Circuiti Combinatori

Corso di Architettura degli Elaboratori 1

Fulvio Valenza

A.A. 2018-2019

Polo di Vercelli

Circuito Digitale

- Uno fra i principali utilizzi dell'algebra booleana risulta la definizione (e realizzazione) di Circuiti Digitali

- Un Circuito Digitale è un interconnessione di Porte Logiche, dette **Reti Logiche**

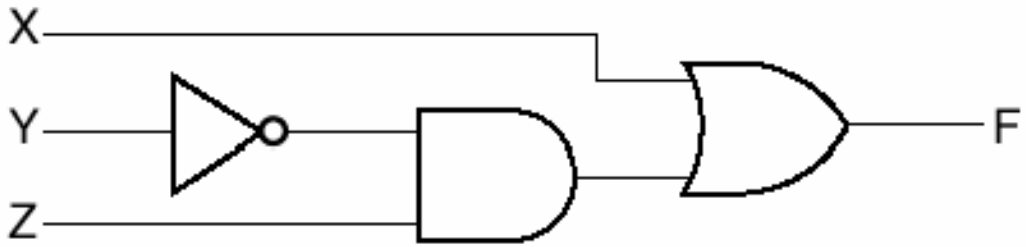


Fig. 2-3 Logic Circuit Diagram for $F = X + \overline{Y}Z$

Porte Logiche

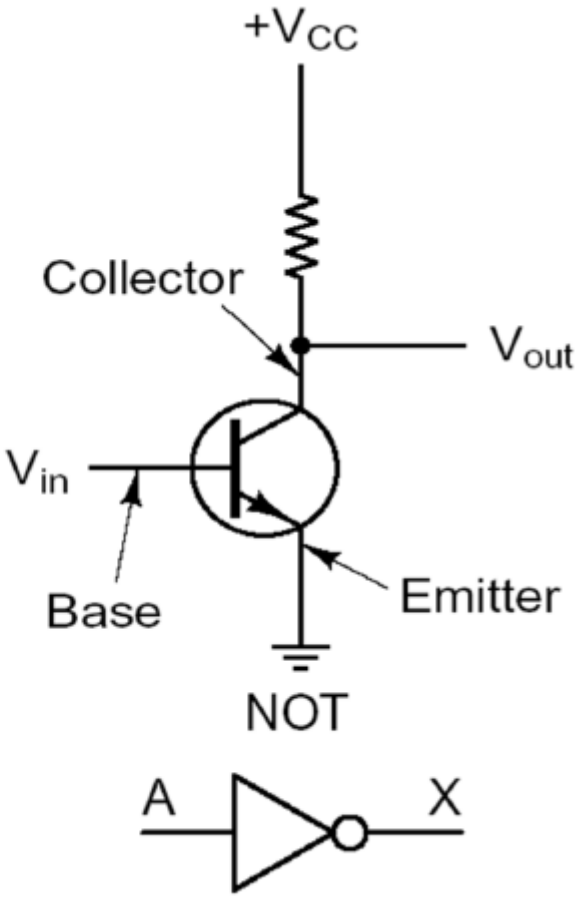
- Sono circuiti elettronici che producono come "segnale" di uscita il risultato di un'operazione booleana sui suoi ingressi. Possiamo quindi considerarle l'implementazione circuitale degli operatori logici booleani
- Ingressi ed uscite delle porte logiche sono segnali elettrici digitali. In questo corso non ci occuperemo della realizzazione circuitale. Considereremo pertanto ingressi ed uscite solo dal punto di vista "logico"
- Ogni porta è definita come simbolo grafico, notazione algebrica, o con la tabella di verità.
  - Secondo lo standard IEEE.
- Gli ingressi alla porta possono essere 1, 2 o più.

Porte Logiche

- Nei fili che interconnettono le porte logiche in un circuito sono presenti solo due valori logici:
  - Il valore **0**, rappresentato da un segnale compreso tra 0 e 1 volt.
  - Il valore **1**, rappresentato da un segnale compreso tra 3 e 5 volt.
- Esistono almeno cinque tipi diversi di porte logiche: **NOT**, **NAND**, **NOR**, **AND**, **OR**.
- Ogni porta logica ha uno o più **ingressi** e un'**uscita**.
- Ogni circuito ha uno o più **ingressi** e una o più **uscite**.
- Gli ingressi di un circuito possono essere connessi:
  - Ad uno degli ingressi di una porta logica.
  - Ad una delle uscite del circuito.
- In un circuito, l'uscita di una porta logica può essere connessa:
  - Ad uno degli ingressi di un'altra porta logica.
  - Ad una delle uscite del circuito.

