Calcolatori Elettronici

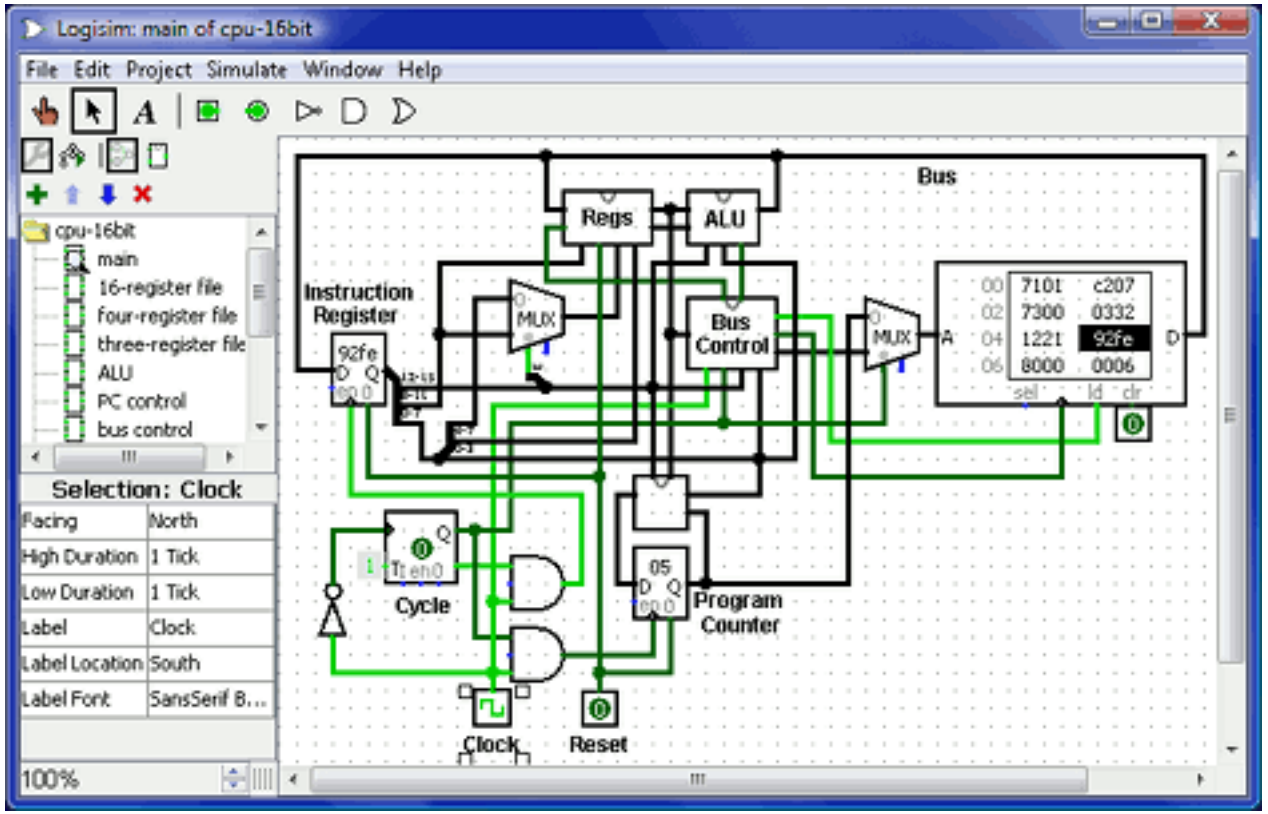
Parte IV: Logica Digitale e Memorie

Prof. Riccardo Torlone

Università di Roma Tre

Logisim

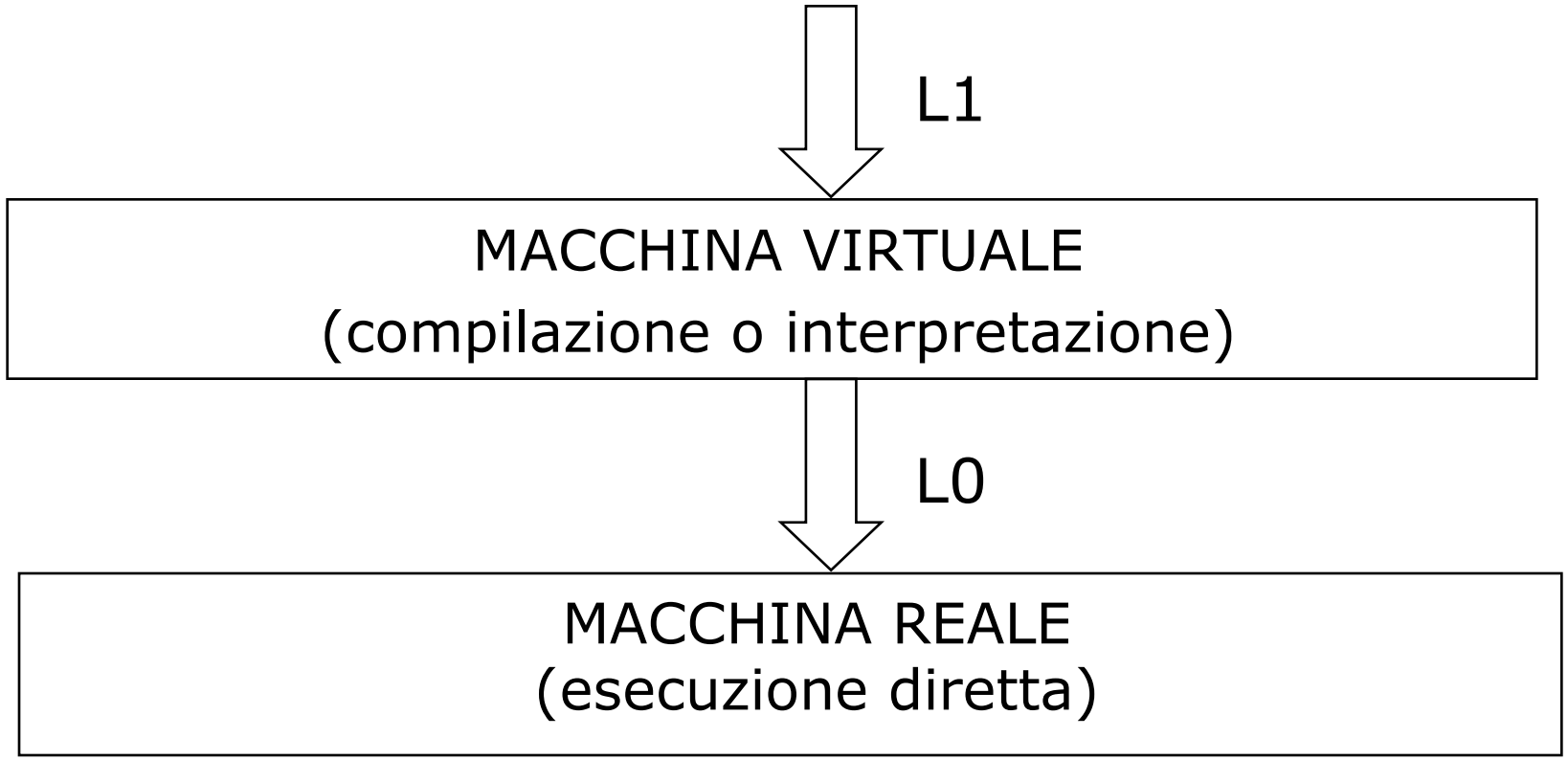
Uno strumento freeware per il progetto e la simulazione di circuiti digitali.



Molto utile per comprendere quello che vedremo a lezione.

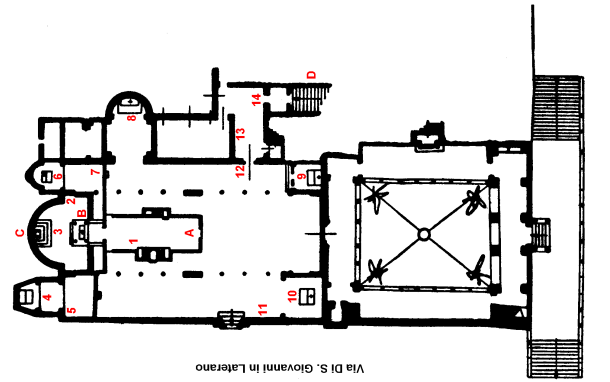
Verrà usato negli homework.

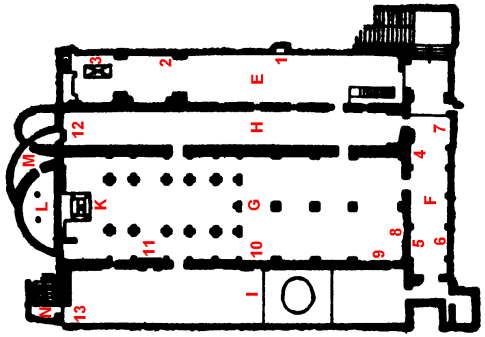
Astrazione di un calcolatore

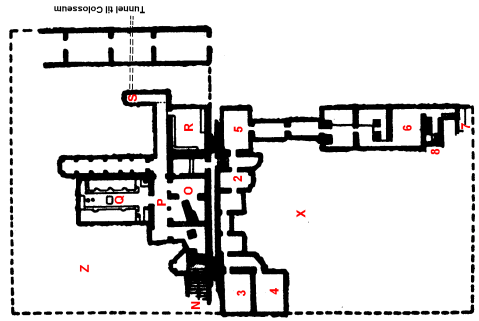


Se L0 ed L1 sono troppo diversi il problema si decompone introducendo livelli intermedi

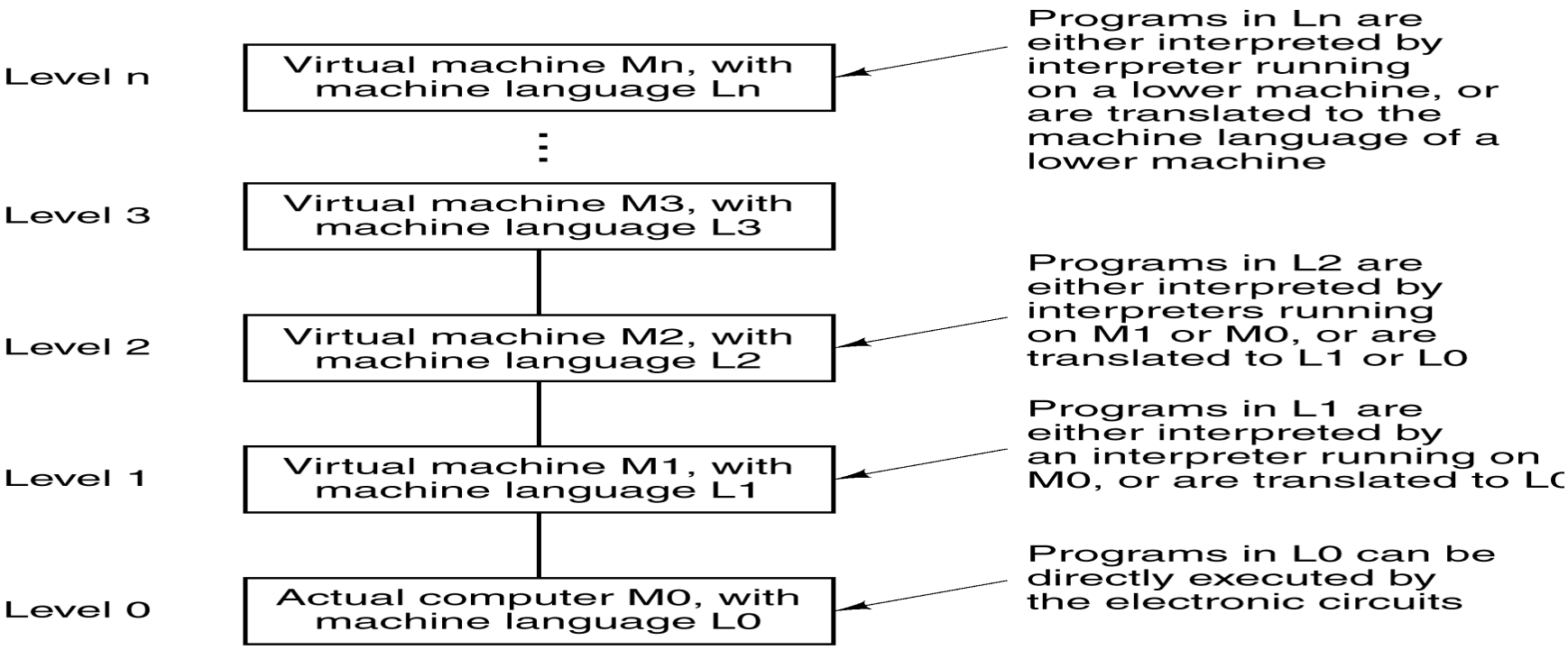
Un approccio stratificato







Architettura a livelli

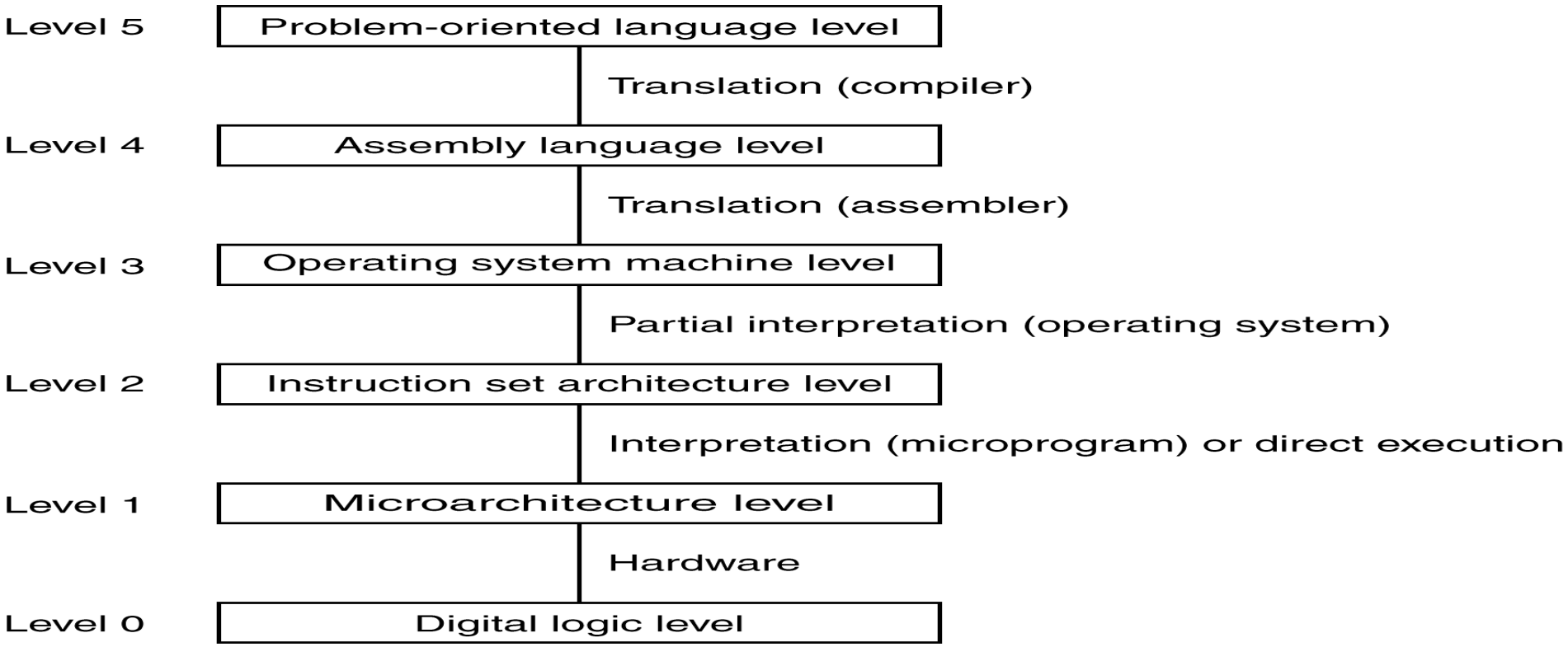


- Al livello i corrispondono una macchina virtuale M_i ed un linguaggio L_i
- Il linguaggio L_i è tradotto nel linguaggio L_{i-1} o interpretato da un programma che gira sulla macchina M_{i-1}

Perché la stratificazione?

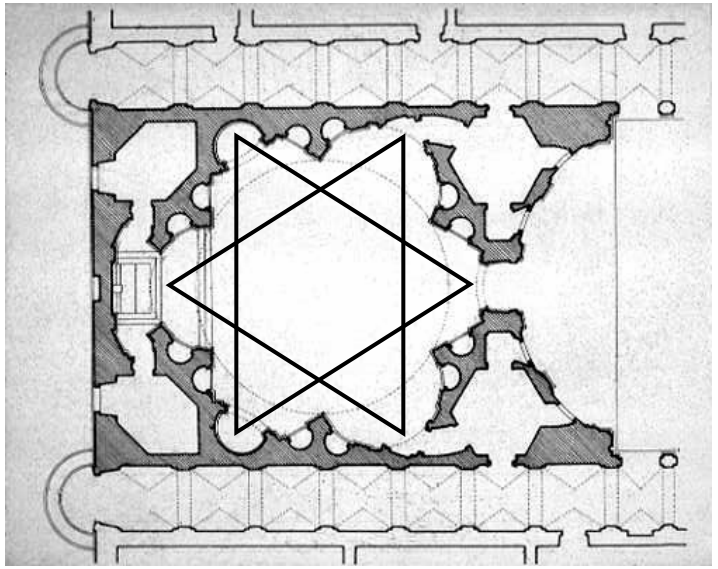
- M_0 è facilmente realizzabile in hardware, ma difficile da programmare
- M_n è facile da programmare ma impossibile da realizzabile in hardware
- Implementazione progressiva e modulare
- Trasparenza per l'utente e le applicazioni
- Il linguaggio L_n non dipende dalla piattaforma (hardware) M_0:
  - Diversi linguaggi disponibili sulla stessa piattaforma
  - Lo stesso linguaggio disponibile su diverse piattaforme

Tipica struttura a livelli

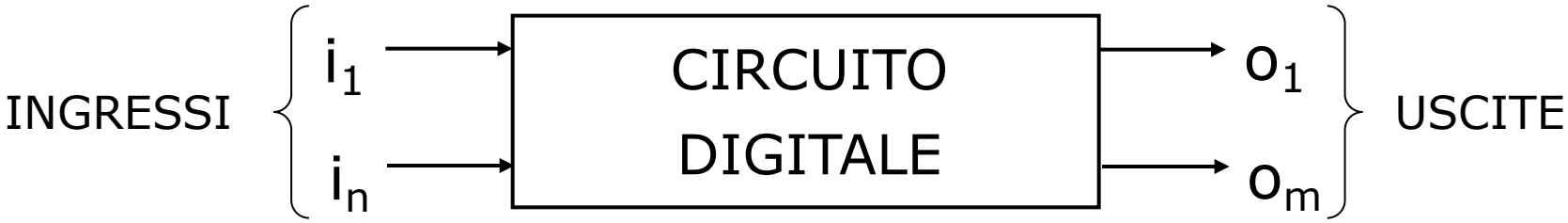


- Il livello 2 è il più basso al quale un utente può programmare la macchina (confine tra software e hardware)
- Normalmente si programma a livello 5

Semplici elementi alla base di sistemi complessi ...



