Reti combinatorie: Codificatori

P. Marincola

(Rev. 1.2)

Come si ricorderà, i decodificatori hanno essenzialmente il compito di convertire un codice binario a n bit in un codice 1-su-m, dove $m = 2^n$. In molte circostanze appare utile disporre di una famiglia di moduli in grado di svolgere una funzione in certo senso inversa a quella espletata dai decodificatori, ossia quella di convertire un codice 1-su-m in un codice binario a n bit, dove è ancora $m = 2^n$; tali moduli verranno genericamente indicati col nome di *codificatori* (*encoder*). Una tipica applicazione del codificatore è quella in cui ci si trova in presenza di m ingressi x_0, x_1, ..., x_{m-1} tutti normalmente inattivi; nel momento in cui uno di tali ingressi, poniamo x_k, diventasse attivo, esso potrà essere identificato semplicemente determinando il valore k del suo indice. Ad esempio, una CPU può ricevere richieste di interruzione da parte di m possibili unità periferiche; quando una tale richiesta perviene alla CPU, è essenziale poter identificare univocamente l'unità periferica che origina la richiesta, in modo da poter attivare una procedura di servizio specifica per quella unità periferica.

1 Codificatori 1-su-n

In base al comportamento sopra delineato, la struttura di un modulo codificatore sarà pertanto quella illustrata in Fig. 1: quando agli $m = 2^n$ ingressi `I[0, ..., m-1]` viene applicato un codice 1-su-m, sulle n uscite `Y[n-1, ..., 0]` viene generato un numero binario di valore pari all'indice dell'unico ingresso `I` attivo: se è attivo `I[0]`, avremo `Y[n-1, ..., 0] = 00...00`, se è attivo `I[1]`, avremo `Y[n-1, ..., 0] = 00...01`, e così via, fino all'ultimo caso in cui se è attivo `I[m-1]` avremo `Y[n-1, ..., 0] = 11...11`. In generale, dunque, se l'unico ingresso attivo è `I[k]`, in uscita si avrà `Y` $= k$. Naturalmente, se nessuno degli ingressi `I` è attivo, dovrà comunque essere generato un codice in uscita, anche se non valido: adotteremo allora la convenzione secondo cui, in tale circostanza, il codice generato è tutto costituito da zeri. In altri termini, ad ogni configurazione di valore in uscita corrisponde una sola possibile configurazione degli ingressi, tranne che per la configurazione di uscita `Y[n-1, ..., 0] = 00...00`, che corrisponde sia al caso `I[n-1, ..., 0] = 00...00` (nessun ingresso è attivo), sia al caso `I[n-1, ..., 0] = 00...01` (solo l'ingresso di indice più basso è attivo).[1]

La struttura interna di un codificatore può facilmente essere ricavata a partire dalla definizione del suo comportamento. Ad esempio, un codificatore a 4 ingressi (Fig. 2*a*) ha la tavola di verità illustrata in Fig. 2*b*; le mappe di Karnaugh corrispondenti alle due uscite `Y[0-1]` e le relative equazioni appaiono rispettivamente in Fig. 3*a* e Fig. 3*b*, mentre la corrispondente realizzazione con due sole porte OR è illustrata in Fig. 3*c*.

2 Codificatori a priorità

In molte applicazioni l'assunzione che *uno solo* degli ingressi sia attivo appare eccessivamente restrittiva; ad esempio, se ciascuno degli ingressi rappresenta una "richiesta" proveniente da una di varie possibili sorgenti, è abbastanza comune la circostanza in cui possano provenire richieste da

[1] Si osservi come, dovendo specificare i valori assunti dalle componenti di un vettore di ingressi o di uscite, possiamo specificare o meno gli indici delle componenti e il relativo ordine. Se gli indici sono specificati esplicitamente, la sequenza di valori assunti dalle componenti segue rigorosamente l'ordine specificato; se invece gli indici delle componenti non vengono specificati, la sequenza di valori viene indicata assumendo un ordinamento *decrescente* degli indici. Così, scriveremo `I[0,1,2,3] = 0100` oppure, in alternativa, `I = 0010`.

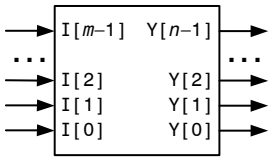
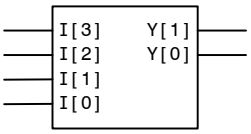


Fig. 1 Generico modulo codificatore 1-su-n.

