POLITECNICO DI TORINO

Laurea Magistrale in Ingegneria Elettronica

Politecnico
di Torino

Tesi di Laurea Magistrale

Progettazione digitale di compressori approssimati 4:2 per interfacce seriali a 112Gps in tecnologia FinFET

Relatori	Candidato
Prof. Maurizio MARTINA	Stefano RIZZELLO
Prof. Guido MASERA	

Supervisori aziendali
STMicroelectronics

Piero BONIFACINO

Mauro NATUZZI

Aprile 2024

Sommario

La richiesta di comunicazioni ad alta velocità ha spinto allo sviluppo di interfacce sempre più veloci, che richiedono l'utilizzo di tecnologie avanzate e tecniche di progettazione innovative. In particolare, le interfacce seriali a 112Gbps sono sempre più diffuse nei data center e nelle reti di telecomunicazioni, dove la velocità di trasmissione dei dati è fondamentale per garantire prestazioni elevate e tempi di latenza ridotti. Tuttavia, la progettazione di interfacce seriali a queste velocità rappresenta una sfida tecnologica che richiede l'utilizzo di soluzioni avanzate. Nel contesto del SerDes, la riduzione del consumo di potenza rappresenta una performance critica. In particolare, l'equalizzatore digitale (FFE) del SerDes è la parte più grande e il principale contributore al consumo di potenza del sistema.
La tesi ha esplorato diverse soluzioni al fine di migliorare l'efficienza energetica del sistema.
Nella prima parte, si è analizzata l'implementazione esatta del filtro, valutando alcuni algoritmi presenti in letteratura e mostrando i risultati ottenibili. In particolare, sono stati esaminati i vantaggi e gli svantaggi di ciascuna soluzione, al fine di individuare quella più adatta alle esigenze specifiche dell'applicazione.
Nella seconda parte, si è invece valutato l'impatto di soluzioni approssimate per questo tipo di applicazione, concentrandosi sull'impiego di compressori 4:2 approssimati per l'albero delle somme, al fine di ridurre il consumo di potenza del sistema. Sono state esplorate tecniche di aritmetiche approssimata, che hanno dimostrato una evidente efficacia negli ultimi anni, valutando l'impiego di questi compressori. Sono stati condotti test di simulazione per valutare le prestazioni del filtro utilizzando 12 compressori approssimati, confrontandoli tra di loro e con quelli ottenuti con tecniche di progettazione tradizionali. I risultati hanno mostrato che alcuni compressori approssimati 4:2 sono più adatti rispetto ad altri, in base alle esigenze specifiche di velocità, potenza e accuratezza del filtro FFE.
Il lavoro svolto può essere utilizzato come base per lo sviluppo e l'ulteriore ottimizzazione di logiche digitali ad alte prestazioni, fornendo importanti indicazioni per migliorare l'efficienza energetica del sistema.

Indice

Elenco delle tabelle

Elenco delle figure

Elenco delle sigle

IoT
Internet of Things

FFE
Feed Forward Equalizer

FIR
Finite Impulse Response

PAM4
Pulse Amplitude Modulation 4-level

NRZ
Non Return to Zero

BER
Bit Error Rate

FEC
Forward Error Correction

PLL
Phase Locked Loop

CDR
Clock Data Recovery

ISI
Inter Symbol Interference

DFE

Decision Feedback Equalizer

FBF

Feed Back Filter

IIR

Infinite Impulse Response

MSB

Most Significant Bit

ASIC

Application Specific Integrated Circuit

SD

Signed Digit

LSB

Least Significant Bit

MBE

Modified Booth Encoding

PP

Partial Product

RA

Reduced Area

FA

Full Adder

HA

Half Adder

ER

Error Rate

ED

Error Distance

PPM

Partial Product Matrix

SNR

Signal to Noise Ratio

Capitolo 1

Introduzione

In un mondo sempre più connesso, dove la tecnologia ci permette di comunicare con persone in ogni parte del mondo in tempo reale, l'importanza di una comunicazione veloce ed efficace diventa sempre più evidente.
La costante crescita del traffico di dati e dei servizi digitali, come l'archiviazione e l'elaborazione in cloud, i big data e l'Internet of Things (IoT), sta determinando una continua accelerazione della larghezza di banda e dell'innovazione tecnologica. In Figura 1.1, l'andamento del traffico dei data center nel periodo 2016-2021, come presentato in [1].

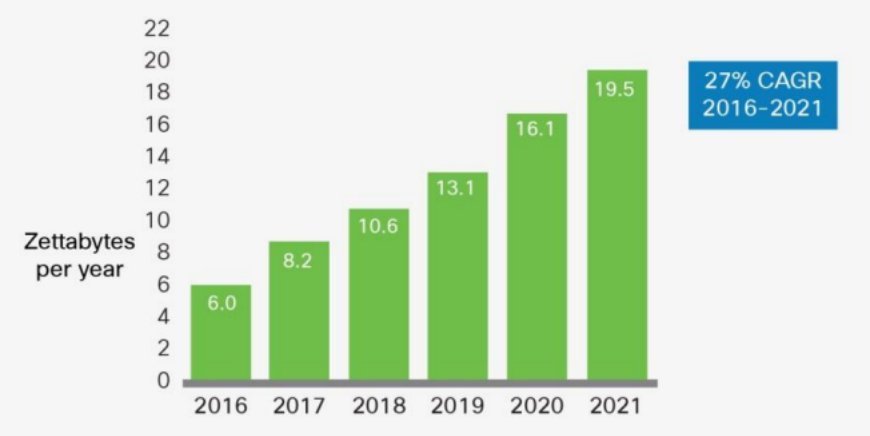


Figura 1.1: Traffico dei Data Center su scala globale

Con l'utilizzo di un dispositivo, chiamato SerDes (Serializer – Deserializer), un'interfaccia che permette di migliorare la velocità delle comunicazioni e lo scambio di informazioni in ambito elettronico, è possibile trasmettere grandi quantità di dati su un singolo canale, riducendo il numero di connessioni necessarie e migliorando l'efficienza del sistema.

Questa tecnologia è particolarmente indicata per applicazioni che richiedono una comunicazione ad alta velocità, come le reti di comunicazione, i sistemi di elaborazione dati e le applicazioni di trasmissione video ad alta definizione.
Inoltre, garantisce una trasmissione affidabile dei dati su lunghe distanze, migliorando la qualità della comunicazione.
Il SerDes utilizza diversi meccanismi per compensare gli effetti negativi della trasmissione del segnale su un canale di comunicazione, tra cui l'equalizzazione. Questa tecnica aiuta a correggere le varie alterazioni che possono verificarsi durante la trasmissione, come l'attenuazione del segnale, il rumore o la dispersione del segnale.
L'equalizzazione nel SerDes può essere realizzata utilizzando diversi tipi di equalizzatori, come l'equalizzatore lineare, l'equalizzatore decisionale, l'equalizzatore adattativo e l'equalizzatore di feed-forward. Questi equalizzatori utilizzano filtri per compensare le distorsioni del segnale e migliorare la qualità della comunicazione.
In particolare, l'equalizzatore di feed-forward (FFE) utilizza un filtro per prevedere le distorsioni del segnale e compensarle in anticipo. Il filtro dell'equalizzatore di feed-forward può essere realizzato con un filtro FIR (Finite Impulse Response).
L'utilizzo di un filtro FIR può presentare alcuni problemi, tra cui la dimensione del filtro e il consumo di potenza. Tuttavia, questi problemi possono essere mitigati mediante una progettazione adeguata del filtro.

1.1 Obiettivi della tesi e struttura

L'obiettivo della tesi consiste nella valutazione dei vantaggi e degli svantaggi associati alle diverse implementazioni dei moltiplicatori all'interno del filtro FIR e dell'albero di compressione, il quale somma tutti i prodotti parziali in uscita dai moltiplicatori.
Diverse configurazioni di queste architetture sono state elaborate utilizzando Verilog e SystemVerilog. Successivamente, attraverso il processo di sintesi con tecnologia FinFET di ultima generazione, si sono potuti confrontare parametri cruciali quali la frequenza operativa massima, l'ingombro di area e il consumo energetico. L'analisi di questi dati ha fornito le basi per determinare la variante più efficiente tra quelle esaminate, in vista del loro impiego in un SerDes.
Dopo questo capitolo di introduzione, la tesi si struttura nei seguenti capitoli:

2. Filtro esatto: Nel capitolo in questione, vengono valutate diverse implementazioni di moltiplicatori e compressori dell'albero delle somme, al fine di ottenere un'uscita accurata dal filtro. Inizialmente, viene utilizzata una combinazione di moltiplicatori e somme eseguite attraverso il segno di somma e il segno della moltiplicazione. Successivamente, vengono esaminate due soluzioni di moltiplicatori veloci per la generazione dei prodotti parziali: il Modified Booth Encoding e il Baugh Wooley. Inoltre, viene utilizzato l'albero di compressione

che sfrutta lo schema di riduzione del Reduced Area Multiplier. Infine, viene effettuato un ultimo test utilizzando i compressori 4:2 esatti all'interno dell'albero

3. Filtro approssimato: in questo capitolo si utilizza l'aritmetica approssimata per cercare di ottimizzare le prestazioni del filtro, impiegando diversi tipi di compressori approssimati 4:2 all'interno dell'albero. L'obiettivo è quello di valutare una soluzione in grado di ridurre l'area e il consumo di potenza, mantenendo una risoluzione accettabile. In seguito, verranno confrontati i risultati ottenuti per individuare la soluzione migliore.

4. Risultati e confronti: in questo capitolo, saranno esaminati i risultati delle diverse implementazioni per individuare le migliori configurazioni.

5. Conclusioni: nell'ultimo capitolo verranno tratte le conclusioni.

1.2 SerDes

Il SerDes è un dispositivo che consente di trasmettere dati ad alta velocità convertendo i dati seriali in interfacce parallele in entrambe le direzioni. Grazie a questa tecnologia, è possibile trasmettere dati su una linea differenziale o singola riducendo il numero di pin e connessioni di ingresso/uscita necessari.
Il lato trasmettitore del SerDes si occupa di convertire i dati paralleli in dati seriali, mentre il lato ricevitore svolge la funzione opposta. Grazie a questa tecnologia, è possibile trasmettere grandi quantità di dati in modo efficiente e con un minimo di ingombro.
Tuttavia, progettarlo presenta alcune sfide che non sono destinate a diminuire, date le crescenti richieste di velocità e i requisiti di dati sempre più elevati.
La richiesta di progetti che utilizzano i SerDes sta crescendo costantemente, soprattutto nei grandi data center che superano i 100 Gbps e che mirano a raggiungere velocità di 400 Gbps o addirittura 800 Gbps [2]. Questa richiesta è spinta principalmente dalla necessità di velocità di trasmissione sempre più elevate, che è ulteriormente amplificata dalle applicazioni di intelligenza artificiale e apprendimento automatico, che richiedono maggiore potenza di elaborazione e capacità di elaborazione parallela.
Gli standard dell'Optical Internetworking Forum e dell'IEEE stanno definendo velocità ancora più elevate su una singola linea, il che richiede prestazioni sempre migliori da parte dei SerDes.
Al momento, l'adozione della modulazione PAM4 (Pulse-Amplitude Modulation 4-level) rappresenta il modo migliore per aumentare le prestazioni complessive dei SerDes e soddisfare le crescenti esigenze di velocità e capacità di elaborazione parallela. Il PAM4 consente di raggiungere velocità di oltre 100 Gbps per canale,

superando di gran lunga le prestazioni della tecnologia NRZ (Non-Return to Zero). Tuttavia, il PAM4 è suscettibile al rumore di ampiezza a causa della sua capacità di trasmettere più livelli di simboli, ma è considerato un'alternativa migliore rispetto alla tecnologia NRZ grazie al suo funzionamento a frequenze elevate e alla possibilità di operare a metà della frequenza NRZ.

1.3 Codifica del segnale

Attualmente, per risolvere le problematiche legate all'intensificarsi del traffico dati, si ricorre a due distinte metodologie di modulazione del segnale: NRZ e PAM4. [3].

La tecnologia NRZ, illustrata in (Figura 1.2), impiega due livelli di tensione per codificare i bit di un segnale digitale (0 e 1), permettendo di trasmettere un bit di informazione per ogni ciclo di clock. Inoltre, nel caso dei segnali NRZ, il Baud-Rate, che indica la velocità di variazione dei simboli, coincide con il Bit-Rate.

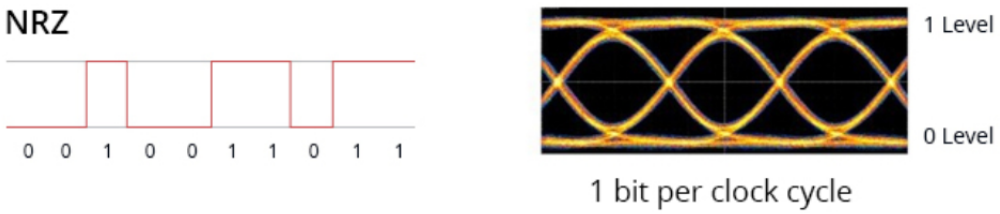


Figura 1.2: Modulazione NRZ

Invece, il PAM4 (Figura 1.3) utilizza quattro livelli di segnale per rappresentare 2 bit di informazione logica (0, 1, 2, 3) in ogni periodo, raddoppiando la quantità di informazioni trasmesse per ciclo rispetto alla tecnologia NRZ. Tuttavia, la forma d'onda del PAM4 ha 4 livelli differenti: 00, 01, 10 o 11, il che lo rende suscettibile al rumore di ampiezza.

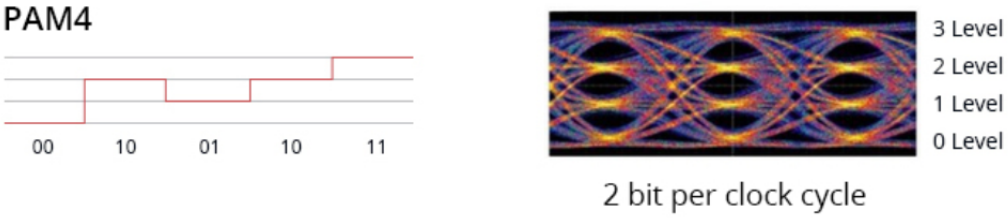


Figura 1.3: Modulazione PAM4

La diminuzione di un terzo dell'altezza dell'occhio, come mostrato nella (Figura 1.4), aumenta la vulnerabilità del segnale PAM4 al rumore, portando a un incremento del

Bit Error Rate (BER). Di conseguenza, per assicurare che il sistema di trasmissione raggiunga il BER richiesto, è stata implementata la tecnica di correzione degli errori in trasmissione, nota come Forward Error Correction (FEC)..

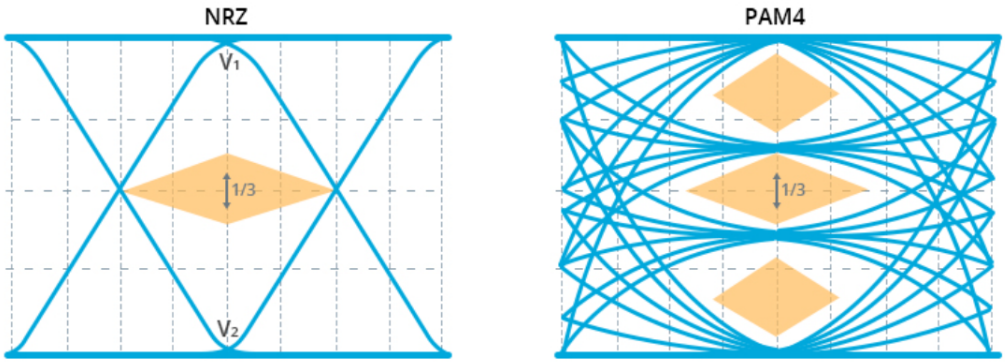


Figura 1.4: Diagramma ad occhio

Per abbassare il BER in un canale che utilizza la modulazione PAM4, è indispensabile impiegare l'equalizzazione nel dispositivo di ricezione e la pre-compensazione nel dispositivo di trasmissione. Questi processi comportano un consumo energetico superiore rispetto a un collegamento NRZ operante alla stessa frequenza di clock. Di conseguenza, i transceiver PAM4 tendono a dissipare una maggiore quantità di calore.

1.4 Serial link

La trasmissione seriale è preferibile rispetto alla trasmissione parallela per diverse ragioni.
In primo luogo, la trasmissione seriale consente di trasmettere i dati su lunghe distanze con una minore perdita di qualità del segnale, garantendo una migliore affidabilità del sistema. Inoltre, la trasmissione seriale richiede meno cavi e componenti rispetto alla trasmissione parallela, semplificando la progettazione e riducendo i costi complessivi del sistema.
Infine, consente di ottenere una maggiore larghezza di banda rispetto alla trasmissione parallela, permettendo di trasmettere un maggior volume di dati in un determinato intervallo di tempo.
Al contrario, la trasmissione parallela è più soggetta a interferenze e perdite di segnale su lunghe distanze, a causa della presenza di più cavi e componenti che possono causare rumore e distorsioni del segnale.

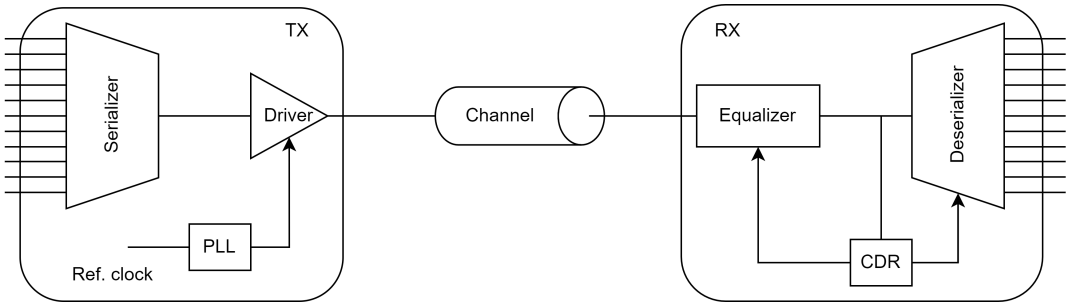


Figura 1.5: Componenti di un SerDes

Il trasmettitore converte i dati in un formato seriale e li modula per la trasmissione attraverso il canale [4]. La sincronizzazione temporale è garantita da un Phase-Locked Loop (PLL).
Il segnale ricevuto attraverso il canale può essere distorto e influenzato dalle interferenze del canale stesso. Il ricevitore campiona i dati e, di solito, determina immediatamente il valore del simbolo di dati ricevuto, grazie alla bassa risoluzione e all'alta velocità di funzionamento.
Le forme d'onda ricevute sono spesso campionate in quadratura per fornire dati e campioni di fronte per il Clock-Data Recovery loop (CDR), che regola la fase del clock generato dal PLL del ricevitore in base ai dati in arrivo.

Il canale viene scelto in base alla distanza a cui deve essere trasmesso il segnale e in base alla banda necessaria. Questo può essere una pista di una PCB, per le comunicazioni a breve distanze, oppure una fibra ottica se il segnale deve essere trasmesso per lunghe distanze.
Tuttavia, con l'aumentare della velocità di trasmissione dei dati, questi fili si comportano come linee di trasmissione con perdite, degradando gravemente i simboli dei dati trasmessi [5].
L'equalizzazione è una tecnica utilizzata per superare le non idealità dei simboli. Il principale parametro di prestazione in tutte le applicazioni che impiegano collegamenti seriali è il Bit Error Rate, mentre, la principale fonte di rumore nei collegamenti seriali ad alta velocità è l'Inter-Symbol Interference (ISI), causata dall'attenuazione del canale dipendente dalla frequenza. Il degrado del margine di rumore dovuto all'ISI è meglio quantificato da un diagramma a occhio.

