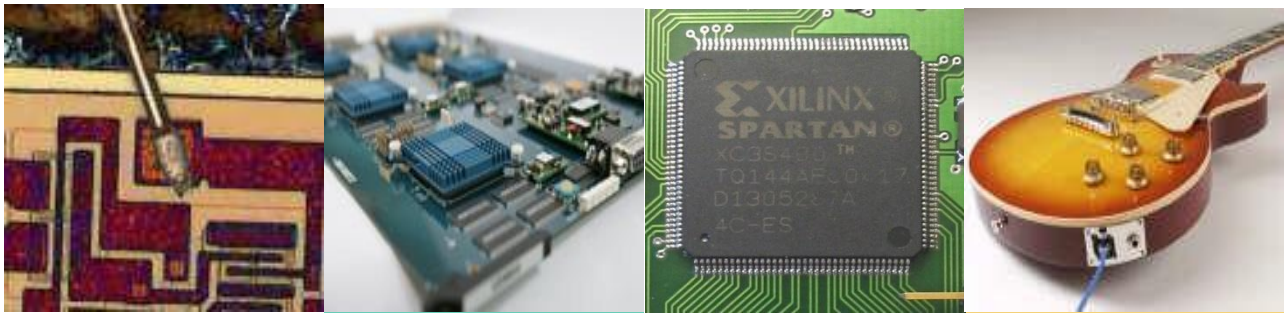
ELE 237
Fundamentals of Digital Systems



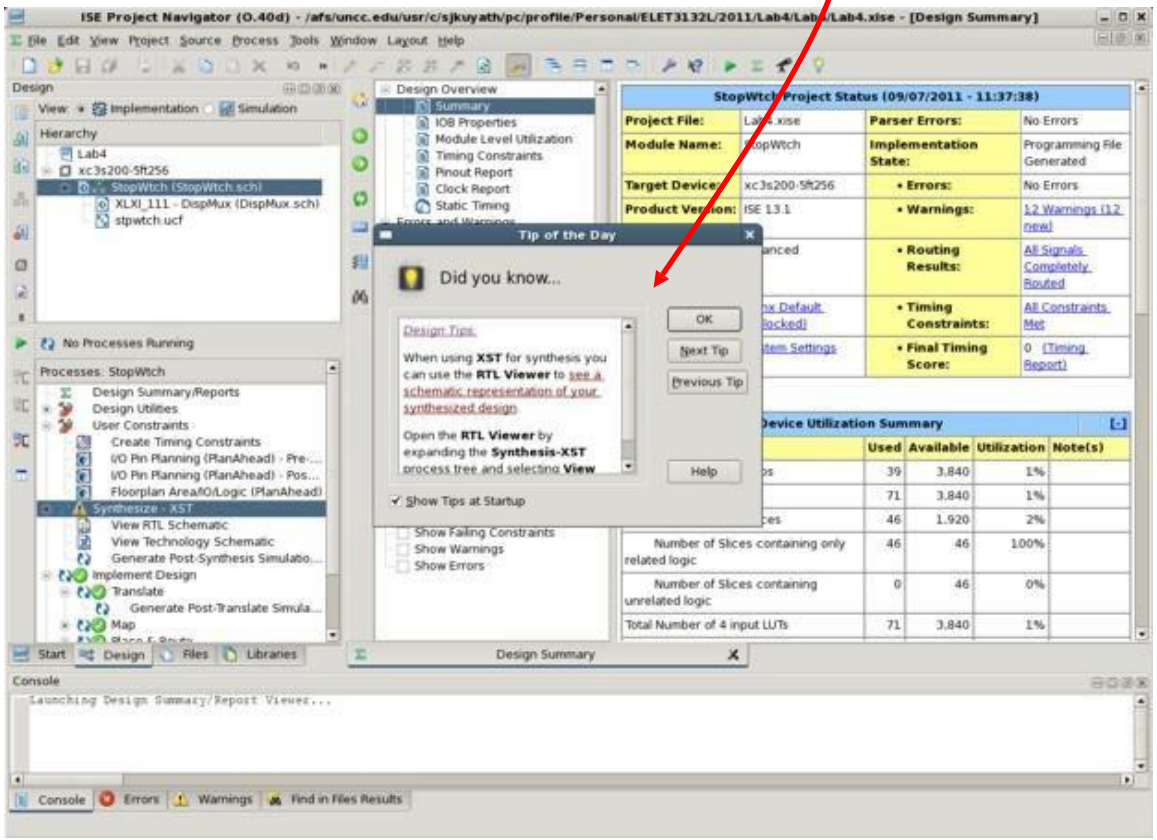
The Xilinx Integrated Software Environment

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

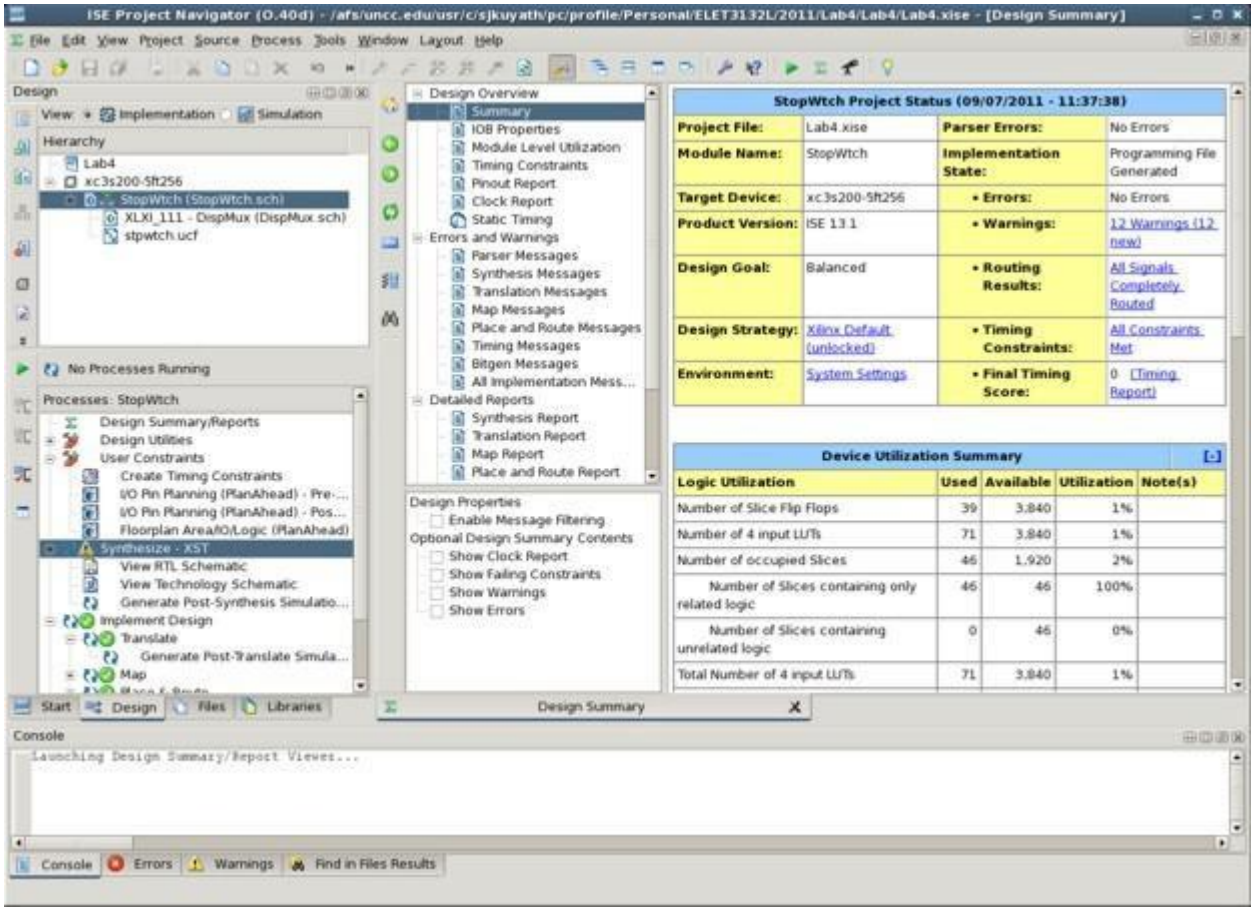
Xilinx Opening Screens

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



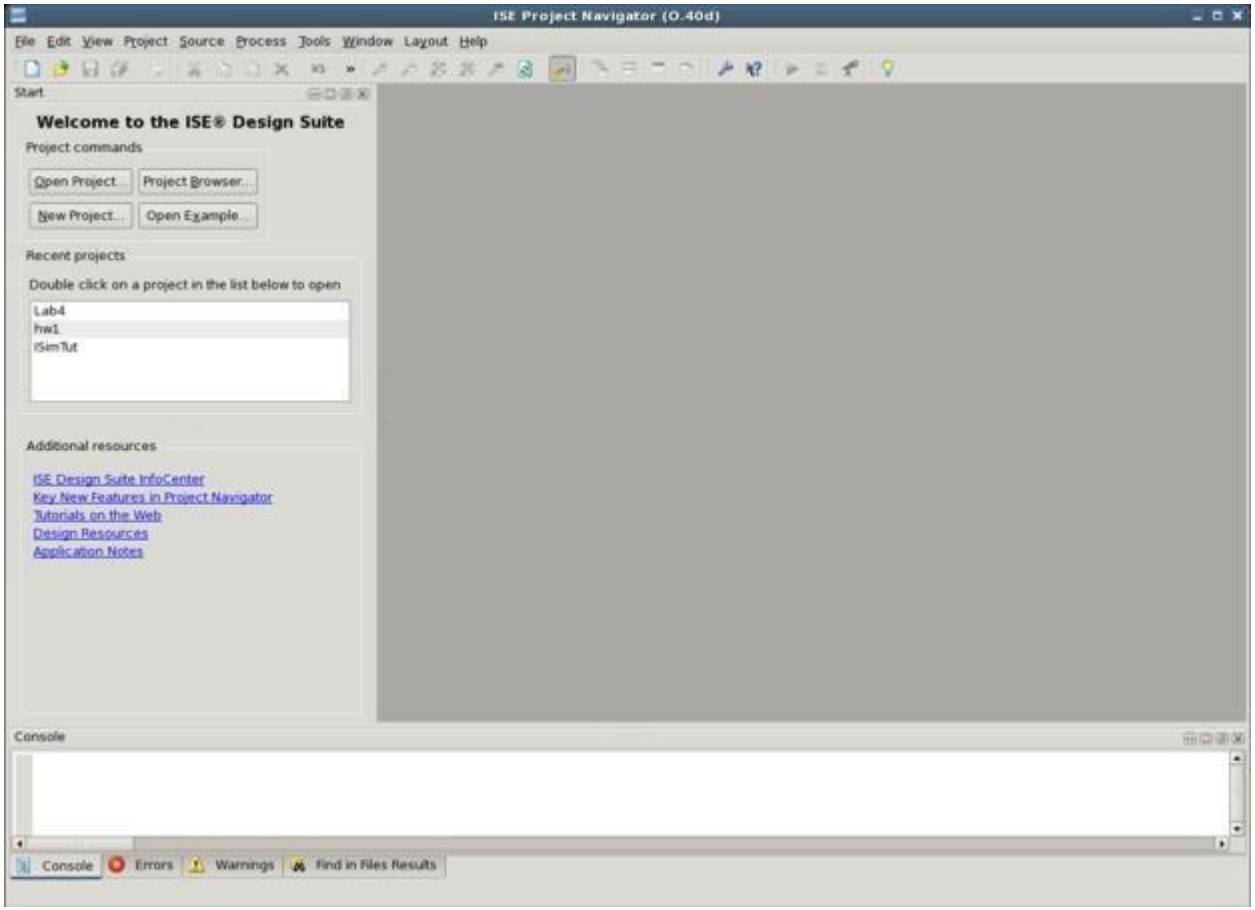
Xilinx Opening Screens

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



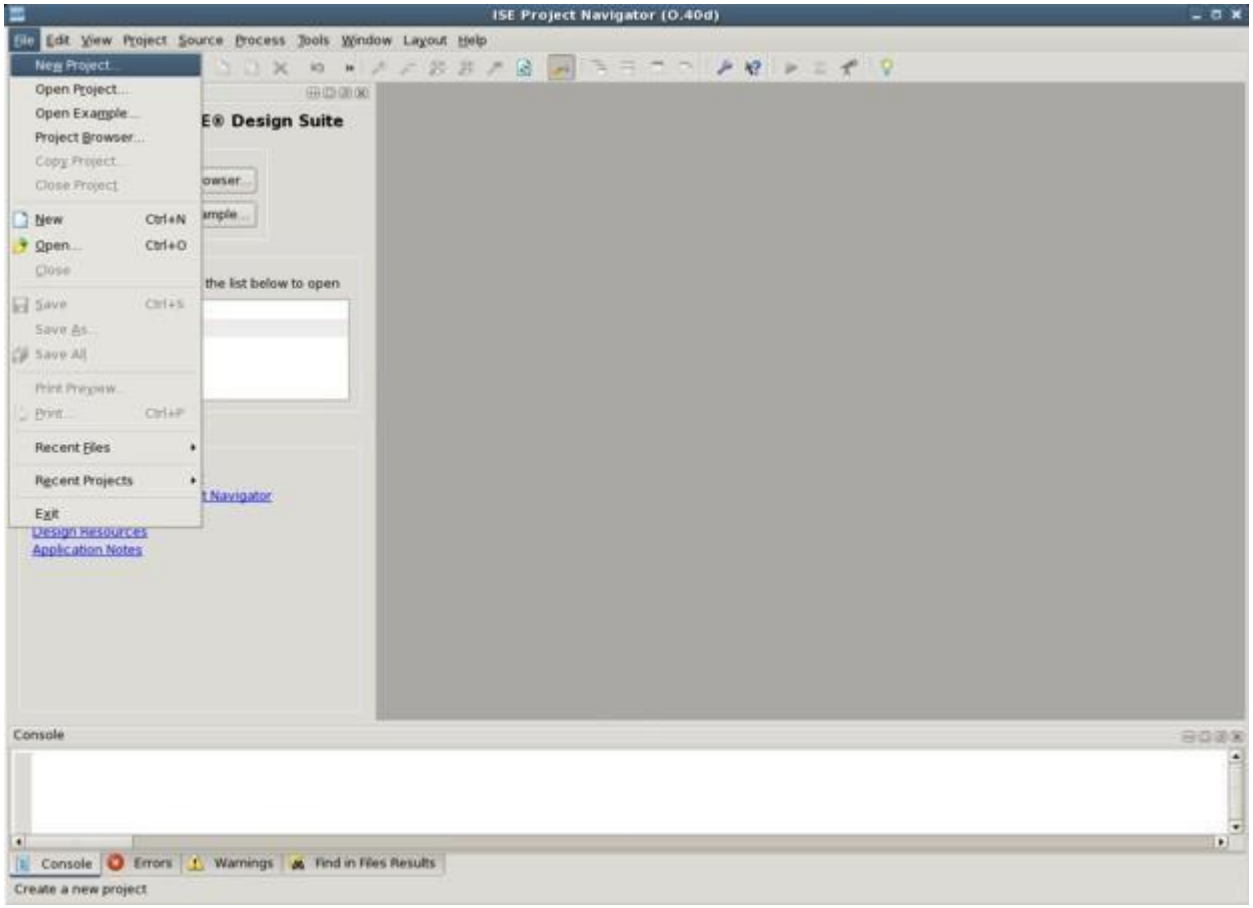
Xilinx Opening Screens

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



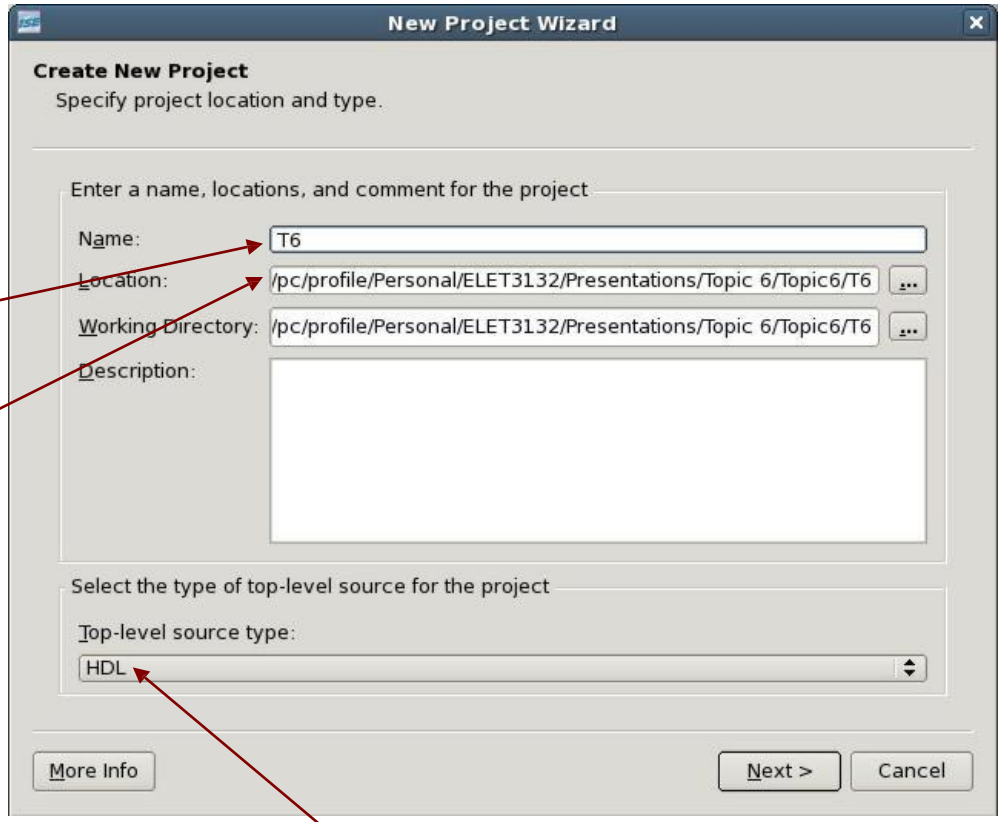
Xilinx Opening Screens

- 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).

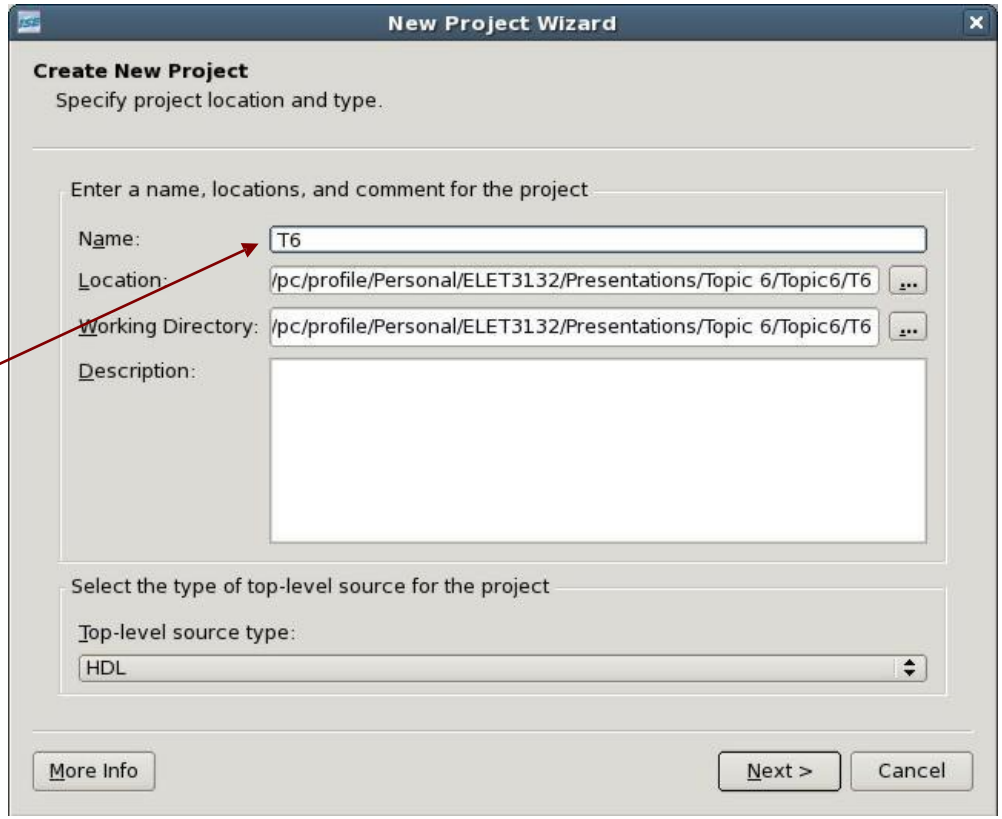
Make sure you have HDL as the Top-Level Module when you are writing VHDL code.




Xilinx Opening Screens

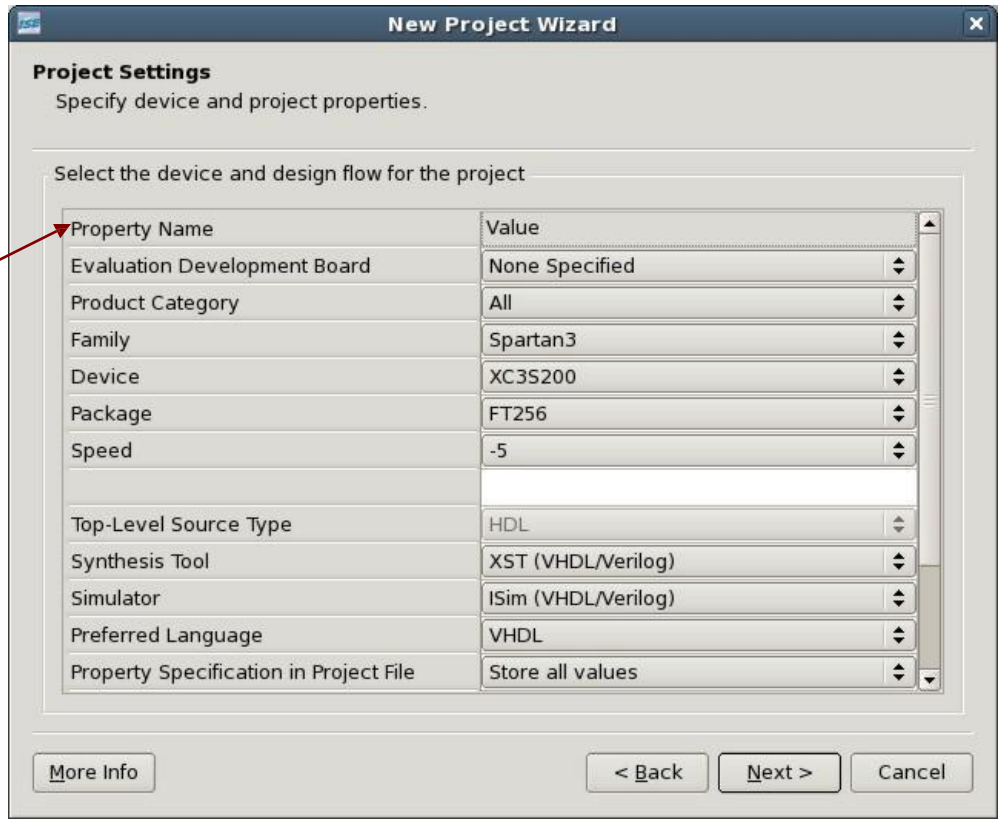
In version 13.1 the file name may be able to have special characters and be longer than 8 characters - IDK

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

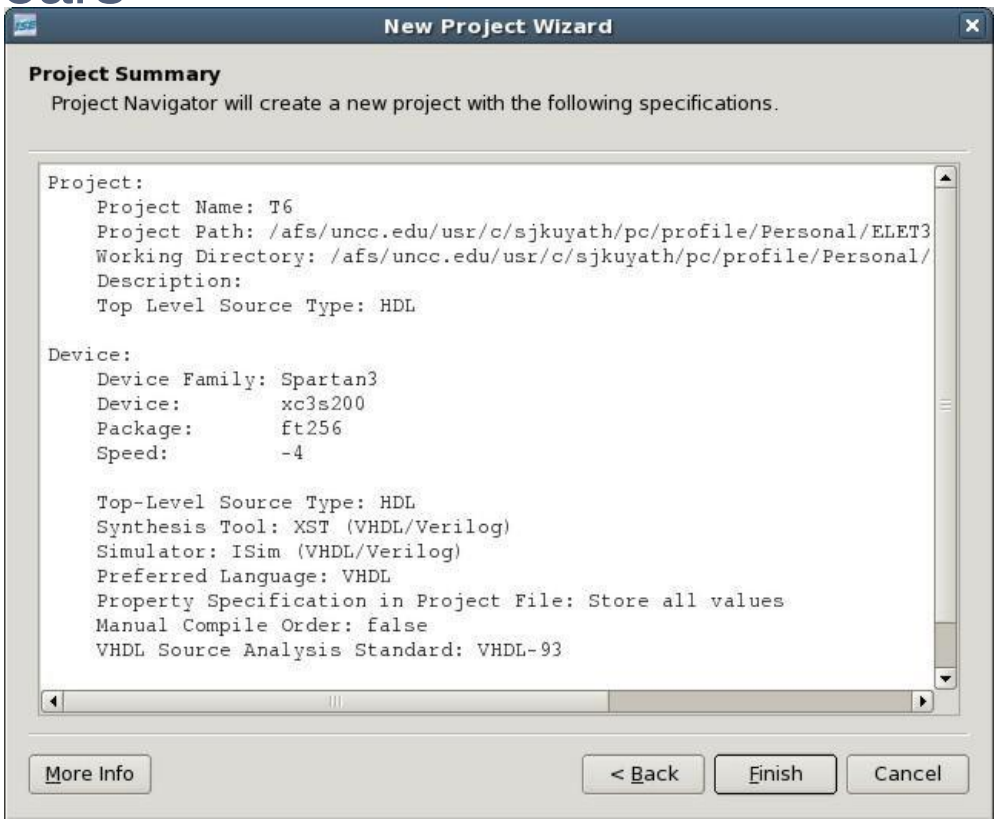


This is mainly a summary page, but make sure the information is the same as it appears here. It specifies the Spartan 3 boards that we will use for projects.

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

Xilinx Opening Screens

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

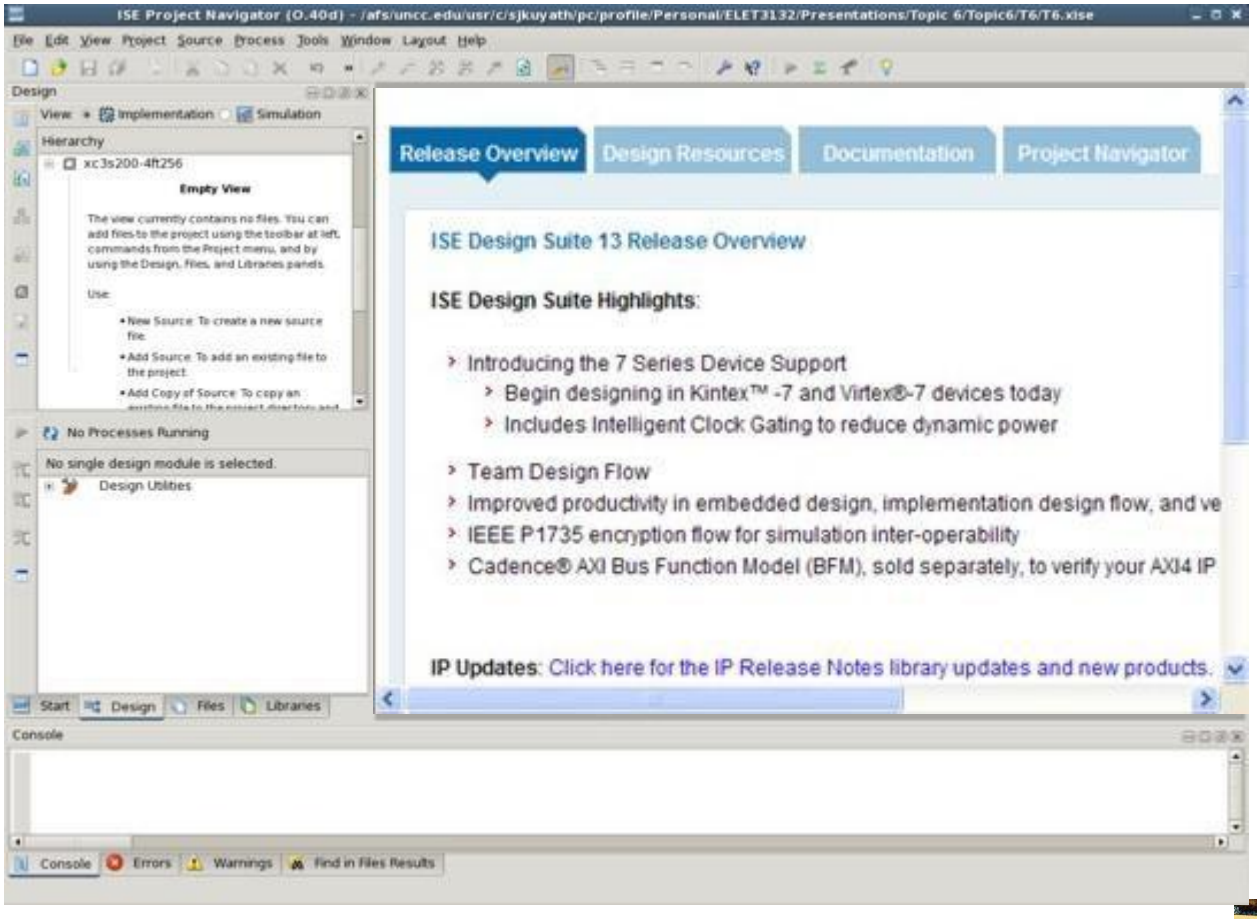


If something is incorrect, click on the back button and fix it.
When everything is as it should be, click [NEXT]



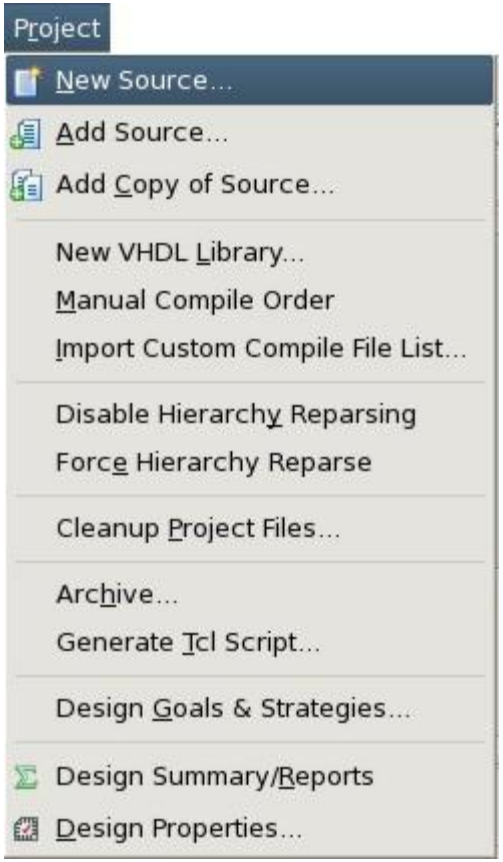

Xilinx Opening Screens

- 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

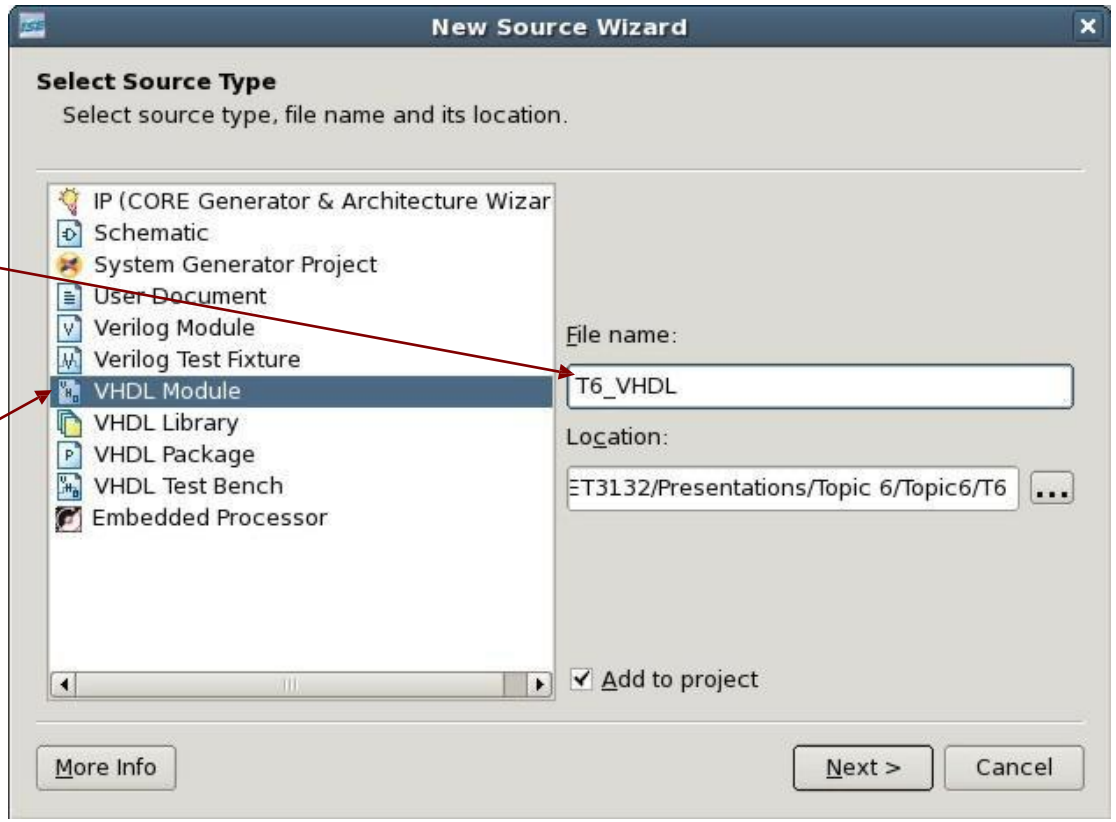


Xilinx Opening Screens

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

Give the source file a name (no Spaces)

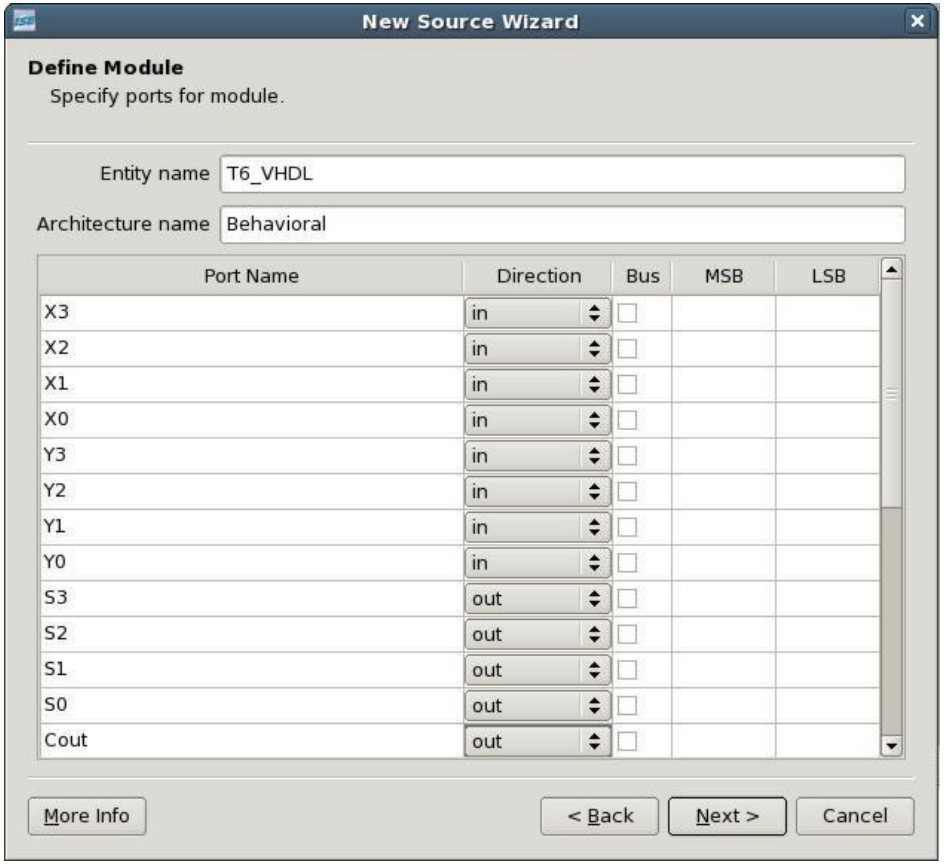
And highlight VHDL Module



Click the [Next] button

Xilinx Opening Screens

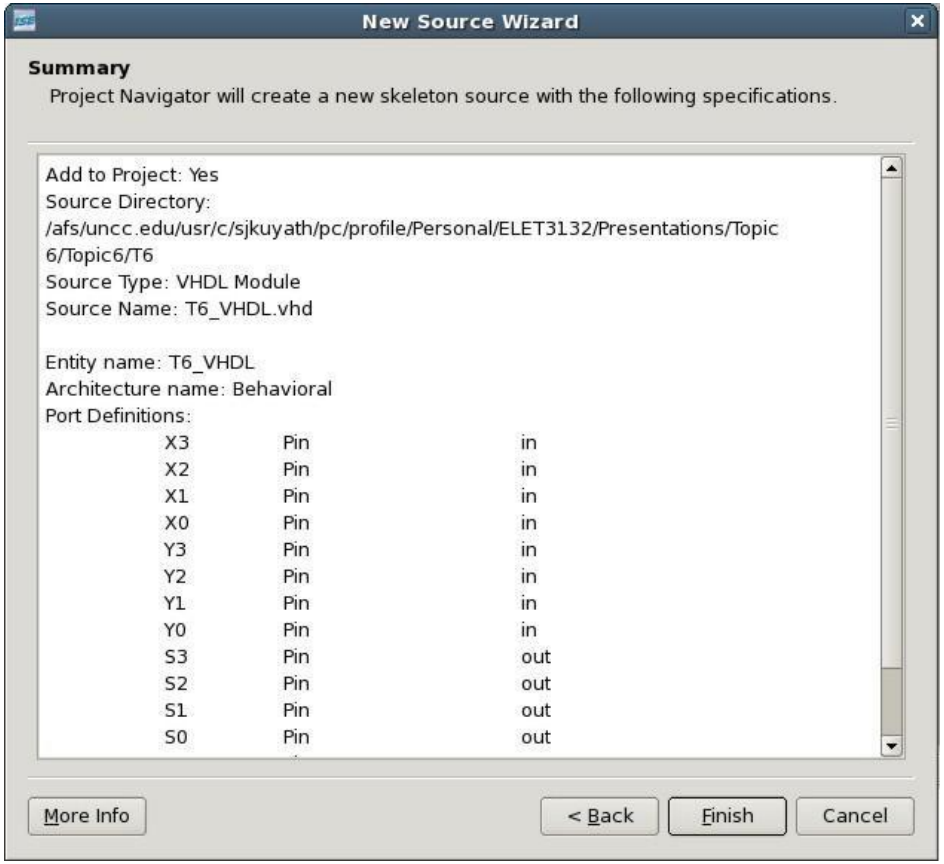
- In this box, you specify the Entity name, the Architecture name, and all the inputs and outputs



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

Xilinx Opening Screens

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

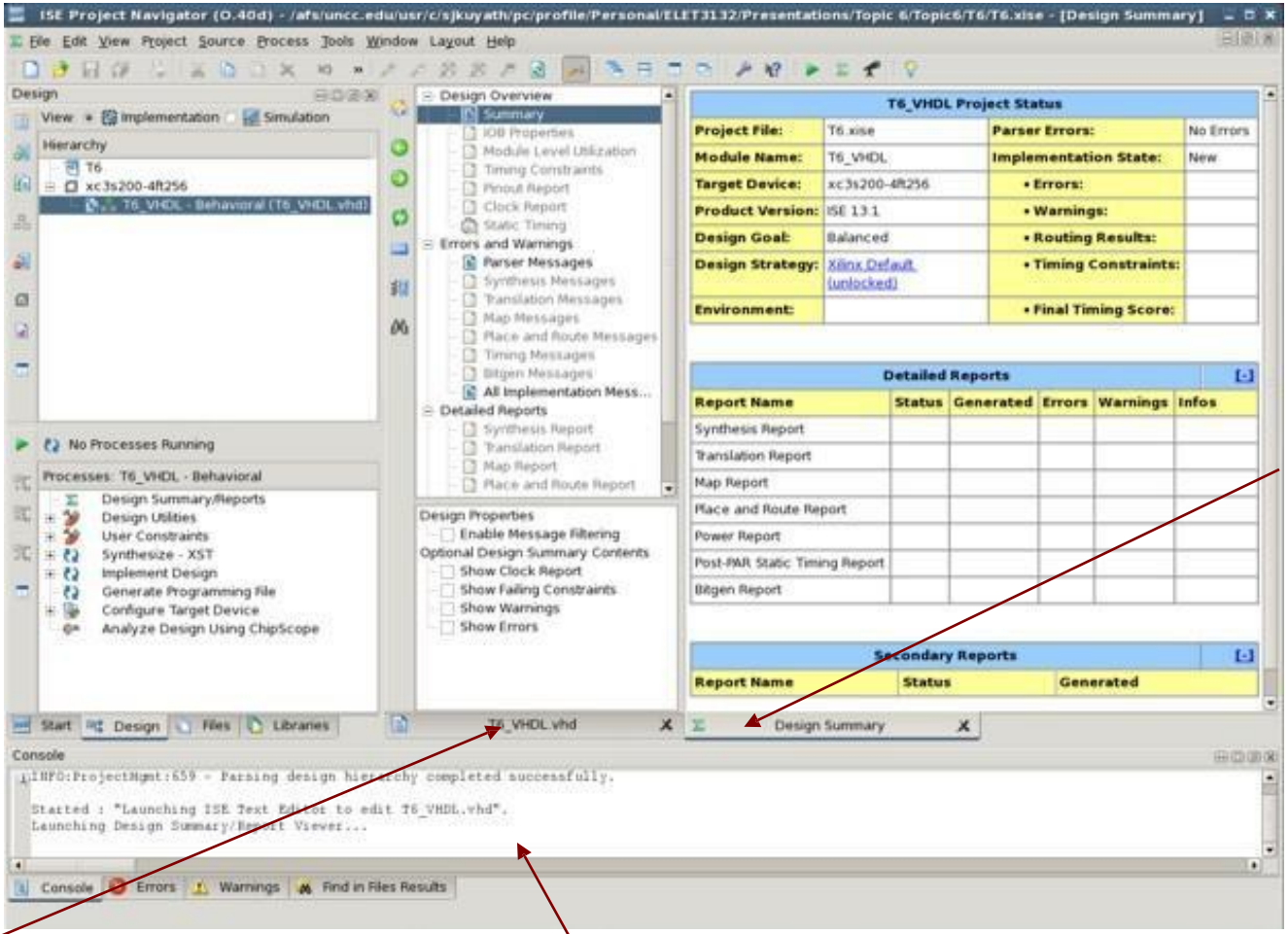


When verified click the [Finish] button

Xilinx Initial Setup

This window shows the files in the project also called the Design Hierarchy window.

