

Introduction to 3BA4-II

Dr. Andrew Butterfield

Timetable:
Monday, 1pm, H5
Tuesday, 10am, M17
Tuesday, 2pm, M20

Topic:
Integrated Circuit *Design*

Textbooks:
Weste & Eshraghian:
Principles of CMOS VLSI Design
Mead & Conway:
Introduction to VLSI Design

Web Page:
`http://www.cs.tcd.ie/Andrew.Butterfield/3BA4/`

CMOS

Complementary Metal-Oxide-Semiconductor

Two types of MOSFETs:
n-channel p-channel △1

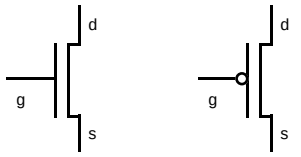
(many alternative notations) △2

CMOS construction:
inputs feeding two networks
one pullup, connecting output to Power, with only p-type devices
one pulldown, connecting output to Ground, with only n-type devices △3

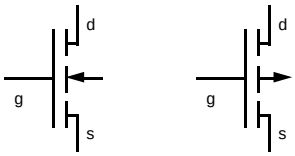
Desired behaviour, for given inputs:
if we want 0 output,
produce path in pulldown network
if we want 1 output,
produce path in pullup network

3BA4—Part II: Lecture 1.2(CMOS)

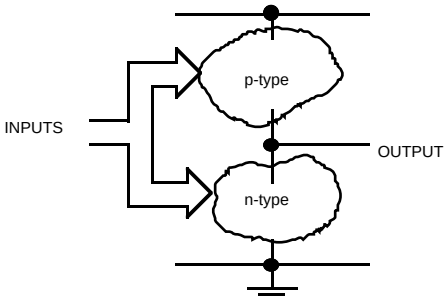
△1 — We use the following symbols for n-channel and p-channel:



△2 — Alternative symbols are:



△3 — The general structure of a CMOS circuit is:



Example CMOS Circuits

Behaviour of MOSFETS:
- n-type: Logic '1' on gate, switch Closed
- p-type: Logic '0' on gate, switch Closed

The Inverter:
- a single p-type pullup MOSFET
- a single n-type pulldown MOSFET

△1

Consider a more complex example:

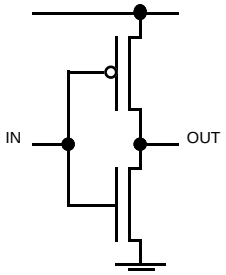
$\text{OUT} = \overline{A \cdot ((B \cdot C) + D)}$ △2

What can single CMOS structures do ?

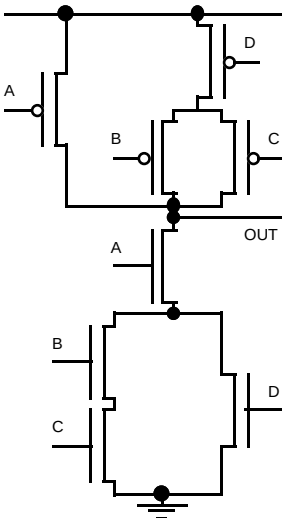
Function	Do-able ?
$\overline{A}$	yes
$\overline{A \cdot ((B \cdot C) + D)}$	yes
$A \cdot B$	no
$A \oplus B$	no
$A \cdot ((B \cdot C) + D)$	no
$\overline{A} \cdot \overline{B}$	yes

3BA4—Part II: Lecture 1.3(Example CMOS Circuits)

△1 — A CMOS Inverter:



△2 — The function $\overline{A \cdot ((B \cdot C) + D)}$:



Single Structure Functions

Single CMOS structures can implement functions of the form:
expression with only AND, OR and variables

Given a truth table, how do we establish if a function can be written like this ?

Consider the following functions as examples:

A	B	$\overline{AB}$	AB	$A \oplus B$
0	0	1	0	0
0	1	1	0	1
1	0	1	0	1
1	1	0	1	0

We can construct a *Hasse Diagram* for 2-inputs,
showing the function $\overline{AB}$. △1

We can build a single circuit if

(i) bottom is labelled with (1) or •

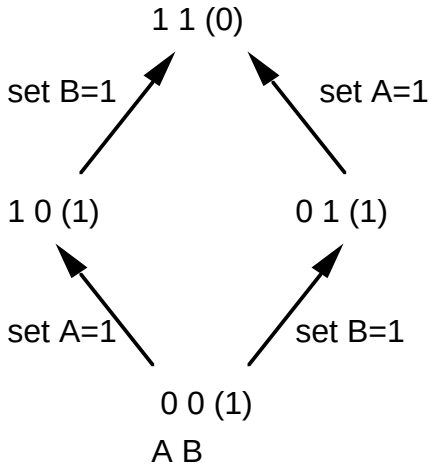
(ii) bottom is labelled with (0) or ∘

(iii) any bottom-top path has labels
which change only once, from (1) to (0)

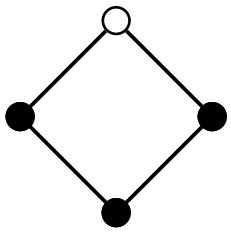
△2,3,4

3BA4—Part II: Lecture 1.4(Single Structure Functions)

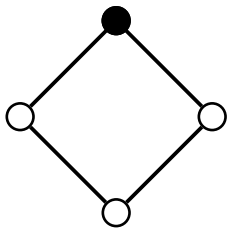
△1 — A fully labelled Hasse Diagram for $\overline{AB}$:



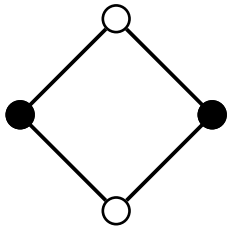
△2 — A compact Hasse Diagram for $\overline{AB}$:



△3 — A compact Hasse Diagram for AB:



△4 — A compact Hasse Diagram for $A \oplus B$:



Monotonic Bit-Functions

Consider the following ordering ($\preceq$)
on bit-strings of the same length (n):

$$a = a_{n-1}a_{n-2}\ldots a_1a_0 \preceq b_{n-1}b_{n-2}\ldots b_1b_0 = b$$

if b can be obtained from a
by changing zero or more 0s in a to 1s in b

if $a \preceq b$, then a will be connected to b
in a Hasse Diagram by zero or more arrows.

