Implementazione di una Arithmetic-Logic-Unit

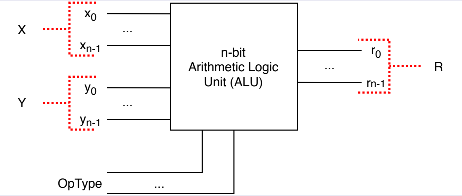
Corrado Santoro

Dipartimento di Matematica e Informatica
santoro@dmi.unict.it

Corso di Architettura degli Elaboratori

Arithmetic-Logic-Unit

- Una **ALU** a *n* bit è un circuito logico in grado di effettuare operazioni matematiche (somme, sottrazioni) e logiche (bitwise-AND, bitwise-OR, bitwise-NOT) su due operandi a *n* bit
- E' un circuito presente in qualunque CPU ed è rappresentato nel seguente modo



- *n* ingressi per l'operando *X*
- *n* ingressi per l'operando *Y*
- *k* ingressi per specificare il tipo di operazione da effettuare *OpType*
- *n* uscite per il risultato *R*

Implementazione della Somma Binaria a 4 Bit tramite porte logiche

Operazione di somma binaria

Somma in base 2

L'operazione di somma binaria, in colonna, può essere svolta applicando le seguenti semplici regole:

$$\begin{array}{ccccl} 0 & + & 0 & = & 0 \\ 0 & + & 1 & = & 1 \\ 1 & + & 0 & = & 1 \\ 1 & + & 1 & = & 0 \text{ con riporto di } 1 \end{array}$$

Esempio

Riporti	1	1	1		
Primo addendo	0	0	1	1	+
Secondo addendo	0	1	0	1	=
Risultato	1	0	0	0	

$$(0011)_2 + (0101)_2 = (1000)_2$$
$$3 + 5 = 8$$

Operazione di somma binaria

Esempio

Riporti	1	1	1		
Primo addendo	0	0	1	1	+
Secondo addendo	0	1	0	1	=
Risultato	1	0	0	0	

$$(0011)_2 + (0101)_2 = (1000)_2$$
$$3 + 5 = 8$$

Somma in base 2

- Per la *prima colonna*, sommiamo **due bit** e produciamo un **risultato** e un **riporto**
- Per le *successive colonna*, sommiamo **due bit** e il **riporto precedente**, e produciamo un **risultato** e un **nuovo riporto**

Nomenclatura

Riporti C	1	1	1			c_3	c_2	c_1	
Primo addendo X	0	0	1	1	+	x_3	x_2	x_1	x_0
Secondo addendo Y	0	1	0	1	=	y_3	y_2	y_1	y_0
Risultato S	1	0	0	0		s_3	s_2	s_1	s_0

Somma in base 2

- Utilizziamo una variabile booleana per ogni bit coinvolto nella nostra operazione
- Rappresentiamo i 4 bit del primo addendo X con le variabili booleane x_3, x_2, x_1, x_0
- Rappresentiamo i 4 bit del secondo addendo Y con le variabili booleane y_3, y_2, y_1, y_0
- Rappresentiamo i 4 bit della somma S con le variabili booleane s_3, s_2, s_1, s_0
- Rappresentiamo i 3 bit dei riporti con le variabili booleane c_3, c_2, c_1

Operazione di somma binaria

Implementazione

Riporti C	1	1	1			c_3	c_2	c_1	
Primo addendo X	0	0	1	1	+	x_3	x_2	x_1	x_0
Secondo addendo Y	0	1	0	1	=	y_3	y_2	y_1	y_0
Risultato S	1	0	0	0		s_3	s_2	s_1	s_0

Implementazione

- Utilizziamo una **rete logica** per ogni **colonna della somma** $\Rightarrow$ **4 reti logiche**
- La prima colonna (colonna "0")sarà una rete con ingressi x_0, y_0 e uscite s_0, c_1
- La altre colonne ($n, n > 0$) saranno delle reti con ingressi x_n, y_n, c_n e uscite s_n, c_{n+1}

Implementazione

Riporti C	1	1	1			c_3	c_2	c_1	
Primo addendo X	0	0	1	1	+	x_3	x_2	x_1	x_0
Secondo addendo Y	0	1	0	1	=	y_3	y_2	y_1	y_0
Risultato S	1	0	0	0		s_3	s_2	s_1	s_0