This window shows the processes you can execute for the design file highlighted above– just double click when you need to execute one of these

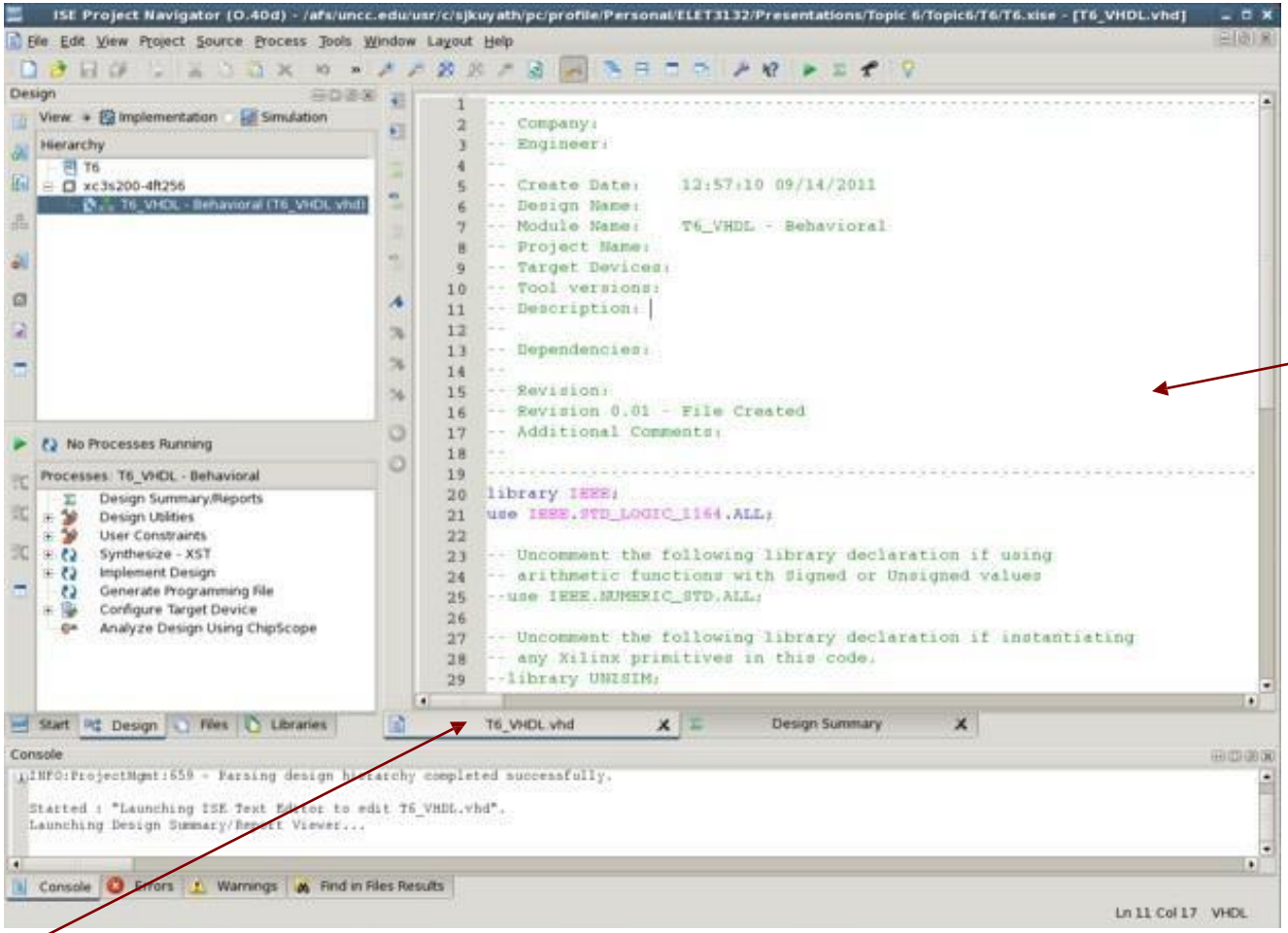


This window shows the summary information and is continually updated. To get to it, just click on the tab.

The VHDL file template is on this tab.

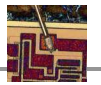
This window is a status window

Xilinx Initial Setup



The info on this screen will be shown in detail on the next page

The VHDL file template is on this tab.




VHDL Template File

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

```
----------------------------------------------------------------------------------
-- 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;
```

```
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 :
end adder4;out 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 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;
```

It reserves space for the logic




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.

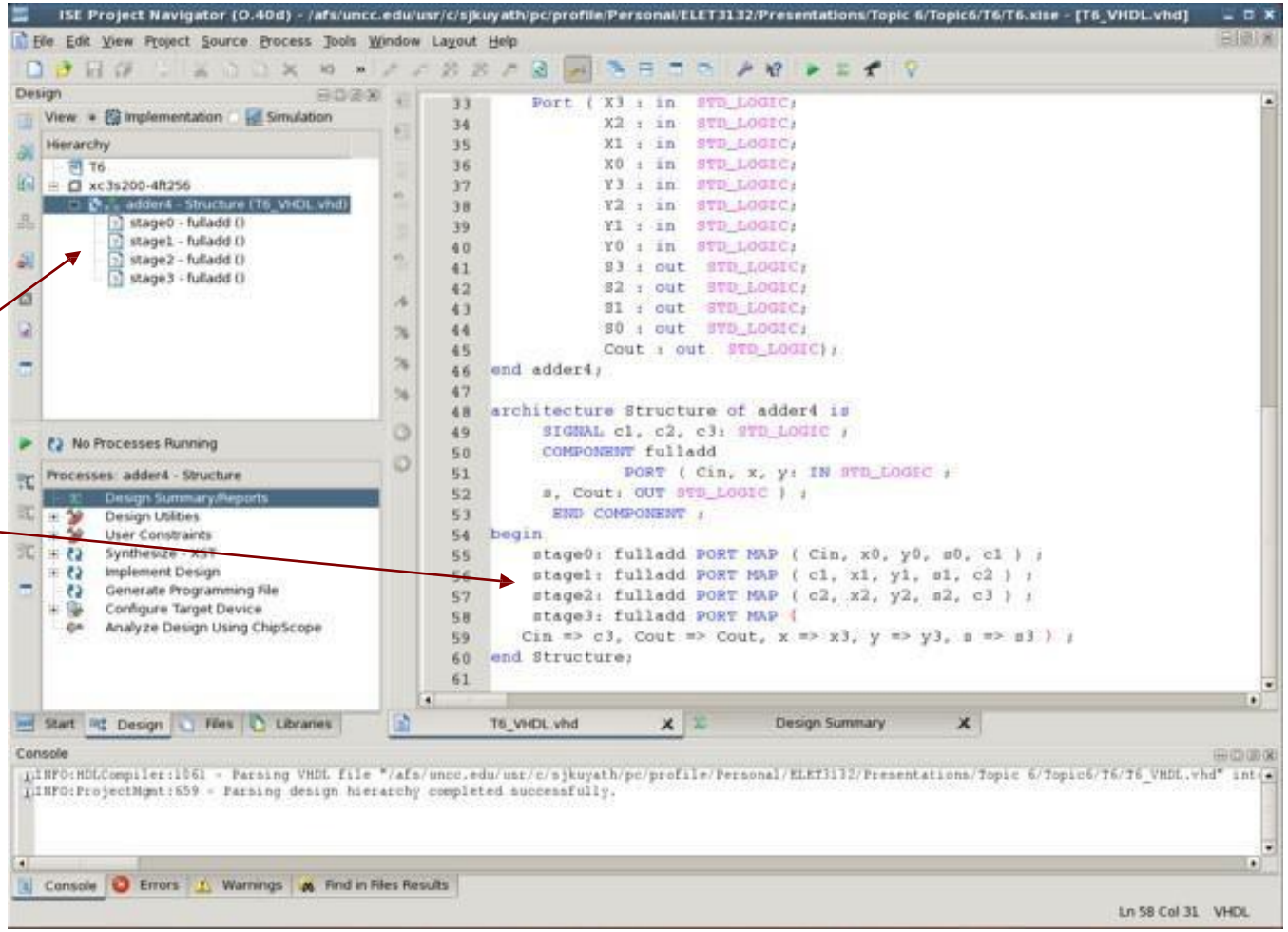
This is because "fulladd" was declared as an entity in a different file.




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

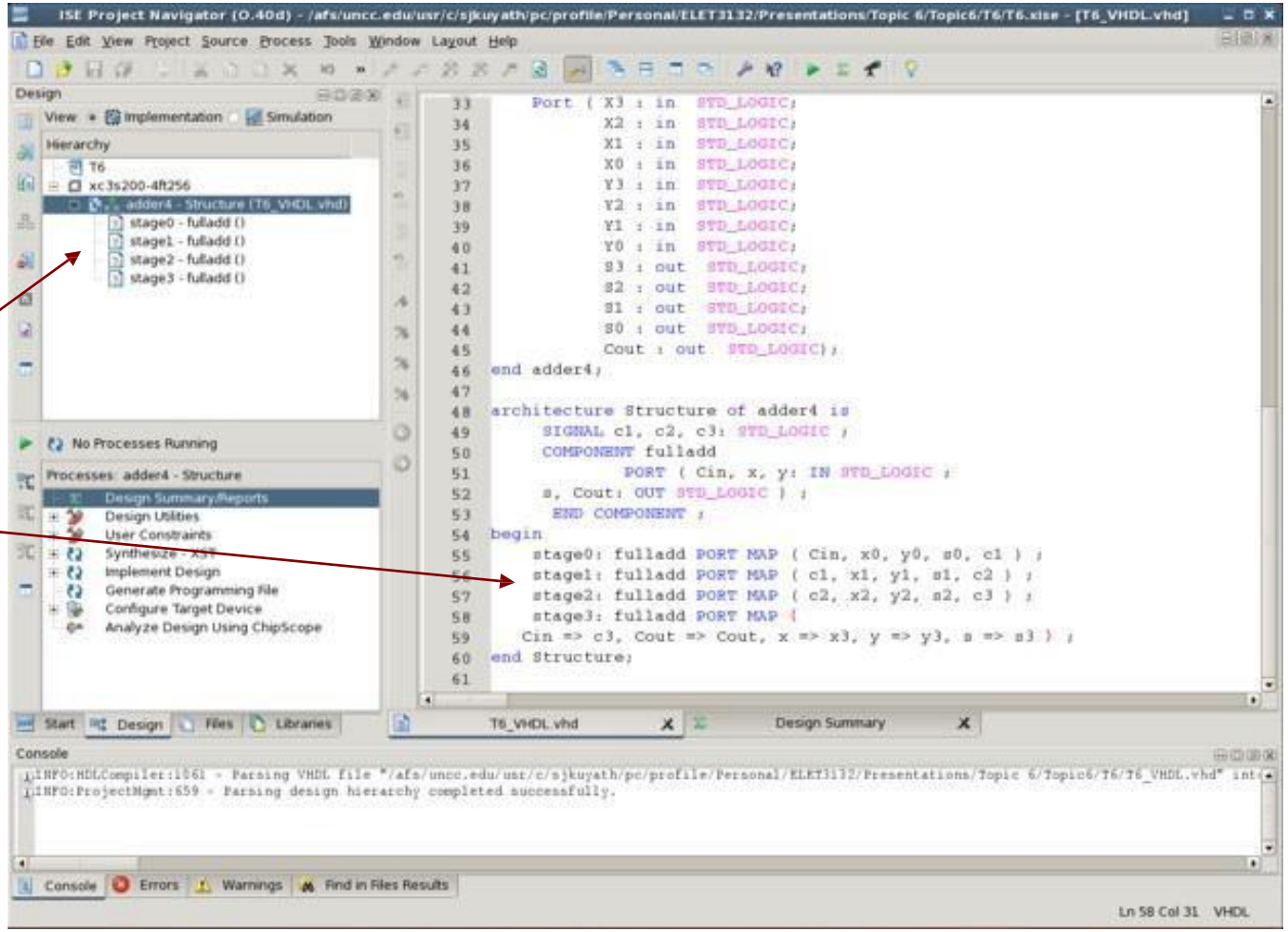


VHDL Template File

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

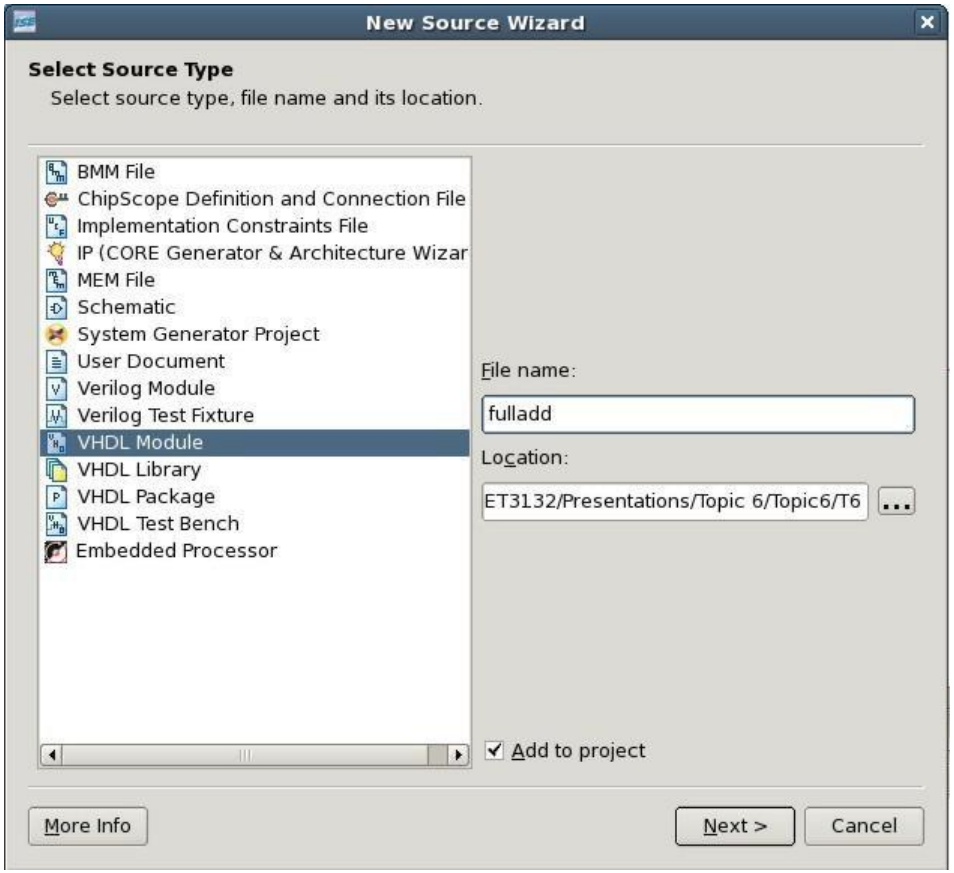
Double click on fulladd



VHDL Template File

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

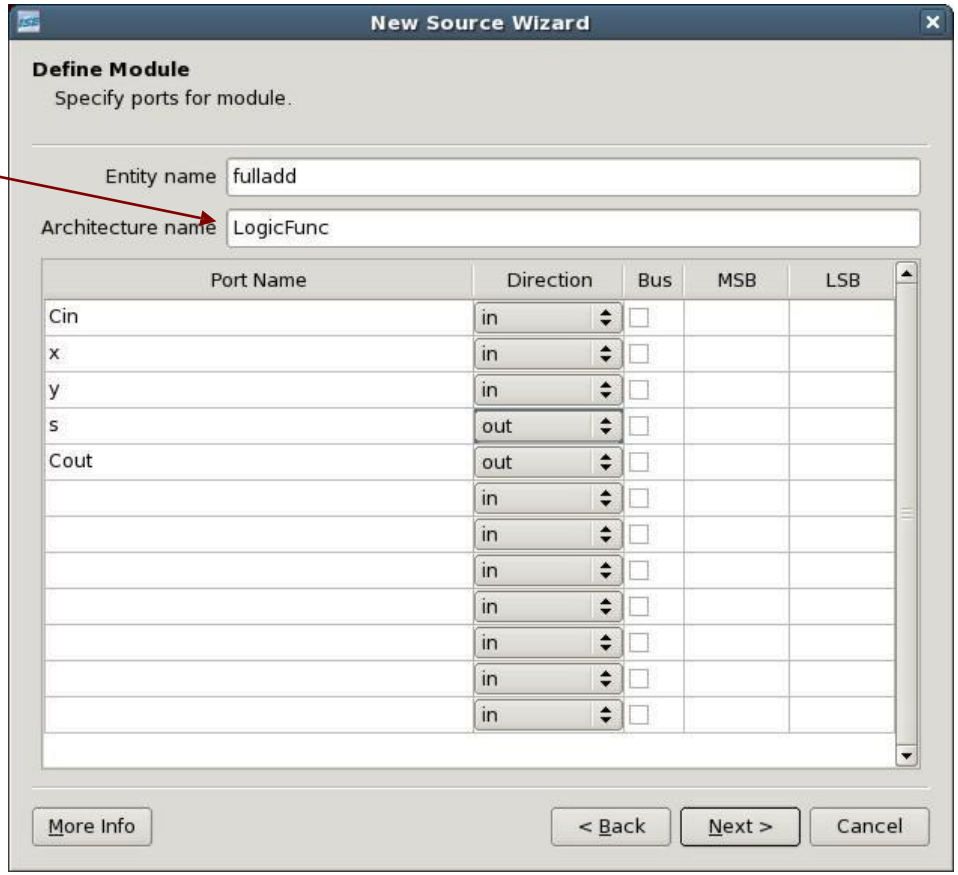
The ISE is wanting to know what type of file fulladd is. We will be entering a VHDL file for fulladd, so click on VHDL Module and then click [Next]



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]

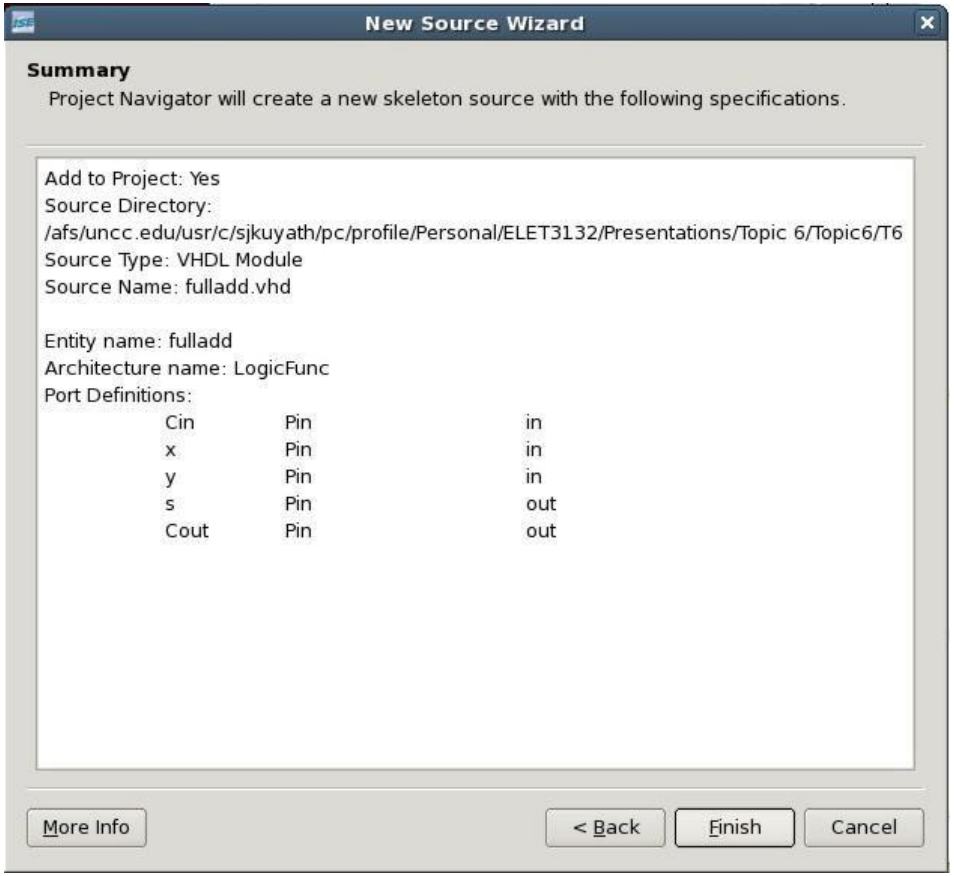





VHDL Template File

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

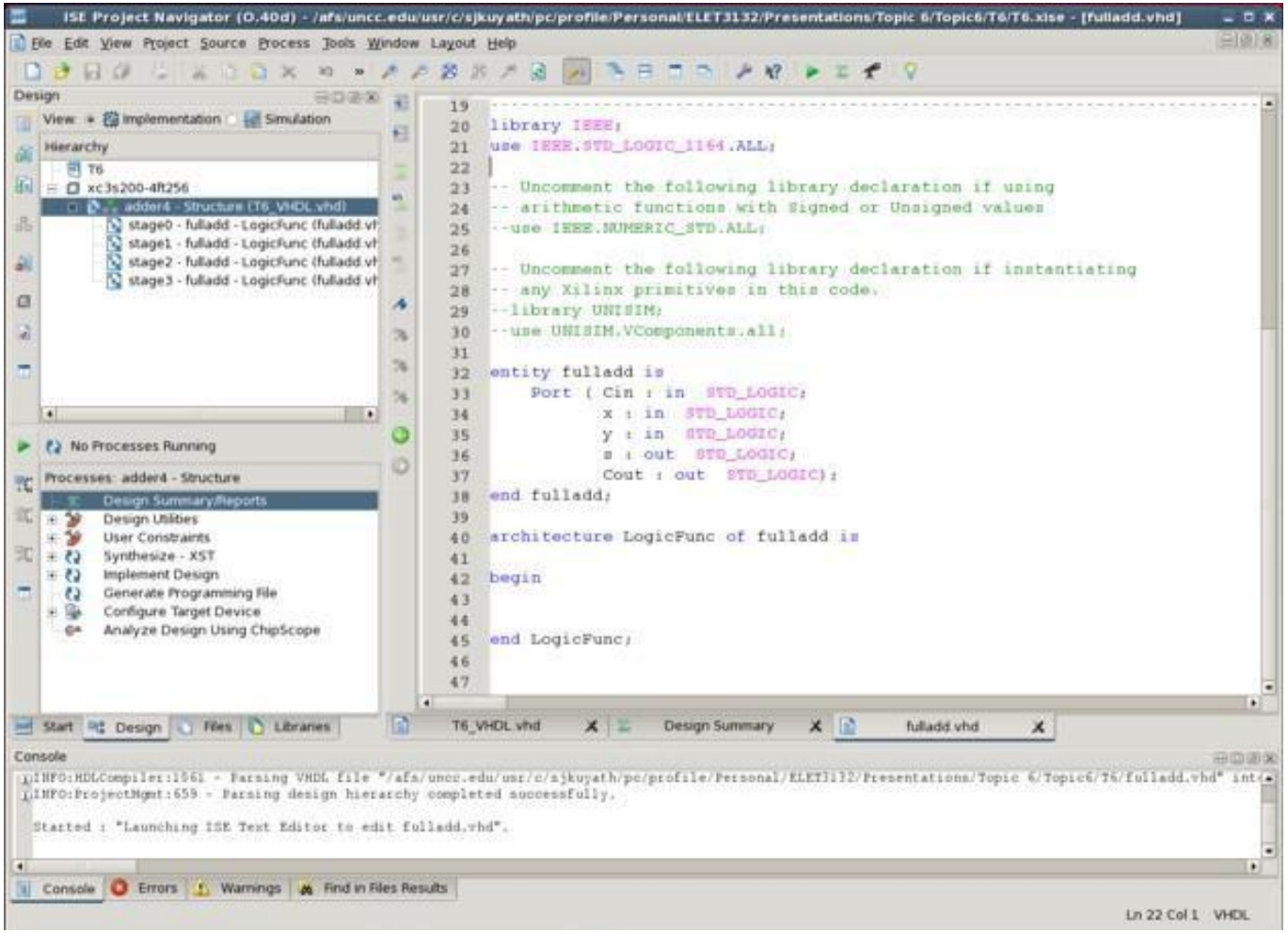
Click [Finish]



