



UNIVERSITÀ DI PISA

CORSO DI LAUREA IN FISICA

LABORATORIO DI FISICA 3

ELETTRONICA DIGITALE
COMPLESSA
COMBINATORIA

Prof. F. Forti

Logica complessa

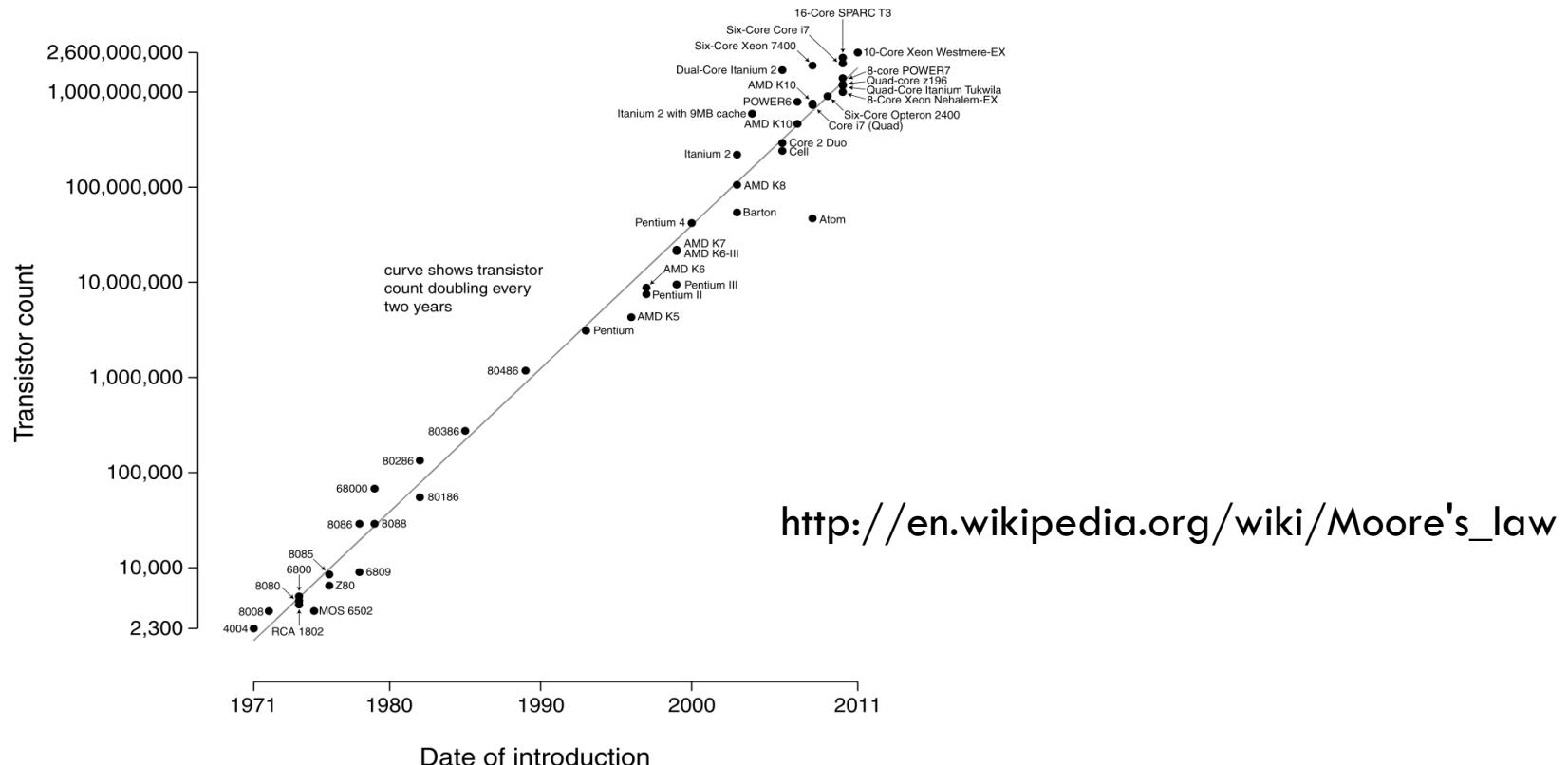
2

- Lo sviluppo delle tecnologie di produzione dei circuiti elettronici ha permesso di costruire circuiti logici sempre più complessi
- Aumento vertiginoso del numero di porte logiche delle possibilità relative
 - Cambio dei paradigmi di progettazione
- Compromessi tra costi di progettazione e produzione, versatilità, velocità
 - ASICs – Application specific Integrated circuits
 - Dedicati ad una singola applicazione.
 - Logica programmabile
 - Versatili e veloci, ma limitati in dimensione
 - Microprocessori
 - Super versatili, ma lenti

Legge di Moore

3

Microprocessor Transistor Counts 1971-2011 & Moore's Law



Tecniche di progettazione

4

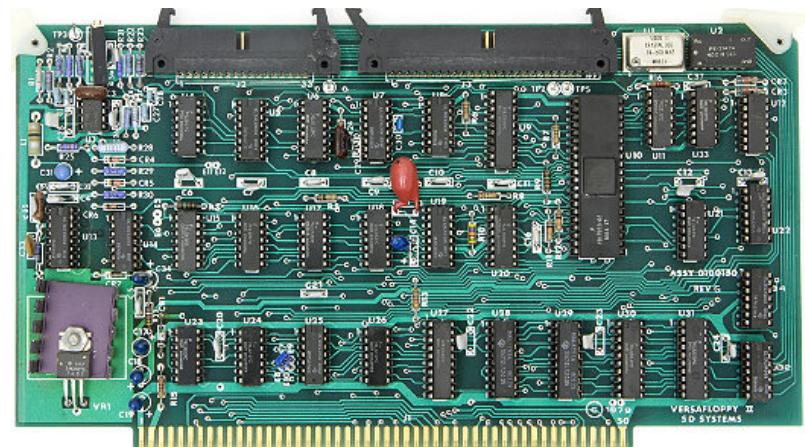
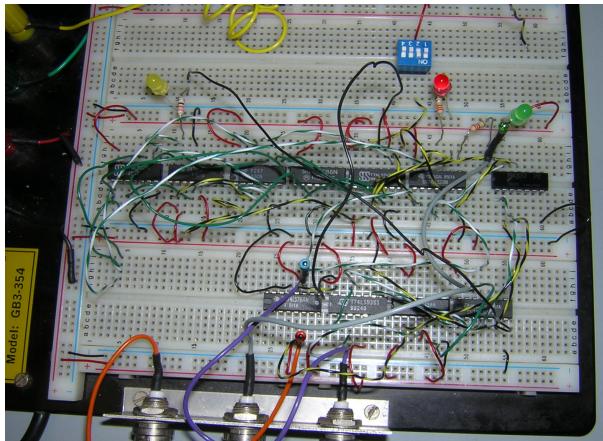
- Se la funzionalità del circuito si complica non si può progettare “a mano”.
- Sistemi software complessi per simulare il funzionamento del circuito.
 - Struttura gerarchica delle funzioni logiche → cambia il modo in cui si pensa al circuito e lo si inserisce nel progetto
- HDL = Hardware description language
 - VHDL e Verilog due standard de facto
- Permette di descrivere il circuito in forma
 - Strutturale = come sono collegate le porte
 - Comportamentale = cosa deve fare il blocco logico
- Permette di “sintetizzare” il circuito a partire dalle funzionalità richieste
 - Generazione semi-automatica o automatica dell’implementazione
- Varie implementazioni possibili

Implementazioni di circuiti logici: gates

5

□ Standard Parts:

- Integrati con porte logiche semplici o più complessi.
- La serie dei componenti 74xx sono stati utilizzati dal 1960 per circa 20 anni per l'implementazione dei circuiti logici
- Varie tecnologie disponibili: TTL, ECL, CMOS

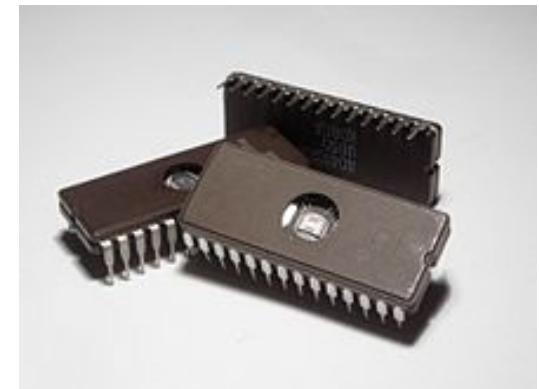


□ ASICS – implementazione diretta su silicio del circuito logico

Implementazioni di circuiti logici: LUTs

6

- Look-up-tables: memorizzare la funzione logica desiderata
- ROM (Read Only Memory) sono matrici di 0 e 1 che permettono di implementare una determinata tabella delle verita`.
- Encoder/decoders e multiplexers
- Originariamente le ROM erano programmate al momento della produzione (non adatte allo studio dei prototipi).
- Poi sono venute le versioni programmabili una sola volta ma dall'utente: PROM – Programmable ROM
- ... e poi nella versione cancellabile e riprogrammabile: EPROM – Erasable (UV) Programmable ROM
- ... EEPROM - Electrical Erasable Programmable ROM – possono essere riprogrammate senza rimuovere il chip dal circuito
- Oggi: Flash memories. Simili come concetto alle EEPROM



EPROM

Implementazione di circuiti logici: Template-based

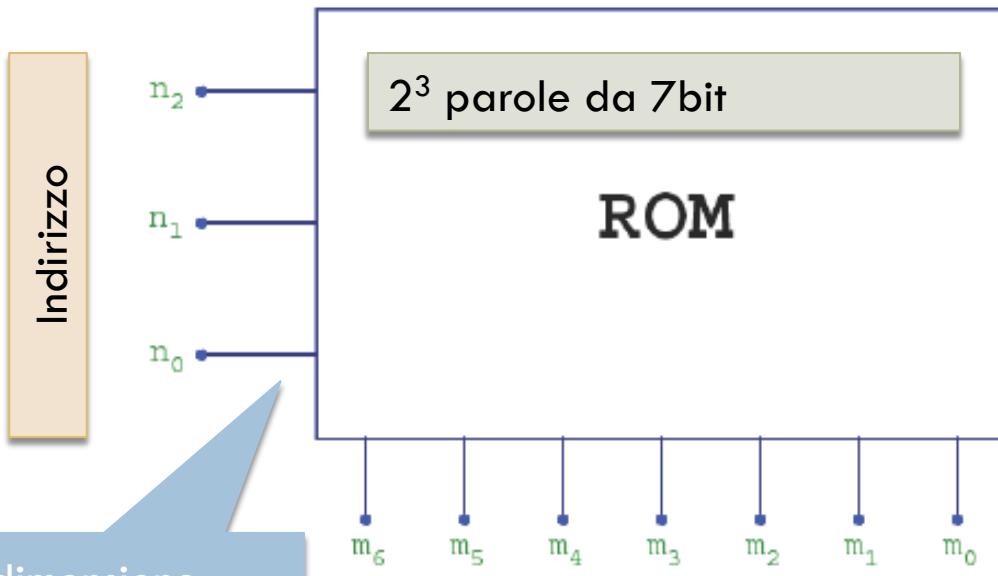
7

- Template = struttura pre-definita ; PLD = Programmable Logic Device
- Meno flessibili ma meno care e più veloci sono le Programmable Array Logic (PAL), Programmable Logica Array (PLA)
 - Introdotte nel 1975-1980
 - Sono matrici organizzate su due livelli: primo livello sono AND ed il secondo livello sono OR + un livello opzionale di NOT → decomposizione standard in somma di minterm
 - Le connessioni delle varie porte possono essere programmate: 1 volta per tutte bruciando dei fusibili o più volte con tecnologie analoghe a quelle utilizzate per EEPROM
 - PLA permette di programma sia il livello degli AND che quello degli OR, PAL solo AND ma sono più veloci
 - NOTA: PLA e PAL non sono quasi più usate
- FPGA = Field Programmable Gate Array
 - Struttura più complessa e flessibile delle PAL, riprogrammabile sul campo
 - Implementazioni di strutture logiche molto complesse, attraverso lo sviluppo di programmi specifici → **firmware**

Implementazioni a Look-up Table: ROM

8

- E' un insieme ordinato di 2^n parole di m bits
- Ingresso (n bits): l'indirizzo (address) in cui e` immagazzinato il dato
- Uscita (m bits = word): rappresenta il valore del dato (content).



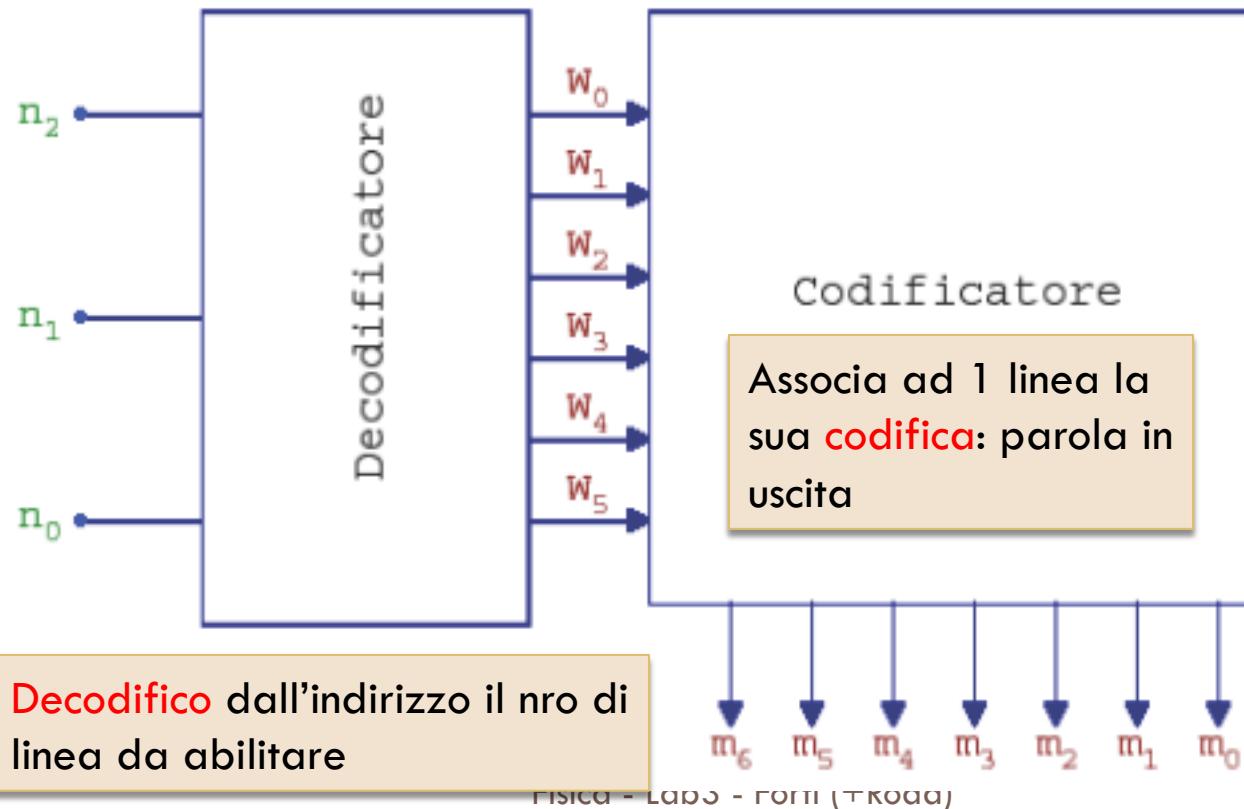
La dimensione
dell'indirizzo indica
quante parole posso
memorizzare