Il ricevitore di un SerDes utilizza un equalizzatore per compensare le attenuazioni e le distorsioni del segnale trasmesso attraversi il canale di comunicazione. Questa tecnologia è in grado di rilevare le variazioni di ampiezza e di fase del segnale e di correggerle per ripristinare il segnale originale. Grazie all'equalizzatore, la qualità del segnale ricevuto viene migliorata e la distanza massima di trasmissione

viene aumentata.

1.5 Decision Feedback Equalizer

Un DFE è un dispositivo utilizzato nei sistemi di comunicazione digitali per compensare le distorsioni del canale di trasmissione. Questo tipo di equalizzatore è in grado di correggere gli errori di decisione causati dalla presenza di interferenza intersimbolica nel segnale ricevuto, utilizza le decisioni precedenti nel tentativo di stimare il simbolo corrente con un rilevatore simbolo per simbolo. Qualsiasi ISI di coda causata da un simbolo precedente viene ricostruita e quindi sottratta.
Il DFE è un dispositivo intrinsecamente non lineare, ma, supponendo che tutte le decisioni precedenti fossero corrette, è possibile effettuare un'analisi lineare.
La struttura dell'equalizzatore è composta da: Filtro lineare anti-causale a feed forward; Filtro lineare causale a retroazione; Rilevatore di simboli [6].
L'ingresso al filtro di retroazione è la decisione del simbolo precedente (dal dispositivo di decisione). La Figura 1.6 mostra il layout del DFE.

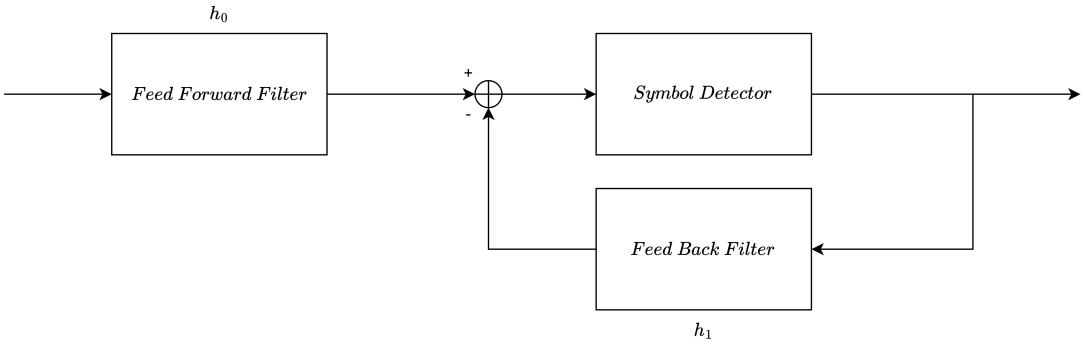


Figura 1.6: Struttura dell'equalizzatore composto dal Feed Forward Equalizer e dal Decision Feedback Equalizer

Il FFE viene utilizzato per compensare le distorsioni del canale di trasmissione, mentre il FBF è impiegato per correggere gli errori di decisione.
Il segnale in ingresso viene convoluto con il FFE per generare un segnale stimato, che viene poi sottratto dal segnale in ingresso per produrre un residuo. Il residuo viene convoluto con il FBF, e il segnale di uscita del FBF viene sommato al segnale stimato per produrre il segnale di uscita finale. In questo modo, il DFE è in grado di compensare le distorsioni del canale e correggere gli errori di decisione, migliorando la qualità del segnale trasmesso.
Il principale inconveniente dell'equalizzatore in esame è la sua tendenza alla propagazione degli errori. Un'errata interpretazione del segnale, dovuta alla presenza di rumore, può compromettere le decisioni successive, incrementando il rischio di

ulteriori errori. Questa catena di decisioni sbagliate continuerà fino a quando non verranno effettuate scelte accurate che interrompano il ciclo.

1.6 Feed Forward Equalizer

Il filtro di equalizzazione può essere di diversi tipi, ma il più comune è un filtro FIR a coefficienti adattivi. Grazie alla sua capacità di adattarsi alle variazioni del canale di trasmissione, questo tipo di filtro è in grado di compensare le distorsioni del canale.
Inoltre, il filtro FIR ha una struttura semplice e può essere facilmente implementato in hardware. Tuttavia, a seconda delle specifiche dell'applicazione, il filtro FFE può anche essere di tipo IIR (Infinite Impulse Response) o di tipo ibrido FIR/IIR.
L'equalizzatore è una componente fondamentale utilizzato per compensare le distorsioni del canale di trasmissione in un sistema di comunicazione digitale. Il FFE è composto da una serie di tap (o coefficienti) che vengono regolati in base alle caratteristiche del canale di trasmissione. Il segnale in ingresso viene convoluto con il FFE per produrre un segnale stimato che viene poi sottratto dal segnale originale per produrre un residuo.
Sebbene teoricamente il filtro FFE potrebbe essere un filtro analogico, nella pratica i filtri digitali sono più comunemente utilizzati a causa della loro maggiore precisione e flessibilità. In particolare, il filtro digitale FIR a coefficienti adattivi è il più utilizzato.

I FFE hanno numerose proprietà desiderabili dal punto di vista dell'implementazione digitale [7]. Non richiedono retroazione, il che consente la parallelizzazione e il pipelining. La velocità di elaborazione del FFE non è un problema limitante, ma la latenza e il consumo di energia possono essere un fattore da considerare, anche se scalano bene con la velocità di trasmissione.

L'utilizzo di un filtro FIR come FFE può presentare alcuni problemi di area e potenza. In particolare, il filtro richiede un numero elevato di moltiplicazioni e somme per implementare la sua funzione di equalizzazione, il che può aumentare significativamente l'area occupata dal circuito integrato e il consumo di potenza.
Per ridurre area e potenza è possibile utilizzare architetture approssimate per ridurre l'area e la potenza del filtro FIR, riducendo così il numero di porte logiche e di connessioni necessari per implementare il filtro.
Tuttavia, queste tecniche di ottimizzazione possono introdurre un certo errore di approssimazione nel risultato dell'equalizzazione, il che può influire sulla qualità del segnale ricevuto.
In generale, la scelta del tipo di filtro da utilizzare come FFE dipende dalle specifiche

dell'applicazione e dalle esigenze di area e potenza del sistema.

Capitolo 2

Filtro esatto

L'equazione di un filtro FIR è data dalla somma dei prodotti tra i campioni in input e i coefficienti del filtro. Formalmente, un filtro FIR a N taps ha la seguente Equazione 2.1

$$y[n] = b[0]*x[n]+b[1]*x[n-1]+b[2]*x[n-2]+...+b[N-1]*x[n-(N-1)] \quad (2.1)$$

dove $y[n]$ è il campione di output al tempo n, $x[n]$ è il campione in input al tempo n, e $b[0]$, $b[1]$,..., $b[N-1]$ sono i coefficienti del filtro. Lo schema è riportato nella Figura 2.1.

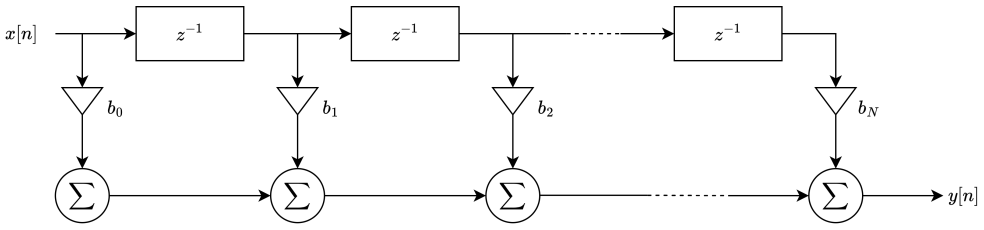


Figura 2.1: Schema di un filtro FIR

Per implementare l' Equazione 2.1 del filtro è necessario moltiplicare le uscite dei taps con i relativi coefficienti. Questo processo viene effettuato in parallelo, ovvero tutti i moltiplicatori producono la loro moltiplicazione in un unico colpo di clock, e le uscite vengono sommate con un'unica operazione di somma. Tale approccio consente di minimizzare i tempi di elaborazione e di massimizzare l'efficienza del sistema.

2.1 Moltiplicazione binaria

Il moltiplicatore binario è un circuito elettronico utilizzato nei computer e nell'elettronica digitale per moltiplicare due numeri binari. Ci sono diverse tecniche aritmetiche disponibili per realizzare un moltiplicatore digitale, ma la maggior parte di esse prevede il calcolo degli elementi dei prodotti parziali, che vengono poi sommati attraverso sommatori binari. Questo processo è simile alla moltiplicazione tradizionale, ma utilizza un sistema numerico in base 2 (binario).
A differenza del metodo di moltiplicazione dei numeri decimali insegnato a scuola, che si basa sul calcolo dei prodotti parziali, sullo spostamento a sinistra e sulla successiva somma, la moltiplicazione binaria è molto più semplice poiché ogni numero lungo viene moltiplicato per una cifra (0 o 1), il che è molto più facile che in decimale, poiché il prodotto per 0 o 1 è semplicemente 0 o lo stesso numero. Pertanto, la moltiplicazione di due numeri binari si riduce al calcolo dei prodotti parziali (che sono 0 o il primo numero), allo spostamento a sinistra e alla successiva somma (un'addizione binaria, ovviamente).
Sebbene il metodo di moltiplicazione binaria sia matematicamente corretto, richiede molte addizioni intermedie, il che lo rende lento. Tuttavia, i moltiplicatori più veloci sono progettati per eseguire meno addizioni. Inoltre, il metodo di moltiplicazione binaria permette a una CPU di piccole dimensioni di eseguire la moltiplicazione utilizzando le funzioni di $shift$ e add della sua unità logica aritmetica, invece di un circuito specializzato.
Il secondo problema riguarda il metodo di moltiplicazione scolastico di base, che gestisce il segno con una regola separata. Al contrario, i computer moderni incorporano il segno del numero nel numero stesso, di solito nella rappresentazione del complemento a due. Questo obbliga a modificare il processo di moltiplicazione per gestire i numeri a complemento a due, rendendo il processo ancora più complicato. Allo stesso modo, i processori che utilizzano il complemento a uno, il segno e la grandezza, l'IEEE-754 o altre rappresentazioni binarie richiedono modifiche specifiche al processo di moltiplicazione.

2.2 Implementazione hardware

La moltiplicazione può essere suddivisa in tre fasi:

- generazione del prodotto parziale
- riduzione del prodotto parziale
- calcolo del prodotto finale

Le generazioni precedenti di architetture moltiplicatrici si avvalevano di uno shifter e di un accumulatore per aggiungere ciascun prodotto parziale, tipicamente uno per ciclo di clock, privilegiando così l'efficienza in termini di superficie del die a scapito della velocità di elaborazione.
Le moderne architetture di moltiplicatori utilizzano l'algoritmo di Baugh-Wooley [8], gli alberi di Wallace [9] o i moltiplicatori Dadda [10, 11] per sommare i prodotti parziali in un singolo ciclo. Utilizzando la codifica Booth modificata, come indicato in [12], per uno dei due operandi, è possibile ottimizzare l'implementazione dell'albero di Wallace, il che comporta una riduzione della quantità di prodotti parziali da sommare.
Per aumentare la velocità, i moltiplicatori shift-and-add richiedono un sommatore veloce (più veloce del ripple-carry). Un moltiplicatore "a ciclo singolo" (o "moltiplicatore veloce") è pura logica combinatoria. Tuttavia, nel moltiplicatore veloce, il processo di riduzione del prodotto parziale di solito contribuisce maggiormente al ritardo, alla potenza e all'area del moltiplicatore.
Per migliorare la velocità, gli stadi di "riduzione del prodotto parziale" sono tipicamente implementati come un sommatore carry-save composto da compressori, mentre la fase di "calcolo del prodotto finale" è implementata come un sommatore veloce (qualcosa di più veloce del ripple-carry).
I moltiplicatori veloci spesso utilizzano full adders come compressori ("compressori 3:2") implementati in CMOS statico. Per ottenere prestazioni migliori nella stessa area o le stesse prestazioni in un'area più piccola, i progetti di moltiplicatori possono utilizzare compressori di ordine superiore, come i compressori 4:2.

Per svolgere calcoli in modo più efficiente, con l'aumento dei bit a disposizione, l'utilizzo di moltiplicatori veloci è diventato indispensabile per garantire un miglioramento del filtro. I moltiplicatori veloci consentono di eseguire moltiplicazioni in un tempo molto breve, riducendo il tempo di esecuzione e aumentando la velocità di elaborazione delle informazioni. Questa esigenza è diventata ancora più critica con l'avvento delle applicazioni multimediali e dell'elaborazione di segnali digitali, che richiedono l'elaborazione di grandi quantità di dati in tempo reale. In questo contesto, i moltiplicatori veloci sono diventati un elemento imprescindibile per molte applicazioni moderne.

2.3 Generazione dei prodotti parziali

Supponiamo di voler moltiplicare tra loro due interi a 8 bit senza segno: $a[7:0]$ e $b[7:0]$. Possiamo ottenere otto prodotti parziali (Figura 2.2) eseguendo otto moltiplicazioni a 1 bit, una per ogni bit del moltiplicando a:
dove $\{8\{a[0]\}\}$ significa ripetere $a[0]$ (il bit 0 di a) 8 volte (notazione $Verilog$).

```
p0[7:0]  = a[0]  x  b[7:0]  =  {8{a[0]}}  &  b[7:0]
p1[7:0]  = a[1]  x  b[7:0]  =  {8{a[1]}}  &  b[7:0]
p2[7:0]  = a[2]  x  b[7:0]  =  {8{a[2]}}  &  b[7:0]
p3[7:0]  = a[3]  x  b[7:0]  =  {8{a[3]}}  &  b[7:0]
p4[7:0]  = a[4]  x  b[7:0]  =  {8{a[4]}}  &  b[7:0]
p5[7:0]  = a[5]  x  b[7:0]  =  {8{a[5]}}  &  b[7:0]
p6[7:0]  = a[6]  x  b[7:0]  =  {8{a[6]}}  &  b[7:0]
p7[7:0]  = a[7]  x  b[7:0]  =  {8{a[7]}}  &  b[7:0]
```

Figura 2.2: Prodotti parziali di una moltiplicazione 8 x 8

Per ottenere il prodotto, dobbiamo quindi sommare tutti e otto i prodotti parziali, come mostrato in Figura 2.3

```
                                          + p0[7] p0[6] p0[5] p0[4] p0[3] p0[2] p0[1] p0[0]
                                    + p1[7] p1[6] p1[5] p1[4] p1[3] p1[2] p1[1] p1[0]
                              + p2[7] p2[6] p2[5] p2[4] p2[3] p2[2] p2[1] p2[0]
                        + p3[7] p3[6] p3[5] p3[4] p3[3] p3[2] p3[1] p3[0]
                  + p4[7] p4[6] p4[5] p4[4] p4[3] p4[2] p4[1] p4[0]
            + p5[7] p5[6] p5[5] p5[4] p5[3] p5[2] p5[1] p5[0]
      + p6[7] p6[6] p6[5] p6[4] p6[3] p6[2] p6[1] p6[0]
+ p7[7] p7[6] p7[5] p7[4] p7[3] p7[2] p7[1] p7[0]
-------------------------------------------------------------------------------------
P[15] P[14] P[13] P[12] P[11] P[10] P[9]  P[8]  P[7]  P[6]  P[5]  P[4]  P[3]  P[2]  P[1]  P[0]
```

Figura 2.3: Somma prodotti parziali senza segno

In altre parole, $P[15:0]$ viene prodotto sommando p_0, $p_1 << 1$ (shift di una posizione a sinistra), $p_2 << 2$ (shift di due posizioni a sinistra) e così via, per ottenere il prodotto finale a 16 bit senza segno.

Nel caso in cui volessi moltiplicare tra loro due interi a 8 bit con segno, la moltiplicazione è comunemente conosciuta come moltiplicazione del complemento a 2. Questa operazione può essere eseguita moltiplicando le grandezze dei due numeri e poi estendendo il segno dei prodotti parziali (MSB) fino al bit di segno del numero finale (m+n-bit), come mostrato in Figura 2.4.

2.3.1 Moltiplicatore di Baugh-Wooley

Quello che è stato appena visto può essere fatto in modo più efficiente con il moltiplicatore di Baugh-Wooley [8] che permette di moltiplicare in modo veloce una matrice parallela di $m - bit$ per $n - bit$ utilizzando il complemento a due.
L'algoritmo converte la moltiplicazione a complemento a due in un problema di addizione parallela equivalente, dove ogni bit del prodotto parziale è ottenuto dall'AND di un bit del moltiplicatore e di un bit del moltiplicando, e tutti i bit del prodotto parziale hanno segno positivo.
Nella moltiplicazione a complemento a due convenzionale, i bit del prodotto parziale

```
p0[7] p0[7] p0[7] p0[7] p0[7] p0[7] p0[7] p0[7] p0[7] p0[6] p0[5] p0[4] p0[3] p0[2] p0[1] p0[0]
p1[7] p1[7] p1[7] p1[7] p1[7] p1[7] p1[7] p1[7] p1[6] p1[5] p1[4] p1[3] p1[2] p1[1] p1[0]
p2[7] p2[7] p2[7] p2[7] p2[7] p2[7] p2[7] p2[6] p2[5] p2[4] p2[3] p2[2] p2[1] p2[0]
p3[7] p3[7] p3[7] p3[7] p3[7] p3[7] p3[6] p3[5] p3[4] p3[3] p3[2] p3[1] p3[0]
p4[7] p4[7] p4[7] p4[7] p4[7] p4[6] p4[5] p4[4] p4[3] p4[2] p4[1] p4[0]
p5[7] p5[7] p5[7] p5[7] p5[6] p5[5] p5[4] p5[3] p5[2] p5[1] p5[0]
p6[7] p6[7] p6[7] p6[6] p6[5] p6[4] p6[3] p6[2] p6[1] p6[0]
p7[7] p7[7] p7[6] p7[5] p7[4] p7[3] p7[2] p7[1] p7[0]
-------------------------------------------------------------------------------------------
P[15] P[14] P[13] P[12] P[11] P[10] P[9]  P[8]  P[7]  P[6]  P[5]  P[4]  P[3]  P[2]  P[1]  P[0]
```

Figura 2.4: Somma prodotti parziali con segno

possono avere segno negativo oltre che positivo. Per ottenere il prodotto a $n+m$-bit $P = (p_{n+m-1}, p_{n+m-2}, \cdots, p_0)$ tra un moltiplicatore a m-bit $Y = (y_{m-1}, Y_{m-2}, \cdots, y_0)$ e un moltiplicando a n-bit $X = (x_{n-1}, \cdots, x_o)$, si utilizza la rappresentazione binaria mostrata nella Figura 2.5. Ogni bit del moltiplicatore e del moltiplicando viene moltiplicato tramite l'operatore AND per ottenere i bit del prodotto parziale, che vengono poi sommati per formare il prodotto finale.
La difficoltà della moltiplicazione a complemento a due consiste nei segni del moltiplicando e del moltiplicatore.
Siano Y_v (Equazione 2.2) e X_v (Equazione 2.3) i valori rispettivi del moltiplicando Y e del moltiplicatore X rappresentati in complemento a due.

