Tutorial: Hardware Trojan Insertion on FPGA

Kan Xiao and Mohammed Tehranipoor
Department of Electrical & Computer Engineering
University of Connecticut

1. Introduction

To lower the cost of IC design and fabrication, the supply chain of the semiconductor industry has been distributed around the world. As the complexity of ICs increases, more and more highly specialized companies become involved in the IC fabrication process to improve efficiency and manufacturability, providing attackers (adversaries?) with more opportunities to make malicious inclusions and alterations. This is a serious problem for security-critical applications, such as military, transportation and financial systems. Some Trojans can be inserted into a design if any untrusted tools or IP blocks are used. Other Trojans can be implanted by modifying the layout of the design during GDSII development and fabrication. Trojans in ICs may cause malfunctions, lower reliability, leak confidential information to adversaries or even destroy the system under specifically designed conditions. Detecting these malicious inclusions and alterations is extremely difficult, due to the following characteristics of Trojans: First, Trojans are small compared to the designs they have altered, which makes attributes of Trojan-inserted ICs almost the same as those of Trojan-free ICs. Second, Trojans can be kept dormant during most of their operation, and be activated under very specific conditions. Third, a Trojan's behavior is unknown. Thus, it would be challenging to devise a Trojan detection technique that can target all types of Trojans.

2. FPGA Board and ISE Tool

This tutorial describes the flow to insert Trojan into a design and implement the design onto FPGAs. The FPGA board I use in this tutorial is **Basys™2 Spartan-3E FPGA Board** from Digilent Inc. (http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,400,790&Prod=BASYS2). The Design tool is ISE WebPACK 14.1 which is a free design suite which can be downloaded from Xilinx official websites (http://www.xilinx.com/support/download/index.htm). If you want to know more about Xilinx FPGA design flow, there are some helpful tutorials available on the Xilinx website (http://www.xilinx.com/support/documentation/dt_ise14-1_tutorials.htm). Note that this tutorial is based on the flow presented in the In-depth ISE tutorial. You can refer to it for more detail information.

3. Trojan Design for Different Detection Approaches

In general, hardware Trojan detection approaches can be divided into two categories: full Trojan activation approaches and side-channel analysis approaches. The first approach tries to activate Trojans by applying test vectors and comparing responses with the correct results.

Side-channel signal analysis has been developed to detect hardware Trojans by measuring circuit parameters, such as power (transient and leakage current) and delay.

In this tutorial, we only talk about the hardware Trojans which are implanted by modifying the layout of the design during GDSII development and fabrication. Unlike with ASICs, it is very difficult to insert a Hardware Trojan into an unused LUT in FPGAs because we lack knowledge of the internal structure of an FPGA and cannot completely control placement and routing by using the available design suites from Xilinx. Fortunately, we can design the hardware Trojan according to our requirements and objectives. In order to evaluate different Trojan detection approaches, different Trojan insertion methods are used. In general, the complexity of Trojan design likes: **full Trojan activation < power-based Trojan detection < delay-based Trojan detection**. The Trojan design flow for different detection methods will be presented in the next section.

3.1 Trojans for Full Trojan Activation

For full Trojan activation approach, only functional testing is performed. We just care about the function instead of the layout, so it is the easiest one. First, I will explain how to design and implement the Trojan-free circuits on an FPGA. Then I will describe how to implement Trojan-inserted circuits on an FPGA.

3.1.1 Trojan-free Design Implementation

To implement Trojan-free design on FPGA, we just need to use the basic implementation flow. In this subsection, I will go through the entire flow for an example. The example design we use can be downloaded with this tutorial.

Starting ISE:

To start the ISE software, double-click the ISE project navigator icon on your desktop, or select **Start > All Program > Xilinx ISE Design Suite 14.1 > ISE Design Tools > Project Navigator**.

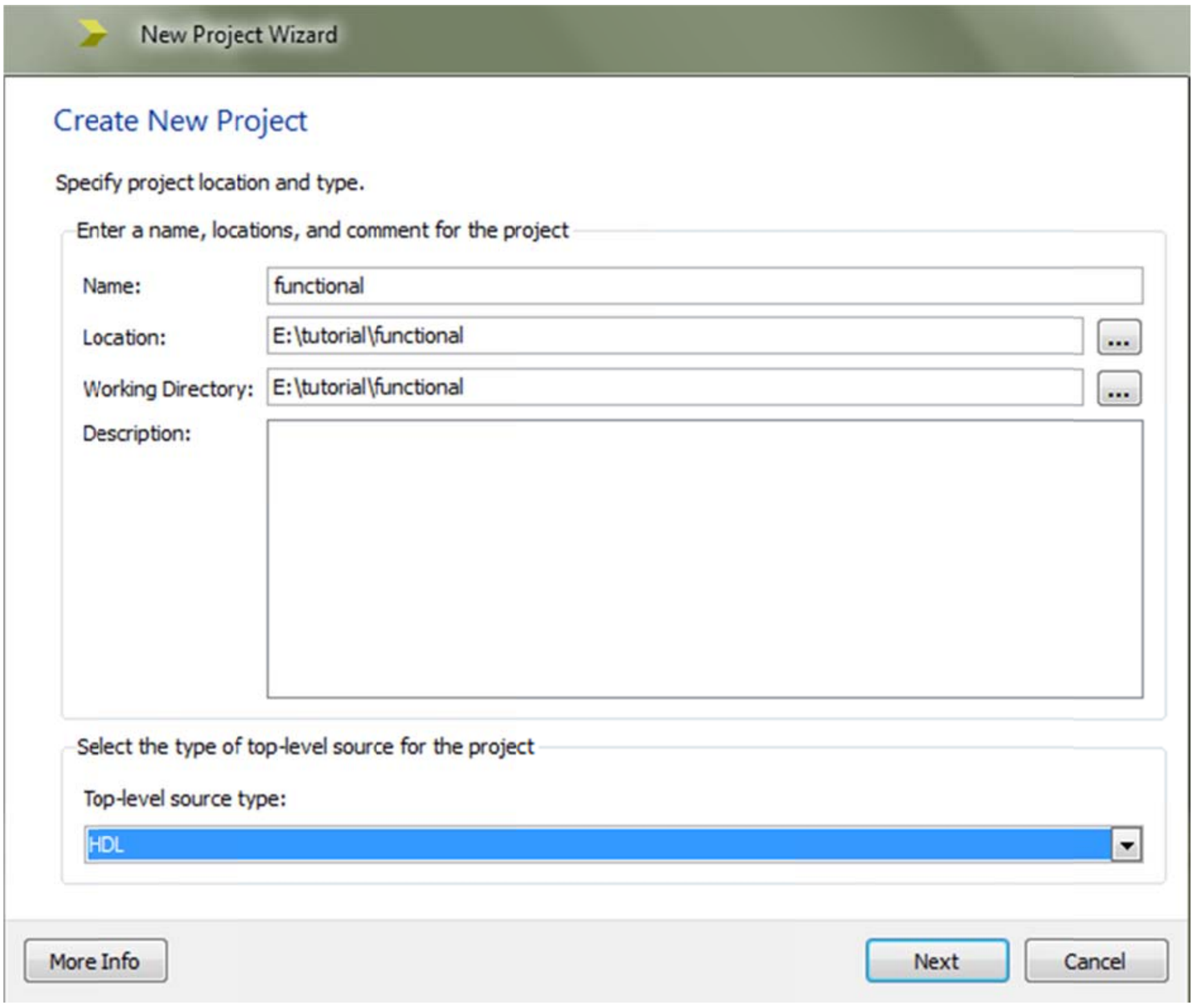
Creating a New Project:

To create a new project, do the following:

a) Select **File > New Project**. A new window wizard appears.

b) Choose Location and give project a name

c) Choose HDL as the Top-level source type, and click **Next**.


New Project Wizard

Create New Project

Specify project location and type.

Enter a name, locations, and comment for the project

Name: functional

Location: E:\tutorial\functional

Working Directory: E:\tutorial\functional

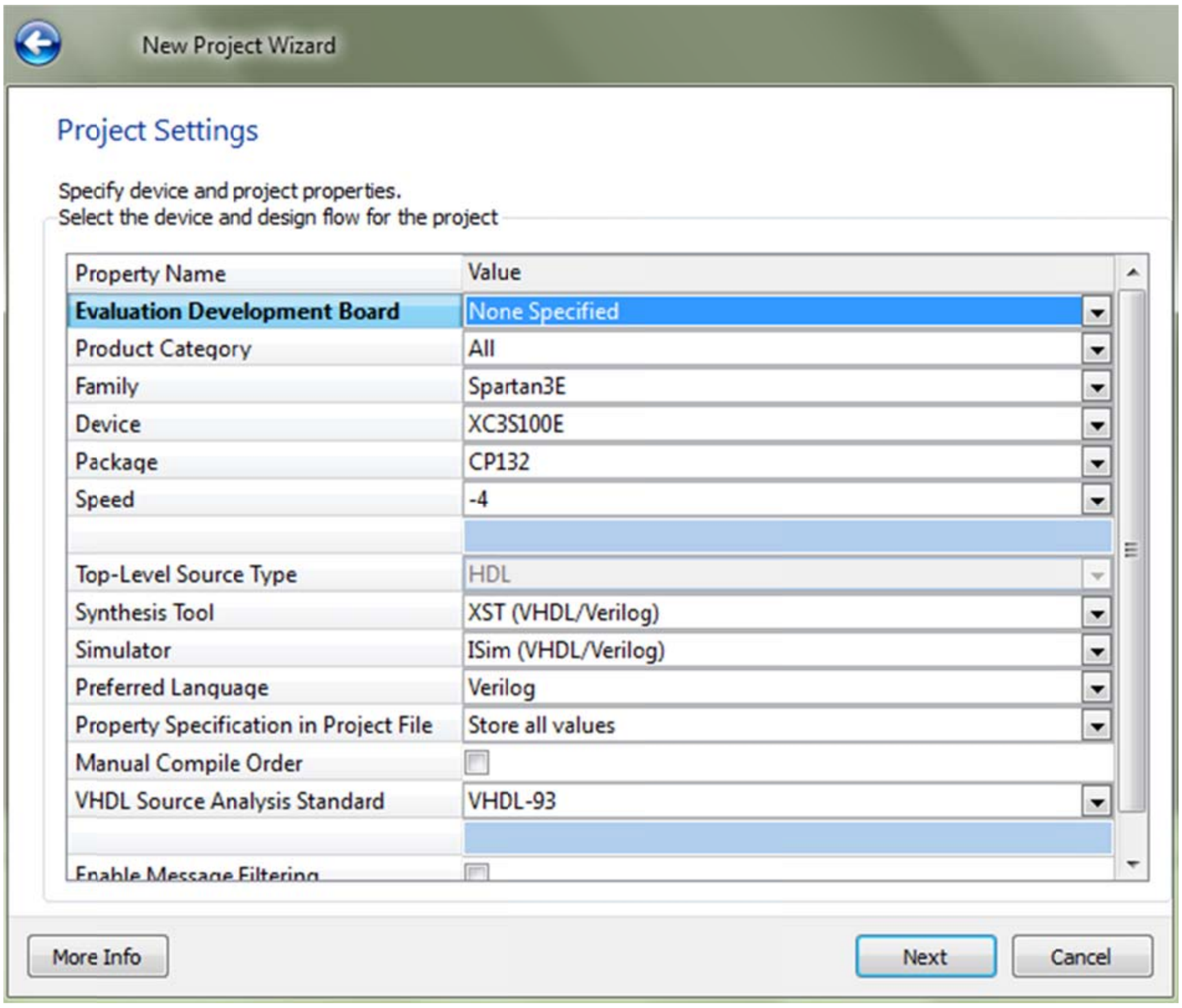
Description:

Select the type of top-level source for the project

Top-level source type: HDL

Figure 1

The new project wizard--- Device Properties page appears.



