Sintesi di reti combinatorie

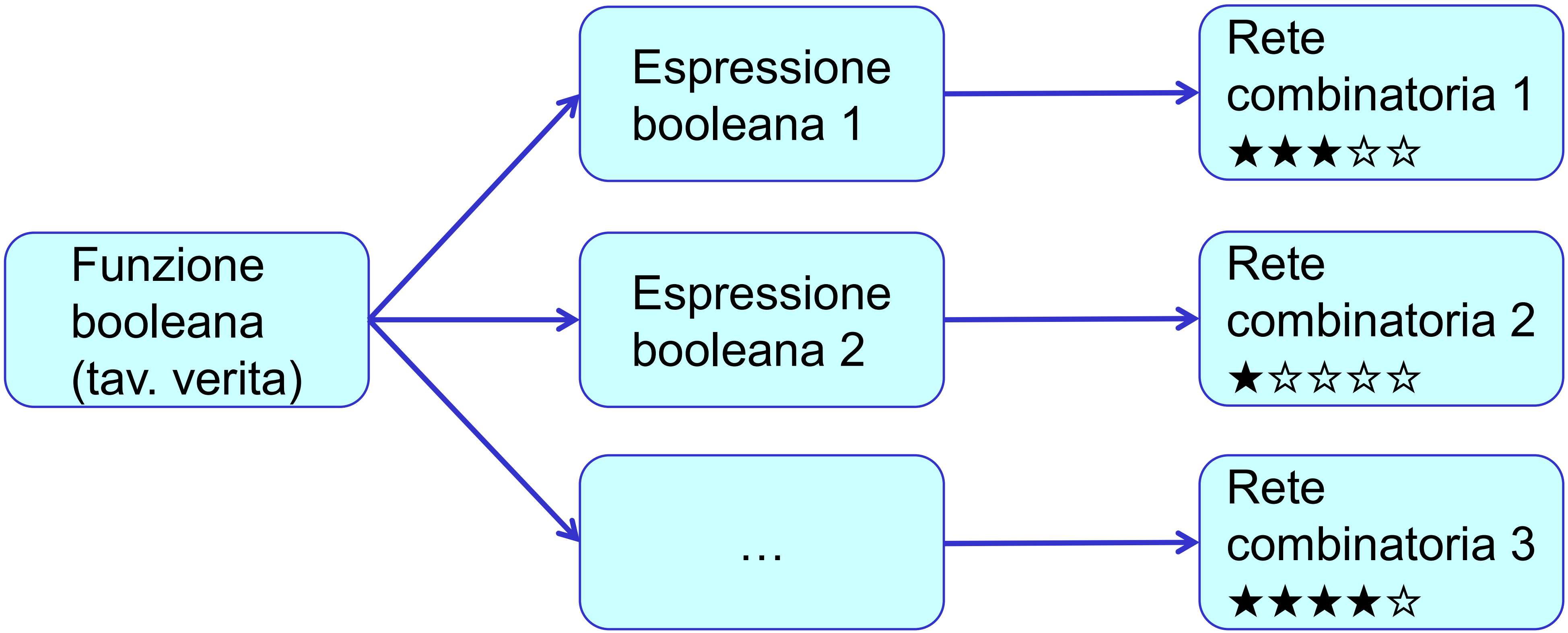
- Problema della sintesi:
  - data: una funzione booleana (espressa come tabella delle verità):
  - costruire (sintetizzare): lo schema logico di un circuito che la calcola
- Usiamo due passi:
  - 1: trovare **una** espressione booleana
    che esprime la funzione booleana data
  - 2: costruire la rete corrispondente all'espressione booleana
- In generale, per una data tabella delle verità possono esistere più espressioni booleane
  - <u>la soluzione al problema di sintesi non è dunque unica!</u>
  - problema: trovare la rete ottima (o almeno buona)

Reminder: reti combinatorie (funzionalmente) equivalenti

- Ogni rete combinatoria realizza <u>una</u> espressione booleana
- Ogni espressione booleana rappresenta <u>una</u> rete combinatoria
- Ma:
  tante espressioni booleane diverse possono esprimere
  la stessa funzione booleana (data come una tabella di verità)!
  - (cioè dare lo stesso valore per gli stessi assegnamenti delle variabili)
- Data una funzione booleana (come tabella di verità)
  esistono molte reti combinatorie che la realizzano
- Reti combinatorie diverse che realizzano la medesima funzione combinatoria si dicono (funzionalmente) equivalenti
- Esse computano tutte la stessa funzione, ma hanno
  - Diverso costo
  - Diverse prestazioni (es: velocità, consumo …)
- Ci dobbiamo porre il problema di scegliere la migliore
  - (o, almeno, una sufficientemente buona)

