Faculty of Electrical & Electronics Engineering
BEE3233 Electronics System Design

Laboratory 1: FPGA Design of a 2-bit Multiplier

Mapping CO, PO, Domain, KI : CO2,PO3,P5,CTPS5
CO2: Construct logic circuit using HDL.[PO3, P5, CTPS3] CO3: Design a digital system using combinational & sequential (medium scale integrated logic) MSI component. PO3: Identify, formulate and provide effective solution to engineering problem P5: Complex Overt Response CTPS3: Ability to get ideas and find alternative solutions

Learning Outcomes:

a) Getting familiar with Spartan 3E FPGA board.
b) Design the full adder schematic and Verilog using ISE, compile and simulate for Xilinx Spartan FPGA
c) Develop a User Constraint File "ucf" that maps the input and output signals to the Spartan 3E FPGA
d) Learn how to program the full Adder into the FPGA Spartan 3E board
e) Test the Results
f) Repeat the same process for Full Adder with Verilog instead of Schematic
g) Test your understanding by applying what you learned in programming a 4-bits Adder/Subtractor

General Xilinx Tips

1. SAVE EARLY AND OFTEN (in your own memory device!!)

   Xilinx is notorious for crashing at the most inopportune times. Do yourself a favor and save.

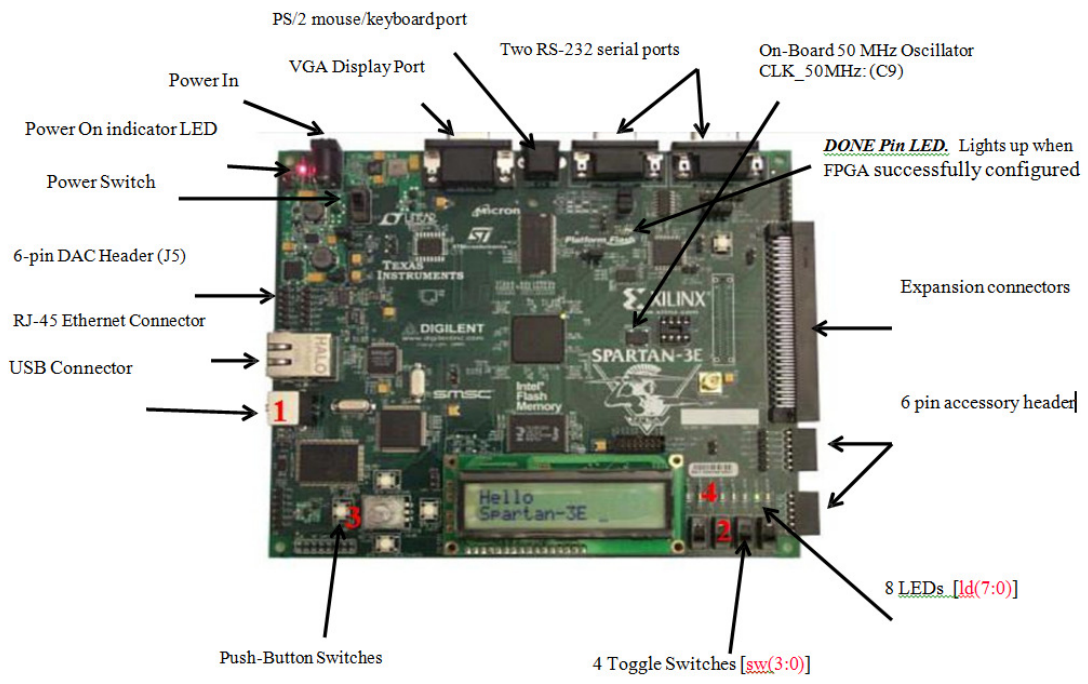
2. At the end of a lab session (or any work session), archive your project using the Xilinx utility (this will ensure you save everything), and save this zip archive on your ENIAC drive or on a flash drive. Do NOT assume files will remain on the lab computers or that "your" computer will be available at a later time.

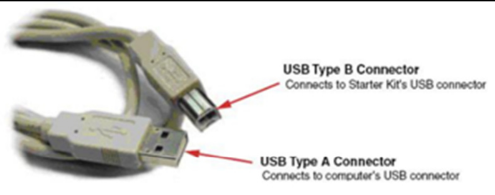
3. Make sure all components are connected. Loose wires are a frequent cause of problems.

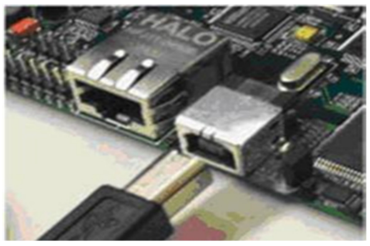
4. Try your hand at debugging first before calling me ☺. You will learn a lot by struggling through problems that seem hard at first.
5. Read all instructions carefully before starting the lab. Often, there will be a little detail that ends up being very important.
6. Make sure you test all important cases, particularly edge/corner cases. You can be sure that your TA will test these as part of the demo.

