AXO – Architettura dei Calcolatori e Sistemi Operativi

reti combinatorie

Sommario

- Il segnale binario
- Algebra di Boole e funzioni logiche
- Porte logiche
- Analisi e sintesi di circuiti combinatori

1- Segnali e informazioni

- Per elaborare informazioni, occorre rappresentarle (o codificarle)
- Per rappresentare (o codificare) le informazioni si usano segnali
- I segnali devono essere elaborati, nei modi opportuni, tramite dispositivi di elaborazione
- In un **sistema digitale** le informazioni vengono **rappresentate**, **elaborate** e **trasmesse** mediante grandezze fisiche (segnali) che si considerano assumere solo **valori discreti**. Ogni valore è associato a una cifra (**digit**) della rappresentazione.

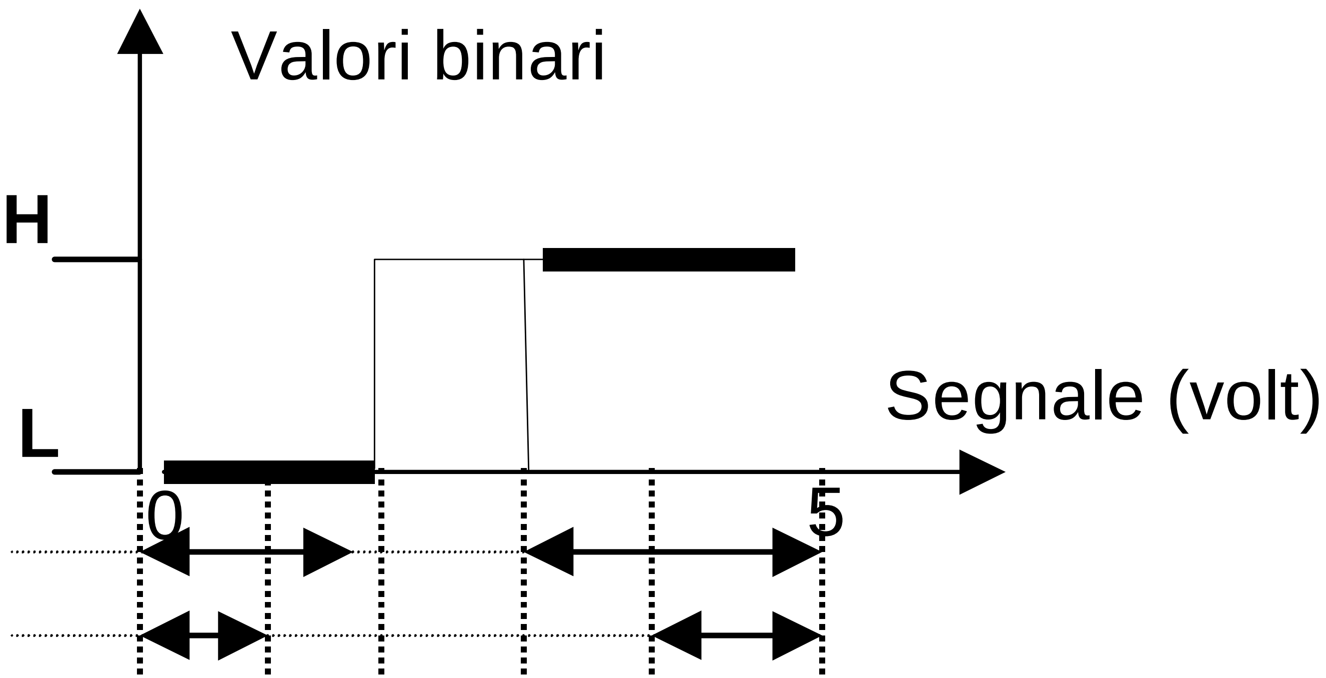
Il segnale binario (i)

- **Segnale binario**: una grandezza che può assumere due valori distinti, convenzionalmente indicati con 0 e 1
  - $s \in \{0, 1\}$ (low, high - False, True)

- **Grandezze fisiche** utilizzate in un sistema digitale per la rappresentazione dell'informazione:
  - ⇒ segnali **elettrici** (**tensione**, corrente)
  - ⇒ grandezze di tipo magnetico (stato di magnetizzazione)
  - ⇒ segnali ottici

Il segnale binario (ii)

La grandezza fisica che si utilizza (segnale elettrico di tensione) assume solo **due valori discreti** (binaria)



valori accettati

valori emessi

Il segnale binario (iii)

- Elaborazione del segnale binario: si usano due classi di dispositivi di elaborazione
  - **reti combinatorie**: l'uscita all'istante t dipende dagli ingressi nello stesso istante
  - **reti sequenziali** (reti con memoria): l'uscita all'istante t dipende dagli ingressi nello stesso istante e dalla "storia passata" (= stato della rete)
- Sono tutti circuiti digitali (o numerici)

2 - Algebra di Commutazione

- Deriva dall'algebra di Boole e consente di descrivere matematicamente i circuiti digitali (o circuiti logici)
- Definisce le **espressioni logiche** che descrivono il **comportamento** del circuito da realizzare nella forma U= f(I)
- A partire dalle equazioni logiche è possibile derivare la **realizzazione circuitale** (rete logica)
- I componenti dell'algebra di Boole sono: le variabili di commutazione, gli operatori fondamentali e le proprietà degli operatori logici tramite le quali è possibile trasformare le espressioni logiche

Variabili di commutazione e operatori

- Una **variabile di commutazione** (o variabile logica) corrisponde al singolo bit di informazione rappresentata e elaborata
- Gli **operatori fondamentali** sono

Negazione	!A oppure /A	=1 per A=0 =0 per A=1
Somma logica	A + B	= 0 se e solo se A=B=0
Prodotto logico	A · B	= 1 se e solo se A=B=1

Precedenza degli operatori: negazione, prodotto, somma

Operatori logici

Somma	Prodotto	Inversione
0 + 0 = 0	0 0 = 0	!0 = 1
0 + 1 = 1	0 1 = 0	!1 = 0
1 + 0 = 1	1 0 = 0	
1 + 1 = 1	1 1 = 1	

Proprietà degli operatori logici (1)

Legge	**AND**	**OR**
Identità	1 A = A	0 + A = A
Elemento nullo	0 A = 0	1 + A = 1
Idempotenza	A A = A	A + A = A
Inverso	A !A = 0	A + !A = 1

Proprietà degli operatori logici (2)

Legge	AND	OR
Commutativa	A B = B A	A + B = B + A
Associativa	(A B) C = A (B C)	(A + B) + C = A + (B + C)
Distributiva	AND rispetto a OR A (B + C) = A B + A C	OR rispetto a AND A + B C = (A + B) (A + C)
Assorbimento	A (A + B) = A	A + A B = A
De Morgan	!(A B) = !A + !B	!(A + B) = !A !B

Esempio di trasformazione

- F(a,b,c)=!a!bc + !abc + !ab!c

Espressione trasformata	proprietà utilizzata
!a!bc+!abc+!ab!c	idempotenza x+x=x
!a!bc+!abc+!abc+!ab!c	distributiva xy+xz=x(y+z)
!ac(!b+b) + !ab(c+!c)	inverso x+!x=1
!ac1 + !ab1	identità x1=x
!ac + !ab	distributiva
!a(c + b)	

Funzioni combinatorie

- Una funzione combinatoria (o funzione booleana, o funzione logica) corrisponde a un'espressione booleana, contenente una o più variabili booleane e gli operatori booleani AND, OR e NOT
  - Dando dei valori alle variabili booleane della funzione combinatoria, si calcola il corrispondente valore della funzione

- Esempio: funzione logica a 2 ingressi a e b e 2 uscite S e C
  - S=1 se e solo se solo uno degli ingressi vale 1
  - C=1 se e solo se entrambi gli ingressi valgono 1
- Espressioni booleane
  - S=a!b + !ab
  - C=ab

Tabella delle verità

- Per specificare il comportamento di una funzione combinatoria è possibile specificare, per ogni possibile configurazione degli ingressi, il valore dell'uscita: **tabella delle verità**

- La tabella della verità di una funzione a *n* ingressi ha 2^n righe, che corrispondono a tutte le possibili configurazioni di ingresso

- La tabella delle verità ha due "gruppi" di colonne:
  - colonne degli ingressi, le cui righe contengono tutte le combinazioni di valori delle variabili della funzione
  - colonna dell'uscita, che riporta i corrispondenti valori assunti dalla funzione

Esempio

- F(A, B, C) = AB + !C è una funzione combinatoria a 3 variabili A, B e C
  - F(0, 0, 0) = 0 0 + !0 = 0 + 1 = 1
  - F(0, 0, 1) = 0 0 + !1 = 0 + 0 = 0
  - F(0, 1, 0) = 0 1 + !0 = 0 + 1 = 1
  - … (e così via)
  - F(1, 1, 1) = 1 1 + !1 = 1 + 0 = 1

Esempio: Tabella delle verità

n = 3
ingressi

$2^n = 2^3 = 8$
righe

colonna
uscita

# riga	A	B	C	A B + /C	F
0	0	0	0	0 0 + /0	1
1	0	0	1	0 0 + /1	0
2	0	1	0	0 1 + / 0	1
3	0	1	1	0 1 + /1	0
4	1	0	0	1 0 + /0	1
5	1	0	1	1 0 + /1	0
6	1	1	0	1 1 + /0	1
7	1	1	1	1 1 + /1	1

(per comodità nella colonna centrale
è riportato anche il calcolo)