Somma Colonna 0

x_0	y_0	s_0
0	0	0
0	1	1
1	0	1
1	1	0

$$s_0 = \overline{x_0}y_0 + x_0\overline{y_0}$$

Riporto Colonna 0

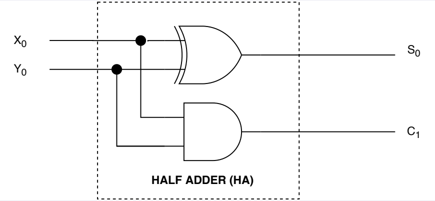
x_0	y_0	c_1
0	0	0
0	1	0
1	0	0
1	1	1

$$c_1 = x_0 y_0$$

Half Adder

$$s_0 = \overline{x_0} y_0 + x_0 \overline{y_0}$$

$$c_1 = x_0 y_0$$



Operazione di somma binaria

Implementazione

Riporti C	1	1	1			c_3	c_2	c_1	
Primo addendo X	0	0	1	1	+	x_3	x_2	x_1	x_0
Secondo addendo Y	0	1	0	1	=	y_3	y_2	y_1	y_0
Risultato S	1	0	0	0		s_3	s_2	s_1	s_0

Somma Colonna $n > 0$

c_n	x_n	y_n	s_n
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

$$s_n = \overline{c_n}\,\overline{x_n}\,y_n + \overline{c_n}\,x_n\,\overline{y_n} + c_n\,\overline{x_n}\,\overline{y_n} + c_n\,x_n\,y_n = c_n\ XOR\ x_n\ XOR\ y_n$$

Operazione di somma binaria

Implementazione

Riporti C	1	1	1			c_3	c_2	c_1	
Primo addendo X	0	0	1	1	+	x_3	x_2	x_1	x_0
Secondo addendo Y	0	1	0	1	=	y_3	y_2	y_1	y_0
Risultato S	1	0	0	0		s_3	s_2	s_1	s_0

Somma Colonna $n > 0$

c_n	x_n	y_n	c_n XOR x_n	$s_n = (c_n \ XOR \ x_n) \ XOR \ y_n$
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
0	1	1	1	0
1	0	0	1	1
1	0	1	1	0
1	1	0	0	0
1	1	1	0	1

$$s_n = \overline{c_n}\,\overline{x_n}\,y_n + \overline{c_n}\,x_n\,\overline{y_n} + c_n\,\overline{x_n}\,\overline{y_n} + c_n\,x_n\,y_n = c_n \ XOR \ x_n \ XOR \ y_n$$

Operazione di somma binaria

Implementazione

Riporti C	1	1	1			c_3	c_2	c_1	
Primo addendo X	0	0	1	1	+	x_3	x_2	x_1	x_0
Secondo addendo Y	0	1	0	1	=	y_3	y_2	y_1	y_0
Risultato S	1	0	0	0		s_3	s_2	s_1	s_0

Riporto Colonna $n > 0$

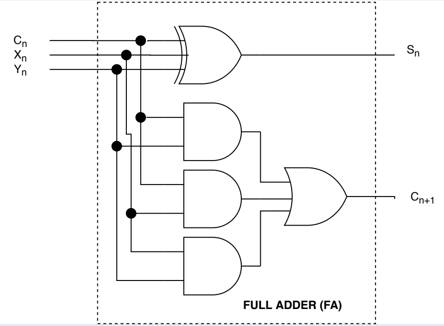
c_n	x_n	y_n	c_{n+1}
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$$c_{n+1} = x_n y_n + y_n c_n + x_n c_n$$

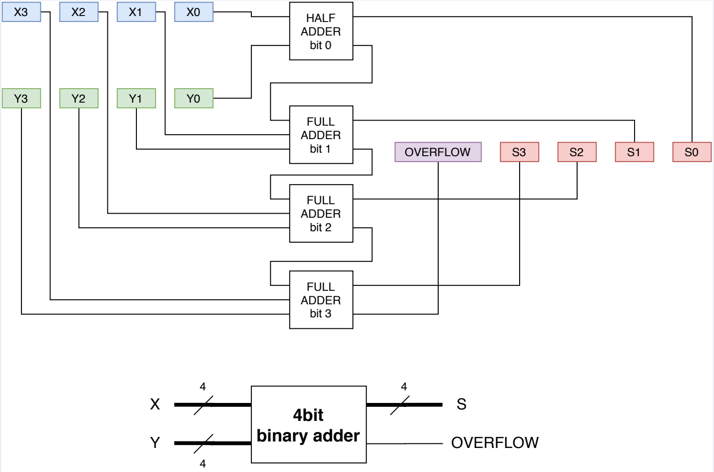
Full Adder

$$s_n = \overline{c_n}\,\overline{x_n}\,y_n + \overline{c_n}\,x_n\,\overline{y_n} + c_n\,\overline{x_n}\,\overline{y_n} + c_n\,x_n\,y_n = x_n\ XOR\ y_n\ XOR\ c_n$$

$$c_{n+1} = x_n y_n + y_n c_n + x_n c_n$$



4 Bit Adder


X3
X2
X1
X0
HALF ADDER bit 0
Y3
Y2
Y1
Y0
FULL ADDER bit 1
OVERFLOW
S3
S2
S1
S0
FULL ADDER bit 2
FULL ADDER bit 3
X
Y
4
4
4
4bit binary adder
S
OVERFLOW