We say a function f is *monotonic decreasing* if

$$a \preceq b \implies f(a) \succeq f(y)$$

These are exactly the functions implementable as single CMOS structures

(i) When all inputs are 0, all pulldowns are open, all pullups are closed, so output is 1. $\triangle 1$

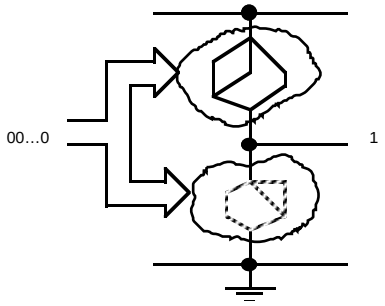
(ii) When all inputs are 1, all pulldowns are closed, all pullups are open, so output is 0. $\triangle 2$

(iii) As we change 0s to 1s, we open pullups, and close pulldowns, so the only possible output transition is 1 to 0. $\triangle 3$

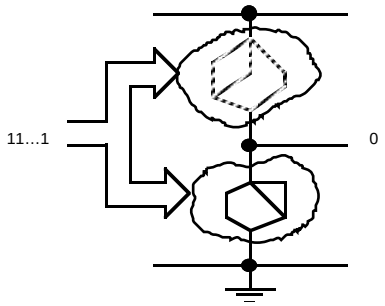
3BA4—Part II: Lecture 1.5(Monotonic Bit-Functions)

Three diagrams illustrating CMOS structure behaviour.

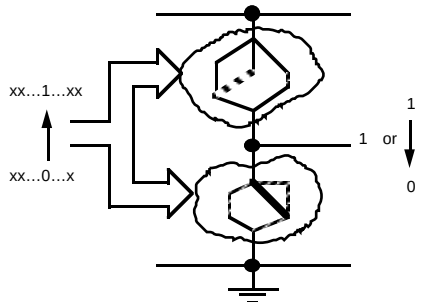
△1 — All inputs 0:



△2 — All inputs 1:



△3 — An input changing from 0 to 1:



Examples of 3-input functions

A Hasse diagram for $\overline{A(B+C)}$ shows that it is implementable as a single CMOS structure. △1

It also stresses the fact that $\preceq$ is a *partial* order.

$$010 \not\preceq 101 \qquad \text{and} \qquad 101 \not\preceq 010$$

From a Hasse diagram for $A \oplus B \oplus C$ we find: △2

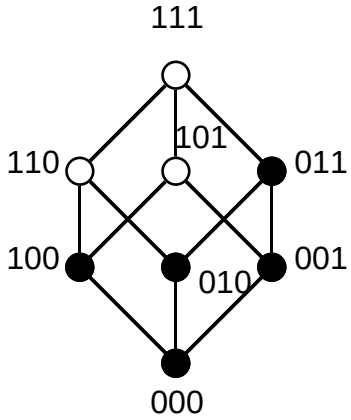
Every single-bit input change
causes output change.

EXOR is in some sense
the "most non-monotonic" bit function

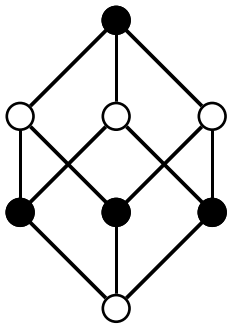
EXOR is the most expensive gate to build
in CMOS, (or any other technology)

3BA4—Part II: Lecture 2.1(Examples of 3-input functions)

△1 — Labelled Hasse diagram for $\overline{A(B+C)}$



△2 — Compact Hasse diagram for $A \oplus B \oplus C$



Electronic Behaviour of MOSFETs

We consider an n-channel device to begin with.

Voltages: Drain-Source (v_{DS}), Gate-Source (v_{GS})
Currents: Drain-Source (i_{DS}), Gate (i_G) △1

$$i_G = 0$$

How do we identify source and drain ?
The *more positive* is designated the *drain*.

$$v_{DS} \geq 0$$

Behaviour: $i_{DS} =$

$$\begin{array}{ll} 0, & \textbf{if } v_{GS} \leq V_{TN} \\ & \textbf{if } v_{GS} \geq V_{TN} \textbf{ and} \\ \beta_N(v_{DS}(v_{GS}-V_{TN}) - v_{DS}^2/2), & \textbf{if } v_{DS} \leq v_{GS} - V_{TN} \\ & \textbf{or} \\ \beta_N(v_{GS}-V_{TN})^2/2, & \textbf{if } v_{DS} \geq v_{GS} - V_{TN} \end{array}$$

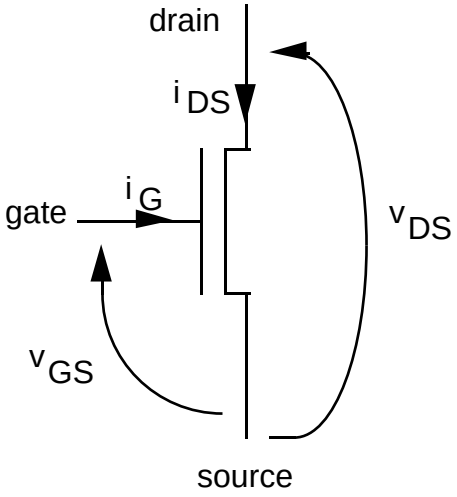
The important parameter is β_N, the MOSFET *transconductance*, as

$$i_{DS} \propto \beta_N$$

For p-channel devices, we multiply by (-1), and observe that the drain is the more *negative* terminal.

3BA4—Part II: Lecture 2.2(Electronic Behaviour of MOSFETs)

△1 — A voltage and current-labelled diagram of an n-channel MOSFET



CMOS Inverter Characteristic

The *Transfer Characteristic* of an inverter maps v_{IN} to v_{OUT}: △1

$$v_{OUT} = g(v_{IN})$$

We shall always assume: Power is $5V$, $V_{TN} = +1V$, $V_{TP} = -1V$.

We assume also, to start, that $\beta_N = \beta_P$.

