Chapter Two - SRAM

Introduction to Memories

The most basic classification of a memory device is whether it is Volatile or Non-Volatile (NVM's). These terms refer to whether or not a memory device loses its contents when power is removed. Volatile memories lose their contents when power is removed. The physical design of the storage circuit, called a cell, determines whether a device is volatile or non-volatile. Volatile memories can be further divided into two groups called static and dynamic. Static memories tend to retain their contents for as long as power is applied to the device. While dynamic memories require periodic reminding, known as refreshing, to retain their data. Dynamic RAMs require a refresh cycle every few milliseconds or the data stored in them is lost.

In this book the SRAM is used as the reference device for the discussions describing the basic functional blocks of a memory device, the various faults related to that function, and various tests to detect each type of fault. Later in this book other memory device types are discussed by describing the differences between the SRAM and that device.

Static Random Access Memory (SRAM)

A Static RAM is a volatile memory device which means that the contents of the memory array will be lost if power is removed. Unlike a dynamic memory device, the static memory does not require a periodical refresh cycle and generally runs much faster than a dynamic memory device.

SRAM cells are larger, that is they take up more real estate on the silicon than DRAM cells. What they offer is that they have faster access times and don't use multiplexed addressing.

Static memories are organized in many different ways from a straight static RAM with addresses, data, and control pins separate, to First In First Out (FIFO) devices that shift the data from cell to cell.

Static RAM devices have the fewest unique characteristics among memory types. Or to state the same concept another way, SRAM's have the minimum attributes to make them a memory and none of the more complicated characteristics of other memory types.

In the following sections the block diagram of a SRAM device is built one major functional block at a time. This will help you to understand the function of the array, the word line, the bit line, the X & Y addresses lines, and all the other components of a memory device.

Note: The semantics in the memory world are confusing because a bit is used to represent several things. There are address bits, data bits, and control bits to name a few. Each has a unique purpose and is described later in this book.

Static RAM

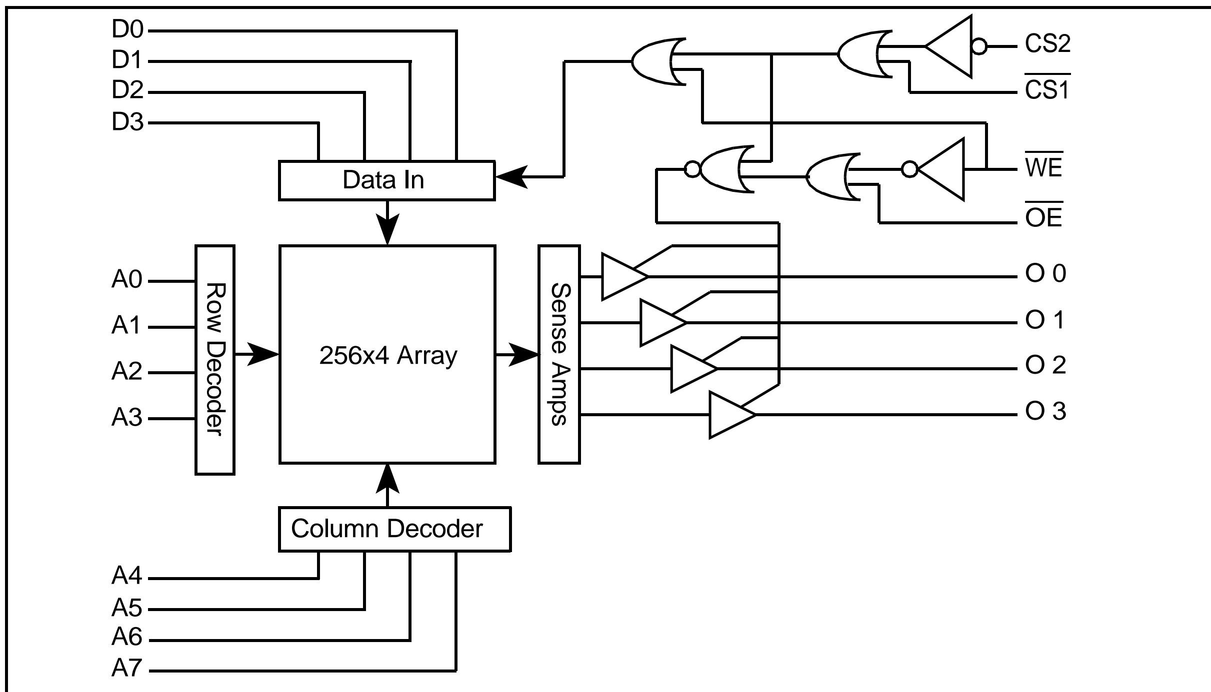

Figure 2-1 Static RAM Block Diagram

If you have taken the logic course from Soft Test, you might recognize or remember this device. It is the same one used as an example for the logic course. In that class the device functionality is discussed in general terms, but we will go into more detail. There, the DUT was tested using vectors and here, we will use an APG (Algorithmic Pattern Generator).

Every memory device consists of addresses, data in, data out, and some type of control logic. The 256x4 static RAM used in this example is made up of 8 address lines, 4 X and 4 Y (arbitrarily assigned to row and column), Data In, and Data Out.

It has Control pins consisting of Write Enable bar (take it low to write the device), Output Enable bar (take it low to read the device), and two Chip Selects; one high true logic; one low true logic.

The SRAM Cell

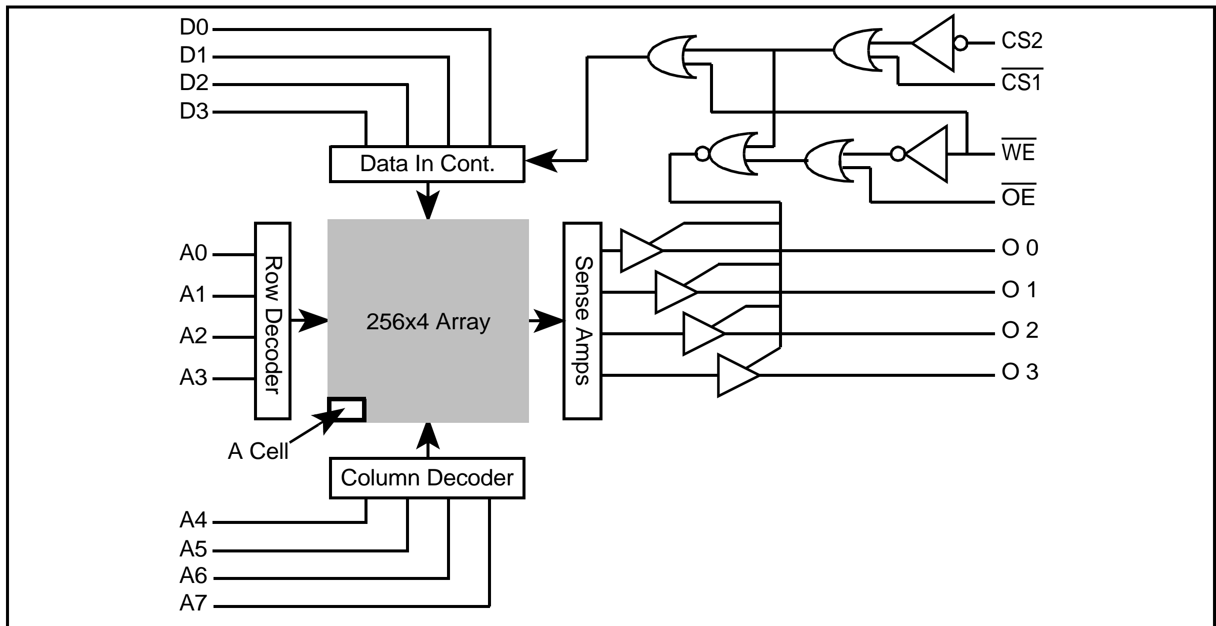

Figure 2-2 The Memory Cell

The first thing to understand with any memory device is what type of cell is used. In this case, it is a Static-Ram cell. Different cells have radically different characteristics regarding power, speed, and functionality.

