ECE451: VLSI Systems and Design
Lab 1: Adder Design
Due: January 19, 2015 (before lecture)

1 Introduction

In this lab, you will first build individual logic gate standard cells, and then combine the cells to form a half-adder, a full-adder, and an 8-bit adder. It demonstrates how careful design and layout techniques can greatly help semi-custom or standard cell designs. For example, you should design each cell such that it interfaces easily to the other cells, i.e., the pitch (height) of each cell should be matched, etc. To help you start on your design, a floorplan is provided to show how high-level modules are interconnected. You may come up with your own floorplan but in that case, you must document any difference. It is a good practice to plan your design using a top-down approach. You should first draw out a high-level floorplan for the 8-bit adder (see Figure 4), and then work on the floorplans for smaller components. The floorplan will help you decide ahead where the input and output ports of the smaller cells should be located and how they interconnect together. Also please check FAQ.pdf file, which under Course Material on portal; especially please refer to Layout section.

This lab is not a part of the upcoming 4-bit processor.

2 Basic cell design, verification, and layout

You need to design and layout the following basic CMOS gates:

NAND **NOR** **INV** **XOR2**

2.1 Design

For each of the cells, do the following:

1. Create a schematic in SUE. You have to determine the size of NMOS and PMOS transistors. For the purpose of this lab, performance is not important, but the transistor sizes should follow from what you've learned previously in other classes.
2. Create an icon for the schematic.
3. Create a test bench within SUE using the icon of the cell, connect it up with the Counter icon (available as `Counter.sue` on the web page).

You should try to design the XOR2 gate with 10 transistors. For example, you can make $\overline{\mathrm{A} \vee \mathrm{B}}$ with 4 transistors and feed A, B and $\overline{\mathrm{A} \vee \mathrm{B}}$ into a network of 3 NMOS and 3 PMOS transistors to generate the XOR2. Avoid 6-transistor XOR2 designs, since they rely on pass-transistor logic, which can be difficult to simulate and usually results in a slower circuit. Name the cell "XOR2" to avoid complications with the built-in XOR symbol in SUE.

2.2 Verification

For each of the cells, do the following:

1. Generate a SPICE netlist on the test bench in SUE.
2. Use NST to simulate the cell for all possible input combinations.
3. Print out and annotate the plots.
4. Generate a sim netlist on the cell (not the test bench) in SUE.
5. Use IRSIM to simulate the cell by creating the following IRSIM script. To learn more about IRSIM commands, see the tutorial in the Lab 0 handout.

```
| nand.cmd - testing nand
| testcase :
| inputs | outputs
| A B    | C
| 0 0    | 1
| 0 1    | 1
| 1 0    | 1
| 1 1    | 0
vector INPUT In1 In2
ana In1 In2 Out
| set time step
stepsize 100
| 0 0 | 1
setvector INPUT 00
s
| 0 1 | 1
setvector INPUT 01
s
| 1 0 | 1
setvector INPUT 10
s
| 1 1 | 0
setvector INPUT 11
s
```

6. Print out the simulation in Analyzer.

2.3 Layout

For each of the cells, do the following in MAX:

1. Name the design using the same name as schematics produced in SUE.
2. Layout the cells as described below. Make the layout as compact as possible.
3. Do **LVS** to compare between schematic and layout, by selecting *Tool →SUE LVS*. If the layout matches the schematic, you should see "LVS is CLEAN!" in the terminal from which MAX was started.
4. Print out the layout.

Here are some guidelines on the layout:

1. DO NOT use the 0.18um technology files (`mmi18`). Instead, use the default 0.24um technology files (`mmi25`) by starting MAX without the `-tech` option.

2. Build transistors using the *fet* from the library.

3. DO NOT use *Tool →Layout Generator*. The layouts produced by this tool are inefficient.

4. **You must provide body contacts as well as all other layers necessary.** For n-well contacts (around PMOS's), use the following layers (from bottom to top): *nw* (n-well), *nplus*, and *nwc* (n-well contact). The n-well must surround all the PMOS's and some of the area around it. Each layer should be smaller than the layer below it. Once you have the *nwc* layer, connect it to VDD. Make the contact from *nwc* to metal in the same way you make contacts from poly to metal. Similarly, for p-well contacts (around NMOS's), use layers *pplus*, and *pwc*. Note that no extra layer is needed for p-well as it is the substrate. Connect the *pwc* layer to GND.