Dato

Struttura della ROM

9

- La ROM e` costituita da un decodificatore che definisce l'indirizzamento e un codificatore che seleziona il dato

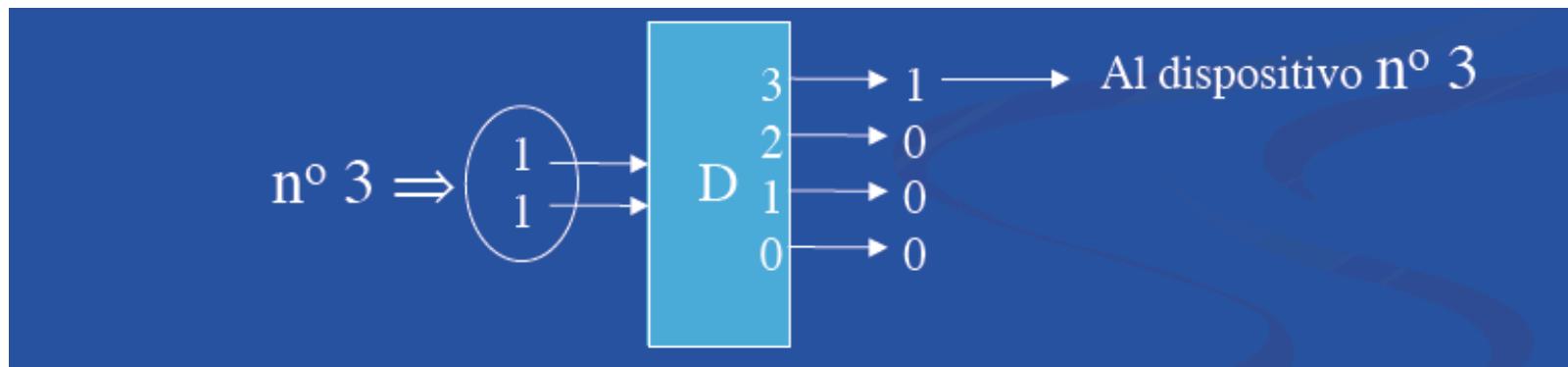


Decodificatore - decoder

10

Il decodificatore associa ad un codice binario in ingresso l'attivazione di una tra n possibili linee.

Esempio: se si vuole abilitare un solo dispositivo scelto tra 4 si puo` utilizzare il Decoder per attivare la linea connessa all'indirizzo programma in ingresso`



Codificatore -Encoder

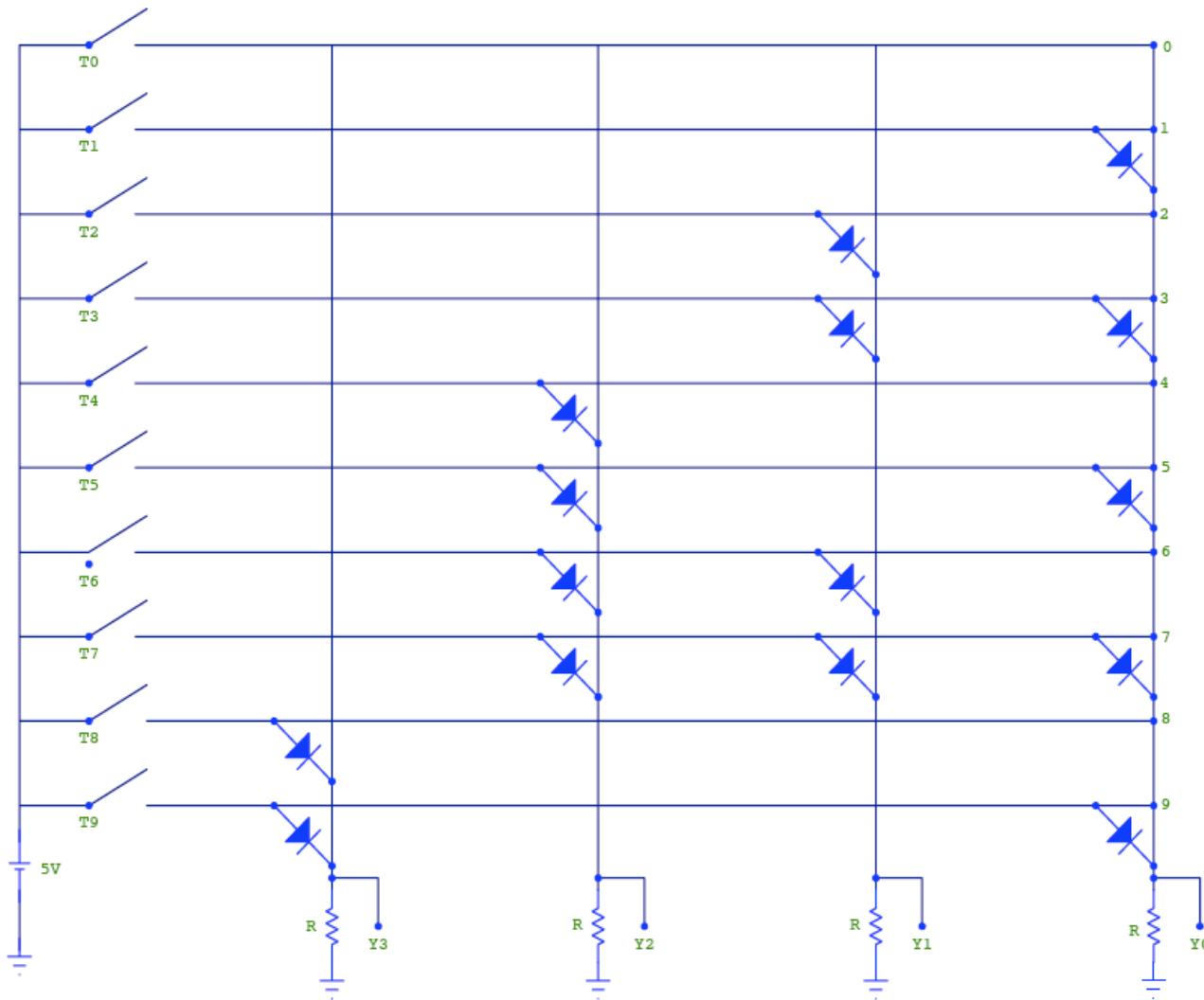
11

- CODIFICATORE (encoder) fa l'operazione inversa
- Se la linea x delle 2^N linee di ingresso è attiva, genera la codifica del numero binario x .
 - La codifica può essere arbitraria (cioè posso generare una codice diverso dal numero x della linea)
 - Gestione di linee attive multiple: priorità, OR, ...

T_9	T_8	T_7	T_6	T_5	T_4	T_3	T_2	T_1	T_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	1

