Logica binaria

Moreno Marzolla
Dipartimento di Informatica—Scienza e Ingegneria (DISI)
Università di Bologna
http://www.moreno.marzolla.name/

Rappresentazione dell'informazione

- I calcolatori elettronici rappresentano qualsiasi tipo di informazione come una sequenza di cifre binarie (bit)
  - In particolare, sia i dati che il calcolatore elabora, sia le istruzioni che esegue, sono codificate con sequenze di bit
- Bit
  - Una singola cifra binaria: 0 oppure 1
- Byte
  - Una sequenza di 8 cifre binarie: es 0010 1110
- Parola (Word)
  - Una sequenza di 4 Byte (= 32 cifre binarie)

Nota storica

- La rappresentazione binaria non è l'unica possibile
- Sono stati realizzati calcolatori basati sulla *logica ternaria*
  - Possibili valori: -1, 0, 1

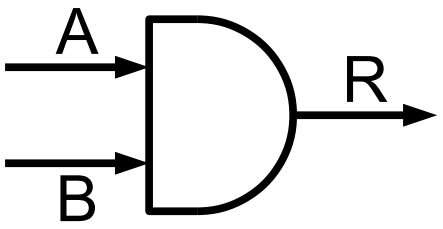
Calcolatore *Setun* (Сетунь) sviluppato nel 1958 presso l'Università di Stato di Mosca da Sergei Sobolev e Nikolay Brusentsov

Operazioni elementari sui bit

- Tre operazioni elementari

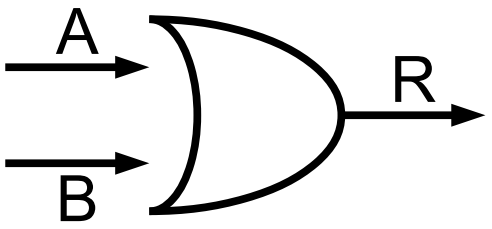
R = A AND B

A	B	R
0	0	0
0	1	0
1	0	0
1	1	1



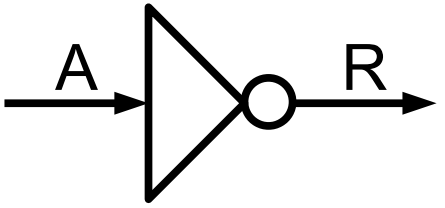
R = A OR B

A	B	R
0	0	0
0	1	1
1	0	1
1	1	1



R = NOT A

A	R
0	1
1	0



Alcune regole

- Elemento neutro
  - A AND 1 = 1 AND A = A
  - A OR 0 = 0 OR A = A
- Massimo e minimo
  - A AND 0 = 0 AND A = 0
  - A OR 1 = 1 OR A = 1
- Distributività
  - A AND (B OR C) = (A AND B) OR (A AND C)
  - A OR (B AND C) = (A OR B) AND (A OR C)
- De Morgan
  - NOT (A AND B) = (NOT A) OR (NOT B)
  - NOT (A OR B) = (NOT A) AND (NOT B)

Altre operazioni sui bit

- Ogni altra operazione su bit si può ottenere tramite composizione di AND, OR, NOT
- Esempio: Or Esclusivo (XOR, *eXclusive OR*)

A XOR B = 1 se e solo se
(A=0 AND B=1) OR
(A=1 AND B=0)

Ossia:

A XOR B ≡
((NOT A) AND B) OR
((A AND (NOT B))

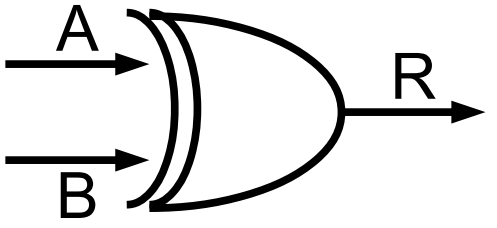
R = A XOR B

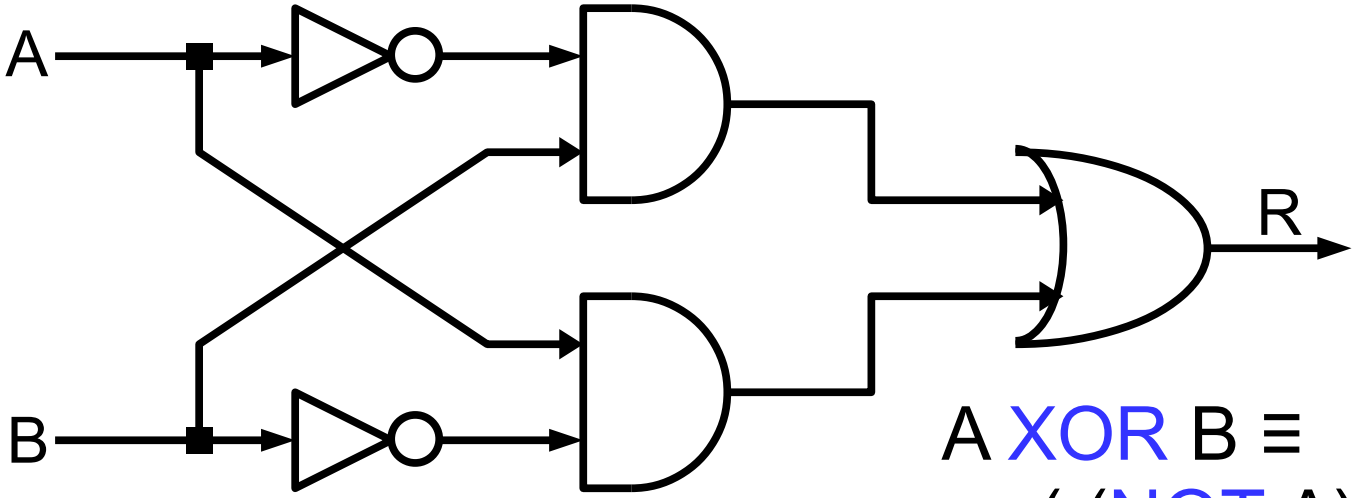
A	B	R
0	0	0
0	1	1
1	0	1
1	1	0

Operazioni sui bit: XOR

R = A XOR B

A	B	R
0	0	0
0	1	1
1	0	1
1	1	0





A XOR B ≡
((NOT A) AND B) OR
((A AND (NOT B))

