


Introduction to CMOS VLSI Design (E158)

Harris

Lecture 11: Memory

David Harris

Harvey Mudd College

David_Harris@hmc.edu

Based on EE271 developed by Mark Horowitz, Stanford University

Memory

Reading

W&E 8.3.1 - 8.3.2 - Memory Design

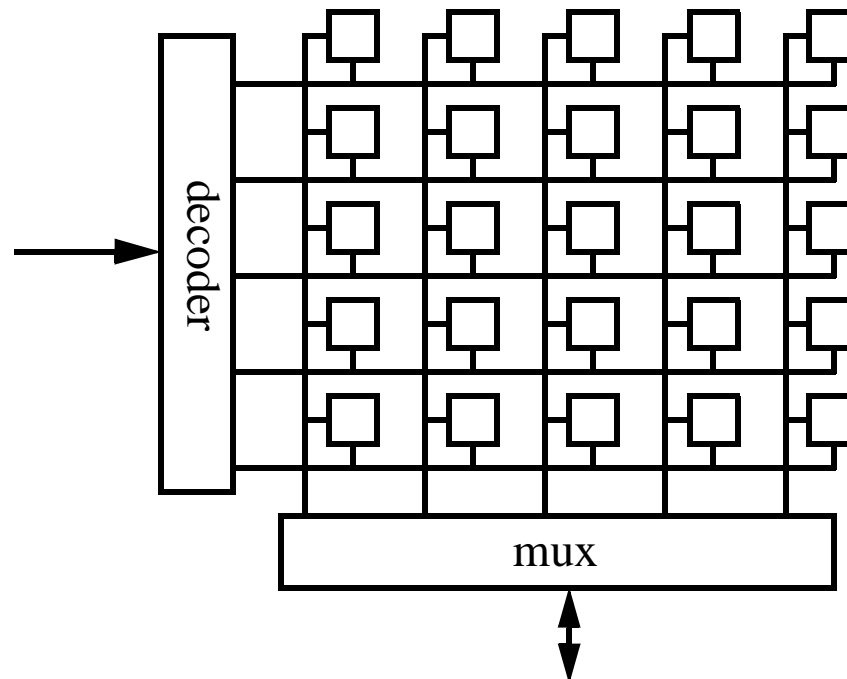
Introduction

Memories are one of the most useful VLSI building blocks. One reason for their utility is that memory arrays can be extremely dense. This density results from their very regular wiring.

Memories come in many different types (RAM, ROM, EEPROM) and there are many different types of cells, but the basic idea and organization is pretty similar. We will look at the most common memory cell that is used today, a 6T sRAM cell, and then look at the other components needed to build complete memory system. We will also look at other types of memories.

Best Case for Structured Wiring

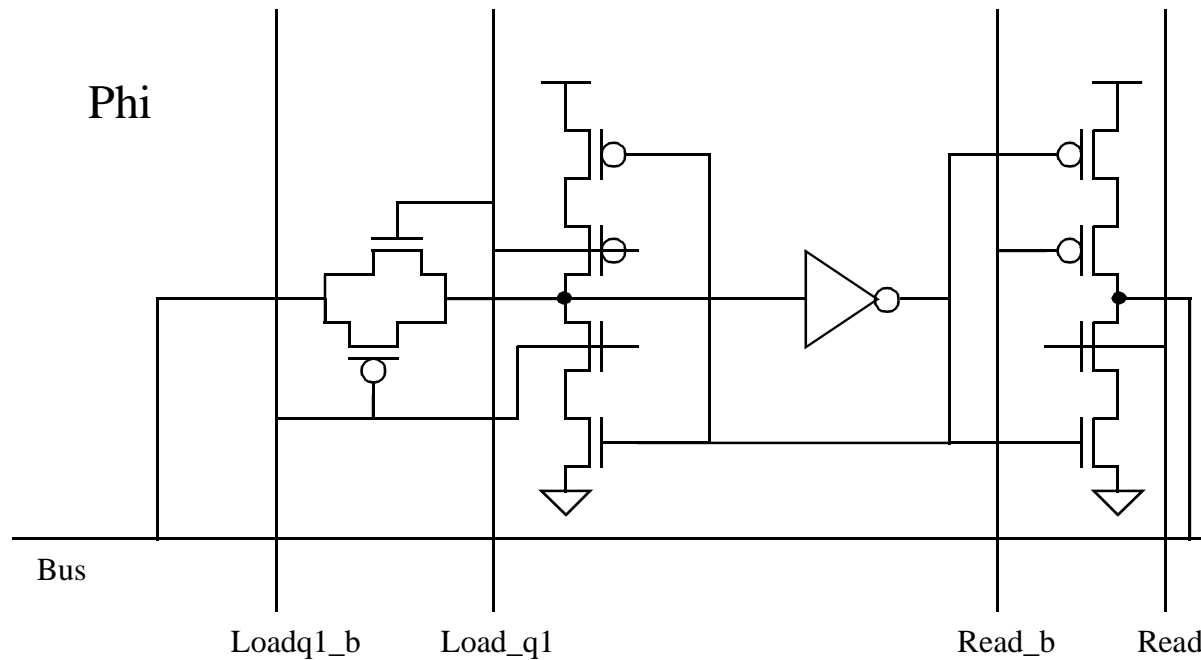
Best example is a memory array:



It has N^2 elements and only $2N$ wires. It is an easy way to use 1M transistors. The layout is quite dense.