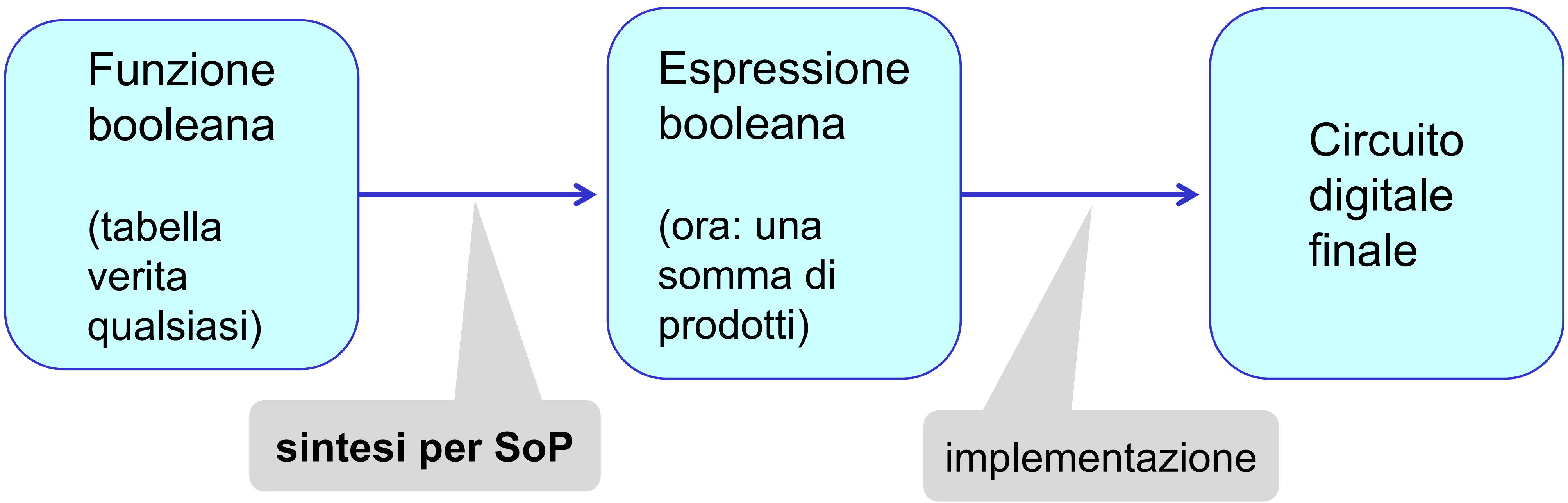
Sintesi di reti combinatorie

- Problema della sintesi:
  - data: una **funzione booleana** (espressa come tabella delle verità):
  - costruire (sintetizzare): lo schema logico di <u>un</u> **circuito** che la calcola
  - tipicamente: passaggio intermedio: una **espressione booleana**



Sintesi di reti combinatorie

- Esistono svariate tecniche di sintesi di reti combinatorie, che differiscono per:
  - Complessità della procedura
  - Qualità della rete combinatoria risultante
    (per dimensioni, costo, velocità, dissipazione di calore…)
- Una tecnica semplice e universale, benché generalmente non ottimale, è la sintesi attraverso **Somma di Prodotti** (detta **1ª forma canonica**)



Sintesi come **Somma di Prodotti (SoP)** (o sintesi in **1a forma canonica**)

- Input: la tabella delle verità della funzione da sintetizzare,

- Output: una somma di prodotti, cioè un'espressione booleana del tipo
  XXXX + YYYY + ZZZZ + ….
  dove ciascuno degli addendi della somma XXXX, YYYY …
  è un prodotto (di un certo numero di fattori).

