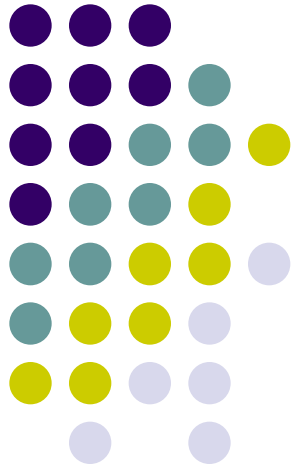
Università di Napoli Federico II - Dip. di Ingegneria Elettrica e delle Tecnologie dell'Informazione
Corso di Laurea in Ingegneria Informatica

DIETI

# Fondamenti di Informatica

## *Calcolo Proposizionale*



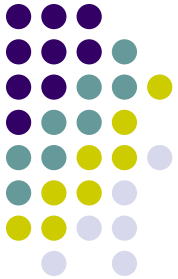
# Dualità

- Legge di dualità:

  Da qualsiasi ***identità*** booleana se ne può trarre un'altra per dualità, sostituendo cioè ad ogni operatore e ai valori F e T il rispettivo duale

- La legge di dualità ha come oggetto IDENTITA' booleane, cioè si applica ad «uguaglianze», dice che ad ogni uguaglianza ne corrisponde sempre una duale ottenuta sostituendo l'operatore AND con l'operatore OR, l'operatore OR con AND, il valore di verità F con V e il valore di verità V con F.

# Teorema di De Morgan



NOT(a OR b) = ((NOT a) AND (NOT b))

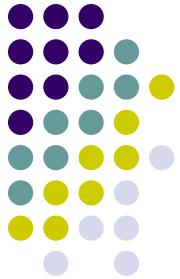
NOT(a AND b) = ((NOT a) OR (NOT b))

Dimostrazione usando le tabelle di verità

# Riepilogo delle proprietà della logica delle proposizioni

| | | | | |
|---|---|---|---|---|
| Commutativa | P1 | A oppure B = B oppure A | P'1 | A e B = B e A |
| Associativa | P2 | (A oppure B) oppure C = A oppure (B oppure C) | P'2 | (A e B) e C = A e (B e C) |
| Idempotenza | P3 | A oppure A = A | P'3 | A e A = A |
| Assorbimento | P4 | A oppure (A e B) = A | P'4 | A e (A oppure B) = A |
| Distributiva | P5 | A e (B oppure C) = (A e B) oppure (A e C) | P'5 | A oppure (B e C) = (A oppure B) e (A oppure C) |
| Min e max | P6 | A e Falso = Falso | P'6 | A oppure Vero = Vero |
| Complemento | P7 | A e (non A) = Falso | P'7 | A oppure (non A) = Vero |

# La logica delle Proposizioni e l'algebra di Boole

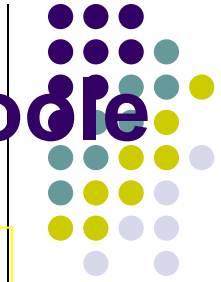


- **George Boole** (1815-1864) studiò **un modello matematico** per descrivere in **forma algebrica** la logica delle proposizioni e definì la cosiddetta **algebra di Boole.**
- L'algebra di Boole ha avuto importanti risvolti ed applicazioni pratiche in molti campi del sapere e dell'ingegneria, in particolare nel campo dei calcolatori e dell'elettronica:
- Ad esempio, si dimostra che è sempre possibile realizzare una qualsiasi tabella di verità componendo i connettivi logici (e, o, non): ciò nell'elettronica rivela che **è possibile realizzare qualsiasi circuito digitale** (e anche calcolatori digitali) con **pochi tipi di componenti elettronici elementari**.

# Tre importanti algebre di Boole

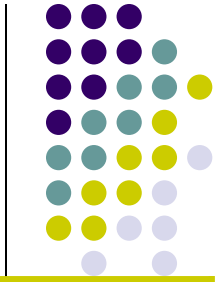
- Una **algebra di Boole** è un reticolo distributivo complementato e dotato di min e max assoluti
- Sono possibili diversi modelli di algebra di Boole. Alcune AdB "notevoli" sono:
  - l'algebra della logica delle proposizioni
  - l'algebra dei circuiti
  - l'algebra degli insiemi