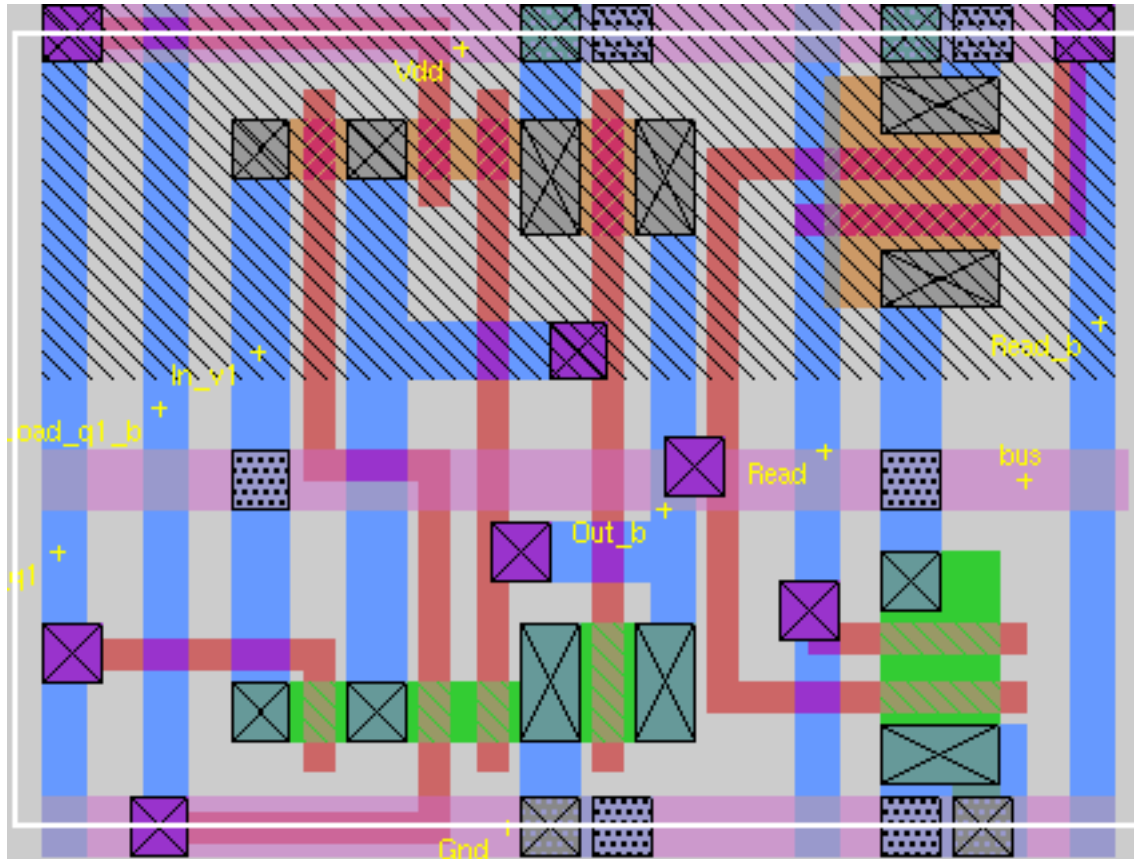
Basic Memory Cell

Could use a basic latch cell:



- But the cell needs to be a static latch
- Needs a large number of control wires to be able to read and write the cell. These wires (and transistors) make the cell pretty large.

Latch Cell Layout



- The cell is 78 wide by 55 tall, the top and bottom rails can be shared
- Notice that the wires take all the space

Small Memory Cell

Often need to have a large number of bits stored:

- In some cases more bits are better
- Willing to take some time to optimize cell
- Have enough cells it is ok to make peripheral circuits more complex

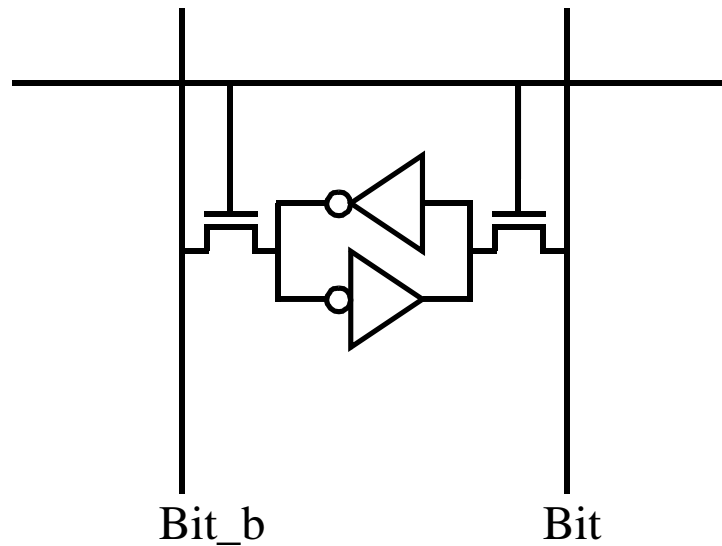
Lead to many innovative cell designs

- 6T RAM cells
- 4T RAM cell with poly loads
- 1T DRAM cell
- And lots of strange layouts

We will look at the 6T RAM, which is the key to all memory cells

Static RAM

Uses only six transistors:



Read and write use the same port. There is one wordline and two bit lines. The bit lines carry the data. The cell is small since it has a small number of wires.

SRAM

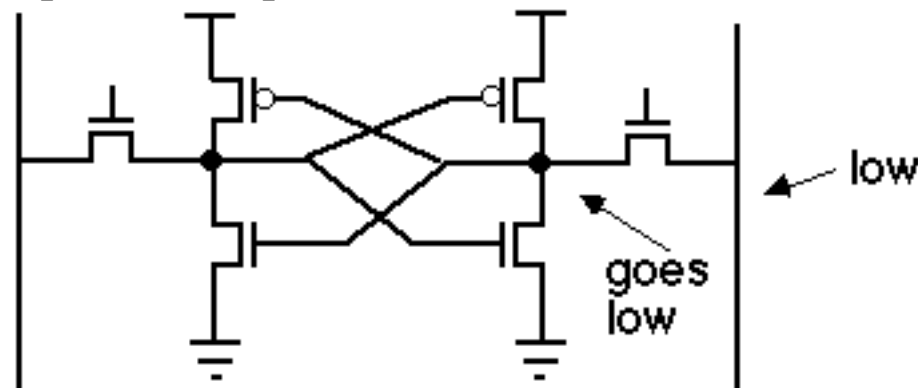
The key issue in an 6T SRAM is how to distinguish between read and writes. There is only one wordline, so it must be high for both reads and writes. The key is to use the fact there are two bitlines.

Read:

- Both Bit and $\overline{\text{Bit}}$ must start high. A high value on the bitline does not change the value in the cell, so the cell will pull one of the lines low

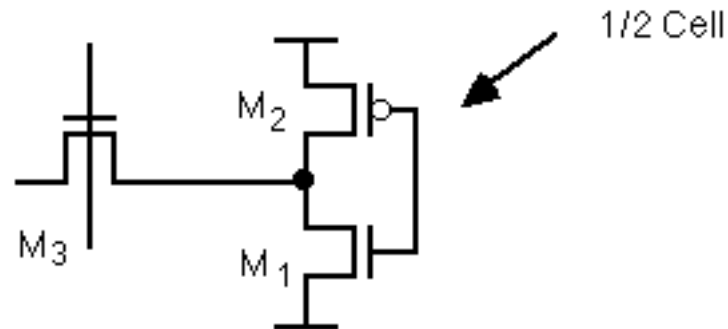
Write:

- One (Bit or $\overline{\text{Bit}}$) is forced low, the other is high
- This low value overpowers the pMOS in the inverter, and this will write the cell.



SRAM Cell Design

For the cell to work correctly a zero on the bit line must overpower the pMOS pull up, but a one on the bit line must not overpower the pull down (otherwise reads would not work)

