Managment and Analysis of Physics Dataset – 1
A.A. 2019-20

Digital Circuits - 2

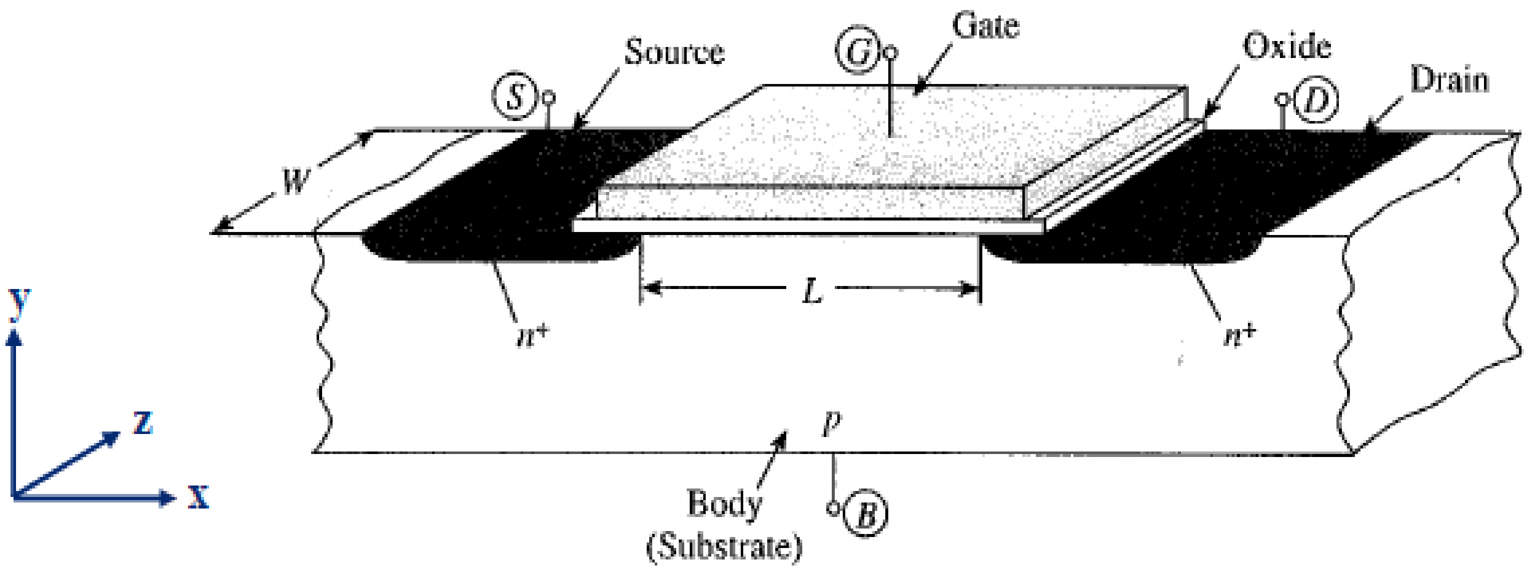
Overview

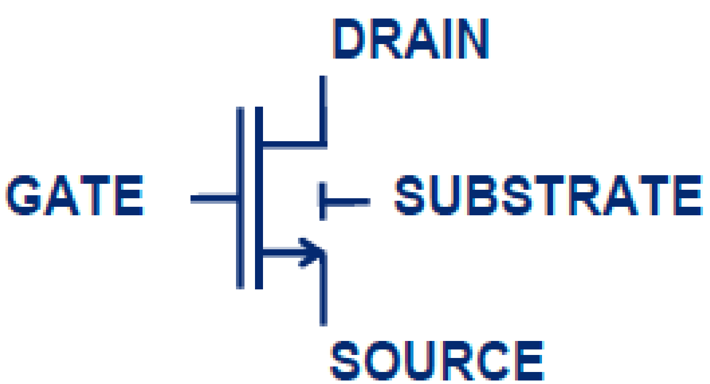
G.Collazuol

- Recap previous lecture
- Boolean algebra and Logic gates
- Representations of Digital Circuits
- Transistor Gates and Logic Families
- Timing in Digital Circuits
- Digital Circuits properties:
  - → Asynchronous vs Synchronous
  - → Combinational vs Sequential

!!! Lectures time table
NEXT WEEK

Lecture #2 – Recap – MOSFET transistor





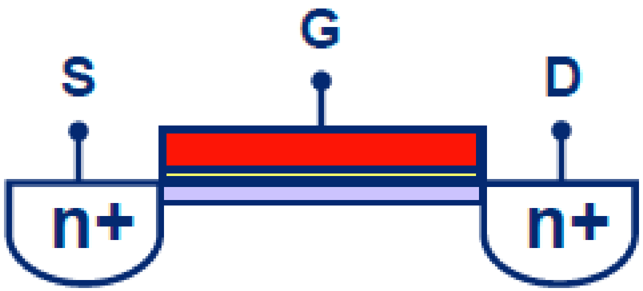
What is a MOS transistor?

Analog circuits: amplifier (V to I)

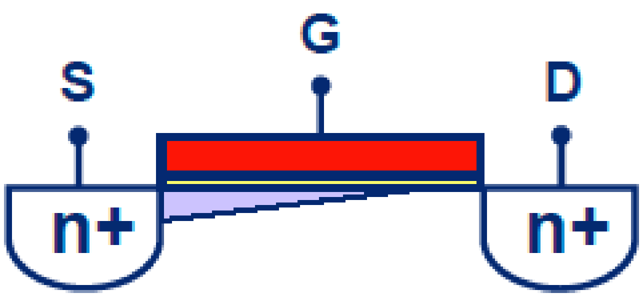
Digital circuits: switch

Y. Tsividis, *Operation and Modeling of The MOS Transistor*, 2nd edition, McGraw-Hill, 1999.

Lecture #2 – Recap – MOSFET working regions



LINEAR REGION (Low V_{DS}):
Electrons are attracted to the SiO_2 – Si interface. A conductive channel is created between source and drain. We have a Voltage Controlled Resistor (VCR).



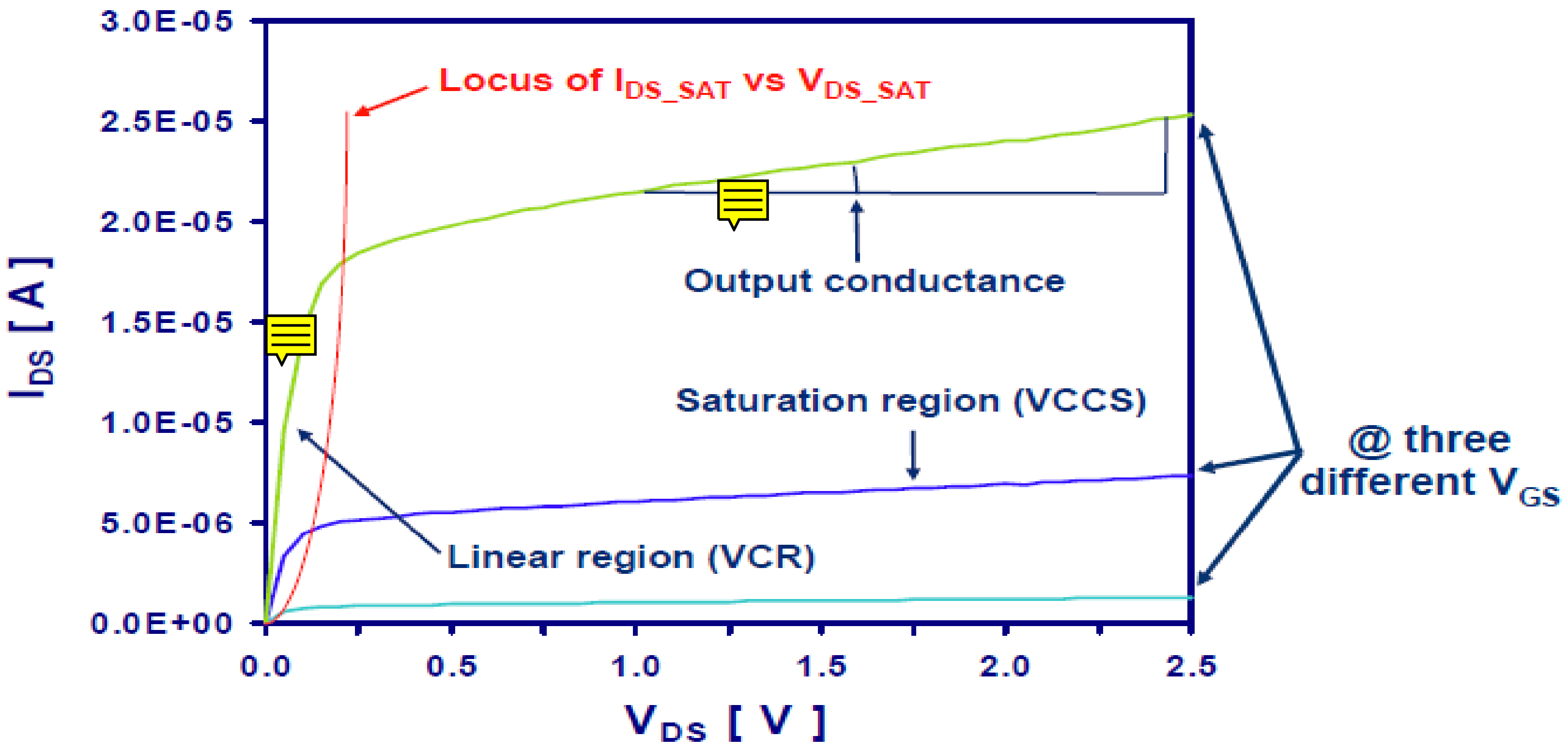
SATURATION REGION (High V_{DS}):
When the drain voltage is high enough the electrons near the drain are insufficiently attracted by the gate, and the channel is pinched off. We have a Voltage Controlled Current Source (VCCS).

Lecture #2 – Recap – MOSFET working regions

SATURATION REGION:

$$V_{DS} > \frac{V_{GS} - V_T}{n} = V_{DS\_SAT}$$

$$I_{DS} = \frac{\beta}{2n}(V_{GS} - V_T)^2$$

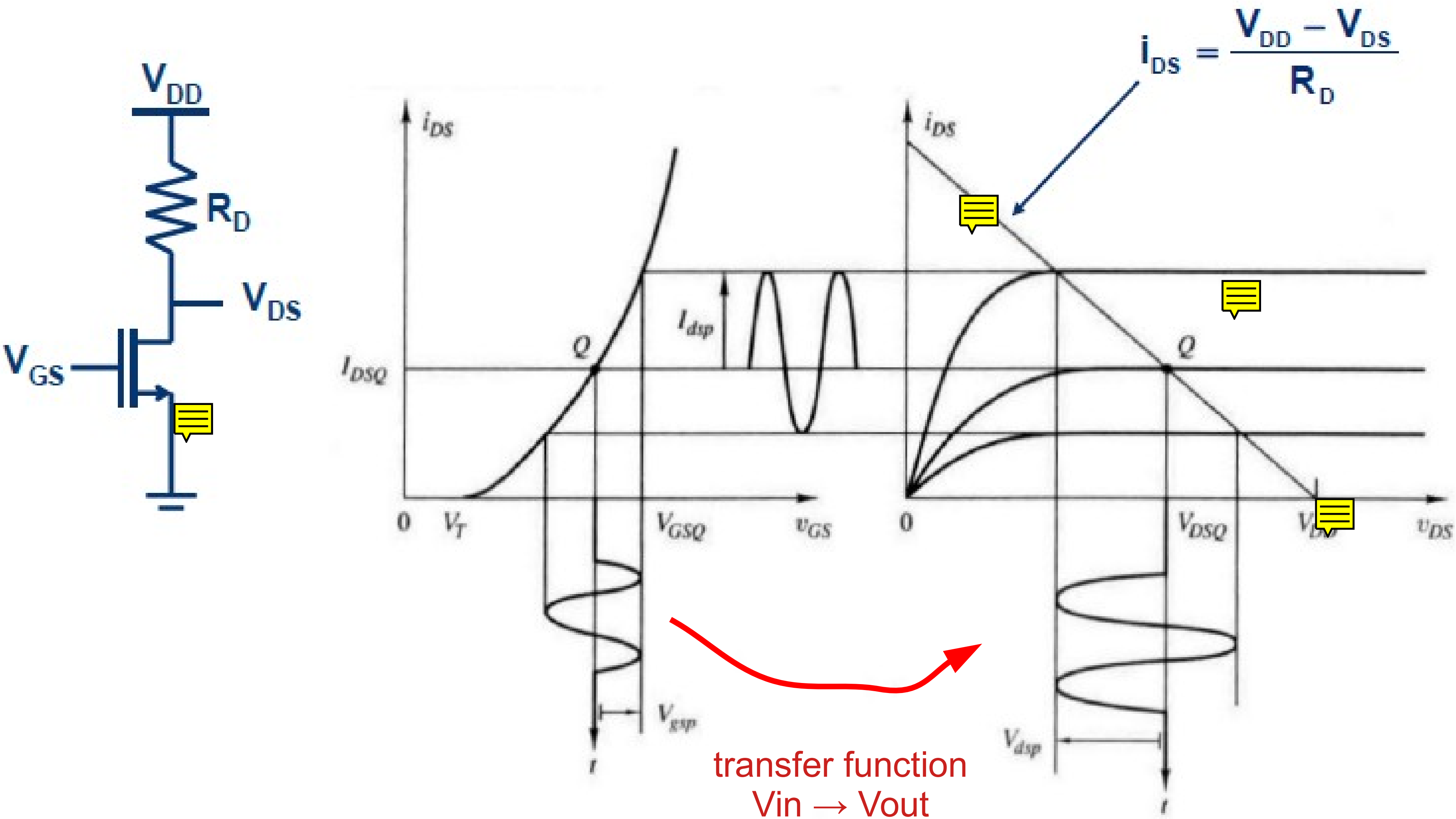


LINEAR REGION:

$$V_{DS} < \frac{V_{GS} - V_T}{n} = V_{DS\_SAT}$$

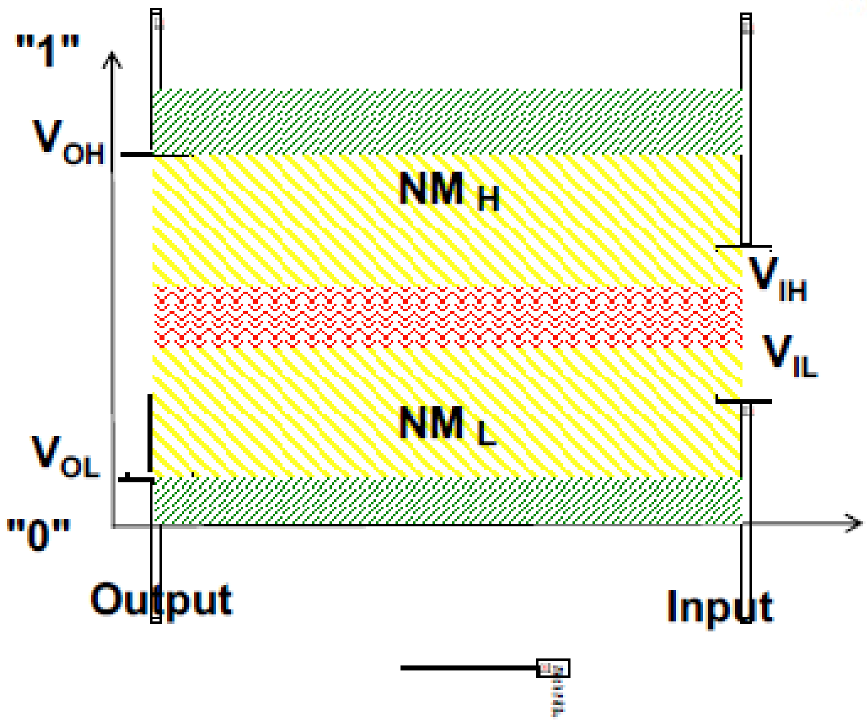
$$I_{DS} = \beta\,(V_{GS} - V_T - \frac{nV_{DS}}{2})V_{DS}$$

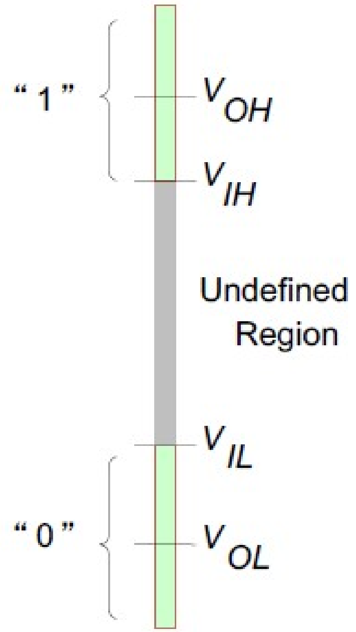
Lecture #2 – Recap – MOSFET with a load R_D

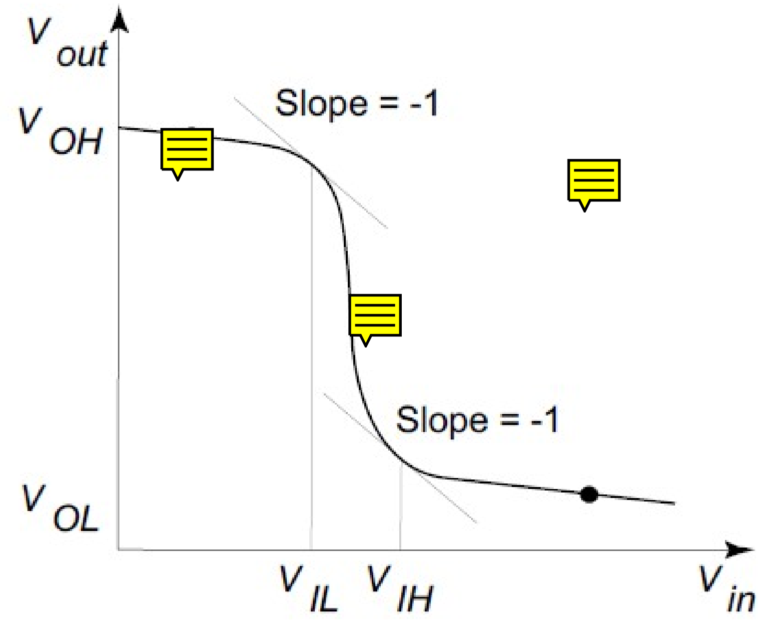


K. R. Laker and W. M. C. Sansen, *Design of Analog Integrated Circuits and Systems*, McGraw-Hill, 1994.

Lecture #2 – Recap – transfer function and logic levels





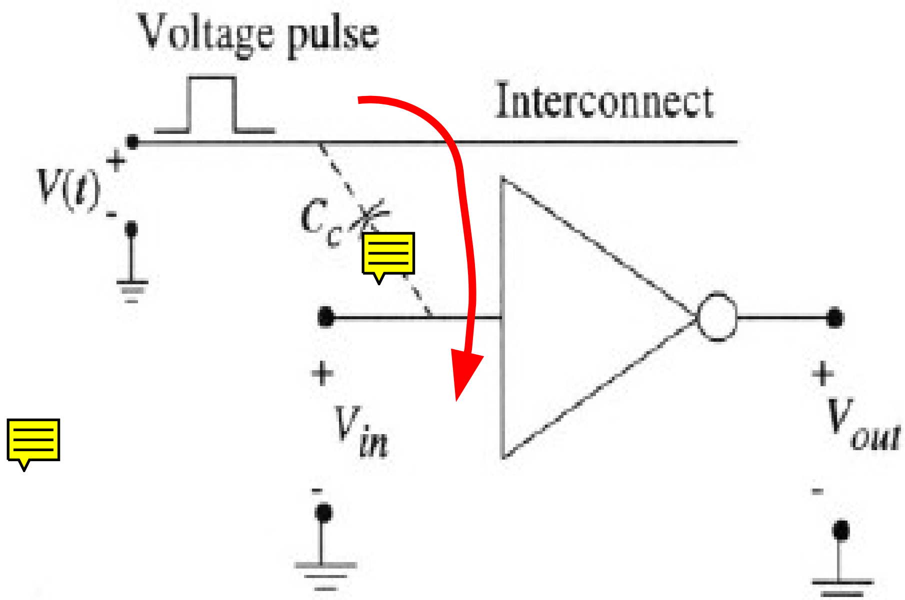


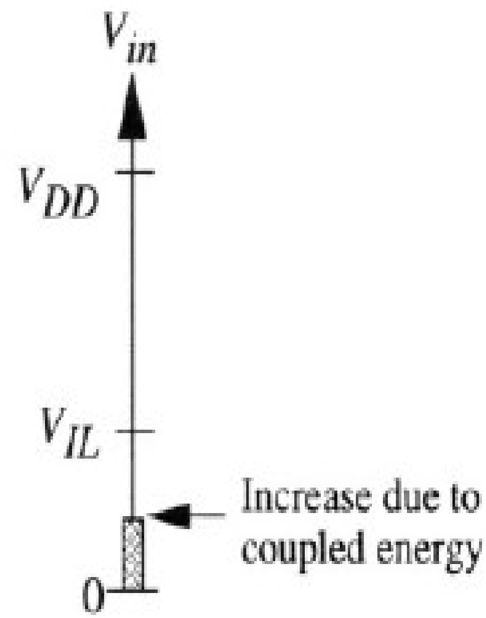
Noise margin high

Noise margin low

Fundamental property #1 → Noise margin

Lecture #2 – Recap – transfer function and logic levels



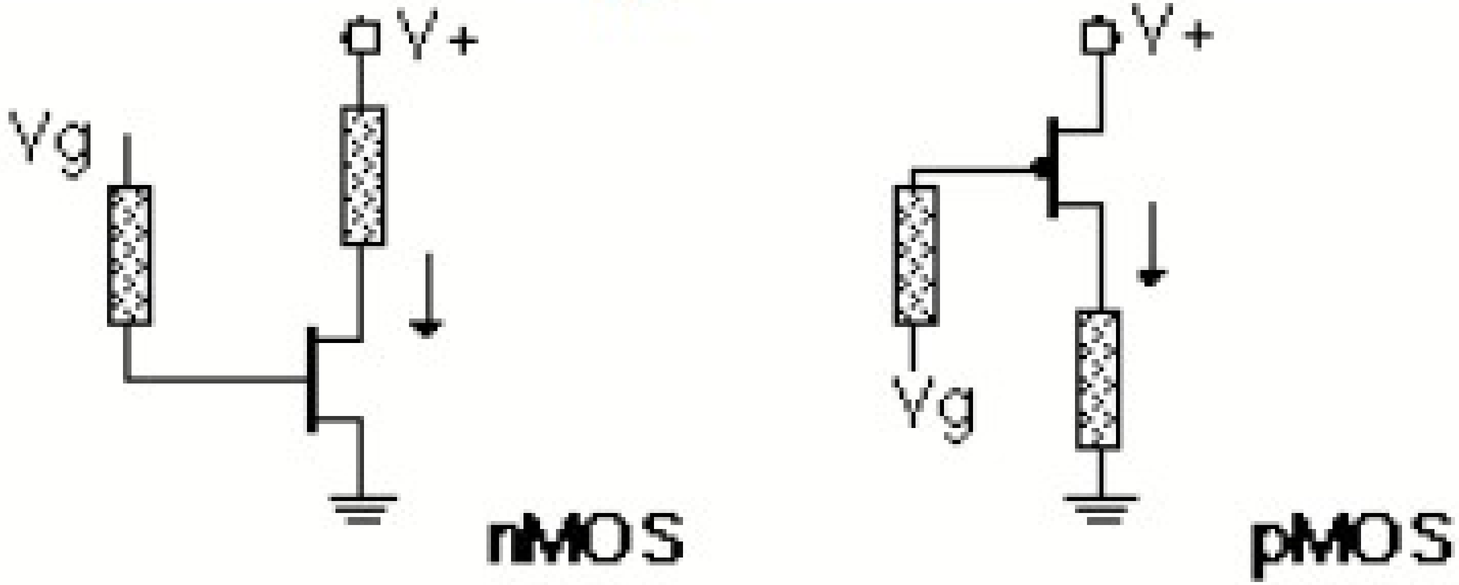


Vin input level perturbation
due to interefecence from other
regiosn of the digital circuit
(via parasitic capacitive coupling)

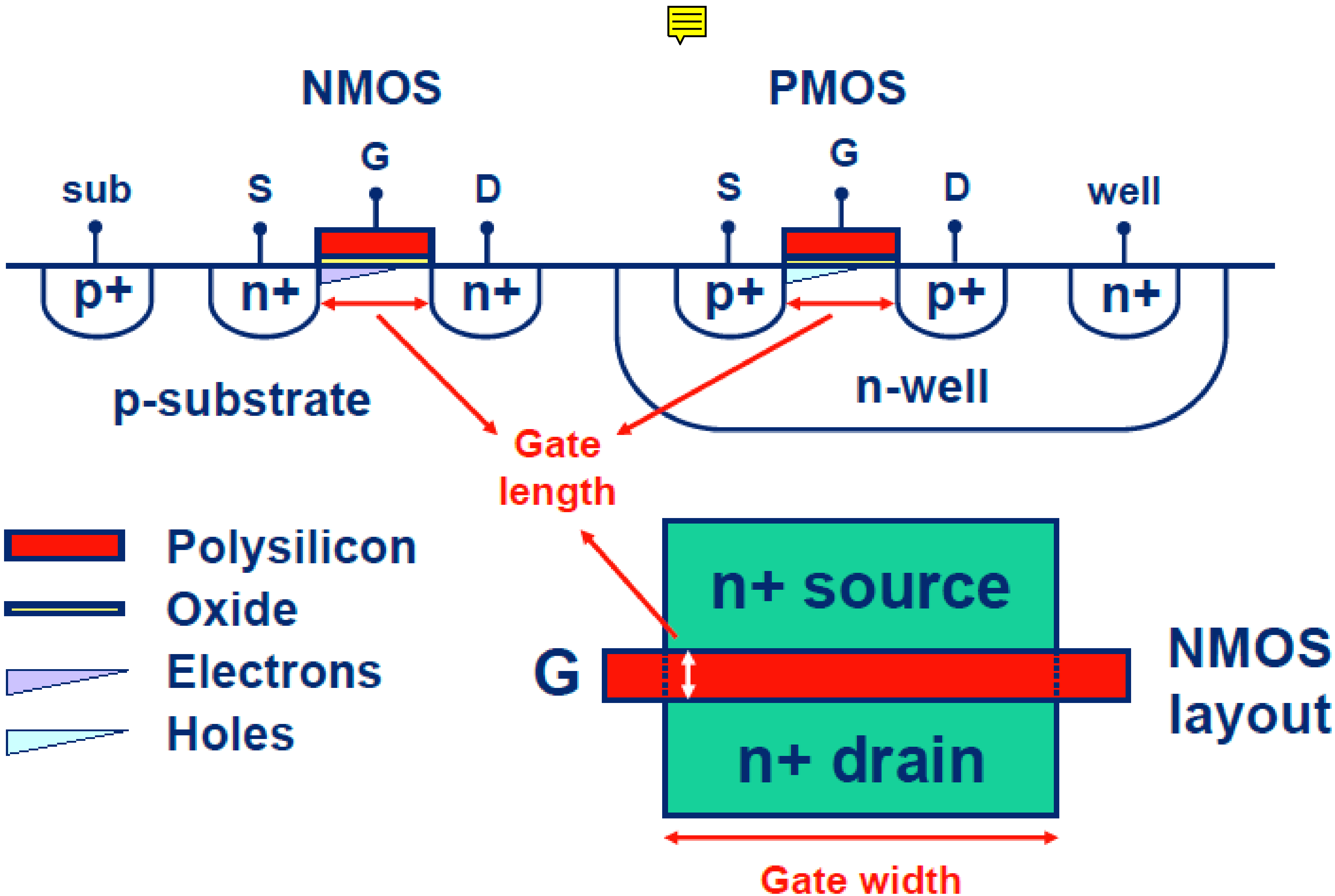
Fundamental property #1 → Noise margin

Lecture #2 – Recap – MOSFET types

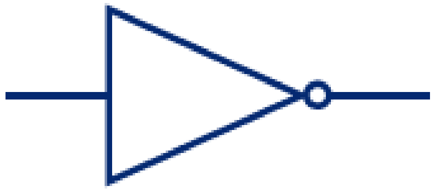
Figura 4.1.4



Lecture #2 – Recap – CMOS tecnology



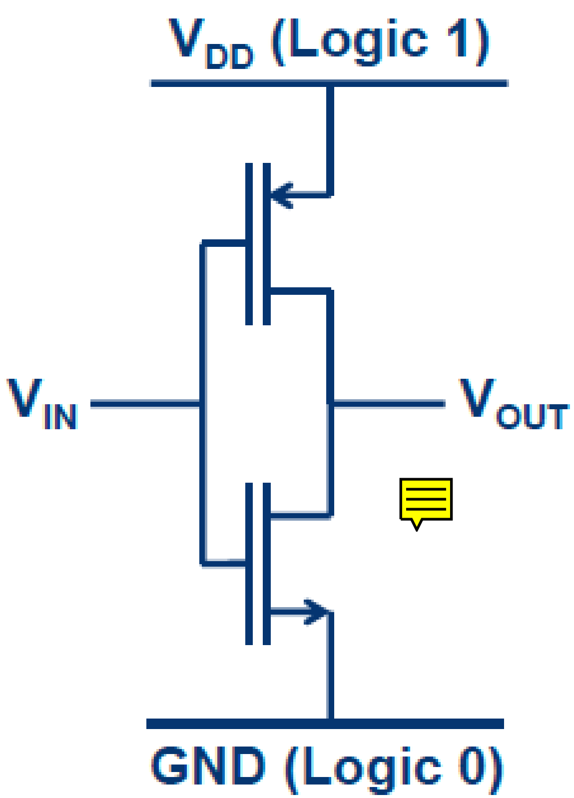
Lecture #2 – Recap – why CMOS ?

