Codifica delle informazioni e Macchine Elementari (Richiami)

Nuovo Corso di Calcolatori Elettronici I

Dipartimento di Informatica e Sistemistica
Università degli Studi di Napoli "Federico II"

Argomenti

- Analisi e Sintesi di Sistemi
- Codifica (cenni)
- Rappresentazione Decodificata
- Transcodificatore
- Controllo di Errore
- Decodificatore
- Codificatore
- Muliplexer Lineare e Indirizzabile
- Demultiplexer Lineare e Indirizzabile

Analisi e Sintesi di un sistema

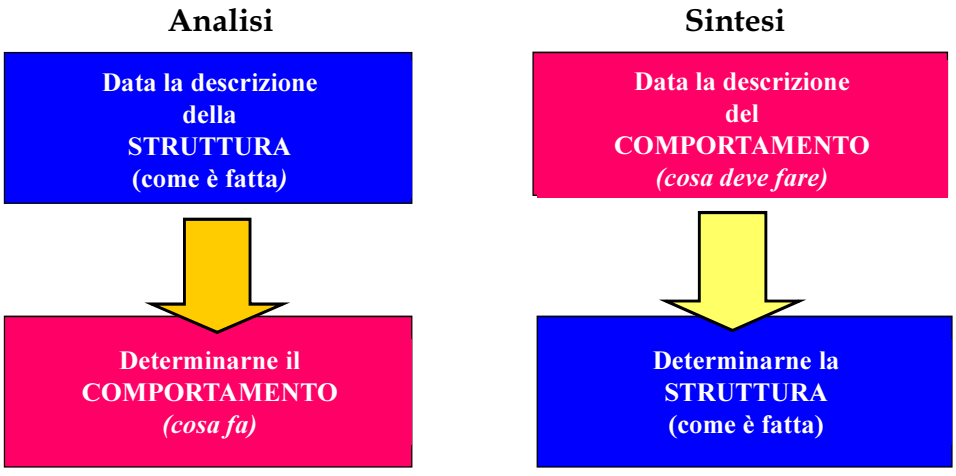
Per ***analisi*** di un sistema si intende l'individuazione delle relazioni di causa/effetto tra i segnali di ingresso e uscita, attraverso l'esame di una rappresentazione schematica dei suoi componenti elementari e dei collegamenti che li interconnettono, ovvero:

- ***data la rappresentazione schematica del sistema, individuarne il comportamento.***

Per ***sintesi*** di un sistema si intende l'individuazione dei componenti e delle interconnessioni necessarie per realizzarlo seguendo la preassegnata specifica funzionale:

- ***data la specifica funzionale individuarne la struttura.***

Analisi e Sintesi di un sistema



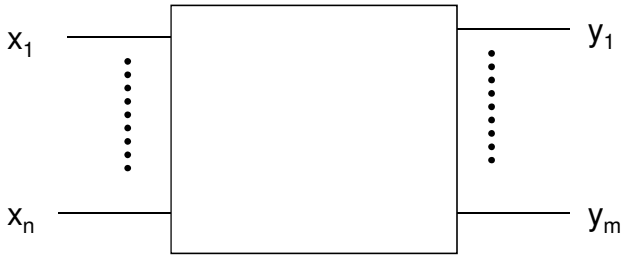
Tassonomia dei circuiti digitali

- I circuiti digitali possono essere classificati in due categorie
- Circuiti combinatori
  - Il valore delle uscite ad un determinato istante dipende unicamente dal valore degli ingressi in quello stesso istante.
- Circuti sequenziali
  - Il valore delle uscite in un determinato istante dipende sia dal valore degli ingressi in quell'istante sia dal valore degli ingressi in istanti precedenti (concetto di tempo)
  - Per definire il comportamento di un circuito sequenziale è necessario tenere conto della storia passata degli ingressi del circuito (concetto di stato)

Macchine combinatorie (1/ 4)

Reti logiche con *n* ingressi $x_1, x_2, \ldots, x_n$ e *m* uscite $y_1, y_2, \ldots, y_m$ che realizzano la corrispondenza:

$$y_1 = f_1(x_1, x_2, \ldots, x_n)$$
$$\ldots\ldots$$
$$y_m = f_m(x_1, x_2, \ldots, x_n)$$



Macchine combinatorie (2/ 4)

- Una macchina combinatoria è una rete logica con *n* ingressi (x_1, x_2, …, x_n) ed *m* uscite (y_1, y_2, …, y_m) ed è tale che ad ogni insieme di valori degli ingressi corrisponde un preciso insieme di valori delle uscite
- Il comportamento di una rete combinatoria *n*x*m* può essere descritto tramite:
  - una tabella di verità in cui viene specificato il valore dell'uscita per ognuna delle possibili combinazioni dei valori degli ingressi
  - m funzioni booleane, una per ogni uscita, ciascuna delle quali esprime il valore della corrispondente variabile di uscita in funzione delle n variabili di ingresso

Macchina combinatoria (3/ 4)

- In una macchina combinatoria i valori delle uscite dipendono esclusivamente dai valori degli ingressi
  - macchina combinatoria ideale: tale dipendenza è istantanea
  - macchina combinatoria reale: presenza di ritardo tra l'istante in cui c'è una variazione in uno degli ingressi e l'istante in cui l'effetto di questa variazione si manifesta sulle uscite
- E' importante notare come
  - ciascuna y_i può essere decomposta in funzioni componenti
  - due distinte y_i possono contenere una identica funzione componente
- Ciò comporta, ad esempio, una potenziale diminuzione di porte elementari rispetto ad una realizzazione indipendente delle y_i

Macchina combinatoria 4/ 4

- Mentre ad ogni rete combinatoria corrisponde un'unica tabella di verità, ad una tabella di verità possono corrispondere più reti combinatorie
- Nel procedimento di sintesi ci possiamo quindi porre particolari obiettivi come ad esempio:
  - utilizzare esclusivamente circuiti elementari di un certo tipo
  - progettare un circuito che abbia il minimo numero di porte logiche

Codifica
(cenni)