Circuiti Digitali



- Circuiti elettronici i cui ingressi e le cui uscite assumono solo due livelli
- Al circuito sono associate le funzioni che calcolano le uscite a partire dagli ingressi

$$\begin{cases} o_1 = f_1(i_1,\ldots,i_n) \\ \vdots \\ o_m = f_m(i_1,\ldots,i_n) \end{cases}$$

Funzioni Logiche (Booleane)

- $y = f(x_1,\ldots,x_n)$ $\quad y, x_1,\ldots, x_n \in \{ 0,1 \}$

$$\{ 0,1 \}^n \xrightarrow{f} \{ 0,1 \}$$

- Variabili che possono assumere due soli valori:

$\{ 0, 1 \}$ **FALSO** **VERO** $\quad\quad$ $\{ F, T \}$ **FALSE** **TRUE**

- Definizione tramite tavola di verità:

x_1	x_2	.....	x_{n-1}	x_n	f
0	0	.....	0	0	0
0	0	.....	0	1	1
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
1	1	.....	1	1	0

- 2^n combinazioni di ingresso
- 2^{2^n} *funzioni distinte* di n variabili

Funzioni Booleane (Esempi)

- Con n=1 si hanno 4 funzioni:

x_1	f_0	f_1	f_2	f_3
0	0	0	1	1
1	0	1	0	1

- La funzione f_2 è detta **NOT**
- Con n=2 si hanno 16 funzioni, tra cui:

x_1	x_2	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7
0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1

- La funzione f_1 è nota come **AND**
- La funzione f_7 è nota come **OR**

Algebra Circuitale (Booleana)

- Rappresentazione algebrica di funzioni booleane

Insieme: $I = \{ 0,1 \}$
Operatori: AND , OR
Complementazione: NOT

Notazione

- Se x e y sono due variabili booleane:
  - L'AND di x e y si indica con $x \cdot y$ (o xy)
  - L'OR di x e y si indica con $x + y$
  - Il NOT di x si indica con $\overline{x}$

Espressioni Algebriche

Teorema: *ogni funzione booleana è* algebrica*, cioè rappresentabile con un'espressione dell'algebra*

- Prima **Forma Canonica** di funzione a *n* variabili:

$$f = \Sigma_{j=1..m} \Pi_{i=1..n} x_{i_j}^{*}$$

- $x_{i_j}^{*}$ vale x_i oppure $\overline{x}_i$
- f è espressa come OR delle combinazioni per cui la funzione è vera (somma di *mintermini*)
- in base al teorema, qualsiasi funzione booleana può essere espressa in questa forma

Funzioni Booleane (Esempio)

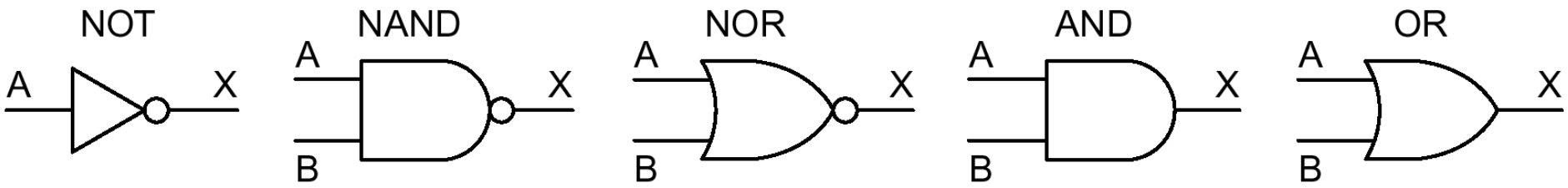
ES

- Tre variabili booleane A, B, C
- Funzione di maggioranza M: è vera solo se almeno due delle tre variabili sono vere

A	B	C	M	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	← $\bar{A}BC$
1	0	0	0	
1	0	1	1	← $A\bar{B}C$
1	1	0	1	← $AB\bar{C}$
1	1	1	1	← ABC

$$M = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$$

Circuiti Logici



A	X
0	1
1	0

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

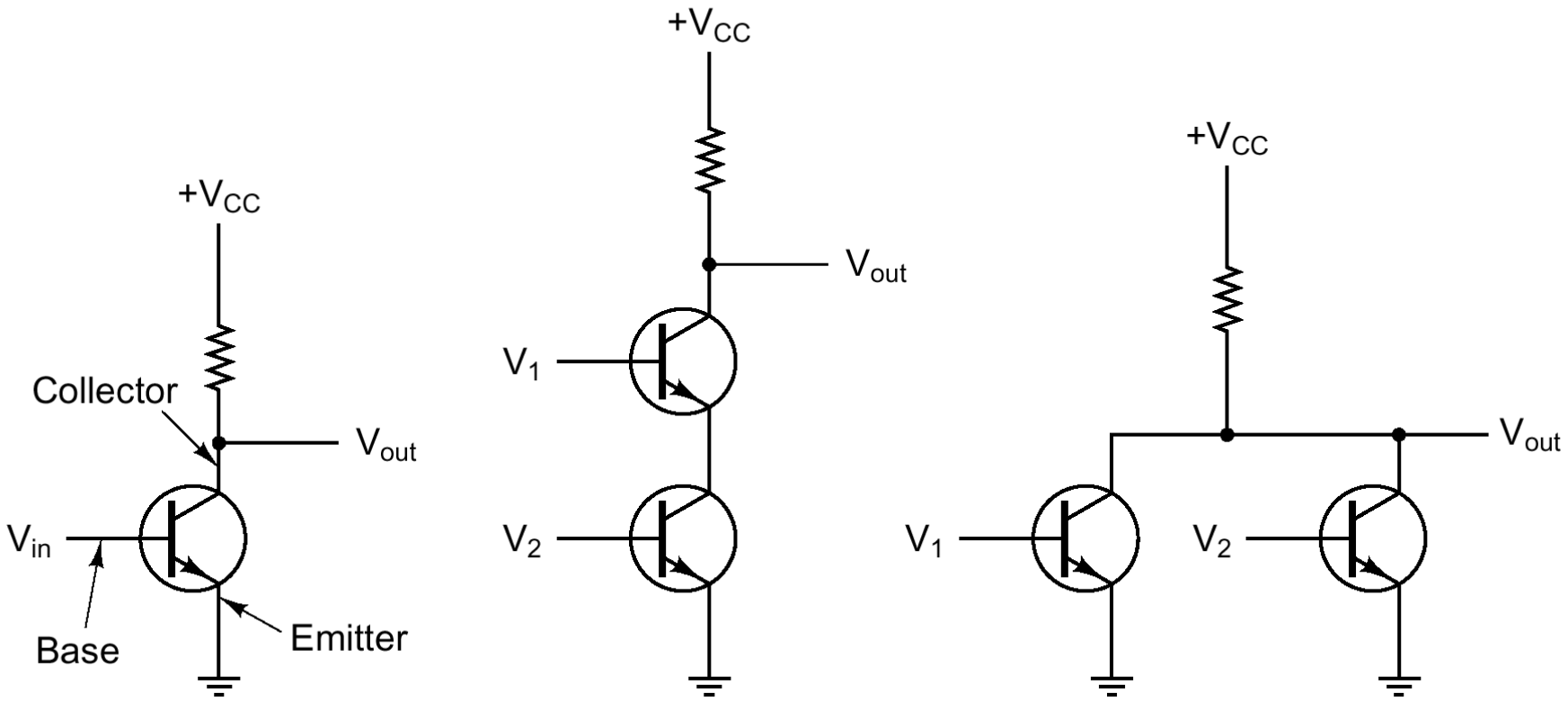
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

- **Porte Logiche**: circuiti elementari che realizzano gli operatori dell'algebra

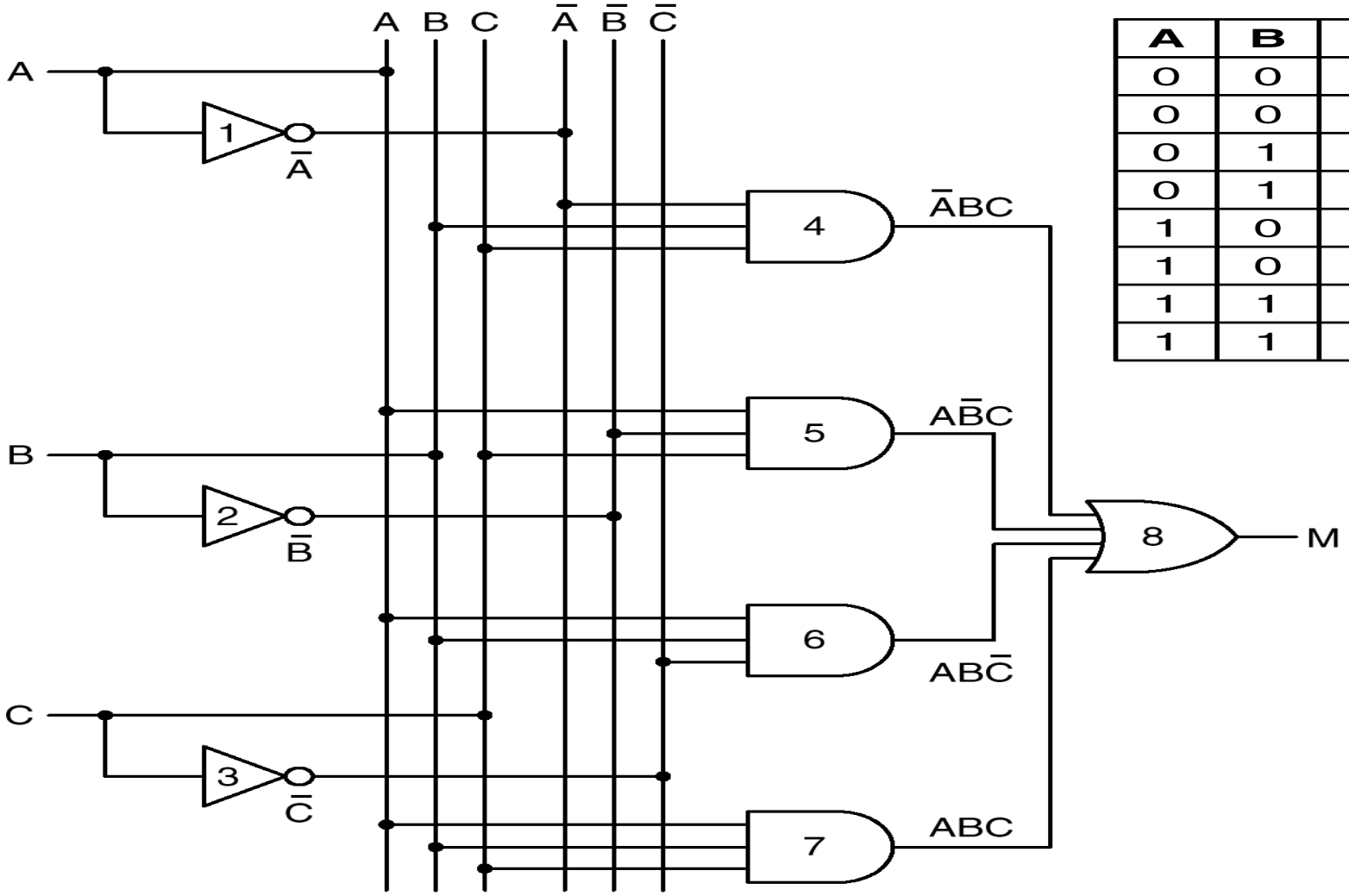
Qualsiasi funzione booleana può essere calcolata con un circuito realizzato con sole porte AND, OR *e* NOT

Realizzazione di porte logiche con circuiti elettronici



NOT NAND NOR

Implementazione di Funzioni Booleane con Circuiti Logici



A	B	C	M
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

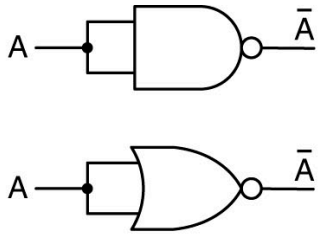
$$M = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$$

Proprietà dell'Algebra Booleana

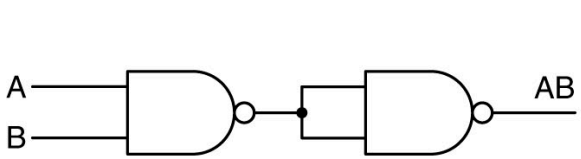
Name	AND form	OR form
Identity law	$1A = A$	$0 + A = A$
Null law	$0A = 0$	$1 + A = 1$
Idempotent law	$AA = A$	$A + A = A$
Inverse law	$A\overline{A} = 0$	$A + \overline{A} = 1$
Commutative law	$AB = BA$	$A + B = B + A$
Associative law	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Distributive law	$A + BC = (A + B)(A + C)$	$A(B + C) = AB + AC$
Absorption law	$A(A + B) = A$	$A + AB = A$
De Morgan's law	$\overline{AB} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A}\,\overline{B}$

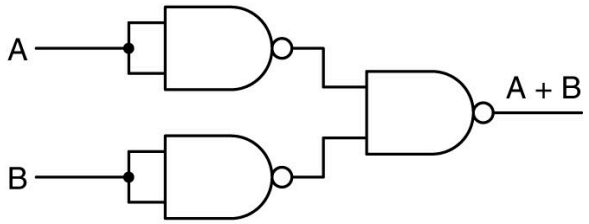
Completezza delle porte NAND e NOR

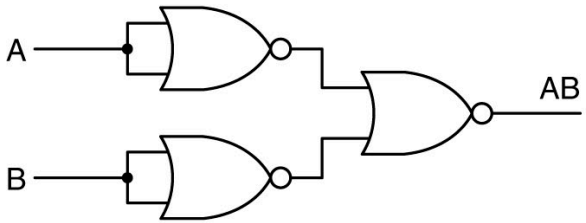
È possibile simulare AND, OR *e* NOT, *e quindi realizzare qualsiasi circuito, usando soli* NAND *oppure soli* NOR

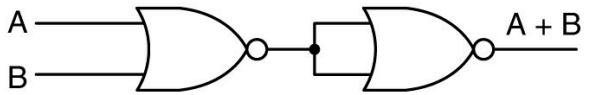


(a)



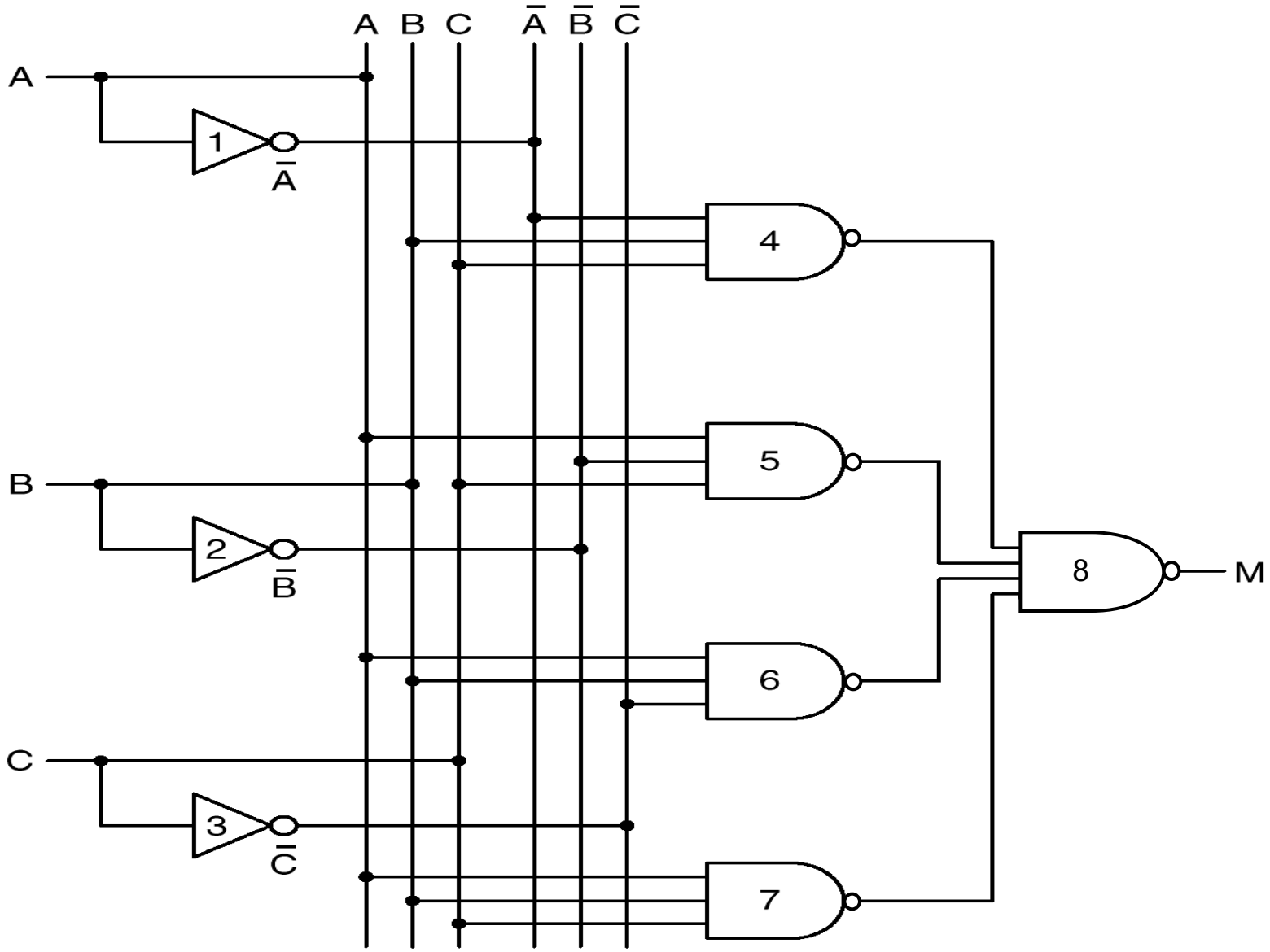




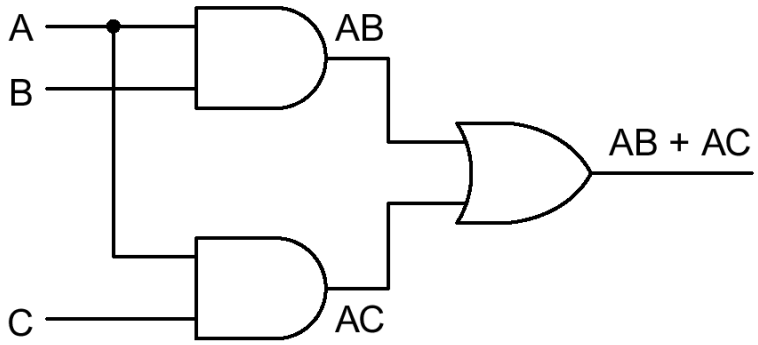


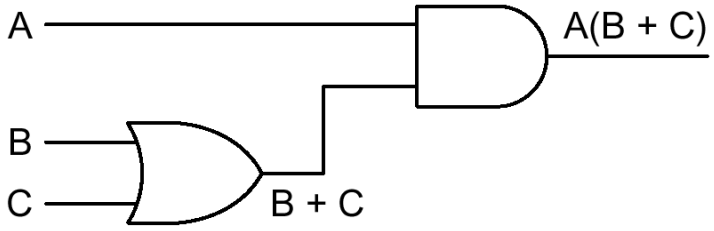
(b) (c)

Realizzazione della maggioranza con solo NAND



Ottimizzazione di circuiti logici (esempio)



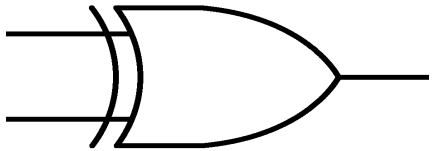


A	B	C	AB	AC	AB + AC
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

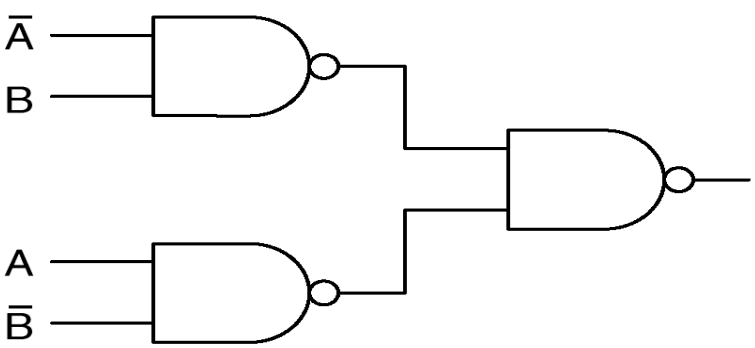
A	B	C	A	B + C	A(B + C)
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

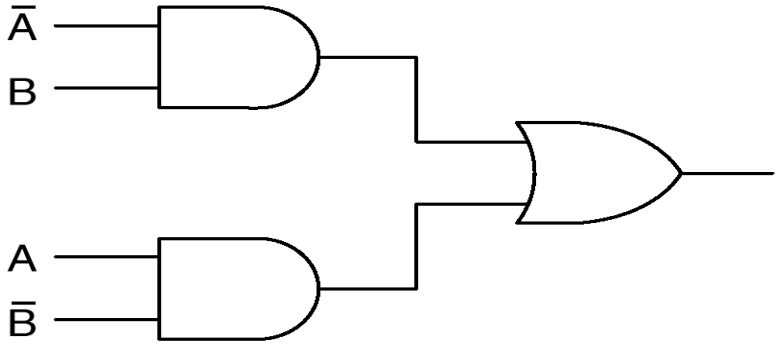
Porte XOR

A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

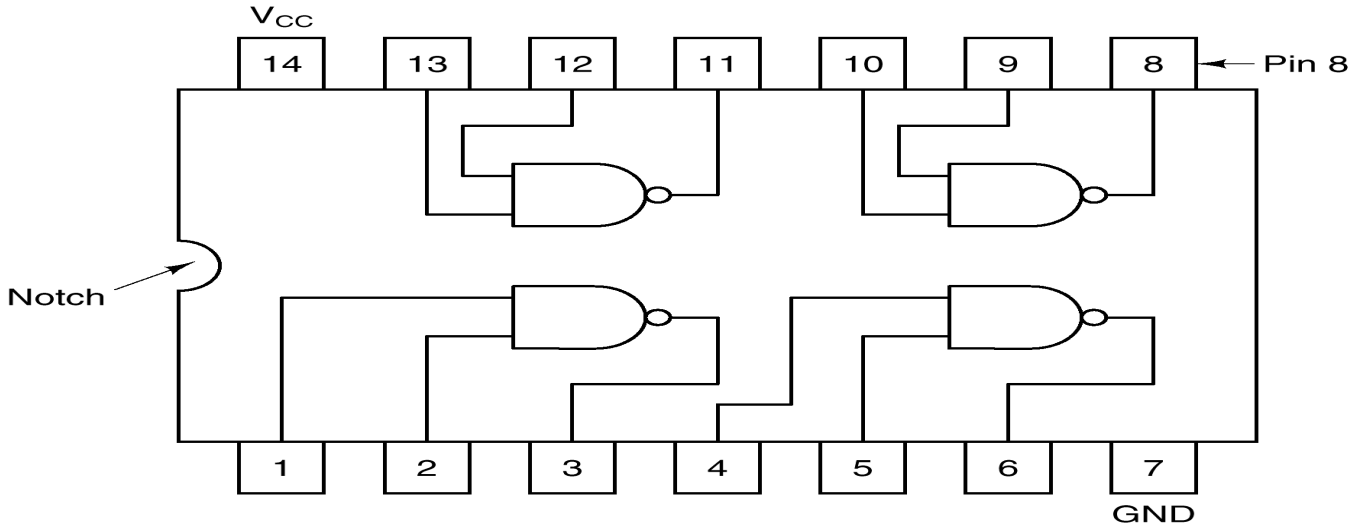


- Calcola la funzione *OR esclusivo*: dà uscita 1 (vero) quando uno solo degli ingressi (ma non entrambi) vale 1
- Facilmente realizzabile con porte AND, OR e NAND