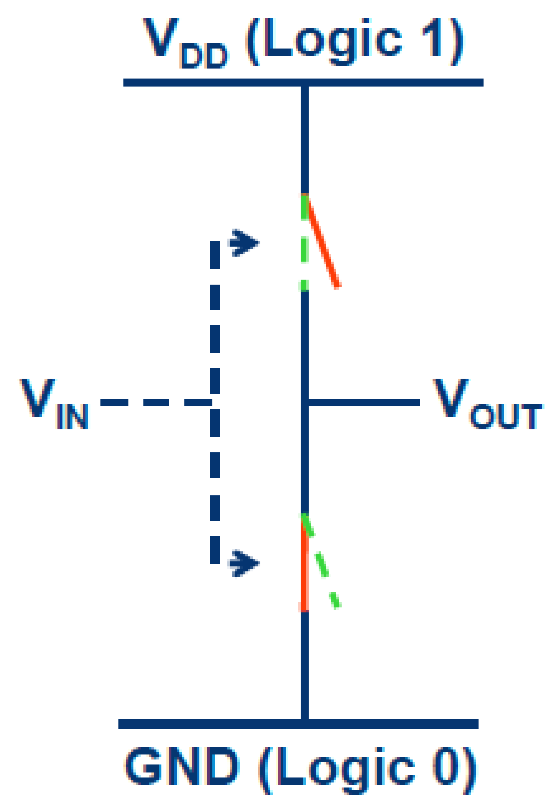


THE INVERTER

Truth table

IN	OUT
0	1
1	0

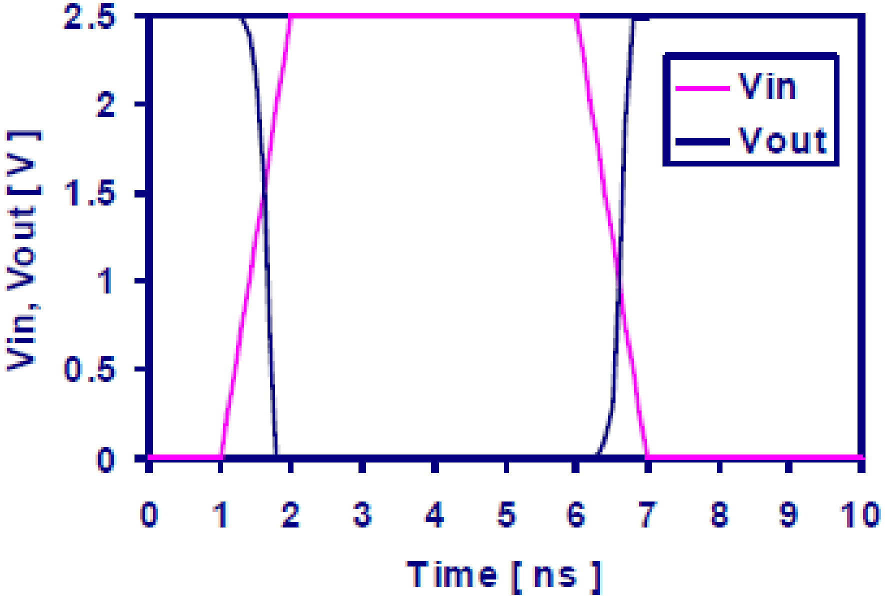


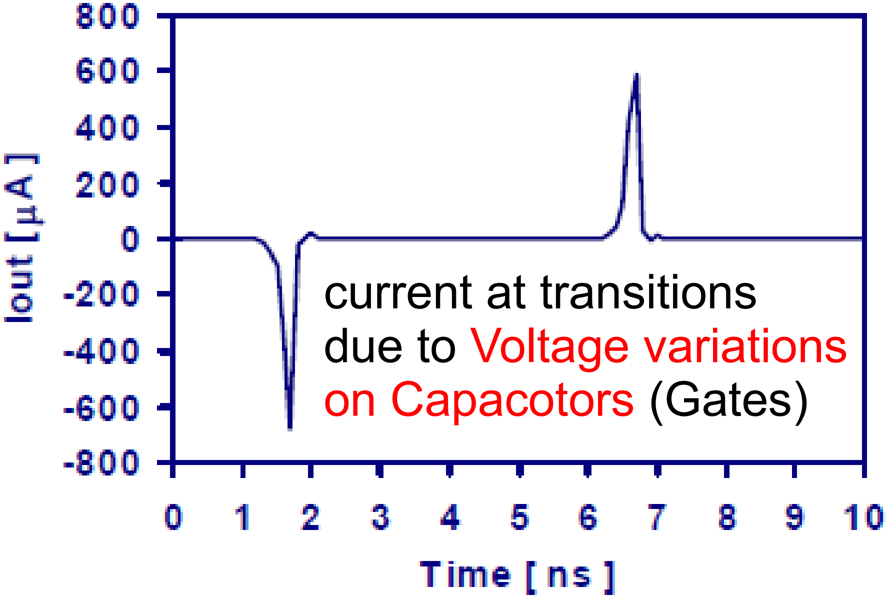


Negligible current is drawn from Power Supplies …

Lecture #2 – Recap – why CMOS ?

Simulation of a chain of two inverters in a 0.25 μm CMOS technology

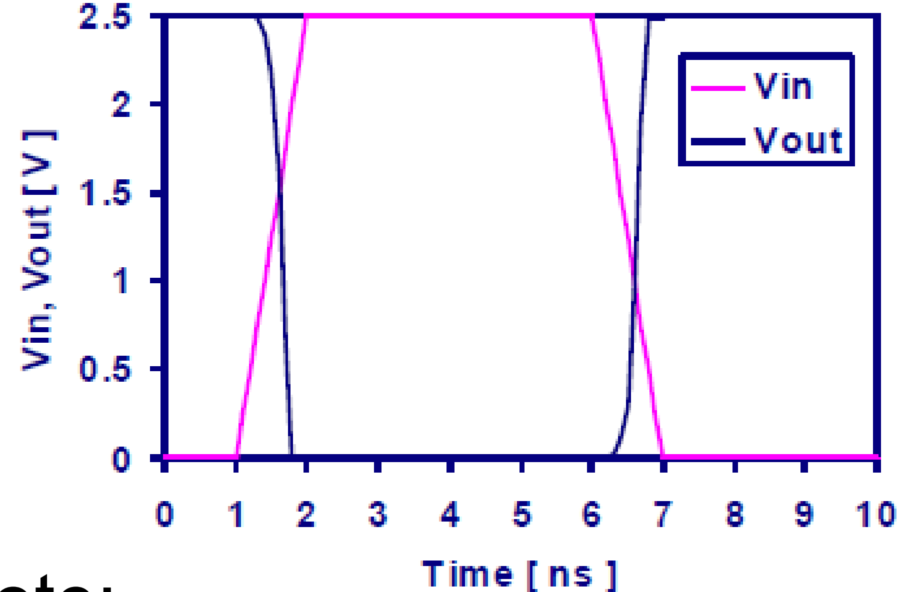


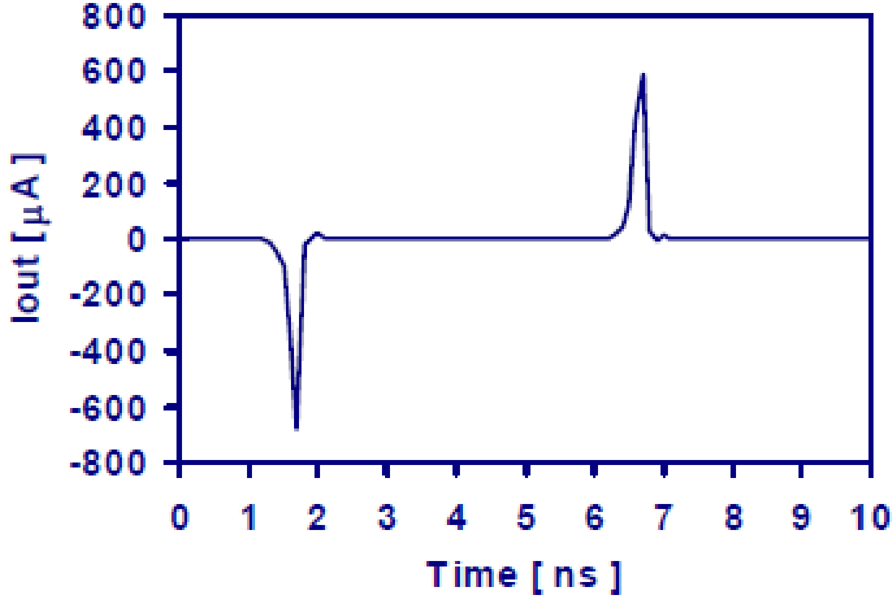


Negligible current is drawn from Power Supplies …
… except during transitions (very fast → current spikes)

Lecture #2 – Recap – why CMOS ?

Simulation of a chain of two inverters in a 0.25 μm CMOS technology



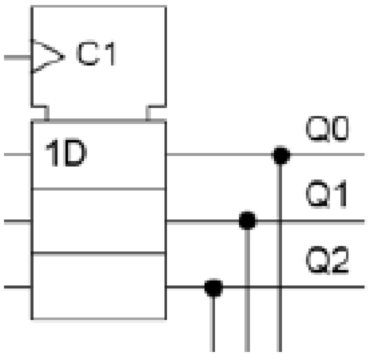


Note:
(1) very fast transition but anyway some delay is there
(2) low_Z output and high_Z input = only 2 levels involved

Lecture #2 – Logic levels propagation:
Fundamental property #2 → Fanout

Digital ports must allow fanout (multiple copies of output signals) !
→ this is one of tha main reasons for Digital Electronics works with **voltage** signals (instead of current)

→ input: high Z
→ output: low Z



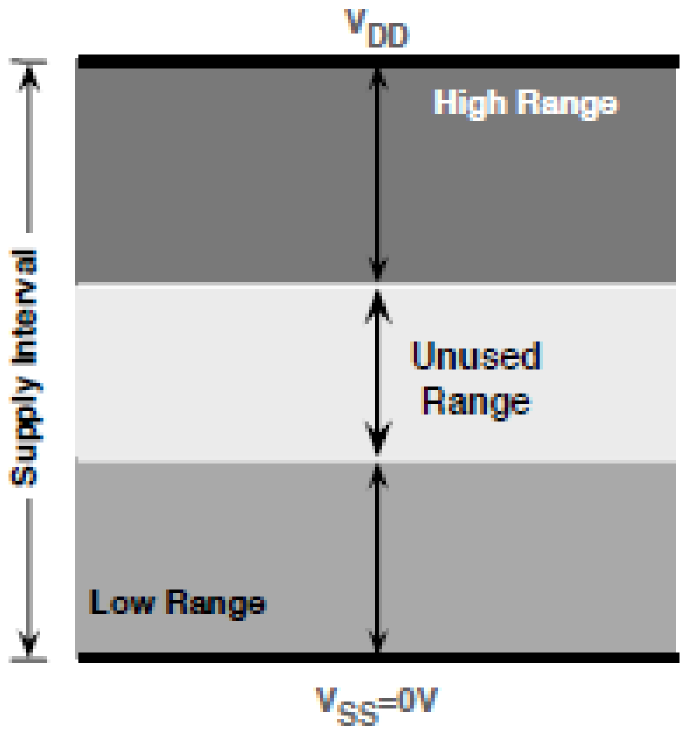
Note: high Z output is used for circuit control (see tristate)

Logic Levels

Analog signals → continuous values within a certain range...
… but when signals take on only discrete values (integers) → digital signals

Electronic devices with two well-defined states arise much more naturally than devices with ten states: binary representation

Digital circuits mandatory for
- logic and arithmetic functions in computers
- transmission of the digitized signals over noisy channels is often perfect



The big advantage of digital electronics is that there are relatively wide margins in the definition of logic levels. Even in the presence of noise that dynamically alters the signal value the digital information does not change.

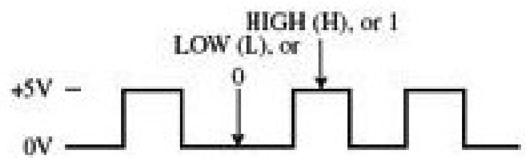
Boolean Algebra and Logic Gates

Binary information

Analog Signal

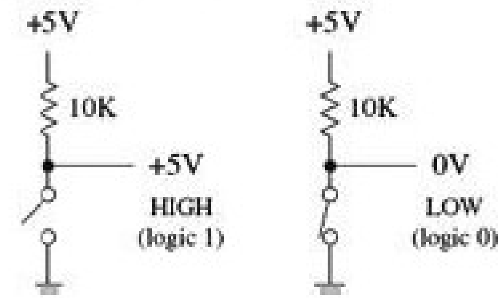
continuous voltage waveform

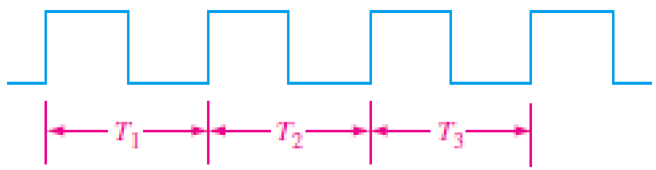
Digital Signal



discrete voltage levels

Using a switch to demonstrate logic states

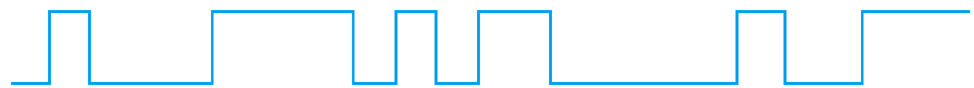




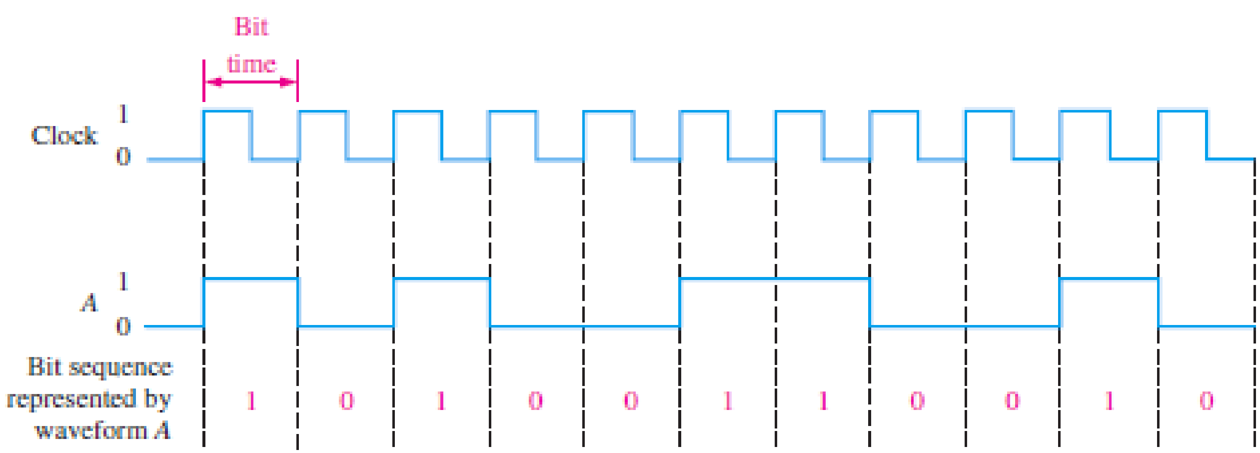
Period $= T_1 = T_2 = T_3 = \ldots = T_n$

Frequency $= \frac{1}{T}$

(a) Periodic (square wave)



(b) Nonperiodic



Integer representations

Binary-to-Decimal Conversion | **Decimal-to-Binary Conversion**

109_{10} to binary

109/2 = 54 w/ remainder 1 (LSB)
54/2 = 27 w/ remainder 0
27/2 = 13 w/ remainder 1
13/2 = 6 w/ remainder 1
6/2 = 3 w/ remainder 0
3/2 = 1 w/ remainder 1
1/2 = 0 w/remainder 1 (MSB)

Answer: 1101101
8-bit answer: 01101101

Take decimal number and keep dividing by 2, while keeping the remainders. The first remainder becomes the LSB, while the last one becomes the MSB.

10100100 to decimal

2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0
(MSB) 1 0 1 0 0 1 0 0 (LSB)

$0 \times 2^0 = 0$
$0 \times 2^1 = 0$
$1 \times 2^2 = 4$
$0 \times 2^3 = 0$
$0 \times 2^4 = 0$
$1 \times 2^5 = 32$
$0 \times 2^6 = 0$
$1 \times 2^7 = 128$

Expand the binary number as shown and add up the terms. The result will be in decimal form.

Answer: 164_{10}

Octal to Binary | **Binary to Octal**

537_8 to binary

5 3 7
101 011 111

Answer: 101011111_2

$111\ 001\ 100_2$ to octal

111 001 100
7 1 4

Answer: 714_8

Hex to Binary | **Binary to Hex**

$3E9_{16}$ to binary

3 E 9
0011 1110 1001

Answer: $0011\ 1110\ 1001_2$

$1001\ 1111\ 1010\ 0111_2$ to octal

1001 1111 1010 0111
9 F A 7

Answer: $9FA7_{16}$

A 3-digit binary number is replaced for each octal digit, and vice versa. The 3-digit terms are then grouped (or octal terms are grouped).

A 4-digit binary number is replaced for each hex digit, and vice versa. The 4-digit terms are then grouped (or hex terms are grouped).

Negative integers → 2's complement

In 2's complement representation
a negative number is represented by
a binary which results in zero when
added to its corresponding positive

Decimal to 2's complement

$+4_{10}$ to 2's complement

true binary = 0010 1001
2's comp = 0010 1001

-41_{10} to 2's complement

true binary = 0010 1001
1's comp = 1101 0110
Add 1 = +1
2's comp = 1101 0111

DECIMAL	SIGN-MAGNITUDE	2'S COMPLEMENT
+7	0000 0111	0000 0111
+6	0000 0110	0000 0110
+5	0000 0101	0000 0101
+4	0000 0100	0000 0100
+3	0000 0011	0000 0011
+2	0000 0010	0000 0010
+1	0000 0001	0000 0001
0	0000 0000	0000 0000
–1	1000 0001	1111 1111
–2	1000 0010	1111 1110
–3	1000 0011	1111 1101
–4	1000 0100	1111 1100
–5	1000 0101	1111 1011
–6	1000 0110	1111 1010
–7	1000 0111	1111 1001
–8	1000 1000	1111 1000

In 2's complement representation the procedures for adding and subtracting positive and negative numbers are the same

Binary Arithmetic: addition / subtraction

Adding

$$\begin{array}{rcl} & & \begin{array}{cccc} 1 & 1 & 1 & \end{array} \\ & 5_{10} = & \begin{array}{cccc} 0 & 1 & 0 & 1 \end{array} \\ + & 3_{10} = & \begin{array}{cccc} 0 & 0 & 1 & 1 \end{array} \\ \hline & & \begin{array}{cccc} 1 & 0 & 0 & 0 \end{array} \end{array}$$

$$\begin{array}{rl} & 20_{10} = 00010100 \\ + & 87_{10} = 01010111 \\ \hline & \phantom{87_{10} =} 01101011 \end{array}$$

Subtracting

$$\begin{array}{r} 4_{10} \\ -\ 1_{10} \\ \hline 3_{10} \end{array} \qquad \begin{array}{rcccc} & & & 1 & \\ & & 0 & \cancel{1}0 \rightarrow & 10 \\ & 0 & \cancel{1} & \cancel{0} & \cancel{0} \\ - & 0 & 0 & 0 & 1 \\ \hline & 0 & 0 & 1 & 1 \end{array}$$

$$\begin{array}{ll} +19_{10} & = 00010011 \\ -7_{10} & = 11111001 \\ \hline \text{Sum} & = 00001100 \end{array}$$

the long way

2's complement way

Binary Arithmetic: multiplication / division (by 2)

Multiplicating

Multiplication by 2 of a nonnegative integer less than 2^{N-2} is obtained by shifting all bits one position to the left (throwing away the MSB) and writing a 0 into the least significant bit

eg $3 \cdot 2$ is left shift $0011_2 = 0110_2 = 6_{10}$

$-3 \cdot 2$ is $1101_2 = 1010_2 = -6_{10}$

The overflow can be detected because the result of the shift is negative

eg $7 \cdot 2$ is 0111_2 results in $1110_2 = -2_{10}$!!! overflow

$-5 \cdot 2$ is 1011_2 results in $0110_2 = 6_{10}$

Dividing

Division by 2 of a non-negative integer is carried out by throwing away the least significant bit (or perhaps storing it elsewhere as the remainder), shifting the remaining bits one position to the right and writing a 0 into the most significant bit

eg shifting $0111_2 = 7_{10}$ to the right, for example, results in $0011_2 = 3_{10}$

Note: Shifts and Sums → easily carried by digital circuits …

Binary Algebra operations and variables

OR .... $0+0=0$ $0+1=1$ $1+0=1$ $1+1=1$ .... union

AND ...... $0\cdot 0=0$ $0\cdot 1=0$ $1\cdot 0=0$ $1\cdot 1=1$ .... intersection

NOT .... $\overline{0}=1$ $\overline{1}=0$ .... complement

Boolean algebra → variables

$$\overline{\overline{A}}=A \quad A+A=A \quad A+0=A \quad A+1=1 \quad A+\overline{A}=1$$

$$A\cdot A=A \quad A\cdot 0=0 \quad A\cdot 1=A \quad A\cdot\overline{A}=0$$

• addition and multiplication are commutative and associative:

$$A+B=B+A \qquad A+(B+C)=(A+B)+C$$

$$A\cdot B=B\cdot A \qquad A\cdot(B\cdot C)=(A\cdot B)\cdot C$$

• multiplication is distributive over addition:
• … and viceversa !

$$A\cdot(B+C)=(A\cdot B)+(A\cdot C)$$

$$A+(B\cdot C)=(A+B)\cdot(A+C)$$

• De Morgan theorems $\overline{A+B}=\overline{A}\,\overline{B}$ $\overline{AB}=\overline{A}+\overline{B}$

Central theorems in Digital electronics

Note: → AND can be implemented by using just OR e NOT
→ OR can be implemented by using just AND e NOT

Binary Algebra logical functions and conditions

If A, B, and C are logical variables $\Phi = A\overline{C} + BC + AB$ is a logical function

A logical function can be described by a truth table that lists all possible arrays of values of the logical variables together with the corresponding values of the function

array of values of the variable pair A,B

A	B	0	$A\cdot B$	$A\cdot\overline{B}$	A	$\overline{A}\cdot B$	B	$A\oplus B$	$A+B$	$\overline{A+B}$	$\overline{A\oplus B}$	$\overline{B}$	$A+\overline{B}$	$\overline{A}$	$\overline{A}+B$	$\overline{A\cdot B}$	1
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

XOR NOR XNOR NAND

Logical conditions

Can detect the condition [A>B] by interpreting A and B as logical variables, executing the logical function $A\underline{B}$ and demanding that the result be 1

If we continue this process, we obtain the following list of correspondences between bit comparisons and logical functions:

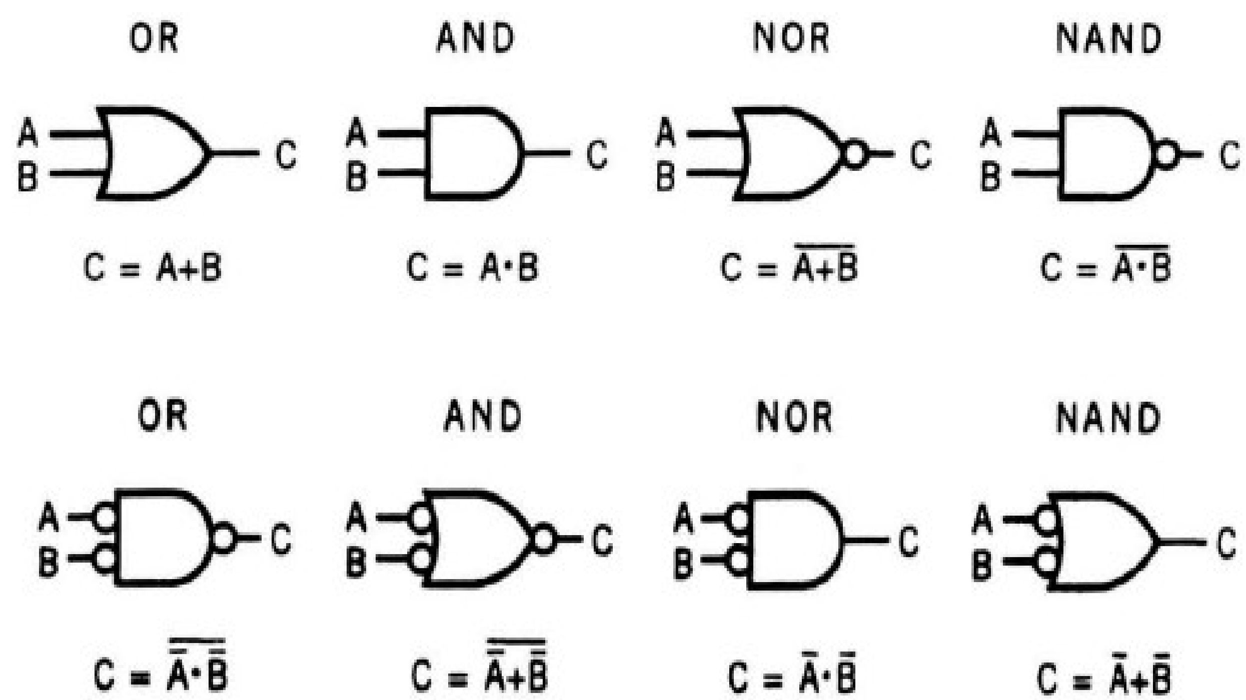
$$[A>B] \leftrightarrow A\cdot\overline{B} \qquad [A\le B] \leftrightarrow \overline{A}+B$$

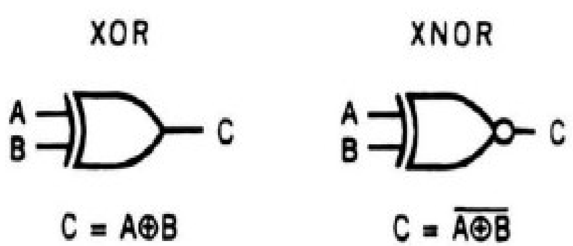
$$[A<B] \leftrightarrow \overline{A}\cdot B \qquad [A\ge B] \leftrightarrow A+\overline{B}$$

$$[A=B] \leftrightarrow \overline{A\oplus B} \qquad [A\ne B] \leftrightarrow A\oplus B$$

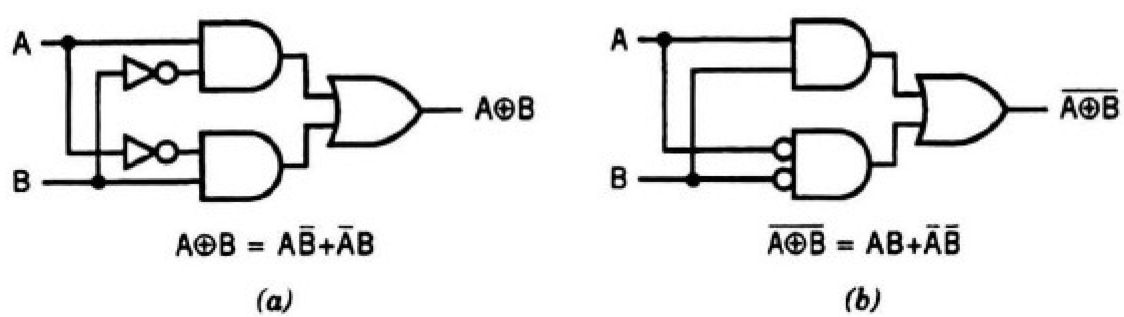
Logic Gates

Equivalent
representation
by using
De Morgan's
theorems





XOR and XNOR
are used very often



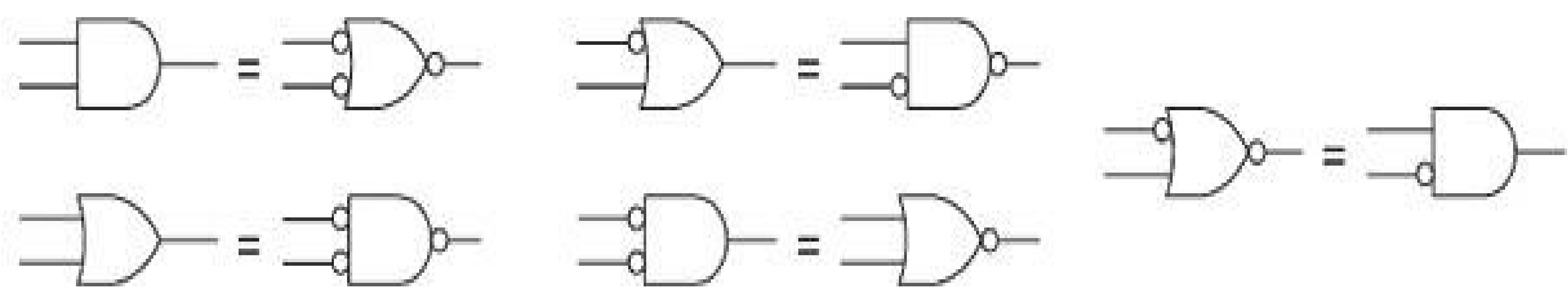
logic circuits for XOR and XNOR

Note: bubble pushing

A shortcut method for forming equivalent logic circuits, based on De Morgan's theorem, is to use what's called *bubble pushing*.

