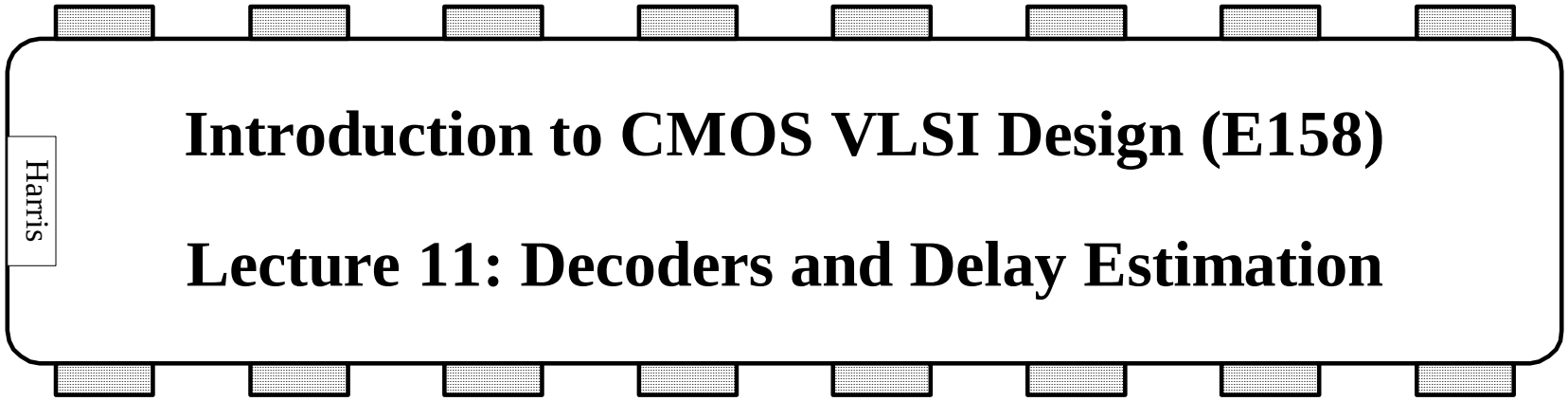


David Harris

Harvey Mudd College

David_Harris@hmc.edu

Based on EE271 developed by Mark Horowitz, Stanford University

Decoders and Delay Estimation

Reading

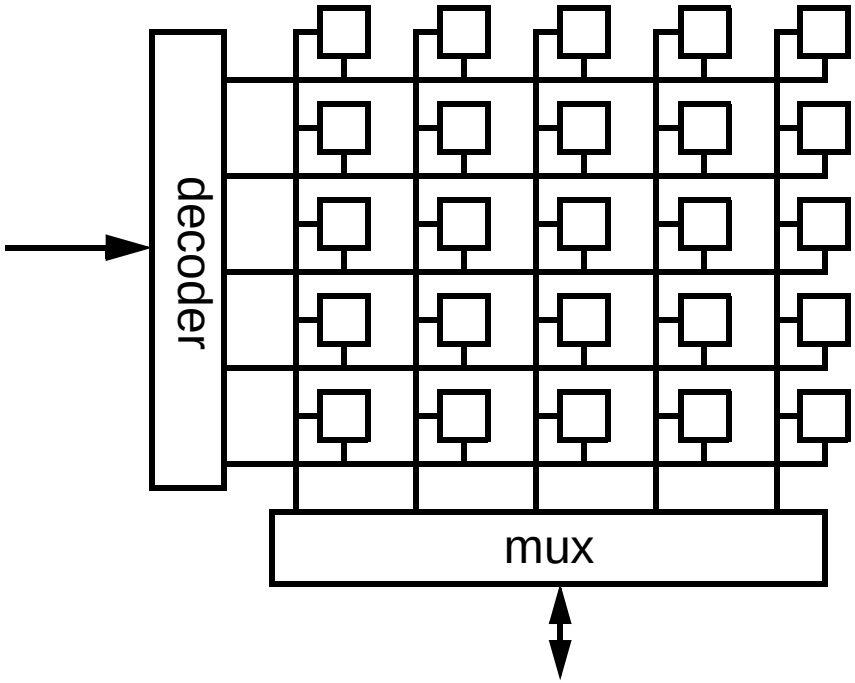
W&E 4.5-4.6

Introduction

In the last lecture, we looked at memory design. Today we will look at various methods for building decoders to drive the word lines and column multiplexer circuitry.

To build a fast memory, we need to minimize the delay of the decoder. This challenge will serve as a jumping off point for delay estimation and gate sizing to minimize delay.