Project Settings

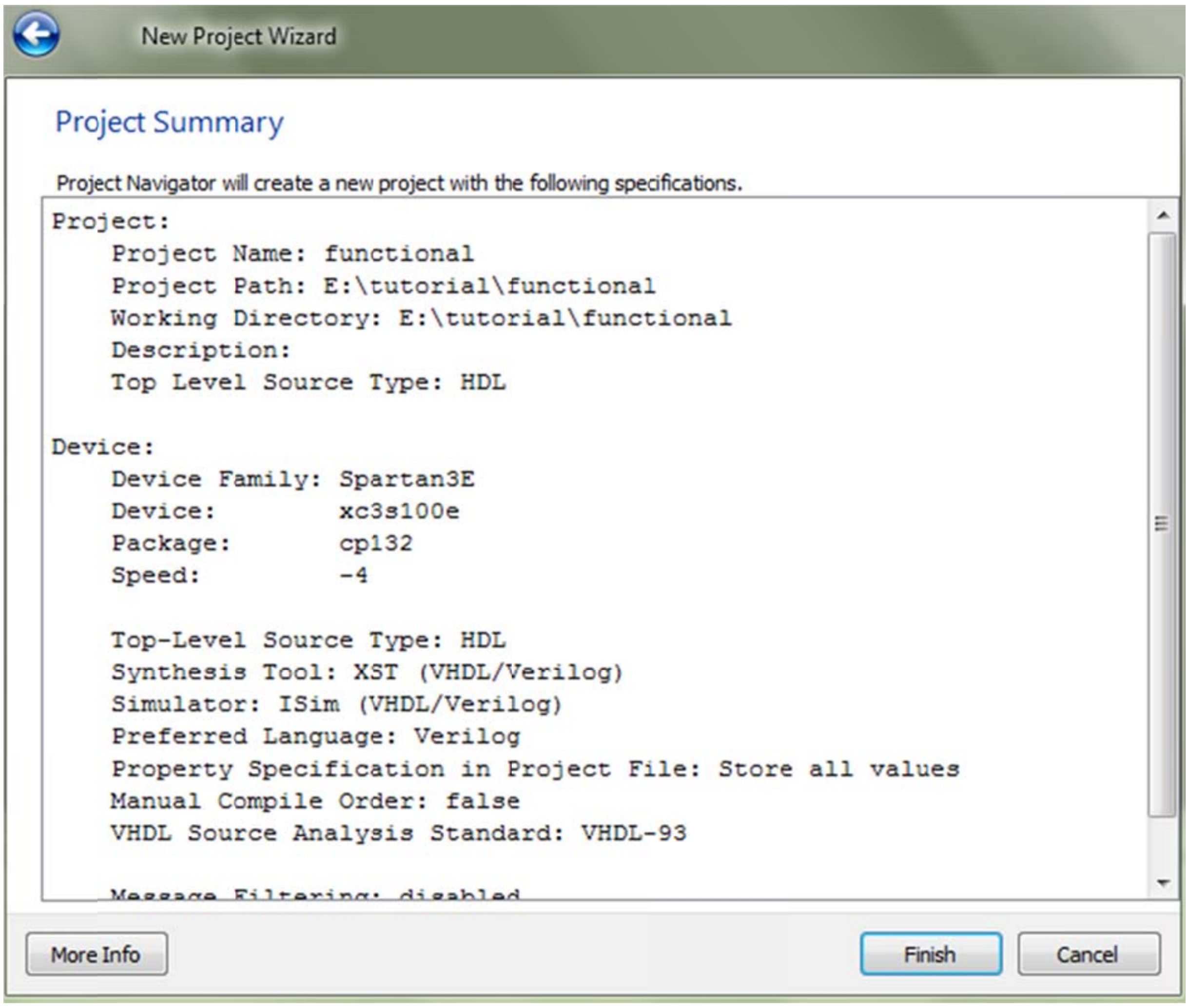
Specify device and project properties.
Select the device and design flow for the project

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

Figure 2

d) Select required information as shown in Figure 2, and click **Next**.

The new project wizard--- Project summary appears.



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

```
Project:
    Project Name: functional
    Project Path: E:\tutorial\functional
    Working Directory: E:\tutorial\functional
    Description:
    Top Level Source Type: HDL

Device:
    Device Family: Spartan3E
    Device:        xc3s100e
    Package:       cp132
    Speed:         -4

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

    Message Filtering: disabled
```

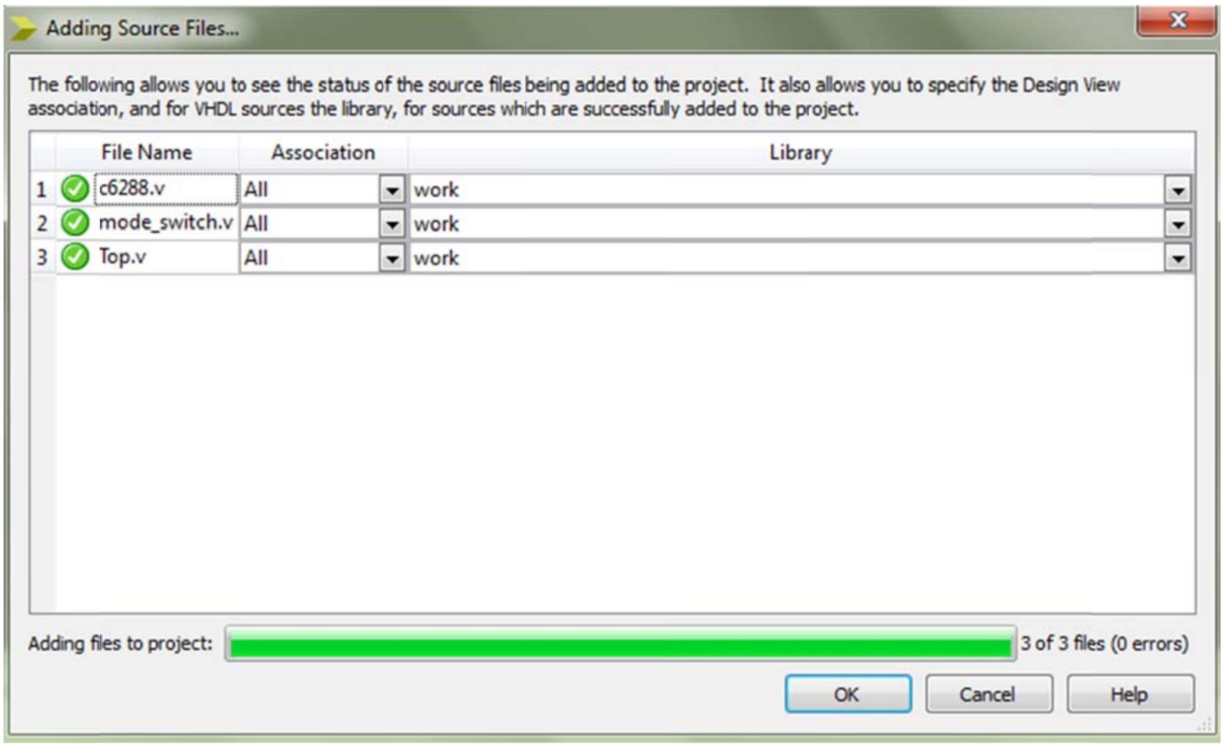
Figure 3

Click **Finish** and the new project is created.

Adding Sources:
For this hierarchical design, you will examine HDL files, correct syntax errors, create an HDL macro, and add a CORE Generator software core and a clocking module. You will create and use each type of design macro. All procedures used in the tutorial can be used later for your own designs.
HDL files must be added to the project before they can be synthesized. Otherwise, you need to create a new source and type the HDL code in it. You will add source files to the project as follows:
a) Select **Project > Add Source** (or **Add Copy of Source** so that all HDL files will be copied to your project directory)
b) Select the following files from the project directory, and click **Open**.
        Top.v
        c6288.v

mode_switch.v

c) In the Adding Source Files dialog box, verify that the files are associated with **All**, that the associated library is **work**, and click **OK**.



Adding Source Files...

The following allows you to see the status of the source files being added to the project. It also allows you to specify the Design View association, and for VHDL sources the library, for sources which are successfully added to the project.

		File Name	Association	Library
1	✓	c6288.v	All	work
2	✓	mode_switch.v	All	work
3	✓	Top.v	All	work

Adding files to project: 3 of 3 files (0 errors)

Figure 4

The Hierarchy pane in the Design panel displays all of the source files currently added to the project, with the associated entity or module names. Each source design unit is represented in the Hierarchy pane using the following syntax: instance name-entity name- (file name). Instantiated components with no entity or module declaration are displayed with a question mark.

