UNIVERSITÀ DEGLI STUDI DI PADOVA

CORSO DI LAUREA IN INGEGNERIA MECCATRONICA

TESI DI LAUREA TRIENNALE

Simulazione SPICE di un moltiplicatore di Wallace a 4 bit

Relatore: Prof. SIMONE BUSO

Laureando: MATTIA CACEFFO

Matricola 562925-IMC

ANNO ACCADEMICO 2011-2012

Sommario

Nel presente elaborato, partendo dalle basi fornite nel corso di “Teoria dei Circuiti Digitali”, viene affrontato il problema della realizzazione di un moltiplicatore a 4bit con ritardi fino alla sua rappresentazione di Wallace. Verrà utilizzato il programma SPICE nella versione commerciale PSpice per l’esecuzione delle simulazioni, essendo diventato negli anni uno standard per la realizzazione di simulazioni circuitali. Partendo dalla scelta delle singole porte e dai loro ritardi dichiarati e simulati, si realizzeranno i vari componenti che costituiscono un moltiplicatore per poi giungere alla sua forma completa. Questo procedimento sarà svolto ponendo sempre particolare attenzione ai tempi di propagazione dei segnali, cercando dove possibile l’ottimizzazione. Di volta in volta sarà controllata la presenza di casi in cui i ritardi misurati differiscano dai valori attesi ideali; scegliendo in ogni occasione, se necessario, i componenti e il circuito che sintetizzano nel modo migliore il moltiplicatore di Wallace.

Indice

Introduzione

Negli ultimi anni si sono diffusi numerosi software in grado di compiere una veloce rappresentazione di circuiti elettronici reali, mediante modelli virtuali di buona approssimazione.
Questi modelli permettono di avere un'idea affidabile di come si può comportare un circuito una volta che sia fisicamente realizzato.
In questa tesi viene schematizzato un moltiplicatore a 4bit, mediante simulazione del suo circuito con il programma SPICE (nella sua versione PSpice) e, durante il percorso, vengono prese in considerazione eventuali ottimizzazioni.
Il primo passo effettuato è stato quello di scegliere le porte fondamentali da usare (AND, OR, NOT, XOR, ecc) con relativi ritardi, per realizzare i primi mattoncini fondamentali.
Tali mattoncini sono il sommatore Half Adder (HA) e Full Adder (FA).
Mediante l'uso di HA e FA viene sviluppato un sommatore a 4 bit con resto che si propaga (Ripple-Carry), ma visti i limiti che avrebbe in termini di ritardi, viene poi realizzato un sommatore con Carry-LookAhead in grado di ridurre il ritardo dovuto al riporto che si propaga lungo il sommatore Ripple-Carry.
A dimostrazione della scelta fatta, è stato realizzato un sommatore a 16 bit composto da blocchi di sommatori a 4bit CLA confermando la notevole differenza di ritardo tra le due rappresentazioni all'aumentare dei bit usati.
Una volta realizzati i componenti fondamentali, si è passati al moltiplicatore vero e proprio, costituito da un insieme di sommatori a 4bit in precedenza realizzati.
In ultima analisi, si vedrà un'ottimizzazione del moltiplicatore, nota con il nome di "Wallace" che permette di risparmiare il numero di porte usate e quindi lo spazio del medesimo componente sul silicio, comportando anche una riduzione del tempo di ritardo complessivo.

1.1 Ritardi

Ogni dispositivo fisico reale ha dei limiti che lo caratterizzano, come delle specifiche di funzionamento, e, a seconda delle condizioni di utilizzo, un determinato ritardo tra l'ingresso del segnale e la sua uscita. Una porta logica elementare è essa stessa soggetta a un ritardo definito come l'intervallo che trascorre dal momento in cui i segnali di ingresso vengono forniti in modo stabile e il momento in cui l'uscita si è stabilizzata, una volta esaurito il transitorio.
Generalmente, le porte logiche non hanno tutte un uguale tempo di propagazione e se combinate tra loro, possono generale malfunzionamenti delle uscite al variare degli ingressi.
Quando si passano a un circuito, dei valori in ingresso variabili nel tempo, è indispensabile attendere sempre lo stabilizzarsi delle uscite, prima di effettuare ulteriori cambiamenti.
In caso contrario, si andrebbero a leggere dei valori in uscita che sono solo provvisori e non definitivi e quindi del tutto irrilevanti come informazioni.
Un effetto visibile di come il segnale sia indeterminato durante il suo transitorio è fornito dalle alee grafiche.
Le alee corrispondono ai valori di un segnale logico, temporaneamente diversi da quello atteso, causati per effetto dei ritardi introdotti dai vari gate di una rete combinatoria.

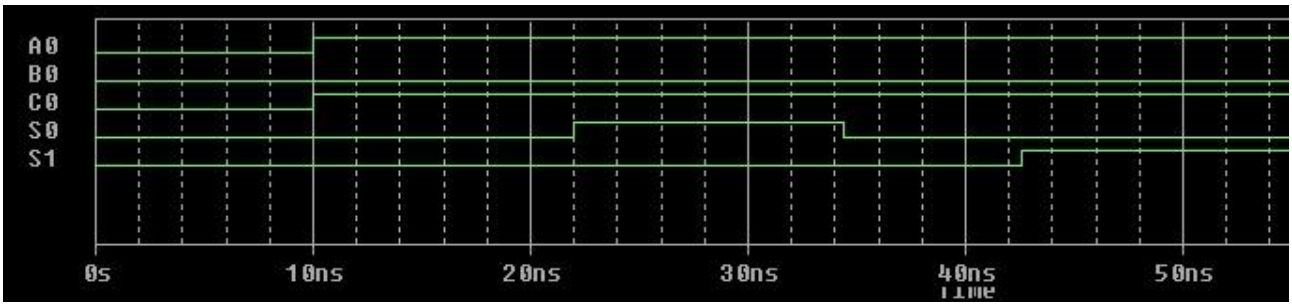

Figura 1.1 *Comportamento dinamico di un Full Adder*

Una lettura del valore dell'uscita prima che sia stabilizzata, comporterebbe un errore e sarebbe inutile a qualsiasi analisi del circuito, per cui, durante le simulazioni, vanno presi sempre ingressi che non variano in modo troppo repentino.
Come si vede in figura 1.1, il segnale S0 che rappresenta l'uscita di un sommatore Full Adder, sarebbe disponibile per una lettura affidabile solo dopo un intervallo τ in questo caso di 25ns circa dall'ultima variazione degli ingressi (A0, B0), mentre prima assume transitoriamente valori errati. Per il segnale S1 (riporto in uscita) il ritardo è ancora maggiore.

1.2 Porte usate e relativi ritardi

PSpice presenta nelle sue librerie una vasta scelta di modelli che rappresentano componenti reali adatti a effettuare simulazioni.
Dopo una prima prova con porte logiche ideali, si è scelto di prendere in esame le porte della famiglia 74HC** prodotte con tecnologia CMOS.
Andando a vedere i dati dichiarati da costruttori di dispositivi analoghi e provando delle semplici simulazioni sulle porte, si sono ottenuti i seguenti risultati:

Porta	**Sigla**	**Ritardo tipico**	**Ritardo misurato**
AND	74HC08	7ns con 6V 9ns con 4,5V	10ns
OR	74HC32	6ns con 6V 8ns con 4,5V	10ns
NOR	74HC02	7ns con 6V 9ns con 4,5V	9ns
XOR	74HC86	11ns con 6V 14ns con 4,5V	12ns
NAND	74HC00	7ns con 6V 9ns con 4,5V	9ns
NOT	74HC04	7ns con 6V 9ns con 4,5V	9ns

Tabella 1.1 *Porte logiche reali e simulate*

Come si può vedere, i valori ottenuti con SPICE, dove viene applicata una tensione di alimentazione a default di 5 Volt rispecchiano abbastanza fedelmente i valori dichiarati nei datasheet dei componenti reali.
Il doppio valore dichiarato a 6 e 4,5 Volt è dovuto al fatto che il ritardo diminuisce al crescere della tensione di alimentazione, fino a un valore limite che dipende dalla tecnologia utilizzata.

1.3 Half Adder

L'Half adder è il generico componente elettronico digitale, noto anche come semisommatore, in grado di sommare due bit A e B e fornire il risultato in una variabile S e un eventuale resto C.
Il semisommatore, come si vede, non tiene conto di un eventuale resto in ingresso e costituisce quindi un elemento a due ingressi e due uscite.
Esso costituisce uno dei mattoncini fondamentali per la realizzazione di componenti digitali.

