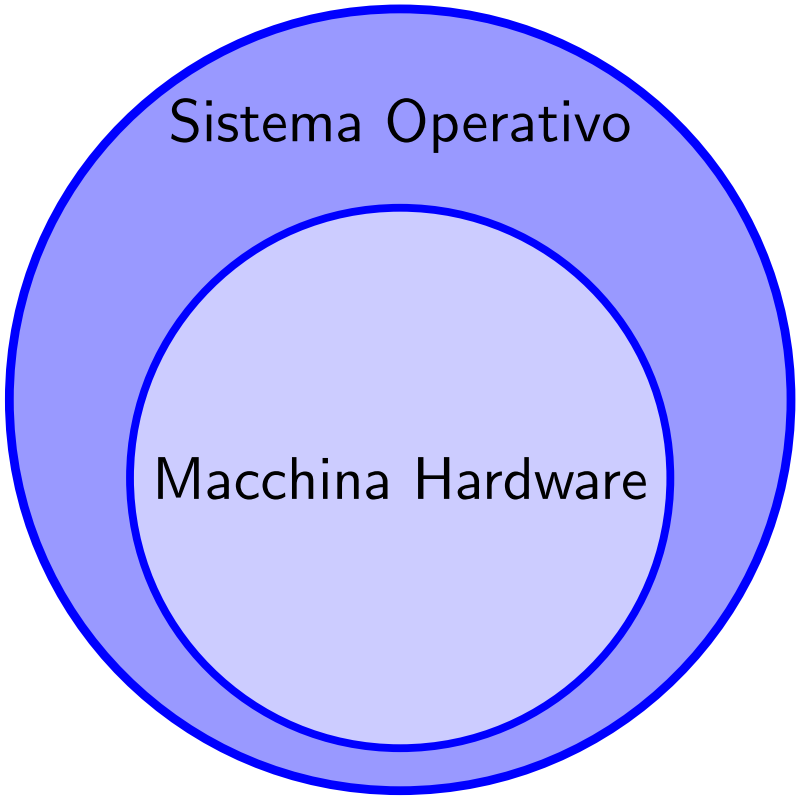
Informatica e Bioinformatica: Circuiti

Mauro Conti

Date TBD

Macchina Hardware/Software

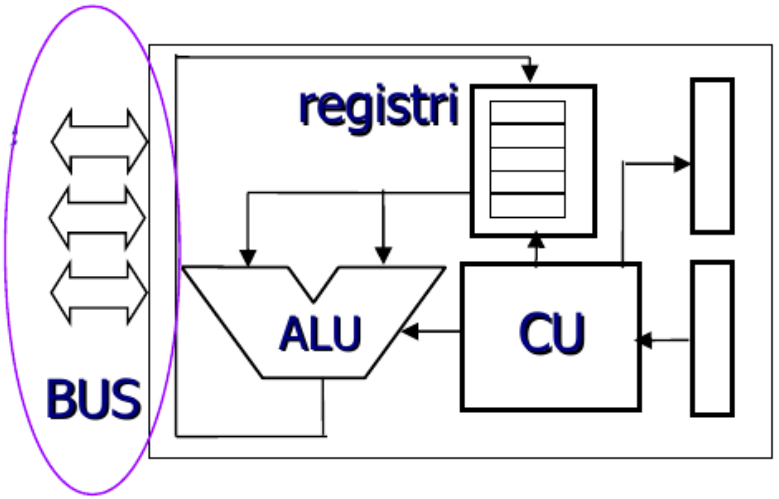


- La macchina hardware corrisponde alle componenti fisiche del calcolatore (quelle viste nella lezione precedente).
- Un sistema operativo è un programma che funge da interfaccia tra l'utente (e/o i programmi applicativi) e l'hardware del calcolatore.

CPU

La CPU (Central Processing Unit) è in grado di eseguire dei programmi, cioè sequenze di istruzioni elementari
È costituita da:

- Registri: piccole celle di memoria temporanea, servono per memorizzare gli operandi per le istruzioni di calcolo dell'ALU.
- Unita' di controllo (CU) che coordina le attività
- Un'unità aritmetico-logica (ALU) per l'elaborazione dati.



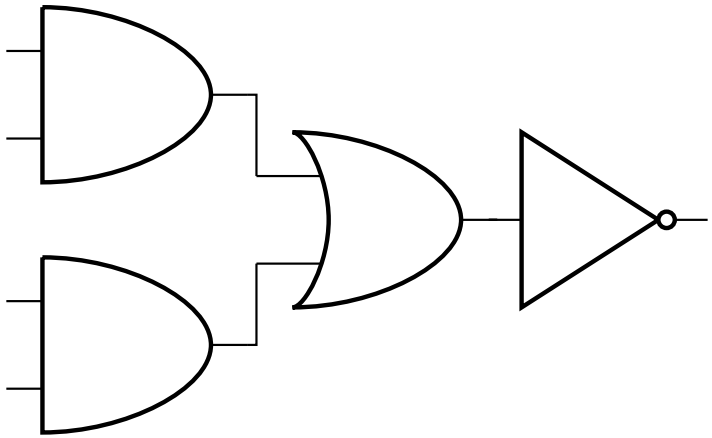
Filosofia di Costruzione:

"tante componenti semplici, se ben organizzate, possono realizzare funzionalità complesse"

Circuiti Elettronici

La ALU è composta da circuiti elettronici
Un circuito può essere rappresentato:

- mostrando i circuiti base (porte logiche) da cui è composto. Una porta logica è un circuito che opera su 1 o 2 segnali in entrata e ne produce 1 in uscita
- un circuito calcola una funzione che, per definizione, è rappresentata da tutte le possibili coppie input/output (vedi tabella a destra)



Input A	Input B	Output R
0	0	0
0	1	0
1	0	0
1	1	1

Algebra di Boole

- L'algebra di Boole opera su due valori di verità, VERO e FALSO, mutuamente esclusivi.
- Nell'algebra di Boole è possibile definire funzioni (che chiameremo operazioni logiche) che prendono come input valori di verità (VERO, FALSO) e calcolano come risultato un valore di verità (VERO, FALSO)
- Esempio: la tabella di verità (così viene chiamata) sotto rappresenta la proposizione: metto il cappotto solo se piove e fa freddo.

Input Piove	Input Fa Freddo	Output Metto Cappotto
FALSO	FALSO	FALSO
FALSO	VERO	FALSO
VERO	FALSO	FALSO
VERO	VERO	VERO

Da Boole ai Circuiti

- Prenderemo in considerazione 3 tipi di operazioni logiche/circuiti: AND, OR, NOT.
  - NOT prende 1 argomento in input, ad esempio A. NOT(A) è vera solamente se A=FALSO

Input A	NOT(A)
VERO	FALSO
FALSO	VERO

Da Boole ai Circuiti

- Prenderemo in considerazione 3 tipi di operazioni logiche/circuiti: AND, OR, NOT.
  - NOT prende 1 argomento in input, ad esempio A. NOT(A) è vera solamente se A=FALSO
  - AND prende 2 argomenti in input, ad esempio A,B. A AND B è vera solamente se A=VERO e B=VERO

Input A	Input B	A AND B
FALSO	FALSO	FALSO
FALSO	VERO	FALSO
VERO	FALSO	FALSO
VERO	VERO	VERO

Da Boole ai Circuiti

- Prenderemo in considerazione 3 tipi di operazioni logiche/circuiti: AND, OR, NOT.
  - NOT prende 1 argomento in input, ad esempio A. NOT(A) è vera solamente se A=FALSO
  - AND prende 2 argomenti in input, ad esempio A,B. A AND B è vera solamente se A=VERO e B=VERO
  - OR prende 2 argomenti in input, ad esempio A,B. A OR B è falsa solamente se A=FALSO e B=FALSO