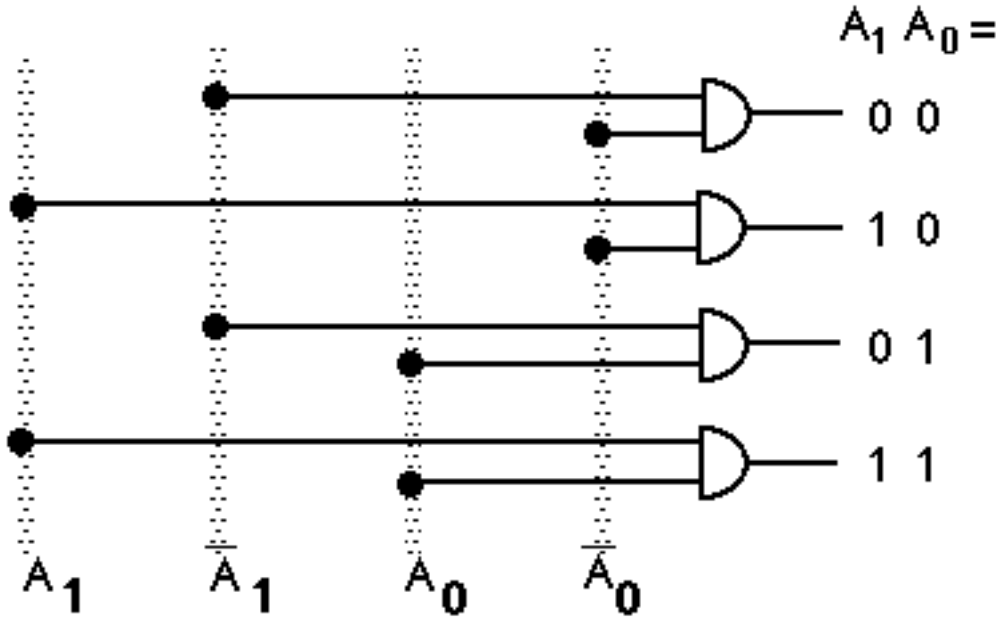
Peripheral Circuits



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.

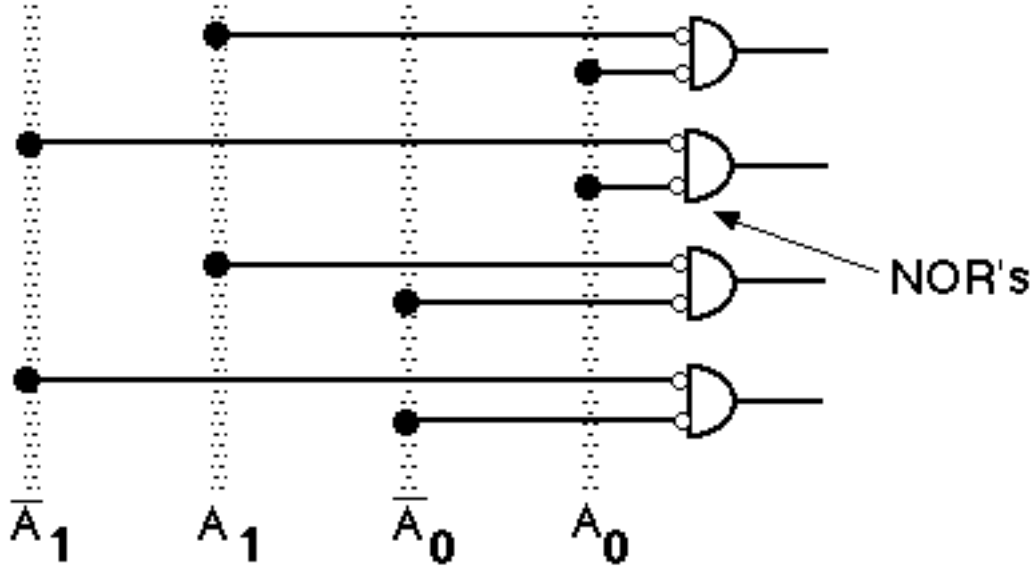


For a n-bit to 2^n decoder, we need to build 2^n, n-input AND gates. And we want to build these AND gates so they layout nicely (in a regular way)

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

  Each gate is an n bit NOR (NAND gate)

