Architettura degli Elaboratori

Lezione 11 – Circuiti combinatori per l'artimetica binaria

Giuseppe Cota
Dipartimento di Scienze Matematiche Fisiche e Informatiche
Università degli Studi di Parma

Indice

- Semisommatore (half adder) e sommatore completo (full adder)
- Sommatore con calcolo anticipato del riporto
- Unità aritmetico logiche (ALU)

Sommatore (half adder) e sommatore completo (full adder)

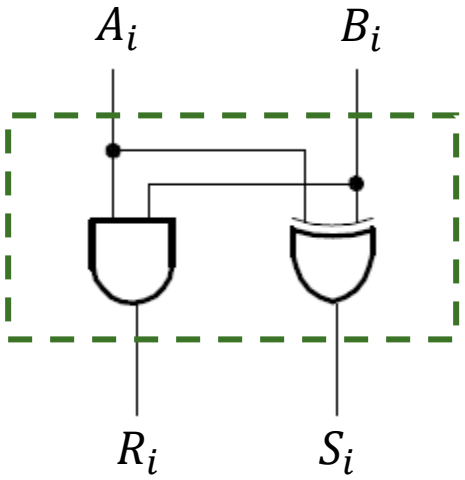
Sommatore

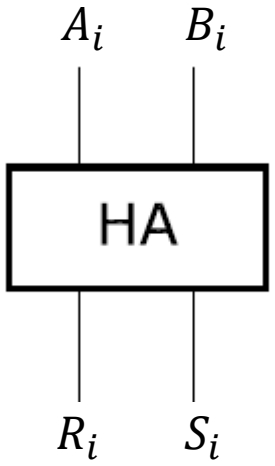
- Il circuito (combinatorio) base per il calcolo dell'addizione è l'addizionatore a un bit ("full adder" o "addizionatore completo").
- Ha in ingresso:
  - due addendi A_i e B_i, da un bit ciascuno
  - il riporto in ingresso R_{i-1}, da un bit
- Ha in uscita:
  - la somma S_i, da un bit
  - il riporto in uscita R_i, da un bit
- È composto da due **half adder** o **semisommatore**.

Semisommatore (half adder)

- Il semisommatore non tiene conto di un eventuale riporto in ingresso.

A_i	B_i	S_i	R_i
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1





versione blackbox

$$S_i = \overline{A_i}B_i + A_i\overline{B_i} = A_i \oplus B_i$$
$$R_i = A_iB_i$$

Somma binaria

- Per poter effettuare la somma di due numeri interi su più bit occorre tener conto dei riporti.

Riporti	R_{n-1}	R_{n-2}	...	R_{i-1}	...	R_0	R_{-1}	
A		A_{n-1}	...	A_i	...	A_1	A_0	+
B		B_{n-1}	...	B_i	...	B_1	B_0	=
S		S_{n-1}	...	S_i	...	S_1	S_0	

A_i	B_i	R_{i-1}	S_i	R_i
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

$$S_i = (A_i \oplus B_i) \oplus R_{i-1}$$

$$R_i = A_i B_i + (A_i \oplus B_i) R_{i-1}$$

Sommatore completo (full adder)

A_i	B_i	R_{i-1}	S_i	R_i
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

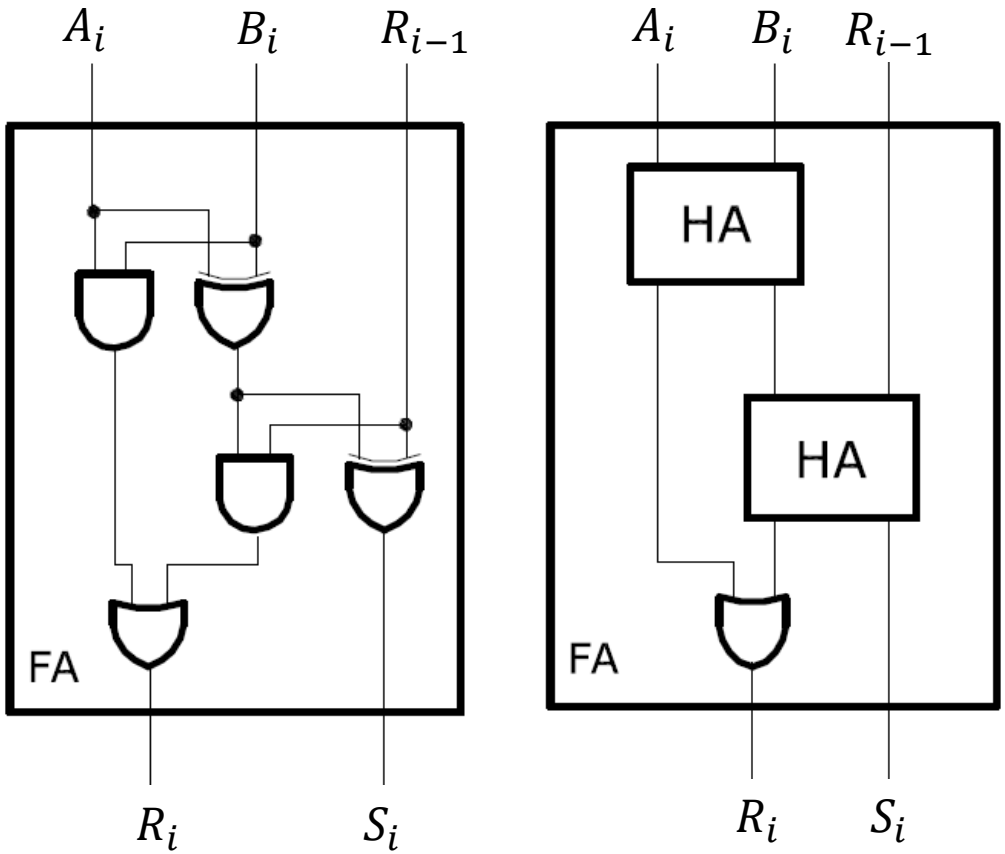
$$S_i = (A_i \oplus B_i) \oplus R_{i-1}$$