Input A	Input B	A OR B
FALSO	FALSO	FALSO
FALSO	VERO	VERO
VERO	FALSO	VERO
VERO	VERO	VERO

Da Boole ai Circuiti

- Prenderemo in considerazione 3 tipi di operazioni logiche/circuiti: AND, OR, NOT.
  - NOT prende 1 argomento in input, ad esempio A. NOT(A) è vera solamente se A=FALSO
  - AND prende 2 argomenti in input, ad esempio A,B. A AND B è vera solamente se A=VERO e B=VERO
  - OR prende 2 argomenti in input, ad esempio A,B. A OR B è falsa solamente se A=FALSO e B=FALSO

Input A	Input B	A OR B
FALSO	FALSO	FALSO
FALSO	VERO	VERO
VERO	FALSO	VERO
VERO	VERO	VERO

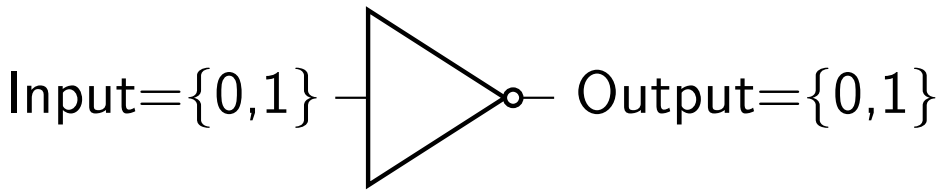
- Se associamo FALSO=0, VERO=1, possiamo applicare i risultati dell'algebra di Boole e costruire funzioni che operino su bit!

NOT

- L'operazione NOT prende in input esattamente un valore di verità e restituisce la sua negazione

Input A	Output NOT(A)
1	0
0	1

- Un circuito corrispondente alla tabella di verità sopra è il seguente:

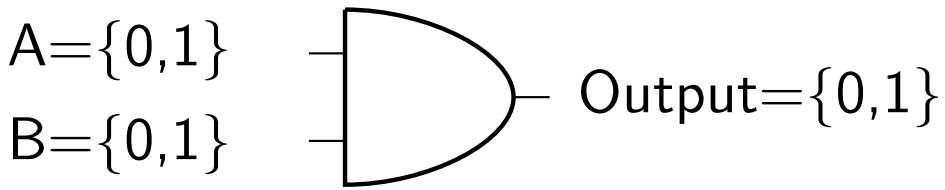


AND

- L'operazione AND prende come input esattamente due valori $\{0,1\}$ e restituisce 1 solamente se entrambi sono 1.

Input A	Input B	Output A AND B
0	0	0
0	1	0
1	0	0
1	1	1

- Un circuito corrispondente alla tabella di verità sopra è il seguente:

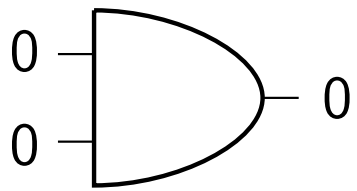


AND

- L'operazione AND prende come input esattamente due valori $\{0,1\}$ e restituisce 1 solamente se entrambi sono 1.

Input A	Input B	Output A AND B
0	0	0
0	1	0
1	0	0
1	1	1

- Un circuito corrispondente alla tabella di verità sopra è il seguente:

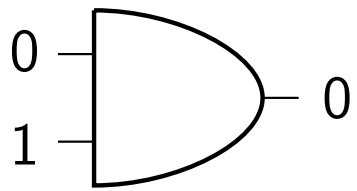


AND

- L'operazione AND prende come input esattamente due valori $\{0,1\}$ e restituisce 1 solamente se entrambi sono 1.

Input A	Input B	Output A AND B
0	0	0
0	1	0
1	0	0
1	1	1

- Un circuito corrispondente alla tabella di verità sopra è il seguente:

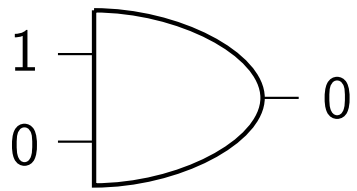


AND

- L'operazione AND prende come input esattamente due valori $\{0,1\}$ e restituisce 1 solamente se entrambi sono 1.

Input A	Input B	Output A AND B
0	0	0
0	1	0
1	0	0
1	1	1

- Un circuito corrispondente alla tabella di verità sopra è il seguente:

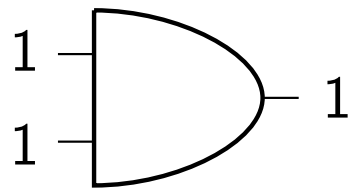


AND

- L'operazione AND prende come input esattamente due valori $\{0,1\}$ e restituisce 1 solamente se entrambi sono 1.

Input A	Input B	Output A AND B
0	0	0
0	1	0
1	0	0
1	1	1

- Un circuito corrispondente alla tabella di verità sopra è il seguente:



OR

- L'operazione OR prende come input esattamente due valori $\{0,1\}$ e restituisce 0 solamente se entrambi sono 0.

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

- Un circuito corrispondente alla tabella di verità sopra è il seguente:

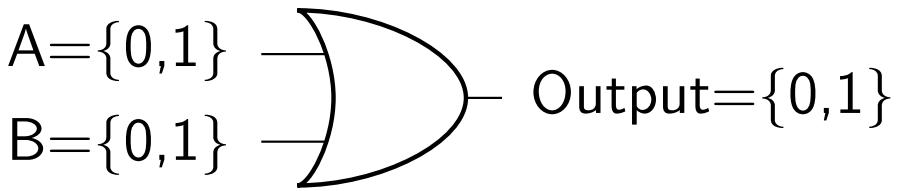
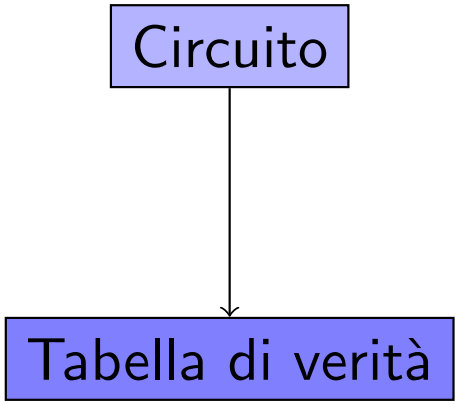


Tabella di verità $\leftrightarrow$ Circuito

Dato un circuito, è possibile definire un'unica tabella di verità corrispondente (con gli stessi input/output)



Data una tabella di verità, si possono costruire più circuiti che la realizzano (con gli stessi input/output)

