Lecture 9:

Cell Design Issues

Overview

Reading

W&E 6.3 to 6.3.6 - FPGA, Gate Array, and Std Cell design

W&E 5.3 - Cell design

Introduction

This lecture will look at some of the layout issues for cell designs. There are two issues in cell layout, what are the internal constraints (how will the cell be built) and what are the interface constraints (how will the cell be used). Datapath cells, and other array cells, have more interface constraints to allow them to connect by abutment.

This lecture first looks at different implementation styles, and then at the cell layout problem in more detail. Wires are now a key component of the design, so we examine the wire properties more closely. The lecture also looks at sizing of Vdd, Gnd wires.

Wires are Key

As discussed in last lecture, the wires are critical to design

- Set the capacitive load on the gates
  - Need to know (estimate) wire length to size gate
  - Planning is critical
- Have resistance too
  - Long wires have their own RC time constants
  - Very little you can do
- Need to have a number of special wires
  - Power, Gnd, clock
  - Need to have very low effective resistance

Wire Properties

For best performance (and area)

- Need to use the right wires for the right job
- Different layers have very different characteristics:

Layer	Resistance	Capacitance	Connects to
metal2	low	low	m1
metal1	low	low	diff, poly, m2
poly	medium*	low	gate, m1
ndiff	medium*	high	s/d, m1
pdiff	medium*	high	s/d, m1

* Could be high if a silicide process is not used. For most technology < 1μ, there is silicide.

Wire Numbers

Like a transistor, the resistance of layer is given by R_{sq} times the number of squares. But unfortunately for wires L is usually much larger than W.

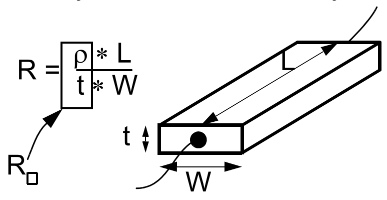


Table 1:

	R_{sq}	R_{sq}/R_{trans}
metal	.05Ω	$1/2.6x10^5$
poly	5Ω	1/2600
ndiff	5Ω	1/2600
pdiff	5Ω	1/2600
nMOS	13KΩ	1
pMOS	26KΩ	2

For a process without silicide the resistance of the ndiff, pdiff and poly layers increase to be 100s of ohms/sq.

Diff should not be used because of its large capacitance. Poly can be used for short wires, but the resistance is too large for any long wires.

Wire Resistance

Look at driving a wire that is 1000 λ long

- Wire cap 0.2fF/μ * 500μ = 100fF
  - Size transistor to be 8λ nMOS, 16λ pMOS
  - Resistance is 13K/4 = 3.25K
- Wire resistance is Rsq * 1000/3 for metal; 1000/2 for poly
  - 17Ω for metal, 2.5KΩ for poly

Look at driving a wire that is 10000 λ long (that is only 5mm in our technology)

- Wire cap 0.2fF/μ * 5000μ = 1000fF
  - Size transistor to be 80λ nMOS, 160λ pMOS
  - Resistance is 13K/4 = 325Ω
- Wire resistance is Rsq * 10000/3
  - 166Ω for metal, 25KΩ for poly

Wire Uses

- Diff
  - This is a terrible wire because of its high capacitance.
  - Only use is to connect to transistors
- Poly
  - Resistance is pretty high. Good for local (short interconnections)
  - Don’t use to route outside a cell, and don’t as a jumper in a long wire
- Metal1
  - Only thing that can connect to poly and diffusion
  - Densely used in a cell
- Metal2 - MetalN-1
  - General wiring areas
- MetalN
  - Thicker metal for Vdd, Gnd and clock routing

Implementation Technology for Cells

There are many constraints that might be placed on the cell design

- To make the fabrication or CAD tool problem simpler

Implementations range from

- Field Programmable Gate Arrays (FPGA)
  - Chip are prefabricated, program fuses/antifuses to get logic.
- Gate Arrays
  - Transistors are prefabricated, customize metal to generate cell
- Standard Cells
  - All cells have fixed height, (wiring maybe restricted to channels)
- Standard Cells with Macros
  - Macros can be datapath and/or memory

+FPGAs

Idea is to fabricate a chip that can be programmed to do logic.

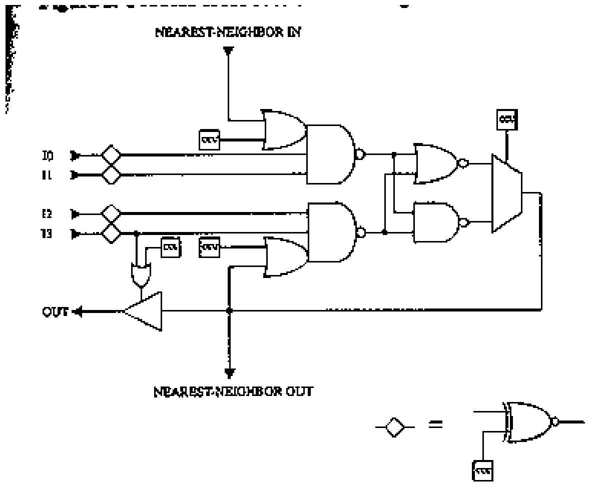
- Logic is programmed into the chip after fabrication
- Programming is done using:
  - Memory cells and CMOS switches
  - Fuses or Antifuses
  - E-PROM technology (floating gates hold a value 10 years)
- Hard problem is “customizing” wires
  - Can program logic ok
  - To program wire connections need to add switches to wire
  - Switches have capacitance and resistance – slow