Codifica delle informazioni

- Codici a lunghezza fissa
  - Codice Fiscale
  - Codice di Avviamento Postale
- Codici a lunghezza variabile
  - Alfabeto Morse
  - Numeri Telefonici

Codifica in Binario

- Dato un insieme di N elementi, il numero minimo m di bit necessario alla codifica è dato da

$$m \geq \{[\log_2 N]\}$$

- Codici Completi e Incompleti
- Codifica Ridondante

Codici Ridondanti

$K^m >= N$

- N = 8, K = 2 → m = 3, l = 3, (l = m) → **Codice Completo & Non Ridondante**
- N = 7, K = 2 → m = 3, l = 3, (l = m) → **Codice InCompleto & Non Ridondante**
- N = 7,8, K = 2 → m = 3, l = 4, (l > m) → **Codice InCompleto & Ridondante**

Rappresentazione di Codici mediante tabelle

$T = (x_1, x_2, \ldots, x_n)$ alfabeto origine

$E = (a_1, a_2, \ldots, a_k)$ alfabeto destinazione

Es. Codice BCD (Binary Coded Decimal)

Dato

	3	2	1	0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Parola Codice

Rappresentazione Decodificata

- L'insieme ha cardinalità N
- Le parole codice hanno lunghezza m = N
- Ad ogni parola codice è associato un solo bit 1
- Motivazioni

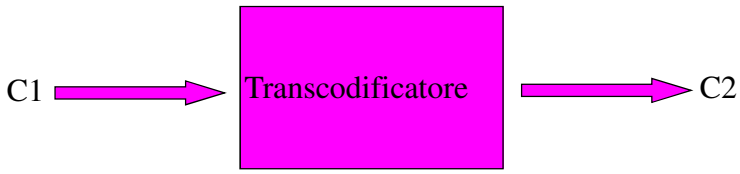
Esempio di rappresentazione Decodificata

Dato	Codice BDC	Codice Decodificato
0	0000	1000000000
1	0001	0100000000
2	0010	0010000000
3	0011	0001000000
…	…	….
…		
9	1001	0000000001

Transcodificatore

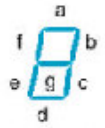
Transcodificatore

- In alcune circostanze si può verificare che lo stesso dato sia rappresentato mediante codici diversi.
- Detti C1 e C2 due codici definiti sullo stesso insieme (cioè costituito dallo stesso numero di parole codice), si dice TRANSCODIFICATORE una macchina che consente di associare a ciascuna parola del codice C1 la corrispondente parola del codice C2.
  - Transcodificatore per visualizzatore a 7 segmenti.



Transcodificatore per visualizzatore a 7 segmenti

- Uno degli indicatori visivi più comuni è l'***indicatore a 7 segmenti***.
- Ogni simbolo è formato da sette segmenti ognuno dei quali è un Led che può essere acceso da un segnale digitale.
- Un ***BCD-To-Seven-Segment-Decoder*** riceve in ingresso un simbolo decimale in BCD e genera l'appropriata uscita selezionando i segmenti che devono essere accesi per mostrare su display il simbolo decimale.



(a) Segment designation


(b) Numeric designation for display

Transcodificatore per visualizzatore a 7 segmenti

- Le 7 uscite le indichiamo con (a,b,c,d,e,f,g) selezionando i corrispondenti segmenti. Si hanno:
  - *4 input*: A B C D.
  - *7 output*: a b c d e f g.
- La tabella di verità sarà pertanto ⇒

BCD Input				Seven-Segment Decoder						
A	B	C	D	a	b	c	d	e	f	g
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
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
All other inputs				0	0	0	0	0	0	0

CONTROLLO DI ERRORE

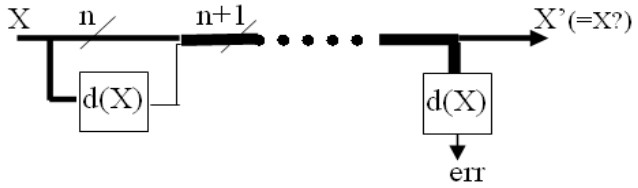
Controllo di errori - 1

Bit di parità e controllo

- Nella trasmissione o nella memorizzazione di un'informazione la "perdita" di un bit (trasformazione da 0 a 1 o viceversa) è la più diffusa causa di errore
- Le reti di parità/disparità si usano per effettuare un semplice controllo di errore
- In fase d trasmissione (o memorizzazione) si aggiunge un bit tale che il numero complessivo di bit "1" sia a parità assegnata (p.e. pari)- Il bit si dice **di parità**
- In fase di ricezione (o lettura) una rete di parità (o disparità) controlla la parità e, se diversa da quella assegnata, segnala l'errore, ponendo ad esempio ad 1 un segnale di controllo err ed avendosi dunque:
  - *err=1* → **CERTEZZA** della sua presenza
  - *err=0* → **PROBABILITA'** di assenza di errori
- Il sistema si basa sulla massimizzazione di questa probabilità

Controllo di errori - 3

Disparità e controllo di errore



- In partenza (o in scrittura), d(X) calcola il bit di parità
- In arrivo (o lettura), d(X) calcola il segnale err
- Si ricorda:
  - ***err=0 → probabilmente è X'=X***
  - ***err=0 → certamente è X'≠X***

  (potrebbe anche essere X'=X perché l'errore è sul bit di parità: trascuriamo questo caso per sicurezza e semplicità)

Rilevazione degli errori

- Natura degli errori
- Codici ridondanti
- Rilevazione e correzione degli errori

- Esempio:
  - Bit di parità: si aggiunge un bit in modo che il numero di 1 sia pari:

    00101101 → 00101101 0

    01101000 → 01101000 1
  - L'errore su un bit viene rilevato (non corretto)

    011010001 → 011000001

Rilevazione e Correzione degli errori

- Rilevazione (mediante bit di parità) e correzione mediante aggiunta di informazioni ridondanti (bit di parità + checksum)