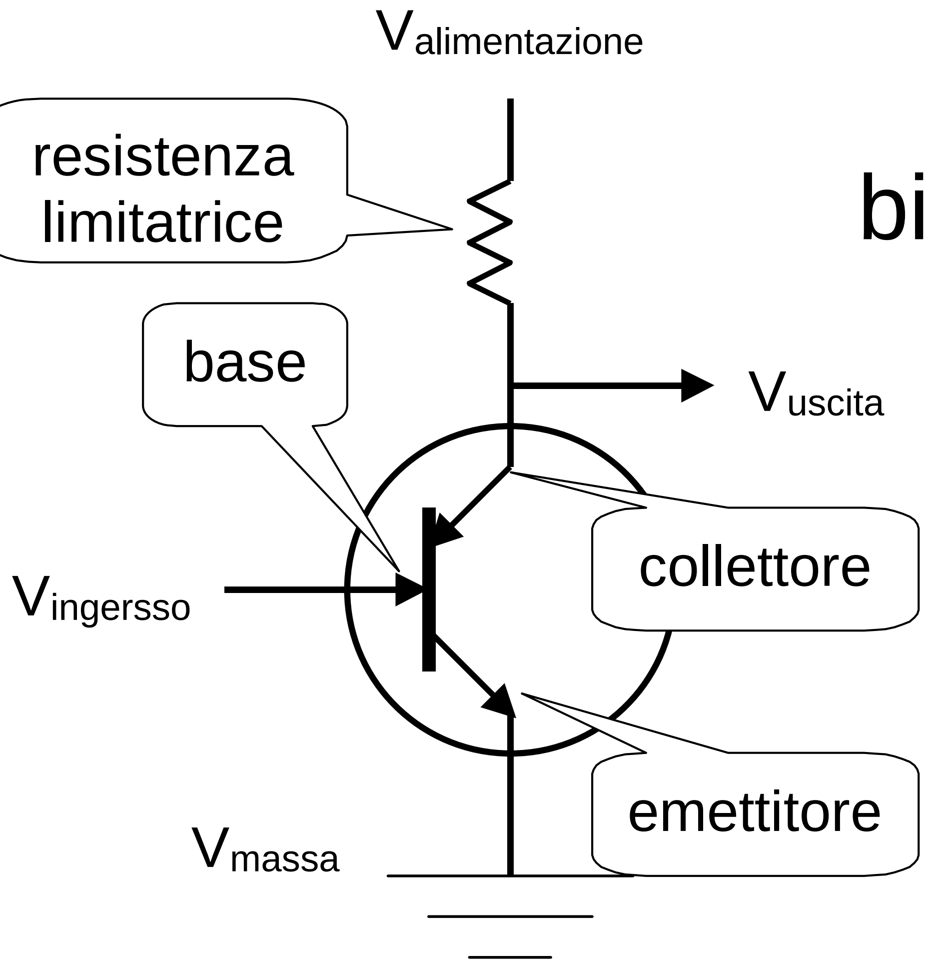
3 - Porte logiche (circuiti combinatori elementari)

- I circuiti digitali sono formati da componenti digitali elementari, chiamati porte logiche
- Le porte logiche sono i circuiti minimi per l'elaborazione di segnali binari e corrispondono agli operatori elementari dell'algebra di commutazione
- Le porte logiche vengono classificate in base al modo di funzionamento: porta NOT, porta AND, porta OR (sono le porte logiche fondamentali e costituiscono un insieme di operatori funzionalmente completo)
  - Classificazione per numero di ingressi: porte a 1 ingresso, porte a 2 ingressi, porte 3 ingressi, e così via …

Transistor

- L'elemento funzionale fondamentale per la costruzione di porte logiche è il transistor
- Il transistor è un dispositivo elettronico
- Il transistor opera su grandezze elettriche: tensione e corrente
- Il transistor funziona come un interruttore
- Ha due stati di funzionamento: interruttore aperto o interruttore chiuso

Struttura del transistor



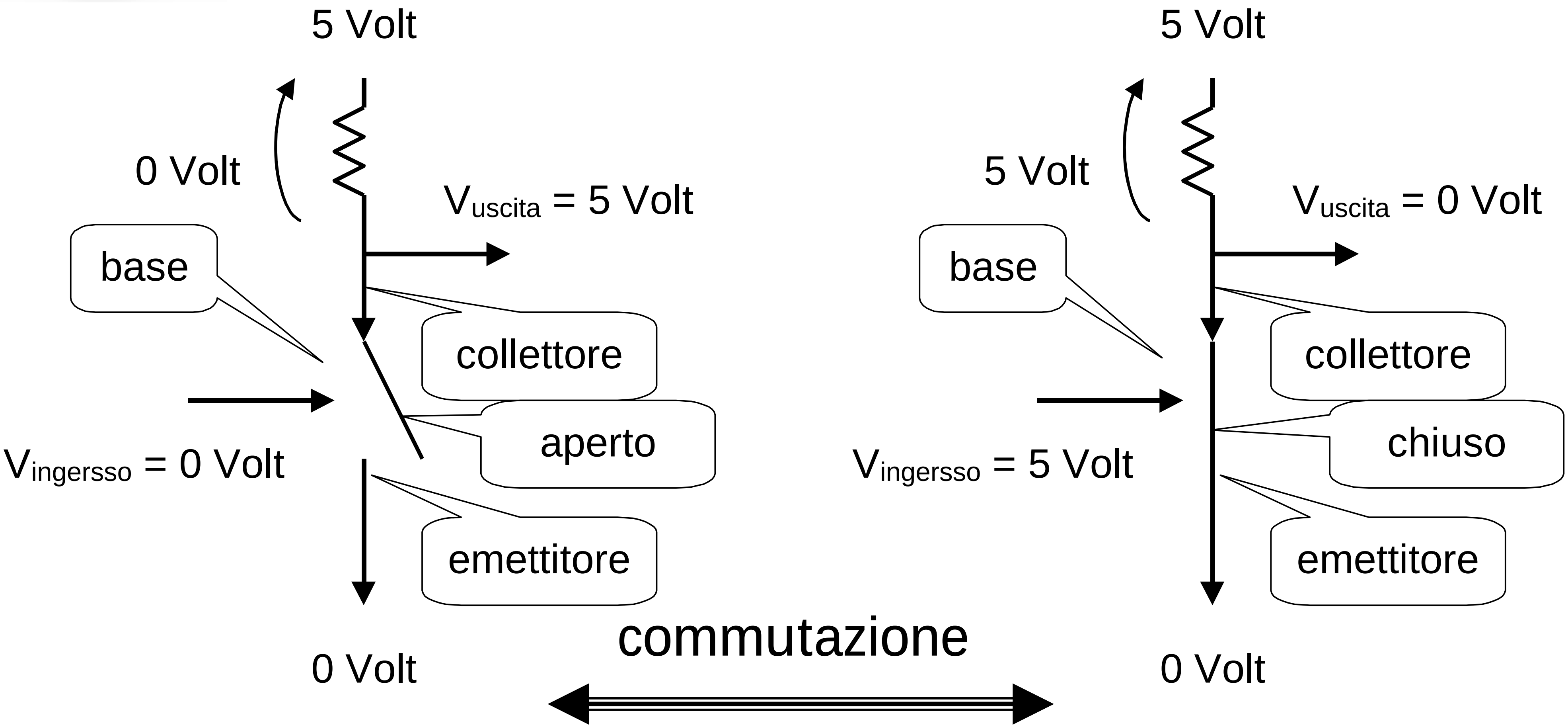
transistor bipolare (BJT)

Funzionamento del transistor

- Se la tensione di base $V_{ingresso}$ è inferiore a una data soglia critica, il transistor si comporta come un interruttore aperto, cioè tra emettitore e collettore non passa corrente, e quindi la tensione di uscita diventa uguale a quella di alimentazione: $V_{uscita} = V_{alimentazione} =$ 5 Volt (in tecnologia TTL)

- Se la tensione di base $V_{ingresso}$ è superiore a una data soglia critica, il transistor si comporta come un interruttore chiuso, cioè tra emettitore e collettore passa corrente, e quindi la tensione di uscita diventa uguale a quella di massa: $V_{uscita} = V_{massa} =$ 0 Volt (in tecnologia TTL)

Funzionamento del transistor



≈ 1 ns = 10^{-9} sec

La porta NOT (invertitore)

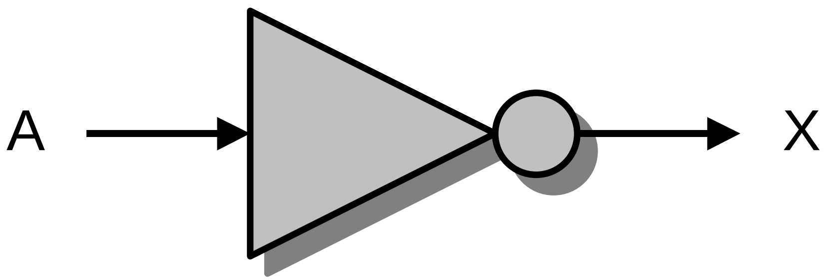
- Il singolo transistor della figura è una porta NOT
- Se l'ingresso vale 0 Volt, l'uscita vale 5 Volt
- Se l'ingresso vale 5 Volt, l'uscita vale 0 Volt

- La tabella rappresenta il funzionamento della porta NOT

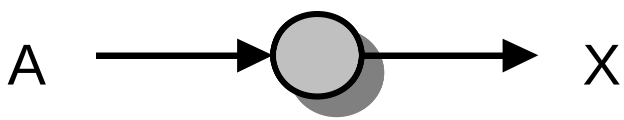
$V_{ingresso}$	V_{uscita}
0 Volt	5 Volt
5 Volt	0 Volt

Porta NOT (invertitore, negatore)

Simbolo funzionale



(a 1 ingresso)



simbolo semplificato

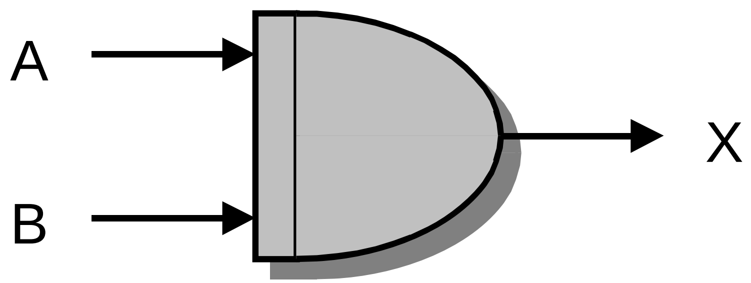
Tabella delle verità

A	X
0	1
1	0

L’uscita vale 1 se e solo se l’ingresso vale 0

Porta AND

Simbolo funzionale



(a 2 ingressi)

