ECE425: Introduction to VLSI System Design
Machine Problem 3
Due: 11:59pm Wednesday, May 11th, 2021

In this MP, you will use automated tools to synthesize the controller module from your MP2 project into a network of standard cells, place those cells onto a layout, and route the cells together into a complete implementation.

You will see the differences between manual and automatic circuit implementation. The datapath you created in MP2 optimizes every transistor for optimum efficiency within the well-defined and detailed constraints of the datapath topology and the TSMC process design rules. As you translated the logic specifications to schematics, you could apply your full knowledge of logic optimization to improve the overall design. However, our automated tools cannot adapt to the specific task of designing an Am2901, as they are designed to solve the most general problem of circuit design. Here is a comparison:

Task	Human engineer	Design automation tools
Logic optimization	Full knowledge of Boolean Algebra, plus you can go back and change the overall topology.	Incremental optimization based on initial specification + "Guess and check." Final results based on initial approximation.
Cell design	You know every instance of how every cell will be used, and design for one process.	Our MOSIS-based cell library is generalized for many manufacturers and for ease of use to tools.
Integration & routing	Ability to discover physical realities, learn, and revise.	Designed for speed and generality. Placement is a best guess; routing cannot modify placement.

A purely automated approach cannot approach human thoroughness, critical thought, and ingenuity. But it is much less work.

Part I. Synthesis

We will use Synopsys' Design Vision package to perform logic synthesis.

Initialize

1. Execute below command in order.

"`module load ece425-sp21`"

"`module load Synopsys_x86-64`"

This will load the Cadence and Synopsys modules and change into your MP directory (*~/ece425.work*).

2. Extract the starter kit into your work directory: run "`tar xvf /class/ece425/ece425mp3_fall17.tar`".
3. Execute the tool by typing "*`design_vision`*" in your work directory. Do not run it in the background with "&".
4. Select *File => Read…* and open your *controller.v* file (from mp2).
5. Choose *File => Execute Script…* and double-click *tsmc250target.dc*.

Set constraints

Most of the controller should not be critical to the timing of the Am2901, but the decoder generating the *select_a* and *select_b* signals must be fast. We will try synthesizing at a couple different speeds to see the effect of changing the desired performance.

1. Select the *controller* module in the leftmost part of the window under "*Logical Hierarchy*".
2. Select *Pins/Ports* from the scroll-down menu of the right window.
3. Click and drag over all the *select_a* and *select_b* signals, as well as a[3:0] and b[3:0], to highlight all the pins on the controller associated with the decoder.
4. Choose *Attributes => Optimization Constraints => Timing Constraints*.
5. Enter 275 for *Max rise* and *Max fall*. (The units are picoseconds.) Leave the minimum constraints blank.
6. Under *Group name*, enter *decoder*. Click OK.
7. Go back and highlight only the *select_a/b* signals.
8. Select *Attributes => Operating Environment => Load….*
9. Set *Capacitive load* to 5. (The units are picofarads.) If you included inverters in your *regbit* cell and provided only one *select_a_hi* and one *select_b_hi* input, use 10 instead. Click OK.

Synthesis

Choose *Design => Compile Ultra…* and click OK, and your design is synthesized! You can see the resulting schematic by right-clicking *controller* in the hierarchy section and selecting Schematic View. Identify the register file decoder which generates the select_* inputs. **Report the area you got in this setting**. If you cannot find the area, select *Design => Report Area…*, and record the *Total cell area*.

Now, we will try loosening the timing budget. Select the select_*, a[3:0], and b[3:0] signals again, and open the timing constraints window. Change the max rise and fall constraints to 325 picoseconds and click OK. Redo the previous paragraph, recording the same values. **Report the area you got in this setting.** To avoid checking the properties of every cell, you may assume all the cells on a given level with the same symbol are the same. The final number in the cell name specifies drive strength, which is unlikely to vary among cells on the same level.

Export the design

Choose *File => Save as…* and select Verilog as the format. Name the file as *controller_synth.v*.

Part II. Place & Route

Most of the remaining work will be set under two constraints for the layout tool:
1. We want to place the controller above the datapath in the global floorplan.
2. The layout from MP2 exactly defines the bus of wires connecting the datapath and controller.

