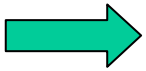
Calcolatori Elettronici A
a.a. 2008/2009

ALU:
UNITA' LOGICO-ARITMETICA

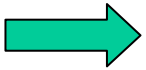
Massimiliano Giacomin

ALU

- Arithmetic Logic Unit – Unità logico aritmetica
- Effettua le operazioni logiche e aritmetiche all'interno della CPU
- Progettiamo una ALU tenendo conto delle esigenze del MIPS



ALU a 32 bit (costruite a partire da ALU a 1 bit)



Operazioni aritmetiche:
- somma
- sottrazione
- set on less than (slt)

Operazioni logiche:
- AND
- OR
- NOR

Operazione di somma

- Richiamo sui numeri in complemento a 2 con *n* bit:
  - positivi rappresentati normalmente, negativi in C.A.2 (si ottiene dal C.A.1 sommando 1)
  - estensione da *n* a *n+k* bit: estensione del segno
  - somma: normalmente scartando l'eventuale riporto, overflow se *pos+pos=neg* o *neg+neg=pos*
- Vogliamo sommare bit a bit due numeri binari di *n* bit:
  $\boldsymbol{a_{n-1}\ldots a_i\ldots a_0}$ e $\boldsymbol{b_{n-1}\ldots b_i\ldots b_0}$

$a_3\,a_2\,a_1\,a_0$

1 0 1 1 +

$b_3\,b_2\,b_1\,b_0$

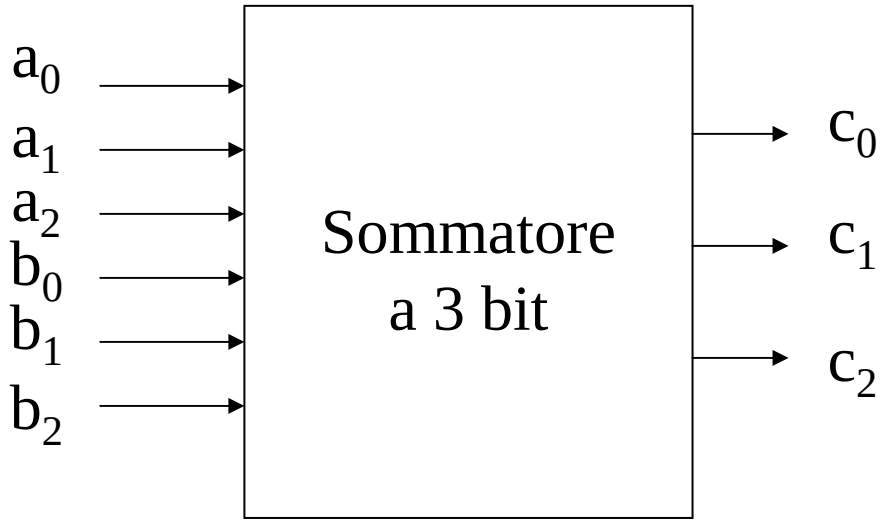
0 1 1 1 =

$c_3\,c_2\,c_1\,c_0$

(1) 0 0 1 0

Una soluzione ad hoc

- Costruire un circuito digitale specializzato ad eseguire la somma di *n* bit
- Ad esempio, con *n* = 3