Introduction – Getting to know FPGA Spartan-3E starter board

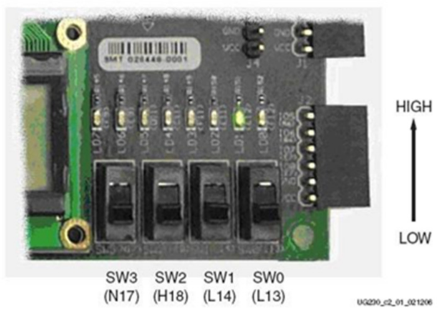
The Spartan-3E Starter Kit board is more advanced and complex compared to other Spartan development boards. The advantage of this board is that it is programmed through USB port. JTAG port is used to program previous versions. A list of the key features and their location on the board is listed below:



Item 1: USB Connector The board comes with power supply and USB cable. The standard USB Type A/Type B cable, similar to the one shown in Figure 1-2. The actual cable color might vary from the picture. The wider and narrower Type A connector fits the USB connector at the back of the	 USB Type B Connector Connects to Starter Kit's USB connector USB Type A Connector Connects to computer's USB connector

computer. After installing the Xilinx software, connect the square Type B connector to the Spartan-3E Starter Kit board, as shown. The USB connector is on the left side of the board, immediately next to the Ethernet connector. When the board is powered on, the Windows operating system should recognize and install the associated driver software.	

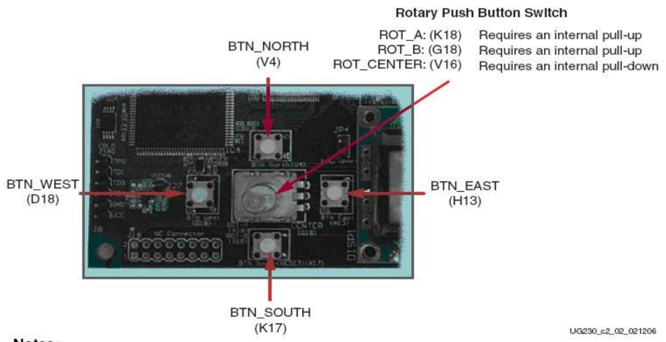
When the USB cable driver is successfully installed and the LED lights up, indicating a good connection.

Item 2 Slide Switches Locations and Labels The Spartan-3E Starter Kit board has four slide switches, as shown in Figure 1-3. The slide switches are located in the lower right corner of the board and are labeled SW3 through SW0. Switch SW3 is the left-most switch, and SW0 is the right-most switch. Operation When in the UP or ON position, a switch connects the FPGA pin to 3.3V, a logic High. When DOWN or in the OFF position, the switch connects the FPGA pin to ground, a logic Low.	 HIGH LOW SW3 (N17) SW2 (H18) SW1 (L14) SW0 (L13)

UCF Location Constraints

The UCF constraints for the four slide switches are listed below, including the I/O pin assignment and the I/O standard used. The PULLUP resistor is not required, but it defines the input value when the switch is in the middle of a transition. Verify the location of the switches on the board.

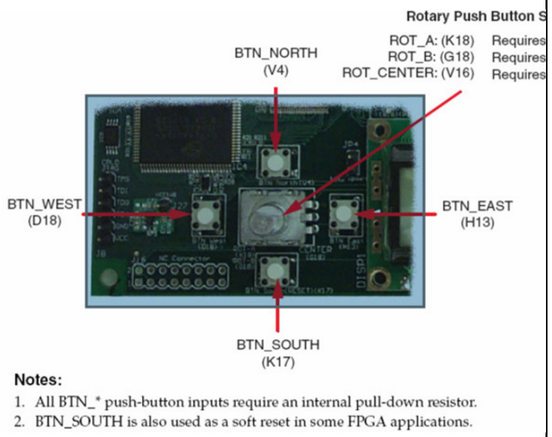
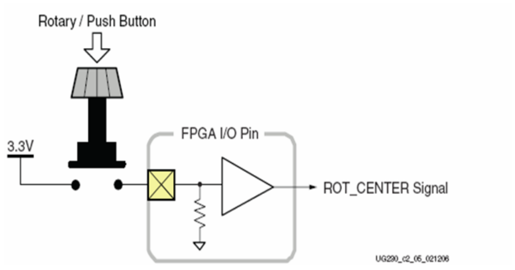
NET "SW<0>" LOC = "L13" NET "SW<1>" LOC = "L14" NET "SW<2>" LOC = "H18" NET "SW<3>" LOC = "N17"