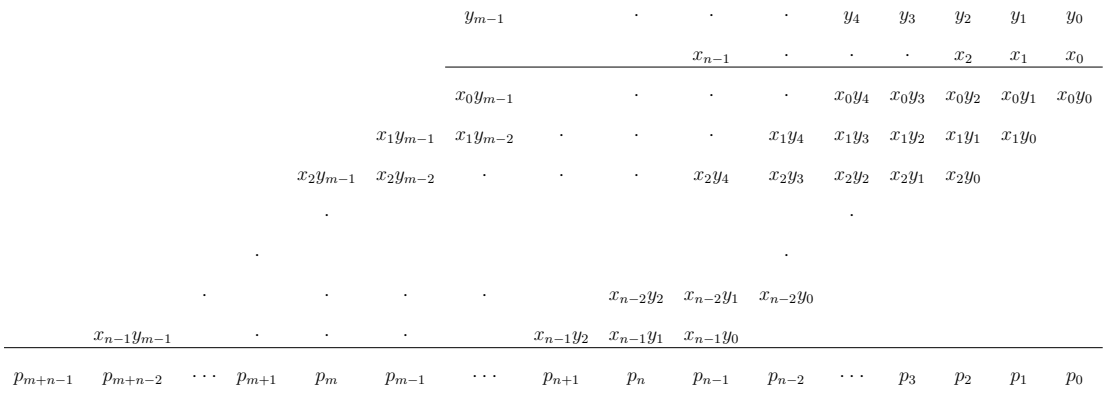


Figura 2.5: Moltiplicazione convenzionale m per n bit con complemento a due [8]

$$Y_v = -y_{m-1}2^{m-1} + \sum_{i=0}^{m-2} y_i 2^i \tag{2.2}$$

$$X_v = -x_{n-1}2^{n-1} + \sum_{i=0}^{n-2} x_i 2^i \tag{2.3}$$

Il valore di P_v del prodotto P è riportato nell'Equazione 2.4.

$$
\begin{aligned}
P_v &= -p_{m+n-1}2^{m+n-1} + \sum_{i=0}^{m+n-2} p_i 2^i = Y_v X_v \\
&= (-y_{m-1}2^{m-1} + \sum_{i=0}^{m-2} y_i 2^i)(-x_{n-1}2^{n-1} + \sum_{i=0}^{n-2} x_i 2^i) \\
&= (x_{n-1}y_{m-1}2^{m+n-2} + \sum_{i=0}^{n-2}\sum_{j=0}^{m-2} x_i y_j 2^{i+j}) \\
&- (\sum_{i=0}^{m-2} x_{n-1}y_i 2^{n-1+i} + \sum_{i=0}^{n-2} y_{m-1}x_i 2^{m-1+i})
\end{aligned}
\tag{2.4}
$$

Quando si forma P sommando i prodotti parziali, è necessario considerare il segno dei bit del prodotto parziale. In particolare, i segni di $x_{n-1}y_i$ per $i = 0, \cdots, m-2$ e $y_{m-1}x_i$ per $i = 0, \cdots, n-2$ sono negativi. Riscrivendo i bit del prodotto parziale come mostrato in Figura 2.6, tutti i bit del prodotto parziale con segno negativo sono collocati nelle ultime due righe.
Il prodotto si forma sommando le prime $n-2$ righe del prodotto parziale e sottraendo le ultime due righe. Invece di sottrarre i prodotti parziali con segno negativo, è possibile aggiungere la negazione dei prodotti parziali. Il valore della negazione di un numero a complemento a due $Z = (Z_{k-1}, \cdots, z_o)$ con valore Z_v riportato nell'Equazione 2.5,

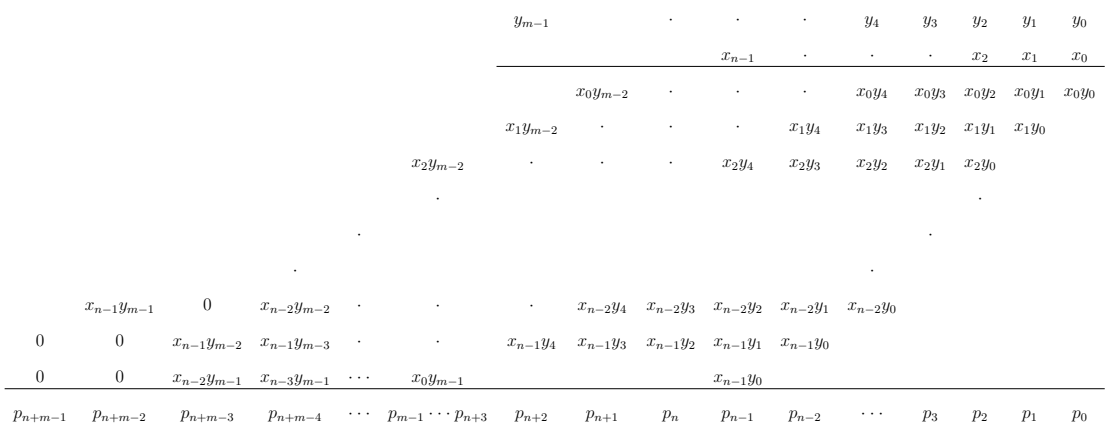


Figura 2.6: Separazione positiva/negativa dei bit del prodotto parziale [8]

$$
-Z_v = 1 - \overline{z}_{k-1}2^{k-1} + \sum_{i=0}^{k-2} \overline{z}_i 2^i \tag{2.5}
$$

dove $\overline{z}_i$ è il complemento di z_i. Pertanto, la sottrazione di

$$2^{n-1}(-0 \cdot 2^m + 0 \cdot 2^{m-1} + \sum_{i=0}^{m-2} x_{n-1} y_i 2^i) \tag{2.6}$$

può essere sostituita con l'aggiunta di

$$2^{n-1}(-1 \cdot 2^m + 1 \cdot 2^{m-1} + 1 + \sum_{i=0}^{m-2} \overline{\mathrm{x_{n-1} y_i}} 2^i) \tag{2.7}$$

Pertanto, la riga del prodotto parziale di Figura 2.6 che contiene

$$0\ 0\ x_{n-1}y_{m-2}\ x_{n-1}y_{m-3} \cdots\ \overline{\mathrm{x_{n-1} y_0}}$$

è sostituito da

$$1\ 1\ \overline{\mathrm{x_{n-1} y_{m-2}}}\ \overline{\mathrm{x_{n-1} y_{m-3}}}\ \cdots \overline{\mathrm{x_{n-1} y_0}}$$

con un "1" aggiunto alla colonna p_{n-1}. Allo stesso modo, la riga contenente

$$0\ 0\ x_{n-2}y_{m-1}\ x_{n-3}y_{m-1} \cdots\ x_0 y_{m-1}$$

è sostituito da

$$1\ 1\ \overline{\mathrm{x_{n-2} y_{m-1}}}\ \overline{\mathrm{x_{n-3} y_{m-1}}}\ \cdots \overline{\mathrm{x_0 y_{m-1}}}$$

Dopo l'aggiunta di un "1" nella colonna p_{m-i}, tutti i bit del prodotto parziale possono essere trattati allo stesso modo per quanto riguarda il segno. Tuttavia, sostituendo l'Equazione 2.7 con l'Equazione 2.6 nella Figura 2.6, si crea una non uniformità nei bit del prodotto parziale, poiché alcuni bit sono la $NAND$ di un bit del moltiplicatore e di un bit del moltiplicando, mentre altri sono formati da una AND. Per semplificare la situazione, si utilizzano le seguenti equivalenze. È importante notare che l'Equazione 2.7 ha un valore specifico

$$\begin{cases} 0, & \text{per } x_{n-1} = 0 \\ 2^{n-1}(-2^m + 2^{m-1} + 1 + \sum_{i=0}^{m-2} \overline{\mathrm{y}}_i 2^i), & \text{per } x_{n-1} = 1 \end{cases} \tag{2.8}$$

Dall'Equazione 2.8 segue che l'Equazione 2.7 può essere riscritta come

$$2^{n-1}(-2^m + 2^{m-1} + \overline{\mathrm{x}}_{n-1} 2^{m-1} + x_{n-1} + \sum_{i=0}^{m-2} x_{n-1} \overline{\mathrm{y}}_i 2^i) \tag{2.9}$$

Pertanto, l'Equazione 2.7 è equivalente all'Equazione 2.9. Poiché la penultima riga di bit del prodotto parziale della Figura 2.6 ha la forma data dall'Equazione 2.7, si sostituisce l'Equazione 2.9 a questa riga. Effettuando una sostituzione simile per

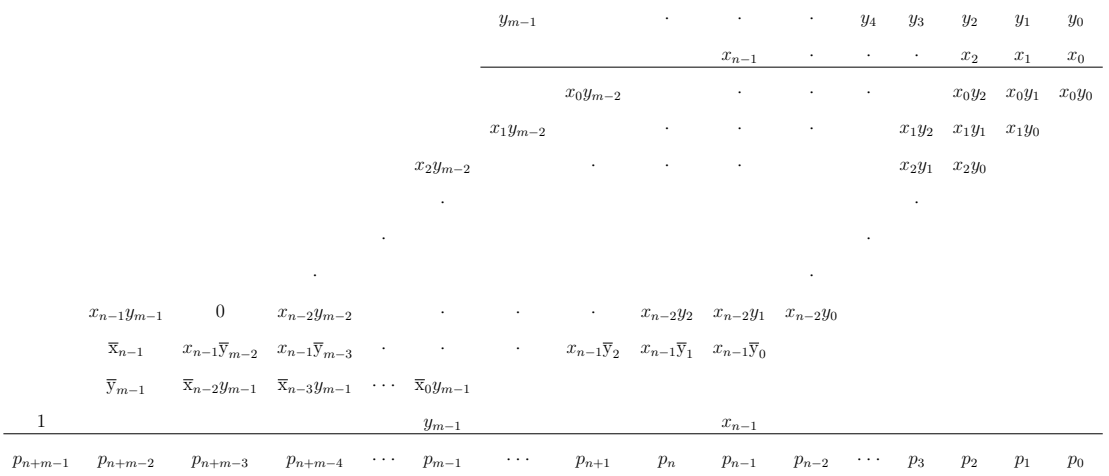


Figura 2.7: Algoritmo con tutti i bit positivi del prodotto parziale [8]

l'ultima riga di bit del prodotto parziale e aggiungendo delle costanti, si ottengono i bit del prodotto parziale della Figura 2.7.
La Figura 2.7 presenta una caratteristica principale dei bit del prodotto parziale, ovvero l'uniformità. Tale uniformità offre due vantaggi importanti:

1. I bit del prodotto parziale si ottengono mediante l'AND tra un bit del moltiplicatore e un bit del moltiplicando.

2. Ogni bit del prodotto parziale ha un coefficiente positivo.

Di conseguenza, il prodotto può essere ottenuto solo mediante l'utilizzo della funzione AND e della funzione ADD. Non è necessario utilizzare la sottrazione o la funzione $NAND$ per formare $\overline{x_i y_i}$.

2.3.2 Codifica di Booth

L'algoritmo di Booth è un'efficiente implementazione hardware di un circuito digitale che moltiplica due numeri binari in notazione a complemento a due. Grazie alla sua capacità di ricodificare i numeri moltiplicati, la moltiplicazione di Booth consente di realizzare circuiti di moltiplicazione più piccoli e veloci. Sono ampiamente utilizzati nei prodotti orientati agli ASIC per la loro maggiore velocità di calcolo e l'area ridotta. La generazione del prodotto parziale dipende dallo schema di ricodifica, come la codifica radix-2, radix-4, etc.

L'algoritmo radix-2 si avvale dell'esame di coppie consecutive di bit del moltiplicatore Y per effettuarne la ricodifica. Per avviare il meccanismo di ricodifica,

occorre adottare una convenzione di rappresentazione particolare che prenda in considerazione i valori assoluti di 1, 0 e -1. Tale convenzione è denominata sistema di codifica Signed Digit (SD), che mantiene inalterati i valori di 1 e 0, mentre il valore -1 è trattato come una singola cifra. Nel caso in cui Y rappresenti il moltiplicatore e X il moltiplicando, per illustrare il procedimento di moltiplicazione $X \cdot Y$ con l'impiego dell'algoritmo di Booth, è necessario riformulare Y nel sistema di codifica SD nel modo seguente:

$$Y = y_{n-1}2^{n-1} + \sum_{i=0}^{n-2} y_i 2^i \tag{2.10}$$

Dove n indica il numero di bit che rappresentano Y e y_{n-1} indica il bit di segno. Con alcune modifiche all'Equazione 2.10, è ovvio che:

$$Y = \sum_{i=0}^{n-1} (y_{i-1} + y_i)2^i \tag{2.11}$$

In cui $y_{-1} = 0$. Un esame dettagliato dell'Equazione 2.11 mostra che il moltiplicatore Y necessita di una ricodifica utilizzando i fattori di scala (-1, 0, 1), giustificando così l'impiego del sistema di codifica SD. Per generare questi fattori, è necessario suddividere il moltiplicatore in gruppi di 2 bit ciascuno (Figura 2.8) e, facendo riferimento alla tabella di verità presentata nella Tabella 2.1, si procede alla loro codifica.

Questo processo, noto come codifica radix-2 di Booth, impiega 3 tipi di operazioni per generare prodotti parziali. In base alla Tabella 2.1, queste funzioni sono:

1. Per 00 e 11, il prodotto è la moltiplicazione di 0 al moltiplicando.
2. Per 01, il prodotto è il moltiplicando stesso.
3. Per 10, il moltiplicando deve essere complementato.

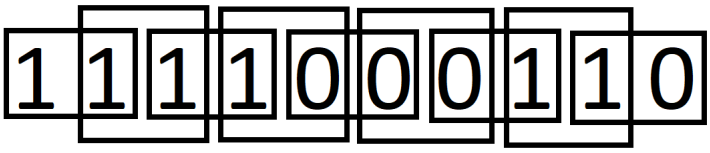


Figura 2.8: Raggruppamento di bit come da ricodifica di Booth utilizzando Radix-2

Il moltiplicatore di Booth con radix-4 supera gli svantaggi del moltiplicatore di Booth con radix-2, riducendo la metà dei prodotti parziali nei moltiplicatori. In

y_i	y_{i-1}	Valore Moltiplicatore
0	0	0*Moltiplicando
0	1	1*Moltiplicando
1	0	-1*Moltiplicando
1	1	0*Moltiplicando

Tabella 2.1: Tabella di codifica di Booth radix-2

radix-4, la codifica dei moltiplicatori si basa sui bit dei moltiplicatori e si confrontano 3 bit alla volta con la tecnica della sovrapposizione, come in Figura 2.9. Il raggruppamento inizia dall'LSB e il primo blocco contiene solo due bit dei moltiplicatori e assume zero per il terzo bit, conformandosi alla tabella di codifica di Booth (Tabella 2.2).

Si consideri la moltiplicazione di due numeri interi di n bit, A (moltiplicando) e B (moltiplicatore), nella rappresentazione del complemento a 2, come riportato nell'Equazione 2.12.

$$\begin{aligned} A &= -a_{n-1}2^{n-1} + \sum_{i=0}^{n-2} a_i 2^i \\ B &= -b_{n-1}2^{n-1} + \sum_{i=0}^{n-2} b_i 2^i \end{aligned} \tag{2.12}$$

In MBE, B dell'Equazione 2.12 diventa

$$B = \sum_{i=0}^{n/2-1} m_i 2^{2i} = \sum_{i=0}^{n/2-1} (-2b_{2i+1} + b_{2i} + b_{2i-1})2^{2i} \tag{2.13}$$

dove $b_{-1} = 0$ e $m_i \in \{-2, -1, 0, 1, 2\}$. In base ai risultati codificati da B, i selettori di Booth scelgono $-2A$, $-A$, 0, A o $2A$ per generare le righe del prodotto parziale, come mostrato nella Tabella 2.2.

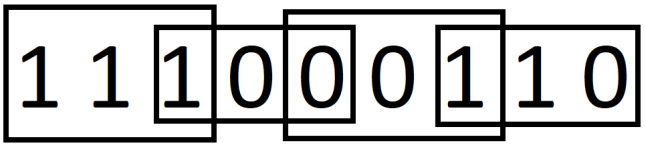


Figura 2.9: Raggruppamento di bit come da ricodifica di Booth utilizzando Radix-4

y_{2j+1}	y_{2j}	y_{2j-1}	Valore Moltiplicatore
0	0	0	0*Moltiplicando
0	0	1	1*Moltiplicando
0	1	0	1*Moltiplicando
0	1	1	2*Moltiplicando
1	0	0	-2*Moltiplicando
1	0	1	-1*Moltiplicando
1	1	0	-1*Moltiplicando
1	1	1	0*Moltiplicando

Tabella 2.2: Tabella di codifica di Booth modificata.

Esistono diversi tipi di radix di un moltiplicatore Booth, che si differenziano per il numero di bit considerati durante il processo di moltiplicazione.
I tipi di radix più comuni sono:

- Radix-2: utilizza due bit per volta durante il processo di moltiplicazione. Questo tipo di radix è il più semplice e richiede un numero minimo di porte logiche, ma può essere meno efficiente rispetto ad altri tipi di radix.

- Radix-4: utilizza quattro bit per volta durante il processo di moltiplicazione. Questo tipo di radix è più efficiente rispetto al radix-2 e richiede meno porte logiche, ma può essere più complesso da implementare.

- Radix-8: utilizza otto bit per volta durante il processo di moltiplicazione. Questo tipo di radix richiede un maggior numero di porte logiche e può essere più complesso da implementare.

Come si può vedere in [13], esistono anche moltiplicatori con radix più alti, come il radix-16 e il radix-32.
Le differenze principali tra questi tipi di radix sono la loro efficienza e la complessità di implementazione. In generale, i tipi di radix più elevati sono più efficienti ma richiedono un maggior numero di porte logiche e possono essere più complessi da implementare. Tuttavia, la scelta del tipo di radix dipende dalle specifiche esigenze del circuito e dalle risorse disponibili.
In questo caso si è utilizzato il radix-4.

