# Elettronica digitale

Giovanni Ambrosi

giovanni.ambrosi@pg.infn.it

Matteo Duranti

matteo.duranti@pg.infn.it

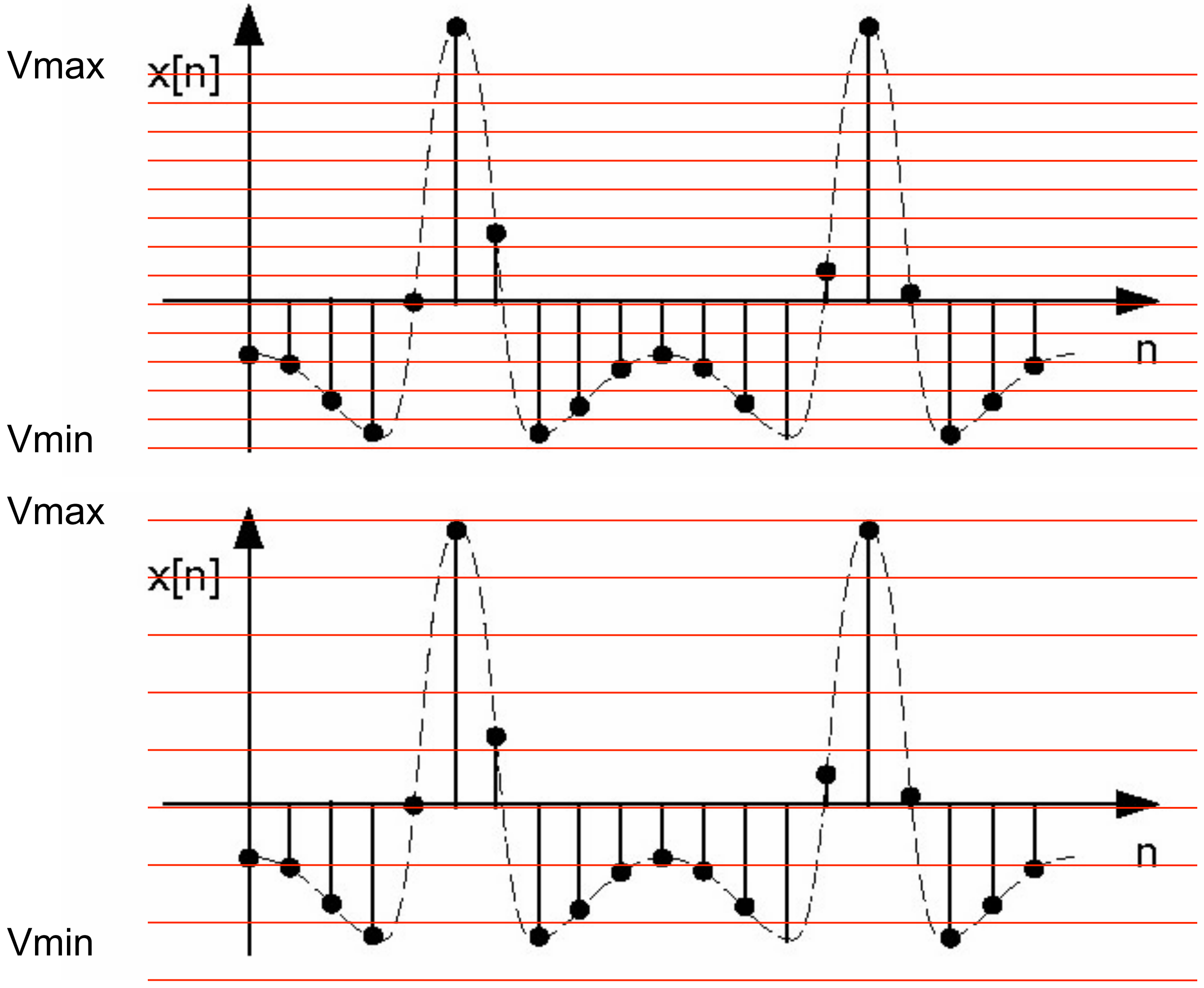
# ADC (I)

- Dal punto di vista funzionale gli ADC sono dei *classificatori:*
  - L'intervallo di variabilità del segnale $V_x$ viene diviso in *n* intervalli, detti *canali*, di ampiezza costante K. Definiamo quindi $V_i = K\,i + V_o$
  - Il segnale in ingresso $V_x$ viene *classificato* nel canale i-esimo se è verificata la relazione

$$V_{i-1} < V_x < V_i$$

  - Inevitabilmente si ha un errore di quantizzazione

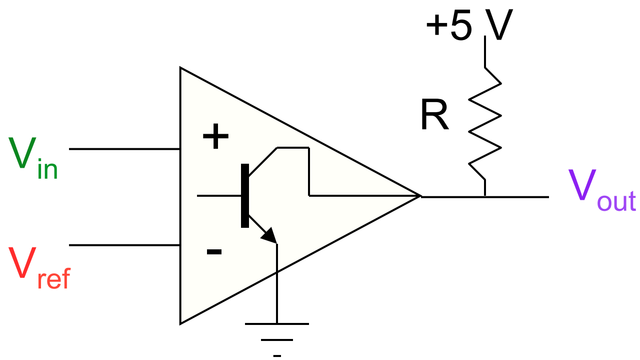
# ADC (2)

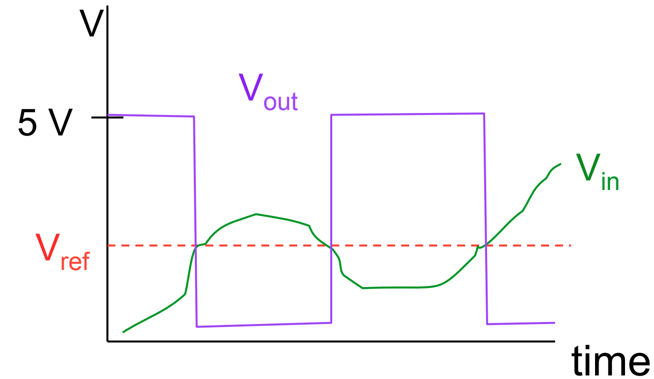


# Comparators

- It is very often useful to generate a strong electrical signal associated with some event
- If we frame the "event" in terms of a voltage threshold, then we use a comparator to tell us when the threshold is exceeded
  - could be at a certain temperature, light level, etc.: anything that can be turned into a voltage
- Could use an op-amp without feedback
  - set inverting input at threshold
  - feed test signal into non-inverting output
  - op-amp will rail (negative rail if test < reference; positive rail if test > reference)
- But op-amps have relatively slow "slew rate"
  - 15 V/μs means 2 μs to go rail-to-rail if powered ±15 V

# Enter the comparator



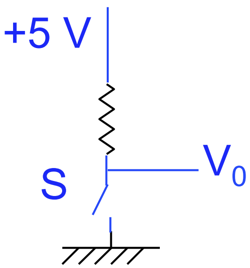

- When $V_{in} < V_{ref}$, $V_{out}$ is pulled high (through the pull-up resistor—usually 1 kΩ or more)
  - this arrangement is called "open collector" output: the output is basically the collector of an npn transistor: in saturation it will be pulled toward the emitter (ground), but if the transistor is not driven (no base current), the collector will float up to the pull-up voltage
- The output is a "digital" version of the signal
  - with settable low and high values (here ground and 5V)
- Comparators also good at turning a slow edge into a fast one
  - for better timing precision

# Elettronica digitale

continuo → discreto

analogico → digitale

Stati logici solo due possibili stati → 1, alto (H), vero (true) / 0, basso (L), falso (false)




Algebra booleana

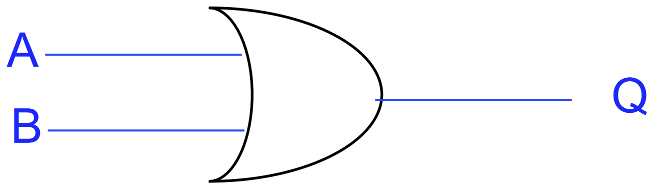
sistema matematico per l'analisi di stati logici

solo 3 funzioni logiche di base → **AND OR NOT** ← Funzioni logiche

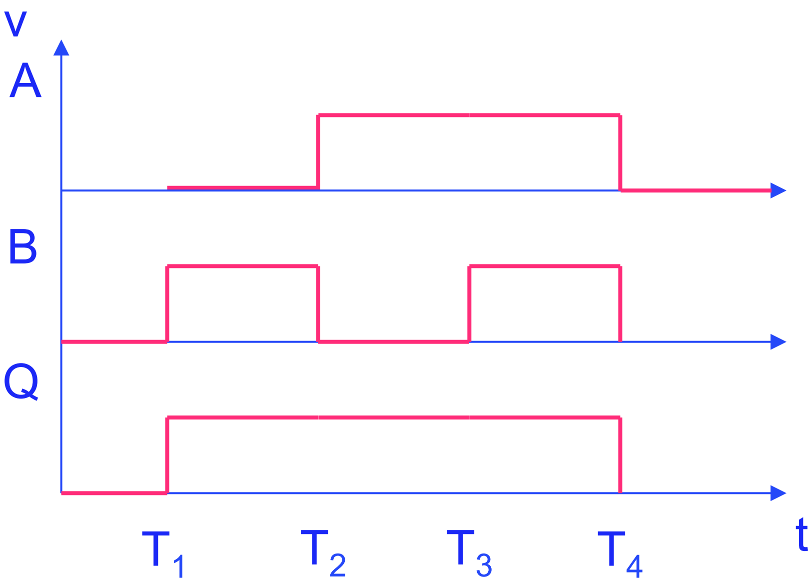
circuiti usati per la realizzazione di funzioni logiche → Porte logiche

