Introduction to FPGA

ECE414 – Fall 2020

Randil Gajasinghe and Prof. Onur Tigli

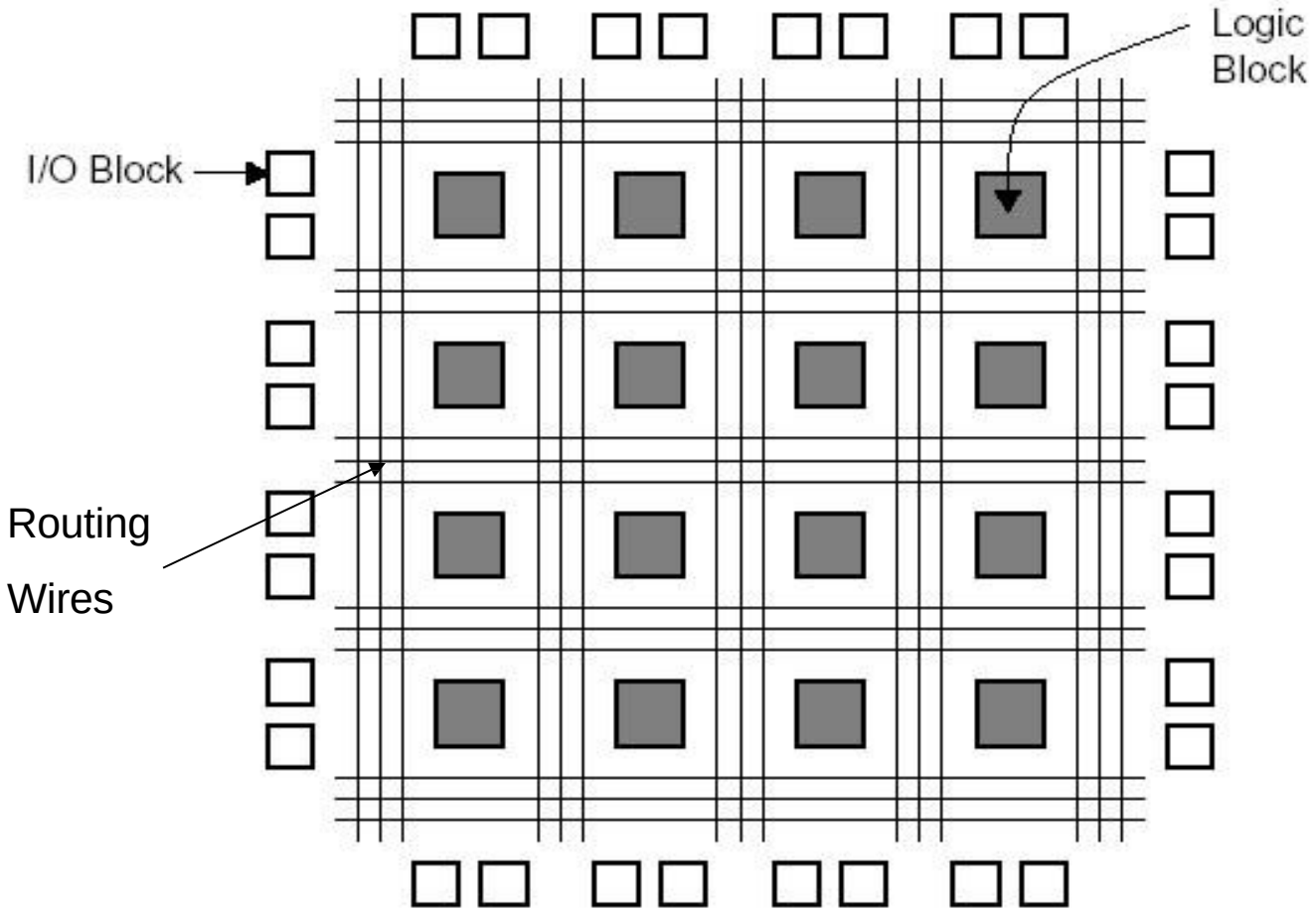
Last edited by Diego Palacios
e-mail : d.palacios31@umiami.edu

FPGA

- **F**ield **P**rogrammable **G**ate **A**rray
  - Reconfigurable hardware
- Alternative to ASIC (**A**pplication **S**pecific **IC**)
  - *Disadvantages:*
    - Low speed, High power – *Advantages:*
    - Shorter time-to-market, Lower cost

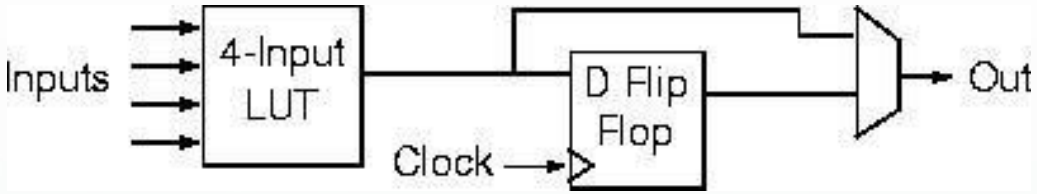
FPGA Architecture

- Xilinx, Altera, Lattice, Actel, Cypress, Atmel

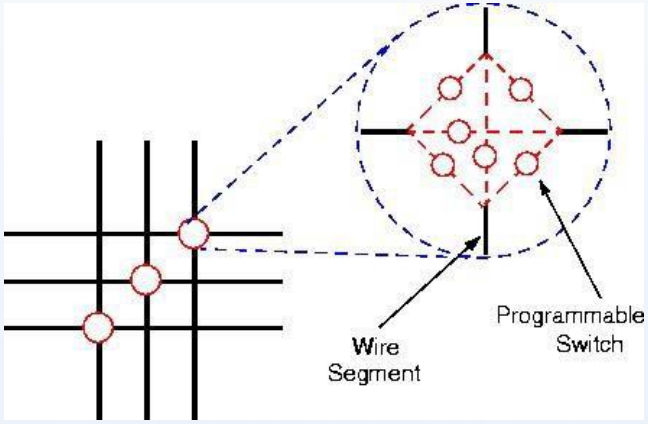

Logic
Block
I/O Block
Routing
Wires

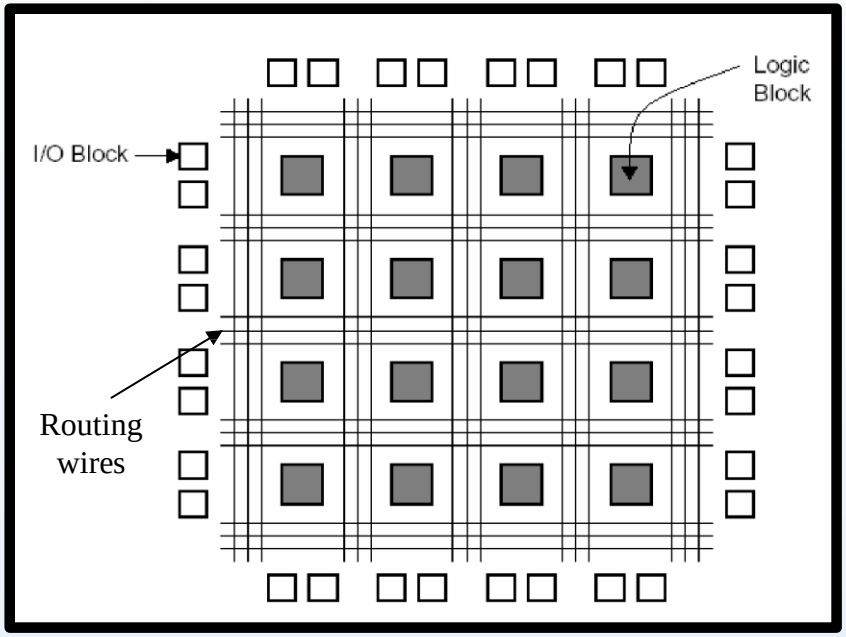
FPGA Architecture

- Logic block

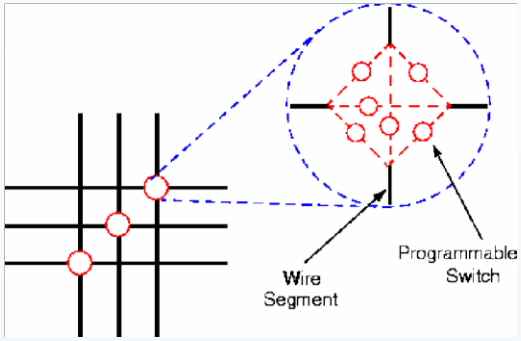


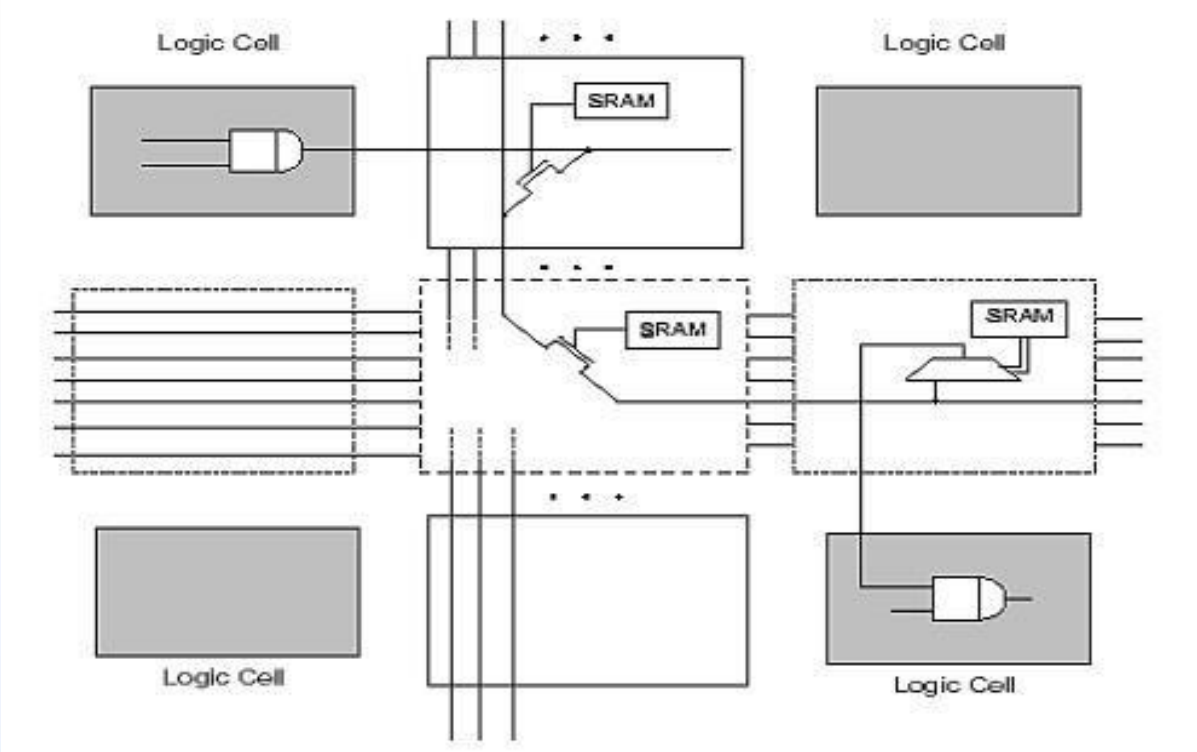
- Switching





- Switching


Wire
Segment
Programmable
Switch


Logic Cell
Logic Cell
SRAM
SRAM
SRAM
Logic Cell
Logic Cell

FPGA Architecture
Nexys 4 FPGA Board

Nexys 4 Board – FPGA Module

Device Family – Artix 7

Device – XC7A100T Package – CSG324C

Nexys 4 FPGA Board

Artix-7 FPGA chip:

- 101,440 logic cells in 15,850 slices
- 240 DSP48E1
- 1,188 Kb of distributed RAM and 4,860 Kb of block RAM
- One 10-bit analog to digital converter block – PLLs, etc.

