INTERNATIONAL ATOMIC ENERGY AGENCY
UNITED NATIONS EDUCATIONAL, SCIENTIFIC AND CULTURAL ORGANIZATION
INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS
I.C.T.P., P.O. BOX 586, 34100 TRIESTE, ITALY, CABLE CENTRATOM TRIESTE

SMR/443 - 2

ICTP - INFN COURSE IN
"BASIC VLSI DESIGN TECHNIQUES"
6 November - 1 December 1989

VLSI DESIGN

S. TURRINI
Microprocessor Laboratory
ICTP
Trieste
Italy

Lectures on
basic VLSI design
techniques

ICTP — INFN
6 November — 1 December 1989

VLSI Design

by

Silvio Turrini

Microprocessor Laboratory
Miramare - Trieste
ITALY

Today's available VLSI technologies (Digital Circuits)

- MOS (CMOS dominant)
- Bipolar (ECL, I2L)
- GaAs (?)

VLSI advantages:
- an entire complex system on a chip
- staggering features (speed, reliability, complexity)
- low power consumption.

Design techniques:

- Full custom design. The best performance, but tedious and very long design time if no state of art CAD available.
  High costs, long turn-around. Expert designers are required.

- Standard cells. Medium performance, quick design, medium costs (all masks still required) medium turn-around. CAD available.

- Gate Arrays. Low performance, but the best choice for prototyping. Extremely fast design time. No specific expertise required.
  Efficient CAD available. Low costs

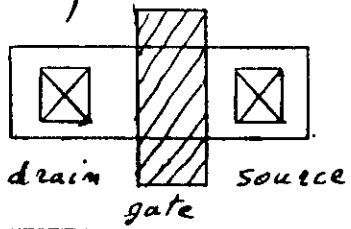
What caused VLSI to became a reality ?

- Technology improvements
- But especially CAD !

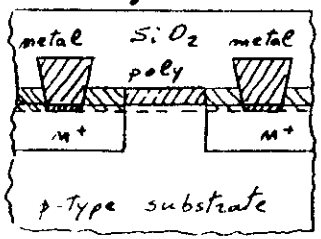
A bunch of experienced people can design a new system (CPU, MMU, FPU) in a year or so.

Basics of the CMOS process

The traditional procedure uses a photolithographic + chemical process.
Transistors, interconnections and additional devices, are built on a very small area of a silicon slice (4", 6") by alternating chemical and photolithographic procedures. Geometrical entities (rectangles of silicon (doped) and metal, with different characteristics) are built in layers. Different layers with different properties can be separated by $Si O_2$ or interact if a contact is needed.



top view

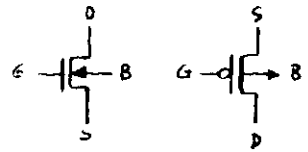


cross section

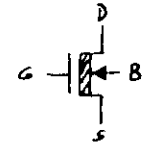
MOS as a switch

3 kinds of devices:

enhancement



depletion



used as an active load

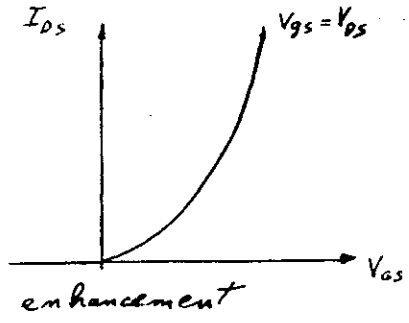
n-type depletion

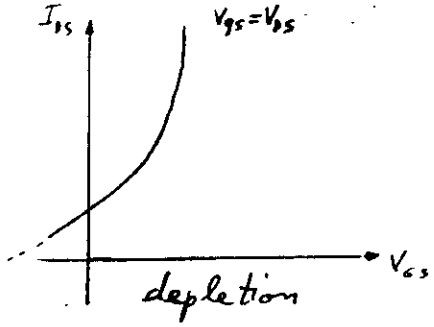
By definition the source is in a N-type the terminal closer GND P-type closer Vdd

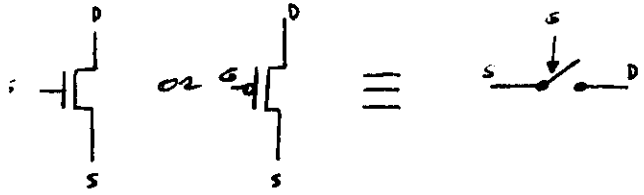
As a first approximation the voltage between the gate and source of the device, controls the current between drain and source

N-type	P-type	Depletion
$V_{gs} > 0$ ON	$V_{gs} = 0$ ON	$V_{gs} = 0$ ON
$V_{gs} = 0$ OFF	$V_{gs} > 0$ OFF	$V_{gs} < 0$ OFF

The real output characteristics are:



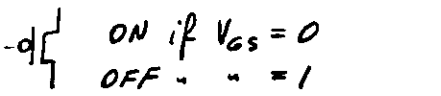




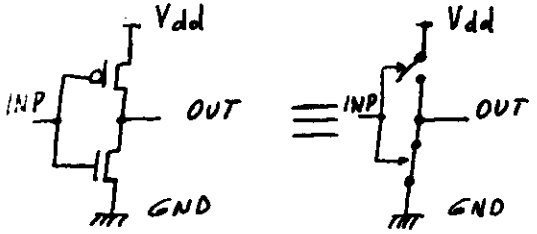
Vdd = the most positi
voltage in the ci

GND = the reference

In boolean logic we consider only 2 values { 0 GND ; 1 Vdd



The basic circuit
the inverter (NOT)

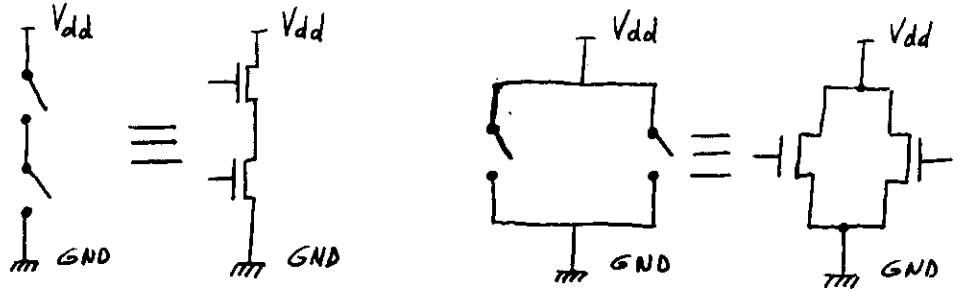


If inp = 0 then out = 1
else out = 0

This is called static inverter. Static logic (or restore, is defined as no direct path between Vdd and GN when the device is in a stable state.

Using the analogy with switches we can design other, more complex functions, such as nand, nor, e

Remember that all the gates designed in this way, will have a complemented output.

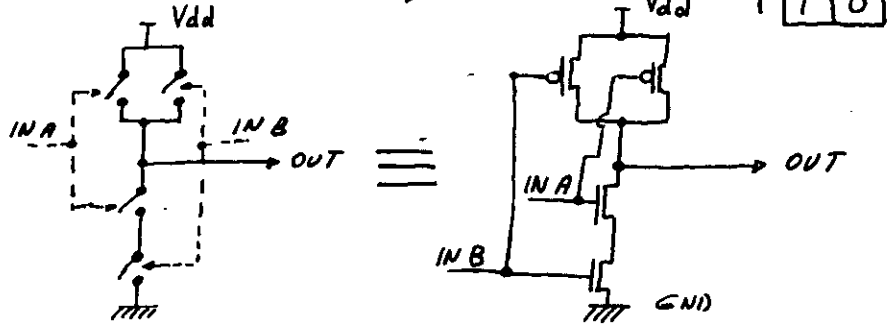


In order to ensure that no direct path exists between Vdd and GND, we are going to use De Morgan laws and apply them to the two sets of N-type and P-type transistors.

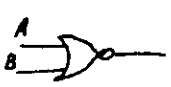
Remember: because N-type transistors (switches) and P-type transistors are controlled and connected to reversed voltages, the functions generated by the two sets must be one the complement of the other

Example NAND

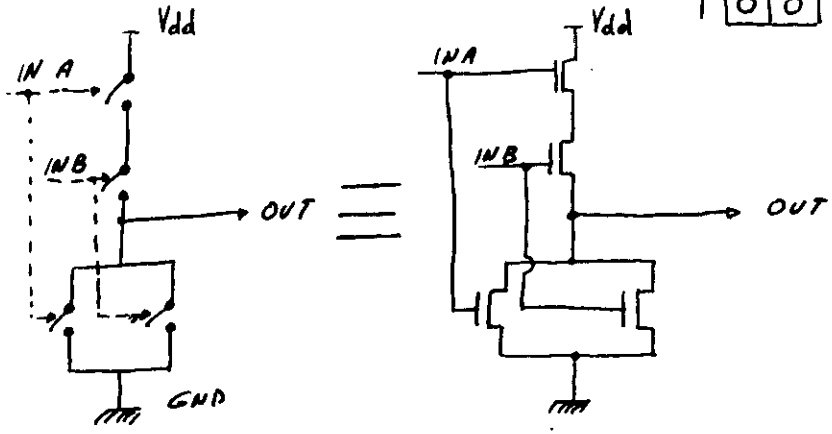
B \ A	0	1
0	1	1
1	1	0



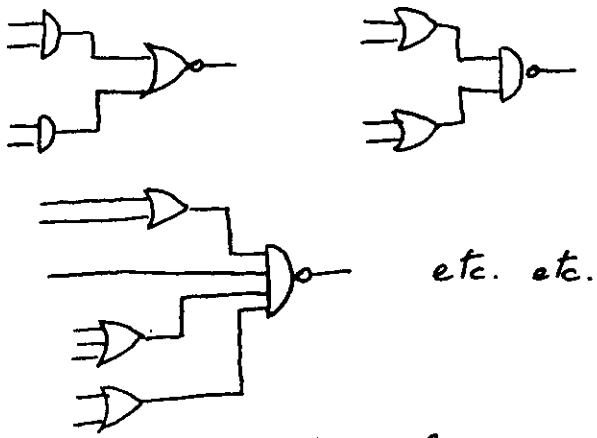
Example NOR



B \ A	0	1
0	1	0
1	0	0



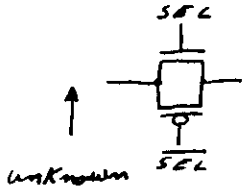
More complex functions are possible, such as:



etc. etc.

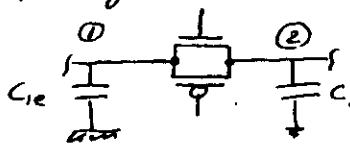
Because the speed of a gate is a function of the capacitance of the output node (n° of trs in //) and current sinking or sourcing capability (n° of trs in series), too much complexity in just one gate can be inefficient.

In all the previous examples, the MOS can be seen as a pass-transistor which passes Vdd or GND to the output node. Notice that P-type transistors always pass a 1 (Vdd) and N-type transistors always pass a 0 (GND).
It will be clear later, for now consider that a N-type transistor is a good switch for passing GND and a P-type a good switch for passing Vdd.
We can build logic gates by connecting pass-transistors (also called transfer-gates) to nodes different from Vdd and GND, which the logic value of, can be determined by previous logic.
This kind of logic is called steering logic, because data (logic values) are steered to the right nodes by means of switches.
If the logic value of a node is unknown and we want to pass that value to another node, we should use both transistors (P-type, N-type) controlled by the true value and the complement of the signal.



Notice that this device is bidirectional, information can flow in both directions.
In static logic one end of the transfer gate is always connected (maybe through a chain of transistors) to power supply.

In other kind of logic we can have situations like this:

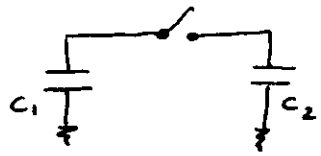


where C_{1e} and C_{2e} are the equivalent cap. of the two nodes.

In this case the final value of the voltage in the capacitor we want to transfer data to, can be different from the full Vdd or GND

$$V_{f2} = \frac{C_1 V_1 + C_2 V_2}{C_2}$$

$$V_{f1} = \frac{C_1 V_1 + C_2 V_2}{C_1}$$



$Q_{in} = C_1 V_1 + C_2 V_2$

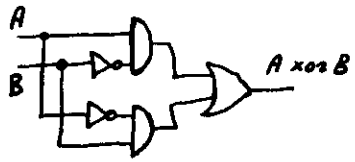
$Q_{f2} = C_2 \cdot V_{f2}$

$Q_{f1} = C_1 \cdot V_{f1}$

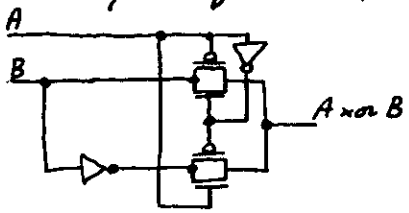
Designers must be very careful about charge sharing problems; later on, we will discuss this problem again.

Example: How can transfer-gate be used for designing an efficient XOR gate?

Standard implementation



By using transfer-gate

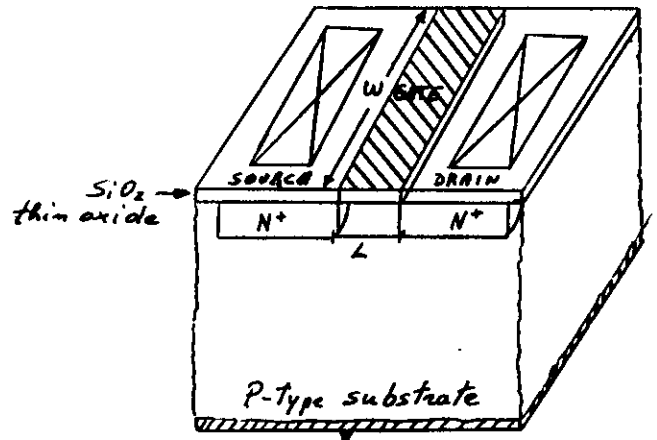


A \ B	0	1
0	0	1
1	1	0

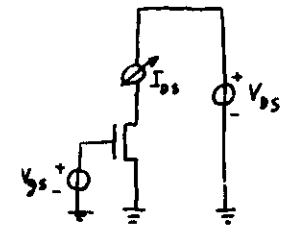
A can be thought as the selection signal. If it is 0, B is passed if it is 1, $\bar{B}$ is passed.

MOS Physics

N-type MOS silicon gate (enhancement)



W = gate width
L = gate length



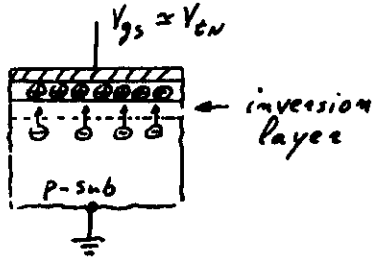
V_{TN} = threshold voltage N-type device

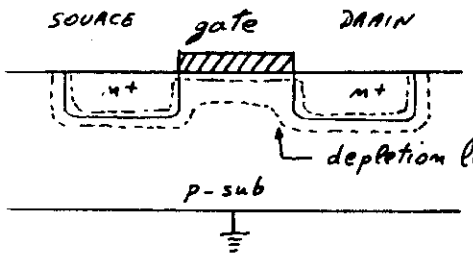
Operating conditions $V_{GS} > 0$ $V_{DS} > 0$

-1 $V_{GS} < V_{TN}$ device OFF $I_D = 0$

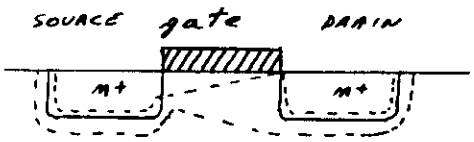
-2 $V_{GS} \simeq V_{TN}$ the inversion layer builds up

-3 $V_{GS} > V_{TN}$ majority carriers (-) increase under the gate



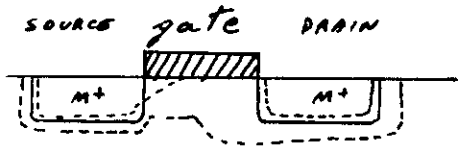


$V_{GS} \geq V_{TN}$; $V_{DS} = 0$



LINEAR REGION

$V_{ds} \leq V_{gs} - V_{TN}$



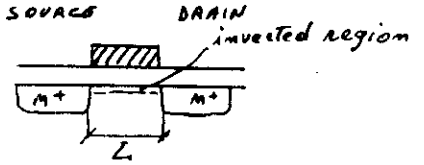
SATURATION REGION

$V_{ds} > V_{GS} - V_{TN}$

$I_{ds} > 0$ is responsible of electrons flow toward DRAIN region.
If V_{ds} increases then the voltage drop causes the channel shape to change. For $V_{ds} > V_{gs} - V_{TN}$ the channel end doesn't reach the drain any more.
The voltage along the channel is fixed at $V_{GS} - V_{TN}$

- Cut off — Only leakage current I_{DSL}
- Linear — I_{DS} increases linearly with V_{GS}
- Saturation — I_{DS} independent from V_{ds}

MOS transit time
(1st approximation analysis)



L = gate length
τ = transit time
≙ time it takes to an electron to go from SOURCE to DRAIN

τ = average time
μ = electron mobility
E = electric field

$$\tau = \frac{L}{\mu \cdot E} = \frac{L}{\mu \frac{V_{DS}}{L}} = \frac{L^2}{\mu \cdot V_{ds}}$$

V_{ds} cannot increase indefinitely, because at $V_{GS} - V_{TN}$ saturation is reached.

$$\tau_{min} \simeq \frac{L^2}{\mu (V_{GS} - V_{TN})}$$

$$I = \frac{Q}{\tau} \qquad Q = \text{charge in the depletion zone}$$

$$C_g = \varepsilon \frac{Area}{t_{ox}} = \varepsilon \cdot \frac{L \cdot W}{t_{ox}} \qquad t_{ox} = \text{oxide } (Si O_2) \text{ thickness}$$

$$Q = C_g (V_{GS} - V_{TN})$$

$$I_{DS} = \frac{Q}{\tau} = \frac{C_g (V_{gs} - V_{TN})}{\tau} = \mu \cdot \frac{\varepsilon \cdot W}{t_{ox} \cdot L} \cdot (V_{GS} - V_{TN}) V_{ds}$$

At saturation $V_{DS} = V_{GS} - V_{TN}$ the channel is:

so:

$$I_{DS} = \frac{\mu \varepsilon}{2 \cdot t_{ox}} \cdot \frac{W}{L} (V_{GS} - V_{TN})^2$$

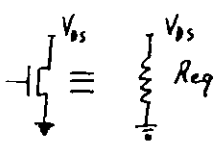
↑ It takes into account the channel shape

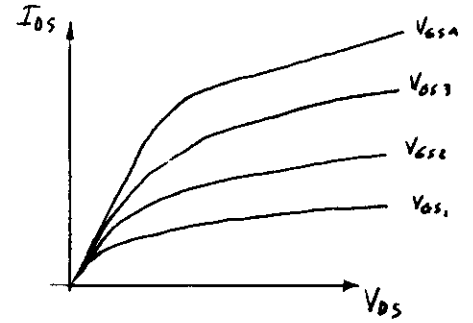
The general formula is:

$$I_{DS} = \frac{\mu \cdot \varepsilon}{2 \cdot t_{ox}} \cdot \frac{W}{L} \left[2 (V_{GS} - V_{TN}) V_{DS} - V_{DS}^2 \right]$$

valid in linear too.

LINEAR ZONE

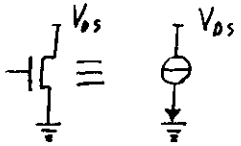


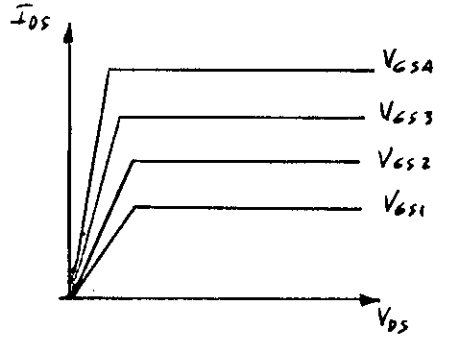


Real curve

Notice that the previous equation represents:

SATURATION





1st approx. curve

The real behaviour can be described by introducing a more complex model. SPICE is a simulation program which models MOS with great accuracy.

The basic equation can be rearranged as:

$$I_{DS_{SAT}} = \frac{\beta}{2} (V_{GS} - V_{TN})^2 \quad \text{SATURATION ZONE}$$

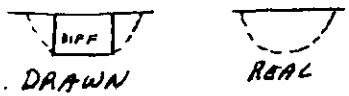
$$I_{DS_{LIN}} = \beta \left[(V_{GS} - V_{TN}) V_{DS} - \frac{V_{DS}^2}{2} \right] \quad \text{LINEAR ZONE}$$

where $\beta = \frac{\mu \cdot \varepsilon}{t_{ox}} \cdot \frac{W}{L}$

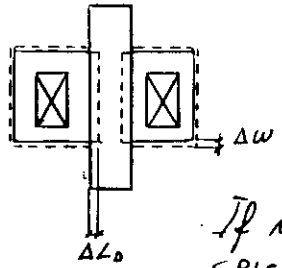
↑ is called K' ↑ geometric parameter

Usually L is fixed by the technology and designers change W in order to meet the specs for that particular device.

