

Logica binaria

Cap. 1.1 e 2.1 dispensa

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

Copyright © 2016–2019 Moreno Marzolla, Università di Bologna, Italy
<http://www.moreno.marzolla.name/teaching/FINFA/>



This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0) License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/4.0/> or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.

Rappresentazione dell'informazione

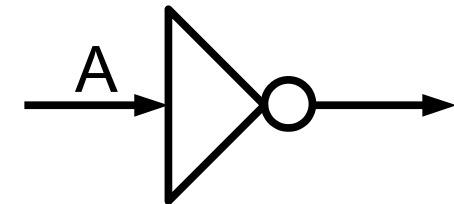
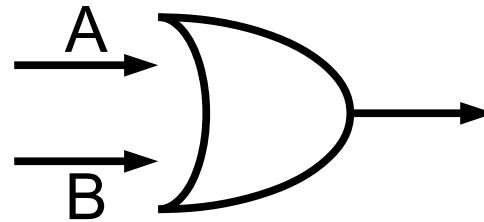
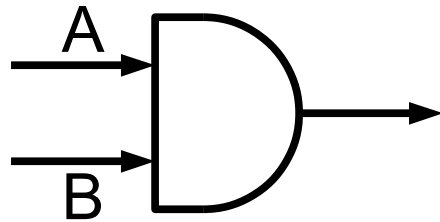
- I calcolatori elettronici rappresentano qualsiasi tipo di informazione come una sequenza di cifre binarie (bit)
 - In particolare, sia i dati che il calcolatore elabora, sia le istruzioni che esegue, sono codificate con sequenze di bit
- **Bit**
 - Una singola cifra binaria: 0 (*F, falso*) oppure 1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

Operazioni elementari sui bit

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

A	NOT A
0	1
1	0



- Notazione standard matematica:

- $A \wedge B = A$ AND B

- $A \vee B = A$ OR B

- $\neg A =$ NOT A

Alcune regole

- Elemento neutro
 - $A \text{ AND } 1 = 1 \text{ AND } A = A$
 - $A \text{ OR } 0 = 0 \text{ OR } A = A$
- Massimo e minimo
 - $A \text{ AND } 0 = 0 \text{ AND } A = 0$
 - $A \text{ OR } 1 = 1 \text{ OR } A = 1$
- Distributività
 - $A \text{ AND } (B \text{ OR } C) = (A \text{ AND } B) \text{ OR } (A \text{ AND } C)$
 - $A \text{ OR } (B \text{ AND } C) = (A \text{ OR } B) \text{ AND } (A \text{ OR } C)$
- De Morgan
 - $\text{NOT } (A \text{ AND } B) = (\text{NOT } A) \text{ OR } (\text{NOT } B)$
 - $\text{NOT } (A \text{ OR } B) = (\text{NOT } A) \text{ AND } (\text{NOT } B)$
- Assorbimento
 - $A \text{ AND } (A \text{ OR } B) = A$
 - $A \text{ OR } (A \text{ AND } B) = A$

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)

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

Ossia:

A **XOR** B \equiv
((**NOT** A) **AND** B) **OR**
((A **AND** (**NOT** B))

In generale

Il metodo è valido per ogni tavola di verità

Esempio:

Il risultato R è 1 se e solo se

$(A=0 \text{ AND } B=0) \text{ OR}$

$(A=1 \text{ AND } B=0) \text{ OR}$

$(A=1 \text{ AND } B=1)$

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

Ossia

$((\text{NOT } A) \text{ AND } (\text{NOT } B)) \text{ OR}$

$(A \text{ AND } (\text{NOT } B)) \text{ OR}$

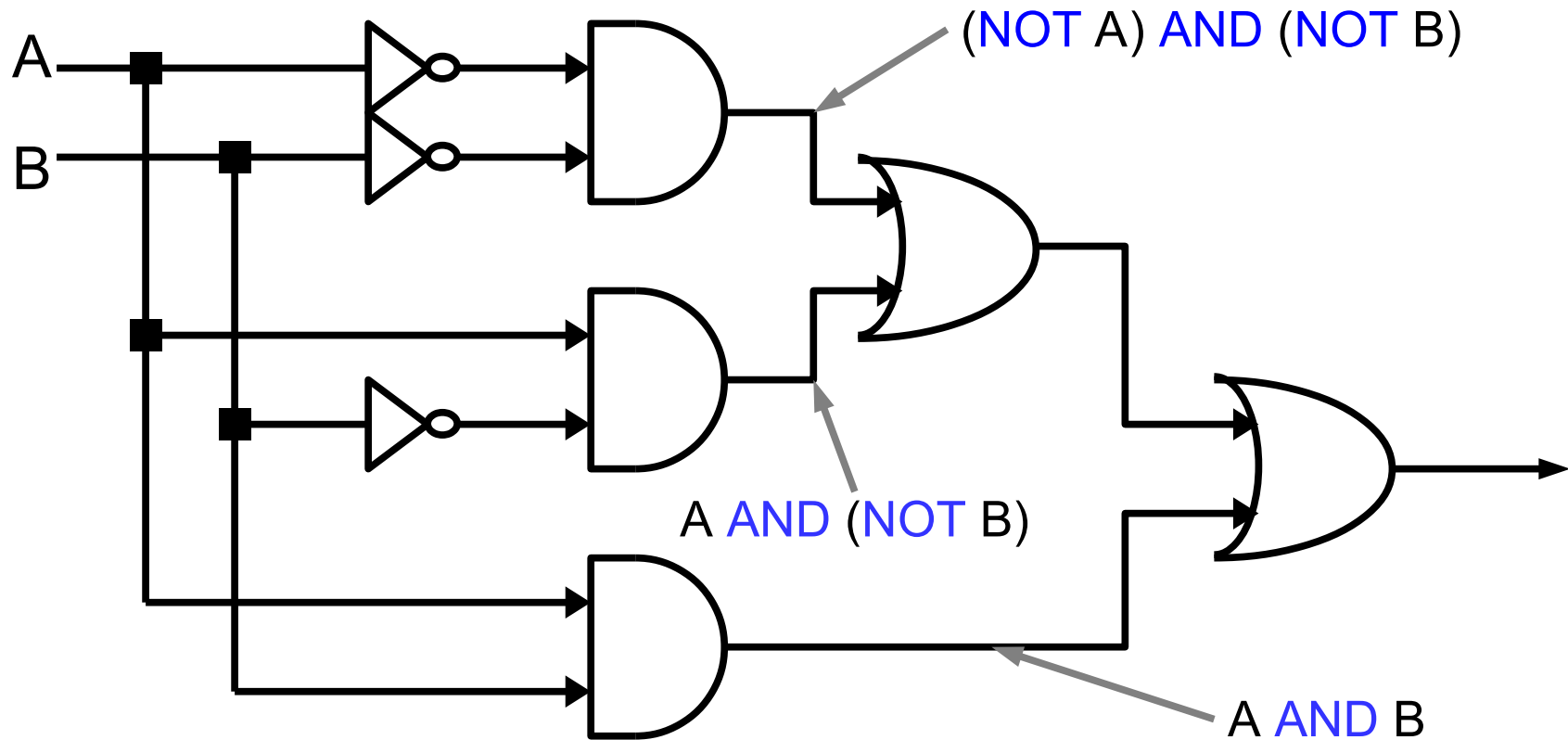
$(A \text{ AND } B)$

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

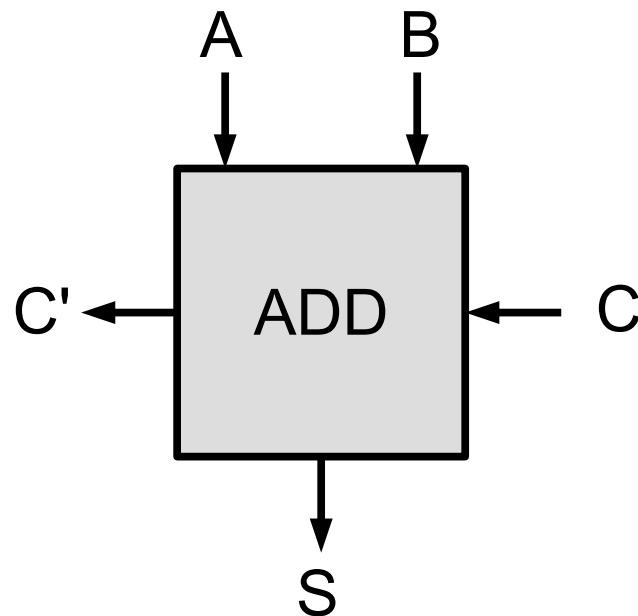
A	B	R
0	0	1
0	1	0
1	0	1
1	1	1

$((\text{NOT } A) \text{ AND } (\text{NOT } B)) \text{ OR}$
 $(A \text{ AND } (\text{NOT } B)) \text{ OR}$
 $(A \text{ AND } B)$



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

$S \equiv ((\text{NOT } A) \text{ AND } (\text{NOT } B) \text{ AND } C) \text{ OR}$
 $((\text{NOT } A) \text{ AND } B \text{ AND } (\text{NOT } C)) \text{ OR}$
 $(A \text{ AND } (\text{NOT } B) \text{ AND } (\text{NOT } C)) \text{ OR}$
 $(A \text{ AND } B \text{ AND } C)$