But FPGAs are completely prefabricated in large volumes, so can be cost effective and quicker than any other option. (Makes it a ‘hot’ area)

- Raw technology might be overkill (if don’t need speed or # of gates)

+Example FPGA Cell

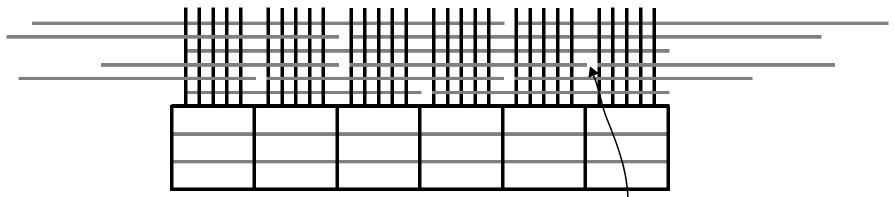
- Presentation by XILINX at 1995 International Solid-State Circuits Conf
- Chip has 1728 Configurable Logic cells each with schematic:



+FPGA Wiring

Basic idea is to have a standard cell like wiring (with channels)

- Each channel has wires of different lengths
- Number of each length is set by statistics from real designs
- Try to use the wire of the length you need
- When needed use a logic block as a repeater



Often a switch in the wire ‘gaps’

- Big problem is that there are switches in the wires -- have high resistance

+Gate Array

The cell designer must use predefined transistors

- W/L are all the same, or at best limited set of sizes
- Transistors are prefabricated in the silicon (many of the mask steps)
- Chip is covered with transistors (sea of gates)

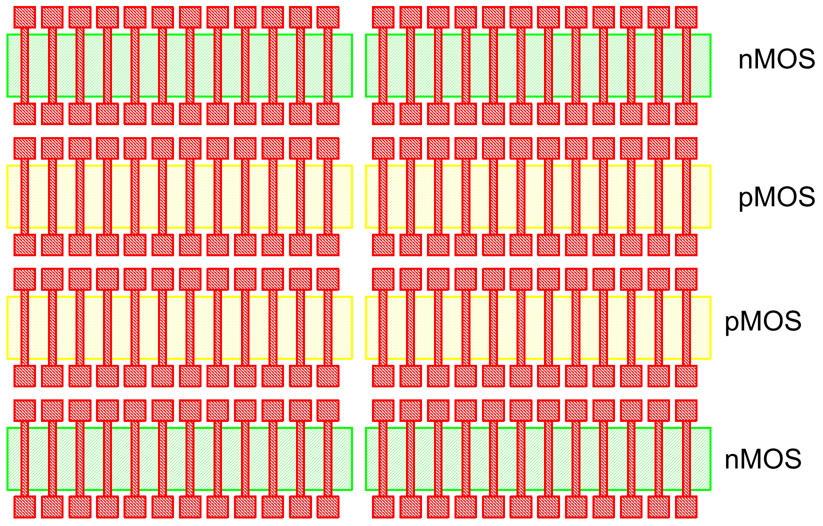
The cell designer provides the metal patterns that forms the transistors into useful logic units.

- The logic units are then placed and routed on the large array of trans.
- Transistor under wiring channels may not be used
- Tie neighboring transistor gate stripes off to separate the transistor drains actually getting used.

User still works in predefined cells (implemented in std transistors)

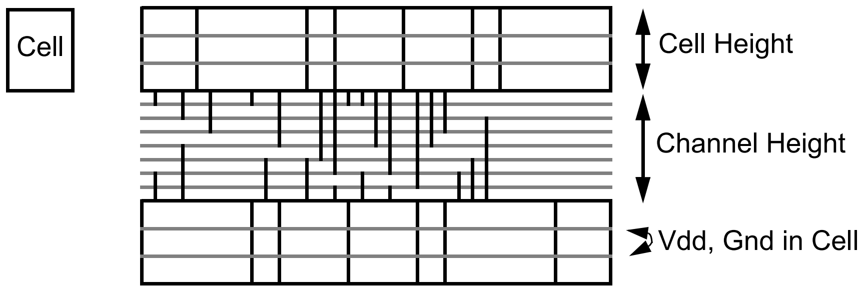
Cheaper (perhaps) and faster to manufacture (perhaps) than standard cells since you need to customize fewer layers

