Instructions on FPGA Board and Xilinx software

Prepared by
Diego Campos (SPRING 2002): 1.0,1.5,2.0
Updated by Matthew Klepeis (SPRING 2003): 3.0
Snehesh Shrestha (SPRING 2004): 4.0
Steve Sanko (SPRING 2005): 4.1
Brian Moran (SPRING 2006): 4.2
Chris Janssen and James Kelly (SPRING 2013) 5.0
Nathan Copros (SPRING 2018) 6.0
Version 6.0
Last update –3/13/13
Edited by Dr. Joseph Wunderlich

I. XILINX AND FPGA INTRODUCTION

Field-Programmable Gate Array (FPGA) - A logic chip which can be programmed to function as an array of computational logic blocks.

Xilinx Software - Xilinx Software allows a designer to graphically create a logic circuit which can be tested and simulated prior to implementation on the FPGA board.

FPGA vs Breadboarding [1]

- FPGA's simplify design, implementation, testing, and debugging
- Reusable - Easily reprogrammed and reconfigured
- Most are non volatile (the memory is saved after the device is turned off)
- Useful in prototyping IC designs
- Low power
- High number of gates/parts in the same area
- HDL helps create libraries (new gates or parts) and functional blocks
- Do not need to worry about voltage, current, or power to each gate or circuit part
- Circuit can be simulated and debugged before implementation; therefore saving time and money

FPGAs can be programmed using schematics to lay out a design, or by using an HDL like Verilog or VHDL. As well as being cheap to program for an application, FPGA hardware is relatively inexpensive. FPGAs can be used for complex applications, such as image processing or image compression, and are suitable for use in portable phones, digital cameras, and other complex digital applications. Xilinx, as well as other companies sell intellectual property (IP) cores, and opencores.org provides cores and source code for free under the GPL.

Evaluating the performance of an FPGA is difficult. Most manufacturers don't provide a gate count for their products, because it tends to be misleading. Mostly, FPGA manufacturers give a number of "logic units" or "logic cells," which consist of a lookup table and a flip-flop, as well as assorted connection logic.

Because different manufacturers use different types of basic gates (AND vs. NAND, etc), the gate count of a logic unit isn't the same across different manufacturers, or even different FPGAs.

FPGAs are becoming more sophisticated, and their applications are expanding as they become more capable. When doing calculations that can be made in parallel, the FPGAs are programmed to deal specifically with this problem. Because of the high speed I/O capabilities of the FPGA, it is ideal for acceleration of certain problems.

II. INSTALLING SOFTWARE

Installation Notes:

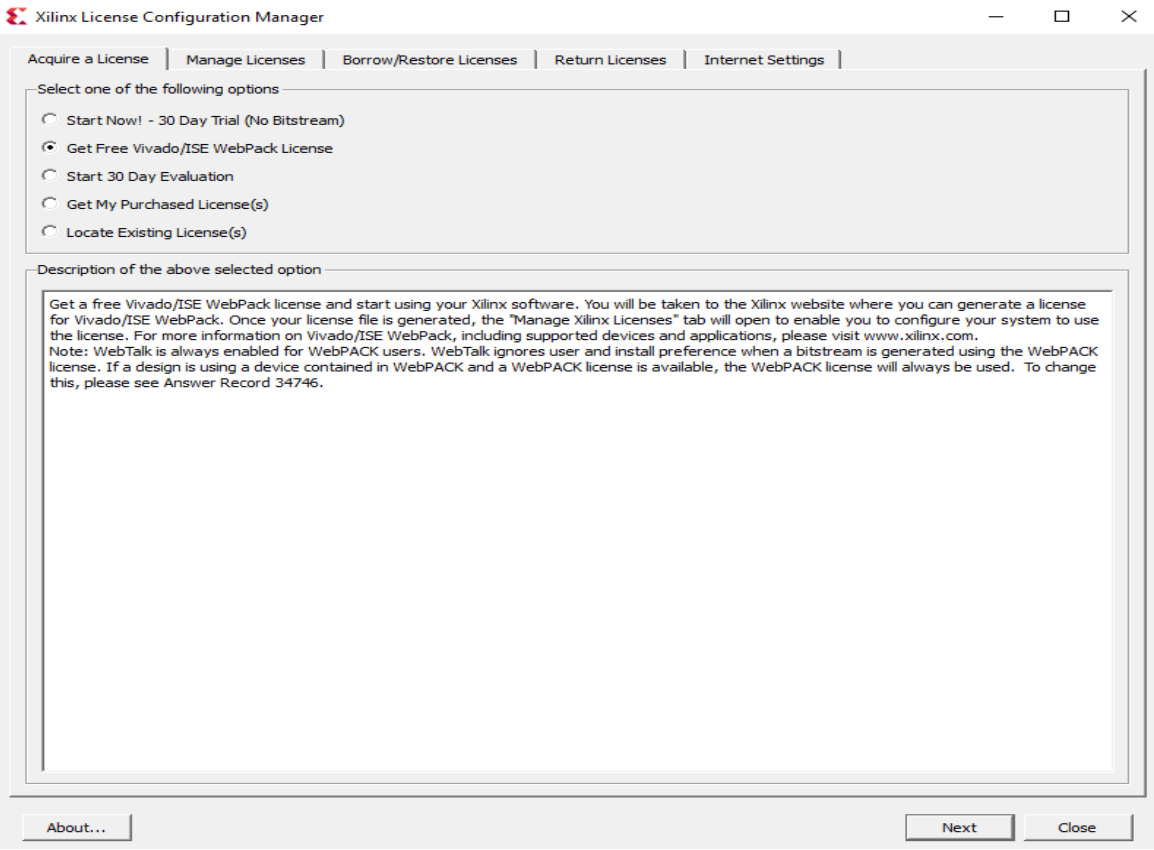
1. With Windows 10, you will need to run the 32-bit version of the ISE Webpack software:

2. You will need to create an account to download the software. Then, fill the verification form, where company name is Elizabethtown College.

3. Only one license can be activated per account, so use a different account for each computer.

4. To activate the license, launch the "Manage Xilinx Licenses" app under the folder Xilinx Design Tools. This can be found by clicking "All Apps" after pressing the Windows button.

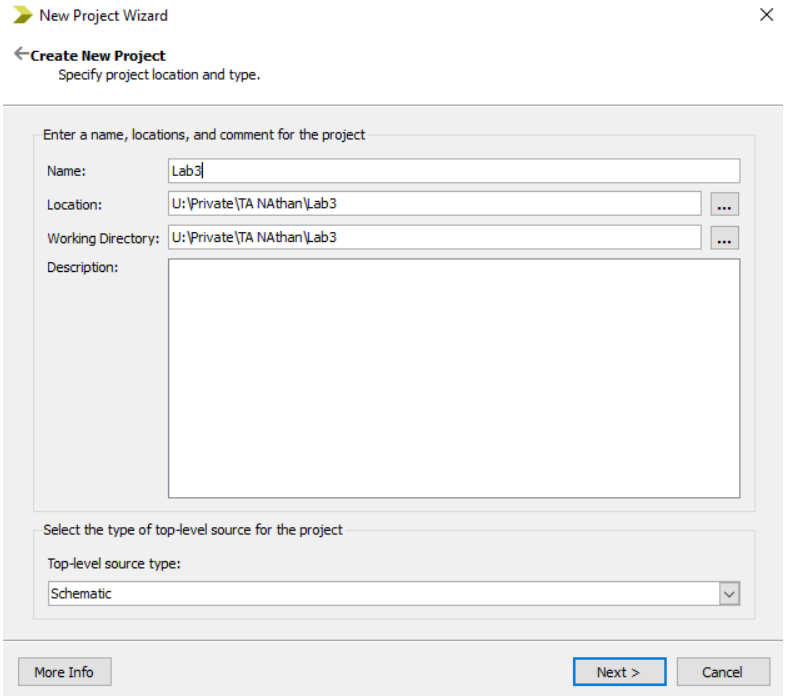
5. Select the "Get Free Vivado/ISE Webpack License" option to aquire a license. Then fill in the information to download a license folder. Then, go to the "Manage Licenses" tab to load the license file and you can now launch the fully functional free software.