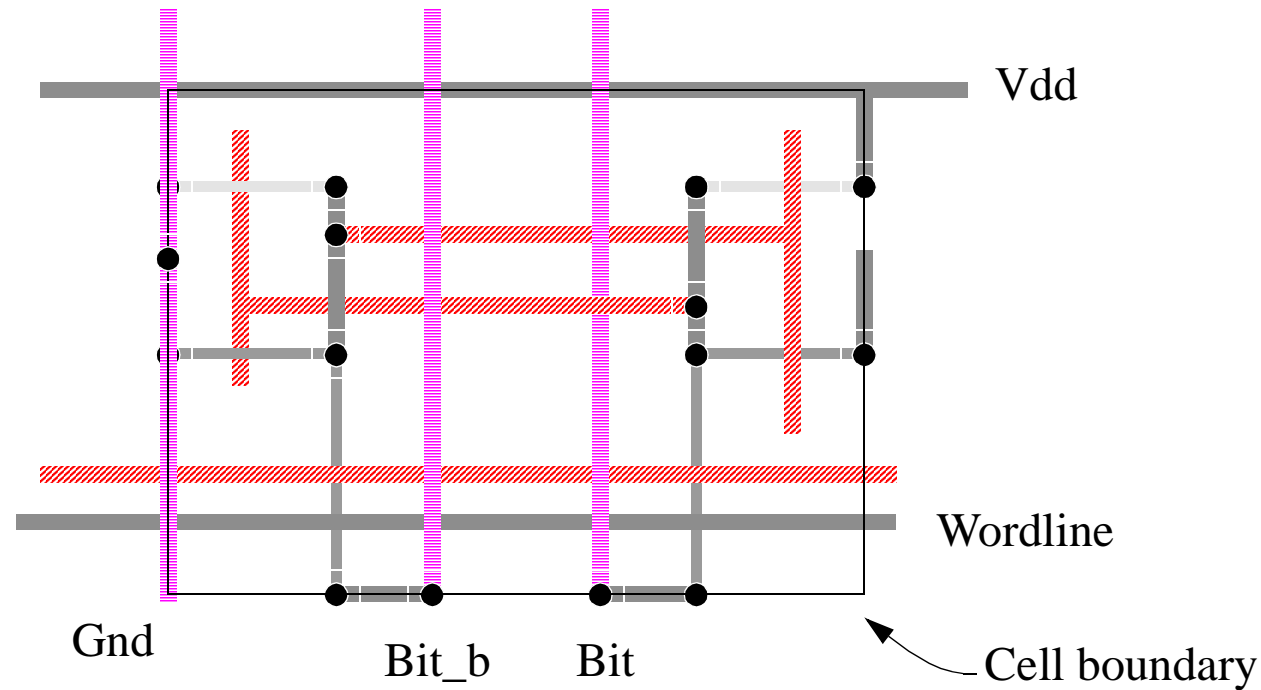


For the pull down M3 is passing a zero, so for it to overpower the pMOS it must be at least as wide (preferably 1.5x as wide). This gives a 2-3:1 current ratio between the nMOS and the pMOS.

For pull up M3 is passing a one so it is somewhat weaker. Still M3 should be 1.5 to 2x smaller than M1 to make sure a read does not disturb the value of the cell.

SRAM Layout

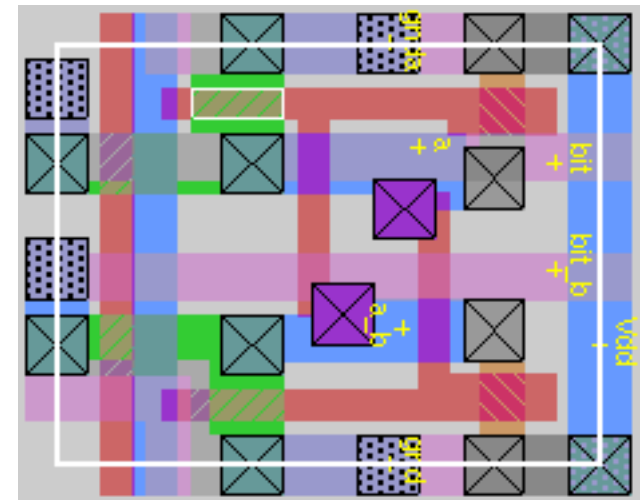
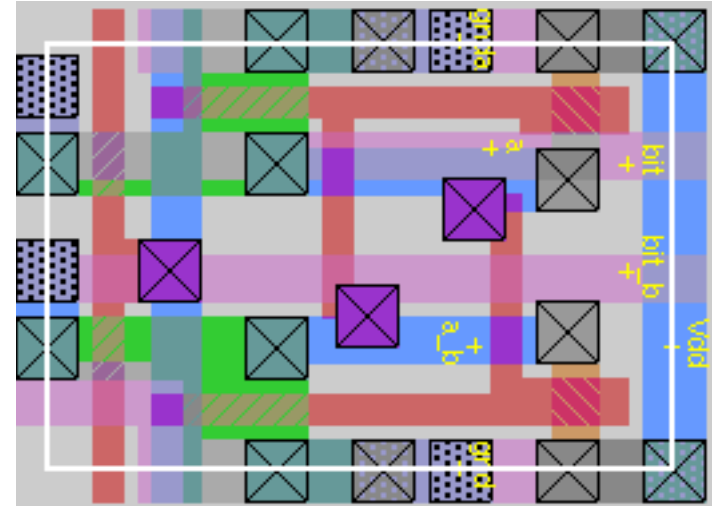
There are many clever SRAM layouts. This is a common one:



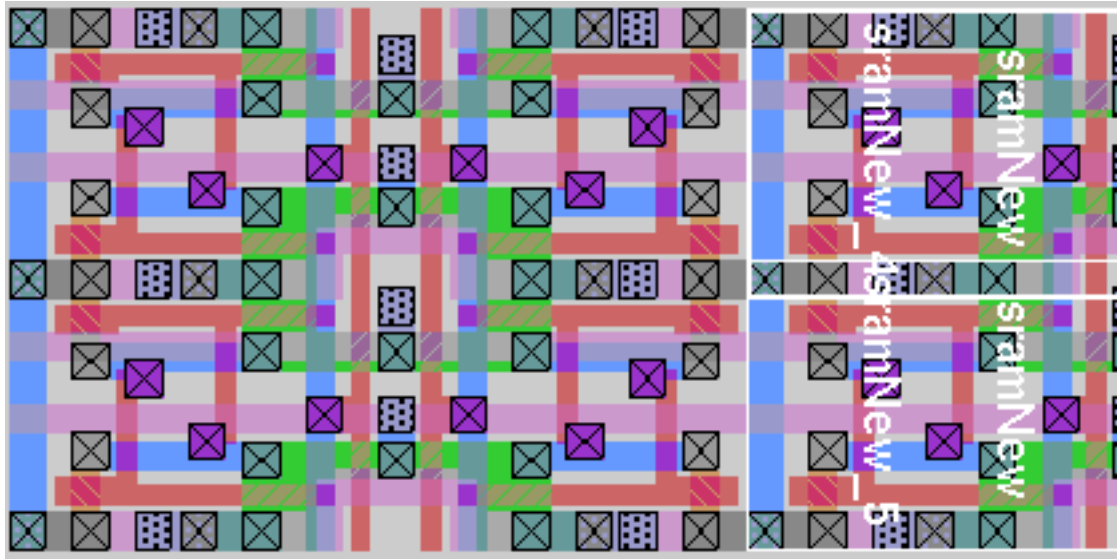
This layout is fairly dense, since the most of the contacts (bitline, Vdd, Gnd) are shared. Also the a clever cross-coupling method is used.

SRAM Layout

- A conservative cell:
 - It has substrate and well connects in each cell
 - It has a wordline poly contact in each cell
 - pMOS transistors are very weak (3:3)
 - nMOS pulldown is 8:2
 - All the boundaries are shared
 - 41 x 28, about 1/4 the size of latch cell
- A slightly smaller cell
 - Only nwell contact in cell
 - pMOS transistors are very weak (3:3)
 - nMOS pulldown is 6:2
 - 36 x 28
- White box is the repeat box. Cells overlap contacts



SRAM Array

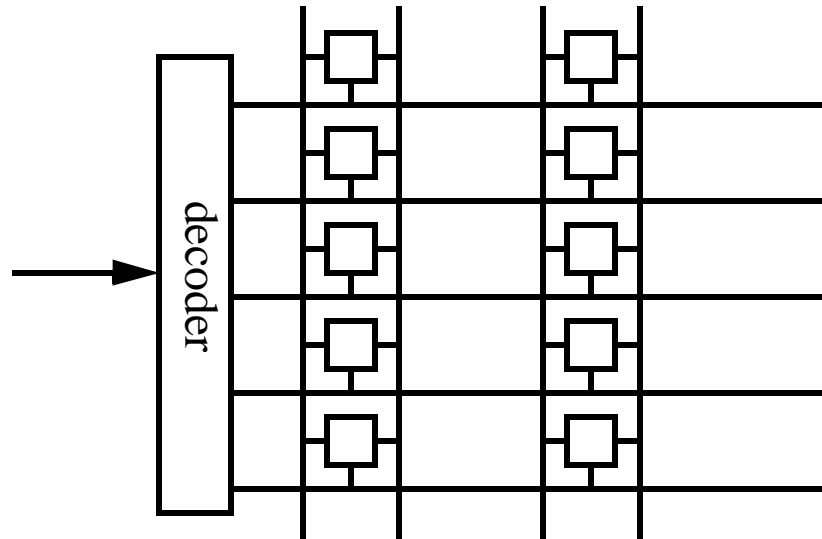


This is an array of 3 cells wide and 2 high.