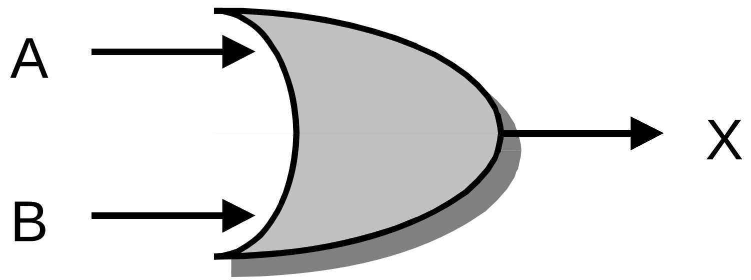
Tabella delle verità

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

L'uscita vale 1 se e solo se entrambi gli ingressi valgono 1

Porta OR

Simbolo funzionale



(a 2 ingressi)

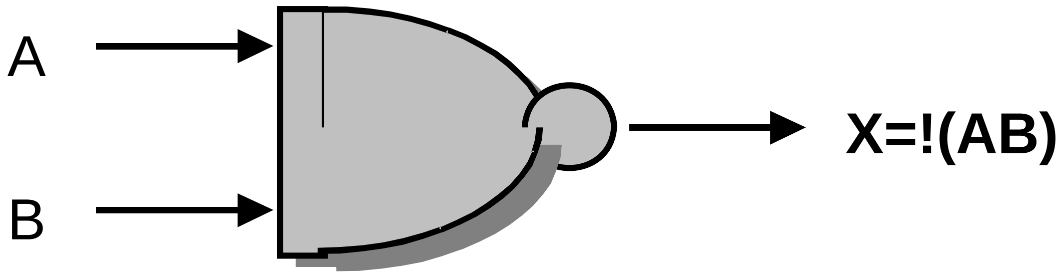
Tabella delle verità

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

L'uscita vale 1 se e solo se almeno un ingresso vale 1

NAND (operatore funzionalmente completo)

Simbolo funzionale



(a 2 ingressi)

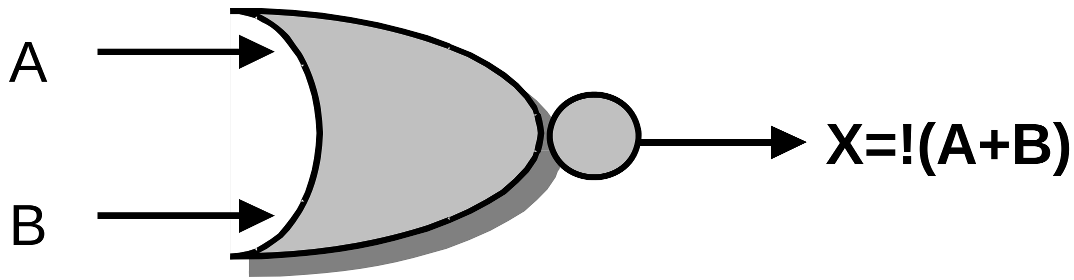
Tabella delle verità

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

L'uscita vale 0 se e solo se entrambi gli ingressi valgono 1

NOR (operatore funzionalmente completo)

Simbolo funzionale



(a 2 ingressi)

Tabella delle verità

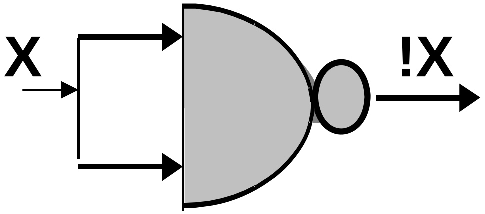
A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

L'uscita vale 1 se e solo se entrambi gli ingressi valgono 0

NAND - realizzazione di NOT, AND, OR

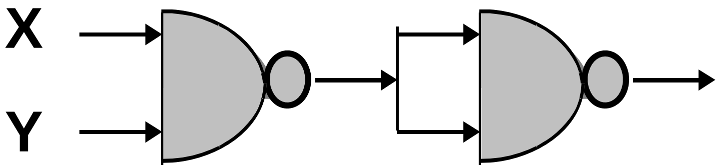
NOT

!X=!(XX)



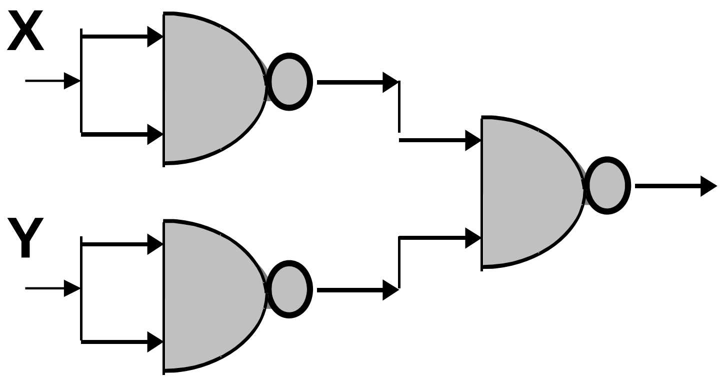
AND

XY=!!(XY)



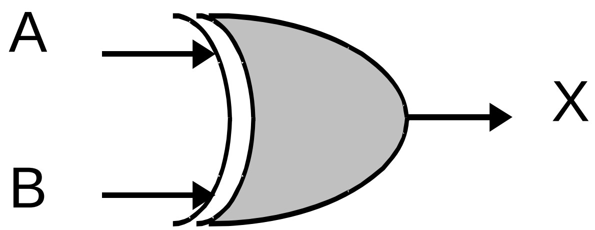
OR

X+Y=!!(X+Y)=!(!X!Y)



Altri operatori: OR esclusivo (X-OR)

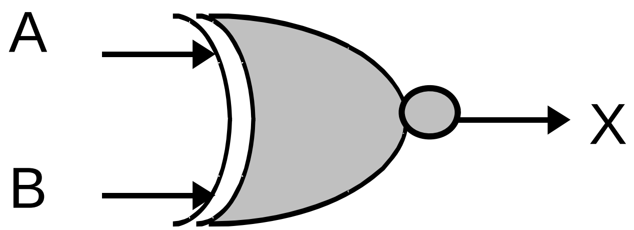
- X-OR a 2 ingressi: l'uscita vale 1 se e solo se una sola variabile vale 1 (diseguaglianza)
  - generalizzato a n variabili di ingresso: l'uscita vale 1 se e solo se il numero di 1 è dispari



A	**B**	**X**=!AB+A!B
0	0	0
0	1	1
1	0	1
1	1	0

Altri operatori: X-NOR esclusivo

- X-NOR a 2 ingressi: l'uscita vale 1 se e solo se entrambe le variabili valgono 0 o 1 (eguaglianza)
  - generalizzato a n variabili di ingresso: l'uscita vale 1 se e solo se il numero di 1 è pari



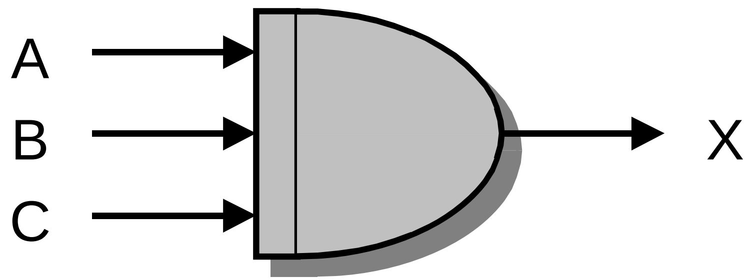
A	B	X=AB+!A!B
0	0	1
0	1	0
1	0	0
1	1	1

Generalizzazioni

- Alcuni tipi di porte a 2 ingressi si possono generalizzare a 3, 4, ecc ingressi
- Le due porte a più ingressi maggiormente usate sono la porta AND e la porta OR
- Tipicamente si usano AND (o OR) a 2, 4 o 8 ingressi (raramente più di 8)

Porta AND a 3 ingressi

Simbolo funzionale



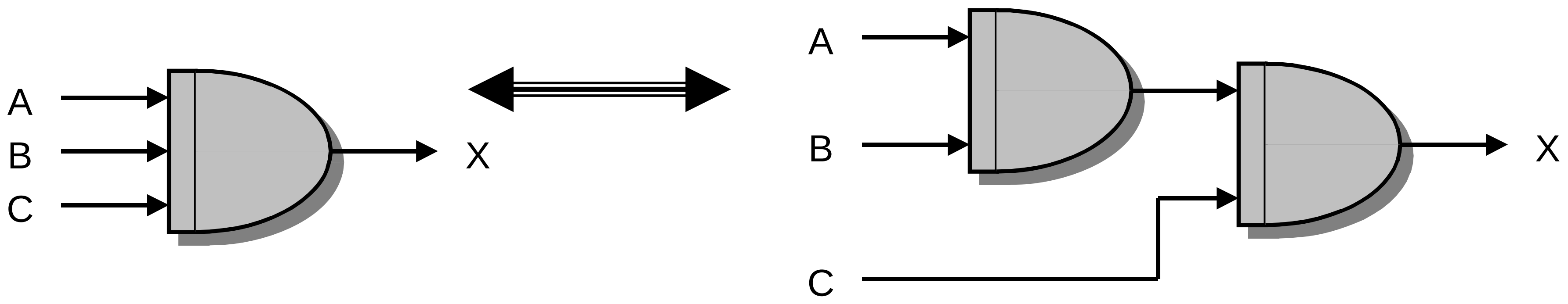
L'uscita vale 1 se e solo se tutti e 3 gli ingressi valgono 1

Tabella delle verità

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Realizzazione ad albero

- La porta AND a 3 ingressi si realizza spesso come albero di porte AND a 2 ingressi (ma non è l'unico modo)