In real cases we should include masks' errors and lateral diffusions



DRAWN REAL



$$\left(\frac{W}{L}\right)_{eff} = \frac{W + 2\Delta W \pm \Delta M_{OSK}}{L - 2\Delta L_D \pm \Delta M_{OSK}}$$

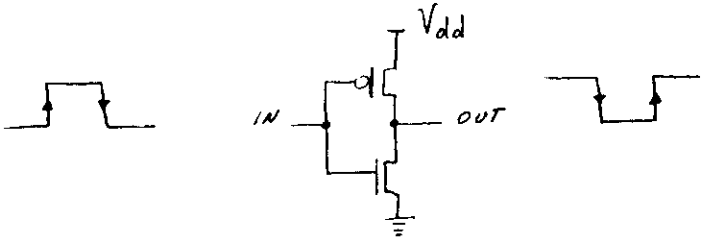
If right parameters are provided SPICE does that for you.

For p-type devices same equation, but negative voltages.

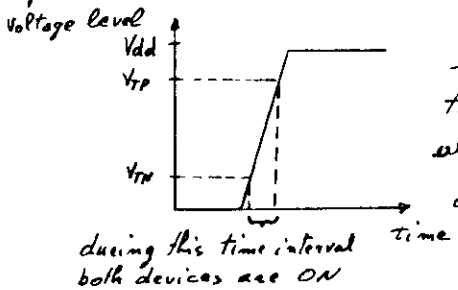
Notice that μ depends on dopant concentration and temperature. It does depend on the kind of carrier too.

p-type $\mu_p \simeq \frac{1}{2} \mu_N$ N-type in the temperature range $25\,^{\circ}C \div 125\,^{\circ}C$.

The inverter
the basic block



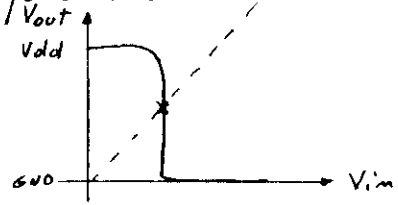
Only one device is ON when the input is in a stable state There is no direct path between Vdd and GND in both stable state (either 0 or 1). Only during the transition the direct path is active for a short time.



If the rise and fall time of internal signal. are too large, CMOS gate dissipate useless powe

An important parameter of a logic gate is its threshold voltage
$V_{in} \triangleq V_{out}$

Transfer function for an inverter:



Let us find the threshold voltage for an inverter: both devices are in saturation

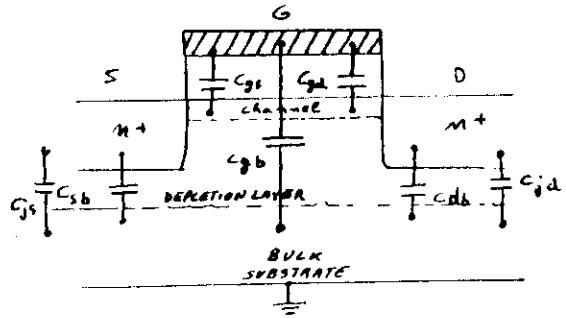
$$\begin{cases} I_{DSN} = \frac{\beta_N}{2}(V_{in} - V_{TN})^2 \\ I_{DSP} = \frac{\beta_P}{2}(V_{dd} - V_{in} - V_{TP})^2 \end{cases}$$

$$\sqrt{\frac{\beta_N}{\beta_P}} = \frac{(V_{dd} - V_{in} - V_{TP})}{V_{in} - V_{TN}}$$

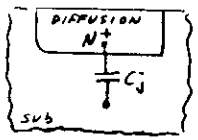
$$V_{in} = \frac{V_{dd} + V_{TP} + V_{TN}\sqrt{\frac{\beta_N}{\beta_P}}}{1 + \sqrt{\frac{\beta_N}{\beta_P}}}$$

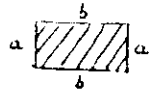
If we want $V_{in} = 0.5\,V_{dd}$ then $\beta_N = \beta_P$ and $V_{TN} = V_{TP}$
Because $\beta = \frac{\mu \cdot \varepsilon}{t_{ox}} \cdot \frac{W}{L}$ and $\mu_n \simeq 2 \cdot \mu_p$ we should design the p-device twice as large as the n-device

MOS capacitances



	OFF	LINEAR	SATURATION
C_{gb}	$\frac{\varepsilon \cdot A}{t_{ox}}$	0	0
C_{gs}	0	$\frac{1}{2}\frac{\varepsilon A}{t_{ox}}$	$\frac{2}{3}\frac{\varepsilon A}{t_{ox}}$
C_{gd}	0	$\frac{1}{2}\frac{\varepsilon A}{t_{ox}}$	0
$C_g = C_{gb} + C_{gs} + C_{gd}$	$\frac{\varepsilon A}{t_{ox}}$	$\frac{\varepsilon A}{t_{ox}}$	$\frac{2}{3}\frac{\varepsilon A}{t_{ox}}$





$$C_j = C_{ja}(a \cdot b) + C_{jp}(2a + 2b)$$

area — perimeter

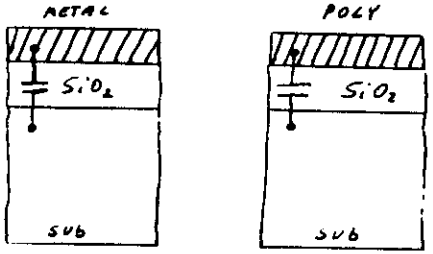
$$C_j = C_{j0}\left(1 - \frac{V_j}{\phi_B}\right)^{-m}$$

$\phi_B \simeq 0.6\ V$

$m \simeq 0.3 \div 0.5$

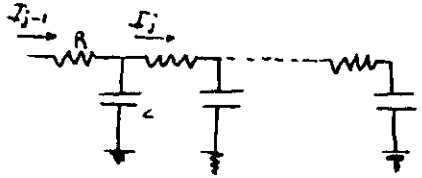
V_j = junction voltage

Routing capacitance



$$c \simeq \frac{\varepsilon \cdot A}{t_{ox}}$$

But for long POLY lines, we have to consider another effect: distributed RC effect



$$c\frac{dV_j}{dt} = (I_{j-1} - I_j) \overset{!}{=} \frac{V_{j-1} - V_j}{R} - \frac{V_j - V_{j+1}}{R}$$

$$rc\frac{dV}{dt} = \frac{d^2V}{dx^2}$$

x = distance from input
r = resistance per unit length
c = capacitance per unit length

The solution for an input step $\sqcap$ is approximated by:

$$t_n = \frac{RC\,n(n+1)}{2}$$

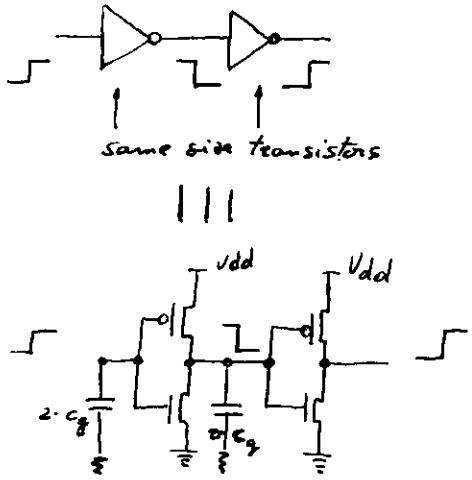
n = number of section

if n is large $t_p = \frac{rc \cdot l^2}{2}$

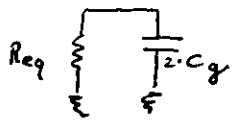
r = resistance per unit length
c = capacitance per unit length
l = length of the wire

Switching characteristics

As a 1st approximation the transit-time τ_c is proportional to the time it takes to an inverter of a certain size, to charge or discharge the capacitance associated with the gate of the same size inverter acting as a load to the previous one.



Equivalent ckt:



P-type and N-type same C_g

$2 \cdot C_g$ will discharge through R_{eq} with a time constant $2 \cdot R_{eq} \cdot C_g$

$$R_{eq} = \frac{V_{ds}}{I_{ds}} \qquad I_{ds} = \mu \cdot \frac{C_g (V_{gs} - V_{TN}) V_{ds}}{L^2}$$

$$R_{eq} = \frac{L^2}{\mu \cdot C_g (V_{gs} - V_{TN})} \qquad 2 \cdot R_{eq} \cdot C_g = 2 \cdot \frac{L^2}{\mu (V_{gs} - V_{TN})} = 2 \cdot \tau_c$$

Notice that in a chain of inverters (logic gates as well) if the size of all transistors increases, the delay of each stage remains constant.

It's a second consequence of the 1st approximation switching characteristics, if now a stage must drive a stage which has <u>twice the capacitance</u>, then it will take <u>twice the time</u>.

If you have to charge (or discharge) a big capacitor C_L the time it will take to an inverter to accomplish it is:

$$\tau_D \simeq 2 \cdot \tau_e \cdot \frac{C_L}{C_g}$$

for large C_L, it can be very large. The rise and fall time is affected too and, as we saw before, power consumption can increase drastically!

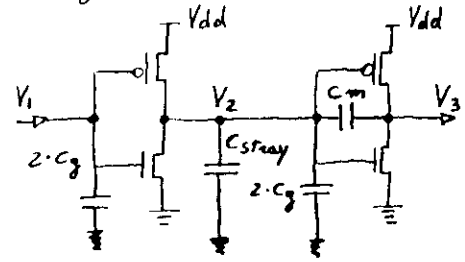
You always drive big capacitances through a chain of inverters appropriately sized.
If the transistors size ratio is equal to e (2.718....) then the total delay of the chain is:

$$\tau_D \simeq 2 \cdot \tau_e \cdot e \cdot \ln \frac{C_L}{C_g}$$

and $\ln \frac{C_L}{C_g} = n$ = number of stages

Gate capacitance (C_g) is also called active capacitance, because its charge is always converted into a useful current.

There are passive capacitances, where the current is always wasted (cannot be used to charge/discharged other nodes). Those are called stray capacitances and are due to wiring and diffusion areas.



	START	FINAL
	$V_1 = 0$	$V_1 = V_{dd}$
	$V_2 = V_{DD}$	$V_2 = 0$
	$V_3 = 0$	$V_3 = V_{dd}$

$Q_{wire} = C_{st} \cdot V_{dd}$
$Q_g = 2 \cdot C_g \cdot V_{dd}$ ⟹ $Q_{wire} = 0$, $Q_g = 0$
$Q_m = C_m \cdot V_{dd}$ $Q_m = -C_m V_{dd}$

$$C_{eq} = \frac{dQ}{dV} = \frac{(-2C_m V_{dd} - C_{st} - 2 \cdot C_g) V_{dd}}{-V_{dd}}$$

C_m = Müller capacitance

$$C_{eq} = 2 \cdot C_m + C_{st} + 2 \cdot C_g$$

↑ notice the factor 2

Standard values
$C_g \simeq .20 \div .25$ pF/mil²
$C_{st} \simeq .03$ pF/mil²
$C_m \simeq .20 \div .25$ pF/mil²

Even in compact structures the term $2 \cdot C_m + C_{st}$ is comparable with $2 \cdot C_g$

2nd-order effects

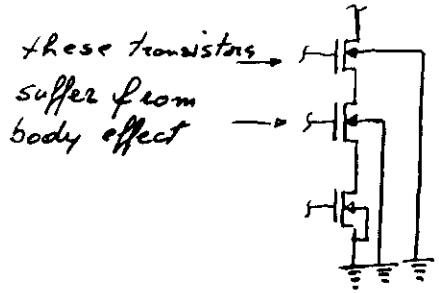
- Body effect If V_{SB} increases V_t increases too:

$$V_t \simeq V_{t(0)} \pm \gamma \left(V_{SB}\right)^{1/2}$$

$V_{t(0)} = V_t$ @ $V_{SB}=0$

$0.4 \leq \gamma \leq 1.2$

Pratical consequences



The bulk of N-transistors is always connected to GND and the bulk of P-transistors to V_{dd}.

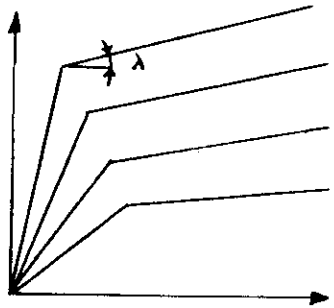
Depending on the process parameters, the body effect can be so bad that structures with more than 4 ÷ 6 transistors in series are not allowed.

- Finite output resistance

This effect is modelled by introducing the λ parameter

$$I_{DS} = \frac{\beta}{2}\left[(V_{GS}-V_{TN})^2(1+\lambda V_{DS})\right]$$

$0.02\,V^{-1} \leq \lambda \leq 0.04\,V^{-1}$



- Temperature

V_t varies $-4\ mV/^{\circ}C$ (highly doped)
$-2\ mV/^{\circ}C$ (lightly doped)

μ drops down $\mu(T) = \mu(300\,^{\circ}K) + \left(\frac{T}{300\,^{\circ}K}\right)^{-2.6}$

the effect is such that

$\beta = \frac{\mu \cdot \varepsilon}{t_{ox}}$ decreases so the current in the MOS decreases

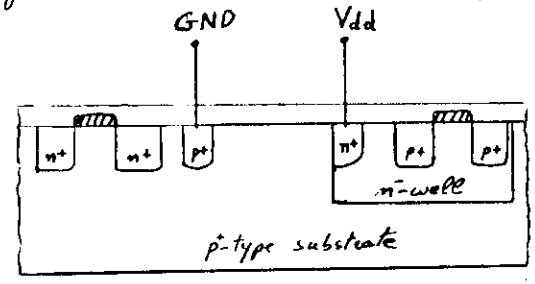
Remember that parameters for P and N transistors can vary independently.
For each type there are 3 sets of parameters which correspond to special cases:

- 1 Typical condition
- 2 Worstcase speed (slow device)
- 3 Worstcase power (fast device)

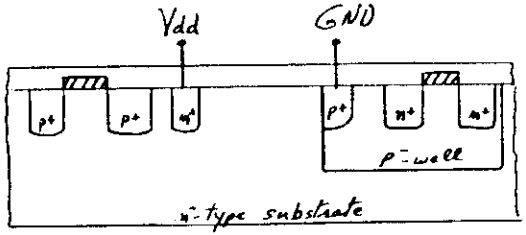
As gate length is getting shorter and shorter, more complex phenomena happen. They can only be modelled by simulators such as SPICE.
To simulate simple circuits, SPICE requires a time which grows exponentially with the number of nodes in the net. For VLSI simulations other kind of simulators are required (SWITCH LEVEL SIMULATORS, such as RSIM). In this case MOS is modelled by simple resistors.

CMOS and latch-up

In order to ensure isolation among N-type devices, we have already seen that the substrate (P-type) has to be connected to GND (all P-N junctions are reversed biased)
The same thing is true for P-type devices, whose substrate has to be connected to Vdd.
The process we have described so far implicitly considers N-type devices built into P-type substrate and P-type devices built into a N^- substrate, called N-well.
It is also possible to start with an N-type substrate, where P-type devices are built, and to diffuse P^- type areas, where N-type devices will be placed.
The former process is called N-well process, the latter P-well.



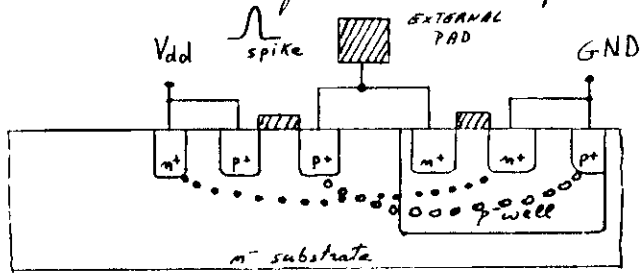
N-well process



P-well process

There are advantages and disadvantages with both processes, but they all suffer from latch-up

Latch-up is the name of the effect that occurs when an accidental injection of electrons or holes in the substrate turns on parasitic devices (bipolar transistors), always present in a CMOS process, and a short between Vdd and GND occurs. The short persists even if the cause is removed.



Latch-up can happen anywhere, but it is more likely to happen in the IN/OUT circuitry (external noise).

If, for instance, an external voltage spike causes an holes injection they get attracted and travel up to GND connection.
Because the high resistivity of the P-well, the voltage drop between n^+ pwell diffusions can turn on the parasitic NPN transistor, inside P-well region, and electrons are injected into the substrate and get attracted by Vdd.
Because the high resistivity in the n-substrate, the PNP transistor is also turned on and its emitter injects even more holes into the n-substrate. At this point there is a positive feedback and a path between Vdd and GND exists, even if the initial external noise is removed.

equivalent circuit

Scaling of MOS dimensions

Principles First order scaling is obtained by applying a dimensionless factor α to:
- all dimensions, including those vertical to the surface
- device voltages
- concentration densities

In this way the basic operational characteristic are preserved.
From what we found before:

$$\tau_t = \frac{L^2}{\mu \cdot V_{ds}} \qquad C_g = \varepsilon \cdot \frac{W \cdot L}{t_{ox}}$$

$$I_{SAT} = \frac{\beta}{2}(V_{gs} - V_{TH})^2 \qquad I_{LIN} = \beta\left[(V_{GS} - V_{TH})V_{ds} - \frac{V_{ds}^2}{2}\right]$$

Parameter	Scaling factor
gate delay τ_t	$1/\alpha$
capacitance	$1/\alpha$
power dissipation	$1/\alpha^2$
current	$1/\alpha$
current density	α
power-speed product	$1/\alpha^3$

Interconnect layer scaling

$$R_\ell = \frac{\rho}{t_{ox}}\left[\frac{L}{W}\right] \quad \text{line resistance} \qquad R'_\ell = \frac{\rho}{t_{ox}/\alpha}\left[\frac{L/\alpha}{W/\alpha}\right] = \alpha R$$

$V'_d = I/\alpha \cdot \alpha R = \underline{I \cdot R \text{ constant}}$ $\quad V'_d$ = new voltage drop along an interconnect line

Delays introduced by interconnections became more important.

Yield (no packaging included)

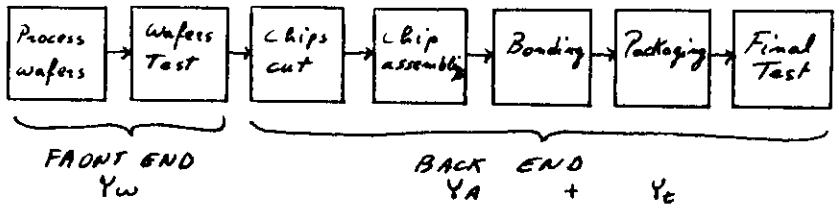
Yield Y is defined as:

$$Y = \frac{\text{n° of Good chips}}{\text{total n° of chips}}\ 100\%$$

- Seed's Model
$$Y = e^{-\sqrt{A \cdot D}}$$
D = defects density
A = chip area
good for large chips

- Murphy's Model
$$Y = \left(\frac{1 - e^{-A \cdot D}}{A \cdot D}\right)^2$$
good for small chips

I.C. fabrication flow.



We have considered Y_W only.

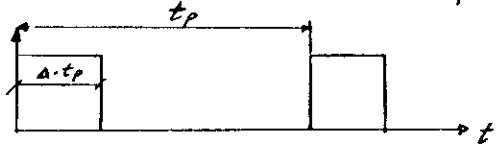
Power dissipation

$$P_t = P_D + P_S$$

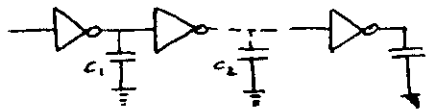
P_D = dynamic power
P_S = static power

During a transition both transistors (P and N) are ON at the same time for a very short period. (P_S)
Other power is spent charging and discharging the capacitances associated with nodes. (P_D)
In a well designed CMOS chip, P_S is very small compared with P_D.
Special circuits must be designed in order to ensure small P_S dissipation in large buffers.

Modeling P_D The assumption is that the rise and fall time of the input step is much shorter than the repetition period.



Δ = duty cycle
t_p = repetition period



The energy provided by the power supply is $V \cdot Q = C_1 V^2$.
For the second inverter it is $C_2 V^2$ (this is for the dynamic power)
Now if a circuit has n nodes, roughly $n/2$ of them are in one state and $n/2$ in the other

If I consider pairs of inverters then I can say:

$$\Delta E_{tot} = (C_1 + C_2) V^2 + E_s$$ (energy due to static current consumption)

$$\text{Power} = \frac{\Delta E_{tot}}{T} = \frac{(C_1 + C_2) V^2}{T} = (C_1 + C_2) V^2 \cdot f$$ dynamic

$$P_{d_{tot}} = V^2 \cdot f \cdot C_t$$ (total capacitance of the node)

Temperature and packaging.

The relationship which expresses the temperature of the junction as a function of the power dissipated is:

$$T_j = T_{amb} + \theta_{jA} \cdot P$$

T_j = junction temp
P = power dissipated
θ_{jA} = thermal resistance

The maximum power that can be dissipated can be computed assuming a max T_j of about 175 °C for Si:

$$P_{max} = \frac{T_j - T_{amb}}{\theta_{ja}}$$

T_{amb} = ambient temperature

θ_{ja} is the sum of many terms which depend on how the die is attached to the leadframe, kind of package, kind of cooling etc. etc.

Notice that in CMOS circuits, the power is dynamic (only a little fraction is due to static dissipation.)

