

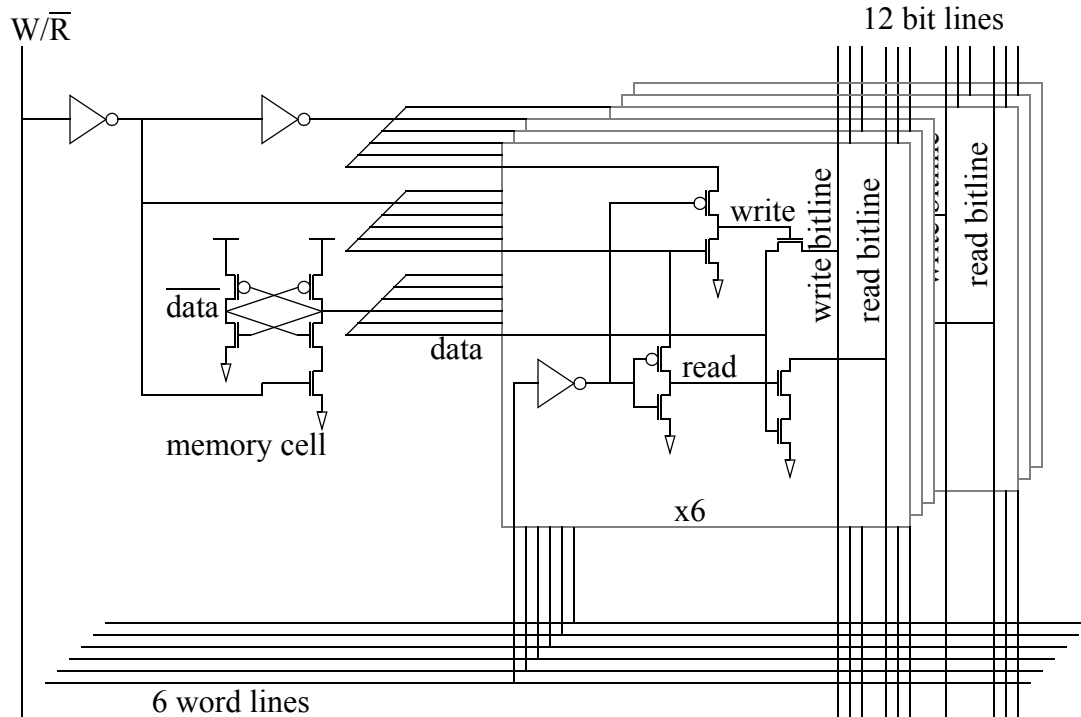
# ECEN 6263

## Fall 2007

### Design Project 1

Part A Due Fri., Sept. 21, 1700

Use the cadence tools to design a multi-port static RAM cell in an area less than  $20,000\lambda^2$  ( $1,800\mu\text{m}^2$ ) to be used as the building block for a larger register file. The register file cell should be capable of 6 simultaneous reads and 6 simultaneous writes in the same clock cycle. Also, the cell must have a 1V noise margin which is not possible with the standard static RAM memory cell. A suggested memory cell design which can meet these requirements is shown below.

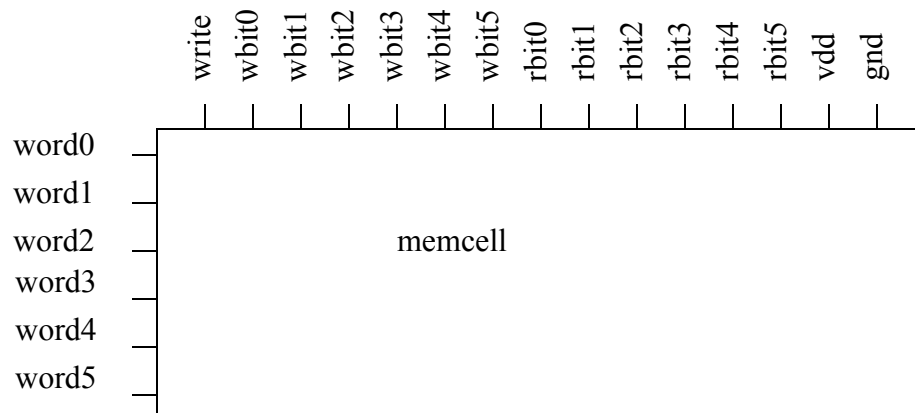


The memory cell has a single ended write, instead of the usual double ended write to reduce the number of transistors required for writing. If the memory cell transistors are all about minimum width,  $V_{inv}$  should be about  $V_{dd}/2$  which should give sufficient noise margin. All other transistors can be near minimum width also, but if there is any extra area, the write pass transistor and the read bitline driver can be made larger to improve access time.

