

Livello Register

Caratteristiche:

- l'unità di dato manipolata è la parola
- i componenti usati nel progetto sono blocchi (combinatori o sequenziali) per la memorizzazione o la trasformazione di parole
- in alcuni casi può essere utile adottare un'algebra booleana operante su vettori di bit; le funzioni utilizzate sono quindi

$$x: (\mathbb{B}^m)^n \text{ @ } \mathbb{B}^m$$

1

M. Sonza Reorda - a.a. 2001/02

Linguaggi di Descrizione

Servono per descrivere un progetto per vari scopi:

- documentazione
- simulazione
- verifica
- valutazione delle prestazioni
- sintesi.

Sono detti *Register Transfer Language* in quanto il loro elemento essenziale è l'istruzione di tipo *register transfer*:

$$X \leftarrow f(X_1, X_2, \dots, X_n)$$

Tali linguaggi hanno spesso notevoli somiglianze con i linguaggi di programmazione (ad es. hanno il costrutto condizionale).

2

M. Sonza Reorda - a.a. 2001/02

Esempio

```
declare register A(0:7), M(0:7), Q(0:7), COUNT(0:2)
declare bus INBUS(0:7), OUTBUS(0:7)
BEGIN:      A ← 0, COUNT ← 0;
INPUT:     M ← INBUS,
           Q ← INBUS;
ADD:       A(0:7) ← A(1:7)+M(1:7)×Q(7);
RIGHTSHIFT: A(0) ← 0,A(1:7).Q ← A.Q(0:6),
TEST:      COUNT ← COUNT+1;
           IF COUNT ≠ 7 THEN GOTO ADD,
FINISH:    A(0) ← M(0) EXOR Q(7), Q(7) ← 0;
OUTPUT:    OUTBUS ← Q;
           OUTBUS ← A;
END;
```

3

M. Sonza Reorda - a.a. 2001/02

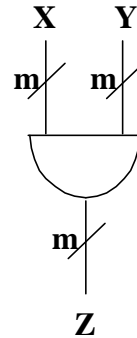
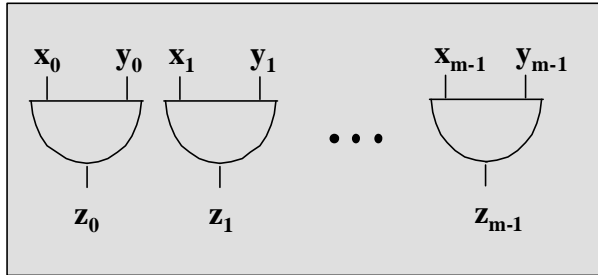
Componenti

- **Porte Logiche (operanti su parole)**
- **Multiplexer**
- **Decodificatori e Codificatori**
- **Array Logic**
- **Moduli Aritmetici (ALU, Sommatore, etc.)**
- **Registri**
- **Contatori**
- **Bus**
- **Memorie.**

4

M. Sonza Reorda - a.a. 2001/02

Porte Logiche operanti su Parole

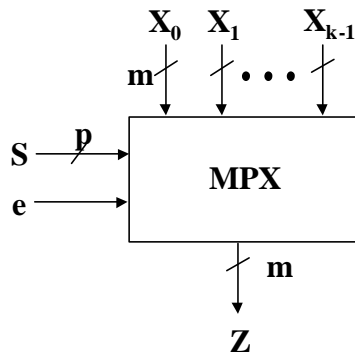


5

M. Senza Reorda - a.a. 2001/02

Multiplexer

Servono per connettere una fra k fonti di dato X_i ad una destinazione, a seconda del valore di p segnali di selezione S ; in genere $k=2^p$.



6

M. Senza Reorda - a.a. 2001/02

Multiplexer: funzione

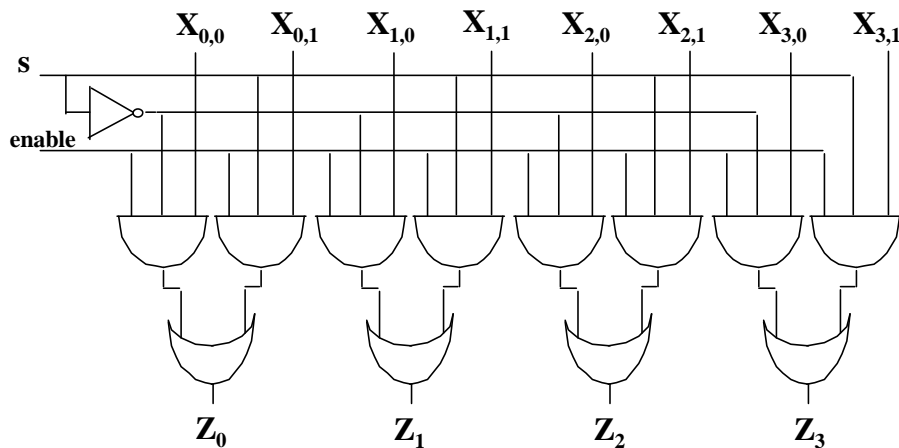
La tavola della verità di un multiplexer $2^2 \times 1$ con ingressi di dato X_0 e X_1 , ingresso di controllo S ed uscita Z è la seguente:

S	e	Z
0	1	X_0
1	1	X_1
-	0	0

7

M. Sonza Reorda - a.a. 2001/02

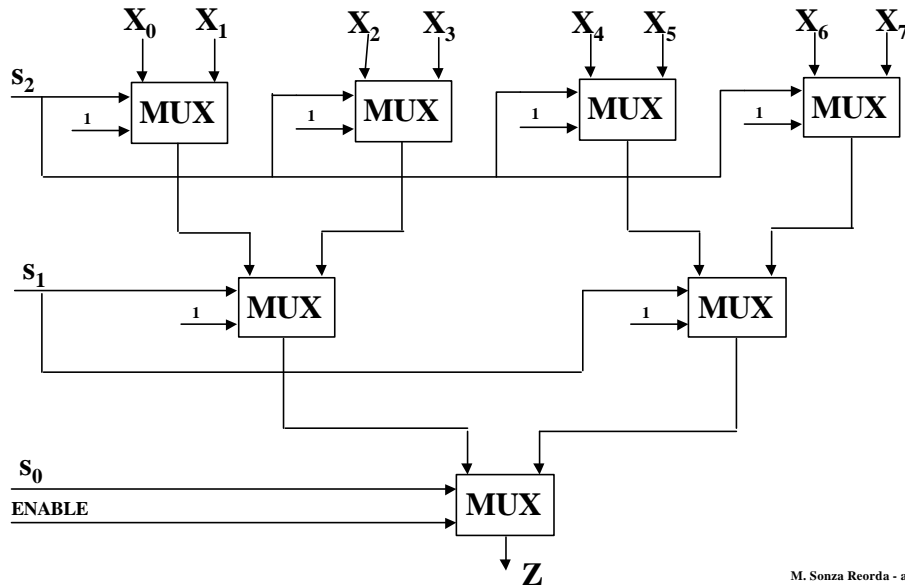
Multiplexer: realizzazione



8

M. Sonza Reorda - a.a. 2001/02

Multiplexer: connessione a cascata



9

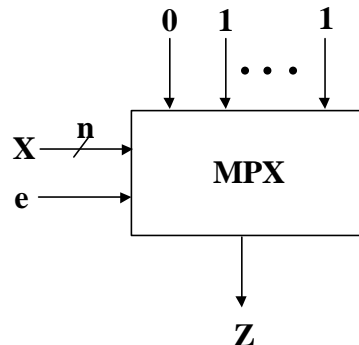
M. Sonza Reorda - a.a. 2001/02

Multiplexer: uso nella sintesi

Possono essere considerati un blocco fondamentale, in quanto permettono l'implementazione di una qualsiasi funzione combinatoria.

$$Z=f(x_0,x_1,\dots,x_{n-1})$$

Ad ogni ingresso di dato corrisponde un minterm.