Define the place & route area

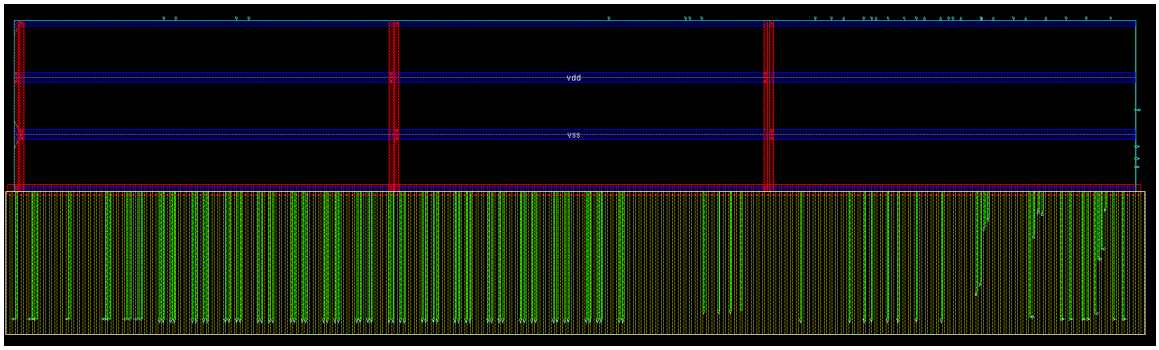
1. Launch *virtuoso* and open your datapath layout.
2. Choose *Create => P&R Objects => P&R Boundary*, draw a rectangle with similar width and height with your MP2 layout, which entirely covers your MP2 layout.
3. Calculate the necessary height of the P&R area:
   a. Start with the cell area from the second synthesis run, which you saved.
   b. Multiply this number by 1.2 to leave a margin.
   c. Divide by the width of the rectangle to find the desired height.
   d. Round up to the nearest multiple of 15.12, which is the cell height in our synthesis library.
4. Select the P&R boundary box, press "q" to pop out its properties window. Enter the four coordinates of the rectangle. For example, if the width of your MP2 layout is 400, and the calculated height of the P&R area is 30.24 (For most of your designs, the height of the P&R should be 30.24), the four coordinates are ***(0 0)(0 30.24) (400 30.24)(400 0)***
5. Choose the *Move* tool and set *Snap Mode* to *Orthogonal*. Move the rectangle up so that its bottom **touches all the pins** in your MP2 layout but **doesn't overlap anything else in the datapath**.

This layout will serve as a floorplan for the whole chip. We will save it in the industry-standard floorplan file format. From the CIW, choose *File => Export => DEF*. Specify the source library, cell, view names to your datapath layout. Set the file name to *am2901.def* and click OK. We are finished with virtuoso for now.

Load your design into Cadence Encounter

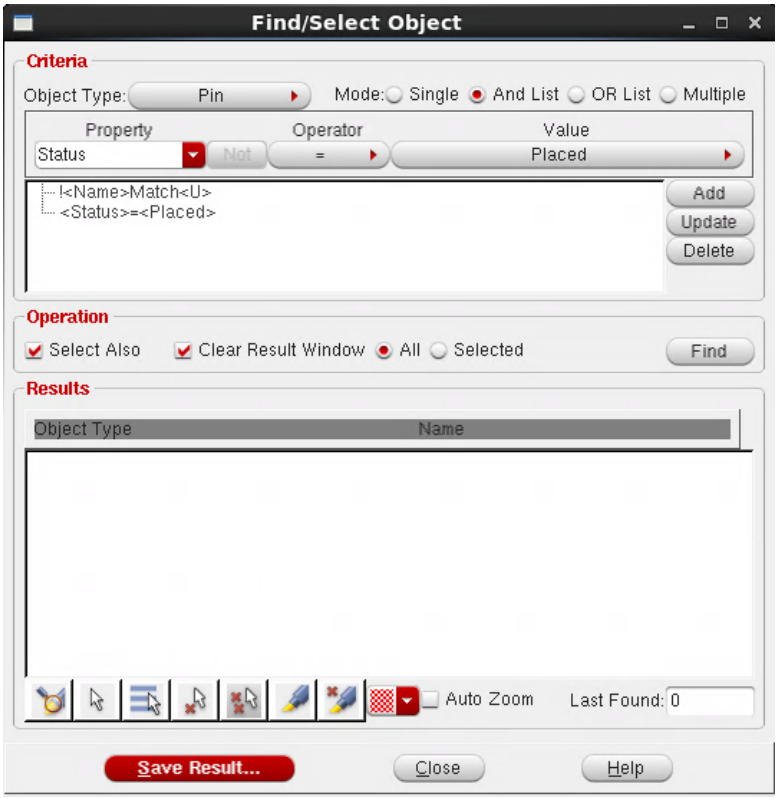
We will use Cadence Encounter to manage the place & route design flow. Launch it from a **new terminal** in your work directory by typing *encounter*. (Do not use an "&" sign to launch it in the background, because it requires an interactive command line.) Before the encounter command, don't forget to cd ece425.work and module load ece425-sp21.

