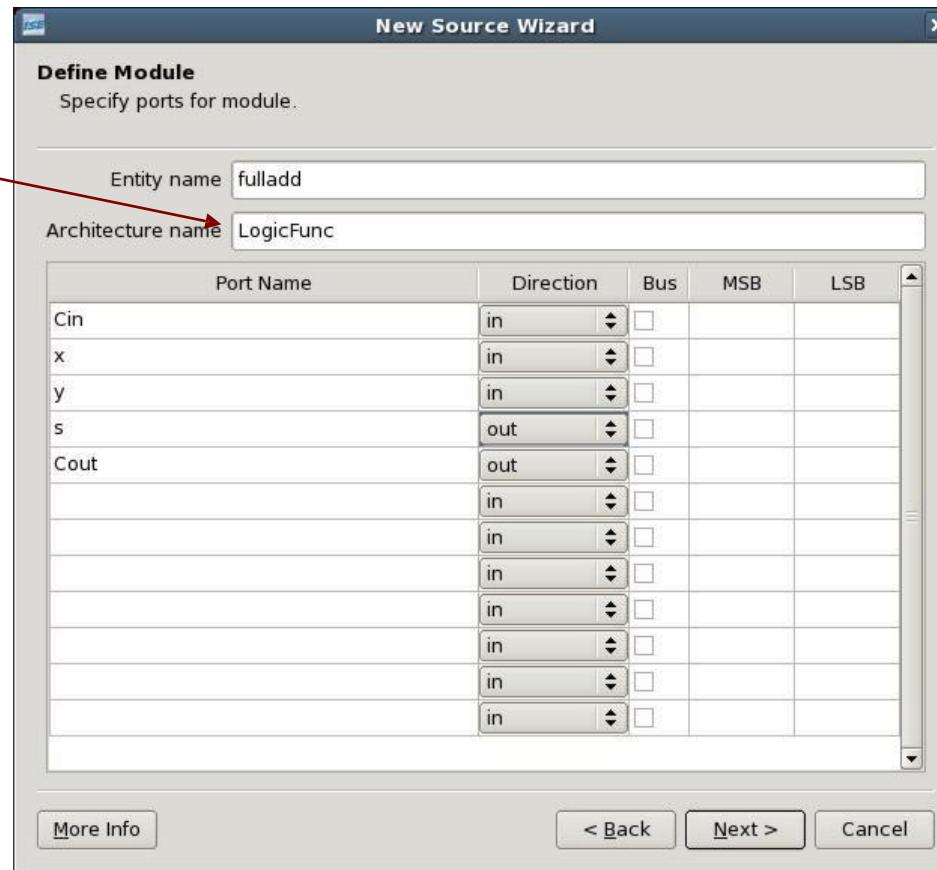
The ISE is wanting to know what type of file fulladd is. We will be entering a VHDL file for fulladd, so click on VHDL Module and then click [Next]



VHDL Template File

Fill in the Architecture name (the entity name is already filled in for you) and all the port names, including their directions

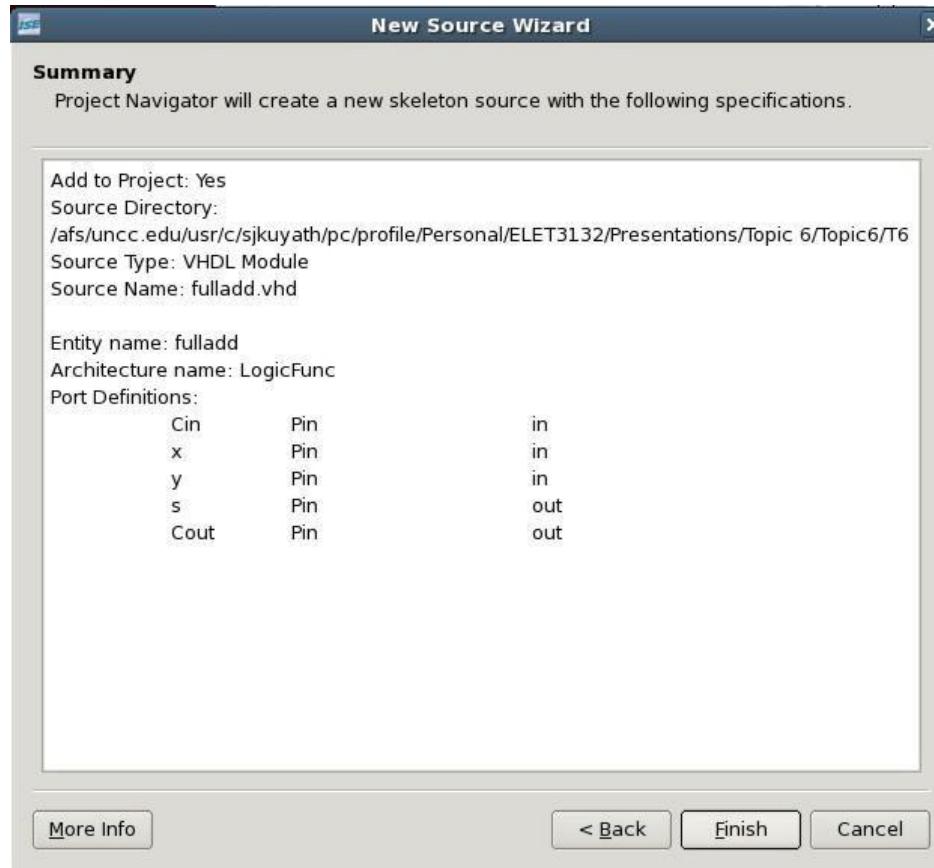
Then click [Next]



VHDL Template File

The information/verification window appears. If everything is as it should be:

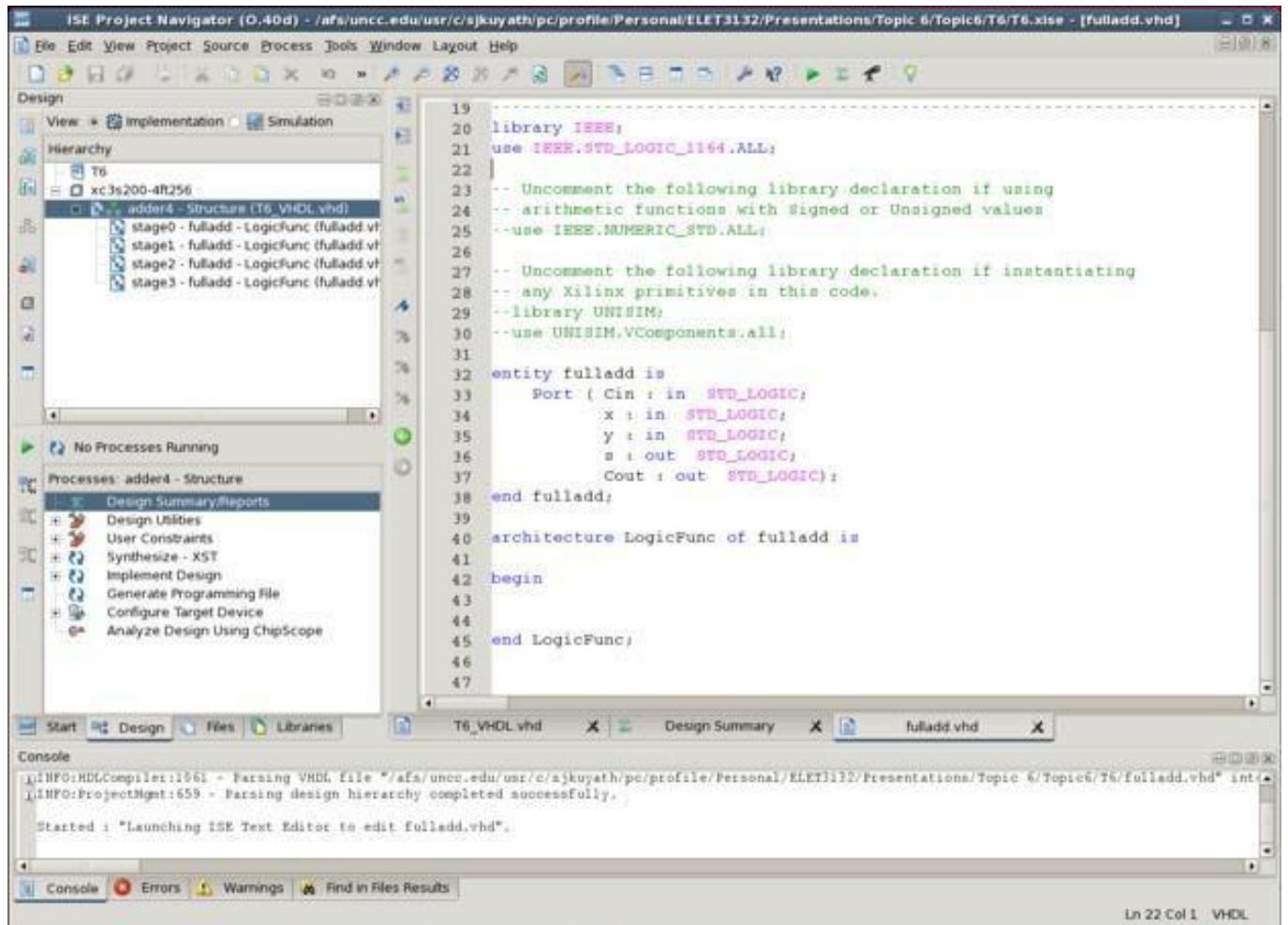
Click [Finish]



VHDL Template File

The template for the fulladd VHDL file pops up, with the entity section filled in.

Note that the fulladd icon no longer has a question mark and is named appropriately



The screenshot shows the Xilinx ISE Project Navigator interface. The left pane displays the project hierarchy under the 'Design' tab, specifically the 'Implementation' view. It shows a top-level entity 'T6' which contains an 'adder4' structure. Inside 'adder4', there are four 'fulladd' components. The 'fulladd' component is currently selected. The right pane shows the VHDL code for the 'fulladd' entity. The code is a template with placeholder comments and declarations. The 'entity' section includes port declarations for Cin, x, y, s, and Cout. The 'architecture' section is named 'LogicFunc'. The code ends with an 'end LogicFunc;' statement. At the bottom of the code editor, the status bar indicates 'Ln 22 Col 1 VHDL'. The bottom navigation bar includes tabs for 'Console', 'Errors', 'Warnings', and 'Find in Files Results'.

```
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 fulladd is
    Port ( Cin : in STD_LOGIC;
           x : in STD_LOGIC;
           y : in STD_LOGIC;
           s : out STD_LOGIC;
           Cout : out STD_LOGIC);
end fulladd;
architecture LogicFunc of fulladd is
begin
end LogicFunc;
```



VHDL Template File

```
LIBRARY ieee ;  
USE ieee.std_logic_1164.all ;  
  
ENTITY fulladd IS  
    PORT ( Cin, x, y : IN STD_LOGIC ;  
           s, Cout : OUT STD_LOGIC ) ;  
END fulladd ;  
  
ARCHITECTURE LogicFunc OF fulladd IS  
BEGIN  
    s <= x XOR y XOR Cin ;  
    Cout <= (x AND y) OR (Cin AND x) OR (Cin AND y) ;  
END LogicFunc ;
```

The fulladd VHDL module is shown above. Note that architecture declares the logic of the full adder.

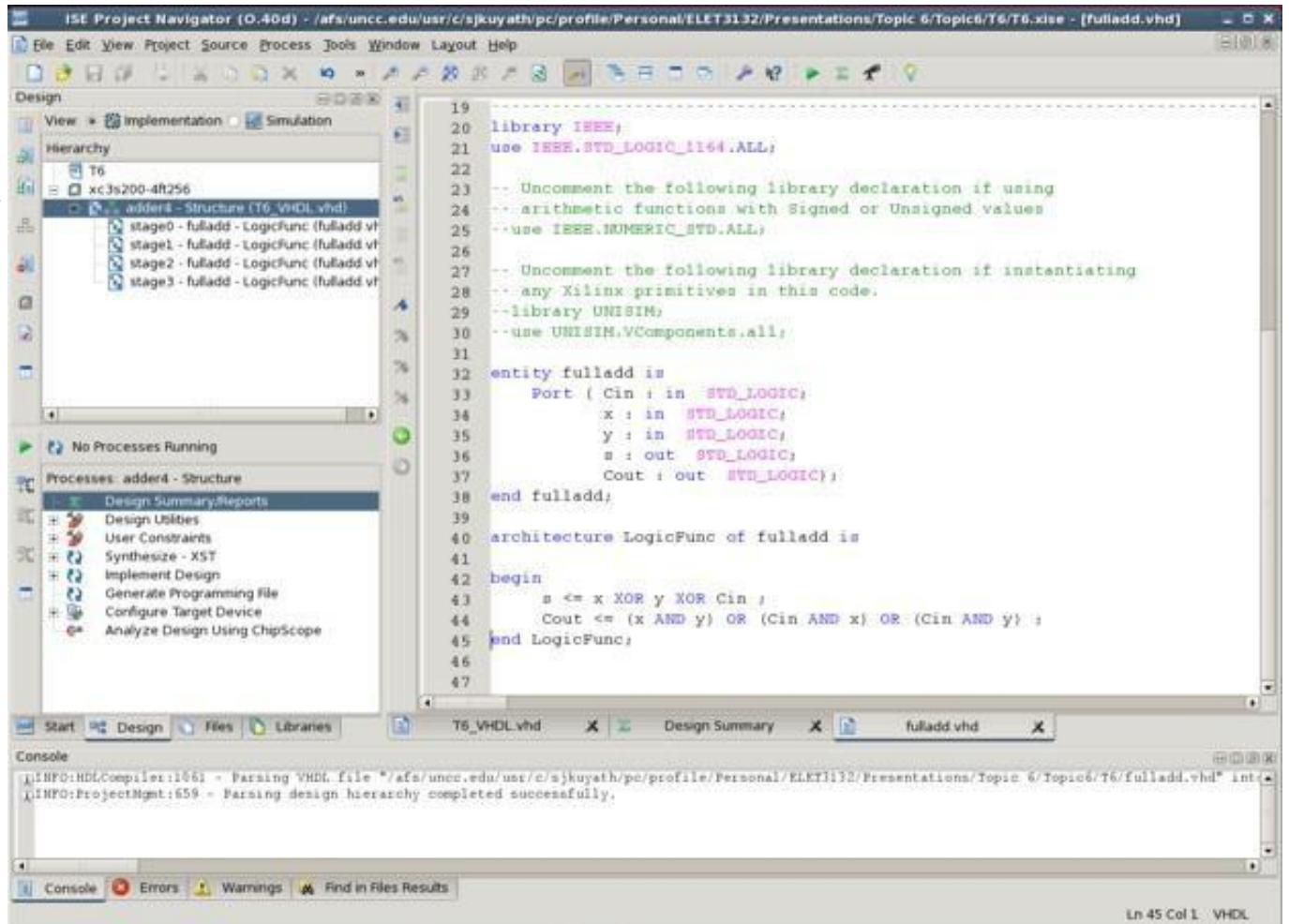
This is VERY important!! It shows that the ISE will allow you to create a component for use in other files: Once the component is debugged, it can be used over and over.



VHDL Template File

The module for the full adder is entered and saved.

It is now time to compile the design and verify that it is functioning properly



The screenshot shows the ISE Project Navigator interface. The left pane displays the project hierarchy for a 'T6' project, which contains an 'add4 - Structure (T6_VHDL.vhd)' file. This file includes four sub-components named 'stage0', 'stage1', 'stage2', and 'stage3', each corresponding to a 'fulladd - LogicFunc' component. The right pane shows the source code for 'T6_VHDL.vhd'. The code defines an entity 'fulladd' with four ports: Cin (STD_LOGIC), x (STD_LOGIC), y (STD_LOGIC), s (STD_LOGIC), and Cout (STD_LOGIC). It also defines an architecture 'LogicFunc' for 'fulladd' that implements the logic $s \leftarrow x \text{ XOR } y \text{ XOR } \text{Cin}$ and $\text{Cout} \leftarrow (\text{x AND y}) \text{ OR } (\text{Cin AND x}) \text{ OR } (\text{Cin AND y})$. The bottom pane shows the 'Console' output, which indicates that the VHDL file was parsed successfully. The status bar at the bottom right shows 'Ln 45 Col 1 VHDL'.

```
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 fulladd is
    Port ( Cin : in STD_LOGIC;
           x : in STD_LOGIC;
           y : in STD_LOGIC;
           s : out STD_LOGIC;
           Cout : out STD_LOGIC);
end fulladd;

architecture LogicFunc of fulladd is
begin
    s <= x XOR y XOR Cin ;
    Cout <= (x AND y) OR (Cin AND x) OR (Cin AND y) ;
end LogicFunc;
```

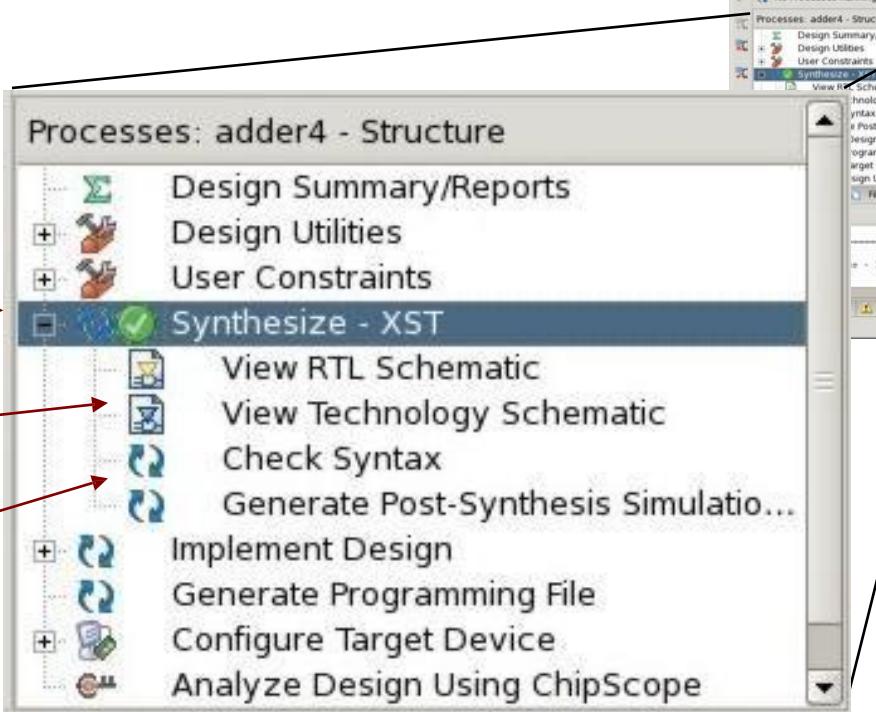


Synthesize

The next step is to synthesize the design. The only step you must take is to double click on

Synthesize – XST

You may also double click on other actions or reports



```
iSE Project Navigator (0.40d) : /afs/uncc.edu/usr/c/sjkuayath/pc/profile/Personal/ELET3132/Presentations/Topic 6/Topic6/T6/T6.xise - [T6_VHDL.vhd]

File Edit View Project Source Process Tools Window Layout Help
Design
View + Implementation Simulation
Hierarchy
T6
  xc3s200-4t256
    adder4 - Structure (T6_VHDL.vhd)
      stage0 - fulladd - LogicFunc (fulladd.vf)
      stage1 - fulladd - LogicFunc (fulladd.vf)
      stage2 - fulladd - LogicFunc (fulladd.vf)
      stage3 - fulladd - LogicFunc (fulladd.vf)
No Processes Running
Processes: adder4 - Structure
  Design Summary/Reports
  Design Utilities
  User Constraints
  Synthesize - XST
  View RTL Schematic
  View Technology Schematic
  Check Syntax
  Generate Post-Synthesis Simulation...
  Implement Design
  Generate Programming File
  Configure Target Device
  Analyze Design Using ChipScope
  Files Libraries
Design Summary (Synthesized) T6_VHDL.vhd
Ln 60 Col 10 VHDL

entity adder4 is
  Port ( Cin : in std_logic;
         X3 : in std_logic;
         X2 : in std_logic;
         X1 : in std_logic;
         X0 : in std_logic;
         Y3 : in std_logic;
         Y2 : in std_logic;
         Y1 : in std_logic;
         Y0 : in std_logic;
         S3 : out std_logic;
         S2 : out std_logic;
         S1 : out std_logic;
         S0 : out std_logic;
         Cout : out std_logic);
end adder4;

architecture Structure of adder4 is
  SIGNAL c1, c2, c3: STD_LOGIC;
  COMPONENT fulladd
    PORT ( Cin, x, y: IN STD_LOGIC;
           s, Cout: OUT STD_LOGIC );
  END COMPONENT ;
begin
  stage0: fulladd PORT MAP ( Cin, x0, y0, s0, c1 );
  stage1: fulladd PORT MAP ( c1, x1, y1, s1, c2 );
  stage2: fulladd PORT MAP ( c2, x2, y2, s2, c3 );
  stage3: fulladd PORT MAP ( c3, x3, y3, s3, Cout );
end;
```

Note that the “adder4” module is highlighted

ver 13 won’t allow you to synthesize the component, but ver 7 did

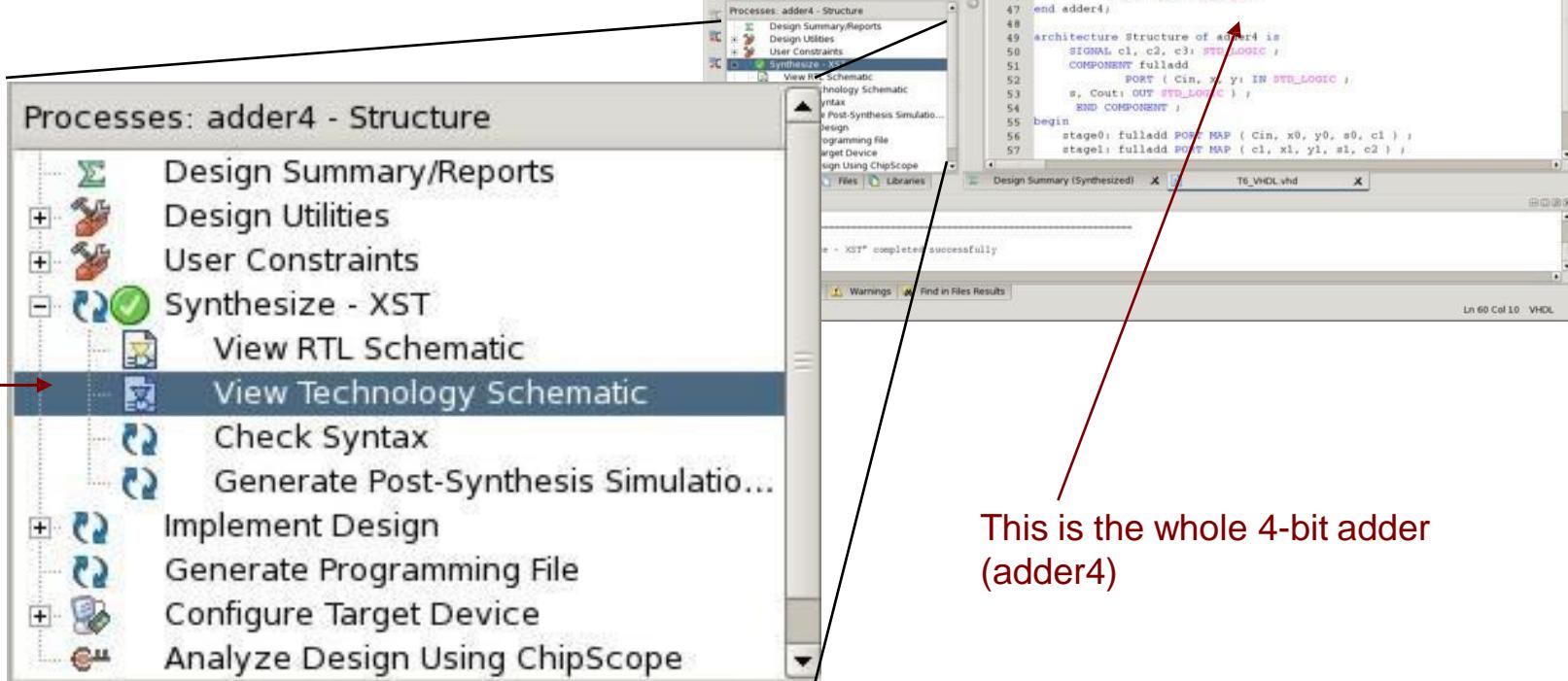


View Technology Schematic

To view:

1. The Block DIAGRAM
2. The Symbol Diagram
3. The Logic Diagram, Truth Table, and Kmap

Double Click on View Technology Schematic



This is the whole 4-bit adder
(adder4)



View Technology Schematic

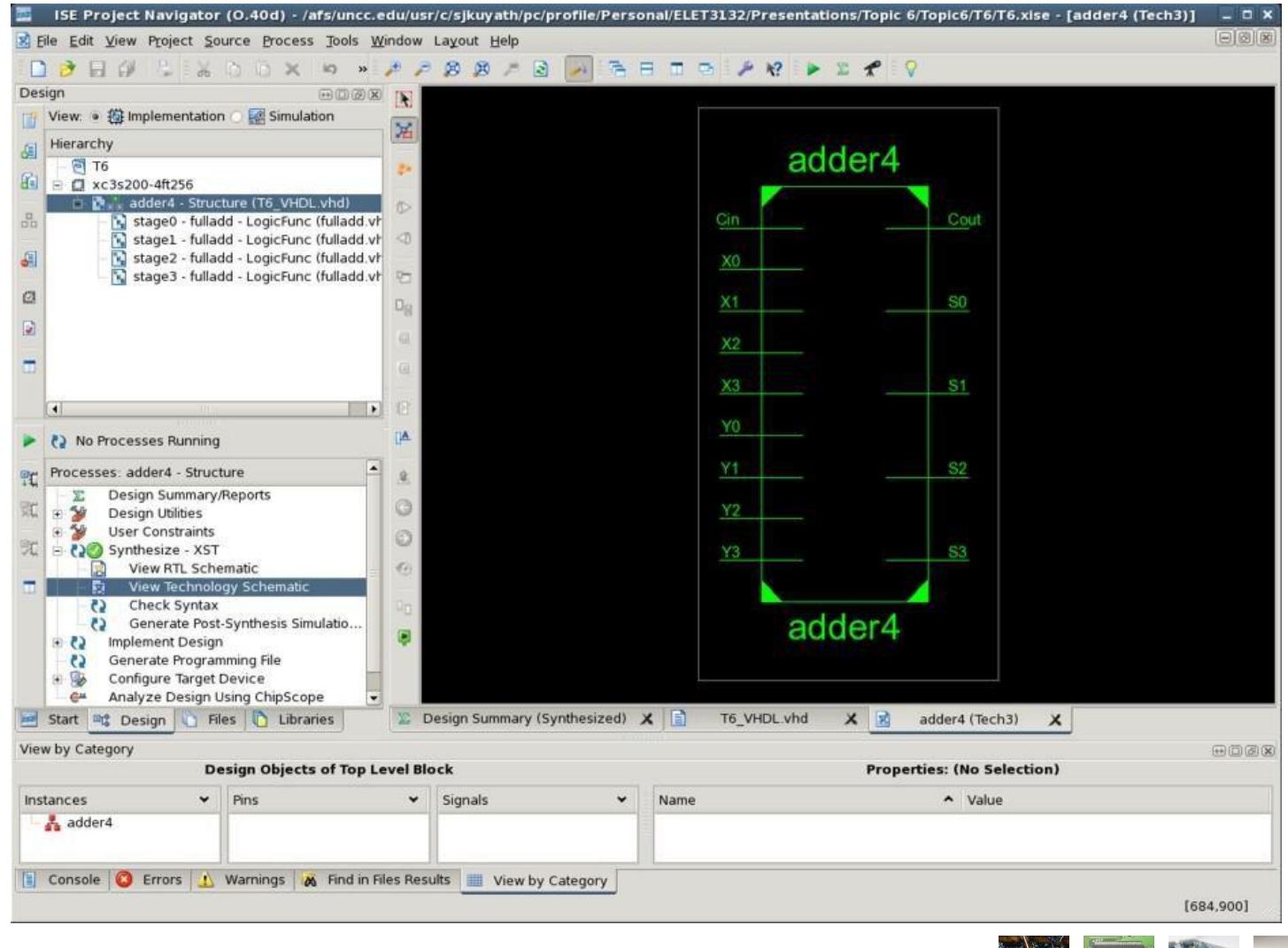
This dialog box appears. Highlight the lower radio button and click OK



View Technology Schematic

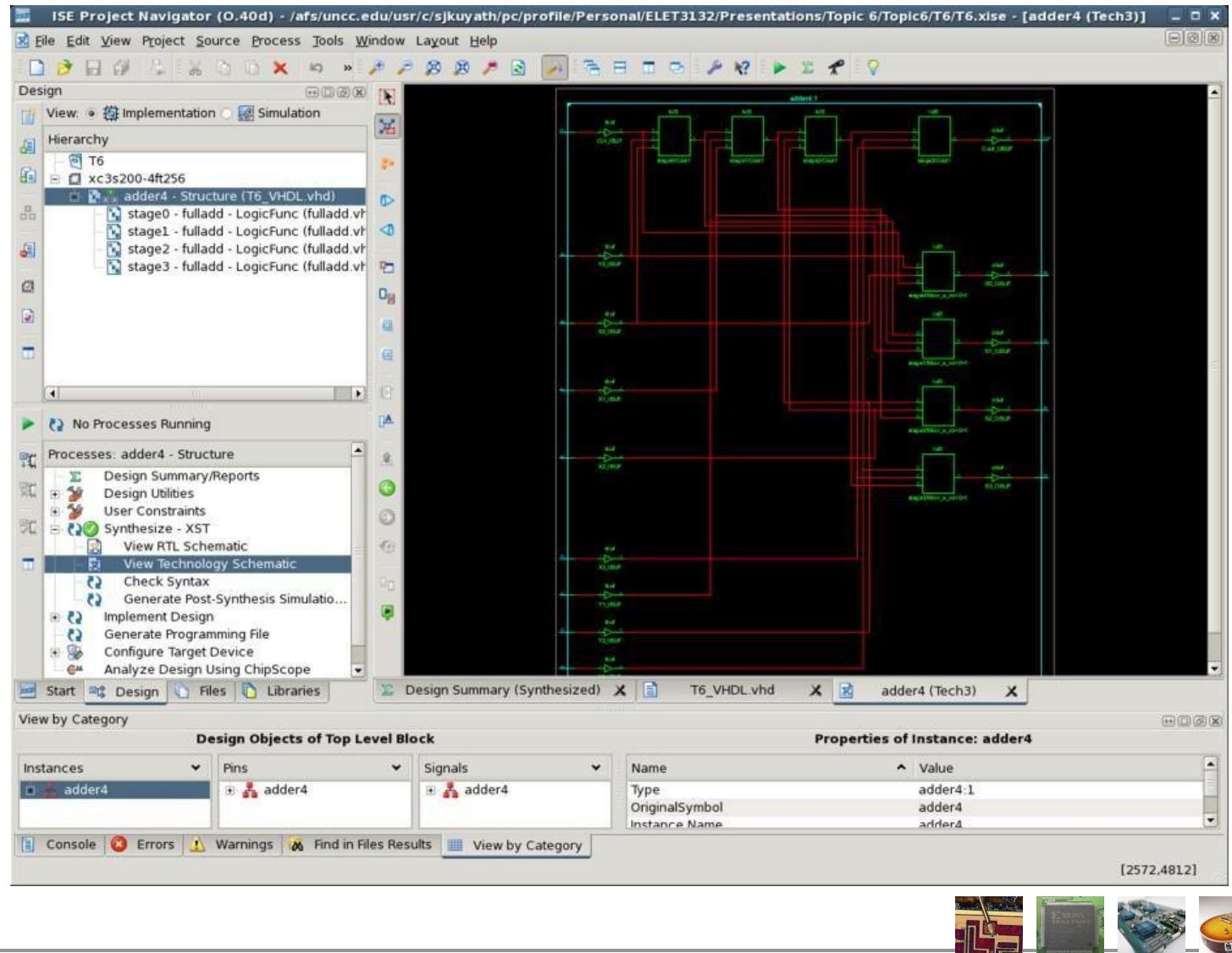
You'll see the complete symbol for the "adder4". It shows the **ENTITY** (the inputs and the outputs, but no functionality).

Double click on the entity.