+ Gate Array Layout



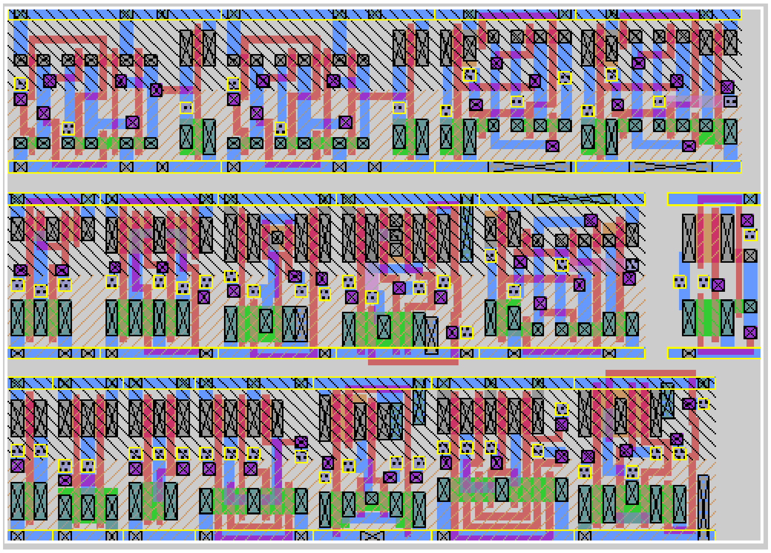
Standard Cells with Channel Routing

- Appropriate for all or part of a custom chip (defining all mask layers)
- All cells are the same height with abutting power and gnd connections
- Cells tiled into rows
- Rows of cells separated by routing. If M3, maybe over-cell routing too.
- Channel height can be set after the routing, so the wires always fit



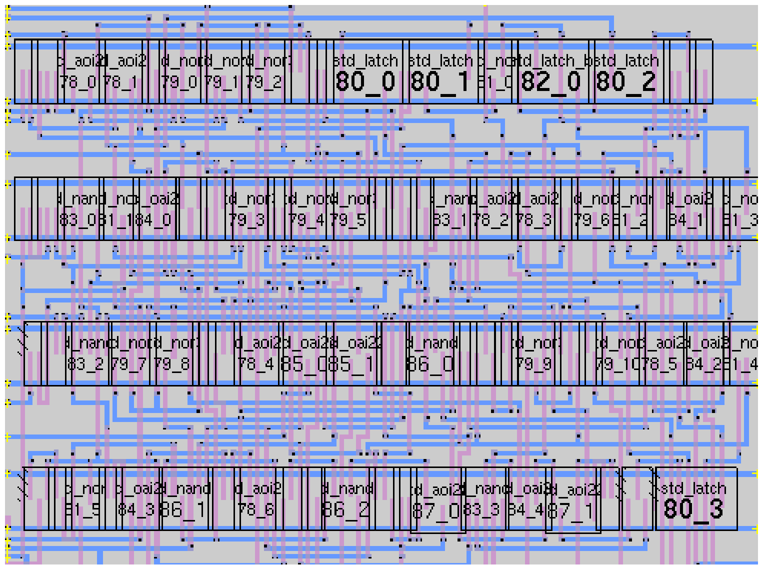
Standard Cell Example

Example of the cells



Standard Cell Routing

With two layers of metal



Standard Cells vs. Macros

Generally macros have more structured wires than standard cells, so you need to use a little different implementation style for the cells. For structured wires, the cells contain the wires and snap together.

- Standard Cells

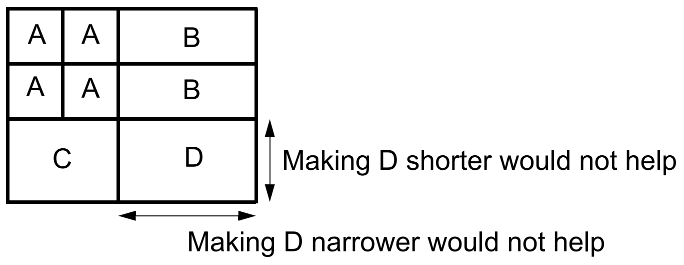
  The logic is done in fixed height cells. The cells are assembled into rows, and the wires that connect the logic together is done in wiring channels that are outside of the cells. This routing is usually done by CAD tools trying to be more automated than Magic.

- 'Snap-Together' Cells

  Rather than having external channels for the wires, in this design style the wires are contained in the cells. The wiring pattern is regular enough that the connection between cells is done by simply abutting the cells. This layout style is more restrictive than the Std Cell style since it supports a more limited wiring topology. Its advantage is that if the wiring can be made to fit this layout style, both the area and wire length (capacitance) can be reduced.