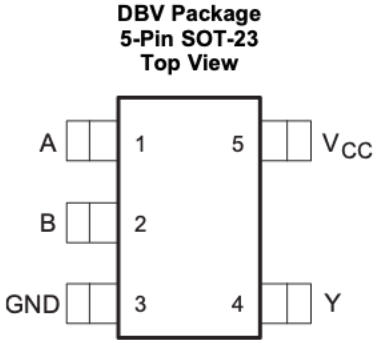


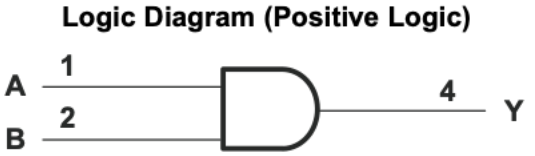
Circuiti Integrati

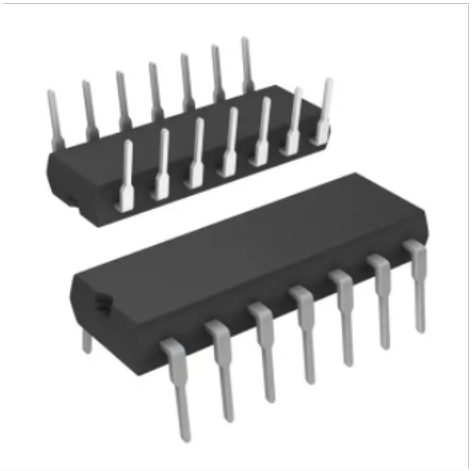


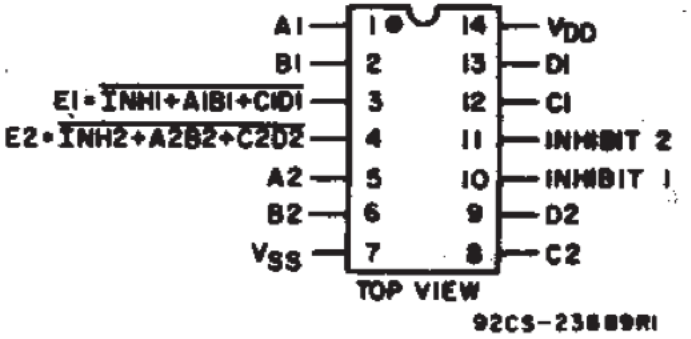
- Molte porte realizzate sulla stessa piastrina di silicio (*chip*)
- Contenitori (schede) da 14 a 68 piedini
- Vari livelli di integrazione:
  - **SSI** (*Small Scale*) 1-10 porte
  - **MSI** (*Medium Scale*) 10-100 porte
  - **LSI** (*Large Scale*) 10^2-10^5 porte
  - **VLSI** (*Very Large Sc.*) > 10^5 porte
- Tempi di commutazione: 0,1-10 nsec

Esempi pratici

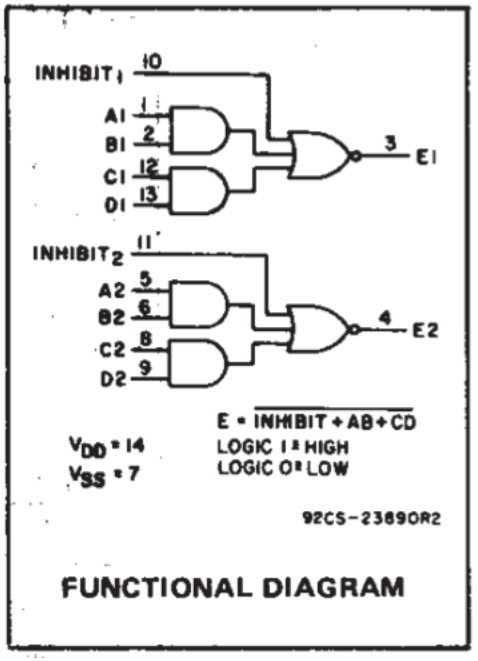






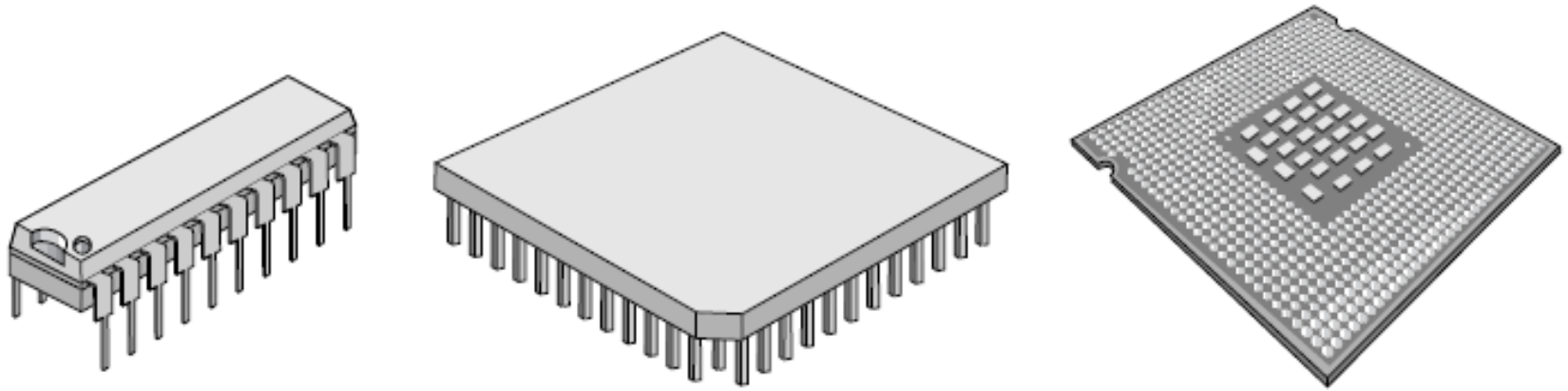


Terminal Assignment



Circuiti integrati moderni

a) Dual Inline Packages (DIPs)
b) Pin Grid Arrays (PGAs)
c) Land Grid Arrays (LGAs)

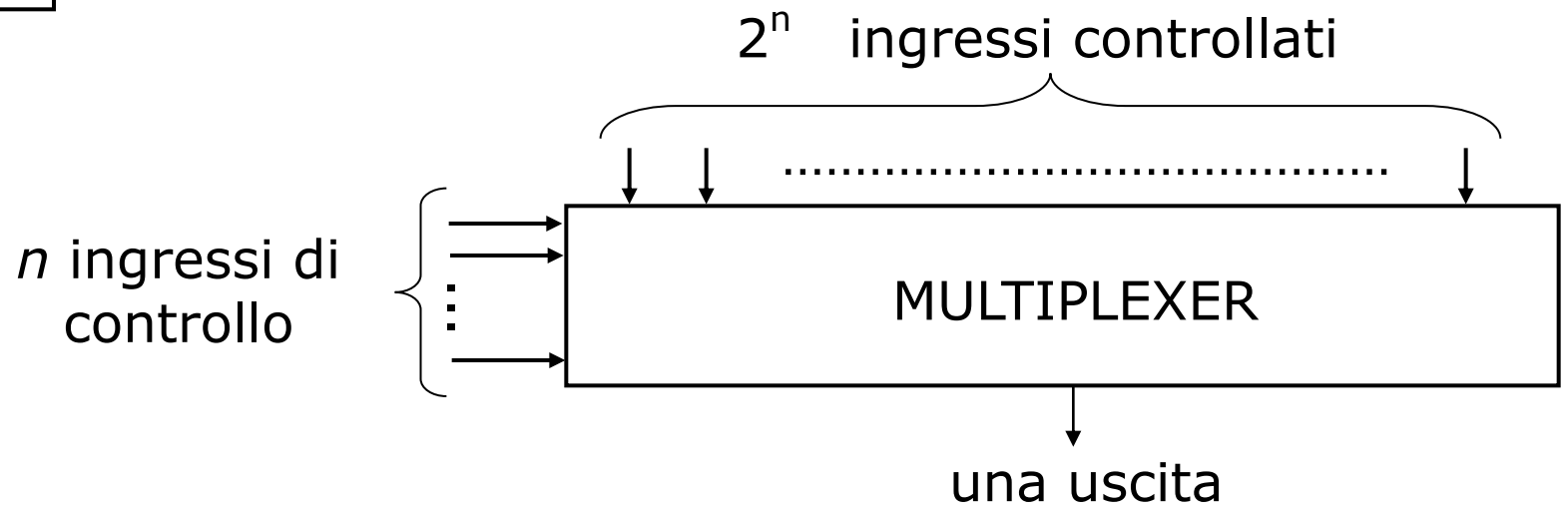


(a) (b) (c)

Circuiti Combinatori

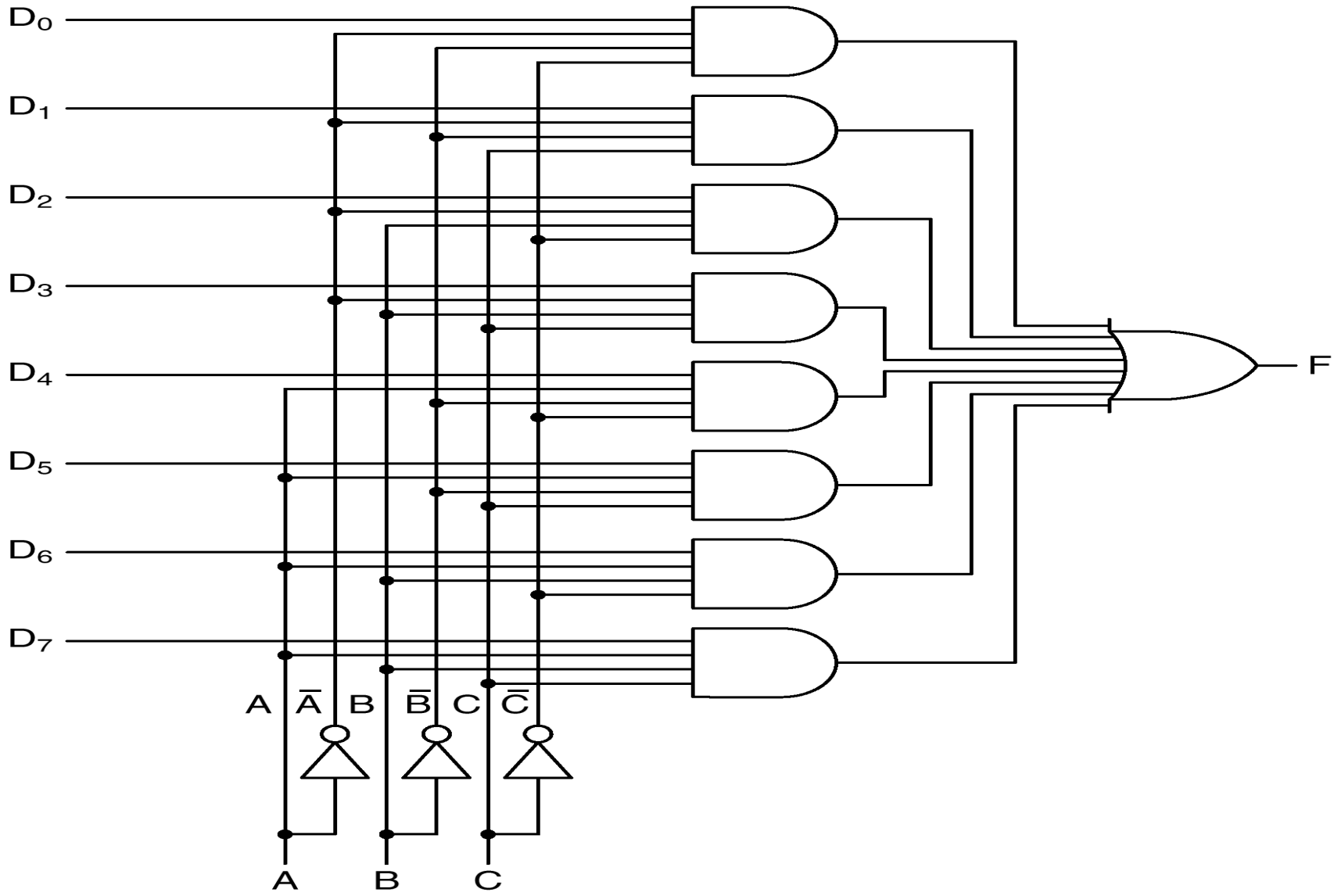
Circuiti in cui l'uscita dipende solo dagli ingressi, e non dallo stato cioè dalla storia passata

ES



- Gli ingressi di controllo selezionano quale degli ingressi controllati viene mandato in uscita

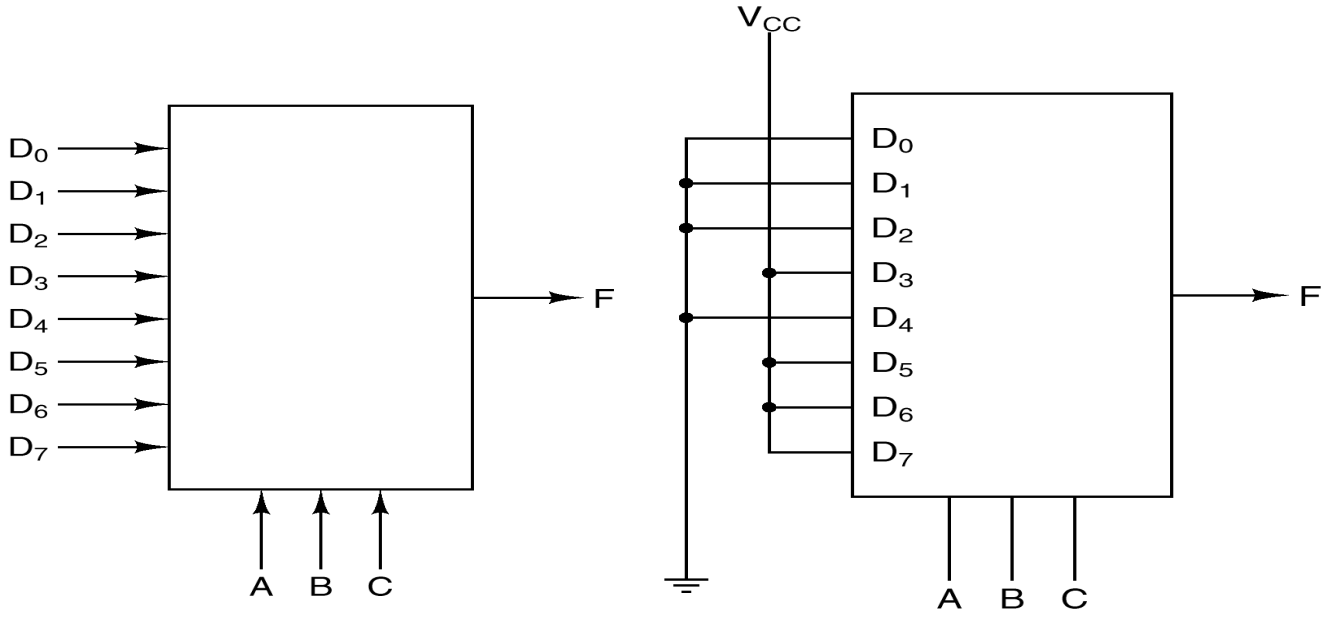
Multiplexer (circuito)



Realizzazione di funzioni booleane tramite multiplexer

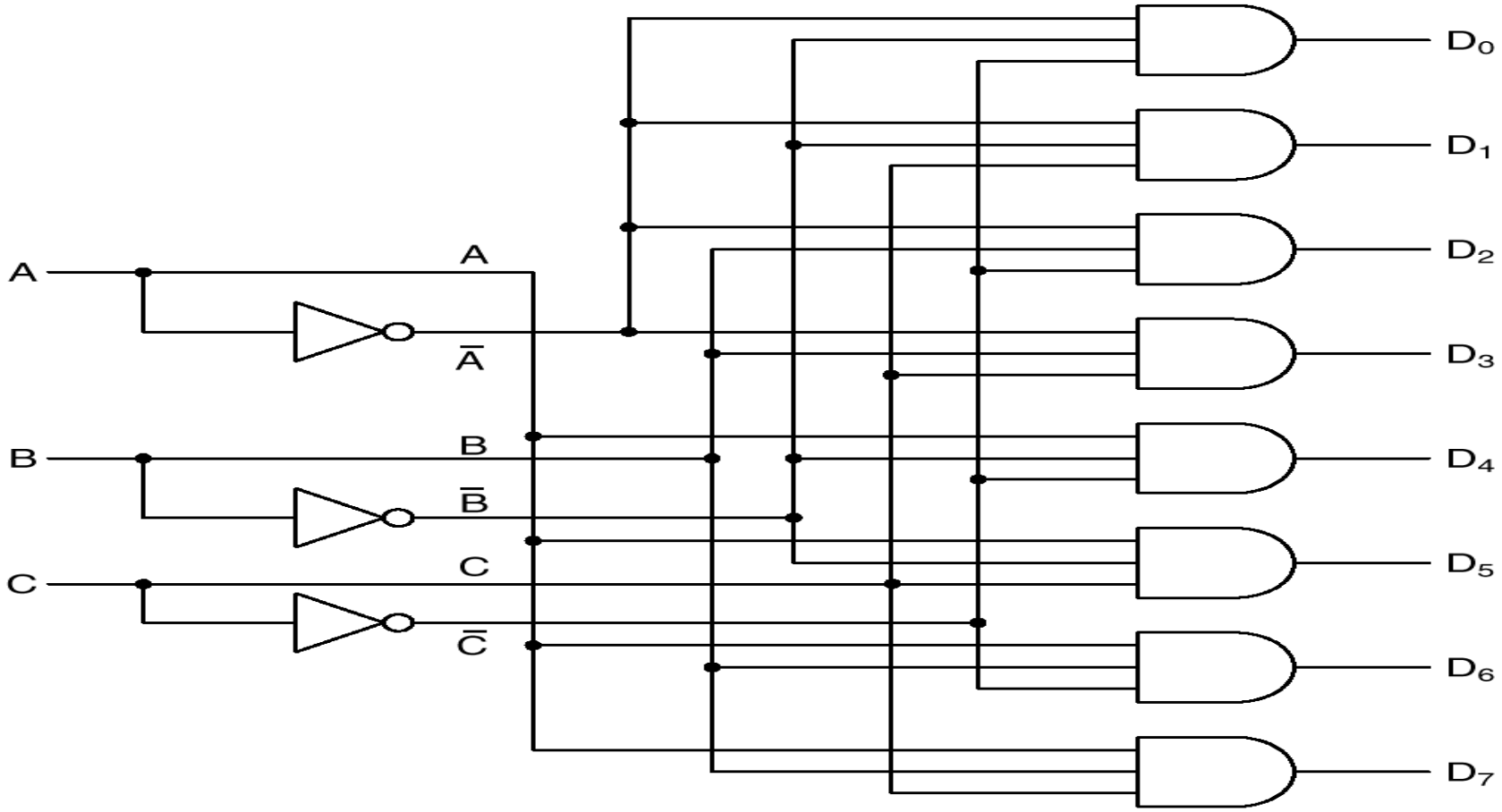
- Con un multiplexer ad n bit si può calcolare qualsiasi funzione di *n* variabili
- Gli ingressi controllati corrispondono ai mintermini
- Si cablano a 0 o 1, a seconda che il mintermine compaia o meno nella forma canonica

ES (*Funzione di maggioranza*)



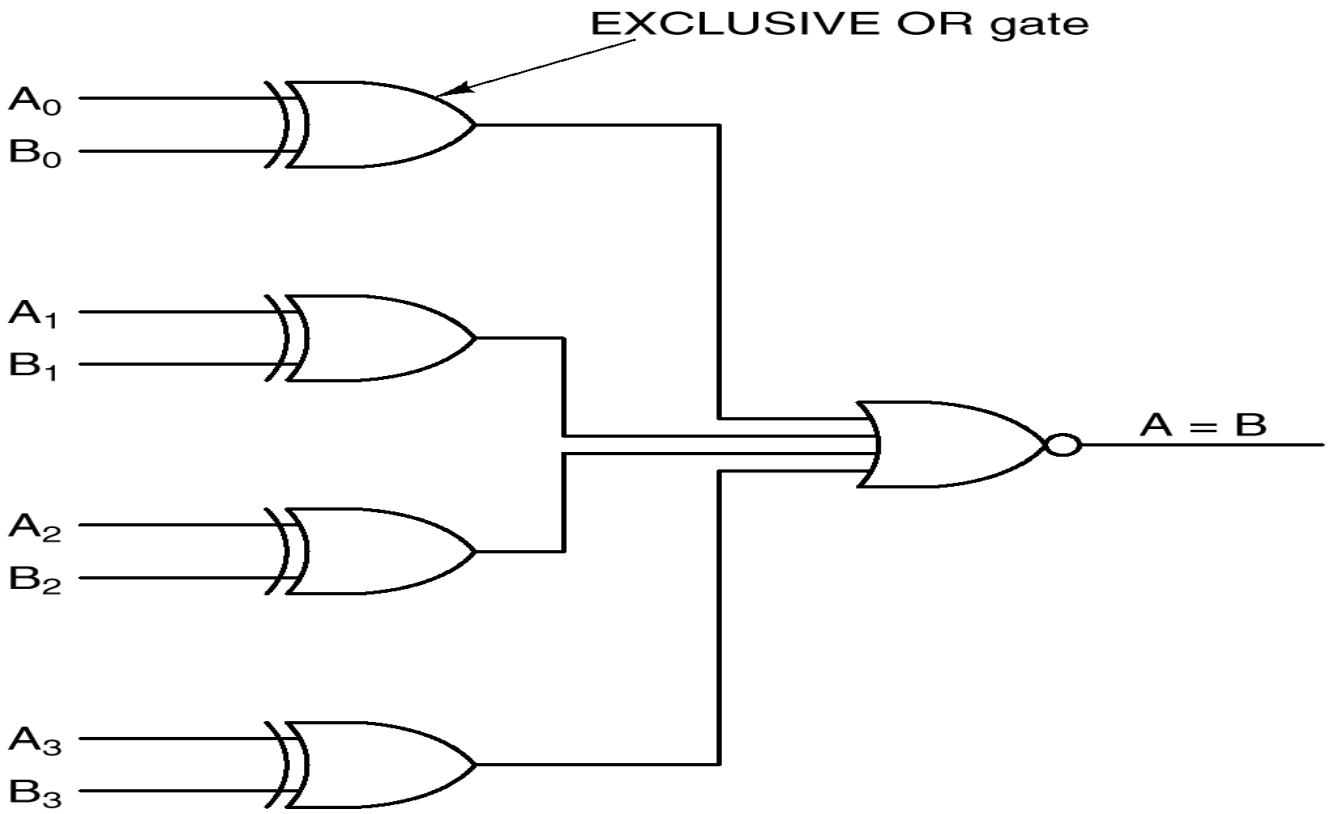
A	B	C	M
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Decodificatore



