

Aritmetica del Calcolatore

Giovanni Stea

Reti Logiche

Corso di Laurea in Ingegneria Informatica, Universita' di Pisa

a.a. 2020/21

Materiale di riferimento

- Paolo Corsini, "Circuiti logici per le operazioni sui numeri naturali e sui numeri interi", edizioni ETS
- Appunti sull'aritmetica - versione 30/10/2019 (con esercizi svolti)

Rappresentazione dei numeri naturali

- Numero naturale: concetto intuitivo
- **sistema numerico di rappresentazione di tipo posizionale:**
 - a) Un numero $\beta \geq 2$, detto **base di rappresentazione**. Nel caso del sistema decimale, è $\beta =$ dieci. $5 \quad 20 \quad 95 \quad 420 \quad 15 \quad 12$
 - b) Un insieme di β simboli, detti **cifre**, a ciascuno dei quali è associato un numero naturale compreso tra 0 e $\beta - 1$.
 - c) Una **legge di rappresentazione** che fa corrispondere ad ogni sequenza di cifre un numero naturale.

Notazione posizionale

$$A \in (-\infty, \infty)_{\beta}$$

- Dato il numero $A \in \mathbb{N}$,
- rappresento A in base β con una sequenza di cifre

$$(a_{n-1} a_{n-2} \dots a_1 a_0)_{\beta}$$

- con $0 \leq a_i \leq \beta - 1$, $0 \leq i \leq n - 1$.

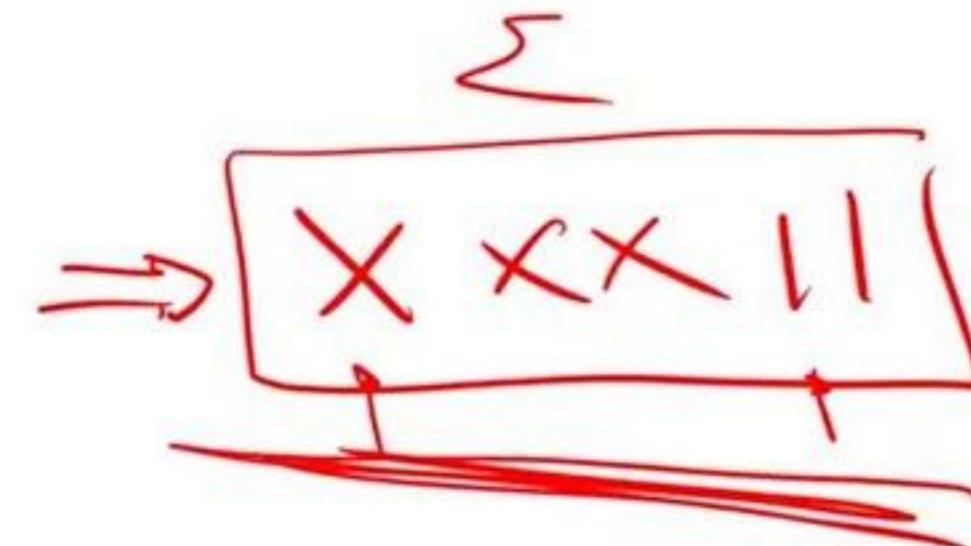
- Dirò in tal caso che $A \equiv (a_{n-1} a_{n-2} \dots a_1 a_0)_{\beta}$

- In un sistema numerico **posizionale**, la legge che fa corrispondere il numero naturale con la sua rappresentazione è

$$A = \sum_{i=0}^{n-1} a_i \cdot \beta^i$$

Sistema posizionale (cont.)

Algoritmi



- Nella rappresentazione di un numero naturale, una cifra contribuisce a determinare il numero in modo **differente** a seconda della propria posizione
- Infatti, a seconda della propria posizione sarà moltiplicata per una differente potenza della base

• $\beta = \text{dieci}$

• Le cifre sono $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

• $(2042)_{10} = 2 \cdot 10^0 + 4 \cdot 10^1 + 0 \cdot 10^2 + 2 \cdot 10^3 = \sum_{i=0}^3 a_i \cdot 10^i$

- Alternativa: sistema additivo (e.g., i numeri romani)

Numero naturale e sua rappresentazione

- Un numero e' un concetto
- La sua rappresentazione dipende
 - Dal sistema numerico adottato
 - Dalla base usata

• $(54)_{10}$ LIV $(36)_{16}$

Esempi

- **sistema numerico esadecimale**
- $\beta = \text{sedici}$. Le cifre sono $\{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$.

$$(1A2F)_{16} = (F)_{16} \cdot (10^0)_{16} + (2)_{16} \cdot (10^1)_{16} + (A)_{16} \cdot (10^2)_{16} + (1)_{16} \cdot (10^3)_{16}$$

$$\begin{aligned} 10\} &= (15)_{10} \cdot (16^0)_{10} + (2)_{10} \cdot (16^1)_{10} + (10)_{10} \cdot (16^2)_{10} + (1)_{10} \cdot (16^3)_{10} \\ &= 15 \cdot 1 + 2 \cdot 16 + 10 \cdot 256 + 1 \cdot 4096 = \underline{\underline{6703}} \end{aligned}$$

- **sistema numerico binario**
- $\beta = \text{due}$. Le cifre sono $\{0,1\}$

$$(1001)_2 = 1 \cdot 2^0 + 0 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3$$

$$\brace{1+8} = 9$$

Teorema della divisione con resto

- Dato un numero naturale A ed una base di rappresentazione β
 - Posso **sempre** rappresentare A in base β ?
 - La sua rappresentazione è **unica**?
 - Come faccio a **trovarla**?

$x \in \mathbb{N}$

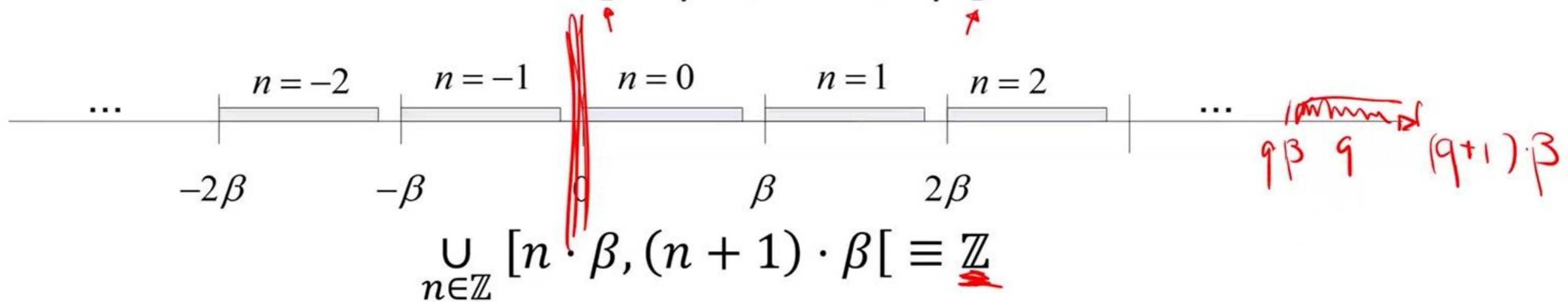
- **Teorema della Divisione con Resto**

- Dato $x \in \mathbb{Z}$, $\beta \in \mathbb{N}$, $\beta > 0$, esiste ed è **unica** la coppia di numeri q, r ,
 - $q \in \mathbb{Z}$
 - $r \in \mathbb{N}$, $0 \leq r < \beta$,
 $\leq \beta - 1$
 - tale che $x = q \cdot \beta + r$.

$$\underline{q \in \mathbb{Z}} \quad \underline{0 \leq r < \beta}$$

Teorema della divisione con resto (cont.)

- Esistenza: Dimostriamo che una coppia q, r esiste sempre
- Divido la retta \mathbb{Z} in intervalli: $[n \cdot \beta, (n + 1) \cdot \beta[, n \in \mathbb{Z}$.



- Pertanto, il numero x fa parte di un intervallo, sia il q -simo.
- $q \cdot \beta \leq x < (q + 1) \cdot \beta$
- Definisco $r = \underline{x - q \cdot \beta}$, e quindi $0 \leq r < \beta$.
- Ho quindi dimostrato che una tale coppia $\overset{\uparrow}{r}$ $\overset{\uparrow}{q}$ esiste sempre.

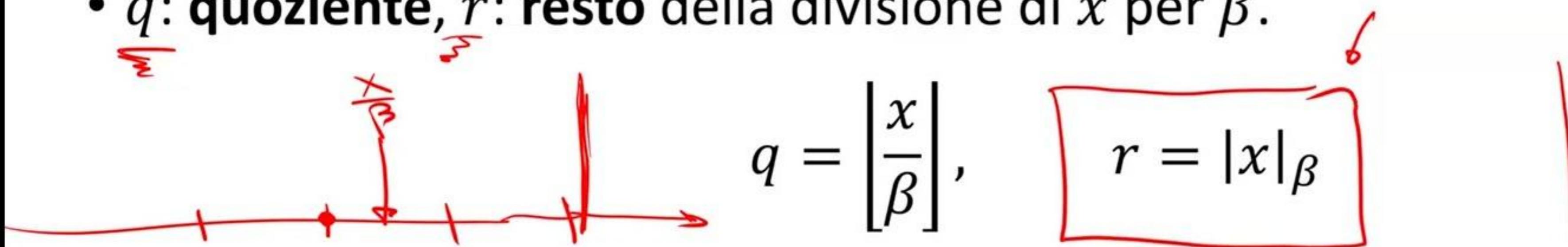
Teorema della divisione con resto (cont.)

- **Unicità:** Supponiamo per assurdo che esistano **due** coppie (q_1, r_1) e (q_2, r_2) diverse tali che $x = \underline{q_1 \cdot \beta + r_1} = \underline{q_2 \cdot \beta + r_2}$,
- con $q_i \in \mathbb{Z}$ e $0 \leq r_i < \beta$.
- Allora $\boxed{(q_1 - q_2) \cdot \beta = r_2 - r_1}$.
- Però per ipotesi $0 \leq r_i < \beta$, quindi $-\beta < (r_2 - r_1) < \beta$.
- Quindi $-\beta < (q_1 - q_2) \cdot \beta < \beta$,
- Cioè $\underline{-1} < (q_1 - q_2) < \underline{1}$. Visto che $q_1, q_2 \in \mathbb{Z}$, ne segue che $q_1 = q_2$.
- Se $q_1 = q_2$, allora anche $r_1 = r_2$, contro l'ipotesi. Questo dimostra che la divisione col resto ha un **unico risultato**.



Teorema della divisione con resto (cont.)

- L'unicità del risultato è garantita dal fatto che $0 \leq r \leq \beta - 1$
- Inoltre, $x \in \mathbb{N} \Rightarrow q \in \mathbb{N}$. Se la divisione è tra naturali, anche q è naturale.
- \underline{q} : **quoziente**, \underline{r} : **resto** della divisione di x per β .



$$x = q \cdot \beta + r = \left\lfloor \frac{x}{\beta} \right\rfloor \cdot \beta + |x|_{\beta}$$

Proprieta' dell'operatore modulo

- Verranno usate molto frequentemente, vanno sapute
- Dato $\alpha \in \mathbb{N}^+$:

$$1) |x + \underline{k \cdot \alpha}|_\alpha = |x|_\alpha, k \in \mathbb{Z}$$

$$\underline{\frac{x}{\alpha}} + \underline{k \cdot \alpha}$$

$$k\alpha + x = \left\lfloor \frac{x}{\alpha} \right\rfloor \cdot \alpha + |x|_\alpha + k\alpha$$

$\underline{\mathbb{Z}}$

$0 \leq |x|_\alpha < \alpha$

$$\underline{x + k \cdot \alpha} = \left(\left\lfloor \frac{x}{\alpha} \right\rfloor + k \right) \cdot \alpha + |x|_\alpha$$

$$\bullet \text{ Ma: } \left\lfloor \frac{x}{\alpha} \right\rfloor + k \in \mathbb{Z}, \quad 0 \leq |x|_\alpha \leq \alpha - 1$$

- Unicità** del risultato: sono quoziente e resto della divisione per α di $x + k \cdot \alpha$

$$|2|_5 = 2$$

$$|\underline{37}|_5$$

$$35$$

$$|-16|_7$$

$$21$$

$$|5|_7 = 5$$

Proprieta' dell'operatore modulo (cont.)

$$2) \underline{|x+y|_\alpha} = \underline{||x|_\alpha + |y|_\alpha|_\alpha}$$

$$\begin{array}{c} |3+5|_2 \\ \phi \end{array} \quad \begin{array}{c} ||3_2 + 1_2|_2 \\ |1 + 1|_2 \end{array}$$

Infatti,

$$\begin{aligned} |x+y|_\alpha &= \left| \left[\frac{x}{\alpha} \right] \cdot \alpha + |x|_\alpha + \left[\frac{y}{\alpha} \right] \cdot \alpha + |y|_\alpha \right|_\alpha \\ &= \left| |x|_\alpha + |y|_\alpha + (\cancel{q_x} + \cancel{q_y}) \cdot \alpha \right|_\alpha \end{aligned}$$

Ma applicando la proprietà precedente si ottiene la tesi

Proprieta' dell'operatore modulo (cont.)

3) $|x \cdot y|_\alpha = \underbrace{| |x|_\alpha \cdot |y|_\alpha |}_\alpha$

Infatti,

$$\begin{aligned} |x \cdot y|_\alpha &= |(|x|_\alpha + q_x \cdot \alpha) \cdot (|y|_\alpha + q_y \cdot \alpha)|_\alpha = \\ &= \underbrace{| |x|_\alpha \cdot |y|_\alpha + \underbrace{(|x|_\alpha \cdot q_y \cdot \alpha)}_{\text{red}} + \underbrace{(|y|_\alpha \cdot q_x \cdot \alpha)}_{\text{red}} + q_x \cdot q_y \cdot \alpha^2 |}_\alpha \\ &= | |x|_\alpha \cdot |y|_\alpha + \underbrace{k \cdot \alpha}_{\text{red}} |_\alpha \end{aligned}$$

Algoritmo delle divisioni successive

- Data una base β , devo trovare $A \equiv (a_{n-1}a_{n-2}\dots a_1a_0)_{\beta}$, n cifre tali che
 $\rightarrow A = \sum_{i=0}^{n-1} a_i \cdot \beta^i$

- Applico iterativamente il teorema del resto

$$A \equiv \dots$$

$$\left\{ \begin{array}{l} A = q_1 \cdot \beta + a_0 \\ q_1 = q_2 \cdot \beta + a_1 \\ \dots \\ q_{n-1} = 0 \cdot \beta + a_{n-1} \end{array} \right.$$

$0 \leq a_i \leq \beta - 1$

- La n -upla di resti, letta **dal basso verso l'alto**, costituisce l'insieme di cifre che rappresentano il naturale A in base β

Correttezza dell'algoritmo

$$A = q_1 \cdot \beta + a_0$$

$$q_1 = q_2 \cdot \beta + a_1$$

$$q_2 = q_3 \cdot \beta + a_2$$

...

$$q_{n-1} = 0 \cdot \beta + a_{n-1}$$

$$A = a_0 + \beta \cdot \cancel{q_1} = a_0 + \cancel{\beta} \cdot \left(\cancel{a_1} + \beta \cdot (\cancel{a_2} + \beta \cdot (\dots)) \right)$$

- E quindi $A = \sum_{i=0}^{n-1} a_i \cdot \beta^i$

- Inoltre, il teorema della divisione con resto garantisce che la n -upla di cifre trovata è **unica**

Rappresentazione su un numero finito di cifre

- Date n cifre in base β , potrò formulare β^n sequenze di cifre differenti, e quindi rappresentare β^n numeri naturali
- Il massimo numero rappresentabile è $\beta^n - 1$
- Qual è la sua rappresentazione?
- Quella in cui tutte le cifre hanno **valore massimo**, cioè $a_i = \beta - 1$

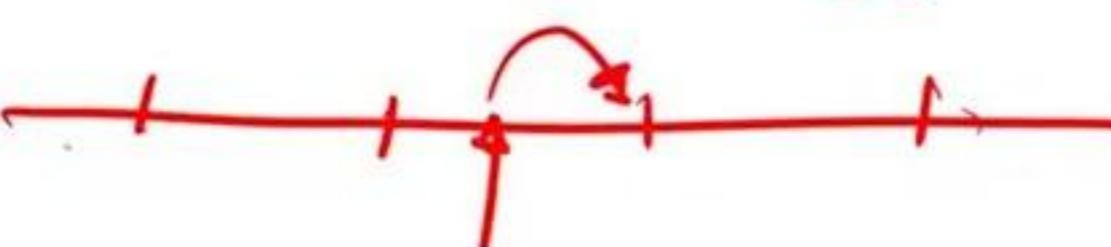
$$[0 : \beta^n - 1]$$

$$\beta = 10 \quad n = 4 \quad 9999 \quad \beta - 1$$

$$A = \sum_{i=0}^{n-1} (\underbrace{\beta - 1}_{a_i}) \cdot \beta^i = \sum_{i=0}^{n-1} \beta^{i+1} - \sum_{i=0}^{n-1} \beta^i = \left[\sum_{i=1}^n \beta^i - \sum_{i=0}^{n-1} \beta^i \right] = \beta^n - 1$$

Il **numero di cifre** necessario per rappresentare A è il minimo n per cui

$$\beta^n - 1 \geq A. \text{ Tale valore è } n = \lceil \log_\beta(A + 1) \rceil$$

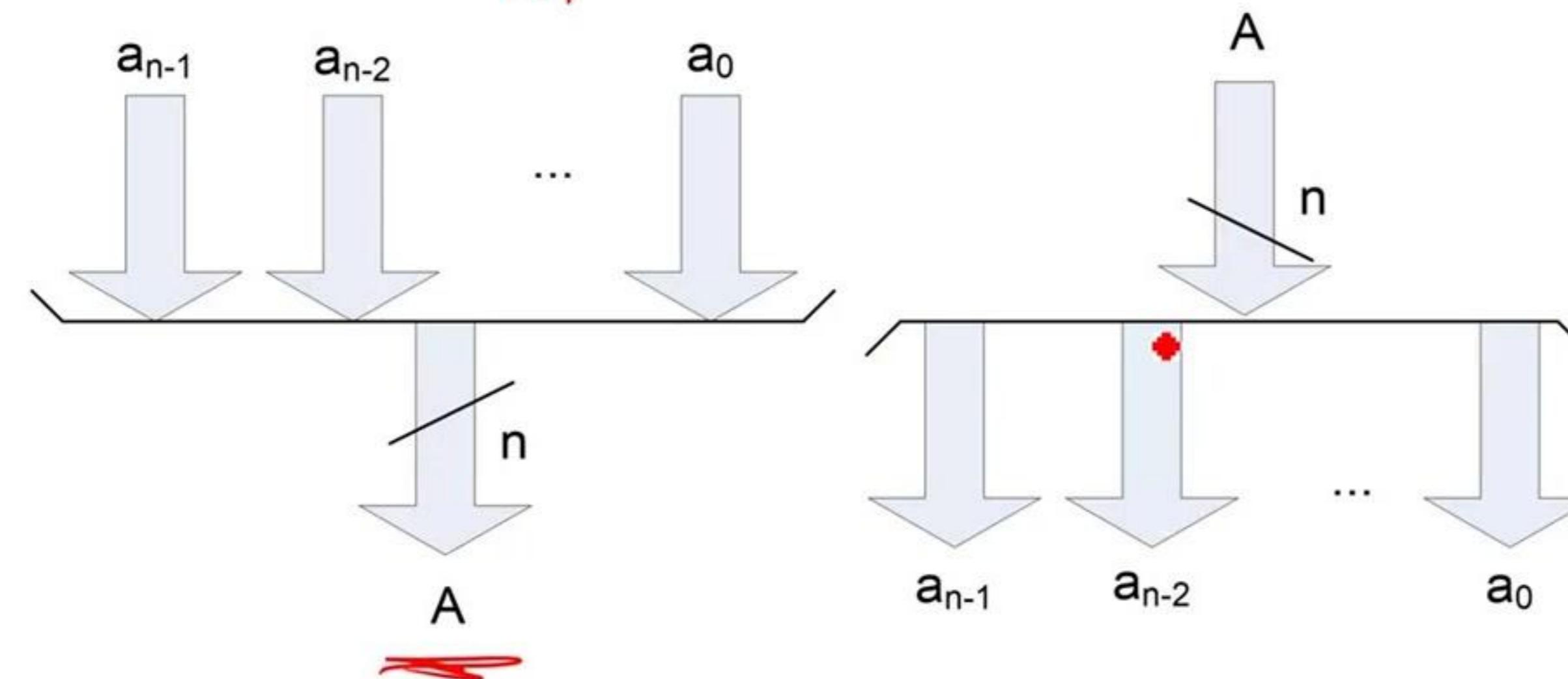
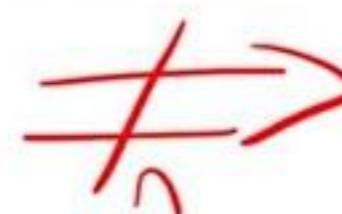


Reti combinatorie per i numeri naturali

- Dobbiamo costruire **reti logiche** che elaborino numeri **naturali** rappresentati in una data base β , generalmente pari a **due**.
- **Reti combinatorie**
 - Stato di uscita (risultato) funzione del solo stato di ingresso **presente** (operandi).
 - Per ogni operazione aritmetica di base (**somma, sottrazione, etc.**)
 - • Daremo una descrizione **indipendente dalla base**
 - Useremo le **proprietà della notazione posizionale** per scomporre l'operazione in **blocchi elementari**
 - Dettaglieremo le reti logiche (a livello di porte elementari) che implementano i blocchi elementari **in base 2**, che è la base in cui lavorano i calcolatori

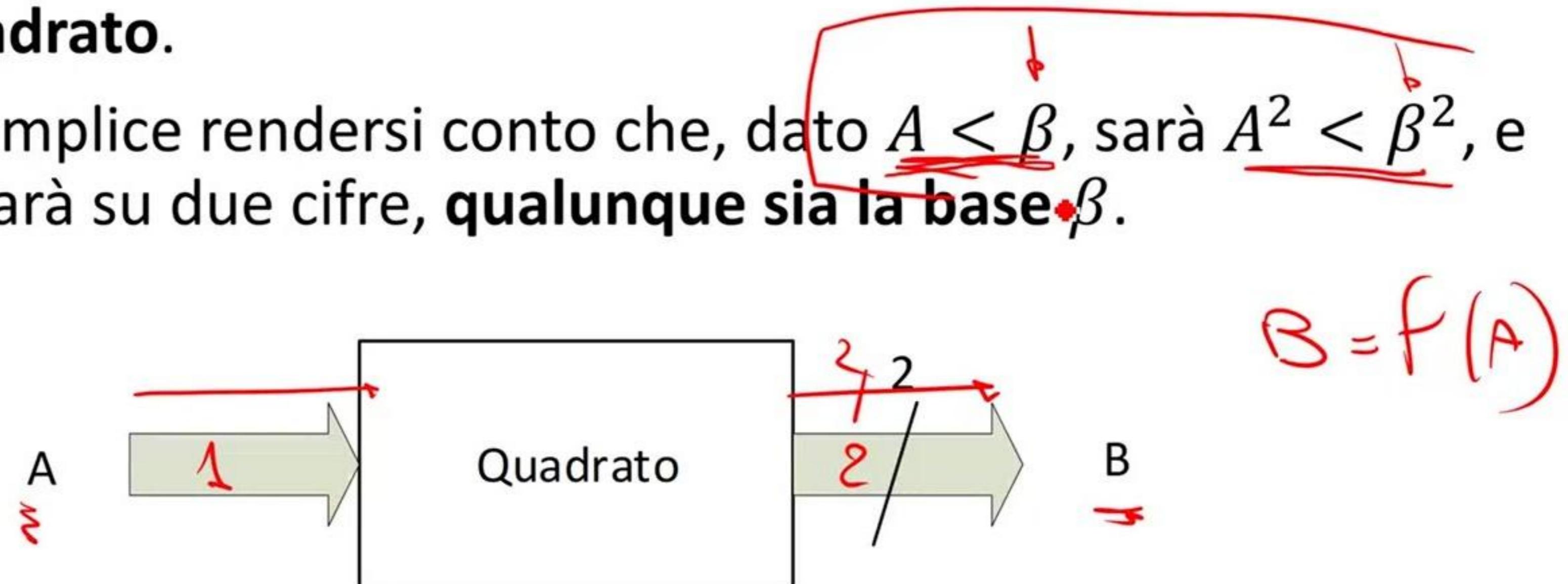
Notazione indipendente dalla base

- Le proprie' della notazione posizionale si comprendono guardando alle cifre come a componenti atomici
- Per poterlo fare, è necessario dotarsi di una notazione **indipendente dalla base**.



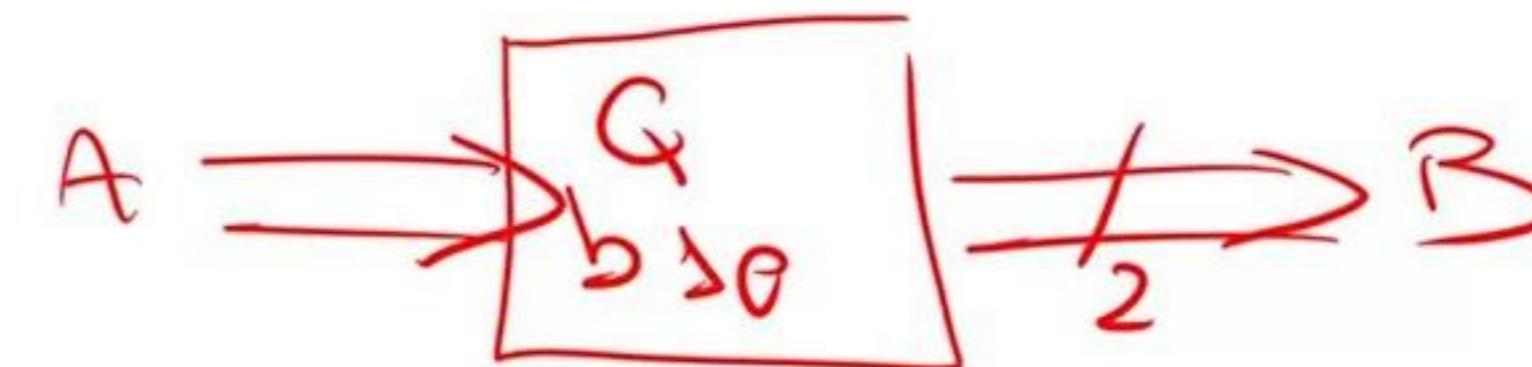
Esempio

- Rete che prende in ingresso un **numero a una cifra** in base β e restituisce in uscita il suo **quadrato**.
- È abbastanza semplice rendersi conto che, dato $A < \beta$, sarà $A^2 < \beta^2$, e quindi l'uscita sarà su due cifre, **qualunque sia la base** β .

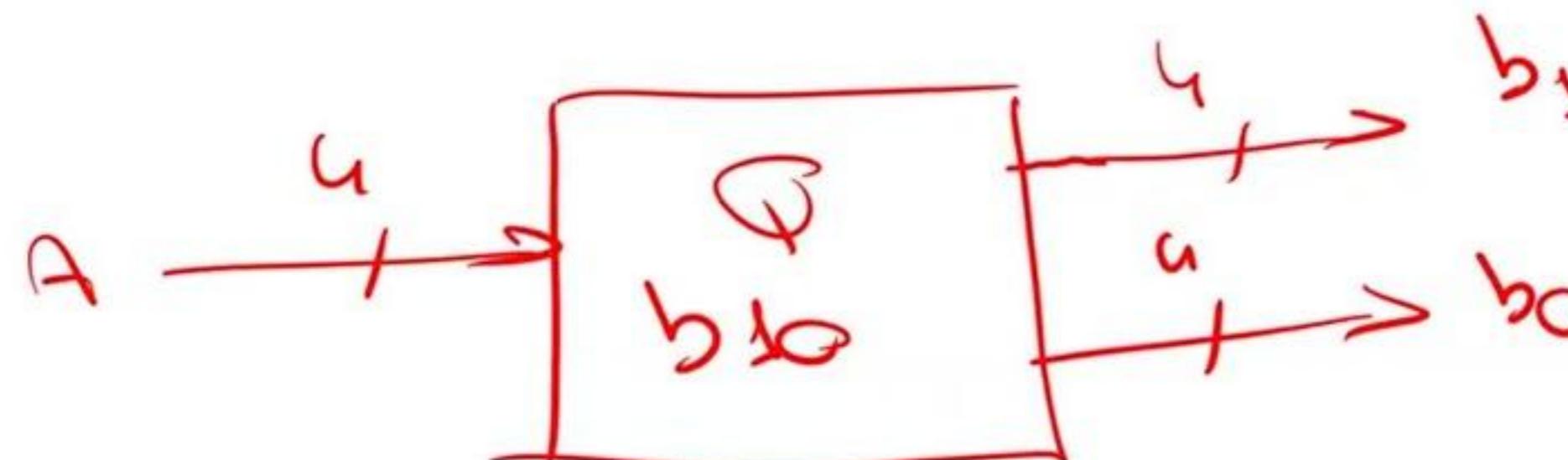


- Noi sappiamo descrivere e sintetizzare reti combinatorie che manipolano **variabili logiche**, non **cifre in una base β** generica
 - Se $\beta=2$, ok
 - Altrimenti?

Esempio (cont.)

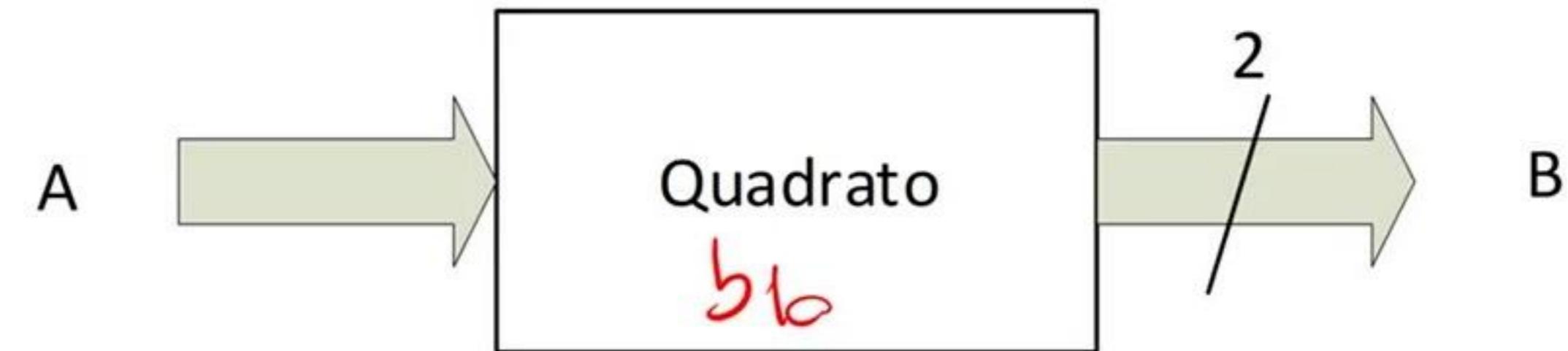


- Quando $\beta > 2$, e' necessario **codificare le singole cifre** tramite gruppi di variabili logiche
- Es: codifica BCD per la base 10



J	X ₃	X ₂	X ₁	X ₀
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Esempio (cont.)



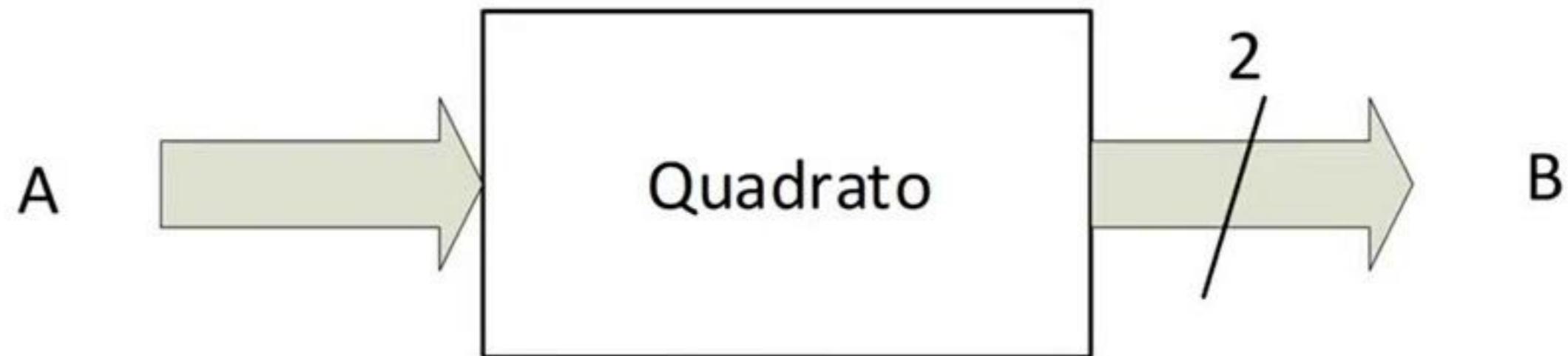
- Se la rete “quadrato” opera in base 10, la corrispondenza tra ingressi e uscite è:

A	$b_1 b_0$
0	00
1	01
2	04
...	...
8	64
9	81

BCD

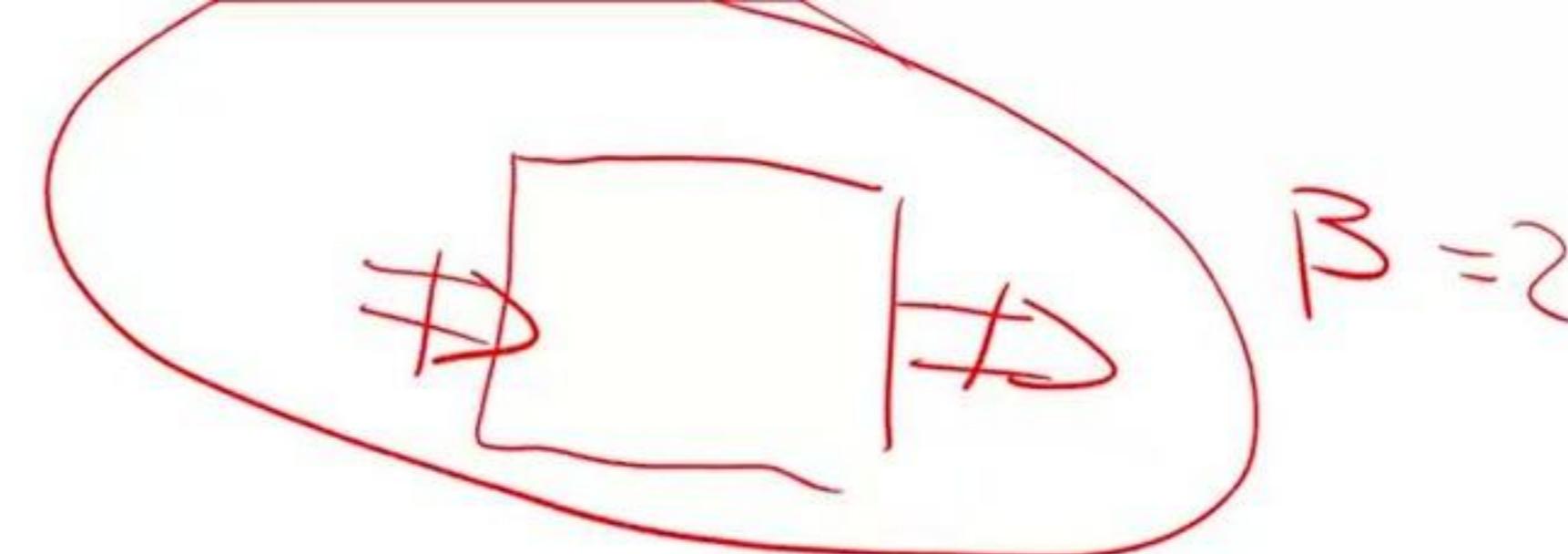
BCD(A)	BCD(b_1)	BCD(b_0)
0000	0000	0000
0001	0000	0001
0010	0000	0100
...
1000	0110	0100
1001	1000	0001

Esempio (cont.)



- Questa rete **opera in base 10**
 - Anche se in ingresso e in uscita ho variabili logiche
- Infatti, restituisce in uscita 1000 0001 (81) quando l'ingresso è 1001 (9)
- Questa è la risposta corretta solo se si interpretano le variabili logiche di uscita come **codifiche di cifre in base 10**.
- In base 2, il quadrato di 1001 è 1010001

Relazioni tra cifre



- Nel seguito, ci preoccuperemo molto poco delle codifiche, e lavoreremo sulle **cifre**.
- Infatti, la maggior parte delle operazioni aritmetiche si possono esprimere in ~~termini~~ di **relazioni tra cifre degli operandi**, indipendentemente dalla base in cui si lavora.
- Parleremo di codifica soltanto quando dovremo **sintetizzare le reti combinatorie**, ma in questo caso lavoreremo quasi sempre in base 2, dove una cifra è anche una variabile logica.

Complemento

- Dato $A \equiv (a_{n-1}a_{n-2}\dots a_1a_0)_{\beta}$, in base β su n cifre, $0 \leq A < \underline{\beta^n}$
- Definisco **complemento** di A (in base β su n cifre) il numero:

9999 - 1034

$$\bar{A} \triangleq (\cancel{\beta^n} - 1) - A$$

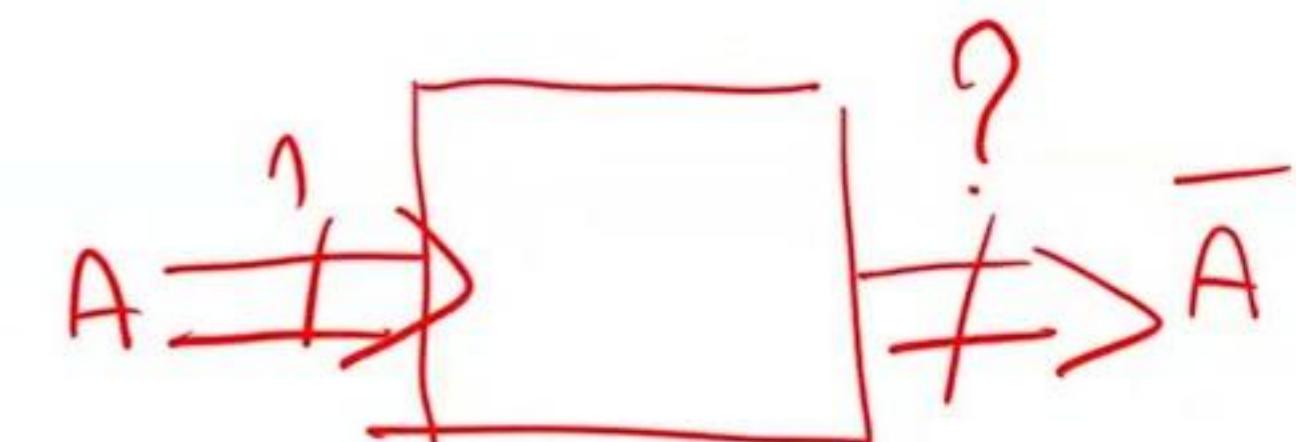
→ • $(\overline{1034})_{10} = \underline{(8965)}_{10}$

→ • $(1034)_5 = (3410)_5$

→ • $(\overline{001034})_{10} = \underline{(998965)}_{10}$

4444 - 1034

999999 - 001034



$$A \equiv (a_{n-1} \dots a_0)_B$$

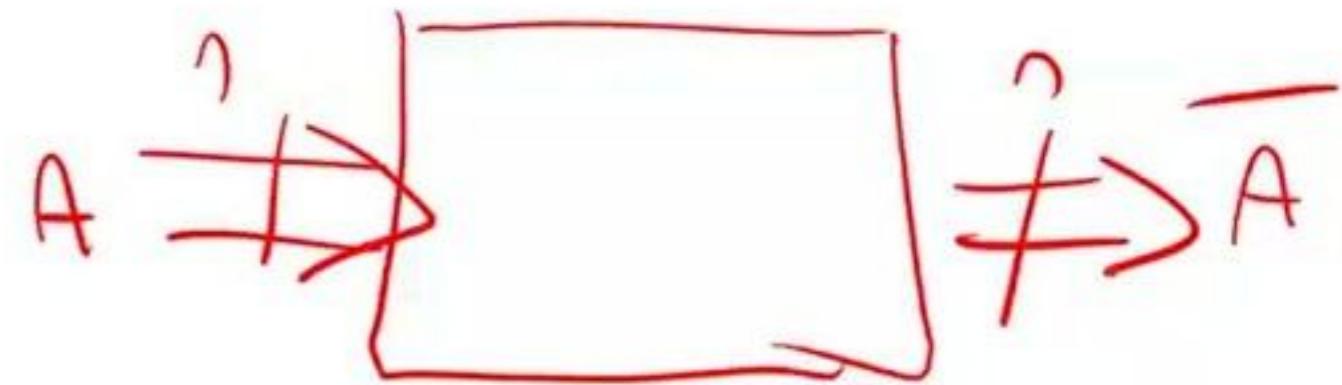
$$A = \sum_{i=0}^{n-1} a_i \cdot \beta^i$$

Rete combinatoria per il complemento

- **prima domanda: su quante cifre sta il risultato**

$$\overline{A} \triangleq \beta^n - 1 - A$$

- Se $0 \leq A < \beta^n$, allora $0 \leq \overline{A} < \beta^n \Rightarrow$ sta su n cifre.



- **seconda domanda: come si trovano le cifre del risultato** a partire da quelle degli operandi

$$\overline{A} = \beta^n - 1 - A = \sum_{i=0}^{n-1} (\beta - 1) \beta^i - \sum_{i=0}^{n-1} a_i \beta^i = \sum_{i=0}^{n-1} (\beta - 1 - a_i) \beta^i$$

Rete combinatoria per il complemento (cont.)

$$a_i = \emptyset \rightarrow \cancel{\beta^{-1}}$$

$$a_i = \beta^{-1} \rightarrow \cancel{\phi}$$

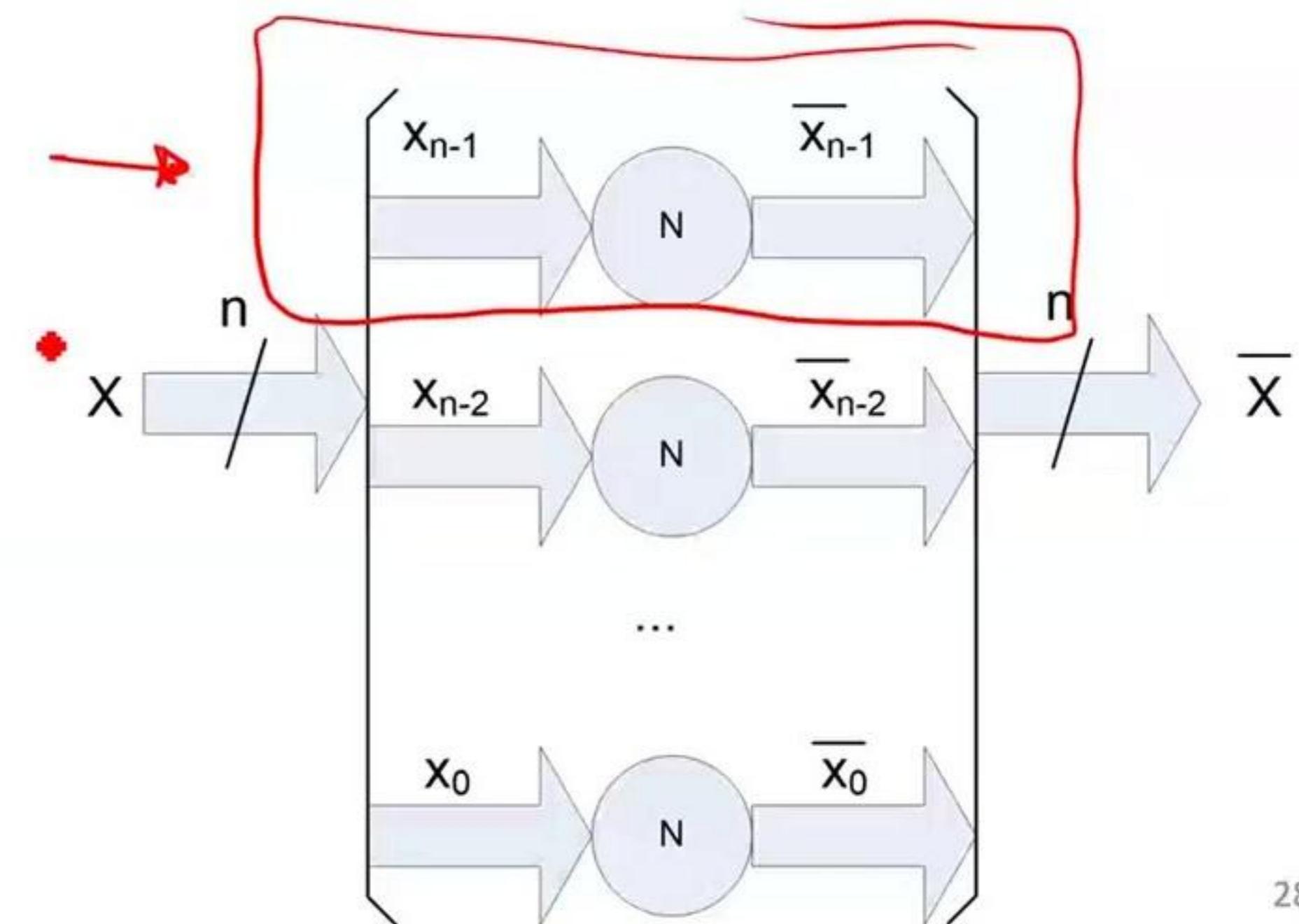
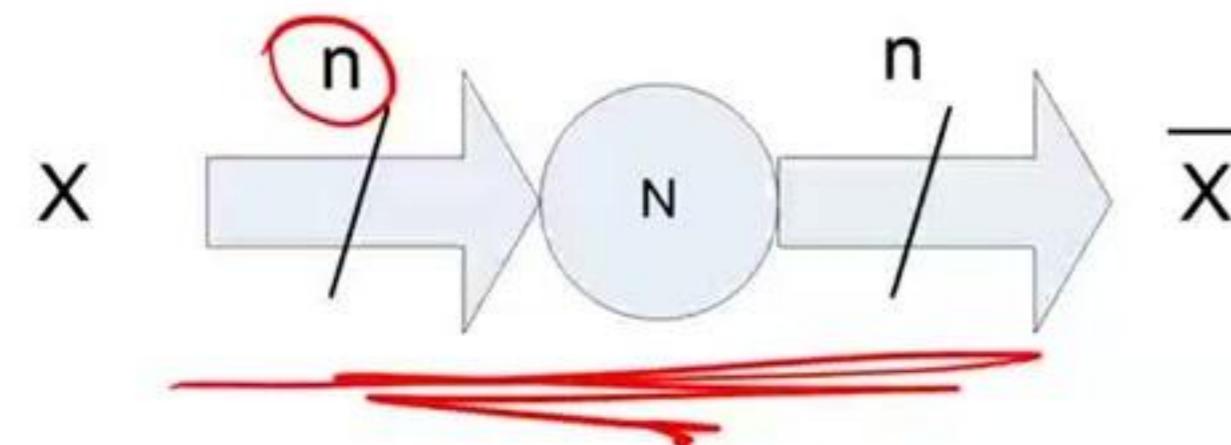
$$\bar{A} = \sum_{i=0}^{n-1} (\cancel{\beta - 1} - a_i) \beta^i$$

$$\begin{array}{r} \overline{8932} \\ - 1067 \\ \hline \end{array} = 1067$$

- $\beta - 1 - a_i$ è una cifra in base β , in quanto compresa tra 0 e $\beta - 1$.
- Dalla definizione, $\beta - 1 - a_i \triangleq \bar{a}_i$
- Quindi, $\bar{A} \equiv (\overline{a_{n-1}} \ \overline{a_{n-2}} \ \dots \ \overline{a_1} \ \overline{a_0})_\beta$
 - Questa proprietà vale in qualunque base β . |
- Posso usare questa proprietà per sintetizzare uno schema di circuito che funziona in qualunque base

Rete combinatoria per il complemento (cont.)

- Posso scomporre il complemento di un numero a n cifre in una serie di complementi di **una singola cifra**
- La cosa piu' complessa che devo saper fare e' il complemento di una singola cifra



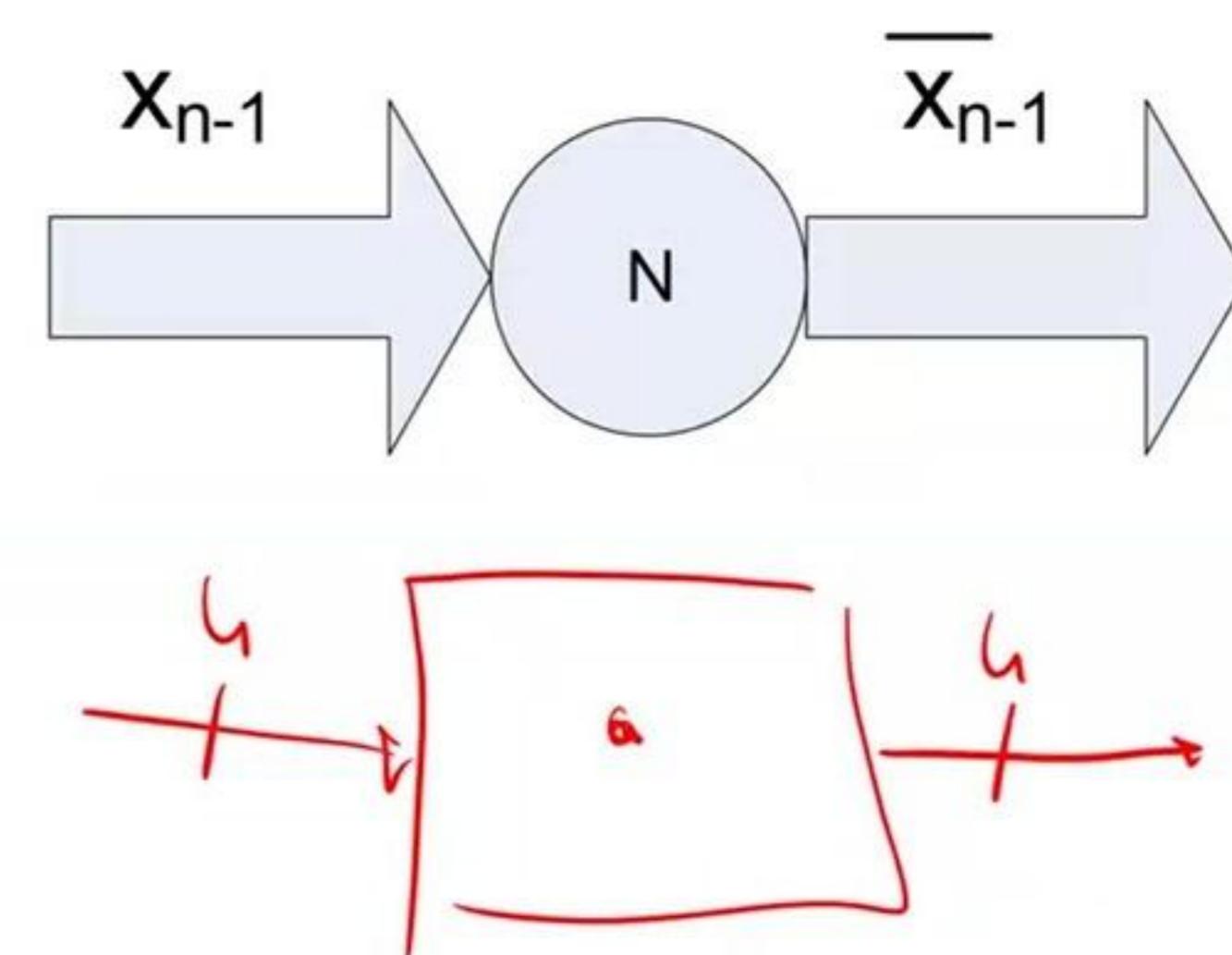
Rete combinatoria per il complemento (cont.)