VHDL Template File

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

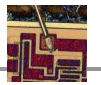
Note that the fulladd icon no longer has a question mark and is named appropriately




```
19  ----------------------------------------------------------------------------------
20  library IEEE;
21  use IEEE.STD_LOGIC_1164.ALL;
22
23  -- Uncomment the following library declaration if using
24  -- arithmetic functions with Signed or Unsigned values
25  --use IEEE.NUMERIC_STD.ALL;
26
27  -- Uncomment the following library declaration if instantiating
28  -- any Xilinx primitives in this code.
29  --library UNISIM;
30  --use UNISIM.VComponents.all;
31
32  entity fulladd is
33      Port ( Cin : in  STD_LOGIC;
34             x : in  STD_LOGIC;
35             y : in  STD_LOGIC;
36             s : out  STD_LOGIC;
37             Cout : out  STD_LOGIC);
38  end fulladd;
39
40  architecture LogicFunc of fulladd is
41
42  begin
43
44
45  end LogicFunc;
46
47
```

```
INFO:HDLCompiler:1561 - Parsing VHDL file "/afs/uncc.edu/usr/c/sjkuyath/pc/profile/Personal/ELET3132/Presentations/Topic 6/Topic6/T6/fulladd.vhd" int
INFO:ProjectMgmt:659 - Parsing design hierarchy completed successfully.

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




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.

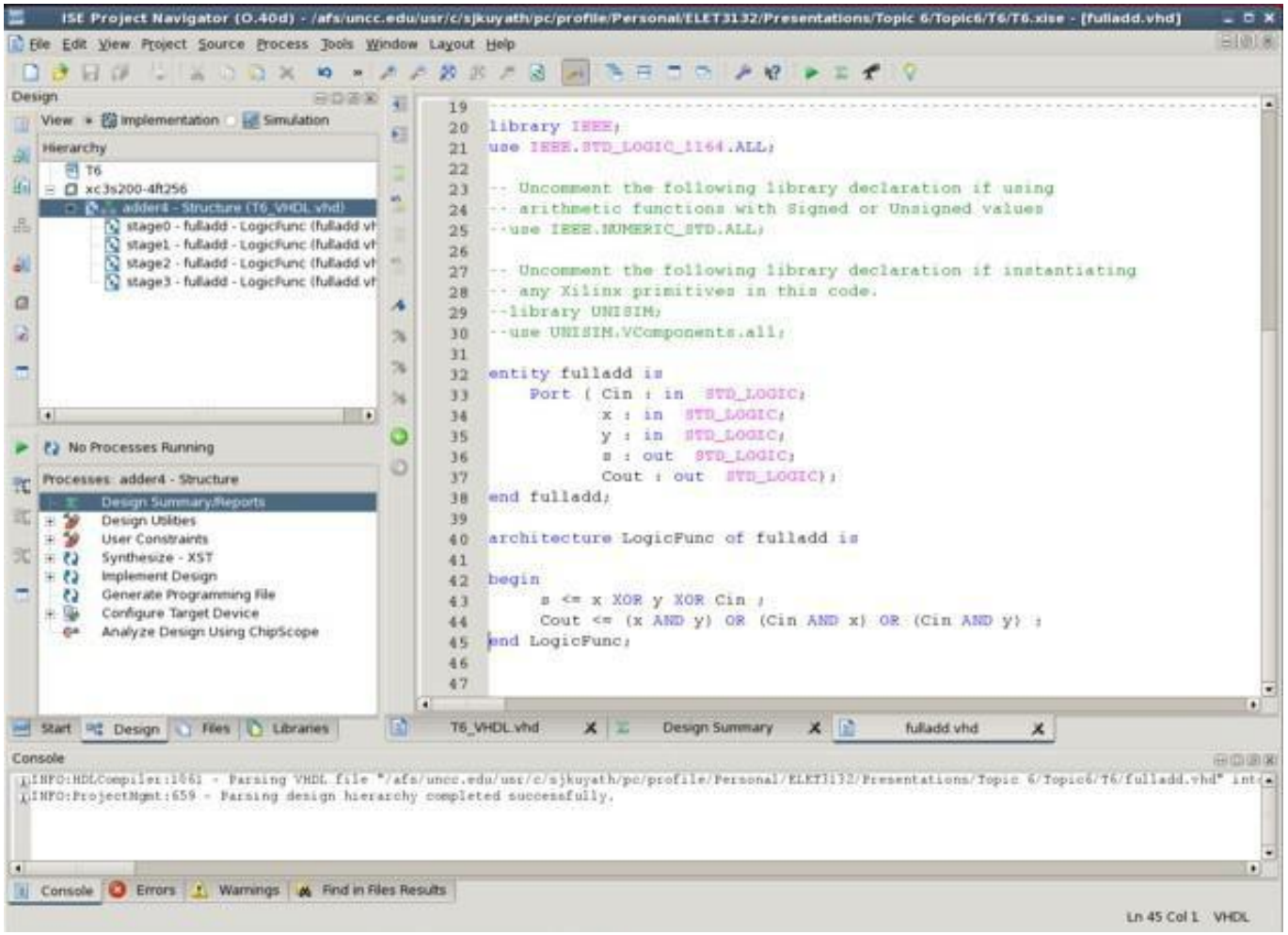
This is VERY important!! It shows that the ISE will allow you to create a component for use in other files: Once the component is debugged, it can be used over and over.




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

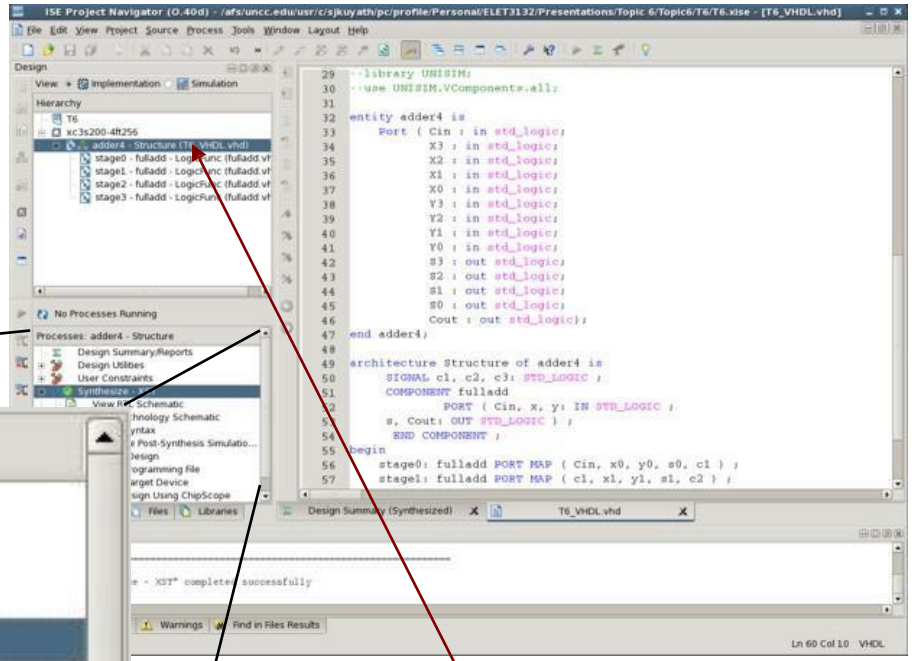


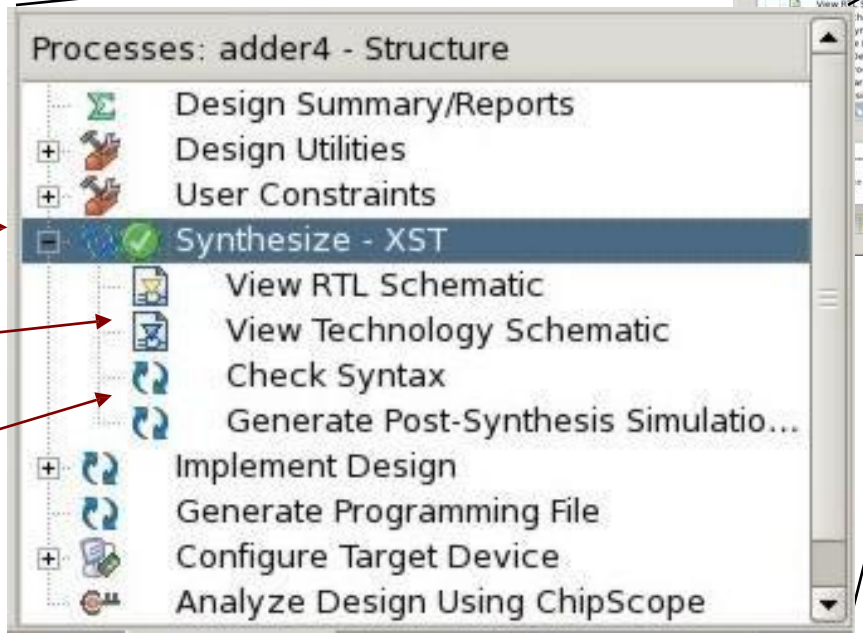
Synthesize

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





Reports

Actions

Note that the “adder4” module is highlighted

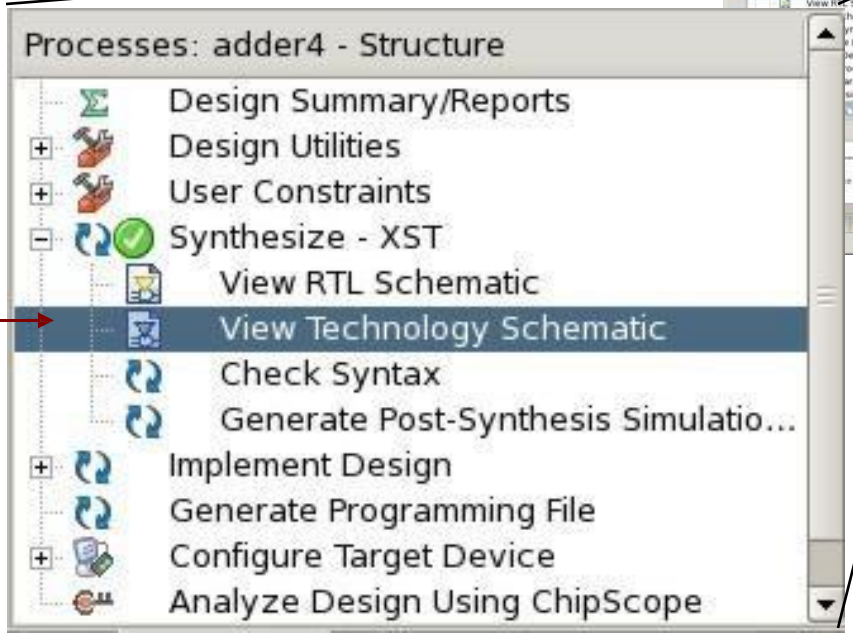
ver 13 won’t allow you to synthesize the component, but ver 7 did

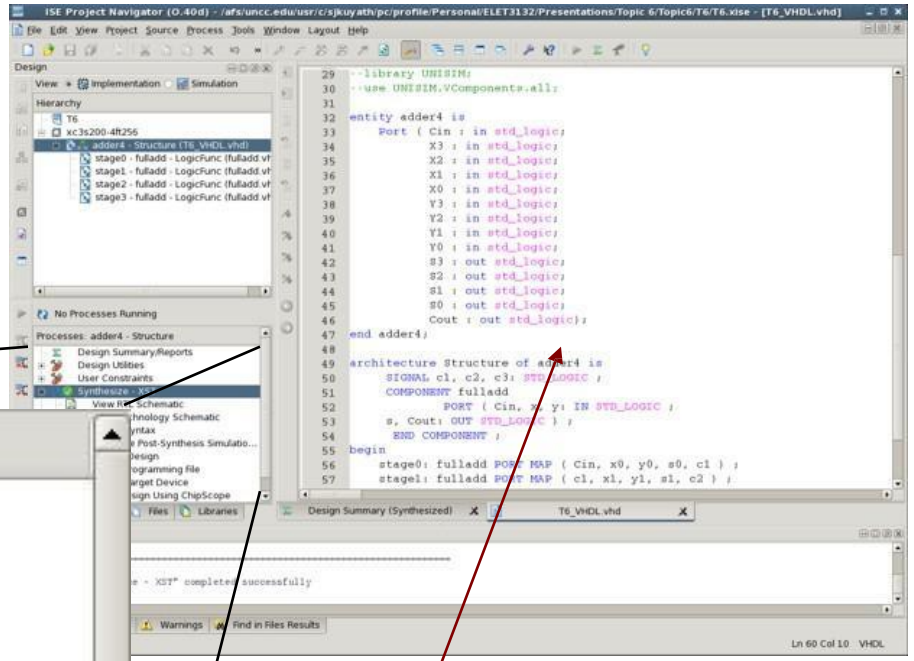
View Technology Schematic

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

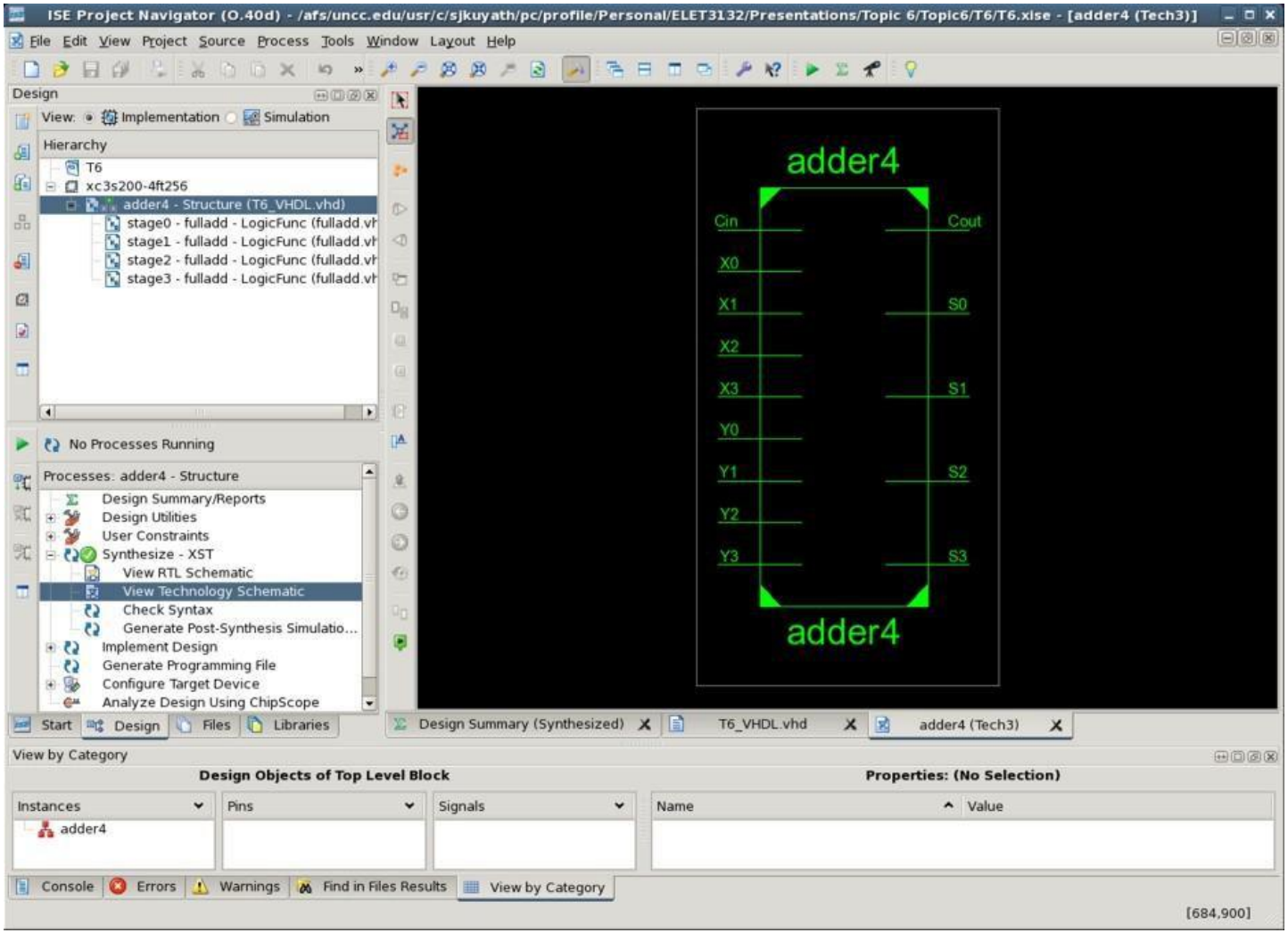
This dialog box appears. Highlight the lower radio button and click OK




View Technology Schematic

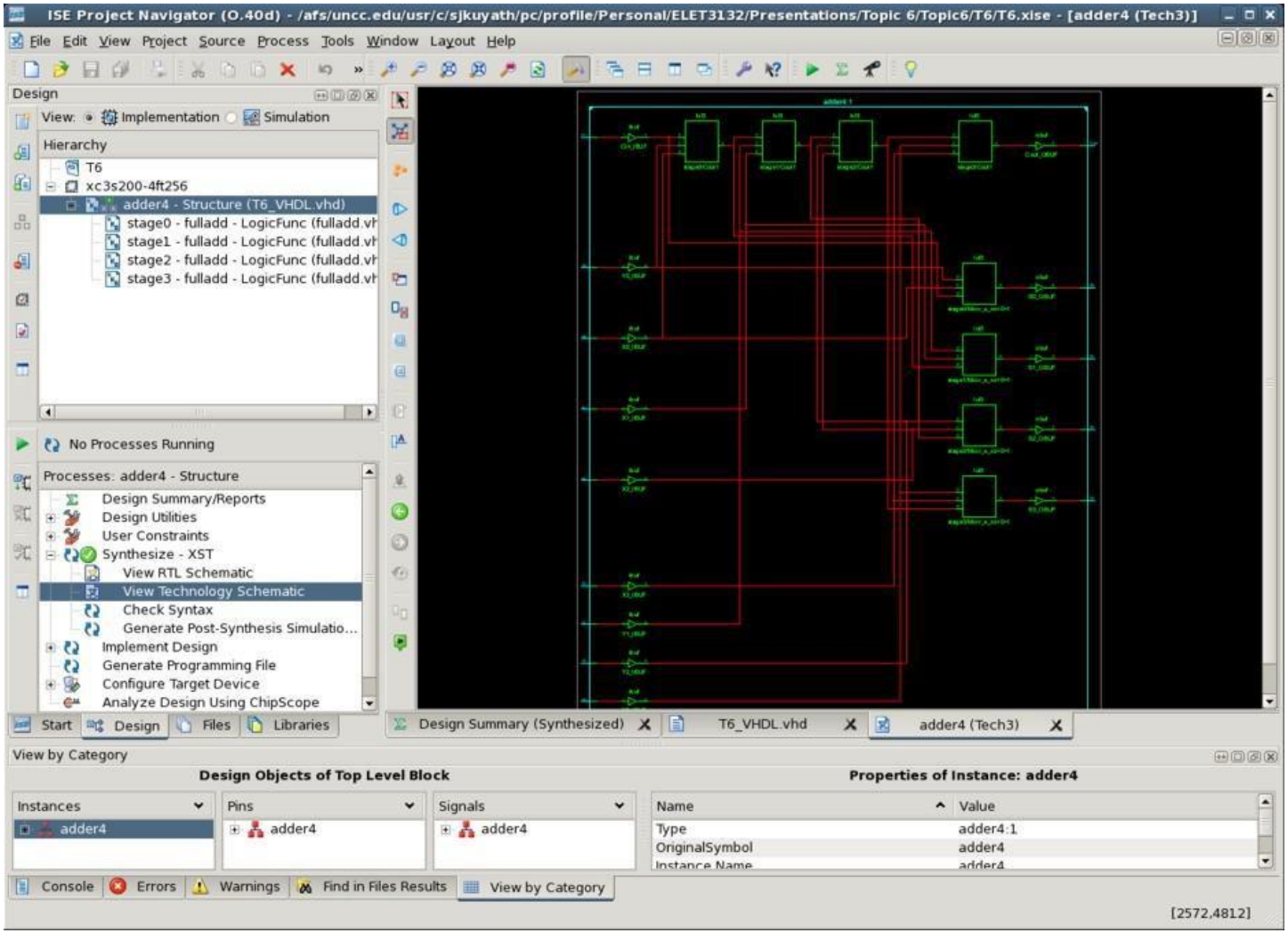
You'll see the complete symbol for the "adder4". It shows the **ENTITY** (the inputs and the outputs, but no functionality).

