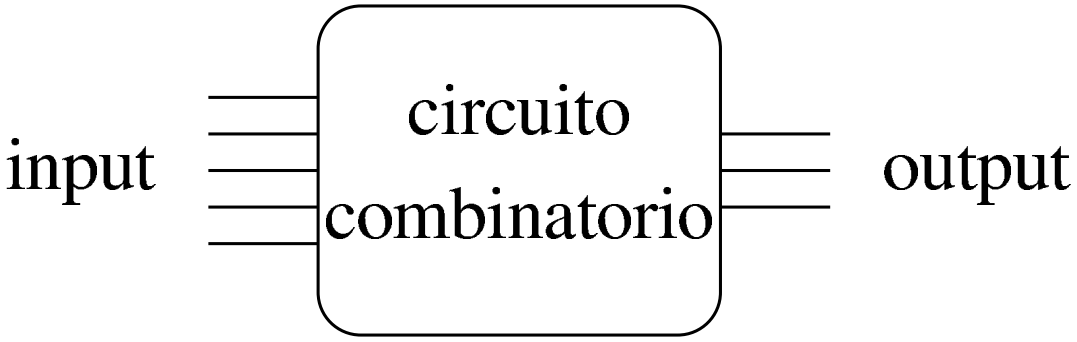
Un ***circuito combinatorio***, ovvero una ***rete combinatoria***, è un sistema di porte logiche, connesse opportunamente tra loro, organizzato con un insieme di ingressi e un insieme di uscite, nel quale i valori logici delle uscite sono determinati direttamente e univocamente dai

valori logici presenti negli ingressi. Il circuito combinatorio si può rappresentare, complessivamente, come una scatola composta da ingressi e da uscite, con una tabella di verità che stabilisce i valori delle uscite in base ai valori degli ingressi.

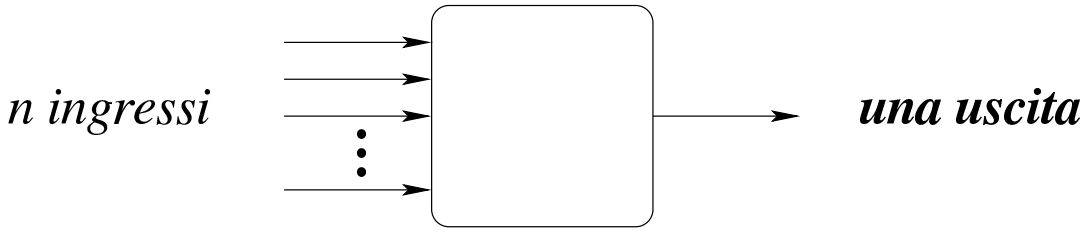


Quello raffigurato sopra è un esempio di circuito combinatorio con cinque ingressi e tre uscite, ma la proporzione tra quantità di ingressi e quantità di uscite dipende solo dalla funzione che deve realizzare tale circuito, ovvero dallo scopo che con questo ci si prefigge di raggiungere.

Si osservi che un circuito combinatorio, per essere tale, non deve essere influenzato dalla variabile tempo e nemmeno dalla variabile casuale dovuta all'accensione del circuito; pertanto, su tali circuiti non si considera il problema del tempo di propagazione, necessario a far sì che le uscite raggiungano i valori previsti in base ai valori pervenuti in ingresso.

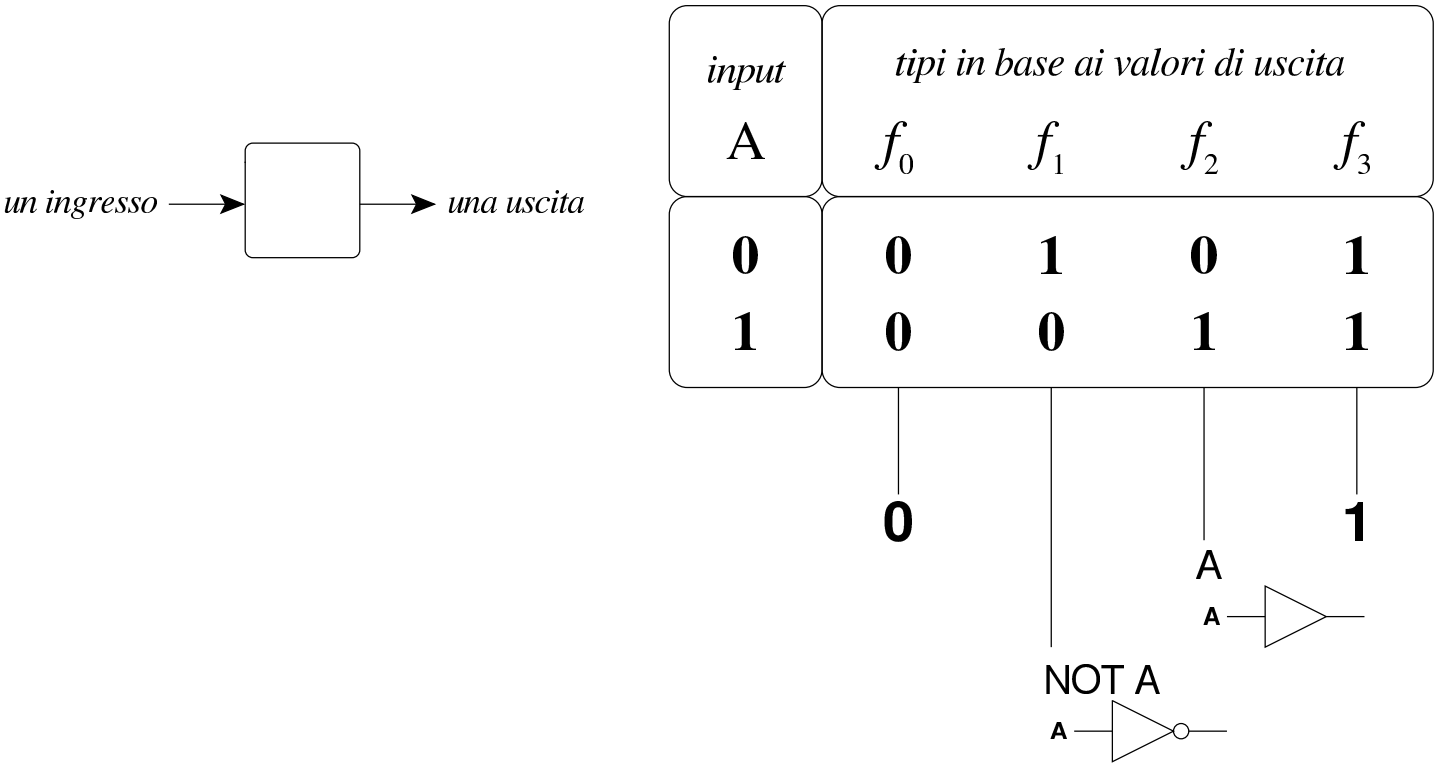
I circuiti combinatori più semplici sono quelli che dispongono di una sola uscita e realizzano quindi delle porte logiche, più o meno complesse.

Figura u98.2. Circuiti combinatori con una sola uscita.



Per comprendere la questione, si può osservare che un circuito composto da un solo ingresso e da una sola uscita può essere di quattro tipi differenti, come evidenziato dallo schema successivo.

Schema u98.3. Tabella di verità per quattro funzioni di una variabile.



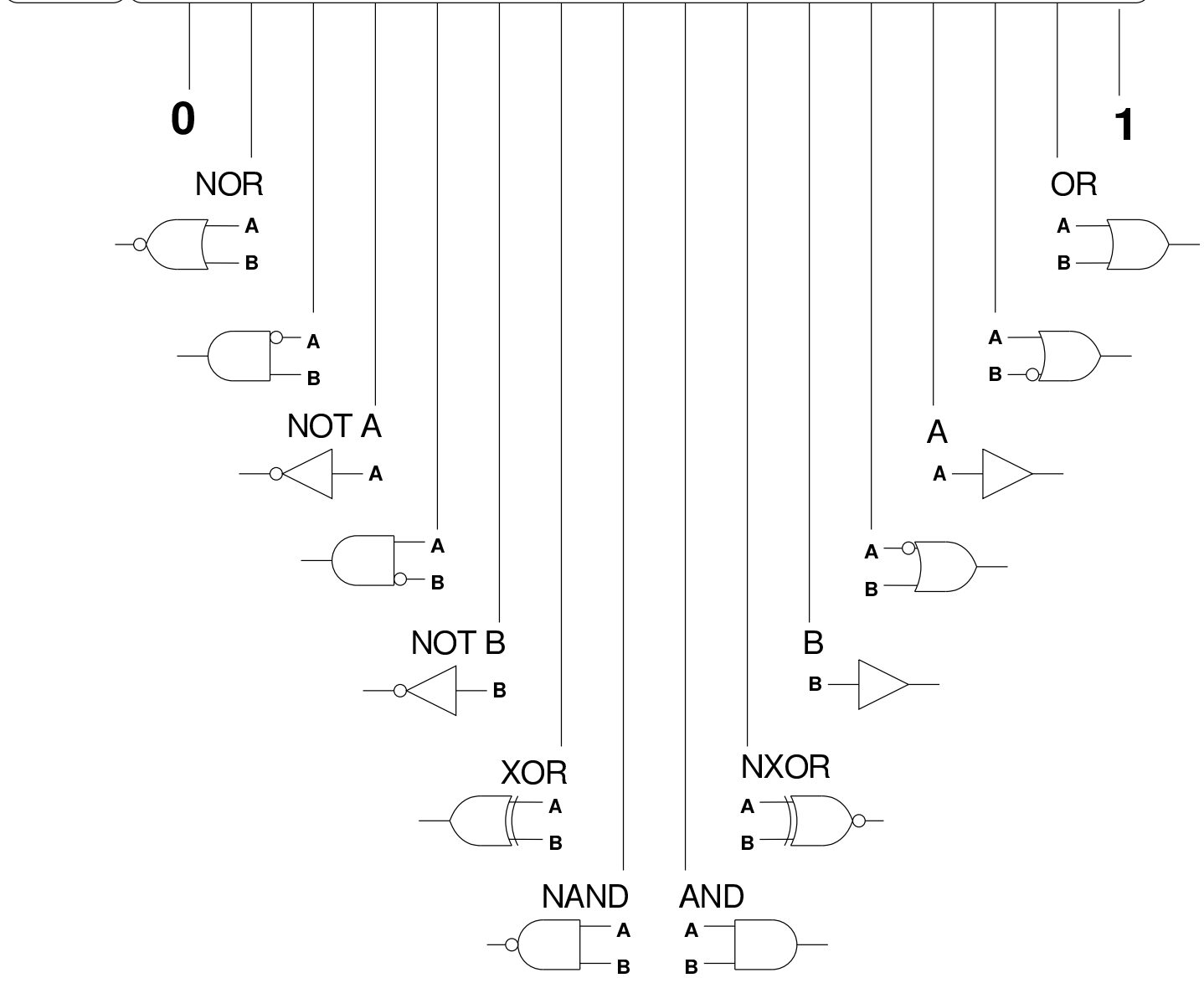
Come si vede dalle annotazioni contenute nello schema, il circuito corrispondente alla funzione f_1, coincide con il circuito invertente (ovvero NOT), mentre quello corrispondente alla funzione f_2 coincide con il circuito non-invertente.

In un circuito combinatorio con **due ingressi** e un'uscita, ci sono

16 funzioni possibili, come si vede nello schema successivo, dove si evidenziano in particolare le corrispondenze con le porte logiche comuni, indicate assieme al loro nome standard.

Schema u98.4. Tabella di verità per sedici funzioni di due variabili.

input		*tipi in base ai valori di uscita*															
A	B	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1



Decodificatore

«

Il decodificatore (*decoder*) è un circuito combinatorio che attiva una sola uscita, selezionandola in base alla combinazione di valori presenti negli ingressi, tenendo conto per ogni combinazione delle variabili di ingresso deve esserci un'uscita differente da attivare. Pertanto, per n ingressi ci sono 2^n uscite. Inoltre, solitamente, tale circuito combinatore è provvisto anche di un ingresso di controllo ulteriore, disattivando il quale si fa in modo che nessuna uscita risulti attiva, indipendentemente dagli altri valori in ingresso.

Figura u98.5. Schema a blocchi di un decodificatore a quattro uscite. I due ingressi di selezione sono contrassegnati dalle sigle a_0 e a_1; l'ingresso di abilitazione è indicato dalla sigla ***en*** (*enable*); le uscite sono indicate con le sigle da y_0 a y_3. Lo schema di destra è reso in una forma più compatta, dove gli ingressi di selezione risultano raccolti assieme.

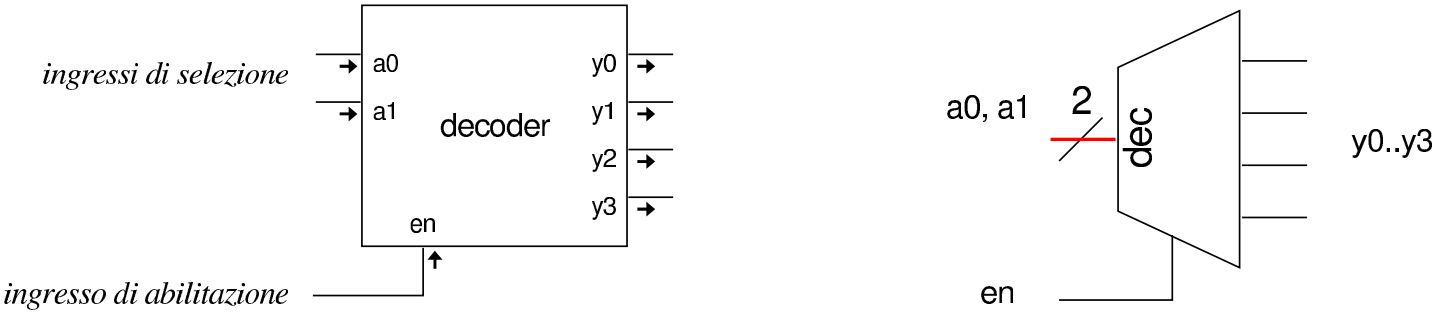
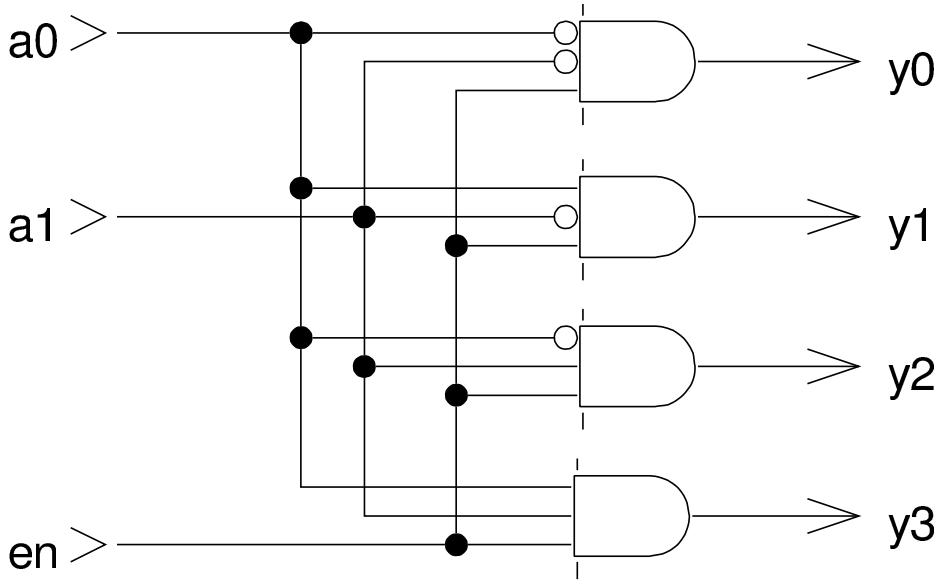


Tabella u98.6. Tabella di verità per un decodificatore a quattro uscite, da y_0 a y_3; due ingressi di selezione, a_0 e a_1; un ingresso di abilitazione en.

en	a_1	a_0	y_3	y_2	y_1	y_0
0	x	x	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

Figura u98.7. Esempio di realizzazione di un decodificatore a quattro uscite: i due ingressi di selezione sono contrassegnati dalle sigle a_0 e a_1; l'ingresso di abilitazione è indicato dalla sigla en; le uscite sono indicate con le sigle da y_0 a y_3.

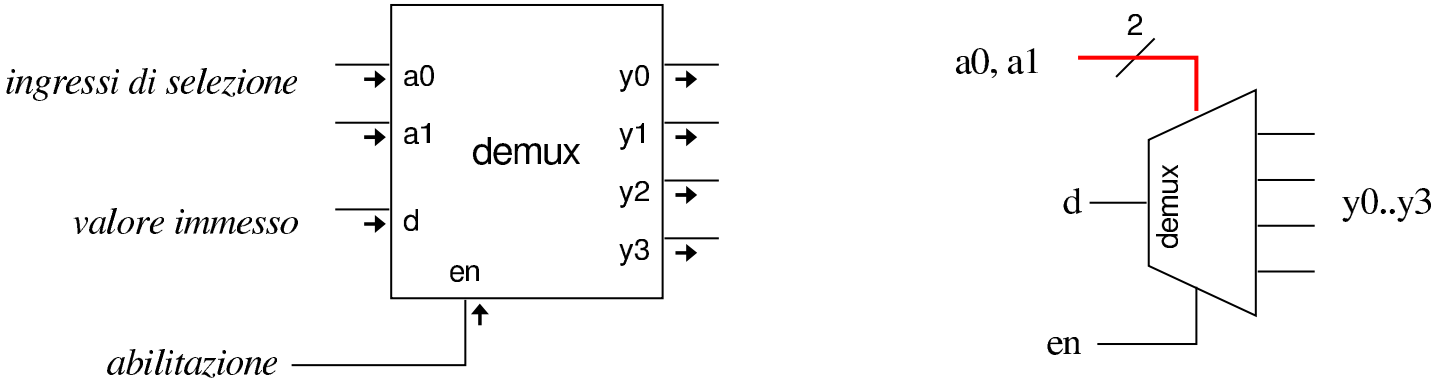


Demultiplatore

«

Il ***demultiplatore*** (*demultiplexer*), spesso abbreviato con la sigla ***demux***, svolge un lavoro simile al decodificatore, ma in qualità di commutatore del valore contenuto in un ingresso aggiuntivo.

Figura u98.8. Schema a blocchi di un demultiplatore a quattro uscite. I due ingressi di selezione sono contrassegnati dalle sigle a_0 e a_1; l'ingresso dati corrisponde alla variabile d; l'ingresso di abilitazione è indicato dalla sigla ***en***; le uscite sono indicate con le sigle da y_0 a y_3. Nel disegno di destra si vede una versione più compatta.

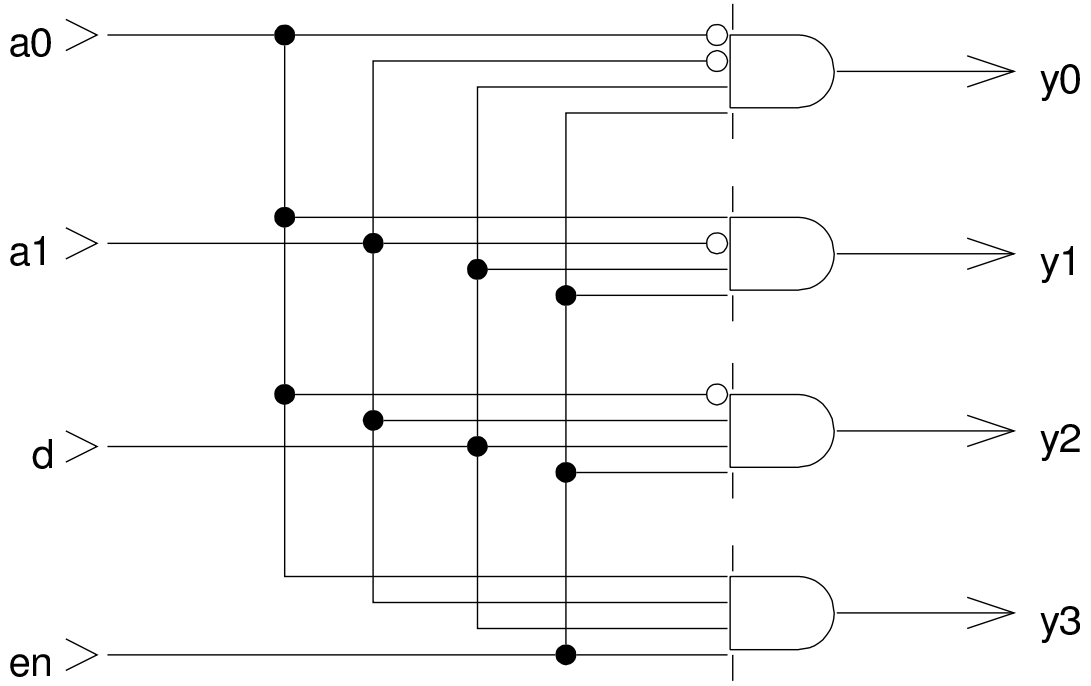


In questo caso, a differenza del decodificatore, l'uscita y_n selezionata riporta lo stesso valore dell'ingresso d.