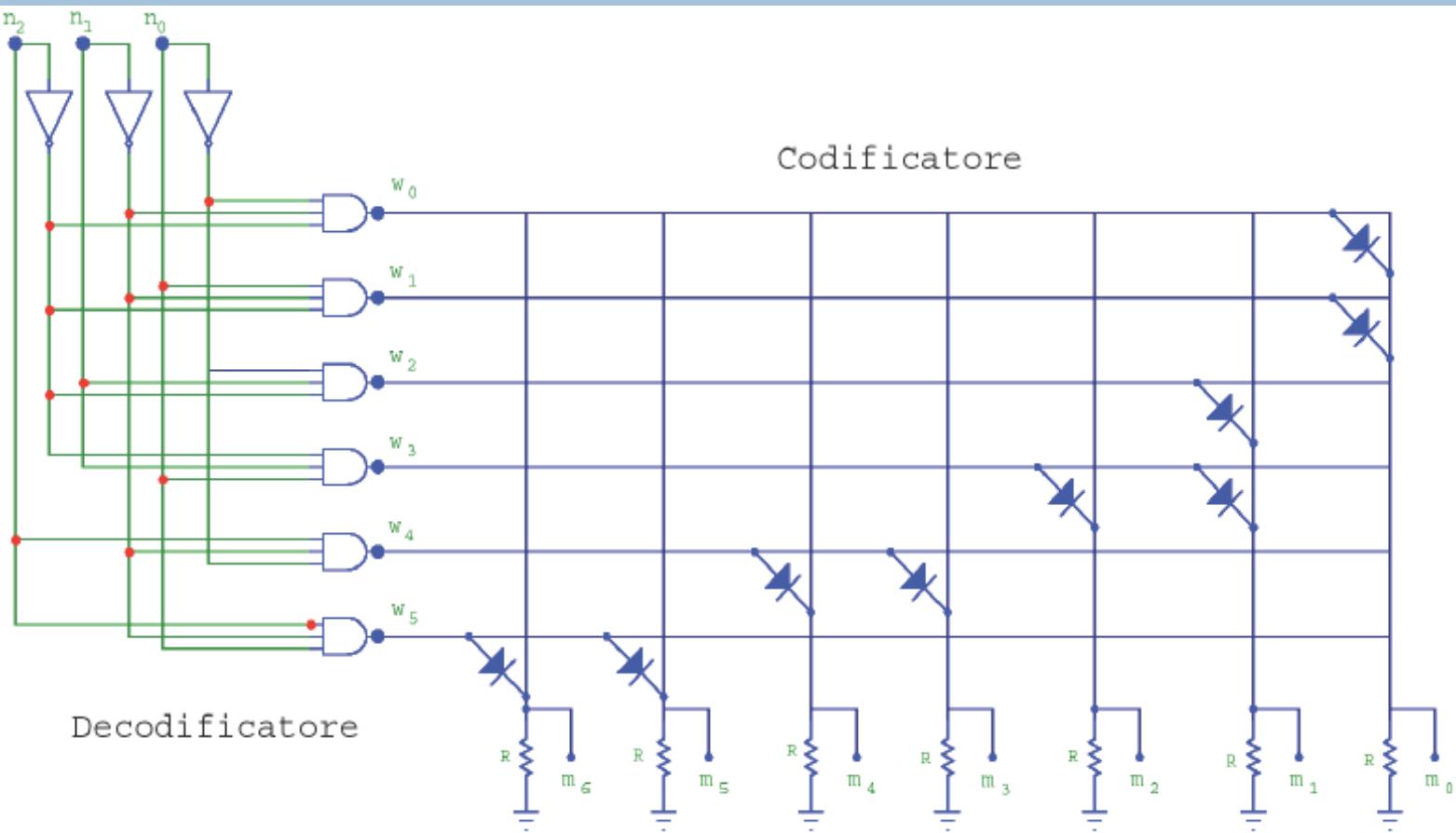
Encoder a diodi

12



Struttura della ROM

13

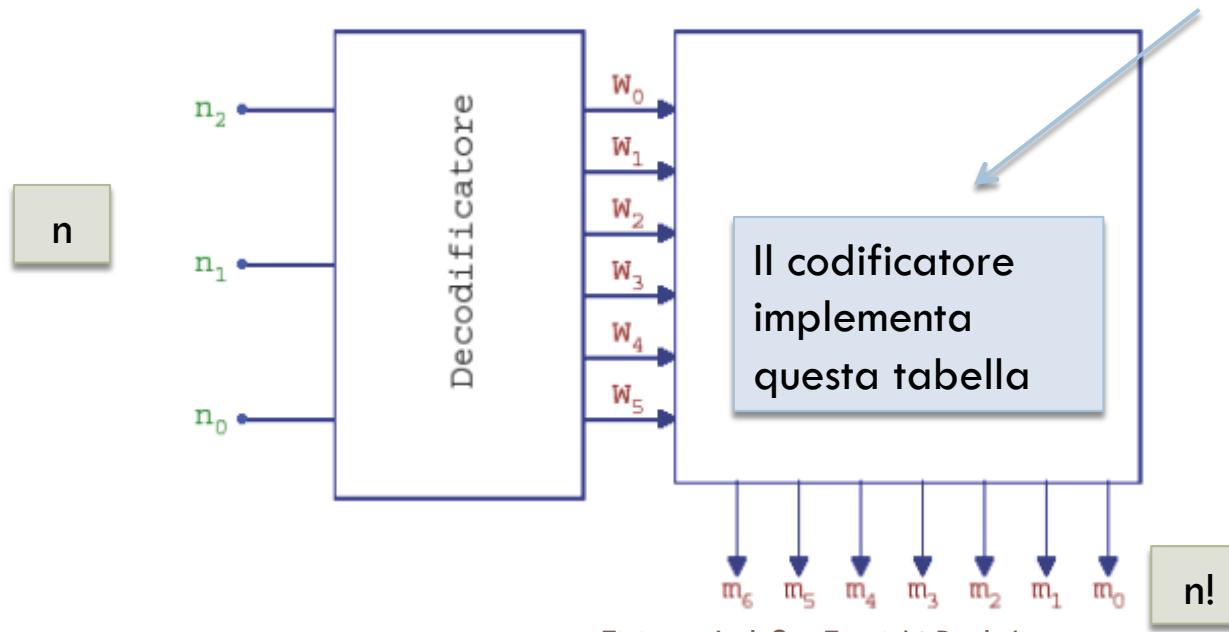


Dimensione della memoria: ROM 8 parole x 7bit/parola

ROM per implementare una tabella della verità

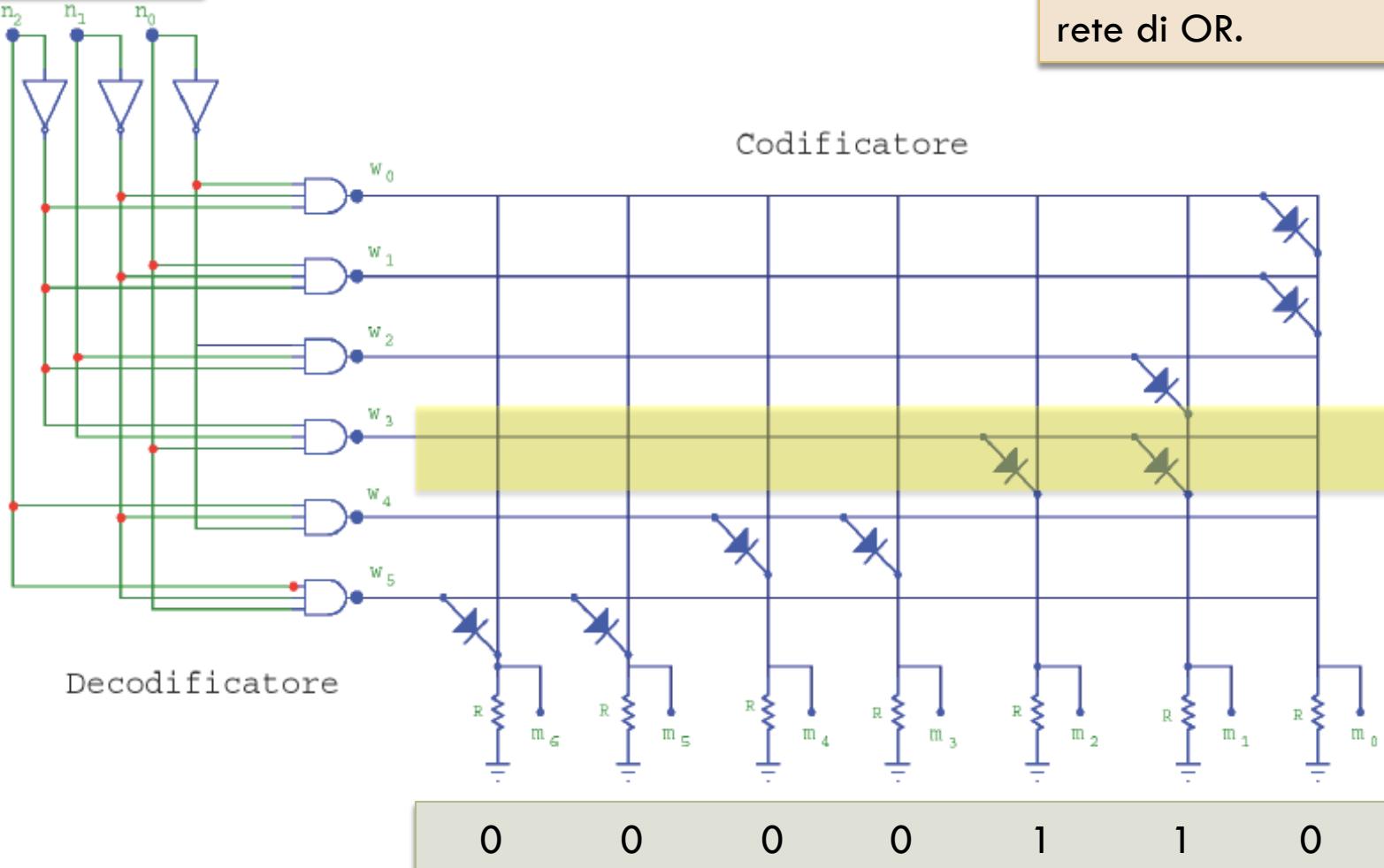
- Esempio: fattoriale di un numero tra 0 e 5
 - Tabella di verità
- 3 ingressi e 7 uscite

n	n_2	n_1	n_0	m	m_6	m_5	m_4	m_3	m_2	m_1	m_0
0	0	0	0	1	0	0	0	0	0	0	1
1	0	0	1	1	0	0	0	0	0	0	1
2	0	1	0	2	0	0	0	0	0	1	0
3	0	1	1	6	0	0	0	0	1	1	0
4	1	0	0	24	0	0	1	1	0	0	0
5	1	0	1	120	1	1	1	1	0	0	0



n! con la ROM

0 1 1

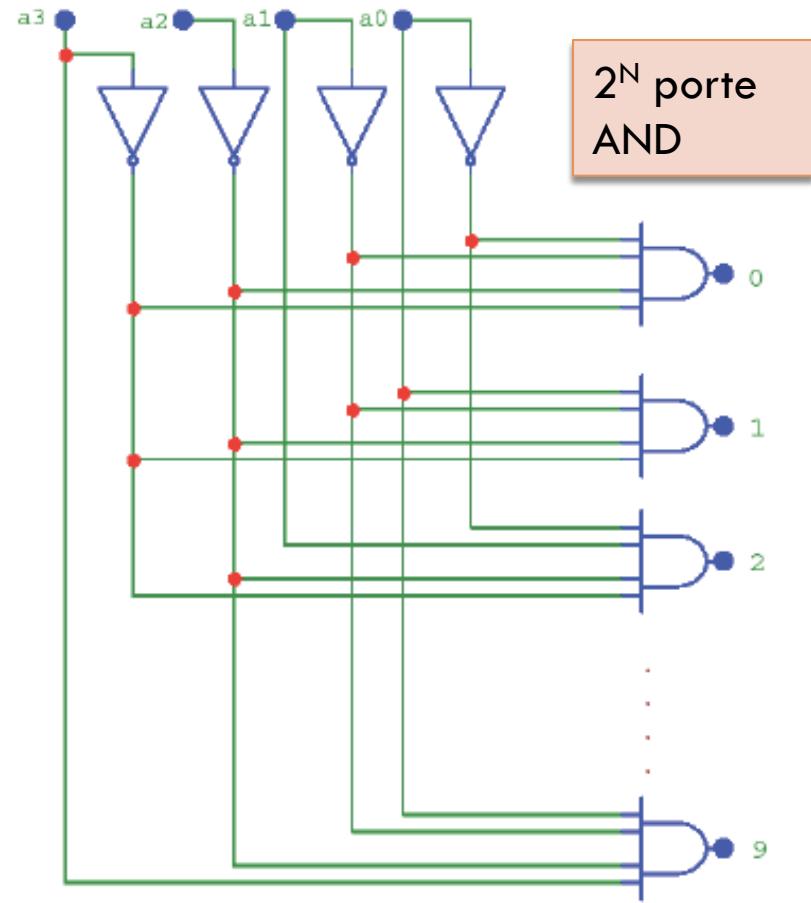


Il codificatore e` un modo semplice e poco costoso per implementare una rete di OR.

Estensione di una ROM

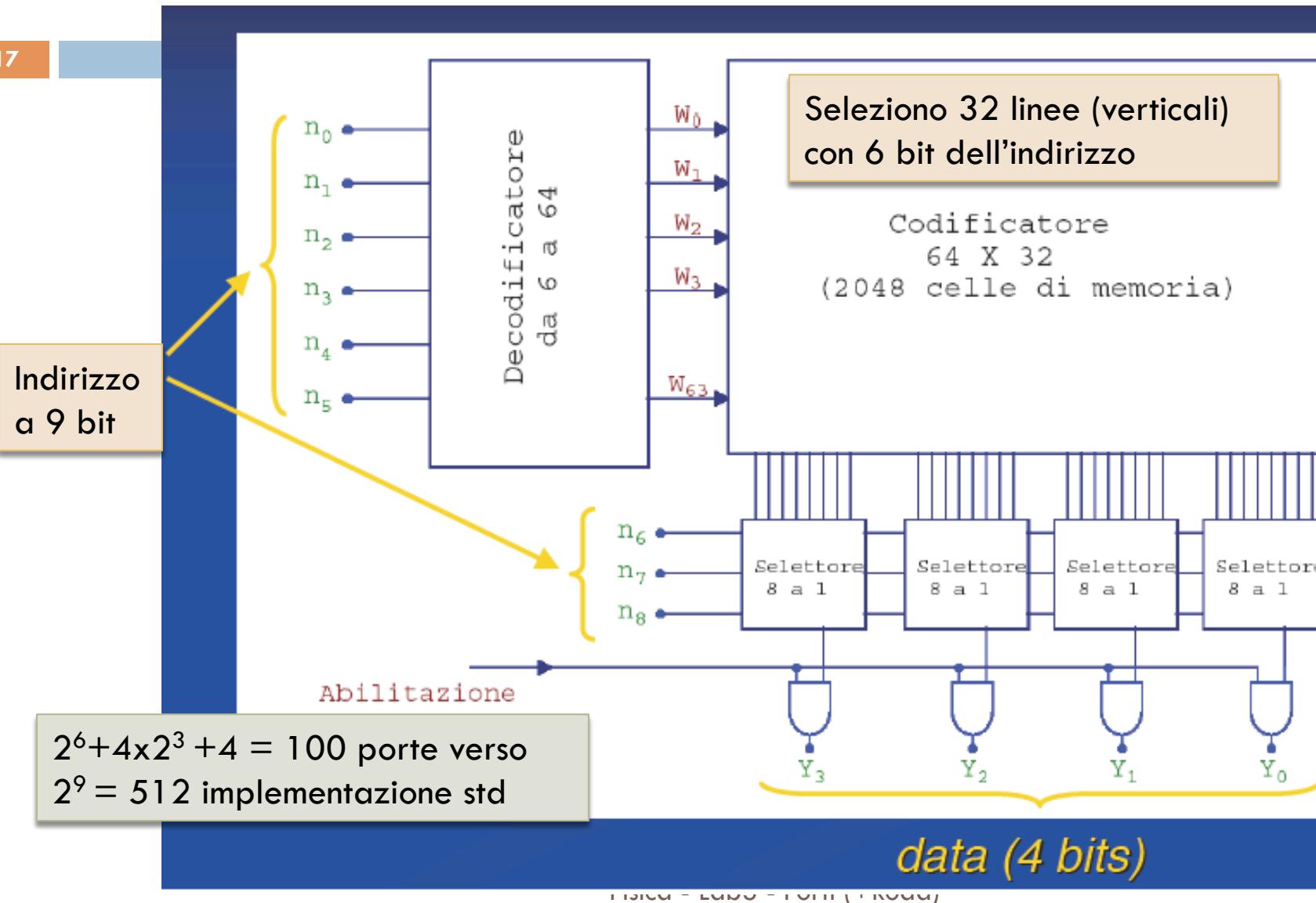
16

- La ROM utilizza un decodificatore che richiede l'utilizzo di una porta AND (a più` ingressi) per parola da memorizzare
- Se il nro di parole e` alto questa architettura diventa troppo complessa
- In questo caso si utilizza il metodo dell'indirizzamento X-Y o bidimensionale
- Supponiamo di volere una ROM che memorizzi **2⁹ parole da 4 bit**. Avremmo bisogno di $2^9 = 512$ porte AND a 9 ingressi.



ROM e indirizzamento X-Y: 512x4bit

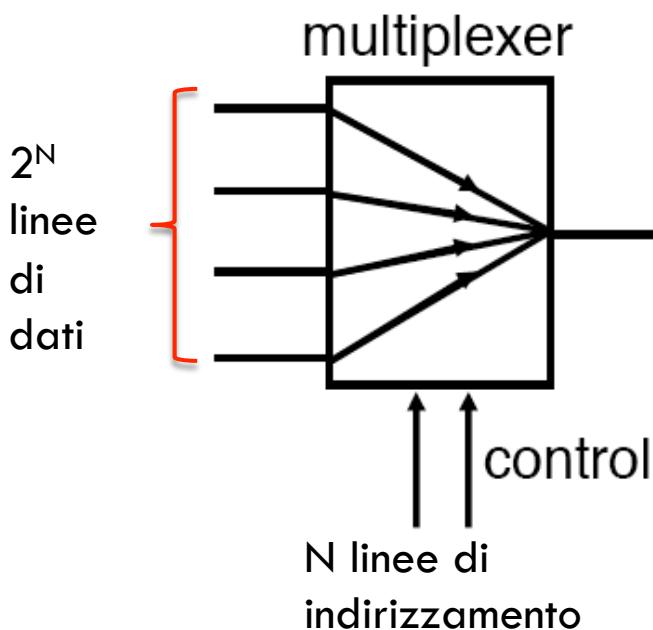
17



Implementazione a Look-up table: Multiplexer

18

- E' un circuito che convoglia una delle N possibili linee di ingresso sulla sua unica uscita in base al valore delle linee di indirizzo.
- Si chiama anche selettore, puo` essere utilizzato per conversione Par/ser
- 2^N ingressi per i dati; N linee per l'indirizzamento; 1 linea di uscita
- Due alternative per esprimere il funzionamento con la tabella della verita: funzionale e logica



Esempio: 2:1 multiplexer

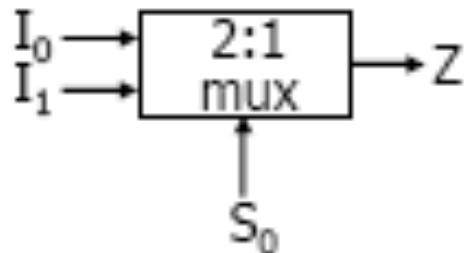
Functional truth table		Logical truth table		
S	Z	In ₁	In ₀	S
0	In ₀	0	0	0
1	In ₁	0	0	1
		0	1	0
		0	1	1
		1	0	0
		1	0	1
		1	1	0
		1	1	1

Below the tables, a logic symbol for a 2:1 MUX is shown with its inputs and output:

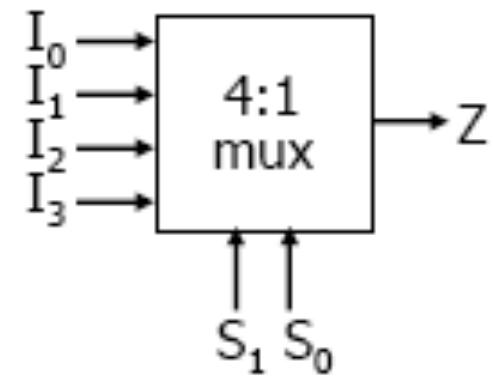
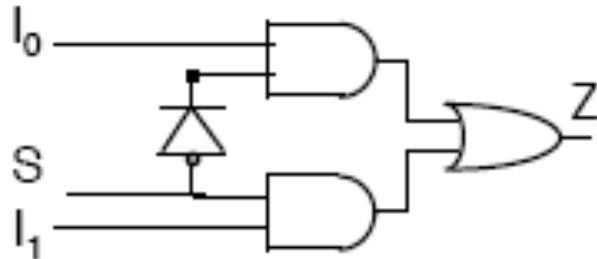
```
graph LR; I0((I0)) --> M[2:1 mux]; I1((I1)) --> M; S0((S0)) --> M; M --> Z((Z))
```

Implementazione dei multiplexer 2:1, 4:1

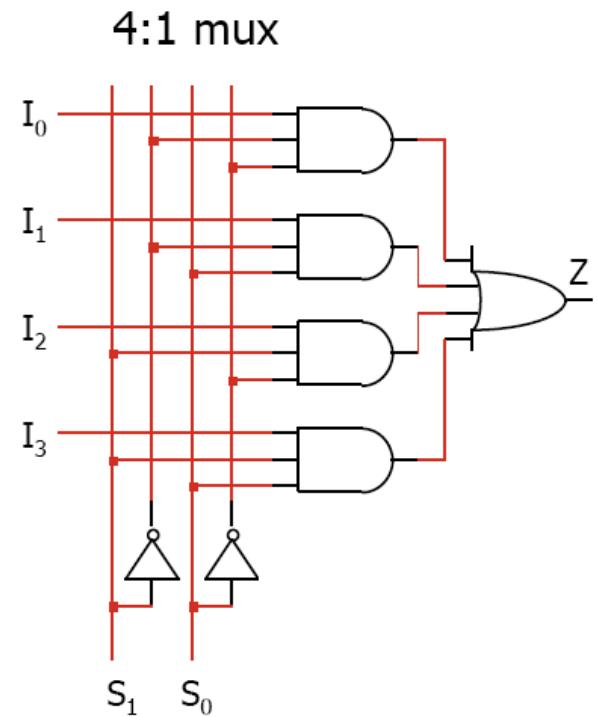
19



$$Z = \bar{S}I_0 + SI_1$$



$$Z = \overline{S_0}\overline{S_1}I_0 + S_0\overline{S_1}I_1 + \overline{S_0}S_1I_2 + S_0S_1I_3$$