***Item 3* Push-Button Switches** **Locations and Labels** The Spartan-3E Starter Kit board has four momentary-contact push-button switches, as shown. The push buttons are located in the lower left corner of the board and are labeled BTN_NORTH, BTN_EAST, BTN_SOUTH, and BTN_WEST. The FPGA pins that connect to the push buttons appear in parentheses.	 Rotary Push Button Switch ROT_A: (K18) Requires an internal pull-up ROT_B: (G18) Requires an internal pull-up ROT_CENTER: (V16) Requires an internal pull-down BTN_NORTH (V4) BTN_WEST (D18) BTN_EAST (H13) BTN_SOUTH (K17) UG230_c2_02_021206 **Notes:** 1. All BTN_* push-button inputs require an internal pull-down resistor. 2. BTN_SOUTH is also used as a soft reset in some FPGA applications.
Operation Pressing a push button connects the associated FPGA pin to 3.3V, as shown. Use an internal pull- down resistor within the FPGA pin to generate a logic Low when the button is not pressed. shows how to specify a pull-down resistor within the UCF. There is no active debouncing circuitry on the push button.	

UCF Location Constraints

These are the UCF constraints for the four push-button switches, including the I/O pin assignment and the I/O
standard used, and defines a pull-down resistor on each input.

NET "BTN_EAST"   LOC = "H13"
NET "BTN_NORTH" LOC = "V4" NET "BTN_SOUTH" LOC = "K17" NET "BTN_WEST" LOC = "D18"

***Item 3* Rotary Push-Button Switch** **Locations and Labels** The rotary push-button switch is located in the center of the four individual push-button switches, as shown. The switch produces three outputs. The two shaft encoder outputs are ROT_A and ROT_B. The center push-button switch is ROT_CENTER.	 Rotary Push Button S ROT_A: (K18) Requires ROT_B: (G18) Requires ROT_CENTER: (V16) Requires BTN_NORTH (V4) BTN_WEST (D18) BTN_EAST (H13) BTN_SOUTH (K17) **Notes:** 1. All BTN_* push-button inputs require an internal pull-down resistor. 2. BTN_SOUTH is also used as a soft reset in some FPGA applications.
Operation The rotary push-button switch integrates two different functions. The switch shaft rotates and outputs values whenever the shaft turns. The shaft can also be pressed, acting as a push-button switch. Push-Button Switch Pressing the knob on the rotary/push-button switch connects the associated FPGA pin to 3.3V, as shown.	 Rotary / Push Button 3.3V FPGA I/O Pin ROT_CENTER Signal UG230_c2_05_021206

UCF Location Constraints

the UCF constraints for the four push-button switches, including the I/O pin assignment and the I/O standard used, and defines a pull-down resistor on each input.

```
NET "ROT_A"     LOC = "K18" NET "ROT_B"     LOC = "G18"
NET "ROT_CENTER" LOC = "V16"
```

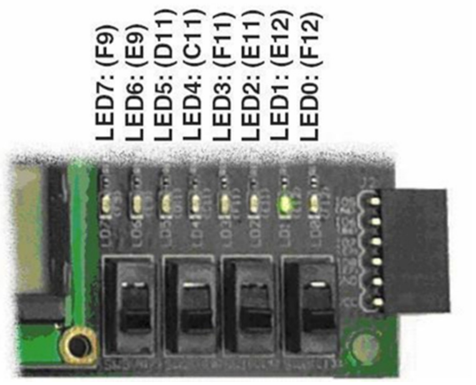
Item 3 **Discrete LEDs**

Locations and Labels

The Spartan-3E Starter Kit board has eight individual surface-mount LEDs located above the slide switches as shown. The LEDs are labeled LED7 through LED0.
LED7 is the left-most LED, LED0 the right-most LED.

Operation

Each LED has one side connected to ground and the other side connected to a pin on the Spartan-3E device via a 390Ωcurrent limiting resistor. To light an individual LED, drive the associated FPGA control signal High.



UCF Location Constraints

The UCF constraints for the four push-button switches, including the I/O pin assignment, the I/O standard used,
the output slew rate, and the output drive current.