10110010	0
11100100	0
01010110	0
00000111	1
10000110	1
10000001	0

Check Sum

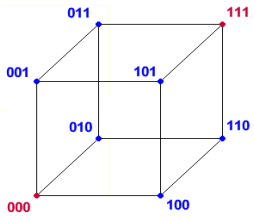
Bit di parità

10110010	0
11100100	0
01000110	0
00000111	1
10000110	1
10000001	0

Controllo di errori - 2

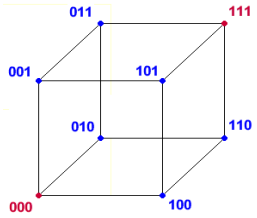
I riferimenti teorici – distanza di Hamming

- L'informazione da trasmettere o memorizzare è un "codice a lunghezza fissa"
- **Distanza di Hamming** D = numero di bit diversi tra due parole codice



- **Distanza minima** di un codice: la distanza minima fra 2 sue parole-codiice
- **Sindrome di errore**: la parola-codice non appartiene al codice

Controllo di errori - 3

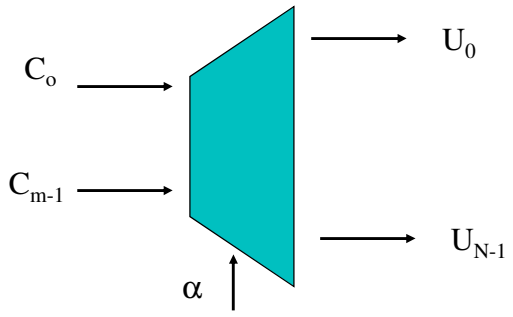


- **I riferimenti teorici-Teorema di Hamming:**
- Un codice a distanza minima
  - d+1 può individuare gli errori simultanei su d bit;
  - 2c+1 può correggere gli errori simultanei su c bit.

Decoder ed Encoder

Decodificatore (decoder)

- Un decodificatore è una macchina che riceve in ingresso una parola codice (C) e presenta in uscita la sua rappresentazione decodificata (linee U_0, … U_{N-1})

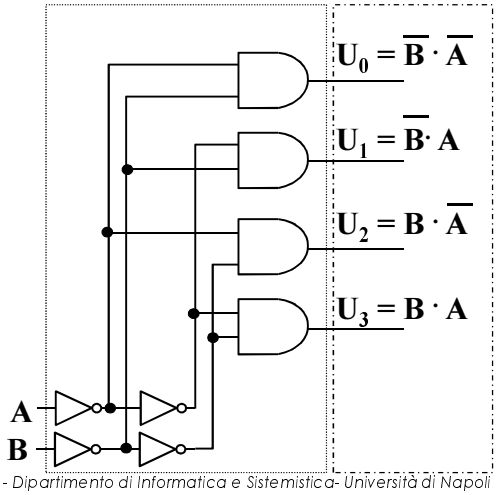


Sintesi della trascodifica da binario a 1 su N

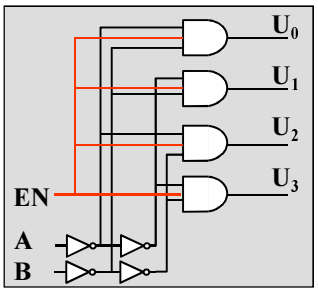
Abbiamo un insieme sorgente di 4 elementi. Sono necessari 2 bit (A,B) per codificarlo. Nella rappresentazione decodificata servono invece 4 bit ($U_0U_1U_2U_3$).

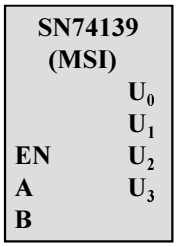
Esempio: Transcodifica 2:4

	B	A	U_0	U_1	U_2	U_3
1	0	0	1	0	0	0
2	0	1	0	1	0	0
3	1	0	0	0	1	0
4	1	1	0	0	0	1



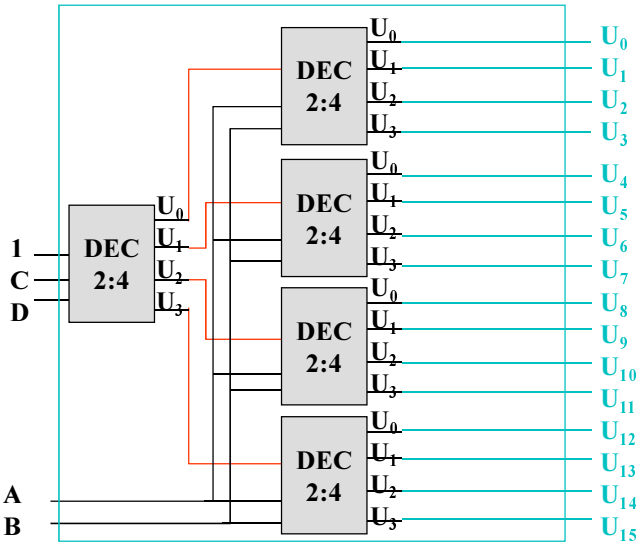
Il circuito integrato DECODER





Quando EN=1 (segnale di abilitazione), vale 1 l'uscita il cui pedice, in decimale, corrisponde al numero binario in ingresso (A bit di minor peso**).**

Composizione modulare di Decoder 4:16



Esempio di utilizzo di Decoder (1/ 2)

• I decoder sono utilizzati in moltissime applicazioni.

• Un esempio è la selezione dell'input/output da parte del processore.

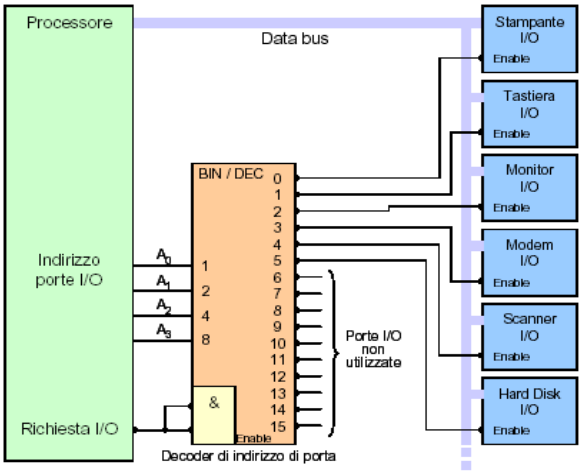
• Ogni porta di I/O possiede un indirizzo (univoco).