# Calcolo proposizionale ed algebra di Boole



- Dunque il calcolo proposizionale “coincide” in pratica con il sistema basato sull’algebra di Boole:
- I valori di verità sono quelli booleani (VERO=1; FALSO=0)
- I connettivi logici sono gli operatori algebrici booleani * (AND) + (OR) – (NOT)
- Le argomentazioni sono quelle derivabili dai postulati dell’algebra

# Postulati dell'algebra di Boole



| | | | | |
|---|---|---|---|---|
| **Commutativa** | **P1** | **a+b = b+a** | **P'1** | **a•b = b•a** |
| **Associativa** | **P2** | **(a+b)+c = a+(b+c)** | **P'2** | **(a•b)•c = a•(b•c)** |
| **Idempotenza** | **P3** | **a+a = a** | **P'3** | **a•a = a** |
| **Assorbimento** | **P4** | **a+a•b = a** | **P'4** | **a•(a+b) = a** |
| **Distributiva** | **P5** | **a•(b+c) = a•b+a•c** | **P'5** | **a+b•c = (a+b)•(a+c)** |
| **Min e max** | **P6** | **a•0 = 0** | **P'6** | **a+1 = 1** |
| **Complemento** | **P7** | **a•ā = 0** | **P'7** | **a+ā = 1** |

# Teorema di De Morgan

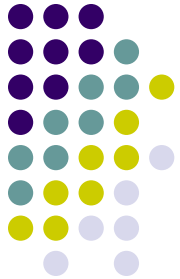
$$\overline{a+b} = \overline{a} \cdot \overline{b}$$

$$\overline{ab} = \overline{a} + \overline{b}$$

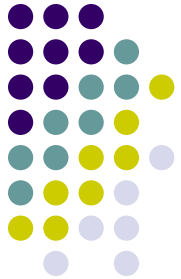
In logica delle proposizioni:

NOT(a OR b) = ((NOT a) AND (NOT b))
NOT(a AND b) = ((NOT a) OR (NOT b))

# Manipolazione algebrica

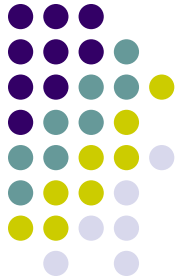


- E’ possibile da una funzione booleana derivare una funzione equivalente più semplice (che utilizza un minor numero di connettivi o anche di variabili)
- Esempio (AND/*, OR/+):
- A*B*C+(NOTA)*B*C+A*B*(NOTC)=
- A*B*C+A*B*C+(NOTA)*B*C+A*B*(NOTC)=
- B*C*(A+NOTA)+A*B*(C+NOTC)=
- B*C+A*B



***VEDIAMO ALCUNE APPLICAZIONI DEI CONCETTI APPENA ILLUSTRATI***

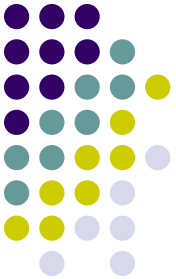
# Esempio 1



- L'algoritmo seguito dall'impiegato di un ufficio postale nell'accettare conti corrente da parte degli utenti in coda potrebbe contenere un passo del tipo:
  - **accetta un altro conto corrente se *l'utente ha già consegnato meno di 5 conti corrente o non ci sono altri utenti in coda***

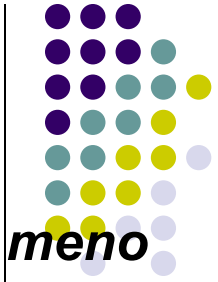
  oppure:
  - **accetta un altro conto corrente se *non* accade che *l'utente ha già consegnato almeno 5 conti corrente e ci sono altri utenti in coda***

# Risoluzione dell'esempio



- Costruiamo le tabelle di verità delle due espressioni
  - valore di verità dell'espressione composta in corrispondenza di ogni possibile combinazione di valori di verità delle condizioni atomiche componenti (*A* e *B*)
- Le due condizioni sono equivalenti se le due tabelle di verità sono uguali

# Risoluzione dell'esempio



**accetta un altro conto corrente se *l'utente ha già consegnato meno di 5 conti corrente o non ci sono altri utenti in coda***

Poniamo:

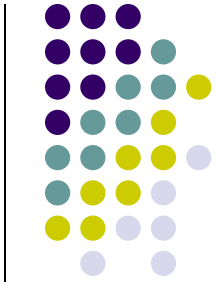
**A**: l'utente ha già consegnato meno di 5 conti correnti

**B**: ci sono altri utenti in coda

- La prima condizione corrisponde alla formula:

**A OR (NOT B)**

# Risoluzione dell'esempio



**accetta un altro conto corrente se *non* accade che**
***l'utente ha già consegnato almeno 5 conti corrente e ci sono altri utenti in coda***

Poniamo:

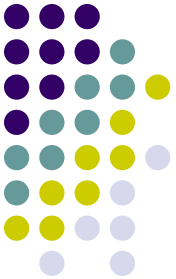
**A**: l'utente ha già consegnato meno di 5 conti correnti

**B**: ci sono altri utenti in coda

- La seconda condizione corrisponde alla formula:

**NOT((NOT A) AND B)**

# Confronto tabelle di verità



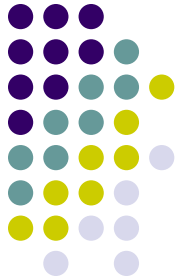
## prima condizione

| A | B | NOT B | A OR (NOT B) |
|---|---|---|---|
| F | F | V | V |
| F | V | F | F |
| V | F | V | V |
| V | V | F | V |

## seconda condizione

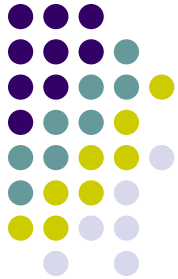
| A | B | NOT A | (NOT A) AND B | NOT ((NOT A) AND B) |
|---|---|---|---|---|
| F | F | V | F | V |
| F | V | V | V | F |
| V | F | F | F | V |
| V | V | F | F | V |

# Esempio 2



- Dati tre numeri reali positivi scrivere un programma in linguaggio C o C++ per stabilire se possono essere le dimensioni dei lati di un triangolo e in caso affermativo stabilire se il triangolo è equilatero, isoscele o scaleno.

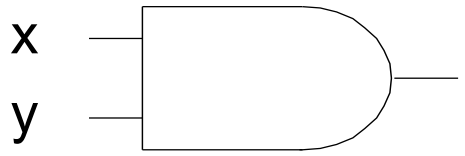
# Risoluzione dell'esempio



```
#include <iostream>
#include <stdlib.h>
using namespace std;

int main(int argc, char *argv[]) {

  float A, B, C;

  cout << "\n inserire tre numeri reali positivi versione 2: ";
  cin >> A >> B >> C;
  if (A>0 && B>0 && C>0)
     if (A<B+C && B<A+C && C<A+B) {
        if (A==B && B==C) cout << "\n il triangolo e' equilatero";
       else if(A==B || A==C || B==C) cout << "\n il triangolo e' isoscele";
             else cout << "\n il triangolo e' scaleno";
     }
     else cout << "\n i numeri inseriti non possono essere lati di un triangolo";
  else cout << "\n i numeri inseriti non sono tutti positivi";
  system("PAUSE");
  return 0; }
```

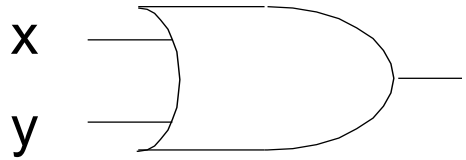
# Algebra dei circuiti

- In elettronica si utilizza l'algebra dei circuiti per modellare circuiti digitali.
- L'algebra dei circuiti è un'algebra di Boole che utilizza componenti circuitali elementari, le porte logiche, per trasformare segnali digitali.



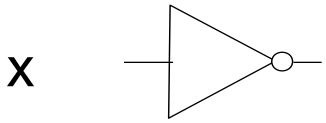

z = x AND y

| x | y | x AND y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |




z = x OR y

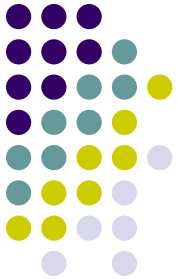
| x | y | x OR y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