I/O structures

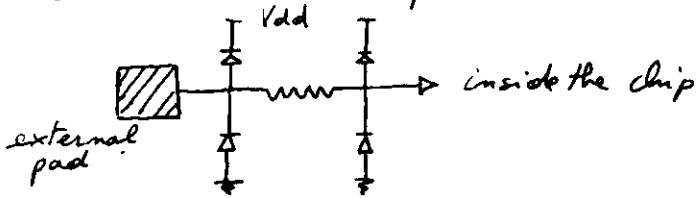
The gate connection of an MOS transistor has a very high input resistance ($10^{12} \div 10^{13}\,\Omega$). The voltage at which the gate breaks down is about 40–100 volts.

$$V = \frac{I \cdot \Delta t}{C_g}$$

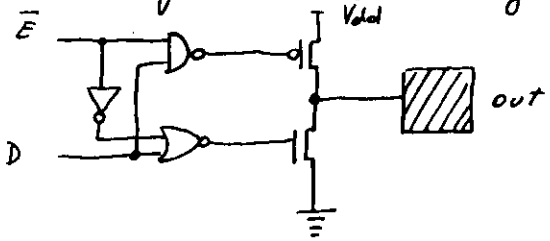
C_g = gate capacitance
Δt = time to charge the gate
I = charging current
V = gate voltage

If $I = 10\,\mu A$; $C_g = .03\,pF$; $\Delta t = 1\,\mu sec$ $\quad V \simeq 330$ volts

Input gates must be protected by a combination of resistance and clamp diodes.



Output structures must be protected from latch-up effect and clamp diodes are also present.
Tri-state pads are also very common.



If $\bar{E} = 0$ then the pad is in a high impedance state

Power distribution

Even if the power (static) dissipated by CMOS logic is very low, dynamic power can be huge and in order to avoid big voltage drops a careful power distribution scheme must be designed.

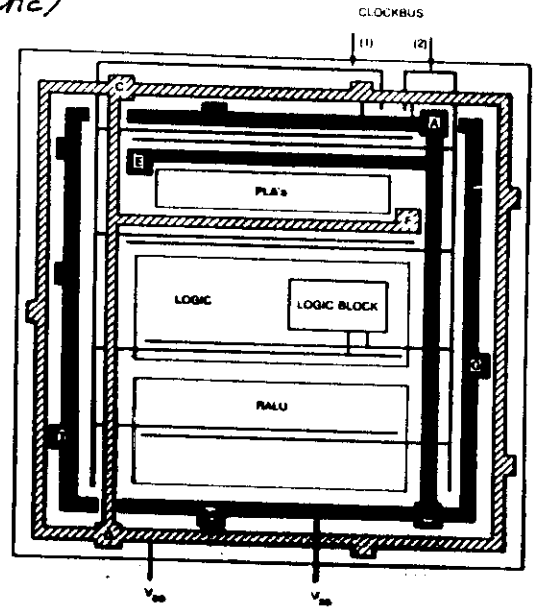
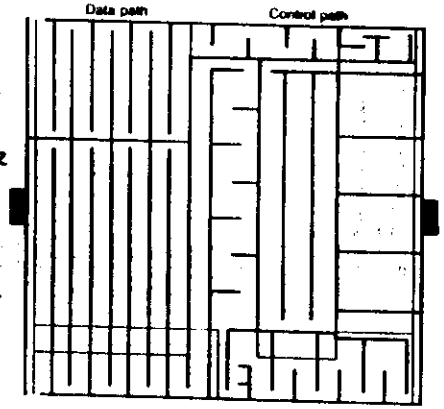


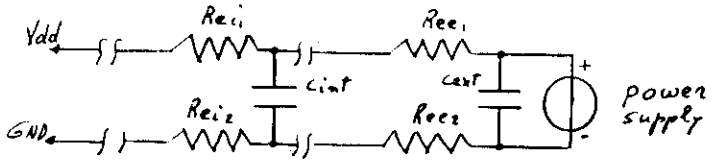
FIGURE 9.88. Clock and power supply distribution schemes that eliminate measurable skew

FIGURE 5.18 Power and ground routing of a microprocessor. Power and ground are routed as interdigitated trees.



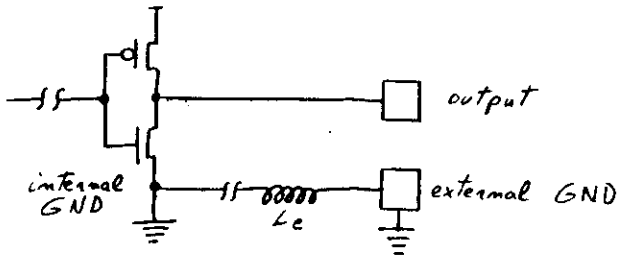
In a full-custom design, the power distribution scheme is not predefined, and metal lines can be sized according to their location in the chip

When several thousand gates switch at the same time, the spike of current that the power supply lines must provide can be so large that there is a large voltage drop caused either by the internal/external resistance of the power lines, or by the internal resistance of the power supply itself.
Careful design of power distribution lines must be always the first designer's concern when a VLSI chip is to be developed.
Internal/external capacitors can be provided, in order to minimize the load to the power supply.



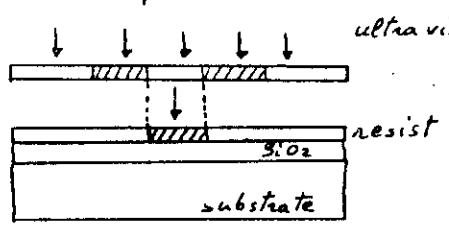
R_{ee} = external equivalent resistance C_{int} = optional internal capacitan
R_{ei} = internal equivalent resistance C_{ext} = optional external capacitanc

The large current during transients in the output buffers and the inductance associated with the output pins of the package, can cause "Ground bounce"



CMOS process and layout rules

The process consists on several different layers separated by an insulator (oxy) Si_2O
The geometric entities are drawn on masks and by a photolithographic procedure are transfered into silicon.
The process starts with a slice of Si (p-type, for instance) which will be the substrate of the chip.
A layer of oxide (Si_2O) grows on the surface of the Si slice after some hours of high temperature (1000 °c) baking in an oxygen environment.
The slice is then covered with photoresist.
If photoresist is hit by light its structure changes and can be removed by some chemical substance.



The photoresist hit by light is removed and the surface of the slide is covered with an acid (HFl) which attacks Si_2O. A hole in the oxy layer is created in this way.
Then the exposed surface of the substrate can be treated accordingly.
The procedure of transfering a drawing (Layout) into Si is called PATTERNING

Each patterning consists on the following steps:

- oxidation
- coating resist
- mask
- develop resist
- etching oxide
- resist removal

Currently there are many CMOS processes available, such as:

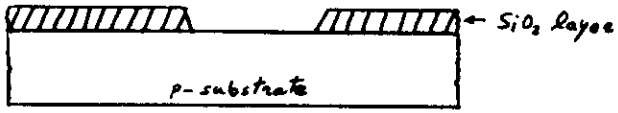
- P-well
- N-well
- Twin-tub

etc. etc.

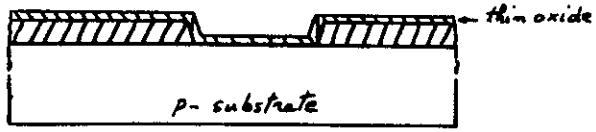
In the following examples an N-well process will be considered for explaining some basic steps common to all CMOS processes.

Fabrication steps for a silicon gate nMOS transistor

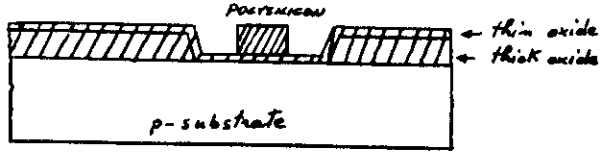
-1 patterning SiO_2



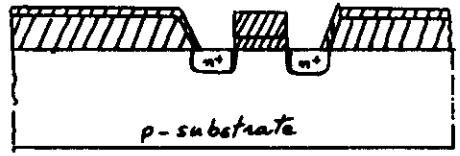
-2 gate oxidation



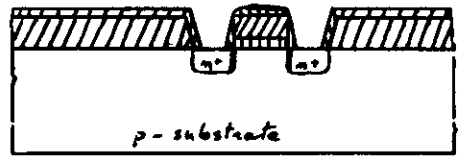
-3 patterning polysilicon



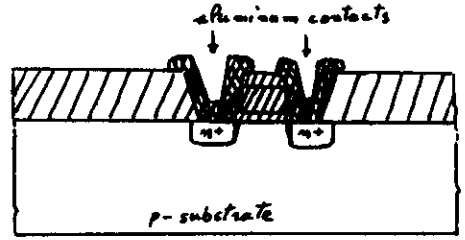
-4 Diffusion or implant



-5 contact cuts



-6 patterning of aluminum layer

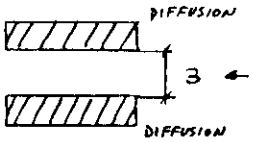


Similar fabrication steps are used for building a P-type device into an island of N-well

The "Lambda" rules

Perhaps the most powerful attribute of modern CMOS wafer-fabrication processes is that they are pattern-independent. There is a clean separation between the processing done during fabrication and the design work that creates the patterns to be implemented.
The designer is told about geometric constraints which must be applied to the patterns, in order not to violate the physics required for the proper operation.
These geometric constraints translate into certain widths, separations, extensions and overlaps of geometrical objects patterned in the various levels of the process. (minimum distance)
All dimensions are given in terms of an elementary distance unit, called length-unit λ (lambda).

Ex:



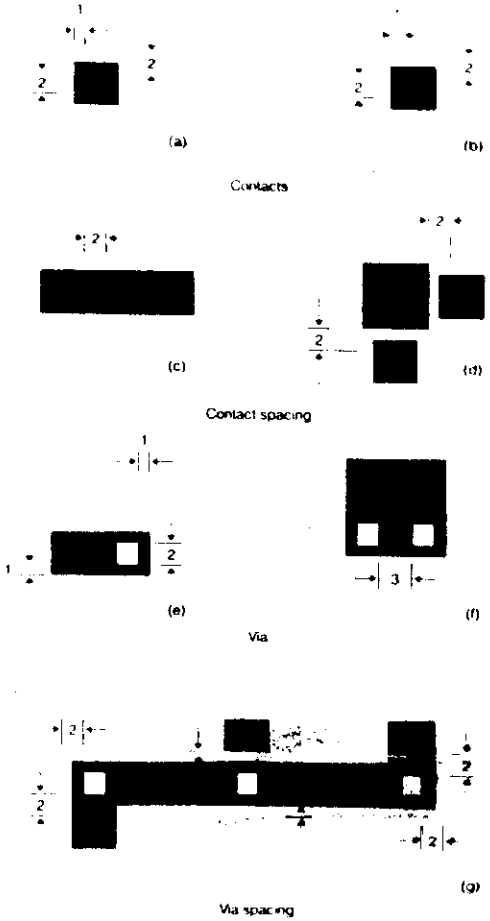
← this means that the minimum gap can be $3 \cdot \lambda$

Notice that this simplify not only the process itself, but the design too. Moreover, it is easier to write programs which understand layout.
By changing λ you change technology (scalable CMOS)

(ST)

Lambda rules

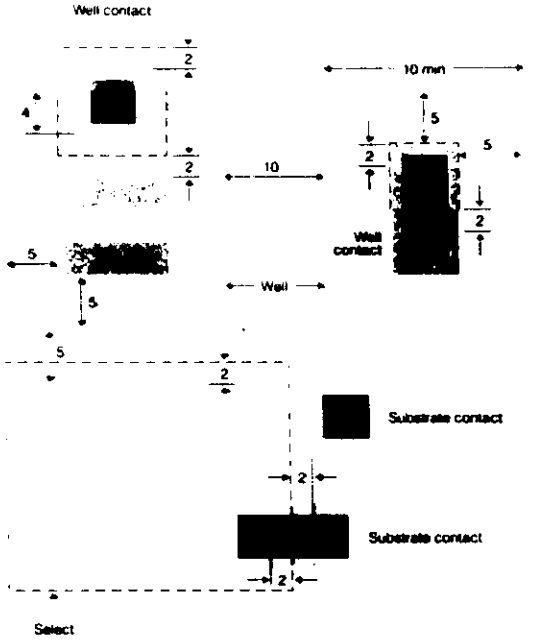
Some of the lambda rules for 3μ MOSIS CMOS process.



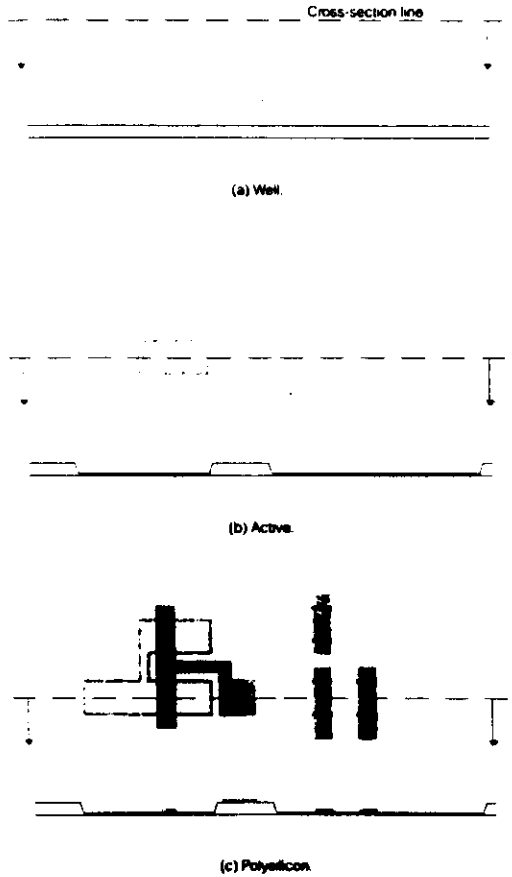
(ST)

Lambda rules

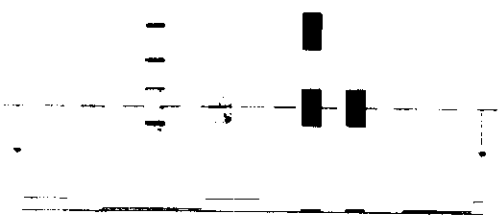
MOSIS 3μ lambda rules for well and substrate contacts.



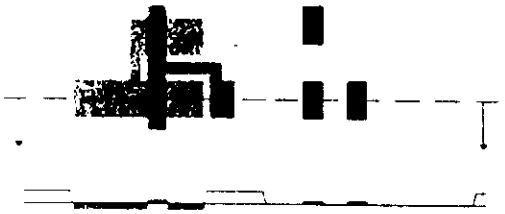
An example with a CMOS process



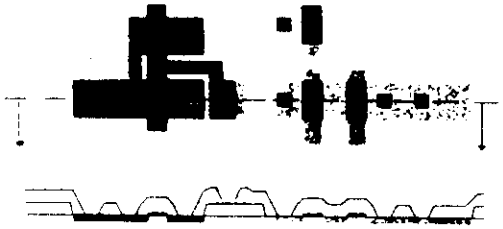
CMOS process (example)



(d) n-Diffusion.

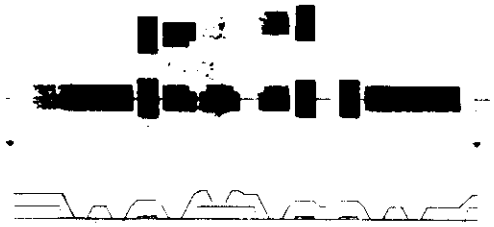


(e) p-Diffusion.



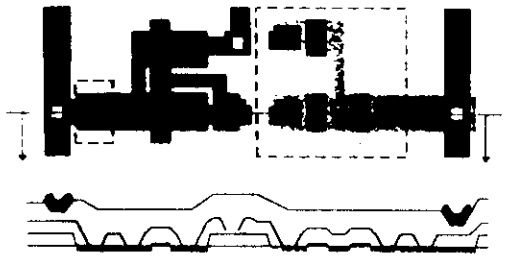
(f) Contacts.

CMOS process (example)



(g) Metal

(h) Via.



(i) Metal 2.

CMOS static logic design

Assumptions :

-1 We will consider positive logic only (Vdd = 1 = true value; GND = 0 = false value.

-2 All gates are inverting

-3 Only static logic will be considered for now.

We already know how to build NOR, NAND and compound gates such as AND-NOR and OR-NAND, by using the analogy:

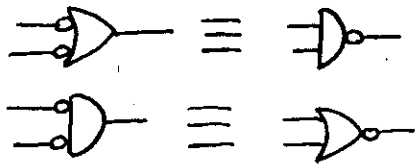
If we start from the truth table and we want a NAND implementation it is convenient to consider minterms, if we are looking for a NOR implementation then maxterms will be considered.

De Morgan laws can be applied afterward:

-1 $\overline{A \cdot B} = \overline{A} + \overline{B}$

-2 $\overline{A + B} = \overline{A} \cdot \overline{B}$

Circuit interpretation

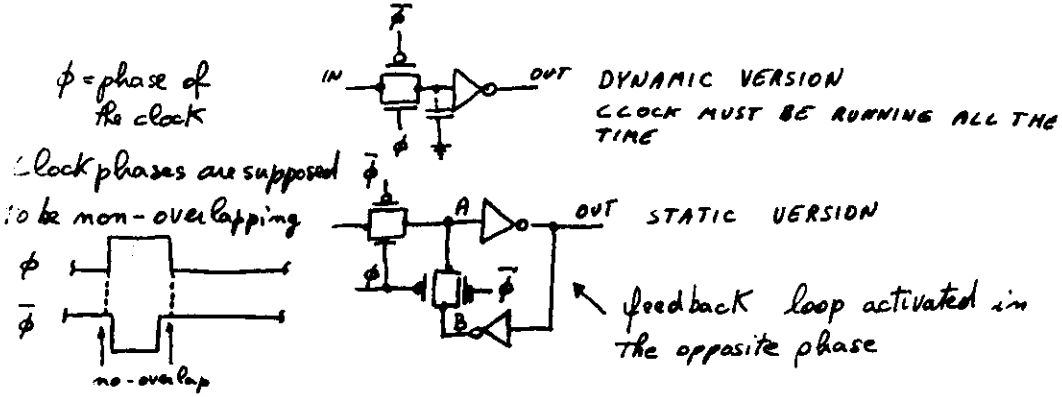


For the most common functions there are more efficient implementations using transfer-gates etc.

This process of optimizing the logic required for implementing a function is fine for random logic. For more complex logic a PLA implementation can be better.
If the set of logic equations is too complex, programs for logic minimization and synthesis are required (ESPRESSO etc.

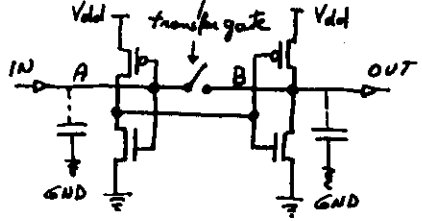
Memory structures

The basic block is a latch:



This is called 2-phase latch. There are many possible alternatives to the clock distribution. Clock strategies will be examined later on.

Let us consider the feedback-loop when activated:



Possible cases:

$V_A > V_B$

$V_A < V_B$

$V_A = V_B$ (intermediate value)

If $V_A < V_B$ capacitor in A will charge at V_{dd} (positive feedback)

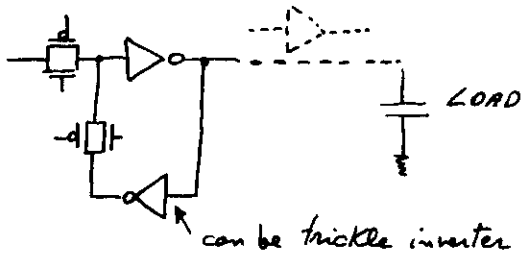
If $V_A > V_B$ before the transfer-gate $\bar{\phi}$ gets switched-ON, capacitor in A will be discharged to GND

If $V_A = V_B$ (intermediate state) very slow evolution, unpredictable; noise driven.

Very important: data must be stable before and after the clock active edge.

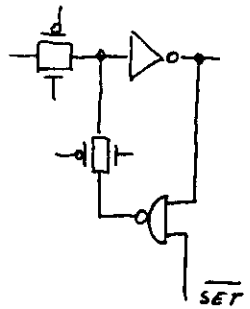
Signal in B must be stable (final value) before switching the feedback transfer gate ON.
Transistor sizes in the 2 inverters must be different to avoid the problem.

A big LOAD can cause problems.
A buffer must be included in the output path.

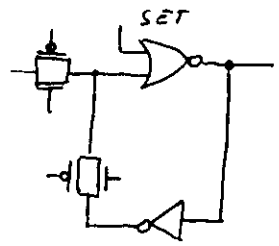


At power-on the state of memory structures like this is unpredictable. A preset (reset) is needed.

1)

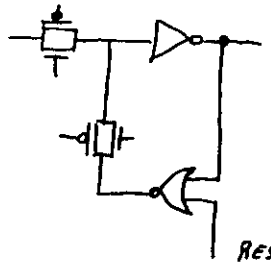


2)

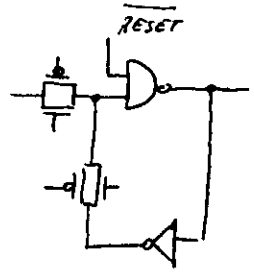


Possible ckts with SET line

1)

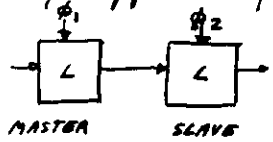


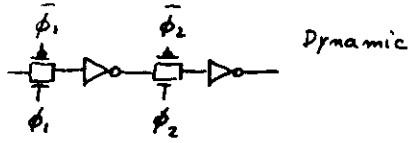
2)

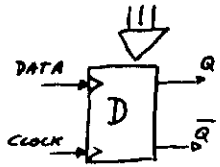


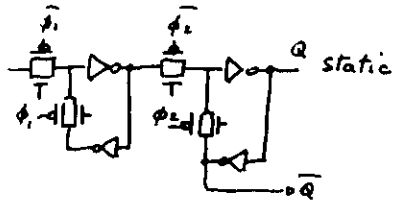
Possible ckts with RESET line

A D-type flip-flop can be formed by the series of 2 latche: controlled by opposite phases









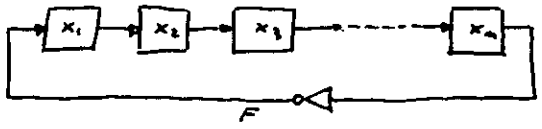
D-type flip-flop are the most used flip-flops in I.C.