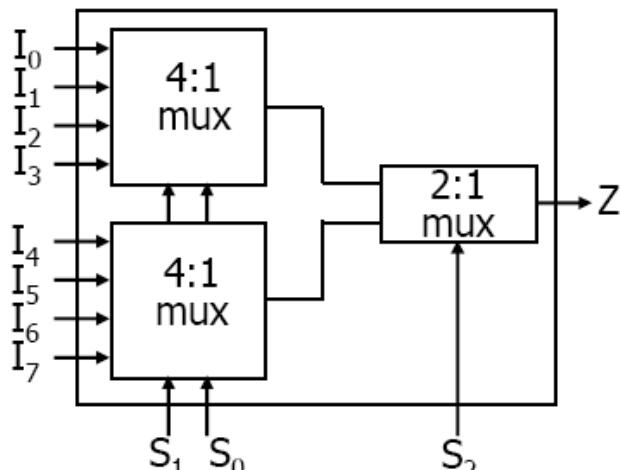


Aumento della dimensione dei multiplexer

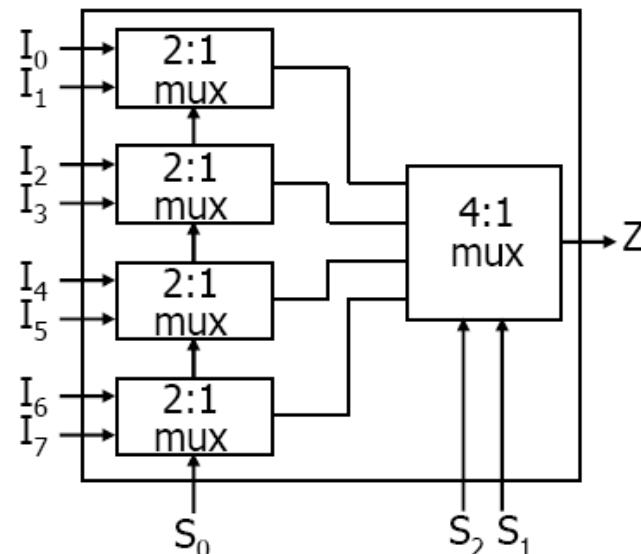
20

- Nel caso di logica TTL il multiplexer più grande ha 16 ingressi.
- Questa limitazione e` dovuta essenzialmente alla dimensione del chip. Nro di pin: 16 linee di ingresso + 4 linee di indirizzamento + 1 pin di uscita + V_{cc} + ground = 23 pins
- Si ottengono però multiplixer di dimensione N:1 utilizzando piu' multiplexer di dimensioni minore collegati in cascata

8:1 mux



8:1 mux



Multiplexer per implementare circuiti combinatori

21

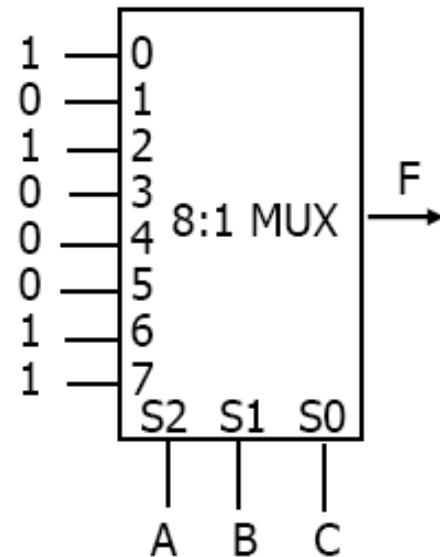
Un multiplexer $2^N:1$ puo` implementare qualsiasi funzione di N variabili.

Esempio: MUX per implementare 8:1 per implementare una funzione di 3 variabili.

$$F(A,B,C) = \overline{ABC} + \overline{ABC} + ABC + ABC$$

Definisco a 1 tutte le linee che corrispondono ad un termine della funzione F.

La funzione F e` espressa come OR di AND (minterm)



In pratica il Multiplexer implementa una generica somma di prodotti (OR di AND) che puo` essere “programmata” definendo quale dei minterm (AND) deve essere attivo tramite il valore degli ingressi I_i

Multiplexer per implementare circuiti combinatori

22

Trucco: un multiplexer $2^{N-1}:1$ puo` implementare qualsiasi funzione di N variabili.

Esempio: MUX 4:1 per implementare una funzione di 3 variabili

$$F(A,B,C) = \overline{ABC} + \overline{AB}\overline{C} + A\overline{B}\overline{C} + ABC$$

Considero 2 delle 3 variabili: A, B

Per ogni combinazione di A,B definisco F in funzione del valore di C.

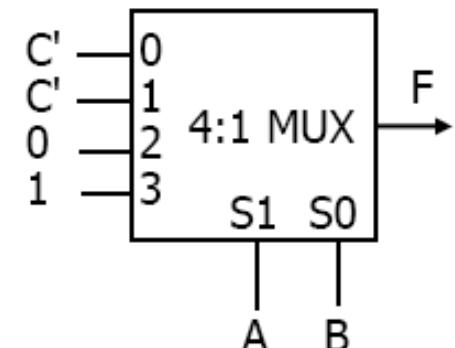
In Tabella $C' = \text{NOT}(C)$.



A	B	C	F	
0	0	0	1	C'
0	0	1	0	
0	1	0	1	C'
0	1	1	0	
1	0	0	0	0
1	0	1	0	
1	1	0	1	1
1	1	1	1	

$$F(A,B,C) = \overline{AB}(\overline{C}) + \overline{AB}(\overline{C}) + A\overline{B}(0) + AB(1)$$

Quindi la funzione F puo` essere implementata con un MUX 4:1 con ingressi 0,1,C' corrispondenti ai vari indirizzi di AB



MUX per implementare circuiti combinatori

23

- Le considerazioni che abbiamo fatto possono essere generalizzate per una funzione di N variabili
- Un MUX $2^{N-1}:1$ puo` implementare qualsiasi funzione di N variabili

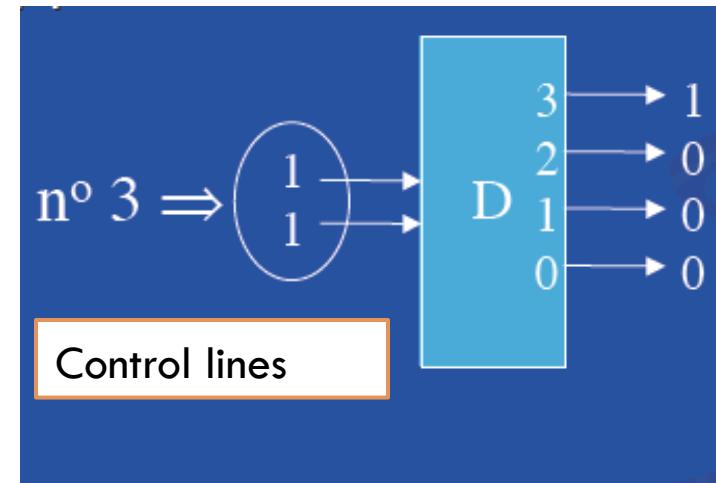
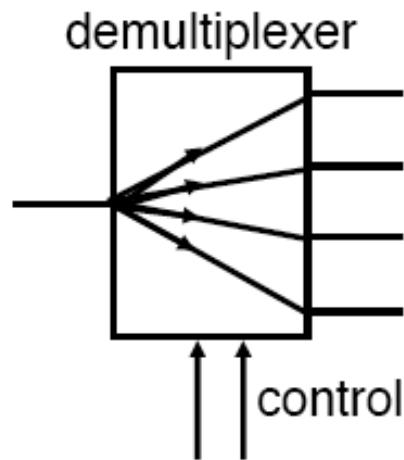
I_0	I_1	...	I_{n-1}	I_n	F			
.	.	.	.	0	0	0	1	1
.	.	.	.	1	0	1	0	1
					↓	↓	↓	↓
					0	I_n	I'_n	1

Variabili per indirizzo

Template-based logic: Demultiplexer

24

- Il demultiplexer fa la funzione opposta al Multiplexer: convoglia il suo unico ingresso su una delle sue 2^N uscite a seconda del valore degli N bit di indirizzo
- Il Decoder e` un particolare tipo di MUX in cui la linea di ingresso e` utilizzata come Enable (1).



Implementazione del demultiplexer

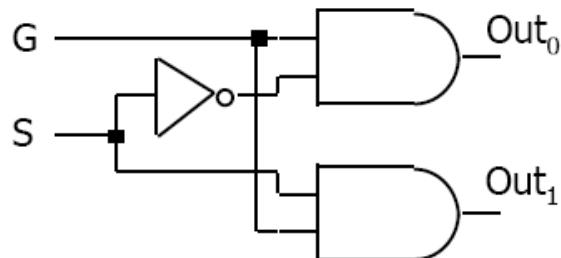
25

1:2 Decoder:

$$\text{Out}_0 = G \bullet S'$$

$$\text{Out}_1 = G \bullet S$$

1:2 demux



2:4 Decoder:

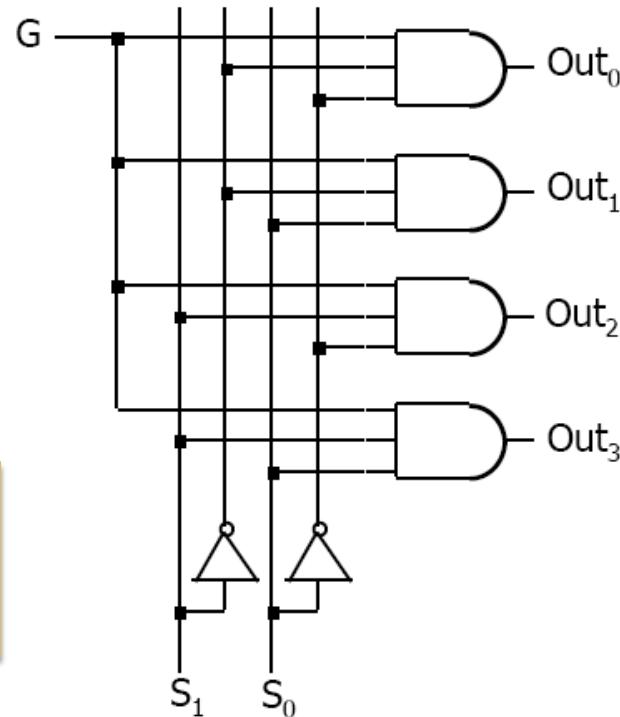
$$\text{Out}_0 = G \bullet S_1' \bullet S_0'$$

$$\text{Out}_1 = G \bullet S_1' \bullet S_0$$

$$\text{Out}_2 = G \bullet S_1 \bullet S_0'$$

$$\text{Out}_3 = G \bullet S_1 \bullet S_0$$

2:4 demux



Convenzione sulla nomenclatura

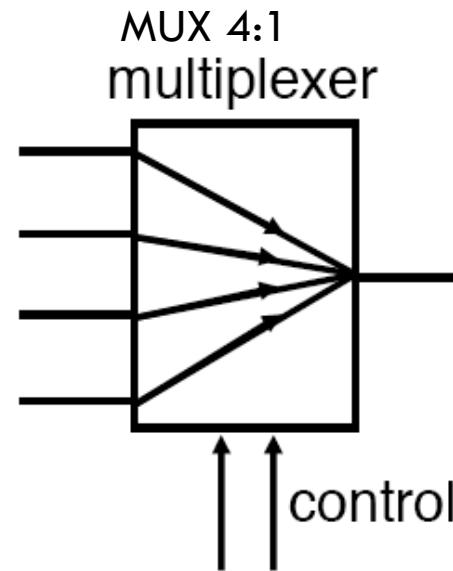
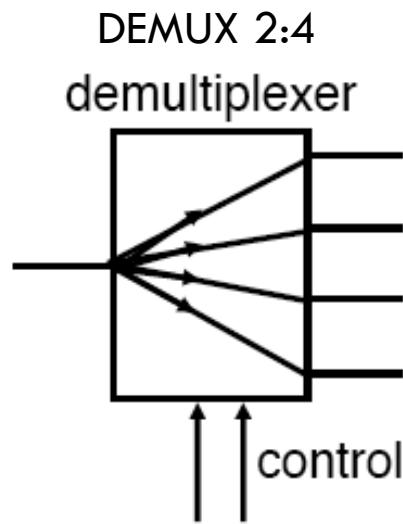
DEMUX: nro linee controllo:nro uscite

MUX: nro ingressi:nro uscite

MUX - DEMUX

26

□ Demoltiplicano o moltiplicano l'informazione portata dalla linea singola.