- Procedimento:
  per ogni **1** nella colonna dell'uscita della tabella delle verità:
  - costruire un addendo della somma come prodotto di tutti i parametri:
    - Se il parametro x_i ha valore 1
      mettere nell'addendo il parametro naturale (es: A)
    - Se il parametro x_i ha valore 0
      mettere nell'addendo il parametro negato (es: /A)
  - costruire la somma di tutti gli addendi così ottenuti

Sintesi come **Somma di Prodotti (SoP)**: spiegazione intuitiva (con un esempio)

- F(A, B) =
  /A /B
  + /A B
  + A B

A	B	F
0	0	**1**
0	1	**1**
1	0	0
1	1	**1**

- F=1 se e solo se *uno qualsiasi dei casi 1 si verifica*, cioè quando…
  - si verifica il primo:
    A vale 0 e B vale 0
- …*oppure*…
  - si verifica il secondo:
    A vale 0 e B vale 1
- …*oppure*…
  - si verifica il terzo:
    A vale 1 e B vale 1

Esempio: funzione maggioranza

- Si chiede di sintetizzare (in 1ª forma canonica) una funzione combinatoria dotata di 3 ingressi A, B e C, e di un'uscita F, definita (a parole) come segue:
  - Se la maggioranza degli ingressi vale 0, l'uscita vale 0
  - Se la maggioranza degli ingressi vale 1, l'uscita vale 1

Esempio: funzione maggioranza
Tabella delle verità

- **Primo passo:**
  scriviamo la tabella delle verità

Esempio: funzione maggioranza
Tabella delle verità

- **Primo passo:**
  scriviamo la tabella delle verità

- E' quella mostrata a lato
- L'uscita vale 1 se e solo se 2 o tutti e 3 gli ingressi valgono 1 (cioè se e solo se il valore 1 è in maggioranza)

A	B	C	F
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

Esempio: funzione maggioranza
Sintesi espressione (in 1ma forma canonica)

- F(A, B, C) =
  /A B C +
  A /B C +
  A B /C +
  A B C

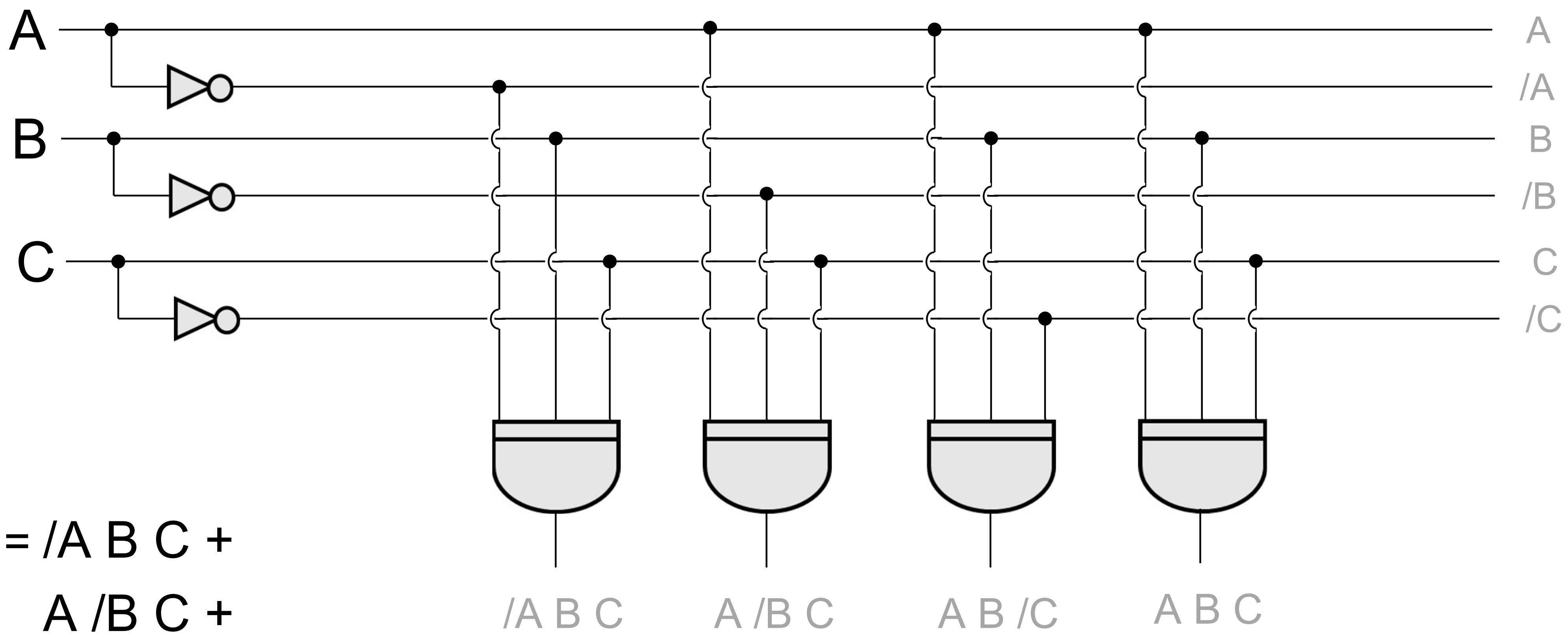
# riga	A	B	C	F
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