Counters

Any kind of counter can be implemented in CMOS I.C. design, anyhow the most common ones are:

- -1 Johnson counters
- -2 Nixon counters
- -3 Polynomial counters
- -4 Binary counters

Johnson counter. General scheme



x_i = state

x_i ≡ D-type F.F.

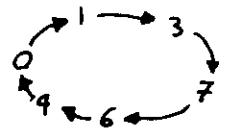
Example with 3 stages

initial state	x_3	x_2	x_1	F	coding
	0	0	0	1	0
	0	0	1	1	1
	0	1	1	1	3
	1	1	1	0	7
	1	1	0	0	6
	1	0	0	0	4
	0	0	0	1	0

F = feedback state

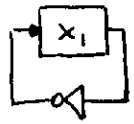
x_i = states

After 6 transitions, I reach the initial state (divider by 6)



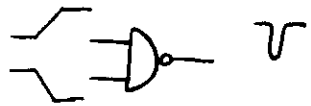
If I start with a different state (0 1 0) I reach it again after 2 transitions

If n = n. stages, I can build a 2·n divisor



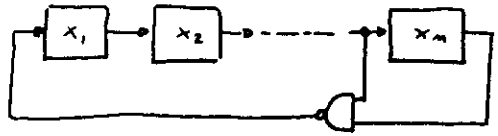
A common block a 2 divider also called TOGGLE

Notice that in Johnson counters only 1 output (state) changes at a time. This allows a very clean coding (no SPIKE problem)



If inputs change at the same time then a spike can be generated

NIXON COUNTER

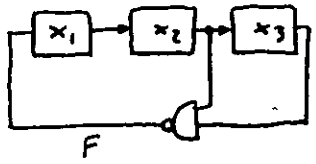


Feedback through a NAND on the last 2 outputs

DIVIDER by $2N-1$ $N \geq 3$

Be careful, Nixon counters can have more than 1 loop, but it always enters the larger loop after a certain number of transitions.

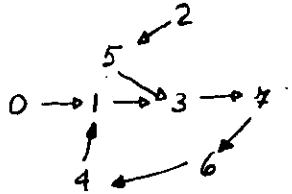
Example



a 3 stages NIXON counter

	X_3	X_2	X_1	F
0	0	0	0	1
1	0	0	1	1
3	0	1	1	1
7	1	1	1	0
6	1	1	0	0
4	1	0	0	1
1	0	0	1	1

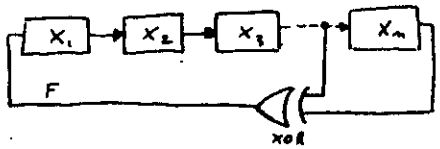
	X_3	X_2	X_1	F
2	0	1	0	1
5	1	0	1	1
3	0	1	1	



It enters the loop, after 2 transections (maximum)

These counters are used for division with small number. For larger divisions, polynomial counters are used.

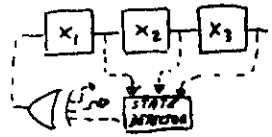
POLYNOMIAL COUNTERS

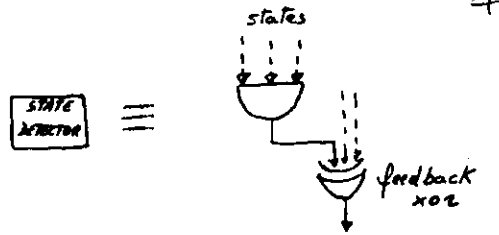


Length $\leq 2^n - 1$

Initial state all "0" must be avoided.
If the period is less than $2^n - 1$, then a special technique must be used to force the counter into the right loop.

Ex: Length of 5

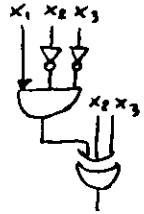




X_3	X_2	X_1	F
0	0	0	0
0	0	1	0
0	1	0	1
1	0	1	1
0	1	1	1
1	1	1	0
1	1	0	0
1	0	0	1
0	0	1	0

Among all sequences of period 5, I look for a pair where only the MSB differs (in our case 101 and 001)

I detect the last one 001 and I force a 1 into the feedback xor



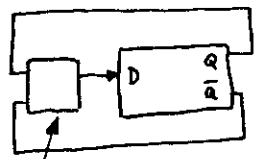
The theory on polynomial counters can be found in many of the most popular books about counters and logic design.

Bynary counters (synchronous)

3-stage example

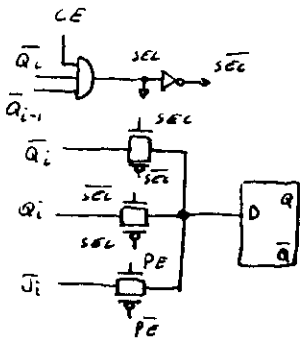
Q_3	Q_2	Q_1	
0	0	0	UP
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
0	0	0	

1st stage always toggles
2nd stage only when $Q_1 = 1$
3rd stage only when $Q_1 = 1$ and $Q_2 = 1$



enables toggle or recirculates data

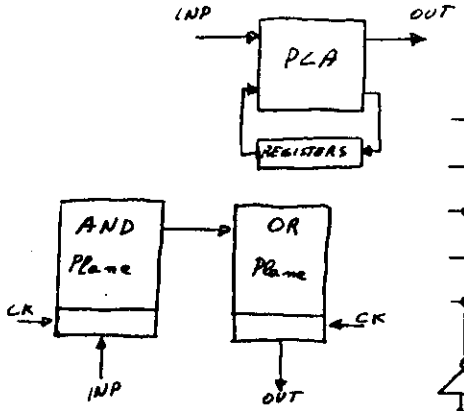
Q_3	Q_2	Q_1	
1	0	1	DOWN
1	0	0	
0	1	1	
0	1	0	
0	0	1	
0	0	0	
1	1	1	
1	1	0	
↑ $Q_1 \cdot Q_2 > 0$	↑ $Q_i = 0$	↑ always	



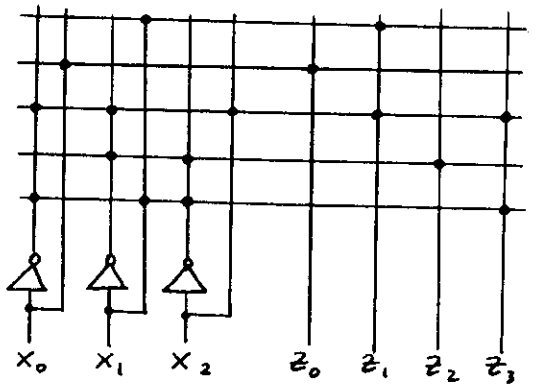
CE = counter enable
PE = preset
J_i = jammed value

PLA programmable logic arrays

A typical PLA uses an AND-OR structure that takes inputs and performs some combinational function or by feeding back outputs to the inputs; a finite state machine



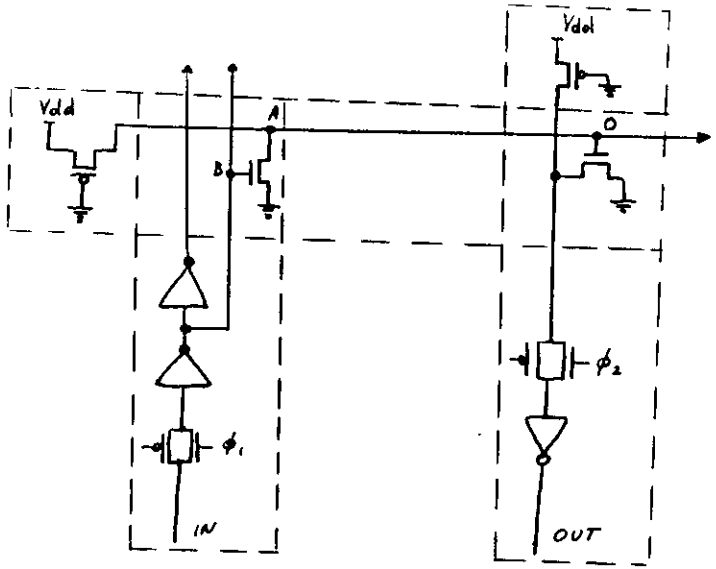
A finite state machine



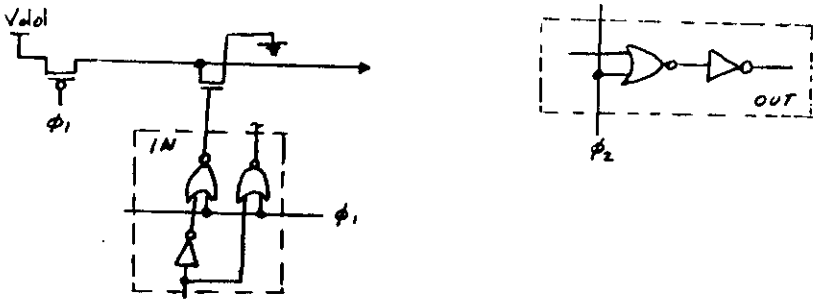
$$Z_2 = \overline{X_1} \cdot \overline{X_2}$$

$$Z_3 = \overline{X_0} \cdot \overline{X_1} \cdot \overline{X_2} + \overline{X_0} \cdot X_1 \cdot \overline{X_2}$$

Ex: Pseudo N-MOS implementation (PLA)



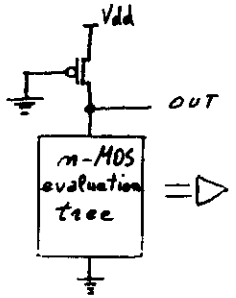
Ex: 2-phase dynamic PLA

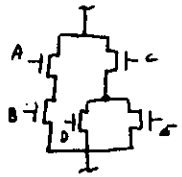


... and many other structures with 2, 4 phases

Alternative structures (CMOS)

Pseudo-nMOS logic



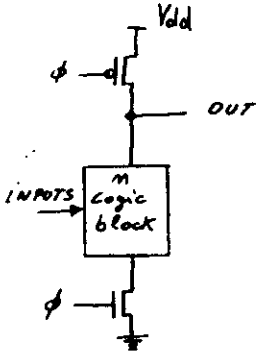


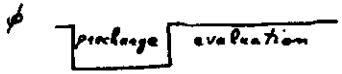
$$\overline{(A \cdot B) + (C \cdot (D + E))}$$

Advantages :
- area is smaller
- speed is higher (no p-transistors in evaluation tree in one transition

Disadvantages:
- power consumption (static)
- size ratio between n-type transistors and the load must be carefully chosen.

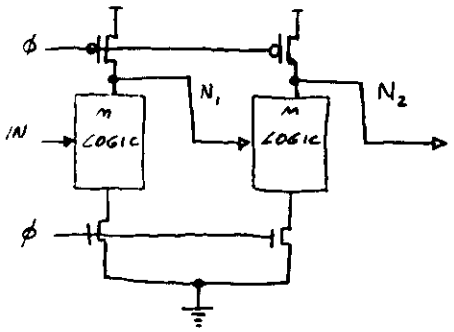
Dynamic CMOS logic

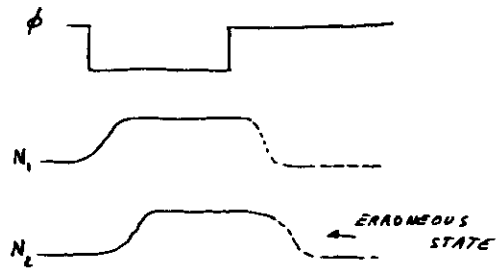




When the clock is LOW, the output node is precharged at Vdd.
When the clock is HIGH, the output node is conditionally discharged

Gates like the ones we just saw, cannot be cascated. More complex structures must be used.



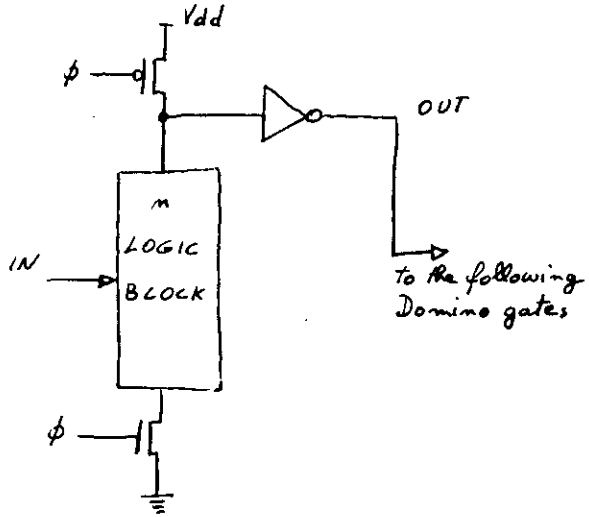


During the evaluation phase node N_1 is conditionally discharged. However some delay will be incurred and the precharged node can discharge the output node of the following gate before the first gate is correctly evaluated.
Improvements to this scheme have been developed using 4 phase logic with the addition of a sample and hold phase.

Domino Logic

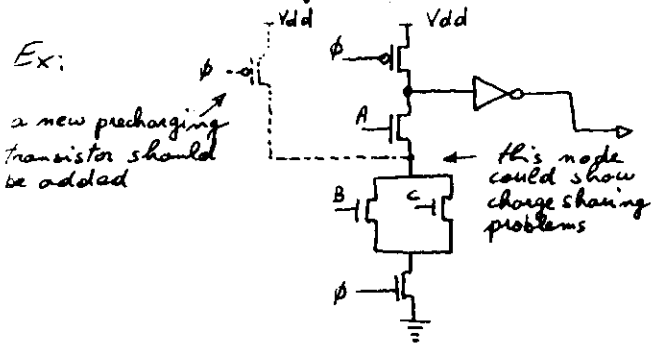
This allows a single clock to precharge and evaluate a cascated set of dynamic blocks.
By including a static inverter in the output stage of dynamic gates. In this way each gate is allowed to make at most one transition 0 → 1 and the problem we previously saw, does not happen any more.

In Domino logic each stage evaluates and causes the next stage to evaluate, in the same manner that a stack of dominos fall. This is the reason of the name.

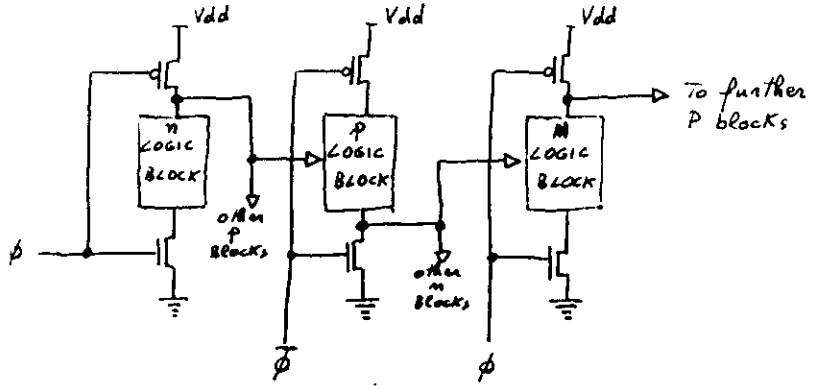


Notice that only non-inverting structures are possible and, depending on the kind of function which is implemented, charge sharing problems can occur.

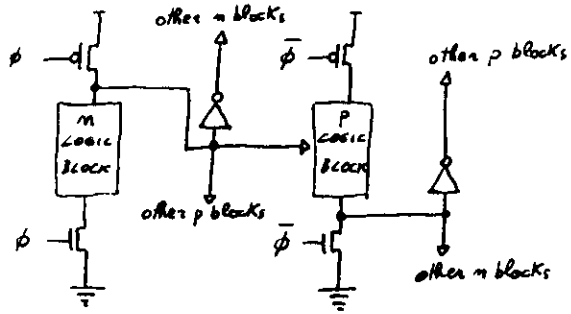
Ex:



Nora (modified Domino logic)



or



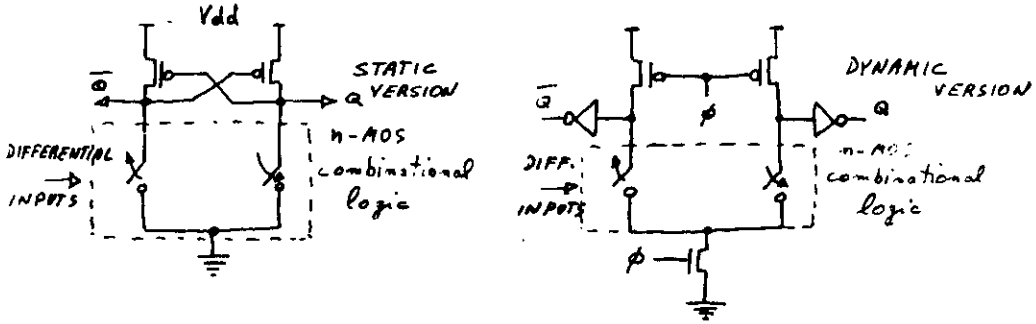
Alternating n blocks and p blocks controlled by complemented phases, more flexible structures can be build.

Problems: -1 P blocks are usually slow

-2 Noise margins are too small (< 1 V) instead of (≈ 2.5 V) as the standard Domino logic.

Very sensitive to noise.

Cascade voltage switch logic (CVSL)



Problems:
- -1 Differential inputs are needed
- -2 The number of wires is doubled
- -3 2 evaluation trees are needed

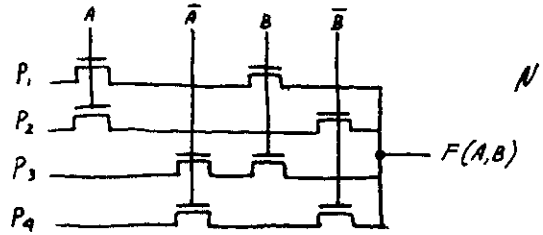
Advantages:
- -1 It is fast
- -2 Inverting and non-inverting signals are available.

Pass transistor logic

It is very popular in N-MOS circuits. In C-MOS it is also popular, but because N-type and P-type transistor are used, it becomes slower (more capacitance) and requires more area.
If the logic function to be implemented is too complex

and signals have to go through long series of pass transistors, then buffering stages must be included in order to avoid large delays due to the distributed R·C characteristic of the line. (delay $\simeq \frac{r \cdot c}{2} \cdot n^2$ n = number of pass ths,

Ex:



N-MOS version of the "function unit".

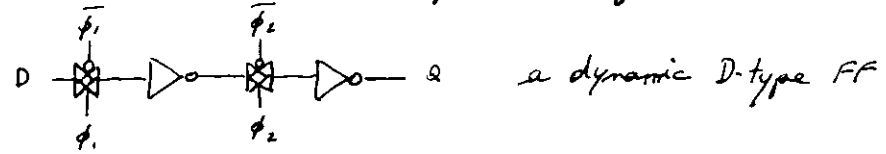
Dynamic structures are also possible
$P_1 \ldots P_4$ specify the wanted logic function which is applied to the 2 operands A, B.

Remember that in CMOS both p and n transistors are used, so that a larger area and capacitance are peculiar to CMOS implementations.
Because of that, it is not always convenient to use transmission gate logic.
Each single case must be carefully analyzed.

When long chains of pass transistors are used, in order to avoid long delays, the chains must be interrupted and buffers included. As a rule of thumb, the chain should be interrupted where the delay introduced is equal to the buffer's delay.

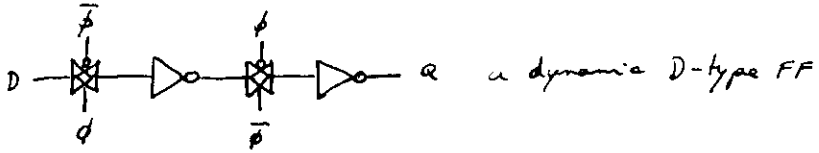
Clocking strategies

Pseudo 2-phase
It takes the 2-phase nonoverlapping nMOS clocking scheme and adds the complementary clocks.



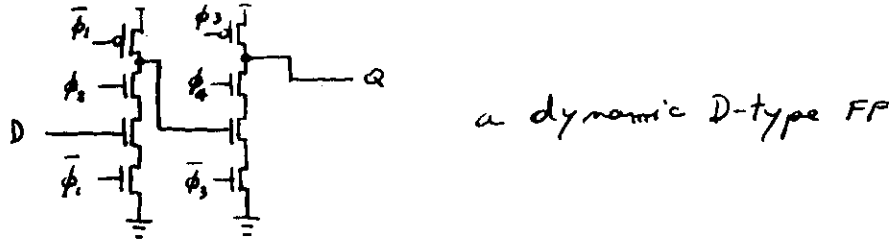
a dynamic D-type FF

2-phase clocking
It uses ϕ and $\bar{\phi}$ in order to reduce the number of clock lines that are routed around the chip.



a dynamic D-type FF

4-phase clocking
The main purpose in adopting this scheme is to enable the 4-phase logic gates to be built.
For bit-serial circuitry, where clock routing and flip-flop complexity is important, this scheme may be the most suitable one.