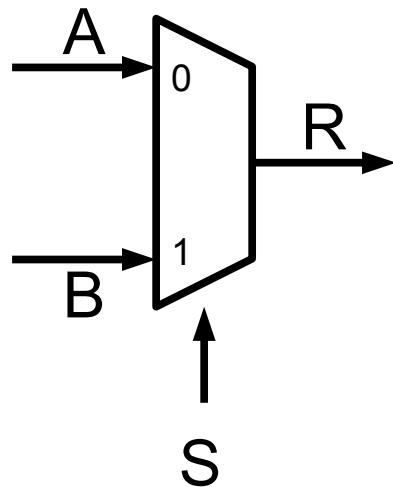
Calcolare S e C'

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

$$C' \equiv ((\text{NOT } A) \text{ AND } B \text{ AND } C) \text{ OR} \\ (A \text{ AND } (\text{NOT } B) \text{ AND } C) \text{ OR} \\ (A \text{ AND } B \text{ AND } (\text{NOT } C)) \text{ OR} \\ (A \text{ AND } B \text{ AND } C)$$

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 $S=1$.



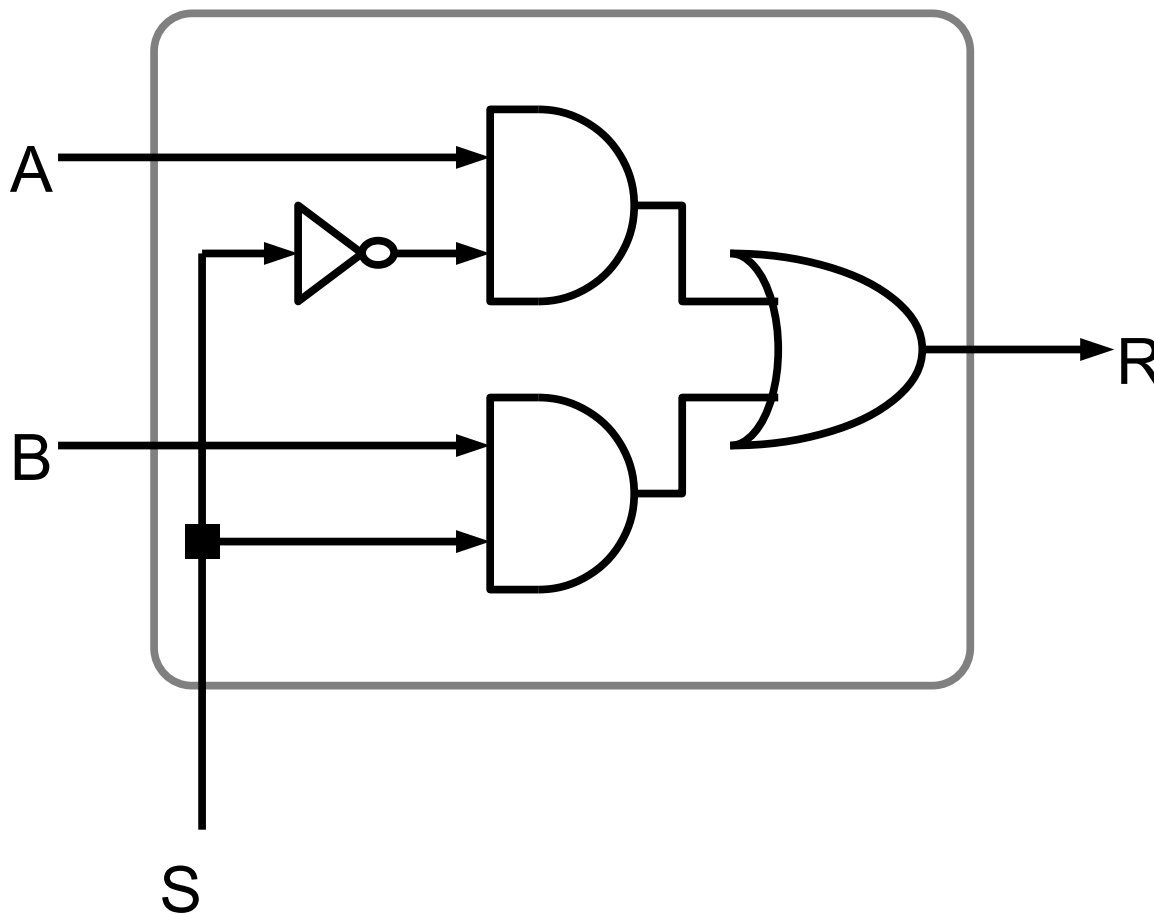