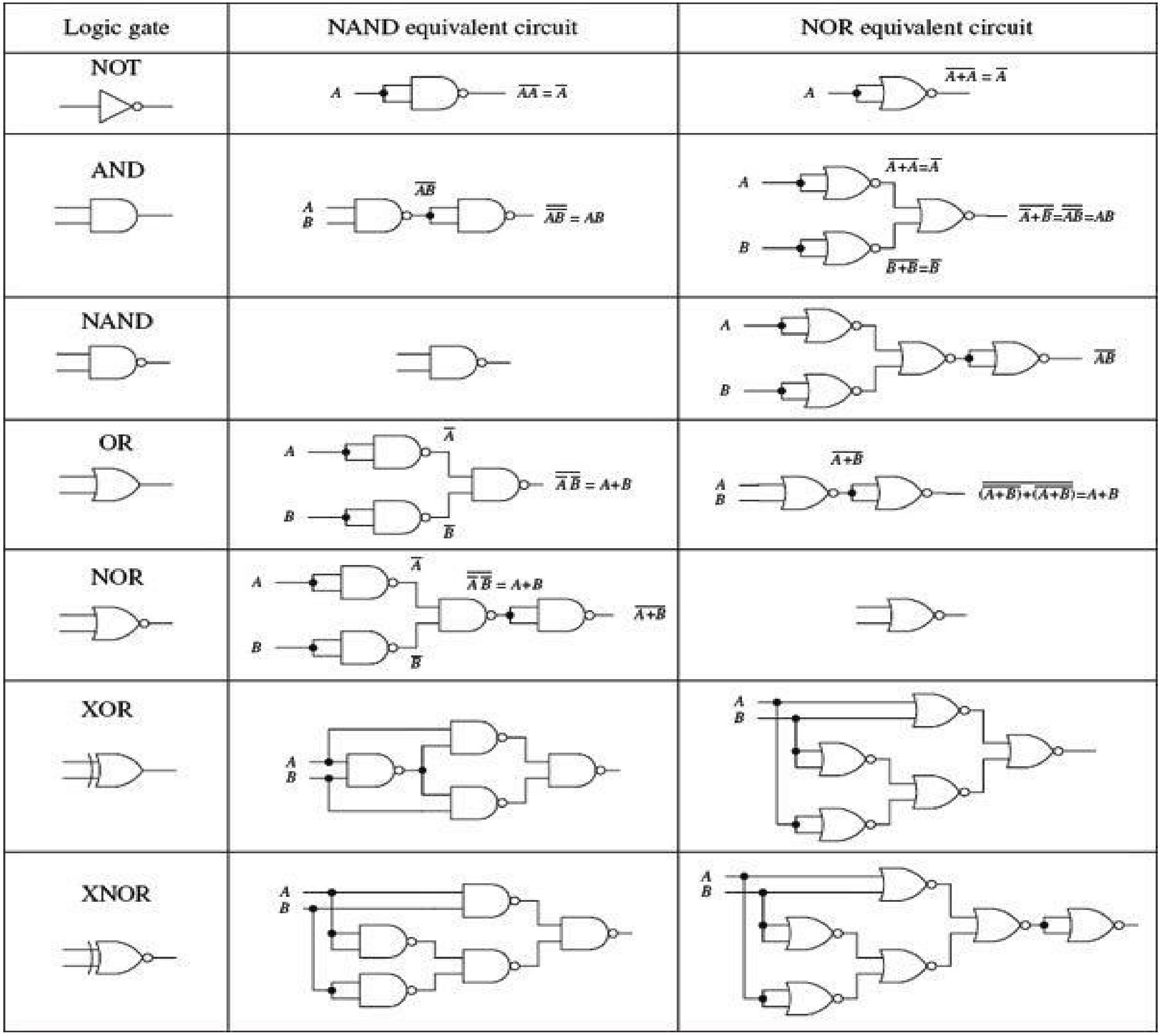
Bubble pushing involves the following tricks:

- Change an AND gate to an OR gate or change an OR gate to an AND gate.
- Add inversion bubbles to the inputs and outputs where there were none, while removing the original bubbles.



Universal capability of NAND and NOR gates

NAND and NOR gates are referred to as universal gates because each one can be combined with itself to form all other possible logic gates



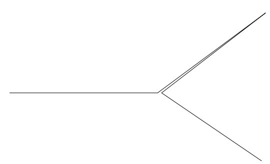
Logic functions with n-inputs to m-outputs

ie n single bit independent variables and m-bits dependent functions

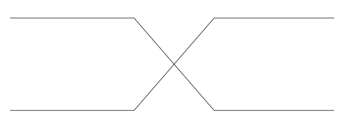
We have seen n-to-1 bits logic functions … but of course n-to-m bits must be consedered

→ need to introduce just 2 new operators in addition

1) FANOUT



2) EXCHANGE



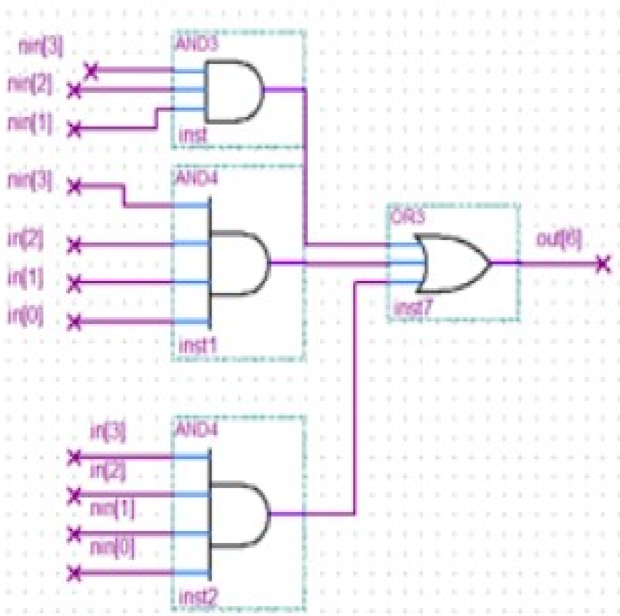
Note: indeed EXCHANGE is enough (for you can short-circuit inputs to get FANOUT)

Digital circuits =

hardware implementation of
logical and arithmetical functions

Representations of Digital Circuits

Schematic diagram & gates



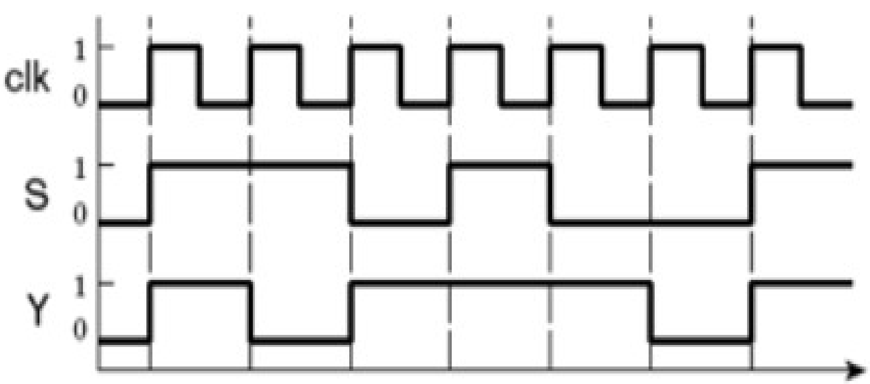
Boolean equation

out6 = /in3*/in2*/in1 + in3*in2*/in1*/in0 + /in3*in2*in1*in0

Truth table

in[3..0]	out[6:0]
0000	1000000
0001	1111001
0010	0100100
0011	0110000
0100	0011001
0101	0010010
0110	0000010
0111	1111000
1000	0000000
1001	0010000

Timing Diagram



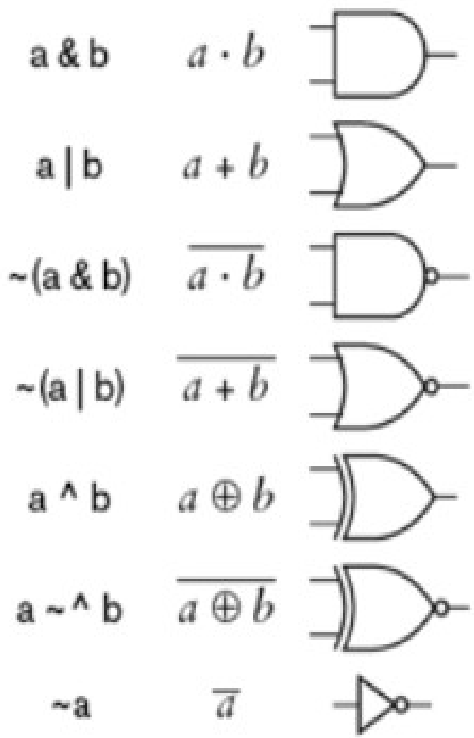
Hardware Description Language (HDL)

3-to-1 MUX

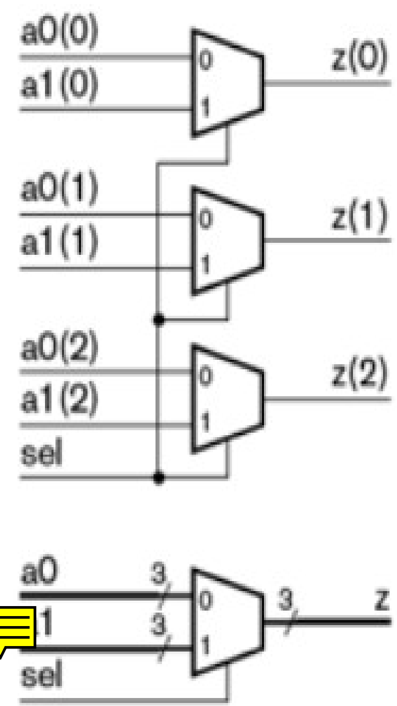
```
module mux32three(i0,i1,i2,sel,out);
input [31:0] i0,i1,i2;
input [1:0] sel;
output [31:0] out;
reg [31:0] out;

always @ (i0 or i1 or i2 or sel)
begin
  case (sel)
    2'b00: out = i0;
    2'b01: out = i1;
    2'b10: out = i2;
    default: out = 32'bx;
  endcase
end
endmodule
```

Basic Digital Building Blocks

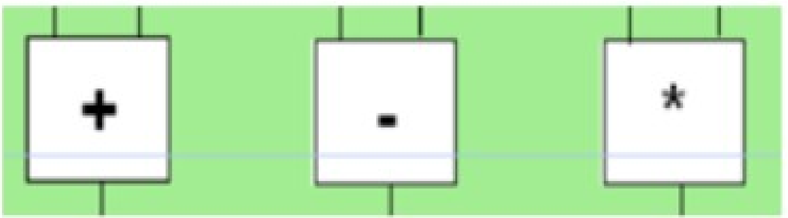
Primitive Logic Gates



Multiplexers

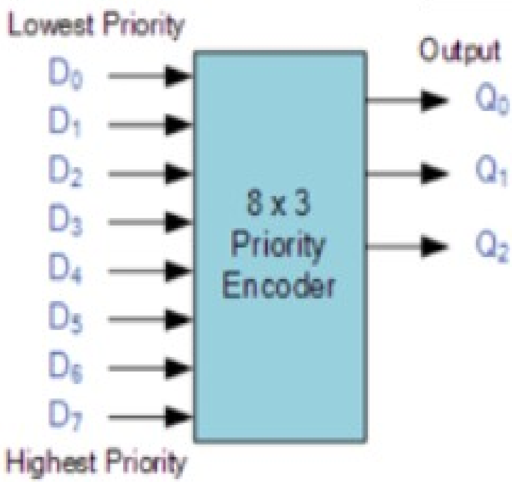


Arithmetic circuits



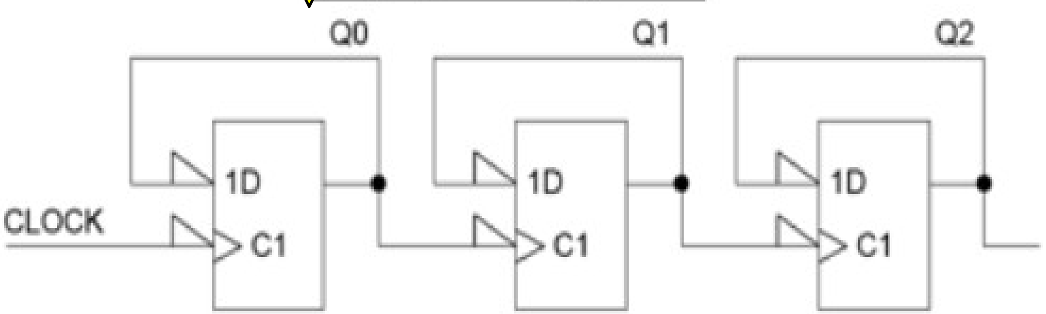
Encoders

(binary output)



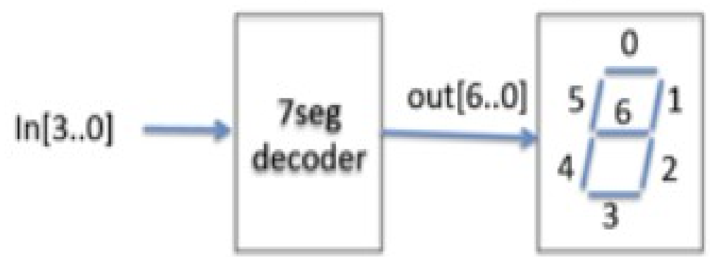
Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Q_2	Q_1	Q_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	x	0	0	1
0	0	0	0	0	1	x	x	0	1	0
0	0	0	0	1	x	x	x	0	1	1
0	0	0	1	x	x	x	x	1	0	0
0	0	1	x	x	x	x	x	1	0	1
0	1	x	x	x	x	x	x	1	1	0
1	x	x	x	x	x	x	x	1	1	1

Flipflops and Registers



Decoders

(binary input)



Basic Digital Building Blocks

1. Primitive gates – We have the basic AND, OR, NAND, NOR, XOR and XNOR gates.

2. Multiplexers MUXs – These are really useful component. Shown here is a 2-to-1 MUX with two data inputs and one select input. The output is one or the other depending on the select input (sel). We often put a number of these together to provide multiplexing function to a mult-bit data word (as shown here with two 3-bit numbers).

3. Arithmetic circuits – Commonly found are adders and multipliers. Subtractor can be built from an adder if we use 2's complement representation of signed integers.

4. Encoders/Decoders – These two are related. **Encoding** is a logic module that reduces (encodes) a large number of bits and produces fewer output bits. **Decoders** are the opposite. Shown here is a 7-segment display decoder, where 4 input bits are decoded into 7 logic signals to drive the seven segments of the display. The **encoder** here is known as a **priority encoder**. It produces a 3-bit output showing where the first '1' is encounters from the most-significant bit D7 to the least significant bit D0.

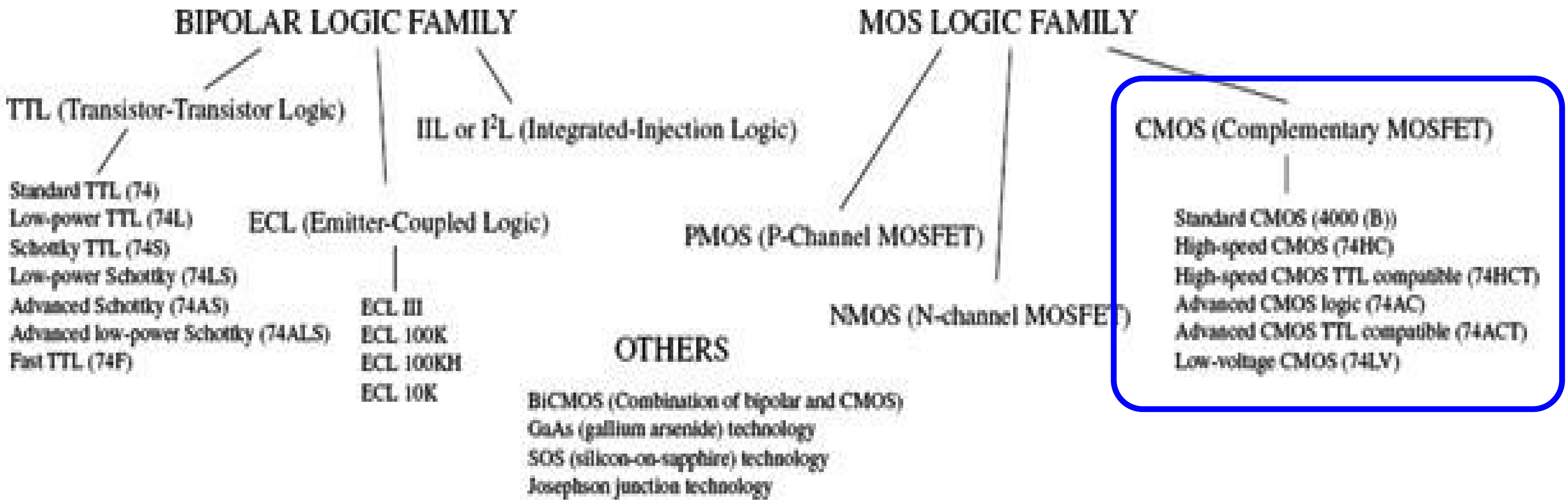
5. Flipflops and Registers – These are the building blocks for all sequential circuits. As will be seen later, we really only use one type of flipflop – the D-FF.

These are all important components that all digital circuit designers need to be familiar with. However, nowadays, we rarely design large digital systems at such low levels. Instead we generally try to express these building blocks in a more abstract manner in a hardware description language (as we will see in later lectures).

In addition to these basic blocks, we also have memory devices and microprocessors.

How logical and arithmetical functions are implemented in digital circuits ?

Transistor Gates and Logic Families



Transistor Gates and Logic Families

Most logic circuits are built using standardized Integrated Circuits (IC) from one of the logic families

Basic gates discussed here:
- TTL family developed by Texas Instruments
- ECL family developed by Motorola Inc.
- CMOS family developed by RCA

Requirements on voltage levels:

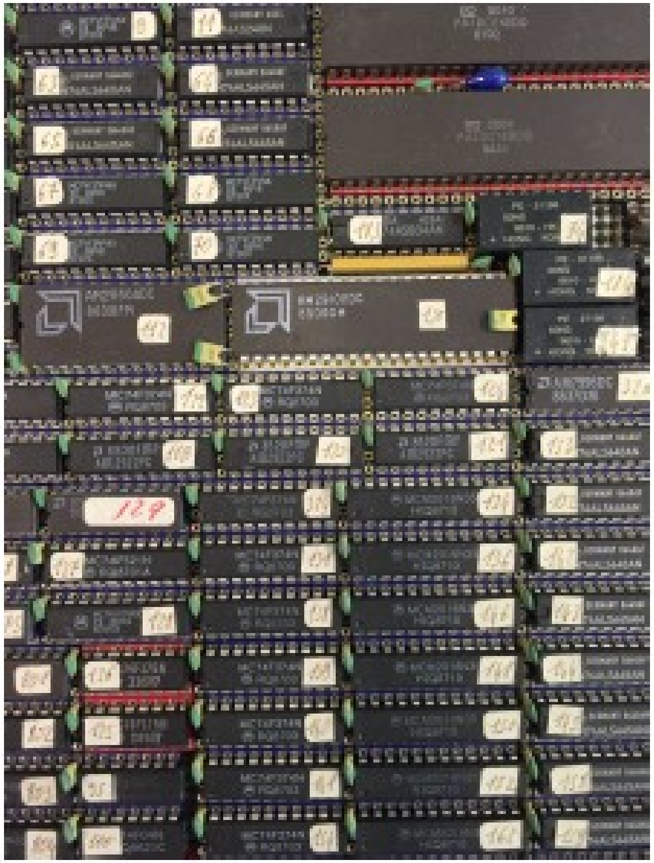
1) voltage levels output → input compatibility
2) large fanout, ie can drive many inputs with minimal change in output levels

We'll see that digital circuits have delays !

On the negative side,
delays result in finite processing times and therefore set a limit on speed …
… but on the positive side,
delays are essential for the existence of flip-flops (ie memory !) and
of state machines (ie “processors“ !)

TTL

- Prior to (C)MOS logic, digital logic systems were built using Small Scale Integration (SSI) and Medium Scale Integration (MSI) integrated circuits.
- These integrated circuits appeared in the mid 70's and were extremely popular for about 20 years (but still in use), eventually supplanted by programmable logic such as FPGA.



A. Marchioro / CERN

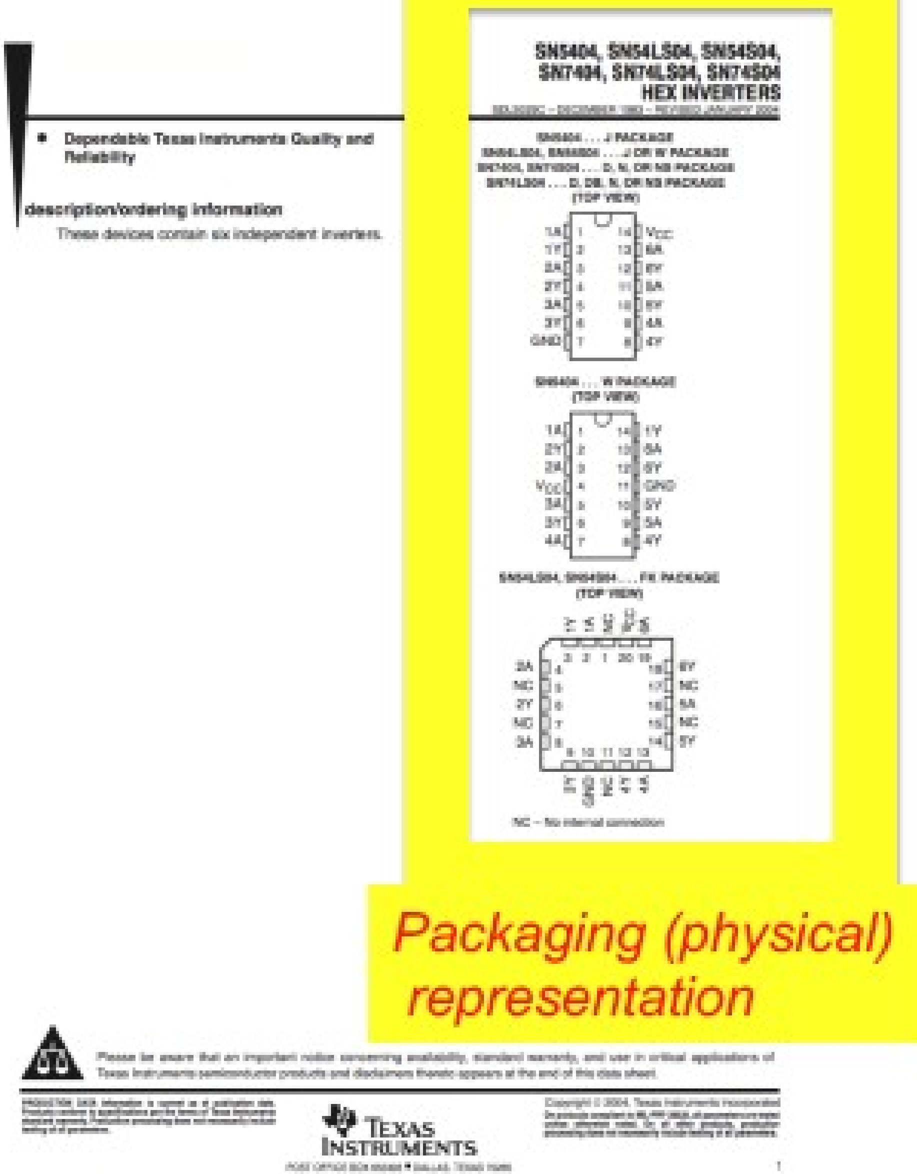
TTL

- Manufacturers designed a family of SSI/MSI ICs that went by the name of TTL (Transistor-Transistor-Logic) logic. Such ICs used bipolar transistors.
- They used a single 5V supply
- Cascading chips required no passive component
- Power dissipation was OK (for the time)
- Chips were robust and reliable
- Cost/chip was reasonably low
- System speed achievable was >10 MHz (if properly designed)

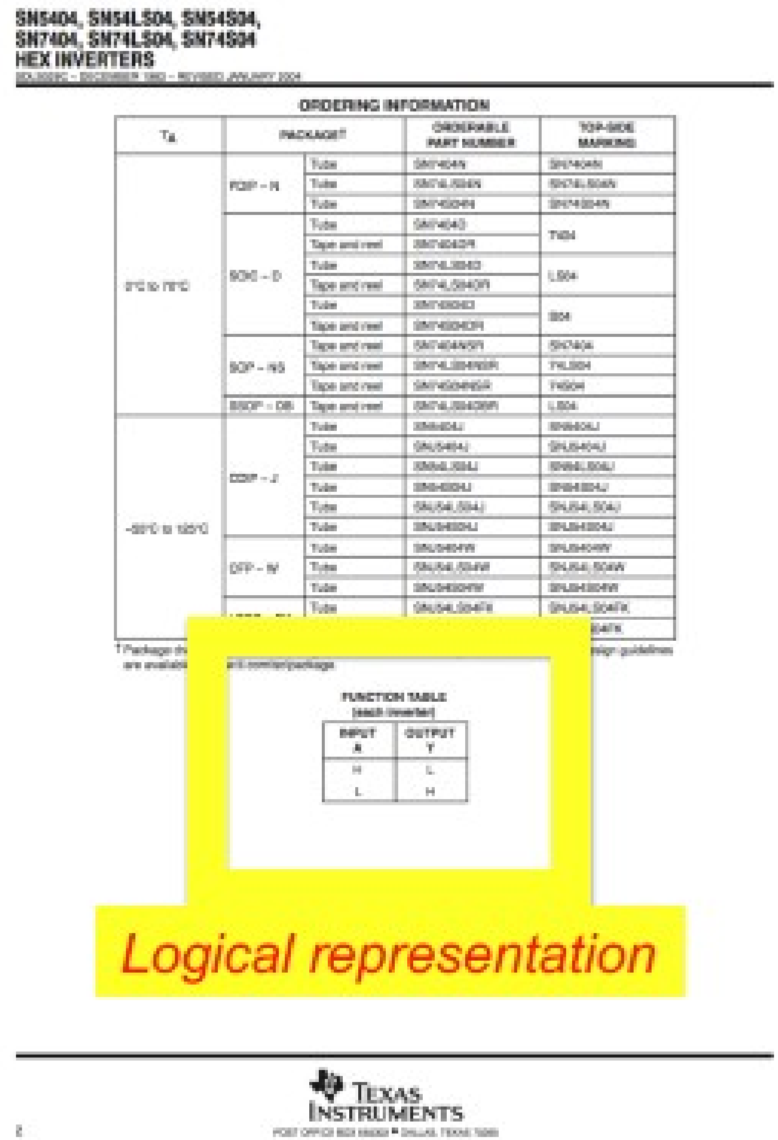
- Several families (almost always compatible with one another) existed:
  - 74xx base series
  - 74LSxx low power,
  - 74Sxx high speed
  - 74Fxx very high speed
  - 74ACTxx CMOS
  - … many many others
- "xx" is a two or three digit number identifying a specific function, e.g. 00 is a quadruple 2-input NAND gate

