5.1. Reti combinatorie:definizione

Una rete combinatoria é un circuito elettronico digitale in grado di calcolare in modo automatico una funzione binaria di una o più variabili booleane.
Lo schema generale di una rete combinatoria é il seguente:

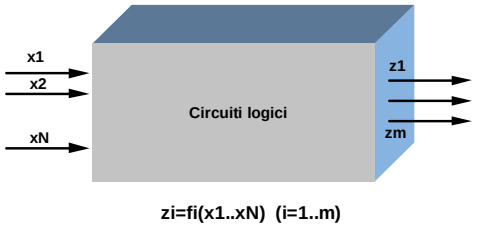

Figura 5.1: *Generico schema di rete combinatoria*

I valori delle uscite in ogni istante sono univocamente determinati dai valori presenti contemporaneamente sugli ingressi Una rete combinatoria si compone di elementi di calcolo logico elementare che sono le porte ovvero gli operatori logici AND OR e NOT fino ad ora trattati.
Ad esempio, una descrizione completa del comportamento di una rete combinatoria a 3 ingressi e due uscite é data dalla solita tabella di verità ed un circuito cosi descritto:

x_0 x_1 x_2	z_0 z_1
0 0 0	0 0
0 0 1	0 1
0 1 0	0 0
0 1 1	0 1
1 0 0	0 0
1 0 1	1 1
1 1 0	0 1
1 1 1	1 1

Come già osservato nel capitolo precedente, possiamo dunque verificare che il comportamento di un circuito combinatorio può essere equivalentemente descritto da un'espressione booleana.
Una rete combinatoria può essere descritta disegnando lo **schema circuitale** ossia un collegamento di circuiti elementari che abbiamo visto fino ad ora e che sono le **porte AND OR NOT** tramite linee.

Possiamo Identificare tre tipi di linee:

linee di ingresso, etichettate con i nomi delle variabili booleane di ingresso

linee di uscita, etichettate con i nomi delle variabili di uscita

linee interne, ciascuna delle quali collega l'uscita di una porta con l'ingresso di un'altra porta

Lo ***schema circuitale*** di una rete combinatoria deve soddisfare i seguenti requisiti:

- Ogni ingresso di ciascuna porta presente nella rete deve essere collegato ad una linea di ingresso oppure ad una linea interna.
- L'uscita di ogni porta presente deve essere collegata ad una linea di uscita oppure ad una linea interna.
- Il collegamento di porte tramite linee non deve dare luogo a cicli.

Un esempio é mostrato in figura:

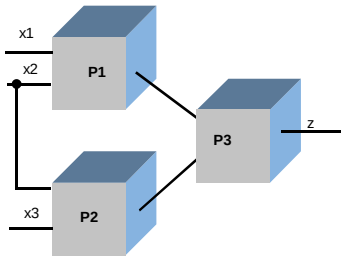

Figura5.2: *Generico schema circuitale.*

5.2. Reti combinatorie notevoli

Nel seguito vedremo alcuni esempi di sintesi di circuiti combinatori "notevoli" i quali implementano funzioni di uso comune, tanto da essere commercialmente disponibili sotto forma di singolo circuito integrato. Si introdurranno questi circuiti seguendo la metodologia di sintesi vista nel capitolo 4.

5.2.2 Multiplexer

il MULTIPLEXER (*o selettore di ingresso*) è un circuito combinatorio con *n* ingressi e un'unica uscita, e $\log_2 n$ variabili di comando *s*. Quindi lo scopo di un MULTIPLEXER è quello di selezionare in un'unica uscita *n* ingressi. Supponendo di considerare solamente due ingressi indicati con *a* e *b*, lo schema logico del MULTIPLEXER sarà:

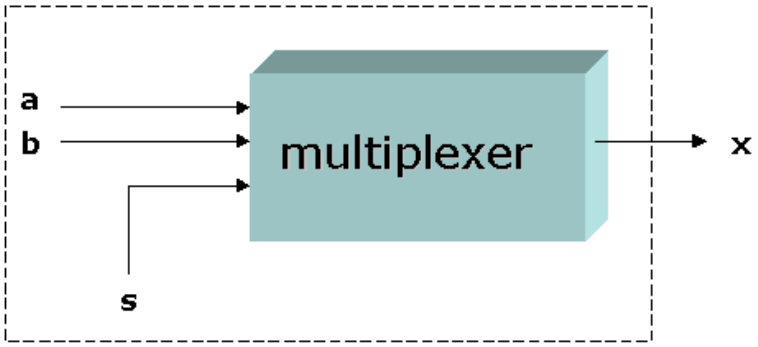

Figura 5.3: *schema logico di un multiplexer*

Ci sono due ingressi a_0 e a_1 e un selezionatore *s* tale che una volta inseriti gli ingressi in un circuito che stiamo per realizzare, restituisce come valore di output un valore x. Il circuito deve rispondere a questi comandi:

$$x = a_0 \text{ se e solo se } s = 0$$
$$x = a_1 \text{ se e solo se } s = 1$$

Possiamo immaginare il selezionatore *s* come una sorta di interruttore tale per cui se s=0 l'interruttore si porta a a_0, se invece s=1 l'interruttore si posiziona in a_1, come da figura sotto:

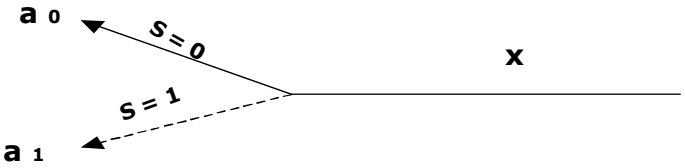

Figura 5.4: *comportamento del selezionatore s.*

L'uscita x sarà connessa o con a_1 o con a_2 a seconda di dove si posiziona s, o dei valori a_i che *s* seleziona. Quindi la posizione di a_i è determinata dal selettore s. Il MULTIPLEXER che stiamo considerando è un circuito dato da tre ingressi: il selezionatore e le due sequenze di *n* bit che possono essere paragonati a due fili, e un uscita data da uno dei due ingressi a seconda del valore a sua volta selezionato da s.
La tabella di verità di un multiplexer a due ingressi e un selezionatore sarà:

Tabella 5.1: *tabella di verità di un multiplexer a due ingressi e un selezionatore*

a_1	a_2	s	x
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Osserviamo dalla tabella 5.1, che quando s=0 allora verrà copiato in uscita a_0, quando invece s=1 il circuito restituirà in uscita l'ingesso a_1 sempre in corrispondenza della riga considerata.

Ad esempio, in corrispondenza della prima riga, l'uscita x vale 0, questo perché essendo s=0, significa che il comando di controllo *s* ha selezionato il bit a_0=0.

Passando alla seconda riga, l'uscita x è ancora pari a 0, perché essendo s=1, il selezionatore ha copiato in uscita bit a_1=0.

Procedendo fino alla quarta riga della tabella 5.1, osserviamo che il valore dell'output x è pari a 1 perché essendo s=1, è andato a selezionare il bit a_1 che nella quarta riga assume il valore 1, portandolo dunque come valore di uscita. Si procede così fino a completare la tabella di verità del MULTIPLEXER di uscita x. Possiamo a sua volta sintetizzare la tabella di verità per X=1 con un circuito AND OR componendo l'equivalente espressione booleana:

$$\overline{a_0}\, a_2 s + a_0 \overline{a_1}\ \overline{s} + a_0 \overline{a_1}\, \overline{S} + a_0 a_1 s = 1$$

che semplificata si riduce a

$$\overline{a_0}\, a_2 s + a_0 \overline{a_1}\, \overline{S} + a_0 a_1 = 1$$

questa espressione può essere a sua volta sintetizzata dal seguente circuito AND OR:

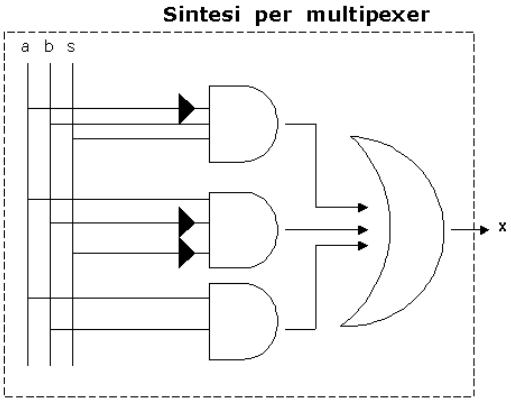

Figura 5.5: *sintesi di un circuito multiplexer a tre ingressi*

Dati allora due ingressi, a_0 a_1, il MULTIPLEXER seleziona in uscita attraverso s il valore di a_0 in x se s = 0, o di a_1 in x se s = 1.

In generale possiamo definire il **Multiplexer** come **quel blocco logico che permette di derivare su un'unica uscita, un segnale proveniente da uno tra n possibili ingressi. Formalmente è una rete logica avente 2^n ingressi di tipo dati e n ingressi di tipo *segnali di controllo* ed uno di uscita: in ogni istante il dato di ingresso corrispondente alla configurazione dei segnali di controllo viene posto in uscita.**

Se volessimo ora selezionare anziché due oggetti, quattro ingressi dati da a_0 a_1 a_2 a_3 e decidere quali di questi vale l'uscita x , dovremmo aggiungere un bit in più di selezione, poiché potendo un selezionatore scegliere con un bit solamente tra due valori, bisognerà aggiungere un altro selezionatore per decidere tra i restanti altri due ingressi.

Avremo così due selezionatori s_0 ed s_1 che decideranno quali dei quattro ingressi dovrà essere associato all'uscita x.

Graficamente un circuito MULTIPLEXER costituito da quattro ingressi di entrata dati dai quattro fili di bit a_0,a_1,a_2,a_3 e due selezionatori s_0,s_1 avrà una rappresentazione di questo tipo:

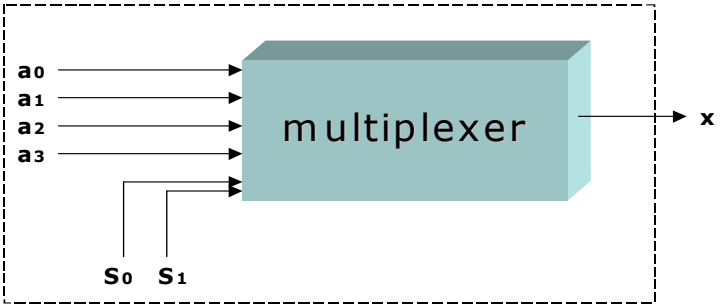

Figura 5.6: *schema logico di un multiplexer a 4 ingressi*

La tabella di verità del multiplexer in questo caso non viene interamente tabulata perché avendo 6 ingressi: ossia 3 bit di entrata e due bit di selezione, dovremmo riportare 2^6 = 64 righe di casi possibili. Consideriamo allora le prime 8 righe:

Tabella 5.2: *tabella di verità di un multiplexer a 4 ingressi e 2 selezionatori*

a_0	a_1	a_2	a_3	s_0	s_1	x
0	0	0	0	0	0	0
0	0	0	0	0	1	0
0	0	0	0	1	0	0
0	0	0	0	1	1	0
0	0	0	1	0	0	0
0	0	0	1	0	1	0
0	0	0	1	1	0	0
0	0	0	1	1	1	1

Osservando la tabella 5.2, alla prima riga abbiamo s_0 =0 ed s_1=0, quindi l'oggetto selezionato associato alla prima uscita x,è a_0=0. Alla seconda riga, abbiamo s_0s_1=01, quindi in uscita è selezionato l'ingresso a_1=0. Alla terza riga, abbiamo s_0s_1=10, ossia il terzo ingresso (infatti $10_{(2)} = 2_{(10)}$) per cui in uscita sarà copiato a_2=0. Alla quarta riga, abbiamo s_0 s_1 = 11 che seleziona il quarto ingresso, ossia a_3=0.

Si procede con lo stesso ragionamento sempre riportando il valore di uscita x ottenuto dagli ingressi selezionati da s_0 ed s_1. Come già ricordato, la tabella è incompleta perché dovremmo in realtà tabulare 64 possibili combinazioni.

Considerando un MULTIPLEXER a 4 ingressi e due selezionatori, è evidente che il circuito che lo realizza sarà un po' più complesso di quello precedente a 2 sole entrate e 1 selezionatore. Ricordando però quanto evidenziato al paragrafo 4.10 del precedente capitolo, ossia che un circuito complesso può essere considerato come composto da una serie di circuiti più semplici, possiamo partire dalla composizione di questi ultimi, come abbiamo fatto per il sommatore, e una volta impostati, inserirli in un circuito più complesso.

Sulla base di questo ragionamento, partiamo allora da due selezionatori MULTIPLEXER a due bit di ingresso rispettivamente per a_0 a_1 e per a_2 a_3 come da figura:

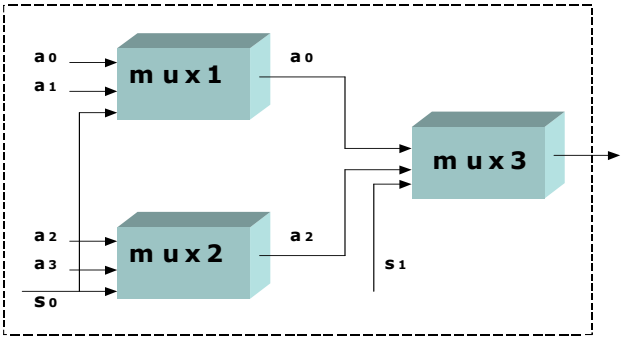

Figura 5.7: *schema di implementazione di un circuito combinatorio a 4 ingressi e un'uscita*

I due primi MULTIPLEXER (MUX1 e MUX2) così costruiti restituiscono un primo valore x_0 E dato da una prima selezione di s_0 tra i quattro ingressi in entrata, i quali vengono a sua volta inseriti in un altro MULTIPLEXER (MUX3) a due bit di ingresso e uno di selezione, ottenendo come valore finale x_1 associato ad uno dei due bit in entrata a sua volta selezionati dai due MULTIPLEXER precedenti.

Se uniamo con un solo filo il primo selezionatore s_0 ai primi due multiplexer indicati in figura con MUX1 e MUX2, il circuito si comporterà in modo tale che il primo selezionatore s_0, seleziona due ingressi tra i quattro inseriti nei MULTIPLEXER 1 e 2. Per cui il primo filo s_0 farà si che se ad esempio s_0=0, darà come primo valore di uscita a_0 a_1, selezionando così primi due bit che occupano la prima posizione in ingresso per ciascun sottocircuito MULTIPLEXER 1 e 2.

Per meglio capire il funzionamento del MULTIPLEXER a quattro ingressi, possiamo considerare la seguente figura che descrive il comportamento del circuito nel caso di s_0 = 0:

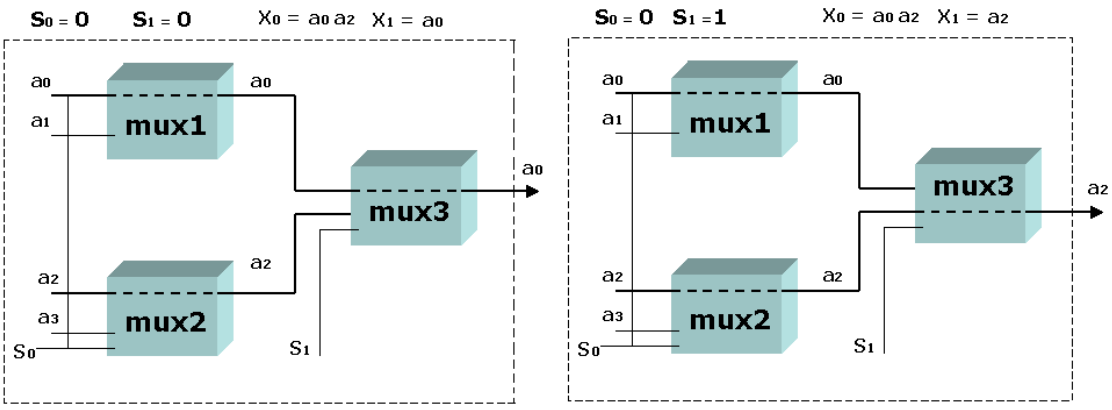

Figura 5.8: *meccanismo di selezione nel caso di* ***s_0 = 0***

La figura 5.8 può essere così descritta:

Se $\mathbf{s_0 = 0} \Rightarrow$

$X_0 = a_0 \Rightarrow$ se $\mathbf{s_1 = 0} \Rightarrow X = X_0 = a_0$

$X_1 = a_2 \Rightarrow$ se $\mathbf{s_1 = 1} \Rightarrow X = X_1 = a_0$

Analogo funzionamento ne caso di $s_0 = 1$

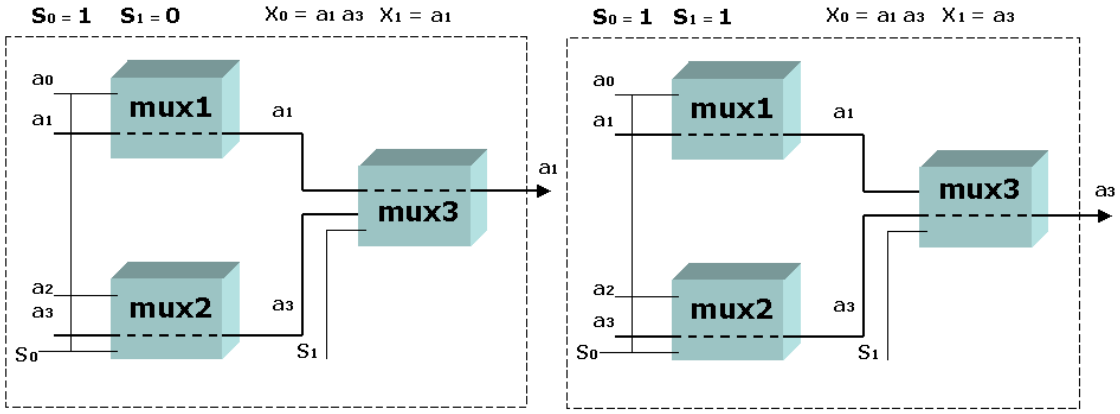

Figura 5.9: *meccanismo di selezione nel caso di* ***$s_0 = 1$***

Se $\mathbf{s_0 = 1} \Rightarrow$

$X_0 = a_1 \Rightarrow$ se $\mathbf{s_1 = 0} \Rightarrow X = X_0 = a_1$

$X_1 = a_3 \Rightarrow$ se $\mathbf{s_1 = 1} \Rightarrow X = X_1 = a_3$

Il tutto può essere riassunto in una tabella dei possibili casi di selezione dei quattro ingressi dati dal selezionatore:

$\mathbf{s_0}$	$\mathbf{x_0}$	$\mathbf{x_1}$	$\mathbf{s_1}$	**U**	**U**
0	a_0	a_2	0	x_0	a_0
1	a_1	a_3	0	x_0	a_1
0	a_0	a_2	1	x_1	a_2
1	a_1	a_3	1	x_1	a_3

Con il circuito della figura 5.7, siamo riusciti dunque a sintetizzare una tabella di 64 righe unendo semplicemente tre sottocircuiti MULTIPLEXER, riducendo notevolmente così la dimensionalità del problema, poiché i tre circuiti così impostati si comportano esattamente come la tabella a 64 righe .

Possiamo ulteriormente complicare lo schema della figura 5.7, considerando otto ingressi da selezionare e impiegando stavolta tre selezionatori, dove i primi due

uniscono due circuiti MULTIPLEXER da quatto ingressi ciascuno, mentre l'altro unisce due circuiti da due ingressi.

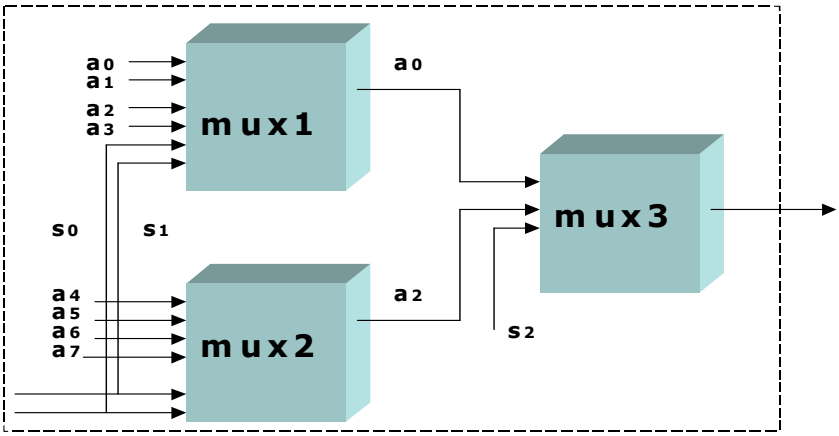

Figura 5.10: *schema di implementazione di un circuito combinatorio a 8 ingressi e un'uscita*

I primi due selezionatori s_0 ed s_1 selezionano sia in MUX1 che in MUX2, uno tra quattro ingressi, mentre il terzo selezionatore s_2, combina insieme uno dei due risultati a sua volta selezionati. Il MULTIPLEXER 3 ha un solo bit di selezione, s_2, mentre i MULTIPLEXER 1 e 2 hanno due bit di selezione: s_0 ed s_1. Siamo così riusciti a sintetizzare con il circuito appena descritto, una tabella di 2^{11} righe.