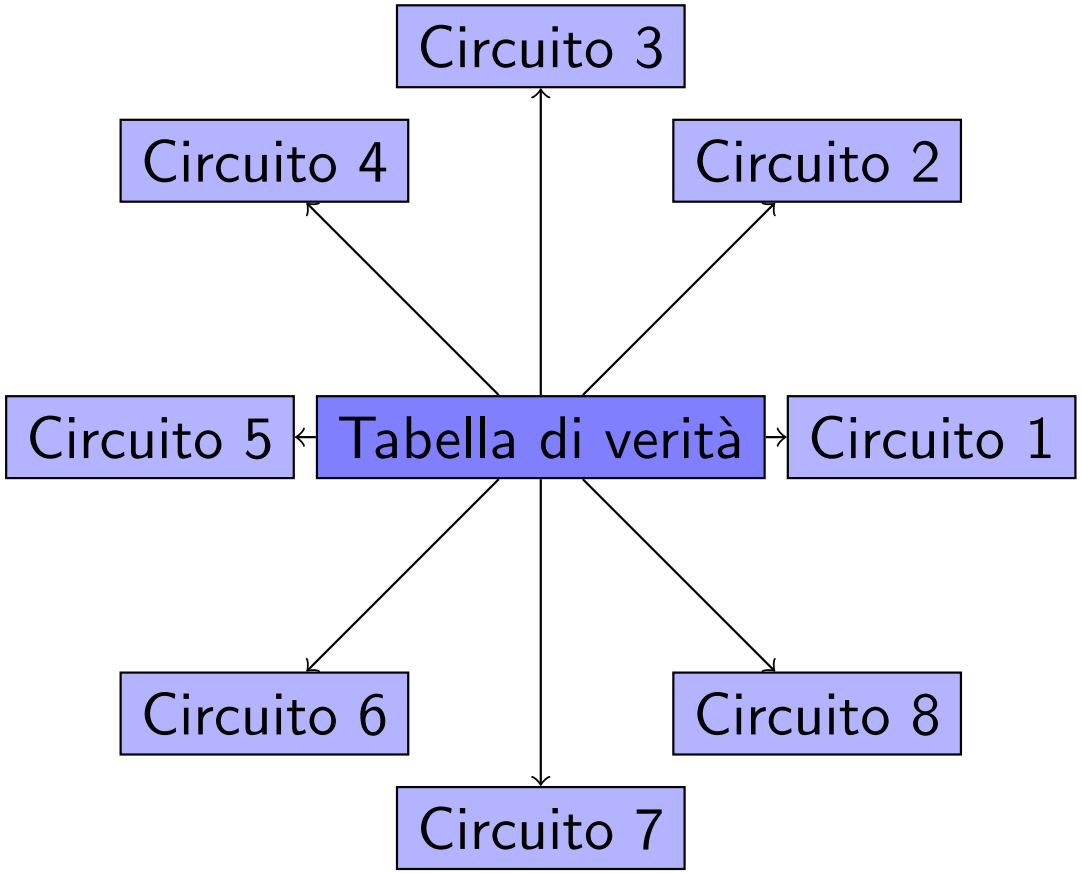
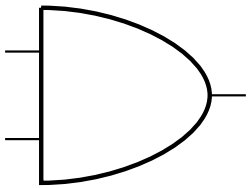


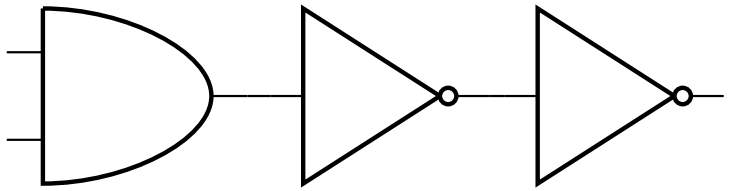
Tabella di verità $\leftrightarrow$ Circuito

- È possibile modificare un circuito ed ottenere una tabella di verità equivalente:

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



Equivale, per esempio, a



- Esistono circuiti che hanno componenti diversi ma sono equivalenti (cacolano la stessa funzione)
- Ogni modifica alla tabella di verità definisce una nuova funzione, per cui non esistono tabelle di verità diverse che siano equivalenti

Tabella di verità

- Quante tabelle di verità/funzioni con due input possono essere create?
- Cambiando un valore nella casella di output si definisce una nuova funzione
- Il numero di righe, e le prime due colonne, sono determinate da tutte possibili configurazioni dei valori di input

A	B	R1	R2	R3	R4	R5	R6	...	R16
0	0	0	1	0	0	0	1	...	1
0	1	0	0	1	0	0	1	...	1
1	0	0	0	0	1	0	0	...	1
1	1	0	0	0	0	1	0	...	1

- Quante sono le possibili configurazioni della colonna di output R? $2^4 = 16$ (2 rappresenta il numero di valori che un bit può assumere, 4 il numero di righe)

Esempi di funzioni logiche: Implicazione

- L'operazione di implicazione logica si denota con il simbolo $\Rightarrow$
- Corrisponde a proposizioni del tipo "se A allora B (se piove allora prendo l'ombrello)"

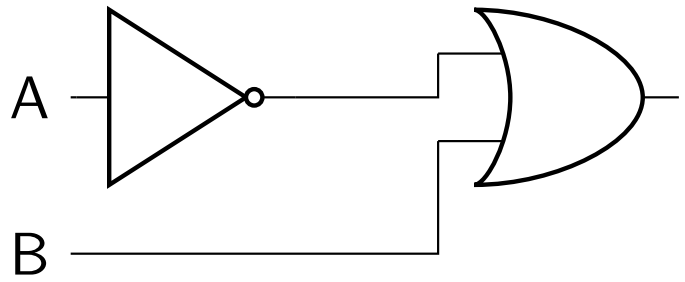
A	B	A $\Rightarrow$ B
0	0	1
0	1	1
1	0	0
1	1	1

- A $\Rightarrow$ B equivale a NOT(A) OR B (per mostrarlo basta verificare che le due tabelle di verità siano identiche):

A	B	NOT(A)	NOT(A) OR B
0	0	1	1
0	1	1	1
1	0	0	0
1	1	0	1

Esempi di funzioni logiche: Implicazione

- Poichè abbiamo riformulato $\Rightarrow$ in termini di AND,OR,NOT possiamo costruire un circuito equivalente alla funzione $\Rightarrow$:



Esempi di funzioni logiche: Equivalenza

- L'operazione di equivalenza logica si denota con il simbolo $\equiv$
- Corrisponde a proposizioni del tipo "A se e solo se B (solo se piove prendo l'ombrello)"

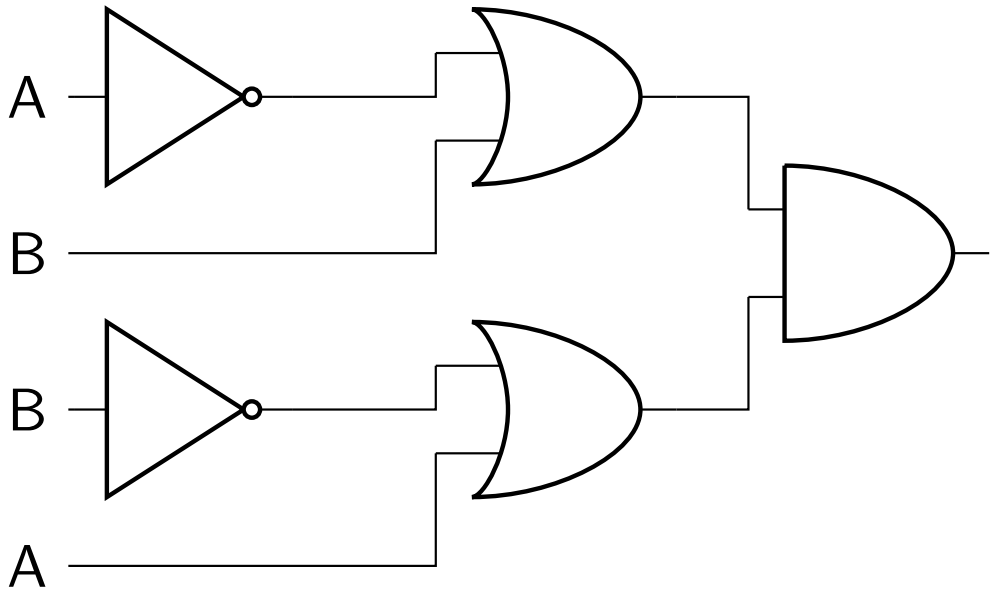
A	B	$A \equiv B$
0	0	1
0	1	0
1	0	0
1	1	1