Nexys 4 FPGA Board

Nexys 4 Board:

- 128 Mbit pseudo-static DRAM
- 128 Mbit serial flash memory
- 10/100 Ethernet
- Digital microphone
- 3-axis accelerometer
- Audio amplifier and speaker jack
- Temperature sensor
- USB port for HID devices (mice and keyboards)

Nexys 4 FPGA Board

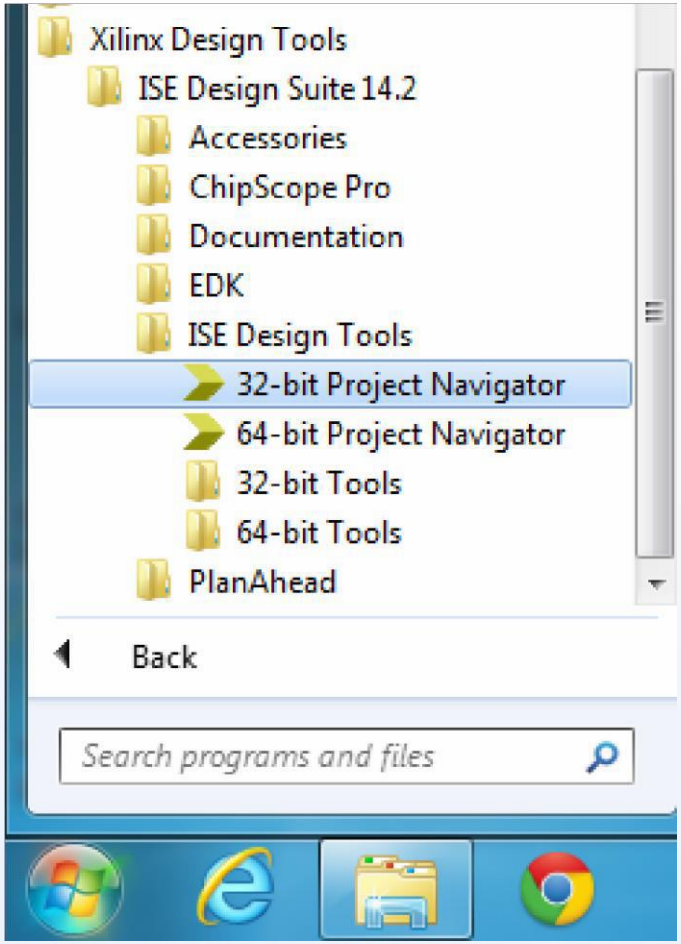
Input/Output

- 16 individual LEDs in various colors
- 2 tri-color LEDs
- 16 slide switches
- Six pushbuttons (one for FPGA reset)
- 8-digit seven-segment display

Xilinx ISE Webpack

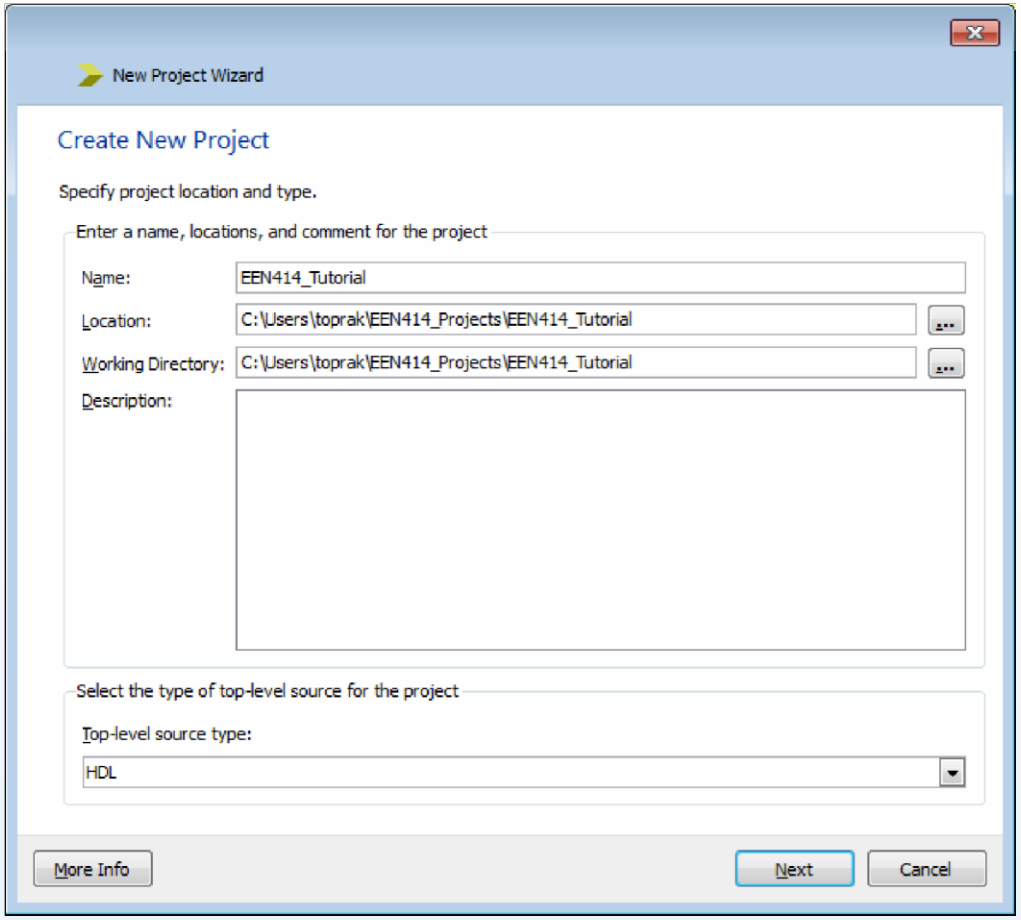
Step by step tutorial

Tutorial



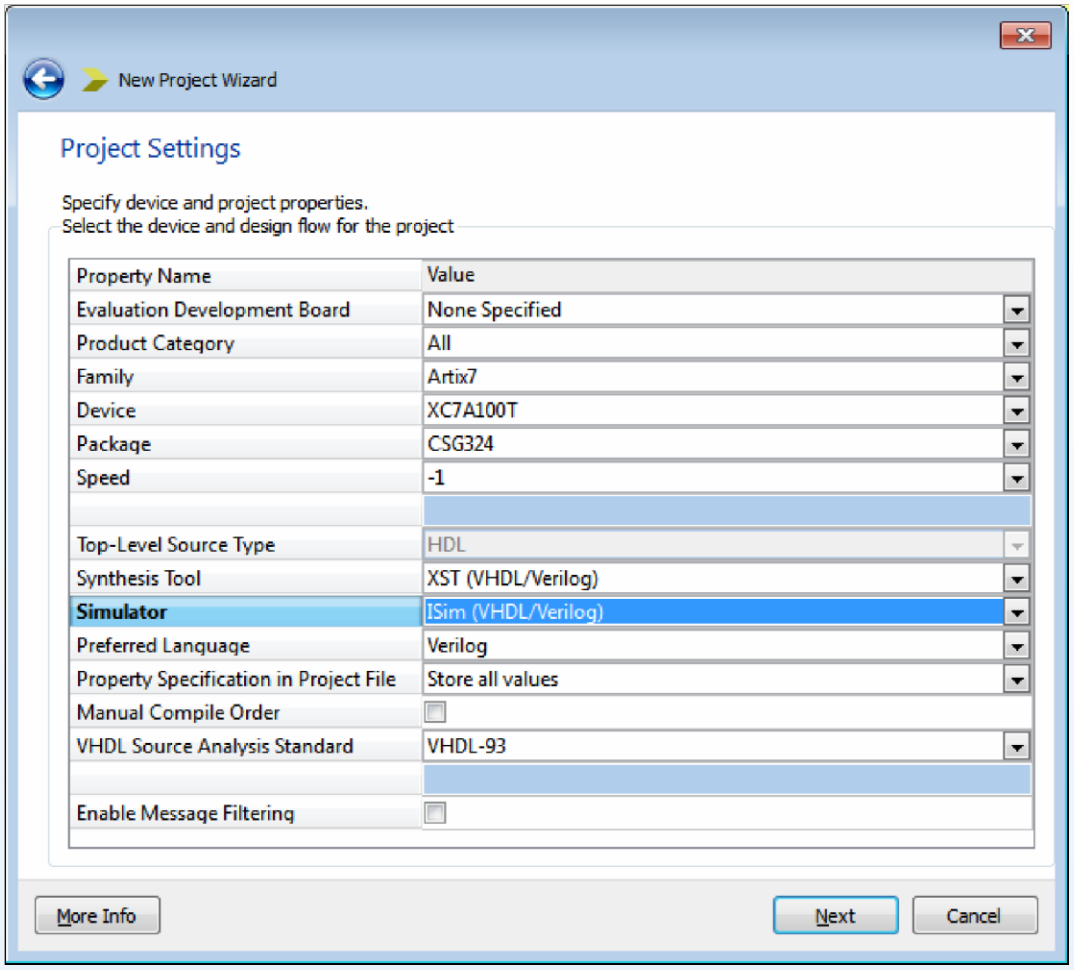
- Run Xilinx ISE project navigator
- File->New Project

Tutorial



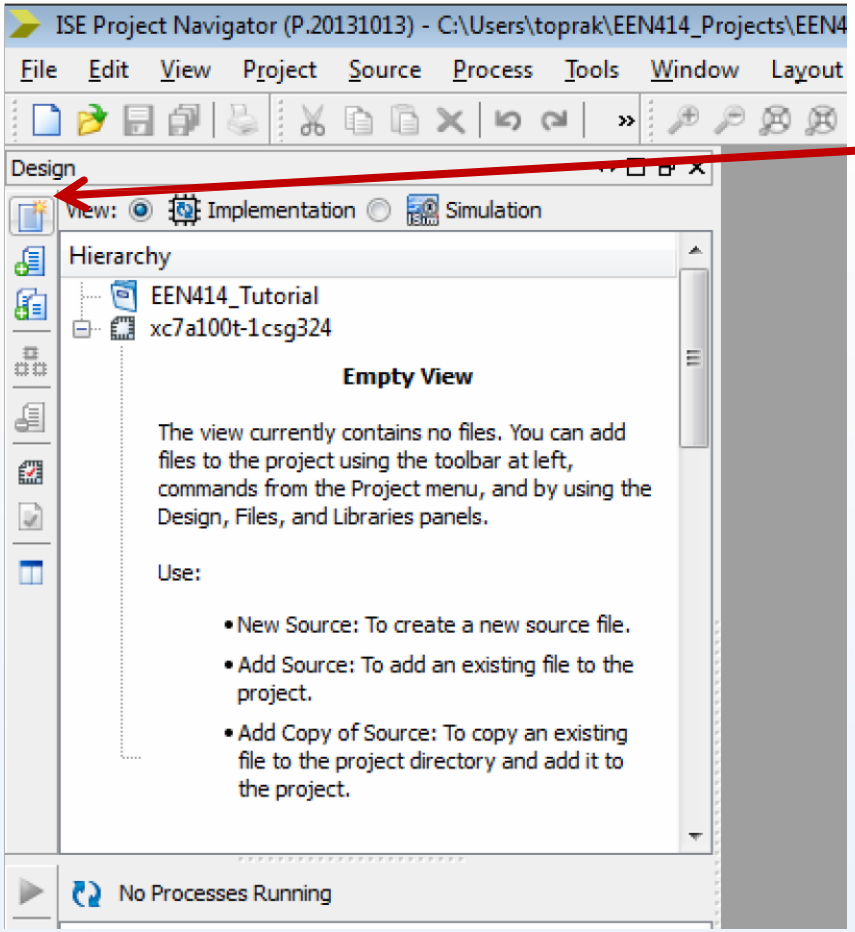
- Project name
- (Optional) Change location