5.2.3 Decoder

Consideriamo ora un altro circuito notevole detto DECODIFICATORE o *Decoder*. Un *decoder* è un circuito che presenta *n* ingressi e 2^n uscite, dove tutte le configurazioni in ingresso sono possibili. Considerando il caso di due soli ingressi, il decoder sarà rappresentato dalla seguente figura:

Figura 5.11: *schema logico di un DECODER*

Il decoder interpreta gli ingressi come u numero intero espresso in codifica binaria e lo invia in uscita ponendo a 1 la i-esima linea di uscita X_i, mantenendo a 0 tutte le altre.

Osservando la figura 5.11, dati allora due ingressi s_o ed s_1 come due bit di un numero intero, il circuito da in uscita tutti i fili X_i pari a zero, tranne quello contrassegnato dal pedice corrisponde al numero inviato in ingresso, per cui $X_0=1$ se e solo se $s_o=0$ e $s_1=0$, ossia se il bit in entrata è dato dalla sequenza 00; mentre $X_0=0$ in tutti gli altri casi in cui s_o e s_1 sono diversi dalla sequenza 00.
Quindi la sequenza di bit 00, data da s_os_1, individua il numero che costituisce il pedice di ciascuna uscita X_i. Avremo dunque che $X_1=1$ se e solo se $s_o=1$ e $s_1=0$, (ovvero il bit $s_1s_0=01$) perché il numero binario dato dalla sequenza $s_1s_o=01$ corrisponde al numero 1 (ossia pedice di X_1), e ancora $X_2=1$ se e solo se $s_1s_o=10$ (pedice di X_2) e così via. Quindi sulla base delle combinazioni binarie dei due ingressi s_1s_o, viene deciso quale bit verrà attivato in uscita, portando così l'uscita pari a 1.
Possiamo allora considerare le uscite X_i come ai numeri dati dal pedice che le identifica, e pensarle come a dei fili che vengono accesi ogni volta che il selezionatore s li seleziona.
Il circuito considerato è detto decodificatore perché essendo il numero entrante s codificato in binario, decodifica appunto la parola codice presente sugli ingressi, tale che per ogni parola codice esiste una diversa uscita binaria, mentre in ogni momento, la singola uscita X_i applicata in ingresso e associata alla parola codice, vale 1.
Il circuito della figura 5.1 in realtà è molto semplice perché ha solamente quattro valori in uscita da selezionare, per cui la tabella di verità del decoder a due ingressi sarà

Tabella 5.3: *tabella di verità di un decoder a due ingressi:*

s_1	s_o	X_0	X_1	X_2	X_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Il corrispondente schema circuitale sarà composto da quattro circuiti separati per ogni uscita $X_i=1$:

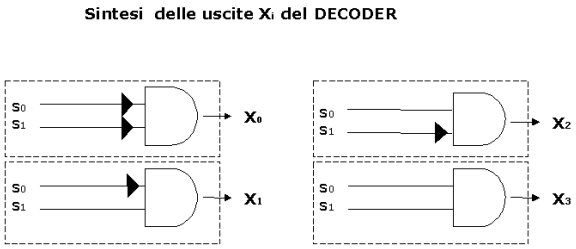

Figura 5.12: *circuiti relativi alle uscite del decoder a due ingressi*

Vediamo che la rete AND OR diventa una rete ad una sola porta AND perché la porta OR in questo caso non serve.

5.2.4 Demultiplexer

Esiste una variante del decoder che si chiama DEMULTIPLEXER, che coincide con il decoder appena descritto, ma che in più è dotato di ingresso di abilitazione: cioè ai due ingressi della figura 5.11, (più in generale n ingressi), se ne aggiunge uno indicato con *a* che viene copiato nel filo di uscita selezionato:

Figura 5.13: *schema logico di un demultiplexer a 1 ingresso di controllo*

Il funzionamento del demultiplexer consiste nell'instradare il valore di un ingresso sull'uscita specificata dalle variabili di comando, mentre tutte le altre uscite, rimangono a 0. L'ingresso viene instradato sull'uscita X_i se e solo se il valore delle variabili di comando, rappresenta la codifica binaria dell'intero i

Un demultiplexer è un circuito logico la cui principale funzione è inversa a quella del Multiplexer. **Formalmente il demultiplexer è una rete logica con 1 ingresso di dato, n segnali di controllo e 2^n uscite: l'uscita contrassegnata dall'indice pari alla configurazione dei segnali di controllo riceve l'ingresso mentre le altre non sono abilitate.**

Quindi il demultiplexer realizza la funzione di smistare un singolo input in una delle n possibili uscite

Infatti mentre il MULTIPLEXER, abbiamo visto che tra tanti ingressi ne seleziona uno solo in uscita, il DEMULTIPLEXER fa esattamente il contrario, nel senso che abbiamo un solo ingresso *a* e un selezionatore *s* che decide su quali delle tante uscite viene copiato l'ingresso *a*.