There are 6 bit lines dedicated for writing and 6 bit lines dedicated to reading. This is necessary because the bit lines are too slow to do reading and writing in the same clock cycle.

However, the word lines can be time shared between reading and writing. When the  $\overline{W/R}$  signal is high during the first half of a clock cycle, the write address decoders are driving the word lines. A high on any of the word lines will cause a high on the corresponding internal write node which connects the memory cell to the write bitline. When the  $\overline{W/R}$  signal is low during the second half of a clock cycle, the read address decoders are driving the word lines. A high on any of the word lines will cause a high on the corresponding internal read node which connects the read bitline driver to the read bitline. The read bitlines are precharged high during the first half of a clock cycle.

The memory cell should have the following terminals.



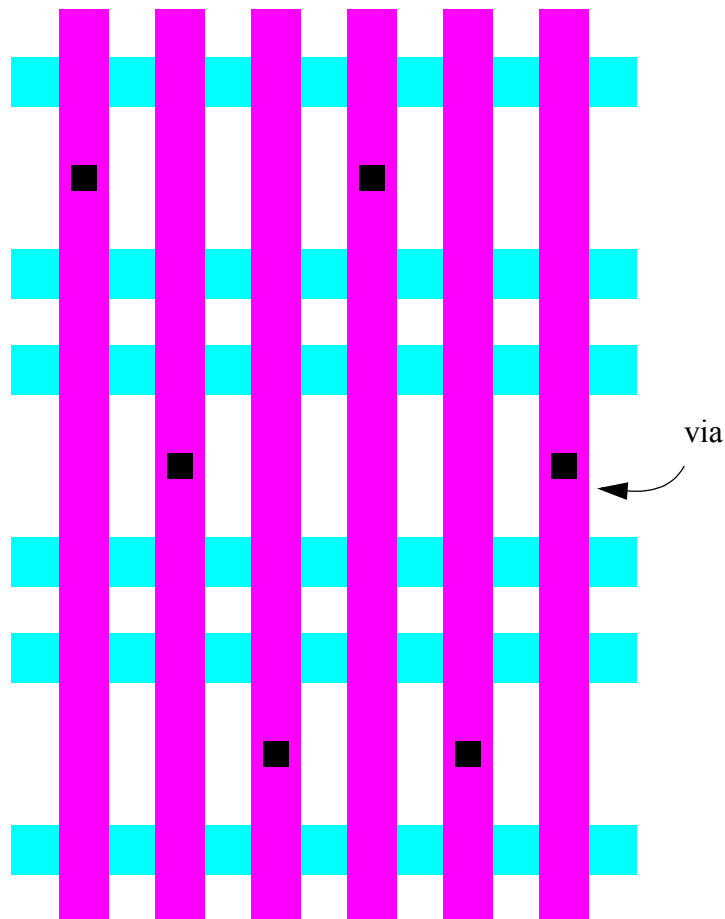
Important: It is also necessary to have terminals for the internal memory data storage nodes as “data” on the bit side and “databar” on the bit bar side. Otherwise, your design will not work correctly with the circuit simulator. Remember, terminals must be added in your highest level cell.

Both word lines and bit lines should be implemented in metal. These metal lines should run all the way across your memory cell so that word and bit lines in adjacent cells will be connected properly.

Meeting the area requirement requires careful planning of the word and bit lines. You should lay out all of the horizontal lines in one metal layer and the vertical lines in the other layer. You will find that the metal lines for the bit lines and the word lines takes up most of the allowed area. The idea is to lay the metal lines out first and then fit the transistors underneath. This is a good layout technique for circuits that have a high degree of interconnections per transistor.

You can use any of the metal layers metal1, metal2 and metal3. When planning your metal line layout, do not forget to allow space for the vias from metal2 and metal3 down through metal1 and metal2 into the underlying layers. This means that you can put the

metal 2 lines as close as possible, but you must allow extra space between the metal1 lines as shown below.