1. Choose *File => Import Design*. Click *Load… ,* change files of type to *All Files (*)* and double-click *mp3.conf*. This tells Encounter about your *controller_synth.v* and the *VTVT* cell library. Click OK. Note that you need to modify *mp3.conf* if you used a different name for your verilog.
2. Choose *File => Load => DEF…* and double-click *am2901.def*. You should only see the P&R Boundary box and internal interface pins.
3. In the current floorplan, Encounter is still unaware that the bottom of the controller is a special interface. (Even though it's full of pins.)
4. In the encounter shell, execute: "setDrawView fplan". Alternatively, choose "Floorplan view" from the GUI toolbar (options tab). Most CAD tools provide a shell to execute commands. In fact, the GUI commands will translate to shell commands internally.
5. Choose *Floorplan => Edit Floorplan => Create Pin Blockage*, draw a narrow rectangle just covering the bottom edge of the controller. This is to prevent the P&R tool from placing pins around this edge.
6. Choose *Floorplan => Edit Floorplan => Create Routing Blockage*, draw a rectangle inside controller, then use the resize tool to make it cover all your pins in datapath. Ideally, the blockage touches the bottom of controller. However, you might need to leave a small gap between your blockage and controller (e.g., the top part of pins are kept unblocked) such that the router can have enough resource to access/connect to those pins. This is to prevent the router routes into datapath and mess up with your datapath design. A floorplan with pin/routing blockage added looks like the following:



7. To prevent the pins being changed by encounter, choose *Edit->Find/Select Object*. In the pop-up window, select Object Type as "Pin", Mode as "And List". Add two items:
    a. Property: status, operator "=", value: PLACED
    b. Property: name, check "NOT", operator: "Match", value: U

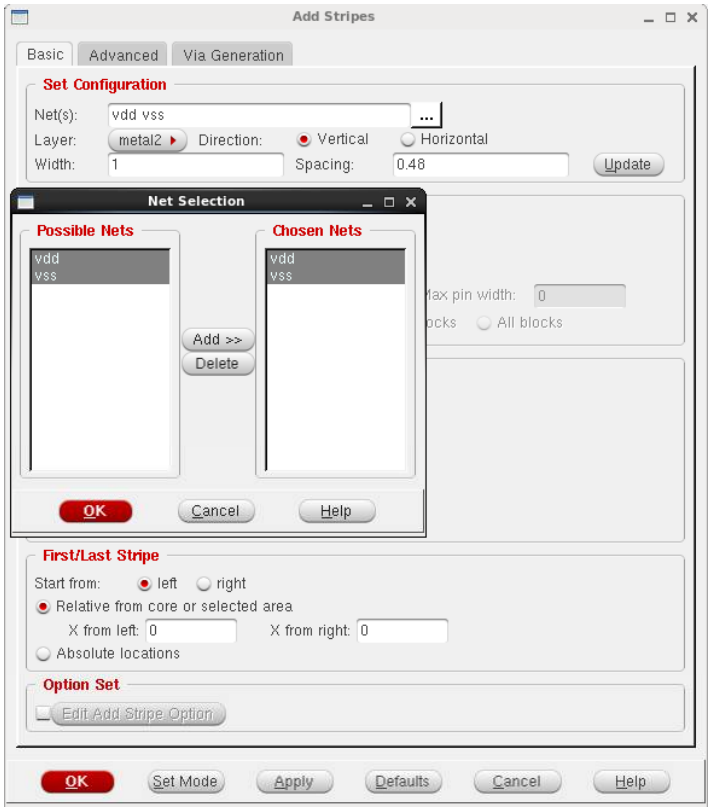
   After you added the above two items to the find criteria, click "Find". This will select all your control pins. Now, go back to the main window, click 'q' to open the property window. Check "Common", change the status from "PLACED" to "FIXED" to prevent them being changed. Select "Apply all" and "OK".



Add a power grid

Now there is a basic floorplan, but we still need to add "power rails". If we don't do this now, Encounter will be very hesitant to draw them on top of the already-placed cells!

1. Choose *Power => Power Planning => Add Stripe...* Browse the "*Net(s)"* and add *vdd* and *vss*
2. Set the width to 1 (the unit is microns). Leave Spacing at 0.48. Click OK.