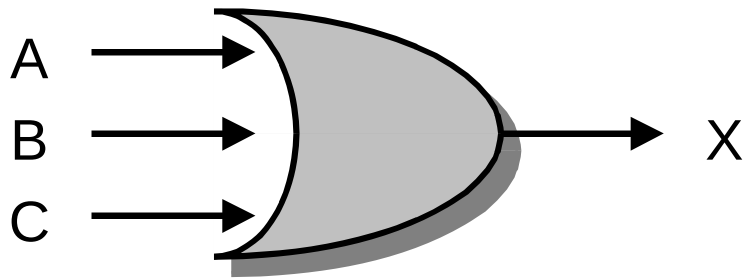


- Nota bene: non tutti i tipi di porte a più di 2 ingressi si possono realizzare come alberi di porte a 2 ingressi (funziona sempre con AND, OR, X-OR, X-NOR)

Porta OR a 3 ingressi

Tabella delle verità

Simbolo funzionale



A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

L'uscita vale 0 se e solo se tutti e 3 gli ingressi valgono 0

Costo e velocità di una porta logica

- Costo di realizzazione
  - Il numero di transistor per realizzare una porta dipende dalla tecnologia, dalla funzione e dal numero di ingressi
  - Porta NOT: 1 oppure 2 transistor, porte AND e OR: 3 oppure 4 transistor, altre porte: $\geq$ 4 transistor
- Velocità di commutazione
  - La velocità di commutazione di una porta dipende dalla tecnologia, dalla funzione e dal numero di ingressi
  - Le porte più veloci (oltre che più piccole) sono tipicamente le porte NAND e NOR a 2 ingressi: possono commutare in meno di 1 nanosecondo (10^{-9} sec, un miliardesimo di sec)
- Il costo delle porte logiche e la velocità di commutazione consentono di calcolare un'indicazione del costo della rete logica che realizza un'espressione booleana e del ritardo di propagazione associato alla rete stessa

4 - Analisi e sintesi di reti combinatorie

Rete combinatoria (1)

- A ogni funzione combinatoria, data come espressione booleana, si può sempre associare un circuito digitale, formato da porte logiche, che viene chiamato **rete combinatoria**

- Gli ingressi della rete combinatoria sono le variabili della funzione

- L'uscita della rete combinatoria emette il valore assunto dalla funzione

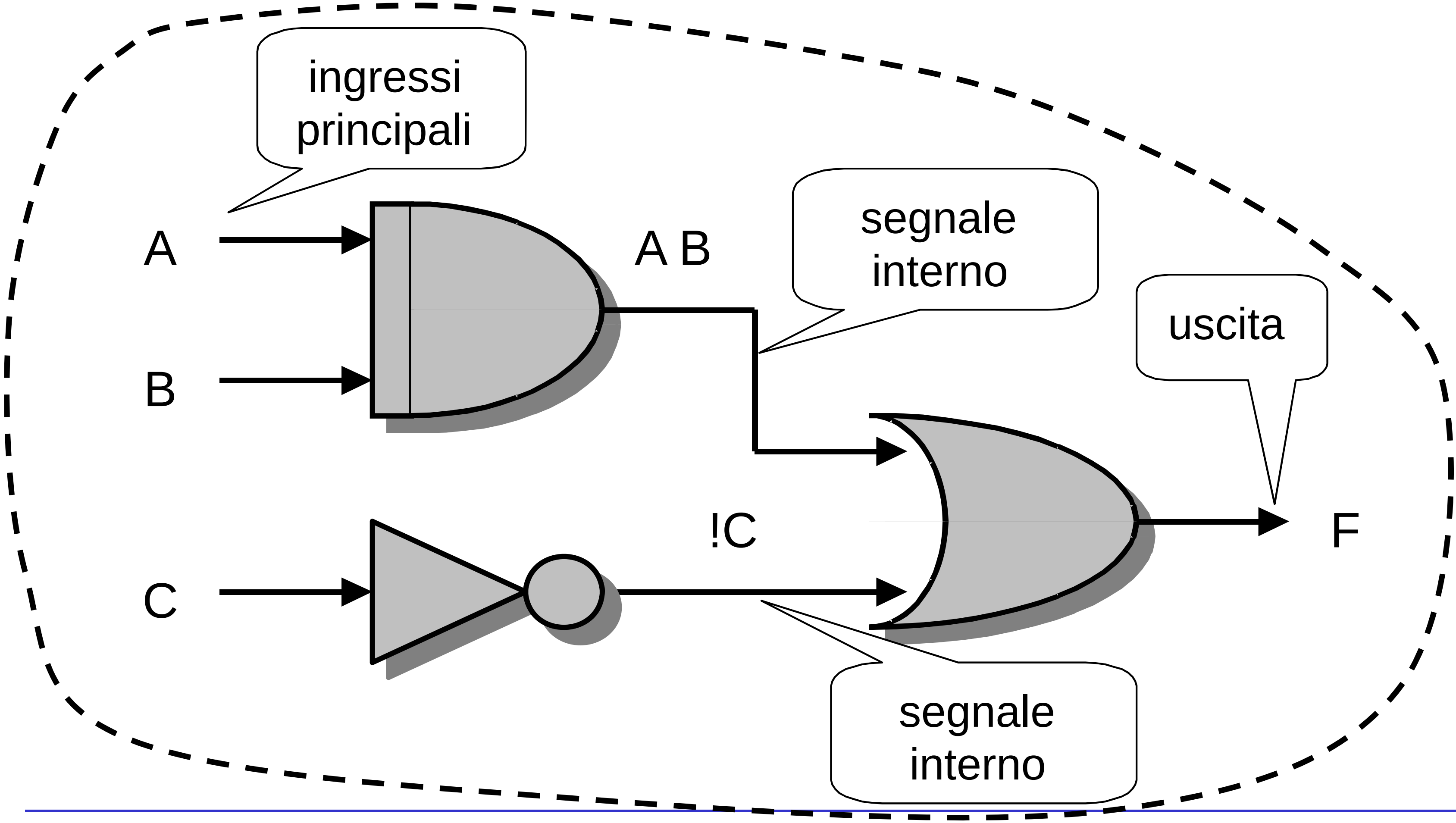
Rete combinatoria (2)

- Una rete combinatoria è un circuito digitale:
  - dotato di $n \geq 1$ ingressi principali e di un'uscita
  - formato da porte logiche AND, OR e NOT (eventualmente anche da altri tipi di porte)
  - e privo di retroazioni

- Costruzione della rete combinatoria a partire dalla funzione logica
  - variabili e variabili negate
  - ogni termine dell'espressione è sostituito dalla corrispondente rete di porte fondamentali
  - le uscite corrispondenti ad ogni termine si compongono come indicato dagli operatori …

Esempio

$$F(A, B, C) = A\,B + !C$$



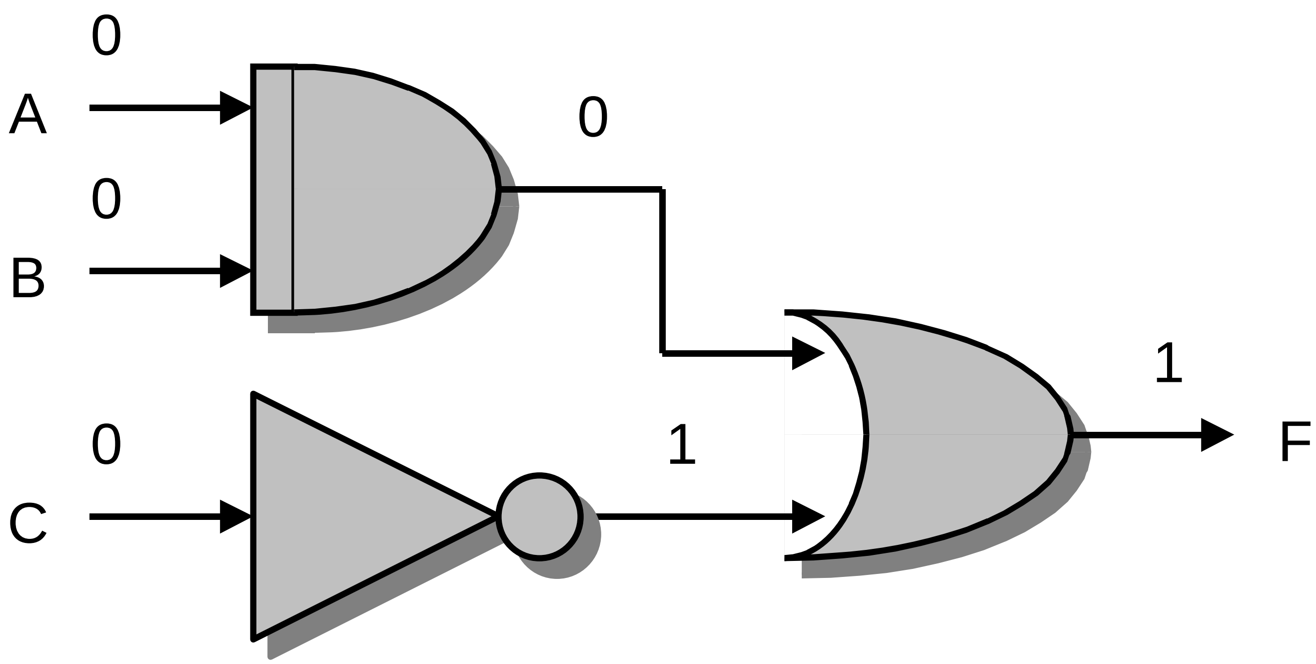
Simulazione circuitale
(dalla rete alla tabella della verità)

- La tabella delle verità di una rete combinatoria può anche essere ricavata per simulazione del funzionamento circuitale della rete combinatoria stessa

- Per simulare il funzionamento circuitale di una rete combinatoria, si applicano dei valori agli ingressi, e li si propaga lungo la rete fino all’uscita

