

ECED2200 – Lab #2

Pre-Lab Information

It is recommended that you read this entire lab ahead of time. Doing so will save you considerable time during the lab.

There is also several videos showing the use of the software tools – you will save considerable time by watching those now. See <http://colinoflynn.com/teaching/eced2200-intro-to-digital-circuits/> and look at the 'lab 2' information, links (if available) are provided.

Overall Objective

The objective of this lab is to experiment with adder circuits and learn their characteristics. In addition some more advanced features of the design software used will be explored such as creating subcircuits.

Deliverables

A standard lab report is due for this lab. See the requirements at colinoflynn.com/teaching . Note: you are requested to take screen-shots of certain results and include that in your lab report. If you don't know how to use the print-screen button/feature see this video:

<http://www.youtube.com/watch?v=6nZjo2WpEe0&hd=1> .



CC BY-SA

This work by [Colin O'Flynn](#) is licensed under a [Creative Commons Attribution-ShareAlike 3.0 Unported License](#) .

Part #1: Simulating Half Adder, Creating a Subcircuit, Simulating a Full Adder

Objective

- Build a half adder from basic logic gates
- Turn that half adder into a subcircuit
- Use half adder to create full adder

Required Materials

- Computer with Xilinx ISE 13.2 Webpack installed.
 - All computers in the lab have this installed.
 - This is free software so you can install on your own computer if you wish, you can download it from <http://www.xilinx.com/support/download/index.htm> - select '13.2' on the side. The file is very large so you may wish to download at school, and you are required to register to license it.
- Example project file DigitalTrainer_Simple.zip
 - This file contains an environment similar to the digital trainer you will use for testing hardware with.

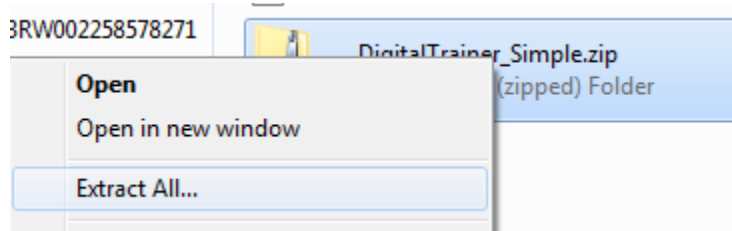
Procedure

NOTE: There is a video version of this procedure at

<http://www.youtube.com/watch?v=OqHU214TlnQ&hd=1> which will be much easier to follow along.

Creating the Half Adder

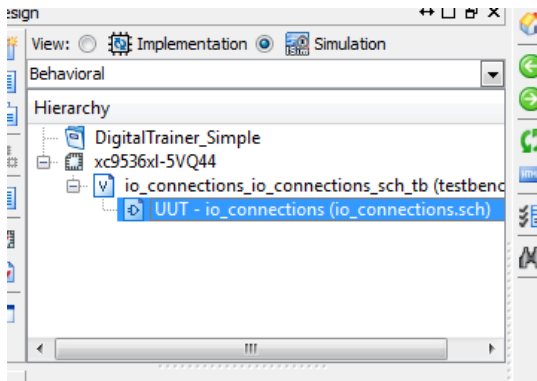
1. Unpack the given ZIP file somewhere. If using lab computers suggested to save to your personal drive or save to USB stick. Rename the folder something memorable, as you may save time by reusing parts of this lab later.



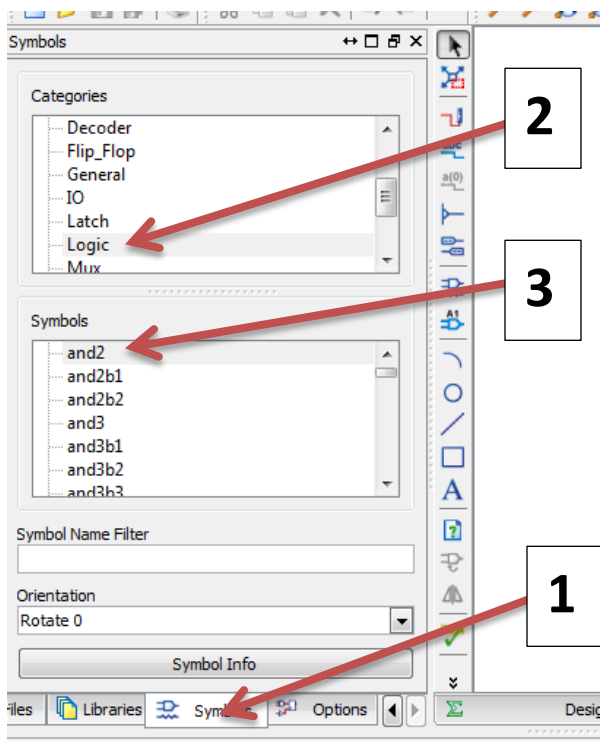
2. Open the resulting folder (it may open automatically), double-click on "DigitalTrainer_Simple.xise" which will open the Xilinx Project Navigator
3. Double-click on the 'UUT' schematic file (io_connections.sh):



Digital Circuits – Lab #2



4. Go to the 'Symbols' tab in the window that opens, select 'Logic' as the category, and 'and2' as the Symbol:



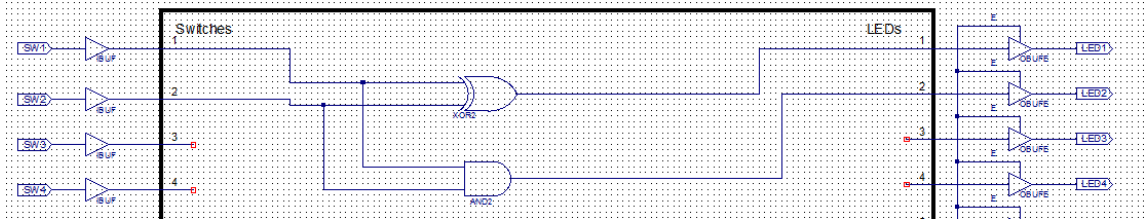
5. Place the AND gate into the center area. You may need to zoom in to see better.
6. Similarly, place a 'xor2' gate.



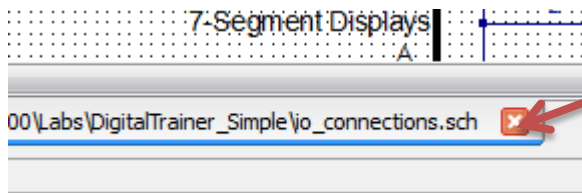
CC BY-SA

This work by [Colin O'Flynn](#) is licensed under a [Creative Commons Attribution-ShareAlike 3.0 Unported License](#).