- Could predecode the inputs

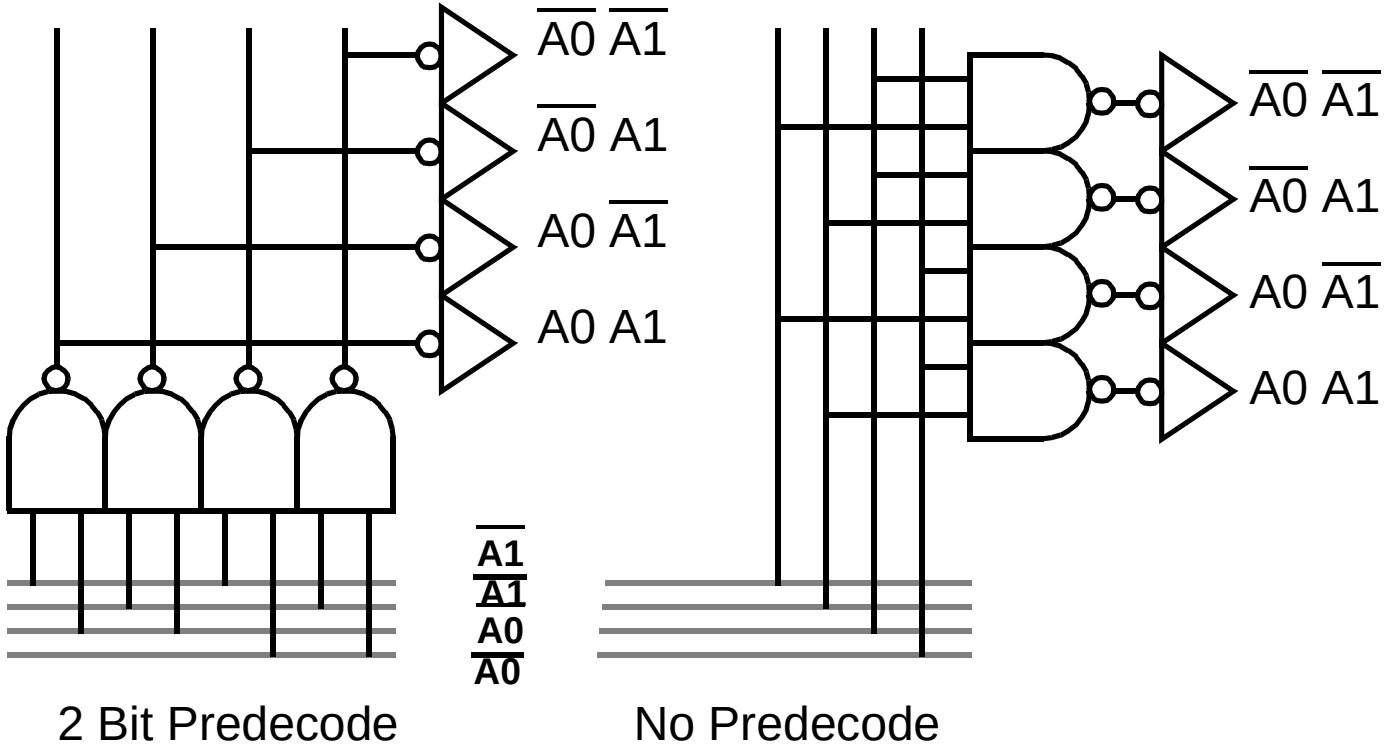
  Send $\overline{A0}\ \overline{A1}$, A0 $\overline{A1}$, $\overline{A0}$ A1, A0 A1, $\overline{A2}\ \overline{A3}$ …

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

  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

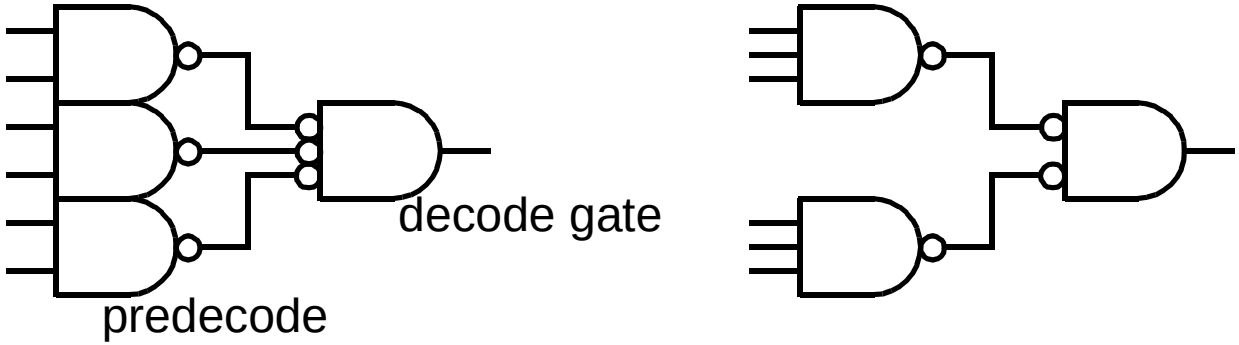


2 Bit Predecode

No Predecode

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:



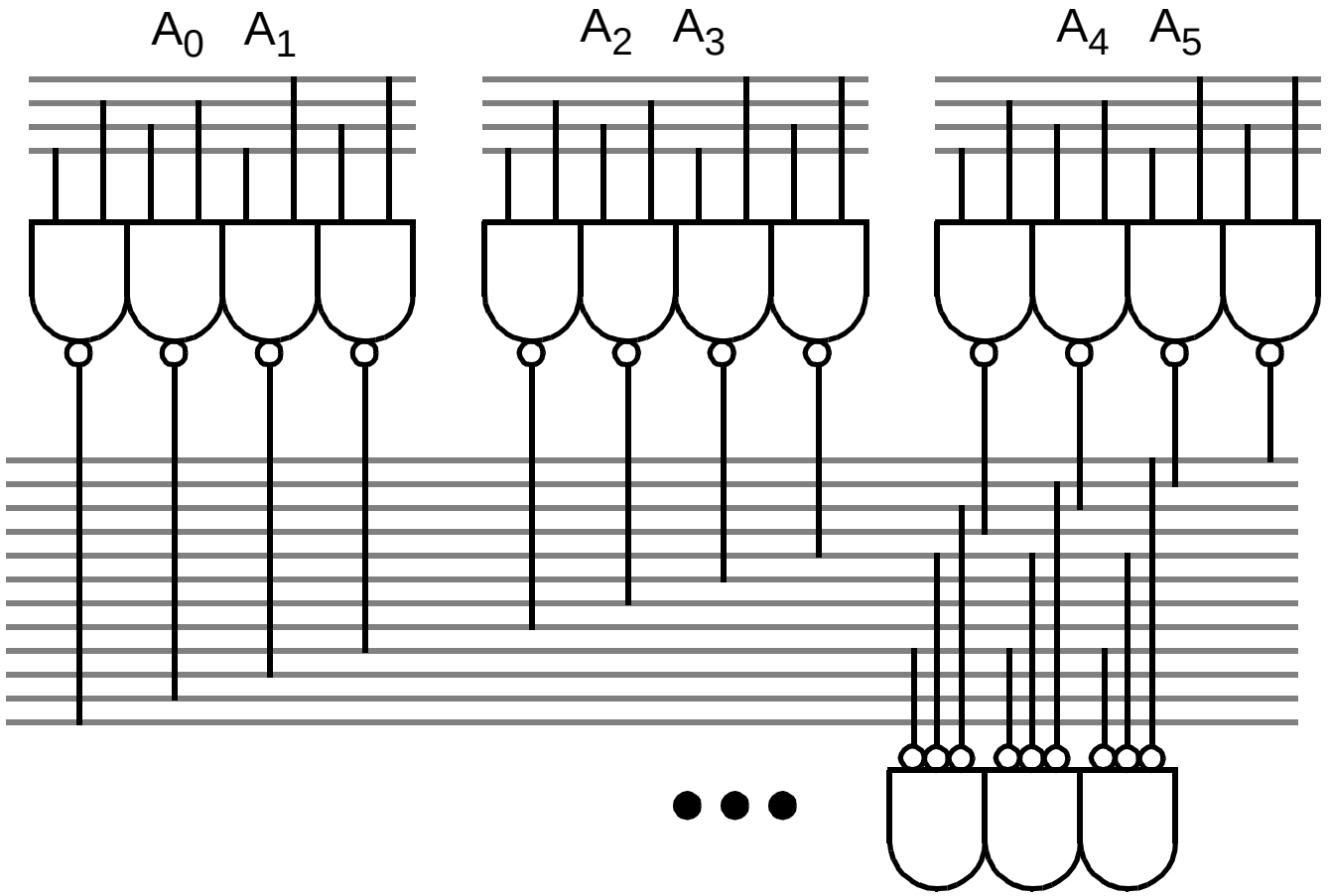
One can do a 2 input predecode, or a 3 input predecode

- 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

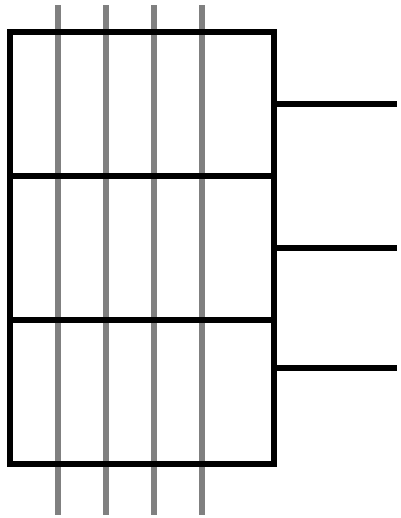
A predecoded decoder:



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.