The shape of the transfer curve depends on the power voltage, V_{TN}, V_{TP}, and the *ratio* β_N/β_P △2

For an optimal speed and noise-immunity trade-off, we prefer to have $\beta_N/\beta_P = 1$

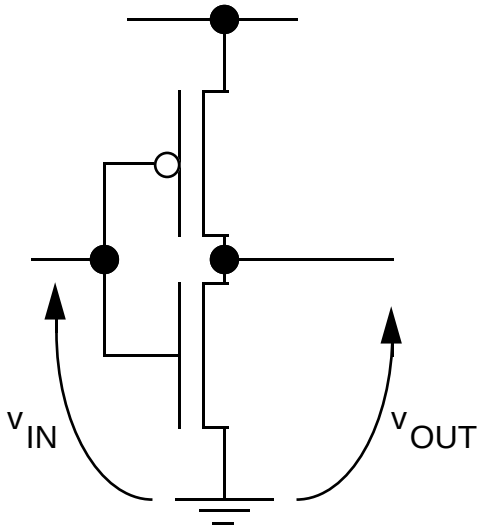
Usually, Power voltage, V_{TN} and V_{TP} are determined by manufacturer

Digital CMOS Designer's only concern is β_N/β_P !

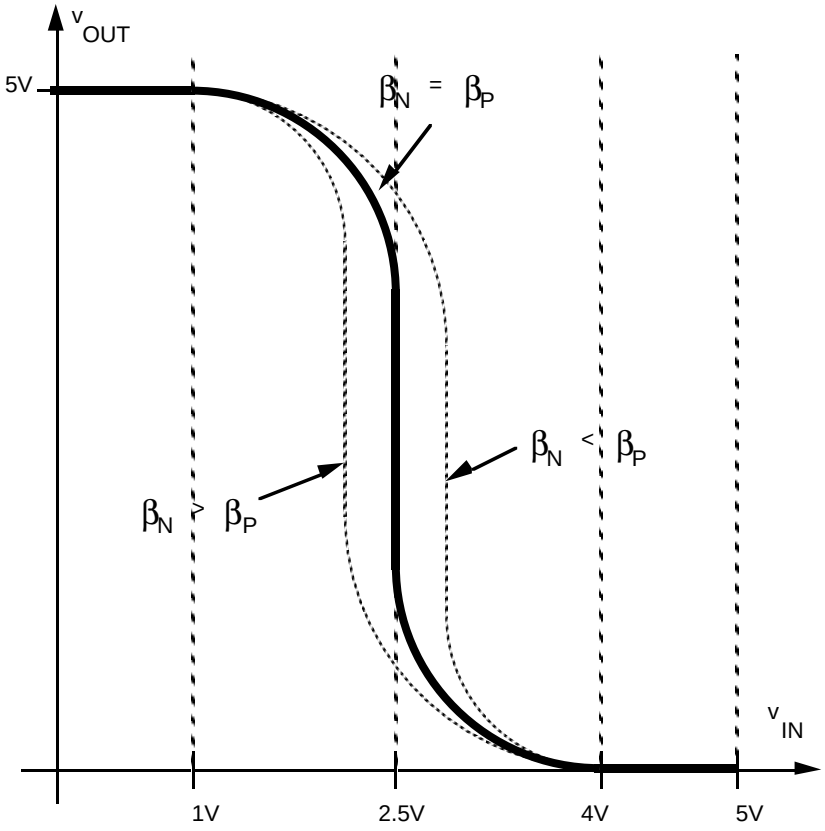
(except for high perfmormance — this gets hairy)

3BA4—Part II: Lecture 2.3(CMOS Inverter Characteristic)

△1 — A CMOS inverter showing v_{IN} and v_{OUT}



△2 — A CMOS inverter transfer curve:



What Determines β_N and β_P ?

n-type MOSFET: Gate electrode, thin insulator p-type channel,
between two n-type source/drain (s/d) regions. △1

Key Parameters:

L : Channel Length (along i_{DS} direction)
W : Channel Width (across i_{DS} direction)
t : Insulator Thickness
ε_i : Insulator Permittivity
μ : Charge Carrier Mobility

The transconductance is given by:

$$\beta = \mu \frac{\varepsilon_i}{t} \cdot \frac{W}{L} = \mu C_i \frac{W}{L}$$

where $C_i = \varepsilon_i / t$ is the capacitance/unit area of the gate.

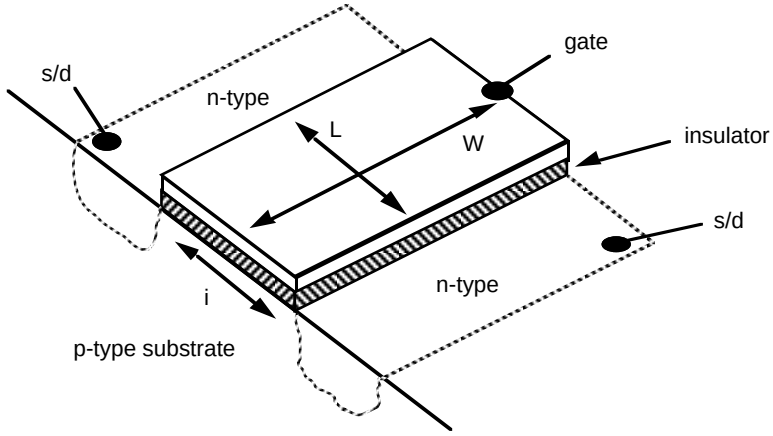
Parameters ε_i, t and μ are fixed by manufacturer.

Parameters W and L are set by designer
(within limits)

One complication remains: $\mu_N \approx 3\mu_P$ (!)

3BA4—Part II: Lecture 2.4(What Determines β_N and β_P ?)

△1 — An oblique sectioned view of an n-channel MOSFET:



Compensating for $\beta_P \approx \beta_N/3$

We want $\beta_N = \beta_P$, where $\beta = \mu C_i W/L$,
but $\mu_N \approx 3\mu_P$ — we must compensate

In practice, devices have a minimum size
(circa $6\mu m$ in 1980, about $0.15\mu m$ in 1999).

