

## HALF ADDER / FULL ADDER

Lunedì 24 ottobre 2022 23:23

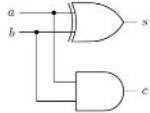
### HALF ADDER

- Il circuito più semplice per effettuare una somma di operandi ad un solo bit deve calcolare il valore della somma e il valore del riporto:

$$s = a \oplus b \quad c = a \cdot b$$

↓  
XOR

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



### OPERATORE XOR: $\oplus$

$$a \oplus b = \bar{a}b + a\bar{b}$$

x <sub>1</sub>	x <sub>2</sub>	y
0	0	0
0	1	1
1	0	1
1	1	0

QUANDO EFFETTUO LA SOMMA DOBBIAMO CALCOLARE DUE ELEMENTI:

- SOMMA
- RIPORTO (BIT PROPAGATO AI MODULI SUCCESSIVI NELLA CATENA ITERATIVA)

LA SOMMA È SEMPLICE:  $S = a \oplus b$ , È UN OPERATORE XOR.

AVRÒ UN RIPORTO SOLO SE STO SOMMANDO DUE BIT CHE VALGONO 1:

$$\begin{array}{r} 1 \\ 1 \\ \hline 0 \end{array}$$

IL RIPORTO È UN OPERATORE AND:  $C = a \cdot b$

MA LO POSSO UTILIZZARE COME MODULO DI UNA CATENA ITERATIVA? NO.

DOBBIAMO MODIFICARE LA FUNZIONE DI SOMMA PER PRENDERE IN CONSIDERAZIONE IL BIT DI CARRY.

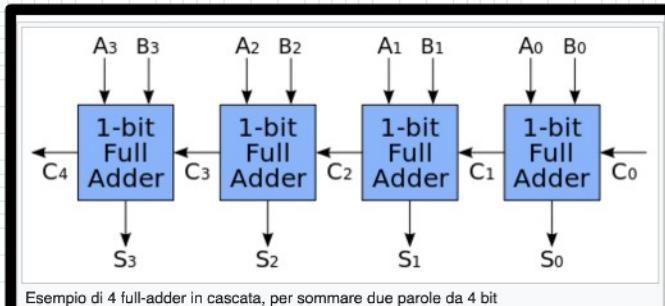


Si chiama "half adder" perché nella somma non prende in considerazione il bit di carry che viene propagato al modulo 2.

MA DOVREMO MODIFICARE ANCHE LA FUNZIONE CHE CALCOLA IL BIT DI RIPORTO PER DECIDERE SE, IL BIT DI CARRY CHE ARRIVA dal modulo precedente, sommato agli operandi A e B del modulo attuale, verrà propagato o meno.

Serve il:

### FULL ADDER



qui prendiamo in considerazione il riporto che viene propagato.

QUELLO CHE ANDIAMO A FARE È :

qui prendiamo in considerazione il rapporto che viene proposto.

QUELLO CHE ANDIAMO A FARE È :

RISCRIVERE LE NOSTRE FUNZIONI DI SOMMA E DI CARRY COME FUNZIONI DI 3 VARIABILI.

23 - 08 008

UN MODULUS HA COME INPUT LE MIE DUE VARIABILI E IL CARRY CHE PROVIENE DAL BIT PRECEDENTE.

$$S_i = f_1(a_i, b_i, c_{i-1})$$

FUNZIONE SOMMA;

$$C = f_2(a_i, b_i, c_{i-1})$$

FUNZIONE CARRY;

PRENDO QUESTE DUE EQUAZIONI E SCRIVO LA TABELLA DI VERTA' E LA TRASFORMO IN UNA MAPPA DI KARNAUGH.