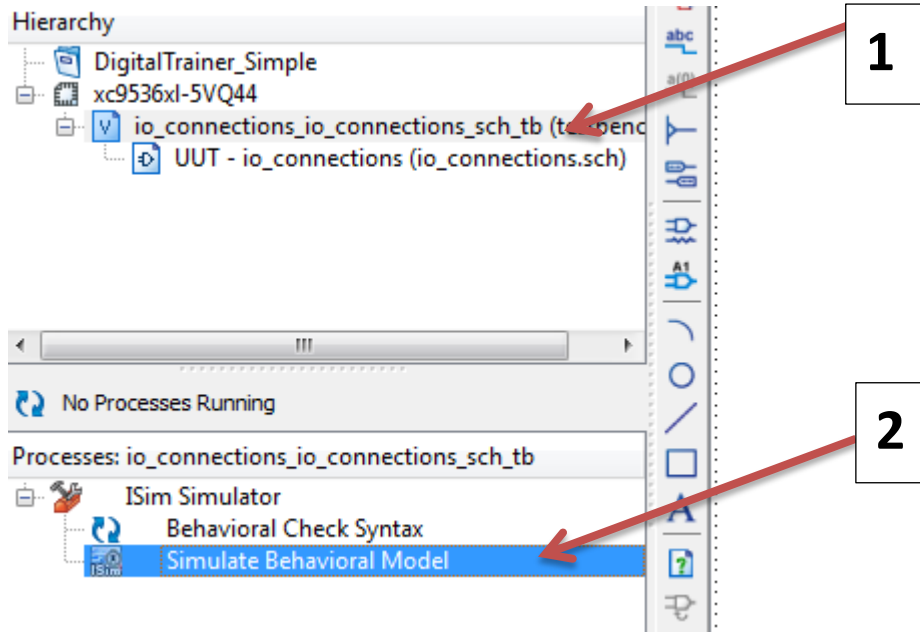
Digital Circuits – Lab #2



7. Connect up the gates as a half adder (schematic of half adder given at end). Use the switches as the inputs and LEDs as the outputs:
8. Save the file, then close JUST that file (NOT the whole project):



9. Select 'io_connections_ioconnections_sch_tb', then double-click 'Simulate Behavioural Model'. You may need to hit the '+' beside 'ISim Simulator':



10. In the window that opens, change to the 'Default.wcfg' tab:

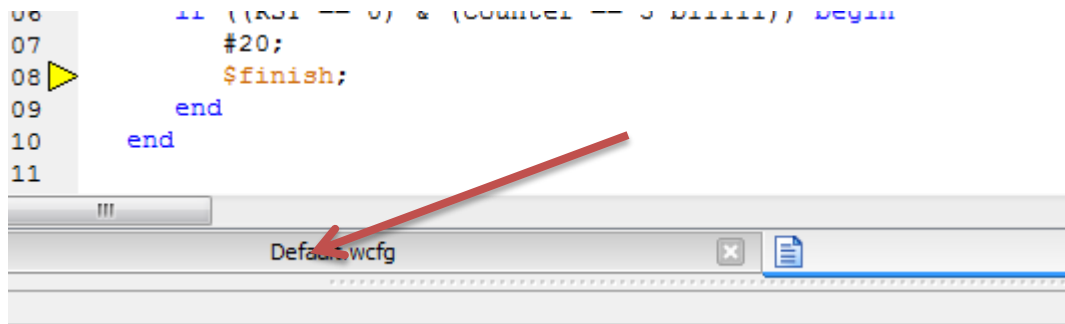


CC BY-SA

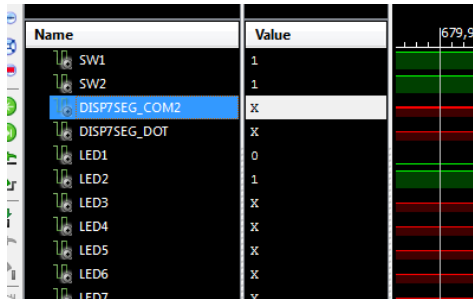
This work by [Colin O'Flynn](#) is licensed under a [Creative Commons Attribution-ShareAlike 3.0 Unported License](#).

Digital Circuits – Lab #2

```
06      11 ((SW1 == 0) & (COUNTER == 0)) begin
07          #20;
08      $finish;
09  end
10 end
11
```

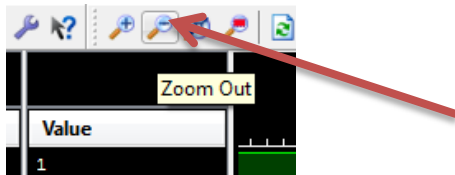


11. Delete any unused signals. In this lab we only use signals SW1,SW2,LED1,LED2. Delete a signal by clicking it and hitting 'delete':



Name	Value
SW1	1
SW2	1
DISP7SEG_COM2	X
DISP7SEG_DOT	X
LED1	0
LED2	1
LED3	X
LED4	X
LED5	X
LED6	X
LED7	X

12. Use the zoom out button to get a good view of the entire waveform.



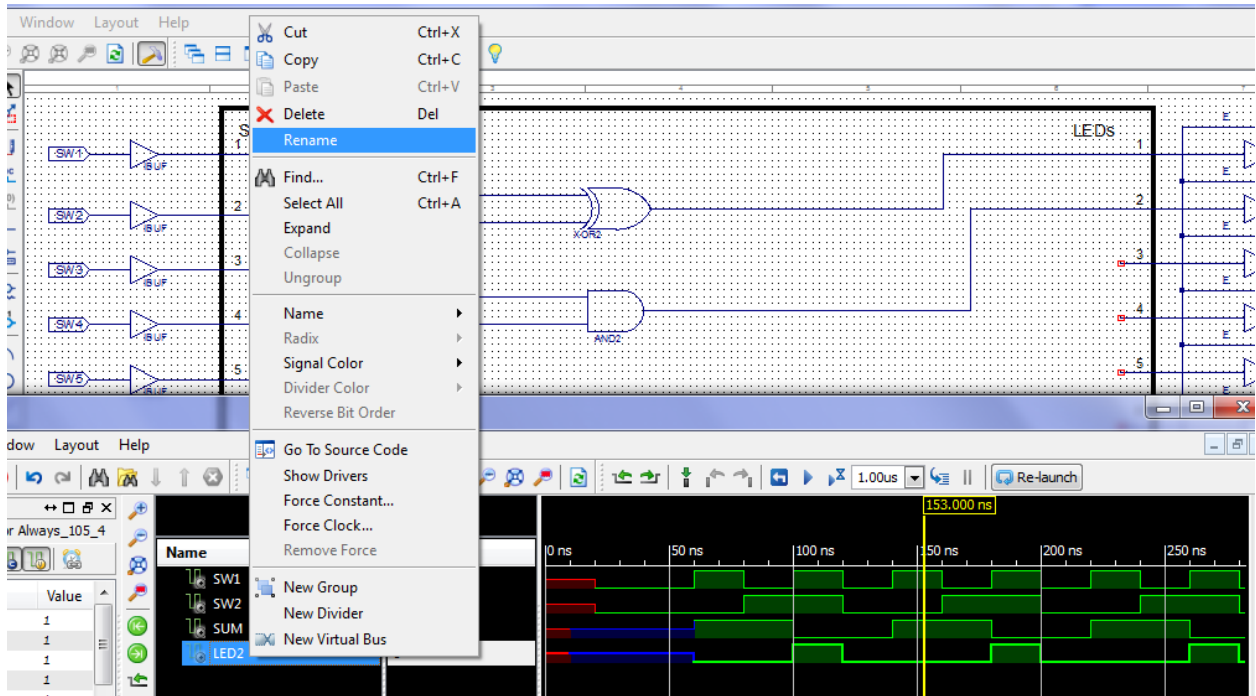
13. You can rename the inputs/outputs by right-clicking them and hitting 'rename'. If you forget which ports are which, you can look back in ISE Project Navigator (not the simulator) and open the schematic:



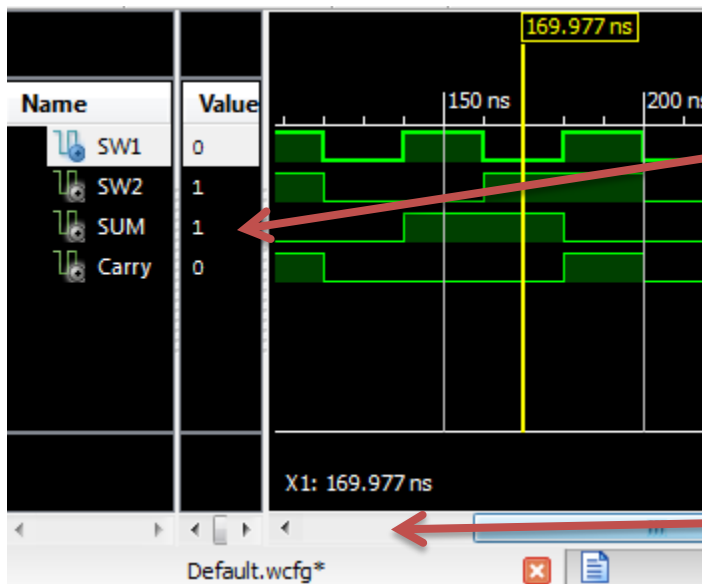
CC BY-SA

This work by [Colin O'Flynn](#) is licensed under a [Creative Commons Attribution-ShareAlike 3.0 Unported License](#).

Digital Circuits – Lab #2



14. Using the waveform display, fill in the truth table. You may need to scroll the waveform to start at time 0. You can click on different times in the waveform and just read A/B/Y directly off. Fill in the observations based on this.



**Value of
A/B/Sum/Carry at
point shown by
yellow line**

Scroll

15. Close the ISim window, it will ask if you really want to exit the application, hit “Yes”.

Creating a Full Adder

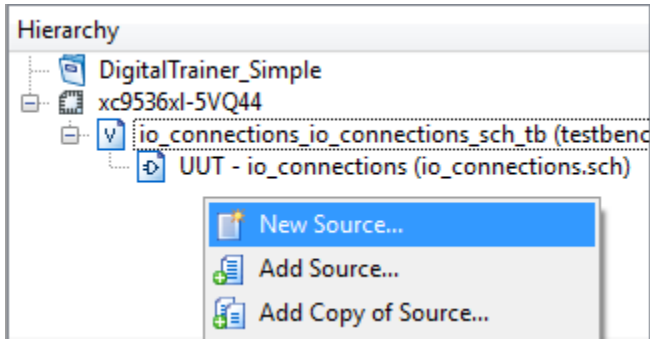
16. Right-click in the ‘Hierarchy’ area and hit ‘New Source’:



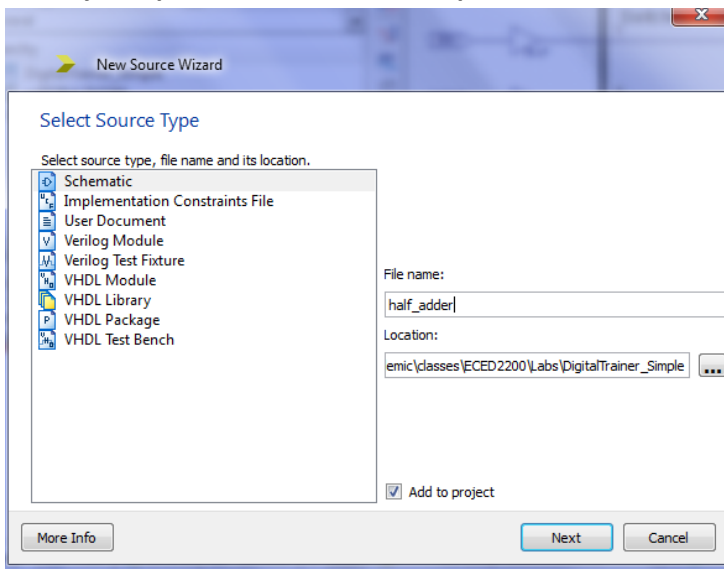
CC BY-SA

This work by [Colin O’Flynn](#) is licensed under a [Creative Commons Attribution-ShareAlike 3.0 Unported License](#).

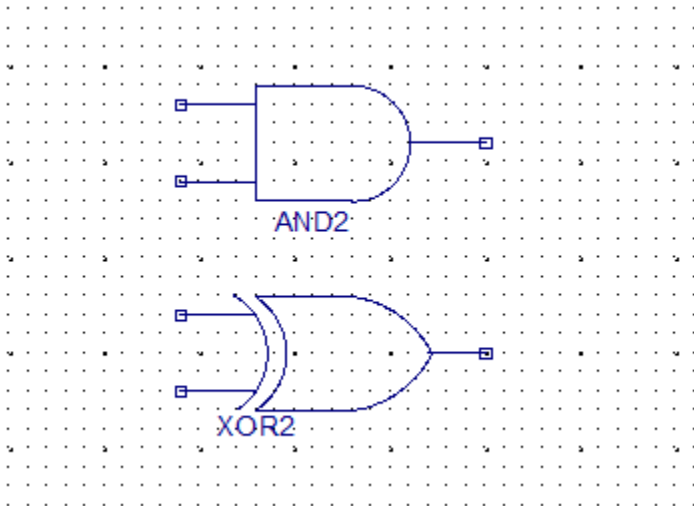
Digital Circuits – Lab #2



17. Select schematic as type of source and name it 'half_adder' or similar, then hit next, then finish.
NOTE: ONLY Use A-Z, 0-9, and underscores in the name. Do not use dashes or spaces, it will result in errors. For example 'half-adder' and 'half adder' are all INVALID names that may be initially accepted but will later cause problems.



18. We are going to redraw the half adder. Add the AND2 and XOR2 gates again:



CC BY-SA

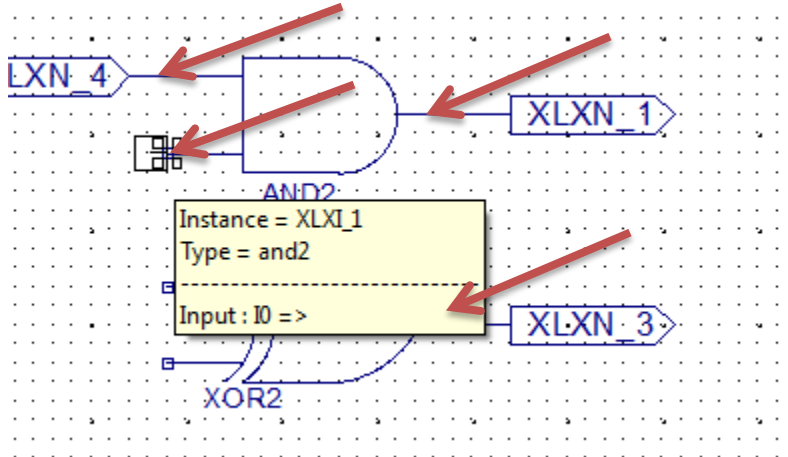
This work by [Colin O'Flynn](#) is licensed under a [Creative Commons Attribution-ShareAlike 3.0 Unported License](#).