A. Marchioro / CERN

7404 circuit data sheet



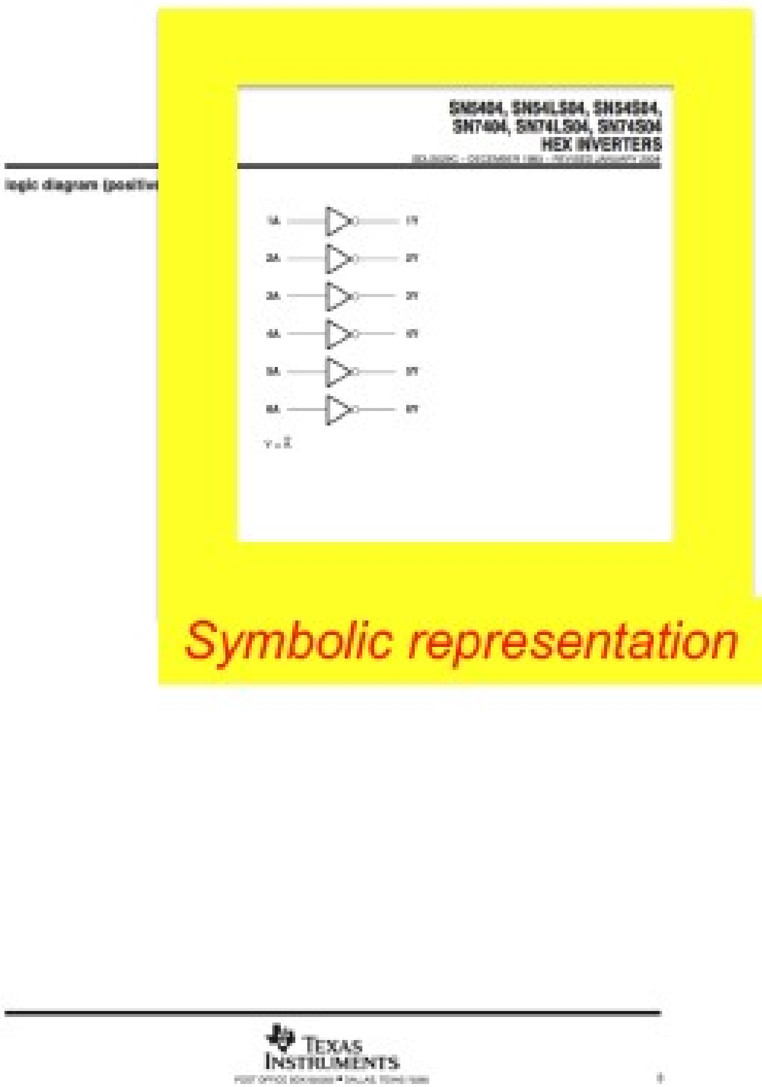
Packaging (physical) representation

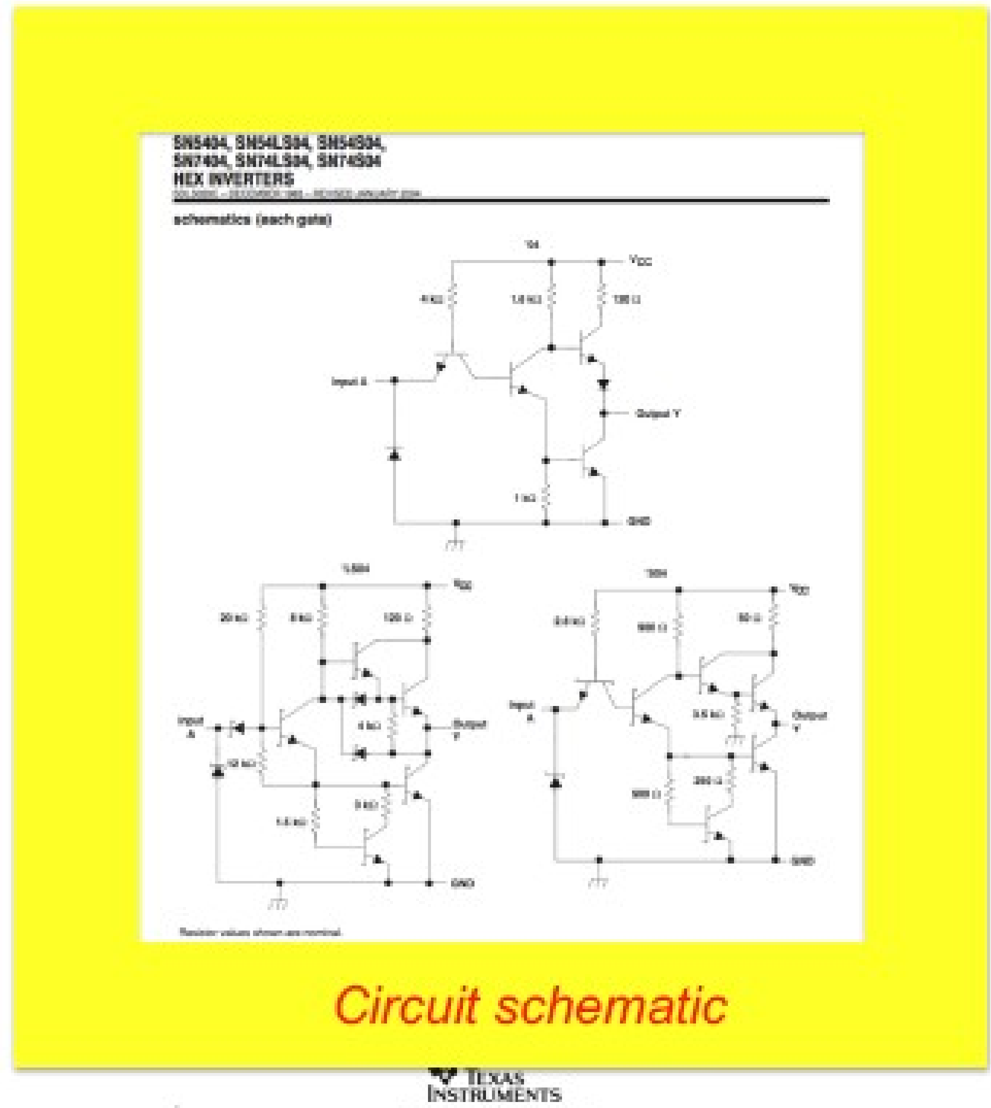


Logical representation

A. Marchioro / CERN

7404 circuit data sheet (2)

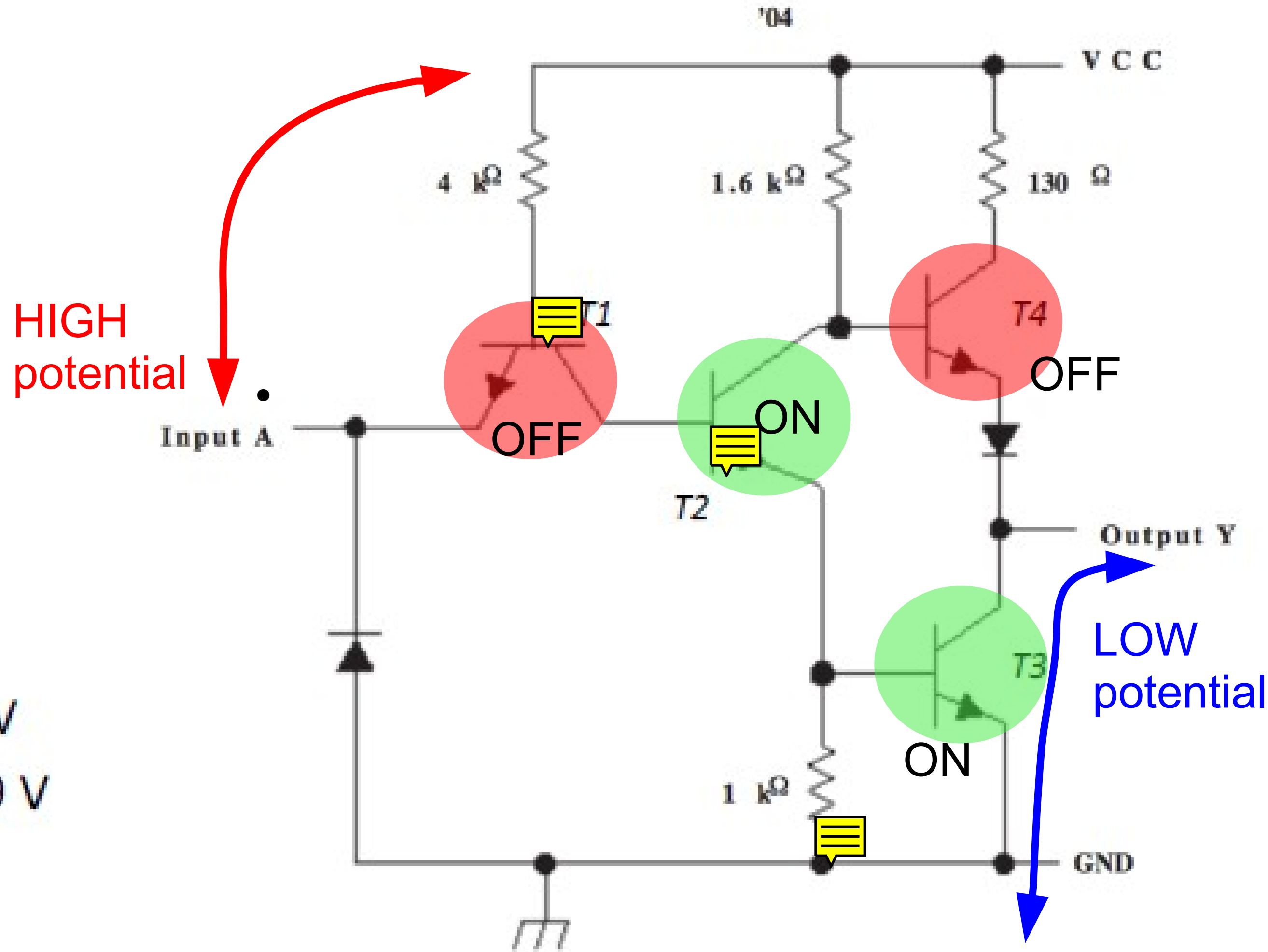




A. Marchioro / CERN

TTL: How does it work?

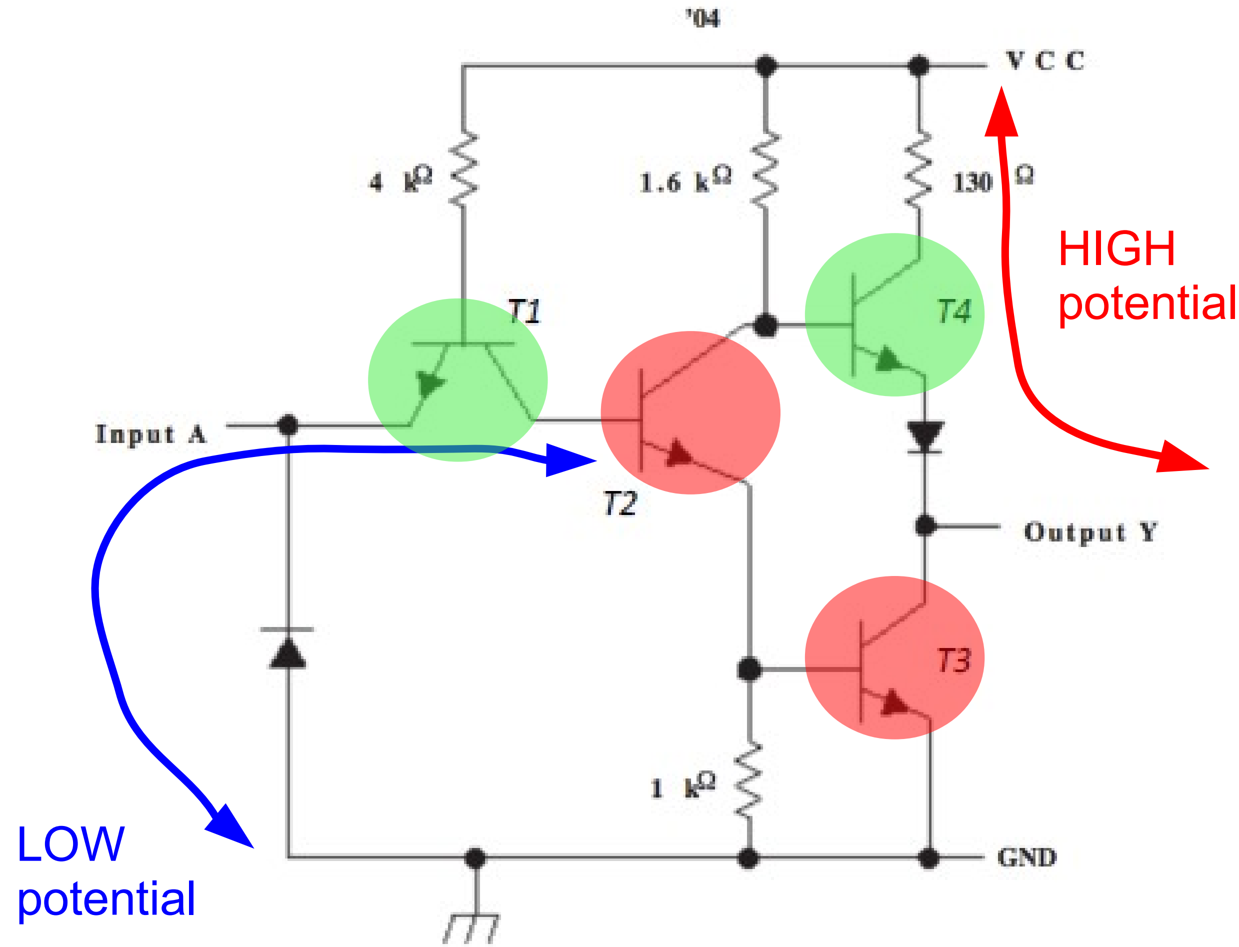
- A = 5 V
  - $V_{B,T1}$ = ~5 V
  - T1 off
  - $V_{C,T1}$ = ~5 V
  - $V_{B,T2}$ = ~5 V
  - T2 on
  - $V_{B,T3}$ = ~0.7 V
  - $V_{CE,T3}$ = ~0.2 V
  - Y = 0.2V
  - $V_{E,T4} = V_{CE,T3} + V_D$ = ~0.9 V
  - $V_{B,T4} = V_{BE,T3} + V_{CE,T2}$ = 0.9 V
  - T4 off



A. Marchioro / CERN

TTL: How does it work?

- A = 0 V
  - $V_{B,T1} = 0$ V
  - T1 on
  - $V_{C,T1} = 0.2$ V
  - $V_{B,T2} = 0.2$ V
  - T2 off
  - $V_{B,T3} = 0$ V
  - T3 off
  - T4 on
  - $V_{B,T4} = 5$ V
  - $V_{E,T4} = 5 - 0.7 = 4.3$ V
  - Y = 4.3 V

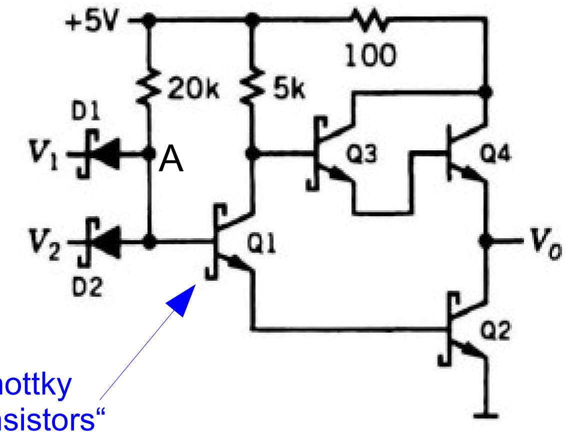


A. Marchioro / CERN

Low Power, Shottky TTL - LS-TTL → NAND gate

The basic circuit of the LS-TTL family is the NAND gate

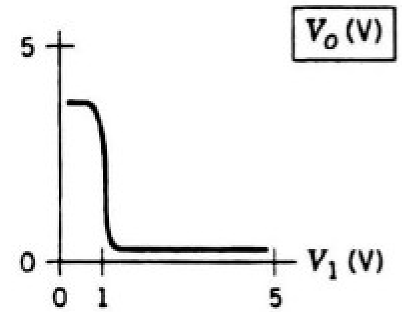
Transistors have a Schottky diode from base to collector and thus do not saturate (Q4 is an exception because it is kept out of saturation by the Schottky diode in Q3)



"Schottky Transistors"

(a)

Voltage transfer characteristic when one input (V2) is held HIGH



(b)

Assume V2 = 5V → D2 is reverse biased and can therefore be ignored

If V1=0 → VB1 ~ 300mV, and Q1 and Q2 are both cut off

Note: D1 contends V_{drop} with BE junctions of Q1 and Q2 (2 x V_{drop}) on node A

→ Darlington Q3/Q4 is on, as at least sees the r_0 load of Q2

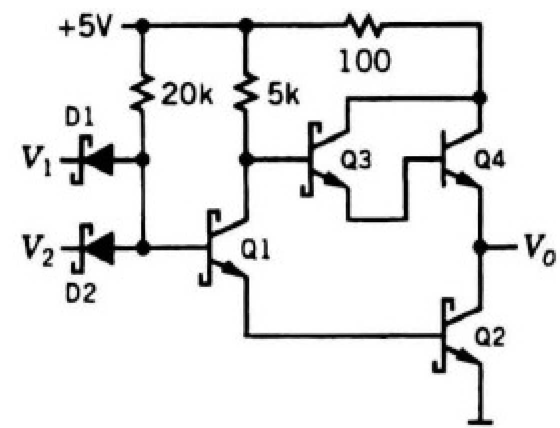
If the output load current is within the limits prescribed by family rules (usually 400 μA) the output voltage VO is 5V-1.4V ~ 3.6V (Vdrop of Darlington follower Q3|Q4 is 1.4V)

VO remains at this value until V1 is raised to about 1V (threshold voltage) at which point Q1 and Q2 start to turn on and VO starts to drop because the base of Q3 is pulled down by Q1

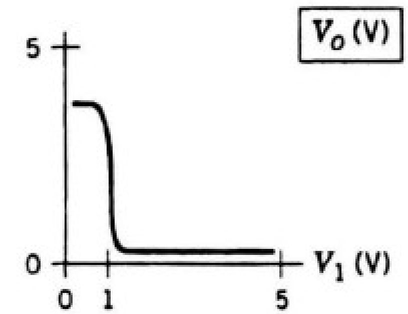
Low Power, Shottky TTL - LS-TTL → NAND gate

If $V1 > 2V_{drop} \sim 1.4V$ → D1 is cut off
and Q1 and Q2 are fully on
(if D1 were on → $V_{node\ A} = 3V_{drop}$
→ Q1 and Q2 on → in conflict on node A)
The base currents in Q1 and Q2
are much more than is required
for saturation
→ VCE1 and VCE2 are clamped at about
400 mV by their B-C Schottky diodes
→ VB3-VE4 is thus about one diode drop,
and Q3 and Q4 are therefore both cut off

NAND gate



(a)



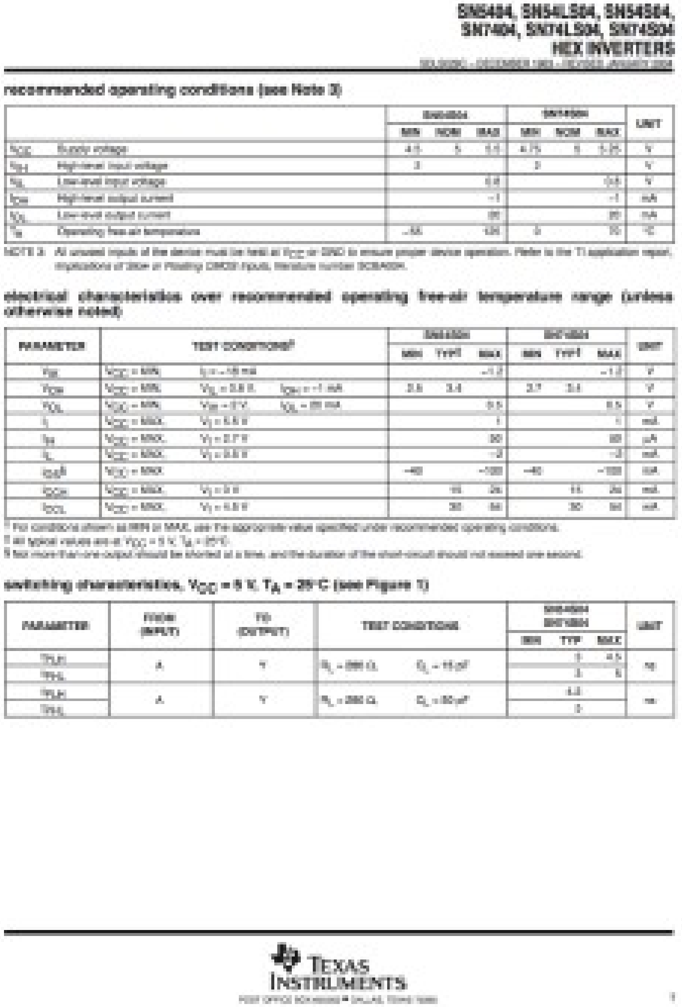
(b)

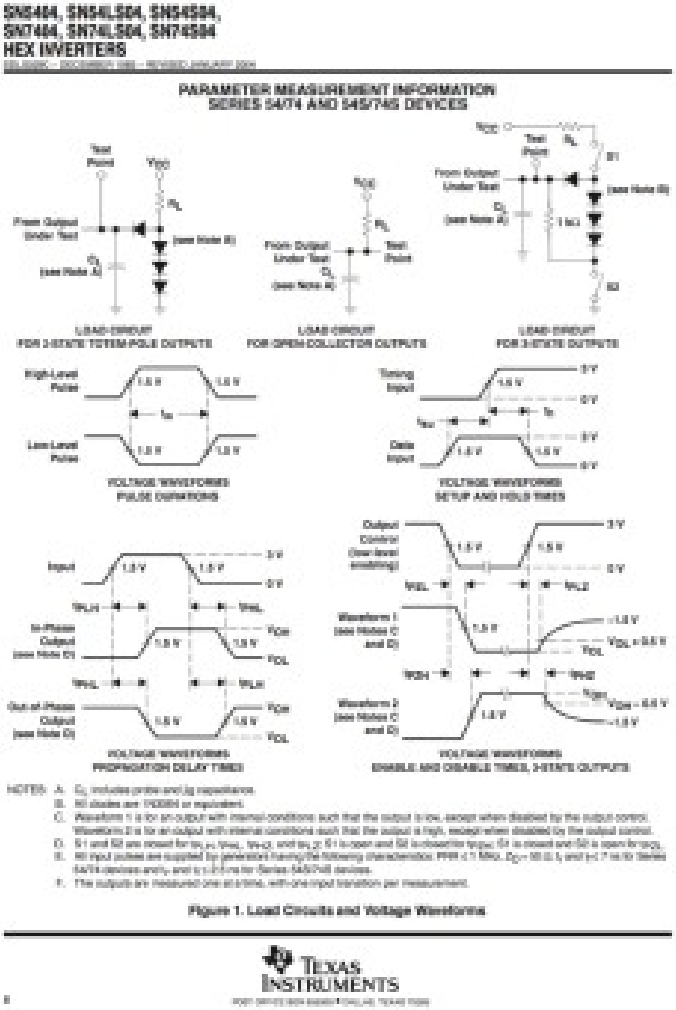
Following family rules we define
- LOW as any voltage under 0.8 V
- HIGH as any voltage above 2.4 V

→ LOW input will give a HIGH output, and vice versa
If V1 and V2 are both allowed to take on LOW or HIGH values
→ VO will be LOW only if V1 and V2 are both HIGH, → we have a NAND gate

Note: A HIGH input to an LS-TTL gate sees a reverse-biased diode
→ the fanout in the HIGH state is large
According to family rules, a LOW input to an LS-TTL gate sinks 0.4mA
in the worst case, and a LOW output must be able to sink 4mA
→ the fanout in the LOW state is thus also large, about 10

7400 circuit data sheet (4)

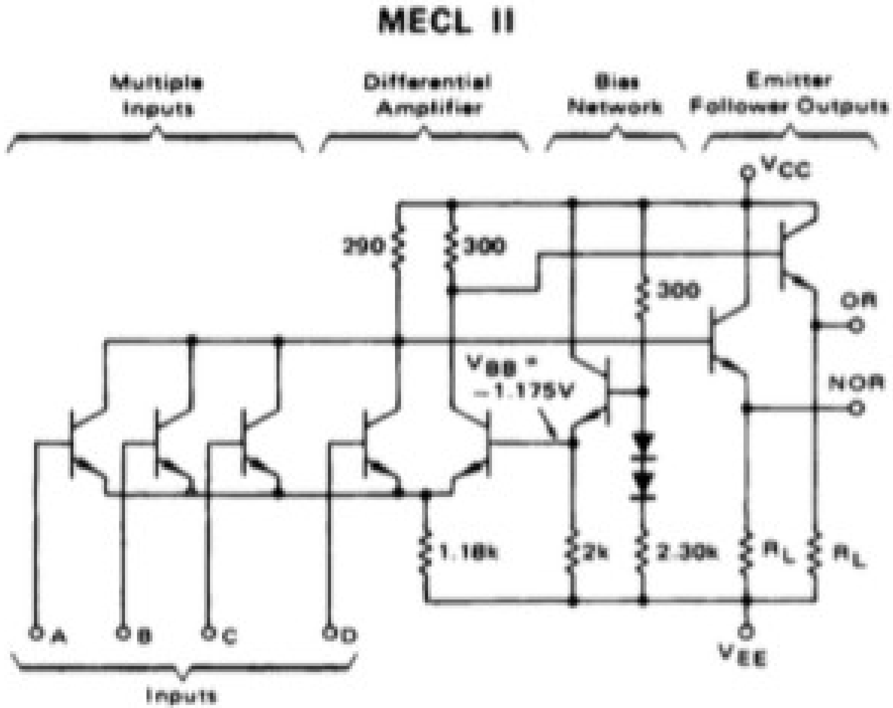


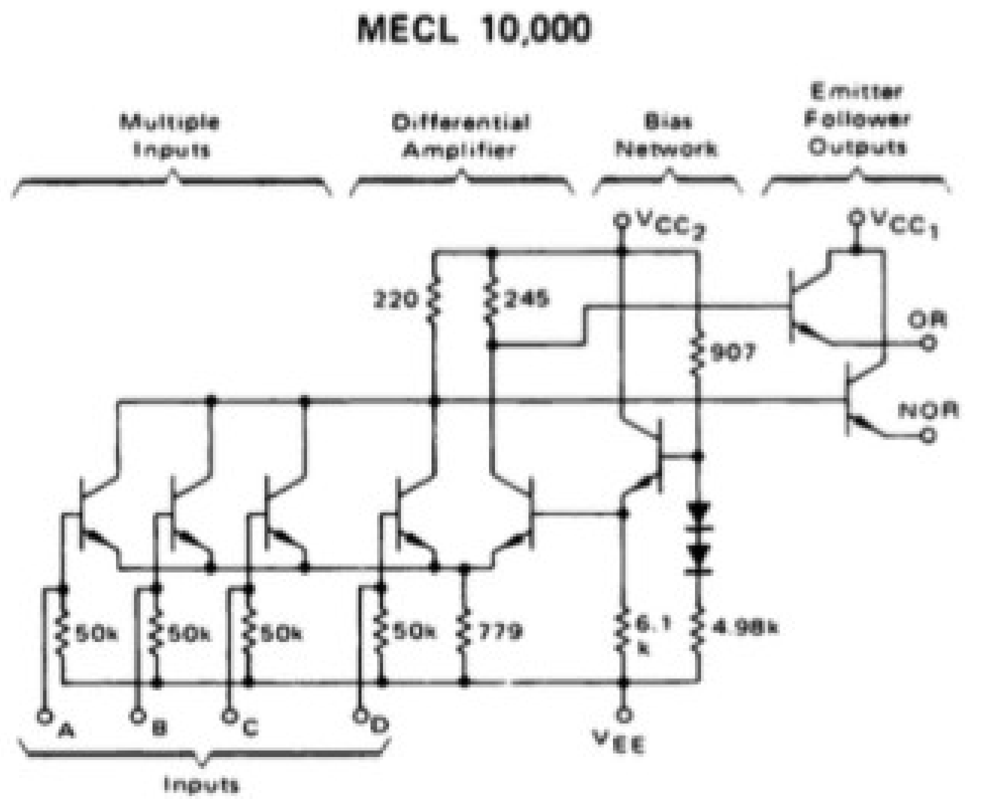


ECL Logic

- A further logic family was developed to achieve even higher speed that TTL.
  - MECLI,
  - MECLII,
  - MECLIII
  - MECL10000
- Used -1.2 and -5.2 V
- Considerable more power hungry (hotter) than TTL
- Expensive
- Temperature sensitive
- Less complex functions than TTL
- System speed achievable > 100 MHz (with very careful board design)
- Requires *transmission lines* for signal propagation
- Non-interchangeable chips between families