Use current minimum size as unit of length
(sometimes called *lambda*, or λ)

Set all channel lengths to minimum $(L = 1)$
(this maximises gate switching speed)

Our balancing condition becomes:

$$\beta_N = \mu_N C_i W_N \qquad = \qquad \mu_P C_i W_P = \beta_P$$

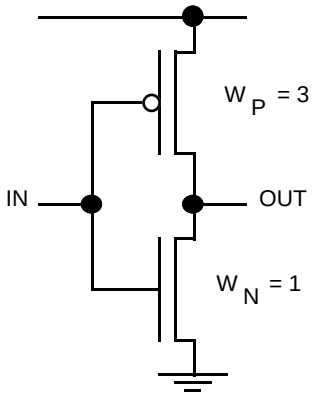
We can only determine W_N, W_P as designers, so we get:

$$W_P = 3 \times W_N$$

△1

3BA4—Part II: Lecture 3.1(Compensating for $\beta_P \approx \beta_N/3$)

△1 — We show an Inverter labelled with correct sizes:



Effective Resistance Networks

Let us consider the example of a NAND gate △1

What is β for a network ?

The reciprocal of β is "effective resistance" R

$$R = 1/\beta \qquad \beta \propto W \qquad R \propto 1/W$$

This is approximate only, but works well enough

For a series connection, the resistance is the sum of the resistances of the components. △2

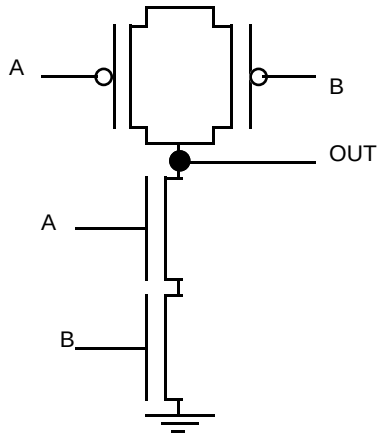
For a parallel connection, we might expect the rule for resistor in parallel to apply:

$$R = \frac{R_1 R_2}{R_1 + R_2}$$

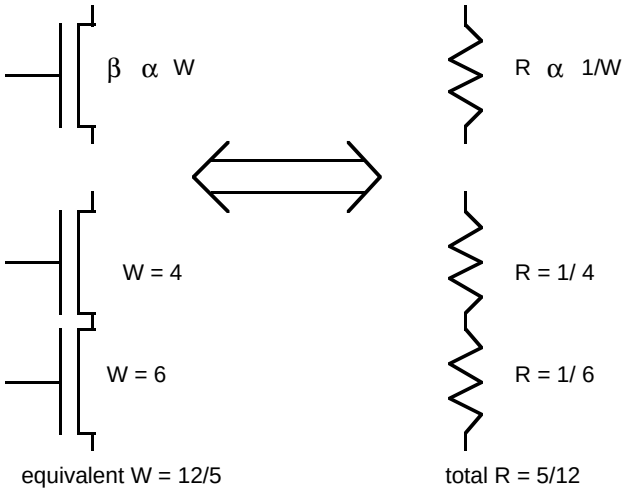
However we find the parallel case is in one sense more complicated, while in another sense it becomes simpler !

3BA4—Part II: Lecture 3.2(Effective Resistance Networks)

△1 — Circuit Diagram for a NAND Gate



△2 — Diagram showing series FETs viewed as equivalent Resistances



Handling Parallel Paths

Reminder: balancing βs to get symmetrical, fats changeover, with good noise properties.

NAND switches 0 to 1 in 2 distinct ways:

(i) A single pullup has just closed

(ii) both pullups have just closed △1

Which case do we use for balancing ?

Worst Case: single branch working alone.

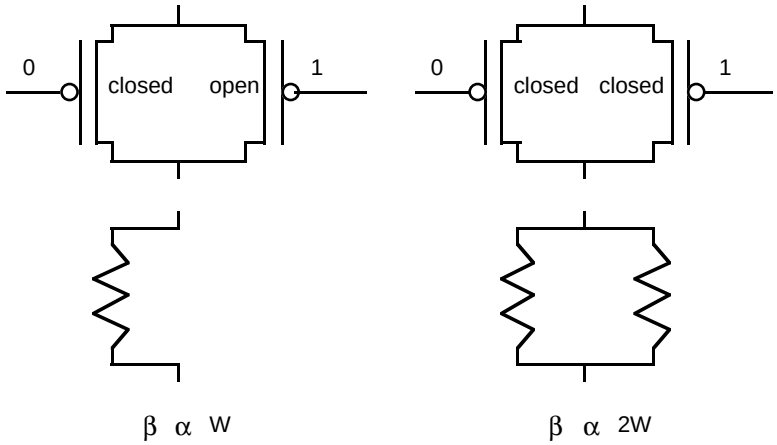
Analyse and consider parallel branches separately. △2

Exploit symmetry:

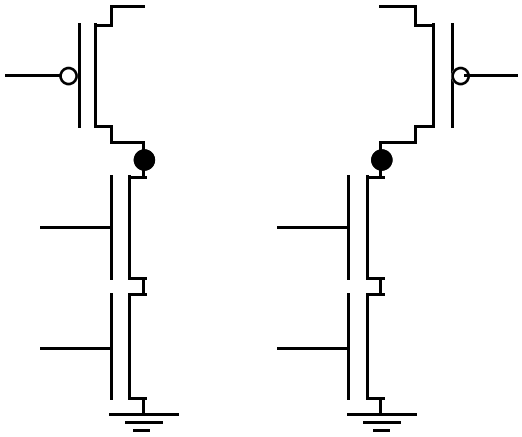
Two pullups are similar
Two pulldowns are similar

3BA4—Part II: Lecture 3.3(Handling Parallel Paths)

△1 — showing the distinction between one and both pullups acting to produce a 0 to 1 transition:



△2 — showing the two branches needing balancing



Lets Calculate !

We assume co-efficient $\mu_N C_i = 1$ for simplicity

$\beta_{PU} = \beta_P = W_P/3$

$\beta_{PD} = 1/R_{PD} \qquad R_{PD} = 2R_N \qquad R_N = 1/\beta_N$

$\beta_N = W_N$

$\implies \quad R_N = 1/W_N$

$\implies \quad R_{PD} = 2/W_N$

$\implies \quad \beta_{PD} = W_N/2$

We find that $W_N/2 = W_P/3$ to balance the gate

We have some freedom to choose what W_N and W_P are.