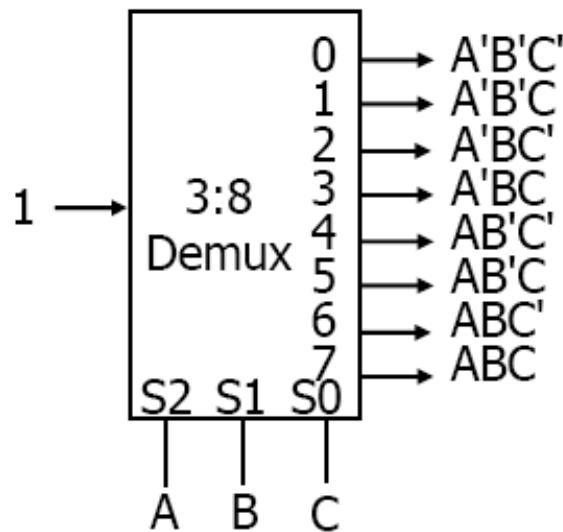


Convenzione sulla nomenclatura
DEMUX: nro linee controllo:nro uscite
MUX: nro ingressi:nro uscite

DEMUX per implementare circuiti combinatori

27

- E' il primo esempio di circuito che implementa la logica template-based: utilizzo un circuiti generico che "personalizzo" a seconda della funzione che devo implementare
- Per il DEMUX:
 - Utilizzo le variabili per definire tutti i mintern in funzione delle variabili indirizzo
 - Utilizzo porte OR per selezionare i minterm di cui ho bisogno



DEMUX per implementare circuiti combinatori

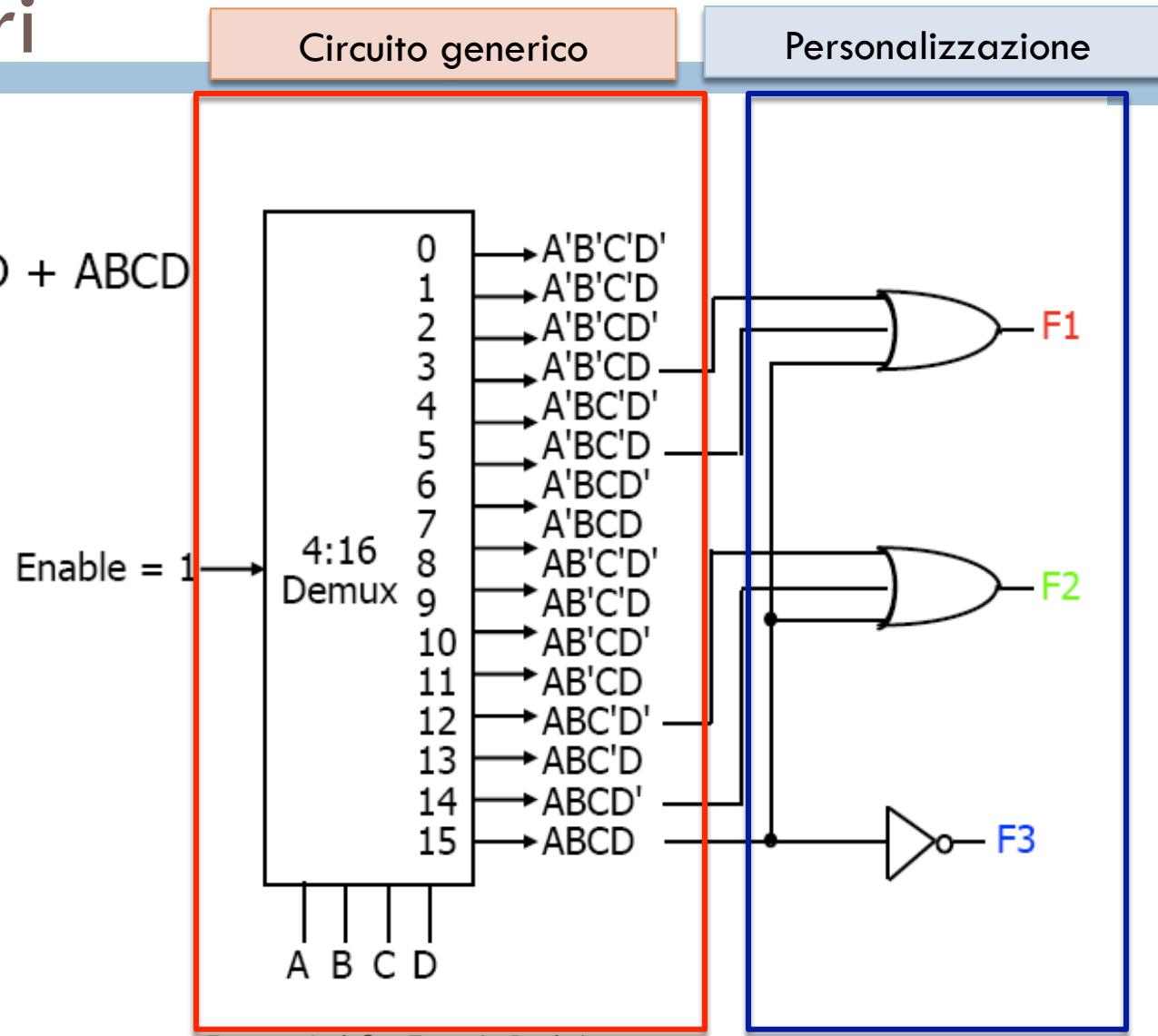
28

Example

$$F_1 = A'B'C'D + A'B'CD + ABCD$$

$$F_2 = ABC'D' + ABC$$

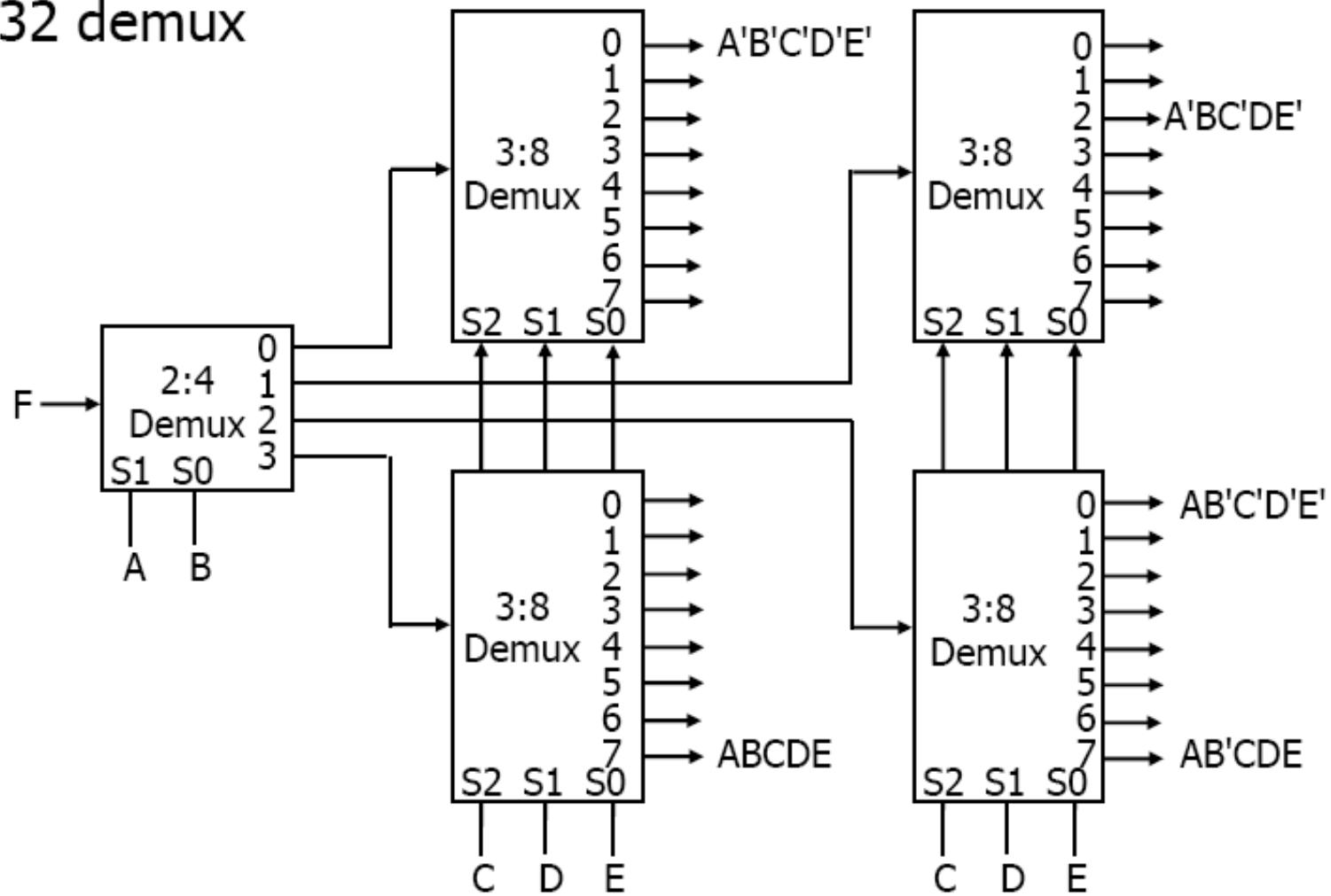
$$F_3 = (A' + B' + C' + D')$$



Aumento della dimensione dei DEMUX

29

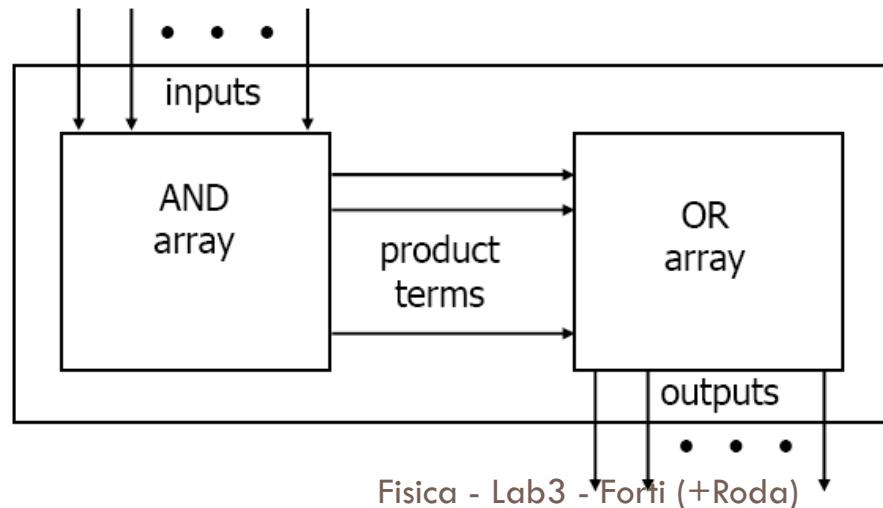
5:32 demux



Programmable Logic (PLAs & PALs)

30

- Gli integrati logici programmabili sono una generalizzazione dei demultiplexer ed una specie particolare di ROM
- Consistono di un livello di AND seguito da un livello di OR piu` una eventuale inversione
- Al contrario dei demux e delle ROM non tutti le possibili combinazioni degli AND degli ingressi sono fornite automaticamente attraverso un decoder generico
- Si formano solo i prodotti (AND) che necessitano e poi si combinano come con le connessioni OR
- Si programma la funzione desiderata eliminando o meno le connessioni

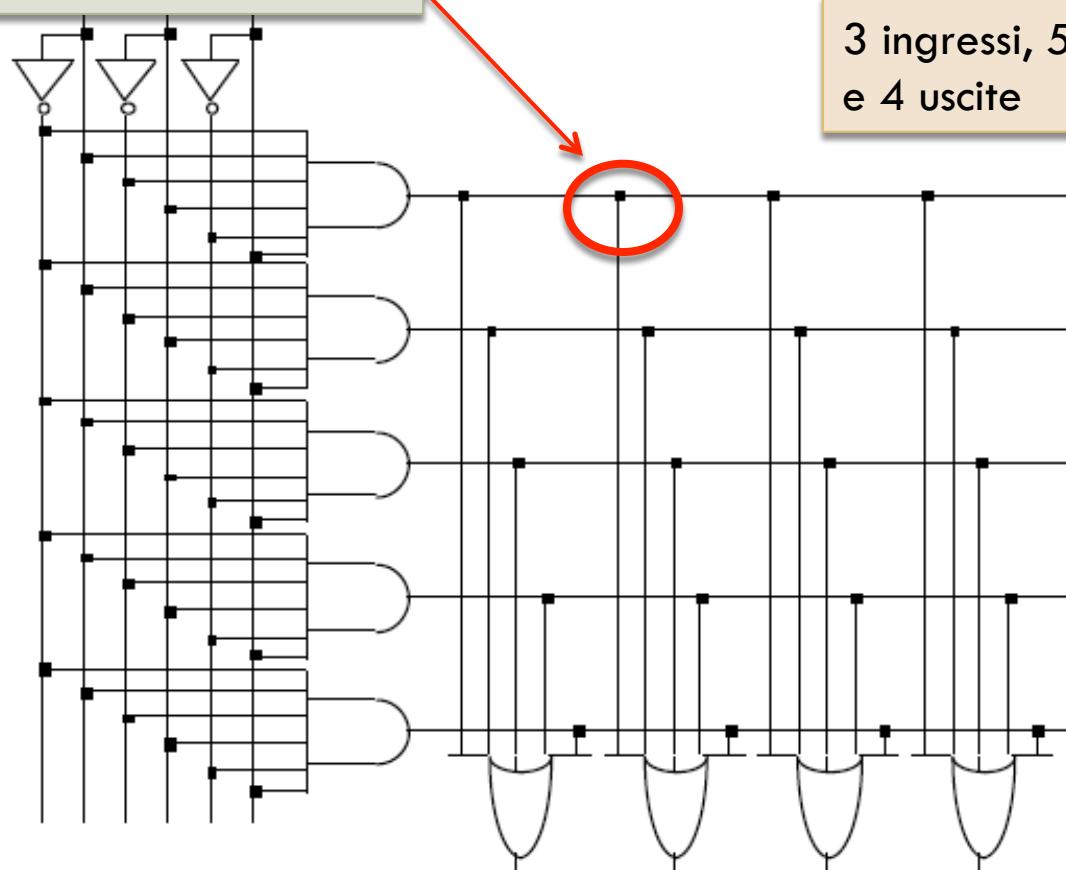


Programmable Logic Array: PLA

31

Tutti questi punti
rappresentano connessioni
programmabili

3 ingressi, 5 prodotti,
e 4 uscite



PLA,PAL verso altre possibili scelte

32

- Se la funzione da implementare ha molti ingressi e non tutti i termini prodotto utilizzati è più pratico utilizzare una PAL, PLA che un demux o una ROM
- Normalmente per funzioni complesse è anche la scelta più economica
- Si hanno anche in questo caso circuiti riprogrammabili sul campo: FPLA, FPAL – field programmable logic array...
- Un tipico esempio di TTL FPLA può avere 16 ingressi, 48 termini prodotto, ed 8 uscite. Questo integrato con 24 pins di dati contiene l'equivalente di 48 AND a 16 ingressi e 8 OR a 48 ingressi. Se si considera che un integrato std Small Scale Integration (SSI) con 18 pin di dati contiene 4 porte a 4 ingressi si capisce l'enorme vantaggio di integrazione
- D'altro canto non ha molto senso implementare funzioni di pochi ingressi con una PLA,PAL

Dimensione della PLA

- Supponiamo di dovere implementare le funzioni

$$F_0 = A + B'C'$$

$$F_1 = AC' + AB$$

$$F_2 = B'C' + AB$$

$$F_3 = B'C + A$$

- Possiamo caratterizzare il circuito da implementare per:
 - nro di variabili (A,B,C) -> nro di ingressi
 - nro di prodotti unici diversi (A, B'C', AC', AB, B'C) -> uscite dal livello di AND
 - nro di funzioni (F0,F1, F2, F3)-> nro di uscite
 - → abbiamo bisogno almeno di una PLA a 3 ingressi, 5 uscite dal livello degli AND, e 4 uscite