Digital Circuits – Lab #2

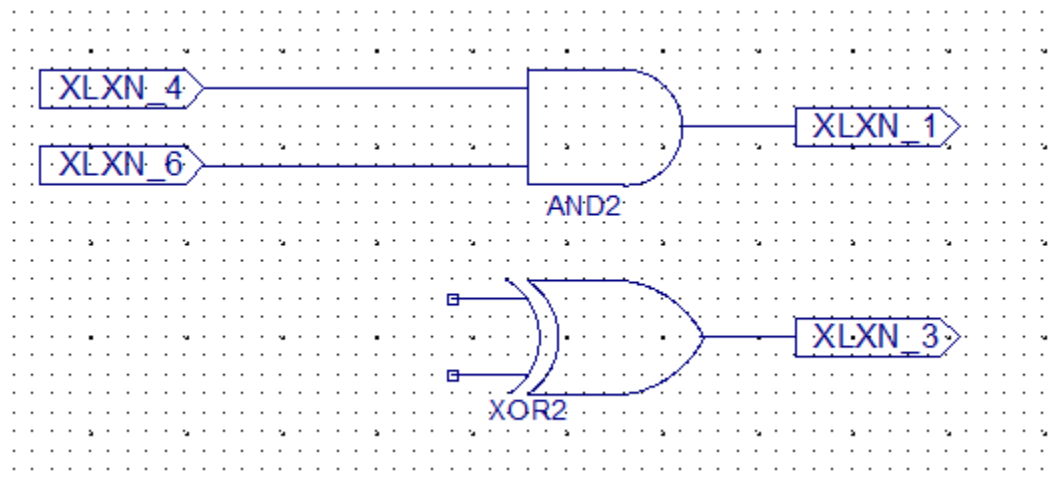
19. Click the 'Add I/O Marker' button:



20. Click on each of the two outputs from each gate, and both inputs to one gate:



21. Use the pointer tool to move the two input ports more to the left, this is required to allow adding additional wiring:



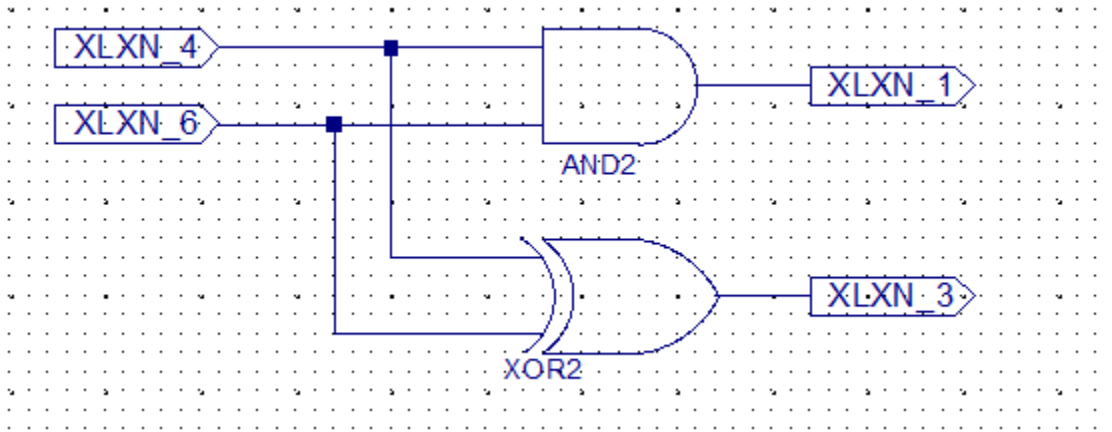
22. Wire up as a half-adder again:



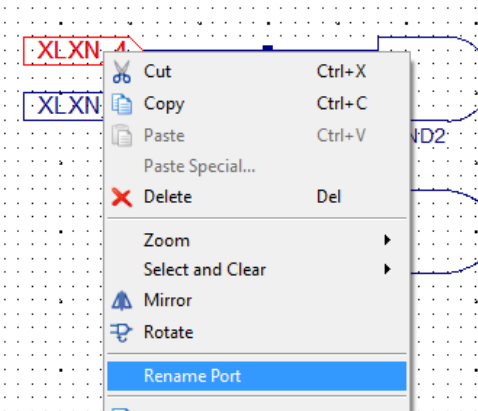
CC BY-SA

This work by [Colin O'Flynn](#) is licensed under a [Creative Commons Attribution-ShareAlike 3.0 Unported License](#).

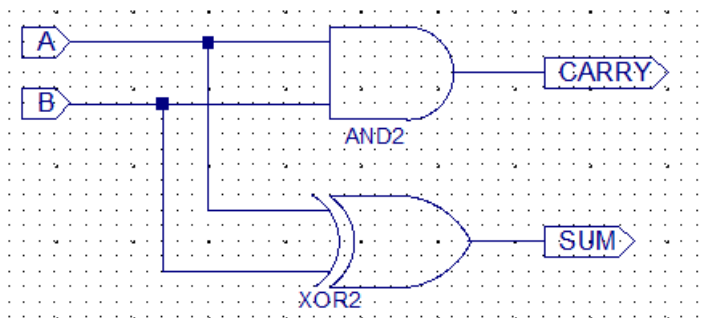
Digital Circuits – Lab #2



23. Hit 'Esc' to exit wire mode, or click on the pointer. Right-click on each port, hit 'Rename Port':



24. Give ports appropriate name:



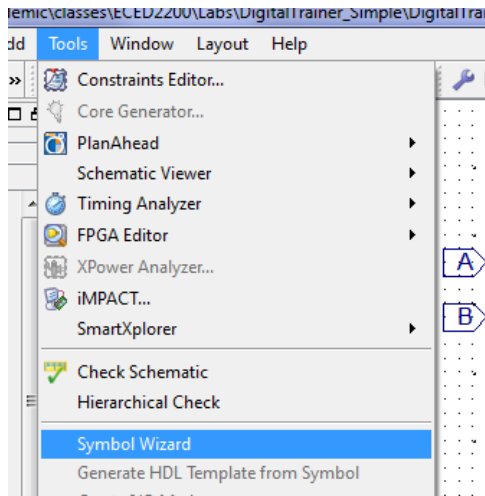
25. Under 'Tools' select 'Symbol Wizard':



CC BY-SA

This work by [Colin O'Flynn](#) is licensed under a [Creative Commons Attribution-ShareAlike 3.0 Unported License](#).

Digital Circuits – Lab #2



26. Change the 'pin name source' to 'Using Schematic'. Then change the 'schematic' to 'half_adder', or whatever you named yours:

Source Page

Select the source for pin names and the symbol shape.

