Laboratorio di Segnali e Sistemi
- Capitolo 6 -
Elettronica digitale: circuiti combinatori

Claudio Luci

SAPIENZA
UNIVERSITÀ DI ROMA

last update : 070117

Sommario del capitolo:

- Numerazione binaria
- Somma e sottrazione di numeri binari, complemento a 2
- Algebra di Boole
- porte logiche
- Forma canonica di un'equazione logica
- Minimizzazione di circuiti
- Mappa di Karnaugh
- Famiglie logiche
- Sommatori logici
- Comparatore digitale
- Multipler, demultiplexer, encoder e decoder

  prossimo file
- Circuiti sequenziali: Flip-Flop (S-R, J-K, Master-Slave, D e T, edge triggered)
- Shift register
- Contatori
- DAC: a pesiera e con rete R-2R
- ADC

Introduzione

Introduzione all'elettronica digitale

- Finora abbiamo visto l'elettronica analogica, dove abbiamo studiato dei circuiti che manipolano il segnale (amplificatori, derivatori, sommatori, filtri, etc…) che non cambiano sostanzialmente la natura del segnale stesso , anzi, in certi casi (amplificatori) si vuole riprodurre la forma del segnale il più fedelmente possibile;
- Con l'elettronica digitale si cambia completamente modo di pensare, infatti le tecniche caratteristiche dell'elettronica analogica (tensioni, resistenza, caratteristiche di un transistor, etc..) passano in secondo piano, per cedere il posto ad un modo di ragionare basato sulla logica.
- Non si lavora più con singoli componenti che richiedono di essere correttamente inseriti in un circuito, ma su blocchi già di per sé completi, ognuno in grado di svolgere una determinata funzione.
- Tutte queste unità fondamentali possono essere collegate le une alle altre in schemi complessi quanto si vuole, unicamente seguendo un ragionamento logico, senza doversi preoccupare degli aspetti elettronici veri e propri.
- L'elettronica digitale utilizza, tra le varie unità funzionali, le cosiddette "porte logiche", tutte caratterizzate dall'avere uno o più ingressi o uscite. Tali componenti devono essere alimentati con la giusta tensione (tipicamente 5 V) ma in ingresso o in uscita il segnale può assumere solo due livelli: lo "0" logico e l'"1" logico. Il valore reale di tensione corrispondente a questi due livelli dipende dal tipo di logica utilizzata (TTL, ECL, CMOS, NIM, etc…), ma in linea di principio non dobbiamo preoccuparci di quest'aspetto del problema, purché si lavori in un ambiente coerente e compatibile.
- Le varie porte logiche possono essere combinate tra loro seguendo l'algebra di Boole e le varie grandezze numeriche devono essere espresse in forma binaria, dato che i segnali possono assumere solo due stati (0 oppure 1).

Numerazione binaria

Numerazione binaria

- La rappresentazione decimale di numeri interi positivi e' una rappresentazione **posizionale** in base **10**

$$(288)_{10} = 2 * 10^2 + 8 * 10^1 + 8 * 10^0$$

- il cui intervallo va da 0 a $10^N - 1$, con N uguale all'estensione della rappresentazione

- In perfetta equivalenza anche la rappresentazione binaria e' posizionale ma in base **2**. Dato un numero a *n* bits:

$$(x)_2 = x_{n-1}2^{n-1} + x_{n-2}2^{n-2} + \ldots + x_1 2^1 + x_0 2^0$$

- dove l'intervallo e' $0 : 2^n - 1$
- A 32 bits l'intervallo va da 0 a +4.294.967.295

Num. decimale	Num. binaria
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
...	...
16	10000
...	...
32	100000
...	...

- Il sistema binario, per il fatto di avere due soli numeri fondamentali, si presta ad essere associato a sistemi elettronici in cui le variabili elettriche assumono solo due valori possibili, quindi può essere utilizzato per costruire "macchine" per l'elaborazione numerica;
- Si possono definire, oltre alle normali operazioni aritmetiche, anche delle operazioni logiche, che costituiscono nel loro complesso, la cosiddetta algebra di Boole. Queste regole si appplicano a tutti gli insieme binari, cioè composti da due elementi: 0 ed 1, VERO e FALSO, BIANCO e NERO, etc..

Cambiamenti di base

- Per convertire da base *qualsiasi* a base 10 sfruttiamo il fatto che la rappresentazione e' posizionale:
  $(427)_8 = 4 * 8^2 + 2 * 8^1 + 7 * 8^0 = 4 * 64 + 2 * 16 + 7 = (279)_{10}$

- Per convertire da base 10 a base *qualsiasi* procediamo per divisioni successive:

```
35 : 2 = 17 resto 1
17 : 2 =  8 resto 1
 8 : 2 =  4 resto 0
 4 : 2 =  2 resto 0
 2 : 2 =  1 resto 0
 1 : 2 =  0 resto 1
```

$(35)_{10} = (100011)_2$

Vedremo ora le operazioni aritmetiche tra numeri binari e come trattare i numeri negativi

Parentesi: esistono anche altre rappresentazioni dei numeri, ad esempio:

MMXVIII = ?
MMXIX = ?

Operazioni tra numeri binari

Numerazione binaria: somma e sottrazione

- Somma e sottrazione di due numeri binari (senza segno) si eseguono nel modo seguente:

Addizione

Bit di riporto →

```
  111111
   1011011000000 +
  10110110000000 =
-----------------------
 100010001000000
```

Sottrazione (come in base 10 ma occhio ai "prestiti")

0→1 0→1 0→10 1→0

```
  1 1 1
  1 0 0 0 -      (8)
  0 1 0 1 =      (5)
-----------------------------
  0 0 1 1
```

decimale

0→9 5→15

```
-1 -1
3  0  5  –
2  8  6  =
---------
0  1 9
```

- Dobbiamo introdurre i numeri negativi;
- La sottrazione, per quanto semplice, è un gran pasticcio. Dobbiamo introdurre una rappresentazione dei numeri binari che permetta di gestire con lo stesso algoritmo (vale a dire con lo stesso circuito logico) sia l'addizione che la sottrazione.

Numerazione binaria: il bit di segno

- Nel sistema decimale i numeri negativi vengono distinti dai positivi tramite i simboli + e -;
- Nel sistema binario si introduce un bit di segno, a sinistra del bit più significativo (MSB):
  0 indica un numero positivo e 1 un numero negativo

$$(+15)_{10} = 0\,1111 \quad ; \quad (-15)_{10} = 1\,1111$$

- **Complemento a 1** di un numero binario: sostituzione di 0 con 1 e di 1 con 0 (tranne il bit di segno):

$$0\,010100 \rightarrow 0\,101011 \; ; \quad 1\,101111 \rightarrow 1\,010000$$

- **Complemento a 2** di un numero binario: si fa il complemento a 1 e si aggiunge 1 al risultato:

$$0\,1011 \rightarrow 0\,0100 + 1 = 0\,0101 \; ; \quad 1\,1011 \rightarrow 1\,0100 + 1 = 1\,0101$$

- Esempio: scriviamo il numero -25 con 6 bit (uno per il segno) nella rappresentazione signed e in quella complemento a 2:

$$(+25)_{10} = 0\,11001 \quad ; \quad (-25)_{10} = 1\,11001$$

Complemento a 2

$$1\,00110+1 = 1\,00111$$

- Vi sono due modi per ricavare il numero negativo dal binario scritto in complemento a 2:

$$1\,00111 = -1\cdot 2^5 + 1\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0 = -32+4+2+1 = -25$$

Oppure complemento a 1 più 1: $1\,00111 \Rightarrow 1\,11000+1 = 1\,11001 = -25$

Numerazione binaria: sottrazione

- Facciamo un esempio di sottrazione con la rappresentazione complemento a 2 (usando 6 bit):

$$12-25=12+(-25)=-13$$

Consideriamo la sottrazione come una somma tra numeri relativi

$$(+12)_{10} = 0\,01100 \;;\; (-25)_{10} = 1\,11001 \rightarrow 1\,00111$$

```
 0 01100  +
 1 00111  =
---------------
 1 10011
```

$$1\,10011 \Rightarrow 1\,01100+1=1\,01101=-13$$

- Con 6 bit, di cui uno viene usato per il segno, possiamo trattare solo i numeri nel range:

da 1 11111 (-31) a 0 11111 (+31)

- Se le operazioni di somma o sottrazione producono un numero al di fuori di questo range, il risultato perde di significato. Un bit di overflow tiene conto di questa eventualità.

- Fare somme o sottrazioni di numeri binari con carta e penna non è molto comodo, però ci sono dei dispositivi logici che lo possono fare al posto nostro.

Moltiplicazione tra numeri binari

- Moltiplicare un numero per 2 equivale a spostare tutti i bit di una posizione verso sinistra;
- Moltiplicare per una potenza di 2 vuol dire ripetere questa operazione più volte;
- Moltiplicare un numero per un altro che non sia una potenza di 2, si scompone quest'ultimo in somma di potenze di 2 e si sommano i risultati. Facciamo ad esempio 27 x 33:

$$27 \times 33 = 27 \times (32 + 1) = 27 \times 2^5 + 27 \times 2^0$$

$27 = 11011$

$27 \times 2^5 = 1101100000 +$

$27 \times 2^0 = \quad 11011 =$

1101111011

$$1101111011 = 2^9 + 2^8 + 2^6 + 2^5 + 2^4 + 2^3 + 2^1 + 2^0 = 512 + 256 + 64 + 32 + 16 + 8 + 2 + 1 = 891$$

- Nella divisione invece si spostano i bit verso destra
- Vedremo dei circuiti che effettuano traslazioni di bit verso destra o verso sinistra.

Algebra di Boole e porte logiche

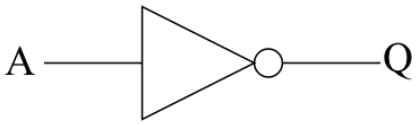
Algebra di Boole: introduzione

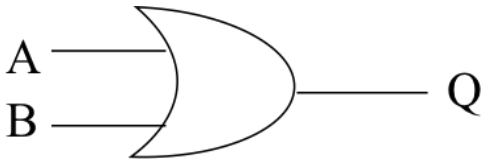
- Nel 1854, George Boole (1815-1864), un matematico inglese, scrisse il suo libro ormai divenuto un classico, intitolato ***Investigazioni delle leggi del pensiero***, in cui egli propose un'algebra che rappresentasse simbolicamente i problemi della logica, in modo che essi potessero essere investigati matematicamente.
- L'intenzione di Boole era quella di fornire un mezzo per stabilire sistematicamente la validità di dichiarazioni logiche complesse, che comportassero proposizioni che fossero soltanto **vere** o **false**.
- Oggi, i sistemi matematici fondati sullo studio di Boole sono chiamati *algebre booleane* in suo onore.
- L'applicazione dell'algebra booleana ad alcuni problemi di ingegneria fu introdotta nel 1938 da Claude E. Shannon, allora al MIT (Massachusetts Insititute of Technology). Shannon dimostrò come l'algebra booleana si poteva applicare al progetto di reti relè nei sistemi telefonici.
- All'apparire dei computer digitali fu chiaro che quest'algebra si poteva impiegare per la realizzazione di qualsiasi sistema di tipo logico, formato da elementi con caratteristiche a due valori.
- L'algebra booleana serve per descrivere convenientemente le proprietà ai terminali di reti che si trovano nei sistemi digitali. Tra le espressioni algebriche e le loro realizzazioni circuitali c'è una correlazione biunivoca.
- Poiché le espressioni algebriche sono soggette a manipolazioni e semplificazioni, ne segue che i diversi modi di scrivere le espressioni forniscono descrizioni di reti differenti, le quali possono essere costruite per ottenere lo stesso comportamento ai terminali.
- Le reti logiche dell'algebra booleana sono divise in due categorie generali: **reti combinatorie** e **reti sequenziali**. L'uscita di una reta combinatoria è univocamente determinata dal valore degli ingressi, mentre in una rete sequenziale essa dipende anche dallo stato pregresso ed hanno bisogno quindi di un elemento di memoria.