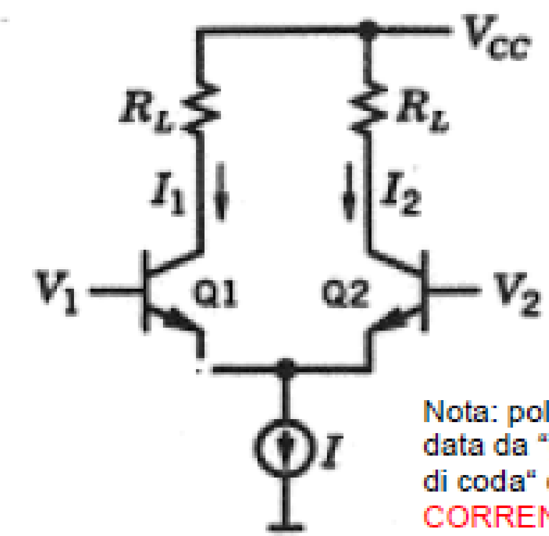
ECL gate example

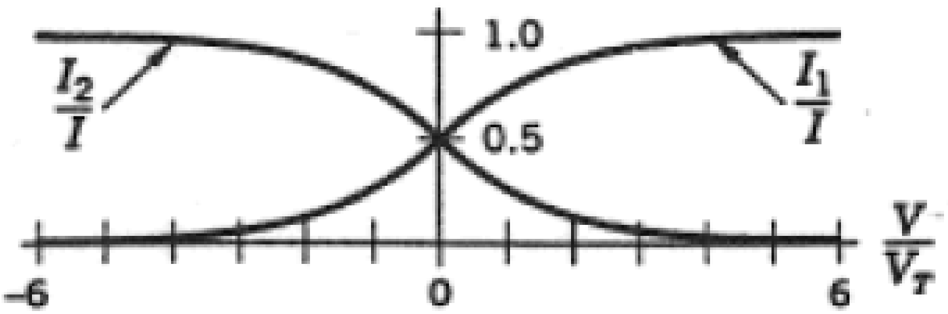




Emitter Coupled Pair

Misura di differenze di tensione e` problema fondamentale (vedere Op.Amp.)
→ discutiamo stadio di ingresso molto utilizzato: emitter coupled pair che si puo` vedere come un base comune pilotato da un emitter follower…





Nota: polarizzazione data da "generatore di coda" che eroga I CORRENTE COSTANTE

$$V = V_1 - V_2$$

$$I_{1,2} = \frac{I}{2}\left(1 \pm \tanh \frac{V}{2V_T}\right)$$

Supponiamo $R_L < r_0$ e consideriamo la differenza di tensione tra i due ingressi $V = V_1 - V_2$ si vede che

$$V = V_{BE1} - V_{BE2} = V_T \ln \frac{I_1}{I_2}$$

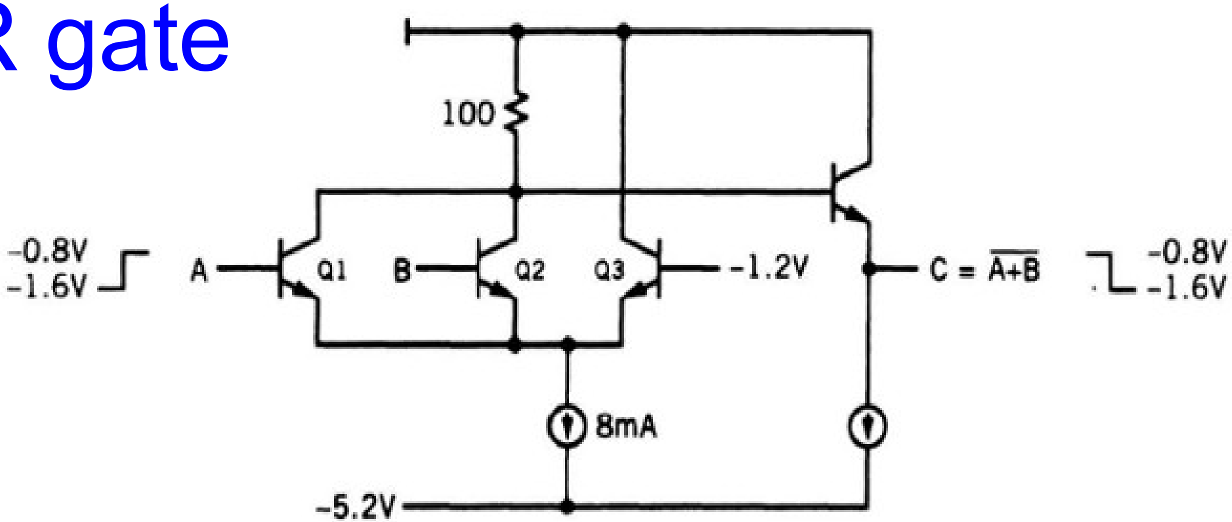
Ignorando le piccole correnti di base, la somma delle correnti di collettore e` corrente si bias $I = I_1 + I_2$

$$I_{1,2} = \frac{I}{2}\left(1 \pm \tanh \frac{V}{V_T}\right)$$

Gia` con una differenza di tensione di 125mV ($5V_T$) si ha il 99% della corrente in un transistor, l'altro e`cut-off
→ abbiamo un interruttore di corrente (circuiti digitali) … meccanismo ad altalena (basculante) …

ECL family → NOR gate

The Emitter-Coupled Logic (ECL) family has NOR as basic gate circuit



For a moment assume that the bases of Q1 and Q2 are tied together. The circuit can then be described as a differential amplifier whose single-ended collector output voltage is buffered by an emitter follower

Assuming that VBE = 0.8 V in all transistors when they are conducting, the output voltage is -0.8 V when Q1|Q2 is cut off, and -1.6 V when Q3 is cut off and all the current in the 8 mA current source goes through Q1|Q2

With a bias of -1.2 V at the base of Q3, the current switches between Q1|Q2 and Q3 when the base voltage on Q1|Q2 switches between levels that are about 120 mV above and below -1.2 V (threshold)

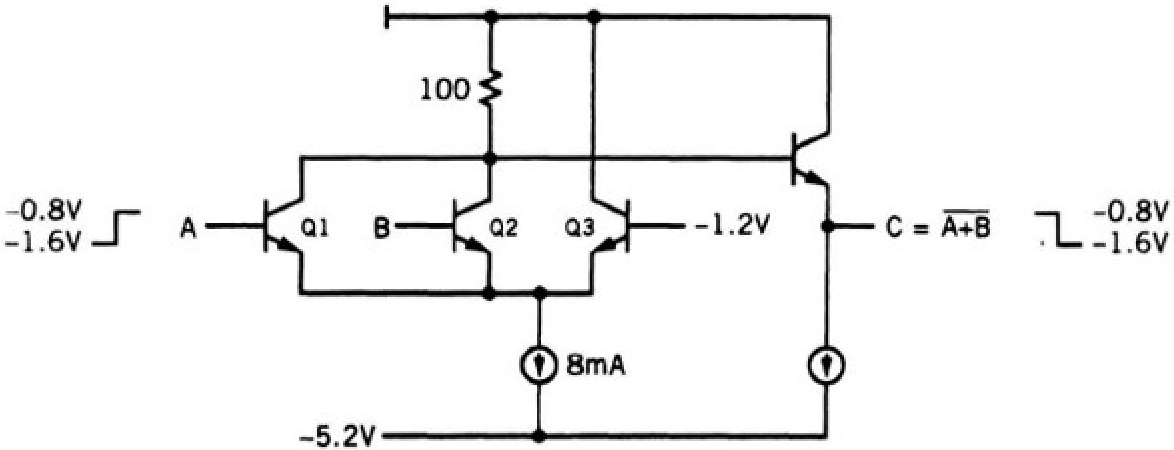
If we define

- HIGH as a voltage above -1.0 V and
- LOW as a voltage below -1.4 V,

a LOW input at the base of Q1|Q2 will give a HIGH output at C, and vice versa

ECL family → NOR gate

The Emitter-Coupled Logic (ECL) family has NOR as basic gate circuit



If we now untie the bases of Q1 and Q2→ HIGH at either A or B will make C LOW → we are dealing with an ECL NOR gate.

In an actual integrated version the logic levels are somewhat different, (-0.9 V and -1.8 V) but the circuit structure is essentially the same

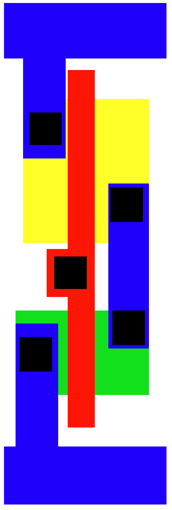
Note:

1) ECL circuits can operate at high frequencies: 100MHz is standard and 500MHz is possible

2) ECL circuits are well suited for driving transmission lines, which are almost mandatory at such frequencies

CMOS gates

- Compact (shared diffusion regions)
- Very low static power dissipation
- High noise margin (nearly ideal inverter voltage transfer characteristic)
- Very well modeled and characterized
- Mechanically robust
- Lends itself very well to high integration levels



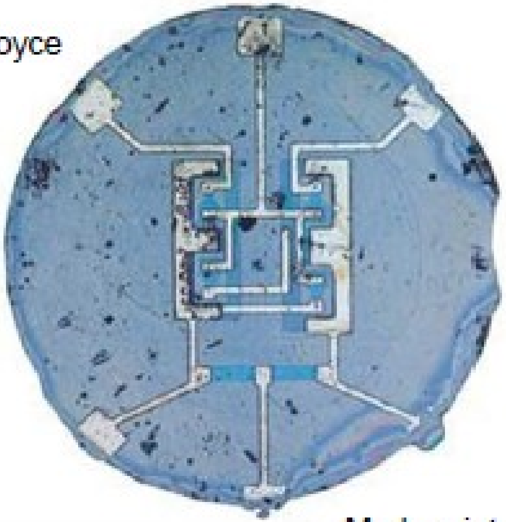
CMOS
inverter
(simplyfied)

CMOS gates

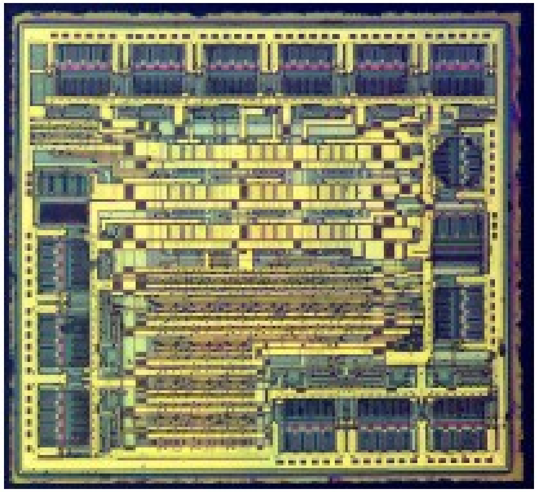
- With development of ICs the MOSFET took the main role in electronics
- We are already producing 10^{18} transistors per year - enough to supply every ant on the planet with ten transistors.
- Twenty years from now, if the trend continues, there will be more transistors than there will be cells in the total number of human bodies on Earth

First IC - Kilby

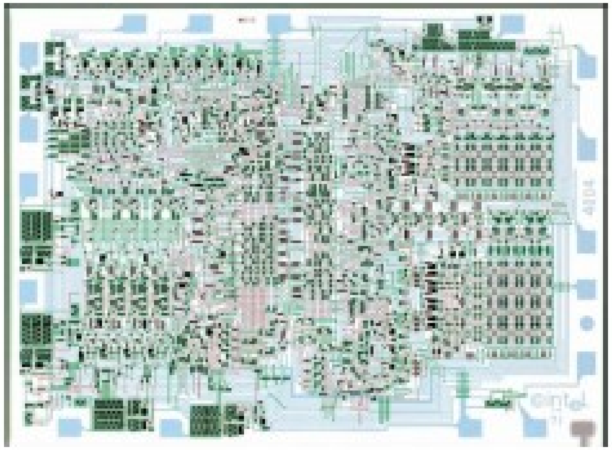
Planar IC Noyce



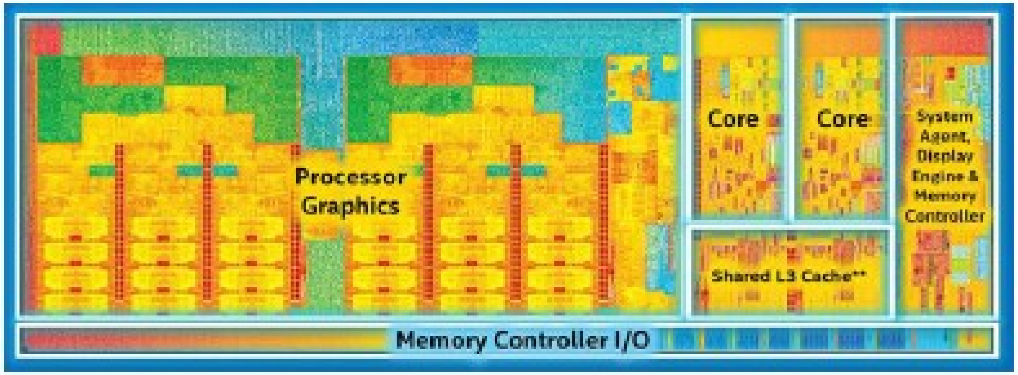
CMOS IC



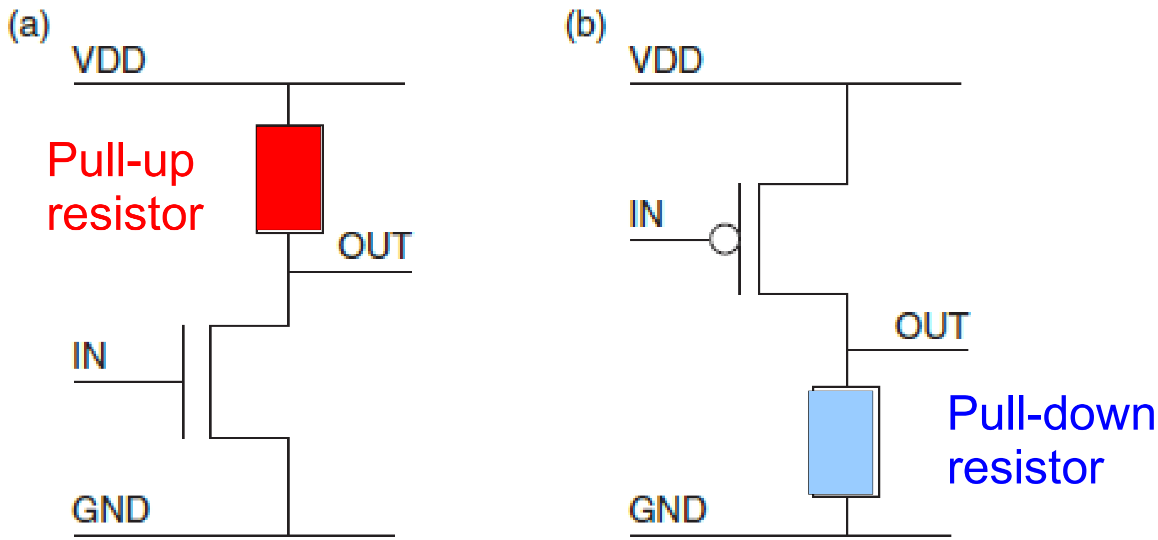
First microprocessor



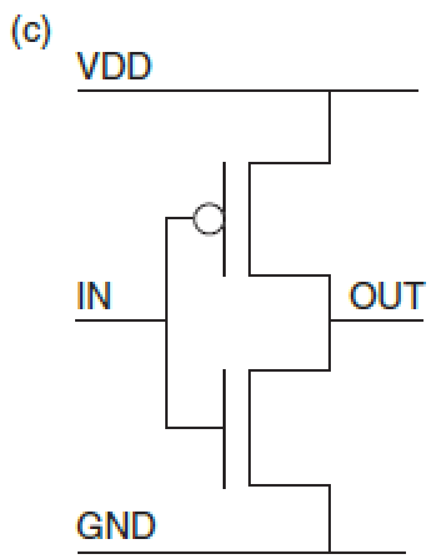
Modern intel processor



NOT gate: PMOS and NMOS inverters

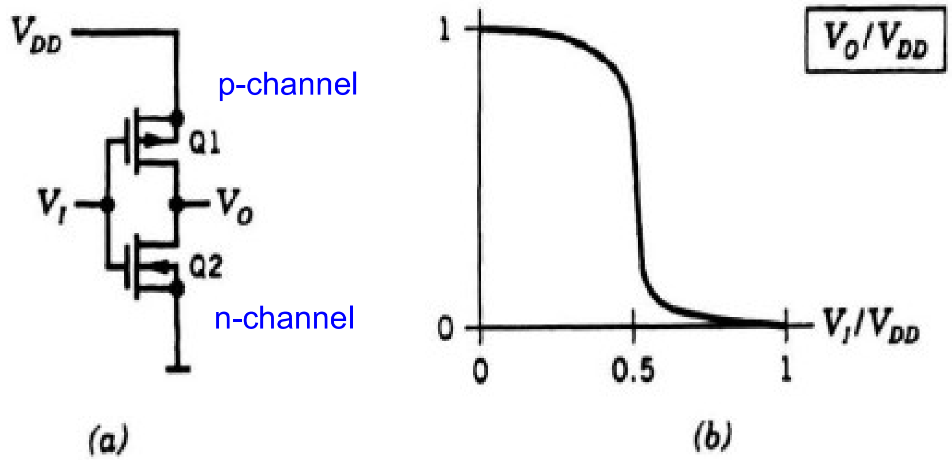


NOT gate: CMOS inverter



CMOS family → Inverter

- Complementary enhancement MOSFETs involved
- Assume for simplicity identical characteristics and threshold voltage V_{TH} = 1 V



If VI = 0 → Q2 is cut off and, assuming that VDD = 5 V
→ Q1 is well above threshold → acts like a resistor of order 1 kΩ
→ output voltage V0 = VDD and V0 remains at this value as long as VI = 1 V

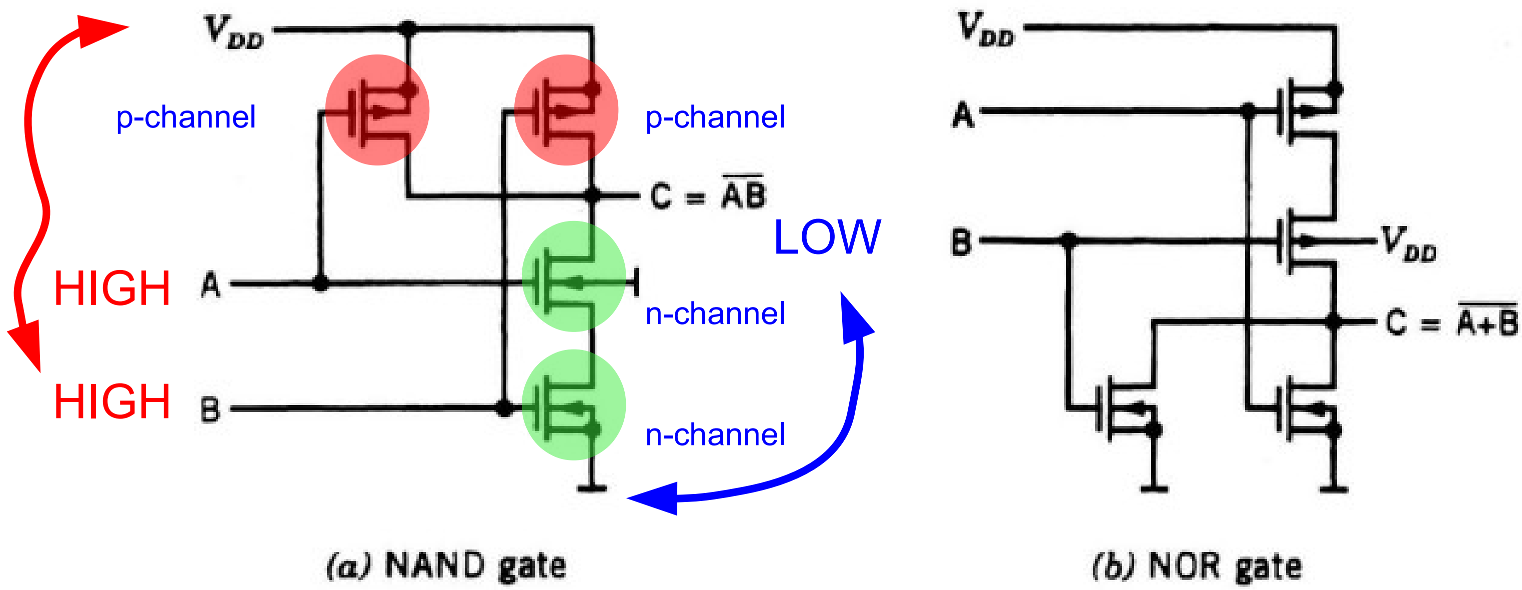
Similarly if VI is within 1V of VDD → Q1 is cut off and Q2 is on, and V0 = 0

If VI = VDD/2, Q1 and Q2 have the same drain current, and V0 ~ VDD/2 (ideally)
By superposition, the small-signal gain at this point is $2(g_m r_o/2)$ ~ 100
→ V0 changes rapidly as a function of V1 (see voltage transfer characteristic)

At room T, taking
- LOW to be any voltage below 1/3 VDD and
- HIGH to be any voltage above 1/3 VDD

is sufficient to guarantee that a LOW input gives a HIGH output, and vice versa

CMOS family → NAND and NOR gates

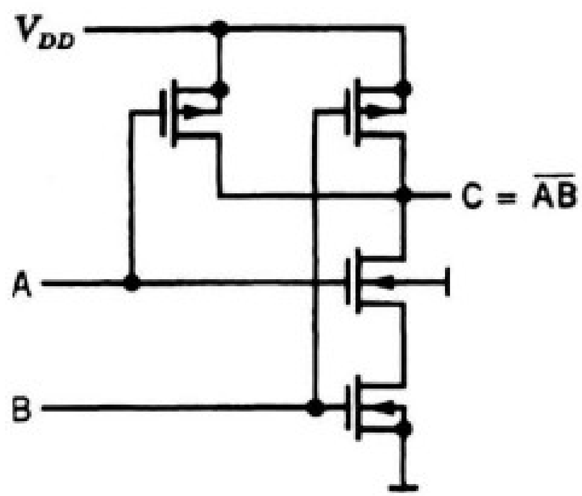


The output of the NAND gate will be LOW only if both p-channel transistors are off and both n-channel transistors are on → this requires that both inputs be HIGH

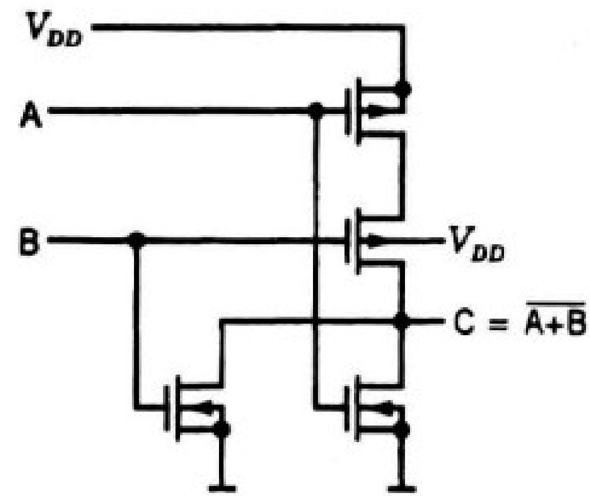
The output of the NOR gate will be HIGH only if both p-channel transistors are on and both n-channel transistors are off → this requires that both inputs be LOW

Note: great advantage of CMOS gates: they draw negligible current from the power supply when they are in a quiescent state …
… this advantage is lost if they change state at a high rate: for output transitions $0 \rightarrow V_{DD}$ the current that charges the load capacitance C_L is provided by the power supply. For output transitions $V_{DD} \rightarrow 0$, C_L discharges to ground. If this happens at a frequency f, the power supply must provide a current f x C_L x V_{DD}
For f=10MHz, C_L=10pF, and V_{DD}=5V, the current is 0.5mA (… far from negligible)

CMOS family → NAND and NOR gates



(a) NAND gate



(b) NOR gate

The output of the NAND gate will be LOW only if both p-channel transistors are off and both n-channel transistors are on
→ this requires that both inputs be HIGH

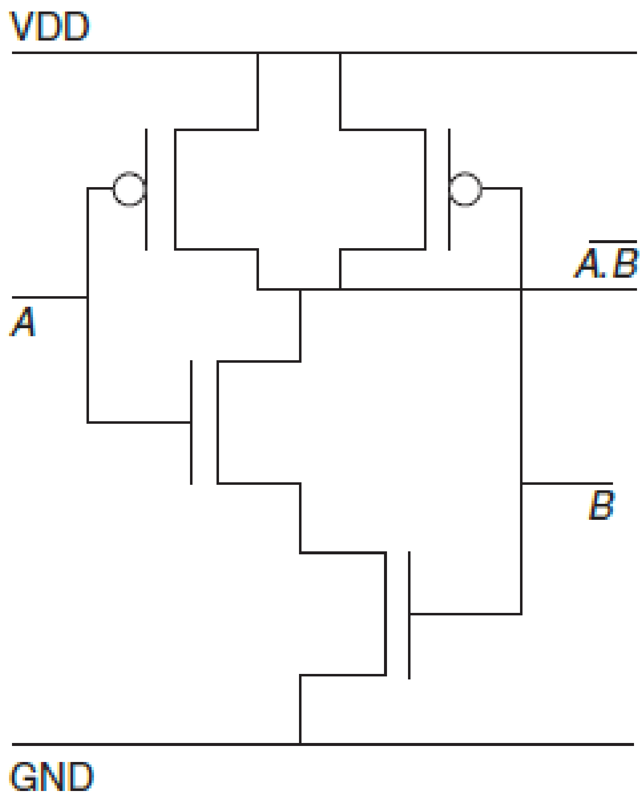
The output of the NOR gate will be HIGH only if both p-channel transistors are on and both n-channel transistors are off
→ this requires that both inputs be LOW

Note:
- input currents in CMOS circuits are the gate currents of few MOSFETs and thus extremely small
- In contrast, a CMOS circuit output can tolerate a load current of order 1mA

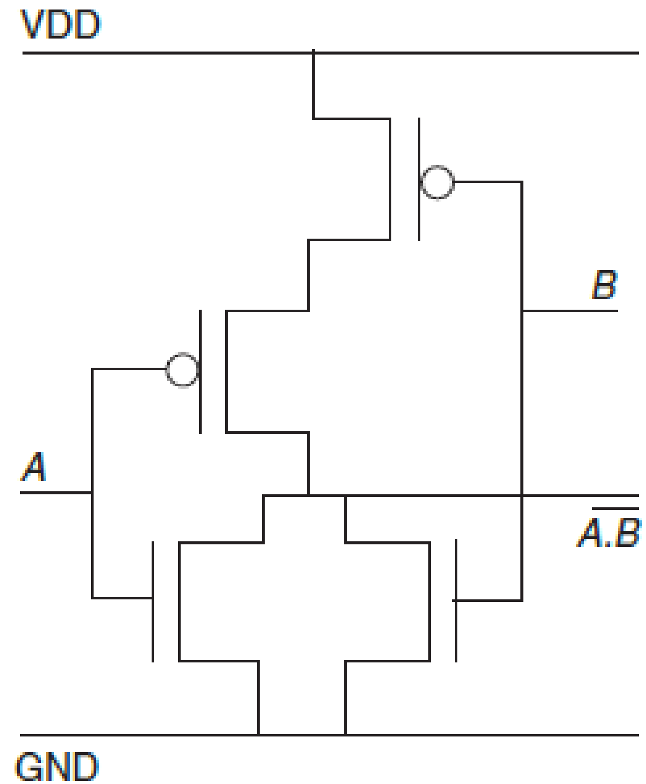
→ fanout capability is very large

Note:
each gate, however, has an input capacitance,
→ driving too many gates can result in an unacceptably slow response

NAND and NOR gates (CMOS)