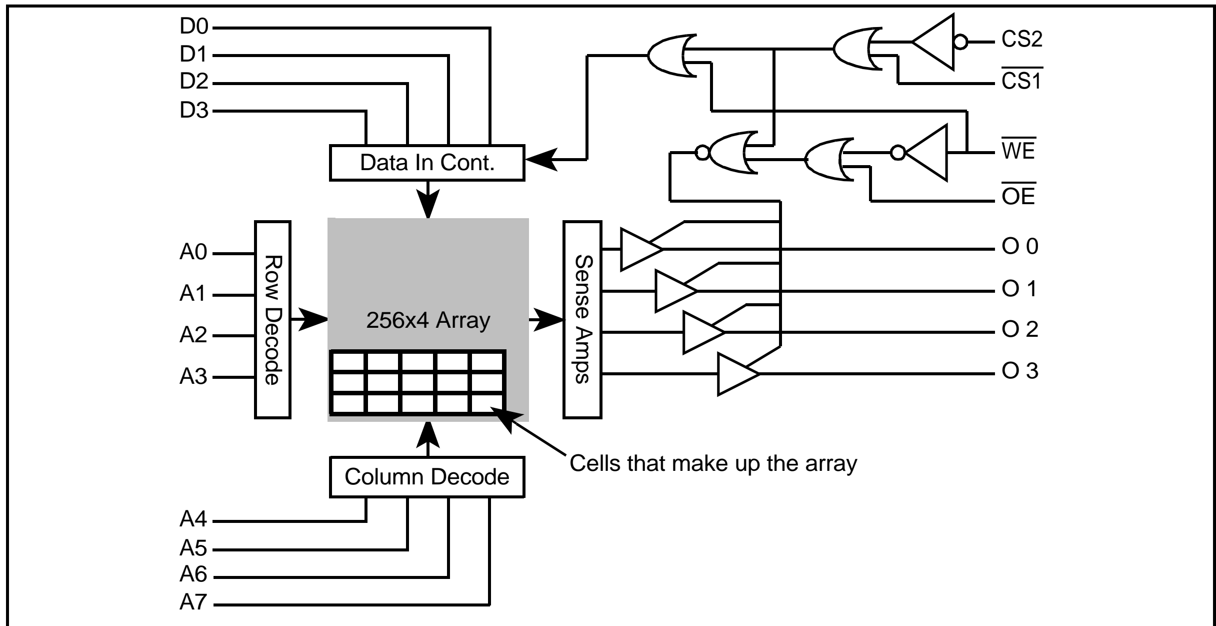

Figure 2-3 Memory Array Cells

The basic cell is repeated over and over again to create an array. Before we move on to the array itself, let's continue to focus on the individual cells.

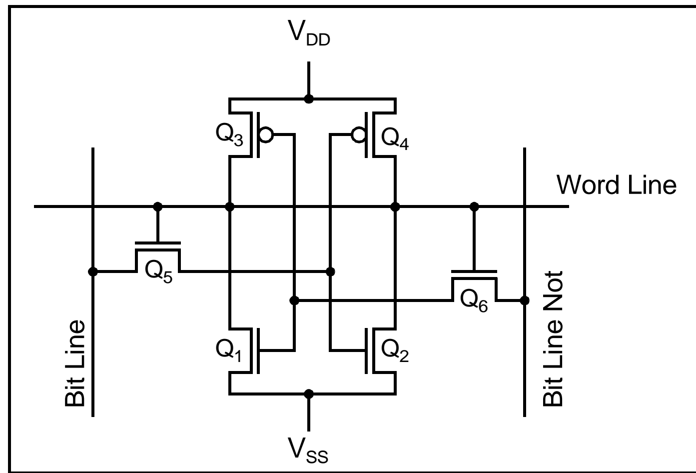

Figure 2-4 SRAM Memory Cell

The SRAM memory cell is actually a flip-flop that uses dual bit lines for Q and Q/. The bit lines actually carry the data that was stored in the cell. In the case of an SRAM, there is a bit line **not**, that carries data bar. The word lines are activated by the address applied to the device.

After the bit line exits the array, it gets manipulated by the address decoder to choose a specific bit to show as data out from the device. Note, that the Word Line actually turns on both bit lines in the cell.

The six transistor design (See Figure 2-4) of a static RAM has a much larger die size for a given memory array and requires more power than that of dynamic memory.

Fault Types

- Stuck at Fault.
- Coupling Fault.
- Neighborhood Pattern Sensitive Fault.
- Address Fault.
- Transition Fault.

Let's pause in development of our memory block diagram and consider faults. We've chosen some specific faults so the class will be able to see what kinds of problems can be associated with each different piece of the device block diagram.

We are going to cover each of these in detail, but let's look at this from a high level, first. What is a fault? It is the same thing as on any other semiconductor device. A fabrication process flaw that causes the device to function improperly. It could be a shorted metal or poly line or a missing contact.

Since memory devices are really large arrays of repeated cells, the types of faults normally found are predictable. Specific types of faults on memory devices have been given generic names. A few of these are listed below:

Stuck at Fault – a cell is stuck at a zero or one level (think of a short in the transistor)

Coupling Fault – connects two or more cells together.

Neighborhood Pattern Sensitive Fault (NPSF) – cell is affected by what happens around it usually during writing (a capacitive coupling fault).

Address fault – connects two or more addresses together.

Transition Fault – when a cell or line fails to undergo a 0->1 or a 1->0 transition.

Stuck at Fault

- When a cell is stuck at a zero or one level.Coupling Fault.
- Often a short in one or more of the transistors that make up the cell.
- Can be detected by writing and reading every cell to a zero AND a one.

Although, a ‘stuck at’ fault is typically caused by one or more shorted transistors within the cell, it can be caused by other things. Regardless of the cause, the detection process is the same.

If a cell was shorted to a zero level and we wrote (then read) all zeroes into the array, the device would pass. So to detect this fault we have to test every cell for the ability to store both a one and a zero.

The main focus in a memory test program is the memory array. Arrays are a large collection of circuits (cells) such as the one in Figure 2-4 which in turn defines the size, or storage capacity, of the memory device. When making a graphical representation of an array most people think of it as two dimensional with each box representing a single memory cell.