• Quando il processore vuole comunicare con una determinata periferica esso produce il codice dell'indirizzo della porta di I/O al quale la periferica è connessa.

• L'indirizzo (binario) della porta è fornito in input al decoder il quale attraverso il suo output abilita la corrispondente porta di I/O.

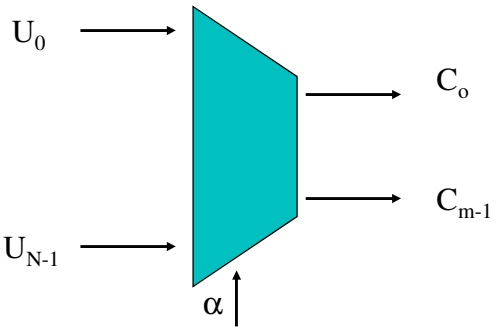
Esempio di utilizzo di Decoder (2/ 2)

- Come mostrato in Figura l'informazione binaria è trasferita su un **bus di dati**, che è un set di fili paralleli.
- **Esempio**: un bus di 8-bit consiste in 8 fili paralleli che trasferiscono un byte di dati alla volta.
- Il bus è connesso a tutte le porte I/O, ma ogni dato in transito passerà attraverso la porta soltanto se questa è abilitata dal decoder.



Codificatore (Encoder)

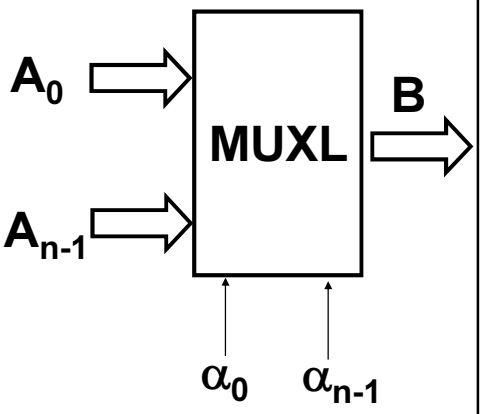
- Un codificatore è una macchina che riceve in ingresso una rappresentazione decodificata (linee U_0, … U_{N-1}) e fornisce in uscita la parola codice associata a U_i se U_i=1 (più in generale se è attivo).



Multiplexer e Demultiplexer

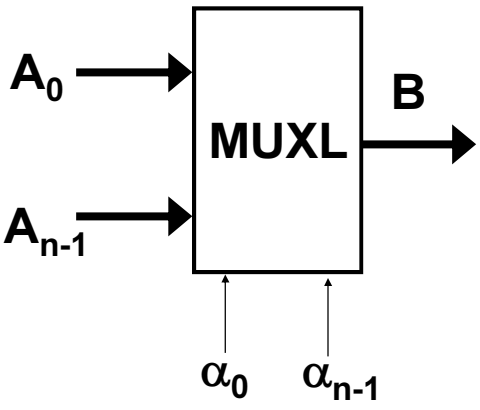
Multiplexer lineare

- Un *Multiplexer lineare (*ML) è una macchina con:
  - n ingressi-dati ($A_0,\ldots,A_{n-1}$)
  - n segnali binari di selezione ($\alpha_0,\ldots,\alpha_{n-1}$), **dei quali al più uno è attivo**
  - una uscita-dati B, che assume
    - valore A_i se è attivo α_i
    - neutro se nessuna delle selezioni è attiva



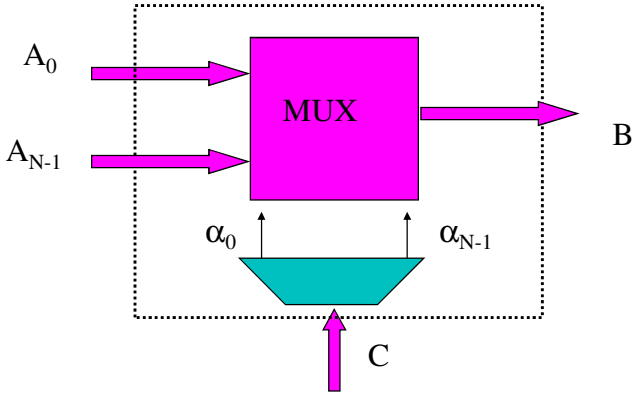
Multiplexer binario

- Se i dati A_i e B sono quelli che viaggiano su un bus …
  - si parla di multiplexer o demultiplexer
- Se i dati A_i e B sono semplici bit …
  - si parla di multiplexer o demultiplexer binario



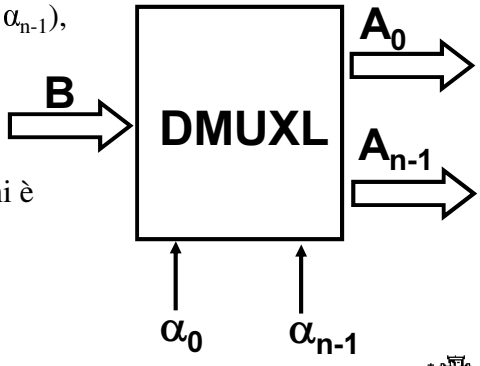
Multiplexer Indirizzabile

- E' un Multiplexer Lineare i cui segnali di abilitazione sono collegati con le uscite di un decodificatore (decoder)



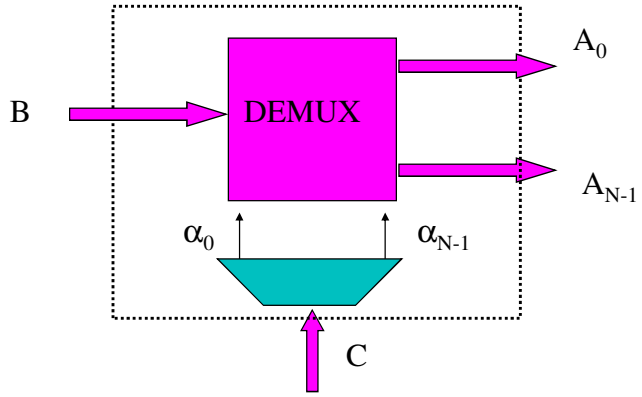
Demultiplexer lineare

- Un *Demultiplexer Lineare* (DL) è una macchina con:
  - 1 ingresso-dati B
  - n segnali binari di selezione (α_0,…, α_{n-1}), dei quali al più uno è attivo
  - n uscite-dati (A_0,…,A_{n-1}), con
    - A_i=B se è attivo α_i
    - neutro se nessuna delle selezioni è attiva



