Corso di Architettura degli Elaboratori - Mod. A
Anno Accademico 2001 / 2002

Capitolo 4 - Reti Combinatorie

Il passaggio obbligato per utilizzare in elettronica digitale il supporto teorico dell'algebra booleana è la cosiddetta "***codifica fisica***" dei due elementi dell'algebra booleana. Questa operazione si realizza associando a ciascuno dei due elementi un intervallo di valori di uno stesso parametro fisico. Si può ad esempio decidere che lo "0" corrisponde ad una pressione vicina a quella atmosferica ed un "1" ad una pressione di 10 atmosfere. Le logiche pneumatiche esistono e si usano, ma di gran lunga più usata è la "***codifica elettrica***".
I due intervalli di valori debbono essere disgiunti e facilmente distinguibili. La situazione ideale sarebbe quella di associare ai due stati logici i due soli intervalli di valori possibili, come accade, ad esempio nel caso dell'interruttore che, in linea di principio, può essere solo o "aperto" oppure "chiuso". Realizzare in pratica situazioni elettroniche di questo tipo è quasi impossibile, ma quello che si riesce a fare è largamente sufficiente.

La **codifica in termini di livello di tensione elettrica continua** è quella a cui si fa riferimento (almeno formalmente) nella progettazione dei circuiti elettronici per l'elaborazione dell'informazione (quei circuiti che in Informatica vengono genericamente indicati come, "***hardware***").
Questa codifica, con l'aiuto della **legge di Ohm** che, come è noto, lega la tensione elettrica agente in un circuito alla resistenza elettrica ed alla corrente che in essa fluisce, può essere fatta scaturire dalla **codifica in termini di interruttori**.
Lo stato "***chiuso***" di un interruttore, se esso è collegato ad una sorgente di tensione continua ed ad una resistenza, **produce passaggio di corrente** nel circuito e questo determina una differenza di tensione ai capi della resistenza. Lo stato "***aperto***", invece **non lascia passare corrente** e quindi ai capi della resistenza c'è differenza di tensione nulla.

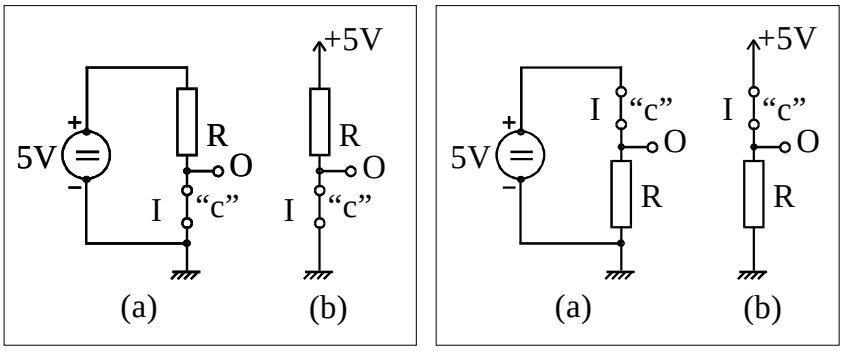
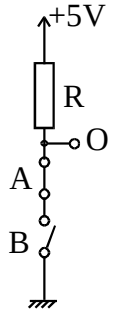


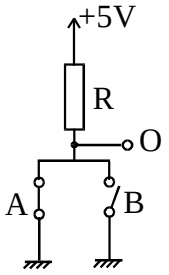
Fig. 1

Nei circuiti di figura, ciascuno dei quali è mostrato prima in forma scolastica e poi nella forma che adotta le **convenzioni tecniche normalmente usate**, quando l'interruttore **I** è chiuso, si ha nel punto contrassegnato con **O (Output)** nel primo caso **0 Volt** e nel secondo caso **+5 Volt**. Se, come spesso avviene, si stabilisce di associare a **0 Volt** (detto anche livello "***basso***" e si indica con "**L**" dall'inglese Low) il significato dell'elemento booleano "**0**" ed alla tensione di **+5 Volt** (livello "***alto***", "**H**" da High) il significato dell'elemento booleano "**1**", si avrà in uscita dal primo circuito, ad interruttore chiuso, un livello "0", mentre nel secondo circuito, con la chiusura dell'interruttore, si avrà in uscita un livello "1".

La configurazione usata per molti anni, prima che l'avvento dei più moderni circuiti integrati eliminasse del tutto la necessità di una resistenza nel circuito, è stata sempre **quella con l'interruttore con un capo a massa e la resistenza verso l'alimentazione**, perché questo tipo di connessione è quello più vantaggioso con gli interruttori di tipo elettronico.
La codifica elettrica considerata come esempio (**+5V / 0V**) è quella più utilizzata, da quando sono in uso le cosiddette **logiche elettroniche a basso livello**.
In precedenza le differenze di tensione tra i due livelli erano più elevate (10, 12, anche 15 Volt) per diminuire la probabilità di confondere uno 0 con un 1, perché i circuiti elettronici in uso avevano livelli di "***rumore***" molto più elevati di quelli attuali.
Il ***rumore*** è tutto ciò che può disturbare la corretta interpretazione logica del livelli di tensione. Ne parleremo con maggior dettaglio nel capitolo 2 quando parleremo della trasmissione a distanza dell'informazione.
Quella che associa l'**1** al livello "***alto***" e lo **0** a quello "***basso***" si dice logica "***diretta***" o "***positiva***". Quando si fa l'assegnazione opposta (l'1 corrisponde ad un livello di tensione più basso dello 0) la logica si dice "***invertita***" o "***negativa***".
I circuiti mostrati possono servire per associare **l'azione meccanica di chiusura od apertura di un interruttore** con i due **elementi booleani codificati elettricamente**, possono quindi essere utili per immettere in un sistema i valori booleani delle variabili d'ingresso. Essi possono anche consentire di realizzare circuiti che si comportano secondo una tabella di verità booleana come mostrato in figura, con però in ingresso i due stati degli interruttori ***a*** per "***aperto***" e ***c*** per "***chiuso***"ed in uscita i due livelli di tensione della codifica elettrica.



A	B	O
a	a	1
a	c	1
c	a	1
c	c	0



A	B	O
a	a	1
a	c	0
c	a	0
c	c	0

Fig. 2

Queste strutture, quindi, non sono adatte per essere costituenti di una funzione logica formata da più livelli di funzioni booleane elementari. Per questo è **necessario disporre di funzioni logiche in grado di ricevere e propagare valori logici già codificati elettricamente**.
In pratica occorrono **interruttori la cui chiusura od apertura possa avvenire con un comando elettrico,** corrispondente ad uno dei due livelli di tensione continua della codifica elettrica, di solito quello "alto".
Dispositivi di questo tipo esistono da moltissimi anni e si chiamano relé (relais). Sono formati da una ***elettrocalamita*** (un filo di rame avvolto su un nucleo di ferro) che quando il suo avvolgimento di eccitazione è attraversato da corrente, attira col suo campo magnetico un pezzo di ferro fissato al contatto elettrico dell'interruttore. Se questo ha una molla, che lo teneva aperto, si chiude (Normally Open - NO). Se invece la molla lo teneva chiuso, il contatto si apre (Normally Closed - NC).
I relé hanno consentito, prima dell'avvento degli interruttori elettronici, di costruire sistemi logici di complessità molto modesta, perché **i relé sono intrinsecamente lenti, consumano tanta energia elettrica ed erano, all'epoca, molto ingombranti**.

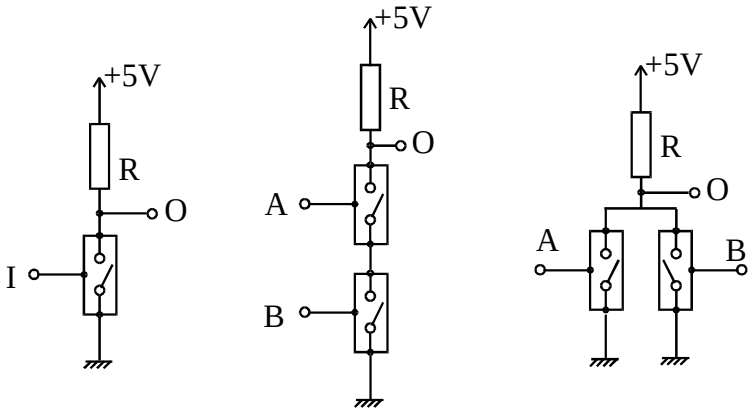
L'avvento dei primi ***interruttori elettronici*** (**tubi a vuoto**, detti anche valvole) ha aperto la strada ai primi elaboratori elettronici degni di questo nome. Il progresso è però diventato vorticoso solo con gli **interruttori elettronici** a "***stato solido***": Prima sono venuti i ***transistori bipolari*** e poi quelli ***unipolari***, entrambi al silicio. Questi ultimi sono più noti come ***MOSFET*** (sigla che significa Metal Oxide Semiconductor Field Effect Transistor).