In generale

Il metodo è valido per ogni tavola di verità

Esempio:

Il risultato è 1 se e solo se

(A=0 AND B=0) OR
(A=1 AND B=0) OR
(A=1 AND B=1)

A	B	R
0	0	1
0	1	0
1	0	1
1	1	1

Ossia

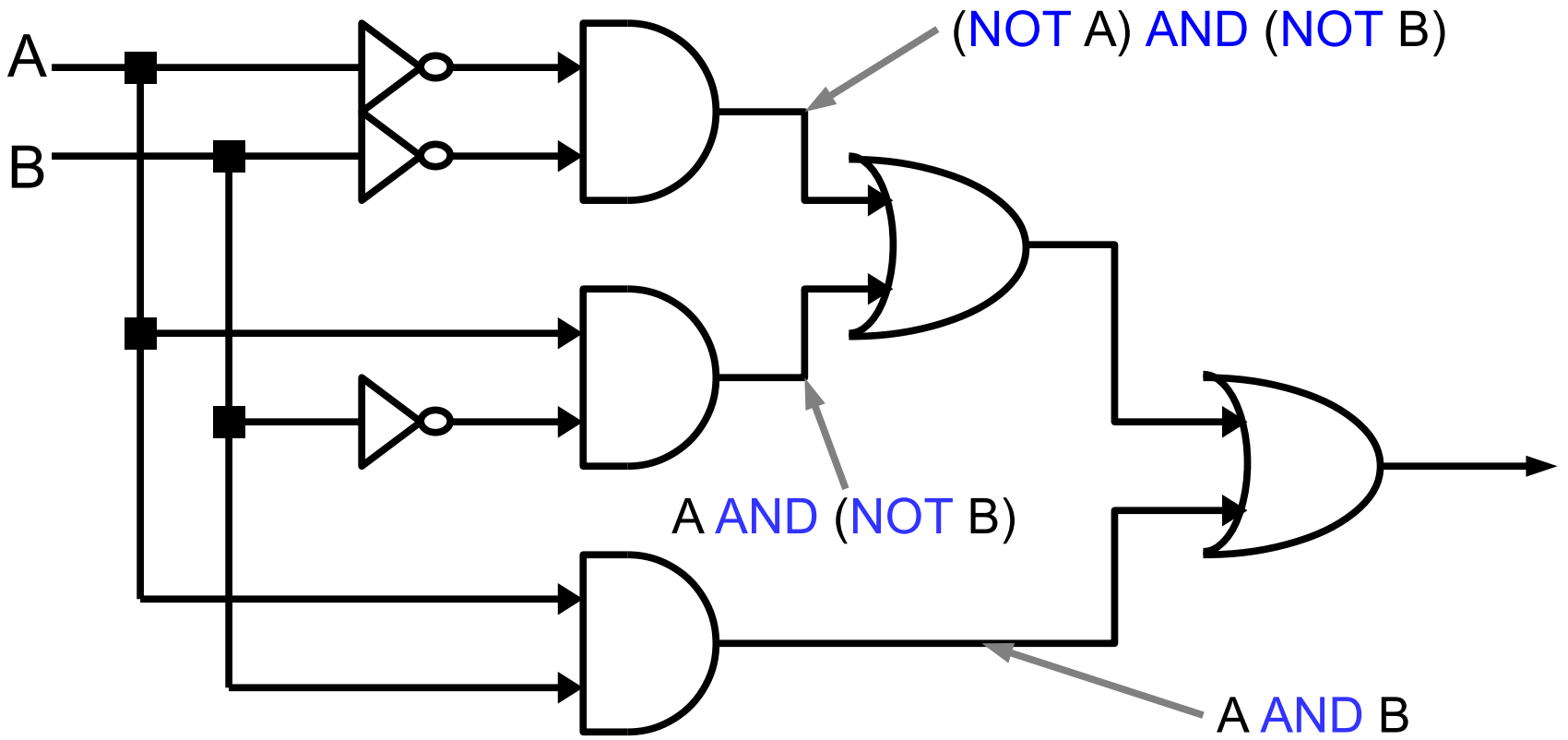
((NOT A) AND (NOT B)) OR
(A AND (NOT B)) OR
(A AND B)

***Nota**: questo procedimento non produce necessariamente l'espressione booleana "più semplice" per descrivere una funzione data. Nel corso di Calcolatori Elettronici verrà illustrato un algoritmo per rappresentare una tabella di verità con il numero minimo di operatori logici*

In generale

A	B	R
0	0	1
0	1	0
1	0	1
1	1	1

((NOT A) AND (NOT B)) OR
(A AND (NOT B)) OR
(A AND B)

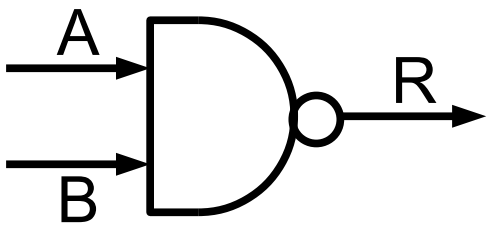


NOR e NAND

A NAND B ≡
NOT (A AND B)

