Sintesi di circuiti notevoli: addizionatore, comparatore, sottrattore e complementatore

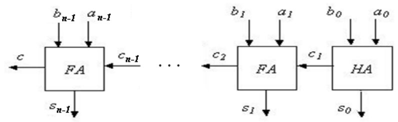
Prof. Daniele Gorla

Sommatore Parallelo a *n* bit

Specifica: un sommatore binario realizza la somma aritmetica fra due stringhe di *n* bit A = $a_{n-1}\ldots a_0$ e B = $b_{n-1}\ldots b_0$, viste come numeri naturali.

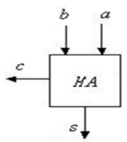
Idea: effettuare la somma come siamo abituati

- somma i bit meno significativi a_0 e b_0;
- questo genera il bit meno significativo del risultato s_0 ed un eventuale riporto c_1;
- procedi a sommare a_1, b_1 e c_1; questo genera s_1 e c_2;
- ...e così via fino ai bit più significativi;
- se l'ultima somma genera un riporto *c*, allora c'è overflow.

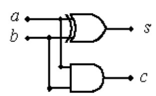


Sintesi della cella elementare HA

La somma fra a_0 e b_0 (qui per semplicità chiamati solo *a* e *b*) non è influenzata da alcun riporto precedente (è la prima somma della sequenza) e può generare un riporto *c*:



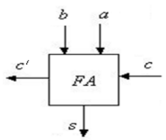
b a	s c
0 0	0 0
0 1	1 0
1 0	1 0
1 1	0 1



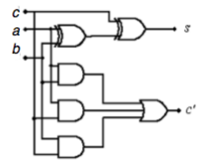
$$s = a \oplus b$$
$$c = ab$$

Sintesi della cella elementare FA

Indicando con *c* il riporto (*carry*) della somma fra a_{i-1} e b_{i-1} e con *c'* quello della somma fra a_i e b_i, avremo la seguente tabella di verità per il modulo che esegue la somma fra a_i e b_i (qui per semplicità chiamati solo *a* e *b*):



c b a	s c'
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



$$s = (a \oplus b)\overline{c} + \overline{(a \oplus b)}c = (a \oplus b) \oplus c$$
$$c' = ac + bc + ab$$

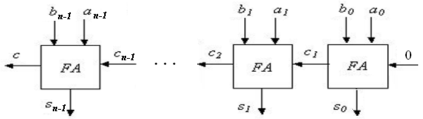
Sommatore uniforme

SAPIENZA
Università di Roma
Dipartimento di Informatica

Avere due circuiti (HA e FA) rende il progetto più complesso.

Per semplificare, si può avere una versione "uniforme" del sommatore in cui si usa solo FA: basta impostare il riporto entrante nella prima cella addizionatrice (quella per i bit meno significativi) a 0.

N.B.: ho qualche porta in più, ma devo produrre un solo tipo di circuito!!



Addizionatore per numeri interi

SAPIENZA
Università di Roma
Dipartimento di Informatica

Come abbiamo visto, per interi rappresentati in Ca2 la somma si effettua esattamente nello stesso modo; quindi, il circuito è lo stesso!

L'unica cosa che cambia è la condizione di overflow:

- Per numeri naturali, basta vedere l'ultimo bit di riporto (1 → overflow)
- Per numeri interi, sia ha overflow se
  - gli operandi hanno lo stesso segno e il risultato ha segno diverso
  - si ottiene la sequenza "non ammessa" 10…0

Quindi, l'EB associata all'overflow per interi è

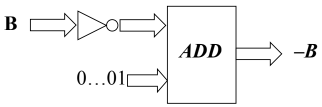
$$a_{n-1}b_{n-1}\overline{s}_{n-1} + \overline{a}_{n-1}\overline{b}_{n-1}s_{n-1} + s_{n-1}\overline{s}_{n-2}\ldots\overline{s}_0$$