View Technology Schematic

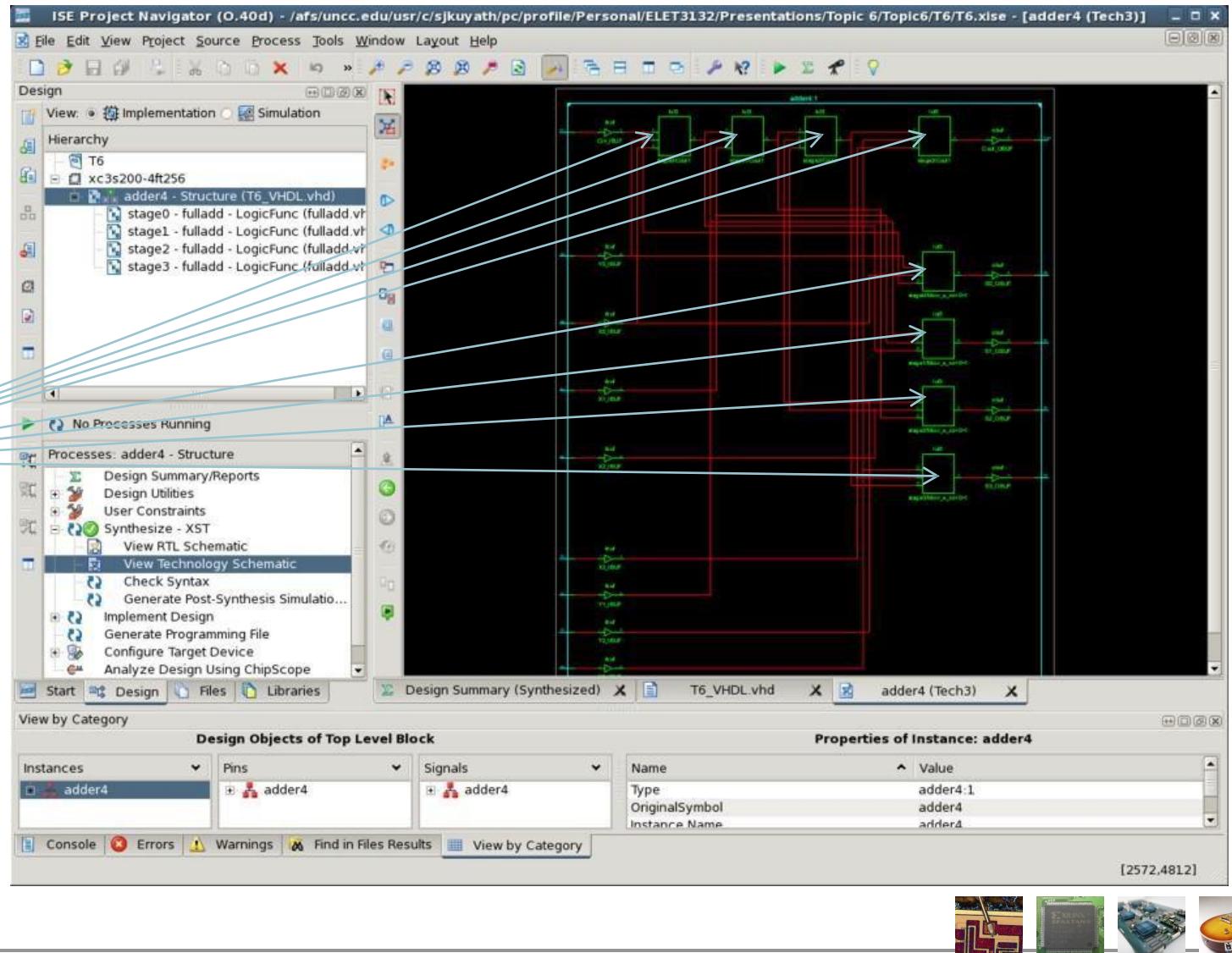
When you double click on the adder4 symbol, the entity, the functionality appears



View Technology Schematic

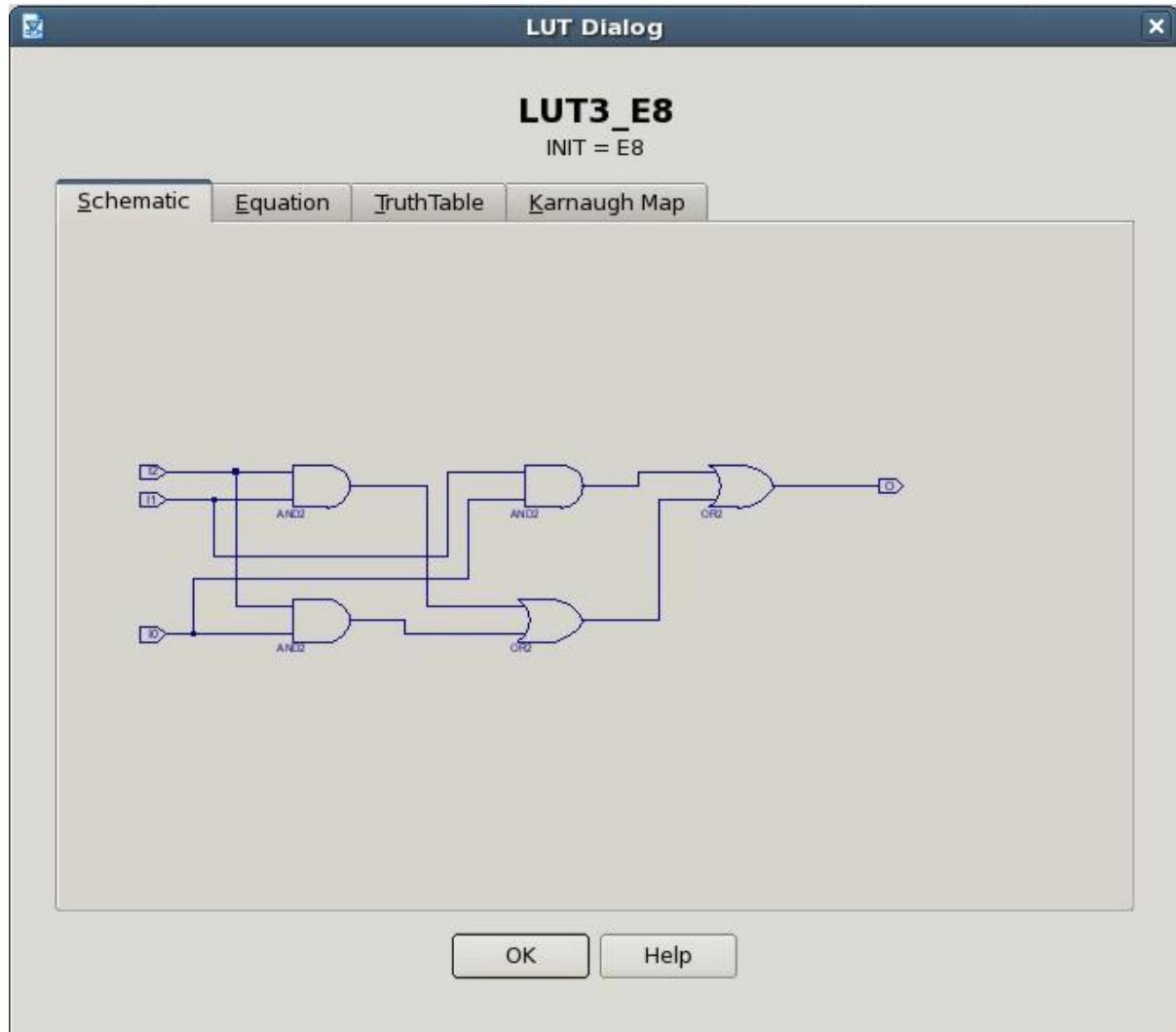
Double click on any of the LUTs and a new box appears

LUTs



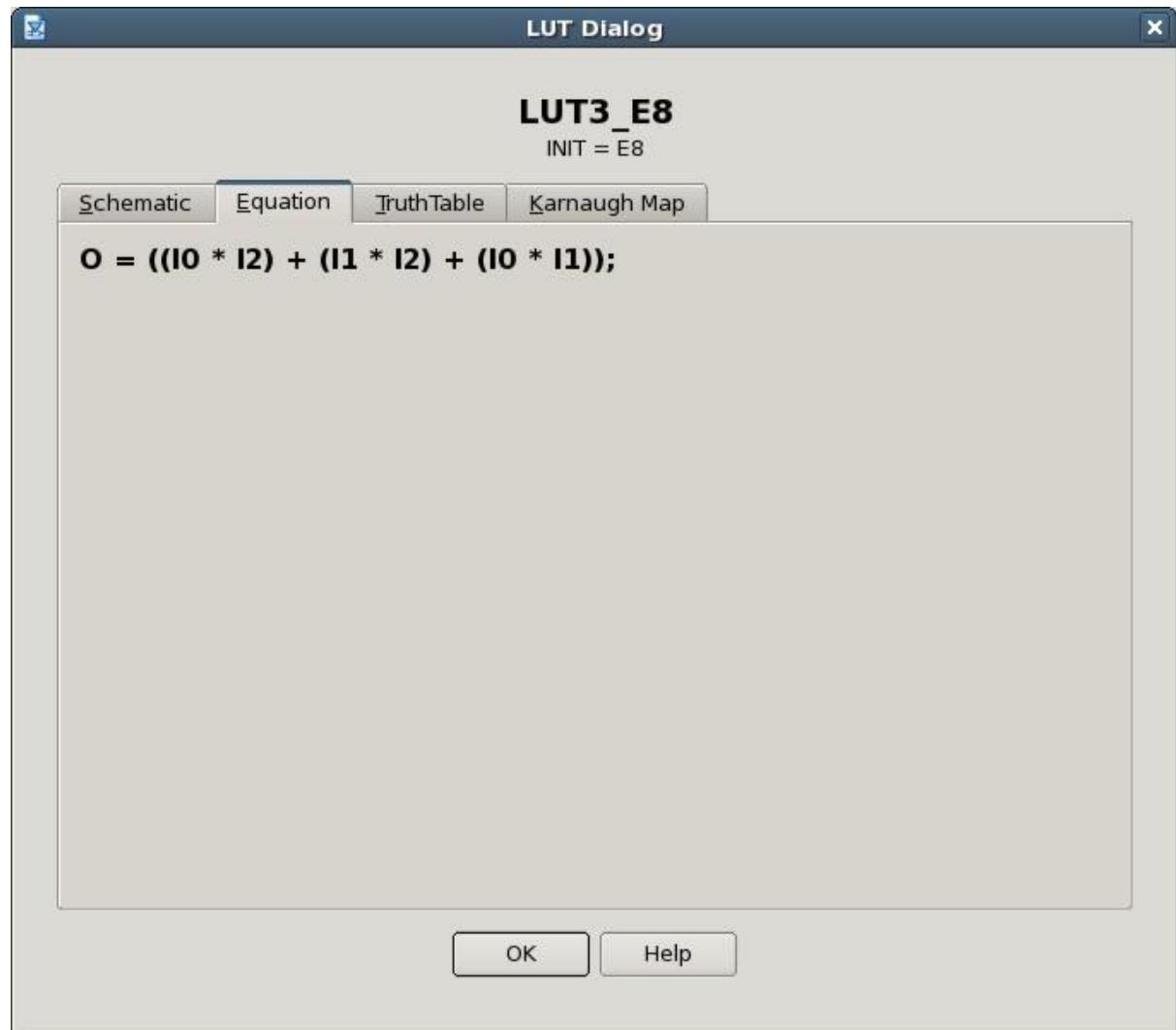
Contents of 1st LUT

The first you'll see is the Logic Diagram of the 1st LUT



Contents of 1st LUT

Click the tabs and you can see the equation of the 1st LUT



Contents of 1st LUT

The Truth Table of the 1st LUT

LUT Dialog

LUT3_E8
INIT = E8

Schematic Equation **TruthTable** Karnaugh Map

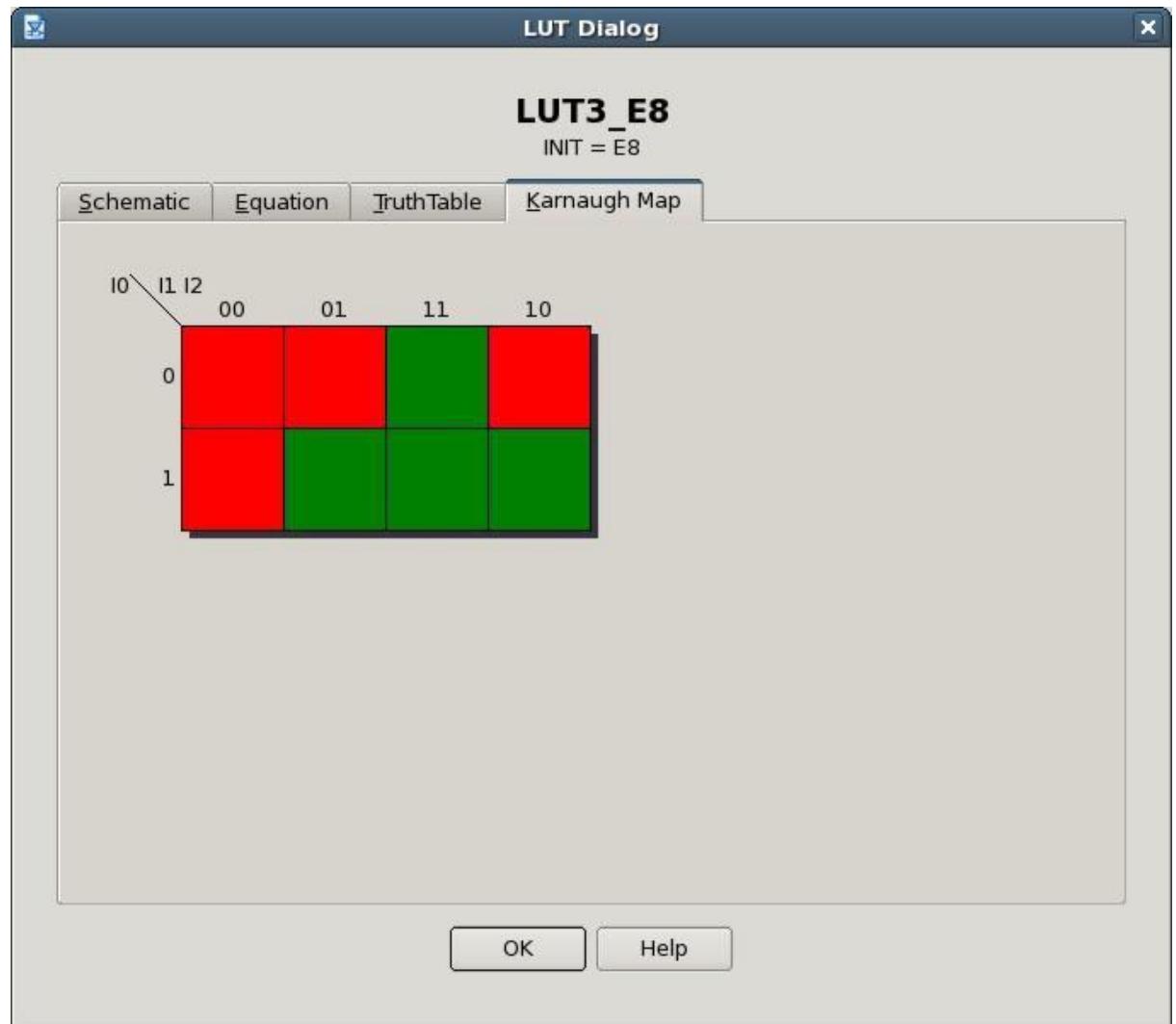
| I2 | I1 | I0 | O |
|----|----|----|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

OK Help



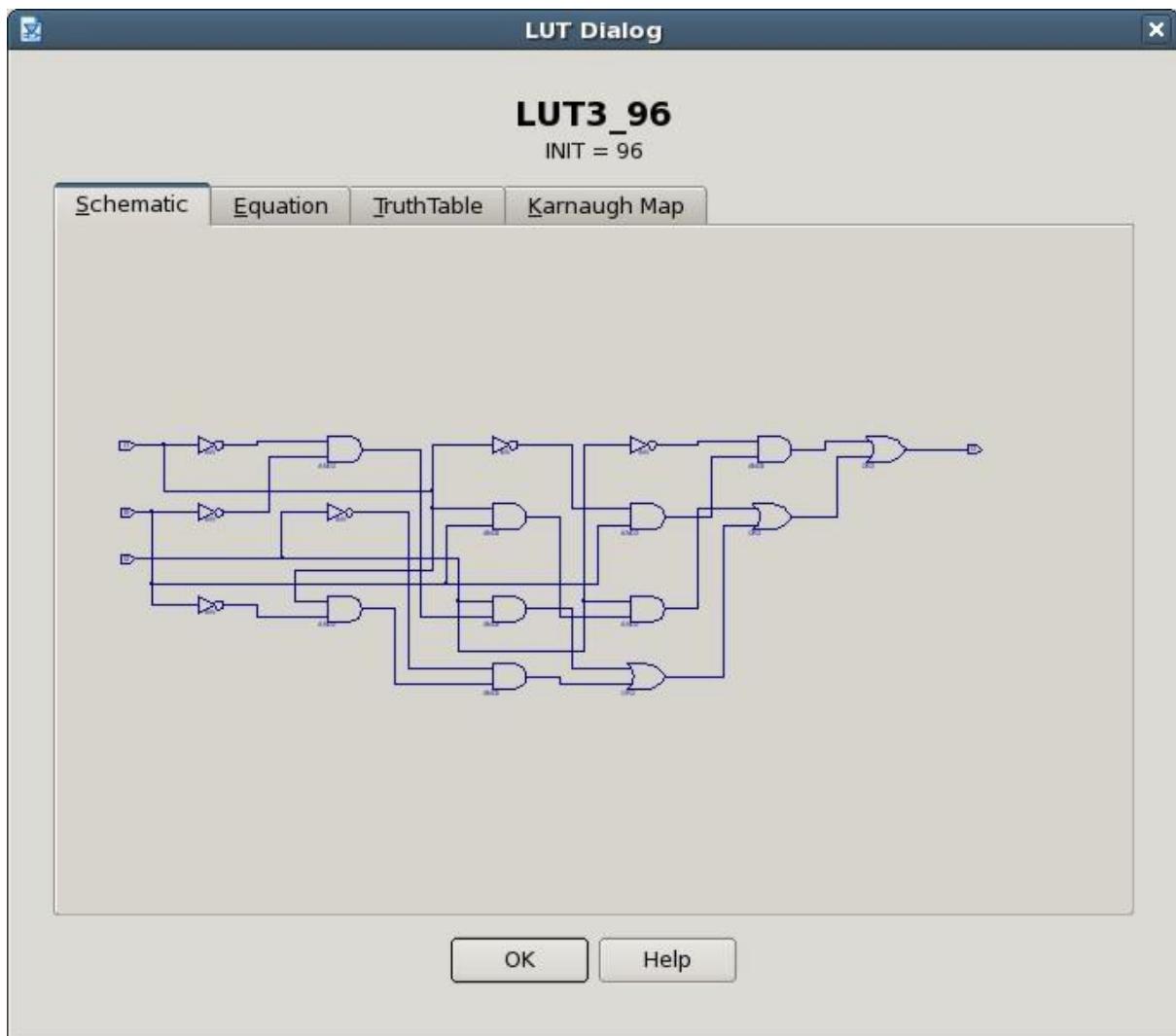
Contents of 1st LUT

The KMap of the 1st LUT
(greens are 1s)



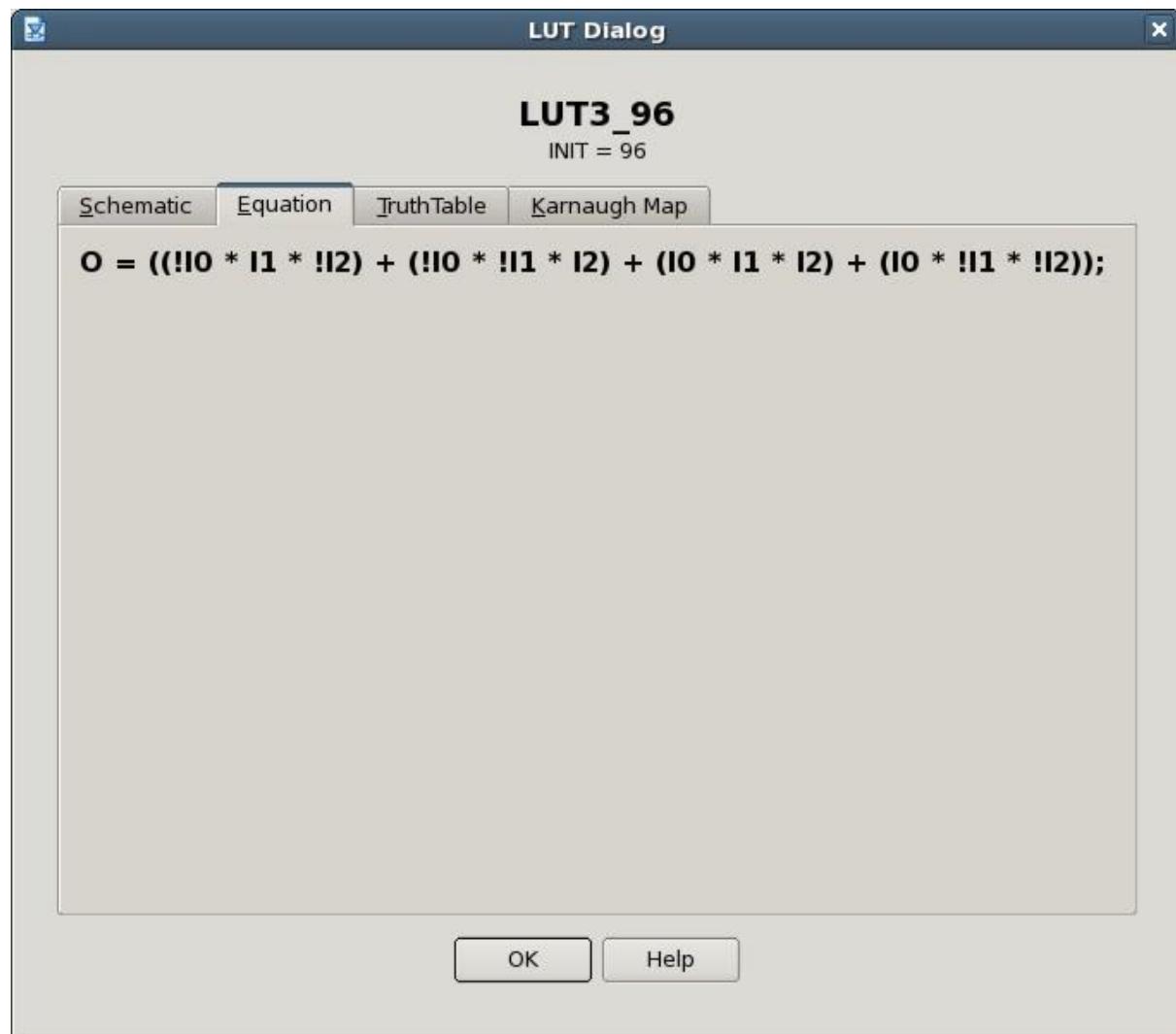
Contents of 2nd LUT

The Logic Diagram of the 2nd LUT



Contents of 2nd LUT

The equation of the 2nd LUT



Contents of 2nd LUT

The Truth Table of the 2nd LUT

LUT Dialog

LUT3_96
INIT = 96

Schematic Equation **TruthTable** Karnaugh Map

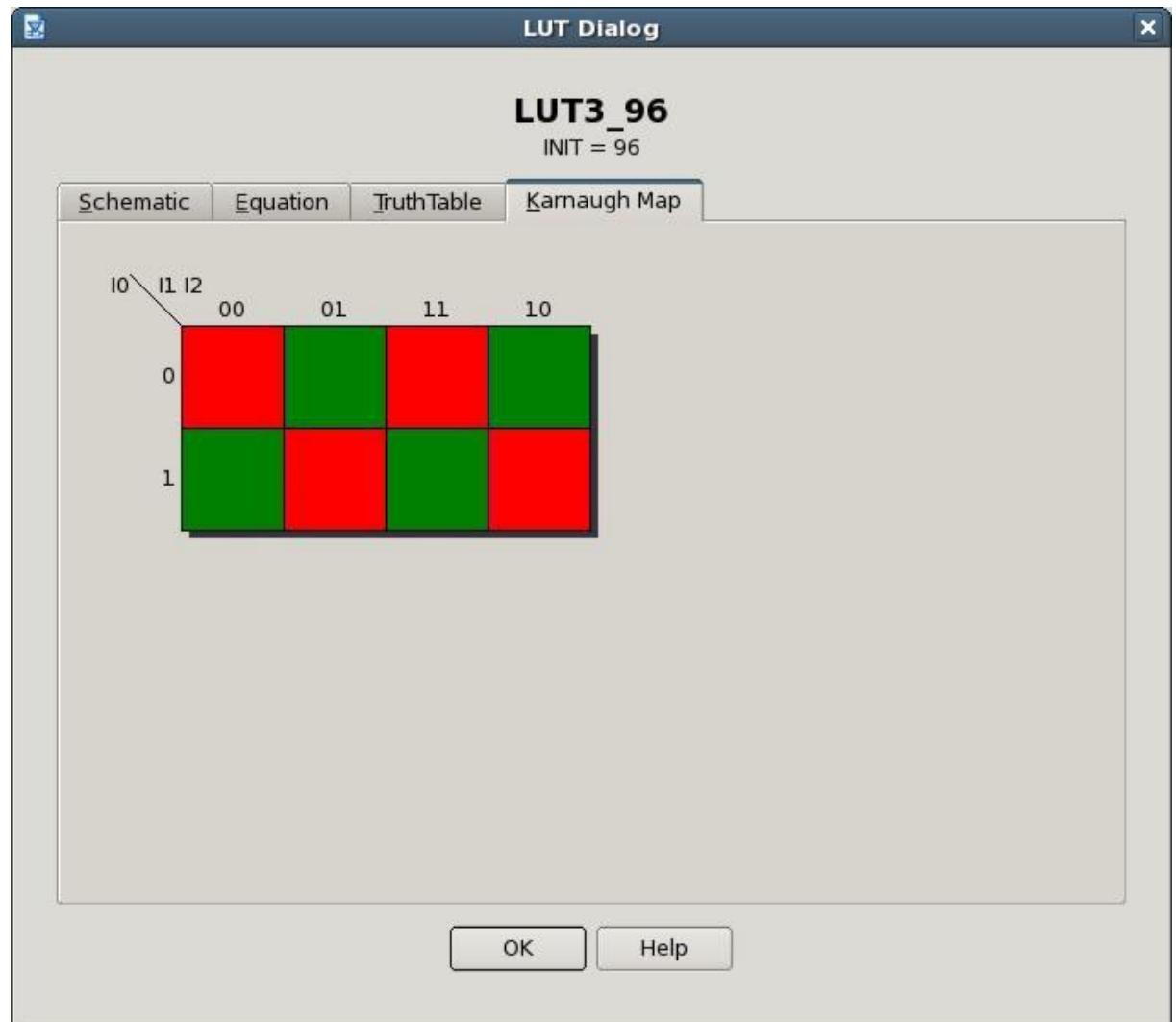
| I2 | I1 | I0 | O |
|----|----|----|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

OK Help



Contents of 2nd LUT

The KMap of the 2nd LUT
(greens are 1s)



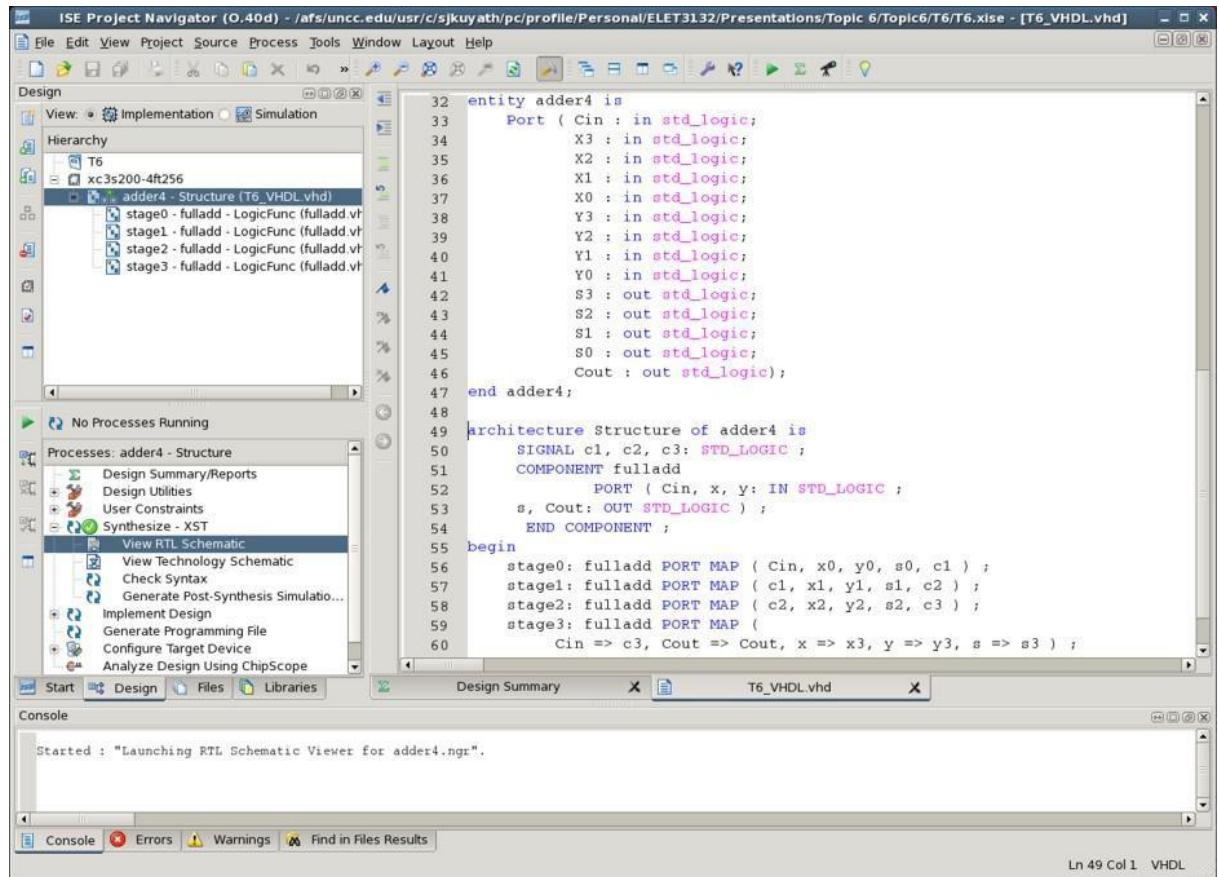
View RTL Schematic

Viewing the RTL Schematic will show you a little different information.

“The RTL schematic shows a representation of the pre-optimized design in terms of generic symbols, such as adders, multipliers, counters, AND gates, and OR gates, that are independent of the targeted Xilinx device. Viewing this schematic may help you discover design issues early in the design process.”¹

It seems to be more directly related to your VHDL file.

Double click on “View RTL schematic”



The screenshot shows the ISE Project Navigator interface with the following details:

- File Menu:** File, Edit, View, Project, Source, Process, Tools, Window, Layout, Help.
- Design View:** Implementation (selected), Simulation.
- Hierarchy:** T6 > xc3s200-4ft256 > adder4 - Structure (T6_VHDL.vhd). This structure contains four instances of the fulladd component: stage0, stage1, stage2, and stage3.
- Processes:** adder4 - Structure > Synthesize - XST > View RTL Schematic (highlighted).
- VHDL Code:** The main pane displays the VHDL code for the adder4 entity. It defines the entity with inputs Cin, X0, X1, X2, X3 and outputs S0, S1, S2, S3, Cout. It also defines the architecture with a fulladd component and four stage assignments.
- Console:** Shows the message "Started : "Launching RTL Schematic Viewer for adder4.ngr".
- Status Bar:** Ln 49 Col 1 VHDL.



View RTL Schematic

This dialog box appears (as it did before). Highlight the lower radio button, if it isn't already highlighted, and click OK

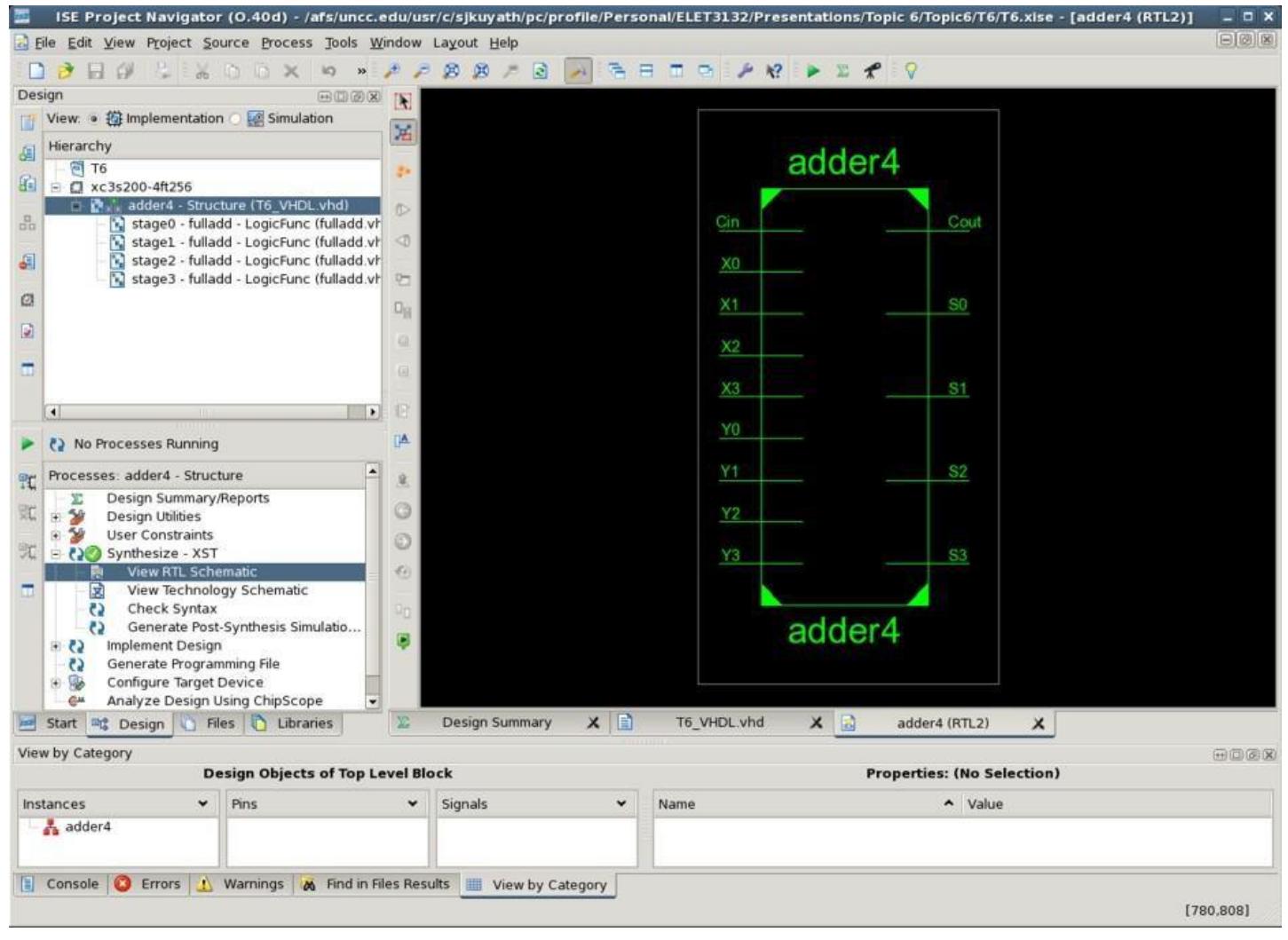


View RTL Schematic

Again, you'll see the complete symbol (the entity) for the "adder4".

The **ENTITY** shows the inputs and the outputs, but no functionality.

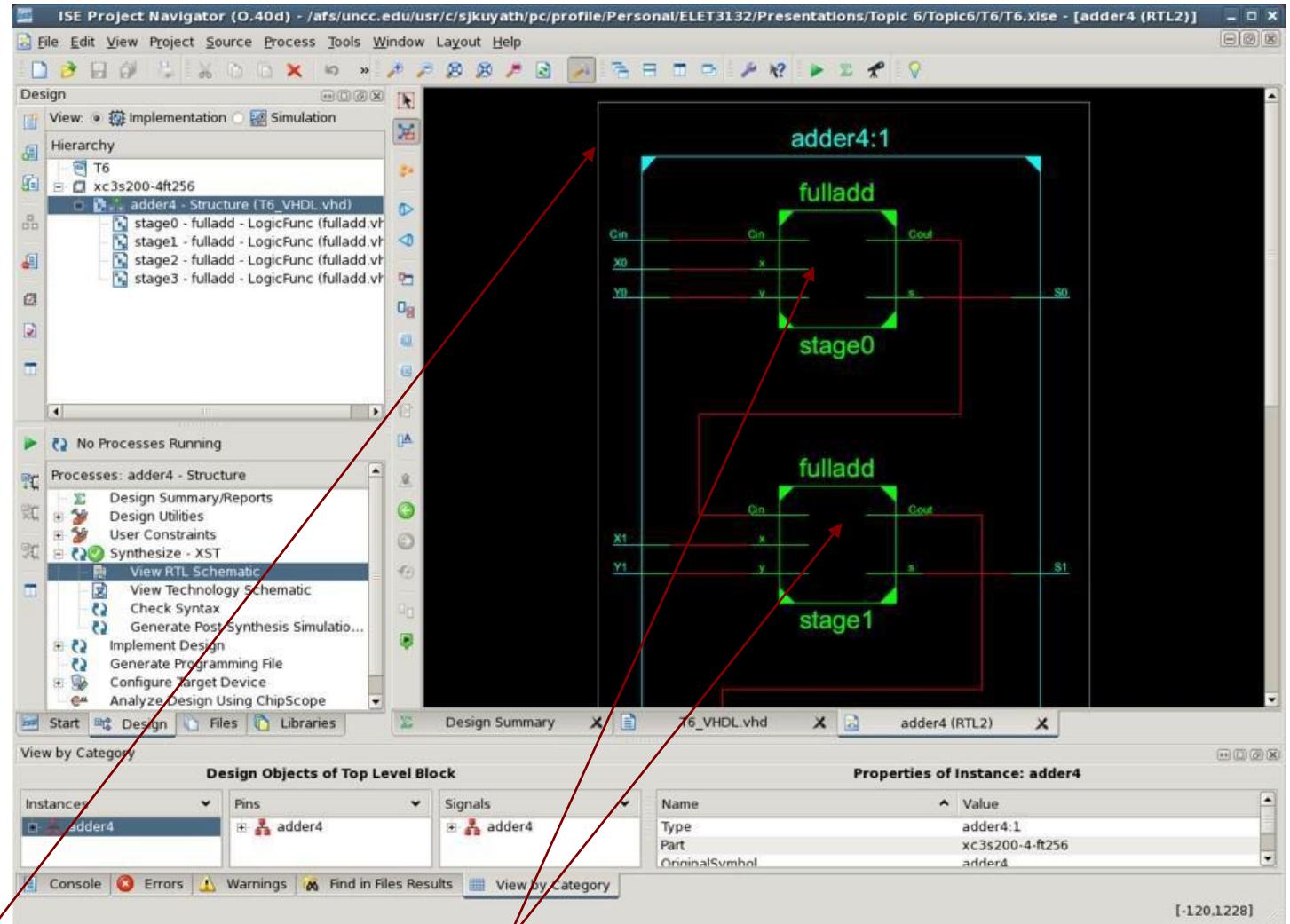
Double click on the entity.



View RTL Schematic

This is the block diagram.