- Come si fa il complemento di una singola cifra?
 - Dipende dalla base (e dalla codifica)

$$\begin{array}{ccc} 0 & \xrightarrow{\quad} & 1 \\ & \downarrow & \downarrow \\ & 1 & 0 \end{array}$$

- Base 2: variabile logica \Leftrightarrow cifra
 - porta NOT

- Base 10, codifica BCD
 - Circuito con 4 ingressi e 4 uscite
 - Descrivere e sintetizzare per casa



0000
0001

⋮

1001
1000

⋮

Mul/Div per potenze della base

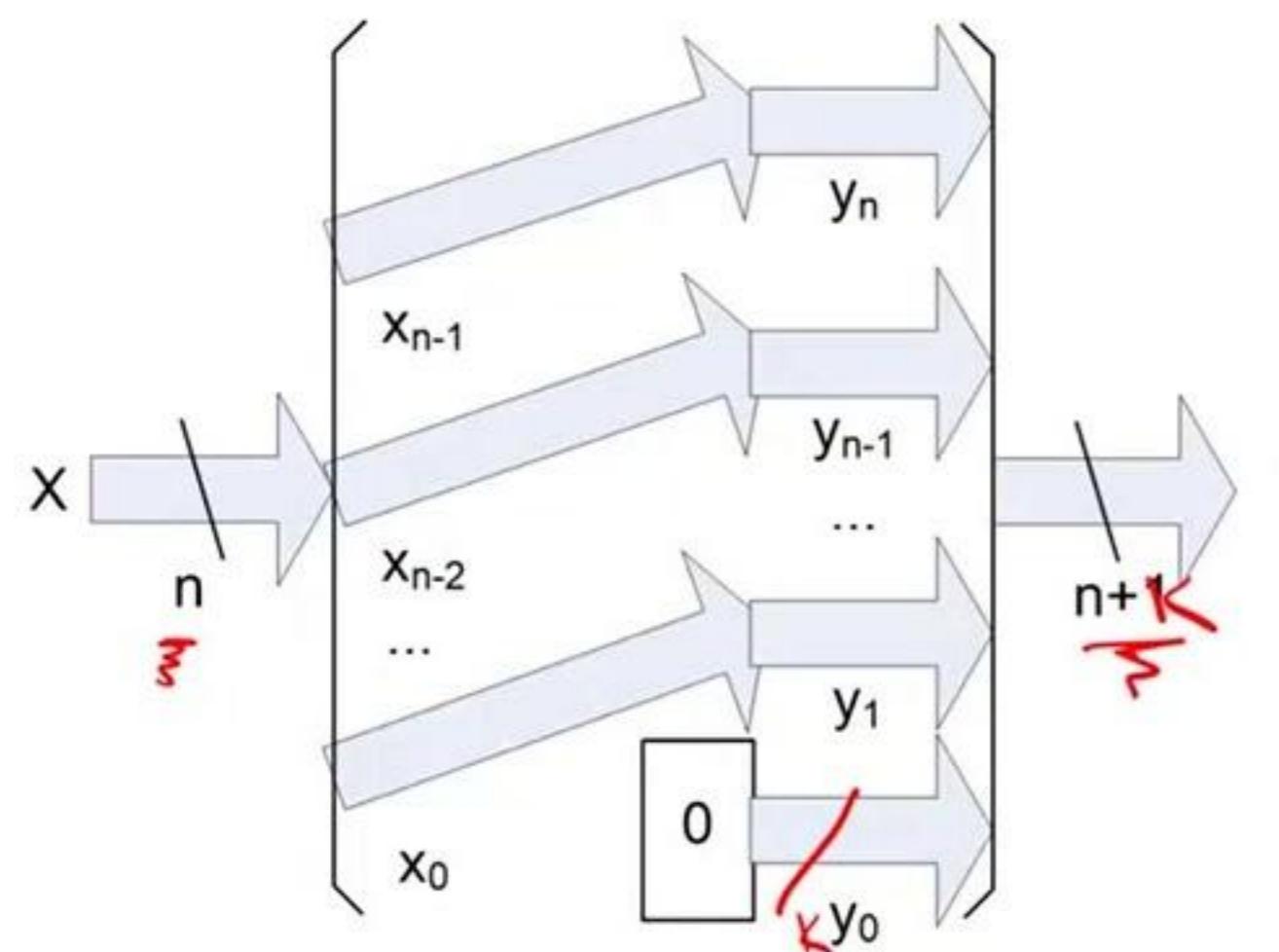
- Si tratta di aggiungere/togliere cifre
 - $25 \times 1\textcolor{red}{000} = 25\textcolor{red}{000}$ \backslash_0^3
 - $2562 \div 1\textcolor{red}{00}$ $= 25, r = 62$
- Valido in qualunque base, e' una propriet'a della notazione posizionale
- In base β , moltiplicare e dividere per β^k sono operazioni di costo nullo
 - Consistono nell'aggiungere zeri e separare gruppi di cifre
- Se il costo «mentale» e' nullo, e' abbastanza probabile che la rete che esegue questa operazione sia **priva di logica**



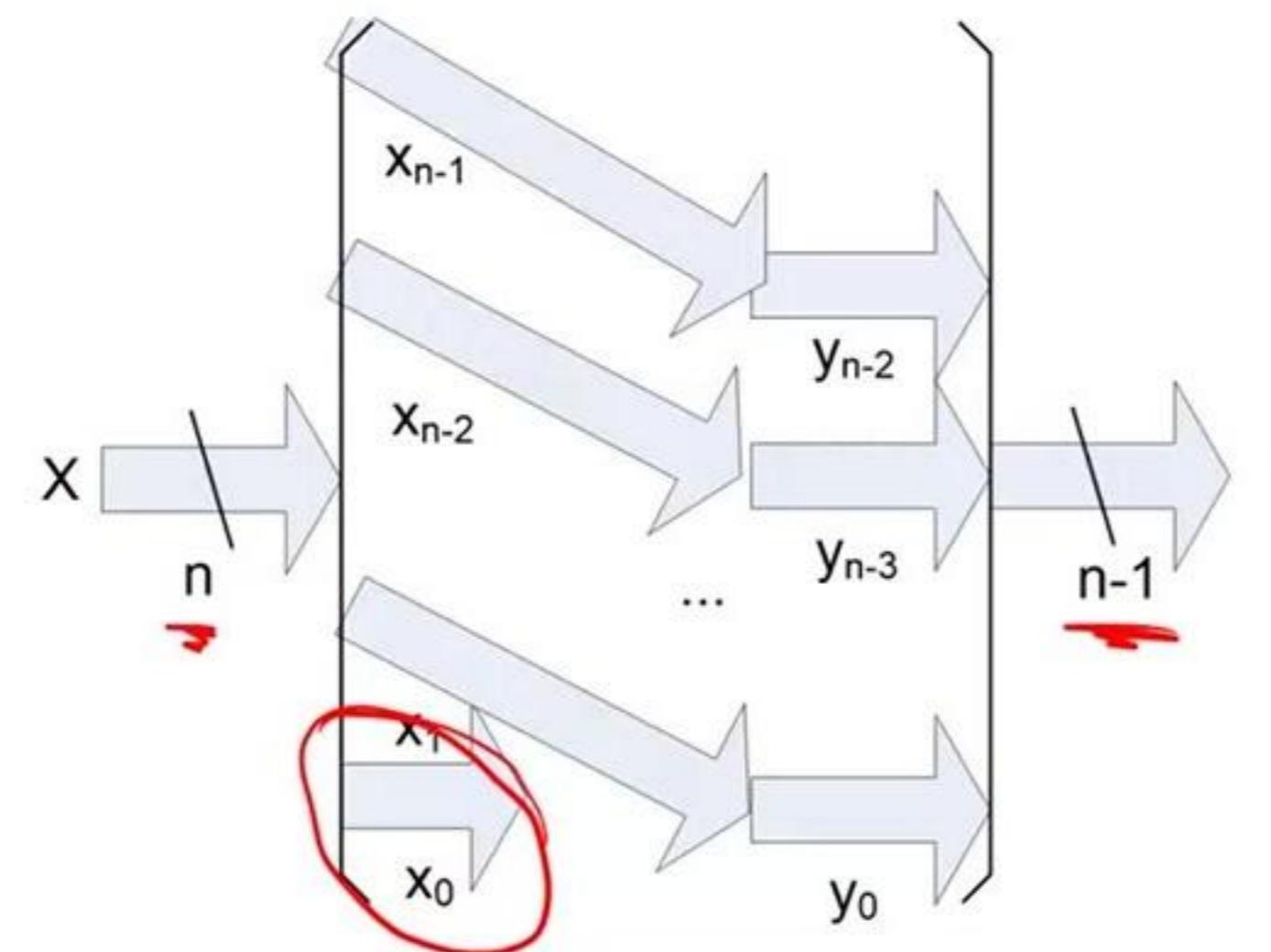
Mul/Div per potenze della base

$a_m \dots a_0 \neq 0$

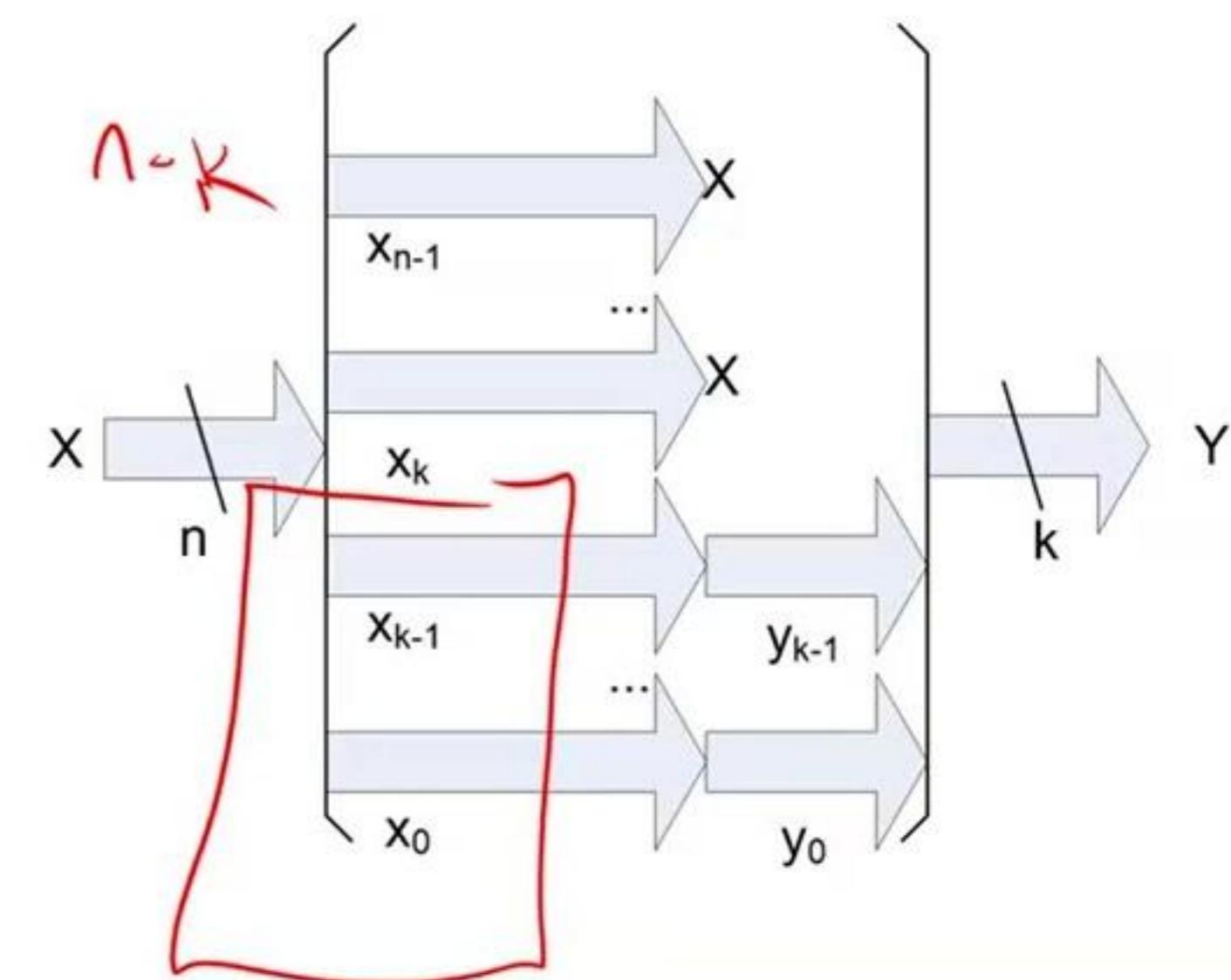
- Prodotto per β^k



- Quoziente $\div \beta^k$



- Resto $\div \beta^k$



- **complessità nulla**, così come è di complessità nulla il procedimento mentale per il calcolo del risultato

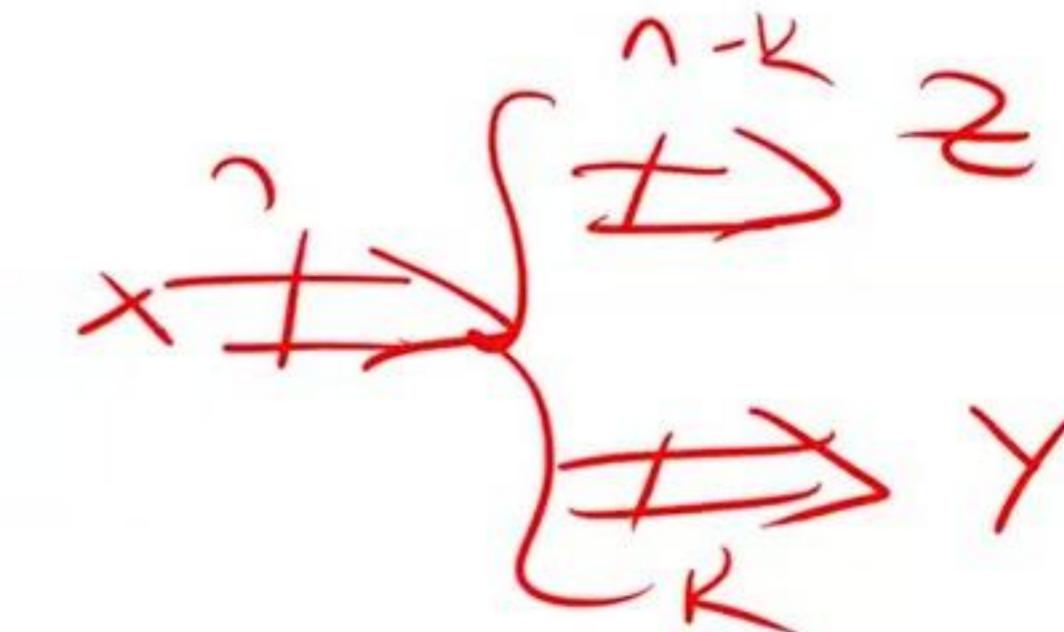
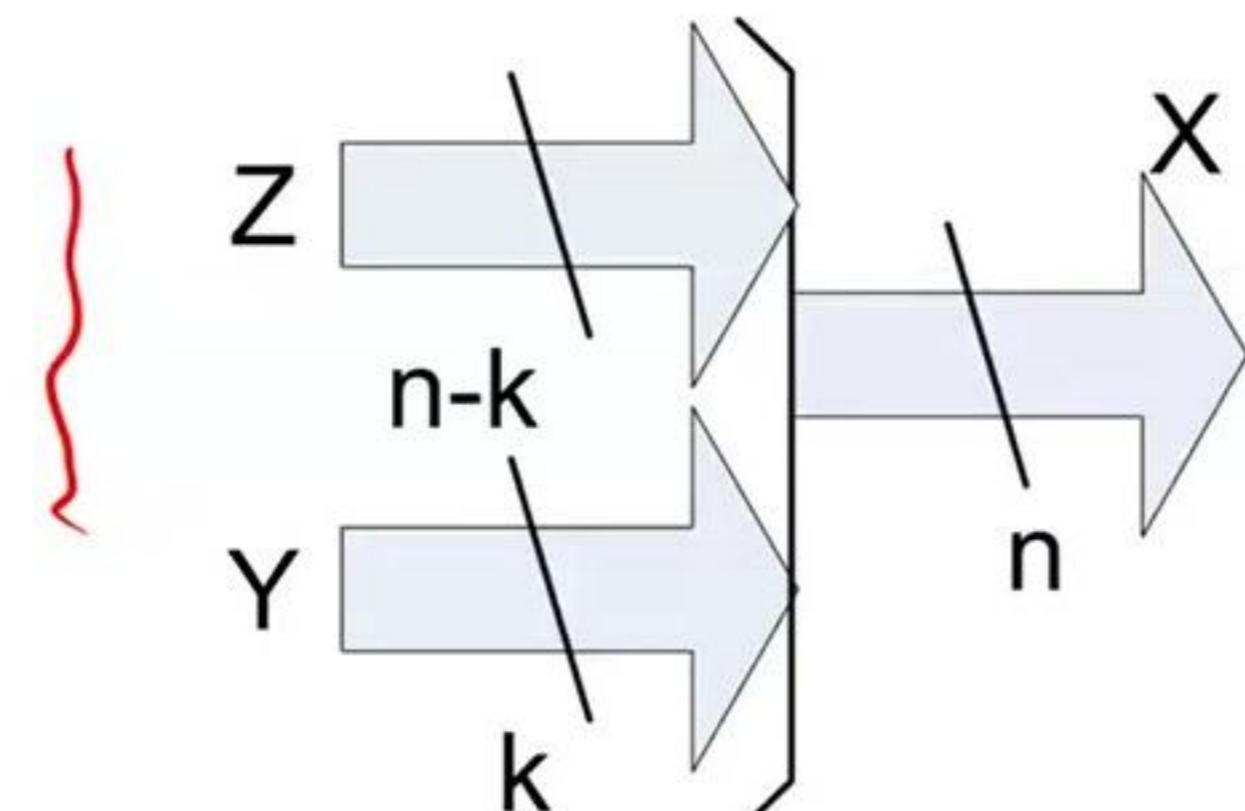
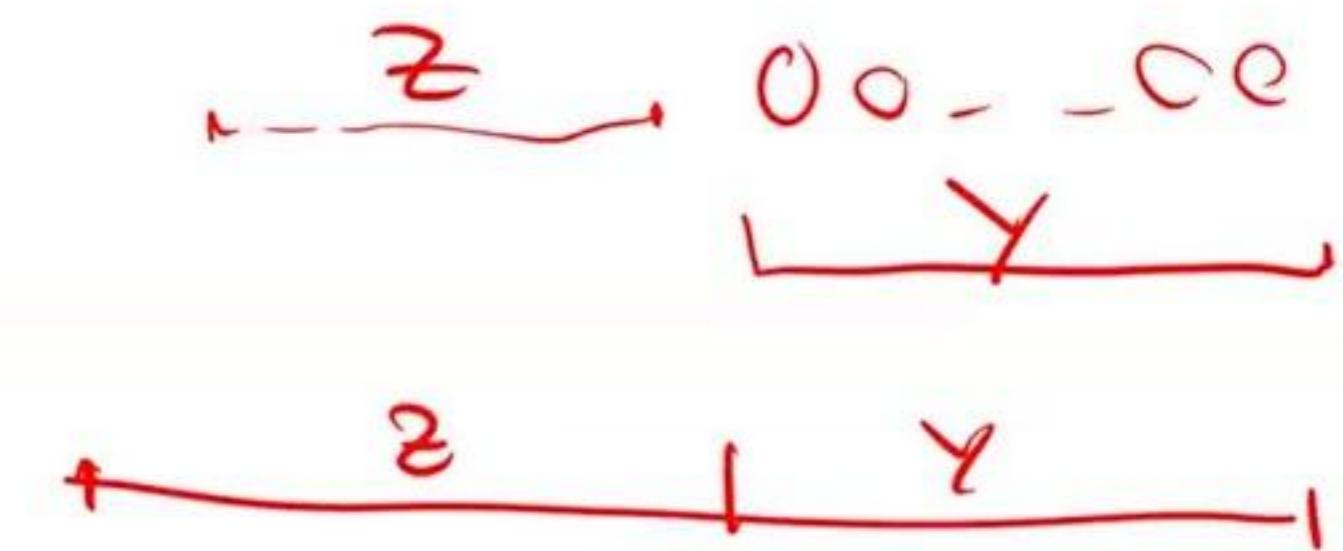
Una conseguenza non ovvia (ma importante)

↖ ↗ ↖

- Dati due numeri Z e Y , rispettivamente a k ed $n - k$ cifre,
- L'operazione di concatenamento

$$\rightarrow \boxed{X = Z \cdot \beta^k + Y}$$

- produce un numero su n cifre
- è di **complessità nulla**.



Estensione di campo

$$x'_i = x_i \quad x'_i = 0 \quad \rightarrow$$

$$\sum_{i=0}^{n-1} x'_i \beta^i = \sum_{i=0}^{n-1} x'_i \cdot \beta^i$$

- l'operazione con la quale si rappresenta un numero naturale su un **numero di cifre maggiore**.

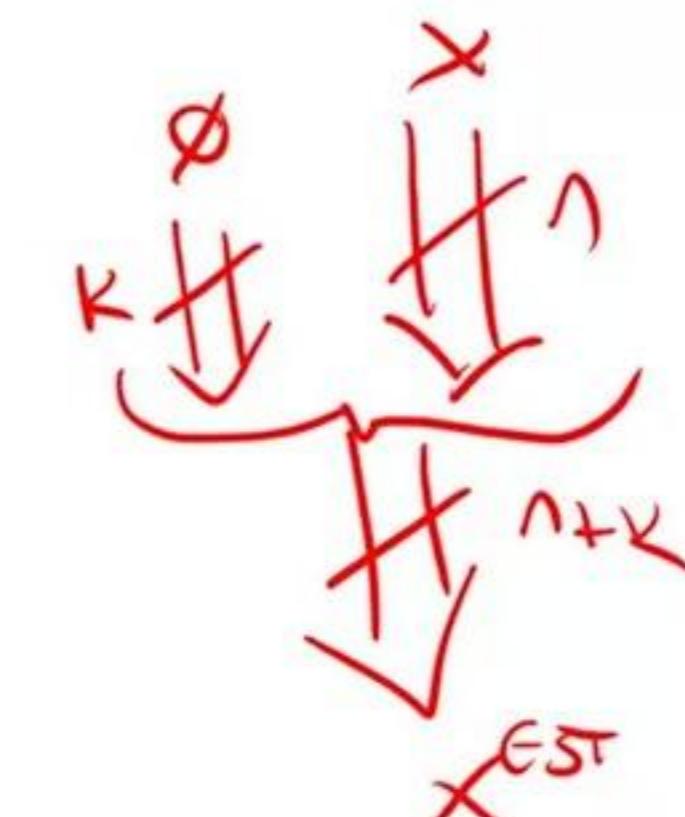
- Per scrivere 32 su 4 cifre, si mettono **due zeri in testa**.

- La stessa cosa si fa in qualunque base con i numeri **naturali** (**attenzione**: per gli interi sarà diverso).

$$X \equiv (x_{n-1} x_{n-2} \dots x_0)_{\beta}$$

$$X^{EST} = X, \text{ ma su } n+1 \text{ cifre}$$

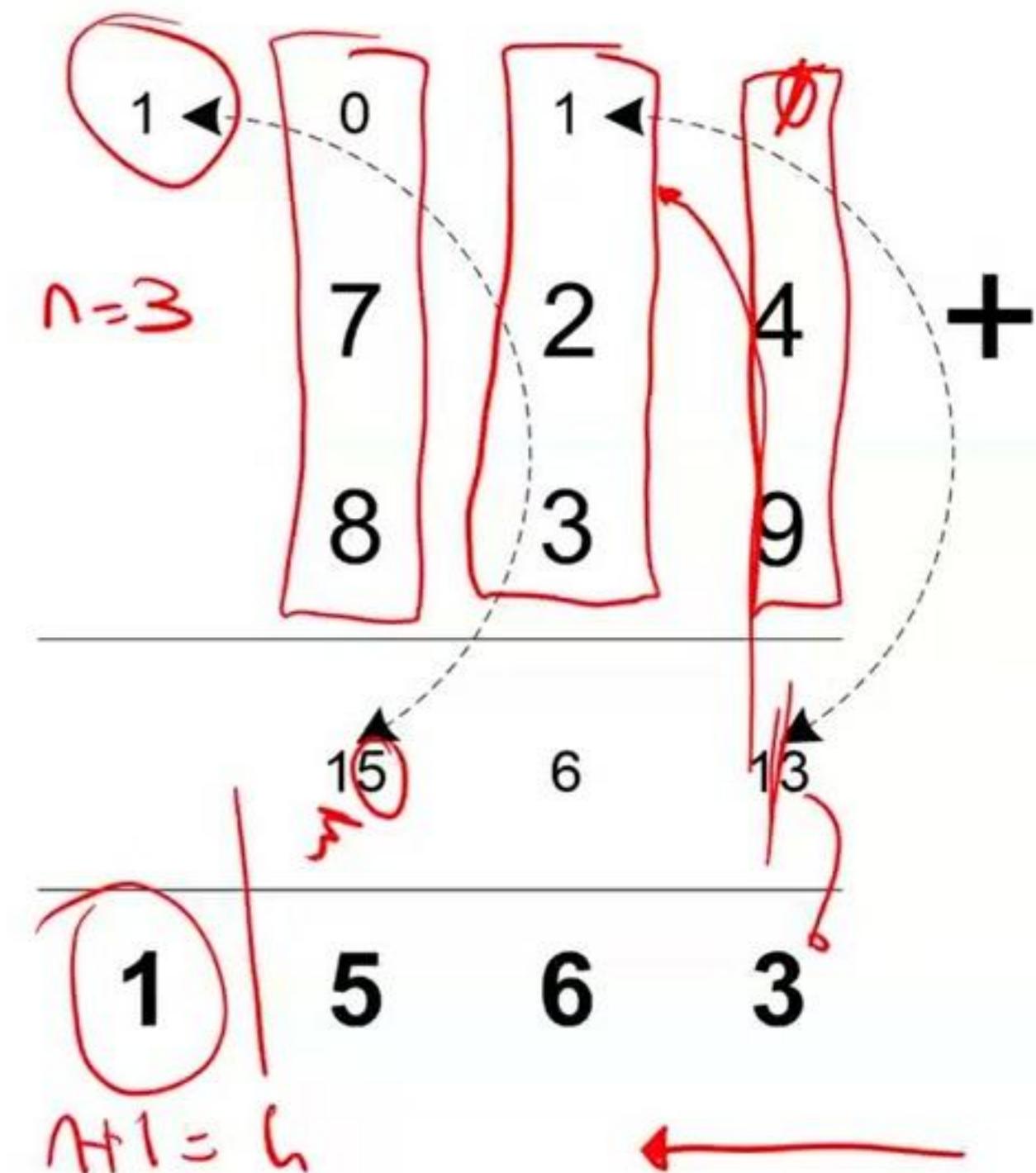
$$X^{EST} \equiv (0 \ x_{n-1} x_{n-2} \dots x_0)_{\beta}$$



Addizione

ADD

- $\beta = 10$. L'algoritmo consiste in:
- sommare le cifre **di pari posizione** singolarmente, partendo dalla meno significativa, andando verso sinistra;
- se la somma di due cifre non è rappresentabile con una sola cifra, usare il **riporto** per la coppia di cifre successive.
- Il riporto vale sempre 0 o 1. Per la prima coppia di cifre (quelle meno significative), possiamo assumerlo nullo.
- Algoritmo non dipende dalla base di rappresentazione, ma soltanto dal fatto che usiamo una **notazione posizionale**.



Addizione (cont.)

$< \beta^n$

- Dati X, Y in base β su n cifre, quindi $0 \leq X, Y \leq \beta^n - 1$
- Dato C_{in} , $0 \leq C_{in} \leq 1$
- Voglio calcolare:

$$Z = X + Y + C_{in}$$

$\leq \beta^n - 1$

- Su quante cifre sta il risultato?

$$0 \leq X + Y + C_{in} \leq 2\beta^n - 1 \leq \beta^{n+1} - 1$$

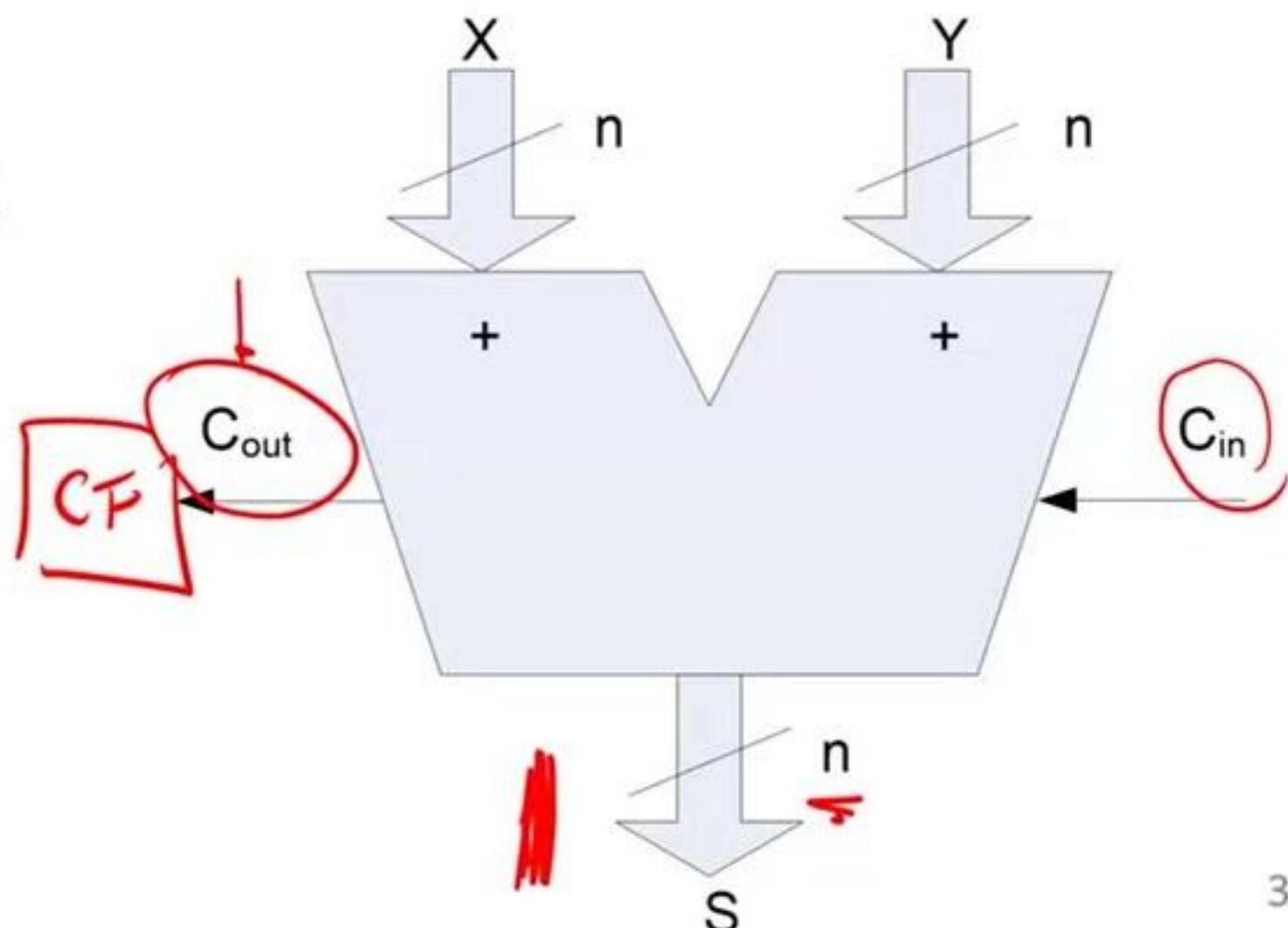
- Z è rappresentabile **sempre** su $n + 1$ cifre
- Potrebbe **non essere rappresentabile** su n cifre.
- La $n + 1^{\text{ma}}$ cifra è il riporto dell'ultima somma, e quindi può essere soltanto 0 o 1. La chiamo C_{out}

Carry

Addizione (cont.)

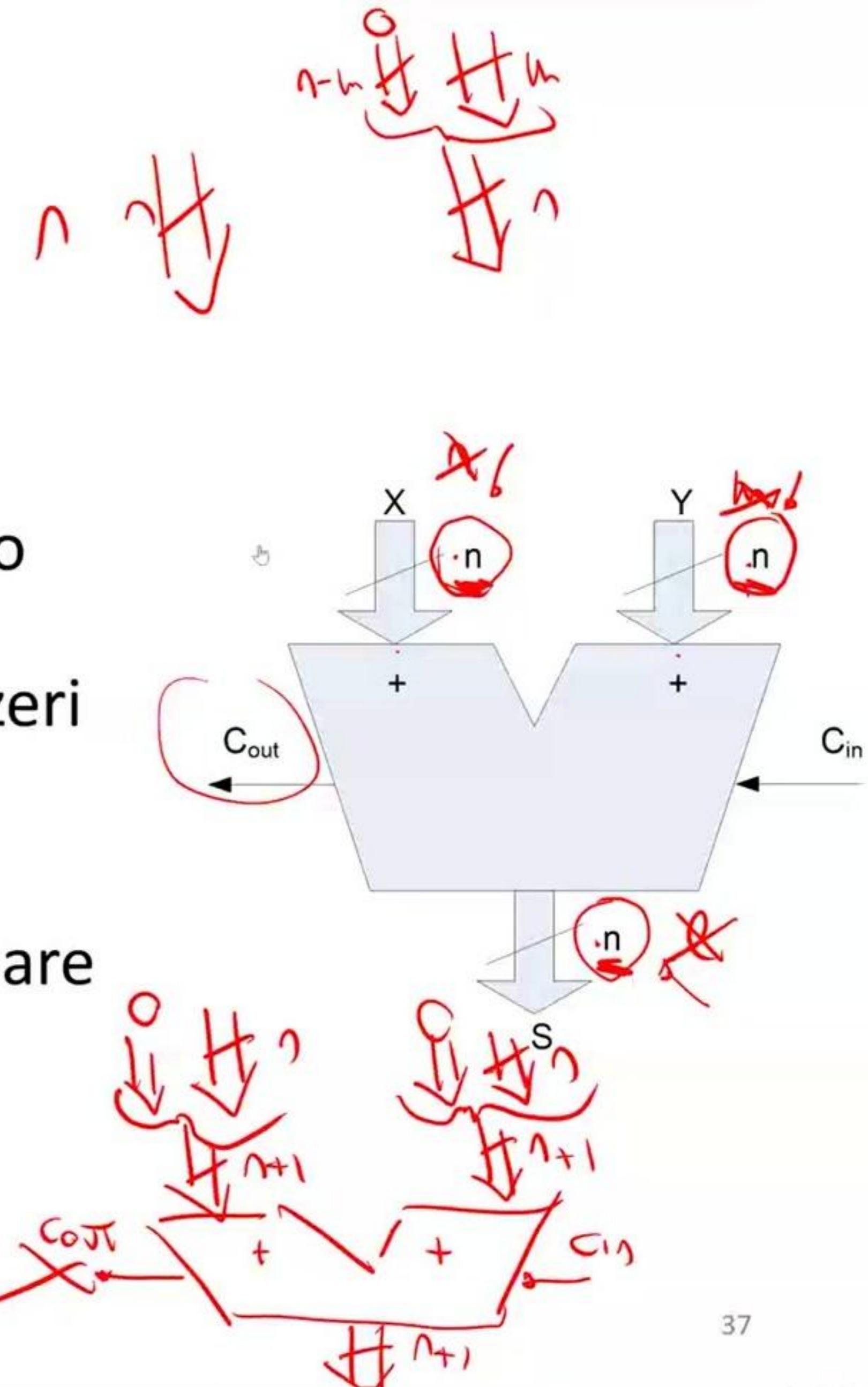
- La somma di due numeri naturali espressi in base β su n cifre, più un eventuale riporto entrante (0 o 1), produce un numero naturale che è sempre rappresentabile con $n + 1$ cifre in base β , l' $(n + 1)^{\text{ma}}$ delle quali, detta riporto uscente, può essere soltanto zero o uno.
- Circuito sommatore in base β a n cifre

- $C_{out} = 0$, la somma è rappresentabile su **n cifre**, cioè sul # di cifre degli operandi;
- $C_{out} = 1$ la somma **non è rappresentabile su n cifre**, ma ce ne vuole una in più.



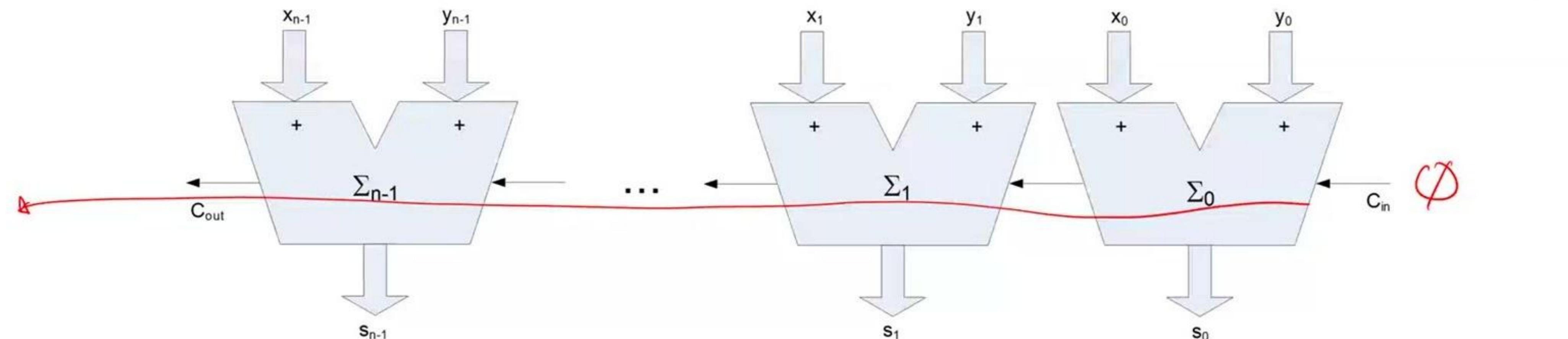
Dimensionamento

- il numero di cifre di **entrambi gli ingressi e dell'uscita è lo stesso** (altrimenti è **errore**).
- se gli addendi sono su un numero di cifre diverso (ad esempio, n ed m cifre, con $n > m$, dovete **estendere quello ad m cifre** (inserendo $n - m$ zeri in testa);
- se gli addendi sono su **n cifre** e volete che la somma S sia **sempre** rappresentabile, dovete usare un sommatore ad **$n + 1$ cifre**, ed estendere gli ingressi su **$n + 1$ cifre**.



Ripple carry e full adder

- La somma in base β su n cifre può essere scomposta in somme in base β su una sola cifra, purché:
 - esegua le somme dalla cifra meno significativa alla più significativa
 - propaghi il riporto tra uno stadio di somma ed il successivo



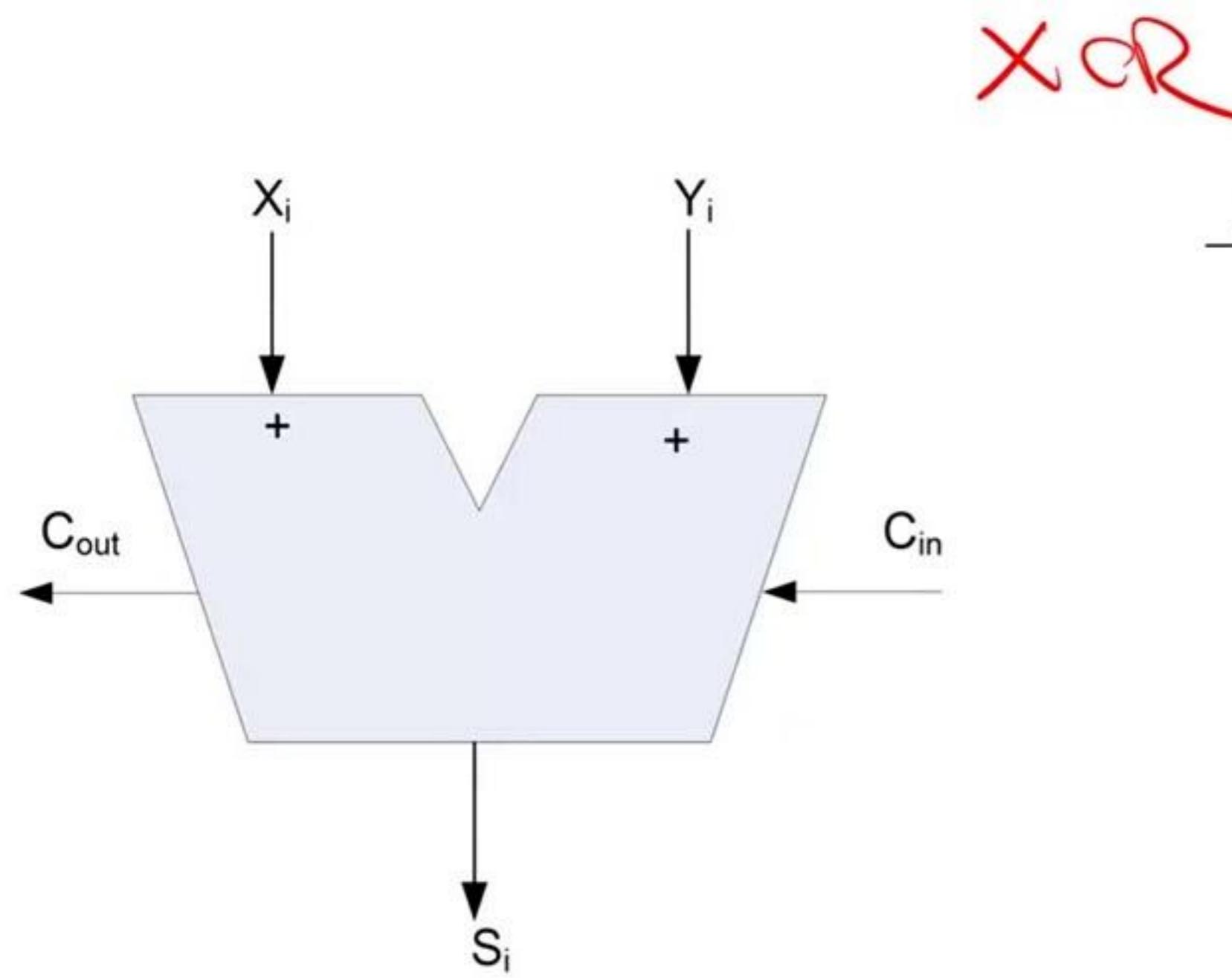
- Montaggio ripple carry
- Ciascuno stadio si chiama **full adder** (somma di addendi a una cifra)

Full adder in base 2

$$\begin{array}{r} \emptyset \quad q \\ \emptyset + \\ \emptyset \\ \hline \emptyset \end{array}$$

$$C_{out} = X_i \cdot Y_i + Y_i \cdot C_{in} + X_i \cdot C_{in}$$

- Rete combinatoria con 3 ingressi e 2 uscite



x_i	y_i	c_{in}	s_i	c_{out}
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

The figure shows two Karnaugh maps for the XOR function. The top map shows the minterms $A=1$, $B=1$, and $C=1$ highlighted with dashed boxes. The bottom map shows the minterms $S_i=1$ highlighted with a dashed box.

$x_i \backslash y_i$	00	01	11	10
0	0	0	1	0
1	0	1	1	1

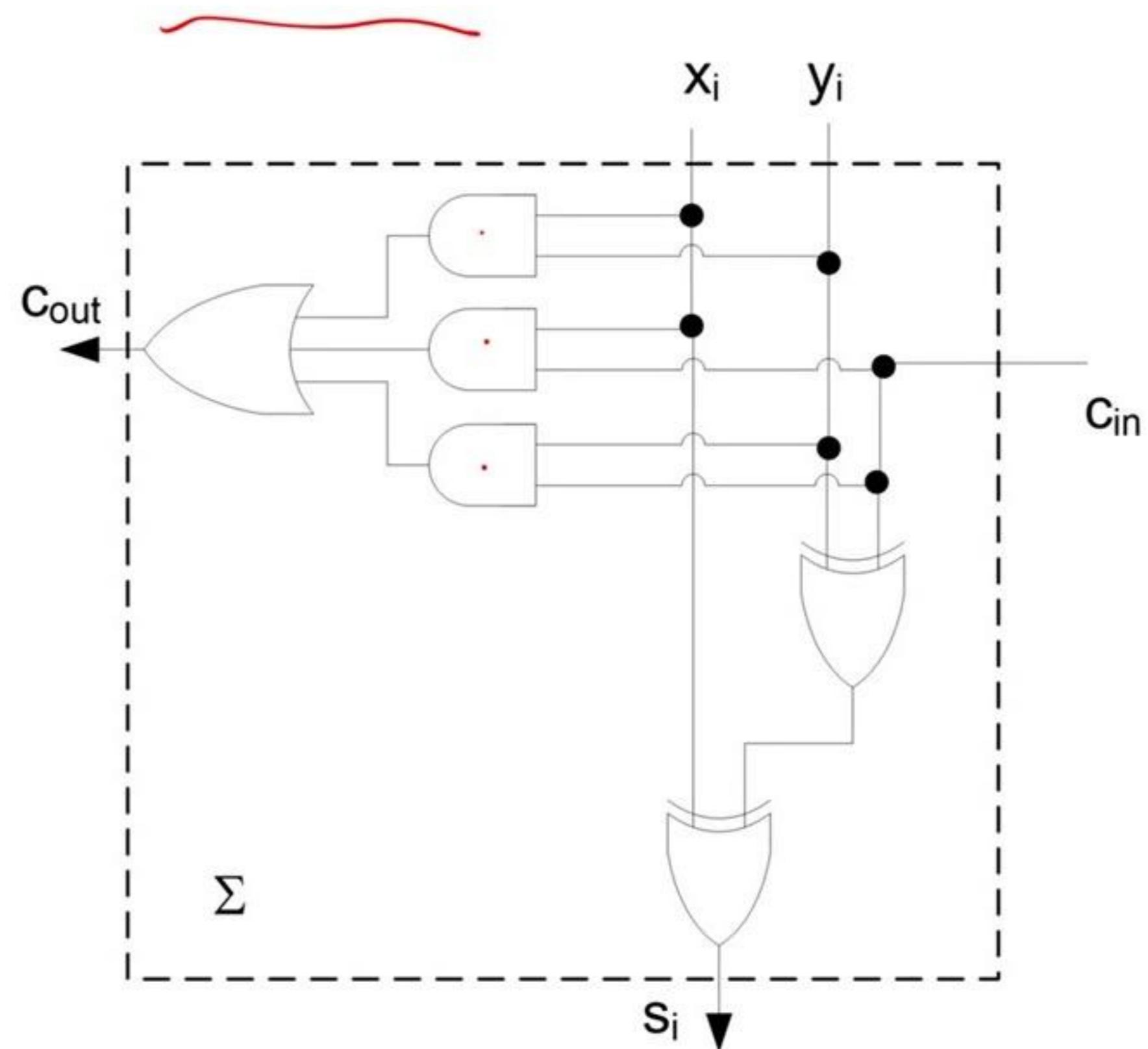
$x_i \backslash y_i$	00	01	11	10
0	0	1	0	1
1	1	0	1	0

Full adder in base 2 (cont.)

- Riconoscere un # **dispari** di 1
 - Porte XOR in cascata

	$x_i y_i$	00	01	11	10	s_i
0		0	1	0	1	
1		1	0	1	0	

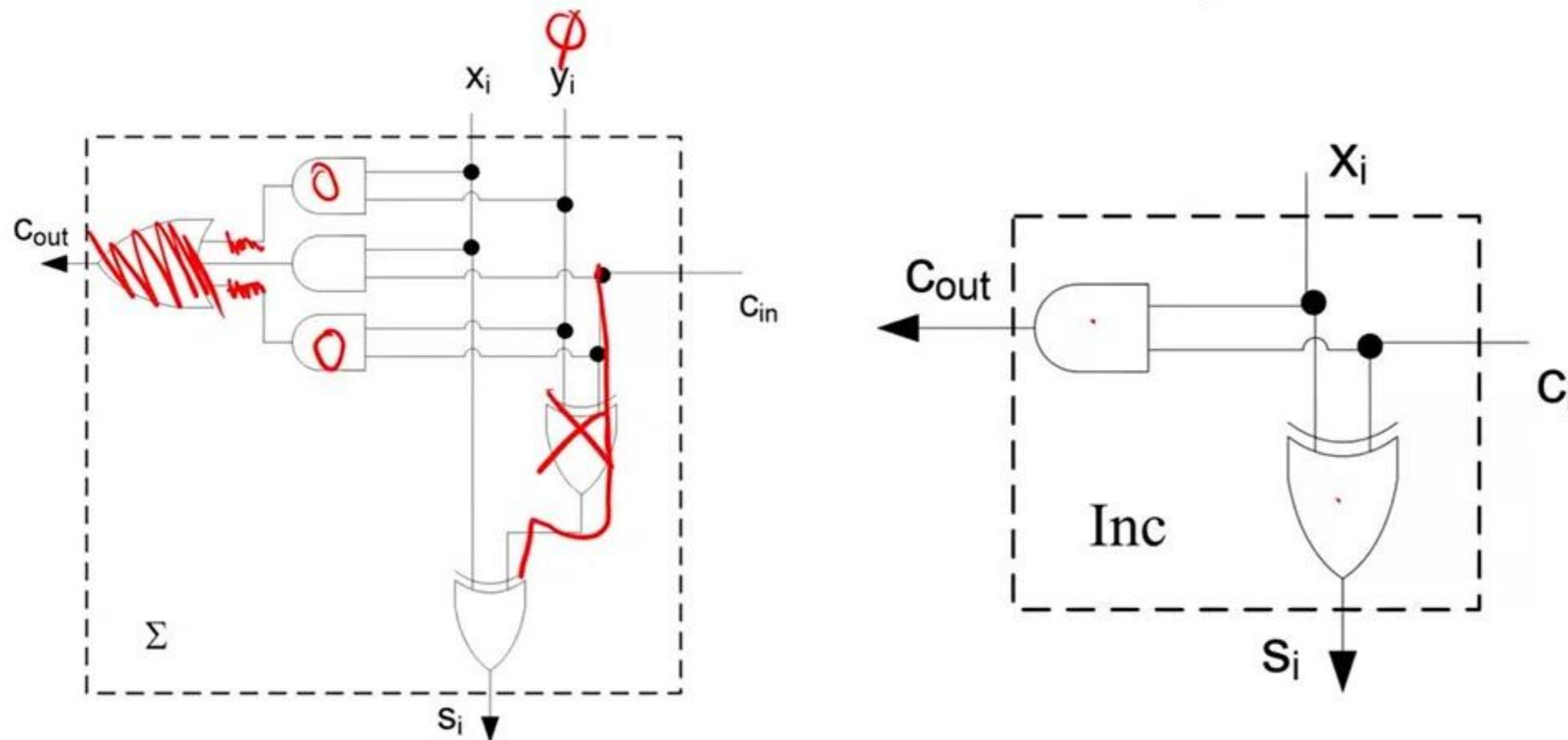
21L



Incrementatore

half adder

- Circuito che somma C_{in} (che vale 0 o 1) ad un numero dato
- Caso particolare di somma, uno degli addendi e' nullo
- Stessa scomposizione ripple-carry
- Half adder si ottiene da full adder, mettendo a 0 un operando e semplificando



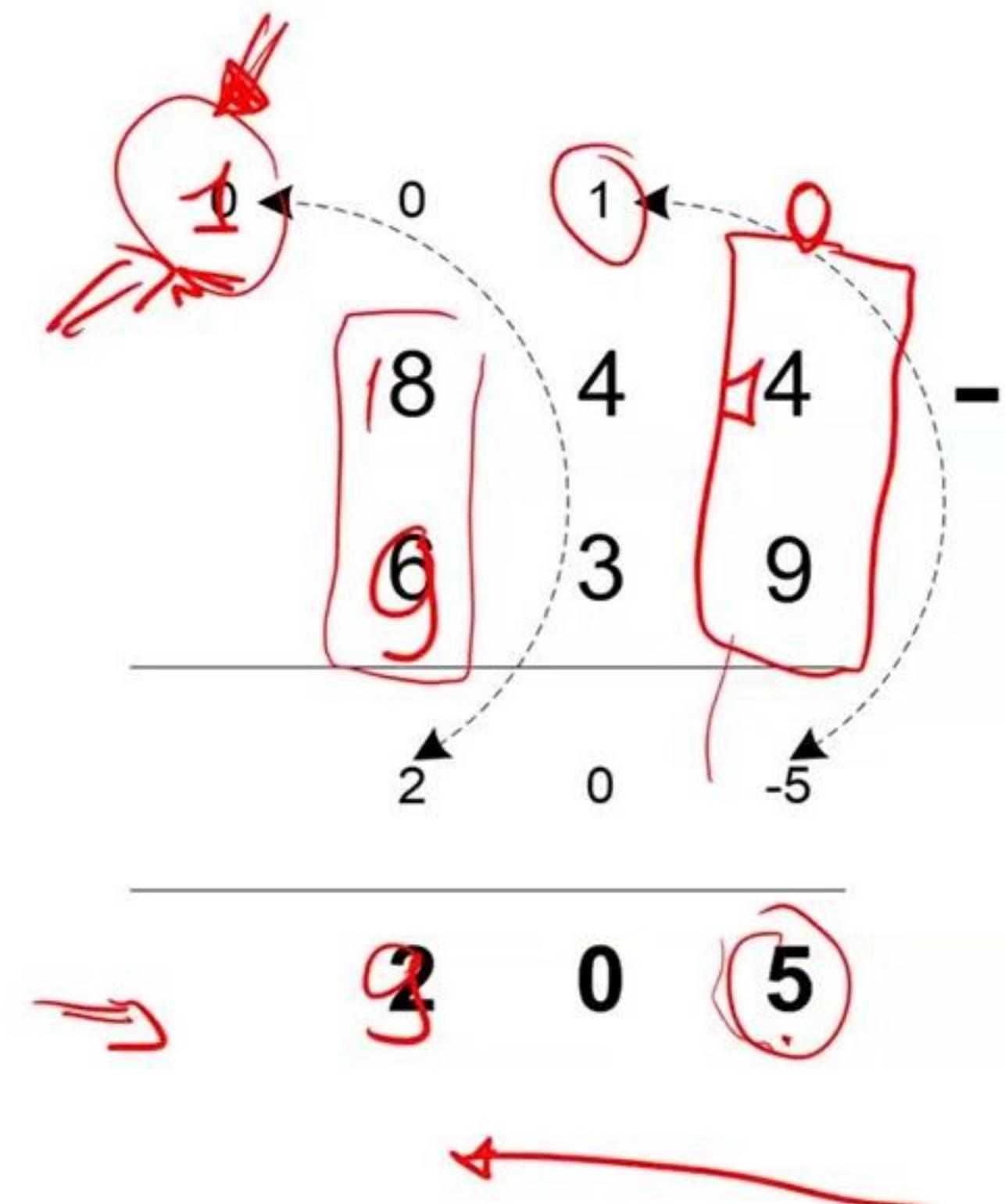
- più semplice del full adder
(ad un solo livello di logica)

ADD \$1, %AL
INC %AL

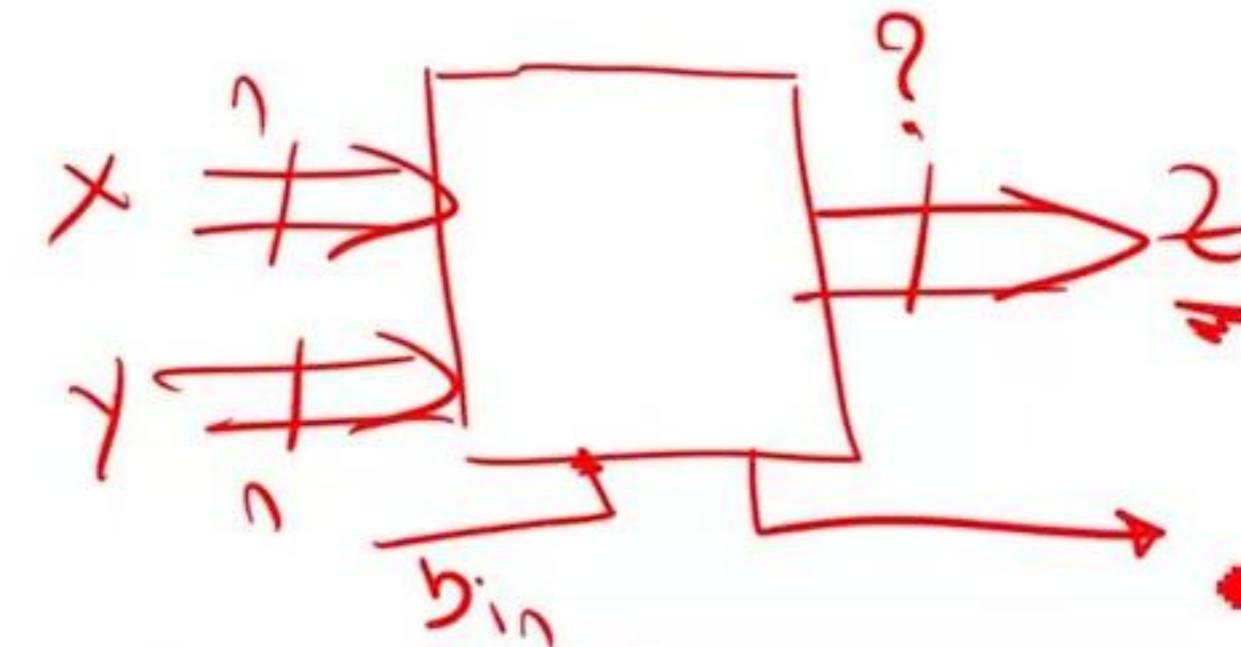
Sottrazione

IN

- L'algoritmo consiste in:
 - sottrarre le cifre di pari posizione singolarmente, partendo dalle meno significative
 - se la differenza di due cifre non è rappresentabile, prendere un **prestito (borrow)** dalla coppia di cifre successive
 - Il prestito **vale sempre 0 o 1**. Per la prima coppia di cifre (quelle meno significative), possiamo assumerlo **nullo**.
- Questo algoritmo **non dipende dalla base di rappresentazione**, ma soltanto dal fatto che usiamo una notazione posizionale.
- Può pertanto essere usato per sottrarre numeri in base qualunque



Sottrazione (cont.)