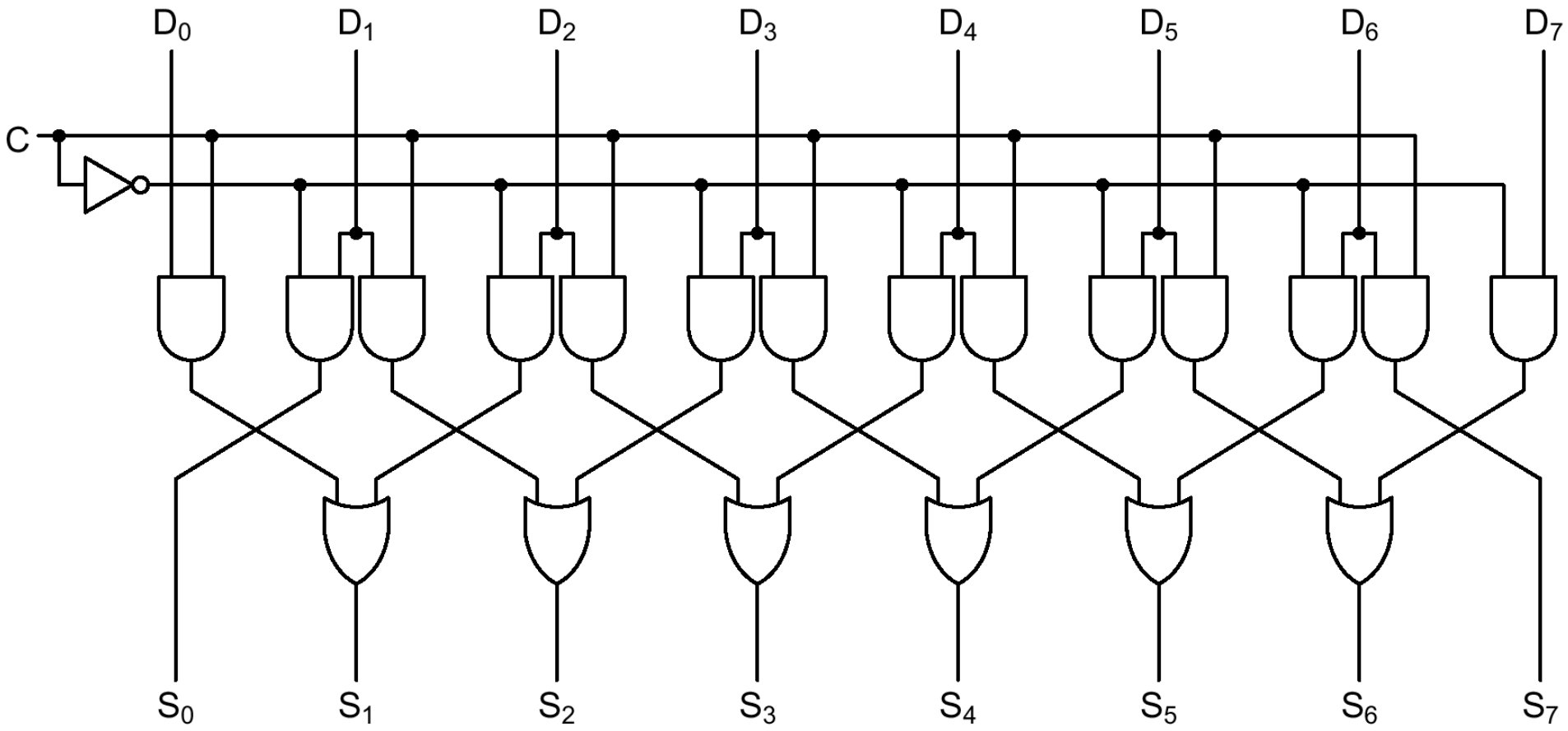
- Circuito a n ingressi e 2^n uscite
- Una ed una sola delle 2^n uscite assume valore vero in corrispondenza della configurazione di n bit in ingresso

Comparatore



- Compara i bit omologhi di due stringhe
- L'uscita vale 1 se e solo se $A_i=B_i \ \forall$ i
- Se $A_i=B_i$ allora $A_i \text{ XOR } B_i = 0$
- Il NOR da uscita 1 solo quando tutti i suoi ingressi valgono 0

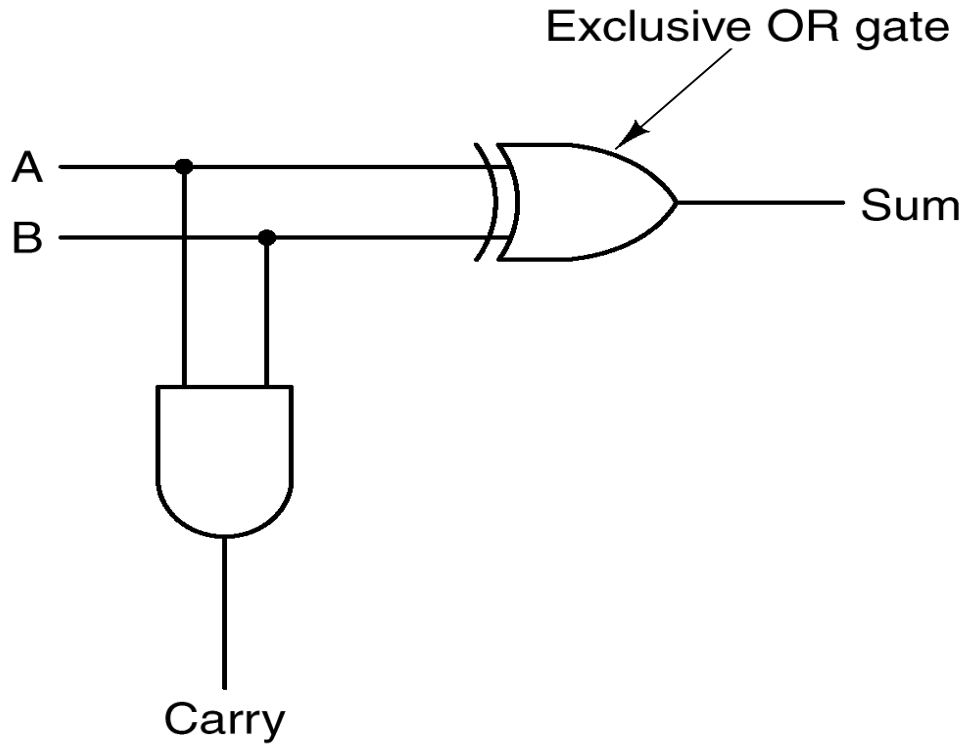
Shifter



- Il segnale C determina il verso dello shift (sinistra/destra)

Semiaddizionatore (Half Adder)

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

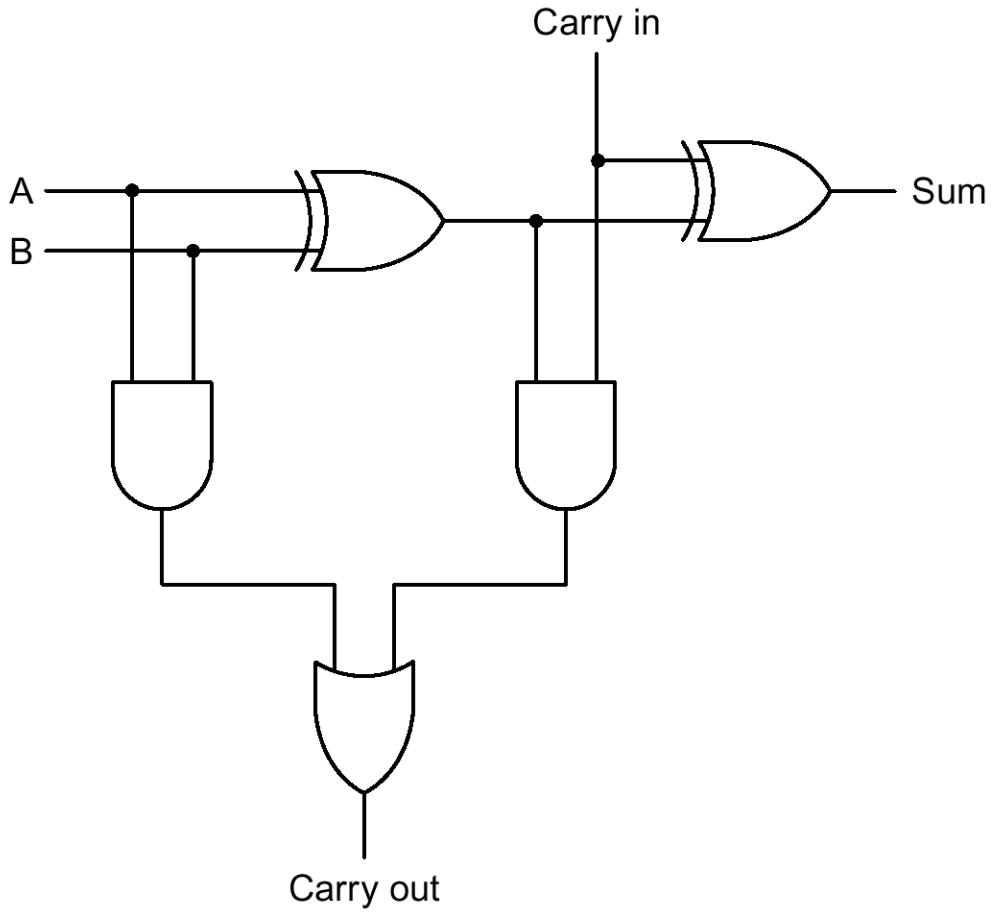


- Circuito a 2 ingressi e 2 uscite: somma e riporto (*carry*)
- Non può essere usato per la somma di numerali a più bit, dove occorre sommare anche il riporto della cifra precedente

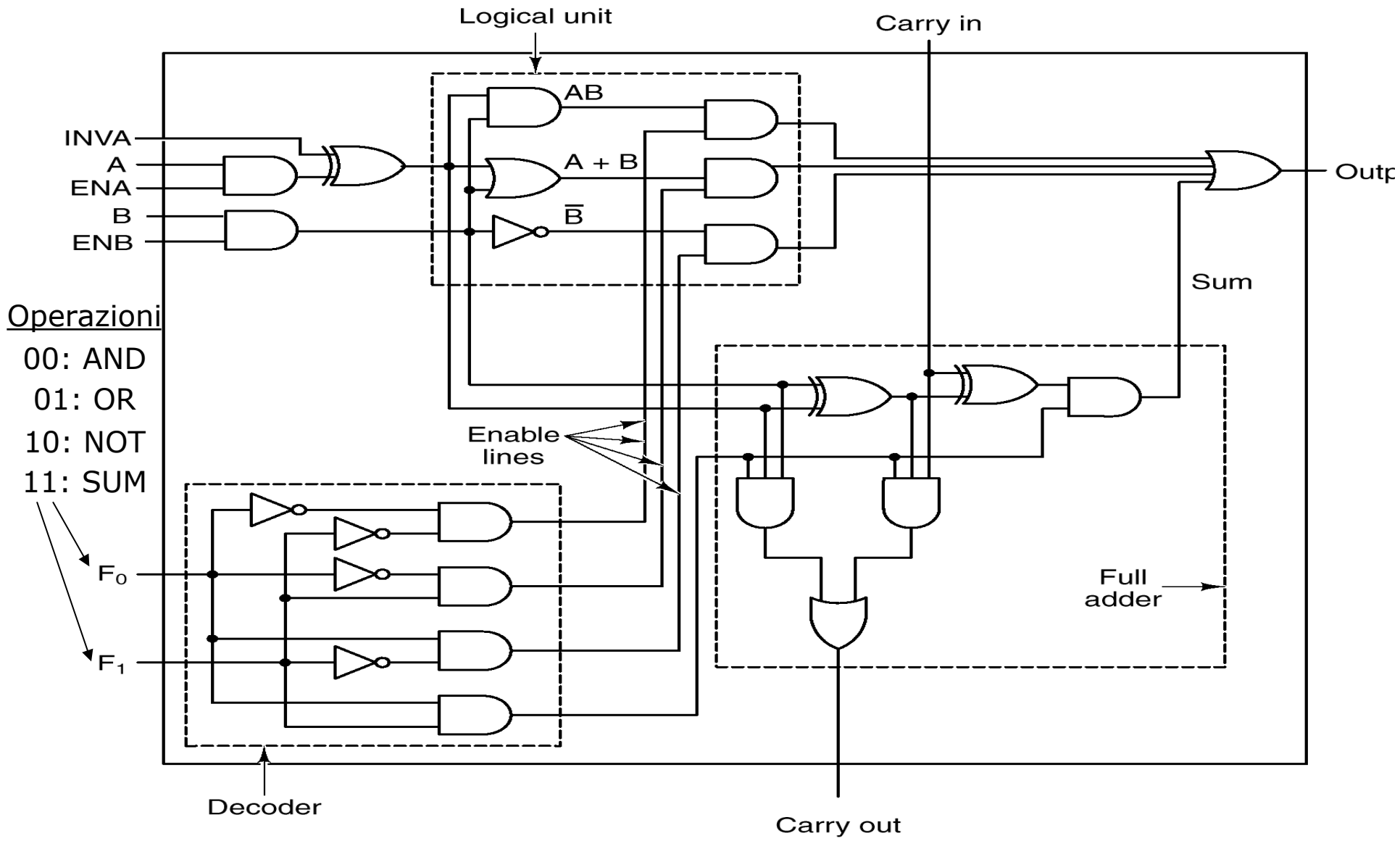
Addizionatore Completo (Full Adder)

- Circuito a 3 ingressi e 2 uscite
- Riceve il riporto dalla cifra precedente

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

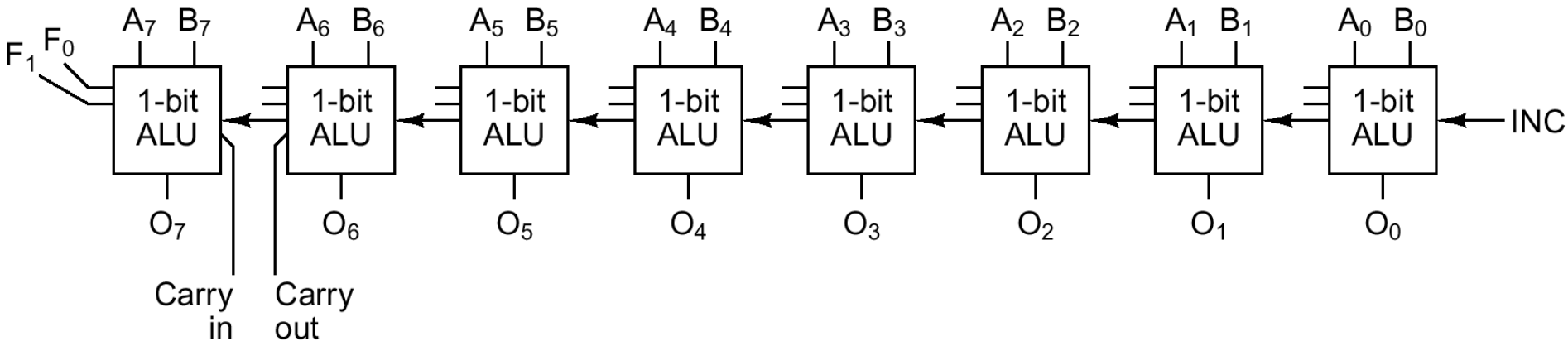


ALU a 1 bit (bit slice)

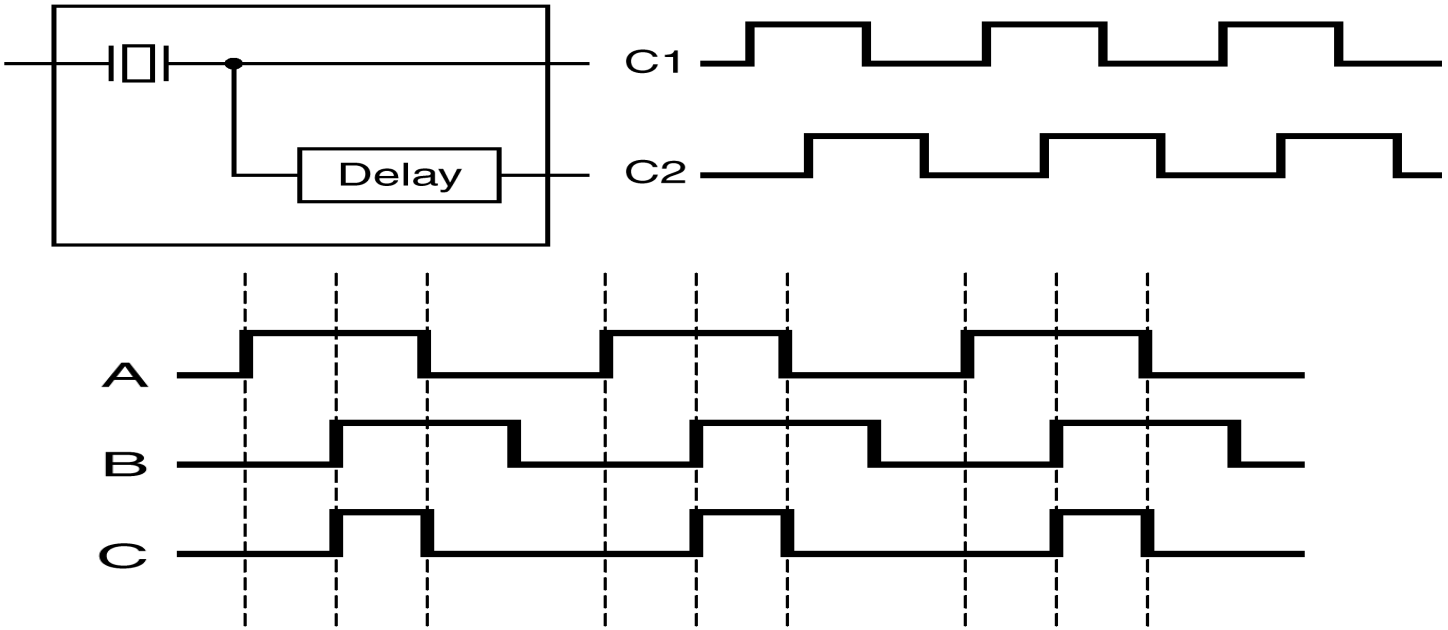


ALU ad n bit

- Realizzata connettendo n ALU ad 1 bit (bit slices)
- INC incrementa la somma di 1 (A+1, A+B+1)
- Problema: propagazione dei riporti
- Ciascuno stadio deve attendere il riporto dal precedente
- Tempo di addizione lineare con n



Clock



- Tutti i cambiamenti di stato vengono sincronizzati da un segnale (*clock*)
- Da un clock primario ne vengono ricavati per sfasatura, sottrazione ecc.
- Le transizioni di stato del circuito possono avvenire:
  1. In corrispondenza dei *livelli*
  2. In corrispondenza dei *fronti*