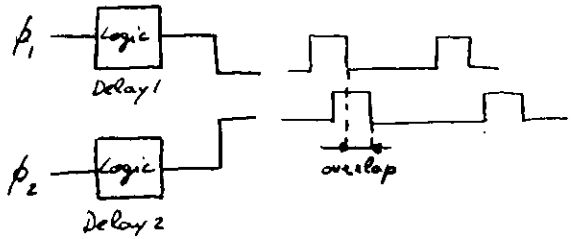
a dynamic D-type FF

No matter how many phases, all these nonoverlapping schemes suffer from clock skew problems.

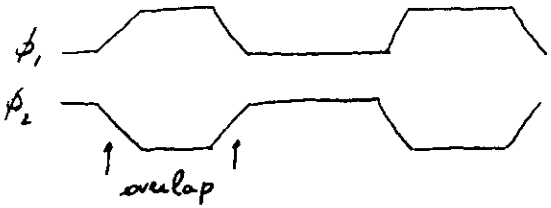
(CT)

Clocking strategies

Clock skew can occur in two forms. One is due to the different delays which occur through different clock paths.
The second happens when the rise and fall times are so slow that the period of the transition region causes the latch transmission gates to couple.

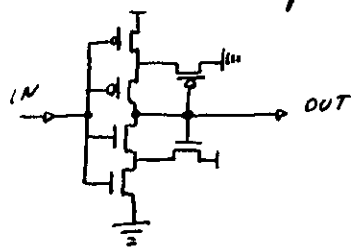


Skew due to different delays

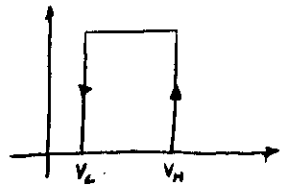


With a careful clock distribution design, the 1st kind of skew can be minimized to a negligible value.

In the second case a more careful buffering scheme must be used or the use of Shmitt Triggers



A typical CMOS Shmitt Trigger



(ST)

Optimizing CMOS gates

The pull-up and pull-down capability of the CMOS gates determine the performance of the circuit.
Delays in CMOS are technology (and topology) dependent, the only way a designer can optimize the speed of a critical logic path, is to size transistors accordingly.
In a full-custom approach there is the additional freedom of resizing p-type and n-type devices according to the performance we want to obtain, capacitance and complexity of the implementation.
SPICE or similar circuit simulators can be used only during the optimization process of very small circuits or when an accurate solution is required, but are out of the question when a large portion of a VLSI chip has to be optimized.
We need a new methodology, simple, easily implementable with a program, which can approximate the optimal solution of the difficult problem of sizing transistors.

The theory we are going to present here (just the very basic points) comes from a work done by I. E. Sutherland and R. F. Sproull, sponsored by Austek Microsystems, Digital Equipment, Floating Point Systems and Schlumberger.
This theory was called "Theory of Logical Effort" by Sutherland and Sproull and during this brief presentation, we are going to use the same names and definitions they first introduced.

The Theory of Logical Effort

This theory provides approximated results, but in real situations and large number of different cases, it works unbelievably well.
The basic idea is that the delay of a logic element (CMOS, but can be applied to other technologies too) depends on two terms:

- 1 Logical effort which tells how much worse a logic element is (compared with a basic inverter) at producing output current. Logical effort depends on the topology of the circuit implementation only.

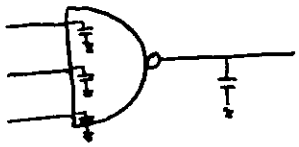
- 2 Electrical effort, it is just the ratio of the output capacitance and input capacitance and represents the electrical constraint imposed by loading conditions of the circuit. Notice that the input capacitance to a logic element is determined by the width of the transistors used. The output capacitance is due to wiring (stray capacitance) and the input capacitance of the following gate.

Delay = Logical effort * Electrical effort

$d = g \cdot h$

$h = \frac{\text{Load capacitance}}{\text{Input capacitance}} = \frac{C_{out}}{C_{in}}$

g = depends on the topology and is normalized to the simplest and most efficient basic block: an inverter.



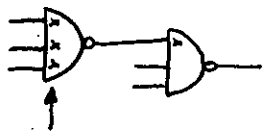
$$delay = g \cdot h = g \cdot \frac{C_{out}}{C_{in}} = g \cdot \frac{\text{width of load tns}}{\text{width of driving tns}}$$

Notice that we can define a total effort for the whole logic gate (all inputs) or just for one of the inputs.

LOGICAL EFFORT FOR SINGLE INPUT 2/1 CMOS (STATIC LOGIC)

N. of inputs		2	3	4	5
Type of gate					
INVERTER	1	–	–	–	–
NAND		4/3	5/3	6/3	7/3
NOR		5/3	7/3	9/3	11/3
MULTIPLEXER		2	2	2	2
MÜLLER C		2	3	4	5
XOR		4	12	32	

Ex:

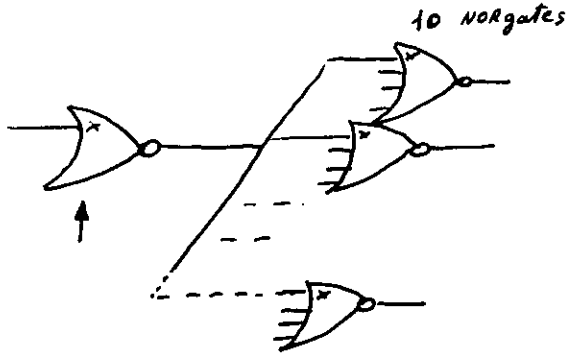


same gates

$g = 5/3$

$$delay = g \cdot h = \frac{5/3 \cdot x}{x} = 5/3$$

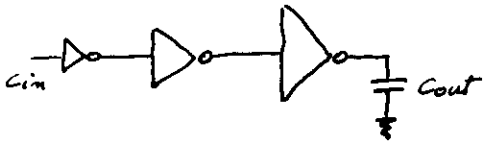
Ex:



$g = 9/3 = 3$

$$delay = g \cdot h = \frac{3 \times 10 \cdot x}{x} = 30$$

Ex:



$C_{out} = 30pF$

$C_{in} = 60fF$

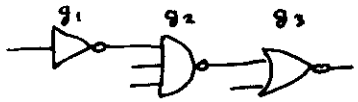
total effort $= g \cdot h$ but $g = 1$ (inverters)

$= 500$

Compound effort

The logical effort of a string of logic element is given by multiplying the logical effort of each element.

Ex:



$g_{tot} = g_1 \cdot g_2 \cdot g_3$

$g_1 = 1$

$g_2 = 5/3$

$g_3 = 5/3$

$= 1 \cdot 5/3 \cdot 5/3 = 25/9$

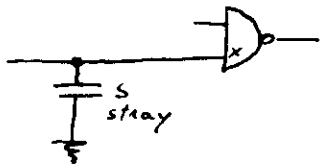
(ST)

Branching effort

Not always the current is driven into the capacitance of the next gate (active capacitance) and gets converted into other current at the output of this gate.
Current gets lost when goes into stray capacitance (passive capacitance).
This effect can be described by a branching effort.

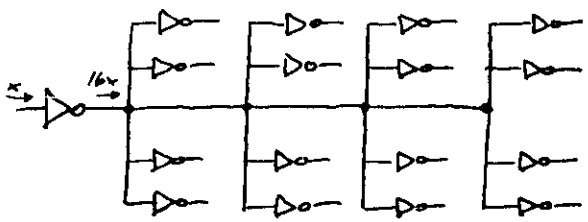
$$b_e = \frac{\text{total capacitance}}{\text{useful capacitance}}$$

Ex:

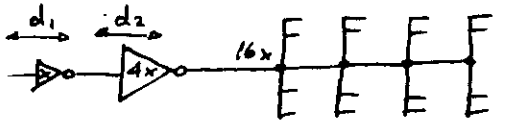


$$b_e = \frac{x+s}{x}$$

Ex:



$$d = \frac{1 \cdot 16x}{x} = \underline{\underline{16}}$$



$d_{tot} = d_1 + d_2$

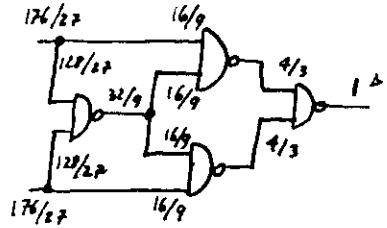
$d_1 = \frac{4x}{x} = 4$

$d_2 = \frac{16x}{4x} = 4$

$d_{tot} = 4+4 = \underline{\underline{8}}$

same driving capability, but less total delay

Ex: exor gate made with 2 NAND's



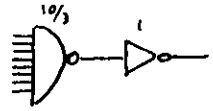
let us start from here by setting the output to 1

11
6.52

Logical effort of a single input = 6.52

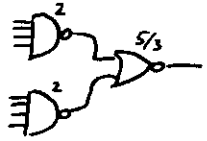
Ex: how to build an 8-input AND

case 1



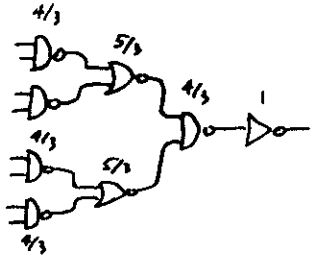
$g = 10/3 = 3.33$

case 2



$g = 10/3 = 3.33$

case 3



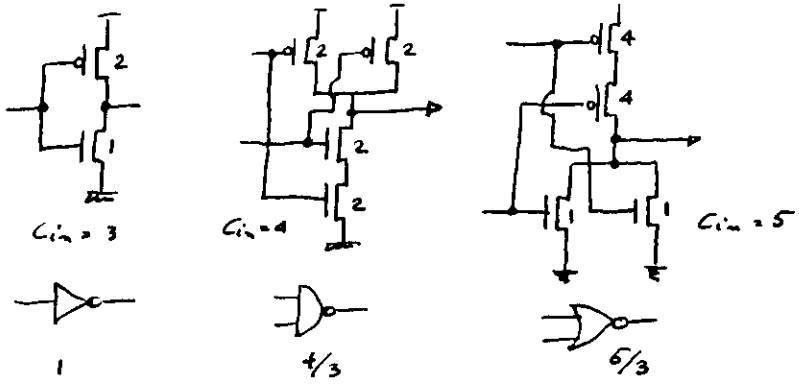
$g = 1 \cdot 4/3 \cdot 5/3 \cdot 4/3 = \frac{80}{27} = 2.96$

better, less logical effort

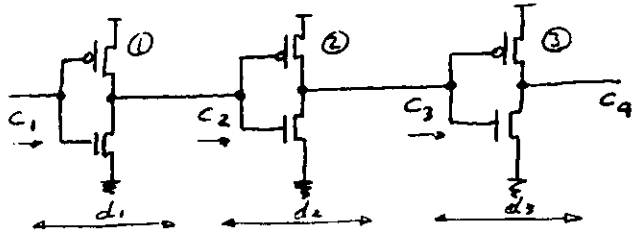
Calculating logical effort

$g = \frac{\text{capacitance of 1 input of logic gate}}{\text{total input capacitance of inverter with same drive as logic gate}}$

Assumption:
- symmetric pull up/down capability
- p-fets twice as big as n-fets
- static logic only



Optimizing strings of inverters



string of 3 inverters

$d_1 = \frac{C_2}{C_1}$ $\quad d_2 = \frac{C_3}{C_2}$ $\quad d_3 = \frac{C_4}{C_3}$

$D = d_1 + d_2 + d_3$

$\frac{\partial D}{\partial C_2} = \frac{1}{C_1} - \frac{C_3}{C_2^2} = 0$ $\quad \frac{\partial D}{\partial C_3} = \frac{1}{C_2} - \frac{C_4}{C_3^2} = 0$

$C_2 = \sqrt{C_1 C_3}$ $\quad C_3 = \sqrt{C_4 C_2}$

$C_2 = C_1^{1/2} \cdot C_2^{1/4} \cdot C_4^{1/4}$ $\quad C_3 = C_1^{1/4} \cdot C_2^{1/4} \cdot C_4^{1/2}$

and so on

$d_1 = \frac{C_2}{C_1} = \frac{C_1^{2/3} \cdot C_4^{1/3}}{C_1} \left(\frac{C_4}{C_1}\right)^{1/3}$

$d_2 = \frac{C_3}{C_2} = \frac{C_1^{1/3} C_4^{2/3}}{C_1^{1/3} \cdot C_4^{1/3}} = \left(\frac{C_4}{C_1}\right)^{1/3}$

$d_3 = \frac{C_4}{C_3} = \frac{C_4}{C_1^{1/3} \cdot C_4^{2/3}} = \left(\frac{C_4}{C_1}\right)^{1/3}$

$d_1 = d_2 = d_3 = \left(\frac{C_4}{C_1}\right)^{1/3}$ The minimum total delay is when all delays (intermediate) are equal

In general if n = n° of stages and H = electrical effort (total)

$D = n \cdot H^{1/n}$

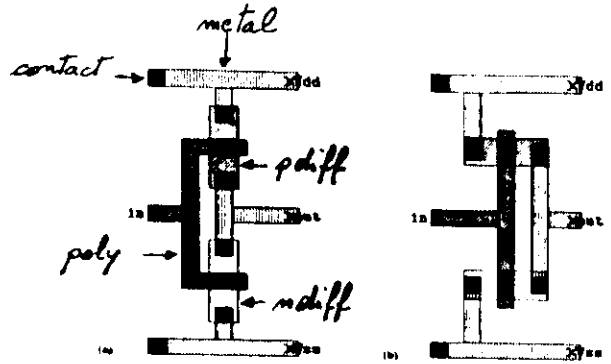
$\frac{\partial D}{\partial n} = -n \cdot (H^{1/n} \ln H) \cdot \frac{1}{n^2} + H^{1/n} = 0$ $\quad H^{1/n}$ cancels out

$\frac{1}{n} \ln H = 1$ or $n = \ln H$ optimized number of stages

Electrical and physical layout of logic gates

The layout style is very much affected by many parameters and requirements.
The first set of constraints comes from system design specification, interconnection, complexity and shape of the final block of logic.
Another set of constraints is related to the technology itself and layout rules drive the designer to choose the right topology which turns out to be the best in terms of area and performance.
In the following examples an 1-metal layer, n-well process is supposed to be available.
In today's technologies a second metal layer is always available, allowing the implementation of almost every possible topology avoiding the use of non-metal interconnections.
This improves not only the cell compactness, but its performance also.
Remember that layout implementations which have low output capacitance and don't have additional resistance in the current paths, show the best performance. Lowering the output drain capacitance of the gates by placing transistors back to back or by using other techniques, is the general rule that must be applied when performance is the first concern.

Physical layout of inverters



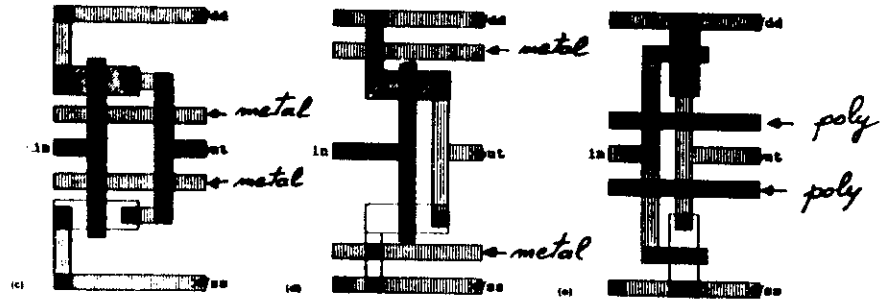
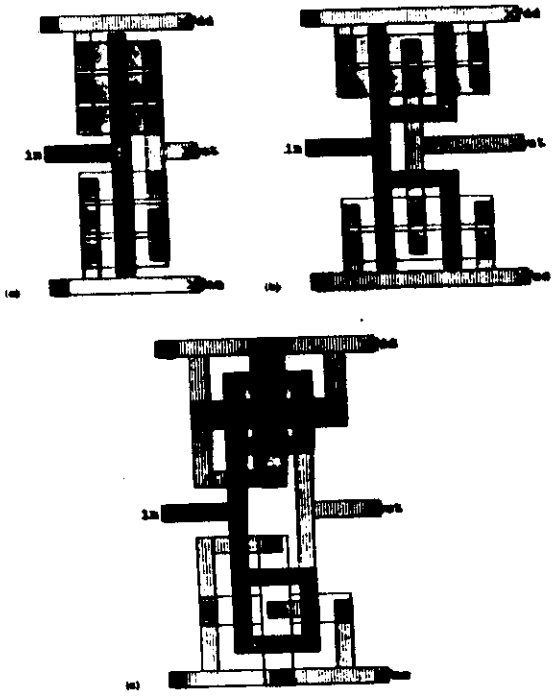


FIGURE 5.17. Various forms of an inverter layout

Figures a) and b) show the two basic layouts using a 1-metal process.
Figures c), d), e) show variations in order to make the cell transparent to metal lines running horizontally or poly lines.
Notice that, wherever contacts, poly and diffusion are added to the more "natural" implementation, additional delay occurs.

Paralleled inverters



Parallel inverters are used whenever a large load has to be driven. Following the general rule, structures that have small drain capacitance and source-drain resistance are more convenient in terms of performance and area.

a) shows the simplest implementation
b) shows a back to back transistor placement
c) a star connection

(ST)

Nand gates

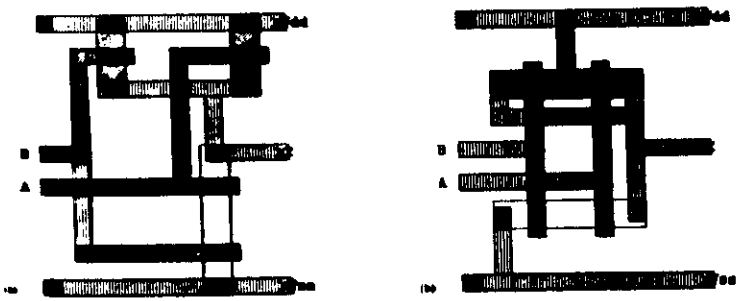


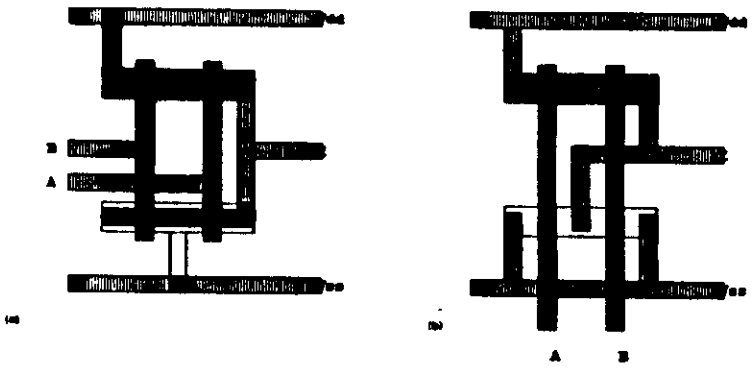
FIGURE 8.10. NAND layouts

Figures a) and b) show two possible implementations. Note that in fig. b), where transistors are placed horizontally, the layout is cleaner (and smaller). This is true in general for multi-input gates and unless some other constraints want a different transistor orientation, the horizontal placement is the best and should be used by default.

Remember: in the examples shown so far, transistors are not sized according to the optimal electrical performance. Bear in mind that p-transistors are weak, therefore structures with p-transistors in parallel (NAND) are better.
It depends also on the transition (critical) we are interested in optimizing.

(ST)

Nor gates



Different implementations of NOR gates.
The second one b) has a smaller drain capacitance so it's performance is better.

Important: in all the reported examples, n-well areas as well as their connections to Vdd are not shown. Because of that and when rows of gates like those are used, then mirroring cells around the horizontal axis is the best way for improving performance and optimizing area.
If large fan-in gates have to be used, body effect must be taken into account.
Usually gates with more than 4 inputs are not used.