Demultiplexer Indirizzabile

- E' un Demultiplexer Lineare i cui segnali di abilitazione sono collegati con le uscite di un decodificatore

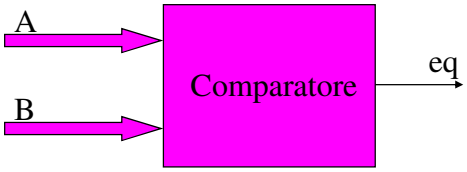


COMPARATORI

Comparatore

- Spesso accade che occorra confrontare due dati A e B in quanto l'unità di controllo dell'architettura deve proseguire con algoritmi differenti a seconda che i due dati siano eguali o diversi.

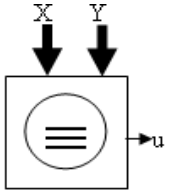
- *Il comparatore è una macchina che ha in ingresso due dati (A, B) ed in uscita un segnale booleano* ***eq*** *che è 1 se è A=B.*



Comparatori

Eguaglianza bit a bit in parallelo

- Confronto tra due stringhe di bit (parole-codice o numeri)



$$u = \prod_i (X_i \odot Y_i)$$

Soluzione parallela:

• per n bit, n porte EQ ed una AND ad n ingressi

COMPARATORI BIT-SERIALE (per un singolo bit)

- Per $A_n = B_n$ la tabella della verità è:

A_n	B_n	E_n
0	0	1
0	1	0
1	0	0
1	1	1

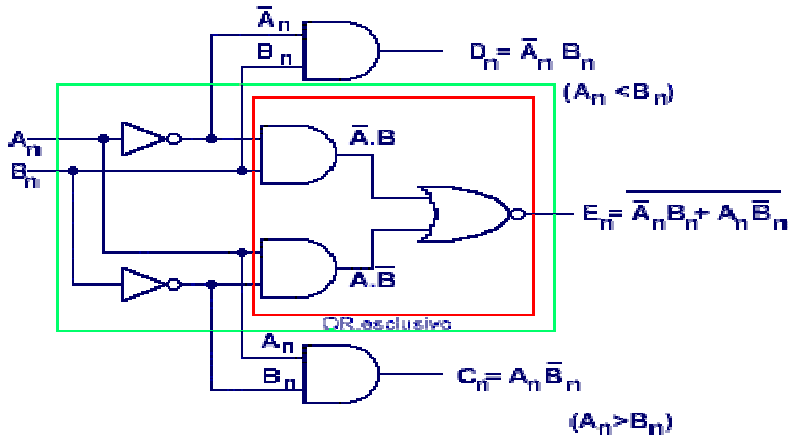
- Con il pedice n si vuole distinguere l'n-esimo bit in una parola o numero presentato con più bit in serie nel tempo. Da cui:

$$E_n = \bar{A}_n \bar{B}_n + A_n B_n = \overline{\bar{A}_n B_n + A_n \bar{B}_n}$$

- Cioè E_n è alto quando A_n e B_n sono uguali. Mentre A_n è minore di B_n se $D_n = \bar{A}_n B_n = 1$, viceversa A_n è maggiore di B_n se $C_n = A_n \bar{B}_n = 1$.

COMPARATORI BIT-SERIALE (per un singolo bit)

- Il circuito comparatore per il singolo bit-serie nel suo insieme ha tre uscite ed è schematizzato come in figura:



Comparatore binario

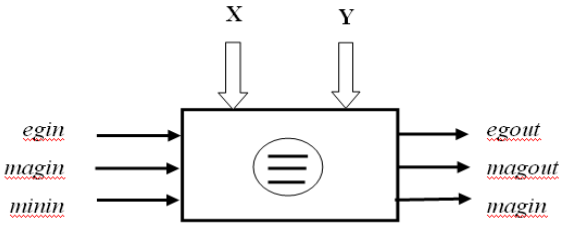
begin

egout : = *(X = Y)* AND *egin* ;

magout : = *(X>Y)* OR *(X = Y)* AND *magin* ;

minout : = *(X < Y)* OR *(X = Y)* AND *minin*

end;



Comparatore binario

- Deriviamo l'espressione di *egout*
  - partiamo dall'eguaglianza X=Y:

$$(X = Y) = \prod_{i=0}^{n-1} (X_i \equiv Y_i) = \prod_{i=0}^{n-1} (\overline{X_i \oplus Y_i}) = \overline{\sum_{i=0}^{n-1} (X_i \oplus Y_i)}$$

ricordiamo che la XOR è il complemento della EQ

  - applicando De Morgan si ricavano due possibili espressioni:

$$egout := \overline{\left[\sum_{i=0}^{n-1} (X_i \oplus Y_i)\right]} + \overline{egin} \qquad egout := \left[\prod_{i=0}^{n-1} (X_i \equiv Y_i) \cdot egin\right]$$

Comparatore binario

- La realizzazione in logica a due livelli *and-or (nand)* non è conveniente
- La funzione (X=Y) non presenta mintermini adiacenti
- Es.:

Y \ X	00	01	11	10
00	1			
01		1		
11			1	
10				1

Mappa di Karnaugh per $f = (X=Y)$ con X e Y ingressi di due variabili ciascuno

Comparatore binario

- Deriviamo l'espressione di *magout*
- Osserviamo che, per le singole cifre binarie, si ha

$$X_i > Y_i \quad \Leftrightarrow \quad X_i \cdot \overline{Y_i} = 1$$

- Quindi, per ottenere il confronto tra ingressi a più cifre è possibile "partire" confrontando le cifre da sinistra e procedendo via via verso destra fino a trovare eventuali differenze

$$magout := X_{n-1} \cdot \overline{Y_{n-1}} +$$
$$(X_{n-1} \equiv Y_{n-1}) \cdot (X_{n-2} \cdot \overline{Y_{n-2}}) +$$
$$(X_{n-1} \equiv Y_{n-1}) \cdot \ldots \cdot (X_1 \equiv Y_1) \cdot X_0 \cdot \overline{Y_0} + (X = Y) \cdot magin$$

Comparatore binario

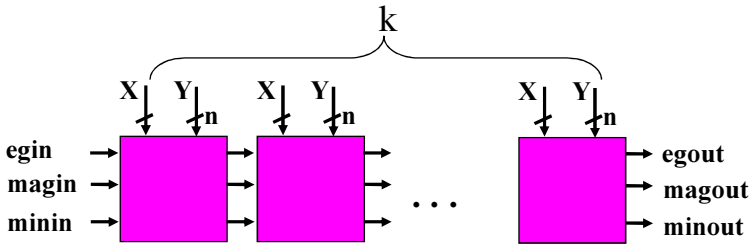
- Per ottenere la terza uscita *minout*, se necessaria, si può procedere analogamente a *magout* oppure semplicemente osservare che

$$\text{minout} := \overline{egout} \cdot \overline{magout}$$

- Si noti che la componente $X_i \equiv Y_i$ è ripetuta nelle tre funzioni e può essere calcolata una sola volta

Comparatore binario

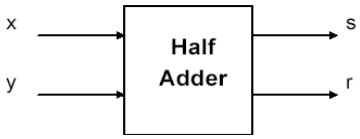
- Un confronto tra stringhe di k bit può essere effettuato con un unico comparatore se $k \leq n$ (numero di bit di X e Y, *ingressi del comparatore)*
- Altrimenti, possono essere collegati in catena $\{[k/n]\}$ comparatori, come in figura



Adder

Half Adder

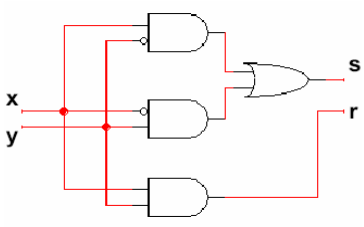
- Scrivere la tabella di verità relativa alla somma di due addendi. Riportare sia l'uscita somma che quella riporto. Disegnare anche il circuito equivalente.

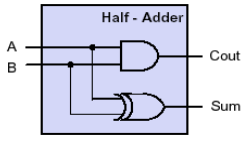


x	y	s	r
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$S = \overline{X}Y + X\overline{Y} = X \text{ xor } Y$

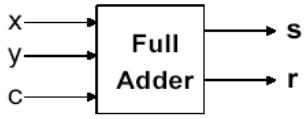
$r = X\,Y$



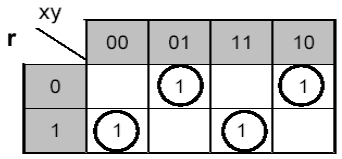


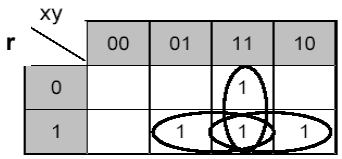
Full Adder (1/ 2)

- Scrivere la tabella di verità relativa alla somma di due addendi più il riporto entrante. Riportare sia l'uscita somma che quella riporto.



X	Y	r	S	R
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



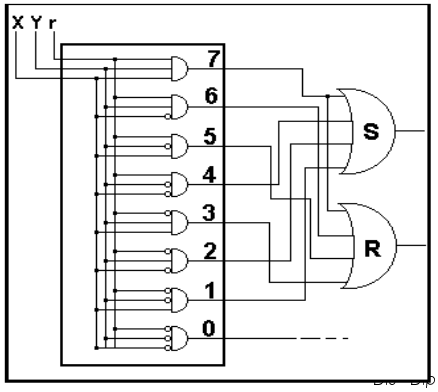


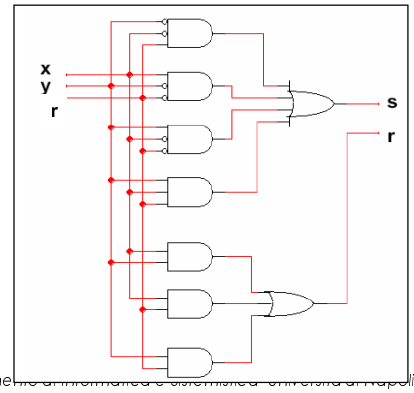
Full Adder (2/ 2)

Le uscite valgono quindi:

$$S = (\overline{X}\,\overline{Y}\,r) + (\overline{X}\,Y\,\overline{r}) + (X\,\overline{Y}\,\overline{r}) + (X\,Y\,r)$$

$$R = \overline{X}Yr + X\overline{Y}r + XY\overline{r} + XYr = XY + Yr + Xr$$





Addizionatore binario

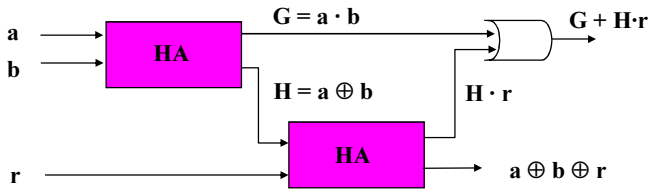
- E' possibile isolare il fattore **(a ⊕ b)**
- Rielaborando le precedenti espressioni è quindi possibile ottenere le seguenti espressioni per l'addizionatore completo:

$$S = (a \oplus b) \oplus r = H \oplus r$$
$$R = ab + r(a \oplus b) = G + rH$$

Addizionatore binario

- Pertanto, un addizionatore completo può essere ottenuto a partire da due semiaddizionatori:

$$S = (a \oplus b) \oplus r = H \oplus r$$
$$R = ab + r(a \oplus b) = G + rH$$

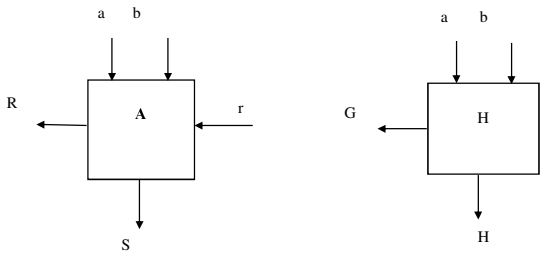


Addizionatore binario: il riporto

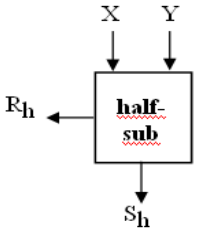
- Le diverse componenti dell'espressione ottenuta per l'addizionatore binario assumono un significato particolare:
  - **G = a·b** "**riporto generato**": indica la creazione di un riporto all'interno dell'addizionatore binario
  - **P = H = a⊕b** "**riporto propagato**": indica se, in presenza di un riporto in ingresso, lo stesso verrò propagato in uscita
  - Il riporto in uscita può quindi essere espresso come **R=G+Pr**

Addizionatore binario: conclusione

- Addizionatore completo (full-adder)
  - ha il riporto entrante
- Semi-addizionatore (half-adder)
  - non include il riporto entrante

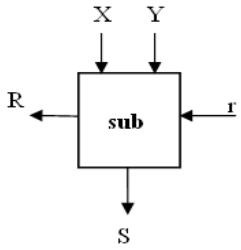


Sottrattore binario



X	Y	S_h	R_h
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

$(S_h,R_h) = X - Y$



X	Y	r	S	R
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$(S,R) = X - Y - r$

Sottrattore binario

- Per ottenere la sottrazione può essere ripetuto lo stesso procedimento usato per definire la struttura degli addizionatori
- Per il semisottrattore si avrà:

$$S_h = X\overline{Y} + \overline{X}Y = X \oplus Y$$

$$R_h = \overline{X}Y$$

- Per il sottrattore completo :

$$S = \overline{X}\,\overline{Y}r + \overline{X}Y\overline{r} + X\overline{Y}\,\overline{r} + XYr = X \oplus Y \oplus r$$

$$R = \overline{X}\,\overline{Y}r + \overline{X}Y\overline{r} + \overline{X}Yr + XYr = \overline{X}Y + \overline{X}r + Yr$$

Addizionatore binario

- Per il semiaddizionatore valgono le eguaglianze

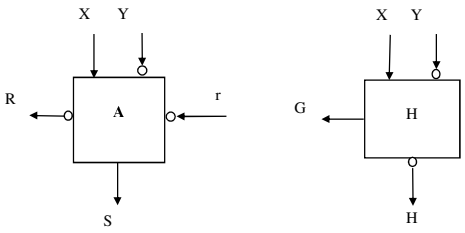
$$H = a \oplus b = d(a,b) = a\bar{b} + \bar{a}b$$
$$G = a \cdot b$$

- Similmente per l'addizionatore completo valgono le eguaglianze

$$S = a \oplus b \oplus r = d(a,b,r) = \bar{a}\bar{b}r + \bar{a}b\bar{r} + a\bar{b}\bar{r} + abr$$
$$R = \bar{a}br + a\bar{b}r + ab\bar{r} + abr = ab + br + ar = ab + r(a+b)$$

Sottrattore binario

- Un sottrattore può essere ottenuto a partire da un addizionatore negando opportunamente alcuni ingressi ed uscite, come mostrato in figura
- Provatelo come esercizio.



Addizionatori binari

- Ulteriori condizioni di interesse:

$$n_i = \overline{r_i}$$

non-riporto

Indica assenza di riporto in ingresso

$$K_i = \overline{a_i} \cdot \overline{b_i}$$

Riporto "killed"

Indica che, indipendentemente dalla presenza di un riporto entrante, il riporto in uscita sarà comunque zero

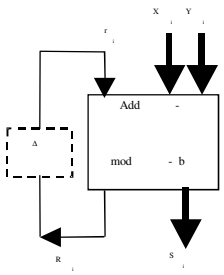
$$N_i = K_i + P_i \cdot n_i$$

Propagazione del non-riporto

Indica assenza di riporto in uscita

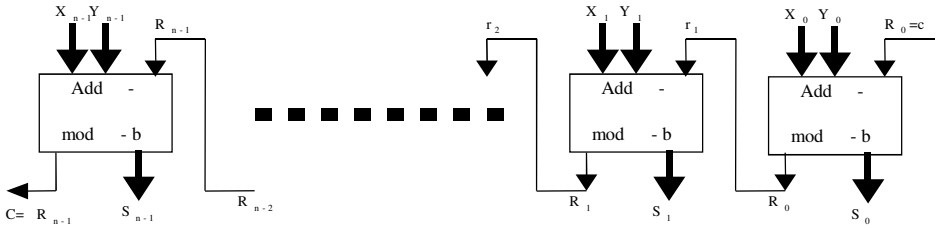
Addizionatori seriali

- Usa un unico addizionatore operante sulla singola cifra
- Opera in momenti successivi su cifre diverse degli addendi
- Richiede un blocco "con memoria"
- E' lento rispetto ad addizionatori che lavorano in parallelo sulle diverse cifre degli addendi



Addizionatori paralleli

- Opera sulle cifre degli addendi in parallelo
- …anche se il riporto deve propagarsi attraverso l'intera struttura
- Richiede un numero maggiore di risorse rispetto all'addizionatore seriale



Addizionatori paralleli

- Gli addizionatori ottenuti collegando in cascata ***n*** addizionatori di cifra sono anche chiamati addizionatori a propagazione del riporto (*carry-ripple* o *carry-propagate*)
- ε = tempo di risposta di uno stadio
- Allo stadio i, il riporto uscente o è **generato** o è **ucciso** o è **propagato**
- Tempo di ritardo complessivo: **Limite inferiore** ε (in tutti gli stadi il riporto è generato o ucciso)
- Tempo di ritardo complessivo: **Limite superiore** $n\varepsilon$ (un riporto entrante nel primo stadio che è propagato in tutti gli stadi)
- Tempo di ritardo complessivo = $k\varepsilon$ ($k \leq n$), dove k è la più lunga catena di condizioni di propagazione.