Il DEMULTIPLEXER è utile quando, avendo un dato, dobbiamo decidere in quale linea mandarlo in uscita a differenza del MULTIPLEXER, dove avendo tanti dati o fili in ingresso, si deve decidere quali di questi dati devono essere copiati in uscita.

Il decodificatore DEMULTIPLEXER è un circuito molto importante e diffuso e comunemente viene impiegato per visualizzare le cifre ad esempio negli orologi digitali:

Figura 5.14: *esempio di visualizzazione in un display a 7 segmenti*

La figura 5.14 è una generica rappresentazione di cifre a sette segmenti.

Le forme di visualizzazione come quella in figura, sono in realtà una forma di decodifica in quanto possiamo pensarle come a tante uscite di un circuito che accende delle luci in maniera tale da riconoscere il carattere comandato in ingresso.

Supponendo di voler rappresentare delle cifre con un indicatore visivo a 7 segmenti dove ogni segmento è un led che può essere acceso da un segnale digitale, ci saranno allora *n* ingressi a seconda di quante cifre si vuole rappresentare: quindi con ad esempio 4 ingressi si potranno rappresentare 16 cifre, con 2 ingressi, quattro cifre e in generale con *n* ingressi si potranno rappresentare 2^n-1 cifre.

Consideriamo un circuito di quattro ingressi dati da s_0,s_1, s_2 ed s_3, e sette uscite date dai led che verranno attivati a seconda del numero inviato in ingresso, ottenendo così una rappresentazione circuitale dei numeri esadecimali:

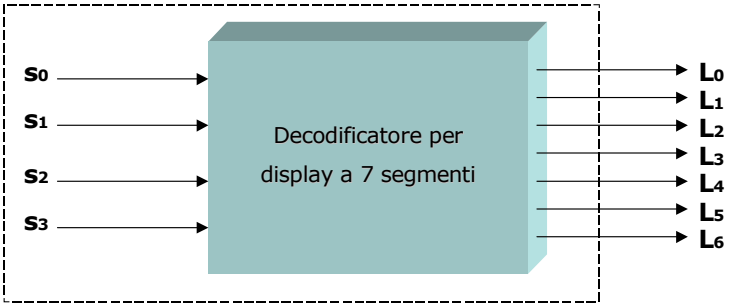

Figura 5.15: *schema logico di un decodificatore per visualizzatore a 7 segmenti*

La figura 5.15 è un decoder a quattro ingressi e sette fili di uscita.

Si è detto che con quattro valori in entrata, riusciamo a dare una rappresentazione esadecimale dei numeri.
Si considerano sette uscite perché si vuole realizzare una visualizzazione a sette segmenti dei numeri esadecimali secondo quanto indicato dalla seguente figura:

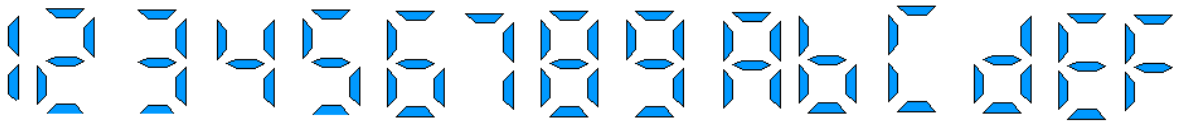

Figura 5.16: *esempio di visualizzazione dei numeri esadecimali in un display a 7 segmenti con quattro ingressi*

Nella visualizzazione, ciascun numero è formato da sette segmenti ognuno dei quali è un led che è acceso da un segnale digitale. Pertanto il circuito della figura 4.15 riceve in ingresso un numero esadecimale inviato dai quattro ingressi, e genera l'appropriata uscita selezionando i segmenti che devono essere accesi per visualizzare sul display il numero esadecimale inviato in ingresso.
I valori di uscita L_i varieranno allora da 0 a 6 secondo l'ordine indicato in figura 5.17 e il circuito farà in modo tale che qualunque numero da 0 a 15 venga imputato dai quattro bit entranti, usciranno sette fili che si accendono o si spengono a seconda del comando dato in ingresso per mostrare sul display il numero esadecimale.
Avendo associato dunque a ciascun led il valore di uscita come da figura:

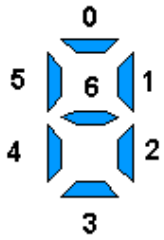

Figura 5.17*: ordine di visualizzazione delle uscite in un display a 7 segmenti*

La tabella di verità del decoder esadecimale a 7 segmenti, sarà costituita da 16 righe e 16 colonne che qui non riportiamo completamente:

s_0	S_1	S_2	S_3	L_0	L_1	L_2	L_3	L_4	L_5	L_6	
0	0	0	0	1	1	1	1	1	1	0	
0	0	0	1	0	1	1	0	0	0	0	
0	0	1	0	1	1	0	1	1	0	1	
0	0	1	1	1	1	1	1	0	0	1	
0	1	0	0	0	1	1	0	0	1	1	
0	1	0	1	1	0	1	1	0	1	1	
0	1	1	0	1	0	1	1	1	1	1	

Ognuna delle variabili in uscita si accenderà o si spegnerà a seconda del comando in entrata.