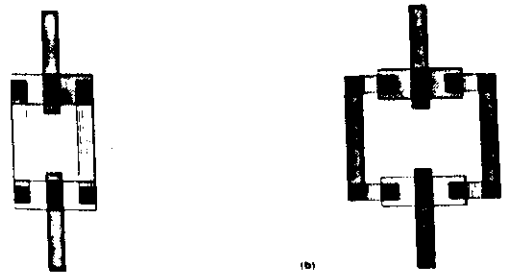
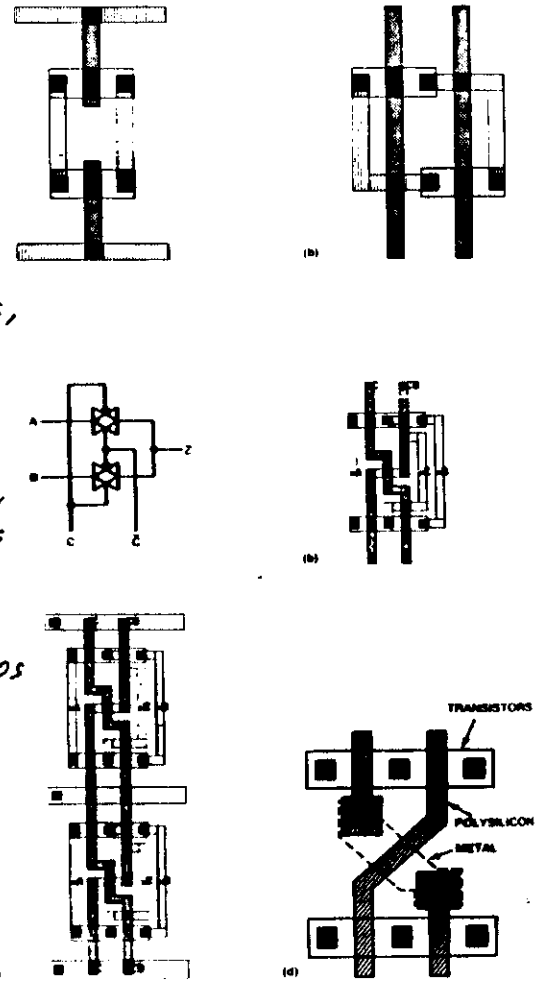


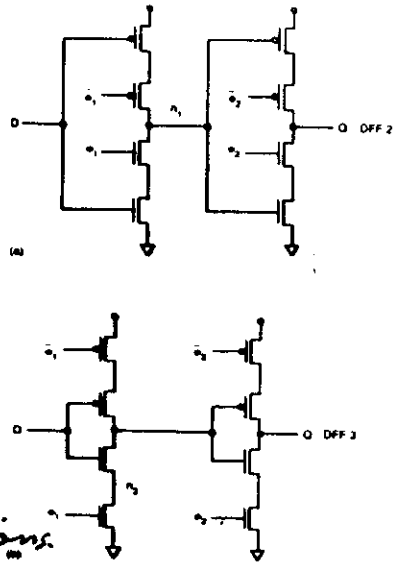
FIGURE 5.36. Transmission gate layout

Transmission gate physical layout

Transmission gate is a basic block in C-MOS circuit design and is used in memory structures, multiplexers, decoders, random logic.
In order to provide the right HIGH and LOW levels, p-type and n-type transistors must be used.
This increases the area and capacitance of TG CMOS gate compared with TG NMOS gate.



Clocked FF



A clocked version of a FF, is shown in fig a) and d). Notice that the circuit of fig. b) has charge sharing problems and should be avoided.

Clocked structures are usually more compact and faster, but require more accurate simulations.

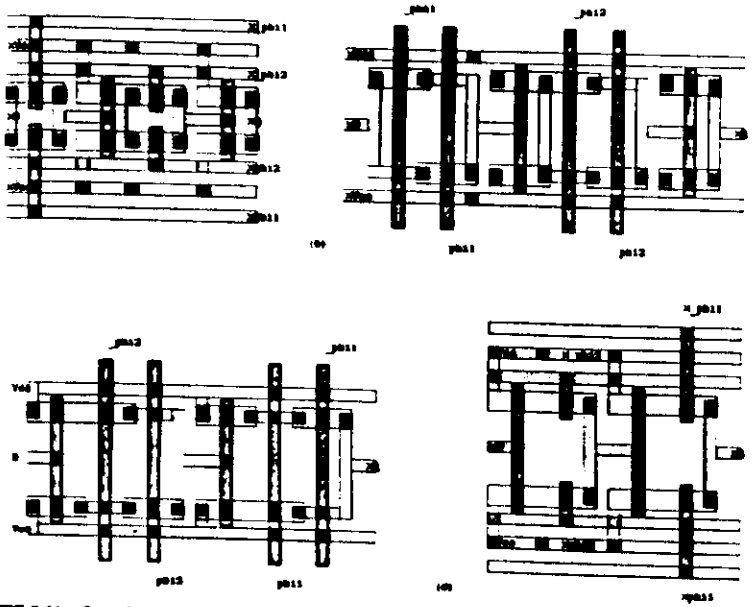


FIGURE 5.41. Pseudo 2-phase latch layouts

When the layout of a basic cell, which will be part of a larger structure, is undertaken, interconnections among them as well as control signals, clock and data interconnections have to be taken into account, in order to optimize area and performance of the final block.

The example shows an efficient implementation of a 4-bit shift register.
Notice the compactness of the layout.

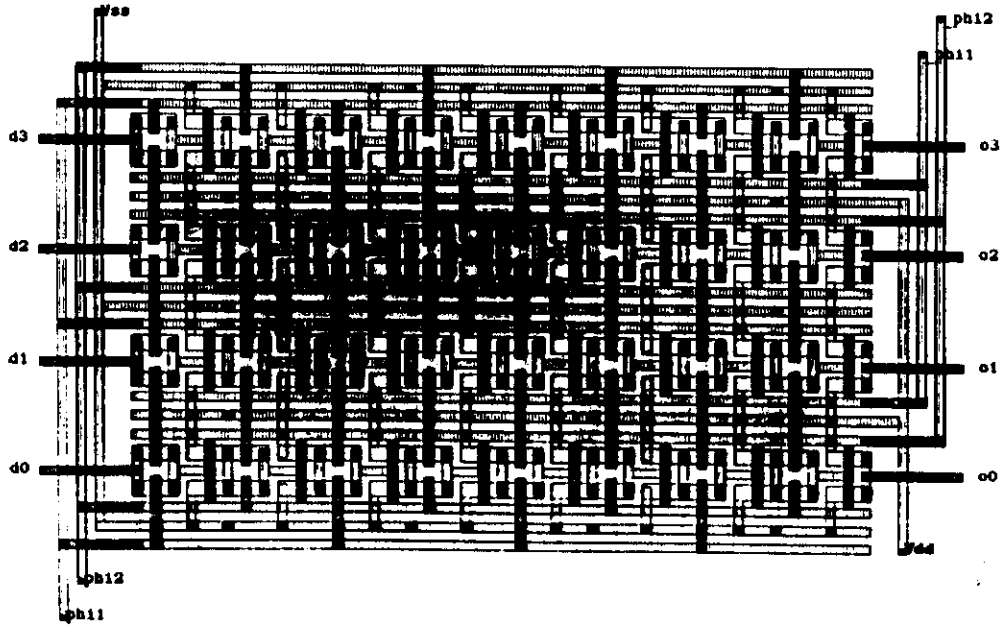
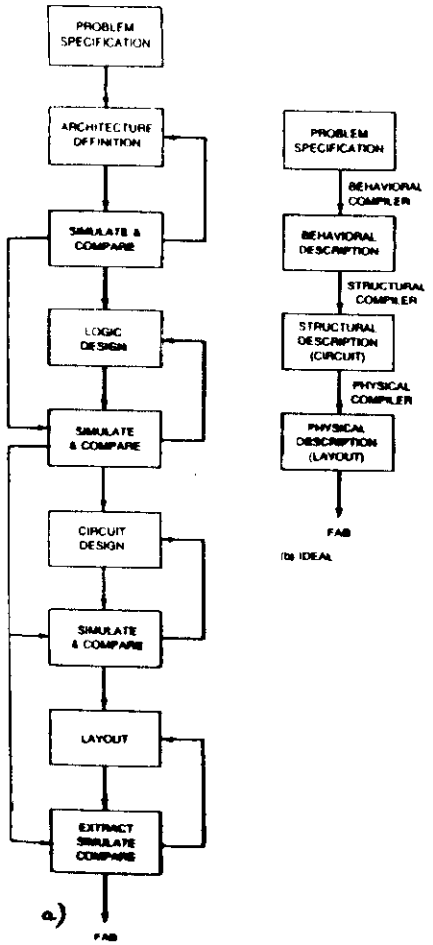


FIGURE 5.42. Shift register array layout

System Design



The design description for an integrated circuit may be described in Terms of 3 domains:

- Behavioral domain
- Structural domain
- Physical domain

fig. a) shows the current design flow up to fabrication and fig. b) the ideal approach.

(ST)

System design

The complexity of today's VLSI systems is such that, no matter which approach is going to be undertaken, some techniques must be used for reducing the complexity of IC design.

- Hierarchy — This involves dividing a module into submodules and then repeating this operation until the complexity of the submodules is such that it is comprehensible at the wanted level of detail. Using an analogy with software design, this is like writing subroutines and breaking the problem down to simple atomic subroutines. (Top-down design)

- Modularity — If modules are "well formed", interactions with other modules can be well characterized. By analogy, this has to do with ambiguity and how software engineering avoids that. In Modula 2, for instance there is a module definition where the interface (connections) is defined, and an implementation where what the module actually does, is specified.

(ST)

System design

- Regularity — Making a structure regular is the best way to ensure semplicity of design.
Regularity can and should be consistently kept at all levels of the design.
Using regularity a design may be judged correct by construction.

- Locality — "Well characterized" interfaces provide a form of "information hiding" that reduces the apparent complexity of a module. In the software world this is paralleled by the reduction of global variables to a minimum.
Modules can also be located to minimized the "global wiring" that may be necessary to connect a number of modules in an unstructured system.

A common theme in design systems today is use "wires first, then modules" rather than the more common "place modules, then route them together".

Needless to say, that a VLSI design today, looks like a software design.

System design
(the design tools)

- Circuit level simulation
Circuit analysis programs, such as SPICE, are used. This simulation is characterized by high accuracy, but long simulation times.
Simulation time is typically proportional to n^m where n = number nonlinear devices in the circuit and m = in the range 1 to 2.
Unrealistic for a global VLSI chip simulation, even with supercomputers.

- Timing simulation and verification
Delays through all paths in a circuit are evaluated and the designer is provided with information about these delays.
Simple structures can be recognized and if the right timing is not applied to input/output signals, the error can be detected.

- Logic level simulation
The logic performance of the system can be analyzed. Standard gates (AND, NOR, NAND etc.) can be easily and efficiently simulated. Circuits which use transmission gates may have problems.

System design
(design tools)

- Switch level simulation
A good way of ensuring correspondence between layout and simulation is to use transistors as primitives in the simulator. They are simulated as they were switches and timing can be taken into account by replacing transistors with equivalent resistors a capacitors at the nodes of the net.
Timing information is not as good as can be with circuit simulators, but the execution time is order of magnitude less than SPICE or similar programs. Recently more accurate models have been investigated and the results are not too far (~10%) from the most accurate simulators.

- Schematic editors
At the beginning of an I.C. design, schematic description is probably the most common and sometimes better way of specifying a new system. These systems do not only offer graphical editing but feedback from simulation too.

- Net list comparison
If layout is not generated automatically, as in the case of Silicon compilers or Gate Arrays and Standard Cells CAD, then a comparison between

physical implementation and desired circuit (schematic or other kind of specifications description) is needed.

- Layout editors
They allow to manipulate multilevel polygons and build the necessary devices and interconnections. Design rules (layout rules) can also be included into some of them and the work of the designer can be monitored as it proceeds.

- Circuit extractors
From the final layout, they extract all the requested connectivity of the devices and parasitic capacitances and resistances.
The system can be simulated again after layout has been completed.

Many other programs can be part of the design tool set, like optimizers, tools for testing etc. etc.

Even if CAD is getting more complex and programs larger, hardware costs lower every day and today a very sophisticated CAD can be installed in low cost workstations.
Main frames or supercomputers may be required only for exaustive simulations on the whole VLSI chip.

Gate Arrays

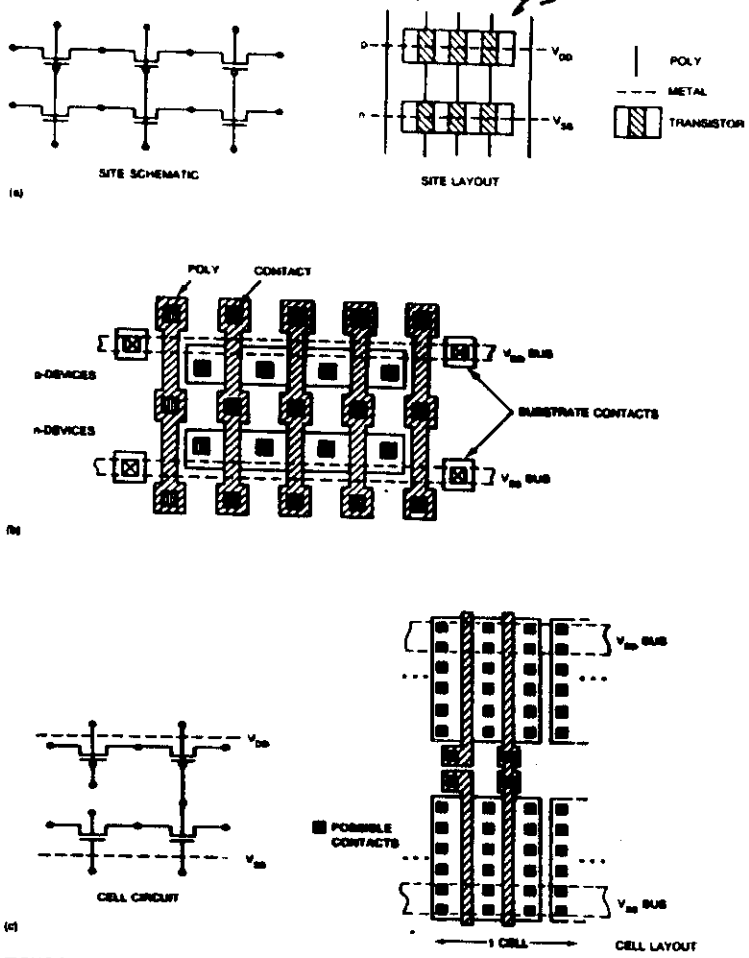


FIGURE 6.2. Gate array site configurations

Gate arrays come in various flavors, but can be categorized by a design that uses a large number of identical "sites", where n-transistors and p-transistors are placed and connected each other according to some arrangement.

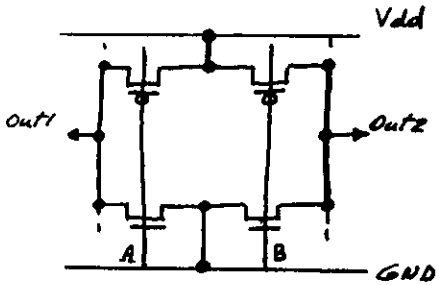
In the previous figures, 2 different arrangements are shown.

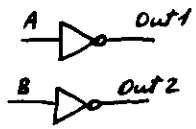
In a) 3 p-transistors and 3 n-transistors are connected in series and node between them and gates can be interconnected by customer's specification.

In c) a 2, n and p transistor structure is implemented. By interconnecting intermediate nodes of the structure and gates with metal lines, all the basic functions can be implemented.

Notice that connections have to be done such a way that the remaining transistors are still available for other functions. If a designer (or software) doesn't apply this rule, a lot of transistors will remain inutilized.

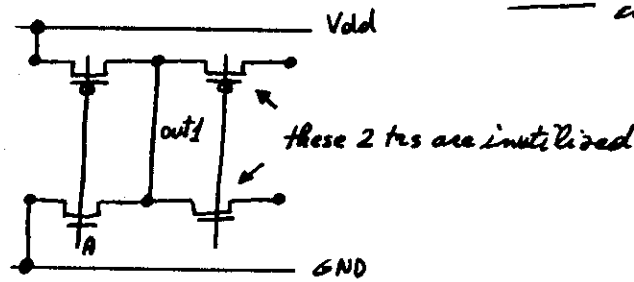
Ex:



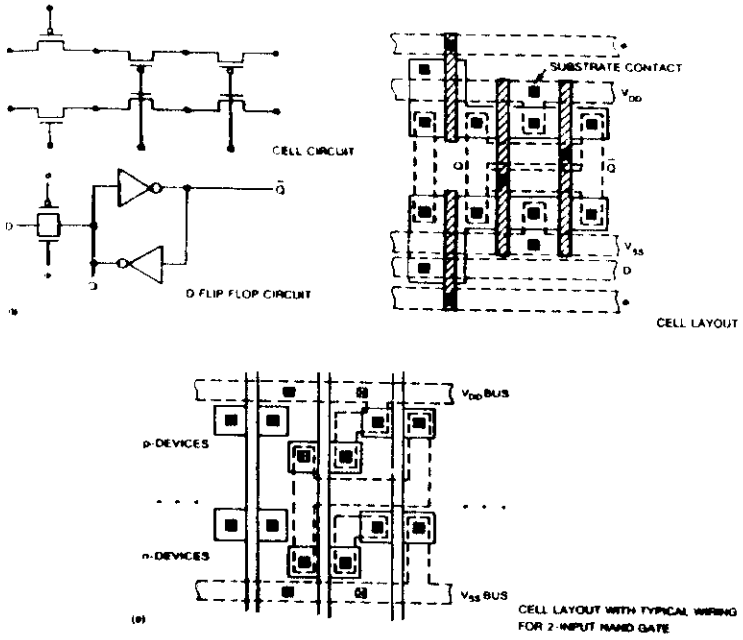


Right interconnection pattern

—— custom connection



Gate arrays can have optimized structures for special circuits



The one shown here, allows an easy implementation of transmission gates and inverters as might be used in latches. This structure can be very convenient to use with memory structures.

In the examples shown, only 1-metal layer is used. Today's technology offers 2-metal and even 3-metal layers for interconnections. This simplify automation by software and avoid incurring in long delays due to long interconnections made with poly lines.

Gate array design flow

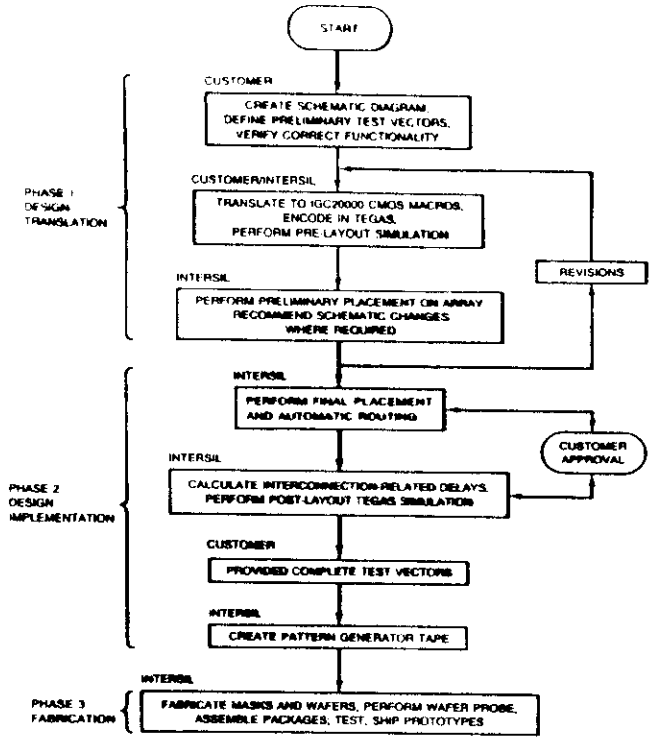
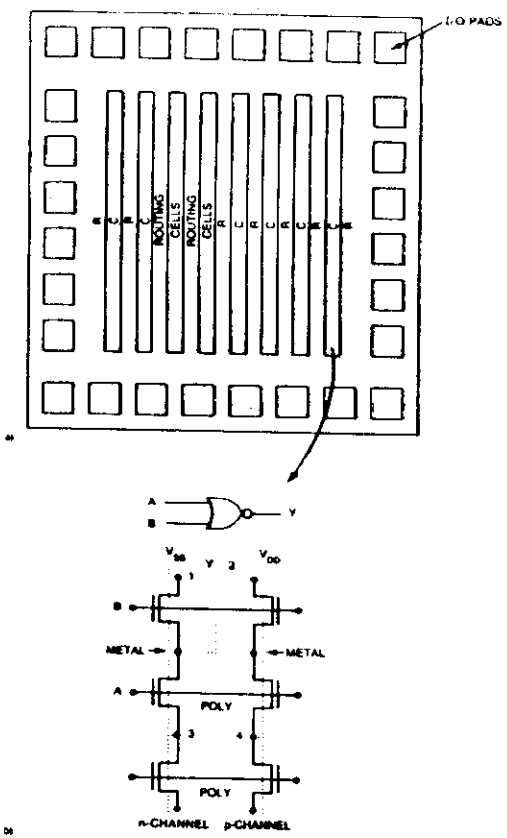


FIGURE 6.4. Gate array design flow (GEIn83)

With available CAD systems, customers are only responsible of logic schematic and the set of test vectors, which are used to verify customer's logic.
It is usually available a library of predefined function (more complex than nand or nor gates) so that the designer's task is made easier.

Gate Arrays



This is how the floor plan of a Gate Array looks like. Notice that routing channels are included between rows of logic cells. This is the standard arrangement. Today, new Gate Arrays, called "sea of gates" are offered. The topology of the cell is such that routing channels are not required any more.