5. Your cells should turn out much smaller if you keep all the N-diffusion near the GND line and the P-diffusion near the VDD line. This is because of the need for body contacts, namely VDD body contacts for PMOS devices and GND body contacts for NMOS devices. You should have a VDD bus running across the top and a GND bus along the bottom of each cell. In general, the inputs should be on the left and top sides of the cells and the outputs on the right and bottom. Make all the cells the same height so that the VDD and GND lines are continuous when two cells are placed side by side (see Figure 1). This is called *pitch-matching*.

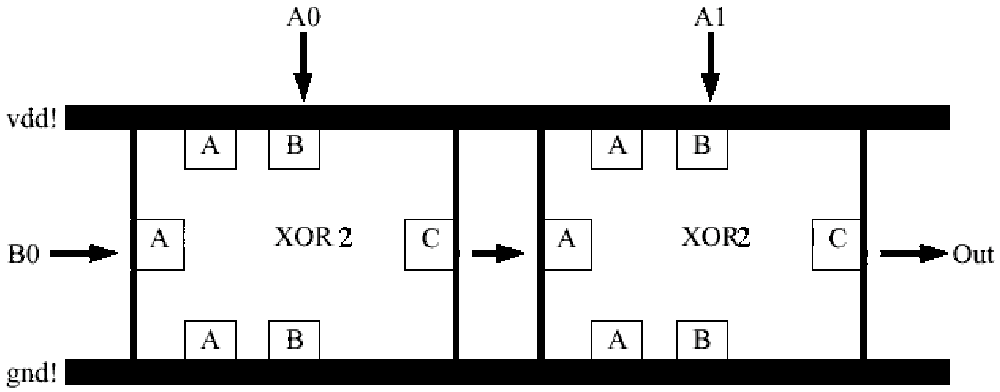


Figure 1: $Out = B0 \bigoplus A0 \bigoplus A1$. Note that inputs pass through the cells vertically, and horizontally, and that the inputs and outputs are aligned. So just by placing the two XOR2 cells side by side, the C output of the first XOR2 cell should be connected to the A input of the second XOR2 cell. There should be no need for extra routing.

   To match the pitch exactly for adjacent cells in MAX, you must ensure that objects that extend to the edges of the cell are aligned (e.g. vdd! and gnd! must have the same starting x location and width). Accomplish this by directly modifying properties (hotkey: *Ctrl-p*) of the drawing layer rectangles.

6. You should put some effort toward making your layouts compact; for example, when laying out the XOR2 cell, don't use your previous NOR as part of the XOR2 layout. By customizing your design and placing all NMOS or PMOS transistors within the same diffusion area, you can pack the XOR2 gate more tightly. Remember, good and compact sub-cells are essential for good and compact overall designs.

7. Make sure that the automatic DRC feature of MAX is enabled, as it can help you produce compact layouts by providing instant feedback on any design rule violations.

8. Be careful to match your layout exactly to your schematic. A common source of LVS errors is switching the inputs of two series transistors. While this may not matter logically, LVS will give you an error since the inputs of the layout do not exactly match the inputs of the schematic. It is a good idea to run LVS on every cell as it is completed. Waiting and running it on only the final cell may give so many errors that you cannot determine what the problem is.

3 Adder cell design, verification, and layout

Using the previously designed cells as sub-cells, build and test the following circuits:

HalfAdder **FullAdder** **EightBitAdder**

3.1 Design

Create schematics for the HalfAdder, FullAdder, and EightBitAdder using SUE. The HalfAdder should be created using an XOR2 and a NAND, the FullAdder using two HalfAdder's and a NAND, and the EightBitAdder using 8 FullAdder's. When using a cell in another cell, if needed, you can flip or rotate the sub-cell to make it fit a particular position.