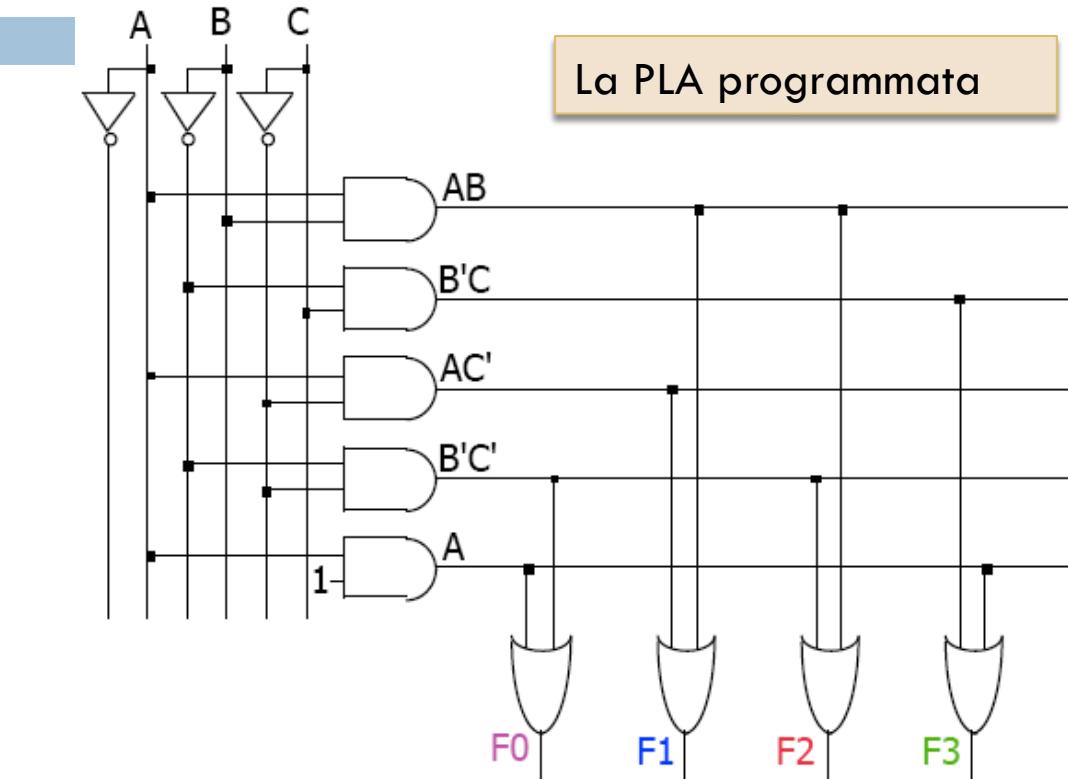
Dimensione della PLA

$$F_0 = A + B'C'$$

$$F_1 = AC' + AB$$

$$F_2 = B'C' + AB$$

$$F_3 = B'C + A$$



La PLA programmata

SW di programmazione

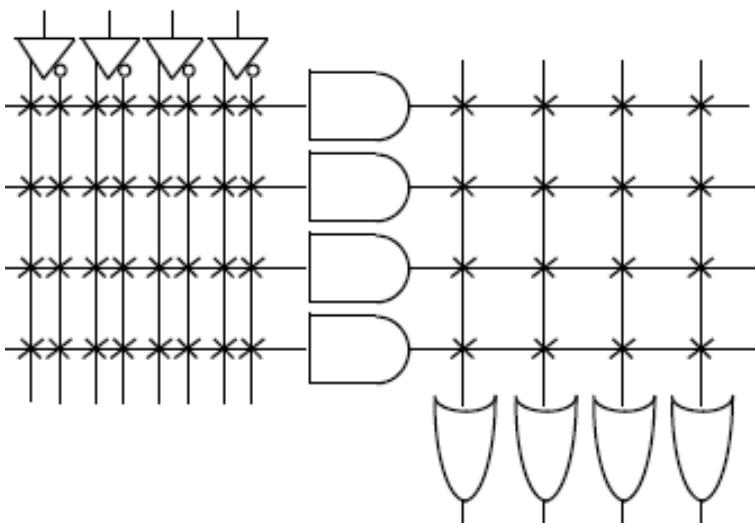
Normalmente si hanno dei programmi che analizzano la funzione che si vuole implementare e decide quale siano le connessioni che devono essere tenute e quali no. Una tecnica std utilizza fusibili che vengono bruciati con una alta corrente per interrompere le connessioni.

Notazione standard per PLA

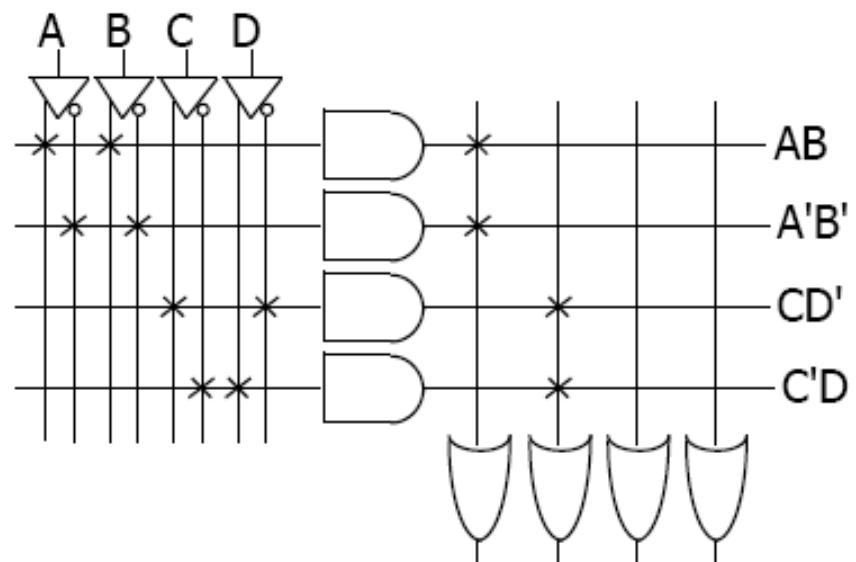
35

- Fili multipli connessi alla stessa porta logica sono disegnati come fili singoli
- Le X rappresentano connessioni, no-X interruzioni

Prima della programmazione



Dopo la programmazione



$$\begin{aligned}F0 &= AB + A'B' \\F1 &= CD' + C'D\end{aligned}$$

Esempio: un generatore di funzioni con PLA

Un circuito che implementa varie funzioni di variabili a 3 ingressi

$$F_1 = ABC$$

$$F_2 = A + B + C$$

$$F_3 = A' B' C'$$

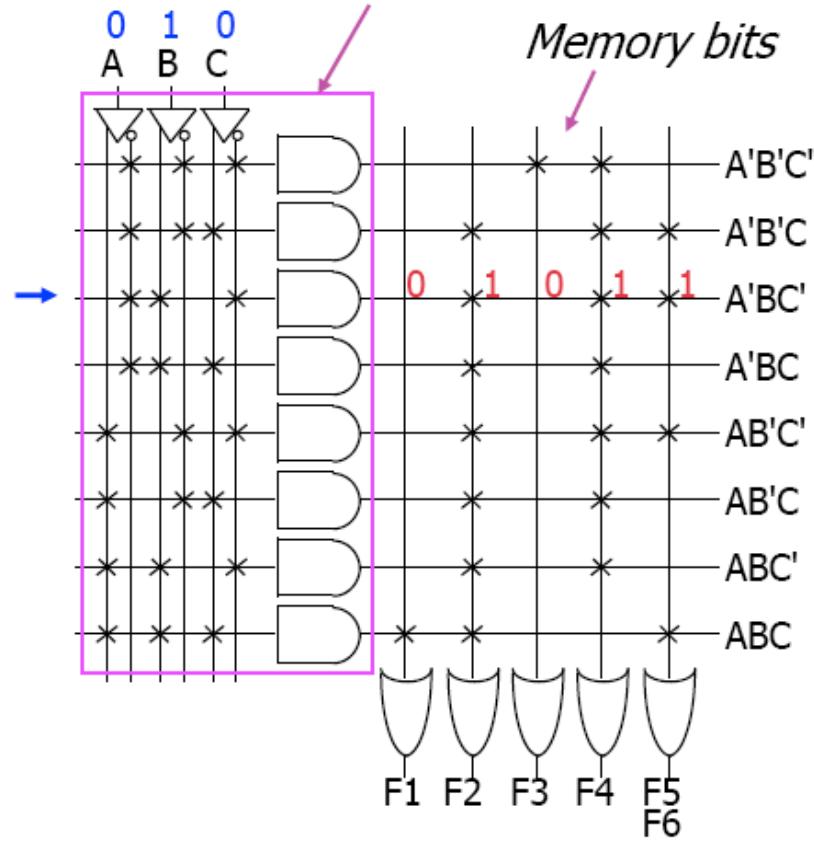
$$F_4 = A' + B' + C'$$

$$F_5 = A \text{ xor } B \text{ xor } C$$

$$F_6 = A \text{ xnor } B \text{ xnor } C$$

A	B	C	F1	F2	F3	F4	F5	F6
0	0	0	0	0	1	1	0	0
0	0	1	0	1	0	1	1	1
0	1	0	0	1	0	1	1	1
0	1	1	0	1	0	1	0	0
1	0	0	0	1	0	1	1	1
1	0	1	0	1	0	1	0	0
1	1	0	0	1	0	1	0	0
1	1	1	1	1	0	0	1	1

Think of as a memory-address decoder

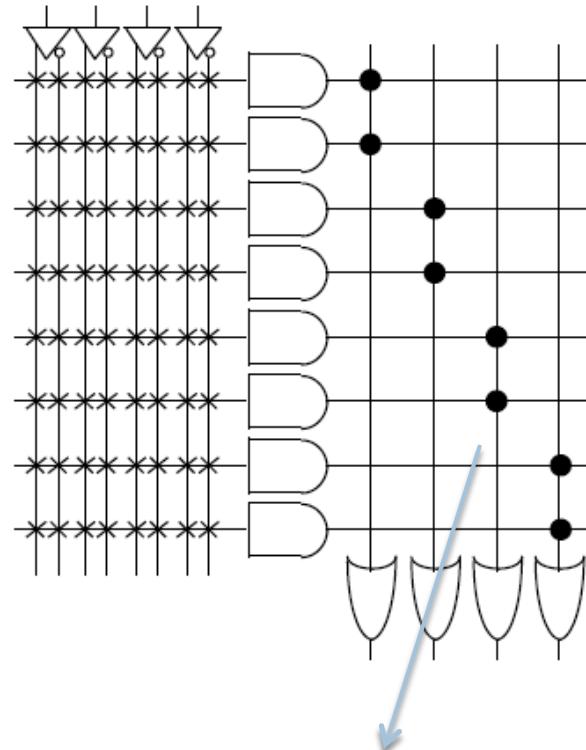


PLA verso PAL

37

- Le PLA che abbiamo visto sono completamente programmabili
- Le PAL hanno solo il livello degli AND programmabili
- Normalmente le PAL si caratterizzano per il nro di ingressi degli OR: a 2, 4, 8 o 16 ingressi
- Maggiore il nro degli ingressi, minore il nro delle uscite
- Le PAL sono più veloci e meno care delle PLA ma meno versatili
- La maggiore velocità è dovuta a meno connessioni programmabili. Le connessioni programmabili hanno una resistenza maggiore delle connessioni std e quindi un maggiore ritardo. E' proprio la caratteristica di maggiore velocità che ha portato allo sviluppo delle PAL
- Nelle PAL non si possono condividere termini prodotto

PAL: 4 ingressi, 8 prodotti, 4 uscite



Connessioni non
programmabili

Altre rappresentazioni utili: Codice Gray

- Codice Gray: è costruito in modo che un solo bit cambi passando tra stati contigui. Molto utile per evitare errori nella trasmissione di dati.

Decimal Symbols	Gray Code
0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101

Un metodo per costruirlo: si scrivono 0 e 1 decimale in modo std; riflessione + inversione 1/0 nella seconda colonna da dx; riflessione + inversione 1/0 nella terza colonna da dx ...

0000	→	0	
0001	→	1	
-----	-----	-----	prima riflessione
0011	→	2	
0010	→	3	
-----	-----	-----	seconda riflessione
0110	→	4	
0111	→	5	
0101	→	6	
0100	→	7	

Esempio: convertitore BCD -> Gray

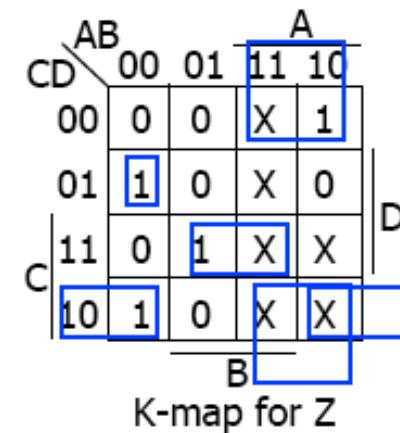
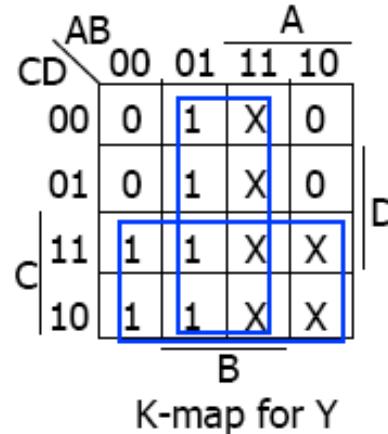
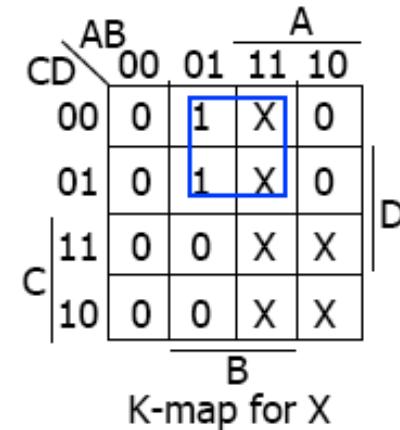
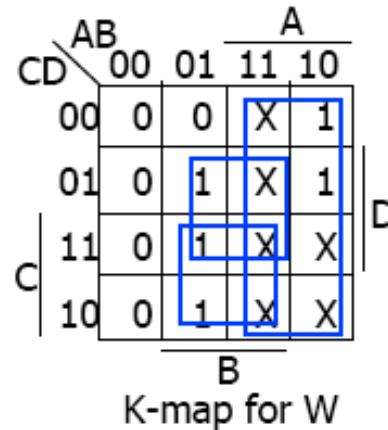
39

A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	1	1	1	0
0	1	1	0	1	0	1	0
0	1	1	1	1	0	1	1
1	0	0	0	1	0	0	1
1	0	0	1	1	0	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