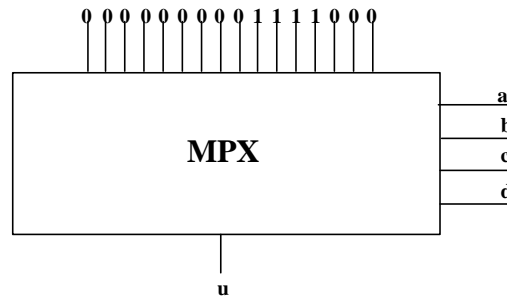
10

M. Sonza Reorda - a.a. 2001/02

Sintesi tramite Multiplexer: esempio

Si supponga di voler sintetizzare un circuito con 4 ingressi a, b, c, d la cui uscita u valga 1 quando $8 < (abcd) < 13$.

Una possibile implementazione è

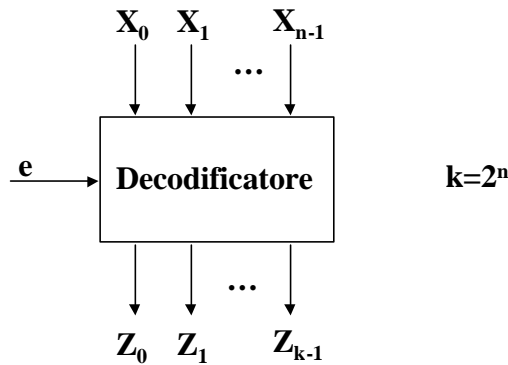


11

M. Senza Reorda - a.a. 2001/02

Decodificatori

Hanno n linee di ingresso e 2^n linee di uscita; di queste è attiva solo quella di indice corrispondente al valore applicato in ingresso.



12

M. Senza Reorda - a.a. 2001/02

Decodificatori: funzione

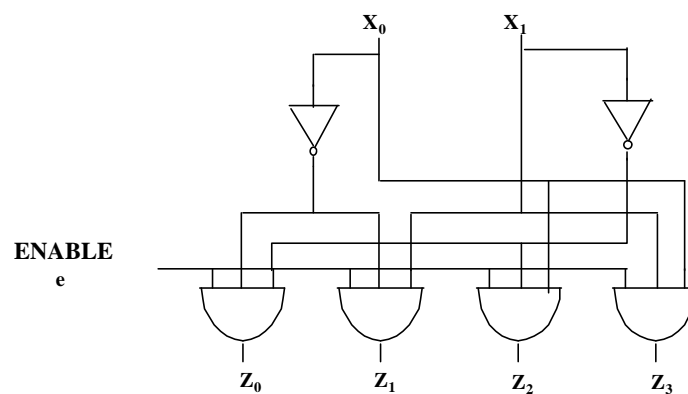
La tavola della verità di un decodificatore $2^2 \times 4$ è la seguente:

X	e	Z
00	1	0001
01	1	0010
10	1	0100
11	1	1000
-	0	0000

13

M. Sonza Reorda - a.a. 2001/02

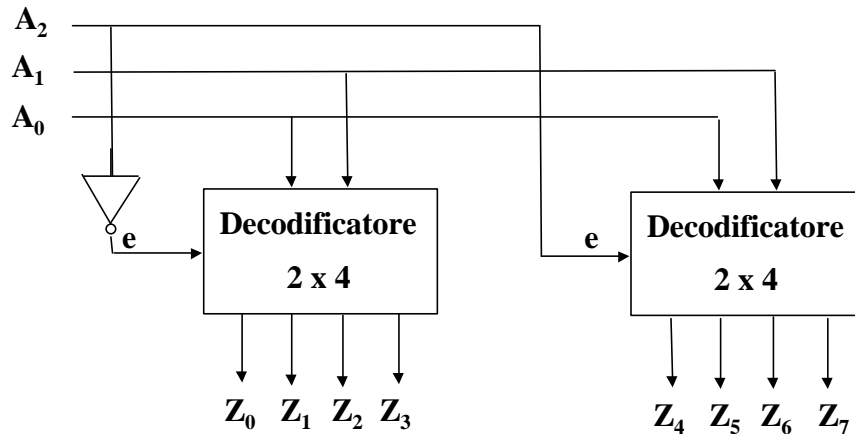
Decodificatori: realizzazione



14

M. Sonza Reorda - a.a. 2001/02

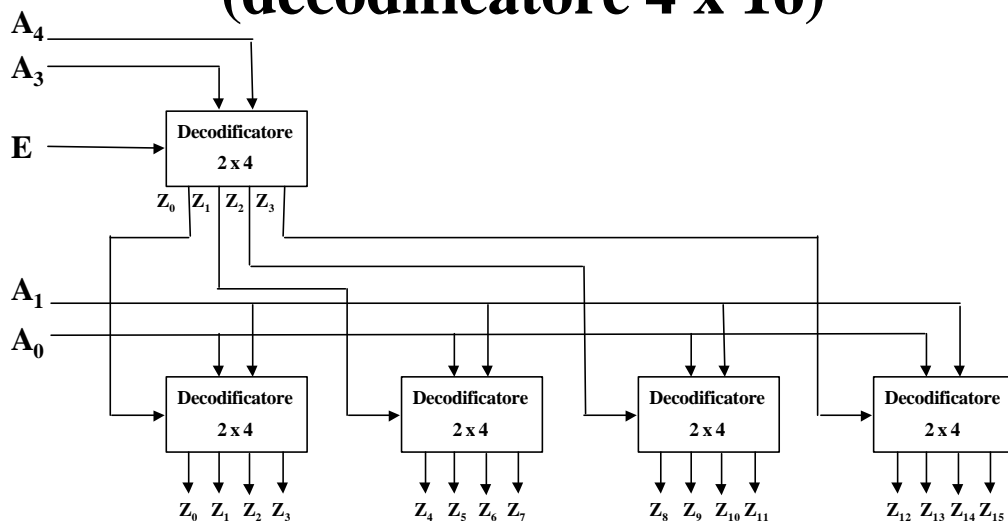
Decodificatori: connessione a cascata (decodificatore 3 x 8)



15

M. Sonza Reorda - a.a. 2001/02

Decodificatori: connessione a cascata (decodificatore 4 x 16)



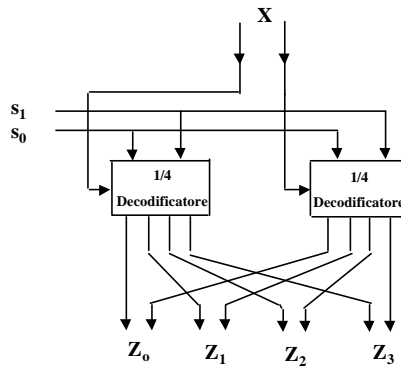
16

M. Sonza Reorda - a.a. 2001/02

Decodificatore = Demultiplexer

I decodificatori possono essere considerati dei *demultiplexer*, in quanto connettono una fonte di informazione (l'*enable*) ad una tra 2^k destinazioni, a seconda del valore delle k linee di ingresso.

Anche in questo caso possono essere connessi ad albero a formare strutture più complesse:



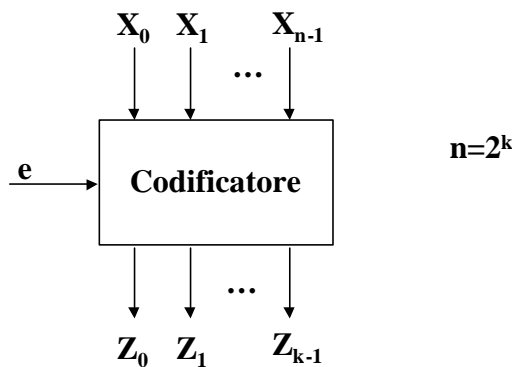
17

M. Senza Reorda - a.a. 2001/02

Codificatori

Hanno 2^k linee di ingresso e k linee di uscita; su queste compare (codificato) il valore corrispondente all'indice della linea di ingresso attiva.

Se nessuna linea di ingresso è attiva viene attivato un apposito segnale (*input inactive*).