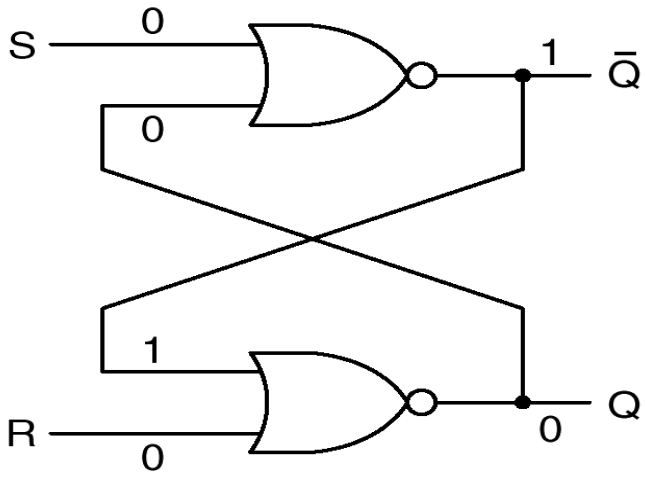
Circuiti Sequenziali

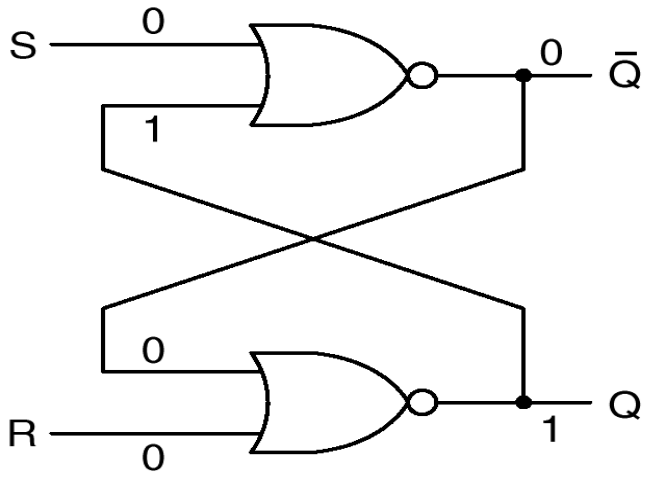
$$o_i = f_i(i_1,\ldots,i_n,s_1,\ldots,s_r) \qquad i=1,\ldots,m$$

$$s'_i = g_i(i_1,\ldots,i_n,s_1,\ldots,s_r) \qquad j=1,\ldots,r$$

- Le uscite del circuito dipendono dagli ingressi e dalla *storia passata*
- La storia passata è riassunta nello stato che è codificato nelle variabili di stato booleane $s_1,\ldots,s_r$
- Le variabili di stato sono memorizzate in elementi di memoria binari
- Circuiti combinatori calcolano le uscite e il nuovo valore dello stato

Latch

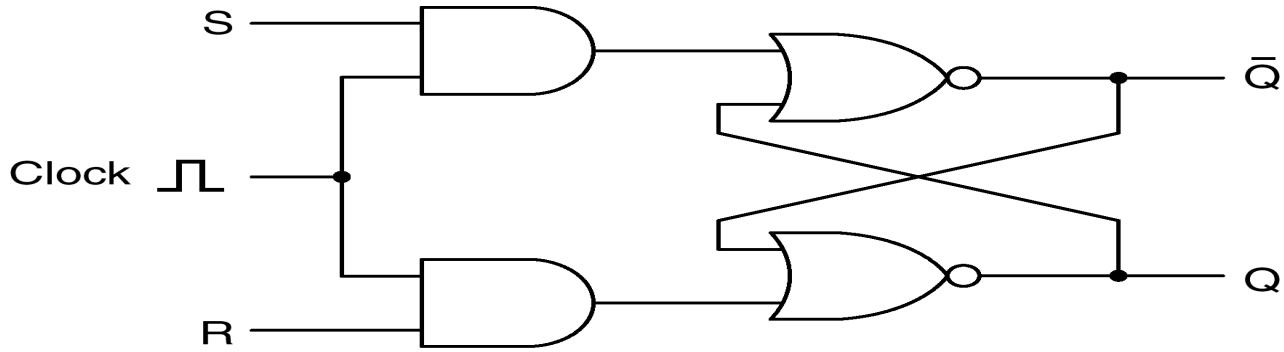




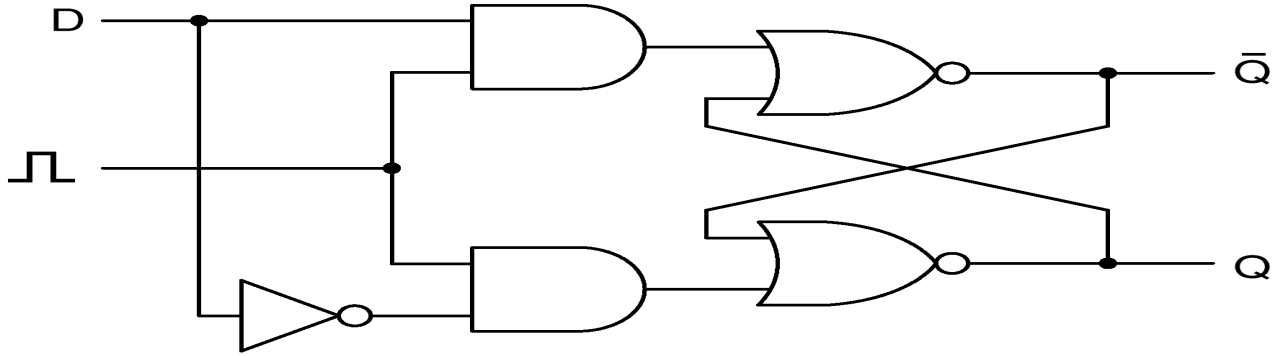
A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

- Dispositivo di memoria elementare
- Due stati stabili Q=0 e Q=1
  - S (SET): forza Q a 1
  - R (RESET): forza Q a 0
- Con S=R=0 il circuito *mantiene lo stato*
- Il circuito commuta sui livelli cioè quando S o R valgono 1
- S ed R non devono mai andare insieme ad 1

Latch con Clock, Latch D



- R ed S vengono trasferiti sugli ingressi del latch solo quando il clock è ad 1
- Quando il clock è a 0 vengono ignorati

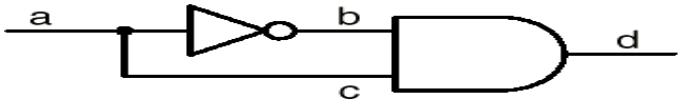


- Il **latch** D (Delay) quando il clock va ad 1 registra nello stato Q il valore dell'ingresso D

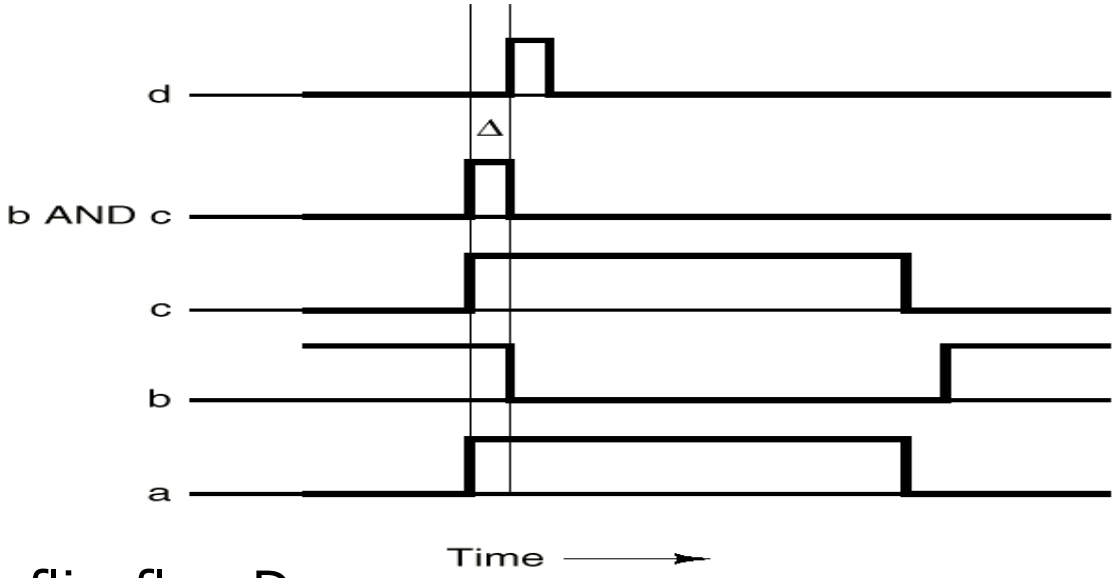
Flip-flop

Il **flip-flop** e' una variante del latch che commuta sui fronti del clock

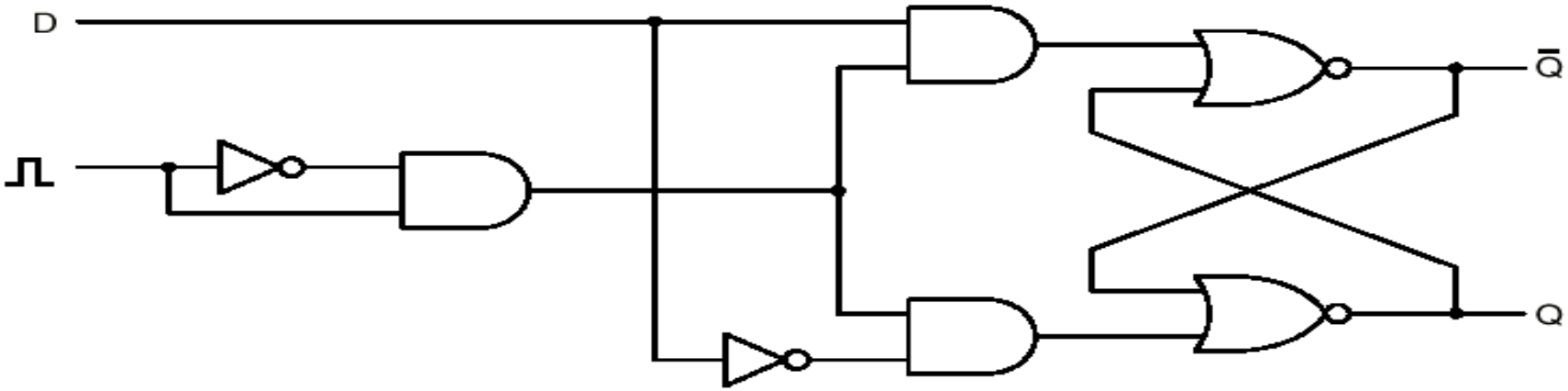
Generatore di impulsi



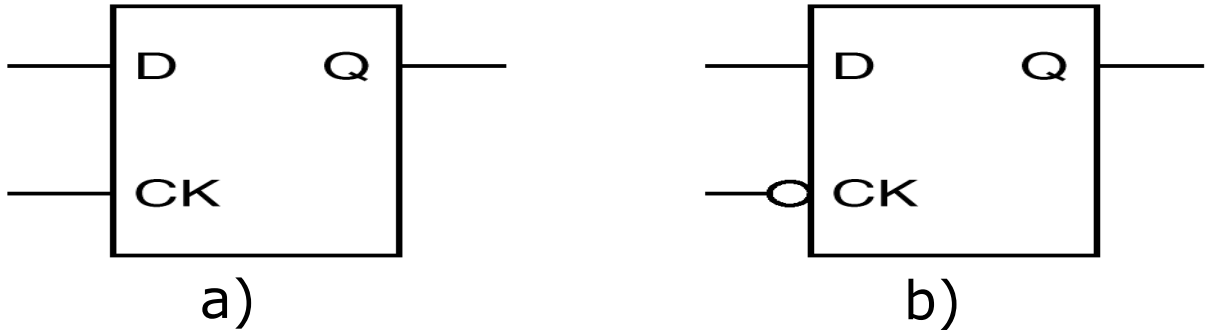
(a)



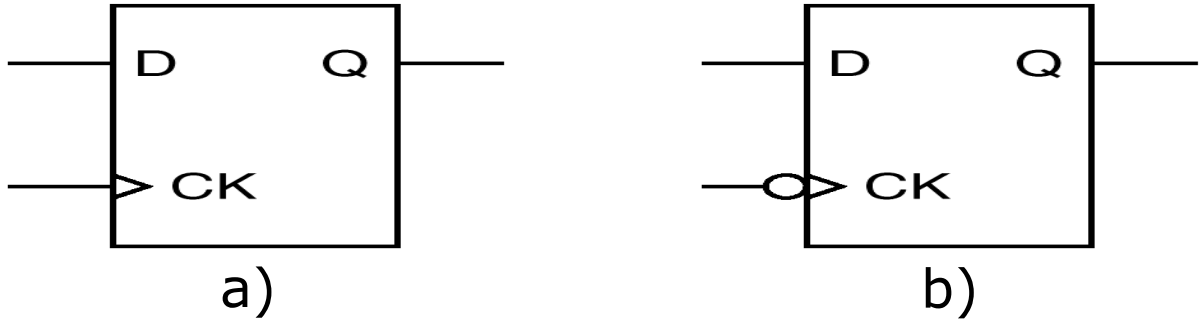
Possibile realizzazione di un flip-flop D:



Latch e Flip-Flop



I **Latch** commutano sui livelli del clock (a) alto, b) basso)

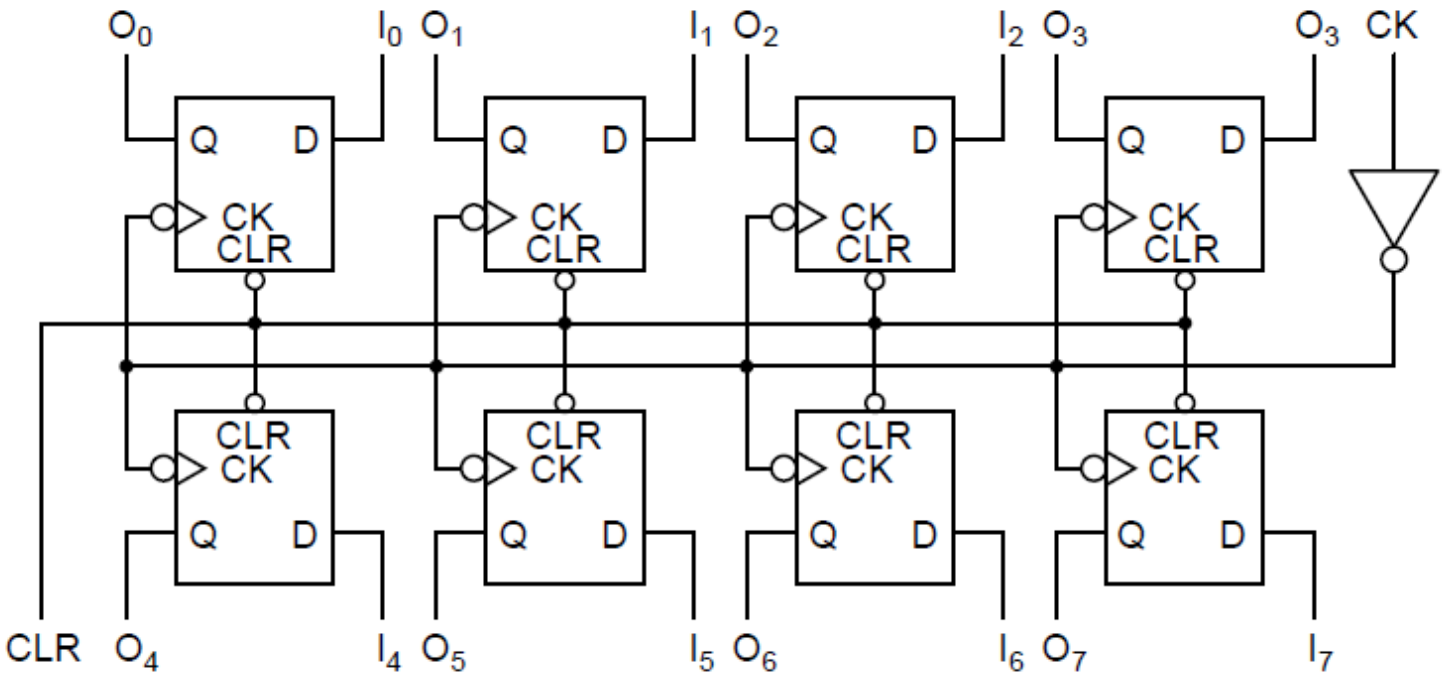


I **Flip-Flop** commutano sui fronti del clock:

a) Commuta sul *fronte di salita*

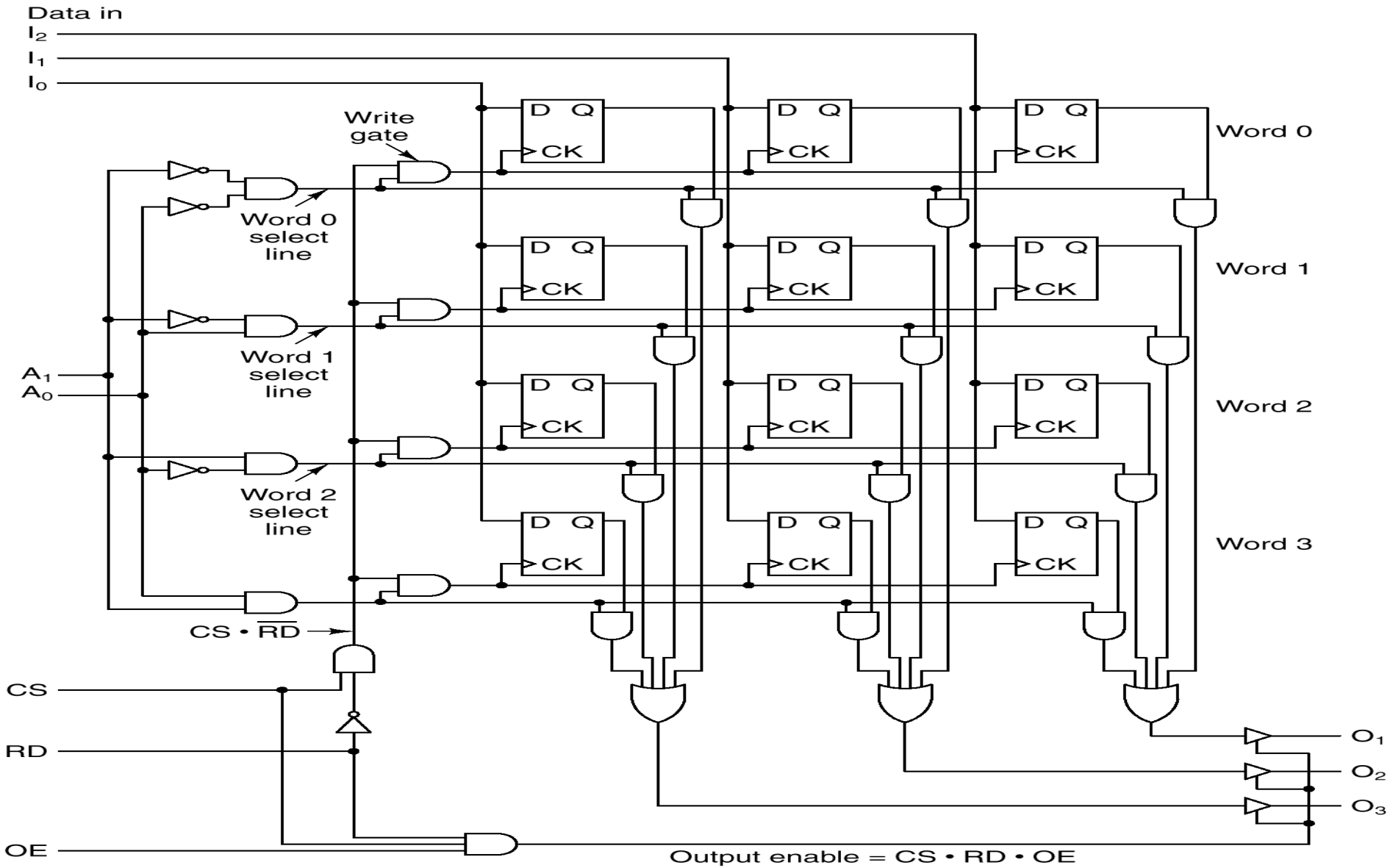
b) Commuta sul *fronte di discesa*

Registri

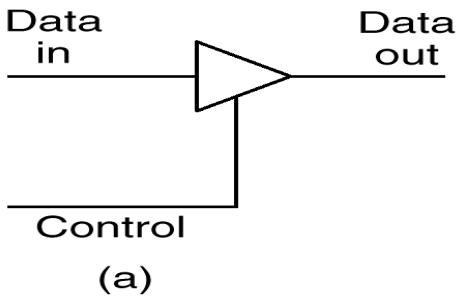


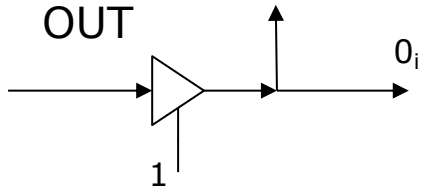
- I Flip-Flop sono gli elementi base di memorizzazione nel computer
- Molti Flip-Flop possono essere messi su un unico chip
- Occorrono in genere da 6 a 10 transistor per ogni Flip-Flop