R = A NAND B

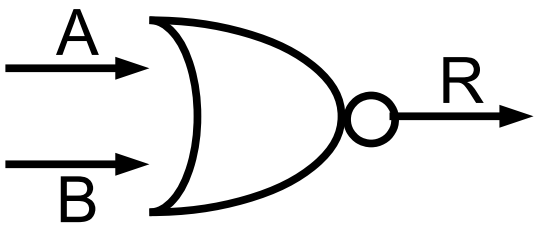
A	B	R
0	0	1
0	1	1
1	0	1
1	1	0



A NOR B ≡
NOT (A OR B)

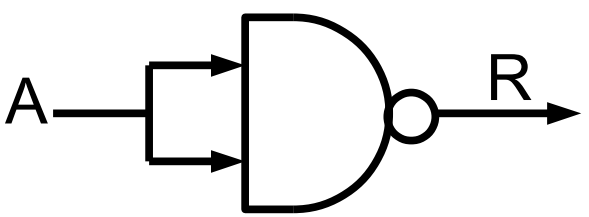
R = A NOR B

A	B	R
0	0	1
0	1	0
1	0	0
1	1	0

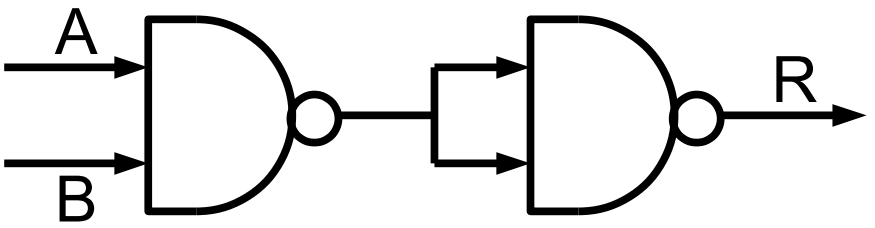


Le porte NAND sono universali

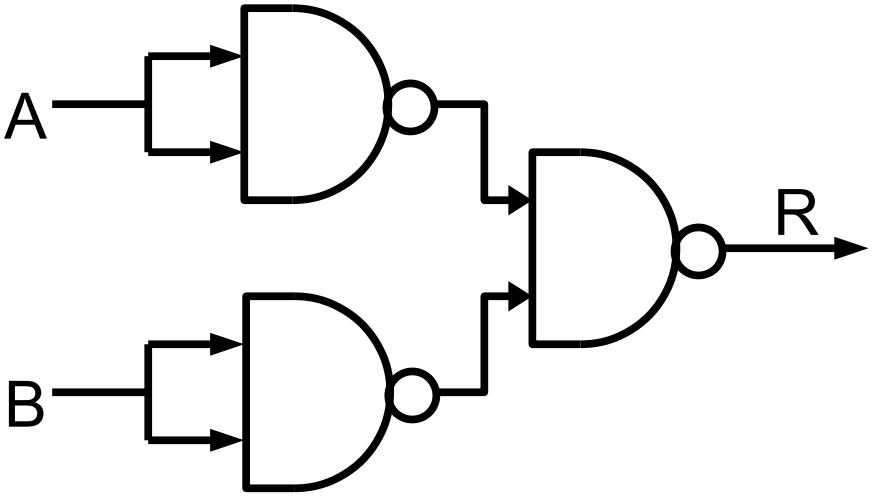
- Le porte AND, OR e NOT si possono ottenere usando esclusivamente porte NAND



NOT A



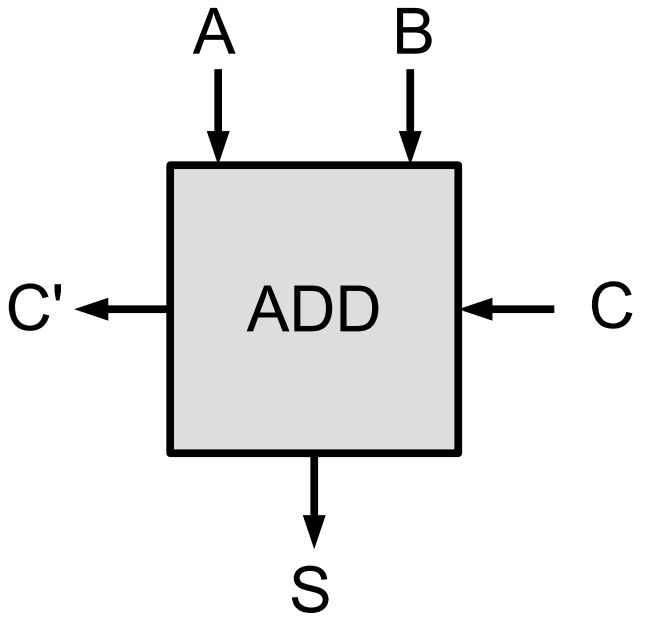
A AND B



A OR B

Sommatore a 1 bit

- Il sommatore a un bit (*1-bit Full Adder*) accetta in ingresso tre bit *A*, *B* e *C* (*Carry*, ossia “riporto”)...
- ...e calcola la somma *S*=*A*+*B*+*C* e il nuovo riporto *C*'



A	B	C	S	C'
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

Calcolare *S* e *C'*

A	B	C	S	C'
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

S ≡ ((NOT A) AND (NOT B) AND C) OR
((NOT A) AND B AND (NOT C)) OR
(A AND (NOT B) AND (NOT C)) OR
(A AND B AND C)

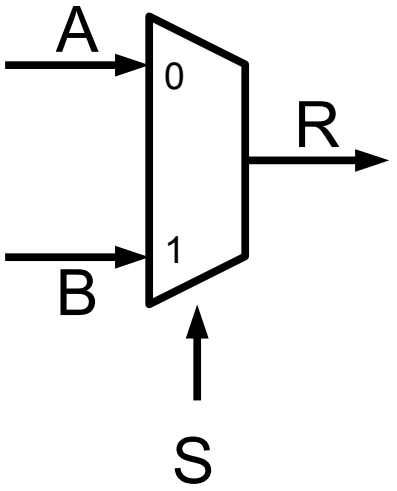
Calcolare *S* e *C'*

A	B	C	S	C'
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

C' ≡ ((NOT A) AND B AND C) OR
(A AND (NOT B) AND C) OR
(A AND B AND (NOT C)) OR
(A AND B AND C)

2-to-1 Multiplexer (Mux)

- Dispositivo con tre ingressi: *A*, *B* e *S* (Selettore)
  - L'output è *A* se *S*=0;
  - L'output è *B* se *S*=1.