- E' una somma di prodotti
- Il passaggio successivo è quello di **implementare** questa espressione in un circuito
- Vediamo un modo semplice di farlo (per le Somme di Prodotti)

Modo pratico di disegnare un circuito per una Somma di Prodotti (step 1)

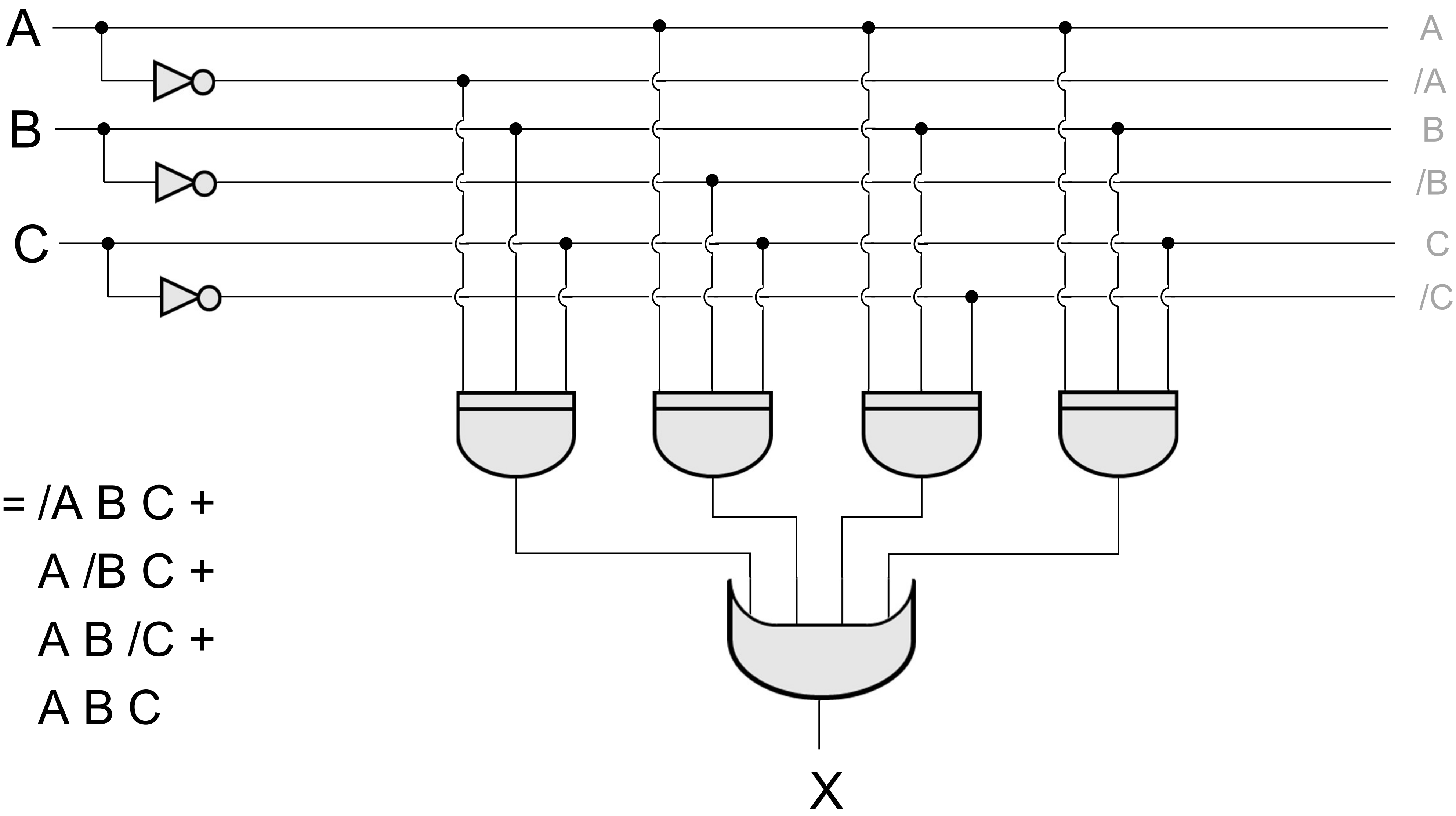
X = /A B C +
A /B C +
A B /C +
A B C

Modo pratico di disegnare un circuito per una Somma di Prodotti (step 2)