# Porte logiche di base - OR

**OR**




$Q = A+B$




| A | B | Q |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

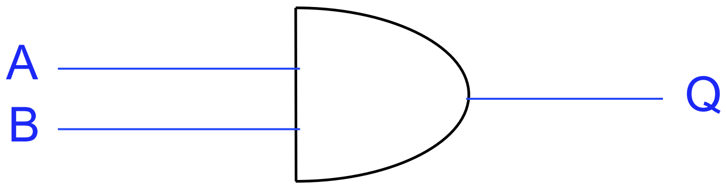
$$A+B+C = (A+B)+C = A+(B+C)$$

$$A+B = B+A$$

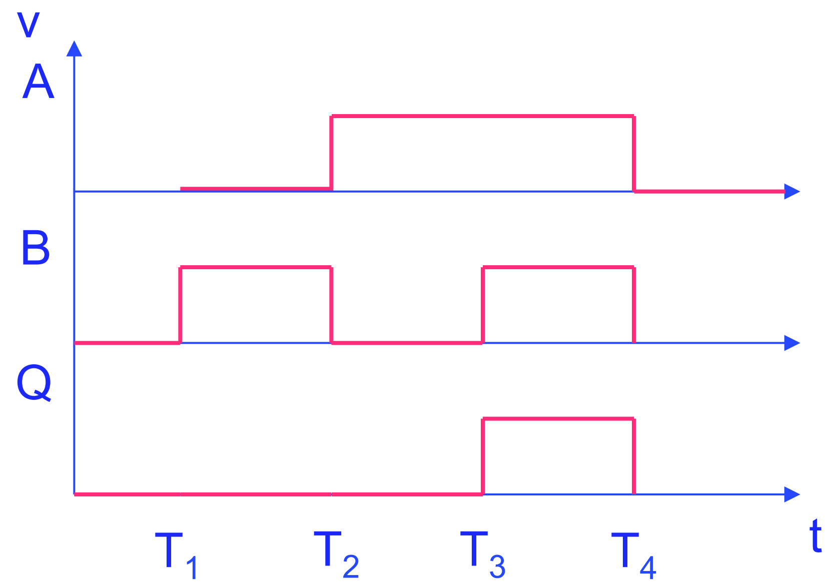
$$A+1 = 1,\ A+A = A,\ A+0 = A$$

# Porte logiche di base - AND

**AND**




$Q = A \cdot B$




| A | B | Q |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

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

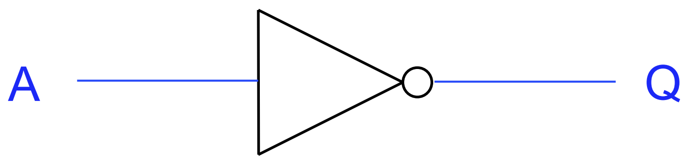
$$A \cdot B = B \cdot A$$

$$A \cdot 1 = A, A \cdot A = A, A \cdot 0 = 0$$

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

# Porte logiche di base - NOT

**NOT**




| A | Q |
|---|---|
| 0 | 1 |
| 1 | 0 |

$$\overline{\overline{A}} = A$$

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

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

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

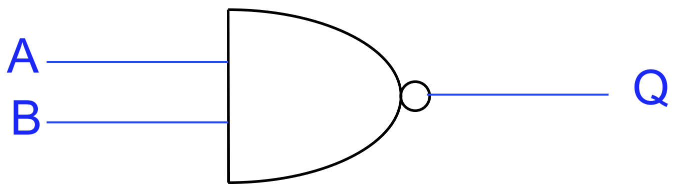
sapendo che

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

$$A + \overline{A} \cdot B = A \cdot (B + 1) + \overline{A} \cdot B = A \cdot B + A + \overline{A} \cdot B =$$
$$(A + \overline{A}) \cdot B + A = B + A = A + B$$

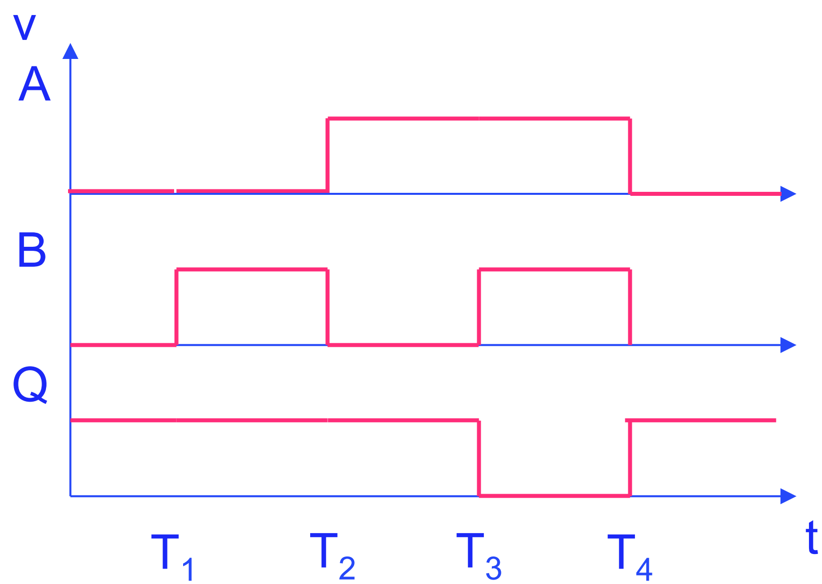
# Porte logiche di base – NAND

**NAND**




$$Q = \overline{A \cdot B}$$

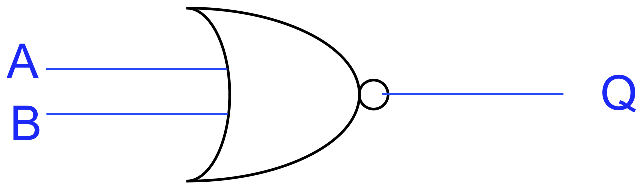
| A | B | Q |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |




porta universale

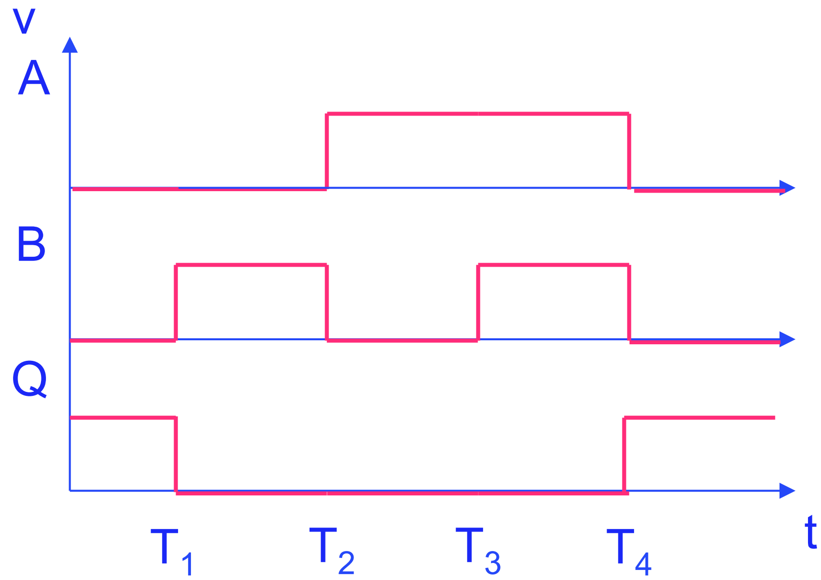
# Porte logiche di base – NOR

**NOR**




$$Q = \overline{A + B}$$

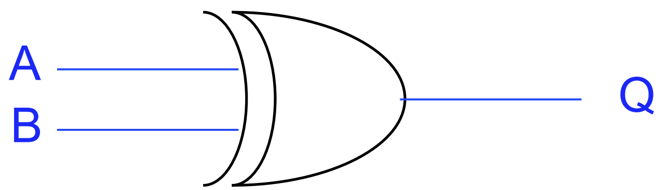
| A | B | Q |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |



# Porte logiche di base – XOR

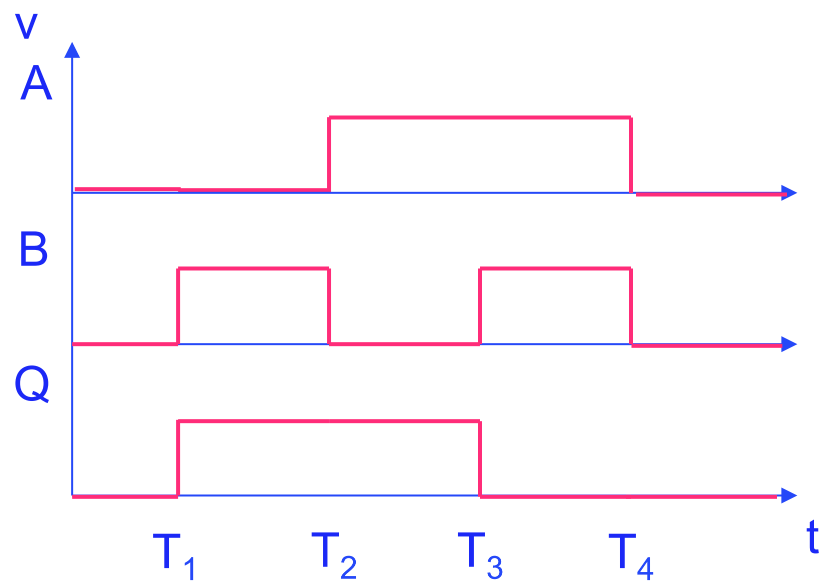
**XOR**

OR esclusivo



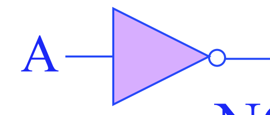

$Q = A \oplus B$

| A | B | Q |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |



# Data manipulation

- All data manipulation is based on *logic*
- Logic follows well defined rules, producing predictable digital output from certain input
- Examples:




NOT

| A | C |
|---|---|
| 0 | 1 |
| 1 | 0 |

AND

| A | B | C |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

OR

| A | B | C |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

XOR

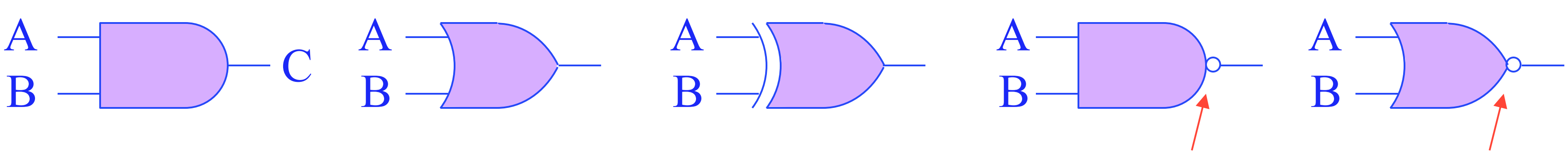
| A | B | C |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

NAND

| A | B | C |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

NOR

| A | B | C |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |




bubbles mean inverted (e.g., NOT AND → NAND)

# Algebra Booleana

Algebra booleana — trasformare una funzione logica in un'altra di più facile implementazione hardware

Teoremi di De Morgan

$$\overline{A \cdot B \cdot C \cdot \ldots\ldots\ldots} = \overline{A} + \overline{B} + \overline{C} + \ldots\ldots\ldots$$

$$\overline{A + B + C + \ldots} = \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot \ldots\ldots\ldots$$

Il complemento dell'AND di più variabili logiche è dato dall'OR dei complementi

Il complemento dell'OR di più variabili logiche è dato dall'AND dei complementi

# Algebra Booleana

| A | B | C | $f(A,B,C)$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

## Prima forma canonica (esempio)

$$f(A,B,C) = (\overline{A}\cdot\overline{B}\cdot\overline{C}) + (\overline{A}\cdot B\cdot C) + (A\cdot B\cdot\overline{C}) + (A\cdot B\cdot C)$$

Ogni riga come prodotto (AND) dei termini naturali (se 1) o complementati (se 0)
Somma (OR) delle righe con valore pari a 1.

## Seconda forma canonica (esempio)

$$f(A,B) = (A+B)(\overline{A}+B)(\overline{A}+\overline{B})$$

Ogni riga come somma (OR) dei termini naturali (se 1) o complementati (se 0)
Prodotto (AND) delle righe con valore pari a 0.

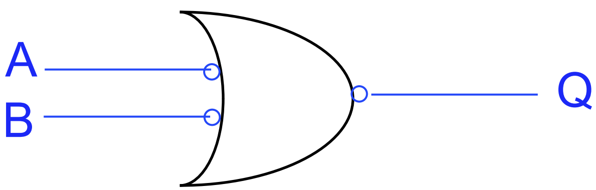
| A | B | $f(A,B)$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

# Algebra Booleana

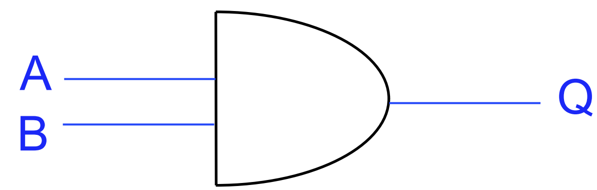
Un circuito AND per logica positiva funziona come un OR per logica negativa

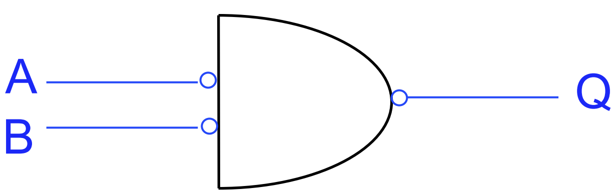
non è necessario usare i tre circuiti di base

bastano due → OR e NOT oppure AND e NOT

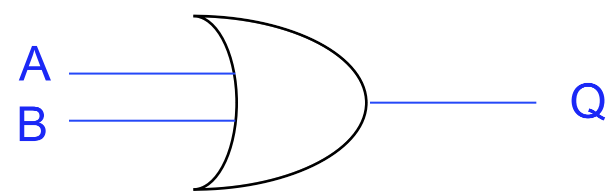



$$\overline{\overline{A}+\overline{B}} \Leftrightarrow A\cdot B$$



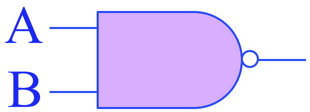

$$\overline{\overline{A}\cdot\overline{B}} \Leftrightarrow A+B$$



# All Logic from NANDs Alone

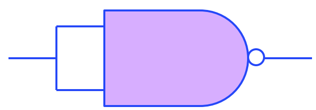
## NAND

| A | B | C |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |




## NOT

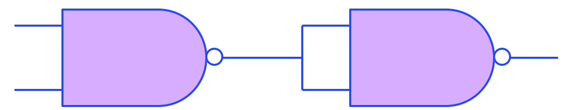
| A | C |
|---|---|
| 0 | 1 |
| 1 | 0 |



## AND

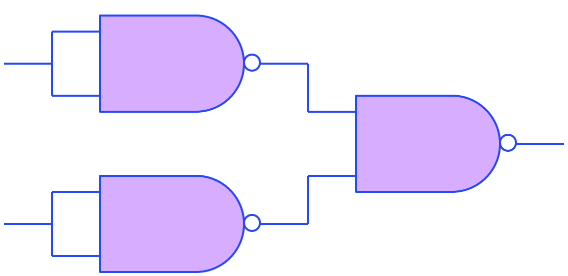
| A | B | C |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

invert output (invert NAND)



## OR

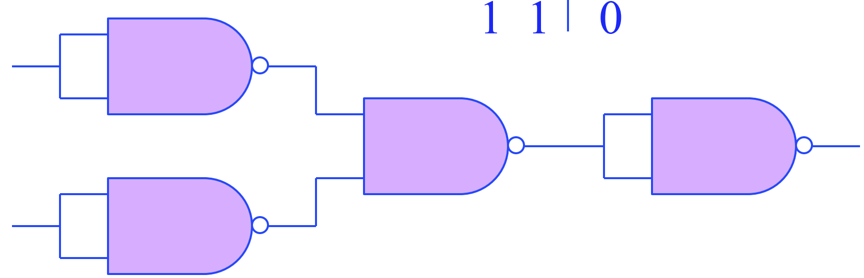
| A | B | C |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |



invert both inputs

## NOR

| A | B | C |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |



invert inputs *and* output (invert OR)

# Famiglie logiche

Famiglie logiche più diffuse e usate

- **CMOS** (Complementary MOS)
- **NMOS** (MOSFET a canale n)

} transistor **FET**

- **TTL** (Transistor-Transistor Logic)
- **ECL** (Emitter Coupled Logic)

} transistor **BJT**

Le porte logiche possono essere fabbricate con le varie tecnologie in un singolo chip con stesse funzioni, compatibili

numero di porte →

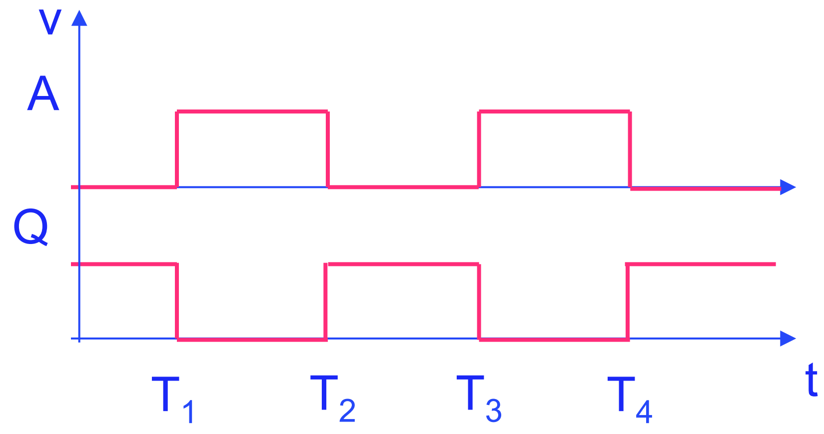
SSI small scale integration (1-10 gates)
MSI medium scale integration (10-100 gates)
LSI large scale integration (~ $10^3$)
VLSI very large scale integration (~ $10^6$)
ULSI ultra large scale integration (> $10^6$)