- $A \equiv B$ equivale a $(A \Rightarrow B)$ AND $(B \Rightarrow A)$

A	B	$A \Rightarrow B$	$B \Rightarrow A$	$(A \Rightarrow B)$ AND $(B \Rightarrow A)$
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	1	1	1

Esempi di funzioni logiche: Equivalenza

- $A \equiv B \Leftrightarrow (A \Rightarrow B)$ AND $(B \Rightarrow A) \Leftrightarrow$
  $\Leftrightarrow$ (NOT(A) OR B) AND (NOT(B) OR A)
- Poichè abbiamo riformulato $\equiv$ in termini di AND,OR,NOT possiamo costruire un circuito equivalente alla funzione $\equiv$:

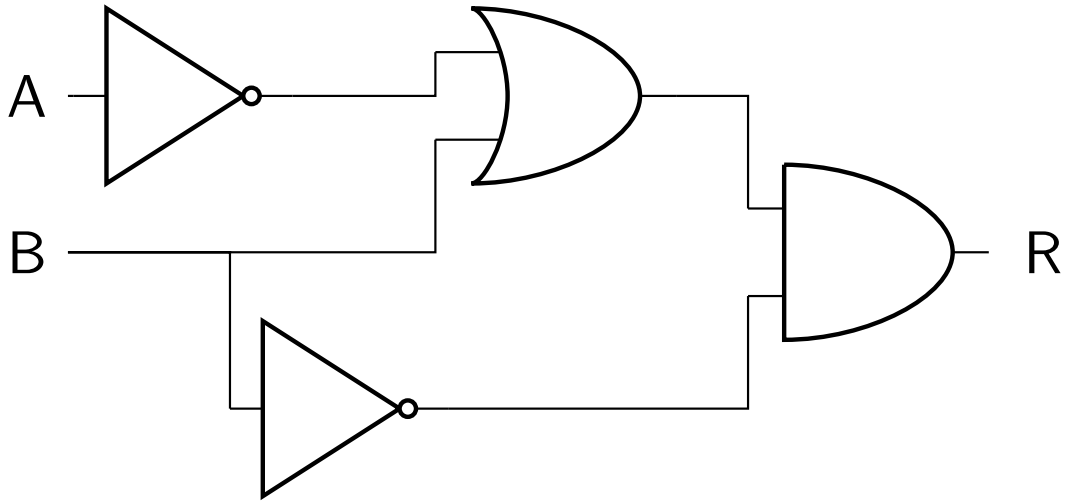


Da Circuito a Tabella di verità

- Notate che una formula logica, per esempio (NOT(A) OR B) AND (NOT(B) OR A), è un modo non grafico per scrivere un circuito. Le parole formula logica e circuito sono quasi "sinonimi", nel senso che ad una formula logica corrisponde un unico circuito e viceversa. Ricordate invece che ad una tabella di verità corrispondono più circuiti.
- Dato un circuito, è possibile ricostruire la tabella di verità corrispondente
  - calcolando, per ogni possibile configurazione degli ingressi, l'uscita delle porte fino alle uscite del circuito
  - costruendo le tabelle di verità partendo dalle formule intermedie più semplici fino alla formula data

Da Circuito a Tabella di verità: esempio

Qual è la tavola di verita del circuito seguente?

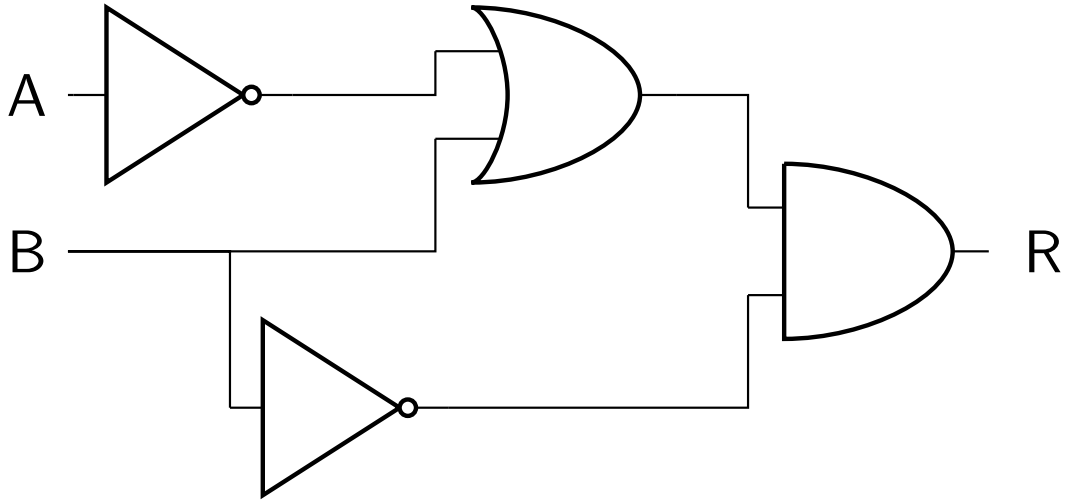


Metodo 1 (calcolando, per ogni possibile configurazione degli ingressi, l'uscita delle porte fino alle uscite del circuito):

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

Da Circuito a Tabella di verità: esempio

Qual è la tavola di verita del circuito seguente?



Metodo 2 (costruendo le tabelle di verità partendo dalle formule intermedie più semplici fino alla formula data):

A	B	NOT(A)	NOT(A) OR B	NOT(B)	(NOT(A) OR B) AND NOT(B)
0	0	1	1	1	1
0	1	1	1	0	0
1	0	0	0	1	0
1	1	0	1	0	0

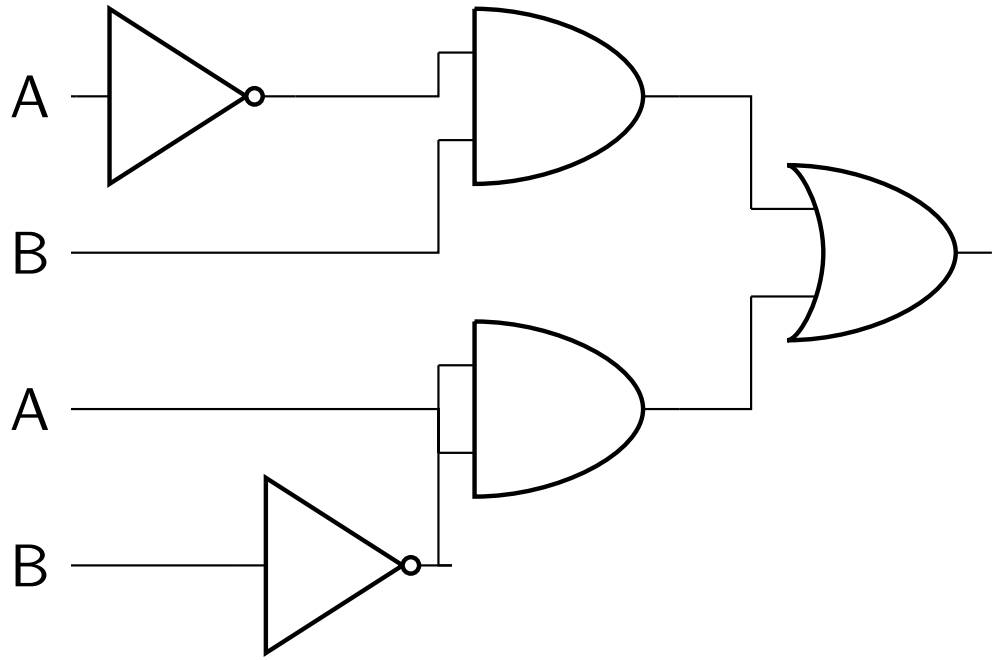
Completezza delle operazioni AND, OR, NOT