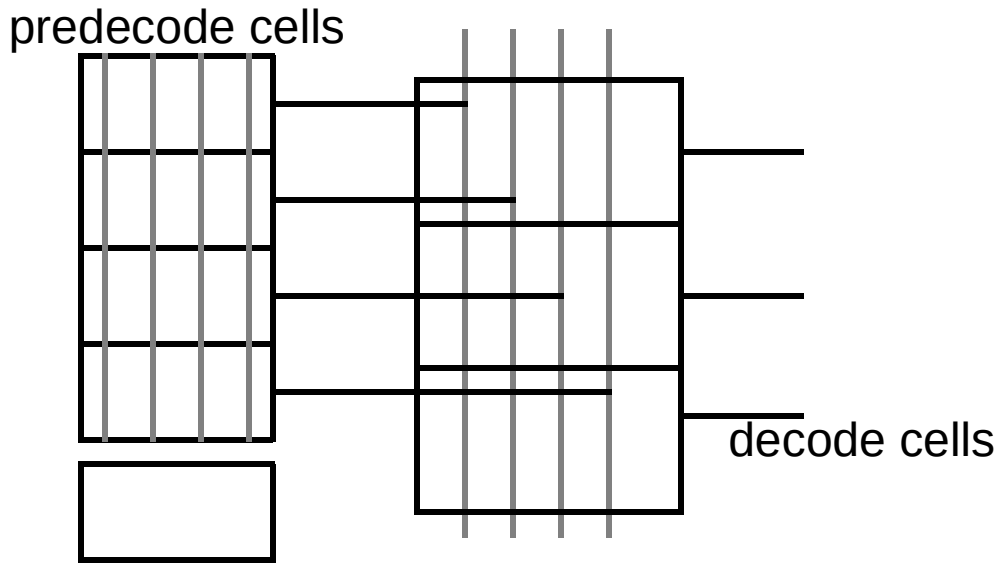
Predecode Layout

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

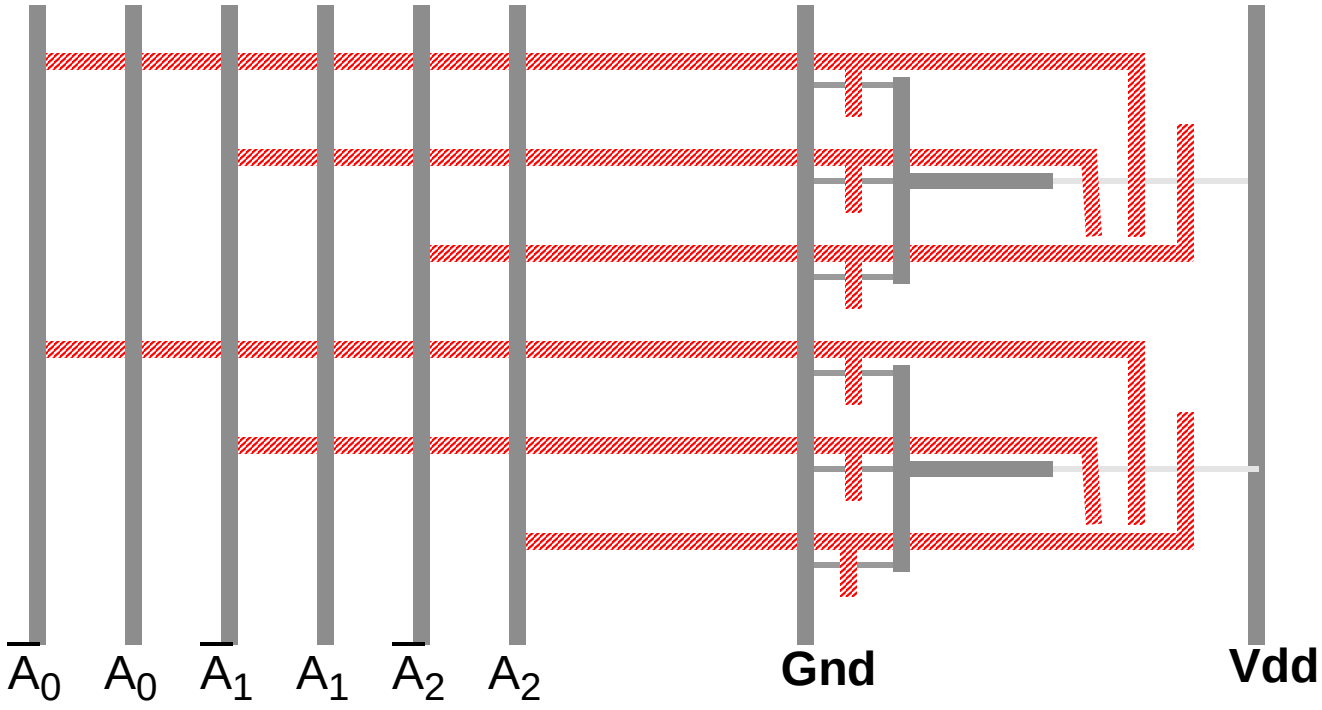
  While layout is regular each cell is different