- Dati X, Y naturali in base β su n cifre, quindi $0 \leq X, Y \leq \beta^n - 1$
- Dato b_{in} , $0 \leq b_{in} \leq 1$,
- voglio calcolare il numero naturale $Z = X - Y - b_{in}$
 - **ammesso che esista**
 - i naturali non sono un insieme chiuso rispetto alla sottrazione.

$$\begin{array}{ccc} \beta^{-1} & 0 & 0 \\ 0 & \beta^{-1} & 1 \end{array}$$

$$\underline{-\beta^n \leq X - Y - b_{in} \leq \beta^n - 1}$$

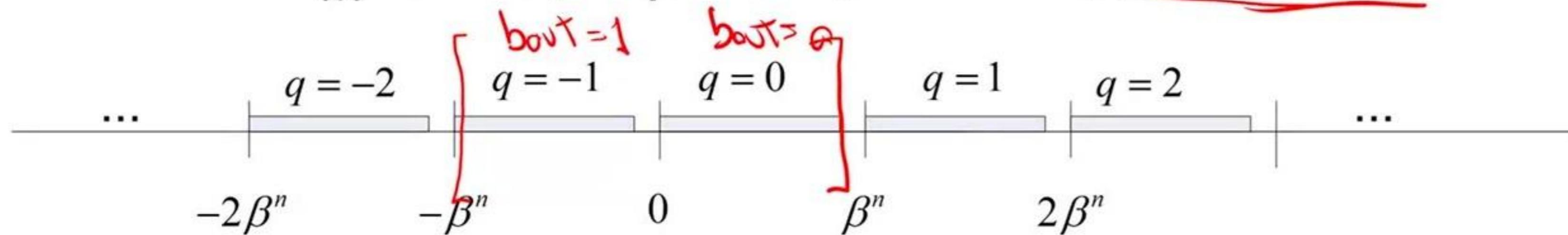
$\not{\exists}$
 \exists

- Quindi, Z può anche non essere un numero naturale

Sottrazione (cont.)

$$-\beta^n \leq Z \leq \beta^n - 1$$

- Posso scrivere Z come quoziente e resto di una divisione per β^n :
- $Z = X - Y - b_{in}$ diviso per β^n dà quoziente al minimo -1.



- Pertanto, definisco:

$$-b_{out} = \left\lfloor \frac{X - Y - b_{in}}{\beta^n} \right\rfloor, D = |X - Y - b_{in}|_{\beta^n}$$

- $b_{out} \in \underline{\{0,1\}}$, **indipendentemente dalla base**. Quindi, posso scrivere:

$$Z = (-b_{out}) \cdot \beta^n + D = X - Y - b_{in}$$

Sottrazione (cont.)

$$Z = -b_{out} \cdot \beta^n + D = X - Y - b_{in}$$

- la differenza tra due numeri naturali in base β su n cifre
- meno un eventuale prestito entrante
- produce un numero che, se naturale, è sempre rappresentabile su n cifre in base β
- Può inoltre produrre un numero non naturale, nel qual caso c'è un prestito uscente
- In ogni caso il prestito uscente può valere soltanto zero o uno

Calcolo della differenza

$$\bar{y} = \beta^{-1} - y$$

{

$$Z = -b_{out} \cdot \beta^n + D = X - Y - b_{in}$$

$$-b_{out} \cdot \beta^n + D = X + \bar{y} - \beta^{-1} - b_{in}$$

!

$$(1 - b_{out}) \cdot \beta^n + D = X + \bar{y} + (1 - b_{in})$$

$$\overline{b_{out}} \cdot \beta^n + D = X + \bar{y} + \overline{b_{in}}$$

•

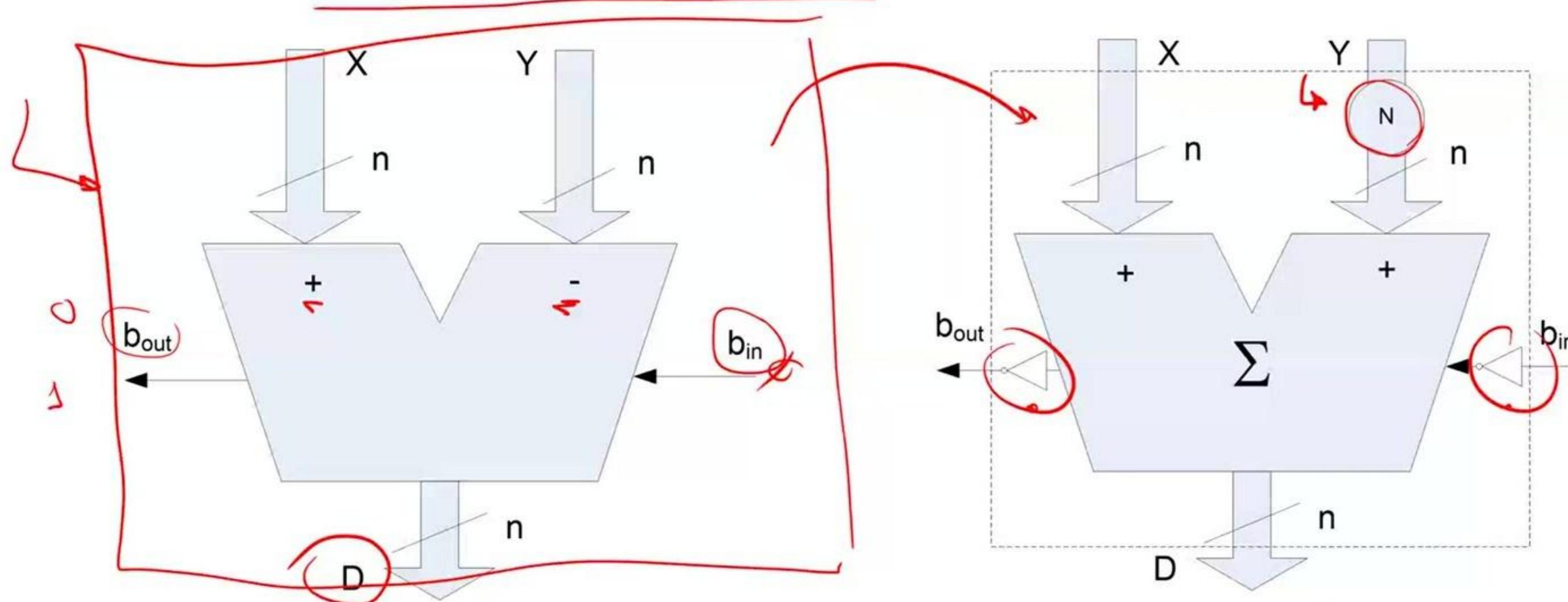
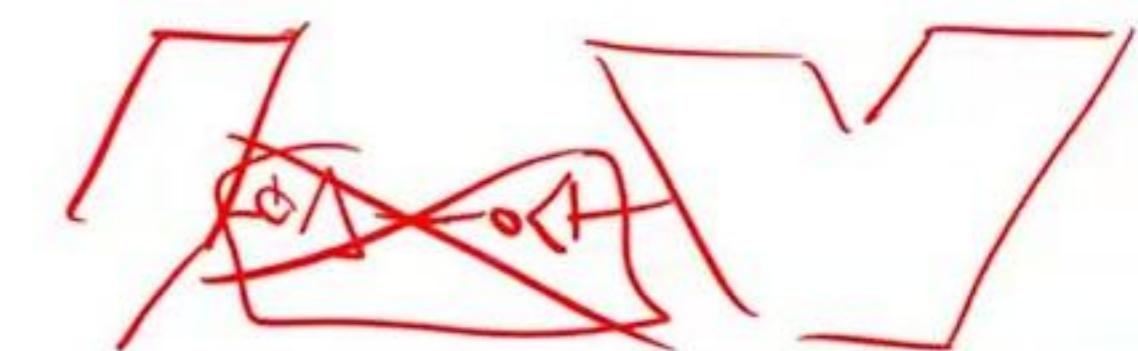
Calcolo della differenza (cont.)

$$\overline{b_{out}} \cdot \beta^n + D = X + \overline{Y} + \overline{b_{in}}$$

- la **differenza** tra X ed Y , meno un eventuale **prestito entrante**, se naturale, può essere ottenuta se **sommo X ed Y complementato**, più un eventuale **riporto entrante**, ottenuto complementando il prestito entrante.
- Se il **riporto uscente** della somma $\overline{b_{out}}$ vale
 - 1, la differenza è un numero naturale pari a D , ed il **prestito uscente** b_{out} è zero
 - 0, la differenza non è un numero naturale, ed il **prestito uscente** b_{out} è uno

Sottrattore

- Puo' essere realizzato a partire da un sommatore
- Scomponibile su singole cifre
- Circuito di decremento semplificato



Comparazione di numeri naturali

- I sottrattori sono spesso usati come **comparatori**

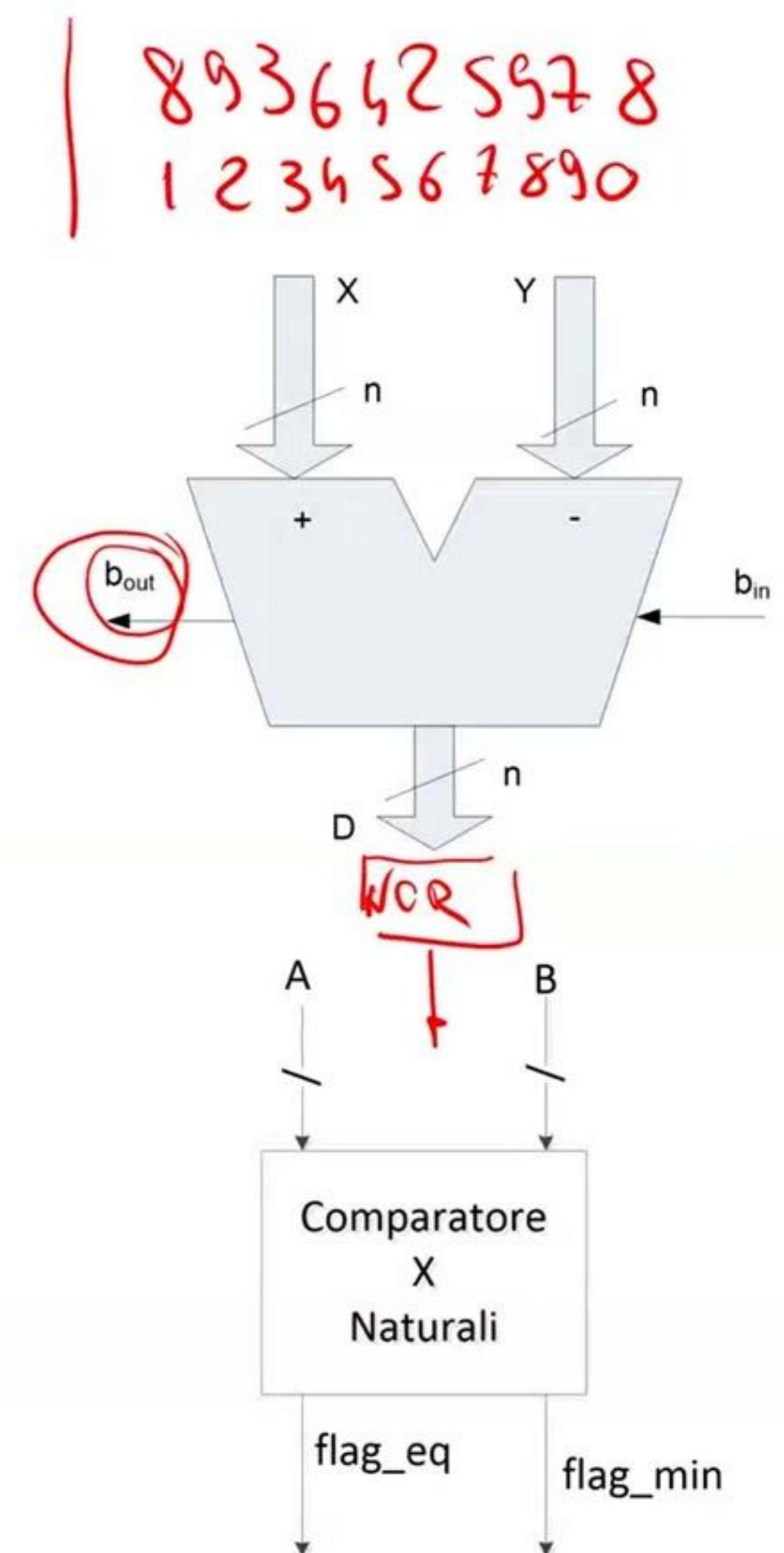
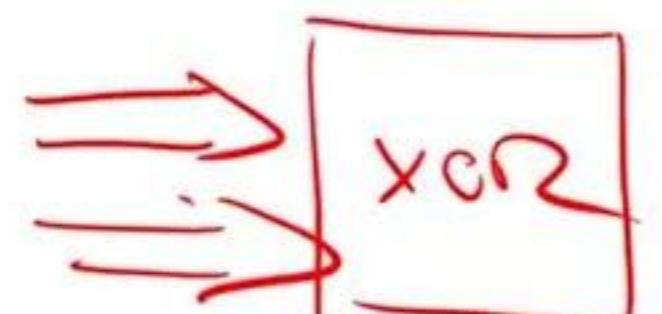
- Dati due *naturali* X e Y , voglio sapere se $X < Y$

- Calcolo $X - Y$ e guardo il prestito uscente.
- Se $b_{out} = 1$, allora X è più piccolo



$X > Y$

- Dati due *naturali* X e Y , voglio sapere se $X = Y$



Moltiplicazione

- Dati:
- X, C numeri naturali in base β su n cifre, tali quindi che $0 \leq X, C \leq \beta^n - 1$
- Y numero naturale in base β su m cifre, tali quindi che $0 \leq Y \leq \beta^m - 1$
- Voglio calcolare:

$P = X \cdot Y + C$
- Il termine C serve a rendere l'operazione **modulare** (come il Cin per la somma)

Moltiplicazione (cont.)

MUL

$2n$

$m+n$

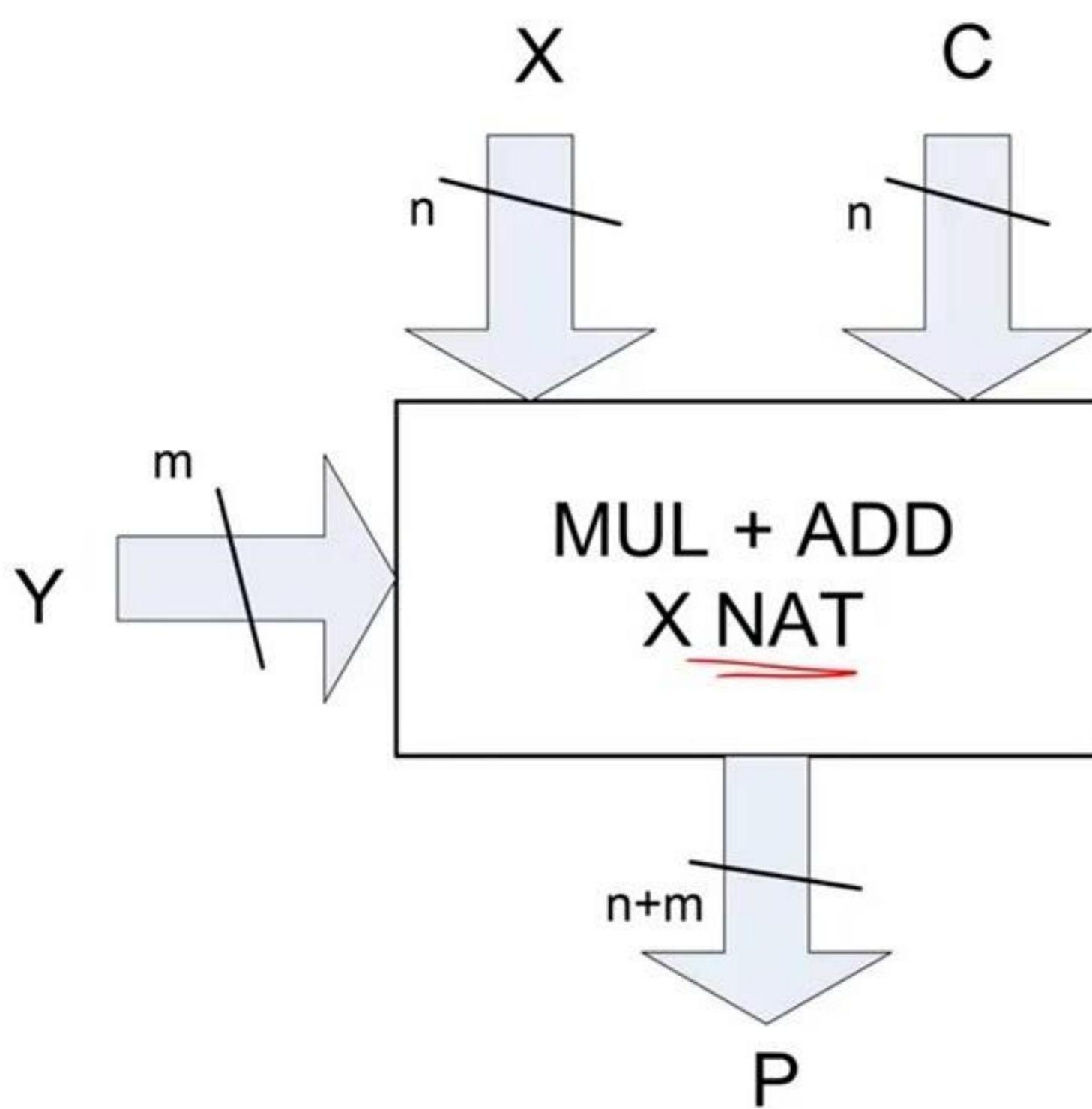


- X, C numeri naturali in base β su n cifre, tali quindi che $0 \leq X, C \leq \underline{\beta^n - 1}$
- Y numero naturale in base β su m cifre, tali quindi che $0 \leq Y \leq \underline{\beta^m - 1}$
- Su quante cifre sta il risultato?

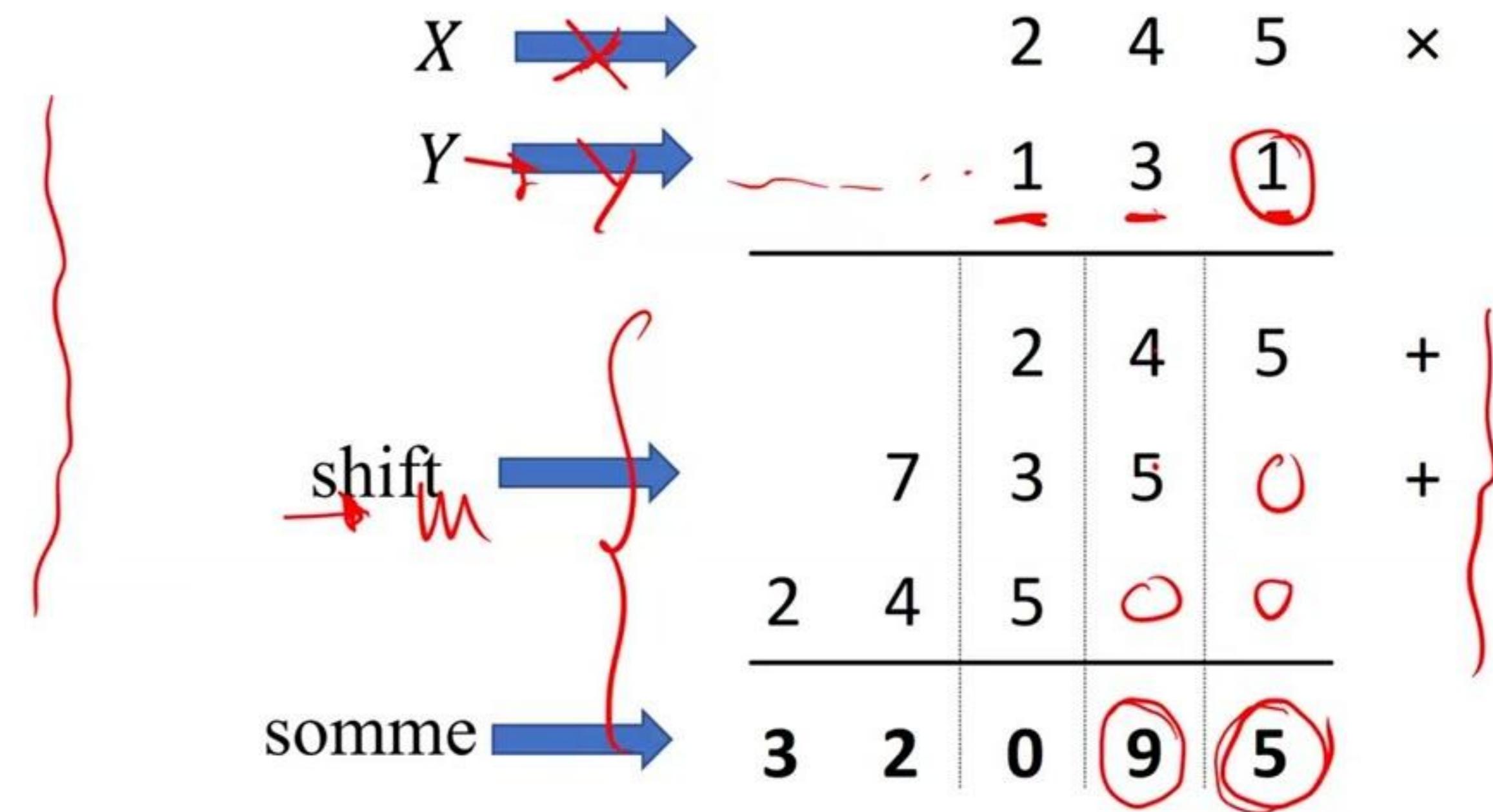
$$P = X \cdot Y + C \leq (\beta^n - 1) \cdot (\beta^m - 1) + (\beta^n - 1) =$$
$$\beta^m \cdot (\beta^n - 1) < \underline{\beta^{n+m} - 1}$$

The diagram illustrates the multiplication of two numbers, X and Y. Number X is represented as a vertical column of digits with a total height of n. Number Y is represented as a horizontal row of digits with a total width of m. The product P is shown as a large rectangle at the bottom, which is the sum of the areas of the rectangles formed by the digits of X and Y. The height of the rectangle P is labeled as $n+m$, indicating it spans the combined height of both factors.

Moltiplicatore con addizionatore per N



- Scomposizione del problema in problemi piu' semplici



Moltiplicazione (cont.)

- Si moltiplica X per **tutte le cifre di Y**, in step successivi.
- I **risultati** di ciascuno di questi prodotti parziali vengono scritti **a partire dal posto occupato dalla cifra per la quale si sta moltiplicando**.
- I risultati di ciascun prodotto parziale sono **sommati (con riporto)** per ottenere il prodotto.
- Algoritmo **indipendente dalla base**

$$\begin{array}{r} 2 \quad 4 \quad 5 \\ \times \\ 1 \quad 3 \quad 1 \\ \hline & & & 2 & 4 & 5 \\ & & & + & & \\ & & & 7 & 3 & 5 \\ & & & + & & \\ & & & 2 & 4 & 5 \\ & & & + & & \\ & & & 3 & 2 & 0 & 9 & 5 \end{array}$$

Moltiplicazione (cont.)

- Passi elementari dell'algoritmo:
 - Moltiplicare **un numero ad n cifre per un numero ad una cifra**
 - Sommare m addendi (traslandoli opportunamente), con riporto, per ottenere il risultato finale.
- Somma di m addendi **tutti in una volta** è complessa, ma:
 - La somma e' associativa
 - **la cifra di posto i** del prodotto, $0 \leq i \leq n - 1$, è determinata unicamente dai prodotti parziali relativi alle cifre $j \leq i$. Cioè: alla fine dell' i -simo prodotto parziale posso già stabilire il valore della i -sima cifra del prodotto.

$$\begin{array}{r} 2 & 4 & 5 & \times \\ 1 & 3 & 1 \\ \hline & & & \\ & 2 & 4 & 5 & + \\ 7 & 3 & 5 & + \\ \hline 2 & 4 & 5 \\ \hline 3 & 2 & 0 & 9 & 5 \end{array}$$

Moltiplicazione (cont.)

- Riscrivo l'algoritmo come segue:
- Moltiplicare il numero X per **una cifra di Y** , **sommando un termine C** (che inizialmente è nullo). Il risultato che si ottiene viene suddiviso:
 - la cifra meno significativa diventa la cifra i -sima del prodotto.
 - le altre cifre diventano **il nuovo termine da sommare** per il prossimo passo.
- Alla fine, si ottiene il prodotto **concatenando tutti i risultati**.
- In questo modo si fanno solo moltiplicazioni su $n \times 1$ cifra e si sommano soltanto due addendi (su $n + 1$ cifre).

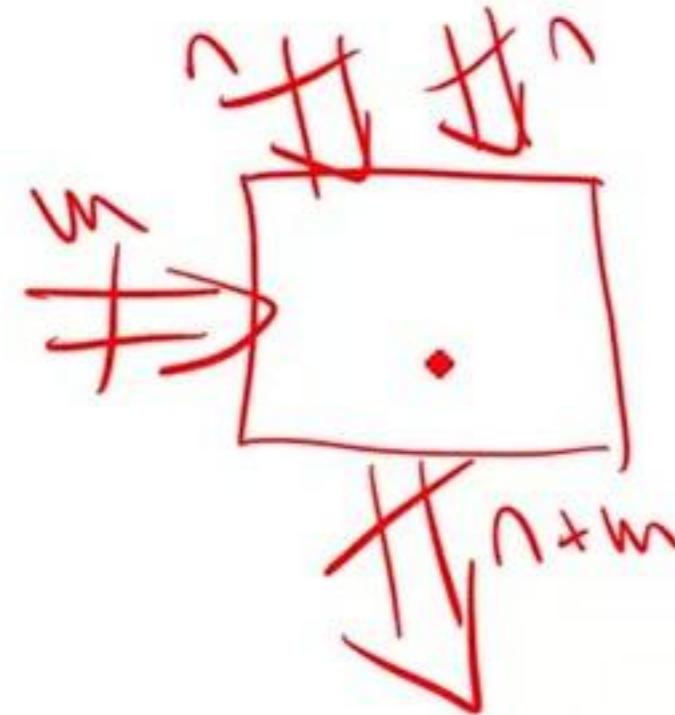
The diagram shows the multiplication of 245 by 13. It is divided into three main steps:

- Step 1:** Shows 245 multiplied by 1. The result is 245, with a carry of 0. The 5 is circled in red.
- Step 2:** Shows the result from Step 1 (245) plus 245 (from multiplying by 3). The result is 490. A carry of 4 is shown above the first column, and the 0 is circled in red.
- Step 3:** Shows the result from Step 2 (490) plus 245 (from multiplying by 2). The result is 735. A carry of 7 is shown above the first column, and the 5 is circled in red.

The final result is 32095, with the 5 circled in red.

Moltiplicazione (cont.)

- Per via circuitale ($m = 3$)



2 4 5 X

1 3 1

	2	4	5	X	
	1	3	1		
	0	5	5		

+ 2	4	5	0	5	
+ 7	3	5	5		
+ 2	4	5	9		
	3	2	0	9	5

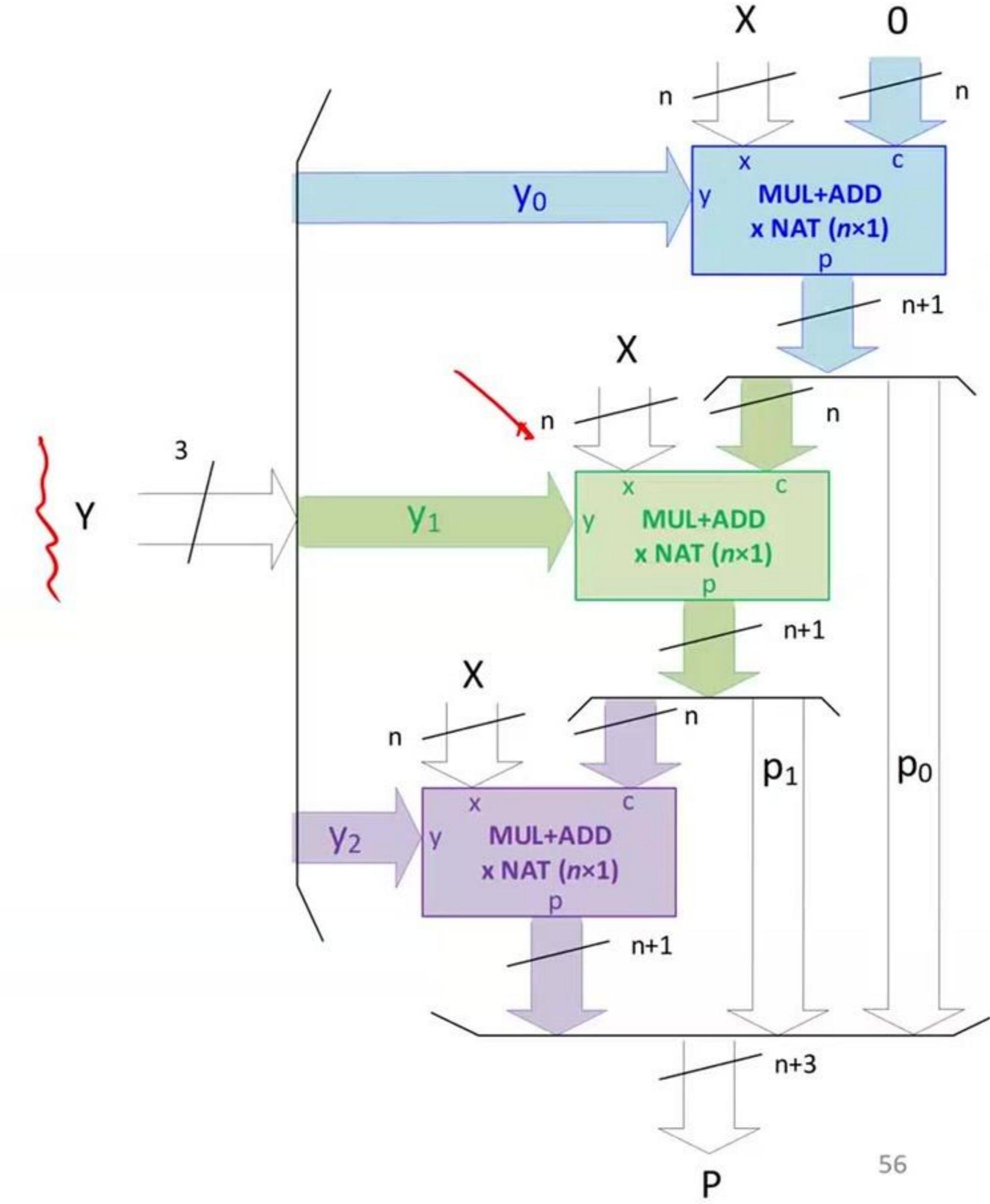
Y

3

+

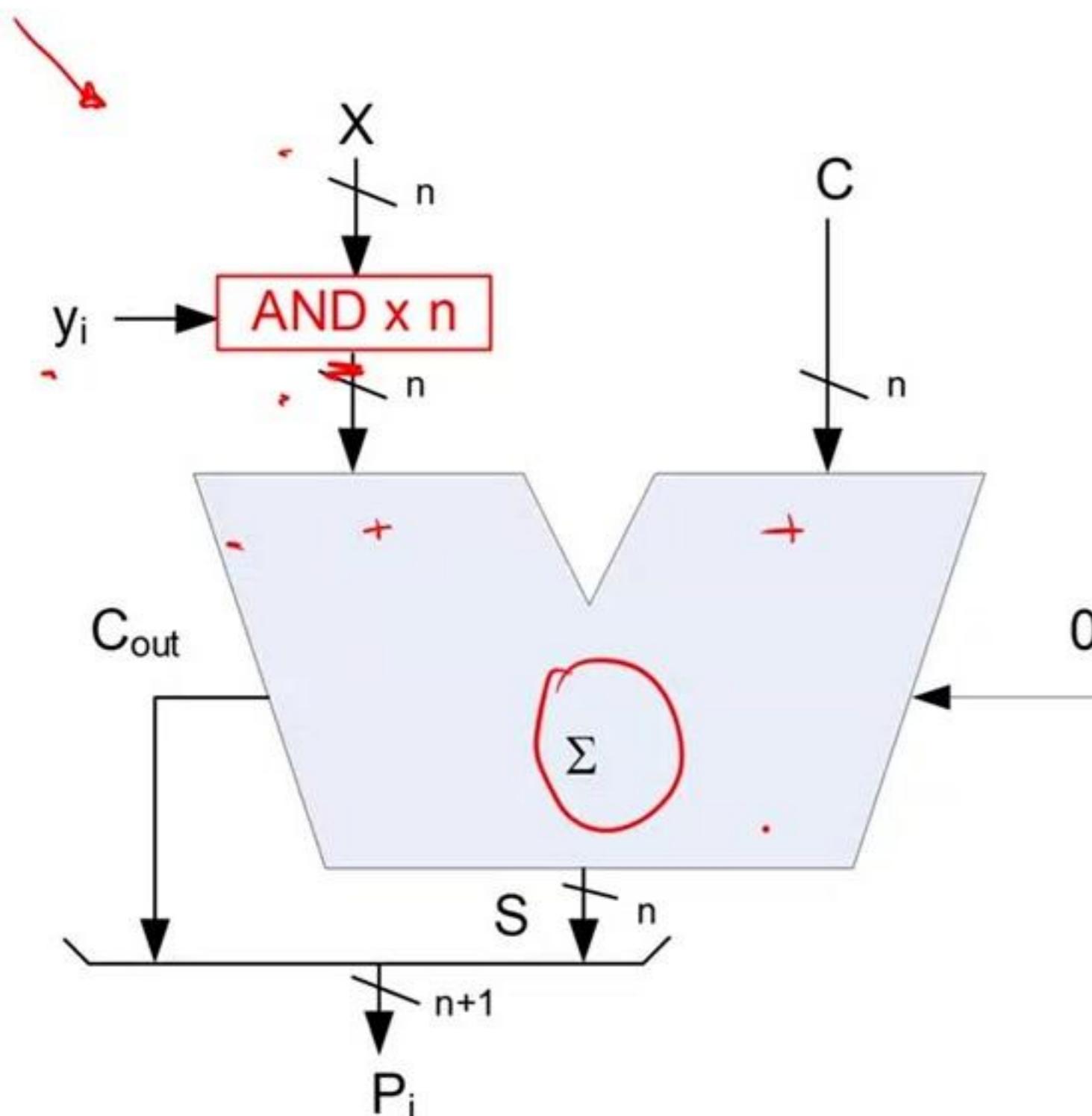
+

+



Moltiplicatore con addizionatore $n \times 1$ in base 2

- Il passo base richiede **moltiplicatori (con addizionatore) ad $n \times 1$ cifra**
- Vediamo come e' fatto in base 2



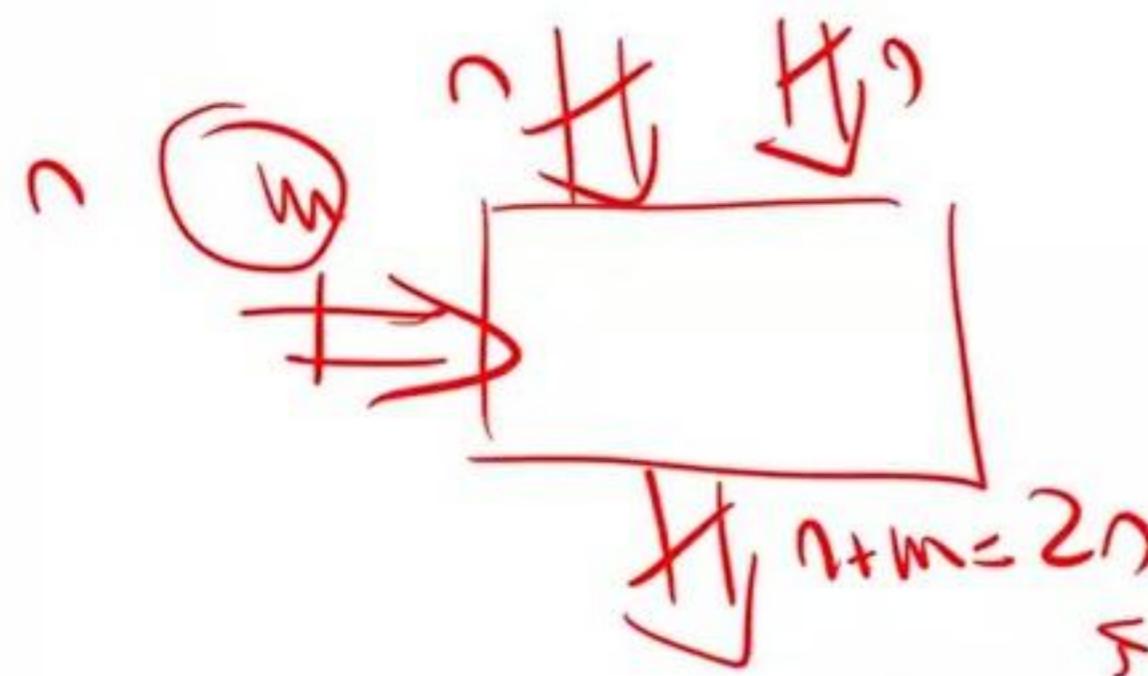
Il risultato che deve uscire da qui è:

$$P_i = y_i \cdot X + C = \begin{cases} 0 + C & y_i = 0 \\ X + C & y_i = 1 \end{cases}$$

3^s E - X

Moltiplicazione di naturali in Assembler

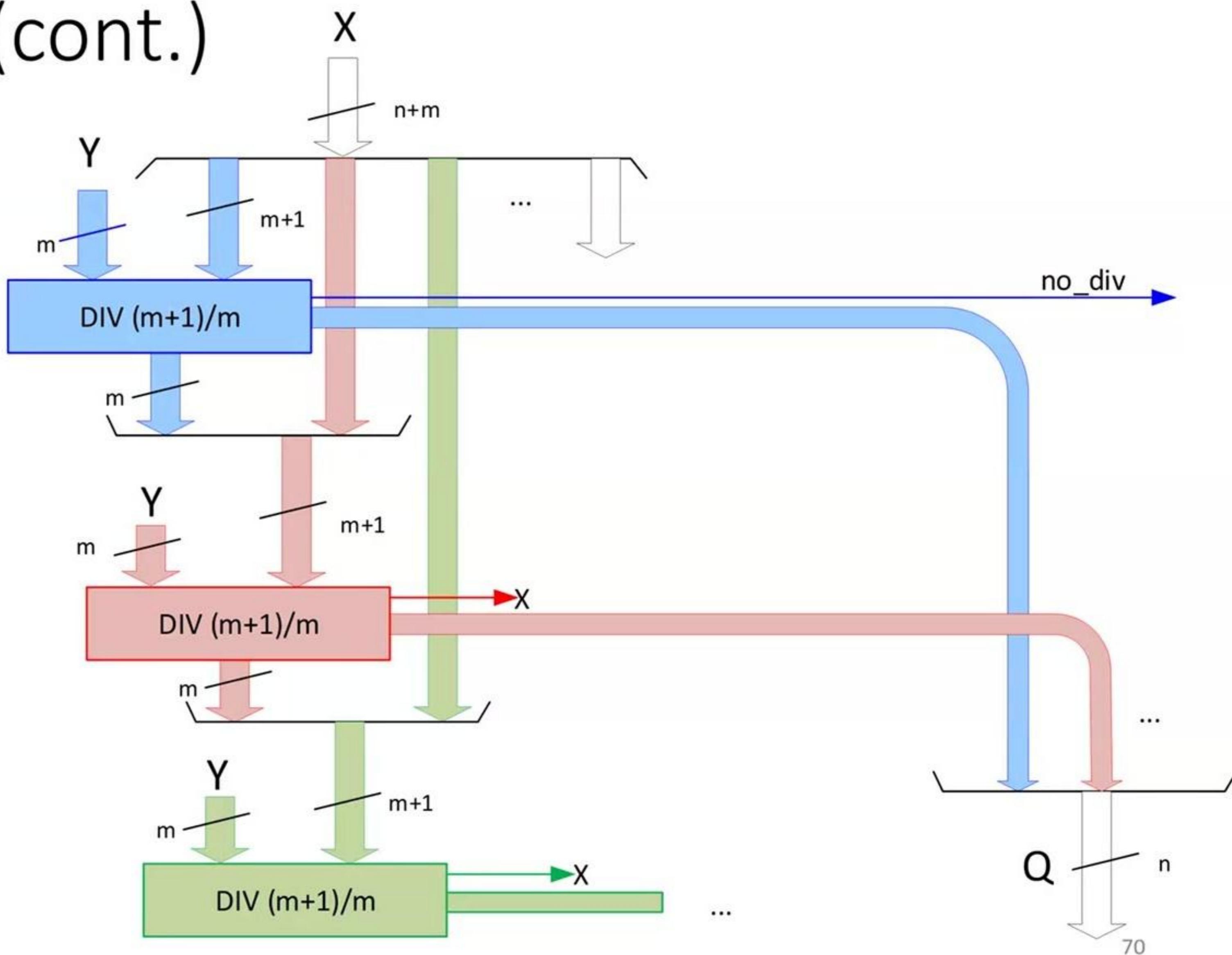
- L'istruzione Assembler **MUL** ha **un solo operando**.
- A seconda della **dimensione dell'operando**, seleziona l'altro operando (implicito: AL, AX, EAX), e mette il risultato in un destinatario implicito (AX, DX_AX, EDX_EAX).
- In pratica, abbiamo $n = m$, e quindi i circuiti che calcolano il prodotto calcolano risultati su $2n$ bit partendo da operandi a n bit.



Scomposizione (cont.)

$$\begin{aligned} \rightarrow X &< \beta^n \cdot Y \\ \Leftrightarrow \\ \rightarrow (x_{n+m-1} \dots x_n)_\beta &< Y \end{aligned}$$

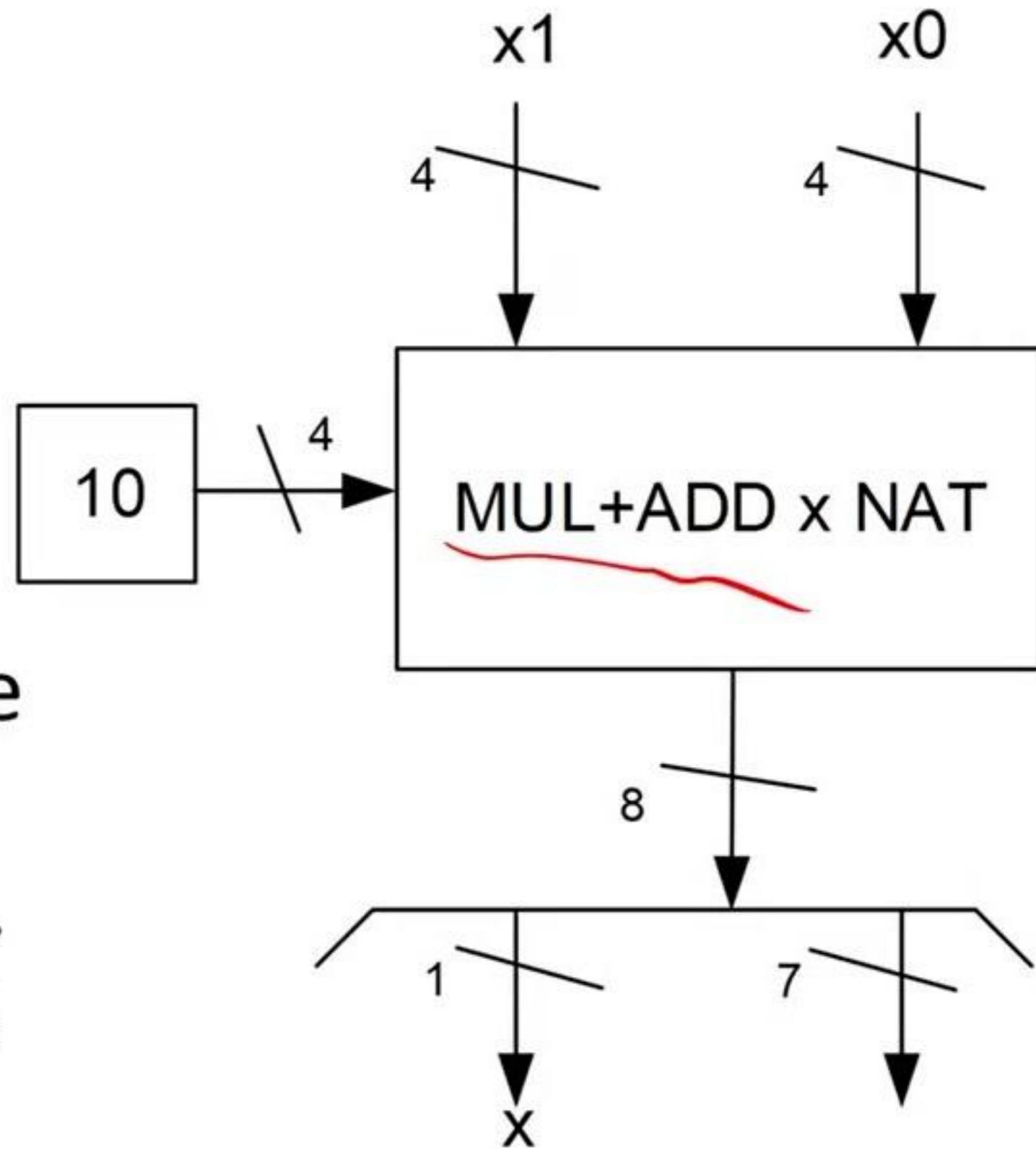
u.s.