Tabella della verità:

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Tabella 1.2 *Tabella della verità di un Half Adder*

Formula di base:
$A + B = S + C$ (1)
Utilizzando nozioni base di algebra booleana possiamo scrivere:
$S = A'B + AB' = A \oplus B$ [1] (2)
$C = A * B$ (3)

Come si vede, il risultato della somma può essere eseguito anche con una semplice porta XOR.
La rappresentazione equivalente in PSpice delle equazioni (2) e (3) è mostrata in Fig.1.2:

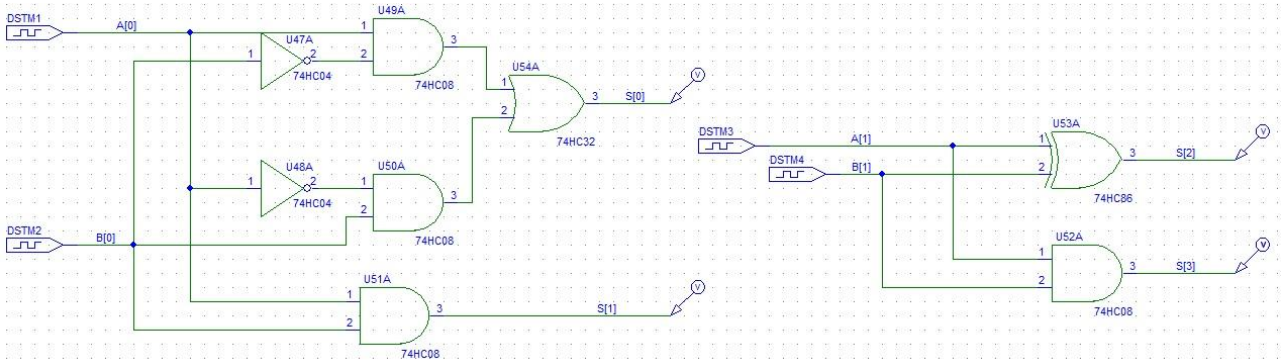

Figura 1.2 *Rappresentazione dei circuiti PSpice per una HA*

Le due rappresentazioni forniscono i medesimi risultati a parità di ingressi, presentando ritardi differenti.

[1] A' e B' corrispondono ad A negato e B negato.

1.4 Full Adder

Il sommatore a un bit Full Adder è alla base dei moderni calcolatori e consente di sommare due numeri binari, tenendo in considerazione anche l'eventuale resto in ingresso.
Esso è un componente a tre ingressi e due uscite.

Tabella della verità:

A	B	Cin	S	Cout
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

Tabella 1.3 *Tabella della verità di un Full Adder*

Formula di base:
$A + B + Cin = S + Cout$ (4)
Utilizzando nozioni base di algebra booleana possiamo scrivere:
$S = A'B'C + A'BC' + AB'C' + ABC$ (5)
da cui derivano le seguenti
$S = C(AB + A'B') + C'(A'B + AB')$ (6)
$S = C(A \oplus B)' + C'(A \oplus B)$ (7)
$S = C \oplus (A \oplus B)$ (8)
Per cui il riporto in uscita vale invece:
$Cout = A'BC + AB'C + ABC' + ABC$ (9)
$Cout = AB(C' + C) + C(A'B + AB')$ (10)
$Cout = AB + C(A \oplus B)$ (11)

La rappresentazione in SPICE di un Full Adder è mostrata in Fig.1.3:

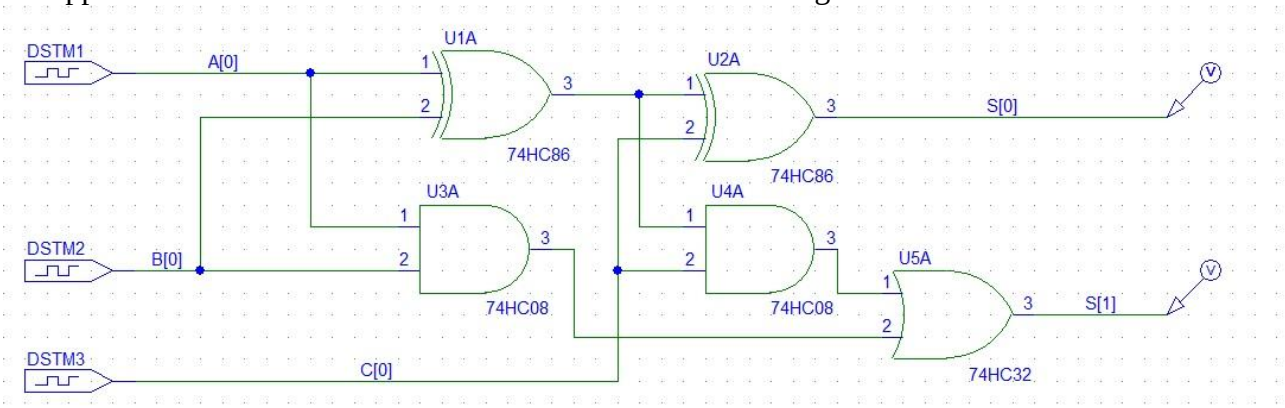

Figura 1.3 *Rappresentazione di un Full Adder in PSpice*

Una prima verifica dell'efficacia del simulatore PSpice, si può avere stimando il tempo d'uscita del Full Adder.
Come riferimento, prendiamo il ritardo massimo che può avere un segnale per attraversare tutto il circuito nel caso percorra il tragitto più lento.
In questo caso, si tratta del percorso in cui debba attraversare una porta XOR, una AND e una OR.
Per cui si stima:
XOR (12ns) + AND (10ns) + OR (10ns) = 32 ns.
Come si vede in Fig. 1.4, la simulazione con SPICE fornisce un valore di circa 32.5 ns confermando quanto stimato.

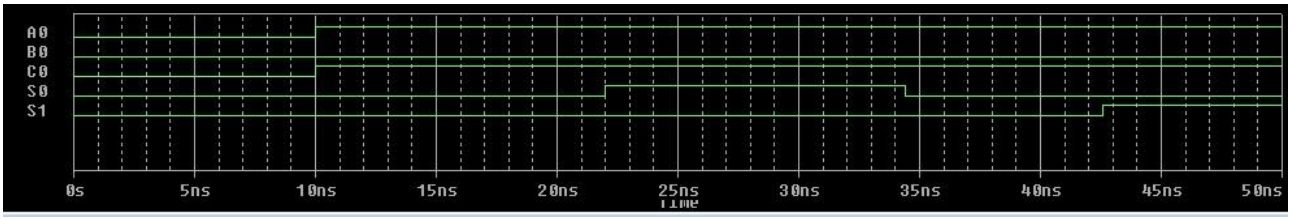

Figura 1.4 *Simulazione PSpice di un Full Adder*

1.5 Considerazioni sui ritardi

Come accennato in precedenza, nello studio dei circuiti digitali, è buona norma, stimare i ritardi prendendo come riferimento quelli dovuti al caso peggiore in cui il segnale è costretto a percorrere il percorso più lento.
Nella pratica, questa scelta rappresenta solo un caso molto particolare, ma è proprio il caso critico per cui l'attesa per ottenere un'uscita stabile è la più lunga possibile.
Come abbiamo potuto osservare, PSpice fornisce dei ritardi congruenti con i dati dichiarati nei datasheet e le stime effettuate fino ad ora.

2.1 Sommatore Ripple-Carry

La realizzazione di un Ripple Carry Adder (RCA), si ottiene collegando in cascata n sommatori Full Adder per effettuare la somma di due numeri composti da n bit ciascuno.
Il principio di funzionamento di questo sommatore si basa sulla propagazione dell'errore, per cui il resto in uscita da un determinato stadio, diventa l'ingresso dello stadio successivo.

$C_i = Cin_i = Cout_{i-1}$ (12)

Nel caso specifico della realizzazione di un sommatore a 4 bit, andiamo a mettere in cascata 4 sommatori Full Adder e passiamo come ingresso a ciascuno di loro, il resto del precedente.
La rappresentazione in SPICE di un sommatore Ripple Carry a 4bit è mostrato in Fig.2.1:

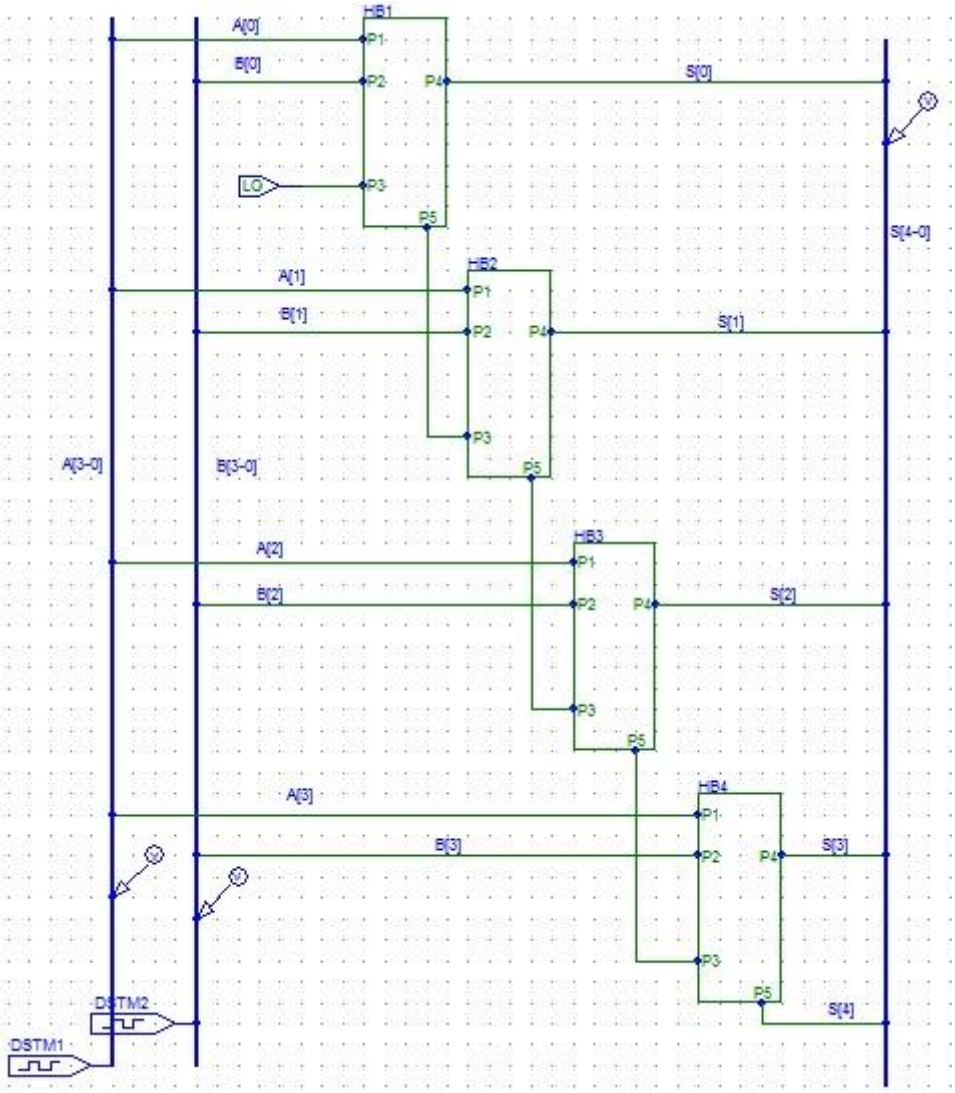

Figura 2.1 *Sommatore a 4bit di tipo Ripple Carry; ogni blocco contiene un Full Adder*

Ancora una volta, il ritardo complessivo viene analizzato percorrendo la strada più lenta da percorrere per il segnale.

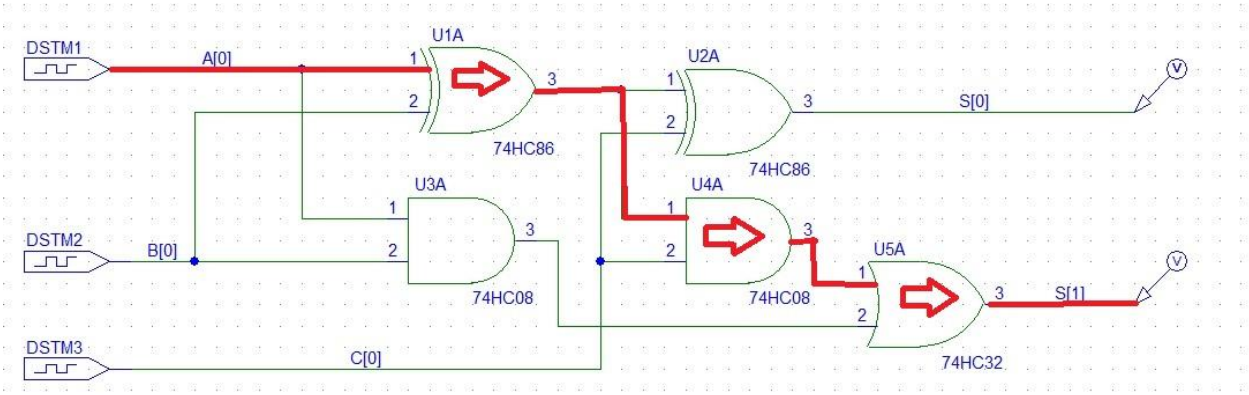

Figura 2.2 *Percorso più lungo per il primo sommatore*

Nel primo sommatore è necessario attraversare un XOR (12ns), un AND (10ns) e un OR (10ns).

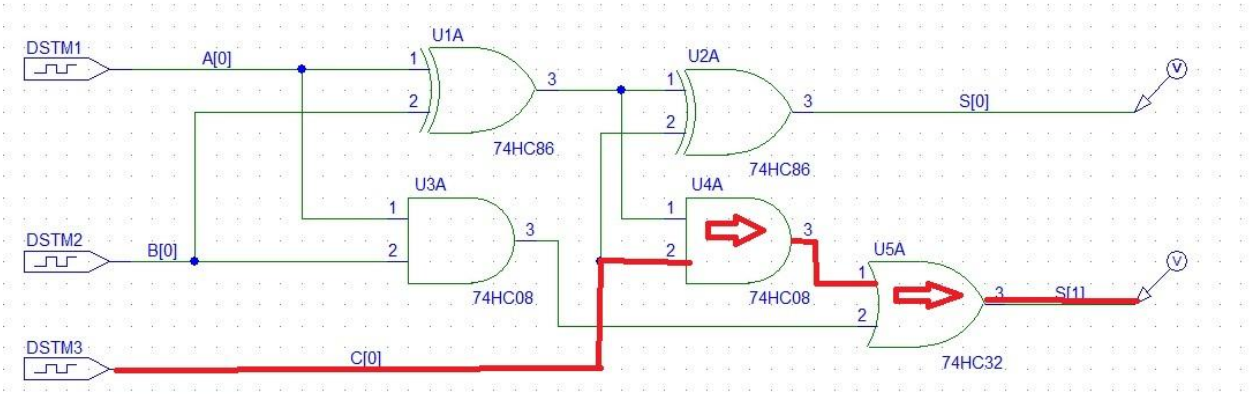

Figura 2.3 *Percorso più lungo per gli stadi successivi*

Negli stadi successivi invece, il resto attraversa solo la porta AND(10ns) e OR(10ns) dei rimanenti Full Adder.
Ogni sommatore dipende dal riporto del precedente e quindi possiamo calcolare un ritardo complessivo cosi composto:

$$12 + 10 + 10 + 3(10 + 10) = 92ns. \quad (13)$$

Nelle simulazioni difficilmente si ottiene questo valore, dovuto appunto al caso peggiore in cui tutte le porte tardino al massimo possibile e che il segnale percorra la strada più lenta.
In una rappresentazione di questo tipo, il tempo di propagazione del carry è proporzionale a 2N (con N = numero di bit delle parole sommate) e quindi, all'aumentare del numero di bit usati, si ha un drastico calo delle prestazioni.

2.2 Sommatore Carry-Look Ahead

Per ovviare ai limiti della rappresentazione RCA, in un sommatore Carry-Look Ahead il resto i-esimo viene ottenuto attraverso una logica dedicata.
Non sfruttando più la normale propagazione attraverso tutti i Full Adder dei bit precedenti, il resto è calcolato utilizzando direttamente come ingressi il carry di ingresso Cin e tutti i bit fino all'i-esimo.
Riesaminando le formule alla base del Full Adder:

$$S = C \oplus (A \oplus B) \quad (14)$$

$$Cout = AB + C(A \oplus B) \quad (15)$$

Si vede come Cout dipenda solamente da un termine legato ai due bit locali e a un termine propagativo che tiene conto del valore di riporto precedente.
Definendo:
Termine di generazione (generatore):

$$G = AB \quad (16)$$

Termine di propagatore:

$P = A \oplus B$ o equivalentemente $A + B$ [2]. (17)

Si può riscrivere Cout come:

$$Cout = G + CinP \quad (18)$$

Cin, come si vede, è proprio il Cout del blocco precedente, di cui rappresento ingressi e uscite con un pedice "-1":

$$Cout=AB+[A_{-1}B_{-1}+Cin_{-1}\,(A_{-1}\oplus B_{-1})](A\oplus B) \quad (19)$$

Questo ragionamento si può iterare fino ad arrivare al primo blocco, in cui l'ingresso Cin è chiaramente 0.
Nel nostro caso, in cui viene usato un sommatore a 4bit il carry finale viene calcolato come:

$$Cout= G_i\{G_{i-1}+[G_{i-2}\,(G_{i-3}+0\,P_{i-3})P_{i-2}]P_i\}=$$
$$=G_i+P_i\,G_{i-1}+P_i\,P_{i-1}\,G_{i-2}+P_i\,P_{i-1}\,P_{i-2}\,G_{i-3}+P_i\,P_{i-1}\,P_{i-2}\,P_{i-3}\,G_{i-4} \quad (20)$$

L'ultimo termine si elide in quanto G_{i-4}=0.
Quindi otteniamo per i riporti in uscita:

$$c1 = g0 + (p0c0) \quad (21)$$

$$c2 = g1 + (p1g0) + (p1p0c0) \quad (22)$$

$$c3 = g2 + (p2g1) + (p2p1g0) + (p2p1p0c0) \quad (23)$$

$$c4 = g3 + (p3g2) + (p3p2g1) + (p3p2p1g0) + (p3p2p1p0c0) \quad (24)$$

In conclusione, si avrà resto in uscita solo se verrà generato nello stadio presente con A e B alti, o se viene propagato quello proveniente dagli stadi a monte.

[2] Tra A⊕B ed A+B il caso che fa la differenza è quello in cui A e B sono uguali 1, ma allora il carry si propaga comunque e P non serve più.