$$R_i = A_i B_i + (A_i \oplus B_i) R_{i-1}$$

Generatore di riporto

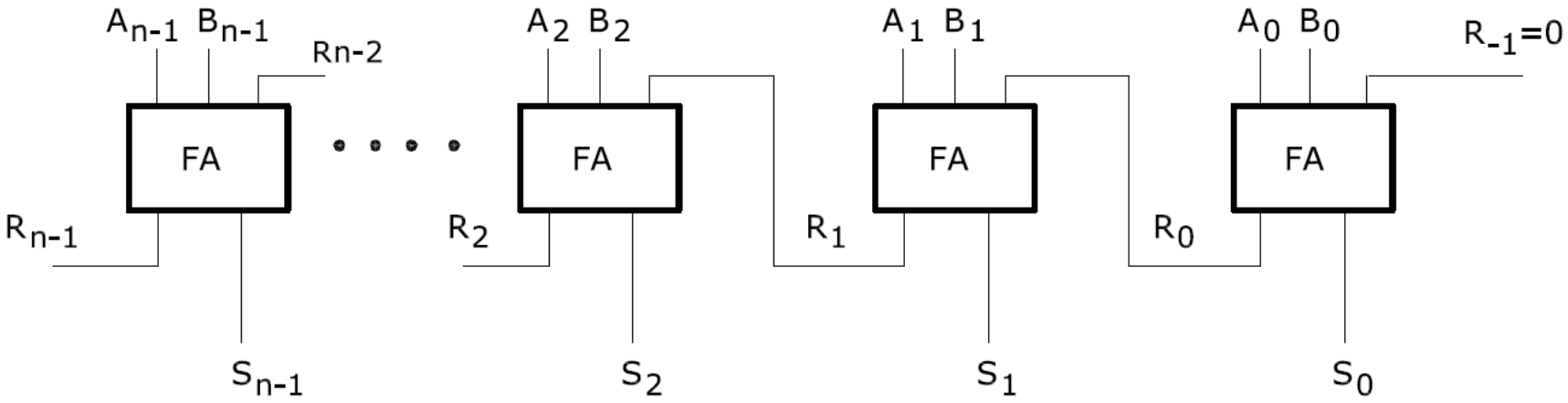
Propagatore di riporto

Versione semplificata



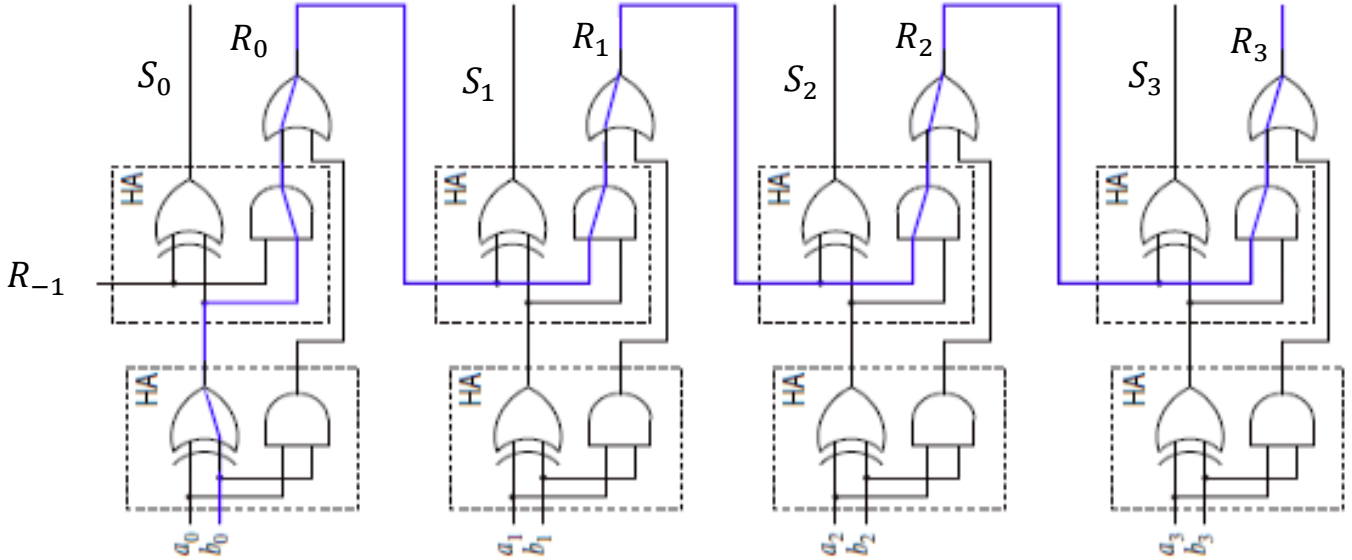
Ripple-carry adder a n bit

Ripple-carry adder a n bit
(Sommatore a propagazione di riporto)



Sommatore con calcolo anticipato del riporto

Tempo necessario per il calcolo per un ripple-carry adder

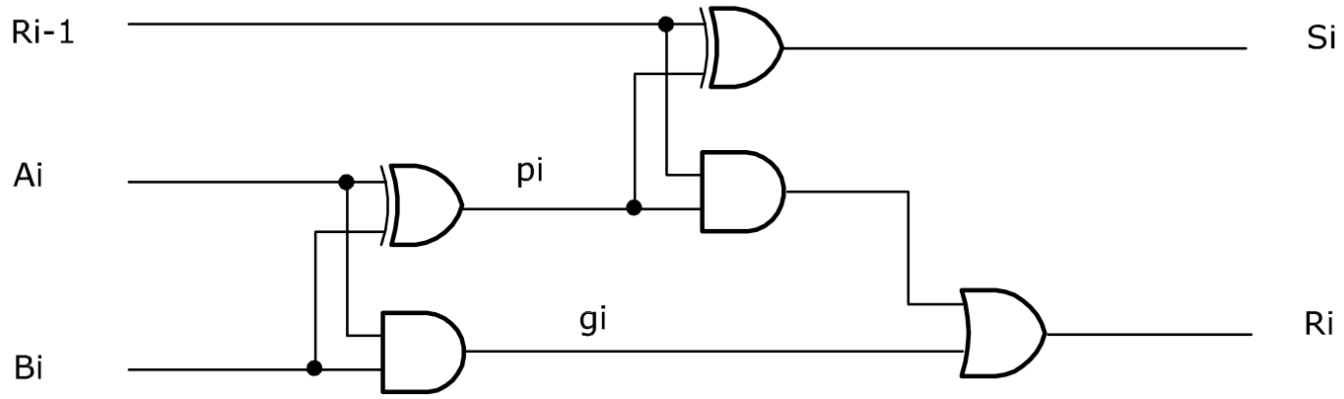


- La rete per il calcolo della somma può produrre un risultato dopo che per ogni FA è divenuto stabile il riporto in ingresso R_i.
- Nel peggiore dei casi, se τ è il tempo di commutazione di una qualunque porta (ritardo di una porta):
  - Tempo per calcolare R_{n-1}: $\Delta_{R_{n-1}} = n \cdot 2\tau + \tau = (2n+1)\tau$
  - Tempo per calcolare S_{n-1}: $\Delta_{S_{n-1}} = \Delta_{R_{n-2}} + \tau = (2(n-1)+1)\tau + \tau = 2n\tau$

Rete lenta!