Tabella u98.9. Tabella di verità per un demultiplatore a quattro uscite, da y_0 a y_3; due ingressi di selezione, a_0 e a_1; un ingresso dati d; un ingresso di abilitazione en.

en	a_1	a_0	d	y_3	y_2	y_1	y_0
0	x	x	d	0	0	0	0
1	0	0	d	0	0	0	d
1	0	1	d	0	0	d	0
1	1	0	d	0	d	0	0
1	1	1	d	d	0	0	0

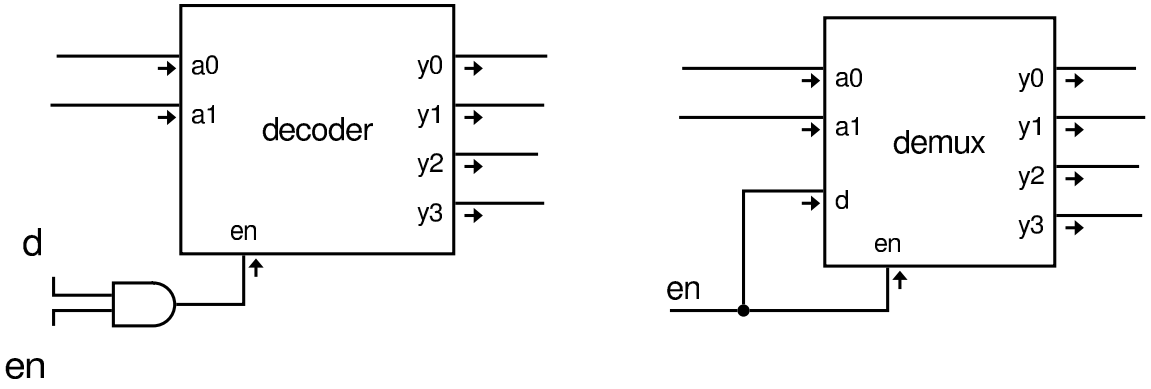
Figura u98.10. Esempio di realizzazione di un demultiplatore a quattro uscite. I due ingressi di selezione sono contrassegnati dalle sigle a_0 e a_1; il dato da commutare nell'uscita selezionata viene letto dall'ingresso d; l'ingresso di abilitazione è indicato dalla sigla en; le uscite sono indicate con le sigle da y_0 a y_3.



Il demultiplatore può essere ottenuto facilmente da un deco-

dificatore munito di ingresso di abilitazione; in modo analogo, un demultiplatore può essere ridotto a funzionare come un decodificatore.

Figura u98.11. A sinistra un decodificatore adattato per funzionare come demultiplatore; a destra un demultiplatore adattato per funzionare come decodificatore.



Multiplatore

«

Il ***multiplatore*** (*multiplexer*), spesso abbreviato con ***mux***, è un circuito combinatorio che seleziona il valore di una sola porta di entrata e lo riproduce nell'uscita. La selezione della porta di entrata dipende dalla combinazione di valori contenuta in un insieme di porte di selezione. Per n porte di selezione si può scegliere tra 2^n porte di entrata.

Figura u98.12. Multiplatore a quattro entrate. I due ingressi di selezione sono contrassegnati dalle sigle a_0 e a_1; le entrate hanno le sigle da d_0 a d_3; l'ingresso di abilitazione è indicato dalla sigla ***en*** (*enable*); l'uscita è indicate con la variabile ***y***. Lo schema di destra rappresenta una forma compatta, nella quale le variabili di selezione sono raccolte assieme in una linea multipla.

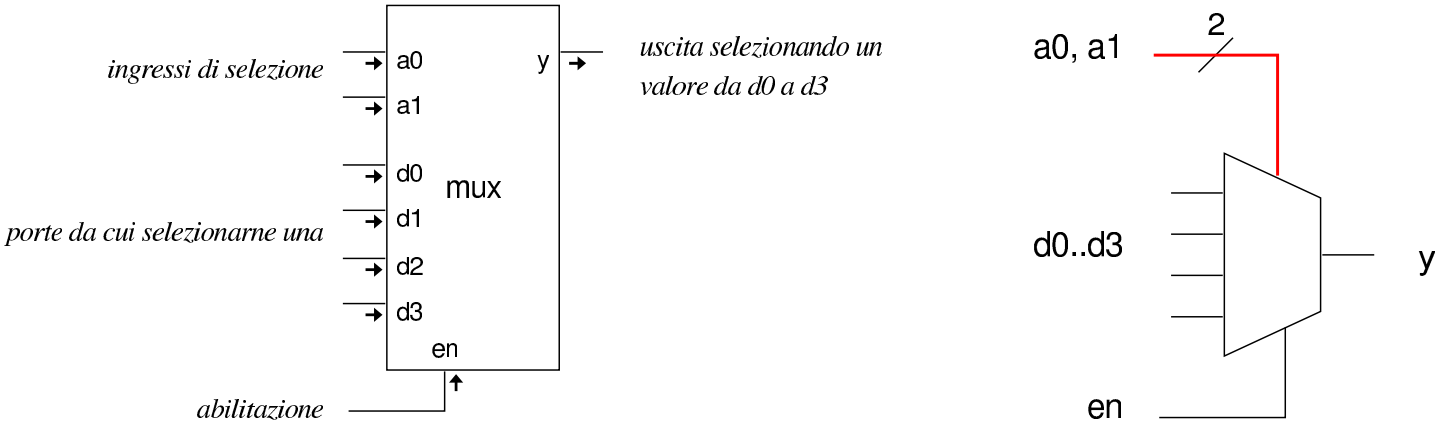
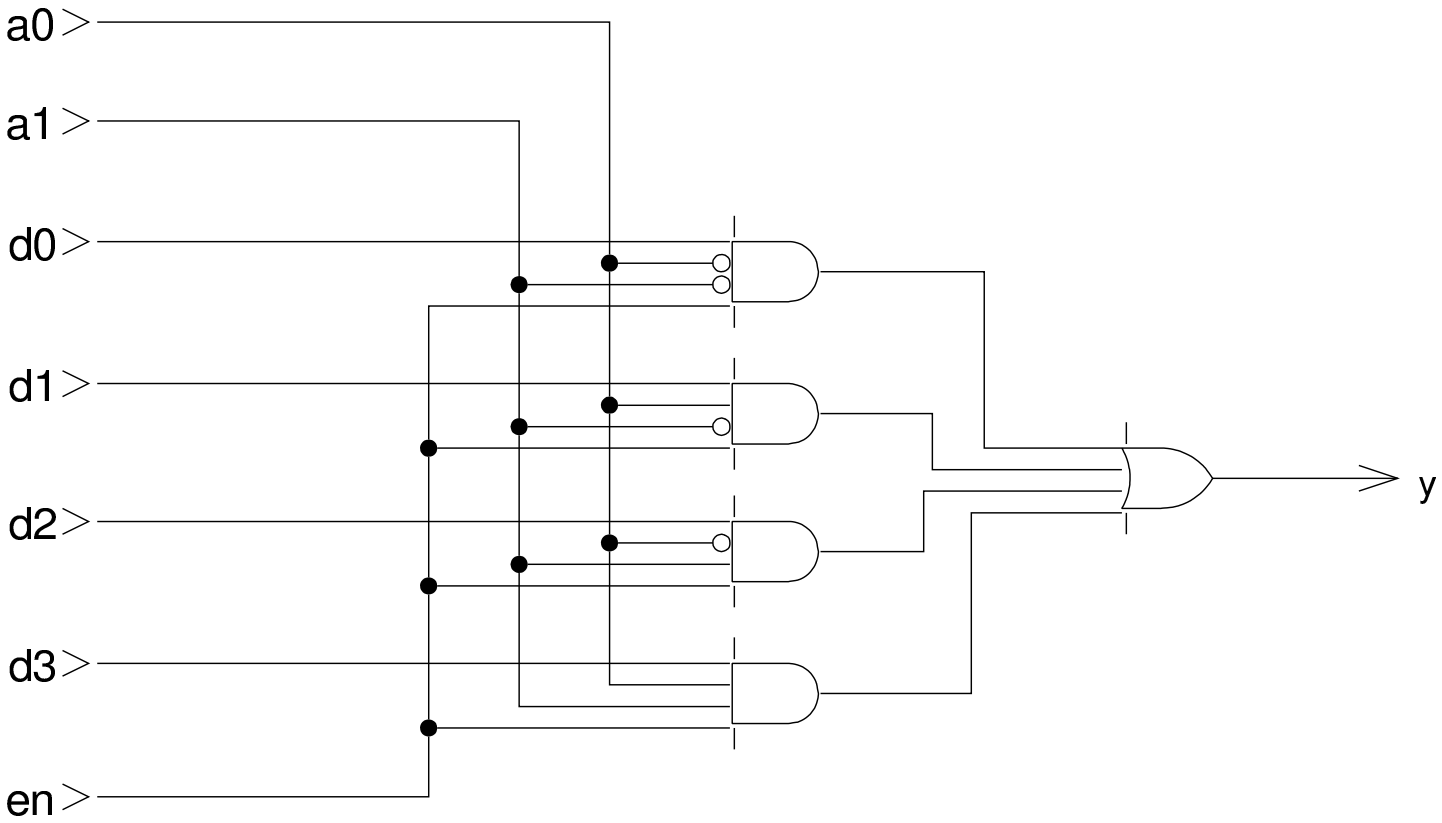


Tabella u98.13. Tabella di verità per un multiplatore a quattro ingressi di dati, da d_0 a d_3; due ingressi di selezione, da a_0 e a_1; un ingresso di abilitazione ***en***; un'uscita ***y***.

en	a_1	a_0	d_3	d_2	d_1	d_0	***y***
0	***x***	***x***	d_3	d_2	d_1	d_0	0
1	0	0	d_3	d_2	d_1	d_0	d_0
1	0	1	d_3	d_2	d_1	d_0	d_1
1	1	0	d_3	d_2	d_1	d_0	d_2
1	1	1	d_3	d_2	d_1	d_0	d_3

Figura u98.14. Esempio di realizzazione di un multiplatore a quattro entrate.



Demultiplatori e multiplatori in parallelo

«

Quando i demultiplatori o i multiplatori vengono usati per commutare dati composti da più bit e quindi trasportati da più linee, servono tanti demultiplatori o multiplatori quanti sono tali bit, mettendo però assieme tutti gli ingressi di selezione e di abilitazione. Tuttavia, nel disegno di un circuito tali linee multiple si annotato in forma compatta. Le figure successive mostrano un demultiplatore e un multiplatore che trattano dati a quattro bit.

Figura u98.15. Demultiplatore con entrata e uscite composte da vettori di quattro valori.

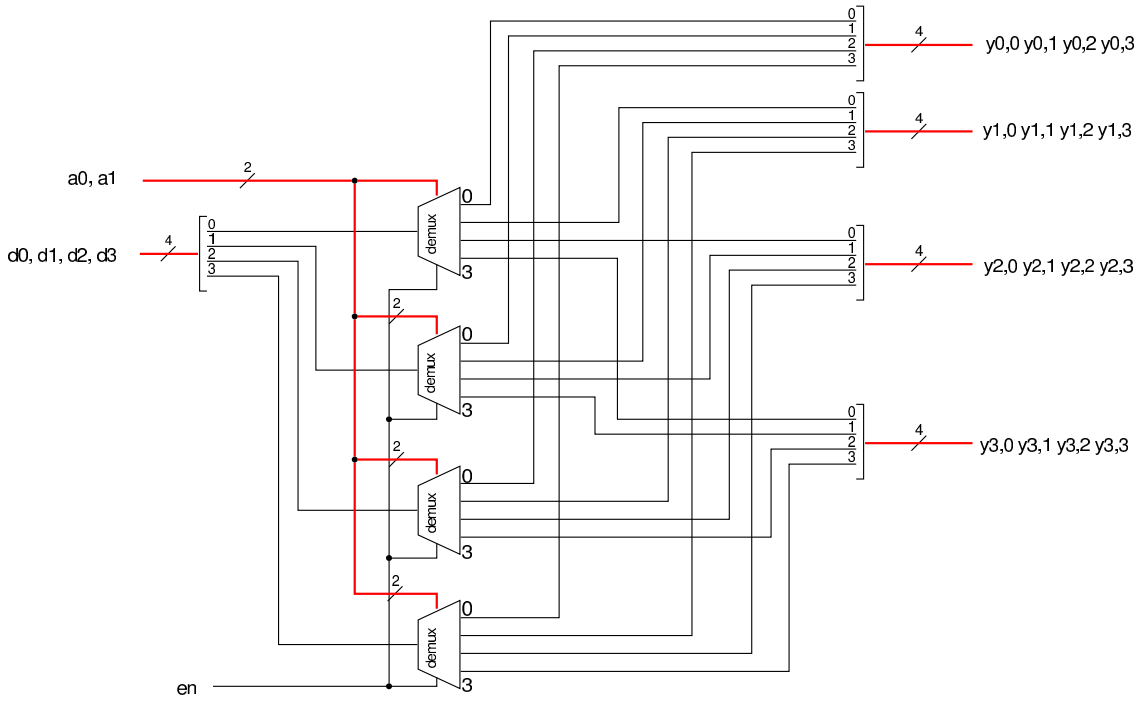
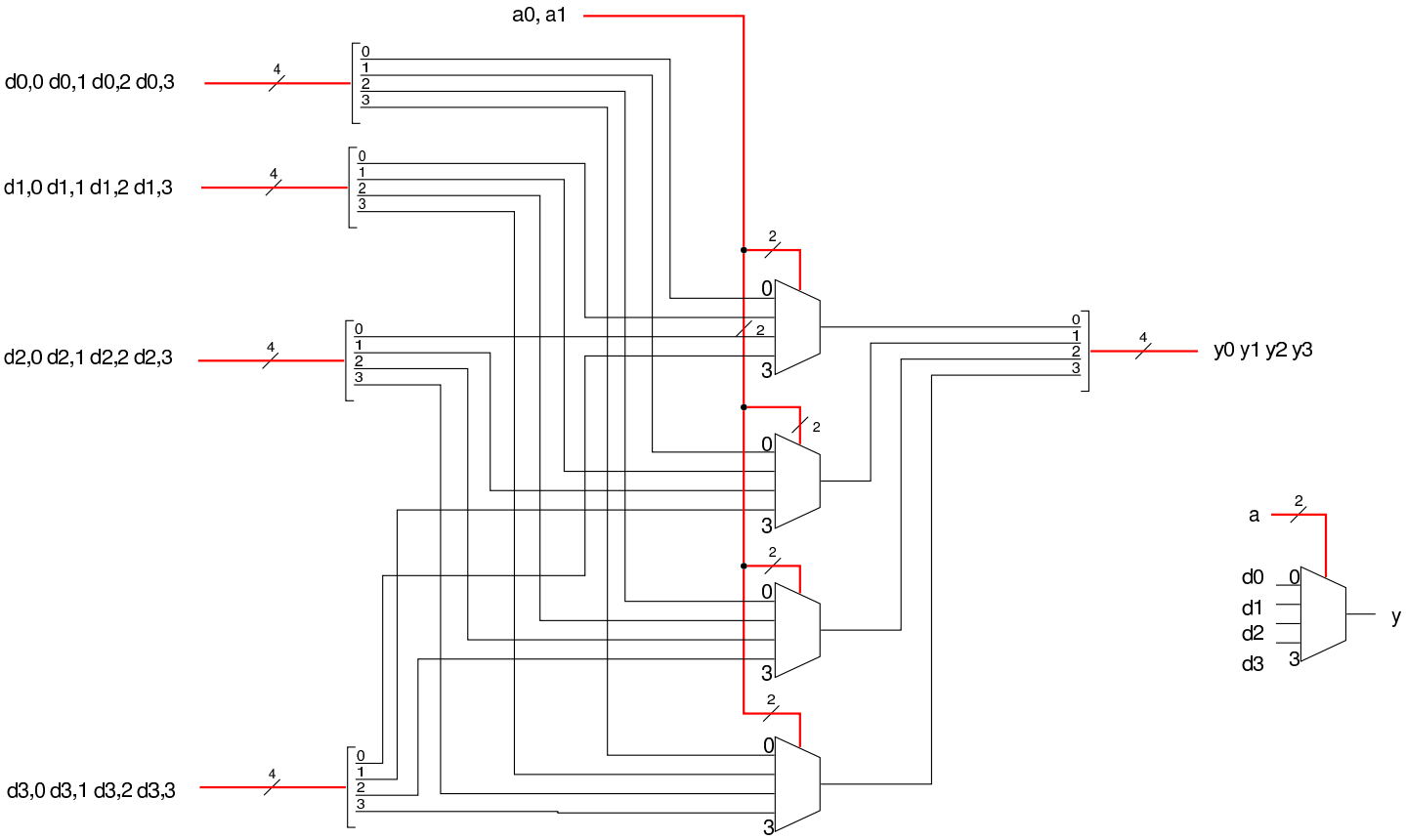


Figura u98.16. Multiplatore con entrate e uscita composte da vettori di quattro valori.



Codificatore binario

Un ***codificatore binario*** (*binary encoder* è un circuito combinatorio nei cui ingressi ci può essere una sola linea attiva e ha lo scopo di tradurre l'ingresso selezionato in un numero binario nelle uscite. In pratica, si tratta generalmente di 2^n ingressi e di n uscite. Vengono mostrati esempi di codificatori con n pari a 1, 2 e 3.

Tabella u98.17. Tabella di verità per un codificatore binario avente due ingressi e una uscita.

w_1	w_0	y_0
0	1	0
1	0	1

Figura u98.18. Il codificatore binario a due ingressi è un circuito inutile, in quanto basta collegare il secondo ingresso all'uscita per ottenere il risultato.

Tabella u98.19. Tabella di verità per un codificatore binario avente quattro ingressi e due uscite.

w_3	w_2	w_1	w_0	y_1	y_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Figura u98.20. Codificatore binario a quattro ingressi e due uscite. Il primo ingresso non viene collegato, perché in quella circostanza l'uscita è comunque pari a zero.

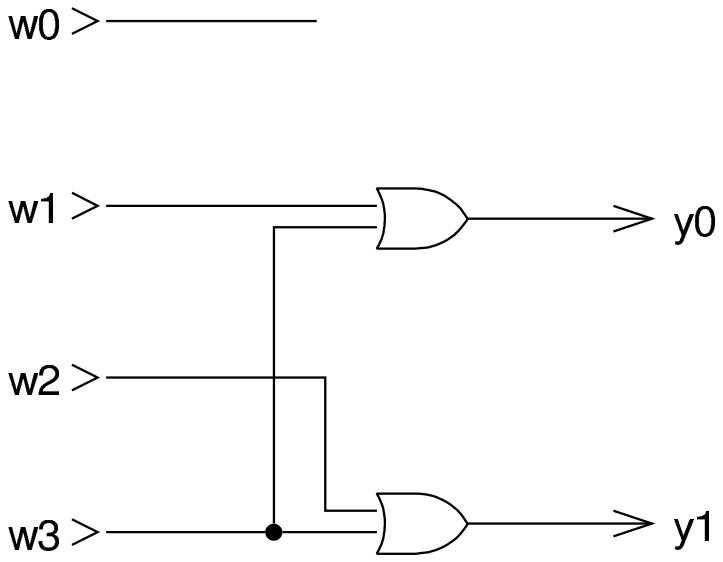
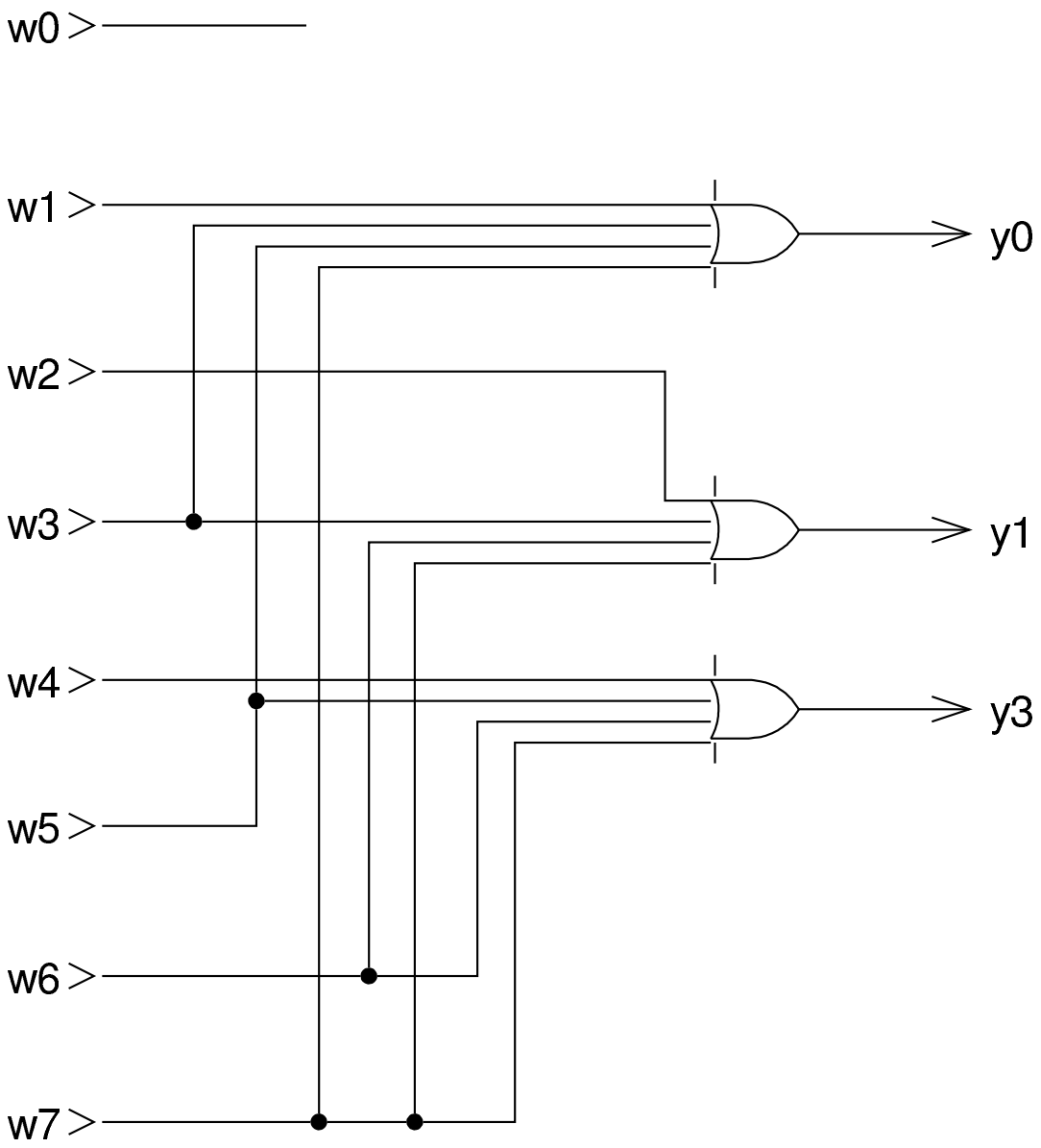


Tabella u98.21. Tabella di verità per un codificatore binario avente otto ingressi e tre uscite.

w_7	w_6	w_5	w_4	w_3	w_2	w_1	w_0	y_2	y_1	y_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Figura u98.22. Codificatore binario a otto ingressi e tre uscite.