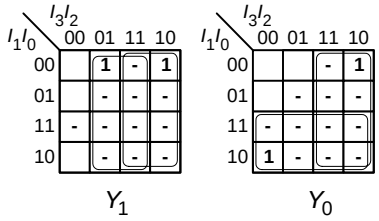


(a)

$I_3I_2I_1I_0$	Y_1Y_0	$I_3I_2I_1I_0$	Y_1Y_0
0000	00	1000	11
0001	00	1001	- -
0010	01	1010	- -
0011	- -	1011	- -
0100	10	1100	- -
0101	- -	1101	- -
0110	- -	1110	- -
0111	- -	1111	- -

(b)

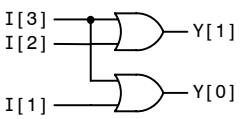
Fig. 2 Modulo codificatore a 4 ingressi (a), e la tavola di verità che ne definisce il comportamento (b).



(a)

$$Y_1 = I_3 + I_2$$
$$Y_0 = I_3 + I_1$$

(b)



(c)

Fig. 3 Realizzazione del codificatore a 4 ingressi: (a) mappe di Karnaugh, (b) equazioni, (c) realizzazione.

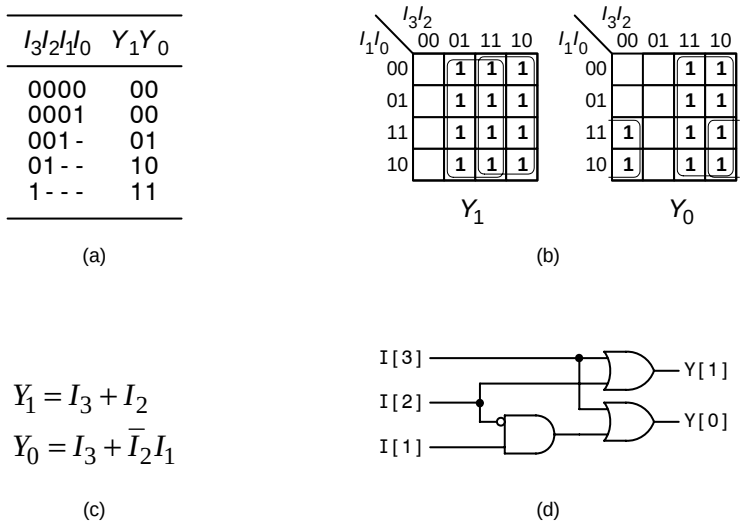


Fig. 4 Codificatore a priorità a 4 ingressi: (*a*) tavola di verità, (*b*) mappe di Karnaugh, (*c*) equazioni di uscita, (*d*) realizzazione in OR di AND.

più sorgenti nel medesimo istante.[2] In tali casi, occorrerà assegnare una *priorità* ai vari ingressi, in modo tale da privilegiare alcune richieste su altre nel caso di richieste multiple contemporanee. La definizione del comportamento del generico codificatore di Fig. 1 può allora essere modificata come segue: se uno o più ingressi `I[k`$_1$`]`, `I[k`$_2$`]`, ..., `I[k`$_p$`]` sono contemporaneamente attivi, allora in uscita `Y` viene presentato il numero $k = \max(k_1, k_2, \ldots, k_p)$, assegnando pertanto la massima priorità all'ingresso di indice *massimo* tra quelli contemporaneamente attivi; un codificatore di questo tipo prende il nome di *codificatore a priorità* (*Priority Encoder*). Si noti come:

- assegnare priorità all'indice massimo è soltanto una convenzione: ugualmente bene si sarebbe potuta adottare anche la convenzione secondo cui la priorità massima viene assegnata all'ingresso di indice *minimo* tra quelli attivi;

- la nuova definizione comprende la precedente come caso particolare: se all'ingresso viene applicato un codice 1-su-n, il codificatore a priorità si comporta esattamente come il codificatore generico discusso sopra, indipendentemente dalla convenzione adottata sulle priorità.

Il codificatore a 4 ingressi sopra esemplificato verrà adesso descritto dalla tavola di verità che appare in Fig. 4*a*, dove, in virtù della priorità assegnata ai vari ingressi, lo stato di alcuni di essi nei vari casi può diventare *don't care*: ad esempio, se l'ingresso `I[2]` è attivo mentre `I[3]` è inattivo, è del tutto indifferente lo stato in cui si trovano gli ingressi `I[0]` e `I[1]`, dal momento che `I[2]` ha priorità rispetto ad essi. Le corrispondenti mappe di Karnaugh, le equazioni di uscita e il circuito finale realizzato in OR di AND appaiono in Fig. 4*b,c,d*.

