Lecture 12:

MOS Decoders, Gate Sizing

Memory

Reading

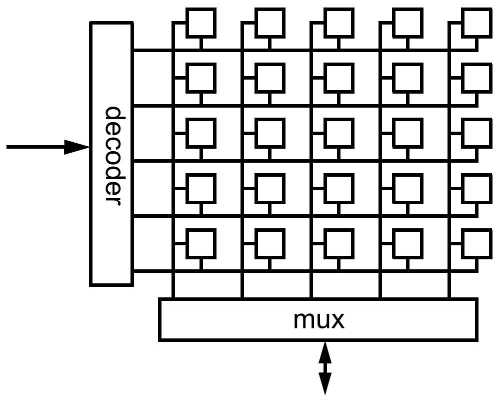
W&E 8.3.1 - 8.3.2 - Memory Design

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.

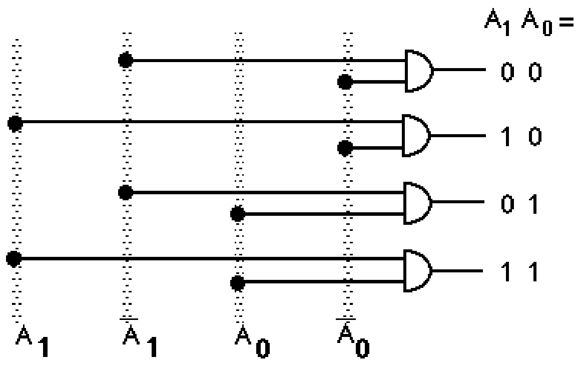
Peripheral Circuits



We need to build the decoder and wordline drive circuits, and the column select and bitline drive circuits. For both we need to build a decoder -- something to select the correct line. Lets look at building decoders for CMOS memories.

Decoders

A decoder is just a structure that contains a number of AND gates, where each gate is enabled for a different input value.

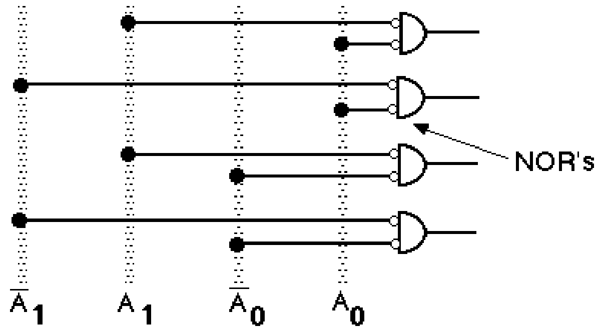


For a n-bit to 2^n decoder, we need to build 2^n, n-input AND gates. And we want to build these AND gates so they layout nicely (in a regular way)

Large Fanin AND Gates

In CMOS building this type of gate causes a problem, since large fanin implies a series stack. We will see a little later in the notes that the best way to do this is to use a two-level decoder by predecoding the inputs.

In nMOS the problem was easy, large fanin NOR gates work well. So



a collection of NOR gates solves the problem very nicely.

CMOS Decoders

In CMOS, a large fanin gate implies a series stack. So we need to build a decoder that does not use a large fanin gate. But how? Use a 2-level decoder.

- An n-bit decoder requires 2n wires

  A0, $\overline{\text{A0}}$, A1, $\overline{\text{A1}}$, …

  Each gate is an n bit NOR (NAND gate)