18

M. Senza Reorda - a.a. 2001/02

Codificatori: funzione

La tavola della verità di un codificatore 4@2 è la seguente:

X	e	Z
0001	1	00
0010	1	01
0100	1	10
1000	1	11
-	0	00

19

M. Senza Reorda - a.a. 2001/02

Priority Encoder

Se più di una linea di ingresso è attiva il risultato può essere scorretto, oppure può apparire il codice della linea attiva con priorità maggiore (*priority encoder*).

20

M. Senza Reorda - a.a. 2001/02

Codificatori prioritari: funzione

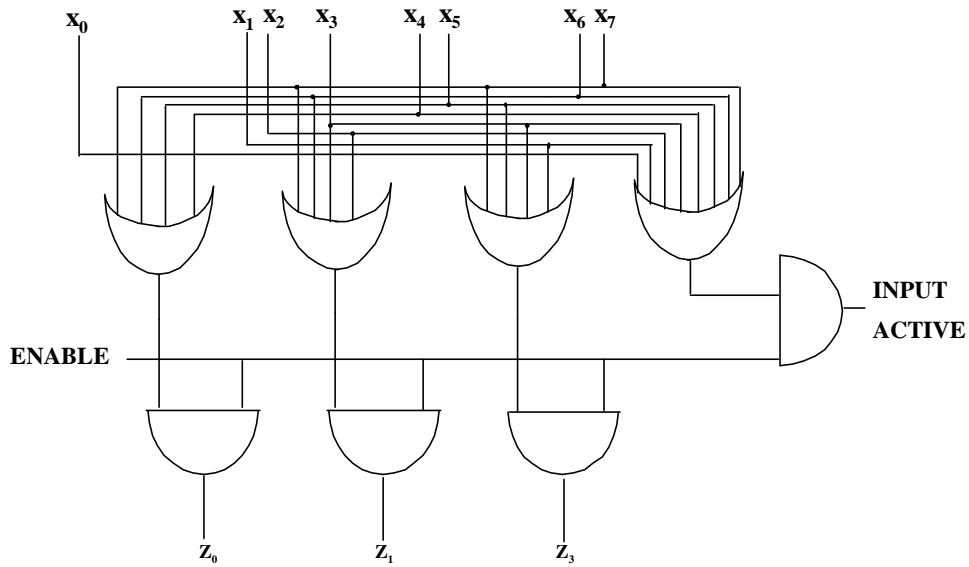
La tavola della verità di un codificatore prioritario 4@2 è la seguente:

X	e	Z
0000	1	00
0001	1	00
0010	1	01
0011	1	01
0100	1	10
0101	1	10
0110	1	10
0111	1	10
1 - - -	1	11
-	0	00

21

M. Senza Reorda - a.a. 2001/02

Codificatore a 8 bit



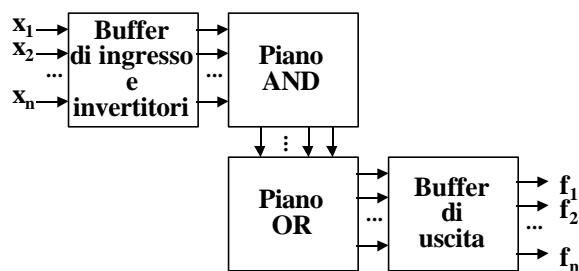
22

M. Senza Reorda - a.a. 2001/02

Dispositivi Logici Programmabili

I Dispositivi Logici Programmabili (o *Programmable Logic Device*, PLD) permettono di realizzare a basso costo delle funzioni logiche combinatorie.

Sono basati su strutture del tipo seguente:



23

M. Sonza Reorda - a.a. 2001/02

PLA

Le PLA (*Programmable Logic Array*) sono componenti che permettono la realizzazione a prezzi economici di funzioni logiche combinatorie (ad es. unità di controllo).

La funzione desiderata si ottiene modificando il piano AND ed il piano OR attraverso comandi elettrici irreversibili.

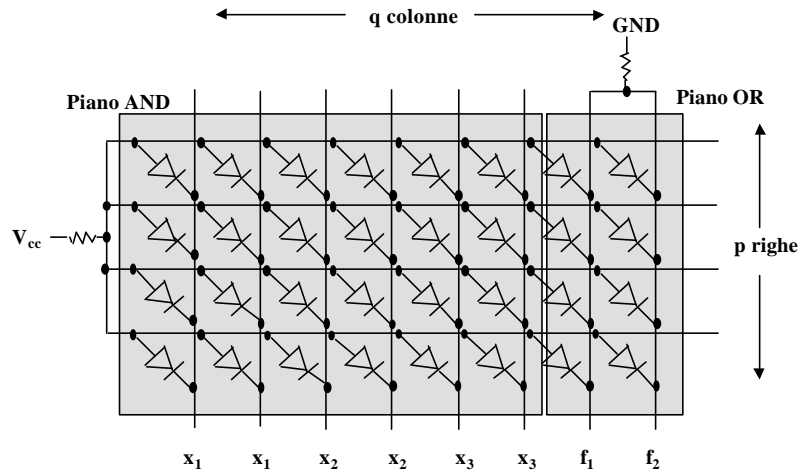
I diodi delle figure successive possono corrispondere a

- fusibili
- transistor.

24

M. Sonza Reorda - a.a. 2001/02

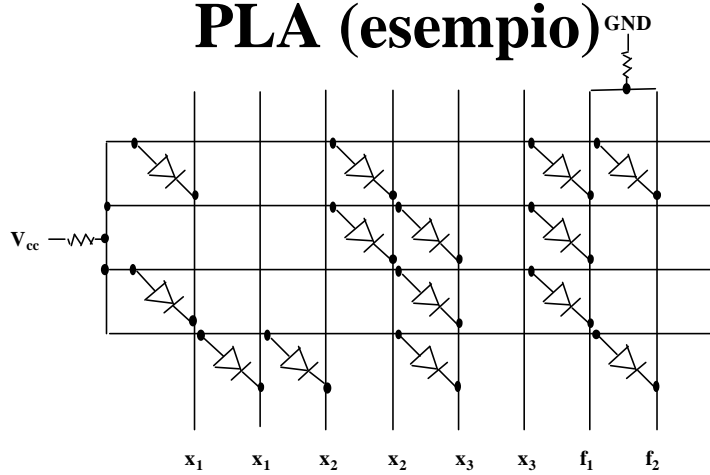
PLA (schema generale)



25

M. Sonza Reorda - a.a. 2001/02

PLA (esempio)



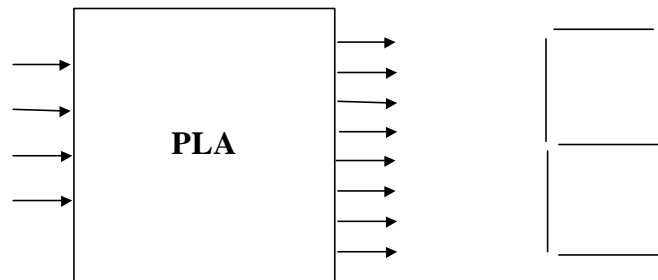
$$f_1 = x_1 \bar{x}_2 + \bar{x}_2 x_3 + x_1 x_3$$

$$f_2 = x_1 \bar{x}_2 + \bar{x}_1 x_2 x_3$$

26

M. Sonza Reorda - a.a. 2001/02

PLA: esempio di applicazione



Display a 7 segmenti

Per ogni configurazione di ingresso alla PLA si deve generare il relativo insieme di segnali di attivazione per i segmenti da attivare.

27

M. Senza Reorda - a.a. 2001/02

PAL

Le PAL (*Programmable Array Logic*) hanno minore flessibilità delle PLA, in quanto

- il piano OR non può essere programmato
- ciascuna linea del piano OR è connessa a priori con determinate uscite del piano AND.