Calcolo anticipato del riporto

- È possibile ridurre i tempi di calcolo della somma con la tecnica del **calcolo anticipato del riporto**.
- Se poniamo:
  - **Generatore di riporto** $g_i = A_i B_i$
  - **Propagatore di riporto** $p_i = A_i \oplus B_i$
- Si ha:
  - $S_i = (A_i \oplus B_i) \oplus R_{i-1} = p_i \oplus R_{i-1}$
  - $R_i = A_i B_i + (A_i \oplus B_i) R_{i-1} = g_i + p_i R_{i-1}$



Full adder

Calcolo anticipato del riporto
Esempio adder 4 bit

Esempio adder 4 bit

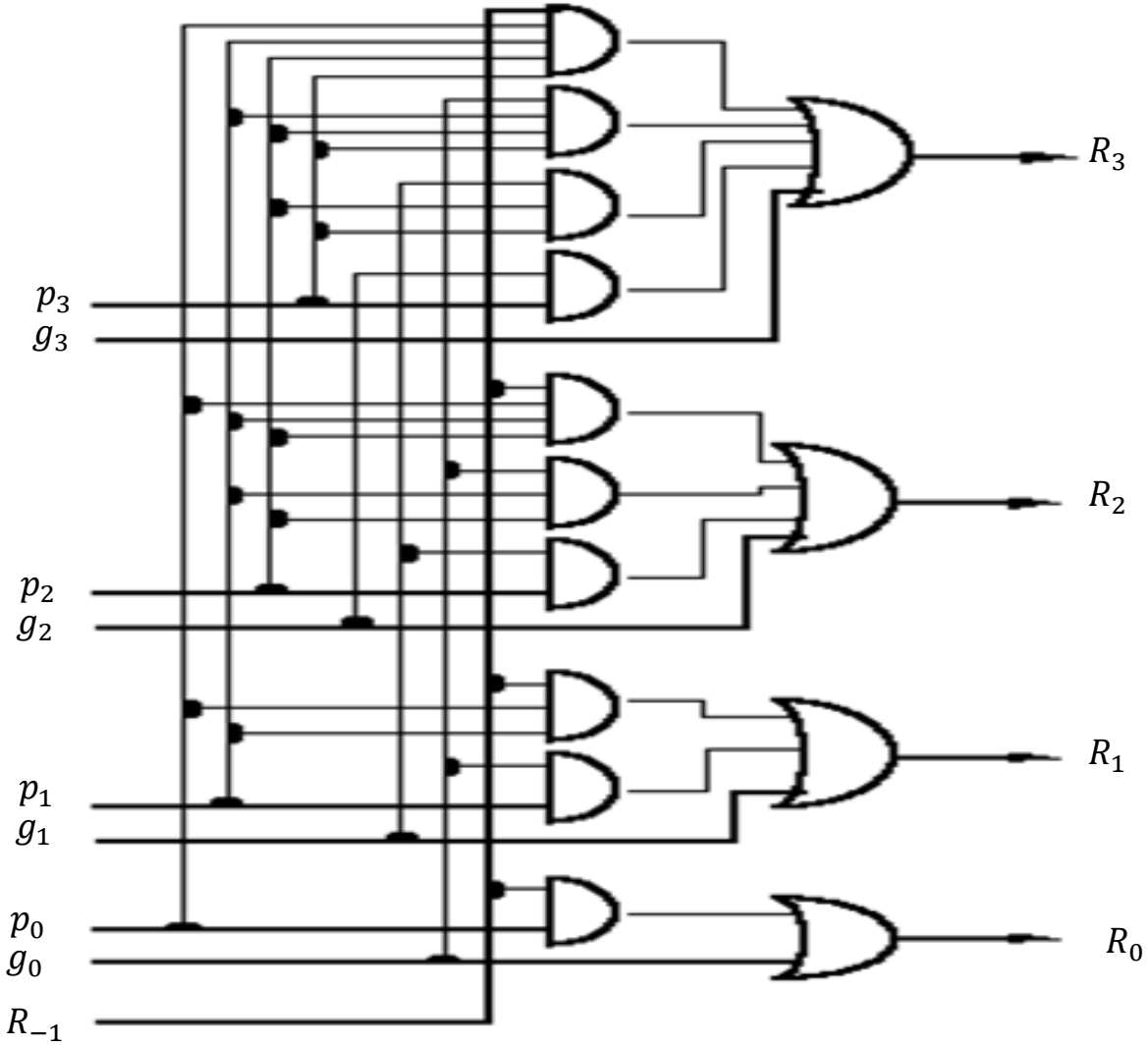
$$R_0 = g_0 + p_0 R_{-1}$$
$$R_1 = g_1 + p_1 R_0 = g_1 + p_1 g_0 + p_1 p_0 R_{-1}$$
$$R_2 = g_2 + p_2 R_1 = g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 R_{-1}$$
$$R_3 = g_3 + p_3 R_2 = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 R_{-1}$$

Se indico $G = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0$ e $P = p_3 p_2 p_1 p_0$ posso scrivere:

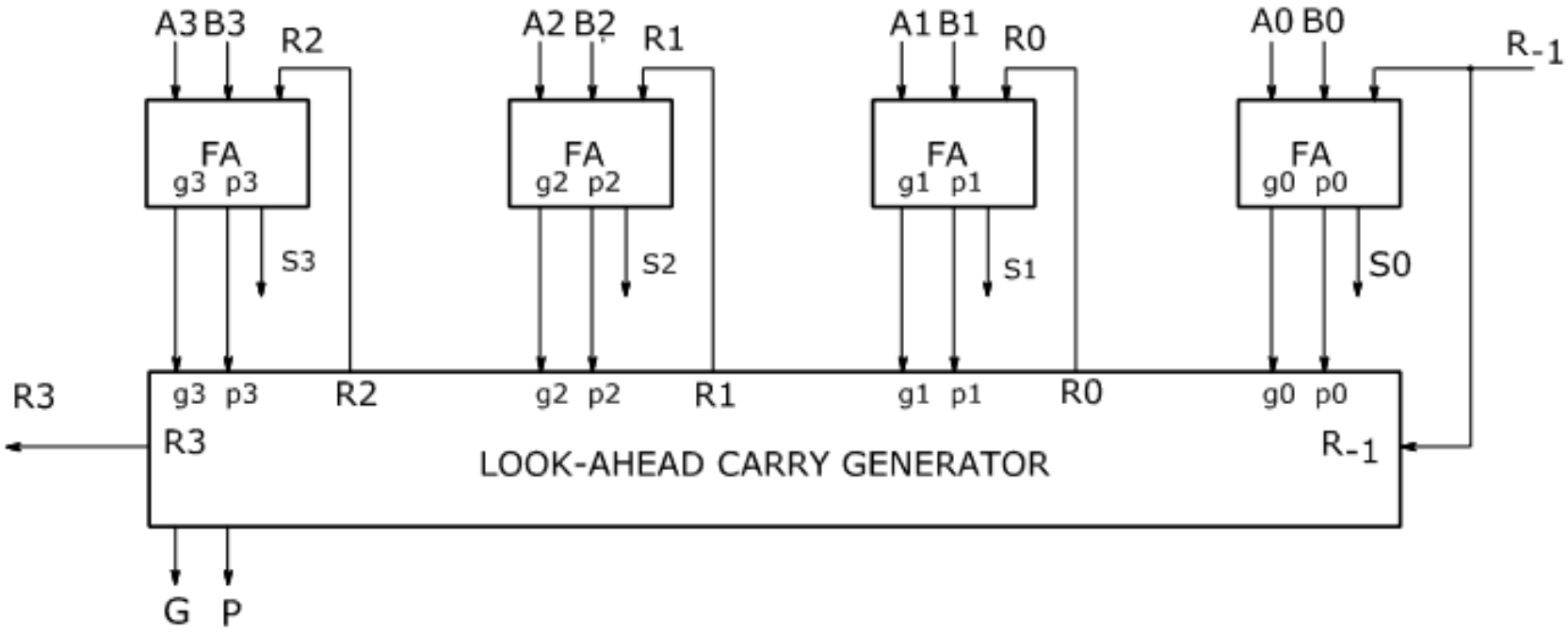
$$R_3 = G + P R_{-1}$$

- Per il calcolo anticipato del riporto serve quindi un circuito in più chiamato **look-ahead carry generator**
  - **Complico un po' il circuito sommatore, ma il calcolo è più veloce rispetto al sommatore di ripple.**

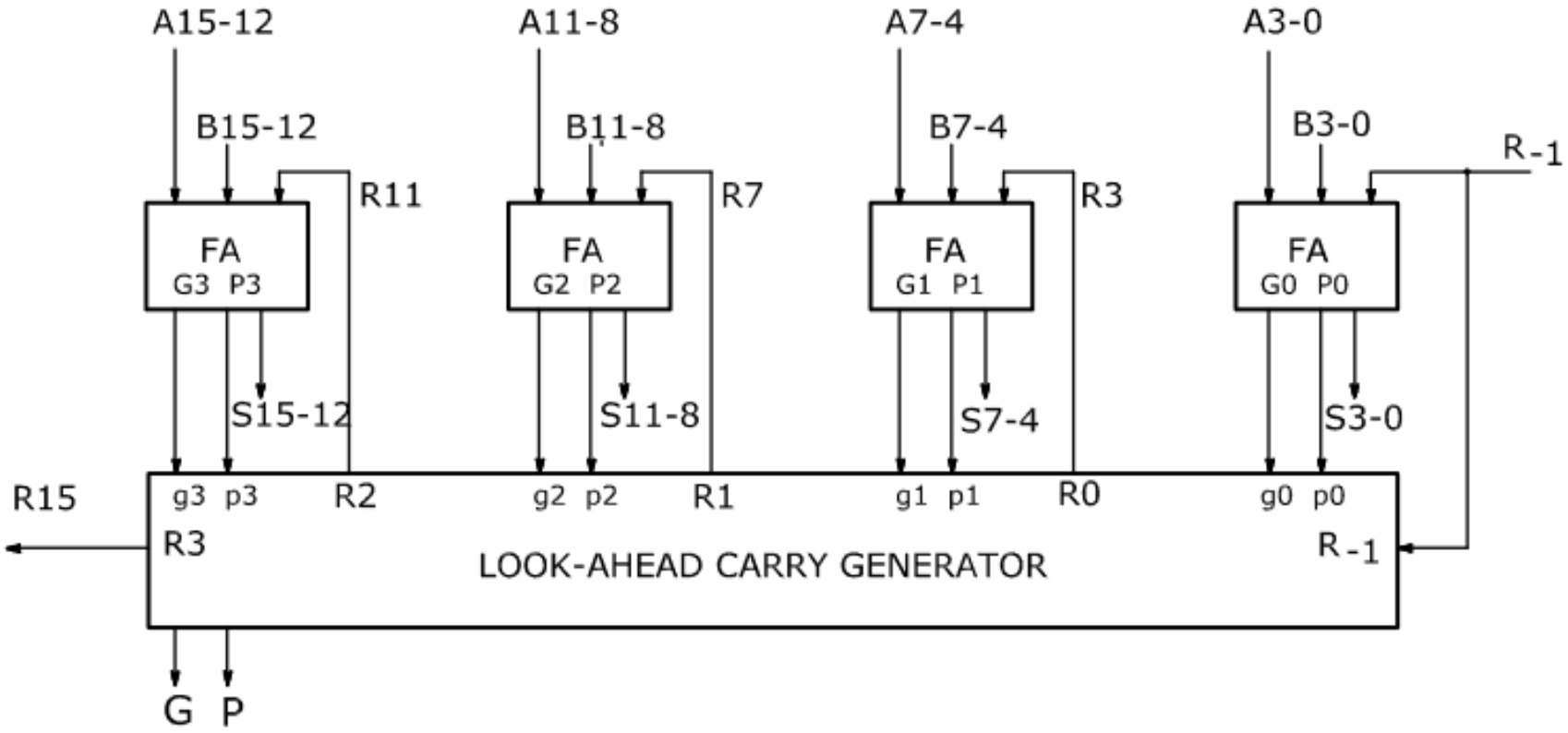
Look-ahead carry generator per sommatore a 4 bit



Sommatore 4 bit con circuito look-ahead carry generator



Sommatore a 16 bit composto da 4 sommatori a 4 bit



Terminologia: Full adder a 4 bit = sommatore composto da 4 full adder a 1 bit

Unità aritmetico logiche (arithmetic logic unit, ALU)

Unità aritmetico logiche (arithmetic logic unit, ALU)