NET "LED<7>" LOC = "F9" NET "LED<6>" LOC = "E9" NET "LED<5>" LOC = "D11" NET "LED<4>" LOC = "C11" NET "LED<3>" LOC = "F11" NET "LED<2>" LOC = "E11" NET "LED<1>" LOC = "E12" NET "LED<0>" LOC = "F12"

Please refer the reference manual for any additional information:
http://www.digilentinc.com/Products/Detail.cfm?Prod=S3EBOARD&Nav1=Products&Nav2=Programmable

Part 1 – Implement Full Adder schematic for Spartan 3E FPGA board:

1. set up a project correctly to reflect Spartan 3 FPGA board

- Select Start
- All Programs
- Xilinx
- Project Navigator

2. Create a new project by selecting:

File from the main menu
New Project

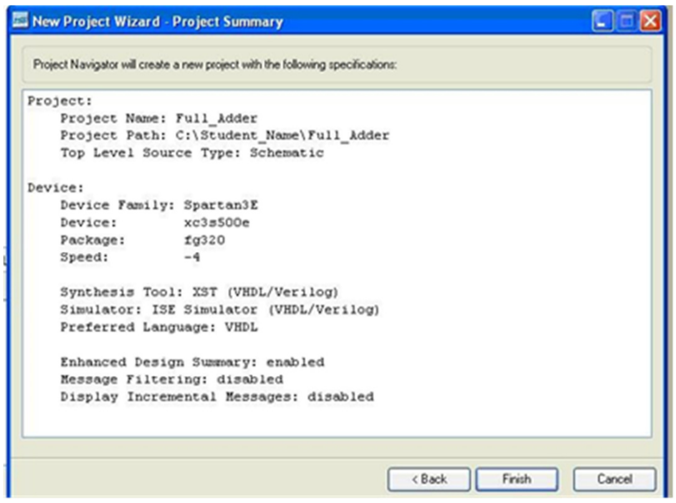
- In the New Project window, name your project Full_Adder in the project name text box.
- In the Project Location selection box, enter the folder or directory
- where your project will be saved. Use your name as for Student_Name and locate the place where you want to save all your files (i.e C:\)
- Under Top-Level Source Type, select Schematic and click Next as shown in figure 1-3

3. We will design our full adder for a particular device "Spartan 3E FPGA".

- Product Category: General Purpose
- Device Family: Choose Spartan 3E, the device we will be using.
- Device: XC3500E, the specific Spartan 3E device we use. This is actually printed (very small) on the FPGA core.
- Package: FG320, this is the package type of our device (Ball Grid Array, 320 pins)
- Speed Grade: The speed grade for this device is -4.

4. Select New Source and create a schematic source file by simply selecting Next> and later this window will allow us to add existing files.

5. Just review the information listed in the figure and make sure it matches the information in the window



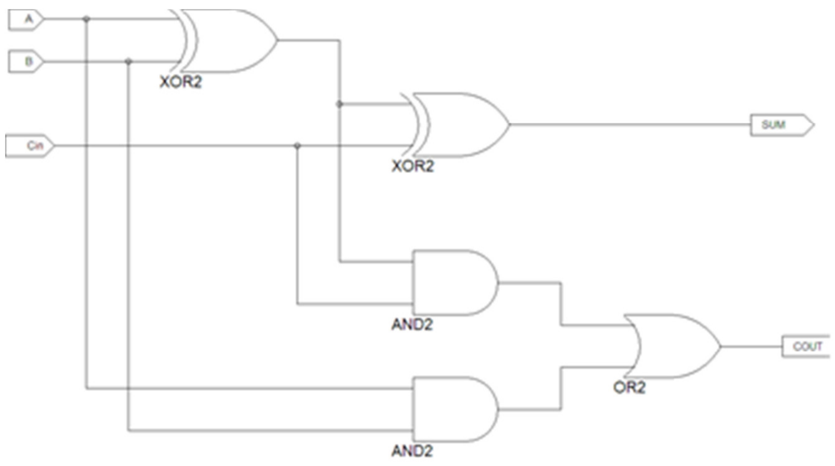
New Project Wizard - Project Summary
Project Navigator will create a new project with the following specifications:
< Back
Finish
Cancel

```
Project:
    Project Name: Full_Adder
    Project Path: C:\Student_Name\Full_Adder
    Top Level Source Type: Schematic

Device:
    Device Family: Spartan3E
    Device:        xc3s500e
    Package:       fg320
    Speed:         -4

    Synthesis Tool: XST (VHDL/Verilog)
    Simulator: ISE Simulator (VHDL/Verilog)
    Preferred Language: VHDL

    Enhanced Design Summary: enabled
    Message Filtering: disabled
    Display Incremental Messages: disabled
```

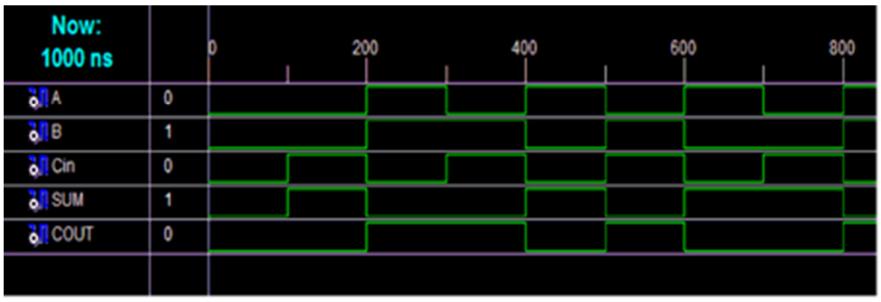
Then click Finish to complete process and verify the file name and type. Click Next once again to proceed and finish.