The Array

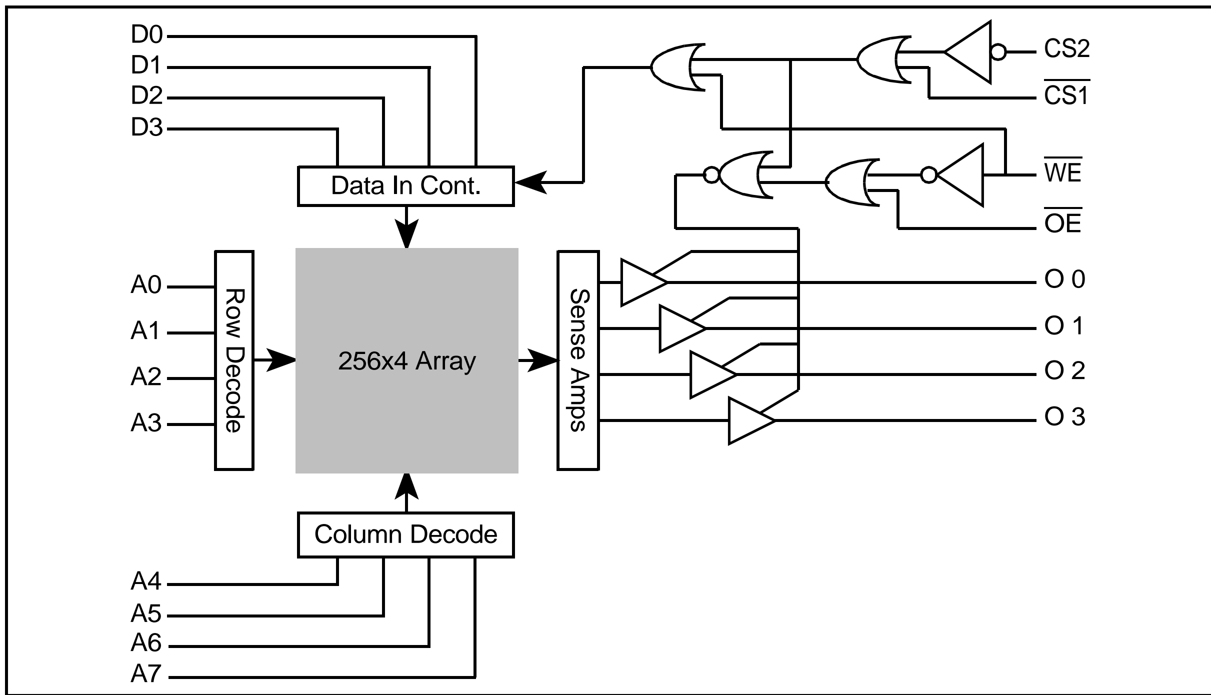

Figure 2-5 The Memory Array

Array related faults are indicated by interaction between two or more cells. As an example, if two cells are shorted together, it is easy to imagine that writing to either cell would cause both cells to store the same written information.

Typically there are only a few places that one cell could short to another cell. By talking to the Design Engineer, it is possible to identify those places. Thus, in turn, the scope of the testing can be reduced to only look for those specific shorts that can occur. It doesn’t make sense to spend test time looking for things that can not occur.

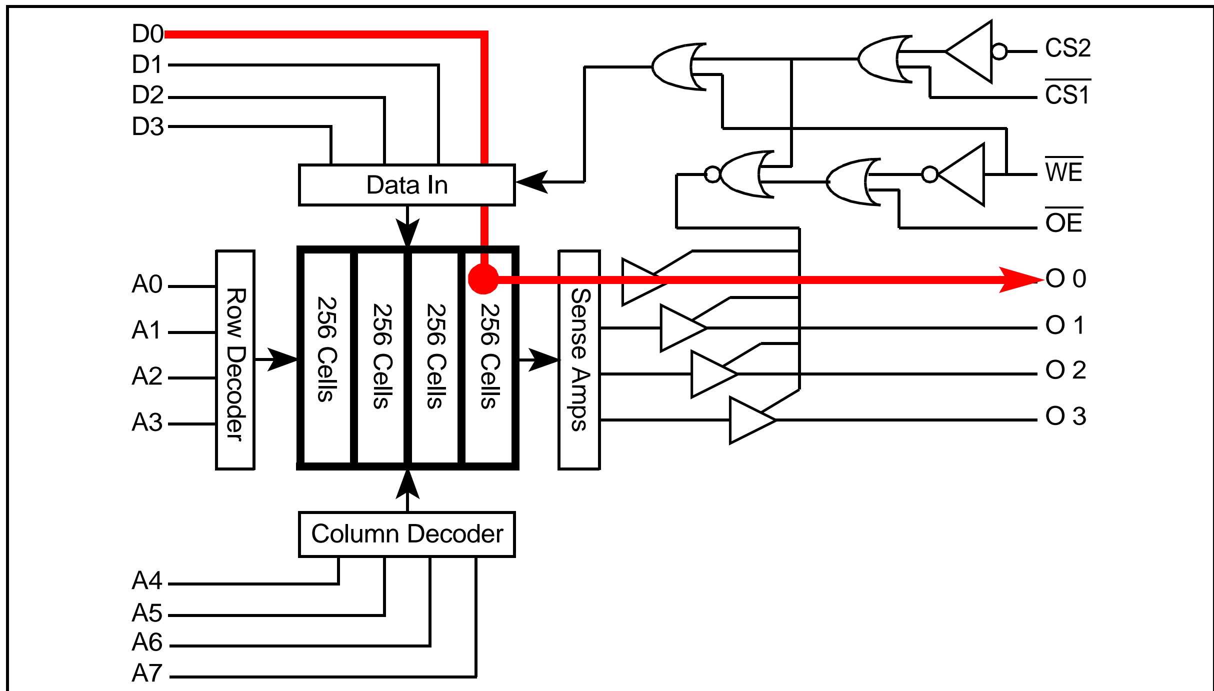

Figure 2-6 Data Bit 0 Path

Figure 2-6 is an example of a Static RAM that has four data lines going in (D0-D3) and four data lines going out (O0-O3) with each data line going to its own array. This allows the four independent bits of data to be written to (or retrieved from) the same logical location within their respective arrays.

The device in Figure 2-6 has eight address lines (A0-A7) that produce 256 (2^{8th}) unique addresses. Because the device has four arrays it can store 1024 (1K) single bits of data.

Because each address applied to the outside of the DUT is sent to all four arrays, the four output pins produce separate data at each address location.

The data arrays share address, power, sense amps, control logic, and all other blocks on the device, but they are really tested in parallel. We are effectively testing four memory 256 cell arrays in parallel.

