# SRAM Compiler

## Chasen Peters, Matthew Guthaus, Marcelo Siero, Seokjoong Kim, UCSC

**Abstract - the main drive behind this project is to design an easy to use, open source, CAD tool capable of providing multiple levels of abstraction for system designers and memory researchers alike. Most tools within the industry require expensive licenses and are unlikely to give the designer access to the lower level details of a particular memory that is being used. With this compiler it is possible to have a community maintained tool that is readily available to those who wish to use it. This tool will offer fast front end testing for those users who wish to stay clear of layout details, as well as back- annotated simulation with the extracted parasitics. At the moment the compiler is configured to layout an array using the start 6 transistor SRAM, but it can easily configured to layout 8 transistor, 10 transistor, or any other desired bit cell. The compiler will also offer options such as reliability through redundancy and automated timing analysis. The end product is a highly tuned memory model that will reflect the behavior of the fabricated memory.**

## I. Introduction

When designing a memory array, there is an abundance of symmetry that can be applied to the construction of it. If you look at the array itself, it is merely the same bit cell tiled thousands of time to construct the desired size and word size. The peripheral hardware is identical in almost any size variation of the memory; each array needs an address decoder, precharge unit, Column mux, Write Driver, Differential Sensing Amplifier, and an Output Latch. Excluding the column mux and address decoder, the designs of these devices are identical for any size of SRAM. The actual number of devices is dependent on the number of bits to a word.

Since each particular SRAM can follow a general skeleton, it would make sense to automate the process of actually laying out each individual cell that makes up the entire SRAM. This is where the SRAM Compiler comes in, the compiler can layout an entire SRAM array just given the word size and total size in bits.

A majority of the work being done this summer is the verification of the SRAM after it has been constructed by the compiler. One of the largest problems being Layout Vs Schematic checking. In addition to this, it is necessary to acquire timing statistics of each individual SRAM to makes sure that it behaves like it is intended to.

## II. 6-transistor SRAM

Figure 1 below shows the general form of the standard 6-transistor SRAM. The core of the cell can be thought of as two inverters in series as shown in figure 2. The logical ‘1’ or ‘0’ is stored within the feedback loop in devices $M_{1-4}$. Devices M5 and M6 are used to access the cell during write and read cycles, if the bit is not desired by the uses, the nmos devices are turned off, isolating the cell from the rest of the array.

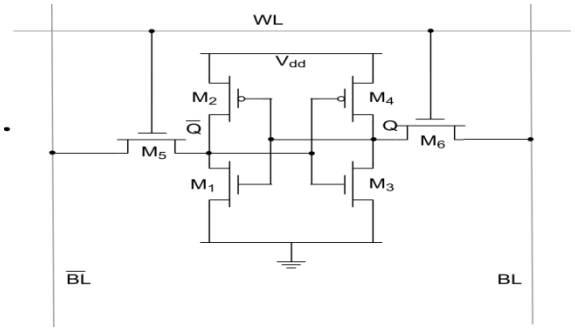


**Figure 1: 6-transistor SRAM cell**

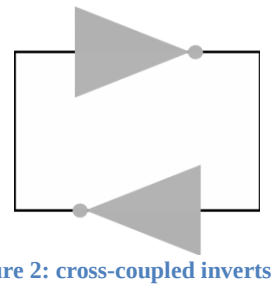

**Figure 2: cross-coupled inverts**

## A. Write Cycle

The write cycle for a 6 transistor cell is built into the cell to begin with. To ensure a proper write, the transistors must be sized in such a way that the PMOS devices can be overpowered to right a '0' to the cell. To write a '1' to the cell, the bit lines must first be driven with the proper values, in this case BL=1 and BLB=0. Next the world line must be asserted which enables the access transistors $M_5$ and $M_6$.

## B. Read Cycle

The read cycle for a 6 transistor cell is started by pre-charging your bit lines to either VDD of VDD/2 depending on how you designed your sensing amp. Now your bit lines charged, the word line is asserted. With the access transistors activated, the bit either discharges the cell or holds at VDD (VDD/2) depending on the value of the cell. At the bottom of the array, this difference is sensed and amplified to the rails. The sense amp is where the output is changed to single ended.