ABCD = Numero a 4 bit BCD

WXYZ = Numero a 4 bit in codice

Gray



BCD -> Gray con una PLA

40

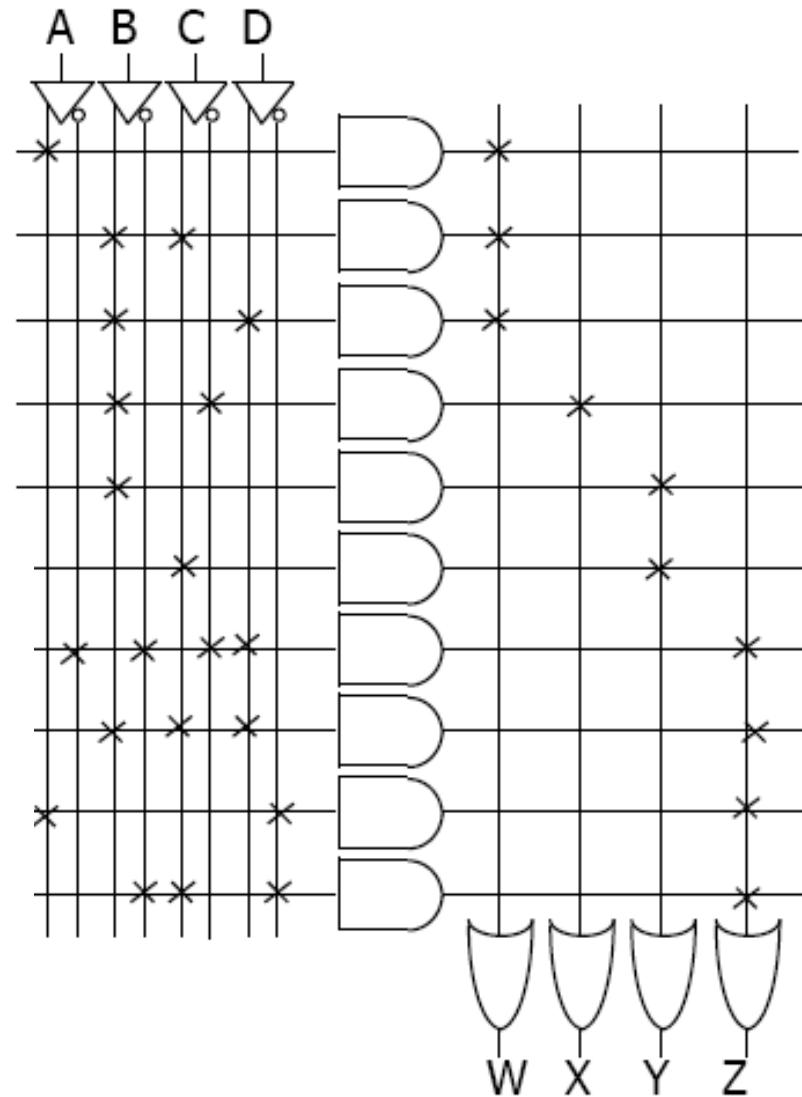
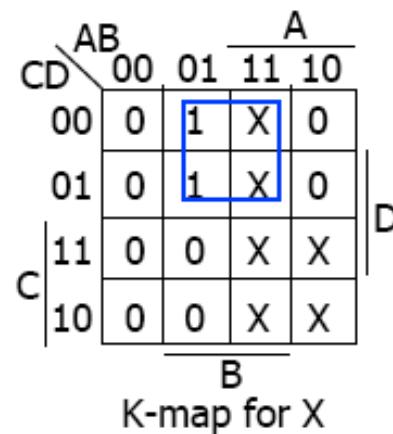
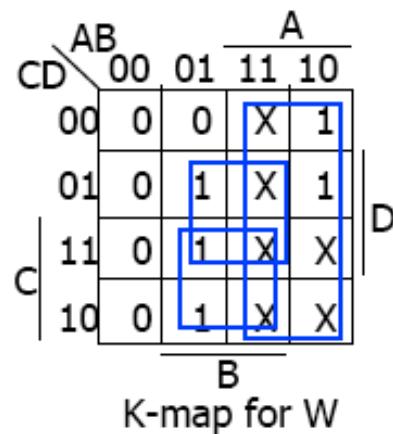
Minimized functions:

$$W = A + BC + BD$$

$$X = BC'$$

$$Y = B + C$$

$$Z = A'B'C'D + BCD + AD' + B'CD'$$



BCD -> Gray con una PAL

41

Example: Wire a PAL

Minimized functions:

$$W = A + BC + BD$$

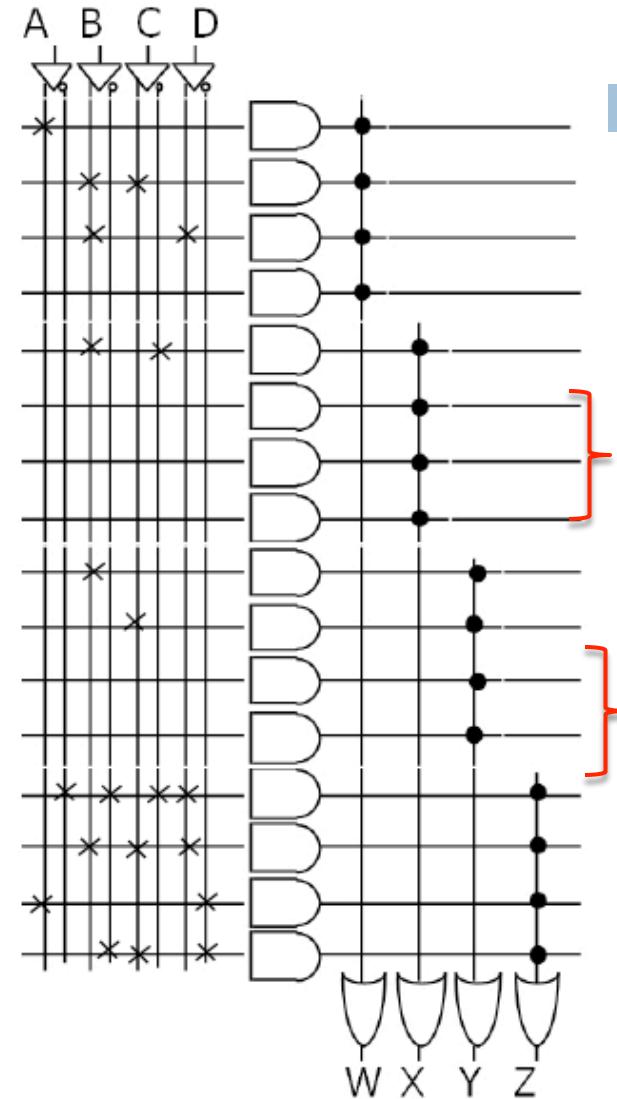
$$X = BC'$$

$$Y = B + C$$

$$Z = A'B'C'D + BCD + AD' + B'CD'$$

Buon esempio per l'utilizzo di una PAL perché non ci sono termini prodotto in comune
Poiché Z è formata da 4 termini prodotto, dobbiamo scegliere una PAL con OR a 4 ingressi -> $4 \times 4 = 16$ livelli di AND

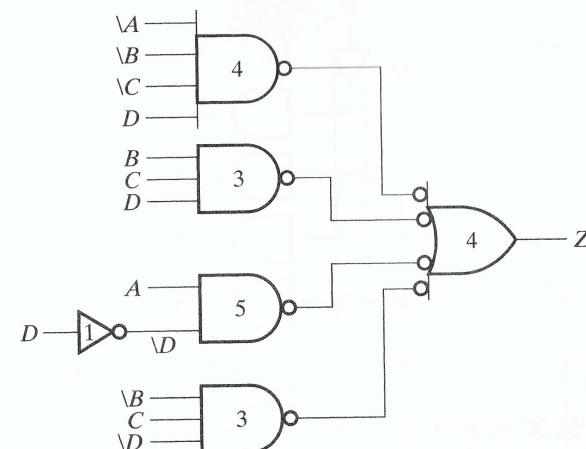
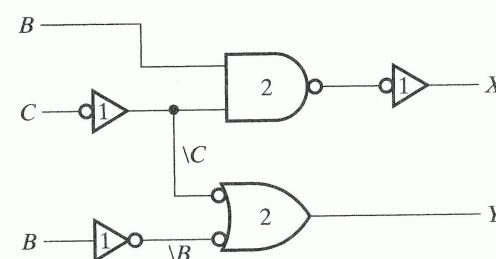
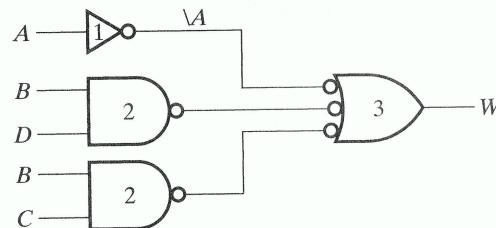
Nonostante molti gate siano sprecati l'utilizzo della PAL risulta più economico e veloce



PAL,PLA vs Standard Parts

42

- L'implementazione con logica programmabile implementa il convertitore con un solo integrato
- Vediamo l'implementazione con porte std TTL. Consideriamo l'implementazione con NAND e NOT
- Si devono utilizzare 15 porte in 5 integrati

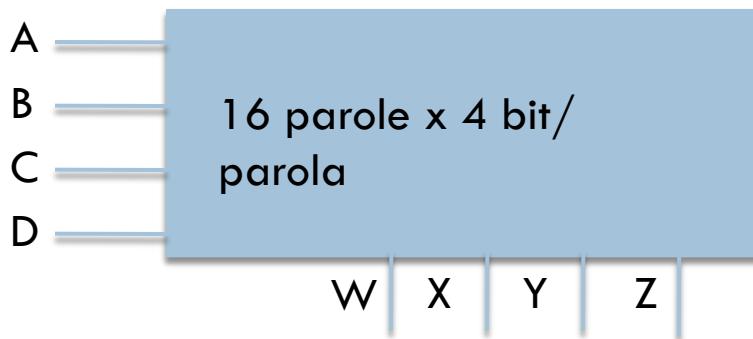


1: 7404 hex inverters
2,5: 7400 quad 2-input NAND
3: 7410 tri 3-input NAND
4: 7420 dual 4-input NAND

PLA, PAL vs ROM

43

- Simile ad una PLA ma con tutti gli stati degli indirizzi decodificati
- L'array di OR e` completamente flessibile (a differenza della PAL)
- Esempio: implementazione della conversione BCD -> Gray



PRO ROM:

Veloci da implementare, semplici e dense

A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	1	1	1	0
0	1	1	0	1	0	1	0
0	1	1	1	1	0	1	1
1	0	0	0	1	0	0	1
1	0	0	1	1	0	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

Contro ROM:

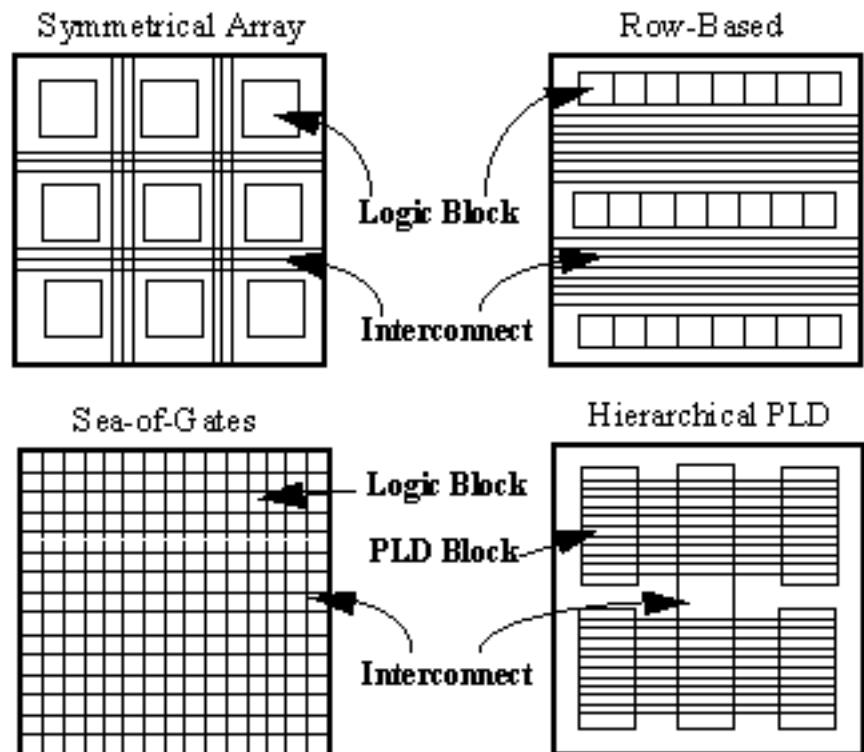
Non si puo` approfittare delle combinazioni non definite "x"

La dimensione raddoppia per ogni bit di indirizzo in piu'

FPGA

44

- Le funzioni logiche più complesse richiedono un numero molto grande di minterms
 - PLA/PAL raggiungono facilmente limiti di dimensioni
- Sviluppo di sw di ottimizzazione con funzioni più complesse
- Matrici programmabili con elementi molto più sofisticati
- Elemento base: CLB (Configurable Logic Block)

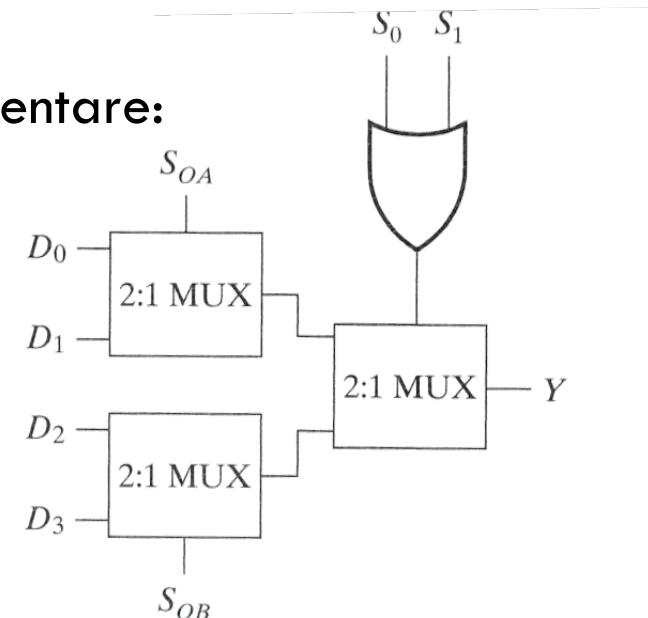


FPGA basati su Multiplexer

45

- Un modello alternativo di circuito logico programmabile e` basato su multiplexer
- $D_0 \dots D_3, S_{OA}, S_{OB}, S_0, S_1$ sono ingressi programmabili
- Y e` l'unica uscita
- Questo semplice **modulo logico** puo` implementare:
 - Tutte le funzione di 1 e 2 variabili
 - 223 delle 256 funzioni a 3 variabili
 - La maggior parte delle funzioni a 4 variabili
 - Alcune a 5
 - Poche a 6 e 7
 - Ed 1 a 8
- La funzione a 8 variabili:

$$Y = \overline{(S_0 + S_1)} \left(\overline{S_{OA}} D_0 + S_{OA} D_1 \right) + (S_0 + S_1) \left(\overline{S_{OB}} D_2 + S_{OB} D_3 \right)$$

