High Speed CMOS Circuit Design

Lecture 9: Arrays

1.0 Introduction

This lecture explores the design of a variety of array structures. SRAM arrays account for the majority of transistors on most processors and must be very fast. We'll look at a number of SRAM issues including overall architecture, cell design, decoding, and bitline sensing.

We'll then extend the ideas from SRAMs to other arrays, looking at ROMs, CAMs (Content-Addressable Memories), and PLAs (Programmable Logic Arrays).

2.0 SRAM Design

Static Random Access Memory (SRAM) is used extensively on chips, especially in register files and the ever-expanding caches. Static RAMs retain their contents until power is turned off. This differs from Dynamic RAMs (DRAM) which hold their value on a capacitor. To combat leakage, data in a DRAM must be periodically read and rewritten to "refresh" the DRAM. To read an SRAM, an address is presented and the contents of that address appear on output lines. To write an SRAM, both address and data are presented and the data is stored into the address. Fast SRAM design is a specialized art, but this section aims to give a general understanding of the issues that SRAM designers face.

2.1 Architecture

The principle characteristics of a RAM are its total size S, the word size W, and the number of ports. Ports may be read-only, write-only, or read/write.

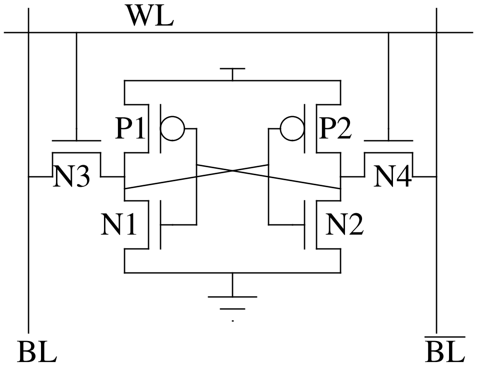
Given these specifications, RAM designers make a number of choices. A small RAM can be built as a single array. The address is decoded to activate one row by asserting the appropriate wordline. The contents of the row may be read or written through bitlines.

Beyond a certain size, a single array becomes too slow because wires are too heavily loaded and contribute significant wire RC. At this point, the designer may divide the memory into multiple banks. Thus, the designer must determine the number of banks, the number of words, and the number of bits per word.

2.2 Cell Design

A schematic of a single port SRAM cell is shown in Figure 1. The cell has a word line to activate the port and two bitlines. When the wordline WL is low, the cell is idle and the cross-coupled inverters within the cell retain the contents. When the wordline is raised, the cell can be read or written. For reads, the bitlines BL and $\overline{\mathrm{BL}}$ start precharged high. When the wordline rises, the access transistors N3 and N4 pull down one of the wordlines. Note that the transistors in the cell must be ratioed so that the cell cannot be inadvertently written during a read. For writes, one bitline is driven high and the complement low. When the wordline rises, the values on the bitlines get written into the cell.