Opposto e Sottrazione

SAPIENZA
Università di Roma
Dipartimento di Informatica

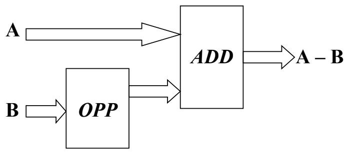
Opposto

Ricordando che l'opposto di $\mathbf{B} = b_{n-1}\ldots b_0$ è $\overline{b}_{n-1}\ldots\overline{b}_0 + 1$, abbiamo che il circuito per calcolare l'opposto è:



Sottrazione

Per fare **A – B**, basta fare **A** + (–**B**) e quindi il circuito per la sottrazione è:



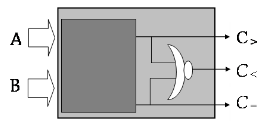
Comparatore aritmetico

SAPIENZA
Università di Roma
Dipartimento di Informatica

Problema: date due stringhe binarie di n bit **A** e **B** rappresentanti due numeri naturali, dire se **A > B, A = B** o **A < B**.
Vogliamo un circuito del tipo

tale che :
- $c_> = 1$ sse **A** > **B**
- $c_< = 1$ sse **A** < **B**
- $c_= = 1$ sse **A** = **B**

OSS.: $c_< = \text{NOR}(c_>, c_=)$; pertanto basterà trovare i circuiti per calcolare $c_>$ e $c_=$, da cui

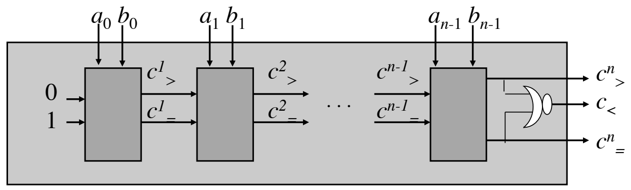


Struttura del comparatore

L'idea è quella di costruire un circuito incrementale, cioè un circuito formato da *n* celle circuitali elementari di confronto messe in cascata.
Per fare ciò, usiamo dei risultati parziali così definiti:

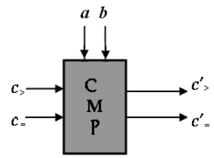
per ogni $i = 1, \dots, n$

- $c^i_> = 1$ sse $a_{i-1} \dots a_0 > b_{i-1} \dots b_0$
- $c^i_= = 1$ sse $a_{i-1} \dots a_0 = b_{i-1} \dots b_0$



Cella elementare del comparatore

SAPIENZA
Università di Roma
Dipartimento di Informatica



La tabella di verità per **CMP** è

a	b	$c_>$	$c_=$	$c'_>$	$c'_=$
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	-	-
0	1	0	0	0	0
0	1	0	1	0	0
0	1	1	0	0	0
0	1	1	1	-	-
1	0	0	0	1	0
1	0	0	1	1	0
1	0	1	0	1	0
1	0	1	1	-	-
1	1	0	0	0	0
1	1	0	1	0	1
1	1	1	0	1	0
1	1	1	1	-	-

Sintesi di $c'_>$

SAPIENZA
Università di Roma
Dipartimento di Informatica

La MdK è

$c_> c_=$ \ $a\ b$	00	01	11	10
00	0	0	0	1
01	0	0	0	1
11	-	-	-	-
10	1	0	1	1

da cui $c'_> = a\bar{b} + ac_> + \bar{b}c_> = a\bar{b} + (a + \bar{b})c_>$

Sintesi di $c'_=$

SAPIENZA
Università di Roma
Dipartimento di Informatica

La MdK è

$c_> c_=$ \ $a\ b$	00	01	11	10
00	0	0	0	0
01	1	0	1	0
11	-	-	-	-
10	0	0	0	0

da cui $c'_= = \bar{a}\bar{b}c_= + abc_= = (\bar{a}\bar{b} + ab)c_= = (\overline{a \oplus b})c_=$

Schema circuitale