Soluzione

1. Su quante cifre sta il risultato?
 - Visto che $z \leq 99$, bastano 7 bit.
2. Come trovo le cifre del risultato, a partire dalle cifre degli operandi?
 - Le due cifre di ingresso sono x_1 e x_0 e sono codificate BCD.
 - Una cifra BCD è anche **un numero naturale in base 2, su 4 bit**
 - Quindi, il risultato da calcolare è:

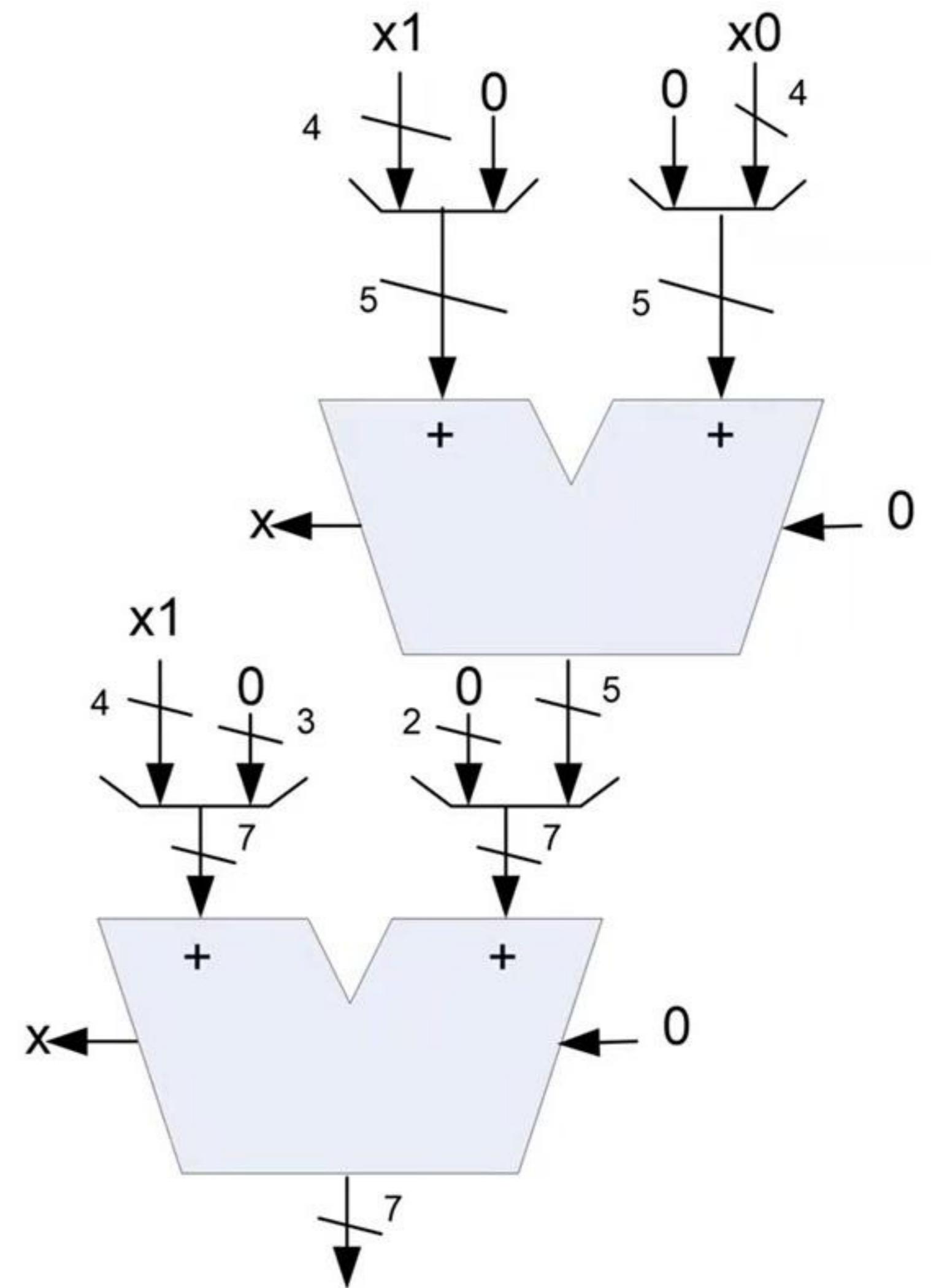
$$Y = 10 \cdot x_1 + x_0 = 8x_1 + 2x_1 + x_0$$



Soluzione alternativa

$$Y = 10 \cdot x_1 + x_0 = 8 \cdot x_1 + 2 \cdot x_1 + x_0$$

- Solo somme e shift



Divisione

- Dati:
- X naturale in base β su $n + m$ cifre (**dividendo**), $0 \leq X \leq \beta^{n+m} - 1$
- Y naturale in base β su m cifre (**divisore**), $0 \leq Y \leq \beta^m - 1$

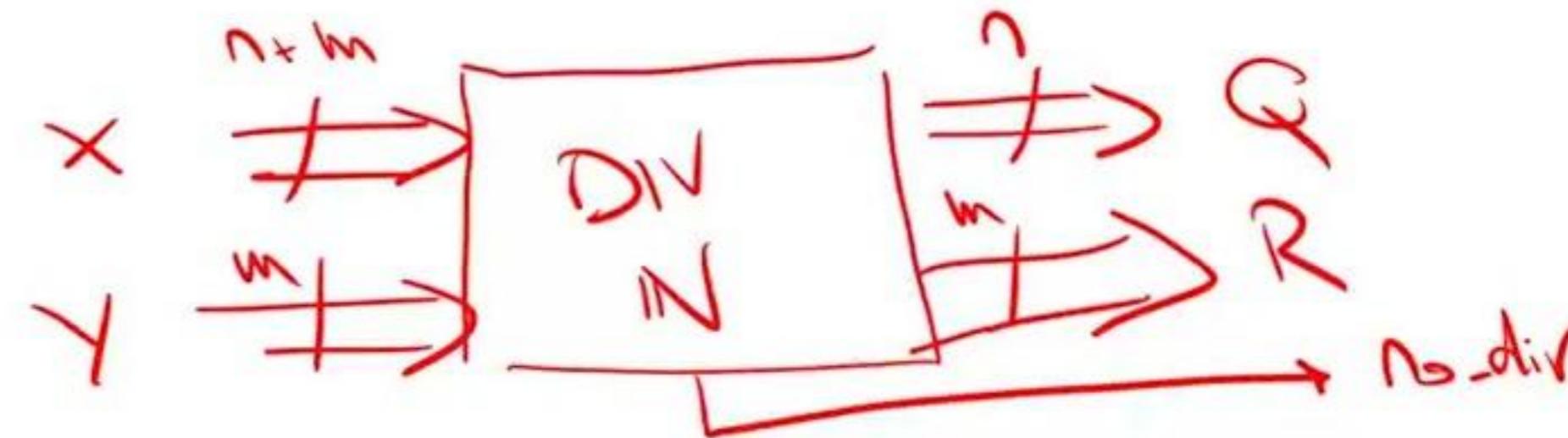
- Voglio calcolare i due numeri Q ed R tali che:

$$X = Q \cdot Y + R$$

$$\begin{cases} Q \in \mathbb{N} \\ R < Y \end{cases}$$

- Non sempre fattibile ($Y = 0$). Sarà necessaria un'uscita di **non fattibilità** no_div

Divisione (cont.)



- Assumendo che $Y > 0$, su quante cifre dovranno essere rappresentati Q ed R ?
 - Il resto, dovendo essere minore del divisore, sta sicuramente su m cifre.
 - Per il quoziente non posso dire molto. Infatti, se $Y = 1$, allora $Q = X$, e quindi al peggio sta su $n + m$ cifre.
- Voglio** poter rappresentare il quoziente n cifre. Assumere che Q stia su n cifre implica che:

$$\underbrace{X}_{\text{ }} = \underbrace{Q}_{\text{ }} \cdot \underbrace{Y}_{\text{ }} + R \leq (\beta^n - 1) \cdot Y + (Y - 1) = \underbrace{\beta^n \cdot Y}_{\text{ }} - 1$$

- Ipotesi aggiuntiva** che garantisce che il quoziente stia su n cifre

$$X < \beta^n \cdot Y$$

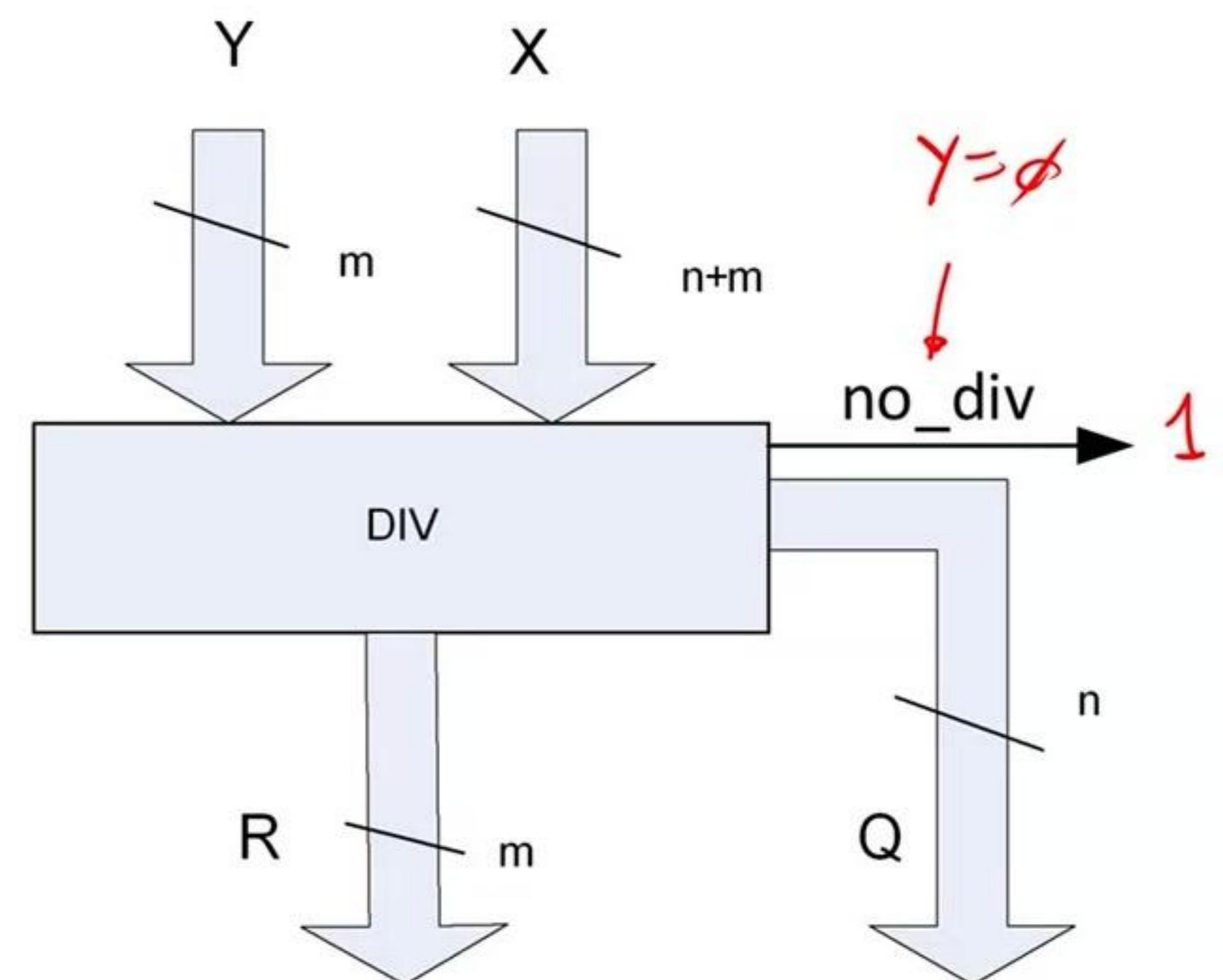
Divisione (cont.)

- Sotto l'ipotesi $X < \beta^n \cdot Y$ non posso fare **tutte le divisioni**, ma **soltanto alcune**.

- Quest'ipotesi è **restrittiva**?
 - Se n, m non sono dati del problema (cioè non sono fissati a priori), dati X ed Y posso **sempre** trovare n tale che la diseguaglianza sia vera
 - posso sempre fare la divisione, purché sia in grado di **estendere** la rappresentazione del dividendo ed abbia un numero sufficiente di cifre per il quoziente).
 - Se il **numero di cifre n, m è un dato del problema**, cioè se si lavora su **campi finiti** (sempre, all'interno di un calcolatore) questa ipotesi è restrittiva

Modulo divisore

- Deve testare la **fattibilità** della divisione nelle ipotesi date.
- Se il quoziente non sta su n cifre, **deve settare la variabile logica *no_div***, ad indicare che la divisione non è fattibile

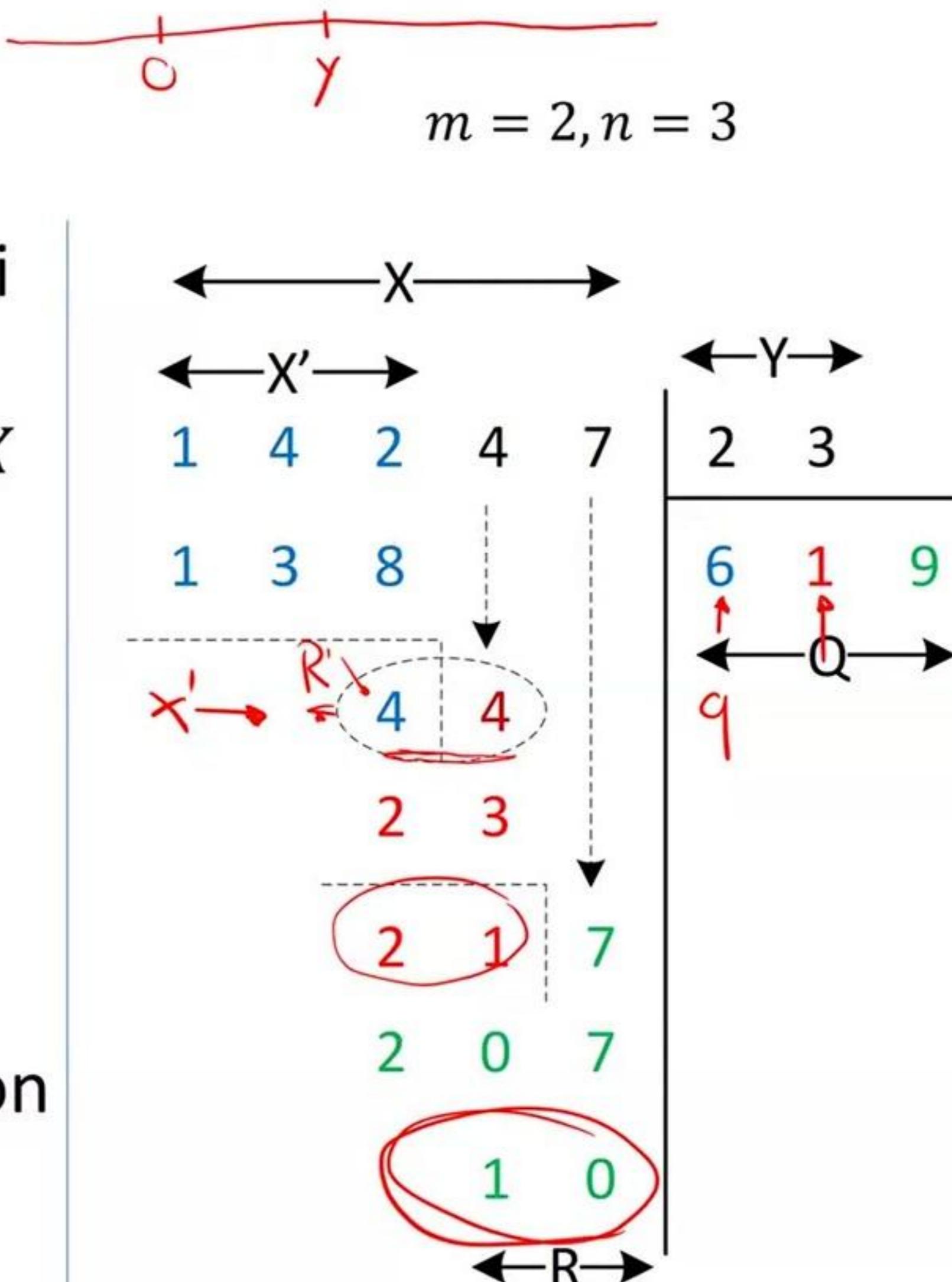


Scomposizione della divisione

$$\cancel{X' < \beta \cdot Y}$$

- Prendo il **min numero** delle cifre **più significative** di X per ottenere un **numero $\underline{X}' \in [Y, \beta \cdot Y[$**
 - \cancel{m} possono non bastare; $\underline{m+1}$ cifre bastano** (purché X non abbia zeri in testa)
- Calcolo un **quoziente e resto parziali** q, R' della divisione di X' per Y .
 - q sta su una sola cifra (perché $X' < \beta \cdot Y$ per ipotesi)
- Calcolo un **nuovo dividendo** X' concatenando $\cancel{R'}$ con la **cifra più significativa** non ancora utilizzata di X .
 - Il nuovo dividendo X' è ancora **minore di $\beta \cdot Y$** , date le ipotesi

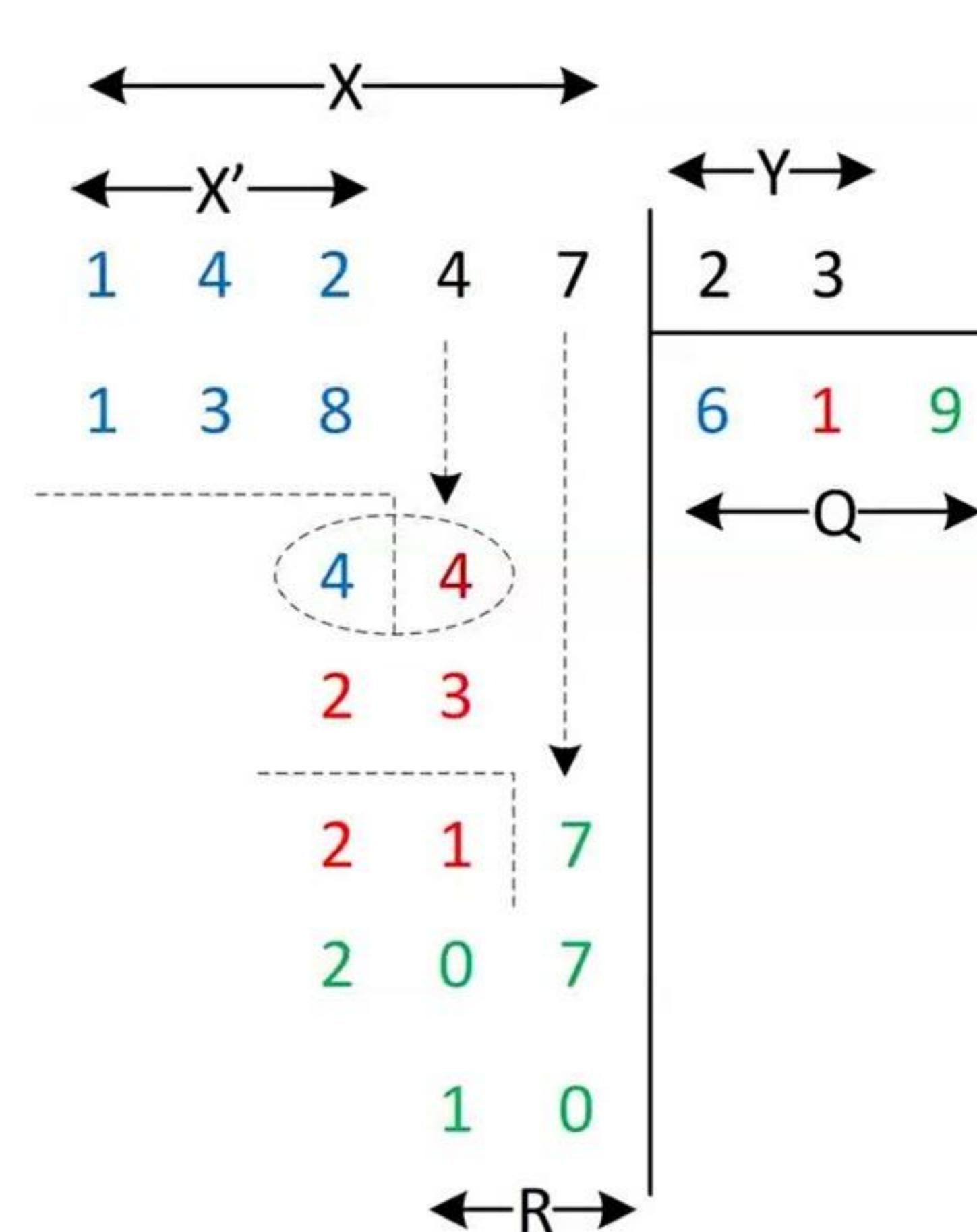
$$R' \leq Y-1 \quad \beta R' + (\beta-1) \leq \beta Y - \cancel{\beta} + \cancel{\beta} - 1$$



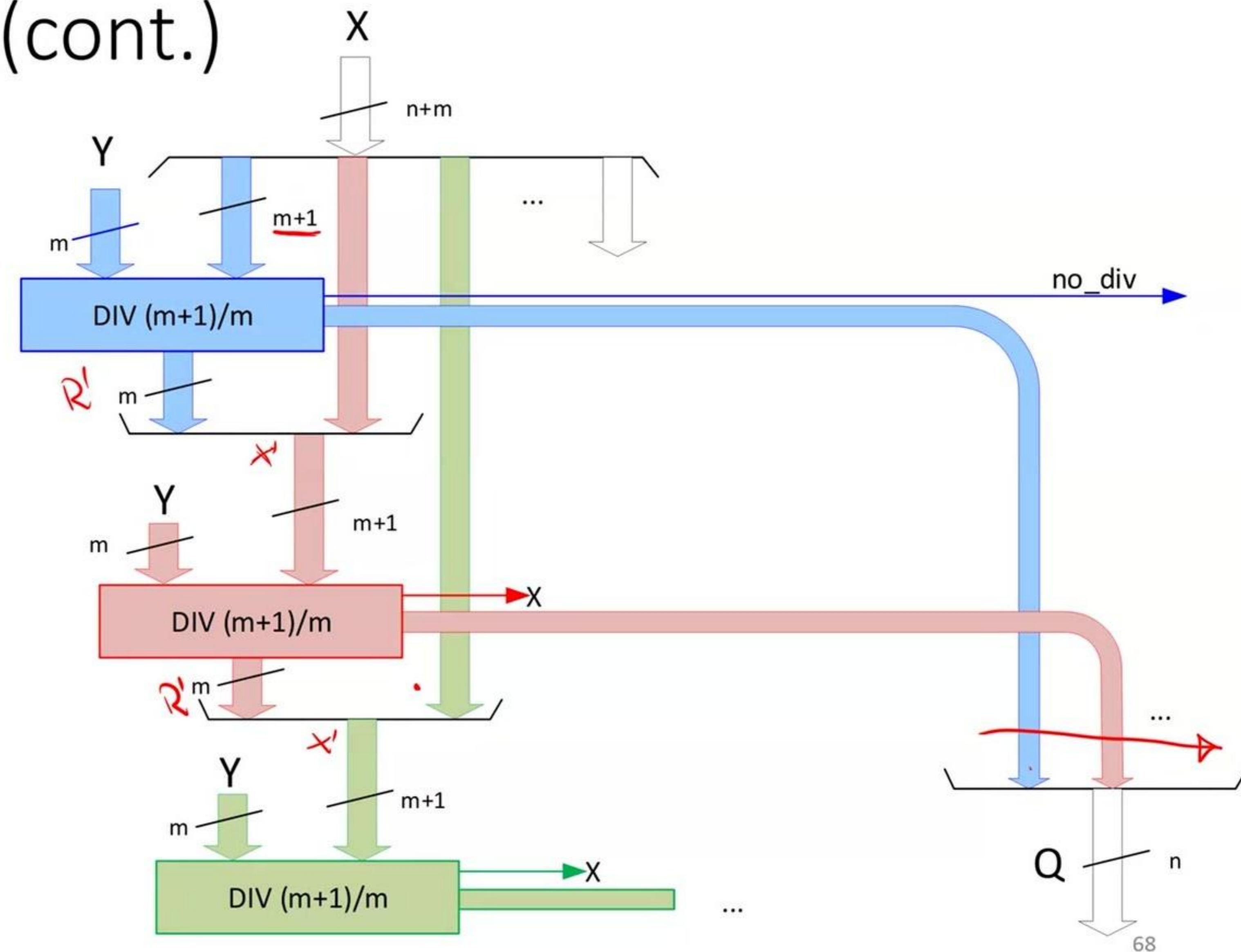
Scomposizione della divisione (cont.)

$m = 2, n = 3$

- Vado avanti fino ad esaurire le cifre del dividendo.
- Il **quoziente** è ottenuto dal concatenamento dei quozienti parziali (tutti su una cifra).
- Il **resto** è il resto dell'ultima divisione parziale.
- Nucleo di questo algoritmo iterativo:
 - divisione di $m + 1$ cifre per m cifre, che produce un **quoziente su una cifra ed un resto su m cifre**.
 - tutto il resto può essere ottenuto concatenando cifre (per produrre i nuovi dividendi e il quoziente finale)



Scomposizione (cont.)



Scomposizione (cont.)

$$(x_{n+m-1} \dots x_0)_\beta < y$$

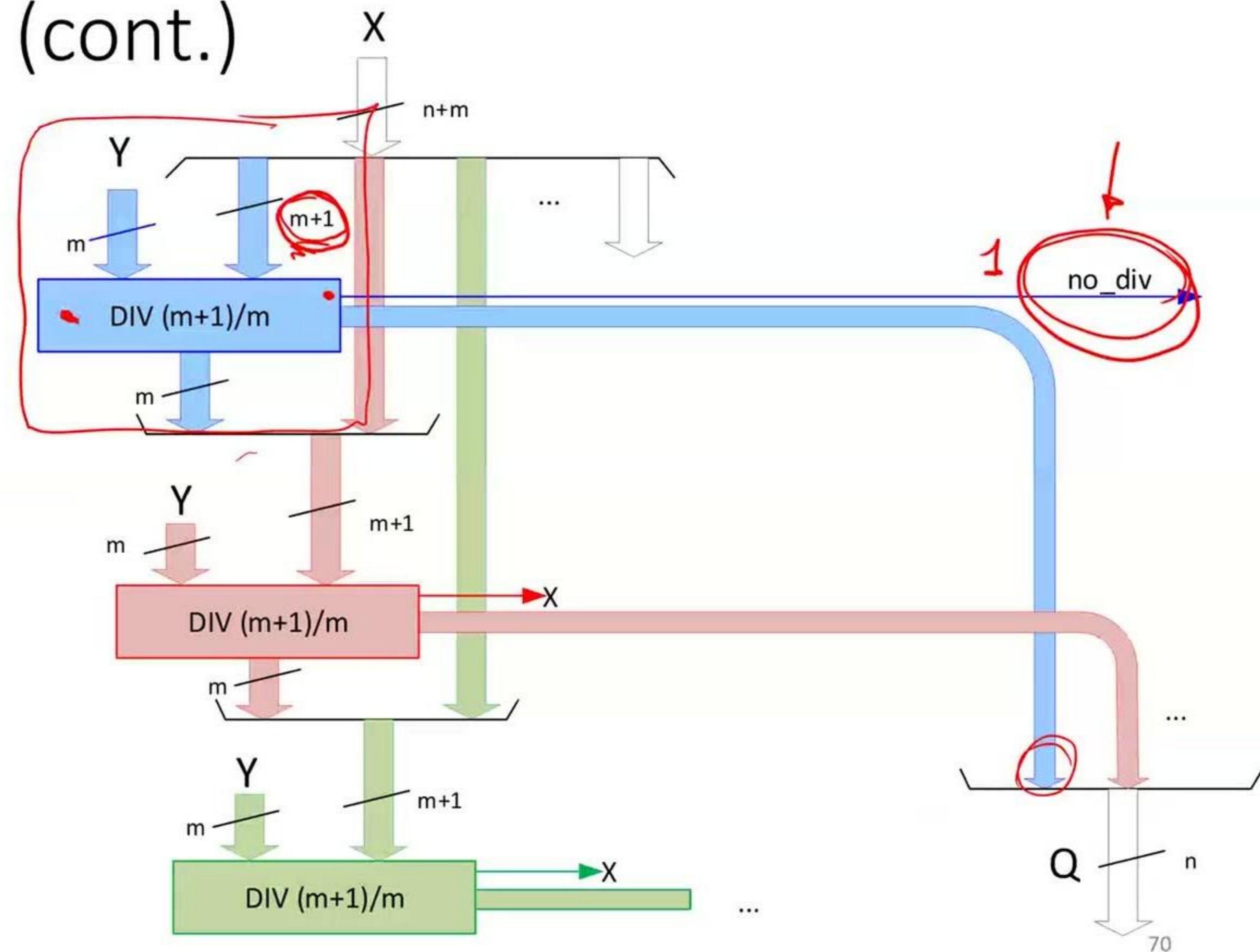
- Condizione di fattibilità: **il quoziente stia su n cifre è $\underline{X < \beta^n \cdot Y}$.** ↗
- equivalente a dire che le m cifre più significative del dividendo rappresentano un numero più piccolo del divisore.** ↗
- Infatti, posso rappresentare X e $\beta^n \cdot Y$ in termini di cifre, come:

$$\begin{array}{rcl} X: & \left[\begin{array}{ccccc} x_{n+m-1} & x_{n+m-2} & \dots & x_{n+1} & x_n \end{array} \right] & | \quad x_{n-1} \quad \dots \quad x_0 \\ \beta^n \cdot Y: & \left[\begin{array}{ccccc} y_{m-1} & y_{m-2} & \dots & y_1 & y_0 \end{array} \right] & | \quad 0 \quad \dots \quad 0 \end{array}$$

- $X < \beta^n \cdot Y$ se e solo se le cifre di X a sinistra della linea verticale rappresentano un numero minore di Y .

Scomposizione (cont.)

$$\begin{aligned}
 & \rightarrow X < \beta^n \cdot Y \\
 & \Leftrightarrow \\
 & \rightarrow (x_{n+m-1} \dots x_n)_\beta < Y \\
 & \Leftrightarrow \\
 & (x_{n+m-1} \dots x_n x_{n-1})_\beta < \beta \cdot Y \\
 & \text{---} \\
 & m+1 \quad \geq \\
 & \text{MSD del dividendo}
 \end{aligned}$$



La divisione nei processori Intel x86

- Tre versioni possibili
 - 8 bit: $AL = \text{quoziente} (AX / \text{source})$; $AH = \text{resto} (AX / \text{source})$
 - 16 bit: $AX = \text{quoziente} (DX_AX / \text{source})$; $DX = \text{resto} (DX_AX / \text{source})$
 - 32 bit: $EAX = \text{quoziente} (EDX_EAX / \text{source})$; $EDX = \text{resto} (EDX_EAX / \text{source})$



source (divisore)	dimensione dividendo	dividendo	quoziente	resto
8 bit	16	%AX	%AL	%AH
16 bit	32	%DX %AX	%AX	%DX
32 bit	64	%EDX %EAX	%EAX	%EDX

- Il programmatore seleziona la versione opportuna, **scegliendo un operando source (divisore) a 8, 16, 32 bit**

La divisione nei processori Intel x86 (cont.)

- La DIV ammette **dividendo su $2n$ bit e divisore su n bit**, con $n = 8, 16, 32$, e richiede che il **quoziente** stia su n bit (altrimenti genera **un'eccezione**)
- E' quello che si otterrebbe ponendo $\underline{n = m}$.
- E' **cura del programmatore** assicurarsi che $\underline{X < \beta^n \cdot Y}$
 - **estendendo la rappresentazione** del dividendo (e del divisore) su un numero maggiore (doppio) di bit.
- Dato che $\underline{n = 32}$, la diseguaglianza può essere resa vera, eventualmente estendendo le rappresentazioni, per **qualunque dividendo su 32 bit e qualunque divisore (non nullo)**

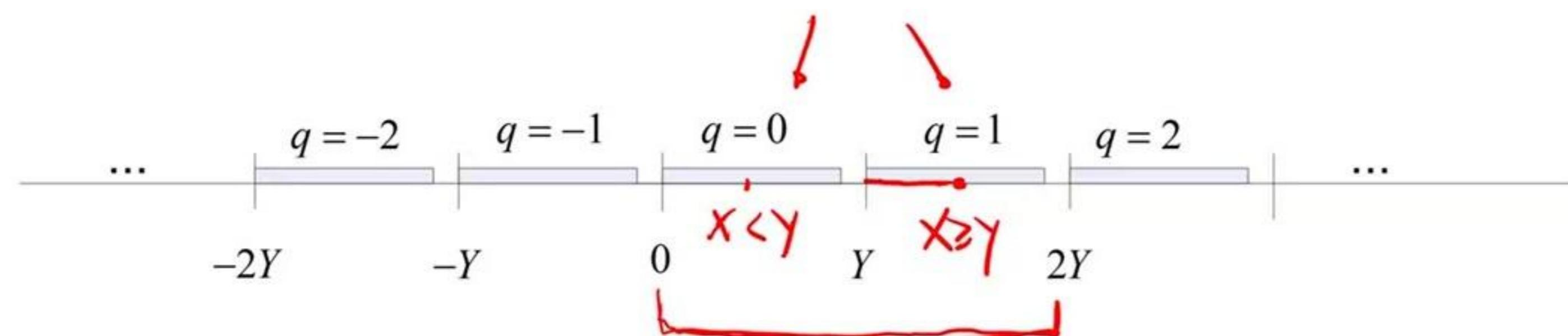
$2n=64\text{ bit}$

32 bit

Divisore elementare in base 2

- Esegue una divisione di un dividendo a $m + 1$ cifre per un divisore ad m cifre, sotto l'ipotesi che $X < 2Y$
 - quoziente su una cifra, resto su m cifre

$$Q = \begin{cases} 0 & X < Y \\ 1 & X \geq Y \end{cases} \quad R = \begin{cases} X & X < Y \\ X - Y & X \geq Y \end{cases}$$



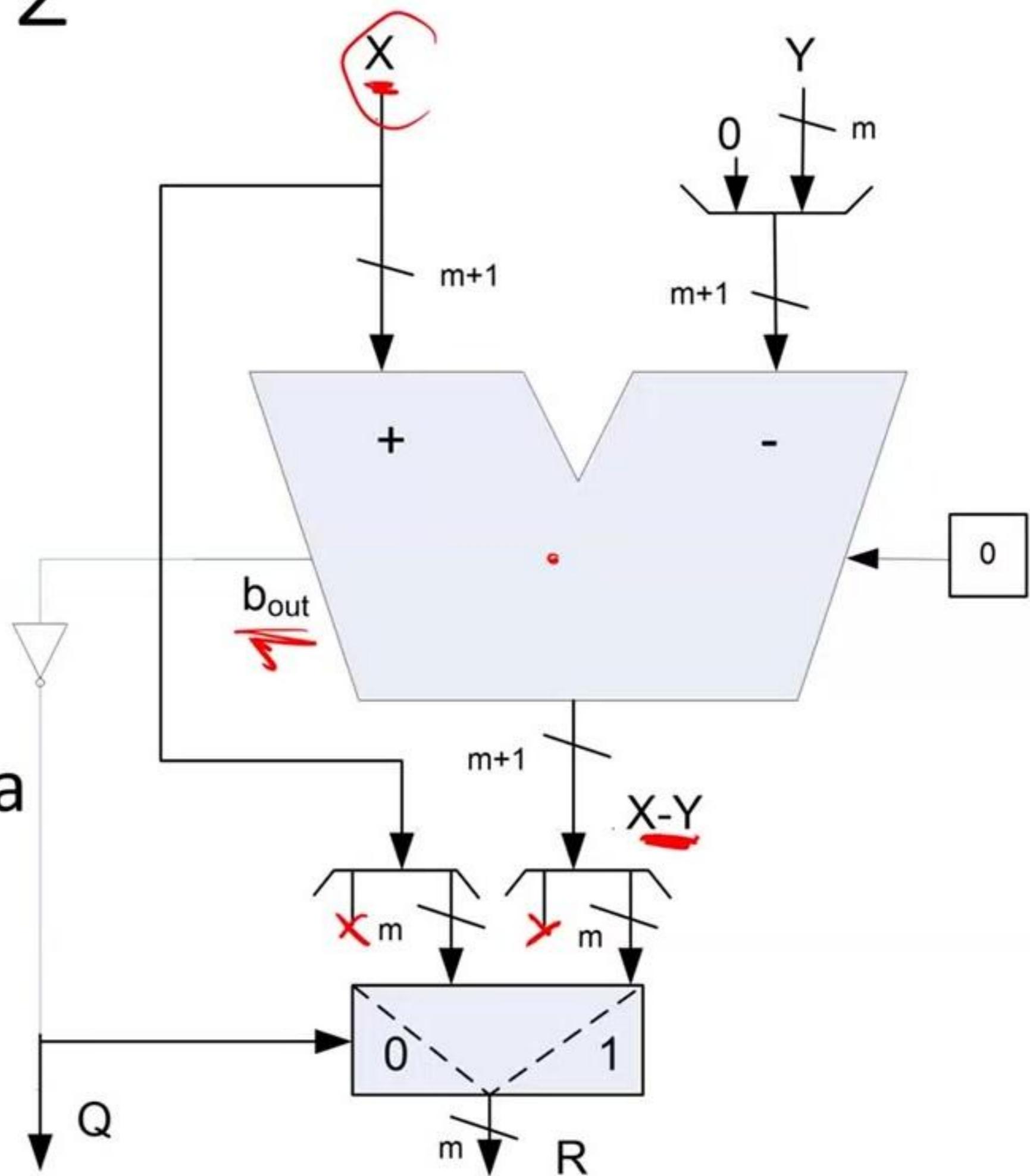
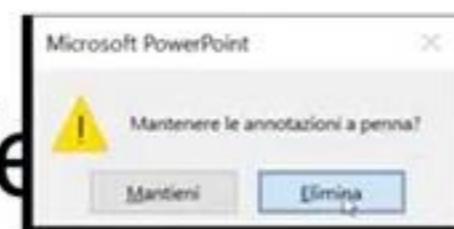
Divisore elementare in base 2

$$Q = \begin{cases} 0 & X < Y \\ 1 & X \geq Y \end{cases} \quad R = \begin{cases} X & X < Y \\ X - Y & X \geq Y \end{cases}$$

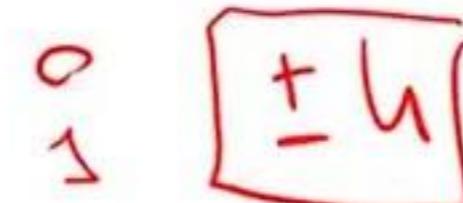
b_{out} = 1

b_{out} = φ

- Confrontare X ed Y
 - Eventualmente, calcolare una differenza
 - Rilevatore di fattibilità (uscita *no_div*) è l'uscita *flag_min* di un *comparatore* tra X e $2Y$
- g^s* *m*



Rappresentazione dei numeri interi

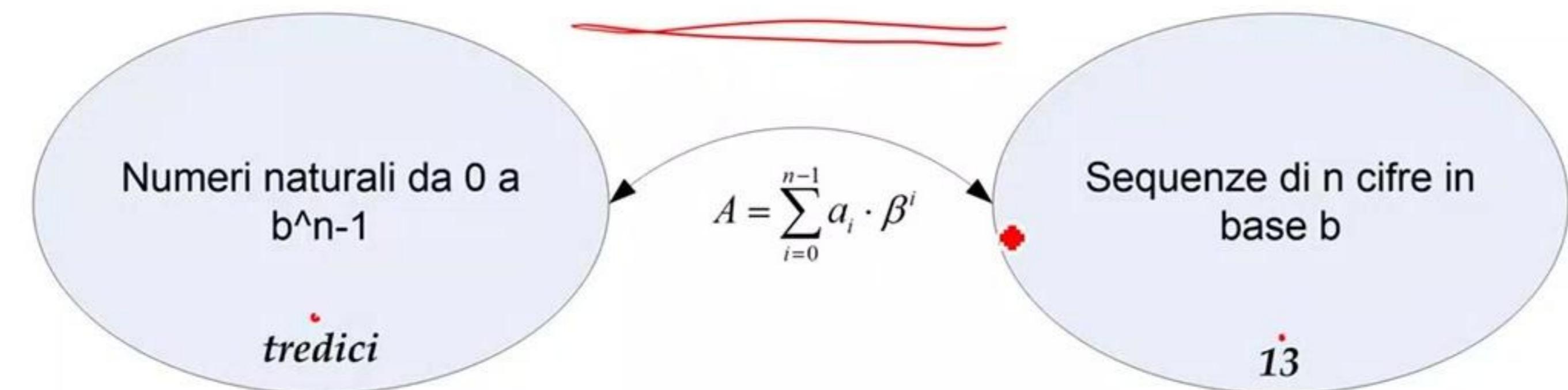


- Numeri **interi**: quelli che siamo abituati a rappresentare con **un simbolo, detto segno, ed un numero naturale, detto modulo** (o valore assoluto)
- Molti anni fa si decise di **non** rappresentare i numeri come modulo e segno (1 bit) all'interno dei calcolatori
 - Questo avrebbe reso le reti che operano sulle cifre (leggermente) più complesse
 - È stata adottata una rappresentazione **non intuitiva**, detta rappresentazione **in complemento alla radice**, che ormai non può che essere mantenuta per compatibilità

Rappresentazione dei numeri interi (cont.)

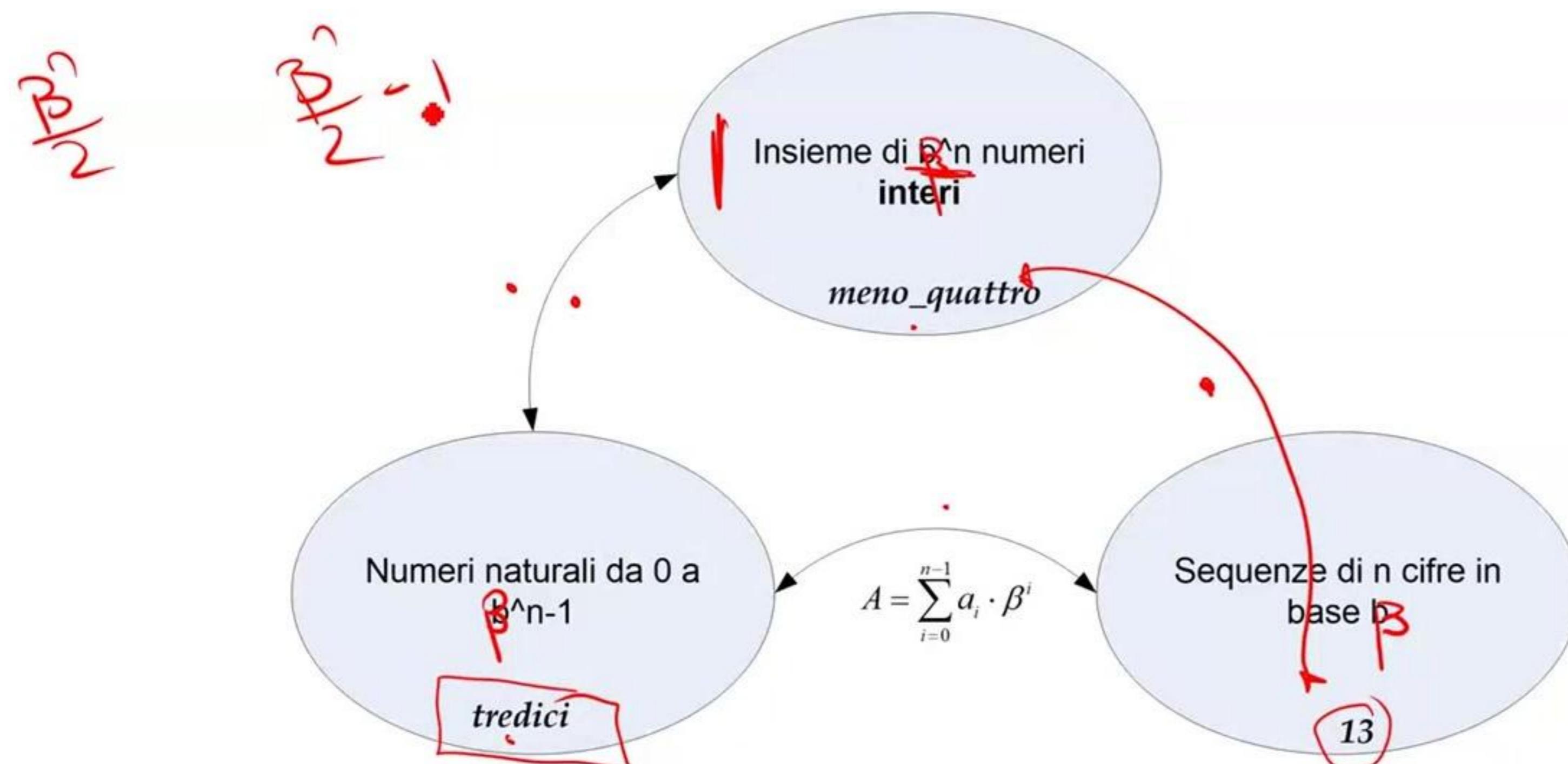
- Sequenze di n cifre in base β (β^n combinazioni diverse).
- Conosco una legge (biunivoca) che mi permette di associare a ciascuna di queste combinazioni un **numero naturale**, cioè:

$$A = \sum_{i=0}^{n-1} a_i \cdot \beta^i$$



Rappresentazione dei numeri interi (cont.)

- Preso un insieme di β^n numeri **interi**, posso **sempre trovare una legge biunivoca** che gli fa corrispondere un insieme di β^n numeri **naturali**



Rappresentazione dei numeri interi (cont.)

- Definisco una legge $L(\quad)$ da $\underline{\mathbb{Z}} \rightarrow \underline{\mathbb{N}}$, tale per cui, detto:
 - \underline{A} un numero naturale (li scrivo **maiuscoli** d'ora in avanti)
 - \underline{a} un numero intero (li scrivo **minuscoli** d'ora in avanti)

$$\underline{A} = L(a), a = L^{-1}(A)$$

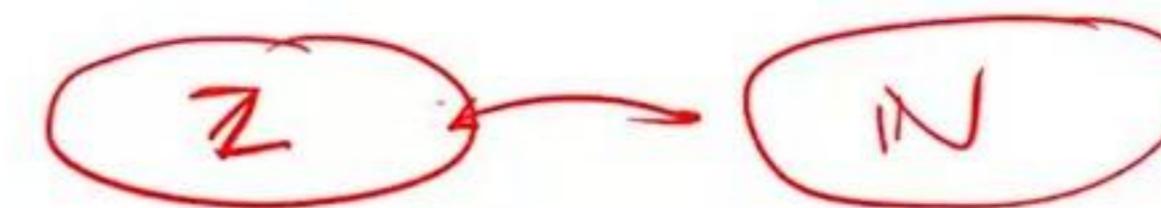
- E quindi posso scrivere:

$$a \stackrel{L}{\leftrightarrow} A \equiv (\underline{a_{n-1}a_{n-2}\dots a_1a_0})_\beta$$

- Quella a destra è la **rappresentazione del numero intero a su n cifre in base β** .

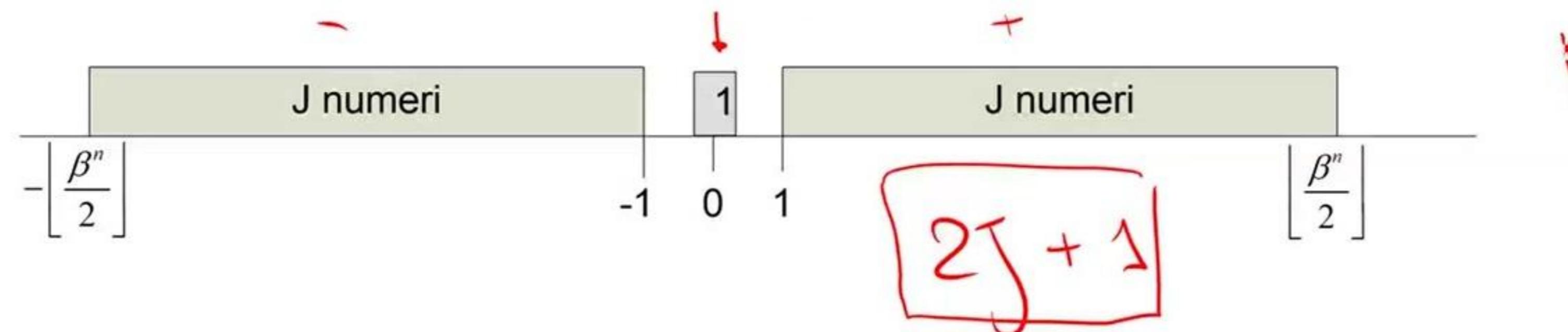
Rappresentazione dei numeri interi (cont.)

- Scegliendo opportunamente la legge $L(\underline{\quad})$ posso ottenere dei vantaggi implementativi
c2
- In particolare, esistono leggi $L(\underline{\quad})$ che consentono di **utilizzare i circuiti già visti** per le operazioni sui naturali anche per operare su rappresentazioni di interi
- Posso sfruttare questo per sintetizzare reti che producono risultati **corretti**
 - sia interpretando le cifre come rappresentazioni di numeri naturali
 - sia interpretandole come rappresentazioni di numeri interi



Rappresentazione dei numeri interi (cont.)

- Descriviamo la relazione biunivoca $L()$, dandone dominio, codominio e legge di corrispondenza
- Il dominio di $L()$ dovrà essere
 - Contiguo (privo di buchi). Dovrà quindi essere un intervallo (altrimenti non potrei calcolare il successivo di qualche numero)
 - Dovrà essere il più simmetrico possibile rispetto allo zero



- Possibile soltanto se β è dispari.

Rappresentazione dei numeri interi (cont.)

- Noi lavoreremo sempre con β pari a 2, 8, 10, 16.
- Se β è pari, dovremo accettare di rappresentare un numero positivo o negativo in più. La scelta che si opera in questo caso sarà di rappresentare l'intervallo di interi:

β^n



$$\left[-\frac{\beta^n}{2}, \frac{\beta^n}{2} - 1 \right]$$

intervallo di
rappresentabilità

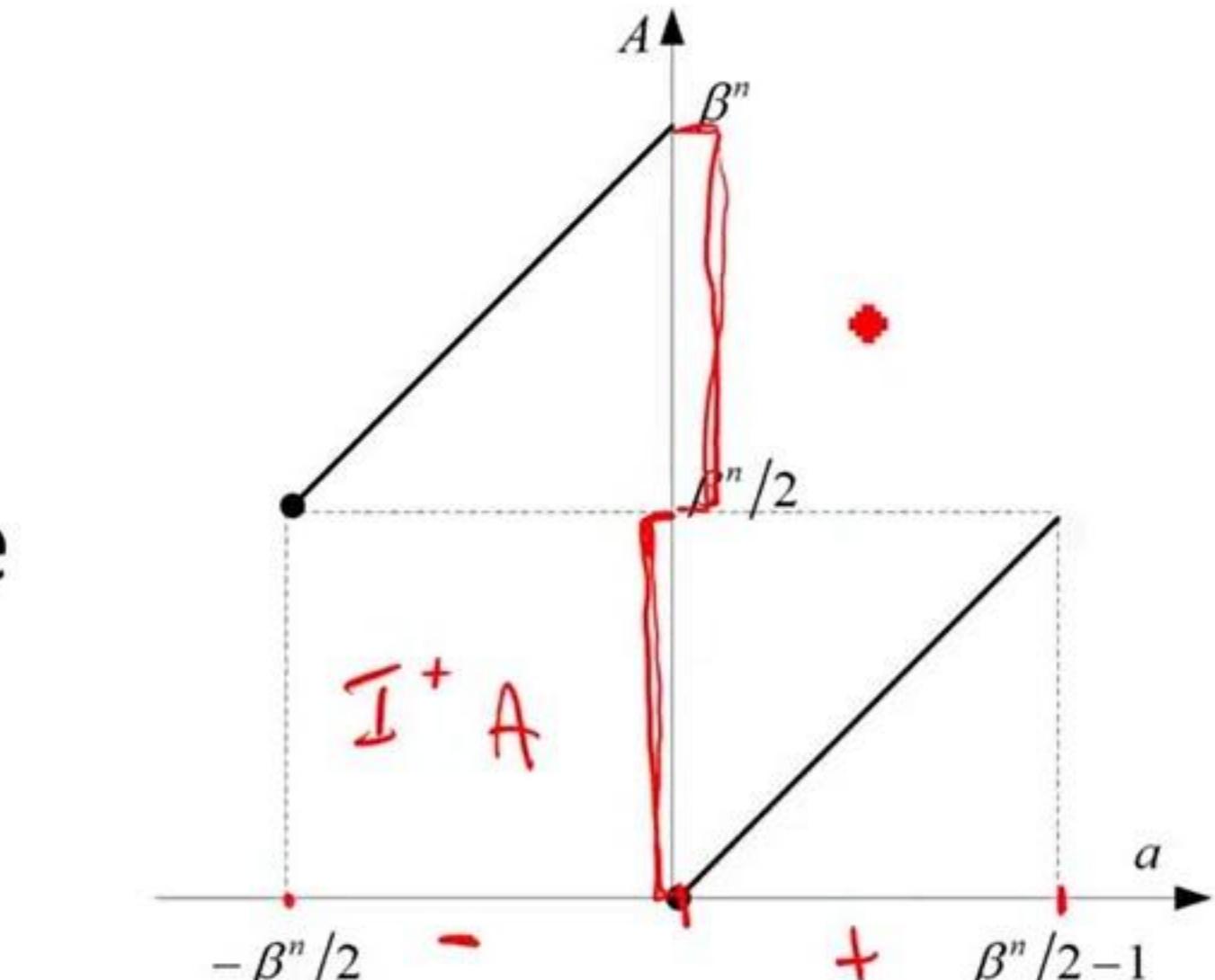
- Cioè di rappresentare un numero negativo in più

Determinazione del segno

- Posso determinare il segno di un numero intero \underline{a} dalle **cifre della sua rappresentazione A**

$$a \geq 0 \Leftrightarrow 0 \leq A < \frac{\beta^n}{2}$$

$$a < 0 \Leftrightarrow \frac{\beta^n}{2} \leq A < \beta^n$$



$$a \leftrightarrow A \equiv (a_{n-1}a_{n-2}\dots a_1a_0)_{\beta}$$

- Il massimo numero a rappresentabile è $\beta^n/2 - 1$
 - Quali sono le cifre della sua rappresentazione A in CR?