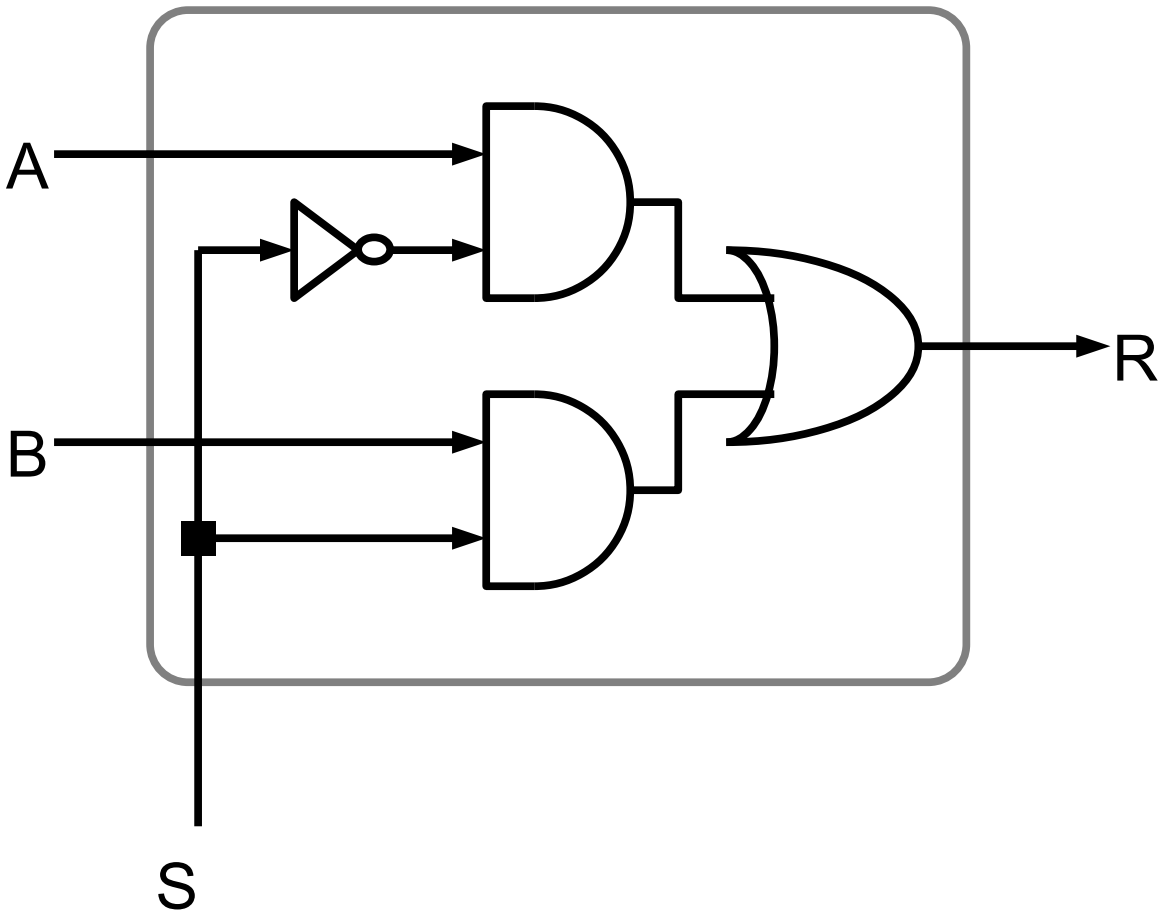


2-to-1 multiplexer

A	B	S	R
a	b	0	a
a	b	1	b

- In maniera analoga si possono definire 4-to-1, 8-to-1, … multiplexer
  - Sono necessari più bit per il selettore

2-to-1 Multiplexer (Mux)
una possibile realizzazione

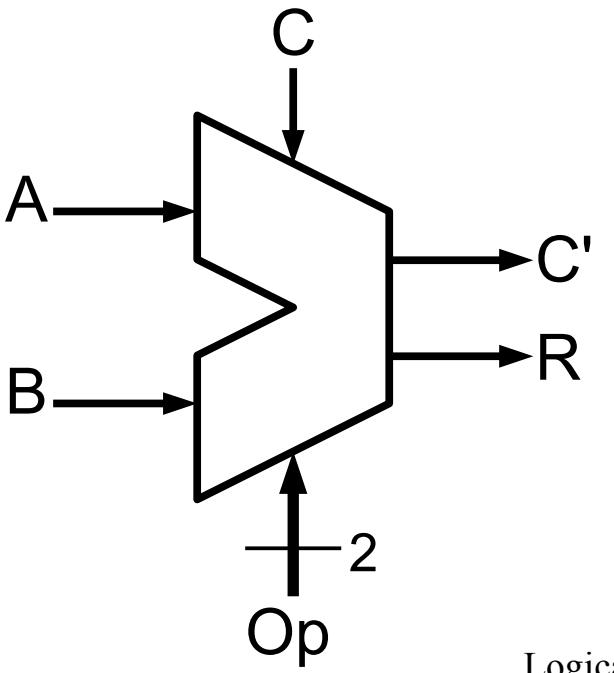


2-to-1 multiplexer

A	B	S	R
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Costruiamo una ALU

- ALU (*Arithmetic Logic Unit*)
  - Una componente fondamentale di una CPU
  - Effettua operazioni aritmetiche (es, somme, prodotti) e logiche (AND, OR, ...)
- Come esempio sviluppiamo una ALU ad un bit