- Perchè abbiamo scelto come operazioni base proprio AND, OR, NOT?
- Perchè ci permettono di rappresentare tutte le funzioni binarie!
- Per mostrarlo definiamo un algoritmo per convertire una tabella di verità in un circuito

Da Tabella di verità a Circuito

1. Si considerano tanti input quante sono le colonne di input della tabella (A,B)
2. Un solo output (R).
3. si crea un circuito AND per ogni output uguale a 1 (cioè per le righe 2 e 3).
4. Input degli AND: collegato direttamente se 1, negato se 0
5. tutti i circuiti AND (se più di 2) vengono collegati ad uno o più circuiti OR

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



NAND e NOR

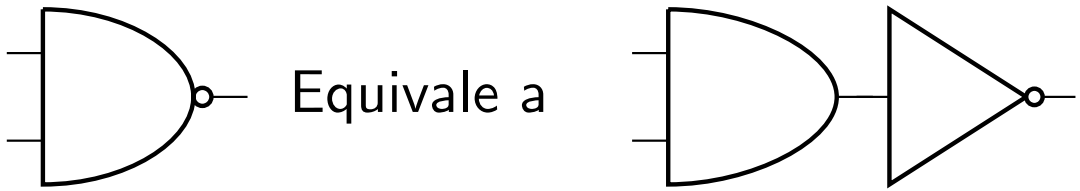
- Notate che l'algoritmo sopra non produce necessariamente il circuito con il minor numero di porte logiche
- Permette però di rappresentare tutte le funzioni binarie e, come vedremo, tutte le funzioni che possono essere definite su un calcolatore
- In realtà è possibile mostrare che ciascuna delle seguenti operazioni
  - NAND (che corrisponde a NOT(AND))
  - NOR (che corrisponde a NOT(OR))

  sono da sole in grado di rappresentare tutte le funzioni binarie

NAND e NOR

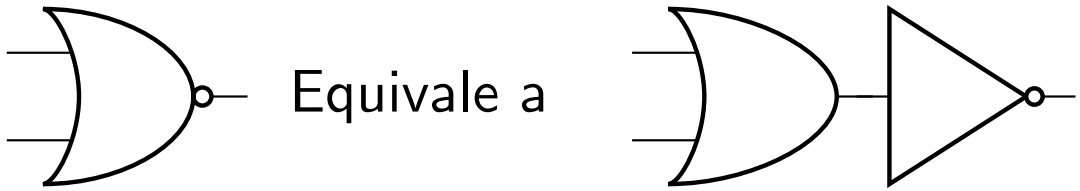
- NAND (da sinistra: tabella di verità, simbolo porta logica, corrispondente circuito con porte AND, NOT)

A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0



- NOR (da sinistra: tabella di verità, simbolo porta logica, corrispondente circuito con porte OR, NOT)

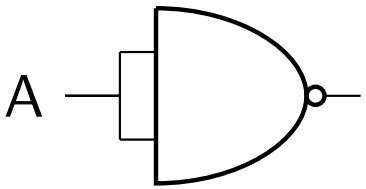
A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0



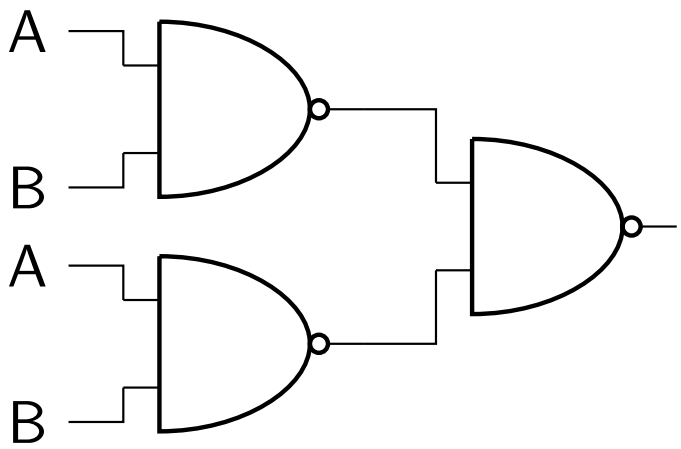
Completezza di NAND

- Per mostrare che il circuito NAND riesce a rappresentare qualsiasi tabella di verità, è sufficiente mostrare che NAND riesce ad esprimere AND, OR, NOT (poichè abbiamo già mostrato che AND, OR, NOT possono rappresentare ogni operazione logica)

NOT(A) $\Leftrightarrow$ A NAND A



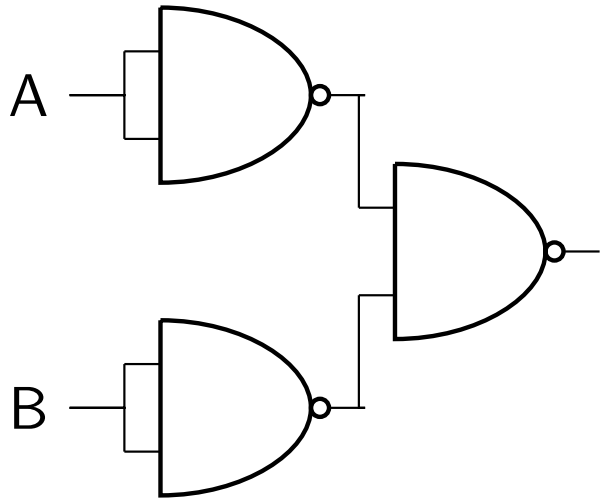
A AND B $\Leftrightarrow$ (A NAND B) NAND (A NAND B)



Completezza di NAND

- Per mostrare che il circuito NAND riesce a rappresentare qualsiasi tabella di verità, è sufficiente mostrare che NAND riesce ad esprimere AND, OR, NOT (poichè abbiamo già mostrato che AND, OR, NOT possono rappresentare ogni operazione logica)

$$A \text{ OR } B \Leftrightarrow (A \text{ NAND } A) \text{ NAND } (B \text{ NAND } B)$$



Tipologie di Esercizi

- Mostrare che due formule logiche (o due circuiti) sono equivalenti
- Dato un circuito determinare una formula logica corrispondente
- Dato un circuito/formula logica determinare la tabella di verità corrispondente
- Data una tabella di verità, costruire un circuito corrispondente

Esempi di Esercizi

- "Mostrare che le seguenti funzioni sono equivalenti":
  NOT(A AND B) $\Leftrightarrow$ NOT(A) OR NOT(B)

Poichè sappiamo che una tabella di verità identifica univocamente una funzione, è sufficiente mostrare che le due formule logiche hanno la medesima tabella di verità.