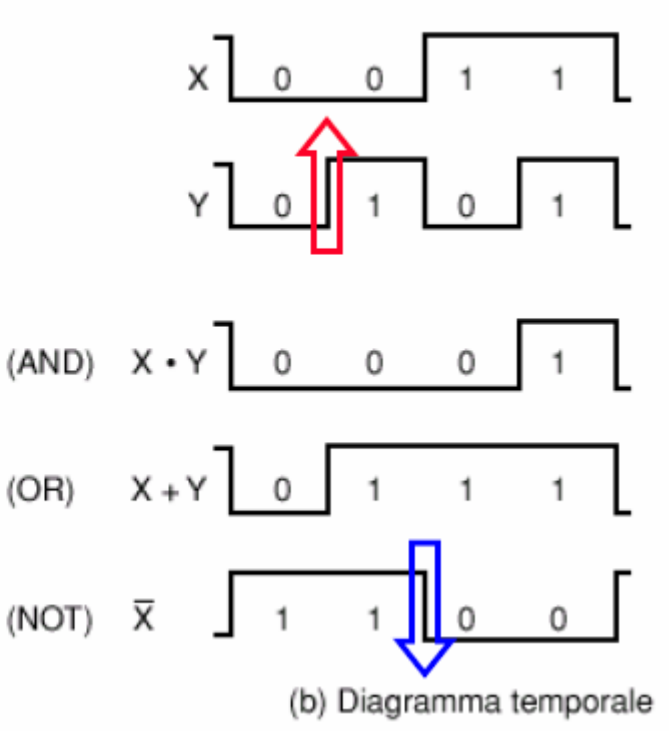
Porte Logiche

- Le porte logiche vengono realizzate attraverso dispositivi elettronici chiamati "transistor"
- Questi dispositivi, opportunamente alimentati, sono in grado di fornire segnali interpretabili come bit: basso livello di segnale come 0, alto livello di segnale come 1
- A fianco mostriamo un esempio di porta NOT realizzata con transistor "bipolari"
- Quando Vin è superiore a una certa soglia, il transistor funziona da corto circuito
- Altrimenti il transistor funziona da circuito aperto



Porte Logiche

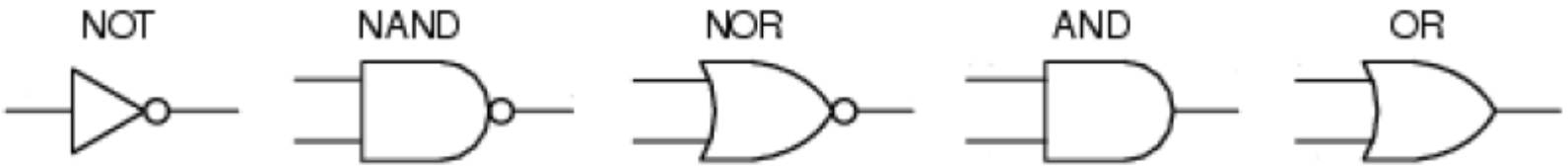
- Diagramma di temporizzazione
  - *I segnali sono campionati ad intervalli regolari*
  - *Le transizioni da un livello logico all'altro si assumono ideali, ovvero che avvengano istantaneamente*
  - ***Fronte di salita*** ⇑*: transizione da 0 a 1*
  - ***Fronte di discesa*** ⇓*: transizione da 1 a 0*