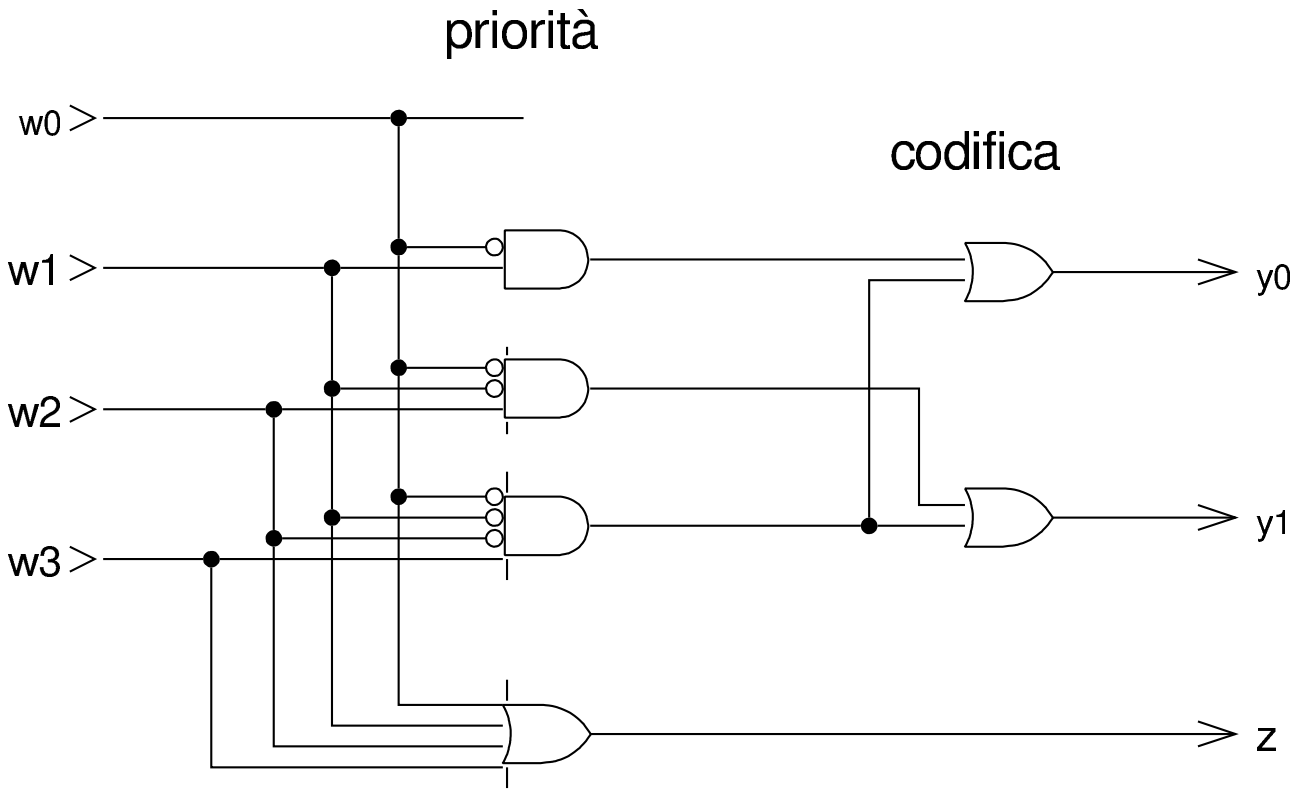
Codificatore di priorità

Il ***codificatore di priorità*** (*priority encoder*) è un codificatore binario che ammette l'attivazione di qualunque ingresso, emettendo il numero corrispondente all'ingresso attivo avente priorità rispetto agli altri. In questo caso è ammissibile anche il fatto che nessun ingresso sia attivo, pertanto, può essere presente un'uscita aggiuntiva che si attiva quando in ingresso c'è almeno un valore attivo.

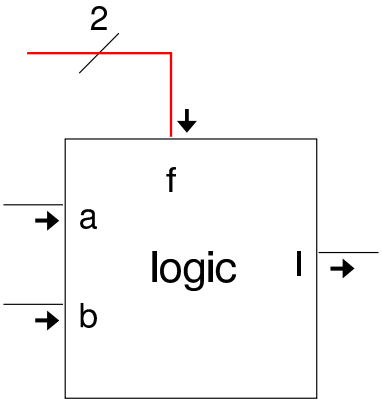
Tabella u98.23. Tabella di verità per un codificatore di priorità avente quattro ingressi. L'ingresso attivo che ha indice più basso ha priorità rispetto agli altri, pertanto si usa la lettera ***x*** per indicare un valore indifferente della variabile corrispondente; l'uscita *z* si attiva se esiste almeno un ingresso attivo e quando l'uscita *z* è a zero, non ha importanza il valore delle altre uscite.

w_3	w_2	w_1	w_0	z	y_1	y_0
0	0	0	0	0	x	x
x	x	x	1	1	0	0
x	x	1	0	1	0	1
x	1	0	0	1	1	0
1	0	0	0	1	1	1

Figura u98.24. Codificatore di priorità a quattro ingressi.

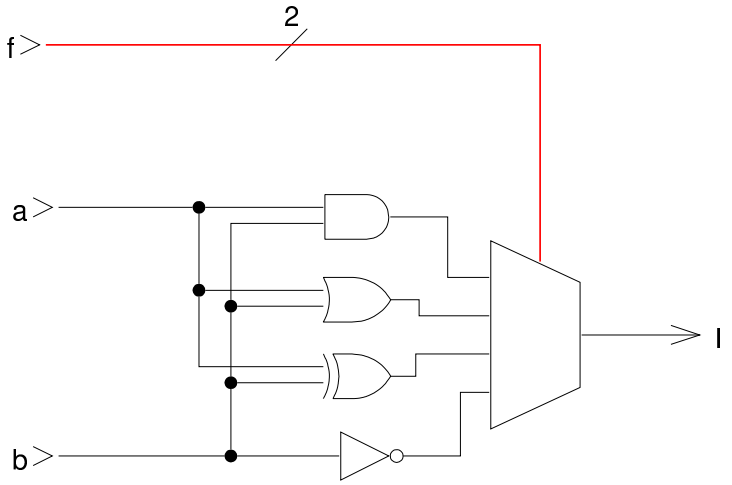


Con l'ausilio di un multiplatore è possibile ottenere facilmente un'unità logica, in grado di eseguire le operazioni logiche comuni, scegliendo quella desiderata attraverso un valore di selezione. Per esempio, si potrebbe realizzare un circuito che svolge il compito schematizzato in questo disegno:



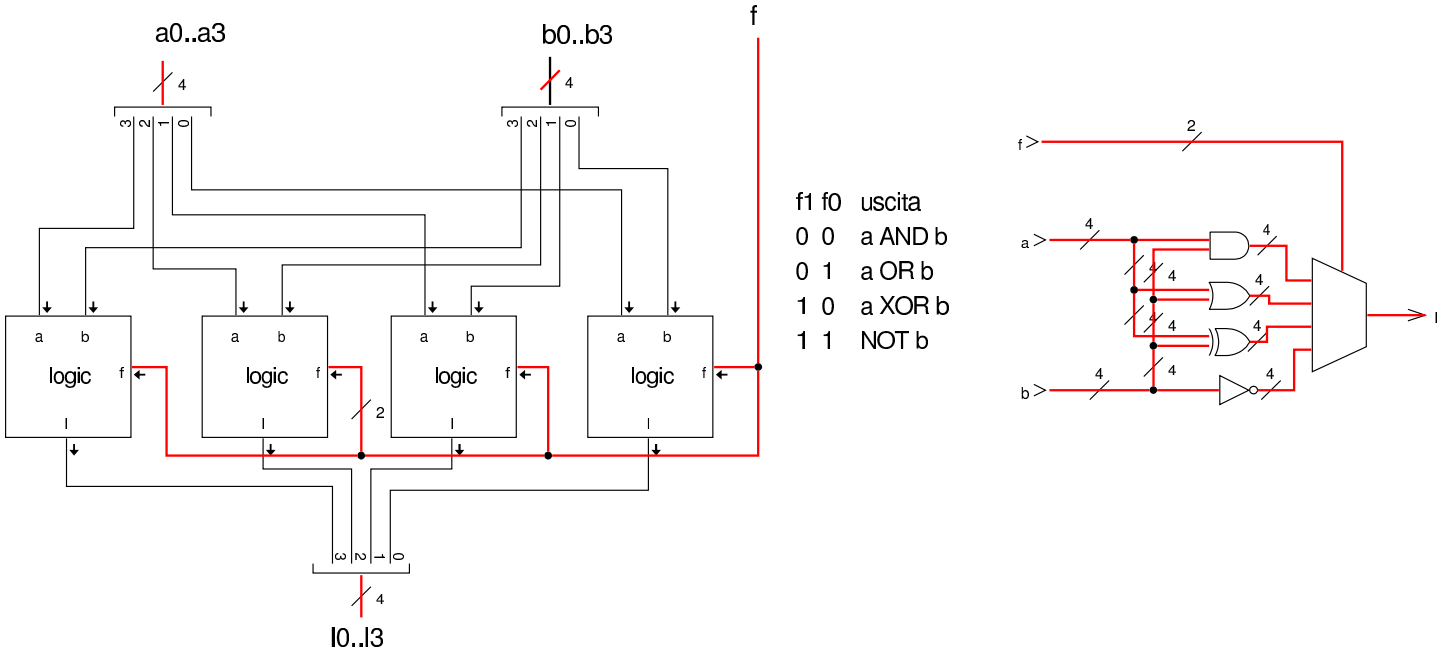
f1	f0	uscita
0	0	a AND b
0	1	a OR b
1	0	a XOR b
1	1	NOT b

Figura u98.26. Esempio di realizzazione di un'unità logica con quattro opzioni.



Le unità logiche di questo tipo sono utili se si applicano a coppie di ingressi che dispongono, ciascuno, di più bit. Per ottenere questo risultato basta abbinare delle unità semplici, a una sola uscita.

Figura u98.27. Le unità logiche possono essere messe in parallelo per operare su interi binari a più cifre. Lo schema di destra è identico dal punto di vista circuitale, ma utilizza un modo compatto per rappresentare la duplicazione delle porte logiche in parallelo.



Scorrimento

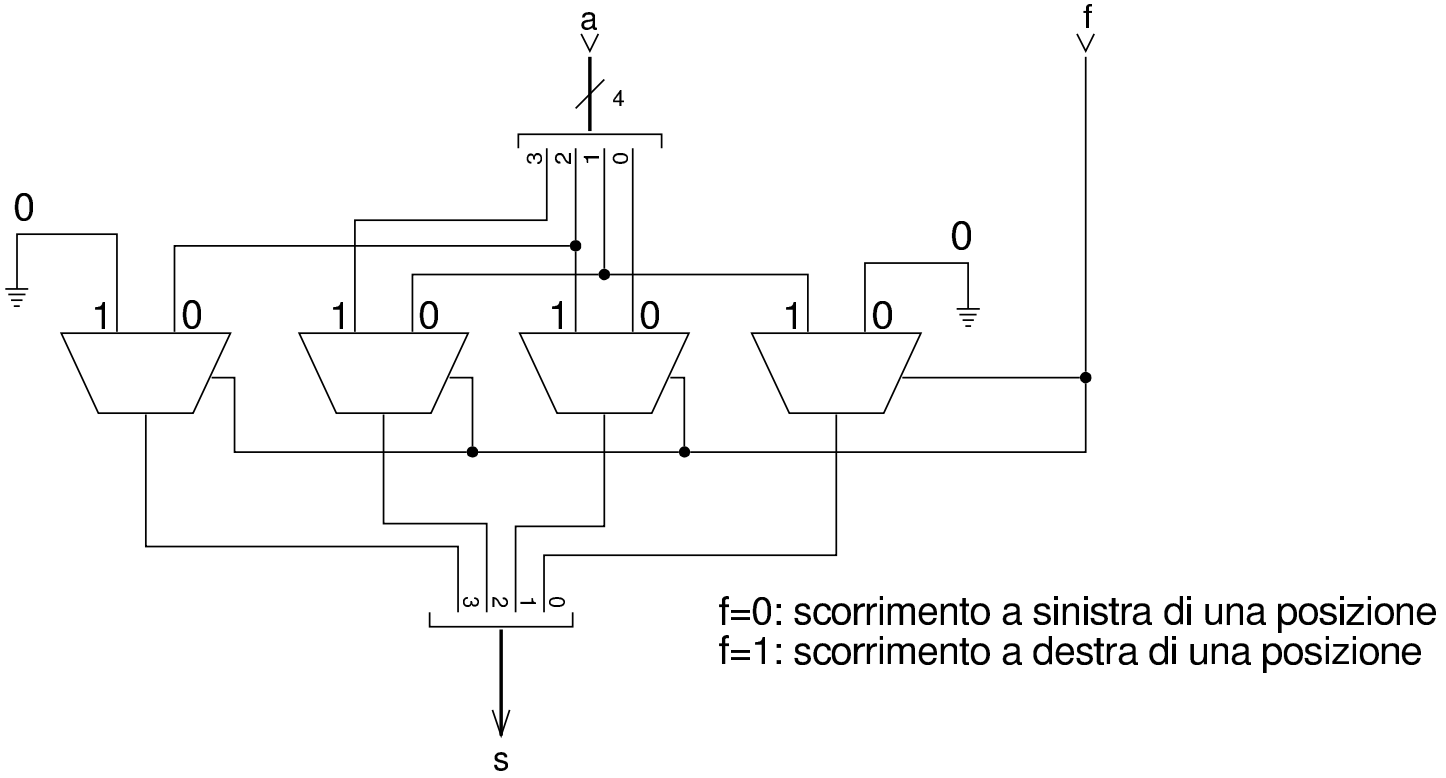
Un tipo di operazione molto importante che si può realizzare con dei circuiti combinatori, è lo scorrimento dei bit. In pratica si ha un gruppo di n ingressi ordinati, da 0 a $n-1$, un gruppo di n uscite, ordinate nello stesso modo; eventualmente ci può essere un riporto in ingresso e uno in uscita. Spesso si considera lo spostamento di una sola unità, perché si possono ottenere spostamenti di più unità ripetendo la stessa operazione ciclicamente.

Il tipo più semplice di scorrimento è quello logico, nel quale lo spostamento a destra fa inserire uno zero nella posizione più significativa, mentre quello a sinistra lo fa entrare dalla parte destra. Nelle

figure successive, per ottenere il valore zero si vede semplicemente un collegamento a massa, la quale si intende essere implicitamente al potenziale corrispondente allo zero.

1	1	1	0	*valore originale*	1	1	1	0	*valore originale*
1	1	0	0	*scorrimento logico a sinistra*	0	1	1	1	*scorrimento logico a destra*

Figura u98.29. Scorrimento logico, a sinistra o a destra, di quattro cifre binarie, con l'uso di multiplatori a due ingressi. Il verso dello scorrimento viene stabilito dal valore fornito nell'ingresso ***f***. Lo scorrimento fa perdere una cifra e fa inserire uno zero dal lato opposto.



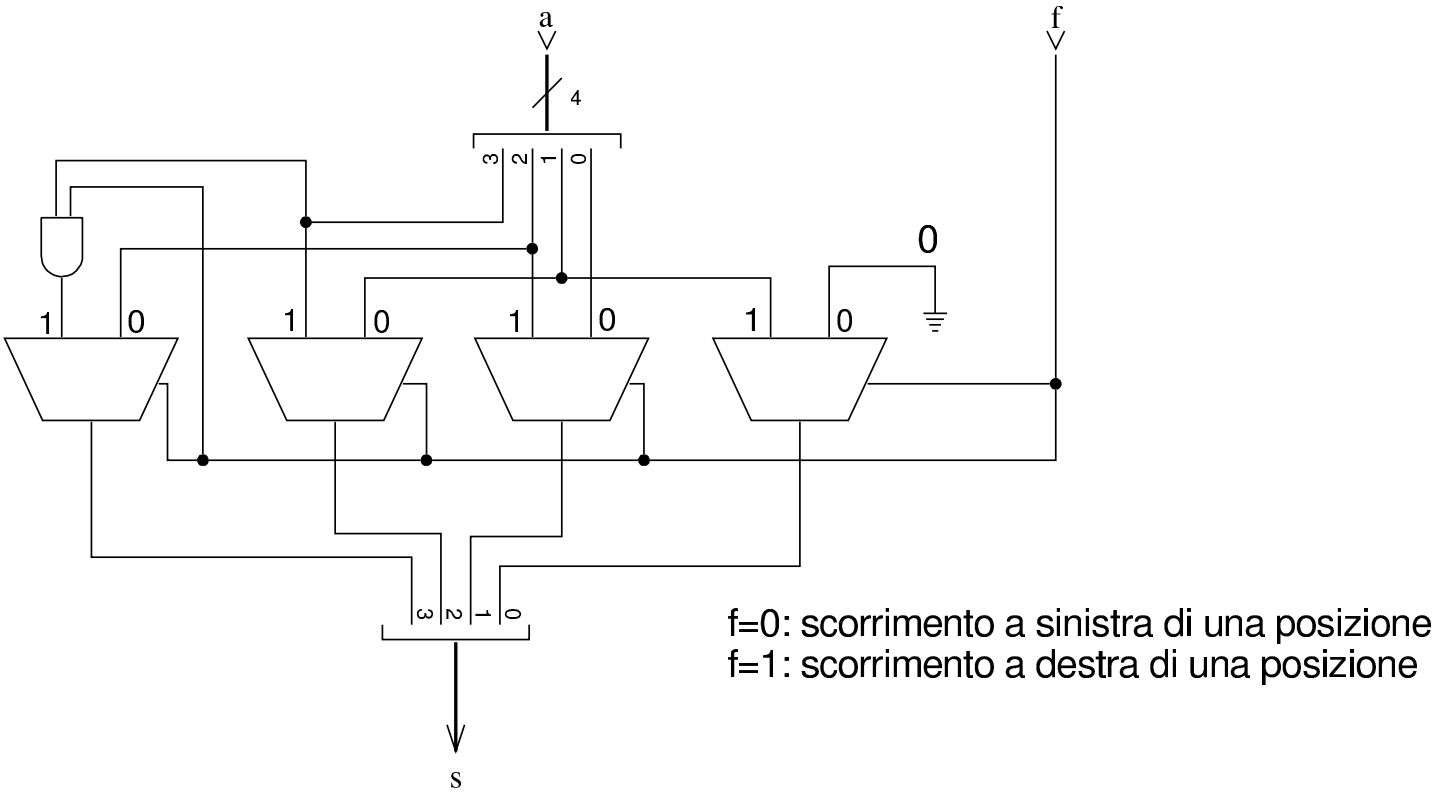
Lo scorrimento aritmetico avviene come lo scorrimento logico, con la differenza che lo scorrimento a destra deve mantenere inalterato il segno. In pratica, nello scorrimento a destra di tipo aritmetico, la cifra che viene immessa nella posizione più significativa, è la copia di quella che era originariamente in quella posizione. Lo scorrimento

aritmetico si chiama così perché corrisponde alla moltiplicazione o alla divisione per due (la base di numerazione); va però osservato che lo scorrimento a sinistra può provocare un'inversione indesiderata di segno.

