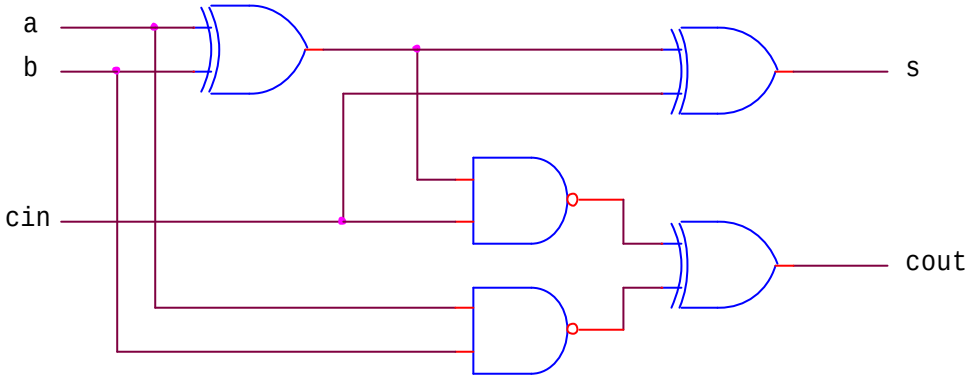
UNIVERSITÀ DELLA CALABRIA

Facoltà d'Ingegneria

Corso di Elettronica dei Sistemi Digitali

Lavoro di Gruppo: *Esercitazione Full-Adder*



Gruppo 19:

Adamo Bosco,	*58834*	adamo.bosco@libero.it
Umile Iaquinta,	*58957*	uiaquinta@virgilio.it
Davide Macrì,	*62529*	macridavide@tiscali.it
Ornella Pisacane,	*60815*	ornypi@libero.it
Leonardo Spataro,	*59176*	LeonHardd@libero.it
Amedeo Zavatto,	*59161*	zavattoame@libero.it

A.A. 2002-2003

INTRODUZIONE:

Nell'architettura degli elaboratori i circuiti di tipo sommatore (e sottrattore) rivestono un ruolo di primissimo piano, dal momento che molte operazioni (come ad esempio la moltiplicazione) sono ricondotte all'esecuzione ripetuta di operazioni di addizione fra bit. Pertanto, visto che l'operazione di somma è richiesta con una frequenza elevata, è necessario che il circuito sommatore sia il più efficiente possibile, cioè veloce nell'eseguire la computazione, il più robusto possibile ai disturbi esterni e soprattutto non eccessivamente costoso.

Il più semplice circuito realizzabile per ottenere la somma fra due numeri ad n *bit si ottiene connettendo in cascata* n *Full-Adder, in modo che l'uscita* c_{out} *di un Full-Adder insista sull'ingresso* c_{in} *del Full-Adder successivo (**Ripple Carry**).*

Esistono poi altre tipologie di circuiti sommatori, intesi a ridurre il tempo necessario al propagarsi del riporto c*. Un esempio in tal senso è fornito dal **Carry Select**, la cui logica prevede la duplicazione di metà del circuito, al fine di riuscire ad elaborare la seconda metà dei bit (quelli che vanno da* $n/2$ *ad* n*) prescindendo temporaneamente dal valore del riporto* $c_{n/2}$*; successivamente, una volta formato il valore definitivo del riporto (carry), si seleziona il corretto risultato in funzione di* $c_{n/2}$ *.*