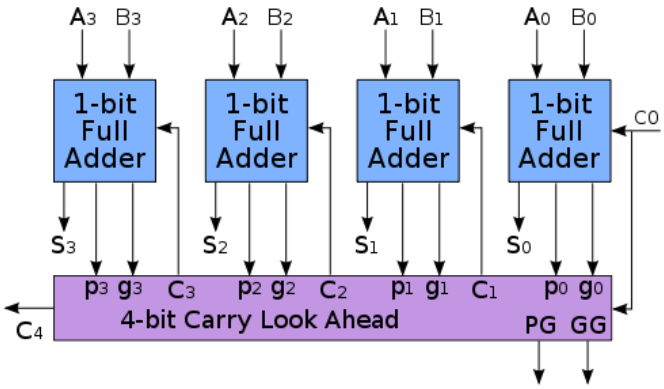

Figura 2.4 *Struttura di un sommatore 4bit Carry-Look Ahead (CLA).*
("Wikipedia, l'enciclopedia libera", http://en.wikipedia.org/, "4-bit carry lookahead adder")

Nella simulazione in PSpice, il segnale che percorre il tragitto più lento, si trova ad attraversare le seguenti porte:

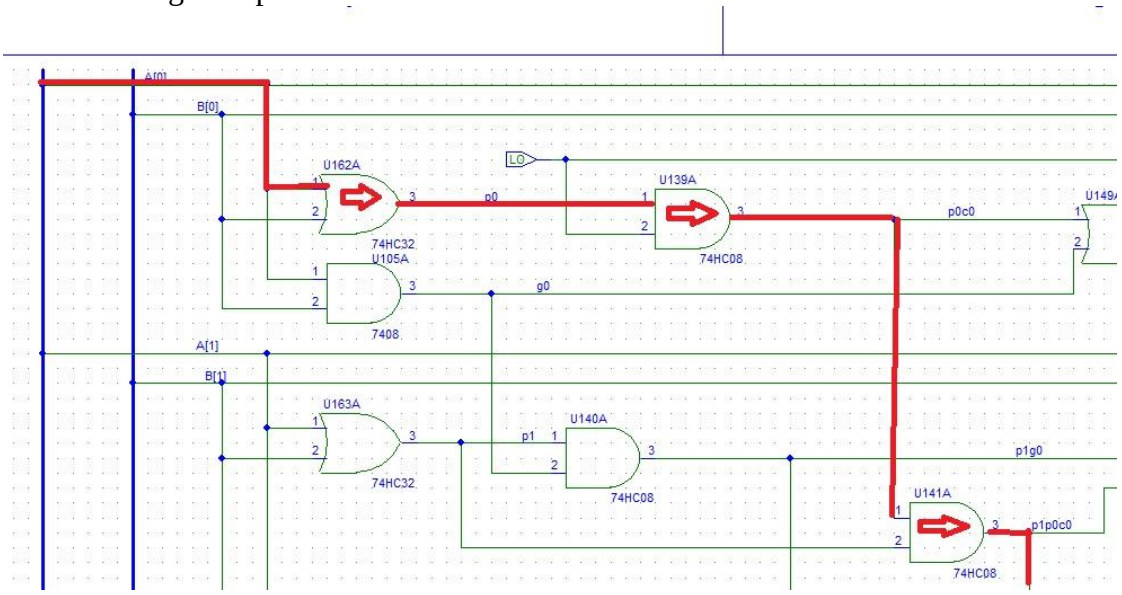

Figura 2.5 *Prima parte del percorso più lento per un segnale che attraversa il sommatore CLA*

Una prima porta OR(10ns), una AND(10ns) e un'altra AND(10ns)

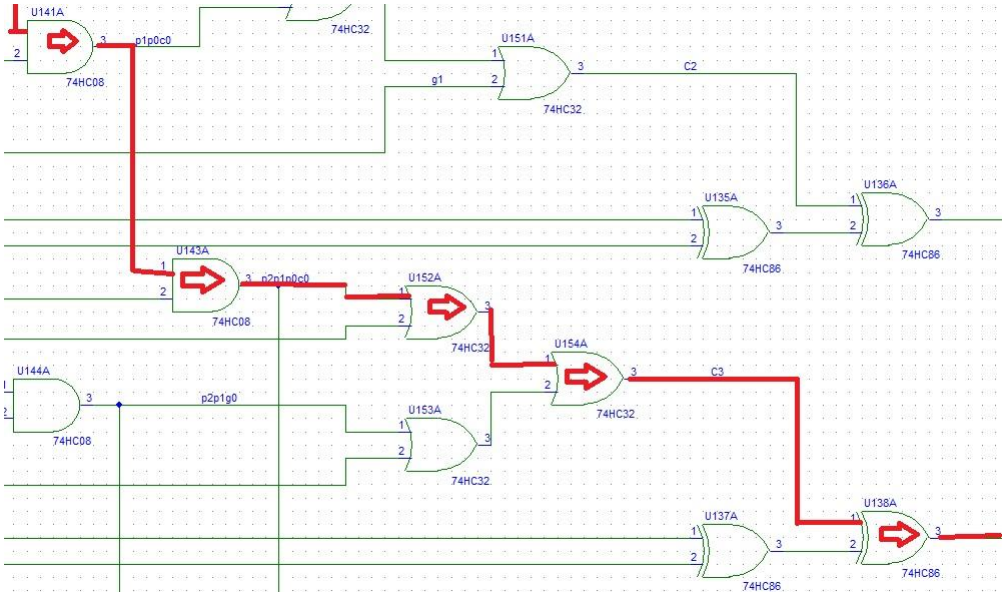

Figura 2.6 *Seconda parte del tragitto più lento per un segnale che attraversa il sommatore CLA*

Successivamente percorre una AND(10ns), OR(10ns), OR(10ns) e XOR(12ns).
Il ritardo complessivo cosi accumulato ammonta a 72 ns, valore verificato ancora una volta nelle simulazioni in PSpice (Fig. 2.7).
La realizzazione di un sommatore mediante CLA, dimostra vantaggi indiscutibili in termini di propagazione del carry, rispetto a una struttura RCA, ma bisogna utilizzarlo solo con un numero modesto di bit, o diventa necessario adottare alcuni accorgimenti.
A livello ideale, all'aumentare dei bit utilizzati, bisognerebbe disporre di porte con un numero sempre maggiore di ingressi, o di un circuito per il calcolo del riporto sempre più complesso.
In entrambi i casi, verrebbe a meno la velocità del sommatore, in quanto porte più grandi implicano tempi maggiori di propagazione, come circuiti più complessi.

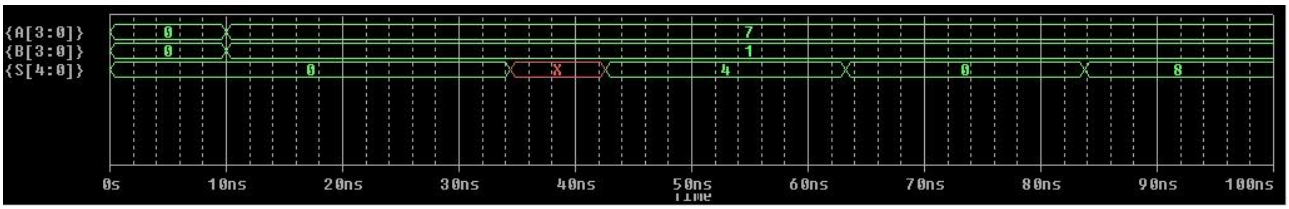

Figura 2.7 *Ritardo uscite, simulazione di un sommatore a 4 bit*

2.3 Sommatore 16 bit

Realizzare sommatori a più di 4 bit e con Carry-Look Ahead, comporta una complessità del circuito di calcolo del riporto sempre maggiore al crescere del numero di bit utilizzati a meno che non si introduca un'unità di Carry-Look Ahead di livello superiore.
Consideriamo il caso della realizzazione di un sommatore a 16 bit composto da quattro sommatori a 4 bit CLA che rappresentano i blocchi elementari.
A tale scopo, definiamo i Superpropagatori e Supergeneratori come input della nuova unità di Carry-Look Ahead di livello superiore.
Si può scrivere in generale che:

$Cout=G+CinP$ (25)

Se consideriamo di utilizzare 4 adder

$Cout=G_i+P_i G_{i-1}+P_i P_{i-1}G_{i-2}+P_i P_{i-1}P_{i-2}G_{i-3}+P_i P_{i-1}P_{i-2}P_{i-3} Cin$ (26)

$Gcomposto=G_i+P_i G_{i-1}+P_i P_{i-1}G_{i-2}+P_i P_{i-1}P_{i-2} G_{i-3}$ (27)

$Pcomposto= P_iP_{i-1}P_{i-2}P_{i-3}$ (28)

Ogni sommatore a 4 bit propaga un eventuale Cin solo se ciascuno dei bit del gruppo propaga un riporto.
I Superpropagatori sono:

$P0 = p3p2p1p0$ (29)

$P1 = p7p6p5p4$ (30)

$P2 = p11p10p9p8$ (31)

$P3 = p15p14p13p12$ (32)

Un sommatore a 4bit genera un riporto se il riporto viene generato in una certa posizione, e viene propagato in tutte le posizioni intermedie.
I Supergeneratori sono:

$G0 = g3 + (p3g2) + (p3p2g1) + (p3p2p1g0)$ (33)

$G1 = g7 + (p7g6) + (p7p6g5) + (p7p6p5g4)$ (34)

$G2 = g11 + (p11g10) + (p11p10g9) + (p11p10p9g8)$ (35)

$G3 = g15 + (p15g14) + (p15p14g13) + (p15p14p13g12)$ (36)

I Superpropagatori e Supergeneratori possono essere usati, analogamente ai propagatori e generatori di ordine inferiore, per calcolare i vari riporto in ingresso a ciascun blocco elementare :

$C1 = G0 + P0C0$ (37)

$C2 = G1 + (P1G0) + (P1P0C0)$ (38)

$C3 = G2 + (P2G1) + (P2P1G0) + (P2P1P0C0)$ (39)

$C4 = G3 + (P3G2) + (P3P2G1) + (P3P2P1G0) + (P3P2P1P0C0)$ (40)

In un sommatore cosi formato, i riporti dei blocchi a 4 bit provengono dall'unità di Carry-LookAhead esterna che lavora in base ai vari Pi e Gi, e di C0.
Il meccanismo del Carry LookAhead rende il calcolo dei riporti più veloce, perché riduce il numero di porte che i segnali devono attraversare per calcolare la somma.