28

M. Senza Reorda - a.a. 2001/02

FPGA

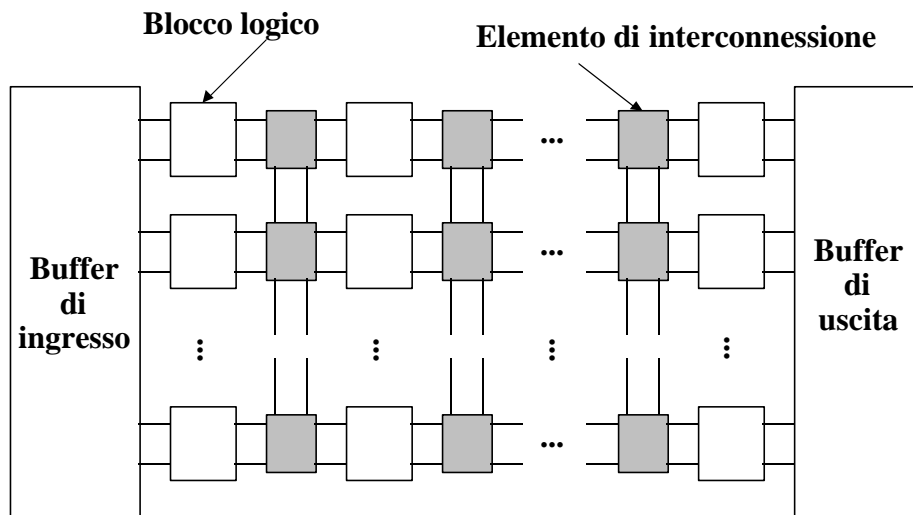
I *Field Programmable Gate Array* (FPGA) sono dispositivi concettualmente simili a PAL e PLA, ma con funzionalità molto superiori.

In pratica un FPGA è composta da una griglia di blocchi logici programmabili (ad esempio PLA) connessi da una rete di interconnessione anch'essa programmabile. Sia i blocchi logici che quelli di interconnessione sono programmabili.

29

M. Senza Reorda - a.a. 2001/02

FPGA: architettura



30

M. Senza Reorda - a.a. 2001/02

FPGA: usi

Le FPGA sono spesso utilizzati per realizzare a basso costo dispositivi altrimenti realizzabili tramite ASIC (*Application Specific IC*). Rispetto agli ASIC le FPGA sono una soluzione

- più economica da progettare
- più costosa da produrre (in termini di solo costo di produzione))
- più lenta
- più costosa in termini di spazio.

Le FPGA sono quindi usate principalmente per

- prodotti realizzati in numero limitato
- applicazioni riconfigurabili
- prototipi.

31

M. Senza Reorda - a.a. 2001/02

FPGA: programmazione

L'uso delle FPGA è reso possibile dalla disponibilità di strumenti CAD a basso costo che generano automaticamente i file di programmazione per il dispositivo.

32

M. Senza Reorda - a.a. 2001/02

FPGA: prodotti

I prodotti più diffusi basati sulla tecnologia FPGA sono

- Xilinx
- Altera.

33

M. Senza Reorda - a.a. 2001/02

Moduli Aritmetici

Sono frequenti quelli che eseguono operazioni di somma e moltiplicazione tra numeri interi.

Tramite i sommatore si può, attraverso l'operazione di complementazione, realizzare anche l'operazione di sottrazione.

34

M. Senza Reorda - a.a. 2001/02

Sommatori

Possono essere realizzati seguendo tre soluzioni alternative:

- Sommatore seriali
- Sommatore combinatori
- Sommatore combinatori modulari.

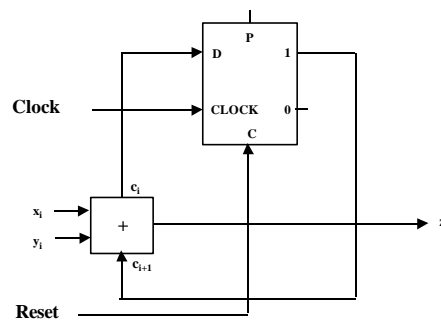
35

M. Senza Reorda - a.a. 2001/02

Sommatore seriale

È la soluzione che richiede in termini di porte logiche un costo minimo ed indipendente da n ; in compenso rappresenta la soluzione con il massimo tempo di risposta (proporzionale ad n).

Il FlipFlop memorizza il Carry di un bit e lo riporta sul bit successivo; all'inizio deve venire azzerato.



36

M. Senza Reorda - a.a. 2001/02

Sommatore Combinatorio

La soluzione più vantaggiosa in termini di tempo richiesto è quella rappresentata da un circuito a 2 livelli progettato ad hoc per sommare 2 numeri su n bit e produrre $n+1$ bit di uscita.

Un simile circuito può essere sintetizzato a partire dall'espressione in termini di somma di prodotti o prodotto di somme della funzione somma.

Tale soluzione ha una complessità che cresce esponenzialmente con n e viene quindi raramente adottata, se non per piccoli valori di n .

37

M. Senza Reorda - a.a. 2001/02

Sommatore Combinatorio Modulare

Più frequentemente si segue un approccio modulare:

- si suddividono i 2 numeri X e Y da sommare in bit (X_i , Y_i)
- si sommano tra loro le coppie X_i , Y_i , partendo dai bit meno significativi
- si combinano i risultati tenendo conto dei riporti.

L'approccio modulare è basato su un circuito elementare denominato *full-adder*.

38

M. Senza Reorda - a.a. 2001/02

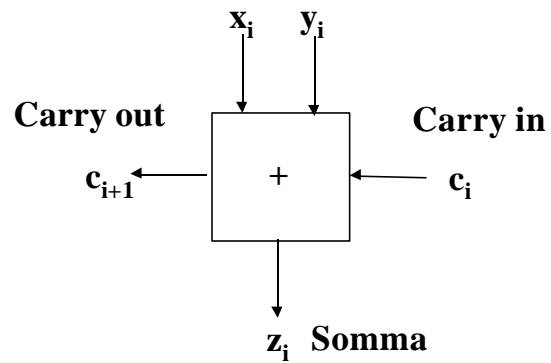
Full-Adder

Calcola la somma di 2 bit x_i , y_i e di un carry in ingresso c_i producendo un bit di risultato z_i ed un bit di carry c_{i+1} :

$$z_i = x_i \oplus y_i \oplus c_i$$

$$c_{i+1} = x_i y_i + x_i c_i + y_i c_i$$

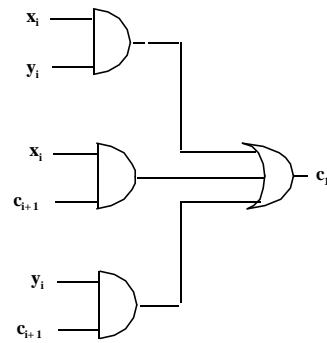
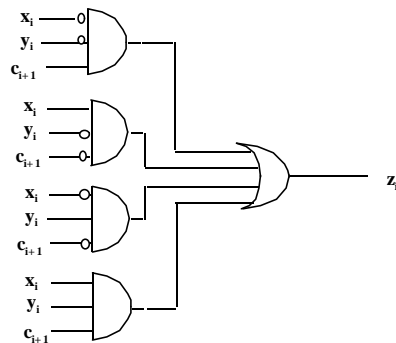
A	B	Carry -In	Sum	Carry -Out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



M. Sonza Reorda - a.a. 2001/02

39

Full-Adder: implementazione



40

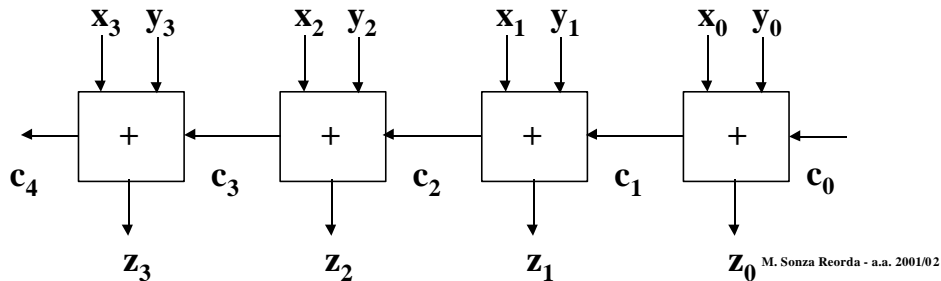
M. Sonza Reorda - a.a. 2001/02

Ripple Carry Adder

Somma 2 numeri su n bit utilizzando una logica esclusivamente combinatoria. Viene costruito connettendo in cascata n full-adder (*ripple carry adder*).