A	B	A AND B	NOT(A AND B)
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

A	B	NOT(A)	NOT(B)	NOT(A) OR NOT(B)
0	0	1	1	1
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0

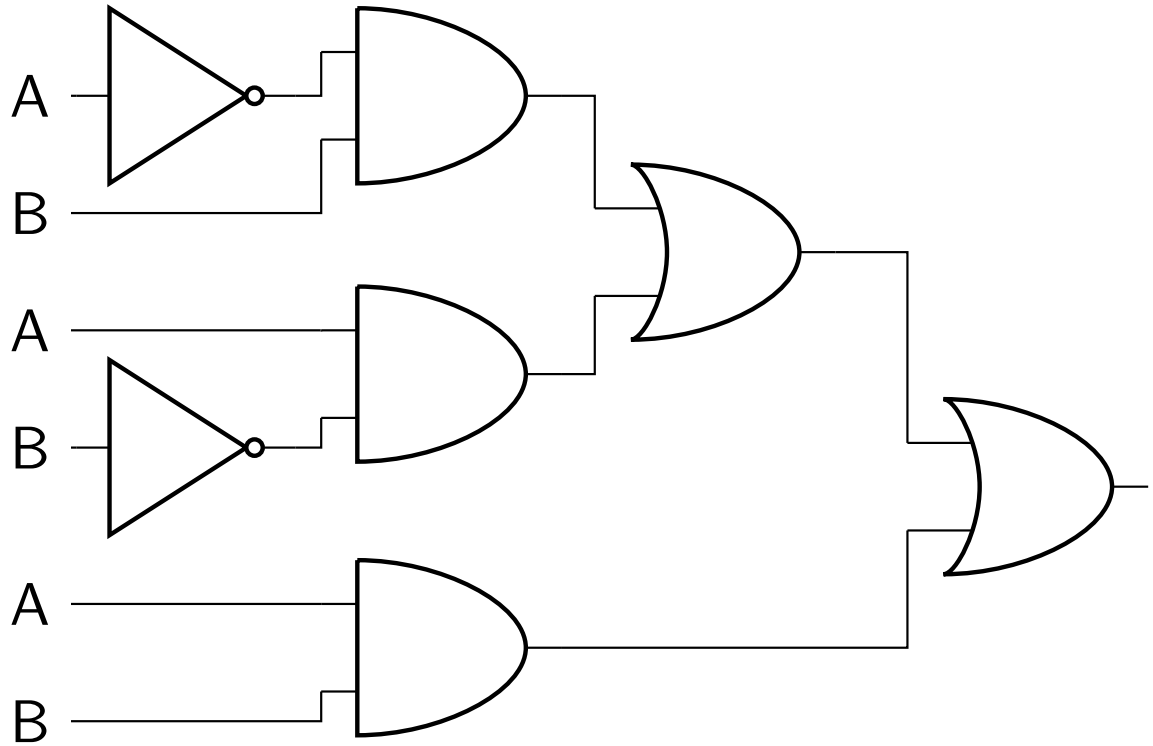
Formule equivalenti

- Notate che se due formule sono equivalenti, come NOT(A AND B) e NOT(A) OR NOT(B), posso sostituire l'una con l'altra, anche all'interno di formule più complesse
- Ad esempio, scrivendo NOT(A AND B) al posto di NOT(A) OR NOT(B), risparmio tempo perchè dovrei calcolare una colonna in meno per ottenere l'output (vedi slide precedente, dove NOT(A AND B) richiede due colonne mentre NOT(A) OR NOT(B) tre)
- Altre formule equivalenti sono:
  - NOT(NOT(A)) $\Leftrightarrow$ A
  - A AND B $\Leftrightarrow$ B AND A
  - A OR B $\Leftrightarrow$ B OR A
  - A AND (B AND C) $\Leftrightarrow$ (A AND B) AND C $\Leftrightarrow$ $\Leftrightarrow$ A AND B AND C
  - A OR (B OR C) $\Leftrightarrow$ (A OR B) OR C $\Leftrightarrow$ A OR B OR C
  - NOT(A OR B) $\Leftrightarrow$ NOT(A) AND NOT(B)

Esempi di Esercizi

- "Data la seguente tabella di verità, costruire un circuito corrispondente":

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



Esempio di Utilizzo dei Circuiti: Somma tra numeri binari

Rappresentazione dei Numeri: Sistema Decimale

- Noi comunemente utilizziamo il sistema decimale per rappresentare i numeri: ogni cifra di un numero può avere 10 valori diversi (da 0 a 9). Poichè ogni cifra può assumere 10 valori diversi, si dice che il numero è espresso in base 10.
- Una sequenza di cifre forma un numero secondo la seguente convenzione: $374 = 3 \cdot 10^2 + 7 \cdot 10^1 + 4 \cdot 10^0$
- Volendo essere formali: ogni cifra viene moltiplicata per la base elevata a k, dove k è la posizione della cifra contando da destra a partire da 0
- Se la base è maggiore di 10 si introducono delle lettere per le cifre rimanenti: ad esempio una base utilizzata in informatica è la base 16, le sue cifre sono:
  $0, 1, 2, 3, 4, 5, 6, 7, 8, 9$,A,B,C,D,E,F.

Rappresentazione dei Numeri: Sistema Binario

- Come abbiamo accennato, il calcolatore utilizza il bit per rappresentare l'informazione. Il bit può assumere 2 valori: 0 o 1. In questo caso si dice che un numero è espresso in base 2 (oppure in binario)
- Per determinare il valore di un numero binario positivo, si utilizza lo stesso algoritmo della slide precedente dove però la base è 2: *"ogni cifra viene moltiplicata per la base elevata a k, dove k è la posizione della cifra contando da destra a partire da* 0*"*

$$1101 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 8 + 4 + 1 = 13$$

Numeri Binari Positivi

- Il numero di configurazioni diverse di n bit è 2^n, per cui si riescono a rappresentare 2^n numeri diversi.
- Il numero più grande rappresentabile con n bit è $2^n - 1$ (perchè si inizia a contare da 0).

binario	decimale
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Somma Binaria

- Si effettua colonna per colonna, da destra a sinistra (come le somme che ci hanno insegnato alle elementari)
- Riporto: quando la somma su una colonna supera la base
- Somma tra bit:

$$\begin{array}{rrrrrrrrrrr} & 0 & + & 0 & + & 1 & + & 1 & + \\ & 0 & & 1 & & 0 & & 1 & \\ \hline & 0 & & 1 & & 1 & & 10 & \end{array}$$

- Esempio di somma tra bit considerando il riporto (prima riga):

$$\begin{array}{rrrrrrr} 0 & 1 & 1 & 1 & 0 & & \\ & 0 & 1 & 0 & 1 & 0 & + \\ & 0 & 0 & 1 & 1 & 1 & \\ \hline & 1 & 0 & 0 & 0 & 1 & \end{array}$$

Somma di bit con Riporto

$$\begin{array}{cccccc} {\scriptstyle 0} & {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 0} & \\ & 0 & 1 & 0 & 1 & 0 \quad + \\ & 0 & 0 & 1 & 1 & 1 \\ - & - & - & - & - & - \\ & 1 & 0 & 0 & 0 & 1 \end{array}$$

Progettiamo un circuito che faccia la somma su una singola colonna.
Abbiamo tre cifre binarie X, Y, R (riporto) in input mentre in output vogliamo ottenere la somma S ed il riporto R'