  It connects to a different set of inputs

- Look at a couple of layout styles

Decoder Layout

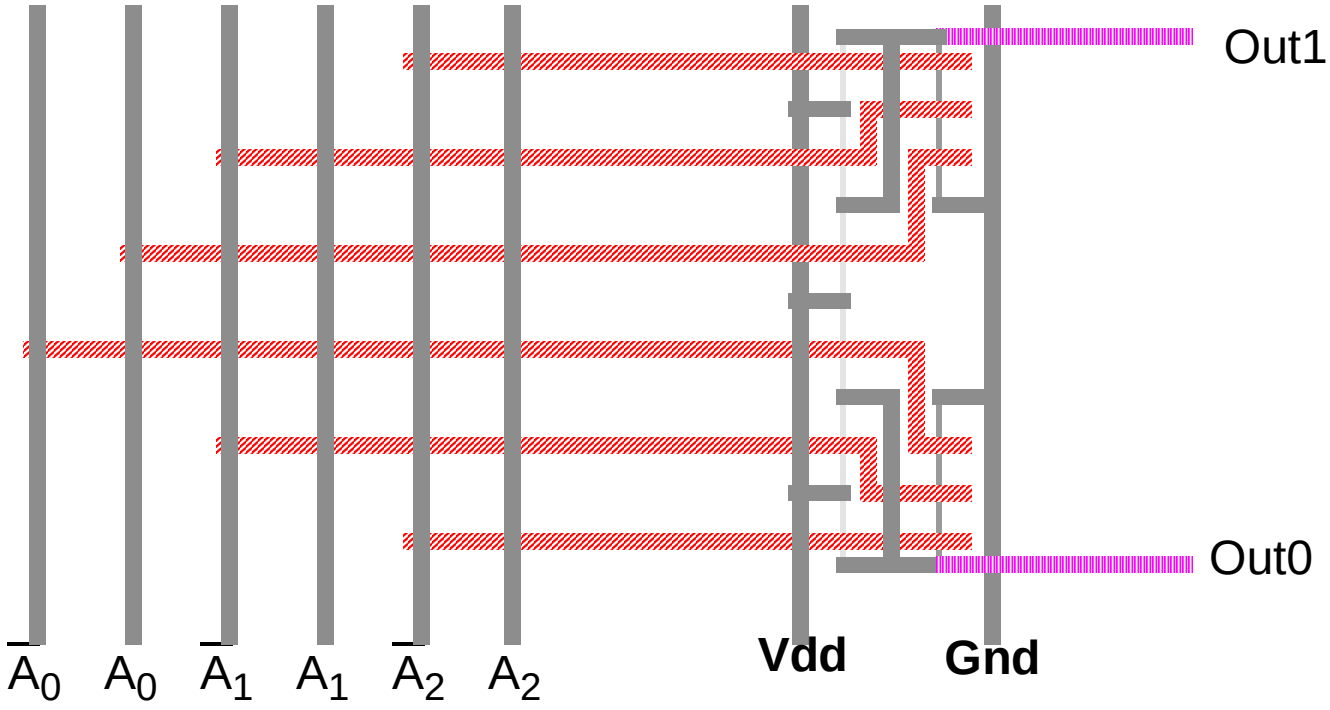
Cell Area is proportional to n^2. Decoder area is n^3.



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

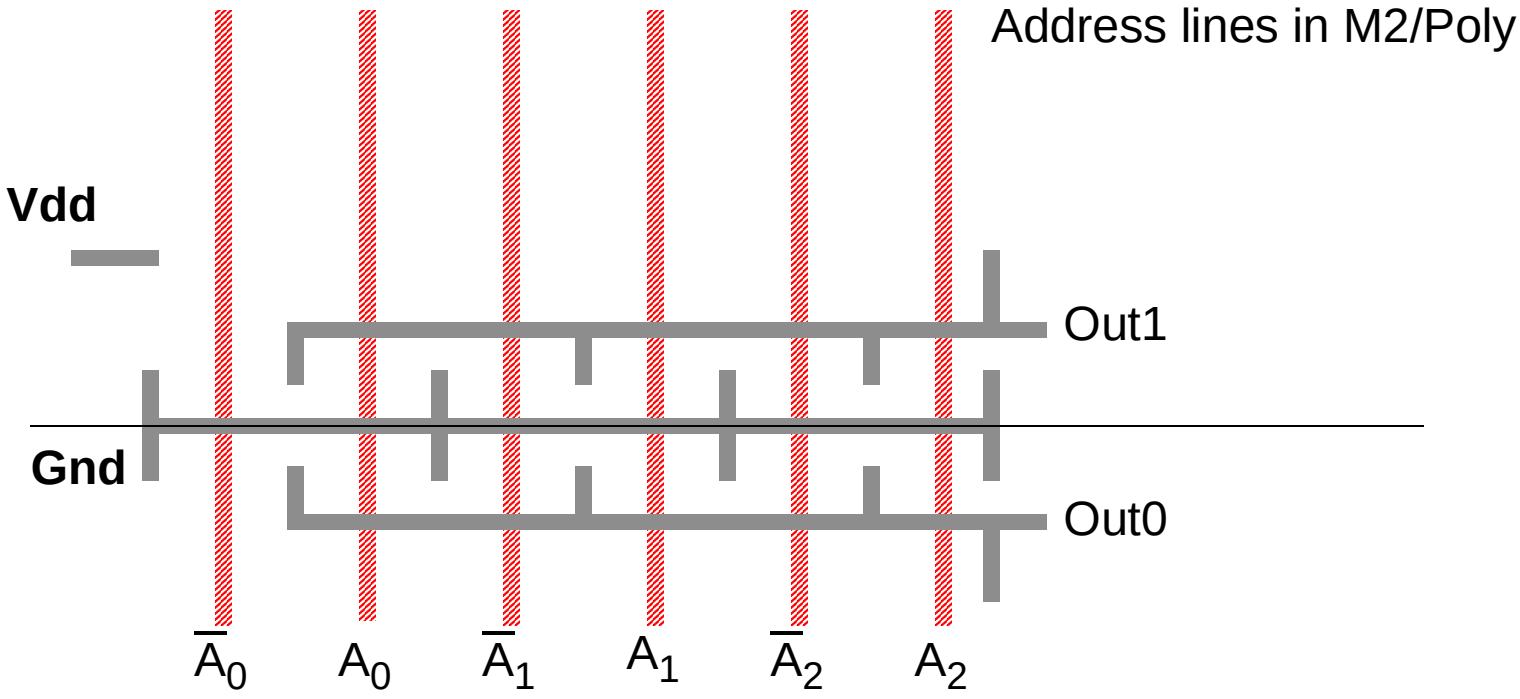
Better cell design (like we have talked about)