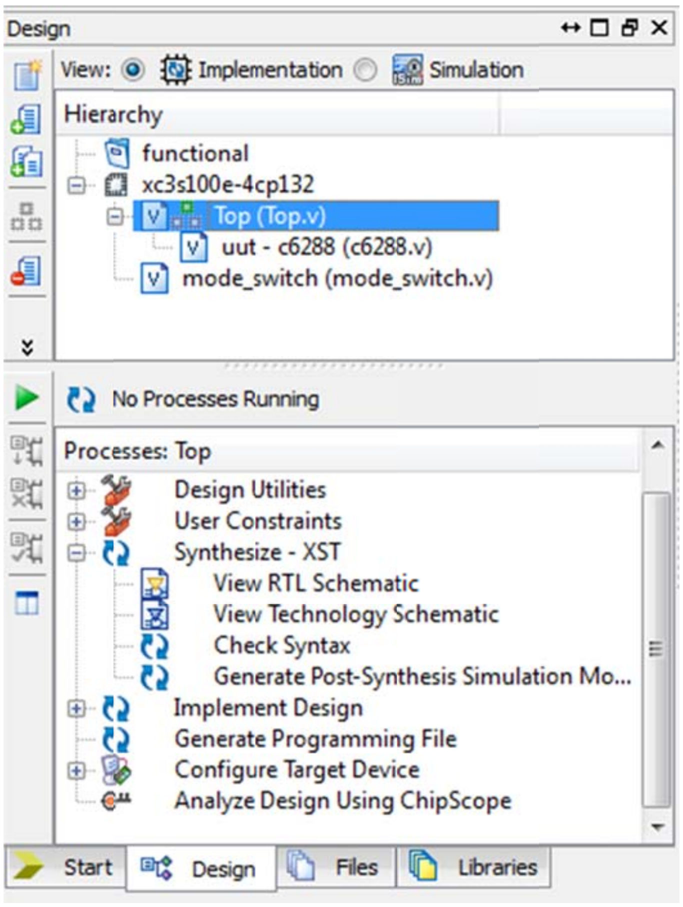


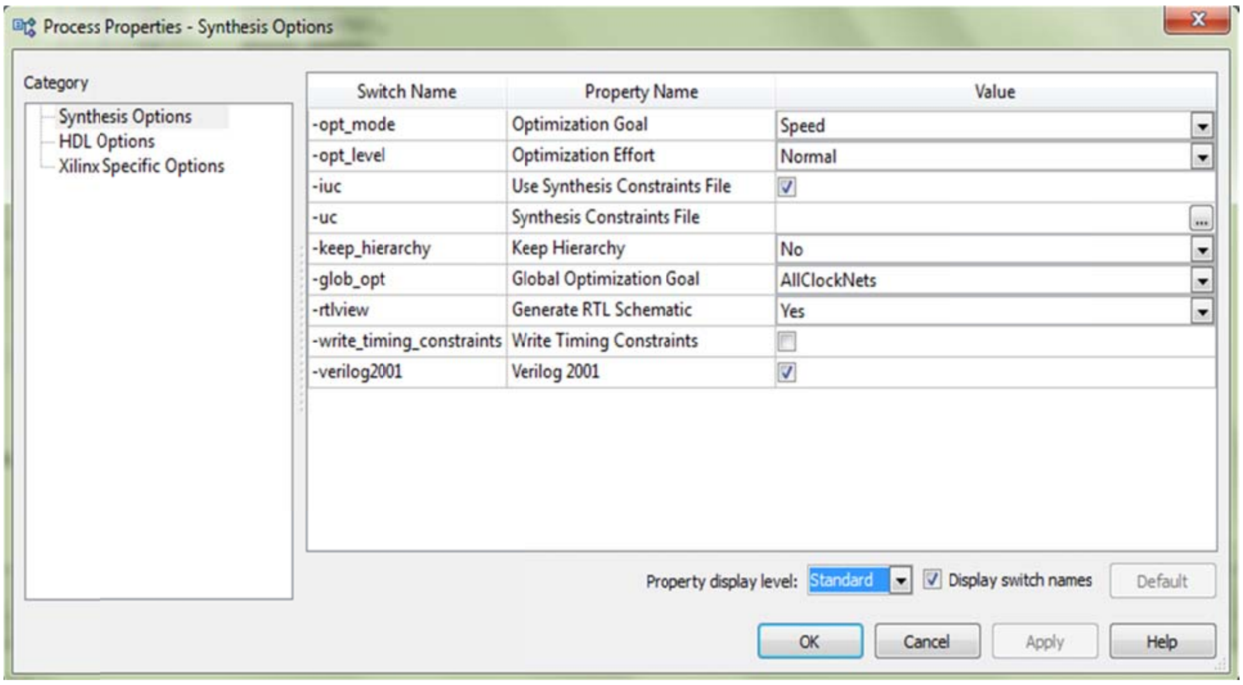
Figure 5

Check HDL Errors:
Select Top file in the Hierarchy pane, and double click Check Syntax in Synthesize-XST in the Processes pane as shown in Figure 5.
If there is an error, the “ERROR” message in the Console indicates the failure and provides a summary and the line number of the syntax problem.

Synthesizing the Design

So far you have been using Xilinx Synthesis Technology (XST) for syntax checking. Next, you will synthesize the design using either XST, Synplify, or Precision software. In this tutorial, we only use XST. The synthesis tool uses the design’s HDL code and generates a supported netlist type (EDIF or NGC) for the Xilinx implementation tools.

Enter Synthesis Options:
Synthesis options enable you to modify the behavior of the synthesis tool to make optimizations according to the needs of the design. To enter synthesis options, do the following:
a) In the Hierarchy pane of the Project Navigator Design panel, select top module (top.v).
b) In the Processes pane, right-click the **Synthesize** process, and select **Process Properties**.
c) The Synthesis Options window will appear.



Switch Name	Property Name	Value
-opt_mode	Optimization Goal	Speed
-opt_level	Optimization Effort	Normal
-iuc	Use Synthesis Constraints File	✓
-uc	Synthesis Constraints File	
-keep_hierarchy	Keep Hierarchy	No
-glob_opt	Global Optimization Goal	AllClockNets
-rtlview	Generate RTL Schematic	Yes
-write_timing_constraints	Write Timing Constraints	☐
-verilog2001	Verilog 2001	✓

Figure 6

d) Click OK after changing options.

Synthesizing the Design:
Now you are ready to synthesize your design. To take the HDL code and generate a compatible netlist, do the following:
a) In the Hierarchy pane, select top module (top.v).
b) In the Processes pane, double-click the **Synthesize** process.

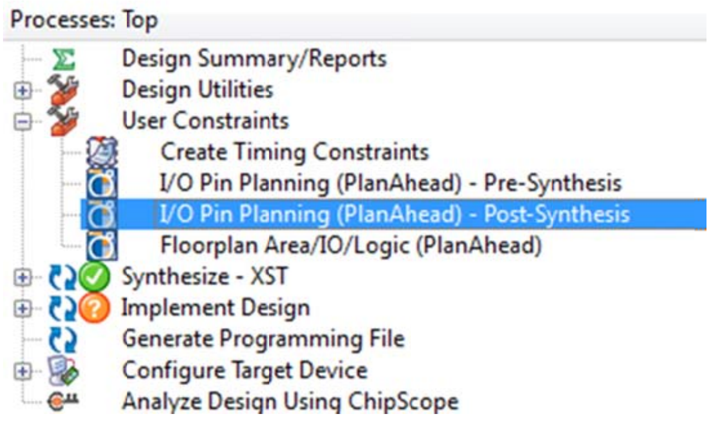
Implementing the Design

Creating Timing Constraints:
The User Constraints File (UCF) is a text file that can be edited directly with a text editor. To facilitate of this file, graphical tools are provided to create and edit constraints. The Constraints Editor and PlanAhead software are graphical tools that enable you to enter timing and I/O and placement constraints.
To launch the Constraints Editor, do the following:
a) In the Hierarchy pane of the Project navigator Design panel, select the top module (top.v).
b) In the Processes pane, expand **User Constraints**, and double-click **Create Timing Constraints**.
Our example is a combinational circuit, so we do not need to set timing constraints.

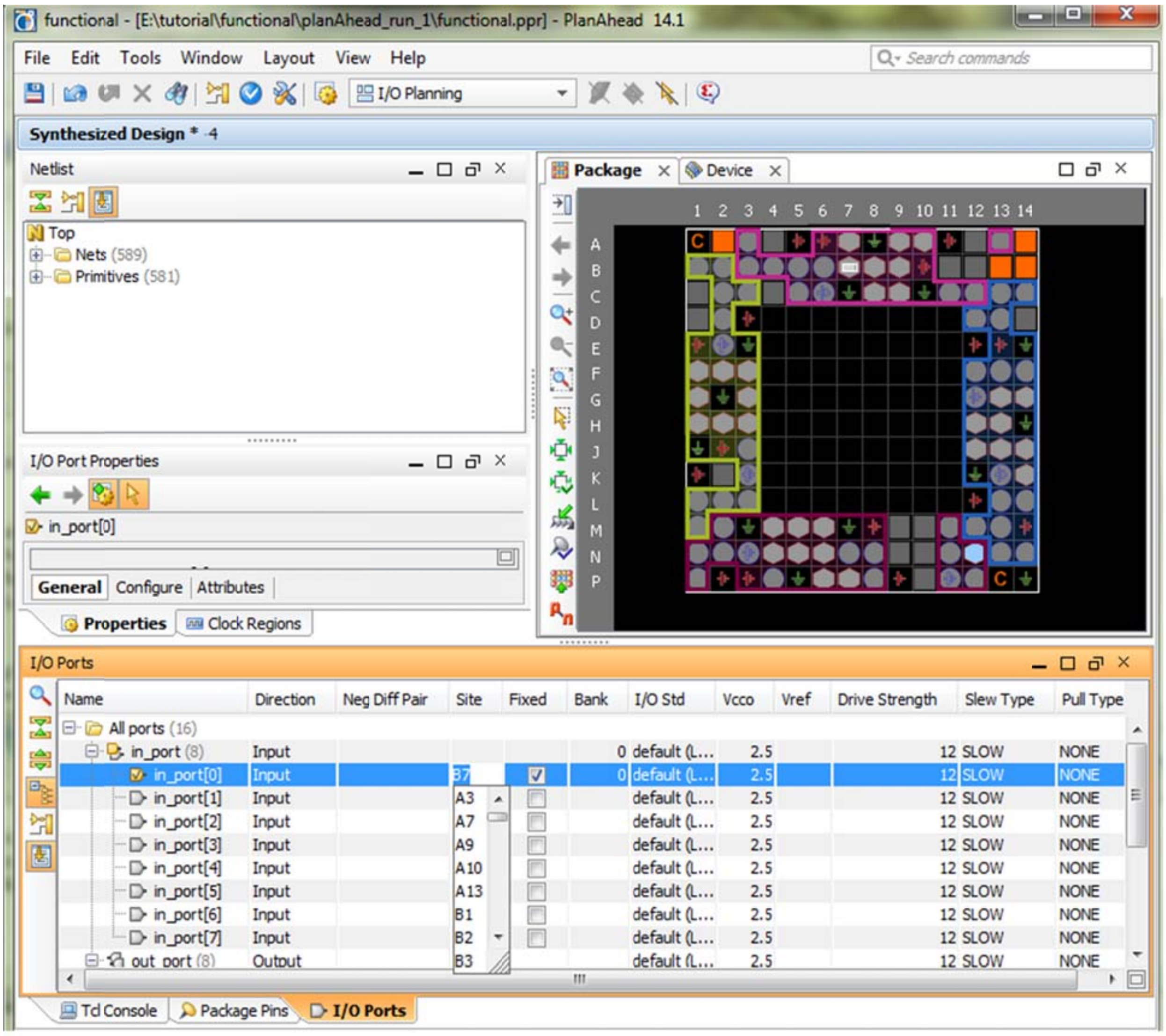
Assign I/O Locations:
We can use a text editor to edit I/O location constraints directly or use the PlanAhead to add and edit the pin locations and area group constraints defined in the NGD file. The PlanAhead software will write the constraints to the project UCF.

Adding the UCF is same as adding other source file:
a) Select Project > Add Source (or Add Copy of Source so that all HDL files will be copied to your project directory)
b) Select the top.ucf, and click Open.

If you want to assign pins using PlanAhead, do the following:
a) In the Hierarchy pane of the Project navigator Design panel, select the top module (top.v).
b) In the Processes pane, expand **User Constraints**, and double-click **I/O Pin Planning (PlanAhead) - Post-Synthesis.**
I/O pin planning can be performed either pre- or post-synthesis. Whenever possible, it is recommended that the process be run post-synthesis.


Processes: Top
Design Summary/Reports
Design Utilities
User Constraints
Create Timing Constraints
I/O Pin Planning (PlanAhead) - Pre-Synthesis
I/O Pin Planning (PlanAhead) - Post-Synthesis
Floorplan Area/IO/Logic (PlanAhead)
Synthesize - XST
Implement Design
Generate Programming File
Configure Target Device
Analyze Design Using ChipScope
Figure 7

This process launches the PlanAhead software in Figure 8.


functional - [E:\tutorial\functional\planAhead_run_1\functional.ppr] - PlanAhead 14.1
File Edit Tools Window Layout View Help
Search commands
I/O Planning
Synthesized Design * - 4
Netlist
Top
Nets (589)
Primitives (581)
I/O Port Properties
in_port[0]
General Configure Attributes
Properties Clock Regions
Package
Device
1 2 3 4 5 6 7 8 9 10 11 12 13 14
A B C D E F G H J K L M N P
I/O Ports
Tcl Console Package Pins I/O Ports