2-to-1 multiplexer

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

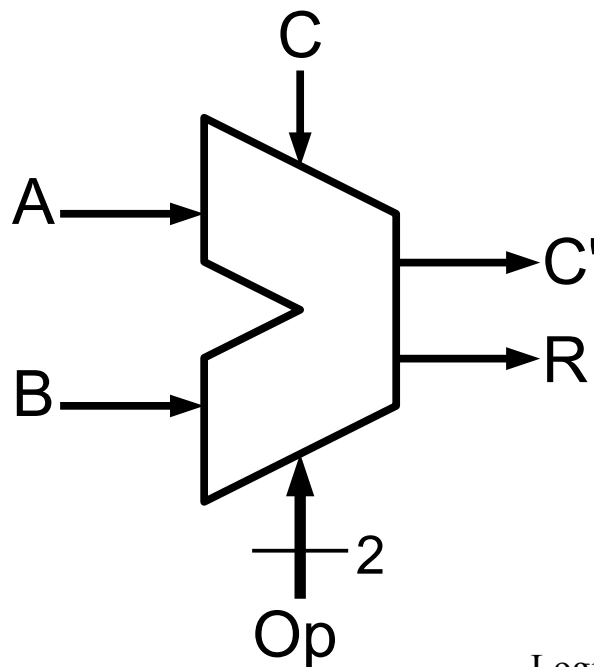


2-to-1 multiplexer

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

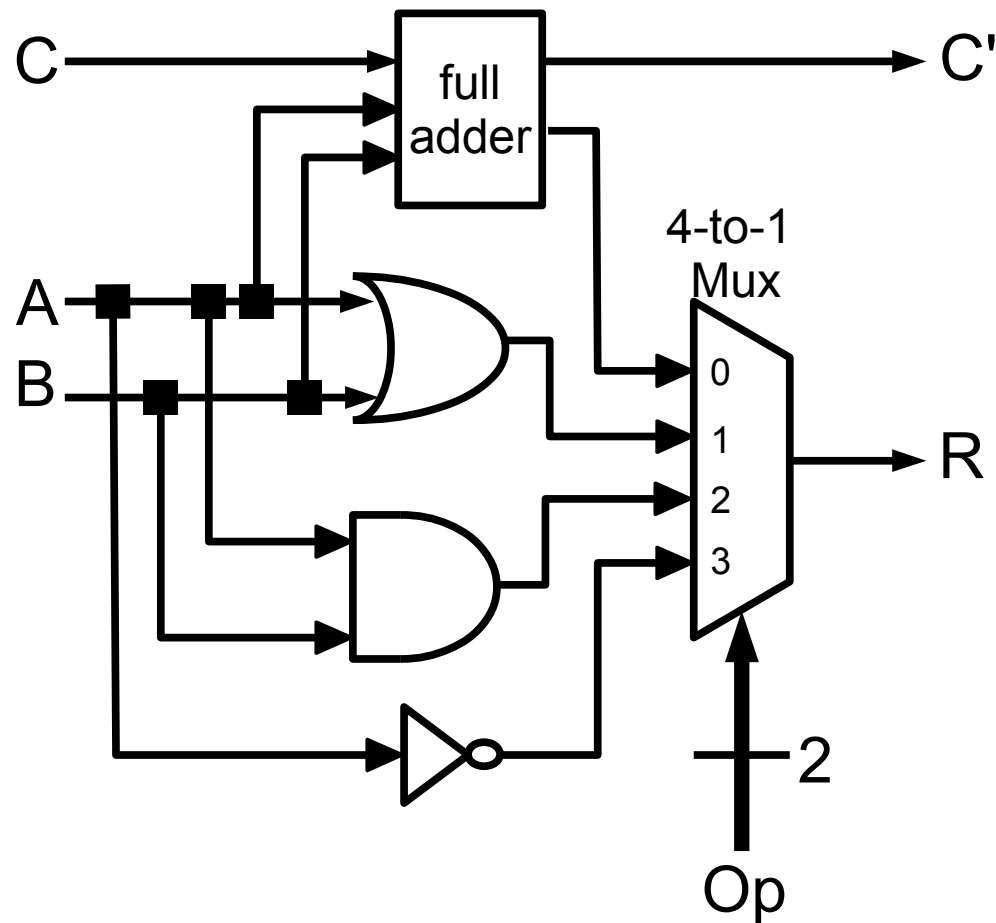


A	B	C	Op	R	C'
a	b	c	00	a+b+c	c'
a	b	x	01	a OR b	x
a	b	x	10	a AND b	x
a	b	x	11	NOT a	x

x = "don't care"

Costruiamo una ALU

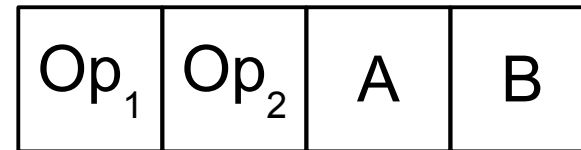
Usando un mux 4-a-1 possiamo selezionare l'operazione da compiere



