Progetto Individuale per la sessione estiva a.a. 2014/2015

Angelo Sasso

Matricola 263143

Introduzione e illustrazione dei moduli principali

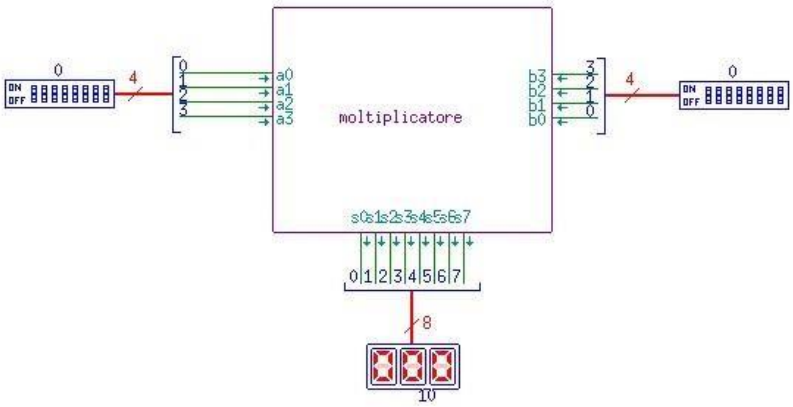
Il progetto realizzato consiste nella creazione di un moltiplicatore a 4 bit, implementato tramite il programma “Tkgate”.

Il progetto sfrutta una serie di AND per ottenere i prodotti parziali che, inseriti nei Ripple-Carry-Adder, forniscono il risultato finale. Il'RCA (addizionatore a propagazione del riporto) sfrutta una serie di Half Adder e Full Adder che propagano in orizzontale il resto e in verticale effettuano le somme tra i vari prodotti parziali.

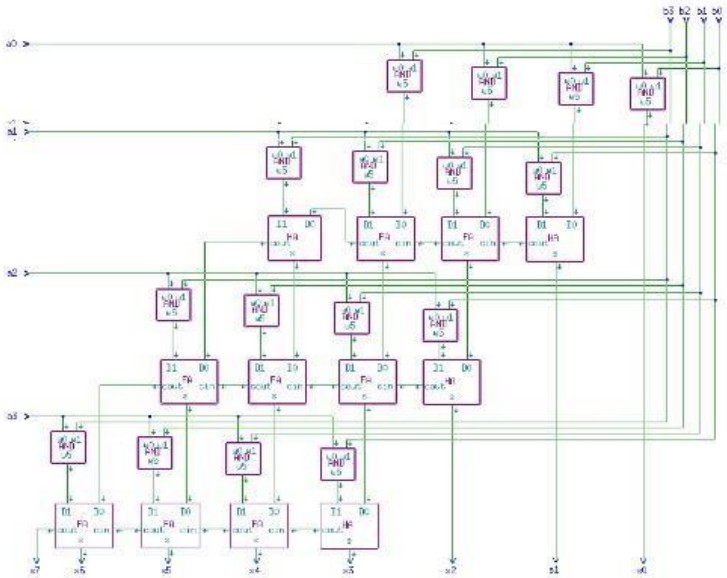
Avendo a disposizione 4 bit d'ingresso, i numeri che si possono rappresentare vanno da 0 a 15, mentre in uscita, avendo 8 bit a disposizione, risultano fino a 225. Sia in uscita che in entrata i numeri sono rappresentati in base esadecimale, quindi fanno uso di 16 simboli che sono le dieci cifre decimali e le lettere dalla a alla f.

Seguono i due moduli principali che costituiscono il progetto:

- All'interno dell'interfaccia iniziale (***main***) sono presenti i 2 interruttori, nei quali l'utente inserirà i dati in ingresso; tali dati vengono prelevati dal moltiplicatore, che produrrà il risultato visibile tramite un LED a base decimale.