		$s_i = f_1(a_i, b_i, c_{i-1})$			
		00	01	11	10
$a_i \backslash b_i$	0	0	1	0	1
	1	1	0	1	0

$s_i = c_{i-1} \oplus a_i \oplus b_i$

		$c = f_2(a_i, b_i, c_{i-1})$			
		00	01	11	10
$a_i \backslash b_i$	0	0	0	1	0
	1	0	1	1	1

$c = a_i b_i + c_{i-1} (a_i \oplus b_i)$

SE I MIEI DUE OPERANDI SONO 0,  
E IL MIO CARRY PRECEDENTE VALE  
1, LA SOMMA È 1.  
VICEVERSA SE IL BIT DI CARRY È  
ZERO E UNO DEI DUE OPERANDI È 0  
E L'ALTRÒ 1, LA SOMMA È 1.  
SE IL BIT DI CARRY È UGUALE A 1, OTTRO  
1 SOLAMENTE SE I MIEI DUE OPERANDI  
SONO uguali.  
e' comune; mi permetterà  
di creare per circuito delle  
componenti che già so  
andare ad utilizzare.  
STANNO NELLO STESSO  
MODULO.

NON POSSIAMO EFFETTUARE  
NESSUNA SEMPLIFICAZIONE E  
RISCRIVO I 4 MINTERMINI:

$$S_i = C_{i-1} \oplus a_i \oplus b_i$$

•OPERATORE XOR:  $\oplus$

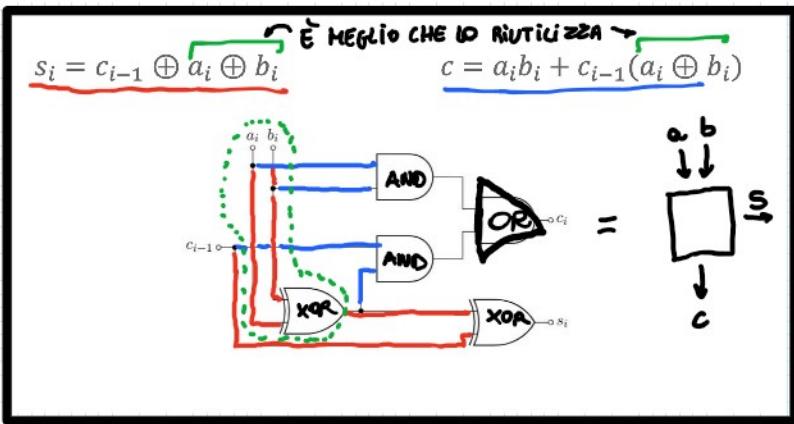
$$a \oplus b = \bar{a}b + a\bar{b}$$

$x_1$	$x_2$	$y$
0	0	0
0	1	1
1	0	1
1	1	0

$$a_i = 0, b_i = 0, C_{i-1} = 1$$

$$\rightarrow 1 \oplus 0 \oplus 0 = 1 \text{ (SOMMA) } \checkmark$$

FATTE QUESTE MINIMIZZAZIONI ARRIVO AL MIO CIRCUITO:



- Somma logica :  $(+)$  =  $(\vee)$  = OR ;
- Prodotto Logico :  $(\cdot)$  =  $(\wedge)$  = AND ;

**PROBLEMA:** IL CRITICAL PATH È LUNGO. SE METTO 32 DI QUESTE SCA TOLETTE IN SEQUENZA, AVRÒ 32 · 3 VOLTE DI CRITICAL PATH.

E POI LE PORTE XOR SONO SFORTUNATE, PERCHÉ SONO difficili da progettare. QUINDI È MOLTO LENTO, TUTTO PER FARE UNA BANALE SOMMA, DOVREI RALLENTARE IL FUNZIONAMENTO DEL MIO PROCESSORE SOLTANTO PER SVOLGERE DELLE BANALI SOMME. **NON VA BENE.**

COME PER IL COMPARATORE, CALCOLIAMO I NOSTRI BIT IN PARALLELLO, IN PARTICOLARE IL BIT-DI-CARRY.

QUINDI DORBIANO COSTRUIRE IL COSÌ DETTO :

### CARRY LOOKAHEAD ADDER

CIOÈ UN SOMMATORE CHE CERCA DI ANTICIPARE IL VALORE DEL CARRY CHE DEVE PROVENIRE DAI MODULI PRECEDENTI.