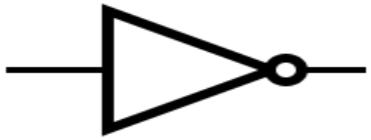



y = NOT x

| x | NOT x |
|---|---|
| 0 | 1 |
| 1 | 0 |

# PORTE LOGICHE



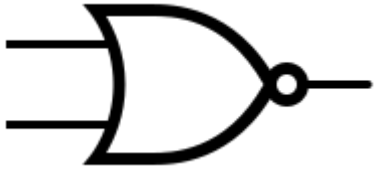


NOT

AND

NAND

OR

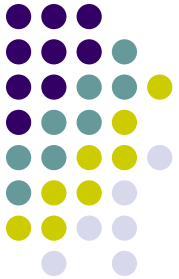


NOR

XOR

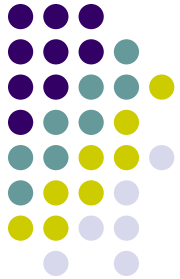
XNOR

# Esempio 3



- Progettazione di un addizionatore di numeri binari

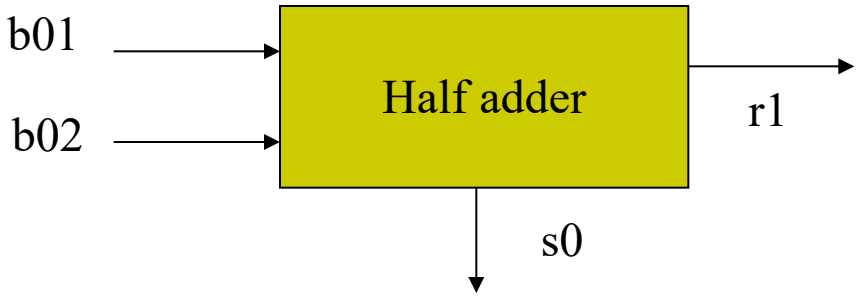
# Risoluzione dell'esempio



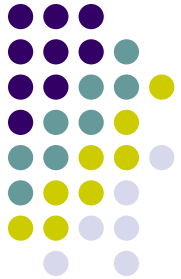
- L'algoritmo per sommare due numeri binari (e in generale due numeri in base b) può essere espresso nel modo seguente:
  - Sommare le due cifre meno significative
  - Per ogni altra coppia di cifre
    - sommare le cifre aggiungendo ad esse l'eventuale riporto della somma precedente
- Pertanto si vede che il problema si scompone in due "sotto-casi": il primo è costituito dalla somma delle cifre meno significative, il secondo dalla somma delle altre cifre che deve tener conto dell'eventuale riporto
- I due sottoproblemi possono essere risolti separatamente e poi "composti" nella soluzione finale

# Risoluzione dell'esempio

- Sommare le due cifre meno significative
- HALF-ADDER
  - dette b01 e b02 le due cifre meno significative dei numeri da sommare, il semi-sommatore le riceve in ingresso e prodice in uscita il risultato della somma (s0) ed il riporto (r1)



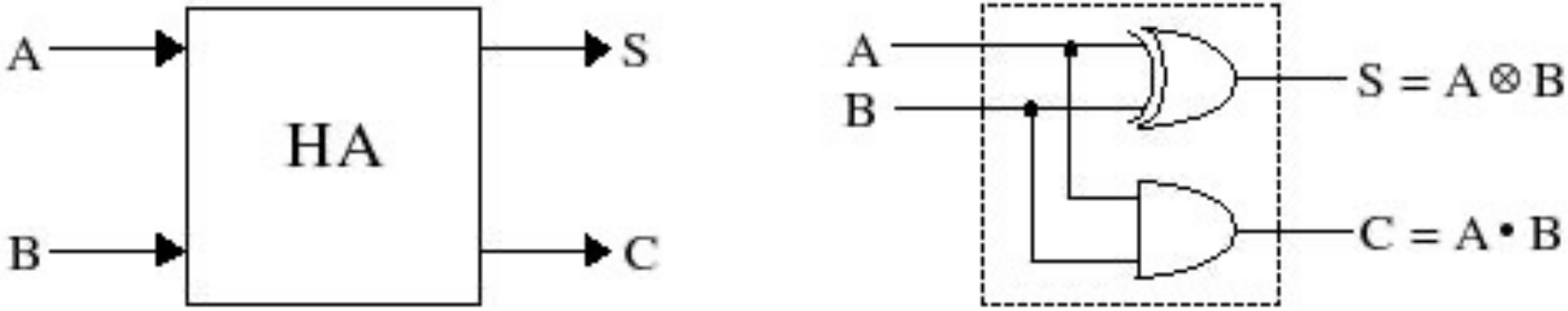
# Tabella di verità dell'half adder



| b01 | b02 | s0 | r1 |
|---|---|---|---|
| F | F | F | F |
| F | V | V | F |
| V | F | V | F |
| V | V | F | V |