'Snap-Together' Cells

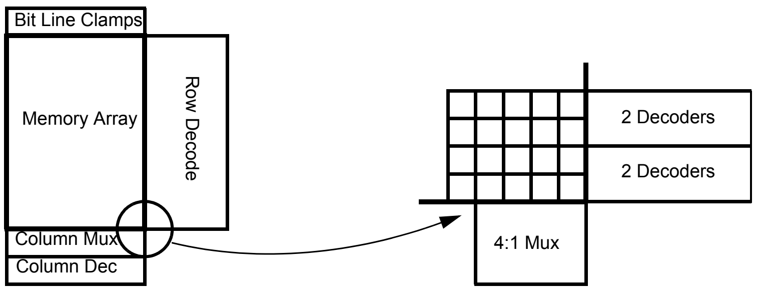
For this design the critical issue is 'pitch-matching' the cells (like std-cells, but requires matching in both dimensions, not just the height). Since we want them all to fit together, not only do we need to have the wire connect at the cell boundaries, but we also want the cells to be able to 'tile' the surface. That is cells that connect together should have the size in the connecting edge. In this design style, smaller is not always better. You need all connecting cells to share height and width on edges.


A
A
B
A
A
B
C
D
Making D shorter would not help
Making D narrower would not help

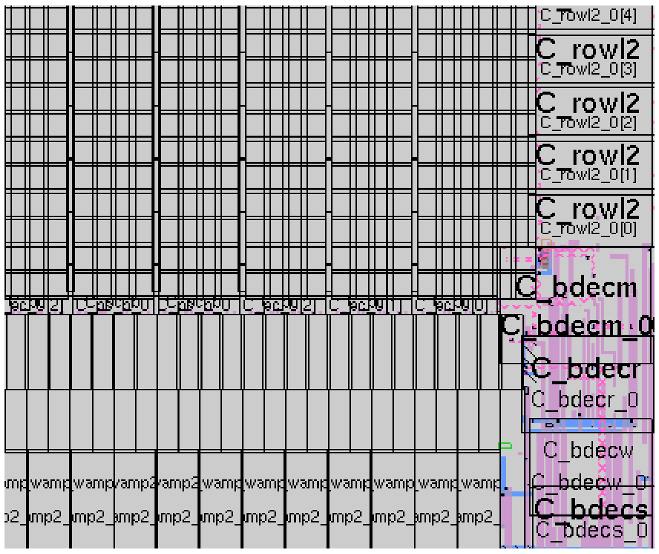
Abutting Cells

Two common examples of blocks that use these cells are regular arrays (usually for memory) and datapath (for dataflow design).

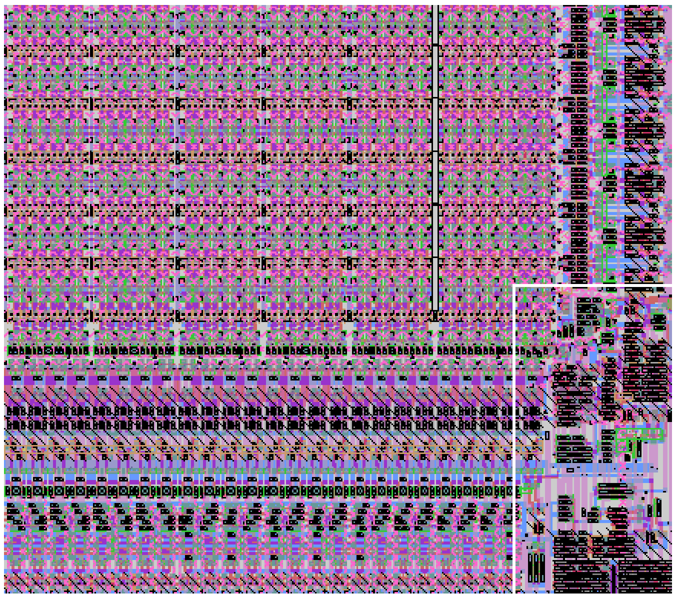
- In memory design, the core of the cells is a two dimensional array of bits. Since the communication between the cells is fixed, it is easy to embed the needed wires in the cells. Key here is to get all the edge decoder and mux cells to pitch-match to the small memory cells:



Memory Layout Example



Expanded

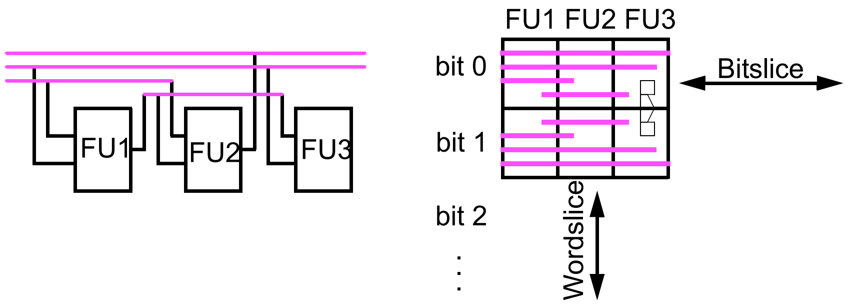


Datapath: Wires are in the Cell.

Datapaths operate on multiple bit data. Most of the communication is between functional units, bit0 of FUx going to bit0 of FUy. At each functional unit, all the bits are operated on roughly the same way. This means that each FU can be constructed from an array of identical cells, and the wires between the FU can be incorporated in the cells, so the whole structure can connect by abutment.