In tutti i circuiti, indipendentemente dalla logica che ne regola il funzionamento, intesa ad ottimizzare un aspetto (es. velocità di propagazione dei riporti), rispetto ad un altro (es. costo), il componente cardine resta il Full-Adder. Va innanzitutto precisato che il Full-Adder rappresenta, per com'è strutturato, un sommatore base di tipo generale, nel senso che è utilizzato, collegandone molti in cascata, per costruire circuiti in grado di sommare due addendi, a *e* b*, aventi un numero qualsiasi di bit. A differenza dell'Half-Adder, che si limita a sommare* a *e* b *ed a dare in uscita* s *e* c_{out}*, il Full-Adder presenta un terzo ingresso, corrispondente al riporto* c_{in}*; mentre allora un sommatore Half-Adder può essere utilizzato solo all'inizio di una catena di sommatori (in cui si ha* $c_{in} = 0$*), il Full-Adder è utilizzabile in tutti i punti della catena. Questo rappresenta un grosso vantaggio dal punto di vista costruttivo*

del componente, in quanto consente di realizzare circuiti complessi come semplici collegamenti fra componenti identici, rende le macchine costruttrici modulari e standardizzate, così da rendere disponibili i circuiti a basso costo. Per di più l'Half-Adder non è utilizzabile qualora si debba eseguire un'operazione del tipo $a-(+b)$*. Infatti, poiché il sommatore è in grado di eseguire solo somme, per effettuare l'operazione è necessario che il calcolo venga impostato come* $a+(-b)$*. Se, come solitamente accade in un elaboratore, i numeri vengono rappresentati nel formato Complemento a due,* C_2*, il numero* $-b$ *viene ricavato da* $+b$ *mediante la somma fra 1 ed il flip dei bit (cioè sostituzione dei bit zero con uno e viceversa, operazione che rappresenta il Complemento ad uno di* b*,* C_1*). La complementazione bit-a-bit è ottenibile prima dell'ingresso del dato nel sommatore tramite, ad esempio, l'uscita negata di un flip-flop, mentre il numero 1 da sommare va impostato su* c_{in} *(ingresso non presente in un Half-Adder). Appare pertanto palese la maggiore flessibilità del Full-Adder rispetto all'Half-Adder, nell'eseguire somme fra numeri positivi (*$c_{in}=0$*) e fra un positivo ed un negativo (*$c_{in}=1$*). Per questi motivi il nostro studio è rivolto esclusivamente al Full-Adder.*

PROGETTAZIONE:

Si realizzi un Full-Adder, utilizzando porte NAND e XOR.

Per poter determinare lo schema del Full-Adder, adoperando porte $NAND$ *e* XOR*, è stato necessario definire la seguente tabella di verità:*

Ingressi			*Uscite*	
a	b	c_{in}	s	c_{out}
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

Introduciamo due funzioni ausiliarie, Propagate, P*, e Generate,* G*, definite nel seguente modo:*

$$P = \begin{cases} 1 \ \ se \ \ a \neq b \\ 0 \ \ altrimenti \end{cases} \qquad G = \begin{cases} 1 \ \ se \ \ a \equiv b = 1 \\ 0 \ \ altrimenti \end{cases}$$

Evidentemente:

$G = a \cdot b$ *mentre* $P = a \oplus b$,

il simbolo $\oplus$ *rappresenta l'operatore* XOR *(eXclusive* OR*), definito come* $a \oplus b = a \cdot \overline{b} + \overline{a} \cdot b$.

Si può notare che* P *e* G *non sono mai contemporaneamente alti.

Possiamo, quindi, definire il bit di somma utilizzando P*:*

$$S = P \oplus c_{in} \qquad (1)$$

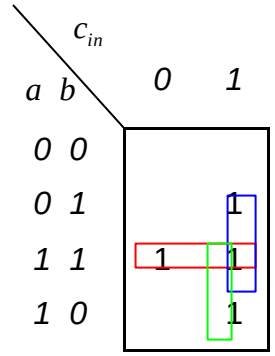
mentre per il bit di riporto si ha:

$$c_{out} = G + P \cdot c_{in} \qquad (2)$$

Al fine di ottenere s e c_{out} come funzione degli ingressi a, b e c_{in}, espressa in Prima Forma Canonica (la DNF, ovvero Disjunctive Normal Form, Forma Normale Disgiuntiva, che esprime la funzione come disgiunzione (OR) di congiunzioni (AND)), ricorriamo all'uso delle Mappe di Karnaugh:

a	b	$c_{in}=0$	$c_{in}=1$
0	0		1
0	1	1	
1	1		1
1	0	1	

Mappa di Karnaugh per s



Mappa di Karnaugh per cout

Le forme analitiche minime sono pertanto:

$$[K]\begin{cases} s = \bar{a}\cdot b\cdot \overline{c_{in}} + \bar{a}\cdot \bar{b}\cdot c_{in} + a\cdot b\cdot c_{in} + a\cdot \bar{b}\cdot \overline{c_{in}} \\ \\ c_{out} = a\cdot b + a\cdot c_{in} + b\cdot c_{in} \end{cases}$$

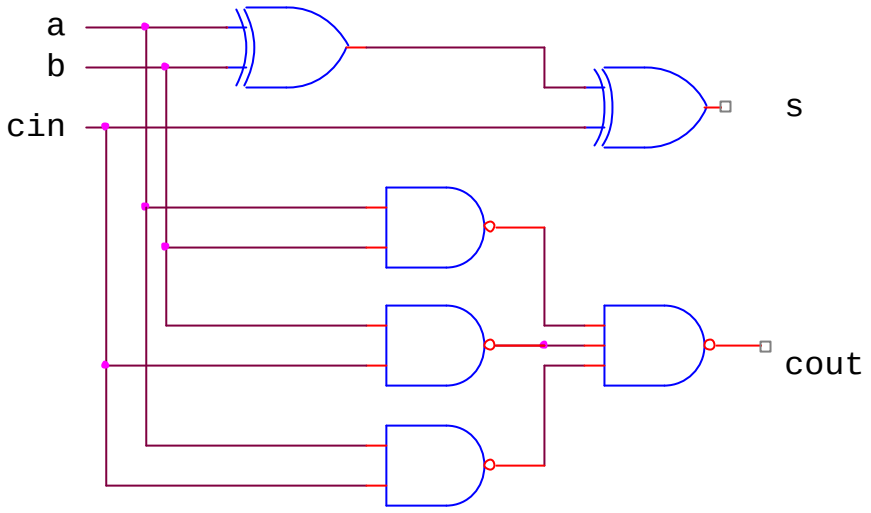
Notiamo che s presenta 4 implicanti primi essenziali di dimensione 1, mentre c_{out} ne ha 3 (in quanto la configurazione 111 è condivisa dai 3 implicanti primi essenziali di dimensione 2).

Quella ottenuta mediante le mappe di Karnaugh è la forma minima (e pertanto ottima) delle funzioni s e c_{out}, in quanto la rete ha profondità due.

Osserviamo inoltre che sono necessarie quattro porte AND a tre ingressi ed una porta OR a quattro ingressi per realizzare la rete che genera s, tre porte AND a due ingressi ed una porta OR a tre ingressi per costruire la rete generatrice di c_{out}. Negando la formula relativa a c_{out} due volte, si ottiene la formula minima in funzione di $NAND$ a tre ingressi, che non abbiamo potuto

realizzare perché ci sono stati forniti circuiti integrati contenenti porte $NAND$ *a due soli ingressi:*

I. $c_{out} = (a \mid b) \mid (a \mid c_{in}) \mid (b \mid c_{in})$



Full-Adder con NAND a 3 ingressi

Arrivati a questo punto, dimostriamo che la formula $c_{out} = G + Pc_{in}$ *è equivalente logicamente alla forma minima ottenuta mediante le mappe di Karnaugh:*

$c_{out} = a \cdot b + (a \cdot \bar{b} + \bar{a} \cdot b) \cdot c_{in} = a \cdot b + a \cdot \bar{b} \cdot c_{in} + \bar{a} \cdot b \cdot c_{in}$,

usando il Metodo del Completamento del prodotto otteniamo:

$c_{out} = a \cdot b \cdot 1 + a \cdot \bar{b} \cdot c_{in} + \bar{a} \cdot b \cdot c_{in} = a \cdot b \cdot (c_{in} + \overline{c_{in}}) + a \cdot \bar{b} \cdot c_{in} + \bar{a} \cdot b \cdot c_{in} =$
$a \cdot b \cdot c_{in} + a \cdot b \cdot \overline{c_{in}} + a \cdot \bar{b} \cdot c_{in} + \bar{a} \cdot b \cdot c_{in} =$

$a \cdot b \cdot (c + \bar{c}) + a \cdot c \cdot (b + \bar{b}) + b \cdot c \cdot (a + \bar{a}) = a \cdot b + a \cdot c_{in} + b \cdot c_{in}$ ≡ *Forma minima ottenuta con Karnaugh.*

In realtà, non avendo a disposizione porte AND *ed* OR*, è stato necessario esplicitare il bit di riporto in funzione di porte* XOR e $NAND$.

Supponiamo di negare due volte la definizione data del bit di carry:

$\overline{\overline{(G + Pc_{in})}} = \overline{\overline{(a \cdot b) + (a \oplus b) \cdot c_{in}}} =$

applicando la regola di DE MORGAN otteniamo:

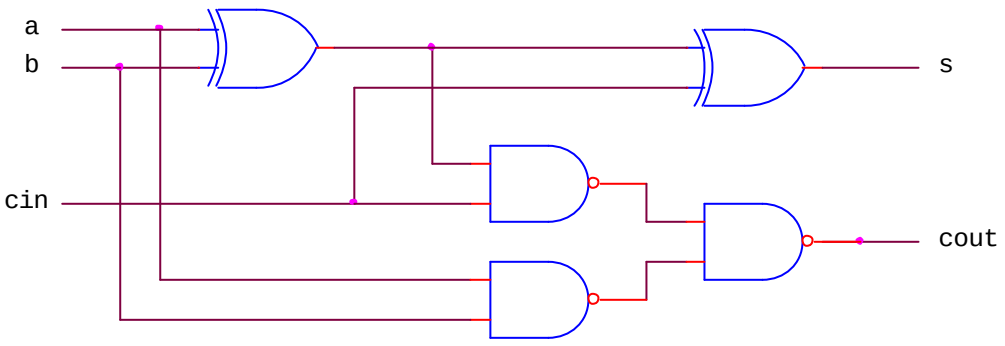
$\overline{\overline{(a \cdot b)} \cdot \overline{((a \oplus b) \cdot c_{in})}} = \overline{(a \mid b) \cdot ((a \oplus b) \mid c_{in})}$

Poiché:

a	b	c_{in}	P	$a \mid b = R$	$P \mid c_{in} = T$	**R | T**	**R ⊕ T**
0	0	0	0	1	1	**0**	**0**
0	0	1	0	1	1	**0**	**0**
0	1	0	1	1	1	**0**	**0**
0	1	1	1	1	0	**1**	**1**
1	0	0	1	1	1	**0**	**0**
1	0	1	1	1	0	**1**	**1**
1	1	0	0	0	1	**1**	**1**
1	1	1	0	0	1	**1**	**1**

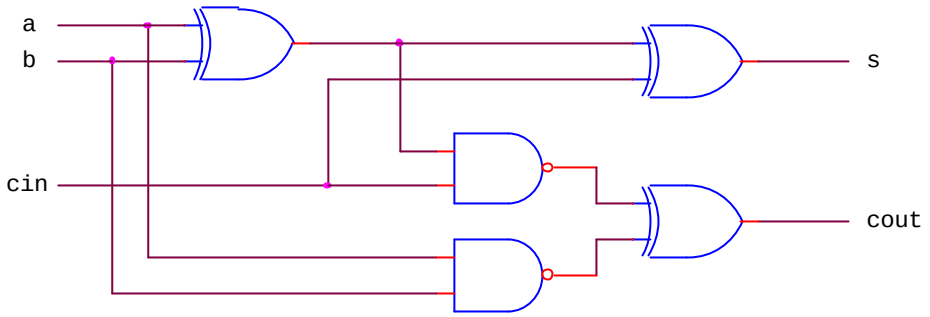
abbiamo dimostrato che, in questo caso, è possibile sostituire la $NAND$ *con una* XOR*. Allora, le altre due possibili forme per il bit di riporto sono:*