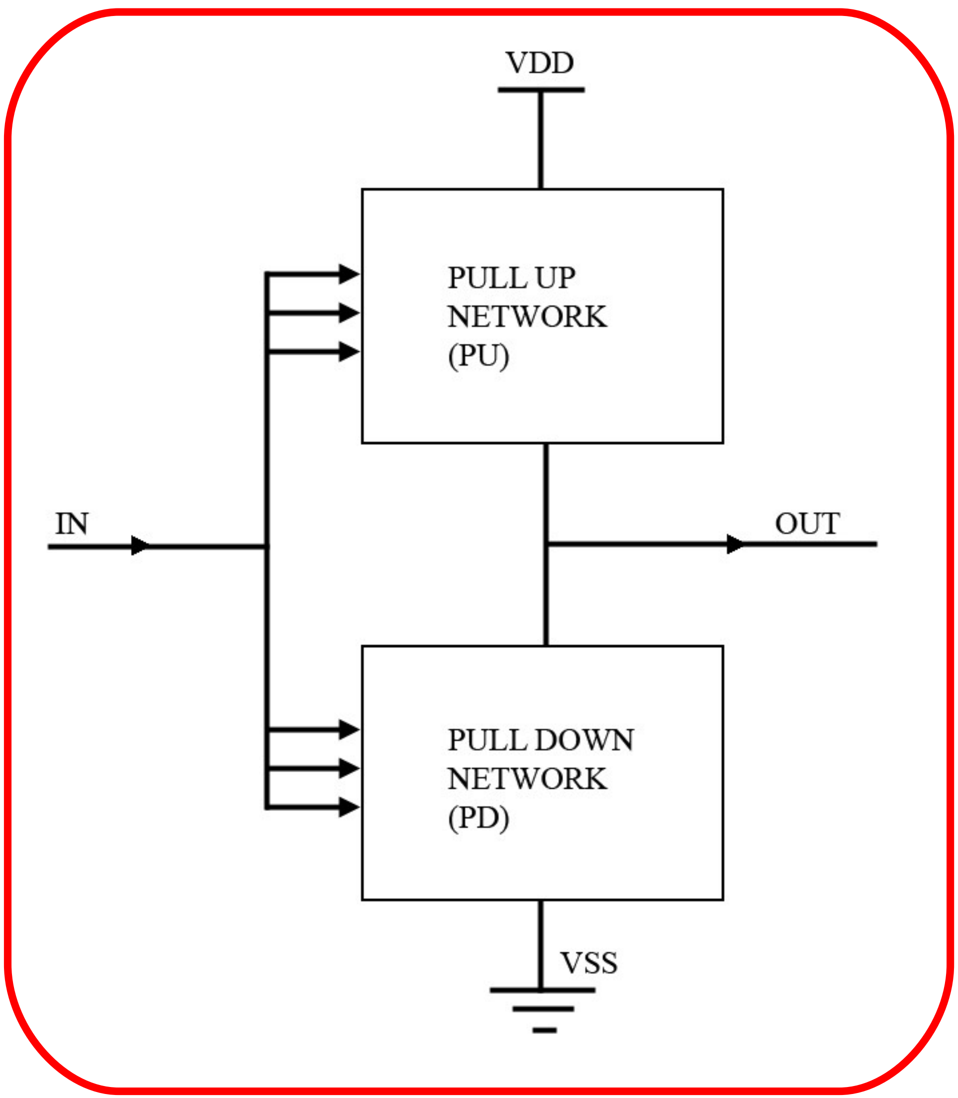
NAND

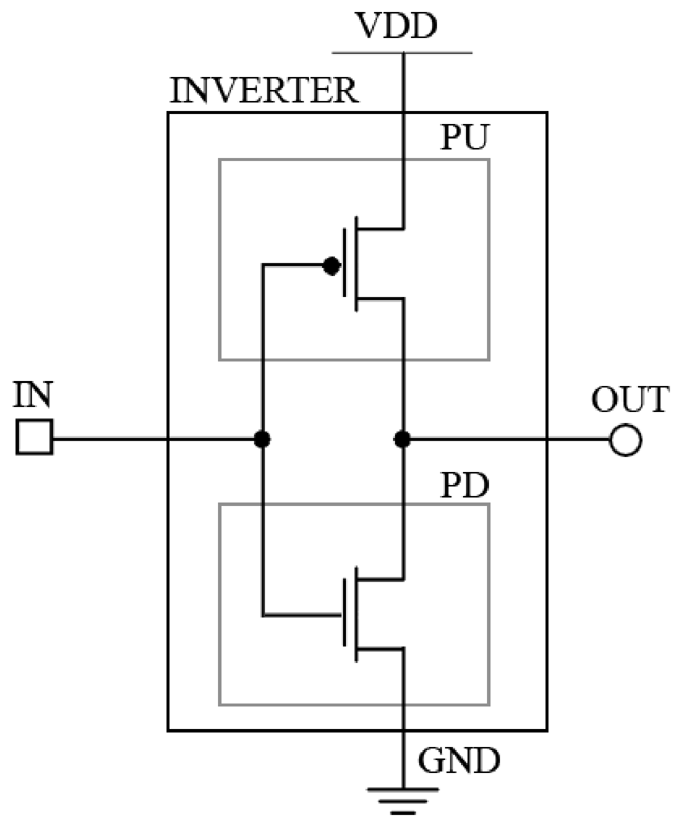


NOR

Note: NAND (and similarly NOR) allow to build any other type of gate

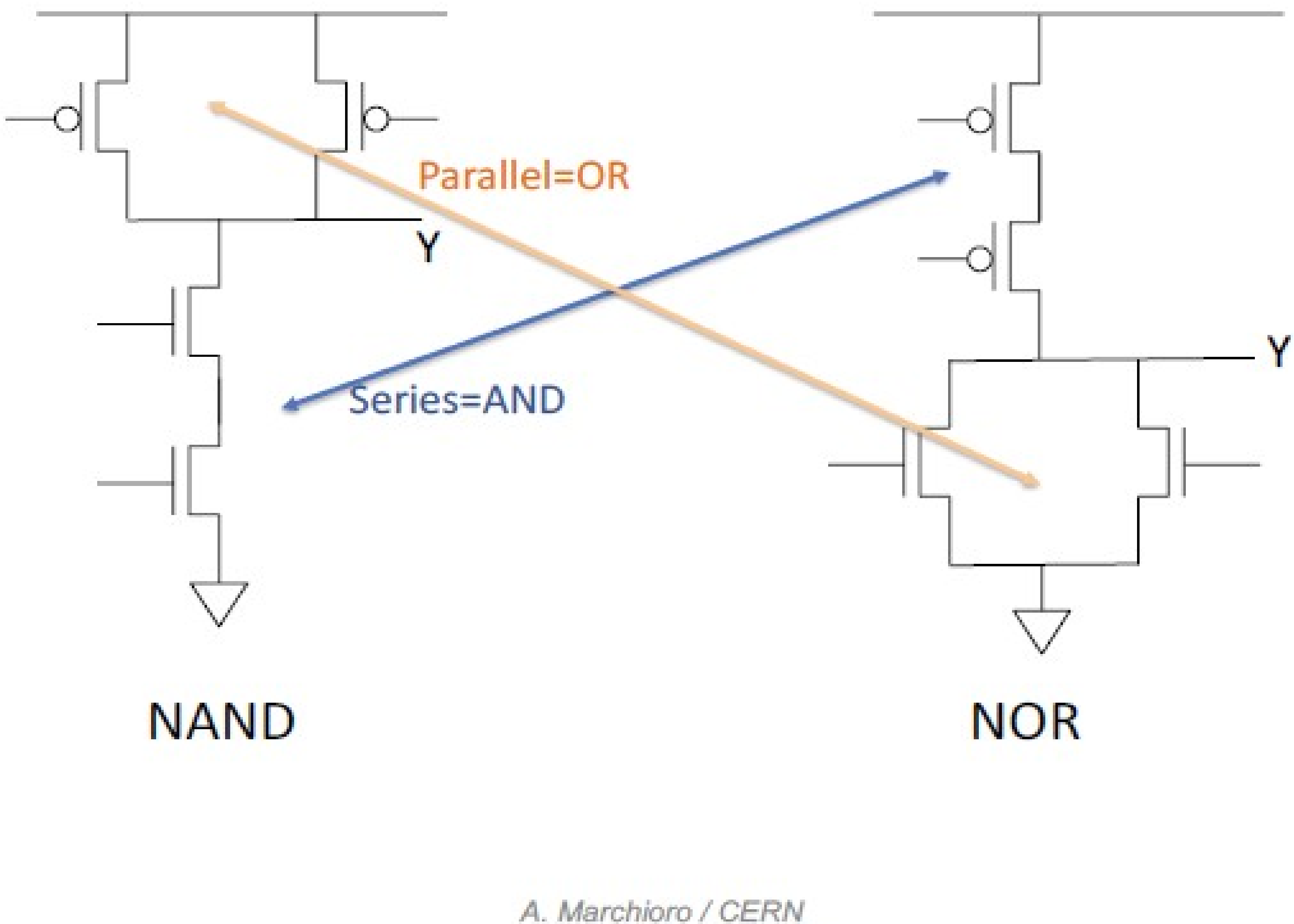
Deciphering static gates structure





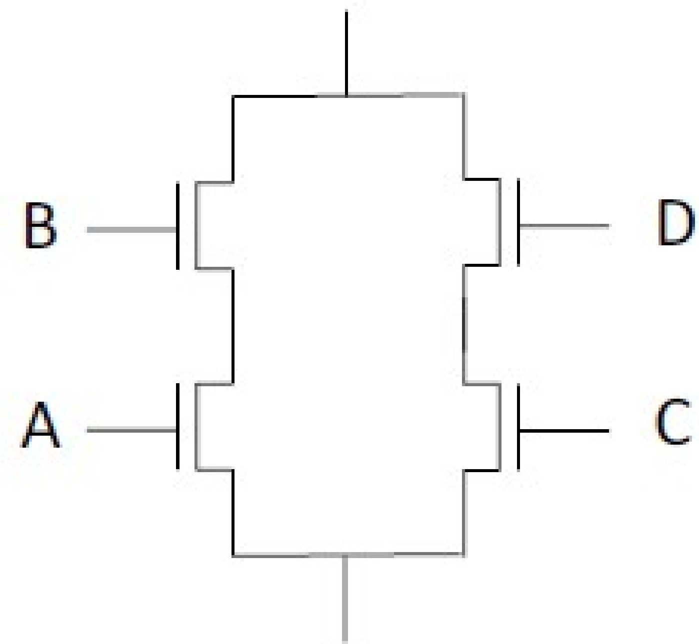
Example… inverter

Deciphering static gates structure



More complex gate

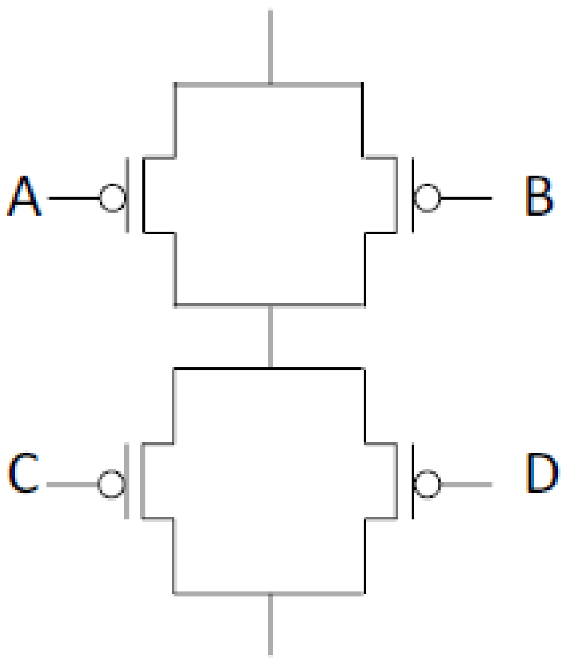
$F = \overline{(A * B) + (C * D)}$



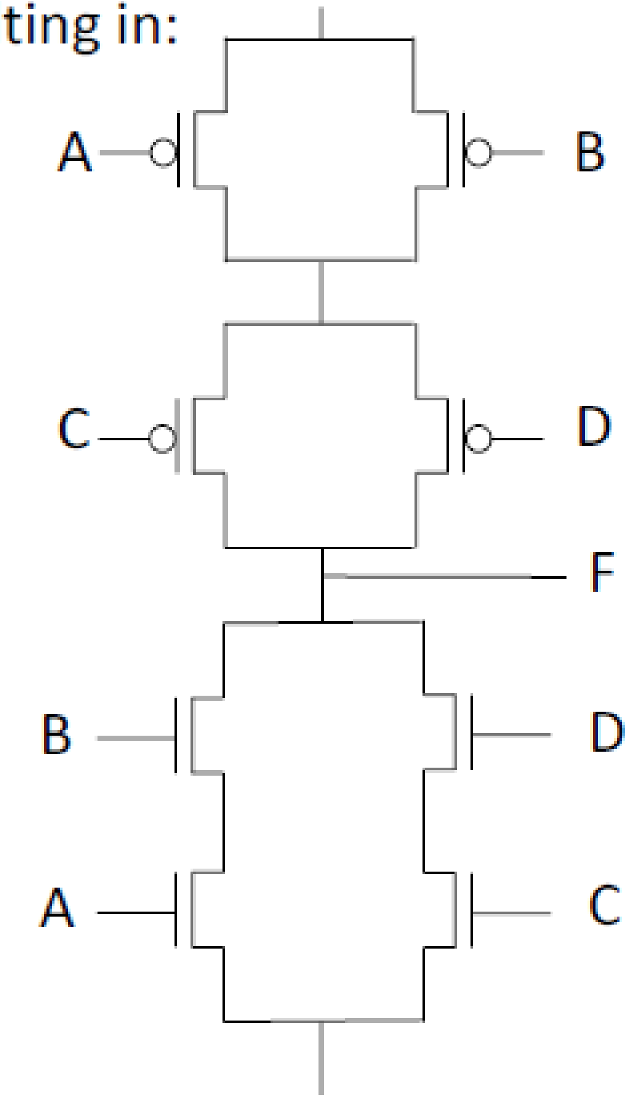
Pulldown network

More complex gate (2)

The P network is constructed from the complement:
$F' = (\overline{A} + \overline{B}) * (\overline{C} + \overline{D})$

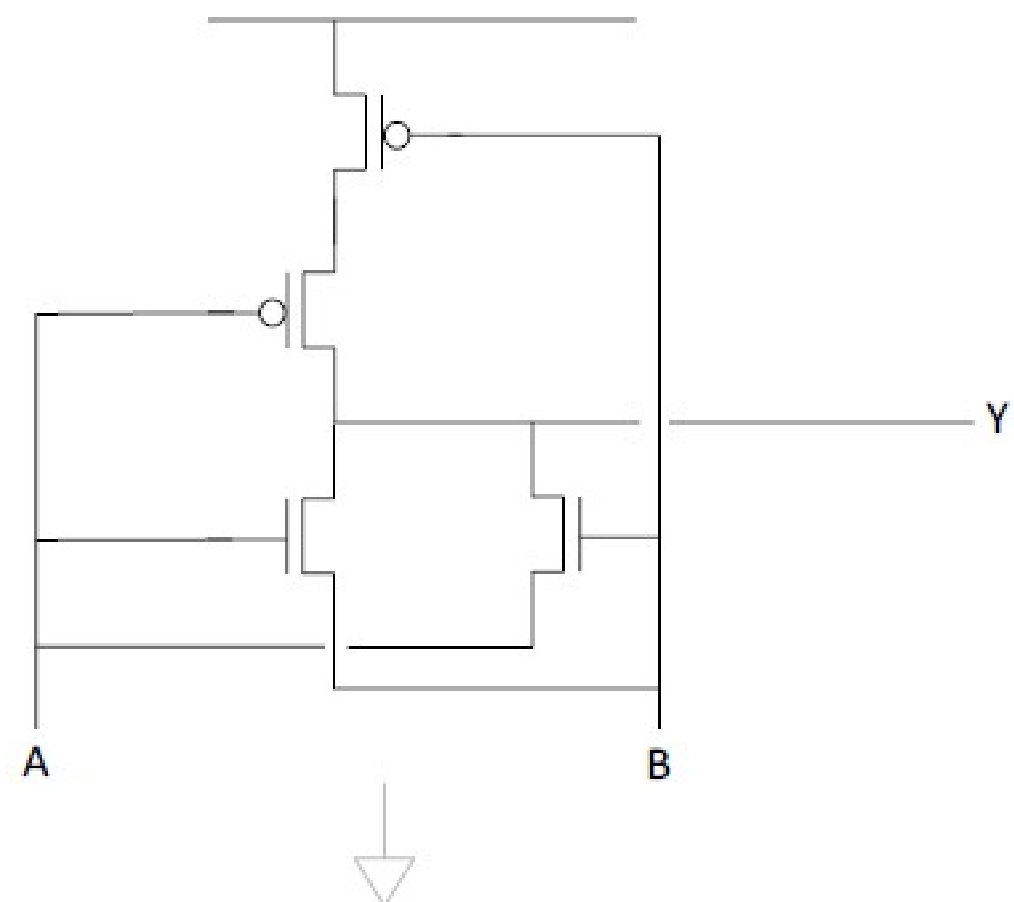


Resulting in:

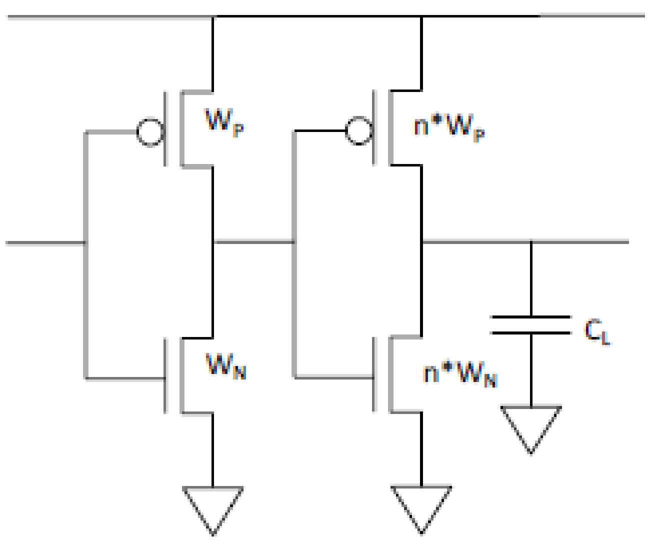


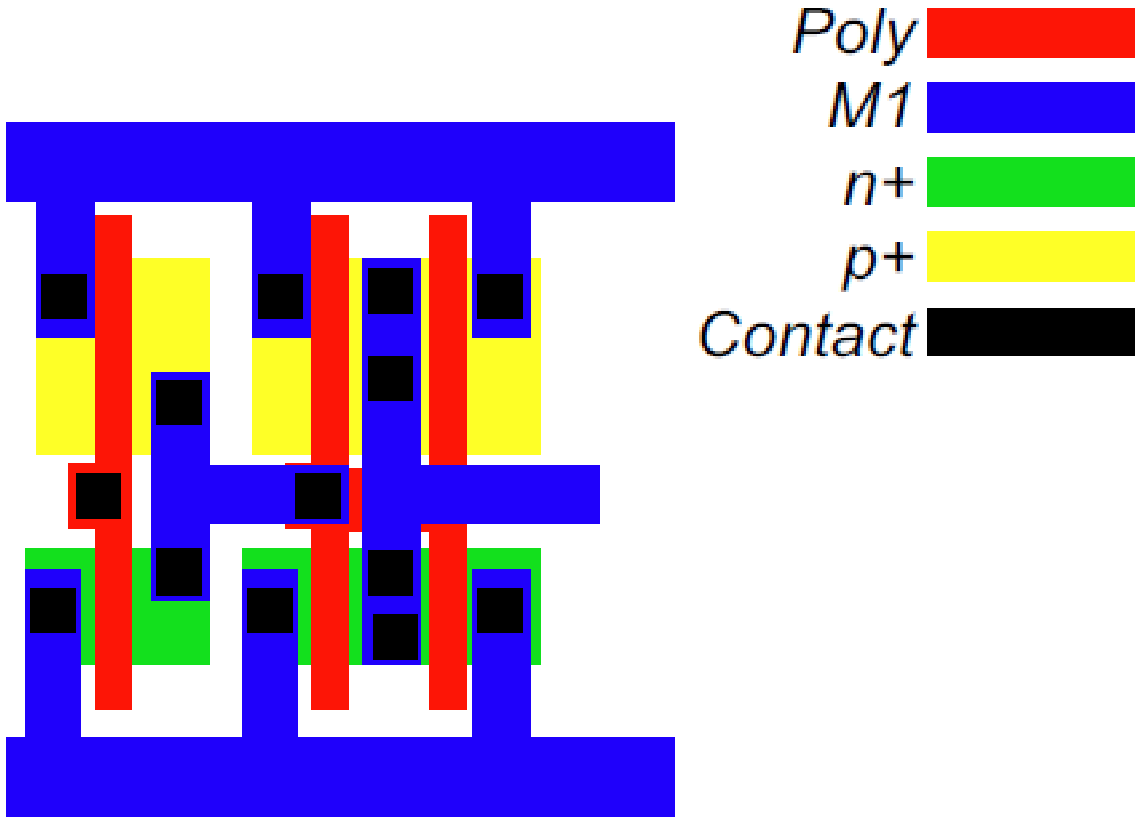
EX-NOR

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1



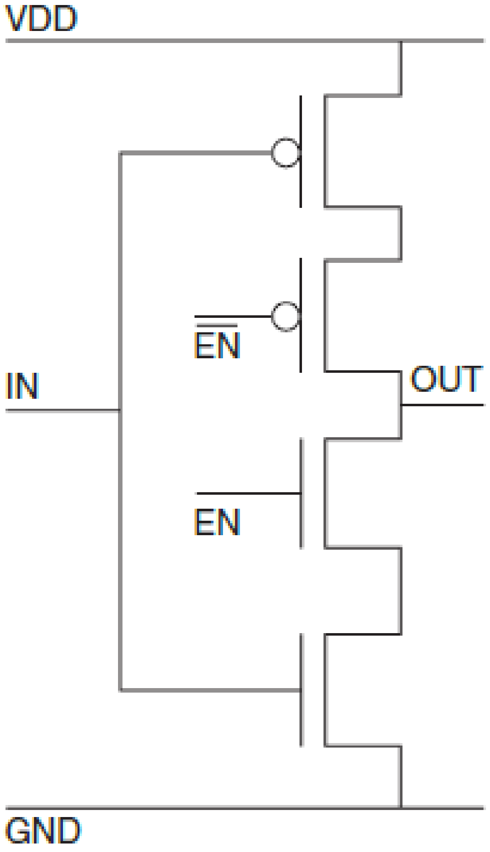
CMOS Buffer





→ allows **delays** in logic level propagation

CMOS Buffer Tristate

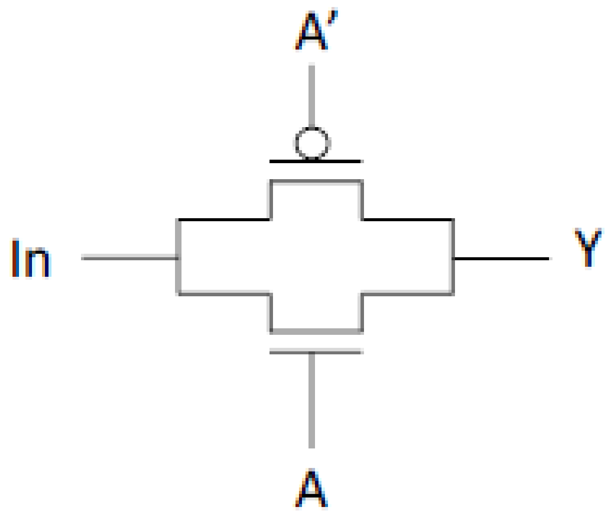


En	A	Y
0	0	?
0	1	?
1	0	1
1	1	0

→ allows **control** of logic level propagation

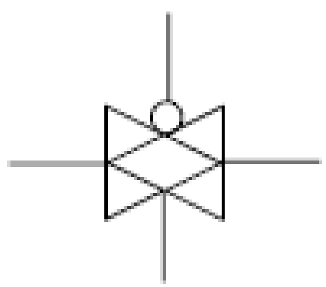
… 3rd type of output state: "disconnected" or "High Z"

Transmission Gate



In	A	Y
0	0	?
1	0	?
0	1	0
1	1	1

Pass Transistor

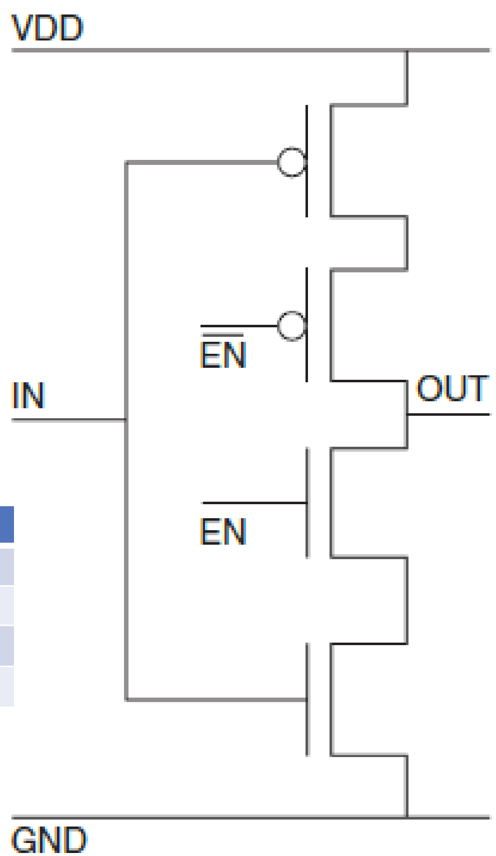


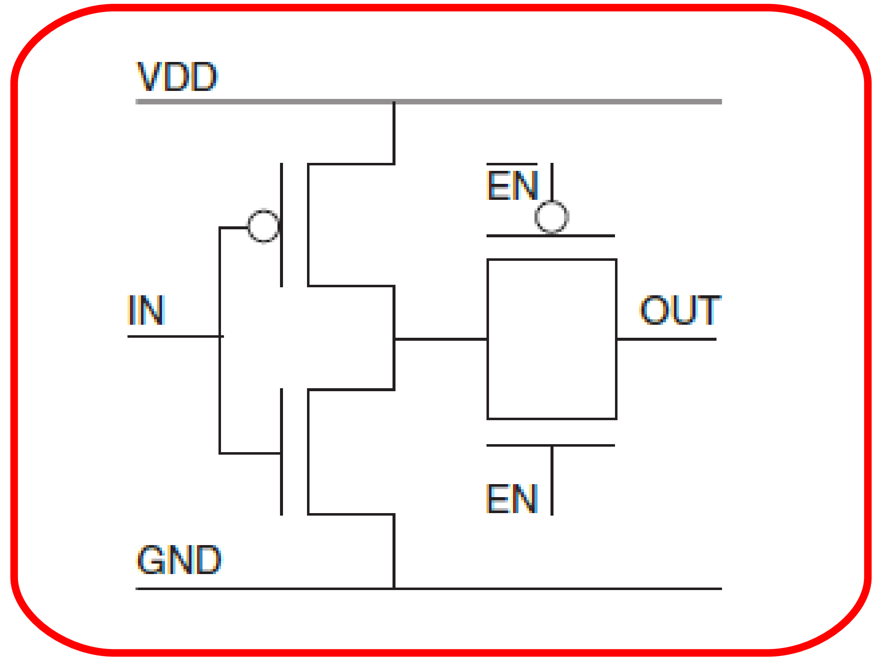
- complementary transistors in parallel
- controlled by complementary voltages
  VG to nFET
  VDD-VG to pFET
  (both transistors ON or both OFF)