3 Espansione dei codificatori a priorità

In molte applicazioni è necessario codificare l'attività di un gran numero di ingressi, spesso talmente elevato da rendere problematico, o comunque antieconomico, il progetto *ad hoc* di un opportuno codificatore. In casi del genere, è senz'altro preferibile utilizzare più moduli codificatori con caratteristiche predefinite e interconnetterli tra loro in modo che la rete finale si comporti esattamente come un codificatore col numero desiderato di ingressi. Affinché un codificatore possa

[2] Un esempio tipico è quello delle *richieste di interruzione* (*Interrupt Request*) da parte dei sottosistemi periferici verso una CPU.

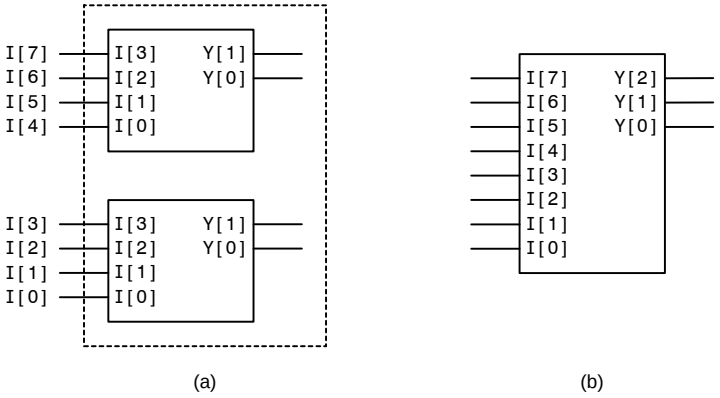


Fig. 5 Uso di due codificatori a priorità a 4 ingressi per la realizzazione di un codificatore equivalente e 8 ingressi.

essere utilizzato in reti di questo genere, è necessario che esso venga dotato di ingressi supplementari di controllo e di uscite che ne definiscano lo stato di attività, in modo da poter interagire opportunamente con gli altri codificatori della rete di espansione.

Supponiamo di voler utilizzare due codificatori a priorità a 4 ingressi (Fig. 5*a*), come quello prima analizzato, in modo da costruire un codificatore a priorità equivalente a 8 ingressi (Fig. 5*b*); gli ingressi con indici da 0 a 3 vengono applicati al modulo inferiore, mentre gli ingressi con indice da 4 a 7 vengono applicati al modulo superiore. Possiamo immediatamente osservare come le 4 uscite totali dai due moduli dovranno in qualche modo essere ricombinate in modo da generare le tre uscite finali di Fig. 5*b*.

Poiché anche in questo caso viene adottata la convenzione secondo cui viene assegnata priorità maggiore agli ingressi di indice maggiore, è chiaro che possiamo trovarci di fronte a tre circostanze:

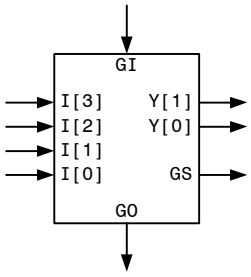
1. nessuno degli ingressi è attivo: in tal caso, le tre uscite finali `Y[2-0]` dovranno essere forzate tutte a 0;

2. è attivo uno degli ingressi applicati al modulo inferiore, mentre gli ingressi applicati al modulo superiore sono tutti inattivi: in tal caso, l'uscita generale `Y[2]` dovrà essere forzata a 0, mentre le uscite generali `Y[1-0]` dovranno essere identiche a quelle prodotte dal modulo inferiore;

3. è attivo uno degli ingressi applicati al modulo superiore; in tal caso, qualunque sia lo stato di attività del modulo inferiore, l'uscita `Y[2]` dovrà essere forzata a 1 e le uscite generali `Y[1-0]` dovranno essere identiche a quelle generate dal modulo superiore.

Affinché si abbia una qualche forma di cooperazione tra i moduli, ogni modulo deve in qualche modo trasmettere informazioni sul proprio stato al modulo successivo; per via del tipo di priorità assegnata, è chiaro che questo flusso di informazioni deve essere generato a livello del modulo più significativo, ossia quello che riceve gli ingressi con indici massimi, e instradato via via verso i moduli meno significativi; inoltre, deve essere possibile disabilitare in qualche modo la funzionalità di un modulo qualora un modulo più significativo – quindi a priorità maggiore – debba entrare in attività. Ad esempio, nel caso 3 il modulo superiore entra in attività e deve inibire il modulo inferiore, nel caso 2 il modulo superiore comunica al modulo inferiore che nessuno dei suoi ingressi è attivo, in modo che il modulo inferiore possa entrare in attività, e infine nel caso 1 entrambi i moduli rimangono inattivi.