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

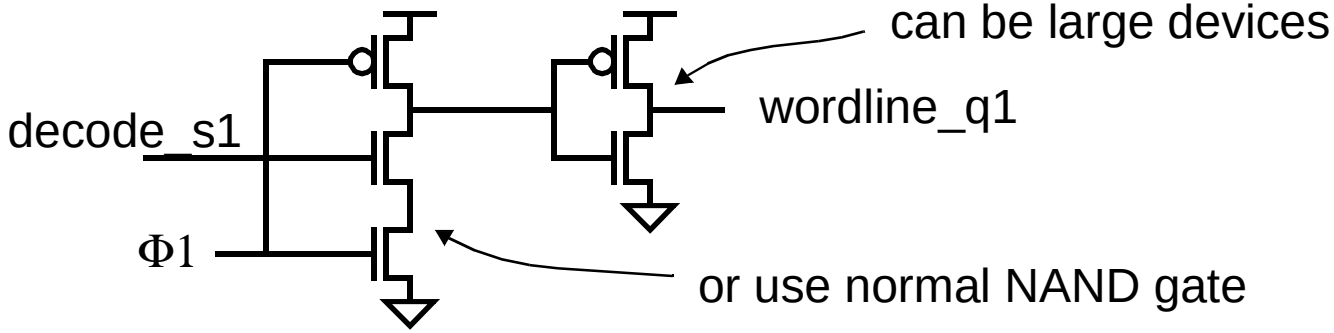
- Also need to qualify the wordline (AND with clock)
- Also need to buffer the signal to drive WL cap

Clock qualification can be done in the decoder

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

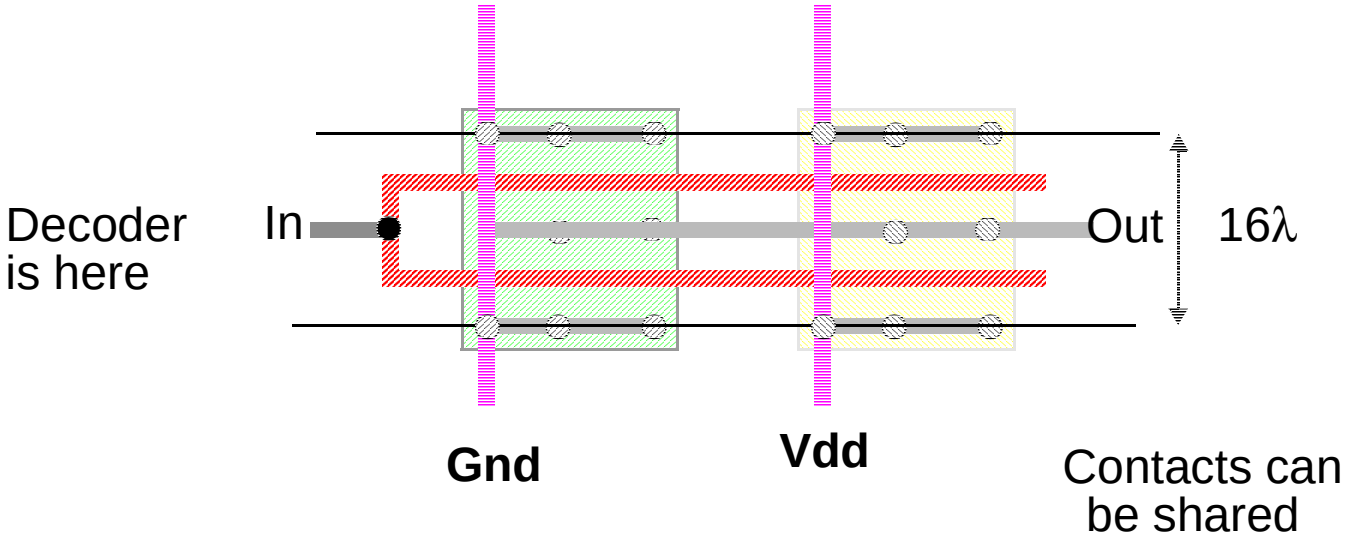
  Usually not a great idea, since this can lead to large skew