FIGURE 1. Single Port SRAM Cell



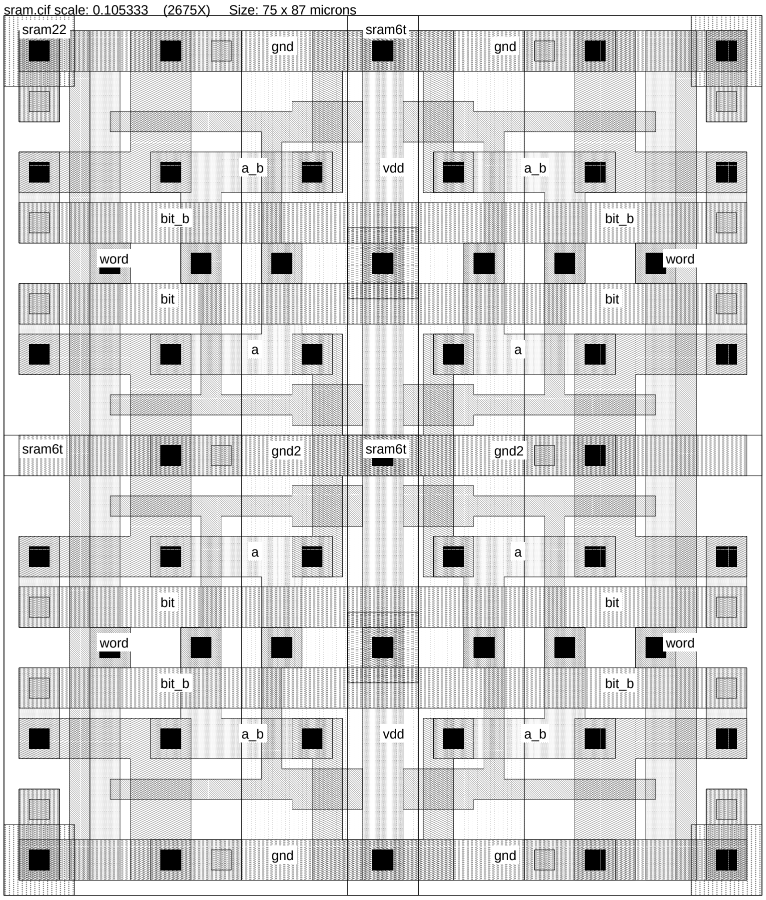
The SRAM circuit is a ratioed cell and certain conditions must be met for the cell to function correctly. The ratios must be set so that reads do not disturb the contents, yet writes successfully change the contents by writing a 0 to one side of the cell. The read disturb property involves a ratio of N1 being stronger than N3. The write property requires a ratio of N3 being stronger than P1.

When a SRAM array has a small number of ports, the diffusion capacitance dominates bitline capacitance. Thus N3 can be minimum sized. N1 can be larger, both to satisfy the read disturb property and to increase the drive for reads. When an array has a large number of ports, wire capacitance may start to be important and there is plenty of room under the metal lines for larger transistors. Thus, all of the transistors may be sized larger.

Minimizing cell area is a major objective when designing SRAM cells. Great effort goes into placing transistors under metal tracks so that the area is limited by necessary metal lines. Often, this includes diagonal routing and special design rules applying only to dense SRAM layout. Cells are also mirrored to share power and ground between them. A few metal lines are required for connections within the cell. Additional lines are required for each port. Thus the area increases quadratically with the number of ports, so ports are very expensive resources.

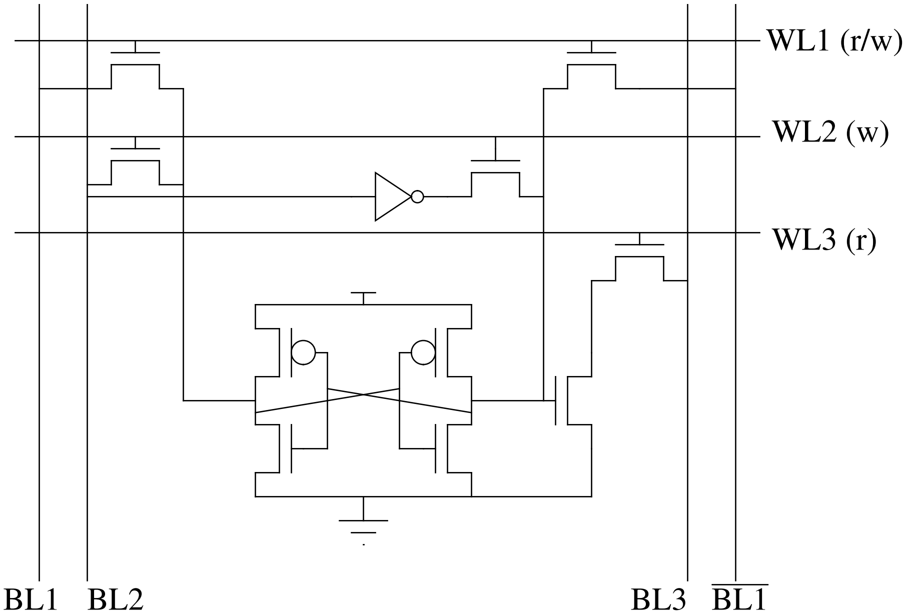
The largest part of SRAM delay comes from bitline sensing, so bitlines are optimized for minimum capacitance. Whenever possible, the drains of access transistors are shared between adjacent RAM cells to halve the diffusion loading on the cell. Connecting the wordlines to the polysilicon gates of the access transistors can be difficult and may increase cell area. To avoid this, the wordline is often also run continuously in poly and just strapped periodically with metal. Figure 2 shows the layout of a 2x2 array of single-ported SRAM cells. Notice how all four cells are mirrored to share VDD and GND and bitline contacts. In this layout there is space for a contact from the metal to poly wordline in every cell.