# Invertitore

La porta logica più semplice da realizzare è l'invertitore (NOT)

**invertitore ideale**

1) transizione istantanea
2) potenza dissipata nulla
3) stato di uscita determinato solo dallo stato di ingresso




Può essere realizzato in una delle diverse famiglie logiche

NMOS permette la maggiore densità di componenti
transistor MOSFET utilizzati sia come interruttori che come resistenze

→ minimizzazione dell'area occupata

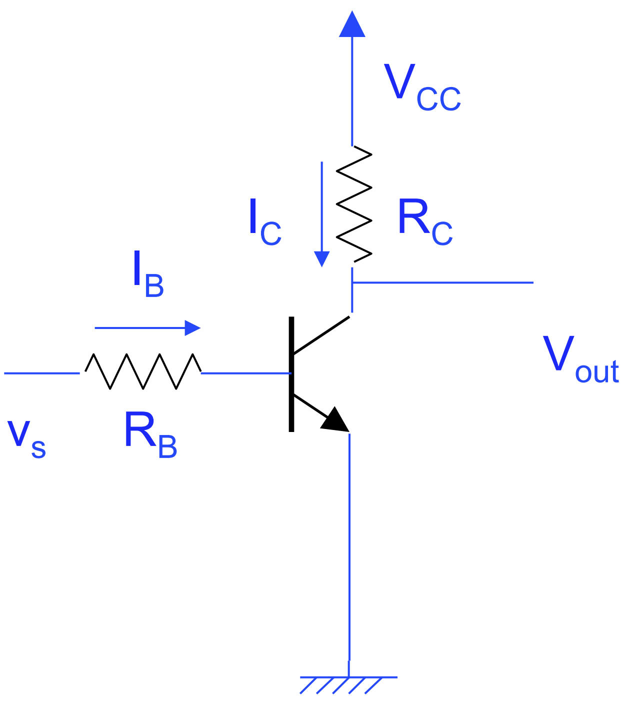
# Logic Families

- TTL: transistor-transistor logic: BJT based
  - chips have L, LS, F, AS, ALS, or H designation
  - output: logic high has $V_{OH} > 3.3$ V; logic low has $V_{OL} < 0.35$ V
  - input: logic high has $V_{IH} > 2.0$ V; logic low has $V_{IL} < 0.8$ V
  - dead zone between 0.8V and 2.0 V
    - nominal threshold: $V_T = 1.5$ V
- CMOS: complimentary MOSFET
  - chips have HC or AC designation
  - output: logic high has $V_{OH} > 4.7$ V; logic low has $V_{OL} < 0.2$ V
  - input: logic high has $V_{IH} > 3.7$ V; logic low has $V_{IL} < 1.3$ V
  - dead zone between 1.3V and 3.7 V
    - nominal threshold: $V_T = 2.5$ V
  - chips with HCT are CMOS with TTL-compatible thresholds

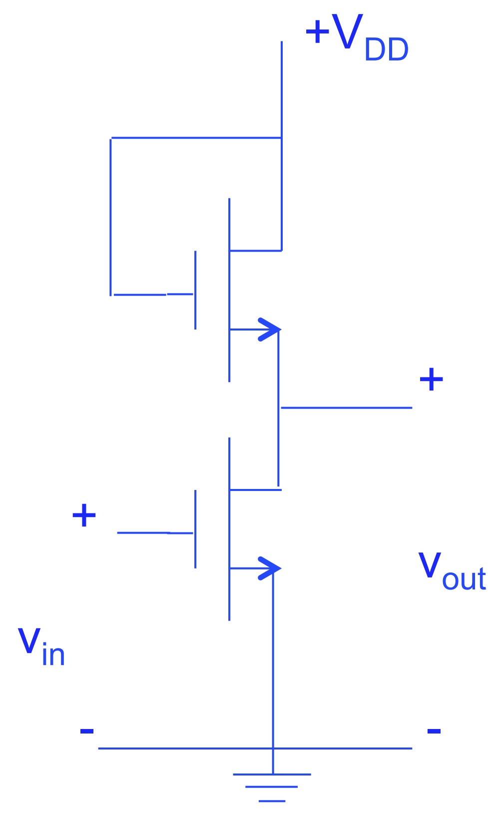
# Invertitore

Realizzazione: è di fatto un interruttore

logica TTL (BJT)

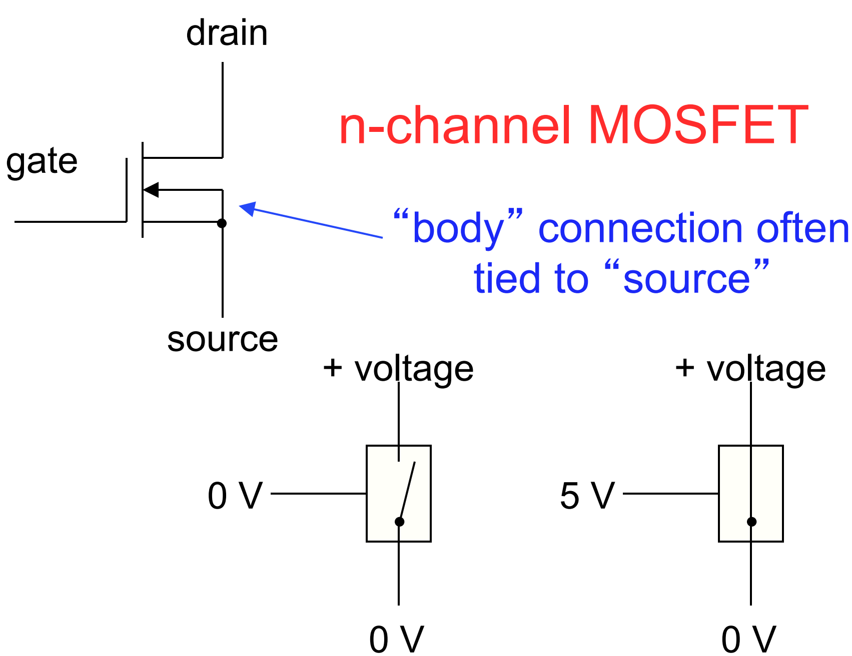


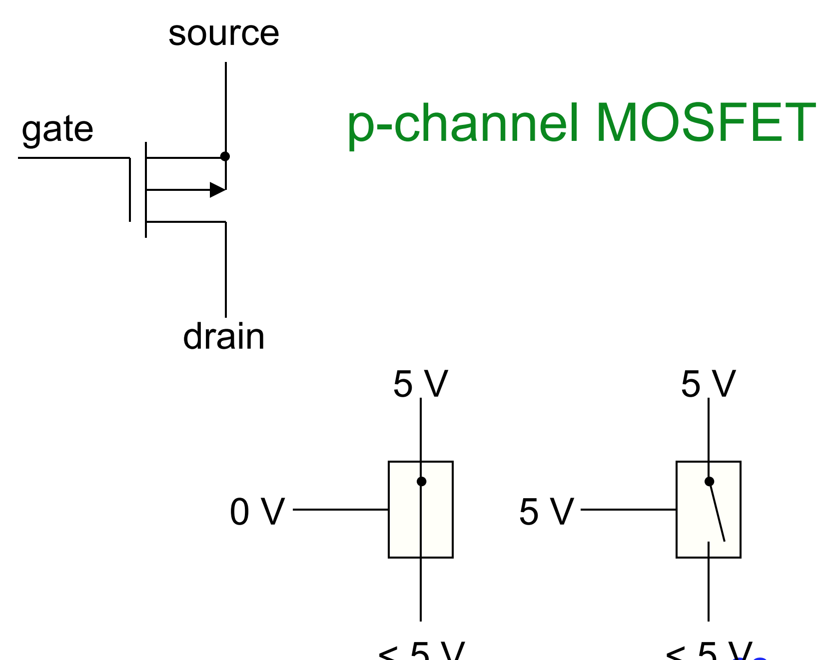

logica NMOS (MOSFET)



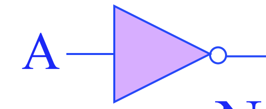
# MOSFET Switches

- MOSFETs, as applied to logic designs, act as voltage-controlled switches
  - n-channel MOSFET is closed (conducts) when positive voltage (+5 V) is applied, open when zero voltage
  - p-channel MOSFET is open when positive voltage (+5 V) is applied, closed (conducts) when zero voltage
    - (MOSFET means metal-oxide semiconductor field effect transistor)