For your convenience, the file

`reg6x12.spin`

has been provided in the class directory (`/home/ljohn/public/ecen6263`) which contains the Spectre model and input commands to exercise your memory cell design.

Turn in you layout for grading. Use the following parameters with the automatic grader.

class name: `ecen6263`

project name: `proj1a`

project directory name: `reg6x12_proj`

project library name: `reg6x12_lib`

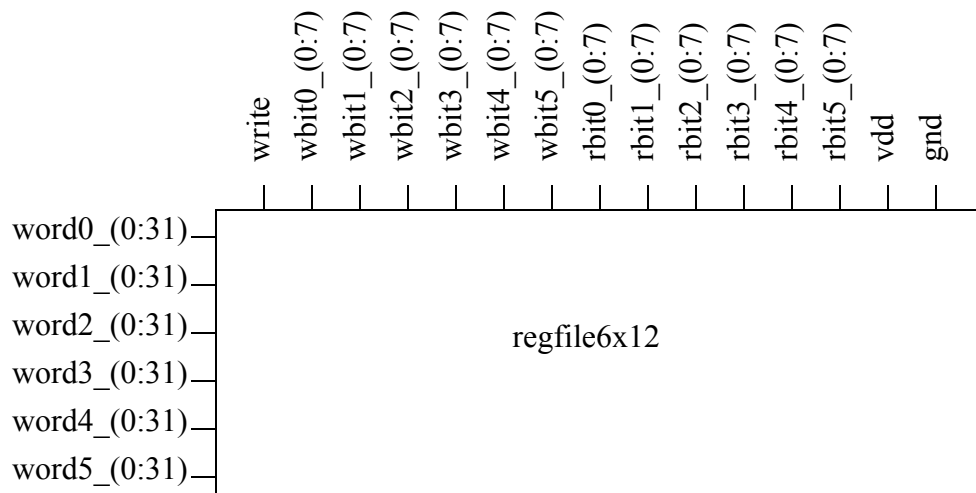
highest level cell name: reg6x12

Remember to encode your project directory and then submit it to the automatic grader. The easiest way to test your file is to use the automatic grader. It does all of the spice translation steps automatically. However, if you want to generate plots on your screen, you will have to manually go through the steps in the inverter tutorial.

### Part B, Due Fri., Oct. 5, 1700

Use the cadence tools to design a 8 bit by 32 word register file in an area of  $5,555,555\lambda^2$  ( $500,000\mu\text{m}^2$ ) and with a worst case delay of 2nsec. Use your memory cell from part A and then make an array of instances of memory cells (do not copy your cell 100's of times).

The register file should have 6 write ports and 6 read ports with the following inputs and outputs.



The notation word0\_(0:31) means that there are 32 word0 lines numbered word0\_0, word0\_1, ... , word0\_31. Similarly for the other terminals.

For the rbit lines to work correctly, they must be precharged. You must provide properly sized pFETs to precharge all of the rbit lines (write drivers for the wbit lines are not necessary). You can use the write control signal to control precharging. The rbit lines should be precharged high when write is high.

This design is too large to simulate easily with spice. The high level circuit simulation irsim will be used instead. An irsim input file in