2.3.3 Moltiplicatore di Booth modificato con matrice regolare di prodotti parziali

L'algoritmo tradizionale del Modified Booth Encoding (MBE) [12] produce una matrice di moltiplicazione irregolare a causa di un bit aggiuntivo nella posizione meno significativa di ogni riga, come mostrato nella figura Figura 2.10.

b_p	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PP_0						$\overline{s_0}$	s_0	s_0	p_{07}	p_{06}	p_{05}	p_{04}	p_{03}	p_{02}	p_{01}	p_{00}
PP_1					1	$\overline{s_1}$	p_{17}	p_{16}	p_{15}	p_{14}	p_{13}	p_{12}	p_{11}	p_{10}		neg_0
PP_2			1	$\overline{s_2}$	p_{27}	p_{26}	p_{25}	p_{24}	p_{23}	p_{22}	p_{21}	p_{20}		neg_1		
PP_3	1	$\overline{s_3}$	p_{37}	p_{36}	p_{35}	p_{34}	p_{33}	p_{32}	p_{31}	p_{30}		neg_2				
PP_4										neg_3						

Figura 2.10: Esempio di prodotti parziali generati dal MBE convenzionale per una moltiplicazione 8 per 8 [14]

Tuttavia, un nuovo approccio proposto in [14] consente di generare una matrice regolare con un numero inferiore di righe e un overhead trascurabile.
Questa modifica riduce la complessità della riduzione dei prodotti e, di conseguenza, l'area, il ritardo e la potenza dei moltiplicatori MBE.
In particolare, l'approccio proposto riduce il numero di righe da $(n/2+1)$ a $(n/2)$ incorporando l'ultimo bit negativo nei bit di estensione del segno della prima riga, senza introdurre alcun overhead significativo.
Ciò porta a una matrice di moltiplicazione più regolare, come mostrato nella figura Figura 2.11, e a un albero di riduzione più piccolo e veloce.

b_p	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PP_0						α_2	α_1	α_0	p_{07}	p_{06}	p_{05}	p_{04}	p_{03}	p_{02}	p_{01}	τ_{00}
PP_1					1	$\overline{s_1}$	p_{17}	p_{16}	p_{15}	p_{14}	p_{13}	p_{12}	p_{11}	τ_{10}	c_0	
PP_2			1	$\overline{s_2}$	p_{27}	p_{26}	p_{25}	p_{24}	p_{23}	p_{22}	p_{21}	τ_{20}	c_1			
PP_3	1	$\overline{s_3}$	p_{37}	p_{36}	p_{35}	p_{34}	p_{33}	p_{32}	τ_{31}	τ_{30}	c_2					

Figura 2.11: Esempio di prodotti parziali generati dal MBE proposto per una moltiplicazione 8 per 8 [14]

Per la ricodifica MB, sono necessari almeno tre segnali per rappresentare l'insieme di cifre: $\{-2,-1,0,1,2\}$.
Per la generazione dei prodotti parziali p_{ij} si utilizza il selettore proposto in [15], di cui riportata la Tabella 2.3.
Dalla Tabella 2.4 possono essere ricavati τ_{i0} e c_i tramite le equazioni riportate in Equazione 2.14 e Equazione 2.15.

b_{2i+1}	b_{2i}	b_{2i-1}	Operation	neg_i	two_i	one_i	p_{ij}
0	0	0	+0	0	0	0	0
0	0	1	+A	0	0	1	a_j
0	1	0	+A	0	0	1	a_j
0	1	1	+2A	0	1	0	a_{j-1}
1	0	0	-2A	1	1	0	$\overline{\mathrm{a_{j-1}}}$
1	0	1	-A	1	0	1	$\overline{\mathrm{a_j}}$
1	1	0	-A	1	0	1	a_j
1	1	1	-0	0	0	0	0

Tabella 2.3: Tabella selettore MBE [14]

$$\tau_{i0} = one_i \cdot a_0 = \overline{\overline{\mathrm{one_i}} + \overline{\mathrm{a_0}}} \tag{2.14}$$

$$c_i = neg_i \cdot (\overline{\mathrm{one_i}} + \overline{\mathrm{a_0}}) = \overline{\overline{\mathrm{neg_i}} + \mathrm{one_i} \cdot \mathrm{a_0}} \tag{2.15}$$

b_{2i+1}	b_{2i}	b_{2i-1}	τ_{i0}	c_i	τ_{i1}	d_i	$\widetilde{\tau}_{i1}$	e_i
0	0	0	0	0	0	0	1	0
0	0	1	a_0	0	a_1	0	$\overline{\mathrm{a_1}}$	a_1
0	1	0	a_0	0	a_1	0	$\overline{\mathrm{a_1}}$	a_1
0	1	1	0	0	a_0	0	$\overline{\mathrm{a_0}}$	a_0
1	0	0	0	1	a_0	$\overline{\mathrm{a_0}}$	$\overline{\mathrm{a_0}}$	1
1	0	1	a_0	$\overline{\mathrm{a_0}}$	$a_0 \oplus a_1$	$\overline{\mathrm{a_0} + \mathrm{a_1}}$	$a_0 \bigodot a_1$	$\overline{\mathrm{a_0} \cdot \mathrm{a_1}}$
1	1	0	a_0	$\overline{\mathrm{a_0}}$	$a_0 \oplus a_1$	$\overline{\mathrm{a_0} + \mathrm{a_1}}$	$a_0 \bigodot a_1$	$\overline{\mathrm{a_0} \cdot \mathrm{a_1}}$
1	1	1	0	0	0	0	1	0

Tabella 2.4: Tabella della verità per i bit del prodotto parziale [14]

Per rimuovere ulteriormente la riga di prodotto parziale aggiuntiva $PP_{n/2}$, combiniamo il c_i per $i = n/2 - 1$ con il bit di prodotto parziale p_{i1} per produrre un nuovo bit di prodotto parziale τ_{i1} e un nuovo riporto d_i.
Il riporto d_i può quindi essere incorporato nei bit di estensione del segno di PP_0. Tuttavia, se τ_{i1} e d_i sono prodotti sommando c_i e p_{i1}, i loro ritardi di arrivo saranno probabilmente maggiori rispetto agli altri bit del prodotto parziale. Pertanto, produciamo direttamente τ_{i1} e d_i per $i = n/2-1$ da A, B e dalle uscite del codificatore Booth (cioè neg_i, two_i e one_i), come mostrato nella Tabella 2.4.

Le espressioni logiche di τ_{i1} e d_i possono essere ricavati dall'Equazione 2.16 e dall'Equazione 2.17.

$$\tau_{i1} = one_i \cdot \epsilon + two_i \cdot a_0 = \overline{(\overline{\mathrm{one_i}} + \overline{\epsilon}) \cdot (\overline{\mathrm{two_i}} + \overline{\mathrm{a_0}})} \tag{2.16}$$

$$d_i = \overline{(\overline{\mathrm{b_{2i+1}}} + \mathrm{a_0})} \cdot \overline{[(\mathrm{b_{2i-1}} + \mathrm{a_1}) \cdot (\mathrm{b_{2i}} + \mathrm{a_0}) \cdot (\mathrm{b_{2i}} + \mathrm{b_{2i-1}})]} \tag{2.17}$$

Il valore di $\overline{\epsilon}$ si può ricavare da Equazione 2.18.

$$\overline{\epsilon} = \begin{cases} a_1, & \text{se } \overline{a_0 \cdot b_{2i+1}} = 0 \\ \overline{\mathrm{a_1}}, & \text{altrimenti} \end{cases} \tag{2.18}$$

Poiché il peso di d_i è 2^n, pari al peso di s_0 in posizione n, d_i può essere incorporato con i bit di estensione del segno $\overline{\mathrm{s_0}} s_0 s_0$ di PP_0. Siano $\alpha_2\alpha_1\alpha_0$ i nuovi bit dopo l'incorporazione di d_i in $\overline{\mathrm{s_0}} s_0 s_0$. Nella Tabella 2.5 sono riassunte le loro relazioni.

$\overline{\mathrm{s_0}}$	s_0	s_0	d_i	α_2	α_1	α_0
1	0	0	0	1	0	0
1	0	0	1	1	0	1
0	1	1	0	0	1	1
0	1	1	1	1	0	0

Tabella 2.5: Tabella della verità per i bit di estensione di segno

La matrice di prodotti parziali generato dall'approccio proposto per il moltiplicatore 8×8 è mostrato in Figura 2.11.

2.4 Riduzione dei prodotti parziali

La riduzione dei prodotti parziali durante la moltiplicazione binaria avviene sommando i prodotti parziali che si trovano nella stessa colonna, ovvero quelli che hanno la stessa posizione. Questo permette di semplificare la somma finale e di ridurre il numero di operazioni necessarie. Tale operazione è di grande importanza nella moltiplicazione binaria, in quanto consente di velocizzare il calcolo e semplificare le operazioni, essendo l'operazione che richiede un'area e un consumo di potenza più alti rispetto a tutto il resto.

Wallace [9] ha sviluppato un metodo per la moltiplicazione veloce basato su contatori paralleli. Dadda [10, 11] ha perfezionato il metodo di Wallace riconoscendo che gli pseudoadditivi sono contatori (3,2) e (2,2) e definendo una strategia

di posizionamento dei contatori che ne richiede un minor numero al costo di un sommatore carry-propagate più grande.

In ogni fase della riduzione, Wallace raggruppa preliminarmente le righe in gruppi di tre. All'interno di ogni gruppo, i contatori (3,2) vengono utilizzati per ridurre le colonne con tre bit a due bit, mentre i contatori (2,2) vengono impiegati per ridurre le colonne con soli due bit. Le righe che non fanno parte di un gruppo di tre vengono trasferite allo stadio successivo senza subire modifiche, e i bit di tali righe saranno ridotti nelle fasi successive. L'altezza della matrice nel j-esimo stadio di riduzione, w_j, è definita dalle equazioni ricorsive.
$w_0 = N$
$w_{j+1} = 2\lfloor w_j/3 \rfloor + (w_j \ mod \ 3)$

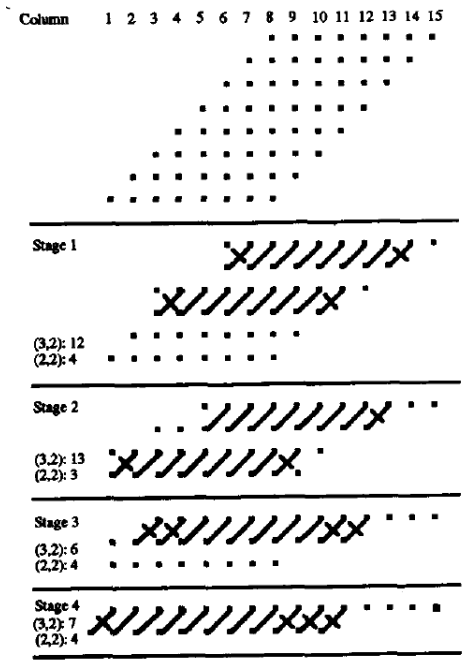


Figura 2.12: 8 per 8 Wallace Multiplier [16]

In Figura 2.12 è riportato un esempio di una moltiplicazione 8 per 8 svolta con l'algoritmo di Wallace.

Dadda ha introdotto una sequenza di altezze di matrici intermedie per l'applicazione parallela dei contatori (3,2) e (2,2), al fine di determinare il numero minimo di stadi di riduzione necessari per un moltiplicatore di dimensioni specifiche. Questa sequenza viene determinata partendo dalla matrice finale a due righe e limita l'altezza di ogni matrice intermedia al più grande numero intero che non supera di

1,5 volte l'altezza del suo successore.

Lo schema di riduzione di Dadda utilizza il seguente algoritmo [17]:

1. Sia $d_1 = 2$ e $d_{j+1} = \lfloor 3 \cdot d_j/2 \rfloor$, dove d_j è l'altezza della matrice per il j-esimo stadio dalla fine. Trovare il più grande j tale che almeno una colonna della matrice di bit abbia più di d_j bit.

2. Utilizzare i contatori (3,2) e (2,2) per ottenere una matrice ridotta con non più di d_j elementi in ogni colonna.

3. Finché non viene generata una matrice con due sole righe, lasciare $j = j - 1$ e ripetere il passaggio 2.

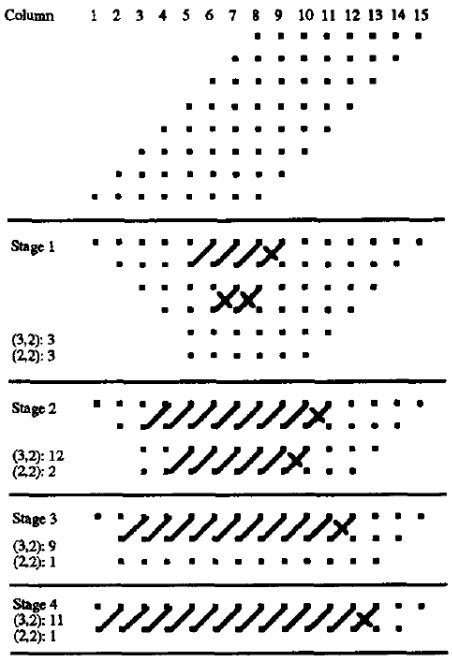


Figura 2.13: 8 per 8 Dadda Multiplier [16]

In Figura 2.13 è riportato un esempio di una moltiplicazione 8 per 8 svolta con l'algoritmo prima discusso.

Il Reduced Area Multiplier (RA) [16], un moltiplicatore che utilizza uno schema di riduzione innovativo per ridurre il numero di componenti e interconnessioni rispetto ai moltiplicatori Wallace o Dadda. Questo schema di riduzione è particolarmente utile per i moltiplicatori pipeline, in quanto minimizza il numero di latch

necessari per la riduzione dei prodotti parziali.
Il moltiplicatore Reduced Area è una versione migliorata dei metodi di Wallace e Dadda. Lo schema di riduzione si differenzia dai metodi precedenti per l'utilizzo tempestivo del numero massimo di contatori (3,2) e l'attenta posizione dei contatori (2,2) per ridurre la word size del carry-propagate adder. L'utilizzo dei contatori (3,2) il più presto possibile minimizza il numero di bit che passano tra gli stadi successivi della riduzione, riducendo così la quantità di interconnessione.

Il moltiplicatore RA esegue la riduzione come segue:
(1) Per ogni stadio, il numero di contatori (3,2) utilizzati in ogni colonna è $\lfloor b_i/3 \rfloor$, dove b_i è il numero di bit nella colonna i. Questo fornisce la massima riduzione del numero di bit che entrano nello stadio successivo.
(2) I contatori (2,2) vengono utilizzati solo (a) quando è necessario ridurre il numero di bit in una colonna al numero di bit specificato dalla serie Dadda, o (b) per ridurre la colonna più a destra contenente esattamente due bit.

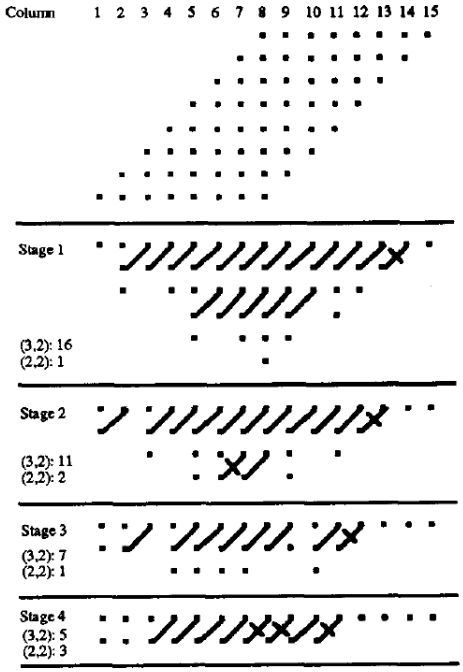
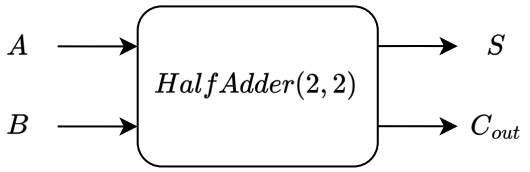


Figura 2.14: 8 per 8 Reduce Area Multiplier [16]

In Figura 2.14 è riportato un esempio di una moltiplicazione 8 per 8 svolta con il moltiplicatore RA.

Lo schema di riduzione del moltiplicatore RA sarà utilizzato per tutte le configurazioni che verranno implementate e analizzate.

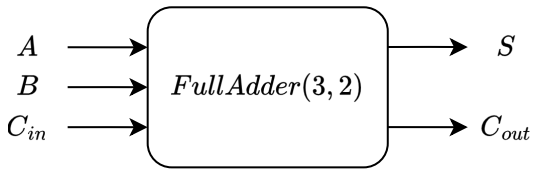
I contatori (3,2) (Full Adder, FA) e (2,2) (Half Adder, HA) sono mostrati in Figura 2.15. Ogni contatore (3,2) accetta tre ingressi da una determinata colonna e produce un bit di somma che rimane in quella colonna e un bit di riporto che va nella colonna successiva più significativa. Un contatore (2,2) accetta due ingressi da una colonna e produce un bit di somma nella stessa colonna e un bit di riporto nella colonna successiva più significativa.



$$S = A \oplus B$$

$$C_{out} = A \cdot B$$

(a)



$$S = A \oplus B \oplus C_{in}$$

$$C_{out} = A \cdot B + B \cdot C_{in} + C_{in} \cdot A$$

(b)

Figura 2.15: (a) Half Adder, (b) Full Adder

Esistono anche altri tipi di contatori (o compressori) di grado più alto che possono comprimere in modo più spinto l'albero delle somme. Uno di questi è il compressore 4:2, che può essere implementato sia per avere un risultato esatto o un risultato approssimato in uscita.
Prima di esaminare i compressori approssimati, sarà utile introdurre il compressore 4:2 esatto, che costituirà il punto di riferimento per valutare i miglioramenti che potrebbero derivare dall'adozione di circuiti approssimati. In questo modo, sarà possibile comprendere più a fondo le differenze tra i diversi approcci e valutare i vantaggi e le limitazioni di ciascuno di essi.

Il compressore esatto 4:2 (Figura 2.16), che presenta cinque ingressi di peso uguale

($x1$, $x2$, $x3$, $x4$, Tin) e tre uscite (S, C e $Tout$), potrebbe essere più propriamente definito come contatore (5,3). In particolare, l'uscita S ha lo stesso peso degli ingressi, mentre C e $Tout$ hanno un peso doppio. Questo tipo di compressore è stato progettato in modo da garantire che $Tout$ non dipenda da Tin, caratteristica che risulta particolarmente utile nei moltiplicatori ad albero. Infatti, il $Tout$ prodotto da un compressore 4:2 nella colonna i può essere collegato a Tin di un compressore nella colonna $i + 1$, senza influire sul ritardo del sistema.

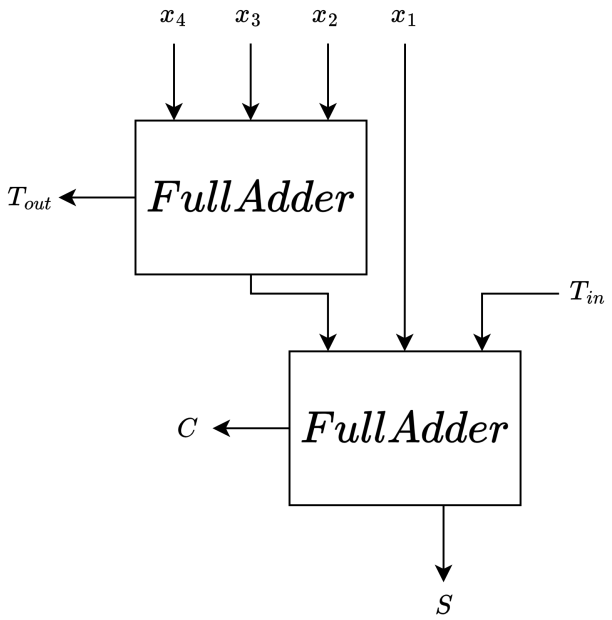


Figura 2.16: Compressore 4:2 esatto

Quello che è stato appena visto è l'implementazione convenzionale del compressore 4:2 che deve rispettare l'Equazione 2.19.

$$x_1 + x_2 + x_3 + x_4 + C_{in} = Sum + 2(carry + C_{out}) \tag{2.19}$$

Il compressore è composta da due full adder collegati in serie. A livello di gate, i compressori sono anatomizzati in porte XOR e generatori di *carry* normalmente implementati da multiplexer (MUX). Pertanto, i diversi progetti possono essere classificati in base al ritardo del percorso critico in termini di numero di porte primitive.
Sia Δ_{XOR} il ritardo di un gate XOR e Δ_{GEN} il ritardo di un generatore di *carry*. Si dice che un compressore ha un ritardo di $(m\Delta_{XOR} + n\Delta_{GEN})$ se il suo percorso critico consiste in porte XOR e generatori di *carry*. Poiché la differenza tra i ritardi dei gate XOR e dei generatori di *carry* ampiamente utilizzati è banale in

un progetto ottimizzato, il ritardo del compressore è più comunemente specificato come $(m+n)\Delta$. Pertanto, l'implementazione del compressore 4:2 ha un ritardo critico di 4Δ.

