

# Componenti per l'aritmetica binaria

M. Favalli

Engineering Department in Ferrara



# Sommario

- 1 Introduzione
- 2 Sommatore binari
- 3 Applicazioni di n-bit adder
- 4 Sommatore CLA

# Sommario

- 1 Introduzione
- 2 Sommatore binari
- 3 Applicazioni di n-bit adder
- 4 Sommatore CLA

# Motivazioni

- I sistemi di calcolo necessitano di componenti che realizzino operazioni di tipo aritmetico (somme, prodotti ....) su numeri interi e in floating point
- Dal punto di vista teorico, le conoscenze che abbiamo ci consentono di realizzare qualsiasi funzione e quindi anche quelle svolte da moltiplicatori e sommatore
- Nel caso di interesse, questo approccio non dà però risultati soddisfacenti
- Ad esempio la sintesi ottima di reti a due livelli dà luogo a funzioni eccessivamente costose e non modulari

# Sommario

- 1 Introduzione
- 2 Sommatori binari**
- 3 Applicazioni di n-bit adder
- 4 Sommatore CLA

## Esempio di funzione aritmetica: 3-bit adder

- Si vuole realizzare un sommatore a 2 operandi per numeri interi positivi rappresentati su 3 bit.
- Siano  $A = \{a_2, a_1, a_0\}$  e  $B = \{b_2, b_1, b_0\}$  tali parole
- Il risultato é rappresentabile su 4 bit:  $S = \{s_3, s_2, s_1, s_0\}$
- Si supponga di sintetizzare le funzioni  $s_i$  come espressioni SP a uscita singola

## 3-bit adder

|      |      |      |      |      |
|------|------|------|------|------|
|      | b1b0 |      |      |      |
| a1a0 | 00   | 01   | 11   | 10   |
| 00   | 0000 | 0001 | 0011 | 0010 |
| 01   | 0001 | 0010 | 0100 | 0011 |
| 11   | 0011 | 0100 | 0110 | 0101 |
| 10   | 0010 | 0011 | 0101 | 0100 |

a2b2=00

|      |      |      |      |
|------|------|------|------|
| 0100 | 0101 | 0111 | 0110 |
| 0101 | 0110 | 1000 | 0111 |
| 0111 | 1000 | 1010 | 1001 |
| 0110 | 0111 | 1001 | 1000 |

a2b2=01

|      |      |      |      |
|------|------|------|------|
| 0100 | 0101 | 0111 | 0110 |
| 0101 | 0110 | 1000 | 0111 |
| 0111 | 1000 | 1010 | 1001 |
| 0110 | 0111 | 1001 | 1000 |

a2b2=10

|      |      |      |      |
|------|------|------|------|
| 1000 | 1001 | 1011 | 1010 |
| 1001 | 1010 | 1100 | 1011 |
| 1011 | 1100 | 1110 | 1101 |
| 1010 | 1011 | 1101 | 1100 |

a2b2=11

s3s2s1s0

## 3-bit adder

Le funzioni hanno un costo che a partire dal bit di minor peso ( $s_0$ ) aumenta molto rapidamente

|    |    |   |
|----|----|---|
|    | b0 |   |
| a0 | 0  | 1 |
| 0  | 0  | 1 |
| 1  | 1  | 0 |

$$s_0 = a_0 b_0' + a_0' b_0$$

|      |      |    |    |    |
|------|------|----|----|----|
|      | b1b0 |    |    |    |
| a1a0 | 00   | 01 | 11 | 10 |
| 00   | 0    | 0  | 1  | 1  |
| 01   | 0    | 1  | 0  | 1  |
| 11   | 1    | 0  | 1  | 0  |
| 10   | 1    | 1  | 0  | 0  |

$$s_1 = a_1' a_0' b_1 + a_1' b_1 b_0' + a_1 a_0' b_1' + a_1' a_0 b_1' b_0' + a_1 a_0 b_1 b_0 + a_1 b_1' b_0'$$