- Shows how the contacts are shared in both X and Y

Memory Array

Lets look at how this cell works in an array:



The decoder selects one cell on each set of bit lines. All the other wordlines are low, disconnecting those cells from the bitlines.

If you don't need to read all the bits at once, you can add a mux to combine the bitlines into fewer IO lines.

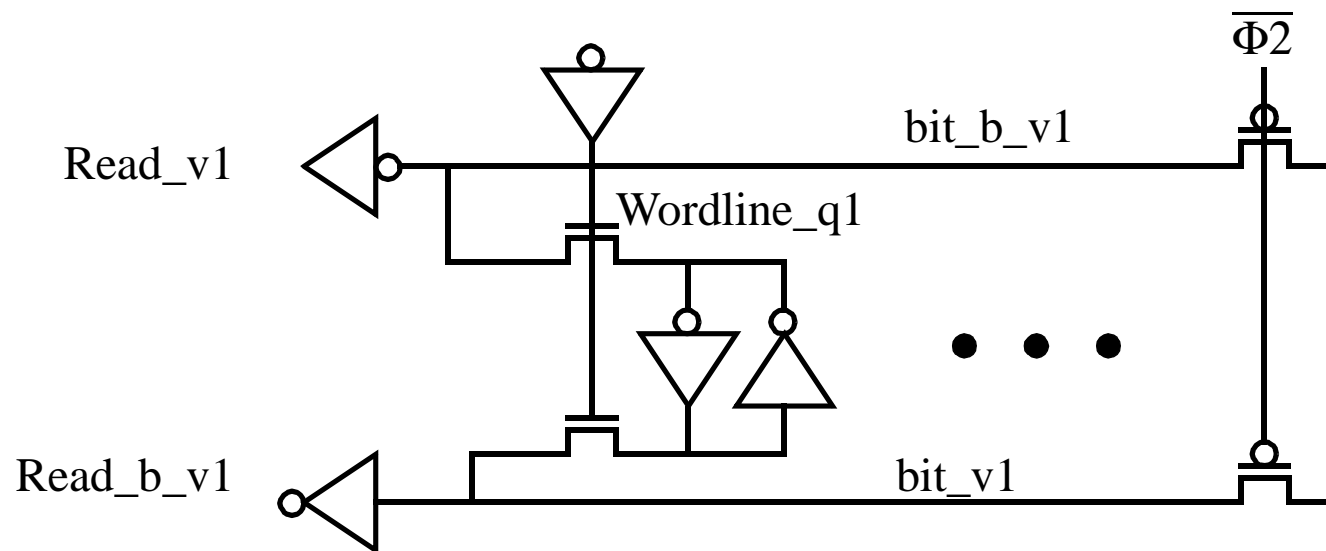
Bitline IO Circuit - Read

For reads both bitlines must be high, for write you need to drive the bitlines to the correct value.

- Bitlines need to be precharged, or use a pseudo nMOS load

We will use a precharged structure:

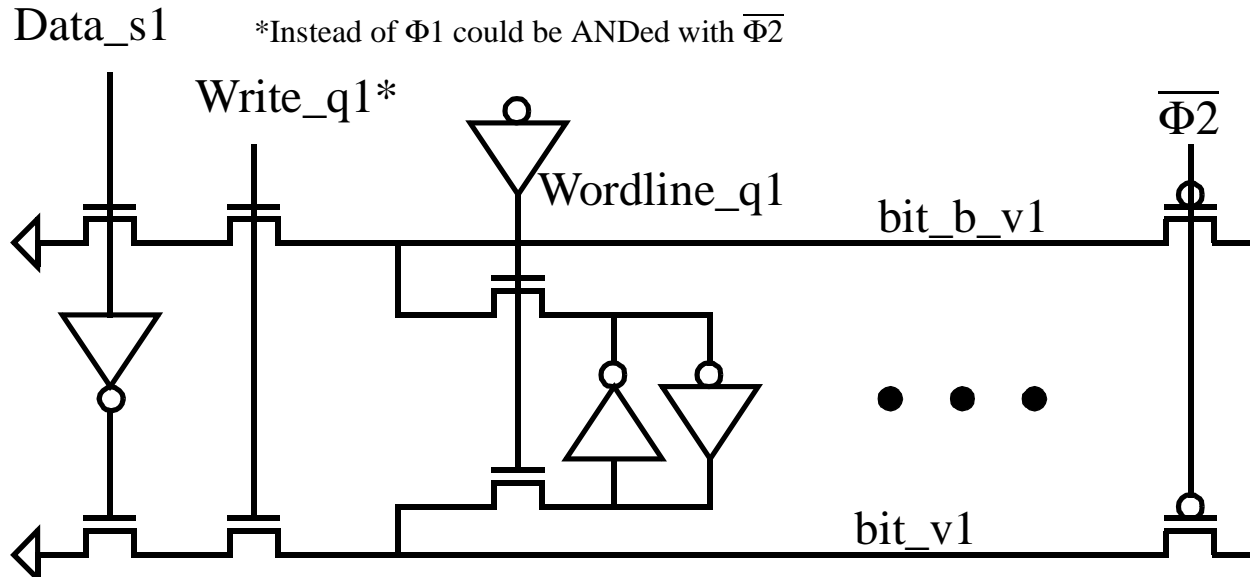
- To avoid a conflict during precharge, make wordline a qualified clock.



- Bitlines are like the outputs of normal precharge gates - `_v` signals

Bitline IO Circuit - Write

Similar to read, but need to drive the bit lines too.



It is safer if the write drivers are complete tristate buffer (had a pullup device too) rather than just pulldowns. This will allow the driver to pull up a bitline that was partially discharged by the cell (if the wordline rises before the write signal)