→ voltage controlled switch

the CMOS Buffer Tristate

En	A	Y
0	0	?
0	1	?
1	0	1
1	1	0

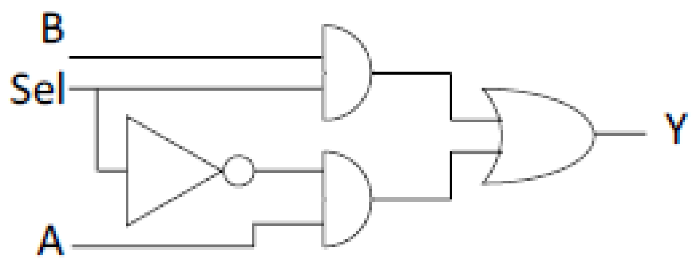




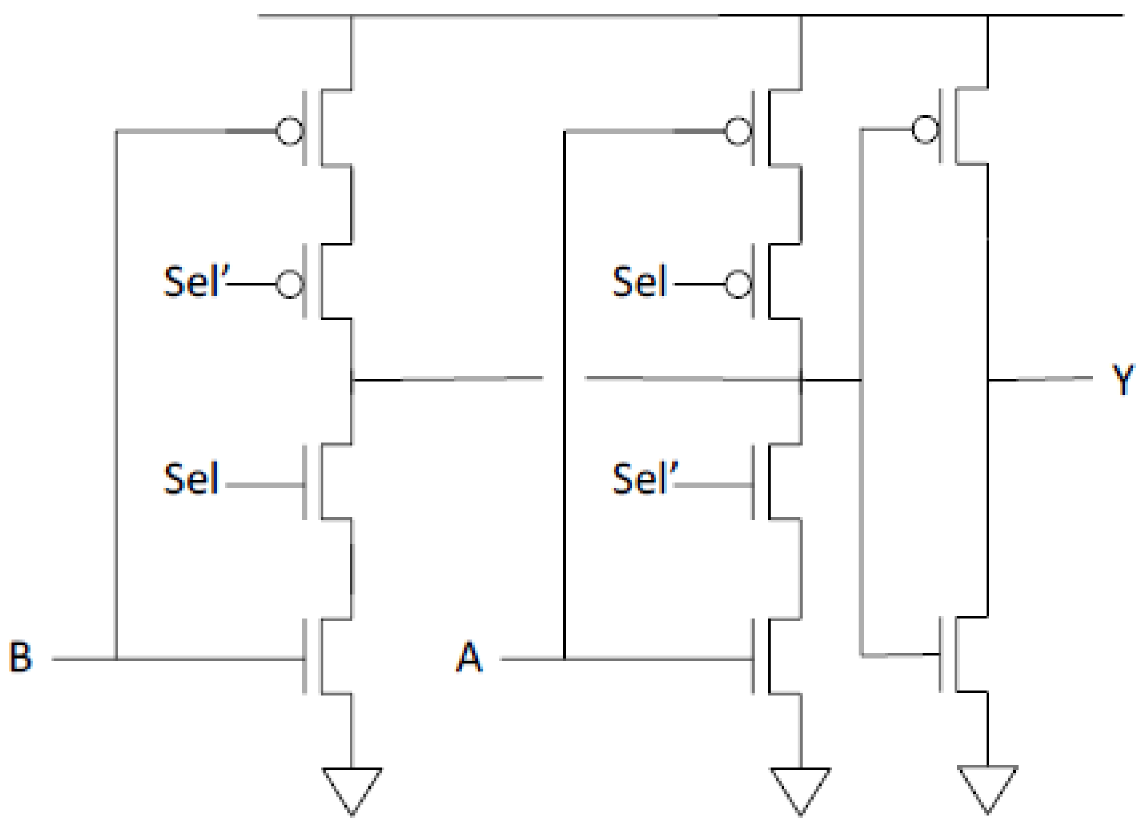
→ allows **control** of logic level propagation

… 3rd type of output state:
“disconnected“ or “High Z“

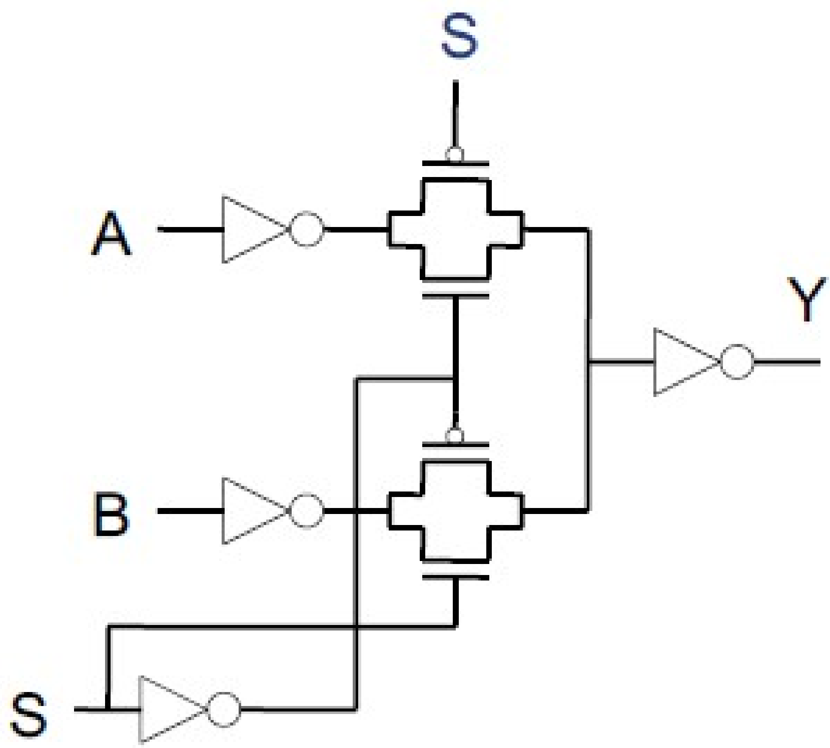
Multiplexer with gates

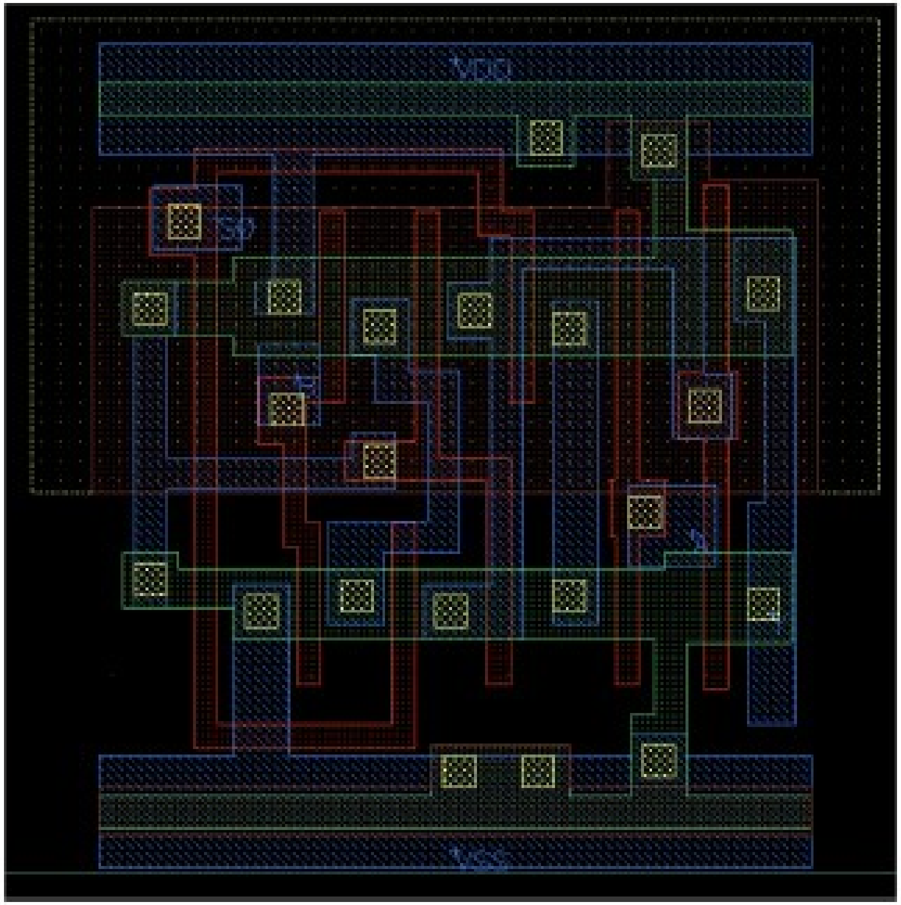


Sel	A	B	Y
0	0	x	0
0	1	x	1
1	x	0	0
1	x	1	1



MUX 2-inputs

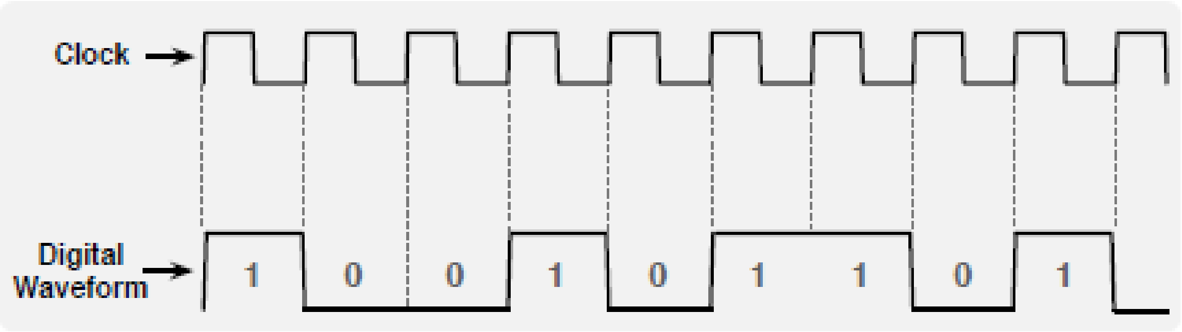




Timing in Digital Circuits

Logic levels propagation vs Time

→ Digital waveforms



The bits possibly change value at the clock rising or falling edge. For example, the sequence of bits 100101101 would give rise to the above digital waveform.

The clock of digital processors determines the speed of mathematical operations; algorithms employ a given number of clock cycles making the processing speed equal to the clock divided by that number.

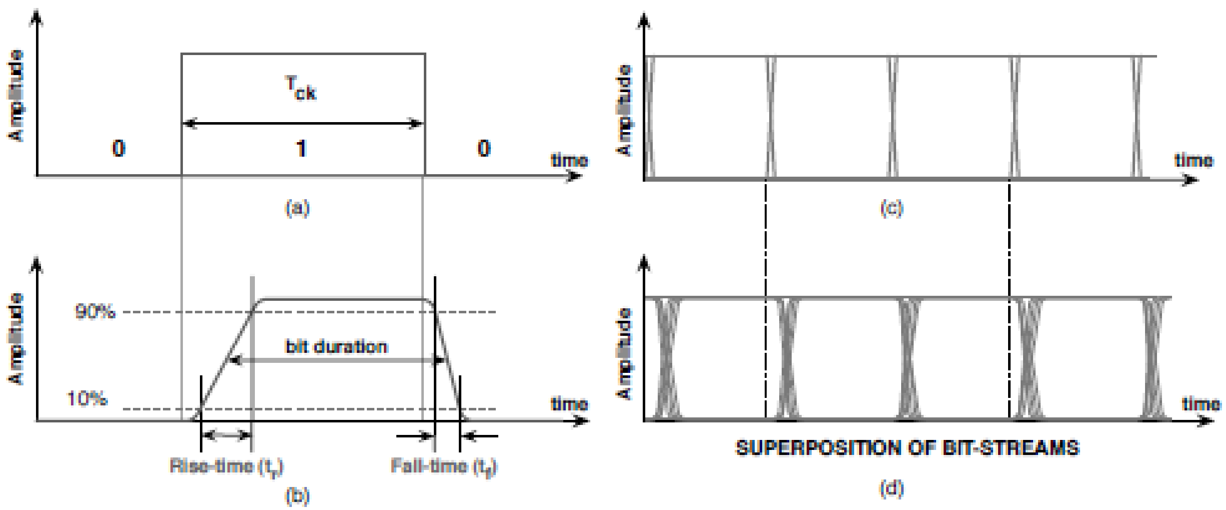
Well... life with Digital Circuits is more complicated...

Delays

Digital circuits have delays ! In addition to interconnections, all real electronic circuits introduces a delay with respect to input transitions with given rise and fall times. The delay caused by a logic cell is named propagation delay (typically it increases with chip temperature and supply voltage)

On the negative side, delays result in finite processing times and therefore set a limit on speed …

… but on the positive side, delays are essential for the existence of **Sequential Logic** circuits (flip-flops → memories and state machines → processors)

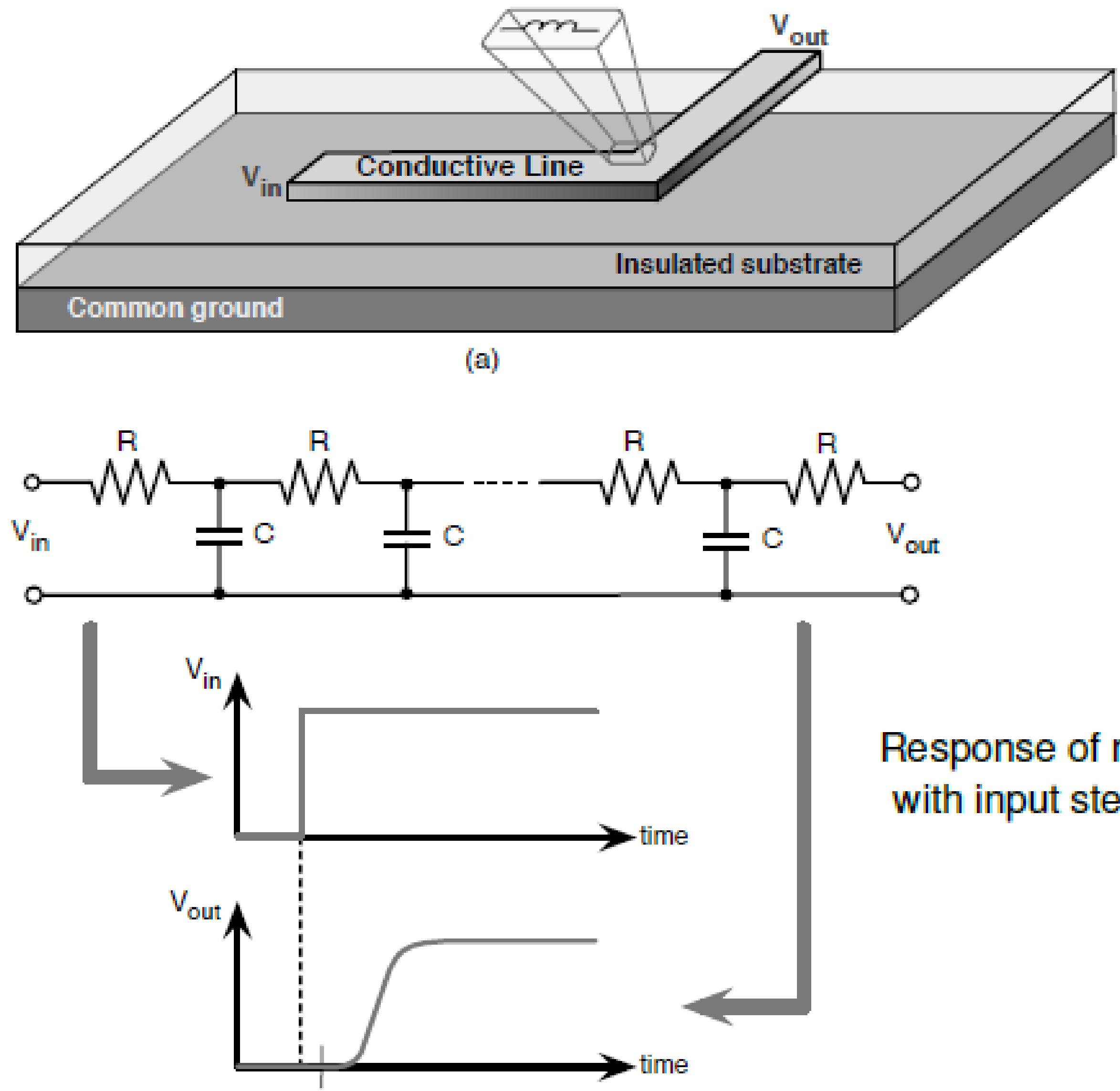


The output should changes at the rising edge of the clock and remains unchanged for the entire clock period.

In real cases, switching from “1” to “0” or vice versa is not instantaneous but occurs with some delay because of the finite speed of electronic circuits that make the rising and falling times are not zero.

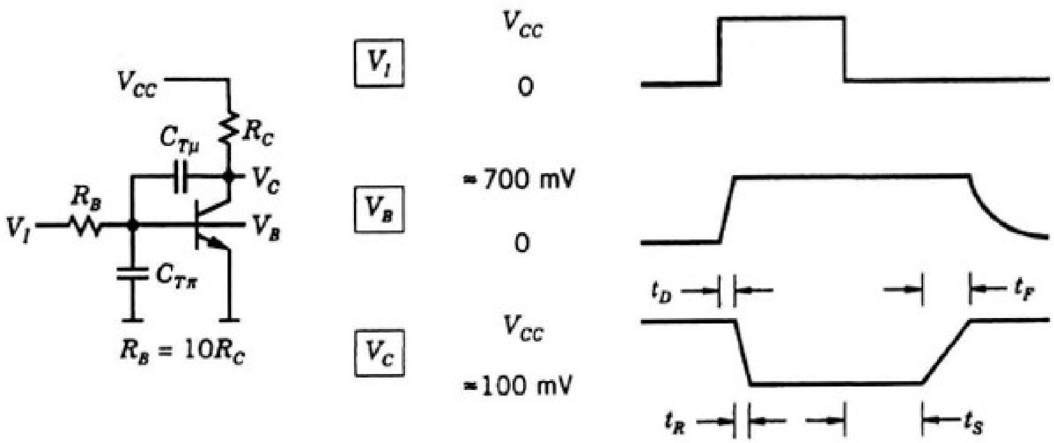
Interconnections → delays

The trace on a PCB or the interconnection of integrated circuits is a metal deposited onto an insulating material.

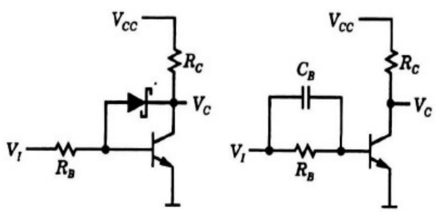


Response of resistive transmission line with input step signal.

Transistors Delays and Transition times



The step response has
a **delay** and then a **transition**
that can be described in terms
of a rise time or a fall time



(a) (b)

The transistors that define the dynamic response in logic gates are essentially in the common-emitter or common-source configuration: they switch back and forth between an off-state and an on-state (see figure)

The step response of logic gates is thus also characterized by delays and transition times. Transition times are important because they set conditions on the layout of digital circuits: for example, that ECL circuits must be interconnected with transmission lines in most cases

Gate delay

Assuming that questions of layout have been properly addressed,
we need not consider delay and transition times separately: the only time we really need to know is the time it takes for a transition at the input of a gate to become well reflected at the output so that it can have an effect on other gates (in case the conditions at the input are such that the output will change state)

This overall time or gate delay is equal to real delay + a fraction of transition time

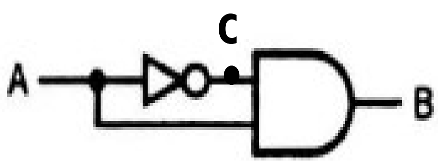
In the logic families we have described, the gate delay depends on the type of gate, but it does not depend much on the direction of the transition if the capacitive load on a gate is light

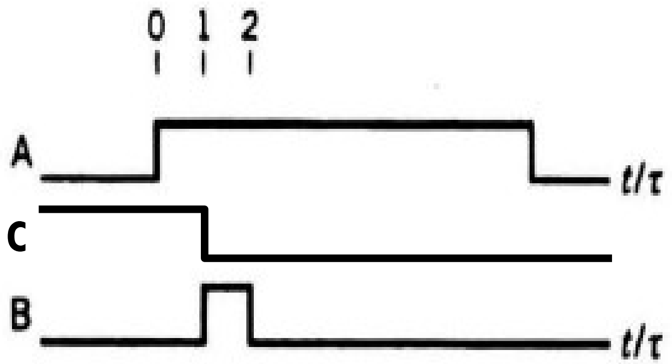
The gate delays of the basic gates are
- about 2 ns in the ECL family
- 10 ns in the LS-TTL family
- anywhere from 1 to 100 ns in the CMOS family

Gate delay - example

Basic model

• gate delays are independent of the direction of the transition,
→ describe the dynamics of logic gates in terms of a single time, the gate delay t
• take transition times to be zero so that we can locate events conveniently





If we consider only the static behavior of the circuit → B=0 always…

...but if we take into account the delays in the inverter and in the AND gate
→ B exhibits a pulse of width t when A goes from 0 to 1, and the leading edge of the pulse is delayed by t with respect to the leading edge of A

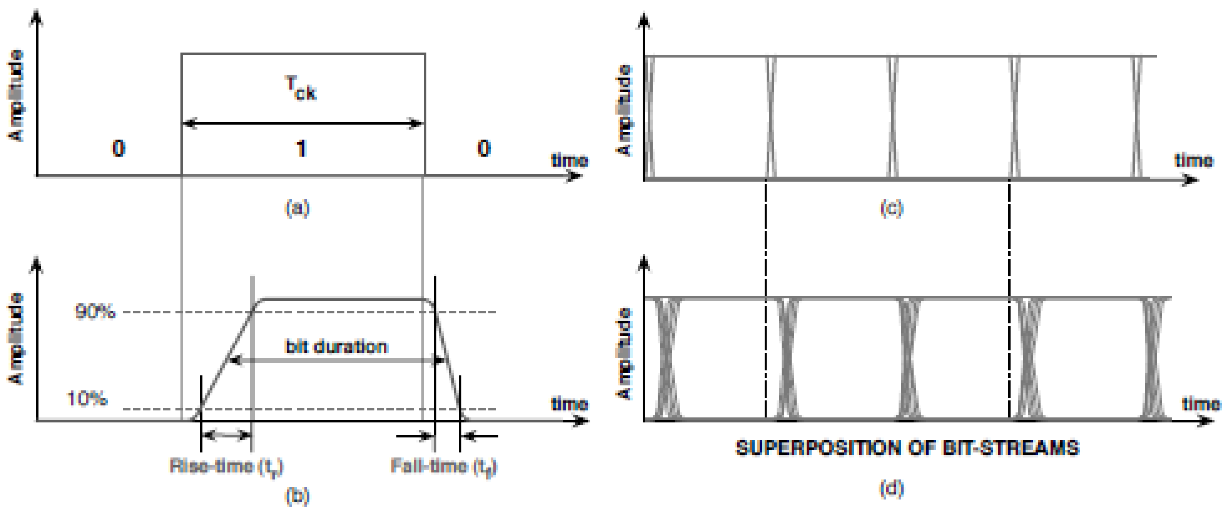
Note: gate delays can be profitably used to generate short pulses,
but they can also result in unintended pulses that make other circuits malfunction or, more generally, in a race condition in which a signal that is meant to block a gate before other inputs change does not arrive in time

Delays

Digital circuits have delays ! In addition to interconnections, all real electronic circuits introduces a delay with respect to input transitions with given rise and fall times. The delay caused by a logic cell is named propagation delay (typically it increases with chip temperature and supply voltage)

On the negative side, delays result in finite processing times and therefore set a limit on speed …

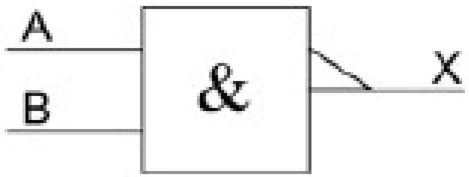
… but on the positive side, delays are essential for the existence of **Sequential Logic** circuits (flip-flops → memories and state machines → processors)

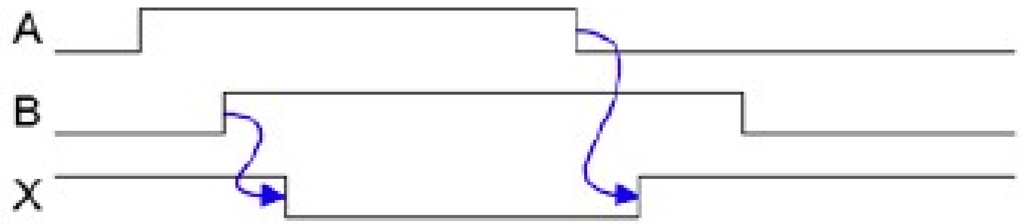


The output should changes at the rising edge of the clock and remains unchanged for the entire clock period.

In real cases, switching from "1" to "0" or vice versa is not instantaneous but occurs with some delay because of the finite speed of electronic circuits that make the rising and falling times are not zero.

Cause and Effect





Input B going high causes X to go low

Input A going low causes X to go high

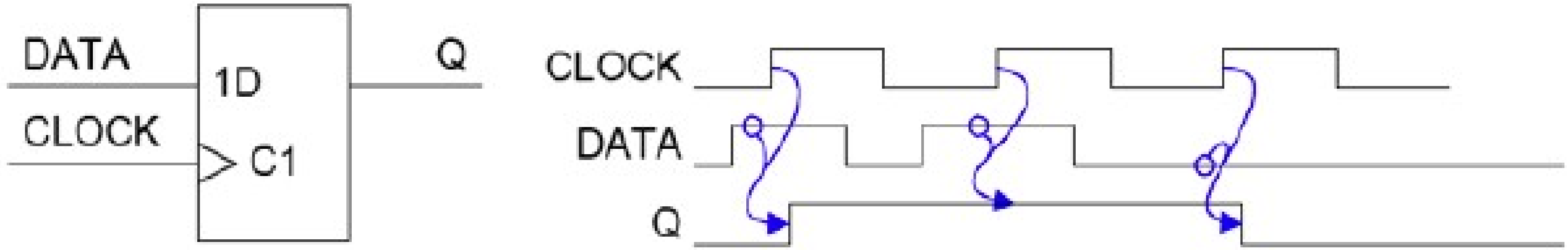
Propagation Delay:
The time delay between a cause (an input changing) and its effect (an output changing), assuming output load capacitance of 30pF.

Example: 74AC00: Advanced CMOS 2-input NAND gate

	min	typ	max	
A↑ to X↓ (t_{PHL})	1.5	4.5	6.5	ns
A↓ to X↑ (t_{PLH})	1.5	6.0	8.0	ns

t_{PHL} and t_{PLH} refer to the direction that the output changes: high-to-low or low-to-high.

D-Flip Flop



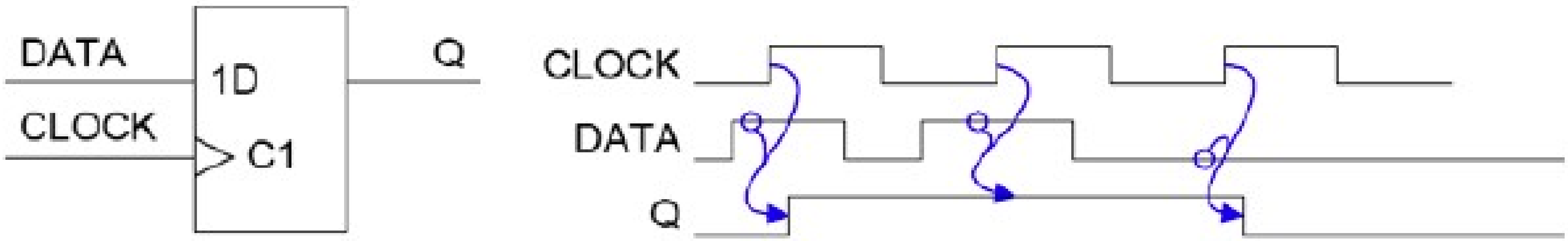
Notation:

>	input effect happens on the rising edge
C1	C ⇒ Clock input, 1 ⇒ This input *is* input number 1.
1D	D ⇒ Data input, 1 ⇒ This input *is controlled by* input number 1.

The meaning of a number depends on its position:

A number *after* a letter is used to identify a particular input.
A number *before* a letter means that this input is controlled by one of the other inputs.

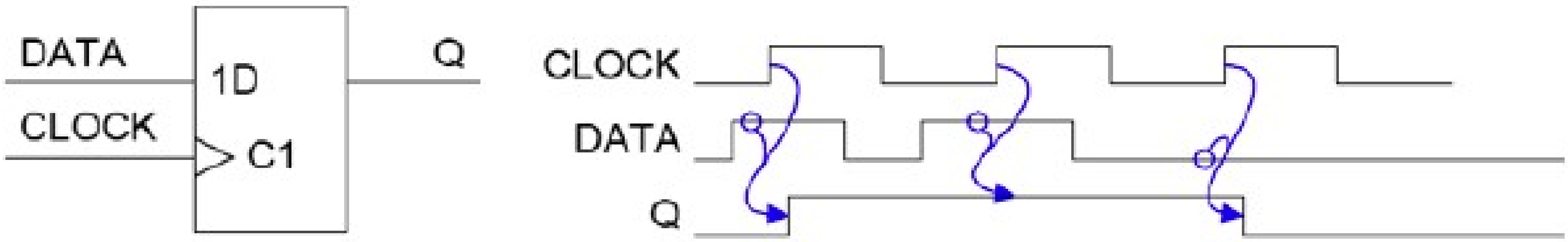
D-Flip Flop



Cause and Effect:

- CLOCK↑ causes Q to change after a short delay. This is the *only* time Q ever changes.
- The value of D *just before* CLOCK↑ is the new Q.
- Propagation delay CLOCK↑ to Q is typically 1 ns.
- Propagation delay DATA to Q **does not make sense** since DATA changing does not cause Q to change.