- Could predecode the inputs

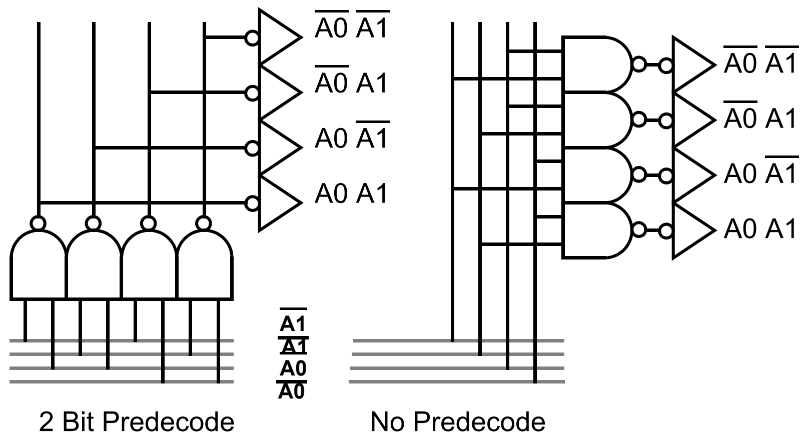
  Send $\overline{\text{A0}}$ $\overline{\text{A1}}$, A0 $\overline{\text{A1}}$, $\overline{\text{A0}}$ A1, A0 A1, $\overline{\text{A2}}$ $\overline{\text{A3}}$ …

  Instead of A0, $\overline{\text{A0}}$, A1, $\overline{\text{A1}}$, …

  Maps 4 wires into 4 wires that need to go to the decoder

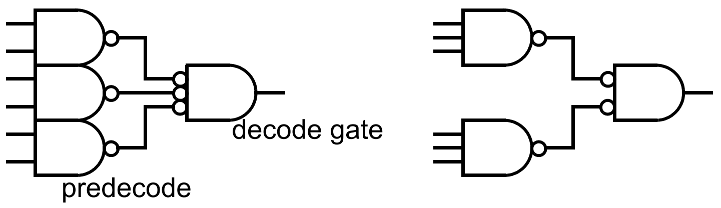
  Reduces the number of inputs to the decode gate by a factor of two.

Predecode Example



Predecode

Predecode is just like what we did when we needed to make a single six input AND gate. Did it in a few levels:



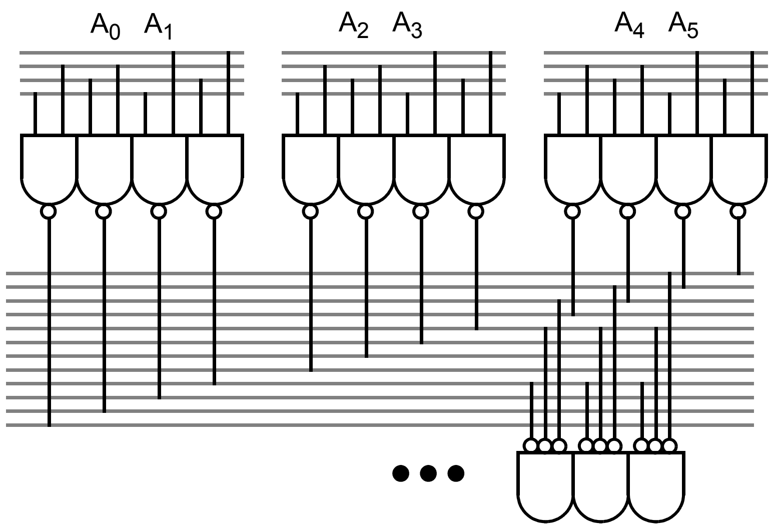
One can do a 2 input predecode, or a 3 input predecode

- A 2 input predecoder generates 4 outputs
- A 3 input predecoder generates 8 outputs

The difference with standard logic is that we need to decode all possible inputs. This means that each predecode gate can be reused by many 'final' decode gates. A little planning can yield a regular layout.

Predecode

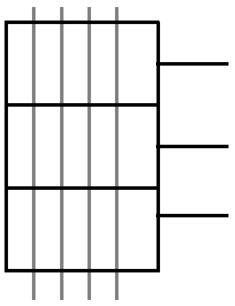
A predecoded decoder:



Layout Issues

Often we need to build large array structures (for example we need a large RAM), so we want to layout the decoder in as little space as possible. We need to find a good way to layout this structure.

Clearly we need to run the address lines through each decoder cell, and stack the decoder cells next to each other.