Algebra di Boole

La funzione logica di un circuito combinatorio e' completamente descritta e specificata da una *Equazione Logica*

- Le variabili (i segnali...) in input e output sono *variabili logiche*.
- la funzione e' la combinazione dei 3 *operatori fondamentali* dell'algebra booleana
  - **OR** $(A + B)$ out $= 1$ *se almeno un input* $= 1$
  - **AND** $(A * B)$ out $= 1$ *se tutti gli input* $= 1$
  - **NOT** $(\overline{A})$ out $=$ inverso dell'input





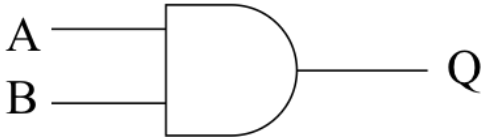


Table: Operatore NOT

A	Q
0	1
1	0

Non segue l'algebra "normale"

Table: Operatore OR

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

Table: Operatore AND

A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

Circuito combinatorio: l'uscita è perfettamente determinata se si conoscono tutti gli ingressi

Proprietà dell'algebra booleana

Proprieta' degli operatori:

Identita':	$A + 0 = A$	$A * 1 = A$
Nullo:	$A + 1 = 1$	$A * 0 = 0$
Idempotente:	$A + A = A$	$A * A = A$
Inverso:	$A + \overline{A} = 1$	$A * \overline{A} = 0$

Proprieta' dell'algebra:

Commutativa:	$A + B = B + A$	$A * B = B * A$
Associativa:	$A + (B + C) = (A + B) + C$	$A * (B * C) = (A * B) * C$
Distributiva:	$A * (B + C) = (A * B) + (A * C)$	$A + (B * C) = (A + B) * (A + C)$

Leggi di De Morgan: $\overline{(A + B)} = \overline{A} * \overline{B}$ $\overline{(A * B)} = \overline{A} + \overline{B}$

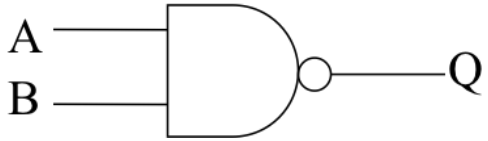
utili per trasformare AND in OR e viceversa.

Manipolazioni algebriche

Assorbimento I	$A + A \cdot B = A$	$A \cdot (A + B) = A$
Assorbimento II	$A + \overline{A} \cdot B = A + B$	$A \cdot (\overline{A} + B) = A \cdot B$
Assorbimento III	$A \cdot B + B \cdot C + \overline{A} \cdot C =$ $A \cdot B + \overline{A} \cdot C$	$(A + B) \cdot (B + C) \cdot (\overline{A} + C) =$ $(A + B) \cdot (\overline{A} + C)$

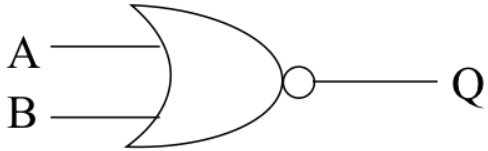
Operatori NAND e NOR

NAND: e' l'operatore NOT(AND)



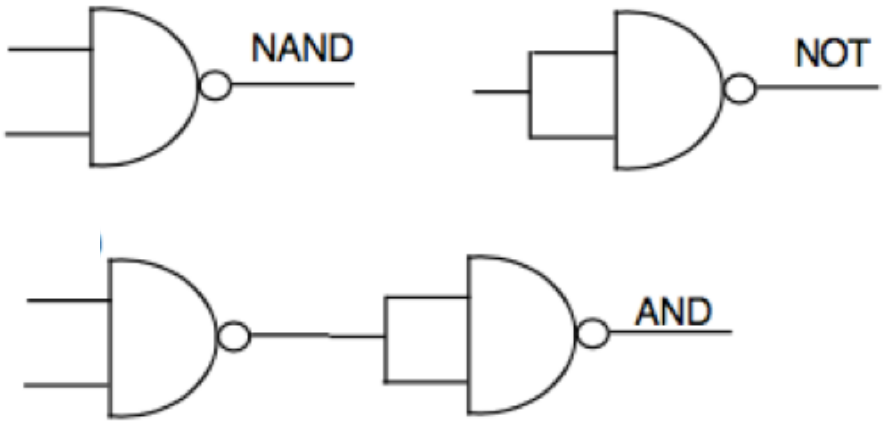
A	B	Q
0	0	1
0	1	1
1	0	1
1	1	0

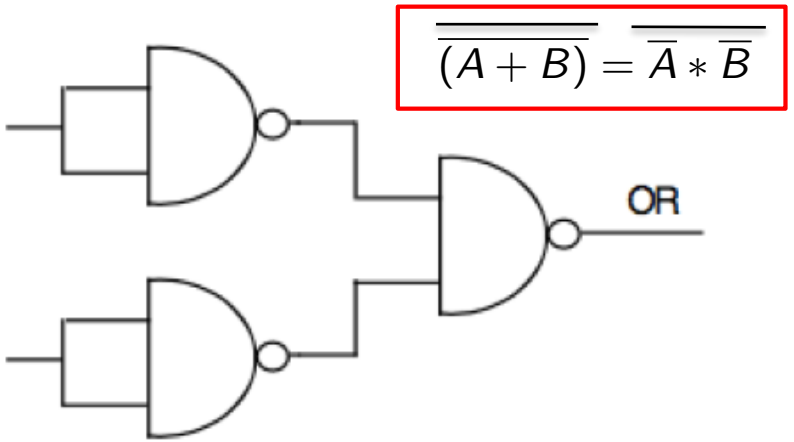
NOR: e' l'operatore NOT(OR)



A	B	Q
0	0	1
0	1	0
1	0	0
1	1	0

Si puo' dimostrare che il NAND(NOR) e' un'*operatore universale* i.e e' l'unico necessario per implementare qualsiasi funzione logica





Porte “universali”

A partire da alcune porte universali si possono costruire tutte le altre

Gruppi di porte universali	NOT	OR	AND
AND NOT	A — $\overline{A}$	A, B — $A+B$	A, B — $A \cdot B$
OR NOT	A — $\overline{A}$	A, B — $A+B$	A, B — $A \cdot B$
NAND	A — $\overline{A}$	A, B — $A+B$	A, B — $A \cdot B$
NOR	A — $\overline{A}$	A, B — $A+B$	A, B — $A \cdot B$

Forme canoniche e tabella della verità

Forme canoniche

- Ogni equazione logica puo' essere rappresentata in *forma canonica* tramite uso di operatori AND, OR, e NOT
- La forma canonica si deriva (ad es..) dalla tabella della verita' in forma di *Somma di Prodotti* (SP)

A	B	C	E
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

- Per ogni entry uguale ad 1 dell'output si genera un prodotto *minterm* degli input dove gli input=0 sono negati
- Per ottenere l'equazione in forma S, si sommano i prodotti cosi' ottenuti

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

- Grazie alla proprieta' di *identita'* della somma logica ($A + 0 = A$) il contributo all'equazione logica viene solo dai minterm non 0

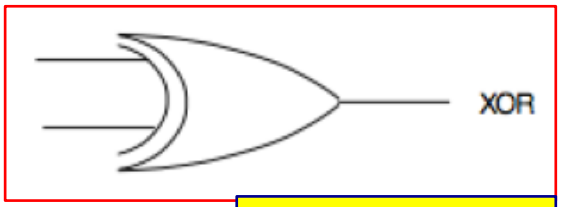
Dalla forma canonica al circuito

- A partire da un'equazione logica espressa come somma di prodotti (SP) si può realizzare il circuito equivalente con variabili (ovvero segnali) negate, oppure non negate, che attraversano:
  - un insieme di porte AND per i prodotti;
  - un insieme di porte OR per la somma dei prodotti.
- Vediamo ad esempio come possiamo realizzare un OR Esclusivo (XOR) di cui abbiamo la tabella della verità:

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

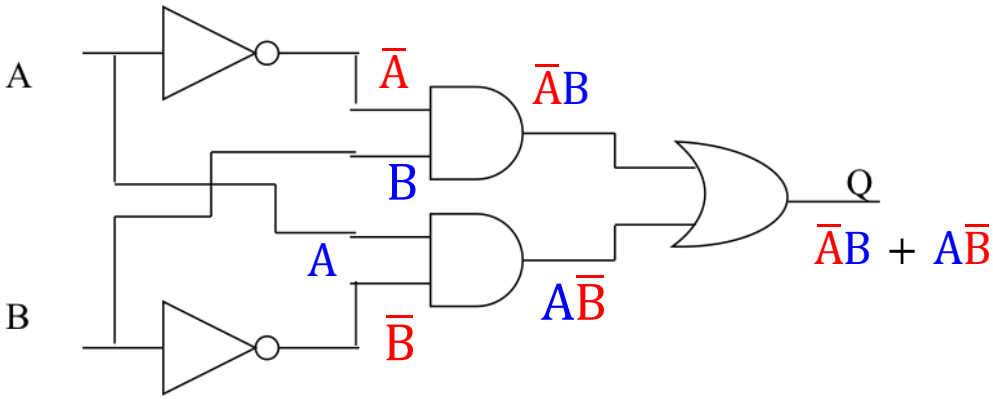
minterms

Diverso dall'OR



Simbolo circuitale

$$Q = (\overline{A} * B) + (A * \overline{B})$$

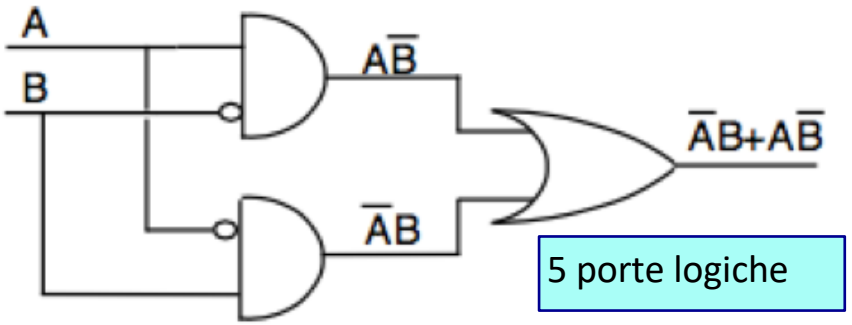


Abbiamo usato 5 porte logiche, potevamo fare meglio?