3. Choose *Route => Special Route*. Browse the "*Net(s)"* and add *vdd* and *vss.* Click OK.
4. Zoom into the intersection of the horizontal and vertical metal and check that there are vias connecting the layers of the grid.

Place and route

Choose *Place => Standard Cells* and click OK. Choose *Route => NanoRoute=> Route* and click OK again. Check the console log and make sure there is no DRC violations. You can switch to "Physical View" to see result of automatic Place & Route. Wasn't that easy?

It's time to record results. Choose *File => Report=> Summary*. **Save and print the report to a file.** Don't worry about routing warnings, but your area of Effective Utilization should be below 1.

Actually, there is one more step. Choose *Place => Physical Cells => Add Filler…* . Click *Select,* choose *filler, Add,* then *Close,* and *OK*. This connects the *nwells* empty gaps across between cells. (Incidentally, these filler cells may overlap, resulting in area usage greater than one.)

Export back to virtuoso

We will integrate the controller with the datapath in Virtuoso.

1. Save the generated layout in the industry-standard GDSII format by choosing *File => Save => GDS*. Set *Output Stream File* to *controller.gds,* and *Map File* to

vtvt/vtvt_tsmc250_lef/vtvt_SocE2df2.map. Click *OK*.

2. In virtuoso CIW, choose *File => Import => Stream…* . Set *Stream File* to *controller.gds*, and *Library* to *ece425mp3*.
3. In Layer Map, set the same map file as before*: vtvt/vtvt_tsmc250_lef/vtvt_SocE2df2.map*.

4. Click *Apply* to import the synthesized layout.

Create a new layout cell *am2901* and place an *ece425mp3/controller* and an *ece425mp2/datapath* inside it. You should click the origin of the canvas to place both instances.

Generate a printout showing your overall integration and measure the height and width using the ruler to generate your total layout area. In the LSW, click *NV* and then center-click the metal3 and metal4 *drw* and *net* layers. Make sure the connections between the controller and datapath are visible on the printed page.

Normally there would be additional verification at this point to double-check everything and extra steps to add the structures supporting the 2901's pins, but you've earned a break. You're done!

Congratulations, you are a qualified chip designer!

In case you couldn't see layout cells in controller:

1. Keep your ece425mp3 in your library, download another folder: ece425mp3oa (on course web page) in your ece425.work directory.
2. In Cadence CIW window, choose File -> New -> Library.
   Tap ece425mp3oa in the Library Name
   In Technology File section, select 'Attach to an existing technology cell library
   Apply ->NCSU_TechLib_tsmc03d -> OK
3. In virtuoso CIW, choose *File => Import => Stream…* . Set *Stream File* to *controller.gds*, and *Library* to *ece425mp3oa*.
4. In Layer Map, set the same map file as before*: vtvt/vtvt_tsmc250_lef/vtvt_SocE2df2.map*.
5. Click *Apply* to import the synthesized layout

MP3 Report (7.5 points):

1. **Synthesis (rise/fall constraints = 275 and 325 respectively):**
   - Record the area of the controller for each constraint settings.
   - Print the controller schematic generated by synthesis
2. **Place and route:**
   - Attach the summary report.
3. **Top level layout:**
   - Print top-level (complete AM2901 with controller and datapath) layout with Metal 3 only. (Include Metal 3 & 4 if metal 4 was used in MP3)
   - Use ruler to measure the height and width, and **calculate the total layout area**. You **MUST** write down clearly your total area on the first page of your MP3 report.
4. Please also submit all your design files in a zipped folder (such that anyone can evaluate your work in cadence).

In addition to the extra credit for area optimization for the design, you will have another Extra credit opportunity.

Extra Credit (3 points in overall grade):

In MP2 and MP3, you have designed a microprocessor layout in cadence, including the datapath and the controller. However, this layout did not have a pad frame to read out voltages or provide power supply to chip components. Therefore, this layout, even though complete, cannot be sent out for fabrication. In this extra credit opportunity, you will add a pad frame to the microprocessor layout, so that the chip components could have connections off chip to facilitate its fabrication and use it as a system component. Also, you will use what you've learned from the HSPICE lecture to measure the power consumption of your microprocessor.