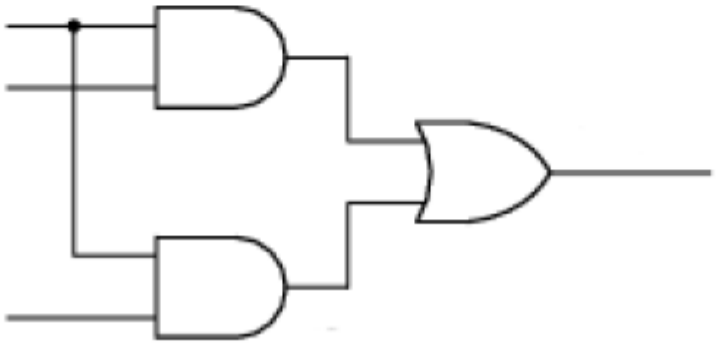
(b) Diagramma temporale

Notazione Grafica per i Circuiti Digitali

Porte Logiche

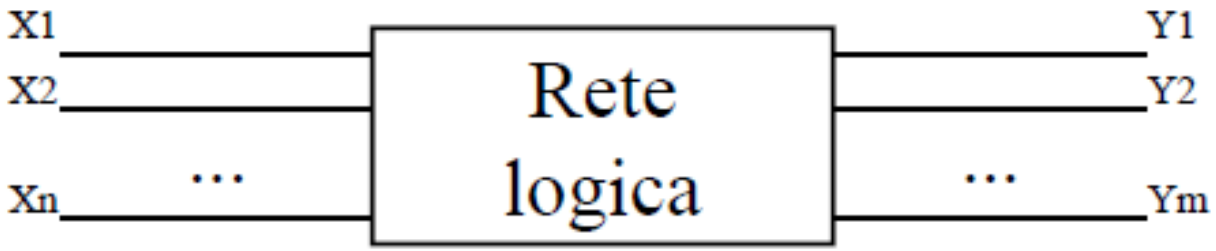


Circuito di Esempio



Reti Logiche

- Una rete logica è un insieme di blocchi funzionali realizzati mediante porte logiche ed elementi di memoria
- E' caratterizzata da n variabili "di ingresso" e m variabili "di uscita"
- Le reti logiche si dividono in:
  - reti senza memoria, o reti combinatorie
  - reti con memoria, o reti sequenziali



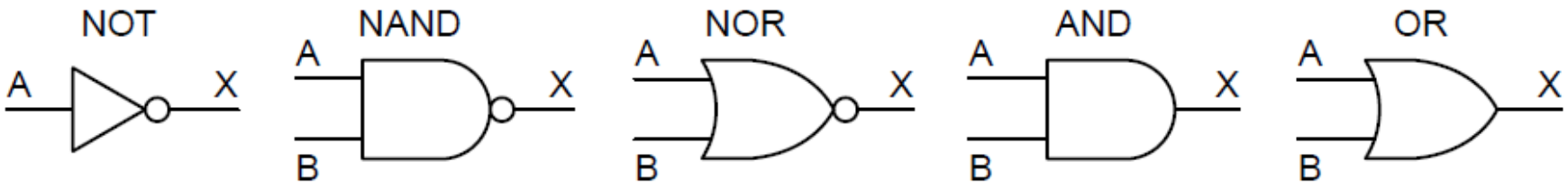
Reti Combinatorie

- Le reti combinatorie sono circuiti digitali che **non contengono cicli**.
  - In altre parole, non possono esistere cammini ciclici all'interno del circuito.
- Un **input** per il circuito consiste in una sequenza di valori binari, uno per ogni ingresso del circuito.
- Un **output** per un circuito consiste in una sequenza di valori binari, uno per ogni uscita del circuito.

Reti Combinatorie

- L'output di una rete combinatoria in corrispondenza ad un certo input si può calcolare come segue:
  - Per ogni porta logica, se sono noti i valori binari in corrispondenza dei suoi ingressi, allora sarà noto anche il valore binario in corrispondenza della sua uscita.
  - Le regole che permettono di determinare, per ogni porta logica, quale sia il valore della sua uscita in corrispondenza di ciascun valore per le entrate dipendono solo dal tipo di porta.
- Si può supporre, a questo punto, che il **tempo** necessario alla valorizzazione dell'uscita di una porta logica sia trascurabile.

Funzione delle porte logiche



A	X
0	1
1	0

(a)

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

(b)

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

(c)

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

(d)

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

(e)

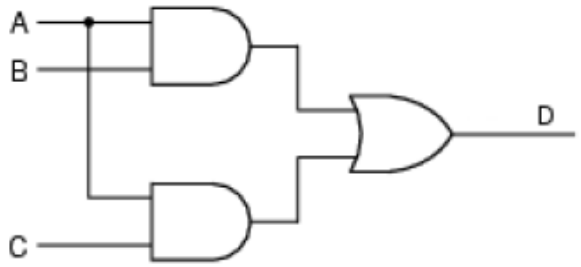
Analisi e Sintesi di una rete combinatoria