Section A –Creating the Schematic of a Full Adder

1. Create the circuit of Full Adder. Please refer to simulation lab 1 for more details. Attach input and output pins to the circuit and label them as shown below. Also, make sure you run a timing diagram to make sure the circuit is working properly.



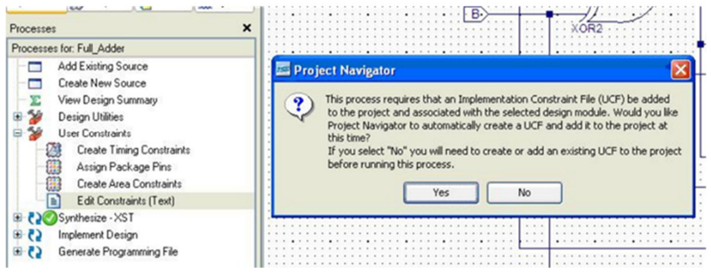
2. Verify your design via waveform simulation



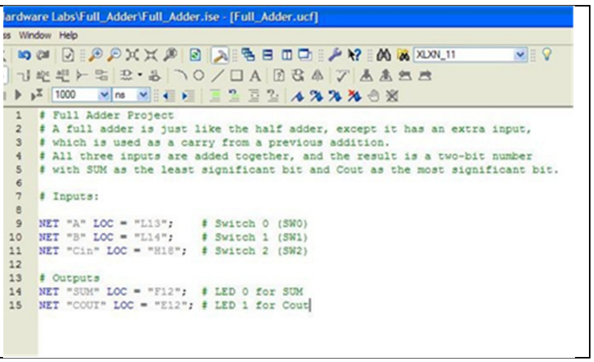
Section B –Develop a User Constraint File “UCF” file for Full adder I/O

1. We need to add the appropriate Spartan 3E FPGA pins to the I/O makers of our design. Let us choose the switches instead of pushbutton for now. We will assign the pins as follows:

```
NET "A" LOC = "L13";       # Switch 0 (SW0)
NET "B" LOC = "L14";       # Switch 1 (SW1)
NET "Cin" LOC = "H18";      # Switch 2 (SW2)

# Outputs
NET "SUM" LOC = "F12";    # LED 0 for SUM
NET "COUT" LOC = "E12"; # LED 1 for Cout
```

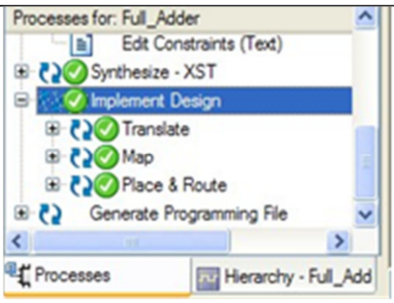
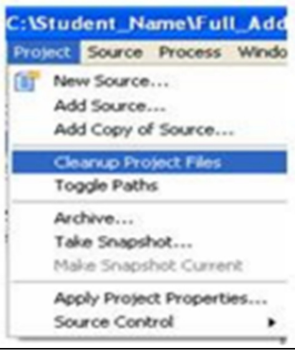
1. In the main window, under process pane right above Synthesize-XST, expand “User Constraints” and select “Edit Constraints (text)” option. This will bring up the following window, press “Yes” to the question about adding a USF file. Full_Adder.ucf file will be added to our project.	 Processes Processes for: Full_Adder Add Existing Source Create New Source View Design Summary Design Utilities User Constraints Create Timing Constraints Assign Package Pins Create Area Constraints Edit Constraints (Text) Synthesize - XST Implement Design Generate Programming File B XOR2 Project Navigator Yes NoThis process requires that an Implementation Constraint File (UCF) be added to the project and associated with the selected design module. Would you like Project Navigator to automatically create a UCF and add it to the project at this time? If you select "No" you will need to create or add an existing UCF to the project before running this process.