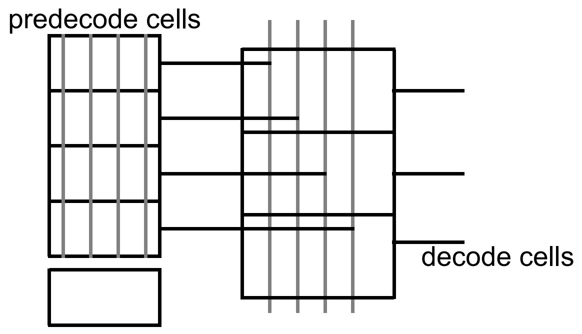
Predecode Layout

The output of the predecode gate need to drive the address lines.

- These address lines are usually high capacitance

  So usually it is better to use a NAND with an inverter buffer as the predecode cells.

- Cells can be placed on top of the address lines, or to the left of the address lines.



Decoder Cell Layout

- Need to have n and p transistors
- Need to take up minimum space
- Want it to be easy to ‘program’ the cell

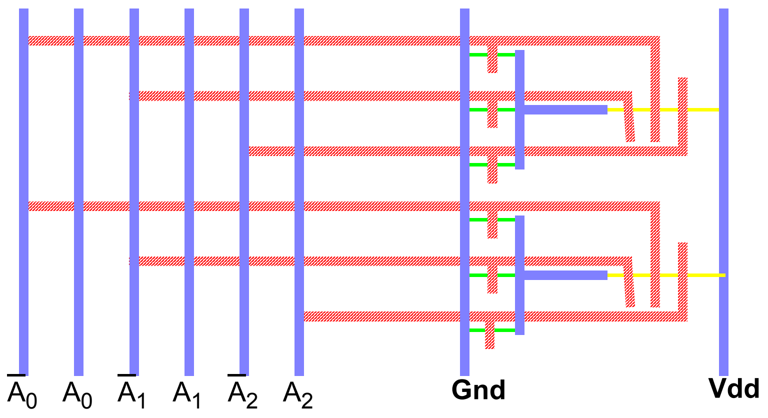
  While layout is regular each cell is different

  It connects to a different set of inputs

- Look at a couple of layout styles

Decoder Layout

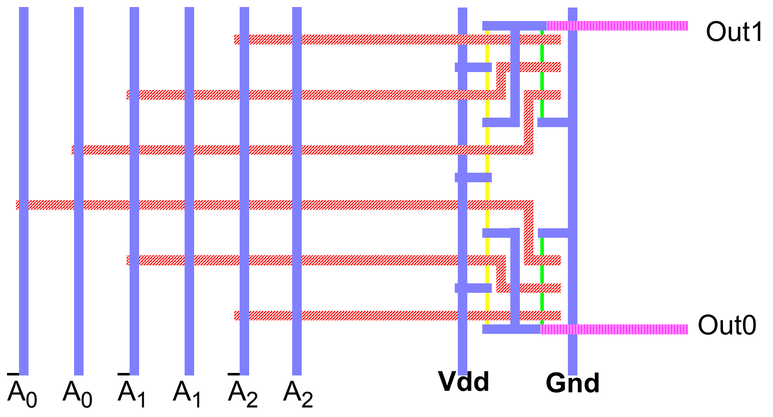
Cell Area is proportional to n^2. Decoder area is n^3.



The problem with this layout is that most of the space is wasted. All of the area under the wires is wasted. We should rotate the gate to fit under the wires.