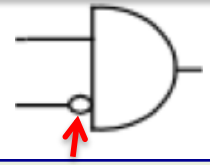
XOR e leggi di De Morgan

- Vediamo come l'espressione della funzione logica dello XOR può essere trasformata in altre formule grazie alle leggi di De Morgan:

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



5 porte logiche



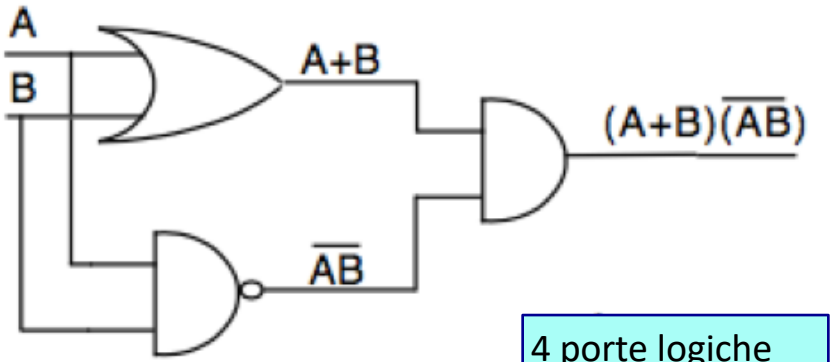
Abbreviazione del NOT (variabile negata)

N.B. $(A+B)*(\overline{A}+\overline{B}) = A\overline{A}+A\overline{B}+\overline{A}B+B\overline{B} = A\overline{B}+\overline{A}B$

Leggi di De Morgan

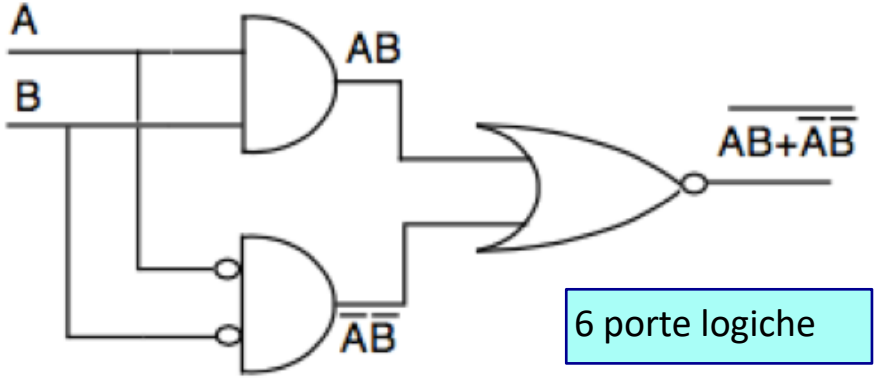
$$\overline{(A+B)} = \overline{A}*\overline{B} \qquad \overline{(A*B)} = \overline{A}+\overline{B}$$

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



4 porte logiche

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



6 porte logiche

Minimizzazione dei circuiti

Minimizzazione di circuiti

- Esempio: multiplexer a due input. Seleziona in uscita il valore di un ingresso scelto tra 2(N) diversi in base ai valori assunti dagli ingressi di *select* (S)

S	A	B	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

- Scrivo la funzione in termini di somme di prodotti

$Z = \overline{S}A\overline{B} + \overline{S}AB + S\overline{A}B + SAB$ –> distributiva

$Z = \overline{S}A(\overline{B} + B) + SB(\overline{A} + A)$ –> inverso

$Z = \overline{S}A + SB$

- 3 gate NOT, 8 gate AND e 3 gate OR –> 1 NOT, 2 AND, 1 OR !!!!!

Minimizzazione di circuiti

- Obiettivo della minimizzazione e' la riduzione del *costo* del circuito combinatorio in termini di numero di porte e variabili necessarie all'implementazione dell'equazione richiesta
- Si ottiene un circuito equivalente che generalmente e' piu' *piccolo* e con tempi di propagazione ridotti.
- Si puo' agire per ispezione utilizzando le proprieta' dell'algebra.

Esempio:

$Q = \overline{A} * B + A * B$ –>applico proprieta' distributiva
$Q = B * (\overline{A} + A)$ –> applico inverso
$Q = B * 1 = B$

- in questo caso la variabile A e' *DON'T CARE* cioe' non conta ai fini della definizione della equazione
- L'individuazione di variabili DON'T CARE e' l'obiettivo ultimo della minimizzazione

Esempio di tabella della verità

- Esempio: vogliamo costruire un sistema logico per accendere i lampioni in città. Si usa un interruttore manuale (A), oppure un timer (B) ed un sensore di luce (C).
- Vogliamo che siano accesi quando: 1) interruttore ON; oppure 2) timer ON e Buio. Abbiamo:

Variabile binaria	Stato del segnale	Livello logico associato
A	Interrutore OFF interruttore ON	0 1
B	Timer OFF Timer ON	0 1
C	Condizione di luce Condizione di buio	1 0

Variabili in ingresso			Variabile in uscita	Note
A	B	C	Q	
0	0	0	0	spenti
0	0	1	0	spenti
0	1	0	1	accesi
0	1	1	0	spenti
1	0	0	1	accesi
1	0	1	1	accesi
1	1	0	1	accesi
1	1	1	1	accesi

- Forma canonica dell'uscita:

$$Q = \bar{A}B\bar{C} + A\bar{B}\bar{C} + A\bar{B}C + AB\bar{C} + ABC$$

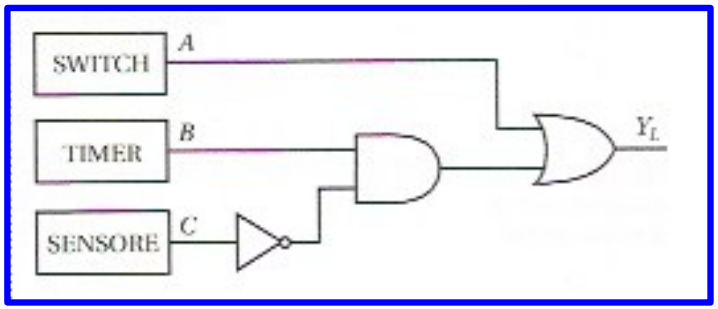
- Semplifichiamo l'espressione:

$$Q = \bar{A}B\bar{C} + A\bar{B}\left(\bar{C}+C\right) + AB\left(\bar{C}+C\right) = \bar{A}B\bar{C} + A\bar{B} + AB$$

$$Q = \bar{A}B\bar{C} + A\left(\bar{B}+B\right) = A + \bar{A}B\bar{C}$$

Assorbimento II	$A + \bar{A} \cdot B = A + B$

$$Q = A + B\bar{C}$$



Mappe di Karnaugh

Mappa di Karnaugh

- La mappa di Karnaugh è un metodo grafico che ha come obiettivo quello di ridurre la complessità della funzione logica espressa in forma canonica.
- È una rappresentazione visiva che consente di mettere in evidenza coppie di termini che differiscono per una sola variabile binaria.
- Prendiamo ad esempio la funzione seguente:

$$Y = A\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\,\overline{C}$$

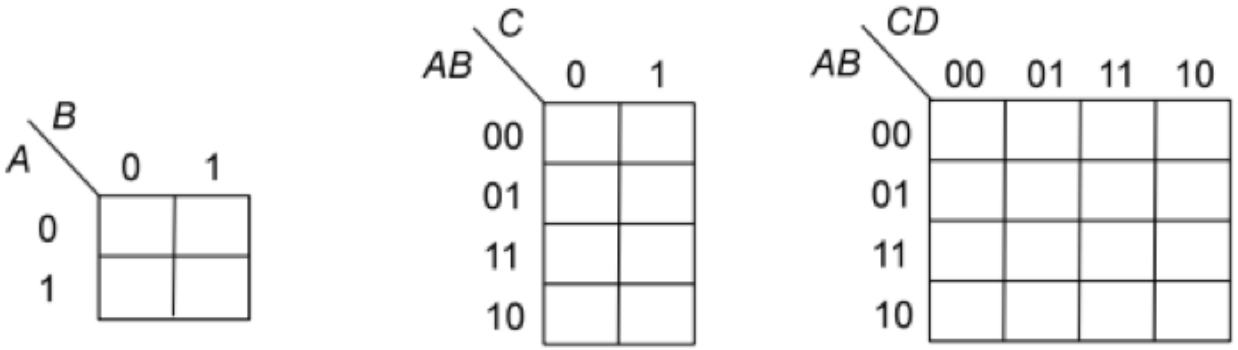
- Il primo e il terzo termine differiscono solo per il valore di C. Riordinando la funziona abbiamo:

$$\begin{aligned} Y &= A\overline{B}C + A\overline{B}\,\overline{C} + \overline{A}B\overline{C} \\ &= A\overline{B}(C + \overline{C}) + \overline{A}B\overline{C} \\ &= A\overline{B} + \overline{A}B\overline{C} \end{aligned}$$

- L'ultimo passaggio deriva dalla proprietà di complementazione, $C + \overline{C} = 1$.
- La mappa di Karnaugh ci consente di individuare in modo sistematico queste coppie di termini per semplificarle.

Mappe di Karnaugh

- La mappa è una tabella che contiene tante celle quante sono le righe della tavola della verità;
- Se abbiamo una funzione di 2 variabili (A e B) la tabella è costituita da 4 celle; ogni cella corrisponde ad un valore assunto dalle due variabili e il valore della funzione è riportato nella cella.
- Ad una funzione di 3 variabili corrisponde una tabella di 8 celle, accorpando A e B come in figura; ad una di 4 variabili corrisponde una tabella di 16 celle.
- Da un punto di vista grafico si possono trattare solo 4 variabili perché altrimenti avremmo bisogno di tabella tridimensionali, ma con dei programmi di calcolo si possono trattare anche più variabili.



2 variabili 3 variabili 4 variabili

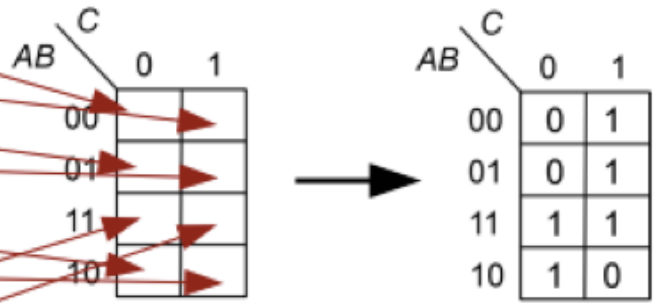
- Le tre (oppure 4) variabili devono essere ordinate come in figura.
- spostandosi da una casella all'altra (in verticale o in orizzontale) solo una variabile cambia valore.
- Da notare che la tabella va pensata come un cilindro, ovvero l'ultima colonna è adiacente alla prima, e analogamente l'ultima riga è adiacente alla prima.

Esempio di funzione di tre variabili

- Riportiamo tutti i valori della tavola della verità sulla mappa di Karnaugh:

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

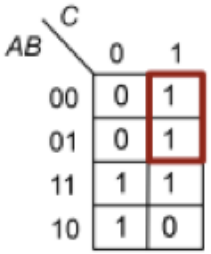
$$X=\overline{A}\,\overline{B}\,C+\overline{A}\,BC+A\,B\,\overline{C}+ABC+A\,\overline{B}\,\overline{C}$$



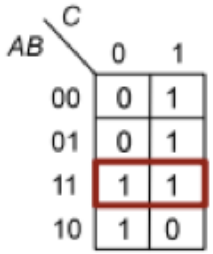
Dobbiamo individuare tutti gli 1 ``adiacenti'' che si possono così semplificare.