In primo luogo, pertanto, il codificatore dovrà essere dotato di un *ingresso di abilitazione*, che, in accordo con la nomenclatura usata nei moduli codificatori commerciali, varrà chiamato *ingresso di gruppo* (*Group Input*, `GI`). Il codificatore opera normalmente come finora descritto quando `GI` = 1; se invece `GI` = 0, allora lo stato dei suoi ingressi viene ignorato e il codice generato in uscita coincide con quello prodotto quando nessun ingresso è attivo, ossia (per la convenzione adottata in



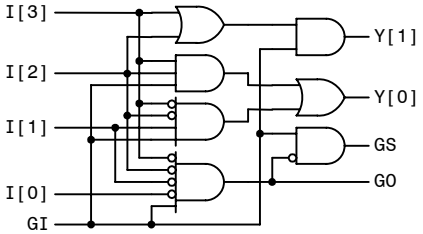
(a)

GI	$I_3 I_2 I_1 I_0$	GO	GS	$Y_1 Y_0$
0	- - - -	0	0	00
1	0000	1	0	00
1	0001	0	1	00
1	001-	0	1	01
1	01- -	0	1	10
1	1- - -	0	1	11

(b)

$$GO = GI\ \overline{I}_3\overline{I}_2\overline{I}_1\overline{I}_0$$
$$GS = GI(I_3 + I_2 + I_1 + I_0) = GI\ \overline{GO}$$
$$Y_1 = GI(I_3 + I_2)$$
$$Y_0 = GI(I_3 I_2 + \overline{I}_3\overline{I}_2 I_1)$$

(c)



(d)

Fig. 6 Modulo codificatore a priorità con ingresso di controllo e uscite di stato: (*a*) struttura del modulo, (*b*) tavola di verità, (*c*) equazioni di uscita, (*d*) realizzazione circuitale.

precedenza) sarà `Y = 00...0`. In secondo luogo, il codificatore dovrà essere dotato di una *uscita di gruppo* (*Group Output*, `GO`), che si attiva quando il modulo è abilitato ma *non* entra in attività, essendo inattivi tutti i suoi ingressi `I`. È anche opportuno, per ragioni che vedremo più avanti, avere disponibile un'ulteriore uscita da ciascun modulo, detta *selezione di gruppo* (*Group Select*, `GS`) che diventi attiva quando e soltanto quando il modulo è abilitato (`GI` = 1) e contemporaneamente uno almeno dei suoi ingressi `I` è attivo, ossia quando le uscite `Y` prodotte dal modulo stesso sono da considerarsi valide.[3] Il modulo a 4 ingressi così modificato si presenta allora come in Fig. 6*a*, ed ha il comportamento descritto dalla tavola in Fig. 6*b*; la sua implementazione è illustrata in Fig. 6*d*.

L'interconnessione tra i vari moduli si realizza allora (Fig. 7) connettendo `GO` di un dato modulo a `GI` del modulo immediatamente meno significativo; l'ingresso `GI` al modulo più significativo assume la funzione di abilitazione generale della rete, mentre l'uscita `GO` dal modulo meno significativo, quando attiva, starà ad indicare che nessuno degli ingressi alla rete è attivo, e che quindi il codice generato in uscita (tutti zeri) è da considerarsi non valido, o comunque non come conseguenza di `I'[0]` = 1. Dal momento che ad ogni istante al più un solo modulo è attivo, e le uscite `Y` da un modulo non attivo sono tutte a 0, le uscite generali `Y'[0-1]` si possono ricavare semplicemente mettendo tra loro in OR le corrispondenti uscite dai singoli moduli. L'uscita generale più significativa `Y'[2]` dovrà valere 1 se il modulo superiore è attivo, mentre dovrà valere 0 in tutti gli altri casi; essa può pertanto essere fatta coincidere con l'uscita `GS` dal modulo più significativo. Infine, l'uscita generale `GS'`, avendo la funzione di indicare che un modulo è attivo e che quindi l'uscita presenta un codice valido, potrà essere generata mediante OR delle uscite `GS` dai due moduli.