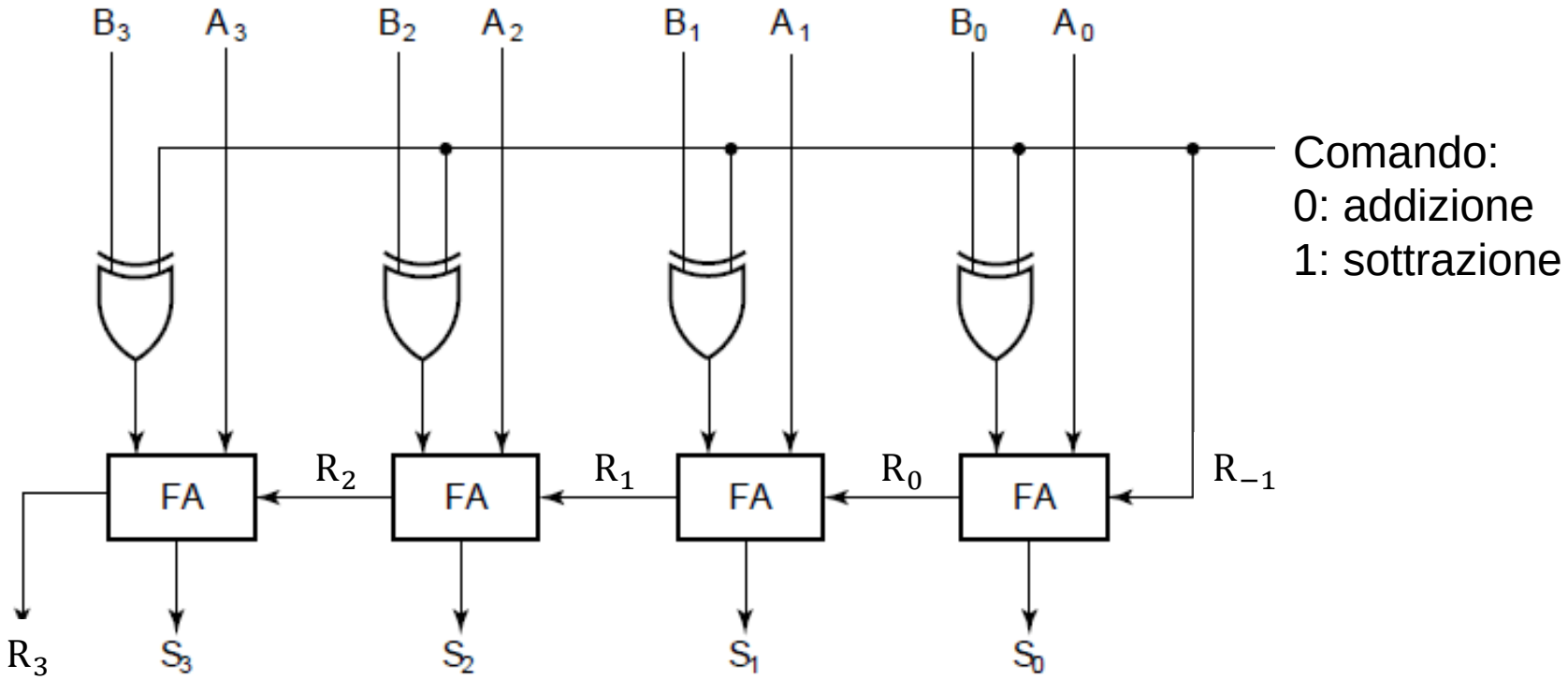
- Il circuito (combinatorio) algebrico completo è l'unità aritmetica-logica (Arithmetic-Logic Unit, ALU).
- È un dispositivo programmabile, capace di eseguire un repertorio di operazioni (per numeri interi):
  - aritmetiche (addizione, sottrazione, moltiplicazione, ecc)
  - logiche (negazione, somma e prodotto logico bit a bit, ecc)
  - di confronto (uguale, diverso, maggiore, ecc)
  - Il repertorio di operazioni varia secondo la ALU.
  - Esistono anche ALU per numeri reali.

Sottrazioni

- Se uso il complemento a due per rappresentare i numeri interi, allora con il sommatore posso fare sia somme che sottrazioni
- Se A e B sono due interi positivi
  - $A + B$
  - $A - B = A + (-B) = A + \tilde{B} + 1$

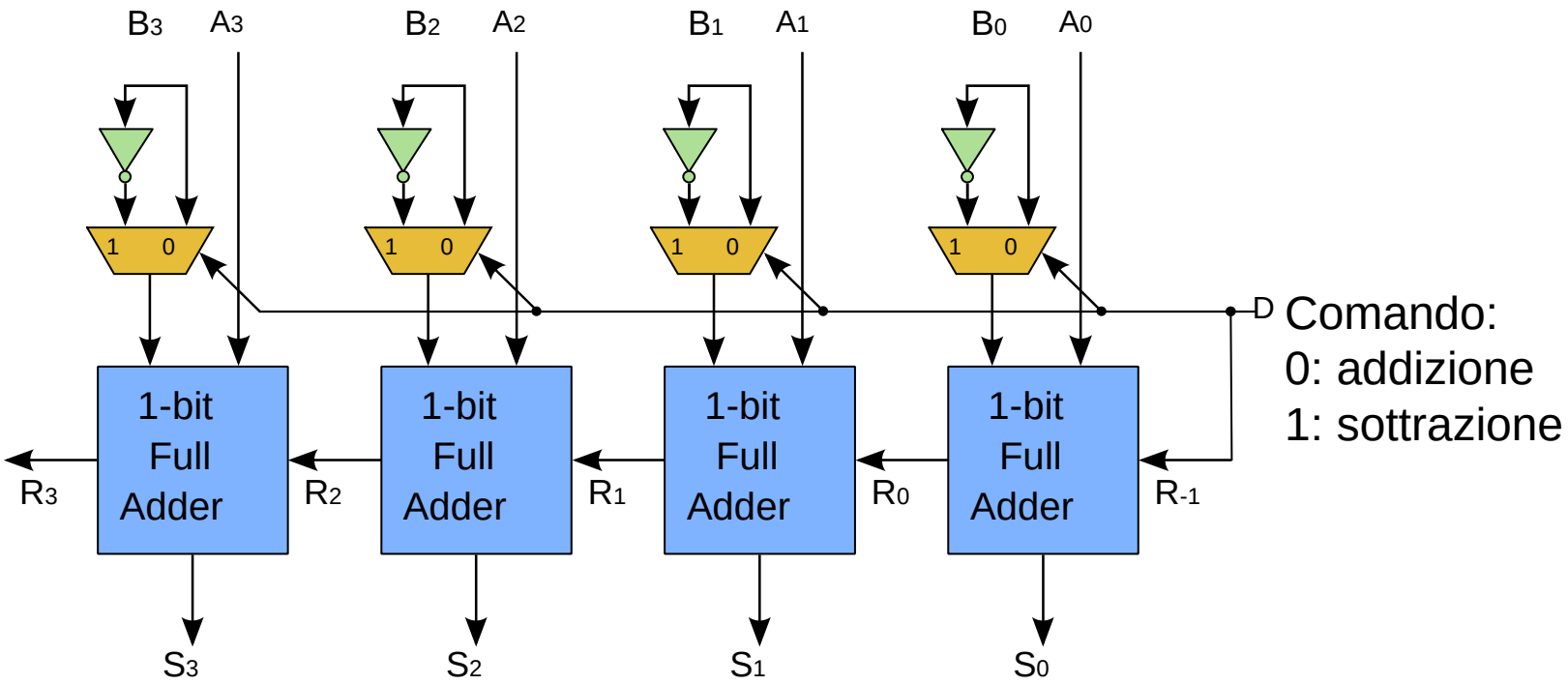
Sommatore (di ripple) per sottrazioni versione 1