Legge di rappresentazione in traslazione

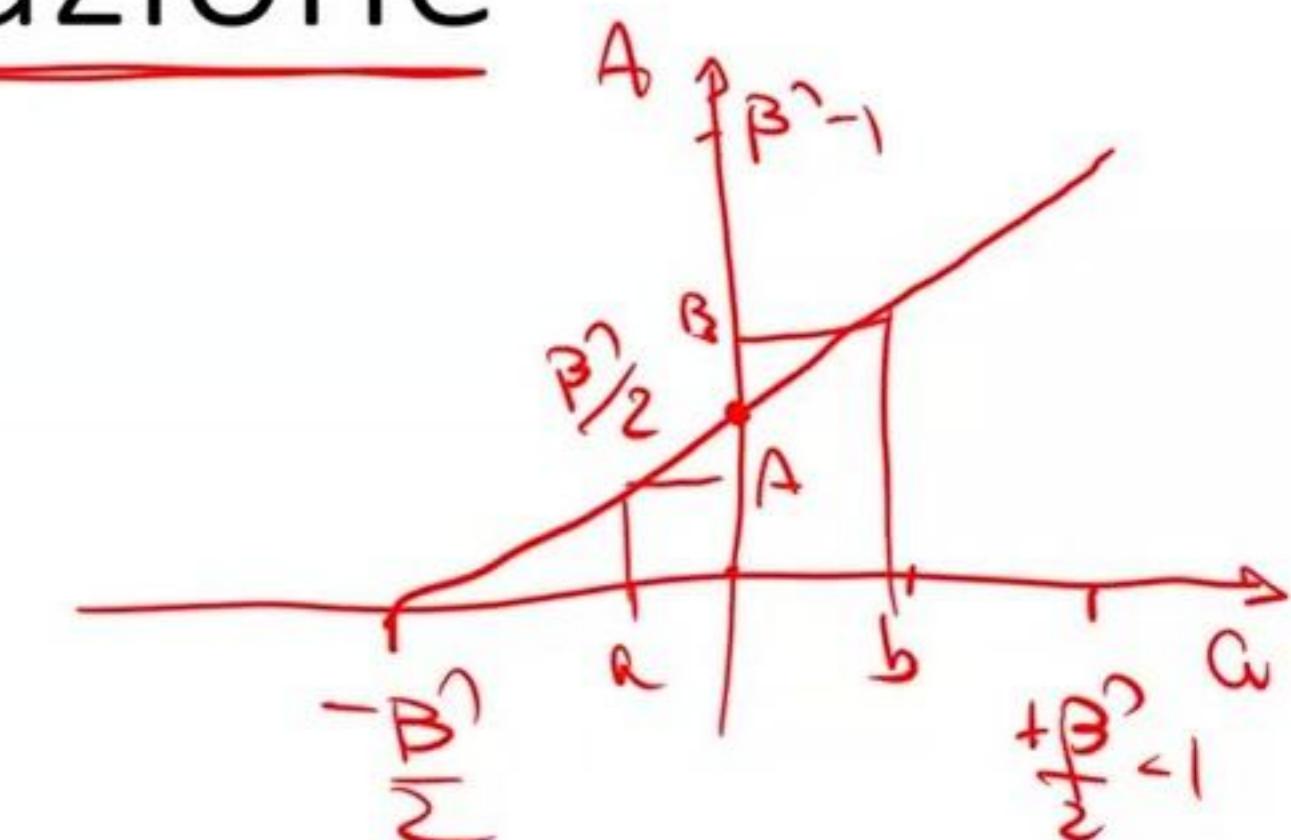
$$\left[-\frac{\beta^n}{2}; +\frac{\beta^n}{2} - 1 \right] \quad \left[-128; +127 \right]$$

\downarrow

$$[0; 255]$$

$$L: \quad A = a + \frac{\beta^n}{2}$$

- $\beta^n/2$ è detto **fattore di polarizzazione**.



- Es: $\beta = 2, n = 8$

a	-128	-127	-1	0	+1	+126	+127
A	0	1	...	127	128	129	254
rapp.	00000000	00000001	01111111	10000000	10000001	11111110	11111111

- monotona:** $a < b \Leftrightarrow A < B$
- Usata nei convertitori A/D e D/A, dove si chiama *binario bipolare* e nel rappresentare l'esponente dei numeri reali.

Complemento alla radice

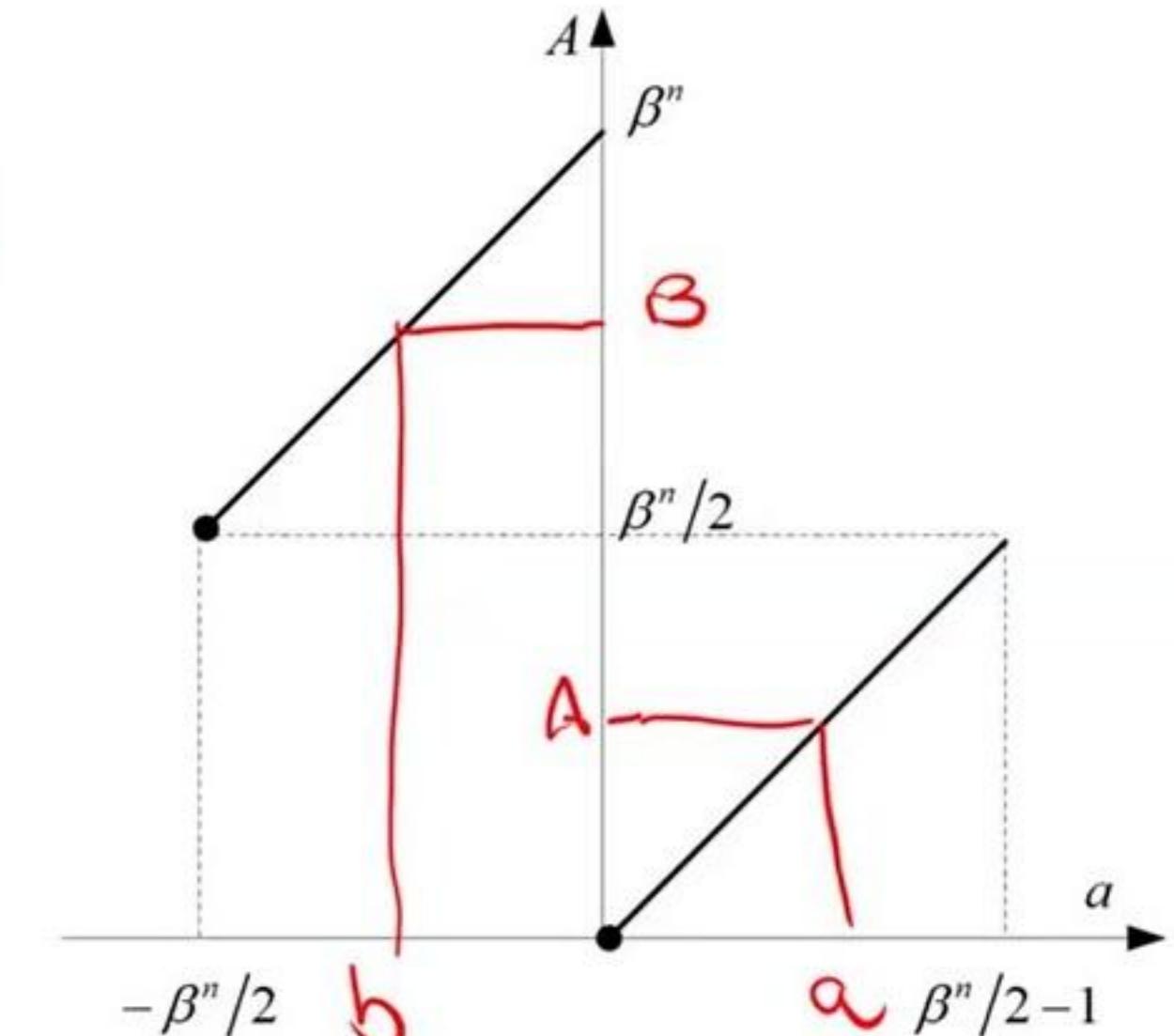
$$\left[-\frac{\beta^n}{2}; +\frac{\beta^n}{2} \right]$$

$$L: A = \begin{cases} a & 0 \leq a < \frac{\beta^n}{2} \\ \beta^n + a & -\frac{\beta^n}{2} \leq a < 0 \end{cases}$$

- Es: $\beta = 2, n = 8$

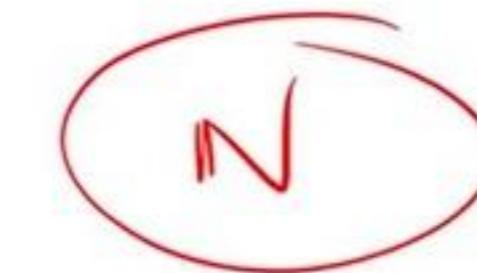
\downarrow $2^8 - 256$ \downarrow

a	-128	-127	-1	0	+1	+126	+127
A	128	129	255	0	1	126	127
rapp.	10000000	10000001	11111111	00000000	00000001	01111110	01111111



- Quella usata all'interno dei calcolatori.
- Non e' monotona

Modulo e segno



- Si fa corrispondere ad un numero intero una **coppia** costituita da **un numero naturale (modulo)** e da **una variabile logica (segno)**.

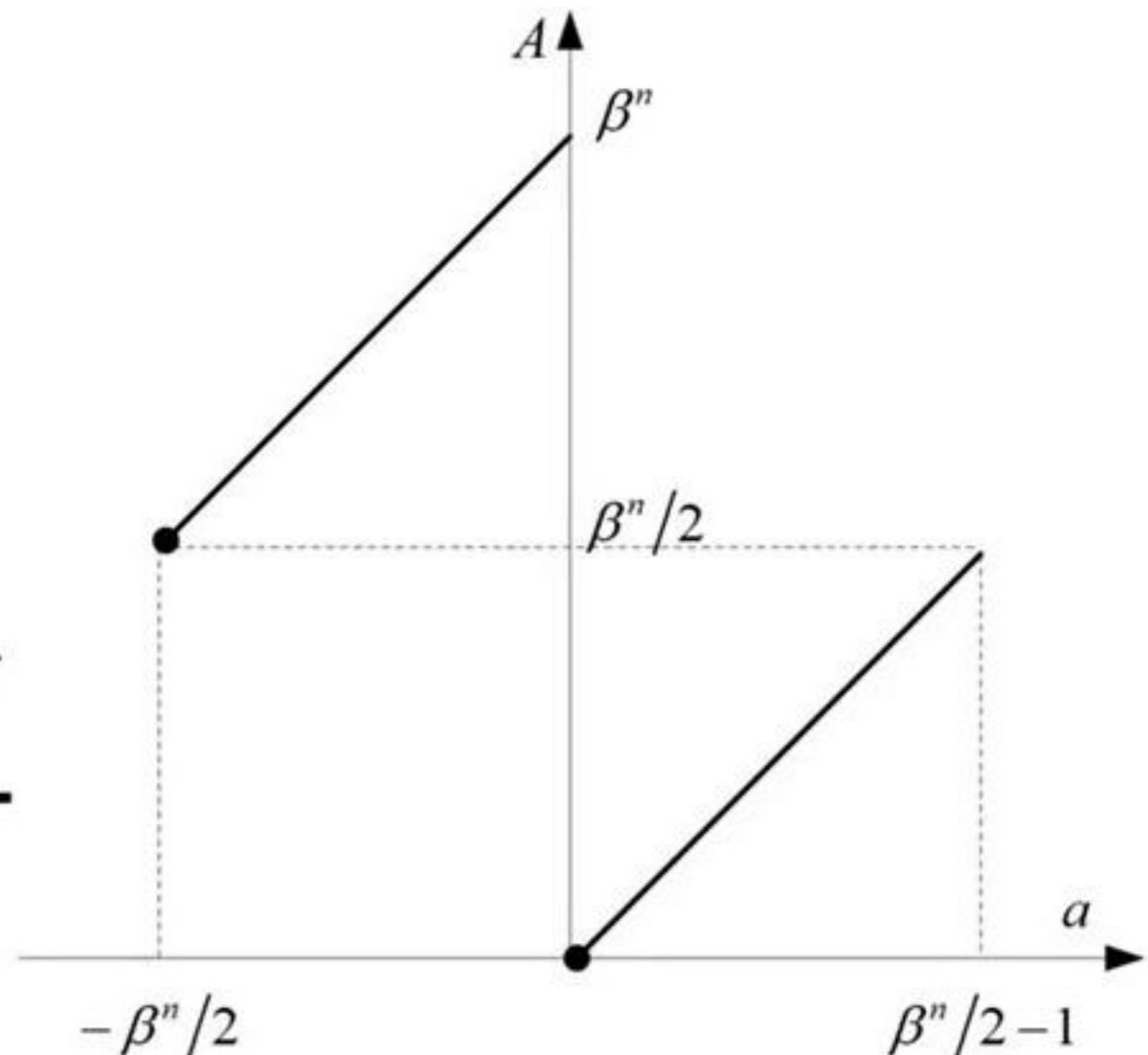
$$(s, M) \leftrightarrow a$$

$$s = \begin{cases} 0 & a \geq 0 \\ 1 & a < 0 \end{cases}, \quad M = \text{abs}(a).$$

- Questo tipo di rappresentazione non ricade nella categoria di prima.
- Infatti, non si mette in corrispondenza un intervallo di interi con un intervallo di naturali. •

Complemento alla radice

$$L: \quad A = \begin{cases} a & 0 \leq a < \frac{\beta^n}{2} \\ \beta^n + a & -\frac{\beta^n}{2} \leq a < 0 \end{cases}$$



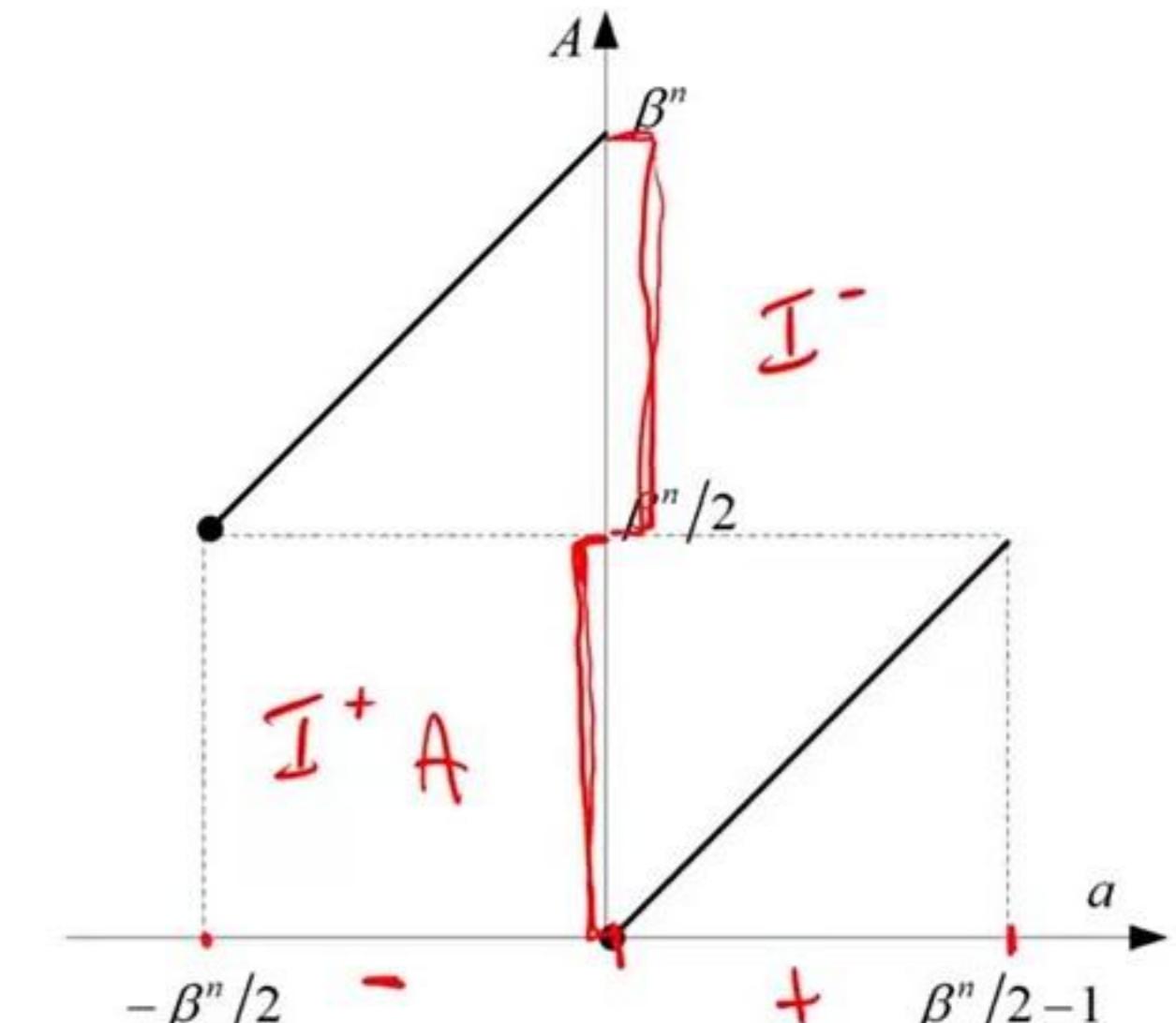
- D'ora in poi faremo riferimento a questa legge
 - La maggior parte delle proprietà vi sono familiari **in base 2**
 - Dobbiamo (ri)vederle in un'ottica generica

Determinazione del segno

- Posso determinare il segno di un numero intero \underline{a} dalle **cifre della sua rappresentazione A**

$$a \geq 0 \Leftrightarrow 0 \leq A < \frac{\beta^n}{2}$$

$$a < 0 \Leftrightarrow \frac{\beta^n}{2} \leq A < \beta^n$$



$$a \leftrightarrow A \equiv (a_{n-1}a_{n-2}\dots a_1a_0)_{\beta}$$

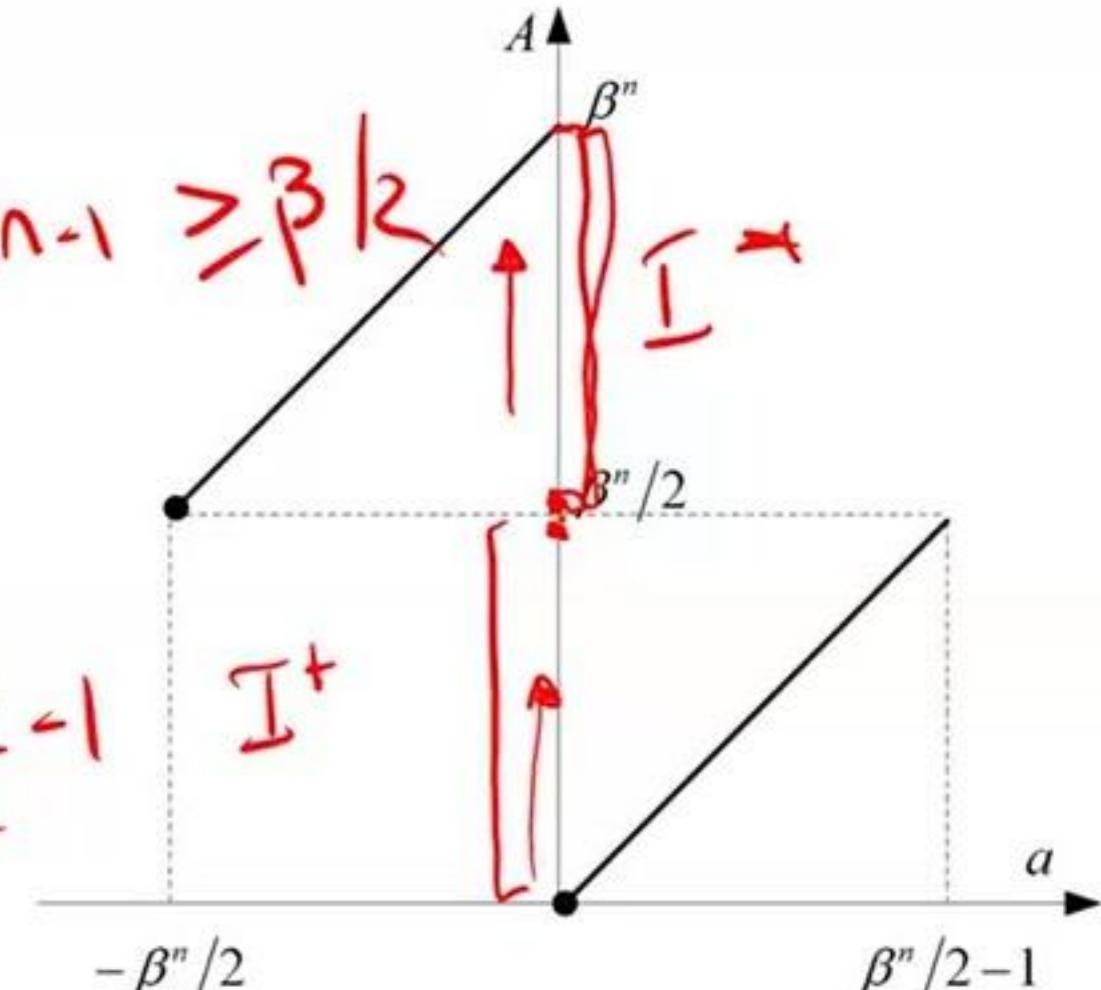
- Il massimo numero a rappresentabile è $\beta^n/2 - 1$
 - Quali sono le cifre della sua rappresentazione A in CR?



Determinazione del segno (cont.)

- La rappresentazione di $\beta^n/2$ si trova così'

$$a_{n-1} \leq \frac{\beta}{2} - 1$$



- Quindi

A

$$\underline{\beta^n/2} = \underline{\beta^{n-1}} \cdot \boxed{\beta/2}$$

$$\beta^n/2 \equiv (\beta/2 \ 00\dots 0)_\beta$$

B/2

$$A \leq \frac{\beta}{2} - 1$$

$$\rightarrow \beta^n/2 - 1 \equiv ((\beta/2 - 1)(\beta - 1)(\beta - 1)\dots(\beta - 1))_\beta$$

- Esempio:

- $\beta = 10, n = 4: \beta^n/2 - 1 \equiv (4999)_{10} = 10^4/2 - 1$

- $\beta = 2, n = 8: \beta^n/2 - 1 \equiv (01111111)_2 = 2^8/2 - 1$

$\frac{\beta}{2} - 1$

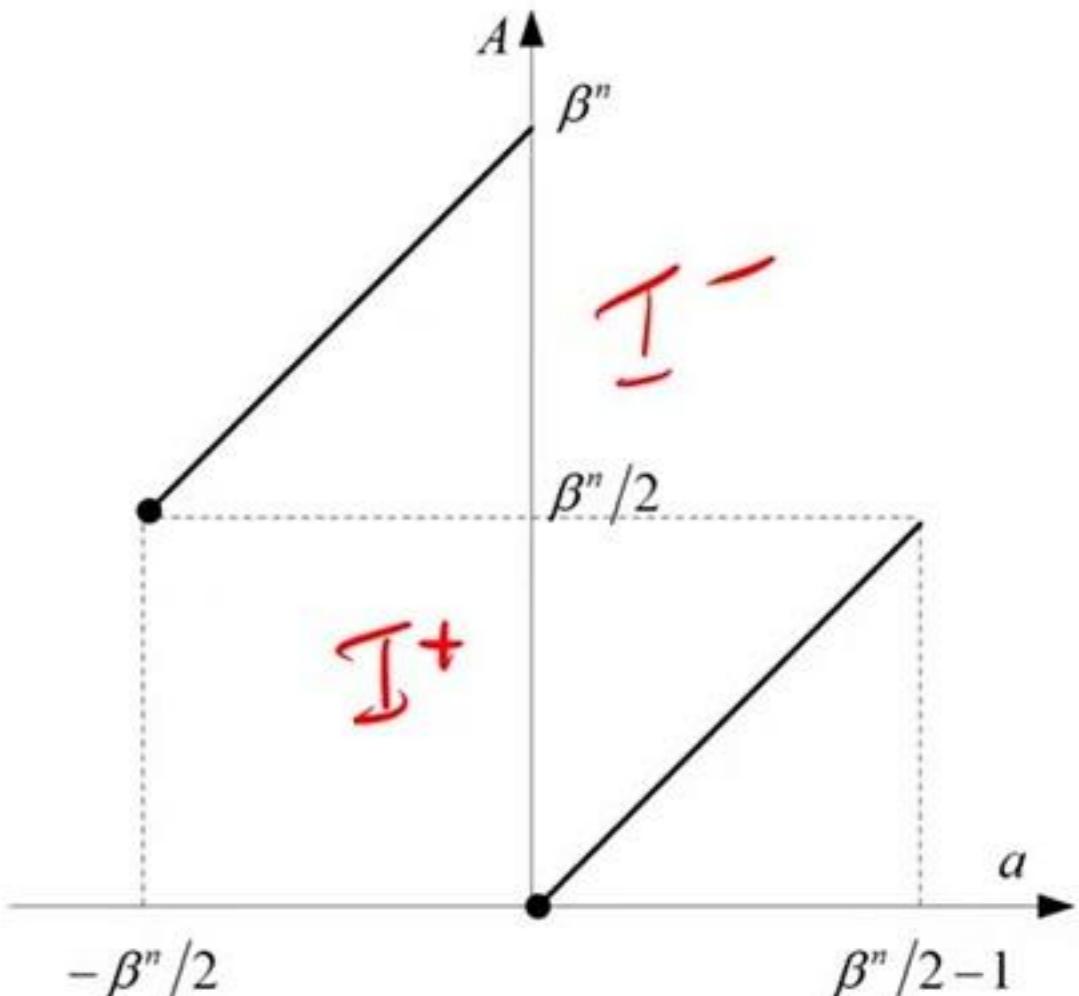
Determinazione del segno (cont.)

- Per capire se la rappresentazione A è un numero naturale **maggior o minore di $\beta^n/2$** basta **guardare la cifra più significativa**.
- Infatti:

$$a_{n-1} < \frac{\beta}{2} \Leftrightarrow 0 \leq A < \frac{\beta^n}{2} \quad I^+$$
$$a_{n-1} \geq \frac{\beta}{2} \Leftrightarrow \frac{\beta^n}{2} \leq A < \beta^n \quad I^-$$

- Quindi,

- i numeri naturali che hanno MSD $a_{n-1} < \frac{\beta}{2}$ rappresentano interi positivi
- I numeri naturali che hanno MSD $a_{n-1} \geq \frac{\beta}{2}$ rappresentano interi negativi



Determinazione del segno (cont.)

- In **base 2**, ritroviamo una legge che gia' conoscevamo

$$\begin{aligned} a \geq 0 &\Leftrightarrow a_{n-1} = 0 \\ a < 0 &\Leftrightarrow a_{n-1} = 1 \end{aligned}$$

$$a_n < \beta$$

$$a_n = 0$$

$$a_{n-1} \geq \beta$$

$$a_{n-1} = 1$$

Legge inversa

$a \leftarrow A$

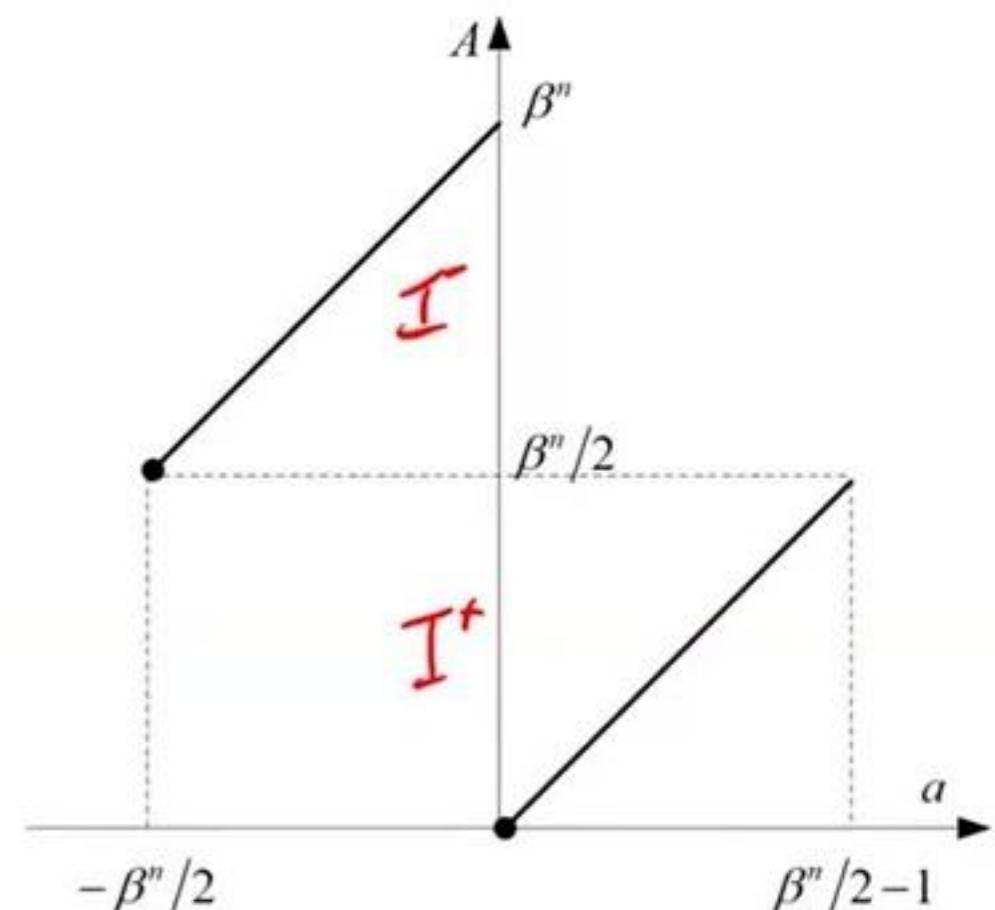
- Si ottiene per sostituzione

$$L: A = \begin{cases} a & 0 \leq a < \frac{\beta^n}{2} \\ \beta^n + a & -\frac{\beta^n}{2} \leq a < 0 \end{cases}$$

$$\Leftrightarrow L^{-1}: a = \begin{cases} A & a_{n-1} < \frac{\beta}{2} \\ -(\bar{A} + 1) & a_{n-1} \geq \frac{\beta}{2} \end{cases}$$

~~$\bar{A} = \beta^{n-1} - A$~~

$A \in \mathcal{I}^+$



- Nella formula di destra posso:

- semplificare le condizioni su A , che possono essere sostituite da condizioni sulla sua MSD a_{n-1}
- sostituire $A - \beta^n$ facendo leva sulla definizione di **complemento**

Esempi

$$\beta = 10, \quad n = 3, \quad A \equiv (852)_{10}$$

- MSD > $\beta/2 - 1 = 4$, il numero rappresentato è **negativo**.
 - Devo calcolare $\bar{A} \equiv (147)_{10}$, sommare uno e cambiare il segno: $a = -148$
- $A \equiv (\underline{\underline{500}})_{10}$. $a = -(499 + 1) = -500$

- $\beta = 2, \quad n = 4$
- $A \equiv (1011)_2$. MSD=1, il numero è negativo.
 - Quindi: $a = -(0100 + 1)_2 = -(101)_2 = -5$
 - $A \equiv (\underline{\underline{1111}})_2$. MSD=1, il numero è negativo.
 - Quindi: $a = -(0000 + 1)_2 = -(1)_2 = -1$

$$a = \begin{cases} A & a_{n-1} < \beta/2 \\ -(\bar{A} + 1) & a_{n-1} \geq \beta/2 \end{cases}$$

Forma alternativa per L

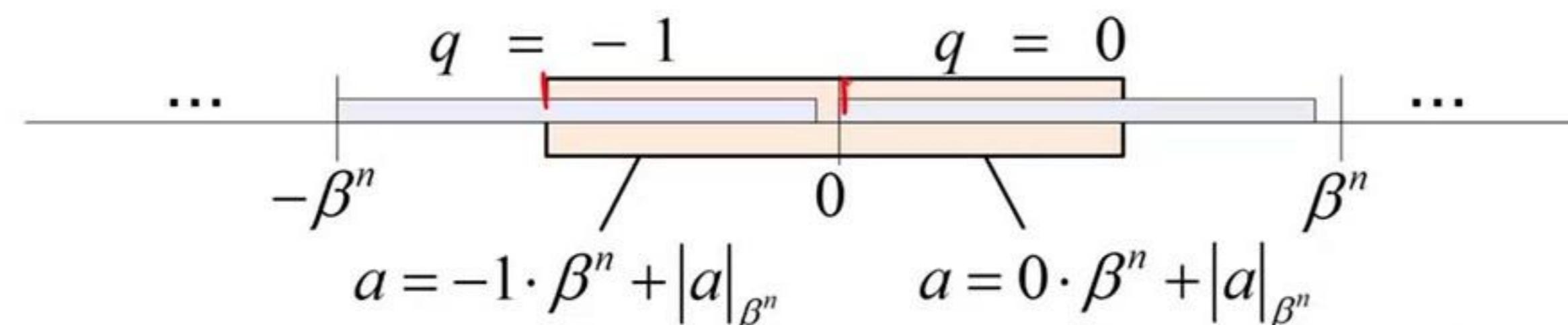
$$L: A = \begin{cases} a & 0 \leq a < \frac{\beta^n}{2} \\ \beta^n + a & -\frac{\beta^n}{2} \leq a < 0 \end{cases} \Leftrightarrow A = |a|_{\beta^n} \quad \begin{array}{l} \text{se } -\frac{\beta^n}{2} \leq a < \frac{\beta^n}{2} \\ \text{se } a < -\frac{\beta^n}{2} \end{array}$$

$\bullet a$ positivo $A = \beta^n + a = \beta^n + [-1 \cancel{\beta^n} + |a|_{\beta^n}]$

- allora è anche minore di β^n , e quindi $A = a = |a|_{\beta^n}$.

$\bullet a$ negativo

- allora è compreso in $[-\beta^n/2, 0]$, e quindi se lo divido per β^n ottengo quoziente -1
- Cioè, $a = -1 \cdot \beta^n + |a|_{\beta^n}$, ma allora si ottiene ugualmente $A = |a|_{\beta^n}$



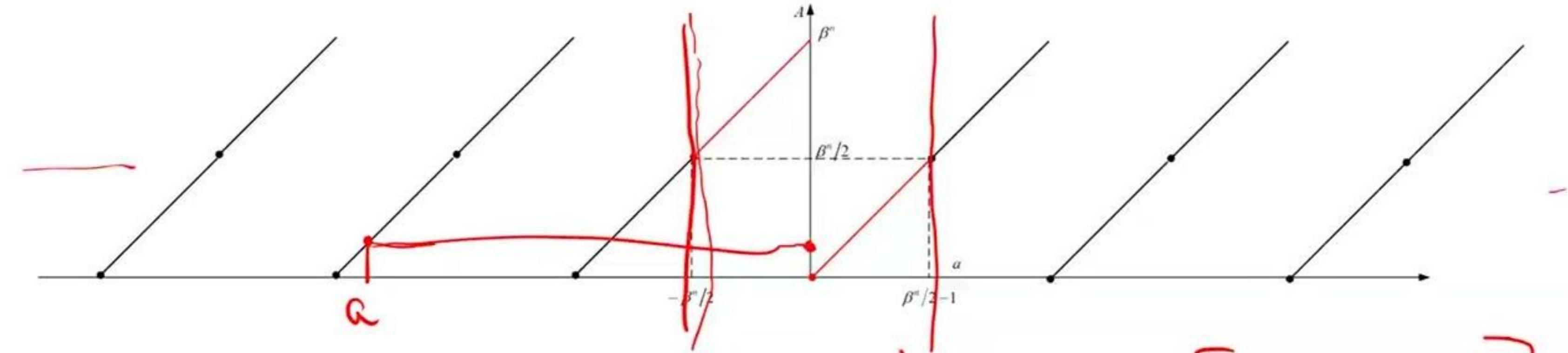
Forma alternativa per L (cont.)

$$L: A = \begin{cases} a & 0 \leq a < \frac{\beta^n}{2} \\ \beta^n + a & -\frac{\beta^n}{2} \leq a < 0 \end{cases} \Leftrightarrow \boxed{A = |a|_{\beta^n}} \text{ se } -\frac{\beta^n}{2} \leq a < \frac{\beta^n}{2}$$

$\left[0 \beta^{-1}\right] \quad a \in \mathbb{Z}$

- **Attenzione:** quanto appena scritto è vero **solo se** a è rappresentabile, cioè se appartiene a $[-\beta^n/2, \beta^n/2 - 1]$.
- È facile scordarselo, e questa cosa è fonte di **errori gravi**.
- In particolare, se a non appartiene all'intervallo scritto sopra, $|a|_{\beta^n}$ esiste ed è un numero su n cifre in base β , ma quel numero non ha niente a che vedere con la rappresentazione di a in complemento alla radice (che invece non esiste su n cifre in base β)

Esempio



- $\beta = 10, n = 3, a = -953.$

$$A = |a|_{\beta^n}$$

$$[-500; +499]$$

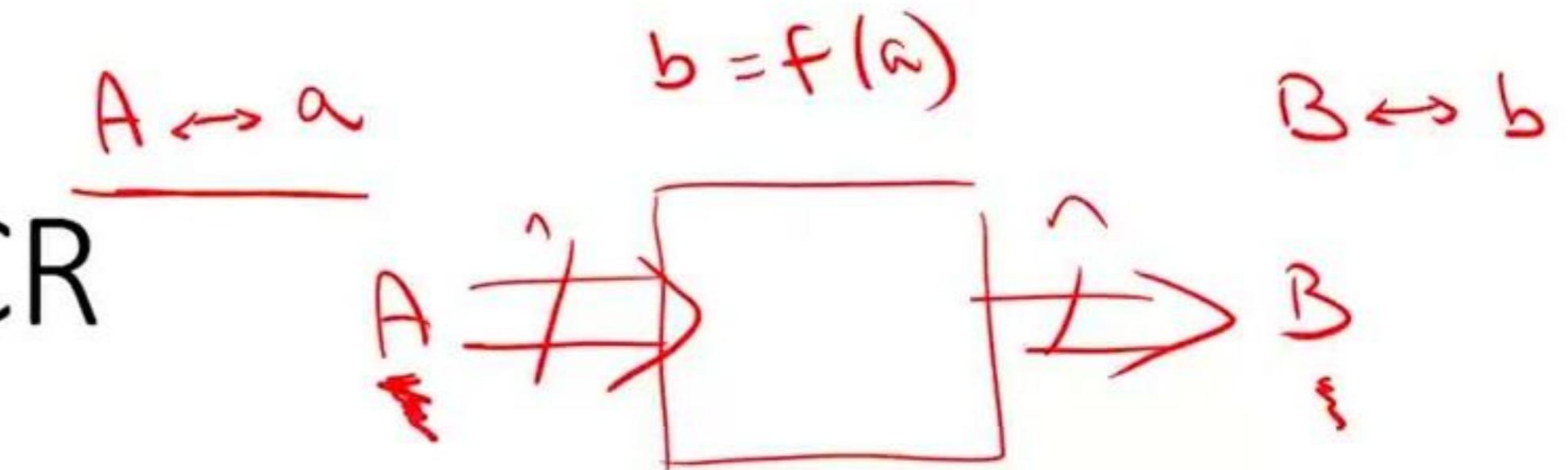
- $|a|_{\beta^n} = |-953|_{10^3} = \underline{47}.$

$$\left[-\frac{10^3}{2}; +\frac{10^3}{2} - 1 \right]$$

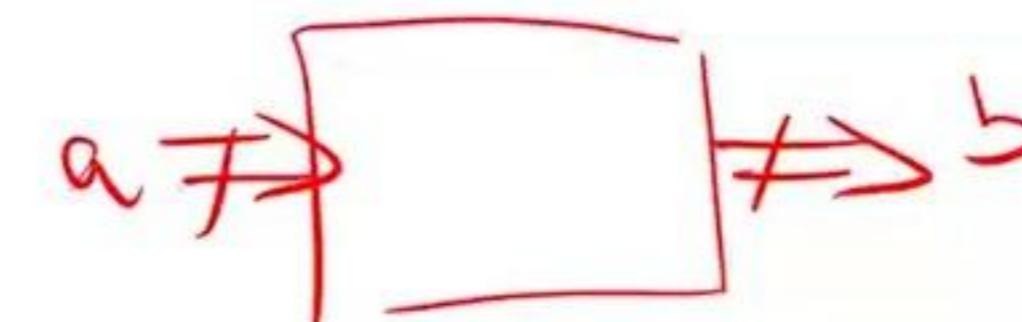
$$|a|_{\beta^n}, |-953|_{10^3} = \underline{9847}$$

- Ma 47 **non è** la rappresentazione di a in complemento alla radice su tre cifre
- Su tre cifre posso rappresentare soltanto i numeri nell'intervallo $[-500, +499]$, ed a **non appartiene** a questo intervallo.
- Il numero $A = 47$ è la rappresentazione (in complemento alla radice, su 3 cifre in base 10) dell'intero $a = +47$, non dell'intero $a = -953$

Operazioni su interi in CR



- Vogliamo progettare circuiti che **lavorano sulle rappresentazioni**
- hanno in ingresso **cifre in base β** e producono in uscita **cifre in base β**
- Interpretando le cifre secondo la legge di rappresentazione in *complemento alla radice*, questi circuiti eseguiranno operazioni significative (e.g., somma, sottrazione, etc.) su numeri interi.
- È importante sottolineare che i **circuiti vedono solo cifre**. Il legame tra quelle cifre ed i numeri interi che queste rappresentano **sta soltanto nella nostra mente**.



Valore assoluto

$$\beta^{n-1} - 1 = \tilde{\beta}^{n-1} - 1$$

- Vogliamo trovare il numero **naturale** $B = ABS(a)$.
- $a \in [-\beta^n/2, \beta^n/2 - 1] \rightarrow B \in [0, \beta^n/2]$
- B è un numero naturale rappresentabile su n cifre
 - attenzione: $n - 1$ non bastano, neanche se $\beta = 2$
- Vogliamo quindi disegnare un circuito che
 - prende in ingresso $A \equiv (\underline{a_{n-1}, \dots, a_0})_\beta$
 - produce in uscita $B \equiv (\underline{b_{n-1}, \dots, b_0})_\beta$
 - con $a \leftrightarrow A$ e $B = ABS(a)$



Valore assoluto (cont.)



$$ABS(a) = \begin{cases} a & a \geq 0 \\ -a & a < 0 \end{cases}$$

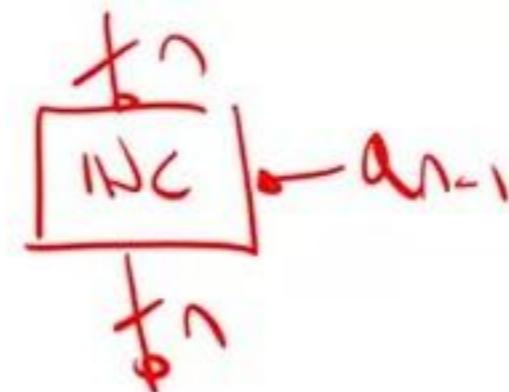
- Conosco un modo semplice per stabilire il segno di a guardandone la rappresentazione A , ed un modo semplice per calcolare $-a$ usando il complemento della sua rappresentazione.

$$B = ABS(a) = \begin{cases} A & \underline{a_{n-1} < \beta/2} \\ \underline{\bar{A} + 1} & \underline{a_{n-1} \geq \beta/2} \end{cases}$$

$\alpha \geq 0$
 $\alpha < \phi$

$L^{-1}: \underline{-a} = \begin{cases} A & a_{n-1} < \frac{\beta}{2} \\ -(\bar{A} + 1) & \underline{a_{n-1} \geq \frac{\beta}{2}} \end{cases}$

Valore assoluto (cont.)



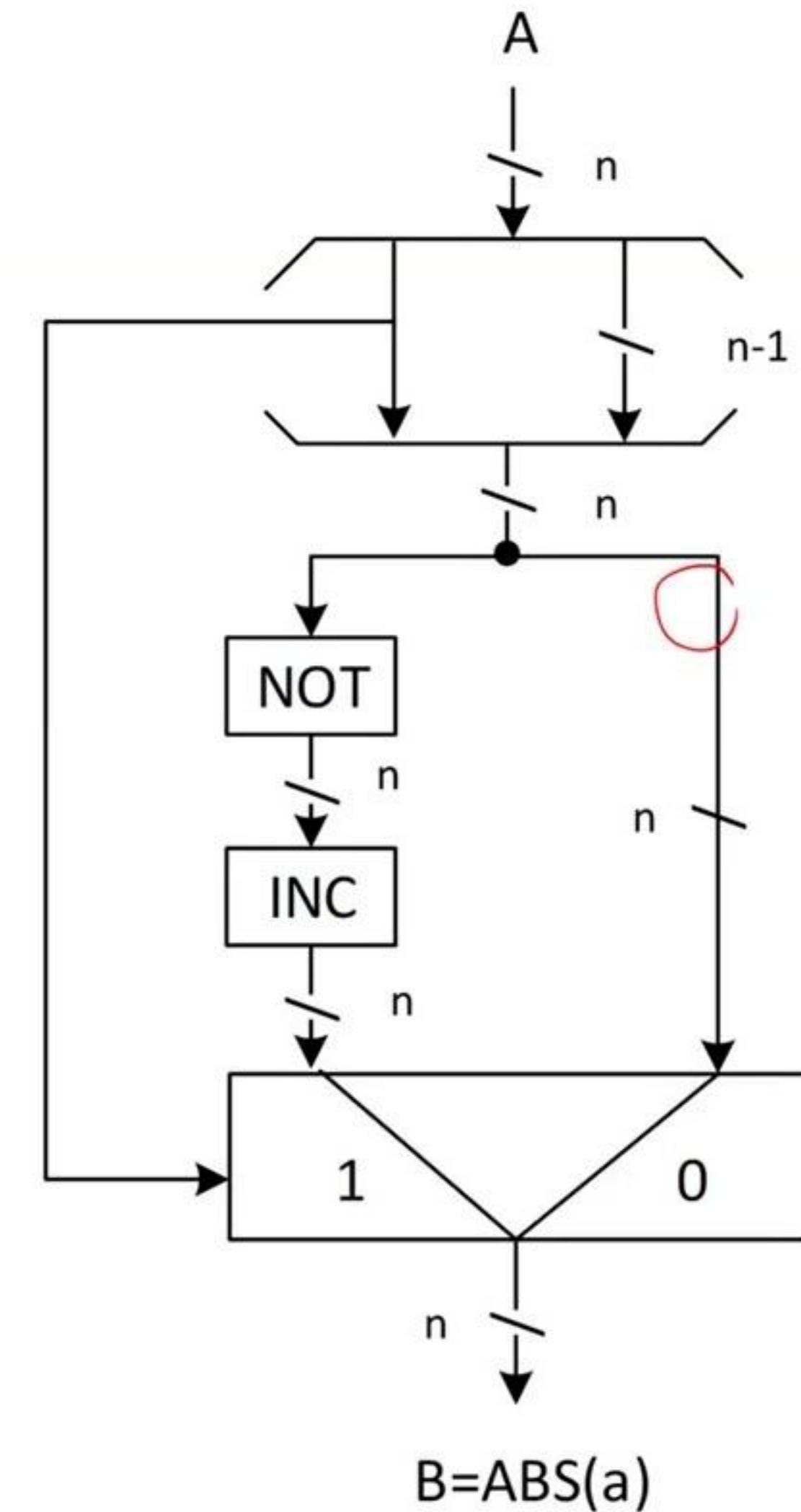
$$\alpha \text{ XOR } 1 = \bar{\alpha}$$

$$B = \begin{cases} A & a_{n-1} < \beta/2 \\ \overline{A} + 1 & a_{n-1} \geq \beta/2 \end{cases}$$

$a_{n-1} = \emptyset$

$a_{n-1} = 1$

- Per sintetizzare il circuito servono:
 - Un multiplexer, per discriminare i due rami dell'espressione;
 - la variabile di comando del multiplexer sarà data dal confronto tra i naturali $a_{n-1}, \beta / 2$ (comparatore); flag-min
 - Un circuito che calcola complemento ed incremento, per il ramo inferiore
- In base 2 è notevolmente più semplice



Conversione da CR a MS

- Completa il circuito appena visto



- Intervallo di rappresentabilità per MS: $[-(\beta^n - 1); +(\beta^n - 1)]$
 - Contiene quello in CR: $[-\frac{\beta^n}{2}; +\frac{\beta^n}{2} - 1] \subset [-(\beta^n - 1); +(\beta^n - 1)]$
 - La conversione è sempre fattibile

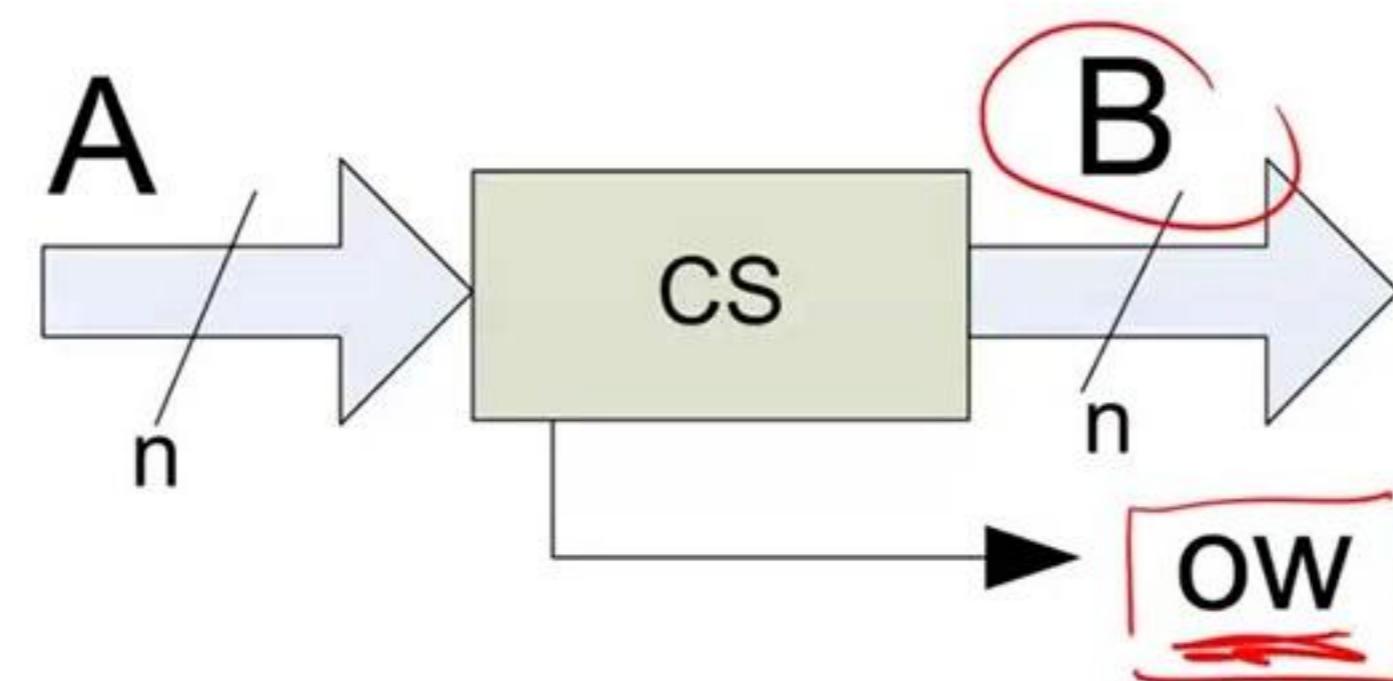
$$sgn_a = \begin{cases} 0 & a_{n-1} < \beta/2 \\ 1 & a_{n-1} \geq \beta/2 \end{cases}$$

)

Calcolo dell'opposto

~~B > A~~

- Dato $A \leftrightarrow a$, voglio trovare $B \leftrightarrow b$ tale che $b = -a$.
- È un'operazione non sempre possibile se decido di rappresentare a e b sullo stesso numero di cifre.
- Infatti, $a \in [-\underline{\beta^n}/2, \beta^n/2 - 1]$,
- Esiste un intero (negativo) che non ha un opposto nell'intervallo



Calcolo dell'opposto (cont.)