La struttura può essere estesa a un numero maggiore di moduli; tuttavia, in tal caso la generazione di alcune uscite generali diventa più complessa che nel caso precedente, al punto da richiedere l'impiego di un ulteriore modulo. Supponiamo di voler realizzare un priority encoder

[3] Il meccanismo di trasmissione delle informazioni tra un modulo e il succesivo presenta molte analogie col protocollo di *daisy-chain*, utilizzato ad esempio per determinare la priorità delle richieste di Interrupt Service alla CPU in una catena di unità periferiche in un sistema a microprocessore.

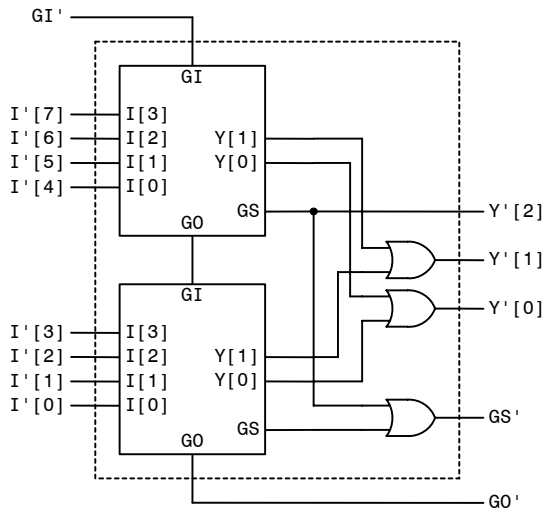


Fig. 7 Realizzazione di un priority encoder a 8 ingressi mediante moduli codificatori a 4 ingressi.

a 16 ingressi utilizzando solo moduli codificatori a 4 ingressi; estendendo a questo caso le stesse considerazioni dell'esempio precedente, possiamo osservare come le uscite generali più significative `Y'[3-2]` siano legate esclusivamente all'attività dei singoli moduli. Possiamo allora applicare le uscite `GS` da ciascun modulo a un priority encoder supplementare (Fig. 8), dedicato a generare appunto le uscite più significative. (Si osservi come la logica usata per generare `Y'[2]` nell'esempio di Fig. 7 equivale all'impiego di un codificatore a due soli ingressi.)

È il caso di osservare come, all'aumentare del numero di moduli, debba necessariamente aumentare parallelamente il numero di ingressi degli OR che producono le uscite generali meno significative. Per ovviare a questo inconveniente, si può modificare il comportamento del modulo elementare in modo che le uscite `Y` siano di tipo *tri-state*, e che vengano forzate nello stato ad alta impedenza (che verrà indicato con Z nelle tavole di verità) quando il modulo non è attivo, ossia quando `GI` $= 0$ oppure quando i suoi ingressi I sono tutti a zero – in pratica, quando `GS` $= 0$. Introducendo questa variante, la tavola di verità del priority encoder a 4 ingressi viene ad essere quella mostrata in Fig. 9*a*, mentre la corrispondente modifica circuitale è illustrata in Fig. 9*b*.

L'uso di priority encoder con uscite tri-state rende molto più semplice l'espansione, eliminando la necessità di usare OR per combinare le uscite dai vari moduli. Ad esempio, l'espansione mostrata in Fig. 8 può essere semplificata come in Fig. 10. A titolo di ulteriore esempio, in Fig. 11 viene mostrato come realizzare un priority encoder a 512 ingressi e 9 uscite mediante moduli codificatori a 8 ingressi organizzati su 3 livelli, dove ciascun livello provvede a generare 3 bit di uscita. I 64 moduli del primo livello vengono raggruppati in blocchi da 8 encoder, ciscuno dei quali afferisce tramite le uscite `GS` a un encoder del secondo livello. A loro volta, gli 8 encoder del secondo livello afferiscono all'unico encoder del terzo livello. Le uscite `Y[0-2]` vengono prodotte dai moduli al primo livello, le uscite `Y[3-5]` dai moduli al secondo livello e infine le uscite `Y[6-8]` dal modulo al terzo livello.