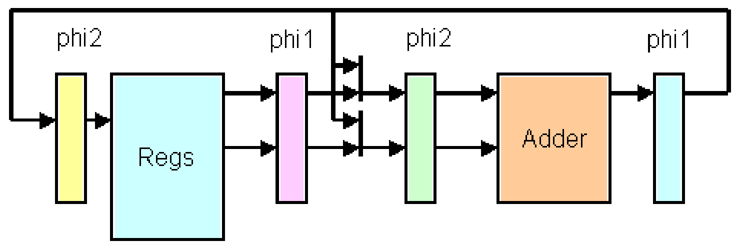
Think:

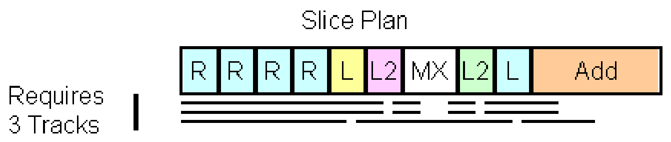
Build:



Often cells are mirrored every other row, so the cells share Vdd and Gnd rails.

Datapaths





Wire lengths (and hence required driver sizes) can be accurately estimated from a slice plan like this

Datapath Cells

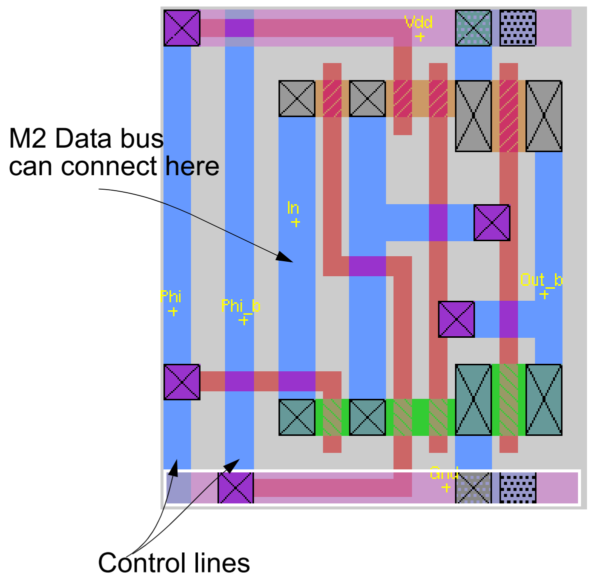
- Fixed-height cells, the height is called the bit pitch
  - Set to height of tallest cell
  - Set to allow required number of wires routing over the cell
  - 100λ works well
- Variable width
  - Width is determined by function
- Wires over cells
  - Horizontal wires carry data between function units, busses in the design

    To nearest neighbor, distant neighbor, or pass through
  - Vertical wires are the control lines for this function

    Sets up the function to be performed (e.g. Mux select)

    Often clock lines for latches

Datapath Cell Example



Power, Ground and Clock Need Wider Wires

Power and Ground

The power supply needs to be distributed to all the cells in the circuit. Resistance in these lines must be very small, since when a gate switches, its current flows through the supply lines. If the resistance of the supply lines is too large, the voltage supplied to gates will drop, which can cause the gate to malfunction. Usually you don’t want the supply to change more than 5-10% due to supply resistance.

So power supply must be on the metal layer

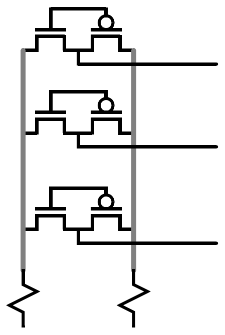
Is that enough? Usually, they have to be wider too.

R_{trans} is much greater (by 10^5) than R_{metal}

But one builds wide devices, and long wires

And in a chip there are many devices connected in parallel to the supplies. So you need still need to be careful even with metal layers, and make the special wires wide enough.

Power IR Drops



Two examples:

- Drive a 32 bit bus, total load on bus wire is 2pf

  We want the delay to be around 0.5ns

  R for each transistor needs to be < 0.25kΩ

  to meet: RC = 0.5ns

  Effective R of bits together is 250/32 = 7.5Ω

  For < 10% drop, Power R must be < ~1Ω

  That is only 8 squares.

- Must support Total Power

  Chips today dissipate 5-50W

  Implies total current is 2-20A (Power = iV)

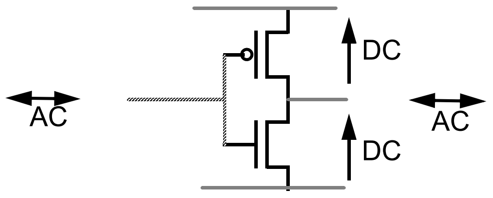
  Use many supply pins (@.2mA each), and wide wires for low R

  Grids of wire are goodness. Helps lower average resistance.

Electromigration

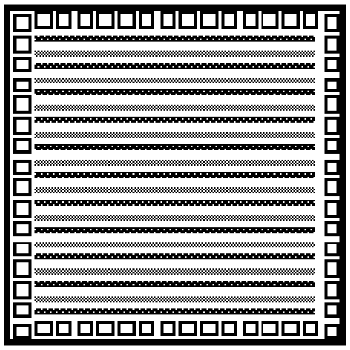
Electromigration is the phenomenon of metal atoms physically moving over time (months). Wires and contacts can thin out and break.

