SRAMs are organized as an array of memory locations, where a memory access is either reading or writing all the bits in a single location.

Here we see the component layout for a 8-location SRAM array where each location hold 6 bits of data.

You can see that the individual bit cells are organized as 8 rows (one row per location) by 6 columns (one column per bit in each memory word).

The circuitry around the periphery is used to decode addresses and support read and write operations.

To access the SRAM, we need provide enough address bits to uniquely specify the location.

In this case we need 3 address bits to select one of the 8 memory locations.

The address decoder logic sets one of the 8 wordlines (the horizontal wires in the array) high to enable a particular row (location) for the upcoming access.

The remaining wordlines are set low, disabling the cells they control.

The active wordline enables each of the SRAM bit cells on the selected row, connecting each cell to a pair of bit lines (the vertical wires in the array).

During read operations the bit lines carry the analog signals from the enabled bit cells to the sense amplifiers, which convert the analog signals to digital data.

During write operations incoming data is driven onto the bit lines to be stored into the enabled bit cells.

Larger SRAMs will have a more complex organization in order to minimize the length, and hence the capacitance, of the bit lines.

The heart of the SRAM are the bit cells.

The typical cell has two CMOS inverters wired in a positive feedback loop to create a bistable storage element.

The diagram on the right shows the two stable configurations.

In the top configuration, the cell is storing a "1" bit.

In the bottom configuration, it's storing a 0 bit.

The cell provides stable storage in the sense that as long as there's power, the noise immunity of the inverters will ensure that the logic values will be maintained even if there's electrical noise on either inverter input.

Both sides of the feedback loop are connected via access FETs to the two vertical bit lines.

When the wordline connected to the gates of the access FETs is high, the FETs are "on", i.e., they will make an electrical connection between the cell's internal circuity and the bitlines.

When the wordline is low, the access FETs are "off" and the bistable feedback loop is isolated from the bitlines and will happily maintain the stored value as long as there's power.

During a read operation, the drivers first recharge all the bitlines to Vdd (i.e., a logical "1" value) and then disconnect, leaving the bitlines floating at 1.

Then the address decoder sets one of the wordlines high, connecting a row of bit cells to their bitlines.

Each cell in the selected row then pulls one of its two bitlines to GND.

In this example, it's the right bitline that's pulled low.

Transitions on the bitlines are slow since the bitline has a large total capacitance and the MOSFETs in the two inverters are small to keep the cell has small as possible.

The large capacitance comes partly from the bitline's length and partly from the diffusion capacitance of the access FETs in other cells in the same column.

Rather than wait for the bitline to reach a valid logic level, sense amplifiers are used to quickly detect the small voltage difference developing between the two bitlines and generate the appropriate digital output.

Since detecting small changes in a voltage is very sensitive to electrical noise, the SRAM uses a pair of bitlines for each bit and a differential sense amplifier to provide greater noise immunity.

As you can see, designing a low-latency SRAM involves a lot of expertise with the analog behavior of MOSFETs and some cleverness to ensure electrical noise will not interfere with the correct operation of the circuitry.

Write operations start by driving the bitlines to the appropriate values.

In the example shown here, we want to write a 0-bit into the cell, so the left bitline is set to GND and the right bitline is set to VDD.

As before, the address decoder then sets one of the wordlines high, selecting all the cells in a particular row for the write operation.

The drivers have much larger MOSFETs than those in the cell's inverters, so the internal signals in the enabled cells are forced to the values on the bitlines and the bistable circuits "flip" into the new stable configuration.

We're basically shorting together the outputs of the driver and the internal inverter, so this is another analog operation!

This would be a no-no in a strictly digital circuit.

Since n-fets usually carry much higher source-drain currents than p-fets of the same width and given the threshold-drop of the n-fet access transistor, almost all the work of the write is performed by the large n-fet pulldown transistor connected to the bitline with the 0 value, which easily overpowers the small p-fet pullup of the inverters in the cell.

Again, SRAM designers need a lot of expertise to correctly balance the sizes of MOSFETs to ensure fast and reliable write operations.

It's not hard to augment the SRAM to support multiple read/write ports, a handy addition for register file circuits.

We'll do this by adding additional sets of wordlines, bitlines, drivers, and sense amps.

This will give us multiple paths to independently access the bistable storage elements in the various rows of the memory array.

With an N-port SRAM, for each bit we'll need N wordlines, 2N bitlines and 2N access FETs.

The additional wordlines increase the effective height of the cell and the additional bitlines increase the effective width of the cell and so the area required by all these wires quickly dominates the size of the SRAM.

Since both the height and width of a cell increase when adding ports, the overall area grows as the square of the number of read/write ports.

So one has to take care not to gratuitously add ports lest the cost of the SRAM get out of hand.

In summary, the circuitry for the SRAM is organized as an array of bit cells, with one row for each memory location and one column for each bit in a location.

Each bit is stored by two inverters connected to form a bistable storage element.

Reads and writes are essentially analog operations performed via the bitlines and access FETs.

The SRAM uses 6 MOSFETs for each bit cell.

Can we do better?

What's the minimum number of MOSFETs needed to store a single bit of information?