La Figura 2.17 [18] mostra un compressore 4:2 ottimizzato a livello di gate per ridurre il ritardo del percorso critico. Questo compressore è descritto con la seguente Equazione 2.20.

$$\begin{aligned} C_{out} &= (x_1 \oplus x_2) \cdot x_3 + x_1 \cdot x_2 = (x_1 \oplus x_2) \cdot x_3 + \overline{(\mathrm{x_1} \oplus \mathrm{x_2})} \cdot x_1 \\ Sum &= x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus C_{in} \\ carry &= (x_1 \oplus x_2 \oplus x_3 \oplus x_4) \cdot C_{in} + \overline{(\mathrm{x_1} \oplus \mathrm{x_2} \oplus \mathrm{x_3} \oplus \mathrm{x_4})} \cdot x_4 \end{aligned} \tag{2.20}$$

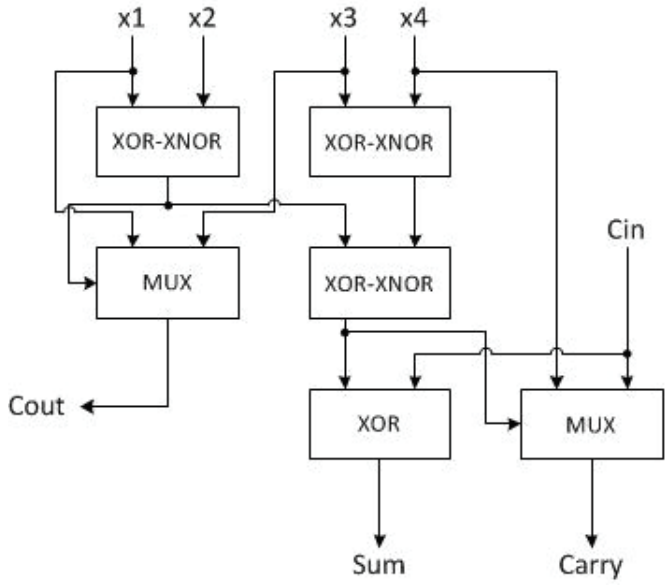


Figura 2.17: Compressore 4:2 presentato in [18]

I due segnali di riporto *carry* e C_{out} sono generati dalle funzioni XOR e $XNOR$ dei segnali di ingresso. L'uscita è generata da diversi circuiti XOR a due ingressi, alcuni segnali interni dei quali possono essere utilizzati per generare i due *carry*. È composta principalmente da sei moduli, di cui quattro sono circuiti XOR e gli altri due sono $MUX\ 2-1$. Tre speciali moduli $XOR-XNOR$ generano simultaneamente i segnali XOR e $XNOR$ per gli altri moduli da essi pilotati. Questo progetto ha un ritardo del percorso critico di 3Δ, che è inferiore di 1Δ rispetto all'implementazione convenzionale. Inoltre, le sue uscite presentano un tempo di arrivo del segnale bilanciato da ciascun ingresso dati (da x_1 a x_2), grazie

agli speciali moduli $XOR - XNOR$.

Diversi autori hanno utilizzato negli ultimi anni un'altra struttura di compressori 4:2 che può essere considerata la migliore e che è stata proposta in [19]. Questa struttura è una versione modificata dell'architettura della Figura 2.17 e viene illustrata nella Figura 2.18.

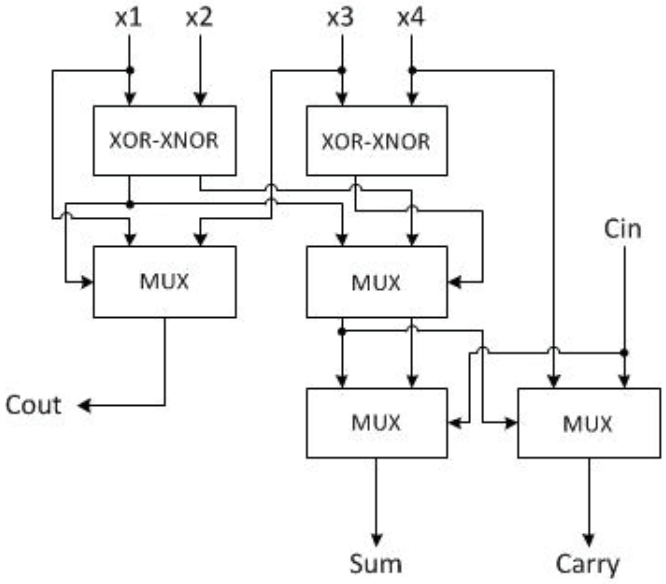


Figura 2.18: Compressore 4:2 presentato in [19]

Le equazioni che regolano le uscite in questa architettura sono identiche a quelle menzionate in precedenza per la versione del compressore 4:2 precedente.

Alcune strutture di compressori hanno invece impiegato blocchi generatori di *carry* ($CGEN$) per migliorare le prestazioni. Un $CGEN$ è un blocco che riceve tre ingressi e produce un segnale di riporto in base alla Tabella 2.6 di [20], che rappresenta la tabella di verità di questo blocco. L'uscita del blocco $CGEN$ viene descritta dall'Equazione 2.21:

$$carry = (a + b) \cdot C_{in} + a \cdot b \tag{2.21}$$

Questa architettura, che utilizza il blocco $CGEN$ per produrre le uscite C_{out} e $carry$, è illustrata nella Figura 2.19.
Nella versione precedente, i segnali $x_1 \oplus x_2$ e $x_3 \oplus x_4$ erano prodotti inizialmente da due blocchi separati. Successivamente, le uscite di questi blocchi venivano alimentate agli ingressi di un altro blocco per produrre il segnale $x_1 \oplus x_2 \oplus x_3 \oplus x_4$, che veniva

a	b	Cin	carry
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

Tabella 2.6: Tabella di verità del blocco CGEN

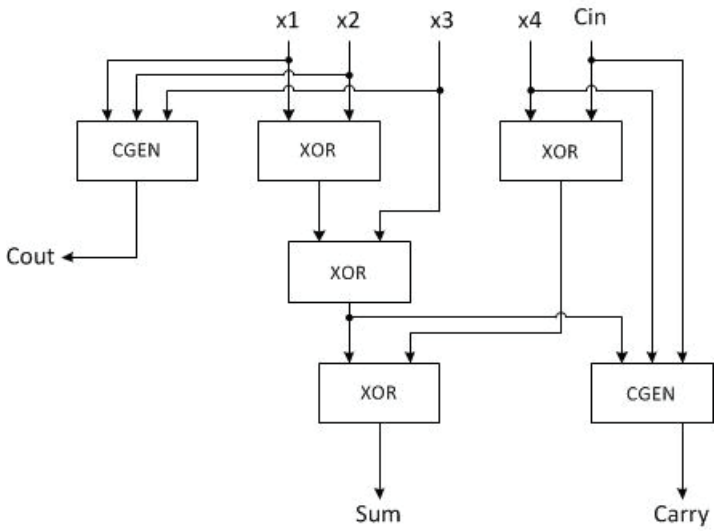


Figura 2.19: Compressore 4:2 che utilizza il blocco *CGEN*

utilizzato per generare il segnale *carry* tramite un blocco MUX. Nell'ultima architettura proposta, invece, il segnale $x_1 \oplus x_2$ viene prodotto da una porta XOR e il segnale $x_1 \oplus x_2 \oplus x_3$ viene prodotto da un altro gate XOR. Quest'ultimo segnale alimenta poi il blocco $CGEN$ per generare il segnale *carry*.
L'utilizzo del blocco $CGEN$ per generare il segnale C_{out} elimina la necessità di attendere il segnale $x_1 \oplus x_2$, riducendo così il ritardo di produzione del segnale C_{out}. Di conseguenza, il meccanismo di generazione dei segnali di uscita di *carry* differisce da quello delle architetture esistenti. Le funzioni di uscita dell'ultima architettura sono descritte dall'Equazione 2.22:

$$\begin{aligned} C_{out} &= (x_1 + x_2) \cdot x_3 + x_1 + x_2 \\ Sum &= x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus C_{in} \\ carry &= (x_4 + C_{in}) \cdot (x_1 \oplus x_2 \oplus x_3) + x_4 \cdot C_{in} \end{aligned} \tag{2.22}$$

2.5 Design dei moltiplicatori e dell'albero delle somme

Tenendo in considerazione che i valori in ingresso ai moltiplicatori sono in complemento a due, sono stati utilizzati due algoritmi differenti per la generazione dei prodotti parziali: il Baugh-Wooley [8] e il Modified Booth Endocoding [14]. L'obiettivo è quello di confrontare i risultati ottenuti e scegliere la soluzione migliore per l'architettura in questione.

Il filtro FIR utilizzato nel FFE è composto da 22 taps, ciò vuol dire che al suo interno ci saranno 22 moltiplicatori. Questi però non hanno tutti lo stesso parallelismo, infatti i moltiplicatori sono:

- 10 moltiplicatori: 5x7
- 6 moltiplicatori: 6x7
- 2 moltiplicatori: 7x7
- 2 moltiplicatori: 8x7
- 2 moltiplicatori: 9x7

che in base al tipo di algoritmo utilizzato daranno luogo a diversi numeri di prodotti parziali.
Il valore che cambia è riferito ai coefficienti dei diversi moltiplicatori.

2.5.1 Abero con Baugh-Wooley, schema di riduzione Reduced Area

Moltiplicatore 9x7:

Y : [Y8 Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0]
X : [X6 X5 X4 X3 X2 X1 X0]

14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X6Y8	0	X5Y7	X4Y7	X3Y7	X2Y7	X1Y7	X0Y7	X0Y6	X0Y5	X0Y4	X0Y3	X0Y2	X0Y1	X0Y0
1	X6!Y7	X6!Y6	X5Y6	X4Y6	X3Y6	X2Y6	X1Y6	X1Y5	X1Y4	X1Y3	X1Y2	X1Y1	X1Y0	
1	!X5Y8	!X4Y8	X6!Y5	X5Y5	X4Y5	X3Y5	X2Y5	X2Y4	X2Y3	X2Y2	X2Y1	X2Y0		
!X6			!X3Y8	X6!Y4	X5Y4	X4Y4	X3Y4	X3Y3	X3Y2	X3Y1	X3Y0			
!Y8				!X2Y8	X6!Y3	X5Y3	X4Y3	X4Y2	X4Y1	X4Y0				
					!X1Y8	X6!Y2	X5Y2	X5Y1	X5Y0					
						!X0Y8	X5!Y1	X6!Y0						
						Y8		X6						

Tabella 2.7: Prodotti parziali del moltiplicatore Baugh-Wooley 9 x 7

Moltiplicatore 8x7:

Y : [Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0]
X : [X6 X5 X4 X3 X2 X1 X0]

13	12	11	10	9	8	7	6	5	4	3	2	1	0
X6Y7	0	X5Y6	X4Y6	X3Y6	X2Y6	X1Y6	X0Y6	X0Y5	X0Y4	X0Y3	X0Y2	X0Y1	X0Y0
1	X6!Y6	X6!Y5	X5Y5	X4Y5	X3Y5	X2Y5	X1Y5	X1Y4	X1Y3	X1Y2	X1Y1	X1Y0	
1	!X5Y7	!X4Y7	X6!Y4	X5Y4	X4Y4	X3Y4	X2Y4	X2Y3	X2Y2	X2Y1	X2Y0		
!X6			!X3Y7	X6!Y3	X5Y3	X4Y3	X3Y3	X3Y2	X3Y1	X3Y0			
!Y7				!X2Y7	X6!Y2	X5Y2	X4Y2	X4Y1	X4Y0				
					!X1Y7	X6!Y1	X5Y1	X5Y0					
						!X0Y7	X6!Y0						
						Y7	X6						

Tabella 2.8: Prodotti parziali del moltiplicatore Baugh-Wooley 8 x 7

Moltiplicatore 7x7:

Y : [Y6 Y5 Y4 Y3 Y2 Y1 Y0]
X : [X6 X5 X4 X3 X2 X1 X0]

12	11	10	9	8	7	6	5	4	3	2	1	0
X6Y6	0	X5Y5	X4Y5	X3Y4	X2Y5	X1Y5	X0Y5	X0Y4	X0Y3	X0Y2	X0Y1	X0Y0
1	X6!Y5	X6!Y4	X5Y4	X4Y4	X3Y4	X2Y4	X1Y4	X1Y3	X1Y2	X1Y1	X1Y0	
1	!X5Y6	!X4Y6	X6!Y3	X5Y3	X4Y3	X3Y3	X2Y3	X2Y2	X2Y1	X2Y0		
!X6			!X3Y6	X6!Y2	X5Y2	X4Y2	X3Y2	X3Y1	X3Y0			
!Y6				!X2Y6	X6!Y1	X5Y1	X4Y1	X4Y0				
					!X1Y6	X6!Y0	X5Y0					
						!X0Y6						
						X6						
						Y6						

Tabella 2.9: Prodotti parziali del moltiplicatore Baugh-Wooley 7 x 7

Moltiplicatore 6x7:

Y : [Y5 Y4 Y3 Y2 Y1 Y0]
X : [X6 X5 X4 X3 X2 X1 X0]

11	10	9	8	7	6	5	4	3	2	1	0
X6Y5	0	X5Y4	X4Y4	X3Y4	X2Y4	X1Y4	X0Y4	X0Y3	X0Y2	X0Y1	X0Y0
1	X6!Y4	X6!Y3	X5Y3	X4Y3	X3Y3	X2Y3	X1Y3	X1Y2	X1Y1	X1Y0	
1	!X5Y5	!X4Y5	X6!Y2	X5Y2	X4Y2	X3Y2	X2Y2	X2Y1	X2Y0		
!X6			!X3Y5	X6!Y1	X5Y1	X4Y1	X3Y1	X3Y0			
!Y5				!X2Y5	X6!Y0	X5Y0	X4Y0				
					!X1Y5	!X0Y5					
					X6	Y5					

Tabella 2.10: Prodotti parziali del moltiplicatore Baugh-Wooley 6 x 7

Moltiplicatore 5x7:

Y : [Y4 Y3 Y2 Y1 Y0]
X : [X6 X5 X4 X3 X2 X1 X0]

10	9	8	7	6	5	4	3	2	1	0
X6Y4	0	X5Y3	X4Y3	X3Y3	X2Y3	X1Y3	X0Y3	X0Y2	X0Y1	X0Y0
1	X6!Y3	X6!Y2	X5Y2	X4Y2	X3Y2	X2Y2	X1Y2	X1Y1	X1Y0	
1	!X5Y4	!X4Y4	X6!Y1	X5Y1	X4Y1	X3Y1	X2Y1	X2Y0		
!X6			!X3Y4	X6!Y0	X5Y0	X4Y0	X3Y0			
!Y4				!X2Y4	!X1Y4	!X0Y4				
				X6		Y4				

Tabella 2.11: Prodotti parziali del moltiplicatore Baugh-Wooley 5 x 7

Le costanti all'interno dei prodotti parziali non sono stati inseriti all'ingresso dell'albero di compressione ma sono state sommate per creare una costante che verrà poi aggiunta alle ultime due righe dello stage finale quando verrà calcolata l'uscita del FIR.
Nella (Tabella 2.12) sono mostrati il numero di bit in uscita dai moltiplicatori.
L'albero delle somme viene riportato nella Tabella 2.13 dove sono presentati il numero di bit che verranno compressi seguendo lo schema di riduzione del moltiplicatore RA, utilizzando compressori (3,2) e compressori (2,2) in modo opportuno, fino ad arrivare ad uno stage con altezza 2.

bit	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
moltiplicatore 9x7	3	2	3	4	5	6	8	7	8	6	5	4	3	2	1
moltiplicatore 8x7		3	2	3	4	5	6	8	8	6	5	4	3	2	1
moltiplicatore 7x7			3	2	3	4	5	6	9	6	5	4	3	2	1
moltiplicatore 6x7				3	2	3	4	5	7	7	5	4	3	2	1
moltiplicatore 5x7					3	2	3	4	6	5	6	4	3	2	1

Tabella 2.12: Bit totali in uscita dai moltiplicatori

In ingresso si hanno 1026 bit, e con questa configurazione sono stati utilizzati 997 compressori (3,2) e 9 compressori (2,2), che hanno permesso di comprimere un totale di 3166 bit.
L'uscita dell'albero di compressione ha una lunghezza di 18 bit, che è il numero di bit che serve per rappresentare la somma dei massimi valori in uscita da tutti i moltiplicatori.

2.5.2 Abero con MBE, schema di riduzione Reduced Area

Dopo aver inserito tutti i bit in uscita da ogni singolo moltiplicatore (Tabella 2.14) all'interno dell'albero di compressione, si otterrà al suo ingresso un numero di bit pari a 652.
Nella Tabella 2.15 sono riportati il numero di bit all'ingresso di ogni stage e il numero di FA ed HA utilizzati per la compressione tra uno stage e il prossimo.
Con questa configurazione sono stati utilizzati 625 compressori (3,2) e 7 compressori (2,2), che hanno permesso di comprimere un totale di 2039 bit.

2.5.3 Albero con MBE, schema di riduzione Reduced Area e compressori 4:2 esatti

Per lo studio in questione verrà utilizzato il compressore 4:2 esatto composto da due full adder in cascata (Figura 2.16).
L'utilizzo di questo tipo di compressore ha causato una variazione nel numero di componenti all'interno dell'albero delle somme, rispetto al caso precedente. Tuttavia, nonostante questa variazione, il numero dei bit in ingresso all'albero è rimasto invariato poiché per la generazione dei prodotti parziali si è utilizzato nuovamente l'algoritmo di Booth.
Durante la compressione, è stata apportata una piccola modifica al primo stage, tenendo conto della progettazione dei tap del filtro FIR: i primi 12 tap sono fissi, mentre gli ultimi 10 possono essere spostati in due gruppi da 5. Ciò comporta la presenza di due multiplexer a valle di questi registri, che introducono un ritardo

bit	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STAGE 1	0	0	0	6	10	16	36	66	68	92	112	152	128	120	88	66	44	22
N° FA	0	0	0	2	3	5	12	22	22	30	37	50	42	40	29	22	14	7
N° HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 2	0	0	2	5	9	18	34	44	54	69	88	94	84	69	52	36	23	8
N° FA	0	0	0	1	3	6	11	14	18	23	29	31	28	23	17	12	7	2
N° HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 3	0	0	3	6	9	17	26	34	41	52	61	60	51	40	30	19	11	4
N° FA	0	0	1	2	3	5	8	11	13	17	20	20	17	13	10	6	3	1
N° HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 4	0	1	3	5	8	15	21	25	32	38	41	37	30	24	16	10	6	2
N° FA	0	0	1	1	2	5	7	8	10	12	13	12	10	8	5	3	2	0
N° HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
STAGE 5	0	2	2	5	9	12	15	19	24	27	27	23	18	13	9	6	3	1
N° FA	0	0	0	1	3	4	5	6	8	9	9	7	6	4	3	2	1	0
N° HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 6	0	2	3	6	7	9	11	15	17	18	16	15	10	8	5	3	1	1
N° FA	0	0	1	2	2	3	3	5	5	6	5	5	3	2	1	1	0	0
N° HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 7	0	3	3	4	6	6	10	10	13	11	11	8	6	5	4	1	1	1
N° FA	0	1	1	1	2	2	3	3	4	3	3	2	2	1	1	0	0	0
N° HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 8	1	2	2	4	4	5	7	8	8	8	7	6	3	4	2	1	1	1
N° FA	0	0	0	1	1	1	2	2	2	2	2	2	1	1	0	0	0	0
N° HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
STAGE 9	1	2	3	3	3	5	5	6	6	6	5	3	2	3	1	1	1	1
N° FA	0	0	1	1	1	1	1	2	2	2	1	1	0	1	0	0	0	0
N° HA	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
STAGE 10	1	3	2	2	3	4	4	4	4	3	4	1	3	1	1	1	1	1
N° FA	0	1	0	0	1	1	1	1	1	1	1	0	1	0	0	0	0	0
N° HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 11	2	1	2	3	2	3	3	3	3	2	2	2	1	1	1	1	1	1
N° FA	0	0	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0
N° HA	0	0	1	0	1	0	0	0	0	1	1	1	0	0	0	0	0	0
STAGE 12	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1

Tabella 2.13: Albero di compressione con algoritmo Baugh Wooley

stimato pari a due full adder in cascata. Per mantenere lo stesso ritardo dei primi 12 tap, gli ultimi 10 non sono stati compressi nel primo stage, ma sono stati portati avanti fino alla stage successivo, dove le uscite dello stage precedente hanno attraversato 2 full adder, come previsto dal compressore 4:2 esatto.

bit	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
moltiplicatore 9x7	1	2	2	3	3	5	4	5	4	4	3	3	2	2	1
moltiplicatore 8x7		1	1	2	2	3	3	5	4	4	3	3	2	2	1
moltiplicatore 7x7			1	2	2	3	3	5	4	4	3	3	2	2	1
moltiplicatore 6x7				1	1	2	2	3	3	4	3	3	2	2	1
moltiplicatore 5x7					1	2	2	3	3	4	3	3	2	2	1

Tabella 2.14: Bit totali in uscita dai moltiplicatori

Questa logica è stata in seguito applicata anche all'albero precedente, al fine di confrontarlo con l'albero attuale e quelli successivi.
La Tabella 2.16 riporta il numero di bit per ogni stage, insieme ai relativi numeri di compressori 4:2 esatti, FA e HA utilizzati per la compressione dell'albero.
Il numero di compressori 4:2 esatti è di 332, mentre quello per i FA è di 4 e quello per gli HA è di 5. Poiché ogni compressore 4:2 esatto è composto da 2 FA in cascata, il numero totale di FA è di 668.