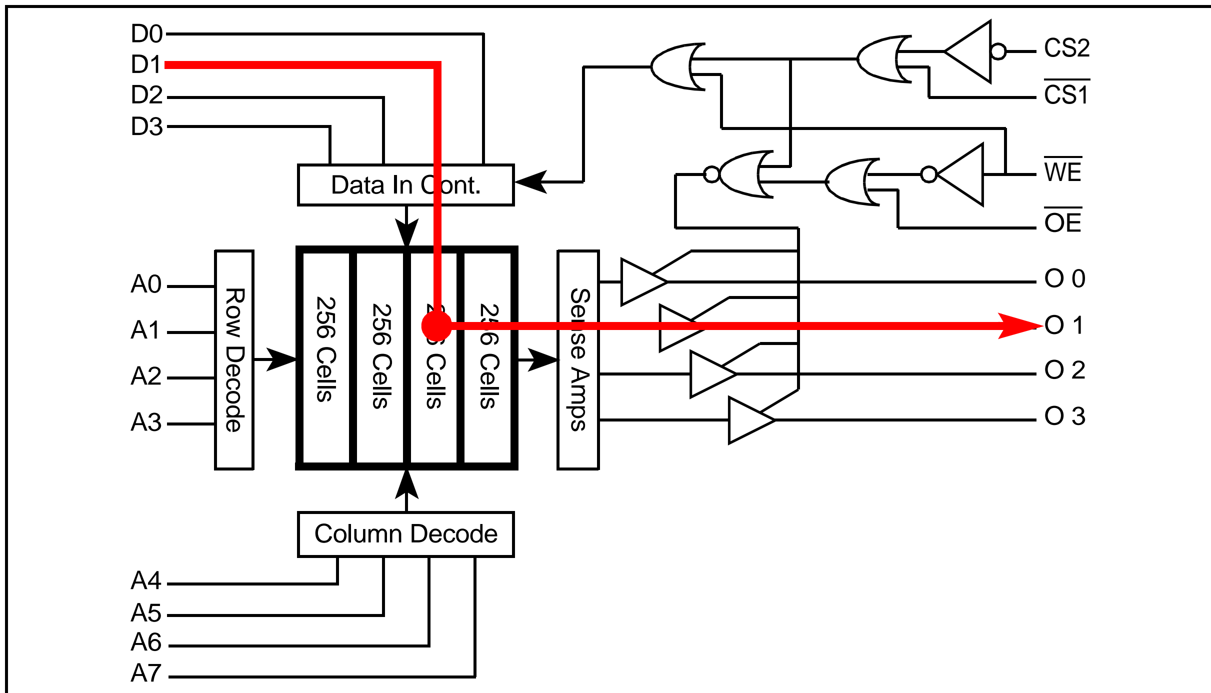

Figure 2-7 Data Bit 1 Path

The Data In and Data Out lines for this device are separate. However, most devices use a common data bus that is used to write and read data on the same DUT pins. This same pattern of the input data bit D^n corresponding to the output data bit O^n continues through all the data bits.

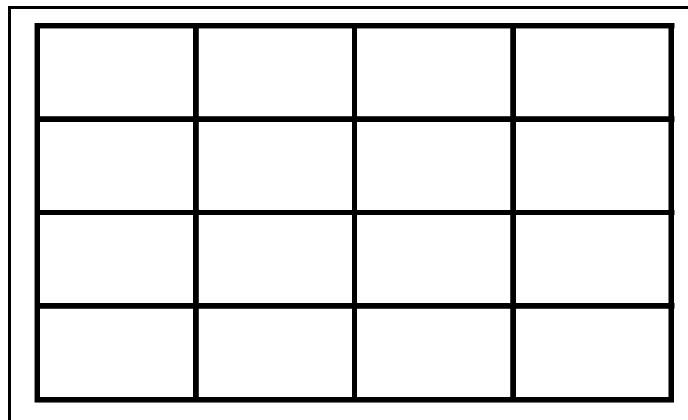

Figure 2-8 A 4x4 Memory Array

Let's focus on a single output and examine how its array is organized.

Figure 2-8 is a graphical representation of a memory array having sixteen cells with four rows, four columns, and one data bit. The number of rows and columns within the array defines how many row address lines (X address) are needed and how many column address (Y address) lines are needed.

While keeping the example device in mind, this table type drawing can make things easier to communicate. We can see details in 16 cells that get lost in a drawing of 256 cells.

Array Faults

Coupling Fault

- When two or more cells are connected.
- Can be detected by writing $\overline{\text{DATA}}$ around each location of DATA.
  - Eight cells adjacent to each cell not along an edge.

0	0	0
0	1	0
0	0	0

Figure 2-9 Coupling Fault

If a 'Stuck at Fault' is categorized as a short in one or more transistors that make up a cell, we can categorize a 'Coupling Fault' as a short between two or more cells. The only way to prove that cells are not shorted together is to store and retrieve different data from adjacent cells.

The pattern shown in Figure 2-9 is theoretically the only way to test for coupling faults. What if we worked with the design group and found that the device could not have a coupling fault on the diagonal? How could our above pattern be changed to make the pattern more efficient? Think checkerboard.

Neighborhood Pattern Sensitive Fault (NPSF)

- When a cell is affected by what happens around it.
- Can be thought of as a capacitive coupling fault.
- Test technique is to write/read a value many times into the adjoining cells.
- Sometimes referred to as a “Disturb” fault.

Shorts are only one type of problem caused by process errors. Process errors can also create capacitors between cells. If the a capacitor is large enough and testing is done very quickly (with a short cycle time), this type of failure can go undetected.

A basic premise for a memory device is that every cell is completely independent. Whatever is written to one cell should not affect any other cell in the array. Due to the test time required to check for this fault on every cell (and the decreasing likelihood that the fault exists) this is seldom tested during production. The algorithm technique used to test for an NPSF is called surround disturb.

There is a pattern shown later in this text called a modified Butterfly, that tests for this fault.

Addressing

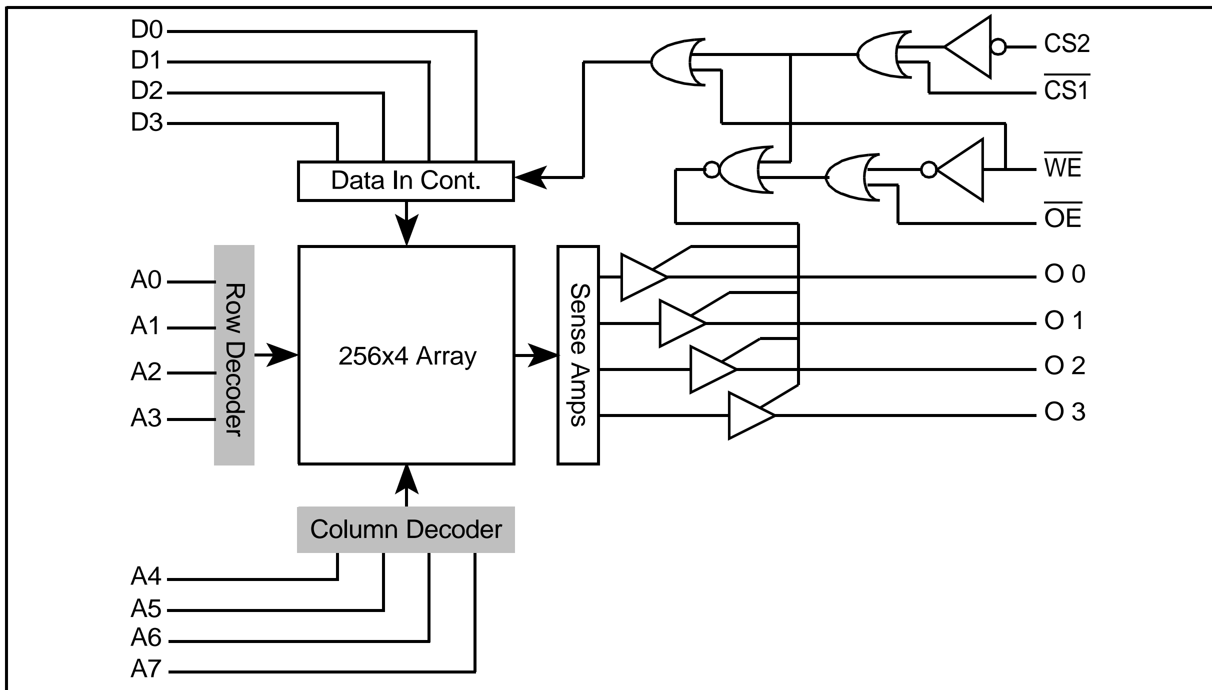

Figure 2-10 256x4 Static RAM Block Diagram - Addressing

Memory addresses are comprised of two parts; Row and Column. Although the Row address, also known as the X address, implies a horizontal direction. There is no standardization that forces us to relate Row, X, and Word lines. Different companies use different semantics and nomenclature. In this manual X is equivalent to Row and they run horizontally through the array.

The next several slides describe the addressing scheme for memory devices in detail.