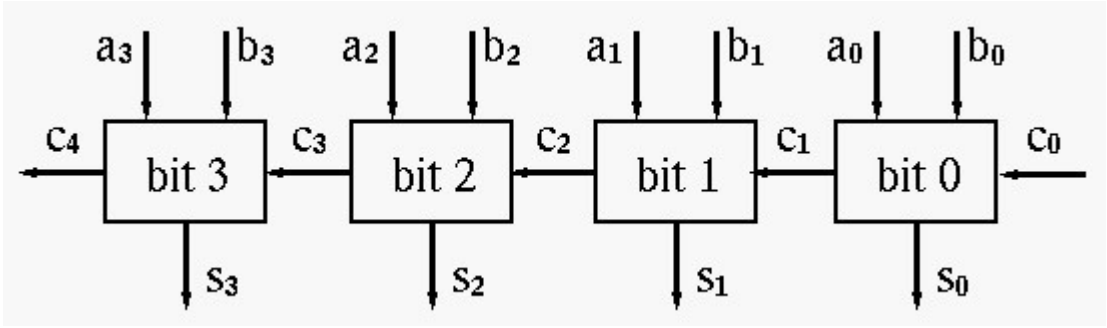
- Nell'interfaccia successiva, il ***moltiplicatore*** combina i bit in entrata eseguendo moltiplicazioni parziali tramite una serie di AND; tali risultati vengono successivamente inviati a una serie di Ripple Carry Adder, che tramite un'opportuna combinazione di Full e Half Adder, sommano i vari risultati parziali per ottenere la soluzione.



Le componenti principali sfruttate all'interno del progetto

1.Ripple-Carry Adder (RCA)

Il metodo più diretto per realizzare un addizionatore ad *n* bit è rappresentato dal Ripple-Carry Adder (RCA), o addizionatore a propagazione del riporto.
Questo circuito richiede *n* Full-Adder in cascata, ovvero il riporto uscente dell'*i-esimo* Full-Adder viene collegato al riporto entrante dell'(*i+1*)-*esimo* Full-Adder.
La sua architettura è semplice, ma anche lenta. Il tempo di calcolo nel caso pessimo, infatti, dipende linearmente dal numero di stadi, in quanto bisogna attendere che il riporto si propaghi dalla prima all'ultima cella per avere il risultato corretto.



Se si indica con Tp(FA) il tempo di propagazione di un Full-Adder, il tempo di propagazione di un RCA vale, nel caso pessimo, Tp(RCAn) = n Tp(FA) = O(n).

Invece il tempo di contaminazione(Tc) di un RCA vale: Tc(RCAn) = Tc(FA) = O(1).

In maniera simile, il numero di porte sfruttate dall'RCA ne determina l'area che risulta essere: A(RCAn) = n A(FA) = O(n).

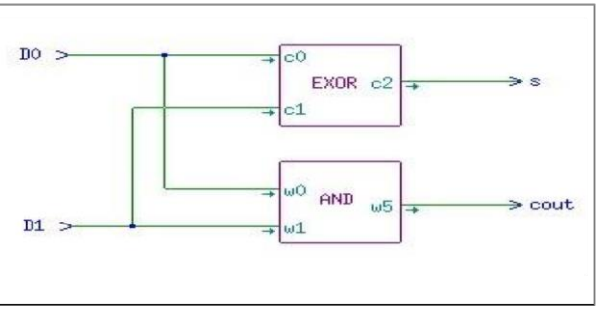
Infine, tramite il tempo di propagazione, possiamo ricavare il rate o throughput nel seguente modo:

Rate(RCAn) < 1/Tp(RCAn) = O(1/n).