Tutorial



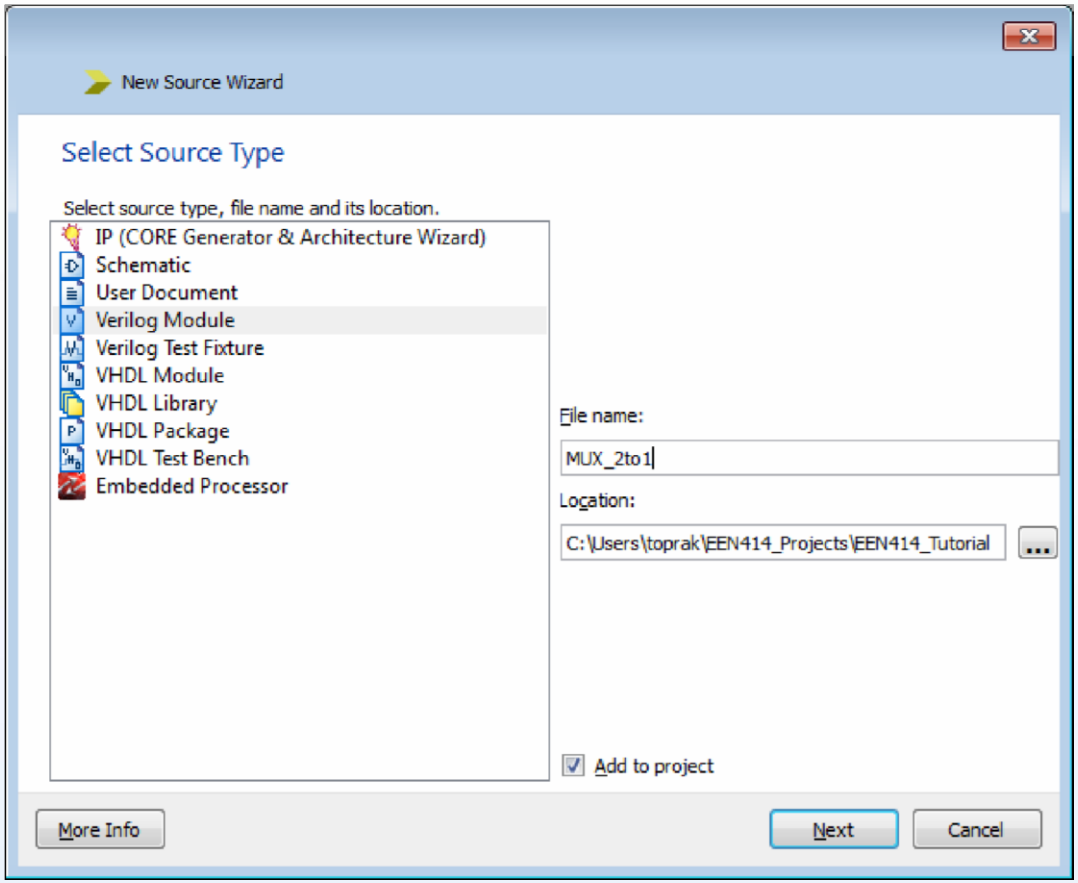
- Enter project settings as shown

Tutorial



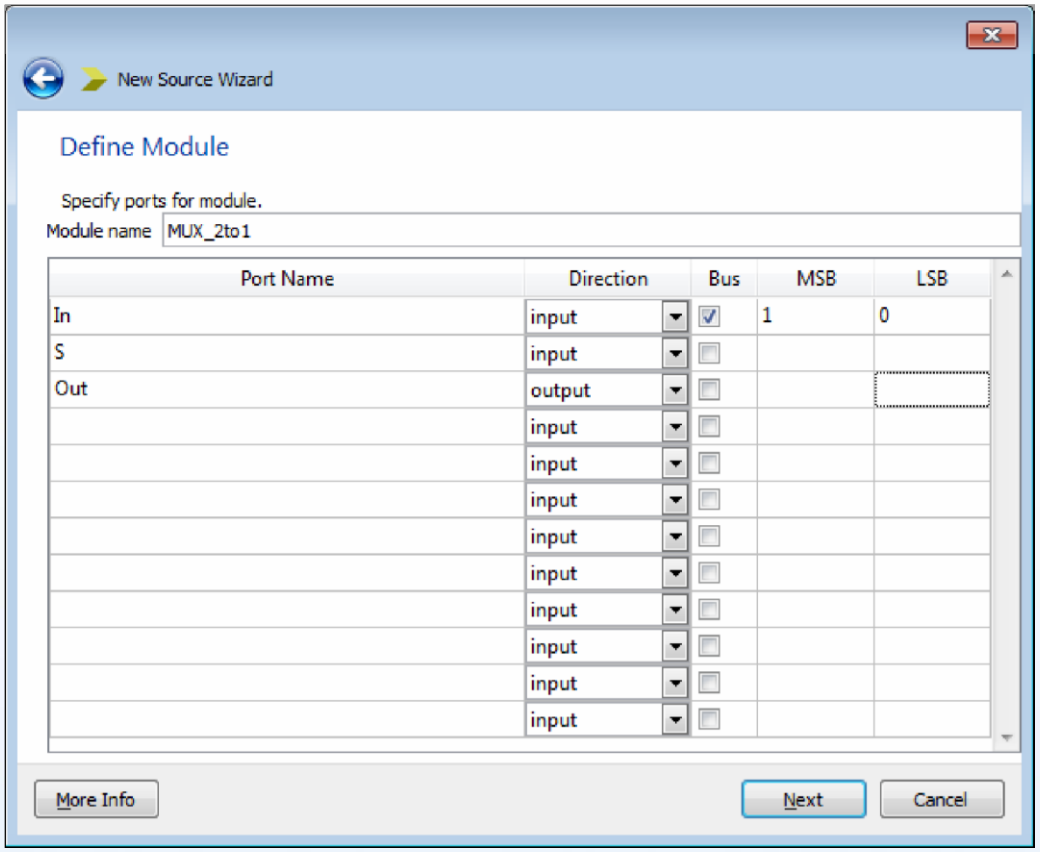
- Click on the "New Source" button

Tutorial



- Select source type as shown
- Type a name for the source file
- Inputs/Outputs of the module
  - Can be entered/modified later

Tutorial



– Note that you create buses

- Define MSB and LSB for buses

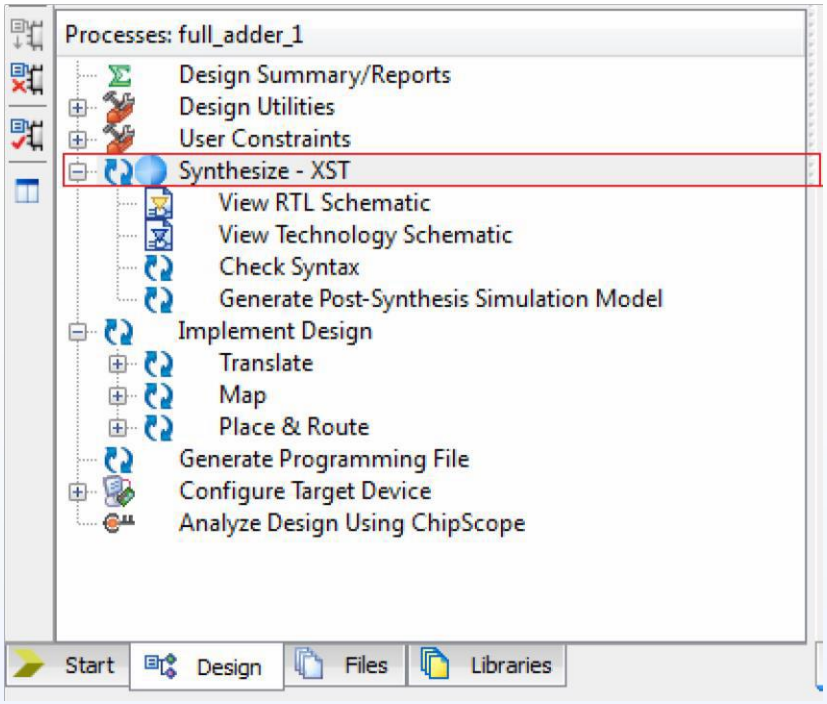
Tutorial

- Type your code

```
//////////////////////////////////////////////////
module MUX_2to1(
    input [1:0] In,
    input S,
    output Out
    );

    wire x, y, z;           // Wires are defined

    and(x, In[1], S);       // AND gate #1
    not(z, S);              // NOT gate
    and(y, In[0], z);       // AND gate #2
    or(Out, x, y);          // OR gate

endmodule
```

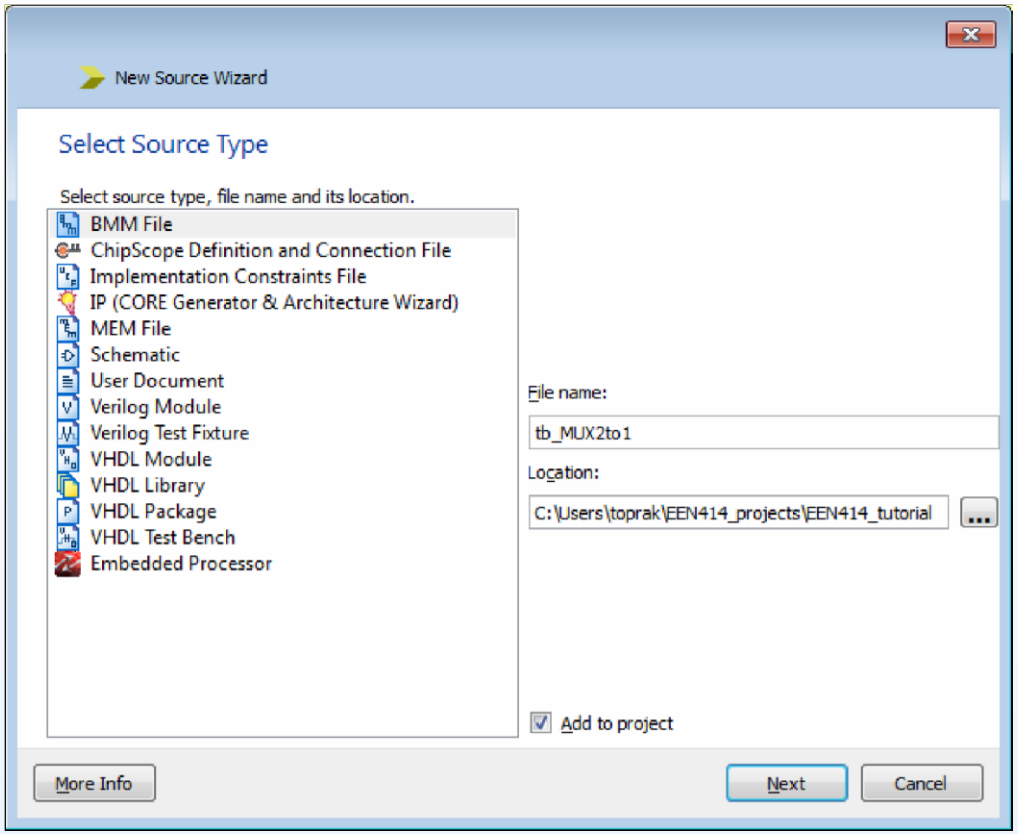
- Comments are useful to explain your code
- Multiple modules and/or source files possible