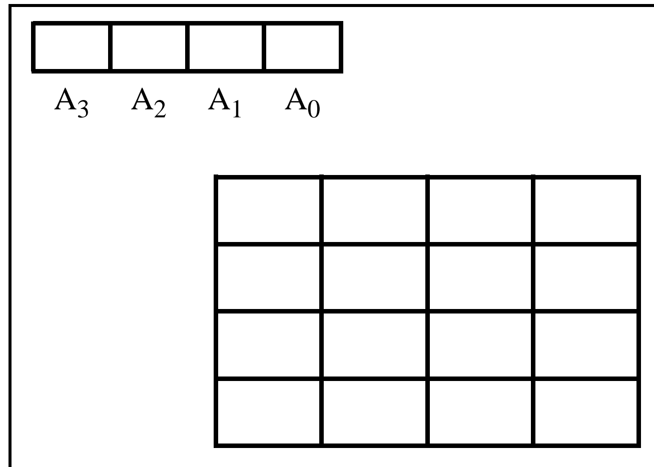

Figure 2-11 Addressing a 4 x 4 Array

The array shown in Figure 2-11 can be depicted by two X address lines (bits) and two Y address lines (bits). If we combine all four lines or bits into a single register, as shown at the top, we can use one binary four bit number to designate a specific address. We can use other numerical bases, besides binary. Hexadecimal is often used as a shorter, more efficient expression of binary numbers.

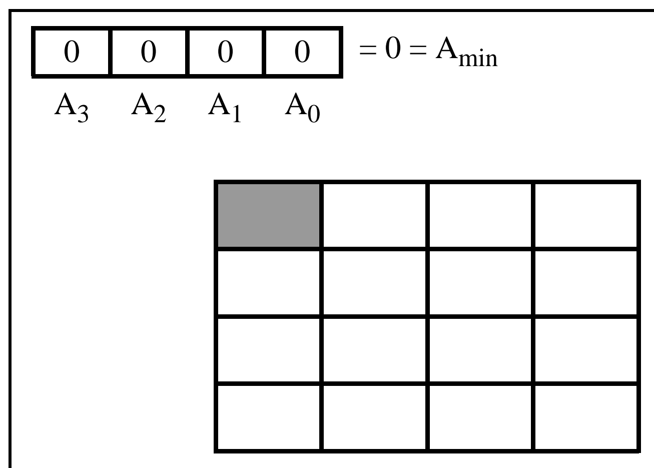

Figure 2-12 Minimum Address

Address zero, which can also be referred to as address minimum, could be designated as the highlighted cell as shown in Figure 2-12.

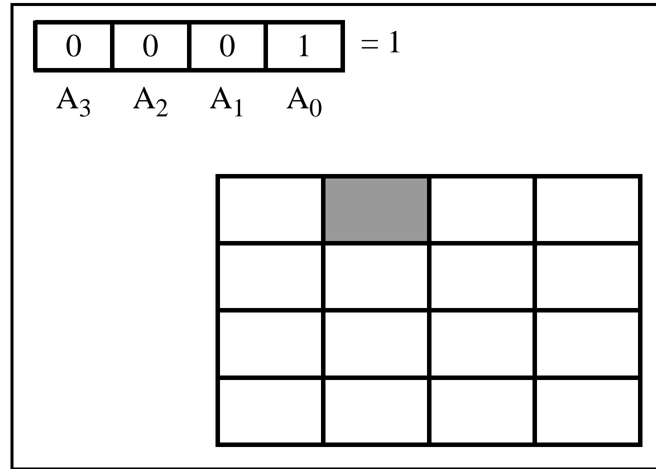

Figure 2-13 Address Sequencing - Column

As we toggle the least significant bit of the address register from zero to one we would step to the next address in the binary sequence. As represented in Figure 2-13 this could be seen as incrementing to the next column. Toggling the A2 from a zero to a one can be seen as incrementing to the next row as seen in Figure 2-14.

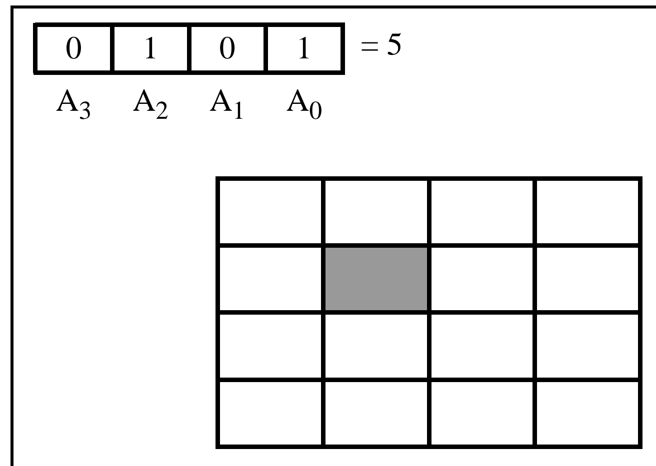

Figure 2-14 Address Sequencing - Row

We could assign each bit in the address register a name. These bit names would match the pin names given on most memory devices, A0-A3 in the example in Figure 2-14.

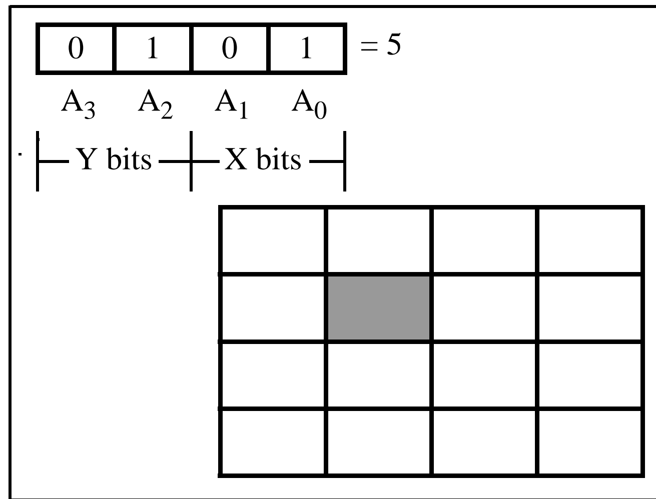

Figure 2-15 Address Sequencing - X and Y Assignments

We will also need to be able to refer to the same bits in the address registers by the names used inside the tester. In order to accomplish this we need to define the direction of X and Y. Although, the direction is really an arbitrary decision, the number of bits grouped into X and the number of bits grouped into Y, is not. They must match the number of cells corresponding to the rows and columns.

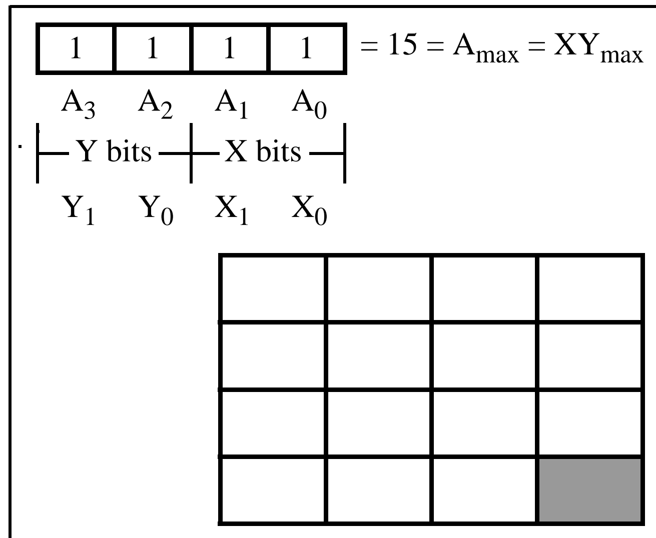

Figure 2-16 Address Sequencing - Maximum

If we assume X to be in the horizontal direction, A0 and X0 would refer to the same bit in the address register. Y1 and A3 would refer to the most significant bit in the register.