2.Half Adder (HA)

L'half adder, detto anche semisommatore, è un componente elettronico digitale che esegue la somma di due bit senza tener conto di un riporto precedente; in uscita vengono calcolati il risultato s e il riporto cout. La somma binaria di due bit si calcola allo stesso modo della somma tra due numeri decimali, per cui il riporto viene a formarsi se e solo se i due addendi sono pari a 1. Il circuito è costituito da una porta AND e una porta XOR, come mostrato in figura nella pagina seguente.

a	b	s	cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



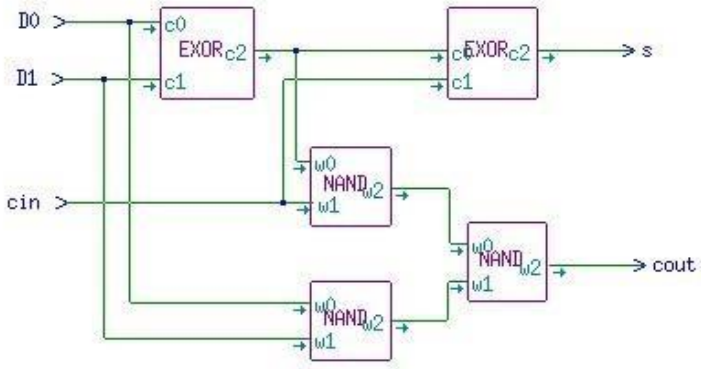
Gli HA sostituiscono il primo FA all'interno di ogni RCA poiché non presentano il riporto in ingresso; inoltre l'ultimo FA del primo RCA viene sostituito con l'HA perché presenta il risultato di un solo prodotto parziale oltre al resto in ingresso.

3. Full Adder (FA)

Il full-adder o sommatore completo è un componente elettronico digitale caratterizzato da tre ingressi e due uscite. La sua funzionalità è quella di eseguire una somma tra due numeri espressi in formato binario, tenendo conto di un eventuale riporto precedente.

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

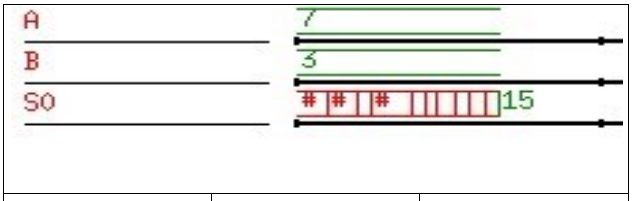
L'operazione eseguita è la seguente: a + b + cin = s + cout, dove a e b sono gli operandi, cin il riporto in ingresso della precedente somma e s e cout sono la somma e il riporto in uscita.
Il circuito è composto da due porte EXOR e tre porte NAND come mostrato in figura.



Testing del programma

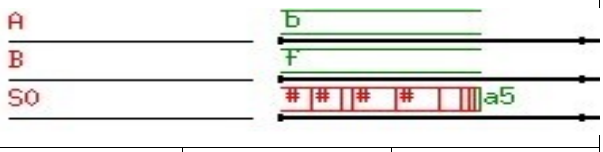
Qui di seguito sono riportati due esempi del funzionamento del moltiplicatore. A quest'ultimo vengono forniti due numeri in ingresso in base esadecimale, il moltiplicatore, in seguito, produce il risultato di tali numeri anch'esso in base esadecimale. Nella tabella sono riportati i rispettivi valori d'ingresso e d'uscita in base decimale e binaria e viene illustrato il cambio di base del risultato da esadecimale a decimale.

- Questo primo esempio prende in ingresso i valori $(7)_{16}$ e $(3)_{16}$, equivalenti ai rispettivi valori in decimale; invece, in uscita, avremo il valore $(15)_{16}$ che, come mostrato nella tabella, corrisponde a $(21)_{10}$.



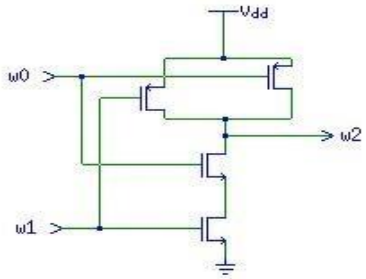
decimale	binario	esadecimale
7 * 3 =	0111 * 0011 =	7 * 3 =
21	0111 0111 0000 0000 00010101	15 $(15)_{16} = 5*16^0$ $+1*16^1 =$ $5+16 = (21)_{10}$