Questa tecnica può essere generalizzata a piacimento. Volendo realizzare una rete equivalente a un priority encoder con n uscite e $m = 2^n$ ingressi, gli n bit di uscita vengono partizionati in p campi, non necessariamente di identiche dimensioni. Il numero di campi determina il numero di livelli della rete di espansione, mentre il numero di bit in un dato campo determina le caratteristiche dei priority encoder disposti sul livello corrispondente. Le uscite corrispondenti al campo meno significativo vengono generate dagli encoder di primo livello, ossia quelli direttamente connessi agli ingressi generali, mentre le uscite corrispondenti al campo più significativo vengono generate dall'unico encoder al p-esimo livello. Ad esempio, per $n = 7$ bit di uscita e di conseguenza $m = 2^7 = 128$ ingressi, potremmo partizionare i 7 bit in vari modi:

- 3 campi da 2, 2, 3 bit: la struttura corrispondente avrà tre livelli, il primo costituito da

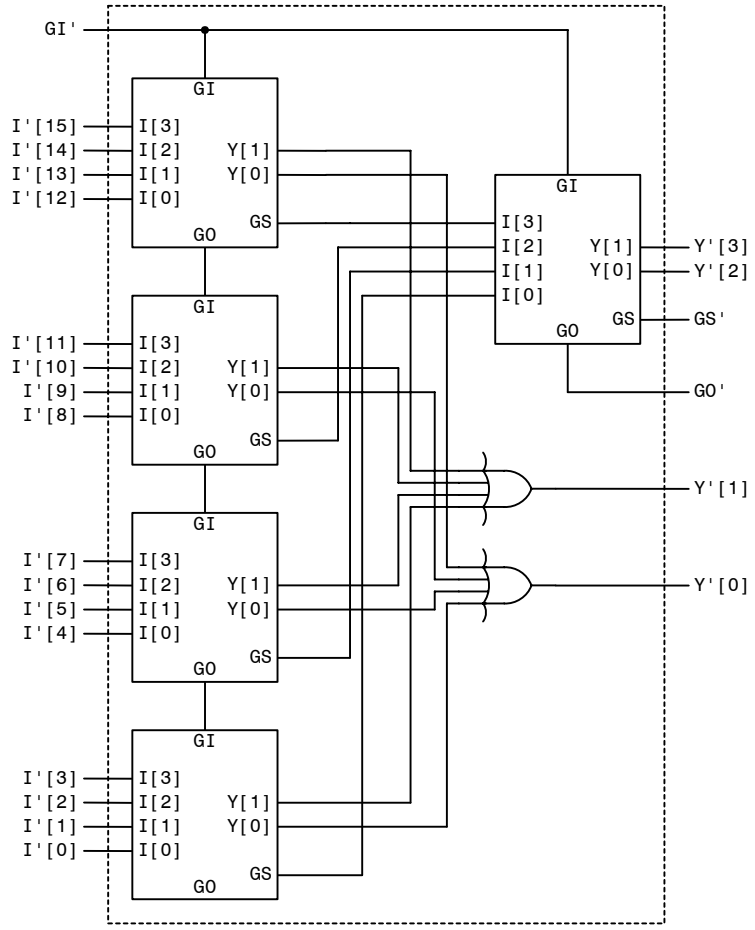
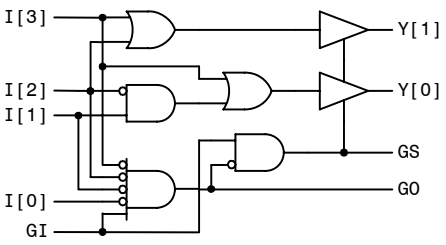


Fig. 8 Realizzazione di un priority encoder a 16 ingressi con moduli codificatori a 4 ingressi.

GI	$I_3I_2I_1I_0$	GO	GS	Y_1Y_0
0	- - - -	0	0	ZZ
1	0000	1	0	ZZ
1	0001	0	1	00
1	001 -	0	1	01
1	01 - -	0	1	10
1	1 - - -	0	1	11

(a)



(b)

Fig. 9 Priority encoder a 4 ingressi con uscite tri-state: (a) tavola di verità, (b) realizzazione circuitale.