III. USING XILINX ISE PROJECT NAVIGATOR

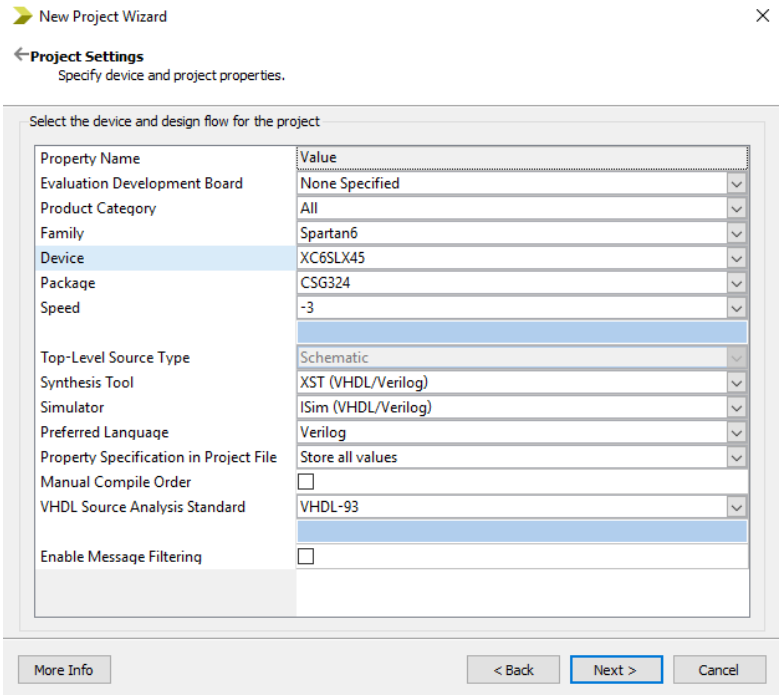
a) Start program called “32-bit Project Navigator”. It can be found by pressing the start menu, “All Apps”, “Xilinx Design Tools”

b) Create a new project



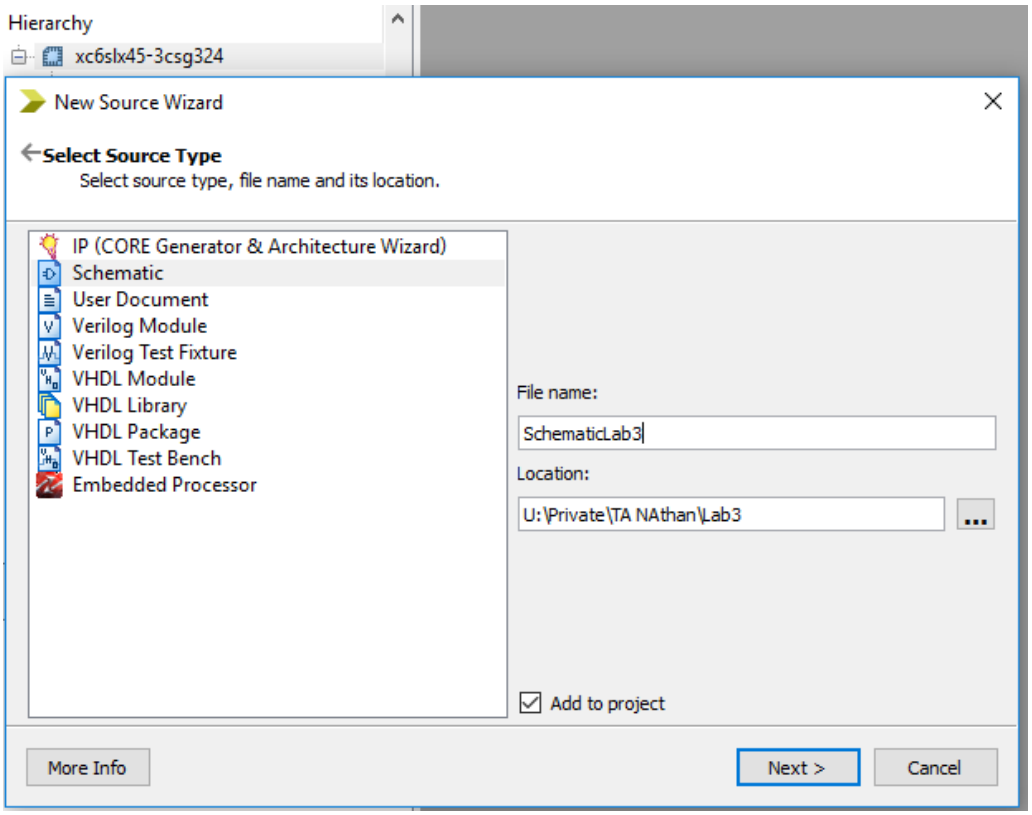
c) Choose a name, and make sure to save it in a location that is not read-only (your public or private folder will work). Change the Top-level source type to “Schematic”. Press “Next”

d) Select “Spartan” in the “Family” box, “XC6SLX45” in the device box and “CSG324” in the “Package” box