It shows the complete entity for “Adder4”, the external connections to internal components, the internal components, and how they are connected to each other (in red). These internal connections in Xilinx are called “Signals”



This is the whole 4-bit adder (adder4)

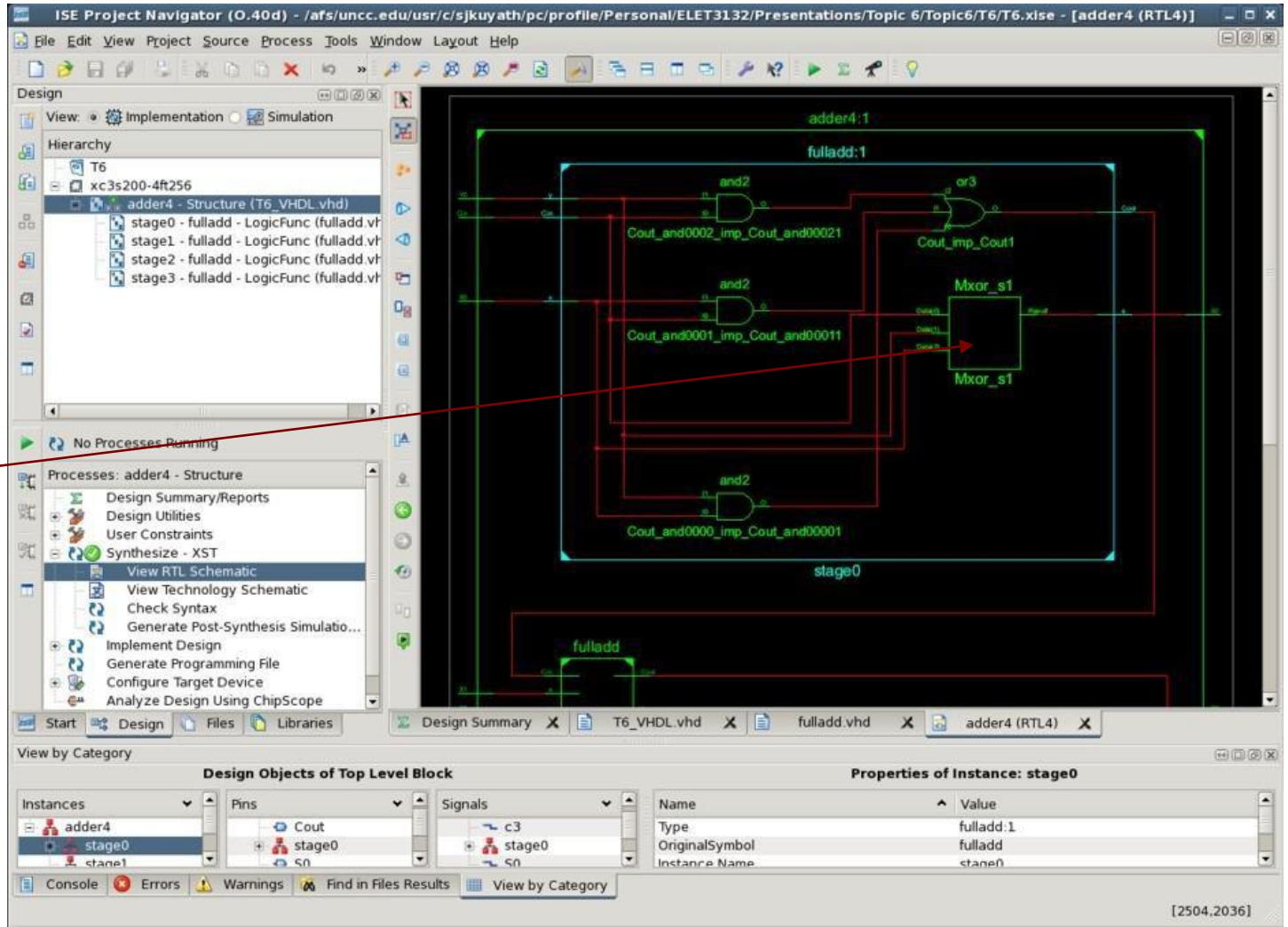
These are the individual full adders (the components)



View RTL Schematic

Double clicking on any full adder brings up the functionality inside the entities:

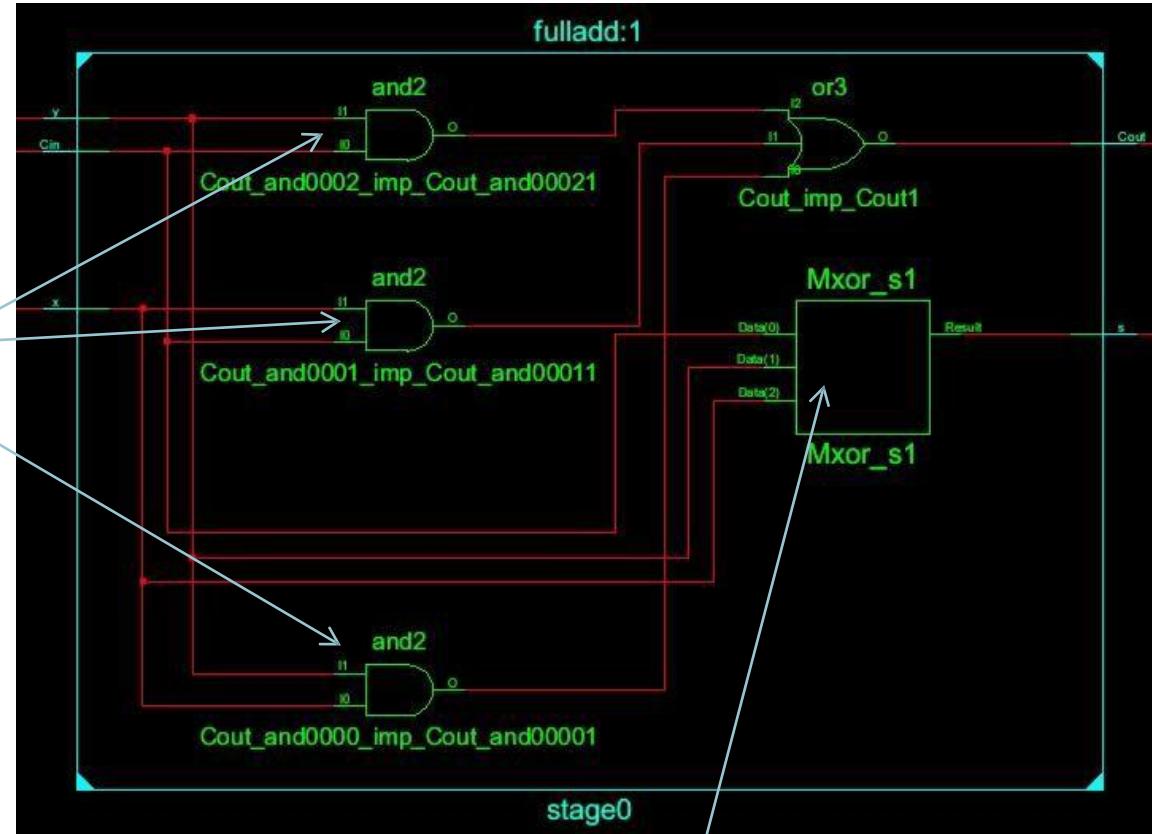
XOR gates are called Mxor



View RTL Schematic

The original equations for the fulladd
are:

$$\begin{aligned} \text{Cout} &\leqslant (x \text{ AND } y) \\ &\text{OR} \\ &(\text{Cin AND } x) \\ &\text{OR} \\ &(\text{Cin AND } y); \end{aligned}$$



$$s \leqslant x \text{ XOR } y \text{ XOR } \text{Cin};$$



Verification

- The contents of the logic diagrams, truth table, or K-Map should match the engineer's design.
 - If not, it is time to troubleshoot the VHDL files
 - If so, it is time to test the design
 - Develop a test bench and execute it
 - Download the file to the Spartan3 board and verify the design



Troubleshooting VHDL Files in ISE

The screenshot shows the ISE Project Navigator interface. On the left, the Hierarchy pane shows a project named 'T6' containing a 'adder4 - Structure (T6_VHDL.vhd)' file. The Processes pane shows 'Synthesize - XST' selected, with 'View RTL Schematic' highlighted. The main window displays the VHDL code for 'adder4'. Two red arrows point to specific lines of code where errors were introduced:

```
35      X2 : in std_logic;
36      X1 : in std_logic;
37      X0 : in std_logic;
38      Y3 : in std_logic;
39      Y2 : in std_logic;
40      Y1 : in std_logic;
41      Y0 : in std_logic;
42      S3 : out std_logic;
43      S2 : out std_logic;
44      S1 : out std_logic;
45      S0 : out std_logic;
46      Cout : out std_logic);
47 end adder4;
48
49 architecture Structure of adder4 is
50   SIGNAL c1, c2, c3: STD_LOGIC
51   COMPONENT fulladd
52     PORT ( Cin, x, y: IN STD_LOGIC ;
53            s, Cout: OUT STD_LOGIC );
54   END COMPONENT ;
55 begin
56   stage0: fulladd PORT MAP ( Cin, x0, y0, s0 );
57   stage1: fulladd PORT MAP ( c1, x1, y1, s1, c2 );
58   stage2: fulladd PORT MAP ( c2, x2, y2, s2, c3 );
59   stage3: fulladd PORT MAP (
60     Cin => c3, Cout => Cout, x => x3, y => y3, s => s3 );
61 end Structure;
```

The errors listed in the Console window are:

```
Line 51: Syntax error near "COMPONENT".
Line 52: Syntax error near "IN".
Line 53: Syntax error near "OUT".
Line 56: Syntax error near ")".
Line 57: Syntax error near ")".
```

Two errors were put in this file
1. The semicolon was removed
2. c1 was removed from this list

Immediately, errors were generated:



Troubleshooting VHDL Files in ISE

```
ISE Project Navigator (0.40d) - /afs/uncc.edu/usr/c/sjkuyath/pc/profile/Personal/ELET3132/Presentations/Topic 6/Topic6/T6/T6.xise - [T6 VHDL.vhd]
File Edit View Project Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
T6
  xc3s200-4ft256
    adder4 - Structure (T6_VHDL.vhd)
      stage0 - fulladd - LogicFunc (fulladd.vhd)
      stage1 - fulladd - LogicFunc (fulladd.vhd)
      stage2 - fulladd - LogicFunc (fulladd.vhd)
      stage3 - fulladd - LogicFunc (fulladd.vhd)
No Processes Running
Processes: adder4 - Structure
  Design Summary/Reports
  Design Utilities
  User Constraints
  Synthesize - XST
    View RTL Schematic
    View Technology Schematic
    Check Syntax
    Generate Post-Synthesis Simulation...
  Implement Design
  Generate Programming File
  Configure Target Device
  Analyze Design Using ChipScope
Start Design Files Libraries Design Summary (out of date) T6_VHDL.vhd fulladd.vhd
Console
file/Personal/ELET3132/Presentations/Topic 6/Topic6/T6 VHDL.vhd" Line 51: Syntax error near "COMPONENT".
Console
file/Personal/ELET3132/Presentations/Topic 6/Topic6/T6 VHDL.vhd" Line 51: Syntax error near "COMPONENT".
file/Personal/ELET3132/Presentations/Topic 6/Topic6/T6 VHDL.vhd" Line 52: Syntax error near "IN".
file/Personal/ELET3132/Presentations/Topic 6/Topic6/T6 VHDL.vhd" Line 53: Syntax error near "OUT".
file/Personal/ELET3132/Presentations/Topic 6/Topic6/T6 VHDL.vhd" Line 56: Syntax error near ")".
file/Personal/ELET3132/Presentations/Topic 6/Topic6/T6 VHDL.vhd" Line 57: Syntax error near ")".
file/Personal/ELET3132/Presentations/Topic 6/Topic6/T6 VHDL.vhd" Line 58: Syntax error near ")".
file/Personal/ELET3132/Presentations/Topic 6/Topic6/T6 VHDL.vhd" Line 59: Syntax error near ")".
file/Personal/ELET3132/Presentations/Topic 6/Topic6/T6 VHDL.vhd" Line 60: Syntax error near ")".
file/Personal/ELET3132/Presentations/Topic 6/Topic6/T6 VHDL.vhd" Line 61: Syntax error near ")".
file/Personal/ELET3132/Presentations/Topic 6/Topic6/T6 VHDL.vhd" Line 62: Syntax error near ")".
file/Personal/ELET3132/Presentations/Topic 6/Topic6/T6 VHDL.vhd" Line 63: Syntax error near ")".

```

States that the error is on line 51, but it was the last character on line 50 that caused the error



Troubleshooting VHDL Files in ISE

The screenshot shows the ISE Project Navigator interface. In the center is a code editor displaying VHDL code for a 4-bit adder. The code defines an architecture named 'Structure' for an entity 'adder4'. It includes four stages of full adders (stage0 to stage3) and various signal declarations. The code editor has line numbers from 35 to 63. A red arrow points from the text "States that the error is on line 56, and that it was near the ")" – this is correct, but it doesn't tell us what the error is" to the closing parenthesis at the end of line 56. The code editor tabs at the bottom show 'T6_VHDL.vhd' and 'fulladd.vhd'. Below the code editor is a 'Console' window showing multiple error messages for 'T6_VHDL.vhd' starting from line 51. Another red arrow points from the text "States that the error is on line 56, and that it was near the ")" – this is correct, but it doesn't tell us what the error is" to the closing parenthesis at the end of line 56 in the console log.

```
35      X2 : in std_logic;
36      X1 : in std_logic;
37      X0 : in std_logic;
38      Y3 : in std_logic;
39      Y2 : in std_logic;
40      Y1 : in std_logic;
41      Y0 : in std_logic;
42      S3 : out std_logic;
43      S2 : out std_logic;
44      S1 : out std_logic;
45      S0 : out std_logic;
46      Cout : out std_logic);
47  end adder4;
48
49  architecture Structure of adder4 is
50    SIGNAL c1, c2, c3: STD_LOGIC;
51    COMPONENT fulladd
52      PORT ( Cin, x, y: IN STD_LOGIC ;
53             s, Cout: OUT STD_LOGIC );
54    END COMPONENT ;
55    begin
56      stage0: fulladd PORT MAP ( Cin, x0, y0, s0 );
57      stage1: fulladd PORT MAP ( c1, x1, y1, s1, c2 );
58      stage2: fulladd PORT MAP ( c2, x2, y2, s2, c3 );
59      stage3: fulladd PORT MAP (
60        Cin => c3, Cout => Cout, x => x3, y => y3, s => s3 );
61    end Structure;
```

file/Personal/ELET3132/Presentations/Topic 6/Topic6/T6_VHDL.vhd" Line 51: Syntax error near "COMPONENT".
Console
file/Personal/ELET3132/Presentations/Topic 6/Topic6/T6_VHDL.vhd" Line 51: Syntax error near "COMPONENT".
file/Personal/ELET3132/Presentations/Topic 6/Topic6/T6_VHDL.vhd" Line 52: Syntax error near "IN".
file/Personal/ELET3132/Presentations/Topic 6/Topic6/T6_VHDL.vhd" Line 53: Syntax error near "OUT".
file/Personal/ELET3132/Presentations/Topic 6/Topic6/T6_VHDL.vhd" Line 56: Syntax error near ")".
file/Personal/ELET3132/Presentations/Topic 6/Topic6/T6_VHDL.vhd" Line 57: Syntax error near ")".

Console Errors Warnings Find in Files Results

States that the error is on line 56, and that it was near the ")" – this is correct, but it doesn't tell us what the error is



Troubleshooting VHDL Files in ISE

The screenshot shows the ISE Project Navigator interface. In the center is a code editor displaying VHDL code for a 4-bit adder. The code defines an architecture named 'adder4' with four stages of full adders. The 'Check Syntax' option is selected in the 'Processes' menu. A callout points from the 'Check Syntax' menu entry to the error message in the 'Console' window below.

```
35      X2 : in std_logic;
36      X1 : in std_logic;
37      X0 : in std_logic;
38      Y3 : in std_logic;
39      Y2 : in std_logic;
40      Y1 : in std_logic;
41      Y0 : in std_logic;
42      S3 : out std_logic;
43      S2 : out std_logic;
44      S1 : out std_logic;
45      S0 : out std_logic;
46      Cout : out std_logic);
47 end adder4;
48
49 architecture Structure of adder4 is
50     SIGNAL c1, c2, c3: STD_LOGIC
51     COMPONENT fulladd
52         PORT ( Cin, x, y: IN STD_LOGIC ;
53                 s, Cout: OUT STD_LOGIC ) ;
54     END COMPONENT ;
55 begin
56     stage0: fulladd PORT MAP ( Cin, x0, y0, s0) ;
57     stage1: fulladd PORT MAP ( c1, x1, y1, s1, c2 ) ;
58     stage2: fulladd PORT MAP ( c2, x2, y2, s2, c3 ) ;
59     stage3: fulladd PORT MAP (
60             Cin => c3, Cout => Cout, x => x3, y => y3, s => s3 ) ;
61 end Structure;
```

Console

```
Topic6/T6_VHDL.vhd" Line 51. parse error, unexpected COMPONENT, expecting AFFECT or SEMICOLON
```

When “Check Syntax” was invoked, the IDE responded with an error in the Console Window that indicated there was an error on line 51 – line 50 is the line that’s missing the semicolon, but at least this message is giving a better idea



Troubleshooting VHDL Files in ISE

```
35      X2 : in std_logic;
36      X1 : in std_logic;
37      X0 : in std_logic;
38      Y3 : in std_logic;
39      Y2 : in std_logic;
40      Y1 : in std_logic;
41      Y0 : in std_logic;
42      S3 : out std_logic;
43      S2 : out std_logic;
44      S1 : out std_logic;
45      S0 : out std_logic;
46      Cout : out std_logic);
47 end adder4;
48
49 architecture Structure of adder4 is
50   SIGNAL c1, c2, c3: STD_LOGIC ;
51   COMPONENT fulladd
52     PORT ( Cin, x, y: IN STD_LOGIC ;
53            s, Cout: OUT STD_LOGIC ) ;
54   END COMPONENT ;
55 begin
56   stage0: fulladd PORT MAP ( Cin, x0, y0, s0 ) ;
57   stage1: fulladd PORT MAP ( c1, x1, y1, s1, c2 ) ;
58   stage2: fulladd PORT MAP ( c2, x2, y2, s2, c3 ) ;
59   stage3: fulladd PORT MAP (
60     Cin => c3, Cout => Cout, x => x3, y => y3, s => s3 ) ;
61 end Structure;
```

Warnings

```
3 - "/afs/uncc.edu/usr/c/sjkuyath/pc/profile/Personal/ELET3132/Presentations/Topic_6/Topic6/T6_VHDL.vhd" line 56: Unconnected output port 'Cout' declared.
3 - Signal <c1> is used but never assigned. This sourceless signal will be automatically connected to value 0.
```

The semicolon was replaced to correct this error.

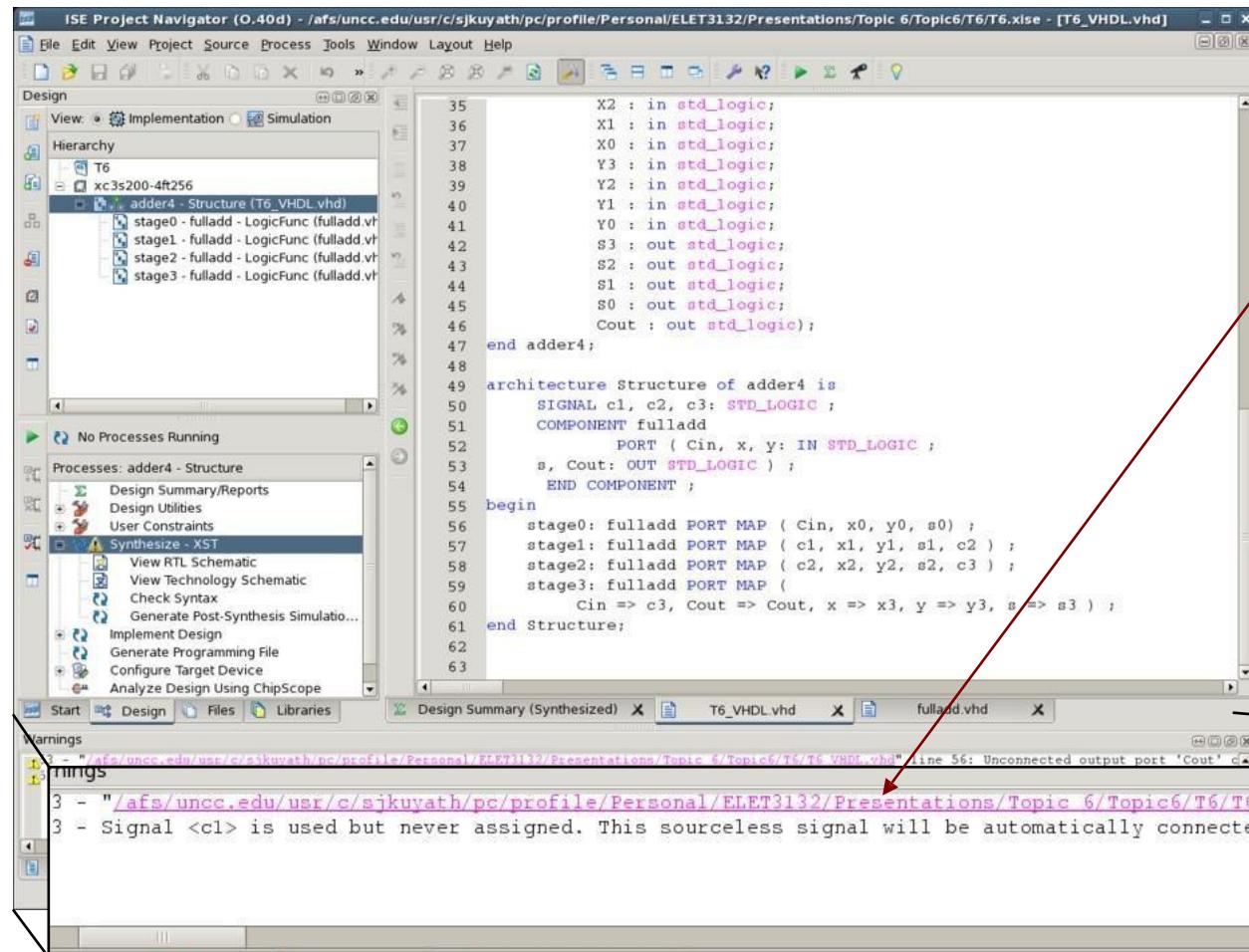
Then

"Synthesize-XST" was invoked again (after the semicolon was replaced). The IDE responded with a warning in the Console Window that indicated c1 was used but never assigned.....

This may not be real helpful



Troubleshooting VHDL Files in ISE



The screenshot shows the ISE Project Navigator interface. The left pane displays the project hierarchy, showing a design named 'T6' containing an 'adder4' component. The right pane shows the VHDL source code for 'adder4'. The code defines an adder4 component with four stages of full adders. A warning message is visible in the bottom right corner of the code editor window.

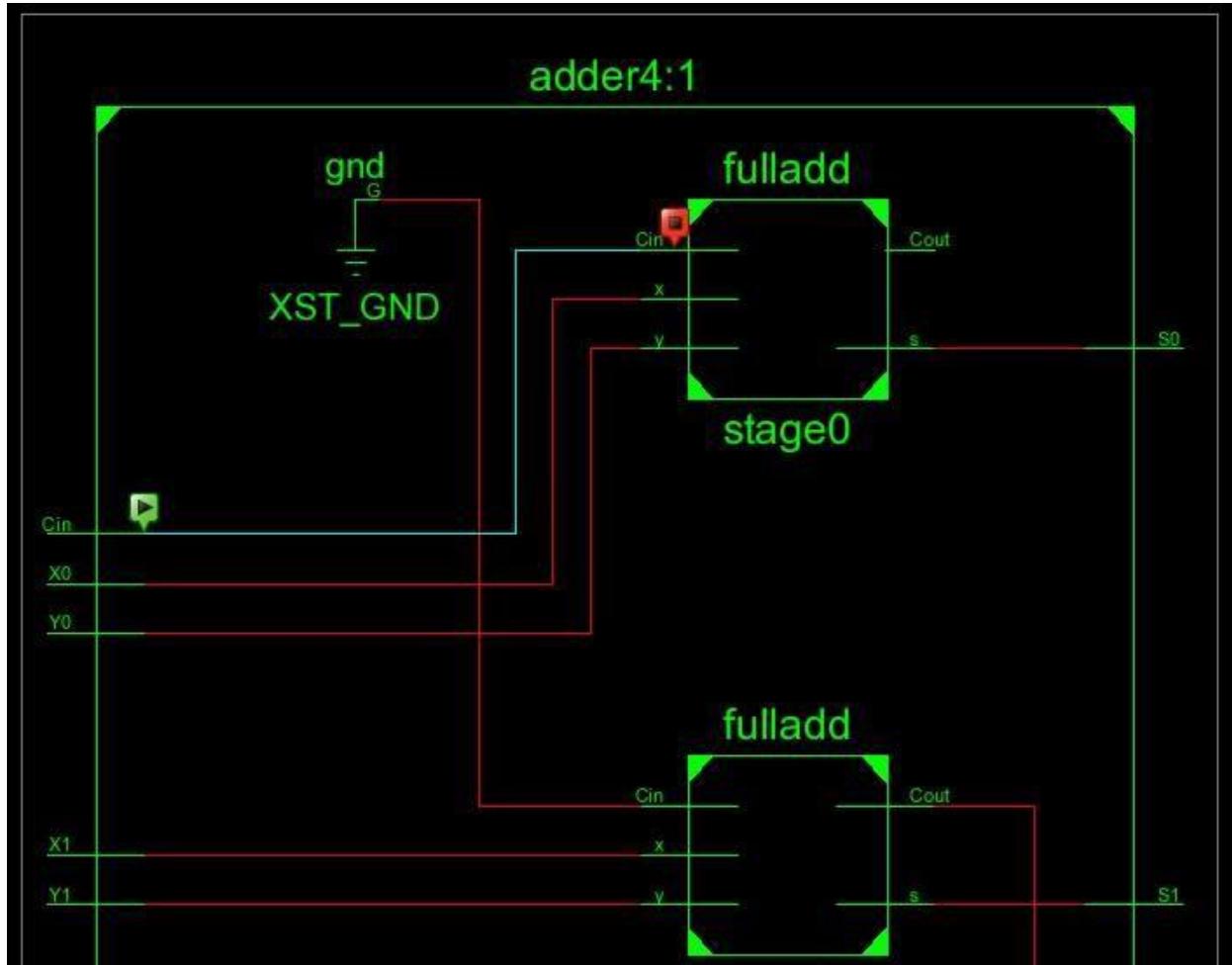
```
35      X2 : in std_logic;
36      X1 : in std_logic;
37      X0 : in std_logic;
38      Y3 : in std_logic;
39      Y2 : in std_logic;
40      Y1 : in std_logic;
41      Y0 : in std_logic;
42      S3 : out std_logic;
43      S2 : out std_logic;
44      S1 : out std_logic;
45      S0 : out std_logic;
46      Cout : out std_logic);
47 end adder4;
48
49 architecture Structure of adder4 is
50   SIGNAL c1, c2, c3: STD_LOGIC ;
51   COMPONENT fulladd
52     PORT ( Cin, x, y: IN STD_LOGIC ;
53            s, Cout: OUT STD_LOGIC ) ;
54   END COMPONENT ;
55 begin
56   stage0: fulladd PORT MAP ( Cin, x0, y0, s0 ) ;
57   stage1: fulladd PORT MAP ( c1, x1, y1, s1, c2 ) ;
58   stage2: fulladd PORT MAP ( c2, x2, y2, s2, c3 ) ;
59   stage3: fulladd PORT MAP (
60     Cin => c3, Cout => Cout, x => x3, y => y3, s => s3 ) ;
61 end Structure;
```

The other warning indicates that Cout was an unconnected port.....

hmmmm.... Again, not real helpful



Troubleshooting VHDL Files in ISE



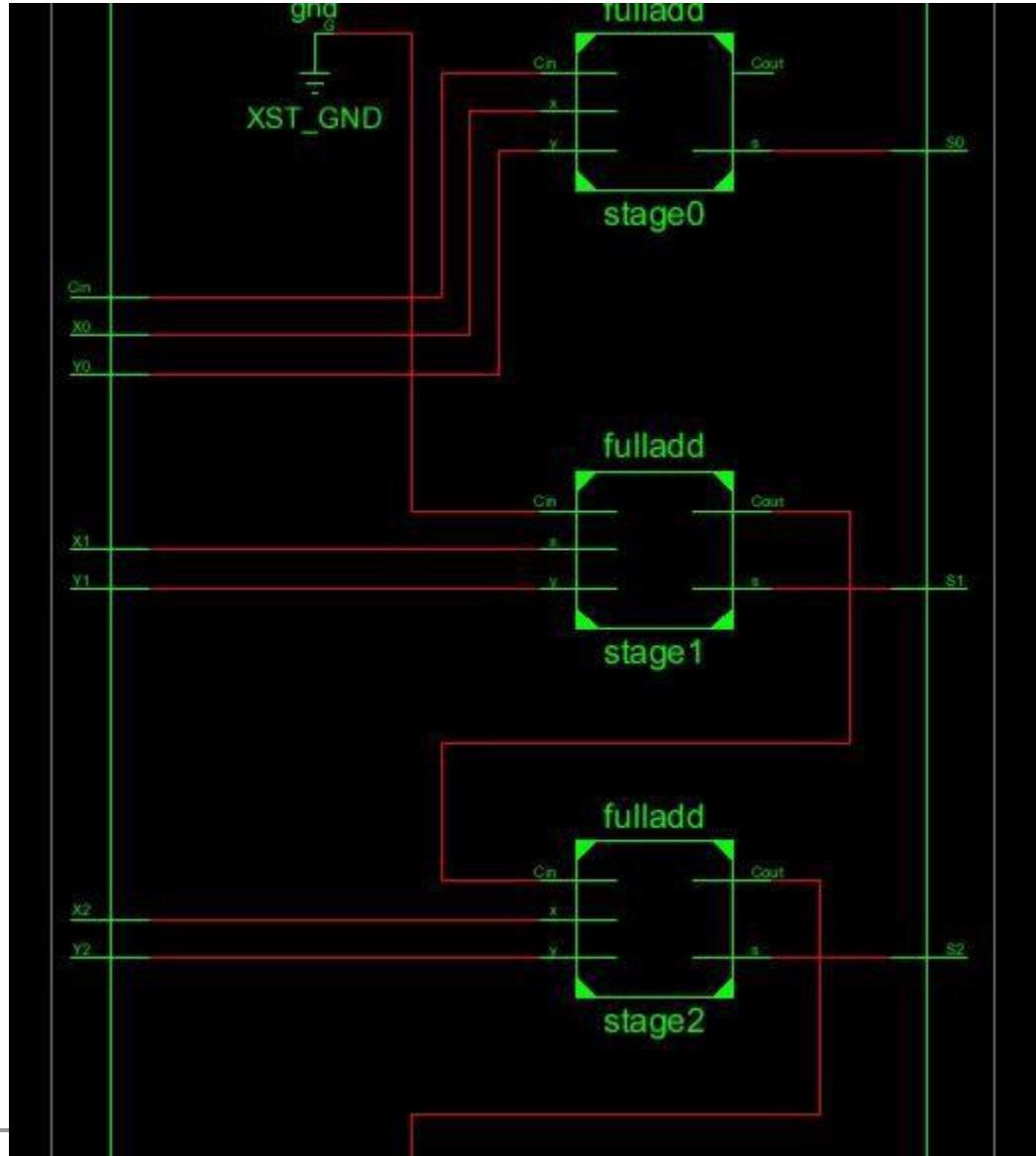
Viewing the RTL might help – clearly this is not correct.

It shows that Cout from stage 0 is not connected and that Cin from stage 1 is grounded.

If you looked further down the RTL schematic you could see the way the others are connected (next page)



Troubleshooting VHDL Files in ISE



Viewing more of the RTL shows the way the other Cout(s) and Cin(s) are connected. This may help fix the problem.



Troubleshooting VHDL Files in ISE

So, Xilinx can point *near* to the problems but it cannot design a circuit for you, nor troubleshoot for you



Testbench Generator/Simulation

In version 7.1 of Xilinx, you could edit waveforms to create a test bench as shown below

To edit the waveforms, just click on the blue box.

In this test bench, the ISE will add the following numbers:

X=1, Y=1, Cin=0

X=2, Y=2, Cin=0

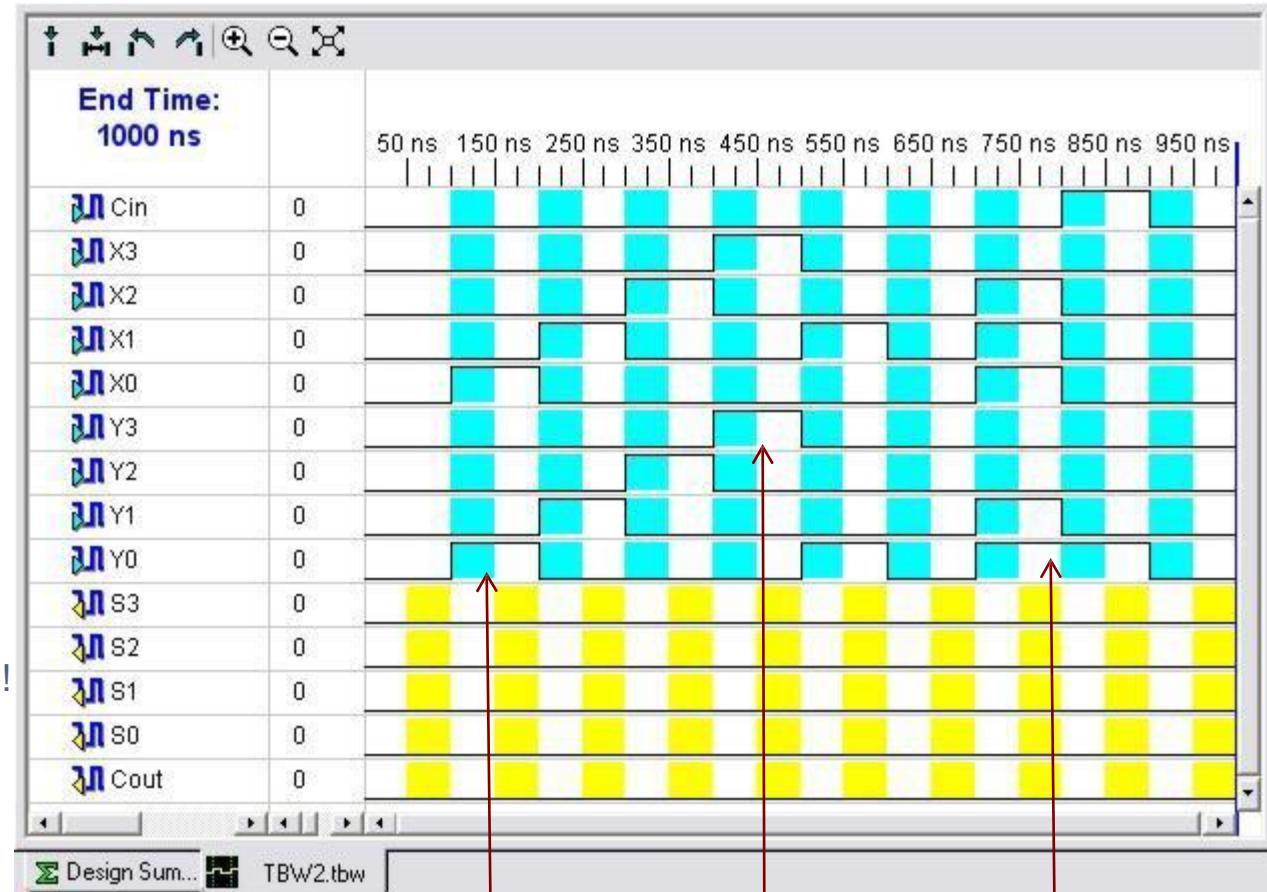
X=4, Y=4, Cin=0

X=8, Y=8, Cin=0

X=2, Y=1, Cin=0

X=7, Y=3, Cin=0

X=0, Y=1, Cin=1



X=1, Y=1, Cin=0

X=7, Y=3, Cin=0

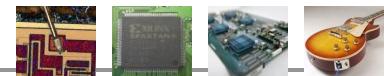
X=8, Y=8, Cin=0

Testbench Generator/Simulation

No longer.

