**Faculty of Engineering and Technology**

**Department of Electrical and Computer Engineering**

**Dr. Khader Mohammad**

**Nour Daghlas**

**4/29/2019**

**ICC Hierarchical Design for VLSI**

**Lab4**

**Goal: Synthesize and Design a Complex System Using ICC Tool**

**Procedure**:

During this lab, the students are expected to be able to perform the following:

1-Create a MilkyWay Design Library

2-Synthesize and Design a MUX2X1 using gates.

3-Synthesize and Design a MUX4X1 using MUX2X1.

4-Write Netlist for full adder using your MUX4X1.

~5-Use premade simple Verilog Code which includes D-FlipFlops and Synthesize Clock Distribution and verify timing.~

6-Implement a hierarchal 4-bit RippleAdderSubtractor using previous components

7-Verify LVS and DRC errors and fix where applicable

8- ¡Bonus! Use your RippleAdderSubtractor and store the output SUM in 4 D-FlipFlops to perform clock tree synthesis and verify the timing of the design.

\*To start the lab, have the ICC Tool open and ready through the icc\_shell –gui command after you’re in an appropriate directory

Part 1: Create a MilkyWay Design Library

For this lab, we’ll create our own Design Library. A design library holds information of the technology we’ll be using in our design and the reference libraries which will link the standard cells that our designs are allowed to use that we can later re-open and view our past designs without having to redo the entire design process.

To create a library, copy the following command and paste it in the ICC Tool Command Line(Main Window) as shown in Figure 1.

create\_mw\_lib -technology /home/iccuser/32-28nm\_EDK\_01312018/SAED32\_EDK/tech/milkyway/saed32nm\_1p9m\_mw.tf -mw\_reference\_library {/home/iccuser/32-28nm\_EDK\_01312018/SAED32\_EDK/lib/stdcell\_lvt/milkyway/saed32nm\_lvt\_1p9m} /home/iccuser/icc\_test/YOUR\_ID/YOUR\_ID.mw

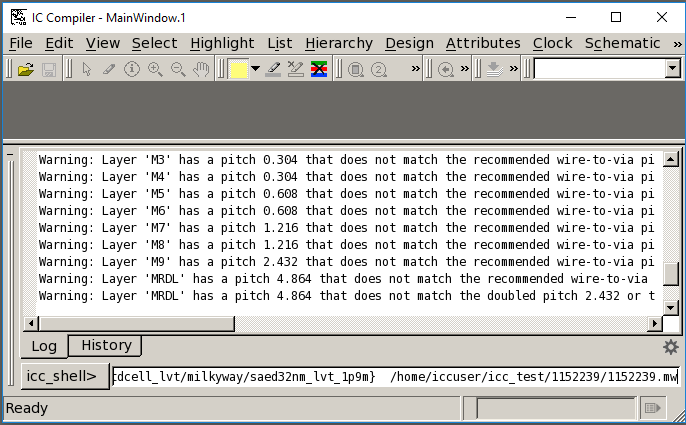


Figure 1

To open the library, go to FILE -> Open Library -> Choose Your Library and press Choose. The library should be opened. This step is crucial: Type in the ICC Command line the following:

source ~/setup.tcl To re-open your library, simply go to File -> Open Library and choose your pre-existing library

For the rest of this lab, we’ll need to switch between the ICC Shell and the Linux Kernel. To not have to close the library and re-open it, you can right click on PuTTY and start a Duplicate Session to access your files and the ICC tool at the same time.

Part 2: Synthesize and Design a MUX2X1 using gates.

1. Using the example AND2X1.v configuration shown in Figure 1, write your own MUX2X1.v file using the Linux “vi” editor and save it to your directory.