Ripple-carry adder (sommatore a propagazione di riporto) per sottrazioni/addizioni con XOR



Sommatore (di ripple) per sottrazioni versione 2

Ripple-carry adder per sottrazioni/addizioni con multiplexer



Overflow con numeri interi con complemento a 2

- Si verifica un overflow in una somma di due numeri di n cifre in complemento a 2 se e solo se i riporti in colonna n e $n+1$ sono diversi

Riporto	R_3	R_2	R_1	R_0	R_{-1}		Decimale
	0	1	1	0			
		0	1	1	1	+	7
		0	0	1	0	=	2
Totale		1	0	0	1		-7
		S_3	S_2	S_1	S_0		

$\neq$

Overflow

Overflow con numeri interi con complemento a 2

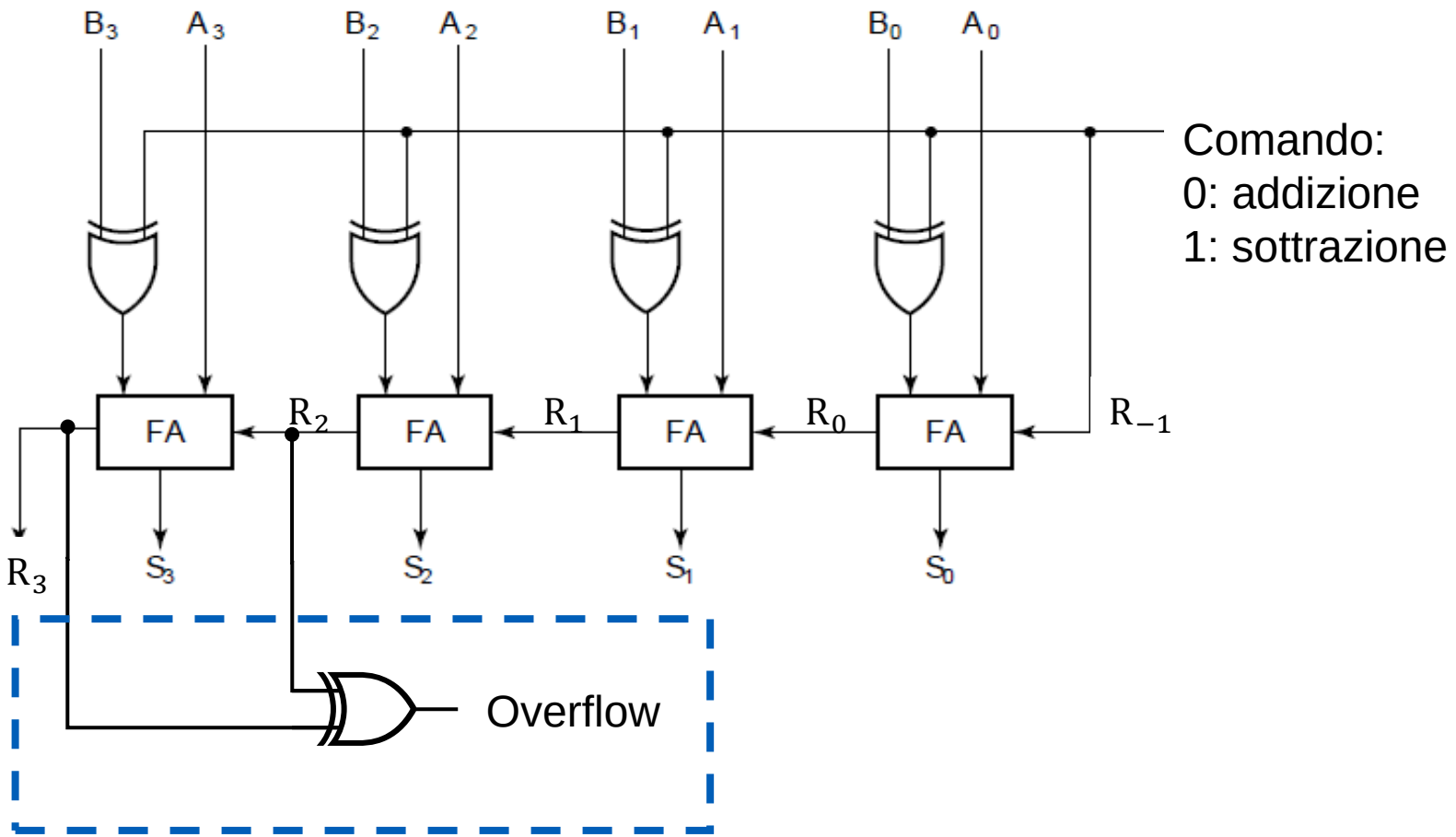
- Si verifica un overflow in una somma di due numeri di n cifre in complemento a 2 se e solo se i riporti in colonna n e $n+1$ sono diversi

Riporto	R_3	R_2	R_1	R_0	R_{-1}		Decimale
	0	1	1	0			
		1	0	1	1	+	-5
		1	1	0	0	=	-4
Totale		0	1	1	1		7
		S_3	S_2	S_1	S_0		