Vuol dire che una data variabile può assumere il valore 0 oppure 1 ma la funzione non cambia valore, rimane sempre 1

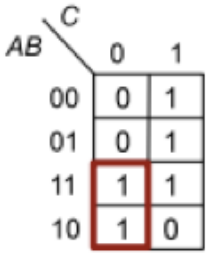
- Troviamo tutte le coppie di 1. Dobbiamo ``mappare'' tutti gli 1 della mappa di Karnaugh:



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



$$A\,B\,\overline{C}+ABC=AB(C+\overline{C})=AB$$



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

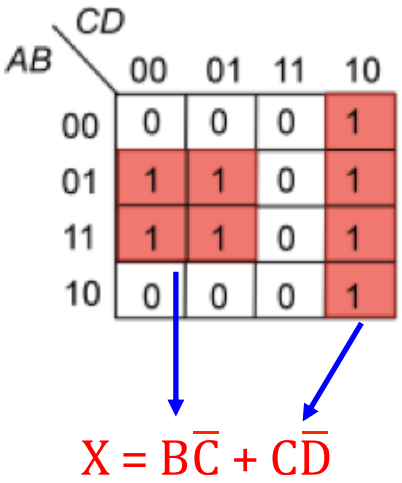
$$X=\overline{A}\,\overline{B}C+\overline{A}\,BC+A\,B\,\overline{C}+ABC+A\,\overline{B}\,\overline{C}=\overline{A}C+AB+A\,\overline{C}$$

Per ogni rettangolo individuato scriviamo un termine della funzione SENZA includere la variabile DON'T CARE

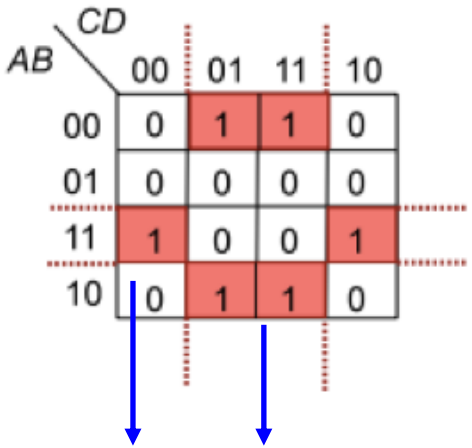
Procedimento generale

1. Individuare i rettangoli, più grandi possibili, che contengono solo ``1''. Il numero di celle racchiuse deve però essere uguale ad una potenza di 2, ovvero 1, 2, 4, 8 ….
   Non sono quindi leciti rettangoli con 3 celle oppure 6 celle.
2. La mappa va considerata come una superficie chiusa (toroidale), quindi il bordo superiore è contiguo a quello inferiore e il bordo destro è contiguo a quello sinistro.
3. Ogni cella può essere contenuta in più di un rettangolo, ma ogni rettangolo deve avere almeno una cella che appartiene solo ad esso. Ogni ``1'' deve essere coperto da almeno un rettangolo.

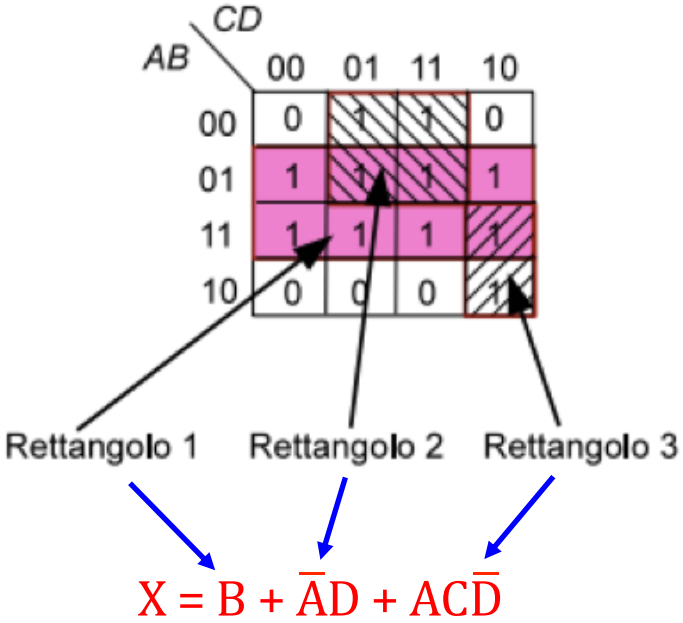
Esempio: tre funzioni diverse



$X = B\overline{C} + C\overline{D}$



$X = AB\overline{D} + \overline{B}D$



$X = B + \overline{A}D + AC\overline{D}$

Esercizio mappa di Karnaugh

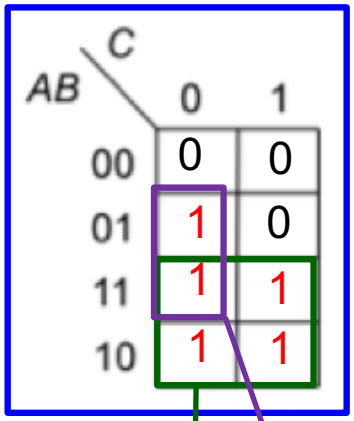
- Esempio: vogliamo costruire un sistema logico per accendere i lampioni in città. Si usa un interruttore manuale (A), oppure un timer (B) ed un sensore di luce (C).
- Vogliamo che siano accesi quando: 1) interruttore ON; oppure 2) timer ON e Buio. Abbiamo:

Variabile binaria	Stato del segnale	Livello logico associato
A	Interrutore OFF interruttore ON	0 1
B	Timer OFF Timer ON	0 1
C	Condizione di luce Condizione di buio	1 0

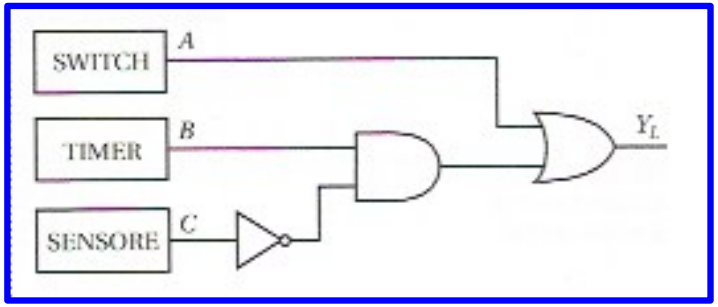
Variabili in ingresso			Variabile in uscita	Note
A	B	C	Q	
0	0	0	0	spenti
0	0	1	0	spenti
0	1	0	1	accesi
0	1	1	0	spenti
1	0	0	1	accesi
1	0	1	1	accesi
1	1	0	1	accesi
1	1	1	1	accesi

- Forma canonica dell'uscita:

$$Q = \bar{A}B\bar{C} + A\bar{B}\bar{C} + A\bar{B}C + AB\bar{C} + ABC$$



$$Q = A + B\bar{C}$$



Famiglie logiche

Famiglie logiche

- Una famiglia logica è definita da un insieme di regole e convenzioni, tali da rendere possibile la costruzione di circuiti complessi utilizzando circuiti logici elementari compatibili pienamente tra loro;
- Nel corso degli anni si sono sviluppate varie famiglie logiche e sul mercato esiste una vasta offerta di circuiti appartenenti ad ognuna di esse.
- Le più importanti famiglie logiche che sono state sviluppare nel tempo sono:

DTL	Diode-Transistor Logic
RTL	Resistor-Transistor Logic
TTL	Transistor-Transistor Logic
DCTL	Direct-Coupling Transistor Logic
ECL	Emitter-Coupled Logic
CMOS	Basata su transistor a effetto di campo

- Oggi la logica di gran lunga più utilizzata è la CMOS seguita dalla TTL, mentre la ECL è utilizzata solo per applicazioni particolari.
- In laboratorio utilizzeremo integrati della famiglia logica TTL

Famiglia TTL: porta NOT

- La famiglia logica TTL è la più nota e utilizzata; essa costituisce uno standard industriale.
- I livelli logici nominali (**in logica positiva**) sono:

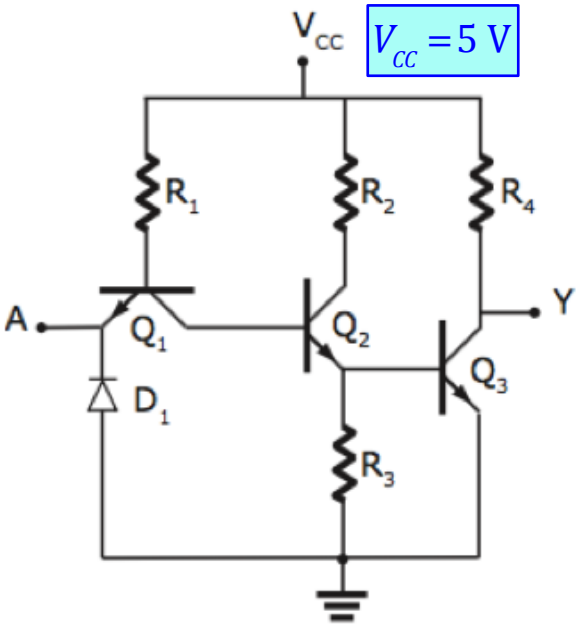
$$\begin{aligned} &0 \text{ logico} \quad \rightarrow 0.2\ V \\ &1 \text{ logico} \quad \rightarrow 5\ V \end{aligned}$$

Logica negativa vuol dire al livello 1 corrisponde il valore di tensione più basso e viceversa

Livelli logica ECL

$$\begin{aligned} &0 \text{ logico} \quad \rightarrow -1.4 \div -1.7\ V \\ &1 \text{ logico} \quad \rightarrow -0.7 \div -0.9\ V \end{aligned}$$

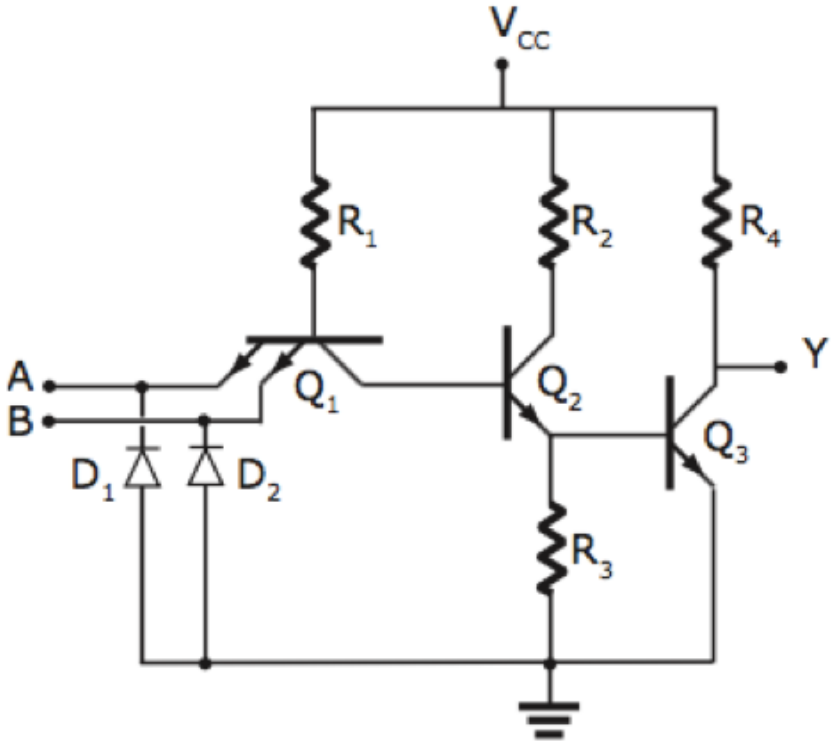
- Vediamo come si realizza una porta NOT:



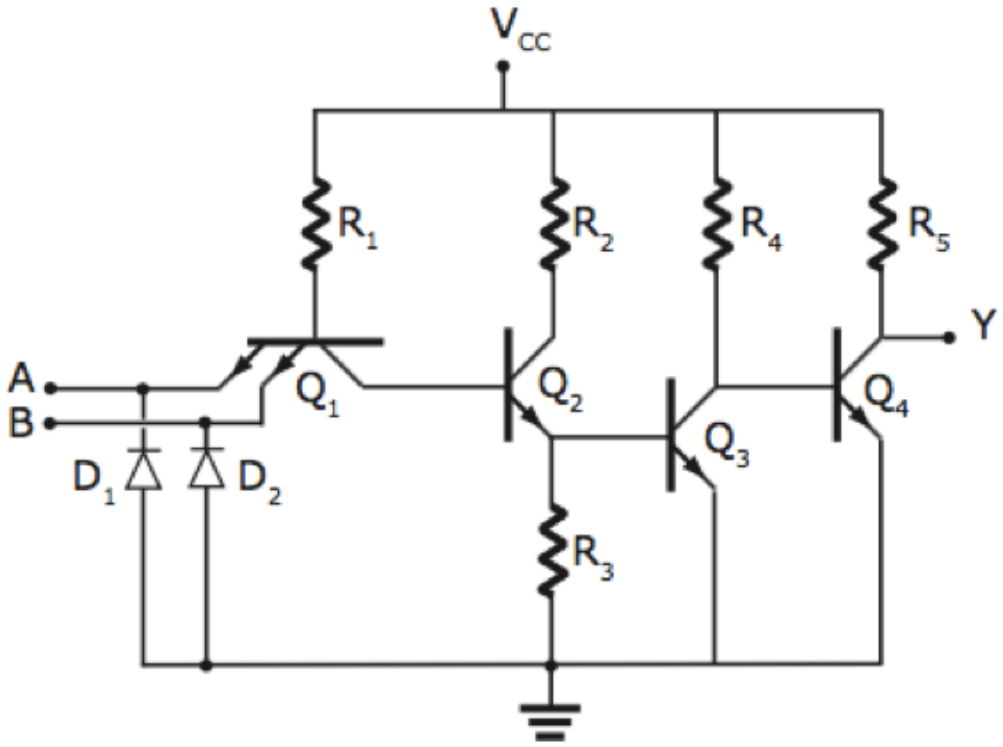
- Il transistor Q_1 è utilizzato come una coppia di diodi (BE e BC) uniti tra loro.
- Se in A c'è 1 (5 V) il diodo BE è interdetto mentre il diodo BC conduce;
- Vi sono in serie giunzioni: BC-Q_1, BE-Q_2 e BE-Q_3, tutte in conduzione
- Q_3 è in saturazione, quindi Y è a zero logico (0.2 V)
- Se in A c'è 0 (0 V) la giunzione BE di Q_1 conduce e la sua base è a circa 0.7 V.
- Q_2 e Q_3 sono in interdizione e l'uscita Y si porta a livello alto (5 V) (ipotizzando che siano in conduzione si arriva ad un assurdo).
- Il diodo D_1 non svolge nessuna funzione, serve a proteggere il transistor in caso di un ingresso negativo.

Famiglia TTL: porta NAND e AND

- La porta NAND si può costruire a partire dalla porta NOT utilizzando un transistor con doppio emettitore; aggiungendo un altro transistor in uscita si ha una porta AND:



NAND



AND

Integrato 74LS00

Datasheet NAND 74LS00

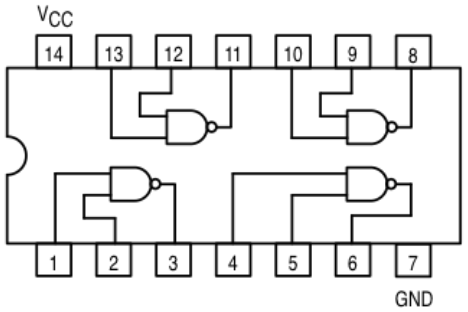
- Vediamo ad esempio l'integrato 74LS00 composto da 4 porte NAND:

MOTOROLA

QUAD 2-INPUT NAND GATE

- ESD > 3500 Volts

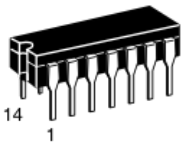
$V_{CC} = 5\ V$



SN54/74LS00

QUAD 2-INPUT NAND GATE

LOW POWER SCHOTTKY



J SUFFIX
CERAMIC
CASE 632-08

Altre porte della serie 74

Sigla	Funzione
7400	4 NAND a due ingressi
7402	4 NOR a due ingressi
7404	6 NOT
7408	4 AND a due ingressi
7410	3 NAND a 3 ingressi
7420	2 NAND a 4 ingressi
7430	1 NAND a 8 ingressi
7432	4 OR a 2 ingressi

Circuito sommatore

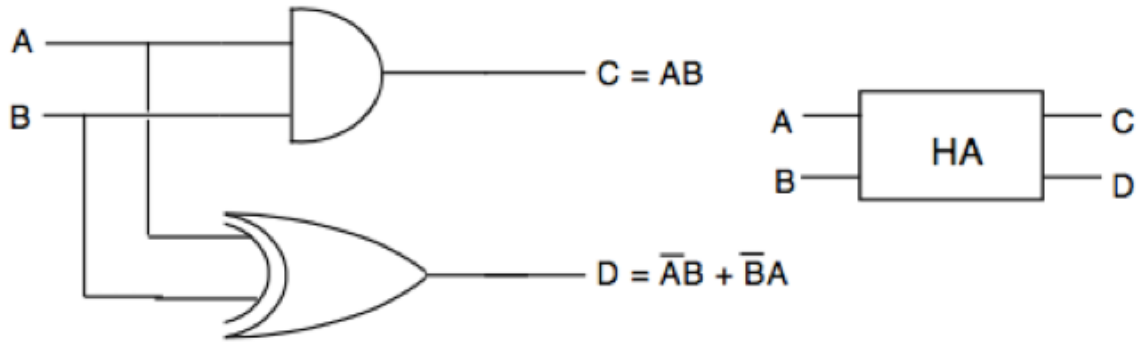
Circuito Sommatore: half-adder

- Costruiamo un circuito che effettua la somma di due numeri binari A e B a 1 bit (cioè ad una cifra). La tabella della verità è la seguente:

A	B	D	C	Somma
0	0	0	0	00
0	1	1	0	01
1	0	1	0	01
1	1	0	1	10

A e B sono i numeri da sommare
D è il risultato della somma (a 1 bit)
C è il bit di "riporto"
Somma sarebbe il risultato della somma a 2 bit

- Guardando la tabella si vede che: **D = XOR di A e B** ; **C = AND di A e B**
- Quindi si può utilizzare il seguente circuito, chiamato semi-sommatore (half-adder) che implementa la tabella riportata sopra:



- Questo modulo è il blocco fondamentale per realizzare un sommatore a più bit (per ogni bit, tranne il primo, occorrono due half-adder, da qui il nome)

Sommatore a n bit

- Costruiamo ora un sommatore di due numeri A e B a n bit, ad esempio a 4 bit come in figura:

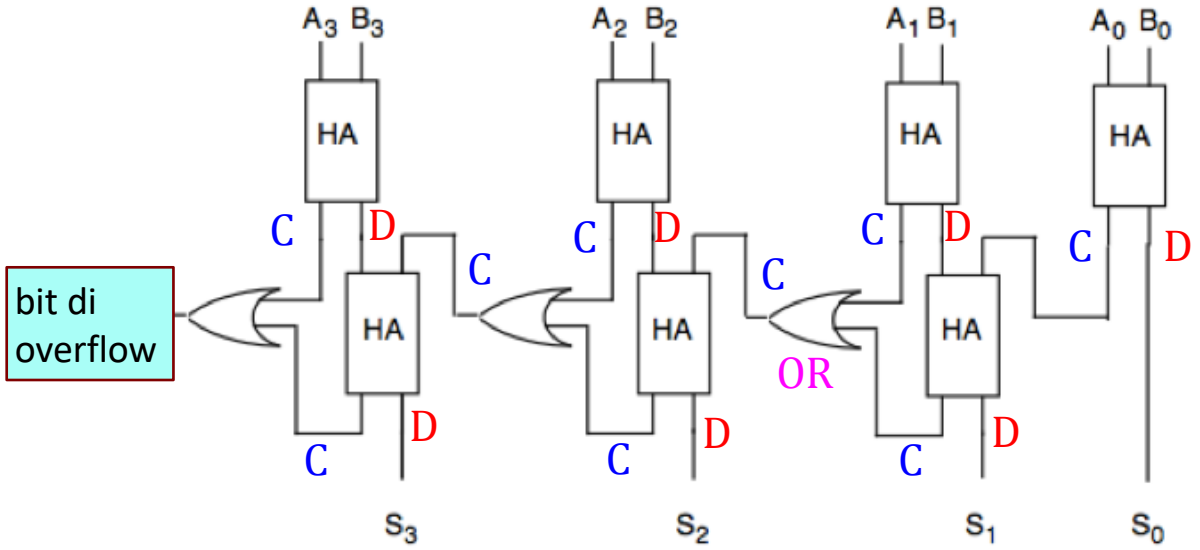
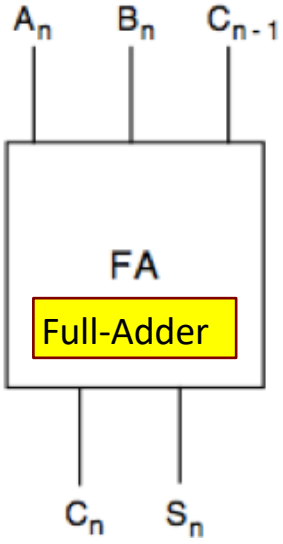


Tabella della verità per il bit n-esimo

A_n	B_n	C_{n-1}	S_n	C_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

- Lo stesso circuito può essere realizzato in altri modi. Partendo dalla tavola della verità del bit n-esimo, si possono scrivere le forme canoniche per S_n e C_n:

$$S_n = \overline{A}_n\overline{B}_nC_{n-1} + A_n\overline{B}_n\overline{C}_{n-1} + \overline{A}_nB_n\overline{C}_{n-1} + A_nB_nC_{n-1}$$
$$C_n = \overline{A}_nB_nC_{n-1} + A_n\overline{B}_nC_{n-1} + A_nB_n\overline{C}_{n-1} + A_nB_nC_{n-1}$$



- che semplificate diventano:

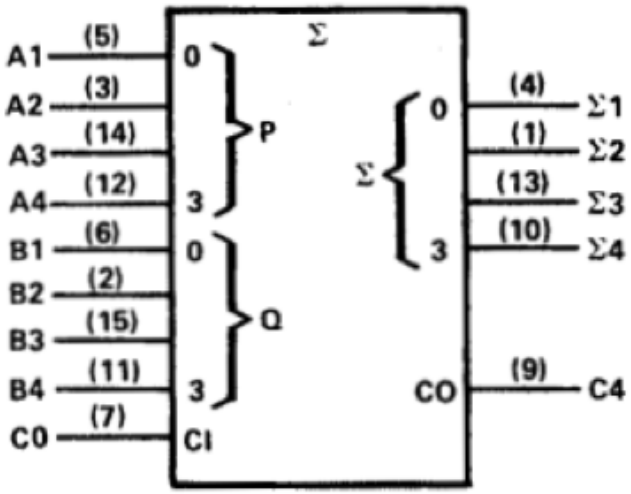
$$S_n = C_{n-1}\overline{(A_n \oplus B_n)} + \overline{C_{n-1}}(A_n \oplus B_n)$$
$$C_n = A_nB_n + A_nC_{n-1} + C_{n-1}B_n$$

Integrato 74LS283

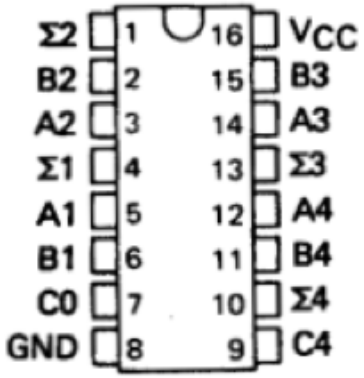
Integrato 74LS283: 4-BIT full-adder

- Integrato che fa la somma di due numeri a 4 bit.:

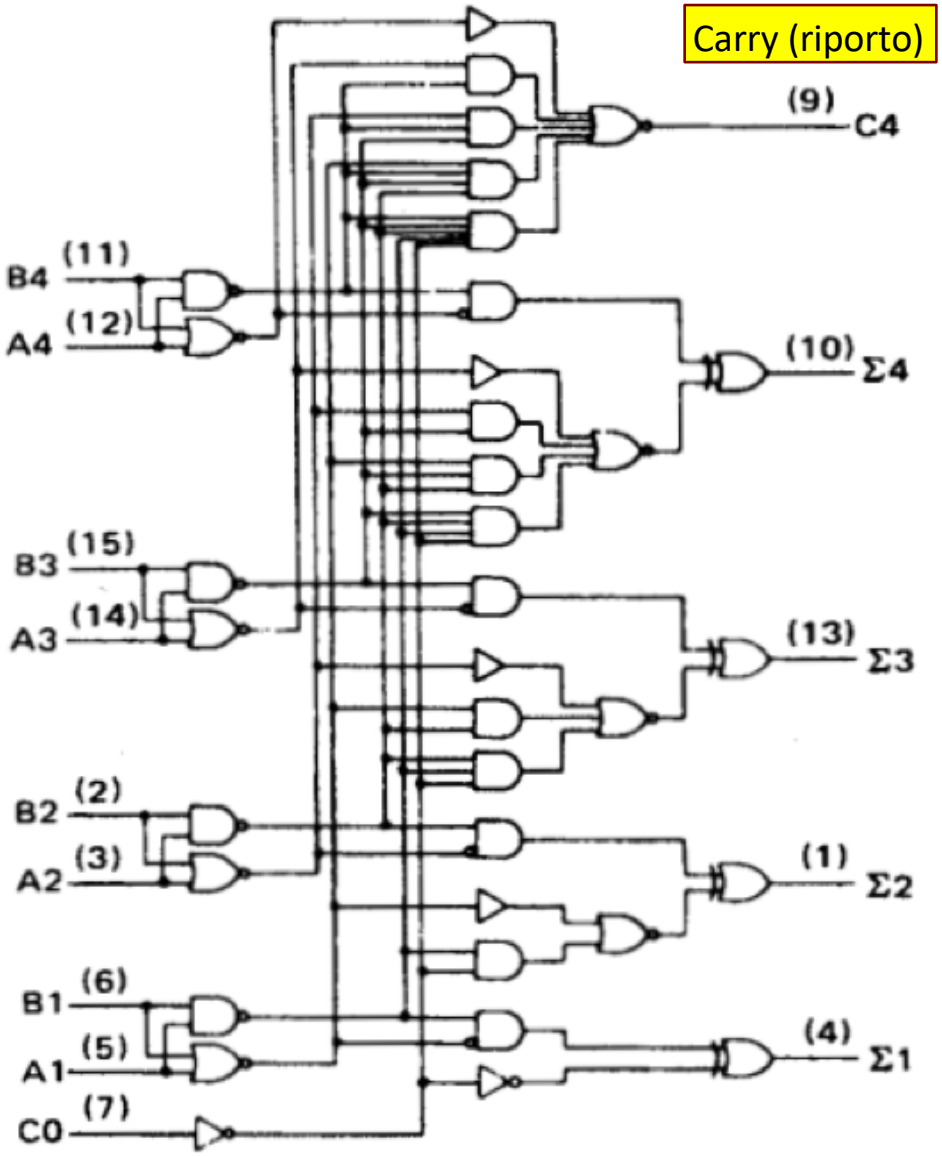
logic symbol†



(TOP VIEW)



logic diagram (positive logic)

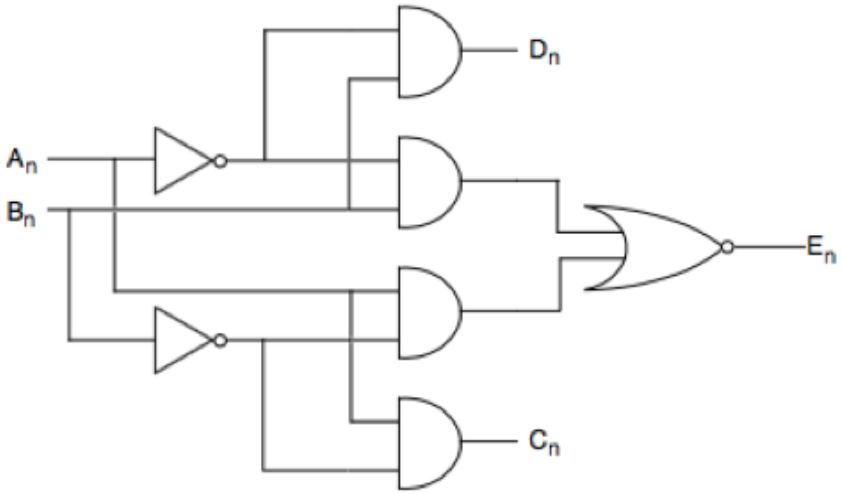


Comparatore digitale

Comparatore digitale

- Vogliamo costruire un dispostivo che confronti due numeri e stabilisca quale sia il maggiore.
- Iniziamo con il fare il confronto tra due numeri A e B ad 1 bit; possiamo costruire le tre funzioni logiche:

$$E_0 = \overline{A\overline{B} + \overline{A}B} \quad = 1 \quad \text{se } A = B$$
$$C_0 = A\overline{B} \quad = 1 \quad \text{se } A > B$$
$$D_0 = \overline{A}B \quad = 1 \quad \text{se } A < B$$



- Nel caso di numeri a n bit, ad esempio 4, abbiamo:

$$E = E_3E_2E_1E_0 \begin{cases} = 1 & A = B \\ = 0 & A \neq B \end{cases}$$

Due numeri sono uguali se e solo se tutti i bit sono uguali uno ad uno

$A_3 > B_3$
o $A_3 = B_3$ e $A_2 > B_2$
o $A_3 = B_3$ e $A_2 = B_2$ e $A_1 > B_1$
o $A_3 = B_3$ e $A_2 = B_2$ e $A_1 = B_1$ e $A_0 > B_0$

Per vedere se A è più grande di B confrontiamo prima il bit più significativo e poi a scalare tutti gli altri

$$C = A_3\overline{B_3} + E_3A_2\overline{B_2} + E_3E_2A_1\overline{B_1} + E_3E_2E_1A_0\overline{B_0}$$

$$C = \begin{cases} 1 & \text{se } A > B \\ 0 & \text{altrimenti} \end{cases}$$

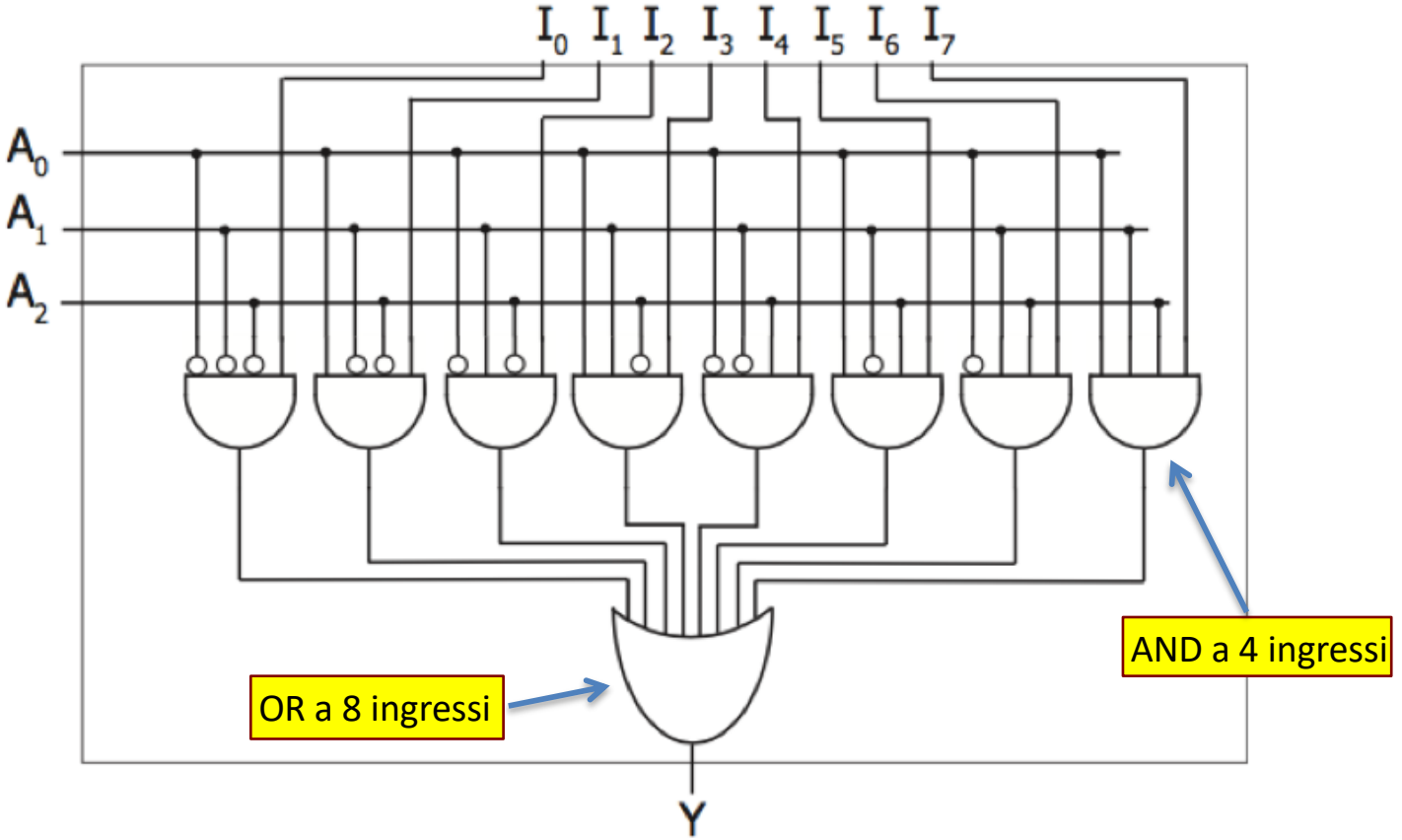
- La condizione A < B si implementa scambiando A con B

Multiplexer e demultiplexer

Multiplexer

- Un multiplexer è un circuito logico in grado di selezionare uno tra diversi ingressi logici e connetterlo all'uscita in base al valore presentato agli ingressi di selezione.
- Lo stato dei 3 ingressi di selezione **A** determina quale delle entrate **I** trasmette il suo livello logico all'uscita **Y**. Il numero binario impostato in **A** seleziona il bit corrispondente di **I**.