Adders

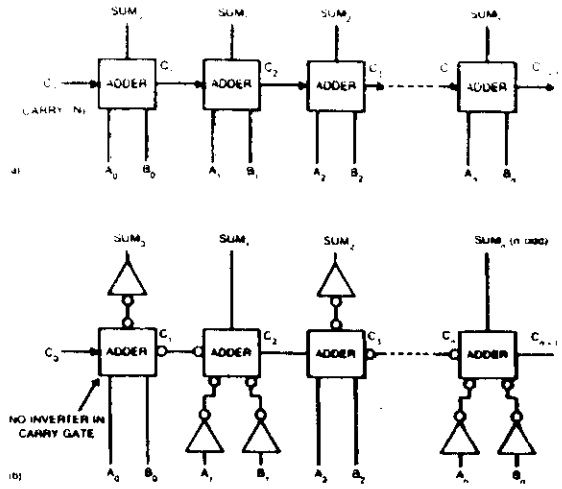


FIGURE 8.2. n-bit ripple carry adder

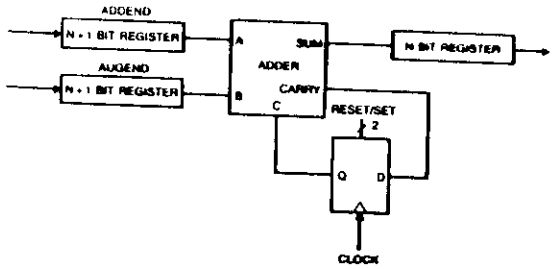


FIGURE 8.3. Serial adder schematic

When speed is not a problem a serial approach can be more convenient, but in general adders are always in a critical path, and parallel structures must be used. A ripple carry adder is the simplest implementation (too slow most of the times)

Adders

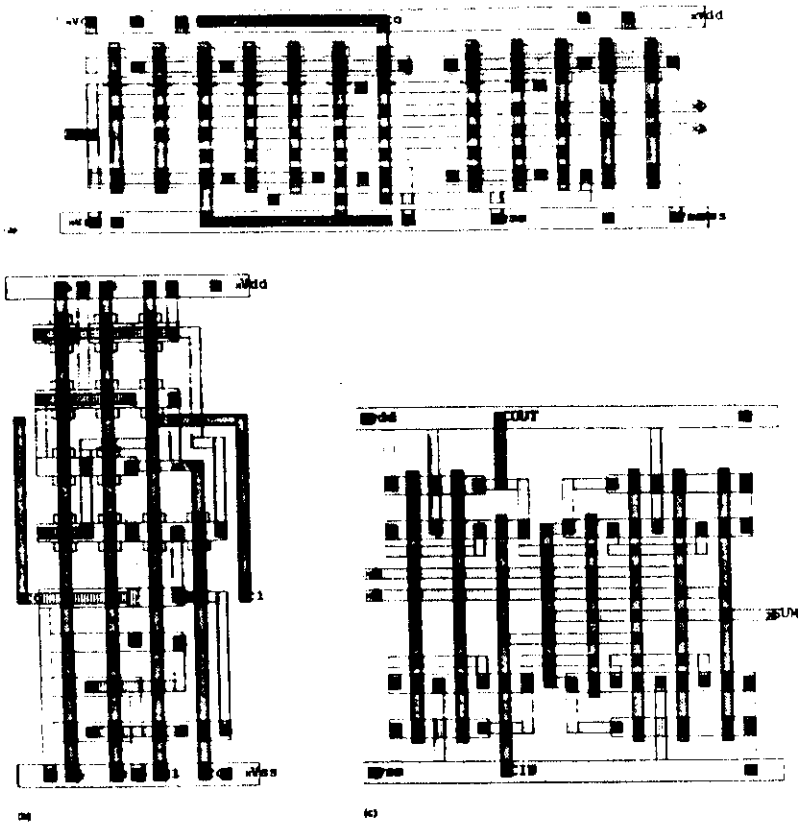


FIGURE 8.5. Combinational adder layouts

The equations are: $SUM = ABC + A\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}\bar{B}C$

A, B operands $CARRY = AB + AC + BC$

C = carry (previous)

This slide shows some layouts for the cells which generate the carry and the Sum according to the equation.

Adders

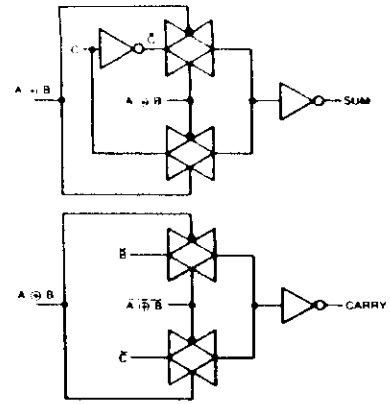


FIGURE 8.8. Transmission gate adder

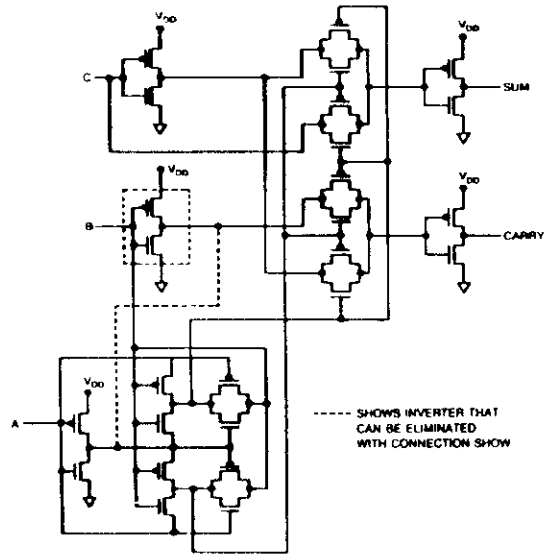


FIGURE 8.9. Complete TG adder

When the same delay for the Carry and Sum generation is requested, this scheme can be used.
Notice the an XOR and XNOR gates must be used.
Their circuit implementation is very interesting and very efficient.

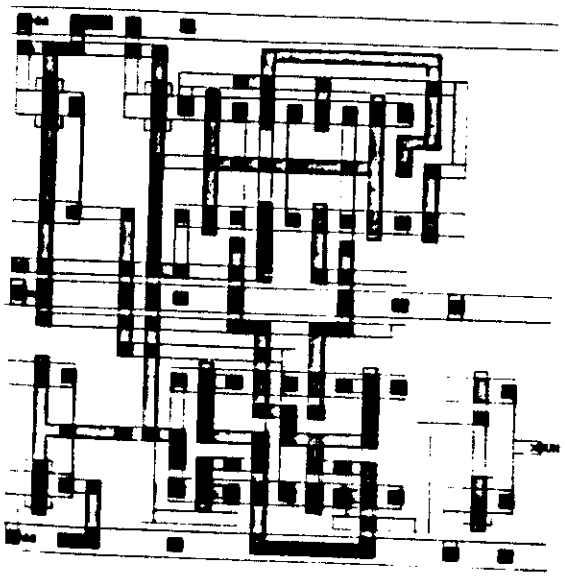


FIGURE 8.10. TG adder layout

The layout for a transmission gate adder is shown.

A very different layout style has been used.
This is because the high usage of transmission gates.
This adder has 24 transistors, the same as the combinational adder seen before.

The equations can be rearranged as:

$C_i = G_i + P_i \cdot C_{i-1}$

$SUM = P_i \oplus C_{i-1}$

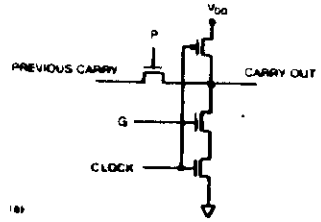
where

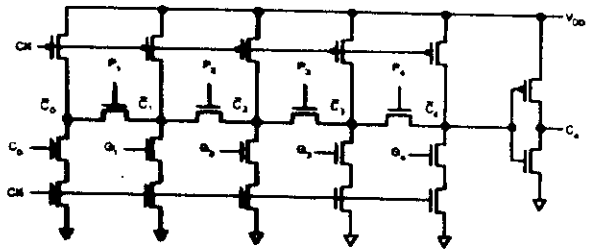
$P_i = A_i \oplus B_i$ (propagate)

C_{i-1} = previous carry

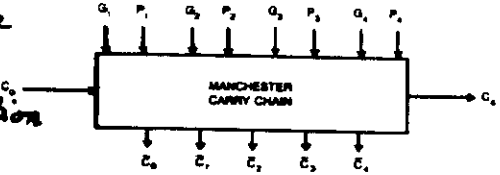
$\oplus$ = xor

$G_i = A_i \cdot B_i$ (generate)





Manchester carry chains are very efficient structures for the next carry computation



Notice that the structure is dynamic

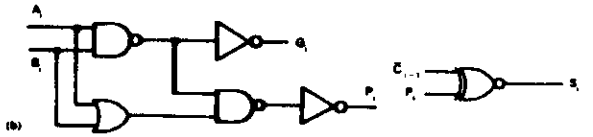
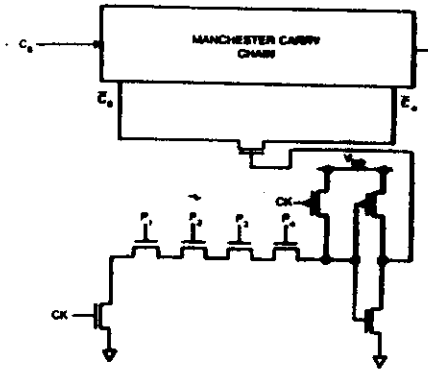


FIGURE 8.14. Manchester carry chain



If all $P_i = 1$ in the group, a carry can be propagated directly to the output of the group.

FIGURE 8.15. Manchester lookahead circuitry

Adders

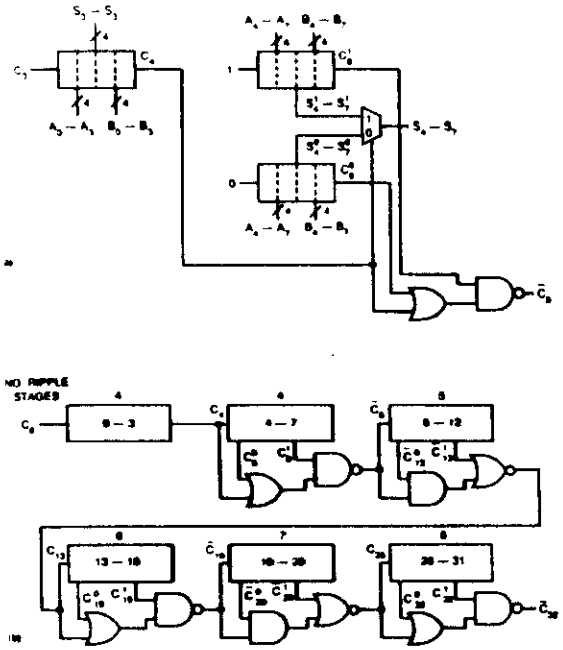


FIGURE 8.22. Carry select adder

Another possible architecture for speeding up the addition.
Carry select adders are not very interesting in VLSI applications, because they require a large silicon area.
A better approach are schemes which use a "multilevel carry-skip" approach.

Multipliers

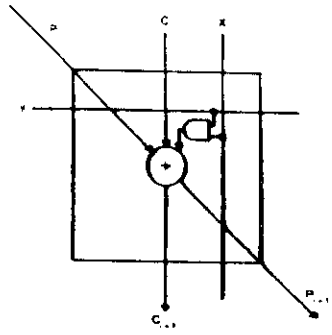


FIGURE 8.35. Parallel multiplier cell

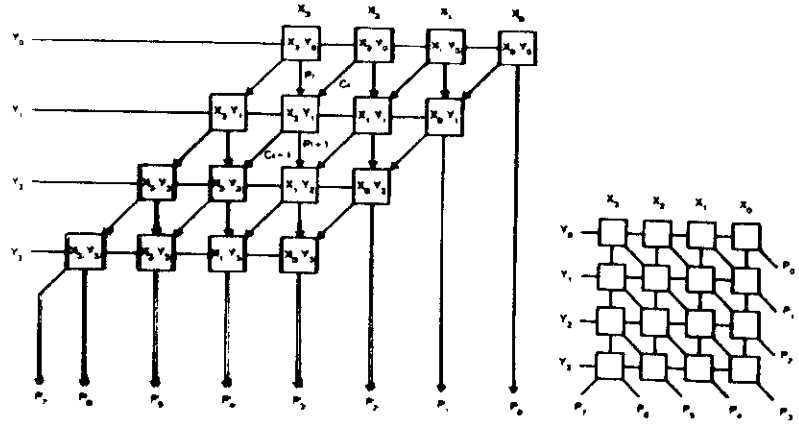


FIGURE 8.36. Parallel multiplier array

Multipliers can be devided into :- serial
- parallel

Parallel multipliers are the fastest and the most common functional blocks which implement this operation.

Multipliers

All parallel multipliers are based on this observation:

$$X = \sum_{i=0}^{n-1} X_i 2^i$$

$$Y = \sum_{j=0}^{m-1} Y_j 2^j$$

$$P_2 = X_i Y_i = \sum_{i=0}^{n-1} X_i 2^i \cdot \sum_{j=0}^{m-1} Y_j 2^j$$

$$= \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} (X_i Y_j) 2^{i+j}$$

$$= \sum_{K=0}^{m+n-1} P_K 2^K$$

where

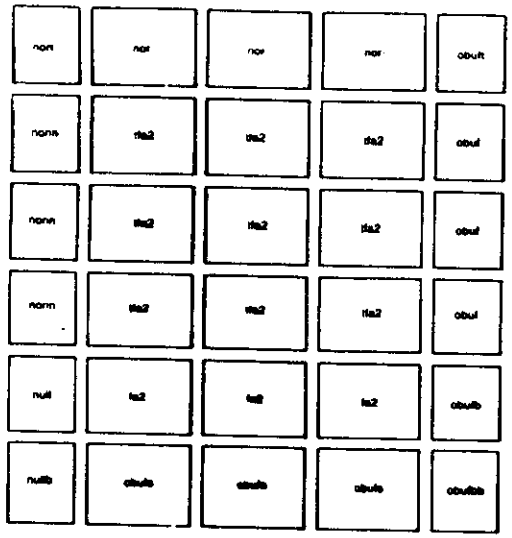


FIGURE 8.37. Parallel multiplier floor plan (4*4)

P_K = partial products called summands.
Notice that in this way the operation is broken down in simple AND operations and additions. If we want an efficient multiplications, adders must be fast and efficient too.

In the previous slide an efficient implementation of a parallel adder is shown.
Notice the high regularity of the structure. Only one basic cell is repeated at each bit position.
In this slide a floor plan of a 4*4 multiplier is shown. For the real implementation we need different cells to take into account, connections, buffering stages etc.

Multipliers

FIGURE 8.38. Multiplier cell layouts

A possible layout for some cells implied in the scheme previously proposed.

Shifters

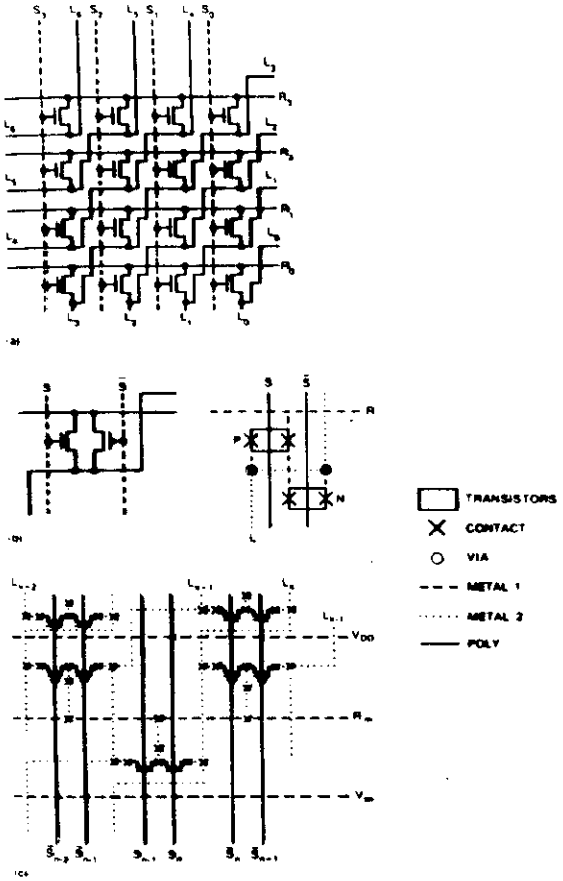


FIGURE 8.62. Barrel shifter implementation

Shifters are common blocks in computers or DSP.
Depending on the data path width, they are usually large blocks and therefore an efficient and compact implementation is very important.
Sometimes only a preferential direction is needed (right shift or left shift). In this case the block shape is triangular.

(ST)

Memory structures (RAM, ROM, static, dynamic)

In computer systems or applications which require these kind of structures anyway, memory blocks are always in the critical path and are the bottleneck of the whole system, most of the times.
Lots of designers are involved with RAM, ROM design and architecture definition and partition of memory.
The techniques for generating all the required signals for accessing the basic memory cell, can be very different and complicated.
The basic operation, anyway, consists in a column and row decoding and a read or write operation on the basic memory structure.
The basic memory cell is made up by a cross-coupled inverter connection and transfer gate transistors.
Usually special processes are used for memory fabrication.
If big blocks of memory have to be included in VLSI chips, then sometimes some vendors offer processes with additional steps for optimizing RAM or ROM structures.

In the following slides the basic cells with some variations are shown, as well as the logic for decoding columns and rows.

(ST)

Memory (RAM) static.

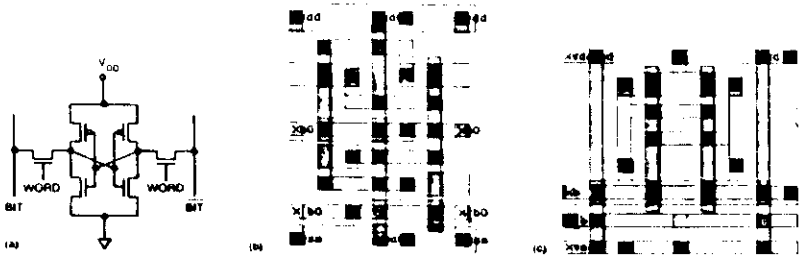


FIGURE 8.41. Static CMOS RAM cell circuit and symbolic layouts

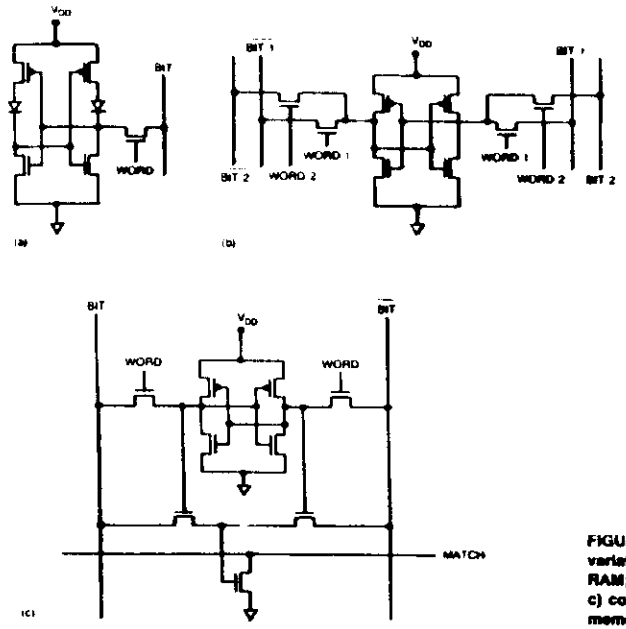


FIGURE 8.42. CMOS RAM variations a) 5-transistor RAM; b) 2-port RAM; c) content-addressable memory (CAM)

Memory (RAM)

Some possible decoding and sensing schemes for RAM structures.

The complexity of the sensing scheme is due to noise problems which are always present with these structures.