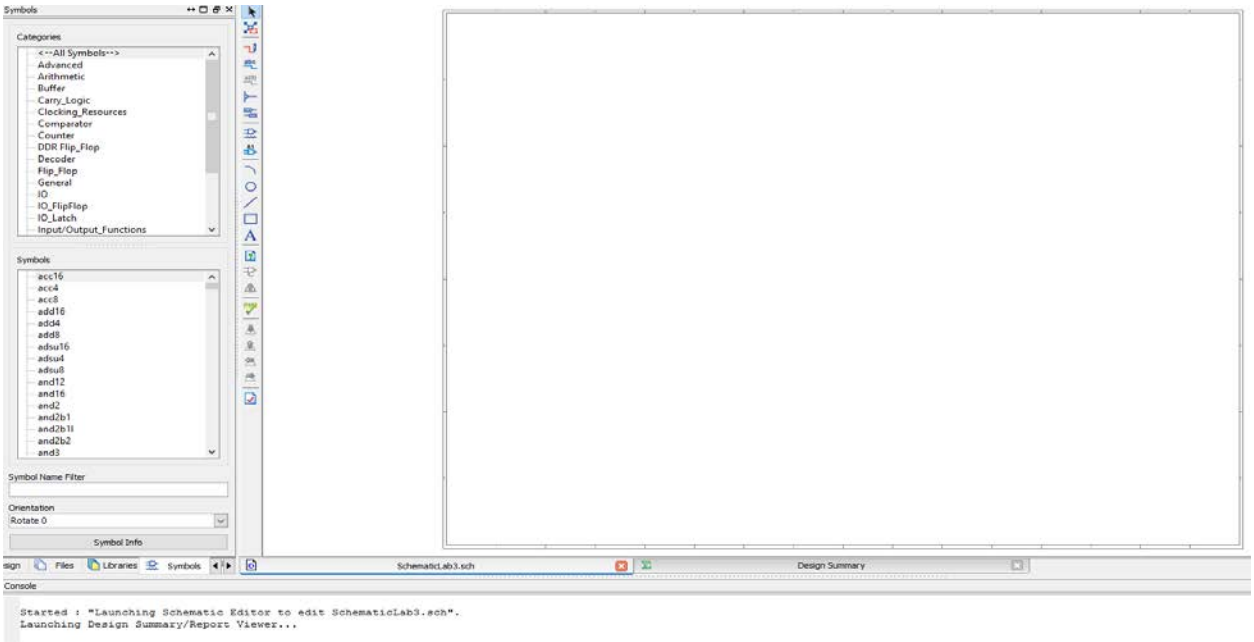


e) When the “Project Summary” box appears, select “Finish”

f) Under the Hierarchy window on the top left, right-click on “xc6slx45-3csg324”, and choose “New Source”. Press “Finish”



g) The schematic window will appear and you will be able to navigate between the hierarchy window under the “Design” Tab and the logic circuit component under the “Symbols” Tab

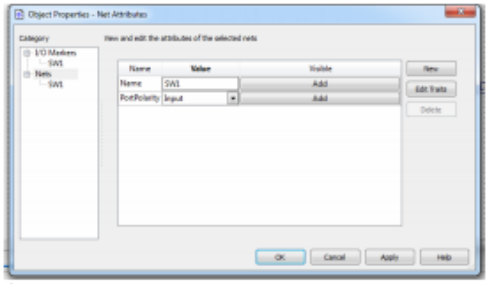


h) Like Logisim, the Symbol tool lets you add parts by connecting the respective leads together with the mouse or with the wire drawing tool. Common gates like the AND gate are located under the “Logic” category. To create an Input, you must first add an “IBUF” from the IO

category. Then, click “I/O Marker” and click the first leg. For an output, add an Obuf and add the I/O marker to the second leg.