Simulazione circuitale

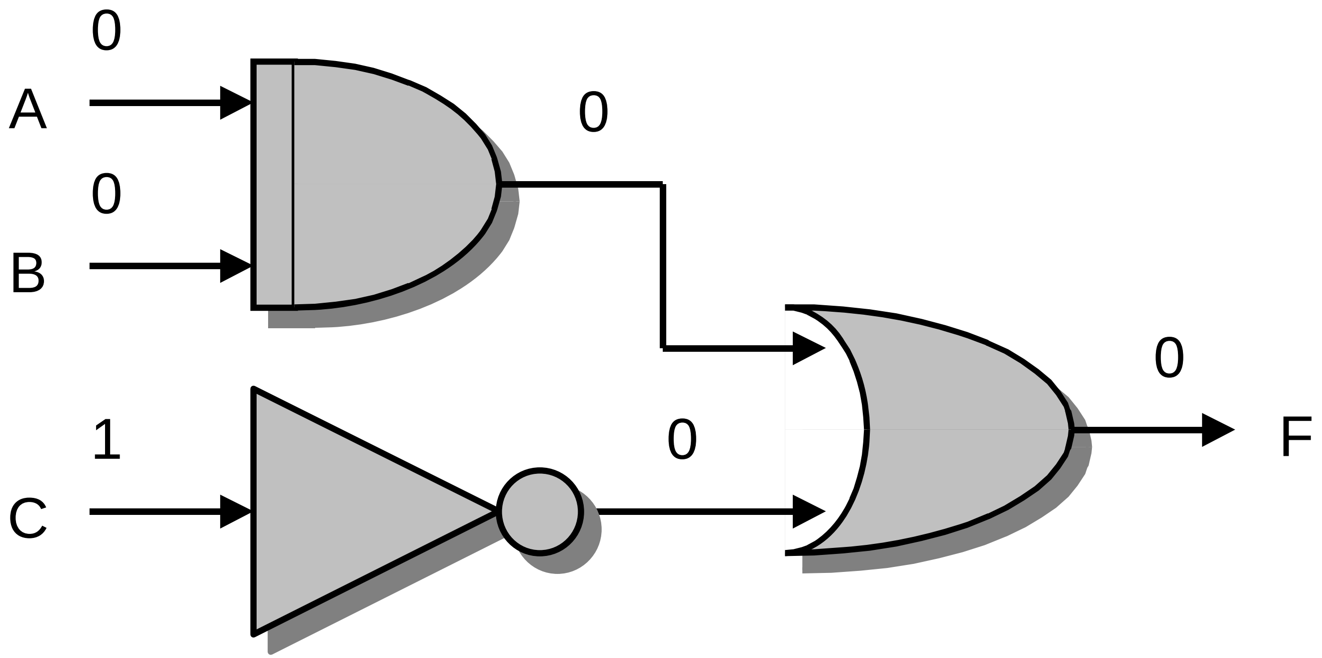
(corrisponde alla riga 0 della tabella)



Risultato della simulazione: F(0, 0, 0) = 1

Simulazione circuitale

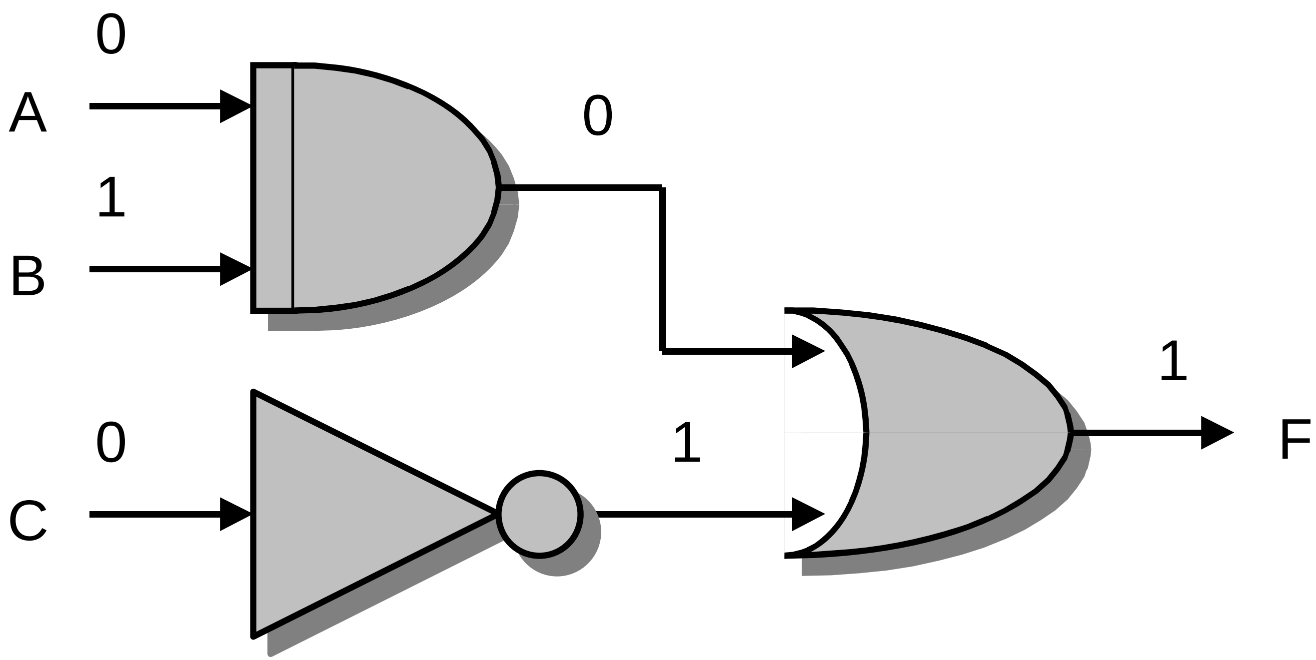
(corrisponde alla riga 1 della tabella)



Risultato della simulazione: F(0, 0, 1) = 0

Simulazione circuitale

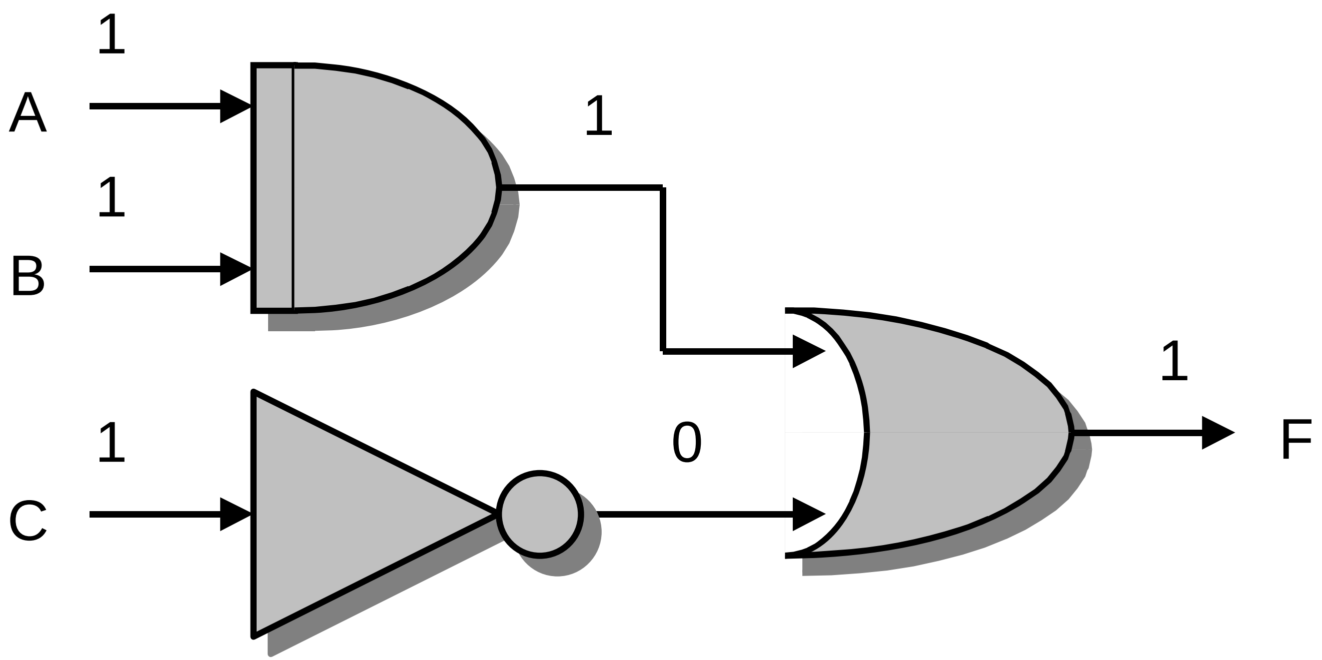
(corrisponde alla riga 2 della tabella)