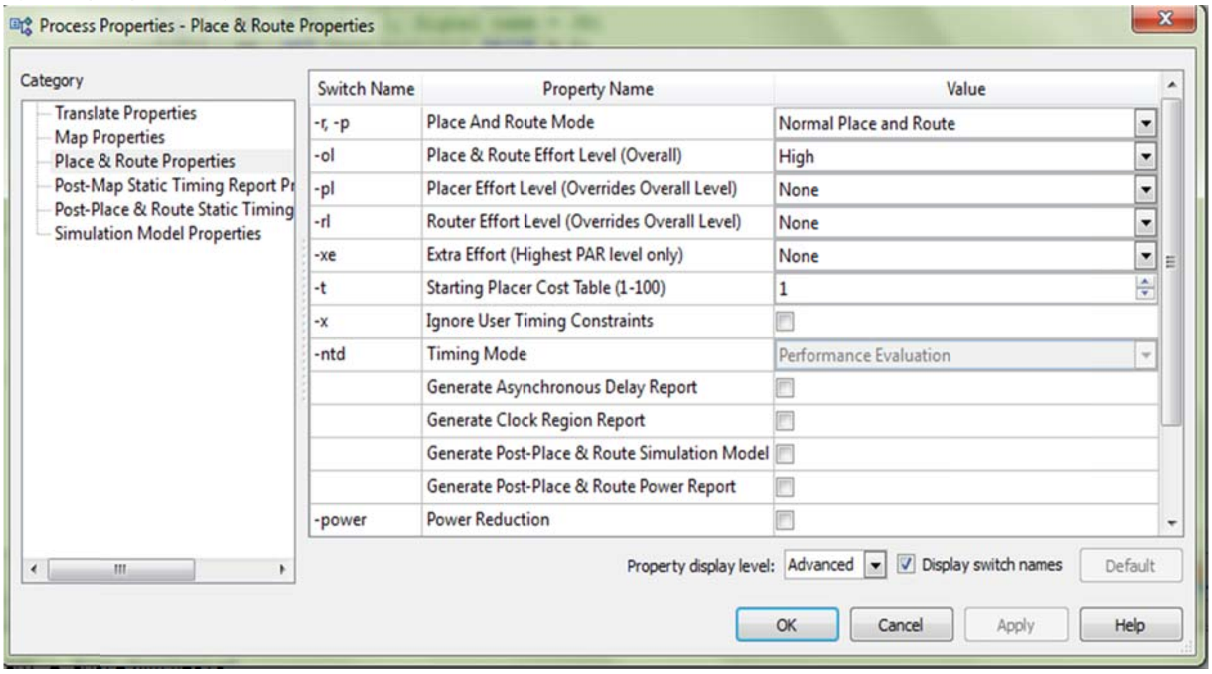
Name	Direction	Neg Diff Pair	Site	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type	Pull Type
All ports (16)											
in_port (8)	Input				0	default (L...	2.5		12	SLOW	NONE
in_port[0]	Input		B7	☑	0	default (L...	2.5		12	SLOW	NONE
in_port[1]	Input		A3	☐		default (L...	2.5		12	SLOW	NONE
in_port[2]	Input		A7	☐		default (L...	2.5		12	SLOW	NONE
in_port[3]	Input		A9	☐		default (L...	2.5		12	SLOW	NONE
in_port[4]	Input		A10	☐		default (L...	2.5		12	SLOW	NONE
in_port[5]	Input		A13	☐		default (L...	2.5		12	SLOW	NONE
in_port[6]	Input		B1	☐		default (L...	2.5		12	SLOW	NONE
in_port[7]	Input		B2	☐		default (L...	2.5		12	SLOW	NONE
out_port (8)	Output		B3			default (L...	2.5		12	SLOW	NONE

Figure 8

c) In the I/O Port tab, expand the in_port and out_port under All ports. You will now create pin assignments for the in_port and out_port signals.
d) You can type the location in the Site field, I/O standard, Vcco, Drive Strength Slew type for each Pin in the design. (All the information can be found in the UCF we provided.)
e) After the pins are locked down, select **File > Save Project**. The changes are saved in the top.ucf file.
f) To exit the PlanAhead software, select **File > Exit**.

Specifying Options:
The implementation properties control how the software maps, places, routes, and optimizes a design. To set the implementation properties, do the following:
a) In the View pane of the Project Navigation Design panel, select **Implementation**.
b) In the Hierarchy pane, select the top module (top.v).

c) In the Processes pane, right-click the **Implementation Design** process, and select **Process Properties**.
d) Ensure that you have set the Property display level to **Advanced**.
e) Set properties you want.



Process Properties - Place & Route Properties

Category
- Translate Properties
- Map Properties
- Place & Route Properties
- Post-Map Static Timing Report Pr
- Post-Place & Route Static Timing
- Simulation Model Properties

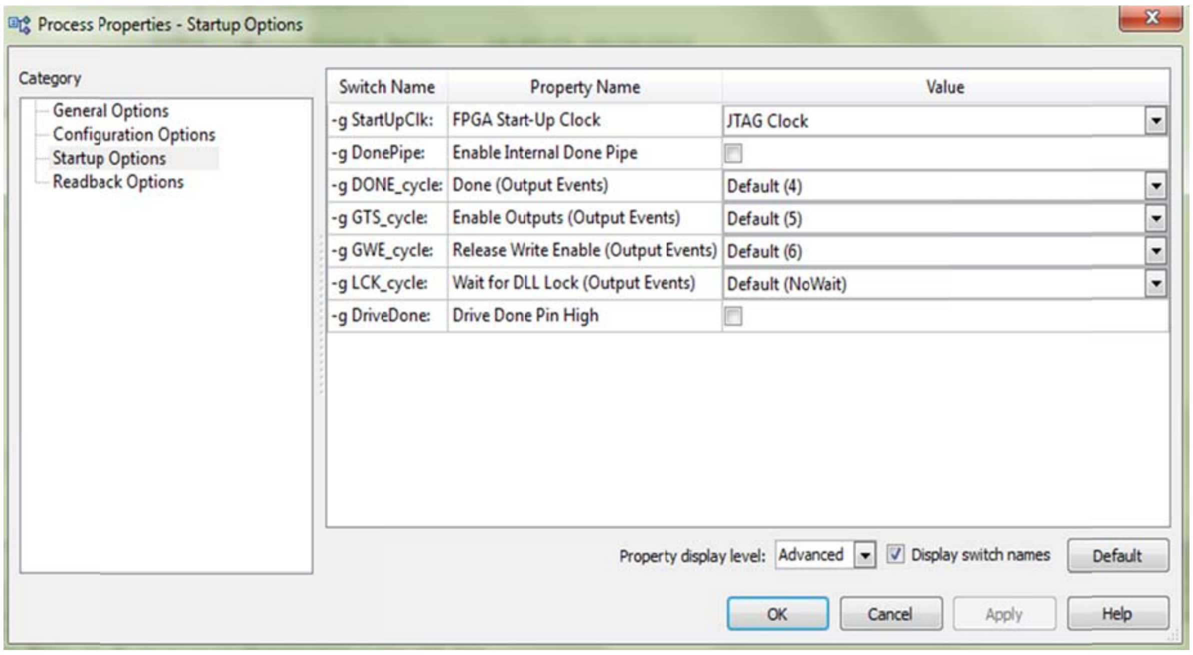
Switch Name	Property Name	Value
-r, -p	Place And Route Mode	Normal Place and Route
-ol	Place & Route Effort Level (Overall)	High
-pl	Placer Effort Level (Overrides Overall Level)	None
-rl	Router Effort Level (Overrides Overall Level)	None
-xe	Extra Effort (Highest PAR level only)	None
-t	Starting Placer Cost Table (1-100)	1
-x	Ignore User Timing Constraints	☐
-ntd	Timing Mode	Performance Evaluation
	Generate Asynchronous Delay Report	☐
	Generate Clock Region Report	☐
	Generate Post-Place & Route Simulation Model	☐
	Generate Post-Place & Route Power Report	☐
-power	Power Reduction	☐

Property display level: Advanced ☑ Display switch names

Figure 9

f) Click **OK** to exit the Process Properties dialog box.

Generating Programming File:
After analyzing the design, you need to create configuration data. A configuration bitstream is created for downloading to a target device or for formatting into a PROM programming file.
a) In the Hierarchy pane of the Project navigator Design panel, select the top module (top.v).
b) In the Processes pane, right-click **Generate Programming File**, and select **Process Properties**.
c) In the Process Properties dialog box, click the **Startup Options** category.
d) Change the PFGA Start-Up Clock property from **CCLK** to **JTAG Clock**.


Process Properties - Startup Options

Category
- General Options
- Configuration Options
- Startup Options
- Readback Options

Switch Name	Property Name	Value
-g StartUpClk:	FPGA Start-Up Clock	JTAG Clock
-g DonePipe:	Enable Internal Done Pipe	☐
-g DONE_cycle:	Done (Output Events)	Default (4)
-g GTS_cycle:	Enable Outputs (Output Events)	Default (5)
-g GWE_cycle:	Release Write Enable (Output Events)	Default (6)
-g LCK_cycle:	Wait for DLL Lock (Output Events)	Default (NoWait)
-g DriveDone:	Drive Done Pin High	☐

Property display level: Advanced ☑ Display switch names

Figure 10

e) Click **OK**.
f) In the Processes panel, double-click **Generate Programming File** to create a bitstream of this design.

Downloading Program on FPGA

To start iMPACT:
To start iMPACT from Project Navigator, double-click Manage Configuration Project (iMPACT) in the Processes pane in the Design panel, as shown in Figure 11.

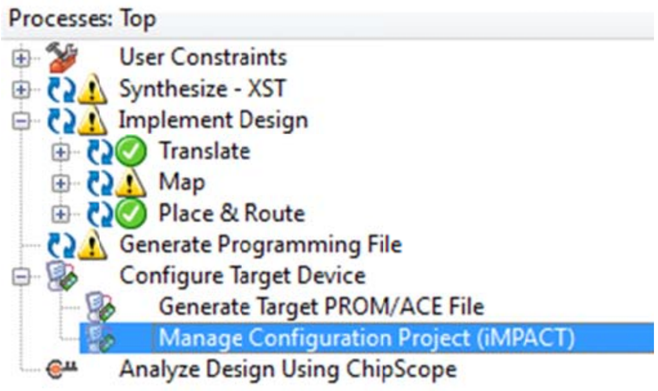

Processes: Top
- User Constraints
- Synthesize - XST
- Implement Design
  - Translate
  - Map
  - Place & Route
- Generate Programming File
- Configure Target Device
  - Generate Target PROM/ACE File
  - Manage Configuration Project (iMPACT)
- Analyze Design Using ChipScope

Figure 11

Specifying Boundary-Scan Configuration Mode:
In iMPACT, creating a new project includes specifying the configuration mode and the device to program. To select Boundary-Scan Mode, do the following:
a) Select **File > New Project**.

b) In the Automatically create and save a project dialog box, select **Yes**.
c) In the Welcome to iMPACT dialog box, select **Configure Devices using Boundary-Scan (JTAG)**.
d) Ensure that **Automatically connect to a cable and identify Boundary-Scan chain** is selected.
e) Make sure turn on the power of FPGA board, and Click **OK**.