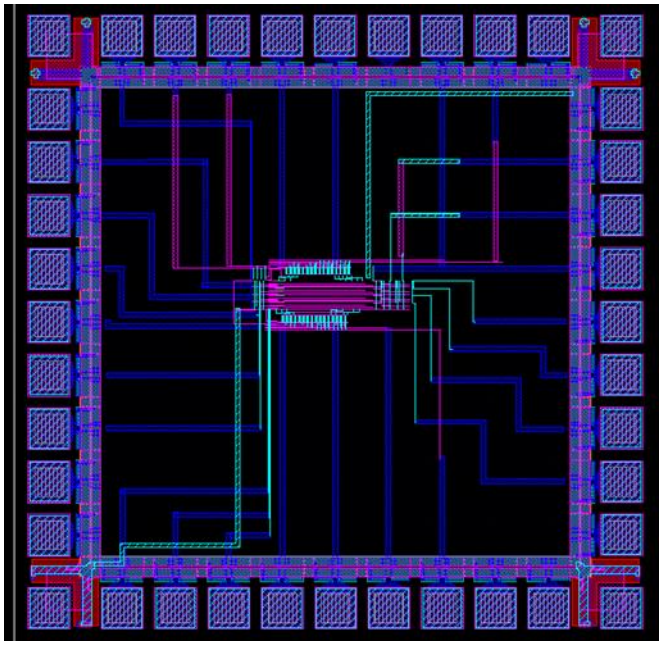


Figure 1: an example of the layout with padframe

PART I: Pad Frame

1. Download the top.gds file (on course web page) in your ece425.work directory
2. Open Cadence Virtuoso, in the CIW window, create a ece425mp3top library:
   In CIW window, choose File -> New -> Library
   In the New Library Window, attach to an existing technology cell library
      tap ece425mp3top as Library Name;
      in Technology File section, select 'Attach to an existing technology cell library'
      Apply -> NCSU_TechLib_tsmc03d -> OK
3. After creating a new library and attaching it to an existing technology, we can import top.gds file into this library.
   a. In virtuoso CIW, choose File => Import => Stream... . Set Stream File to top.gds, and Library to ece425mp3top.
   b. In Layer Map, set the same map file as before: Anu19.map.
   c. Apply.
4. Now, your GDS file has been imported into virtuoso and you should be able to open the entire design and specifically, the cell named padframe in ece425mp3top library.
5. Place the cell padframe layout in our design and connect the padframe pins to your layout using metal wires (M1, M2 and M3).
6. Pins 5 and 15 of the padframe are Vdd, pins 25 and 35 are gnd. So, make connections accordingly.

PART II: Power Consumption Measurement

Low power design is a necessity today in all integrated circuits. Now let's see how much power our microprocessor would take. You'll use HSPICE here to do the measurement, as it is a robust industry standard.

1. In your ece425.work directory, run the following commands:

```
module load ece425-sp21
export CDK_DIR='/class/ece482/NCSU_CDK'
virtuoso &
```

2. Open the layout of your "*am2901*" (you should have the pad frame on it). To extract from the layout, choose *Verify => Extract,* and click OK.

3. Open the extracted view of am2901, click *Launch->ADE L*. Then *Simulation->Netlist->Create***.** Copy the netlist in the pop out window to a .cir file. Here it is **top.cir** .

4. Open the top.cir file in your ece425.work directory. There are several modifications we need to do with this file

(1) add **.lib '/class/ece482/models18' MOS** .This imports the library to your .cir file
(2) change transistor names to pmos and nmos

Example:

m0 y<3> n871 vdd! vdd! PMOS L=600e-9 W=4.05e-6 AD=6.075e-12 AS=3.645e-12 PD=7.05e-6 PS=1.8e-6 M=1

m2023 n92 f<2> 0 0 NMOS L=600e-9 W=1.95e-6 AD=1.755e-12 AS=2.925e-12 PD=1.8e-6 PS=4.95e-6 M=1

(3) provide simulation inputs; add the following lines to your netlist

```
vcp cp 0 pulse(0 5 0 0ms 0ms 5ms 10ms) *pulse voltage
vvdd vdd! 0 5 *DC voltage
vgnd gnd! 0 0 *DC voltage
...
vregwr reg_wr 0 0
*inputs part ends
.tran 1ns 20ms *add this command for transition simulation
.option post *generate plot file
.END *end of file
```

5. Run the following command to set up the 482 work environment

```
source /class/ece482/init_NCSU.sh
```

6. Move to 482 work directory, run the following command to do hspice simulation.

```
hspice top.cir > top.out
```

7. Run the following command to view simulation results. This will open scope.

```
scope &
```

8. In scope, *File->Open->Plotfiles* , change files of type to All Plotfiles(), then open *top.tr0*. Plot signals you want to watch in the pop out window.