1	1	1	0	*valore originale*	1	1	1	0	*valore originale*
1	1	0	0	*scorrimento aritmetico a sinistra*	1	1	1	1	*scorrimento aritmetico a destra*
0	1	1	0	*valore originale*	0	1	1	0	*valore originale*
1	1	0	0	*scorrimento aritmetico a sinistra (*)*	0	0	1	1	*scorrimento aritmetico a destra*
1	0	1	0	*valore originale*	0	1	1	0	*valore originale*
0	1	0	0	*scorrimento aritmetico a sinistra (*)*	0	0	1	1	*scorrimento aritmetico a destra*

(*) nello scorrimento a sinistra si può produrre un'inversione di segno (overflow)

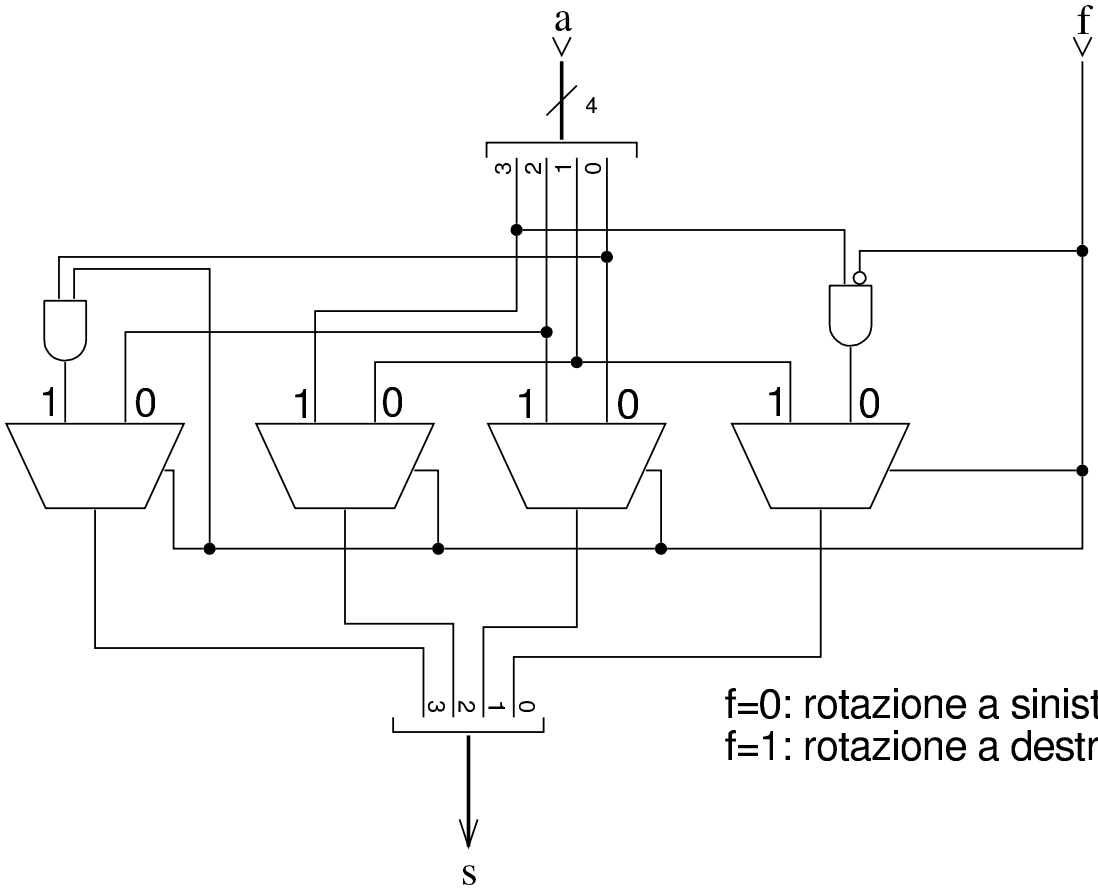
Figura u98.31. Scorrimento aritmetico, a sinistra o a destra, di quattro cifre binarie, con l'uso di multiplatori a due ingressi. Il verso dello scorrimento viene stabilito dal valore fornito nell'ingresso ***f***.



Lo scorrimento circolare comporta il recupero della cifra che fuoriesce da una parte, dal lato opposto, sia nello scorrimento a sinistra, sia in quello a destra.

1	1	1	0	*valore originale*		1	1	1	0	*valore originale*
1	1	0	1	*rotazione a sinistra*		0	1	1	1	*rotazione a destra*
0	1	1	0	*valore originale*		0	1	1	0	*valore originale*
1	1	0	0	*rotazione a sinistra*		0	0	1	1	*rotazione a destra*
1	0	1	0	*valore originale*		1	0	1	0	*valore originale*
0	1	0	1	*rotazione a sinistra*		0	1	0	1	*rotazione a destra*

Figura u98.33. Scorrimento circolare di quattro cifre binarie, con l'uso di multiplatori a due ingressi. Il verso dello scorrimento viene stabilito dal valore fornito nell'ingresso ***f***.



Lo scorrimento circolare con riporto, utilizza il riporto preesistente per la cifra da immettere, da un lato o dall'altro, mentre mette nel riporto in uscita la cifra che viene espulsa.

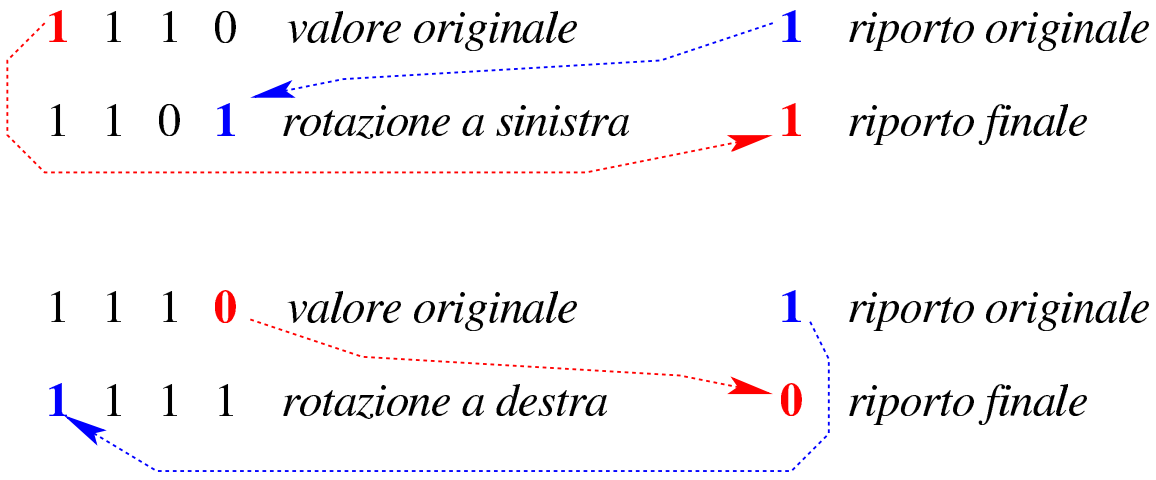
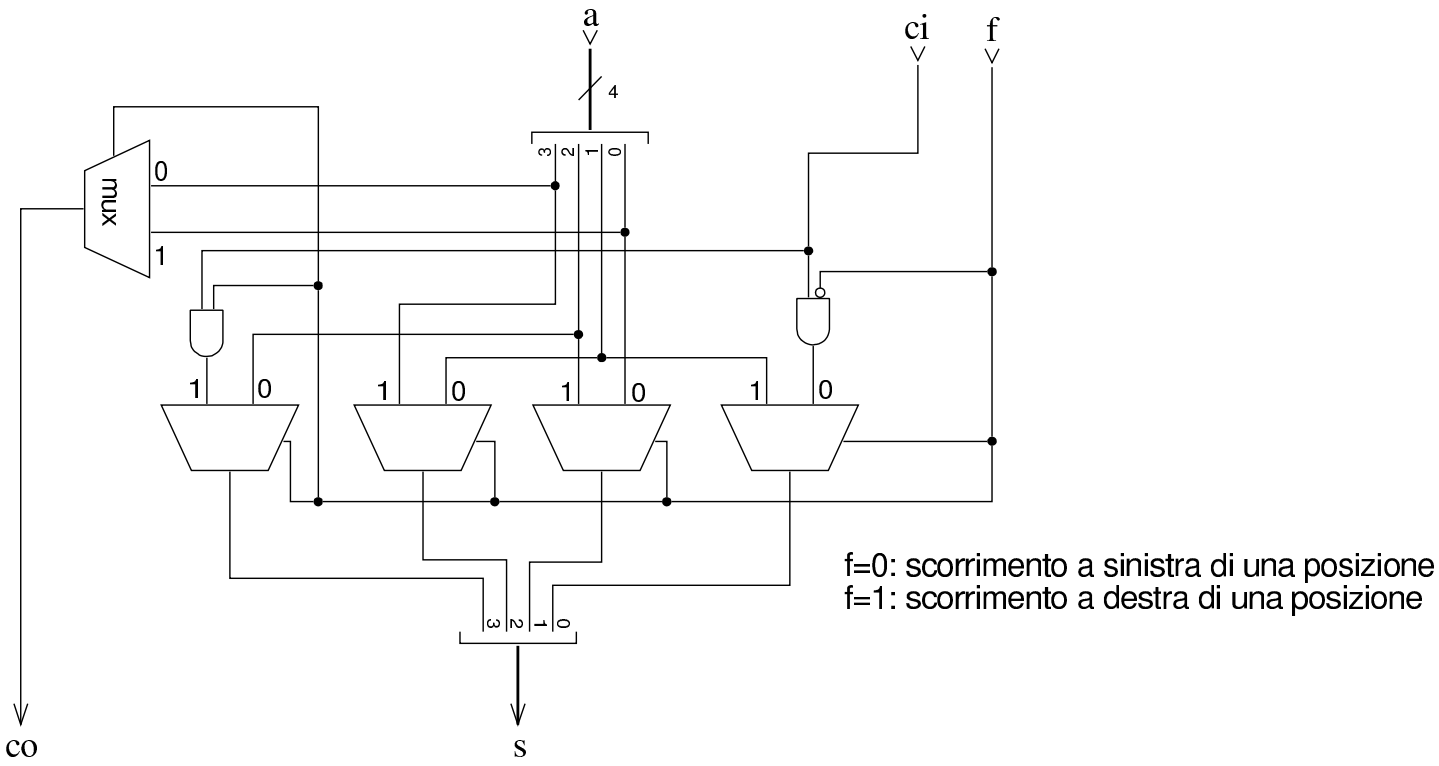


Figura u98.35. Scorrimento circolare con riporto, di quattro cifre binarie, con l'uso di multiplatori a due ingressi. Il verso dello scorrimento viene stabilito dal valore fornito nell'ingresso ***f***.



Addizionatore

L'addizione in binario, eseguita con la stessa procedura consueta per il sistema di numerazione decimale, non genera mai un riporto superiore a uno. Lo si può verificare facilmente attraverso la tabella successiva.

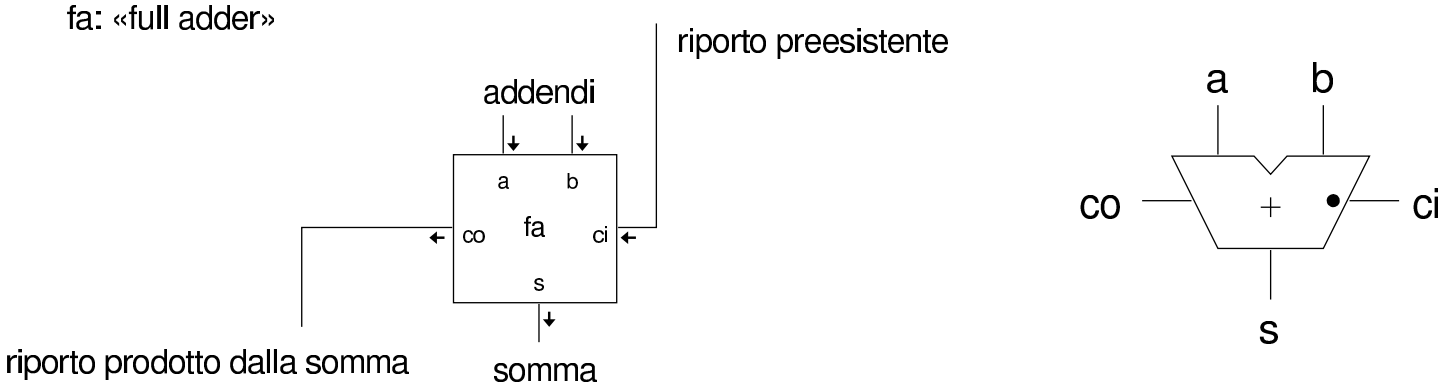
Tabella u98.36. Addizione binaria.

primo addendo	secondo addendo	riporto preesistente	riporto generato	risultato
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1

primo addendo	secondo addendo	riporto preesistente	riporto generato	risultato
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Pertanto, il circuito combinatorio che può svolgere questo lavoro deve avere tre ingressi (primo addendo, secondo addendo, riporto preesistente) e due uscite (risultato e riporto generato). Di solito si usa la lettera ***c*** (*carry*) per indicare un riporto; in questo caso si distingue tra riporto in ingresso (***ci***) e riporto in uscita (***co***).

Figura u98.37. Schema a blocchi di un addizionatore. La sigla «fa» sta per *full adder*, ovvero «addizionatore completo».



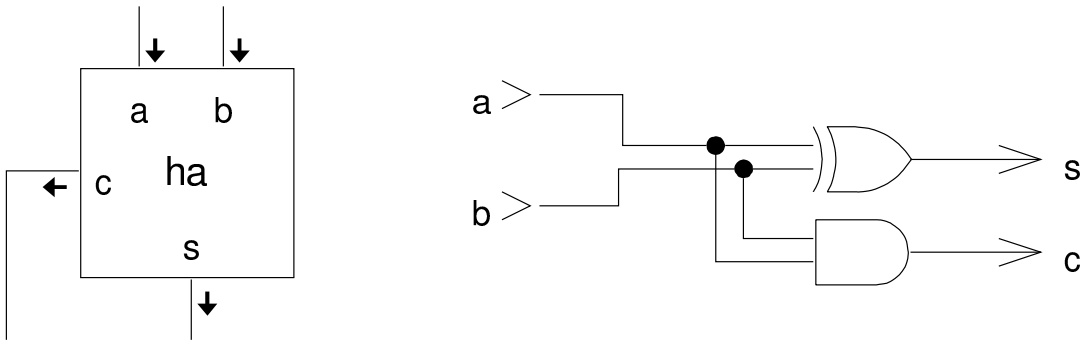
Si considera tradizionalmente che l'addizionatore completo (quello che si vede nello schema appena mostrato) sia costituito da due moduli più piccoli, noti come ***semiaddizionatori***, ovvero *half adder*. Il semiaddizionatore è un circuito combinatorio con due ingressi, ***a*** e ***b***, corrispondenti alle variabili da sommare, e due uscite, ***s*** e ***c***, corrispondenti al risultato della somma e al suo riporto.

Tabella u98.38. Tabella di verità per il semiaddizionatore.

a	*b*	*c*	*s*
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

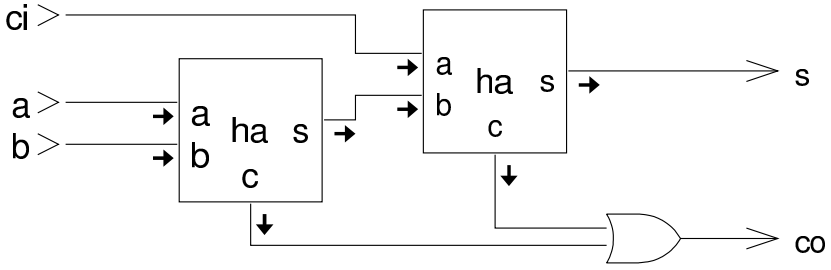
Si può intuire facilmente che, nel semiaddizionatore, l'uscita ***s*** si ottenga con una porta XOR e che il riporto si ottenga con una porta AND.

Figura u98.39. Schema a blocchi ed esempio di realizzazione di un semiaddizionatore. La sigla «ha» sta per *half adder*.



Per arrivare a ottenere l'addizionatore completo, occorre sommare anche il riporto precedente.

Figura u98.40. Schema a blocchi dell'addizionatore completo, ottenuto attraverso due semiaddizionatori e realizzazione conseguente.



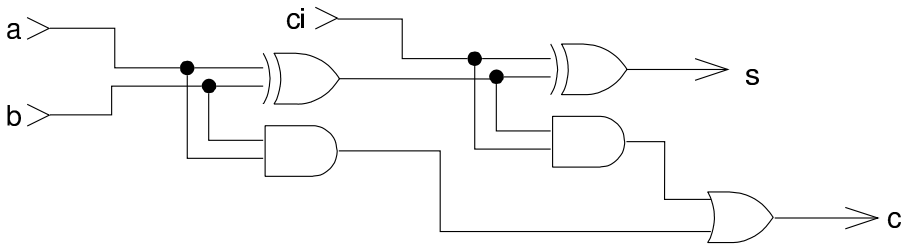
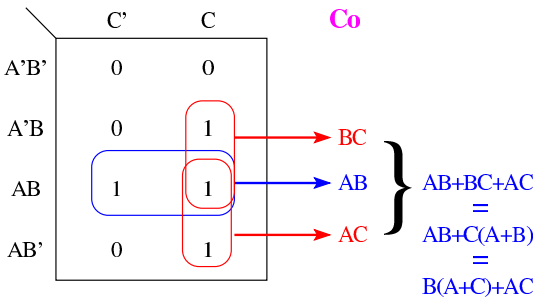
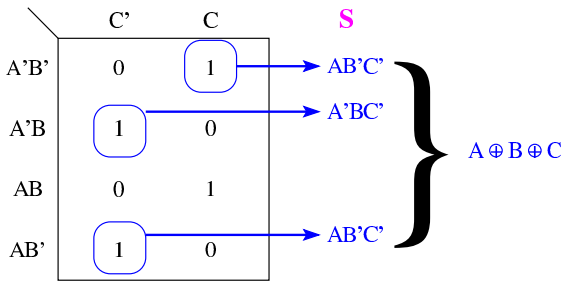
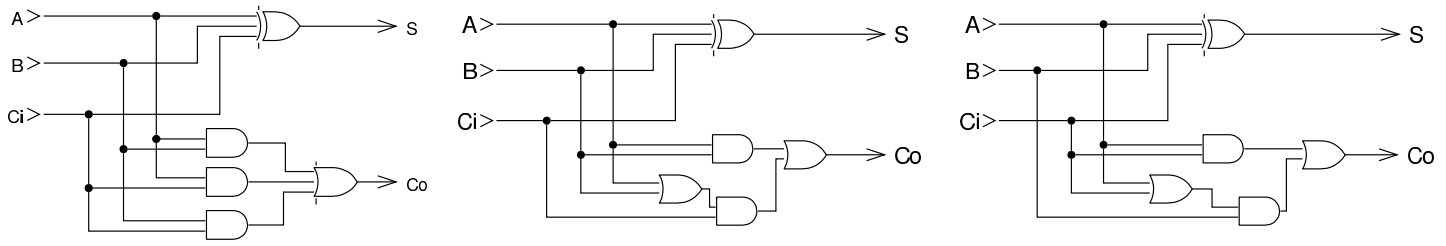


Figura u98.41. Soluzioni alternative per l'addizionatore completo, con l'aiuto delle mappe di Karnaugh.

A	B	C	Co	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	1
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



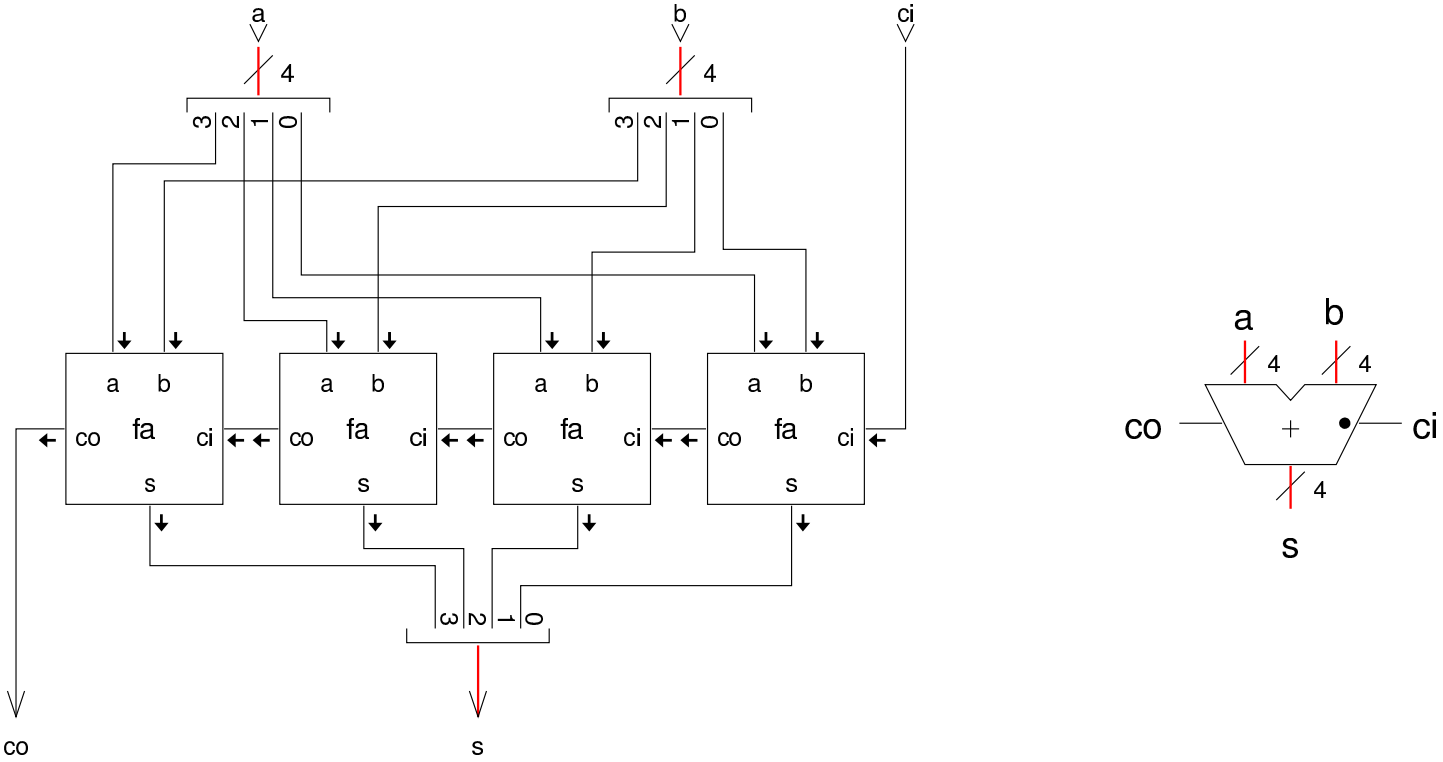




Perché l'addizionatore sia utile, è necessario che intervenga su un numero binario composto da diverse cifre, pertanto occorre assemblare in parallelo più addizionatori, passando opportunamente il riporto, dalla cifra meno significativa a quella più significativa, una

cifra alla volta.