Il tempo richiesto per la generazione dell'ultimo carry è pari a nd , ove d è il ritardo del singolo modulo.

Il costo in termini di hardware è proporzionale ad n .



41

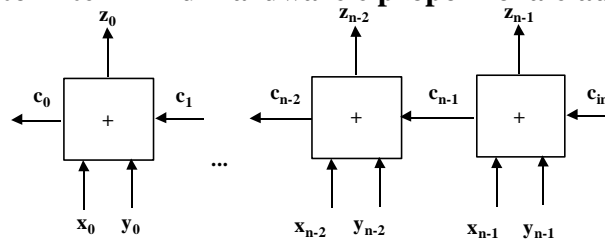
Sommatore parallelo

Sommano 2 numeri su n bit utilizzando una logica esclusivamente combinatoria.

Possono essere costruiti connettendo in cascata n full-adder (*ripple carry adder*).

Il tempo richiesto per la generazione dell'ultimo carry è pari a nd , ove d è il ritardo del singolo modulo.

Il costo in termini di hardware è proporzionale ad n .



42

M. Sonza Reorda - a.a. 2001/02

Sommatore con carry-lookahead

Permette di ridurre il ritardo nella generazione del risultato, dovuto al fatto che ogni modulo deve attendere il carry generato dal modulo precedente.

Il *Carry-Lookahead Generator* è un circuito in grado di generare il bit di carry di ogni modulo sulla base dei segnali che gli vengono in parallelo da tutti i moduli.

Questi sono una versione modificata di full-adder in cui vengono generati 2 segnali particolari:

$$g_i = x_i y_i$$

$$p_i = x_i + y_i$$

43

M. Senza Reorda - a.a. 2001/02

Carry-Lookahead Generator

Il carry out del singolo modulo è dato da

$$c_i = x_i y_i + x_i c_{i+1} + y_i c_{i+1} = g_i + p_i c_{i+1}$$

Analogamente

$$c_{i+1} = g_{i+1} + p_{i+1} c_{i+2}$$

Sostituendo

$$c_i = g_i + p_i g_{i+1} + p_i p_{i+1} c_{i+2}$$

Ad esempio in un sommatore con carry-lookahead da 4 bit si ha:

$$c_3 = g_3 + p_3 c_{in}$$

$$c_2 = g_2 + p_2 g_3 + p_2 p_3 c_{in}$$

$$c_1 = g_1 + p_1 g_2 + p_1 p_2 g_3 + p_1 p_2 p_3 c_{in}$$

$$c_0 = g_0 + p_0 g_1 + p_0 p_1 g_2 + p_0 p_1 p_2 g_3 + p_0 p_1 p_2 p_3 c_{in}$$

44

M. Senza Reorda - a.a. 2001/02

Significato di g_i e p_i

I due coefficienti g_i e p_i derivano il loro nome dal fatto che permettono

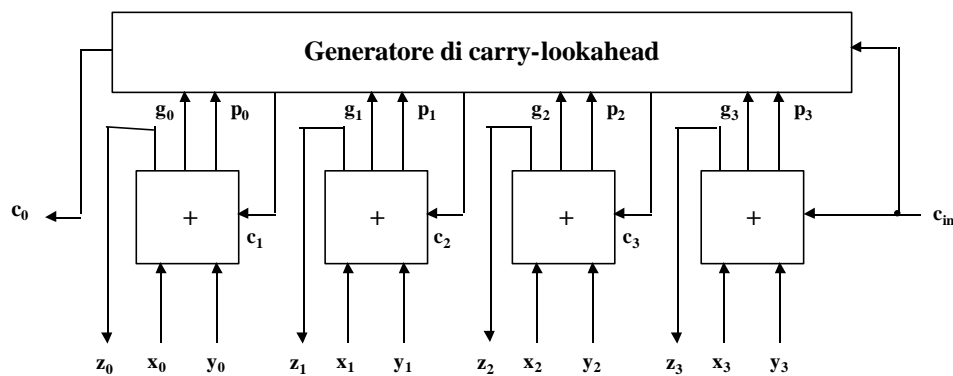
- la generazione, per g_i
- la propagazione, per p_i

dei segnali di carry relativi ai vari moduli.

45

M. Senza Reorda - a.a. 2001/02

Implementazione



46

M. Senza Reorda - a.a. 2001/02

Vantaggi

Detto d il ritardo introdotto da un circuito a 2 livelli, il ritardo di un sommatore con carry-lookahead è pari a $3d$; con particolari tecniche il ritardo può ulteriormente essere ridotto a $2d$.

La complessità hardware richiesta rende in genere impraticabile questa soluzione per $n > 8$.

47

M. Sonza Reorda - a.a. 2001/02

Approccio Misto

Può essere utilizzato per sommare numeri su n bit, con n grande.

Vengono utilizzati n/k carry-lookahead adder per sommare i gruppi di k bit, e un ripple-adder per riportare i carry da un gruppo all'altro.

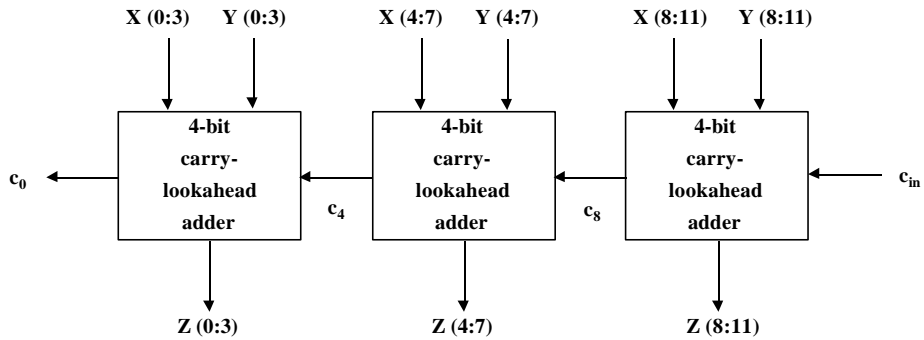
Se ci sono m gruppi il ritardo complessivo sarà $(m+2)d$.

Ad esempio se $n=12$, ed $m=3$, si ha che il ritardo è pari a $5d$ con l'approccio misto, mentre sarebbe $12d$ con un ripple-adder.