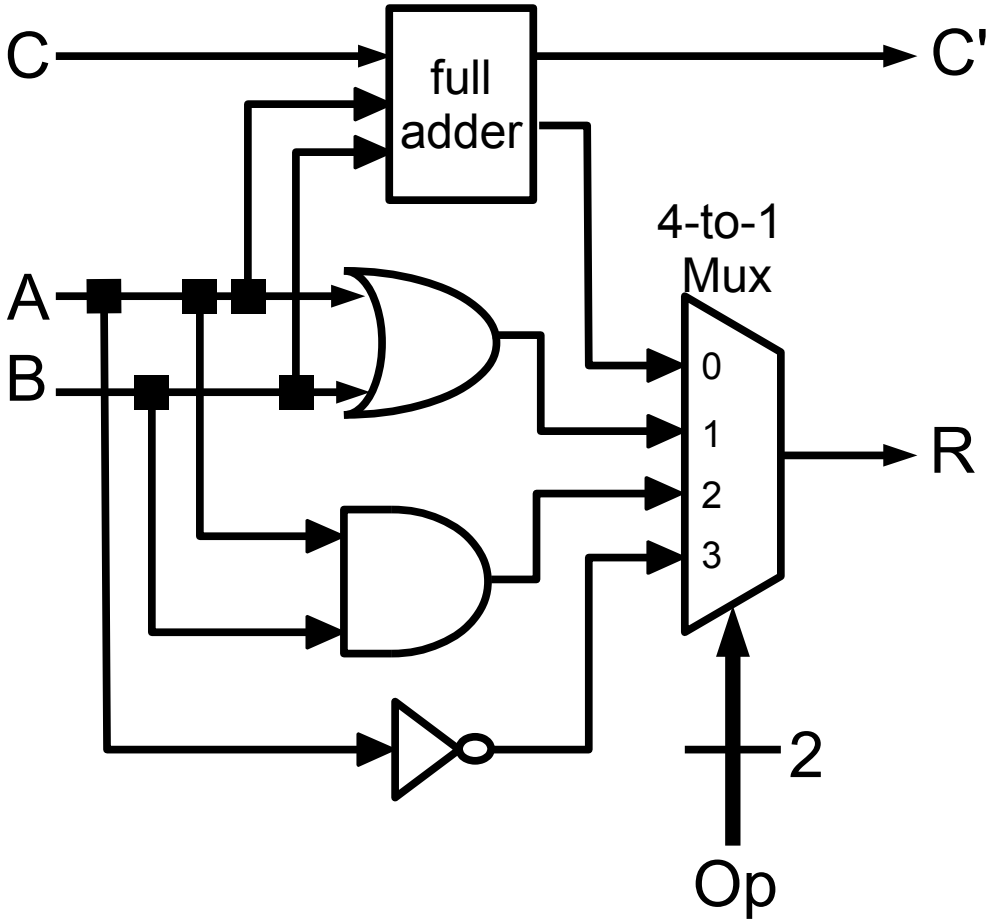


A	B	C	Op	R	C'
a	b	c	00	a+b+c	c'
a	b	x	01	a OR b	x
a	b	x	10	a AND b	x
a	b	x	11	NOT a	x

x = “don't care”

Costruiamo una ALU

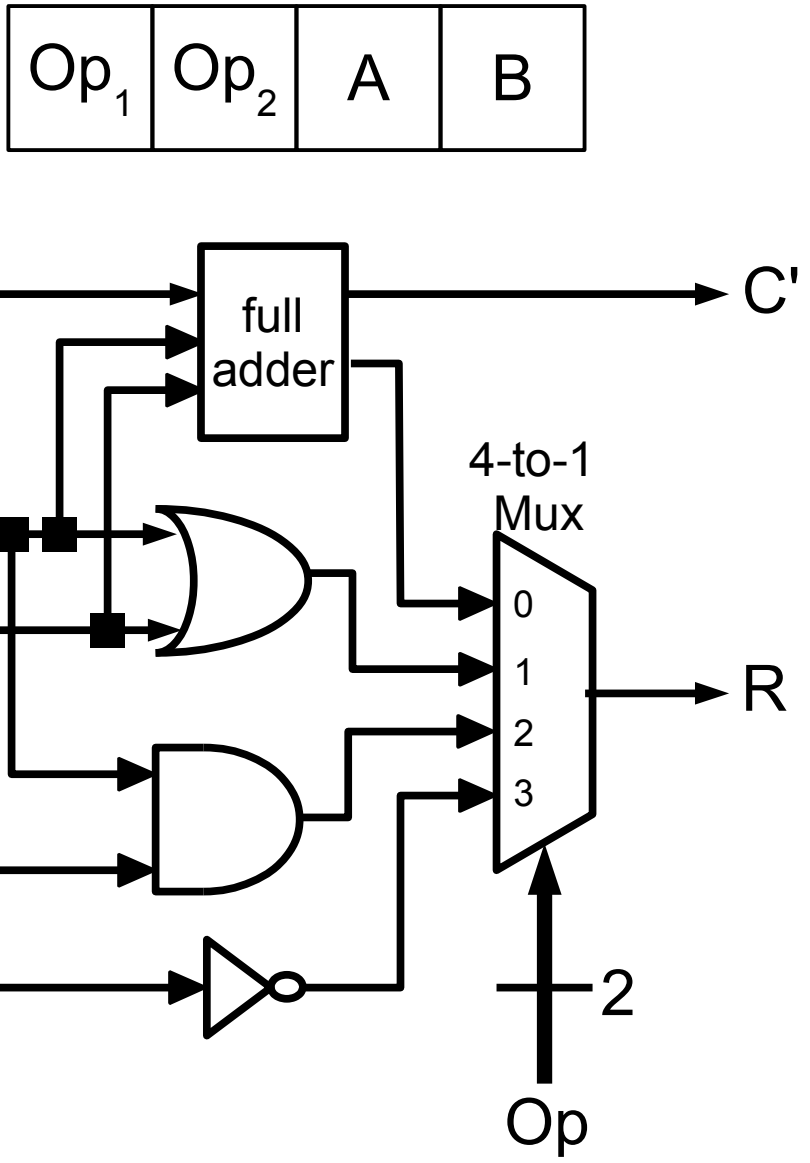
Usando un mux 4-a-1 possiamo selezionare l'operazione da compiere