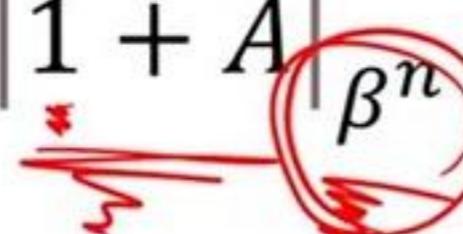
$$\begin{aligned} B &\leftarrow b \quad b = -a \\ B &= |b|_{\beta^n} = |-a|_{\beta^n} \end{aligned}$$

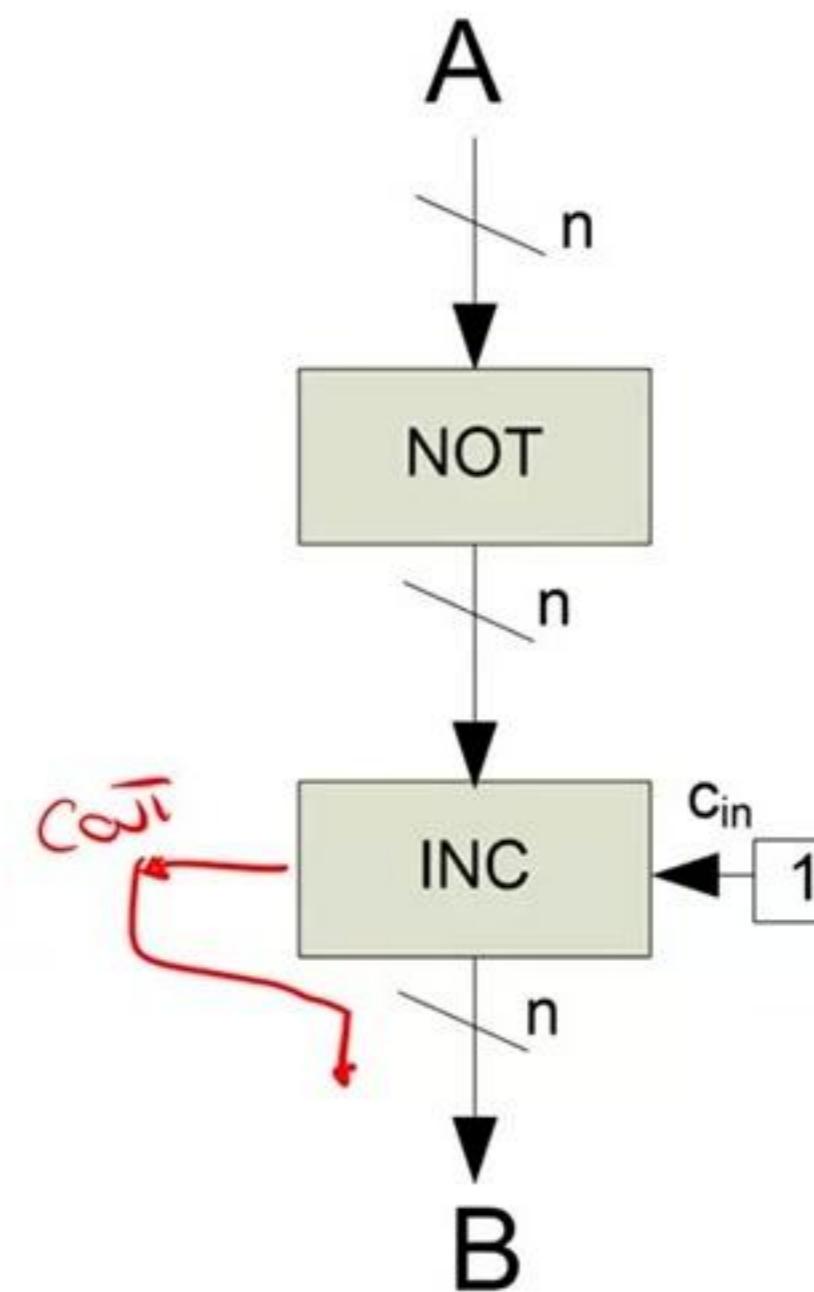
- Le due uscite vanno sintetizzate separatamente

- Per sintetizzare B, si assume che $ow=0$, cioè che $a \neq -\beta^n/2$

$$\begin{aligned} B &\triangleq \downarrow | -a |_{\beta^n} \\ &= \left| | -1 |_{\beta^n} \cdot \underline{| a |_{\beta^n}} \right|_{\beta^n} = | (\beta^n - 1) \cdot A |_{\beta^n} \\ &= | \cancel{\beta^n} \cdot A - A |_{\beta^n} = | -A |_{\beta^n} \\ &= | -\cancel{\beta^n} + 1 + \bar{A} |_{\beta^n} = | 1 + \bar{A} |_{\beta^n} \end{aligned}$$

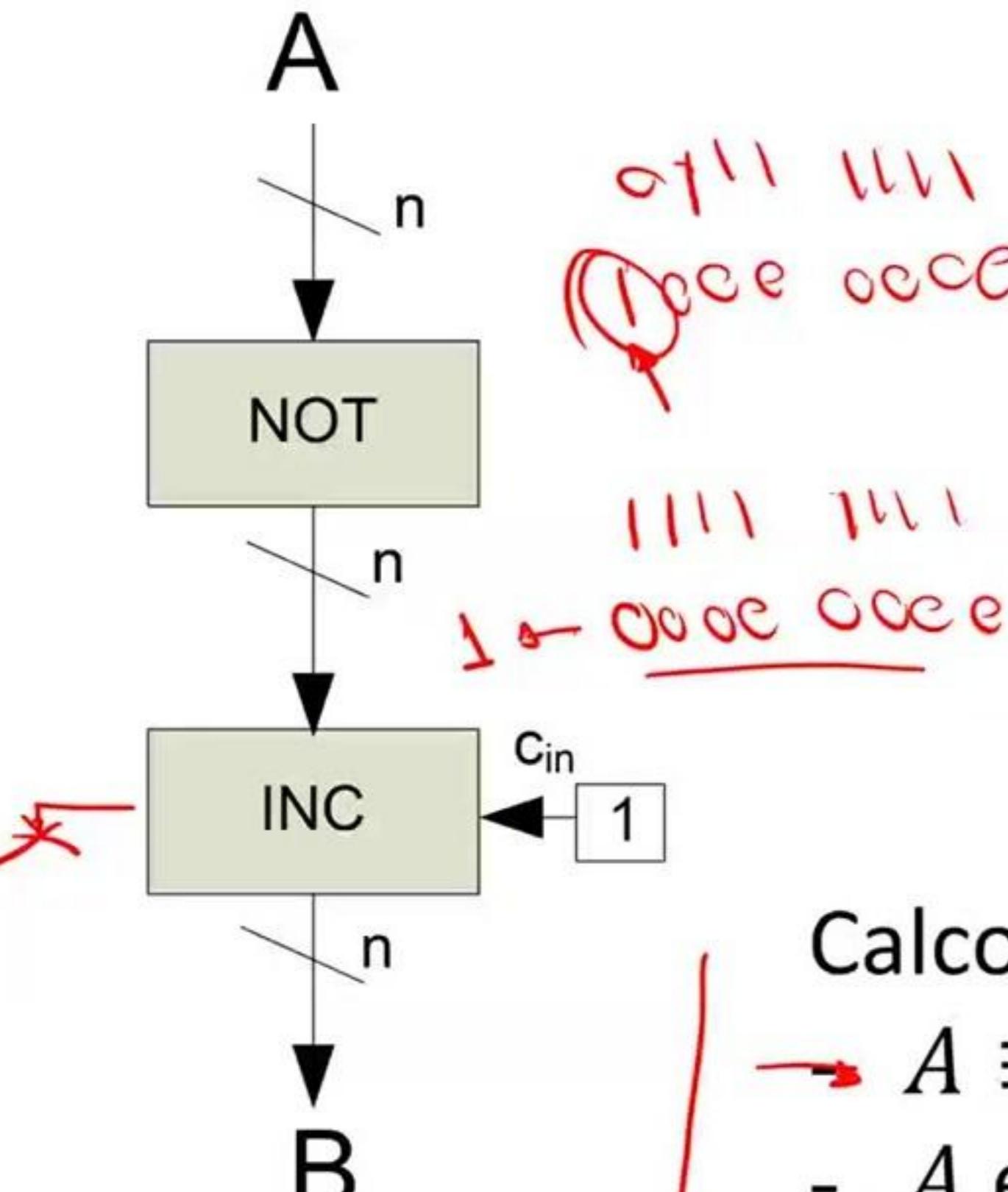
$\bar{A} = \beta^n - 1 - A$





Calcolo dell'opposto (cont.)

$$3=2, n=8$$



- $A = \cancel{1}0000000$
 - Uscita non corretta ($B = 10000000$).
 - $A \leftrightarrow -\beta^n/2$, e quindi l'opposto non è rappresentabile.
- $A = \cancel{0}0000000$
 - Riporto in uscita dall'INC.
 - Questo conferma che, nell'espressione di prima, il modulo serve.

Calcolo **overflow**:

- $A \equiv (\underline{10\dots 00})$, che richiede una AND a n ingressi.
- A e B hanno **segno negativo**, cioè se $\underline{a_{n-1}} = \underline{b_{n-1}} = 1$, Si può usare una **AND** a due ingressi.

Calcolo dell'opposto (cont.)

- **Richiamo sull'assembler:** esiste un'istruzione **NEG**, che interpreta una sequenza di bit come la rappresentazione di un numero **intero**, e produce la rappresentazione dell'opposto, **se questo esiste**.
- Altrimenti setta **il flag di overflow OF**.

Estensione di campo per numeri interi

- Dato il naturale $\underline{A} \equiv (a_{n-1}a_{n-2} \dots a_1a_0)_\beta$,
 - tale che $\underline{A} \leftrightarrow a$ in CR su n cifre
- Voglio trovare $\underline{A^{EST}} \equiv (a'_na'_{n-1}a'_{n-2} \dots a'_1a'_0)_\beta$,
 - tale che $\underline{A^{EST}} \leftrightarrow a$ in CR su $n + 1$ cifre
- Il problema ha sempre soluzione: l'intervallo di rappresentabilità su $\underline{n + 1}$ cifre contiene quello su n cifre



Estensione di campo per numeri interi (*cont.*)

- Calcoliamo A^{EST} prima per via algebrica:

$$L_{n+1}': \quad \overline{A^{EST}} = \begin{cases} a & 0 \leq a < \frac{\beta^n}{2} \\ \beta^{n+1} + a & -\frac{\beta^n}{2} \leq a < 0 \end{cases}$$

$$L_n^{-1}: \quad a = \begin{cases} A & a_{n-1} < \frac{\beta}{2} \\ -(\bar{A} + 1) = -\beta^n + A & a_{n-1} \geq \frac{\beta}{2} \end{cases}$$

$\beta^{n+1} + (-\beta^n + A)$
 $= \beta^n (\beta - 1) + A$

$$A^{EST} = \begin{cases} A & a_{n-1} < \frac{\beta}{2} \\ (\beta - 1) \cdot \beta^n + A & a_{n-1} \geq \frac{\beta}{2} \end{cases}$$

• $A^{EST} = \begin{cases} \underline{0 \cdot \beta^n + A} & a_{n-1} < \frac{\beta}{2} \\ \underline{(\beta - 1) \cdot \beta^n + A} & a_{n-1} \geq \frac{\beta}{2} \end{cases}$

Estensione di campo per numeri interi (cont.)

~~l'intero $a_{n-1} \dots a_0$~~

concatenamento

$$A^{EST} = \begin{cases} 0 \cdot \beta^n + A & a_{n-1} < \frac{\beta}{2} \\ (\beta - 1) \cdot \beta^n + A & a_{n-1} \geq \frac{\beta}{2} \end{cases}$$

- Si ricava una relazione esprimibile in termini di **cifre**:

$$\underline{a_i}' = a_i \quad 0 \leq i \leq n - 1,$$

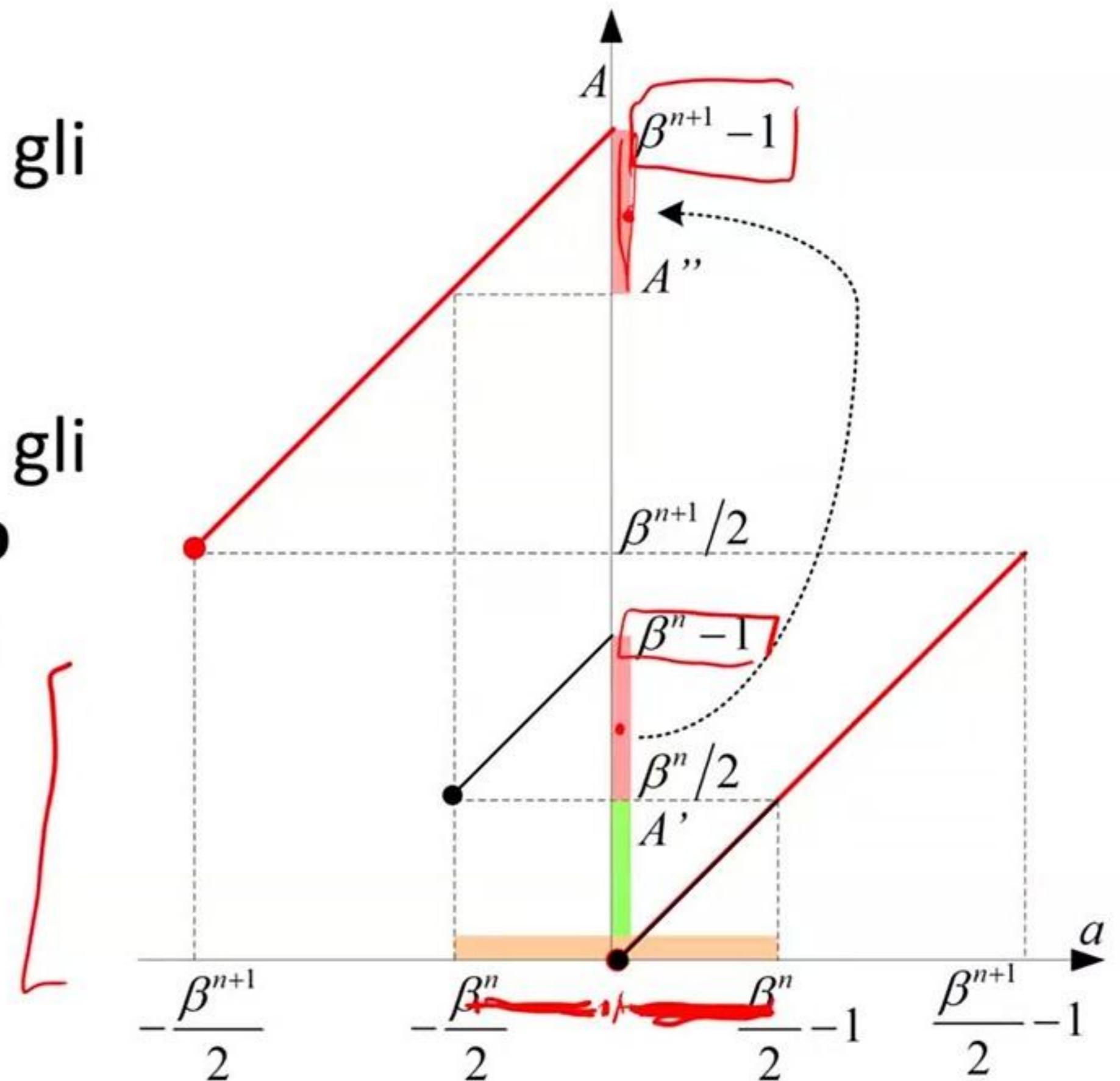
$$a_n' = \begin{cases} 0 & \parallel a_{n-1} < \beta/2 \\ \beta - 1 & \parallel a_{n-1} \geq \beta/2 \end{cases} .$$

~~B~~

- In generale, l'estensione di campo degli interi **richiede logica**

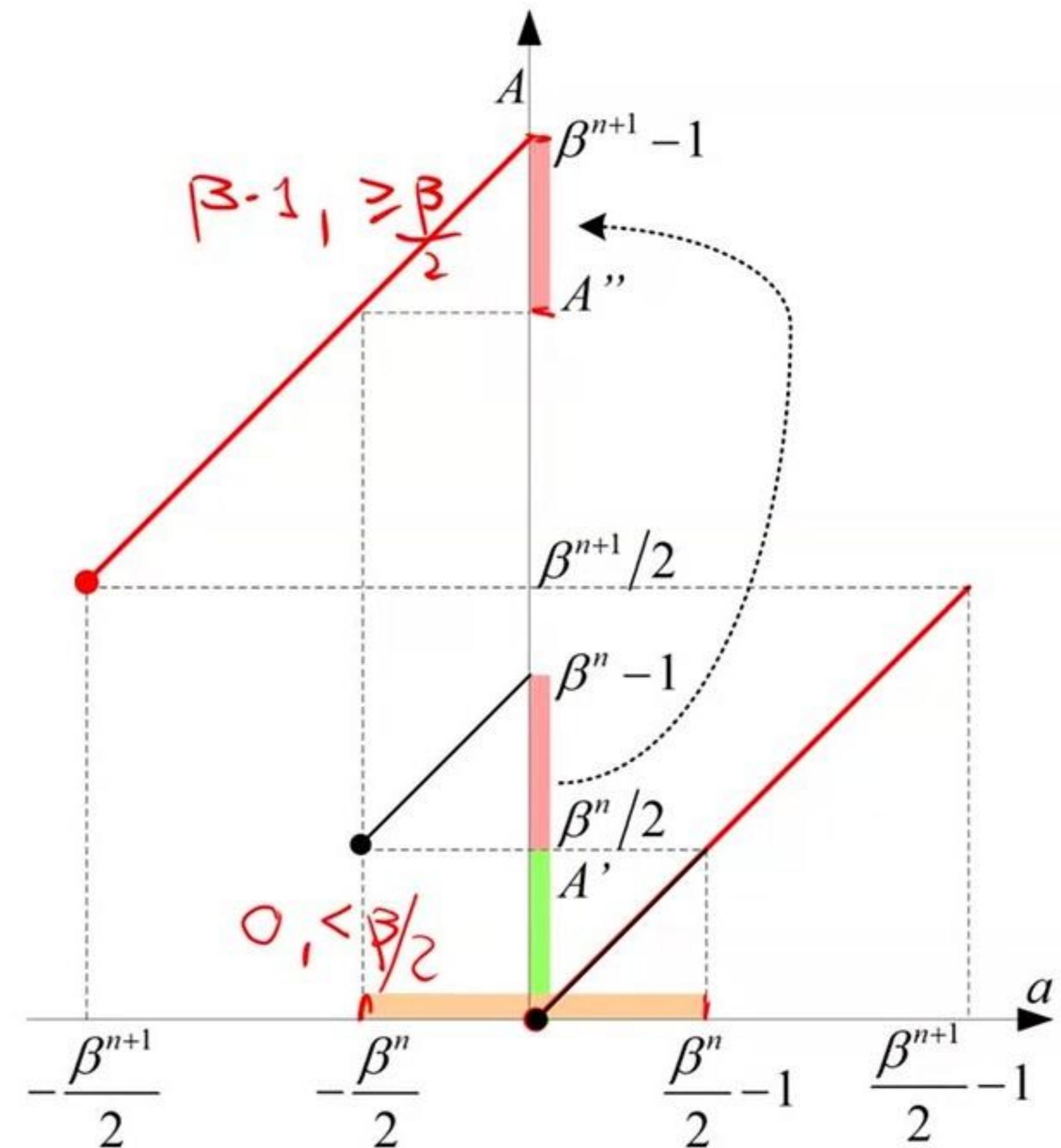
Estensione di campo dei numeri interi (cont.)

- L'intervallo di ordinate che rappresenta gli **interi positivi** deve rimanere invariato
 - aggiungo una MSD di peso nullo
- L'intervallo di ordinate che rappresenta gli **interi negativi** deve **traslare verso l'alto**
 - Di $(\beta^{n+1} - 1) - (\beta^n - 1) = \beta^n \cdot (\beta - 1)$
 - Aggiungo una **MSD** di peso $(\beta - 1)$



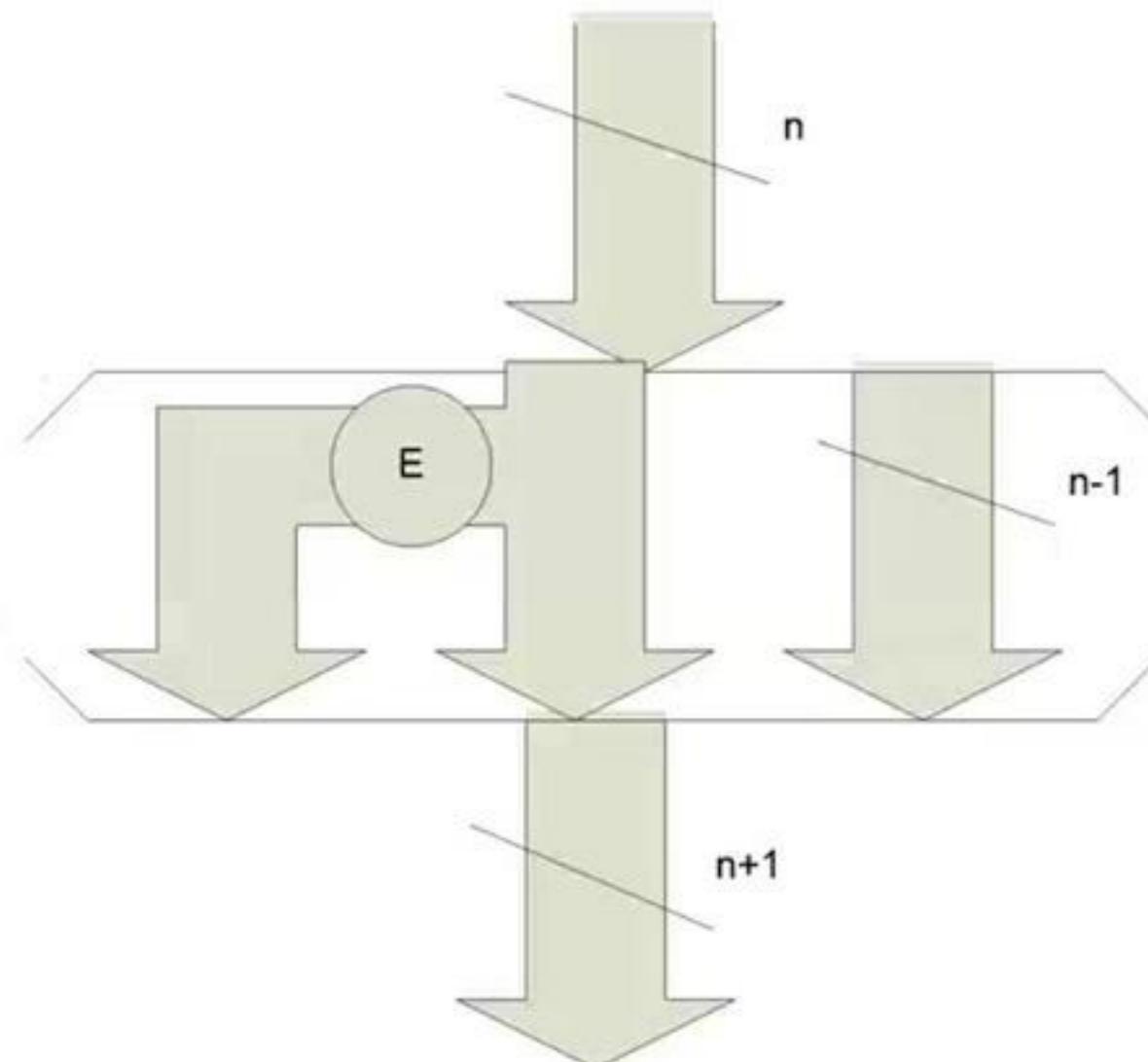
Estensione di campo dei numeri interi (cont.)

- Le rappresentazioni **estese** degli interi appartenenti all'intervallo $[-\frac{\beta^n}{2}, \frac{\beta^n}{2} - 1]$ saranno fatte così:
- Se rappresentazioni di numeri **positivi**
 - $MSD = 0$ AND cifra successiva $< \frac{\beta}{2}$
- Se rappresentazioni di numeri **negativi**
 - $MSD = \beta - 1$ AND cifra successiva $\geq \frac{\beta}{2}$



Estensione di campo dei numeri interi (cont.)

- L'estensione di campo dei numeri **interi**, a differenza di quella dei **naturali**, richiede in generale della logica
 - necessaria a discriminare il **segno** del numero intero rappresentato



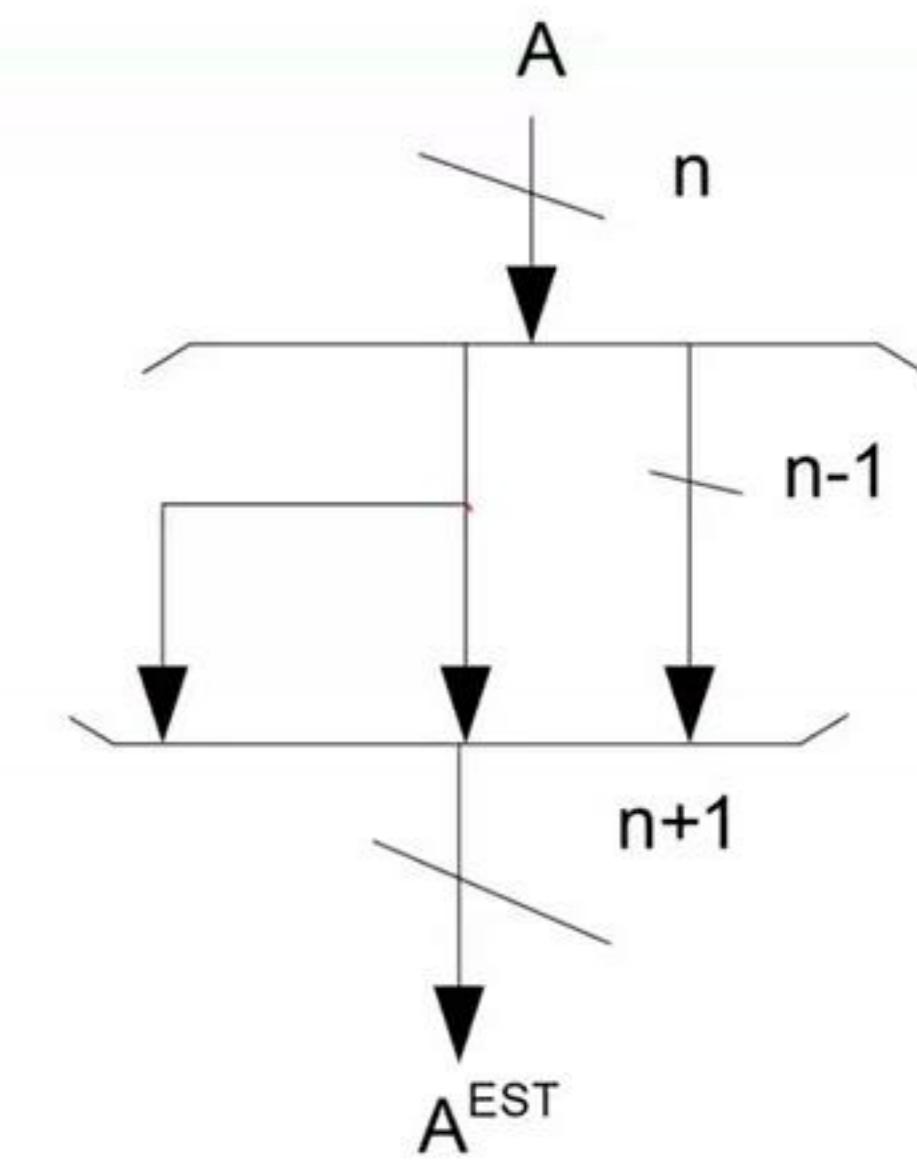
$$a_n' = \begin{cases} 0 & a_{n-1} < \beta/2 \\ \beta - 1 & a_{n-1} \geq \beta/2 \end{cases}$$

$a_{n-1} = 0$
 $a_{n-1} = 1$

In base 2, però, la relazione si semplifica:

$$\underline{\underline{a'_n = a_{n-1}}}$$

E non c'è bisogno di logica



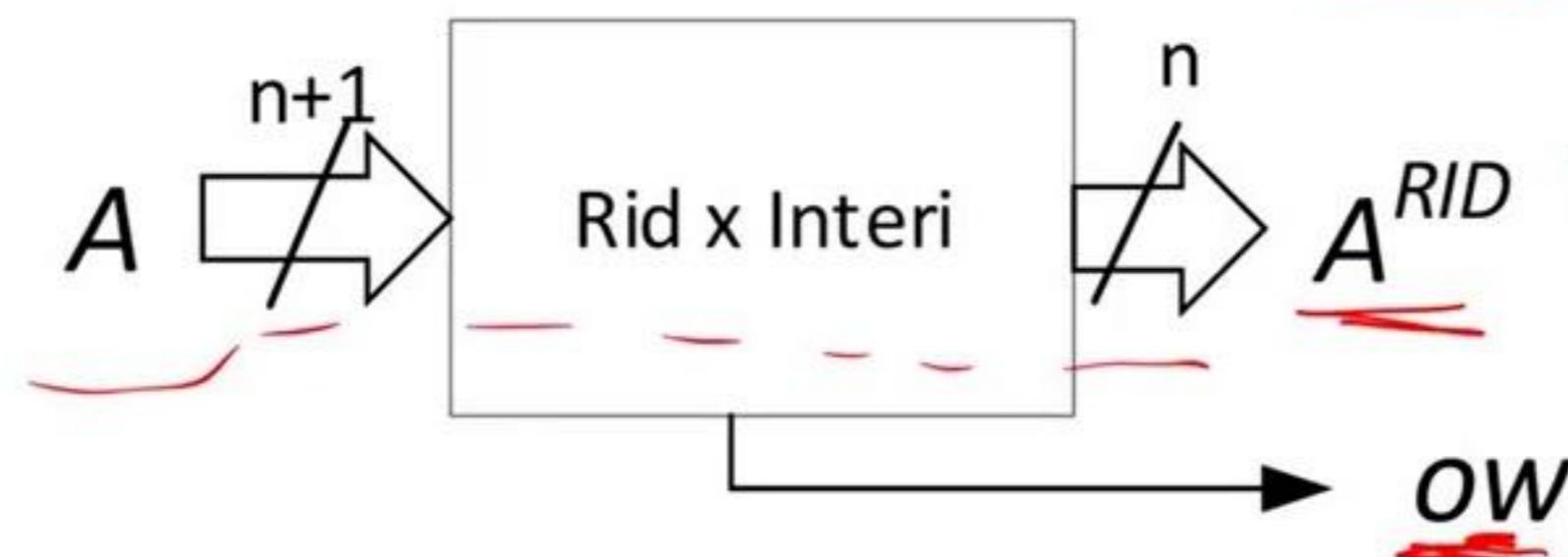
Estensione di campo dei numeri interi (cont.)

- **Richiamo sull'assembler:** esistono istruzioni **CBW, CWD, CDQ, CWDE**, che interpretano una sequenza di bit come la rappresentazione di un numero **intero** (su 8, 16, 32 bit), e producono la rappresentazione dello stesso numero **estesa su 16, 32, 64 bit**.
- Per contro, non esiste nessuna istruzione dedicata allo stesso scopo per i naturali: i naturali si estendono aggiungendo zeri in testa

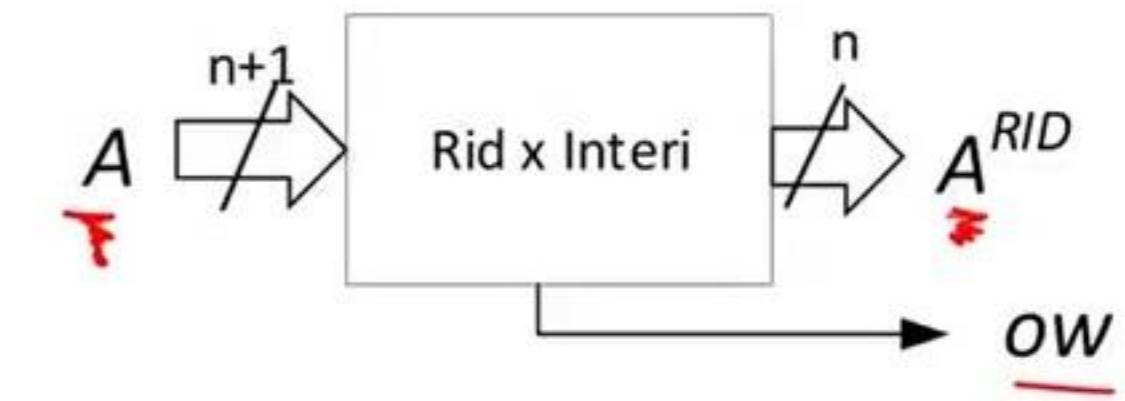
Riduzione di campo

- Dato $\underline{A} \equiv (\underline{a_n} a_{n-1} a_{n-2} \dots a_1 a_0)_{\beta}$, su $n + 1$ cifre, tale che $a \leftrightarrow A$
- Voglio trovare $A^{RID} \equiv (a_{n-1}' a_{n-2}' \dots a_1' a_0')_{\beta}$, su n cifre, tale che $a \leftrightarrow A^{RID}$
- Il problema ha soluzione **solo se**

$$\underline{a} \in [-\beta^n/2, \beta^n/2 - 1] \subset [-\beta^{n+1}/2, \beta^{n+1}/2 - 1]$$



Riduzione di campo (cont.)



$$a \in [-\beta^n/2, \beta^n/2 - 1] \subset [-\beta^{n+1}/2, \beta^{n+1}/2 - 1]$$

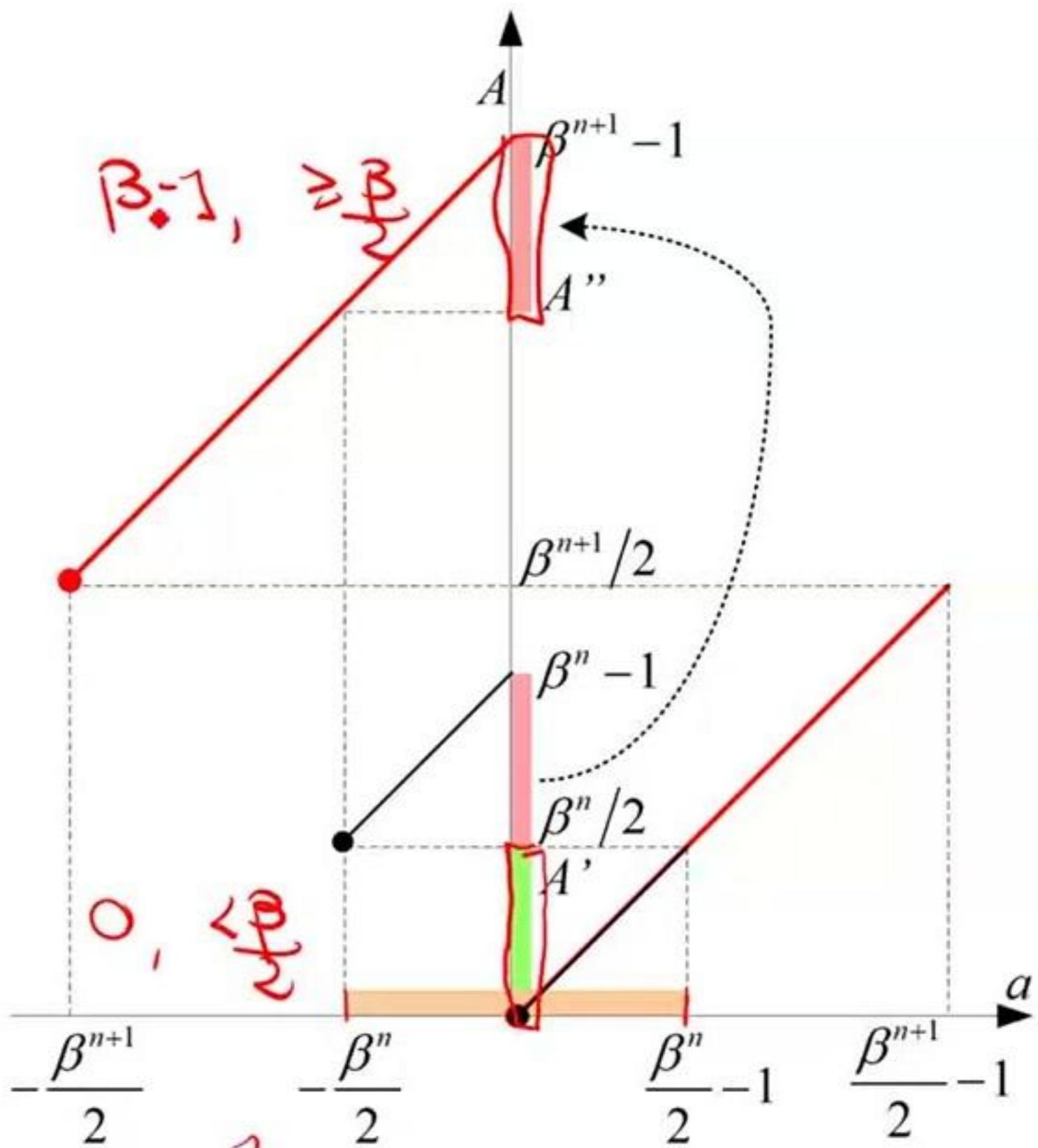
- Voglio calcolare A^{RID} a partire dalle cifre di A .

- $A \in [0; A']$, (rapp. di interi positivi):

- $MSD = 0$ AND cifra successiva $< \frac{\beta}{2}$

- $A \in [A''; \beta^{n+1} - 1]$, (rapp. di interi negativi):

- $MSD = \beta - 1$ AND cifra successiva $\geq \frac{\beta}{2}$

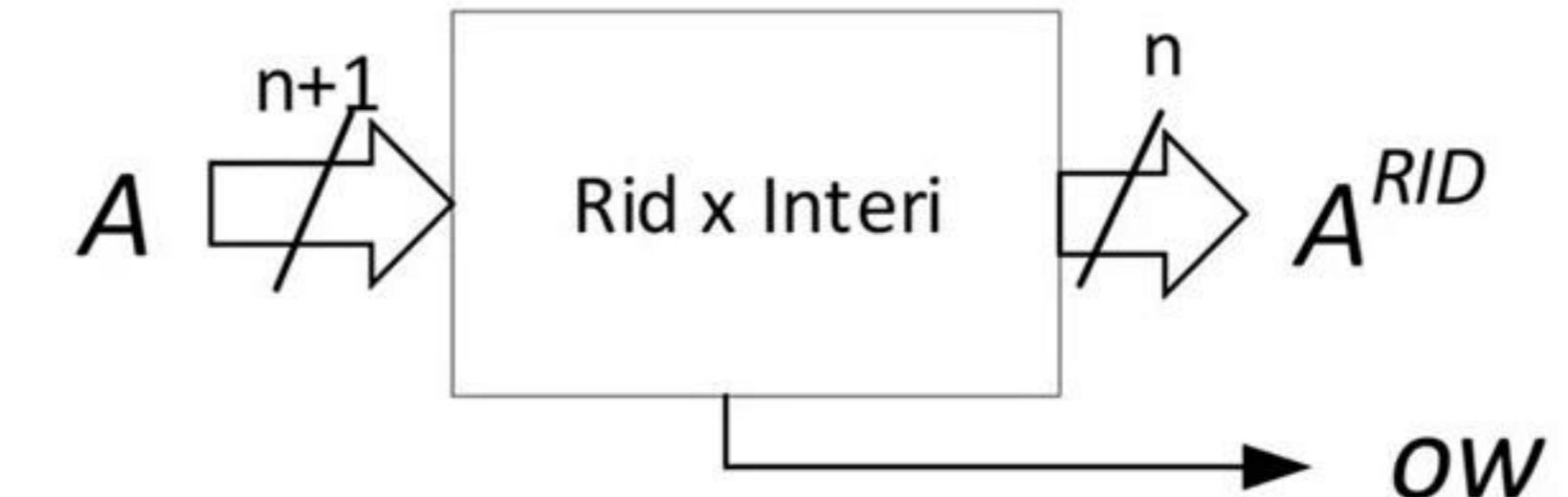


$$\underline{ow} = 0 \leftrightarrow \left(a_n = 0 \wedge a_{n-1} < \frac{\beta}{2} \right) \vee \left(a_n = \beta - 1 \wedge a_{n-1} \geq \frac{\beta}{2} \right)$$

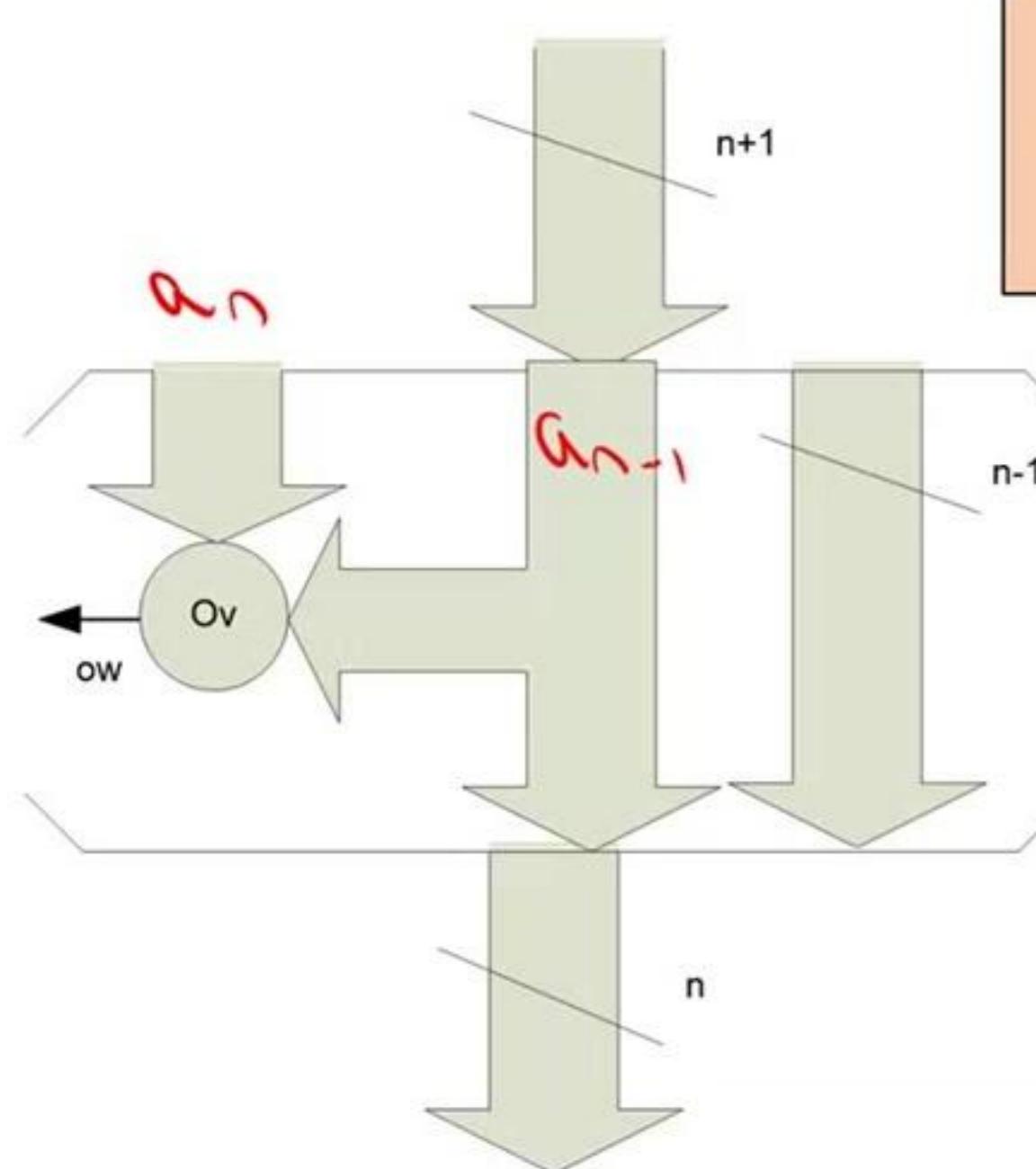
Riduzione di campo (cont.)

- Nel caso in cui il numero sia riducibile, la sua rappresentazione su n cifre è data **dalle n cifre meno significative di A**
 - ovvio se si pensa che $\underline{A} = \underline{(A^{EST})^{RID}}$, e che nell'estensione le n cifre meno significative rimangono identiche.
- Quindi: $\underline{\underline{A^{RID}}} = |A|_{\beta^n}$

Riduzione di campo (cont.)



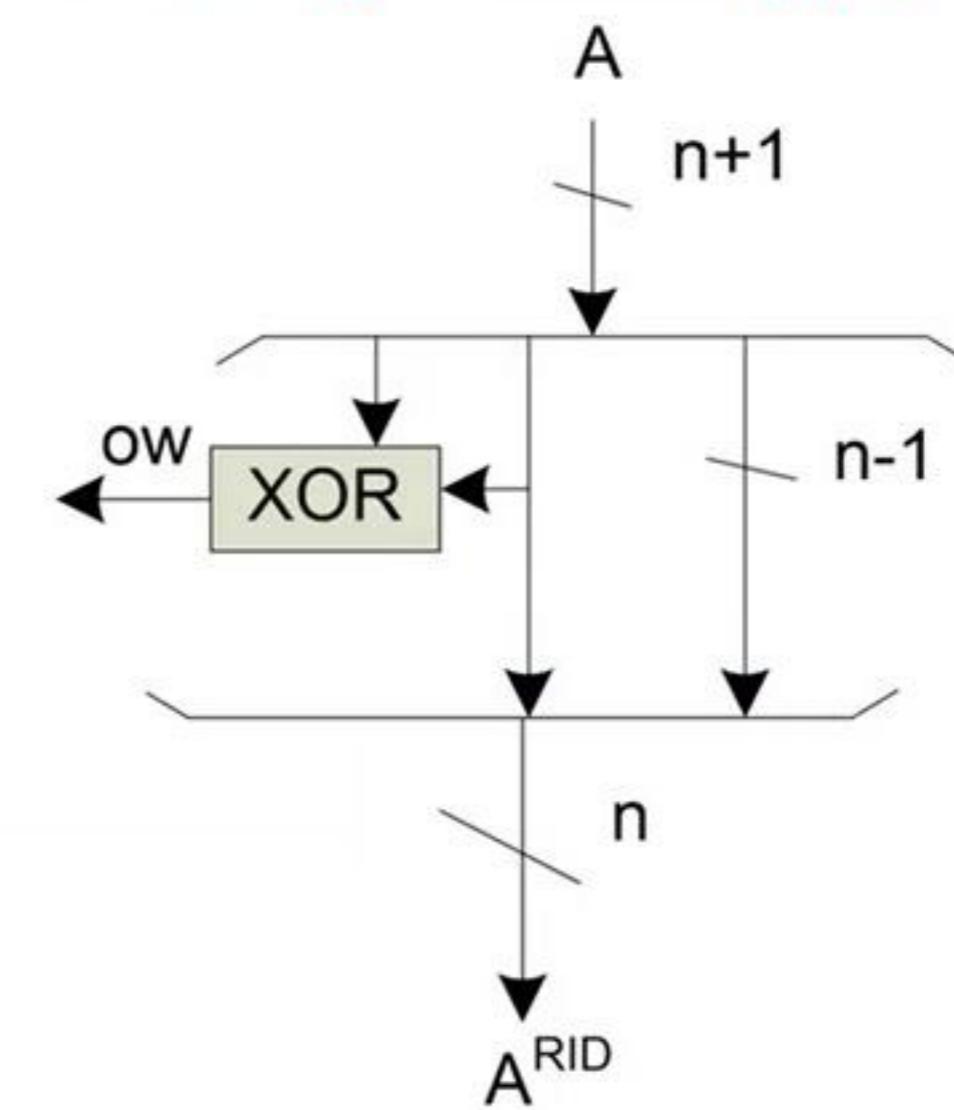
- Il circuito che riconosce la **non riducibilità** si chiama **circuito di overflow**



$$ow = 0 \leftrightarrow \left(a_n = 0 \wedge a_{n-1} < \frac{\beta}{2} \right) \vee \left(a_n = \beta - 1 \wedge a_{n-1} \geq \frac{\beta}{2} \right)$$

$a_{n-1} = \emptyset$ $a_n = 1$ $a_{n-1} = 1$

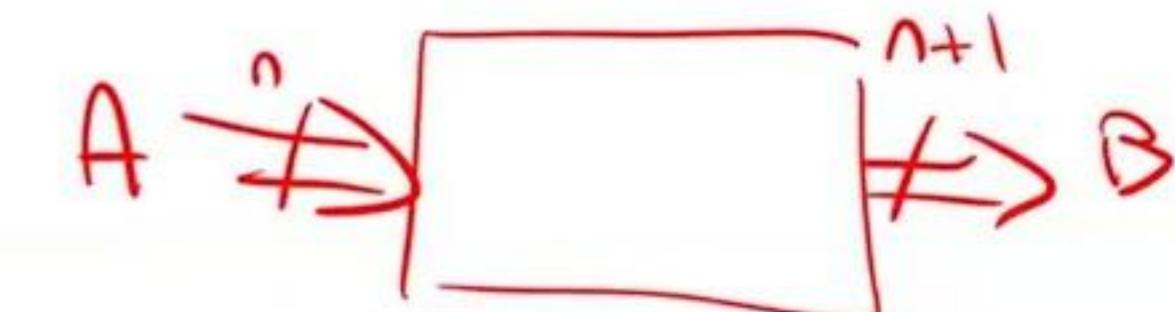
Base 2: le **due cifre più significative sono uguali**: XOR a 2 ingressi



~~B~~X

- - - Ø Ø - Ø

Moltiplicazione per potenza della base



- Dato $A \equiv (a_{n-1}a_{n-2}\dots a_0)_\beta$ su n cifre, con $a \leftrightarrow A$

- voglio trovare B su $n + 1$ cifre tale che $b \leftrightarrow B$ e

$$b \in \left[-\frac{\beta^{n+1}}{2}; \frac{\beta^{n+1}}{2} - \beta \right]$$

$$\rightarrow b = \beta \cdot a$$

$$a \in \left[-\frac{\beta}{2}; +\frac{\beta}{2} - 1 \right]$$

- In realtà dovremmo prima chiederci se b è rappresentabile su $n + 1$ cifre, ma la risposta è ovvia.