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

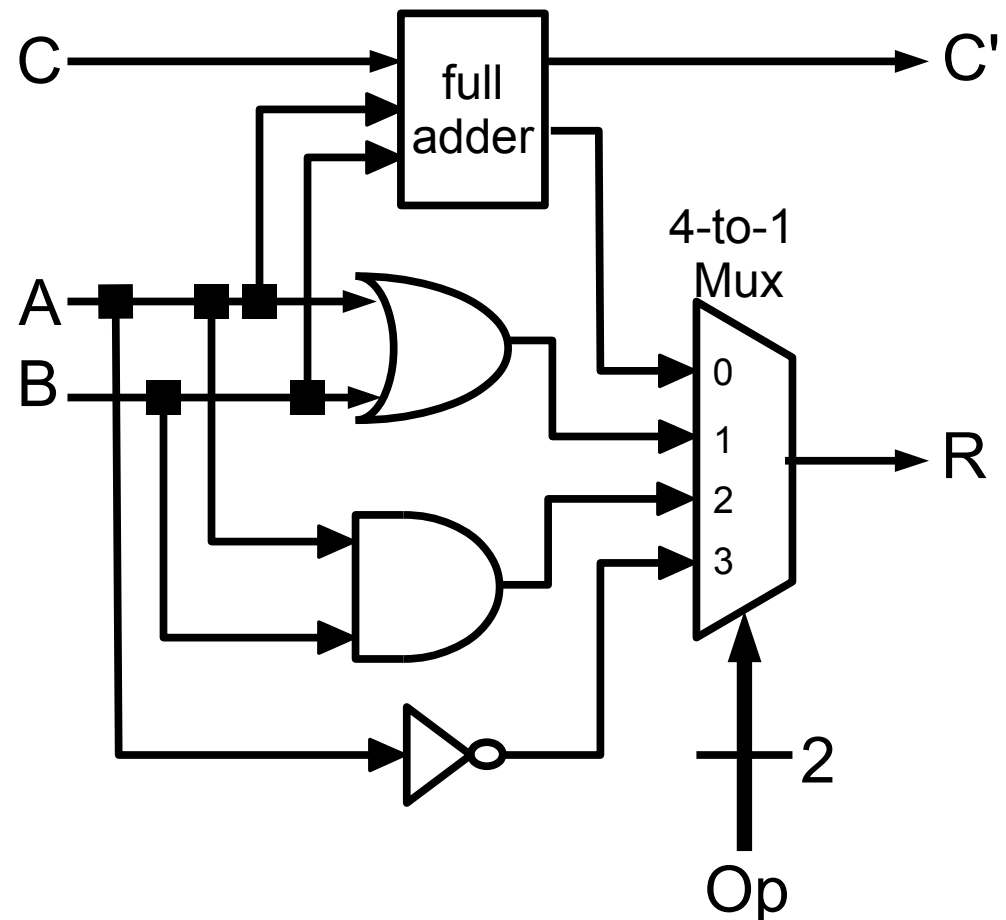
- 00 → ADD
- 01 → OR
- 10 → AND
- 11 → NOT

- Gli altri due indicano i valori di *A* e *B*

- Supponiamo che il valore di *C* provenga da altre vie

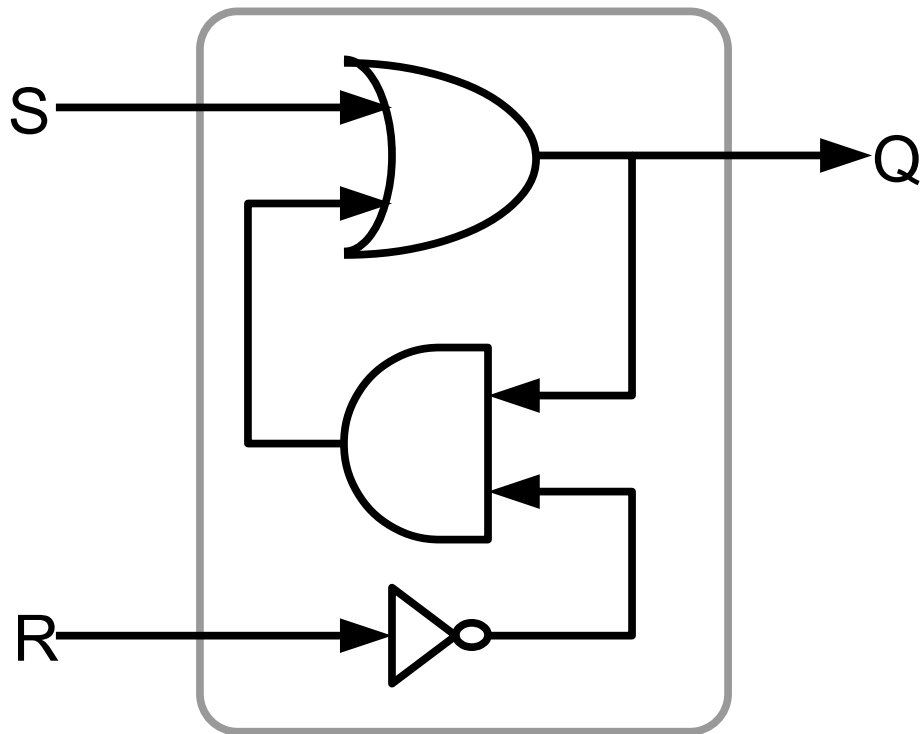
- Es:

- 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

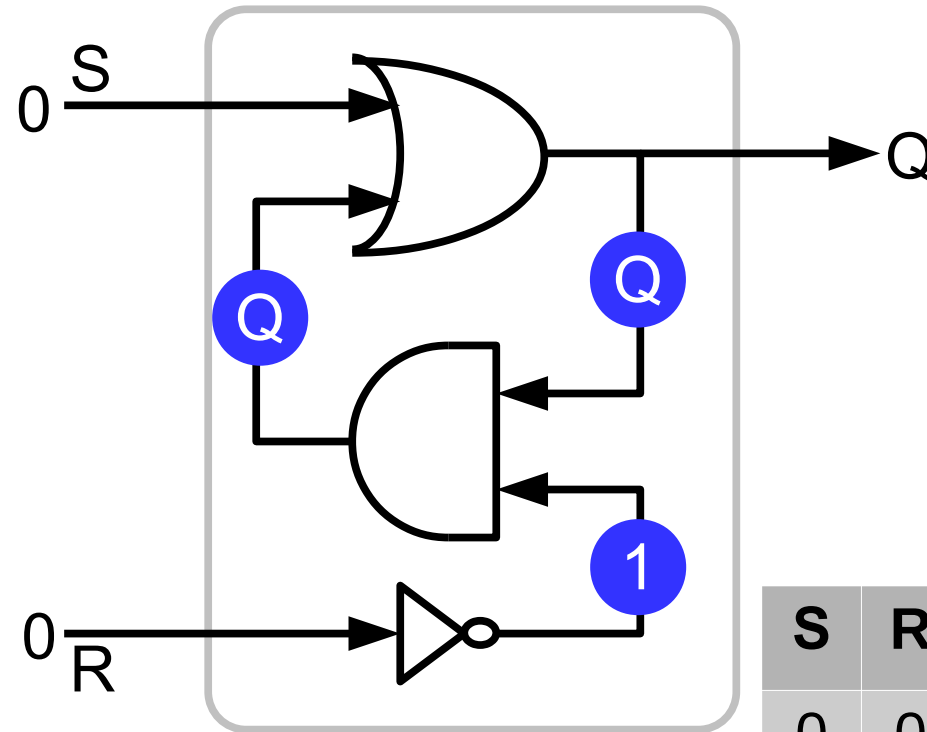
Ricordiamo:

$X \text{ AND } 0 = 0$

$X \text{ AND } 1 = X$

$X \text{ OR } 1 = 1$

$X \text{ OR } 0 = X$



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

Logica binaria

Flip-flop

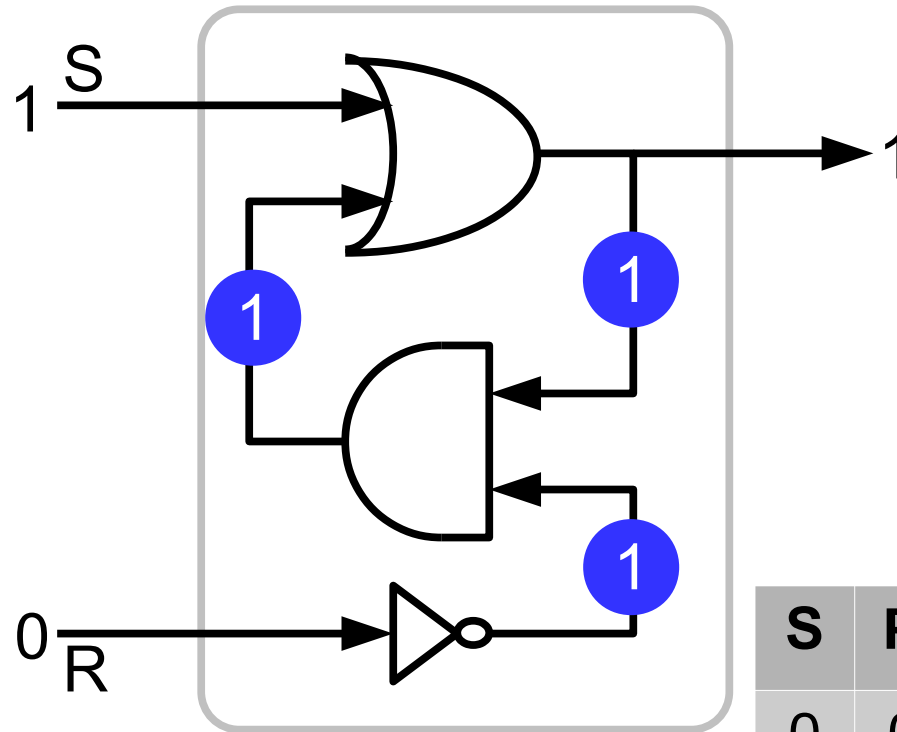
Ricordiamo:

$X \text{ AND } 0 = 0$

$X \text{ AND } 1 = X$

$X \text{ OR } 1 = 1$

$X \text{ OR } 0 = X$



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

Logica binaria

Flip-flop

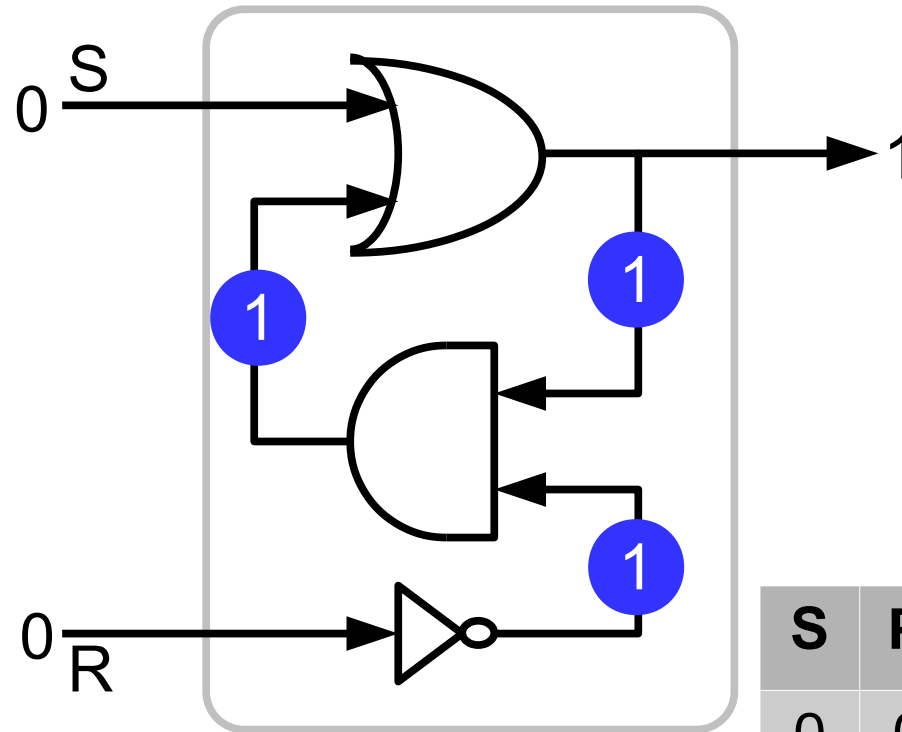
Ricordiamo:

$X \text{ AND } 0 = 0$

$X \text{ AND } 1 = X$

$X \text{ OR } 1 = 1$

$X \text{ OR } 0 = X$



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

Logica binaria

Flip-flop

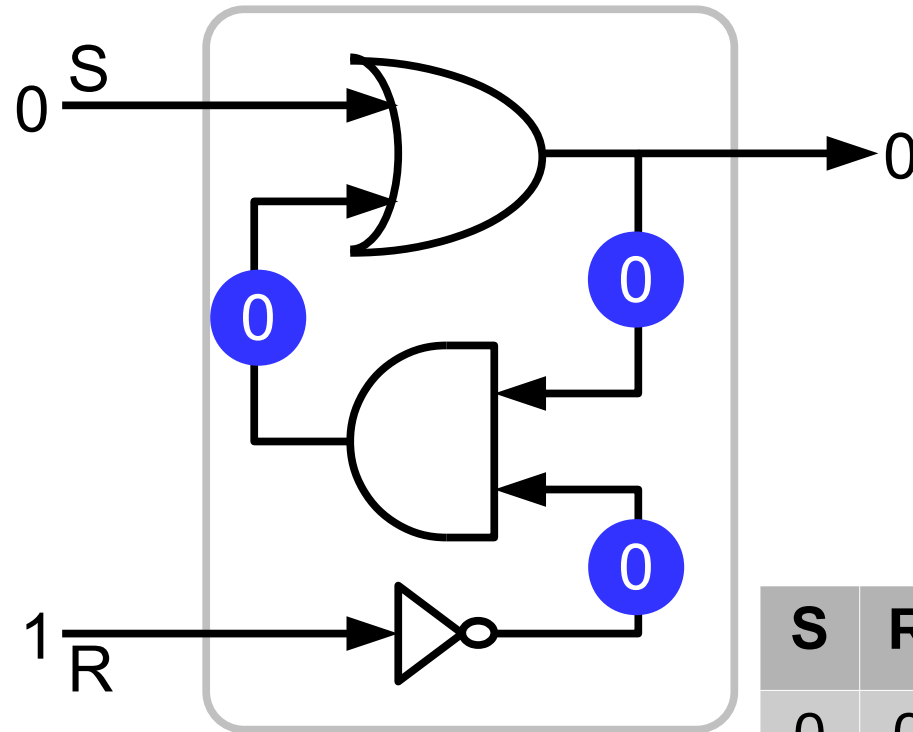
Ricordiamo:

$X \text{ AND } 0 = 0$

$X \text{ AND } 1 = X$

$X \text{ OR } 1 = 1$

$X \text{ OR } 0 = X$



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

Logica binaria

Flip-flop

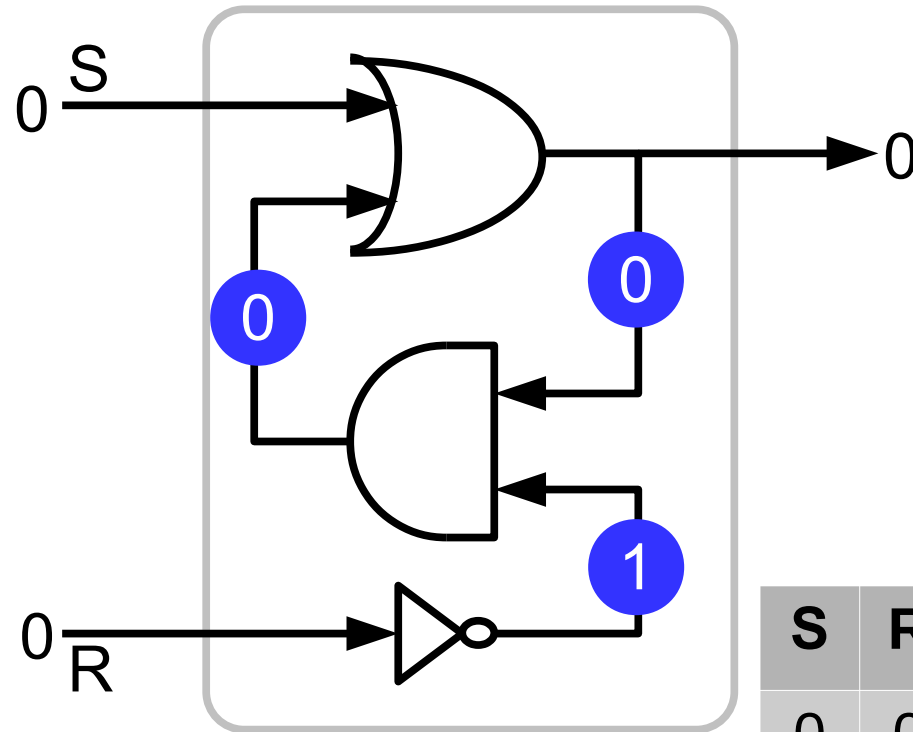
Ricordiamo:

$X \text{ AND } 0 = 0$

$X \text{ AND } 1 = X$

$X \text{ OR } 1 = 1$

$X \text{ OR } 0 = X$



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

Logica binaria

Esercizi

Per ciascuno dei circuiti seguenti, determinare la tabella di verità e realizzare il circuito corrispondente (non è richiesto di semplificare il circuito)

- [Parità, es. 2.5 disp.] Progettare un circuito con tre ingressi, A , B , C , e una singola uscita P . Il valore di P deve essere tale che la lista di quattro bit $ABCP$ deve avere un numero pari di cifre 1
 - Es: $ABC=110$, $P=0$; $ABC=111$, $P=1$; $ABC=010$, $P=1$
- [Maggioranza, es. 2.6 disp.] Progettare un circuito con tre ingressi, A , B , C , e una singola uscita M . M vale uno se e solo se la lista di tre bit ABC contiene due o tre cifre 1
 - Es: $ABC=011$, $M=1$; $ABC=001$, $M=0$; $ABC=111$, $M=1$

Idee chiave

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