Figure 2-16 shows that with all address bits high the maximum address is accessed. This address is also known as XY Maximum.

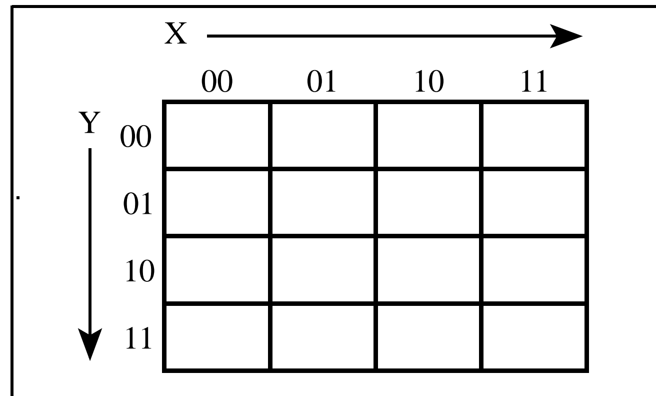

Figure 2-17 Addressing - X & Y

For documentation purposes, in this book we will use the style shown in Figure 2-17. When speaking about a specific address, we will refer to it as if it is a single register. We will usually refer to it in hex.

Addresses are really two separate registers. One for the X addresses and one for the Y. They can be tied together so they look like a single register when referenced from inside our test program.

Note that either the X address or the Y address can be the LSB's (least significant bits) of the complete address. If the X address is the LSB then it is called an X fast pattern because the LSB's change faster than the MSB's. If the Y address is the LSB then it is called a Y fast pattern. We can develop our pattern in either or both of these techniques.

Referencing our example device (Figure 2-17), we can see that there are specific address pins connected to row decoders and column decoders inside the device. It is possible to connect X address lines of a memory tester to either the row or column decoder pins with the Y address lines connected to the remaining decoder. As long as we know which address line goes to which decoder (column or row) on the memory device we will be able to trace a specific address back to a physical location within the memory device.

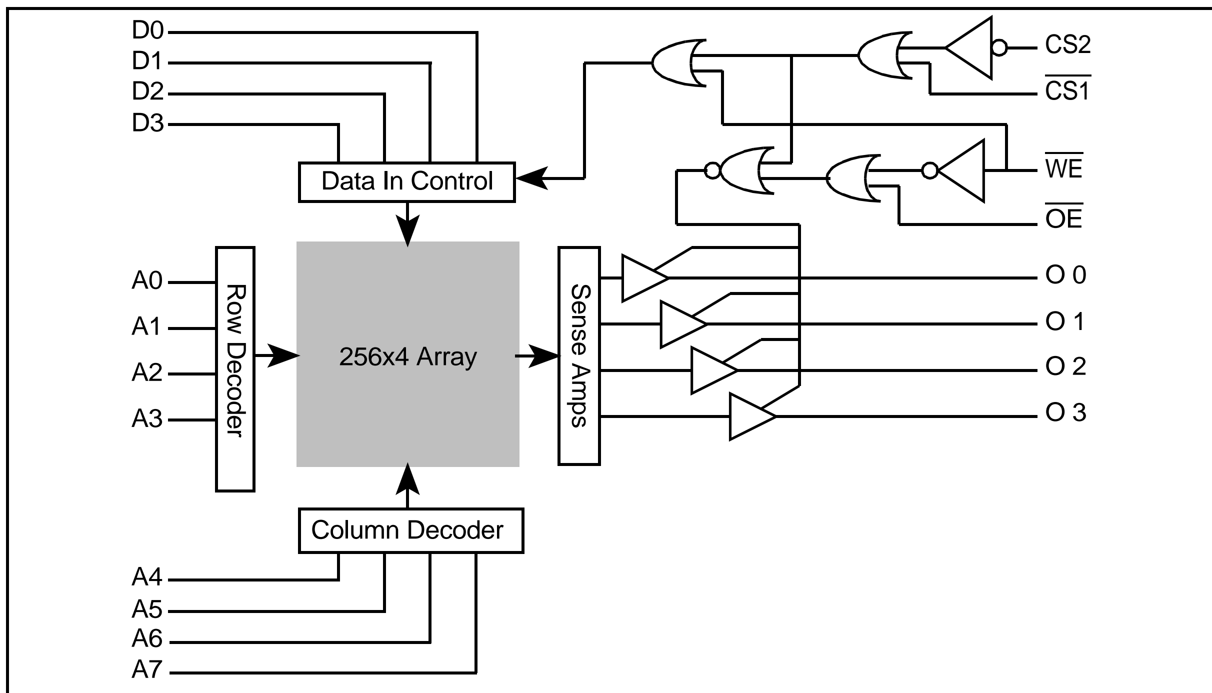

Figure 2-18 256x4 Static RAM Block Diagram - Bit Lines

Inside the array we don't really refer to X or Y, or even to row or column. We refer to word lines and bit lines.

As soon as a choice like X or Y connections becomes arbitrary we open the door for confusion. Row and column definitions add to that confusion because their meaning is not clear.

One decoder (row or column) will select an entire group of cells while the other will select which cell in the group is selected when reading or writing.

Address Faults

- When two or more addresses are electrically connected.
- Sometimes NOT obvious.
  - Usually shows up as row or column problems.
- Can be tested by proving address uniqueness.
  - i.e. Galpat pattern (discussed later).

With an understanding of what addresses are and how they work, we can start to understand how they can fail. The key is to test for address uniqueness. Each address must access one, and only one, cell within an array.

If we write the array to all ones as we have mentioned earlier, how many cells must function in order for us to be able to read back all ones? That’s right, one single cell. All addresses could be shorted together and we would still pass the all ones and also the all zeroes patterns.