Risultato della simulazione: F(0, 1, 0) = 1

Simulazione circuitale

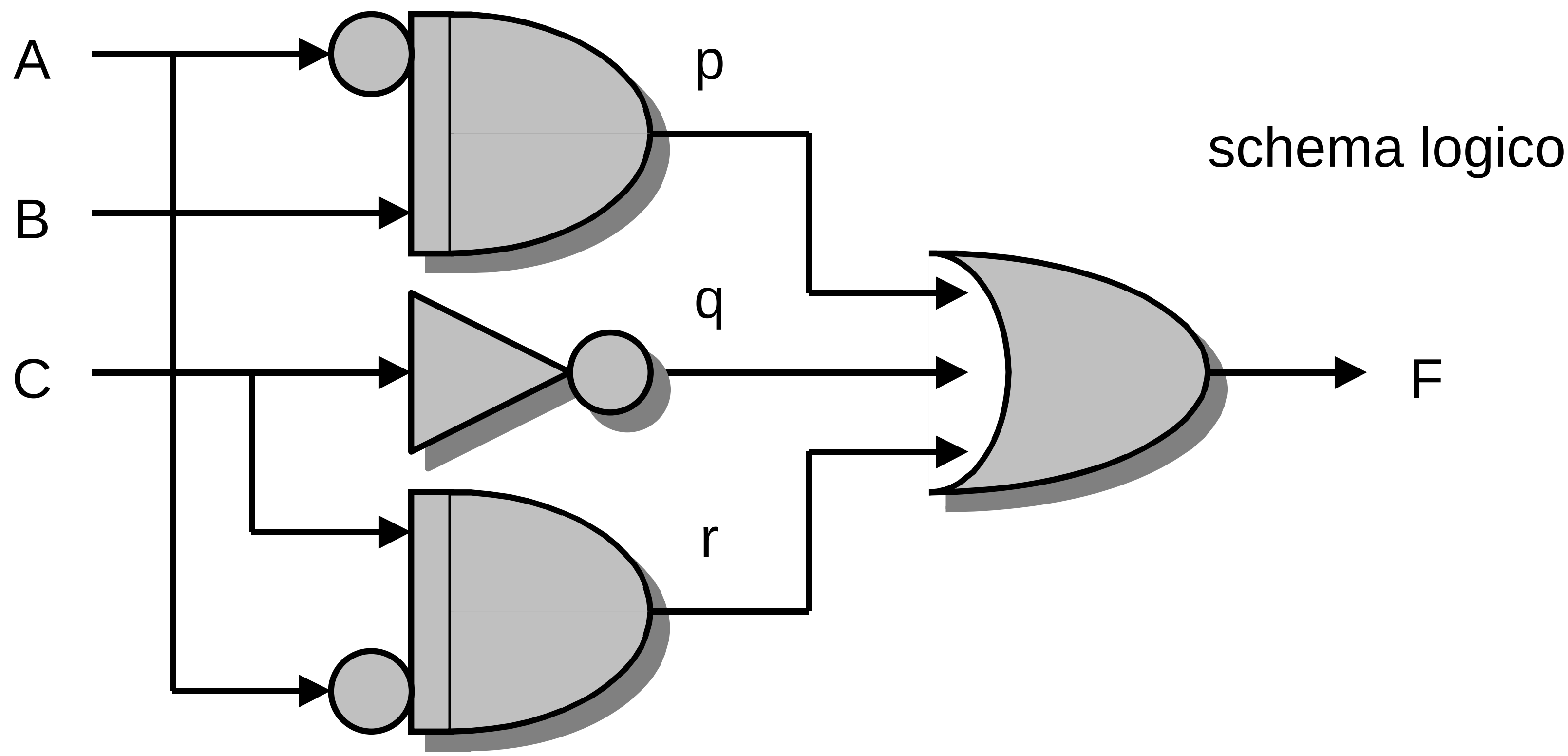
(corrisponde alla riga 7 della tabella)



Risultato della simulazione: F(1, 1, 1) = 1

4a - Analisi di reti combinatorie
- dalla rete all'espressione

(1) Si applicano nomi ai segnali interni:p, q e r

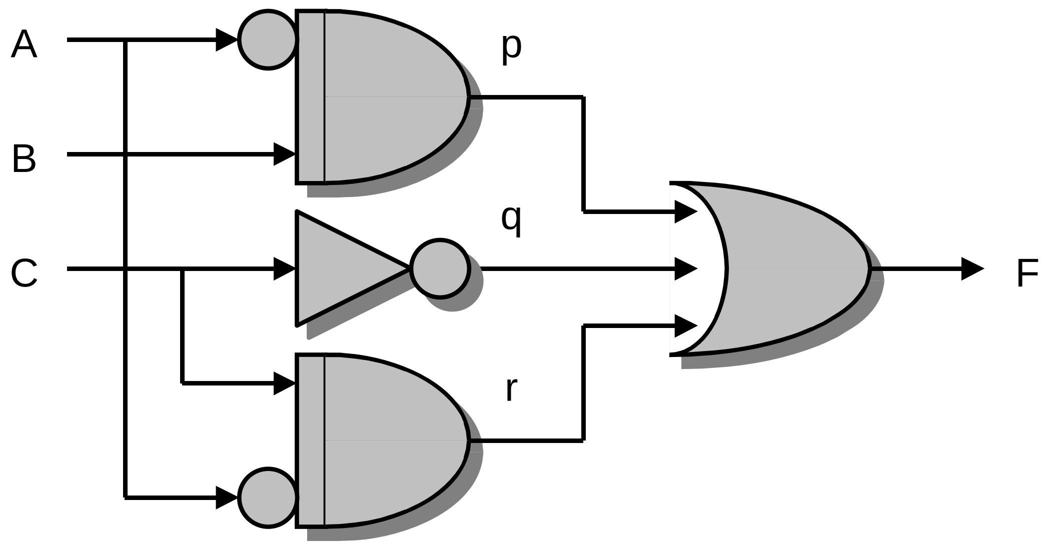


4a - Analisi di reti combinatorie
- dalla rete all'espressione

(2) Si ricavano le espressioni booleane corrispondenti ai segnali interni:

- p = !A B
- q = !C
- r = !A C

schema logico



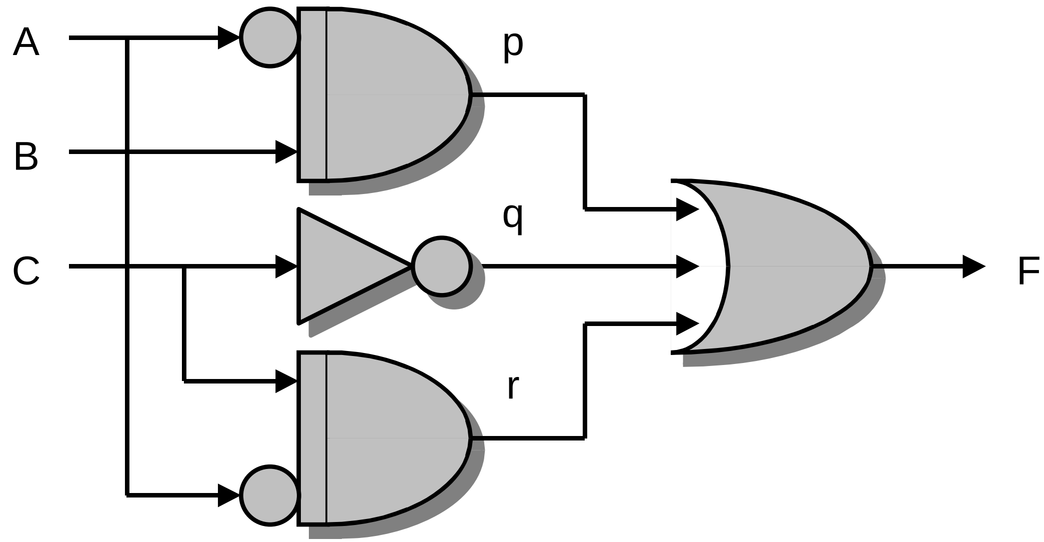
(3) Si ricava l’uscita come espressione booleana in funzione dei segnali interni:

- F = p + q + r

(4) Per sostituzione, si ricava l’uscita come espressione booleana in funzione degli ingressi principali:

- F = p + q + r
- F(A, B, C) = !A B + !C + !A C

L’espressione booleana così trovata ha una struttura conforme allo schema logico di partenza



4a - Analisi di reti combinatorie
- dall'espressione alla tabella delle verità

(per comodità è riportato anche il calcolo)

# riga	A	B	C	/A B + /C + /A C	F
0	0	0	0	/0 0 + /0 + /0 0	1
1	0	0	1	/0 0 + /1 + /0 1	1
2	0	1	0	/0 1 + /0 + /0 0	1
3	0	1	1	/0 1 + /1 + /0 1	1
4	1	0	0	/1 0 + /0 + /1 0	1
5	1	0	1	/1 0 + /1 + /1 1	0
6	1	1	0	/1 1 + /0 + /1 0	1
7	1	1	1	/1 1 + /1 + /1 1	0

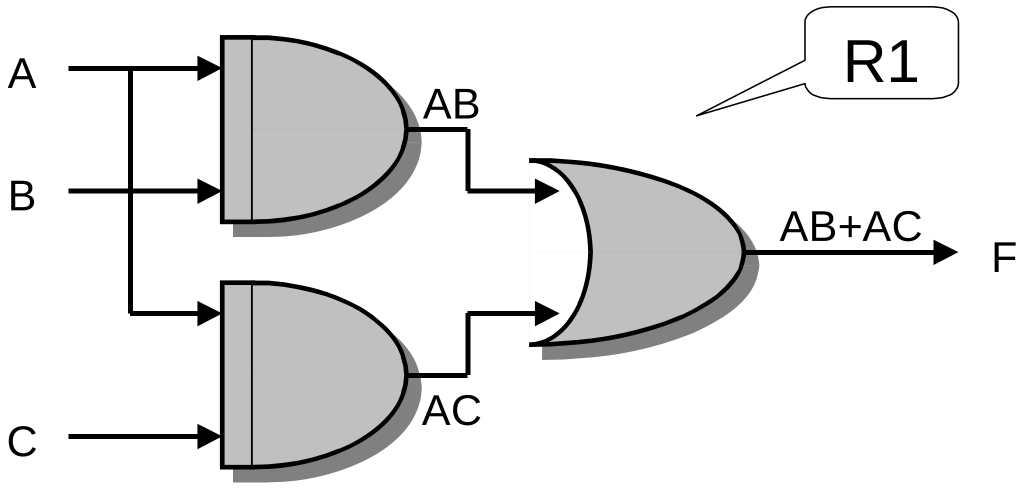
Reti combinatorie equivalenti

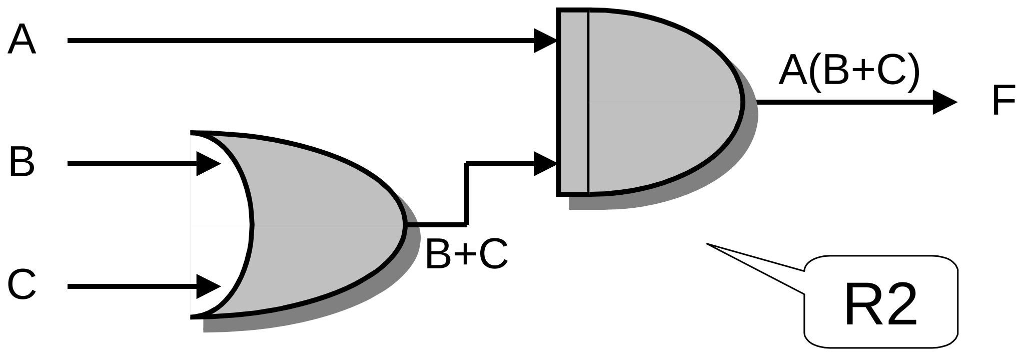
- Una funzione combinatoria può ammettere più reti combinatorie differenti che la sintetizzano