II. $c_{out} = (a \mid b) \mid ((a \oplus b) \mid c_{in})$



Full-Adder con porta NAND

III. $c_{out} = (a \mid b) \oplus ((a \oplus b) \mid c_{in})$



Full-Adder con porta XOR

Per dimostrarlo per via analitica basta osservare in primo luogo che $c_{out} = \overline{(a \cdot b)} \oplus \overline{(P \cdot c_{in})}$ *è equivalente a* $c_{out} = (a \cdot b) \oplus (P \cdot c_{in})$*, cioè che lo* XOR *fra due* $NAND$ *è logicamente equivalente allo* XOR *fra due* AND*, infatti*

$$c_{out} = \overline{(a \mid b)} \cdot (P \mid c_{in}) + (a \mid b) \cdot \overline{(P \mid c_{in})} =$$

$$(a \cdot b) \cdot \overline{(P \cdot c_{in})} + \overline{(a \cdot b)} \cdot (P \cdot c_{in}) = (a \cdot b) \oplus (P \cdot c_{in}).$$

A sua volta la formulazione $c_{out} = (a \cdot b) \oplus (P \cdot c_{in})$ *è equivalente a*

$c_{out} = (a \cdot b) + (P \cdot c_{in}) = G + Pc_{in}$*,*

in cui allo XOR *è stato sostituito un* OR*, dal momento che* P *(e conseguentemente* $P \cdot c_{in}$*) e* G *non hanno mai contemporaneamente valore* 1*.*

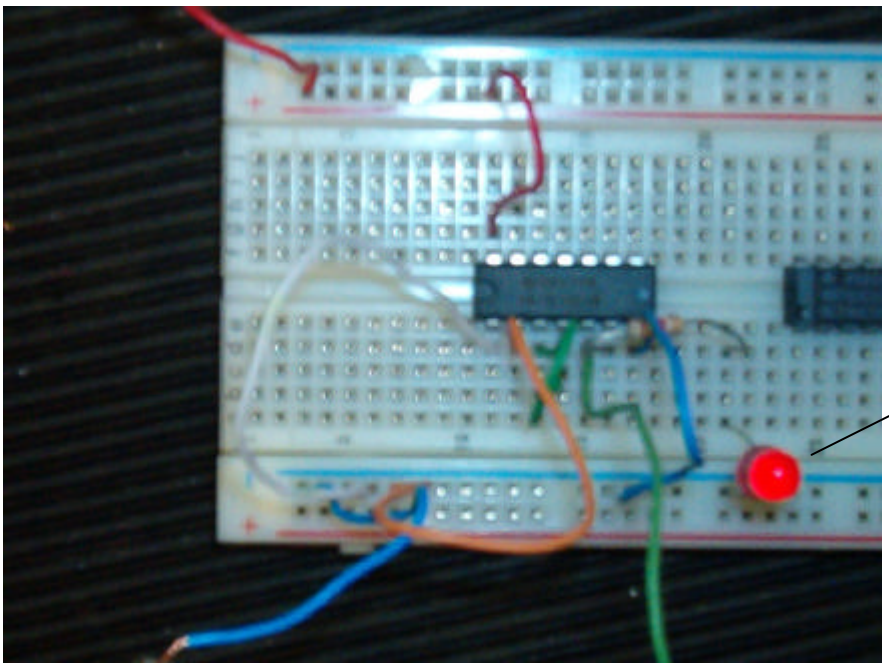
Il circuito da noi realizzato è quello relativo allo schema 3.