48

M. Sonza Reorda - a.a. 2001/02

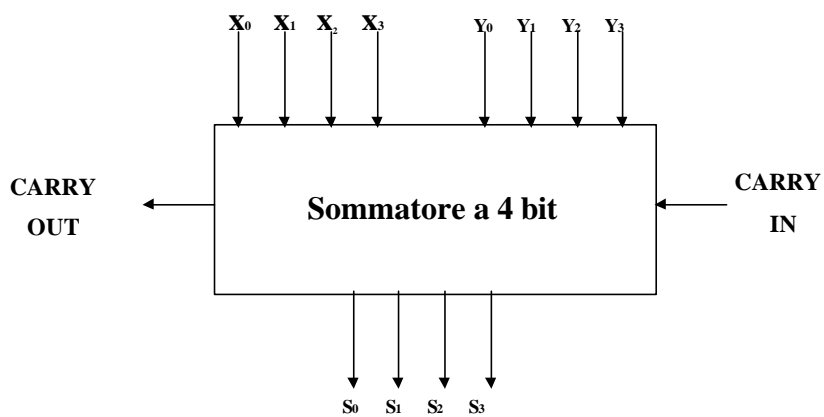
Implementazione



49

M. Sonza Reorda - a.a. 2001/02

Sommatore a 4 bit

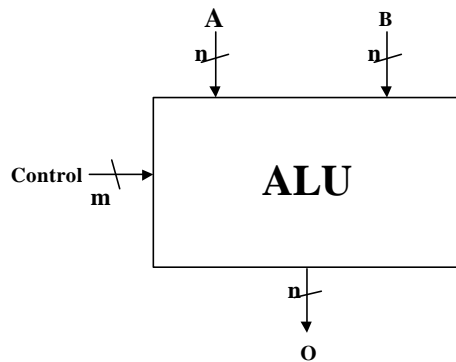


50

M. Sonza Reorda - a.a. 2001/02

ALU

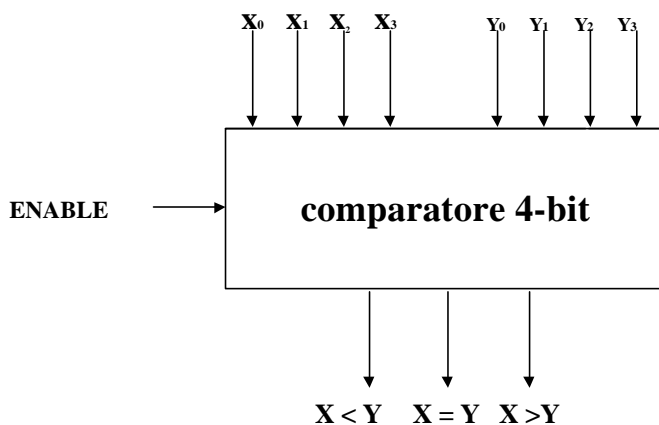
Le Unità Aritmetico-Logiche (ALU) sono componenti combinatori che integrano in un unico blocco le principali funzioni aritmetiche e logiche (tipicamente somma, sottrazione, negazione, and, or, not, exor).



51

M. Senza Reorda - a.a. 2001/02

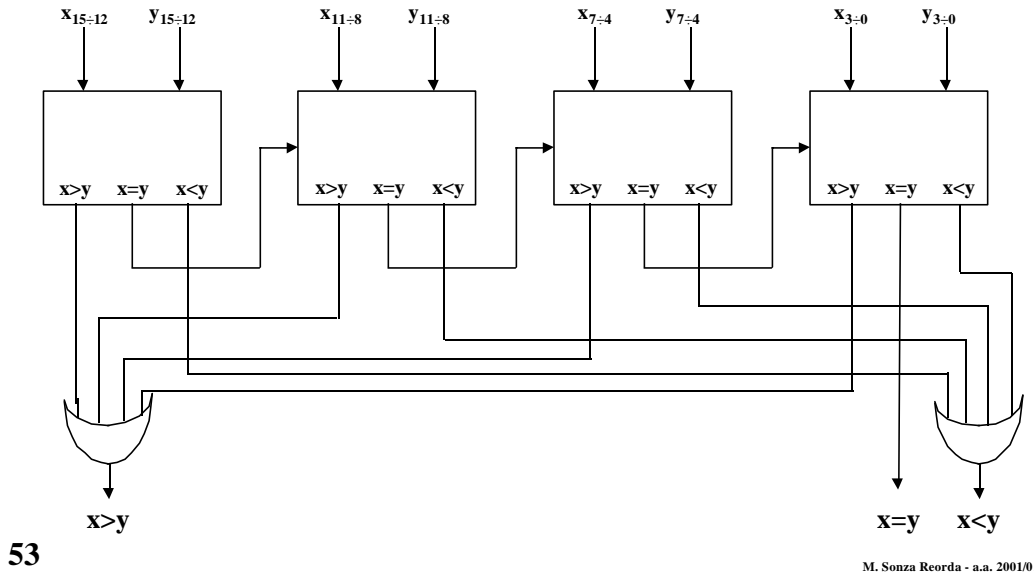
Comparatore a 4 bit



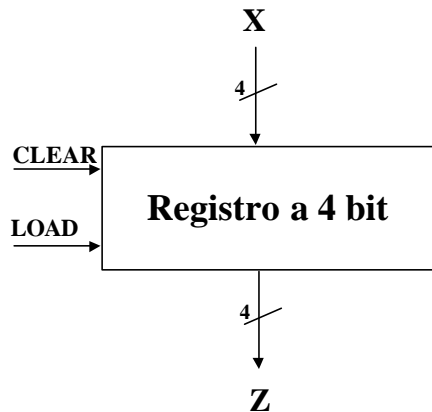
52

M. Senza Reorda - a.a. 2001/02

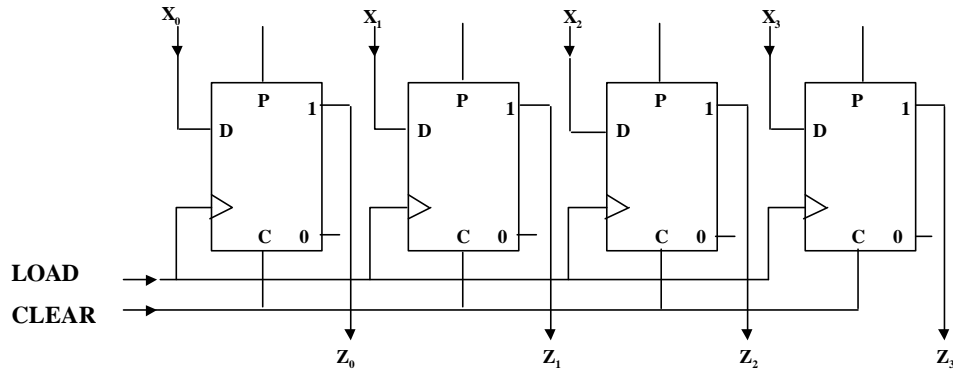
Connessione di Comparatori



Registro a 4 bit



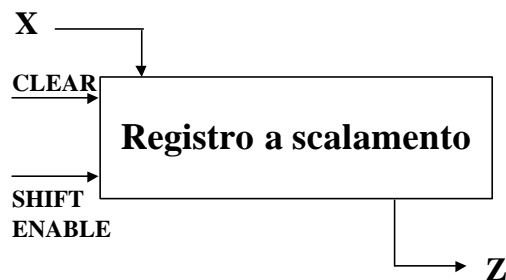
Registro a 4 bit (II)



55

M. Sonza Reorda - a.a. 2001/02

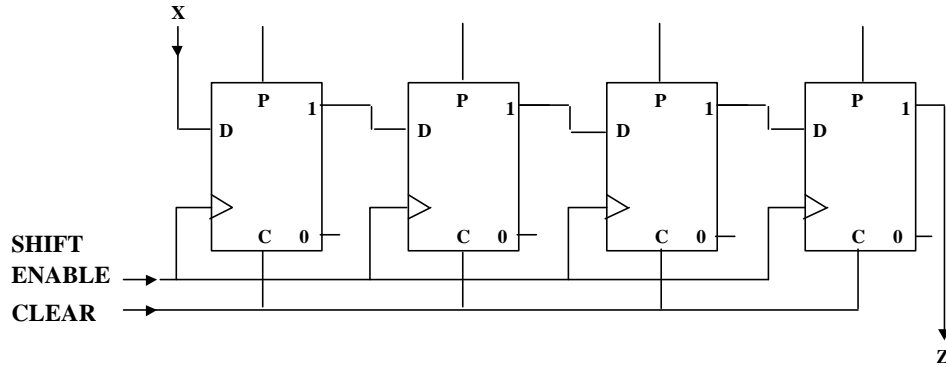
Registri a Scalamento (Shift Register)