A Slightly Better Decoder Layout

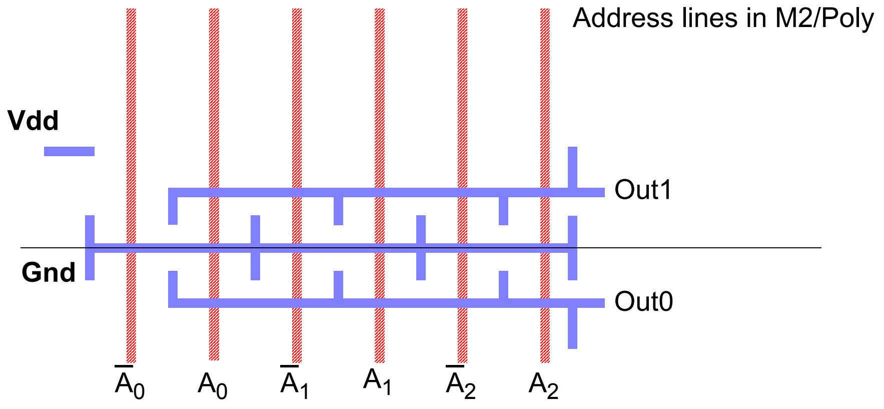
Better cell design (like we have talked about)



In this layout, the basic cell remains unchanged, it is the wire contacts that are programmed. This is sometimes a good idea, since it lets you optimize the decode cell (in this case the 3 input gate)

A Smaller Layout

Leave space for all the tracks in the cell



Need to program the decoder by placing transistors, or metal.

With predecode, you have more tracks per transistor.

Wordline Driver

Decoder is just part of the wordline drive circuit

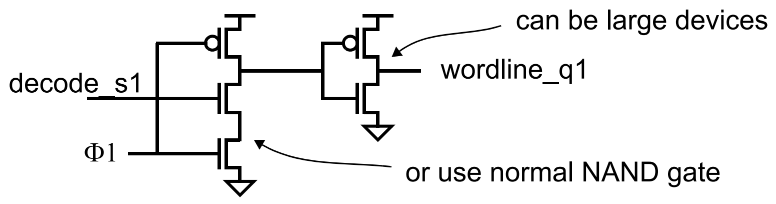
- Also need to qualify the wordline (AND with clock)
- Also need to buffer the signal to drive WL cap

Clock qualification can be done in the decoder

- $\overline{\text{A0}}$ … $\overline{\text{An}}$ Phi1 - just another input to the decoder

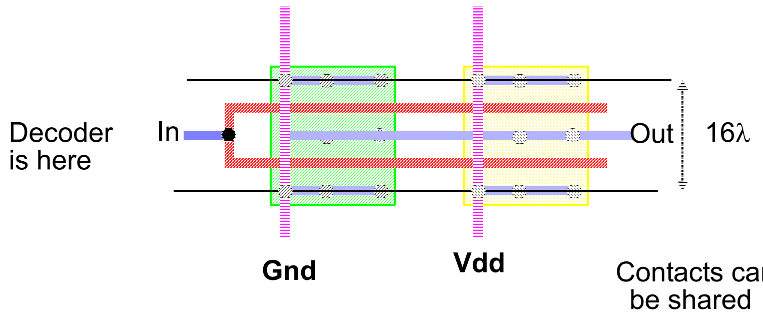
  Usually not a great idea, since this can lead to large skew

  Clock AND is usually done in last stage before driver



Thin Drivers

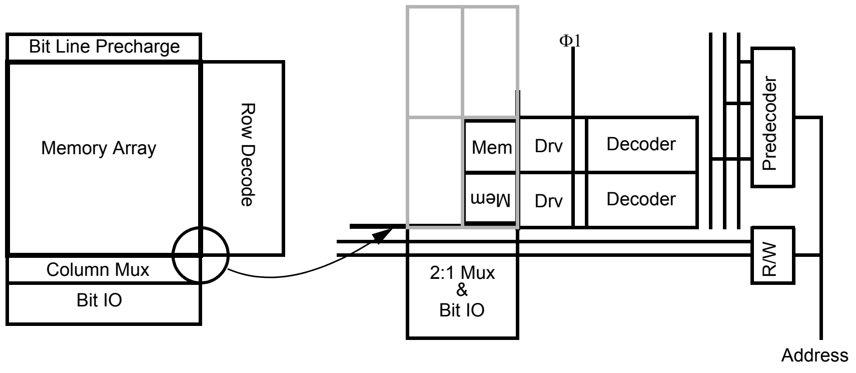
Wordline pitch of memory cell is not that tight (about 40λ), but not that large either. There are some memories (ROMs, dRAMs) with much tighter pitch. For many of these applications you need thin gates and drivers. The minimum useful space is 16λ