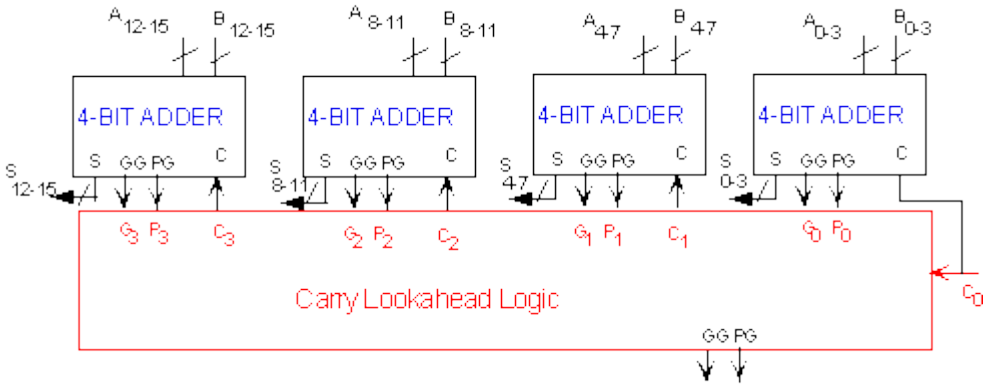

Figura 2.8 *Schema logico di un sommatore 16bit Carry-Look Ahead. (http://www.pldworld.com," Block diagram of a 16-bit CLA Adder")*

Procedere con questo metodo, implica una logica su più livelli all'aumentare del numero di bit utilizzati.
Per ovviare alla complessità di realizzazione di tale logica in grado di predire il resto durante la somma di due numeri con un sempre maggiore numero di bit, possiamo più semplicemente sfruttare il risparmio che comporta ogni singolo adder CLA a 4bit rispetto a un RCA e poi usarli in cascata.
Passando alla realizzazione in PSpice, possiamo stimare un tempo massimo d'attesa per l'uscita del sommatore cosi realizzato, di 232ns.
La struttura che si vede in Fig. 2.9 è composta da quattro blocchi contenenti ognuno un sommatore a 4bit CLA come quelli studiati nel Capitolo 2.2.

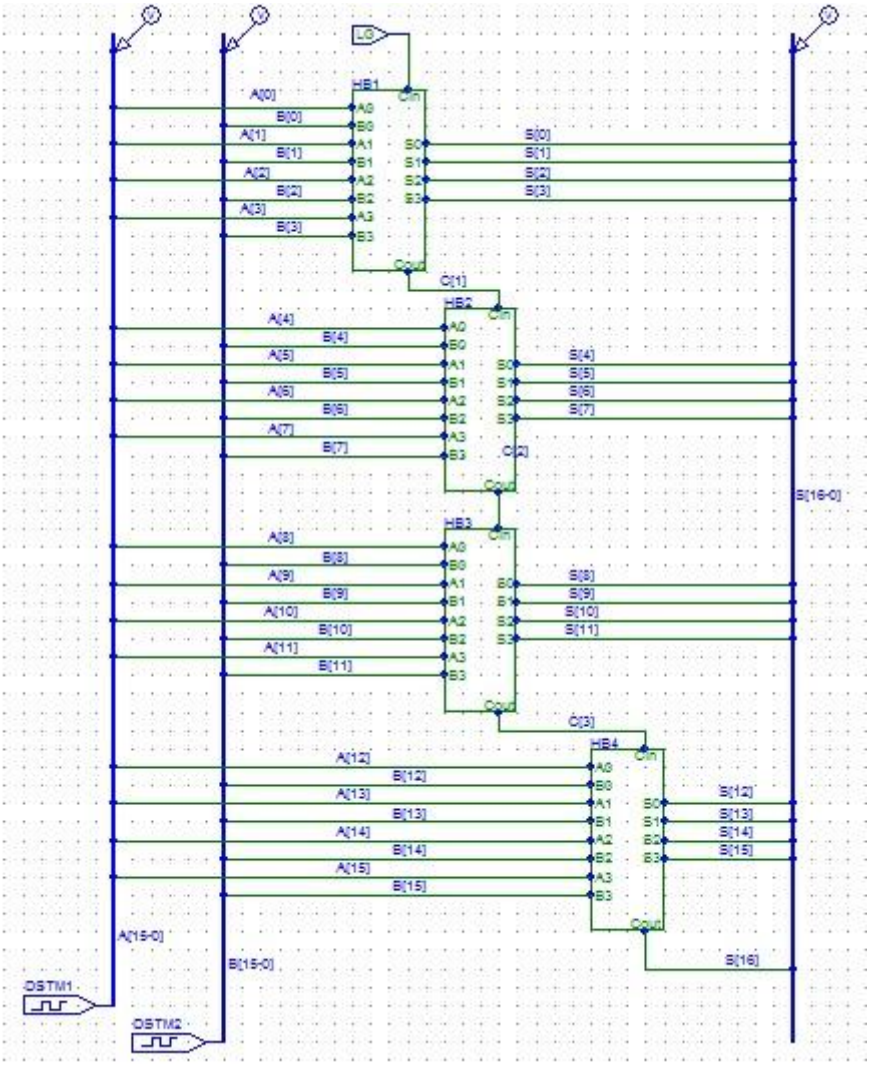

Figura 2.9 *Rappresentazione di un sommatore 16 bit basato su quattro sommatori a 4bit CLA*

Questo valore, seppur stimato, è inferiore al 4*92 ns = 368 ns che si otterrebbe usando quattro sommatori Ripple-Carry in cascata.

2.4 Considerazioni sui ritardi

Si può concludere che un sommatore di tipo RCA è di più facile intuizione e realizzazione rispetto alla sua controparte CLA, ma generalmente impiega un tempo maggiore a fornire i propri risultati a causa della lenta propagazione del resto attraverso tutti i suoi sommatori elementari.
Un adder CLA invece, quando si lavora con pochi bit riesce a mostrare un discreto guadagno in termini di tempo, fino a diventare nettamente più vantaggioso al crescere dei bit impiegati.
Come accennato in precedenza, il limite maggiore è dovuto al fatto che non si possono realizzare porte a molti ingressi e nello stesso tempo veloci, da impiegare nella rete logica per il calcolo del riporto.
Anche il caso in cui si dovesse realizzare una logica per la gestione del carry di un sommatore a molti bit non divisa su più livelli, potrebbe portare a una paradossale perdita di prestazioni di un CLA rispetto alla versione RCA.

3.1 Teoria alla base del moltiplicatore

Ci sono diversi modi e tecniche per risolvere la moltiplicazione in binario tra due parole a 4bit che consentono di ottenere medesimi risultati ma con ritardi e numero di porte impiegate differenti.
Consideriamo per cominciare la matrice dei prodotti parziali.

				x3	**x2**	**x1**	**x0**
				y3	**y2**	**y1**	**y0**
				x3y0	x2y0	x1y0	x0y0
			x3y1	x2y1	x1y1	x0y1	
		x3y2	x2y2	x1y2	x0y2		
	x3y3	x2y3	x1y3	x0y3			
S7	S6	S5	S4	S3	S2	S1	S0

Essa è uguale per tutti i moltiplicatori realizzabili.
Il passo successivo consiste nella somma dei prodotti parziali.
Noi prenderemo in esame il caso in cui otteniamo i vari Si sommando i prodotti parziali presenti su ogni colonna, operando sempre con porte a due singoli ingressi.
Questo metodo è analogo al modo di procedere quando si esegue una moltiplicazione senza il supporto di calcolatori, eseguendo sempre operazioni elementari.

3.2 Realizzazione del moltiplicatore

Per realizzare un moltiplicatore ad array sono necessari dei sommatori da N bit quanti sono quelli che compongono gli ingressi.
Si creano dei blocchi elementari in grado di svolgere il prodotto parziale tra i bit delle parole in ingresso utilizzando delle semplici porte AND.
Successivamente, sfruttando dei sommatori a 4 bit realizzati con Carry-Look Ahead collegati in cascata tra loro, si realizza la somma di coppie di prodotti parziali fino ad ottenere la somma totale delle colonne.
Per la realizzazione di un moltiplicatore di questo tipo, sono state utilizzate un totale di 124 porte logiche, 36 per ogni sommatore a 4bit e quattro porte per ogni blocco che svolge il prodotto parziale tra i bit di ingresso.