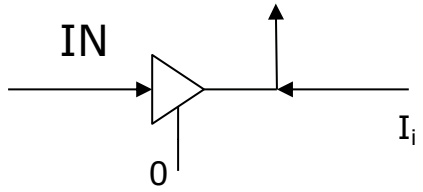
Organizzazione della Memoria



Dispositivi a 3 stati

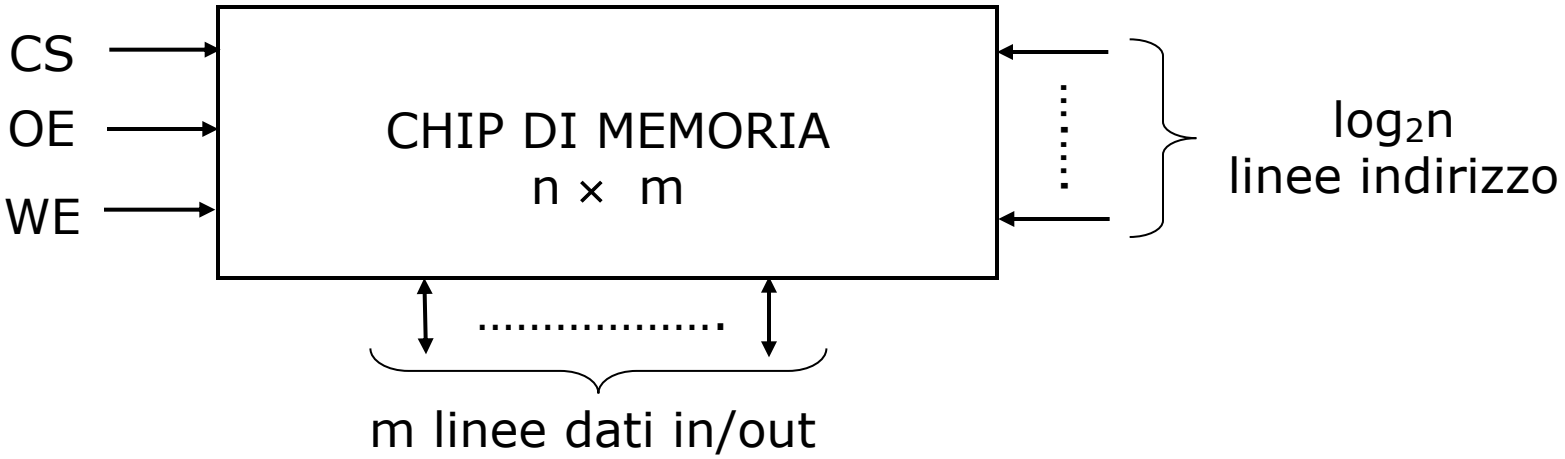






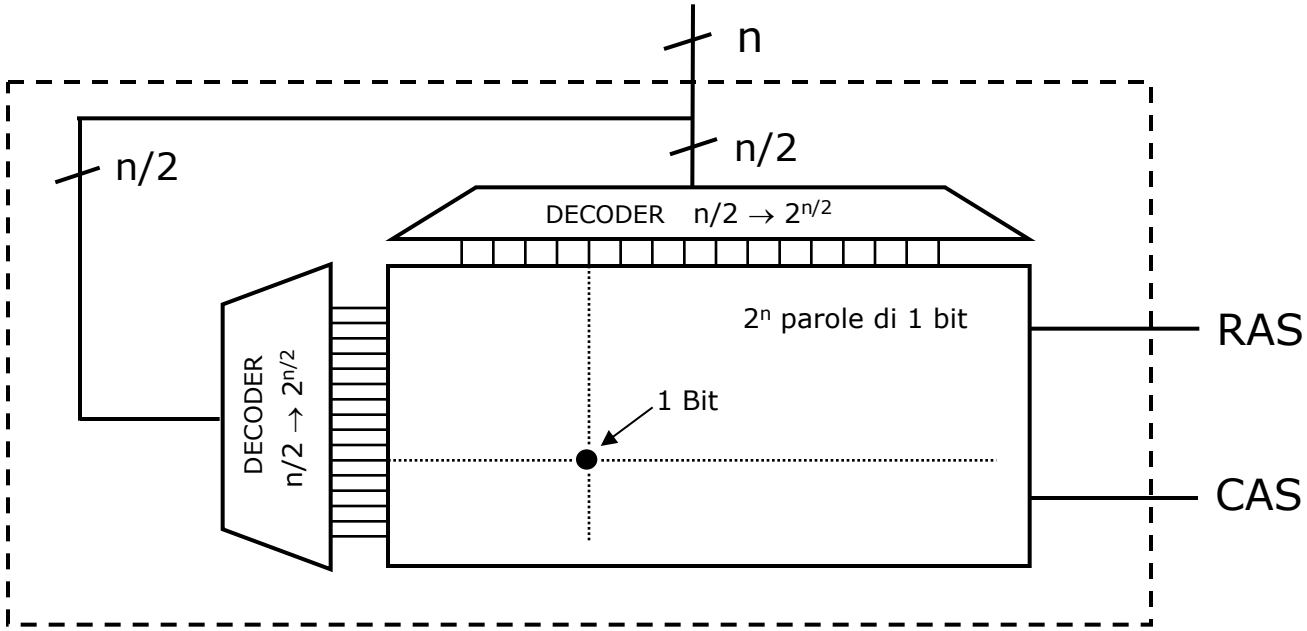
- In base ad un segnale di controllo C si comporta:
  - C=1: come circuito chiuso
  - C=0: come circuito aperto
- Tempo di commutazione: pochi *nsec*
- Consente di usare gli stessi piedini sia per la lettura che per scrittura
- Usato anche per la connessione ai bus e a qualsiasi linea bidirezionale

Chip di Memoria



- Chip da n × m bit complessivi (*n* parole da *m* bit)
- m linee dati bidirezionali
- $\log_2 n$ linee di indirizzo
- Segnali di controllo:
  - CS (Chip Select)
  - OE (Output Enable)
  - WE (Write Enable)
- Problema: numero limitato di piedini del contenitore

Matrice di selezione



- Si risparmia nella complessità della logica di decodifica
- Un decoder $n \rightarrow 2^n$ richiede 2^n porte AND
- RAS (Row Address Strobe), CAS (Column Address Strobe)

ES

- 4M parole da 1 bit $\rightarrow$ 22 linee di indirizzo
- 1 decoder a 22 $\rightarrow$ 4M porte AND
- 2 decoder a 11 $\rightarrow 2 \cdot 2^{11} =$ 4K porte AND

Segnali asseriti e negati

In alcuni casi (a seconda delle scelte di progetto) un segnale provoca l'azione corrispondente quando la sua tensione è alta, in altri quando è bassa

Per evitare confusione si parla di:

- Segnale asserito: quando assume il valore che provoca l'azione
- Segnale negato: altrimenti

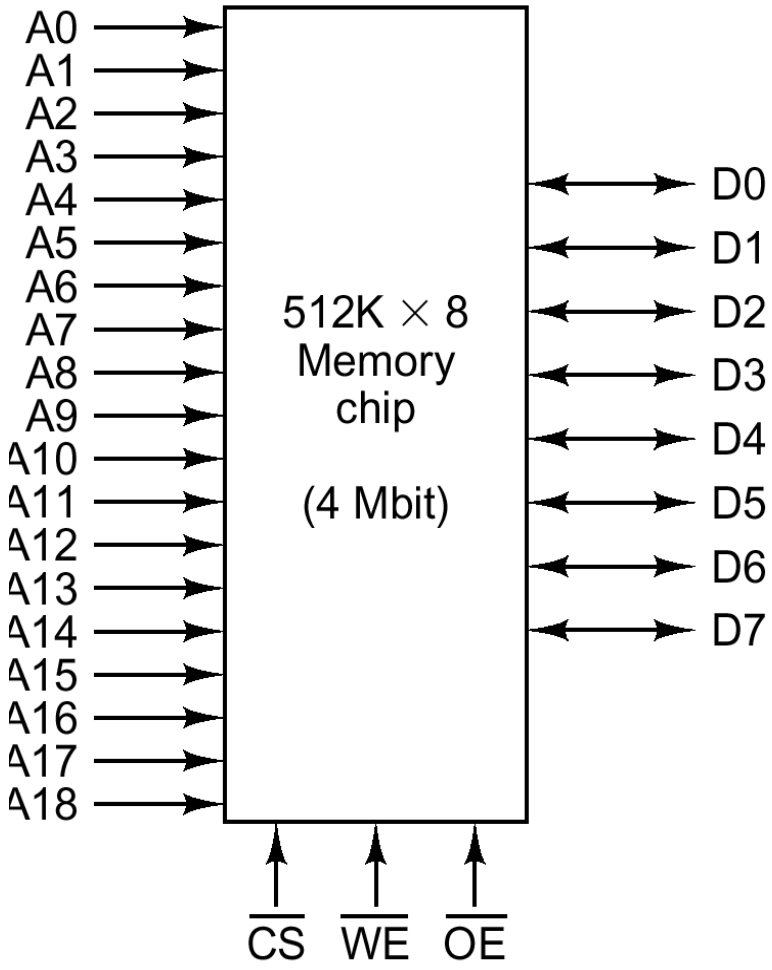
Si adotta la seguente notazione:

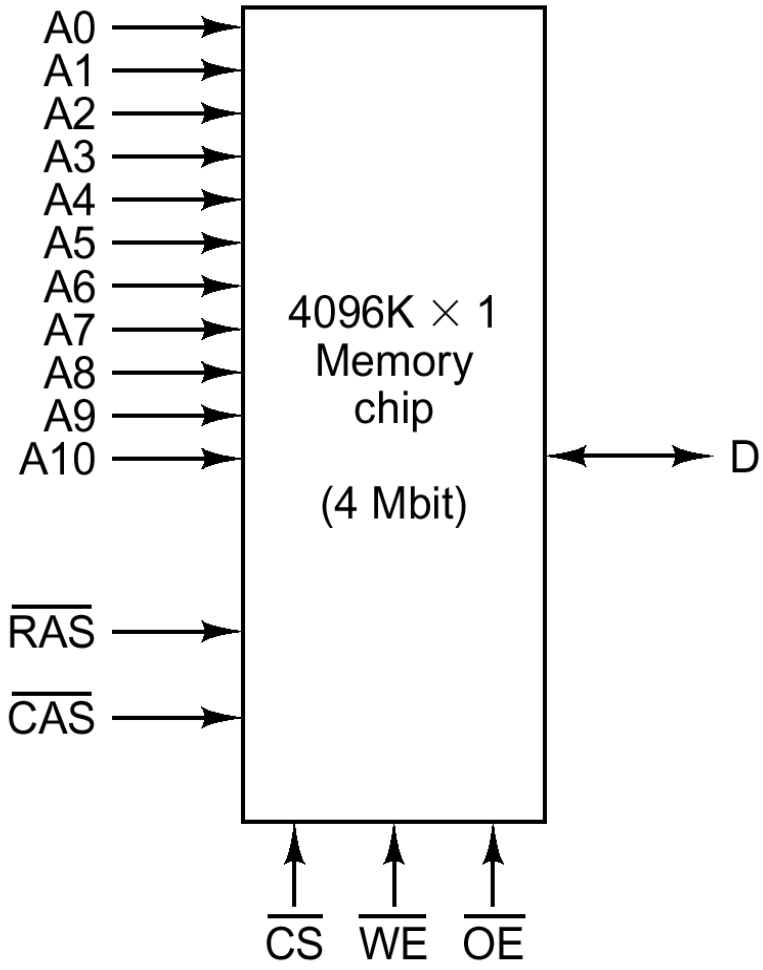
- S: segnale che è asserito alto
- $\overline{S}$: segnale che è asserito basso

Ulteriore notazione (usata da Intel):

- S: segnale che è asserito alto
- S#: segnale che è asserito basso
  - (adatta al set di caratteri UNICODE)

Chip di Memoria (Esempi)





Tassonomia delle RAM e ROM

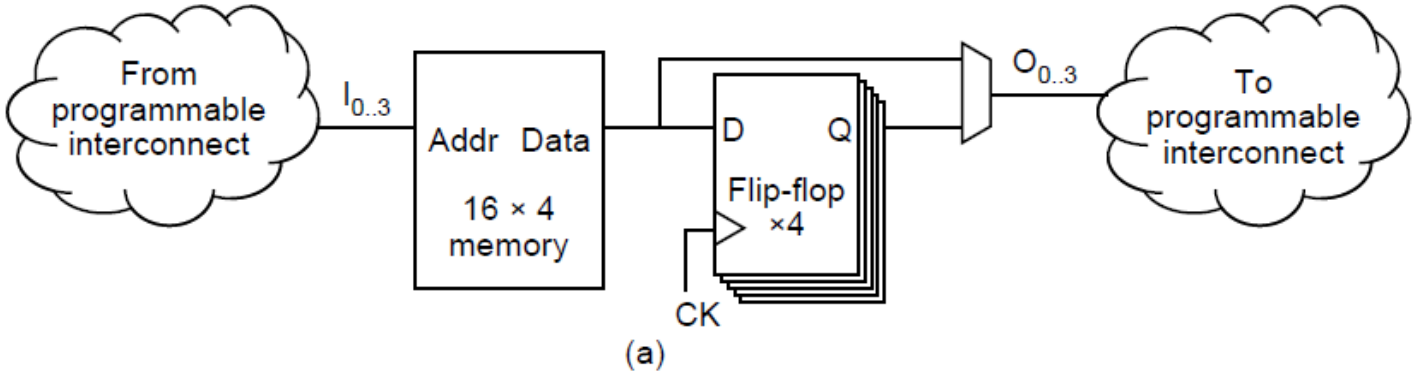
- RAM (Random Access Memory)
- ROM (Read Only Memory)
- SRAM (Static RAM): a Flip-Flop, molto veloce (~5nsec)
- DRAM (Dynamic RAM): basata su capacità parassite; richiede refresh, alta densità, basso costo (~70 nsec)
  - FPM: selezione a matrice
  - EDO: (Extended Data Output) lettura in pipeline, più banda
- SDRAM (Synchronous DRAM)
  - Sincrona, prestazioni migliori
- DDR (Double Data Rate)
  - Lettura/scrittura in pipeline
  - DDR2-3-4: 100Mhz/1.6Ghz, fino a 25.6 GBs
- PROM (Programmable ROM)
- EPROM (Erasable PROM) raggi UV
- EEPROM: cancellabile elettricamente
- Flash Memory: tipo di EEPROM, ciclo 50 nsec, max 100.000 riscritture

Tipi di RAM e di ROM e loro impieghi

Type	Category	Erasure	Byte alterable	Volatile	Typical use
SRAM	Read/write	Electrical	Yes	Yes	Level 2 cache
DRAM	Read/write	Electrical	Yes	Yes	Main memory (old)
SDRAM	Read/write	Electrical	Yes	Yes	Main memory (new)
ROM	Read-only	Not possible	No	No	Large-volume appliances
PROM	Read-only	Not possible	No	No	Small-volume equipment
EPROM	Read-mostly	UV light	No	No	Device prototyping
EEPROM	Read-mostly	Electrical	Yes	No	Device prototyping
Flash	Read/write	Electrical	No	No	Film for digital camera

Field Programmable Gate Array (FPGA)

- Consentono di realizzare circuiti logici arbitrari
- Due componenti replicati
  - LookUp Tables (LUT): piccola memoria che si usa per implementare una qualunque funzione booleana
  - Connessioni programmabili



(a)

Signal Assignment

FPGA	Counter
I_3	CLR
$O_{2..0}$	$O_{2..0}$
CK	CK

Addr	Data
0	1
1	2
2	3
3	0

Addr	Data
4	0
5	0
6	0
7	0

(b)

	f1	f2	f3	f4
0000				
0001				
0010				
0011				

Esercizio

Si vuole realizzare un circuito combinatorio che effettui un controllo di parità su tre linee digitali:

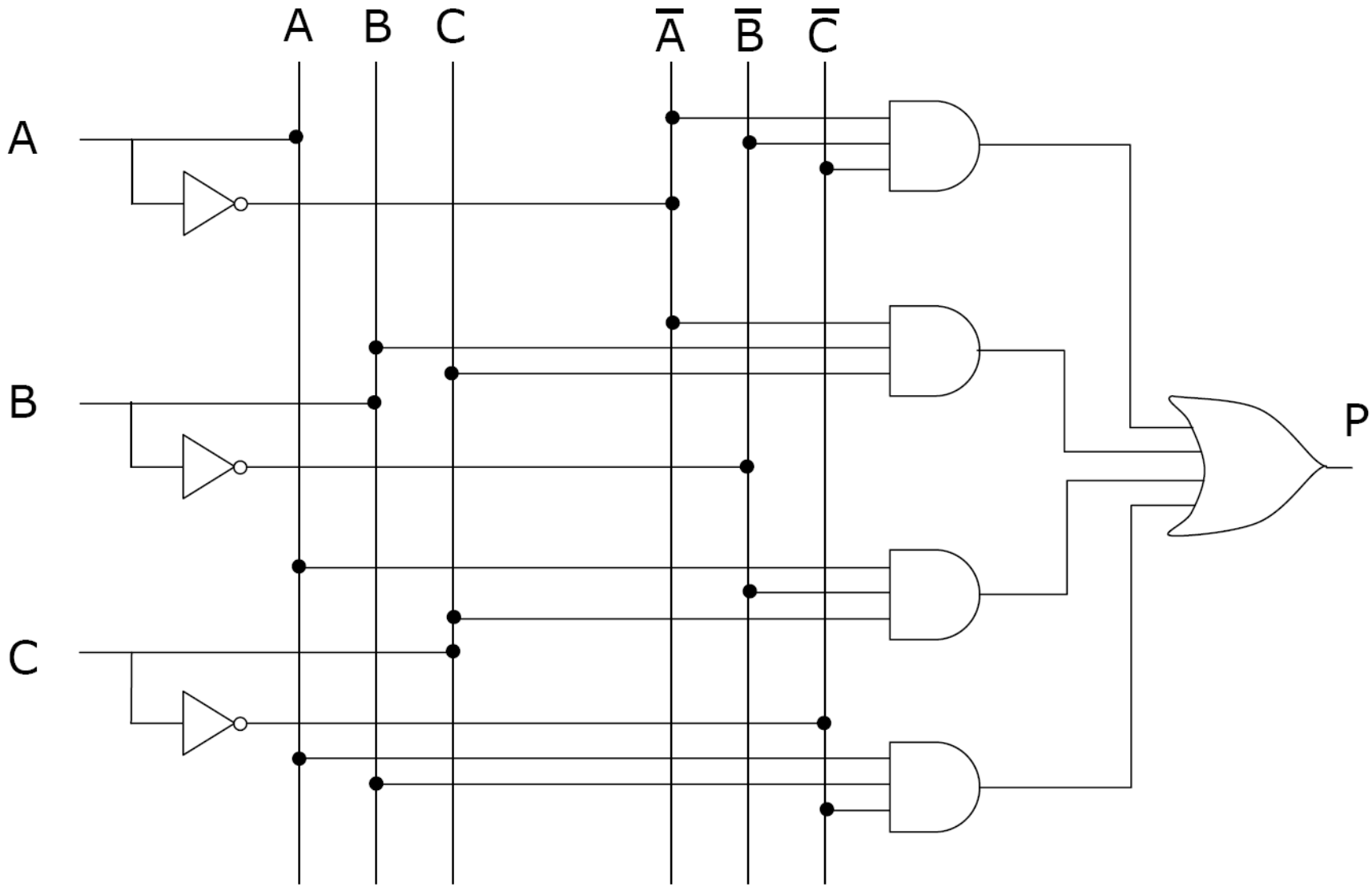
- realizzare il circuito mediante porte logiche;
- indicare come bisogna trasformare il circuito ottenuto per ottenere un circuito equivalente contenente solo porte NAND;
- realizzare il circuito con un singolo multiplexer.