Implementazione della Sottrazione Binaria a 4 Bit tramite porte logiche

Operazione di sottrazione binaria

Sottrazione in base 2

L'operazione di **sottrazione** $X - Y$ (con X e Y due variabili a n bit) può essere trasformata in una **somma** sfruttando il complemento a 2:

$$X - Y = X + (-Y) \quad = \quad X + (\overline{Y} + 1)$$

dove $\overline{Y}$ è la negazione "bitwise" (cioè bit per bit) i Y.

- L'operazione $\overline{Y}$ può essere facilmente implementata usando n porte NOT
- Occorrerebbe quindi un ulteriore circuito sommatore per poter effettuare l'operazione $... + 1$, tuttavia ....

Verso la sottrazione in base 2

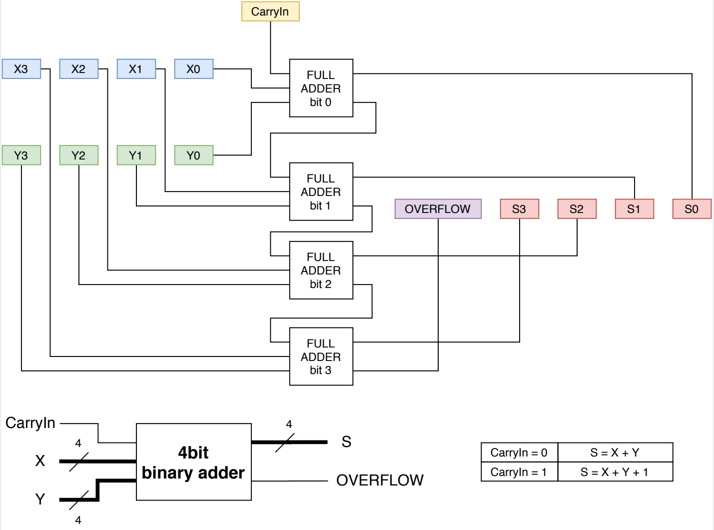
- Supponiamo di voler implementare la seguente operazione:

$$X + Y + 1$$

- Possiamo pensare al .. $+1$ come alla presenza di un “riporto” già nella prima colonna (bit 0) della nostra operazione
- Allora usiamo un “Full Adder” anche sul primo bit, e sfruttiamo il suo ingresso di carry (CarryIn) per ottenere i seguenti casi:

$$\begin{array}{rl} X+Y & \quad se \;\; CarryIn = 0 \\ X+Y+1 & \quad se \;\; CarryIn = 1 \end{array}$$

4 Bit Adder con CarryIn



CarryIn = 0	S = X + Y
CarryIn = 1	S = X + Y + 1

Verso la sottrazione in base 2

- A questo punto, se riuscissimo a sfruttare il "CarryIn" per pilotare anche un ulteriore circuito in grado di invertire i bit di Y otterremmo il seguente effetto:

$$\begin{array}{rl} X + Y & \quad se \;\; CarryIn = 0 \\ X + \overline{Y} + 1 & \quad se \;\; CarryIn = 1 \end{array}$$

- Chiamando l'ingresso "CarryIn" come **OpType** potremmo raggiungere il nostro obiettivo:

$$\begin{array}{rl} X + Y & \quad se \;\; OpType = 0 \\ X + \overline{Y} + 1 & \quad se \;\; OpType = 1 \end{array}$$

- Cioe un circuito in grado di comportarsi come **sommatore** se OpType = 0, e come **sottrattore** se OpType = 1