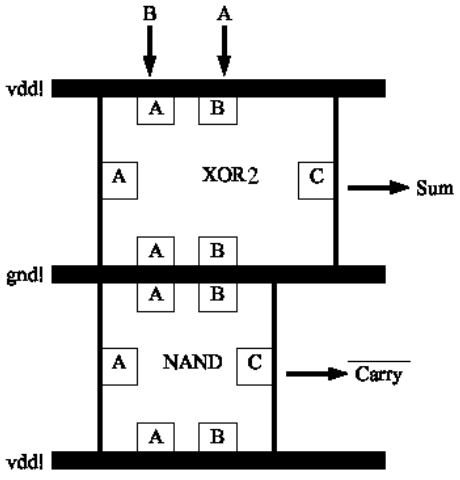
3.2 Verification

Simulate all the adders using IRSIM only.

3.3 Layout

3.3.1 HalfAdder

The HalfAdder is built from an XOR2 and a NAND. You should place an upside-down NAND below the XOR2 and move it close enough so that the XOR2 and NAND share the same ground line (see Figure 2). Note that you now have two vdd! lines: one at the top and one at the bottom. notCarry is generated (instead of Carry) in order to make the design of FullAdder more efficient.



(a) Floorplan.

A	B	Sum	$\overline{Carry}$
0	0	0	1
0	1	1	1
1	0	1	1
1	1	0	0

(b) Truth table.

Figure 2: HalfAdder.

The A and B inputs should feed into the top of the XOR2, run through the cell, and enter the top of the NAND. You may need to edit the NAND cell to align its inputs with respect to the XOR2. To edit a sub-cell in place, press *Shift-e* after selecting it. You may also press *v* to optimize the viewing of this sub-cell. To return to editing the HalfAdder, press *Ctrl-e*.

Flattening or not flattening? There are mainly two options while drawing a top-level layout containing sub-cells' layouts. The first option is not to flatten sub-cell's layouts. If you use this option the inherited net labels will be discarded by the top-level cell. So, you need to place and name new labels to the top-level layout. In the second option you can flatten all the sub-cells as follows: Select both XOR2 and NAND cells; go to *Misc →Flatten Cells* and click *Done*. After the flattening, all labels in sub-cells are exposed to the top-level hierarchy and are prefixed with sub-cell instance names (e.g., nand_91.In1). Now you need to rename the input and output labels and remove unnecessary internal labels. Sometimes it is useful to keep some of the internal labels to remind yourself what these signals represent. You must ensure that 1) the input and output labels of the top-level design are of the correct types (i.e., input/output), 2) all VDD (or GND) nets are "global" and have exactly the same name, and 3) all internal labels are "local". Label type conversion can be controlled *before* flattening, in *Misc →Flatten Cells →Flatten Setup ...*. However, sometimes you cannot avoid manual post-processing. Finally, you can always undo such flattening and label renaming by typing *u* as usual.

To decide flatten or not to flatten is your call as a designer. If you have very large circuits (not applicable to this lab), not flattening can help the LVS tool and also MAX itself to operate faster, and also you can analyze the top-level layout easier when sub-cells are defined. On the other hand, flattening give you the option to modify each sub-cell's layout to make your top-level layout a more compact one. Also, keep in mind that if you do NOT use flatten option in the top-level layout, and change the sub-cell layout, all of the top-level layouts containing that sub-cell will be affected (which can be both good and bad depending on your intentions).

3.3.2 FullAdder

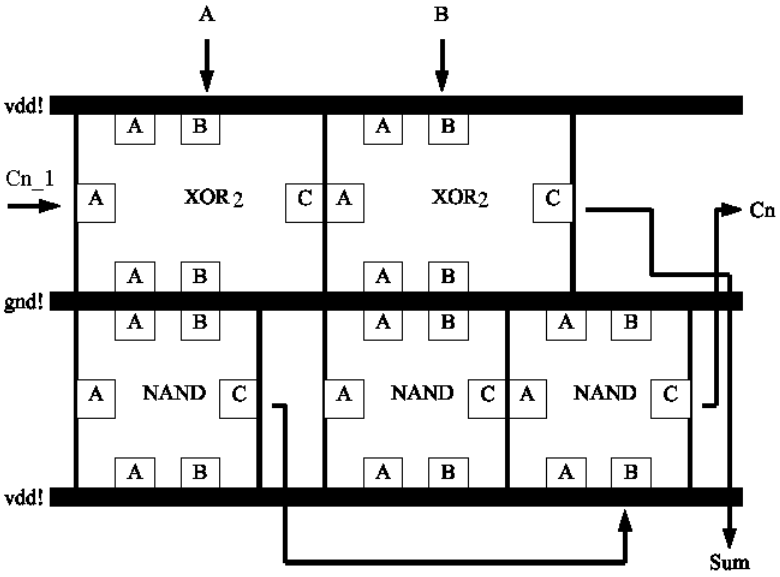
The FullAdder is made from two HalfAdder's and a NAND gate. It should add A, B, and Cn_1 ($Carry_{n-1}$) to give Sum and Cn ($Carry_n$). An example placement of blocks is given in Figure 3.