Pin name source

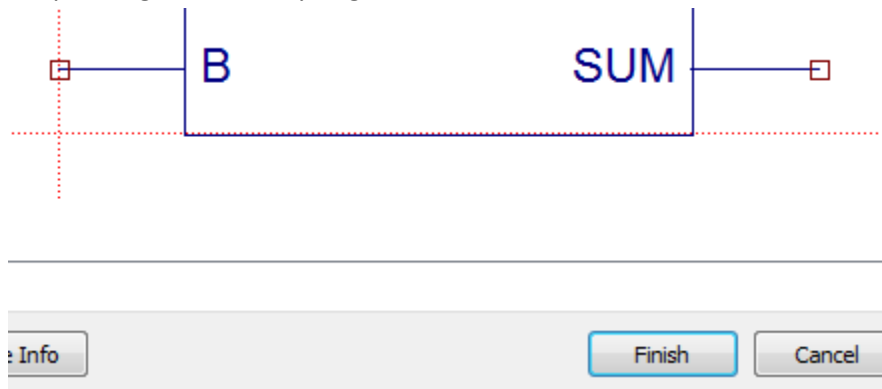
Specify manually

Using schematic

Using symbol

Import symbol attributes

27. Keep hitting 'Next' until you get to finish, then hit that too:



28. Open io_connections.sch again (you can close the half_adder schematic & symbol file). Under 'symbols' you should have a new Category, which is your personal library. Add the 'half_adder' symbol twice:



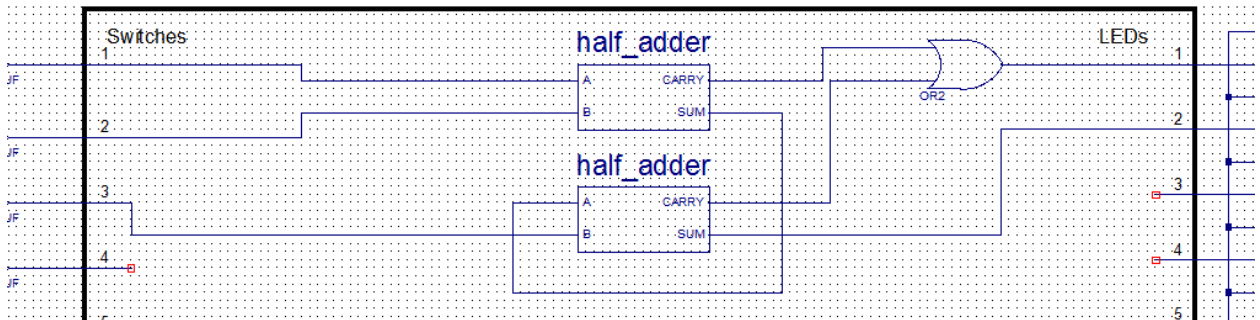
CC BY-SA

This work by [Colin O'Flynn](#) is licensed under a [Creative Commons Attribution-ShareAlike 3.0 Unported License](#).

Digital Circuits – Lab #2



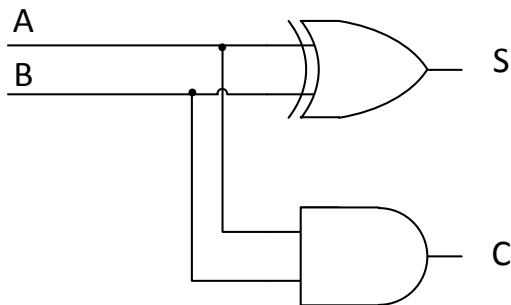
29. Wire up as a full adder by using an OR gate to combine the two carry outputs, and feeding one half adder to another half adder:



30. Simulate this in a similar manner to the half_adder. This time there is an extra input, so you need to keep SW1, SW2, SW3, LED1, LED2.

Half/Full Adder/Subtractor Schematics

Half Adder



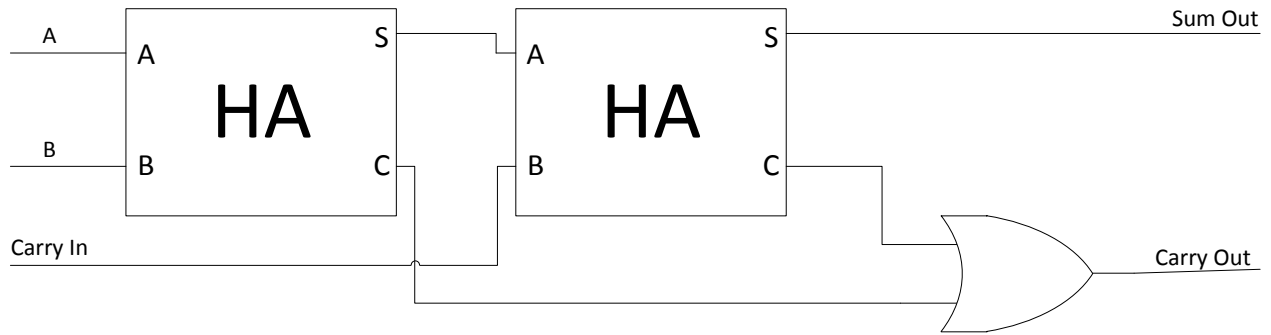
Full Adder



CC BY-SA

This work by [Colin O'Flynn](#) is licensed under a [Creative Commons Attribution-ShareAlike 3.0 Unported License](#).

Digital Circuits – Lab #2



Observations

Half Adder Truth Table

A	B	Carry	Sum
0	0		
0	1		
1	0		
1	1		

Full Adder Truth Table:

A	B	Carry -in	Carry -Out	Sum-Out
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Questions

Please answer the following questions in the discussion part of your lab:

1. Discuss how we use the inputs to the full adder. In class we showed for example how when we add one column, the resulting carry goes over to the next column. With such an example show where the inputs/outputs to the full adder come from/go to.
2. Perform the same discussion as in #1 but with the full subtractor. Again you may find it helpful to write a simple example (e.g.: something like $10_2 - 01_2$) and show how the A,B, Borrow In, Difference Out, and Borrow Out inputs/outputs are used for each bit.



CC BY-SA

This work by [Colin O'Flynn](#) is licensed under a [Creative Commons Attribution-ShareAlike 3.0 Unported License](#).

Part #2: Xilinx CPLD Blocks - Physical Implementation

Objective

- Learn about addition blocks provided by the CPLDs
- Implement chained adders

Background & Procedure

Required Materials

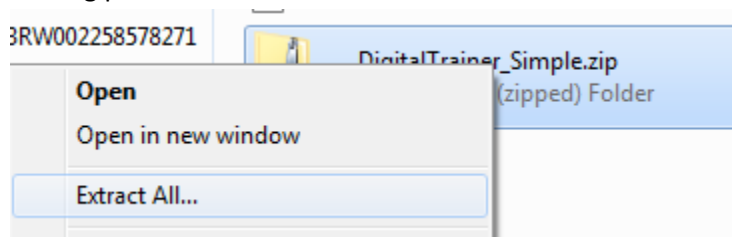
- Computer with Xilinx ISE 13.2 Webpack installed.
 - All computers in the lab have this installed.
 - This is free software so you can install on your own computer if you wish, you can download it from <http://www.xilinx.com/support/download/index.htm> - select '13.2' on the side. The file is very large so you may wish to download at school, and you are required to register to license it.
- Example project file DigitalTrainer_Simple.zip
 - This file contains an environment similar to the digital trainer you will use for testing hardware with. Note if you have already performed the simulation portion (section 1) you may keep that same file open.
- Digital Trainer Board

Procedure

ADSU1 Component

Xilinx provides an 'ADSU1' part, which is available under the 'Arithmetic' symbol library. You will place this part and explore what it does.