VERIFICA FUNZIONAMENTO:

Al fine di verificare il corretto funzionamento del circuito, lo stato delle uscite relative ai bit di somma e di riporto verrà visualizzato mediante diodi LED .Se correttamente connesso, ciascun diodo LED si accenderà in corrispondenza dell'1 logico. Gli ingressi a, b, c_{in} possono essere imposti connettendoli, alternativamente e secondo la combinazione degli ingressi che si vuole ottenere, a massa (0V) o a V_{cc} (5V).

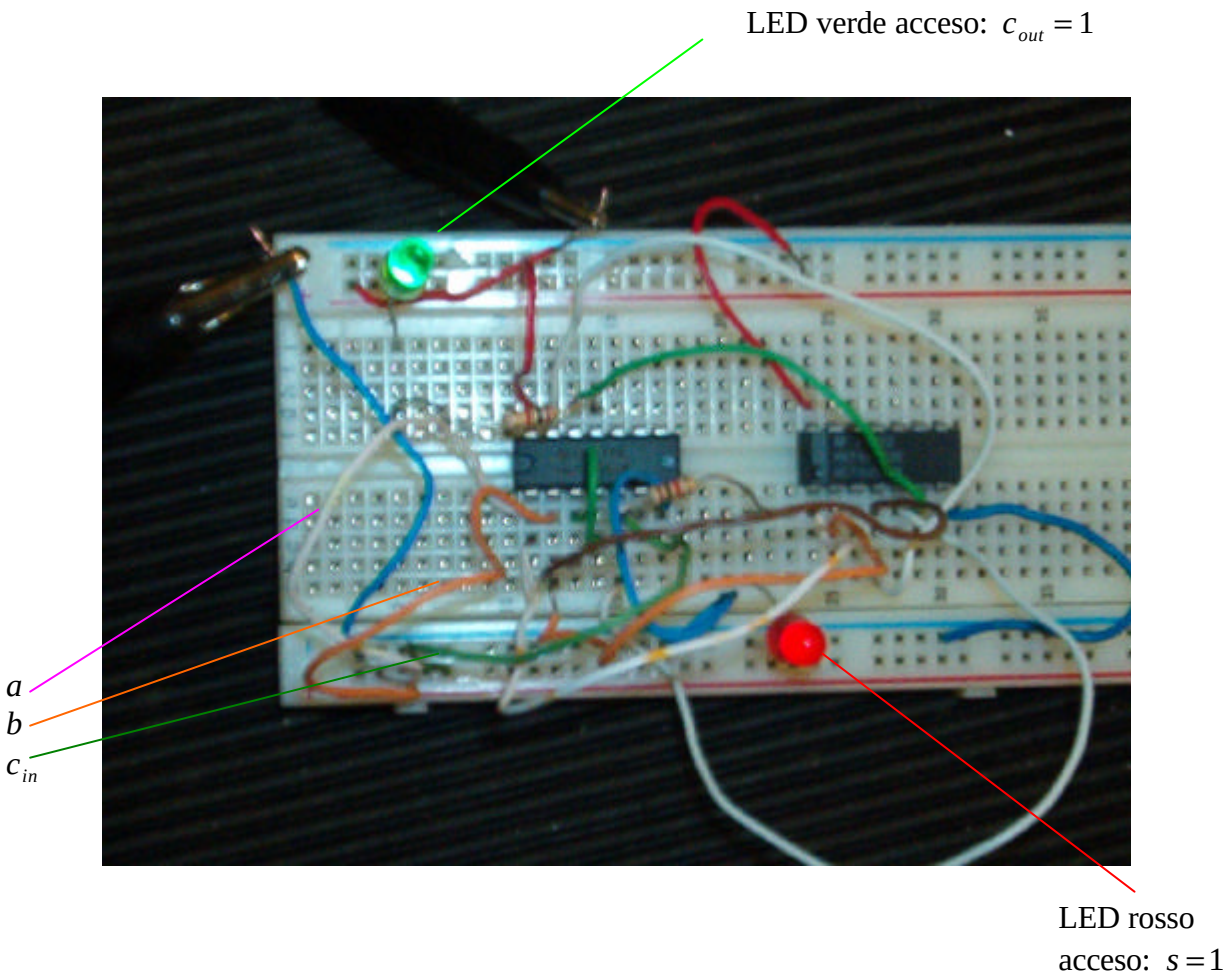
In primo luogo si è verificata la correttezza del circuito relativo al bit di somma s. *Riportiamo solo un esempio del corretto funzionamento del circuito in esame, relativamente alla seguente configurazione degli ingressi:*