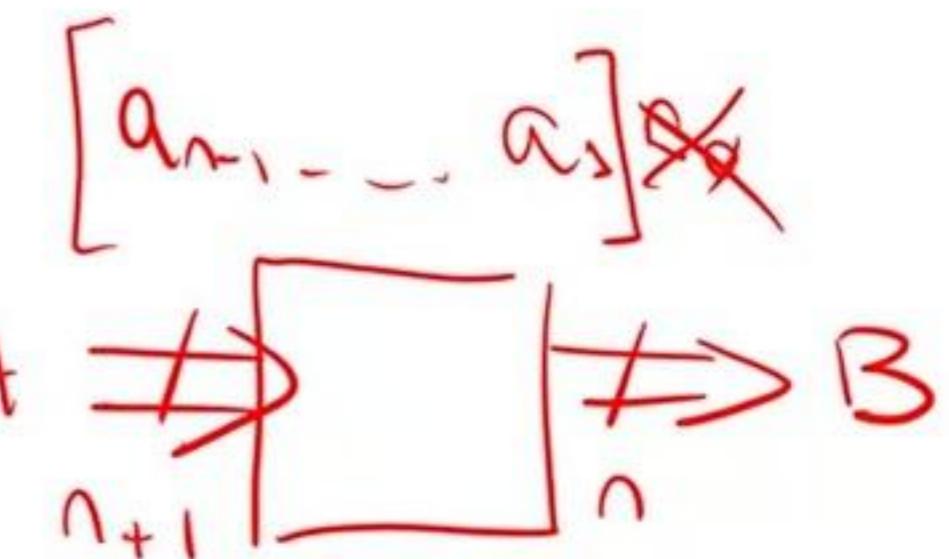
L: $B = \begin{cases} \beta \cdot a & A \\ \beta^{n+1} + b & A \end{cases}$

$$0 \leq a < \frac{\beta^n}{2}$$

$$-\frac{\beta^n}{2} \leq a < 0$$

- La soluzione è $B = \beta \cdot A$

Divisione per potenza della base



- Analogamente, dato $A \equiv (a_n a_{n-1} a_{n-2} \dots a_0)_\beta$ su $n+1$ cifre, tale che $a \leftrightarrow A$
- Voglio trovare B su n cifre tale che $b \leftrightarrow B$ e

$$B = |b|_{\beta^n}$$

$$B = \left\| \frac{a}{\beta} \right\|_{\beta^n} = \left\| \frac{\lfloor a/\beta^{n+1} \rfloor \cdot \beta^{n+1} + |a|_{\beta^{n+1}}}{\beta} \right\|_{\beta^n} = \left\| \lfloor a/\beta^{n+1} \rfloor \cdot \beta^n + \frac{|a|_{\beta^{n+1}}}{\beta} \right\|_{\beta^n}$$

$$\left\| \left\lfloor \frac{A}{\beta} \right\rfloor \right\|_{\beta^n}$$

$$= \left\| \frac{A}{\beta} \right\|_{\beta^n} = \left\lfloor \frac{A}{\beta} \right\rfloor$$

- La soluzione è $B = \lfloor A/\beta \rfloor$

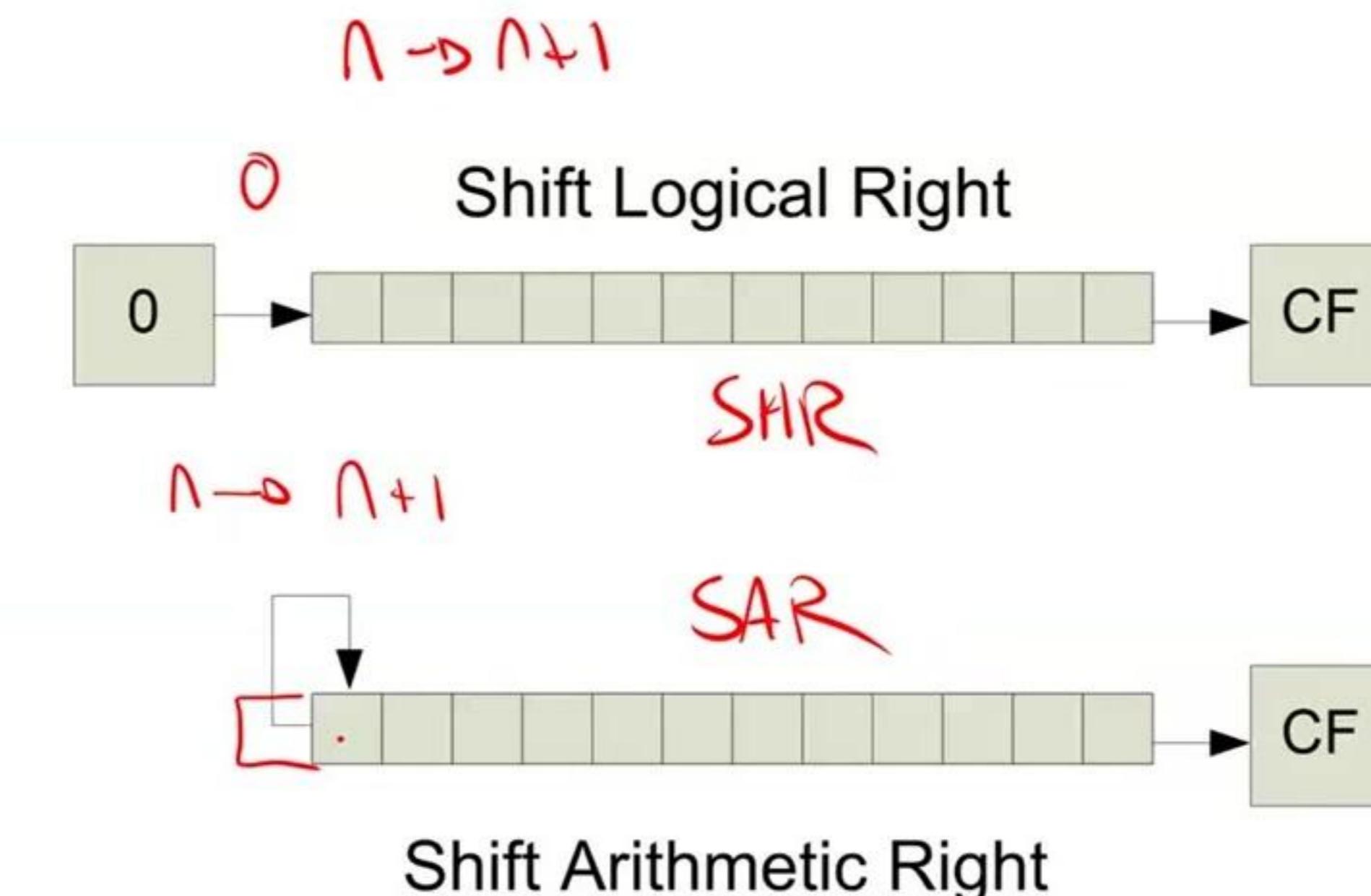
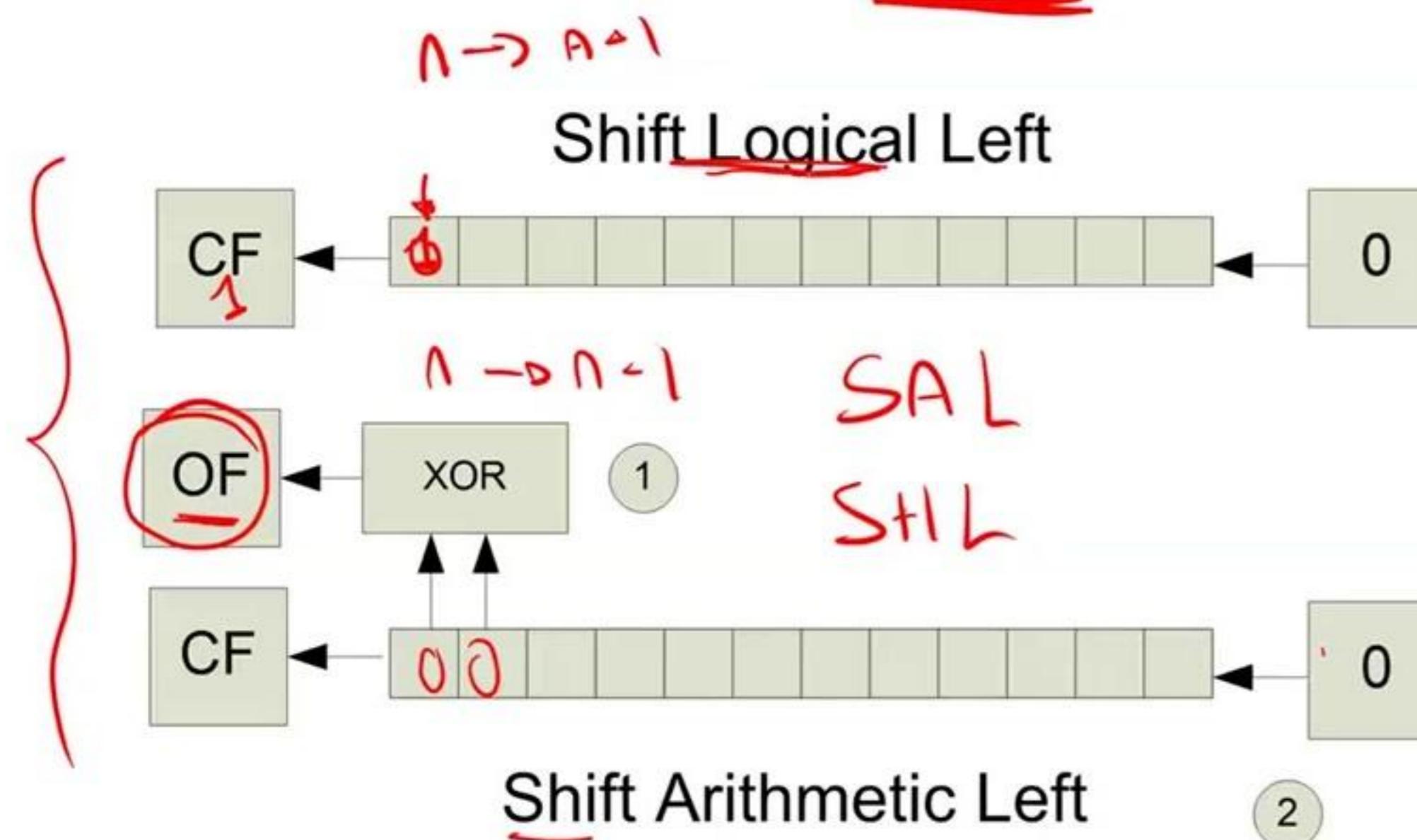
Shift logico e aritmetico

- In Assembler esistono **due tipi di istruzione di shift** (per ciascuna direzione)
 - shift **logical** left (right) – corrisponde alla moltiplicazione (divisione) per 2^k di un **naturale**
 - shift **arithmetic** left (right) – corrisponde alla moltiplicazione (divisione) per 2^k di un **intero**
- Per quale motivo, visto che abbiamo appena detto che MUL/DIV per una potenza della base si fanno nello stesso modo sia per i **naturali** che per gli **intери**?

Shift logico e aritmetico

$$\begin{array}{l} n \rightarrow n+1 \\ n+1 \rightarrow n \end{array}$$

- Si fanno nello stesso modo se posso aumentare/diminuire le cifre
- Ma in Assembler lavoriamo su operandi che hanno dimensione (i.e., numero di cifre) fissato



Addizione

- Dati A, B in base β su n cifre, tali che $a \leftrightarrow A$ e $b \leftrightarrow B$
- Voglio calcolare S su n cifre tale che $s \leftrightarrow S$ ed

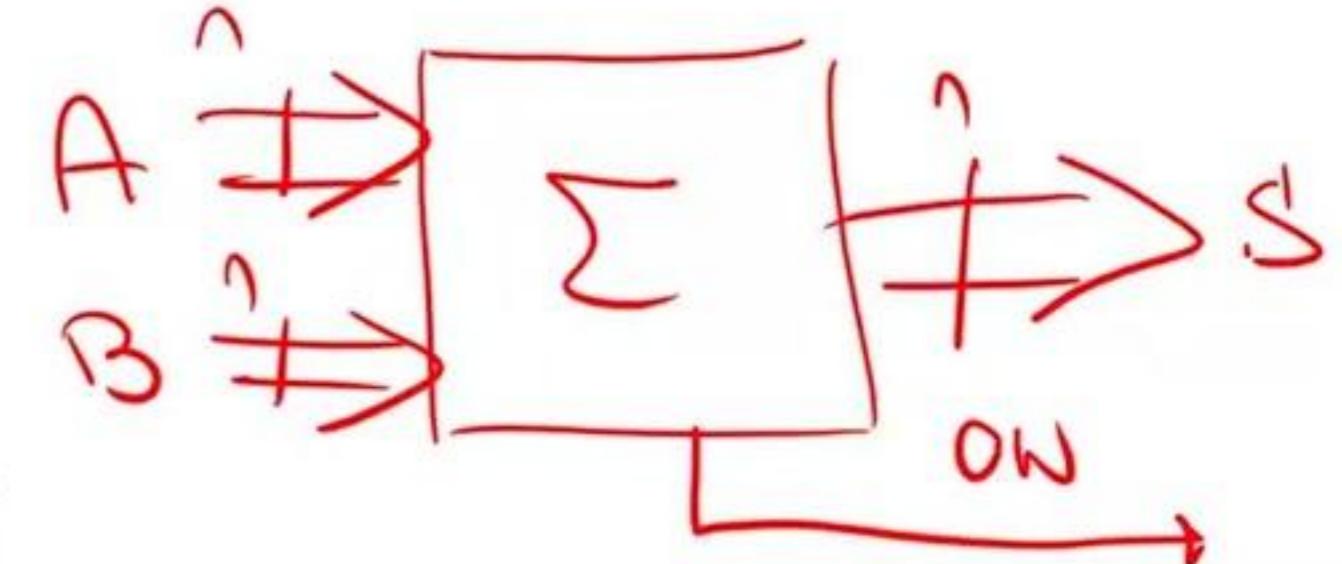
$$\left[-\frac{\beta^n}{2}; +\frac{\beta^n}{2} - 1 \right]$$

$$s = a + b$$

- Su quante cifre è rappresentabile il risultato?

- $-\beta^n \leq s \leq \beta^n - 2$
- quindi la somma può **non essere rappresentabile su n cifre**
- Pertanto se il circuito sommatore ha un'uscita su **n cifre**, dovrà essere dotato di un'uscita ulteriore di **overflow**

- { • Però è sicuramente su **$n + 1$ cifre**: $[-\beta^n; \beta^n - 2] \subset [-\frac{\beta^{n+1}}{2}; +\frac{\beta^{n+1}}{2} - 1]$



Addizione (cont.)

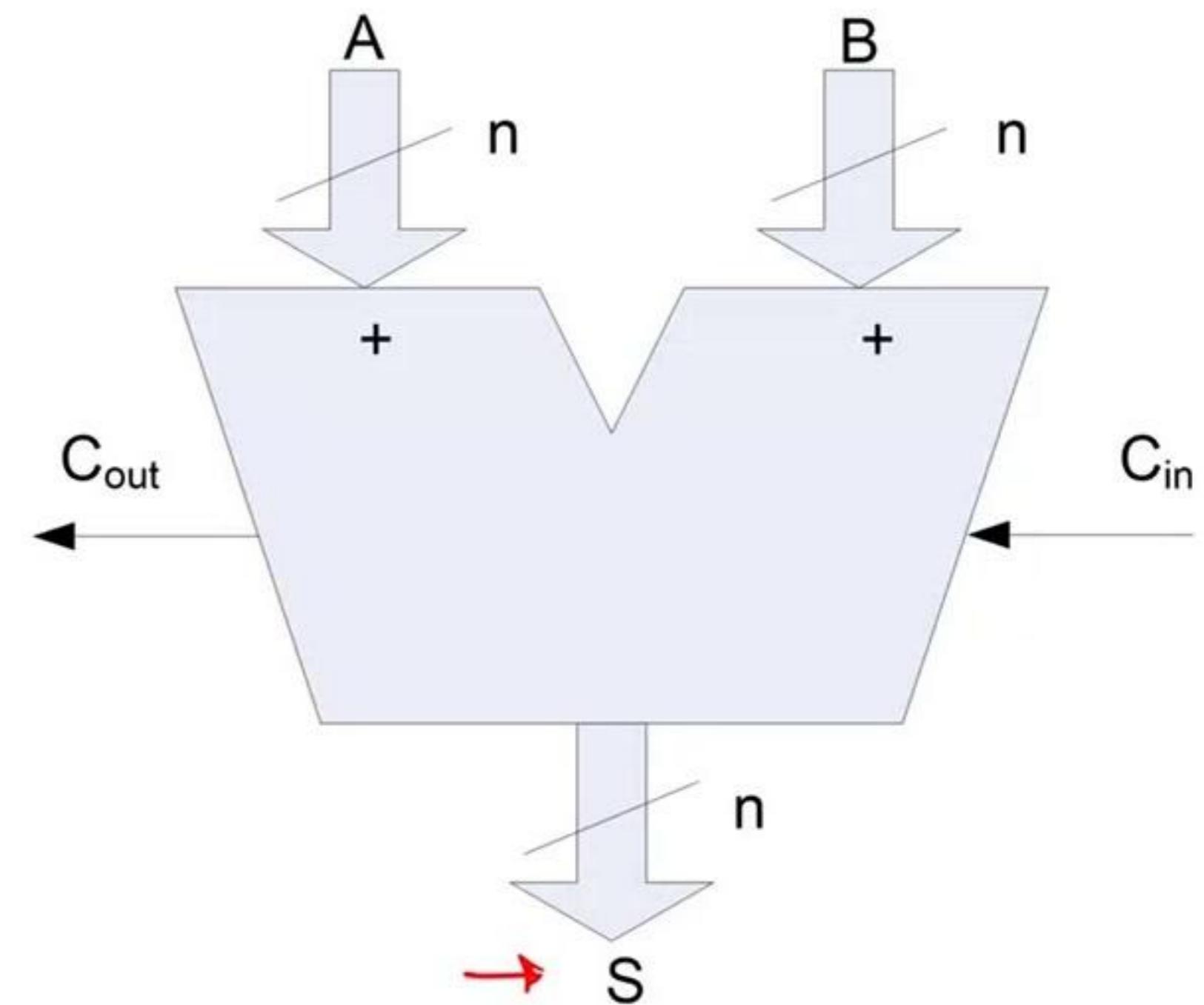
- Quando s è rappresentabile su n cifre, è vero che

$$S \stackrel{(*)}{=} |s|_{\beta^n} = |a + b|_{\beta^n} = \left| |a|_{\beta^n} + |b|_{\beta^n} \right|_{\beta^n} = |A + B|_{\beta^n}$$

- La rappresentazione della somma è uguale alla somma delle rappresentazioni modulo β^n .
- Questa proprietà da sola motiva l'utilizzo della **rappresentazione in complemento alla radice** dei numeri interi

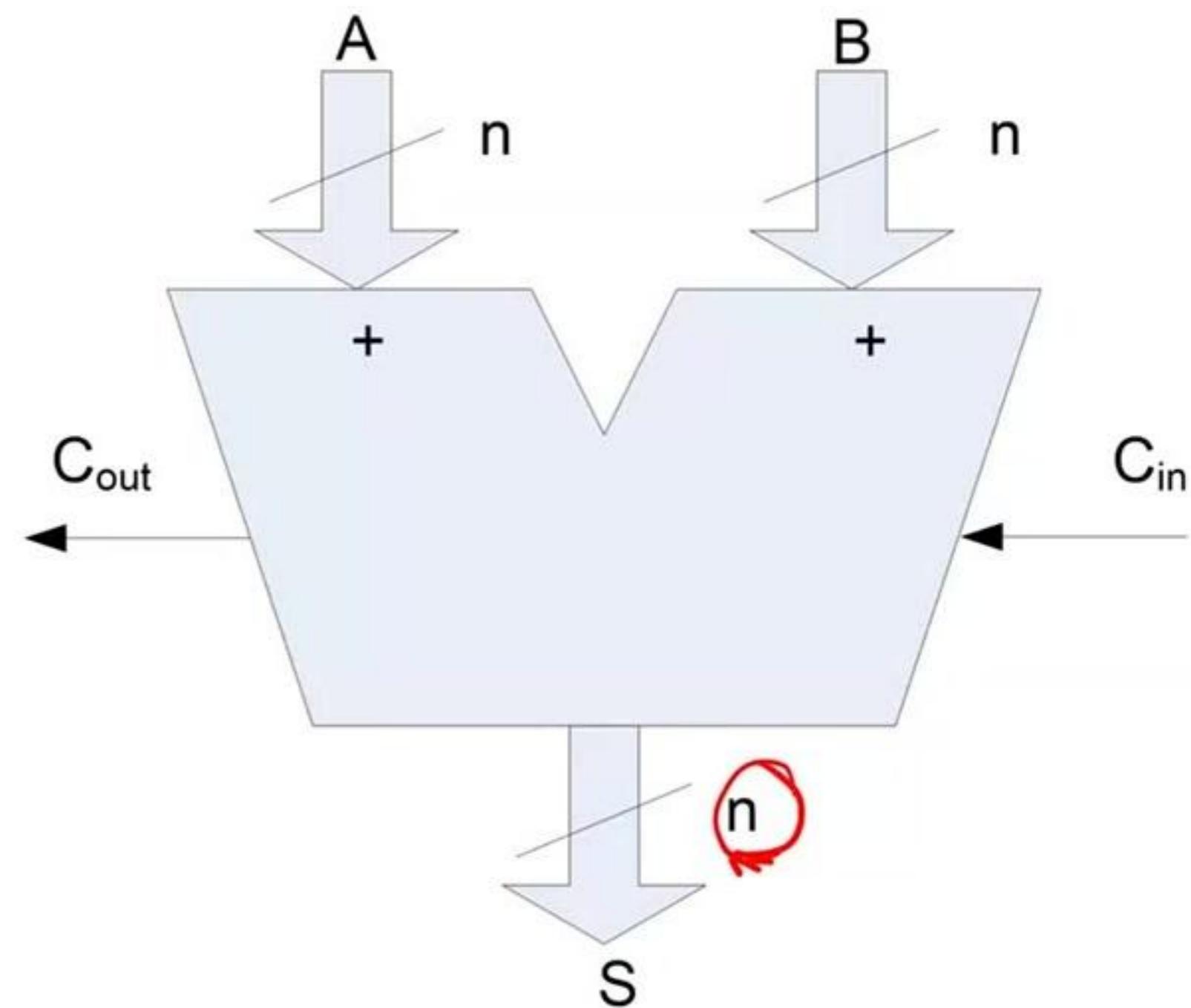
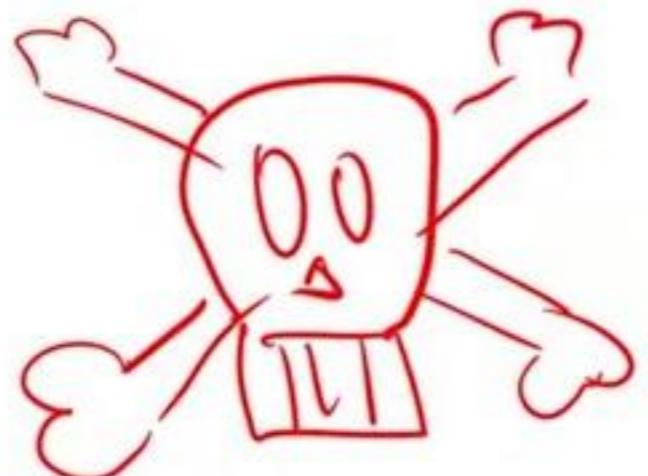
Addizione (cont.)

- Posso usare il circuito **sommatore** anche per sommare (rappresentazioni di) interi
- Il risultato, quando l'operazione è fattibile, sarà comunque corretto
- Devo capire come sintetizzare l'uscita di overflow.



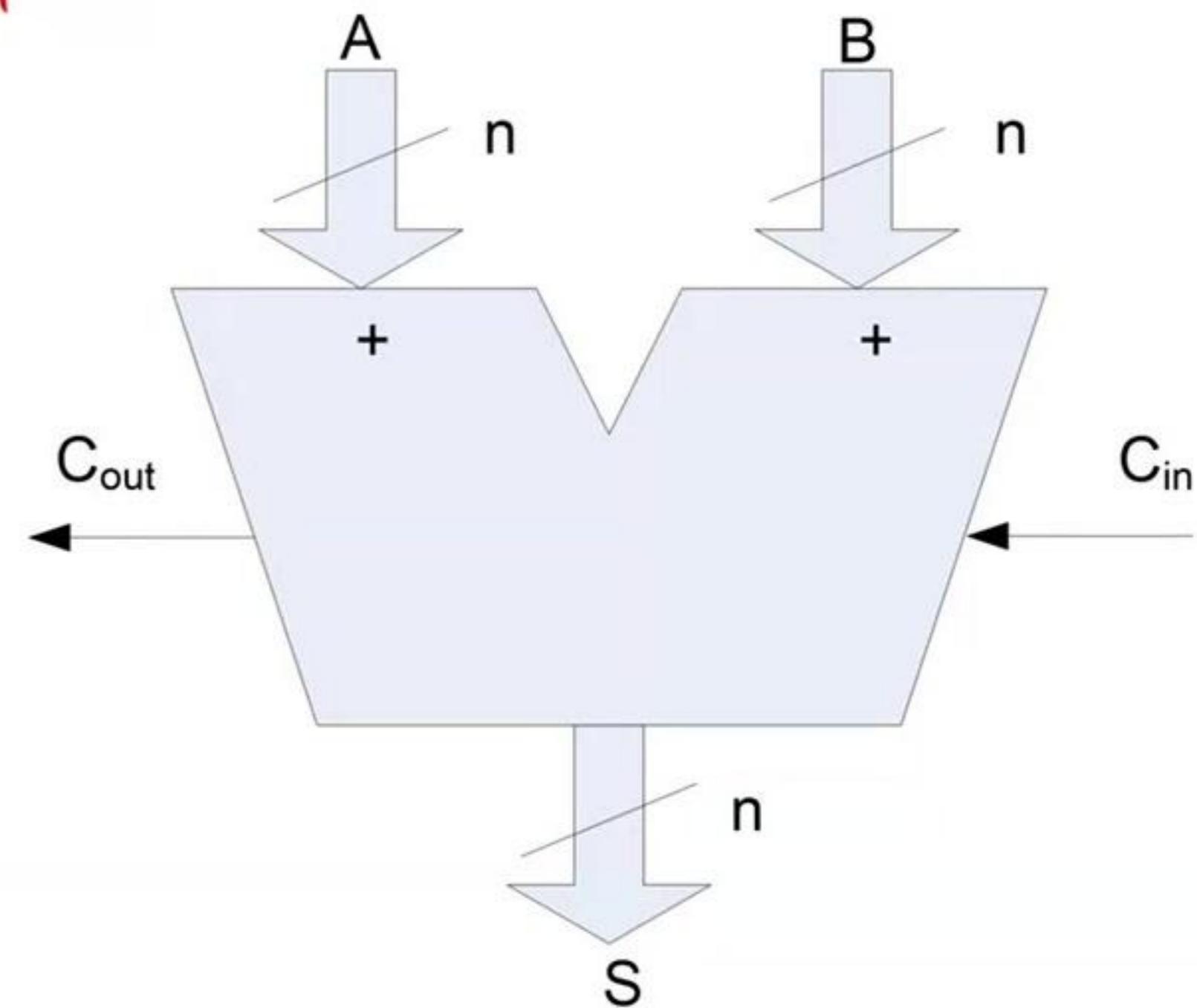
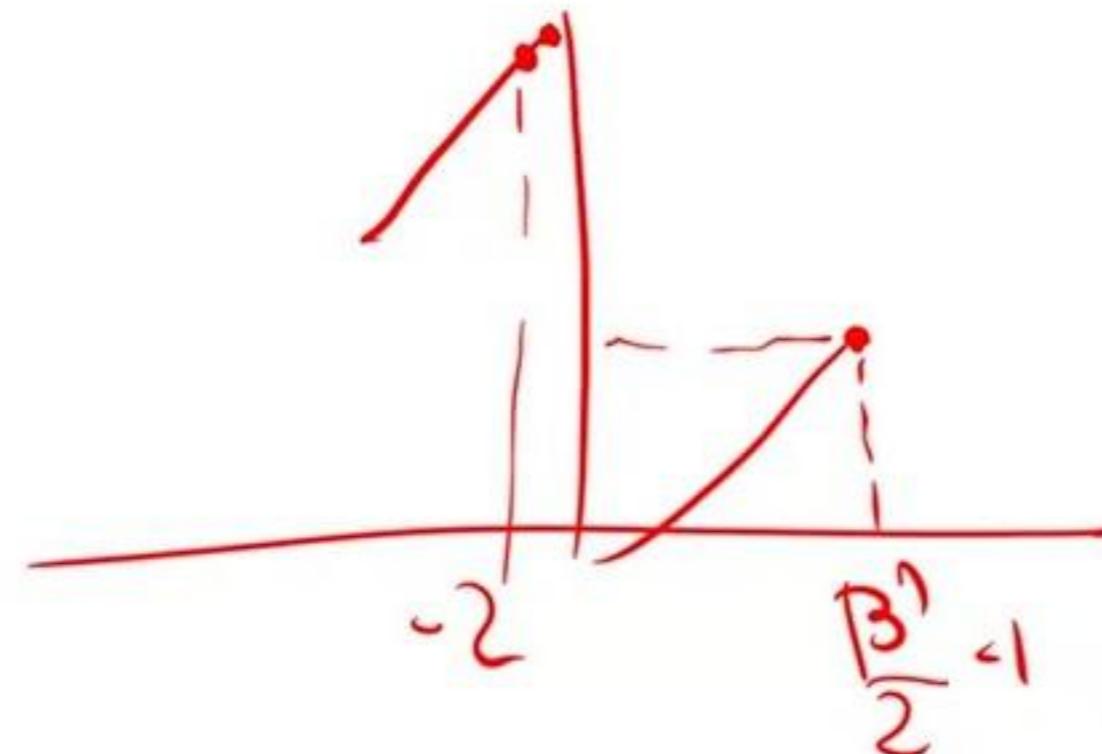
Addizione (cont.)

- Ricordiamo che, presi due naturali A e B ,
il **circuito sommatore** produce in uscita
- $|A + B|_{\beta^n}$
- C_{out} , che mi dice se la somma tra **naturali**
è rappresentabile, in quanto minore di β^n
- Posso usare C_{out} come overflow?
 - No.
 - No.
 - No.



Addizione (cont.)

- $A = \beta^n - 1, B = 1$
 - $S = 0, \underline{\underline{C_{out} = 1}}$
 - Ma se $A = \beta^n - 1$, allora $a \leftrightarrow -(\bar{A} + 1) = \underline{-1}$.
 - Quindi $s = a + b = 0$, numero intero rappresentabile
 - ow = 0
- $A = B = \beta^n/2 - 1$
 - $S = \beta^n - 2, \underline{\underline{C_{out} = 0}}$
 - Ma se $S = \beta^n - 2$, allora $s \leftrightarrow -(\bar{S} + 1) = -2$,
 - che non può essere la somma di due numeri **positivi**
 - ow = 1



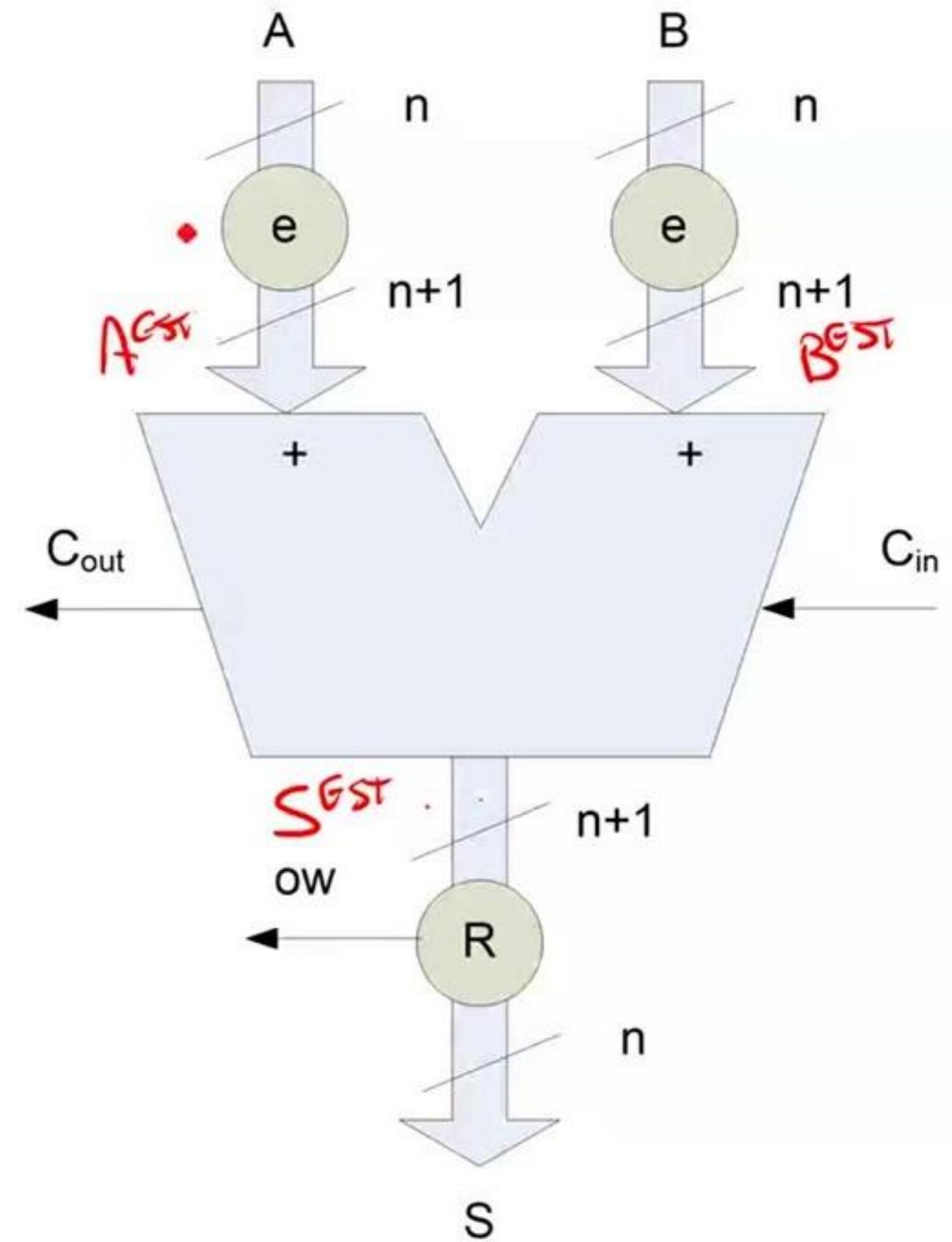
Addizione (cont.)

$\beta \rightarrow 2$

- La somma è sempre rappresentabile su $n + 1$ cifre.
- Sommo su $n + 1$ cifre, estendendo gli addendi.
Calcolo cioè:

$$\underline{\underline{S^{EST}}} = |s|_{\beta^{n+1}} = |a + b|_{\beta^{n+1}}$$

$$= \left| |a|_{\beta^{n+1}} + |b|_{\beta^{n+1}} \right|_{\beta^{n+1}} = \underline{\underline{A^{EST}}} + \underline{\underline{B^{EST}}} |_{\beta^{n+1}}$$



- Controllo la riducibilità della somma con un apposito detettore di riducibilità

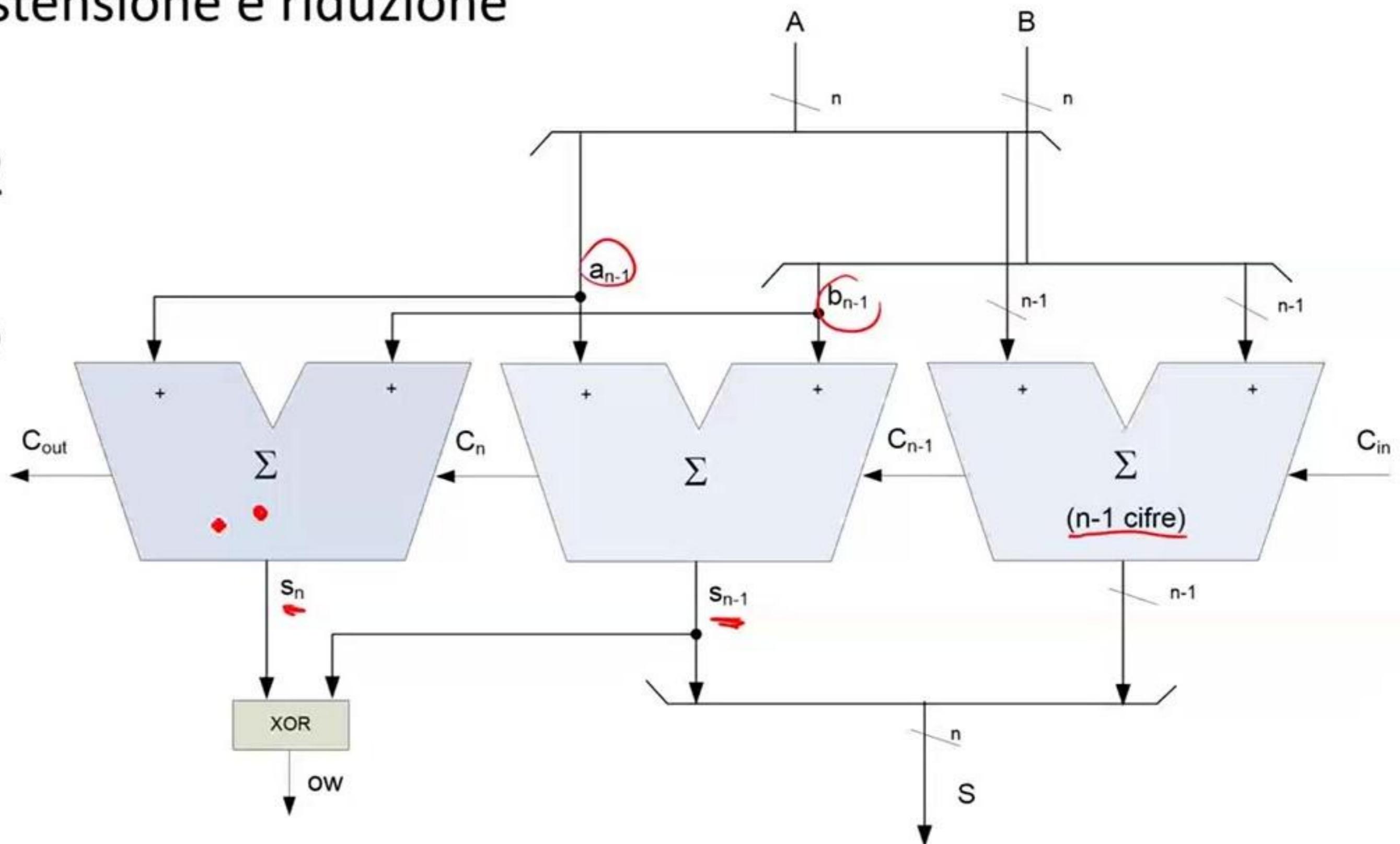
Addizione (cont.)

- In base β generica non posso che usare **uno stadio di sommatore in più** rispetto al # di cifre degli addendi

- Più un po' di altra logica, per estensione e riduzione

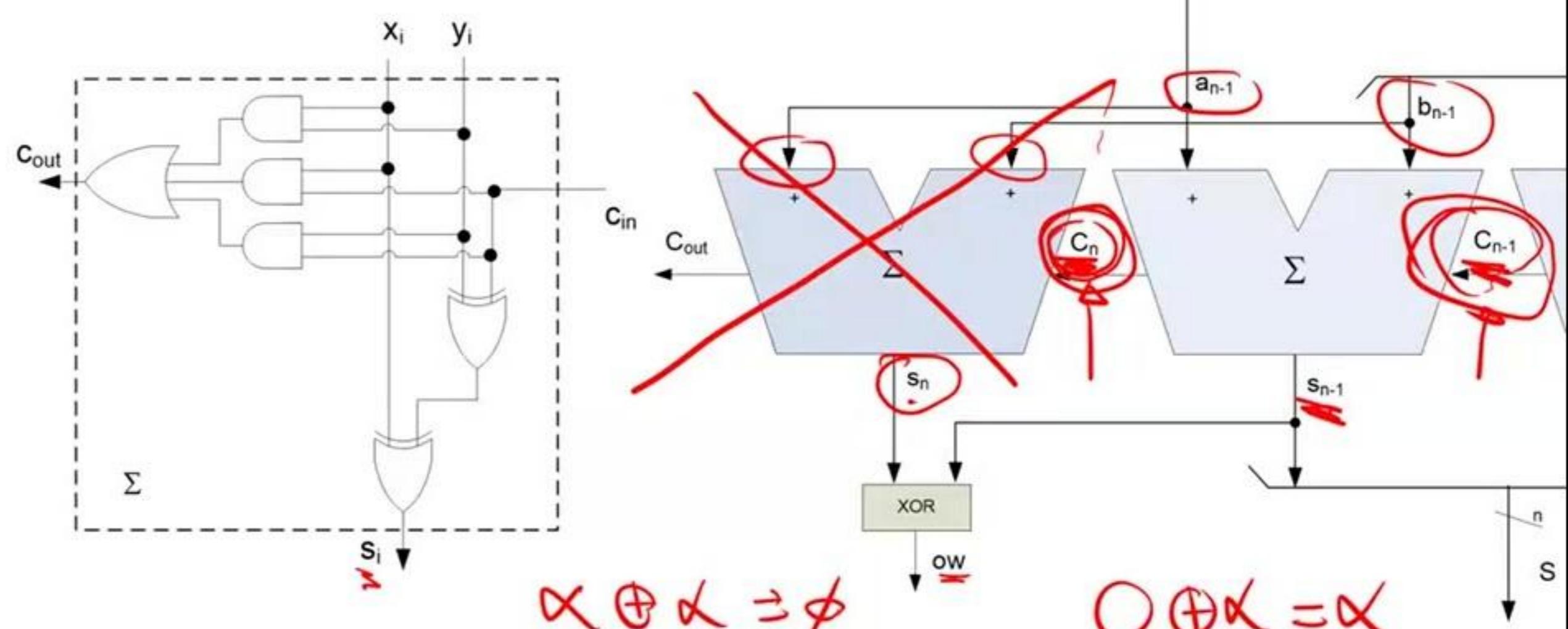
- Vediamo cosa succede in base 2

- L'estensione è a costo nullo
- La riducibilità si controlla con uno ~~XOR~~



Addizione (cont.)

- Ricordando
 - Come è fatto un full adder in base 2
 - Le proprietà algebriche dello XOR



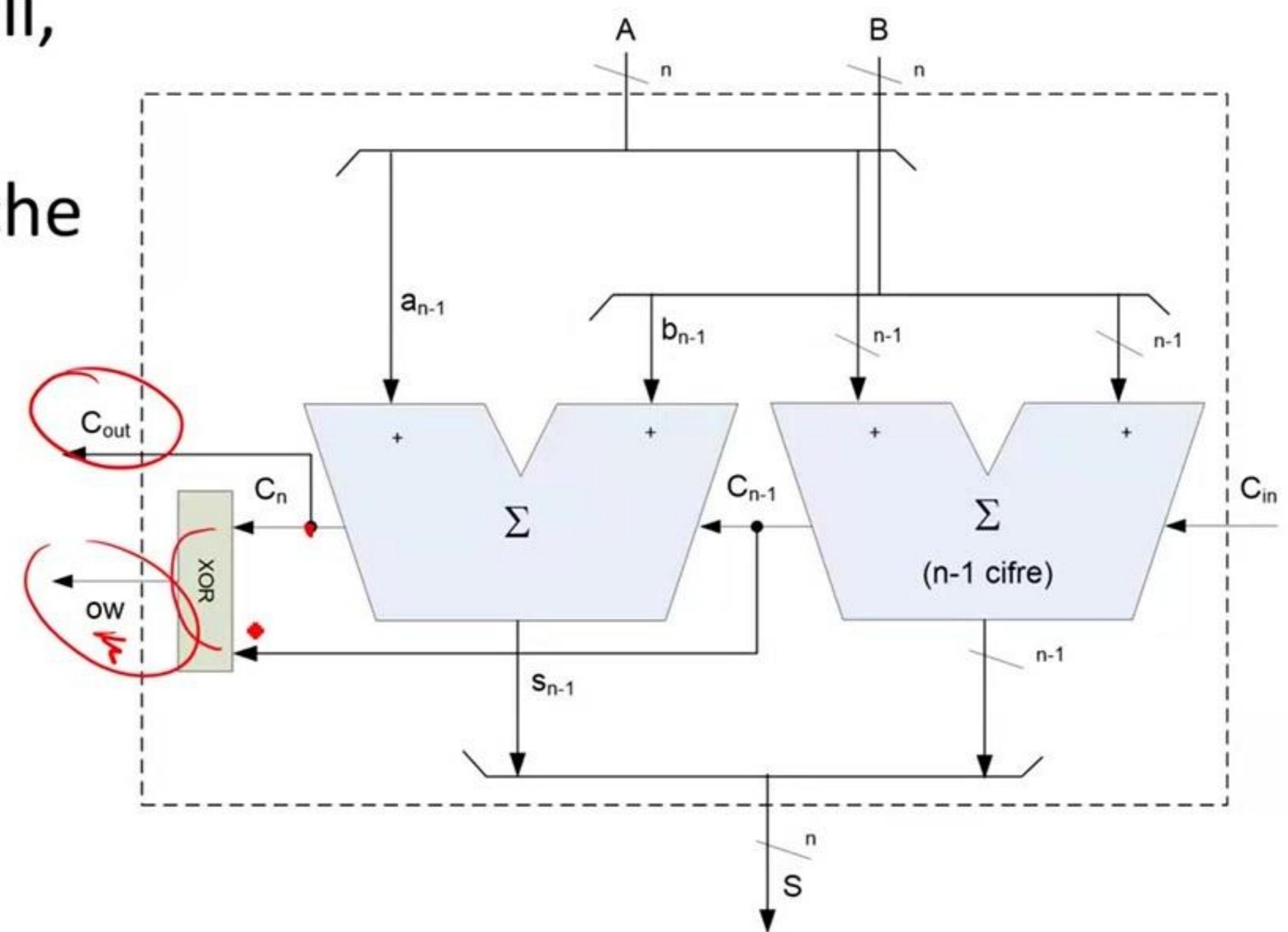
$$\begin{aligned}
 \cancel{ow} &= s_n \oplus s_{n-1} = (\cancel{a_{n-1}} \oplus \cancel{b_{n-1}} \oplus c_n) \oplus (\cancel{a_{n-1}} \oplus \cancel{b_{n-1}} \oplus c_{n-1}) \\
 &= 0 \oplus c_n \oplus c_{n-1} = \cancel{c_n \oplus c_{n-1}}
 \end{aligned}$$

β

- Quindi in base 2 l'overflow può essere calcolato dai **riporti uscenti** degli ultimi **due full adder**, e non c'è bisogno di aggiungere un altro stadio di sommatore
 - Basta aggiungere la porta XOR

Addizione (cont.)

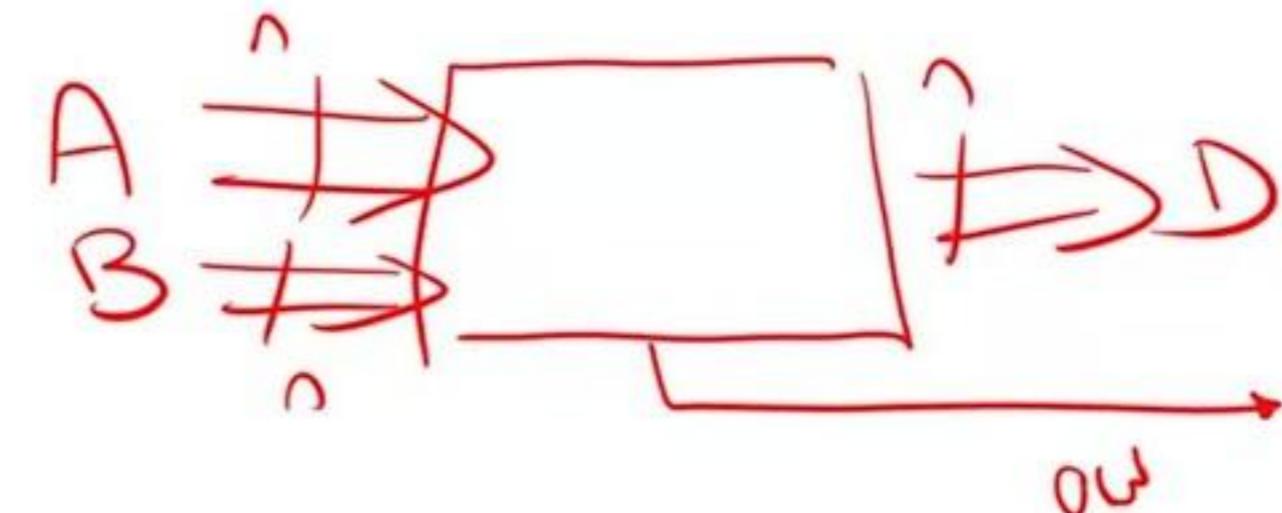
- La stessa circuiteria che somma naturali, **somma anche (rapp. di) interi.**
- Basta considerare l'uscita di overflow, che può essere prodotta quasi gratis



Addizione (cont.)

- **Richiamo sull'Assembler:** esiste una sola istruzione **ADD**, che somma numeri **naturali**. Il suo risultato è corretto anche se quei numeri naturali sono la rappresentazione di numeri interi
- La rappresentabilità del risultato si controlla guardando se:
 - CF=0, se gli addendi sono naturali
 - OF=0, se gli addendi sono (rappresentazioni di) interi
- La ADD setta sia **CF** che **OF**. Sarà poi il programmatore ad usare JC/JNC oppure JO/JNO a seconda di cosa sta facendo

Sottrazione



- Si fa in modo simile all'addizione. La differenza è sempre rappresentabile su $n + 1$ cifre, non sempre su n .
- Dati A e B in base β su n cifre, $a \leftrightarrow \underline{A}$ e $b \leftrightarrow \underline{B}$,
- Voglio calcolare D su n cifre, con $d \leftrightarrow \underline{D}$ e $\underline{d} = a - b$. $\overline{B} = \beta^n - 1 - B$
- Quando \underline{d} è rappresentabile su n cifre, abbiamo

$$D \stackrel{(*)}{=} |d|_{\beta^n} = |a - b|_{\beta^n} = \left| |a|_{\beta^n} - |b|_{\beta^n} \right|_{\beta^n} = \boxed{|A - B|_{\beta^n}} = |A + \overline{B} + 1|_{\beta^n}$$

- uscita da un sottrattore per numeri naturali, che abbia in ingresso le rappresentazioni A e B .