FIGURE 2. Layout of 4 SRAM cells



A SRAM array with multiple ports may dedicate ports as read-only, write-only, or read/write. Read-only and write-only ports can save area by using only a single bitline per port. Such a design for an array with a read-only, a write-only, and a read/write port is shown in Figure 3. Notice how an inverter is used on the write port to save routing a bitline. Also notice how the read port is single-ended to save a bitline. This works well for small arrays, but large arrays requiring fast access time usually use differential bitlines to allow faster sensing and reject common-mode noise.

FIGURE 3. Multi-port SRAM cell



Data caches often have 2 or more ports to allow multiple simultaneous LOAD/STORE operations. Register files may have 3 ports, 2 read-only and 1 write-only, for each instruction that can be issued to the integer units. Thus, 4-6 way superscalar machines have register files completely dominated by metal.

2.3 Decoding

An array may require several levels of address decoding to access the proper address. A wordline decoder decodes N address bits into 2^N one-hot wordlines. The wordlines are usually gated with a clock so they are active for part of the cycle. A column decoder may use a few more address bits to select which data bits are requested when the array word size exceeds the access word size. For large arrays divided into multiple banks, the remaining bits are used to select which bank should be accessed.

If all address bits arrived early, as they often do in a register file, decoding is non-critical. Cache access may use an address bypassed from the execution units, so the address arrives late and wordline decoding is critical. Other decoding usually remains less critical because the results are not needed until later and because the decoders are smaller.

A decoder is essentially 2^N N-input AND gates. The gates are usually built from several stages of smaller NAND, NOR, and NOT gates. A rough estimate of the right number of stages of a decoder can be obtained by looking at the fanout. Any address bit toggling can turn on or off any wordline. Thus, the decoder necessarily involves a fanout equal to the total wordline capacitance divided by the input capacitance of an address bit. Consider an array with 128 words, each 256 bits wide. Let each bit present a capacitance of 5 fF to the wordline. Let the input capacitance on any address line be 100 fF. The total capacitance of all the wordlines is 128 * 256 * 5 = 163 pF. Thus, the fanout from an address line is 1630. If no logical effort were involved to do the decode, $\log_4 1630 = 5.3$ stages of fanout-of-4 inverters should be used to drive the large load. Since there is some logical effort, about 6-8 stages of decoding is optimal. Therefore, the stages can be very simple to minimize the logical effort, such as alternating 2-input NANDs and inverters.

2.4 Bitline Sensing

Writes are usually fairly fast because strong drivers can drive the appropriate bitlines and flip the contents of the cells when the wordline is raised. Reads are more critical because when the wordline activates, small access transistors must pull down the heavily loaded bitline. Common approaches are to use short bitlines or to use sense amplifiers.

For small arrays such as register files with up to 32 words, the bitlines are not loaded too heavily. Hence, accessing the array is much like driving a domino gate. The bitlines resemble precharged dynamic A2O32 gates. They can be connected to a high-skewed CMOS gate which trips when the bitline has pulled down by less than 50%.