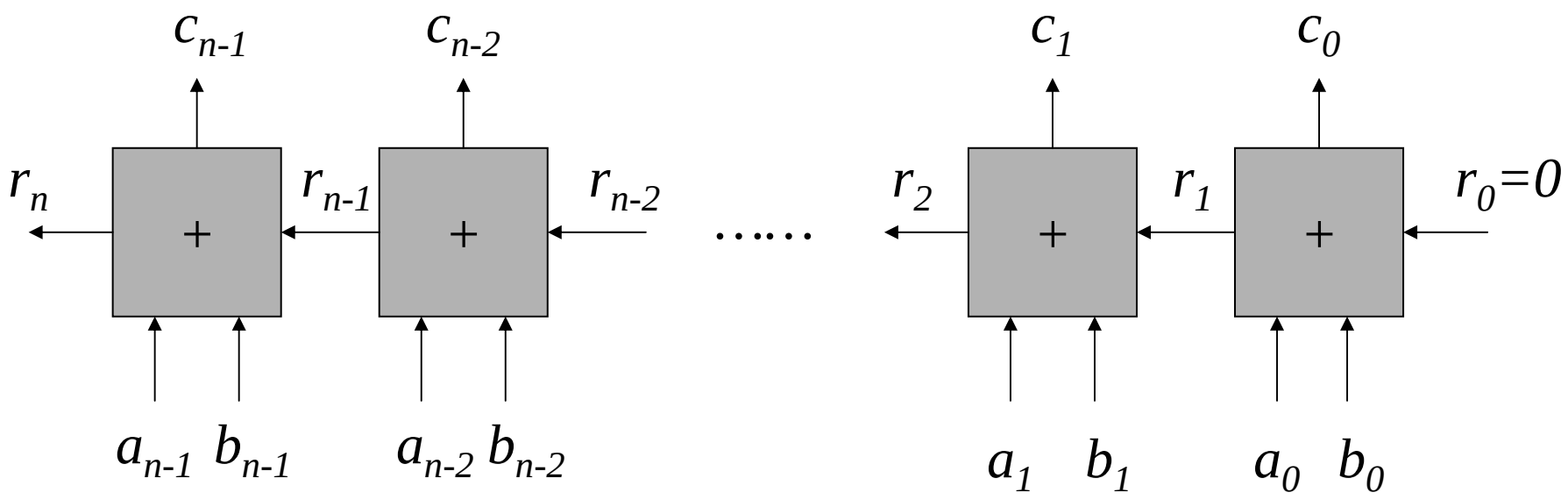


- Problema: se si cambia *n* occorre costruire un altro circuito

Una soluzione basata sulla tecnica di decomposizione

- **Decomposizione di funzioni** in funzioni più semplici:
  - si ottengono reti combinatorie più semplici
  - si introducono nuove variabili booleane chiamate **“variabili di appoggio”** che permettono di connetterle tra loro
- Nel ns. caso:
  - decomposizione in reti combinatorie ciascuna delle quali serve per ottenere c_i (da a_i e b_i)
  - variabili di appoggio = variabili di riporto $\boldsymbol{r_i}$

Sommatore a *n* bit con n “full-adder”



Per ogni *i* da *0* a *n-1* il full-adder:

- somma $\boldsymbol{a_i}$ e $\boldsymbol{b_i}$ e il bit di riporto $\boldsymbol{r_i}$ dalla posizione precedente
- Genera in tal modo il bit $\boldsymbol{c_i}$ di somma e il bit $\boldsymbol{r_{i+1}}$ di riporto per le cifre successive

Sommatore di tipo “full adder”