Sottrazione (cont.)

$$ow = b_{out} \oplus b_{n+1}$$

- Posso usare un sottrattore **sia** per sottrarre numeri naturali, sia per sottrarre rappresentazioni di numeri interi.
- Per testare la fattibilità, osservo che

$$\begin{aligned} D^{EST} &= |d|_{\beta^{n+1}} = |a - b|_{\beta^{n+1}} = \left| |a|_{\beta^{n+1}} - |b|_{\beta^{n+1}} \right|_{\beta^{n+1}} \\ &= |A^{EST} - B^{EST}|_{\beta^{n+1}} = \left| A^{EST} + \overline{B^{EST}} + 1 \right|_{\beta^{n+1}} \end{aligned}$$

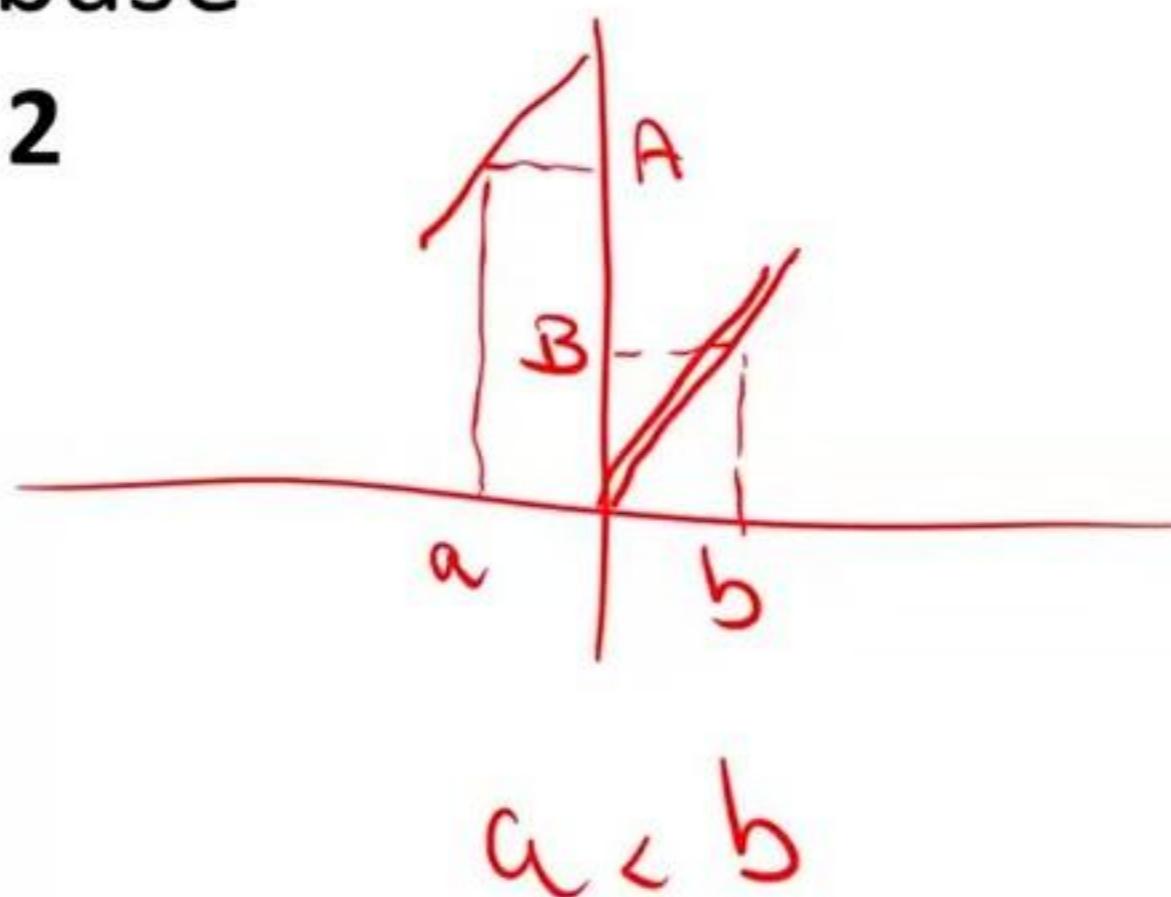
- In base β generica, posso calcolare *ow* testando la **riducibilità** del risultato della differenza su $n + 1$ cifre
- Per la base due *ow* è lo XOR degli ultimi due prestiti

~~ow~~

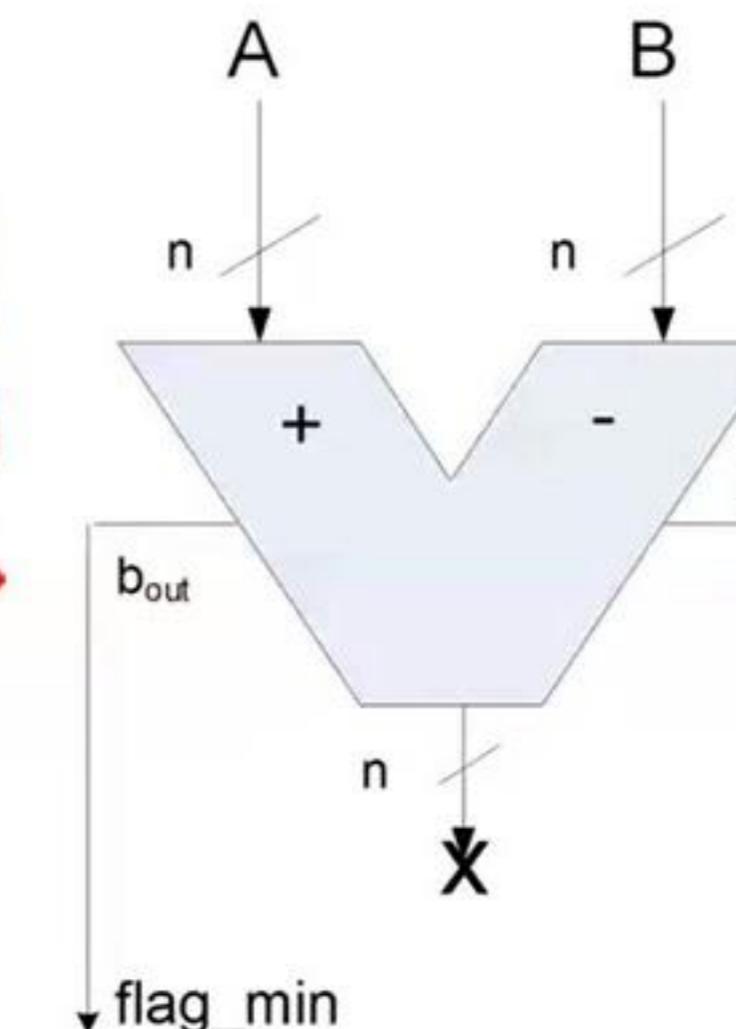
Comparazione di numeri interi

A = B

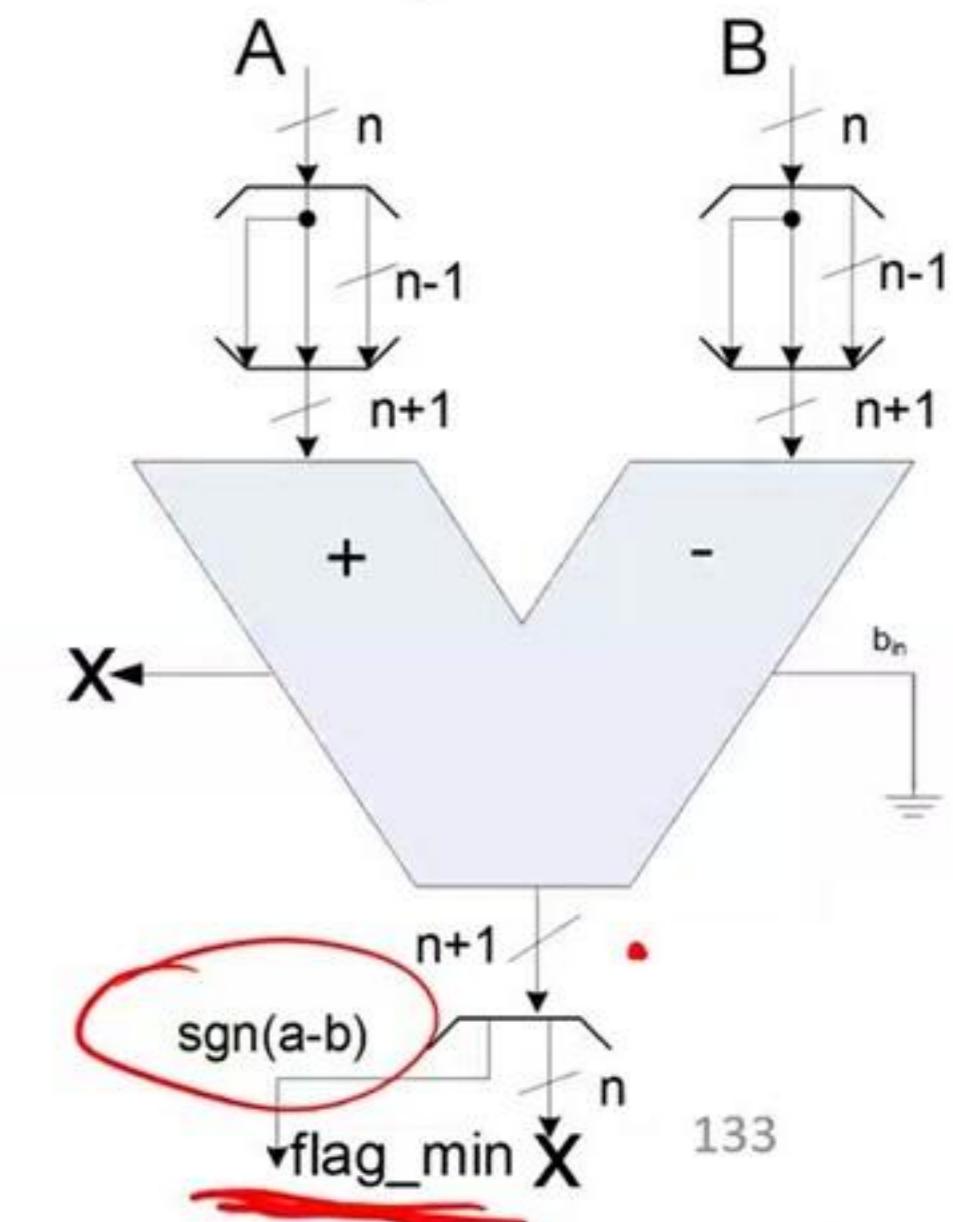
- **Uguaglianza ($a = b$)**: si fa nello stesso modo che con i naturali
- **Minoranza ($a < b$)**: non ha senso guardare il prestito uscente della differenza $A - B$
 - Devo guardare il segno del risultato della sottrazione
 - Devo quindi poter **svolgere la sottrazione**
 - Devo estendere gli operandi su $n + 1$ cifre
- Vale in qualunque base
 - compresa la base 2



Comparatore
per naturali



Comparatore
per interi



~~TOPP~~ +
=
~~+ NPP~~

Moltiplicazione e divisione di interi

$CR \rightarrow MS$; $\rightarrow CR$

- Conviene riferirsi ai valori assoluti degli operandi, ed **aggiustare i segni** successivamente.
- Ciò consente di riutilizzare la circuiteria per le moltiplicazioni/divisioni tra numeri naturali.
- Dovremo quindi far uso di reti che trasformano:
 - da CR a modulo e segno (già vista)
 - da modulo e segno a CR

Circuito di conversione da MS a CR

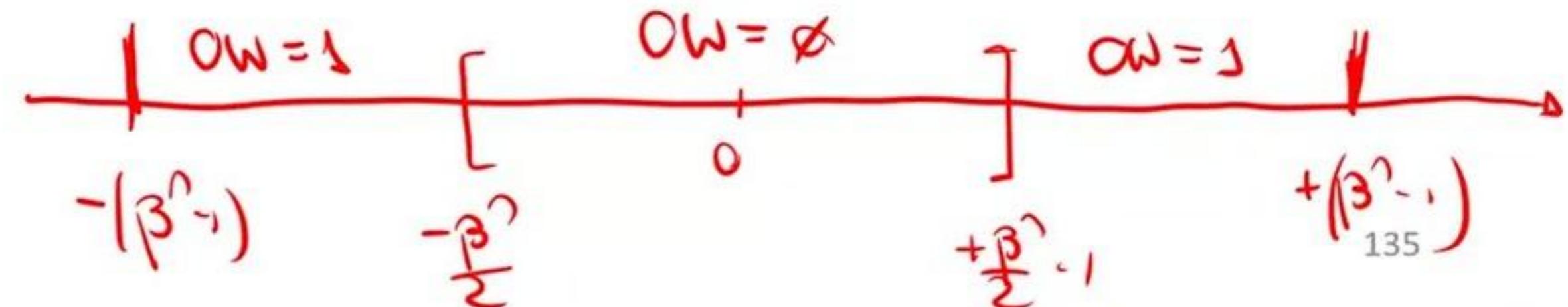
- Prende in ingresso il valore assoluto (su n cifre) ed il segno della rappresentazione di un numero intero, e produce in uscita la sua rappresentazione in complemento alla radice su n cifre.



- L'operazione **non è sempre possibile**. Infatti, abbiamo:

- in ingresso: $-(\beta^n - 1) \leq a \leq +(\beta^n - 1)$

- in uscita: $-\frac{\beta^n}{2} \leq a \leq \frac{\beta^n}{2} - 1$



Circuito di conversione da MS a CR (cont.)

- Se l'operazione è fattibile, è

$$\underline{\overline{A}} = \underline{|a|}_{\beta^n} = \begin{cases} |ABS_a|_{\beta^n} & \text{sgn}_a = 0 \\ | -ABS_a |_{\beta^n} & \text{sgn}_a = 1 \end{cases} = \begin{cases} ABS_a & \text{sgn}_a = 0 \\ |\overline{ABS_a} + 1|_{\beta^n} & \text{sgn}_a = 1 \end{cases}$$

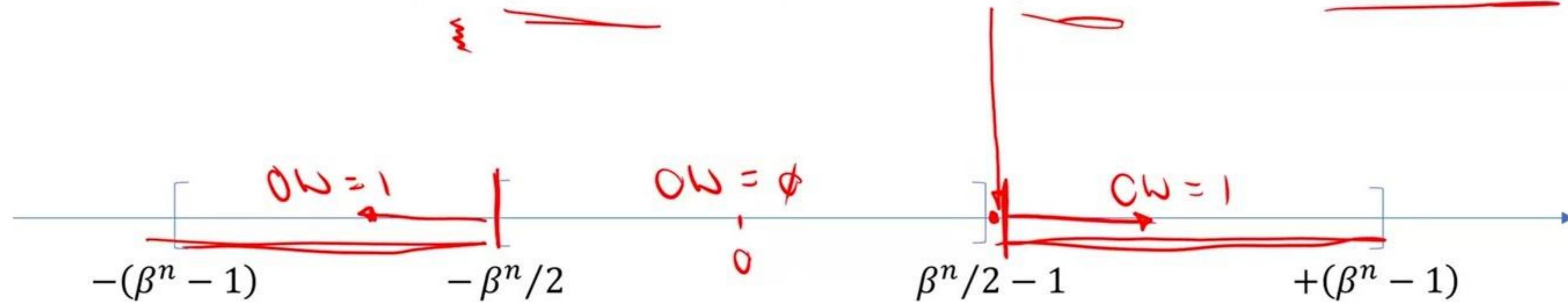
~~$\star \bar{x} = \beta^n - 1 - x$~~ $-x = \bar{x} - \beta^n + 1$

- Si fa con un multiplexer ed un circuito per il calcolo dell'opposto

Circuito di conversione da MS a CR (cont.)

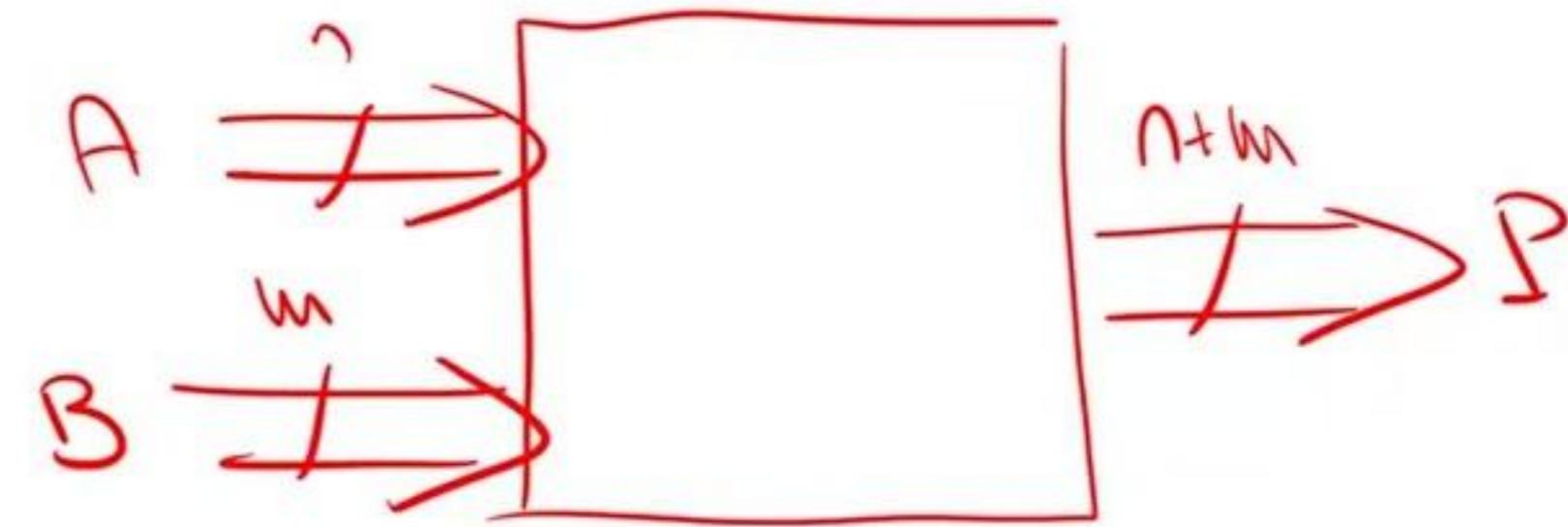
- Per quanto riguarda l'overflow, abbiamo:

$$ow = 1 \Leftrightarrow (\text{ABS}_a > \beta^n/2) \text{ or } (\text{ABS}_a = \beta^n/2 \text{ and } \text{sgn}_a = 0)$$



- Quest'ultima operazione si sintetizza con un comparatore e poco più
 - farla per esercizio

Moltiplicazione



- Dati A su n cifre e B su m cifre, $a \leftrightarrow A$ e $b \leftrightarrow B$,
- Voglio calcolare P su $n + m$ cifre tale che $p \leftrightarrow P$ e $p = a \cdot b$.
- $-\beta^{n+m}/4 \leq p \leq \beta^{n+m}/4$
 - non ci sono problemi di rappresentabilità per il risultato.

$$a \in \left[-\frac{\beta^n}{2}; +\frac{\beta^n}{2} - 1 \right]$$

$$b \in \left[-\frac{\beta^m}{2}; +\frac{\beta^m}{2} - 1 \right]$$

$$P \in \left[-\frac{\beta^{n+m}}{4}; +\frac{\beta^{n+m}}{4} \right]$$



$$P \in \left[-\frac{\beta^{n+m}}{2}; +\frac{\beta^{n+m}}{2} - 1 \right]$$

Moltiplicazione (cont.)

- Osserviamo che:

$$p = \begin{cases} ABS(a) \cdot ABS(b) & a, b \text{ concordi} \\ -ABS(a) \cdot ABS(b) & a, b \text{ discordi} \end{cases}$$

- Quindi:

$$\left\{ \begin{array}{l} \textcolor{red}{\overbrace{ABS(p) = ABS(a) \cdot ABS(b)}} \\ \textcolor{red}{\overbrace{\operatorname{sgn}(p) = \operatorname{sgn}(a) \cdot \operatorname{sgn}(b)}} \end{array} \right. \quad \text{MUL} \times \text{NAT}$$

- Posso calcolare $ABS(a) \cdot ABS(b)$, risultato di una moltiplicazione tra **naturali**,
- Rappresento **questo risultato o il suo opposto**, a seconda del fatto che a e b siano concordi o discordi.

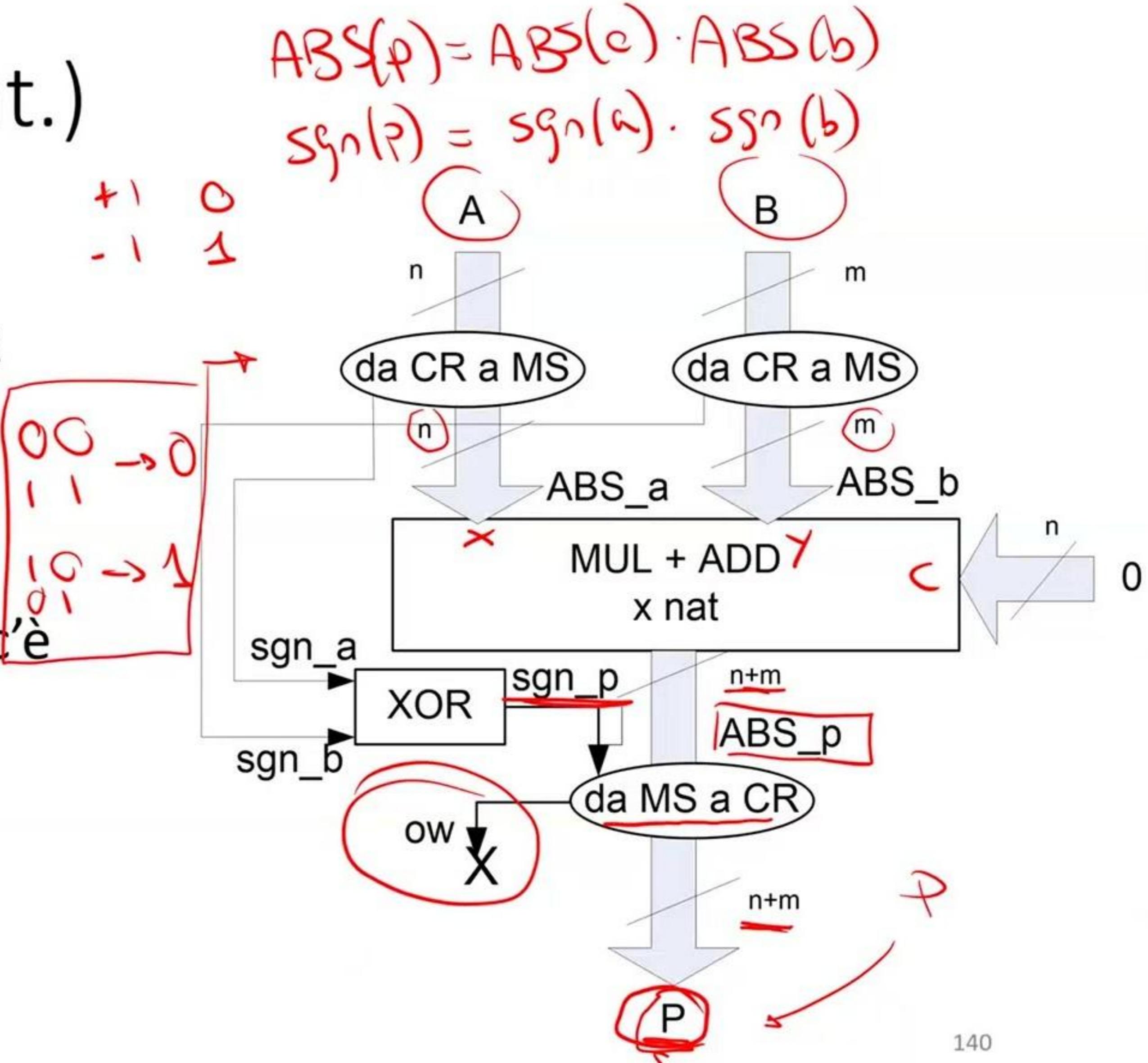
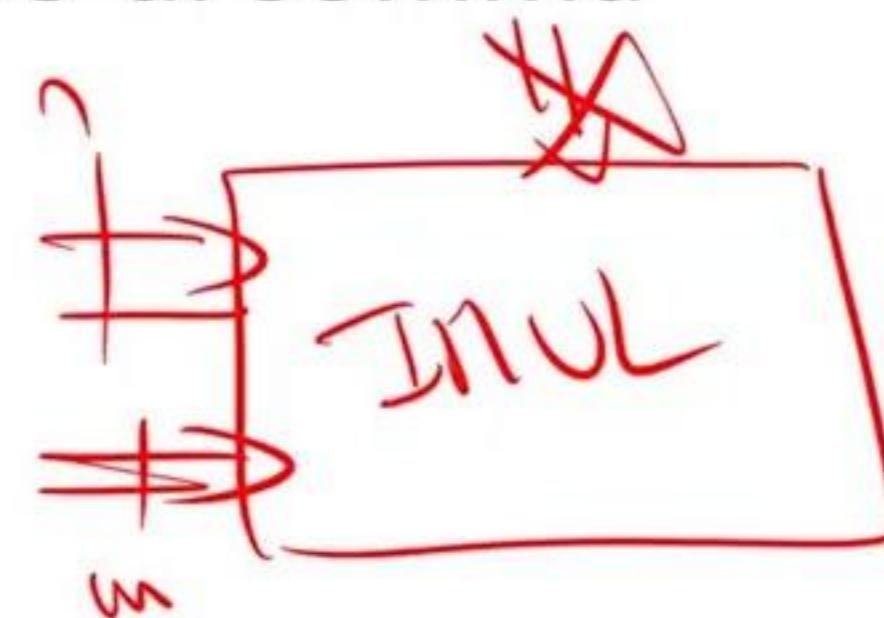
$$\operatorname{sgn}(x) = \begin{cases} +1 & x \geq 0 \\ -1 & x < 0 \end{cases}$$
$$x = \operatorname{sgn}(x) \cdot ABS(x)$$

Moltiplicazione (cont.)

- Posso trascurare l'overflow in uscita dal convertitore MS/CR

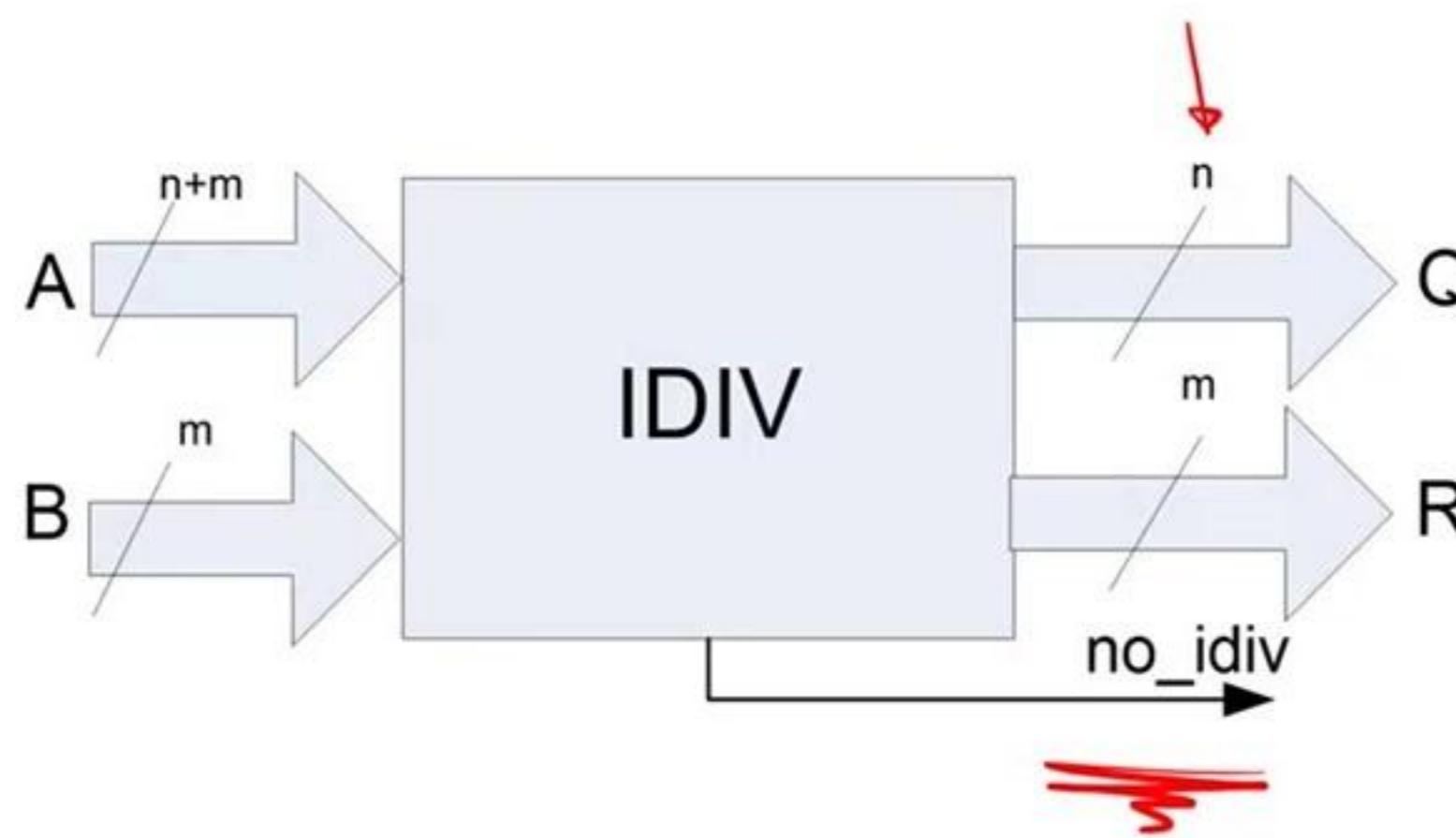
- Non ci sono problemi di rappresentabilità

- Nel circuito per gli interi non c'è un ingresso di somma



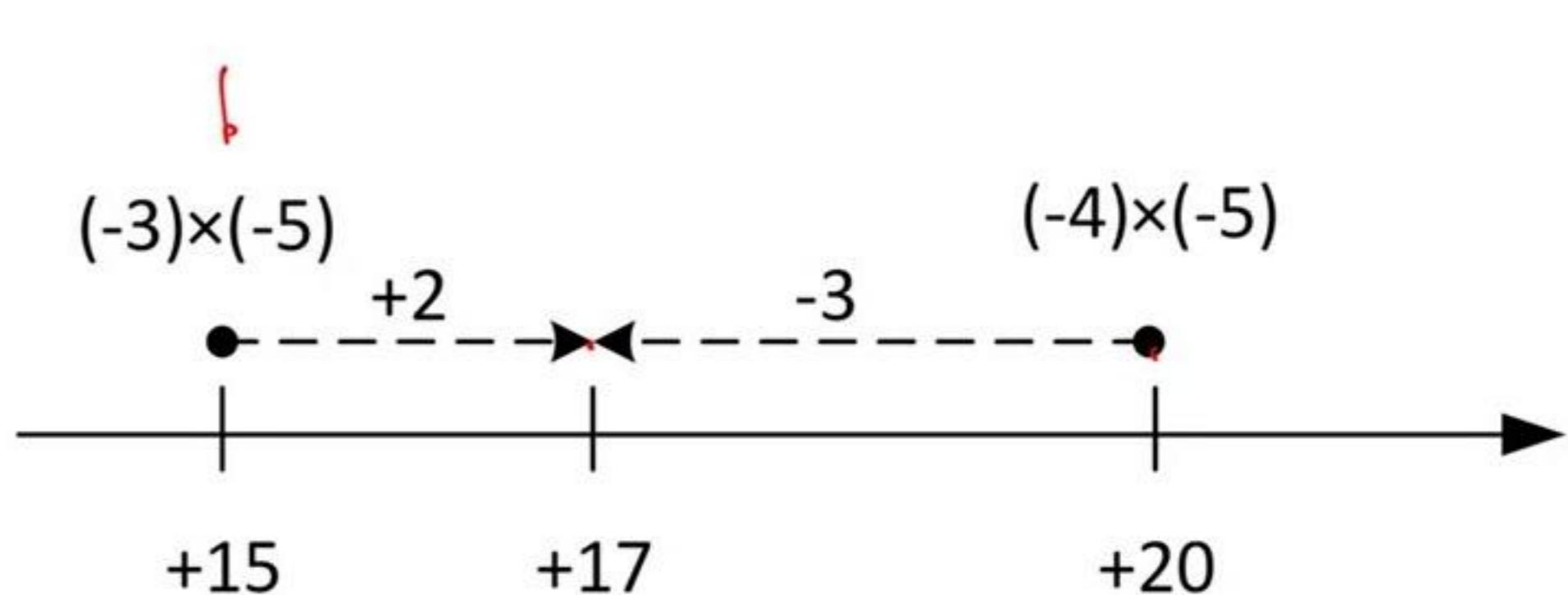
Divisione

- Dati A su $n + m$ cifre e B su m cifre, $a \leftrightarrow A$, $b \leftrightarrow B$
- Voglio calcolare Q ed R , rispettivamente su $\underbrace{n}_{\leftarrow}$ ed $\underbrace{m}_{\leftarrow}$ cifre, tali che
 - $q \leftrightarrow Q$ e $r \leftrightarrow R$
 - $a = q \cdot b + r$ *(intero)*
- **q può non essere un intero rappresentabile su n cifre, e quindi Q può non esistere**

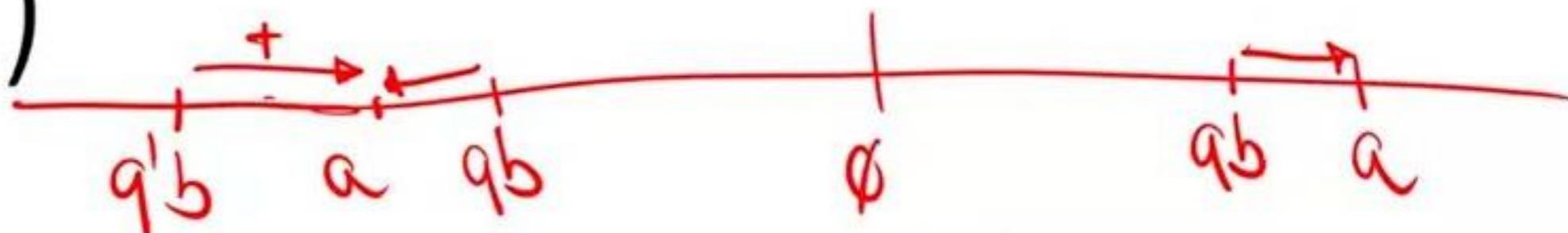


Divisione tra interi

- Divisione **naturale** (teorema della divisione con resto)
 - Risultato unico se $0 \leq r \leq b - 1$
- Nella divisione **intera** questa condizione non ha senso
 - b può essere un numero negativo
- Generalizzazione: $\underline{ABS(r)} < \underline{ABS(b)}$
 - Non basta
 - $a = +17, b = -5$:
 - $q = -3, r = +2$, oppure $q = -\underline{4}, r = -3$.
 - In entrambi i casi, $ABS(r) < ABS(b)$.



Divisione tra interi (cont.)



- Devo aggiungere **un'altra condizione**.
- La condizione che si sceglie è che **il resto abbia il segno del dividendo**. Le ipotesi che rendono unico il risultato sono quindi:

$$\rightarrow \begin{cases} ABS(r) < ABS(b) \\ \text{sgn}(r) = \text{sgn}(a) \end{cases}$$

- **quoziente approssimato per troncamento**

• $q \cdot b$ è sempre più vicino all'origine rispetto ad a

• la divisione che abbiamo usato finora, invece, approssima **a sinistra**

Divisione tra interi (cont.)

$$a = q b + r$$

- Posso riscrivere la divisione evidenziando modulo e segno degli operandi:

$$\underline{\operatorname{sgn}(a) \cdot ABS(a)} = q \cdot \underline{\operatorname{sgn}(b) \cdot ABS(b)} + \underline{\operatorname{sgn}(r) \cdot ABS(r)}$$

- Ma: $\underline{\operatorname{sgn}(a) = \operatorname{sgn}(r)}$ $\operatorname{sgn}(k)^2 = 1$

- Moltiplicando entrambi i membri per $\operatorname{sgn}(a)$, si ottiene:

$$\underline{ABS(a)} = [\underline{q} \cdot \underline{\operatorname{sgn}(b) \cdot \operatorname{sgn}(a)}] \cdot \underline{ABS(b)} + \underline{ABS(r)}$$

$\brace{ABS(q)}$

Divisione tra interi (cont.)

$$ABS(a) = ABS(q) \cdot ABS(b) + ABS(r)$$

- Posso calcolare $ABS(r)$ e $ABS(q)$ utilizzando un modulo **divisore tra naturali**
- purché $ABS(q)$ sia **un numero (naturale) rappresentabile su n cifre**.
- Per testare se questo è vero, dobbiamo controllare se:

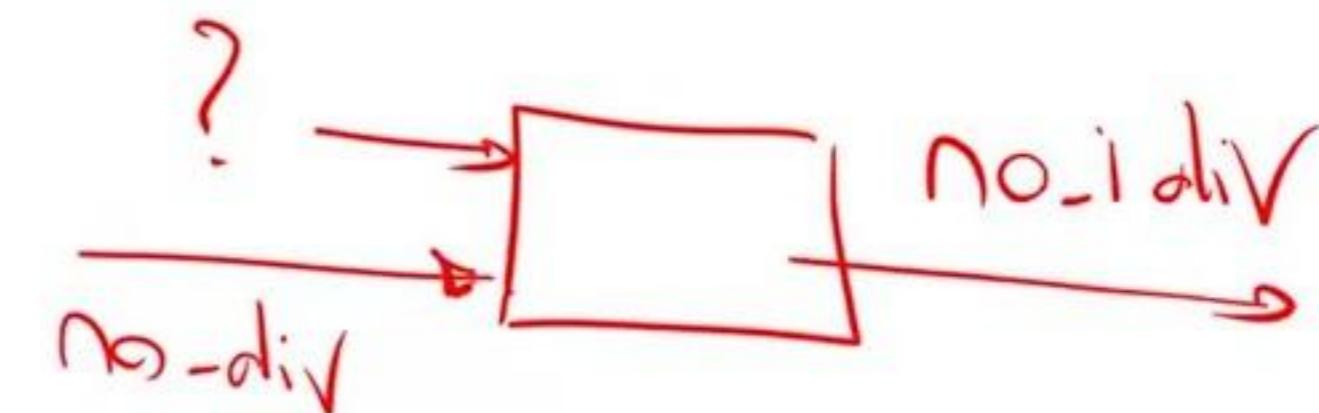


$$ABS(a) < \beta^n \cdot ABS(b)$$

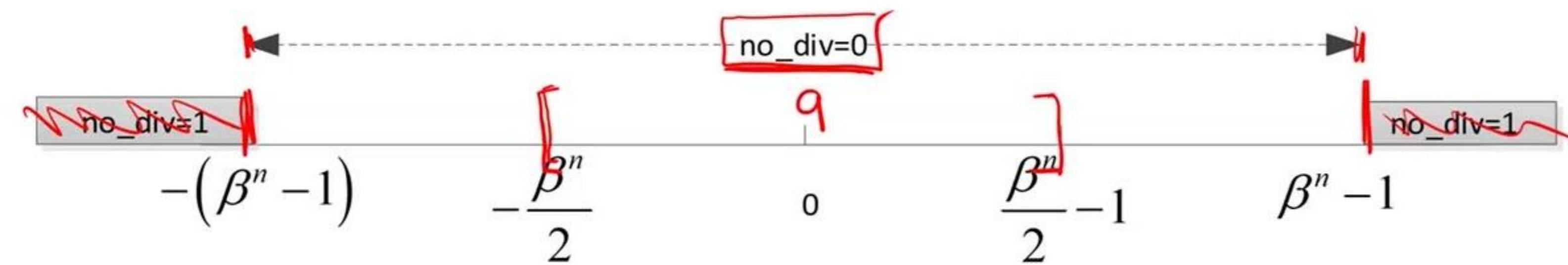
~~no-div~~

- si fa con un sottrattore ad $n + m$ cifre (lo stesso che useremmo per testare la fattibilità di una divisione naturale).
- Se questa diseguaglianza è falsa, il quoziente ~~della~~ della divisione naturale di sopra è **un numero naturale che sta su più di n cifre**, $ABS(q) \geq \beta^n$

Divisione tra interi (cont.)



- Quindi, la **fattibilità della divisione (naturale) tra i valori assoluti è condizione necessaria per la fattibilità della divisione intera.**



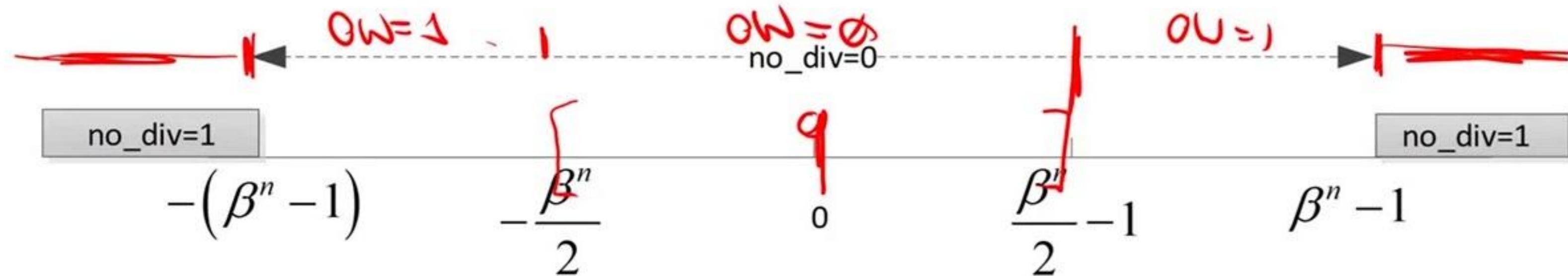
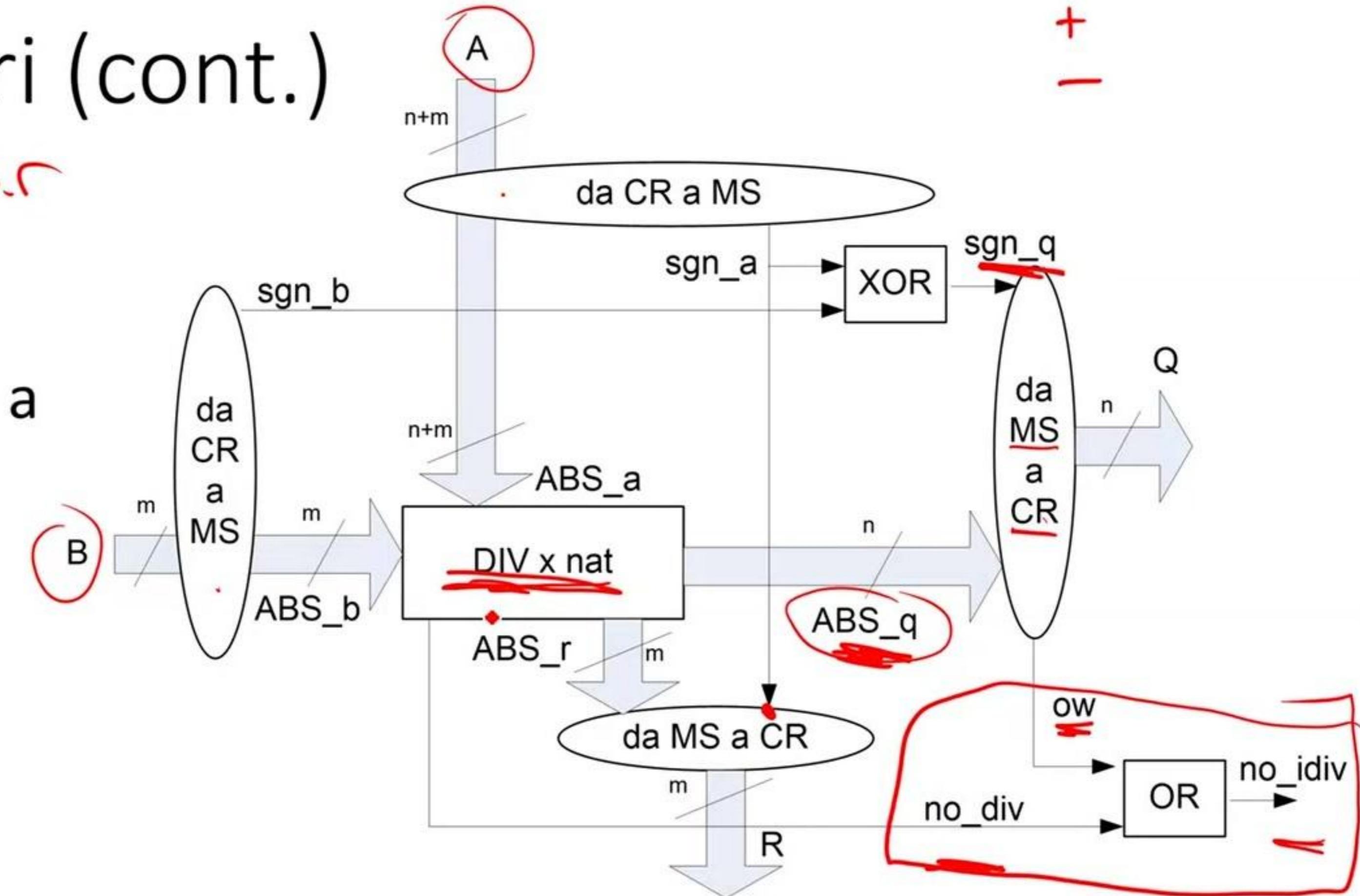
- non è però una condizione sufficiente.** Quando $no_div = 0$, infatti, non abbiamo garanzia che q sia un numero intero rappresentabile su n cifre
 - Dobbiamo aggiungere qualcosa

Divisione tra interi (cont.)

ABS_q

ABS_r

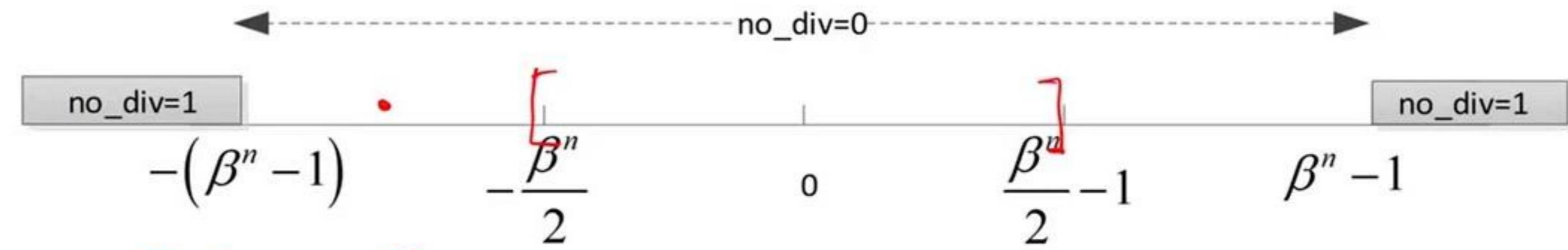
- Nella conversione della rappresentazione di q da MS a CR posso avere **overflow**
- $\text{no_idiv} = \text{no_div OR ow}$



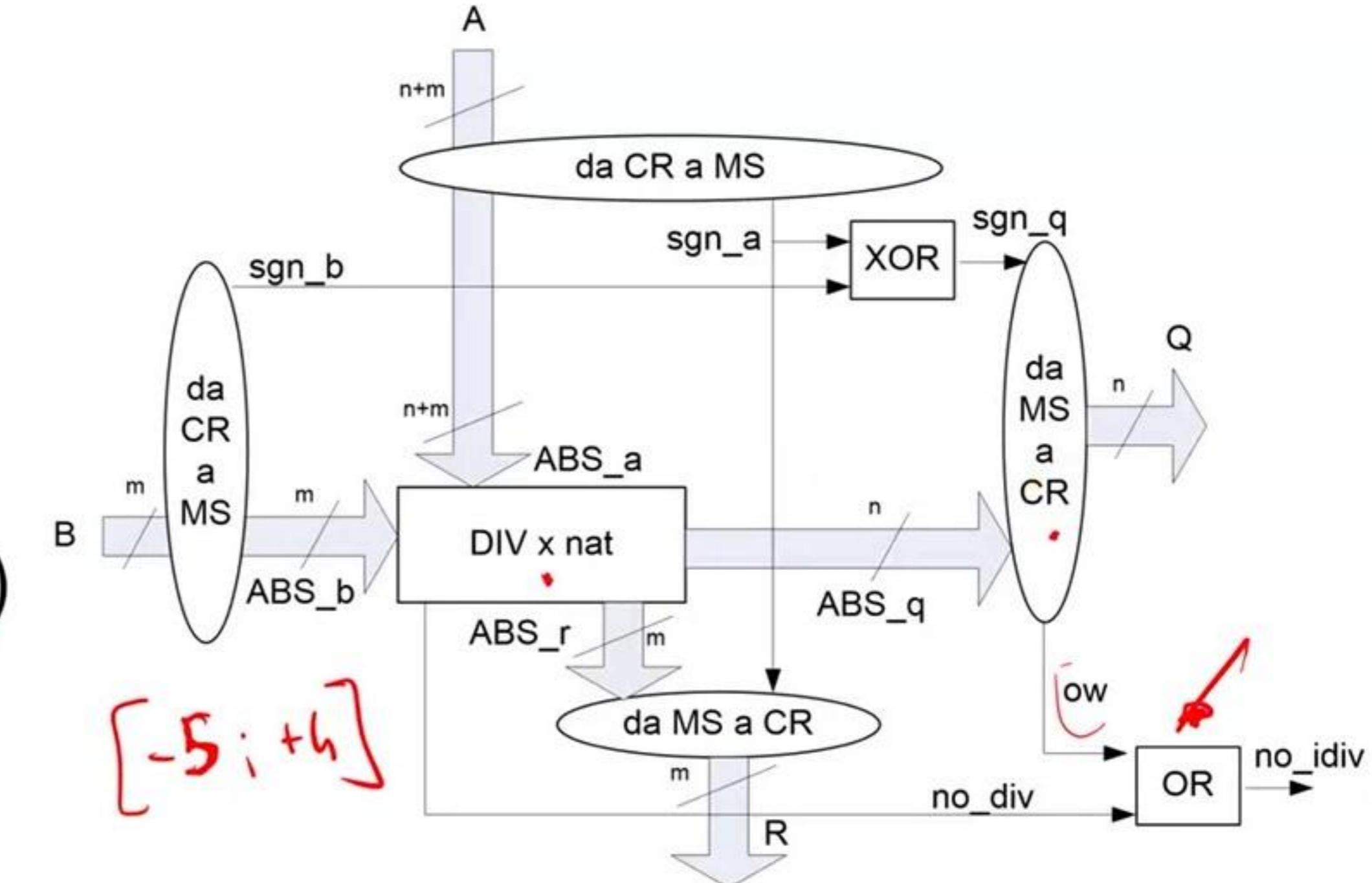
$$B = |+28|_{100}$$

Esempio

- $(-223) \div (+28)$
- $a = -223, b = +28, n = 1, m = 2$
- $A = |-223|_{10^3} = 777, B = 28$
- Divisione tra i valori assoluti: $223 \div 28$.
 - Tale divisione naturale è fattibile (no_div=0)
 - quoziente su 1 cifra, $Q^* = ABS(q) = 7$
 - $R^* = ABS(r) = 27.$
- I segni degli operandi sono discordi
 - Il quoziente q è negativo
 - $q = -7$ non rappresentabile su una cifra in base 10
 - $ow=1 \Rightarrow no_idiv=1$

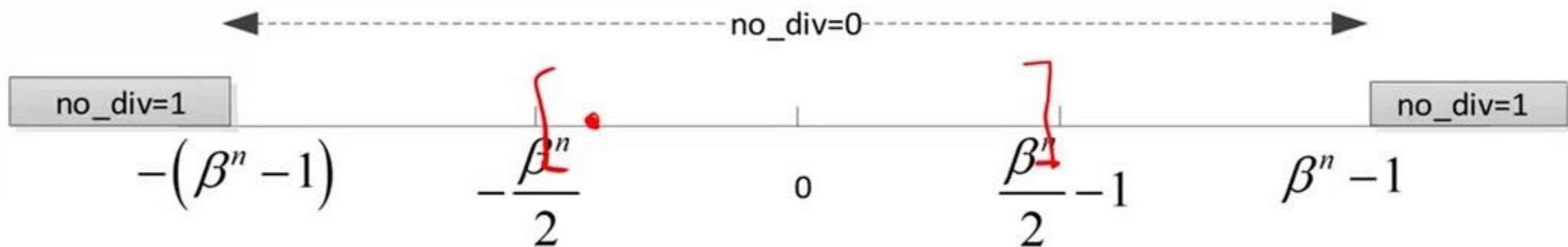


$$\left[-\frac{10^2}{2}; +\frac{10^2 - 1}{2} \right]$$

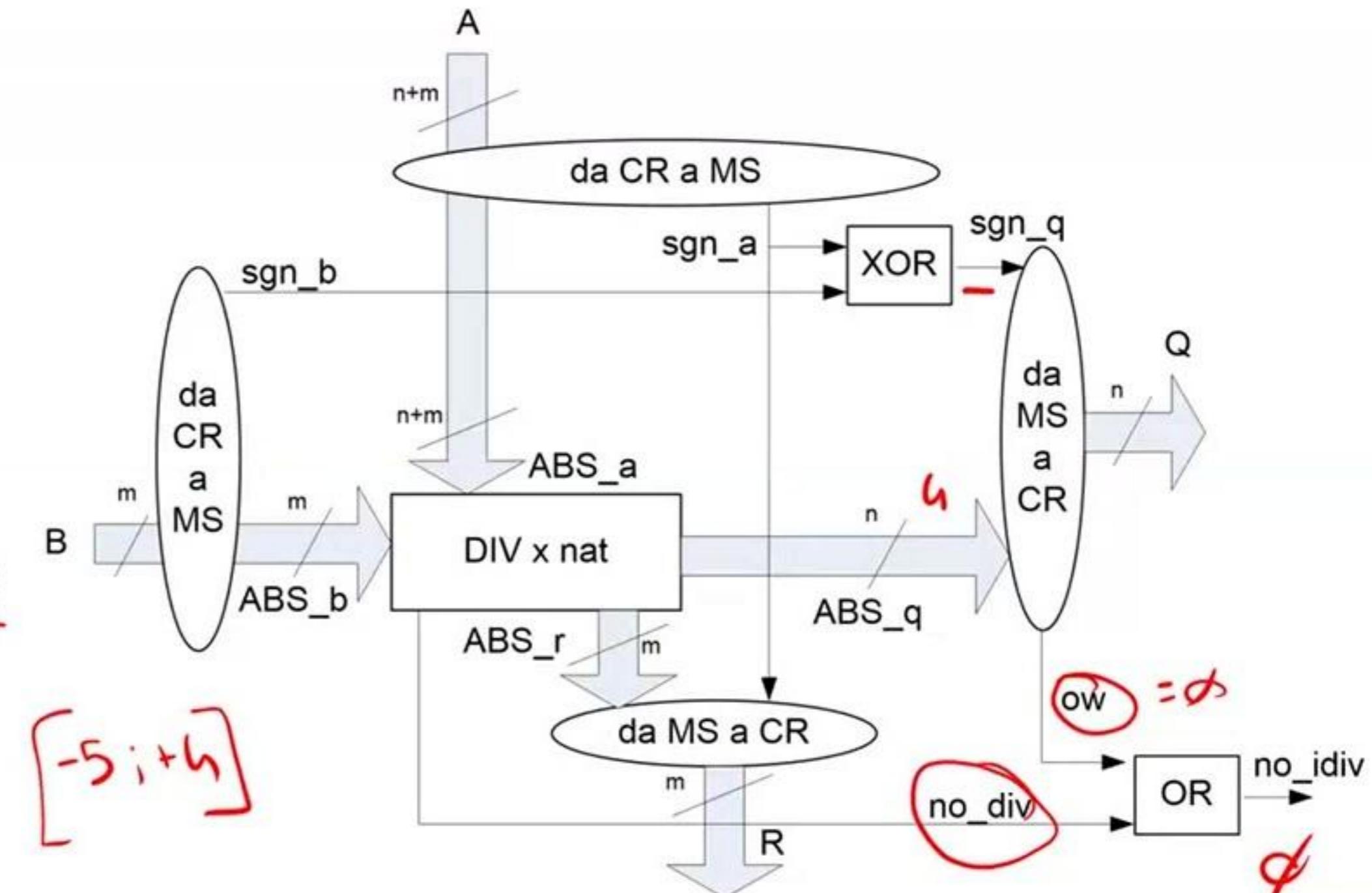


$$[-5; +6]$$

Esempio



- $(-123) \div (+28)$
- $a = -123, b = +28, n = 1, m = 2$
- $A = |-123|_{10^3} = 877, B = 28$
- Divisione tra i valori assoluti: $123 \div 28$.
 - Tale **divisione naturale è fattibile** (no div=0)
 - quoziente su 1 cifra, $Q^* = ABS(q) = 4$
 - $R^* = ABS(r) = 11$
- I segni degli operandi sono **discordi**
 - Il **quoziente q** è **negativo**
 - $q = -4$ rappresentabile su una cifra in base 10
 - ow=0**



$$Q = |-4|_{10^1} = 6$$

$$r = -11, \text{ ed } R = |-11|_{10^2} = 89$$

Richiamo sull'Assembler

- Esistono **due istruzioni distinte** per la moltiplicazione e la divisione, **MUL/IMUL, DIV/IDIV**, che moltiplicano/dividono rispettivamente naturali ed interi.
- Infatti, anche se la moltiplicazione (divisione) tra interi si fa in modo simile a quella tra naturali (si può riciclare parte della circuiteria), sono necessarie anche altre operazioni, quali calcolo del valore assoluto, etc.