Double click on the entity.



View Technology Schematic

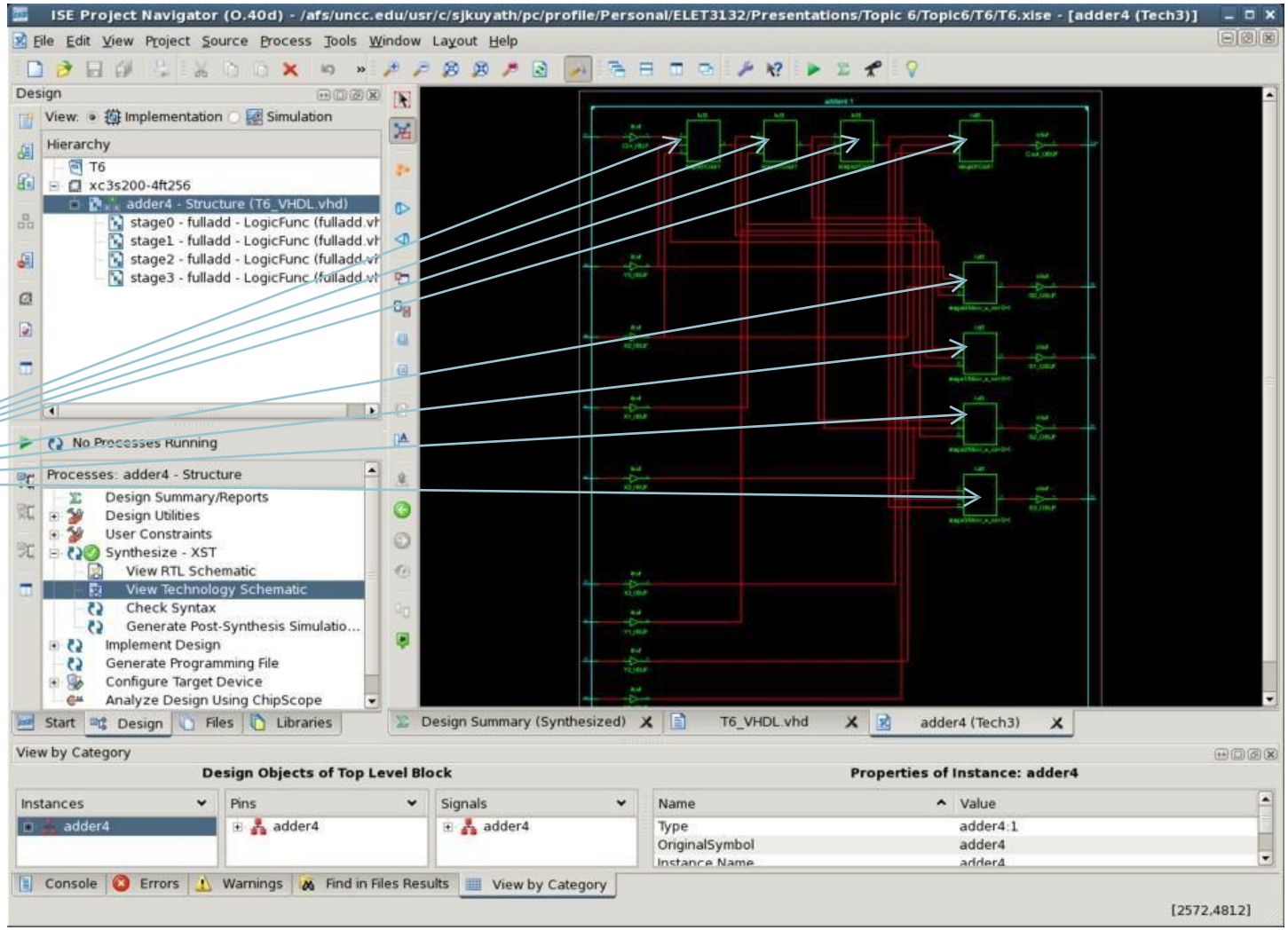
When you double click on the adder4 symbol, the entity, the functionality appears



View Technology Schematic

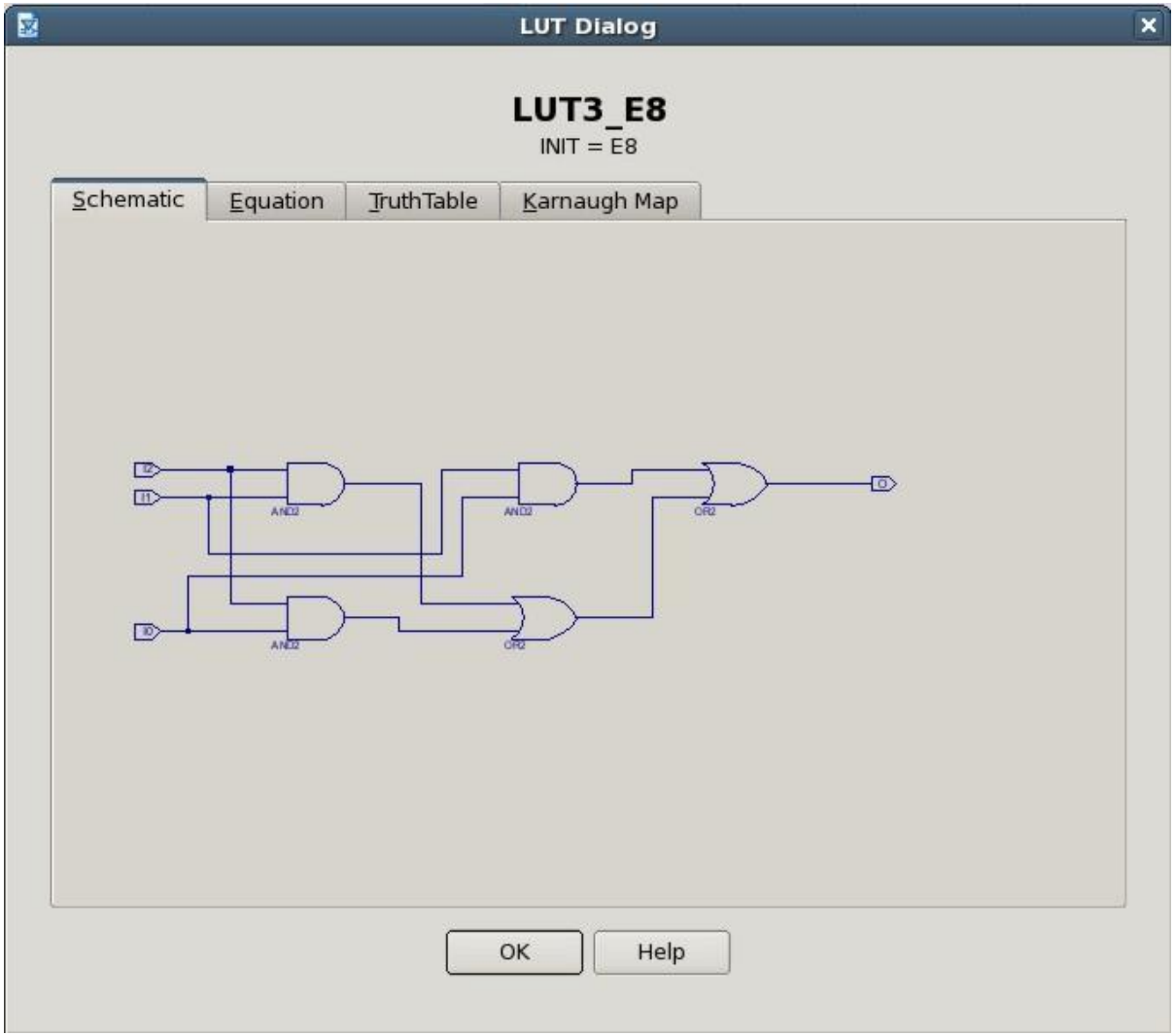
Double click on any of the LUTs and a new box appears

LUTs



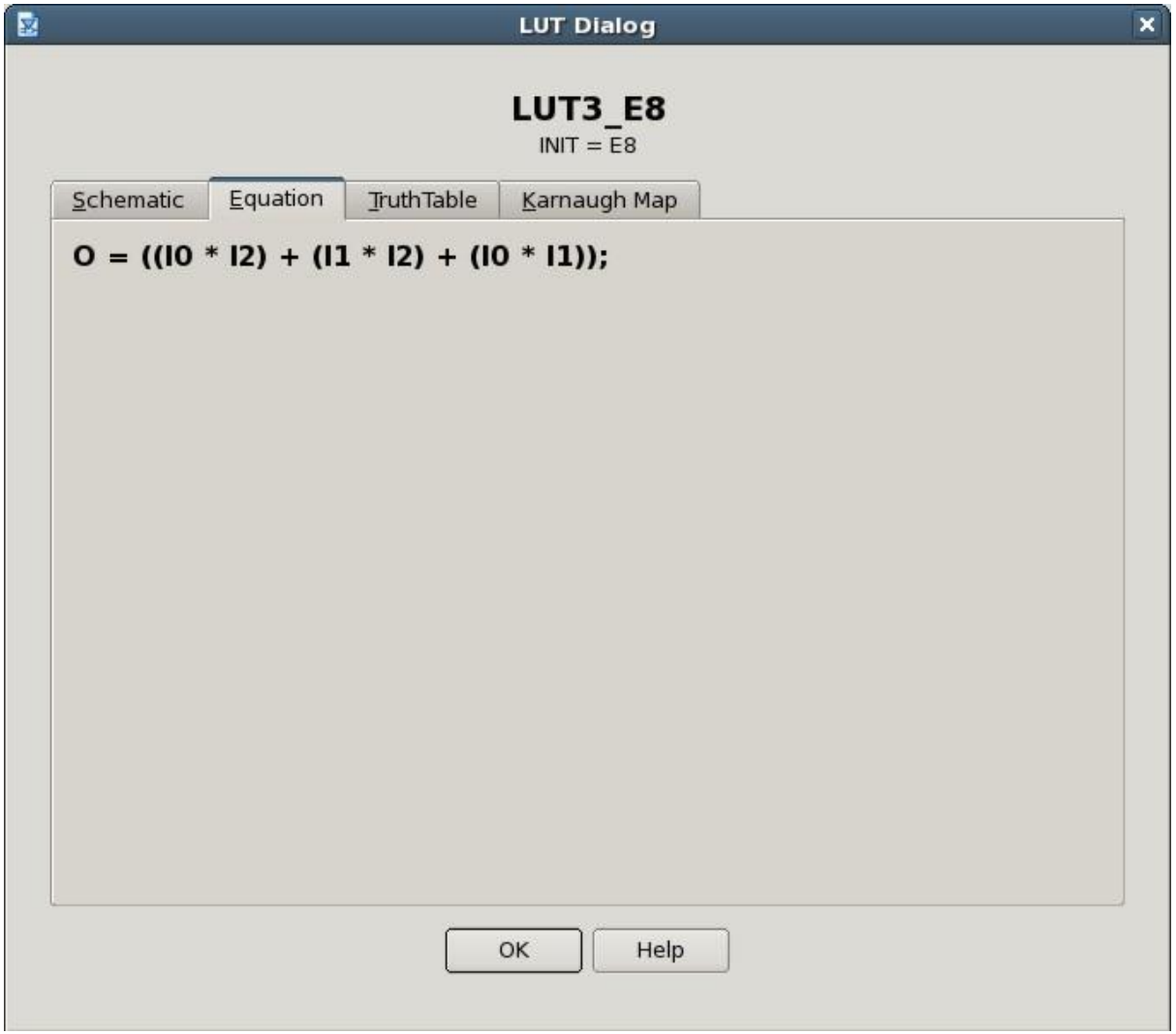
Contents of 1st LUT

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



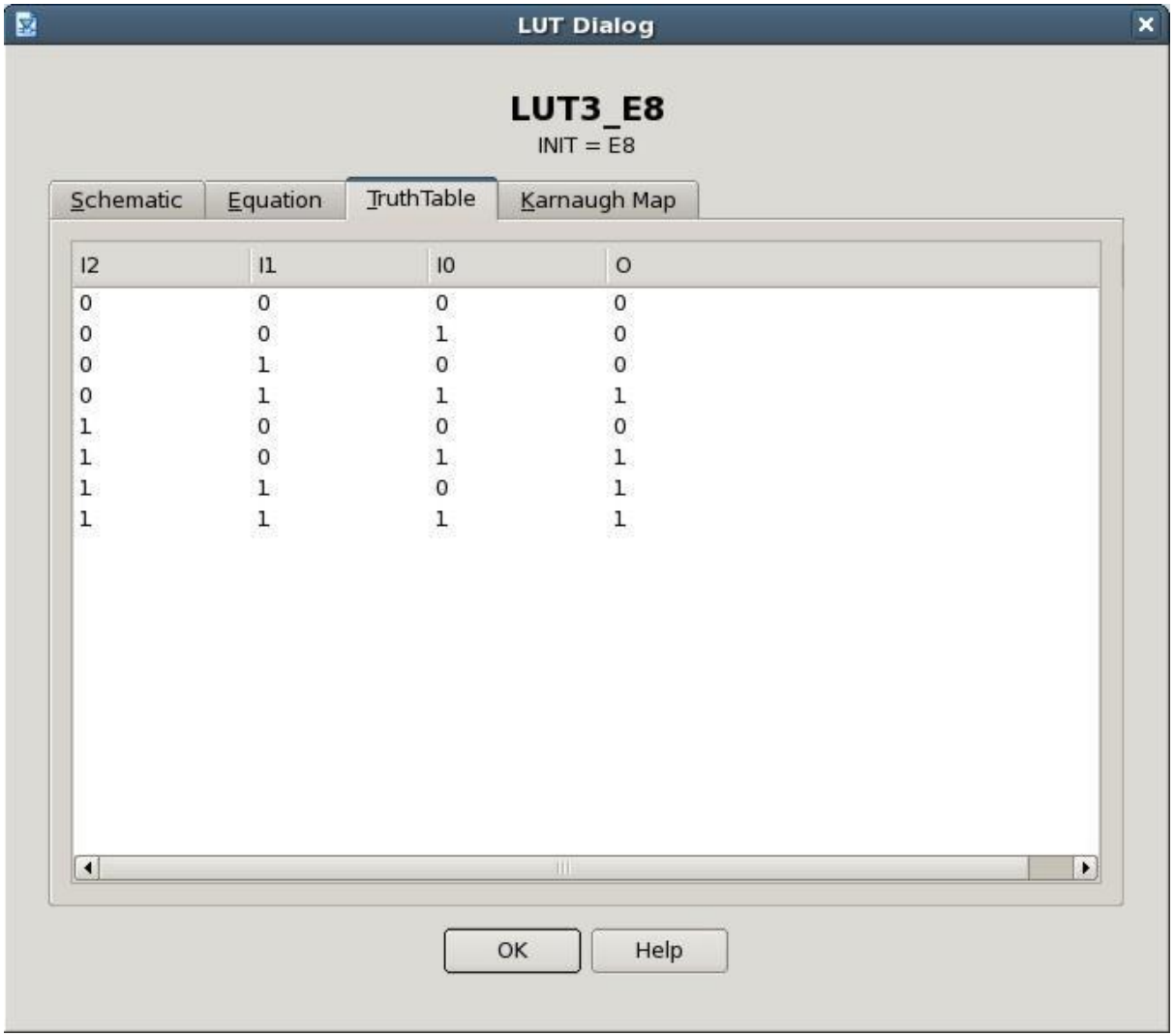
Contents of 1st LUT

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



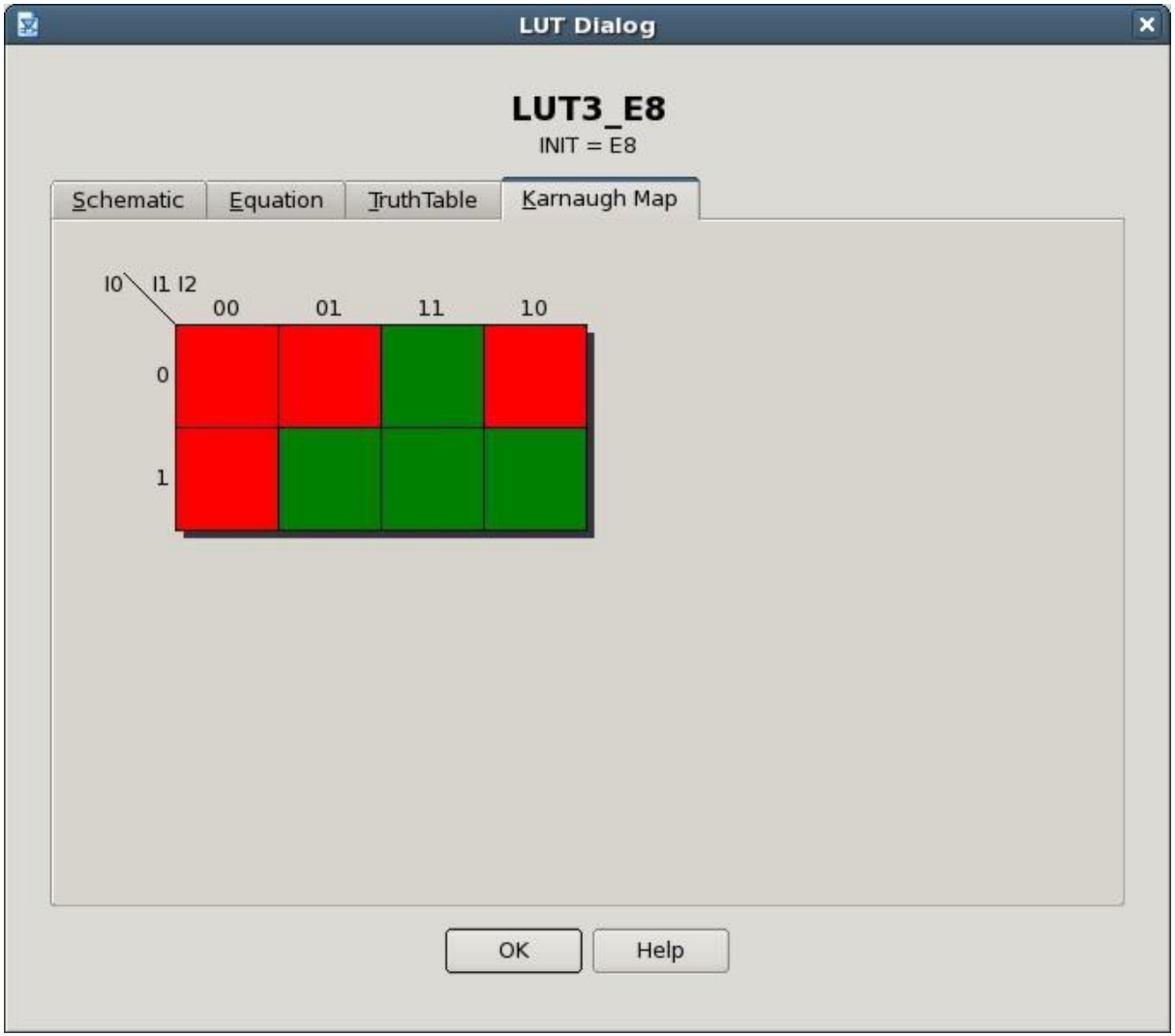
Contents of 1st LUT

The Truth Table of the 1st LUT



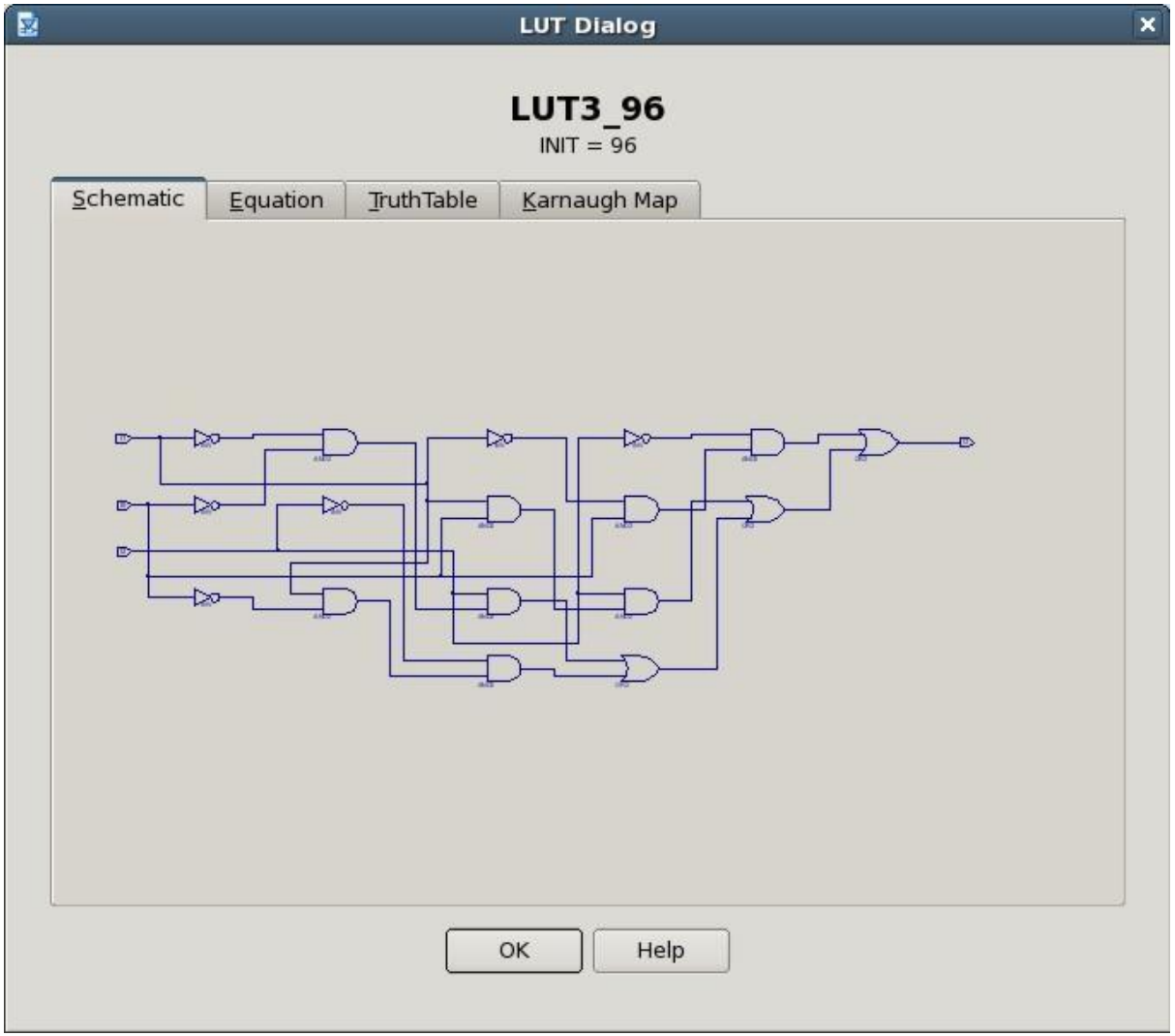
Contents of 1st LUT

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



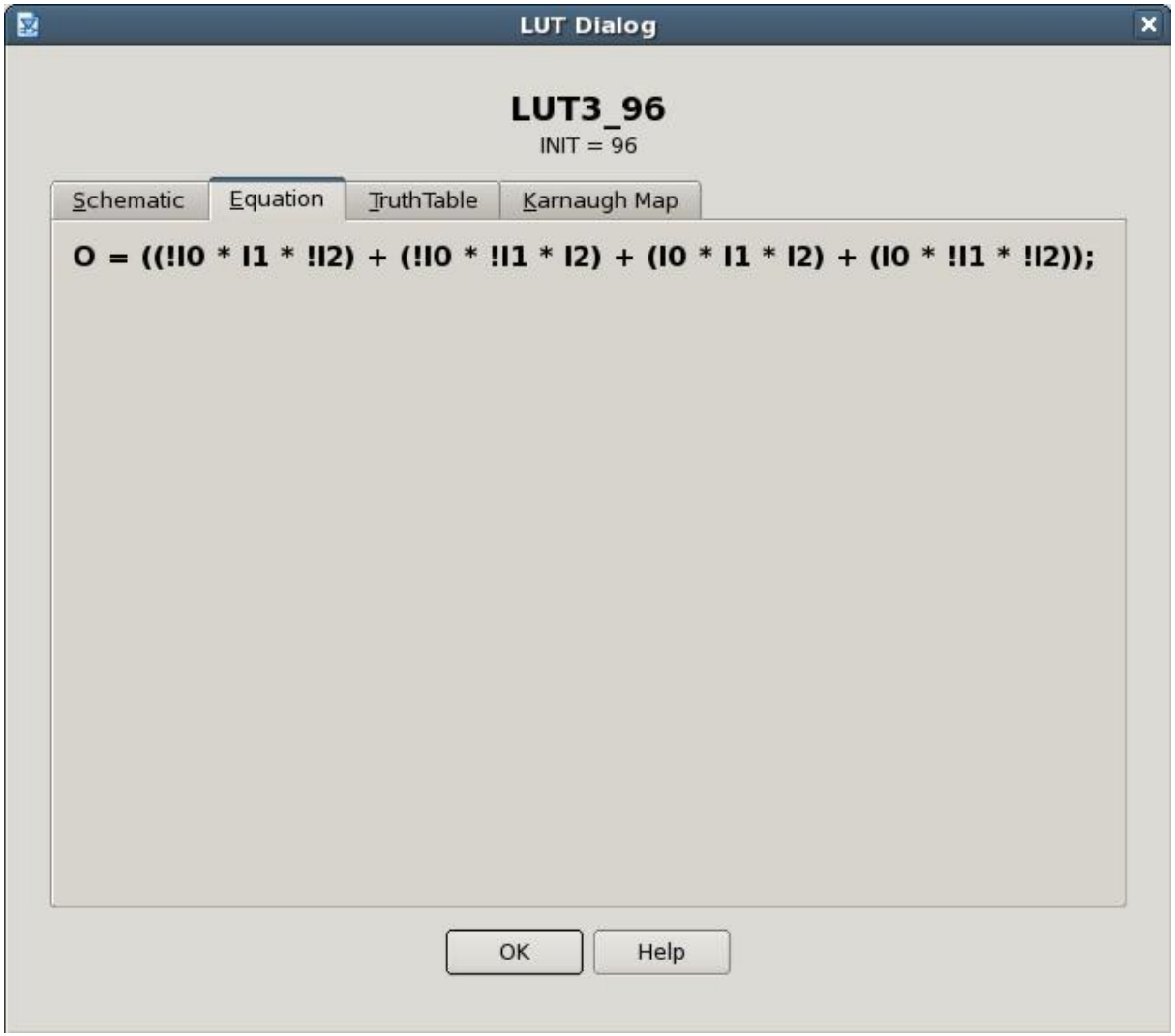
Contents of 2nd LUT

The Logic Diagram of the 2nd LUT



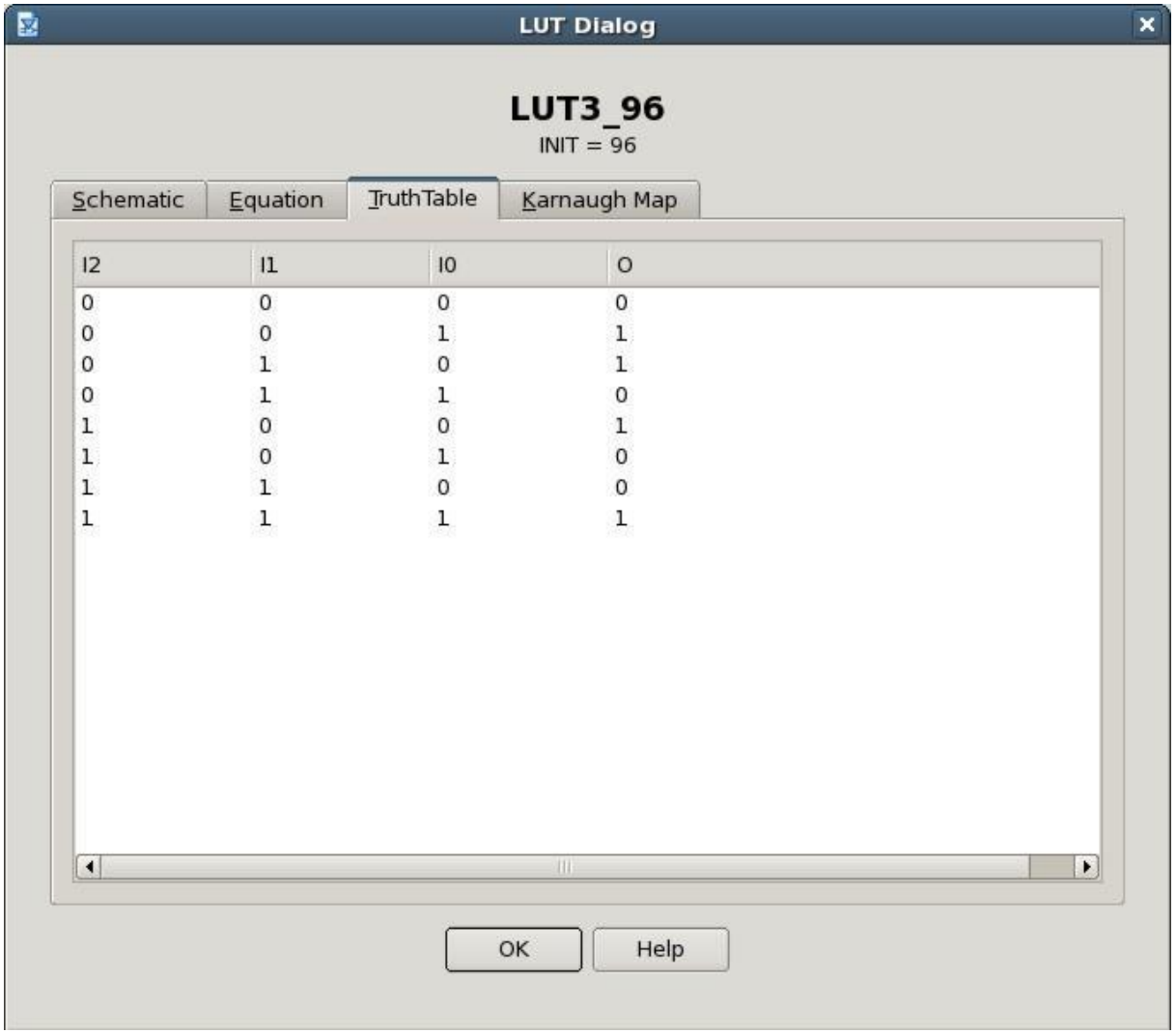
Contents of 2nd LUT

The equation of the 2nd LUT



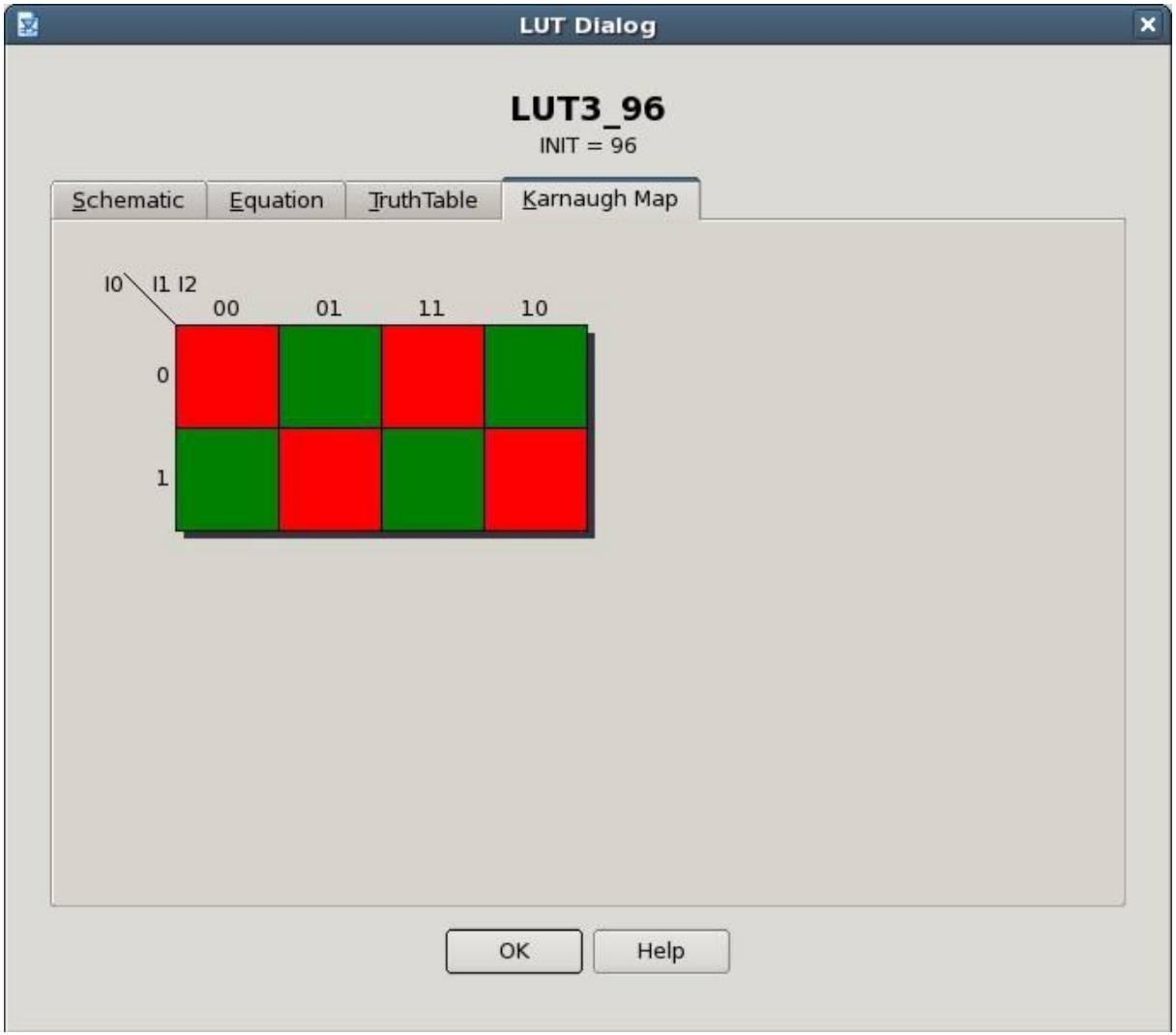
Contents of 2nd LUT

The Truth Table of the 2nd LUT



Contents of 2nd LUT

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



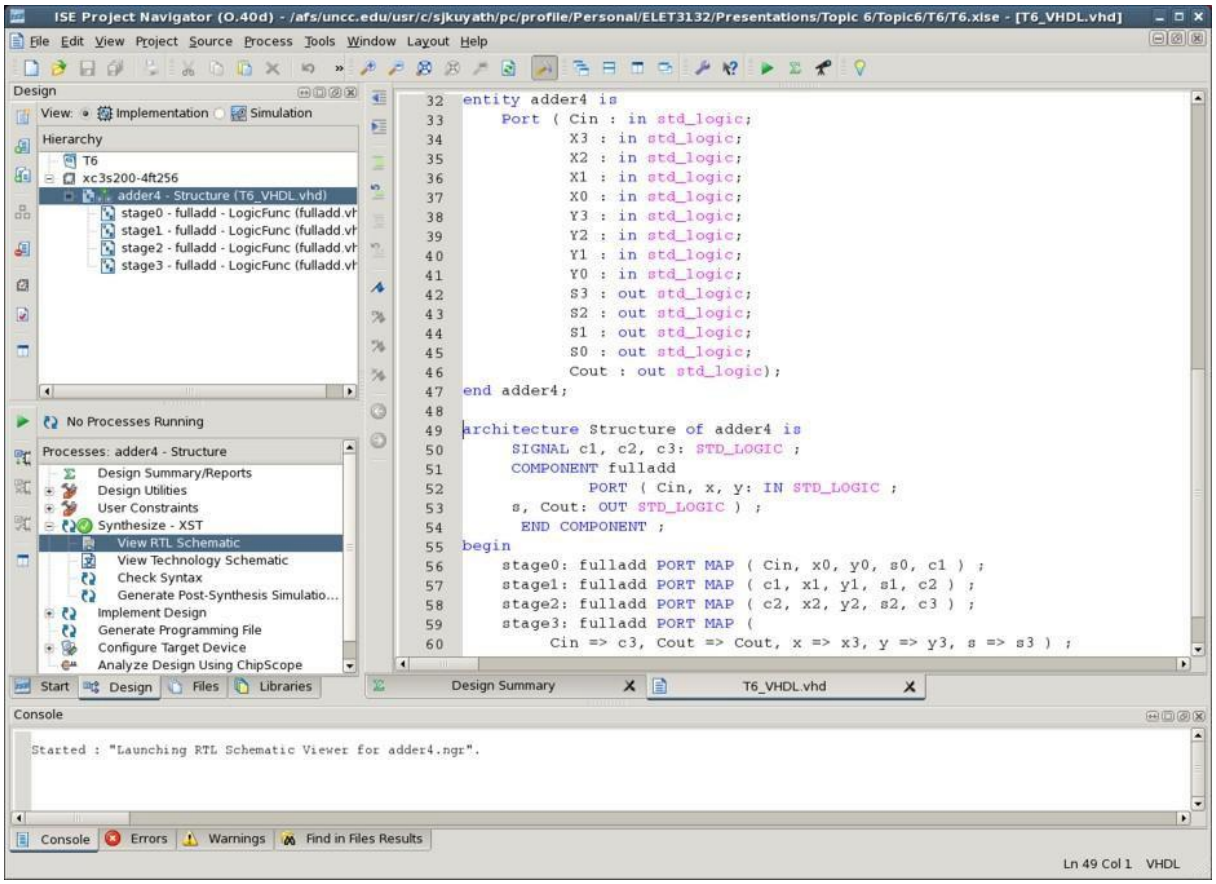
View RTL Schematic

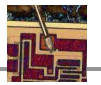
Viewing the RTL Schematic will show you a little different information.

"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"






1: From the XILINX help files on RTL Schematic

View RTL Schematic

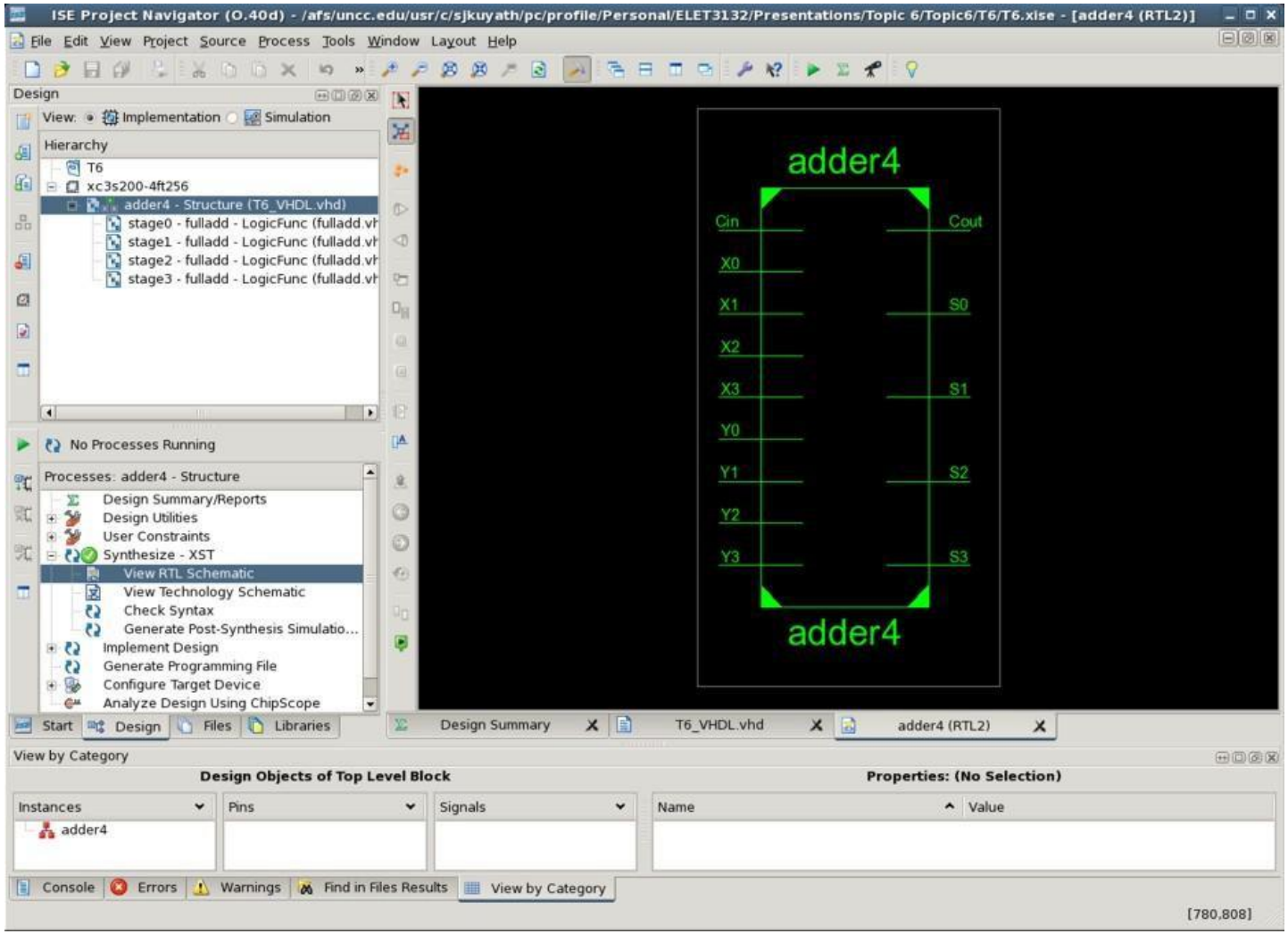
This dialog box appears (as it did before). Highlight the lower radio button, if it isn't already highlighted, and click OK

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.

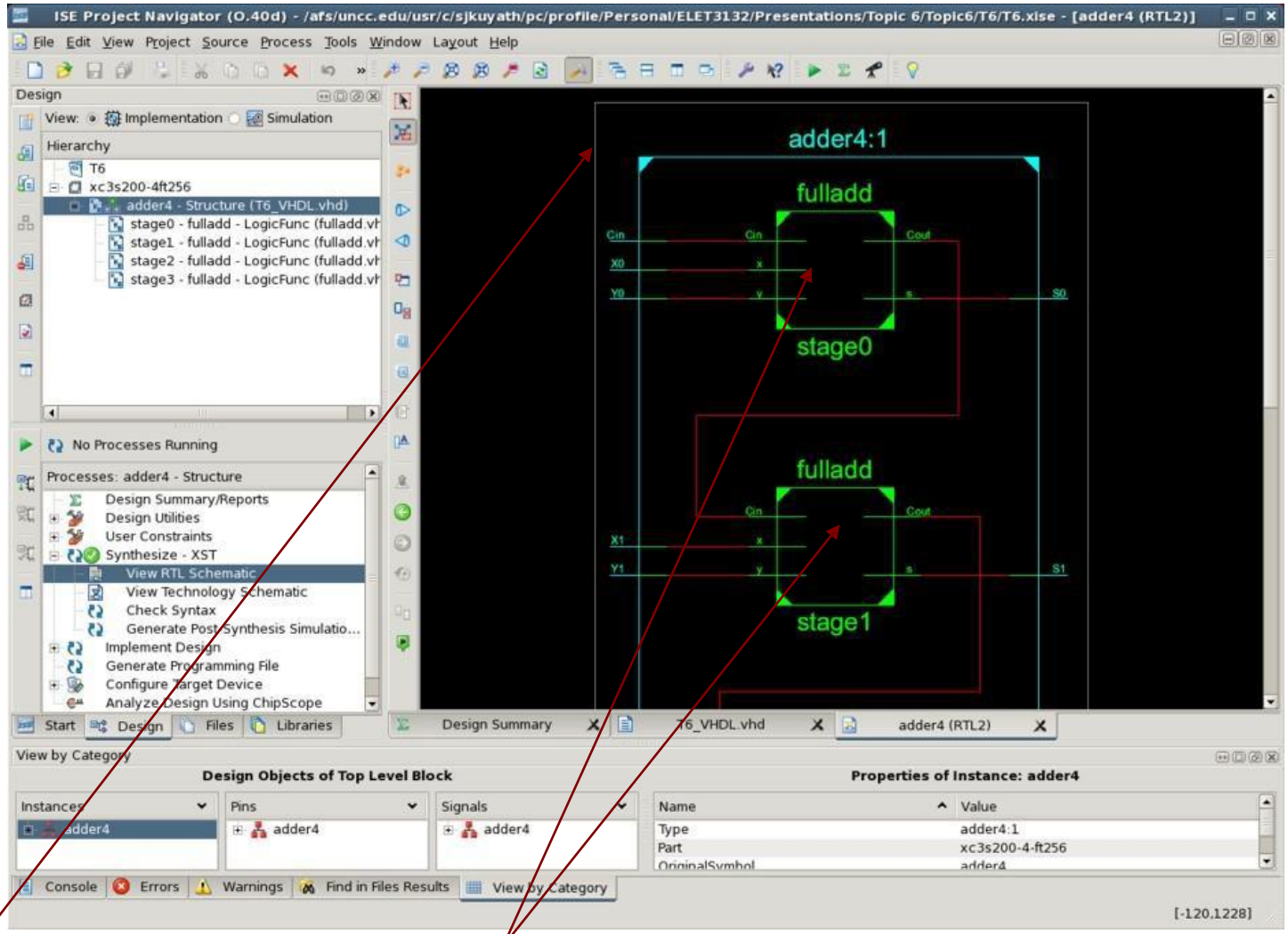
Double click on the entity.



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)

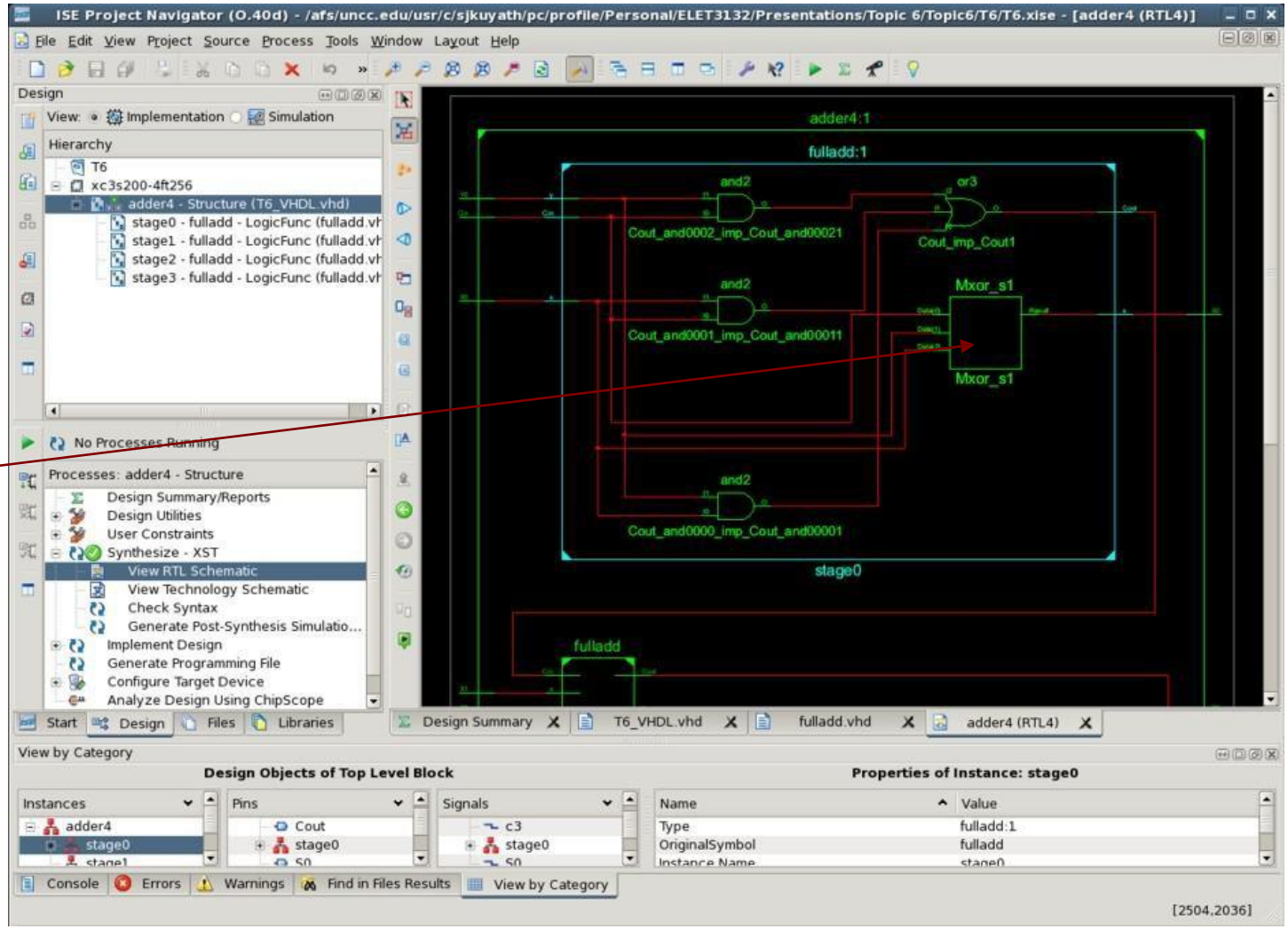
These are the individual full adders (the components)




View RTL Schematic

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

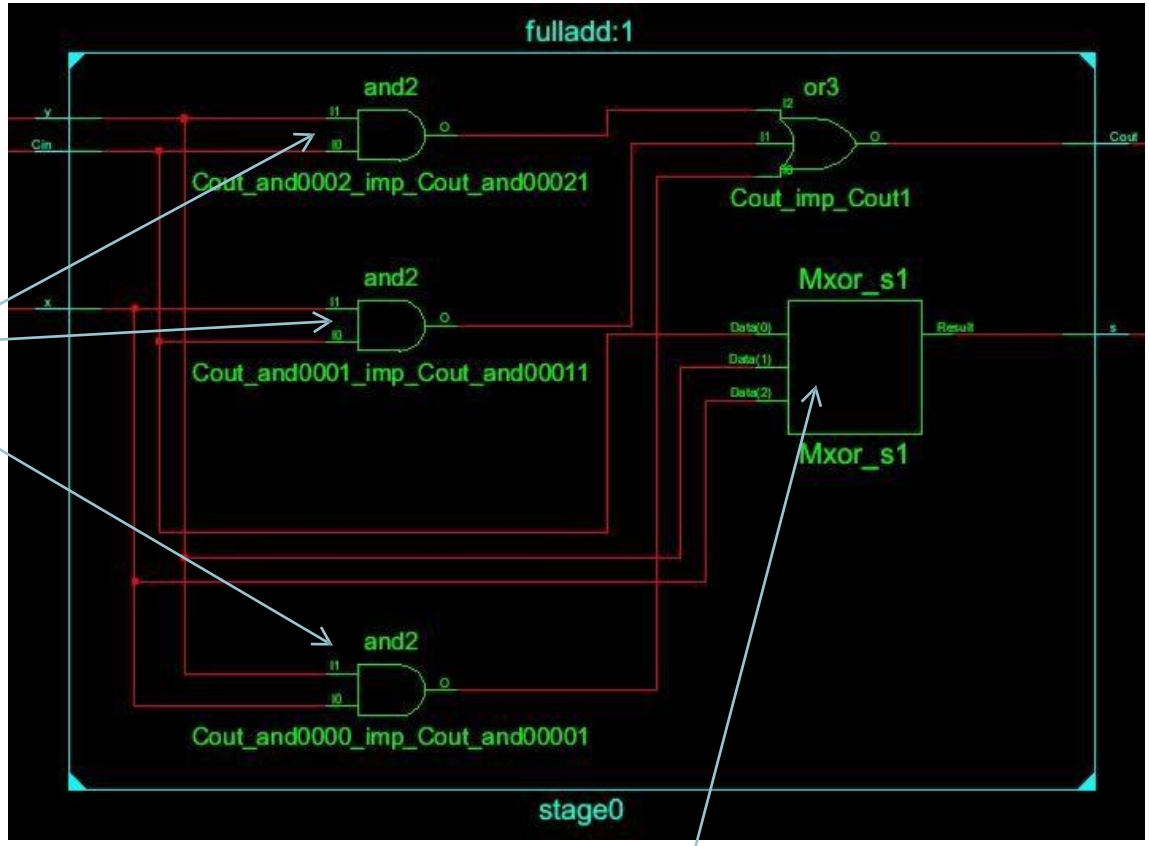
XOR gates are called Mxor



View RTL Schematic

The original equations for the fulladd are:

Cout <= (x AND y)
OR
(Cin AND x)
OR
(Cin AND y) ;



s <= x XOR y XOR Cin ;

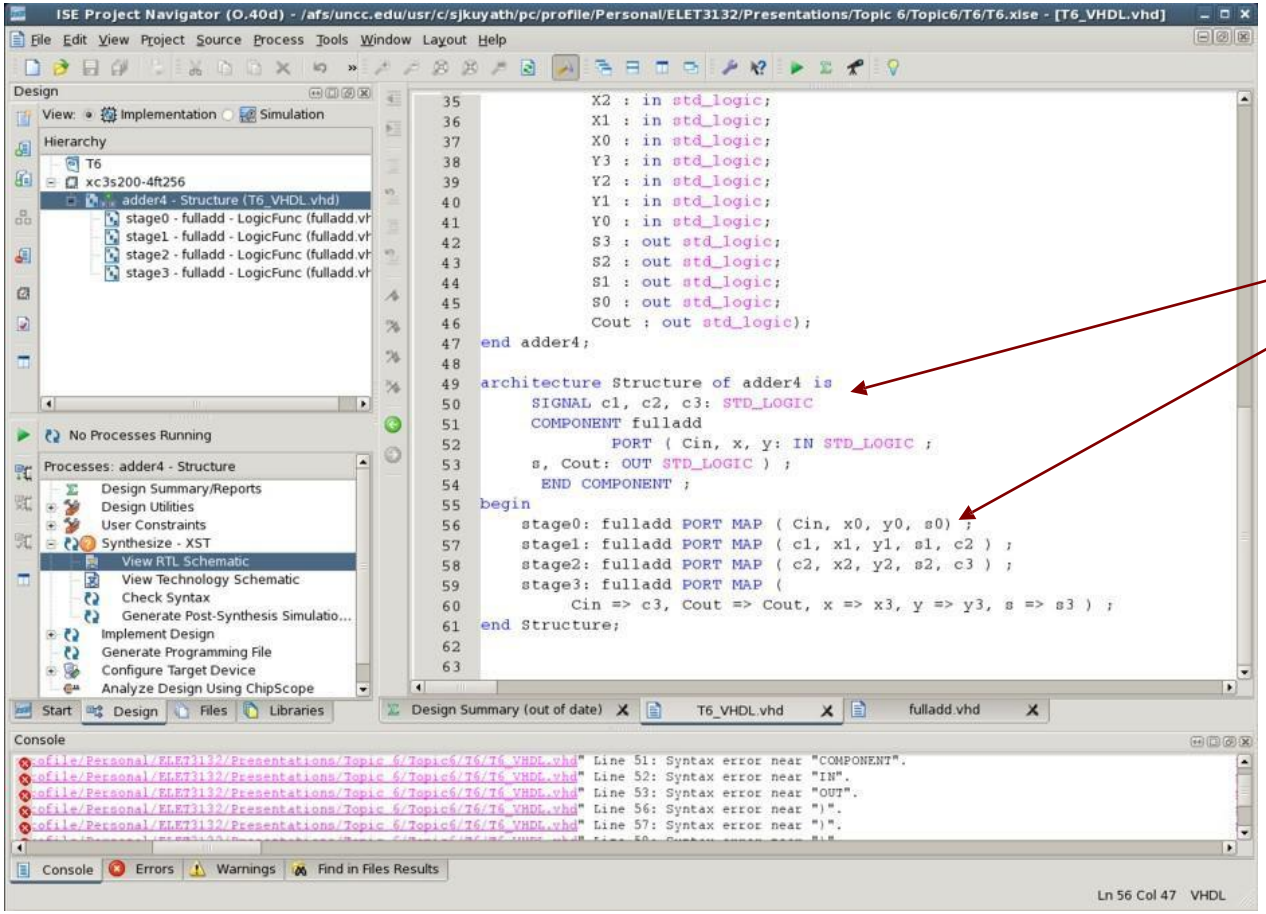



Verification

- 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

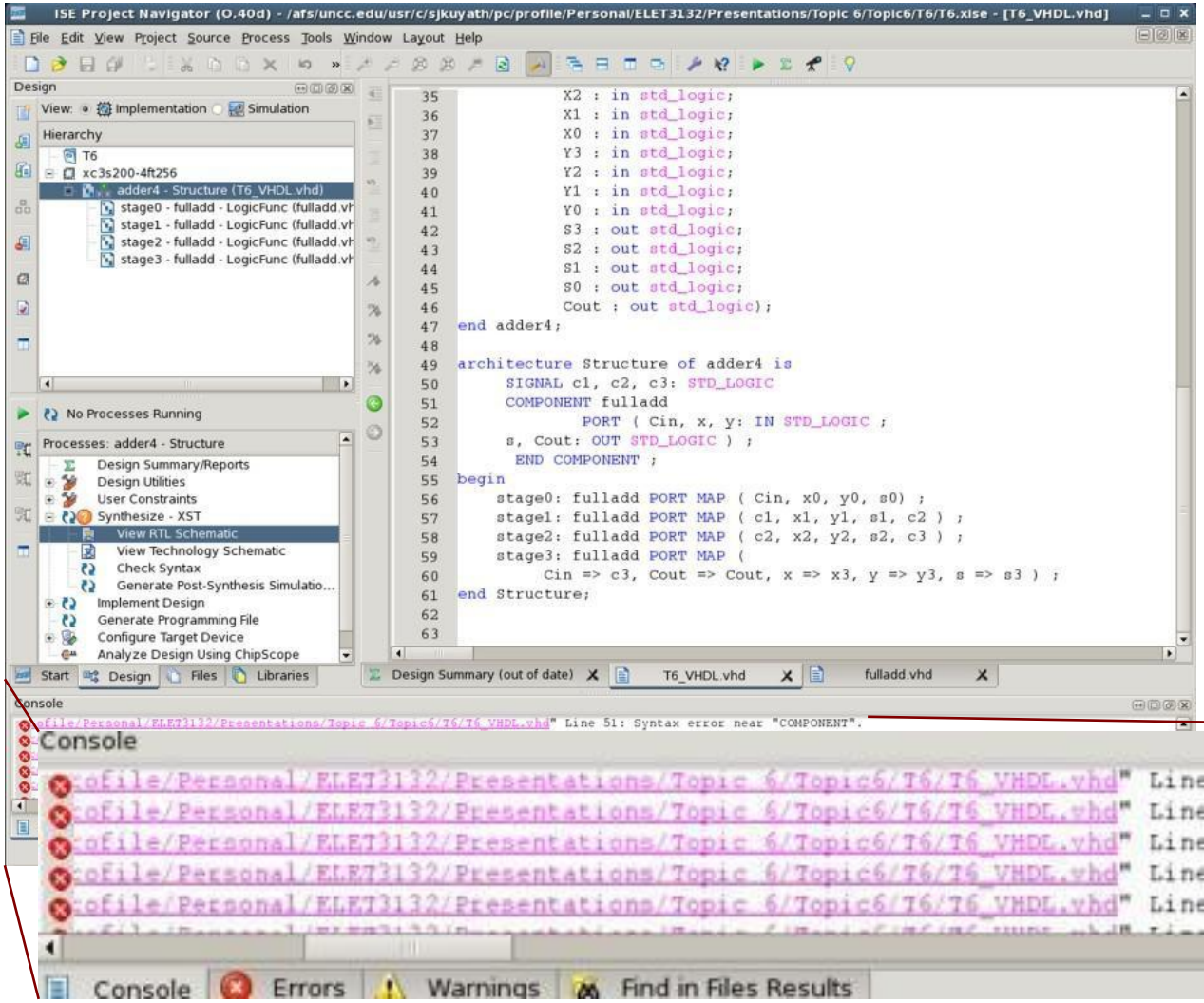


Two errors were put in this file