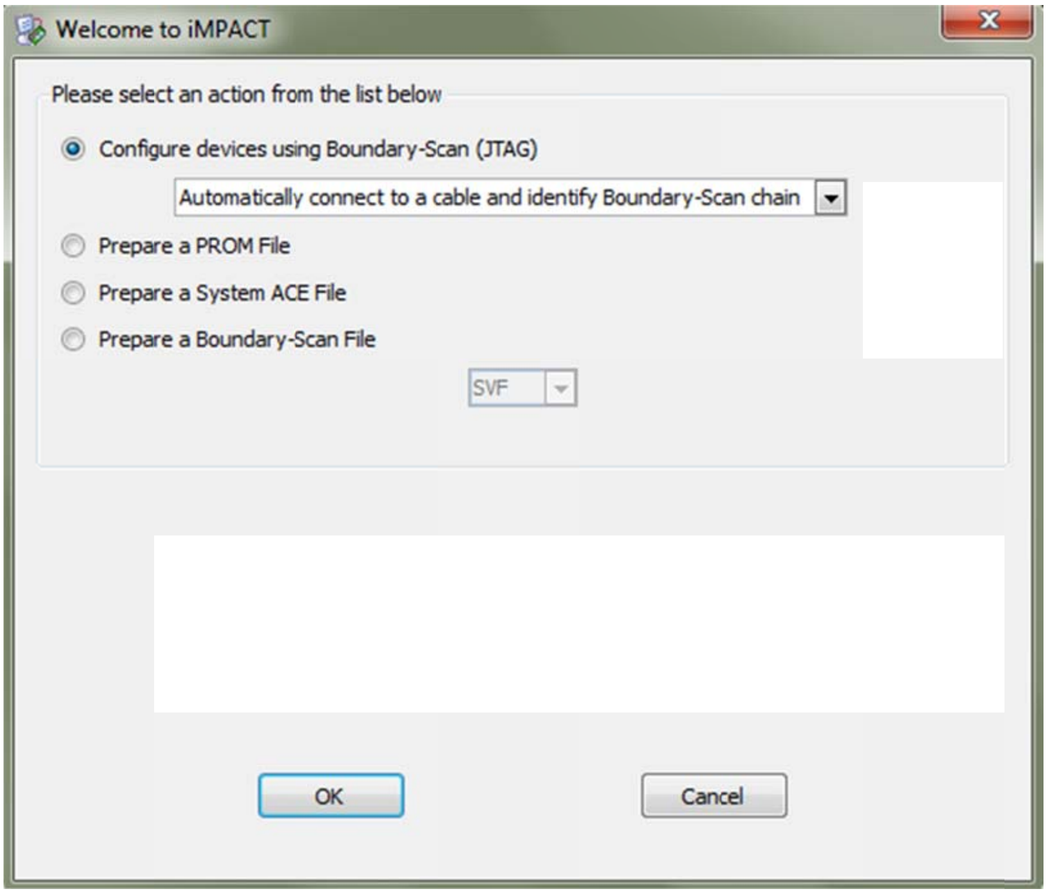
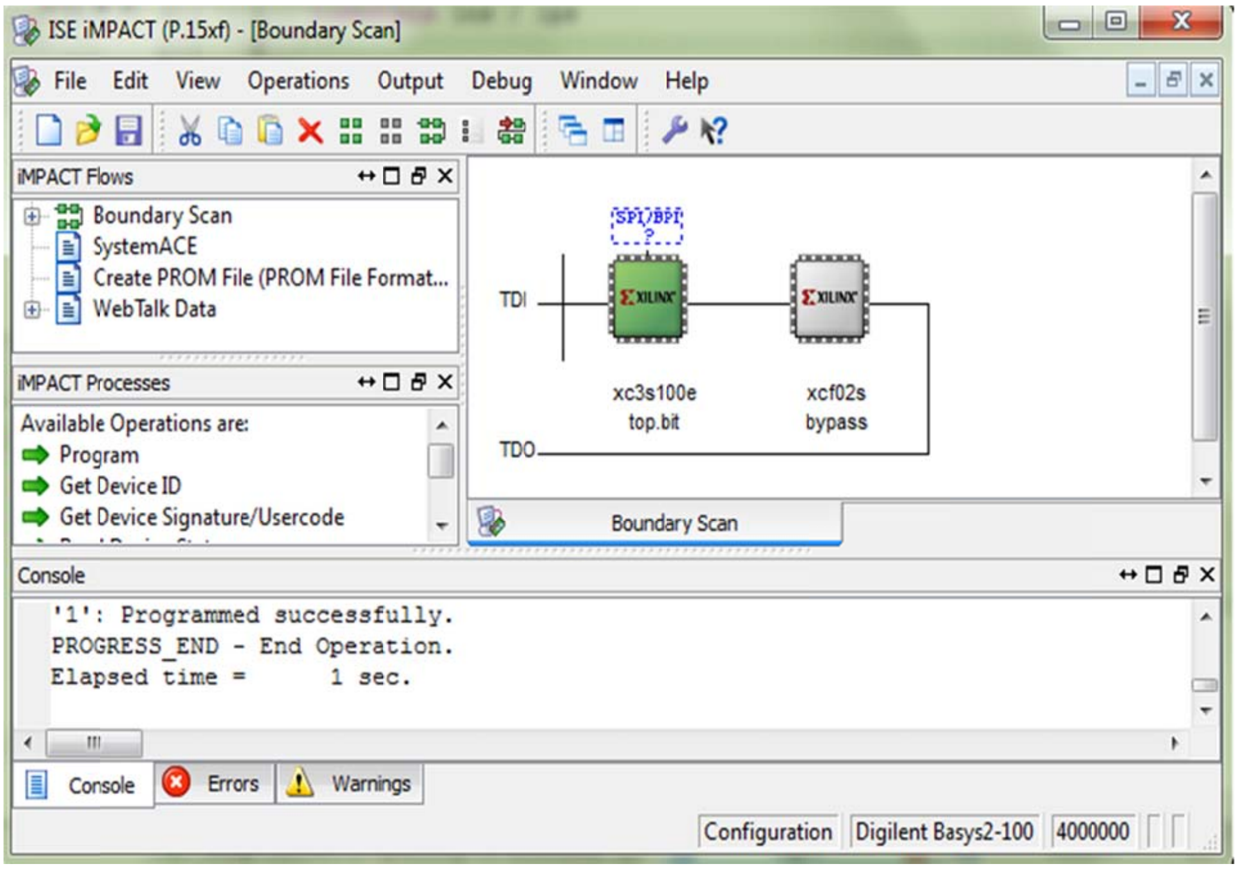

Welcome to iMPACT
Please select an action from the list below
Configure devices using Boundary-Scan (JTAG)
Automatically connect to a cable and identify Boundary-Scan chain
Prepare a PROM File
Prepare a System ACE File
Prepare a Boundary-Scan File
SVF
OK
Cancel

Figure 12

Performing Boundary-Scan Operations:
a) Right-click on the xc3s100e device, and select **Get Device ID**.
b) Right-click on the xc3s100e device, and select **Assign New Configuration File.** You need to browse and select the programming file (.bit) generated in step.
c) Right-click on the xc3s100e device again, and select **Program**.



```
'1': Programmed successfully.
PROGRESS_END - End Operation.
Elapsed time =      1 sec.
```

Figure 13

3.1.2 Trojan-Inserted Design Implementation

The procedure to implement Trojan-insert circuits on FPGA is the same as the procedure to implement Trojan-free circuits presented in 3.1.1. We simply need to insert the Trojan in HDL code, and then repeat the entire flow until downloading to FPGA board. By providing patterns to trigger Trojan inside, the faulty result induced by the Trojan will be detected.

3.2 Trojans for Power-based Detection Approach

An FPGA can be used to verify power-based detection approaches which detect Trojan-inserted circuits based on transient or dynamic power. Leakage power is very difficult to detect, because all LUTs are already in the FPGA and we cannot remove them or eliminate their leakage currents.

Since Trojan gates may create additional switching activities when proper patterns are applied, Trojan can be detected by measuring the transient current from a power pin. On the FPGA, the easiest way is to use one switch on the board to control an enable signal which can enable Trojan gates. If the Trojan is disabled, it will be always quiet no matter what patterns are applied. If the Trojan is enabled, it can produce switching activities when it is partially activated. By using this method, we can guarantee that both Trojan-free and Trojan-inserted circuits have exactly the same layout.

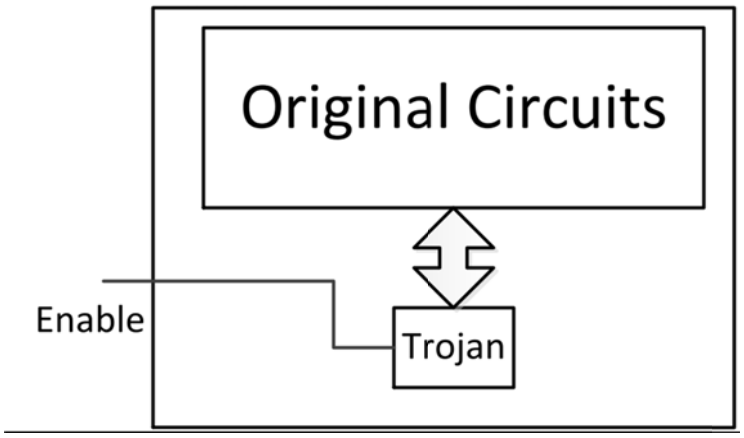


Figure 14

Therefore, one enable signal is required to be added to the original design, as shown in Figure 14. When you enable Trojans, the circuit becomes a Trojan-inserted FPGA. When you disable the Trojans, it is a Trojan-free FPGA.

3.3 Trojans for Delay-based Detection Method

For delay-based Trojan detection method, the change of path delay results from the extra capacitances induced by Trojan gates and their interconnections. We need two identical designs, one with Trojans and the other without. It is very difficult to control wire routing on an FPGA. Therefore, my strategy is to get a Trojan-inserted design first, and then disconnect the interconnections between Trojans LUTs and the original design to obtain the Trojan-free design.

3.3.1 Implement Trojan-Inserted Design

（Do the same thing, need to keep nets during synthesis）

We need to pay attention to the method of inserting the delay-based Trojan. Since many nets will be renamed or optimized during synthesis, we should set the do not touch attribute for these nets so that we can still find them after synthesis. Thus, when we start to design Trojans in HDL code, we need to keep all nets which connect between Trojans and original circuits so that we can identify and remove them in layout to obtain our Trojan-free design.

How to set don’t touch attribute:

The syntax for preventing optimization in Exemplar is as follows:

```
library ieee;
use ieee.std_logic_1164.all;

entity test is port(    a: in std_logic;
                        y : out std_logic);
                        attribute preserve_signal : boolean;
end test;

architecture beh of test is
      signal b : std_logic;
      attribute preserve_signal of b : signal is true;
begin
      b <= not a;
      y <= not b;
end beh;
```

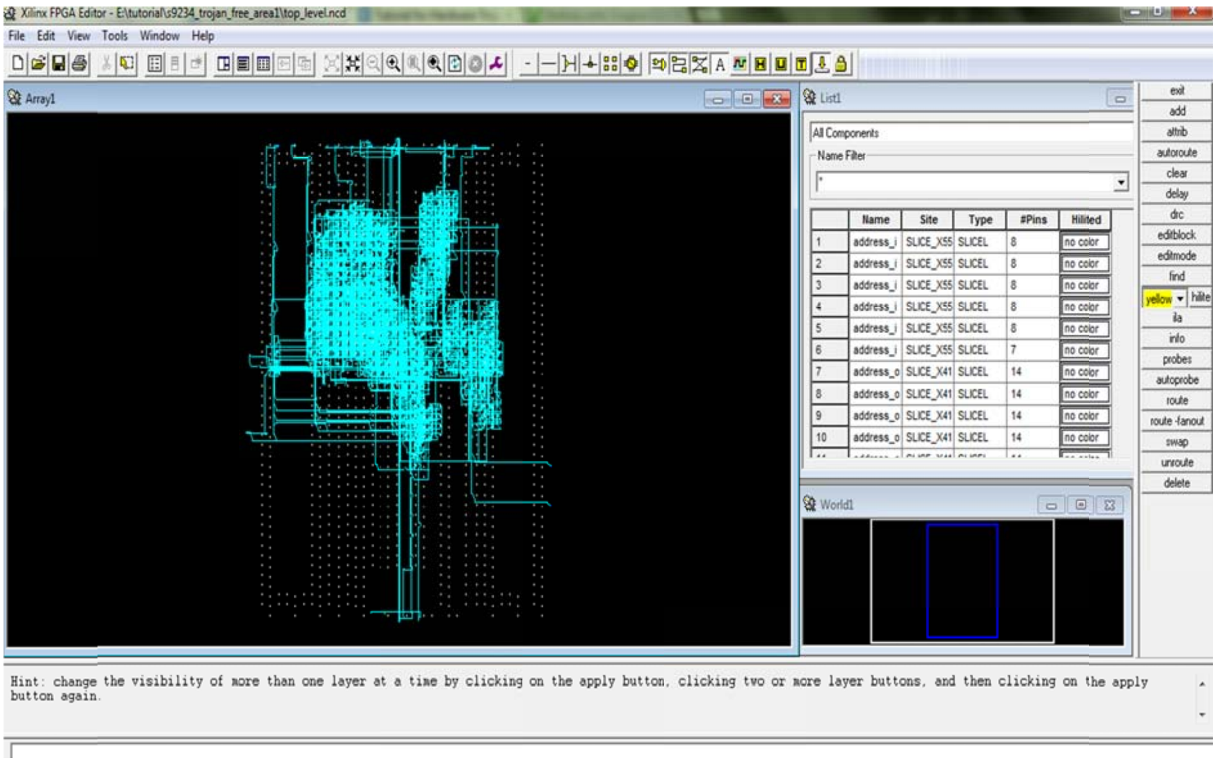
There are two highlighted commands in the example above which are needed to assure signal b will not be optimized during synthesis. Note that “don’t touch” operation is only valid in VHDL. Thus, for the module with Trojan, we need to use VHDL code to keep these nets.