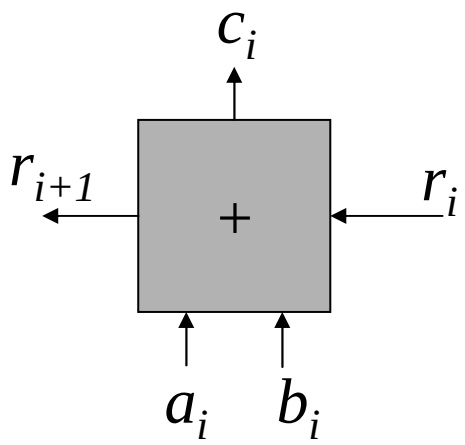


Tabella di verità

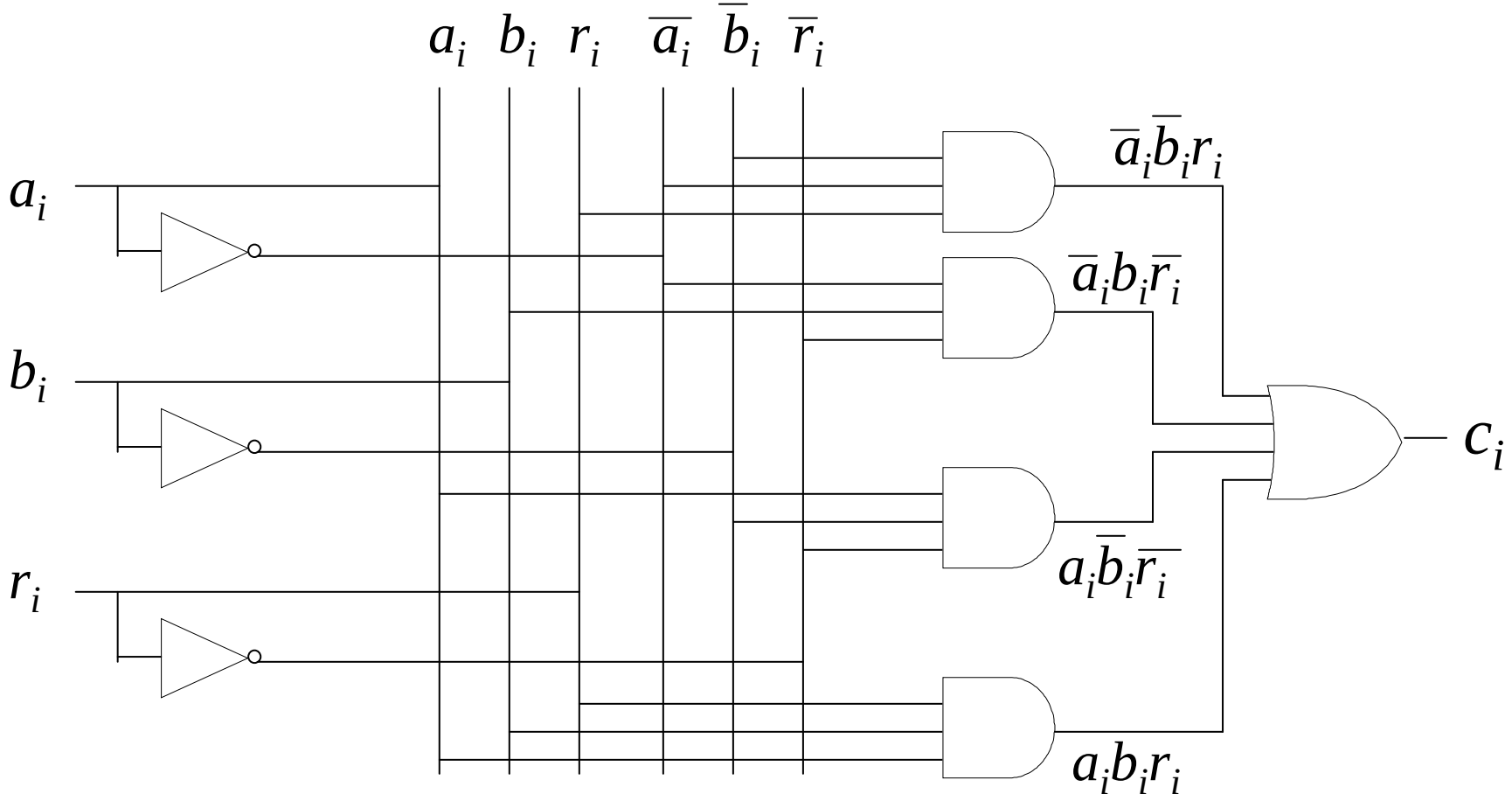
a_i	b_i	r_i	c_i	r_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$\boldsymbol{c_i} = \bar{a}_i\bar{b}_i r_i + \bar{a}_i b_i\bar{r}_i + a_i\bar{b}_i\bar{r}_i + a_i b_i r_i$$