1. The semicolon was removed
2. c1 was removed from this list

Immediately, errors were generated:

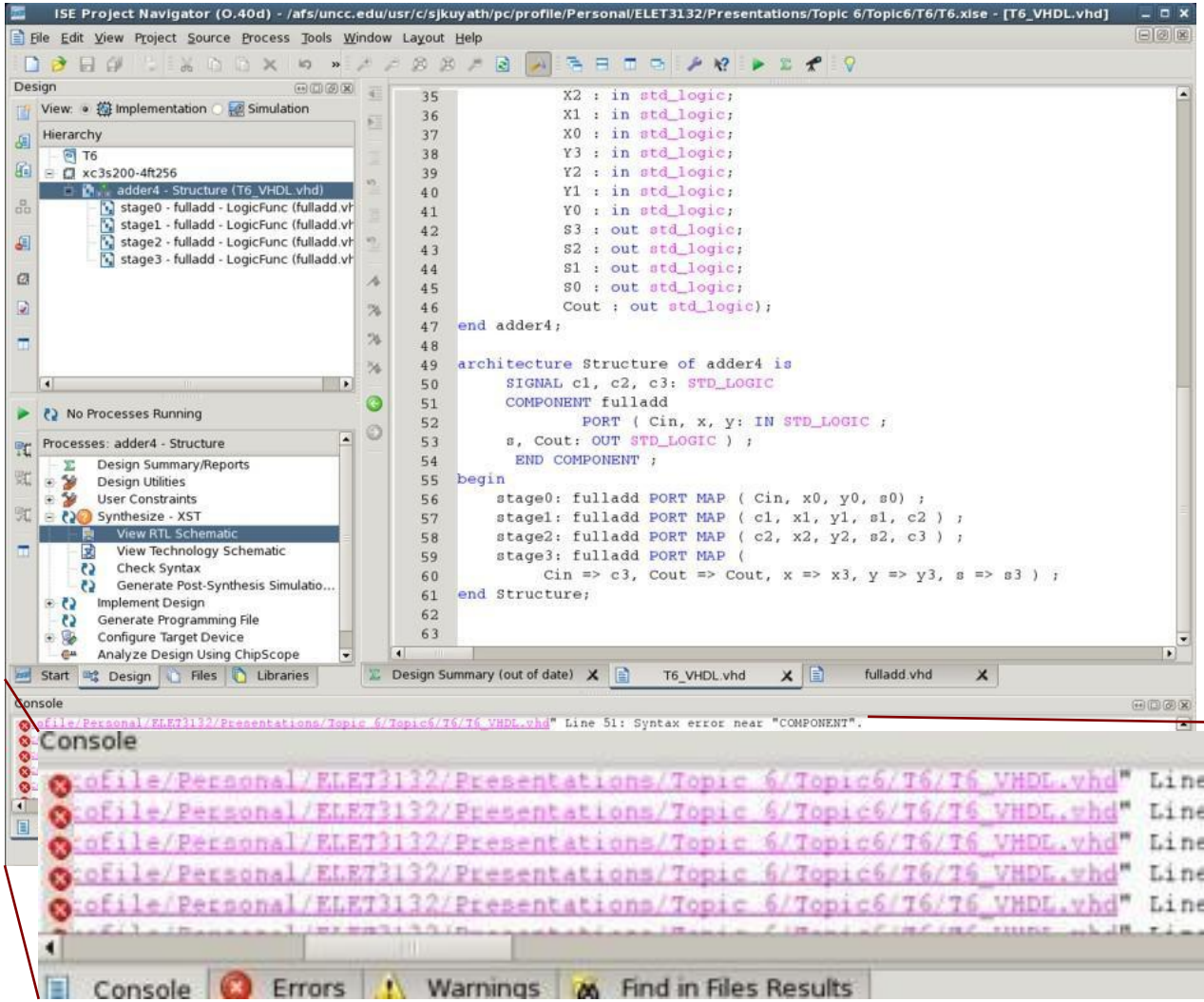



Troubleshooting VHDL Files in ISE



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

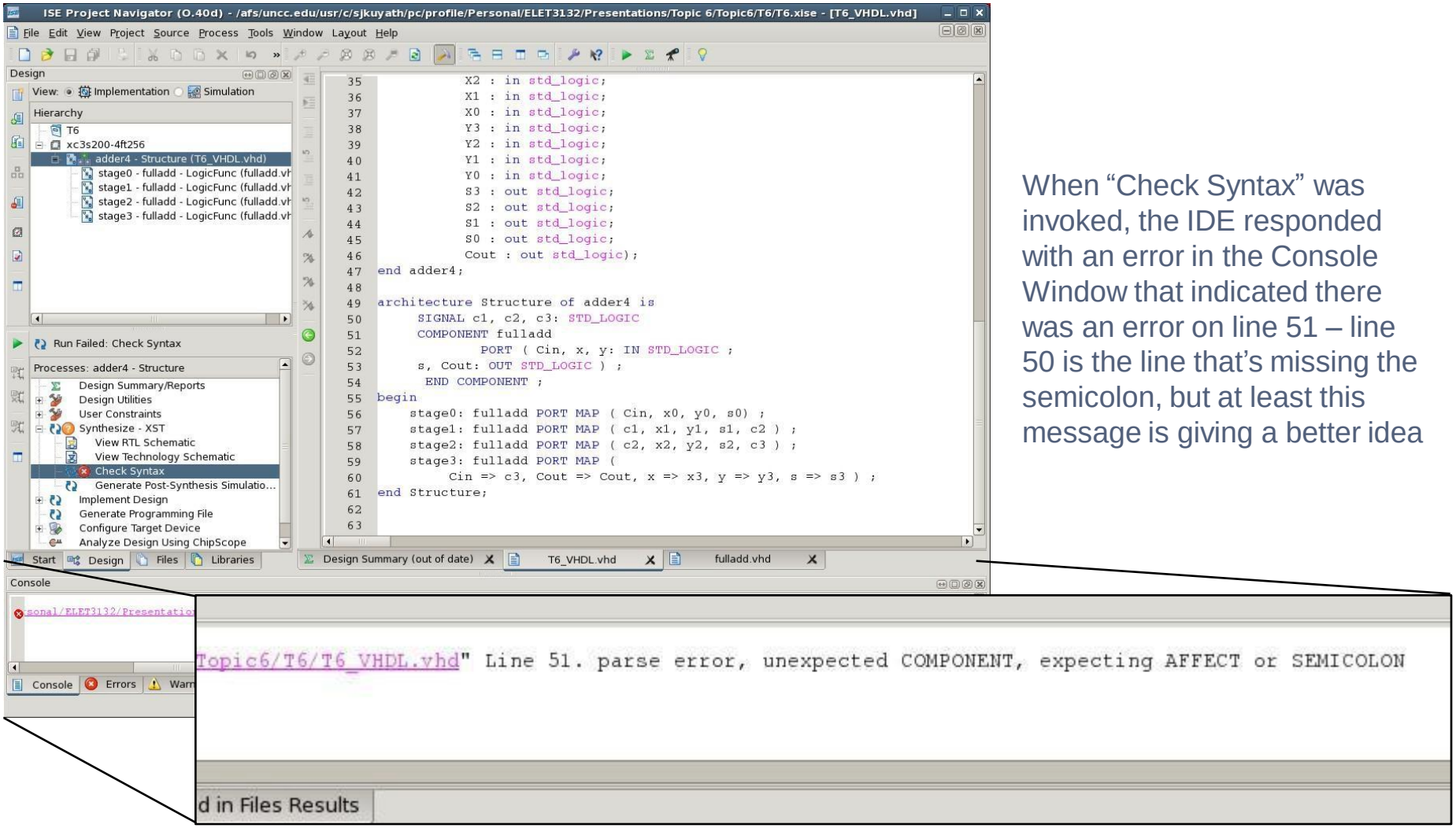
Troubleshooting VHDL Files in ISE



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



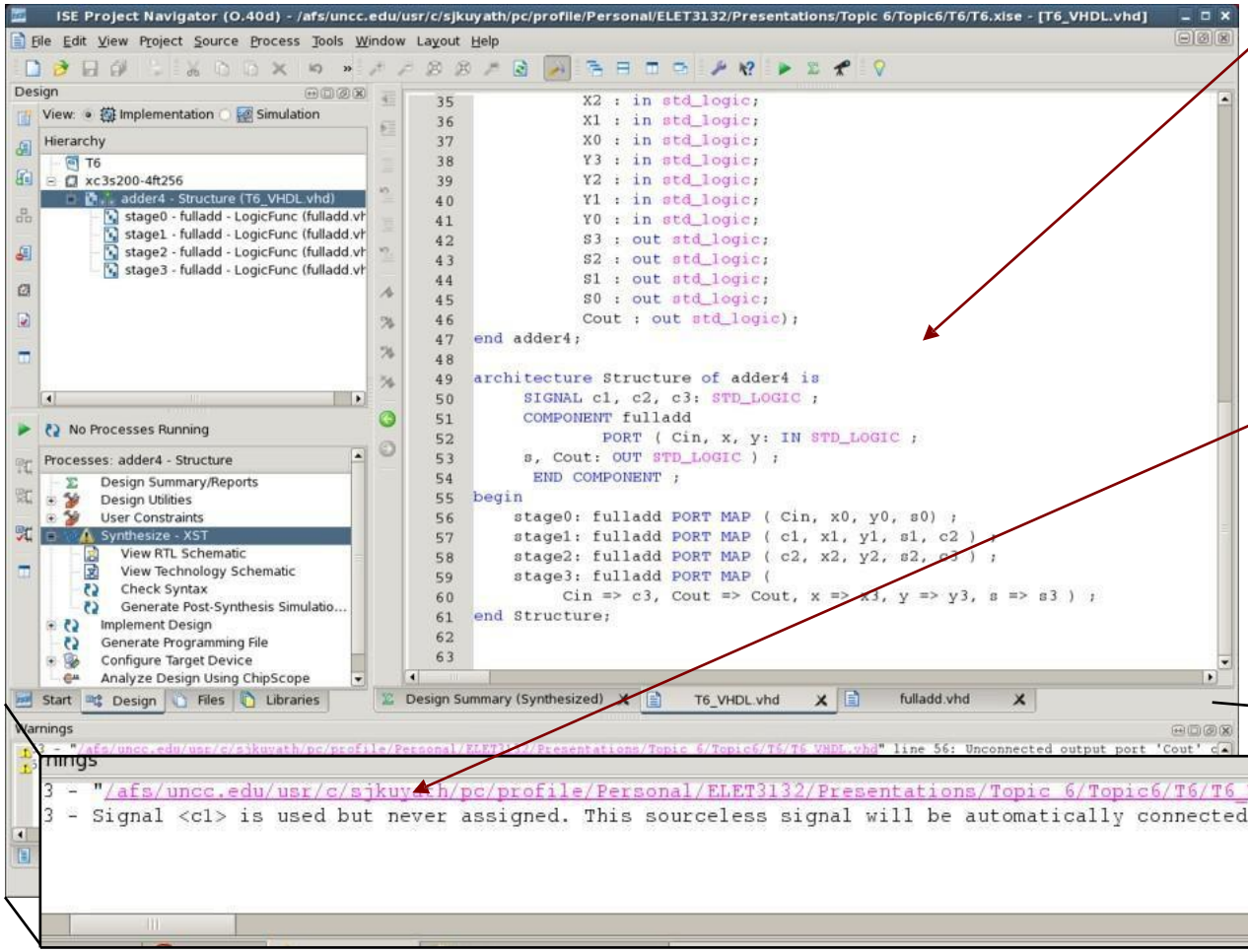

Troubleshooting VHDL Files in ISE



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

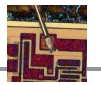


The semicolon was replaced to correct this error.

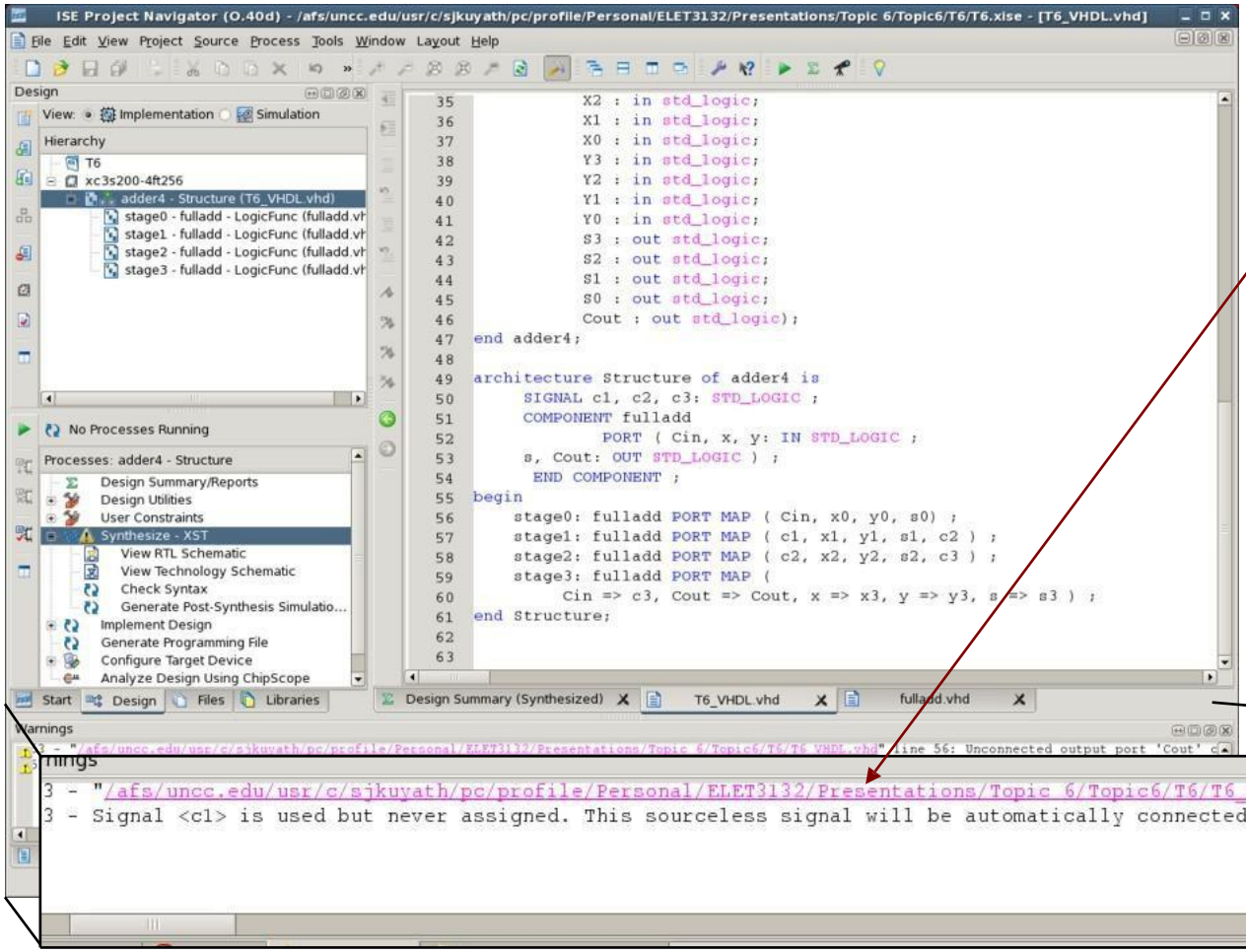
Then …..

"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

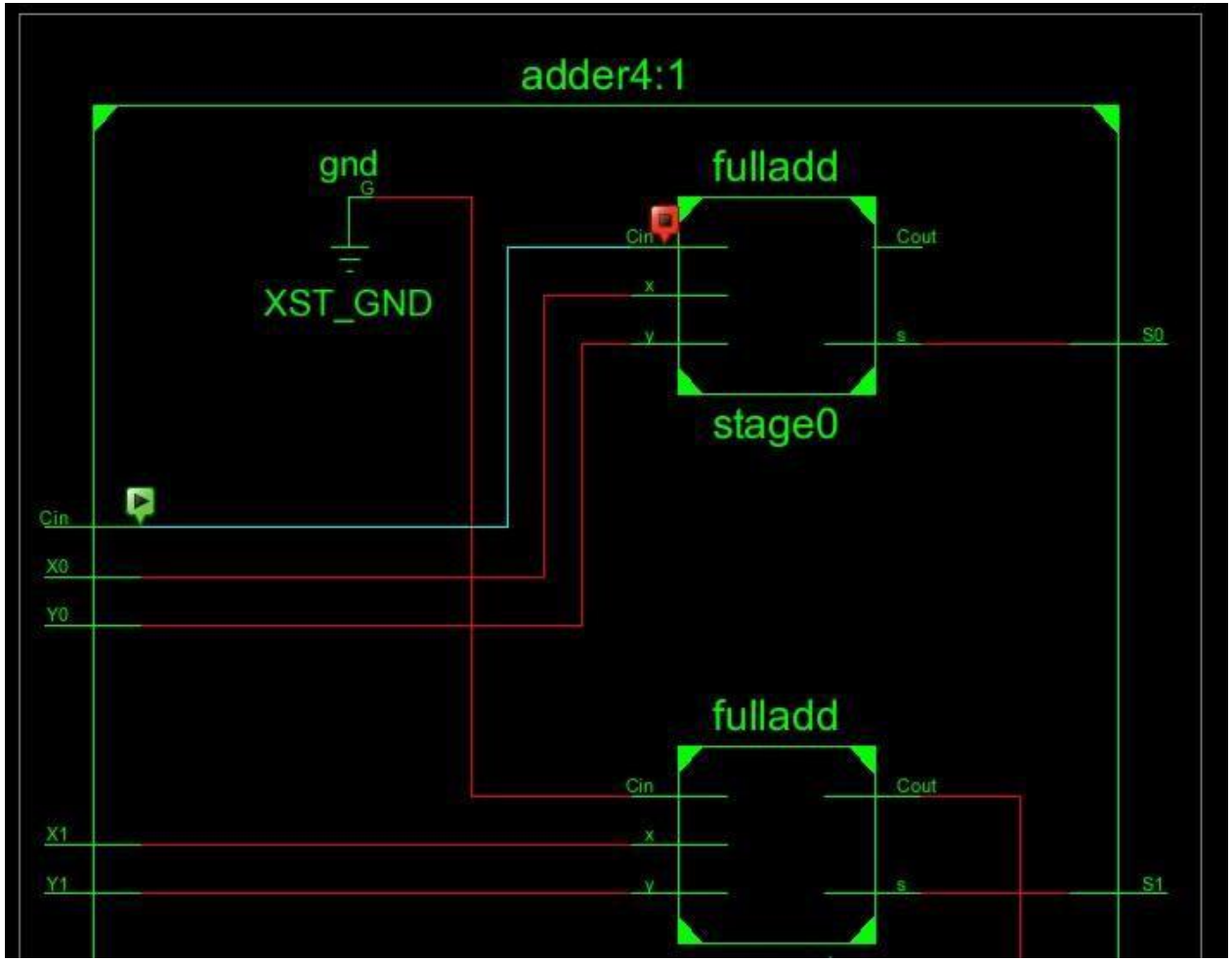


The other warning indicates that Cout was an unconnected port……

hmmmm…. Again, not real helpful




Troubleshooting VHDL Files in ISE



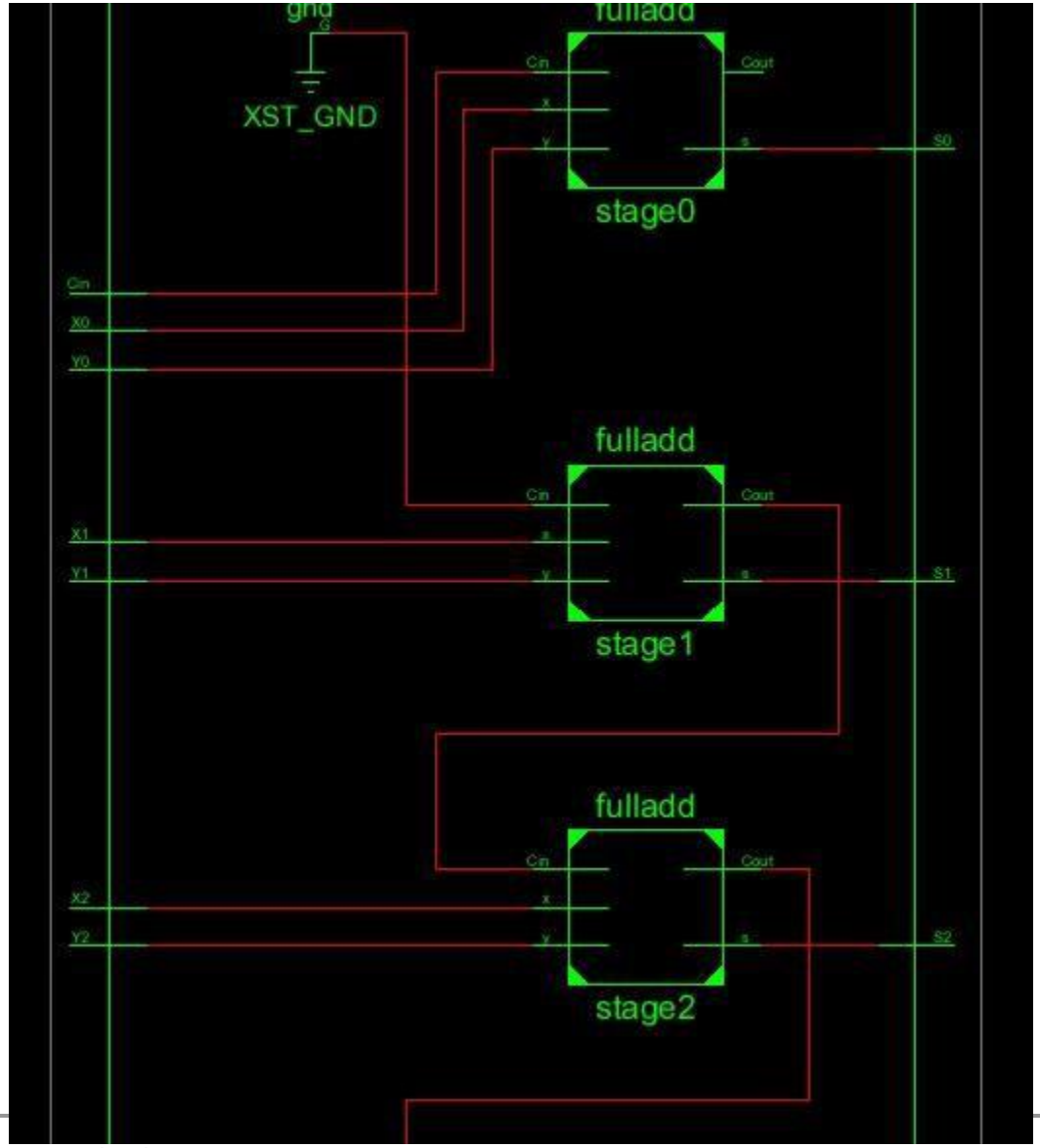
Viewing the RTL might help – clearly this is not correct.

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

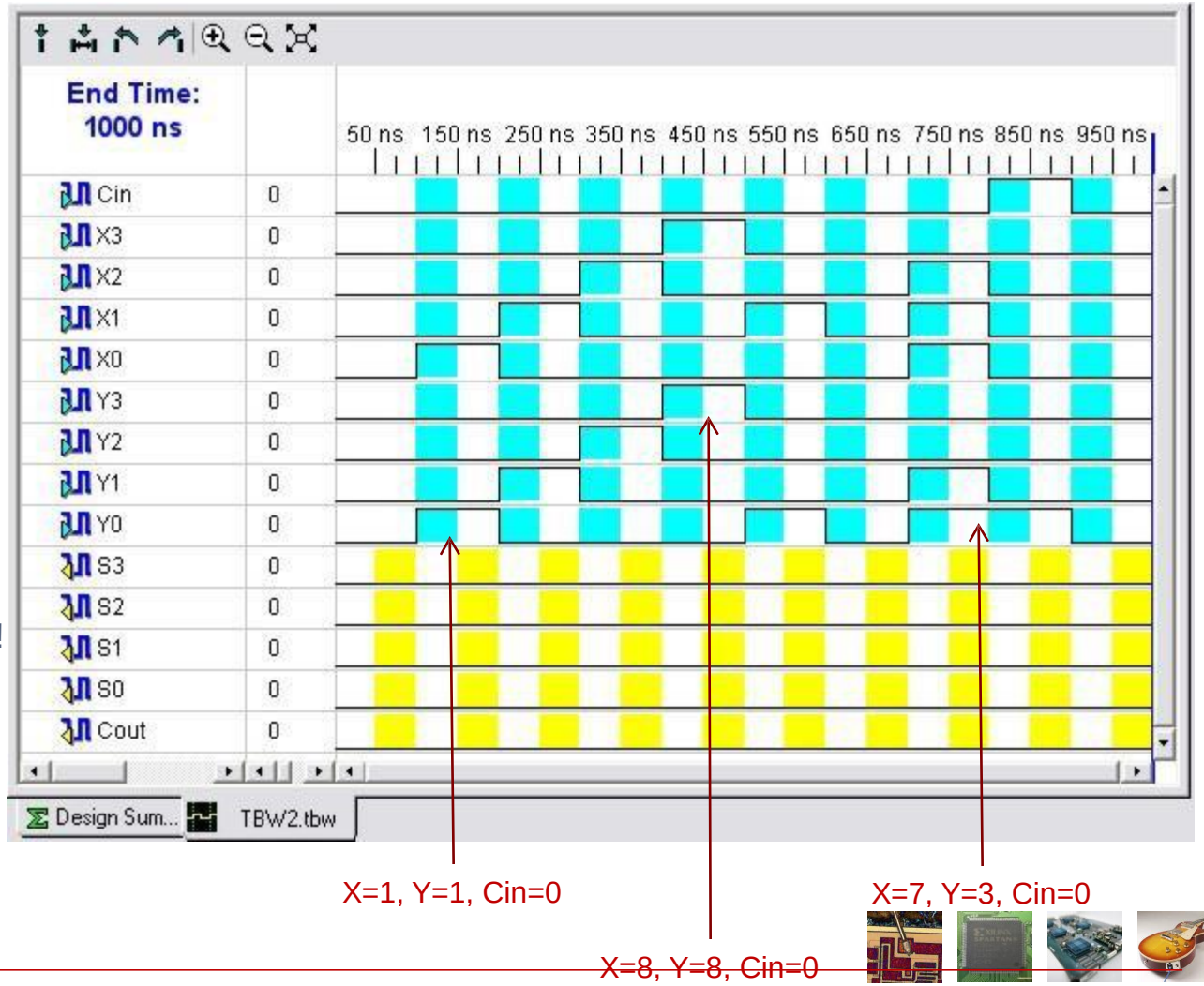
In version ***7.1*** of Xilinx, you could edit waveforms to create a test bench as shown below

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

Before moving on, save the project!!



Testbench Generator/Simulation

No longer.

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

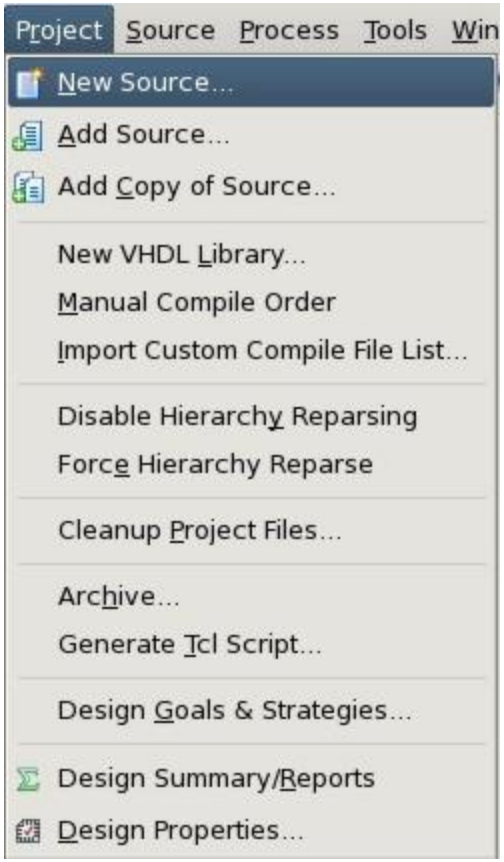
Now we have to create a VHDL file




Testbench Generator/Simulation

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

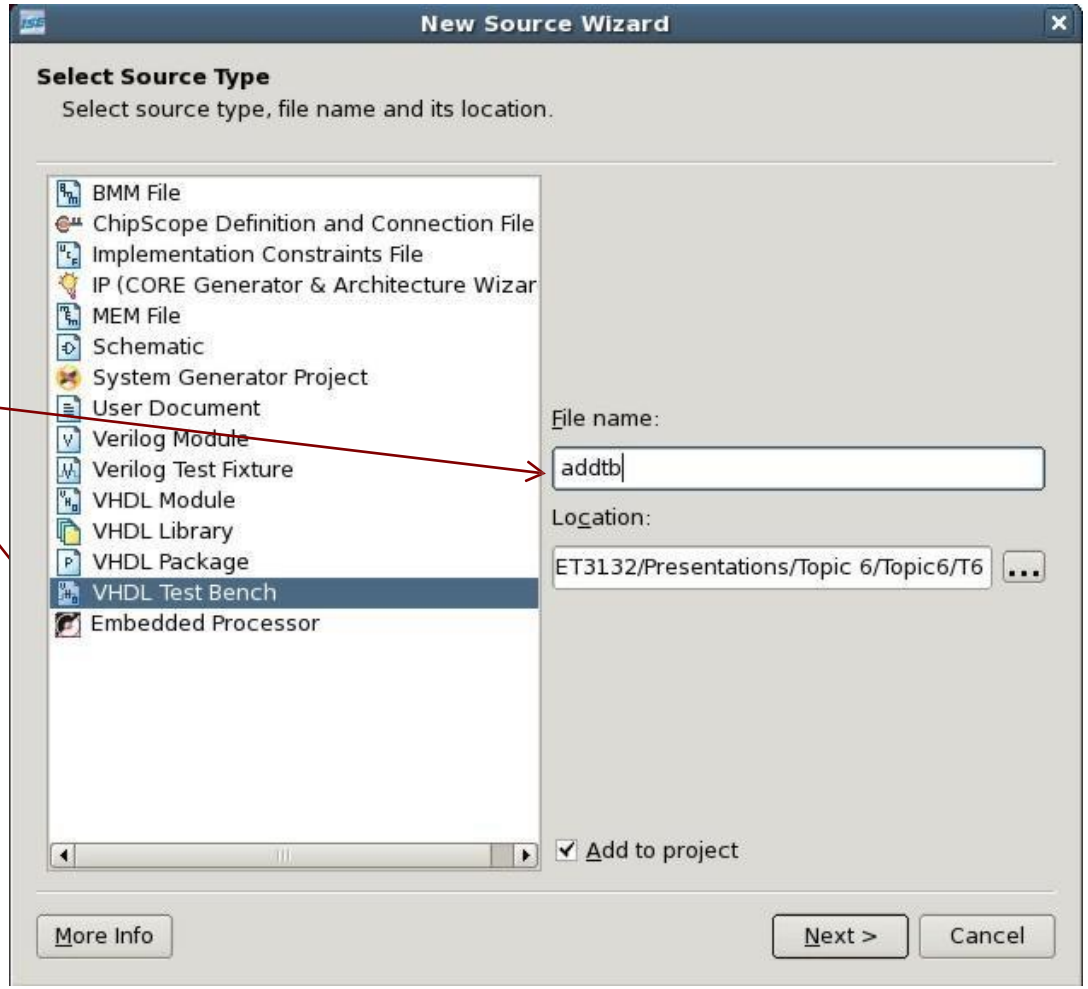
To create a testbench, click on Project and then New Source.



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.

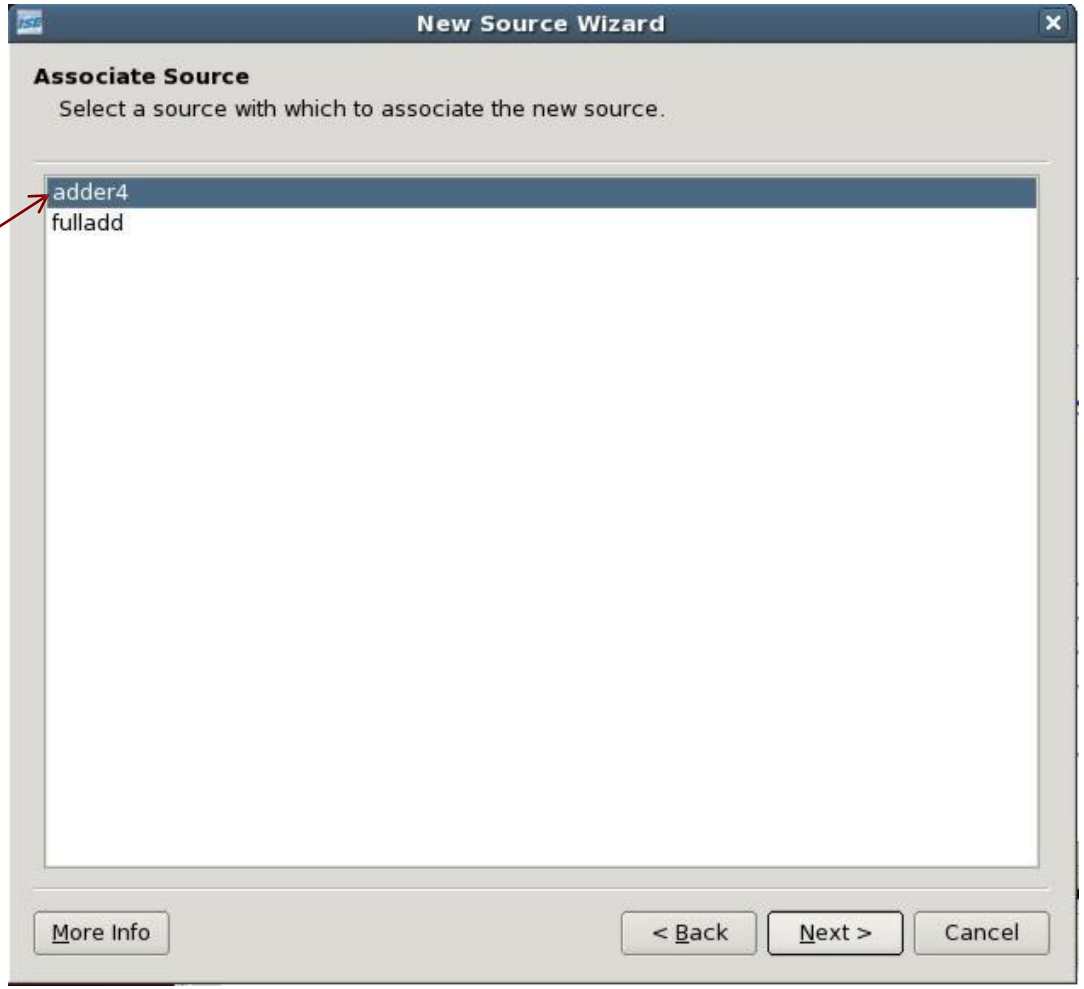


Testbench Generator/Simulation

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

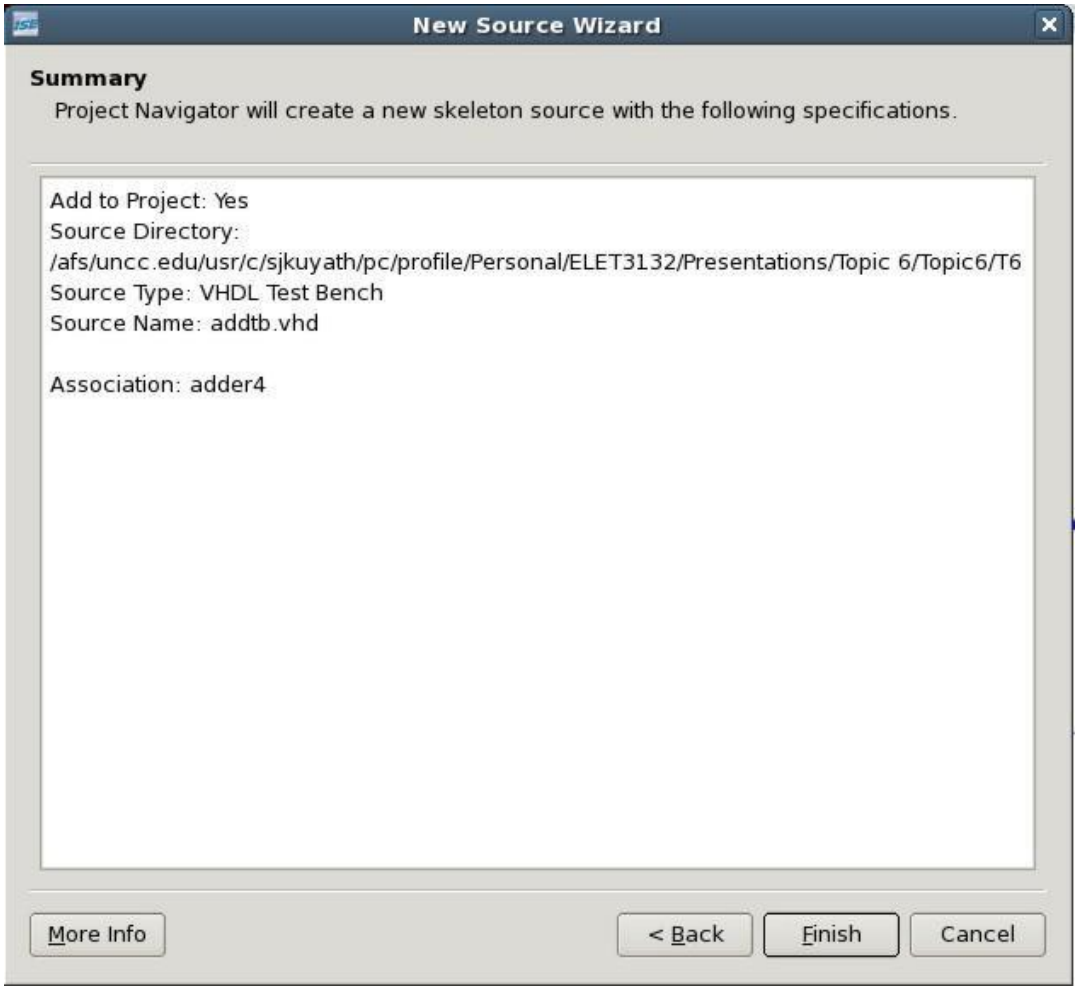
Because I want to simulate the whole design, I will choose the highest level design: “adder4”

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  ----------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date:   13:24:13 09/19/2011
6  -- Design Name:
7  -- Module Name:   /afs/uncc.edu/usr/c/sjkuyath/pc/profile/Personal/ELET
8  -- Project Name:  T6
9  -- Target Device:
10 -- Tool versions:
11 -- Description:
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;
```