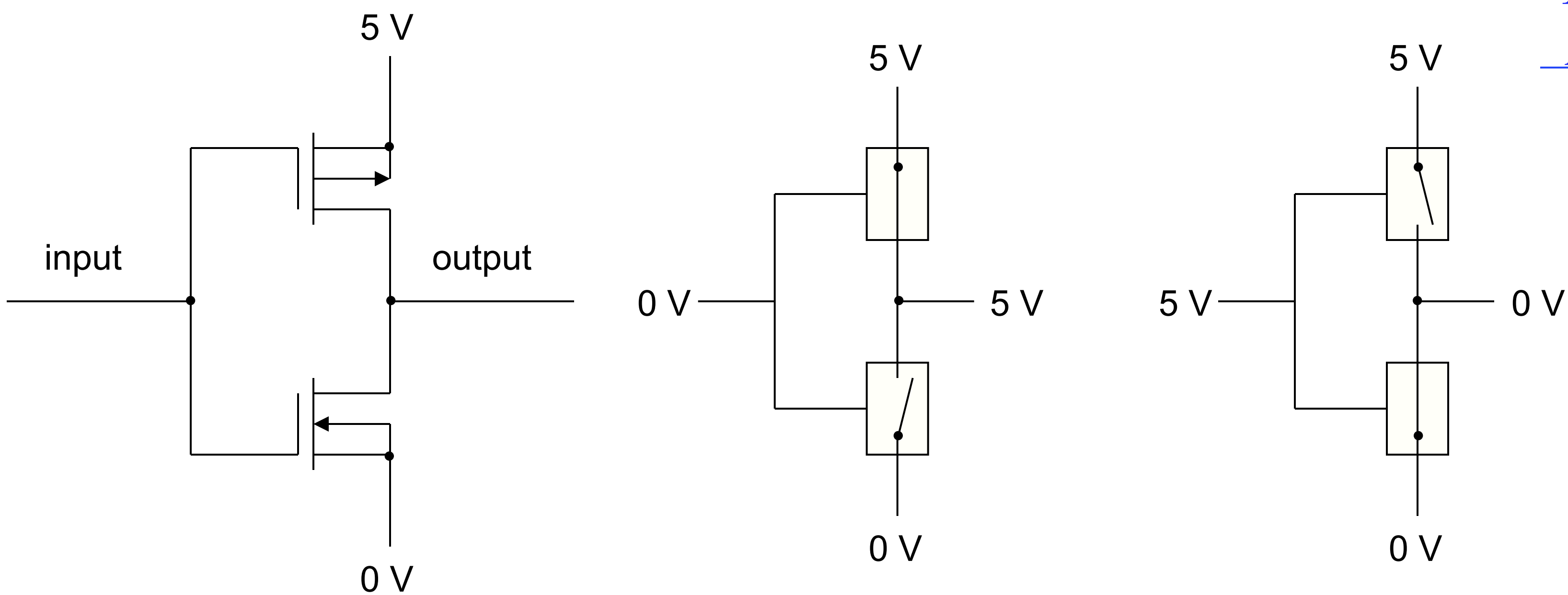



# An inverter (NOT) from MOSFETS:



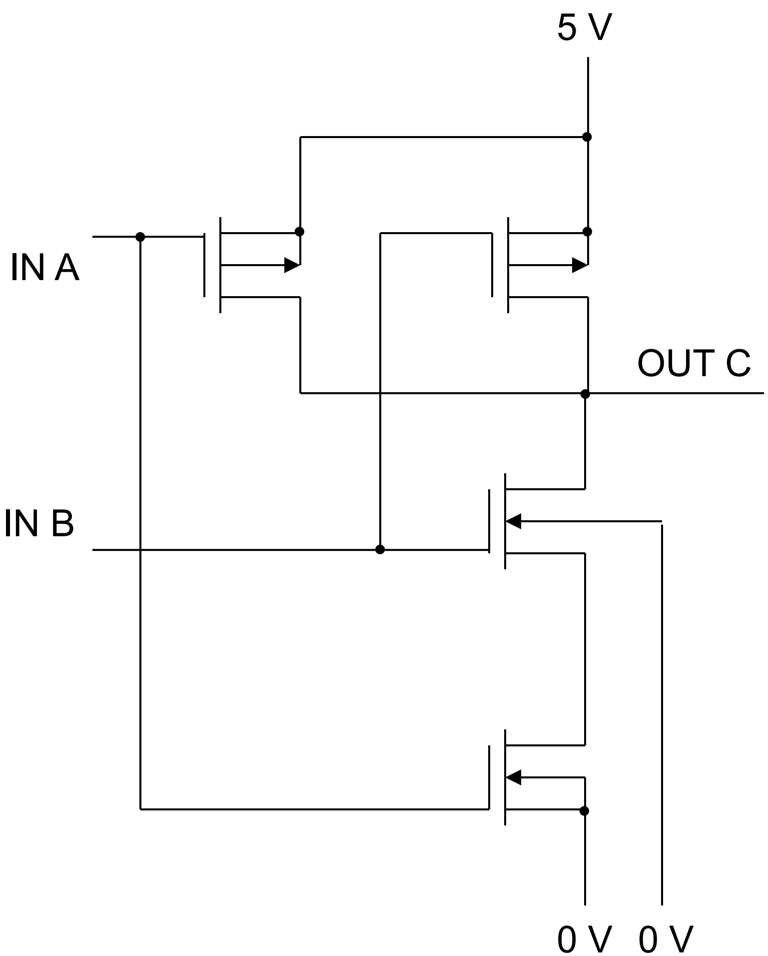

NOT

| A | C |
|---|---|
| 0 | 1 |
| 1 | 0 |




- 0 V input turns OFF lower (n-channel) FET, turns ON upper (p-channel), so output is connected to +5 V
- 5 V input turns ON lower (n-channel) FET, turns OFF upper (p-channel), so output is connected to 0 V
  - Net effect is logic inversion: 0 → 5; 5 → 0
- Complementary MOSFET pairs → CMOS

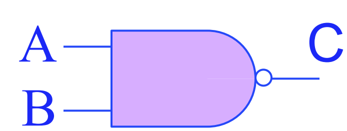
# A NAND gate from scratch:




- Both inputs at zero:
  - lower two FETs OFF, upper two ON
  - result is output HI
- Both inputs at 5 V:
  - lower two FETs ON, upper two OFF
  - result is output LOW
- IN A at 5V, IN B at 0 V:
  - upper left OFF, lowest ON
  - upper right ON, middle OFF
  - result is output HI
- IN A at 0 V, IN B at 5 V:
  - opposite of previous entry
  - result is output HI

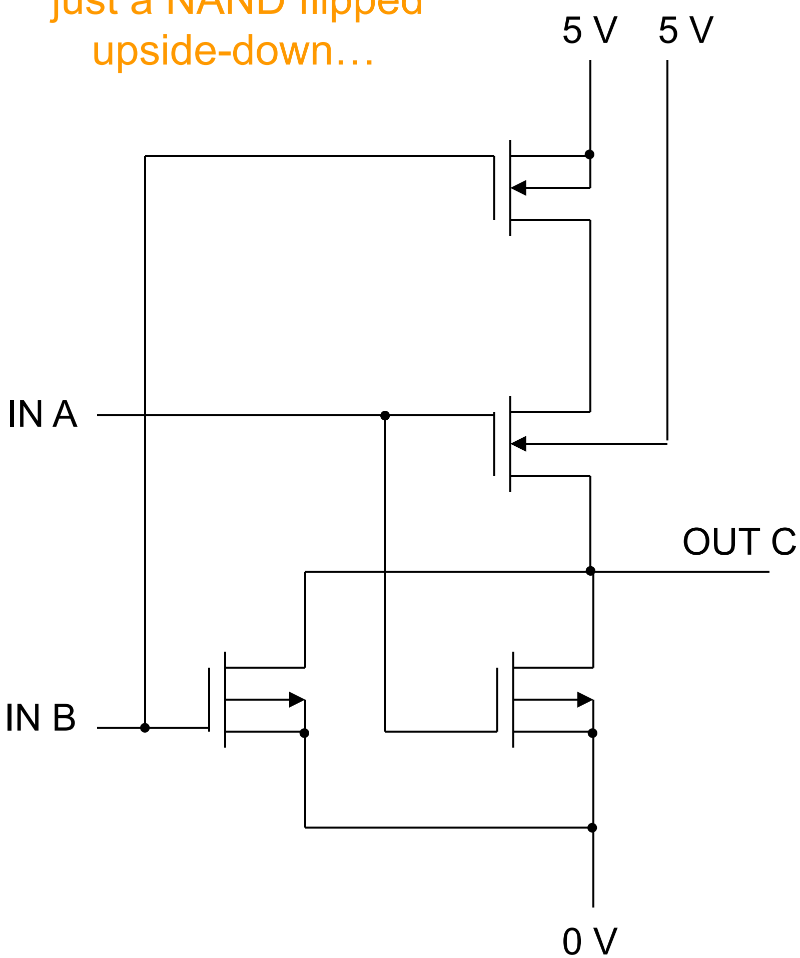
NAND

| A | B | C |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

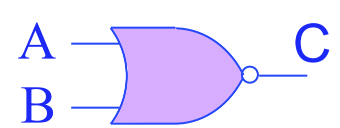


# A NOR gate from scratch:

just a NAND flipped upside-down…




- Both inputs at zero:
  - lower two FETs OFF, upper two ON
  - result is output HI
- Both inputs at 5 V:
  - lower two FETs ON, upper two OFF
  - result is output LOW
- IN A at 5V, IN B at 0 V:
  - lower left OFF, lower right ON
  - upper ON, middle OFF
  - result is output LOW
- IN A at 0 V, IN B at 5 V:
  - opposite of previous entry
  - result is output LOW




NOR

| A | B | C |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

# Rule the World

- Now you know how to build ALL logic gates out of n-channel and p-channel MOSFETs
  - because you can build a NAND from 4 MOSFETs
  - and all gates from NANDs
- That means you can build computers
- So now you can rule the world!