Tutorial



- Synthesize the code and verify that there are no errors

Tutorial



- Testbench to <u>simulate</u> the module

1. Add a new source file
2. Source type: "*Verilog test fixture*"
3. Enter the name and location of the file

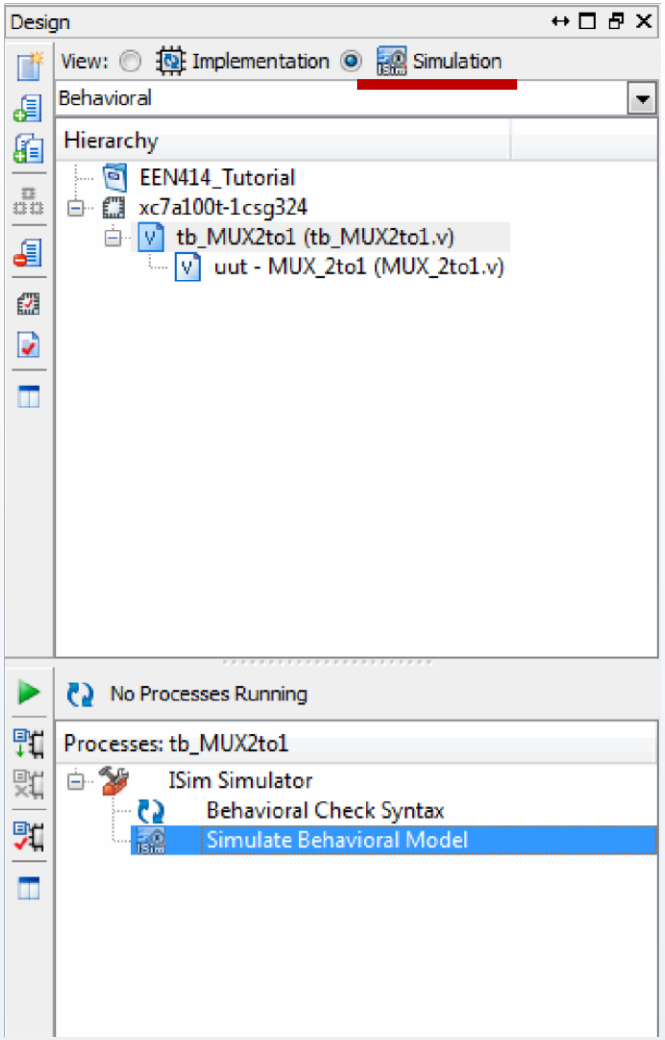
- Initialization parts are already written

Tutorial

```
31    // Outputs
32    wire Out;
33
34    // Instantiate the Unit Under Test (UUT)
35    MUX_2to1 uut (
36       .In(In),
37       .S(S),
38       .Out(Out)
39    );
40
41    initial begin
42       // Initialize Inputs
43       In = 0;
44       S = 0;
45
46       // Wait 100 ns for global reset to finish
47       #100;
48
49       // Add stimulus here
50
51       In = 2'b00;    // The 2-bit input "In" is denoted using binary notation
52       S = 0;
53       #100;
54
55       In = 2'b01;    // Binary notation here as well
56       S = 0;
57       #100;
58
59       In = 2;        // This is decimal notation. Note that d(2) = b(10)
60       S = 0;
61       #100;
62
63       In = 3;        // Again decimal notation; d(3) = b(11)
64       S = 0;
65       #100;
66
67    end
68
69 endmodule
70
```

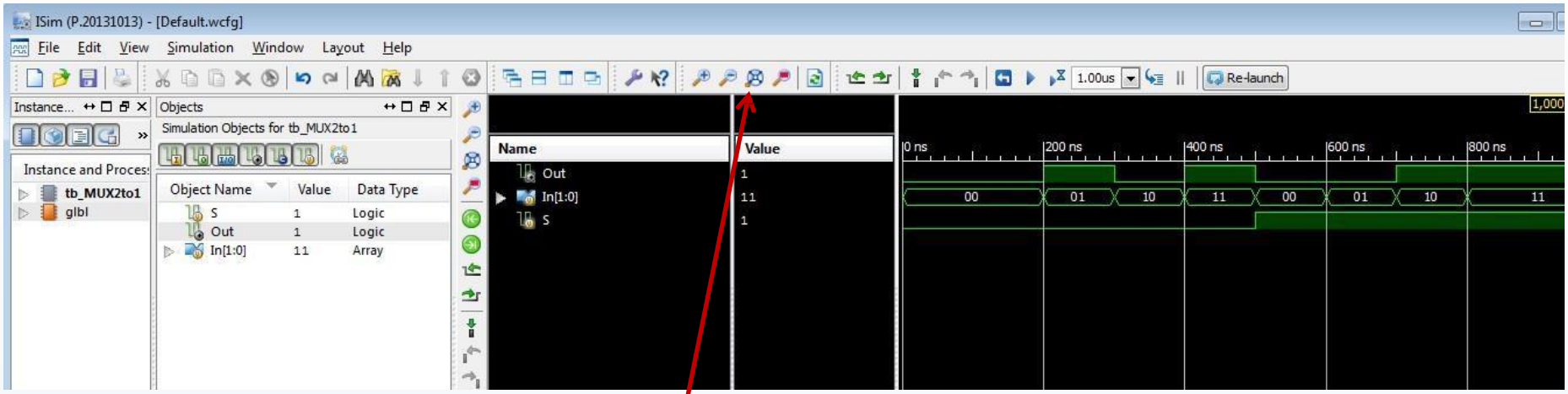
- Add test inputs to the module
  - Comment line shows where to add
  - Add **all** possible input combinations

Tutorial



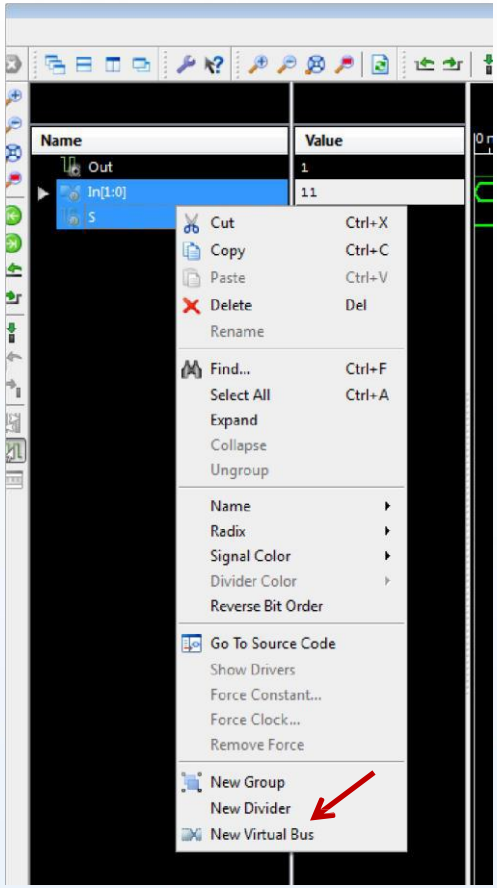
- Select "*Simulation*" radio button
- Double click on "Simulate Behavioral Model" to run the simulation

Tutorial



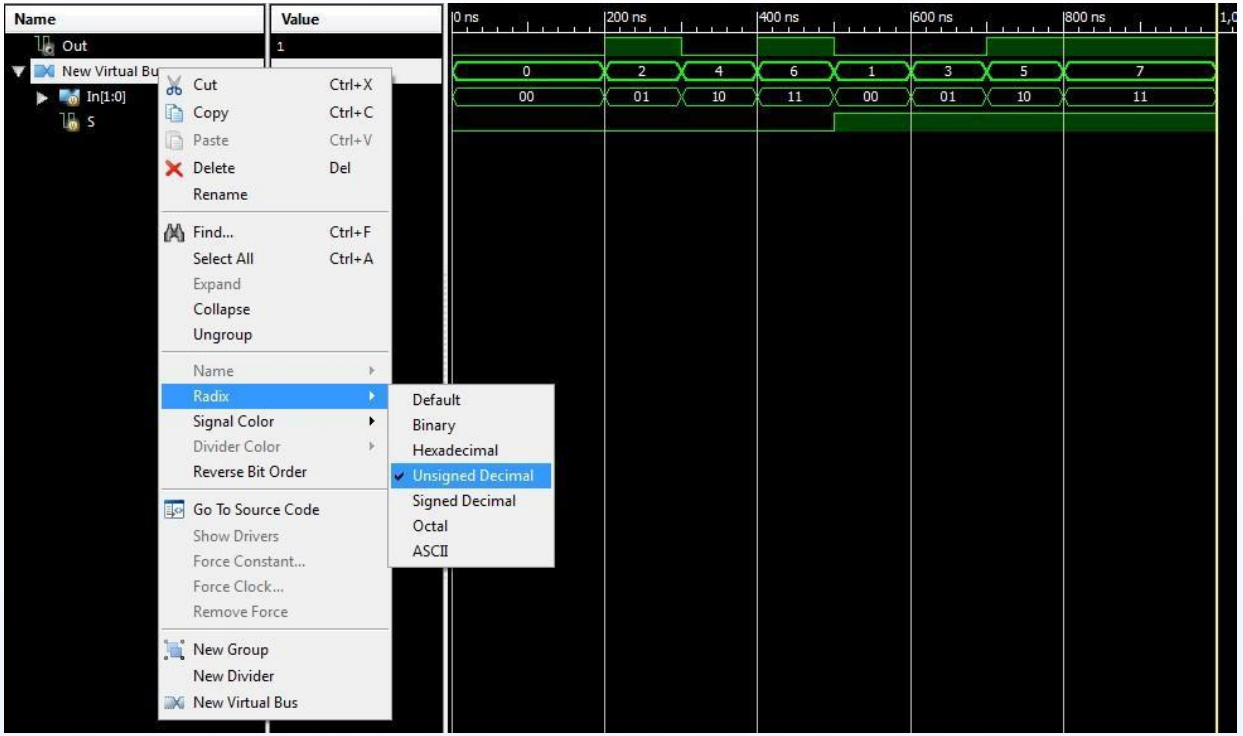
- ISim simulator
- Click on this button to fit waveforms on screen

Tutorial



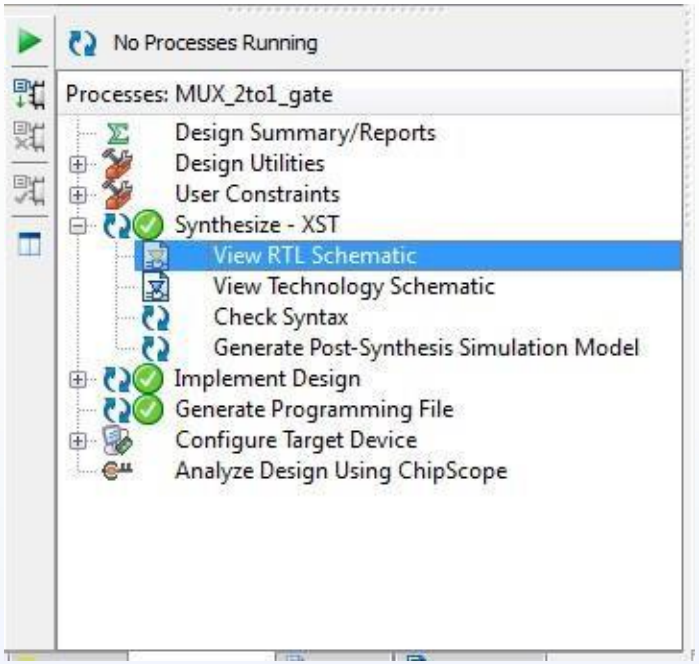
- Multiple single bit signals can be combined into a *'virtual bus'*
  - *Convenient method to see several bits at once – Binary, decimal, hexadecimal, etc.*
  - *Not necessary for this example, just to demonstrate here*