## 3-bit adder

Le funzioni hanno un costo che a partire dal bit di minor peso ( $s_0$ ) aumenta molto rapidamente

|      |    | b1b0 |    |    |    |
|------|----|------|----|----|----|
|      |    | 00   | 01 | 11 | 10 |
| a1a0 | 00 | 0    | 0  | 0  | 0  |
|      | 01 | 0    | 0  | 1  | 0  |
|      | 11 | 0    | 1  | 1  | 1  |
|      | 10 | 0    | 0  | 1  | 1  |

a2b2=00

|   |   |   |   |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 |

a2b2=01

|   |   |   |   |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 |

a2b2=10

|   |   |   |   |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 |

a2b2=11

$$s_3 = a_2'b_2'a_1b_1 + a_2'b_2'a_1a_0b_0 + a_2'b_2'a_0b_1b_0 + a_2'b_2b_1'b_0' + a_2'b_2a_1'b_0' + a_2'b_2a_1'a_0' + a_2'b_2a_1'a_0'$$

$$+ a_2b_2'b_1'b_0' + a_2b_2'a_1'b_1' + a_2b_2'a_1'a_0' + a_2b_2'b_0'b_1'$$

$$+ a_2b_2a_1b_1 + a_2b_2a_1a_0b_0 + a_2b_2b_0a_1a_0$$

s3s2s1s0

# Problemi nella realizzazione di sommatore binari

- Con l'aumentare della dimensione delle parole ( $n$ ), il costo di sommatore binari realizzati come reti a 2 livelli aumenta molto rapidamente
- L'utilizzo dei metodi di sintesi multilivello da luogo a miglioramenti relativi
- Le soluzioni che si ottengono non risultano modulari
- Come alternativa si vedrà un metodo che è basato sulla realizzazione hardware dell'algoritmo di somma per colonne

# Ripple-carry adder

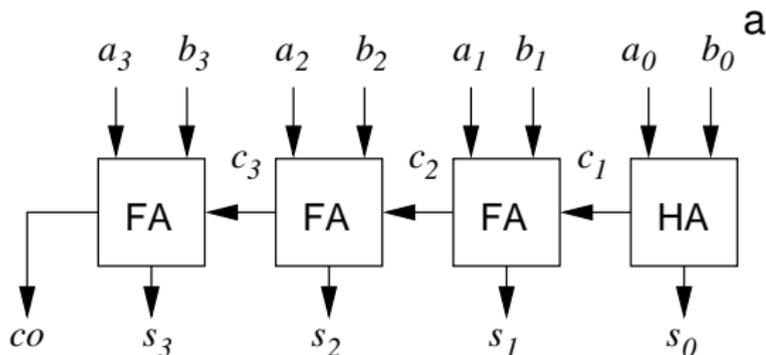
- Algoritmo di somma per colonne di due parole di  $n$  bit basato sulla propagazione del riporto (carry)

$$\begin{array}{r}
 c_4 \quad c_3 \quad c_2 \quad c_1 \\
 \swarrow \quad \swarrow \quad \swarrow \quad \swarrow \\
 a_3 a_2 a_1 a_0 + \\
 b_3 b_2 b_1 b_0 = \\
 \hline
 c_0 \quad s_3 s_2 s_1 s_0
 \end{array}$$

- $s_0 = (a_0 + b_0)_{mod2}$ ,  $c_1 = (a_0 + b_0)/2$
- $s_i = (a_i + b_i + c_i)_{mod2}$ ,  $c_{i+1} = (a_i + b_i + c_i)/2$
- $c_0 = c_n$
- ove  $+$  é la somma aritmetica e  $/$  é la divisione intera

# Ripple-carry adder

La descrizione funzionale si traduce in questo schema:



Dove i blocchi che gestiscono i bi di indice  $i > 0$  sono detti full-adder (FA) e quello che gestisce il caso  $i = 0$  è detto half-adder (HA)

# Half-adder

É un componente ampiamente utilizzato nell'aritmetica binaria.

Ingressi  $a = a_0$  e  $b = b_0$ , uscite  $s = s_0$  e  $c_{out} = c_1$

| $ab$ | $c_{out} s$ |
|------|-------------|
| 00   | 00          |
| 01   | 01          |
| 10   | 01          |
| 11   | 10          |

$$c_{out} = ab$$

$$s = ab' + a'b$$

# Full-adder

Il sommatore completo é anch'esso un componente fondamentale per l'aritmetica binaria. Ingressi  $a = a_0$ ,  $b = b_0$ , e  $c_{in} = c_i$ , uscite  $s = s_0$  e