To implement on FPGA:
After completing Trojan-inserted design, the implementation procedure is the same as presented in 3.1.1. Finally, the Trojan-inserted design is programmed on an FPGA.

3.3.2 Trojan-free Design Implementation

We already implemented the Trojan-inserted design on an FPGA so far. What we need to do is to disconnect the nets between Trojans and the original circuit. This task should be done in FPGA editor.

To start FPGA editor:
To launch FPGA editor, Select Tools > FPGA Editor > Post-Place & Route. This process launches the FPGA editor software as shown in Figure 15.



	Name	Site	Type	#Pins	Hilited
1	address_i	SLICE_X55	SLICEL	8	no color
2	address_i	SLICE_X55	SLICEL	8	no color
3	address_i	SLICE_X55	SLICEL	8	no color
4	address_i	SLICE_X55	SLICEL	8	no color
5	address_i	SLICE_X55	SLICEL	8	no color
6	address_i	SLICE_X55	SLICEL	7	no color
7	address_o	SLICE_X41	SLICEL	14	no color
8	address_o	SLICE_X41	SLICEL	14	no color
9	address_o	SLICE_X41	SLICEL	14	no color
10	address_o	SLICE_X41	SLICEL	14	no color

```
Hint: change the visibility of more than one layer at a time by clicking on the apply button, clicking two or more layer buttons, and then clicking on the apply
button again.
```

Figure 15

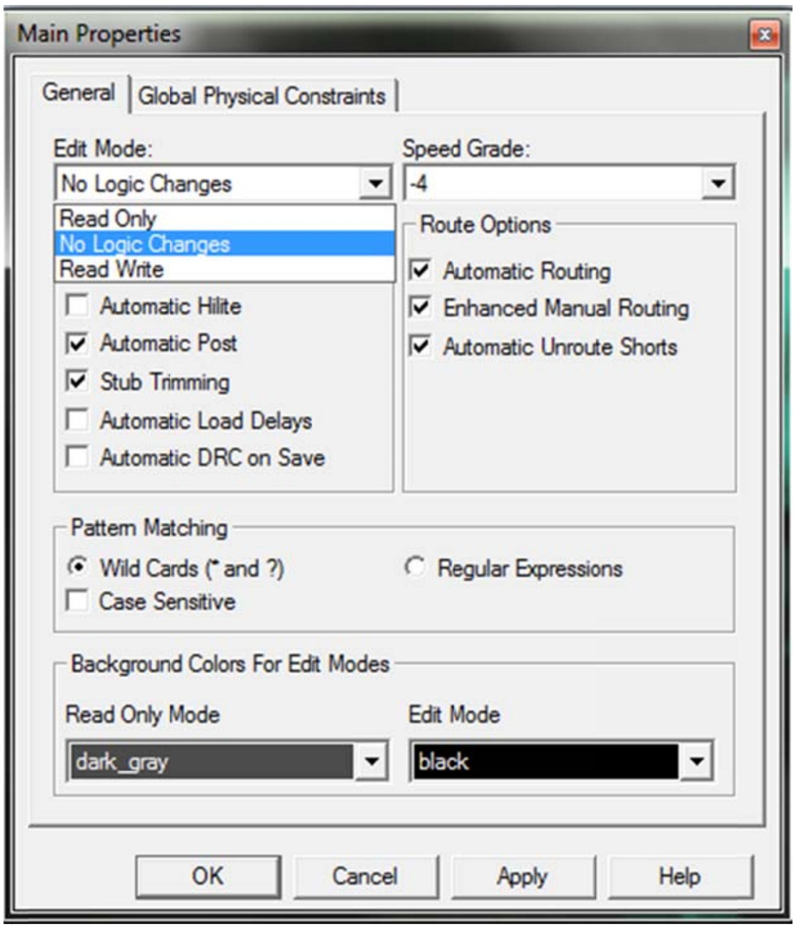
Change Edit Mode:

Before we make any changes, we need to change the edit mode.

a) Select **File > Main Properties,** Main Properties window appear.

b) Expand **Edit Mode**, and select **Read Write**.

c) Click **OK**.


Main Properties

General | Global Physical Constraints

Edit Mode: No Logic Changes
- Read Only
- No Logic Changes
- Read Write

Speed Grade: -4

- [ ] Automatic Hilite
- [x] Automatic Post
- [x] Stub Trimming
- [ ] Automatic Load Delays
- [ ] Automatic DRC on Save

Route Options
- [x] Automatic Routing
- [x] Enhanced Manual Routing
- [x] Automatic Unroute Shorts

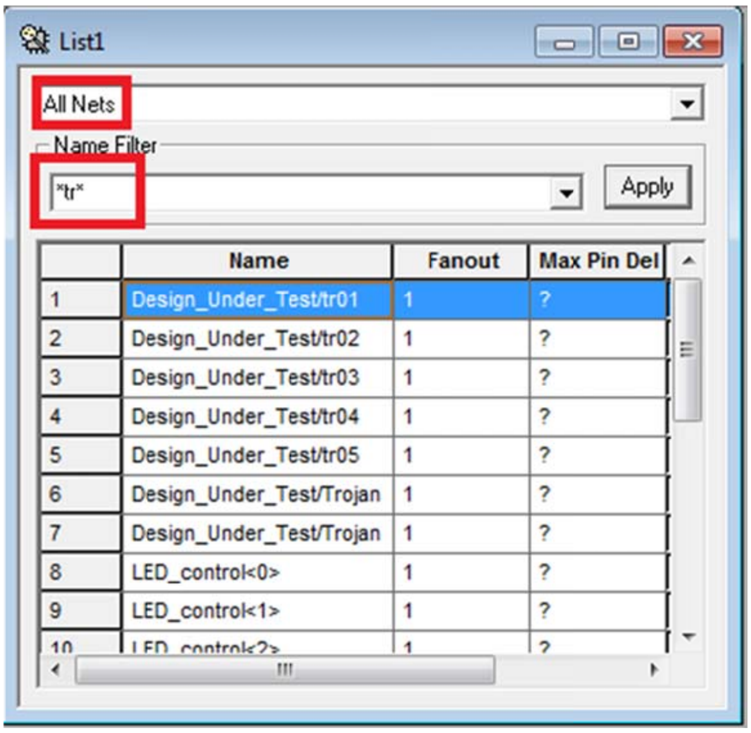
Pattern Matching
- [✓ Wild Cards (* and ?)] Regular Expressions
- [ ] Case Sensitive

Background Colors For Edit Modes
- Read Only Mode: dark_gray
- Edit Mode: black

Figure 16

Modify Routing:

a) Expand to select **All Nets** and then Type *tr* in **Name Filter** to search all nets containing string “tr” in their name.
b) Click **Apply** and all specified nets are listed in the table below.
c) Double-click to select the net you want to change in the table**.** For example, when we double-click **Design_under_Test/tr01**, the net is shown in the array window and highlighted in red.



tr

	Name	Fanout	Max Pin Del
1	Design_Under_Test/tr01	1	?
2	Design_Under_Test/tr02	1	?
3	Design_Under_Test/tr03	1	?
4	Design_Under_Test/tr04	1	?
5	Design_Under_Test/tr05	1	?
6	Design_Under_Test/Trojan	1	?
7	Design_Under_Test/Trojan	1	?
8	LED_control<0>	1	?
9	LED_control<1>	1	?
10	LED_control<2>	1	?

Figure 17

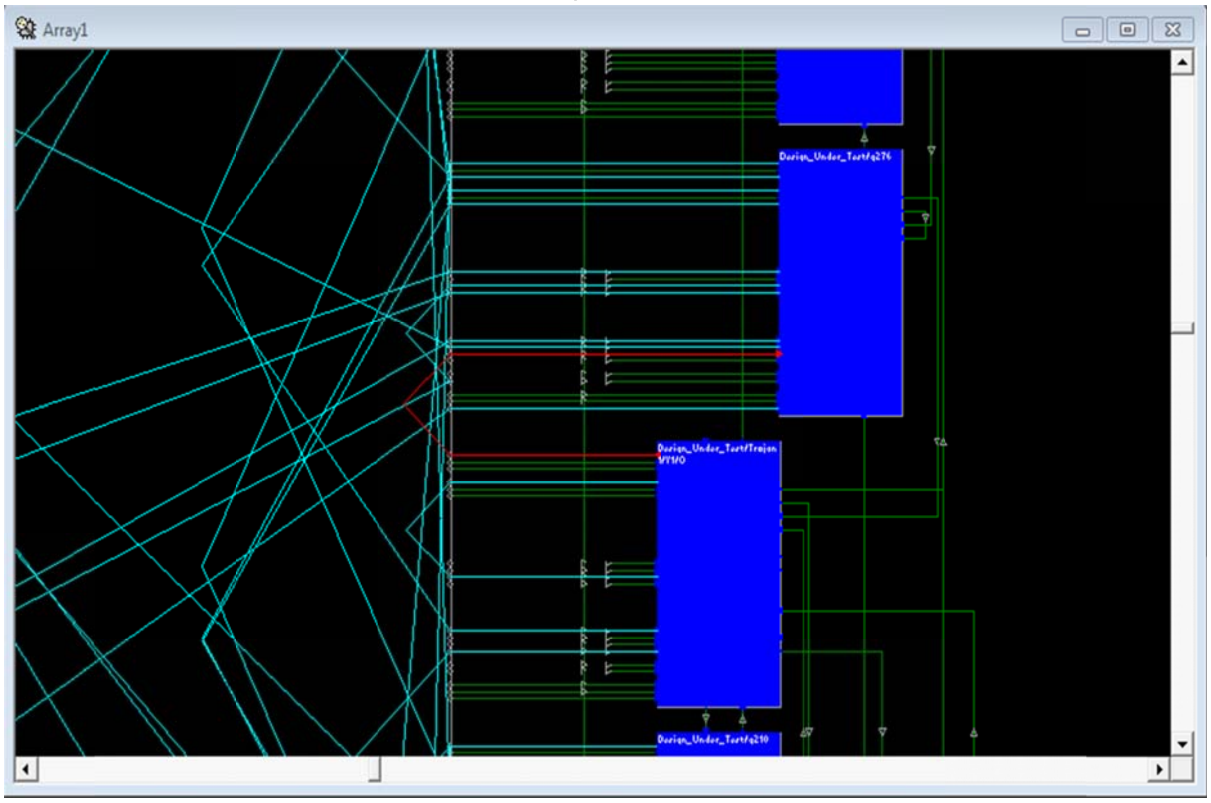


Figure 18

d) Select **Tools > Route > Unroute** to unroute the net which is indicated by the green line.