Figura u98.42. Addizionatore a quattro cifre, costruito secondo la modalità della «propagazione del resto». Nella parte destra si vede una rappresentazione compatta di un addizionatore, senza specificare in che modo sia effettuato il calcolo.



Il circuito dell'addizionatore descritto, opera correttamente per la somma di numeri positivi o negativi, quando i numeri negativi si rappresentano attraverso la tecnica del complemento a due.

Il complemento a due che serve a trasformare il sottraendo, si ottiene con il complemento a uno del suo valore, sommandogli una unità attraverso il riporto in ingresso.

Sottrazione

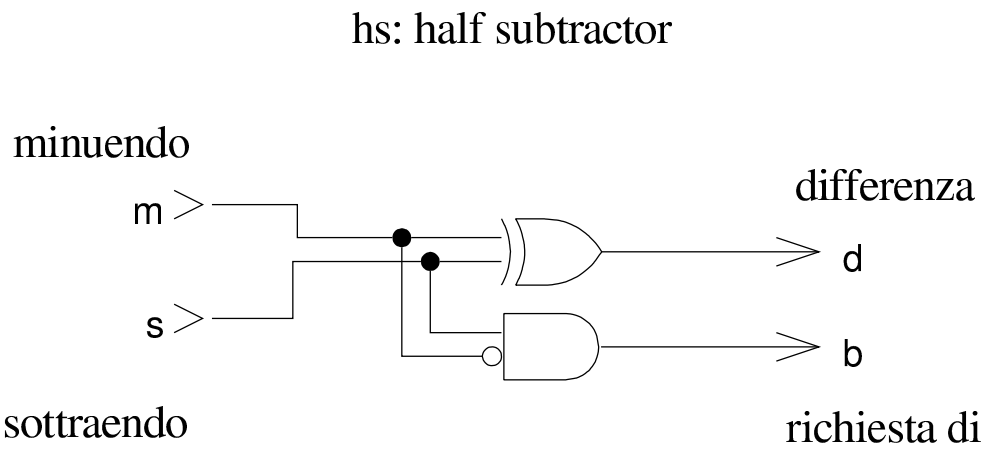
«

Per la sottrazione, si parte, anche in questo caso, dal semi-sottrattore, ovvero *half subtractor*. Trattandosi di una sottrazione, è necessario distinguere tra gli operandi il minuendo e il sottraendo; in pratica, negli esempi vengono usate queste variabili: $d=m-s$.

Tabella u98.43. Tabella di verità per il semisottrattore.

m	s	b	d
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

Figura u98.44. Semisottrattore.



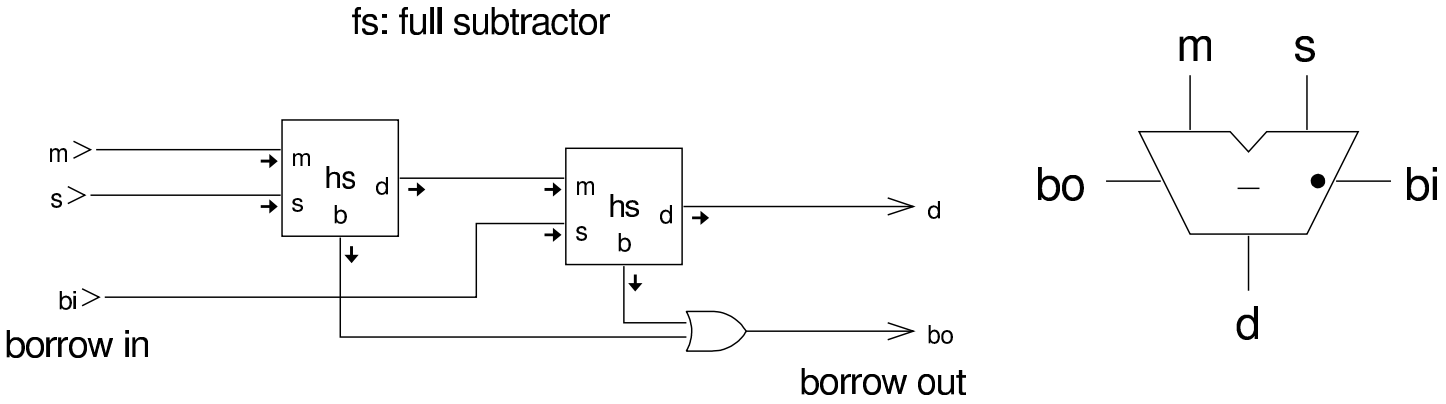
Al posto del riporto c'è un'uscita denominata *borrow* che rappresenta la richiesta del prestito di una cifra. Questa uscita diventa attiva quando il minuendo è pari a zero, mentre il sottraendo è pari a uno; ovvero quando il sottraendo è maggiore del minuendo. Il sottrattore completo ha un ingresso in più, costituito dalla richiesta di una cifra proveniente dallo stadio precedente; per gestire questo nuovo

ingresso si possono usare due semisottrattori, dove il secondo sottrae al risultato del primo la richiesta di prestito.

Tabella u98.45. Tabella di verità per il semisottrattore.

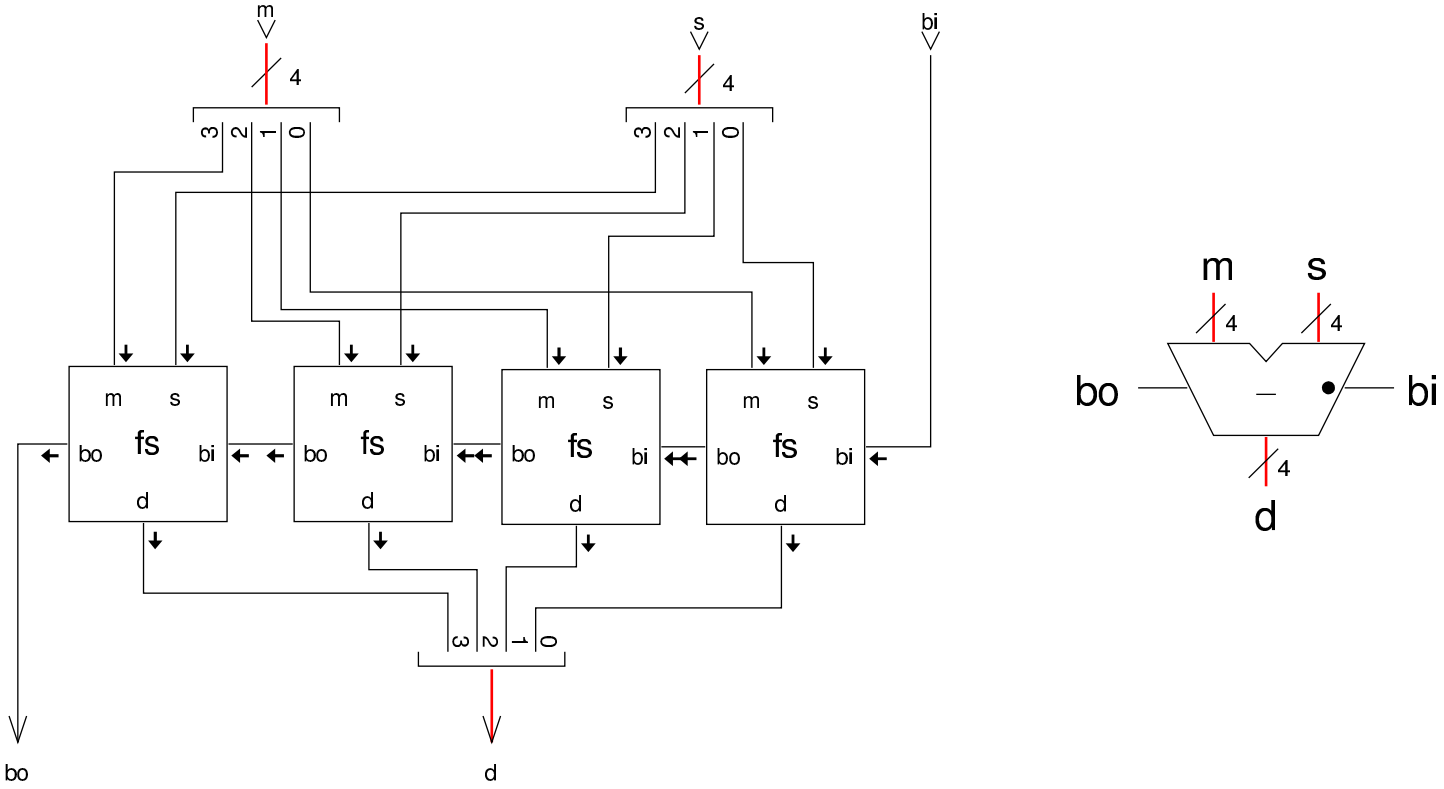
m	s	b_i	b_o	d
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Figura u98.46. Sottrattore completo.



Come nel caso della somma, si possono creare delle catene di sottrattori completi; tuttavia, nel confronto con il circuito della somma, modificato per effettuare la sottrazione, la linea *borrow* (quella della richiesta del prestito), funziona in modo inverso, in quanto se attiva, rappresenta una cifra da togliere.

Figura u98.47. Sottrazione su più bit simultaneamente.

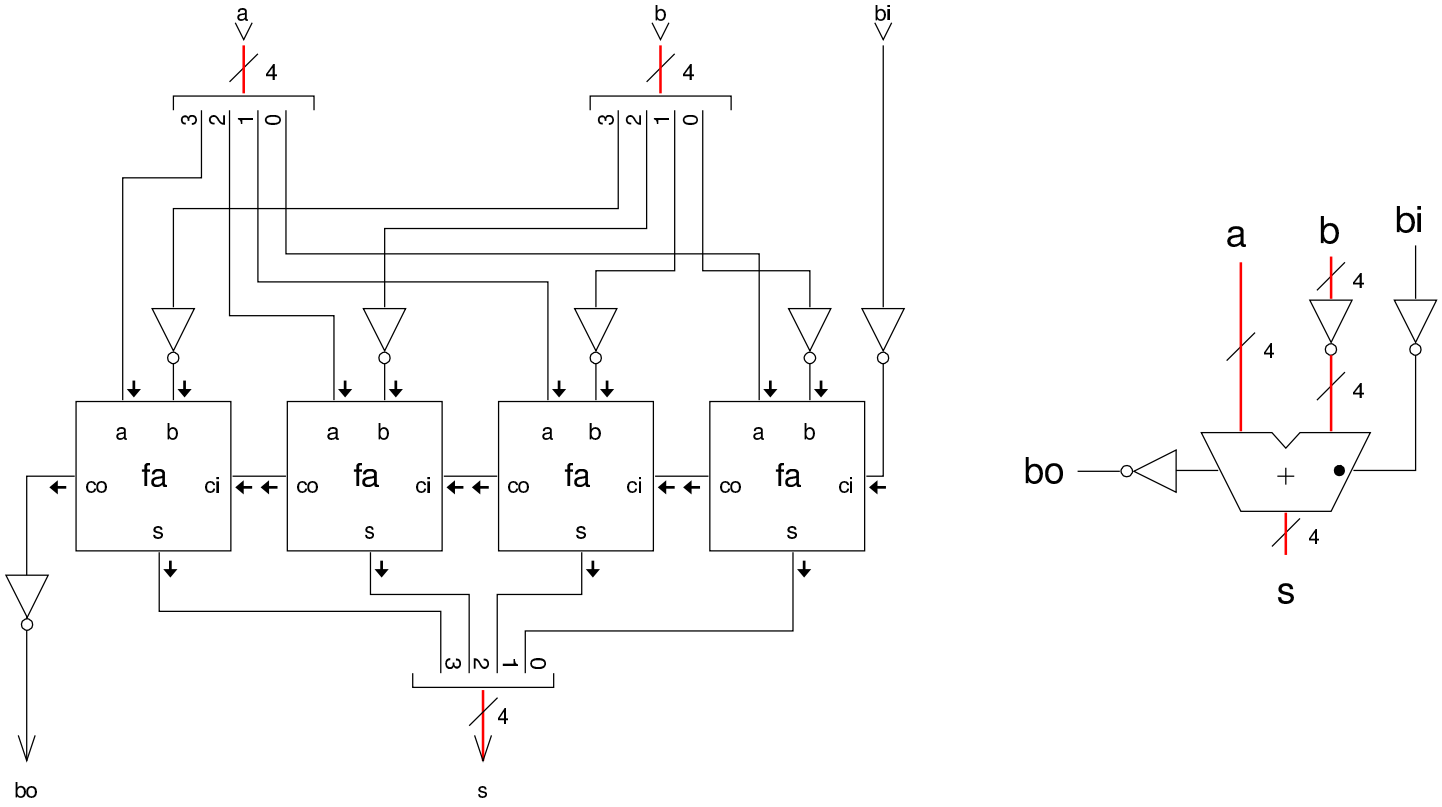


Somma e sottrazione assieme

«

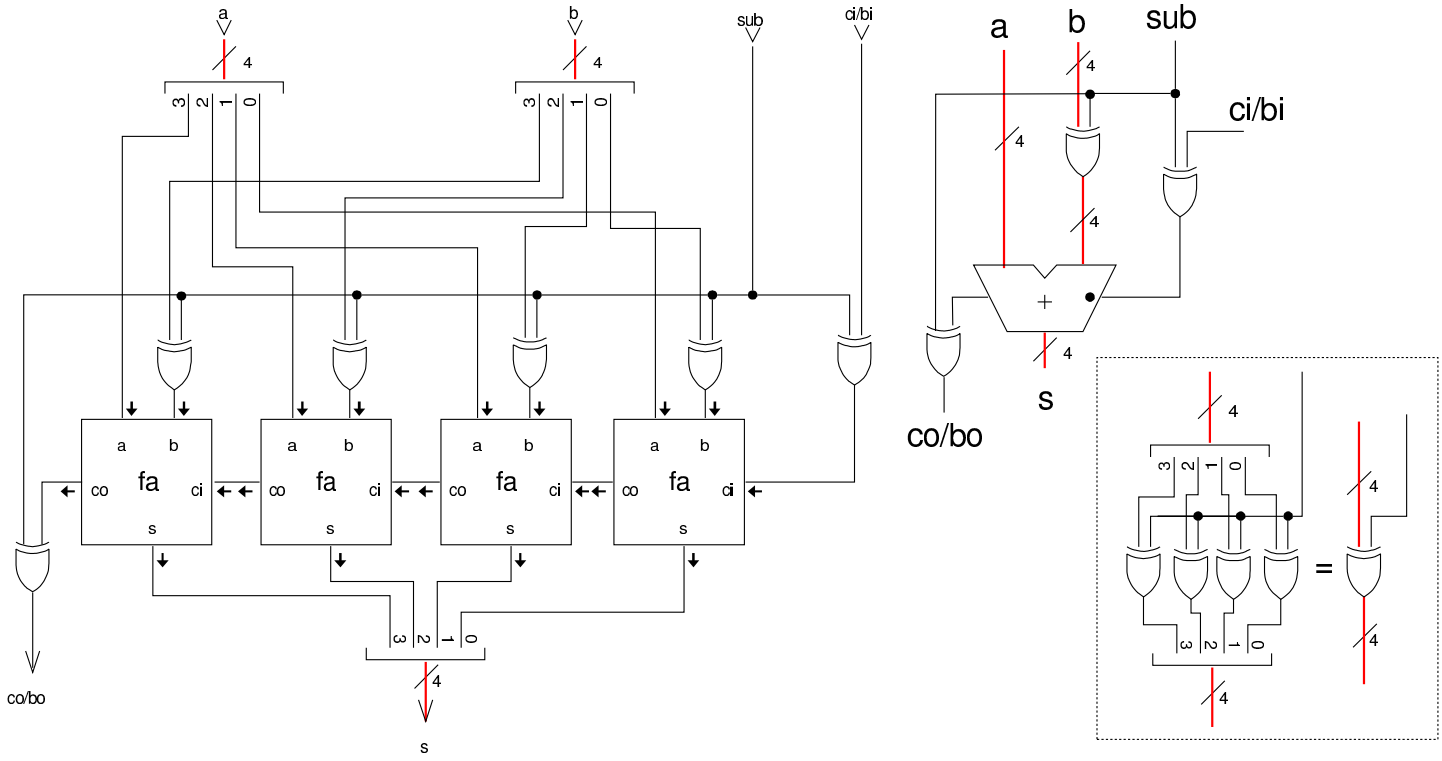
Il circuito dell'addizionatore opera correttamente per la somma di numeri positivi o negativi, quando i numeri negativi si rappresentano attraverso la tecnica del complemento a due; pertanto, per trasformare l'addizionatore in un circuito che invece sottrae uno dei due operandi, è sufficiente calcolare per quello il complemento a due e per ottenerlo si devono invertire tutti gli ingressi, compreso il riporto. Va ricordato che nella sottrazione, il riporto assume il significato della richiesta di una cifra.

Figura u98.48. Sottrattore a quattro cifre: in questo caso si tratta precisamente di $a-b$. Nella parte destra si vede una rappresentazione compatta della stessa cosa, dove il simbolo della porta NOT va intesa come un'abbreviazione di quattro porte NOT indipendenti.



Eventualmente, si trasforma facilmente il circuito combinatorio in un complesso unico, in grado di sommare o di sottrarre, sfruttando una porta XOR al posto della porta NOT, aggiungendo un ingresso ulteriore che permetta di stabilire se si esegue una somma o se l'operando stabilito va invece sottratto.

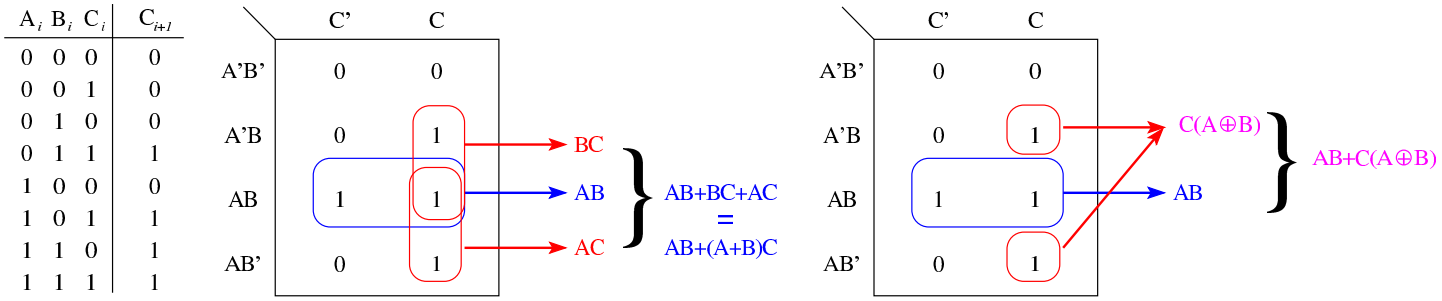
Figura u98.49. Somma o sottrazione a quattro cifre. Nella parte destra si vede una rappresentazione compatta della stessa cosa, dove il simbolo della porta XOR va inteso come un'abbreviazione di quattro porte XOR collegate assieme come si vede nel piccolo riquadro esplicativo.



Riporto anticipato

Nella somma di una quantità significativa di bit, la propagazione del riporto richiede un tempo relativamente elevato, dato che la somma di una certa cifra è corretta solo se è già avvenuta la somma di quelle che la precedono. Per poter accelerare l'esecuzione della somma, è necessario che per ogni cifra si possa sapere, nel tempo più breve possibile, qual è il riporto generato fino allo stadio precedente. Nelle espressioni successive, le variabili A_i, B_i e C_i, rappresentano i due addendi e il riporto in ingresso dello stadio i; pertanto, il riporto generato da questo stadio è rappresentato dalla variabile C_{i+1}.

Figura u98.50. Sintesi alternative del calcolo del riporto.