Dare un'idea anche approssimativa della struttura degli interruttori elettronici esula certamente dagli scopi di questo corso e, pertanto, mi limiterò ad indicare con un **simbolo convenzionale** un generico **interruttore ad azionamento elettrico**, adatto quindi a chiudere od aprire un circuito ricevendo in ingresso un livello "alto" (+5 Volt) di tensione, che corrisponde ad un "1" in logica diretta ed ad uno "0" in logica invertita.
Negli schemi che seguono **gli interruttori saranno disegnati con i contatti aperti** e va inteso che essi **quando ricevono in ingresso il livello di +5 Volt si chiudono** perché questo è il modo di funzionamento dei transistori bipolari e di quelli MOS nei circuiti logici integrati.

Le più semplici combinazioni di questi interruttori a comando elettrico sono mostrate in figura 3.

A sinistra è disegnato uno solo di questi dispositivi con la **resistenza R** posta tra il dispositivo e l'alimentazione (R si chiama "carico").
In base alla descrizione che ne abbiamo dato, quando l'interruttore riceve sull'ingresso **I** un livello "***alto***" collega l'uscita **O** a massa (**0 Volt**). In questo modo **O** che era a **+5 Volt** si porta rapidamente a **0 V**, mentre la resistenza **R limita la corrente** che il generatore fa passare nell'interruttore chiuso.
Questo significa che in logica diretta, l'arrivo in ingresso di un "1" fa passare l'uscita O (che era ad "1") a "0". In sostanza **il circuito realizza le funzione NOT**. La tabella di verità sotto il circuito riassume in termini booleani il comportamento elettrico del circuito.

Nel circuito al centro **due dispositivi sono messi in serie**, cosa che comporta che solo quando **entrambi** sono "***chiusi***" (ingresso ad "1") l'uscita **O** sarà a **0 V**, cioè a "0".



I	O
0	1
1	0

A	B	O	$\overline{O}$
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

A	B	O	$\overline{O}$
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	1

Fig. 3

La tabella di verità riportata sotto il circuito mostra che il circuito si comporta da **AND con una inversione dello stato logico finale**, quindi si può sinteticamente dire che esso realizza una funzione AND negata che si chiama **NAND**.

L'ultimo circuito mostra **due dispositivi in parallelo** e ciò fa si che **appena uno dei due è** "***chiuso***" l'uscita **O si porta a 0 V** e lo stato dell'altro diventa irrilevante. La tabella che descrive il comportamento del circuito è riportata sotto e dimostra che si tratta di una funzione **NOR**.
La rappresentazione simbolica delle funzioni è mostrata in figura 4.

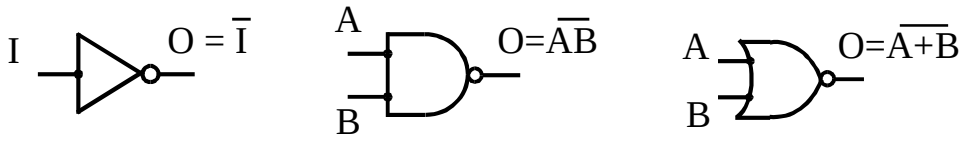
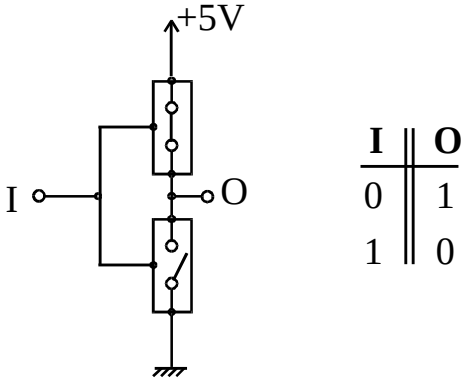


Fig. 4

E' molto importante ricordare che **non è assolutamente necessario disporre sia della configurazione serie dei dispositivi "interruttore" che di quella parallela**, per realizzare una qualunque funzione logica complessa, visto che **la funzione NOT ed i teoremi di De Morgan ci consentono di costruire un OR con un AND ed un AND con un OR.**
Questa considerazione è rilevante perché i vari interruttori elettronici che si sono succeduti nelle realizzazioni hardware in questi anni, in funzione delle loro caratteristiche specifiche, hanno volta per volta suggerito ai progettisti di adottare come configurazione base dei dispositivi una sola delle due teoricamente possibili senza che ciò creasse problemi.

Gli "***interruttori***" elettronici più moderni e veloci sono i dispositivi **CMOS**. La sigla significa **Complementary MOS** e deriva dal fatto che l'elemento base della logica, la funzione NOT, detta anche "***inverter CMOS***" invece di essere costituito da un interruttore "***normalmente aperto***" e da una resistenza è costituito da **due interruttori in serie,** uno "***normalmente aperto***" ed uno "***normalmente chiuso***", che, quando ricevono in ingresso un livello logico "1" **invertono il loro stato**. Il risultato è che, come nell'invertitore con la resistenza di carico, lo stato logico di uscita passa da "1" a "0" ma poiché l'interruttore, che teneva la tensione d'uscita a +5 V, si apre contemporaneamente alla chiusura dell'interruttore verso massa, **non c'è passaggio di corrente apprezzabile nel circuito** e questo costituisce un **vantaggio enorme per l'assorbimento di energia delle funzioni logiche realizzate con questa tecnologia**.
Questo non è il solo vantaggio dei circuiti CMOS ma è certamente il principale.



I	**O**
0	1
1	0

Fig. 5

In figura 5 è mostrato lo **schema di principio dell'invertitore CMOS** in cui l'interruttore elettronico MOS "*complementare*", quello che ricevendo un livello "1" si apre, è disegnato come un interruttore elettromeccanico "*normalmente chiuso*".
Questo tipo di configurazione elettrica ha tali vantaggi dal punto di vista elettronico da essere replicata nel circuito d'uscita di tutte le funzioni logiche CMOS anche se non contengono una inversione logica.

Nello schema tecnico di figura 6 è rappresentata la struttura di una **funzione AND a due variabili ottenuta aggiungendo un invertitore finale ai due "interruttori" MOS in configurazione serie**.
La caratteristica peculiare della versione CMOS della funzione è costituita dai due **MOS complementari** (riconoscibili dalla freccetta con la punta verso destra) in parallelo tra di loro, tra l'uscita e l'alimentazione, al posto della resistenza.
Solo quando entrambi gli ingressi sono "alti" e, quindi, entrambi gli interruttori MOS tra l'uscita e massa sono "chiusi", si "apre" il collegamento tra alimentazione ed uscita.

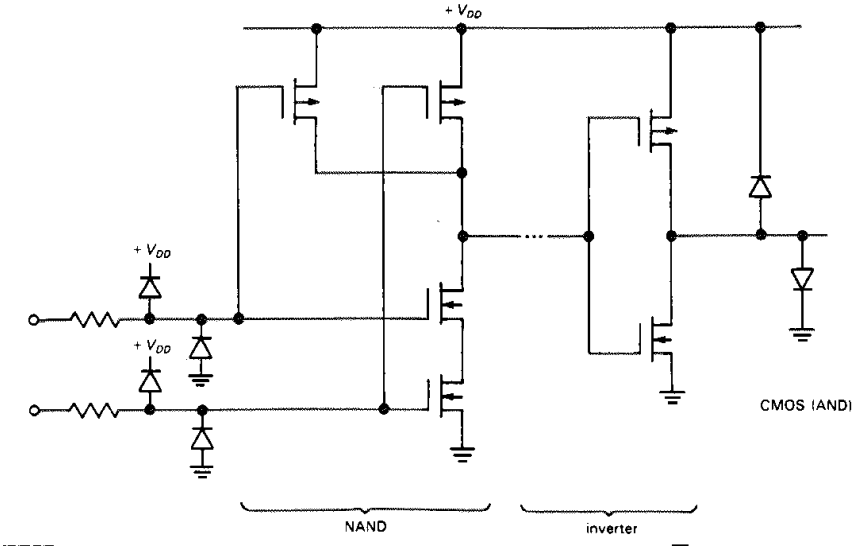


Fig. 6

L'associazione di un interruttore "normalmente aperto" e di uno "normalmente chiuso", è una caratteristica comune ai circuiti di uscita di tutte le funzioni logiche integrate, indipendentemente dalla tecnologia di costruzione, perché rappresenta la soluzione ottimale per ridurre il tempo che il circuito impiega a passare dal livello "basso" a quello "alto" o viceversa.

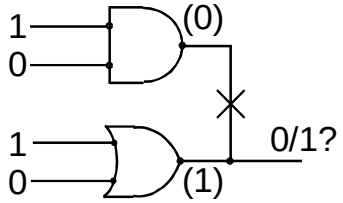


Fig. 7

Questa caratteristica elettrica delle funzioni logiche integrate, comporta che: "**la connessione tra loro delle uscite di due funzioni logiche normali è assolutamente da evitare perché se una delle due cercasse di portare l'uscita comune al livello logico opposto a quello dell'altra, il conflitto**

elettrico conseguente a quello logico danneggerebbe i circuiti d'uscita delle due funzioni" (Figura 7).