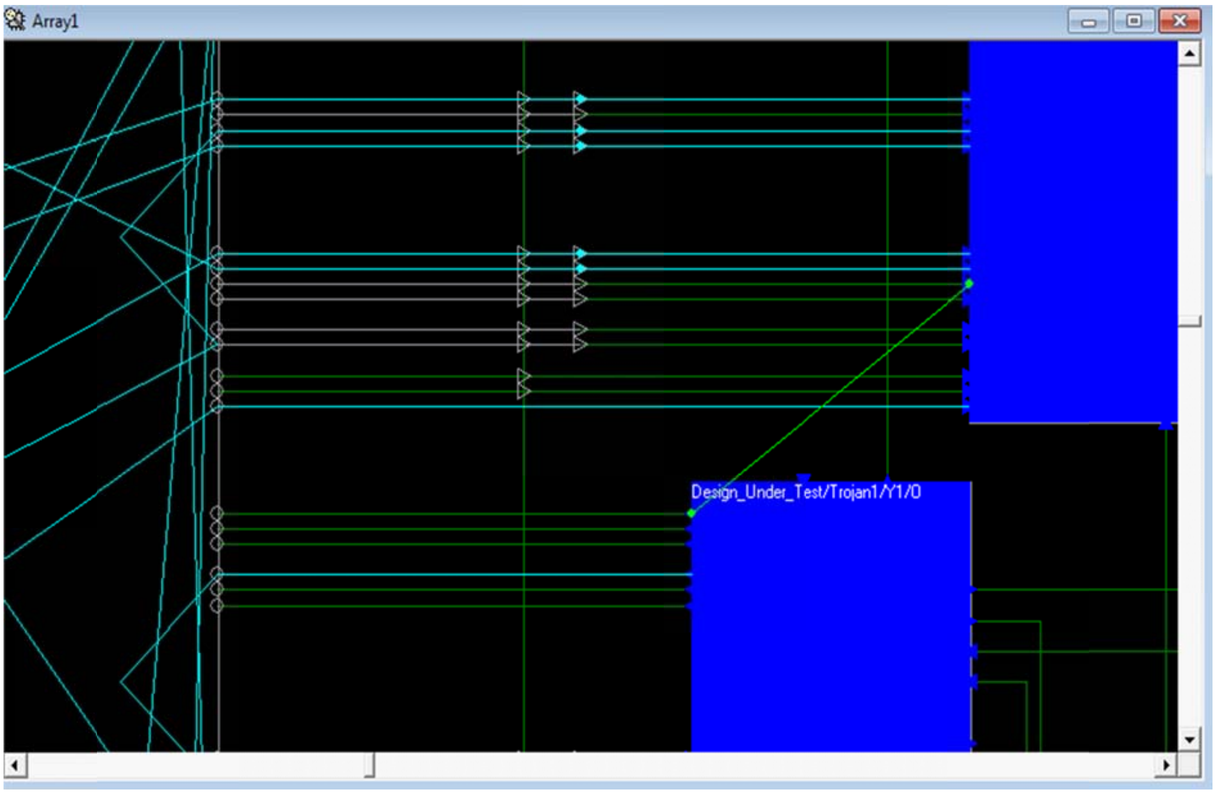
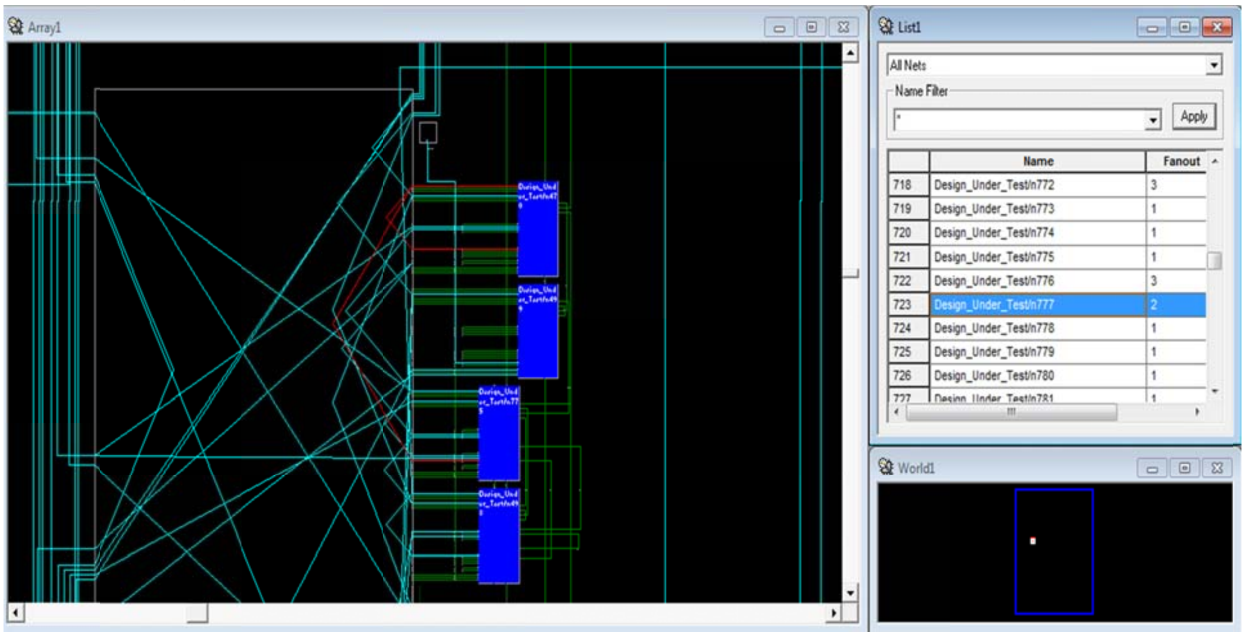


Figure 19

Manual Routing:
If the fanout of the unroute net is 1, it is done. If it is more than 1, that means this net also connect to other genuine gates besides the Trojan gates. Therefore, we need to manually route these genuine gates. We take net **Design_under_Test/n777** for example. The original routing is shown in Figure 20.



	Name	Fanout
718	Design_Under_Test/n772	3
719	Design_Under_Test/n773	1
720	Design_Under_Test/n774	1
721	Design_Under_Test/n775	1
722	Design_Under_Test/n776	3
723	Design_Under_Test/n777	2
724	Design_Under_Test/n778	1
725	Design_Under_Test/n779	1
726	Design_Under_Test/n780	1
727	Design_Under_Test/n781	1

Figure 20

a) Double-click the net to select it.
b) Select **Tools > Route > Unroute** to unroute the net.
c) If the net is selected, we can see the triangle at each pin. The triangle pointing to the cell means this is an input port. Otherwise, it is an output pin. We see that this net connects to one output port and two input ports, since its fanout is 2. Suppose the gates in cell **Design_Under_Test/n775** (left) is Trojan gate, we only need to route two ports in cell **Design_Under_Test/n470** (right) to assure original functionality.

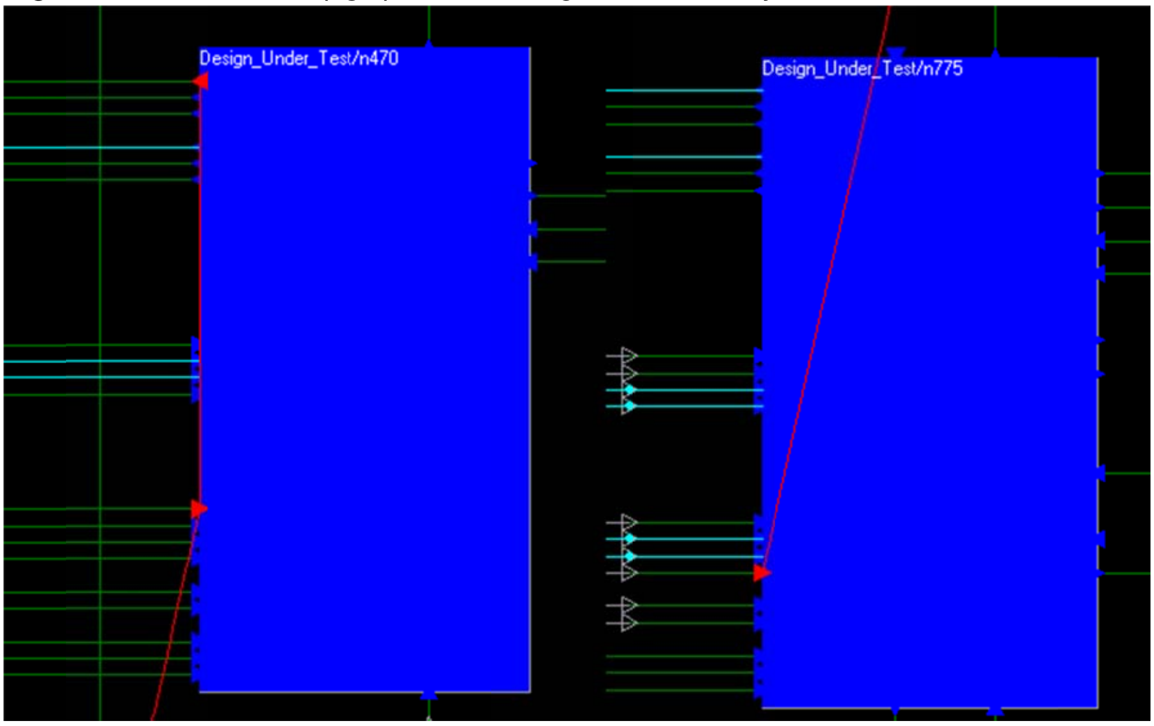


Figure 21

If you are not sure which is which, you can double-click the cell. A new window appears.