X = /A B C +
A /B C +
A B /C +
A B C

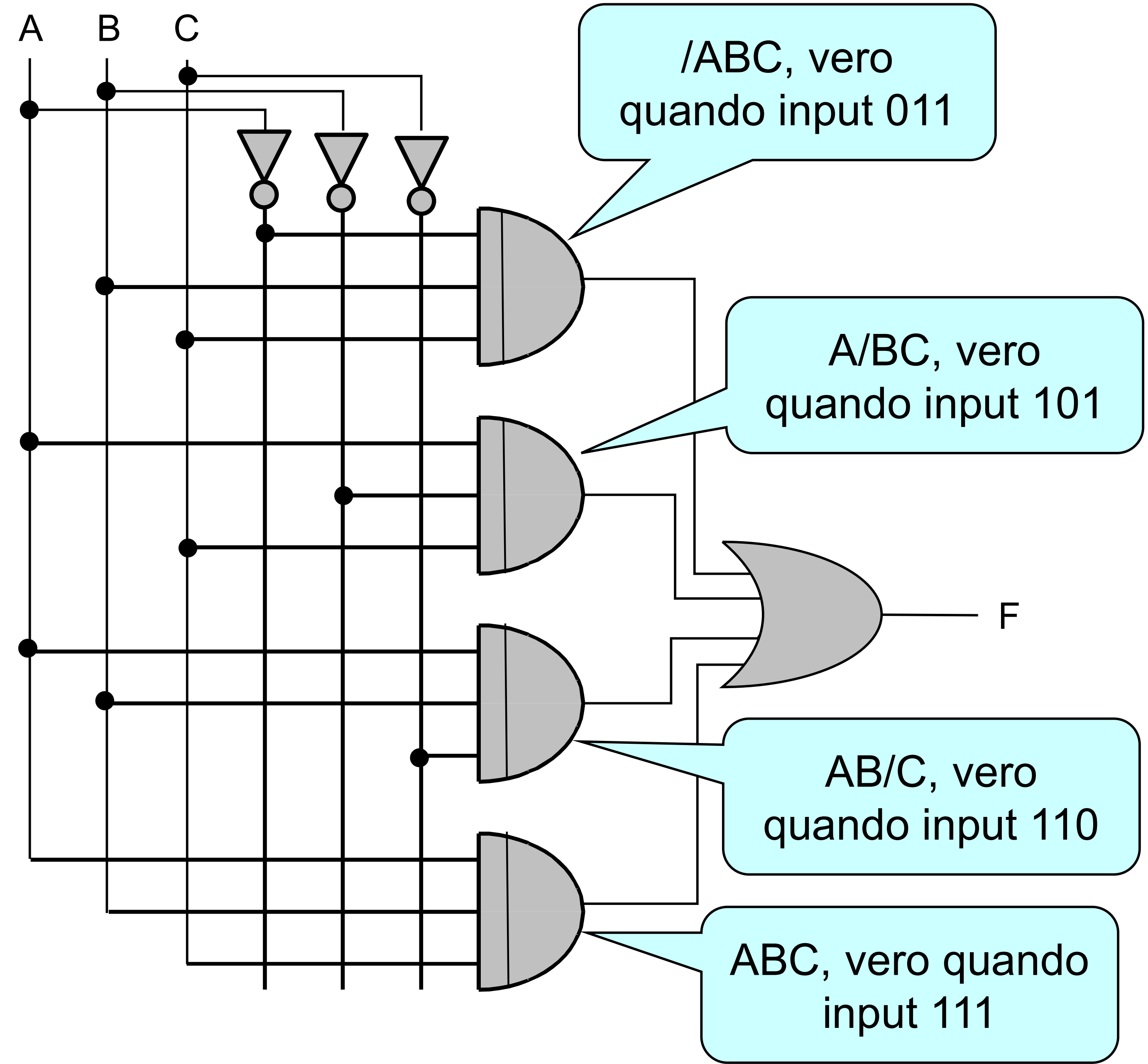
Modo pratico di disegnare un circuito per una Somma di Prodotti (step 3)