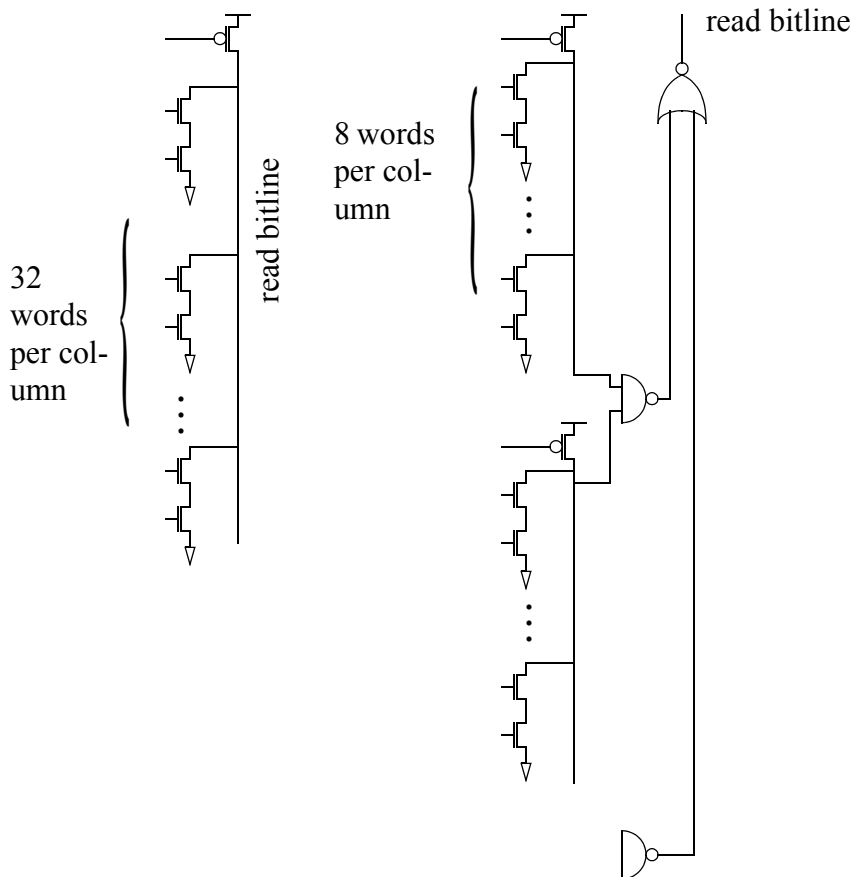
`/x/lgjohn/public/ecen6263/regfile6x12.cmd`

is provided for your convenience in testing your design. A similar file will be used to grade your design.

IMPORTANT: For compatibility with irsim, your power and ground lines should be named vdd and gnd respectively in your highest level cell.

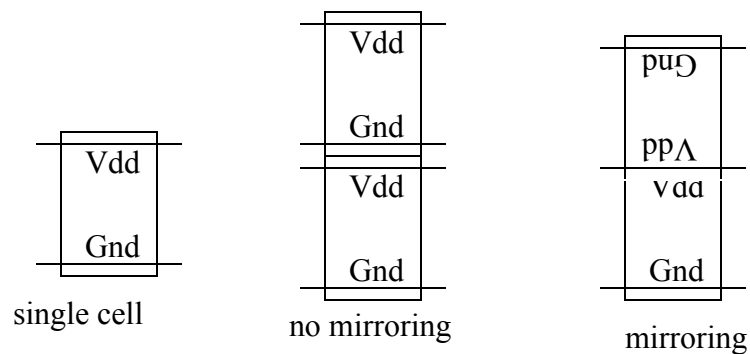
Some of you who have taken the digital computer design course may be wondering why our register file has 6 read and 6 write ports. A 6-wide superscalar would require at least 12 read and 6 write ports. It takes less area to make such a register file out of two of our 6 read and 6 write modules. The write bit lines are connected so that whatever is written into one module is also written into the other which gives an effective 6 write ports. Separate reads can now be done in each of the two modules giving a total of 12 read ports. This trick can always be used to add more read ports.

The timing requirement will be difficult to meet if your layout has all 32 words together with common read bit lines. If we examine the read bit lines, they are the output of multi-input NOR gates (in our case 32 inputs). We know that it is faster to have a tree of gates with smaller numbers of inputs. A better approach is to make submodules of say 8 words and then combine them together as shown below. Note that each submodule needs its own precharge.



To meet the area requirement, you should overlap adjacent cells as much as possible. It may be advantageous to use mirroring techniques to increase the overlap and reduce area. Mirroring helps adjacent cells share common structures like power/ground lines and contacts. For example, suppose your memory cell has power and ground lines as shown.

Stacking the cells vertically would waste considerable space between the metal power and ground lines. Mirroring the cells vertically allows adjacent cells to share the same power and ground lines.



Use the following with the automatic grader.

class name: ecen6263

project name: proj1b

project directory name: regfile6x12\_proj

project library name: regfile6x12\_lib

highest level cell name: regfile6x12

You must include your reg6x12 from project 4a. You should copy the reg6x12\_lib into the regfile6x12\_proj. Remember to encode your project directory and then submit it to the automatic grader. The easiest way to test your design is to use the automatic grader. It does all of the simulation steps automatically. However, if you want to generate plots on your screen, you will have to manually go through the steps in the tutorial.