Tutorial



- Verify proper operation of your module

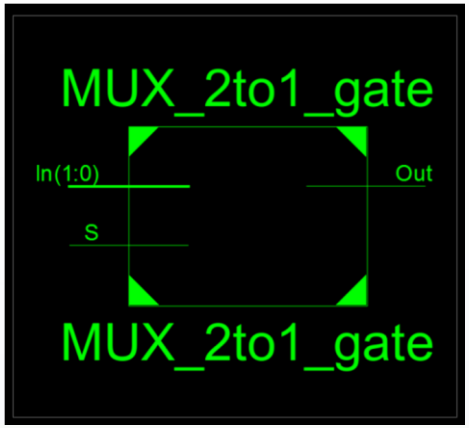
Tutorial

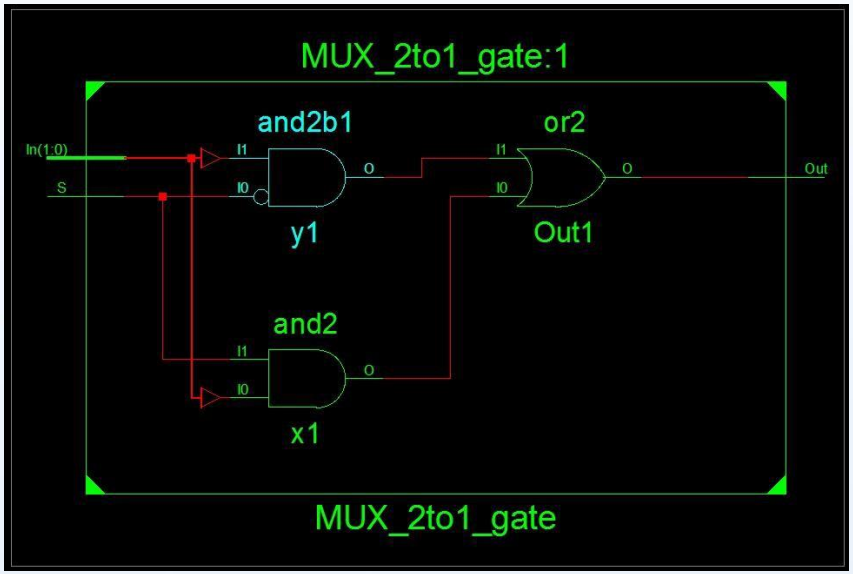


- RTL and technology schematic views
  - Under "Synthesize"

- Schematic views of the circuit defined in your code
  - Top level RTL schematic

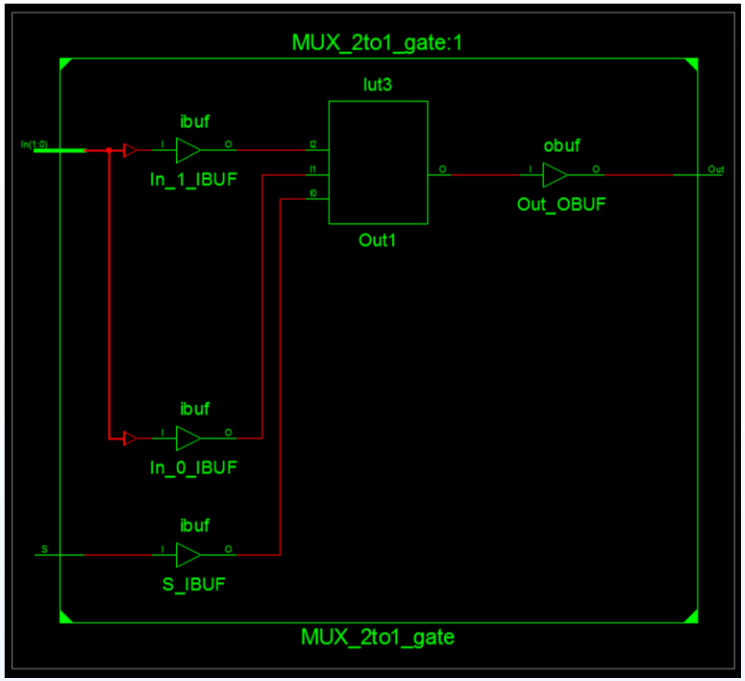
Tutorial

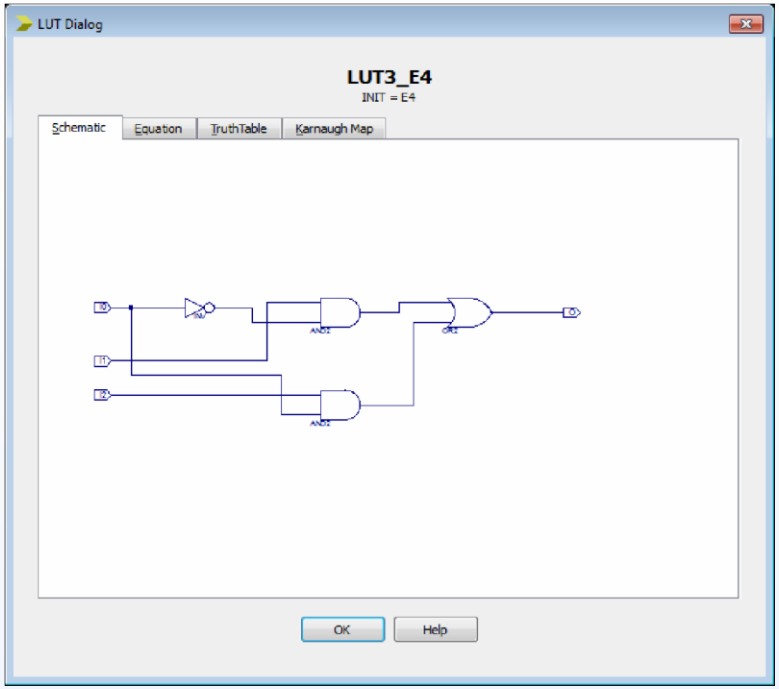




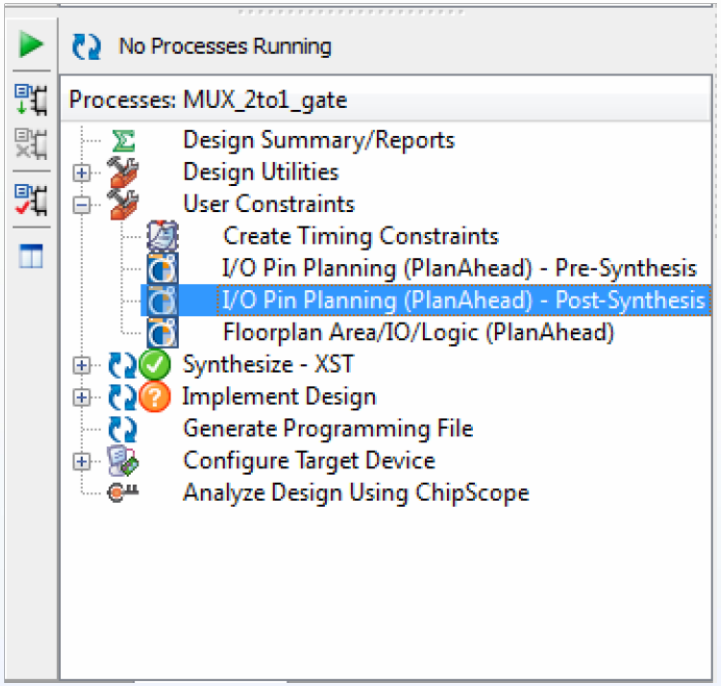
- Double click on the cell to see its components
  - Possible up to primitives
- Similar for technology schematic
- Double click on the cell to open a menu
  - Schematic, Equation, Truth Table, Karnaugh Map

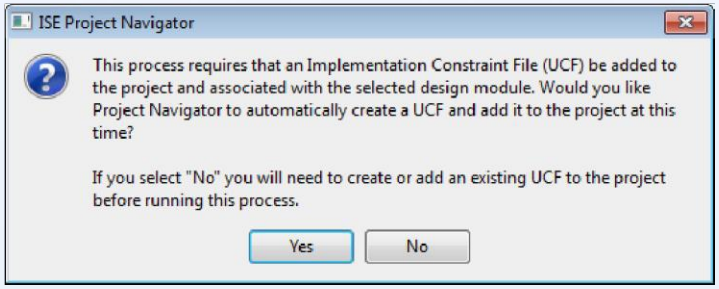
Tutorial


MUX_2to1_gate:1
lut3
ibuf
In_1_IBUF
obuf
Out_OBUF
Out1
ibuf
In_0_IBUF
ibuf
S_IBUF
MUX_2to1_gate


LUT Dialog
LUT3_E4
INIT = E4
Schematic
Equation
TruthTable
Karnaugh Map
OK
Help

Tutorial

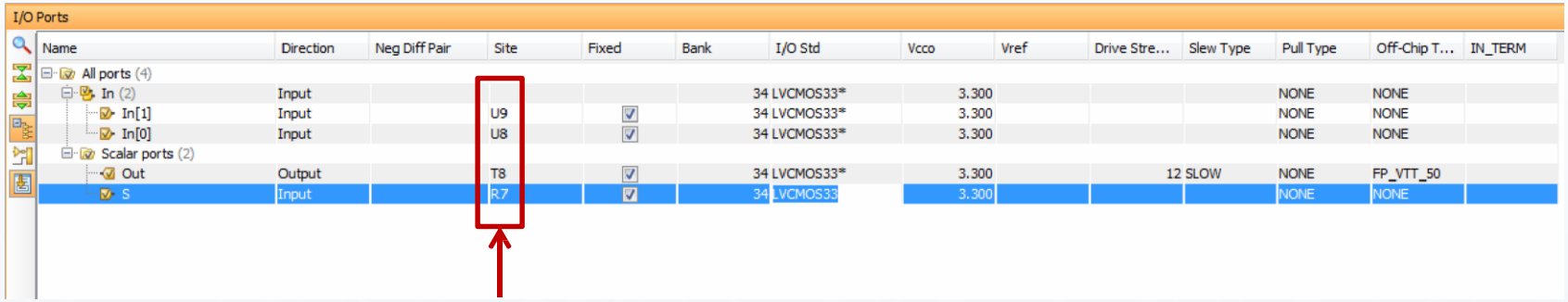




- Define the I/O ports and timing constraints
  1. Double click on "I/O Pin Planning"
  2. Select "*Yes*" to create a UCF file
- Find your I/O pins at the bottom
- Assign locations
  - Inputs to slide switches, output to LED
  - Refer to Nexys 4 manual page 18 for locations