X = /A B C +
A /B C +
A B /C +
A B C

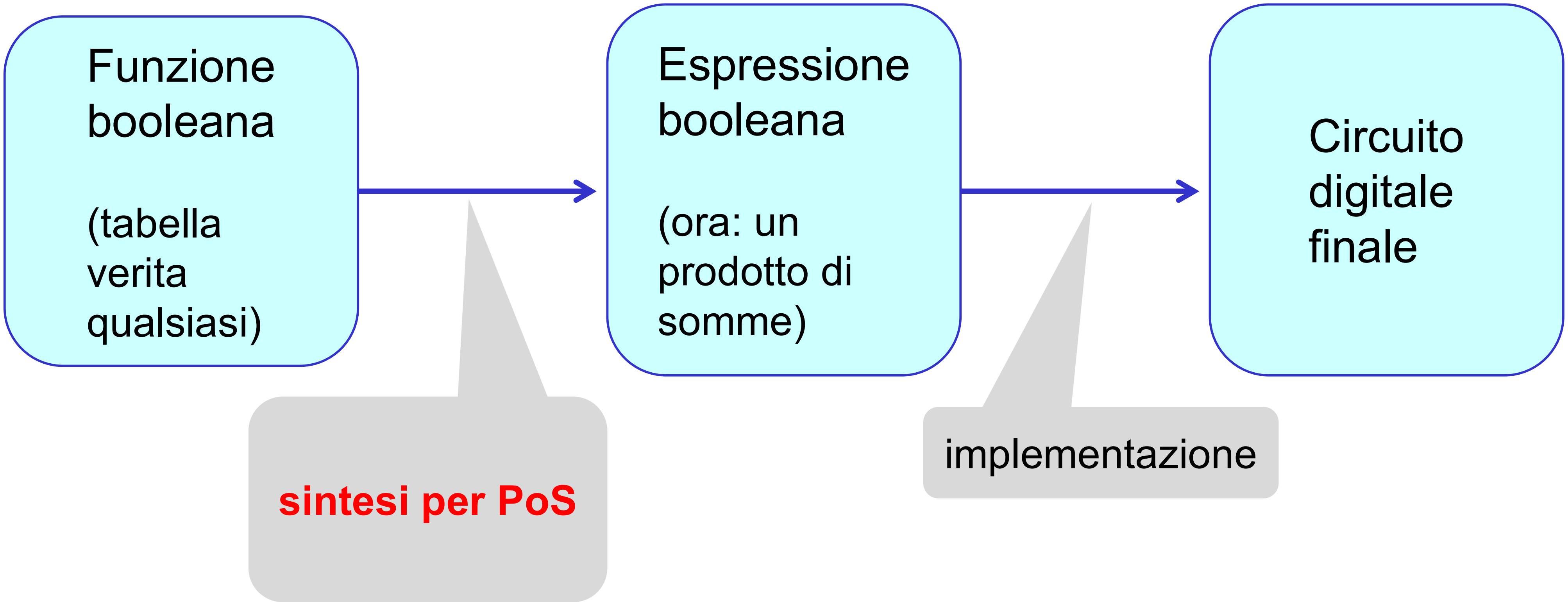
Modo pratico di disegnare un circuito per una Somma di Prodotti: una variante

F(A, B, C) =
/A B C +
A /B C +
A B /C +
A B C



Sintesi di espressioni, secondo modo: usare invece un **Prodotto di Somme (PoS)**

- Detto anche "**Seconda forma canonica**"



Sintesi come Prodotto di Somme (PoS): spiegazione intuitiva (con un esempio)

- F(A, B) =

  (A+B)

  (A+/B)

  (/A+B)

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

- F=1 se e solo se <u>*nessuno*</u> *dei casi 0 si verifica*, cioè quando…
  - Non si verifica il primo:
    A non 0 <u>oppure</u> B non 0
- …*e inoltre*…
  - non si verifica il secondo:
    A non 0 <u>oppure</u> B non 1
- …*e inoltre*…
  - non si verifica il terzo:
    A non 1 <u>oppure</u> B non 0