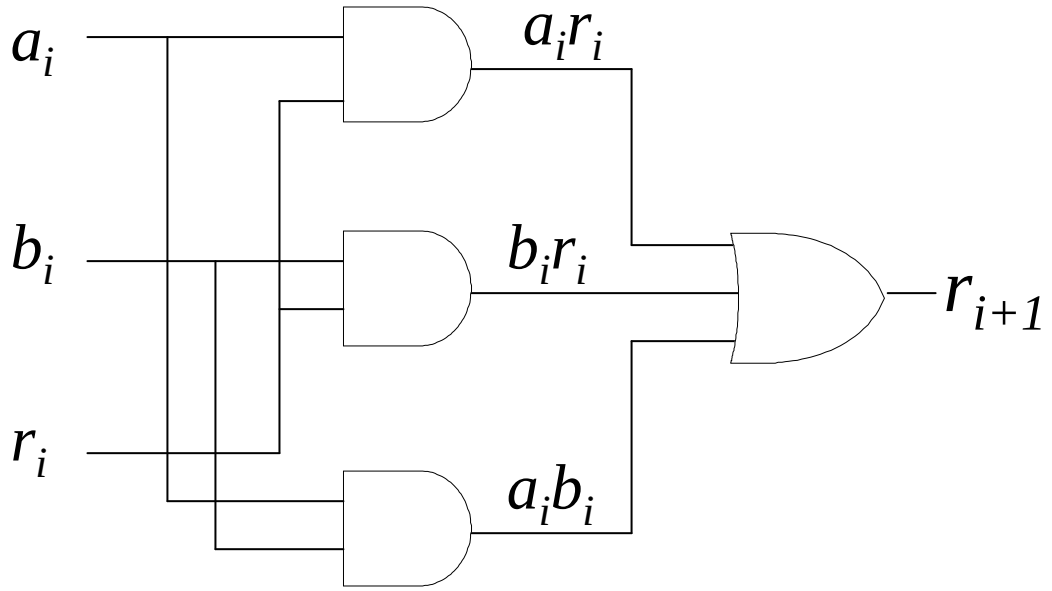
$$\boldsymbol{r_{i+1}} = \bar{a}_i b_i r_i + a_i\bar{b}_i r_i + a_i b_i\bar{r}_i + a_i b_i r_i =$$
$$\bar{a}_i b_i r_i + a_i\bar{b}_i r_i + a_i b_i\bar{r}_i + a_i b_i r_i + \boldsymbol{a_i b_i r_i}$$
$$+ \boldsymbol{a_i b_i r_i} =$$

$$= b_i r_i(\bar{a}_i + a_i) + a_i r_i(\bar{b}_i + b_i) + a_i b_i \cdot$$
$$(\bar{r}_i + r_i) = \boldsymbol{b_i r_i + a_i r_i + a_i b_i}$$