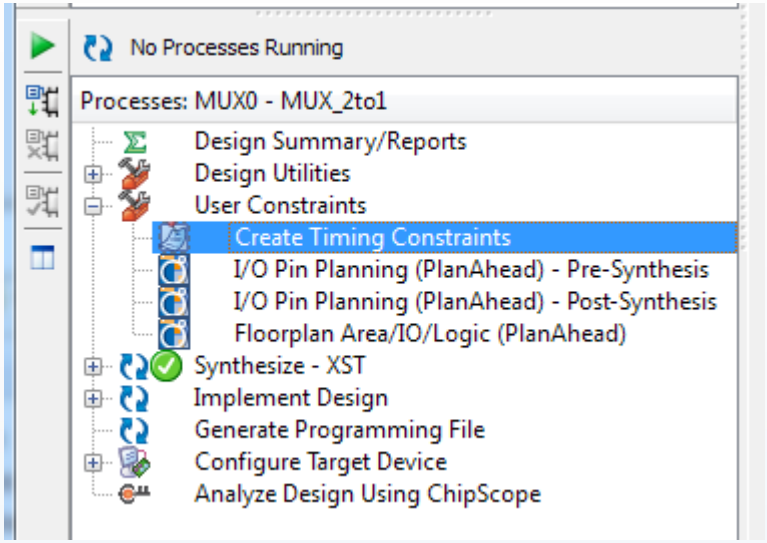
Tutorial

- Change I/O std to LVCMOS33
- Save the file



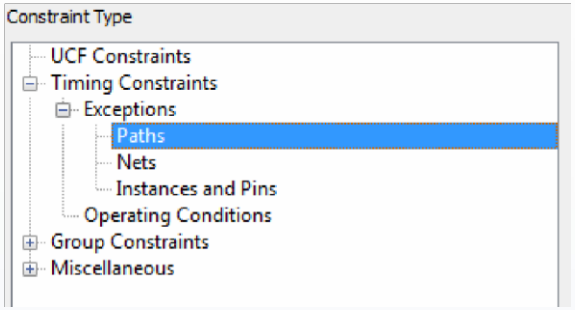
From Nexys 4 manual

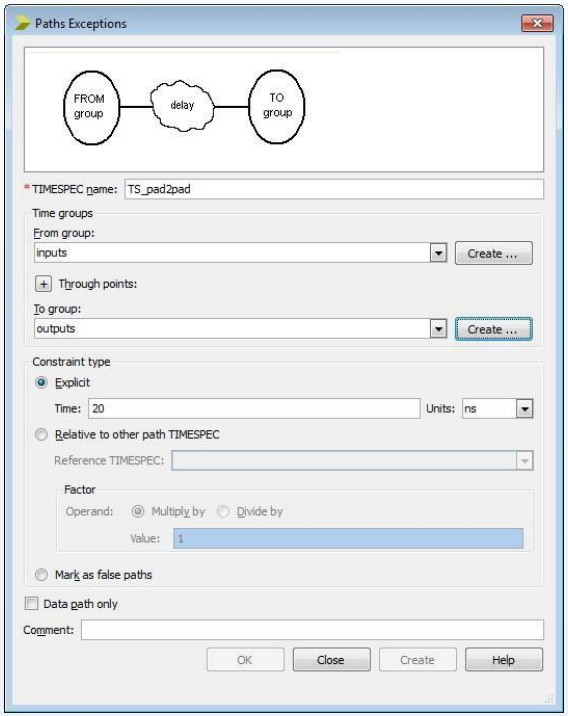
Tutorial



- Define timing constraints
  - Double click on “Create Timing Constraints”
  - Select “*Yes*” to open your UCF file

Tutorial

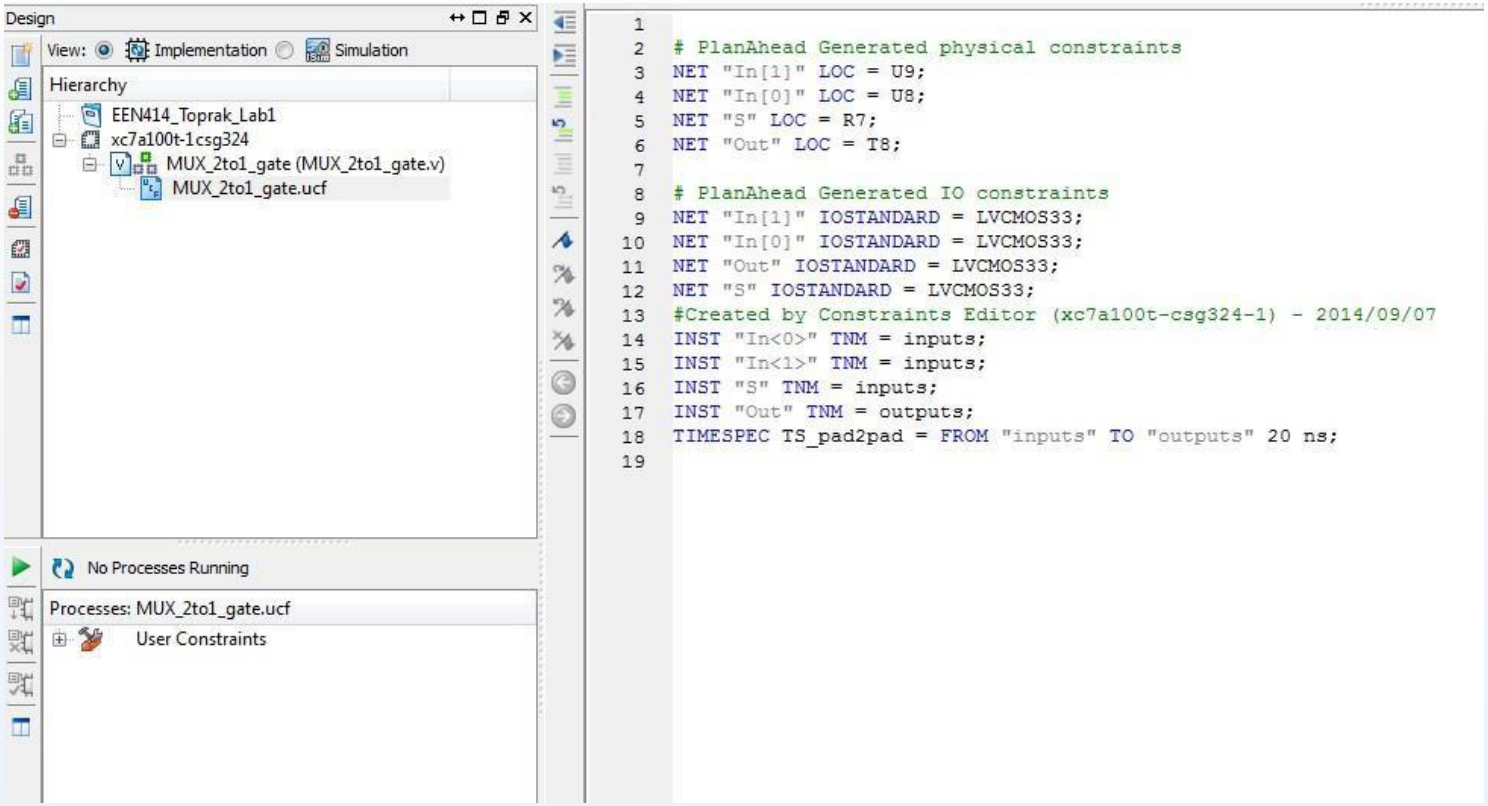




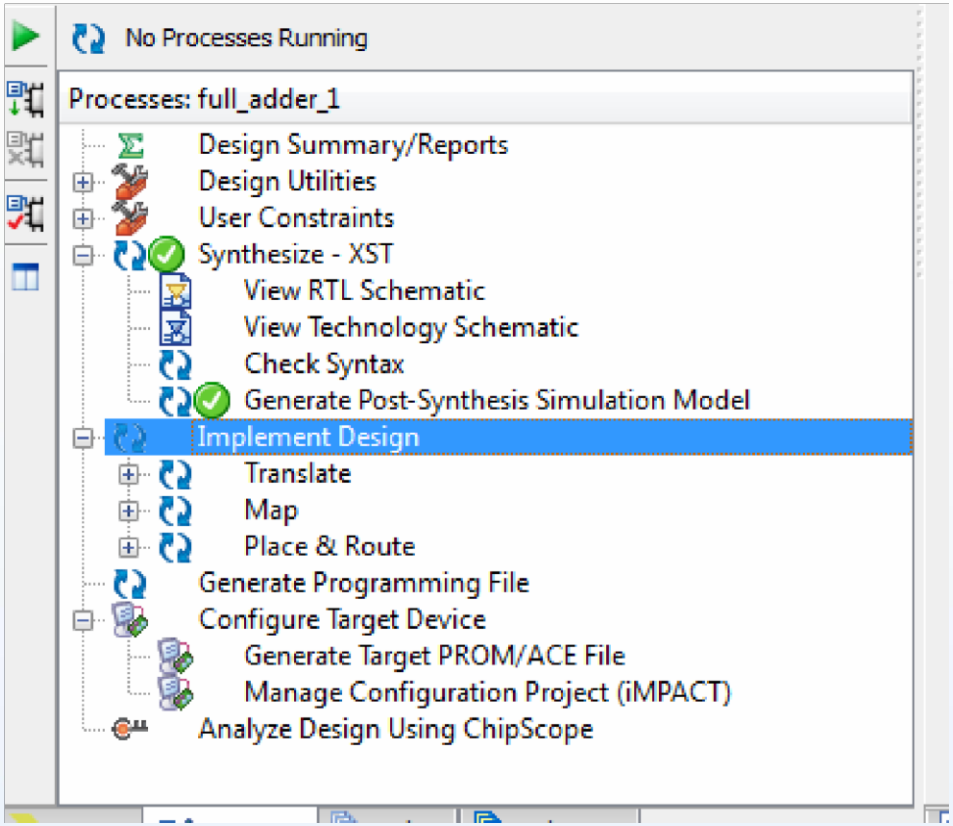
- "*Timing constraints -> Exceptions -> Paths*"
  - Right click and select "Create constraints"
  - Name the timespec (e.g., pad2pad)
  - Create groups for inputs and outputs
  - Select explicit, 20 ns
  - Make sure to see the constraint on the main screen