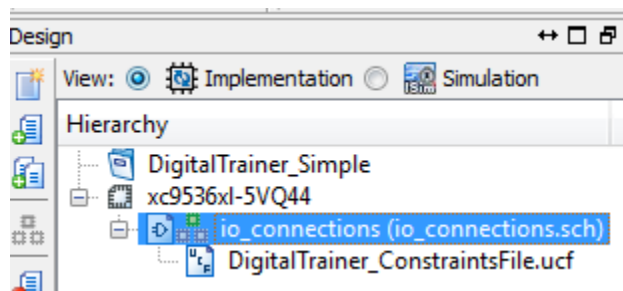
1. Unpack the given ZIP file somewhere. If using lab computers suggested to save to your personal drive or save to USB stick. Rename the folder something memorable, as you may save time by reusing parts of this lab later.



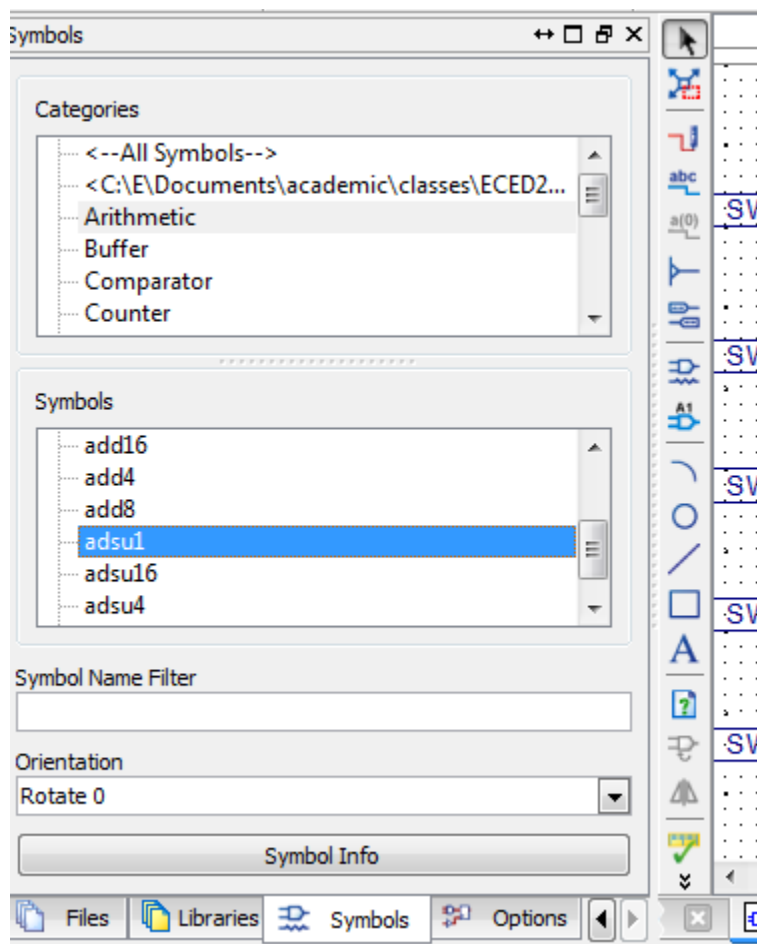
2. Open the resulting folder (it may open automatically), double-click on "DigitalTrainer_Simple.xise" which will open the Xilinx Project Navigator
3. From the top left menu select "Implementation" instead of "Simulation" as the view:



Digital Circuits – Lab #2



4. Double-click on the 'io_connections' file.
5. Place the 'adsu1' part under the 'Arithmetic' category:



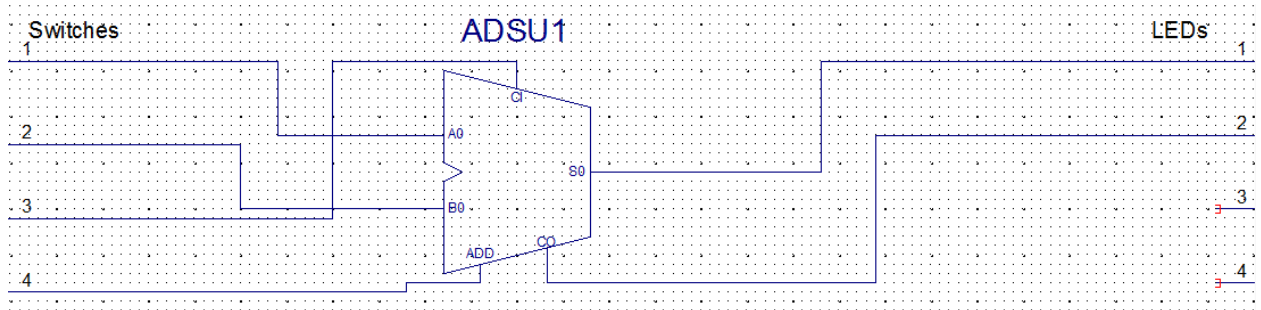
6. Wire the inputs to the switches, the two outputs to the LEDs. You may wire it up any way you wish, but remember which input goes to which SW/LED. Here is one example:



CC BY-SA

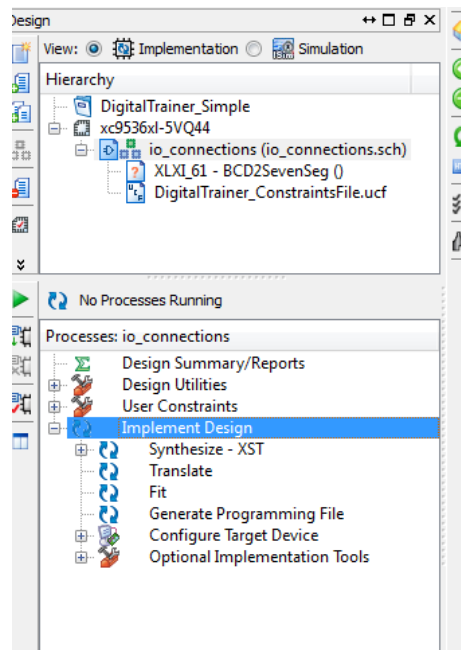
This work by [Colin O'Flynn](#) is licensed under a [Creative Commons Attribution-ShareAlike 3.0 Unported License](#).

Digital Circuits – Lab #2



The mapping used here is: SW1= A0, SW2 = B0, SW3 = C0, SW4 = ADD. LED1 = S0, LED2 = C0.