Version 11 was the last to use a graphical test bench generator

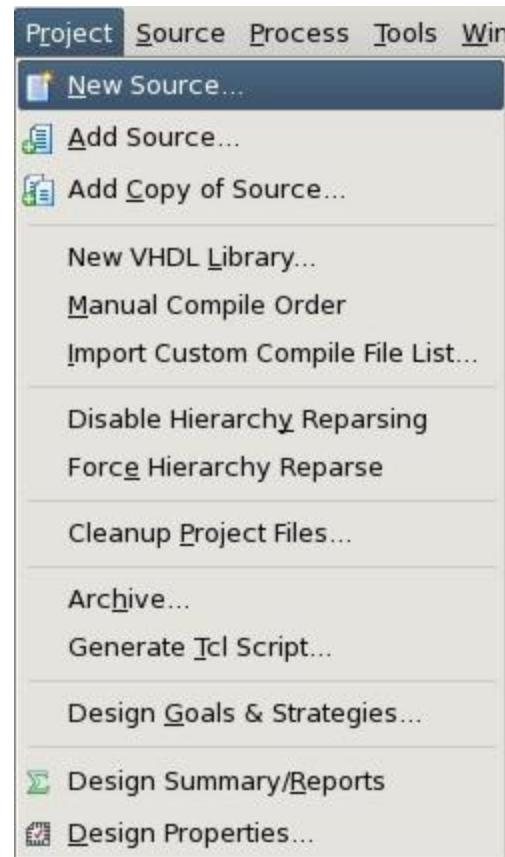
Now we have to create a VHDL file



Testbench Generator/Simulation

You create a test bench program as a VHDL file to simulate the logic of your design.

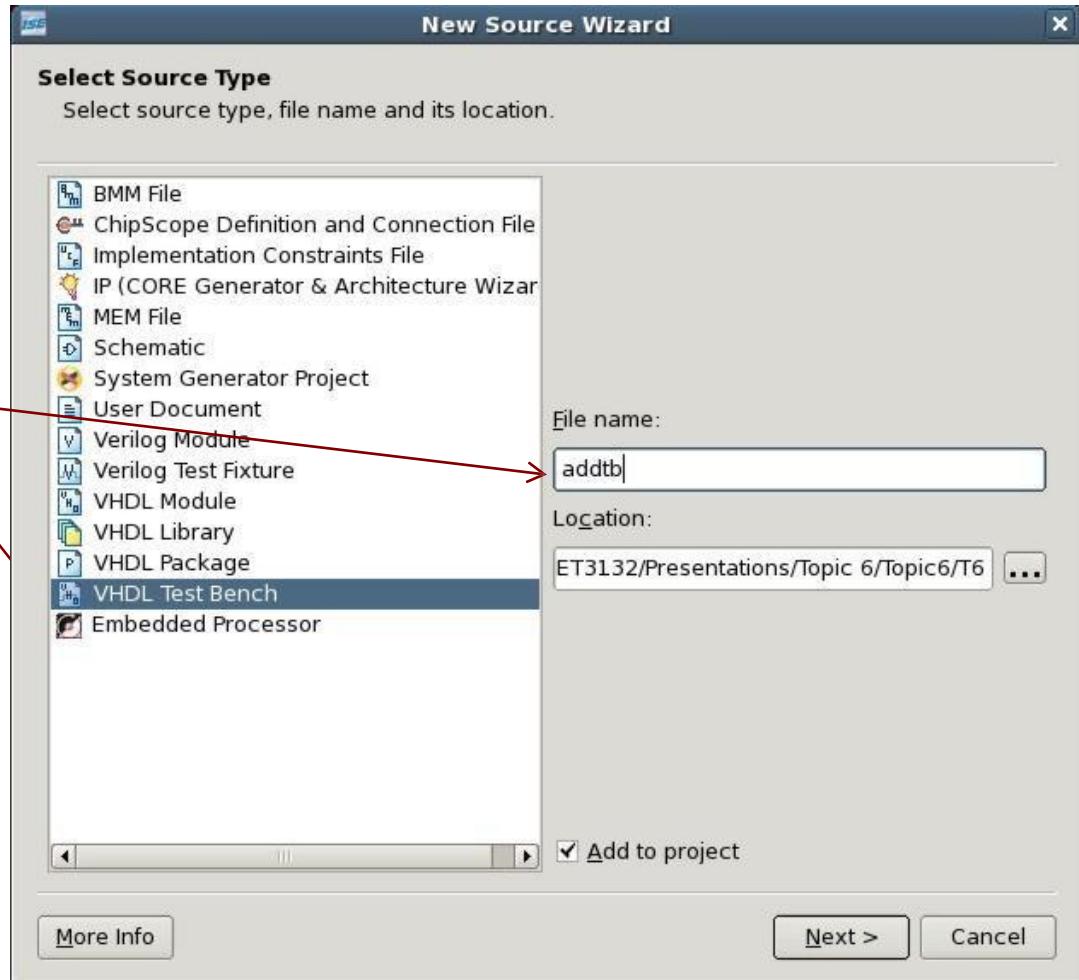
To create a testbench, click on Project and then New Source.



Testbench Generator/Simulation

The New Source Window will appear.

Choose VHDL Test Bench and provide a filename for this source file and then click on the [Next] button.

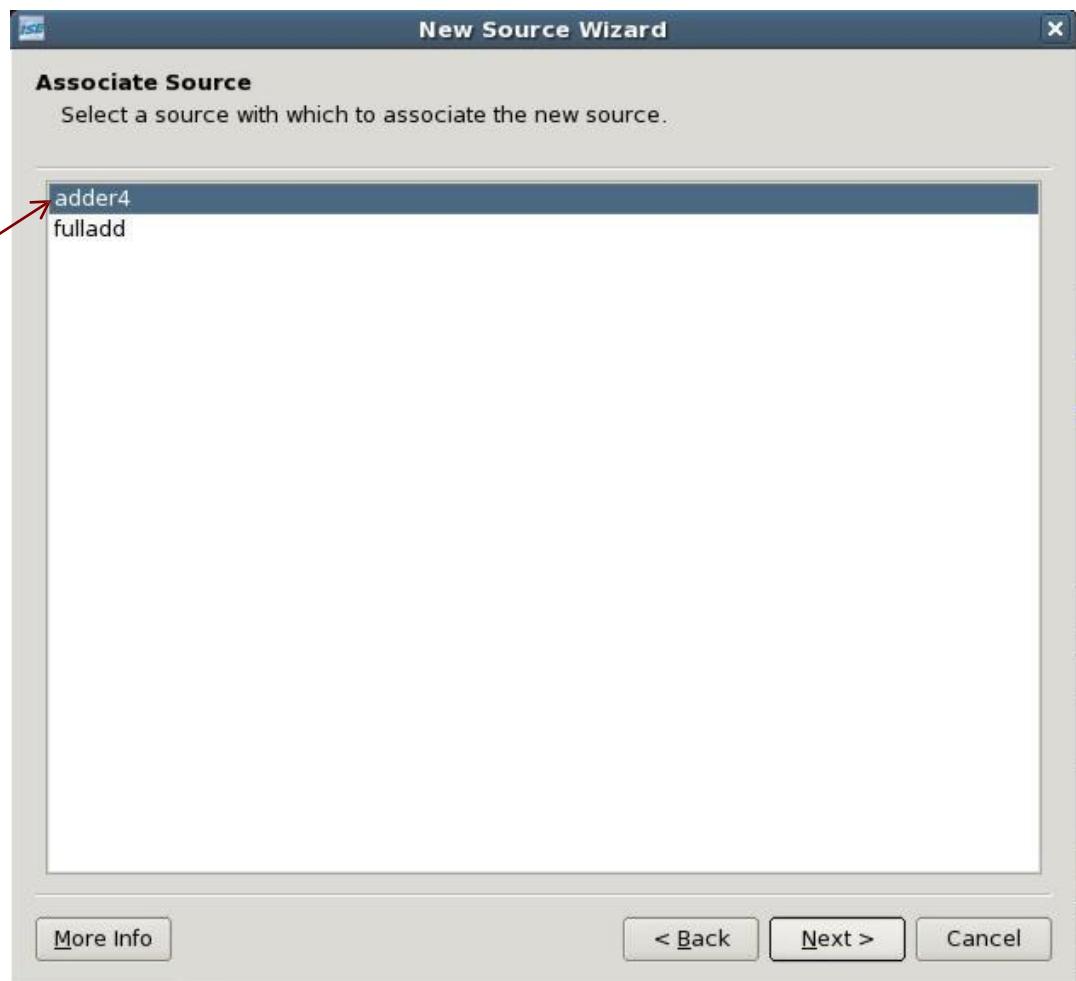


Testbench Generator/Simulation

The ISE will ask which VHDL file you would like to simulate.

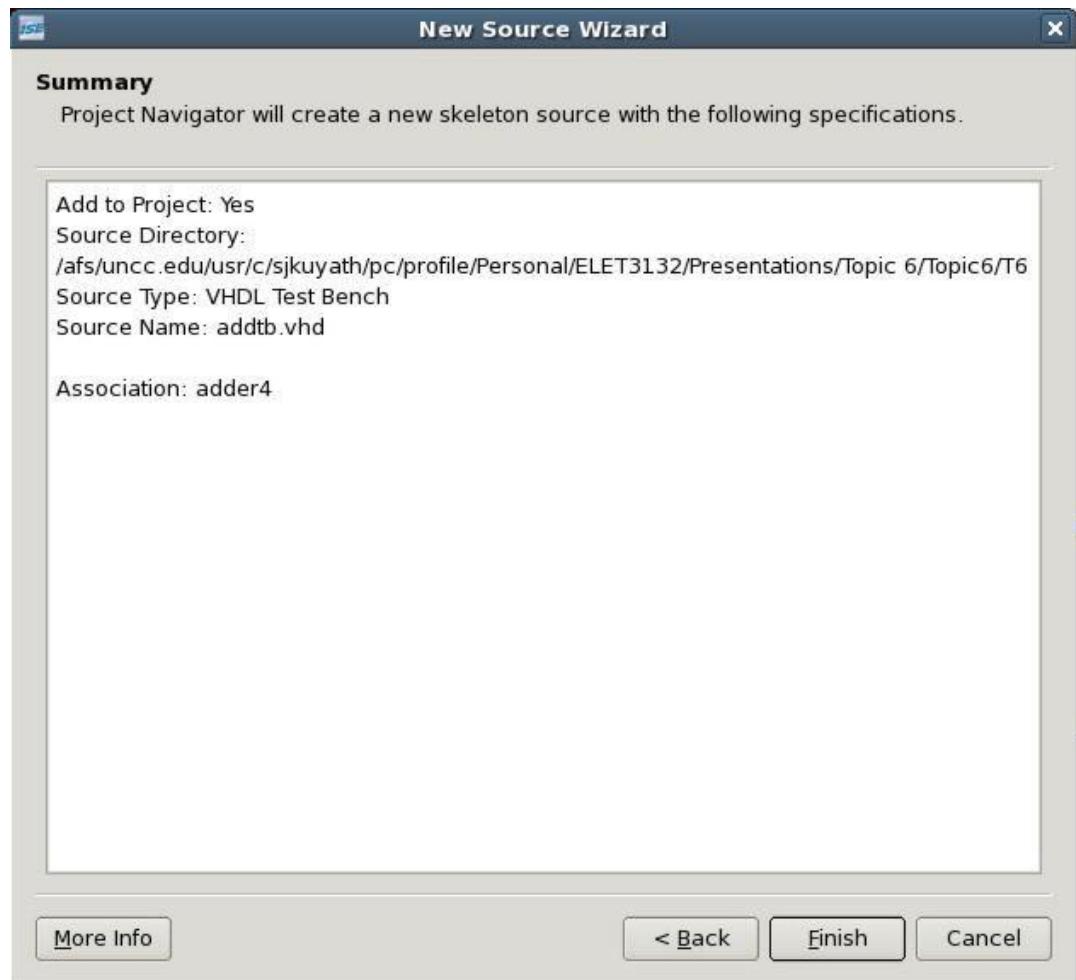
Because I want to simulate the whole design, I will choose the highest level design: “adder4”

Then click on [Next]



Testbench Generator/Simulation

The next screen is an information screen, click on [Finish]



Testbench Generator/Simulation

A basic VHDL file “template” will appear

```
1 -- Company:  
2 -- Engineer:  
3 --  
4 -- Create Date: 13:24:13 09/19/2011  
5 -- Design Name:  
6 -- Module Name: /afs/uncc.edu/usr/c/sjkuyath/pc/profile/Personal/ELET  
7 -- Project Name: T6  
8 -- Target Device:  
9 -- Tool versions:  
10 -- Description:  
11 --  
12 --  
13 -- VHDL Test Bench Created by ISE for module: adder4  
14 --  
15 -- Dependencies:  
16 --  
17 -- Revision:  
18 -- Revision 0.01 - File Created  
19 -- Additional Comments:  
20 --  
21 -- Notes:  
22 -- This testbench has been automatically generated using types std_logi  
23 -- std_logic_vector for the ports of the unit under test. Xilinx recom  
24 -- that these types always be used for the top-level I/O of a design in  
25 -- to guarantee that the testbench will bind correctly to the post-impl  
26 -- simulation model.  
27 -----  
28 LIBRARY ieee;  
29 USE ieee.std_logic_1164.ALL;
```



Testbench Generator/Simulation

A basic VHDL file “template” will appear

```
1 -- Company:  
2 -- Engineer:  
3 --  
4 --  
5 -- Create Date: 13:24:13 09/  
6 -- Design Name:  
7 -- Module Name: /afs/uncc.edu  
8 -- Project Name: T6  
9 -- Target Device:  
10 -- Tool versions:  
11 -- Description:  
12 --  
13 -- VHDL Test Bench Created by  
14 --  
15 -- Dependencies:  
16 --  
17 -- Revision:  
18 -- Revision 0.01 - File Created  
19 -- Additional Comments:  
20 --  
21 -- Notes:  
22 -- This testbench has been aut  
23 -- std_logic_vector for the po  
24 -- that these types always be  
25 -- to guarantee that the testb  
26 -- simulation model.  
27 -----  
28 LIBRARY ieee;  
29 USE ieee.std_logic_1164.ALL;  
  
29 USE ieee.std_logic_1164.ALL;  
30  
31 -- Uncomment the following library declaration if using  
32 -- arithmetic functions with Signed or Unsigned values  
33 --USE ieee.numeric_std.ALL;  
34  
35 ENTITY addtb IS  
36 END addtb;  
37  
38 ARCHITECTURE behavior OF addtb IS  
39  
40     -- Component Declaration for the Unit Under Test (UUT)  
41  
42     COMPONENT adder4  
43     PORT(  
44         Cin : IN  std_logic;  
45         X3 : IN  std_logic;  
46         X2 : IN  std_logic;  
47         X1 : IN  std_logic;  
48         X0 : IN  std_logic;  
49         Y3 : IN  std_logic;  
50         Y2 : IN  std_logic;  
51         Y1 : IN  std_logic;  
52         Y0 : IN  std_logic;  
53         S3 : OUT  std_logic;  
54         S2 : OUT  std_logic;  
55         S1 : OUT  std_logic;  
56         S0 : OUT  std_logic;  
57         Cout : OUT  std_logic
```

Scrolling down



Testbench Generator/Simulation

and down....

```
57      Cout : OUT  std_logic
58  );
59 END COMPONENT;
60
61
62 --Inputs
63 signal Cin : std_logic := '0';
64 signal X3 : std_logic := '0';
65 signal X2 : std_logic := '0';
66 signal X1 : std_logic := '0';
67 signal X0 : std_logic := '0';
68 signal Y3 : std_logic := '0';
69 signal Y2 : std_logic := '0';
70 signal Y1 : std_logic := '0';
71 signal Y0 : std_logic := '0';
72
73 --Outputs
74 signal S3 : std_logic;
75 signal S2 : std_logic;
76 signal S1 : std_logic;
77 signal S0 : std_logic;
78 signal Cout : std_logic;
79 -- No clocks detected in port list. Replace <clock> below with
80 -- appropriate port name
81
82 constant <clock>_period : time := 10 ns;
83
84 BEGIN
85
```



Testbench Generator/Simulation

and down....

```
57      Cout : OUT  std_logic
58  );
59 END COMPONENT;
60
61
62 --Inputs
63 signal Cin : std_logic := '0';
64 signal X3 : std_logic := '0';
65 signal X2 : std_logic := '0';
66 signal X1 : std_logic := '0';
67 signal X0 : std_logic := '0';
68 signal Y3 : std_logic := '0';
69 signal Y2 : std_logic := '0';
70 signal Y1 : std_logic := '0';
71 signal Y0 : std_logic := '0';
72
73 --Outputs
74 signal S3 : std_logic;
75 signal S2 : std_logic;
76 signal S1 : std_logic;
77 signal S0 : std_logic;
78 signal Cout : std_logic;
79 -- No clocks detected in port list. Replace <clock> below with
80 -- appropriate port name
81
82 constant <clock>_period : time := 10 ns;
83
84 BEGIN
85
```

and hopefully we catch this error.

I didn't and found many errors generated and had to come back and turn line 82 into a comment (because I wasn't using a clock in my design)



Testbench Generator/Simulation

and then scrolling down....

```
57      Cout : OUT std_logic
58  );
59 END COMPONENT;
60
61
62 --Inputs
63 signal Cin : std_logic := '0';
64 signal X3 : std_logic := '0';
65 signal X2 : std_logic := '0';
66 signal X1 : std_logic := '0';
67 signal X0 : std_logic := '0';
68 signal Y3 : std_logic := '0';
69 signal Y2 : std_logic := '0';
70 signal Y1 : std_logic := '0';
71 signal Y0 : std_logic := '0';
72
73 --Outputs
74 signal S3 : std_logic;
75 signal S2 : std_logic;
76 signal S1 : std_logic;
77 signal S0 : std_logic;
78 signal Cout : std_logic;
79 -- No clocks detected in port list. Re
80 -- appropriate port name
81
82 --constant <clock>_period : time := 10
83
84 BEGIN
85
86 BEGIN
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
-- Instantiate the Unit Under Test (UUT)
uut: adder4 PORT MAP (
    Cin => Cin,
    X3 => X3,
    X2 => X2,
    X1 => X1,
    X0 => X0,
    Y3 => Y3,
    Y2 => Y2,
    Y1 => Y1,
    Y0 => Y0,
    S3 => S3,
    S2 => S2,
    S1 => S1,
    S0 => S0,
    Cout => Cout
);
-- Clock process definitions
--<clock>_process :process
--begin
--    <clock> <= '0';
--    wait for <clock>_period/2;
--    <clock> <= '1';
--    wait for <clock>_period/2;
--end process;
```

and either commenting or deleting
these lines



Testbench Generator/Simulation

and then scrolling down....

```
57      Cout : OUT std_logic
58  );
59 END COMPONENT;
60
61
62 --Inputs
63 signal Cin : std_logic := '0';
64 signal X3 : std_logic := '0';
65 signal X2 : std_logic := '0';
66 signal X1 : std_logic := '0';
67 signal X0 : std_logic := '0';
68 signal Y3 : std_logic := '0';
69 signal Y2 : std_logic := '0';
70 signal Y1 : std_logic := '0';
71 signal Y0 : std_logic := '0';
72
73 --Outputs
74 signal S3 : std_logic;
75 signal S2 : std_logic;
76 signal S1 : std_logic;
77 signal S0 : std_logic;
78 signal Cout : std_logic;
79 -- No clocks detected in port list. Re
80 -- appropriate port name
81
82 --constant <clock>_period : time := 10
83
84 BEGIN
85
```



```
113      -- Stimulus process
114 stim_proc: process
115 begin
116     -- hold reset state for 100 ns.
117     wait for 100 ns;
118
119
120     --wait for <clock>_period*10;
121
122     -- insert stimulus here
123
124     wait;
125 end process;
126
127 END;
```



```
101
102 );
103
104 -- Clock process definitions
105 --<clock>_process :process
106 --begin
107   --<clock> <= '0';
108   --wait for <clock>_period/2;
109   --<clock> <= '1';
110   --wait for <clock>_period/2;
111
112 --end process;
```

and this line



Testbench Generator/Simulation

and then scrolling down....

```
57      Cout : OUT std_logic
58  );
59 END COMPONENT;
60
61
62 --Inputs
63 signal Cin : std_logic := '0';
64 signal X3 : std_logic := '0';
65 signal X2 : std_logic := '0';
66 signal X1 : std_logic := '0';
67 signal X0 : std_logic := '0';
68 signal Y3 : std_logic := '0';
69 signal Y2 : std_logic := '0';
70 signal Y1 : std_logic := '0';
71 signal Y0 : std_logic := '0';
72
73 --Outputs
74 signal S3 : std_logic;
75 signal S2 : std_logic;
76 signal S1 : std_logic;
77 signal S0 : std_logic;
78 signal Cout : std_logic;
79 -- No clocks detected in port list. Re
80 -- appropriate port name
81
82 --constant <clock>_period : time := 10
83
84 BEGIN
85
```



```
113      -- Stimulus process
114 stim_proc: process
115 begin
116     -- hold reset state for 100 ns.
117     wait for 100 ns;
118
119     --|wait for <clock>_period*10;
120
121     -- insert stimulus here
122
123     wait;
124 end process;
125
126
127 END;
```



```
101
102 );
103
104 -- Clock process definitions
105 --<clock>_process :process
106 --begin
107   --<clock> <= '0';
108   --wait for <clock>_period/2;
109   --<clock> <= '1';
110   --wait for <clock>_period/2;
111   --end process;
112
```

and this line.

We can add or stimulus commands here



Testbench Generator/Simulation

and get....

```
112
113
114    -- Stimulus process
115    stim_proc: process
116    begin
117        -- hold reset state for 100 ns.
118        wait for 100 ns;
119        Cin<='0'; X3<='0'; X2<='0'; X1<='0'; X0<='1';
120                Y3<='0'; Y2<='0'; Y1<='0'; Y0<='1';
121        -- X=2, Y=2, Cin=0
122        wait for 100 ns;
123        Cin<='0'; X3<='0'; X2<='0'; X1<='1'; X0<='0';
124                Y3<='0'; Y2<='0'; Y1<='1'; Y0<='0';
125        -- X=4, Y=4, Cin=0
126        wait for 100 ns;
127        Cin<='0'; X3<='0'; X2<='1'; X1<='0'; X0<='0';
128                Y3<='0'; Y2<='1'; Y1<='0'; Y0<='0';
129        -- X=8, Y=8, Cin=0
130        wait for 100 ns;
131        Cin<='0'; X3<='1'; X2<='0'; X1<='0'; X0<='0';
132                Y3<='1'; Y2<='0'; Y1<='0'; Y0<='0';
133        -- X=2, Y=1, Cin=0
134        wait for 100 ns;
135        Cin<='0'; X3<='0'; X2<='0'; X1<='1'; X0<='0';
136                Y3<='0'; Y2<='0'; Y1<='0'; Y0<='1';
137        -- X=7, Y=3, Cin=0
138        wait for 100 ns;
139        Cin<='0'; X3<='0'; X2<='1'; X1<='1'; X0<='1';
140                Y3<='0'; Y2<='0'; Y1<='1'; Y0<='1';
```



Testbench Generator/Simulation

and get.... and

```
112  
113  
114    -- Stimulus process  
115    stim_proc: process  
116    begin  
117        -- hold reset state for 100 ns.  
118        wait for 100 ns;  
119        Cin<='0'; X3<='0'; X2<='0'; X1<='0'; X0<='1';  
120            Y3<='0'; Y2<='0'; Y1<='0'; Y0<='1';  
121        -- X=2, Y=2, Cin=0  
122        wait for 100 ns;  
123        Cin<='0'; X3<='0'; X2<='0'; X1  
124            Y3<='0'; Y2<='0'; Y1  
125        -- X=4, Y=4, Cin=0  
126        wait for 100 ns;  
127        Cin<='0'; X3<='0'; X2<='1'; X1  
128            Y3<='0'; Y2<='1'; Y1  
129        -- X=8, Y=8, Cin=0  
130        wait for 100 ns;  
131        Cin<='0'; X3<='1'; X2<='0'; X1  
132            Y3<='1'; Y2<='0'; Y1<='0'; Y0<='0';  
133        -- X=2, Y=1, Cin=0  
134        wait for 100 ns;  
135        Cin<='0'; X3<='0'; X2<='0'; X1<='1'; X0<='0';  
136            Y3<='0'; Y2<='0'; Y1<='0'; Y0<='1';  
137        -- X=7, Y=3, Cin=0  
138        wait for 100 ns;  
139        Cin<='0'; X3<='0'; X2<='1'; X1<='1'; X0<='1';  
140            Y3<='0'; Y2<='0'; Y1<='1'; Y0<='1';  
  
141        -- X=0, Y=1, Cin=1  
142        wait for 100 ns;  
143        Cin<='1'; X3<='0'; X2<='0'; X1<='0'; X0<='0';  
144            Y3<='0'; Y2<='0'; Y1<='0'; Y0<='1';  
145        wait;  
146    end process;  
147  
148 END;
```