Soluzione

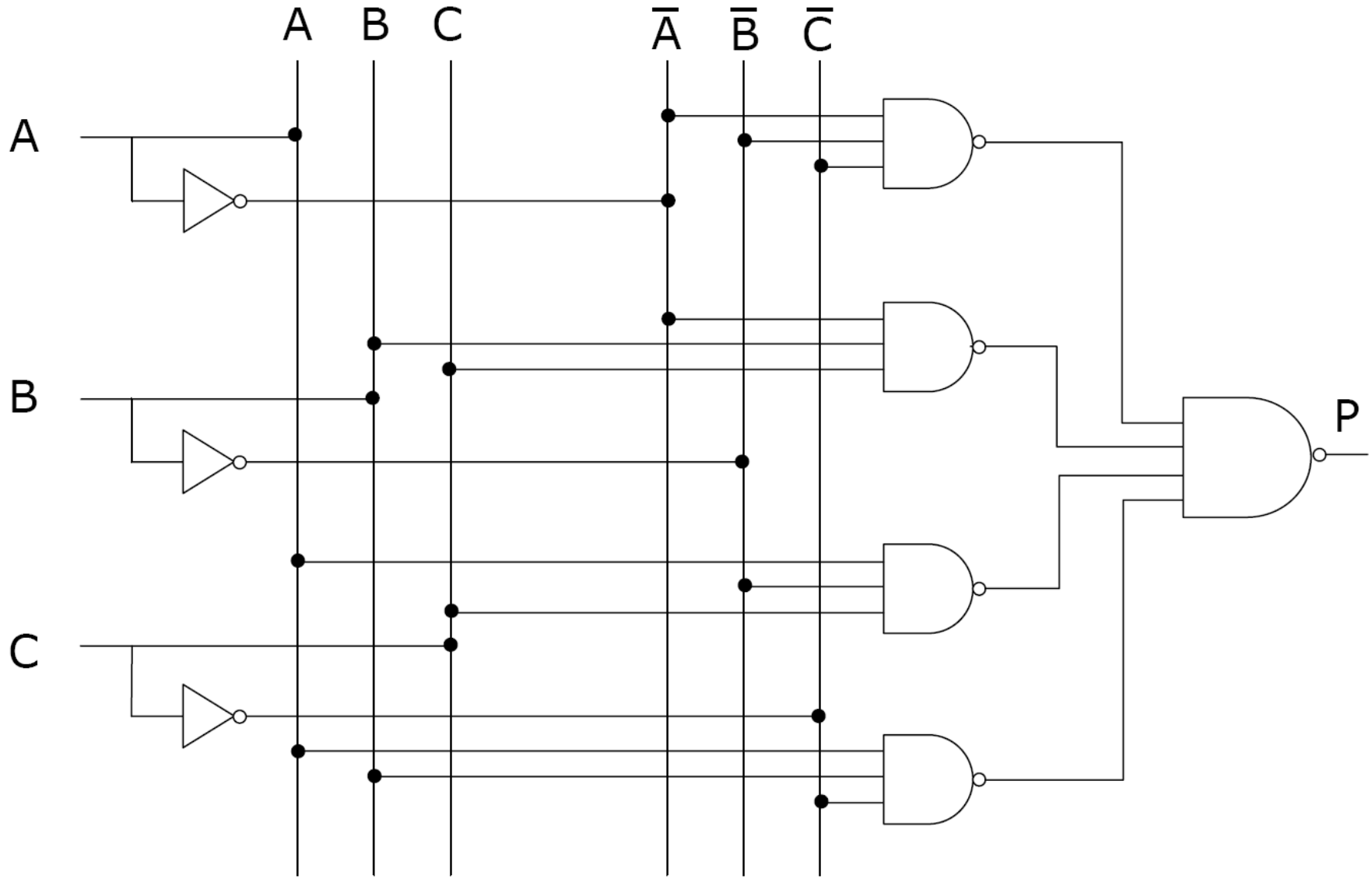
A	B	C	OUT
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

$$P = \overline{A}\overline{B}\overline{C} + \overline{A}BC + A\overline{B}C + AB\overline{C}$$

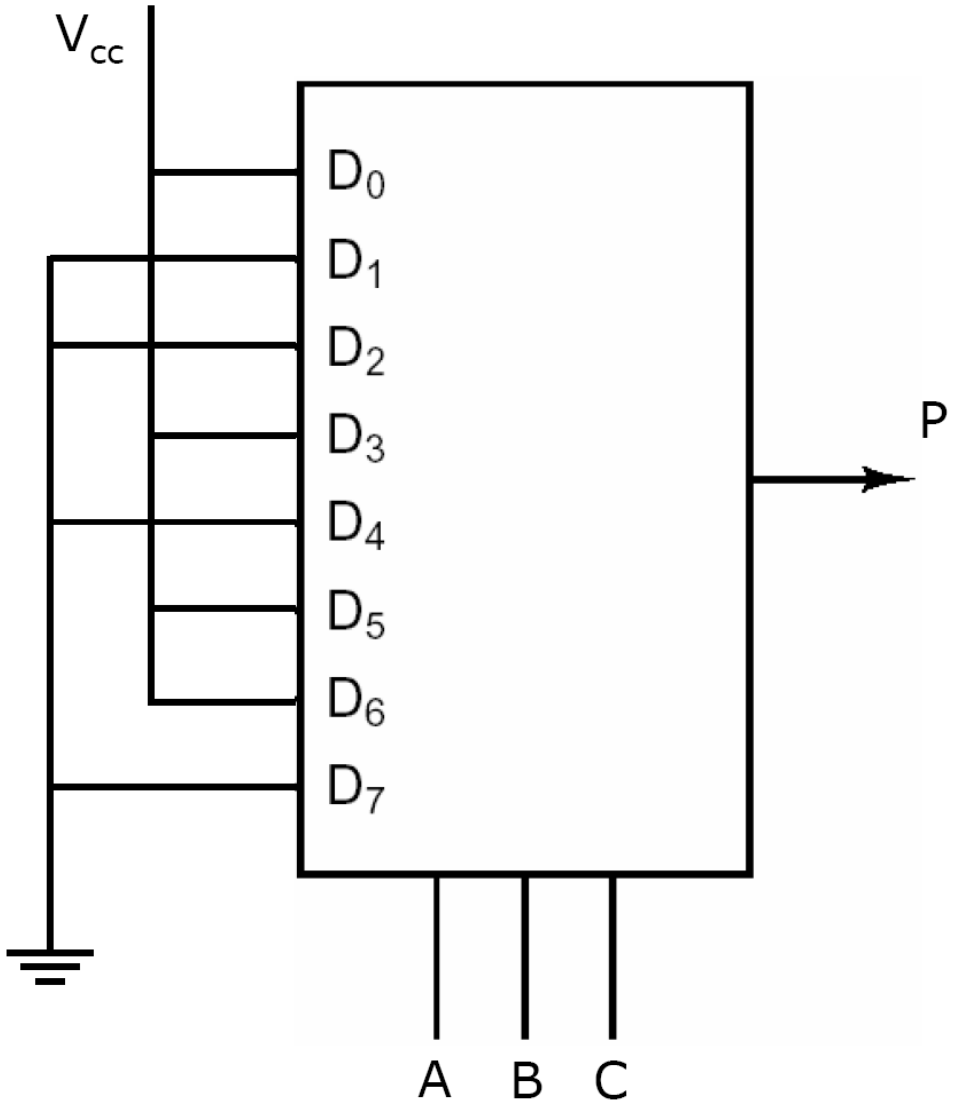
Soluzione con porte logiche qualsiasi



Soluzione con solo Porte NAND



Soluzione con multiplexer



Esercizio

Si vuole realizzare un circuito combinatorio che ha in ingresso tre segnali A, B e C e che si comporta come segue:

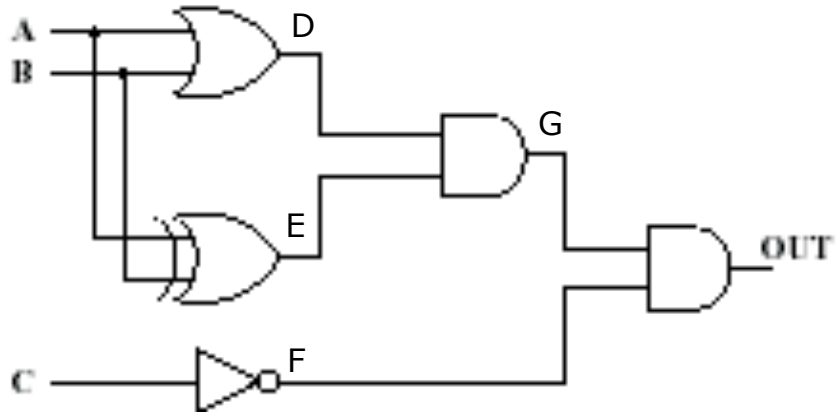
a) quando C=0 fa un test di uguaglianza ovvero restituisce 1 se A e B sono uguali e 0 altrimenti,

b) quando C=1 fa un test di disuguaglianza ovvero restituisce 1 se A e B sono diversi e 0 altrimenti.

- realizzare il circuito mediante porte logiche qualunque;
- realizzare il circuito con un multiplexer;
- realizzare il circuito utilizzando solo porte NAND e XOR.

Esercizio

Si consideri il seguente circuito combinatorio:



- Determinare la tabella di verità corrispondente al circuito.
- Indicare la funzione booleana in prima forma canonica corrispondente alla tabella di verità ottenuta
- Semplificare, se possibile, l'espressione booleana rappresentata dalla prima forma canonica ottenuta e disegnare il circuito corrispondente

Esercizio

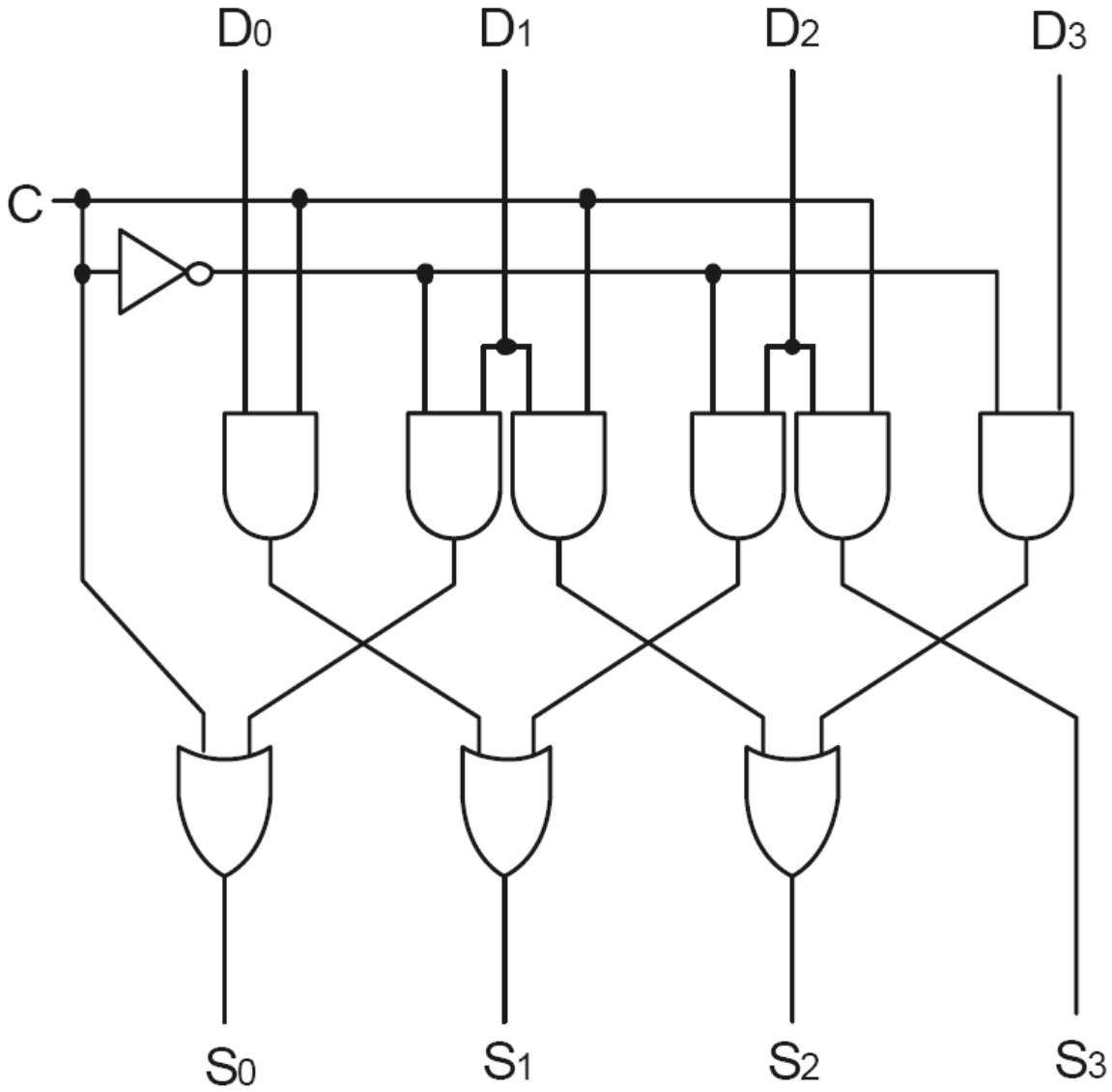
Fornire lo schema di uno shifter a 4 ingressi e 4 uscite che, sulla base di un segnale di controllo C:

a) sposta l'ingresso di un bit a sinistra riempiendo il bit meno significativo con 0 se C=0 e

b) sposta l'ingresso di un bit a destra riempiendo il bit piu significativo con 1 se C=1.

Illustrare sinteticamente il funzionamento del circuito.

Possibile soluzione



Esercizio

Fornire lo schema di uno shifter a 2 ingressi e 2 uscite che, sulla base di un segnale di controllo C:

a) sposta l'ingresso di un bit a sinistra riempiendo il bit meno significativo con 0 se C=0 e

b) sposta l'ingresso di un bit a destra riempiendo il bit più significativo con il bit più significativo dell'ingresso se C=1.

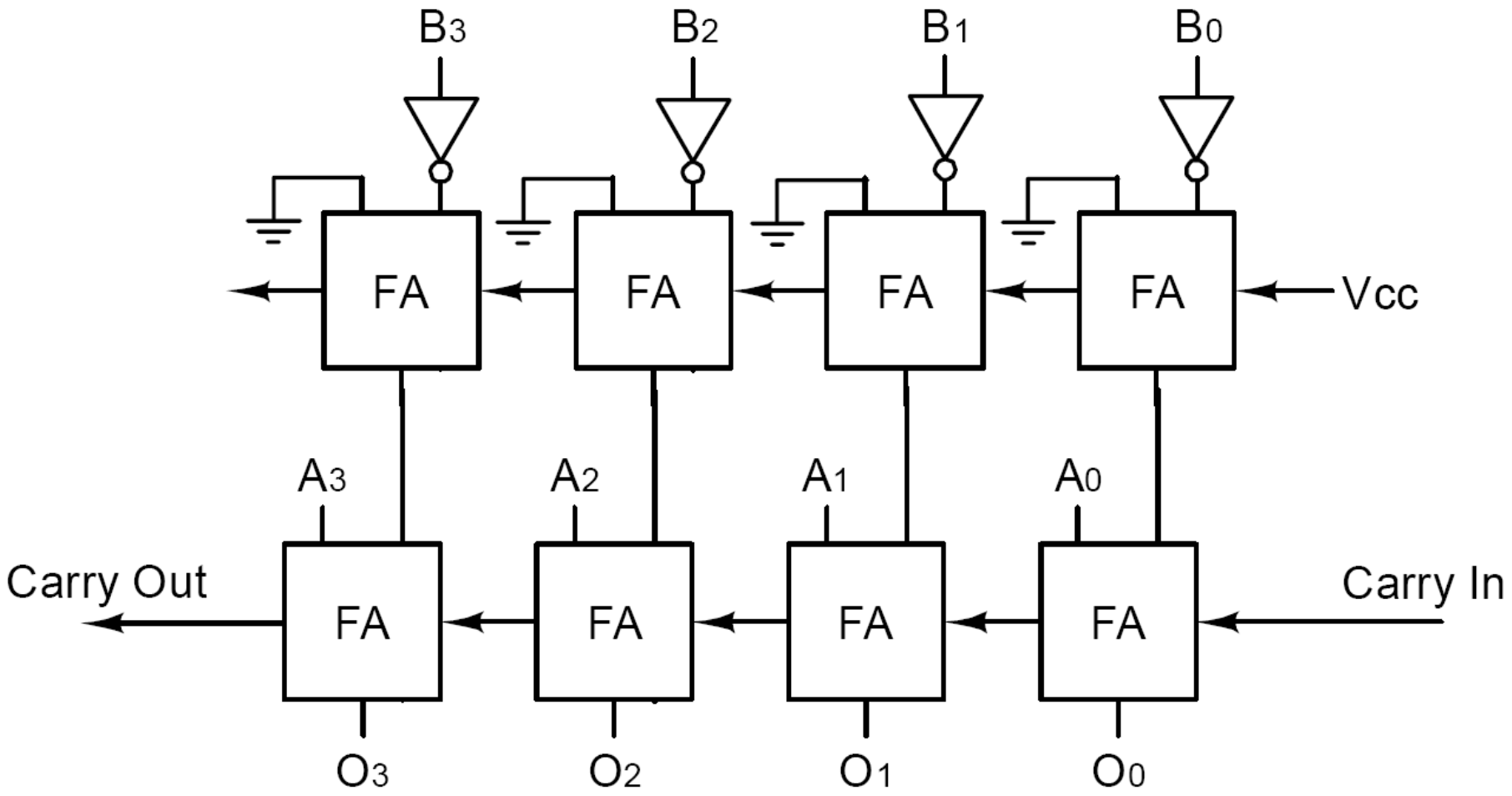
Illustrare sinteticamente il funzionamento del circuito.

Esercizio

Fornire lo schema di un sottrattore a 4 bit per notazione in complemento a 2 realizzato con sommatori completi.

- Illustrarne concisamente il funzionamento,
- specificare il valore di uscita di ciascuna componente quando in un ingresso (minuendo) c'è il numerale 0011 e nell'altro (sottraendo) il numerale 0100.

Una possibile soluzione



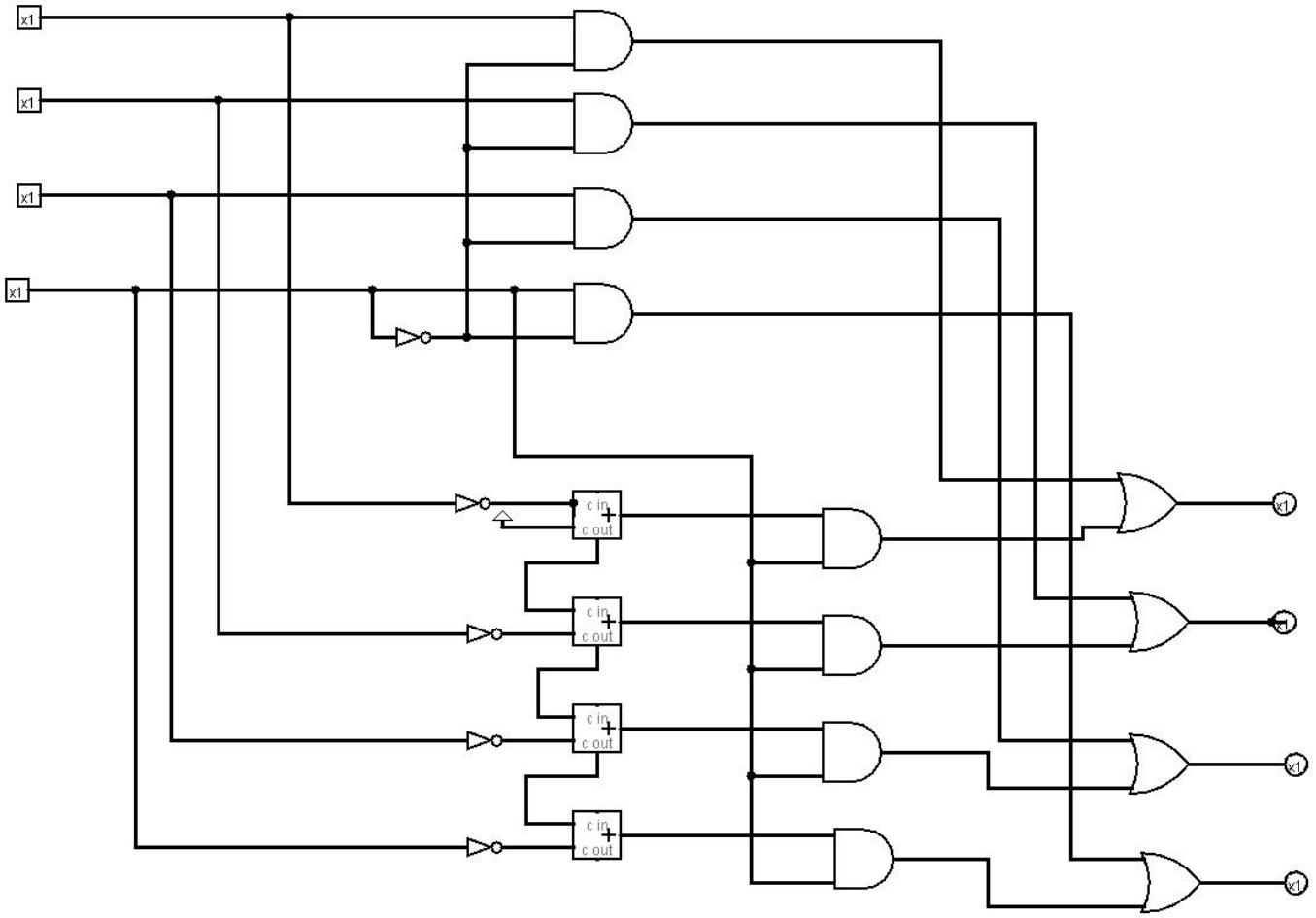
Esercizio

Fornire lo schema di un circuito combinatorio a 4 bit in grado di calcolare il valore assoluto di un numero, secondo il sistema di rappresentazione in complemento a due. Tale circuito, ricevuto in ingresso un numerale X a 4 bit, deve restituire in uscita:

- lo stesso numero in ingresso, se X rappresenta un numero positivo, e
- il numero in ingresso con il segno invertito, se X rappresenta un numero negativo (per esempio, se X = 3 allora l'uscita vale 3, se X = −1 allora l'uscita vale 1)

E' possibile utilizzare componenti di base quali half-adder e full-adder. Illustrare concisamente il funzionamento del circuito e specificare il valore di uscita di ciascuna componente quando l'ingresso si trova a 1011.