$$C_{out} = C_1$$

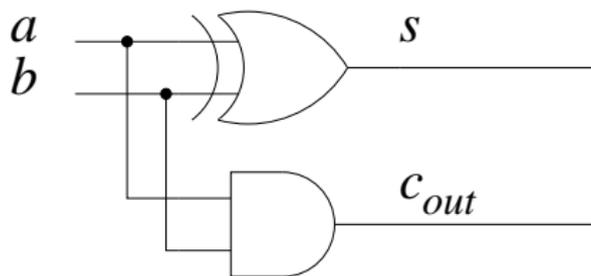
| $a$ $b$ $c_{in}$ | $C_{out}$ $s$ |
|------------------|---------------|
| 000              | 00            |
| 001              | 01            |
| 010              | 01            |
| 011              | 10            |
| 100              | 01            |
| 101              | 10            |
| 110              | 10            |
| 111              | 11            |

$$C_{out} = ab + ac_{in} + bc_{in}$$

$$s = a'b'c_{in} + a'bc_{in} + ab'c'_{in} + abc_{in}$$

# Realizzazione di un HA

$$c_{out} = ab \quad s = ab' + a'b = a \oplus b$$



Il gate EXOR che realizza la somma modulo 2 é un componente che può essere realizzato in tecnologia CMOS al livello switch (in maniera più complessa di NAND e NOR)

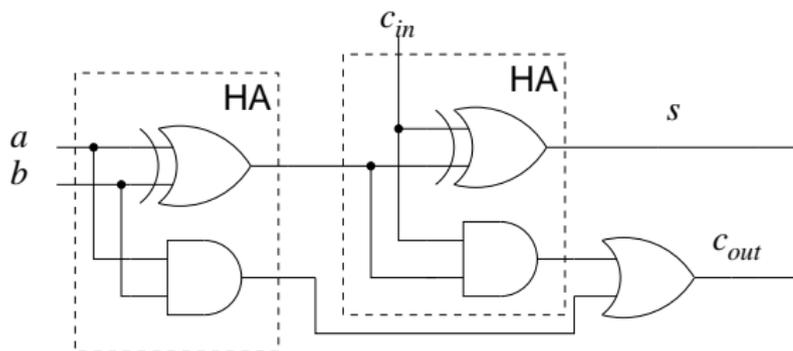
# Realizzazione di un FA

$$\begin{aligned}
 c_{out} &= ab + ac_{in} + bc_{in} \\
 &= ab + ac_{in}(b' + b) + bc_{in}(a' + a) \\
 &= ab + ab'c_{in} + abc_{in} + a'bc_{in} + abc_{in} \\
 &= ab + abc_{in} + abc_{in} + c_{in}(ab' + a'b) \\
 &= ab + c_{in}(a \oplus b)
 \end{aligned}$$

$$\begin{aligned}
 s &= a'b'c_{in} + a'bc'_{in} + ab'c'_{in} + abc_{in} \\
 &= c_{in}(a'b' + ab) + c'_{in}(ab' + a'b) \\
 &= c_{in}(ab' + a'b)' + c'_{in}(ab' + a'b) \\
 &= c_{in} \oplus (ab' + a'b) \\
 &= c_{in} \oplus (a \oplus b)
 \end{aligned}$$

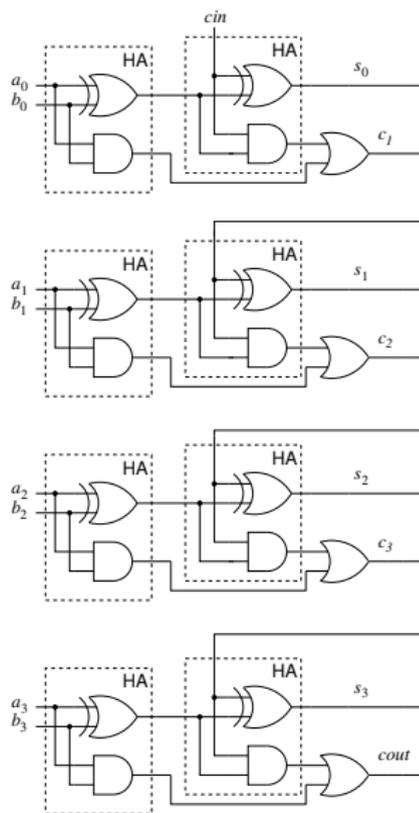
# Realizzazione di un FA

La realizzazione delle equazioni viste in precedenza consente di riconoscere la presenza di due HA