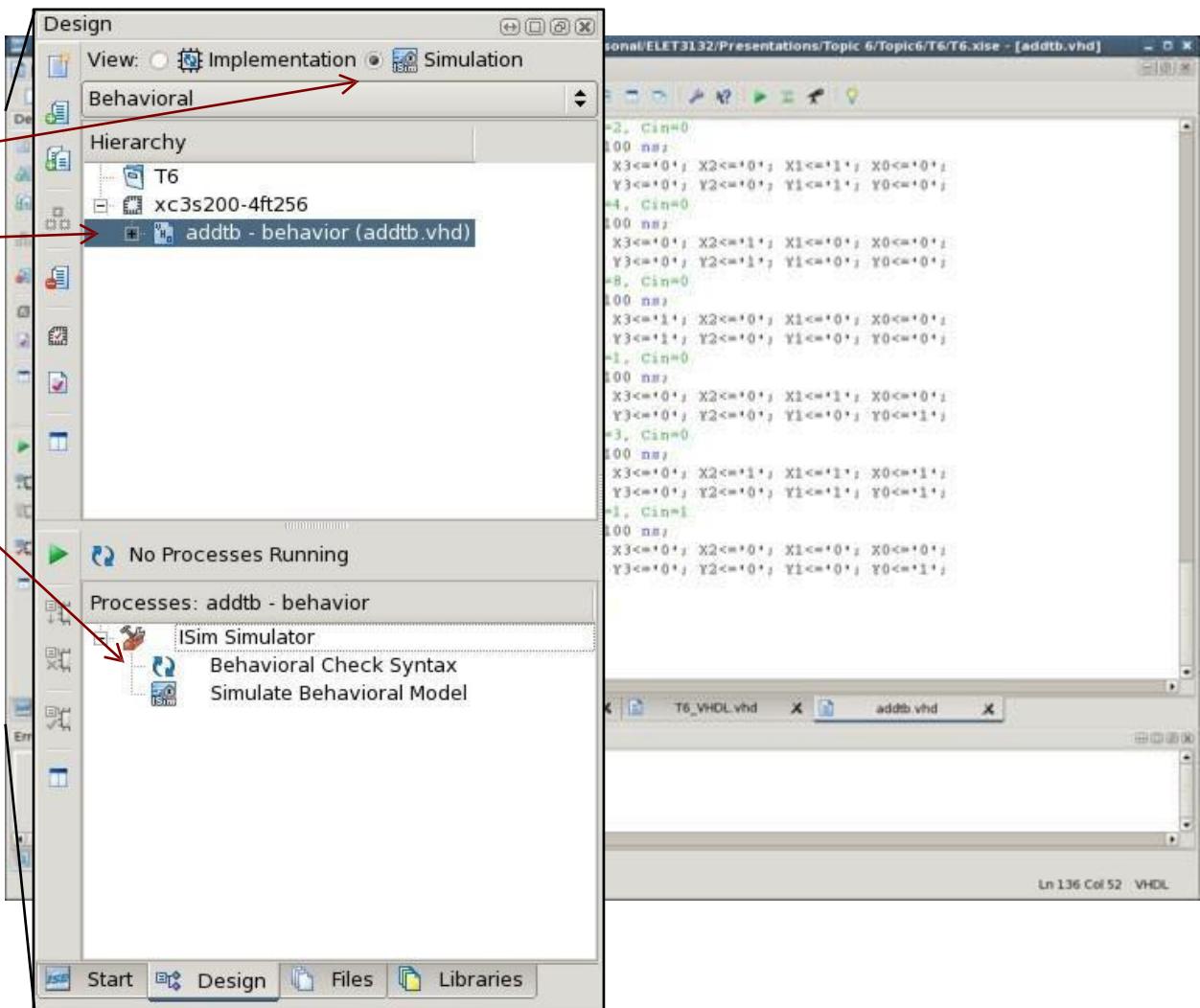


Testbench Generator/Simulation

Save everything

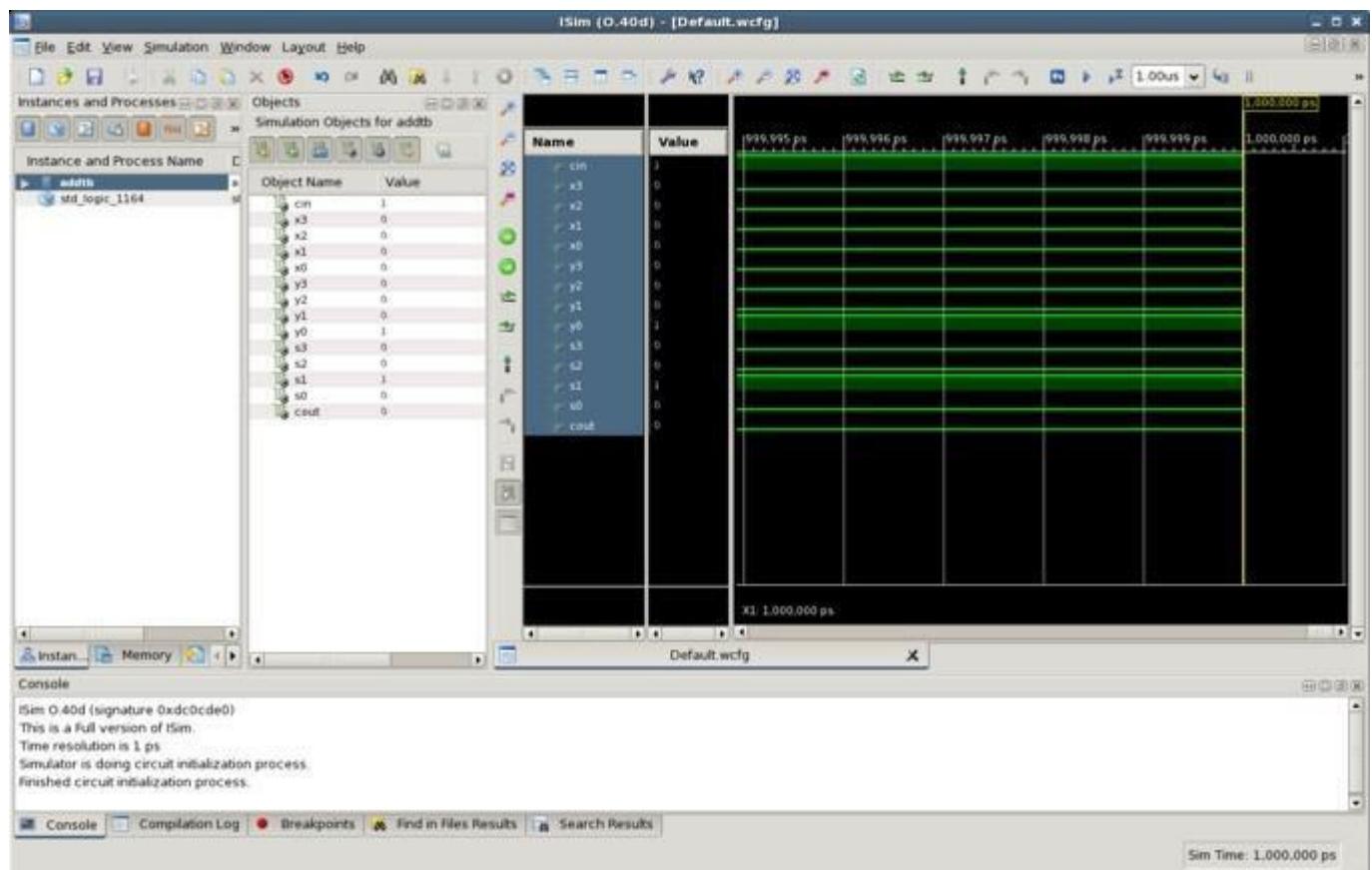
Then click on Simulation

And the test bench and
process appear



Testbench Generator/Simulation

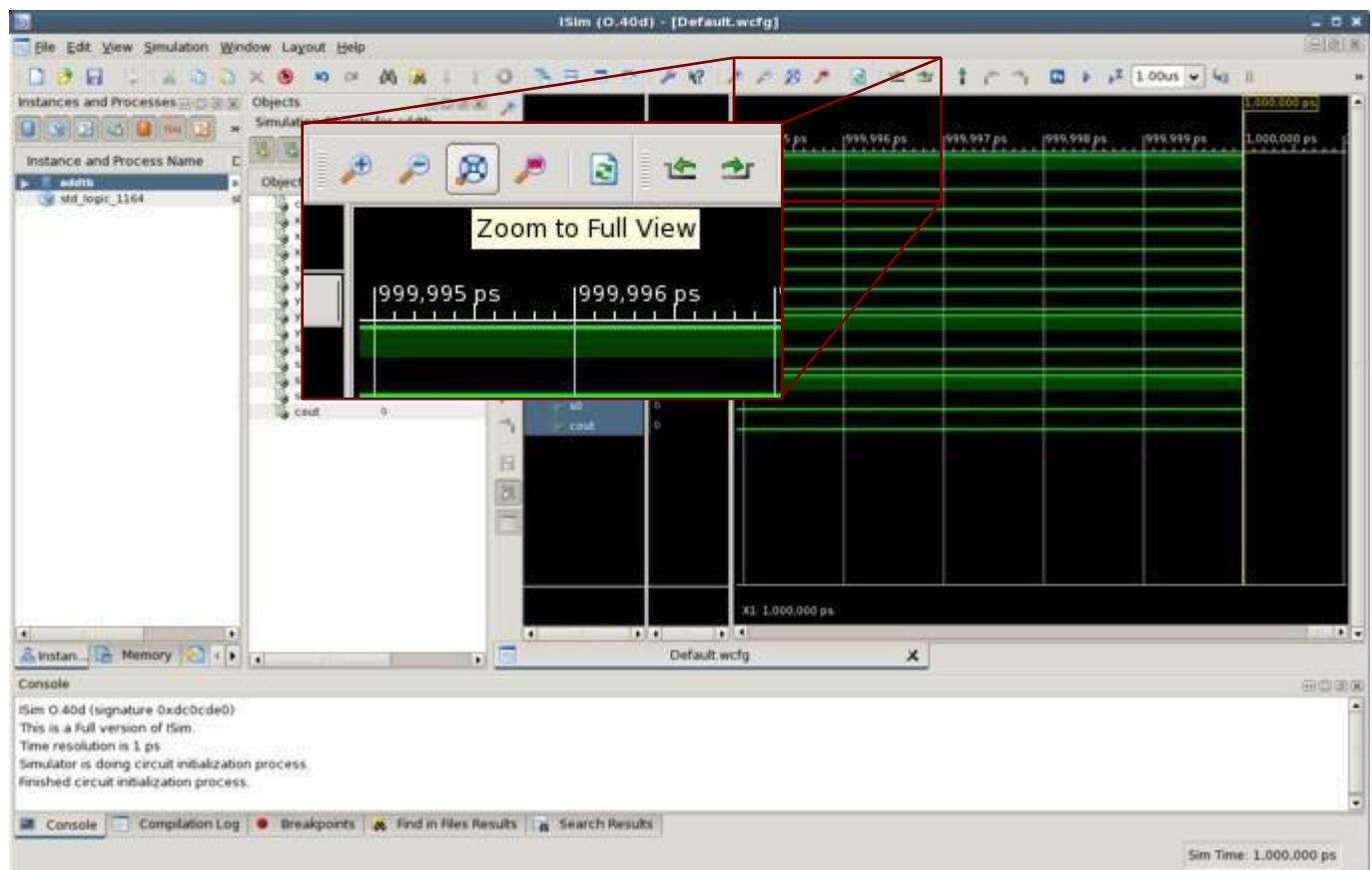
And the ISim window appears



Testbench Generator/Simulation

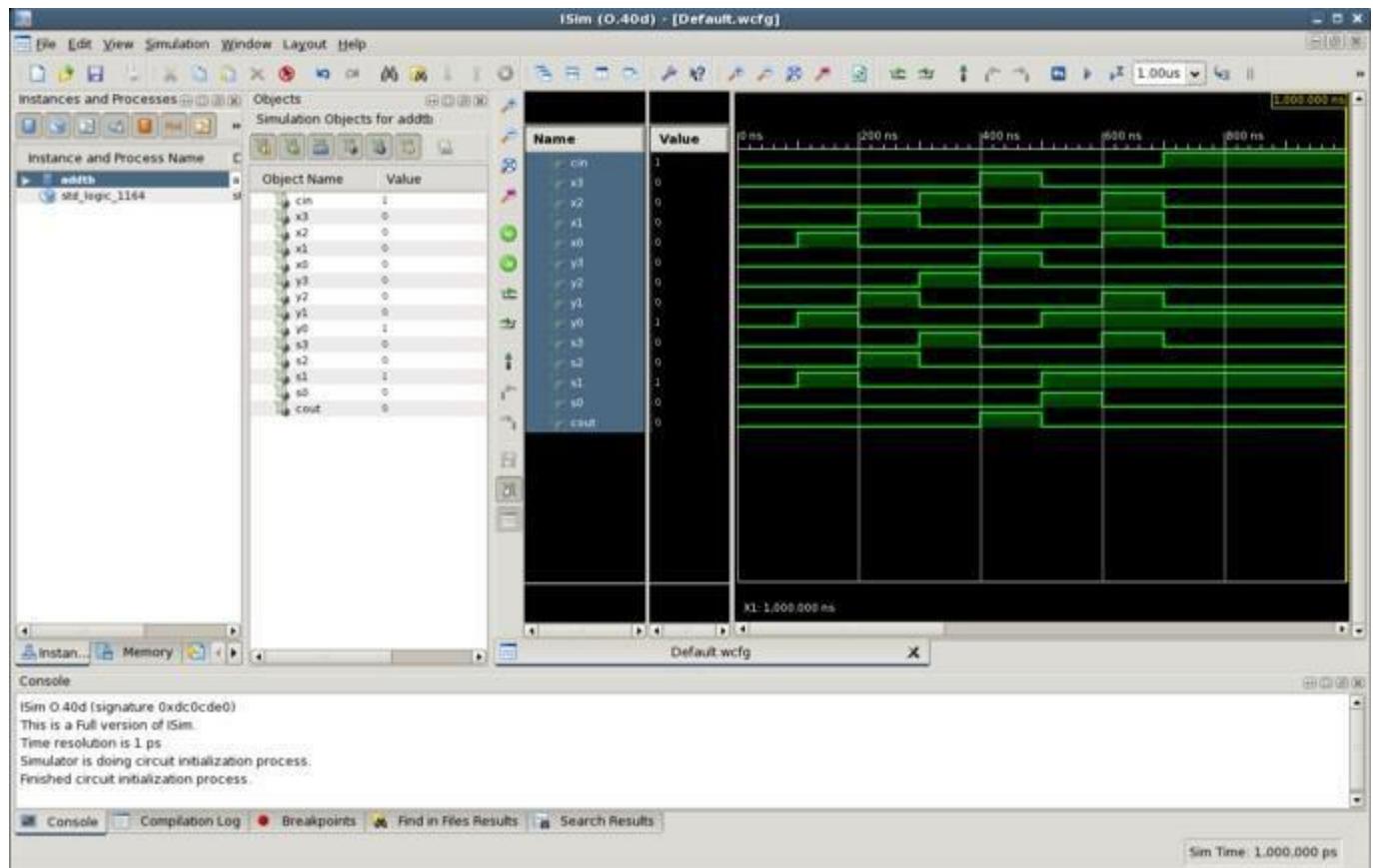
And the ISim window appears

Zoom to full view



Testbench Generator/Simulation

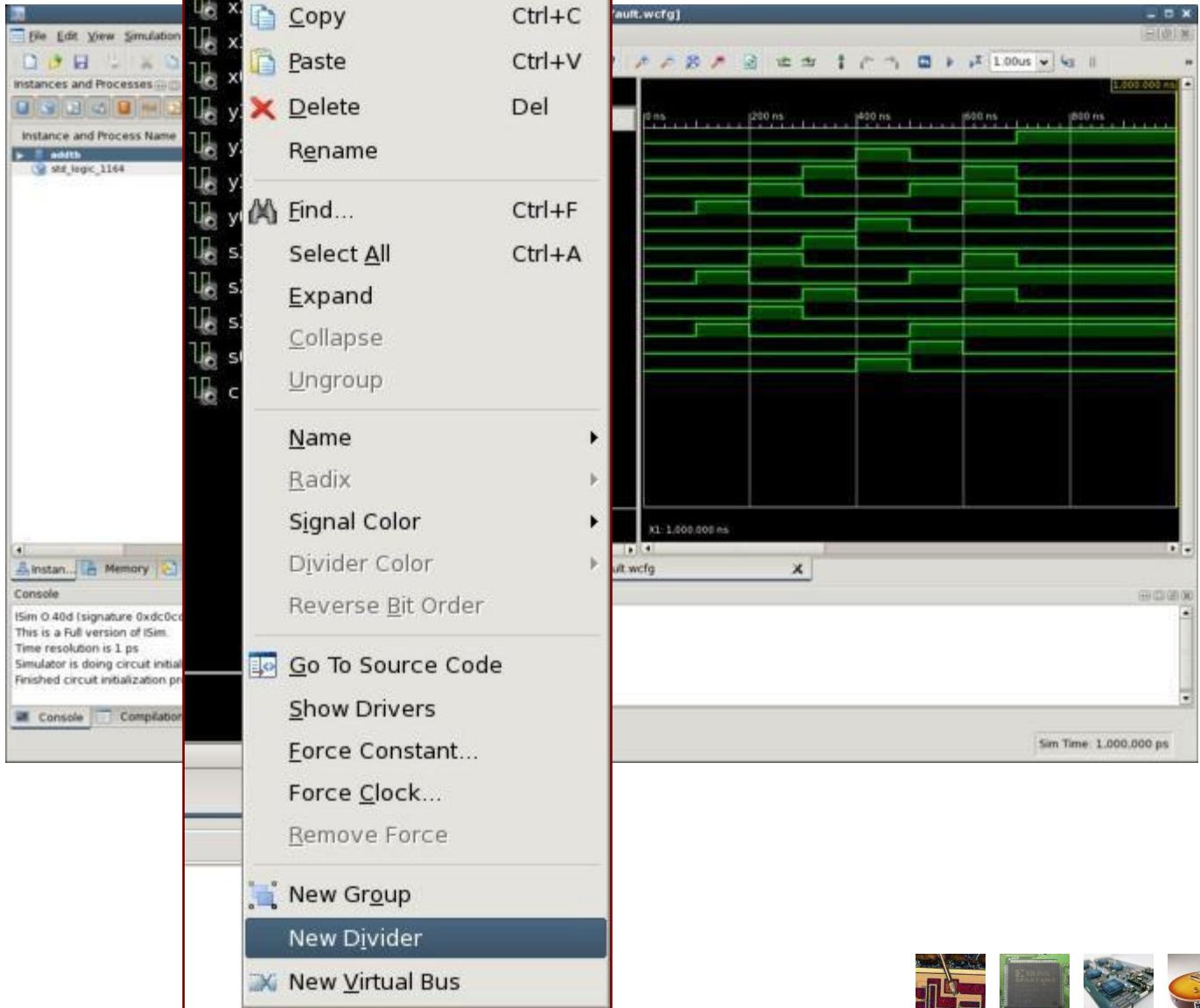
And this view come up,
but it's hard to see with
everything packed
together.



Testbench Generator/Simulation

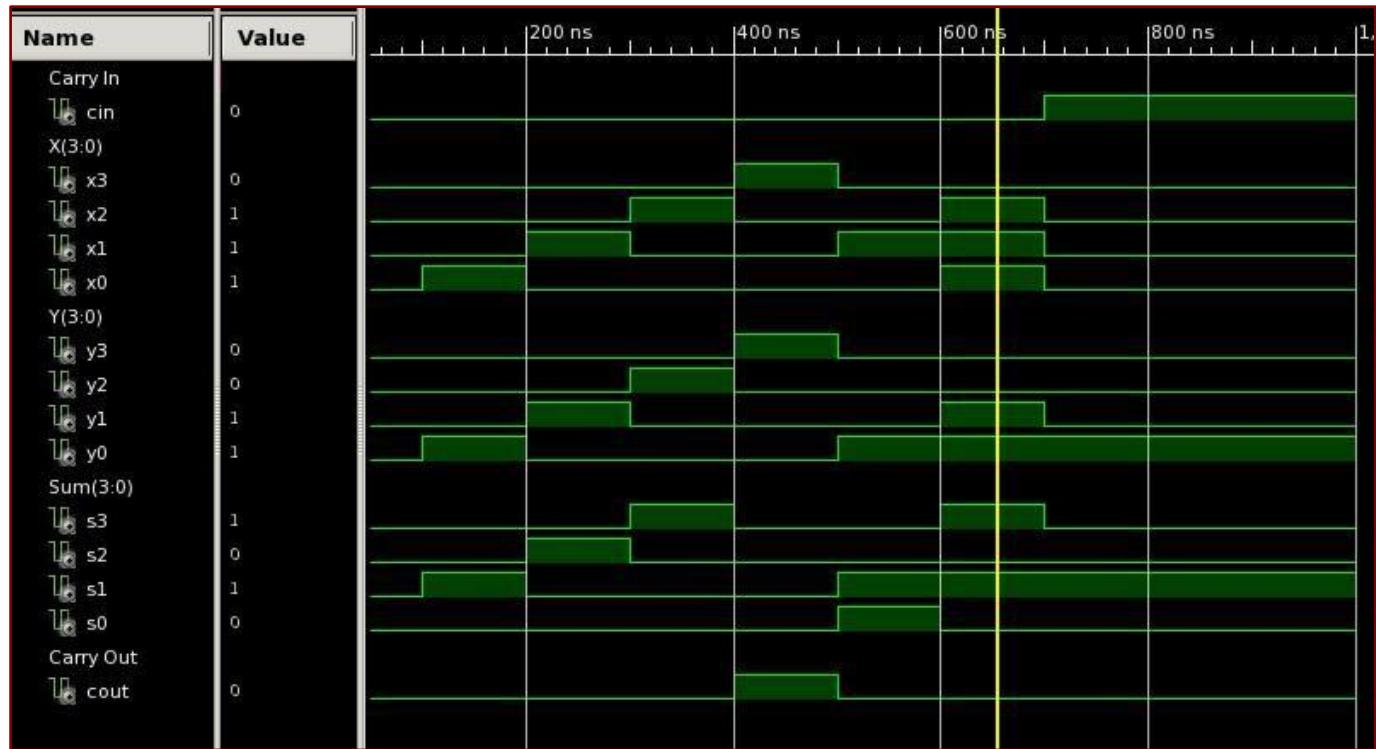
And this view come up, but it's hard to see with everything packed together.

So right clicking pops up this menu and allows us to insert new dividers.



Testbench Generator/Simulation

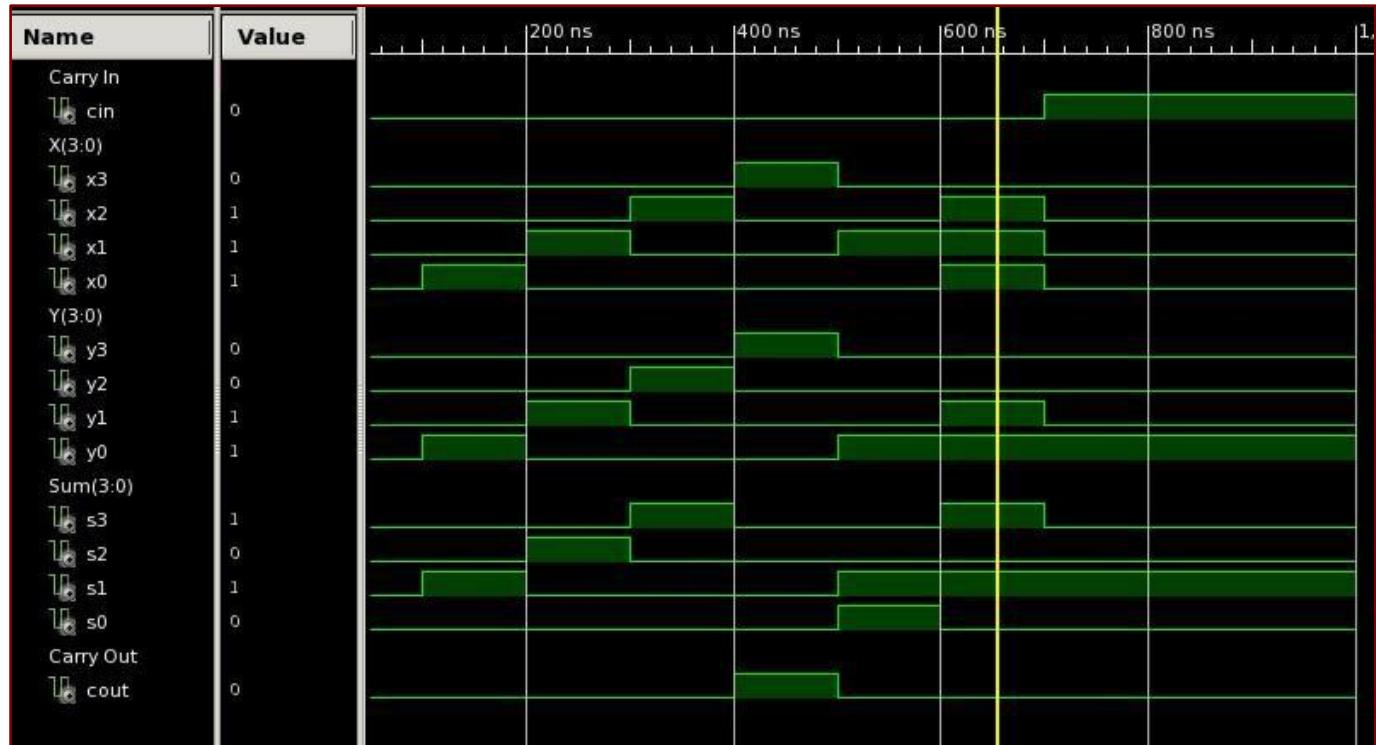
With the new dividers (I also colored them black – an option on the pop up menu), it's easier to see the different states of the test.



Testbench Generator/Simulation

With the new dividers (I also colored them black – an option on the pop up menu), it's easier to see the different states of the test.

In the 1st division (from 100 to 200ns) we can see that X=1 ($x=0001$) and Y=1 ($y=0001$), so the sum is 2 ($s=0010$).

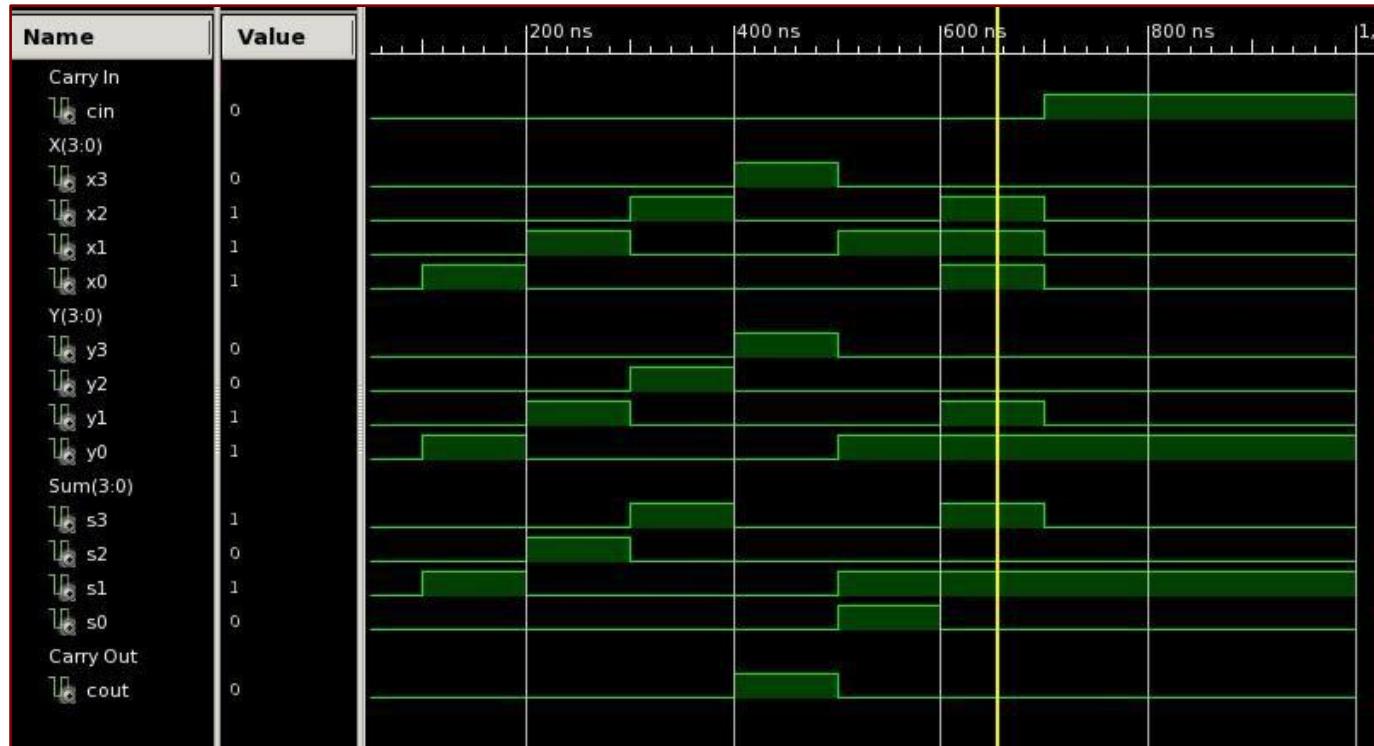


Testbench Generator/Simulation

With the new dividers (I also colored them black – an option on the pop up menu), it's easier to see the different states of the test.

In the 1st division (from 100 to 200ns) we can see that X=1 ($x=0001$) and Y=1 ($y=0001$), so the sum is 2 ($s=0010$).

In the 2nd division we have X = 2, Y = 2, so the Sum = 4



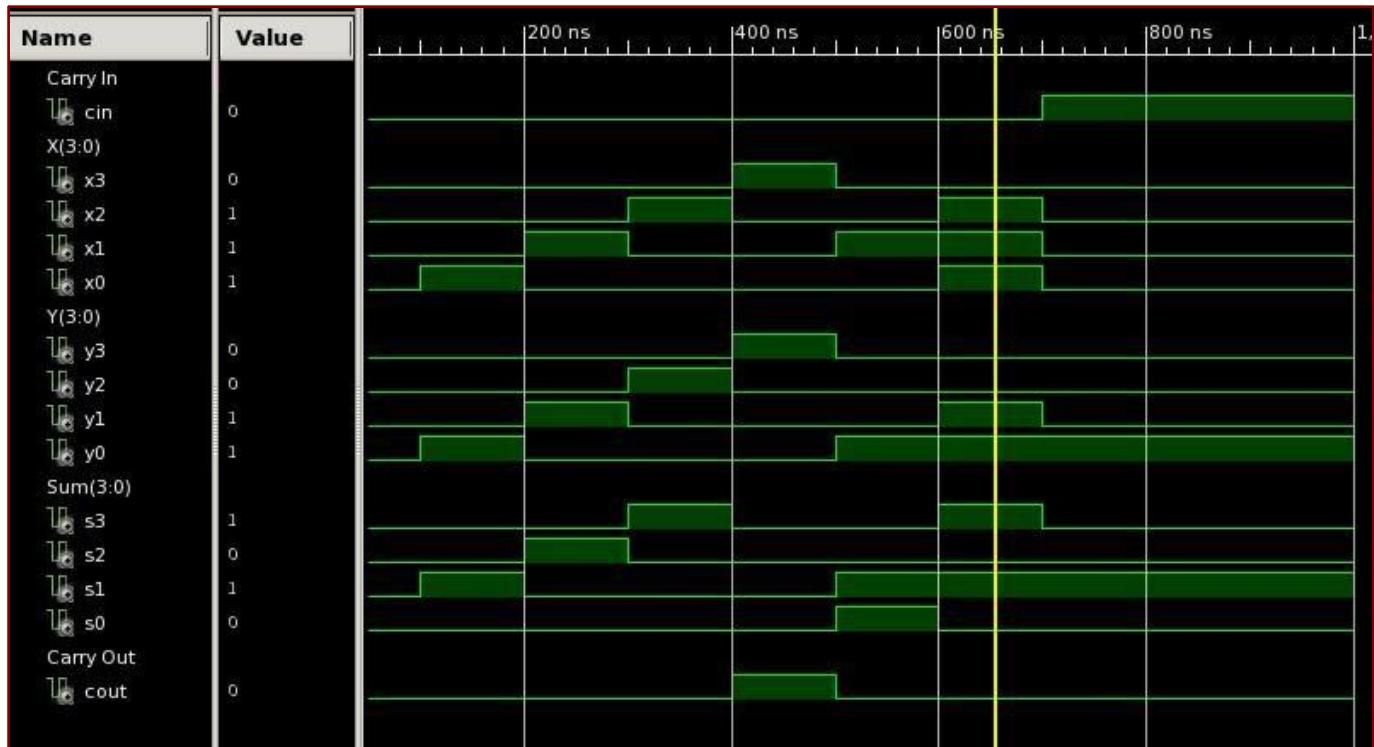
Testbench Generator/Simulation

The marker is showing the 6th division.

X = 7 (x=0111)
Y = 3 (y=0011)

and the sum is 10 with no carry out

S = 10 (s=1010)



Testbench Generator/Simulation

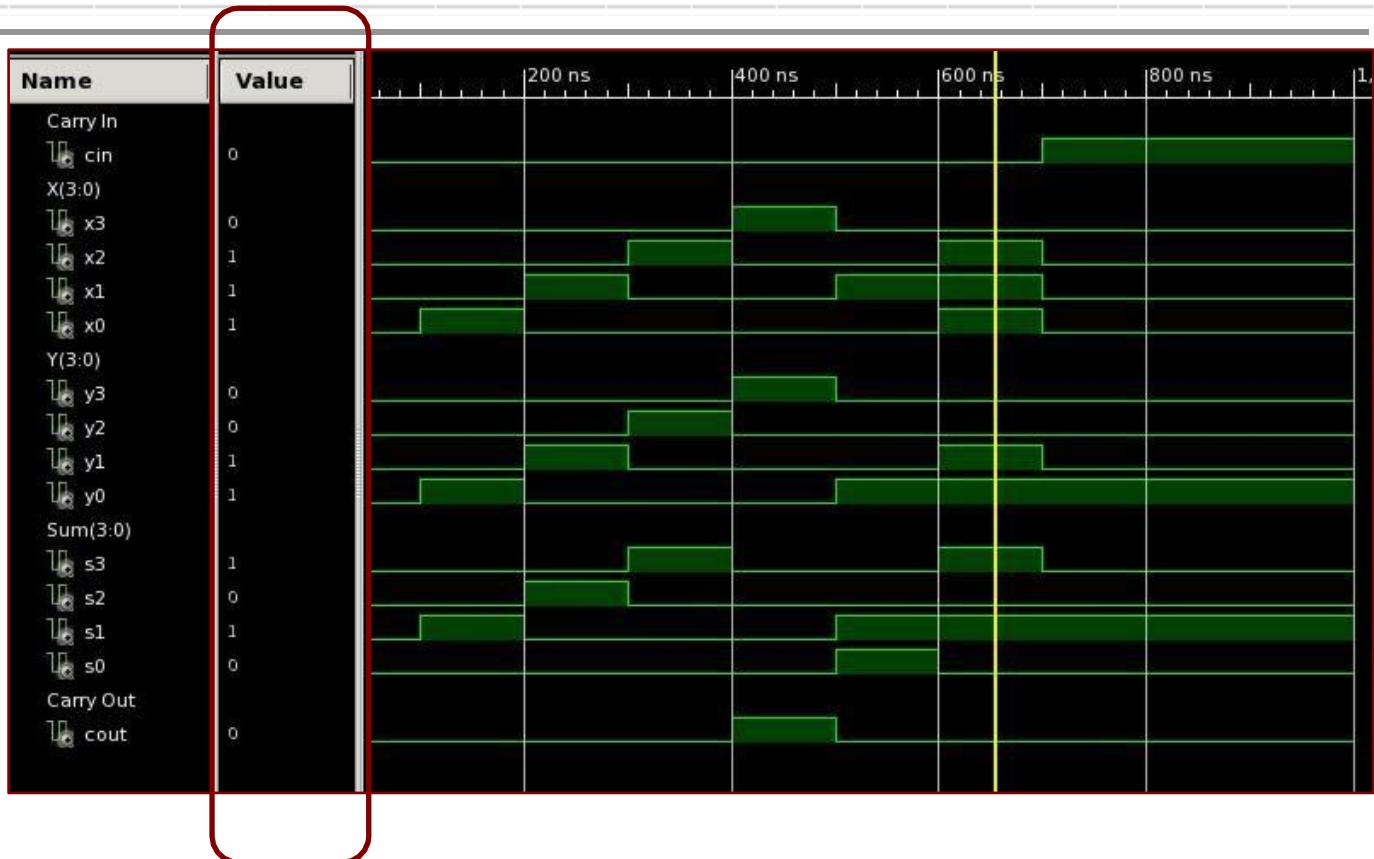
The marker is showing the 6th division.

X = 7 (x=0111)
Y = 3 (y=0011)

and the sum is 10 with no carry out

S = 10 (s=1010)

This also shows up in the "Value" column



Testbench Generator/Simulation

The marker is now showing the 7th division.

Cin = 1

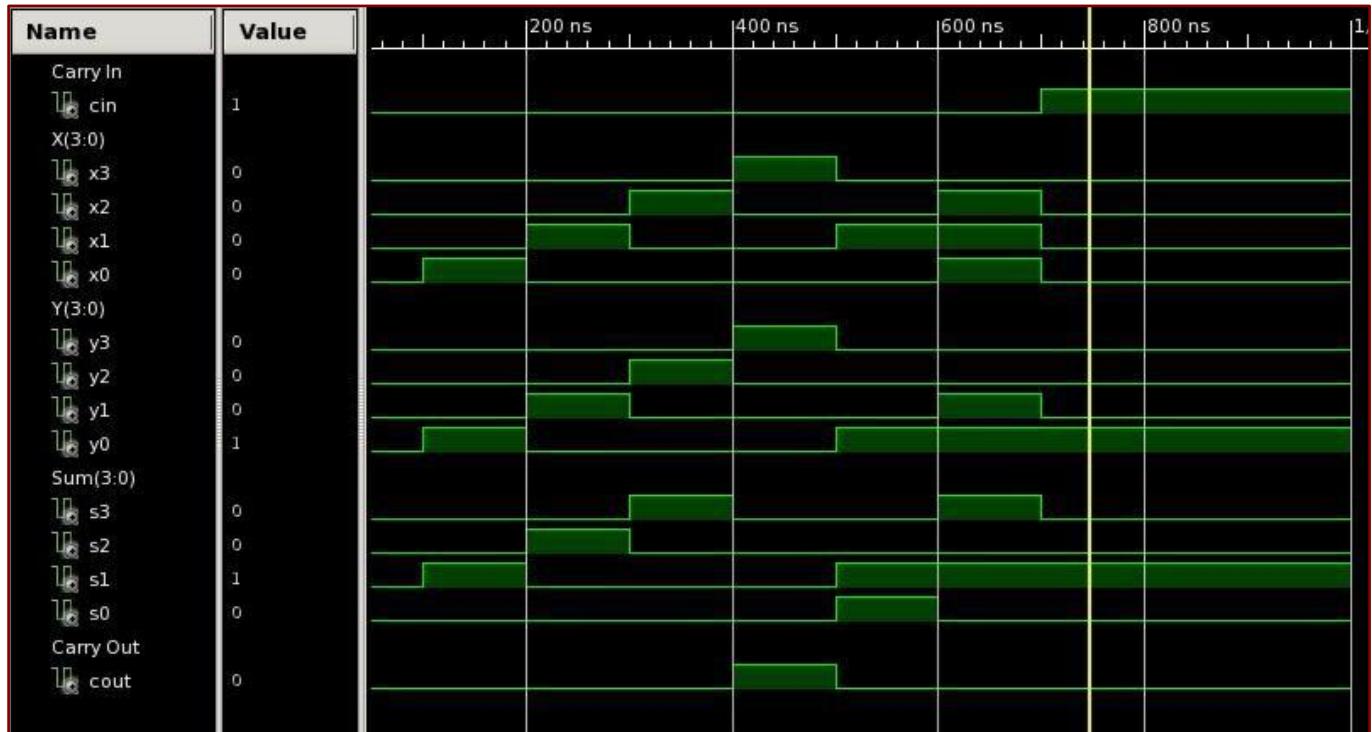
X = 0 (x=0000)

Y = 1 (y=0001)

and the sum is 2 with no carry out

S = 2 (s=0010)

This also shows up in the "Value" column



Download to the Spartan3

- After simulation (if it passes), it is time to download the design to the Spartan3 board for implementation....
 - If it does not pass the simulation, it is time to go back and troubleshoot the logic and start the process again



Troubleshooting VHDL with the Spartan3

- Before you can download your file to the Spartan3 board, it must compile
 - A warning is OK (but this file would not have worked correctly without c1 connected)
- You also must create a Constraints File
 - Assigns the ISE's named signals to actual pins, switches, or LEDs on the Spartan3
 - Ex:
 - There are 8 SPST switches (SW0 – SW7)
 - There are 8 single LEDs
 - There are 4 7-Segment LEDs
 - There are 4 PB switches



Constraints File

```
# LEDs
# NET "LED0" LOC = "K12" ;
# NET "LED1" LOC = "P14" ;
# NET "LED2" LOC = "L12" ;
# NET "LED3" LOC = "N14" ;
# NET "LED4" LOC = "P13" ;
# NET "LED5" LOC = "N12" ;
# NET "LED6" LOC = "P12" ;
# NET "LED7" LOC = "P11" ;

# 50MHz oscillator
# NET "CLK50" LOC = "T9" ;

# pushbuttons
# NET "PB0" LOC = "M13" ;
# NET "PB1" LOC = "M14" ;
# NET "PB2" LOC = "L13" ;
# NET "PB3" LOC = "L14" ;
# NET "RSTPB" LOC = "M13" ;

# switches
# NET "SW0" LOC = "F12" ;
# NET "SW1" LOC = "G12" ;
# NET "SW2" LOC = "H14" ;
# NET "SW3" LOC = "H13" ;
# NET "SW4" LOC = "J14" ;
# NET "SW5" LOC = "J13" ;
# NET "SW6" LOC = "K14" ;
# NET "SW7" LOC = "K13" ;
```

This is an example of a Constraints file. It is an ASCII Text file (also known as DOS txt)

The # in the first columns indicate that the remainder of the line is to be ignored (they are comments)

```
# segments on the 7 seg displays
# NET "ssg<0>" LOC = "E14" ;
# NET "ssg<1>" LOC = "G13" ;
# NET "ssg<2>" LOC = "N15" ;
# NET "ssg<3>" LOC = "P15" ;
# NET "ssg<4>" LOC = "R16" ;
# NET "ssg<5>" LOC = "F13" ;
# NET "ssg<6>" LOC = "N16" ;
# NET "ssg<7>" LOC = "P16" ;

# anodes on the 7 seg displays
# NET "an<0>" LOC = "D14" ;
# NET "an<1>" LOC = "G14" ;
# NET "an<2>" LOC = "F14" ;
# NET "an<3>" LOC = "E13" ;

# video adapter
# NET "blu" LOC = "R11" ;
# NET "grn" LOC = "T12" ;
# NET "red" LOC = "R12" ;
# NET "hs" LOC = "R9" ;
# NET "vs" LOC = "T10" ;
```

This is a template file, so everything is commented out



Constraints File

```
# LEDs
# NET "LED0" LOC = "K12" ;
# NET "LED1" LOC = "P14" ;
# NET "LED2" LOC = "L12" ;
# NET "LED3" LOC = "N14" ;
# NET "LED4" LOC = "P13" ;
# NET "LED5" LOC = "N12" ;
# NET "LED6" LOC = "P12" ;
# NET "LED7" LOC = "P11" ;

# 50MHz oscillator
# NET "CLK50" LOC = "T9" ;

# pushbuttons
# NET "PB0" LOC = "M13" ;
# NET "PB1" LOC = "M14" ;
# NET "PB2" LOC = "L13" ;
# NET "PB3" LOC = "L14" ;
# NET "RSTPB" LOC = "M13" ;

# switches
# NET "SW0" LOC = "F12" ;
# NET "SW1" LOC = "G12" ;
# NET "SW2" LOC = "H14" ;
# NET "SW3" LOC = "H13" ;
# NET "SW4" LOC = "J14" ;
# NET "SW5" LOC = "J13" ;
# NET "SW6" LOC = "K14" ;
# NET "SW7" LOC = "K13" ;
```

The names(in quotes) associated with the keyword NET are signal names in the Xilinx ISE – the names we have given the signals in our VHDL files.

```
# segments on the 7 seg displays
# NET "ssg<0>" LOC = "E14" ;
# NET "ssg<1>" LOC = "G13" ;
# NET "ssg<2>" LOC = "N15" ;
# NET "ssg<3>" LOC = "P15" ;
# NET "ssg<4>" LOC = "R16" ;
# NET "ssg<5>" LOC = "F13" ;
# NET "ssg<6>" LOC = "N16" ;
# NET "ssg<7>" LOC = "P16" ;

# anodes on the 7 seg displays
# NET "an<0>" LOC = "D14" ;
# NET "an<1>" LOC = "G14" ;
# NET "an<2>" LOC = "F14" ;
# NET "an<3>" LOC = "E13" ;

# video adapter
# NET "blu" LOC = "R11" ;
# NET "grn" LOC = "T12" ;
# NET "red" LOC = "R12" ;
# NET "hs" LOC = "R9" ;
# NET "vs" LOC = "T10" ;
```



Constraints File

```
# LEDs
# NET "LED0" LOC = "K12" ;
# NET "LED1" LOC = "P14" ;
# NET "LED2" LOC = "L12" ;
# NET "LED3" LOC = "N14" ;
# NET "LED4" LOC = "P13" ;
# NET "LED5" LOC = "N12" ;
# NET "LED6" LOC = "P12" ;
# NET "LED7" LOC = "P11" ;

# 50MHz oscillator
# NET "CLK50" LOC = "T9" ;

# pushbuttons
# NET "PB0" LOC = "M13" ;
# NET "PB1" LOC = "M14" ;
# NET "PB2" LOC = "L13" ;
# NET "PB3" LOC = "L14" ;
# NET "RSTPB" LOC = "M13" ;

# switches
# NET "SW0" LOC = "F12" ;
# NET "SW1" LOC = "G12" ;
# NET "SW2" LOC = "H14" ;
# NET "SW3" LOC = "H13" ;
# NET "SW4" LOC = "J14" ;
# NET "SW5" LOC = "J13" ;
# NET "SW6" LOC = "K14" ;
# NET "SW7" LOC = "K13" ;
```

The names (in quotes) associated with the LOC keyword are associated with the Spartan3 board.

These names must not be changed!!!

```
# segments on the 7 seg displays
# NET "ssg<0>" LOC = "E14" ;
# NET "ssg<1>" LOC = "G13" ;
# NET "ssg<2>" LOC = "N15" ;
# NET "ssg<3>" LOC = "P15" ;
# NET "ssg<4>" LOC = "R16" ;
# NET "ssg<5>" LOC = "F13" ;
# NET "ssg<6>" LOC = "N16" ;
# NET "ssg<7>" LOC = "P16" ;

# anodes on the 7 seg displays
# NET "an<0>" LOC = "D14" ;
# NET "an<1>" LOC = "G14" ;
# NET "an<2>" LOC = "F14" ;
# NET "an<3>" LOC = "E13" ;

# video adapter
# NET "blu" LOC = "R11" ;
# NET "grn" LOC = "T12" ;
# NET "red" LOC = "R12" ;
# NET "hs" LOC = "R9" ;
# NET "vs" LOC = "T10" ;
```



Constraints File

```
# LEDs
# NET "LED0" LOC = "K12" ;
# NET "LED1" LOC = "P14" ;
# NET "LED2" LOC = "L12" ;
# NET "LED3" LOC = "N14" ;
# NET "LED4" LOC = "P13" ;
# NET "LED5" LOC = "N12" ;
# NET "LED6" LOC = "P12" ;
# NET "LED7" LOC = "P11" ;

# 50MHz oscillator
# NET "CLK50" LOC = "T9" ;

# pushbuttons
# NET "PB0" LOC = "M13" ;
# NET "PB1" LOC = "M14" ;
# NET "PB2" LOC = "L13" ;
# NET "PB3" LOC = "L14" ;
# NET "RSTPB" LOC = "M13" ;

# switches
# NET "SW0" LOC = "F12" ;
# NET "SW1" LOC = "G12" ;
# NET "SW2" LOC = "H14" ;
# NET "SW3" LOC = "H13" ;
# NET "SW4" LOC = "J14" ;
# NET "SW5" LOC = "J13" ;
# NET "SW6" LOC = "K14" ;
# NET "SW7" LOC = "K13" ;
```

The names (in quotes) associated with the LOC keyword are associated with the Spartan3 board.