The notCarry output from the first HalfAdder must be routed to the NAND through or around the second HalfAdder. Two possible ways of doing this are by using metal2 or by running a wire around the bottom of the cell.

Your FullAdder cell should have its inputs and outputs routed so that it can be pitch-matched with other FullAdder's in the EightBitAdder. A and B should enter at the top of the cell and Sum should exit at the bottom. Cn_1 should enter from the left and Cn should exit to the right. Cn_1 and Cn should cross the borders of the cell at the same position and be of the same material. This allows multiple cells to be stacked next to each other, with Cn of one cell feeding Cn_1 of the next cell.

3.3.3 EightBitAdder

To make the EightBitAdder, stack 8 FullAdders together (see Figure 4).



(a) Floorplan.

Cn_1	A	B	Sum	Cn
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

(b) Truth table.

Figure 3: FullAdder.

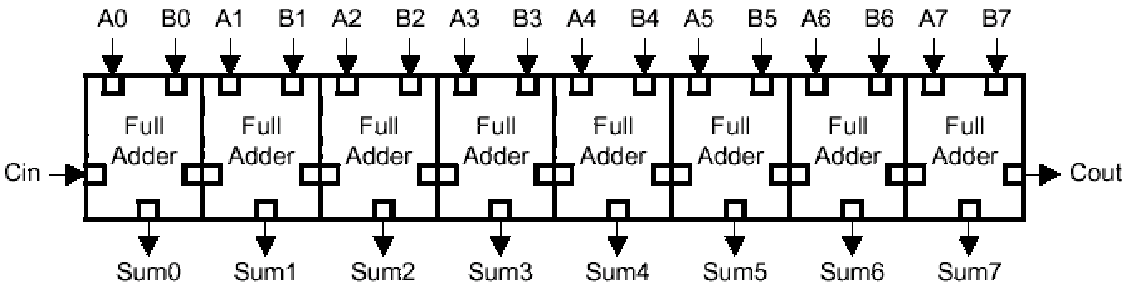


Figure 4: EightBitAdder.

4 Submission

The submission includes both electronic and paper parts.

4.1 Electronic submission

1. Schematics and layouts of all the cells (both Section 2 and 3). **Make sure you name the cells this way:** `INV NAND NOR XOR2 HalfAdder FullAdder EightBitAdder`
2. IRSIM test case `.cmd` files for the adders in Section 3. For the HalfAdder and FullAdder, do all possible test cases. For the EightBitAdder, do only this: A = 255, B = 0, Cin = 1 as inputs. **Make sure the files' base names are the same as the cell names listed previously.**

You will submit the above files with the following command:

```
submitece451s 1 EightBitAdder.sue EightBitAdder.max EightBitAdder.cmd ...
```

You can check your submission with the command `submitece451s -l 1`

4.2 Paper submission

You submit **one** report as a group. **Use the cover page on Blackboard and clearly indicate your names, student numbers and (a single) UG account where you do electronic submission.**

1. If you use an alternative floorplan to the one given in the figures, document it using figures similar to those in the lab and turn them in.

2. Schematic and layout print-outs of all the cells (both Section 2 and 3).

   Before printing in SUE, go to *File* →*Print Setup* ... to set the proper print command (e.g., `lpr -Plj251 -r %s`).

   Direct printing in MAX does not work properly. You have to take a screenshot of the layout and convert the resulting image to a PS/PDF file. Here is the detailed procedure:

   (a) In MAX, go to *File* →*User Preferences* →*Color Editor* ... and set the background to white. Without this step, the default grey background would make the printed layout illegible.

   (b) To take a screenshot, go to *Applications* →*Accessories* →*Take Screenshot* at the GNOME panel. In the pop-up window, use the option *Select area to grab* to avoid post-processing of the image.