Once you click **yes**, you will have an empty text widow. This window will allow us to type the code that is needed to define the relationship between our I/O makers and Spartan 3E FPGA board

2. Enter the following text as shown below and click on save. The location of the pin is described above.	

```
# Full Adder Project
# A full adder is just like the half adder, except it has an extra input,
# which is used as a carry from a previous addition.
# All three inputs are added together, and the result is a two-bit number
# with SUM as the least significant bit and Cout as the most significant bit.

# Inputs:

NET "A" LOC = "L13";    # Switch 0 (SW0)
NET "B" LOC = "L14";    # Switch 1 (SW1)
NET "Cin" LOC = "H18";  # Switch 2 (SW2)

# Outputs
NET "SUM" LOC = "F12";  # LED 0 for SUM
NET "COUT" LOC = "E12"; # LED 1 for Cout
```

Section C – Translate and Fit and Map the design to the Spartan 3E FPGA board

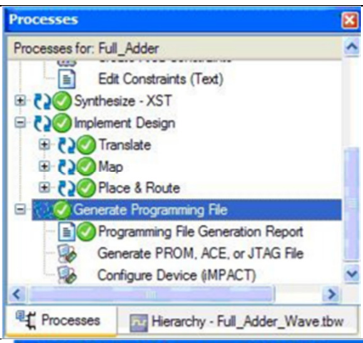
1. Implementation. Double click on “Implement Design” under the process pan on the left hand side, under synthesize. If all goes well, you should get green check marks on “translate, map and Place & Route” as shown. This process implements your Full Adder project with Spartan 3E FPGA board.	
2. If you get yellow warning, you are OK. If you get red marks “Errors”, you will need to go back and trace your process to see if you might have missed something. It is always a good idea to select Project Clean up Project before you go back and synthesize.	

Section D – Program Full Adder into the FPGA Spartan 3 board

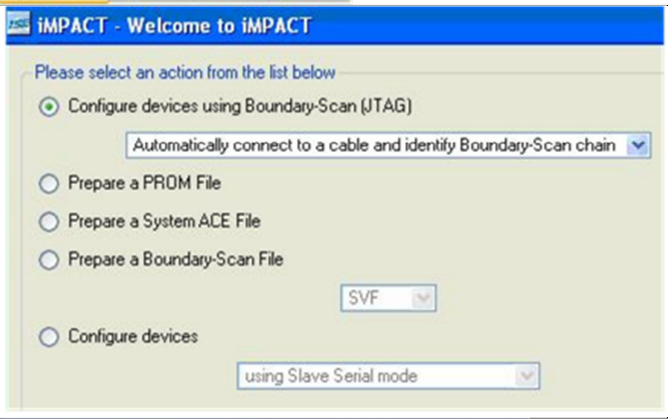
Attach your power and the USB cable to the board. Make sure your USB is connected properly.

1. Double click on “Generate Programming File” under the implement design on the left	

hand side. If all goes well, you should get green check marks on “Programming File Generation Report” as shown.

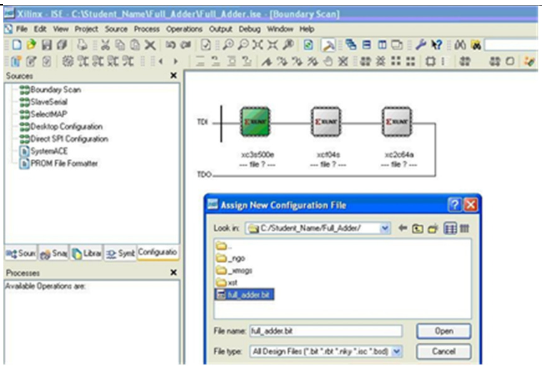

Processes
Processes for: Full_Adder
Edit Constraints (Text)
Synthesize - XST
Implement Design
Translate
Map
Place & Route
Generate Programming File
Programming File Generation Report
Generate PROM, ACE, or JTAG File
Configure Device (iMPACT)
Processes
Hierarchy - Full_Adder_Wave.tbw

2. Now double click “Configure Device (IMPACT)” under Generate Programming File and make sure “Configure devices using Boundary Scan IJTAG” is selected and click Finish.


iMPACT - 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
Configure devices
using Slave Serial mode

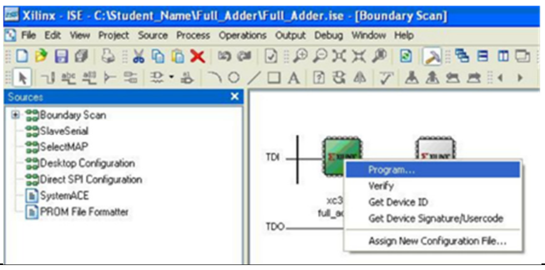
3. You should be prompted with the following screen.

As you can see there are two devices listed in the window. One is xc3s500e FPGA and the xcf04s serial flash ROM. We will need to program only the (FPGA xc3s500e). We will do this by selecting full_adder.bit file from the window and click on open.