Nella figura si vede che il riporto si può sintetizzare in vari modi e in uno di questi appare anche l'uso dell'operatore XOR. Di quelle mostrate nella figura si scelgono due espressioni equivalenti:

$$C_{i+1} = A_iB_i+(A_i+B_i)C_i = A_iB_i+(A_i \oplus B_i)C_i$$

Da queste espressioni, si dichiarano due variabili nuove, $\boldsymbol{G_i}$ e $\boldsymbol{P_i}$, le quali stanno rispettivamente per «generazione» e «propagazione», per cui si definisce che il riporto $\boldsymbol{C_{i+1}}$ è prodotto come funzione delle variabili $\boldsymbol{G_i}$, $\boldsymbol{P_i}$ e $\boldsymbol{C_i}$.

$$C_{i+1} = G_i+P_iC_i$$

È evidente che $\boldsymbol{G_i}$ equivale a $\boldsymbol{A_iB_i}$, mentre $\boldsymbol{P_i}$ può essere considerata pari a $\boldsymbol{A_i+B_i}$ oppure $\boldsymbol{A_i \oplus B_i}$, indifferentemente. Se il primo riporto generato ($\boldsymbol{C_1}$) si può ottenere come $\boldsymbol{C_1=G_0+P_0C_0}$, il secondo si ottiene come $\boldsymbol{C_2=G_1+P_1(G_0+C_0P_0)}$, e di conseguenza si può proseguire per determinare i riporti successivi. Vengono mostrate le soluzioni per i primi quattro riporti:

$$C_1 = G_0 + P_0 C_0$$

$$C_2 = G_1 + P_1 C_1$$
$$C_2 = G_1 + P_1 (G_0 + P_0 C_0)$$
$$C_2 = G_1 + P_1 G_0 + P_1 P_0 C_0$$
$$C_3 = G_2 + P_2 C_2$$
$$C_3 = G_2 + P_2 (G_1 + P_1 G_0 + P_1 P_0 C_0)$$
$$C_3 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$$
$$C_4 = G_3 + P_3 C_3$$
$$C_4 = G_3 + P_3 (G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0)$$
$$C_4 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_0$$

Per semplificare le espressioni, si definiscono $\boldsymbol{P_n}$ e $\boldsymbol{G_n}$ nel modo seguente:

$$P_n = P_{n-1} P_{n-2} P_{n-1} \cdots P_1 P_0$$

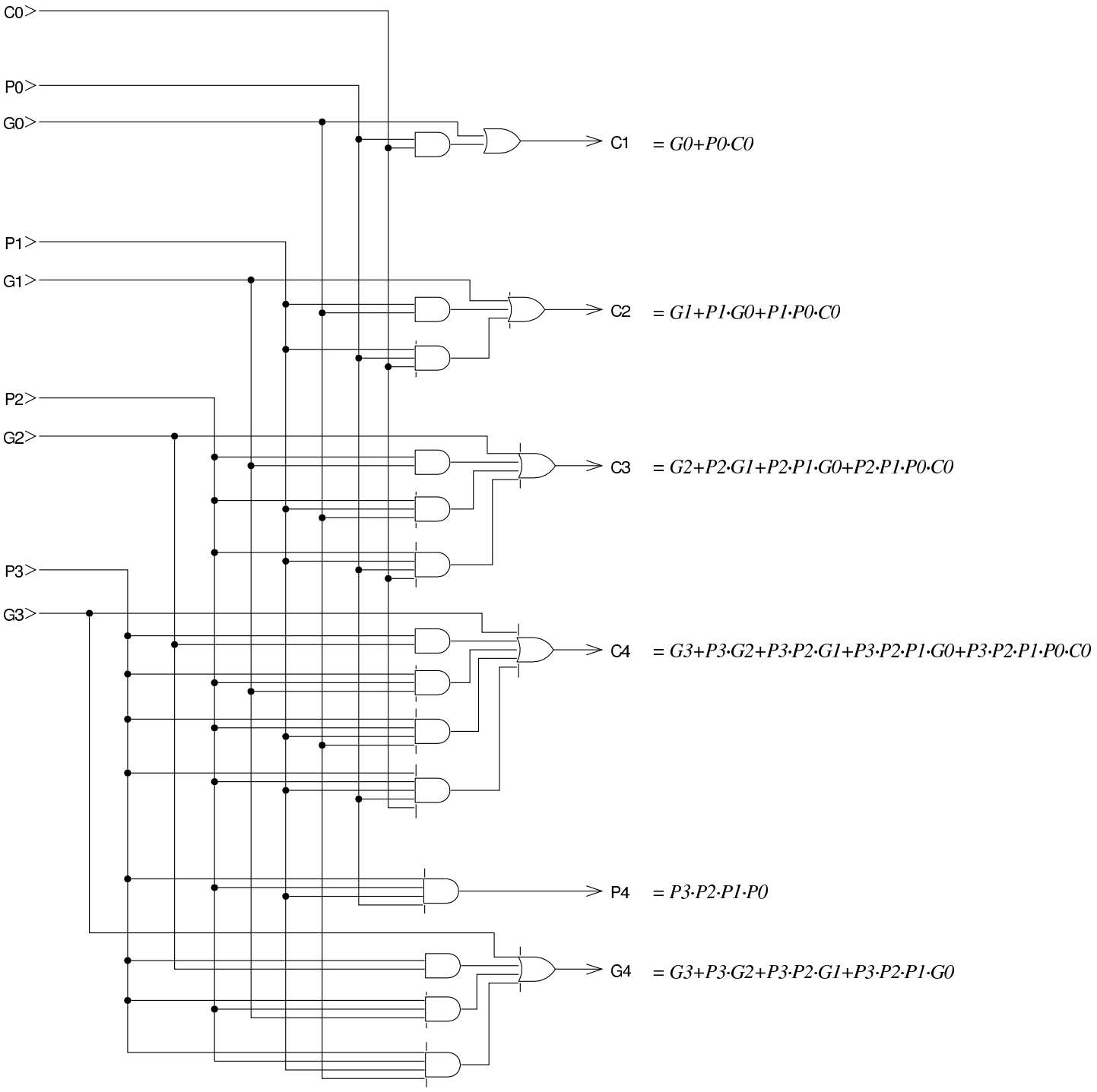
$$G_n = G_{n-1} + P_{n-1} G_{n-2} + P_{n-2} P_{n-1} G_{n-3} + \cdots + P_{n-2} P_{n-1} \cdots P_1 G_0$$

Avendo definito ciò, il riporto $\boldsymbol{C_n}$ si può definire come:

$$C_n = G_n + P_n C_0$$

La figura successiva, mostra un circuito combinatorio che determina i riporti di quattro cifre binarie, partendo dal riporto iniziale e dai valori di $\boldsymbol{B_{3..0}}$ e $\boldsymbol{G_{3..0}}$, come descritto dalle equazioni che definiscono questa relazione. Il disegno contiene anche la logica necessaria a determinare il valore di $\boldsymbol{B_4}$ e $\boldsymbol{G_4}$ che possono servire per collegare assieme più moduli di questo tipo.

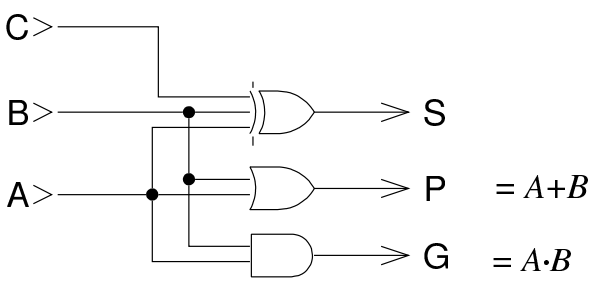
Figura u98.56. Schema per la determinazione di quattro riporti, a partire dal riporto iniziale (C_0) e dai valori di P_i e G_i, come descritto nelle equazioni logiche.

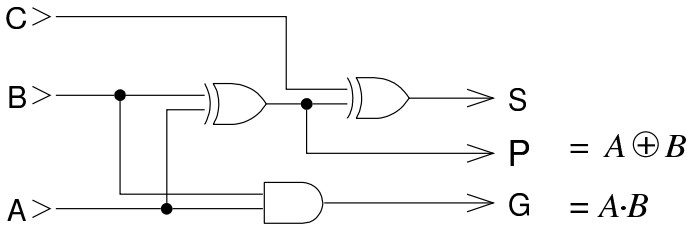


Avendo la necessità di disporre delle uscite G e P, si può sintetizzare un modulo per l'addizione privo della funzione che determina il

riporto in uscita, dal momento che questo compito viene affidato al modulo che si vede nella figura precedente. Ci sono due soluzioni alternative, nelle quali il valore di ***P*** viene determinato nei due modi diversi già descritti con le tabelle di Karnaugh.

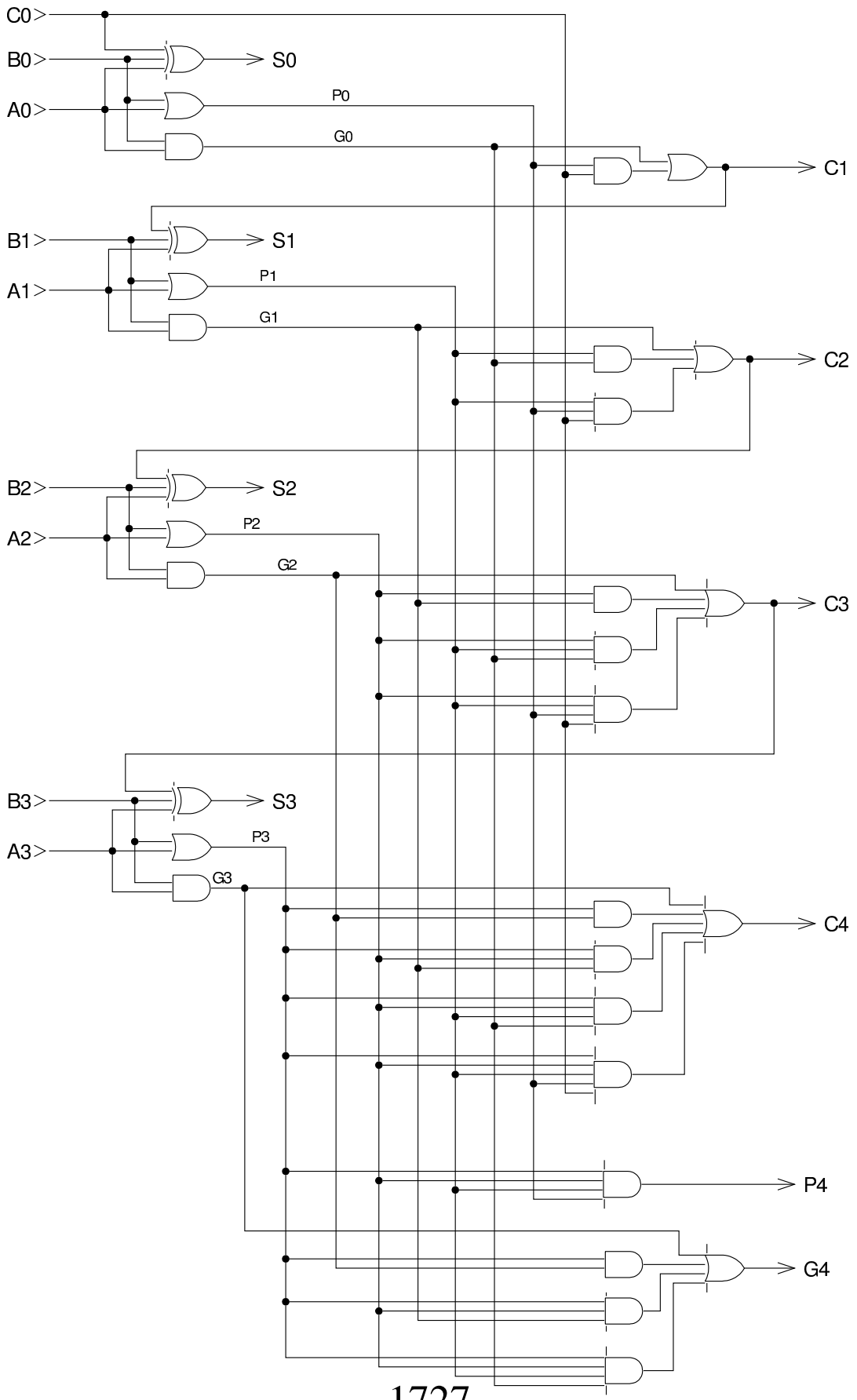
Figura u98.57. Moduli per l'addizione con le uscite ***P*** e ***G***, ma privi dell'uscita con il riporto per la cifra successiva.





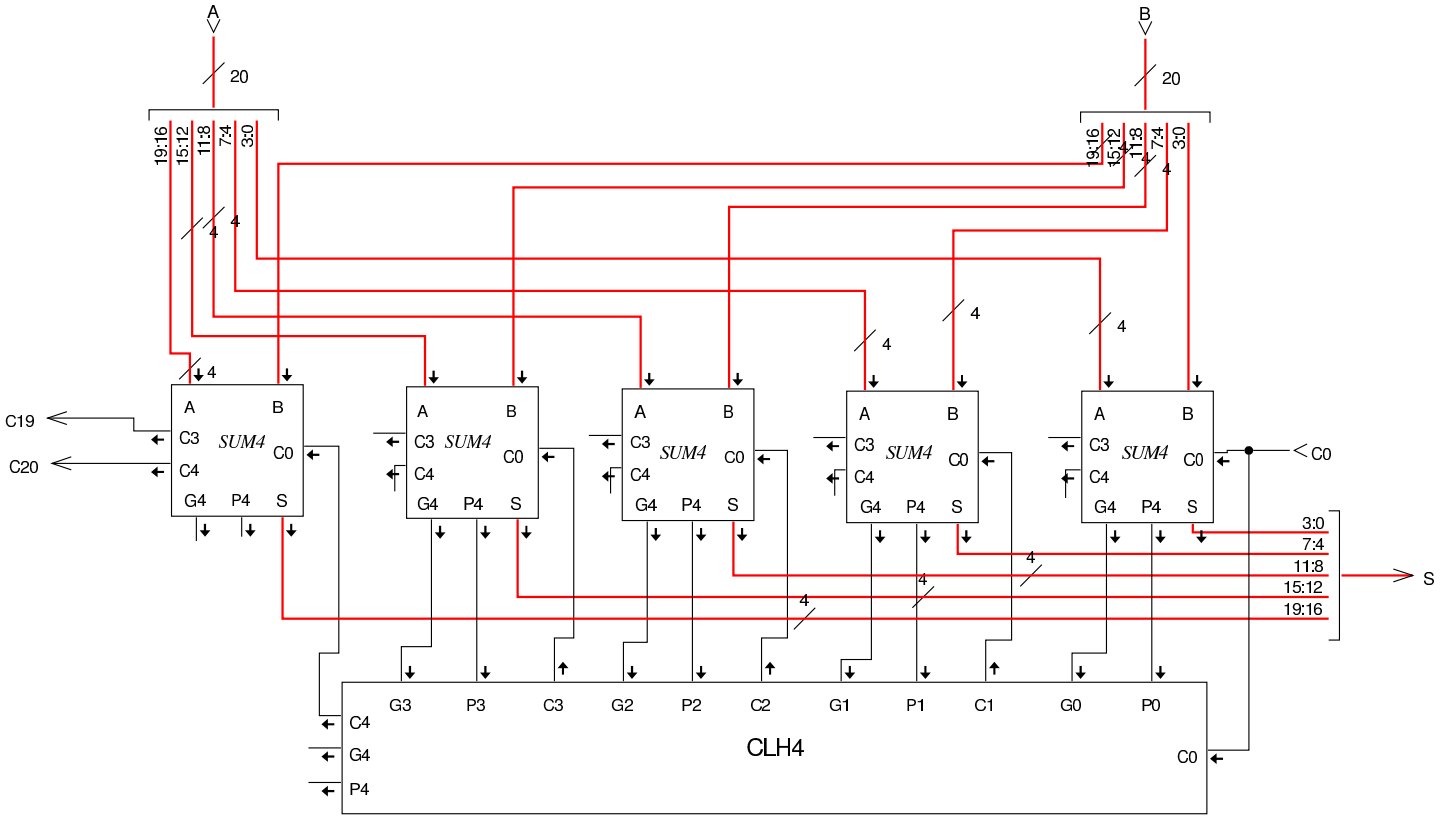
La figura successiva mostra un addizionatore a quattro cifre, dove si utilizzano i moduli di addizione e di determinazione del riporto già apparsi.

Figura u98.58. Addizionatore a quattro cifre.



È possibile collegare assieme i moduli mostrati, quando non si può disporre in partenza della quantità di cifre che servono. Questo collegamento comporta un aumento del ritardo di propagazione, ma si tratta comunque di un grande miglioramento rispetto al calcolo del riporto in cascata, come mostrato nelle sezioni precedenti. Nella figura successiva, il modulo SUM4 corrisponde allo schema di figura u98.58, dal quale si prelevano solo l'ultimo e il penultimo riporto (perché in sezioni successive viene mostrato che questi due consentono di determinare la presenza di uno straripamento); invece, il modulo CLH4 corrisponde allo schema di figura u98.56, il quale viene usato per mettere assieme i moduli di addizione.

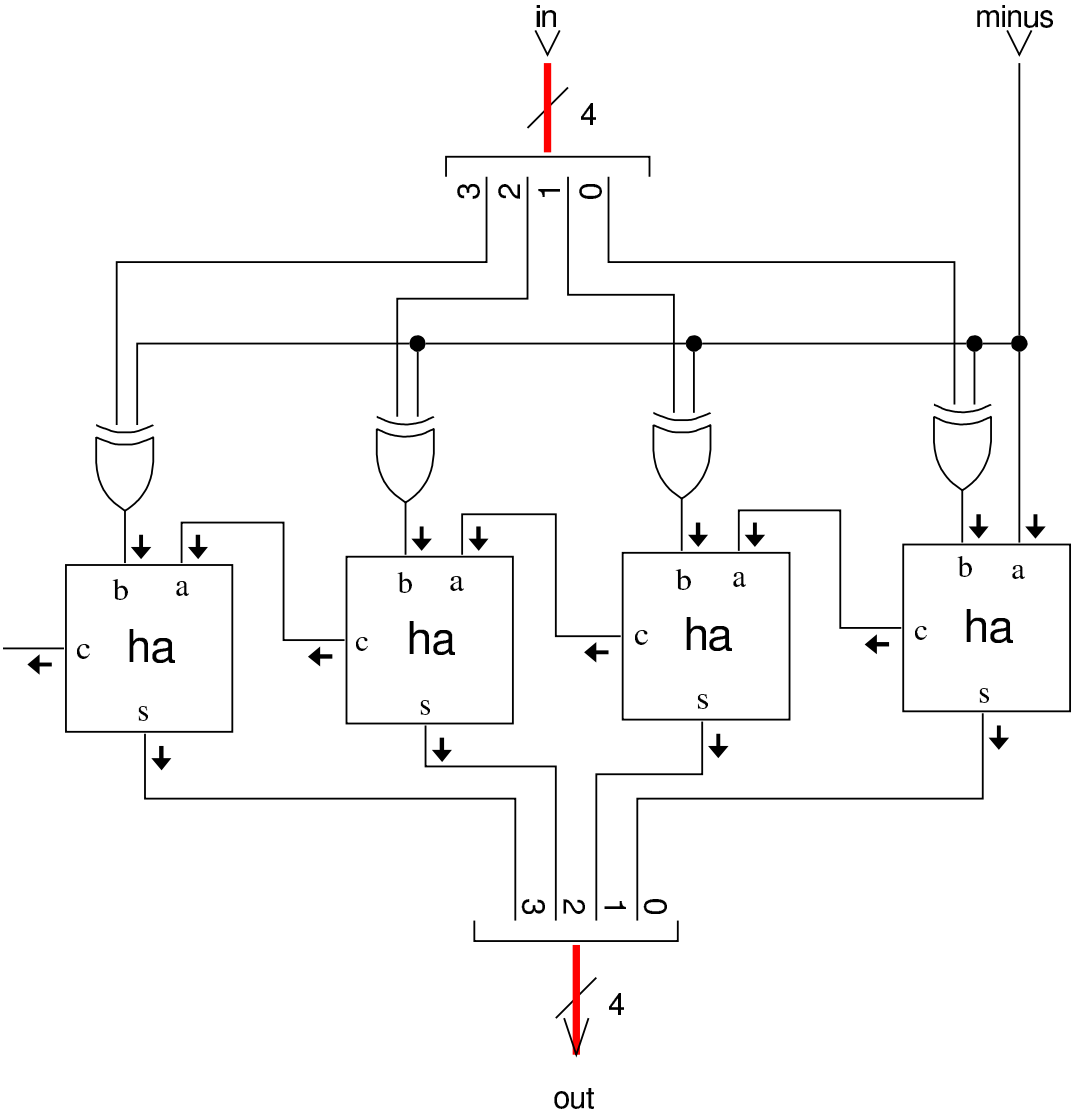
Figura u98.59. Addizionatore a 20 cifre, ottenuto partendo da moduli da quattro cifre concatenati assieme.



Complemento a due

Se si ha la necessità di invertire il segno di un numero intero, rappresentato in forma binaria, c'è bisogno di costruire un circuito che esegua il complemento a uno, invertendo le cifre binarie, sommando poi una unità per trovare il complemento a due.

Figura u98.60. Calcolo del complemento a due, corrispondente all'inversione del segno di un numero intero a quattro cifre.



L'esempio proposto nella figura mostra che il dato in ingresso, viene invertito (negato) se risulta attivo il segnale ***minus***, ma lo stesso segnale ***minus*** viene sommato, producendo alla fine il complemento a

due. Se invece il segnale ***minus*** non fosse attivo, il dato in ingresso non verrebbe alterato e non ci sarebbe l'incremento di un'unità, per cui il risultato sarebbe lo stesso, senza variazione.

Moltiplicazione

La moltiplicazione binaria è un procedimento che richiede la somma e lo scorrimento. Per poter tenere conto del segno, il calcolo andrebbe fatto come se si operasse su una quantità doppia di cifre; per esempio, se moltiplicatore e moltiplicando sono di sole quattro cifre binarie, il risultato deve poter essere di otto cifre e il calcolo andrebbe fatto come se anche gli operandi fossero di questo rango. Si osservino gli esempi che appaiono nella figura successiva.