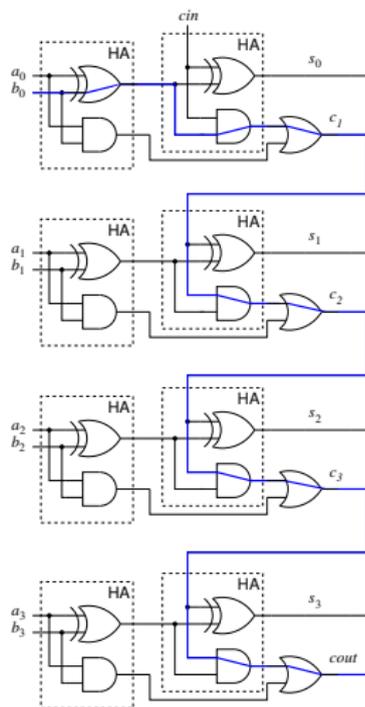
# Struttura di un n-bit adder (n=4)

Struttura al livello gate di un n-bit adder. Si noti che l'half-adder che somma  $a_0$  e  $b_0$  è stato sostituito da un full-adder in modo da poter utilizzare un carry-in di ingresso.



# Vantaggi e svantaggi

- Vantaggi: modularità e ridotto costo
- Svantaggi: ritardo



# Sommario

- 1 Introduzione
- 2 Sommatore binari
- 3 Applicazioni di n-bit adder**
- 4 Sommatore CLA

# n-bit adder: applicazioni

- Sommatore a  $kn$ -bit [▶ soluzione](#)
- Sommatore a più operandi [▶ soluzione](#)
- Valutazione di semplici espressioni aritmetiche
- Contatore di uni

# Sommario

- 1 Introduzione
- 2 Sommatore binari
- 3 Applicazioni di n-bit adder
- 4 Sommatore CLA**

# Sommatore carry-look ahead

- Per superare i problemi dovuti alle prestazioni del sommatore ripple-carry sono stati proposti diversi sommatore
- Uno dei primi ad essere stato proposto é il sommatore carry look ahead (CLA)
- Il sommatore CLA utilizza una rete a 3 livelli che si occupa di calcolare i carry di un n-bit adder senza bisogno di propagare il riporto

# Sommatore carry-look ahead

- In uscita all' $i$ -mo FA si ha riporto ( $c_{i+1} = 1$ ) se  $a_i$  e  $b_i$  hanno valori tali da produrre un riporto in uscita indipendentemente da  $c_i$  o se il loro valore é tale da garantire la propagazione di  $c_i = 1$
- Generazione del riporto per il bit di peso  $i$  (carry generate):  
$$g_i = a_i b_i$$
- Propagazione del riporto per il bit di peso  $i$  (carry propagate):  
$$p_i = a_i \oplus b_i$$
- $c_{i+1} = g_i + p_i c_i$
- L'applicazione iterattiva di questa formula porta alla logica di generazione dei riporti di un CLA
- I bit della somma sono calcolati come:  $s_i = a_i \oplus b_i \oplus c_i$