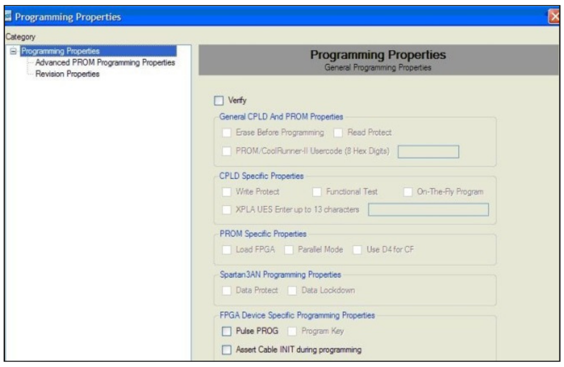


Go ahead and cancel the second window to program the flash ROM. We will not use the Flash ROM for this lab.

4. Right Click on the left most device and select “Program…”:


Xilinx - ISE - C:\Student_Name\Full_Adder\Full_Adder.ise - [Boundary Scan]
File Edit View Project Source Process Operations Output Debug Window Help
Sources
Boundary Scan
SlaveSerial
SelectMAP
Desktop Configuration
Direct SPI Configuration
SystemACE
PROM File Formatter
TDI
TDO
Program...
Verify
Get Device ID
Get Device Signature/Usercode
Assign New Configuration File...

The program gets downloaded to the FPGA board. There are a couple of additional choices that are available but are not necessary at this point. Click OK.



Programming Properties

Category

- Programming Properties
  - Advanced PROM Programming Properties
  - Revision Properties

Programming Properties
General Programming Properties

☐ Verify

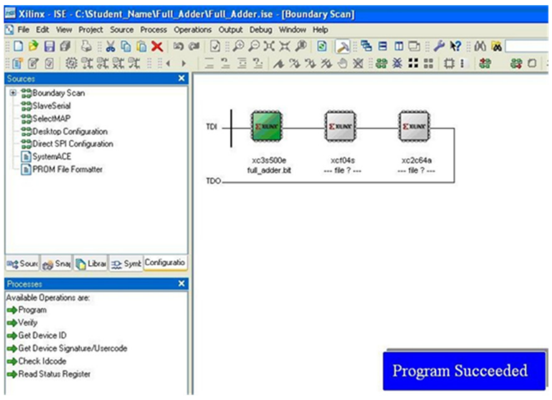
General CPLD And PROM Properties
☐ Erase Before Programming ☐ Read Protect
☐ PROM/CoolRunner-II Usercode (8 Hex Digits)

CPLD Specific Properties
☐ Write Protect ☐ Functional Test ☐ On-The-Fly Program
☐ XPLA UES Enter up to 13 characters

PROM Specific Properties
☐ Load FPGA ☐ Parallel Mode ☐ Use D4 for CF

Spartan3AN Programming Properties
☐ Data Protect ☐ Data Lockdown

FPGA Device Specific Programming Properties
☐ Pulse PROG ☐ Program Key
☐ Assert Cable INIT during programming



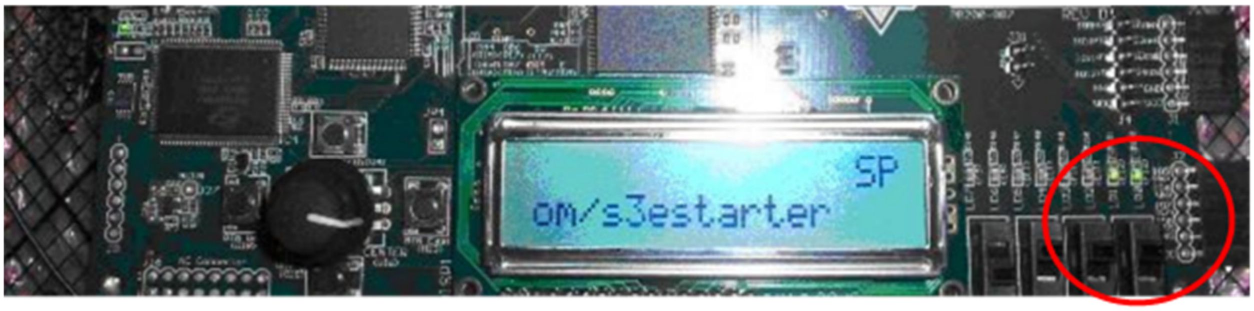
Xilinx - ISE - C:\Student_Name\Full_Adder\Full_Adder.ise - [Boundary Scan]

Program Succeeded

At this point the FPGA should be programmed and ready for test. Test it by sliding the programmed switches up and down for ON and OFF. See if the appropriate LEDs for the two outputs turn on. Here is a truth table that should guide you with your test.