# Arithmetic Example

- Let's add two binary numbers:

  00101110 = 46

  \+ 01001101 = 77

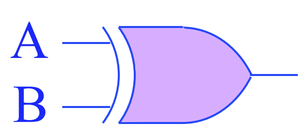
  01111011 = 123

- How did we do this? We have rules:

  0 + 0 = 0; 0 + 1 = 1 + 0 = 1; 1 + 1 = 10 (2): (0, carry 1);

  1 + 1 + (carried 1) = 11 (3): (1, carry 1)

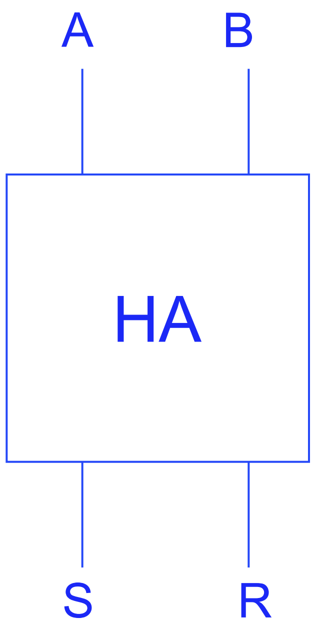
- Rules can be represented by gates
  - If two input digits are A & B, output digit looks like XOR operation (but need to account for carry operation)




XOR

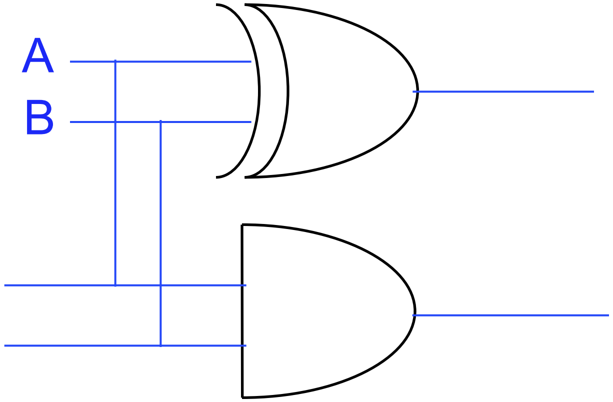
| A | B | C |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

# Half Adder




| A | B | S | R |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

XOR AND




$$Q = A \oplus B = \bar{A} \cdot B + A \cdot \bar{B}$$

$$R = A \cdot B$$

# Half Adder

Somma binaria è analoga alla somma decimale:

1) sommare i due bit corrispondenti al digit $2^n$
2) sommare il risultato al riporto dal digit $2^{n-1}$

Il circuito sommatore a due ingressi è detto Half Adder
ne occorrono due per fare una somma completa

due input ➡ i bit da sommare
due output ➡ la somma e il riporto

può essere costruito con i circuiti di base

# Full Adder

Tabella di verità della somma di 3 bit

| $A_n$ | $B_n$ | $R_{n-1}$ | $S_n$ | $R_n$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

# Full Adder

## Espressione booleana corrispondente alla tabella di verità

$$S_n = \overline{A_n}\,\overline{B_n}R_{n-1} + \overline{A_n}B_n\overline{R_{n-1}} + A_n\overline{B_n}\,\overline{R_{n-1}} + A_nB_nR_{n-1}$$

$$R_n = \overline{A_n}B_nR_{n-1} + A_n\overline{B_n}R_{n-1} + A_nB_n\overline{R_{n-1}} + A_nB_nR_{n-1}$$

possiamo riscrivere $R_n$, sapendo che Q+Q+Q = Q

$$R_n = \left(\overline{A_n}B_nR_{n-1} + A_nB_nR_{n-1}\right) + \left(A_n\overline{B_n}R_{n-1} + A_nB_nR_{n-1}\right) + \left(A_nB_n\overline{R_{n-1}} + A_nB_nR_{n-1}\right)$$

$$R_n = \left(\overline{A_n} + A_n\right)B_nR_{n-1} + \left(\overline{B_n} + B_n\right)A_nR_{n-1} + \left(\overline{R_{n-1}} + R_{n-1}\right)A_nB_n$$

$$R_n = B_nR_{n-1} + A_nR_{n-1} + A_nB_n = A_nB_n + \left(A_n + B_n\right)R_{n-1}$$

# Full Adder

possiamo riscrivere la somma $S_n$

$$S_n = R_{n-1}\left(A_n B_n + \overline{A_n}\,\overline{B_n}\right) + \overline{R_{n-1}}\left(\overline{A_n} B_n + A_n \overline{B_n}\right)$$

ma

$$\left(A_n B_n + \overline{A_n}\,\overline{B_n}\right) = \overline{A_n \oplus B_n}$$
$$\left(\overline{A_n} B_n + A_n \overline{B_n}\right) = A_n \oplus B_n$$

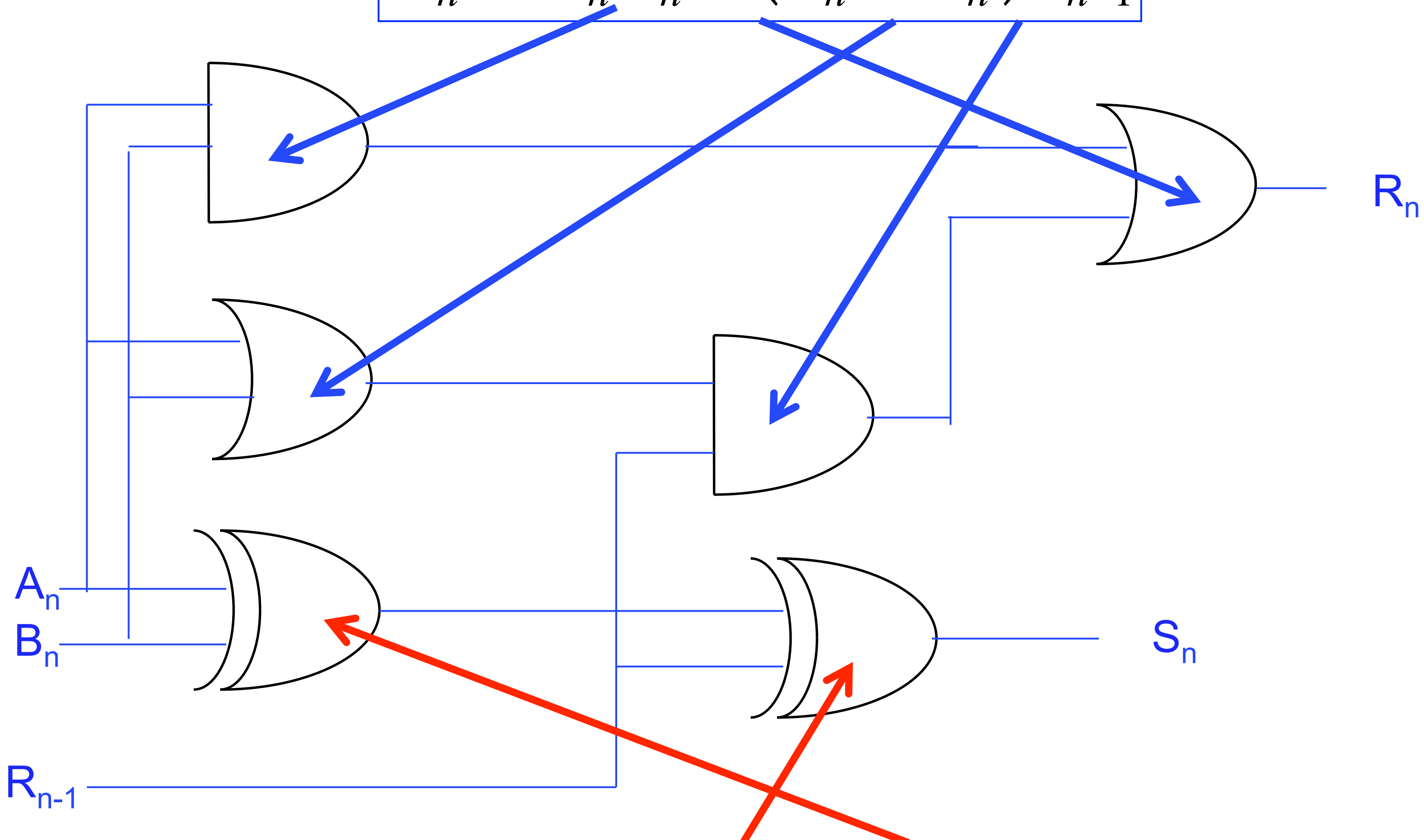
quindi

$$S_n = R_{n-1} \cdot \overline{A_n \oplus B_n} + \overline{R_{n-1}} \cdot A_n \oplus B_n$$