   (c) Open the saved image in a viewer (e.g., `gthumb`) and print it to a PS/PDF file.

3. SPICE and IRSIM plots for cells in Section 2.

4. IRSIM plots for the adders in Section 3. Use the test case `.cmd` files you submit electronically.

Please carefully read Appendix A for general guidelines of annotating your layouts and simulations.

A Layout and Simulation Annotations

Every semester, students turn in layouts and simulations that have few or no annotations. It is not possible to grade a simulation or layout if it cannot be understood. Here are some guidelines of annotation:

- The labels on a layout printout must be legible. Otherwise, add the labels manually.
- Each test case in a simulation printout must be clearly identified. For each test case, you must indicate 1) the value each input takes, 2) the expected value of each output and 3) how it is calculated.

You will receive 0 points for simulations or layouts that we cannot understand. You don't need to give long explanations, but be sure to be clear. Here is a sample of annotations we expect to see:

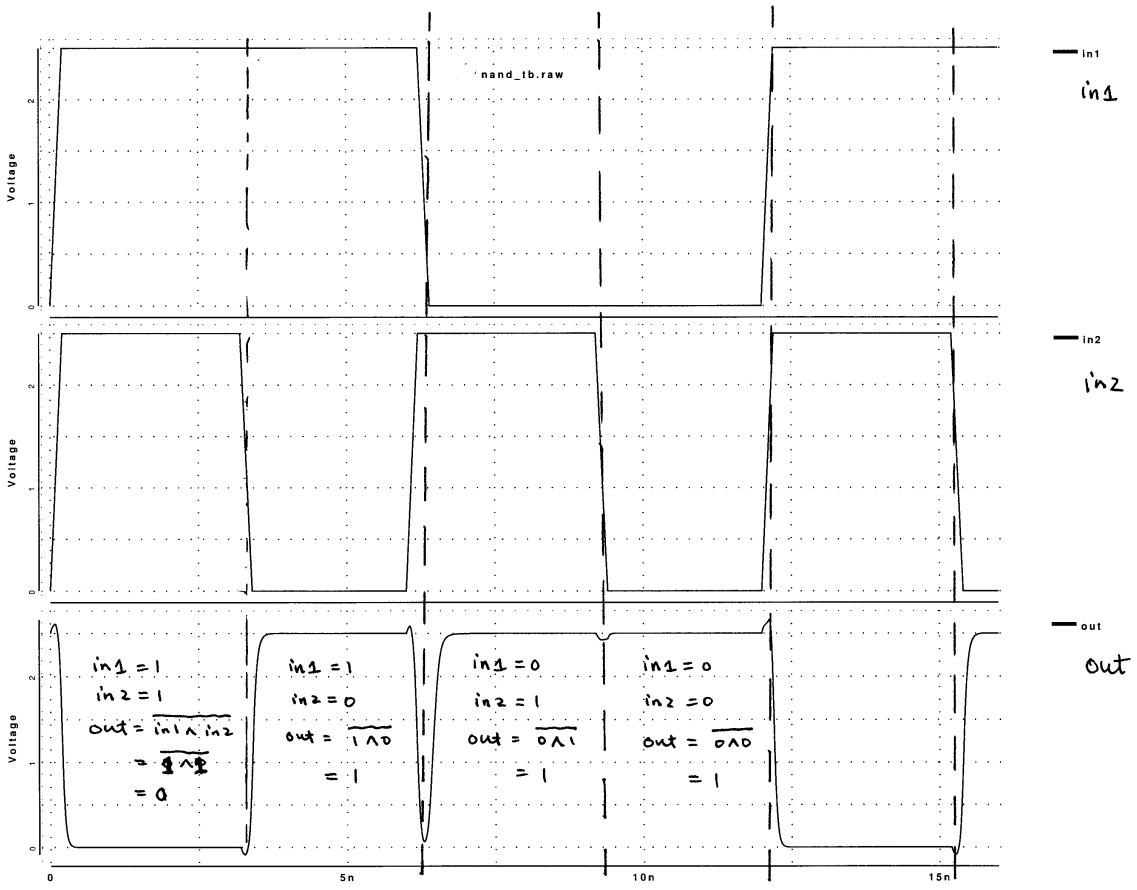


Figure 5: Sample annotations for a SPICE simulation.