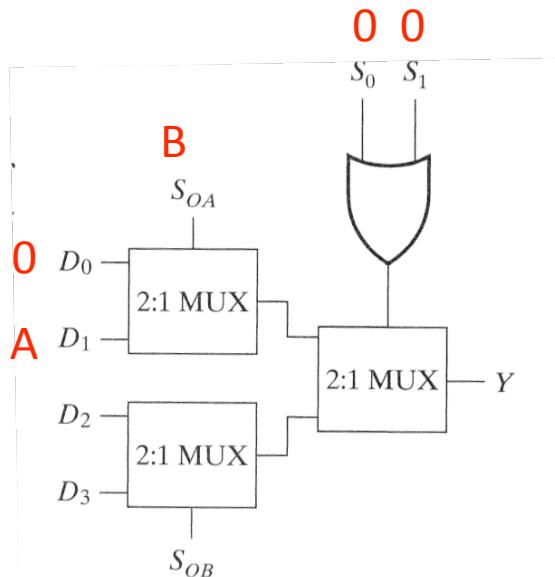


Utilizzo del modulo logico: esempio AND, XOR

46

- Implemento la funzione: $Y = \text{AND}(A, B)$

Se $B = 1 \rightarrow Y = A$
altrimenti e` 0



$$Y = \overline{(S_0 + S_1)}(\overline{S_{OA}}D_0 + S_{OA}D_1) + (S_0 + S_1)(\overline{S_{OB}}D_2 + S_{OB}D_3) =$$

$$Y = \overline{(0+0)}(\overline{B}0 + BA) + (0+0)(\overline{S_{OB}}D_2 + S_{OB}D_3)$$

$$Y = AB$$

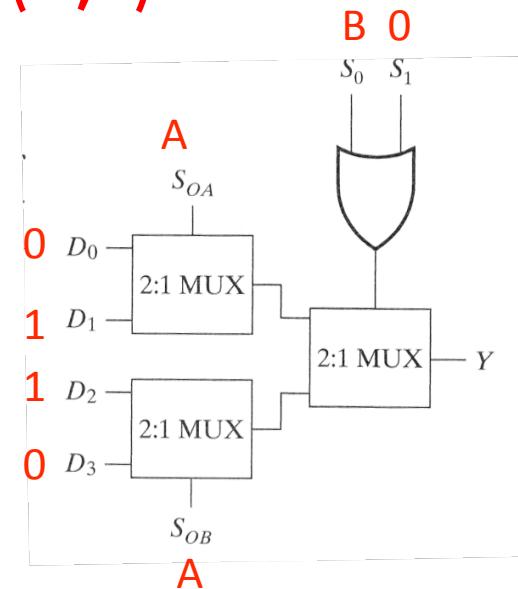
Utilizzo del modulo logico: esempio AND, XOR

47

- Implemento la funzione: $Y = \text{XOR}(A, B)$

I MUX S_{OA} genera A: se $A=0$
 $Y=0$, se $A=1$ $Y=1$. S_{OB} genera
in modo analogo Abar

Se $B=0$ seleziona A se $B=1$
selezione Abar



$$Y = (\overline{S_0 + S_1})(\overline{S_{OA}}D_0 + S_{OA}D_1) + (S_0 + S_1)(\overline{S_{OB}}D_2 + S_{OB}D_3)$$

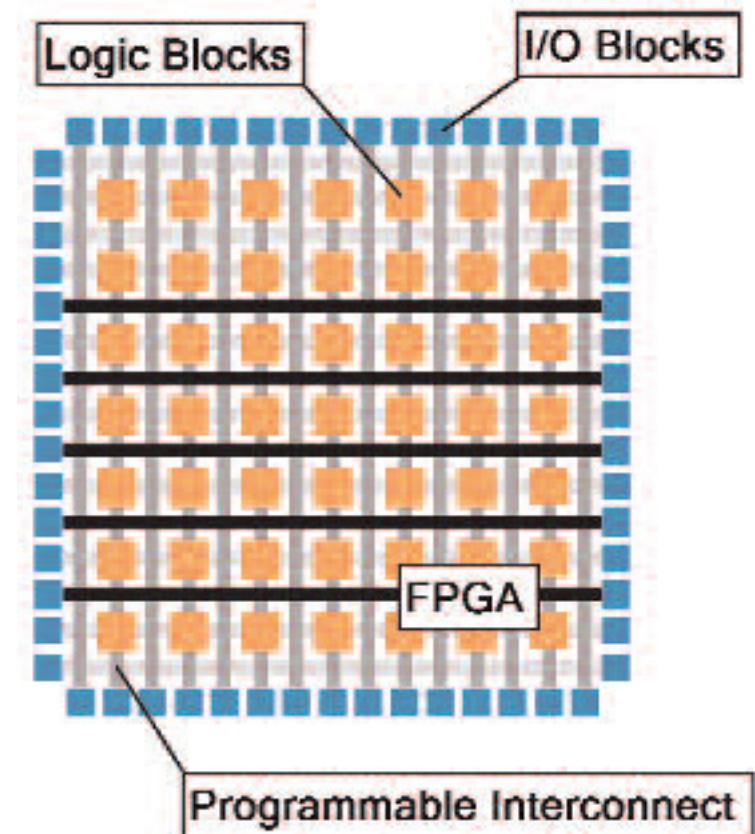
$$Y = (\overline{B+0})(\overline{A}0 + A1) + (B+0)(\overline{A}1 + A0)$$

$$Y = \overline{B}A + B\overline{A}$$

Utilizzo del CLB

48

- E` chiaro che questo modulo logico e` molto versatile e compatto.
- Per funzioni piu` complicate tuttavia si ha bisogno di sw dedicati che trovino la migliore implementazione possibile
- HDL + Sistemi di sviluppo



Dove si trovano gli argomenti svolti oggi

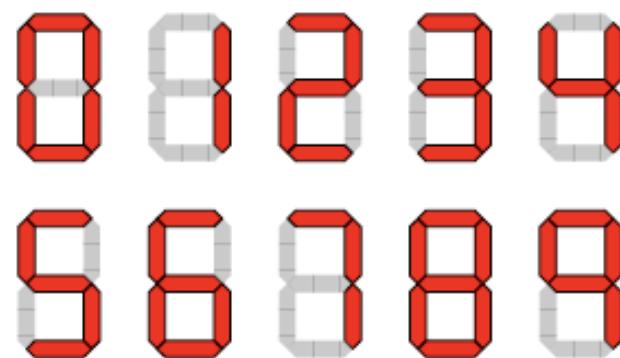
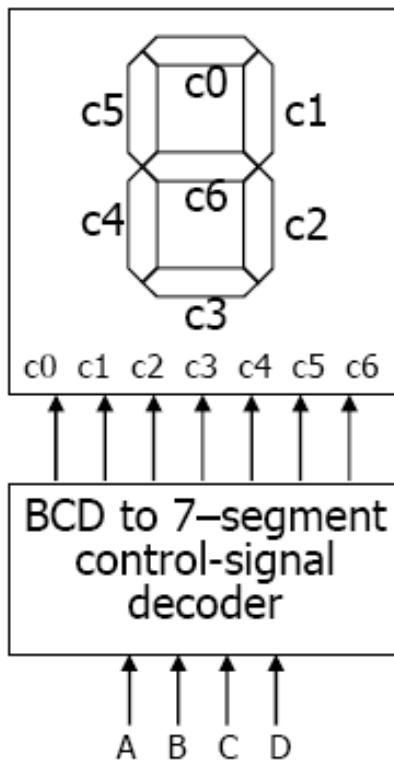
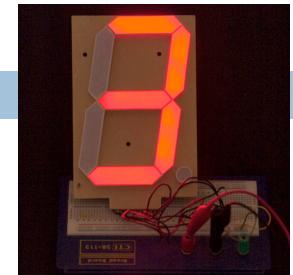
49

- Introduzione all'elettronica Parte I – E.Falchini et al.
- Microelectronics – I.Millman
- Contemporary Logic Design – Katz, Borriello

Un altro esempio: controllo per il display a 7 segmenti

50

- In ingresso 1 nro a 4 bit in BCD (A,B,C,D)
- in uscita 7 segnali di controllo C0...C7



Circuito di controllo per il display a 7 segmenti

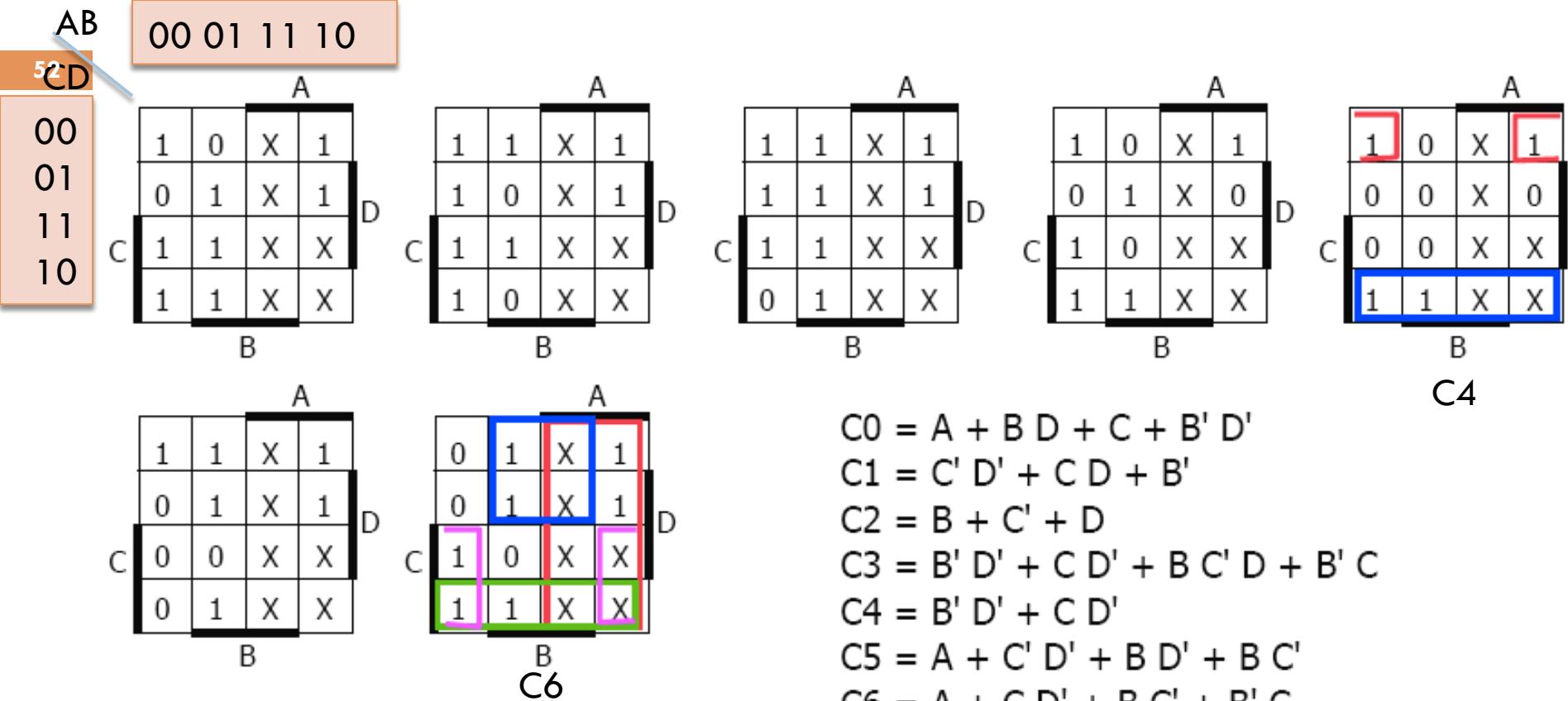
51

- Passo 1: Tabella della verita`
- Passo 2: si sceglie l'implementazione
 - ROM, non c'e` niente altro da fare che implementare la tab della verita`
 - Le "x" possono significare che PAL/PLA sia una scelta migliore
- Passo 3:
 - Minimizzazione
 - Mappatura in PLA/PAL

A	B	C	D	C0	C1	C2	C3	C4	C5	C6
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	0	0	1	1
1	0	1	X	X	X	X	X	X	X	X
1	1	X	X	X	X	X	X	X	X	X

Non tutte le linee sono descritte separatamente

Implementazione con Sum-Of-Products



- 15 termini di prodotti unici se si minimizza ogni Ci individualmente

Implementazione

53

- Se pensiamo di utilizzare una PAL (programmazione solo di AND) la nostra scelta deve essere basata su:
 - 4 ingressi
 - 7 uscite
 - OR di uscita almeno a 4 ingressi
- Nel data-sheet della PAL P16H8 ad esempio troveremmo specificato che ha 10 ingressi, 8 uscite formate da OR a 7 ingressi. OK per noi.
- Con una PLA dovremmo invece basare la nostra scelta su ingressi, uscite ed il nro di prodotti unici che sono 15 e che definisce il nro di porte AND necessario.
- Ma possiamo fare anche meglio...

Ottimizzazione con una PLA

54

- Utilizzando un programma di ottimizzazione ci fornirebbe una diversa soluzione per le equazioni booleane che descrivono il circuito:

		A	
		C2	D
		B	
1	1	X	1
1	1	X	1
1	1	X	X
0	1	X	X



Ma e` davvero migliore ?

		A	
		C2	D
		B	
1	1	X	1
1	1	X	1
1	1	X	X
0	1	X	X

$$C_0 = A + BD + C + B'D'$$

$$C_1 = C'D' + CD + B'$$

$$C_2 = B + C' + D$$

$$C_3 = B'D' + CD' + BC'D + B'C$$

$$C_4 = B'D' + CD'$$

$$C_5 = A + C'D' + BD' + BC'$$

$$C_6 = A + CD' + BC' + B'C$$



$$C_0 = BC'D + CD + B'D' + BCD' + A$$

$$C_1 = B'D + C'D' + CD + B'D'$$

$$C_2 = B'D + BC'D + C'D' + CD + BCD'$$

$$C_3 = BC'D + B'D + B'D' + BCD'$$

$$C_4 = B'D' + BCD'$$

$$C_5 = BC'D + C'D' + A + BCD'$$

$$C_6 = B'C + BC' + BCD' + A$$

Implementazione con una PLA

$$C_0 = BC'D + CD + B'D' + BCD' + A$$

$$C_1 = B'D + C'D' + CD + B'D'$$

$$C_2 = B'D + BC'D + C'D' + CD + BCD'$$

$$C_3 = BC'D + B'D + B'D' + BCD'$$

$$C_4 = B'D' + BCD'$$

$$C_5 = BC'D + C'D' + A + BCD'$$

$$C_6 = B'C + BC' + BCD' + A$$

In effetti la cosa e` migliore
perche`:

Il nro di termini prodotto unici
e` sceso da 15 -> 9

permettendo di scegliere un
integrato piu` piccolo e di
sfruttarlo meglio.