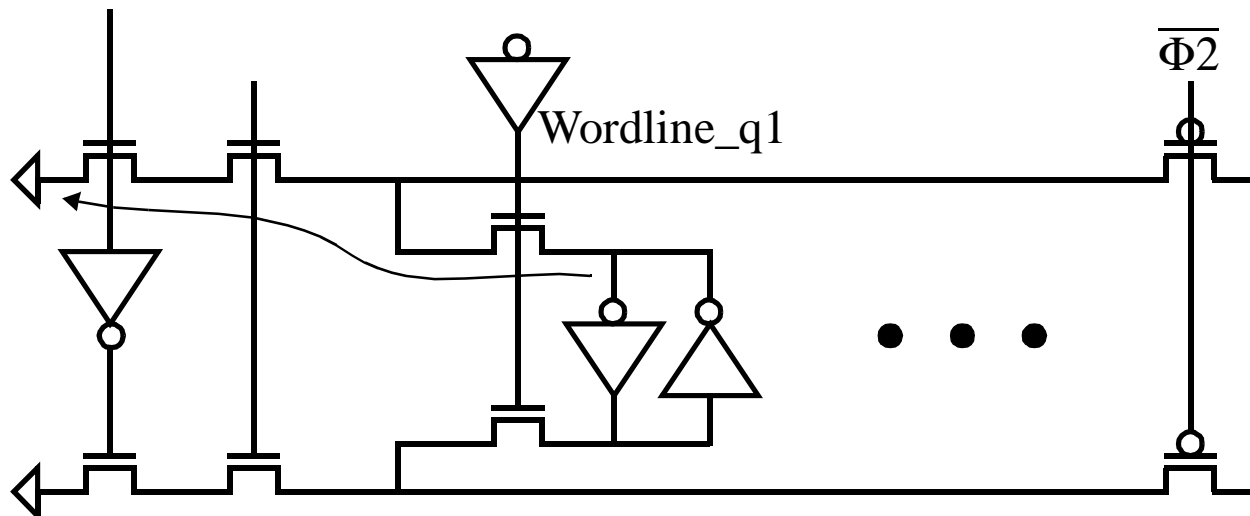
Notice that since the memory cell is a storage element, its enable (the wordline) needs to be a $_q$ signal. That will ensure that the clock falls latching in the data BEFORE the data has a chance to change. The wordline is really the clock to the latch (memory) cell.

Need to isolate the write driver so it does not fight with the precharge (power issue), which is why the write signal is qualified.

Write Drivers

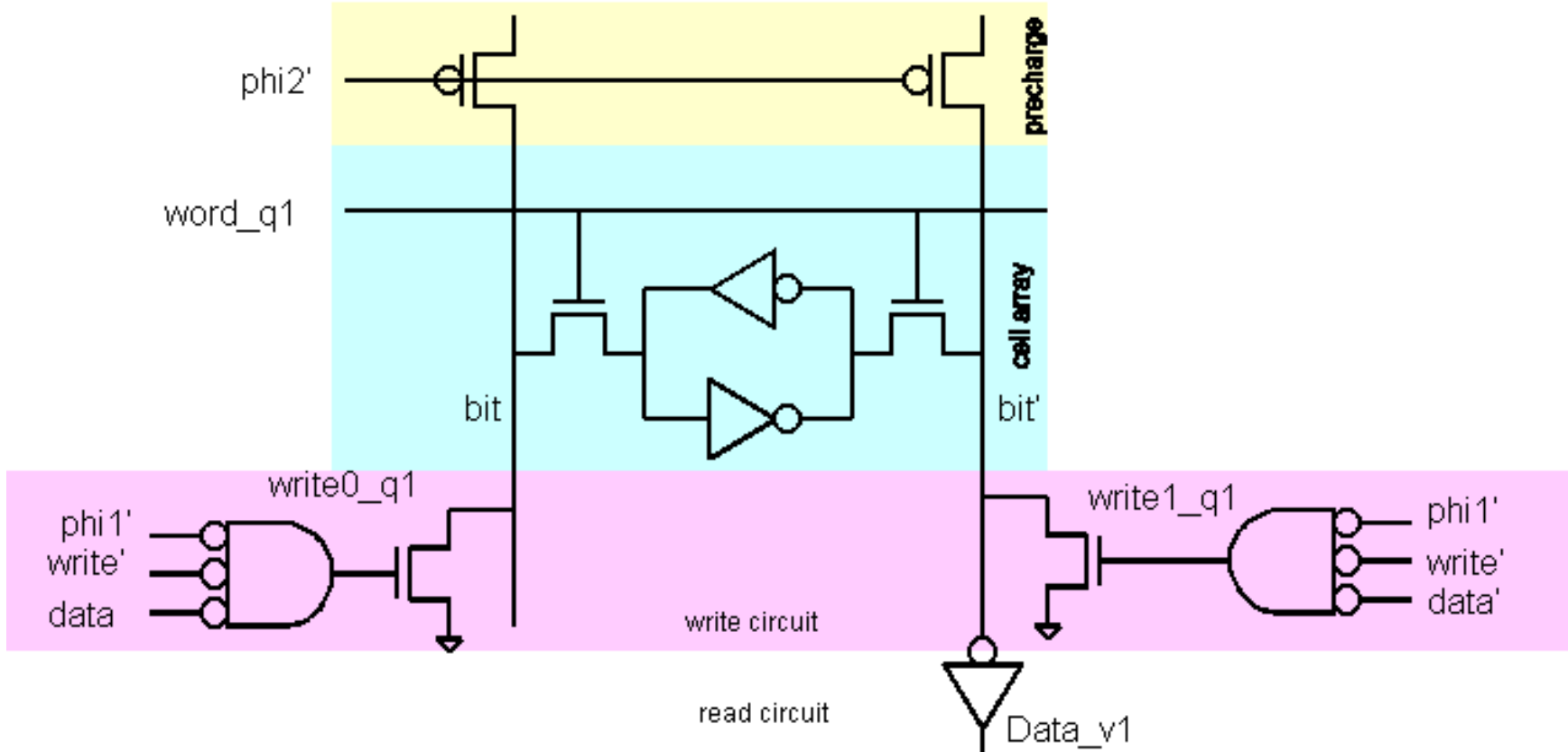
Also need to worry about the series resistance of the driver

- The resistance of the 3 series nMOS transistors must still be 2x less than the resistance of the pMOS in the cell



- If the pass device in the cell is 4:2, and the pMOS load is 3:4, then each nMOS in the driver must be at least 8:2. If the pMOS was 3:2, it would be hard to get the cell to write.

Write Alternative



+ Bitline Muxing

The bitline pitch is pretty small (about 28λ) and there is a lot of stuff that is needed for the bitline (read and write circuits). Often many bitlines are muxed together, and one set of IO circuits is used for these bitlines