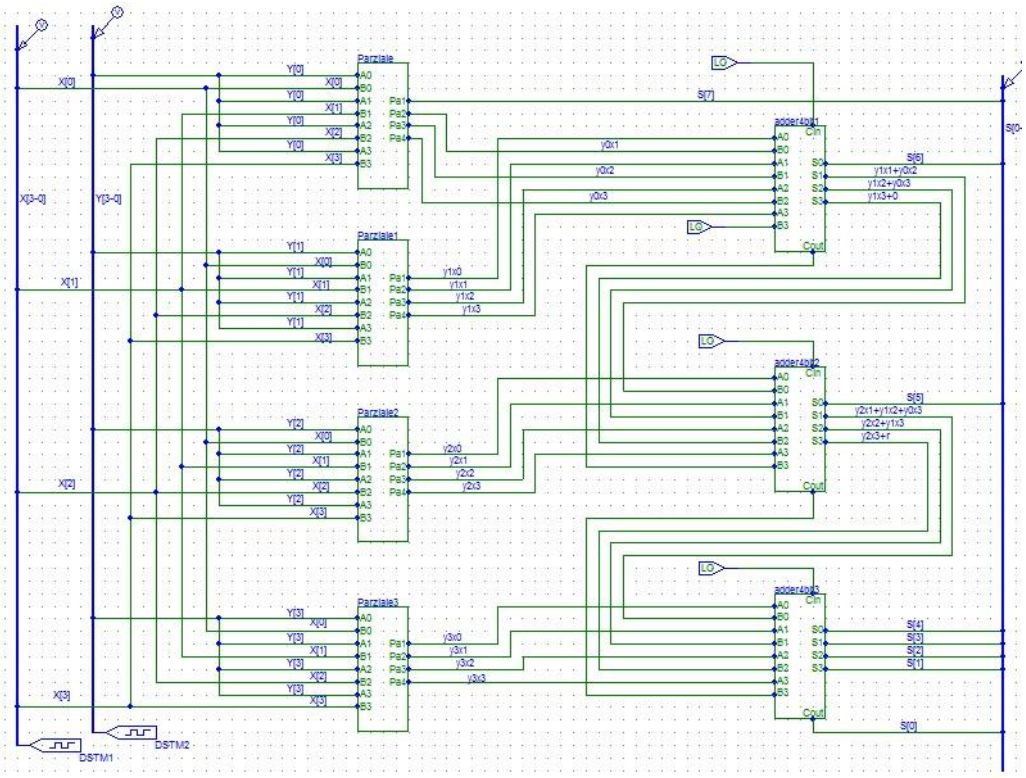

Figura 3.1 *Schema circuitale di un moltiplicatore a 4bit in SPICE*

3.3 Considerazioni sul moltiplicatore

Il moltiplicatore cosi realizzato è scarsamente ottimizzato, e non sfrutta neppure appieno tutte le porte disponibili.
Come si può notare, non viene mai utilizzato il Cin nei blocchi adder a 4bit e l'ingresso B3 del primo sommatore.
Per ovviare al primo problema, sarebbe sufficiente sostituire il primo Full-Adder di ogni sommatore a 4bit con un Half-Adder.
Ora procediamo alla stima del tempo massimo di percorrenza per un segnale che attraversa il moltiplicatore.
L'ipotetico segnale in ingresso, attraversa il primo blocco "parziale", dove viene fatto il prodotto parziale tra due bit in ingresso (10ns dovuto alla porta AND), poi attraversa il primo adder a 4bit in modo da scegliere il caso che realizzi il tempo peggiore di percorrenza (72ns):

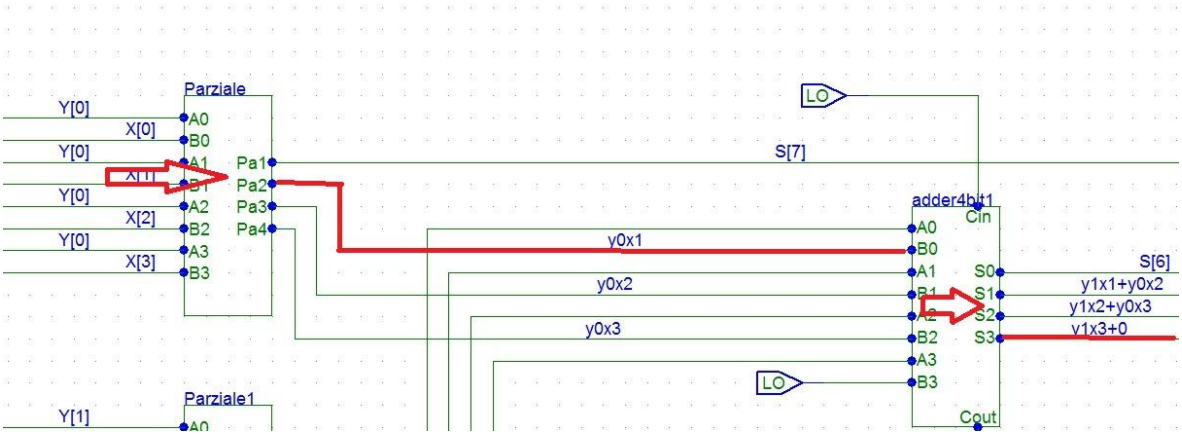

Figura 3.2 *Prima parte del tragitto più lungo che percorre un segnale per attraversare il moltiplicatore*

Il segnale, una volta uscito dal primo sommatore dalla porta S2', entra nel secondo sommatore alla porta B2' per poi uscire come resto del blocco Cout' (ulteriore tempo di percorrenza di 60ns)

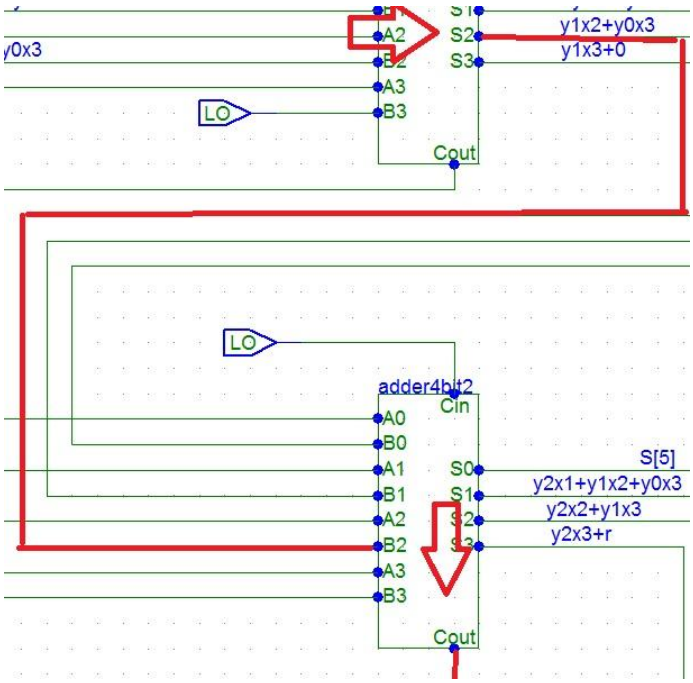

Figura 3.3 *Continuo del tragitto per attraversare il moltiplicatore*

Infine, il segnale uscito come Cout' entra nell'ultimo sommatore dalla porta B3'' ed esce come Cout'' accumulando altri 50 ns di ritardo massimo possibile.

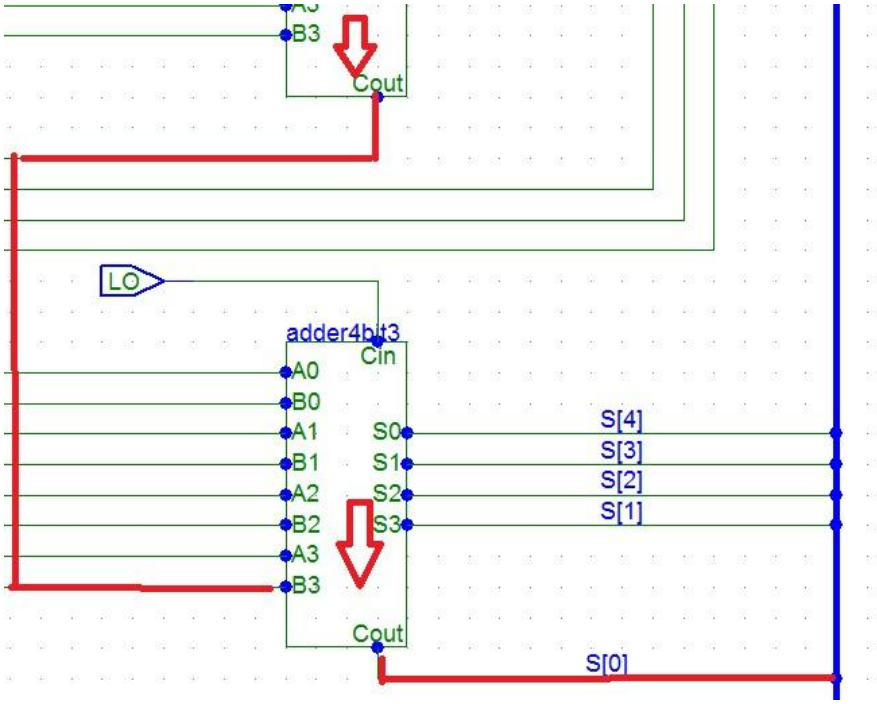

Figura 3.4 *Ultima parte del tragitto che il segnale percorre*

Il tempo massimo stimato impiegato dal segnale per attraversare tutto il moltiplicatore è di: 10+72+60+50 = 192 ns, contro un tempo massimo misurato mediante simulazione PSpice di 155ns (Fig.3.5).