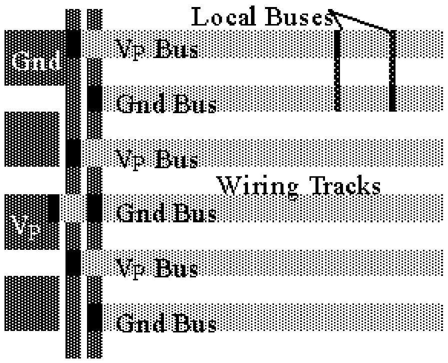
Electromigration occurs both in signal (AC=Alternating Current direction) and power wires (DC = Uniform Current direction). But problem is 10 times more severe for the same current if DC rather than AC. The DC currents occur two places: Inside of cells, and in the power busses.



So, sometimes need more contacts on transistor sources and drains to meet electromigration limits. And width of power buses must support both iR and electromigration requirements.

Power and Gnd Routing





Usually on the top layer of metal, and then distributed to the lower levels.

Power Supply Rules

The exact rules depend on the technology

- Each technology file should have its rules for resistance, electromigration

Example Rules:

- Must have a contact for each 16λ of transistor width (more is better)
- Wire must have less than 1mA/μ of width
- Power/Gnd width = Length of wire * Sum (all transistor connected to wire) / 3*$10^6\lambda$ (very approximate)

For small designs, power supply design is less of an issue

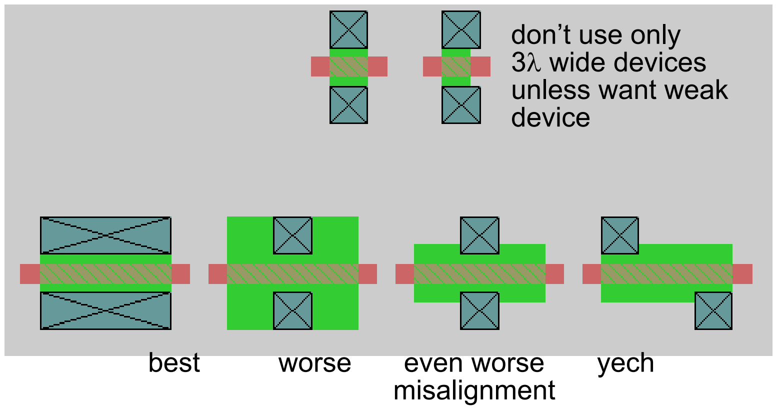
- Total power is small
- Chip is small, so wires are short

Will not be an issue in EE271

Now lets look at the components of the cells starting with transistors

Transistor Layouts

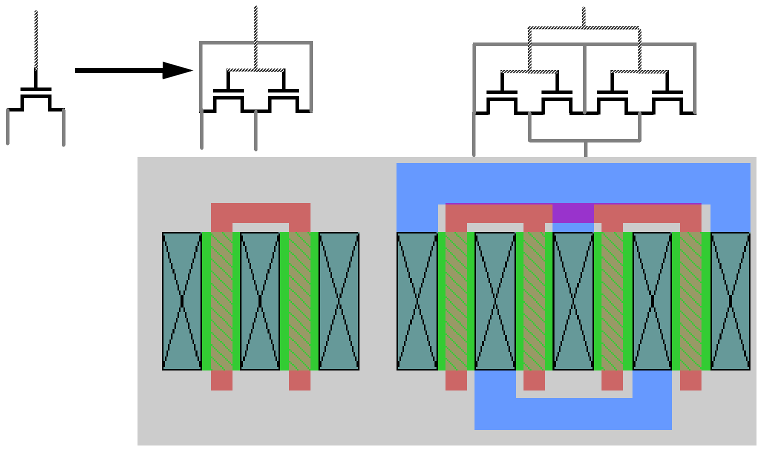
What this means for transistor layouts:



- Transistors should be at least as wide as contacts (4λ). Use as many contacts as possible for wider transistors. No diffusion anywhere else.

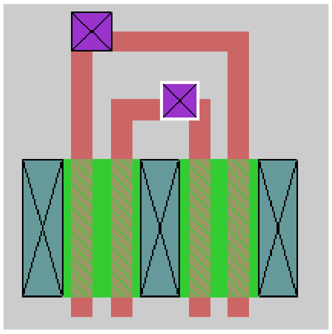
Use Folding to Reduce Diffusion

For very large transistors you end up with a bad aspect ratio. To make it a more square shape, fold the transistor. This **folding** also halves the size of the high capacitance diffusion regions of the drains.