7. Implement the design in the board:
 - a. Ensure the 'Implementation' view is selected, and select file io_connections. Then double-click on 'Implement Design':



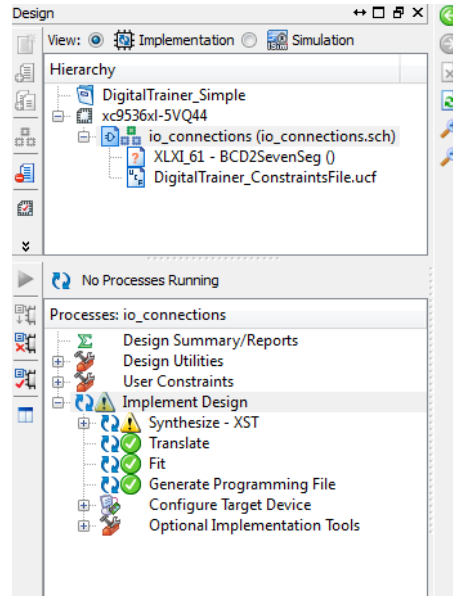
- b. The implementation phase should run. Afterwards your view will look like this, you must have a Green checkmark beside 'Generate Programming File':



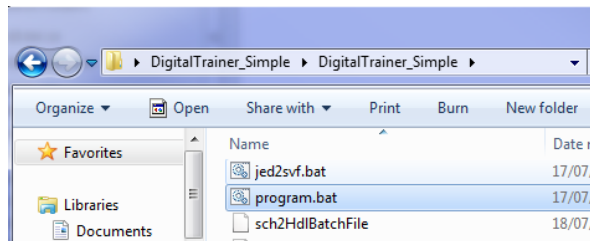
CC BY-SA

This work by [Colin O'Flynn](#) is licensed under a [Creative Commons Attribution-ShareAlike 3.0 Unported License](#).

Digital Circuits – Lab #2



- c. Plug in the Digital Explorer board. The power LED should light briefly and the Activity LED should blink on then off. **If the activity LED stays on unplug & replug the board.**
- d. Open the folder you created in step 2, the same one with the .xise file in it. Find the file called either **program** or **program.bat** and double-click it:



- e. This will open a command-line window, and take a little bit (~30-60 seconds) to finish:

```
C:\Windows\system32\cmd.exe

'1': Erasing device...
'1': Erasure completed successfully.
Elapsed time = 0 sec.

'1': Programming device...
done.
'1': Putting device in ISP mode...done.
'1': Putting device in ISP mode...done.
'1': Programming completed successfully.
Elapsed time = 0 sec.
Could Not Find C:\Users\Neolin\Desktop\DigitalTrainer_Simple\DigitalTrainer_Simpl
OPENDOUS reading command 0x01 failed (-116)
OPENDOUS reading command 0x02 failed (-116)
IR length: 8
Chain length: 1
Device Id: 0101100101100000010000010010011 <0x0000000059602093>
Manufacturer: Xilinx
Part(0): XC9536XL_UQ44
Stepping: 0
Filename: data/xilinx/xc9536x1_vq44/xc9536x1_vq44
```

At the end it will say “press any key to continue...”. If this happens immediately something is wrong, probably drivers were not loaded for the Digital Explorer board.



CC BY-SA

This work by [Colin O'Flynn](#) is licensed under a [Creative Commons Attribution-ShareAlike 3.0 Unported License](#).

Digital Circuits – Lab #2

```
Manufacturer: Xilinx
Part(0):      XC9536XL_UQ44
Stepping:    0
Filename:    data/xilinx/xc9536xl_uq44/xc9536xl_uq44
Parsing 1950/1950 (100%)
Scanned device output matched expected TDO values.
'#Pause' is not recognized as an internal or external command,
operable program or batch file.
Press any key to continue . . .
```

During this time your 'activity' LED should be on as well.

- f. Once it is downloaded, ensure the 'CPLD Reset' switch is set to '0' (down towards bottom).
 - g. Now you can control the inputs & watch the outputs on the LED.
8. Vary the inputs and record the outputs in the truth table. You must remember what 'SW1' corresponds to if you used a different mapping when recording your truth table!
 9. As the same suggests, this part can perform either an ADDITION or SUBTRACTION based on certain inputs. What input controls the mode, and what is the function of the remaining inputs/outputs for the different modes?

Observations

Steps 8 & 9 of the procedure include the observations you should include.



CC BY-SA

This work by [Colin O'Flynn](#) is licensed under a [Creative Commons Attribution-ShareAlike 3.0 Unported License](#).

Part #3: A 4-Bit Adder

Objective

- Design your own 4-bit adder

Required Materials

- Computer with Xilinx ISE 13.2 Webpack installed.
 - All computers in the lab have this installed.
 - This is free software so you can install on your own computer if you wish, you can download it from <http://www.xilinx.com/support/download/index.htm> - select '13.2' on the side. The file is very large so you may wish to download at school, and you are required to register to license it.
- Example project file DigitalTrainer_Simple.zip
 - This file contains an environment similar to the digital trainer you will use for testing hardware with.

Procedure

For this portion, you will be required to build a 4-bit adder.

SW1 – SW4 = Input Number (M). SW1 is the Least Significant Bit (LSB), SW4 is the Most Significant Bit (MSB). The following table gives a few examples to show you this interpretation:

SW1	SW2	SW3	SW4	Decimal (M)
0	0	0	0	0
1	0	0	0	1
0	0	0	1	8

Five of the LEDs will be used for the output R, LED1-LED5. LED1 is the LSB, LED5 is the MSB. Again the following table shows some examples of the LED binary to decimal interpretation:

LED1	LED2	LED3	LED4	LED5	Decimal (R)
0	0	0	0	0	0
1	0	0	0	0	1
0	0	0	1	0	8

The circuit must perform the operation $R \text{ (output)} = M \text{ (input)} + P$, where P is a constant. The value of P is the right-most number of your banner ID (student number). For example if your banner number is B00123456, you would be adding $P=6$ to the input. The following provides additional guidance:

1. You may use any components in the library, including ones you created in the 'simulation' portion of this lab. There are many different ways to solve this problem: you could chain the

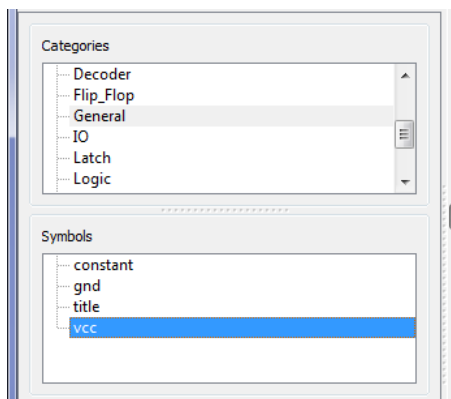


Digital Circuits – Lab #2

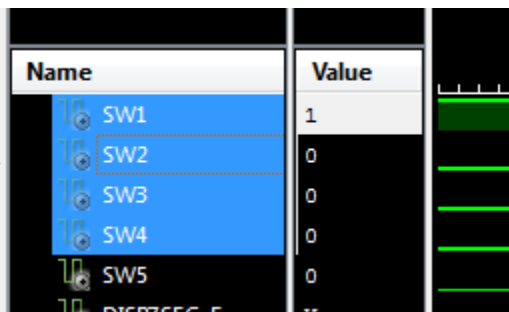
ADSU1 components, chain the ADD1 components, chain the full adders you build in part 1, or even use a larger logic block. A detailed description of the logic blocks available for your schematic is given at

http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_2/cpld_all_scm.pdf .