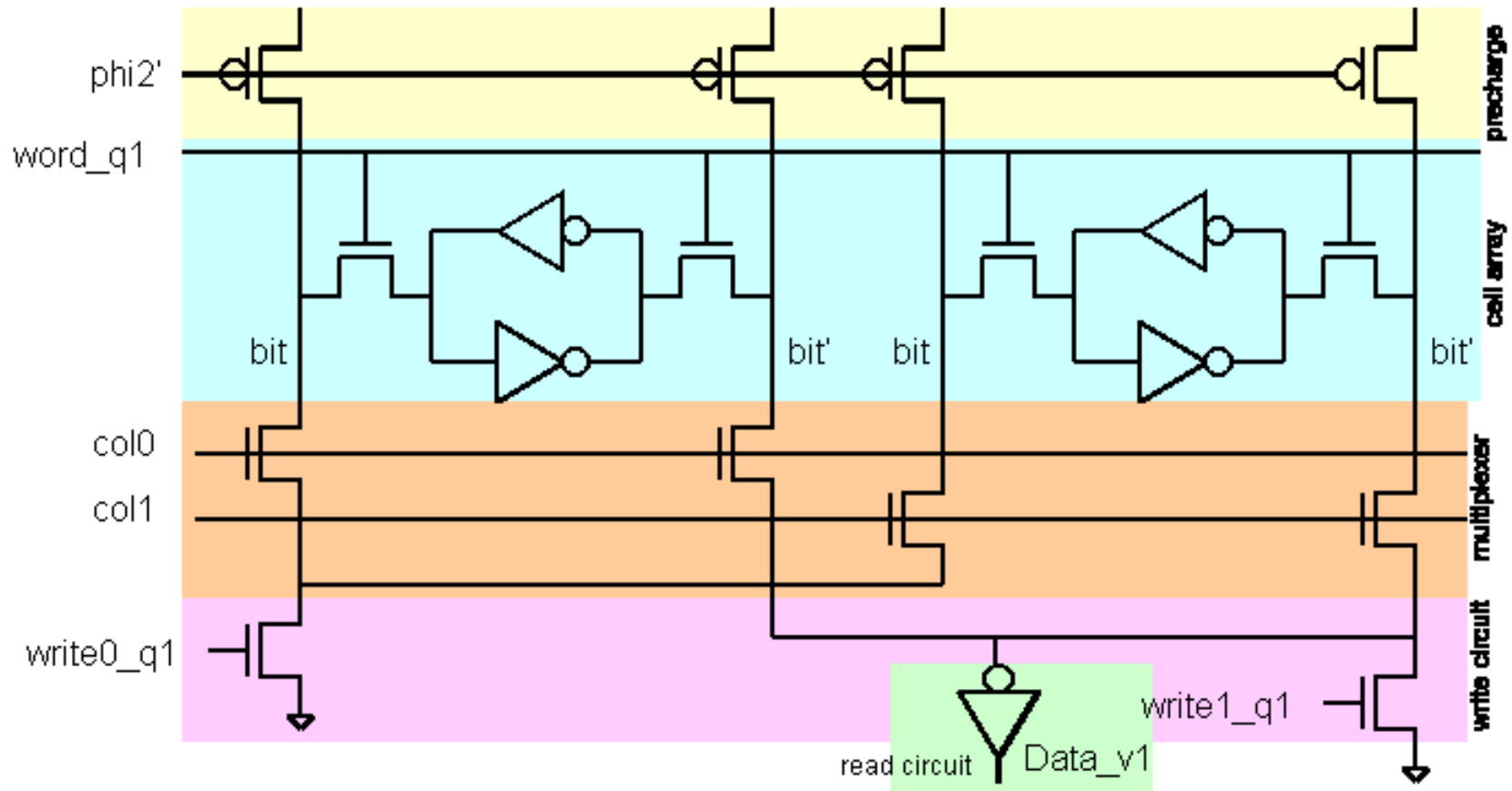
Two basic options:

- Share mux between read and write circuits
 - + Least amount of logic needed on bit pitch
 - Adds another series device to write drivers, need to use single write device
- Use different mux for read and write
 - + No series devices in write path

Write mux can be qualified with Write & Clock

- Adds some more muxes to bit logic

+ Bitline Mux, Option 1



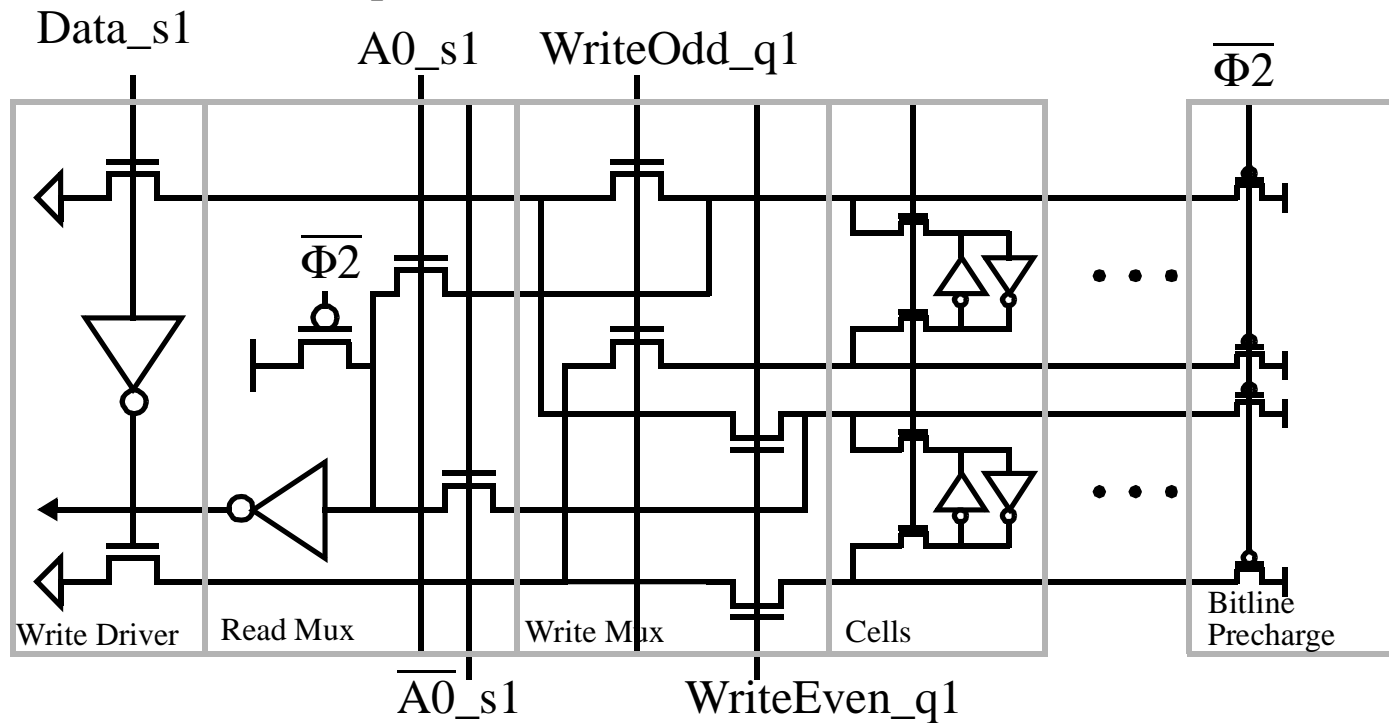
Should have a precharge on the output of the mux too, since otherwise the output will have a degraded high level.

+ Muxing Example

Uses separate mux for read and write

Notice that the read mux is precharged

- You don't need to use pMOS devices in the mux



+ Sense Amplifiers

- Since $C_{diff} \sim C_{gate}$, the diffusion contacts of access devices cause large cap on the bitlines, for large arrays
- Bitline cap becomes an issues around 32 cells/bitline

Example:

If the diffusion contacts are shared (adjacent cells), 128 cells @ $4fF/2cells = 256fF$ + wire cap. This would lead to access times of around 3ns.