$\neq$

Overflow

Rete logica per controllo overflow versione 1



Controllo overflow

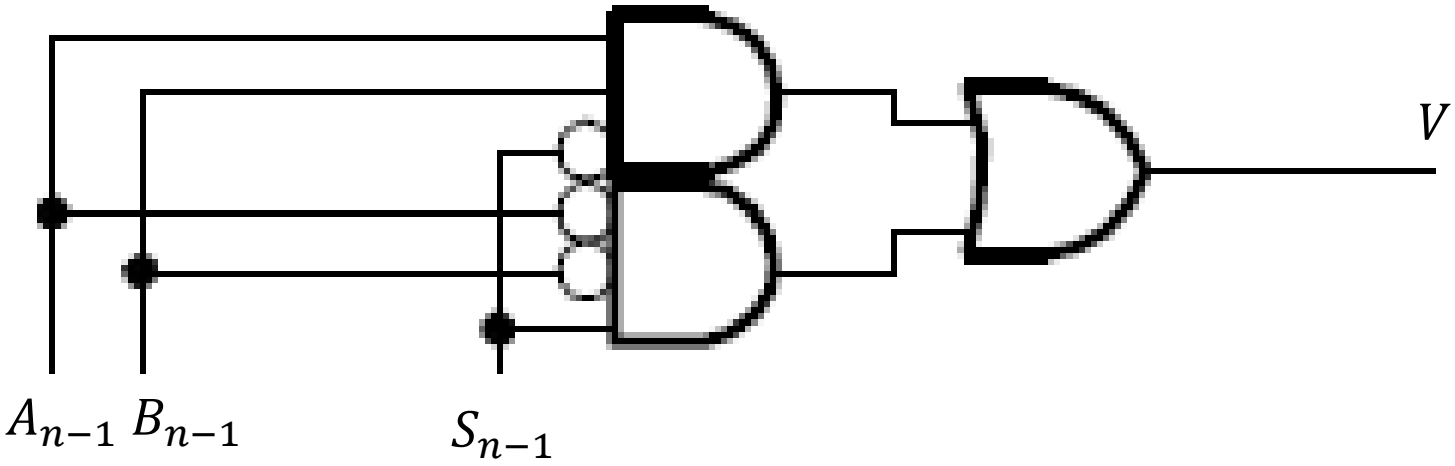
- *In modo alternativo*, dati due numeri interi a n bit posso dire che ho un overflow:
  - Se sommo due interi positivi e ottengo un numero negativo; **oppure**
  - Se sommo due interi negativi e ottengo un numero positivo.

Riporti	R_{n-1}	R_{n-2}	…	R_{i-1}	…	R_0	R_{-1}	
A		A_{n-1}	…	A_i	…	A_1	A_0	$+$
B		B_{n-1}	…	B_i	…	B_1	B_0	$=$
S		S_{n-1}	…	S_i	…	S_1	S_0	

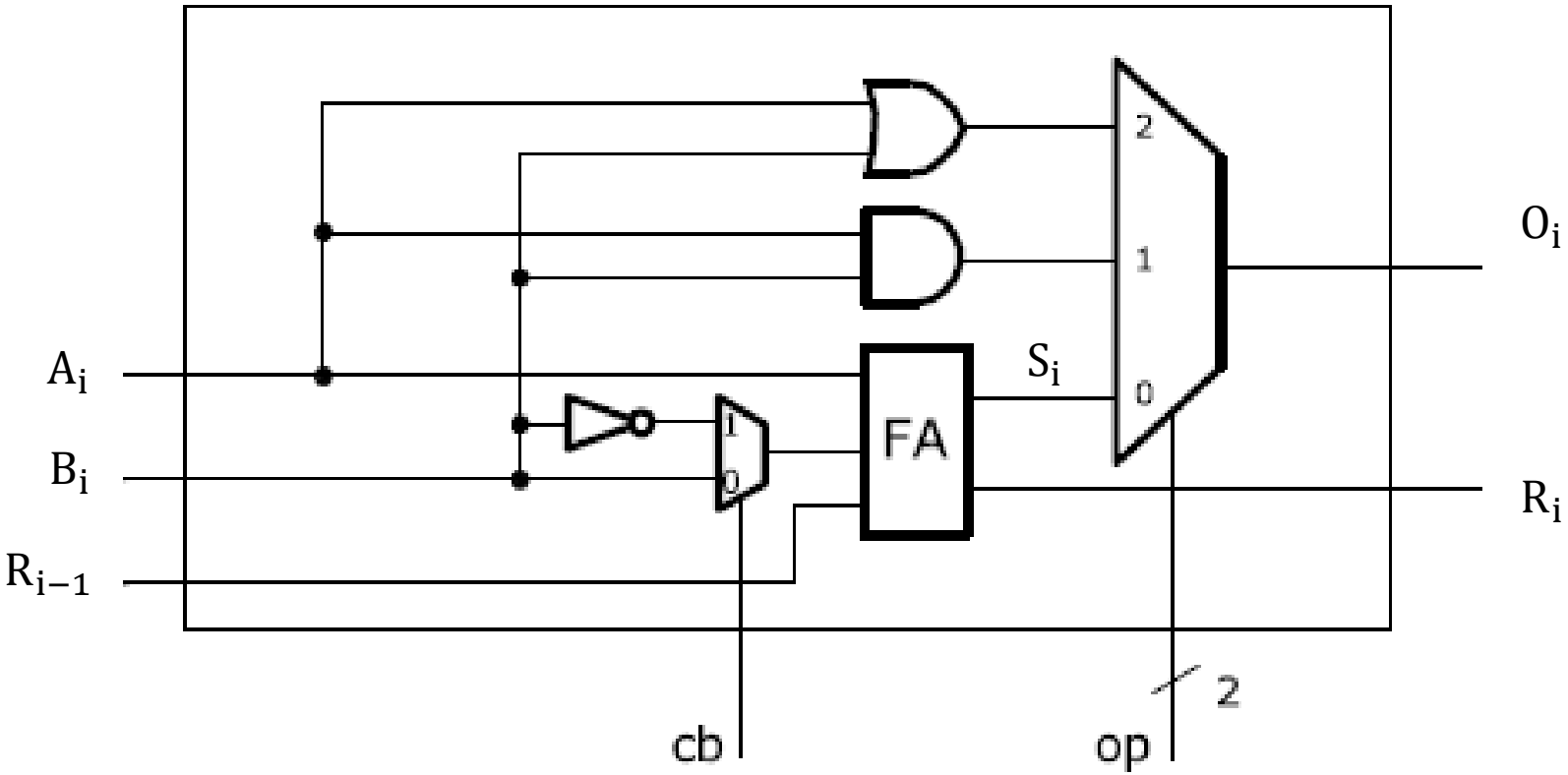
$$O = \overline{S_{n-1}} A_{n-1} B_{n-1} + S_{n-1} \overline{A_{n-1}}\ \overline{B_{n-1}}$$

Rete logica per controllo overflow versione 2

$$Overflow\ V = \overline{S_{n-1}} A_{n-1} B_{n-1} + S_{n-1} \overline{A_{n-1}}\ \overline{B_{n-1}}$$