- Reti combinatorie che realizzano la medesima funzione combinatoria si dicono equivalenti

- Esse hanno tutte la stessa funzione (tabella delle verità), ma possono avere struttura (e costo) differente

Due reti equivalenti





F1 = AB + AC

F2 = A(B + C)

Trasformazione:

F1 = AB + AC =

= A(B + C) =

= F2

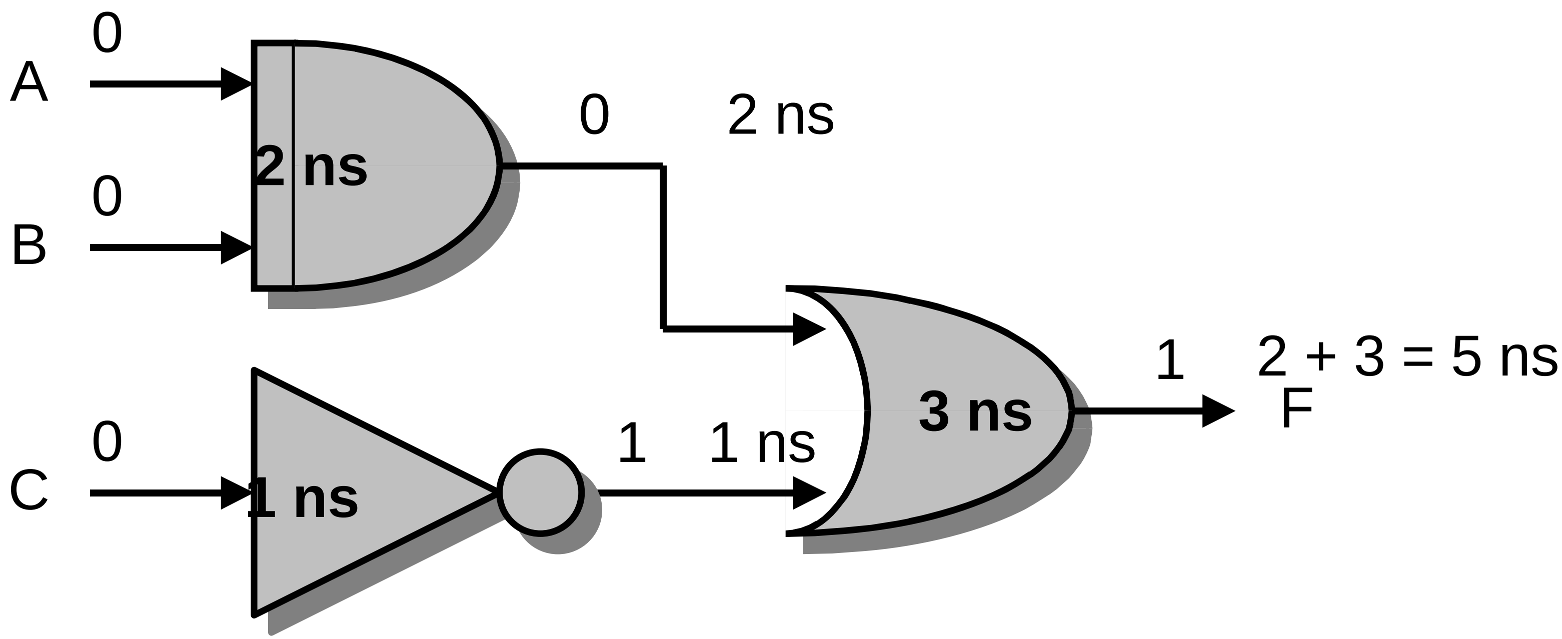
(prop. distributiva)

Costo e velocità di reti combinatorie

- Il costo di una rete combinatoria si valuta in vari modi (criteri di costo):
  - Numero di porte, per tipo di porta e per quantità di ingressi della porta
  - Numero di porte universali (NAND o NOR)
  - Numero di transistor
  - Complessità delle interconnessioni
  - e altri ancora …

- La velocità di una rete combinatoria è misurata dal tempo che una variazione di ingresso impiega per modificare l’uscita della rete (o ritardo di propagazione)
  - Per calcolare la velocità di una rete combinatoria, occorre conoscere i ritardi di propagazione delle porte logiche componenti la rete, e poi analizzare i percorsi ingressi-uscita

Velocità: esempio

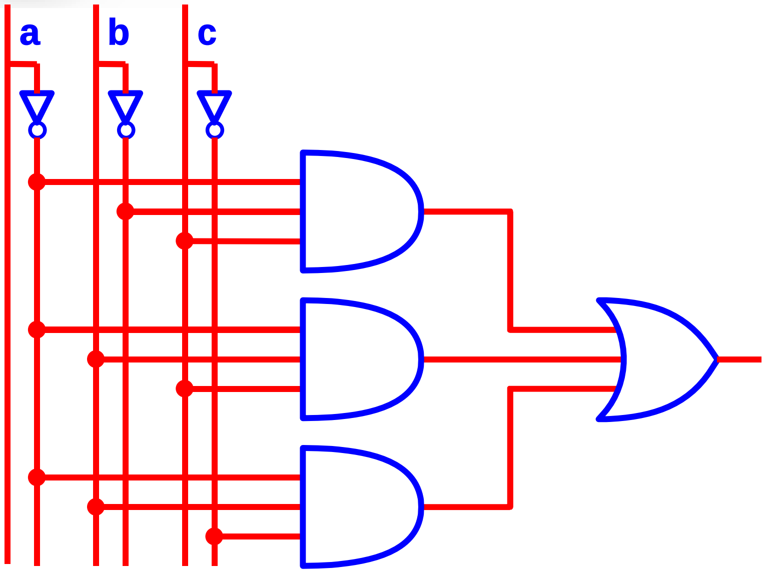


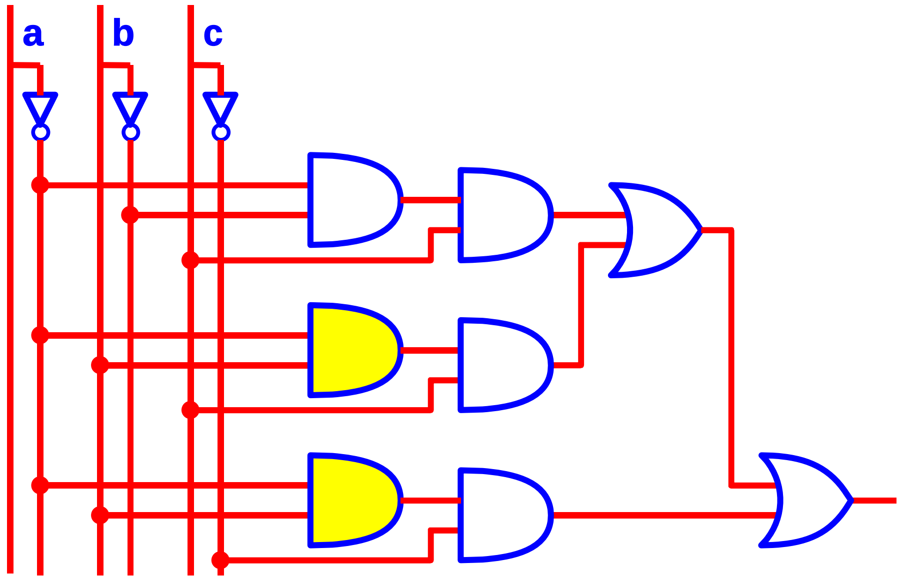
Ritardo totale = 5 ns = 5 10^{-9} sec

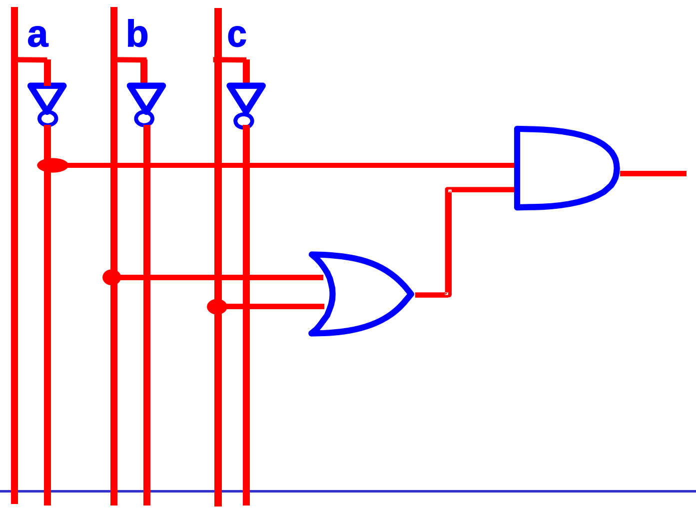
Freq. di commutazione = 1 / 5 ns = 200 MHz

Costo e ritardo di propagazione di reti equivalenti (porte a 2 ingressi)

F(a,b,c)=!a!bc + !abc + !ab!c = !a(c+b)







Porte a 2 ingressi

AND: Costo = 4, Rit. = 10

OR: Costo = 4, Rit. = 12

NOT: Costo = 1, Rit. = 2

4b - Sintesi di reti combinatorie

- La sintesi di una rete combinatoria espressa come tabella delle verità, consiste nel ricavare lo schema logico (il circuito digitale) che calcola la funzione combinatoria
- In generale, per una data tabella delle verità possono esistere più reti combinatorie (la soluzione al problema di sintesi non è dunque unica)
- Esistono svariate procedure di sintesi di reti combinatorie, che differiscono per:
  - Complessità della procedura di sintesi
  - Ottimalità della rete combinatoria risultante, per dimensioni e velocità

- Data una *funzione booleana,* la soluzione iniziale al problema di determinare una sua espressione consiste nel ricorso alle *forme canoniche.*

- Le forme canoniche sono, rispettivamente, la forma *somma di prodotti* (SoP) (sintesi in 1ª forma canonica) e quella *prodotto di somme* (PoS) (sintesi in 2ª forma canonica).