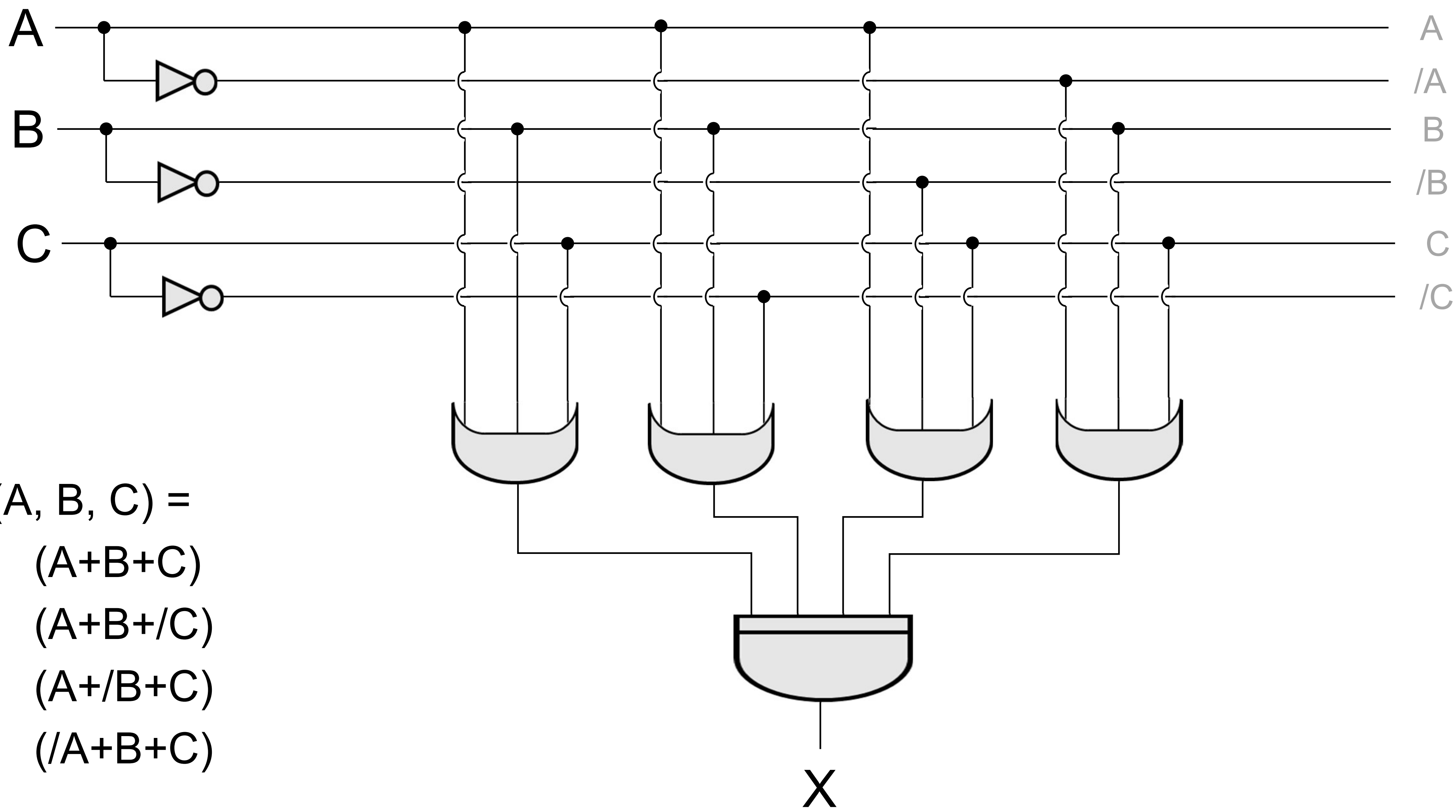
Sintesi per PoS con l'esempio precedente

- F(A, B, C) =
  (A+B+C)
  (A+B+/C)
  (A+/B+C)
  (/A+B+C)

# riga	A	B	C	F
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

- Nota bene: è un prodotto di somme

Modo pratico di disegnare un circuito per Prodotto di Somme



F(A, B, C) =

(A+B+C)

(A+B+/C)

(A+/B+C)

(/A+B+C)

Quale metodo conviene?

- PoS o SoP?
- In genere
  - se ci sono pochi 1 conviene SoP
  - se ci sono pochi 0 conviene PoS

Il Livello Logico-Digitale:

Trasformazione di Espressioni Booleane

Idea: ottimizzare l’espressione prima di implementarla in un circuito

Funzione booleana (tav. verita)