Waiting for that much swing on a large array is often too slow. Thus, longer bitlines normally use sense amplifiers. The sense amplifiers are triggered by some delay after the wordlines rise to amplify the small differential signal between true and complementary bitlines. This difference must be large enough to exceed any offset voltages in the sense amplifier. The sense signal is often produced by a self-timed delay. For most processes 128 words per bitline gives good performance when sense amplifiers are used.

When sense amplifiers are used, it is important to minimize the differential noise on the bitlines to allow the sense of a small signal. Large equalization transistors are used during precharge to balance the voltage between the two bitlines. Bitlines may also be twisted so that coupling noise from adjacent lines appears as common-mode, rather than differential noise.

Larger arrays can also be created with hierarchical bitlines. The bitlines are divided into segments of 16 or 32 bits. Each segment, or local bitline, drives a global bitline running the entire height of the bank. For a 256 word array, this could be viewed as breaking the slow A2O256 domino gate into two stages: an A2O16 gate for the local bitline and an OR16 gate for the global bitline. Such a hierarchical technique is becoming more feasible as more levels of metal are available.

The time required to drive wordlines is logarithmic with the number of bits per wordline, while the time to sense bitlines is linear with the number of wordlines, albeit with a smaller constant factor. To balance the time for word line drive and bitline sense, most arrays are roughly square or have more bits per wordline than words per bitline. If there are 128 words per bitline, there will be at least 128 bits per wordline. Since a data cache usually needs to read 32 or 64 bit quantities, column multiplexors are used to select the appropriate bits from the wider word read from the array.

2.5 Conclusions

As mentioned before, the details of high speed SRAM design are quite specialized. We have surveyed the basic issues. Cells are carefully designed to minimize bitline capacitance. Large arrays are broken into multiple banks to keep loading and RC delay of bitlines and wordlines reasonable. Sense amplifiers or hierarchical bitlines are needed to reduce the delay of long bitlines.

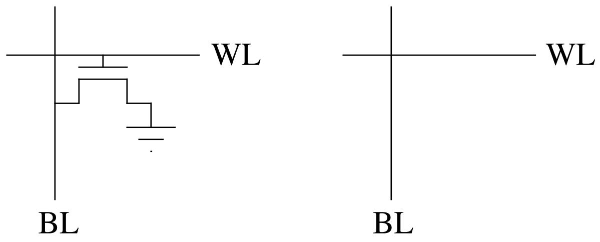
SRAM arrays may consume a large portion of total chip power. This can be greatly reduced through some simple techniques such as only activating the bank which will be used. Self-resetting circuits are also popular to reduce the power consumed by switching word lines.

There is growing interest in integrating DRAM onto chips in place of SRAM. DRAM can be a factor of 10 denser than SRAM. Since cache memories are taking such a large fraction of the area of processors, the level 2 caches could be smaller or could contain significantly more bits if DRAM were used. Other chips like signal processors could cut system costs by integrating the entire system memory on the same chip as the processor. Unfortunately, no attempt to merge DRAM and logic on a chip has met with good results yet. Logic processes are optimized for speed; they are expensive per unit area and don't contain the specialized capacitor structures used to build dense DRAM arrays. DRAM processes are optimized for DRAM yield; their transistors are too slow and they have insufficient metal layers for good processor implementations. Nevertheless, the benefits of merging logic and DRAM are becoming more compelling so it seems likely that eventually engineers will develop good techniques combining most of the strengths of each.

3.0 ROM Design

Read-Only Memories (ROMs) share many of the issues of RAM design. They have wordline decoders and bitline sense circuits. ROMs are much denser, however, because each cell uses only one transistor, as shown in Figure 4. If the transistor is present, it pulls down the bitline when the wordline is asserted, corresponding to a logic 0 in the cell. If the transistor is absent, the bitline is unaffected, corresponding to a logic 1 in the cell.