Vedremo più avanti in questo capitolo che, quando la connessione diretta delle uscite di funzioni logiche è indispensabile si adottano dei circuiti speciali.

Funzioni logiche a molti ingressi ed a molti livelli di funzioni.

Abbiamo visto, finora, solo schemi che realizzano **funzioni elementari booleane con due sole variabili** ma è chiaro che è possibile costruire **circuiti che realizzano funzioni logiche a 3, a 4 e perfino ad 8 ingressi** e, nel caso fosse necessario realizzare funzioni **AND od OR a più di 8 variabili, queste possono sempre essere spezzate in due o più funzioni, le cui uscite confluiscono in una funzione dello stesso tipo**. La disponibilità dell'hardware necessario consente di realizzare con facilità funzioni logiche complesse, assegnate con tabelle di verità, ed, eventualmente, minimizzate attraverso i procedimenti visti.
Nella figura 8 è, ad esempio, disegnato lo schema di principio della funzione che è stata minimizzata a suo tempo con la tecnica delle mappe di Karnaugh:

$$f = x_3 \xi_1 x_0 + x_3 x_2 + x_2 x_1 x_0 + x_3 \xi_2 \xi_0$$

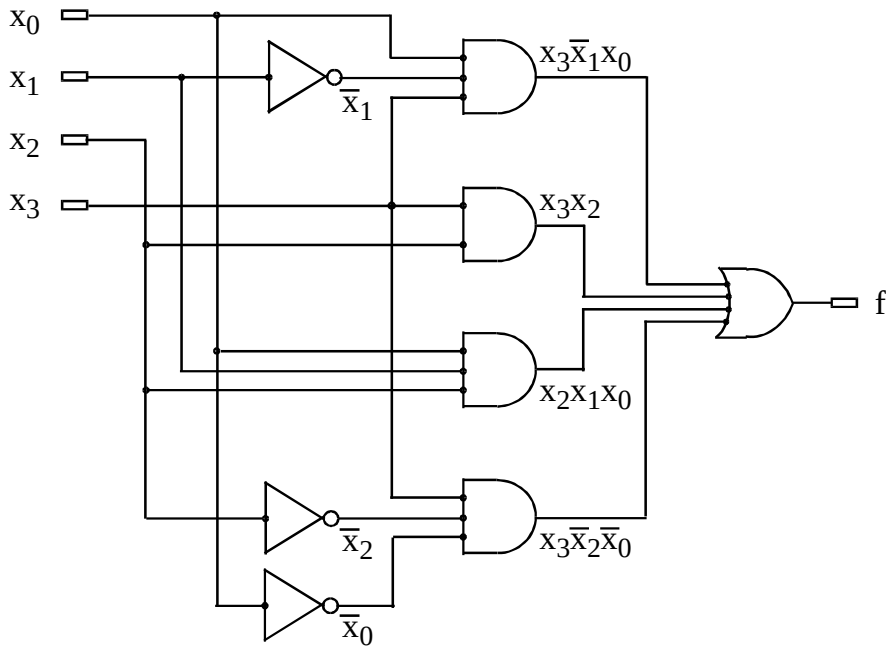


Fig. 8

Il cerchietto della negazione può anche immaginarsi incorporato nel corrispondente ingresso della funzione elementare.

Definizione di "*rete combinatoria*"

Un insieme di AND od OR (entrambi si dicono genericamente "porte logiche" o "gate" in inglese) **in cui ci sono delle linee di ingresso, delle linee di uscita e linee che collegano le uscite di porte logiche agli ingressi di altre porte od invertitori** si dice "***rete combinatoria***".

Si badi che la funzione della figura precedente ha una struttura SOP ma ciò non è assolutamente vincolante per una rete combinatoria in cui i gate AND ed OR possono succedersi in un ordine qualsiasi.
Si tenga ben presente che l'espressione "**porta logica**" o "**gate**", in inglese, fa esplicito riferimento alla presenza di più ingressi, ciascuno dei quali condiziona attraverso il suo stato logico il manifestarsi in uscita di una transizione logica su un altro ingresso. La conseguenza ovvia di questa osservazione è che: **la funzione NOT non è una porta logica**.

Tempi di "*propagazione*".

Nell'algebra di Boole la **variabile tempo non è assolutamente prevista**; ma, quando dallo studio matematico di una funzione, **si passa a realizzare un circuito** che ricevendo in ingresso le configurazioni delle variabili (codificate elettricamente) riproduca in uscita, in forma elettrica, i valori booleani previsti dalla tabella di verità, il discorso cambia radicalmente.
L'Elettronica digitale, infatti, come qualunque scienza sperimentale, deve porsi il problema del "***tempo di risposta o di propagazione***", inteso come il **tempo necessario perché la variazione dello stato di una variabile determini lo stato d'uscita corrispondente della funzione**.
Poiché **ogni tipo di "*interruttore*", elettromeccanico od elettronico, ha un suo tempo finito di chiusura od apertura,** ogni funzione booleana, a seconda della tecnologia di realizzazione, ha un "***tempo di risposta tipico***". Questo significa che il particolare esemplare di funzione che stiamo usando potrebbe averne uno maggiore o minore. Il tempo "***tipico***" è solo il più probabile: Spesso il costruttore del circuito integrato fornisce anche il tempo di propagazione massimo.
Queste considerazioni fanno apparire alcuni discorsi fatti in precedenza sotto una luce diversa. Quando, ad esempio si è detto **che una funzione AND di molte variabili può essere realizzata facendo prima l'AND delle variabili primarie a gruppi e poi l'AND delle uscite,** bisogna considerare che **il tempo di "*propagazione*" tra uno degli ingressi e l'uscita finale della funzione sarà determinato dalla somma dei ritardi delle due funzioni AND in cascata**.

L'esistenza di un tempo di propagazione finito (e **non sempre esattamente prevedibile**) può costituire, in reti combinatorie di una certa complessità, un problema di non facile soluzione. In particolare quando la rete logica ha molti ingressi e più di una uscita, può succedere che **il tempo di propagazione tra due diversi ingressi ed una uscita o quello tra un ingresso e due diverse uscite, siano notevolmente diversi tra loro** perché è diverso il numero di livelli di funzioni booleane da attraversare.
In questi casi **l'utilizzazione delle uscite della rete combinatoria deve avvenire solo quando si ha la certezza che tutte le uscite sono aggiornate ai valori corrispondenti alla configurazione corrente delle variabili d'ingresso**.

Esempi di funzioni combinatorie che si incontrano spesso: Codificatori e decodificatori.

La codifica binaria di un elemento d'informazione può, nella sua accezione generale, meritare qualche considerazione specifica che verrà sviluppata più avanti, ma nella sua forma elementare la codifica è una operazione molto semplice. Il numero sulla maglia di un calciatore è un codice numerico, ovviamente decimale. L'unico vincolo in una operazione di codifica è rappresentato dal numero massimo di elementi codificabili, per il fatto che non ci possono essere combinazioni numeriche usate più di una volta. Con due cifre decimali, le combinazioni distinte sono 100 (da 0 a 99). In binario con 4 cifre si può andare da 0000 a 1111 (da 0 a 15).

Per chiarire con un esempio a che cosa può servire un codificatore, partiamo da un caso che si presenta molto frequentemente nella progettazione di semplici sistemi hardware. Supponiamo di voler costruire un circuito logico che debba effettuare una di quattro possibili diverse operazioni la cui selezione è affidata al **valore di due bit: x_1 ed x_2 che, come è noto, possono dar luogo a 4 combinazioni distinte: 00, 01, 10, 11**. Se sul pannello frontale dell'apparecchio si dispongono **quattro pulsanti contrassegnati dalle lettere A, B, C e D**, **l'azionamento di uno dei pulsanti dovrà generare una delle quattro possibili combinazioni dei due bit che attraverso un meccanismo logico che vedremo subito dopo, determinerà l'esecuzione dell'operazione associata al pulsante**. E' ovvio che il circuito elettrico associato al pulsante deve, in corrispondenza dell'azionamento dello stesso, assegnare un "1" logico alla corrispondente variabile booleana che costituisce uno degli ingressi della funzione logica che vogliamo costruire.
Si tratta di una operazione di "***codifica***", ovvero di **associazione di un elemento, in questo caso il pulsante, ad un codice binario**. Volendo citare una operazione dello stesso tipo ma che sarebbe molto più laborioso sviluppare, si tratta di quello che succede quando **un operatore preme un tasto di una tastiera numerica od alfanumerica**.

Poiché sappiamo già ***costruire*** (si dice anche ***sintetizzare***) una funzione booleana complessa partendo da una tabella di verità, quando avremo tradotto il problema descritto discorsivamente in una tabella di verità, esso sarà sostanzialmente risolto.
La tabella di verità corrispondente all'esempio è la seguente:

A	B	C	D	X_2	X_1
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

Tab. 1

Le forme canoniche SOP delle due funzioni sono:

$$x_1 = \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot D + \overline{A} \cdot B \cdot \overline{C} \cdot \overline{D} = \overline{A} \cdot \overline{C} \cdot (\overline{B} \cdot D + B \cdot \overline{D}) = \overline{A} \cdot \overline{C} \cdot (B \oplus D)$$

$$x_2 = \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot D + \overline{A} \cdot \overline{B} \cdot C \cdot \overline{D} = \overline{A} \cdot \overline{B} \cdot (C \oplus D)$$

Se si disegnassero le mappe di Karnaugh delle due funzioni X_1 e X_2, sarebbe evidente che **le funzioni non sono riducibili**.