Two main possibilities:
Poss. 1 — let smallest have $W = 1$

$$W_N = 1 \qquad W_P = 1.5$$

Poss. 2 — Choose smallest values keeping everything integral

$$W_N = 2 \qquad W_P = 3$$

(often required by CAD tools)

3BA4—Part II: Lecture 3.4(Lets Calculate !)

A Simpler Way

There is a simple procedure that eliminates a lot of tedious error-prone algebra

△1

Step 1. Set all widths equal to 1

Step 2. Multiply all p-type widths by 3

Step 3.

For a series connection of length n,
multiply all the widths by n.

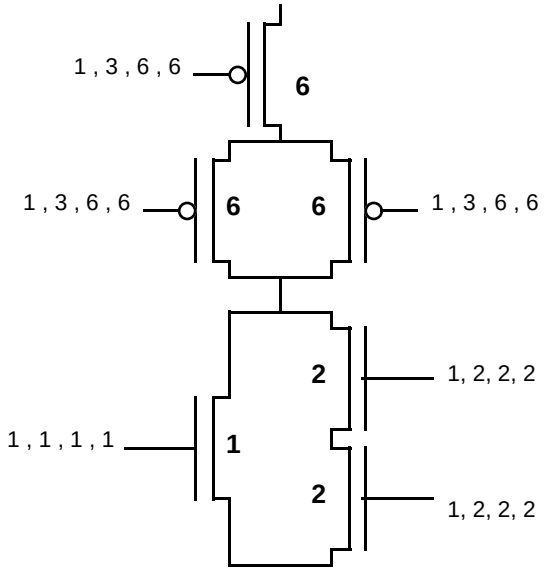
Step 4.

Scale everything to minimise widths,
(subject to keeping them integral, if reqd.)

Notes:
— Steps 1 & 2 can be combined
— A MOSFET in several parallel paths
is only multiplied once during Step 3.
— A MOSFET on several parallel paths
of different lengths needs special treatment

3BA4—Part II: Lecture 3.5(A Simpler Way)

△1 — showing complicated circuit sized the easier way



The sequence of 4 comma-separated numbers show the outcome of each of the four steps.

Manufacturing Process for ICs

Silicon (Si) — semiconductor (wanted very pure)

Sand(SiO_2) → heat → Si

Silicon Dioxide (SiO_2,glass) - good insulator

Impurities (n-type,p-type) added
to control electronic behaviour

Pure Si arrives as tubular ingots △1

Ingots sliced into *Wafers* (3 to 8 inches) △2

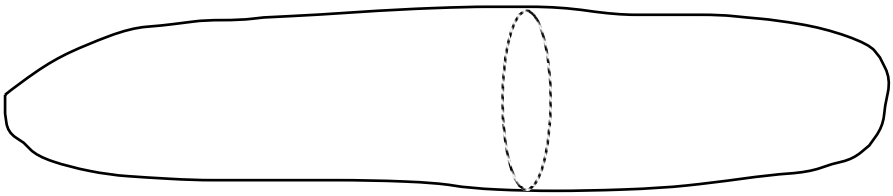
Chip implemented as small rectangle (*Die*)
side dimensions typically a few mm. △3

The Die is the "unit" of design.

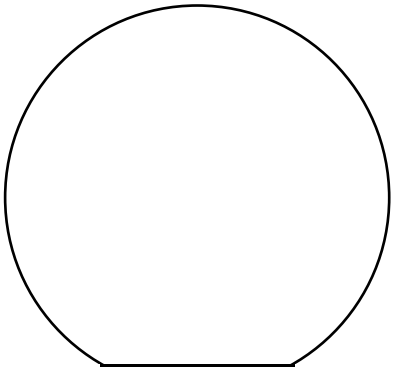
Identical copies of Die tiled over wafer. △4

3BA4—Part II: Lecture 4.1(Manufacturing Process for ICs)

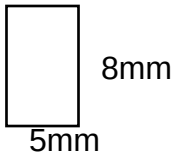
△1 — of Si ingot:

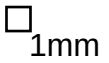


△2 — of Si wafer:

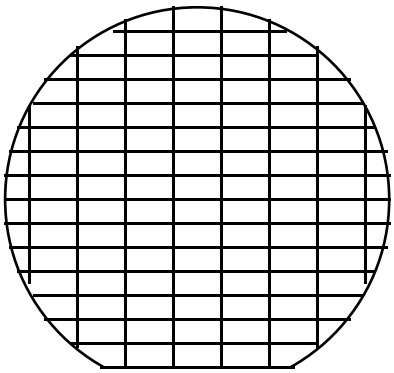


△3 — of sample dies:





△4 — of wafer tiled with dies:



IC Design Documents

Design "Documents" specify shapes on *"Layers"*.