Realizzazione circuitale: generazione di c_i

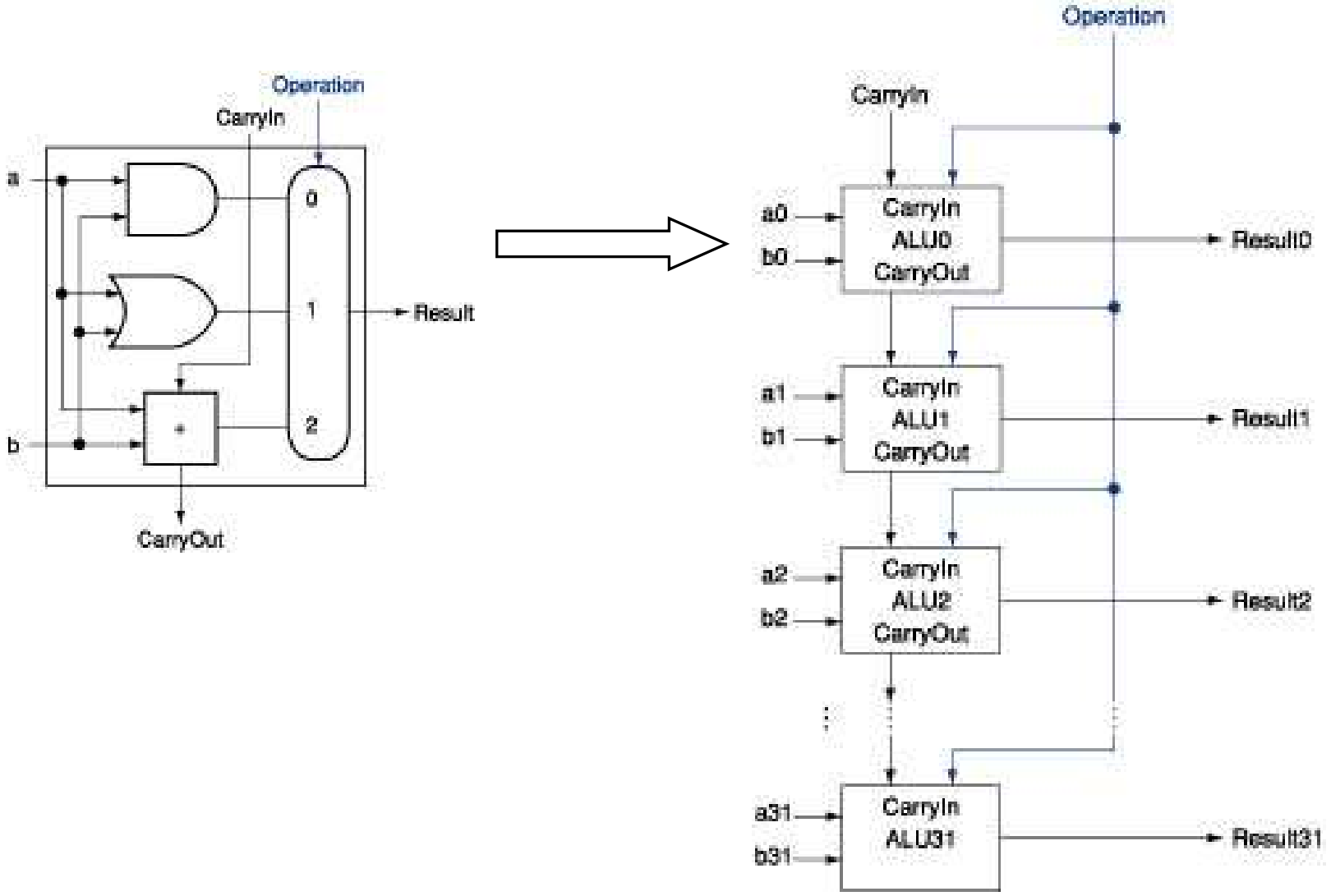


Realizzazione circuitale: generazione di r_{i+1}



$$r_{i+1} = b_i r_i + a_i r_i + a_i b_i$$

Aggiunta delle operazioni logiche di AND e OR

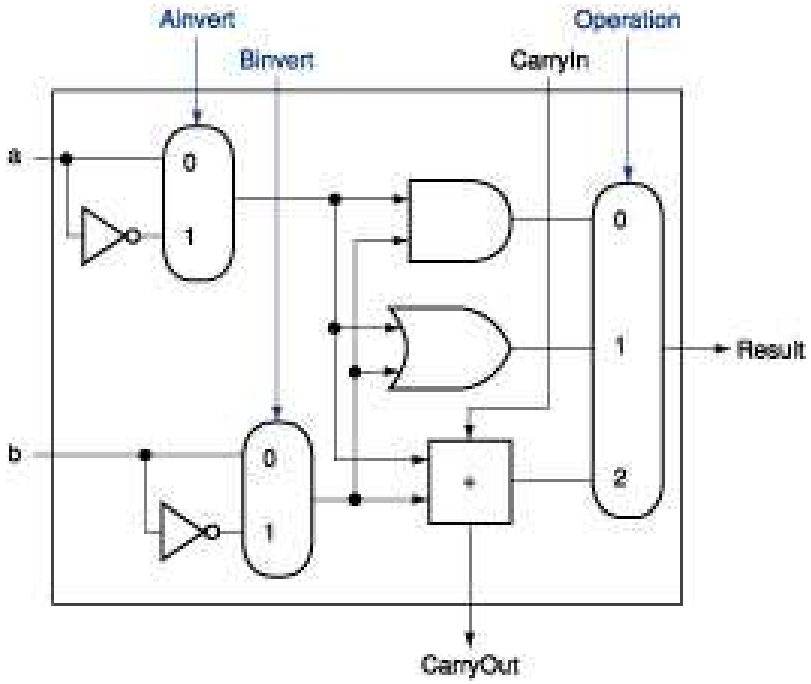


Aggiunta di NOR e sottrazione

- NOR: $\overline{a+b} = \bar{a} \cdot \bar{b}$
- sottrazione: a-b = a + (-b)

$\bar{b} + 1$

Posso sfruttare r_0!