2.6 Confronto

Nella Tabella 2.17 sono riportati il numero di bit, di full adder e di half dder utilizzati nelle diverse implementazioni. Tutti i moltiplicatori utilizzano lo schema di riduzione del Reduced Area.
Per determinare la soluzione ottimale, sono state effettuate sintesi e simulazioni al fine di valutare l'area occupata dai vari circuiti e la loro potenza. Per aumentare il confronto si è svolta anche una sintesi del filtro andando a sostituire i moltiplicatori e l'albero delle somme rispettivamente con i segni matematici * e +, lasciando al sintetizzatore la scelta dei moltiplicatori e dei sommatori da assegnare al circuito.
I moltiplicatori che che non utilizzano i compressori 4:2 esatti, sfruttano i contatore 3:2 (FA) e 2:2 (HA) per comprimere l'albero delle somme.
Da qui si può notare come l'utilizzo dell'algoritmo di Booth porti ad avere dei vantaggi.
I risultati sono riportati nella Tabella 2.18.
In base ai risultati riportati nelle tabelle precedenti si è scelto di utilizzare l'algoritmo MBE per la generazione dei prodotti parziali in quanto può ridurre l'area e la potenza del sistema.

bit	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STAGE 1	0	0	0	2	6	8	20	30	54	52	78	72	88	66	66	44	44	22
FA	0	0	0	0	2	2	6	10	18	17	26	24	29	22	22	14	14	7
HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 2	0	0	0	4	4	10	18	28	35	44	50	53	52	44	36	30	23	8
FA	0	0	0	1	1	3	6	9	11	14	16	17	17	14	12	10	7	2
HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 3	0	0	1	3	5	10	15	21	27	32	35	36	32	28	22	17	11	4
FA	0	0	0	1	1	3	5	7	9	10	11	12	10	9	7	5	3	1
HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 4	0	0	2	2	6	9	12	16	19	23	25	22	21	17	13	10	6	2
FA	0	0	0	0	2	3	4	5	6	7	8	7	7	5	4	3	2	0
HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
STAGE 5	0	0	2	4	5	7	9	12	14	17	16	15	12	11	8	6	3	1
FA	0	0	0	1	1	2	3	4	4	5	5	5	4	3	2	2	1	0
HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 6	0	0	3	3	5	6	7	8	11	12	11	9	7	7	6	3	1	1
FA	0	0	1	1	1	2	2	2	3	4	3	3	2	2	2	1	0	0
HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 7	0	1	2	2	5	4	5	7	9	7	8	5	5	5	3	1	1	1
FA	0	0	0	0	1	1	1	2	3	2	2	1	1	1	1	0	0	0
HA	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
STAGE 8	0	1	2	3	4	3	5	6	5	5	5	4	5	3	1	1	1	1
FA	0	0	0	1	1	1	1	2	1	1	1	1	1	1	0	0	0	0
HA	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0
STAGE 9	0	1	3	2	3	3	4	3	4	4	4	4	3	1	1	1	1	1
FA	0	0	1	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0
HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 10	0	2	1	3	2	2	3	2	3	3	3	3	1	1	1	1	1	1
FA	0	0	0	1	0	0	1	0	1	1	1	1	0	0	0	0	0	0
HA	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0
STAGE 11	0	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1

Tabella 2.15: Albero di compressione con algoritmo MBE

bit	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STAGE 1	0	0	0	2	6	8	20	30	54	52	78	72	88	66	66	44	44	22
TAP 0-11	0	0	0	2	6	8	17	20	34	32	48	42	48	36	36	24	24	12
N 4:2	0	0	0	0	1	2	4	5	8	8	12	10	12	9	9	6	6	3
N FA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 2	0	0	0	4	6	8	15	24	39	42	52	56	61	48	45	32	29	13
N 4:2	0	0	0	1	1	2	3	6	9	10	13	14	15	12	11	8	7	3
N FA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 3	0	0	2	2	6	7	15	16	24	26	28	30	28	23	20	15	11	4
N 4:2	0	0	0	0	1	1	3	4	6	6	7	7	7	5	5	3	2	1
N FA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
N HA	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0
STAGE 4	0	0	2	4	4	9	11	12	13	15	15	15	12	13	8	9	4	1
N 4:2	0	0	0	1	1	2	2	3	3	3	3	3	3	3	2	2	1	0
N FA	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0
N HA	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
STAGE 5	0	0	4	2	4	8	8	8	8	8	8	7	6	6	4	4	1	1
N 4:2	0	0	1	0	1	2	2	2	2	2	2	1	1	1	1	1	0	0
N FA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N HA	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
STAGE 6	0	4	4	4	4	4	4	4	4	4	4	4	4	4	2	1	1	1
N 4:2	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
N FA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
STAGE 7	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1

Tabella 2.16: Albero di compressione con compressori 4:2 esatti

	Bit in ingresso	Bit totali	N° FA	N° HA	N° Stage
Baugh-Wooley	1026	3166	997	9	12
MBE	652	2039	625	7	11
MBE con compressori 4:2 esatti	652	1674	668	5	7

Tabella 2.17: Confronto numero bit e numero compressori

	Area [μm^2]	Potenza [W]
Sintetizzatore	18201.365	0.0829993
Baugh-Wooley	14571.415	0.0930701
MBE	13666.465	0.0670953
MBE con compressori 4:2 esatti	14306.202	0.0731191

Tabella 2.18: Confronto area e potenza

Capitolo 3

Filtro approssimato

L'approssimazione a livello hardware è stata studiata soprattutto per le unità aritmetiche, come i sommatori e i moltiplicatori, che sono spesso molto esigenti in termini di energia.
Numerose strategie sono state avanzate per sviluppare moltiplicatori approssimati di alta efficienza [21]. Tali moltiplicatori si basano sull'uso di moduli elementari 2x2, combinati insieme per formare moltiplicatori di dimensioni maggiori n x n [22, 23, 24]. Alcuni studi [25, 26, 27] hanno esplorato la possibilità di approssimare i risultati di moltiplicatori n x n impiegando moltiplicatori più piccoli di dimensione m x m, dove m è inferiore a n. Il riferimento [28] descrive un metodo che approssima la generazione di prodotti parziali, evitando il calcolo di alcuni di essi all'interno del moltiplicatore. I moltiplicatori basati su logaritmi [29], [30] operano sommando un'approssimazione dei logaritmi degli operandi e successivamente trovando una stima dell'antilogaritmo della somma ottenuta. I moltiplicatori troncati [31], [32] non formano alcuni prodotti parziali e riducono l'errore di troncamento attraverso l'uso di specifiche funzioni di correzione. Esistono poi metodi che consentono la configurazione in tempo reale dell'approssimazione [33], l'impiego di una codifica approssimativa di Booth [34], l'uso di moltiplicatori binari ridondanti approssimati [35] e l'integrazione di diverse tecniche di approssimazione [36].

L'uso di compressori approssimati è diventato una valida alternativa per implementare moltiplicatori approssimati. I compressori più comunemente utilizzati sono il full adder, l'half adder e quelli di ordine superiore, come 4:2 o 5:3. Anche compressori 6:3 e 7:3 efficienti dal punto di vista hardware sono stati sviluppati. Tuttavia, i moltiplicatori approssimati possono essere ottenuti sostituendo alcuni dei compressori esatti con circuiti più semplici che introducono alcuni errori ma offrono un guadagno di efficienza in termini di potenza, velocità e area.

3.1 Compressori 4:2 approssimati

Ci sono diverse tecniche per progettare compressori approssimati, tra cui l'utilizzo di porte OR come contatori approssimati e la creazione di una nuova famiglia di compressori approssimati senza uscite di riporto. Questo studio si concentra sui compressori 4:2 e presenta una rassegna e un confronto delle diverse tecniche proposte per progettare moltiplicatori approssimati con compressori approssimati 4:2, focalizzandosi sulle architetture impiegate nei moltiplicatori ad albero.

Nei compressori approssimati 4:2 proposti i pin T_{in} e T_{out} (o C_{in} e C_{out}) non sono utilizzati per semplificare l'implementazione del circuito.
Il valore massimo che può essere codificato utilizzando solo le uscite S e C è 3.
Avendo quattro ingressi x_1, x_2, x_3, x_4 è ovvio che almeno un errore (quando tutti gli ingressi sono '1') è inevitabile.
Ora andiamo a proporre i 12 compressori approssimati [37] che sono stati utilizzati in questo studio.

3.1.1 Momeni

La Figura 3.1 mostra l'implementazione a livello di porte logiche di questo compressore 4:2 approssimato [38] e le espressioni di seguito descrivono le sue uscite.

$$Sum = (\overline{x_1 \oplus x_2} + \overline{x_3 \oplus x_4}) \tag{3.1}$$

$$Carry = \overline{(\overline{x_1 x_2} + \overline{x_3 x_4})} \tag{3.2}$$

La Tabella 3.1 mostra la tabella di verità del compressore 4:2.
Questa Tabella 3.1 mostra anche la differenza tra il valore decimale esatto dell'addizione degli ingressi e il valore decimale delle uscite prodotte dal compressore approssimato. Per esempio, quando tutti gli ingressi sono 1, il valore decimale dell'addizione degli ingressi è 4. Tuttavia, il compressore approssimato produce un 1 per il riporto e la somma. In questo caso il valore decimale delle uscite è 3. Questo design ha quindi quattro uscite errate su 16 uscite, quindi il suo tasso di errore è ridotto al 25%.

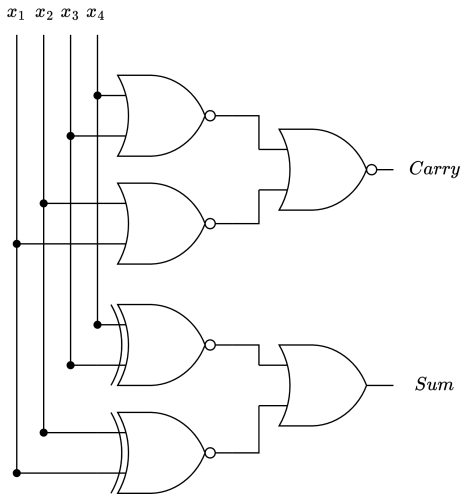


Figura 3.1: Compressore 4:2 approssimato Momeni

x_4	x_3	x_2	x_1	C	S	$Differenza$
0	0	0	0	0	1	1
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	1	-1
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	1	0	0
0	1	1	1	1	1	0
1	0	0	0	0	1	0
1	0	0	1	1	0	0
1	0	1	0	1	0	0
1	0	1	1	1	1	0
1	1	0	0	0	1	-1
1	1	0	1	1	1	0
1	1	1	0	1	1	0
1	1	1	1	1	1	-1

Tabella 3.1: Tabella di verità compressore 4:2 Momeni

3.1.2 Venka

Nel compressore 4:2 (Figura 3.2) [39], tre bit sono necessari per l'uscita solo quando tutti e quattro gli ingressi sono 1, il che accade solo una volta su 16 casi. Questa

proprietà viene presa in considerazione per eliminare uno dei tre bit di uscita nel compressore. Per mantenere la differenza minima di errore pari a uno, l'uscita "100" (il valore di 4) per quattro ingressi che sono uno deve essere sostituita con l'uscita "11" (il valore di 3). Per il calcolo della somma, una delle tre porte XOR viene sostituita con una porta OR. Inoltre, per fare in modo che la somma corrisponda al caso in cui tutti gli ingressi sono uno, all'espressione della somma viene aggiunto un circuito supplementare $x_1 \cdot x_2 \cdot x_3 \cdot x_4$. Ciò comporta un errore in cinque casi su sedici. La funzione $Carry$ è semplificata come nell' Equazione 3.3. La tabella di verità corrispondente è riportata nella Tabella 3.2.

$$\begin{aligned} W_1 &= x_1 \cdot x_2 \\ W_2 &= x_3 \cdot x_4 \\ Sum &= (x_1 \oplus x_2) + (x_3 \oplus x_4) + W_1 \cdot W_2 \\ Carry &= W_1 + W_2 \end{aligned} \tag{3.3}$$

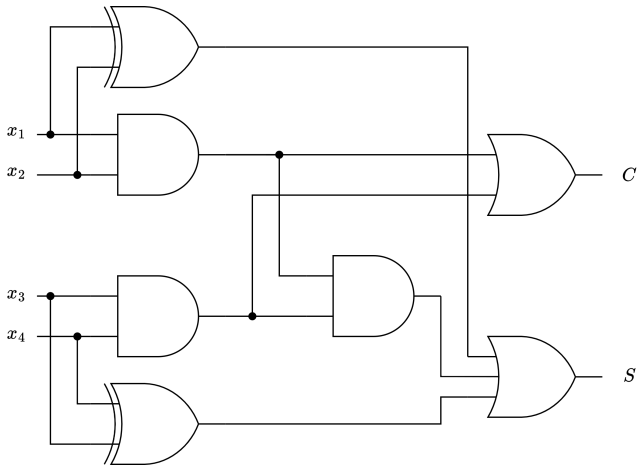


Figura 3.2: Compressore 4:2 approssimato Venka

x_4	x_3	x_2	x_1	C	S	$Differenza$
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	1	0	0
0	1	0	0	0	1	0
0	1	0	1	0	1	1
0	1	1	0	0	1	-1
0	1	1	1	1	1	0
1	0	0	0	0	1	0
1	0	0	1	0	1	-1
1	0	1	0	0	1	-1
1	0	1	1	1	1	0
1	1	0	0	1	0	0
1	1	0	1	1	1	0
1	1	1	0	1	1	0
1	1	1	1	1	1	-1

Tabella 3.2: Tabella di verità compressore 4:2 Venka

3.1.3 Yang1 Yang2 Yang3

Questa sezione descrive la progettazione di compressori approssimati con il vincolo di un basso tasso di errore [40]. Si suppone che gli ingressi di un moltiplicatore siano distribuiti uniformemente, il che significa che la probabilità che un prodotto parziale (PP) generato da una porta AND sia uguale a '1' ('0') è rispettivamente di 1/4 (3/4). Tuttavia, la probabilità che PP sia '0' è molto più alta di quella che sia '1'. Quando gli ingressi sono tutti 1, l'uscita effettiva richiede tre bit, ma viene ignorato un bit e vengono invece utilizzati due bit come $Carry$ e Sum (indicati come CS). Si noti che per $x_1 - x_4$ la probabilità di '1' logico è 1/4.
È possibile ridurre l'ER (Error Rate) modificando le voci nella riga quando x_3x_4 è '11' o modificando la colonna per $x_1x_2 =' 11'$, poiché la tabella nella Figura 3.3 è simmetrica lungo la diagonale. Sulla base di questa approccio, sono stati progettati tre compressori approssimati.
In un compressore accurato, quando gli ingressi sono tutti "1", l'uscita è "100", mentre questa uscita viene modificata in "11" in $ACCI1$ (Figura 3.4) con un ER di 1/256. Poiché il $Carry$ è più importante della Sum ai fini dell'accuratezza, in tutti i progetti proposti il $Carry$ è fissato a "1" quando gli ingressi sono tutti 1.
Nell' Equazione 3.4 e nell' Equazione 3.5 vengono riportare le espressioni delle sue uscite.

CS	X_1X_2 00	01	11	10
X_3X_4 00	00	01	10	01
01	01	10	11	10
11	10	11	**1**00	11
10	01	10	11	10

Figura 3.3: Tabella di verità di un compressore accurato senza C_{in} e C_{out}

$$\begin{aligned} inter1 &= \overline{x_3 \oplus x_4} \\ inter2 &= \overline{(x_1 + x_2) \cdot (\overline{x_1 \cdot x_2})} \\ inter3 &= \overline{x_1 \cdot x_2} \\ inter4 &= \overline{x_3} \end{aligned} \tag{3.4}$$

$$\begin{aligned} Carry &= \overline{(\text{inter1} + \text{inter2}) \cdot \text{inter3} \cdot (\text{inter4} + \overline{x_4})} \\ Sum &= \overline{(\text{inter3} + \text{inter4}) \cdot (\overline{\text{inter1} \oplus \text{inter2}})} \end{aligned} \tag{3.5}$$

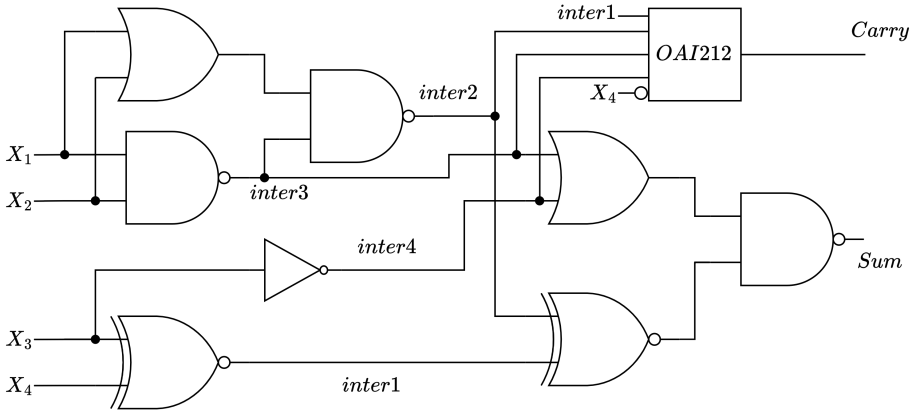


Figura 3.4: Compressore 4:2 approssimato Yang1

Tuttavia, la tabella di verità modificata dà luogo a una somma piuttosto complessa, $Sum = (x_1 \oplus x_2) \oplus (x_3 \oplus x_4) + x_1x_2x_3x_4$, dove l'ultimo termine è dovuto all'ingresso '1111'. Per ridurre la complessità in $ACCI2$, la somma viene modificata in '1' per l'ingresso '0011' (Figura 3.5). Di conseguenza, il primo termine diventa $(x_1 \oplus x_2) \oplus (x_3 + x_4)$ e l'ultimo termine diventa x_3x_4 per Sum, cioè $Sum = (x_1 \oplus x_2) \oplus (x_3 + x_4) + x_3x_4)$.
Nell' Equazione 3.6 e nell' Equazione 3.7 vengono riportate le sue espressioni.

$$\begin{aligned} inter1 &= x_1 \\ inter2 &= x_2 \\ inter3 &= x_1 \oplus x_2 \\ inter4 &= x_3 + x_4 \\ inter5 &= x_3 \\ inter6 &= x_4 \end{aligned} \tag{3.6}$$

$$\begin{aligned} Carry &= (inter1 \cdot inter2) + (inter3 \cdot inter4) + (inter5 \cdot inter6) \\ Sum &= (inter3 \oplus inter4) + (inter5 \cdot inter6) \end{aligned} \tag{3.7}$$

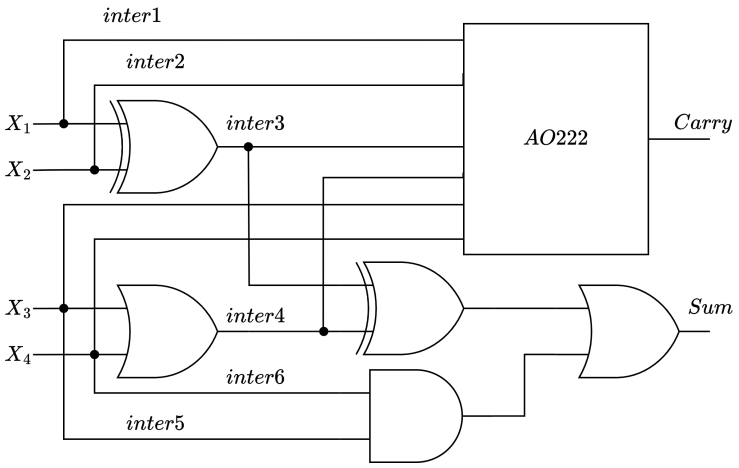


Figura 3.5: Compressore 4:2 approssimato Yang2

Il terzo compressore, $ACCI3$ (Figura 3.6), si basa su $ACCI2$ per ridurre ulteriormente la complessità logica per la generazione della Sum, eliminando l'ultimo termine, come mostrano l' Equazione 3.8 e l' Equazione 3.9.

$$\begin{aligned} inter1 &= x_1 \\ inter2 &= x_2 \\ inter3 &= x_1 \oplus x_2 \\ inter4 &= x_3 + x_4 \\ inter5 &= x_3 \\ inter6 &= x_4 \end{aligned} \tag{3.8}$$

$$\begin{aligned} Carry &= (inter1 \cdot inter2) + (inter3 \cdot inter4) + (inter5 \cdot inter6) \\ Sum &= inter3 \oplus inter4 \end{aligned} \tag{3.9}$$

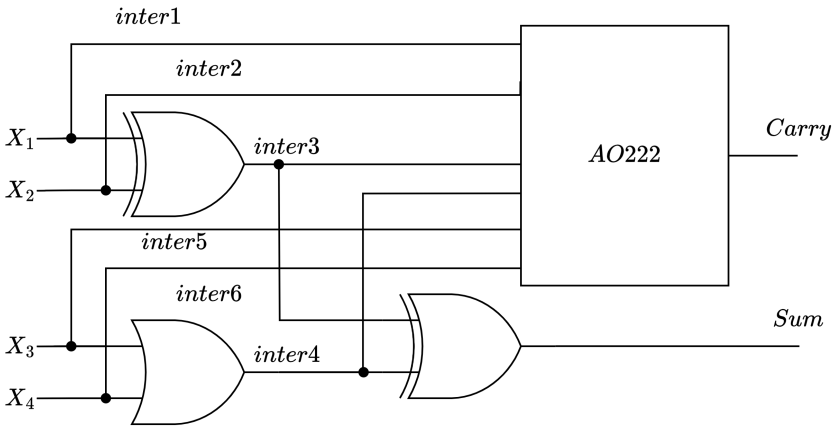


Figura 3.6: Compressore 4:2 approssimato Yang3

La tabella di verità corrispondente è riportata nella Tabella 3.3.