- Dalla tabella risulta che s0 è XOR e r1 è AND

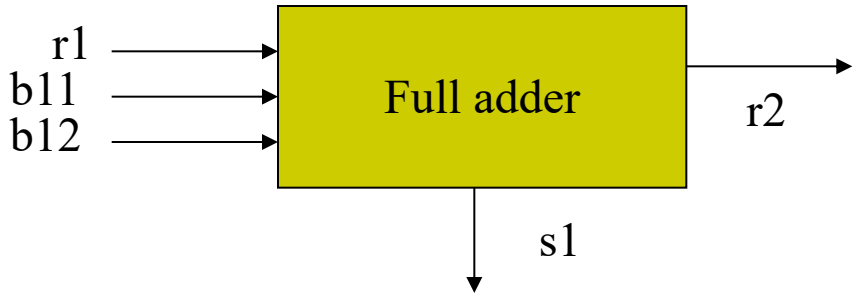
# Half adder



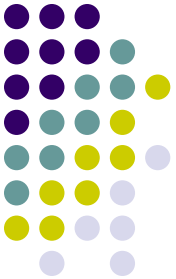

Simbolo logico e realizzazione circuitale di un HA

# Risoluzione dell'esempio

- Sommare due cifre considerando il riporto
- FULL ADDER
  - dette b11 e b12 le due cifre dei numeri da sommare ed r1 il riporto precedente, il sommatore li riceve in ingresso e produce in uscita il risultato della somma (s1) ed il riporto (r2)

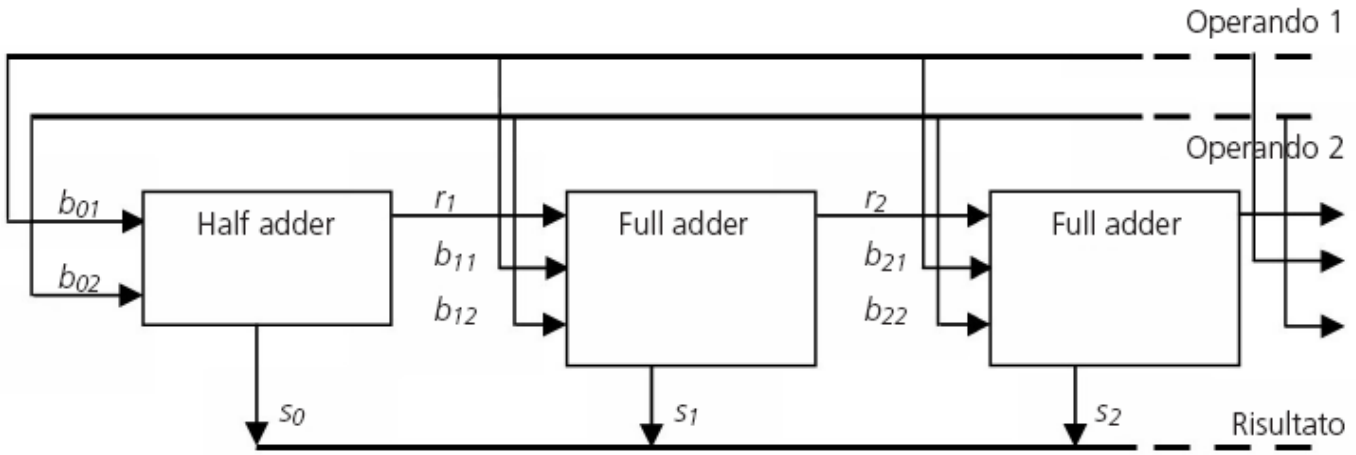


# Tabella di verità del Full adder



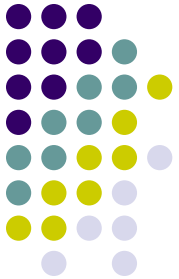
| b11 | b12 | r1 | s1 | r2 |
|---|---|---|---|---|
| F | F | F | F | F |
| F | F | V | V | F |
| F | V | F | V | F |
| F | V | V | F | V |
| V | F | F | V | F |
| V | F | V | F | V |
| V | V | F | F | V |
| V | V | V | V | V |

# Addizionatore di numeri binari




**Legenda** Le linee in grassetto indicano sequenze di più bit; le linee semplici indicano singoli bit.

# Riferimenti



- Dal libro di testo:
  - Teoria Capitolo 1 (parte centrale)
- Dalle dispense del Prof. Iannello: Capitolo 2