Supponendo allora che l'ordine dei led per la visualizzazione sia dato dalla figura 5.17, per visualizzare il numero 0 ad esempio, dovranno accendersi tutti i led tranne il led 6 corrisponde all'uscita L_6, per visualizzare invece il numero 1 le uscite attivate saranno L_1 ed L_2, per il numero 2, i segmenti selezionati

dall'ingresso saranno rispettivamente L_0, L_1, L_6, L_4 ed L_3. Lo stesso ragionamento vale per le restanti cifre che possiamo verificare anche osservando la tabella.
Basta dunque guardare i vari led con cui vengono visualizzati i numeri e assegnare di conseguenza il valore 1 all'uscita L_i che deve essere accesa quando va a rappresentare il numero esadecimale comandato dalla variabile in ingresso da 4 bit s_i. Per ognuna delle uscite, ci saranno allora sette sintesi da fare perché sette sono le uscite date dai led e ciascuna di queste sintesi piloterà un segmento.

5.2.5 Sottrattore

Un sottrattore è una macchina logica in grado di realizzare la sottrazione tra due numeri binari.
Per realizzare questa rete logica si impiega una rappresentazione in complemento a due degli interi con segno, poiché questa rappresentazione ha il vantaggio di trattare la sottrazione e la somma come la stessa operazione.
Infatti la differenza tra due numeri a e b può essere scritta indifferentemente così:

$$a - b = a + (-b)$$

in questo caso bisogna individuare un sottocircuito relativo al segno, che comanda al circuito complessivo se comportarsi come sottrattore o come sommatore. Quindi dati due numeri binari *a* e *b*, il sottocircuito deve decidere se attuare la somma o la sottrazione tra i due numeri.
Il circuito che si sta per descrivere non si discosta molto dal circuito sommatore realizzato nel paragrafo 4.10 del capitolo precedente. L'unica differenza è che per il numero (binario) *b*, potendo questo assumere segno positivo o negativo, dovrà essere inserito per esso un circuito in più, in modo tale che prima di fare entrare *b* nel sommatore per sommarlo algebricamente al numero (binario) *a*, inverta il segno del bit *b* (trasformandolo in numero negativo) qualora si decida di sottrarlo al numero *a*.
Il sottrattore allora si comporterà in maniera tale che il bit *b*, prima di entrare nel sommatore, nel caso lo si voglia sottrarre ad *a* , invertirà il segno del numero entrante *b*.
La rete logica del sottrattore sarà impostata nel modo seguente:

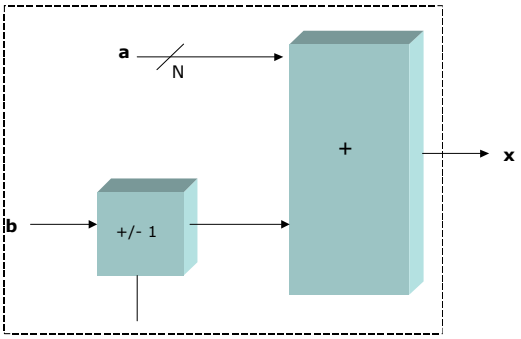

Figura 5.18: *schema logico di un sottrattore*

Al generico sommatore già realizzato nel capitolo precedente, viene anteposto un circuito che ha il compito di invertire il segno di *b* ogni volta che si vuole sottrarre ad *a* il numero *b*.

Ci sarà dunque il numero *a* con n fili di ingresso, che entra direttamente nel sommatore, mentre il numero *b*, prima di essere sommato algebricamente ad al bit *a*, entrerà in un circuito che gli attribuisce il segno più o il segno meno, facendo così uscire o *b* o *–b*, a seconda che lo si voglia sommare o sottrarre ad *a*. Dopo questo passaggio il circuito esegue la somma algebrica tra i due numeri *a* e *b*.

Operativamente il sottocircuito che inverte il segno funzionerà secondo quanto indicato dalla seguente figura:

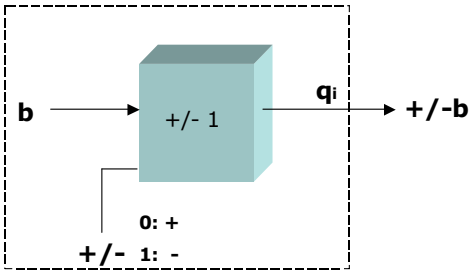

Figura 5.19: *schema logico dell'invertitore di segno o moltiplicatore per 1*

Con riferimento alla figura 5.19, il comportamento del bit del segno, sarà tale che quando questo vale 0, allora il segno di *b* sarà positivo, quando invece il bit del segno vale 1, il segno di *b* sarà negativo. Per cui ogni volta che si vorrà sottrarre *b* al numero binario *a*, il segno del bit *b* verrà invertito quando entrerà nel sottocircuito della figura 5.19 e il bit del segno varrà 1. La tabella di verità relativa la bit del segno sarà allora:

segno	Valore del bit
+	0
-	1

se dunque il bit del segno vale 1, allora il circuito della figura 5.18 si comporta come un sottrattore: ossia quando entra il i-esimo bit b_i e il filo del segno da un comando pari ad uno, il sottocircuito della figura 5.19 restituisce in uscita un valore q_i=- b_i. Se invece il bit del segno è a zero, allora il circuito copia in uscita l'ingresso originario per cui $q_i=b_i$.
La tabella di verità che descrive il comportamento della figura 5.19 sarà:

Tabella 5.4: *Tabella di verità dell'operatore XOR*

b_i	+/-	Q_0
0	0	0
0	1	1
1	0	1
1	1	0

Lo schema circuitale per q=1 sarà una porta logica XOR o OR esclusivo[1]:

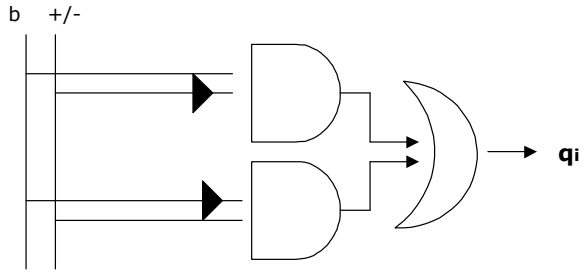

Figura 5.20: *circuito che sintetizza l'inversione del segno per ogni ingresso* b_i

XOR sta per OR esclusivo, l'operatore XOR opera solamente tra due bit. Il sottocircuito così individuato dalla figura 5.20 è un invertitore su comando, nel senso che se il filo del segno è pari a 1, il bit entrante viene cambiato di segno, se invece il filo è pari a 0, il bit in ingresso viene sommato ad un valore nullo e quindi viene copiato in uscita così com'è.

[1] L'operatore **XOR** (detto anche **OR esclusivo**) restituisce 1 (*vero*) se *solo uno* degli elementi è 1 e tutti gli altri 0.
0 XOR 0 = 0
0 XOR 1 = 1
1 XOR 0 = 1
1 XOR 1 = 0

Avendo individuato il circuito che inverte opportunamente a comando il segno dei bit *b*, resta solo da inserire il comando che somma 1 al bit invertito di segno; si è infatti visto che per rappresentare in completo a due un numero negativo, prima si invertono i bit del numero preso in modulo, e poi si aggiunge al suo complemento[2].

Ciò significa che quando il bit del segno vale 1, il circuito oltre ad invertire il segno del bit *b*, somma ad esso il valore stesso del bit del segno, mentre quando il filo del segno assume il valore 0, allora il circuito, oltre a lasciare invariato il segno di b, sommerà a quest'ultimo il valore 0 assunto dal bit del segno, lasciando così invariato il numero binario *b*.

Dopo questo primo passaggio, l'uscita restituita dal sottocircuito che a seconda del caso inverte il segno di b, sarà algebricamente sommata al numero binario *a*.

Il circuito finale sarà dunque dato dalla figura seguente:

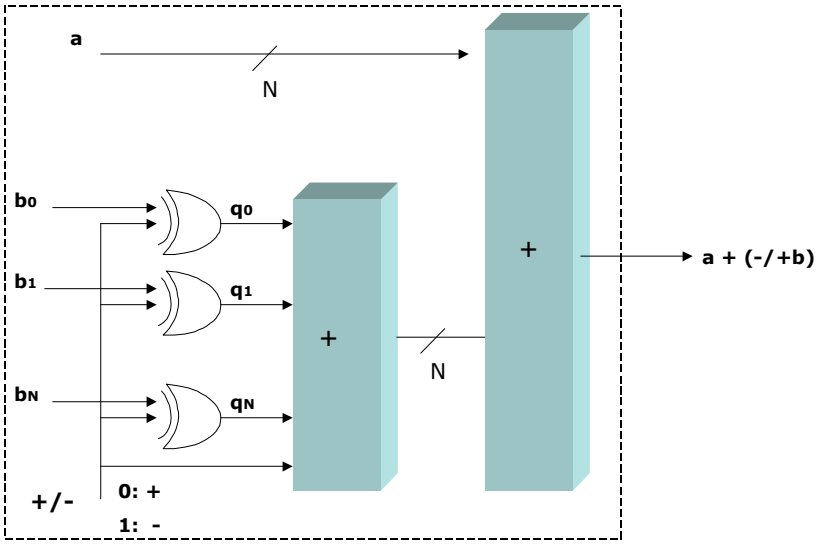

Figura 5.21: *schema finale di un sottattore con invertitore di segno*

Osserviamo che il primo bit d'ingresso a_i entra direttamente cosi com'è, mentre per il bit b_i ci sono due stadi: i vari b_i entrano in tanti piccoli sottocircuiti uguali a quelli individuati dalla figura 5.19 che daranno come uscite i q_i individuati dalla tabella XOR, il filo con il segno +/- decide se fare la somma o la sottrazione. Il sottocircuito sarà a sua volta collegato ad un altro sommatore in cui entra il bit del segno che assumerà i valori (o 0 o 1) che dovranno essere sommati ai vari q_i.

[2] Vedi cap. 2 paragrafo 2.8.3

Questo è un esempio di circuito che usa come componenti fondamentali altri circuiti più semplificati e che già conosciamo: infatti la figura 5.21 usa due sommatori, nonché il circuito XOR della figura 5.20 che si è costruito appositamente per invertire il segno del bit quando serve.