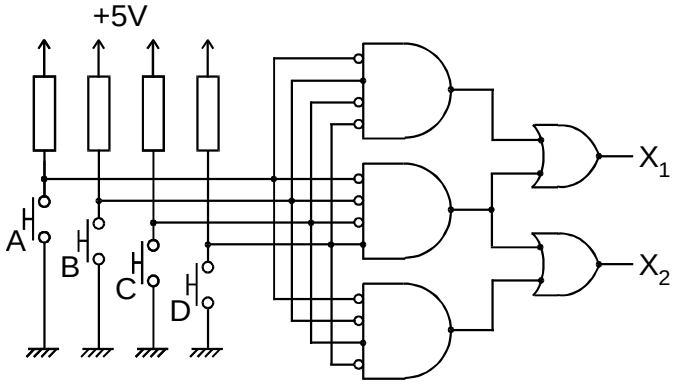


Fig. 9

Ma, a parte la possibilità di raggruppare opportunamente i due mintermini per semplificare la realizzazione del circuito, trattandosi di due distinte funzioni, tra le quali esiste un mintermine comune, una ovvia semplificazione è quella di utilizzare la funzione logica comune in entrambi gli OR.
In figura 9 si può vedere una delle possibili realizzazione del codificatore con i circuiti elettrici che trasformano la pressione del pulsante in un cambio di stato logico della corrispondente linea (da 0 ad 1).
In funzione del repertorio di funzioni logiche hardware disponibili sono possibili anche altre strategie di realizzazione per ridurre il numero e la complicazione dei componenti hardware.

Un problema tipico dei codificatori (**encoder in inglese**) può derivare dalla possibilità di **attivazione contemporanea di due ingressi della funzione**. Nel caso dell'esempio appena analizzato, essendo la funzione incompleta, tutto dipende da come si sono trattate le configurazioni delle variabili non previste. Con le espressioni delle uscite ricavate in precedenza in caso di azionamento contemporaneo di due pulsanti (ad esempio C e D) la rete darebbe in uscita sempre 00.

A	B	C	D	X_2	X_1
1	0	0	0	0	0
1	0	0	1	0	0
1	0	1	0	0	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	0	0
0	1	0	0	0	1
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	0	1
0	0	1	0	1	0
0	0	1	1	1	0
0	0	0	1	1	1

Tab. 2

Questo problema si risolve utilizzando una variante della funzione detta: "***priority encoder***" che si ottiene inserendo nella tabella di verità (tabella 2) tutte le combinazioni delle variabili d'ingresso ed imponendo che prevalga tra quelle attivate (cioè ad "1"), sempre la variabile d'ingresso associata al codice d'uscita più basso, cui è assegnata la priorità più alta.
Nell'esempio la priorità andrebbe da da A verso D.

A	B	C	D	X_2	X_1
1	X	X	X	0	0
0	1	X	X	0	1
0	0	1	X	1	0
0	0	0	1	1	1

Tab. 3

La tabella 2 può essere messa in forma sintetica se si adopera il simbolo X che significa che lo stato logico è "irrilevante". Il risultato è la tabella 3.
In questo caso è facile verificare che sono possibili semplificazioni della forma canonica SOP.

La funzione inversa dei "***codificatori***" è svolta dai "***decodificatori***" che ricevono in ingresso un codice binario ed in risposta selezionano l'elemento o la combinazione di elementi associato a quel codice. Questa è la funzione logica a cui abbiamo accennato parlando del codificatore. La logica interna dello strumento che ha sul pannello i quattro pulsanti, deve per prima cosa riconoscere il codice ricevuto (una delle quattro combinazioni dei bit x_1 ed x_2) per poter effettuare l'operazione richiesta. La funzione che effettua questa operazione è un decodificatore come quello a destra in figura 10. Riceve in ingresso la combinazione logica di x_1 ed x_2 ed attiva l'uscita corrispondente tra le quattro A, B, C e D. La corrispondente tabella di verità è riportata in tabella 4.

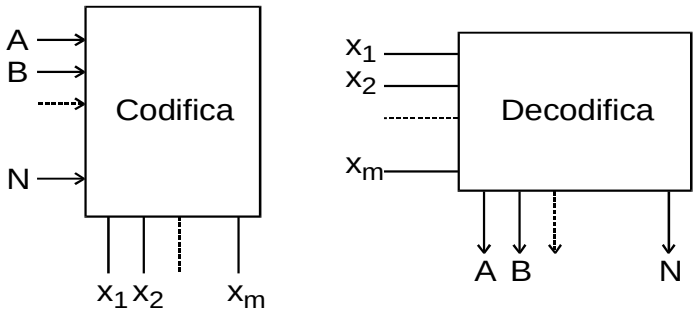


Fig. 10

X_2	X_1	A	B	C	D
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Tab. 4

Si tratta ovviamente di quattro funzioni delle stesse due variabili x_2 ed x_1, delle quali x_2 è la più significativa.
Ci si potrebbe domandare se è sensato avere in uno strumento due funzioni logiche in cascata, una che effettua una operazione e l'altra che effettua l'operazione opposta. E' chiaro che in un caso così semplice la soluzione migliore sarebbe stata quella di collegare i pulsanti direttamente alla logica interna dello strumento visto che ridurre da 4 a 2 linee i collegamenti tra pannello e logica è un piccolo vantaggio; ma se i pulsanti fossero stati molto più di quattro, come in una tastiera, diciamo in generale *n*, il passaggio da *n* fili ad un numero di fili pari a $log_2 n$ può essere notevole se *n* è grande.
Un ulteriore vantaggio di questa tecnica si apprezza, come vedremo nel modulo B del corso quando l'operazione di decodifica viene effettuata in maniera *distribuita*, quando cioè ciascuna combinazione delle variabili d'ingresso viene riconosciuta da una specifica funzione logica.

Un decoder in regime dinamico: Il demultiplexer.

In apparecchiature semplici il modo più immediato per effettuare la decodifica di un codice binario è quello di disporre sul pannello frontale dello strumento tante piccole lampade o LED (Light

Emitting Diode) e fare in modo che ogni lampada corrisponda ad uno dei codici binari utilizzati. Per aiutare un osservatore ad interpretare l'informazione si può incidere sul pannello il simbolo od il numero decimale che è associato all'operazione.
In casi come questo se non si vuole che rimanga accesa l'ultima lampada selezionata o sempre la stessa lampada, quella il cui codice corrisponde alla condizione di "riposo" (di solito lo 0), bisogna introdurre una variabile d'ingresso aggiuntiva, detta comunemente "abilitazione" (in inglese "enable").
Il problema è generale ed avremmo dovuto introdurlo anche per la codifica. Una realizzazione della funzione dell'esempio avrebbe selezionato perennemente l'ultima configurazione usata, col risultato che l'azionamento in successione dello stesso tasto non avrebbe prodotto variazioni. Una possibile soluzione è quella di rinunciare all'uso della configurazione di "riposo". Scegliere cioè una configurazione a cui il sistema si riporta quando non ci sono ingressi attivi. Nel caso della codifica si sarebbero potuti mettere solo tre pulsanti e far partire lo stato delle linee di uscita sempre da 00.
La tabella di verità della decodifica decimale dovrà essere opportunamente aggiornata perché **in assenza di abilitazione ("enable" a 0) nessuna delle uscite sia "attiva" e tutte le lampade siano spente**.
Quando l'abilitazione è "attiva" anche l'uscita che corrisponde al codice in ingresso va nello stato "attivo".
La presenza dell'ingresso di "***abilitazione***" mette tutti i codici sullo stesso piano e razionalizza il funzionamento della decodifica, ma fa in modo che essa possa essere utilizzata anche per un altro uso, solo apparentemente diverso.