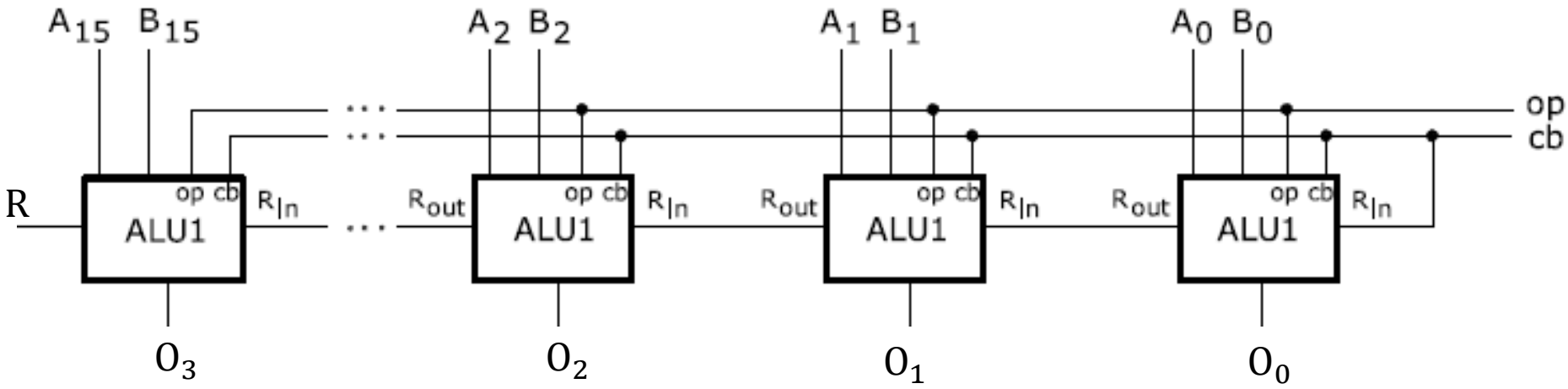
ALU semplificata a 1 bit



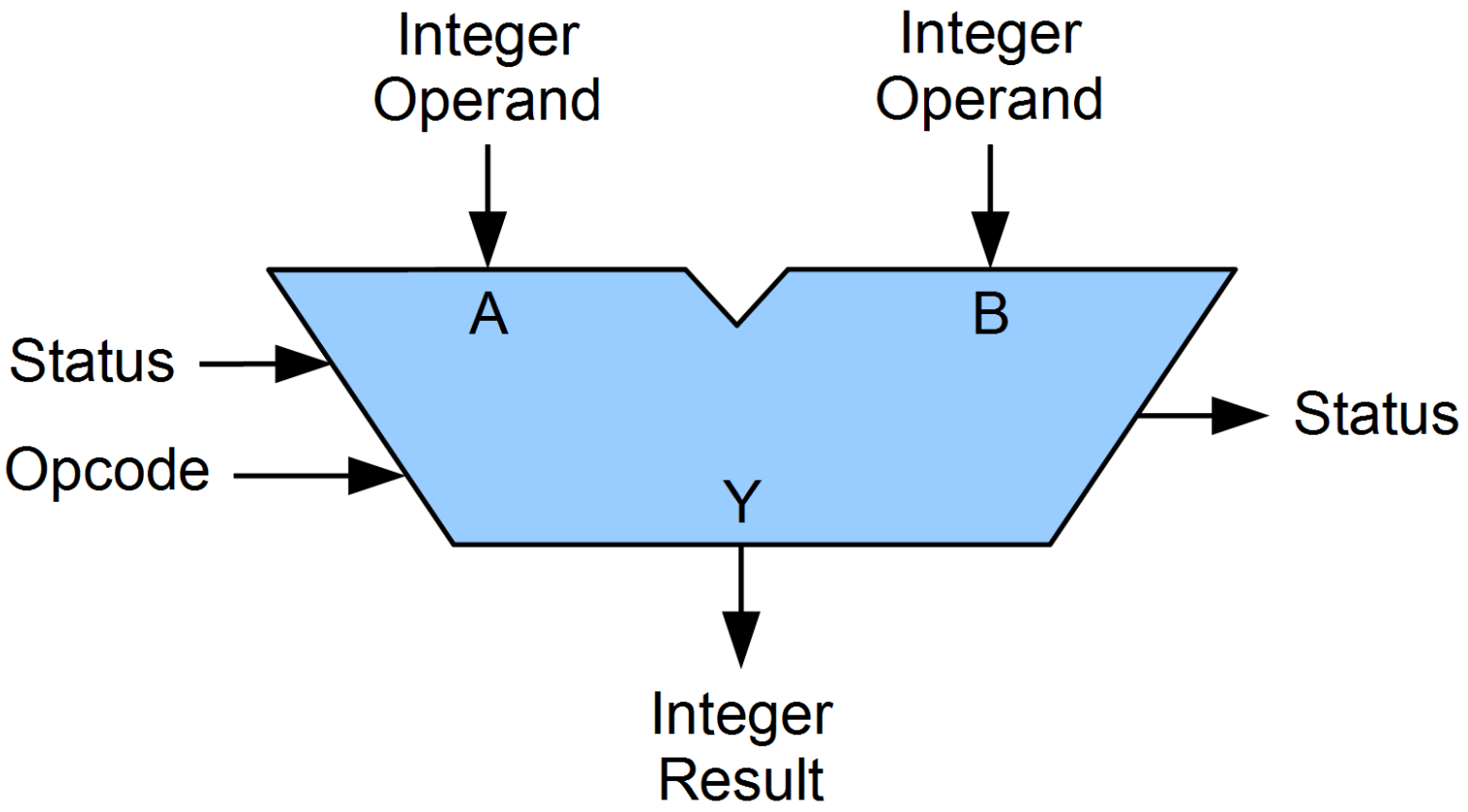
ALU semplificata a 1 bit

- Il multiplexer di destra attraverso l'ingresso di selezione *op* (operazione) presenta in uscita una di queste alternative:
  - (0) il risultato del calcolo di FA,
  - (1) il risultato dell'AND,
  - (2) il risultato dell'OR.
- Per quanto riguarda il FA, l'ingresso di controllo *cb* (complementa B) determina cosa viene sommato:
  - (0) somma $A_i + B_i + R_{i-1}$;
  - (1) somma $A_i + \overline{B_i}$ (sottrazione tramite complemento).
- In sostanza l'ALU nella slide precedente è in grado sommare o sottrarre due bit, farne l'AND o l'OR.

ALU n bit



ALU


Integer
Operand
Integer
Operand
A
B
Status
Opcode
Y
Status
Integer
Result

Domande?

Riferimenti principali

- Appendice B di ***Calcolatori elettronici. Architettura e Organizzazione,*** Giacomo Bucci. McGraw-Hill Education, 2017. http://highered.mheducation.com/sites/dl/free/8838675465/1098336/AppB.pdf (download gratuito)
- ***Slide Architettura degli Elaboratori AA 2019/2020*** corso di laura triennale in Informatica dell'Università di Ferrara, Prof. Davide Bertozzi.