Addizionatori con anticipo del riporto

- Normalmente chiamati addizionatori *carry lookahead*
- Per ogni stadio *i*, dal *(k+1)*-esimo al *(k+j)*-esimo, r_i si ottiene direttamente dai bit degli addendi X ed Y e dal riporto entrante nella catena (invece che dal riporto uscente R_{i-1})

$$r_{k+1} = G_k + r_k P_k$$
$$r_{k+2} = G_{k+1} + G_k P_{k+1} + r_k P_k P_{k+1}$$
....
$$r_{k+j} = G_{k+j-1} + G_{k+j-2} P_{k+j-1} + ... + G_k P_{k+1} ... P_{k+j-1} + r_k P_k P_{k+1} ... P_{k+j-1}$$

Addizionatori carry lookahead

$$r_{k+j} = G_{k+j-1} + G_{k+j-2} P_{k+j-1} + ... + G_k P_{k+1} ... P_{k+j-1} + r_k P_k P_{k+1} ... P_{k+j-1}$$

r_{k+j} è alto se è verificata la condizione di generazione nell'ultimo stadio

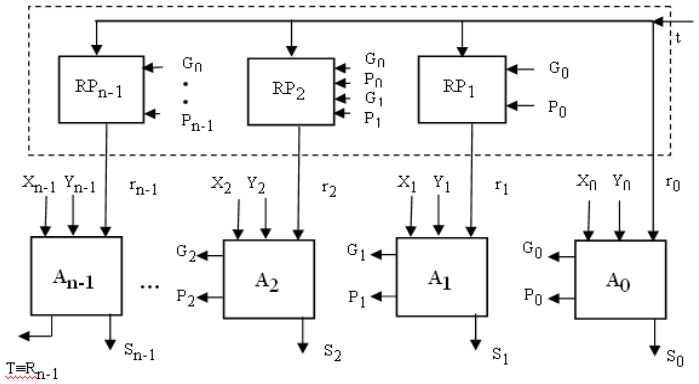
....
....

…oppure se è verificata la condizione di generazione G_k e se questa viene propagata dagli stadi dal (k+1)-esimo fino all'ultimo

…oppure è pari al riporto entrante r_k se questo viene propagato in tutti gli stadi

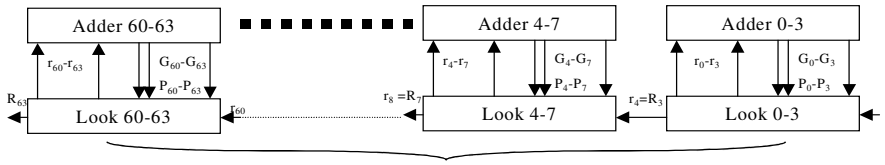
Addizionatori carry lookahead

- L'espressione precedente può essere realizzata nella maniera riportata in figura



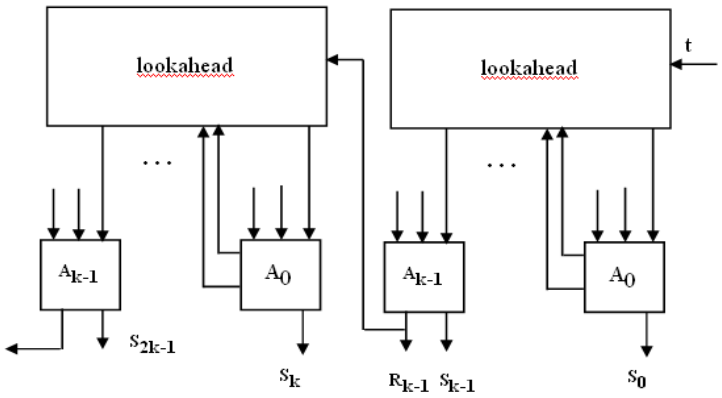
Addizionatori carry lookahead

- L'idea di base negli addizionatori carry lookahead è quella di calcolare la relazione tra r_k ed r_{k+j} separatamente per ogni gruppo di cifre $k+1 \ldots k+j$
- Una rete a livello superiore valuta la propagazione del carry tra *gruppi* di cifre
- Ad esempio, per un adder a 64 bit suddiviso in blocchi di quattro cifre (bit), la parte di "*look*" lungo cui si propaga il riporto è lunga 64/4=16 stadi



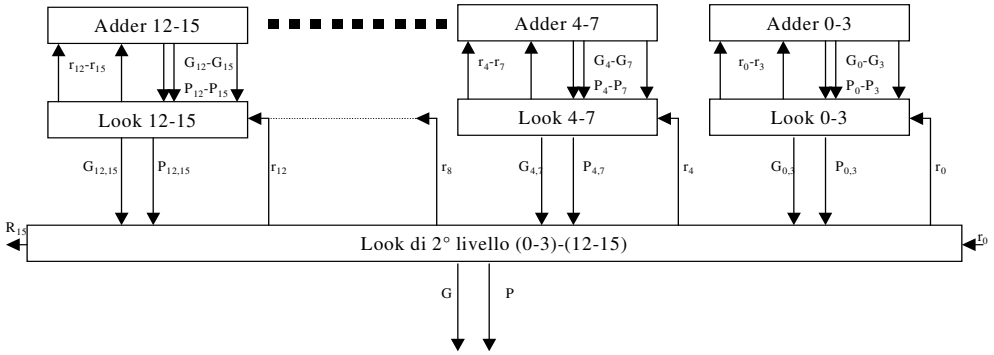
16 blocchi

Addizionatori carry lookahead



Addizionatori carry lookahead

- La rete di lookahead può a sua volta essere realizzata a più livelli, secondo lo stesso principio visto prima



Supporto Didattico

Capitolo 3 e Capitolo 7, Macchine per l'elaborazione delle Informazioni

Fadini – De Carlini

e

Dispense a cura di B. Fadini

Domande?