Possibile soluzione



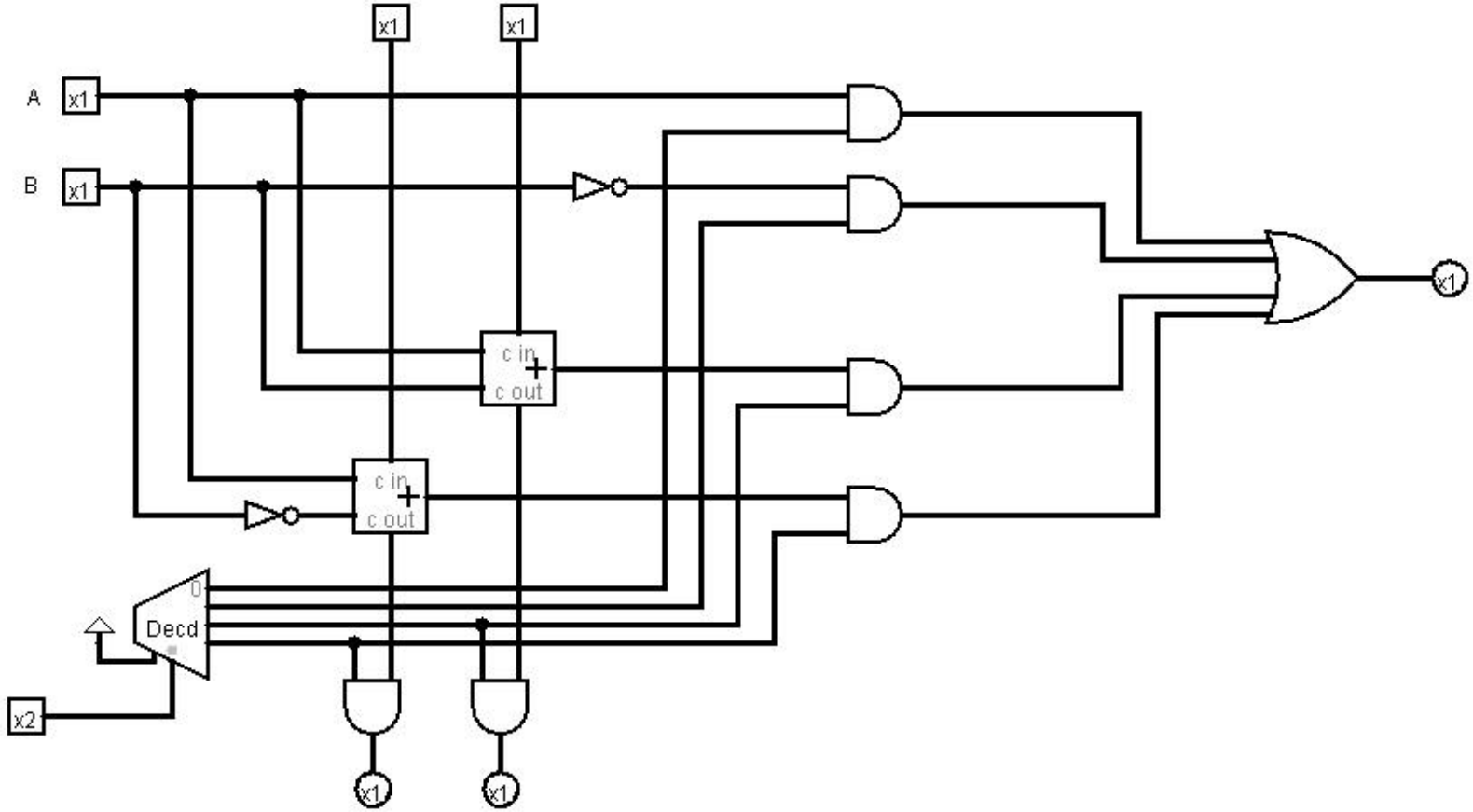
Esercizio

Si vuole progettare una piccola ALU avente due operandi in ingresso da 4 bit (A e B). Tale ALU deve essere in grado di svolgere, in base al valore di due segnali di controllo, le seguenti operazioni:

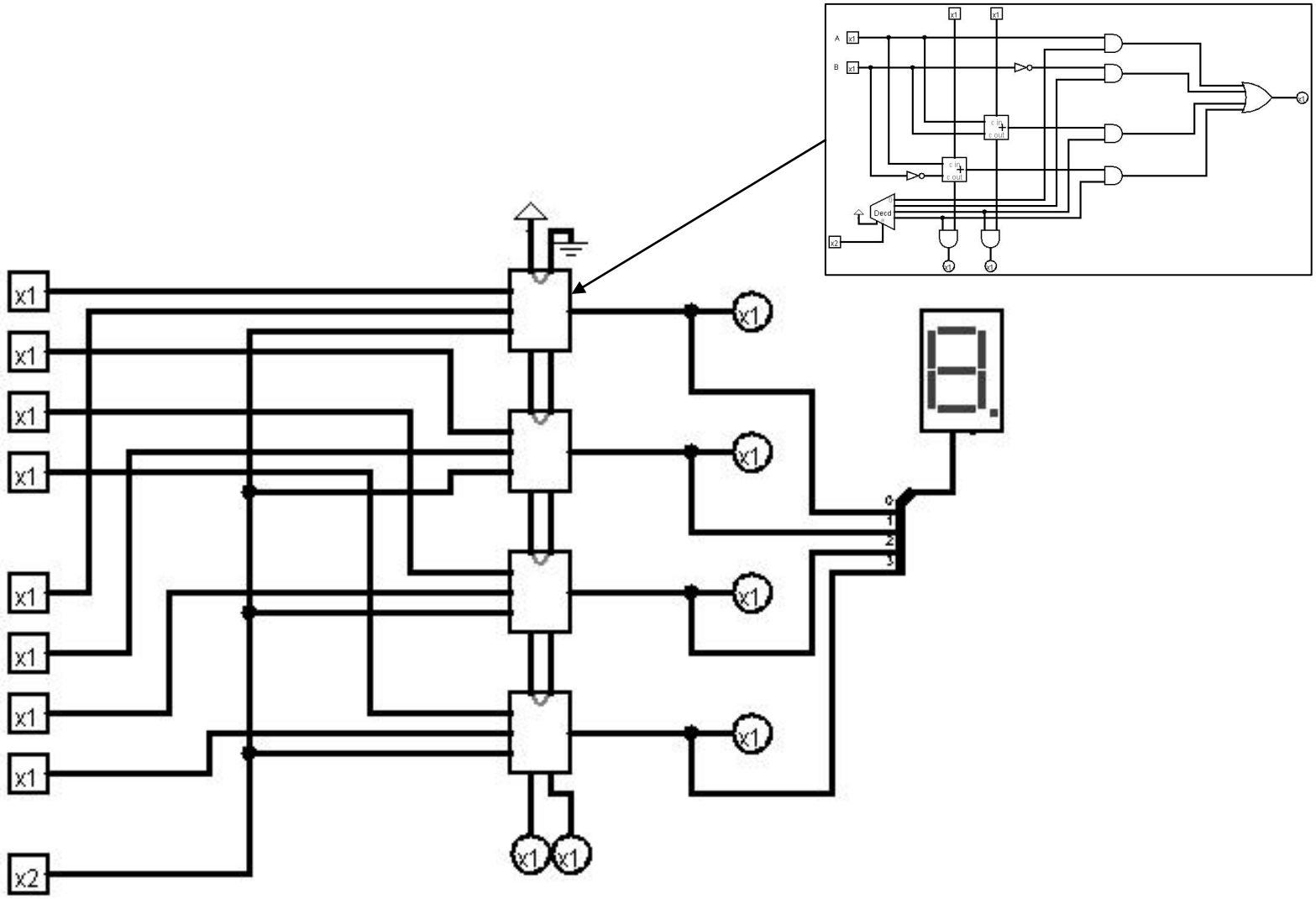
- la trasmissione di A inalterato (segnali di controllo: 00),
- l'inversione bit a bit di B (segnali di controllo: 01)
- la somma di A e B (segnali di controllo: 10) e
- la differenza di A e B (segnali di controllo: 11)

Definire lo schema di una ALU di questo tipo e illustrare sinteticamente il suo funzionamento. E' possibile utilizzare componenti predefiniti quali decodificatori e full adder.

Possibile soluzione a 1 bit



ALU a 4 bit ottenuta componendo quella a 1 bit



Esercizio

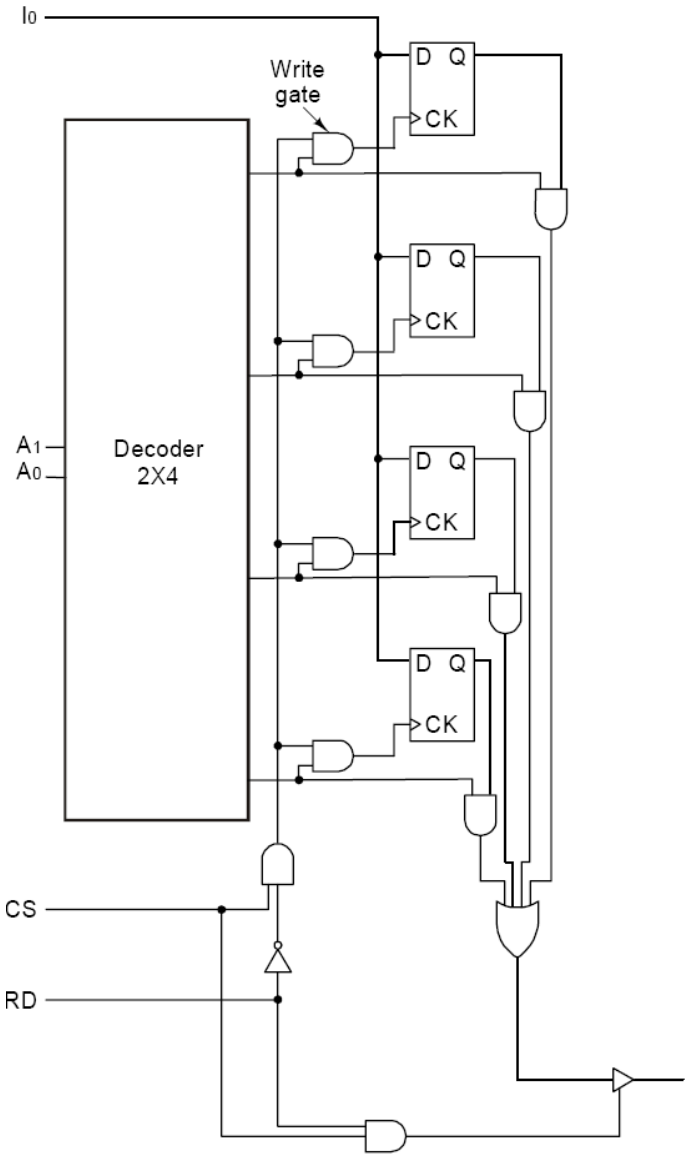
Fornire lo schema e illustrare sinteticamente il funzionamento di una piccola memoria di 4 locazioni da 1 bit ciascuna, realizzata con flip-flop e porte logiche.

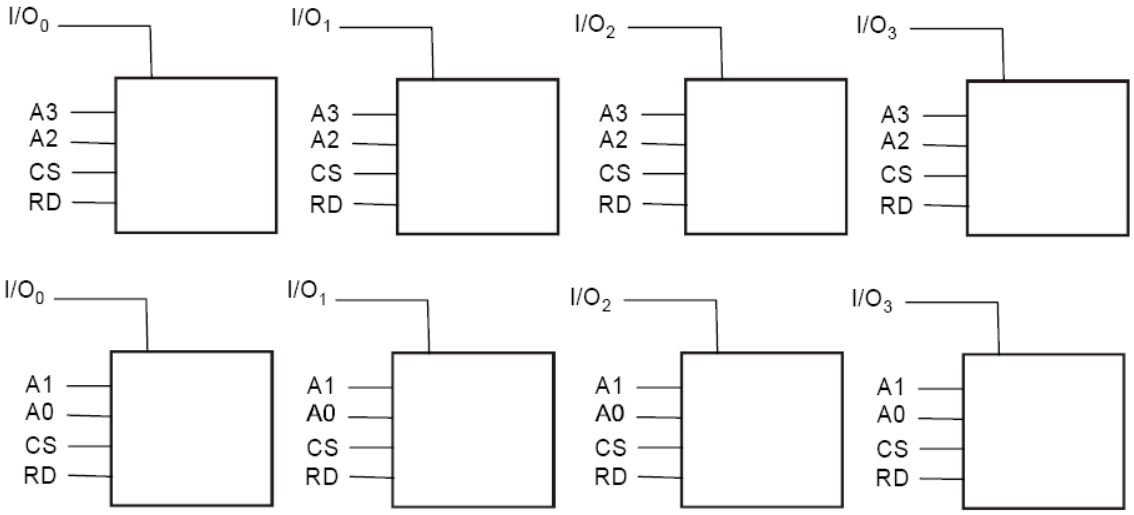
Tale memoria deve avere:

- 2 linee per la selezione della locazione,
- 1 linea condivisa per gli ingressi e le uscite,
- una linea di *chip select*,
- una linea per indicare se si vuole compiere una operazione di lettura o scrittura.

Indicare poi come è possibile utilizzare la memoria concepita per costruire una memoria di 8 locazioni da 4 bit.

Una possibile soluzione





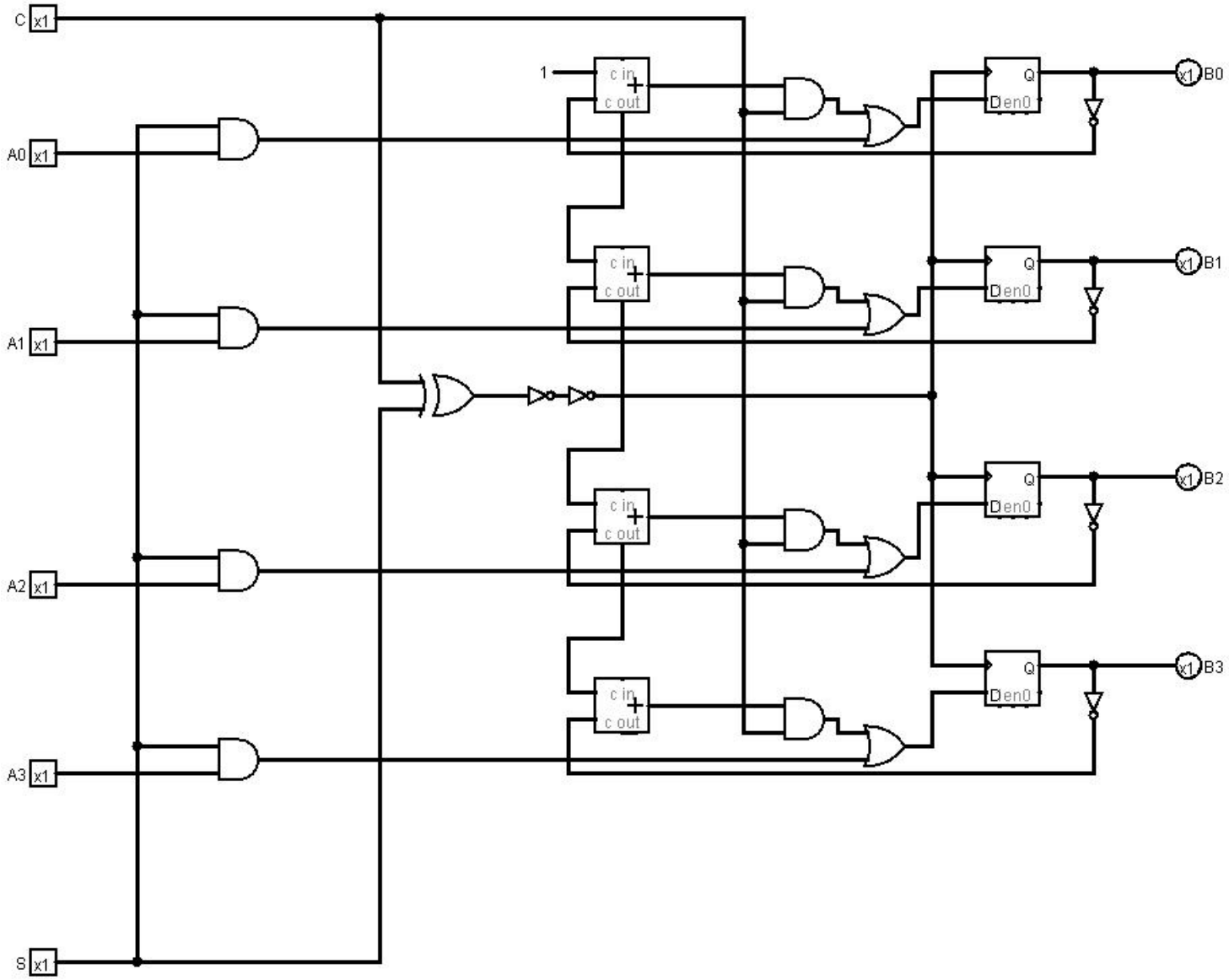
Esercizio

Fornire lo schema di un circuito sequenziale che realizza un registro a 4 bit complementabile, utilizzando half-adder, full-adder e flip-flop (come scatole chiuse). Tale circuito deve avere un segnale di set (S), un segnale di controllo (C), 4 linee di ingresso ($X_3X_2X_1X_0$) e 4 linee di uscita ($Y_3Y_2Y_1Y_0$).

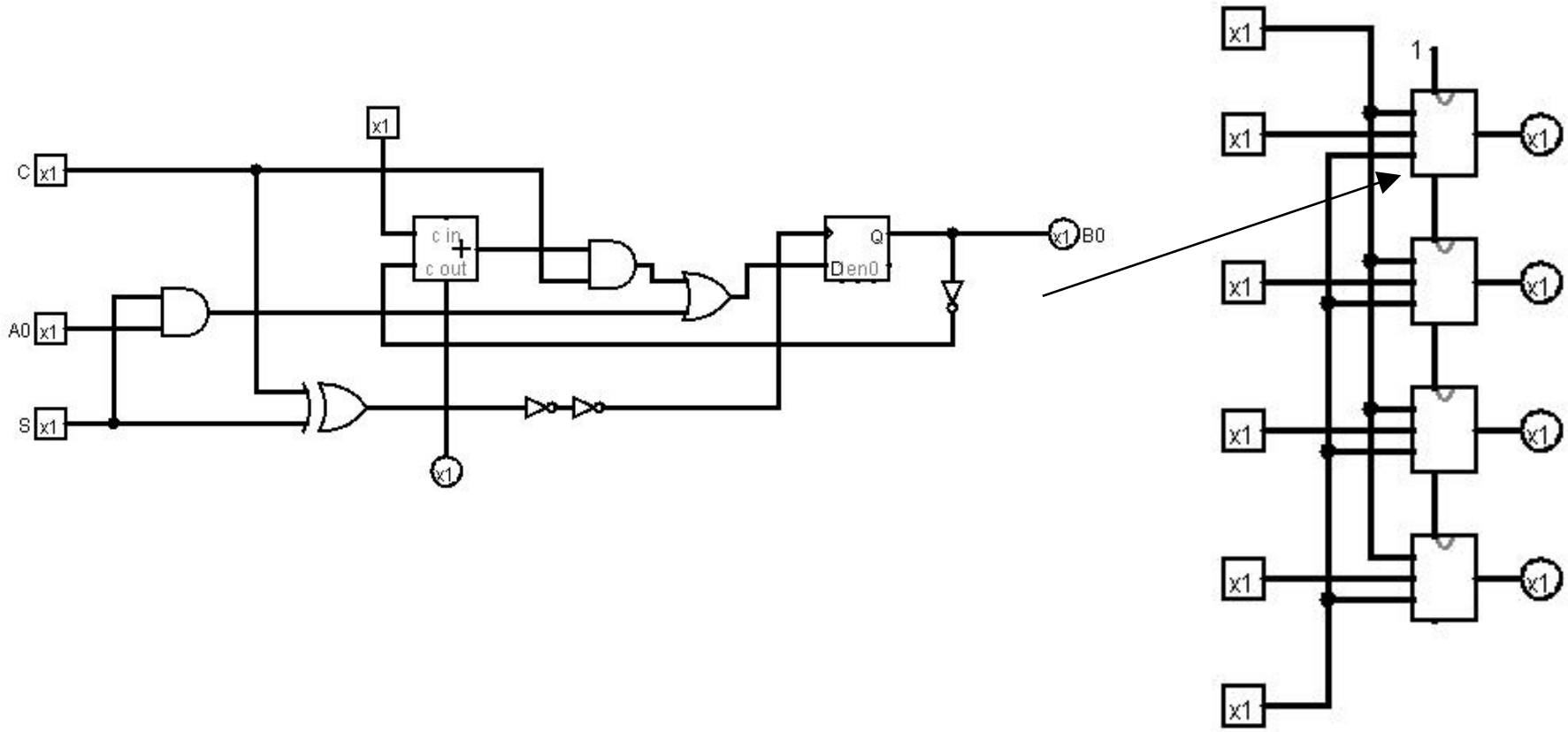
- Quando S=1 e C=0 il registro memorizza i segnali presenti sugli ingressi.
- Quando C=1 e S=0 gli ingressi vengono ignorati e il contenuto del registro viene complementato a due (es. da 0101 si passa a 1011).
- In ogni istante il contenuto del registro può essere letto sulle uscite.

Illustrare concisamente il funzionamento e specificare il valore di uscita di ciascuna componente del circuito quando C=1, S=0 e il registro memorizza 0111.

Possibile soluzione



Altra soluzione che compone un modulo da 1 bit



Homework primo esercizio

Fornire lo schema di un circuito logico che realizza un contatore di segnali digitali bassi. Tale circuito ha tre linee di ingresso e due di uscita: in un dato istante, l'uscita restituisce nel **sistema di rappresentazione binaria dei numeri naturali** il numero delle linee di ingresso pari a 0 (esempi: ingresso 101 – uscita 01; ingresso: 111 – uscita 00; ingresso 010 – uscita 10; ingresso 000 – uscita 11).

Homework secondo esercizio

Fornire lo schema di un circuito sequenziale che implementa un *registro a scalatura circolare* a 4 bit. Tale circuito ha 4 ingressi (A1A2A3A4), due segnali di controllo (W e S) e 4 uscite (B1B2B3B4). Quando W=1 e S=0 (comando di write) nel registro vengono memorizzati i dati in ingresso. Con W=0 e S=0 il registro mantiene il suo contenuto. Quando W=0 e S=1 (comando di shift) il contenuto del registro viene scalato a destra e il bit meno significativo viene spostato nella posizione più significativa. In ogni istante, il contenuto del registro è presente sulle linee di uscita (esempi: con ingresso=0110, W=1 e S=0 le uscite vanno a 0110; poi con ingresso qualunque, W=0 e S=1 le uscite vanno a 0011; poi con ingresso qualunque, W=0 e S=1 le uscite vanno a 1001; poi con ingresso qualunque, W=0 e S=1 le uscite vanno a 1100, ecc.).