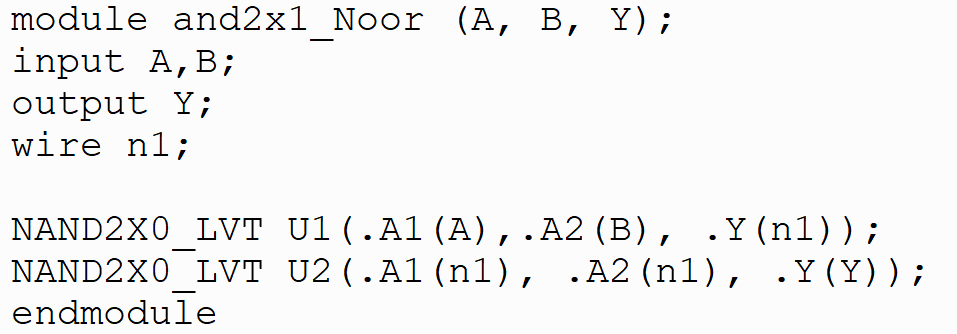


Figure 2

The MUX2X1 Circuit using basic gates is as shown in Figure 2.

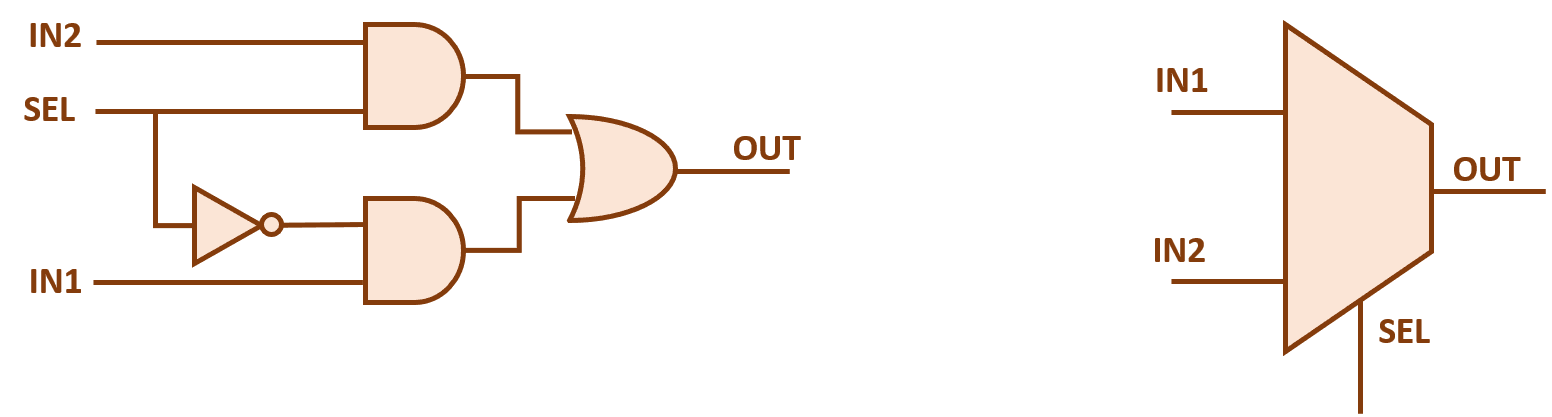


Figure 3

The AND2X1.v and MUX2X1.v are called “Netlists”, which are a gate-level representation of the design we want to implement.

1. Now Synthesize(Import) the netlist into the IC Compiler and proceed to create the Floorplan, pre-routing, placement and routing and check the design to see how the connected nets and ports would give us a MUX2X1 functionality.

Part 3: Design and Synthesize a MUX4X1 using MUX2X1.

This part is an introduction into hierarchal design, there are two approaches when we’re performing hierarchal design(as opposed to flat design) as follows:

1. top-down decomposition of a behavior specification into less complex behavior specification modules.

2. bottom-up combination of physical building blocks into larger building blocks.

During this lab, we’ll rely mainly on Top-Down approach where we construct the smaller parts needed to implement our main design.

In order to synthesize designs with hierarchal properties in the ICC Tool, we’ll have to import files using **File->Import Design -> Choose Verilog -> Add** and **Select** all the needed netlists for Synthesizing the design.

The MUX4X1 using MUX2X1’s netlist is one of your tasks, so you can hand-draw it and then translate it to the Netlist. In order to use the premade MUX2X1, we’ll have to pay special attention to our MODULE names(Be careful! Not the file names). Using a MUX2X1 in a netlist is shown in Figure 4.