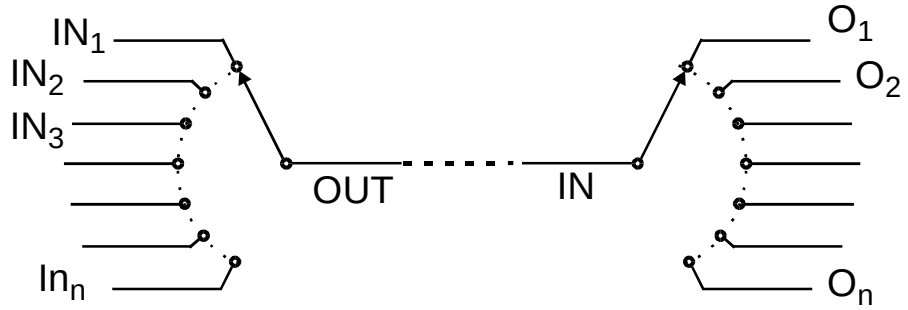


Fig. 11

Il fatto che lo stato logico dell'abilitazione venga trasferito in uscita sulla linea che corrisponde al codice presente sugli ingressi, ci suggerisce la possibilità di usare questa funzione logica come "***commutatore logico***" in cui la funzione della manopola è svolta dalle linee di ingresso del codice.
Lo schema di un commutatore elettromeccanico è riportato nella figura 11, sia nel caso che essa sia utilizzato per **inoltrare su una unica linea informazioni che arrivano, volta per volta, su un certo numero di ingressi**, che nella utilizzazione inversa, in cui le **informazioni provenienti da una sola linea sono smistate su diverse linee**.
La funzione logica che svolge il compito del commutatore di sinistra si chiama "***multiplexer da n ad 1***" o "***selettore di dati***", quella che effettua a livello logico l'operazione del commutatore di destra si chiama "***demultiplexer da 1 ad n*** "
Vale la pena di sottolineare che un commutatore elettromeccanico effettua l'operazione di "***multiplexer/demultiplexer***" a livello "***analogico***" perchè trasferisce segnali elettrici **conservandone l'ampiezza** e non soltanto livelli logici booleani, come fa una funzione logica.

Esistono circuiti integrati MOS che effettuano l'operazione di multiplexing analogico e sono, quindi una perfetta versione elettronica su silicio del commutatore elettromeccanico. Anche in questi dispositivi la funzione della manopola è svolta da un codice binario costituito da un numero di bit

adeguato al numero di linee: 1 bit per selezionare una linea tra due, 2 per scegliere una linea tra quattro linee, 3 per otto linee, 4 per sedici e così via.

E	D	C	B	A	0	1	2	3	4	5	6	7	8	9
0	X	X	X	X	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	1	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	1	0	0	0	0	0
1	0	1	0	1	0	0	0	0	0	1	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0	1	0	0	0
1	0	1	1	1	0	0	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	1	0	0	1	0	0	0	0	0	0	0	0	0	1

Tab. 5

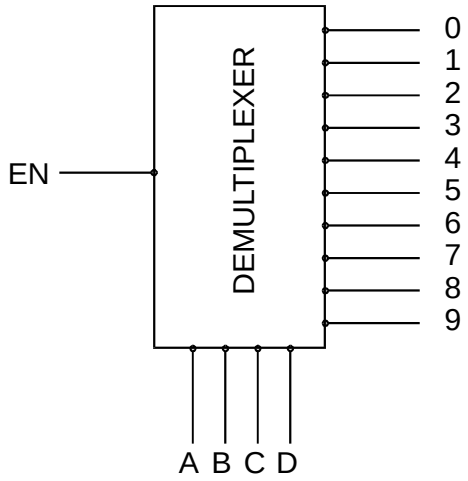


Fig. 12

La tabella di verità ed il simbolo di un "***demultiplexer/selettore di linee***" "***uno a dieci***" sono riportate in tabella 5 ed in figura 12, rispettivamente.

E' ovvio che esiste una versione logica anche del commutatore usato come "***multiplexer/selettore di dati***" in cui i livelli logici che si susseguono nel tempo sull'ingresso, il cui codice è presente sulle apposite linee, vengono trasferiti sull'unica uscita.
Lo schema sintetico di un **multiplexer** "***Dieci in uno***" con abilitazione è mostrato in figura 13. **La relativa tabella di verità è molto semplice perché, in corrispondenza di ogni combinazione di 0 e di 1 sugli ingressi A, B, C, D, l'uscita OUT assume il valore logico del corrispondente ingresso dati da IN 0 ad IN 9**.

E' chiaro che una versione professionale di questa funzione non potrà prescindere da un **ingresso di abilitazione**, altrimenti, anche in questo caso ci sarà sempre in uscita lo stato logico di uno degli ingressi, quello il cui codice corrisponde alla condizione delle linee A, B, C, D.

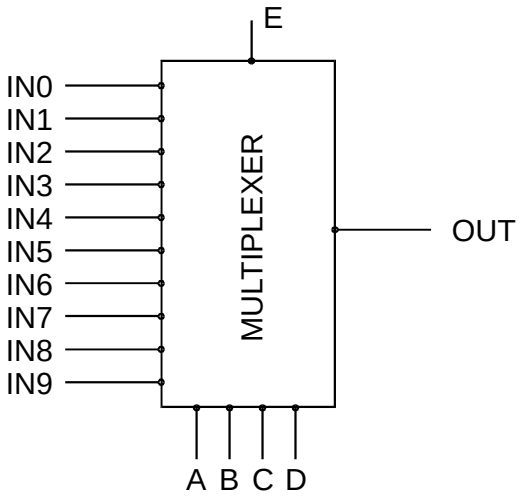


Fig. 13

Il multiplexer ed il demultiplexer sono due funzioni logiche molto usate e meritano, quindi, una analisi di dettaglio che però è opportuno riferire a strutture minimali
Consideriamo, quindi, un multiplexer "***da quattro ad uno***" che richiede due linee di selezione.
In figura 13 è utilizzato un possibile simbolo alternativo del multiplexer.

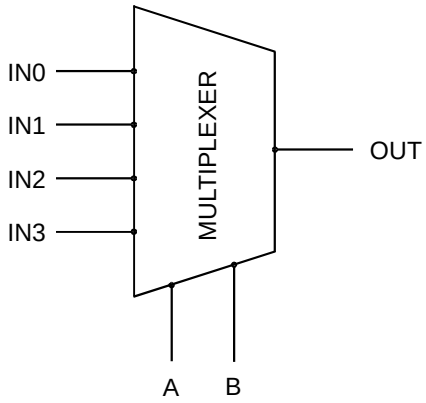


Fig. 14

Una possibile tabella di verità della funzione è mostrata in tabella 6, dove le colonne OUT* ed OUT** si riferiscono ad una logica elettronica in cui lo stato logico "***di riposo***" è "0" o "1", rispettivamente. Se non si vuole lasciare adito a dubbi ed esprimere compiutamente il problema logico si può adottare la tecnica della tabella 7:

B	A	OUT	OUT*	OUT**
0	0	IN 0	1	0
0	1	IN 1	1	0
1	0	IN 2	1	0
1	1	IN 3	1	0

Tab. 6

IN 0	IN 1	IN 2	IN 3	B	A	OUT
0	X	X	X	0	0	0
1	X	X	X	0	0	1
X	0	X	X	0	1	0
X	1	X	X	0	1	1
X	X	0	X	1	0	0
X	X	1	X	1	0	1
X	X	X	0	1	1	0
X	X	X	1	1	1	1

Tab. 7

La rete combinatoria che ne risulta è mostrata in figura 15.

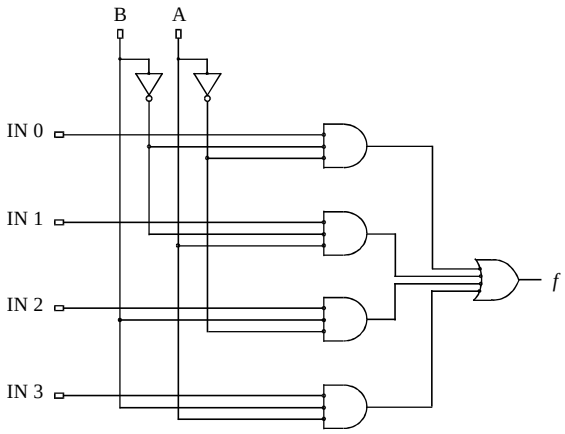


Fig. 15

Se si osserva con attenzione la tabella 7, è facile rilevare che la tabella di verità si compone di due parti logicamente ben distinte, una relativa agli ingressi A e B e una relativa ai quattro ingressi dati che sono tra loro in alternativa.
Se si realizzano separatamente le due parti si ha la struttura di figura 16

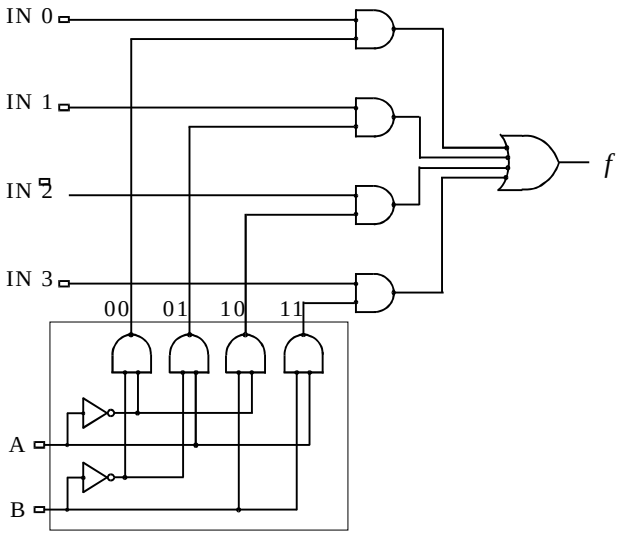


Fig. 16

Le considerazioni fatte per il multiplexer si possono ripetere quasi perfettamente per la sua funzione duale, il demultiplexer.
Anche in questo caso la tabella di verità si può scrivere, in maniera non equivoca, nella forma di tabella 8, in cui ogni combinazione degli ingressi A e B è associata ai due possibili valori booleani degli ingressi dati.
Dalla tabella 8 si perviene alla rete combinatoria di figura 17.