## Sommatore carry look ahead (n=4)

$$C_1 = g_0 + p_0 C_{in}$$

$$C_2 = g_1 + p_1 C_1$$

$$C_3 = g_2 + p_2 C_2$$

$$C_{out} = g_3 + p_3 C_3$$

Sostituendo iterativamente

$$C_1 = g_0 + p_0 C_{in}$$

$$C_2 = g_1 + p_1 (g_0 + p_0 C_{in})$$

$$C_3 = g_2 + p_2 (g_1 + p_1 (g_0 + p_0 C_{in}))$$

$$C_{out} = g_3 + p_3 (g_2 + p_2 (g_1 + p_1 (g_0 + p_0 C_{in})))$$

# Sommatore carry look ahead (n=4)

Applicando la proprietà distributiva

$$c_1 = g_0 + p_0 c_{in}$$

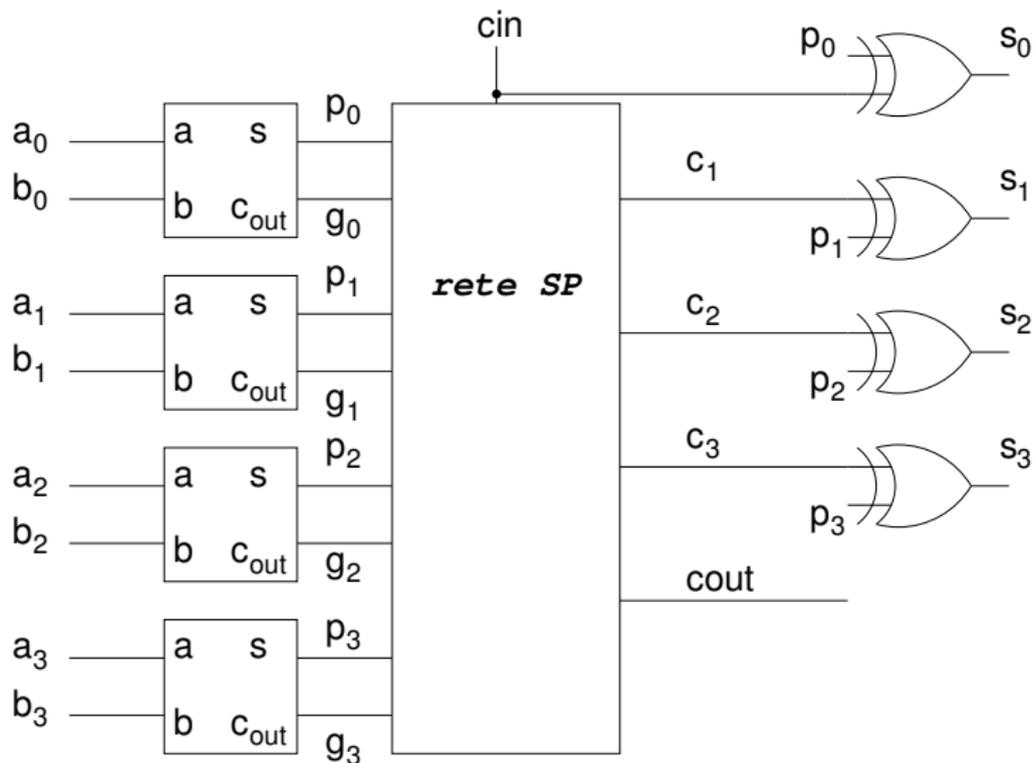
$$c_2 = g_1 + p_1 g_0 + p_1 p_0 c_{in}$$

$$c_3 = g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_{in}$$

$$c_{out} = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 c_{in}$$

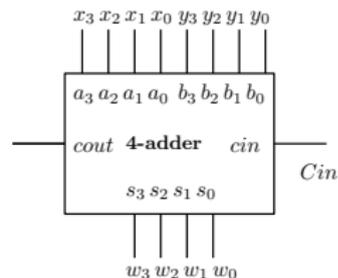
Come si osserva ciascun bit di carry viene espresso come un'espressione SP a due livelli in funzione dei bit di carry generate e propagate e del carry-in esterno

## Struttura di un sommatore CLA per n=4



# Sommatore a $kn$ bit usando $k$ sommatore a $n$ bit [▶ return](#)

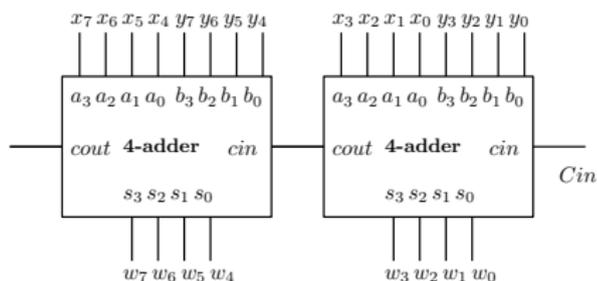
Sia  $k = 4$  e  $n = 4$



1. si prendono i 4 bit di minor peso di  $x$  e  $y$  e si connettono a un 4 bit adder