PRENDO LE MIE EQUAZIONI DI PRIMA E CERCO DI DECOMPOSERLE IN MODO TALE DA ESTRAERRE DELLE PARTI CHE MI POSSO CALCOLARE IN PARALLELLO;

$$S_i = C_{i-1} \oplus a_i \oplus b_i \quad C = a_i b_i + C_{i-1} (a_i \oplus b_i)$$

Chiamo  $a_i \cdot b_i = G_i$  e  $a_i \oplus b_i = P_i$   
 $\downarrow \text{And}$        $\downarrow \text{Xor}$

•  $G_i$  È IL GENERATORE DI CARRY, PERCHÉ SE ENTRAMBI I BIT  $a$  E  $b$  SONO 1 STO SOMMATORI IN UN MODULO VALGONO 1, CI SARÀ UN CARRY. (PREVEDERE SE UN MODULO GENERA UN CARRY)

•  $P_i$  È IL PROPAGATORE DI CARRY, PERCHÉ SE I MIEI DUE BIT IN XOR SONO TALI PER CUI DOVE IL VALORE  $a_i$  O  $b_i$  VALE 1, ALLORA IL MIO CARRY DOVRÀ VALE 1.

(PREVEDERE SE L'EVENTUALE CARRY CHE È IMPOSTATO A 1 CHE ENTRELLA DENTRO UN MODULO, DOVRÀ ESSERE PROPAGATO O meno).

ADESSO SOSTITUISCO :

$$Si = C_{i-1} \oplus Pi \quad e \quad C_i = Gi + C_{i-1} Pi$$

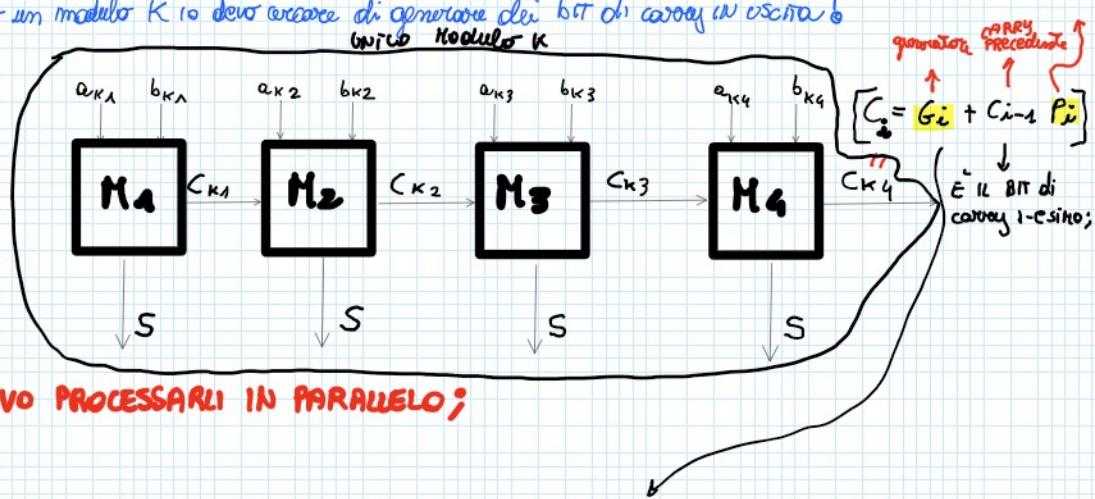
VEDIAMO COME ANTICIPARE IL CALCOLO DELLA PROPAGAZIONE DEL CARRY.

IMMAGINIAMO DI PRENDERE I NUMERI  $a$  e  $b$  che vogliamo sommare, IN GRUPPI DI 4-bit, PERCHÉ devo fare come prima, ossia se utilizzo un numero di parole come potenze di 2, posso suddividerlo.

Quindi devo avere questi moduli: somma 8 bit comunque.