Op	R
00	A+B+C
01	A or B
10	A and B
11	not A

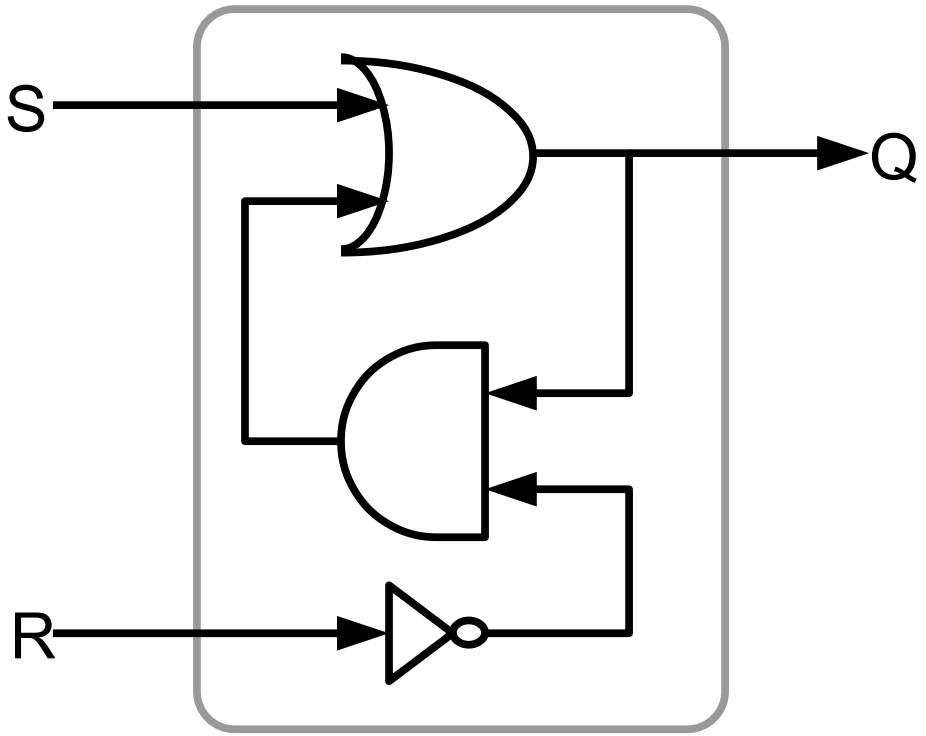
Programmare la ALU

- Si programma tramite micro istruzioni composte da 4 bit
  - I primi due indicano l'operazione
    - 00 → ADD
    - 01 → OR
    - 10 → AND
    - 11 → NOT
  - Gli altri due indicano i valori di *A* e *B*
  - Supponiamo che il valore di *C* provenga da altre vie
- Es:
  - 0001 → ADD 0 1
  - 1011 → AND 1 1



Flip-flop

- Semplice circuito logico che può essere utilizzato per memorizzare il valore di un singolo bit

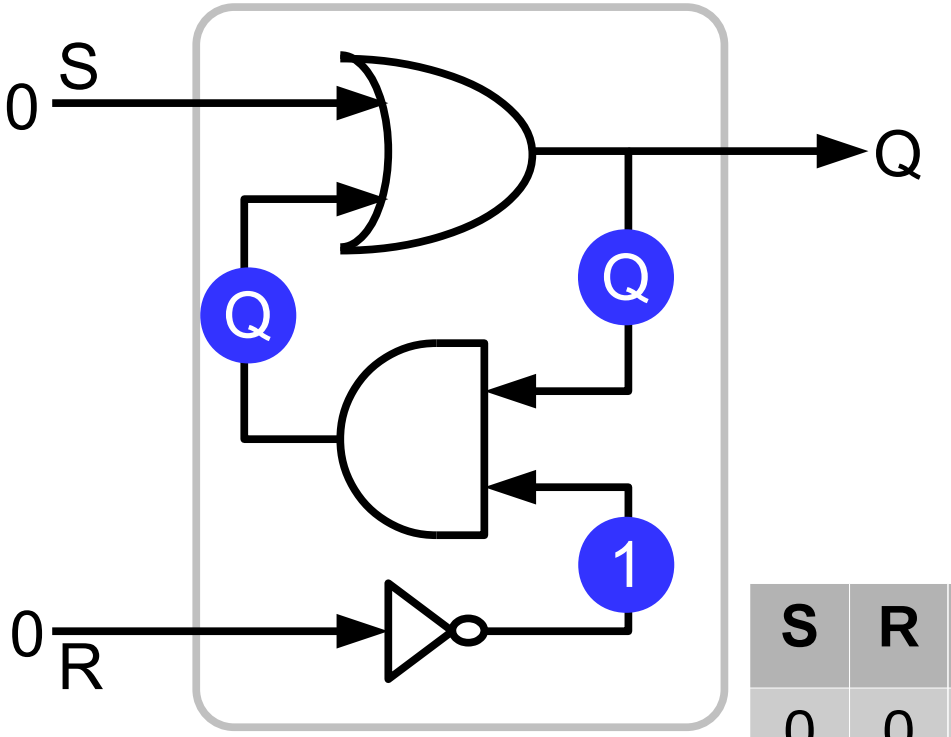


S	R	Q_{next}	Azione
0	0	Q	Mantieni stato precedente
0	1	0	Reset
1	0	1	Set
1	1	1	Set