$a = b = 0$ *e* $c_{in} = 1$



LED rosso acceso: $s = 1$

A questo punto abbiamo proceduto alla verifica dell'intero circuito considerando anche il bit di riporto.



relativamente alla configurazione $a = b = c_{in} = 1$

VERIFICA DINAMICA:

Misurare i tempi di propagazione relativi al bit di riporto.

Arrivati a questo punto abbiamo effettuato la misurazione dei tempi di propagazione del circuito realizzato, relativamente alla formazione del bit di riporto c_{out}.

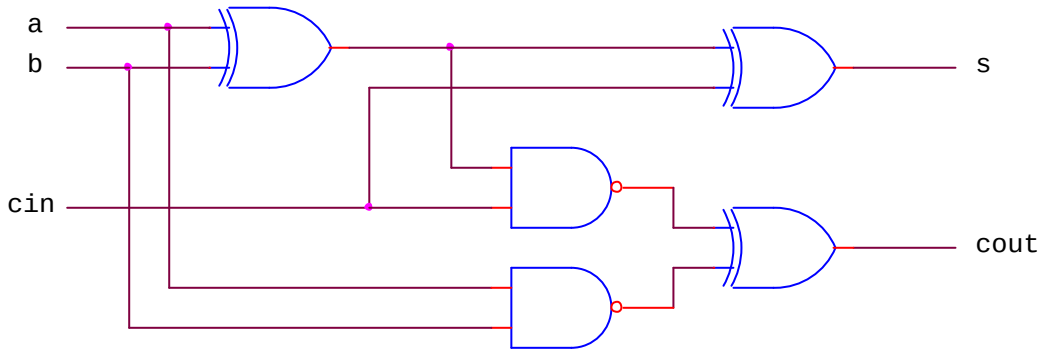
Le condizioni sperimentali sono il bit a *pari ad uno e quello* b *pari zero. Sul bit* c_{in} *abbiamo imposto il segnale variabile, nella fattispecie un'onda quadra di frequenza pari a circa* $6KHz$.

Abbiamo scelto questa configurazione (che viene di seguito schematizzata):

$a = 1 \quad b = 0 \quad c_{in} = \text{var}\,iabile$

in modo da generare le condizioni peggiori di funzionamento. È infatti quando $a \neq b$ *che si ha il fenomeno della Propagazione del Riporto, ossia* $c_{out} = c_{in}$.

Inoltre è da notare che in questo caso il segnale di riporto viene composto tramite due segnali. Uno di essi ha attraversato due porte logiche; quindi c_{out} *è ottenuto tramite l'attraversamento di 3 livelli che è appunto la profondità massima della nostra rete.*