$$\boxed{S_n = R_{n-1} \oplus \left(A_n \oplus B_n\right)}$$

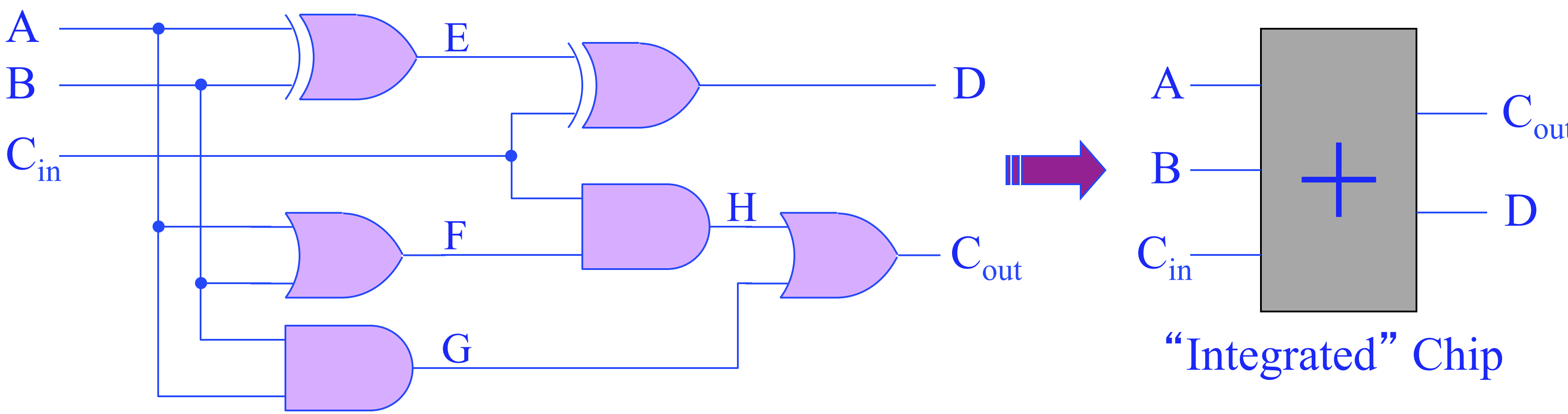
# Full Adder -circuito

$$R_n = A_n B_n + (A_n + B_n) R_{n-1}$$




$$S_n = R_{n-1} \oplus (A_n \oplus B_n)$$

# Binary Arithmetic in Gates




"Integrated" Chip

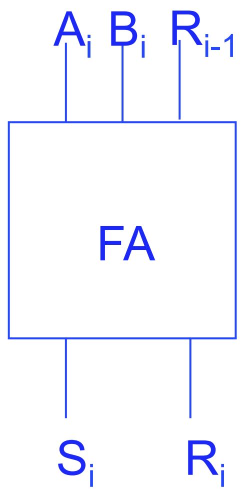
| Input | | | Intermediate | | | | Output | |
|---|---|---|---|---|---|---|---|---|
| A | B | $C_{in}$ | E | F | H | G | D | $C_{out}$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |

Each digit requires 6 gates

Each gate has ~6 transistors

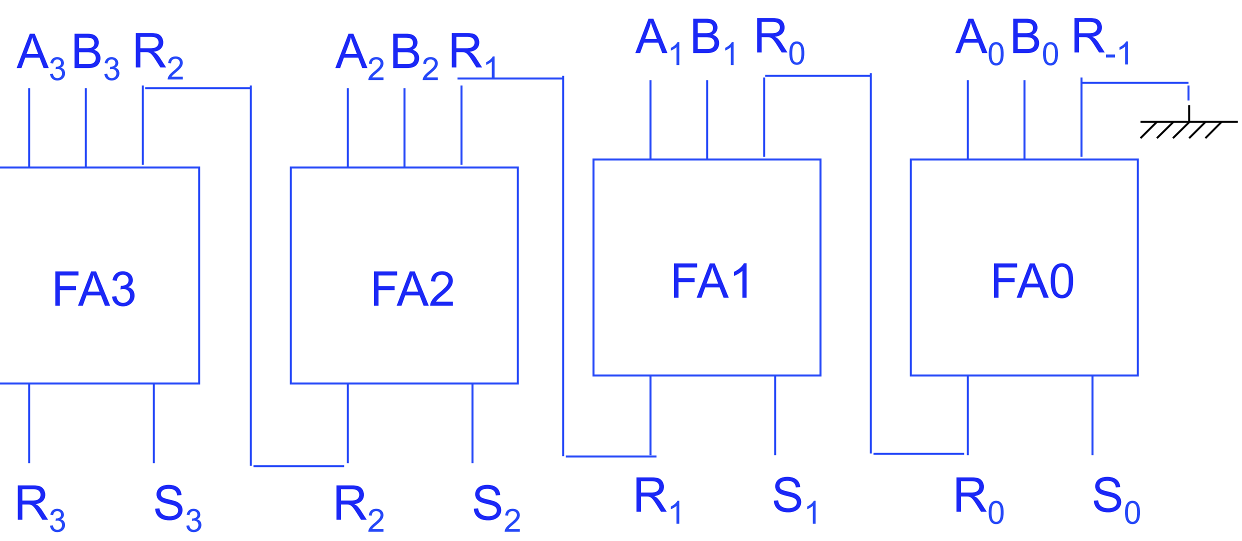
~36 transistors per digit

# Full Adder



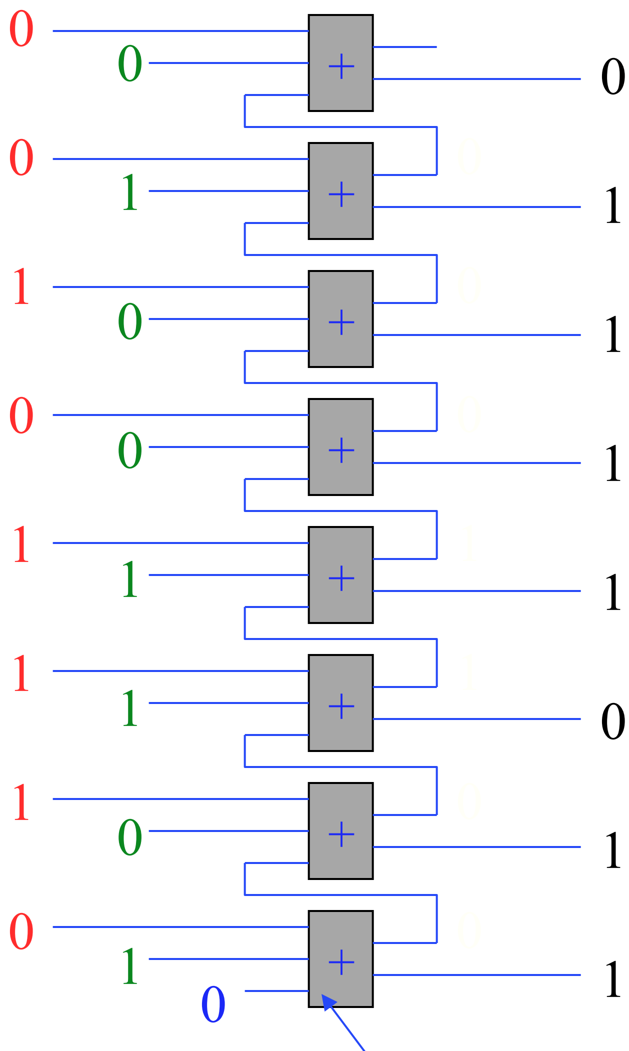

3 input e 2 output

Una somma di 4 bit può essere eseguita in parallelo usando 4 Full Adders



# 8-bit binary arithmetic (cascaded)

MSB




LSB = Least Significant Bit

Integrated one-digit binary arithmetic unit (prev. slide)

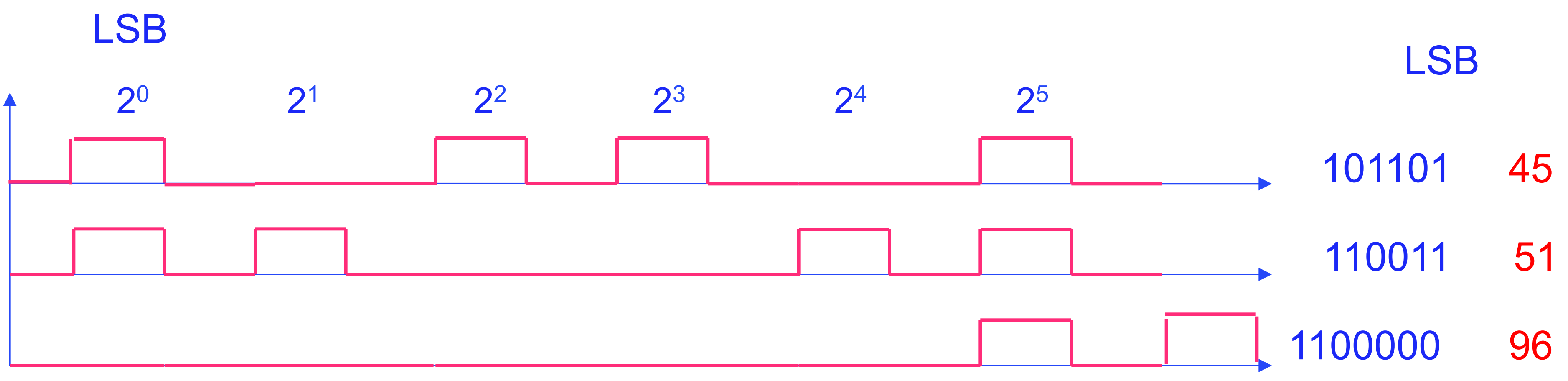
$$
\begin{array}{rcr}
00101110 & = & 46 \\
+\ 01001101 & = & 77 \\
\hline
01111011 & = & 123
\end{array}
$$

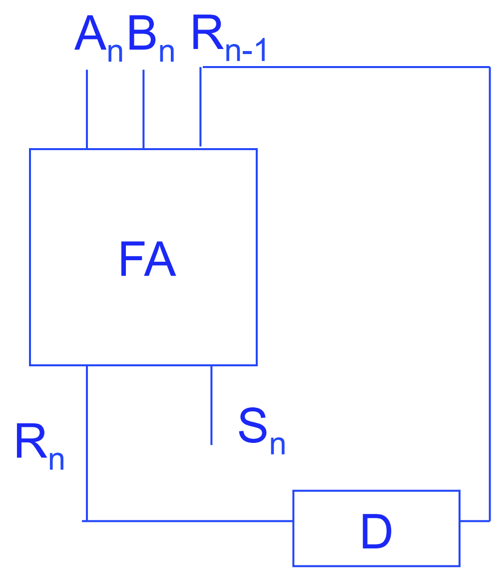
Carry-out tied to carry-in of next digit.