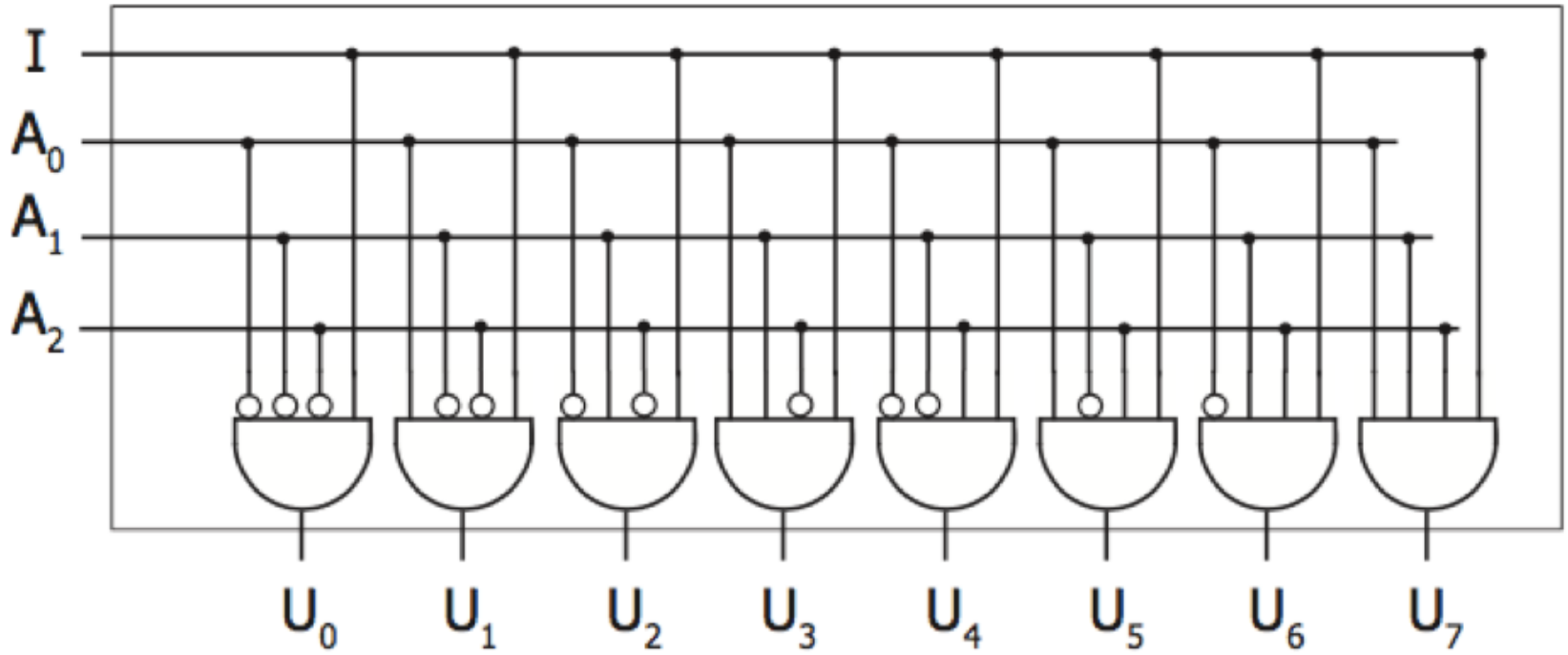
Variabili di selezione			Variabile in uscita
A_2	A_1	A_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7



- La porta AND trasmette il contenuto del bit **I** in uscita solo quando il contenuto degli altri tre ingressi (variabili di controllo) è uguale a 1

Demultiplexer

- Un demultiplexer svolge la funzione contraria: prende il contenuto di un ingresso e lo inoltra in una delle molte uscite sulla base del contenuto degli ingressi di selezione:

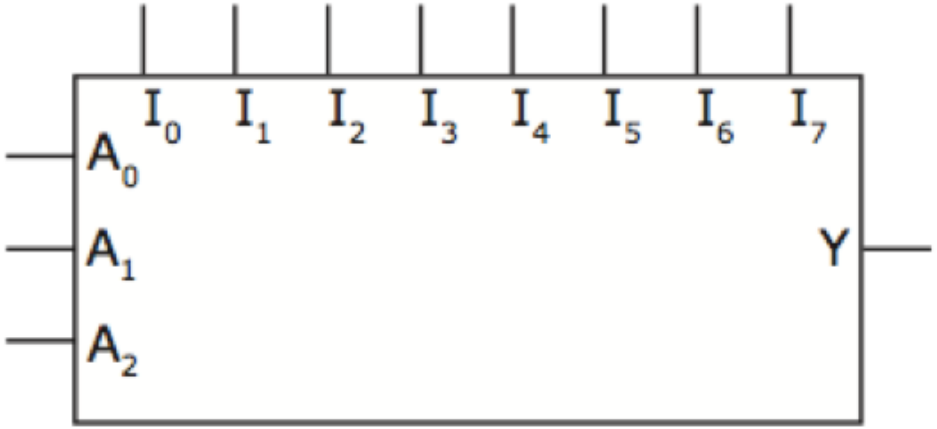


Multiplexer: generatore di funzioni

Esercizio 144

Un multiplexer è un dispositivo capace di selezionare un singolo ingresso fra tanti e trasmetterlo in uscita, in base al valore degli ingressi di selezione (vedi figura in cui e' mostrato un esempio a 8 ingressi). Lo stesso dispositivo puo' anche essere utilizzato per costruire facilmente qualunque funzione logica, con un numero di variabili pari al numero di ingressi di selezione.
Utilizzando un multiplexer a 8 ingressi costruire la funzione logica

$$Y = \overline{A}BC + A\overline{B}C + AB\overline{C}$$



Multiplexer: generatore di funzioni

La tavola della verita' del multiplexer e' la seguente:

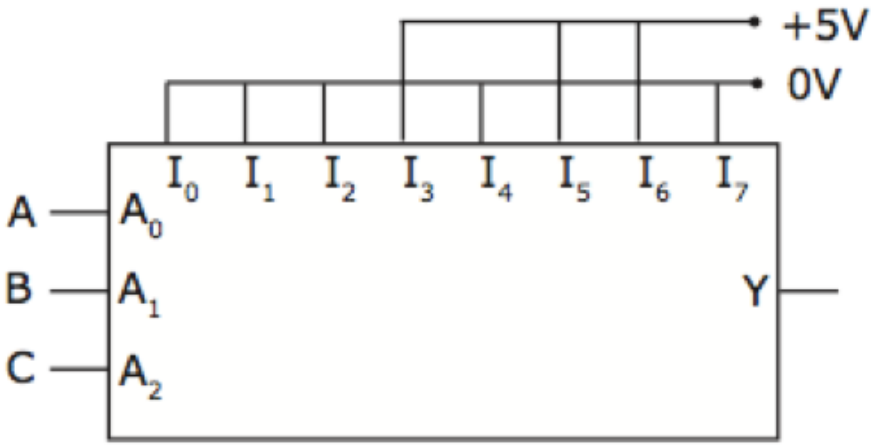
	C	B	A	
	A_2	A_1	A_0	Y
	0	0	0	I_0
	0	0	1	I_1
	0	1	0	I_2
$AB\overline{C}$ ⇨	0	1	1	I_3
	1	0	0	I_4
$A\overline{B}C$ ⇨	1	0	1	I_5
$\overline{A}BC$ ⇨	1	1	0	I_6
	1	1	1	I_7

$$Y = \overline{A}BC + A\overline{B}C + AB\overline{C}$$

Mettiamo 1 negli ingressi indicati dalla freccia (che corrispondono alle combinazioni della funzione logica) e 0 a tutti gli altri ingressi.
In questo modo in uscita avremo 1 solo quando gli ingressi di selezione selezioneranno proprio gli ingressi con gli 1, altrimenti avremo 0

Si ottiene quindi il risultato voluto applicando le variabili A,B,C agli ingressi di selezione e dando opportuni livelli logici agli ingressi I, come in figura.

N.B. +5 V corrisponde a 1 logico, mentre 0 V corrisponde a 0 logico



Encoder e decoder

Encoder e priority encoder

- Un encoder genera un'uscita binaria che fornisce l'indirizzo di quello tra gli ingressi che è stato attivato. In genere, se più ingressi sono attivati, viene fornito l'indirizzo dell'ingresso più alto (**priority encoder**).

- Vediamo la tabella della verità di un prority encoder (la x vuol dire qualunque stato logico):

I_3	I_2	I_1	I_0	A_1	A_0
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	x	0	1
0	1	x	x	1	0
1	x	x	x	1	1

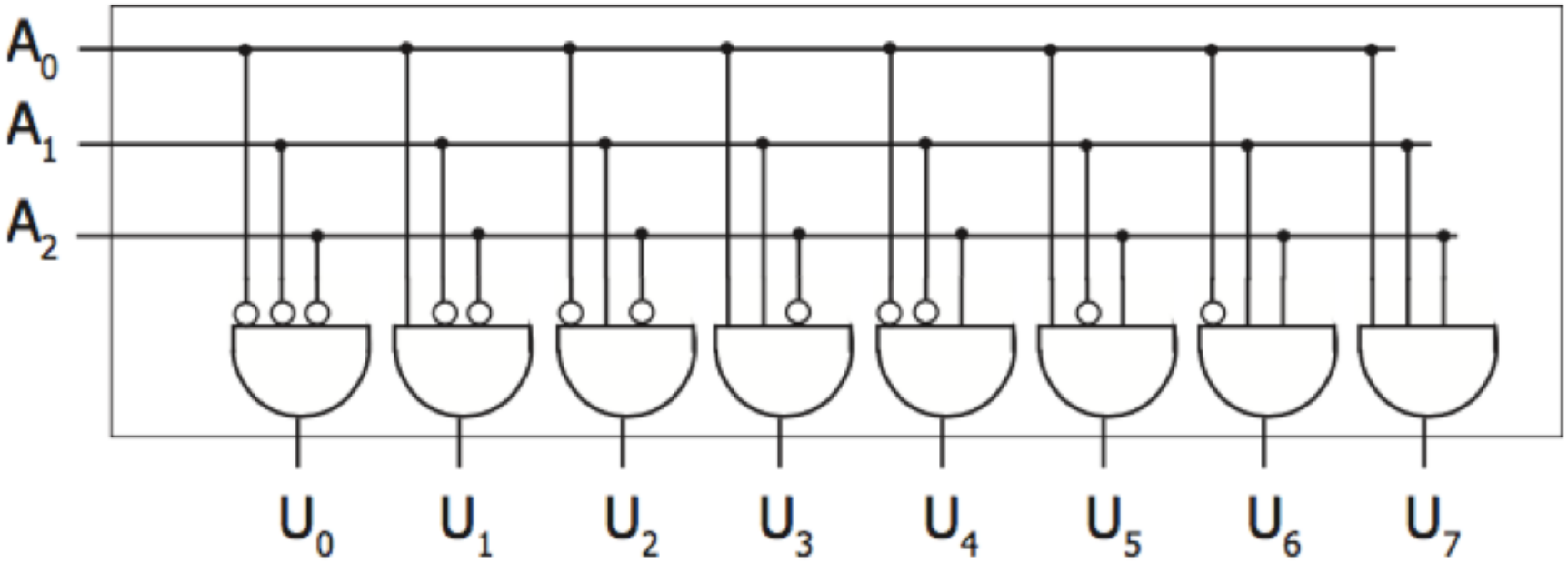
N.B. se abbiamo 4 ingressi occorrono due bit (A_1 e A_0) per scrivere l'indirizzo dell'ingresso più alto

- Se, ad esempio, prendiamo l'ultima riga, le uscite binarie A_1 e A_0 forniscono 1-1 se I_3=1 qualunque sia lo stato degli altri ingressi; in modo analogo si leggono le righe precedenti.

- Si noti che la configurazione di uscita 0-0 è ambigua: si può avere se nessuno degli ingressi è attivo oppure se è attivo l'ingresso I_0. Nei circuiti commerciali questa ambiguità è in genere risolta con un'uscita supplementare che indica se almeno uno degli ingressi è attivo.