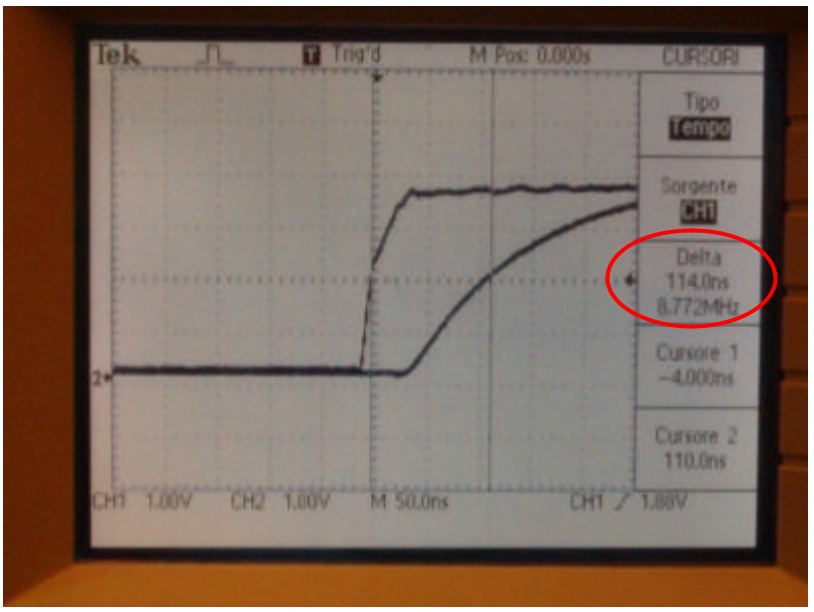
Circuito Full-Adder analizzato

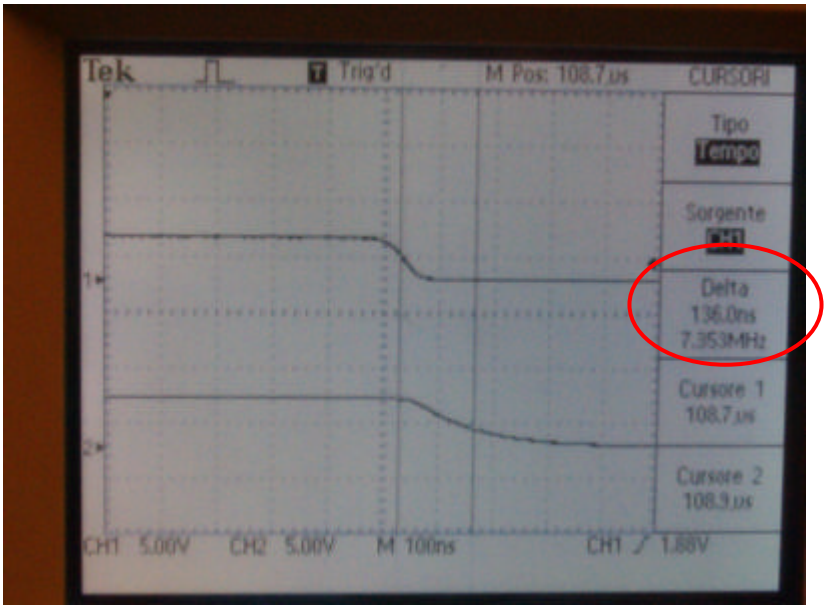
Ponendo la sonda dell'oscilloscopio al morsetto di uscita della resistenza interposta tra il circuito integrato contenente le porte *XOR* ed il diodo LED abbiamo misurato i seguenti tempi di propagazione:

$c: 0 \rightarrow 1 \quad \Rightarrow \quad \mathbf{tp = 114ns}$

$c: 1 \rightarrow 0 \quad \Rightarrow \quad \mathbf{tp = 136ns}$

come viene documentato con le seguenti foto:





Ora, quello che possiamo osservare è che i tempi di propagazione sono dovuti, come si evince dallo schema a porte logiche, da una somma di ritardi dovuti proprio al ritardo intrinseco in ogni componente logico nel generare il risultato di output.

In ingresso all'ultimo XOR *abbiamo due segnali: uno che ha un ritardo dovuto ad uno* XOR *che confluisce a sua volta in una porta* $NAND$; *un secondo segnale che proviene da una porta* $NAND$.

Schematizzando abbiamo che:

$t_{Signal1} \Rightarrow t_{XOR} + t_{NAND}$

$t_{Signal2} \Rightarrow t_{NAND}$

Abbiamo quindi che il ritardo totale sarà dell'ordine di

$t_{c_{out}} \Rightarrow t_{XOR} + t_{NAND} + t_{XOR}$.