FIGURE 4. ROM Cell



Cell contains 0

Cell contains 1

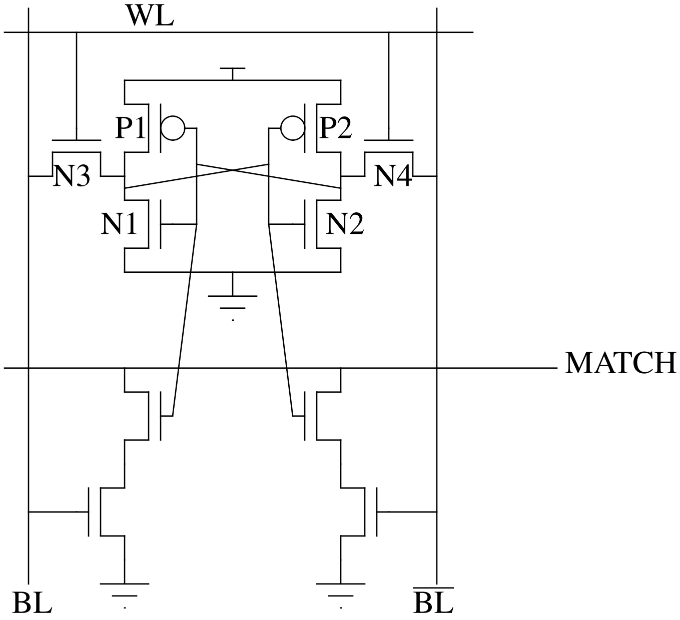
ROMs are not used as much in modern processors as they were in microcoded CISC machines.

4.0 CAM Design

Content-addressable memories (CAMs) are often used in translation lookaside buffers (TLBs) and other associative memories. They are similar to SRAMs in that they can be read or written. They can also perform a match operation, in which a data word is presented and any rows of the CAM containing the same value assert a match signal.

A CAM cell is shown in Figure 5. It typically occupies 2-3 times the area of a SRAM cell, so CAMs are relatively expensive. It can be read or written when the word line is high. When the word line is low, it just performs a match operation, pulling the match line low if the contents of the cell does not match the value on the bitline. The match line performs a wired-OR across multiple bits so it is pulled low if any bit in the word mismatches.

FIGURE 5. CAM Cell



Additional circuitry is needed on the edge of the array to precharge the MATCH signal. Also, notice that the match signal cannot drive other domino gates because it is monotonically falling.

5.0 PLA Design

Programmable Logic Arrays (PLAs) are popular in certain design styles because they can quickly calculate complex AND/OR functions and because they can be automatically generated. Improvements in synthesis tools have shifted most control logic to synthesized standard cell implementations instead, but certain applications like radix 4 divider quotient selection algorithms are still fastest to implement with PLAs.

A PLA consists of two arrays. The first, called the AND plane, takes a number of true and complementary inputs and computes the logical AND of various inputs. These ANDs are called minterms. The second produces outputs which are the logical OR of the minterms. Therefore, the PLA can compute any logic functions written in sum-of-products cannonic form.

The PLA takes advantage of the speed of pseudo-NMOS or precharged NOR gates to implement the AND and OR planes with NOR arrays, using DeMorgan's law. A simple PLA describing a full adder is shown in Figure 6.

FIGURE 6. Pseudo-NMOS PLA full adder

Pseudo-NMOS PLAs have the usual strengths and weaknesses associated with pseudo-NMOS circuits. Since static power consumption is unacceptable for most processors and since dynamic PLAs are faster anyway, dynamic PLAs are more popular.

The main difficulty with dynamic PLAs is triggering the OR plane. The AND plane produces minterms which are precharged high and may drop low. The OR plane must not evaluate until the AND plane has completed. This is usually done by constructing a self-timed circuit. The self-timed circuit contains a replica of the most heavily-loaded AND plane row being pulled down through the latest arriving input. When the row completes, it toggles an inverter which rises, acting as a clock to the OR plane. A more elaborate picture of this scheme is in Figure 15 of the Skew-Tolerant Domino Circuits paper.