iii. ***Input(s) & Output(s):*** Must be assigned **PIN LOCALIZATION NUMBERS**. (see step l)

i. Inputs and outputs should be named in such a way to help illustrate their purpose
   i. To input a name, double click on the symbol, click “Nets” in the category on the left of the window that appears and type the desired name in the value field under the “Name” row



j. Assigning pin locations
   i. Create a constraint file by clicking on the “design” at the bottom of the design window (if the tap is not visible click the arrows in the bottom right corner of the window to scroll through the different tabs)
   ii. Select your schematic in the hierarch window
   iii. Click the “New Source” button
   iv. Select the “implementation constraint file” under “source type” and input a name
   v. To assign a pin to an input/output, type “**NET** **”<**`block name`**>”** **LOC=”**`<pin location>`**”;**” into the text file that was created when

you created the constraint file

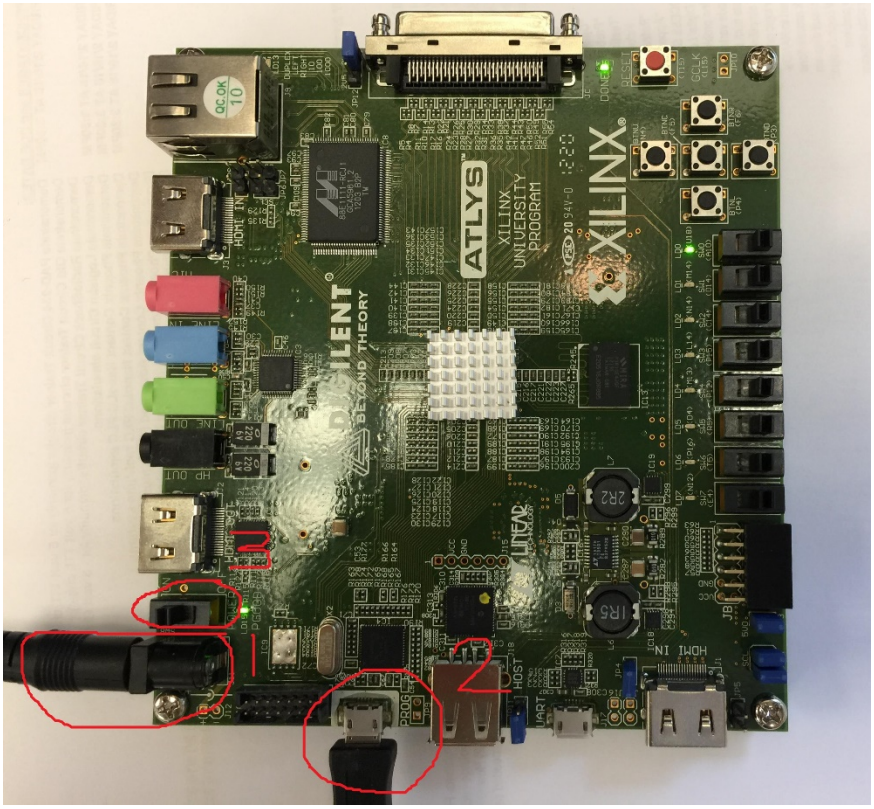
Pushbuttons	Slide Switches	LEDs
BTNU: N4	SW0: A10	LD0: U18
BTNC: F5	SW1: D14	LD1: M14
BTNR: F6	SW2: C14	LD2: N14
BTNL: P4	SW3: P15	LD3: L14
BTND: P3	SW4: P12	LD4: M13
BRST: T15	SW5: R5	LD5: D4
	SW6: T5	LD6: P16
	SW7: E4	LD7: N12

Important Pin Locations (for full list see Appendix A)

IV. IMPLEMENTATION/PROGRAMMING

Make sure to use the power cord from the Xilinx box. It should be a 5V output. Any voltage superior to this could permanently break the FPGA board.

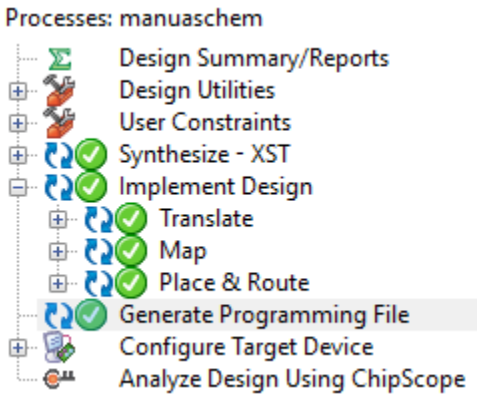
a) Once the circuit is properly designed, you can proceed to implement your circuit on the FPGA.
b) Setup the FPGA board in three steps as illustrated in the picture below:



Plug in the power cord, then the usb cable from the board to the computer, and finally turn on the power switch.