				Yang1			Yang2			Yang3		
x_4	x_3	x_2	x_1	C	S	D	C	S	D	C	S	D
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	1	0	0	1	0
0	0	1	0	0	1	0	0	1	0	0	1	0
0	0	1	1	1	0	0	1	0	0	1	0	0
0	1	0	0	0	1	0	0	1	0	0	1	0
0	1	0	1	1	0	0	1	0	0	1	0	0
0	1	1	0	1	0	0	1	0	0	1	0	0
0	1	1	1	1	1	0	1	1	0	1	1	0
1	0	0	0	0	1	0	0	1	0	0	1	0
1	0	0	1	1	0	0	1	0	0	1	0	0
1	0	1	0	1	0	0	1	0	0	1	0	0
1	0	1	1	1	1	0	1	1	0	1	1	0
1	1	0	0	1	0	0	1	1	1	1	1	1
1	1	0	1	1	1	0	1	1	0	1	0	-1
1	1	1	0	1	1	0	1	1	0	1	0	-1
1	1	1	1	1	1	-1	1	1	-1	1	1	-1

Tabella 3.3: Tabella di verità compressori 4:2 Yang

3.1.4 Lin

L'idea principale è di utilizzare un $MUX\ 2-1$ al posto di un gate XOR per ridurre il ritardo di un contatore 4:2 approssimato e quindi ridurre il ritardo e la potenza del moltiplicatore [41]. L'architettura proposta è mostrata nella Figura 3.7, dove gli ingressi sono da x_1 a x_4 e le uscite sono Sum e $Carry$. L'errore si verifica quando tutti e quattro i sommatori sono '1' e l'uscita 111_2 si riduce a 10_2, ma questo errore è stato generato intenzionalmente per ottenere un algoritmo semplificato. La probabilità che un prodotto parziale diventi '1' è 1/4, quindi l'errore del contatore 4:2 approssimato si verifica con una probabilità molto bassa.
Nell' Equazione 3.10 e nell' Equazione 3.11 vengono riportate le sue espressioni.

$$\begin{aligned} xor1 &= x_1 \oplus x_2 \\ nand1 &= \overline{\mathrm{x_1 \cdot x_2}} \\ xor2 &= x_3 \oplus x_4 \\ nand2 &= \overline{\mathrm{x_3 \cdot x_4}} \end{aligned} \tag{3.10}$$

$$Carry = \overline{(\text{nand1} \cdot (\overline{\text{xor1} \cdot \text{xor2}}) \cdot \text{nand2})}$$
$$Sum = \begin{cases} xor1, & \text{per } xor2 = 0 \\ \overline{\text{xor1}}, & \text{per } xor2 = 1 \end{cases} \tag{3.11}$$

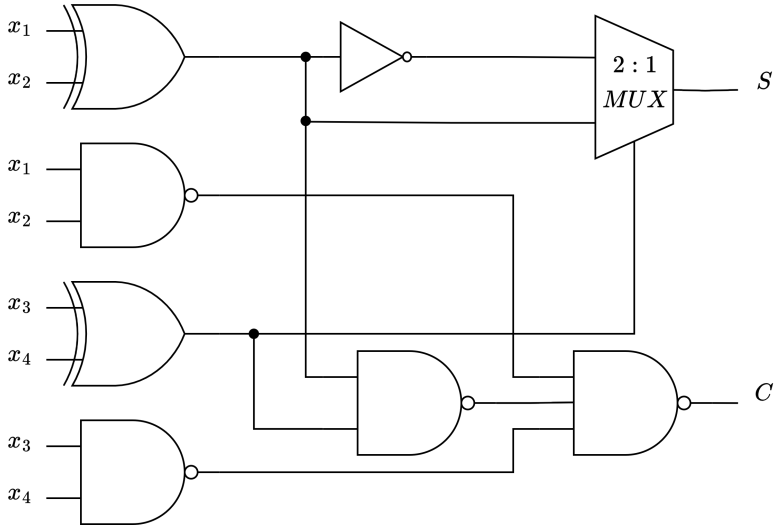


Figura 3.7: Compressore 4:2 approssimato Lin

La tabella di verità corrispondente è riportata nella Tabella 3.4.

x_4	x_3	x_2	x_1	C	S	$Differenza$
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	1	0	0
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	1	0	0
0	1	1	1	1	1	0
1	0	0	0	0	1	0
1	0	0	1	1	0	0
1	0	1	0	1	0	0
1	0	1	1	1	1	0
1	1	0	0	1	0	0
1	1	0	1	1	1	0
1	1	1	0	1	1	0
1	1	1	1	1	0	-2

Tabella 3.4: Tabella di verità compressore 4:2 Lin

3.1.5 Ha

Con questo compressore è stato ideato un progetto di moltiplicatore migliorato (Figura 3.8) [42] manipolando le voci della tabella di verità che danno luogo a uscite errate in un compressore 4:2. Nel compressore proposto, tutte le voci della tabella di verità sono state progettate per produrre $D = 0$ o $D = -1$. Ciò avviene producendo le uscite 01 quando gli ingressi sono 1100. Poiché ora c'è lo stesso numero di voci inesatte, ma l'errore è sempre nella stessa direzione (verso la riduzione delle uscite), è possibile utilizzare un semplice circuito di recupero degli errori per correggere l'errore quando $x_4x_3 = 11$.
L'Equazione 3.12 e l'Equazione 3.13 mostrano le equazioni logiche per i termini C e S nel compressore 4:2 proposto. È possibile progettare nuovi circuiti moltiplicatori approssimati utilizzando questo design del compressore in combinazione con un semplice modulo di recupero degli errori.

$$Carry = x_1 \cdot x_2 + x_1 \cdot x_3 + x_1 \cdot x_4 + x_2 \cdot x_3 + x_2 \cdot x_4 \tag{3.12}$$

$$Sum = (x_1 \oplus x_2) \oplus (x_3 + x_4) \tag{3.13}$$

La tabella di verità corrispondente è riportata nella Tabella 3.5.

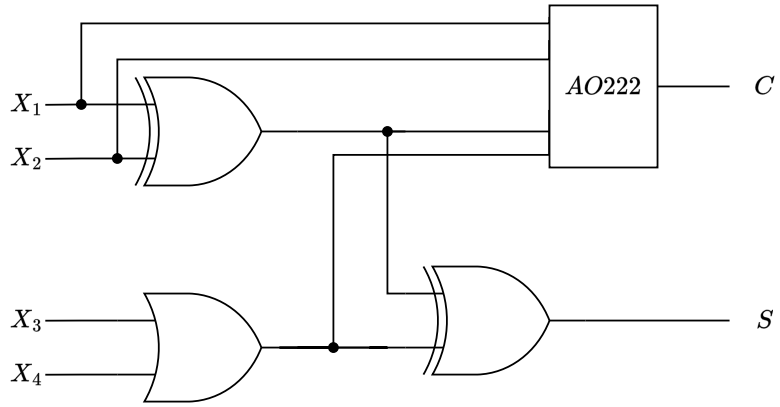


Figura 3.8: Compressore 4:2 approssimato Ha

x_4	x_3	x_2	x_1	C	S	$Differenza$
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	1	0	0
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	1	0	0
0	1	1	1	1	1	0
1	0	0	0	0	1	0
1	0	0	1	1	0	0
1	0	1	0	1	0	0
1	0	1	1	1	1	0
1	1	0	0	0	1	-1
1	1	0	1	1	0	-1
1	1	1	0	1	0	-1
1	1	1	1	1	1	-1

Tabella 3.5: Tabella di verità compressore 4:2 Ha

3.1.6 Akbar1 Akbar2

Nelle strutture proposte [43], si utilizza la tecnica del power gating per spegnere i componenti inutilizzati e ridurre il consumo di energia. L'obiettivo è minimizzare l'errore tra le uscite dei compressori esatti e quelle approssimate.
Nella prima struttura (Figura 3.9), l'accuratezza della modalità approssimata è migliorata aumentando la sua complessità, il che consente di aumentare la precisione

della somma delle uscite.
Le uscite sono riportate nell' Equazione 3.14 e nell' Equazione 3.15.

$$Carry = x_4 \tag{3.14}$$

$$Sum = \overline{(\overline{x_1 \oplus x_2} \cdot (\overline{x_3 \oplus x_4}))} \tag{3.15}$$

Nella seconda struttura (Figura 3.10), l'accuratezza del $Carry$ in uscita viene migliorata rispetto alla struttura precedente, ma a costo di un maggiore ritardo e consumo di energia.
Le uscite sono riportate nell' Equazione 3.16 e nell' Equazione 3.17.

$$Carry = \overline{(\overline{x_1 \cdot x_2} \cdot (\overline{x_3 \cdot x_4}))} \tag{3.16}$$

$$Sum = \overline{(\overline{x_1 \oplus x_2} \cdot (\overline{x_3 \oplus x_4}))} \tag{3.17}$$

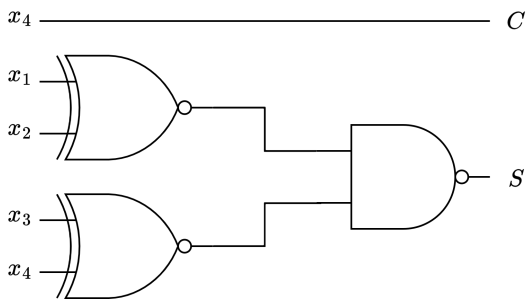


Figura 3.9: Compressore 4:2 approssimato Akbar1

La tabella di verità corrispondente è riportata nella Tabella 3.6.

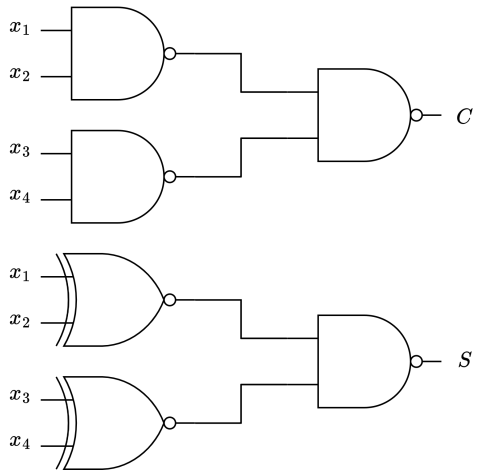


Figura 3.10: Compressore 4:2 approssimato Akbar1

				Akbar1			Akbar2		
x_4	x_3	x_2	x_1	C	S	D	C	S	D
0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	1	0
0	0	1	0	0	1	0	0	1	0
0	0	1	1	0	0	-2	1	0	0
0	1	0	0	0	1	0	0	1	0
0	1	0	1	0	1	-1	0	1	-1
0	1	1	0	0	1	-1	0	1	-1
0	1	1	1	0	1	-2	1	1	0
1	0	0	0	1	1	2	0	1	0
1	0	0	1	1	1	1	0	1	-1
1	0	1	0	1	1	1	0	1	-1
1	0	1	1	1	1	0	1	1	0
1	1	0	0	1	0	0	1	0	0
1	1	0	1	1	1	0	1	1	0
1	1	1	0	1	1	0	1	1	0
1	1	1	1	1	0	-2	1	0	-2

Tabella 3.6: Tabella di verità compressori 4:2 Akbar

3.1.7 Sabetz

Di solito, i compressori approssimati precedenti sono stati progettati utilizzando le logiche $AND-OR$ e XOR. Tuttavia, l'uso della logica XOR aumenta l'attività

di commutazione complessiva e, di conseguenza, la potenza dinamica. Al contrario, alcuni studi recenti hanno dimostrato che l'utilizzo della logica maggioritaria può portare a una maggiore efficienza di progettazione rispetto alle altre implementazioni comuni nelle nanotecnologie emergenti.
Il compressore approssimato 4:2 proposto [44] funziona secondo le equazioni riportate in Equazione 3.18, Equazione 3.19 e Equazione 3.20, ignorando i segnali C_{in} e C_{out} per motivi di efficienza progettuale. Inoltre, gli ingressi x_1, x_3 e x_4 sono inviati a una porta di maggioranza che produce l'uscita $Carry$. La somma è considerata costante e uguale a '1', eliminando la necessità di hardware aggiuntivo per calcolare il valore della somma. Questa semplificazione comporta una significativa riduzione del consumo energetico complessivo e del ritardo di propagazione del progetto.

$$C_{out} = C_{in} = 0 \tag{3.18}$$

$$Carry = Majority(x_1, x_3, x_4) = x_4(x_1 + x_3) + x_1x_3 \tag{3.19}$$

$$Sum = V_{DD} \tag{3.20}$$

La Tabella 3.7 riporta la tabella di verità del progetto proposto, in cui la preoccupazione principale è minimizzare la distanza di errore (ED) tra i risultati dei compressori esatti e inesatti. La distanza di errore è definita come la differenza aritmetica tra un'uscita errata e il suo valore corretto. Ad esempio, quando tutti gli ingressi sono '1', il valore decimale della somma degli ingressi è 4, ma il compressore approssimativo produce '1' per entrambe le uscite $Carry$ e Sum, il che comporta un valore decimale di 3 e una distanza di errore di -1. È importante notare che nell'approccio proposto non ci sono uscite false con un valore ED di 2 o superiore, poiché queste uscite errate potrebbero causare errori inaccettabili quando il compressore viene utilizzato nella struttura di un moltiplicatore approssimato. Inoltre, le distanze di errore di segno opposto, come -1 e +1, possono compensarsi reciprocamente nella struttura di un moltiplicatore. La Figura 3.11 illustra il compressore approssimato 4:2 proposto, che ha una struttura semplice con un solo gate di maggioranza progettato con 12 transistor basati sullo stile logico complementare. Questo progetto riduce il numero di transistor e la lunghezza del percorso critico, riducendo quindi il consumo di energia e il ritardo rispetto alle sue controparti precedenti.

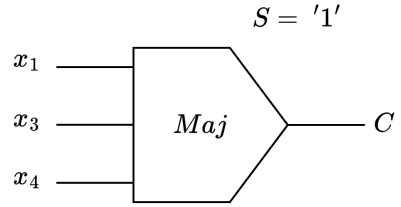


Figura 3.11: Compressore 4:2 approssimato Sabetz

x_4	x_3	x_2	x_1	C	S	$Differenza$
0	0	0	0	0	1	1
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	1	-1
0	1	0	0	0	1	0
0	1	0	1	1	1	1
0	1	1	0	0	1	-1
0	1	1	1	1	1	0
1	0	0	0	0	1	0
1	0	0	1	1	1	1
1	0	1	0	0	1	-1
1	0	1	1	1	1	0
1	1	0	0	1	1	1
1	1	0	1	1	1	0
1	1	1	0	1	1	0
1	1	1	1	1	1	-1

Tabella 3.7: Tabella di verità compressore 4:2 Sabetz

3.1.8 Ahma

Anche questo compressore [45] è molto efficiente dal punto di vista hardware, in quanto utilizza solo tre porte NOR e una $NAND$, come mostrato nella Figura 3.12. Le uscite sono riportate nell' Equazione 3.21 e nell' Equazione 3.22.

$$Carry = \overline{(\overline{x_1 + x_2} + (\overline{x_3 + x_4}))} \tag{3.21}$$

$$Sum = \overline{(\overline{x_1 + x_2} \cdot (\overline{x_3 + x_4}))} \tag{3.22}$$

La tabella di verità corrispondente è riportata nella Tabella 3.8.