SW(0) (A) L13	**SW(1)(B) L14**	**SW(2)(Cin) H18**	**LD0 (SUM) F12**	**LD1 (Cout) E12**
0	0	0	OFF	OFF
0	0	1	ON	OFF
0	1	0	ON	OFF
0	1	1	OFF	ON
1	0	0	ON	OFF

1	0	1	OFF	ON
1	1	0	OFF	ON
1	1	1	ON	ON



When all three Switches are ON, both outputs are High (SUM and Cout).

As you can see with this approach, you do not need to replace wires and ICs if you make mistake as you might have done in Digital Electronics. Instead, you will have to reconfigure the FPGA with the updated design. This type of technology allows you to concentrate on digital principles and not on the electrical wiring. Larger and more complex projects can be undertaken now that the tedious manual procedures are automated.

Part 2 - Implement Full Adder with (Verilog) for Spartan 3E FPGA board:

Close the previous project, and create a new one

1. Create a new project by selecting:
File from the main menu
New Project

a. In the New Project window, name your project Full_Adder_Verilog in the project name text box.
b. In the Project Location selection box, enter the folder or directory where your project will be saved. Use your name for Student_Name and locate the place where you want to save all your files (i.e C:\)
c. Under Top-Level Source Type, select HDL instead of schematic and click Next.

2. We will design our full adder for a particular device "Spartan 3E FPGA".

Product Category: All
Device Family: Choose Spartan 3E,
the device we will be using. Device: XC3500E, the specific Spartan 3E device we use. This is actually printed (very small) on the FPGA core.
Package: FG320, this is the package type of our device (Ball Grid Array,320 pins)
Speed Grade: The speed grade for this device is -4.
Preferred Language: Verilog

3. Click Next and add a New Source. Select Verilog Module and type Full_Adder_Verilog for File Name. Click Next.

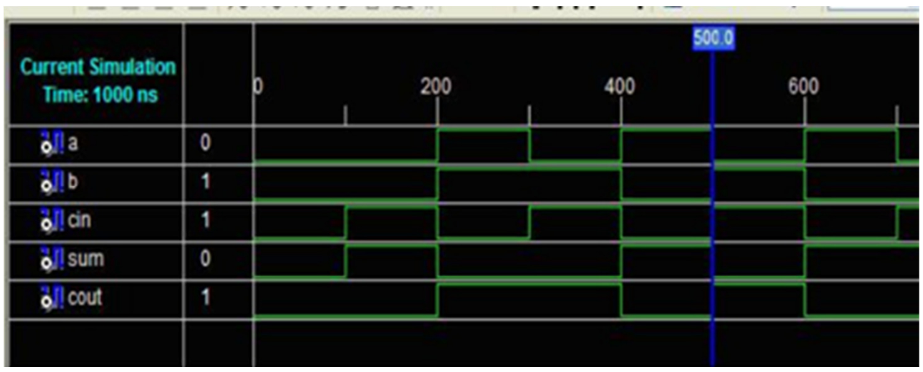
4. Fill in the inputs (in) and Outputs (out) as shown. Click Next and Finish

5. Between begin and end Behavior, we need to enter the following expression for Full Adder. Based on our previous schematic design, the logic expression for the output Sum and Cout are:

SUM = $A \oplus B \oplus Cin$

Cout = $(A \oplus B)Cin + A.B$

6. Follow the same steps as before in part 1 to synthesize and obtain a timing diagram to verify correct outputs. You should obtain a similar timing diagram.



7. Implement the design on Spartan 3E and demo your work.

NAME:-
ID:-

Evaluation Lab 1 (5 %)

Print this and present it to me when you demonstrate your work.

Requirement	**Points**
Complete the lab on time (2 lab sessions)	/4
Build, debug and demonstrate the operation of Full adder schematic in Xilinx and Spartan 3E FPGA board. Demo your design	/6
Build, debug and demonstrate the operation of Full adder Verilog in Xilinx and Spartan 3E FPGA board. Demo your design	/6
Show Verilog syntax code for 2-bit Multiplier	/4
Total	/20

1. 2-bit Multiplier. A) schematic diagram B) verilog syntax code

2. Sketch / snapshot the simulation results of the full adder

3. Record the FPGA resources consumed by your 2-bit multiplier design

FPGA Resources	Utilization	Total Available	Percentage Utilization
Slice LUTs (LookUp Tables)			
Bounded IOBs			

4. What is the maximum estimated frequency at which this design in your 2-bit multiplier can run?

    a. What is the corresponding minimum clock period?

    b. Maximum Frequency: ________ MHz Minimum Clock Period: ______ ns

5. Record the path reported under “Pad to Pad Report” – Place and Route Report.

Source Pad	**Destination Pad**	**Delay**

* Please hand in the design summary report of your 2-bit multiplier design along with this Lab 1 – Evaluation form.