D-Flip Flop



Timing and delay parameters for flipflop is different from that with gates. Shown here is a D-FF that responses to a rising edge on the clock signal. A D-FF is like a camera, taking a "picture" from the scene (input is D). The clock input C1 is like the trigger on the camera – when pressed it samples the input and take a picture. The "cause" here is the rising edge of the CLOCK and the "effect" is the Q output sampling the D input, and keep the value until the next rising edge of the clock.

The delay here is from CLOCK rising edge to Q output changing. However, for the D-FF to work properly, there are two other timing parameters which are important: the **setup time** and the **hold time**. I will be talking about these in a later lecture.

Asynchronous vs Synchronous Operation

Asynchronous operation serves for diverse specific processing that do not require using a clock (mismatches between delays can impair the results).

More reliable is when timing is controlled by a clock (synchronous operation). The period must be such as to accommodate the most time consuming processing section.

Often, the circuit includes sections where processing is performed in multiple clock periods.

Sequential Operation

A sequential system relies on a sequence of actions, required to occur in the right order.

Sequential architectures can be synchronous and asynchronous.

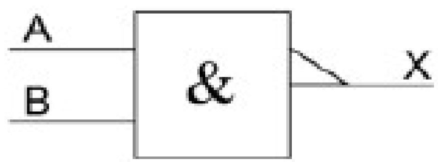
Use of a clock to establish the update times of the feedback signals or no time control the circuit sends back outputs continuously as soon they change value.

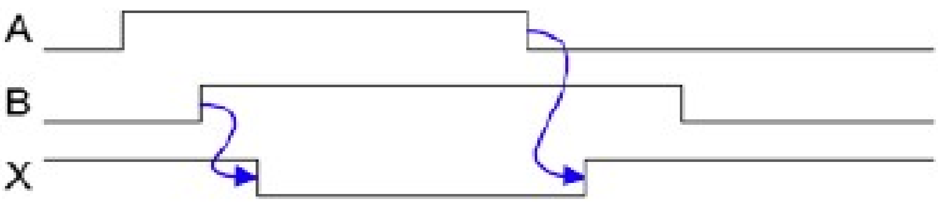
Sequential circuits, both synchronous and asynchronous architectures enable higher level processing than combinational schemes.

With given inputs possible instability. It persists until a different input configuration takes the output out of unstable conditions.

Combinatorial vs Sequential Operation

The task of combinational logics is just to relate signals at input and generate defined logic signals at output.





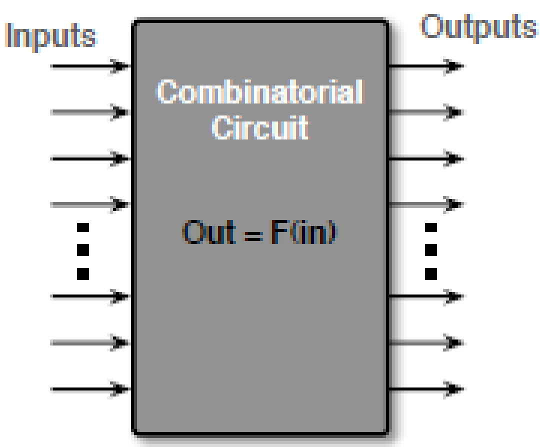
Input B going high causes X to go low

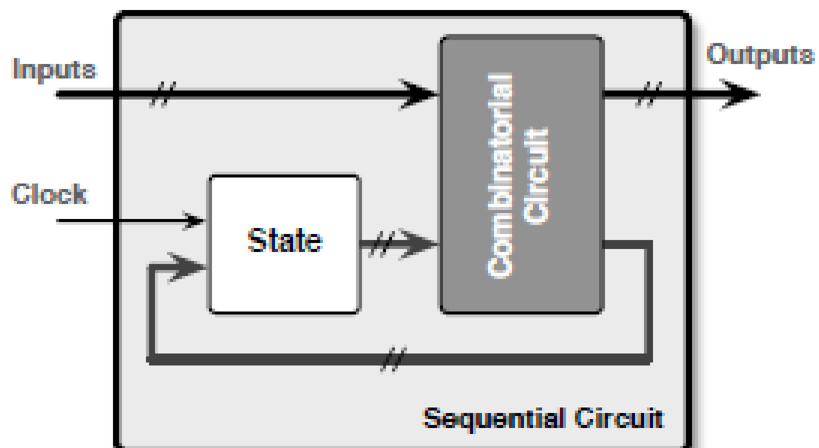
Input A going low causes X to go high

The output of combinational circuits depends only on input.

Sequential circuits produce output on the basis of both input and output.

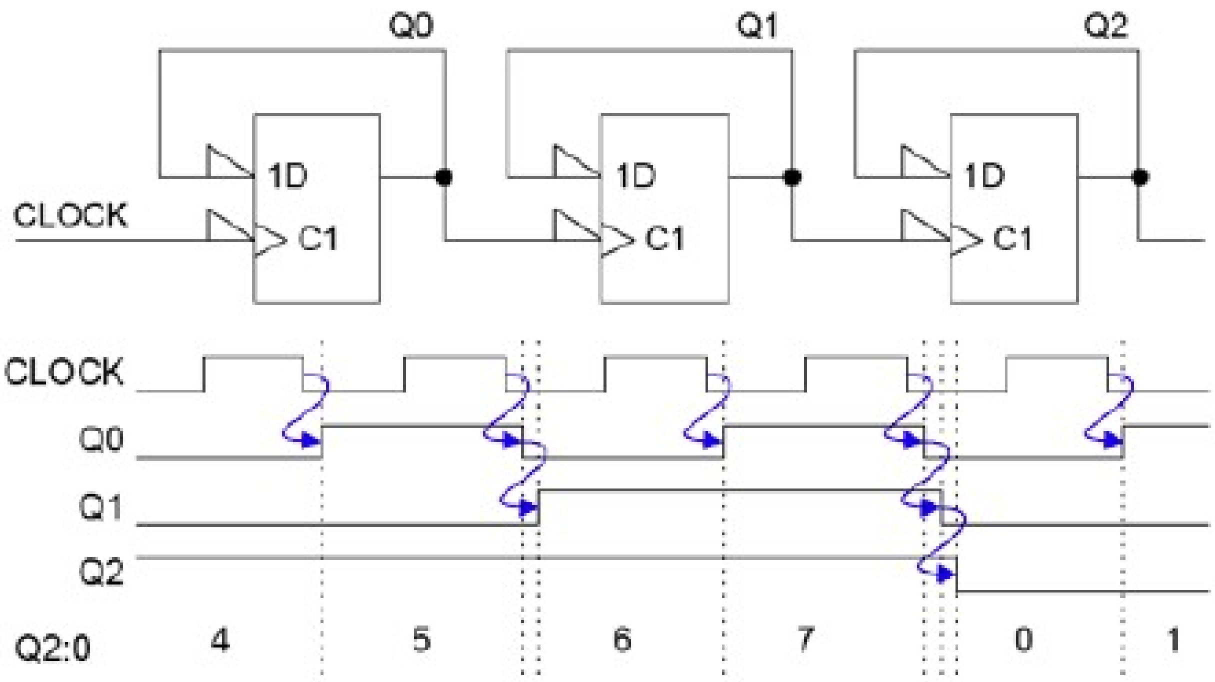
For example, a memory circuit is essentially sequential, because its output depends on the input that occurred in the past. Instead, the addition of two inputs is combinational because the result just depends on changing inputs.





Ripple Counter

example of a D-FF used in a ripple counter.

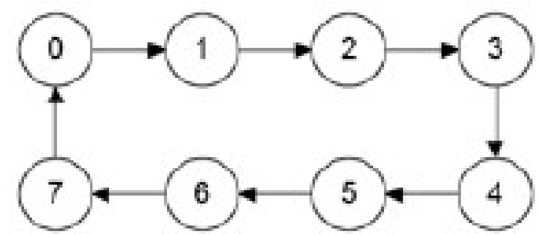


- Notice inverters on the CLOCK and DATA inputs
- Least significant bit of a number is always labelled 0

Propagation Delay: CLOCK↓ to Q2 = 3 x 1ns = 3ns

State Diagram

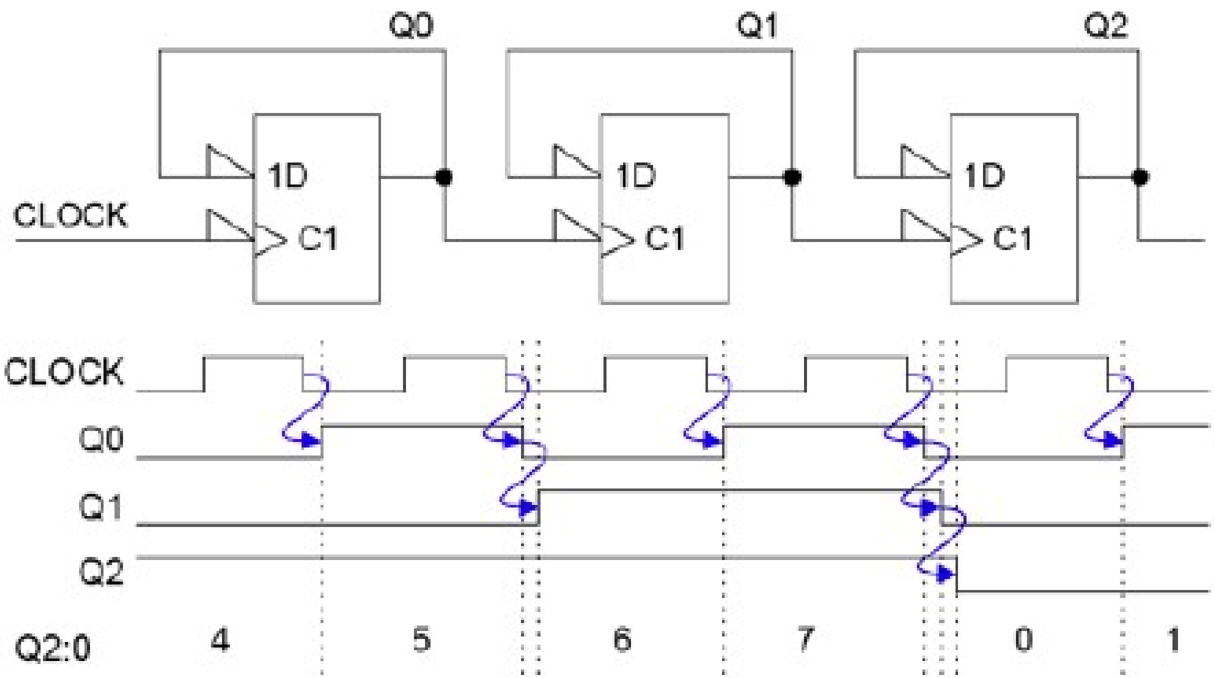
(not including transient states):



This counter is also known as an **asynchronous sequential circuit**. It is "**asynchronous**" because the output signals are NOT synchronised to a single clock signal (since there are many clock signals), and "**sequential**" because its current output value (or state) depends on previous output values in the sequence.

Ripple Counter

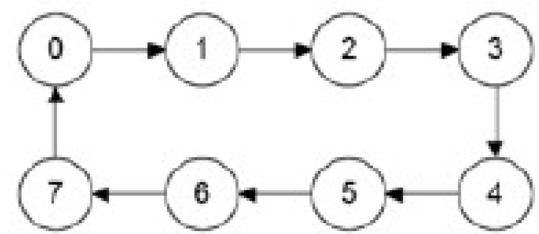
example of a D-FF used in a ripple counter.



- Notice inverters on the CLOCK and DATA inputs
- Least significant bit of a number is always labelled 0

State Diagram

(not including transient states):



Q0 value is first inverted (represented by the triangle) and then used as D input on the next clock cycle. The flipflop is triggered on the FALLING edge of CLOCK. Therefore the Q output "TOGGLES" on each active edge of the clock (i.e. falling edge). Q0 is therefore changing at half the rate of CLOCK, hence this flipflop acts as a divide-by-2 circuit.

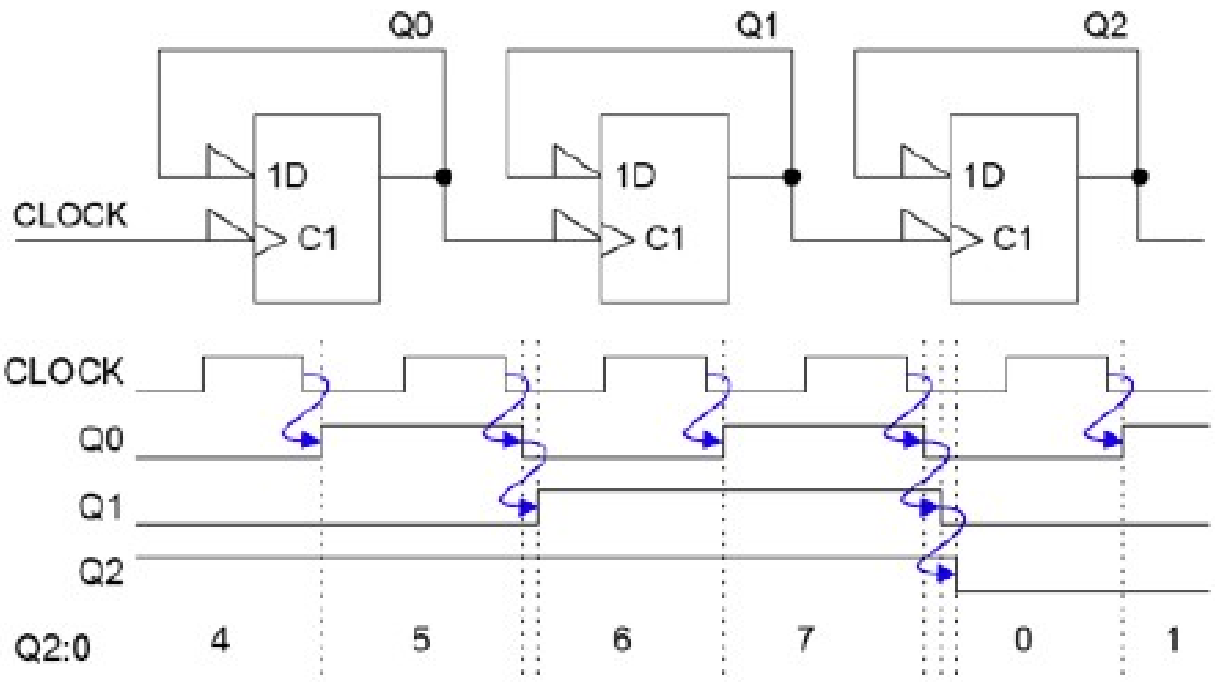
The Q0 signal is now used as clock input to the next D-FF. Hence Q1 is toggling at half the frequency of Q0. The circuit is effectively a binary counter.

This is a simple finite state machine (FSM) because it has 8 states which cycles through in a sequence. FSM will be covered in some later lectures in details and it is a very important topic in digital designs.

We then use the Q0 output as the clock input the next stage etc. Note that because the 2nd stage only starts to work once the first stage is completed, the propagation of effects "ripples" through the circuit – hence we call this a "ripple counter".

Ripple Counter

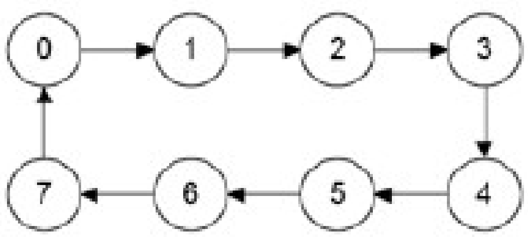
example of a D-FF used in a ripple counter.



- Notice inverters on the CLOCK and DATA inputs
- Least significant bit of a number is always labelled 0

State Diagram

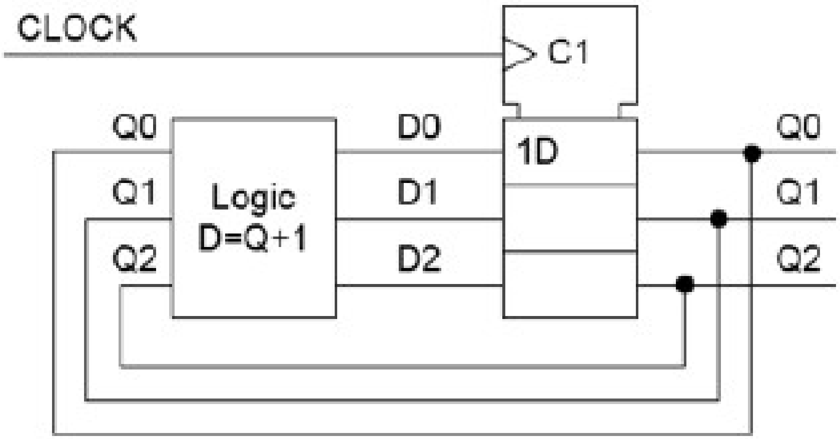
(not including transient states):



Propagation Delay: CLOCK↓ to Q2 = 3 x 1ns = 3ns

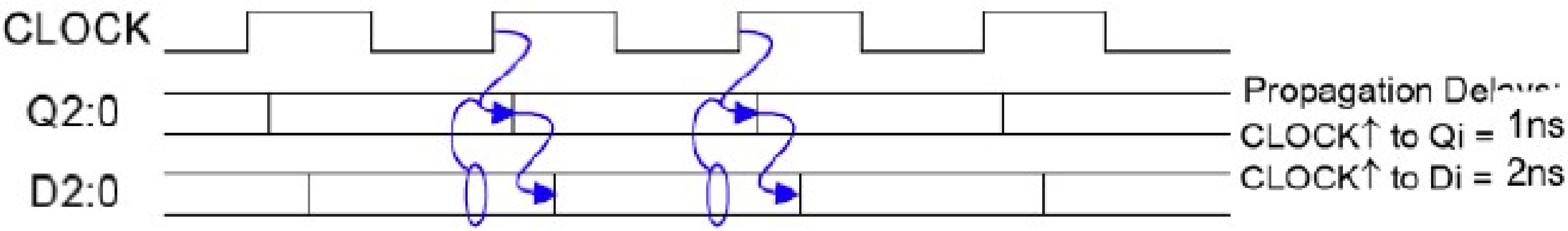
The ripple counter is potentially slow. The delay between the active edge of the clock and the counter output giving the correct value is dependent on the number of flipflops in the circuit and therefore the size of the counter (i.e. how many stages) .

Synchronous Counter



The logic block must add 1 onto the current value of the counter, Q, to generate the next value of the counter, D. Suppose it has a propagation delay of 1ns.

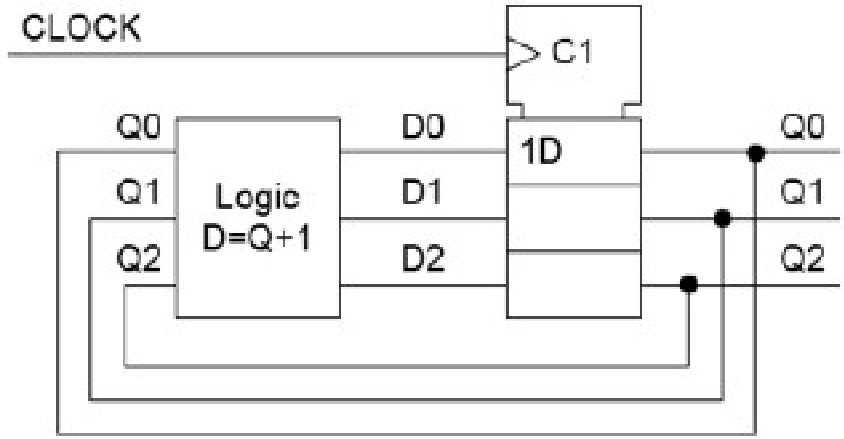
- A *register* is a bunch of flipflops with the same CLOCK.
- The individual flipflops are rectangles stacked on top of each other. Only the top one is labelled.
- All shared signals (e.g. the CLOCK input) go to the notched *common control block* at the top of the stack.



All flipflops change state within a fraction of nanosecond of each other.

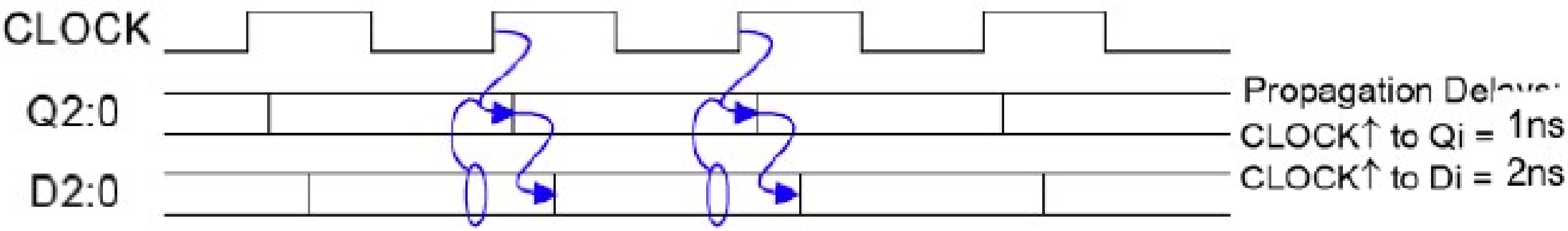
A far better approach is to use the flipflops TOGETHER as a group, and clock them using THE SAME CLOCK signal as shown here. The Logic Block is a combinatorial circuit which computes the next D value D2:0 from the current Q value Q2:0. (D has three bits D0, D1 and D2. We use the notation D2:0 to represent this.) The relationship between D and Q is simple: D2:0 = Q2:0 + 1.

Synchronous Counter



The logic block must add 1 onto the current value of the counter, Q, to generate the next value of the counter, D. Suppose it has a propagation delay of 1ns.

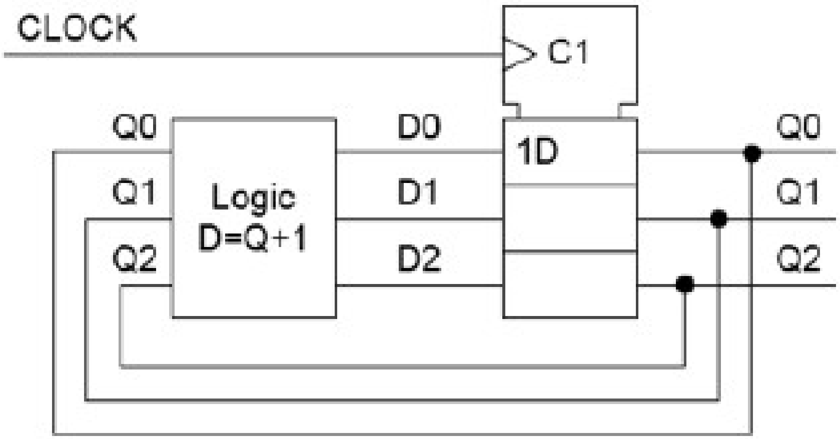
- A *register* is a bunch of flipflops with the same CLOCK.
- The individual flipflops are rectangles stacked on top of each other. Only the top one is labelled.
- All shared signals (e.g. the CLOCK input) go to the notched *common control block* at the top of the stack.



All flipflops change state within a fraction of nanosecond of each other.

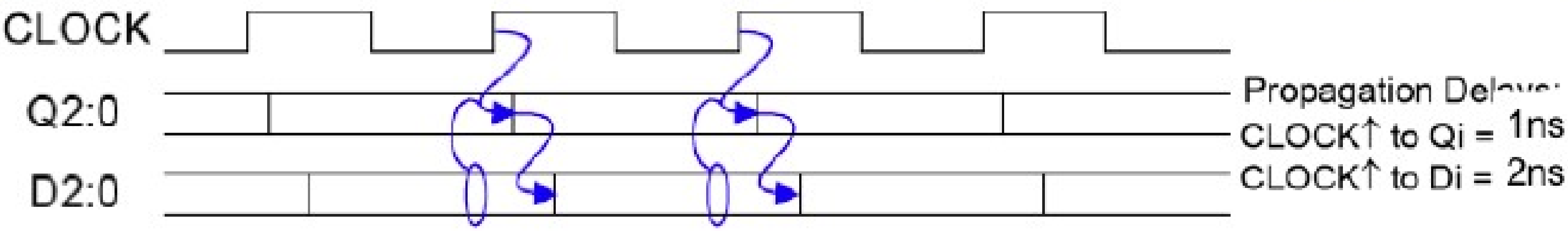
Since the three output bits Q2:0 change within a fraction of a nanosecond of each other, this circuit is: 1) faster than the ripple counter; 2) the "delay" is constant instead of dependent on the size of the counter.

Synchronous Counter



The logic block must add 1 onto the current value of the counter, Q, to generate the next value of the counter, D. Suppose it has a propagation delay of 1ns.

- A *register* is a bunch of flipflops with the same CLOCK.
- The individual flipflops are rectangles stacked on top of each other. Only the top one is labelled.
- All shared signals (e.g. the CLOCK input) go to the notched *common control block* at the top of the stack.

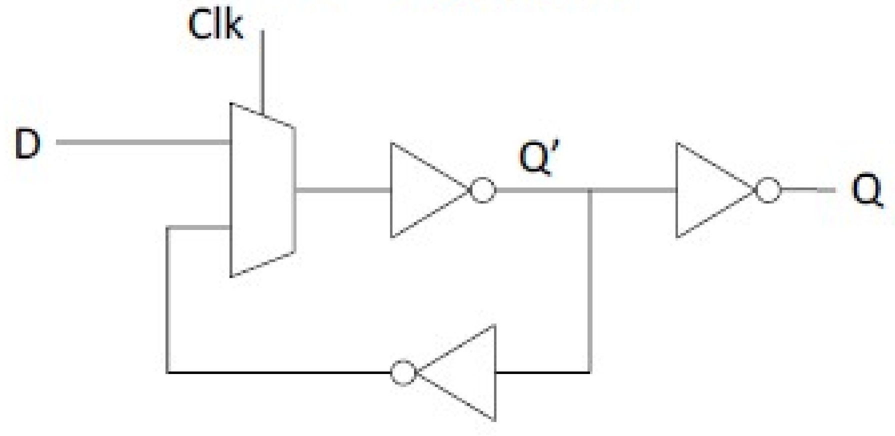


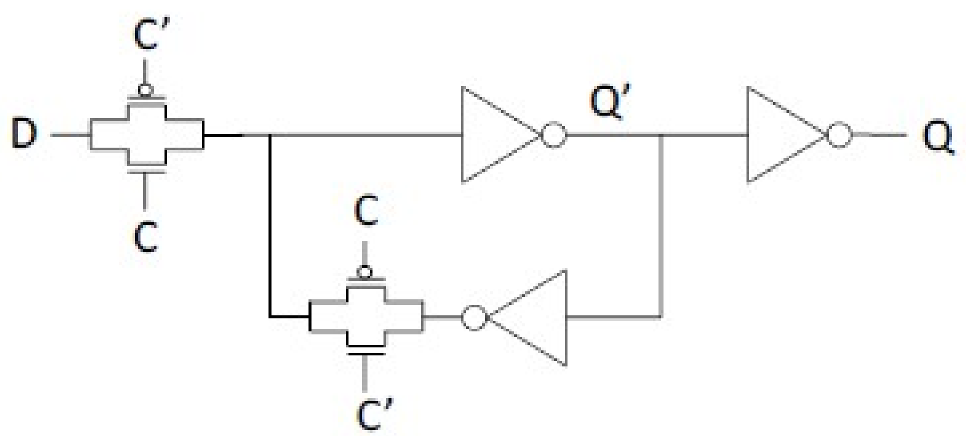
All flipflops change state within a fraction of nanosecond of each other.

This circuit is known as a **synchronous sequential circuit** because its function is synchronous to a single clock signal. If you regard the Q2:0 output value as a state value, it follows a finite number of states in a defined sequence. Therefore it is also a form of **Finite State Machine**.

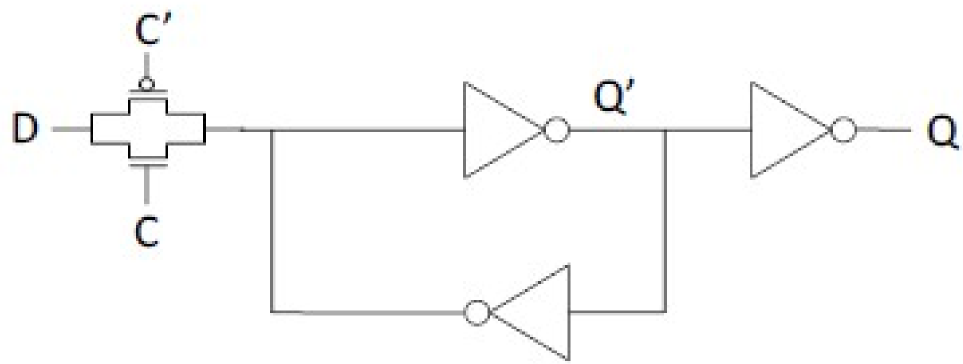
Additional Material

D-Latch





D-Latch (2)



FF