56

M. Sonza Reorda - a.a. 2001/02

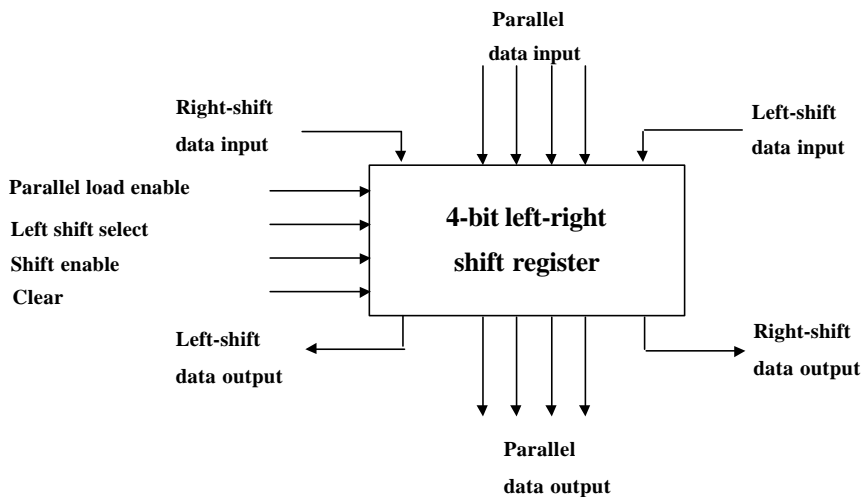
Registro a Scalamento a 4 bit: realizzazione



57

M. Sonza Reorda - a.a. 2001/02

Registro a Scalamento Universale



58

M. Sonza Reorda - a.a. 2001/02

Usi dei Registri a Scalamento

- Memorizzazione di dati seriali (FIFO)
- Conversione seriale-parallelo e parallelo-seriale
- Moltiplicazione e divisione su numeri in fixed-point senza segno e con segno (*rotazione*).

59

M. Senza Reorda - a.a. 2001/02

Contatori

Evolgono attraverso k stati in risposta a k impulsi in ingresso; la codifica degli stati permette di contare il numero di impulsi.

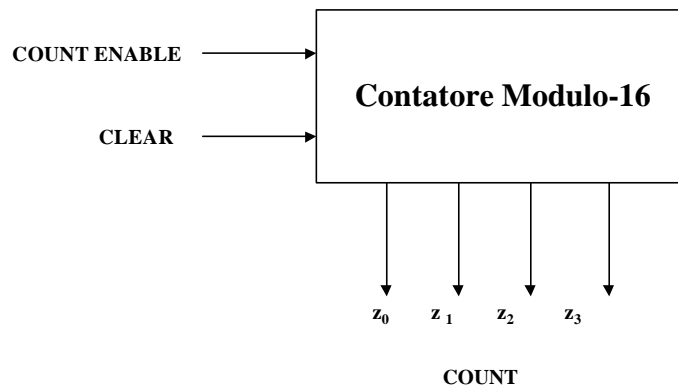
Tipologie:

- *up-down counter*: possono contare avanti e indietro
- *programmable counter*: il valore del modulo può essere modificato.

60

M. Senza Reorda - a.a. 2001/02

Contatore semplice



61

M. Senza Reorda - a.a. 2001/02

Usi dei contatori

- Come *Program Counter* in una macchina a stati.
- Per la generazione di segnali di temporizzazione (*divisori di frequenza*).
- Come contatori di eventi.

62

M. Senza Reorda - a.a. 2001/02

Contatori: modalità di realizzazione

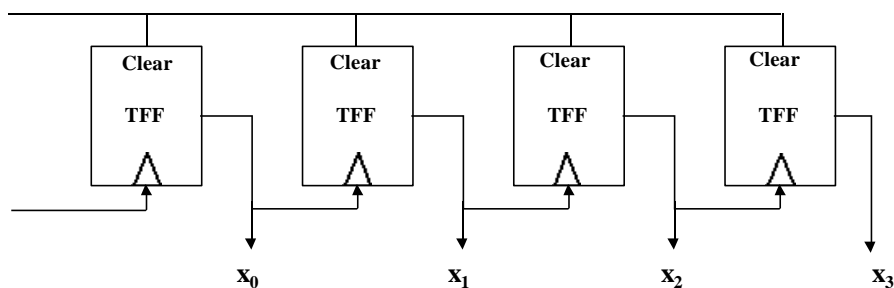
Esistono due principali modalità di realizzazione:

- *ripple counter* (o asincroni); hanno un ritardo dipendente dalla lunghezza
- *contatori sincroni*: tutte le uscite assumono contemporaneamente il valore corrente.

63

M. Senza Reorda - a.a. 2001/02

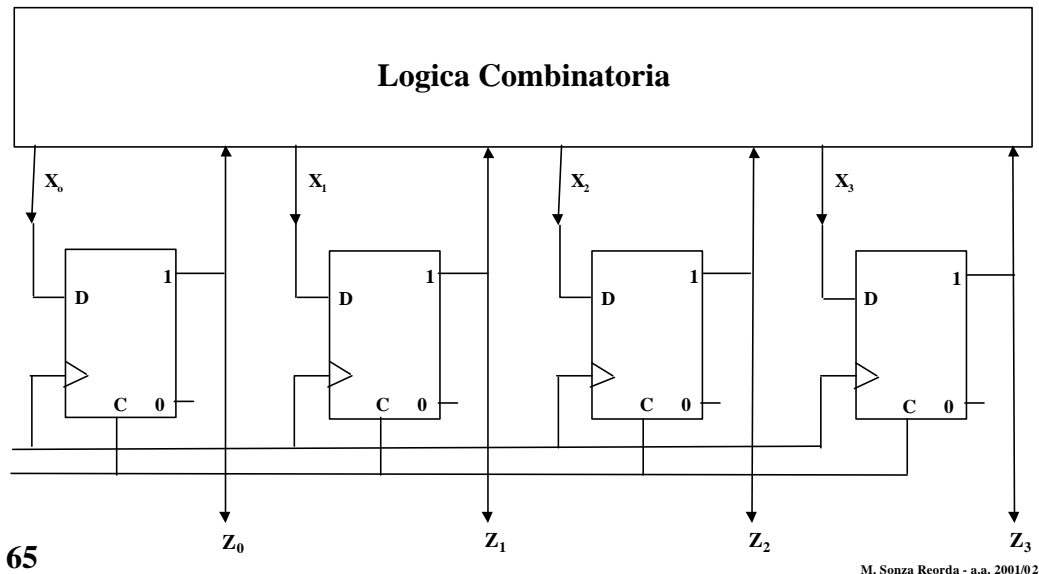
Contatore Asincrono



64

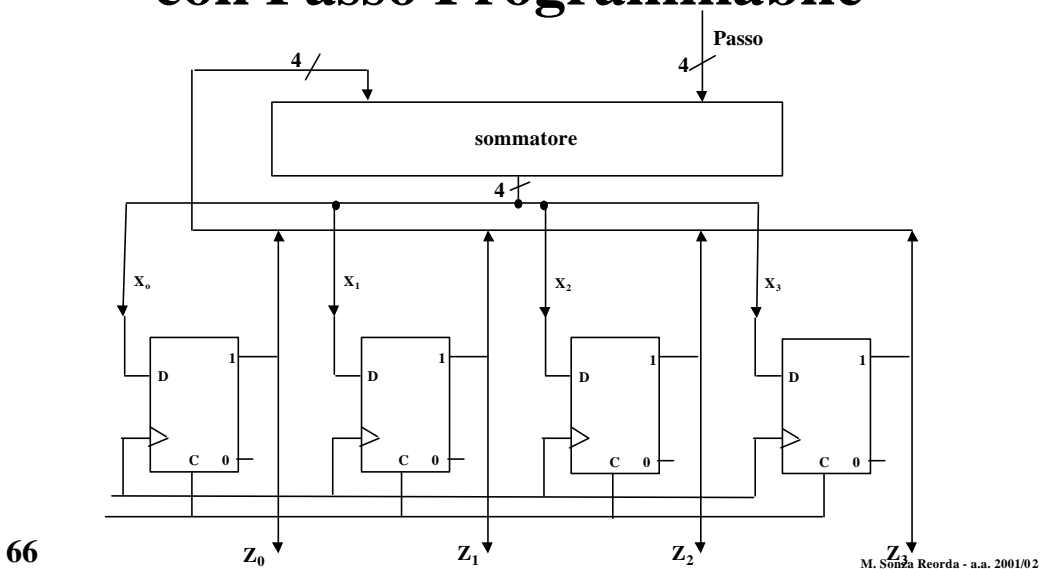
M. Senza Reorda - a.a. 2001/02

Contatore Sincrono



M. Sonza Reorda - a.a. 2001/02

Contatore Sincrono Modulo 16 con Passo Programmabile



M. Sonza Reorda - a.a. 2001/02

Bus

Permettono lo scambio di dati tra diversi componenti.