  Clock AND is usually done in last stage before driver



Thin Drivers

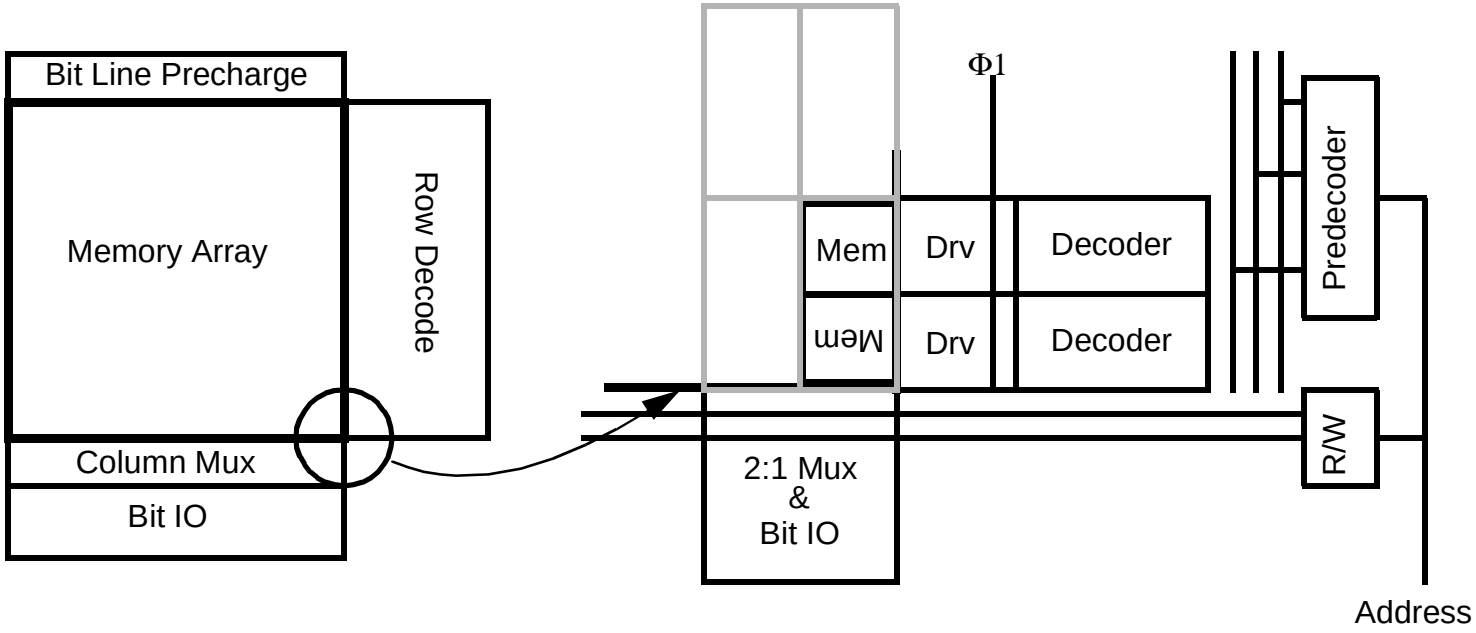
Wordline pitch of memory cell is not that tight (about 40λ), but not that large either. There are some memories (ROMs, dRAMs) with much tighter pitch. For many of these applications you need thin gates and drivers. The minimum useful space is 16λ



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

- 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

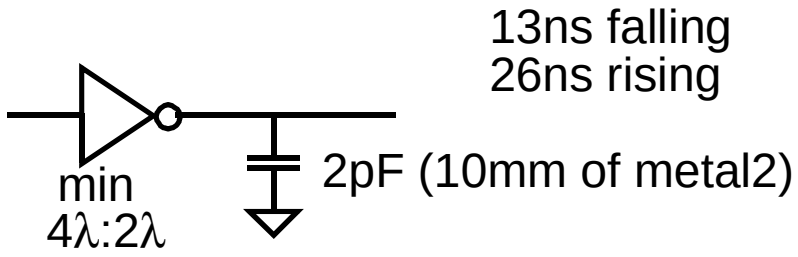
For memories (and other structures) you end up with long high cap wires

- 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:

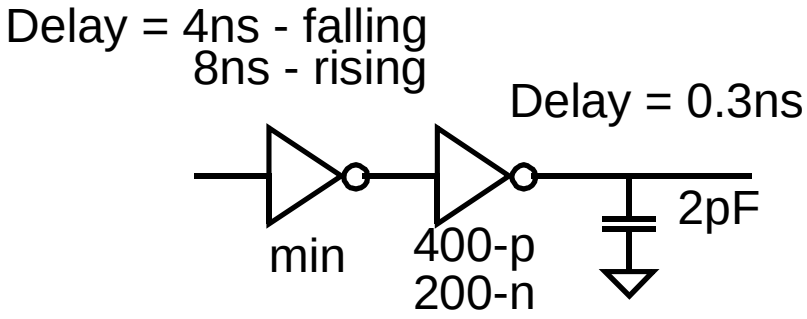
- Need to think about the load you are driving
- Need to think about the load you present to your predecessor

Why transistor sizes matter when you are driving a large capacitance



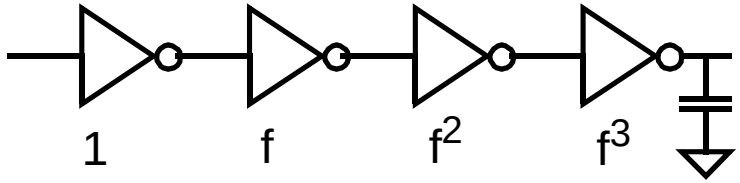
Buffer (or Gate) Sizing

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.