- Analisi
  - dall'esame delle porte logiche che compongono la rete capire quale è la funzione implementata
- Sintesi
  - dall'analisi dei requisiti, ovvero dalle corrispondenze ingressi-uscite, implementare la funzione richiesta
  - utilizzare il minimo numero possibile di porte logiche ed ingressi. In altre parole, dalla tabella di verità, "sintetizzare" la rete "minima"
  - si desidera che il numero di porte/ingressi sia il più piccolo possibile
    - riduzione di spazio sul 'chip'
    - costo del 'chip'

Dai Circuiti alle funzioni (Analisi)

- Ad ogni uscita di ogni rete combinatoria corrisponde una funzione booleana.
  - Prima di tutto associamo n variabili booleane $A_1, \ldots, A_n$ agli n ingressi e una variabile C all'uscita che ci interessa
  - Per ogni assegnamento di valori di verità alle variabili booleane $A_1, \ldots, A_n$, il corrispondente valore di C sarà quello ottenuto valutando il circuito.
  - In questo modo, si possono associare funzioni booleane ad ogni uscita del circuito. Diremo che il circuito in questione **implementa** tali funzioni booleane.
- Due circuiti si dicono **equivalenti** se implementano la stessa funzione.
- Una volta costruita una funzione, si potrà poi ricavare un'espressione booleana

Dai Circuiti alle funzioni: Esempio



$\Rightarrow$

A	B	C	D
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$\Downarrow$

$$A\overline{B}C + AB\overline{C} + ABC$$

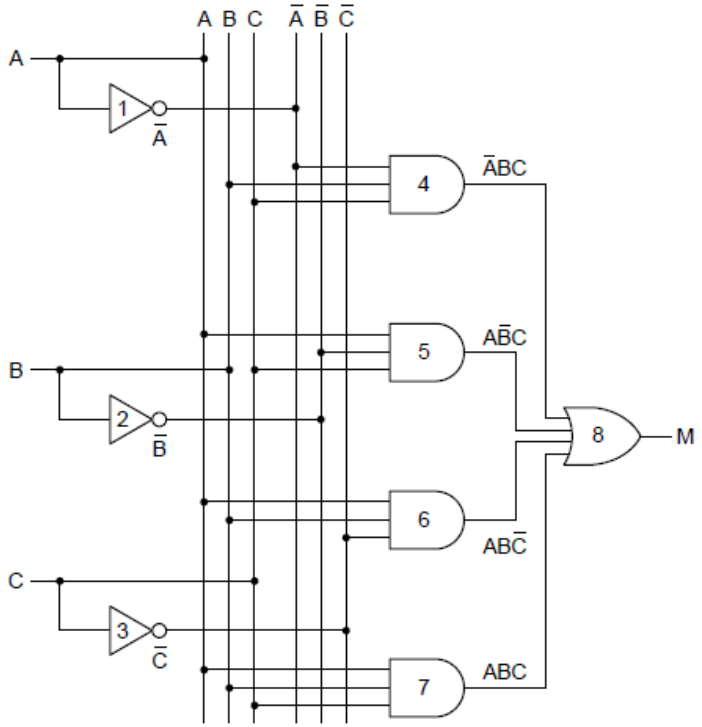
Dalle Espressioni ai Circuiti

- Ad ogni sequenza di espressioni booleane sulle stesse variabili corriponde un circuito.
  - Data un'espressione booleana E contenente le variabili booleane $A_1, \ldots, A_n$, un circuito con n entrate e un'uscita si può costruire seguendo la struttura di E.
  - La funzione implementata dal circuito sarà la funzione corrispondente ad E.
  - Partendo da una sequenza di espressioni, si potrà costruire un **singolo** circuito con più uscite.
- Per ottenere le espressioni booleane di partenza, si può procedere applicando ad altrettante funzioni booleane la procedura vista in precedenza.
  - In questo modo, si può passare da n funzioni booleane su m variabili ad un circuito con m ingressi ed n uscite.

Dalle Espressioni ai Circuiti

A	B	C	M
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

(a)



(b)

$$\overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

Sintesi

- Supponiamo di voler costruire una rete combinatoria che soddisfi certe proprietà.
- Prima di tutto: come possiamo **specificare** il comportanento che la rete combinatoria deve avere?
- Ciò che ci interessa è il suo comportamento funzionale.
  - La rete combinatoria dovrà avere un certo numero m di entrate.
  - La rete combinatoria dovrà avere un certo numero n di uscite.
  - Il suo comportamento atteso sarà poi specificato tramite n funzioni booleane in m variabili.
- Sappiamo poi che data una funzione booleana esiste sempre il modo di costruire un'espressione booleana ad essa corrispondente e, da quest'ultima, si passa facilmente ad una rete combinatoria.