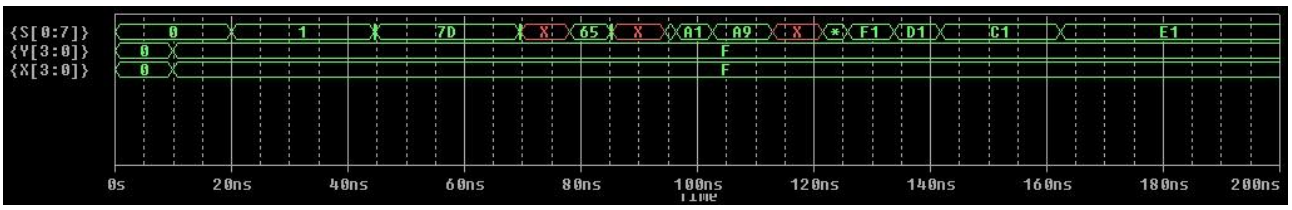

Figura 3.5 *Uscita moltiplicatore a 4 bit*

Questa differenza è dovuta all'elevato numero di porte utilizzato, che commutando mediamente a tempi inferiori al caso peggiore, comportano nella pratica a un risparmio di tempo rispetto al caso peggiore.
Inoltre è da considerare che difficilmente si verifica il caso peggiore in cui il segnale percorre proprio il percorso più lungo e ogni altra situazione quindi, porterebbe a un tempo misurato inferiore.

4.1 Teoria alla base del moltiplicatore di Wallace

Il moltiplicatore di Wallace è un'ottimizzazione del normale moltiplicatore, realizzata per risparmiare il numero di porte utilizzato e di conseguenza, i costi di realizzazione, lo spazio sul silicio e come vedremo anche il tempo massimo del ritardo misurato.
Il principio fondamentale alla base del moltiplicatore di Wallace è la riduzione in livelli e lo sfruttamento di contatori in parallelo.
Per prima cosa è necessario creare la matrice dei prodotti parziali, poi mediante l'utilizzo di soli contatori paralleli si riducono i livelli fino a quando diventa possibile concludere la moltiplicazione con un solo sommatore a 4bit.
È necessario collegare il più possibile con FA o HA gli ingressi di ogni colonna ricordando che le uscite S rimangono sulla stessa colonna, mentre le uscite C vanno sulla colonna successiva.
Notare che le colonne con un solo ingresso forniscono direttamente il valore anche del livello successivo.
Per la riduzione utilizzo inizialmente due HA e tre FA:

				x3 **y3**	**x2** **y2**	**x1** **y1**	**x0** **y0**
				x3y0	x2y0	x1y0	x0y0
			x3y1	x2y1	x1y1	x0y1	
		x3y2	x2y2	x1y2	x0y2		
	x3y3	x2y3	x1y3	x0y3			
	x3y3 R''''	S'''' R'''	S''' R''	x3y0 S'' R'	S' R	S1	S0

Ora, nel secondo livello utilizzo quattro HA e un FA:

	x3y3 R''''	S'''' R'''	S''' R''	x3y0 S'' R'	S' R	S1	S0
R''''	S'''' R'''	S''' R''	S'' R'	S' R	S2	S1	S0

A questo punto è possibile svolgere le ultime somme utilizzando un solo sommatore a 4 bit CLA

4.2 Realizzazione del moltiplicatore di Wallace

Per la realizzazione dei prodotti parziali, si utilizzano blocchi che eseguono l'operazione AND tra i vari bit.
Una volta disponibili tali prodotti, utilizzando soltanto contatori Full-Adder e Half-Adder si riduce la matrice a solo due righe.
Ovviamente il Full-Adder per tre ingressi presenta due uscite, mentre l'Half-Adder ha due ingressi e due uscite.
Una volta fatte tutte le riduzioni, si utilizza un sommatore a 4bit CLA, come quelli precedentemente visti per effettuare le ultime somme.
Nella simulazione in PSpice, vengono utilizzate in totale 85 porte contro le 124 del moltiplicatore precedente.

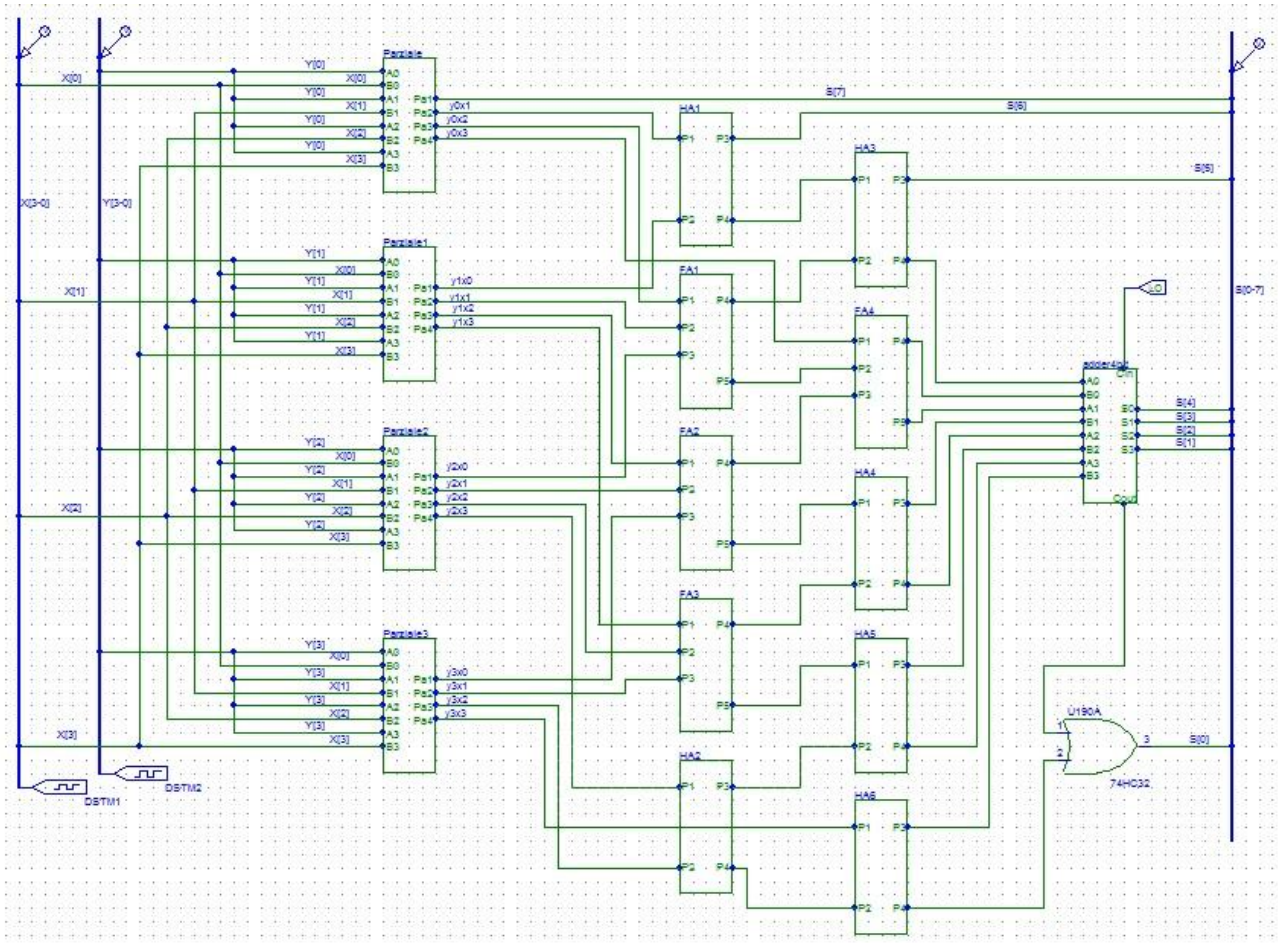

Figura 4.1 *Schema circuitale del moltiplicatore di Wallace*

Per il calcolo del percorso più lento, ipotizziamo il segnale che attraversato il "parziale" (10ns), entri nel primo Full Adder ed esca da P5' (32ns).