Folding Series Gates

For series stack of devices fold the whole stack, not the individual transistors



Basic Cell Layout Issues

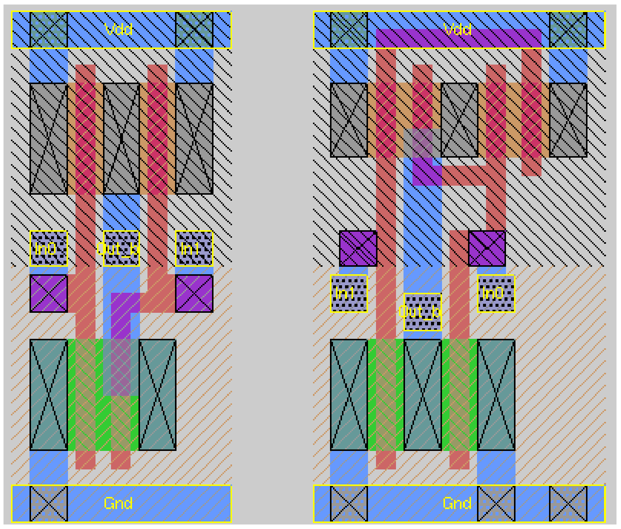
1. P-N spacing is large --> Keep pMOS together and nMOS together. Often mirror cells to keep nMOS in one cell close to nMOS in the other cell. Datapath cells sometimes mirror in both dimensions.
2. Vdd and Gnd distribution needs to be in metal, and often needs wide wires. Vdd runs near the pMOS groups, and Gnd runs near the nMOS
3. Poly can be used for intra-cell wires only
4. Layers alternate directions.
   M1 and M2 should run (predominantly) in orthogonal directions.

   Otherwise you can easily get into a situation where it is impossible to get any wire into a region.
5. Every cell should be DRC correct in isolation. If a bus or contact is created by abutment, put in both cells, and overlap the edges.
6. If you need to make several versions of something, put the common part in one cell, and then make multiple parents. Don’t squash (flatten and copy) unnecessarily. Much easier to make fixes later.

Example Std Cell Layout

Color plan

- M1 horizontal, M2 vertical; power on M1



There are well and substrate contacts under the supply lines (on top and bottom of cell)

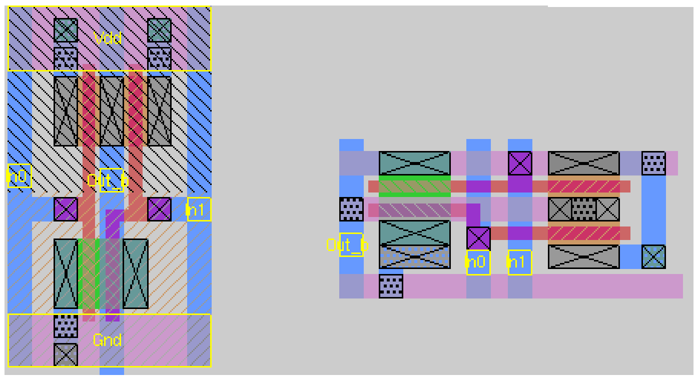
One of the poly lines jogs over to reduce the diffusion cap in the series transistor

Notice the space that a poly M2 connection requires (since it needs a poly contact spaced from a via)

Example Std Cell

Color plan

- M1 vertical, M2 horizontal; power on M2

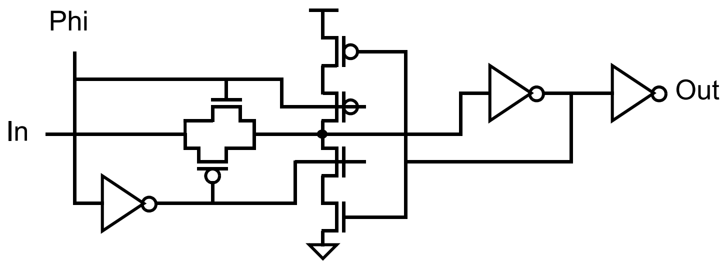


Notice the wide M2 power lines on the left cell. Also notice that the M1 wires generally can't run on top of stuff

On the left cell notice the compact M2-diff contact, by using a m2c adjacent to pdc.

Std Cell Latch

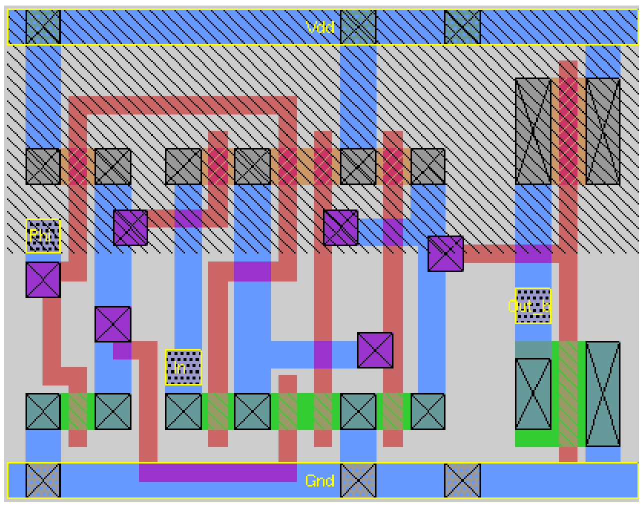
A std-cell latch is more likely to be both static and have its own inverter for the clock. Also, since we want to have a safe static latch, the feedback node would be isolated from the output.