These names must not be changed!!!

To use the constraint files, you must remove the # from the lines you want to use, then make the assignments. You may leave, as comments, the remaining lines or delete them

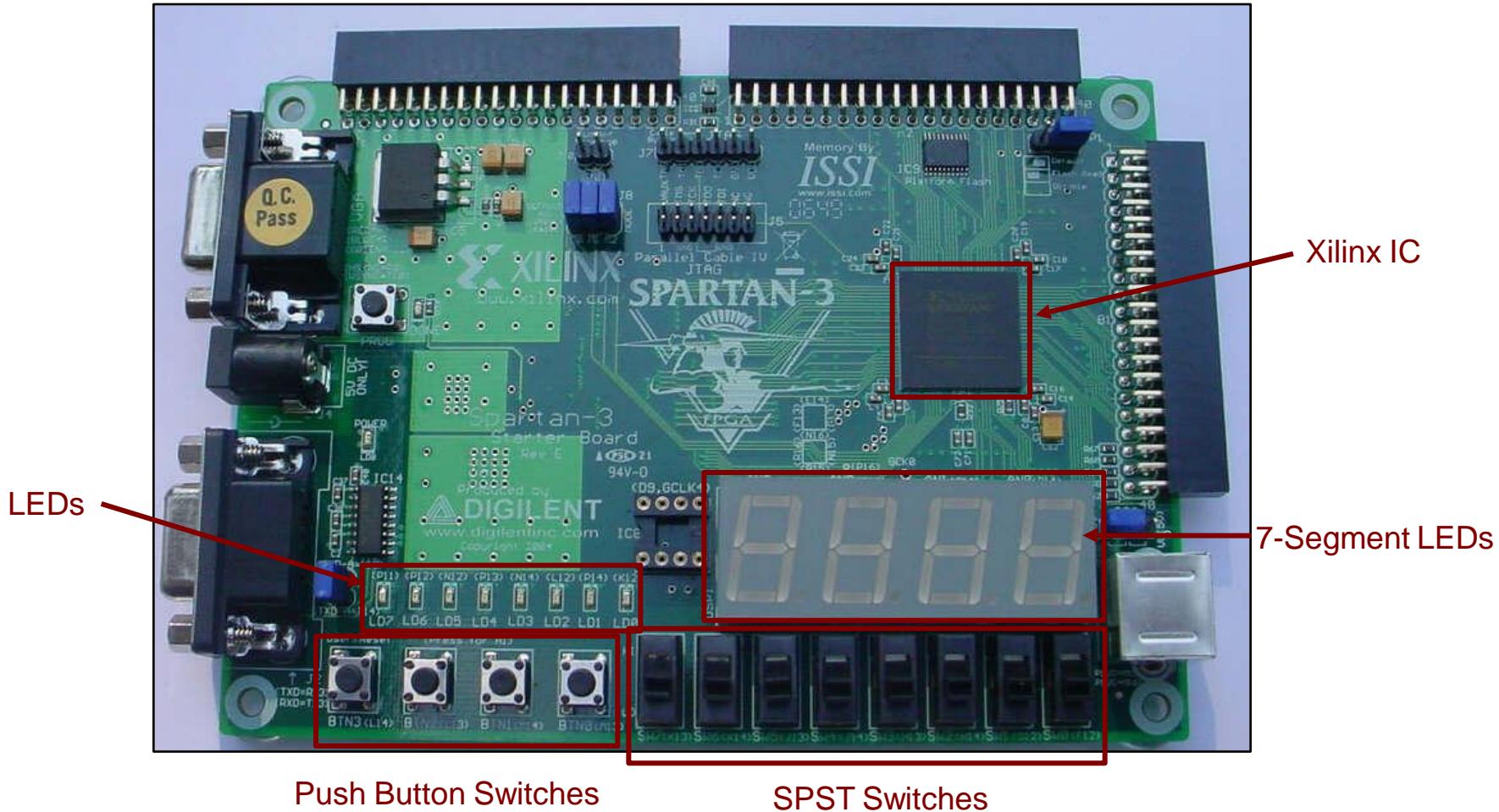
```
# segments on the 7 seg displays
# NET "ssg<0>" LOC = "E14" ;
# NET "ssg<1>" LOC = "G13" ;
# NET "ssg<2>" LOC = "N15" ;
# NET "ssg<3>" LOC = "P15" ;
# NET "ssg<4>" LOC = "R16" ;
# NET "ssg<5>" LOC = "F13" ;
# NET "ssg<6>" LOC = "N16" ;
# NET "ssg<7>" LOC = "P16" ;

# anodes on the 7 seg displays
# NET "an<0>" LOC = "D14" ;
# NET "an<1>" LOC = "G14" ;
# NET "an<2>" LOC = "F14" ;
# NET "an<3>" LOC = "E13" ;

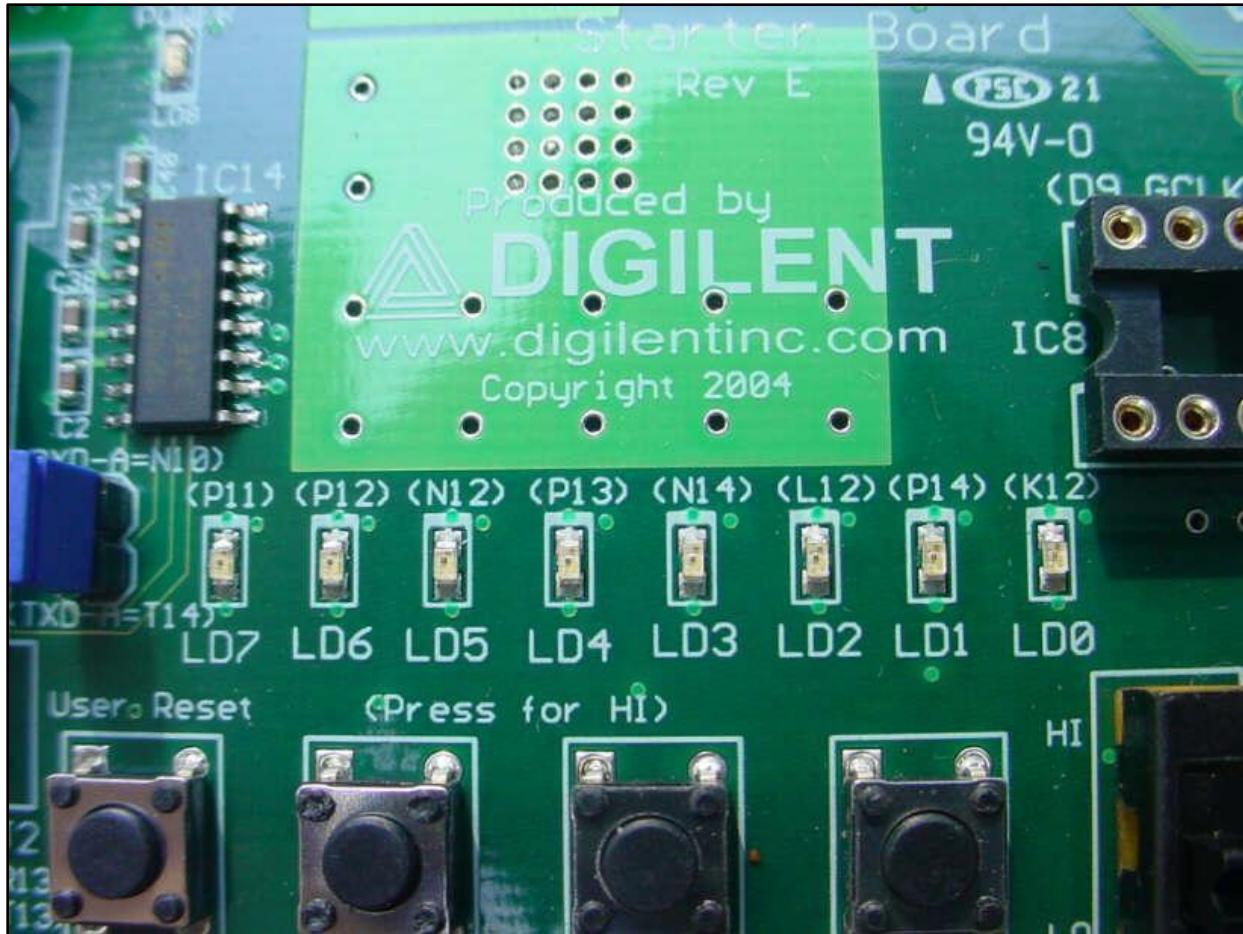
# video adapter
# NET "blu" LOC = "R11" ;
# NET "grn" LOC = "T12" ;
# NET "red" LOC = "R12" ;
# NET "hs" LOC = "R9" ;
# NET "vs" LOC = "T10" ;
```



Spatan3 Board



Spartan3 LED Close Up

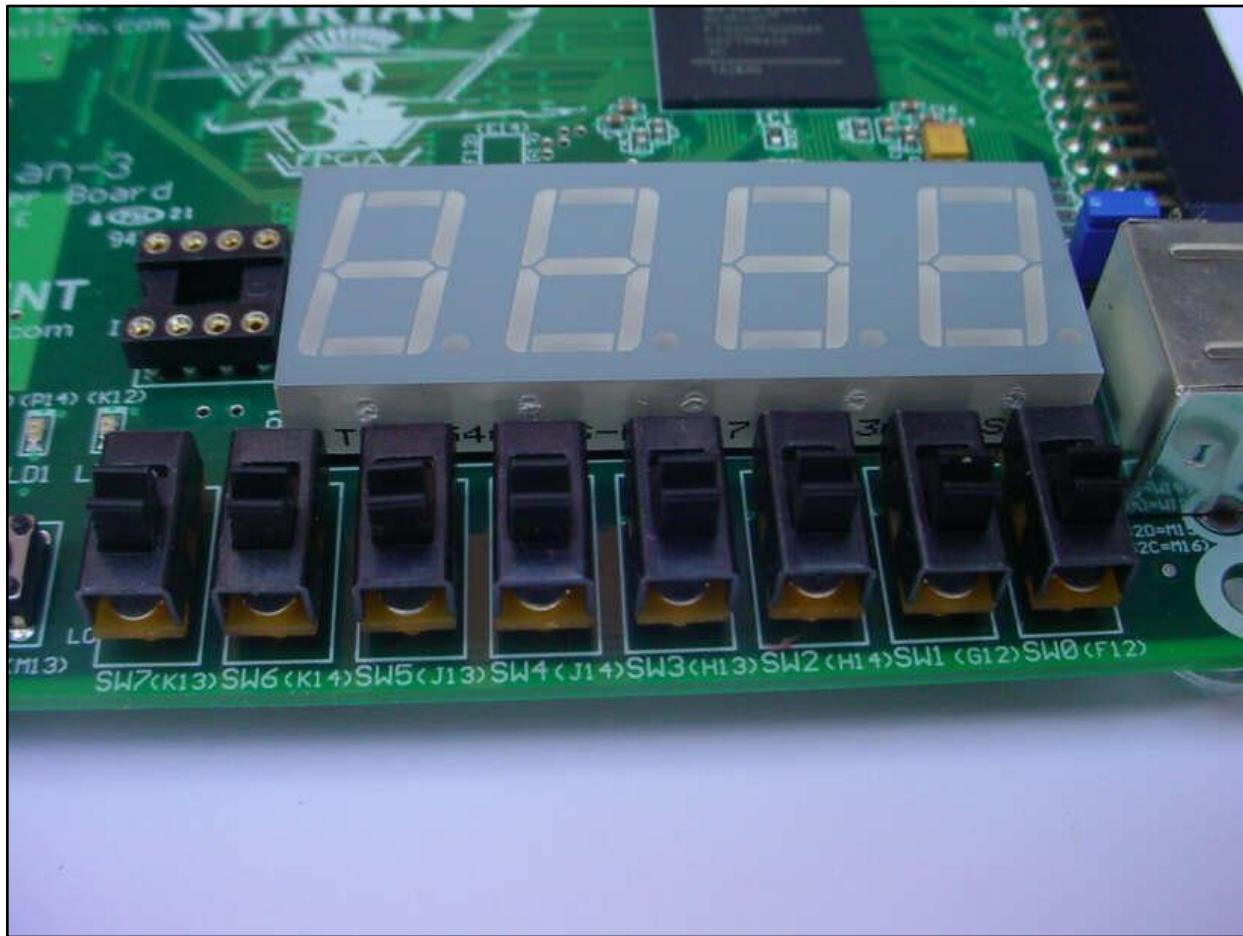


Note the locations
of the LEDs:
P11, P12, N12,
P13, N14, L12,
P14, and K12



Spartan3 Switch Close Up

Note the locations
of the switches:
K13, K14, J13,
J14, H13, H14,
G12, and F12



Minimal Constraints File

```
# LEDs  
NET "S0" LOC = "K12" ;  
NET "S1" LOC = "P14" ;  
NET "S2" LOC = "L12" ;  
NET "S2" LOC = "N14" ;  
NET "Cout" LOC = "P13" ;  
  
# switches  
NET "X0" LOC = "F12" ;  
NET "X1" LOC = "G12" ;  
NET "X2" LOC = "H14" ;  
NET "X3" LOC = "H13" ;  
NET "Y0" LOC = "J14" ;  
NET "Y1" LOC = "J13" ;  
NET "Y2" LOC = "K14" ;  
NET "Y3" LOC = "K13" ;  
  
# pushbuttons  
NET "Cin" LOC = "M13" ;
```

The LEDs are assigned to S0, S1, S2, S3, and Carry out

The input SPST switches are assigned as follows:

X0 = SW0

X1 = SW1

X2 = SW2

X3 = SW3

Y0 = SW4

Y1 = SW5

Y2 = SW6

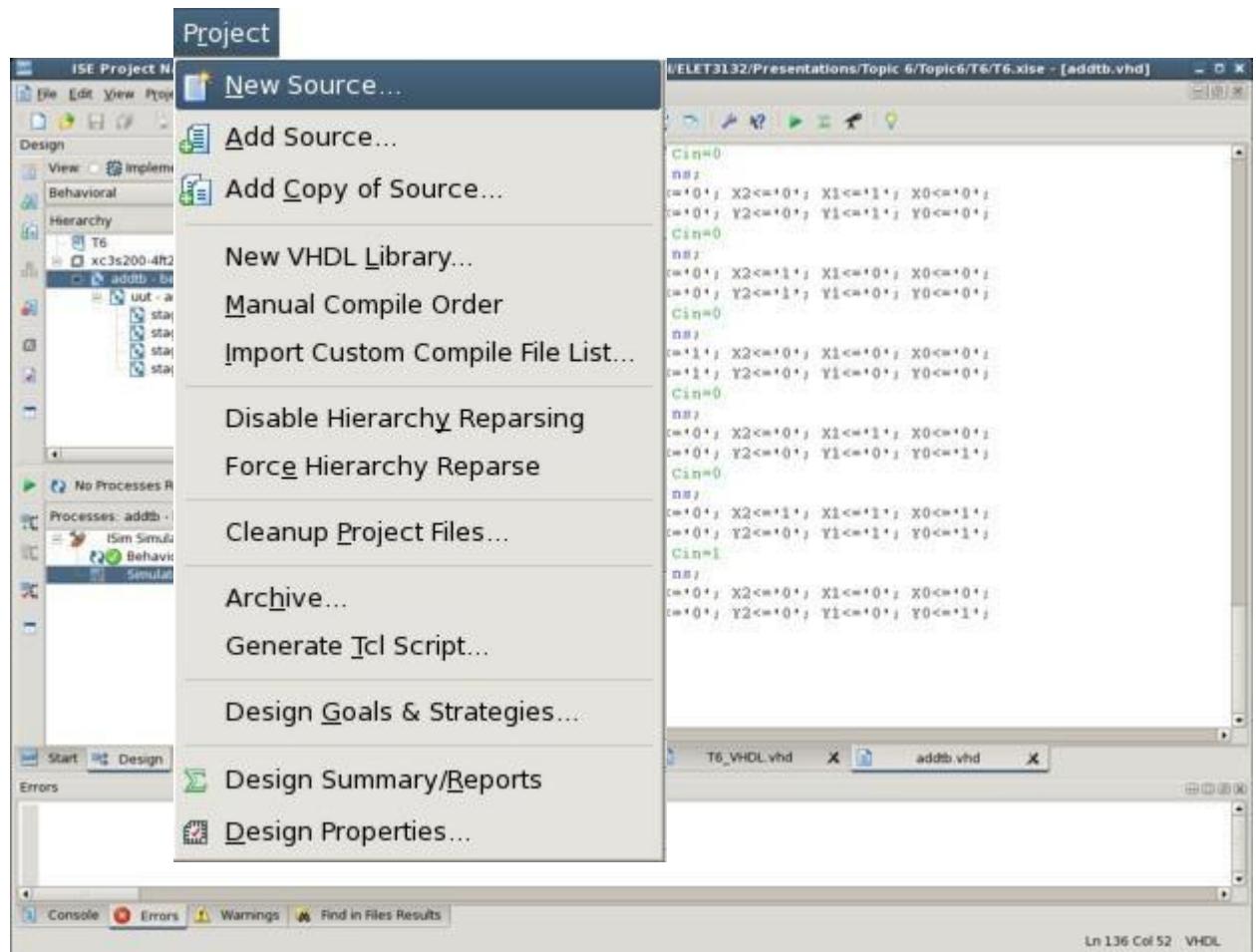
Y3 = SW7

There were no switches left for the Carry in, so Carry in was assigned to Push Button 0, which means it will have to be held steady when the design is executed



Add Constraints File

To add a constraints file, click on Project → New Source

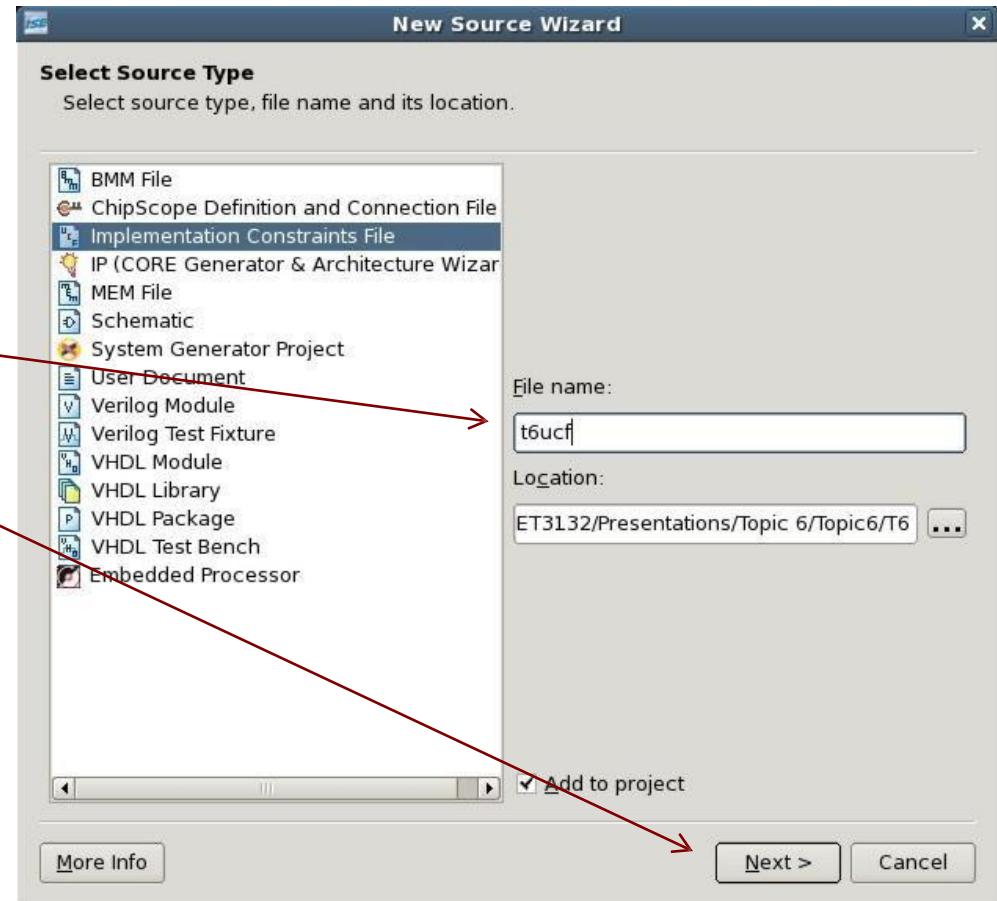


Add Constraints File

To add a constraints file, click on Project → New Source

The New Source Wizard appears.

Type in the name of your UCF file, and click Next



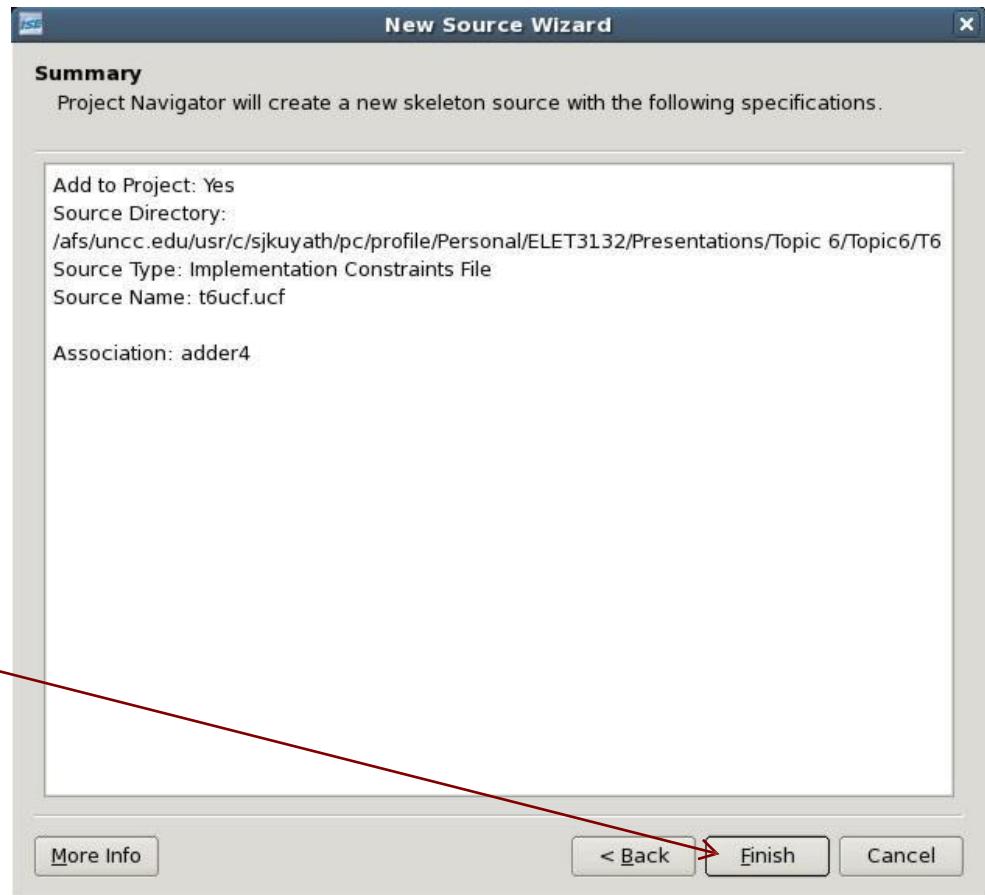
Add Constraints File

To add a constraints file, click on Project → New Source

The New Source Wizard appears.

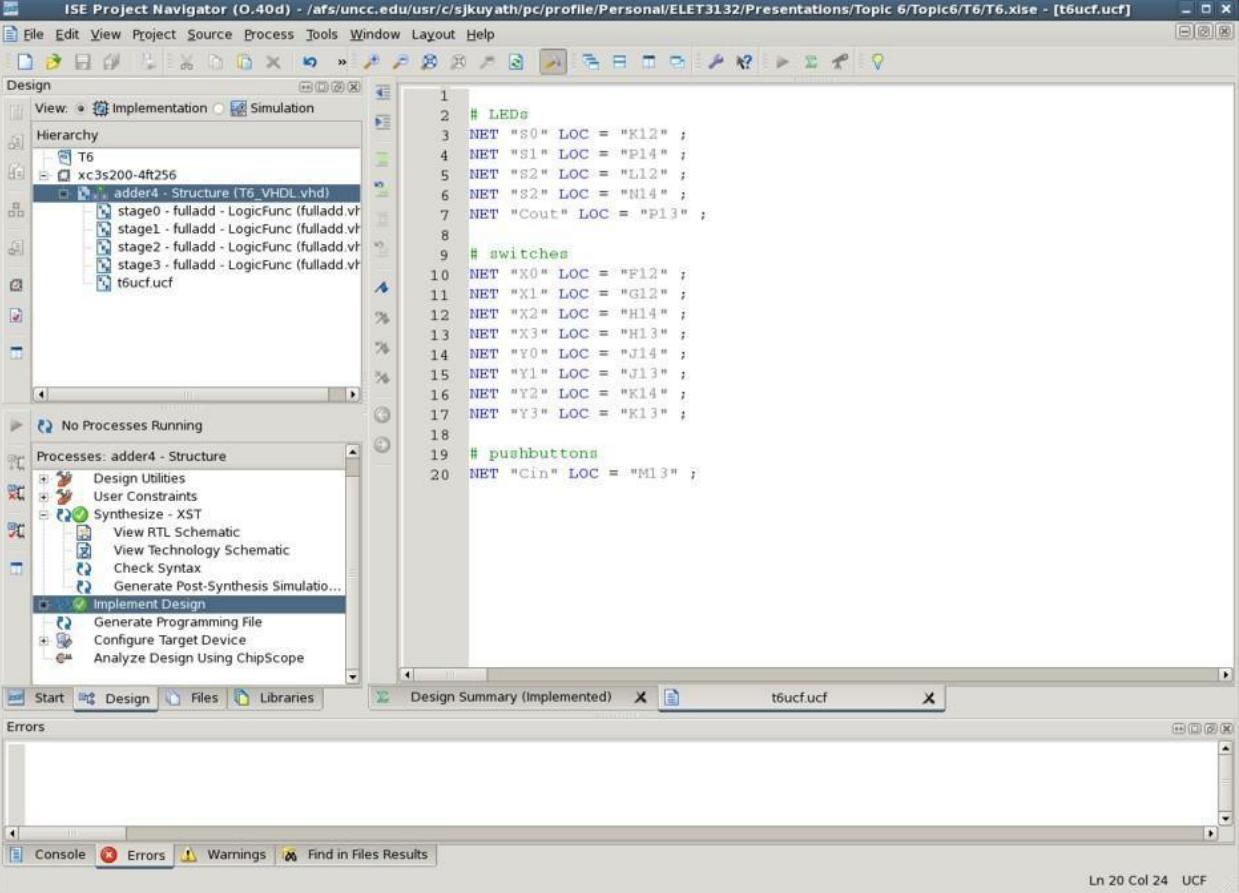
Type in the name of your UCF file, and click Next.

The verification screen appears. If everything is ok, click Finish



Add Constraints File

Type in or copy-paste the UCF file



The screenshot shows the ISE Project Navigator interface. The left pane displays the project hierarchy under 'T6' and 'xc3s200-4ft256'. The right pane shows the contents of the 't6.ucf' file:

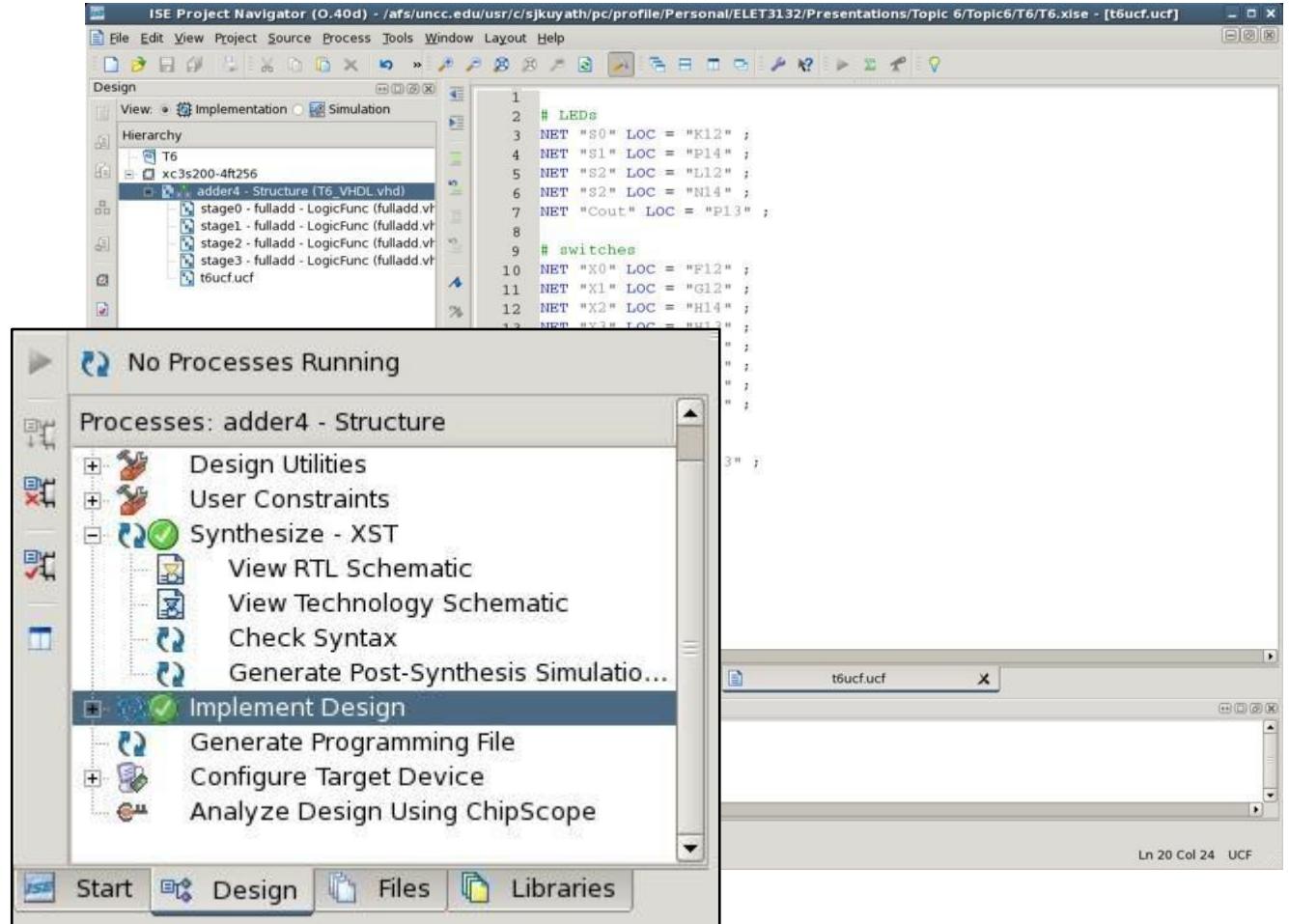
```
1 # LEDs
2 NET "S0" LOC = "K12" ;
3 NET "S1" LOC = "P14" ;
4 NET "S2" LOC = "L12" ;
5 NET "S3" LOC = "M14" ;
6 NET "Cout" LOC = "P13" ;
7
8 # switches
9 NET "X0" LOC = "P12" ;
10 NET "X1" LOC = "G12" ;
11 NET "X2" LOC = "H14" ;
12 NET "X3" LOC = "H13" ;
13 NET "Y0" LOC = "J14" ;
14 NET "Y1" LOC = "J13" ;
15 NET "Y2" LOC = "K14" ;
16 NET "Y3" LOC = "K13" ;
17
18 # pushbuttons
19 NET "Cin" LOC = "M13" ;
```



Add Constraints File

Type in or copy-paste the UCF file

Save everything and then double click on Implement Design. If everything is correct, you'll get the green check mark



Add Constraints File

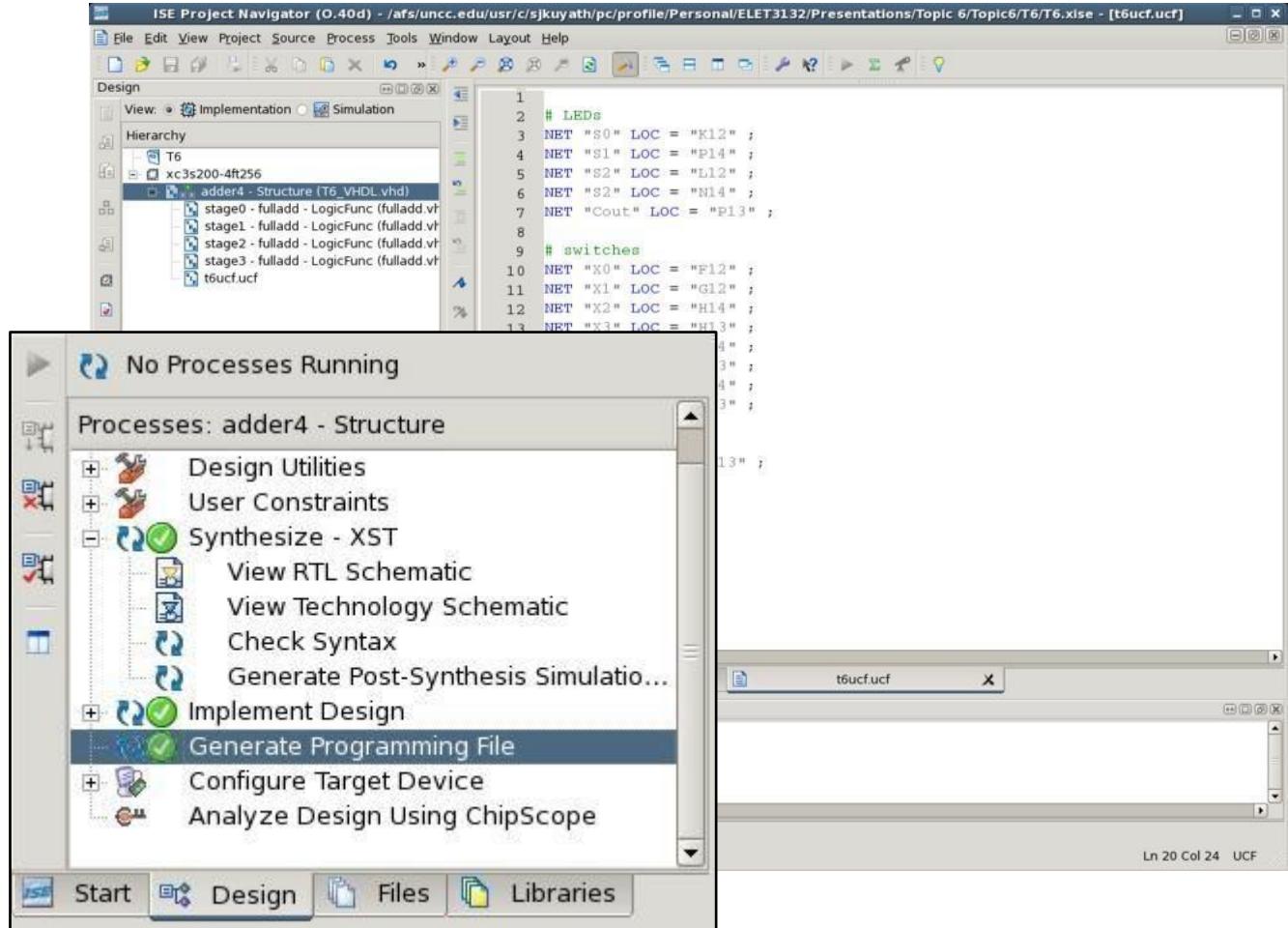
Type in or copy-paste the UCF file

Save everything and then double click on Implement Design. If everything is correct, you'll get the green check mark

Double click on Generate Programming File

If everything is correct, you'll get another green check

And, it's time to move to the download machine

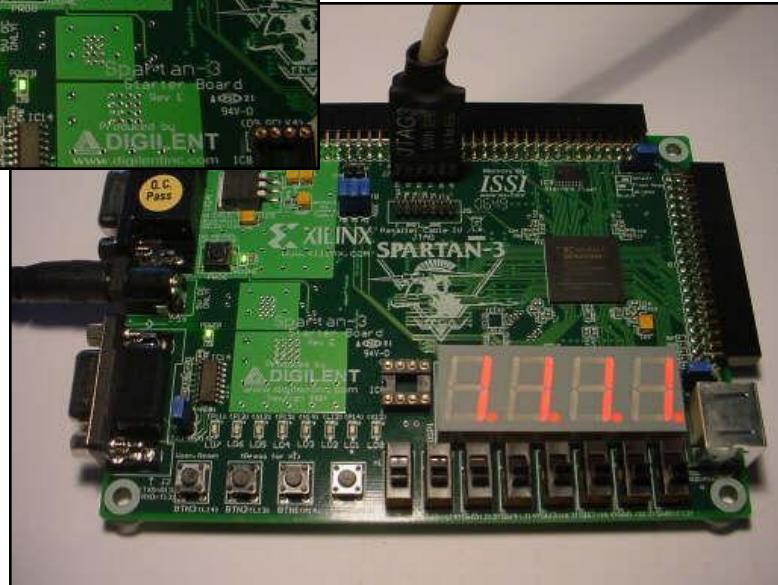
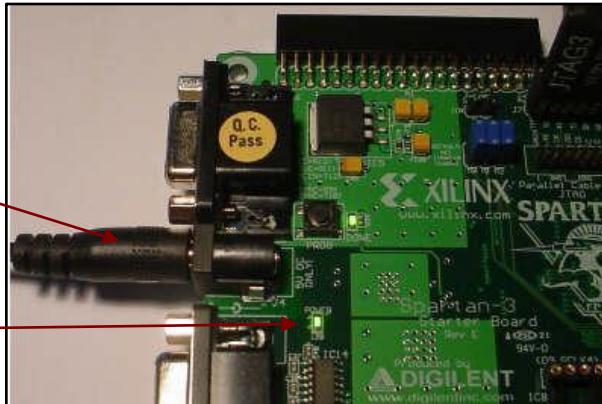


Download and Execute

Before you can download the design to the Xilinx board, you have to make a few connections....