Shapes on different layers overlap to make devices

A *Mask* is constructed per layer. △1

Mask comprises opaque material on glass

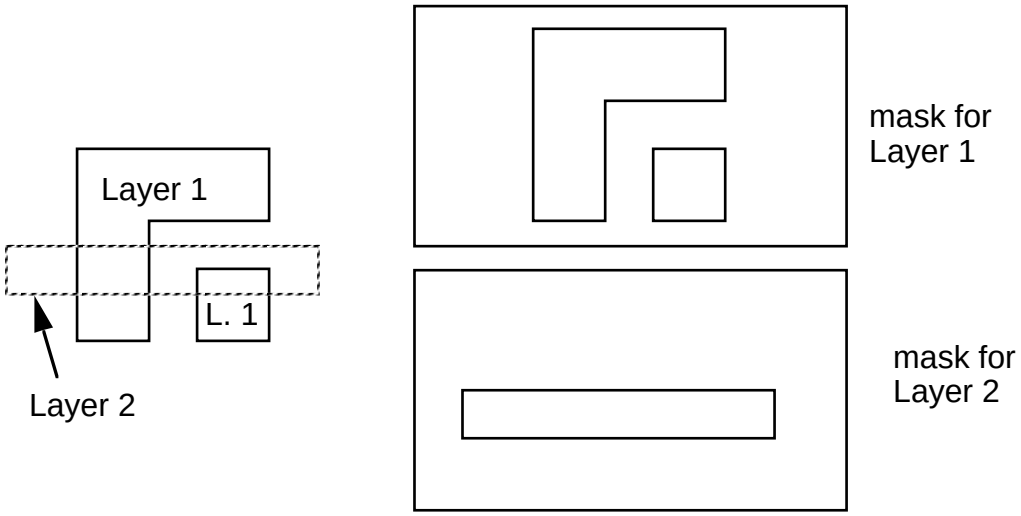
— with some opaque material removed △2

Masks cover entire wafer

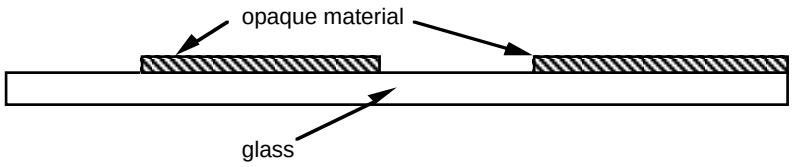
Masks used to generate regions of desired shape — material determined by layer.

3BA4—Part II: Lecture 4.2(IC Design Documents)

△1 — of document layers producing separate masks



△2 — of mask in cross-section



Patterning Steps

Using Mask to generate material regions:

1. Cover wafer with *Photo-Resist*
2. Expose photo-resist to light through mask
3. Develop photo-resist — exposed resist removed, unexposed resist remains △1
4. Process wafer to add/remove material

   4a — add (diffusion, sputter ...) △2

   OR

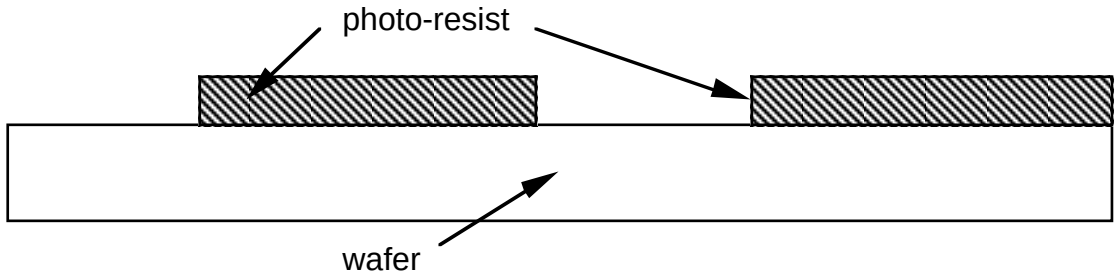
   4b — remove (etch ...) △3
5. Remove remaining resist

Variations

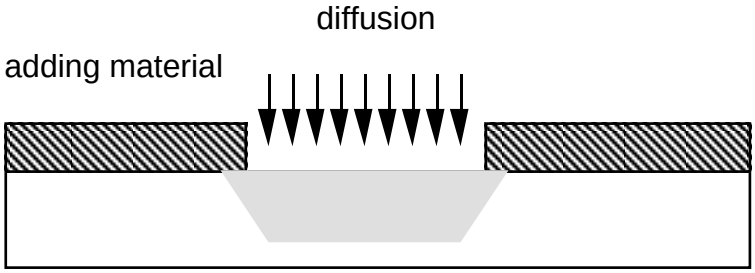
may apply material before Step 1
then etch in step 4b.

3BA4—Part II: Lecture 4.3(Patterning Steps)

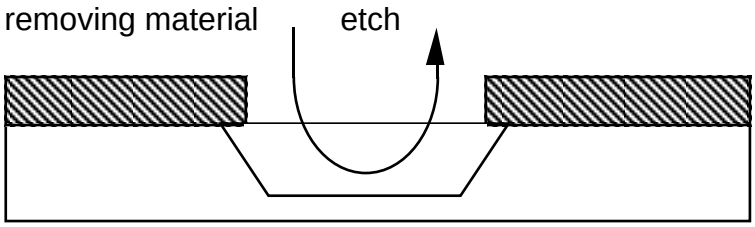
△1 — of state after Step 3.



△2 — of adding material in Step 4a



△3 — of removing material in Step 4b



Producing an nMOS Device

Design uses 2 layers — *nDiff* and *Poly* △1

nDiff : n-type impurities diffused into wafer

Poly : Poly-crystalline Silicon used as wiring

Transistor constructed by crossing nDiff and Poly. △2

1. Cover wafer with thick SiO_2 *(Field Oxide)* △3

Why most ICs are built on silicon !!!

2. use nDiff mask to etch SiO_2. △4

3a. Grow thin oxide layer *(Thinox)*.

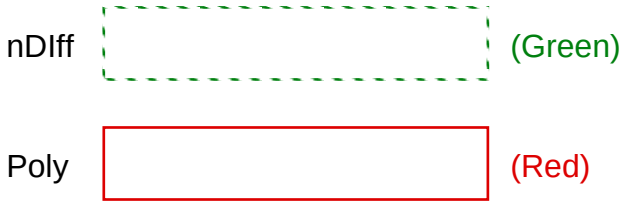
3b. Use Poly mask to pattern Poly and Thinox. △5

4. Diffuse n-type impurities into substrate △6

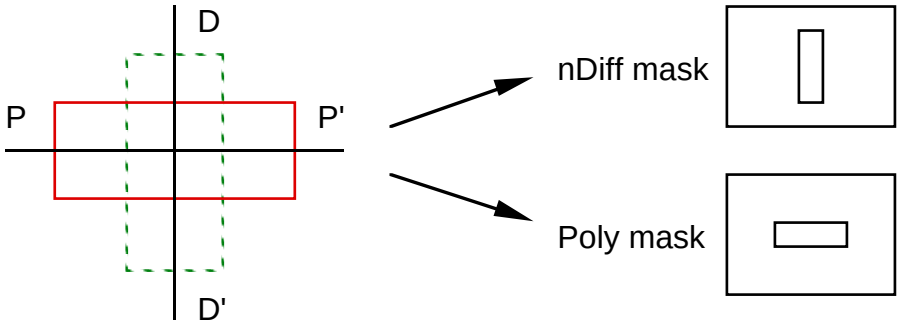
Self-Aligned Polysilicon-Gate Process △7