SU QUESTI GRUPPI DI 4 BIT voglio calcolare IL BIT di carry, in modo tale di generarlo all'uscita.

Dato un modulare  $K$  io devo cercare di generare dei bit di carry in uscita!



DEVO PROCESSARLI IN PARALLELO;

Secondo questa equazione, ciascuno di questi moduli AVRA' UN GENERATORE ed UN PROPAGATOR.

Quindi il bit di carry del modulare  $k_4$  sarà:

$$C_{k4} = G_{k4} + P_{k4} \cdot C_{k3}$$

ORA È ITERATIVO:

$$\begin{aligned} C_{k4} &= G_{k4} + P_{k4} \cdot (C_{k3} + P_{k3} \cdot C_{k2}) \\ &= G_{k4} + P_{k4} \cdot (C_{k3} + P_{k3} \cdot (G_{k2} + P_{k2} \cdot C_{k1})) \\ &= G_{k4} + P_{k4} \cdot (C_{k3} + P_{k3} \cdot (G_{k2} + P_{k2} \cdot (G_{k1} + P_{k1} \cdot C_{k-1}))) \\ &= G_{k4} + P_{k4}G_{k3} + P_{k4}P_{k3}G_{k2} + P_{k4}P_{k3}P_{k2}G_{k1} + P_{k4}P_{k3}P_{k2}P_{k1}C_{k-1} \end{aligned}$$

↓      ↓      ↓      ↓

IL BIT di carry in uscita è dato dall'OR di questi prodotti!

I TERMINI  $G$  E  $P$  DIPENDONO DA  $a$  E  $b$  CHE SONO INPUT DEL MODULARE 1-esimo.

$C_{k-1}$  ce l'ho subito,  $= 0$ .

Calcolo  $C_{k4}$  IN FUNZIONE DEGLI INPUT DEL BLOCCO GENERALE.

NON HO PIÙ 4 MODULI CHE LAVORANO ITERATIVAMENTE PER ANDARE A CALCOLARE IL BIT DI CARRY, QUANDO voglio calcolare  $C_{k4}$  che sarà IL BIT DI CARRY DI TUTTO QUEL MODULARE  $K$ , GLI INPUT LI HO GIÀ TUTTI A DISPOSIZIONE.

USO GLI STESSI INPUT (mentre quei moduli si calcolano le somme) PER CALCOLARE IN PARALLELO IL BIT DI CARRY.

LE SOMME VENGONO CALCOLATE ESATTAMENTE COME facevamo prima, quindi utilizzando lo XOR MA IN PARALLELO viene calcolato IL BIT DI CARRY.

0	1	1	1	0	0	1	0	AND
AND - -	$a_{k4}$	$b_{k4}$	$a_{k3}$	$b_{k3}$	$a_{k2}$	$b_{k2}$	$a_{k1}$	$b_{k1}$