Possono essere realizzati sia all'interno del singolo IC, sia sulla piastra, sia come connessioni tra piastre (*backplane*).

Al bus possono essere associati:

- **dispositivi di amplificazione**
- **buffer.**

67

M. Senza Reorda - a.a. 2001/02

Bus: costo

Il costo di un bus può essere elevato in termini di area di silicio (se connette moduli distanti sullo stesso IC) e di pin (se è accessibile dall'esterno del chip).

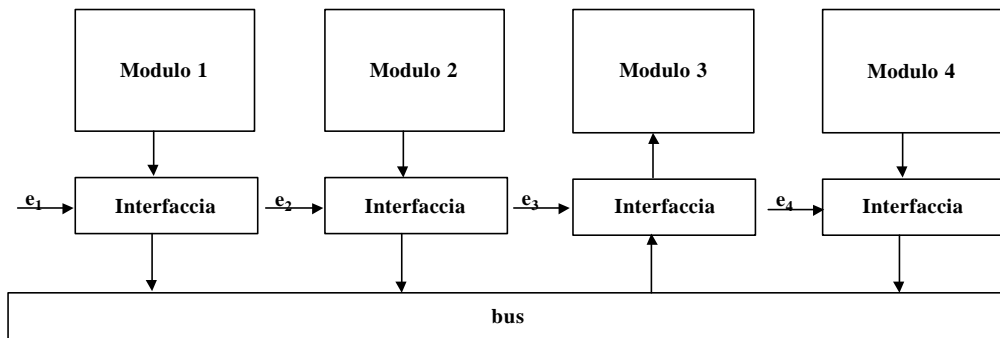
68

M. Senza Reorda - a.a. 2001/02

Connessione al Bus

Ad ogni istante uno ed un solo dispositivo tra quelli le cui uscite sono connesse al bus deve pilotarne il valore.

Gli altri devono assumere un valore particolare noto come **Z** (alta impedenza). I buffer che fungono da interfaccia verso il bus sono detti *tri-state*.



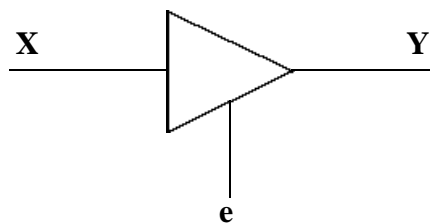
69

M. Senza Reorda - a.a. 2001/02

Buffer Tri-State

Possiedono n ingressi di dato X , 1 ingresso di controllo e , ed n uscite di dato Y . Il valore di Y è

- quello di X , se $e=1$
- Z , se $e=0$.



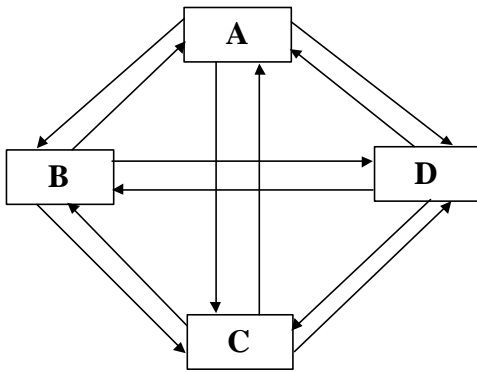
70

M. Senza Reorda - a.a. 2001/02

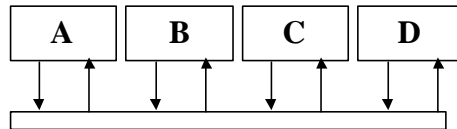
Bus: tipologie

Possono essere:

- *dedicati* (alto costo, elevate prestazioni)
- *condivisi* (basso costo, basse prestazioni): possono richiedere un meccanismo di regolamentazione degli accessi.



Bus Dedicato



Bus Condiviso

M. Senza Reorda - a.a. 2001/02

71

Memorie

Sono componenti corrispondenti funzionalmente ad insiemi di celle di memorizzazione, ciascuna in grado di memorizzare un bit.

Le celle sono organizzate in *parole*. Tutte le celle di una parola sono lette e scritte insieme durante le operazioni di I/O da/verso la memoria.

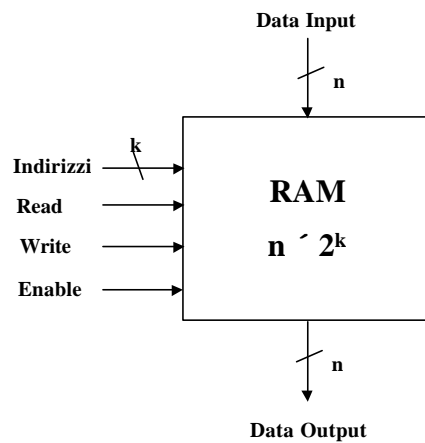
Ogni parola è caratterizzata da un *indirizzo*, corrispondente ad un intero tra 0 e n (dimensione della memoria).

La lettura o scrittura di una parola avviene applicando il corrispondente indirizzo agli ingressi della memoria.

72

M. Senza Reorda - a.a. 2001/02

RAM



73

M. Sonza Reorda - a.a. 2001/02

RAM: funzionamento

Ciclo di scrittura:

- Si fornisce l'indirizzo
- Si fornisce il dato
- Si attiva il segnale di write.

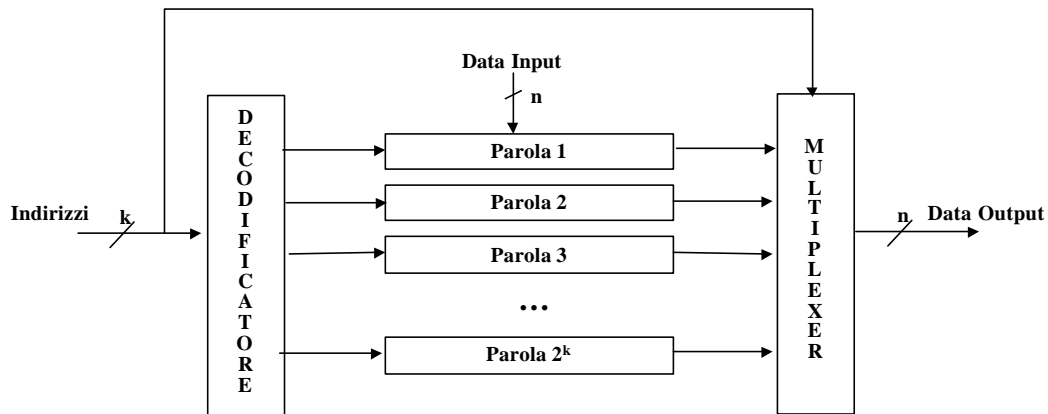
Ciclo di lettura:

- Si fornisce l'indirizzo
- Si attiva il segnale di read
- Si legge il dato.

74

M. Sonza Reorda - a.a. 2001/02

RAM: struttura interna



75

M. Senza Reorda - a.a. 2001/02

ROM

Sono memorie a sola lettura.

Sono equivalenti ad una rete combinatoria.

Sono realizzate con 4 possibili tecnologie:

- *mask programming* (per grandi quantità)
- *PROM* (programmabili con programmatore)
- *EPROM* (programmabili dopo esposizione a raggi ultravioletti)
- *EEPROM* (programmabili dopo applicazione segnali elettrici).

76

M. Senza Reorda - a.a. 2001/02

ROM