c) Click on the “Implement” button (green arrow on the middle left of the design window)

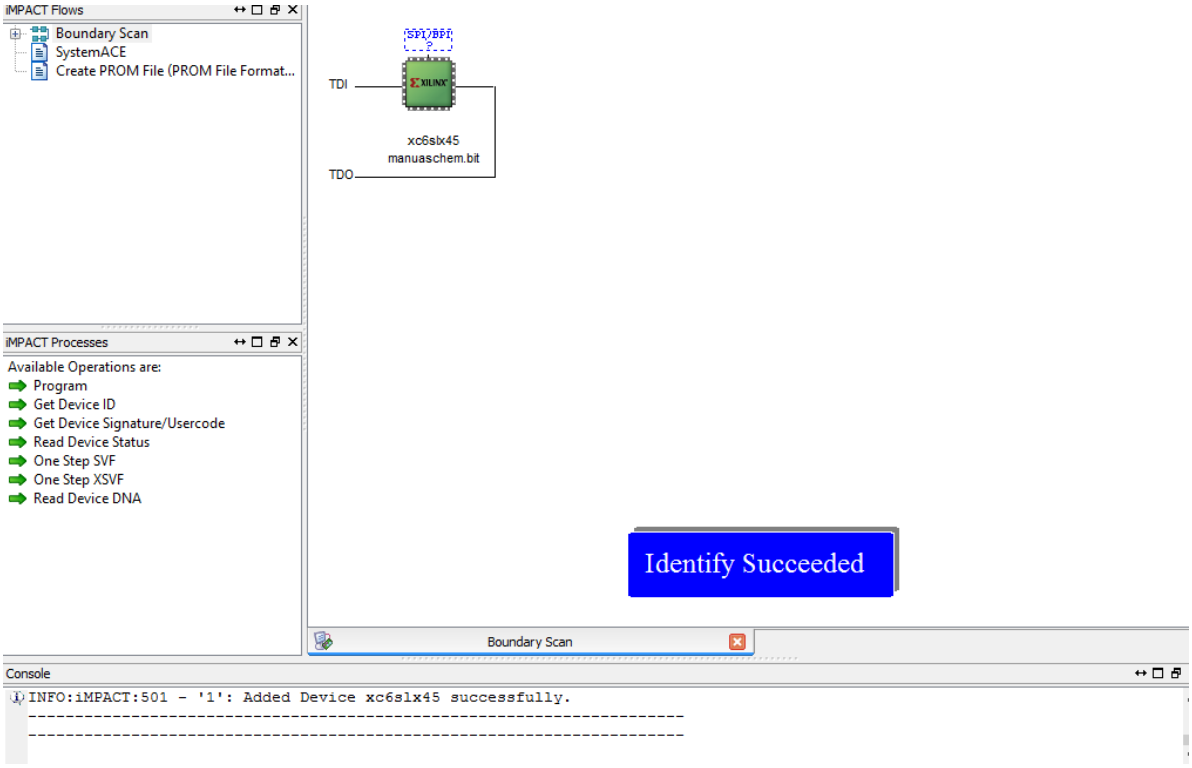
d) In the hierarchy, click on the schematic file. On the bottom left, you should see the list of processes. Click on “Synthesize –XST” and “Generate Programming File”. After that, you should see only green status like in the picture below:



e) The next step is to download the circuit into the FPGA. To accomplish this, press the Tools tab then click on “Impact”, and press “OK” to the warning message.

f) Double-click “Boudary Scan”, then right-click and select “Initialize Chain”. A message should pop-up to assign configuration files, press “Yes”. Then open the saved schematic. Another message should pop-up about Flash PROMs, you should press “No”. Then the Device Programming Properties window will come up and click on Device 1 under Boundary-Scan and press “OK”.

g) Finally, if everything went well, you should see the equivalent of the picture below. To launch the program, press “Program” under Available Operations. It should display “Program Succeeded”


iMPACT Flows
Boundary Scan
SystemACE
Create PROM File (PROM File Format...
iMPACT Processes
Available Operations are:
Program
Get Device ID
Get Device Signature/Usercode
Read Device Status
One Step SVF
One Step XSVF
Read Device DNA
TDI
TDO
xc6slx45
manuaschem.bit
Identify Succeeded
Boundary Scan
Console
INFO:iMPACT:501 - '1': Added Device xc6slx45 successfully.