Figura u98.61. Moltiplicazione di numeri a quattro cifre binarie, senza segno e con segno.

moltiplicazione di interi senza segno	*moltiplicando negativo, moltiplicatore positivo*	*moltiplicando positivo, moltiplicatore negativo*	*moltiplicando negativo, moltiplicatore negativo*
0 0 0 0 **1 0 1 1** ×	1 1 1 1 **1 0 1 1** ×	0 0 0 0 **0 1 1 1** ×	1 1 1 1 **1 0 1 1** ×
0 0 0 0 **1 1 0 1** =	0 0 0 0 **0 1 0 1** =	1 1 1 1 **1 1 0 1** =	1 1 1 1 **1 1 0 1** =
0 0 0 0 **1 0 1 1** +	1 1 1 1 **1 0 1 1** +	0 0 0 0 **0 1 1 1** +	1 1 1 1 **1 0 1 1** +
0 0 0 **0 0 0 0** 0 +	0 0 0 **0 0 0 0** 0 +	0 0 0 **0 0 0 0** 0 +	0 0 0 **0 0 0 0** 0 +
0 0 **1 0 1 1** 0 0 +	1 1 **1 0 1 1** 0 0 +	0 0 **0 1 1 1** 0 0 +	1 1 **1 0 1 1** 0 0 +
0 **1 0 1 1** 0 0 0 +	0 **0 0 0 0** 0 0 0 +	0 **0 1 1 1** 0 0 0 +	1 **1 0 1 1** 0 0 0 +
0 0 0 0 0 0 0 0 +	0 0 0 0 0 0 0 0 +	0 1 1 1 0 0 0 0 +	1 0 1 1 0 0 0 0 +
0 0 0 0 0 0 0 0 +	0 0 0 0 0 0 0 0 +	1 1 1 0 0 0 0 0 +	0 1 1 0 0 0 0 0 +
0 0 0 0 0 0 0 0 +	0 0 0 0 0 0 0 0 +	1 1 0 0 0 0 0 0 +	1 1 0 0 0 0 0 0 +
0 0 0 0 0 0 0 0 =	0 0 0 0 0 0 0 0 =	1 0 0 0 0 0 0 0 =	1 0 0 0 0 0 0 0 =
1 0 0 0 1 1 1 1	1 1 1 0 0 1 1 1	1 1 1 0 1 0 1 1	0 0 0 0 1 1 1 1

Nella figura, l'esempio di sinistra mostra la moltiplicazione tra 11 e 13; trattandosi di numeri privi di segno, vanno aggiungi degli zeri

nelle posizioni più significative e di conseguenza va svolta la moltiplicazione. Nel secondo esempio, i numeri vanno intesi con segno e si tratta della moltiplicazione tra −5 e +5; in questo caso, il numero che viene inteso come negativo, deve essere completato con cifre a uno, per mantenere il segno negativo, e di conseguenza la moltiplicazione ne tiene conto. Nel terzo esempio è il moltiplicatore a essere negativo: 7·−3. In tal caso è il moltiplicatore che viene completato con le cifre a uno nella parte più significativa, condizionando di conseguenza il calcolo della moltiplicazione. L'ultimo esempio è uguale al primo, con la differenza che i due valori sono intesi con segno, pertanto sono completati con cifre a uno. In conclusione, la moltiplicazione deve tenere conto del fatto che i numeri siano con segno o senza; nel caso lo siano, devono essere estesi nella porzione più significativa con la cifra necessaria a mantenere il segno che hanno.

Per risolvere il problema in forma di circuito combinatorio, occorre incrociare i valori di moltiplicando e moltiplicatore, verificando in ogni posizione utile la coincidenza di valori a uno. I due disegni successivi vanno sovrapposti idealmente, in quanto mostrano il concetto in due fasi: le uscite delle porte AND devono essere sommate verticalmente per generare il prodotto.

Figura u98.62. Intreccio tra moltiplicando e moltiplicatore, abbinato a delle porte AND.

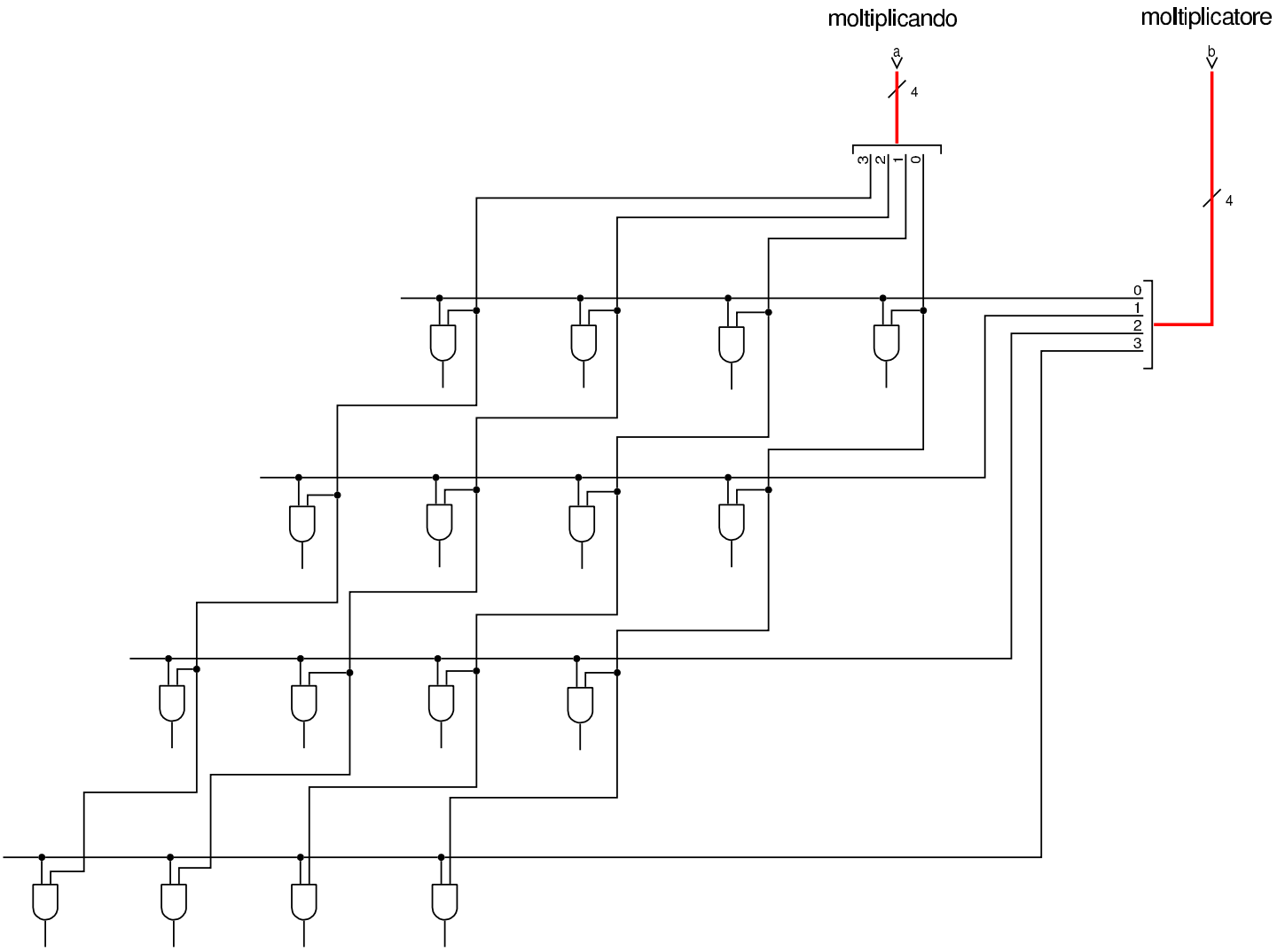
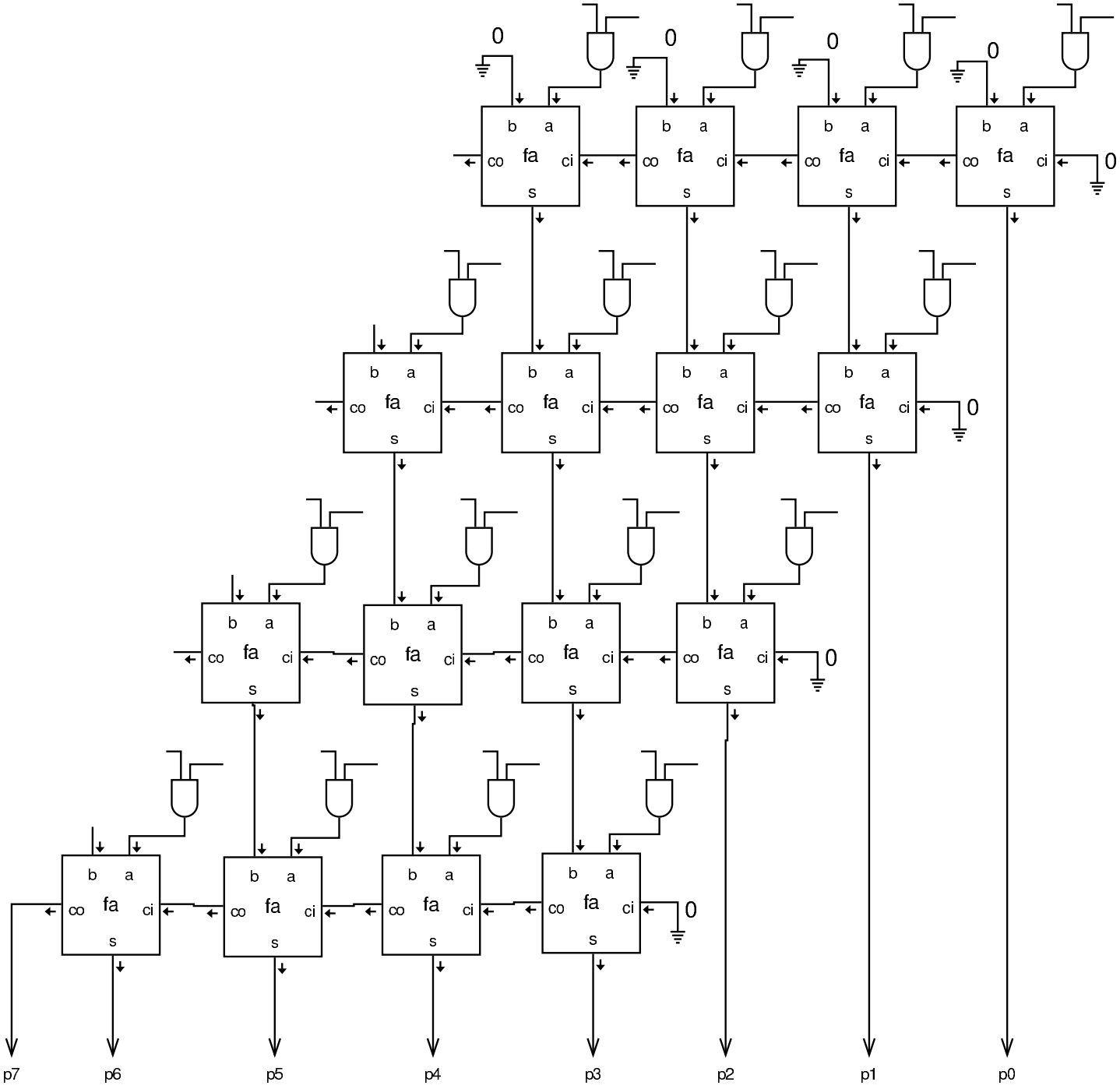


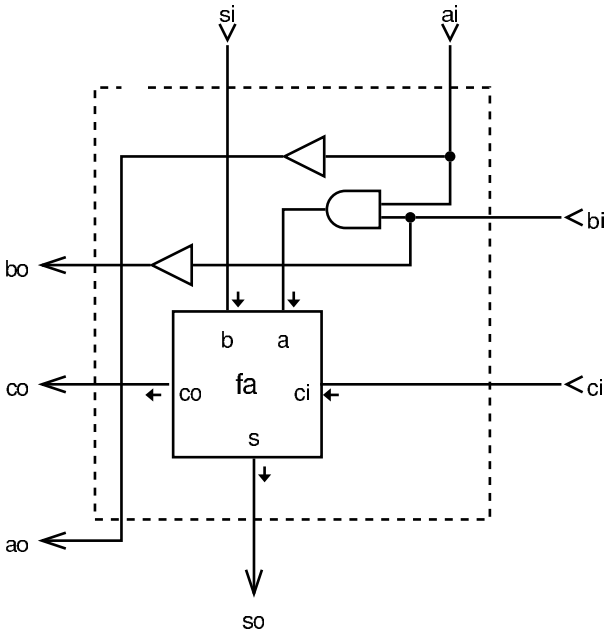
Figura u98.63. Quanto generato dalle porte AND, viene sommato con degli addizionatori completi.



Per mettere assieme la moltiplicazione, svolta dall'operatore AND, e la somma, occorre costruire delle celle apposite, utilizzano un addizionatore completo. Come si vede dalla figura, uno degli addendi riceve il risultato del prodotto ottenuto con l'operatore AND, mentre

l'altro addendo riporta il valore proveniente dalla riga superiore.

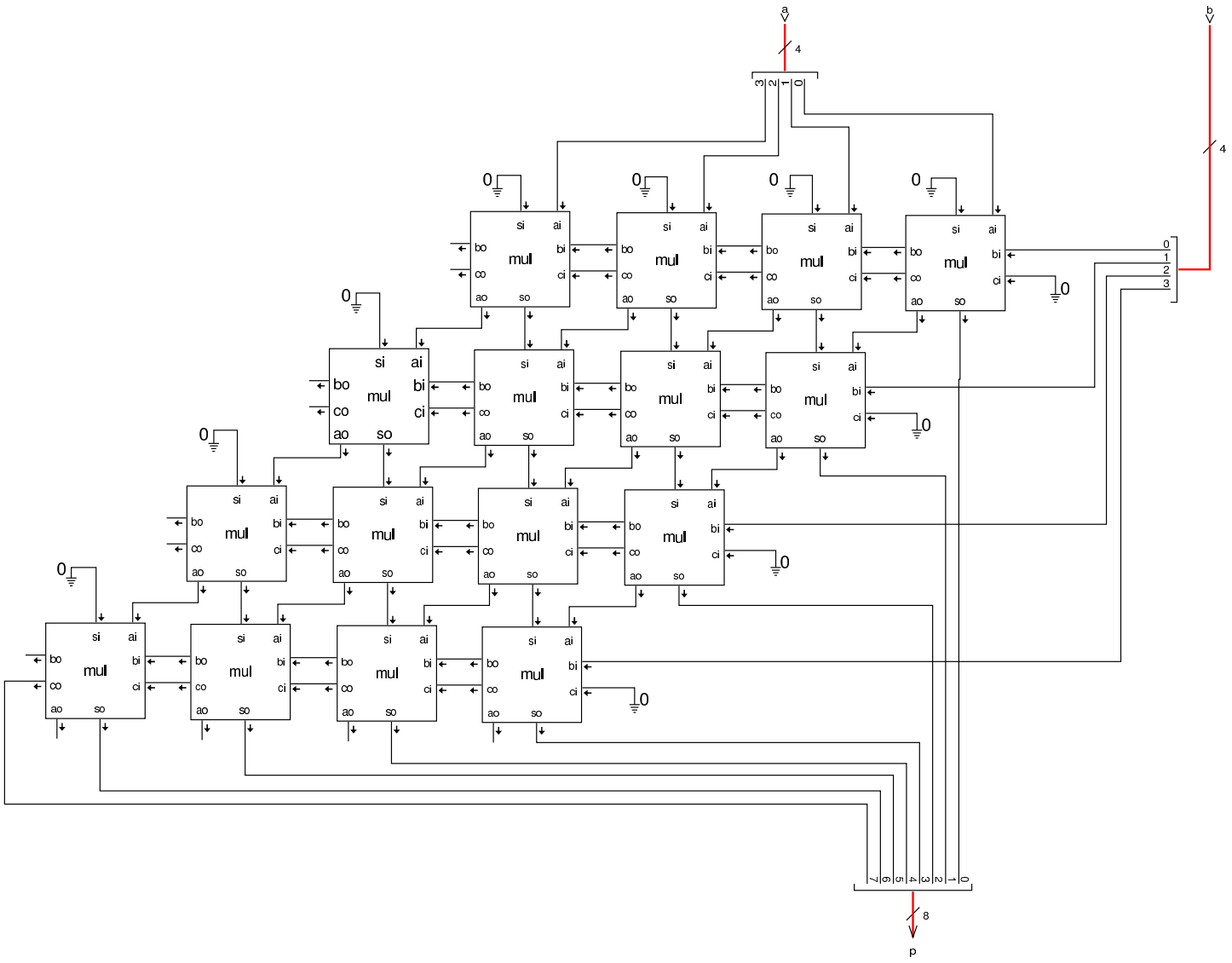
Figura u98.64. Cella usata per la costruzione della matrice che somma e fa scorrere il moltiplicando.



si = sum input
ai = a input = moltiplicando
bi = b input = moltiplicatore
ci = carry in
so = sum output = ((ai AND bi) + si + ci)
ao = ai
co = carry out = riporto della somma risultante da so
bo = bi

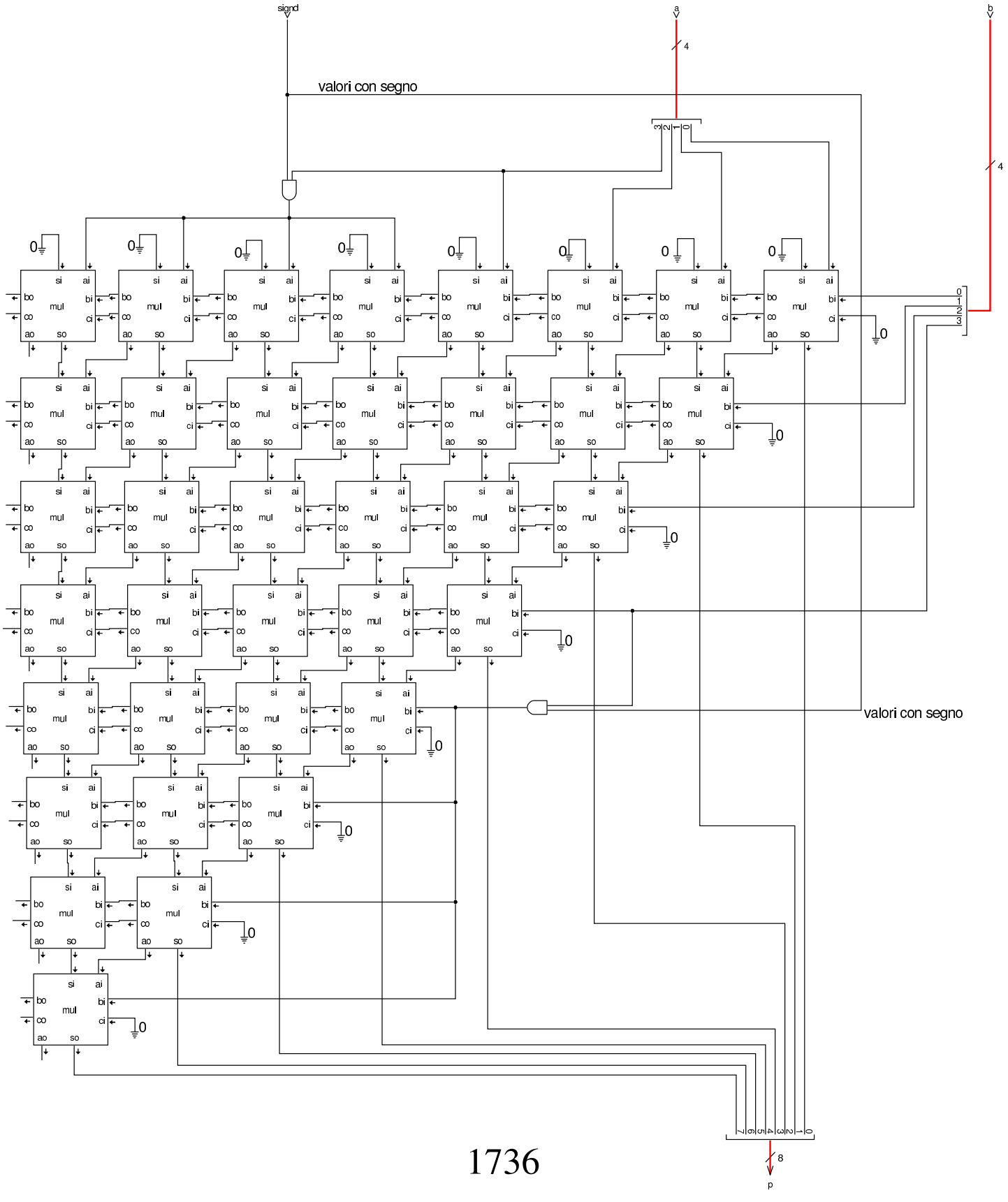
Le celle della figura vanno connesse per costruire le somme che costituiscono la moltiplicazione. La figura successiva mostra una soluzione limitata al caso di numeri privi di segno.

Figura u98.65. Moltiplicazione di interi **senza segno**, a quattro cifre, producendo un risultato a otto cifre: $p=a \cdot b$.



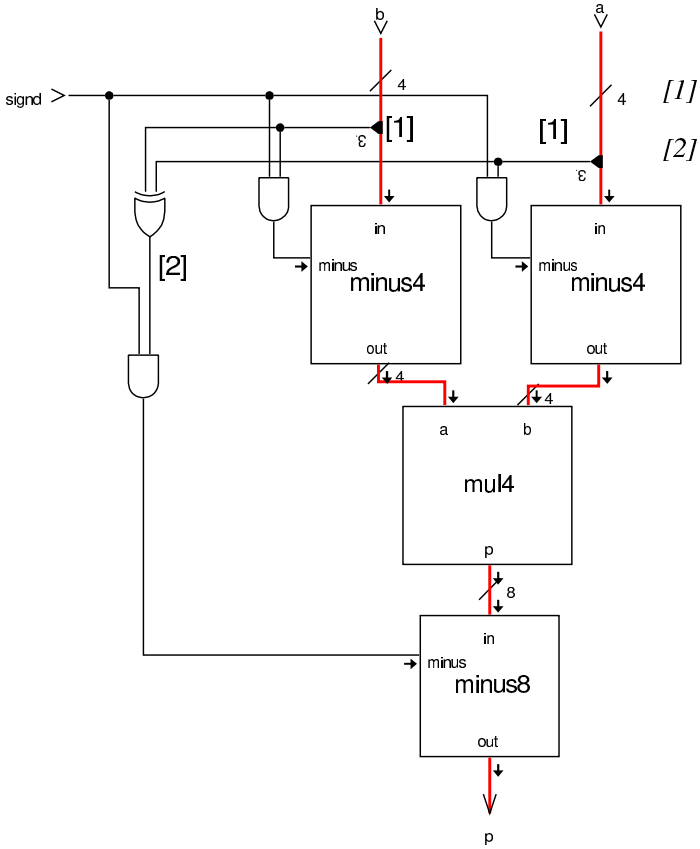
Per risolvere il problema dei valori con segno, occorre un ingresso aggiuntivo, per comunicare il fatto che si stia usando o meno numeri con segno, quindi occorrerebbe estendere la maglia come se si moltiplicassero valori con un numero doppio di cifre, estese secondo il segno che viene loro attribuito.