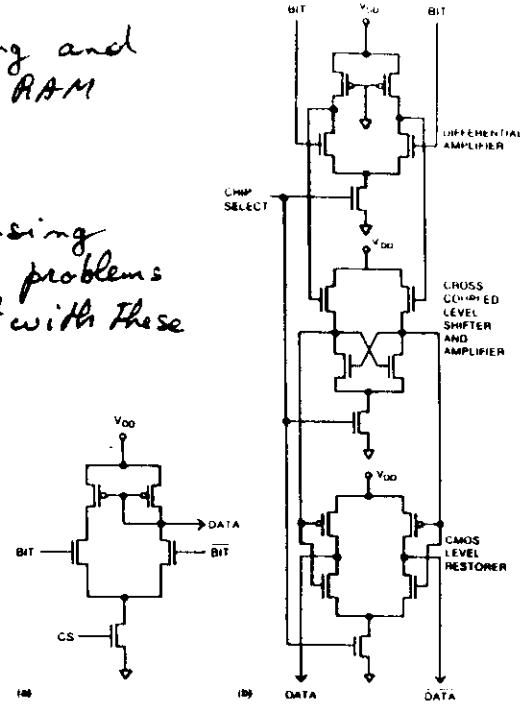


FIGURE 8.60. Differential sense amplifiers

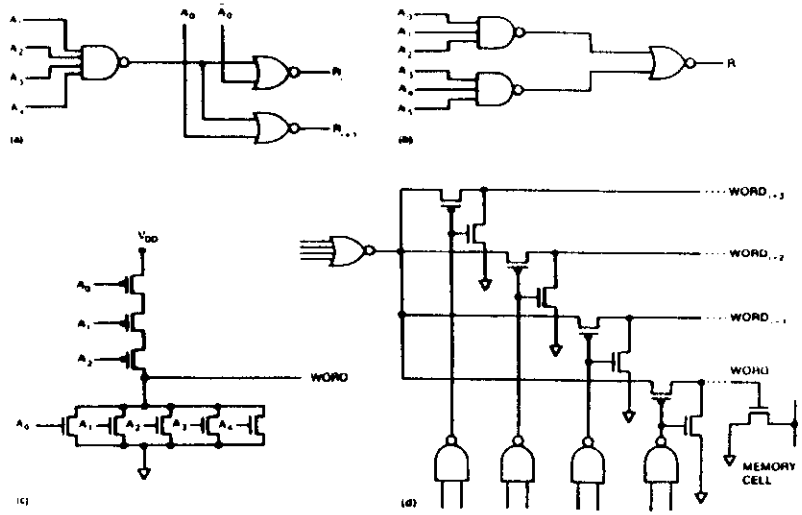


FIGURE 8.50. Static row decoder circuits

Memory (RAM)
dynamic

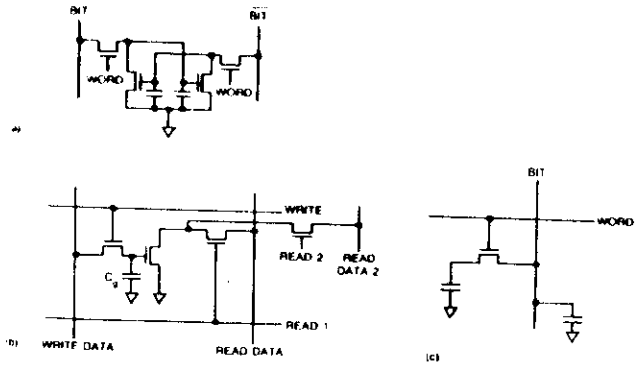


FIGURE 8.45. Dynamic RAM circuits

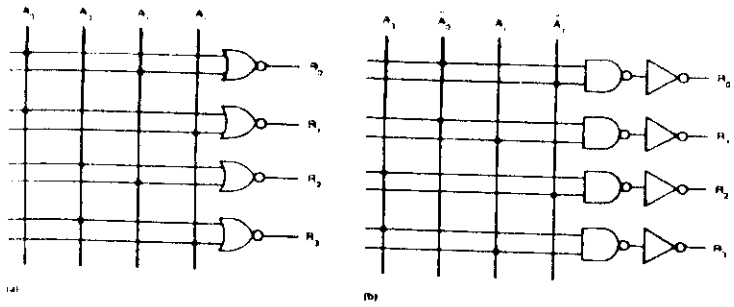


FIGURE 8.49. Row decoder logic schematics

Memory (ROM)

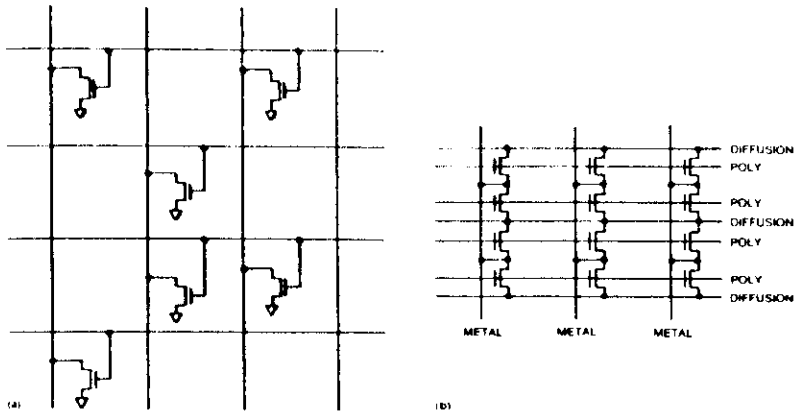


FIGURE 8.47. ROM arrays

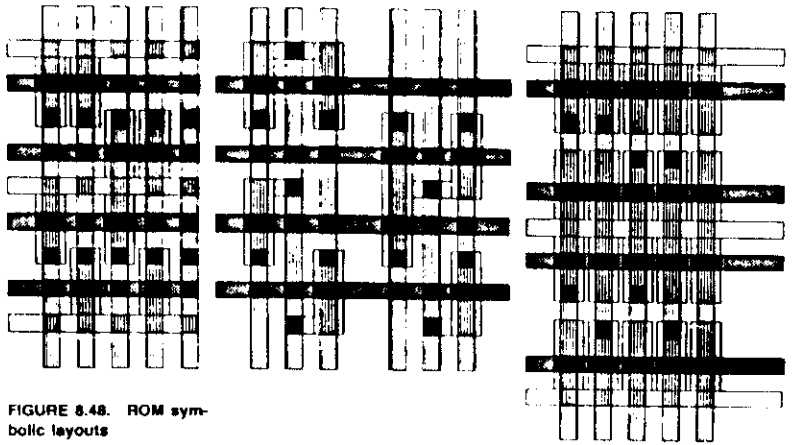


FIGURE 8.48. ROM symbolic layouts

ST

CT

New approaches for VLSI

- Non-silicon technologies (Ga As)
- Bipolar ECL (ultra fast technology)
- Analog VLSI
- Asynchronous design

Non-silicon technologies

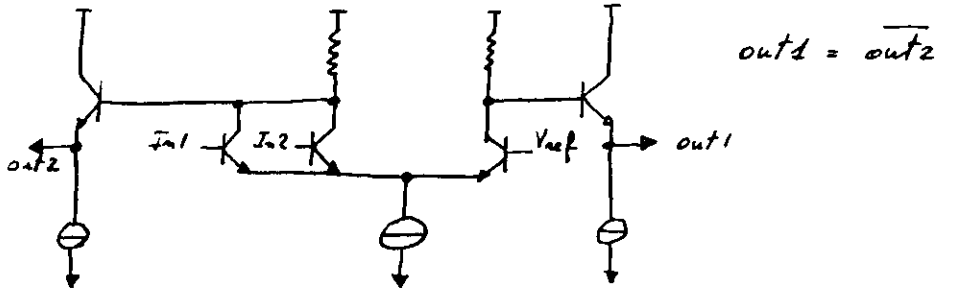
Probably the only one available today for developing real circuits, is Ga As.
Anyhow it is still immature for fabricating complex circuits or VLSI. Yield is still a problem and it seems to be very difficult to control transistors threshold over the whole chip.
It doesn't seem very likely that more than 20,000 gates can be implemented in a Ga As chip even in the next few years.
In addition to that only circuits with a few interconnections and small load can take advantage of the fastest technology.

Bipolar ECL

Even if it is a relatively old technology, only today

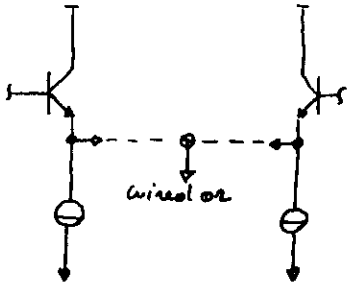
is getting popular. This is because only today bipolar ECL technologies allow millions of devices to be integrated.
It is a very fast technology, but the design is more difficult than with CMOS technology.
The logic swing is around 500 mV, so it is an order of magnitude less than CMOS logic swing.
Even if it is a less noisy technology than CMOS, because of the huge static power dissipation, variations in transistor's emitter current density and the use of some useful, but noisy configurations, the design is critical and requires specific expertise.

Basic gate:



Notice that a voltage reference is required and the output complement can be generated at the same time.

By using more voltage references and different kind of configurations, more complex gates can be designed



Notice that emitter-follower output stages can be connected together directely, to simulate an OR function.
The delay introduced by a wired-or connection is slightly higher than the delay of a single emitter follower stage, but far less than an OR gate made in the standard way.

Drive capability of an emitter follower is very good (the best if we compare it with CMOS or GaAs).

If we compare the best available ECL technology with the best available CMOS technology, ECL is about an order of magnitude faster, but twice less dense.

This technology is today available in all the possible forms: Gate Arrays
Standard Cells
Full-custom

Design tools are not as sophisticated as they are for CMOS.

Analog VLSI

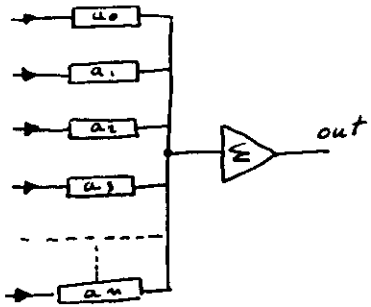
What analog VLSI means, is a new approach (actually old, but technology has made it feasible only today) such that computations are carried out in analog fashion instead of the more common digital way.
These systems look much more similar to living systems (human beings?) than the standard digital systems used to be.
Digital computers, for instance, are very effective at producing precise answers to well-defined questions.
The nervous system, on the contrary, accepts fuzzy, poorly conditioned input, perform a computation and produce approximate output. Nevertheless the visual system of a single human being does more image processing than do the entire world's supply of supercomputers.
The point is that there are problems where a different approach (analoge) seems to be much better.
Even if a neuron is not a very powerful basic block the organizing principles of a complex interconnection network of large numbers (billions) of them seem to be extremely powerful and effective at solving very complex problems.

Neural systems are not well understood today, but we shouldn't forget that this is a relatively new application.
What, in fact prevented us from creating interesting

applications was the lack of technology.
VLSI gives us a medium where it is possible to fabricate tens of millions of devices interconnected on a single silicon wafer.

In addition to that, wafer integration techniques can also be used and even larger networks can be built.

If our design is based on organizing principles, such that the failure of some components don't affect the performance of the whole system, testing can be simplified.



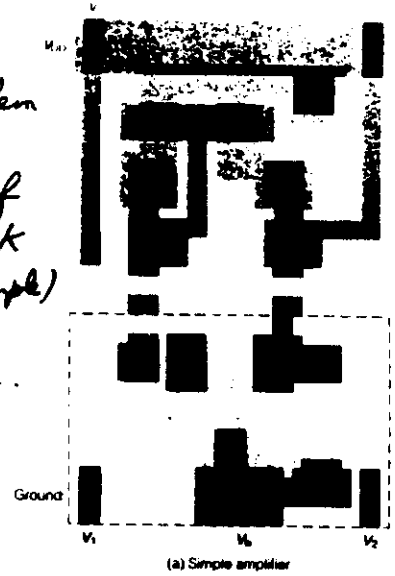
$$out = \begin{cases} +1 \text{ if } \sum_{j=1}^{n} a_j u_j \geq t_i \\ -1 \text{ if } \sum_{j=1}^{n} a_j u_j < t_i \end{cases}$$

t_i = threshold

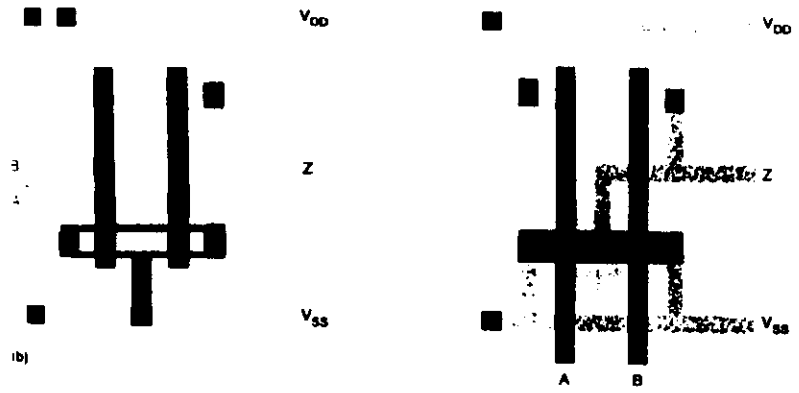
Neurons' behavior can be modeled as a threshold rule: the neuron will fire (output +1) if the weighted sum of inputs exceeds an internal threshold, otherwise it will not fire (output −1).
This is like a binary decision (2 output values); other models allow intermediate values, that can represent soft decisions.

The basic block for simulating a neuron is a simple transconductance amplifier.

If you look at the layout of one of them and you compare its side with the side of a basic digital block (NOR gates in this example) you won't find a very large difference.

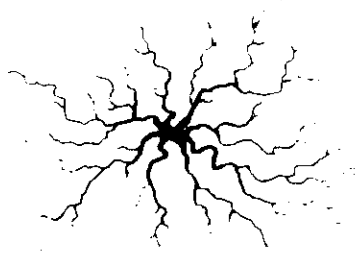


(a) Simple amplifier



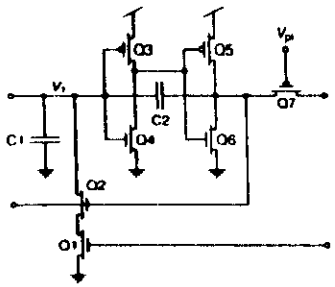
Standard static NOR gates

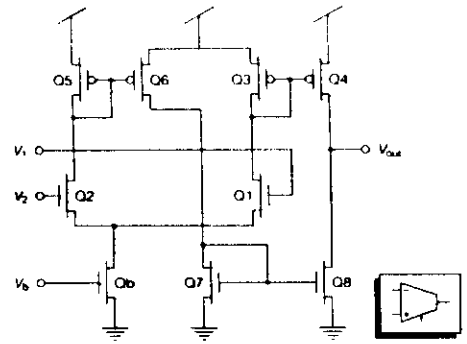
A real neuron with synapses and axons

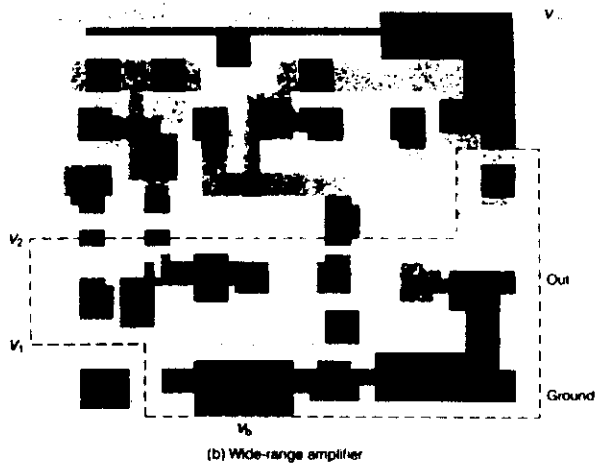


The large number of inputs makes the implementation of large neural networks a big problem.
It can be proved that with threshold function a larger number of inputs is better, in terms of area, than fewer inputs.

This simple circuit can simulate the function of an axon







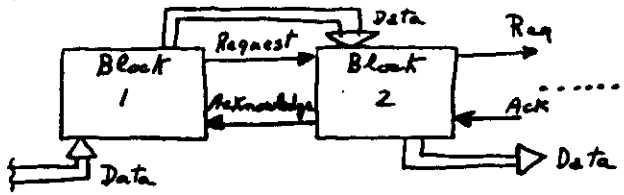
(b) Wide-range amplifier

Even for a fancier implementation of a transconductance amplifier the used silicon area is not too large.

Asynchronous design

This actually is a very old idea which didn't get developed because the difficulty of controlling subsystems delays. As a matter of fact, in asynchronous schemes there is no clock which controls the right sequence of operations, but the delays of the different logic elements impose the final sequence and order of computations.

If a communication protocol is defined between parts and each single part can be made delay insensitive (the local problem is by far easier to solve than the general problem), then we can build a self-timed system and we don't have to worry about delays any more.



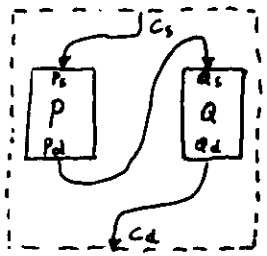
When Block 1 has completed its operations, sends a request to Block 2 and actually transfers data only when Block 2 is ready and asserts its acknowledge signal.

By using this simple communication protocol we don't only ensure the right exchange of information between blocks, but we can also build a family of composable parts.

If composition rules are simple and "well defined", it is very easy to build very complex and large system by

using parts from the set with the composition property.

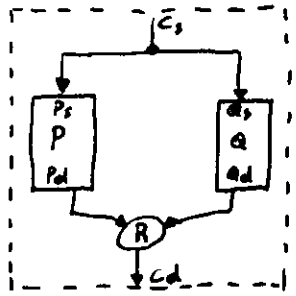
Ex 1:



s = start
d = done

sequential connection of parts

Ex 2:

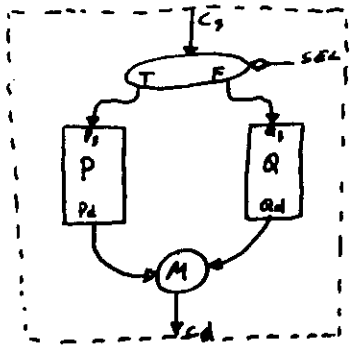


s = start
d = done
R = rendezvous module

concurrent connection of parts

The output of a "rendezvous module" occurs only after both of its inputs.

Ex 3:

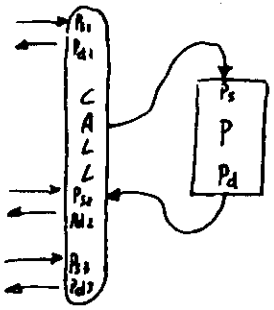


T = true F = false
sel = selecting value
M = merge module

IF sel THEN P ELSE Q
selector connection of parts

The output of a "merge module" occurs, as soon as either Pd or Qd signal occurs.

Ex 4:



Call connection of parts

If part P is started by way of P_{s2}, for instance, it will eventually send a done signal back on P_{d2}.

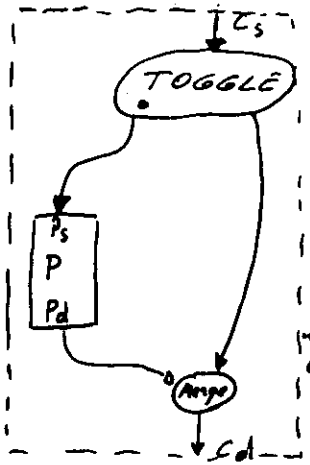
This implements in silicon the notion of "procedure" which is common in software.

Other connections (composable) such as, reversing order, until loop, arbiters etc. eta, can be easily designed. Notice that all these parts are implementations of the well-known software basic statements of a structured language.
A similar set of tools which is successfully used for software development can be used for hardware design.

A very nice property which a composable system built in this way, has, is that its speed is limited by only technology and different parts can use different technologies, as long as the communication procedure doesn't change.

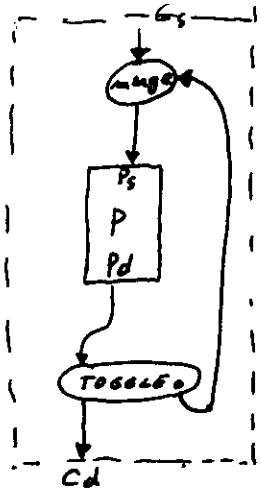
The reason why this kind of design has not been developed until now, is that only today by using VLSI technologies, it pays to integrated the additional part necessary for communications.

The second reason is that there still are some parts of the theory behind asynchronous design, not completely clear and more investigation must be done.
For communication purposes (send and done signals, event signalling (only changes are meaningful), better than logic levels transmission, is conveniently used.
A HI → LO, or LO → HI change is considered exactly the same thing.
This is also called "NON RETURN TO ZERO" or NRZ recording, or "two phase" signalling.
Another way to signal an event uses two changes instead of one.
It starts from LO and a LO → HI transition signals the event, and then a return to LO as soon as the first change has arrived.
In magnetic recording this is called "return to zero" signalling.
It is also called "four phase" signalling.
Two phase is faster, but uses more equipment.
Four phase it is often chosen, because it seems easier to understand. Anyway, conversion in both directions can be easily accomplished, by simple circuits like "connection for twice" or "connection for every other" (see next slide for the implementation)



A toggle receives a start signal and send an output along one of the two alternate outputs. The one marked with a dot is the first to be used.

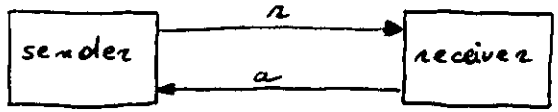
"Connection for every other"

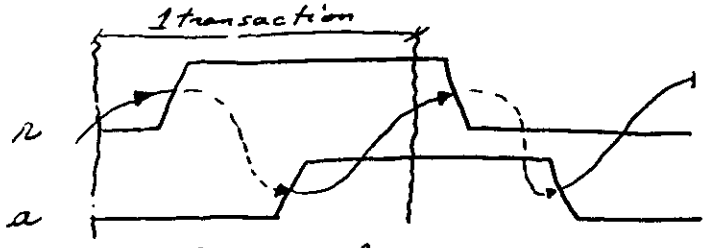


"Connection for twice"

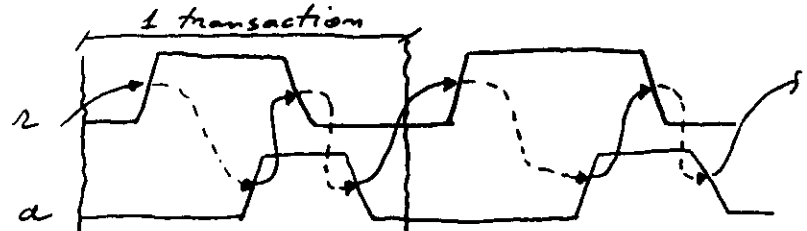
These two new parts can also be used as composable parts.

(ST)





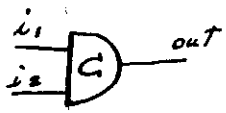
two-phase signaling



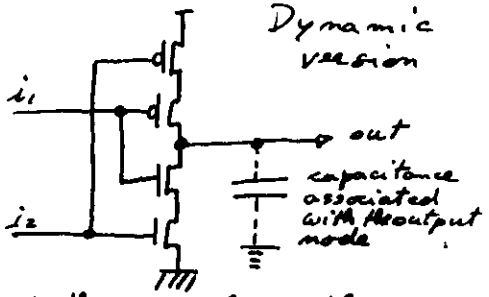
four-phase signaling

These signalling schemas may be implemented by using circuits which make use of Müller-C-element.

Müller-C-element



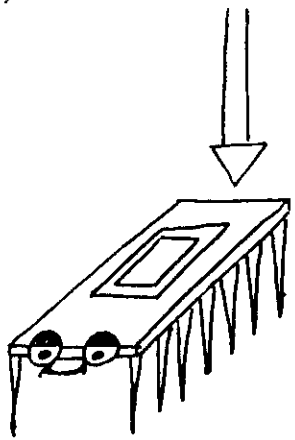
Dynamic version



The output changes only when both i_1, i_2 have the same transition.

(ST)

And now that you know all this about VLSI design, a little bit of experience, a lot of patience and....



shouldn't be the mysterious thing it used to be.

THE END

(ST)