Tutorial

- Open your UCF file to verify the constraints



Tutorial



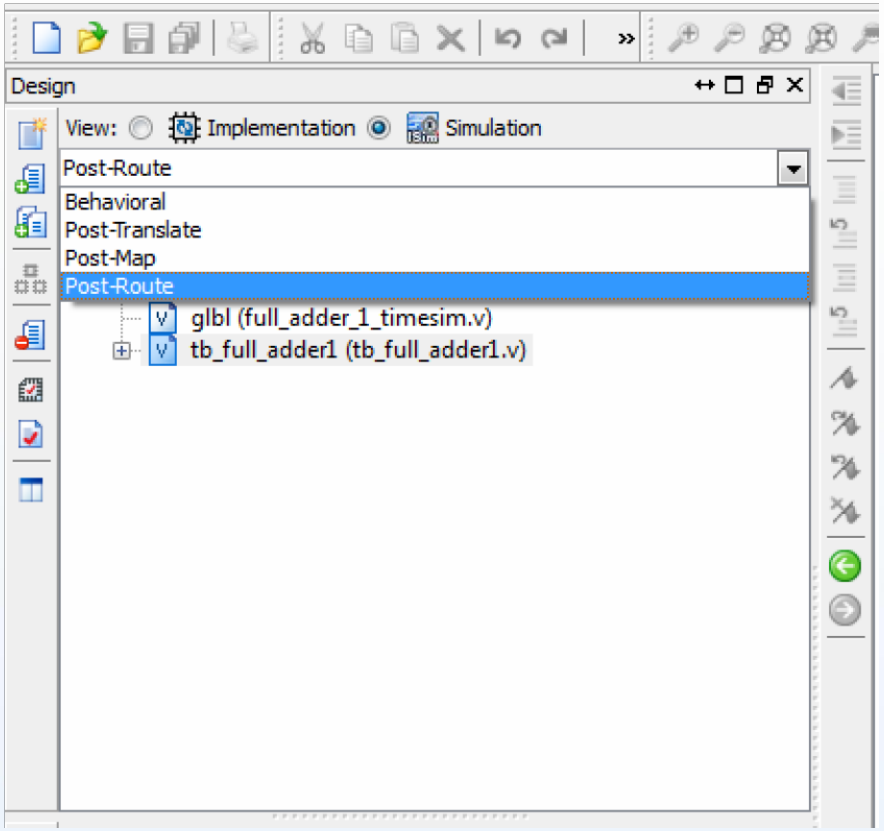
- Implement design

–Make sure there are no errors

–In case you get errors

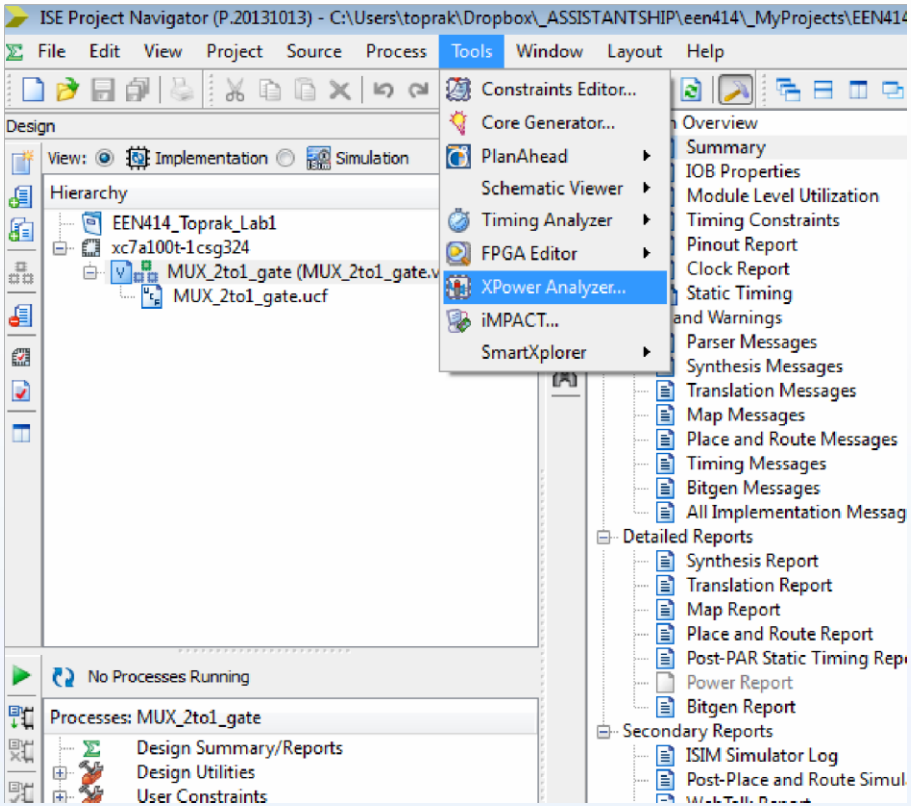
- Check the “Errors” tab at bottom left

Tutorial



- Post implementation simulations
  - Select "*Simulation*" radio button again
  - Select "Post-Route" from the dropdown menu
  - Simulate as you did before

Tutorial



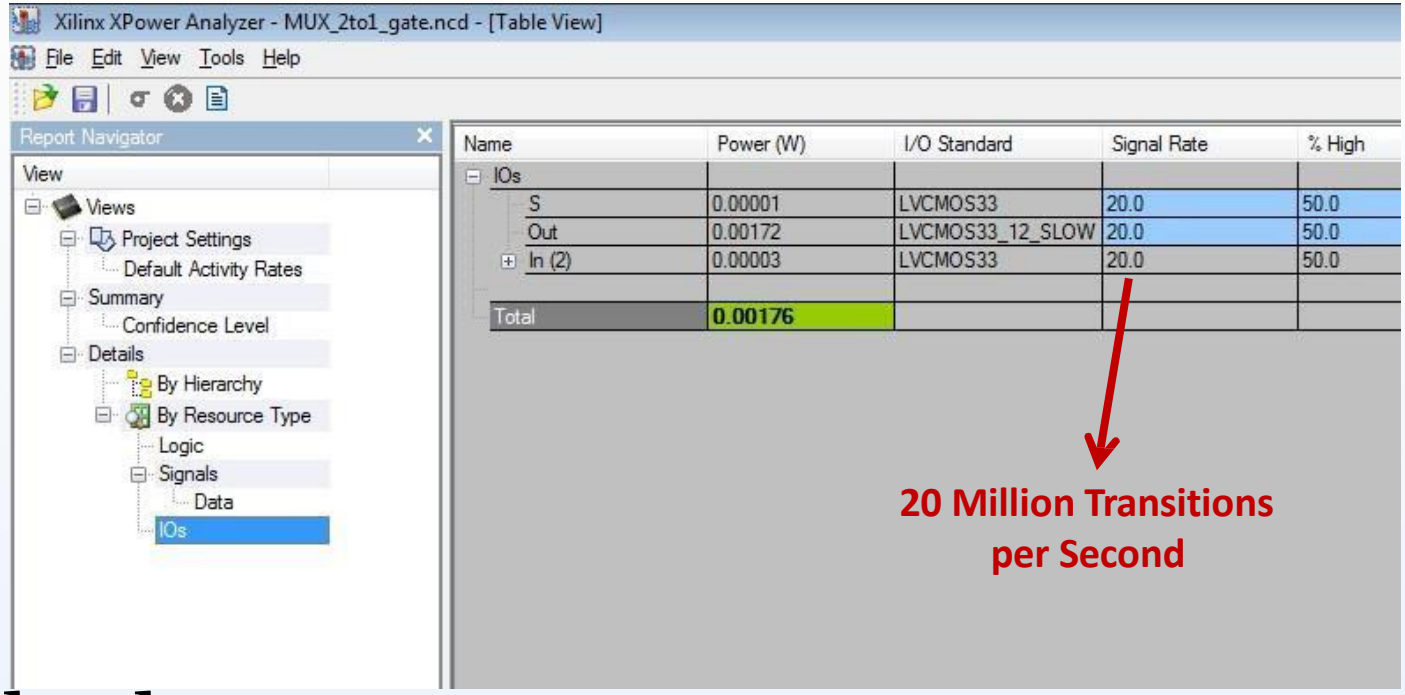
- Power analysis
  - Tools -> XPower Analyzer
  - New window opens
  - Summary page

Tutorial

- Change signal rate for all I/O signals –
  Default rate is *million transitions per second*

Tutorial

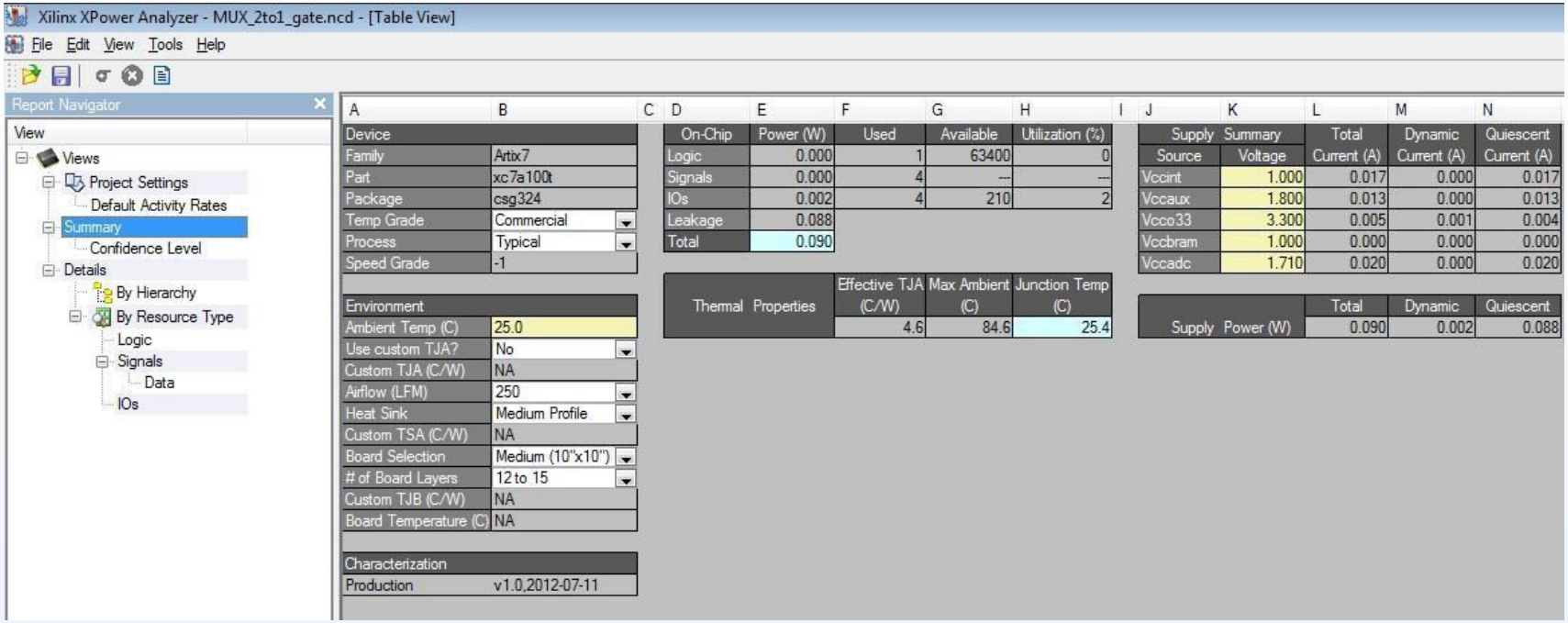
- Update results: Tools -> Update Power Analysis