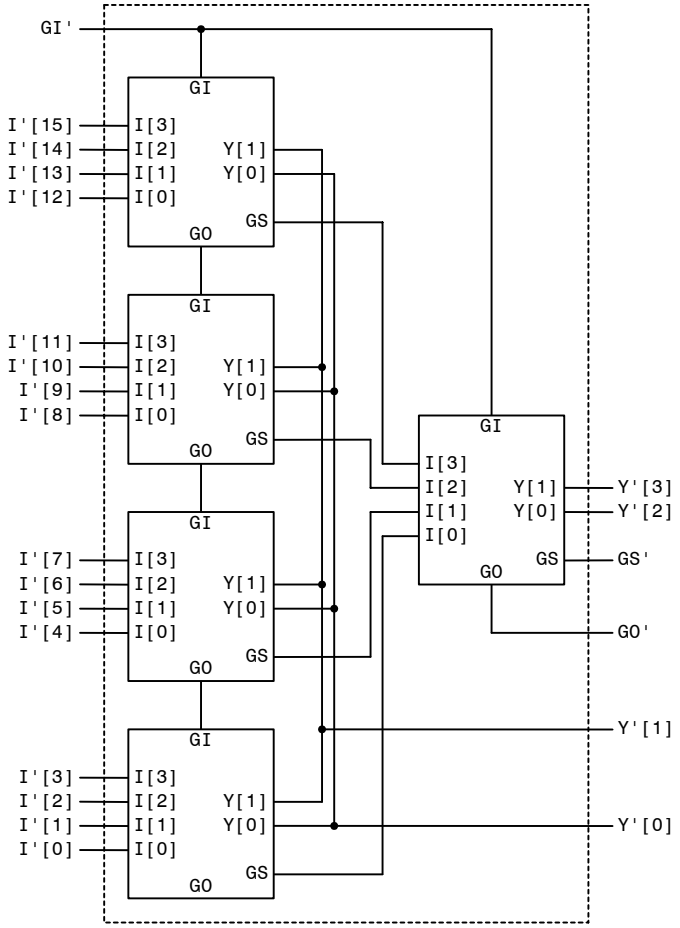


Fig. 10 Realizzazione di un priority encoder a 16 ingressi mediante moduli codificatori a 4 ingressi con uscite tri-state.