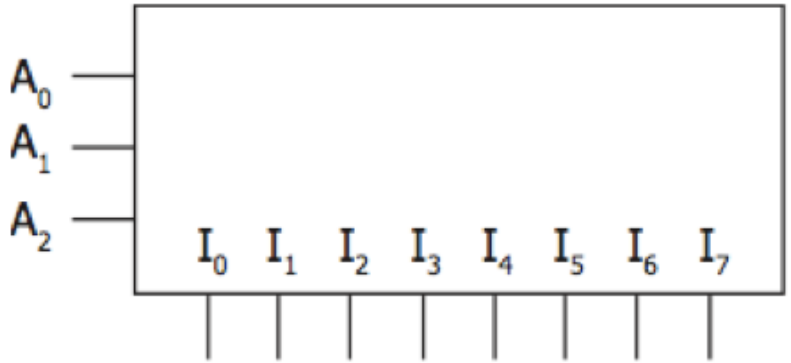
Decoder

- Il decoder è molto simile al demultiplexer, con la differenza che gli unici ingressi sono le linee di indirizzo, la cui decodifica provoca l'attivazione dell'uscita corrispondente.



Esercizio sul decoder (149)

Un decoder e' un circuito con n ingressi e 2^n uscite: lo stato logico degli ingressi determina quale delle uscite viene attivata, cioe' posta ad 1 logico. In Figura e' riportato un esempio di decoder a 3 ingressi e 8 uscite: I_0 e' attiva quando agli ingressi si ha $A_0 = 0$, $A_1 = 0$, $A_2 = 0$, I_1 e' attiva quando agli ingressi si ha $A_0 = 1$, $A_1 = 0$, $A_2 = 0$, e cosi' via.



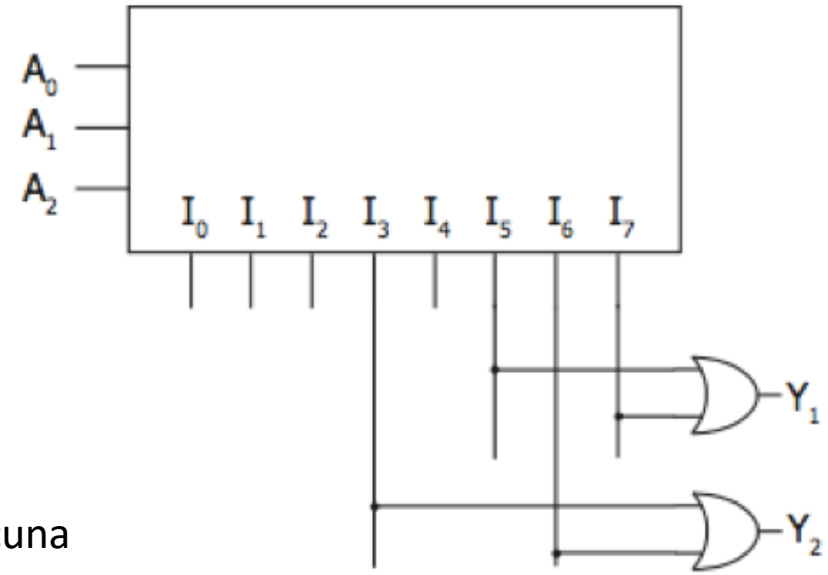
Un decoder, usato insieme a opportune porte logiche, consente anche di costruire simultaneamente e indipendentemente piu' funzione logiche degli ingressi. Utilizzando un decoder a 3 ingressi e quant'altro necessario costruire le 2 funzioni logiche:

$$Y_1 = A_0 A_1 A_2 + A_0 \overline{A_1} A_2$$
$$Y_2 = \overline{A_0} A_1 A_2 + A_0 A_1 \overline{A_2}$$

Soluzione esercizio sul decoder

Le uscite I_5 e I_7, messe in OR, forniscono la funzione Y_1.
Le uscite I_3 e I_6, messe in OR, forniscono la funzione Y_2.

$$\begin{aligned} Y_1 &= A_0A_1A_2 + A_0\overline{A_1}A_2 \\ Y_2 &= \overline{A_0}A_1A_2 + A_0A_1\overline{A_2} \end{aligned}$$



- Dobbiamo "decodificare" le uscite corrispondenti a ciascuna combinazione:

$$Y_1) \Rightarrow A_2A_1A_0 = 111 = 7 \quad + \quad A_2\overline{A}_1A_0 = 101 = 5$$

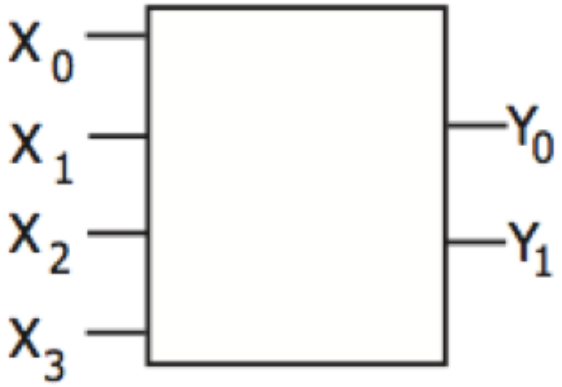
$$Y_2) \Rightarrow \overline{A}_2A_1A_0 = 011 = 3 \quad + \quad A_2A_1\overline{A}_0 = 110 = 6$$

- Se in ingresso è presente un altro indirizzo diverso da quelli indicati, l'uscita di questi ultimi rimarrà a zero, realizzando così la funzione voluta, che è uguale a 1 solo quando c'è il giusto indirizzo nei registri di ingresso.

Esercizio sul priority encoder

Esercizio 139

Il "priority encoder" è un circuito che presenta in uscita l'indirizzo più alto tra gli ingressi che sono attivi (cioè con 1 logico). Realizzare tale circuito (a 4 ingressi), utilizzando porte logiche elementari.



Soluzione esercizio 139

Esercizio 139
La tavola della verità è (x vuol dire qualunque):

X_3	X_2	X_1	X_0	Y_1	Y_0
1	x	x	x	1	1
0	1	x	x	1	0
0	0	1	x	0	1
0	0	0	x	0	0

In forma canonica si ha quindi:

$$Y_1 = X_3 + \overline{X}_3 X_2$$

$$Y_0 = X_3 + \overline{X}_3 \overline{X}_2 X_1$$

Esercizi circuiti combinatori

Esercizio di esonero 2017

Esercizio 3. (6 punti)

Si costruisca un circuito "rivelatore di minoranza" a 3 bit, cioè una logica che produca un segnale Q uguale a 1 quando la minoranza dei suoi bit d'ingresso è a 1 (lo stato 0-0-0 produce 0). Per semplicità si usino porte logiche a tre ingressi.

(Se è possibile disegnare il circuito nello spazio qui a fianco).

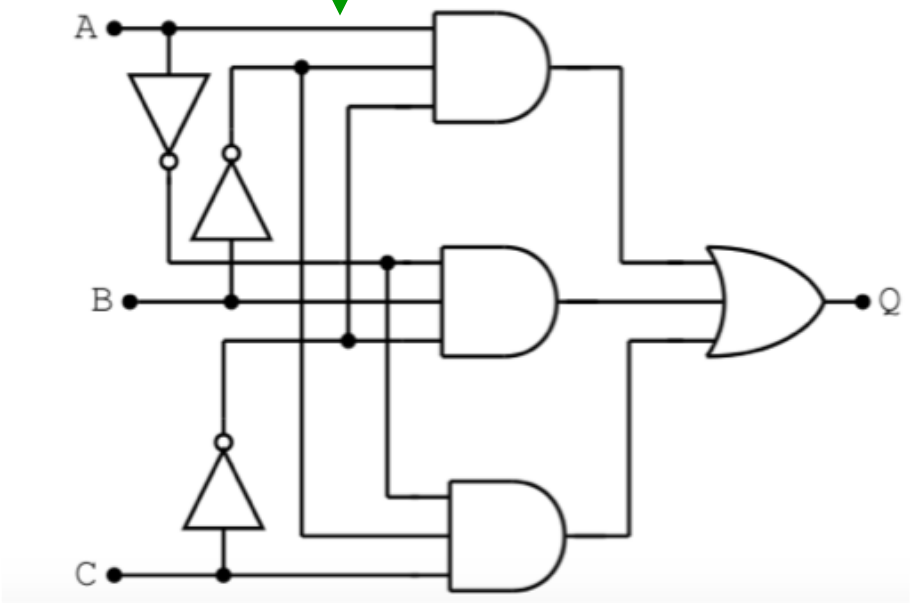
$Q =$ _____

Soluzione

Si costruisca un circuito "rivelatore di minoranza" a 3 bit, cioè una logica che produca un segnale Q uguale a 1 quando la minoranza dei suoi bit d'ingresso è a 1 (lo stato 0-0-0 produce 0). Per semplicità si usino porte logiche a tre ingressi.

A	B	C	Q
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

$$Q = \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}\bar{B}C$$

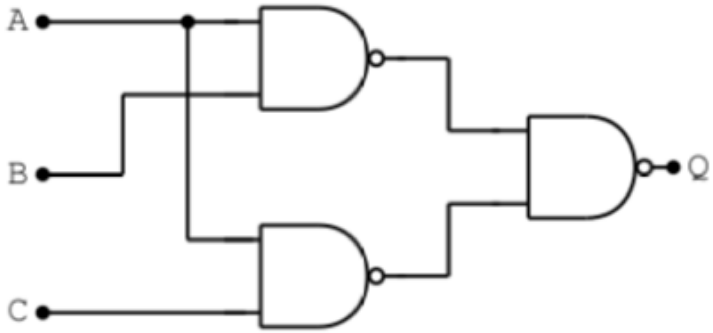


AB \ C	0	1
00	0	1
01	1	0
11	0	0
10	1	0

Non ci sono ``1'' adiacenti. La funzione non si può semplificare ulteriormente.

Esercizio di esonero 2017

Esercizio 5. (6 punti)
Si ricavi la tavola della verità corrispondente al circuito in figura. Si scriva poi la funzione in forma canonica e la si riduca ai minimi termini.

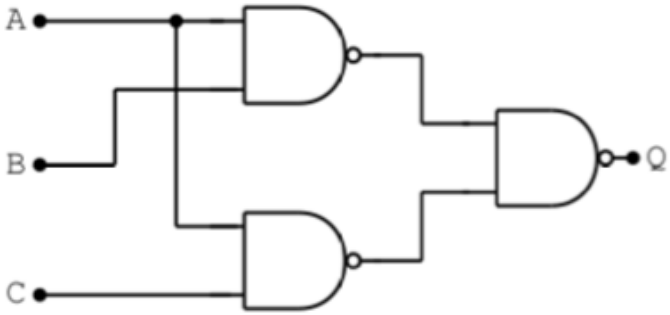


$Q =$ _____

Soluzione

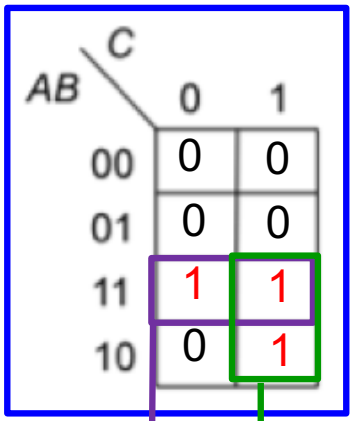
Esercizio 5. (6 punti)
Si ricavi la tavola della verità corrispondente al circuito in figura. Si scriva poi la funzione in forma canonica e la si riduca ai minimi termini.

$Q =$ _____



A	B	C	Q
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

$$Q = A\bar{B}C + AB\bar{C} + ABC = A\bar{B}C + AB = A(B + \bar{B}C) = A(B + C)$$



$Q = AB + AC$

SAPIENZA
UNIVERSITÀ DI ROMA

Fine del capitolo 6