Can take advantage of the differential nature of the bitlines

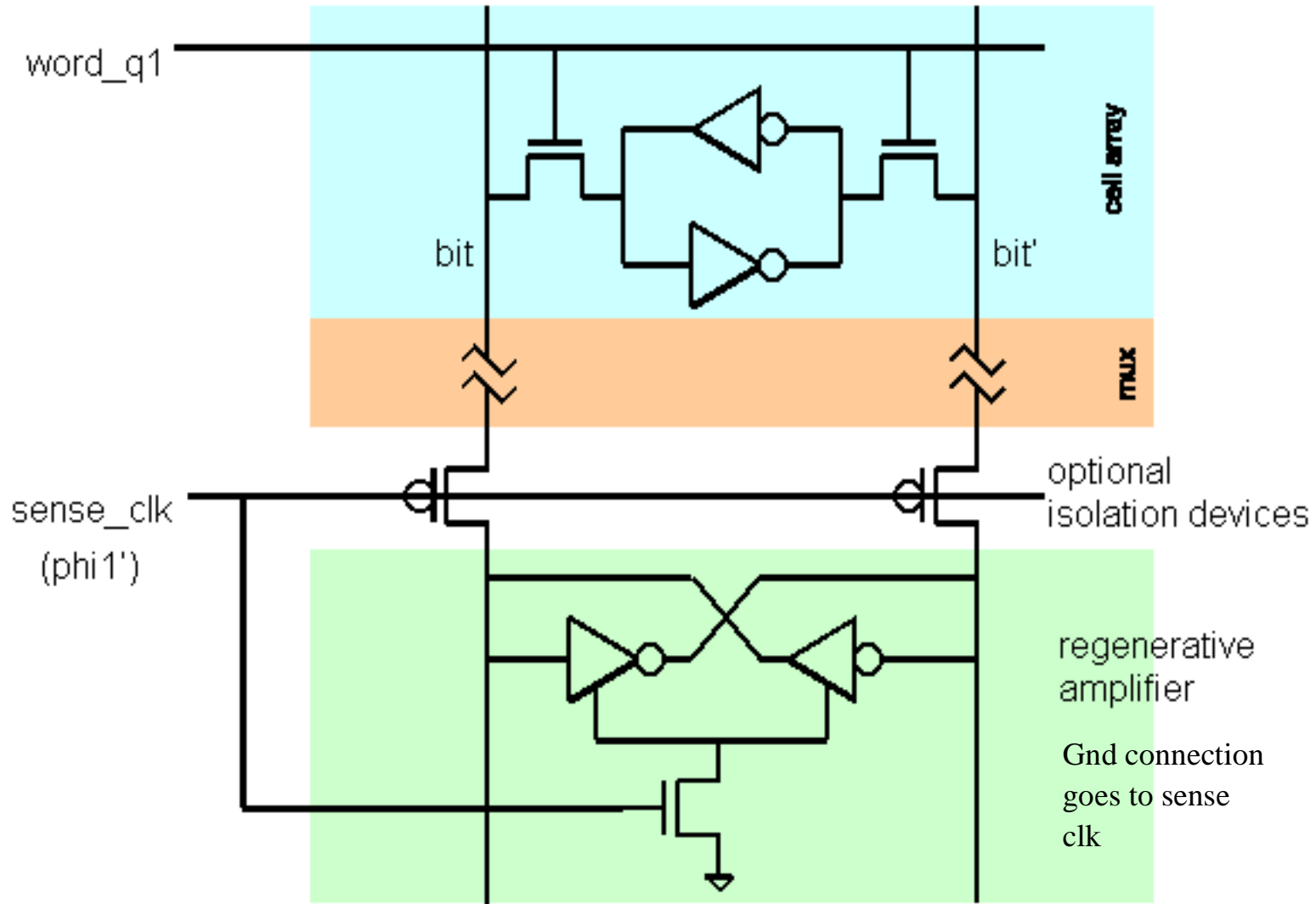
- Decrease delay by sensing smaller signals

Noise margin is ok, most noise is common mode

Build a differential amplifier (sense amp)

(Not needed in this class)

+ Sense Amps



+ RAM Variations

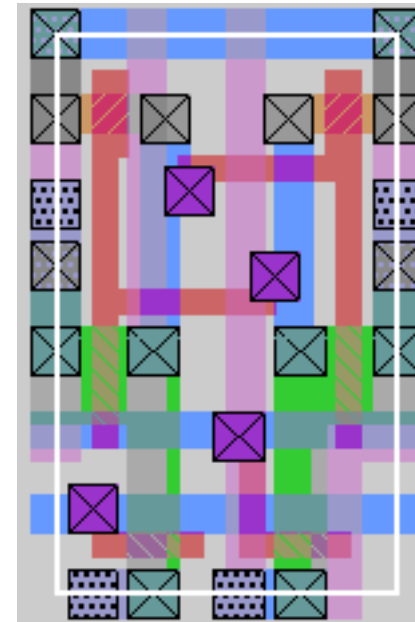
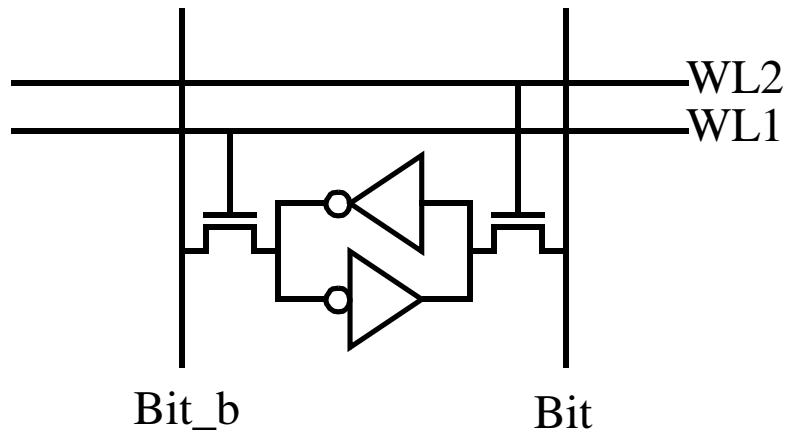
There are many variations to the basic 6T SRAM cell. Some are cells with more functionality, while others are smaller memory cells (that are dynamic, not static). We won't talk about them much in the class, but I will go through them in the notes, so you can see what is possible.

Options:

- Dual read or single write cell
- True multiported cells (for registerfiles, etc.)
- Content addressable cells (CAMs)
- 4T dynamic memory cell
- 3T
- 1T DRAM cell

+ Dual Ported Cell

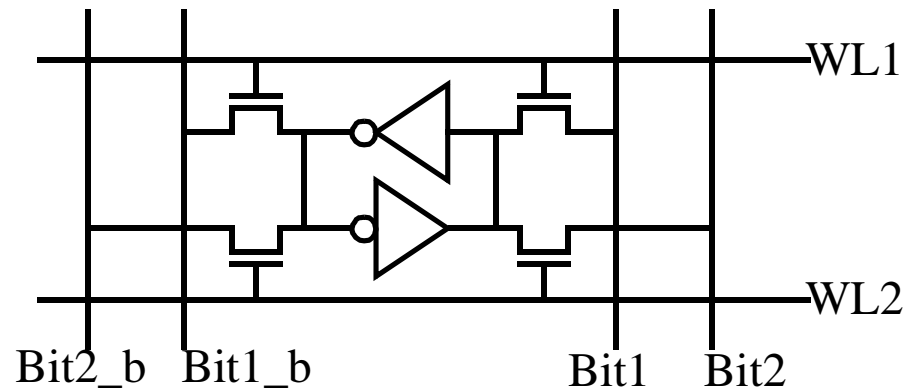
Split wordline so there are two wordlines, one for each pass transistor.



- Nearly the same size as SRAM, 46 x 28
- Can read two different cells in one cycle or perform one write.
 - Raise WL1 on Register 5, and WL2 on Register 7. Register 5 value will be on bit_b (complemented, of course), and Register 7 will be on bit. Since you need both bit and bit_b to write the cell, you can only do one write per cycle.