Sintesi minima di una rete logica

- Significa trasformare una generica funzione booleana in una equivalente che abbia il minor numero possibile di termini (ovvero di porte) e di letterali (ovvero di ingressi).
- Possono essere usati tre metodi:
  - Semplificazione algebrica
    - Si usano le relazioni notevoli viste per ridurre un'espressione booleana in una con il minor numero di termini possibile.
  - Mappe di Karnaugh
    - Si basa sulla rappresentazione di una funzione su una mappa (tabella) di 2^n "celle" (n è il numero di variabili/ingressi alla rete).

Esempio di sintesi

- Progettare una rete combinatoria che, dati due ingressi A e B, presenti in uscita somma e riporto

Esempio di sintesi

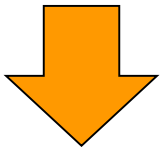
- 1) Determinare la tavola di verità

A	B	S	R
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Esempio di sintesi

- 2) Dalla Tavola ottenere l'espressione algebrica

A	B	S	R
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

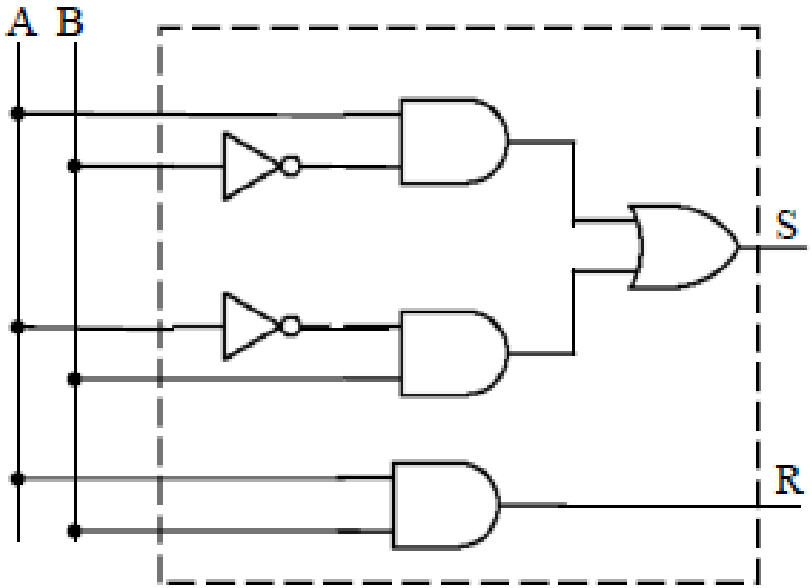


$$S = A\bar{B} + \bar{A}B\ ,\ R = AB$$

Esempio di sintesi

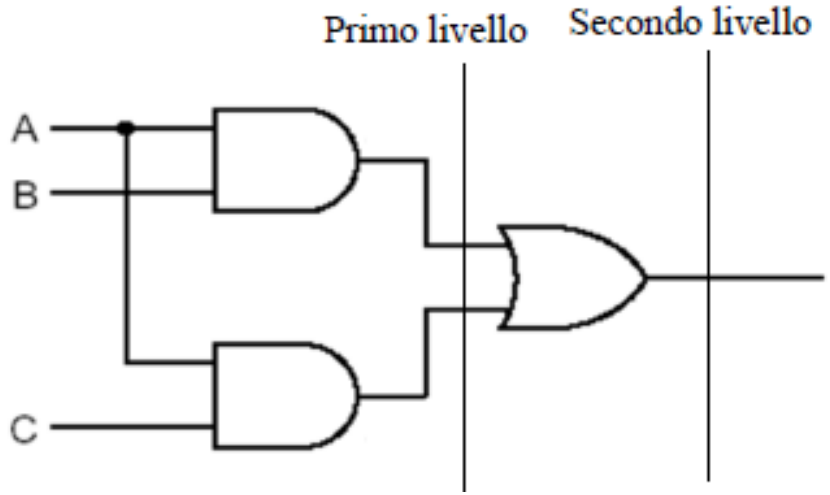
- 3) Semplificare l'espressione, tramite il metodo preferito
  - Nel nostro abbiamo già espressioni minime