IN	B	A	OUT 0	OUT 1	OUT 2	OUT 3
0	0	0	0	0	0	0
1	0	0	1	0	0	0
0	0	1	0	0	0	0
1	0	1	0	1	0	0
0	1	0	0	0	0	0
1	1	0	0	0	1	0
0	1	1	0	0	0	0
1	1	1	0	0	0	1

Tab. 8

Anche in questo caso la sintesi logica della tabella può essere spezzata in due fasi.
Quella relativa agli ingressi A e B è la solita decodifica del codice dell'uscita a cui collegare logicamente l'ingresso. L'altra è il collegamento logico dell'ingresso all'uscita selezionata.

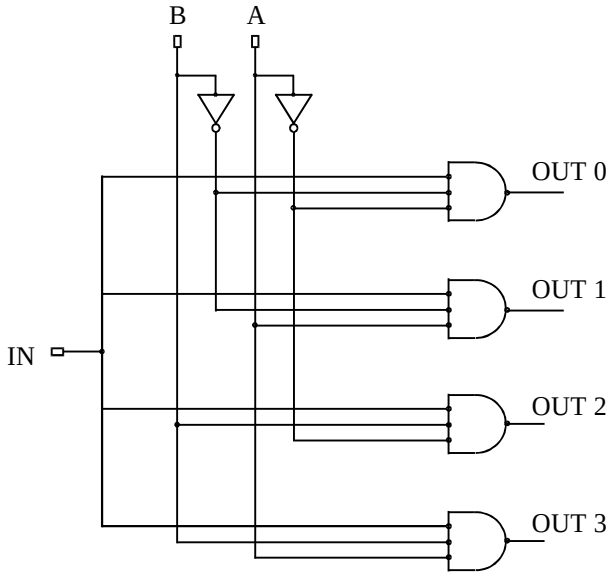


Fig. 17

Se si separano i due aspetti del problema, come è facile immaginare si arriva alla rete combinatoria di figura 18 in cui l'uscita attiva della decodifica va ad abilitare l'AND a due ingressi corrispondente e, quindi viene trasferito in uscita lo stato logico dell'ingresso.

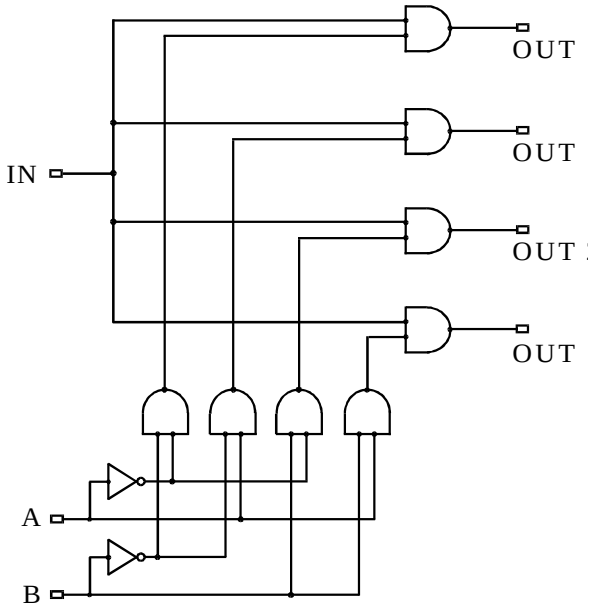


Fig. 18

Altri esempi di funzioni combinatorie: Sommatore binario

L'operazione di somma aritmetica binaria di due bit A e B da luogo a due funzioni booleane, una è la funzione ***Somma (S)*** e l'altra è la ***funzione Riporto*** (**C da carry in inglese**). La tabella di verità che le definisce è la tabella 9.
Dalla doppia tabella di verità si estraggono due forme canoniche SOP per S e C, chiaramente non riducibili che sono:

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

$$C = A \cdot B$$

A	B	S	C_n
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

Tab. 9

Il circuito booleano corrispondente si chiama "***semi-sommatore***" (**half adder**) ed è mostrato in figura 19. La funzione introdotta per rendere più sintetica la forma canonica è la funzione "***OR esclusivo***" che è disponibile in tutti i cataloghi di funzioni logiche realizzate con le varie tecnologie costruttive che sono state messe a punto negli ultimi decenni.

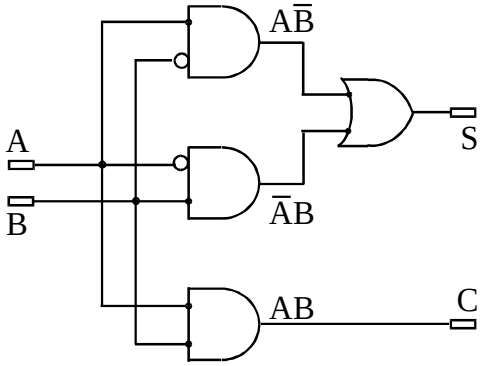


Fig. 19

Se si tratta, invece, di **sommare tra loro due bit, ciascuno dei quali fa parte di un numero di più bit**, bisogna prevedere la possibilità di **un riporto in ingresso proveniente dalla somma di bit meno significativi** ed, allora, la tabella di verità deve prevedere un ingresso in più (tabella 10).

A	B	C_{n-1}	S	C_n
0	0	0	0	0
1	0	0	1	0
0	1	0	1	0
1	1	0	0	1
0	0	1	1	0
1	0	1	0	1
0	1	1	0	1
1	1	1	1	1

Tab. 10

Da cui si ricavano le seguenti espressioni algebriche della funzione somma S e di quella riporto C.

$$S = A\overline{B}\overline{C} + \overline{A}B\overline{C} + \overline{A}\overline{B}C + ABC = A \oplus B \oplus C$$

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

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

Lo schema sintetico del "***sommatore completo***" o "***full-adder***" è riportato in figura 20.

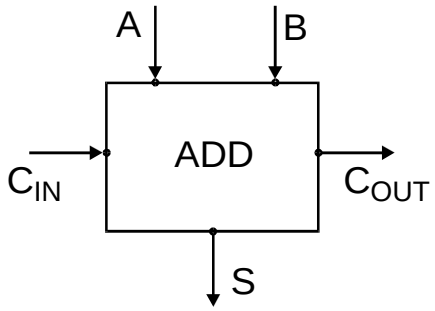


Fig. 20

La configurazione simbolica dettagliata delle due funzioni S e C è mostrata in figura 21 in cui l'autore del testo da cui è tratta ha adottato un simbolo per la funzione NOT particolarmente originale. Non c'è da stupirsi l'Informatica e l'Elettronica digitale sono campi in espansione vorticosa e le convenzioni sui simboli non hanno il tempo di consolidarsi.

Lo schema è relativo alla forma non minimizzata

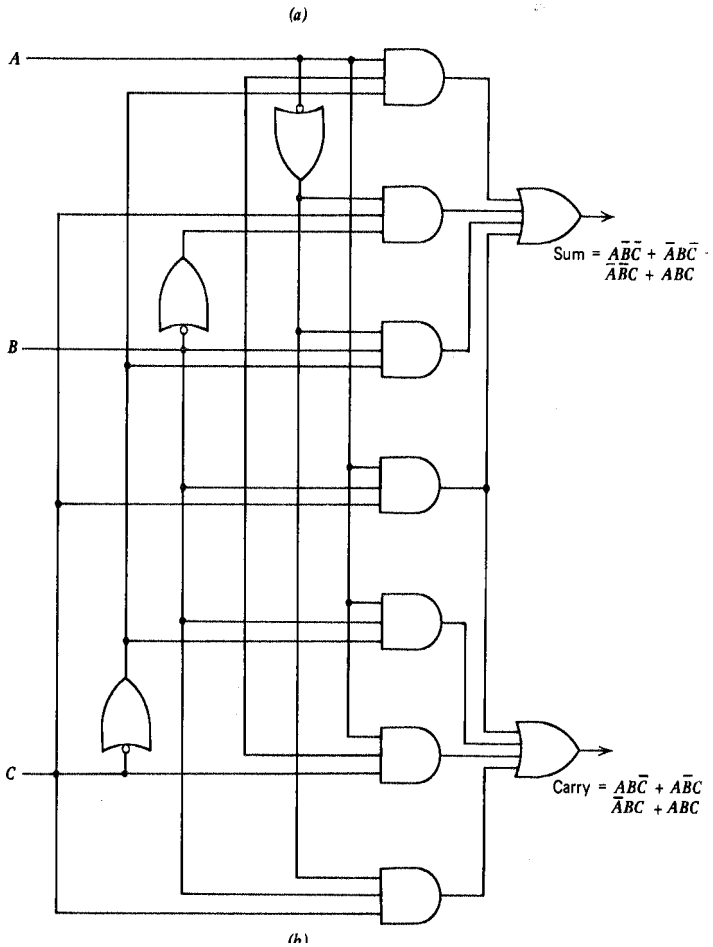


Fig. 21

Lo schema sintetico di un sommatore ad n bit è riportato in figura 22.