Somma di bit con Riporto

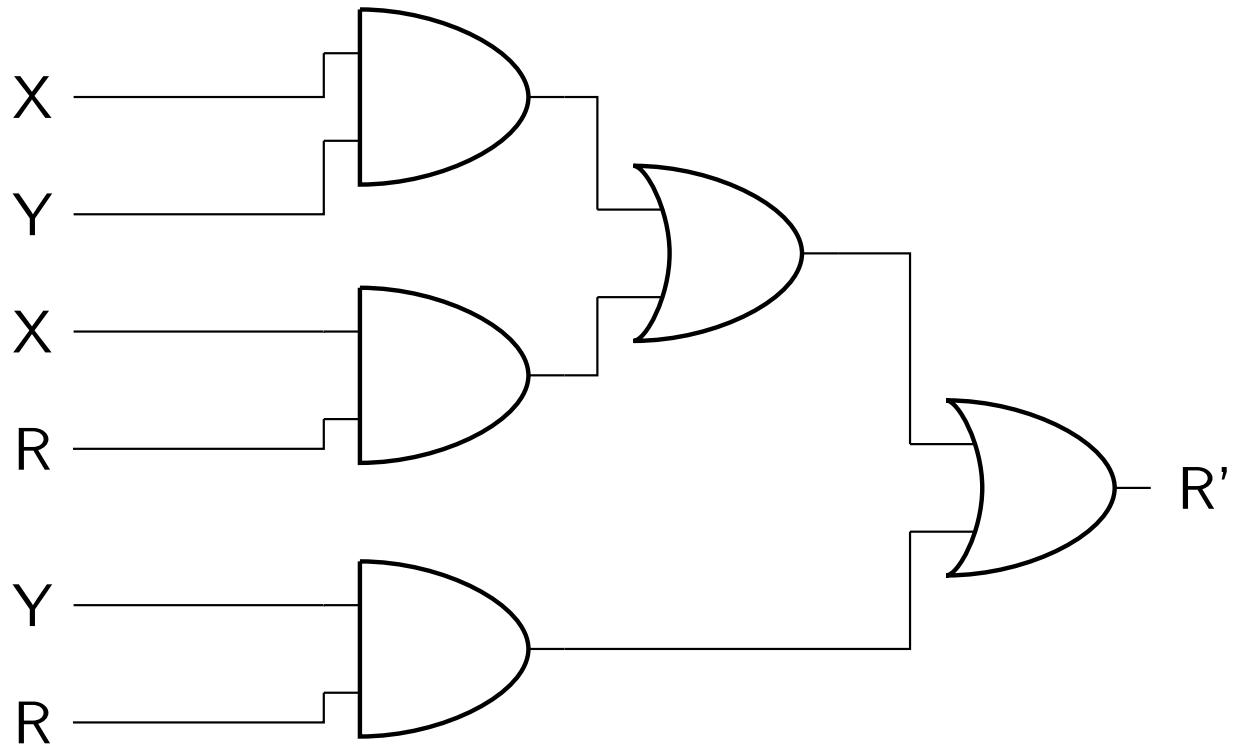
La somma di bit con riporto è rappresentata dalla seguente tabella di verità:

X	Y	R	S	R'
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
1	0	0	1	0
0	1	1	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

S è il risultato dell somma, R' il riporto prodotto

Il Circuito Riporto

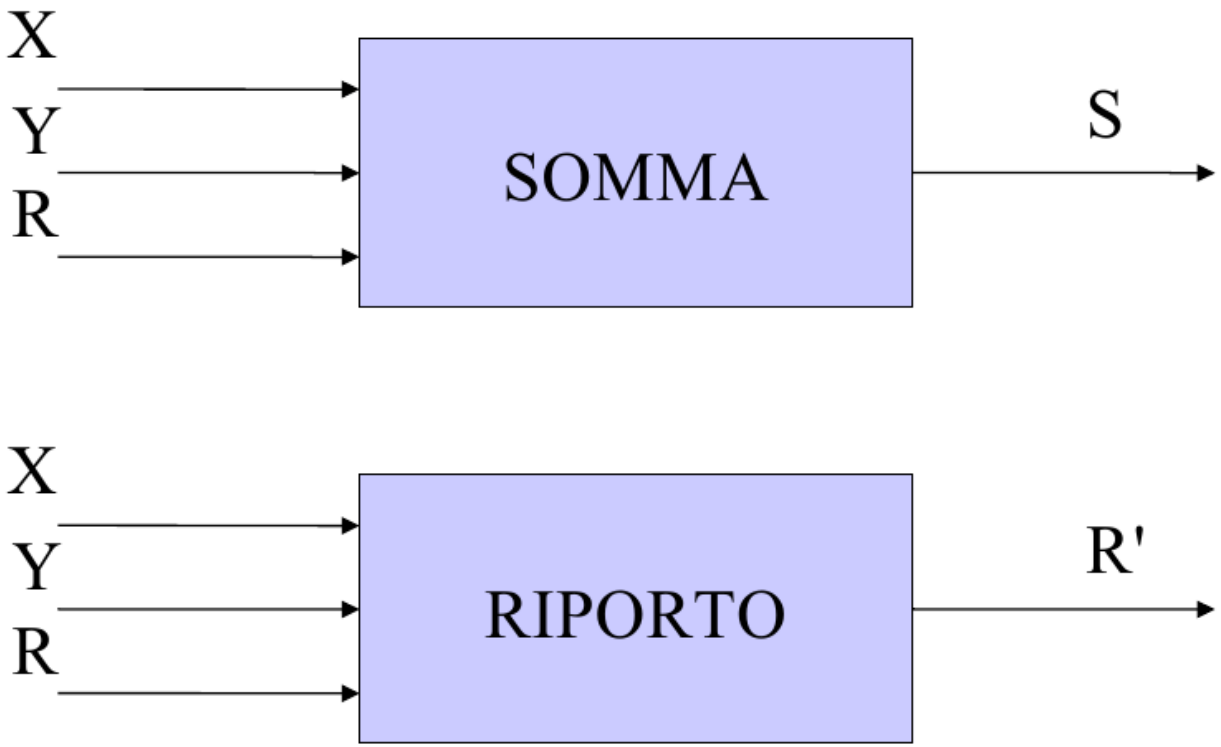
Il circuito Riporto può essere definito del seguente modo:



Per verificare se è corretto basta calcolare la sua tabella di verità e confrontarla con quella della slide precedente.
Notate che il circuito non è stato ottenuto dall'algoritmo che abbiamo definito per ottenere un circuto da una tabella di verità