## III. Layout vs. Schematic Checking

Figure 3 represents a schematic from virtuoso, Cadence CAD tool, of the standard 6 transistor SRAM cell. Layout vs. Schematic checking compares these two figures and determines if the two are electrically equivalent; in other words, it makes sure that each transistor connected within the schematic, exists in the layout with identical connections. Looking at figure 4, you can notice a number of different colors; each color represents a different layer of material. The colors allow for a 3 dimensional design to be depicted on a 2 dimensional display. Table 1 gives a list of layers in figure 4.

| Color | Material | Layer |
|---|---|---|
| **Green** | n-doping | 1 |
| **Orange** | p-substrate | 1 |
| **Blue** | Metal 1 | 2 |
| **Purple** | Metal 2 | 3 |
| **Teal** | Metal 3 | 4 |

**Table 1: Layers in layout**

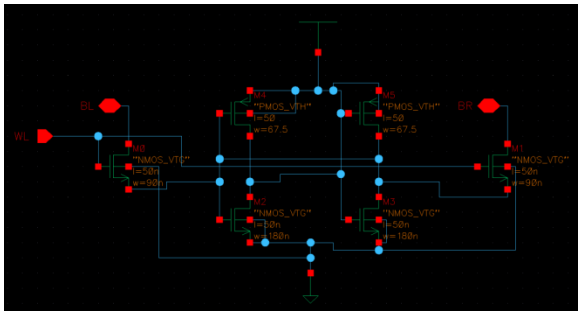


**Figure 3: Schematic of 6-T cell**

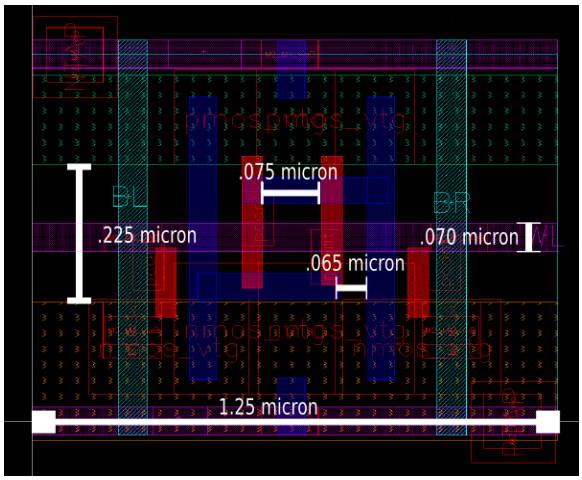


**Figure 4: Layout of 6-T Cell**

## IV. Parasitic Extraction

After the design has been deemed LVS clean, it is now possible to extract parasitic resistances and capacitances from the materials in order to construct a better behavioral model. For any material, this unwanted resistance/capacitance is dependent on the dimensions of the material. If we assume that a particular metal has a sheet resistance of 25 mΩ per square and you have a sheet of metal with dimensions of 65 nano meters by .650 microns, the net resistance of this material would be approximately 250 mΩ because that sheet is made up of 10 65 nano meter squares in series. This is all assuming that the voltage is applied along the length of the sheet. Similar calculations can be made to find the net capacitance of the sheet.

## V. Other Work

Besides the option to compile a standard SRAM array, work is being done to offer options such as redundant hardware to improve reliability. When fabricated integrated circuits, there exist a number of factors that can cause failure within a circuit. By introducing redundant hardware, it is increases the probability that the chip works after it has been manufactured. With the SRAM compiler, it will be possible to introduce redundant rows and columns when necessary. Additional features will include a tool for timing analysis. After the SRAM has been constructed, the tool can exercises the SRAM for functionality and timing constraints.

## VI. Acknowledgments

I would like to thank Matthew Guthaus for giving me the opportunity to work on this project, in addition to all the guidance he and other members of the lab provided. Thank you to the National Science Foundation and SURF-IT for giving me the funding necessary for me to work on this project. And thanks to UCSC hosting the whole event.