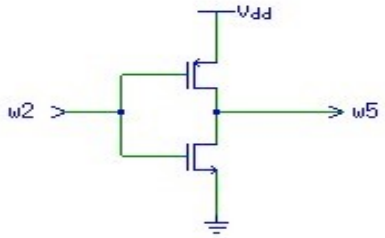
- Nell'esempio che segue abbiamo in ingresso i valori $(b)_{16}$ e $(f)_{16}$, equivalenti rispettivamente ai valori $(11)_{10}$ e $(15)_{10}$ in base decimale; in uscita, invece, abbiamo il valore $(a5)_{16}$ in base esadecimale che corrisponde al valore $(165)_{10}$ in base decimale.



decimale	binario	esadecimale
11 * 15 =	1011 * 1111 =	b * f =
55 11	1011 1011 1011 1011	a5 $(a5)_{16} =$ $5*16^0 + a*16^1 =$
165	10100101	$5+160=(165)_{10}$

Richiami alla tecnologia CMOS





Queste illustrazioni ci mostrano come sono state implementate a livello transistor rispettivamente le porte NAND e NOT. Entrambe sono basate su una tecnologia di costruzione CMOS.
Tutte le porte del tipo CMOS sono divise in due parti: la rete "pull-up", strutturata con transistor del tipo "p" e connessa alla sorgente; e la rete "pull down", costruita con transistor del tipo "n" e connessa a terra.
Le due parti sono duali logicamente l'una dell'altra, cosicché se la pull-up è attiva, allora la pulldown è inattiva, e viceversa. In questo modo non ci può essere mai un collegamento diretto tra la sorgente e la terra.

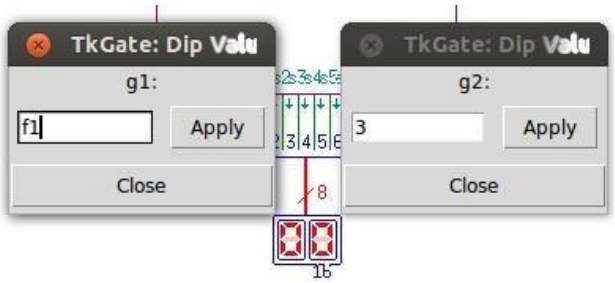
Note finali

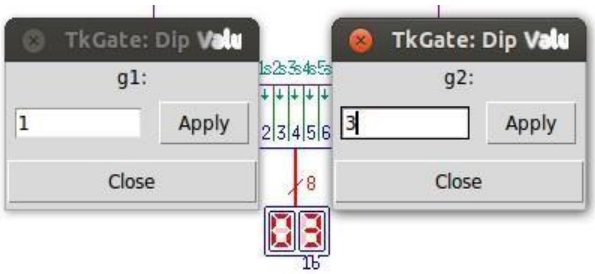
Questo paragrafo è un chiarimento sull'assenza di overflow. Come è stato già dichiarato sopra, i dati in ingresso hanno una dimensione pari a 4 bit e 8 bit per quelli in uscita, quindi il più grande numero rappresentabile tramite 4 bit è 15 (“f” in base esadecimale), che moltiplicato per 15 restituisce il valore 225 (“e1” in base esadecimale: $e1=e*16^1+1*16^0=14*16+1=225$). Avendo 8 bit, per i dati di output, non si presenta un errore di overflow perché il numero più grande rappresentabile con 8 bit è proprio 225.

L'unico problema potrebbe presentarsi nell'inserimento dei dati, in quanto se si dovesse superare il limite di “f” (15 in decimale) o si sbagliasse a scrivere la codifica del numero in esadecimale, il programma preleverà solo la cifra con peso minore. Seguono due esempi che denotano i tipi di errori elencati precedentemente:

1. Si può notare come il primo numero inserito sia maggiore al massimo numero rappresentabile ovvero “f”, perciò viene presa in considerazione solo la cifra 1 dando un risultato diverso da quello atteso.
2. Nel secondo esempio, invece, possiamo notare un errore nella codifica, in quanto il numero 13 in esadecimale va rappresentato con il carattere “d”, portando anche qui ad un risultato errato.

Primo esempio





Secondo esempio