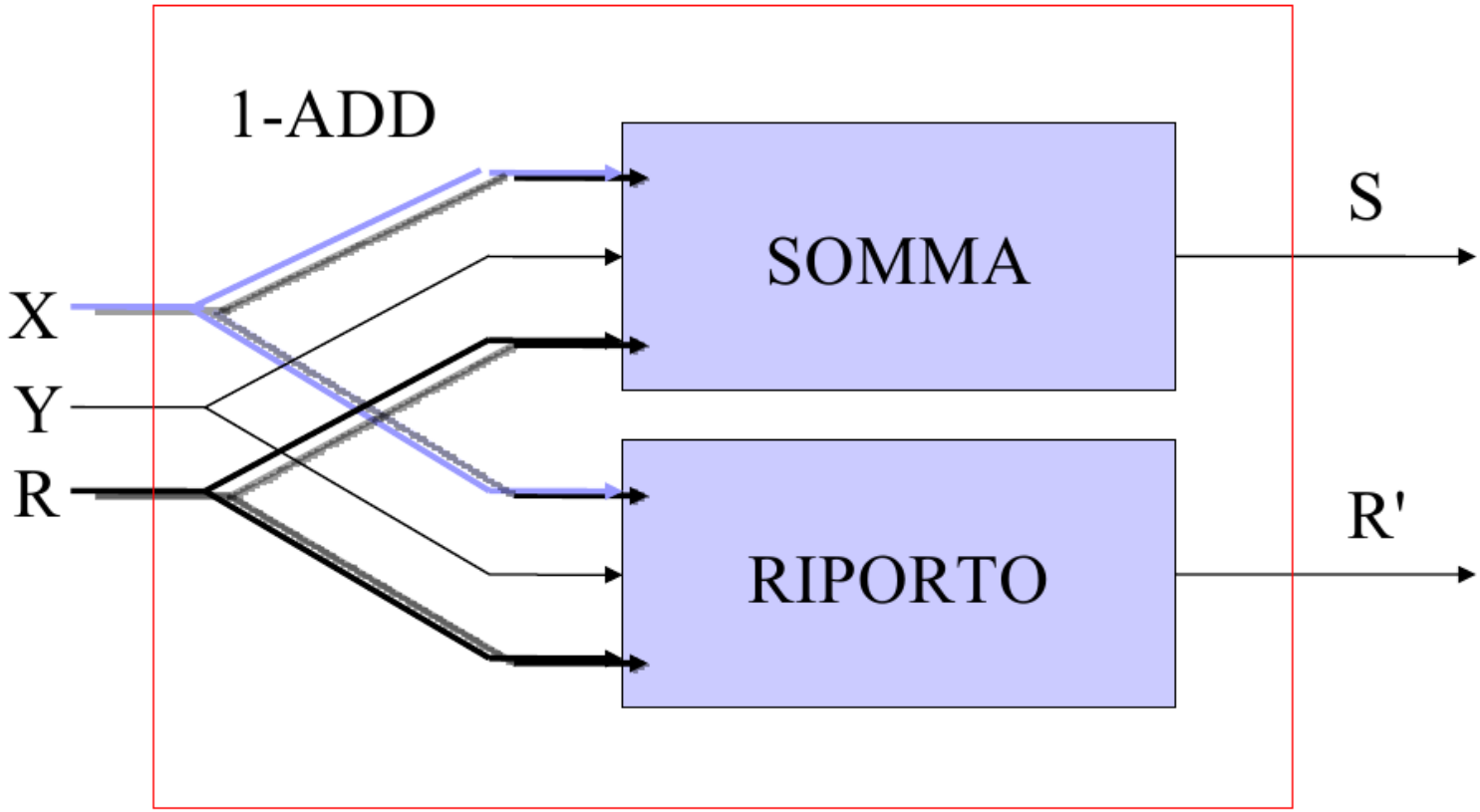
Somma di bit

Supponiamo di avere i circuiti che calcolano somma e riporto



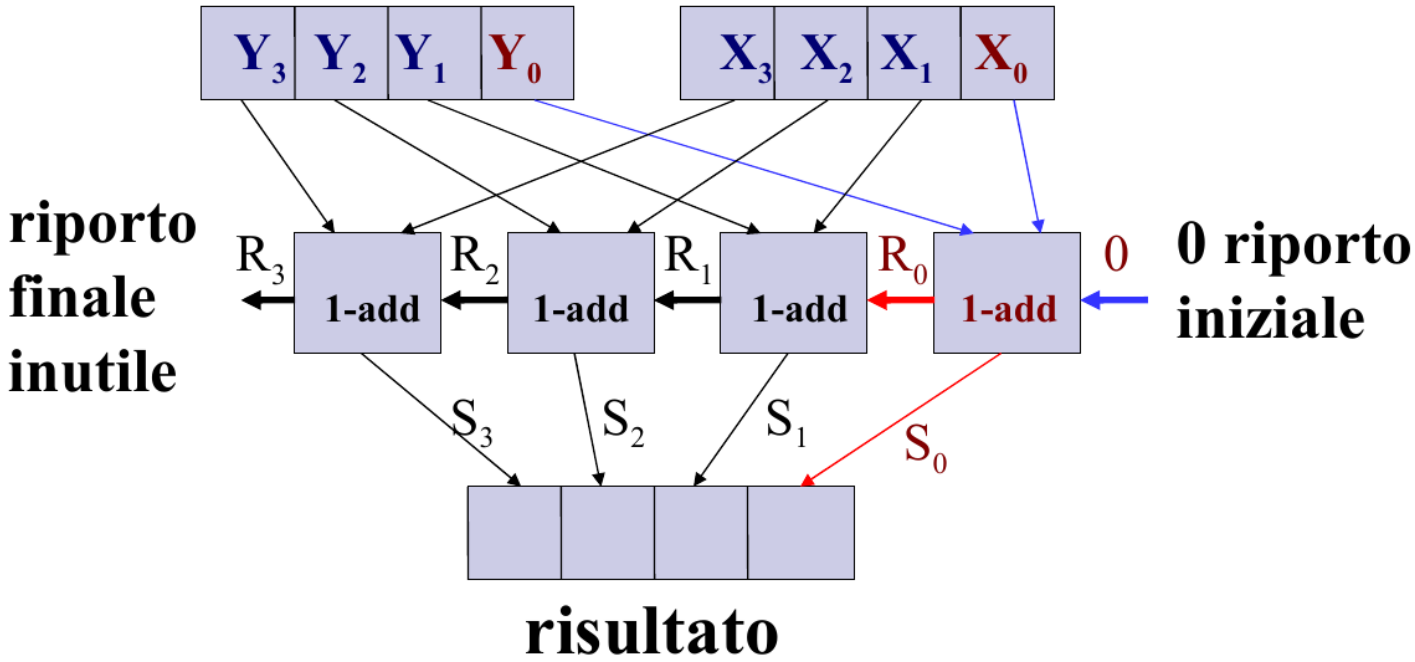
Circuito che somma due bit

Supponiamo di avere i circuiti che calcolano somma e riporto.
Possiamo combinarli per ottenere il circuito 1-ADD



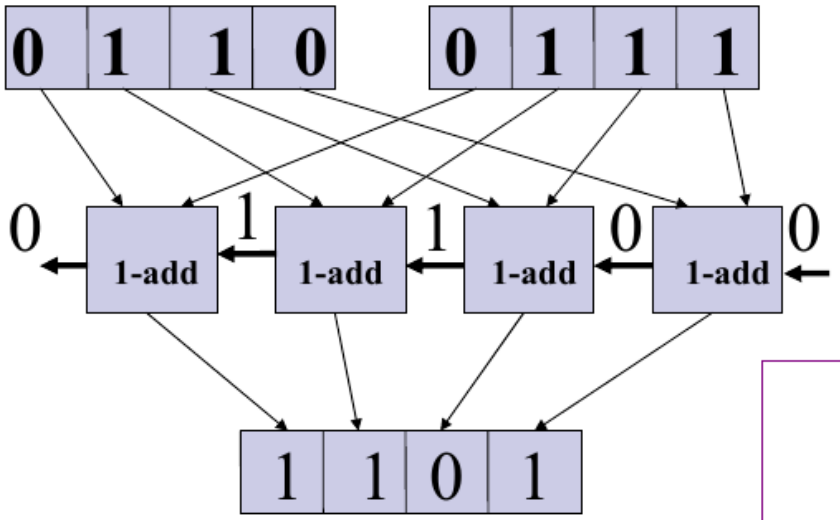
Circuito che somma k bit

Infine possiamo combinare insieme k circuiti 1-ADD ed ottenere un circuito K-ADD che somma numeri binari di k cifre.
Vediamo un esempio con $k = 4$



Circuito che somma k bit

Vediamo un esempio di utilizzo di un circuito 4-ADD



Esempio

```
0111 +
0110 =

------
1101
```

Overflow

- Si parla di overflow quando il risultato di un'operazione è troppo grande per essere rappresentato dai bit a disposizione.
- Per esempio, supponendo di avere a disposizione 5 bit per rappresentare un numero binario, la seguente somma produce overflow:

$$\begin{array}{cccccc} {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 1} & {\scriptstyle 0} & \\ & 1 & 1 & 0 & 1 & 0 \quad + \\ & 0 & 1 & 1 & 1 & 1 \\ - & - & - & - & - & - \\ 1 & 1 & 1 & 0 & 0 & 1 \end{array}$$

- Il circuito K-ADD da noi definito, per semplicità, non tratta il caso di overflow.