"Magically" adds two binary numbers

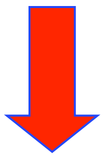
Up to ~300 transistors for this basic function. Also need –, ×, /, & lots more.

# Somma seriale




Una unità di ritardo in più
D = T fra gli impulsi



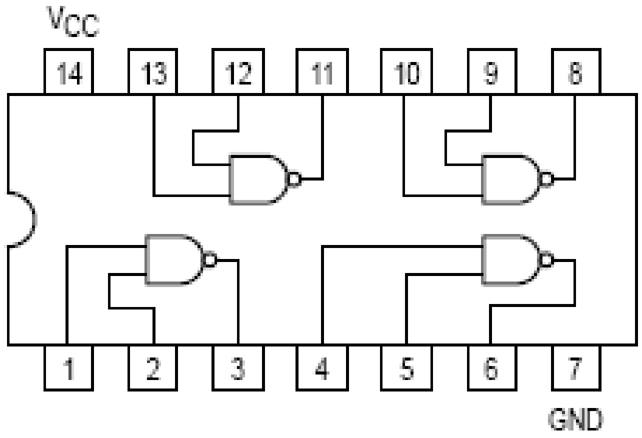
impulso di riporto in tempo
con i bit da sommare

# Circuiti digitali combinatoriali

## Output dipende solo dalla configurazione degli input

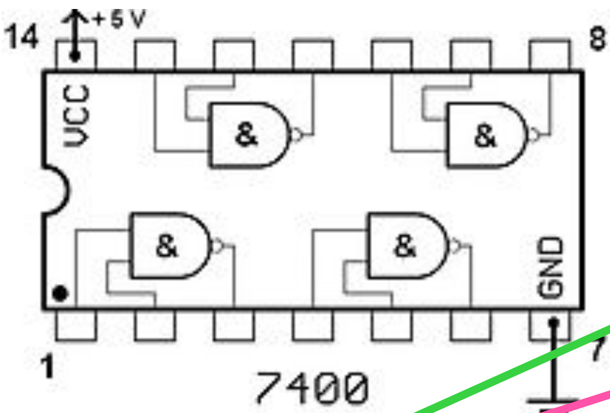
- Operazioni aritmetiche
  - Selezione di dati
    - Decodifica

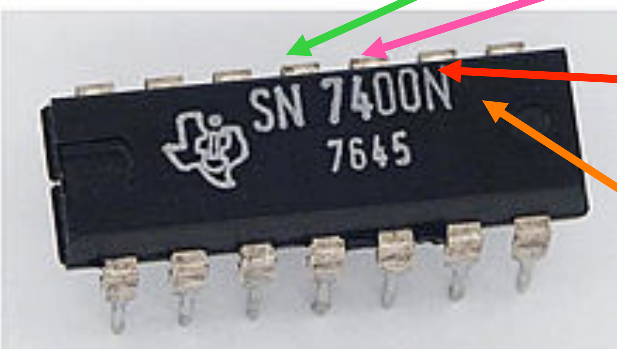
Operazioni base: addizione e sottrazione




14 piedini
1 alimentazione + 1 massa
4 circuiti separati

# Nomenclatura circuiti




AA 74 AAA XXX P

due lettere indicano la casa costruttrice

74, sempre uguale

tre lettere che indicano la sottofamiglia

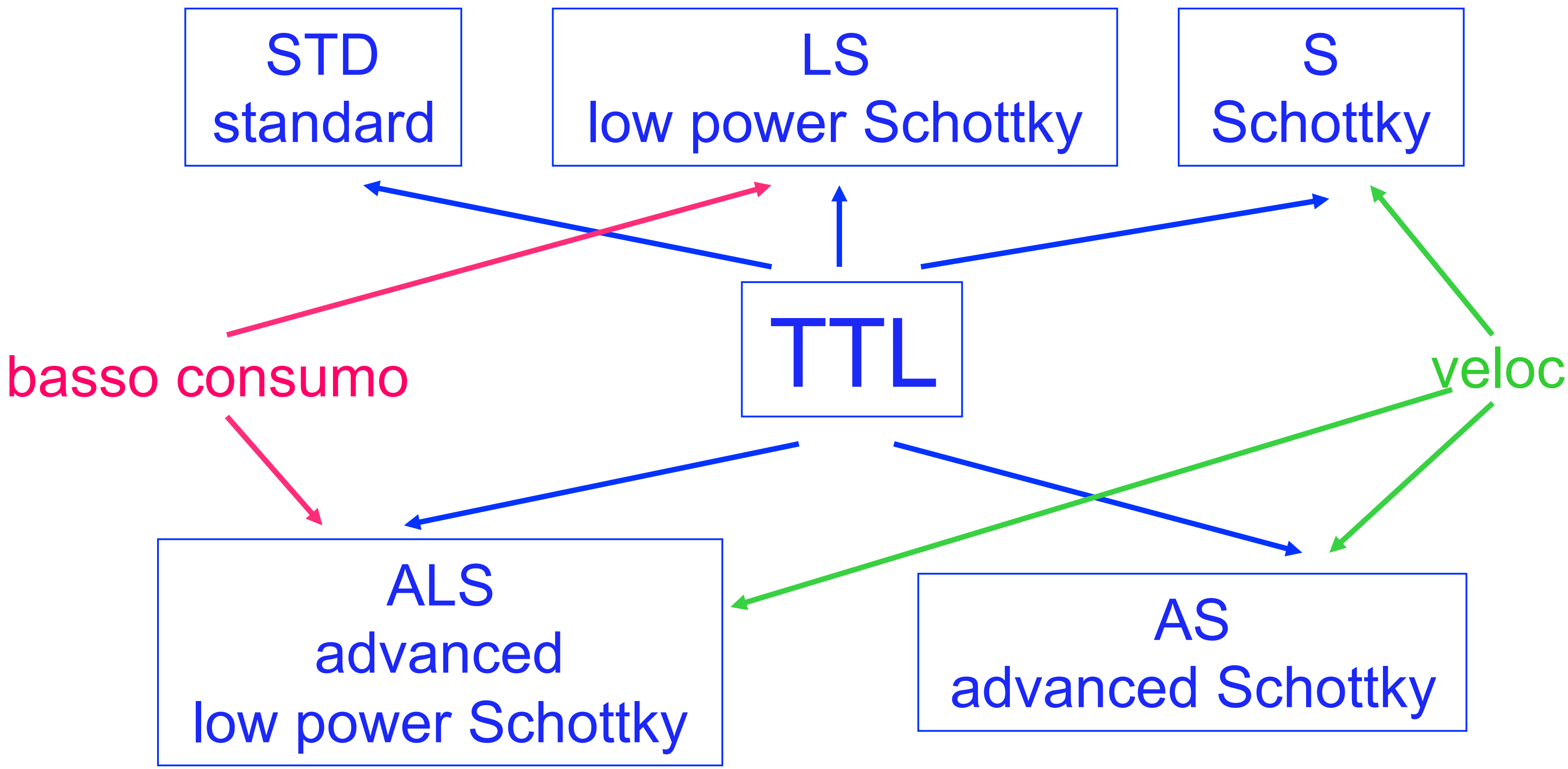
numeri indicano la funzione del circuito

lettere che identificano il contenitore (packaging)

SN74ALS245N

means this is a device probably made by Texas Instruments (SN), it is a commercial temperature range TTL device (74), it is a member of the “Advanced Low-power Schottky" family (ALS), and it is a bi-directional eight-bit buffer (245) in a plastic through-hole DIP package (N).

# Sottofamiglie TTL



# Confronto famiglie logiche

| | TTL | CMOS | ECL |
|---|---|---|---|
| tensione massima di alimentazione | 5 | 5 | -5.2 |
| valore massimo $V_{in}$ identificato come 0 | 0.8 | 1 | -1.4 |
| valore minimo $V_{in}$ identificato come 1 | 2.0 | 3.5 | -1.2 |
| valore massimo $V_{out}$ identificato come 0 | 0.5 | 0.4 | -1.7 |
| valore minimo $V_{out}$ identificato come 1 | 2.7 | 4.2 | -0.9 |

# Circuiti digitali

Sistema digitale
- Unità di controllo (logica)
- Unità aritmetica
- Memoria

Meccanismi di input e output

pochi circuiti fondamentali

NAND
- tutte le operazioni logiche
- celle di memoria

# Computer technology built up from pieces

- The foregoing example illustrates the way in which computer technology is built
  - start with little pieces (transistors acting as switches)
  - *combine* pieces into functional blocks (gates)
  - *combine* these blocks into higher-level function (e.g., addition)
  - *combine* these new blocks into cascade (e.g., 8-bit addition)
  - blocks get increasingly complex, more capable
- Nobody on earth understands Pentium chip inside-out
  - Grab previously developed blocks and run
  - Let a computer design the gate arrangements (eyes closed!)