XOR MA IN PARALLELO viene calcolato il bit di carry

0

1

1

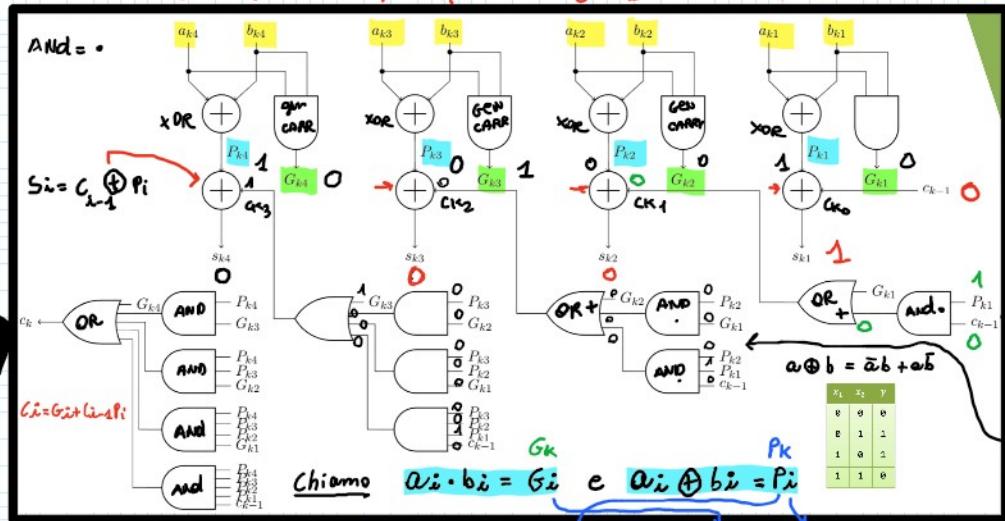
1

0

0

1

0



AND

00 0

01 0

10 0

11 1

$$C_K = G_{K4} + P_{K4} \cdot C_{K3}$$

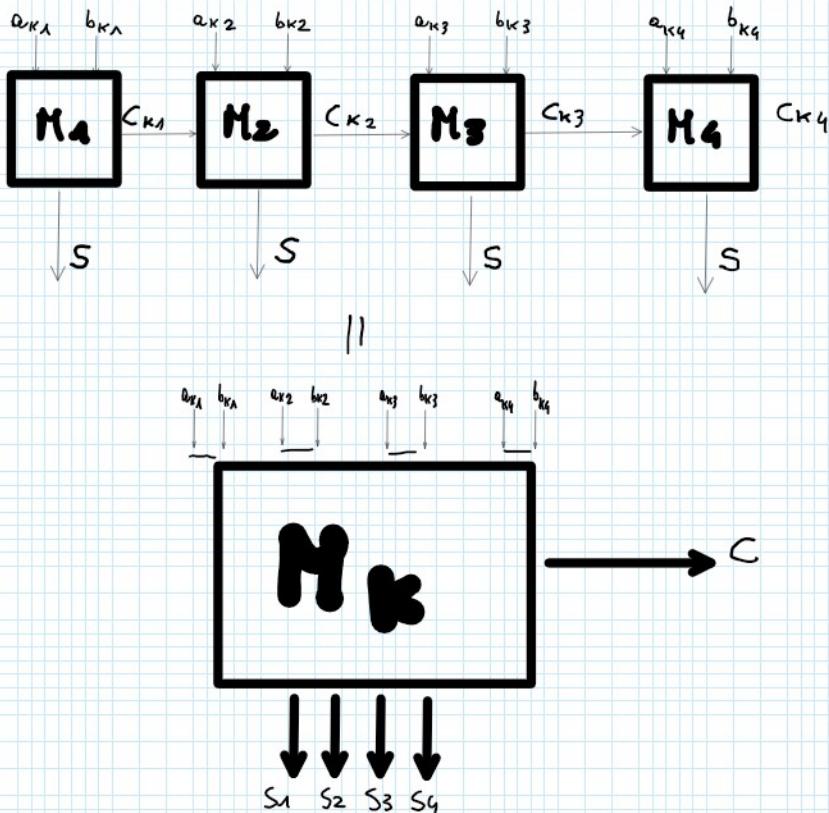
$$C_{K1} = G_{K1} + P_{K1} \cdot C_{K0}$$

$$C_{K2} = G_{K2} + P_{K2} \cdot C_{K1}$$

$$= G_{K2} + P_{K2} \cdot (G_{K1} + P_{K1} \cdot C_{K0})$$

$$= G_{K2} + P_{K2} \cdot G_{K1} + P_{K2} \cdot P_{K1} \cdot C_{K0}$$

$$= G_{K4} + P_{K4}G_{K3} + P_{K4}P_{K3}G_{K2} + P_{K4}P_{K3}P_{K2}G_{K1} + P_{K4}P_{K3}P_{K2}P_{K1}c_{K-1}$$



$$\begin{array}{r}
 & 1 \\
 \begin{array}{rrrrr}
 0 & 1 & 0 & 1 & +
 \\ 1 & 1 & 0 & 0 & =
 \end{array}
 \end{array}$$

$$0 \ 0 \ 0 \ 1$$