For the wordline driver, I might use two of these drivers in parallel, to reduce the horizontal length (effectively fold the transistors again)

Putting it Together

Floorplan for a memory



Built using Array constructs in Magic

- Decoder base is often array, with programming done by software

Memory is built by arraying a cell that contains the cell and its mirror

Transistor Sizing

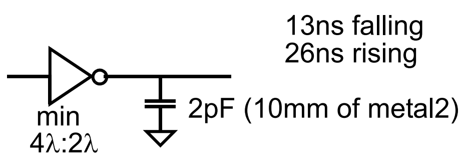
For memories (and other structures) you end up with long high cap wires

- Need to drive these large capacitors quickly, and this sets the device size
- We will look at chain of inverters first, and then think about gates

Factors to consider in gate sizing:

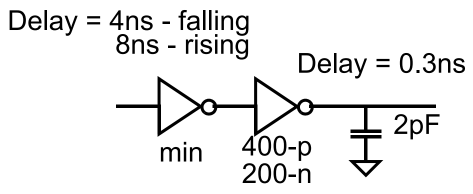
- Need to think about the load you are driving
- Need to think about the load you present to your predecessor

Why transistor sizes matter when you are driving a large capacitance



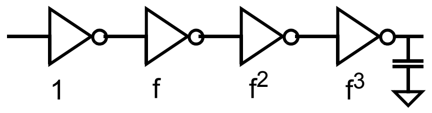
Buffer (or Gate) Sizing

But bigger gates have bigger input capacitance too:



Clearly we need to make the predriver larger too.
Is there an optimal solution? Yes, in a way

- Minimize delay of chain - for the minimum all delays will match (why?)



- Equalizing delay principle applies to any critical path through gates.

Buffer Chains to drive a Big Load

Each gate drives a gate that is f times as large as it is.

- Assume that Wp = 2 Wn, so rise and fall times are the same

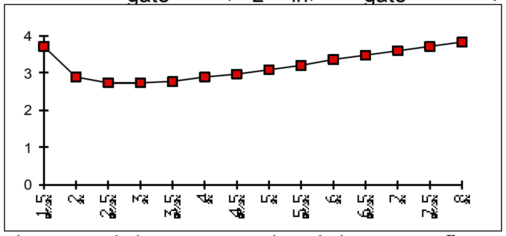
1. Final load cap is C_L since the fanout (C_L/C_{in}) of each gate is the same

$f^N * C_{in} = C_L$ $\quad\quad N = \ln(C_L/C_{in})/\ln(f)$

- Most books assume that the delay of a gate is:

$= f * T_{gate} = f *$ delay of a gate with a fanout of 1

Total delay $= N * f * T_{gate} = \ln(C_L/C_{in}) * T_{gate} * f / \ln(f)$



has a minimum at *e*, but it is pretty flat

Analysis is Slightly Wrong

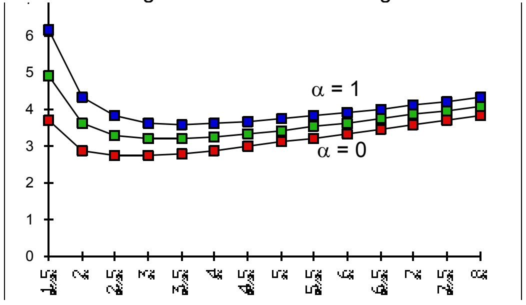
The optimal point of 'e' assumes no wire delay and that the gate delay is

Gate Delay $= f * T_{gate}$

- But this is not true, since the delay of a gate with no load is not zero.
- Each gate has some intrinsic delay from its own diffusion capacitance

Gate Delay $= T_0 + f * T_g = (f + \alpha)\, T_g$ $\quad\quad .5 < \alpha < 2$

Total delay $= N * f * T_{gate} = \ln(C_L/C_{in}) * T_{gate} * (f + \alpha) / \ln(f)$



So, really should use f about 4 between stages.