3BA4—Part II: Lecture 4.4(Producing an nMOS Device)

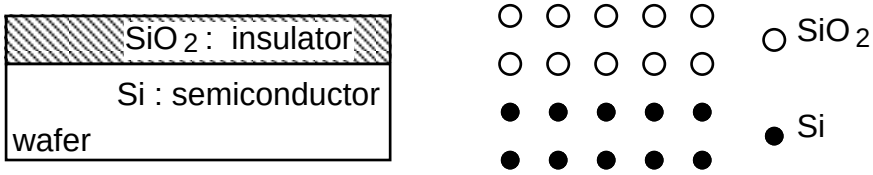
△1 — of nMOS device layers:



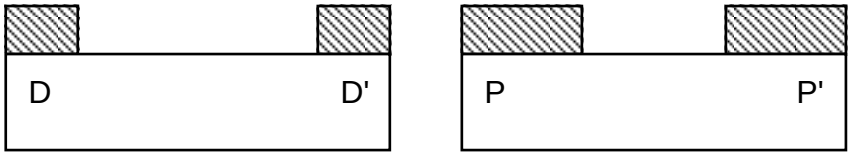
△2 — of nMOS device and masks, with cross section lines indicated:



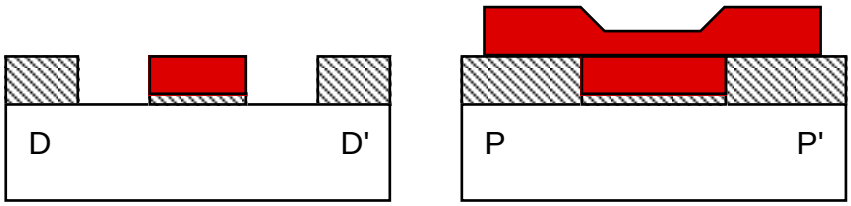
△3 — of wafer covered with oxide, also showing nice regular structure:



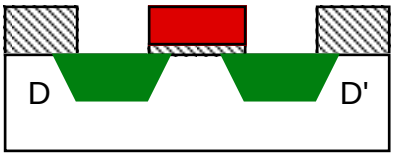
△4 — Cross sections of wafer after field oxide patterned by nDiff mask:



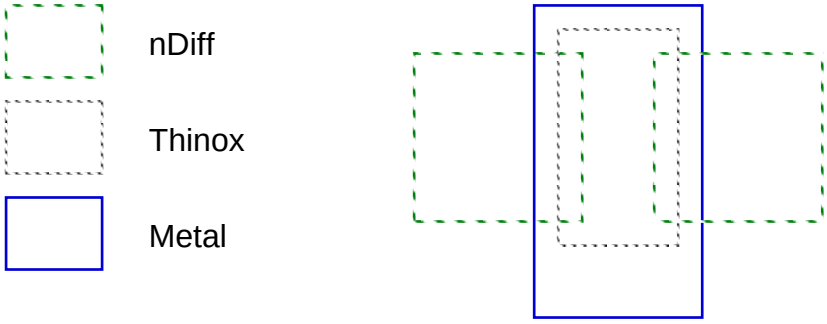
△5 — Cross sections of wafer after Poly/Thinox patterned by Poly mask:



△6 — Cross sections of wafer after n-type impurities are diffused:



△7 — showing how it was done prior to self-aligned gate processes:



Mixing nMOS and pMOS

nMOS have n-type s/d regions in p-type substrate
pMOS have p-type s/d regions in n-type substrate △1

How do we combine the two ?

One solution: Silicon on Sapphire (SOS) △2
Isolates devices, but expensive

Cheaper solution:
build *Wells* of one substrate type in another.

nWell Technology:

p-type substrate

— n-channel devices built here

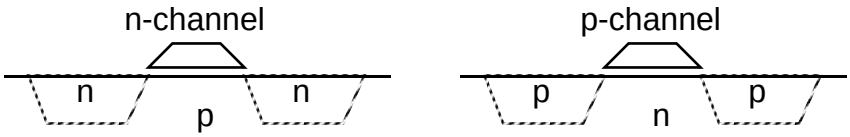
n-type well embedded into substrate

— p-channel devices built here △3

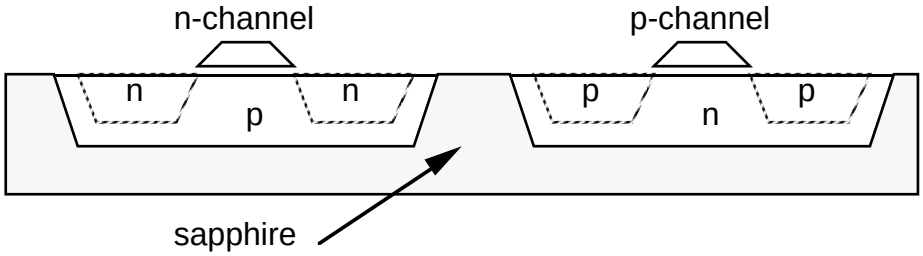
Two new layers: nWell and pDiff △4

3BA4—Part II: Lecture 5.1(TITLE)

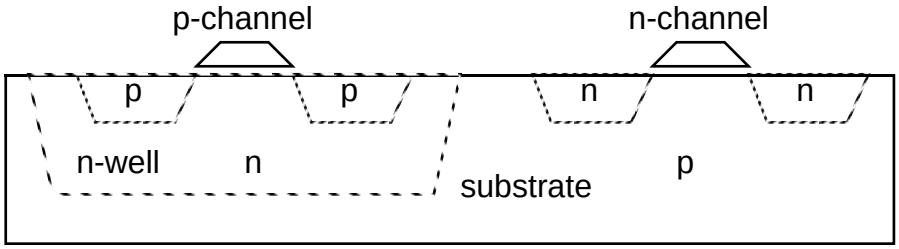
△1 — of nMOS and pMOS in cross-section:



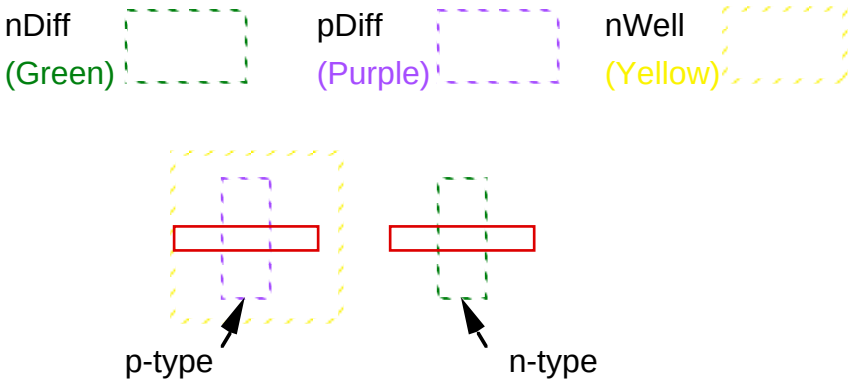
△2 — of nMOS and pMOS in SOS:



△3 — of nMOS and pMOS in nWell:



△4 — of new pDiff and nWell layers:



Connecting Devices

Poly and Diff act as wiring,
but cannot cross each other without making FETs.

Diff cannot cross nWell boundaries either

Need wiring layer — *Metal1*.

(often have Metal2,3, ... as well.)

Aluminium - layered on SiO_2 above Poly. △1

It does not interact with lower layers
(with one exception, see later ...) △2

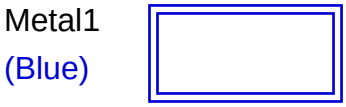
How do we connect Metal to other layers ?

New "layer" called *CCut* (contact-cut) △3

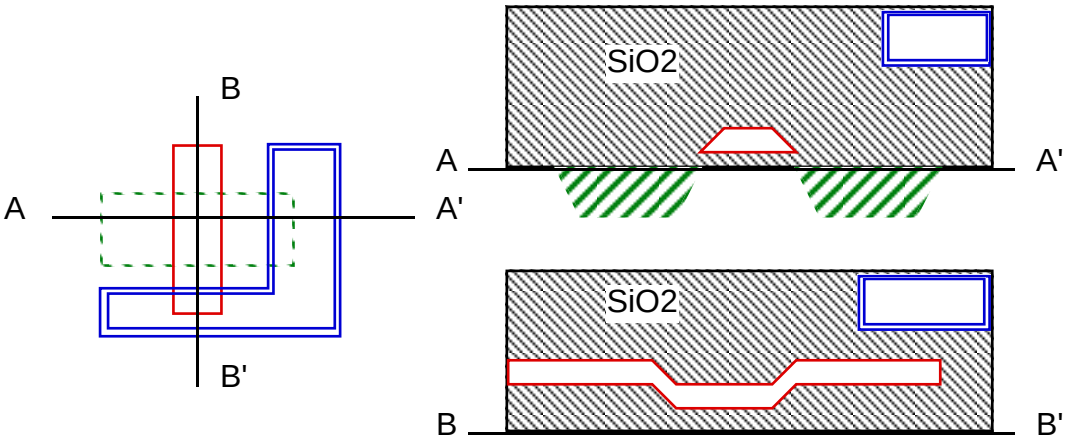
Specifies where holes should be cut,
through $SiO2$, to underlying layers.

3BA4—Part II: Lecture 5.2(Connecting Devices)

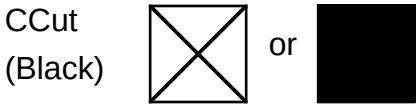
△1 — of Metal Layer:



△2 — of Metal Layer passing over Poly and Diff:



△3 — showing CCut layer



Joining Metal to Poly/Diff

After Poly layer is done . . .

(i) Grow oxide over it

(ii) Use CCut mask
to cut holes to underlying Poly or Diff

(iii) place and pattern Metal1 △1

Contact Cut holes reach down to to Poly if below,
or else proceed to substrate surface.

Design Rules — "graphical syntax"

(i) ensure design makes some sense

(ii) ensure design can be reliably manufactured

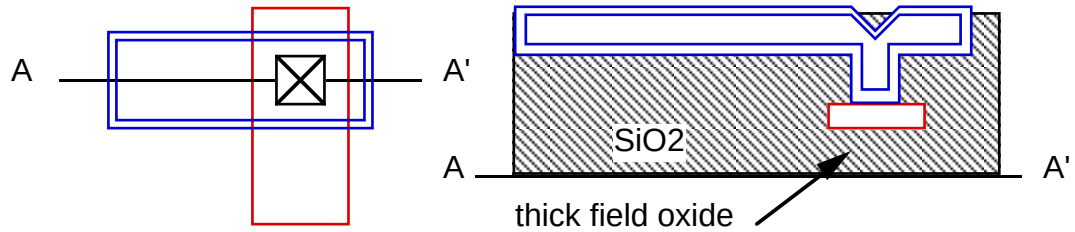
A CCut Design Rule:
Never place CCut over Poly/Diff overlap (Gate)

Why ? Metal can spike through Poly △2

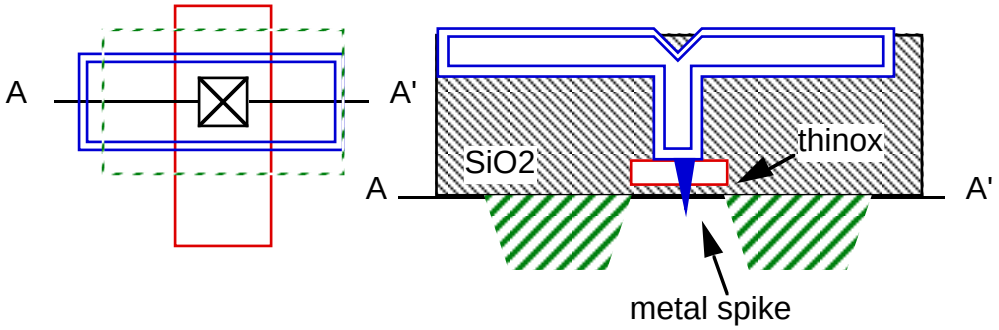
Note: Metal1 may cross over Gate regions

3BA4—Part II: Lecture 5.3(Joining Metal to Poly/Diff)

△1 — of Metal-Poly contact



△2 — of Metal-Gate contact showing spiking



TITLE

item

△1