- Go back to summary page – Summary of the power dissipation

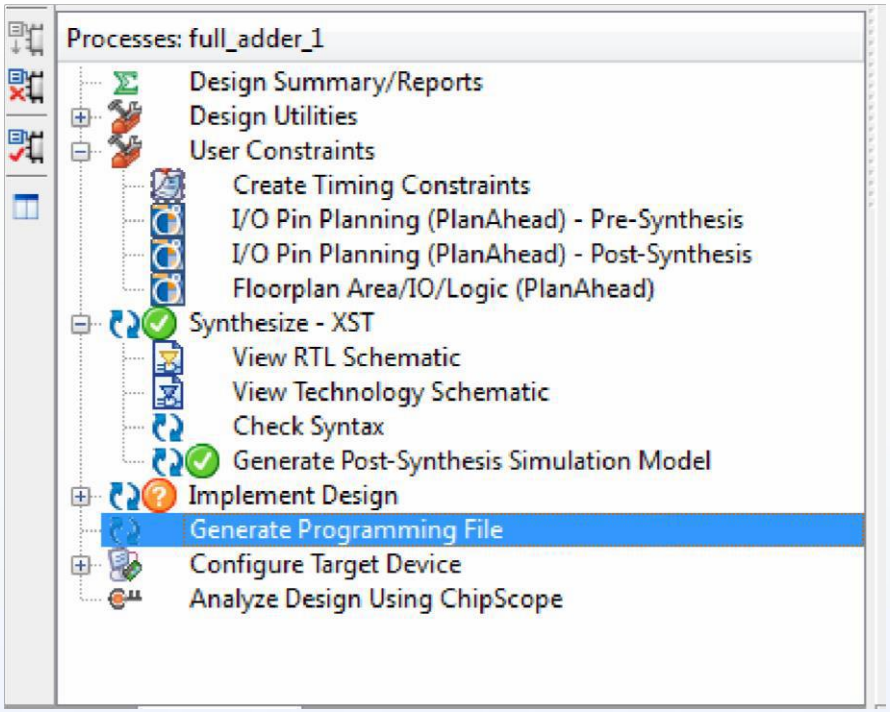
Tutorial

– IO power depends on signal rate



- Next step: Loading the design to FPGA

Tutorial

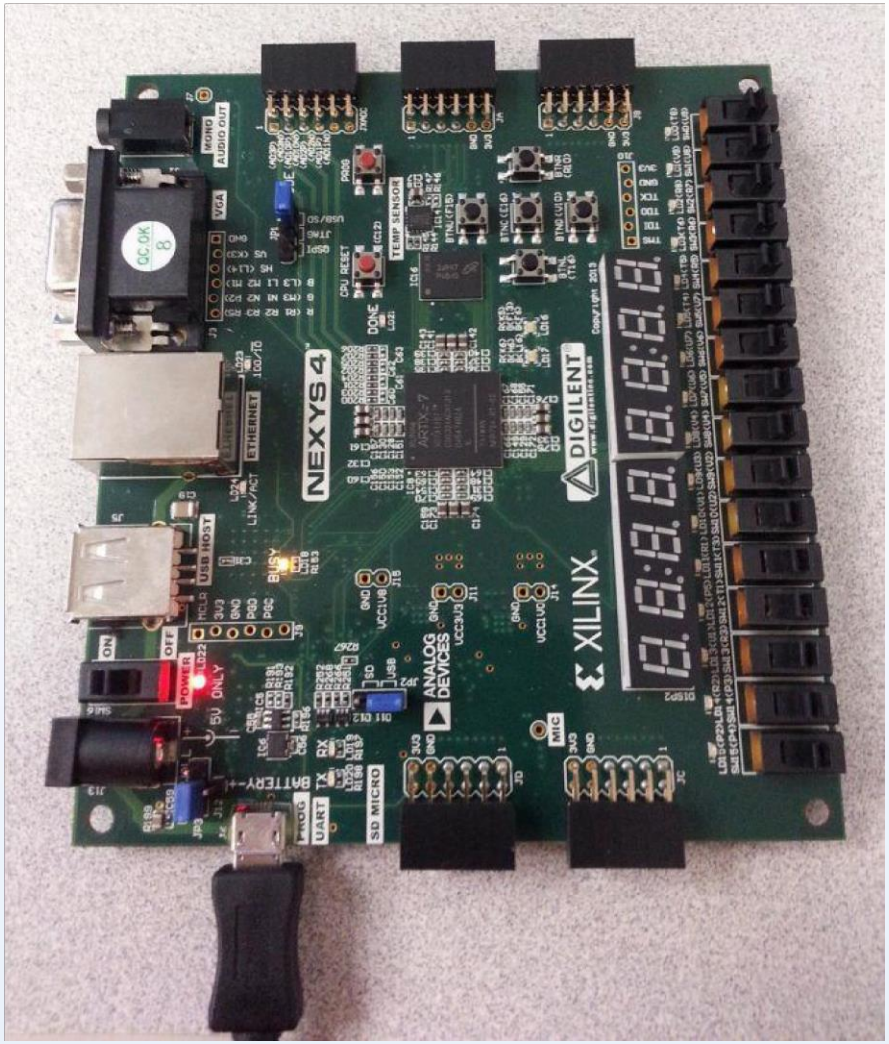


- *"Generate Programming File"*

–Creates a "*bit*" file in the project folder

– Will be loaded to FPGA

Tutorial

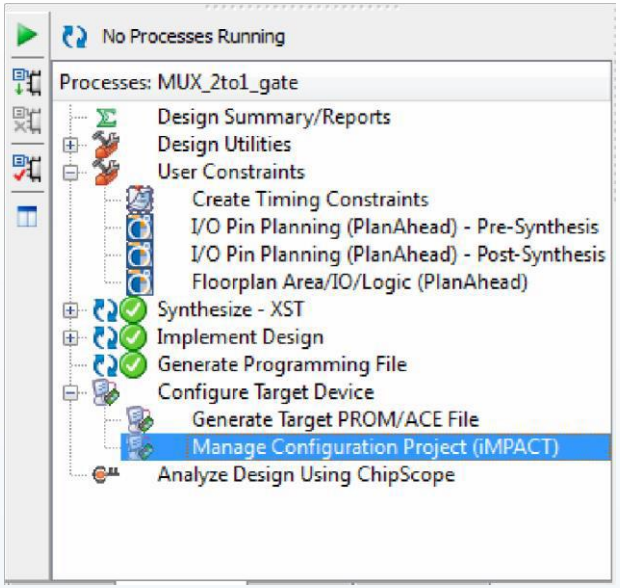


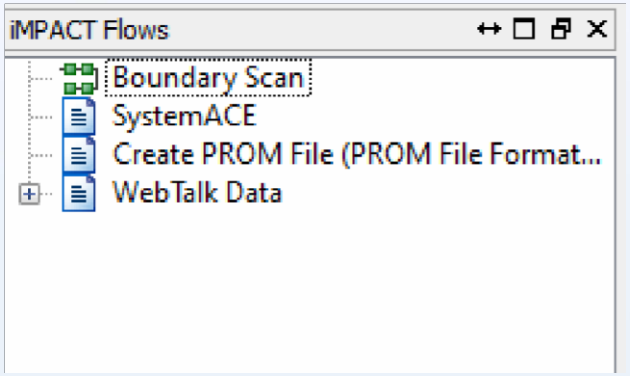
- Loading the bit file to FPGA

1. Connect Nexys4 to the computer using the USB cable

2. Turn on device power

Tutorial

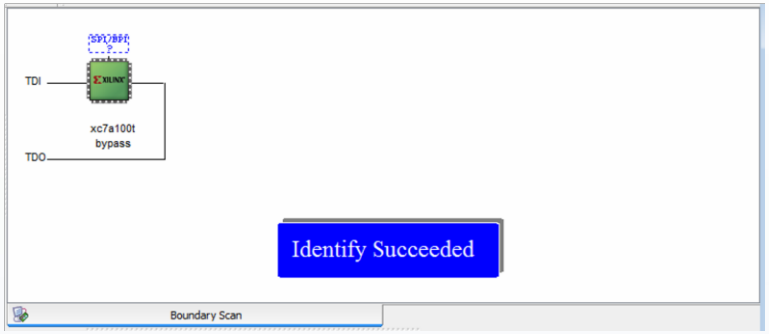




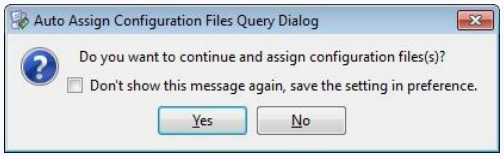
3. Select “Manage Configuration Project (iMPACT)”
4. Select “Boundary Scan” in iMPACT
5. Right click on the main window and select “Initialize Chain”

Tutorial

- FPGA will be recognized



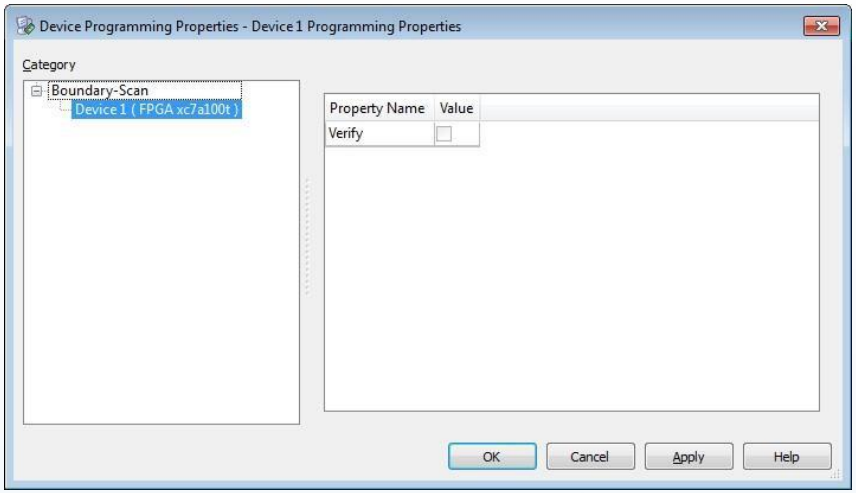
  - Select “Yes”



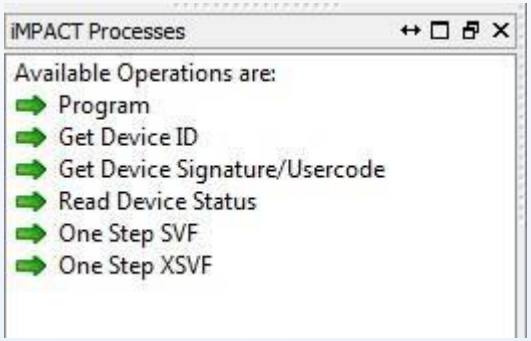
  - Select your bit file

  - Select “No”

Tutorial

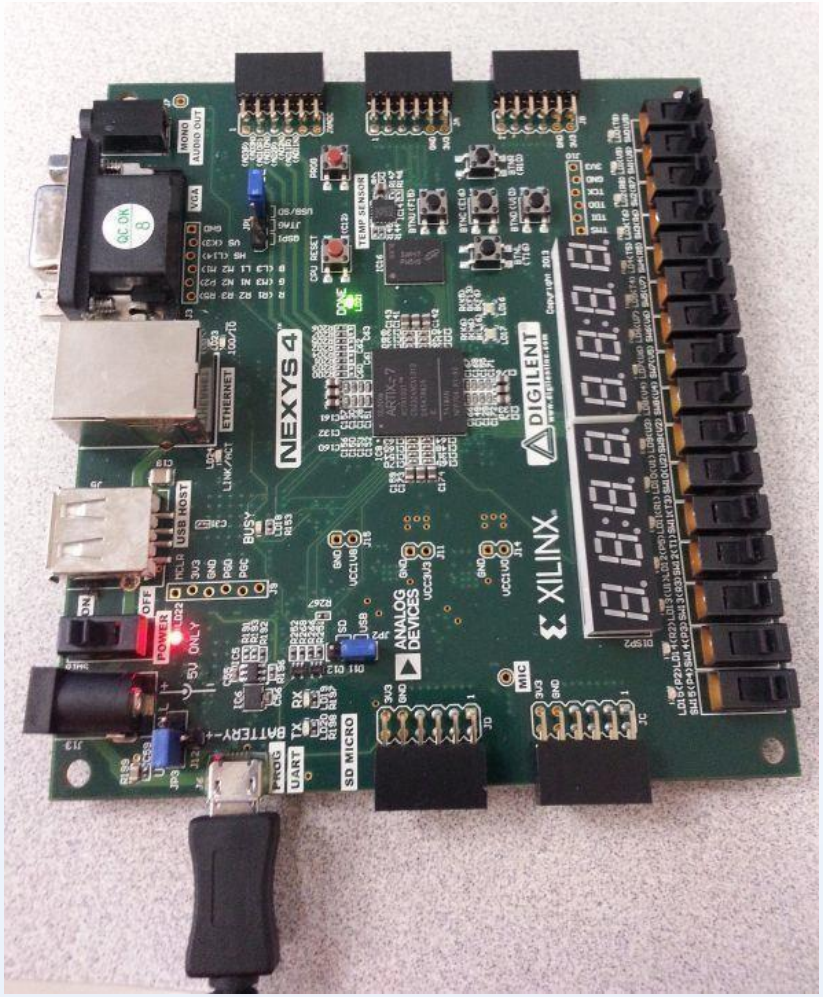


- Select “OK” on the “Programming Properties” window



- Double click “Program” under “Available Operations”
- Verify that “DONE” LED on the board is on

Tutorial



- Test your design using switches and LEDs!

Reference

- Digital Design with FPGAs – Dr. Onur Tigli
- www.altera.com
- www.xilinx.com
- www.digilent.com
- Nexys4 User Manual