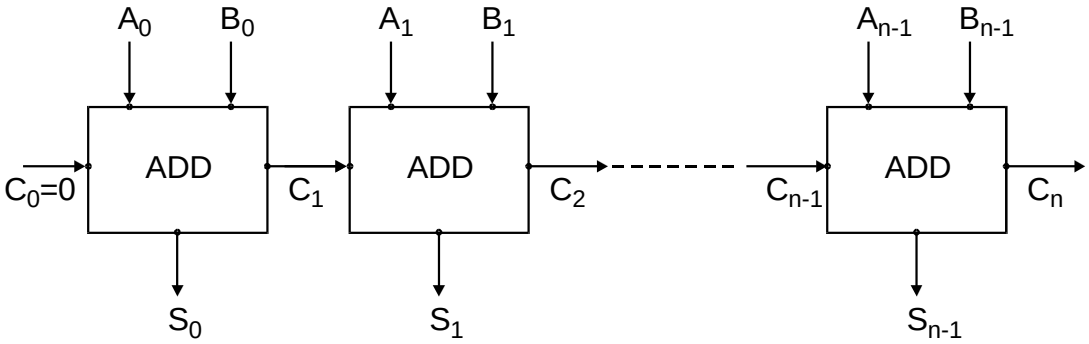


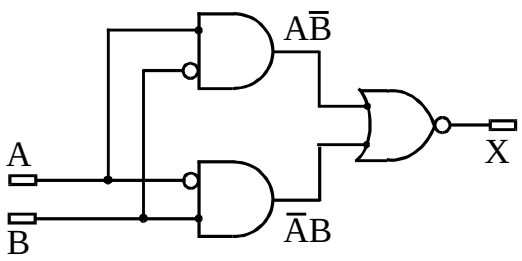
Fig. 22

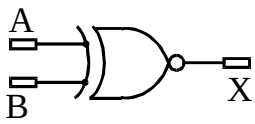
Comparatore di numeri binari

In Elettronica digitale ed in Informatica capita spesso di dover confrontare due numeri binari e decidere se sono uguali. La funzione logica che esegue l'operazione si chiama "***comparatore***".

Un comparatore ad un bit in grado di dire se il bit A è uguale al bit B o se A ≠ B è molto semplice da realizzare. Basta fare riferimento alla funzione XOR o OR-esclusivo introdotta per la prima volta nel ricavare una forma compatta della funzione "*semi-sommatore*"

In figura è rappresentata la funzione che si ottiene negando l'uscita di un OR-esclusivo (A ⊕ B) cioè un NOR-esclusivo e la relativa tabella di verità





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

Fig. 23

La tabella di verità consente di ricavare immediatamente che X = 1 solo se A = B.
La stessa cosa si può ricavare dalla forma algebrica della funzione.
Infatti:

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

Se si vogliono confrontare numeri di quattro bit il circuito di figura 24 è in grado di dire se tutti i bit sono a due a due uguali.
E' ovvio che se anche il valore logico di un solo bit non coincide l'uscita A=B è al valore "0" ed non è possibile dire altro sul valore comparativo dei due numeri.

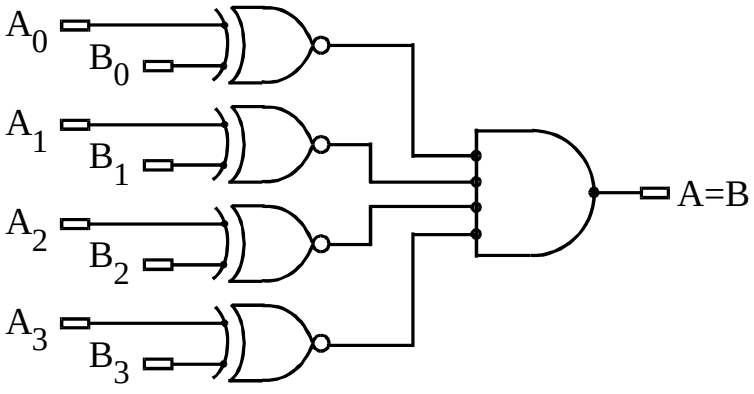


Fig. 24

Se si vuole un circuito in grado di effettuare una comparazione completa in modo da poter stabilire in caso di diseguaglianza se A > B oppure A < B bisogna sintetizzare un circuito logico che corrisponda alla seguente tabella di verità:

A	B	A>B	A=B	A<B
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

Tab. 11

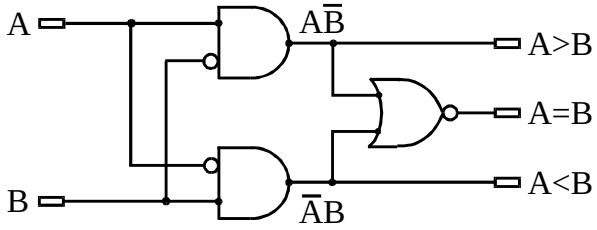


Fig. 25

La funzione logica di figura 25 risponde perfettamente alla richiesta di comparazione completa di due bit ma, è ormai chiaro, da quanto detto a suo tempo nel passaggio dal "*semi-sommatore*" al "*sommatore completo*", che il circuito deve essere messo in una forma modulare per poter confrontare numeri di molti bit. Ma di questo aspetto ci occuperemo più avanti dopo aver parlato diffusamente della rappresentazione binaria dei numeri e degli elementi di memoria digitale.

Funzioni logiche "*wired*".

Abbiamo accennato in precedenza al fatto che esistono situazioni in cui è utile ed efficiente collegare elettricamente tra loro le uscite di più funzioni logiche. Vedremo, nel modulo B, quando parleremo di strutture logiche complesse che si chiamano "***bus***", che questi casi esistono, e vengono risolti usando funzioni logiche con circuiti d'uscita "***speciali***", in cui **la possibilità che si manifesti il conflitto elettrico di cui si è parlato i precedenza è stata eliminata in partenza, sopprimendo nel circuito d'uscita l'interruttore superiore** (funzioni logiche "***Open Collector***" (O.C.) od "***Open Drain***" (O.D.), a seconda della tecnologia con cui sono costruite).

La funzione dell'interruttore eliminato, che era quella di portare l'uscita al livello di tensione "alto" viene affidata ad una resistenza aggiunta all'esterno, che si dice appunto di "***pull-up***".

L'operazione corrisponde, in un certo senso, a sostituire una corda, che mantiene sospeso ad una certa altezza un peso, con un elastico. Se l'elastico esercita una forza sufficiente, il peso sta più o meno alla stessa quota ma, mentre prima, se qualcuno avesse tentato di tirare giù con forza il peso, la corda si sarebbe spezzata, mettendo in crisi il sistema, con l'elastico il peso può arrivare anche fino al pavimento senza che il sistema si danneggi.

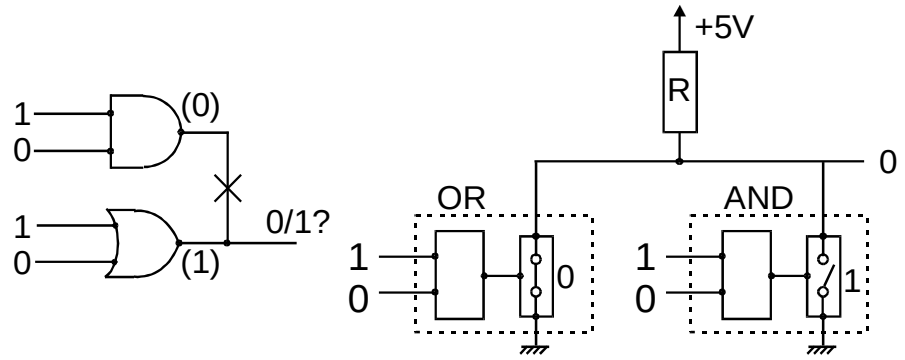


Fig. 26

Se si osserva la figura 26 si vede che accanto al circuito che nella figura 7 era stato dichiarato improponibile è disegnata una situazione elettricamente corretta e in questo caso non ci sono dubbi che l'uscita sarà a "0". La nuova struttura, infatti fa prevalere in maniera evidente l'interruttore "chiuso" e, quindi il livello "basso".

Questa semplice osservazione apre una interessante prospettiva e cioè: se si assegna il livello di tensione "basso" (0 Volt) al valore booleano "1" e il livello di tensione "alto" (+5 Volt), si adotta cioè una ***logica negata***, il filo di collegamento tra due o più funzioni con circuito di uscita Open Drain realizza una funzione logica OR che, per il fatto di scaturire da un semplice filo di collegamento, si dice "***wired OR***". Questo tipo di funzione si adotta, come vedremo più avanti quando si deve realizzare un OR tra un numero a priori imprecisato di unità logiche dislocate a notevole distanza l'una dall'altra. (figura 27). E' ovvio che in logica positiva la stessa struttura realizzerebbe un "***wired AND***".