espressione booleana

prodotto di somme
OPPURE
somma di prodotti

espressione booleana più semplice

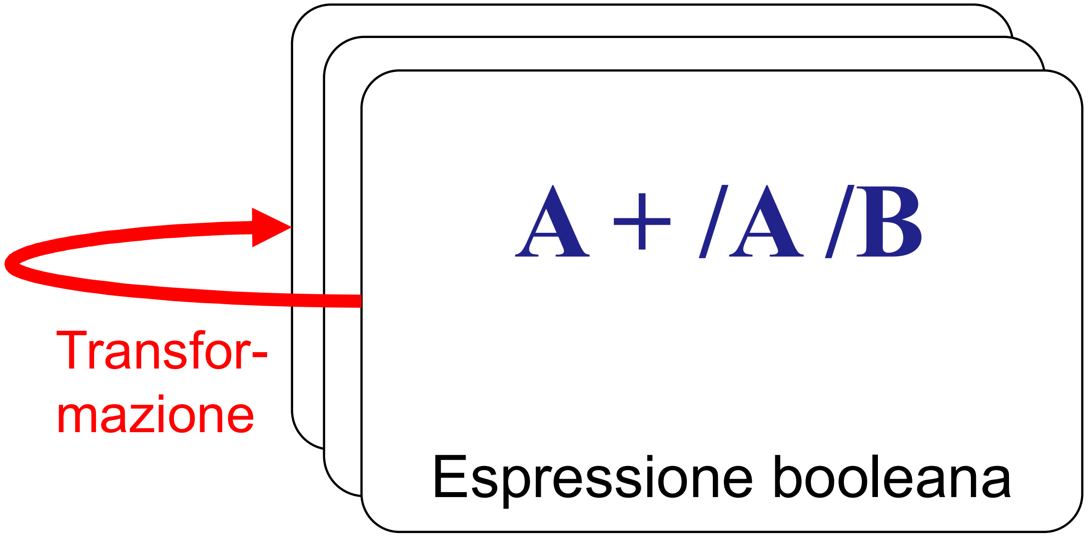
rete combinatoria ottimizzata

sintesi di espressione in 1ma oppure 2da forma canonica

Ottimizzazione usando regole di riscrittura delle espr. booleane

implementazione dell’espressione booleana in un circuito combinatorio

Trasformazione di espressioni booleane



passare da una espressione booleana ad una equivalente (stessa funzione booleana) con lo scopo di ridurne la complessità

Regole di trasformazione delle espressioni booleane

Legge	con AND	con OR (duale)
Identità	1 A = A	0 + A = A
Elemento nullo	0 A = 0	1 + A = 1
Idempotenza	A A = A	A + A = A
Inverso	A /A = 0	A + /A = 1
Commutativa	A B = B A	A + B = B + A
Associativa	(A B) C = A (B C)	(A + B) + C = A + (B + C)
Distributiva	A + B C = (A + B) (A + C)	A (B + C) = A B + A C
Assorbimento	A (A + B) = A	A + A B = A
De Morgan	/(A B) = /A + /B	/(A + B) = /A /B
Tertium non datur	/ / A = A	

Regole di trasformazione delle espressioni booleane

Dimostrare che: $A + B\,C = (A + B)\,(A + C)$

Regole di trasformazione delle espressioni booleane

Dimostrare che: $A + B\,C = (A + B)\,(A + C)$

$\mathbf{(A + B)\,(A + C)}$	$= AA + AC + BA + BC$	
	$= A + AC + BA + BC$	(idempot. con AND)
	$= A(1+C) + BA + BC$	(elem. nullo con OR)
	$= A + BA + BC$	(identità con AND)
	$= A(1+B) + BC$	(elem. nullo con OR)
	$\mathbf{= A + BC}$	(identità con AND)

Regole di trasformazione delle espressioni booleane

Dimostrare che: $A(A + B) = A$

Regole di trasformazione delle espressioni booleane

Dimostrare che: $A(A + B) = A$

$$
\begin{aligned}
\mathbf{A(A + B)} &= AA + AB \\
&= A + AB && \text{(idempot. con AND)} \\
&= A(1+B) && \text{(elem. nullo con OR)} \\
&\mathbf{= A}
\end{aligned}
$$

Regole di trasformazione: note

- Le regole di trasformazione (o «di riscrittura»)
  ci consentono di passare da una espressione ad un'altra, **equivalente**.
  - l'equivalenza è garantita dalla teoria!
  - Obiettivo delle riscritture: ottimizzare l'espressione di partenza
  - Cioè: rendere il circuito associato piú economico, o piú veloce, etc
- Gli A, B nelle regole rappresentano sotto-espressioni qualsiasi
  - (non necessariamente variabili: es: (A(B+C) + A(B+C)) = A(B+C)
- Tutte le regole sono in doppia copia: una per l'AND una per l'OR
  - una è la regola *DUALE* dell'altra
  - cioè una è ottenuta dall'altra scambiando fra di loro:
    AND <==> OR e 0 <==> 1
- Ciascuna regola si può usare in un verso, o nel verso opposto
  - XXX = YYY ➔ posso passare da XXX a YYY… oppure viceversa
- Alcune regole somigliano a quelle dell'algebra numerica tradizionale
  - Altre sono piuttosto diverse (per esempio i due assorbimenti)!