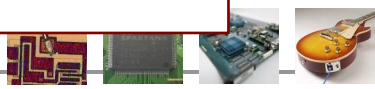

Testbench Generator/Simulation

A basic VHDL file “template” will appear

```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date:    13:24:13 09/
6  -- Design Name:
7  -- Module Name:    /afs/uncc.ed
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 Create
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)



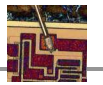

Testbench Generator/Simulation

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
```

```
84  BEGIN
85
86     -- Instantiate the Unit Under Test (UUT)
87     uut: adder4 PORT MAP (
88             Cin => Cin,
89             X3 => X3,
90             X2 => X2,
91             X1 => X1,
92             X0 => X0,
93             Y3 => Y3,
94             Y2 => Y2,
95             Y1 => Y1,
96             Y0 => Y0,
97             S3 => S3,
98             S2 => S2,
99             S1 => S1,
100            S0 => S0,
101            Cout => Cout
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 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
114     -- Stimulus process
115     stim_proc: process
116     begin
117        -- hold reset state for 100 ns.
118        wait for 100 ns;
119
120        --wait for <clock>_period*10;
121
122        -- insert stimulus here
123
124        wait;
125     end process;
126
127  END;
```

```
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




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

```
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;
```

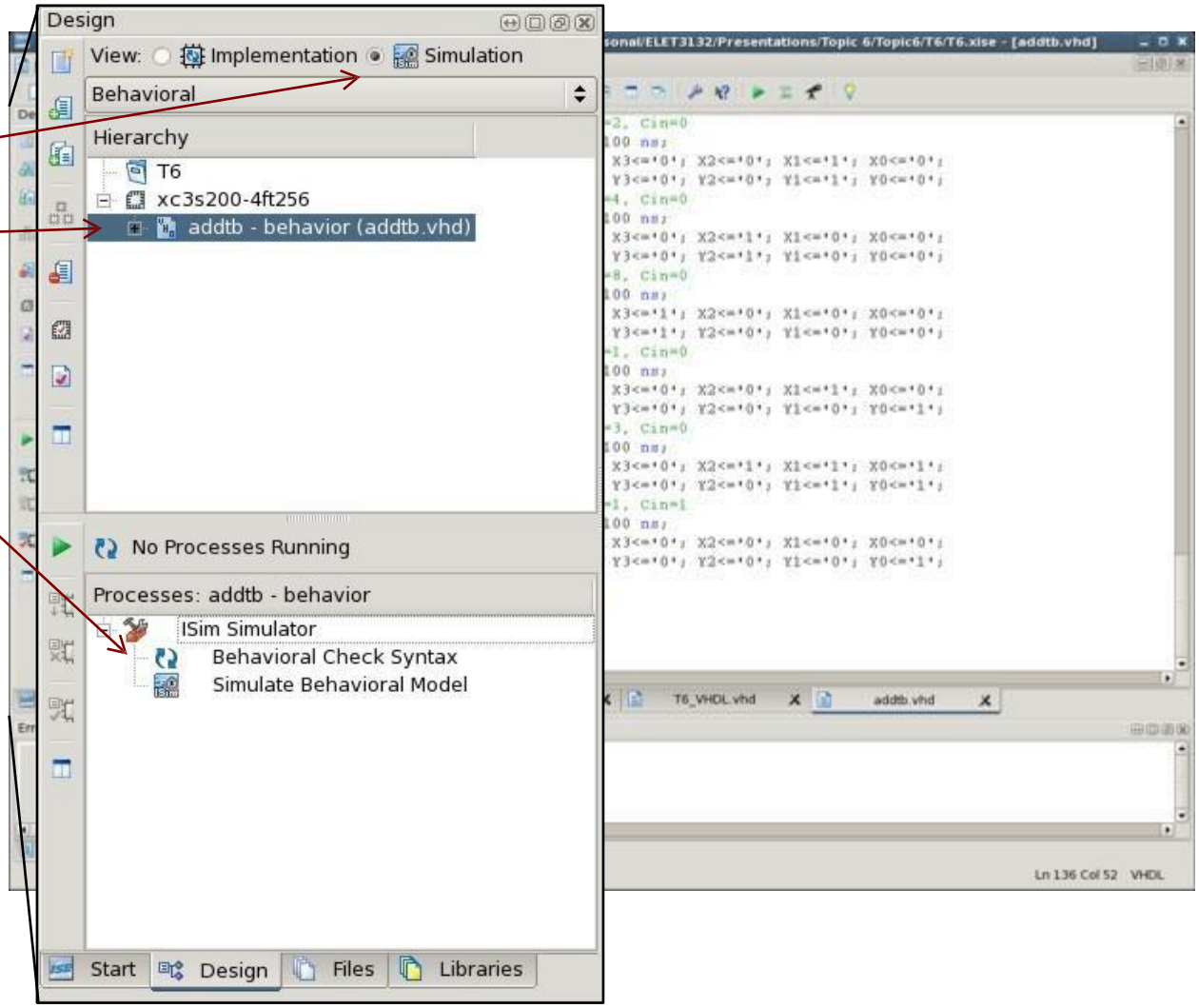



Testbench Generator/Simulation

Save everything

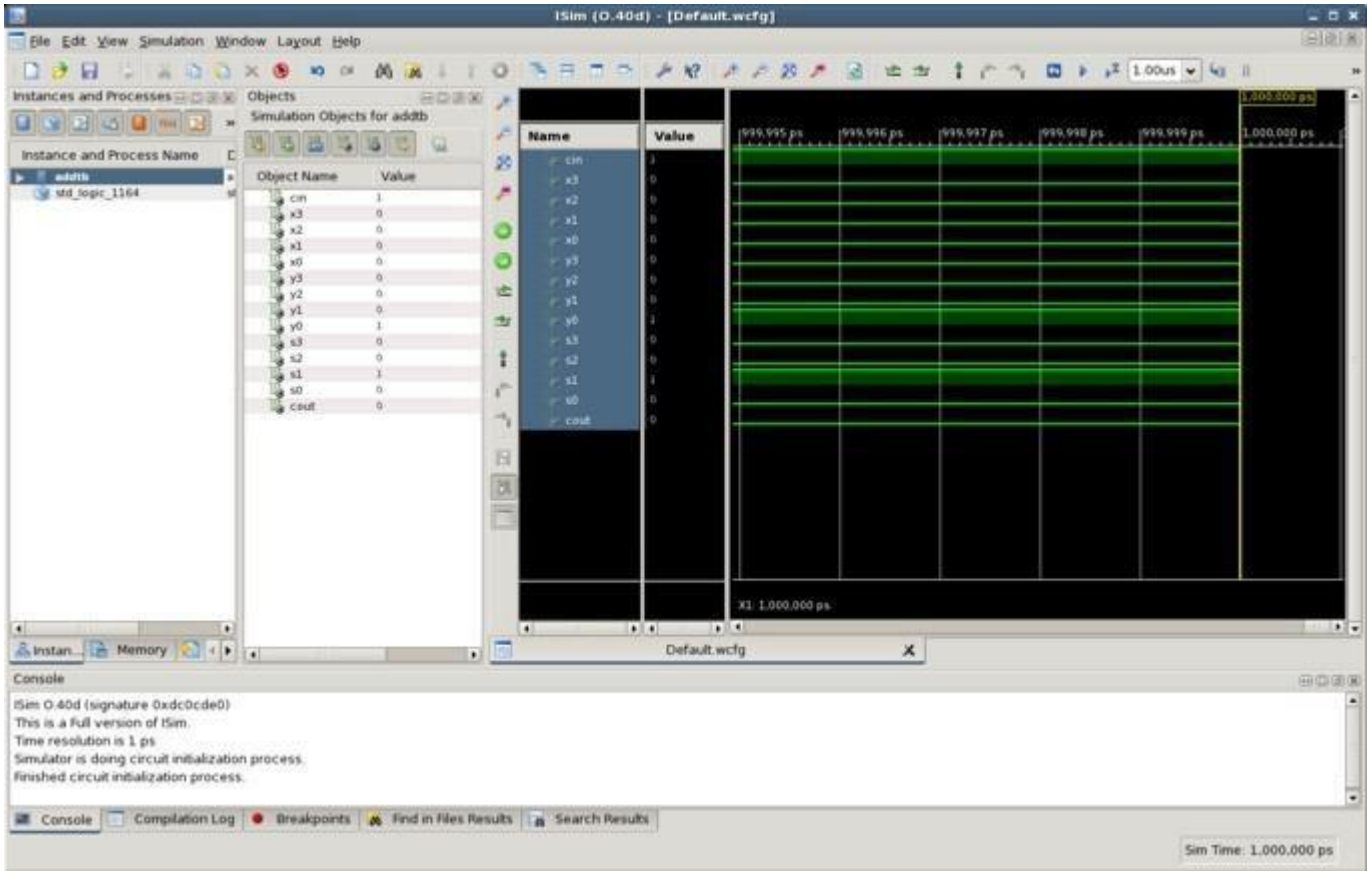
Then click on Simulation

And the test bench and process appear



Testbench Generator/Simulation

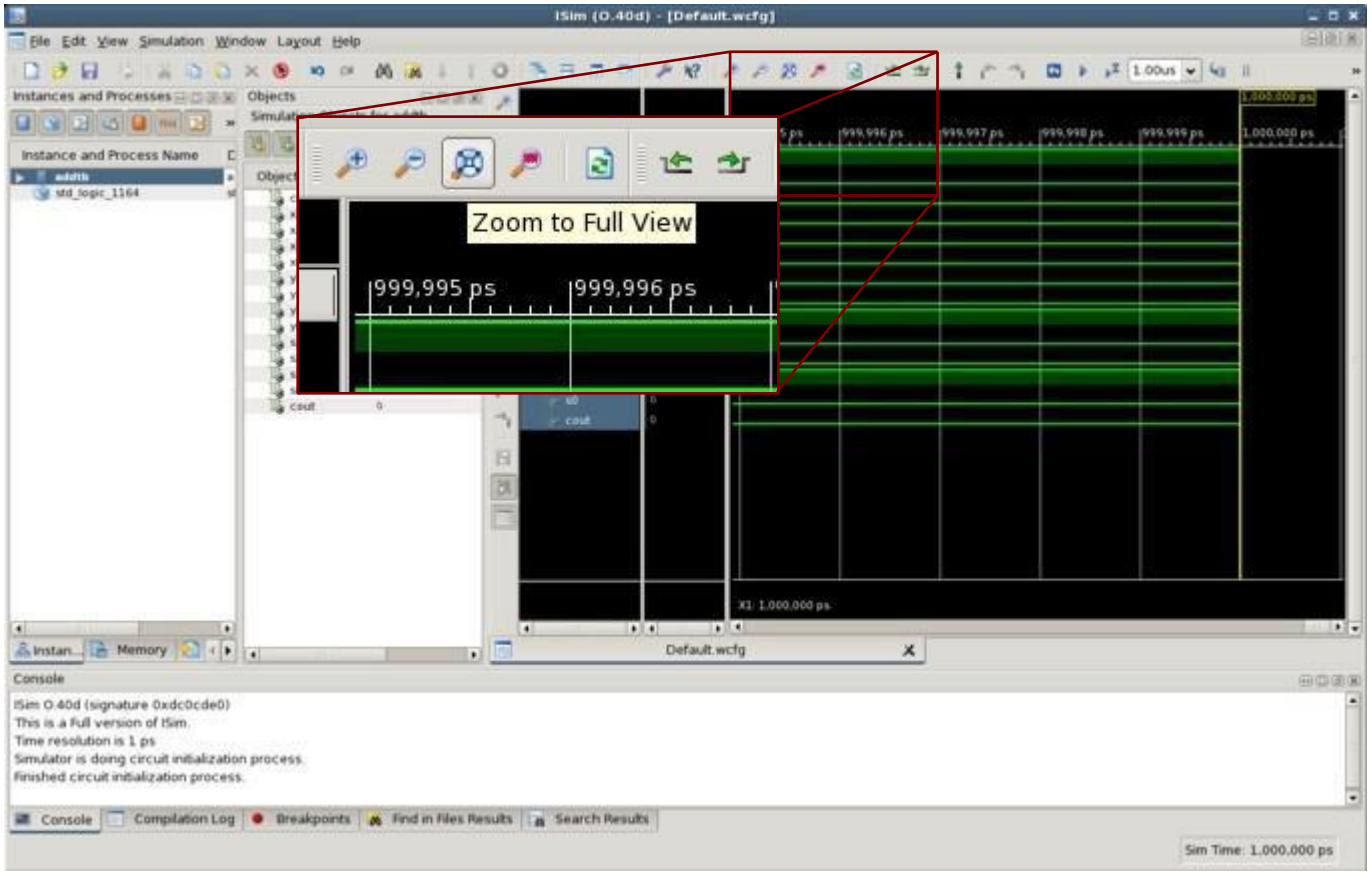
And the ISim window appears



Testbench Generator/Simulation

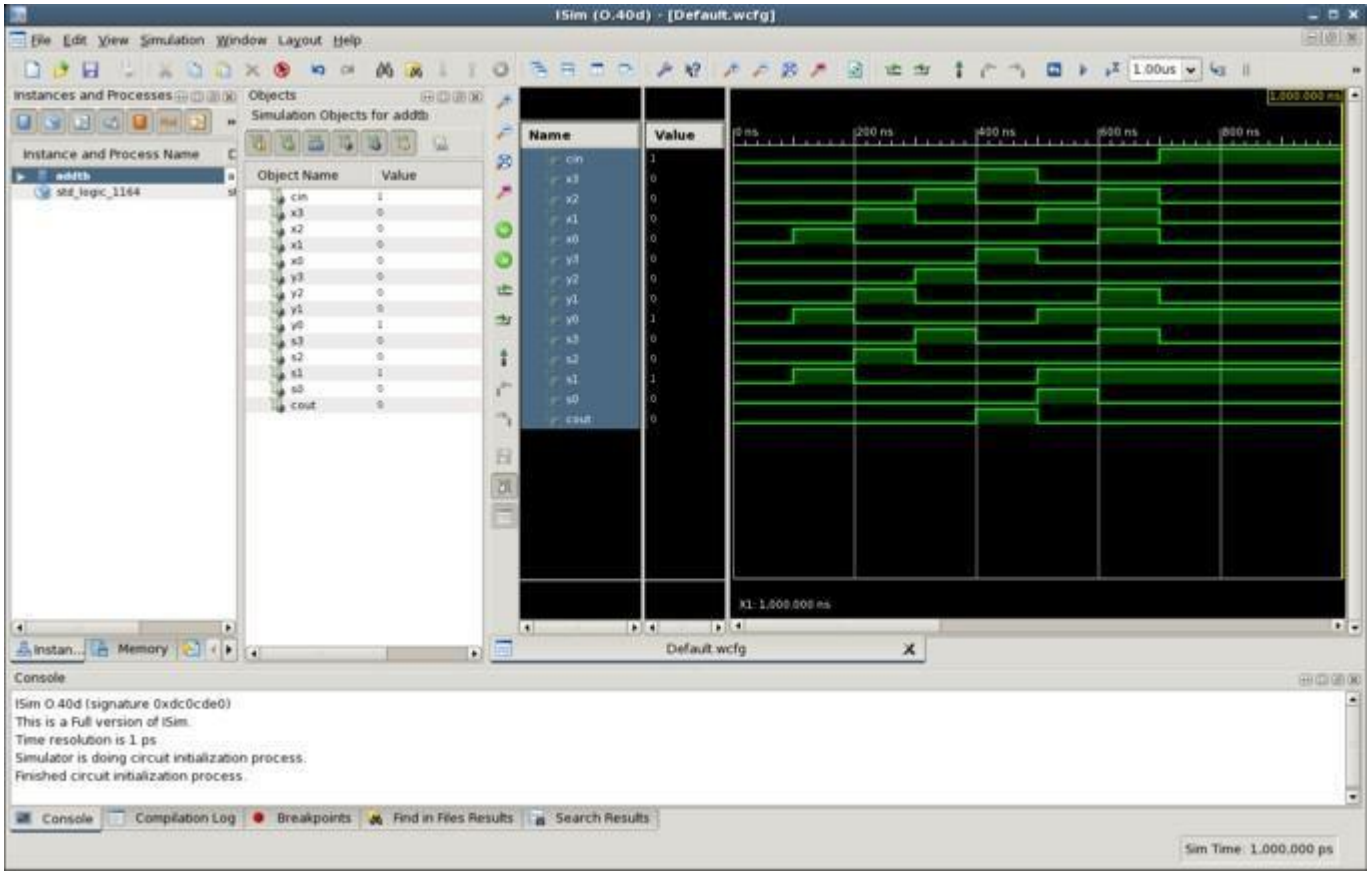
And the ISim window appears

Zoom to full view



Testbench Generator/Simulation

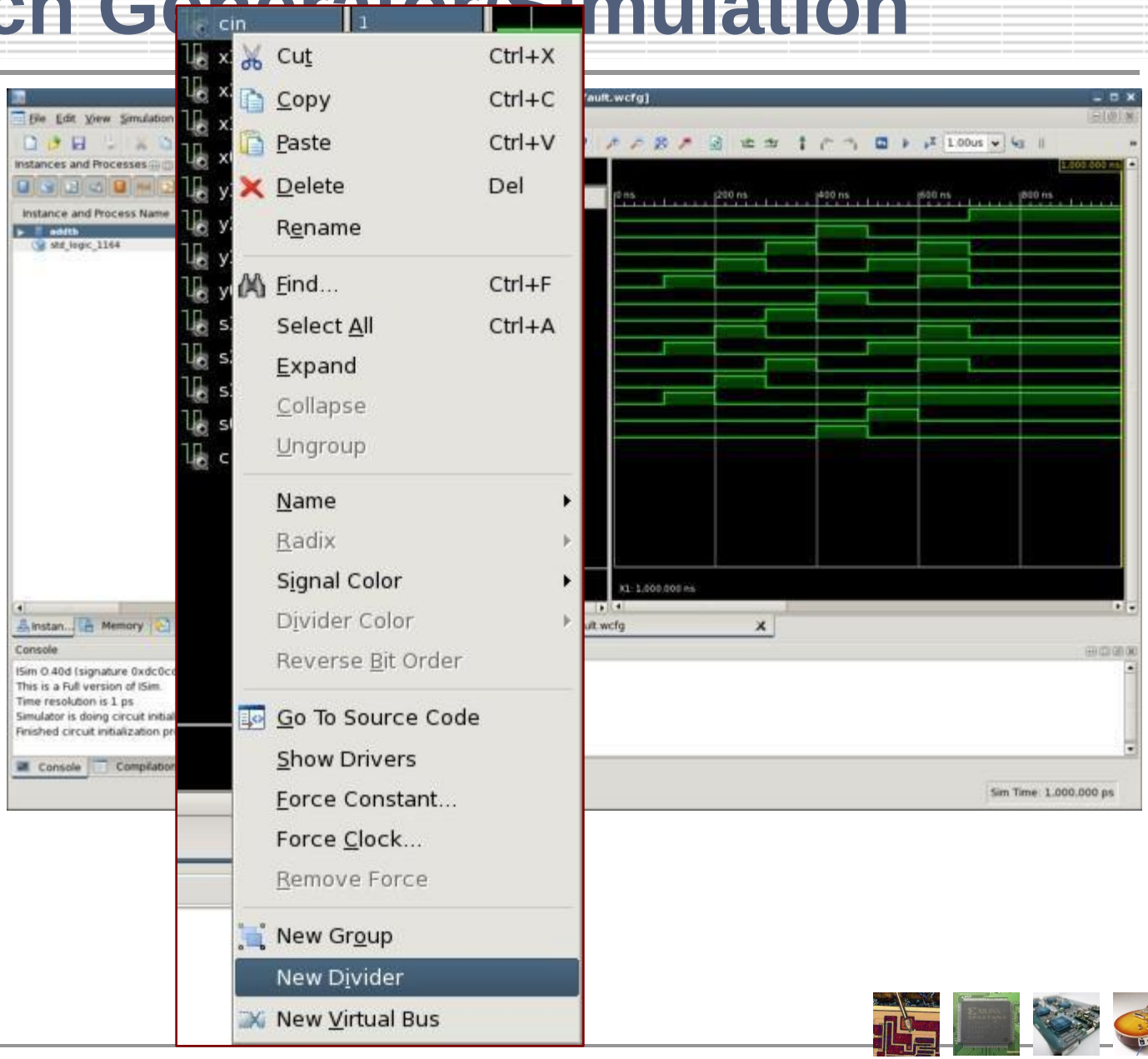
And this view come up, but it's hard to see with everything packed together.



Testbench Generator/Simulation

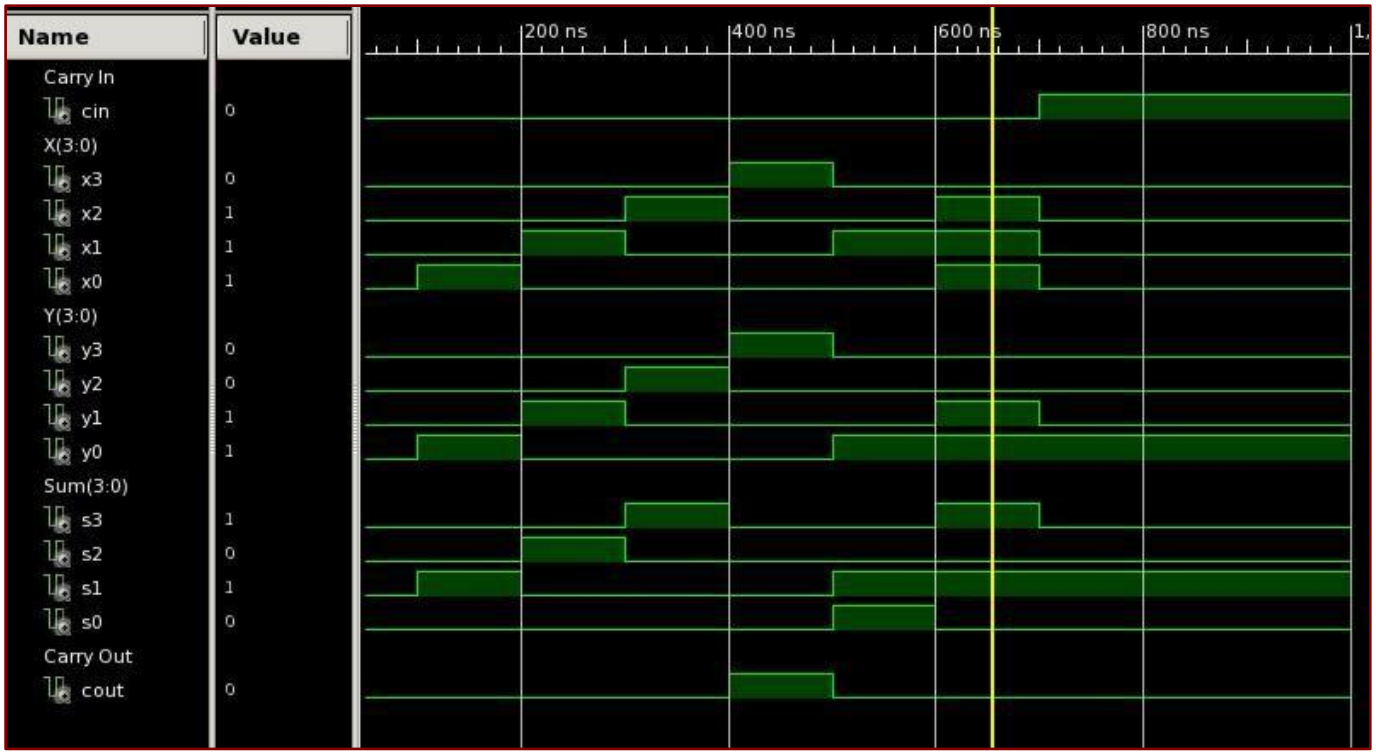
And this view come up, but it's hard to see with everything packed together.

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



Testbench Generator/Simulation

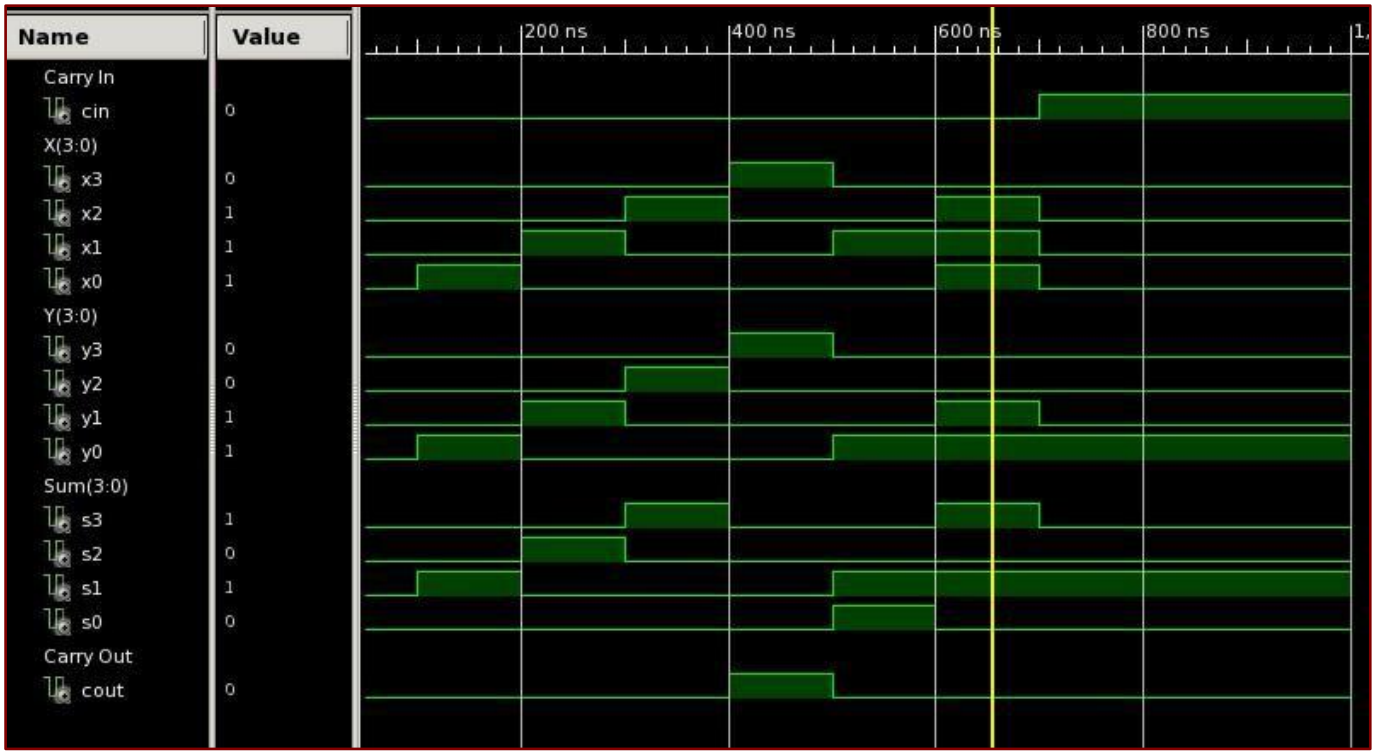
With the new dividers (I also colored them black – an option on the pop up menu), it's easier to see the different states of the test.



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).

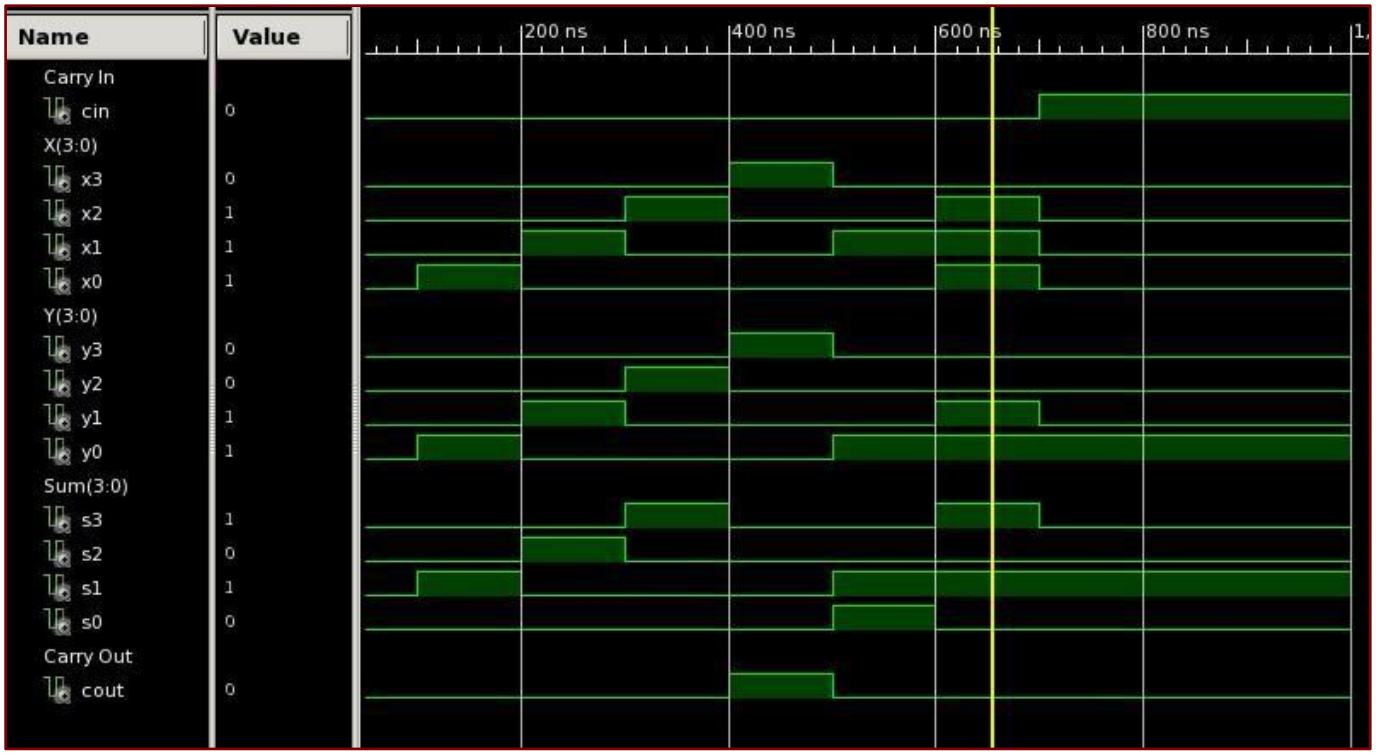


Testbench Generator/Simulation

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



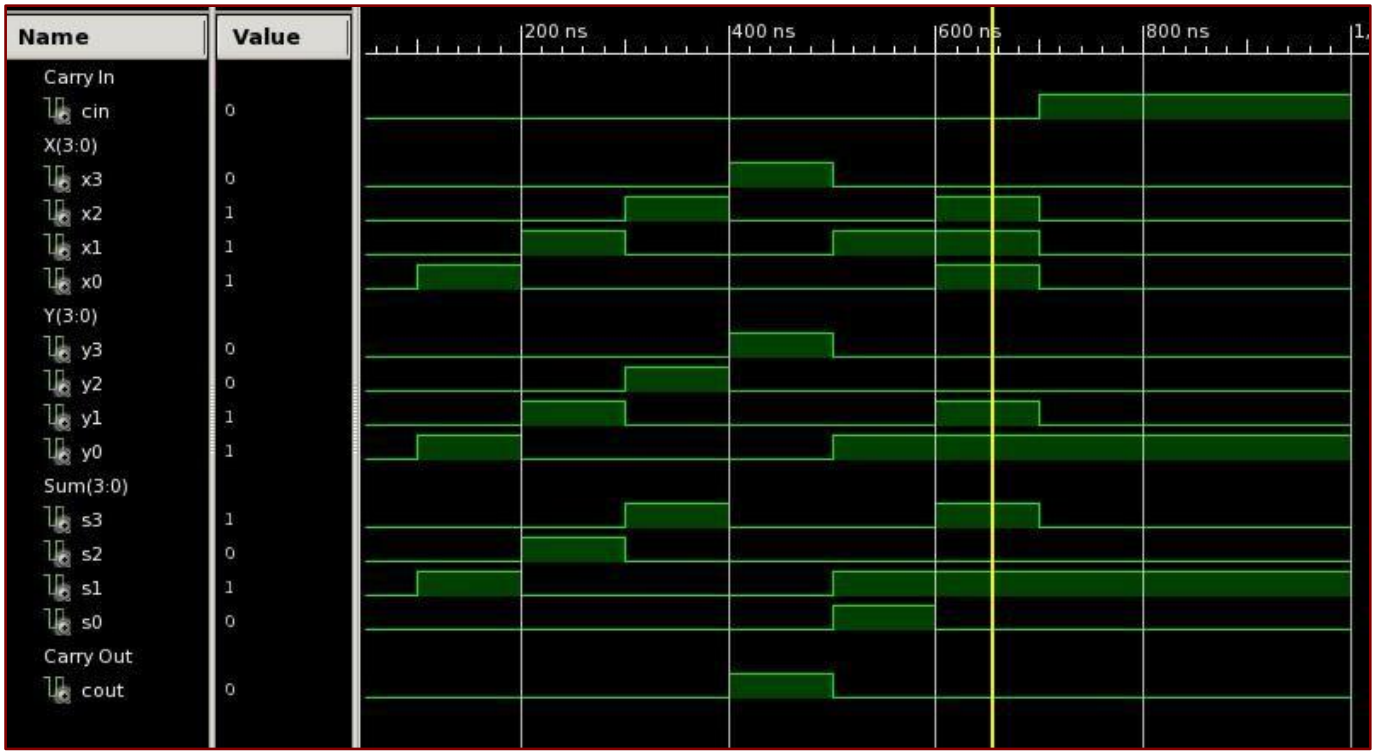
Testbench Generator/Simulation

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

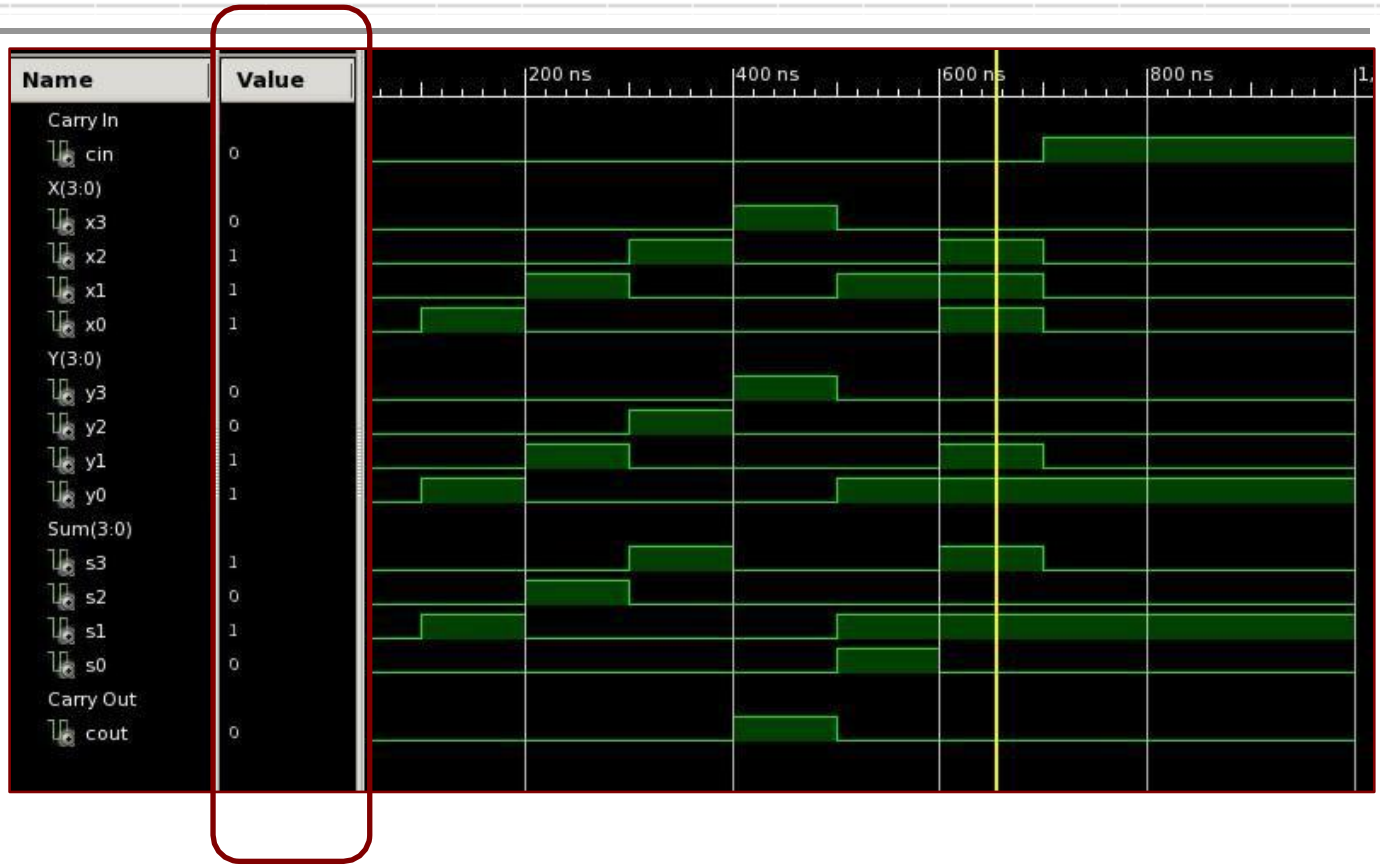
The marker is showing the 6th division.

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

and the sum is 10 with no carry out

S = 10 (s=1010)

This also chows up in the “Value” column



Testbench Generator/Simulation

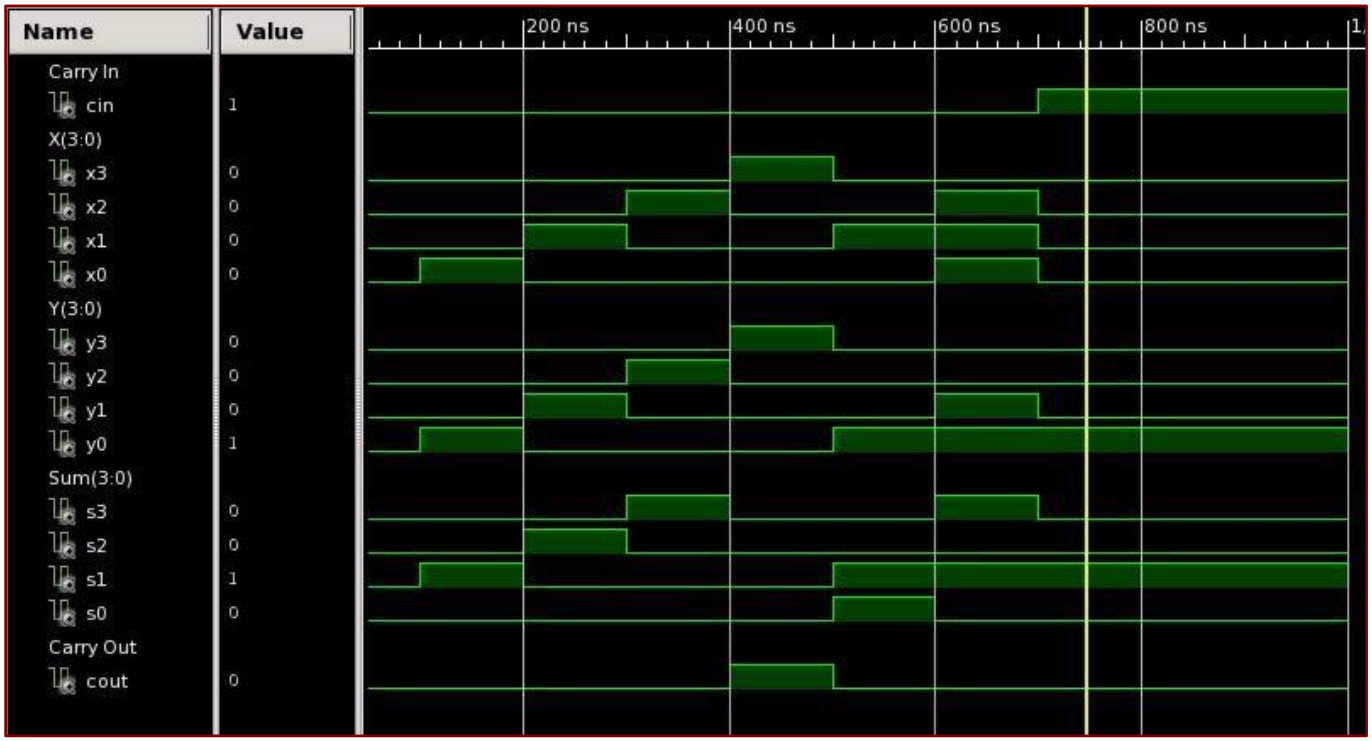
The marker is now showing the 7th division.

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 chows 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.

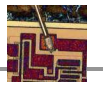
These names must not be changed!!!

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

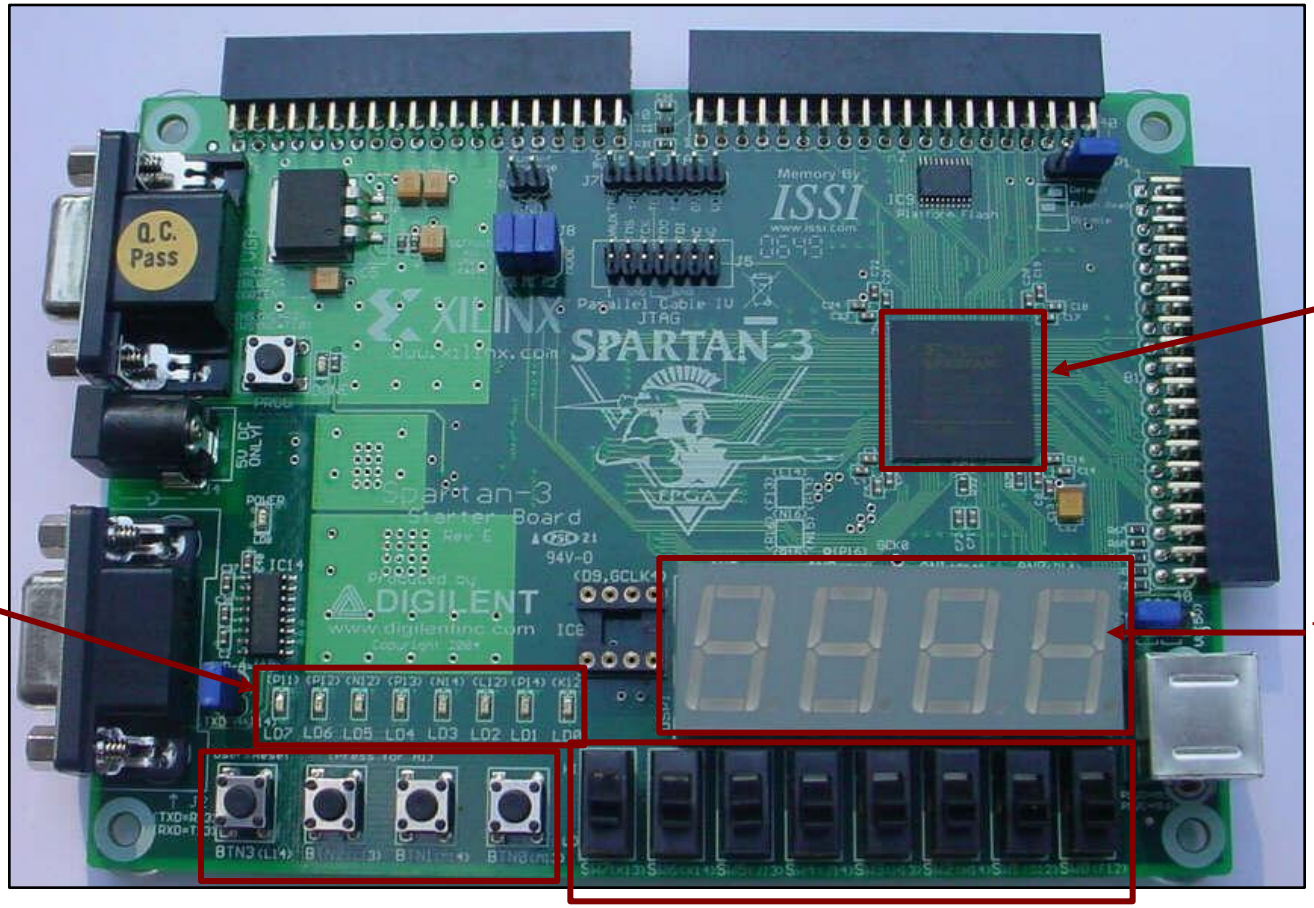
```
# segments on the 7 seg displays
# NET "ssg<0>" LOC = "E14" ;
# NET "ssg<1>" LOC = "G13" ;
# NET "ssg<2>" LOC = "N15" ;
# NET "ssg<3>" LOC = "P15" ;
# NET "ssg<4>" LOC = "R16" ;
# NET "ssg<5>" LOC = "F13" ;
# NET "ssg<6>" LOC = "N16" ;
# NET "ssg<7>" LOC = "P16" ;

# 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

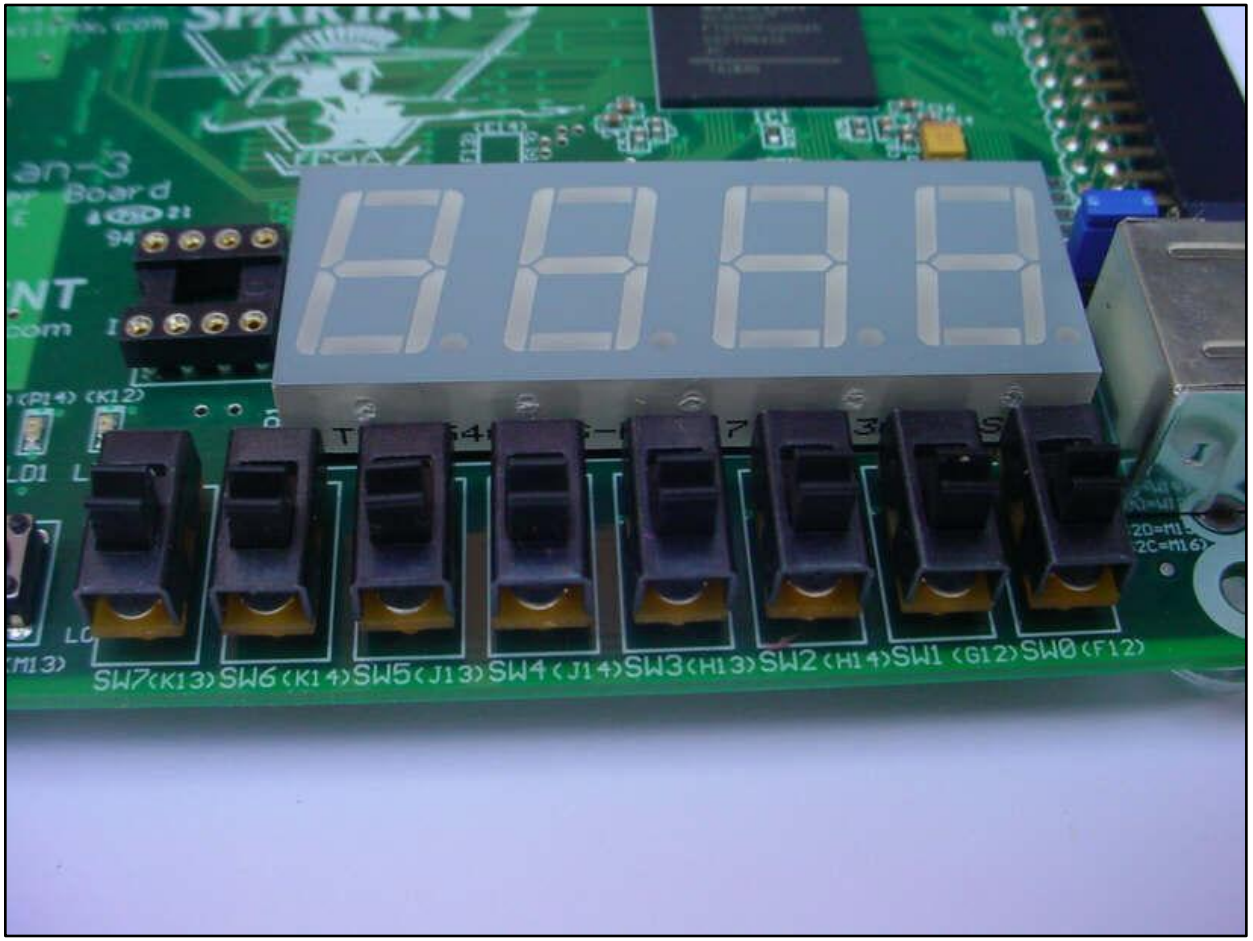


Spartan3 LED Close Up

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

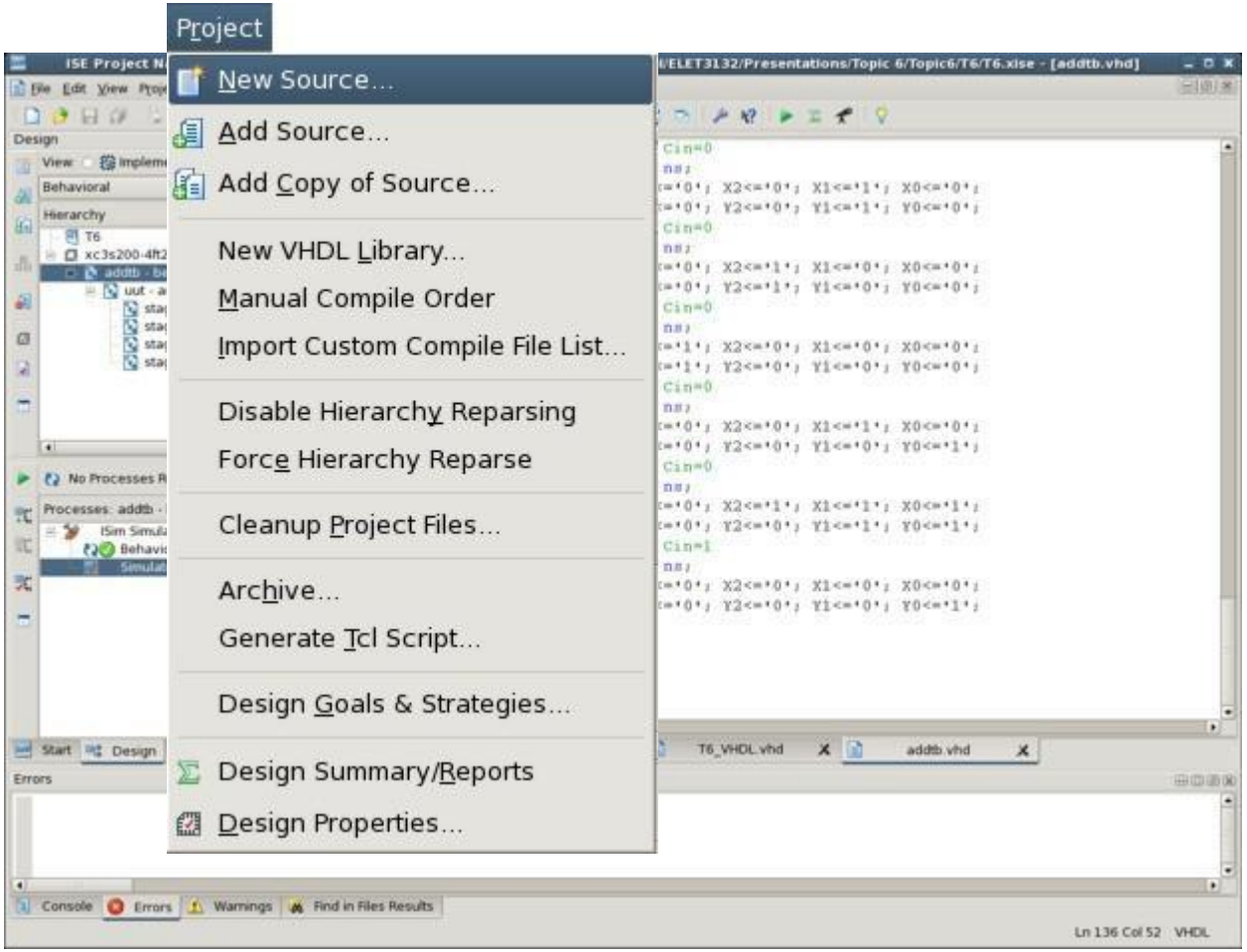
Y0 = SW4
Y1 = SW5
Y2 = SW6
Y3 = SW7

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

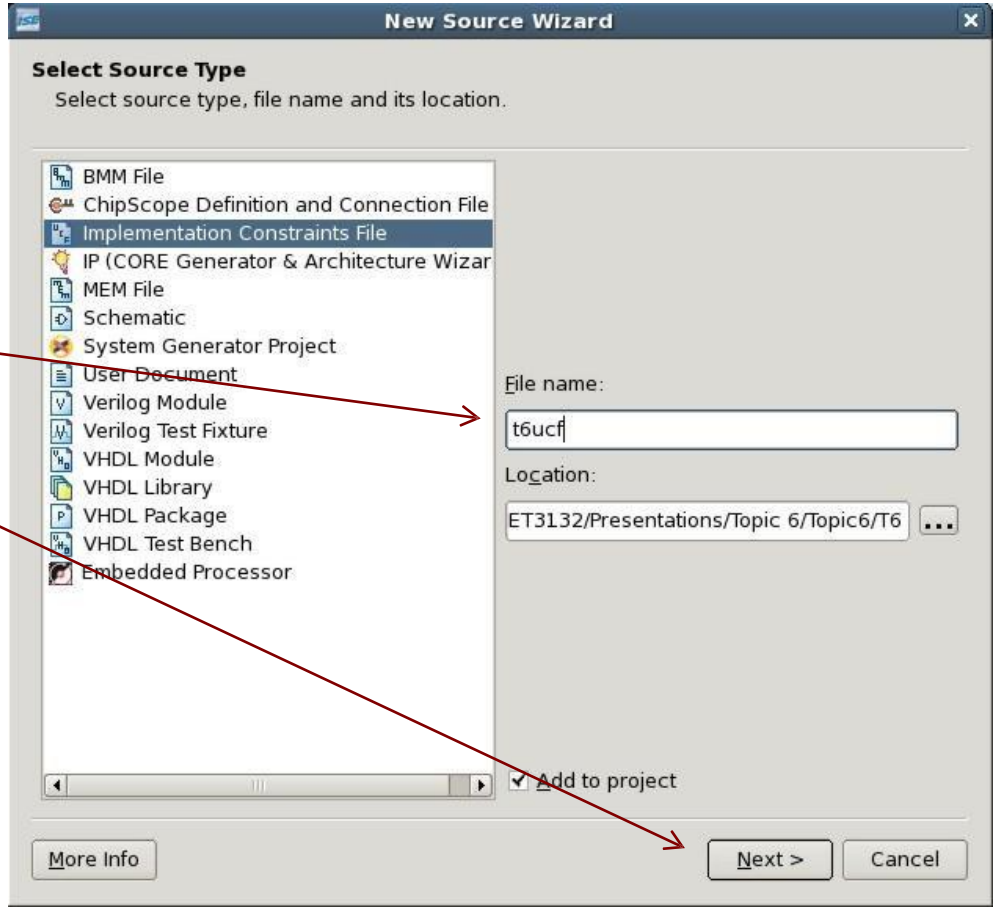


Add Constraints File

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



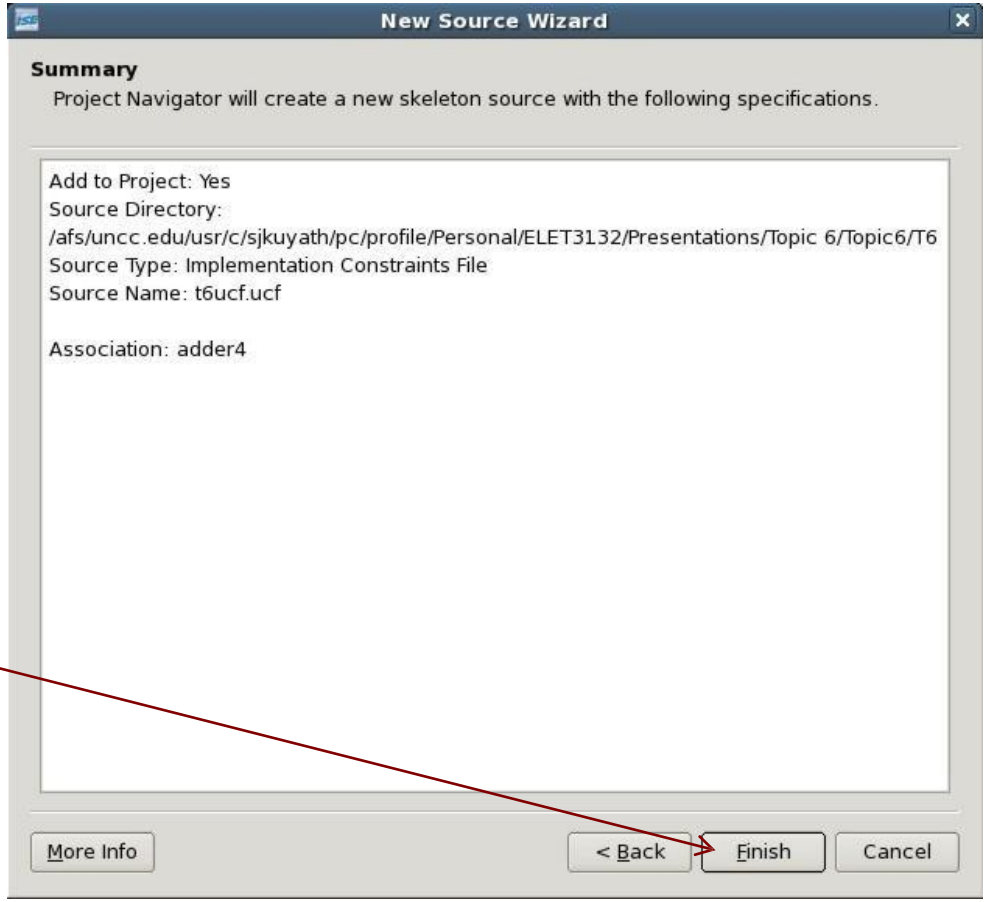
Add Constraints File

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

The New Source Wizard appears.