Output Buffers

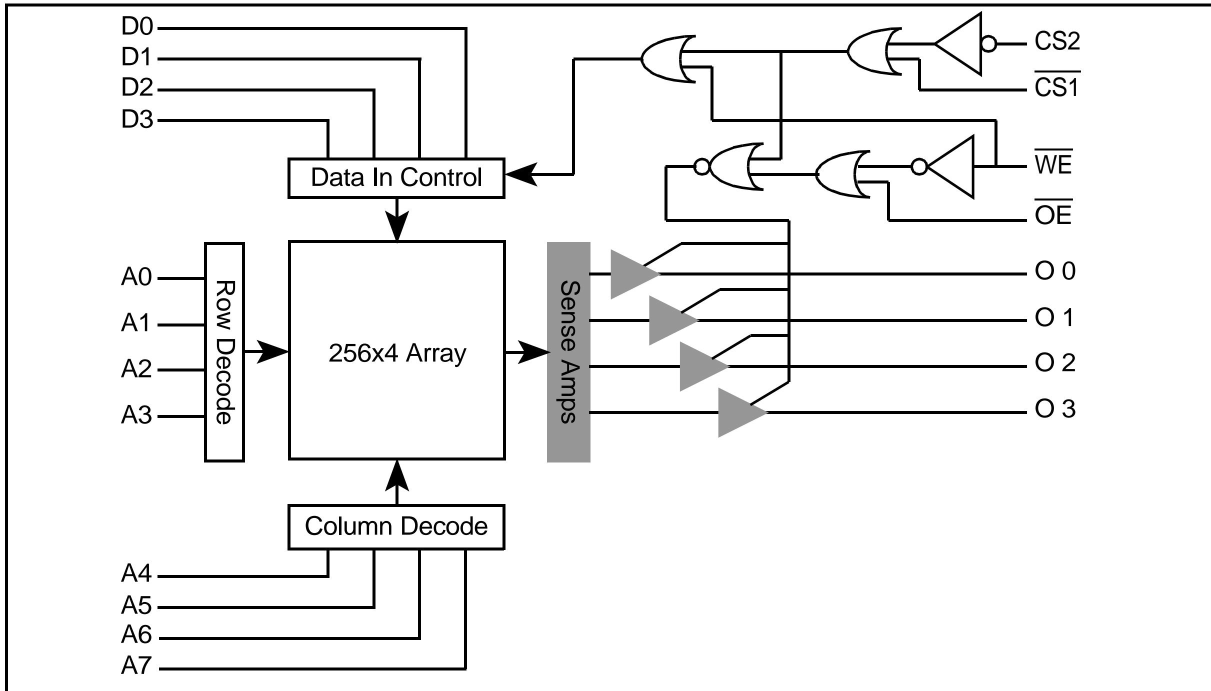

Figure 2-19 256x4 Static RAM Block Diagram - Sense Amps

Cells are really transistors (in the case of a DRAM they are a single transistor) and silicon real estate is the most expensive in the world, we try and shrink transistors as much as possible. It is easy to understand that the cells have very little drive capability. Therefore we need sense amps to be able to recognize the very small voltage swings and translate them into levels that can be distinguished as logical ones and zeroes.

Once the sense amps have ‘amplified’ the voltages, the output buffers can then provide current drive capability sufficient to use the device in a bus oriented environment.

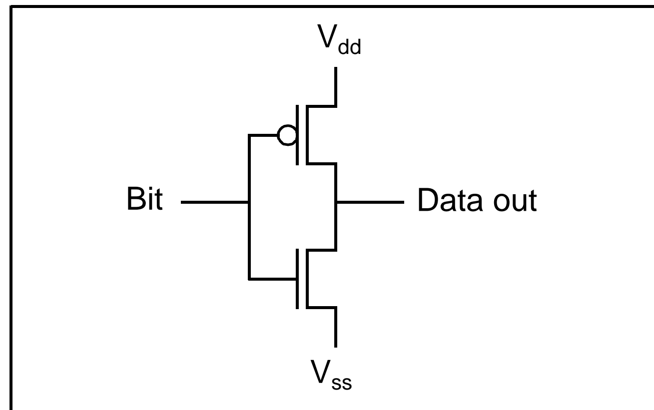

Figure 2-20 Output Buffer Basic Circuit

Figure 2-20 illustrates an output buffer with a totem pole design, named for the way the two transistors are stacked on top of one another. It is the most common output structure for MOS devices.

The waveform that data out produces is the same as that on the bit line, but with added drive capability supplied by the two transistors shown. The output current capacity (IOH and IOL) is determined by size of these two transistors.

In most cases we have the ability to turn off both transistors using the Output Enable (OE) or the Write Enable (WE) pin. This feature is not shown in Figure 2-20.

Other types of output buffers include open drain and open collector. Both of these require an external circuit (usually just a resistor) to provide a pull up, or pull down, voltage.

Transition Faults

- When a cell or line fails to undergo a 0 to 1 or a 1 to 0 transition.
- Less easily identified than a Stuck at Fault, but detected in much the same way.
- Write and read every cell to both data values.

This type fault might have exactly the same symptoms as a stuck at fault, but the actual flaw in processing is in a completely different physical location on the die.

An example would be when a Bit Line is charged to a high voltage in order to write data into an EPROM or FLASH cell. If the bit line was accessed to read/write the opposite data before the discharge was complete the 'wrong' data would be presented on the output of the device. But if discharge time had expired the Bit Line would work fine.

Once again in English. If a cell is stuck, or coupled, the erroneous state does not change based on time, but based on some other action the user takes (writing a different cell or trying to write different data). With a transition fault, the fault can only be seen based on specific, time related, conditions. The cell fails until after a certain amount of time has expired.

Control Logic

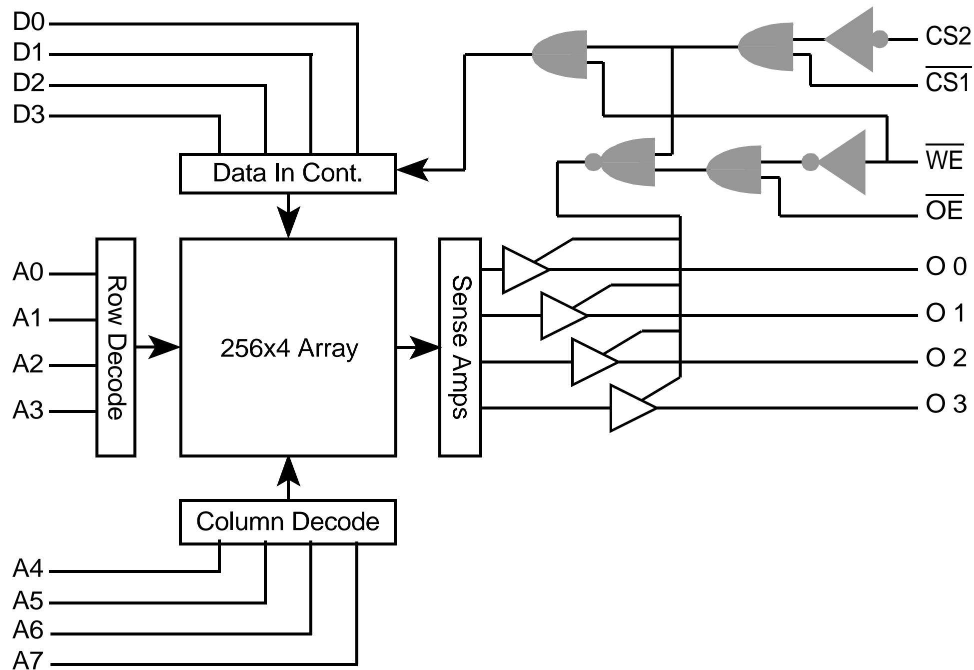

Figure 2-21 256x4 Static RAM Block Diagram - Control Lines

If we over simplify the array as a group of cells that hold data until we need it again, we can use addresses as a way to point to a specific cell, but we still need a way to decide whether we are putting data into a given cell, or reading data from a given cell.

Obviously, the $\overline{\text{Write Enable}}$ ($\overline{\text{WE}}$) pin allows us to write into the location that is applied to the address pins when the WE pin is true (low in this case).

The $\overline{\text{Output Enable}}$ ($\overline{\text{OE}}$) turns on the output buffers and presents data to the outside world.

The Chip Selects (also called Chip Enable on some devices) are really on/off switches. They allow us to select or deselect (turn on or turn off) this specific device in its entirety. This is useful to conserve power when we know the device won’t be used for a long time (e.g. several seconds). Deselecting the device will not destroy the data that has been stored in the array (as would happen if power were removed) but it will turn off the circuitry used for everything else (control pins, decoders, etc.) and thus reduce the power consumption of the device while in “stand-by” mode.

Logic Tables

$\overline{OE}$	$\overline{CS1}$	CS2	$\overline{WE}$	D0-D3	O 0-O 3	
X	H	X	X	X	Z	Not Selected
X	X	L	X	X	Z	Not Selected
L	L	H	H	X	L/H	Read Stored Data
X	L	H	L	L	Z	Write a zero
X	L	H	L	H	Z	Write a one
H	L	H	H	X	Z	Output Disabled

Figure 2-22 Sample Truth Table

Most device data sheets show a truth table of the control pins and most devices use the same control pin organization.