Apply power to the board

and the power light should come on



You'll notice that the 4 7-segment LEDs may cycle through the digits 1 – 9 continuously, until the Spartan3 is programmed

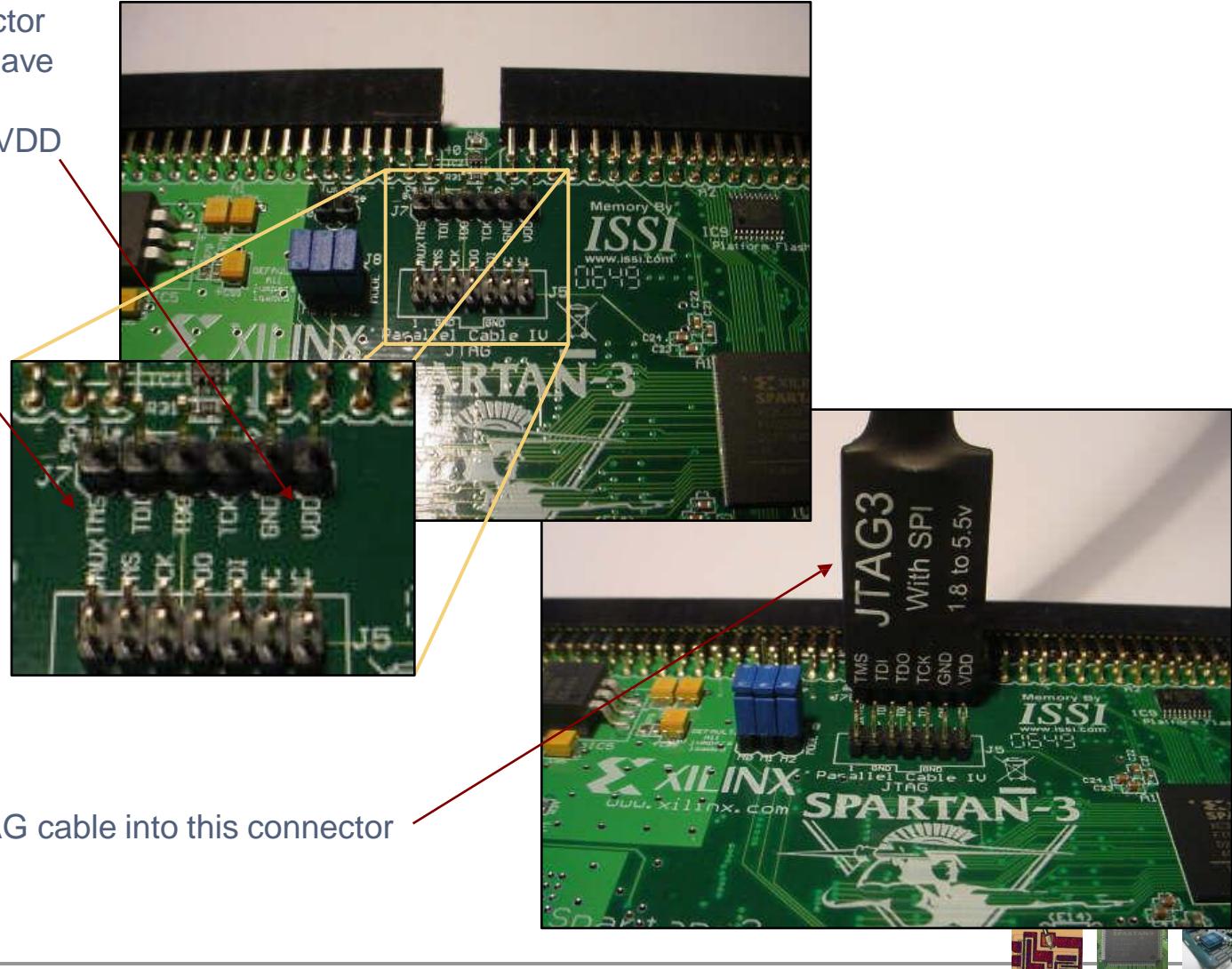


Download and Execute

Locate the JTAG connector
on the Spartan3. It will have

on one end and

TMS
on the other



Download and Execute

Plug the other end into the Parallel Port on the back of your computer or the USB port if you are using a USB JTAG

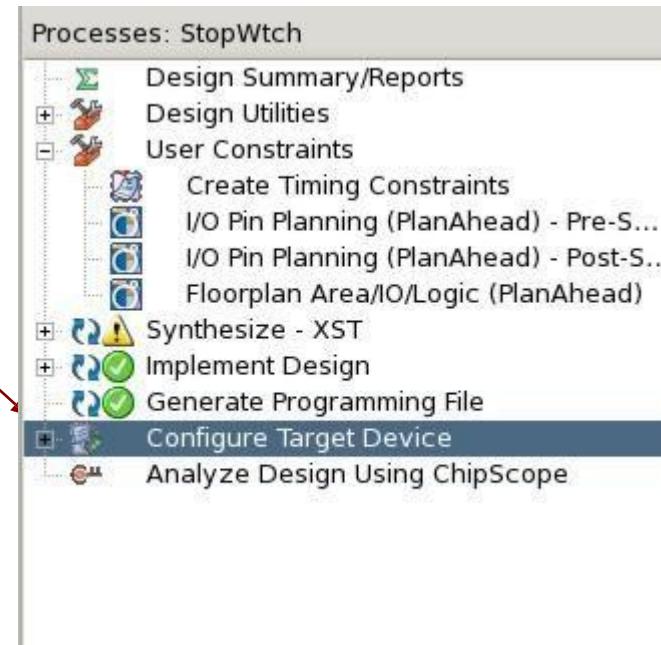


To download your design, go back to the ISE



Download and Execute

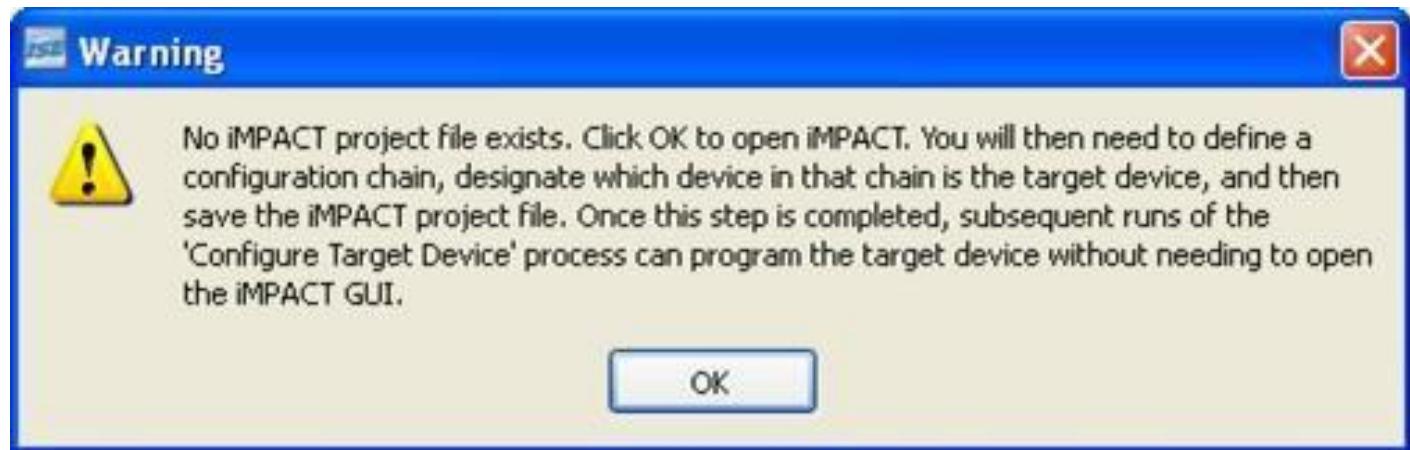
In the ISE double click on Configure Target Device – you will be invoking a program called “iMPACT”



Download and Execute

Unless you have saved from a previous session, you'll get a warning indicating that no iMPACT project file exists.

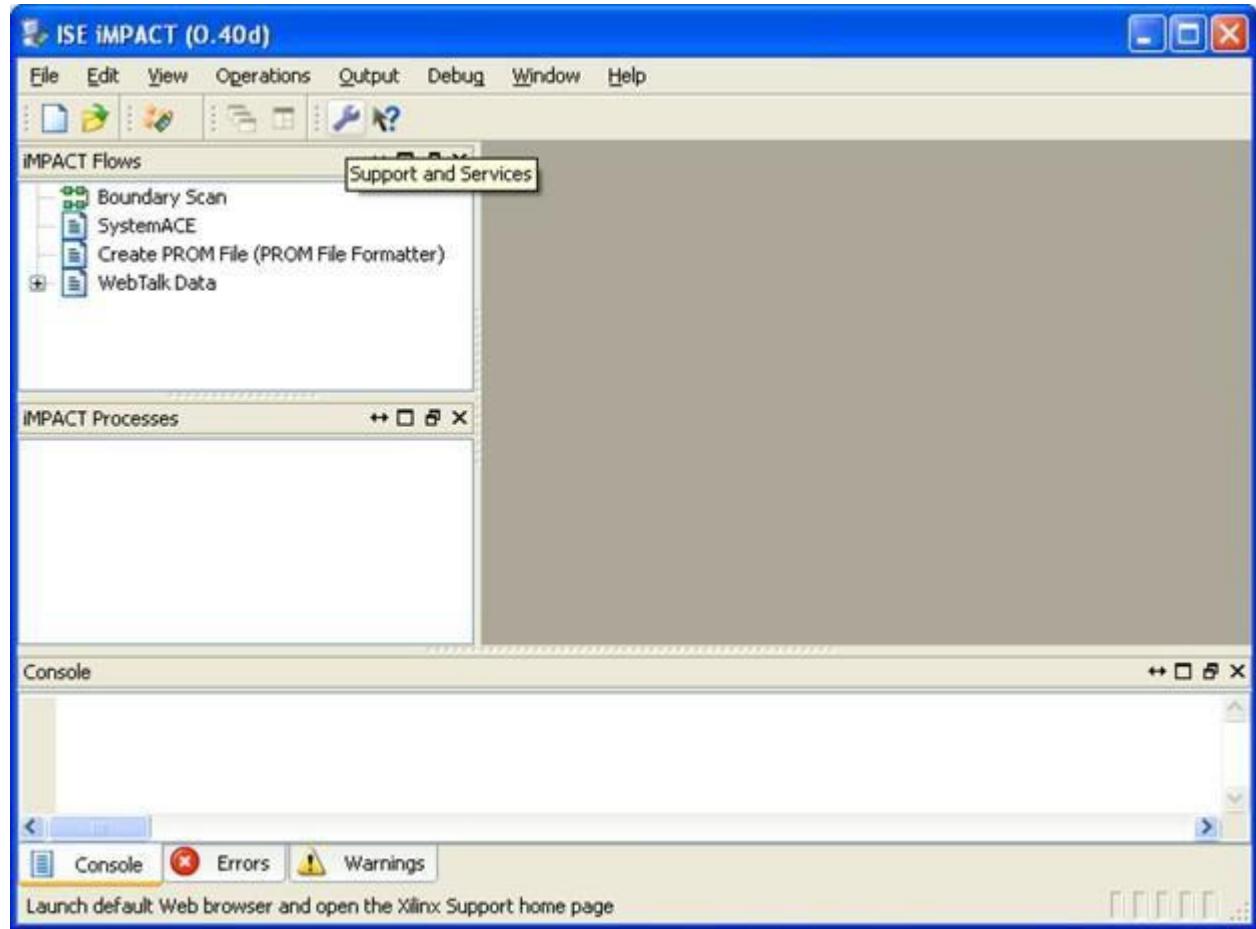
Just click OK



Download and Execute

iMPACT will open.

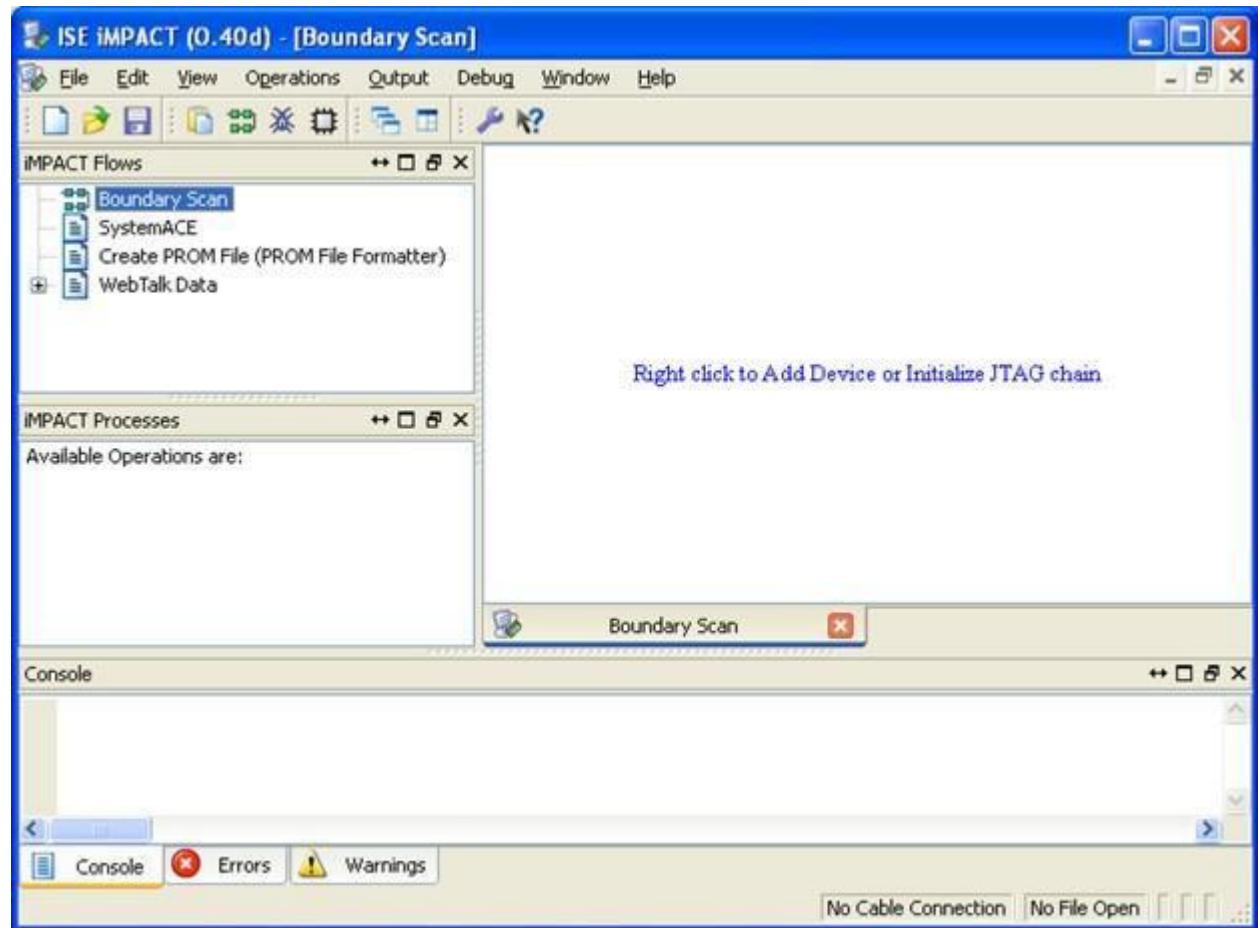
Double Click on Boundary Scan



Download and Execute

iMPACT will open.

Double Click on Boundary Scan and then right click on the new window

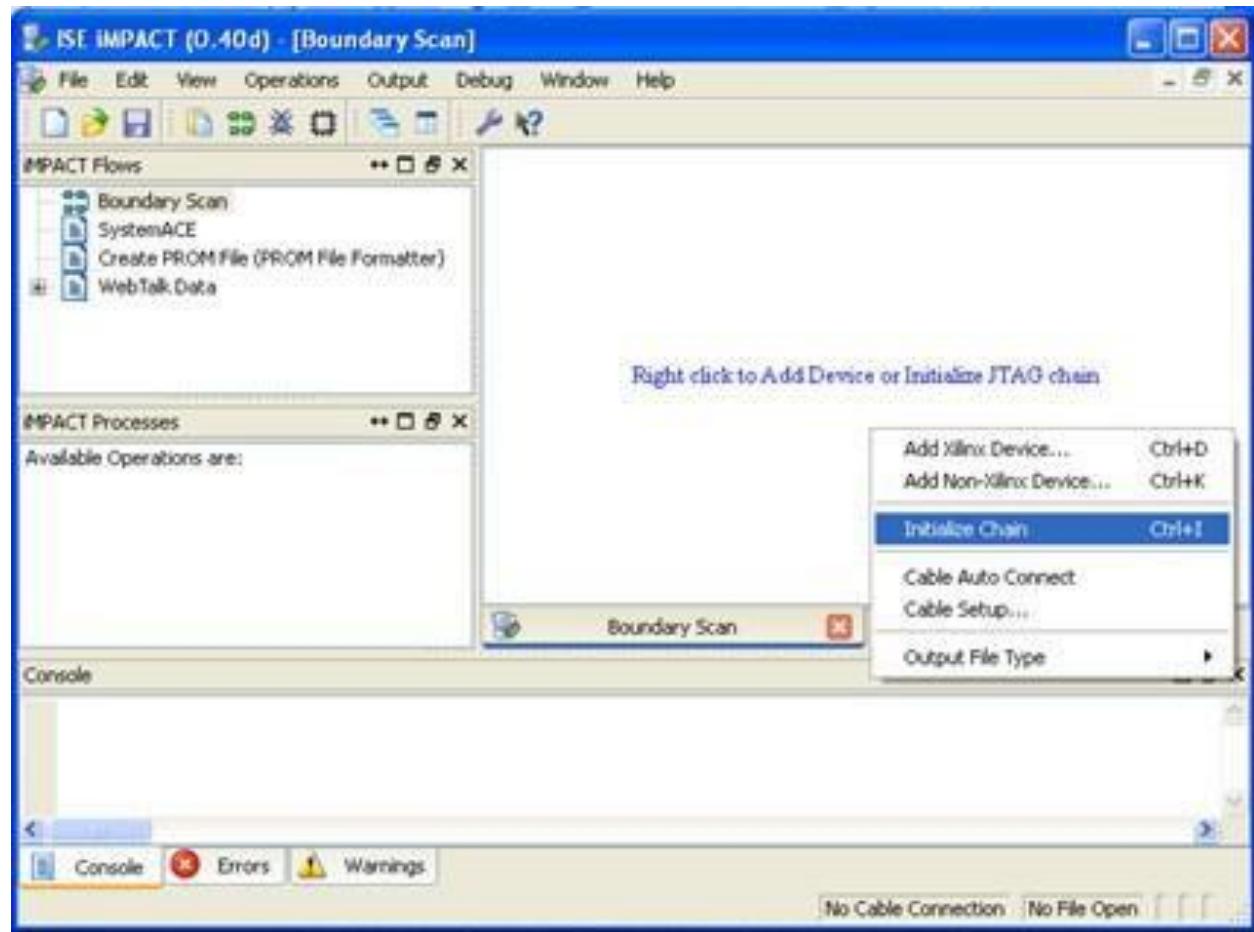


Download and Execute

iMPACT will open.

Double Click on Boundary Scan and then right click on the new window and click on Initialize Chain

NOTE: this is only one way to proceed.

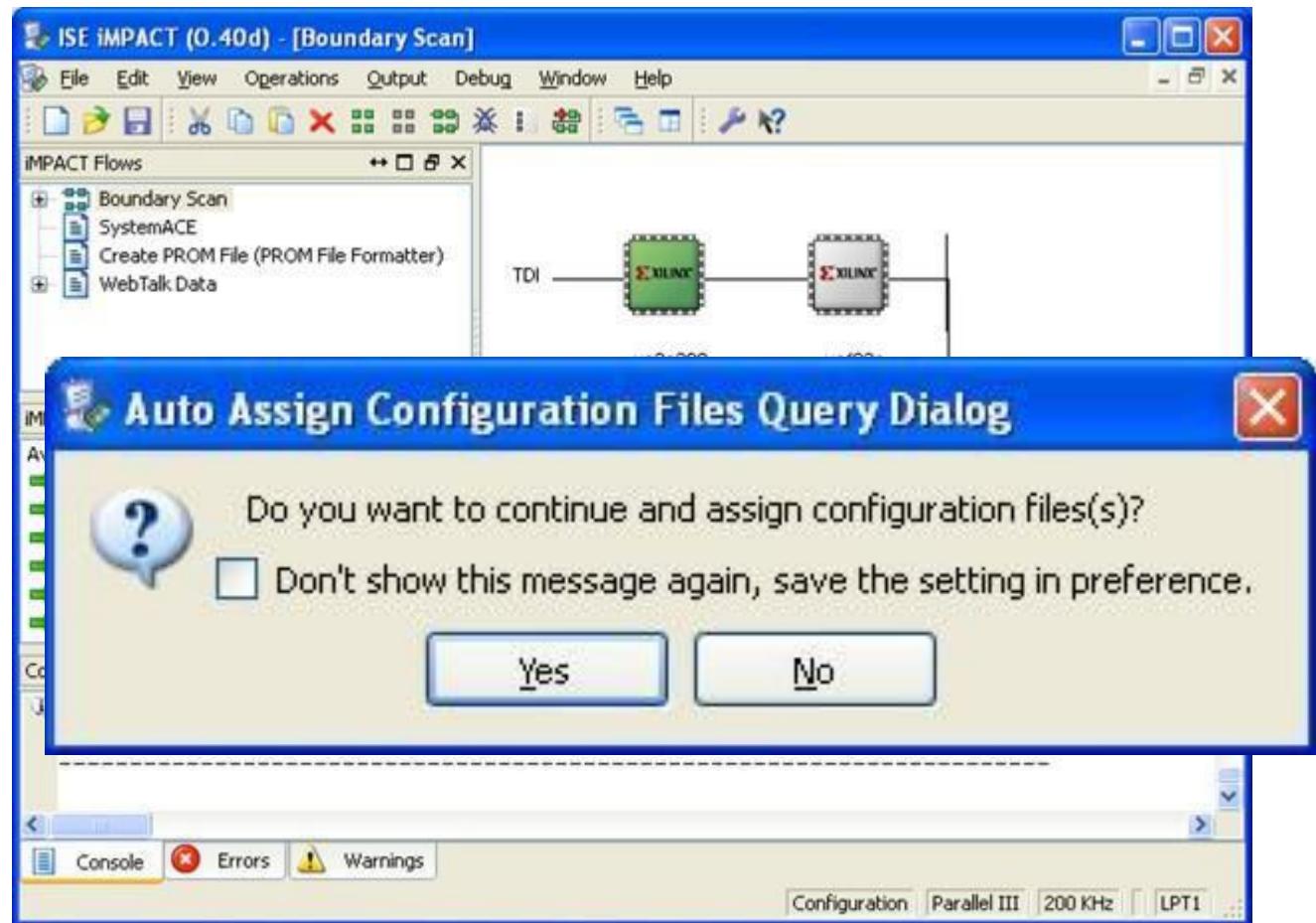


Download and Execute

iMPACT will open.

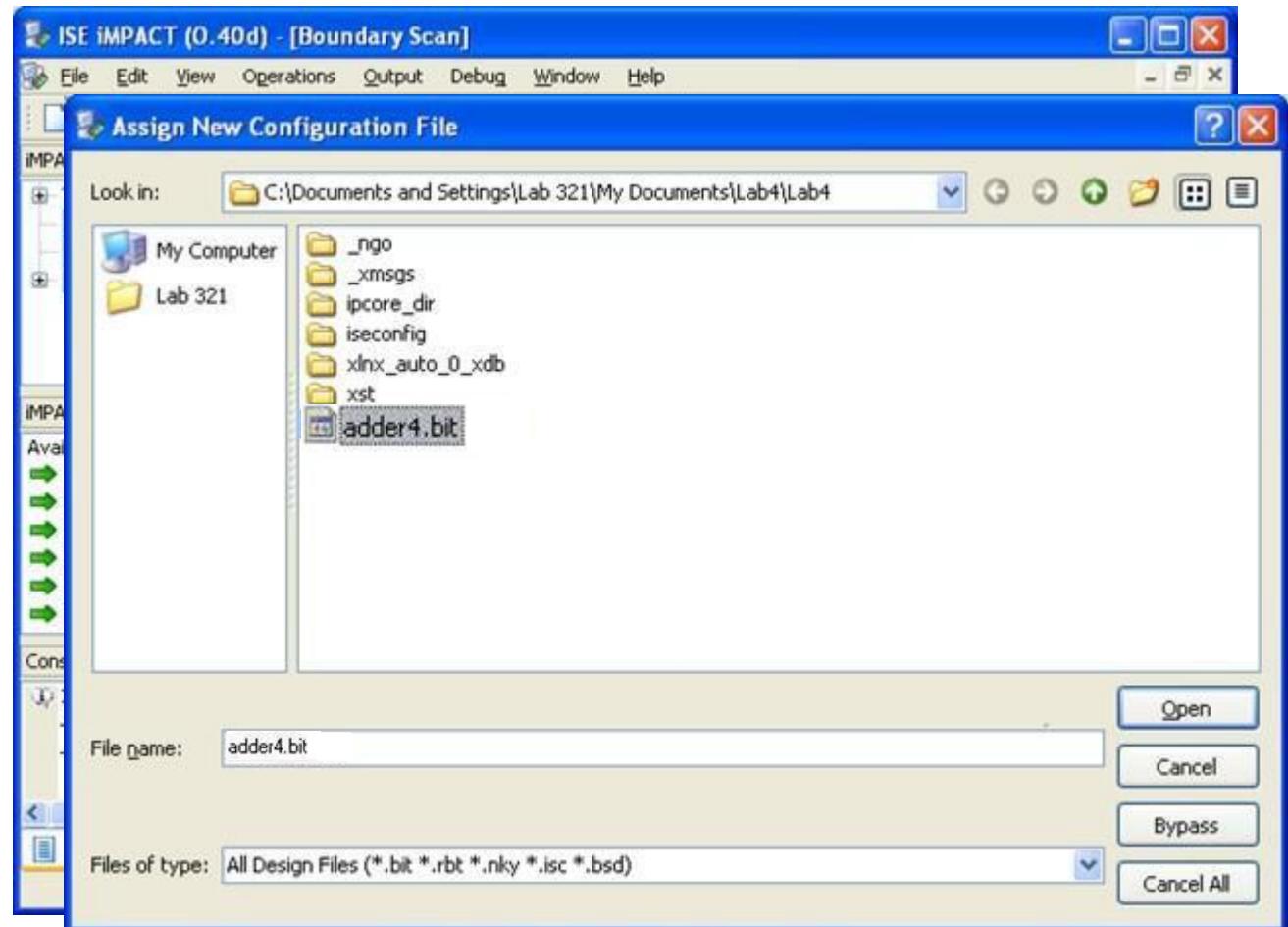
Double Click on Boundary Scan and then right click on the new window and click on Initialize Chain

iMPACT will ask if you want to assign configuration files. Click on yes



Download and Execute

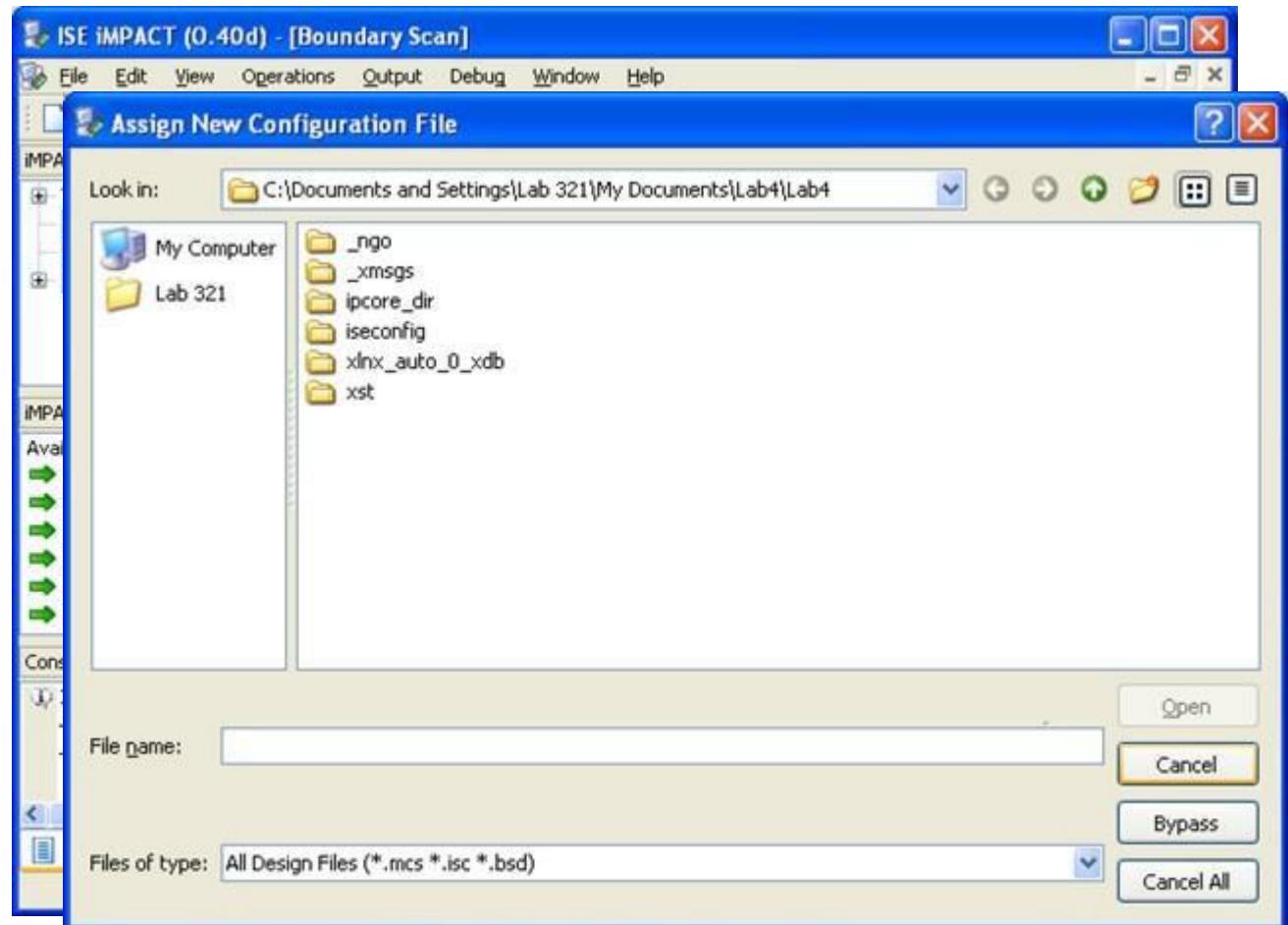
Click on the appropriate *.bit file and then click open



Download and Execute

Click on the appropriate *.bit file and then click open

On the next, click either cancel or bypass

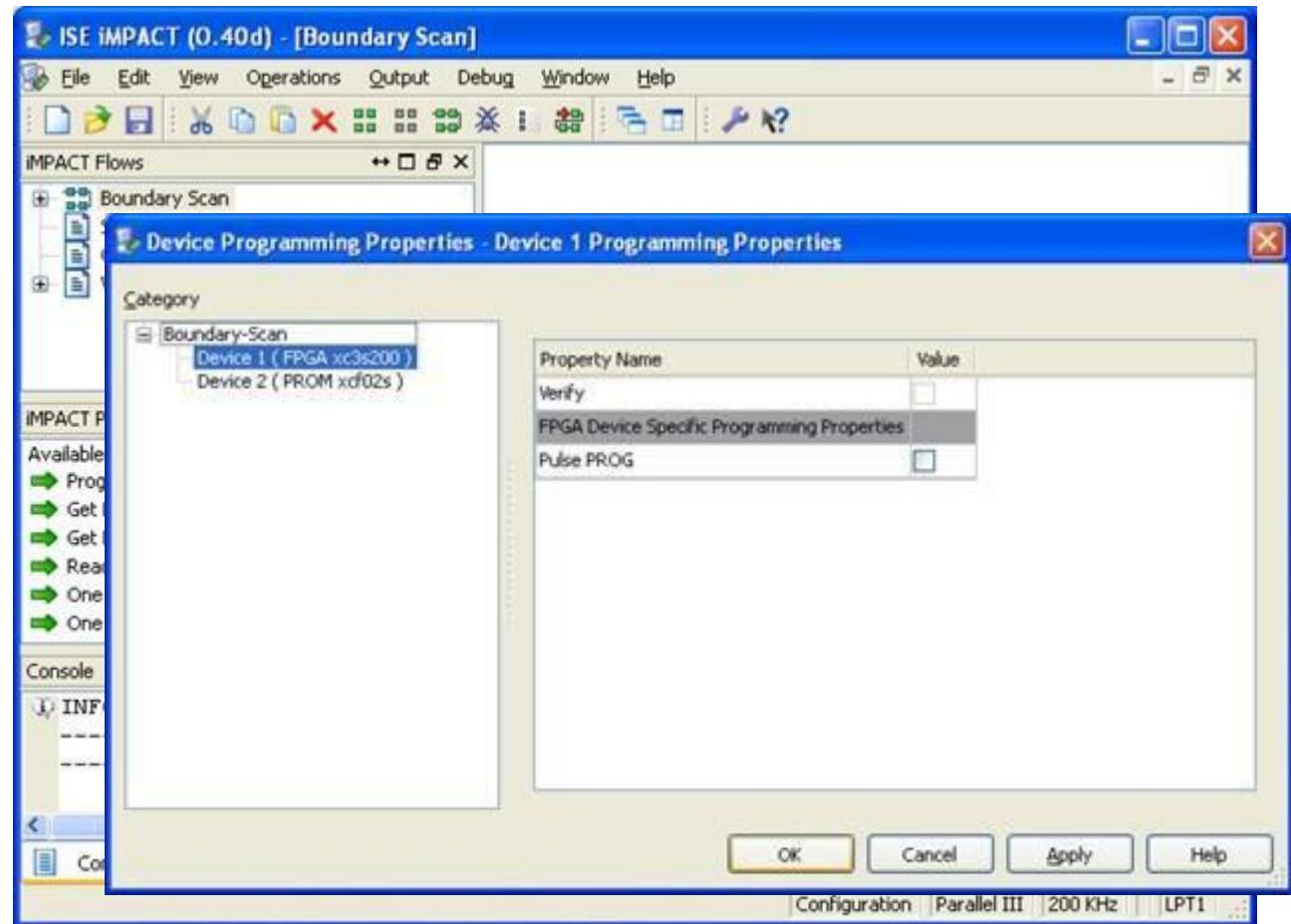


Download and Execute

Click on the appropriate *.bit file and then click open

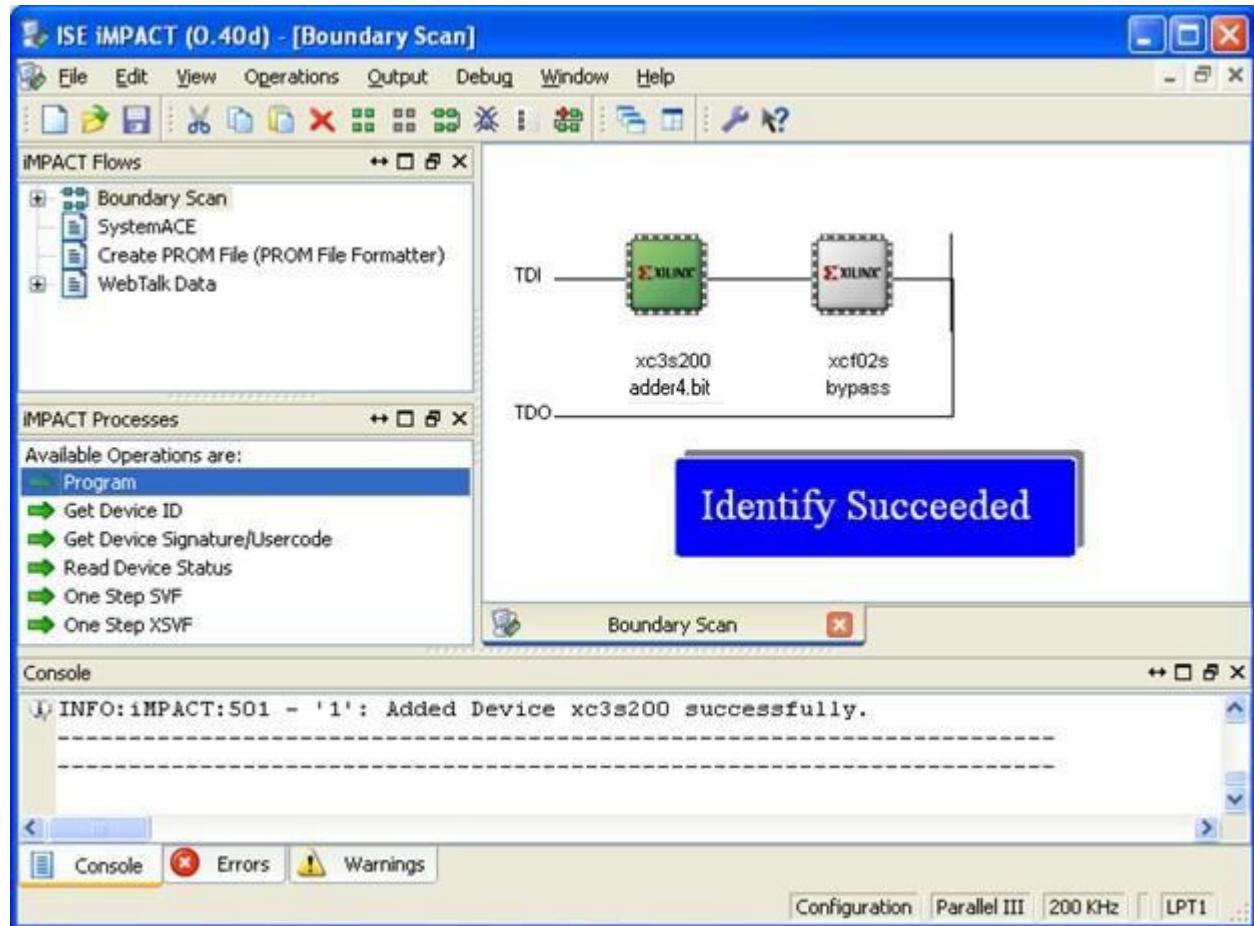
On the next, click either cancel or bypass