Verso la sottrazione in base 2

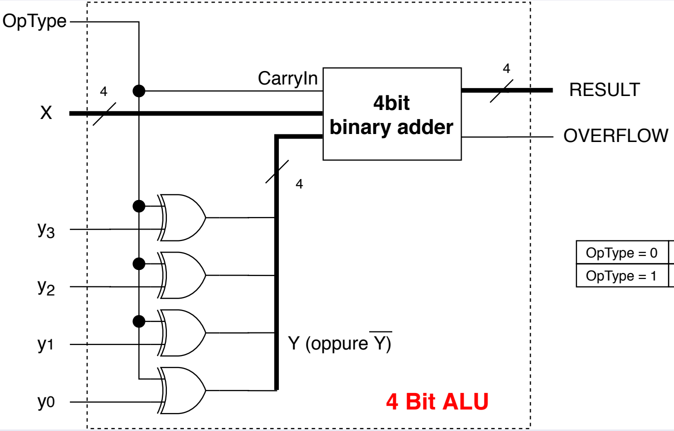
- Sintetizziamo un “invertitore su comando”:

y_i	$OpType$	y'_i
0	0	0
0	1	1
1	0	1
1	1	0

- Otteniamo $y'_i = y_i \;\; xor \;\; OpType$

ALU a 4 Bit

- Mettiamo “tutto insieme” in un unico circuito:



OpType = 0	RESULT = X + Y
OpType = 1	RESULT = X - Y

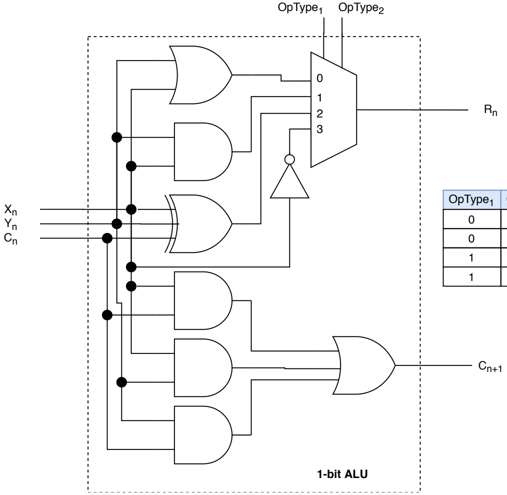
Operazioni Logiche Bitwise

Inclusione di Operazioni Logiche

- Estendiamo la nostra ALU includendo anche la possibilià di effettuare le seguenti **operazioni logiche bitwise**:

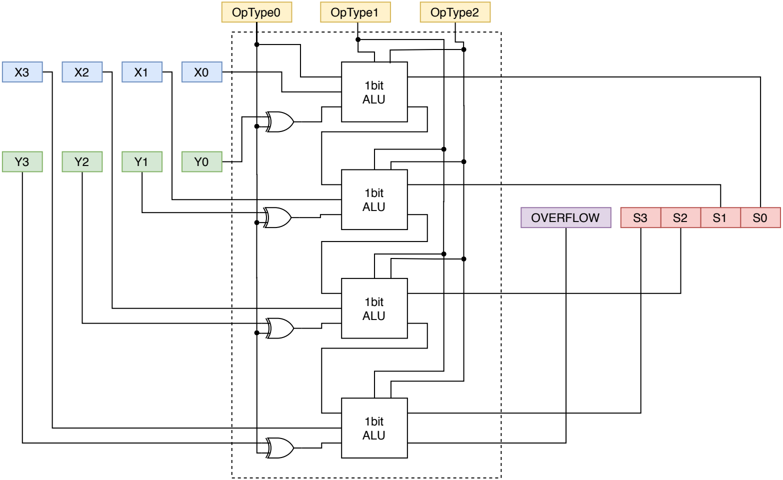
$$\begin{array}{ccc} X & AND & Y \\ X & OR & Y \\ & NOT(X) & \end{array}$$

- A tale scopo, possiamo modificare il **Full Adder** includendo la possibilità di fare le operazioni citate sul singolo bit
- Aggiungiamo dunque un **multiplexer** che consente di mandare in output, alternativamente, l'uscita del:
  - Sommatore, $x_i + y_i$
  - Porta AND, $x_i \;\; AND \;\; y_i$
  - Porta OR, $x_i \;\; OR \;\; y_i$
  - Negazione, $\overline{x_i}$



$OpType_1$	$OpType_2$	R_n
0	0	X_n or Y_n
0	1	X_n and Y_n
1	0	$X_n + Y_n$
1	1	not(X_n)

4-bit ALU Completa



$OpType_0$	$OpType_1$	$OpType_2$	R
0	1	0	$X + Y$
1	1	0	$X - Y$
0	0	0	$X\ AND\ Y$
0	0	1	$X\ OR\ Y$
0	1	1	$NOT(X)$

Implementazione di una Arithmetic-Logic-Unit

Corrado Santoro

Dipartimento di Matematica e Informatica
santoro@dmi.unict.it

Corso di Architettura degli Elaboratori