+ Multiported Memory

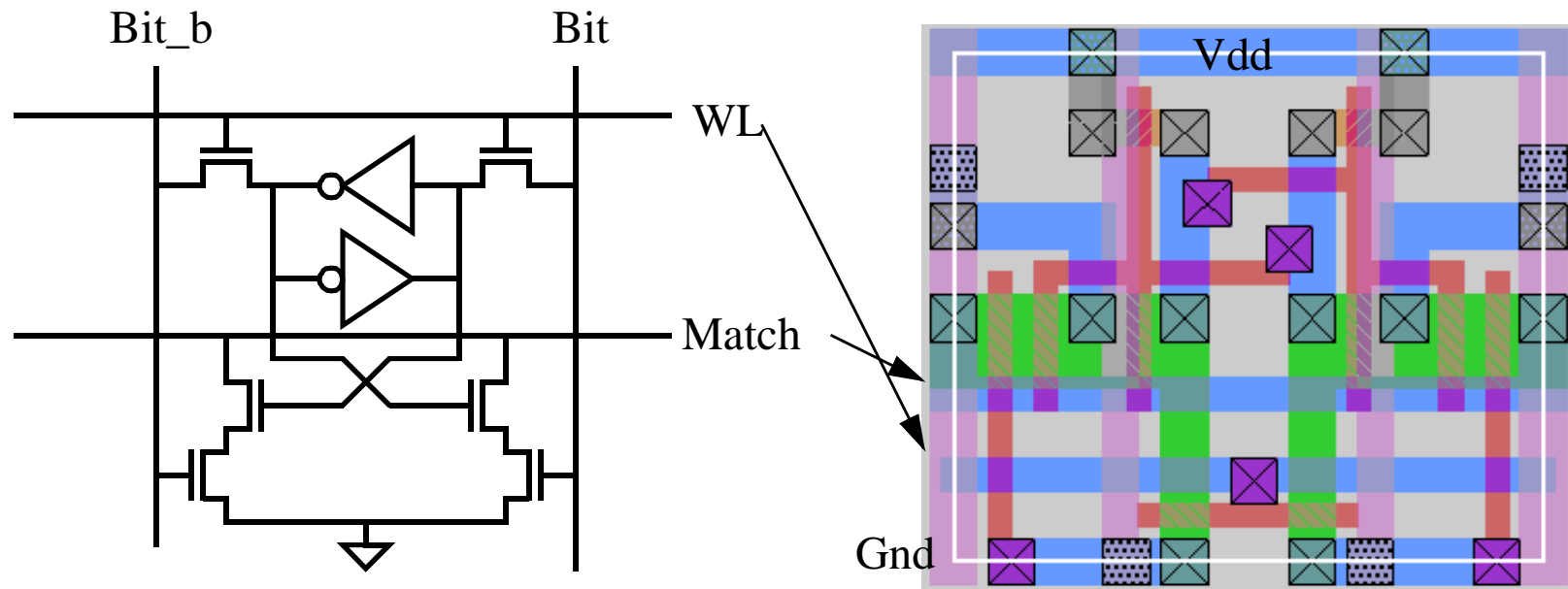
Can build true multi-ported memory cell, by adding more bitline pairs and wordlines to a cell.



Shown in the figure is a true dual port cell. You can read or write on each port every cycle. Since it has more bitlines than the previous cell, it is much larger in area.

+ Content Addressable Memory (CAM)

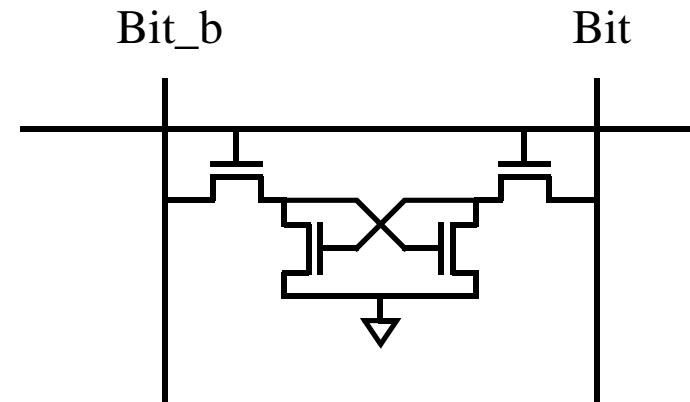
In some applications it is nice to find out if anything in the memory matches a certain key value. This can be done by a special memory cell called a CAM cell. Each CAM cell contains an XOR gate that compares the cell value with the data on the bitlines. If this bit matches, nothing happens. If it does not match the bitline value, it pulls the match line low. Connecting all the bits in a word to a precharged Match line (the bitline must be low during the match line precharge) means that the Match line will remain high, only if the value in the memory matches the key. One can even have don't cares in the key by driving both bitlines low.



+ 4T RAM and 3T RAM

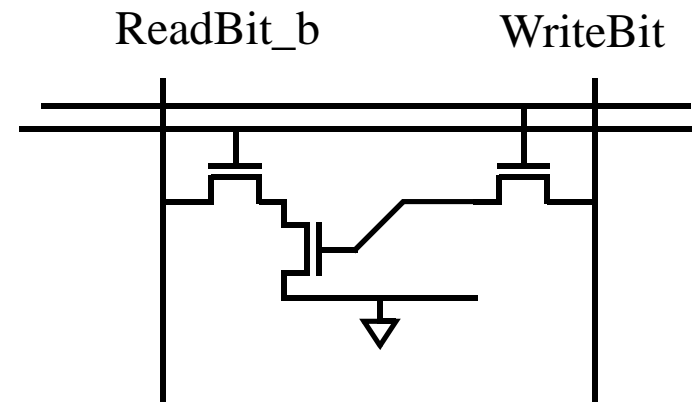
Take 6T cell and remove pMOS transistors

- Makes cell smaller (28 x 28)
- Circuit design is much harder
 - Internal nodes don't go to Vdd
 - Cell won't work at low Vdd
 - High value stored is degraded,
effective strength of nMOS pulldown is reduced



Make this circuit single ended, and get 3T cell

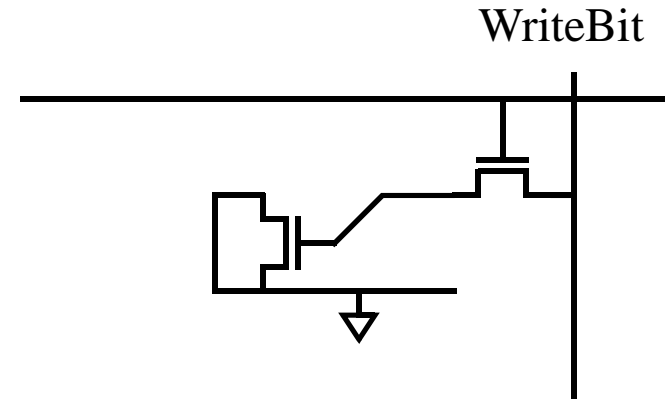
- Need 2 wordlines, Read WL, Write WL
- Can have 1 or 2 bit lines (Read / Write)
- Not very small, since it has more wires



+ 1T DRAM

Continue to remove transistors and wires

- Get 1 T, one WL, and one BL
- Pretty weird, no read bus
 - Read by cap charge sharing



When WL goes high, cap shares charge with bitline, changing its value slightly. It is this change in value that is detected by the sense amps

- Read are destructive

Once the charge sharing occurs, the cell does not have its value any more. After every read, the cell need to be rewritten by driving the bitline high (or low) before the wordline is lowered

- Wordlines generally go $> V_{dd}$

Would like to get as much charge as possible on the cap. Loosing a V_{th} does not leave enough signal on the capacitor for sensing.