# Sommatore a $kn$ bit usando $k$ sommatore a $n$ bit ▶ return

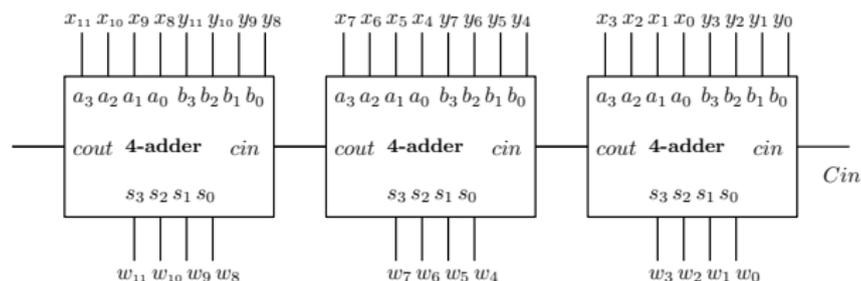
Sia  $k = 4$  e  $n = 4$



1. si prendono i 4 bit di minor peso di  $x$  e  $y$  e si connettono a un 4 bit adder
2. si procede aggiungendo 4 bit adder connessi in modo da avere il *cout* di uno connesso al *cin* del successivo

# Sommatore a $kn$ bit usando $k$ sommatore a $n$ bit ▶ return

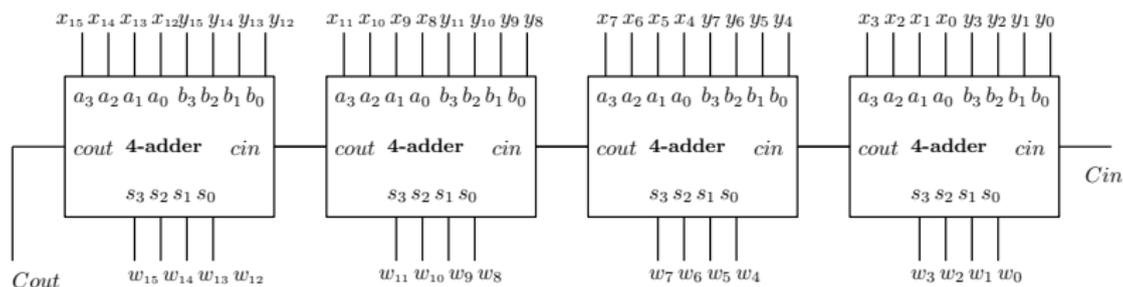
Sia  $k = 4$  e  $n = 4$



1. si prendono i 4 bit di minor peso di  $x$  e  $y$  e si connettono a un 4 bit adder
2. si procede aggiungendo 4 bit adder connessi in modo da avere il  $cout$  di uno connesso al  $cin$  del successivo

# Sommatore a $kn$ bit usando $k$ sommatore a $n$ bit ▶ return

Sia  $k = 4$  e  $n = 4$



1. si prendono i 4 bit di minor peso di  $x$  e  $y$  e si connettono a un 4 bit adder
2. si procede aggiungendo 4 bit adder connessi in modo da avere il  $cout$  di uno connesso al  $cin$  del successivo

# Sommatore a 3 operandi da 4 bit ciascuno ▶ return

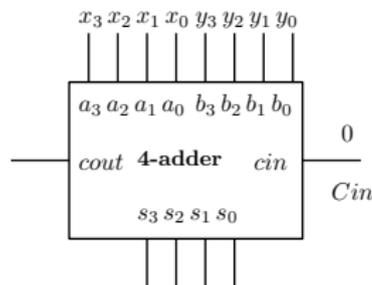
Si vuole calcolare  $S = X + Y + W$

1. in questa soluzione non vengono perse informazioni:  $S$  ha un numero di bit sufficiente per contenere il risultato
2.  $S \leq 45$  e quindi deve avere 6 bit

# Sommatore a 3 operandi da 4 bit ciascuno ▶ return

Si vuole calcolare  $S = X + Y + W$

1. in questa soluzione non vengono perse informazioni:  $S$  ha un numero di bit sufficiente per contenere il risultato
2.  $S \leq 45$  e quindi deve avere 6 bit
3. per prima cosa si sommano  $X$  e  $Y$  con un 4 bit adder di cui vanno usate tutte le uscite



# Sommatore a 3 operandi da 4 bit ciascuno ▶ return

Si vuole calcolare  $S = X + Y + W$

1. in questa soluzione non vengono perse informazioni:  $S$  ha un numero di bit sufficiente per contenere il risultato
2.  $S \leq 45$  e quindi deve avere 6 bit
3. per prima cosa si sommano  $X$  e  $Y$  con un 4 bit adder di cui vanno usate tutte le uscite
4. poi bisogna utilizzare un 5 bit adder per sommare  $W$  a  $X + Y$  e anche in questo caso si utilizzano le 6 uscite