- 4) Disegnare il circuito Minimo



Livelli di una rete combinatoria

- Una rete combinatoria è caratterizzata da un determinato numero di "livelli" di porte logiche
- Un livello è costituito dalle porte i cui ingressi ricevono i "bit" ("segnali" elettrici) nello stesso istante
- Spesso una rete combinatoria non presenta più di tre livelli di logica (porte NOT + porte AND + porte OR)



Operatori funzionalmente completi

- Le funzioni booleane sono state espresse in termini di operatori AND, OR e NOT
- Implementazioni a due livelli:
  - *SOP: AND-OR*
  - *POS: OR-AND*
- Esistono altri due operatori in grado di esprimere da soli gli altri tre:
  - *Operatore NAND, (NOT AND), simbolo (/)*
  - *Operatore NOR (NOT OR), simbolo (↓)*
- Una funzione booleana può quindi essere espressa
  - *In forma SOP usando solo NAND*
  - *In forma POS usando solo NOR*

Operatori funzionalmente completi

- Finora non abbiamo utilizzato molto le porte logiche **NAND** e **NOR**.
- Nelle implementazioni a due livelli c'è bisogno solo delle porte AND, OR, NOT.
  - Della porta NOT non c'è in realtà bisogno, perché si suppone di avere a disposizione due ingressi per ogni variabile: uno in cui la variabile è affermata e uno in cui è negata.
- Le porte NAND e NOR, a differenza delle altre, hanno un proprietà interessante: sono **universali**.
  - Ciò significa, in particolare, che ogni rete combinatoria può essere trasformata in una rete equivalente contenente solo la porta NAND o, in alternativa, contenente solo la porta NOR.

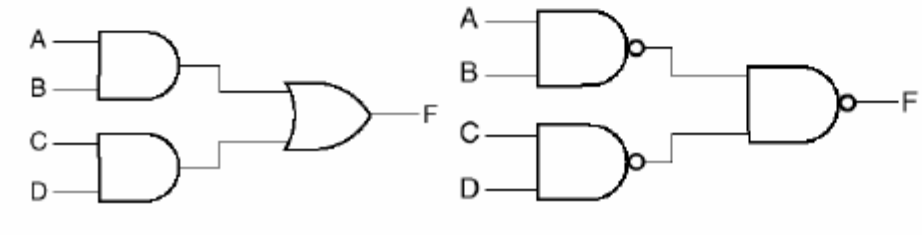
Proprietà delle NAND e NOR

- A / A = A'
  - $A/A = (A\ A)' = (A)' = A'$
- A / A' = 1
- A / 1 = A'
- A / 0 = 1

- A ↓ A = A'
- A ↓ A' = 0
- A ↓ 0 = A'
- A ↓ 1 = 0

- SOP
- F = A B + C D
- DeM => ((A B)' (C D)')'
- = (A/B)/(C/D)

- POS
- G = (A +B) (C+D)
- DeM =>((A + B)' + (C+D)')'
- = (A ↓ B) ↓ (C ↓ D)

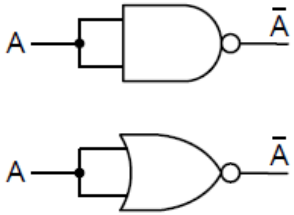


Operatori funzionalmente completi

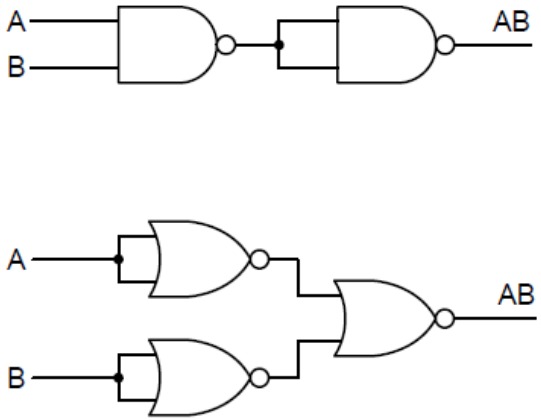
- Applicando le proprietà di De Morgan si possono ottenere simboli alternativi per le porte NAND e NOR:

NAND e NOR

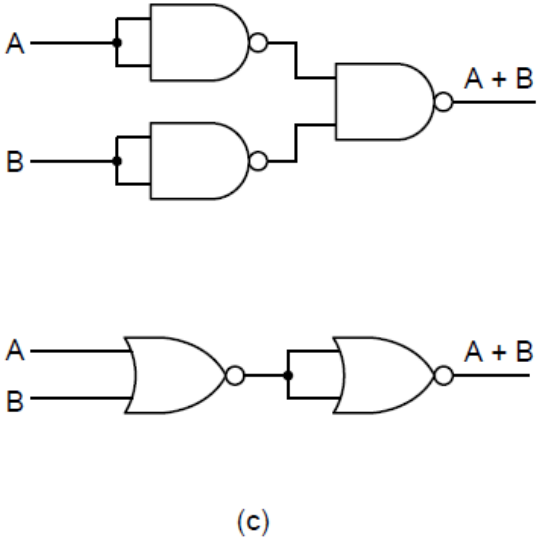
Tramite NAND e NOR si possono costruire reti combinatorie equivalenti alle porte NOT (in (a)), AND (in (b)) e OR (in (c))



(a)



(b)



(c)

Universalità

- Esistono poi un certo numero di tecniche alternative per dimostrare l'**universalità** delle porte NAND e NOR, ossia per dimostrare che per ogni funzione booleana F esistono:
  - Una rete combinatoria contenente soltanto porte NAND.
  - Una rete combinatoria contenente soltanto porte NOR.
- Per quanto riguarda la porta NAND, si può procedere come segue:
  1. Prima di tutto si costruisce un'espressione E in somma di prodotti per F.
  2. Poi si costruisce la rete combinatoria a due livelli corrispondente a E.
  3. Si trasformano poi tutte le porte AND in porte NAND e tutte le porte OR in porte NAND.

Universalità

- Per la porta NOR si procede in modo duale, costruendo un'espressione in prodotto di somme, la relativa rete a due livelli e trasformando porte OR e AND in porte NOR.
- Della **correttezza** di questa procedura ci si può rendere conto osservando il comportamento della rete combinatoria modificata.

Universalità

- Una **tecnica alternativa** è la seguente (per esempio nel caso della porta NAND):
  1. Si costruisce una rete combinatoria per la funzione F.
  2. Si rimpiazza ogni porta logica diversa da NAND presente nella rete con una rete combinatoria equivalente ad essa.

  Questa tecnica, però, produce una rete combinatoria avente un numero di porte logiche in generale maggiore rispetto a quello necessario nel caso della tecnica precedente.

Universalità

- Esiste poi una **terza possibilità** (di seguito nel caso della porta NAND):
  1. Si costruisce una rete combinatoria (contenente solo porte AND e OR) per la funzione F.
  2. Si rimpiazza ogni porta AND con un NAND e ogni porta OR con un NAND (nella notazione alternativa).
  3. Si controlla che in ogni filo ci siano un numero pari di pallini e, in caso contrario, si inserisce una rete equivalente alla porta NOT lungo il filo.
- Entrambe le tecniche illustrate sono facilmente adattabili a tecniche che dimostrino l'universalità della porta NOR.

Operatori XOR e NOR

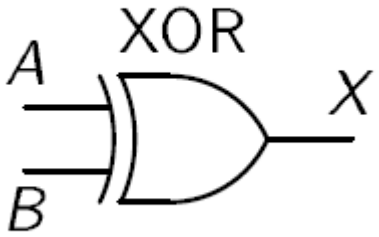
- Con **XOR** indicheremo sia una porta logica che un'operazione binaria sull'Algebra di Boole (quest'ultima indicata anche con $\oplus$).
- L'operazione $\oplus$ gode delle proprietà associativa e commutativa.
- Si può esprimere $\oplus$ nei termini delle operazioni di addizione, moltiplicazione e negazione come segue:

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

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

Operatori XOR e NOR

- Intuitivamente, si può pensare che il risultato dell'applicazione di $\oplus$ a due cifre binarie valga 1 se e solo se esattamente uno dei due operandi vale 1.
  - Esistono quindi due configurazioni (tra le quattro possibili) in cui il risultato dell'applicazione di $\oplus$ vale 1.
- Il simbolo $\oplus$ diventerà anche parte del linguaggio per le espressioni booleane



Operatori XOR e NOR

- Proprietà.

  $X \oplus 0 = X$

  $X \oplus 1 = X'$

  $X \oplus X = 0$

  $X \oplus X' = 1$

- Proprietà.

  $(X \oplus 0)' = X'$

  $(X \oplus 1)' = X$

  $(X \oplus X)' = 1$

  $(X \oplus X')' = 0$