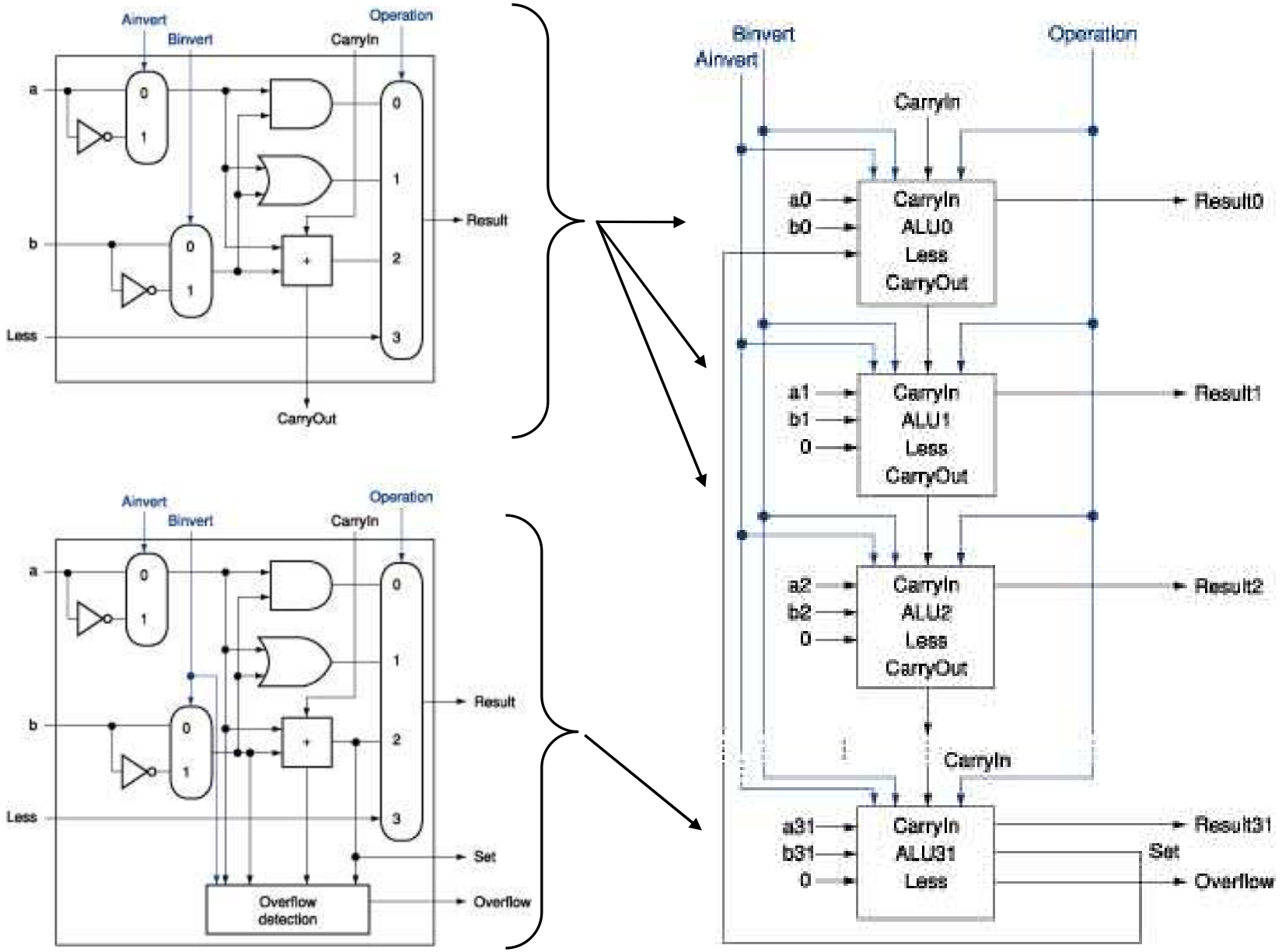
NB: per fare a-b seleziono:

- Binvert
- Operation=2

e pongo $CarryIn_0=1$

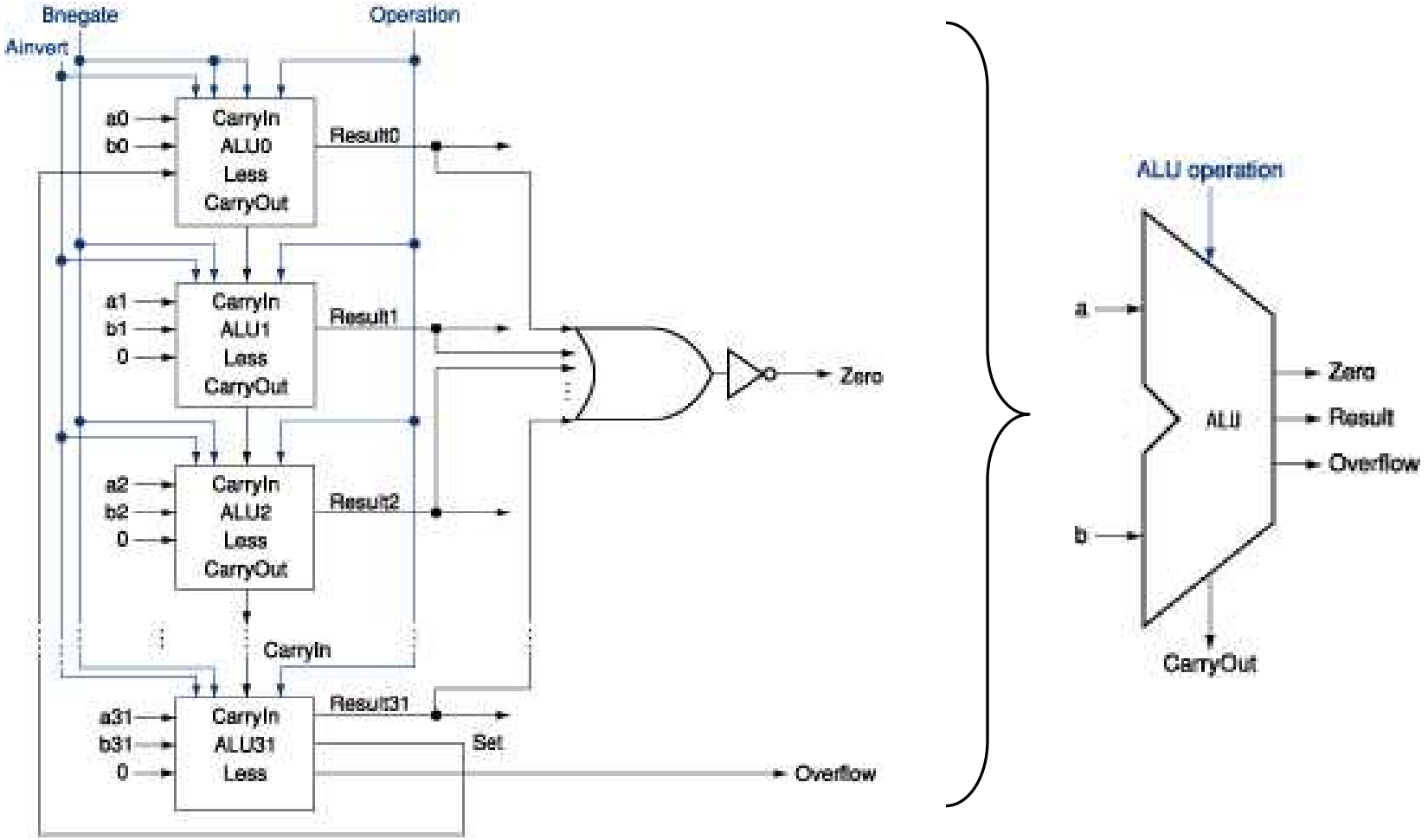
Aggiunta di slt e overflow detection

a<b sse a-b<0



Aggiunta del bit di zero

- Utile per il test di eguaglianza (cfr. *beq*): a=b sse a-b=0
- Nello schema Bnegate e CarryIn sono combinati
  (ogni volta che nego B, se uso il sommatore è per sommare –B)



NB: la ALU ottenuta è controllata secondo la tabella seguente

AInvert	Bnegate	ALUOp	Function
0	0	00	AND
0	0	01	OR
0	0	10	add
0	1	10	subtract
0	1	11	slt
1	1	00	NOR