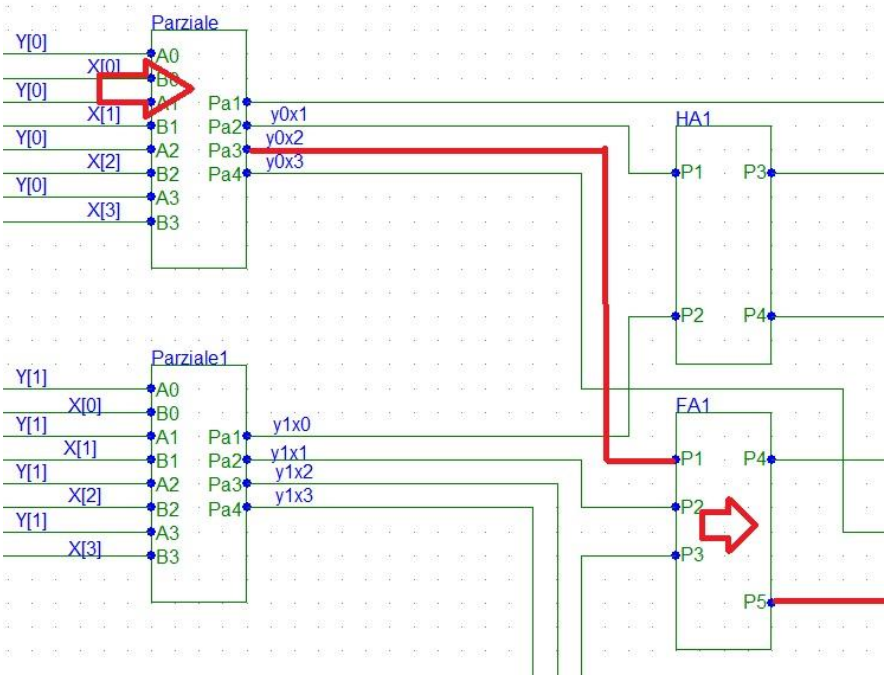

Figura 4.2 *Percorso più lento che attraversa il segnale*

Successivamente entri nel quarto Full Adder da P2'''' ed esca da P5'''' (accumulo 20ns) per poi attraversare l'adder a 4 bit (ulteriori 70ns) e infine passi per l'ultima porta OR (10ns).

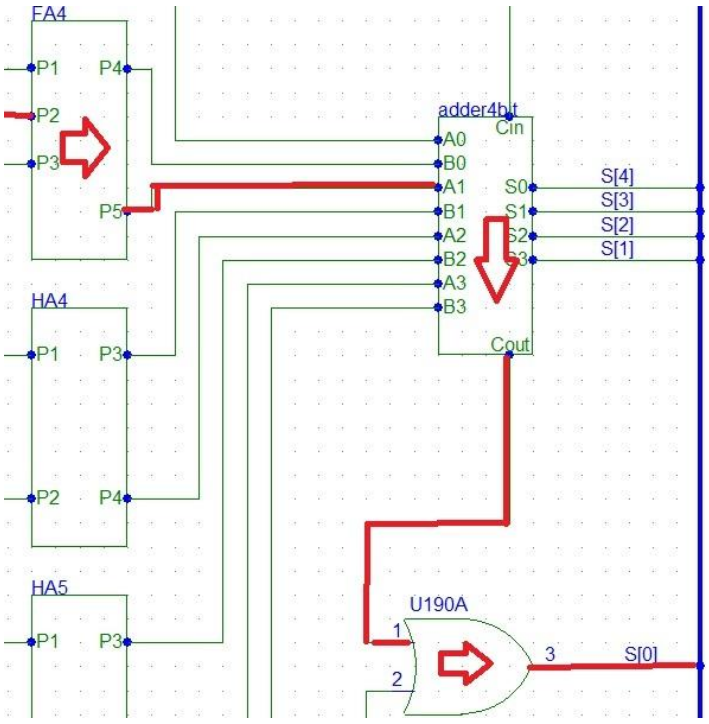

Figura 4.3 *Seconda parte del percorso più lento che attraversa il segnale*

Il ritardo complessivo cosi stimato è di 142ns per un ritardo misurato simulando in PSpice di 115ns.
Ancora una volta la stima è accettabile se confrontata con i risultati ottenuti.

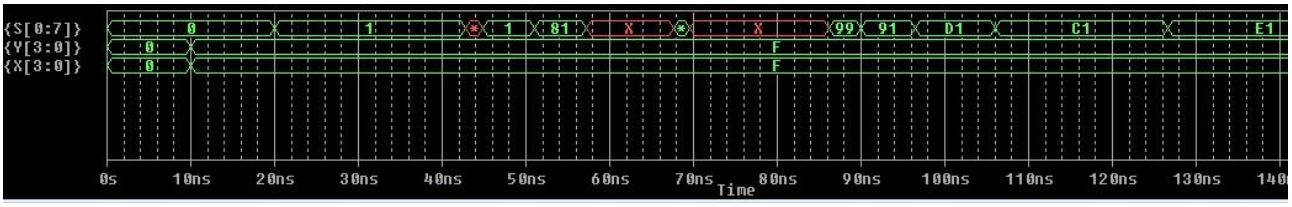

Figura 4.4 *Uscite della simulazione del Moltiplicatore di Wallace*

4.3 Considerazioni sul moltiplicatore di Wallace

Il moltiplicatore di Wallace comporta dei pregi e difetti legati proprio alla sua filosofia.
Per prima cosa si riscontra un risparmio delle porte da 124 a 85.
Questo risparmio dovuto a una logica di base migliore e più ottimizzata, si traduce anche in risparmio di tempo che nel nostro caso passa da 192ns a 142ns.
Il valore poteva anche essere proporzionalmente maggiore se avessimo aumentato il numero di bit delle parole con cui eseguire la moltiplicazione.
Per contro, il moltiplicatore di Wallace è di più difficile realizzazione e in base alla sua struttura e logica di realizzazione, non presenta simmetrie.

Conclusioni

In questo elaborato, è stato studiata la realizzazione di un moltiplicatore di Wallace mediante il programma di simulazione PSpice.
Confrontando i valori ottenuti dalle simulazioni, durante le fasi di progetto, con quelli delle prove simulate, si nota nella maggior parte dei casi, che i valori coincidono.
Questo è dovuto al fatto che PSpice è un programma molto affidabile, che simula fedelmente i casi che si è andati a studiare.
I casi in cui ci sono state discrepanze, sono dovuti soprattutto al fatto che non si è riusciti a studiare una simulazione adeguata che simulasse effettivamente il caso peggiore ottenibile.
Oltre a questo ovviamente, bisogna sempre considerare, che all'aumentare del numero delle porte, aumenti sempre anche la possibilità che qualche valore cambi leggermente.
La simulazione del moltiplicatore di Wallace, ha dimostrato come sia possibile migliorare un componente addetto a svolgere una determinata operazione non esclusivamente aumentando le performance delle singole porte, ma anche migliorato la progettazione e la struttura stessa.
In questo caso particolare, l'operazione di ottimizzazione, ha portato anche una riduzione del numero di porte e di spazio occupato sul silicio, che sarebbe traducibile in un risparmio di risorse una volta che si andasse a realizzare fisicamente il moltiplicatore.
Oltre alla versione di Wallace, sono disponibili altri metodi di ottimizzazione a seconda del campo di utilizzo e delle esigenze particolari che si sceglie di studiare, per realizzare un moltiplicatore in ogni singolo caso più o meno efficace.

Nomenclatura

τ	=	ritardo
V	=	Volt
ns	=	Nano secondi
CLA	=	Carry Look Ahead
RCA	=	Ripple Carry Adder
HA	=	Half Adder
FA	=	Full Adder
X’	=	Numero negato, nell’algebra Booleana

Elenco delle figure

Ringraziamenti

A conclusione di questa parte del mio percorso di studio, volevo ringraziare le persone che mi hanno aiutato in questi anni.
Vorrei ringraziare il Professor Simone Buso che si è sempre dimostrato disponibile nei miei confronti seguendomi e aiutandomi quando servivano aiuti o chiarimenti.
Un altro ringraziamento và ai miei amici dell'università, da quelli con cui sono partito a quelli con cui ho finito, compagni di lunghe giornate di studio e ma anche di svago.
In particolare vorrei ringraziare la mia famiglia, che è sempre stata al mio fianco ad aiutarmi e sostenermi durante questi anni, senza mai farmi mancare la loro fiducia.
Per finire un speciale ringraziamento lo faccio alla mia ragazza, senza la quale probabilmente non sarei riuscito ad essere e fare ciò che sono.

Bibliografia

S. Buso, “Teoria dei Circuiti Digitali”, “2009-2010.
G. Manduchi, “Calcolatori elettronici”, “2008-2009.
“Wikipedia, l’enciclopedia libera”, http://en.wikipedia.org/, ”Carry-lookahead_adder”, 22/8/12
http://www.pldworld.com, “ Digital Design Laboratory”, ultimo accesso 25/8/12
http://www.pldworld.com,” Carry-Look-ahead Adder”, ultimo accesso 25/8/12
M.Loghi, http://www.diegm.uniud.it,”Reti logiche”, versione 2011/12
M.Palesi, http://www.diit.unict.it, “Progetto di circuiti aritmetici”, ultimo accesso 20/8/12
F.Ferrandi, http://home.dei.polimi.it, “Architetture aritmetiche”, versione 11/11/03
M.Rovini, http://vlsi.iet.unipi.it, “Tempo di propagazione di una rete combinatoria”, versione 2007
A.Zanchi, tesi di laurea “Decimatore digitale a 200 MegaHertz per A/D Sigma-Delta: dall’architettura al Layout”, anno accademico 1994/95

Immagini:

Fig.2.4) “Wikipedia, l’enciclopedia libera”, http://en.wikipedia.org/, “4-bit carry lookahead adder”, 25/8/12
Fig.2.8) http://www.pldworld.com,” Block diagram of a 16-bit CLA Adder”, 25/8/12