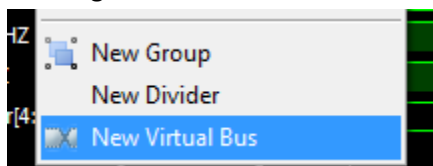
2. Be sure to record the P number you used in the observations. I will confirm the selected P matches your banner ID as expected.
3. When inputting the binary constant, you may find it useful to use the 'GND' and 'VCC' components to tie a specific pint to 0 or 1 respectively. They are found in the 'general' category:



4. When simulating the design, it will be useful to have a 'Virtual Bus' so you do not need to decode the binary values. To do this, first select the lines you want together by holding down the 'CTRL' button while clicking the lines:



Then right-click and select 'New Virtual Bus':



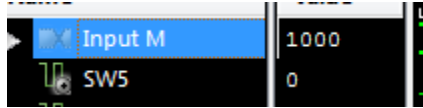
Give the bus a name:



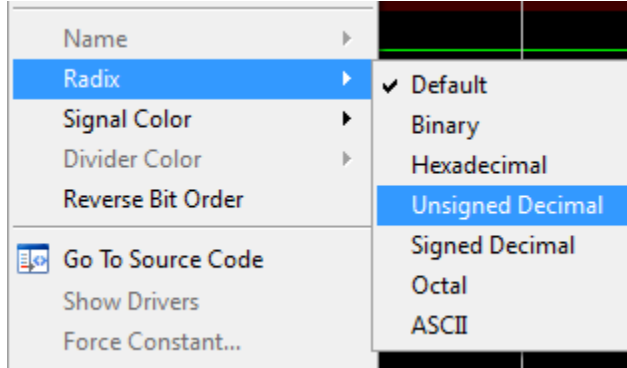
CC BY-SA

This work by [Colin O'Flynn](#) is licensed under a [Creative Commons Attribution-ShareAlike 3.0 Unported License](#) .

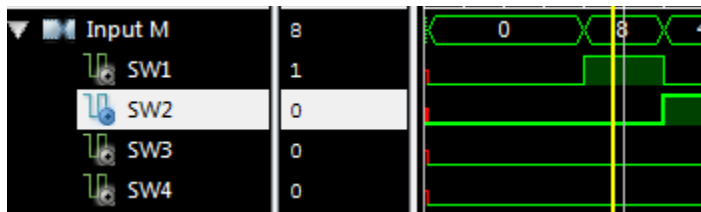
Digital Circuits – Lab #2



Then right-click on that name, and select 'Radix->Unsigned Decimal':



Check the bit order (MSB/LSB) is correct. To do this expand the bus, and compare the resulting bit order with the 'expected' bit. So for example here we see SW1 is '1' and SW2,SW3,SW4 is '0'. The result is expected to be decimal 1, but is instead decimal 8:



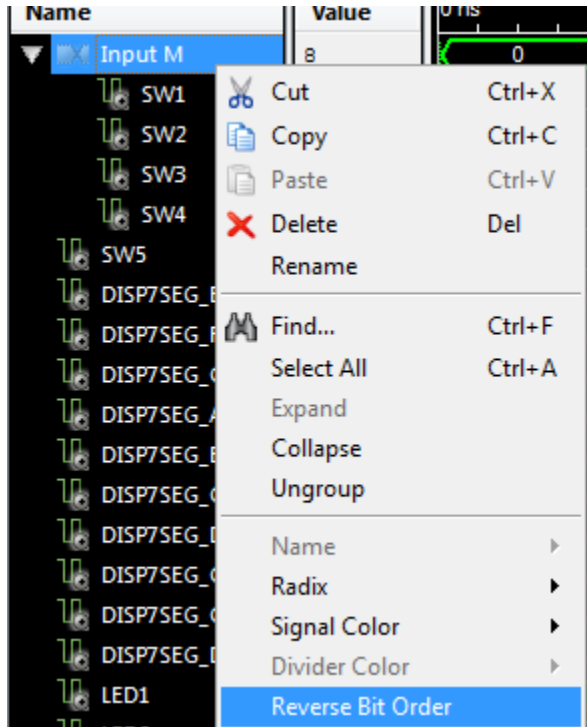
So we right-click on it again, and select 'Reverse Bit Order', which brings the result to the expected decimal 1:



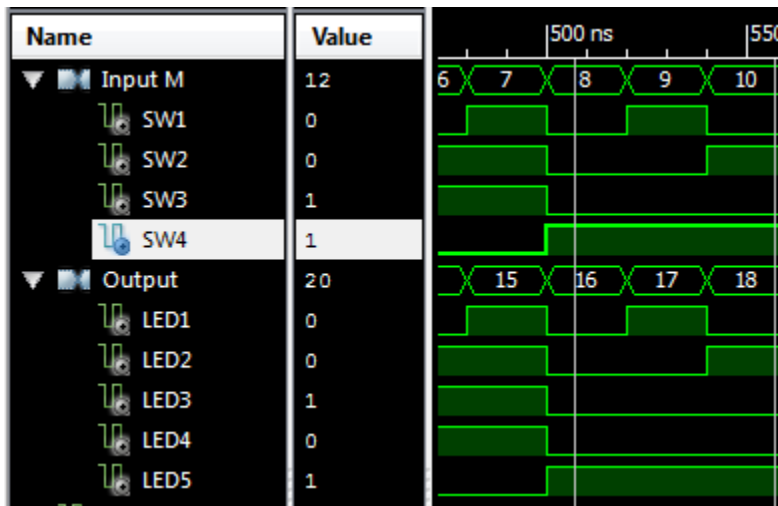
CC BY-SA

This work by [Colin O'Flynn](#) is licensed under a [Creative Commons Attribution-ShareAlike 3.0 Unported License](#).

Digital Circuits – Lab #2



5. You can perform the same virtual bus on the output lines. Again verify if the bit order is as expected. For example if my Banner ID ended in '8', the following would be the expected input & output:



6. Be sure to include the schematic you implemented (you can use the 'printscreen' button), and also include an image of the input and output waveform for the entire input count sequence (e.g.: 0,1,2,3,...,14,15,16).



CC BY-SA

This work by [Colin O'Flynn](#) is licensed under a [Creative Commons Attribution-ShareAlike 3.0 Unported License](#).

Digital Circuits – Lab #2

Observations

Include the schematic of your implementation and input/output waveform showing the input is the expected value (e.g.: $R=M+P$).

Include a few sentences explaining how you choose the design. For example was the design based on class notes, class slides, an external website, or some other reference? Did it work first time or were there problems?



CC BY-SA

This work by [Colin O'Flynn](#) is licensed under a [Creative Commons Attribution-ShareAlike 3.0 Unported License](#).