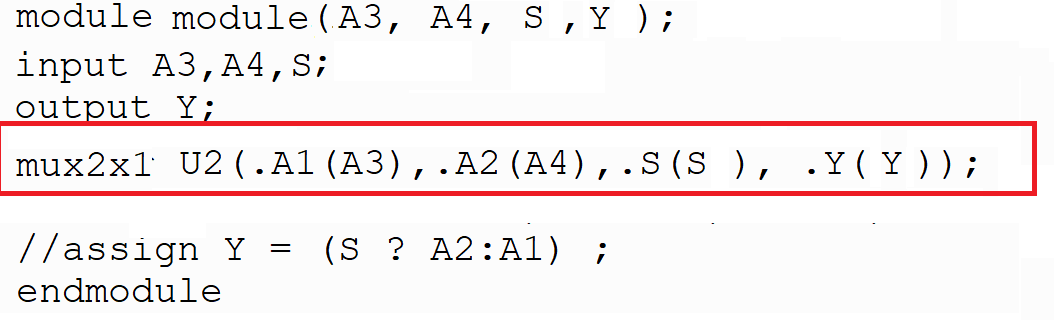


Figure 4

Part 4: Write Netlist for full adder using your MUX4X1.

In this part, we only want to create a netlist for a full adder and save it to use it in a later design. It is not required to synthesize the design, but it’s optional in order to add it to your library for future use. You could use MUX2X1 as shown in figure 6 but that’s up to the designer. Is there a reason why we might use MUX2X1 if we could use MUX4X1 for our design?

The Full Adder circuit is shown in the following Figure 5.

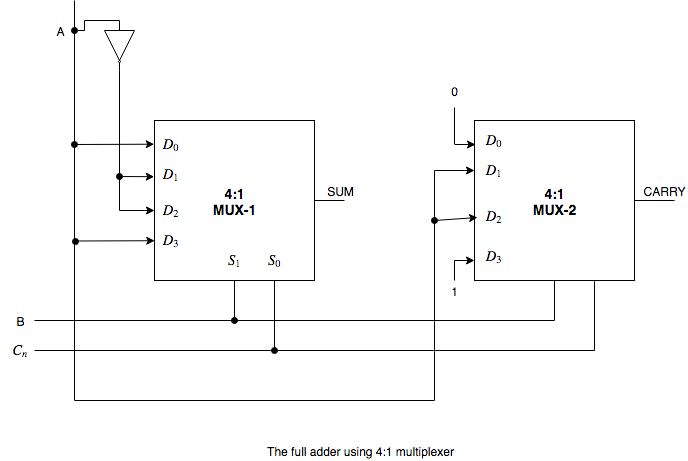


Figure 5

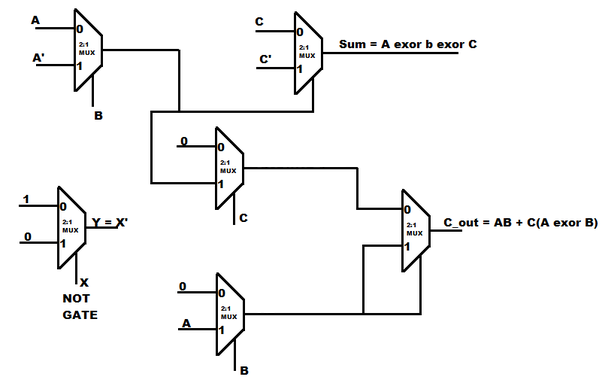


Figure 6

Part 5: Use premade simple Verilog Code which includes D-FlipFlops and Synthesize Clock Distribution and verify timing.

File: ~/flipFlops.v

Use commands: create\_clock -name myClock [list clk] -period 100

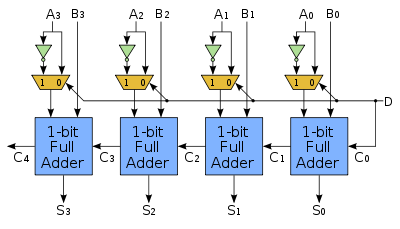
clock\_opt

route

Main Window -> Schematic -> New Design Schematic View (From, To, Through)

Report Clock Tree

Part 6: Implement a hierarchal 4-bit RippleAdderSubtractor using previous components



Lastly, if you have any questions, please ask your lab assistant.