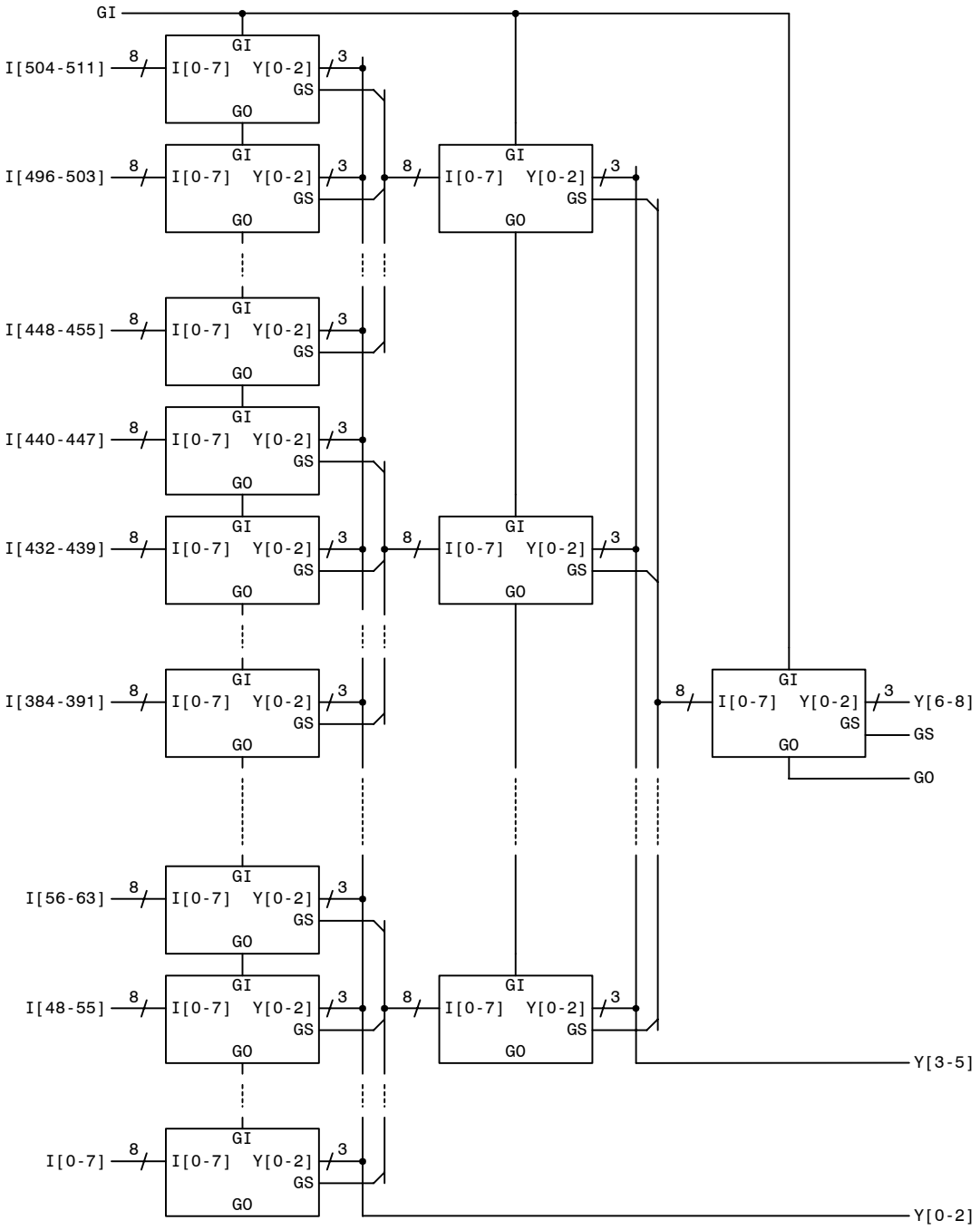


Fig. 11 Realizzazione di un priority encoder a 512 ingressi e 9 uscite mediante moduli codificatori a 8 ingressi.

$128/4 = 32$ moduli da 4 ingressi ciascuno, il secondo da $32/4 = 8$ moduli da 4 ingressi ciascuno, e il terzo da un singolo modulo da 8 ingressi;

- 3 campi da 3, 2, 2 bit: la struttura corrispondente avrà tre livelli, il primo costituito da $128/8 = 16$ moduli da 8 ingressi ciascuno, il secondo da $16/4 = 4$ moduli da 4 ingressi ciascuno, e il terzo da un singolo modulo da 4 ingressi;
- 2 campi da 4, 3 bit: la struttura corrispondente avrà due livelli, il primo costituito da $128/16 = 8$ moduli da 16 ingressi, il secondo da un solo modulo da 8 ingressi;

e via dicendo. Se n_k (con $1 \leq k \leq p$) è il numero di bit del k-esimo campo, otterremo una struttura a p livelli, dove il primo livello è costituito da 2^{n-n_1} moduli da 2^{n_1} ingressi, il secondo livello è costituito da $2^{n-n_1}/2^{n_2} = 2^{n-n_1-n_2}$ moduli da 2^{n_2} ingressi, e così via; in generale, il k-esimo livello è costituito da $2^{n-n_1-n_2-\ldots-n_k}$ moduli a 2^{n_k} ingressi; l'ultimo livello, ricordando che $n_1 + n_2 + \ldots + n_p = n$, è costituito infine da $2^{n-n_1-n_2-\ldots-n_p} = 1$ modulo da 2^{n_p} ingressi.

Esercizi

Esercizio 1 Determinare la tavola di verità di un codificatore 1-su-8 e ricavarne una realizzazione circuitale.

Esercizio 2 Realizzare un codificatore 1-su-16 utilizzando moduli codificatori 1-su-4.

Esercizio 3 Determinare la tavola di verità di un priority encoder a 8 ingressi e ricavarne una realizzazione circuitale.

Esercizio 4 Determinare la tavola di verità di un priority encoder a 4 ingressi con priorità sull'indice minimo, e ricavarne una realizzazione circuitale.

Esercizio 5 Determinare la tavola di verità di un priority encoder a 8 ingressi con priorità sull'indice minimo, e ricavarne una realizzazione circuitale.

Esercizio 6 Realizzare un priority encoder a 8 ingressi con priorità sull'indice minimo utilizzando un priority encoder a 8 ingressi con priorità sull'indice massimo.

Esercizio 7 Ricavare la struttura di espansione corrispondente alla Fig. 7 utilizzando moduli codificatori con priorità sull'indice minimo.

Esercizio 8 Ricavare la struttura di espansione corrispondente alla Fig. 8 utilizzando moduli codificatori con priorità sull'indice minimo.

Esercizio 9 Determinare la tavola di verità di un priority encoder a 4 ingressi con priorità sull'indice minimo e uscite tri-state, e ricavarne una realizzazione circuitale.

Esercizio 10 Ricavare la struttura di espansione corrispondente alla Fig. 10 utilizzando moduli codificatori con priorità sull'indice minimo.

Esercizio 11 Ricavare la struttura di espansione corrispondente alla Fig. 11 utilizzando moduli codificatori con priorità sull'indice minimo.