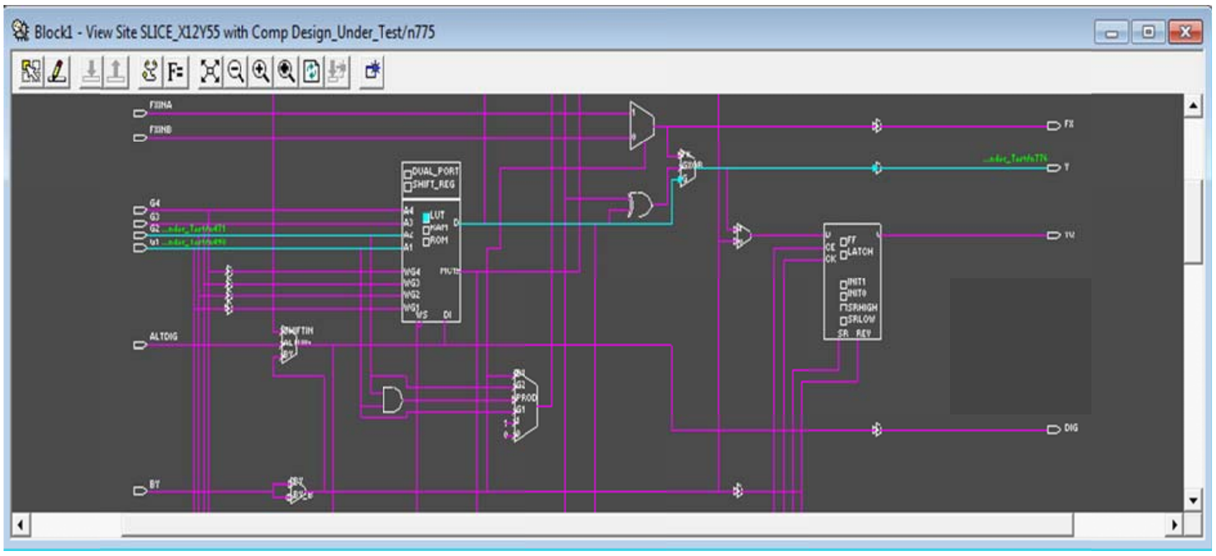
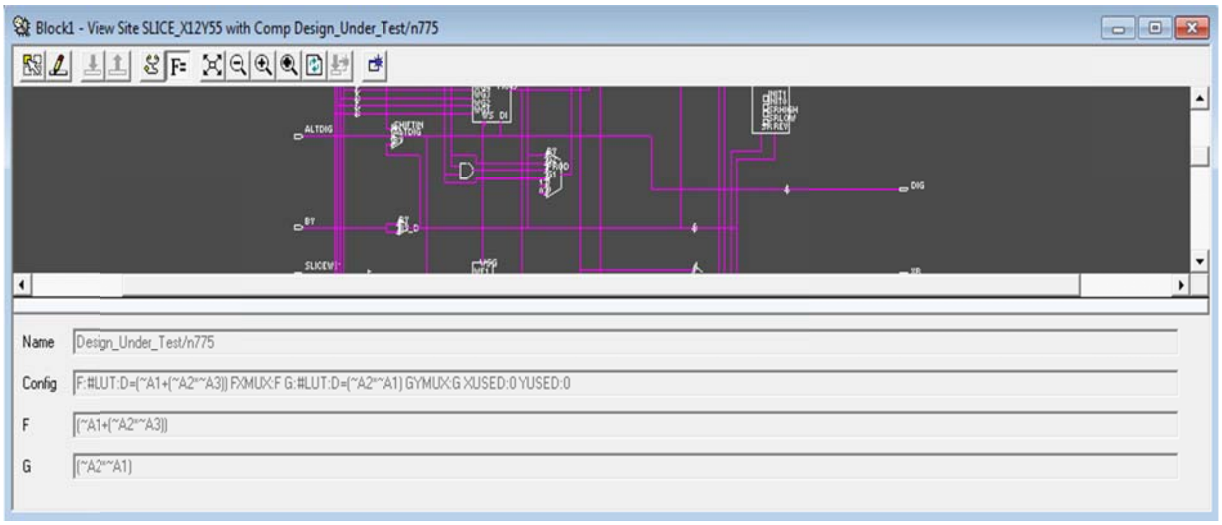


Figure 22

You can check interconnections inside and check the boolean equation by click **F=**.


Block1 - View Site SLICE_X12Y55 with Comp Design_Under_Test/n775
F=
ALTDIG
BY
SLICEWE1
DIG

```
Name    Design_Under_Test/n775
Config  F:#LUT:D=(~A1+(~A2*~A3)) FXMUX:F G:#LUT:D=(~A2*~A1) GYMUX:G XUSED:0 YUSED:0
F       (~A1+(~A2*~A3))
G       (~A2*~A1)
```

Figure 23

d) After you locate the ports you will route, you click to select the output port triangle and then select one input port triangle, as shown in Figure 24. You must follow this sequence. If you select input port first, manual routing will fail. Please note that manual routing can only work between two ports (one input port and one output port) every time. If there are several input ports, you need to do this multiple times to connect all input ports.

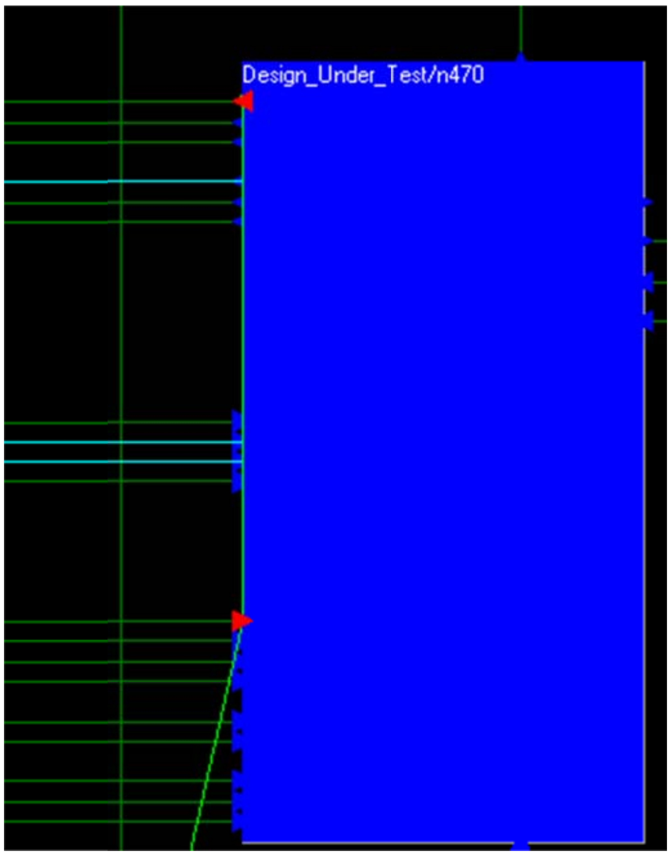


Figure 24

Select > Route > Manual Route, these two ports are routed.

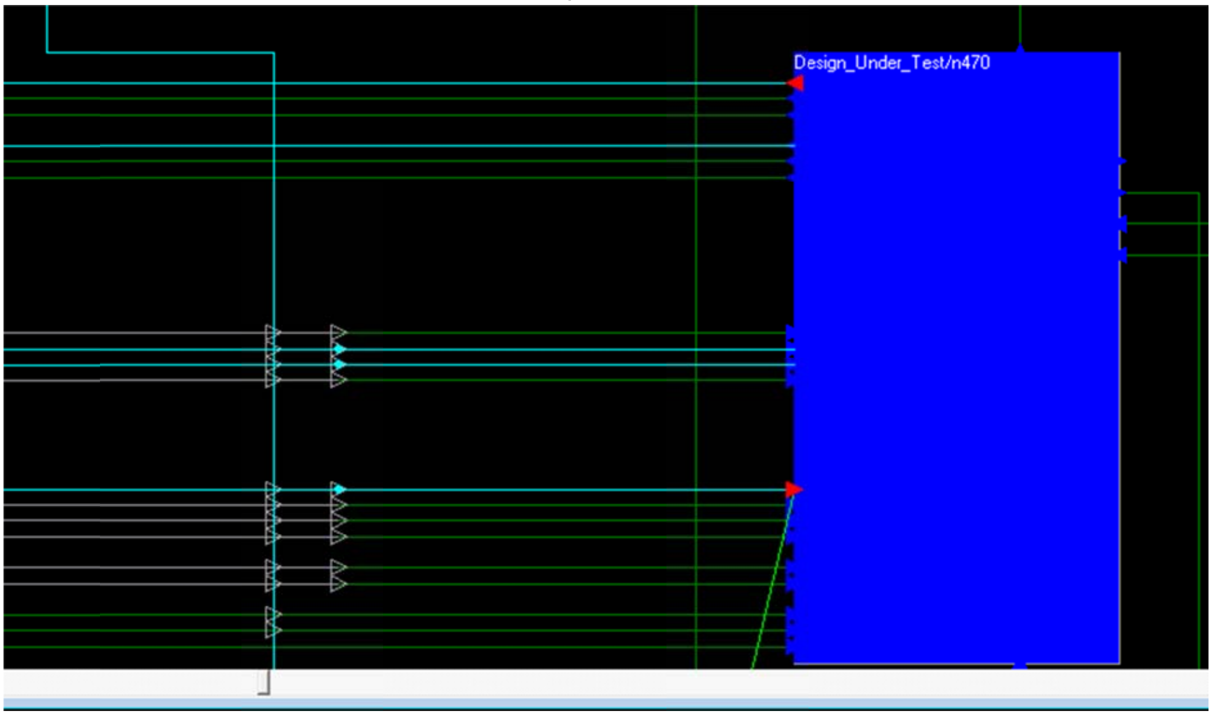


Figure 25

Save and exit:

a) Repeat the previous process until the Trojans are disconnected from the original design.

b) Select **File > Save.**

c) Select **File > Exit.**

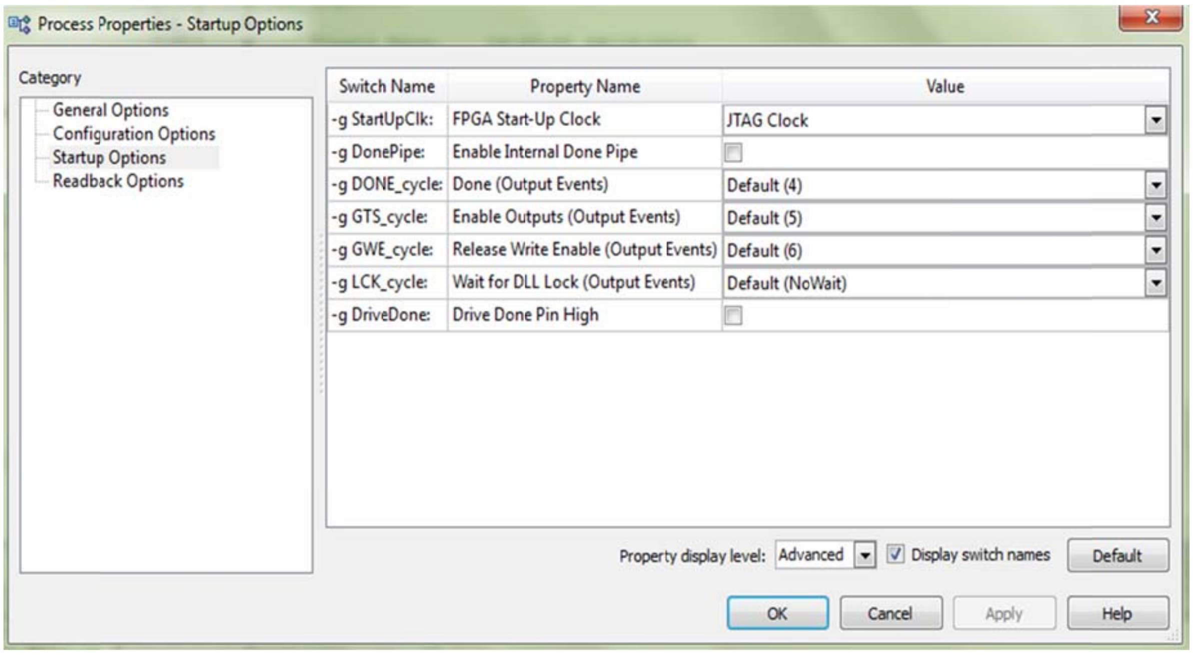
Generate Programming File Again:

a) In the Hierarchy pane of the Project navigator Design panel, select the top module (top.v).

b) In the Processes pane, right-click **Generate Programming File**, and select **Process Properties**.

c) In the Process Properties dialog box, click the **Startup Options** category.

d) Change the PFGA Start-Up Clock property from **CCLK** to **JTAG Clock**.



Switch Name	Property Name	Value
-g StartUpClk:	FPGA Start-Up Clock	JTAG Clock
-g DonePipe:	Enable Internal Done Pipe	☐
-g DONE_cycle:	Done (Output Events)	Default (4)
-g GTS_cycle:	Enable Outputs (Output Events)	Default (5)
-g GWE_cycle:	Release Write Enable (Output Events)	Default (6)
-g LCK_cycle:	Wait for DLL Lock (Output Events)	Default (NoWait)
-g DriveDone:	Drive Done Pin High	☐

Figure 26

e) Click **OK**.

f) In the Processes panel, double-click **Generate Programming File** to create a bitstream of this design.

Downloading on the FPGA:

After downloading the program on the FPGA by using method in 3.1.1, the implementation for delay-based Trojan is done.