Gates

Two issues:

- What about things other than inverters? Principle still holds.
- What happens when the wire load is not negligible?

In general it is a little complicated

- General rule still holds -

  You should not be able to make any transistor bigger and decrease the delay (cost to your predecessor should equal your gain)

- For gates, you want to keep the loaded delays roughly equal

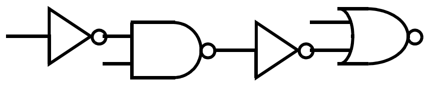
  This is not the same as keeping the fanout the same

  A NOR gate has a larger delay than an INV at the same fanout

  This difference is sometimes called logical effort (more later)

Gate Sizing

For a fixed number of stages:



- If any of the delays are not equal,

  Make the gate with the largest delay larger.

  Decreases its delay, and increases its predecessor's delay.

  But since its delay started larger, there will be a net win.

  Optimal is when the delays are equal

- For design, you don't want tons of SPICE or irsim simulations.
  What you really want is a spreadsheet or a program that solves the sizing problem as a linear optimization problem where each size is a dimension of the solution space.

Wire Cap

For fastest systems, want the wire capacitance to be small compared to gate capacitance

- But this leads to very large transistors.
- Compromise is to try to keep ratio from 30% to 70% wire

For a standard cell library how big should the transistors be?

- Want the delay to have some tolerance to placements.
- Implies that the wire capacitance should be a small fraction of total
- Long wires are probably millimeters (.2pF)
- So, transistors should be pretty large (10-20x minimum size)
- OK, since transistors are the free things that fit under wires.

Trend is toward larger transistors.   Stick layout diagrams should 'show' transistor widths. In industry, you don't default to minimum size transistors, you should default to 5-10x minimum size.

+ Wire Resistance

Previous slides ignored wire resistance

- For short wires this is ok ($R_{wire} \ll R_{trans}$)
- As wire gets longer

  R_{wire} gets larger

  R_{trans} gets smaller (larger transistor to drive larger capacitance)

  Can become an issue

- Wire resistance delay is proportional to length^2,

  Capacitance of wire is proportional to length

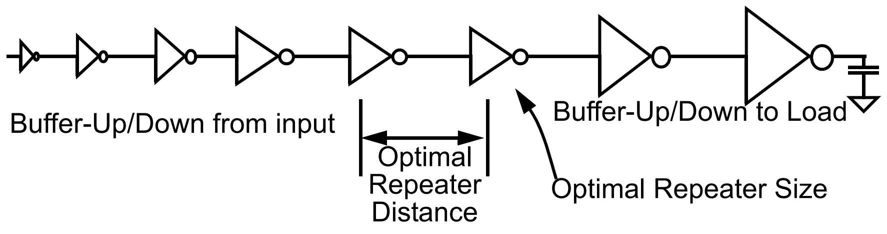
  Resistance is proportional to length too

- Sometimes add repeaters to reduce the total wire delay.

  Break the quadratic increase, but adds buffer delay

+ Long Wires

For long wires, we can separate the optimal wire into three regions:



In the middle region, there is an optimal spacing that minimizes the delay

- Added buffer delay is matched by reduced wire delay

    Can use RC model to find optimal length, and repeater size

    The chain at the start and end buffer up or down from the driver and load to this “natural” size.

    This optimum distance is about 4 to 7mm in typical 0.5μ fabs.

General Rules of Thumb (for speed)

- Try to keep the fanout of all gates to be less than 5
- Try to keep the delays of the gates in a critical path roughly the same.

    Large fanin gates should have smaller fanout

- Keep fanin limited
- Often need short buffer chains (one inverter)
- Be flexible on sense of logic (push inversions around)
- Don’t use minimum size transistors, unless you know the wire is short