EEE 414-514 Project Assignment

Due Wednesday, 26 May 2021, **19:30 pm**

SRAM Design

SRAM specifications

In your previous assignment, you have designed a 4x4 SRAM where you have four rows for each 4-bit data.

As your project assignment, you are going to design this 4x4 SRAM array. But assume that the array is two-folded. This means there are now 8 words with 2 bits in each word, placed in a 4x4 grid. The decoder you designed previously will still select between 4 rows. But in addition to the address bit A1 and A0, you have an additional address bit, A2, which is connected too column circuitry and is used to control the bit selection.

If A2 is at logic 0, the left-most column of the folded pairs is selected. If it at logic 1, then the right column is selected (for both read/write operations).

E.g.:

If A2=0, the columns <0, 2> are selected.

If A2=1, the columns <1, 3> are selected.

The circuit you are expected to design is very similar to the example on Slide 32 of Lecture11 titled “Ex: 2-way Muxed SRAM”. You need to design the column circuitry to implement this multiplexing logic with a large-signal sensor.

- In a two-way folded 4x4 SRAM array, you need to extract 2 bits from the 4 bits on each row, as shown in the above example. You can implement this multiplexing by using nMOS pass transistors as we have covered in the class.
- For the bitline conditioning, the input connected to the gate of the pMOS transistors (see Figure 26.a of the “Array Subsystems” chapter) can be the complement of a signal (phi2) in a non-overlapping two-phase clock pair (***phi1***, ***phi2***) as illustrated on Slide 11 of Lecture 13 (where phi1 should be the same clock you used in the decoder design stage). Since the SRAM cell is a memory element like a latch, two-phase clock implementation is suggested here to be safe in terms of both setup and hold time related problems. (You can refer to Lecture 9 to remember the two-phase clocks. The idea is basically decoupling the clock signal by two wires with non-overlapping pulses).
- In order to implement the write enable signals and qualify them with the clock (see the example on Slide 32 of Lecture 13), you can use three-input NOR gates shown in Figure 7.b of the “Array Subsystems” chapter in the textbook.

Note: Remember that the wires in the slides/textbook figures of which labels are appended with “_q1” means that the corresponding signal is a qualified clock—generated AND’ing phi1 with an enable signal (wordline or read/write enable, for example). Similarly, “_v1”

stands for valid during phase1, and "_s1" is used for signals which should remain stable during the phase1.

- Decide on an initial set of transistor sizes for SRAM cells so that read stability and the writability constraints hold (using the smallest area that you can make the circuit function as expected). You may update this when simulating your circuit on Cadence.

SRAM on Cadence

- Open the decoder schematic on Cadence that you have completed for HW7 and update it to draw the corresponding SRAM circuitry.
- Simulate and try to optimize your circuit using Spectre (pre-layout) SPICE simulation.

Note: Now that you have the actual implementation if you prefer, you may re-size the transistors in the decoder as well to optimize the delay.

- Similarly, update the corresponding decoder layout to include the SRAM logic according to the topology drawn on Slide 13 of Lecture11 titled "Thin Cell" (***please check the latest version***). Here, instead of the local interconnection layers drawn in black to connect polysilicon to the metal layer, just extend the same polysilicon layer for connection. (The local connections are mostly used to avoid bends in the poly layer for nanometer processes.)
- Connect the outputs of the decoder you completed in HW7 to the wordlines of the SRAM so that each decoder row controls the read/write operation of the SRAM cells in one row (through the access transistors in the cell). An example decoder-to-memory array interface is shown in Figure 56 of the "Array Subsystems" chapter in the textbook. This is for a ROM rather than an SRAM, but it should give you an idea about how to connect the decoder to the SRAM you designed.
- Measure the time it takes to precharge the bitlines (phase2).
- Set the clock period to the smallest value (to use the highest clock frequency) at which your design functions correctly. You should consider both phase2 (precharge) and phase1 (where read/write operations are performed) as well as the decoder delay. The address bits should be set up before the clock switches. You can assume that the time interval between the two consecutive rising edge of the phi1 is the clock period.
- Read the paper below (given the material you learned in the course, it should be easy to understand most of it). Then, simulate your SRAM for read/write delay, area, and power consumption (for 20 clock cycles) as explained in Section 6 of the paper. Use Cadence and run a post-layout SPICE simulation with Spectre.

  Apostolidis, G., Balobas, D. and Konofaos, N., 2016. Design and simulation of 6T SRAM cell architectures in 32nm technology. Journal of Engineering Science and Technology Review, 9(5), pp.145-149.

Note: If you prefer, you may iterate over the design and re-size the transistors for both the decoder and SRAM to optimize the delay in post-layout simulation (keeping in mind that area is an important design constraint too).

Power Optimization (3pts + [bonus 5pts])

Read subsections 2.6.1 and 2.6.2 in the "Array Subsystems" chapter of the textbook.

- Check if you can decrease the Vdd without disturbing the read stability/writability. Record the minimum value to which you can lower Vdd. Compare how much the initial power consumption improved (without changing the clock frequency you initially set). Report it as a percentage.
- *[Bonus: 5pts]* Apply one of the other low-power techniques suggested in the text and try to improve the power consumption you measured (again, keep the clock frequency at the initial value you set). Report it as a percentage.
  - These techniques may include lowering Vdd even more with a combination of read/write asist techniques as described in the textbook, or
  - The leakage control methods illustrated in Figure 37.

Deliverables:

- A report explaining all your work and results, including a block diagram for the design.
- SPICE netlist of your design with simulation directives. Submit the file separately to Moodle.
- Once you are done with the design, export your layout files using Cadence Virtuoso to GDS II files. In the Layout Editor, use "File → Export Stream From VM" to generate the file and submit to Moodle.
- Convert schematics and simulation results to images or PDF. Do not take screenshots as we need high resolution images to view your schematic and simulation results.
  - For schematics:

    Use "File → Export Image" and set "Size → Scale exported region by" to a value large enough (more than 1.00x) so that details of your schematic are clearly visible. Set a name for your file from "Output → Name" and press "Save to File". Submit the images separately to Moodle.
  - For simulations:

    Use "File → Print → Options" and set "Graph Options → Match window", "Header/Footer → (unclick both)" and press "Print". Submit the PDFs separately to Moodle.