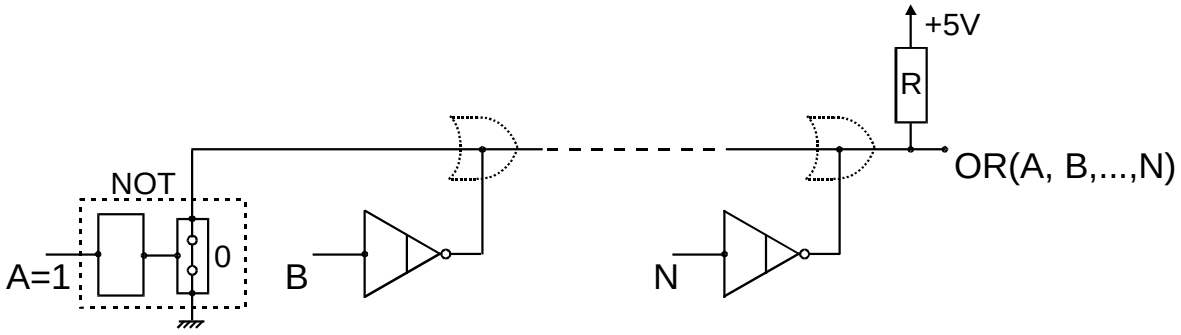


Figura 27

Le funzioni logiche "O.D." e la struttura "wired OR" che con esse si realizza forniscono tempi di propagazione molto lunghi rispetto ad una funzione logica tradizionale e, pertanto, questa soluzione viene usata solo quando è stretta mente indispensabile.
Se, così non fosse tutti gli OR finali di tutte le forme SOP potrebbero essere realizzate con funzioni AND con uscita Open Drain e con inversione finale per realizzare l'OR con la tecnica "***wired***".

Altre tecniche per la sintesi di una funzione booleana

La disponibilità di circuiti integrati che realizzano funzioni logiche complesse in tecnologia MSI o LSI ha reso possibili ed anche convenienti tecniche per la sintesi di funzioni booleane che sono certamente non ottimali da un punto di vista astratto, perché danno luogo a reti logiche tutt'altro che minimizzate, ma sono molto facili da realizzare.

Una di queste si basa sull'affermazione, che sarà subito dimostrata che: **Qualunque funzione logica ad n variabili può essere sintetizzata con un decoder ad n ingressi ed una funzione OR ad un numero di ingressi pari (o superiore) al numero di "mintermini" della tabella di verità.**
La dimostrazione può essere semplicemente affidata alla figura 28 nella quale è stato adottato per il decoder un simbolo che viene usato da alcuni autori ed è adottato anche da alcuni programmi di progettazione elettronica con l'ausilio dell'elaboratore.
Accanto è indicato il corrispondente simbolo per il codificatore

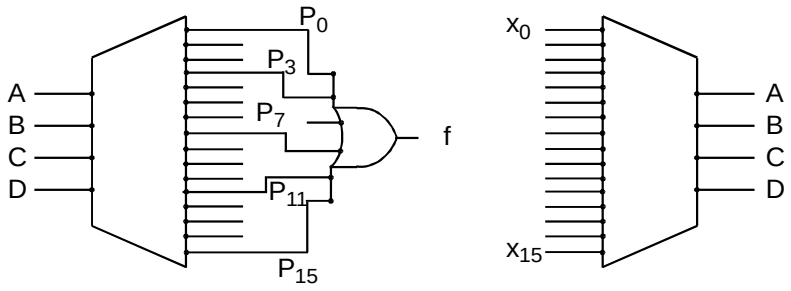


Fig. 28

Anche il multiplexer può essere utilizzato per sintetizzare funzioni logiche.
In figura 29 è riportata la realizzazione con questa tecnica di una funzione di quattro variabili con solo quattro "1" nella tabella di verità, ma è ovvio che il maggior vantaggio si ha quando gli "1" sono molti.

$f(A,B,C,D)=P_1+P_2+P_{12}+P_{15}$

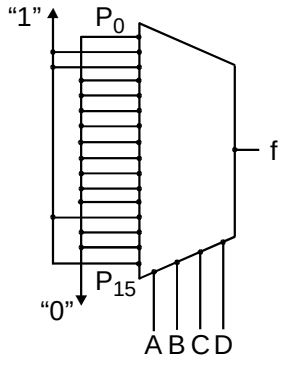


Fig. 29

E' importante rilevare che la tecnica si applica alla "***forma canonica***" della funzione.
Se la funzione non è in forma canonica, bisogna portarcela effettuando a ritroso il procedimento di minimizzazione e ri-introducendo tutte le ridondanze logiche eliminate.
Se per esempio la funzione di quattro variabili avesse avuto un termine del tipo:

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

esso avrebbe dovuto essere espanso nei quattro termini:

$$A \cdot \overline{B} \cdot C \cdot D + A \cdot \overline{B} \cdot \overline{C} \cdot D + A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} + A \cdot \overline{B} \cdot C \cdot \overline{D}$$

Una ulteriore interessante notazione è che utilizzando una funzione NOT esterna si può abbassare di una unità il numero di bit del codice di selezione del multiplexer.
Se per esempio si vuole sintetizzare la funzione di tre variabili:

$$f(X,Y,Z) = X \cdot Y + \overline{Y} \cdot \overline{Z}$$

poiché la funzione non è in forma canonica, bisognerebbe espanderla per avere che tutti i termini contengano tutte le variabili.
Se, però si scelgono *X* ed *Y* come "*variabili di selezione*" del multiplexer "***quattro in uno***" che si vuole usare, il termine che contiene solo le variabili X ed Y si può lasciare immutato e si espandono solo quelli che contengono la terza variabile, *Z*. Nell'esempio ce n'è uno solo e si ottiene:

$$f(X,Y,Z) = X \cdot Y + X \cdot \overline{Y} \cdot \overline{Z} + \overline{X} \cdot \overline{Y} \cdot \overline{Z}$$

In figura 30 è mostrato come si presenta globalmente la funzione sintetizzata.

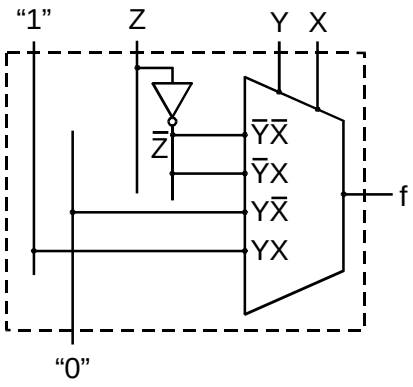


Fig. 30

In figura 31 è sviluppato un ulteriore esempio di questa tecnica.

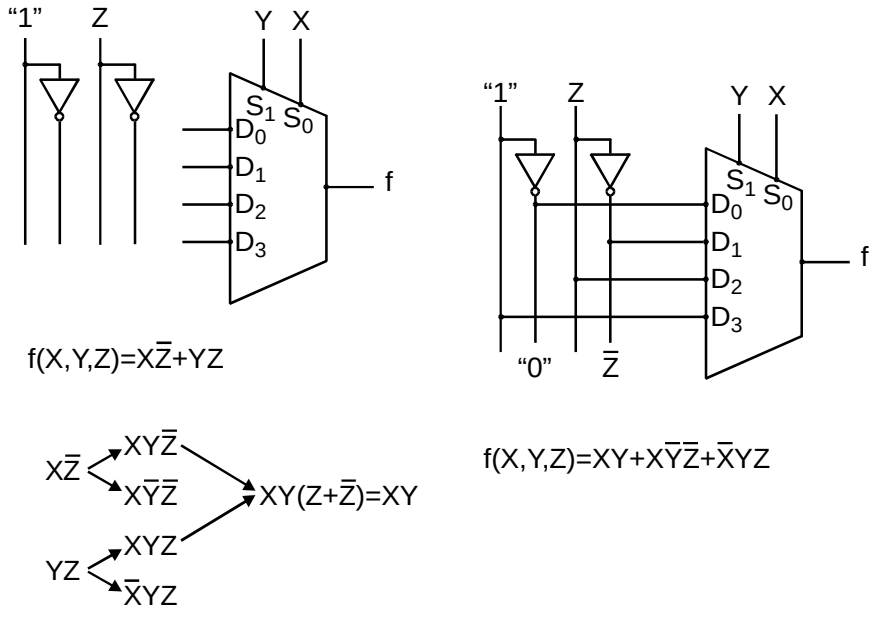


Fig. 31

Vedremo più avanti che la tecnica più duttile e più economica per sintetizzare funzioni logiche complesse è quella di utilizzare una memoria.

Torneremo a parlare più avanti di reti combinatorie perché una comprensione completa di alcune strutture richiede una conoscenza dettagliata della codifica elettronica binaria delle informazioni che verrà dopo lo studio dei capitoli 1 e 2.
Vedremo anche reti combinatorie interessanti che però richiedono la conoscenza degli elementi di memorizzazione.