The next screen verifies which IC on the Spartan 3 board will be programmed. In this case, just the FPGA will be programmed. Click on OK



Download and Execute

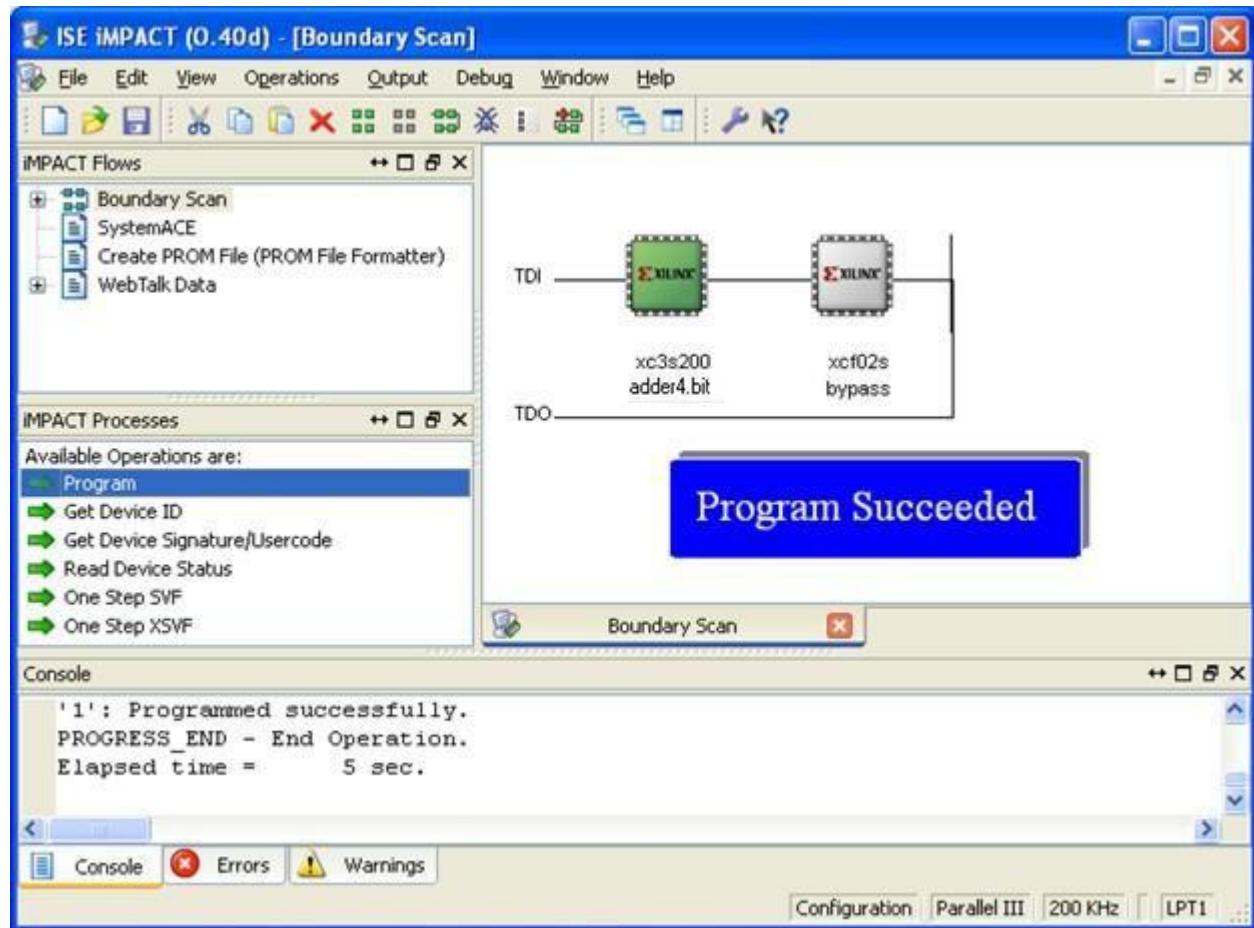
Double Click on Program



Download and Execute

Double Click on Program

..... and hopefully, you will
see "Program Succeeded"



Download and Execute

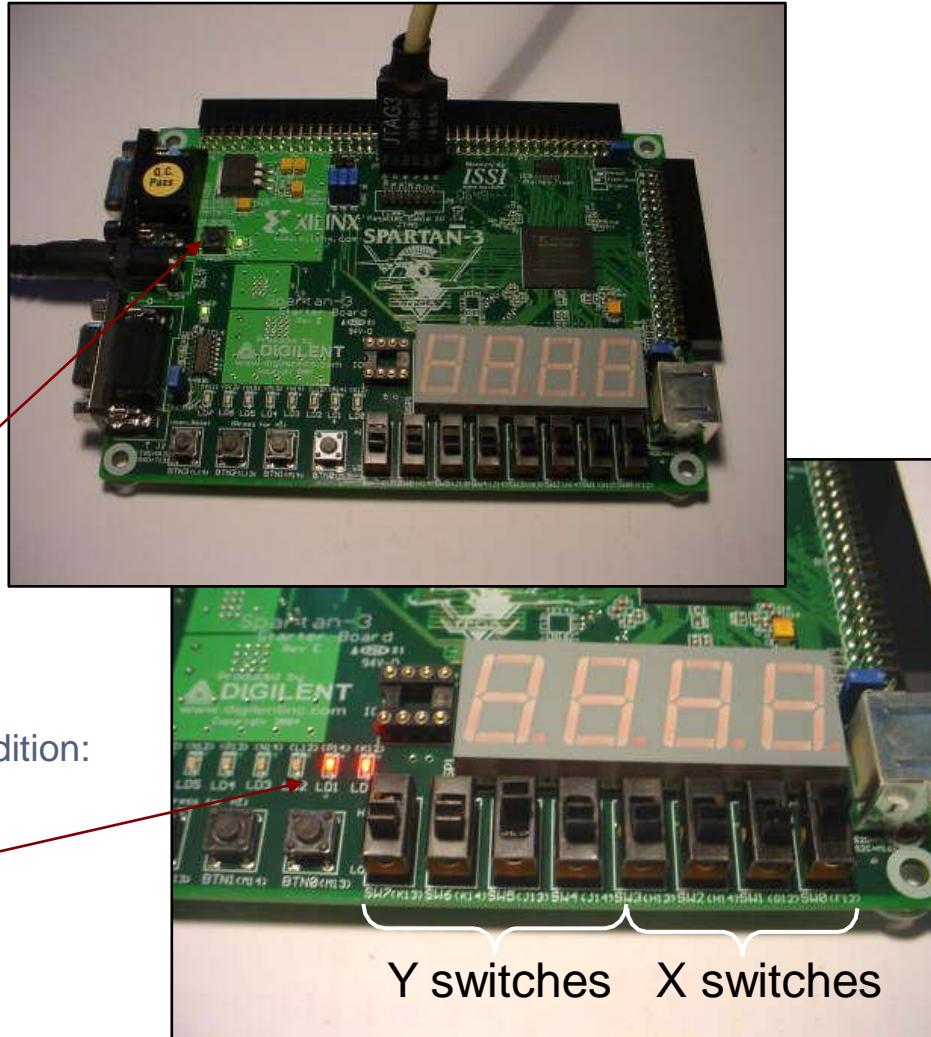
Once the file is downloaded to the Spartan3, the blinking 7-segment LEDs should turn off.

It is time to test the design

(Note: the design will continuously execute, until you hit the RESET button or you turn off the power)

I first tested a simple addition:
 $1+2$ ($X=1$, $Y=2$)

The result was 3
(So far, so good)



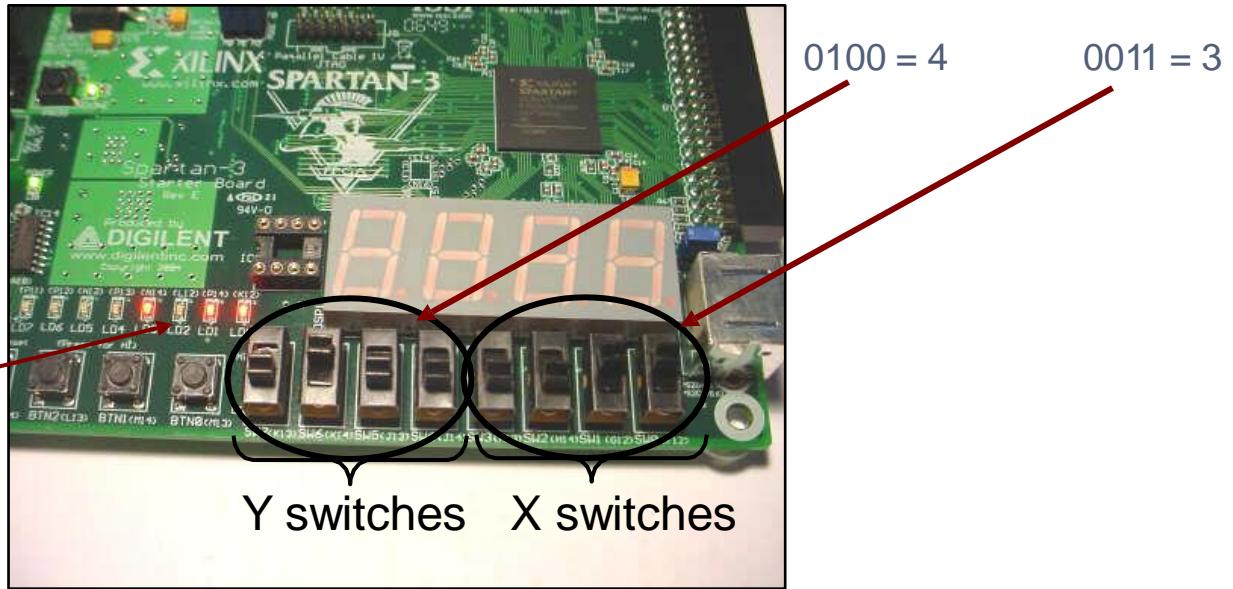
Download and Execute

I then tested:
 $3+4$ ($X=3$, $Y=4$)

The result was 11
(Not so good)

I tested a few other numbers,
and sometimes got the
correct result, sometimes I
got an incorrect result.

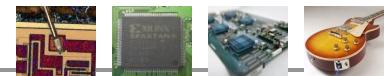
hmmmmm.....



Troubleshooting

The problem had to be somewhere in the design or in the *.ucf file. If in the design, it could be in the adder4 design or the fulladd design (the sub-circuit)

hmmmm....



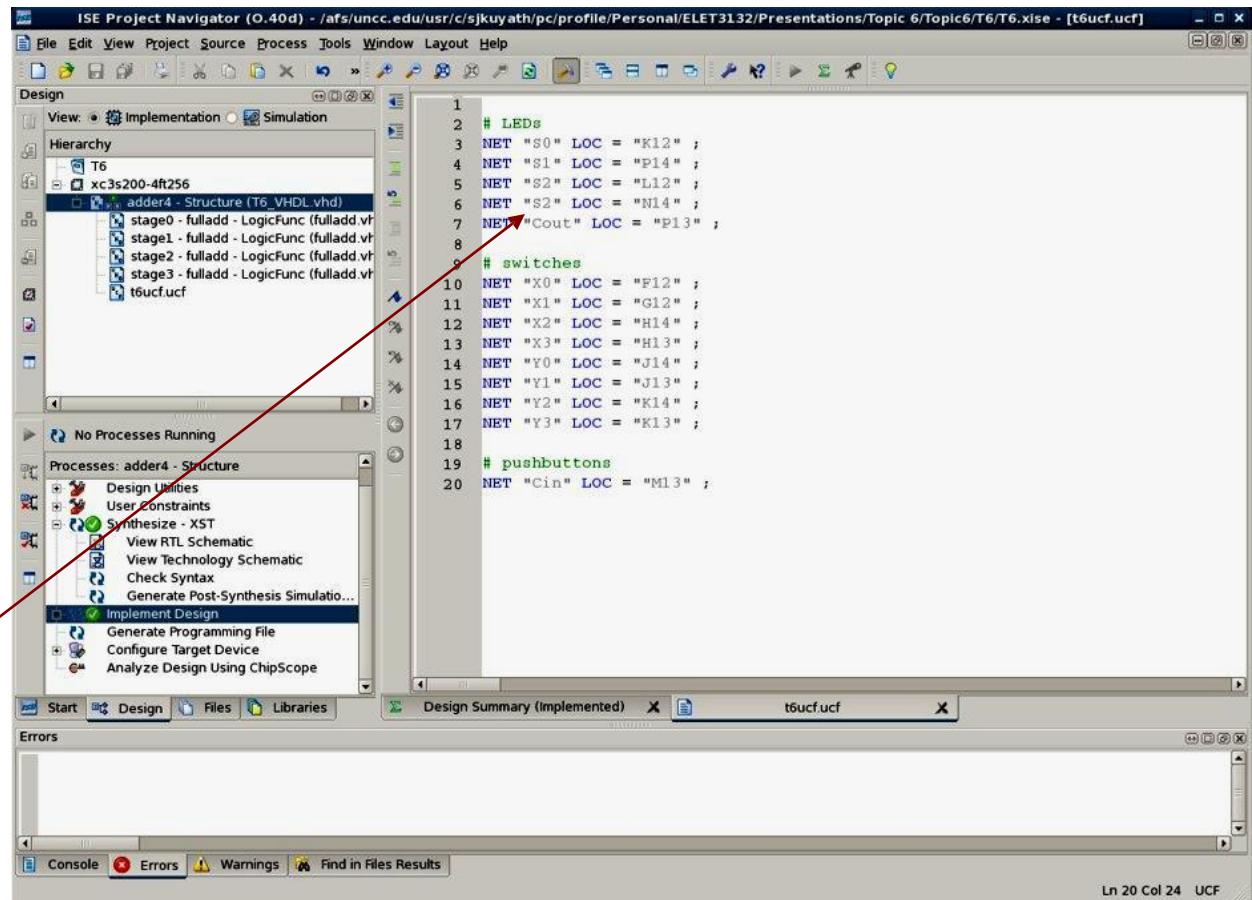
Troubleshooting

The problem had to be somewhere in the design or in the *.ucf file. If in the design, it could be in the adder4 design or the fulladd design (the sub-circuit)

hmmmm....

So, I opened the ISE and staring me in the face was the UCF!!

I noticed that I had unintentionally, assigned S2 (sum, bit 2) to both LED2 and LED3, but I left out S3 (sum bit 3, the MSb of the result.



```
ISE Project Navigator (O.40d) - /afs/uncc.edu/usr/c/sjkuyath/pc/profile/Personal/ELET3132/Presentations/Topic 6/Topic6/T6/T6.xise - [t6.ucf.ucf]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
T6
  xc3s200-4ft256
    adder4 - Structure (T6_VHDL.vhd)
      stage0 - fulladd - LogicFunc (fulladd.vhd)
      stage1 - fulladd - LogicFunc (fulladd.vhd)
      stage2 - fulladd - LogicFunc (fulladd.vhd)
      stage3 - fulladd - LogicFunc (fulladd.vhd)
    t6.ucf.ucf
No Processes Running
Processes: adder4 - Structure
  Design Utilities
  User Constraints
  Synthesize - XST
    View RTL Schematic
    View Technology Schematic
    Check Syntax
    Generate Post-Synthesis Simulation...
  Implement Design
    Generate Programming File
    Configure Target Device
    Analyze Design Using ChipScope
Start Design Files Libraries
Design Summary (Implemented) t6.ucf
Errors
Console Errors Warnings Find in Files Results
Ln 20 Col 24 UCF
```

1
2 # LEDs
3 NET "S0" LOC = "K12" ;
4 NET "S1" LOC = "P14" ;
5 NET "S2" LOC = "L12" ;
6 NET "S2" LOC = "N14" ;
7 NET "Cout" LOC = "P13" ;
8
9 # switches
10 NET "X0" LOC = "F12" ;
11 NET "X1" LOC = "G12" ;
12 NET "X2" LOC = "H14" ;
13 NET "X3" LOC = "H13" ;
14 NET "Y0" LOC = "J14" ;
15 NET "Y1" LOC = "J13" ;
16 NET "Y2" LOC = "K14" ;
17 NET "Y3" LOC = "K13" ;
18
19 # pushbuttons
20 NET "Cin" LOC = "M13" ;



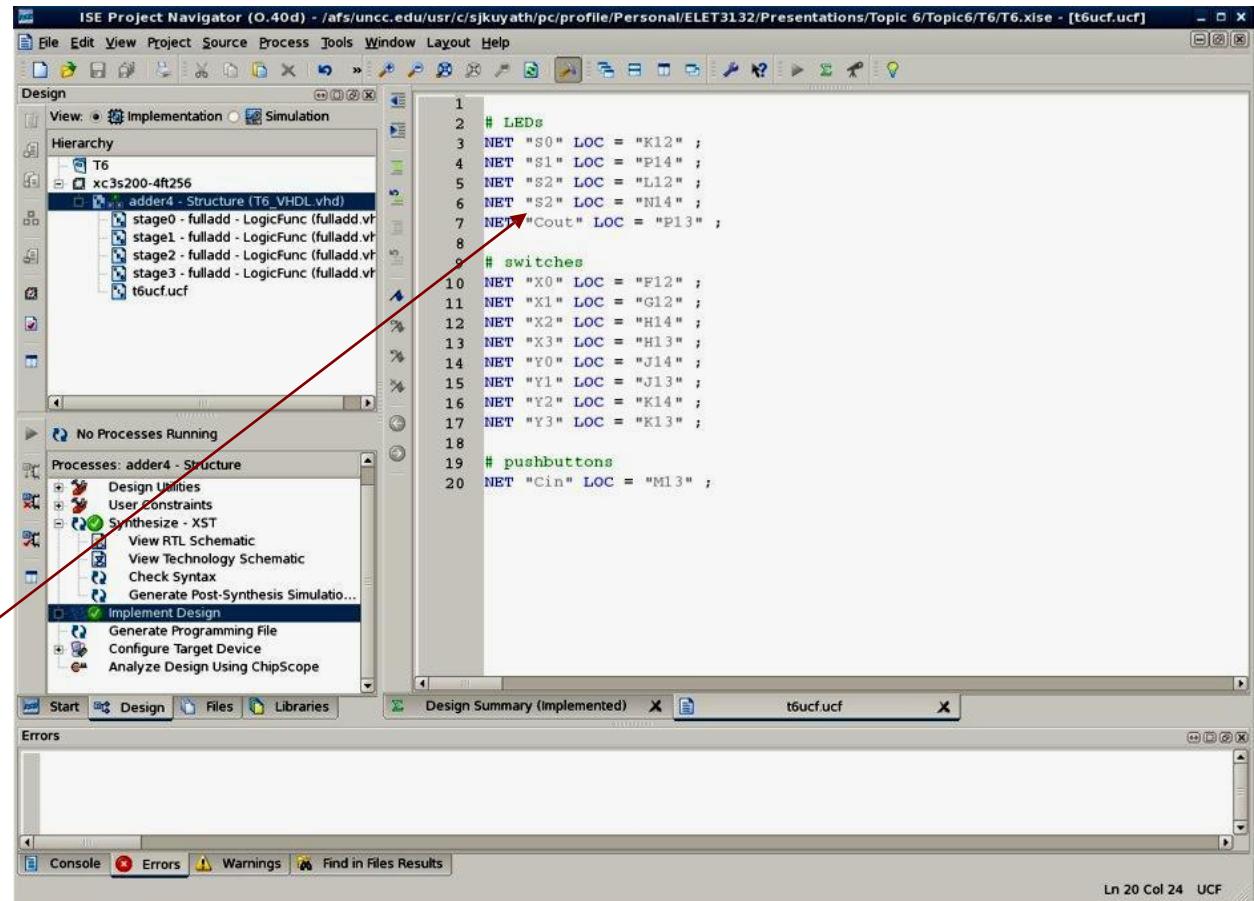
Troubleshooting

The problem had to be somewhere in the design or in the *.ucf file. If in the design, it could be in the adder4 design or the fulladd design (the sub-circuit)

hmmmm....

So, I opened the ISE and staring me in the face was the UCF!!

I noticed that I had unintentionally, assigned S2 (sum, bit 2) to both LED2 and LED3, but I left out S3 (sum bit 3, the MSb of the result.



A screenshot of the ISE Project Navigator interface. The main window shows the 'Design' view with the 'Implementation' tab selected. In the 'Hierarchy' pane, there is a tree structure for a project named 'T6' containing an 'xc3s200-4ft256' device and a 'adder4 - Structure (T6_VHDL.vhd)' component. The 'adder4' component has four sub-components: 'stage0 - fulladd', 'stage1 - fulladd', 'stage2 - fulladd', and 'stage3 - fulladd', all of which are 'LogicFunc' type components. Below the component tree is a file named 't6ucf.ucf'. The right-hand panel displays the contents of this UCF file. A red arrow points from the text 'I'll admit.... I got lucky..... it happens..... but don't count on it!!' at the bottom of the slide to the line 'NET "S2" LOC = "N14";' in the UCF file, highlighting a syntax error where 'NET "S2"' is missing a closing quote. The code in the UCF file is as follows:

```
1 # LEDs
2 NET "S0" LOC = "K12";
3 NET "S1" LOC = "P14";
4 NET "S2" LOC = "L12";
5 NET "S2" LOC = "N14";
6 NET "Cout" LOC = "P13";
7 # switches
8 NET "X0" LOC = "F12";
9 NET "X1" LOC = "G12";
10 NET "X2" LOC = "H14";
11 NET "X3" LOC = "H13";
12 NET "Y0" LOC = "J14";
13 NET "Y1" LOC = "J13";
14 NET "Y2" LOC = "K14";
15 NET "Y3" LOC = "K13";
16
17 # pushbuttons
18 NET "Cin" LOC = "M13";
19
20
```

I'll admit.... I got lucky..... it happens..... but don't count on it!!

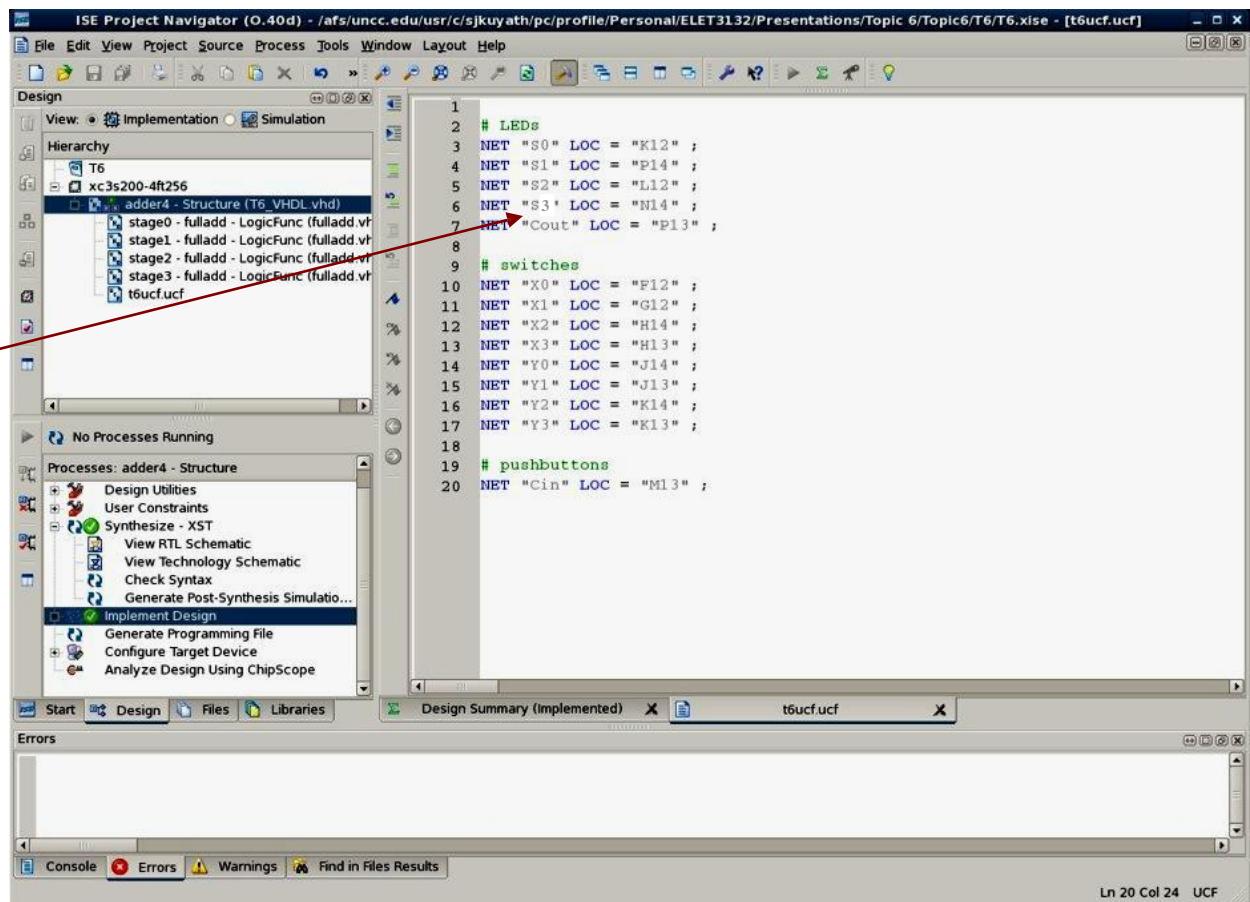


Troubleshooting

So I fixed the problem and saved everything

Then I re-implemented the design, I re-generated the programming file, I re-configured the target device

And then I re-downloaded the program



The screenshot shows the Xilinx ISE Project Navigator interface. The 'Design' tab is selected, and the 'Implementation' view is active. In the 'Hierarchy' pane, the project structure is shown with 'T6' as the top-level component, containing 'xc3s200-4ft256' which further contains 'adder4 - Structure (T6_VHDL.vhd)' and several 'fulladd' logic functions. A red arrow points from the text 'I re-implemented the design, I re-generated the programming file, I re-configured the target device' to the 'adder4 - Structure' node in the hierarchy. The 'Processes' pane shows a list of synthesis and implementation steps, with 'Implement Design' highlighted. The 'Script Editor' pane displays a UCF (User Constraints File) script with code defining net locations for LEDs, switches, and pushbuttons. The 'Design Summary (Implemented)' and 't6ucf.ucf' tabs are visible at the bottom.

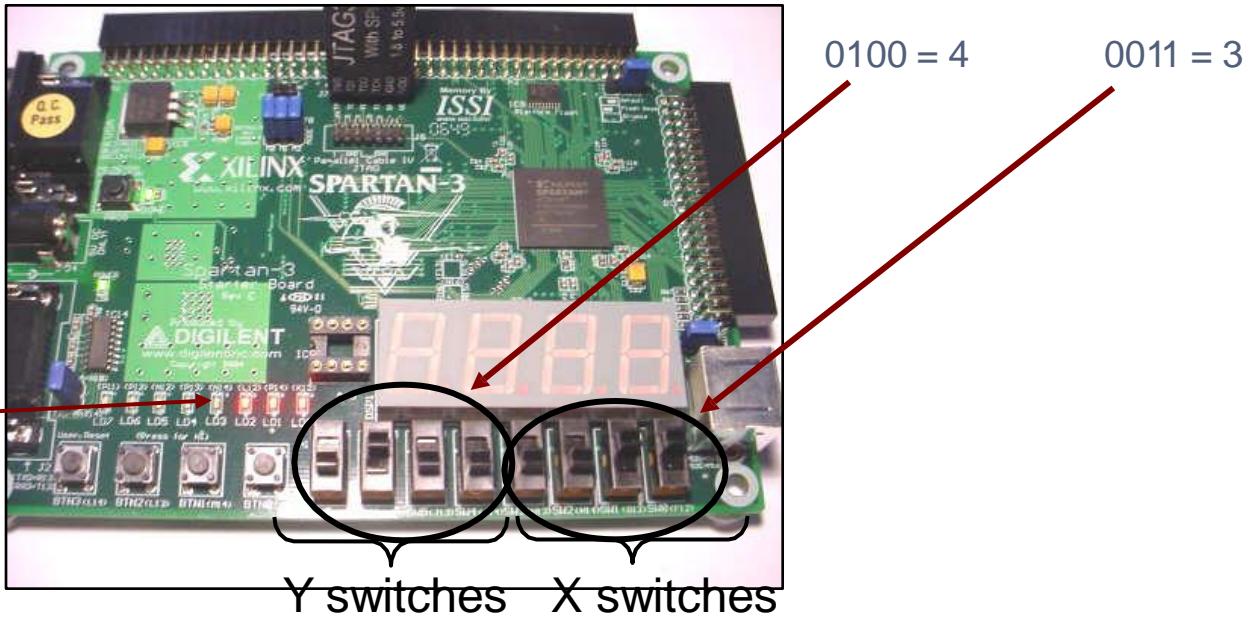
```
1 # LEDs
2 NET "S0" LOC = "K12" ;
3 NET "S1" LOC = "P14" ;
4 NET "S2" LOC = "L12" ;
5 NET "S3" LOC = "N14" ;
6 NET "Cout" LOC = "P13" ;
7
8 # switches
9 NET "X0" LOC = "F12" ;
10 NET "X1" LOC = "G12" ;
11 NET "X2" LOC = "H14" ;
12 NET "X3" LOC = "H13" ;
13 NET "Y0" LOC = "J14" ;
14 NET "Y1" LOC = "J13" ;
15 NET "Y2" LOC = "K14" ;
16 NET "Y3" LOC = "K13" ;
17
18 # pushbuttons
19 NET "Cin" LOC = "M13" ;
20
```



Download and Execute

I then tested:
 $3+4$ ($X=3$, $Y=4$) again

The result was 7
(ahhh....)

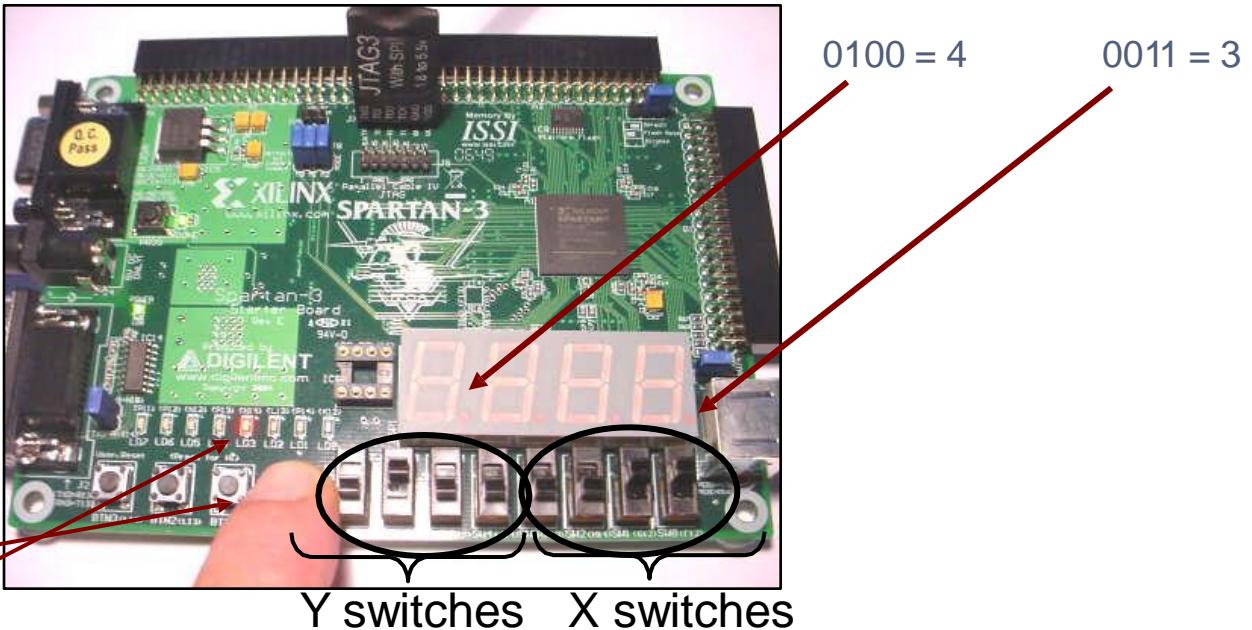


Download and Execute

I then tested:
 $3+4$ ($X=3$, $Y=4$) again

But, held down the Carry In
button (PB0)

The result was 8
(ahhh....)



Summary

- In this topic, we :
 - Wrote VHDL code
 - The main design
 - A component (a sub-design or sub-circuit)
 - A UCF file
 - Synthesized the VHDL designs
 - Troubleshot the design
 - Verified the results
 - Viewed:
 - Block Diagram
 - Symbol Diagram
 - Logic Diagram
 - Truth Table
 - K-Maps



Summary

- In this topic, we :
 - Simulated the design
 - Developed a testbench waveform
 - Simulated the testbench with ModelSim
 - Verified the results
 - Downloaded the design to the Spartan3 board
 - Verified the results
 - Troubleshoot the design
 - Downloaded the design to the Spartan3 board
 - Executed the design to verify and test