Notice this latch requires no ratioing of transistors or capacitance.

Std Cell Latch Layout

Color plan (std cell latch): M1 horizontal, M2 vertical; power in M1



This layout has been hacked some from simply laying out poly. But only real cleverness is the merging of the TG diffusion with the tristate inverter, and the routing of the clock lines to this section. And even in this section it is still poly mostly vertical.

Datapath/Array Layout

Creating datapath cells is like Std Cell design:

- Need to come up with a consistent wire plan
- Need to make sure all the cells are the same height

But is a little more complex

- Wires need to be in the cell

  Need to think about data wires (the buses) and the control wire

    Need to think more about the application first

    Sometimes need to have wires for your neighbor cells

  Wires that route through cell must be in metal

    This goes for the control and data wires

    Must not have not have any poly jumpers in these wires

- Cell dimension can be more constrained (don't want to expand datapath for just one cell)

Datapath Cell Design

Have option of wire plan:

- Bus in M1, control in M2
- Bus in M2, control in M1

    M2 wires can run over transistors, but M1 wires consume area

And independent of this you get to choose the Vdd and Gnd routing

- Power in the control direction (vertical)
- Power in the data direction (horizontal)

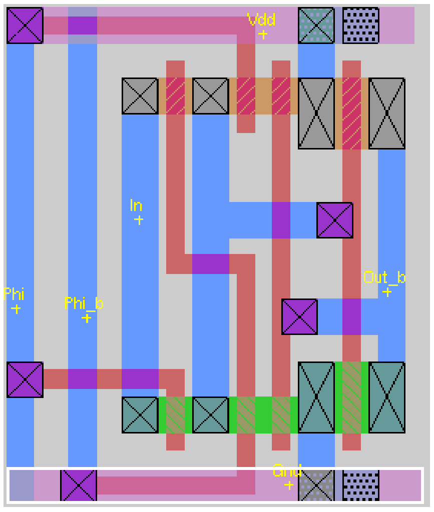
Look at two different latch implementations, with horizontal power routing

- Since many bus lines are not used by the cell, I like to place buses on M2. Then I can route these signals over the transistors and save area. I generally have more buses then control wires

Datapath Latch: M1 vert, M2 horiz

- This has horizontal data buses in M2. These bus lines run over the cell (in the parent), so they are not shown.
- Control is in M1 vertically
- Input and output have m2c so they can connect with the m2 bus lines. Sliding the contacts up and down allows them to connect to the correct bus line
- Vdd and Gnd are made to be shared (mirror adj bitslices)

Latch's output is not isolated

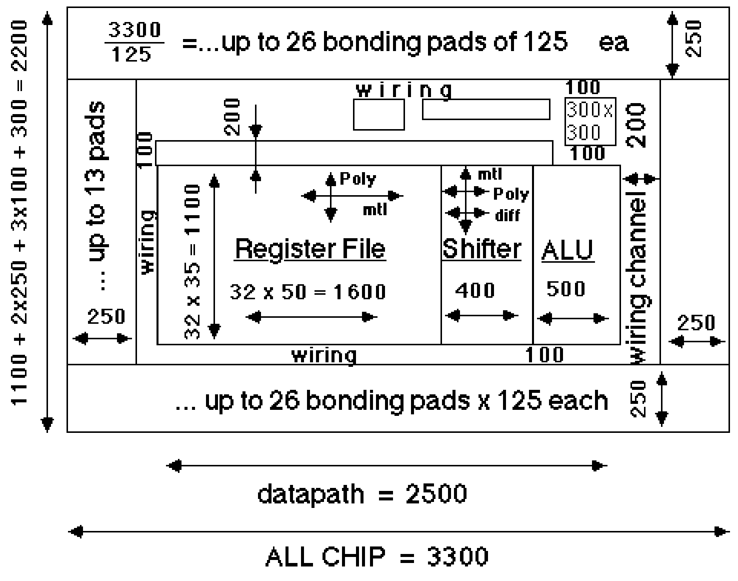


Wire Floorplanning (Color Plan)

- This is a plan of the chip/block/cell, that shows not only the subcells/transistors, but also the space needed for wires. Within a cell it is usually called the "color plan"; at the top level it is called a "floorplan".
- For both the cells and the wiring areas the dominant direction of metal wires are shown. If poly is used for wiring its direction should be shown too.
- The floorplan should also note how Vdd and Gnd are being distributed, and the width of the wires.

A little planning up front will save lots of time in the back end.

RISC II Floorplan



3300/125 =...up to 26 bonding pads of 125 ea
250
1100 + 2x250 + 3x100 + 300 = 2200
... up to 13 pads
250
wiring
100
200
300x300
100
100
200
Poly
mtl
Register File
32 x 35 = 1100
32 x 50 = 1600
mtl
Poly
diff
Shifter
400
ALU
500
wiring channel
250
wiring
100
... up to 26 bonding pads x 125 each
250
datapath = 2500
ALL CHIP = 3300