In Figure 2-22 it is important to remember that although the conditions marked with an “X” represent a ‘don’t care’ state for that pin, inputs to a CMOS device can never be placed in a high impedance state. If an input pin is allowed to float, the pin will most likely go to the threshold voltage of that input and oscillate just enough to cause the pin to switch at maximum frequency, finally resulting in latch up. Latch up can damage or destroy the device.

Write Timing

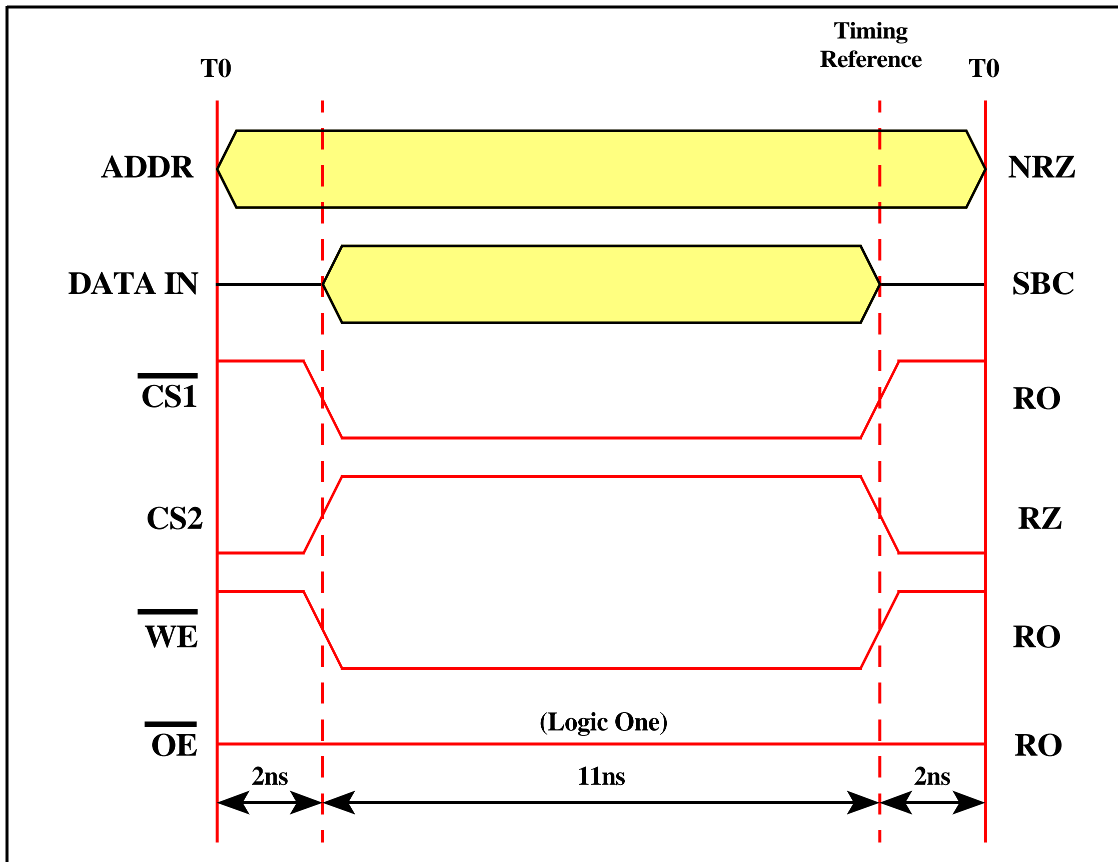

Figure 2-23 Write Timing

As we transition from the device block diagram to the patterns, let's consider what the control truth table in Figure 2-22 would look like on a scope for a write (and then read) cycle.

In Figure 2-23 the Address and Data In signals are drawn in such a manner as to indicate they are bus signals representing a group of pins.

The mnemonics on the right hand side of the diagram in Figure 2-23 indicate which format could be used to produce that waveform. These mnemonics have not been defined in this class since they are discussed in detail in the digital logic class.

The times listed across the bottom of the diagram are examples of what we might see on our example device.

Notice that the Data Out pins are not indicated during a write cycle. Typically they would present a high impedance (float) condition. It is acceptable for the device to present a high impedance to the outside world (on an output pin from the DUT) but it is unacceptable to present a high impedance to any input pin of a CMOS device.

Read Timing

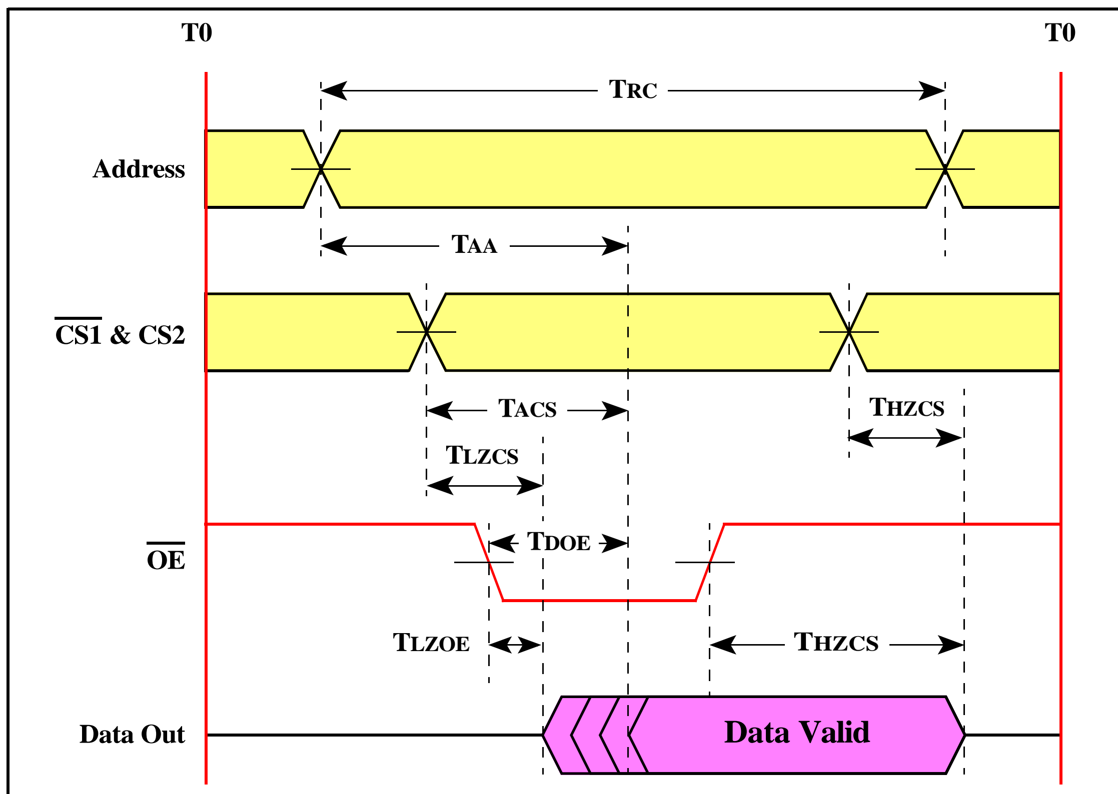

Figure 2-24 Read Timing

Unlike the timing shown in Figure 2-23, Figure 2-24 has device specifications (i.e. Trc is read cycle time) listed on it, just as you would see in the data book.

The write signal isn't included in the drawing since it is expected to be at a logic one level (false) during any read operation.

Waveforms are a good communication medium between engineers. When combined with good pattern documentation, timing drawings can describe exactly what is happening in a test.