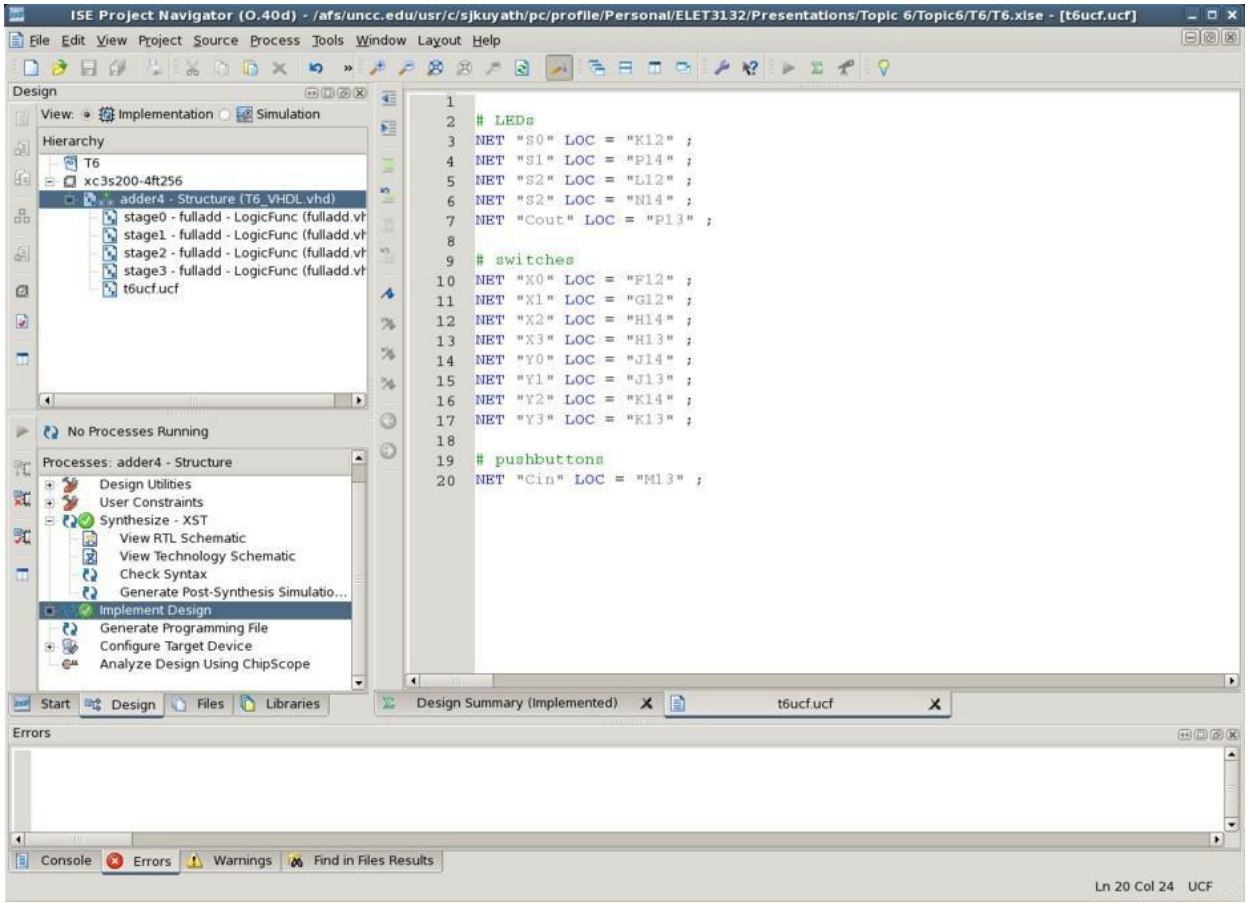
Type in the name of your UCF file, and click Next.

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



Add Constraints File

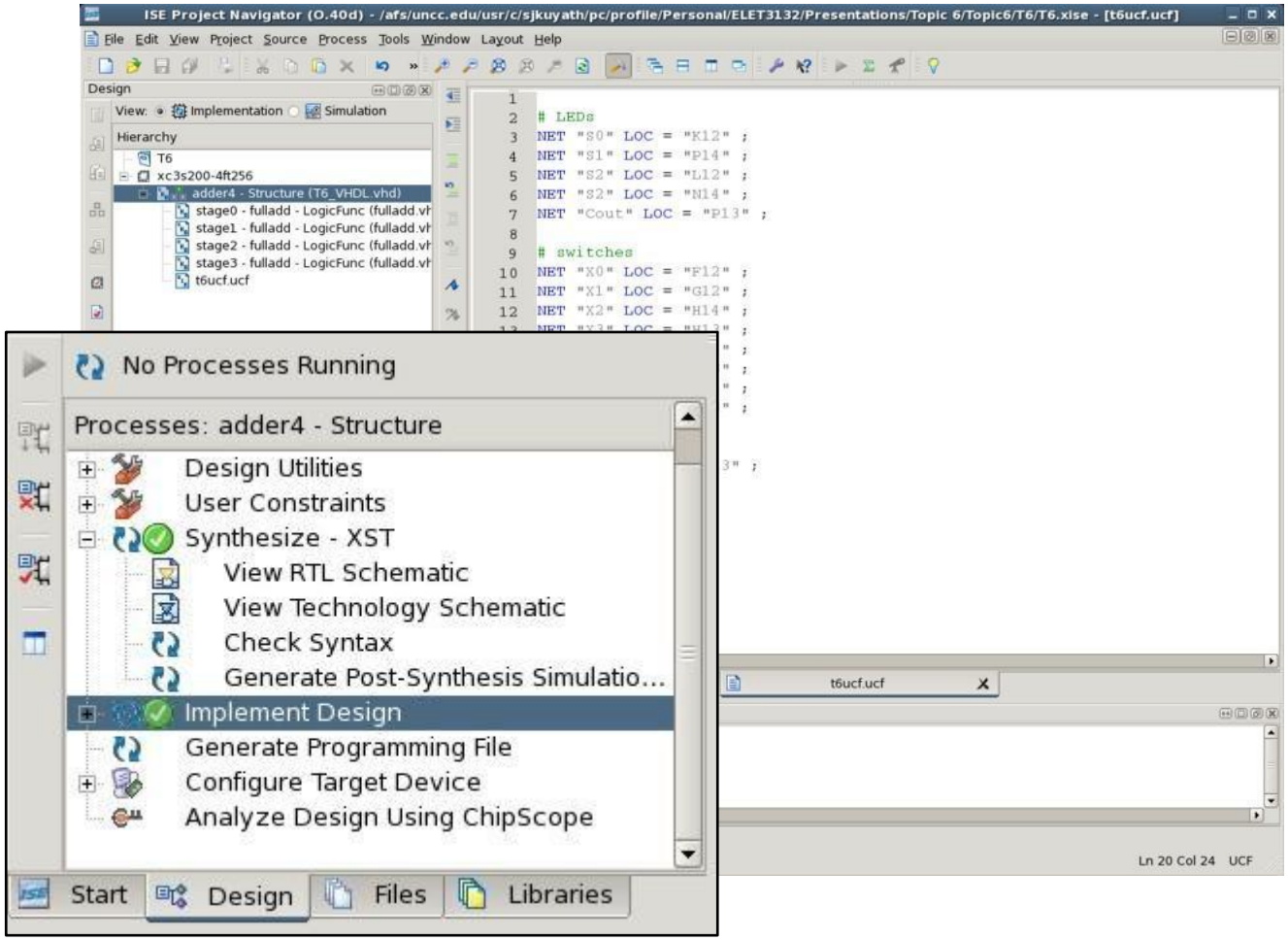
Type in or copy-paste the UCF file



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

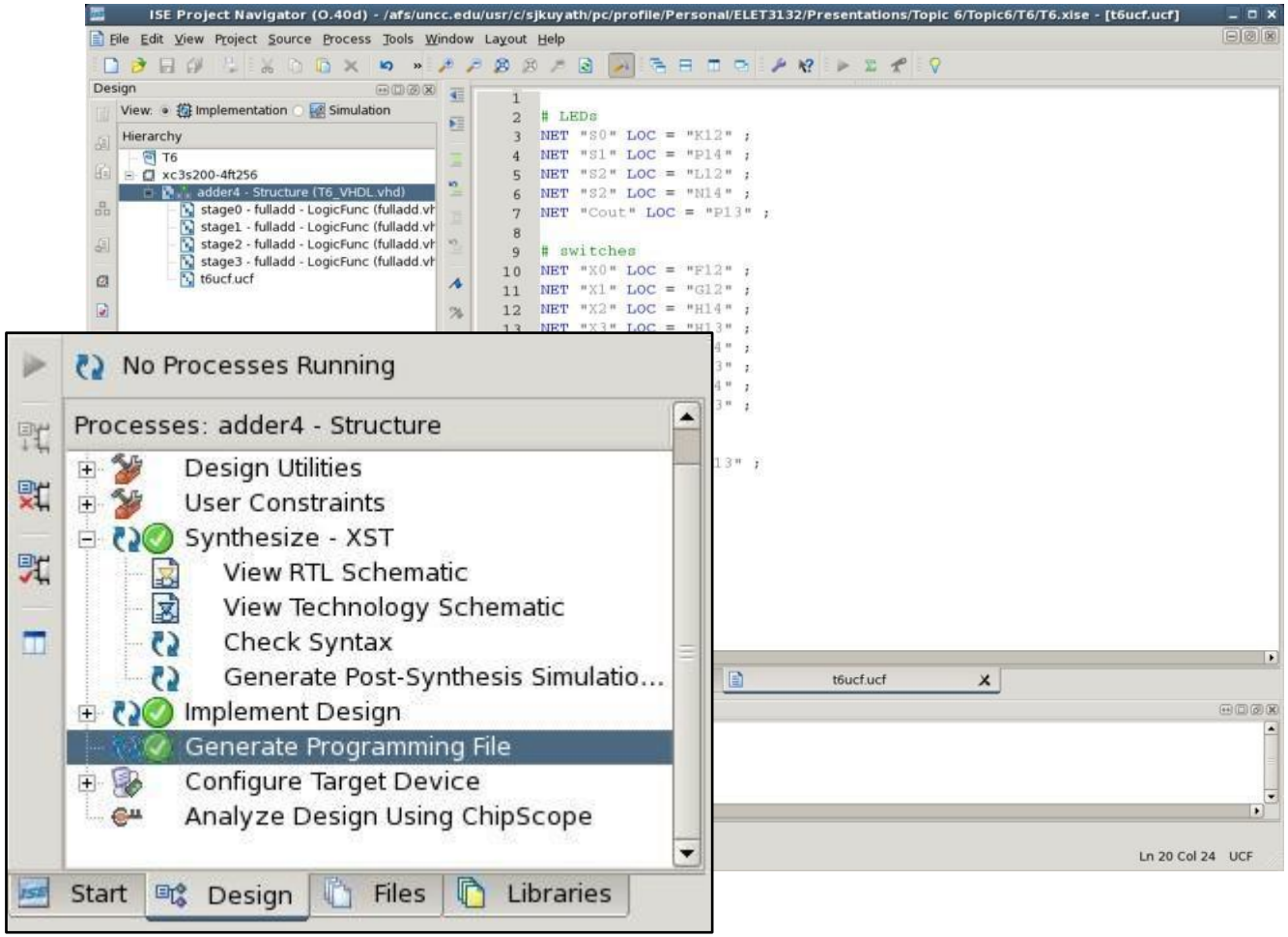
Type in or copy-paste the UCF file

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



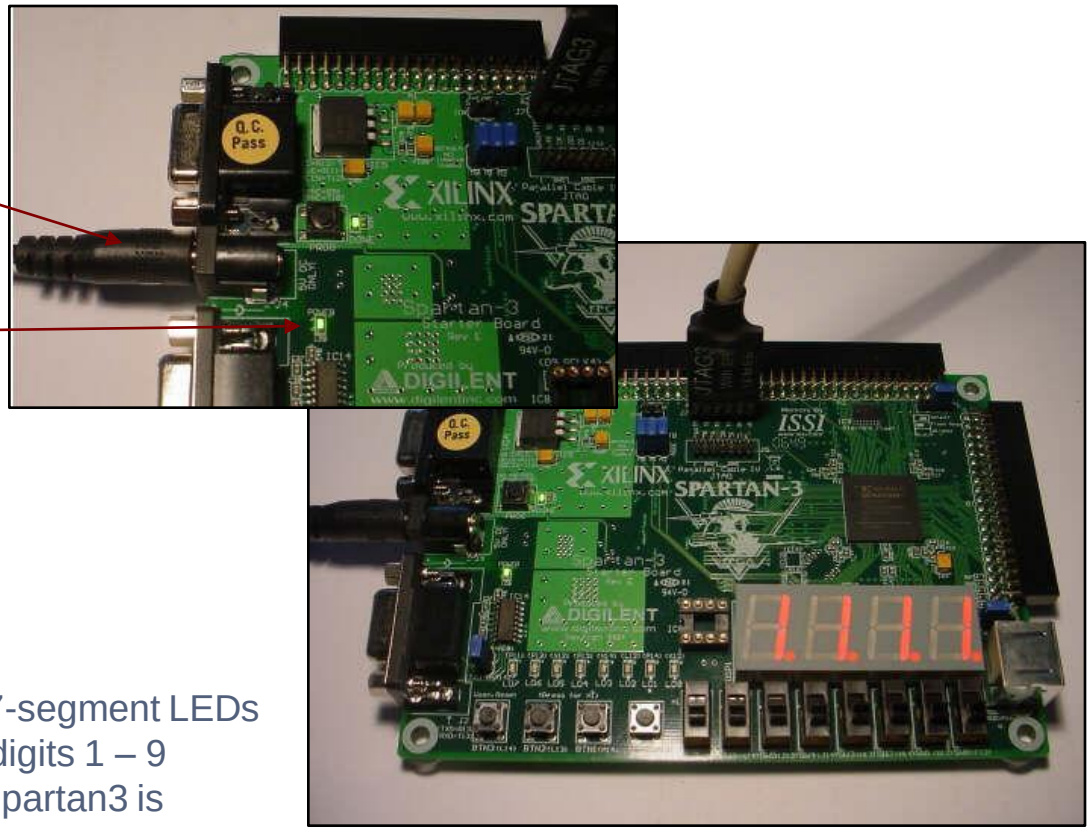



Download and Execute

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

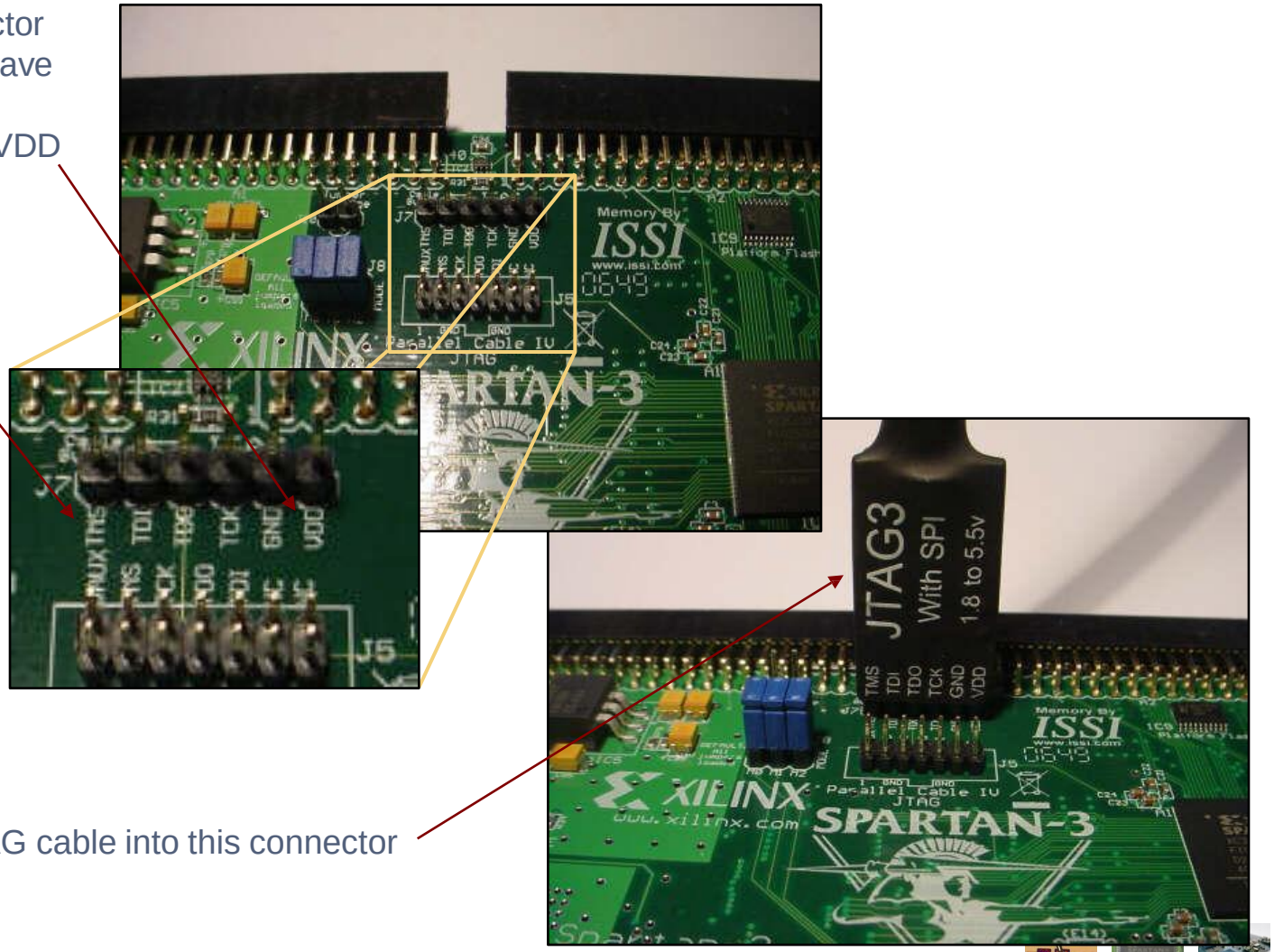
Locate the JTAG connector on the Spartan3. It will have

VDD

on one end and

TMS

on the other



Plug the JTAG cable into this connector

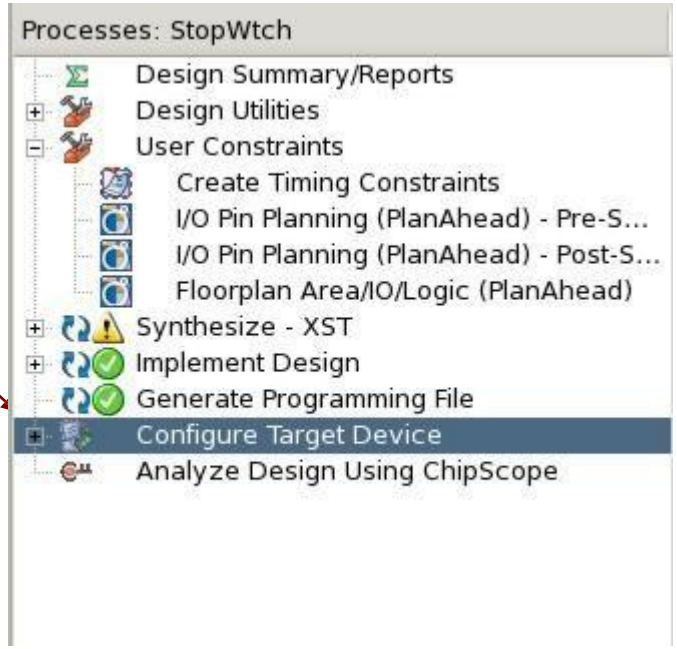
Download and Execute

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

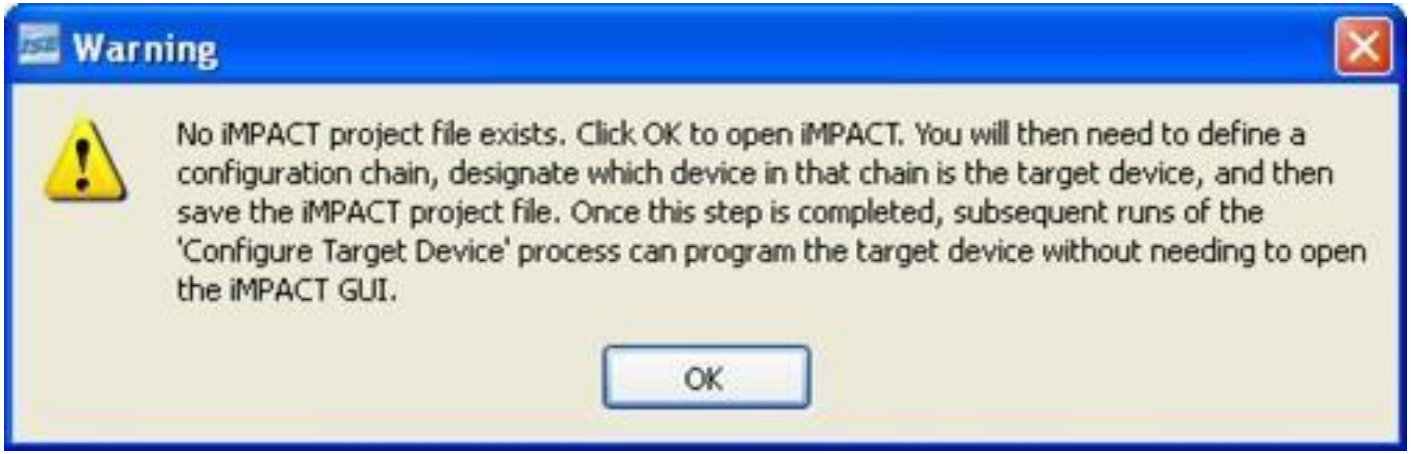
In the ISE double click on Configure Target Device – you will be invoking a program called “iMPACT”



Download and Execute

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

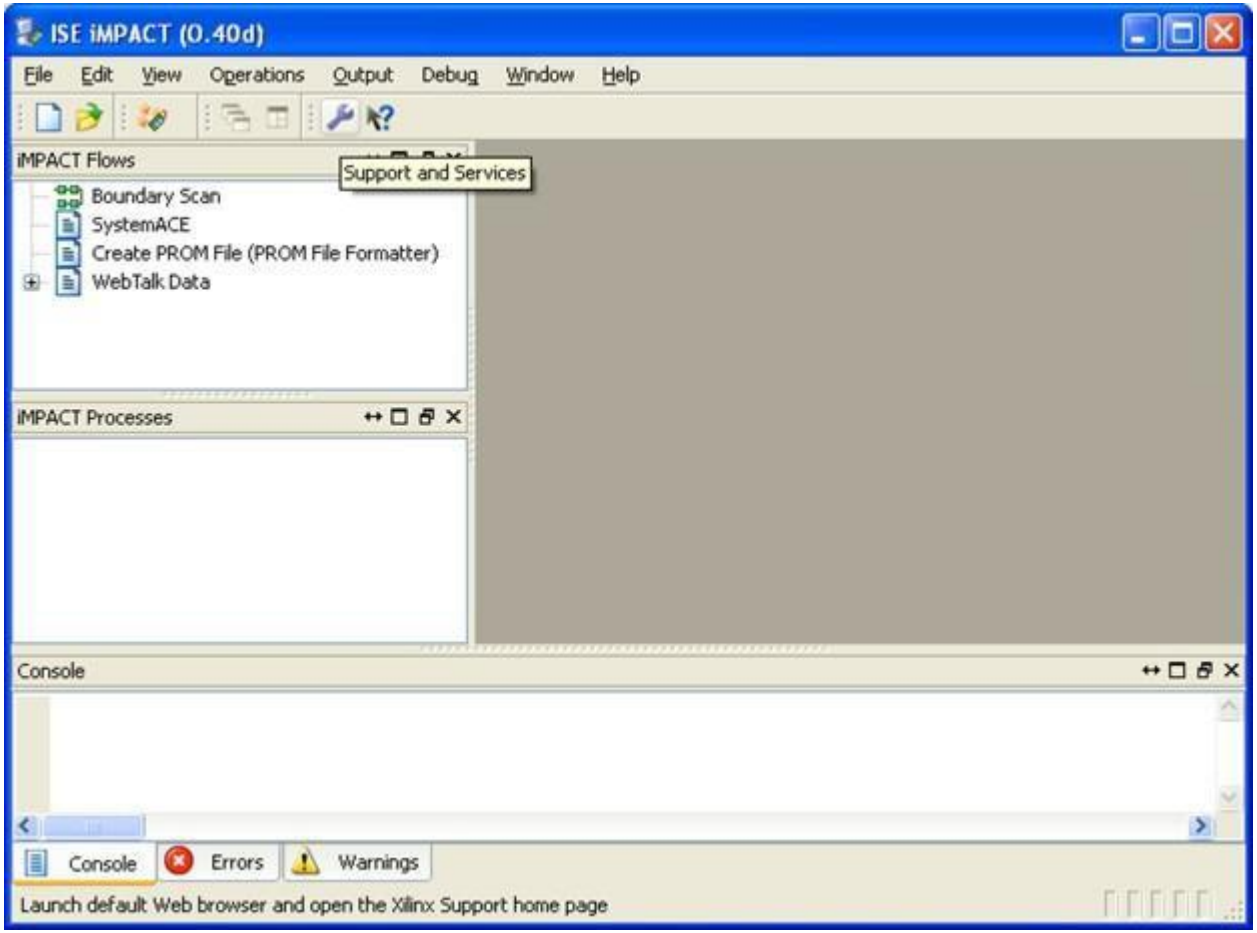
Just click OK



Download and Execute

iMPACT will open.

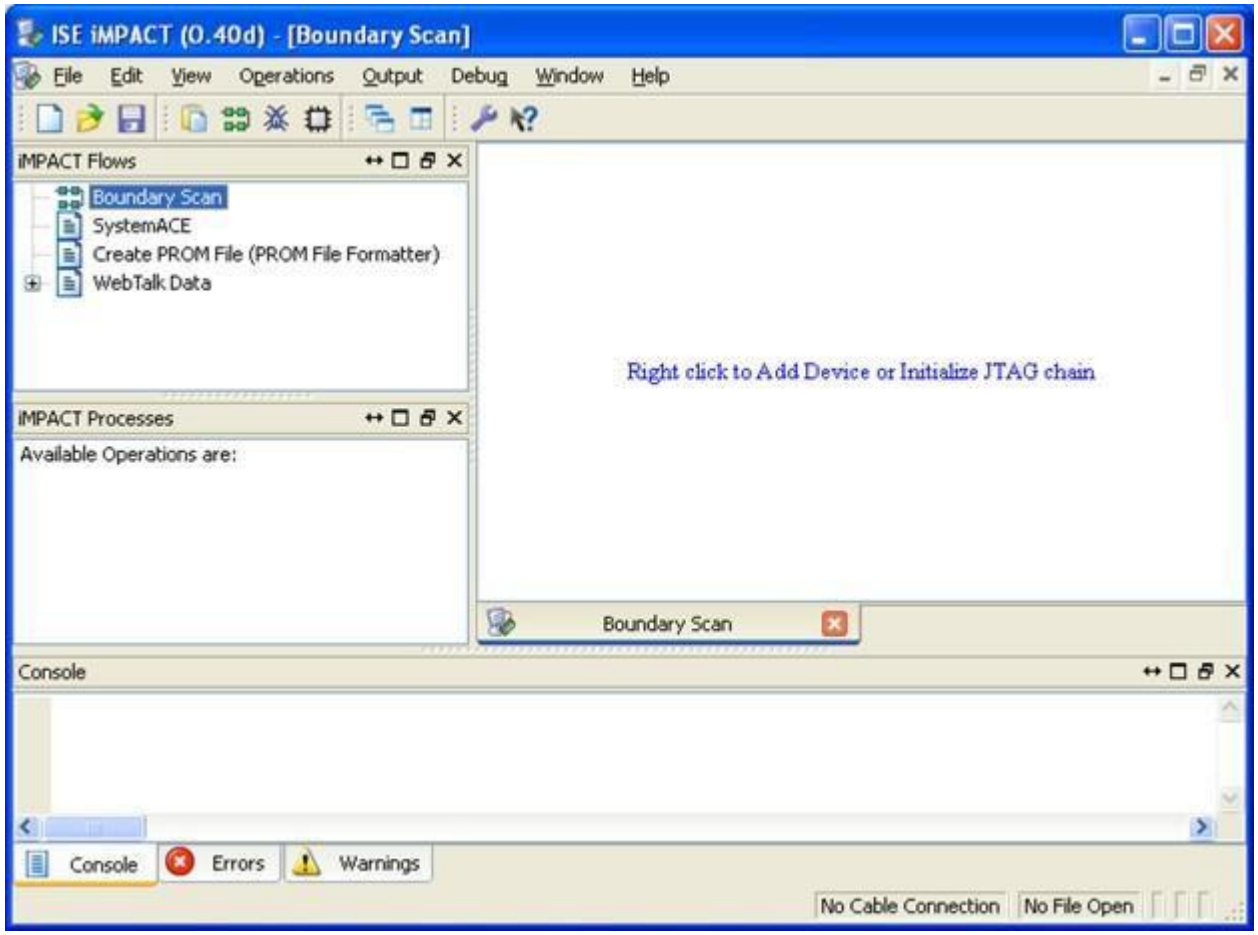
Double Click on Boundary Scan



Download and Execute

iMPACT will open.

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

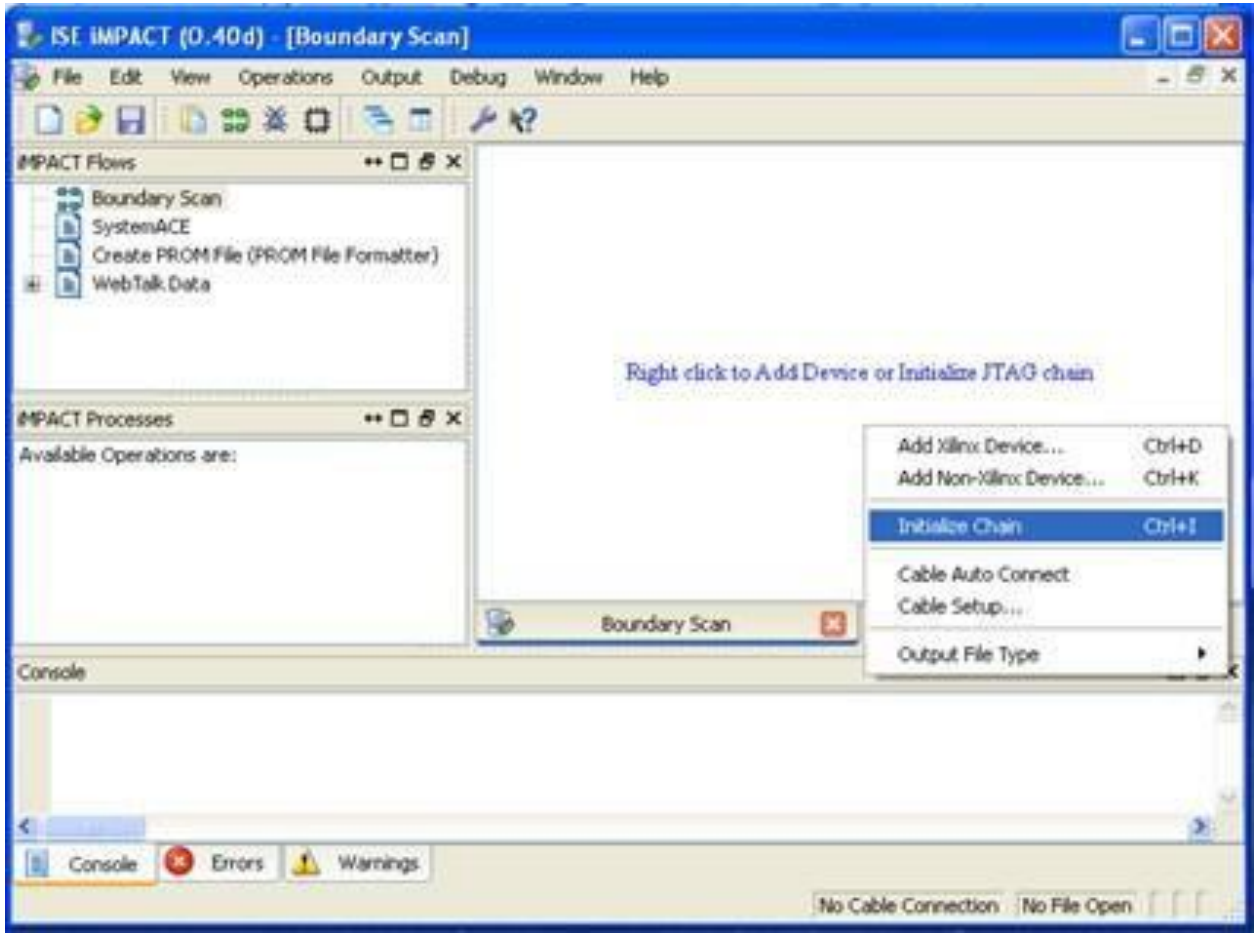


Download and Execute

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.

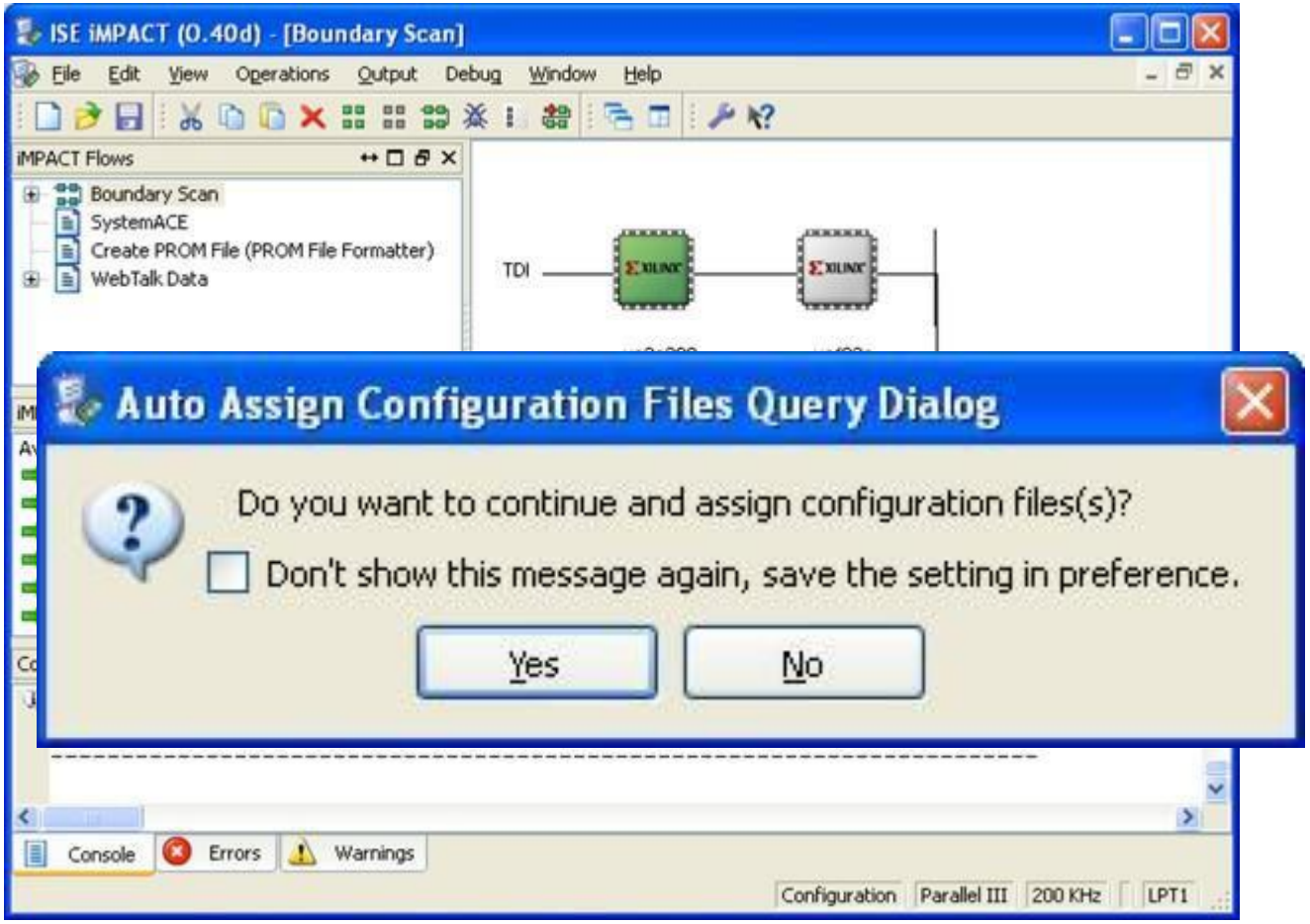


Download and Execute

iMPACT will open.