Figura u98.66. Soluzione completa, ma eccessivamente dispendiosa, per la moltiplicazione di due valori espressi a soli quattro bit.



Si può risolvere il problema della moltiplicazione quando si hanno valori con segno, adattare un circuito moltiplicatore di valori privi di segno, provvedendo a calcolare il complemento a due (ovvero a cambiare di segno) quando i valori risultano essere negativi. Nello schema della figura successiva, la funzione ***mul4*** corrisponde a un circuito moltiplicatore che opera solo su valori privi di segno; ***minus4*** e ***minus8*** sono circuiti che si occupano di invertire il segno se l'ingresso ***minus*** risulta attivo.

Figura u98.67. Adattamento di un moltiplicatore di valori senza segno, in modo da recepire anche valori con segno.



[1] viene letto il bit più significativo del gruppo, per determinare il segno

[2] i segni di moltiplicando e moltiplicatore sono diversi

Nella figura si vede che dagli ingressi ***a*** e ***b***, viene prelevato il bit più significativo, per determinare se i valori rispettivi sono negativi; se lo sono, i loro valori vengono moltiplicati per -1, prima di passare alla moltiplicazione; al termine, il risultato viene moltiplicato per -1

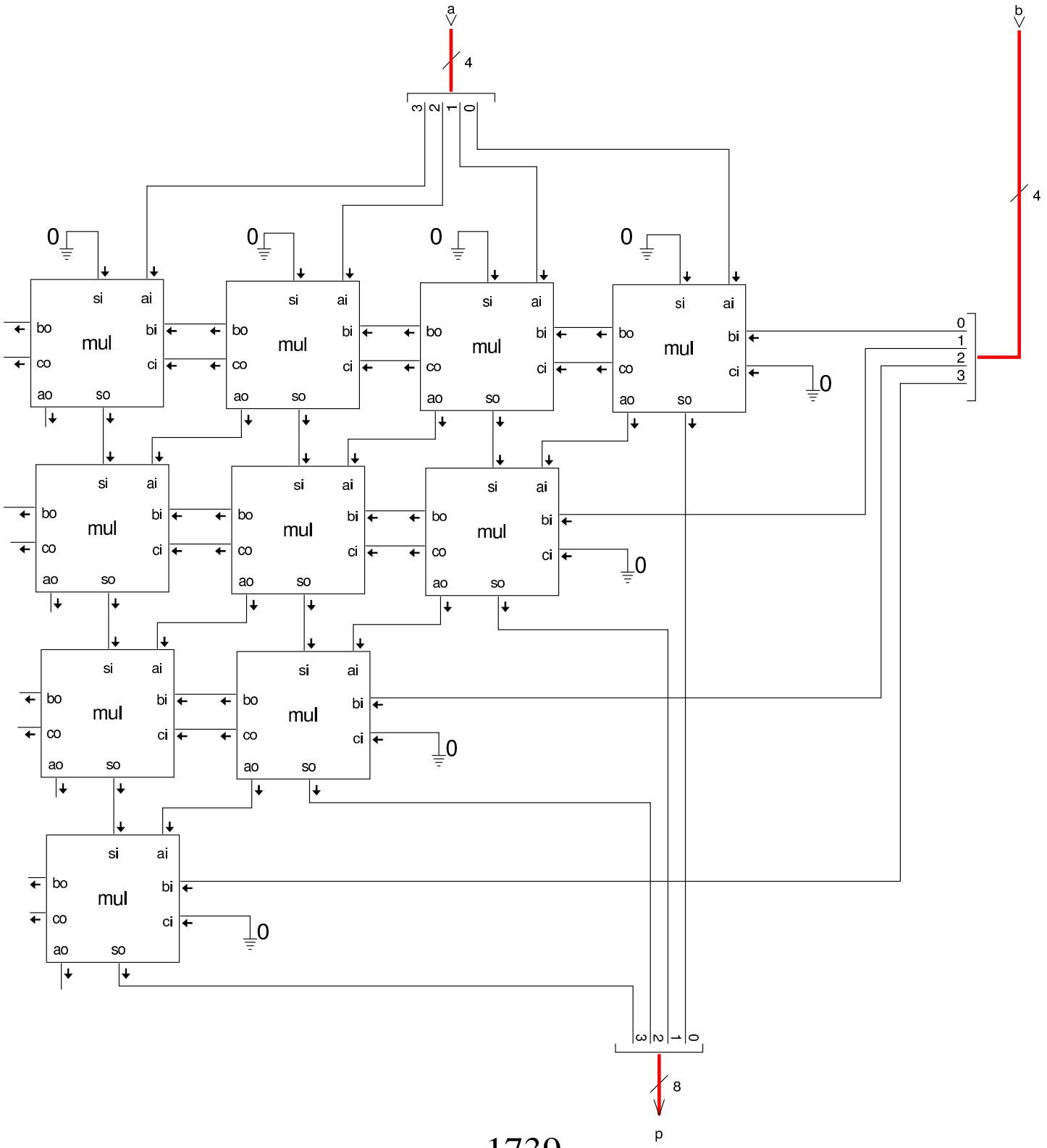
solo se i segni di ***a*** e ***b*** sono diversi. Naturalmente, tutto questo è subordinato al fatto che venga richiesto un calcolo che tenga conto dei segni, attraverso l'ingresso ***signd***. Il circuito denominato ***minus4*** corrisponde a quanto mostrato nella sezione u0.13; naturalmente, ***minus8*** è solo un'estensione dello stesso a 8 bit.

Se invece si riduce il risultato della moltiplicazione a una quantità di cifre uguale a quelle di moltiplicando e di moltiplicatore, ammesso che il risultato non venga troncato, ciò che si ottiene è valido, indipendentemente dai segni, senza bisogno di dover comunicare la gestione dei segni al circuito. Si mostra nuovamente un esempio di calcolo della moltiplicazione, simile a quello iniziale, per dimostrare questa affermazione.

Figura u98.68. Moltiplicazione di numeri a quattro cifre binarie, troncando il risultato alla stessa ampiezza di moltiplicando e di moltiplicatore.

moltiplicazione di interi senza segno	*moltiplicando negativo, moltiplicatore positivo*	*moltiplicando positivo, moltiplicatore negativo*	*moltiplicando negativo, moltiplicatore negativo*
1011×	**1101**×	**0101**×	**1011**×
1101=	**0010**=	**1110**=	**1101**=
1011+	**0000**+	**0000**+	**1011**+
0000+	**1010**+	**1010**+	**0000**+
1100+	**0000**+	**0100**+	**1100**+
1000=	**0000**=	**1000**=	**1000**=
1111	1010	0110	1111
risultato non valido perché troncato troppo presto	*risultato valido*	*risultato valido*	*risultato valido*

Figura u98.69. Circuito combinatorio semplificato per il prodotto tra due numeri, senza dover tenere conto della gestione del segno, ma con il risultato troncato al rango di moltiplicando e moltiplicatore.

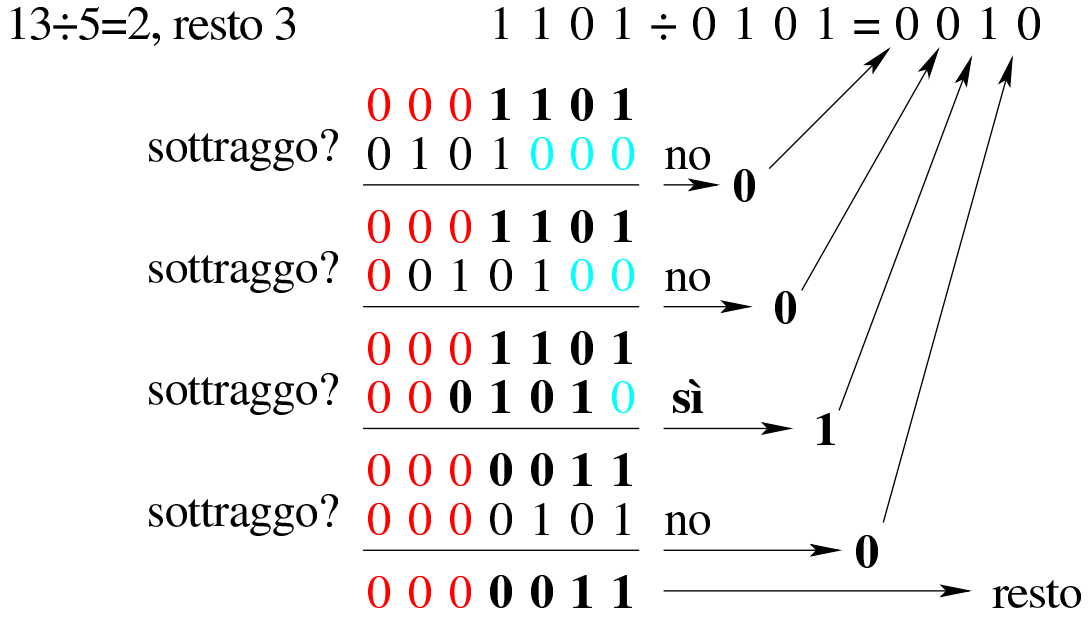


Divisione

«

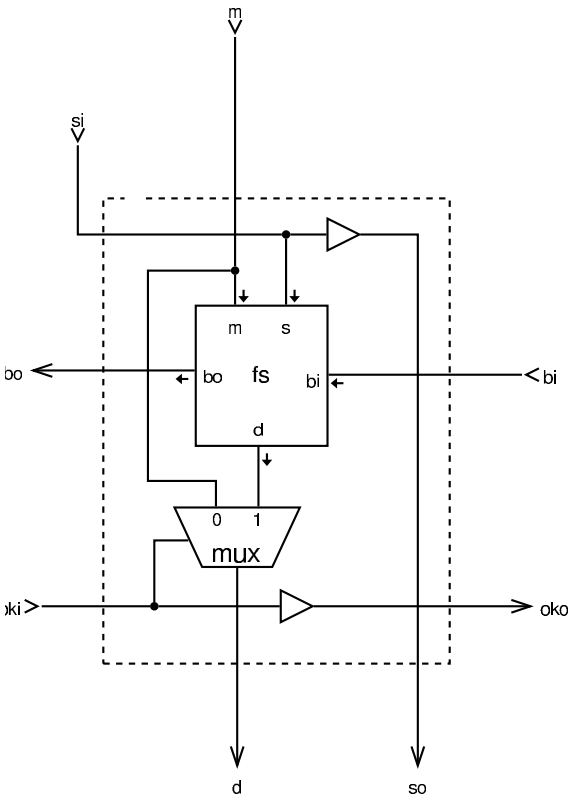
La divisione richiede la sottrazione e lo scorrimento, come si può vedere dall'esempio successivo, in cui il divisore viene confrontato con il dividendo, partendo dalle cifre più significative, sottraendo il divisore al dividendo solo quando ciò è possibile. In tal modo, il risultato intero della divisione è dato dal successo o meno di tali sottrazioni, mentre il resto è ciò che rimane dopo tutte le sottrazioni.

Figura u98.70. Procedimento usato per la divisione intera: dividendo e divisore si intendono privi di segno.



Per risolvere il problema attraverso un circuito combinatorio, si possono usare dei moduli per la sottrazione completa, da integrare con un controllo sul risultato, il quale deve essere prodotto solo se la sottrazione è ammissibile, altrimenti va riproposto il valore originale anche in uscita. Come nel caso della moltiplicazione, si possono costruire delle celle apposite.

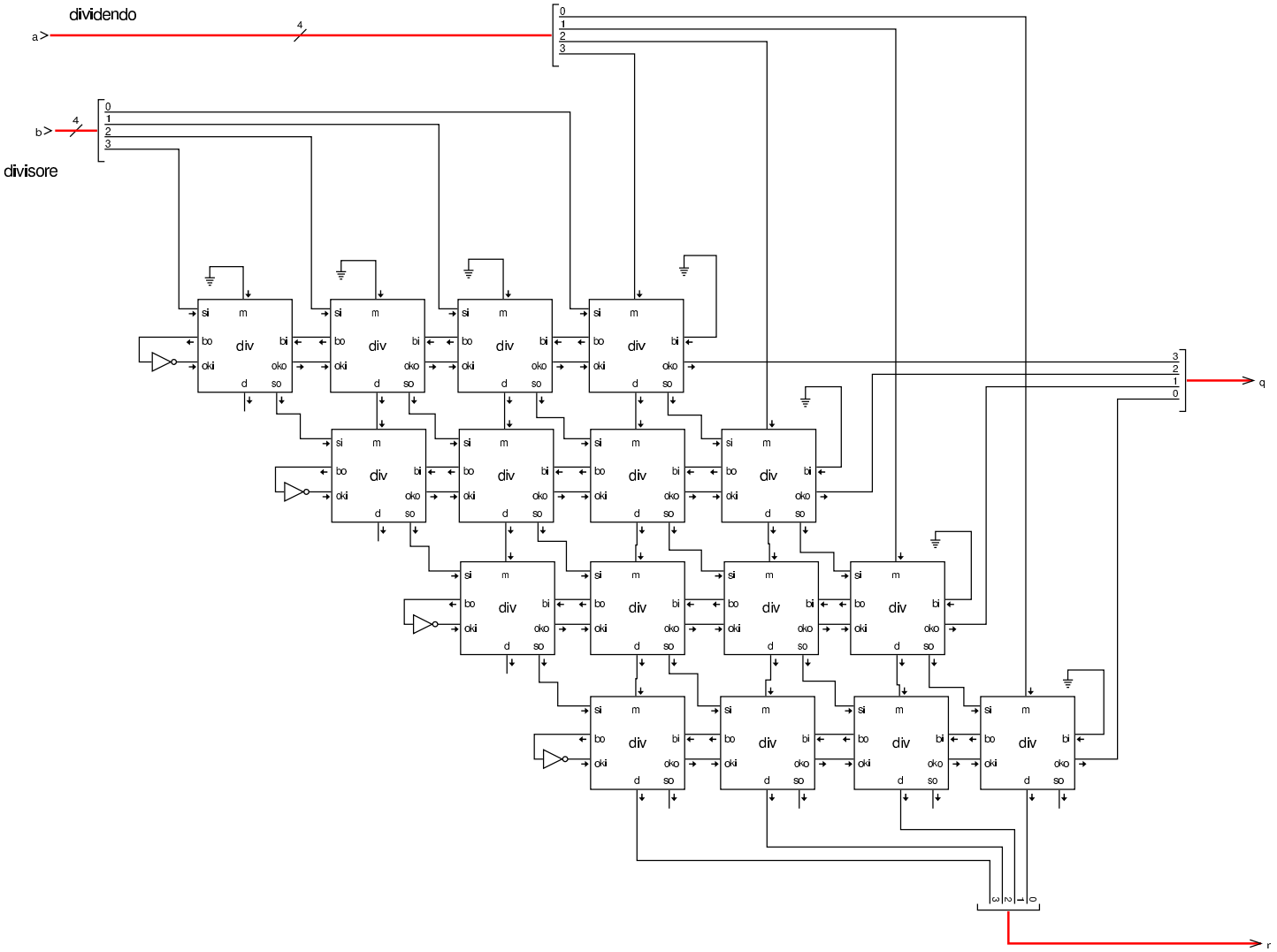
Figura u98.71. Cella usata per la sottrazione condizionata.



fs = full subtractor, sottrattore completo
m = minuendo
si = sottraendo in ingresso
so = sottraendo, copiato in uscita
bi = borrow in, richiesta di prestito di una cifra dal modulo precedente
bo = borrow out, richiesta di prestito di una cifra al modulo successivo
oki = ok in, la sottrazione è valida, ingresso
oko = ok out, la sottrazione è valida, copia in uscita

Con queste celle, dopo ogni sottrazione, si ottiene l'informazione se c'è la richiesta di un prestito o meno, dall'uscita ***bo*** (*borrow out*: se c'è tale richiesta di prestito, significa che la sottrazione non può avere luogo, quindi, tale segnale viene invertito e usato per avvisare tutte le celle di una fila che devono riproporre il valore originale anche in uscita, rinunciando alla sottrazione.

Figura u98.72. Esempio di soluzione per la divisione di valori privi di segno.

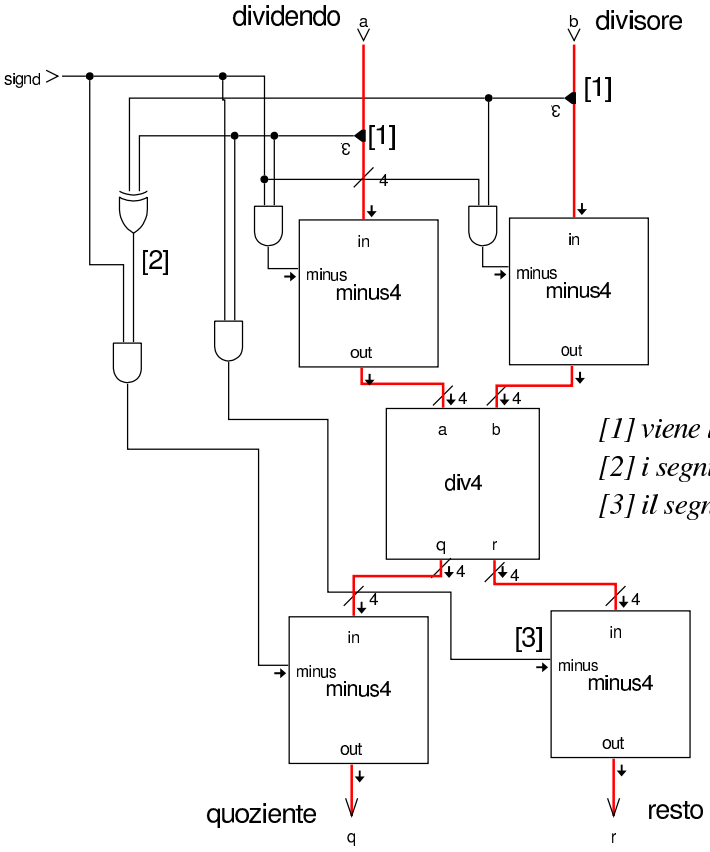


La divisione è complicata al riguardo della gestione dei valori con segno. Si pongono i casi seguenti, con cui stabilire il segno che devono avere i risultati:

Dividendo	Divisore	Quoziente	Resto
+	+	+	+
+	−	−	+
−	+	−	−
−	−	+	−

Per risolvere il problema della divisione di valori con segno, conviene aggiungere a ciò che divide valori senza segno, un controllo che inverta opportunamente i segni di dividendo, divisore, quoziente e resto, quando necessario.

Figura u98.74. Completamento del divisore con la gestione dei valori con segno: ***div4*** è il divisore di valori privi di segno; ***minus4*** si occupa di invertire il segno del valore in ingresso, se il segnale ***signd*** è attivo.

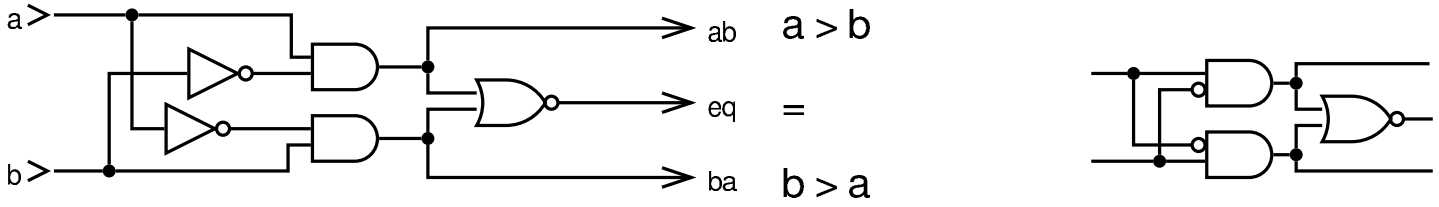


[1] viene letto il bit più significativo del gruppo, per determinarne il segno
[2] i segni di dividendo e divisore sono diversi
[3] il segno del resto dipende soltanto dal segno del dividendo

Comparazione

La comparazione tra due valori espressi in forma binaria, richiede il confronto, bit per bit, partendo dalla cifra più significativa. Da ogni confronto bit per bit, occorre sapere se sono uguali o se uno dei due sia maggiore. La figura successiva mostra la realizzazione di questo confronto tra due soli valori logici.

Figura u98.75. Confronto tra due soli ingressi; nel disegno di destra si usa una forma compatta per rappresentare gli ingressi negati delle due porte AND.



Nel confronto tra più bit, si fanno le stesse verifiche a coppia, facendo prevalere la prima differenza a partire dal bit più significativo.

Figura u98.76. Confronto tra numeri senza segno, a quattro bit.