- Data una funzione boolena esistono una ed una sola forma canonica SoP ed una e una sola forma PoS che la rappresenta.

Sintesi in 1^a forma canonica

(o sintesi come somma di prodotti)

Data una tabella delle verità, a $n \geq 1$ ingressi, della funzione da sintetizzare, la **funzione F** che la realizza può essere **specificata** come

- la **somma logica** (**OR**) di tutti (e soli) i **termini prodotto** (**AND**) delle **variabili di ingresso corrispondenti agli 1** della funzione.
- Ogni **termine prodotto** (o **mintermine**) è costituito dal prodotto logico delle variabili di ingresso (letterale) prese in forma naturale se valgono 1, in forma complementata se valgono 0.

1ª forma canonica

- Si consideri il seguente esempio:

a	b	f(a,b)
0	0	0
0	1	1
1	0	0
1	1	1

- È intuitivo osservare che la funzione possa essere ottenuta dal OR delle seguenti funzioni:

a	b	f(a,b)
0	0	0
0	1	1
1	0	0
1	1	1

=

a	b	f_1(a,b)
0	0	0
0	1	1
1	0	0
1	1	0

+

a	b	f_2(a,b)
0	0	0
0	1	0
1	0	0
1	1	1

1ª forma canonica

- Per cui, intuitivamente, si ottiene:

a b	f(a,b)
0 0	0
0 1	1
1 0	0
1 1	1

=

a b	f_1(a,b)
0 0	0
0 1	1
1 0	0
1 1	0

+

a b	f_2(a,b)
0 0	0
0 1	0
1 0	0
1 1	1

$f_1(a,b)=a'b$ $f_2(a,b)=ab$

- Poiché, ad esempio, quando a=0 e b=1 il prodotto a'b assume valore 1 mentre vale 0 in tutti gli altri casi.

1ª forma canonica

- Ne consegue:

a b	f(a,b)
0 0	0
0 1	1
1 0	0
1 1	1

=

a b	f_1(a,b)
0 0	0
0 1	1
1 0	0
1 1	0

+

a b	f_2(a,b)
0 0	0
0 1	0
1 0	0
1 1	1

$$f(a,b) = a'b + ab$$

- Mettendo in OR i *mintermini* della funzione si ottiene l'*espressione booleana* della funzione stessa espressa come somma di prodotti.
  - nel *mintermine* (prodotto) una variabile compare nella forma naturale (x) se nella corrispondente configurazione di ingresso ha valore 1, nella forma complementata (x') se ha valore 0

Esempio: funzione maggioranza

- Si chiede di sintetizzare (in 1ª forma canonica) una funzione combinatoria dotata di 3 ingressi A, B e C, e di un'uscita F, funzionante come segue:
  - Se la maggioranza degli ingressi vale 0, l'uscita vale 0
  - Se la maggioranza degli ingressi vale 1, l'uscita vale 1

Tabella delle verità

- L'uscita vale 1 se e solo se 2 o tutti e 3 gli ingressi valgono 1 (cioè se e solo se il valore 1 è in maggioranza)

# riga	A	B	C	F
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

mintermini

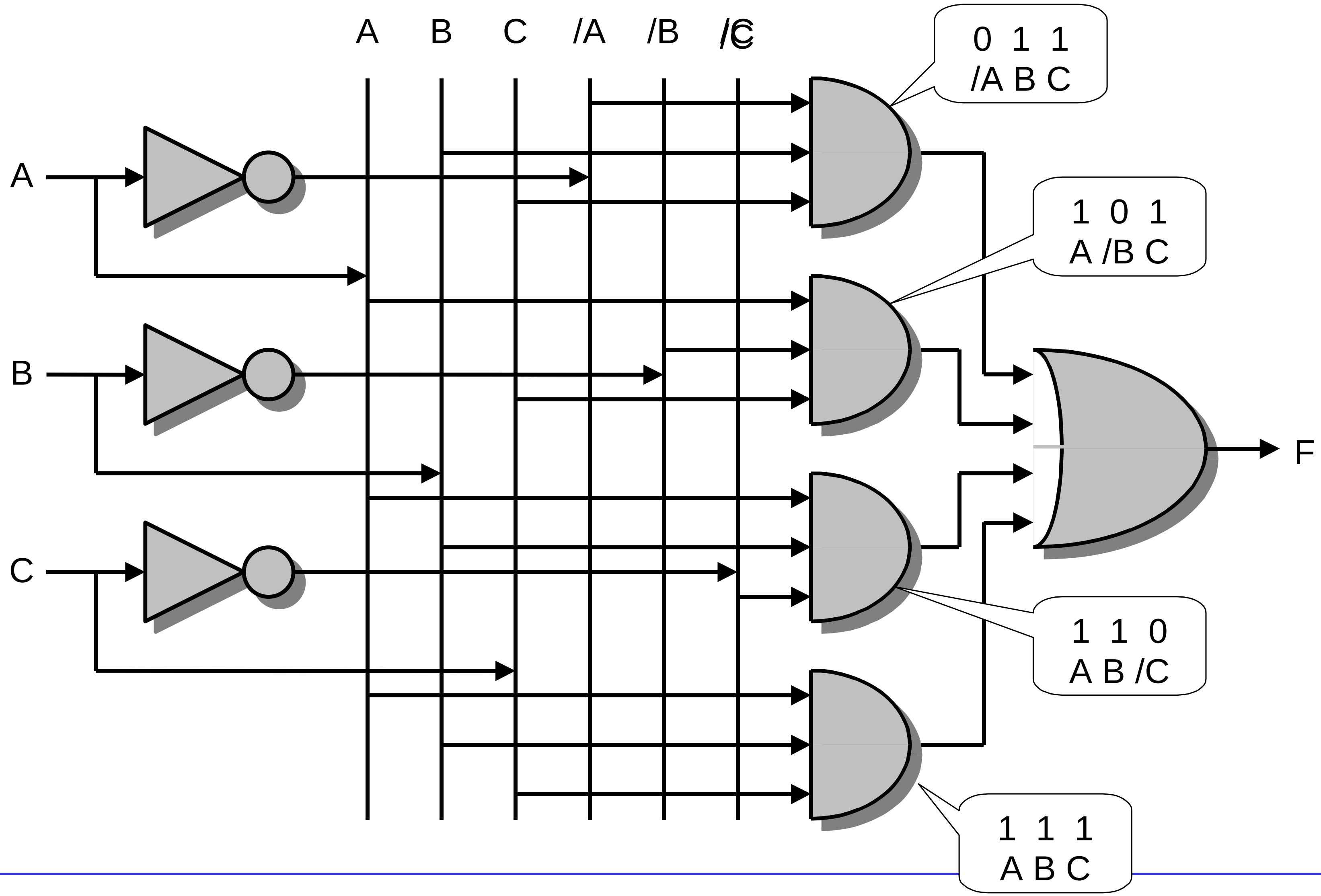
Espressione booleana

- Dalla **tabella della verità** si ricava, tramite la sintesi in 1a forma canonica (somma di prodotti), **l'espressione booleana** che rappresenta la funzione

- F(A, B, C) = !A B C + A !B C + A B !C +
  + A B C

- Dall'**espressione booleana** si ricava la **rete combinatoria** (il circuito) che è sempre a 2 livelli

Rete combinatoria: schema logico



Sintesi in 2ª forma canonica
(o sintesi come prodotto di somme)

Data una tabella delle verità, a $n \geq 1$ ingressi, della funzione da sintetizzare, la **funzione F** che la realizza può essere **specificata** come

- la **prodotto logico** (**AND**) di tutti (e soli) i **termini somma** (**OR**) delle **variabili di ingresso corrispondenti agli 0** della funzione.
- Ogni **termine somma** (o **maxtermine**) è costituito dalla somma logica delle variabili di ingresso (letterale) prese in forma naturale se valgono 0, in forma complementata se valgono 1.

2ª forma canonica

- Si consideri nuovamente lo stesso esempio:

a b	f(a,b)
0 0	0
0 1	1
1 0	0
1 1	1

- È intuitivo osservare che la funzione possa essere ottenuta dall'AND delle seguenti funzioni:

a b	f(a,b)
0 0	0
0 1	1
1 0	0
1 1	1

=

a b	f_1(a,b)
0 0	0
0 1	1
1 0	1
1 1	1

*

a b	f_2(a,b)
0 0	1
0 1	1
1 0	0
1 1	1

2ª forma canonica

- Per cui, intuitivamente, si ottiene:

a b	f(a,b)
0 0	0
0 1	1
1 0	0
1 1	1

=

a b	f_1(a,b)
0 0	0
0 1	1
1 0	1
1 1	1

*

a b	f_2(a,b)
0 0	1
0 1	1
1 0	0
1 1	1

$f_1(a,b) = a+b$ $\qquad$ $f_2(a,b) = a'+b$

Infatti, ad es., quando a=0 e b=0 allora (a+b) assume valore 0 mentre vale 1 in tutti gli altri casi.

f(a,b) = (a+b) * (a'+b)

Mettendo in AND i *maxtermini* della funzione si ottiene l'*espressione booleana* della funzione stessa espressa come prodotto di somme.

nel *maxtermine* (somma) una variabile compare nella forma naturale (x) se nella corrispondente configurazione di ingresso ha valore 0, nella forma complementata (x') se ha valore 1

Sintesi POS

- F=(A+B+C)(A+B+!C)(A+!B+C)(!A+B+C)

maxtermini

# riga	A	B	C	F
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1