Flip-flop

Ricordiamo:
X AND 0 = 0
X AND 1 = X

X OR 1 = 1
X OR 0 = X

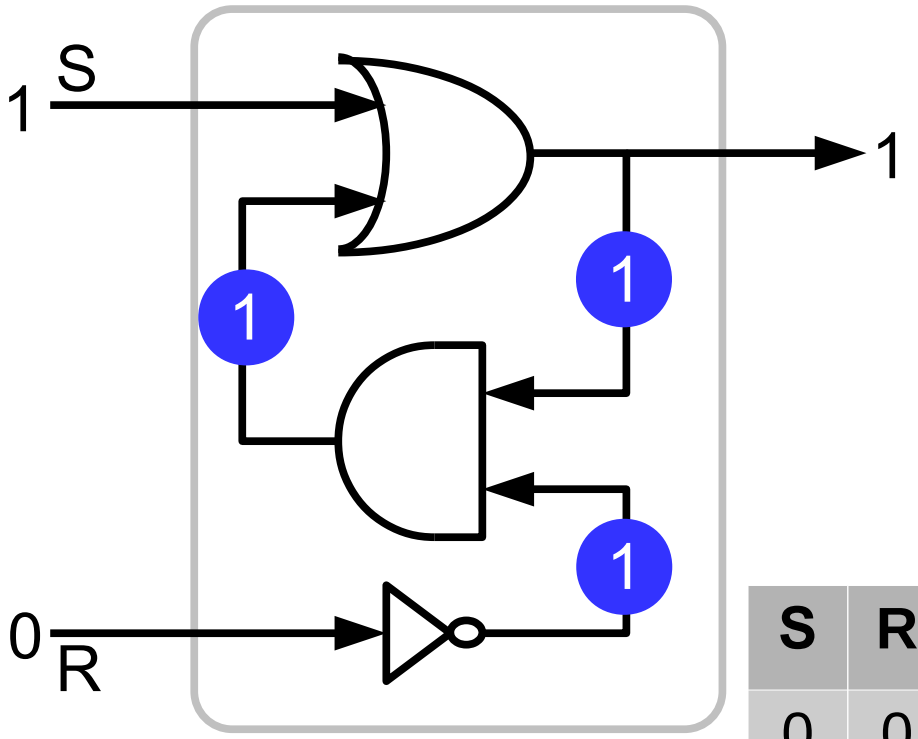


S	R	Q_{next}	Azione
0	0	Q	Mantieni stato prec.
0	1	0	Reset
1	0	1	Set
1	1	1	Set

Flip-flop

Nota:
X AND 0 = 0
X AND 1 = X

X OR 1 = 1
X OR 0 = X

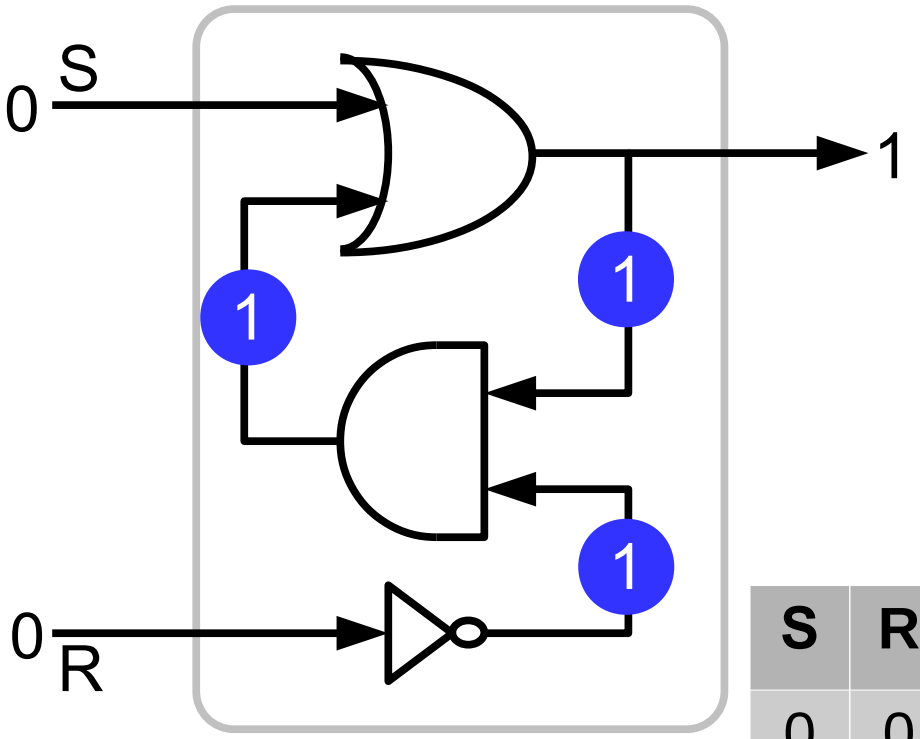


S	R	Q_{next}	Azione
0	0	Q	Mantieni stato prec.
0	1	0	Reset
1	0	1	Set
1	1	1	Set

Flip-flop

Nota:
X AND 0 = 0
X AND 1 = X

X OR 1 = 1
X OR 0 = X

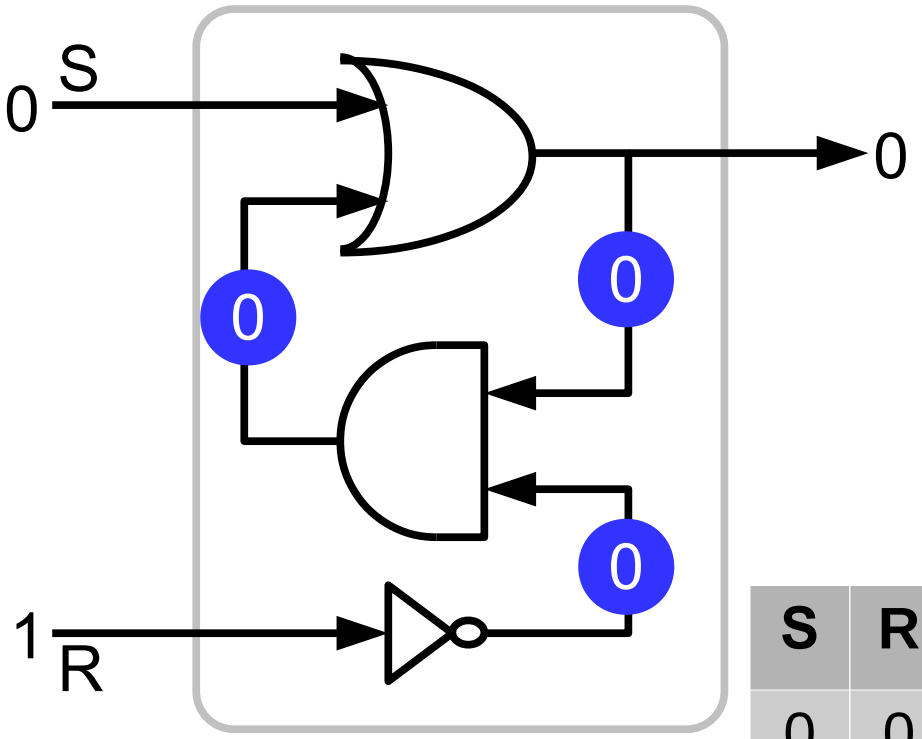


S	R	Q_{next}	Azione
0	0	Q	Mantieni stato prec.
0	1	0	Reset
1	0	1	Set
1	1	1	Set

Flip-flop

Nota:
X AND 0 = 0
X AND 1 = X

X OR 1 = 1
X OR 0 = X

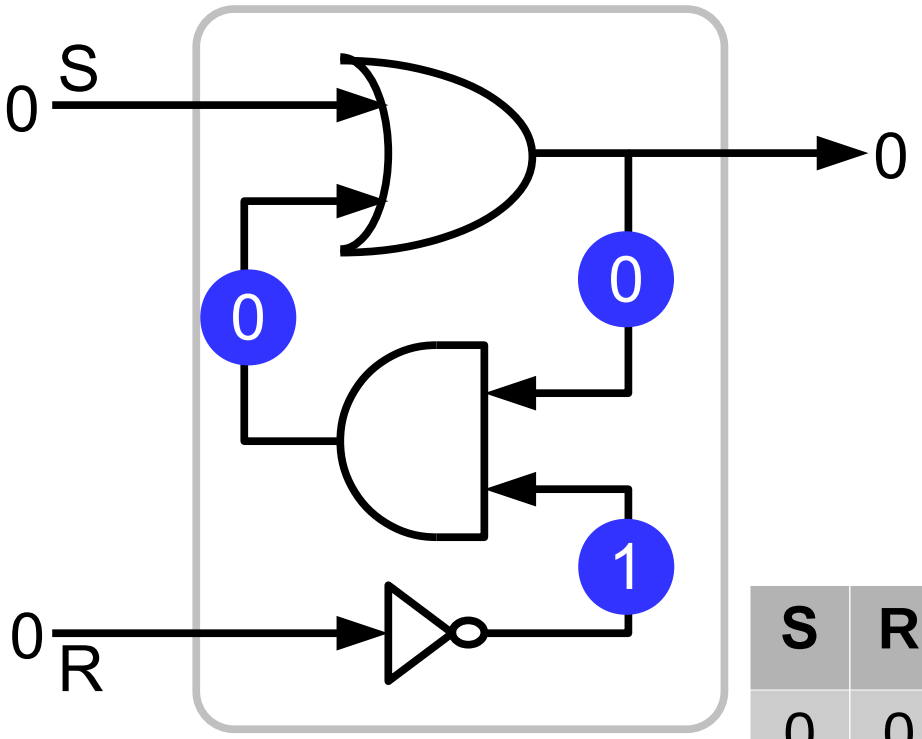


S	R	Q_{next}	Azione
0	0	Q	Mantieni stato prec.
0	1	0	Reset
1	0	1	Set
1	1	1	Set

Flip-flop

Nota:
X AND 0 = 0
X AND 1 = X

X OR 1 = 1
X OR 0 = X



S	R	Q_{next}	Azione
0	0	Q	Mantieni stato prec.
0	1	0	Reset
1	0	1	Set
1	1	1	Set

Esercizi

Per ciascuno dei circuiti seguenti, determinare la tebella di verità e disegnare il circuito booleano corrispondente

- [Parità] Disegnare un circuito booleano con tre ingressi, *A*, *B*, *C*, e una singola uscita *R*. Il valore di *R* deve essere tale che il gruppo di 4 bit *ABCR* deve avere un numero pari di cifre 1
  - Es: *ABC* = 110, *R*=0; *ABC*=111, *R*=1; *ABC*=010, *R*=1
- [Controllo parità] Disegnare un circuito booleano con quattro ingressi, *A*, *B*, *C*, *D*, e una singola uscita *R*. *R* vale uno se e solo se il gruppo di 4 bit *ABCD* contiene un numero pari di cifre 1
  - Es: *ABCD*=0110, *R*=1; *ABCD*=0111, *R*=0; *ABCD*=0101, *R*=1

Idee chiave

- Porte logiche (AND, OR, NOT, XOR)
- Tabelle di verità
- Espressioni logiche
- Semplici circuiti combinatori
  - MUX / DEMUX
  - 1-bit ALU
  - Flip-flop