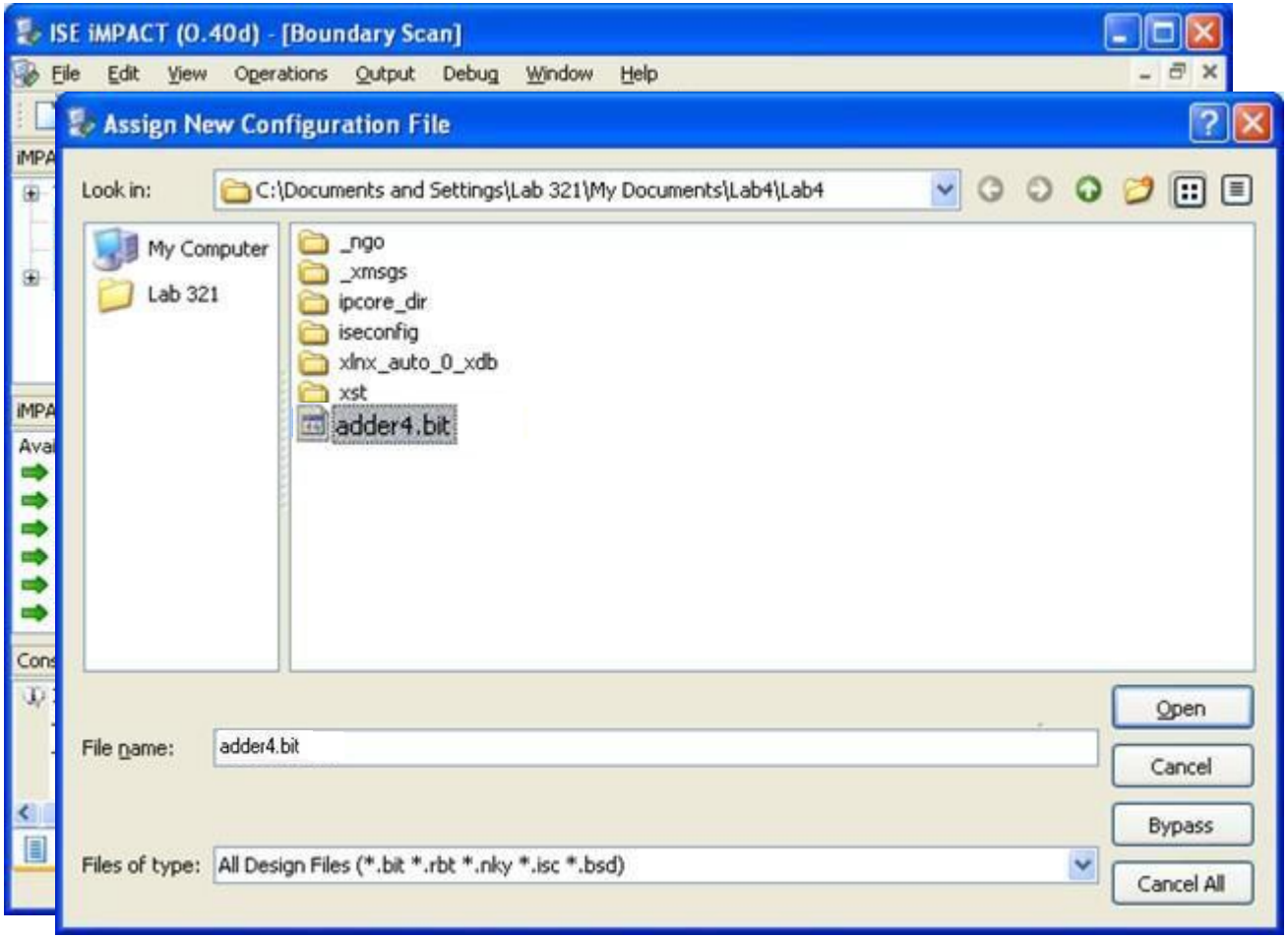
Double Click on Boundary Scan and then right click on the new window and click on Initialize Chain

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



Download and Execute

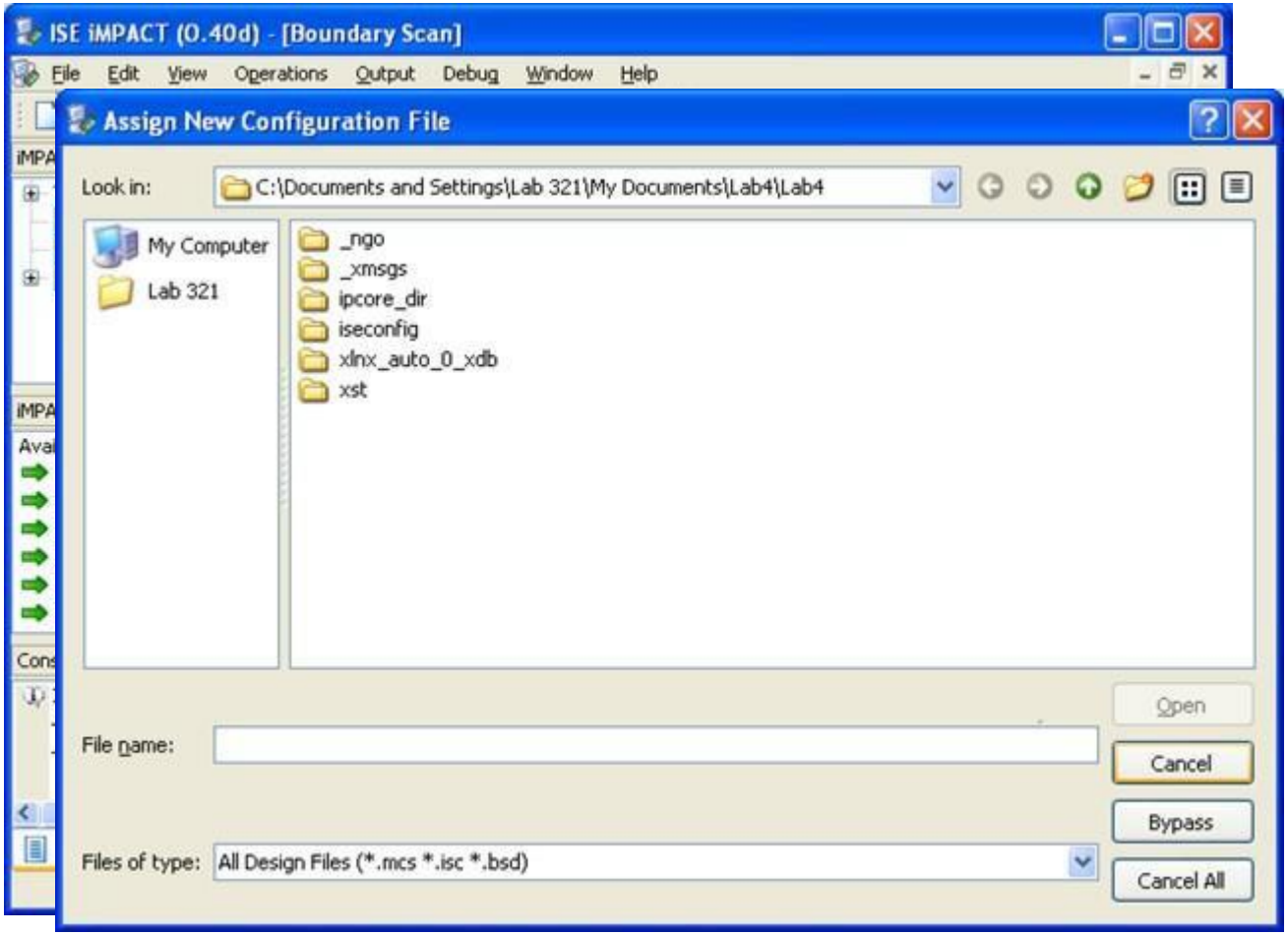
Click on the appropriate *.bit file and then click open



Download and Execute

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

On the next, click either cancel or bypass

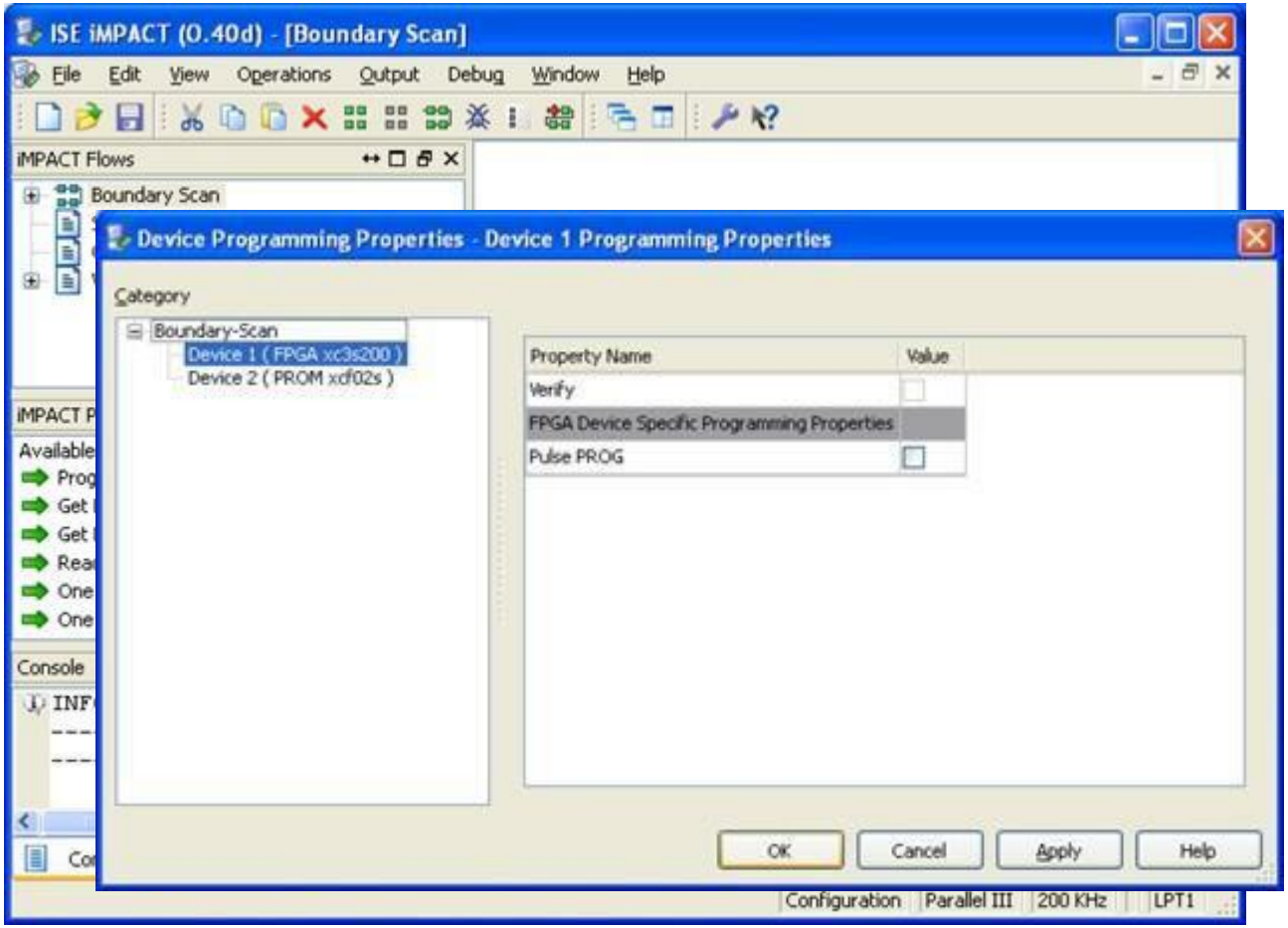


Download and Execute

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

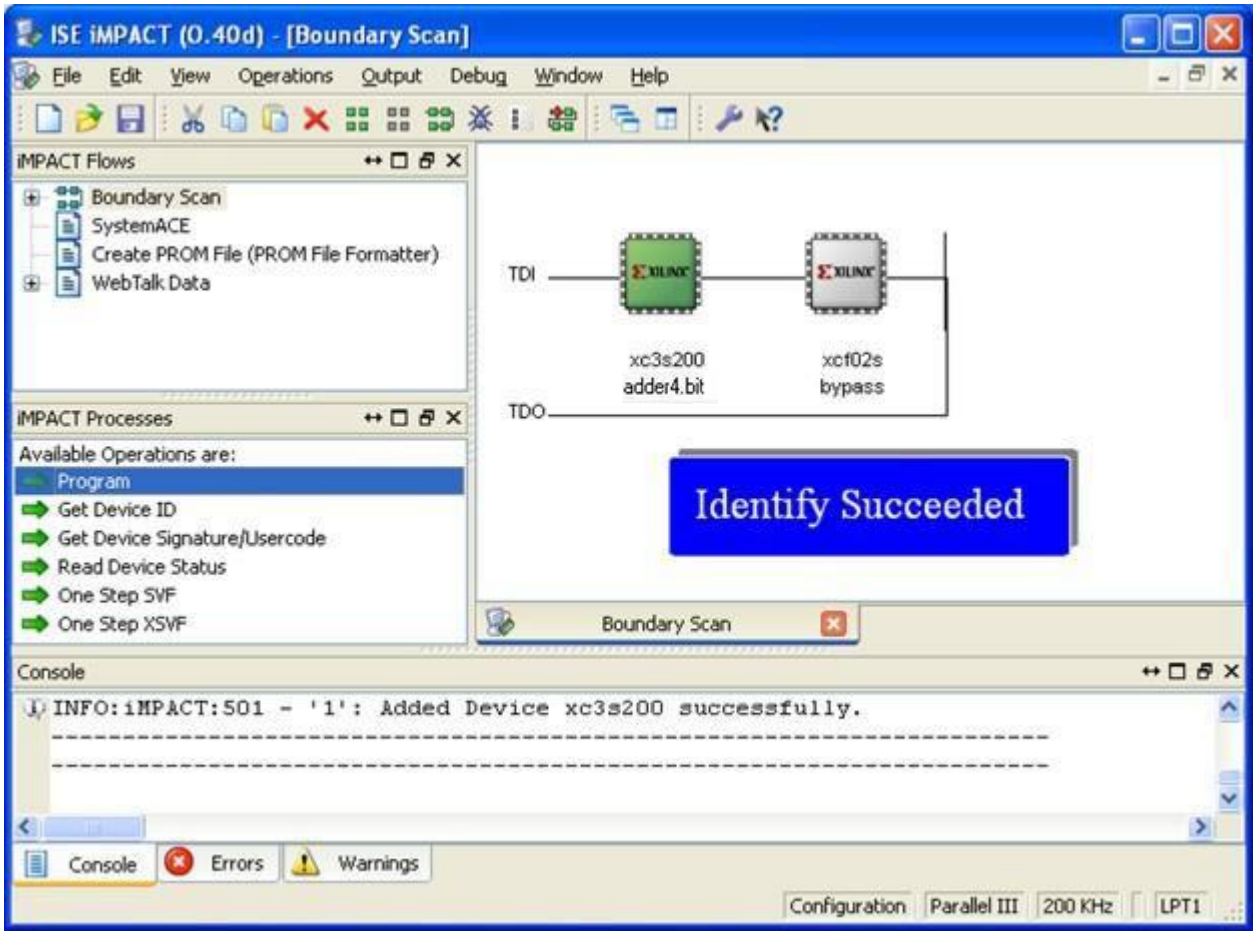
On the next, click either cancel or bypass

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

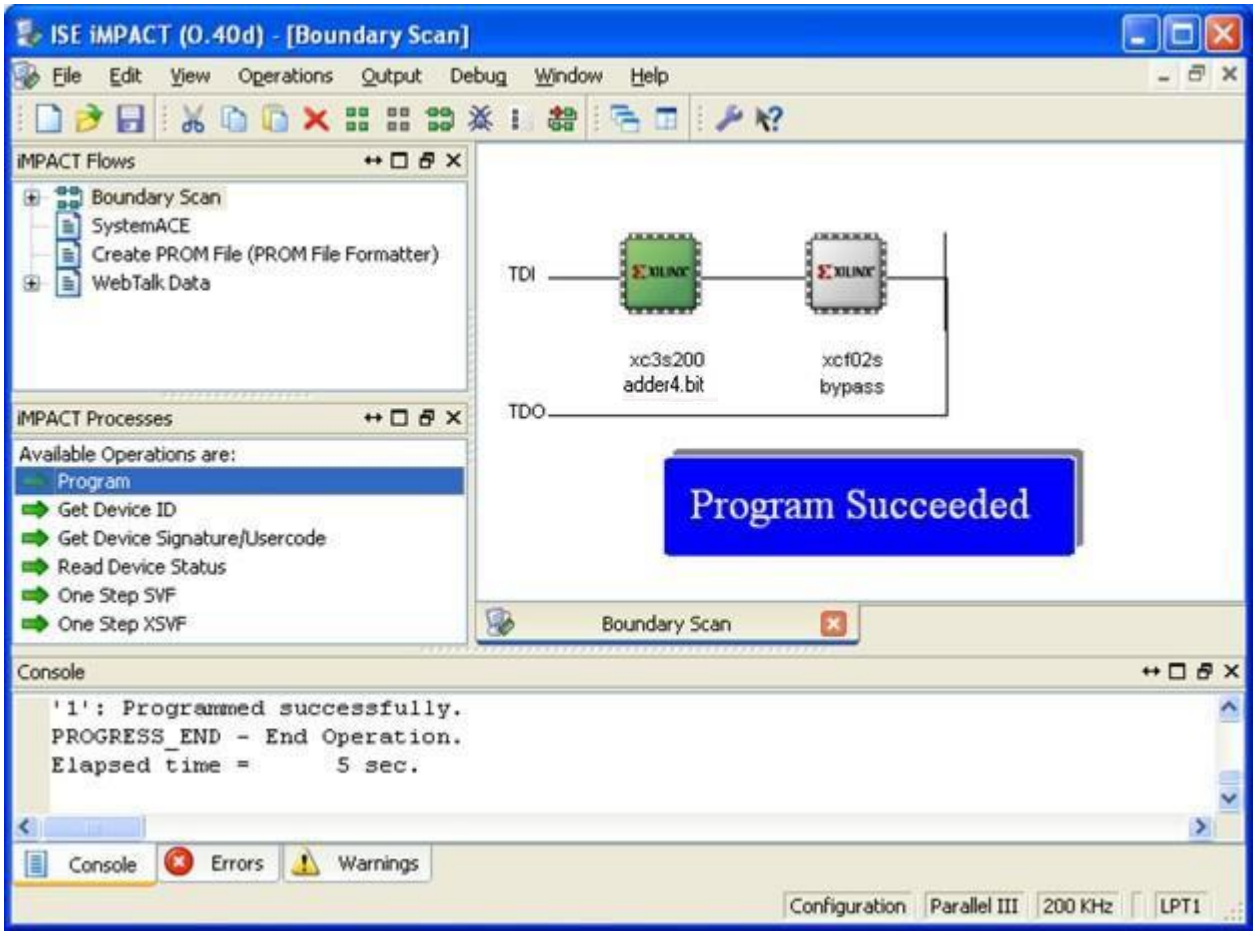
Double Click on Program



Download and Execute

Double Click on Program

…… and hopefully, you will see “Program Succeeded

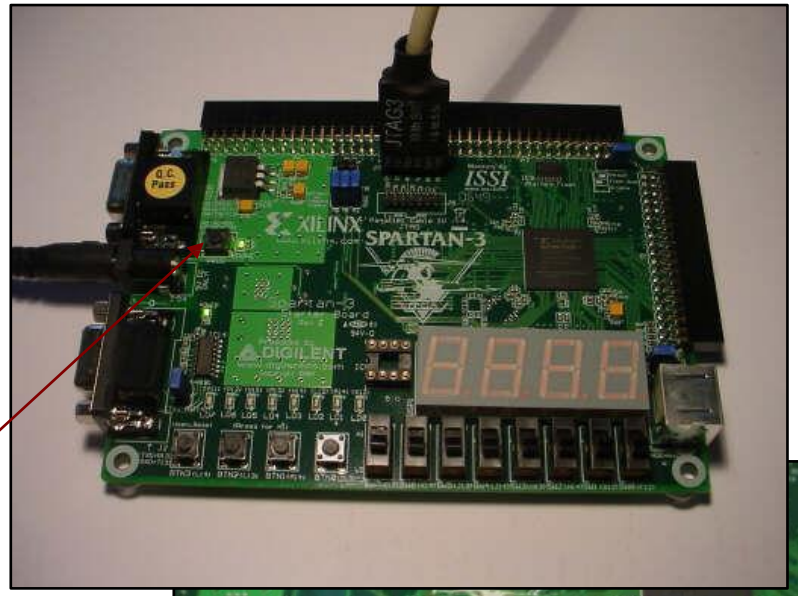


Download and Execute

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

It is time to test the design

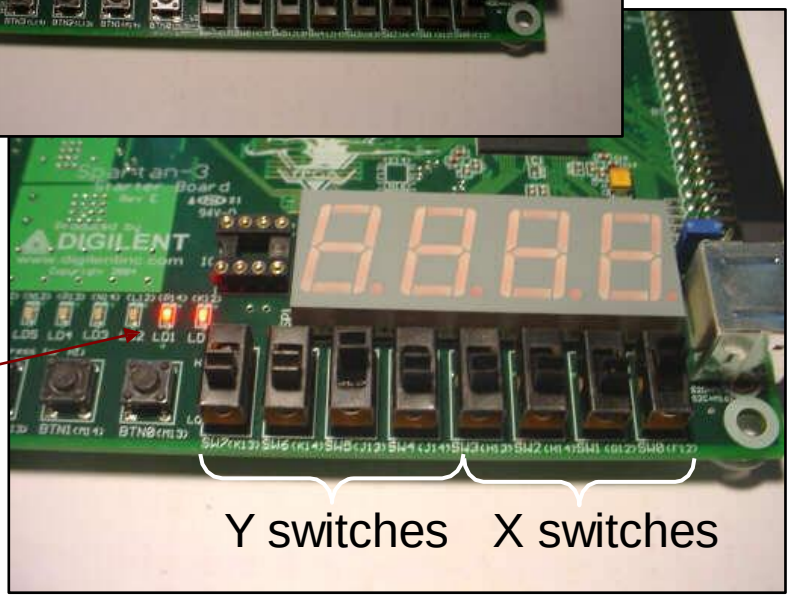
(Note: the design will continuously execute, until you hit the RESET button or you turn off the power)



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

The result was 3

(So far, so good)



Download and Execute

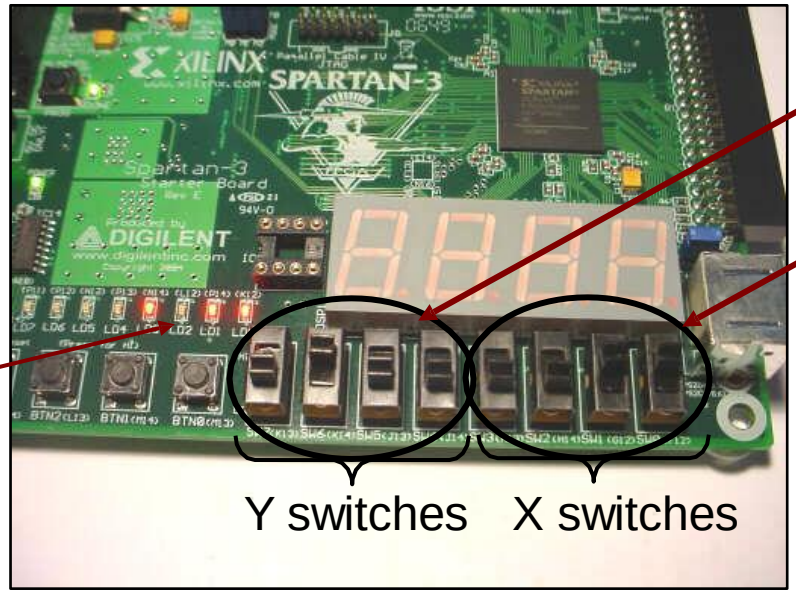
I then tested:
3+4 (X=3, Y=4)

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….



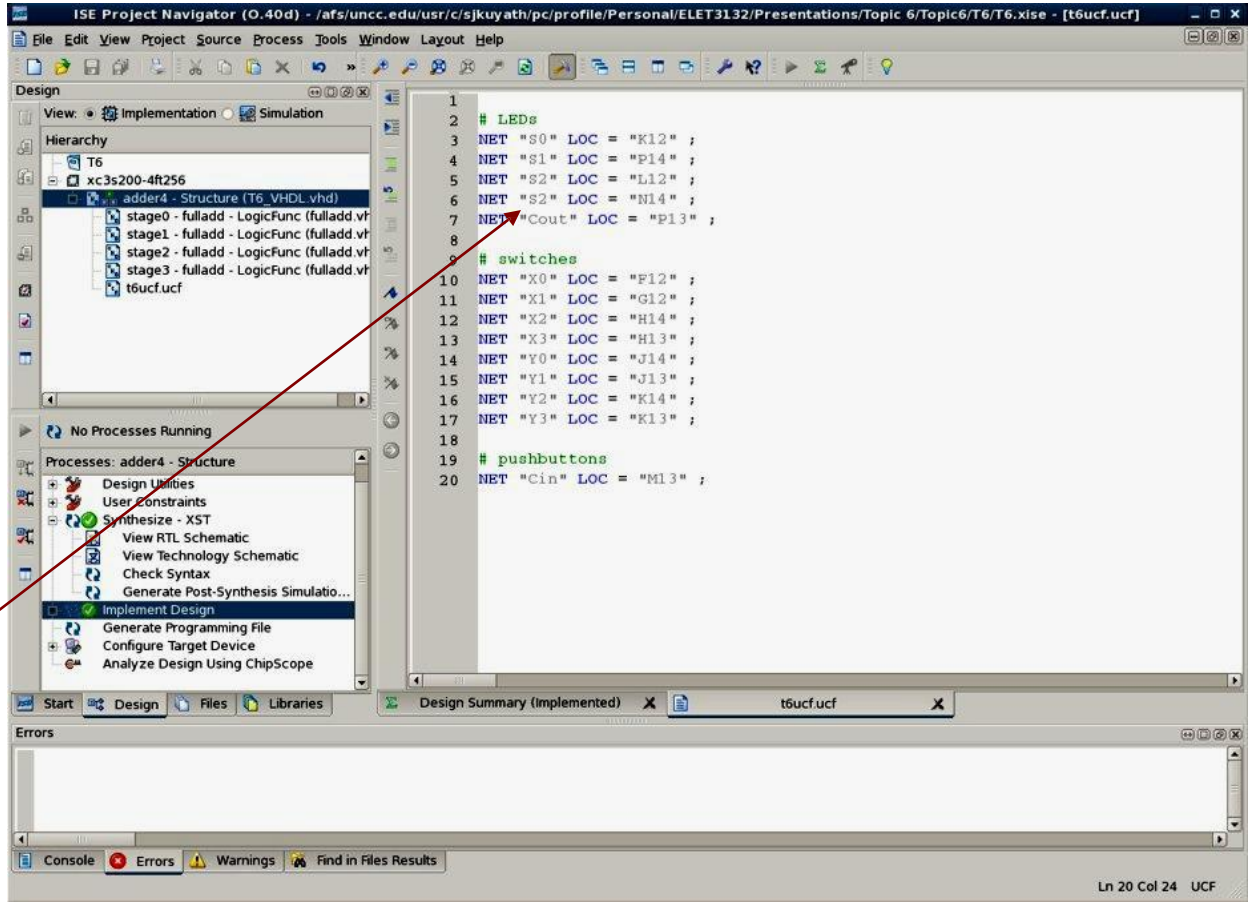

Troubleshooting

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.



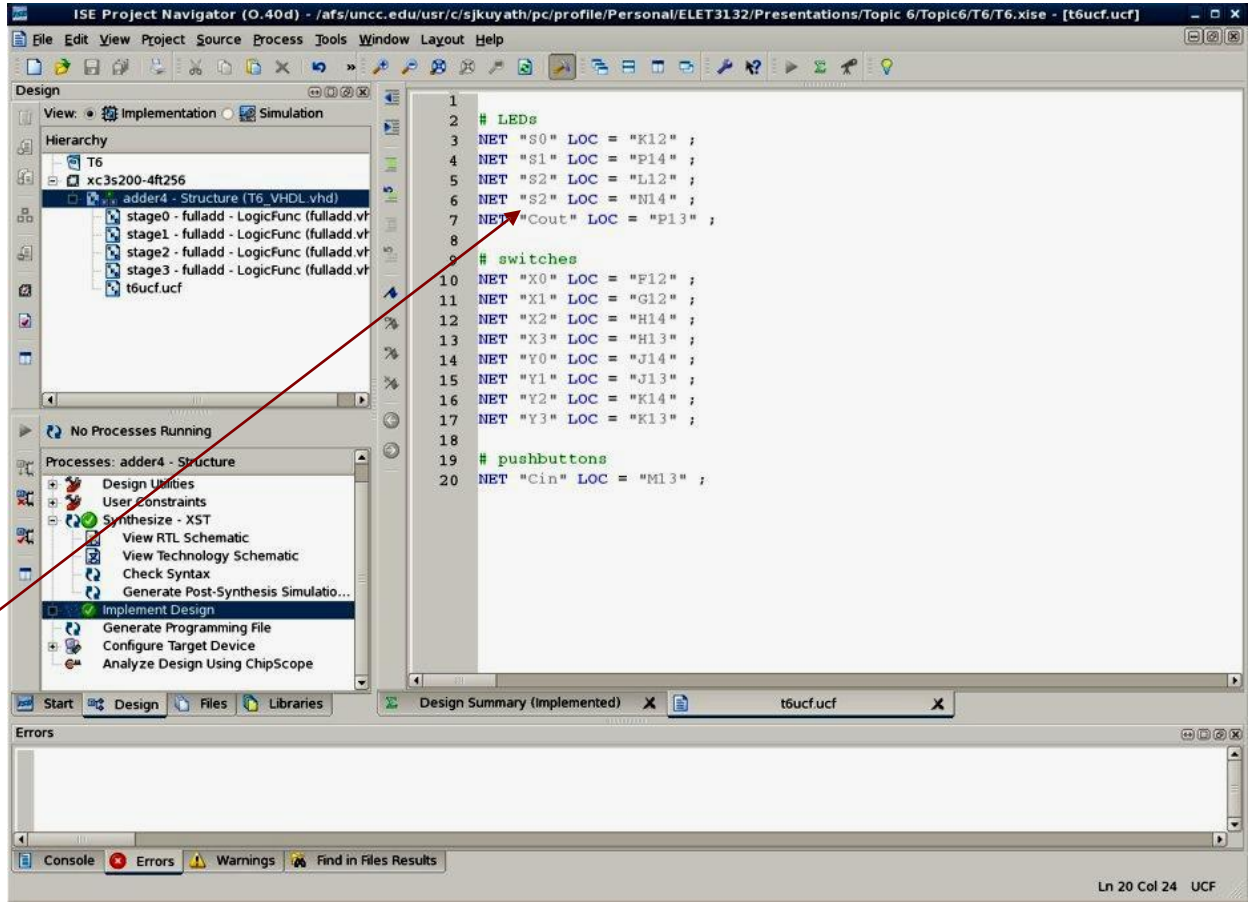
Troubleshooting

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.



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

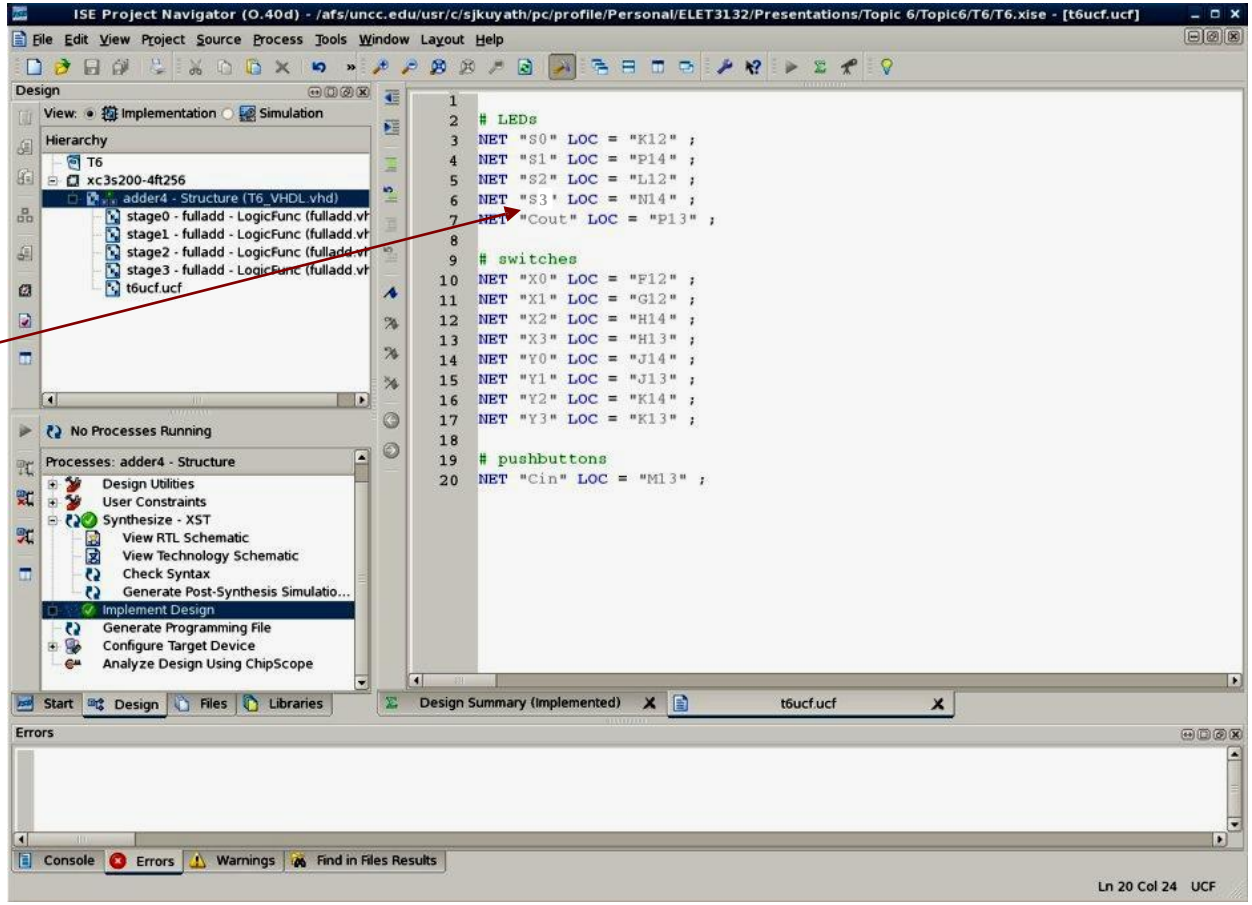



Troubleshooting

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

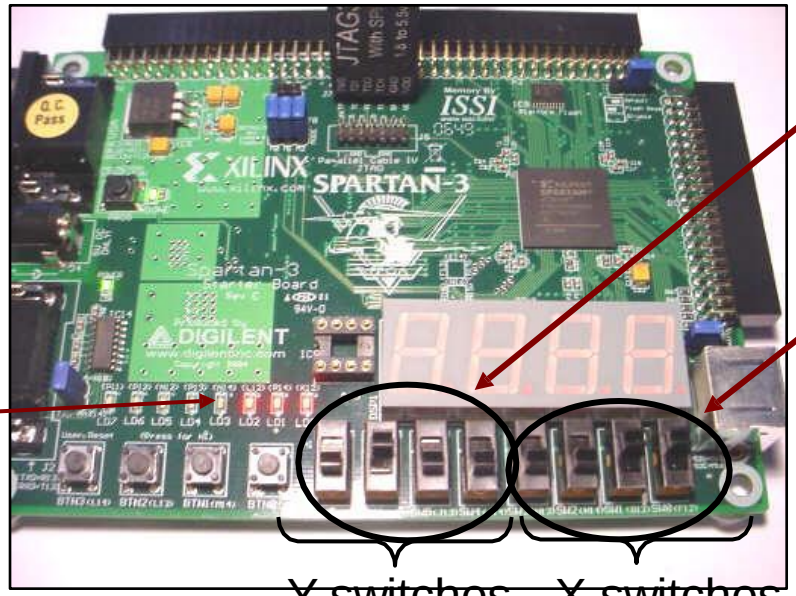


Download and Execute

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

The result was 7

(ahhh….)



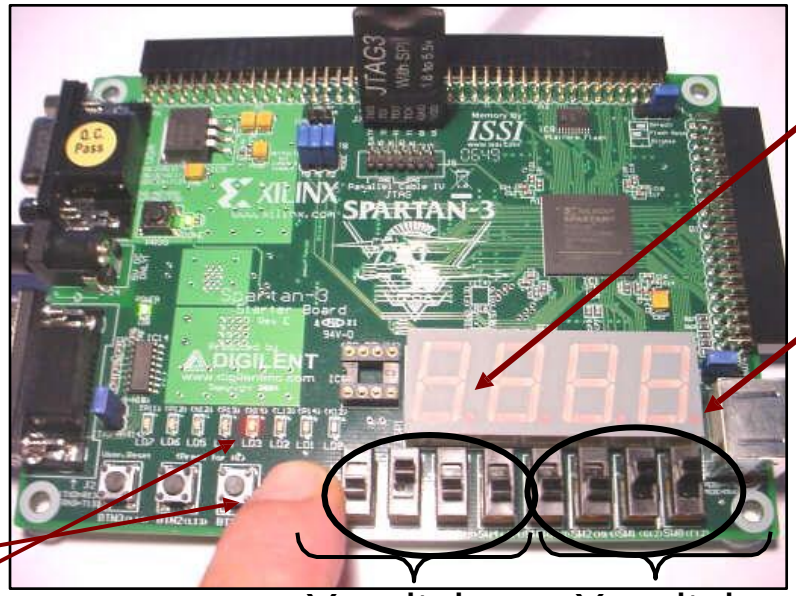
Download and Execute

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

But, held down the Carry In button (PB0)

The result was 8

(ahhh….)



0100 = 4

0011 = 3

Y switches

X switches

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
    - Troubleshot the design
    - Downloaded the design to the Spartan3 board
    - Executed the design to verify and test