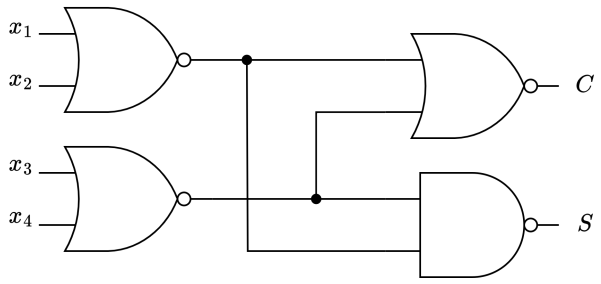


Figura 3.12: Compressore 4:2 approssimato Ahma

x_4	x_3	x_2	x_1	C	S	$Differenza$
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	1	-1
0	1	0	0	0	1	0
0	1	0	1	1	1	1
0	1	1	0	1	1	1
0	1	1	1	1	1	0
1	0	0	0	0	1	0
1	0	0	1	1	1	1
1	0	1	0	1	1	1
1	0	1	1	1	1	0
1	1	0	0	0	1	-1
1	1	0	1	1	1	0
1	1	1	0	1	1	0
1	1	1	1	1	1	-1

Tabella 3.8: Tabella di verità compressore 4:2 Ahma

3.1.9 Strollo

Il compressore approssimato 4:2 è un compressore proposto in [46] e utilizza il concetto dello stacking. Dati i quattro ingressi x_1, x_2, x_3, x_4, un circuito stacking a 4 bit ha quattro uscite y_1, y_2, y_3, y_4 tali che: y_1 sarà alta se uno qualsiasi degli ingressi è '1', y_2 sarà alta se due qualsiasi degli ingressi sono '1' e così via. Il circuito a quattro bit è descritto dalle seguenti equazioni booleane:

$$\begin{aligned} y_1 &= x_1 + x_2 + x_3 + x_4 \\ y_2 &= x_1x_2 + x_1x_3 + x_2x_3 + x_3x_4 \\ y_3 &= x_1x_2x_3 + x_1x_2x_4 + x_1x_3x_4 + x_2x_3x_4 \\ y_4 &= x_1x_2x_3x_4 \end{aligned} \tag{3.23}$$

Nel progettare un compressore, possiamo contare il numero di y_i che sono '1' invece di contare gli x_i, poiché è facile vedere che: $\sum_{i=1}^{4} x_i = \sum_{i=1}^{4} y_i$ (la sommatoria indica il conteggio dei termini booleani che sono '1').
Assumiamo che gli ingressi $x_1..x_4$ siano indipendenti tra loro e indichiamo come p la loro probabilità di essere '1'. L'analisi dell' Equazione 3.23 rivela che y_1 è il termine con la più alta probabilità di essere '1', seguito da y_2, y_3 e da y_4 che è il termine con la più bassa probabilità.
Poiché il valore massimo del conteggio di un compressore approssimato 4:2 è 3, dobbiamo trascurare uno dei termini y_i nell' Equazione 3.23; dalla discussione precedente la scelta che minimizza la probabilità di errore è quella di trascurare y_4.
Il nostro obiettivo è ora quello di ottenere tre nuove funzioni booleane w_1, w_2, w_3, tali che $\sum_{i=1}^{3} w_i = \sum_{i=1}^{3} y_i$.
Si procede per approssimazioni successive. In una prima fase, le w_i sono costruite in modo da essere il più semplici possibile, pur coprendo tutti i casi in cui y_1 è '1'. Una possibile soluzione è

$$\begin{aligned} w_1 &= x_1 + x_2 \\ w_2 &= x_3 \\ w_3 &= x_4 \end{aligned} \tag{3.24}$$

Contando il numero di w_i che sono '1' nell' Equazione 3.24 copriamo non solo tutti i casi in cui y_1 = '1', ma anche tutti i casi in cui y_2 = '1', con l'eccezione del caso in cui x_1x_2 = '1'. Inoltre, due dei quattro casi in cui y_3 è '1' sono coperti anche nell' Equazione 3.24. Pertanto, un compressore approssimato 4:2 può essere ottenuto contando il numero di w_i che sono '1' nell' Equazione 3.24, come mostrato nella Figura 3.13. La tabella di verità di questo circuito corrisponde al compressore Ha, [42].
Un compressore più accurato può essere ottenuto includendo nell' Equazione 3.24 il termine x_1x_2 necessario per coprire tutti i casi in cui y_2 è alto. Ciò può essere ottenuto modificando w_2 come segue:

$$w_2' = x_3 + x_1x_2 \tag{3.25}$$

Il compressore approssimato 4:2 ottenuto dall' Equazione 3.24 e sostituendo w_2 con w_2' dell' Equazione 3.25 copre tutti i casi in cui y_3 è alto, tranne il caso in cui

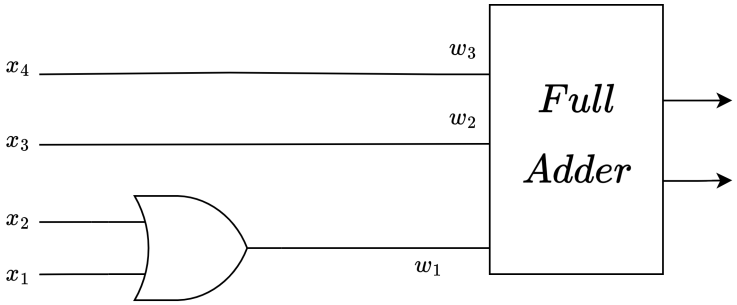


Figura 3.13: Compressore 4:2 approssimato Strollo ottenuto dell' Equazione 3.24

$x_1x_2x_3 =$ '1. Il circuito corrispondente, mostrato in Figura 3.14, sarà il compressore Strollo utilizzato in questo studio.
La tabella di verità corrispondente è riportata nella Tabella 3.9.

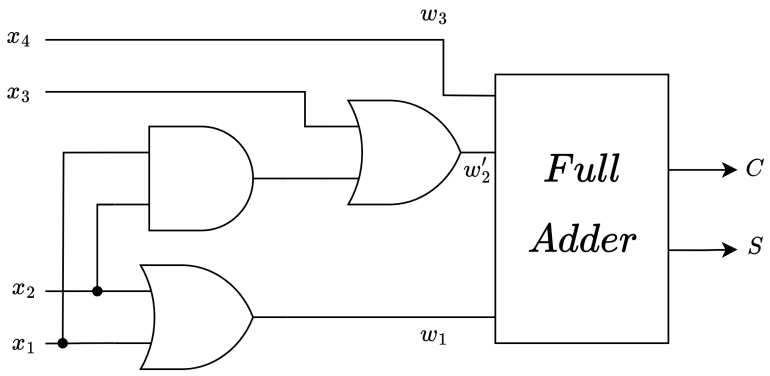


Figura 3.14: Compressore 4:2 approssimato Strollo utilizzato nello studio

Un circuito ancora più preciso può essere ottenuto includendo il termine $x_1x_2x_3$ in w_3:

$$w_3' = x_4 + x_1x_2x_3 \tag{3.26}$$

w_1 nell' Equazione 3.24, l' Equazione 3.25 e l' Equazione 3.26 soddisfano la condizione. Il circuito corrispondente è mostrato in Figura 3.15 e presenta la stessa tabella di verità del compressore Yang1 [40].

x_4	x_3	x_2	x_1	C	S	$Differenza$
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	1	0	0
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	1	0	0
0	1	1	1	1	0	-1
1	0	0	0	0	1	0
1	0	0	1	1	0	0
1	0	1	0	1	0	0
1	0	1	1	1	1	0
1	1	0	0	1	0	0
1	1	0	1	1	1	0
1	1	1	0	1	1	0
1	1	1	1	1	1	-1

Tabella 3.9: Tabella di verità compressore 4:2 approssimato Strollo

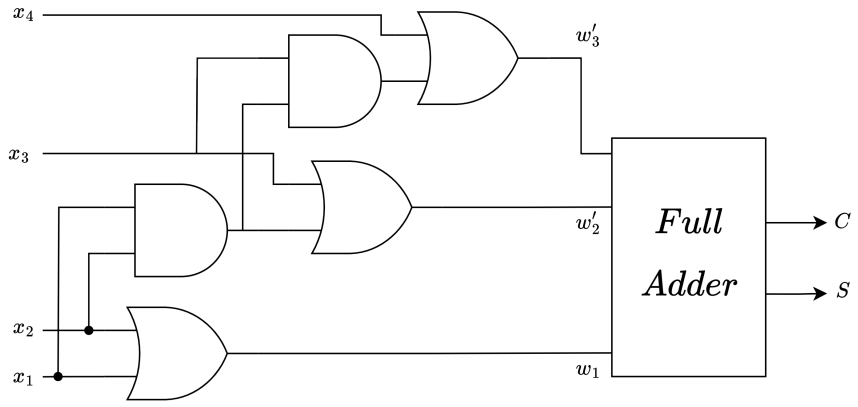


Figura 3.15: Compressore 4:2 approssimato Strollo ottenuto con la modifica dell' Equazione 3.26

3.2 Design del filtro con i compressori approssimati

Esistono due tipi di configurazione per il design del moltiplicatore:

- C-N: utilizza compressori approssimati 4:2 solo nelle n colonne meno significative della matrice dei prodotti parziali (PPM). Mira a minimizzare gli errori.
- C-FULL: utilizza compressori approssimati 4:2 in tutte le PPM. Si tratta di un approccio più aggressivo, volto a minimizzare la dissipazione di potenza.

Come riportato in [46], le topologie sulla frontiera pareto-ottimale dei moltiplicatori con segno sono le stesse trovate per i moltiplicatori senza segno. Vale la pena notare, tuttavia, che i due gruppi, corrispondenti alle configurazioni C-FULL e C-N, sono più nettamente separati, rispetto a quelli senza segno, soprattutto per i moltiplicatori 16 × 16. Ciò implica che qualsiasi miglioramento di potenza nella configurazione C-FULL (rispetto a quella C-N) non è possibile e comporta un drastico peggioramento dell'errore; pertanto, l'uso di moltiplicatori C-FULL nei moltiplicatori con segno è sconsigliato.

La Figura 3.16 mostra, a titolo di esempio, lo schema di riduzione per un moltiplicatore 8×8 senza segno, con configurazione C-N. In questo diagramma, le uscite $Carry$ e Sum dei compressori approssimati 4:2 sono collegate da linee tratteggiate, mentre le linee solide sono utilizzate per le uscite dei compressori esatti 4:2 e dei full adder. Le uscite degli half adder sono collegate da una linea continua con una barra verticale. Le linee tratteggiate indicano i collegamenti tra i pin T_{in} - T_{out} dei compressori 4:2 esatti. Si noti il FA nella colonna 12 del primo stadio e il FA nella colonna 14 del secondo stadio, devono accettare il segnale T_{out} del compressore 4:2 esatto della colonna precedente.

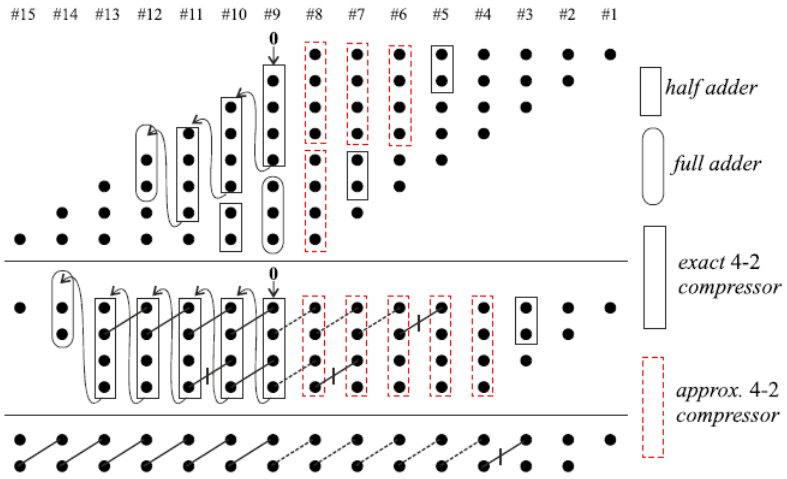


Figura 3.16: Schema di riduzione per un moltiplicatore 8 × 8 senza segno con configurazione C - N

Per minimizzare il tasso di errore, è importante considerare la probabilità dei segnali che guidano gli ingressi dei compressori approssimati durante la progettazione dell'albero di compressione dei prodotti parziali. La prestazione di errore dipende infatti dalla connessione specifica di ogni segnale ad ogni ingresso dei compressori. Nell'analisi del compressore, per un moltiplicatore senza segno, nella prima fase di riduzione dei prodotti parziali, quest'ultimi sono indipendenti tra loro e hanno una probabilità di essere '1' pari a 1/4. Non c'è quindi una connessione preferenziale tra questi prodotti parziali e gli ingressi dei compressori approssimati. Successivamente, le probabilità dei prodotti parziali della seconda fase vengono calcolate dalla tabella di verità del compressore [47]. Ad esempio, nel compressore Ha [42], la probabilità che S sia '1' è calcolata come segue:

$$\begin{aligned} p(S) &= p(x_4x_3x_2x_1 = 0011) + p(x_4x_3x_2x_1 = 0001) + ... \\ &\quad ... + p(x_4x_3x_2x_1 = 1100) + p(x_4x_3x_2x_1 = 1111) \\ &= 124/256 \end{aligned} \tag{3.27}$$

Il contributo di ogni termine nella somme è facilmente ottenuto dalla probabilità di x_i. Ad esempio, la probabilità che $x_4x_3x_2x_1$ sia uguale a 1100 è pari a $p(x_4)p(x_3)(1-p(x_2))(1-p(x_1)) = 9/256$. Le probabilità di uscita degli altri compressori vengono calcolati in modo simile.
Nel compressore approssimato proposto da Strollo [46], la condizione di errore è $x_1x_2x_3$ = '1', pertanto, per minimizzare la probabilità di errore, x_4 dovrebbe essere guidato dal prodotto parziale con la probabilità più alta. Ciò porta ad una configurazione diversa, come mostrato in Figura 3.17. Per gli altri compressori approssimati 4:2, è più difficile determinare la connessione ottimale tra i prodotti parziali e gli ingressi dei compressori approssimati che minimizza la probabilità di errore poiché gli errori sono dispersi nella tabella di verità (vedere Tabella 3.10). Quindi è stato usato lo stesso ordine degli ingressi del compressore Strollo anche per gli altri compressori per poter completare questo studio.

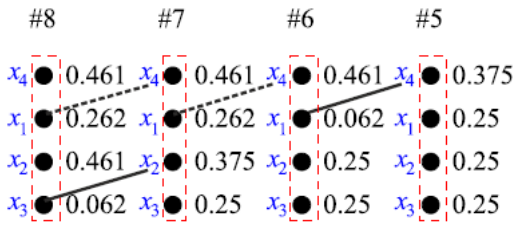


Figura 3.17: Probabilità dei prodotti parziali della seconda fase del moltiplicatore della Figura 3.16 utilizzando il compressore 4-2 approssimato Strollo [46]

x4...x1	Momeni		Venka		Yang1		Yang2		Yang3		Lin		Ha		Akbar1		Akbar2		Sabetz		Ahma		Strollo	
	CS	D	CS	D	CS	D	CS	D	CS	D	CS	D	CS	D	CS	D	CS	D	CS	D	CS	D	CS	D
0000	01	+1	00		00		00		00		00		00		00		00		01	+1	00		00	
0001	01		01		01		01		01		01		01		01		01		01		01		01	
0010	01		01		01		01		01		01		01		01		01		01		01		01	
0011	01	-1	10		10		10		10		10		10		00	-2	10		01	-1	01	-1	10	
0100	01		01		01		01		01		01		01		01		01		01		01		01	
0101	10		01	-1	10		10		10		10		10		01	-1	01	-1	11	+1	11	+1	10	
0110	10		01	-1	10		10		10		10		10		01	-1	01	-1	01	-1	11	+1	10	
0111	11		11		11		11		11		11		11		01	-2	11		11		11		10	-1
1000	01		01		01		01		01		01		01		11	+2	01		01		01		01	
1001	10		01	-1	10		10		10		10		10		11	+1	01	-1	11	+1	11	+1	10	
1010	10		01	-1	10		10		10		10		10		11	+1	01	-1	01	-1	11	+1	10	
1011	11		11		11		11		11		11		11		11		11		11		11		11	
1100	01	-1	10		10		11	+1	11	+1	10		01	-1	10		10		11	+1	01	-1	10	
1101	11		11		11		11		10	-1	11		10	-1	11		11		11		11		11	
1110	11		11		11		11		10	-1	11		10	-1	11		11		11		11		11	
1111	11	-1	11	-1	11	-1	11	-1	11	-1	10	-2	11	-1	10	-2	10	-2	11	-1	11	-1	11	-1

Tabella 3.10: Tabella di verità dei compressori 4:2 approssimati

D'ora in poi, nell'albero verranno inseriti i compressori approssimati nella configurazione C-N, posizionandoli nei primi 8 bit, ovvero nelle prime 8 colonne dell'albero. In seguito, sono stati condotti ulteriori studi per valutare se posizionare i compressori nei primi 7, 6 o 5 bit potesse offrire vantaggi al sistema.
Verranno misurate le performance di queste configurazioni in termini di area e potenza, mentre le metriche di errore saranno valutate attraverso il calcolo dell'errore medio, della deviazione standard, della varianza e dell'errore quadratico medio.

Iniziamo col vedere come queste diverse configurazioni fanno variare la forma e la grandezza dell'albero di compressione.
Nella Tabella 3.11 è riportato l'albero con i compressori 4:2 approssimati nei primi 8 bit, nel quale si hanno 211 di quest'ultimi, 111 compressori 4:2 esatti, 11 FA e 5 HA.
Per la configurazione a 7 bit invece si fa riferimento alla Tabella 3.12, dove il numero dei compressori 4:2 approssimati è di 174, quello dei compressori 4:2 esatti è di 151, mentre si hanno 11 FA e 5 HA.
Posizionando i compressori approssimati solo nei primi 6 bit invece si ha la configurazione riportata nella Tabella 3.13. In questo caso si hanno 139 compressori 4:2 approssimati, 187 compressori 4:2 esatti, 11 FA e 4 HA.
Nell'ultimo caso, nel quale i compressori approssimati sono posti solo nei primi 5 bit, si ha la la disposizione riportata nella Tabella 3.14. Qui i compressori 4:2 approssimati sono 100, i compressori 4:2 esatti sono 229, i FA sono 10 e gli HA sono 4.
Nella Tabella 3.15 viene ricapitolato il numero di compressori 4:2 approssimati ed esatti nelle 4 configurazioni.

bit	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STAGE 1	0	0	0	2	6	8	20	30	54	52	78	72	88	66	66	44	44	22
TAP 0-11	0	0	0	2	6	8	17	20	34	32	48	42	48	36	36	24	24	12
N 4:2	0	0	0	0	1	2	4	5	8	8	12	10	12	9	9	6	6	3
N FA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 2	0	0	0	4	6	8	14	26	38	40	52	54	61	48	45	32	29	13
N 4:2	0	0	0	1	1	2	3	6	9	10	13	13	15	12	11	8	7	3
N FA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 3	0	0	2	2	6	6	14	20	22	23	26	30	28	23	20	15	11	4
N 4:2	0	0	0	0	1	1	3	5	5	5	6	7	7	5	5	3	2	1
N FA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
N HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 4	0	0	2	4	4	8	12	12	12	14	15	16	12	13	9	7	4	1
N 4:2	0	0	0	1	1	2	3	3	3	3	3	4	3	3	2	1	1	0
N FA	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0
N HA	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
STAGE 5	0	0	4	2	4	6	6	6	8	8	8	7	6	6	5	3	1	1
N 4:2	0	0	1	0	1	1	1	1	2	2	2	1	1	1	1	0	0	0
N FA	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
N HA	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
STAGE 6	0	2	2	2	4	4	4	4	4	4	4	3	4	4	3	1	1	1
N 4:2	0	0	0	0	1	1	1	1	1	1	1	0	1	1	0	0	0	0
N FA	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0
N HA	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 7	1	2	2	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1

Tabella 3.11: Albero di compressione con compressori 4:2 approssimati sui primi 8 bit

bit	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STAGE 1	0	0	0	2	6	8	20	30	54	52	78	72	88	66	66	44	44	22
TAP 0-11	0	0	0	2	6	8	17	20	34	32	48	42	48	36	36	24	24	12
N 4:2	0	0	0	0	1	2	4	5	8	8	12	10	12	9	9	6	6	3
N FA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 2	0	0	0	4	6	8	14	26	38	44	52	54	61	48	45	32	29	13
N 4:2	0	0	0	1	1	2	3	6	9	11	13	13	15	12	11	8	7	3
N FA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 3	0	0	2	2	6	6	14	20	24	26	26	30	28	23	20	15	11	4
N 4:2	0	0	0	0	1	1	3	5	6	6	6	7	7	5	5	3	2	1
N FA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
N HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 4	0	0	2	4	4	8	12	12	12	14	15	16	12	13	9	7	4	1
N 4:2	0	0	0	1	1	2	3	3	3	3	3	4	3	3	2	1	1	0
N FA	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0
N HA	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
STAGE 5	0	0	4	2	4	6	6	6	8	8	8	7	6	6	5	3	1	1
N 4:2	0	0	1	0	1	1	1	1	2	2	2	1	1	1	1	0	0	0
N FA	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
N HA	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
STAGE 6	0	2	2	2	4	4	4	4	4	4	4	3	4	4	3	1	1	1
N 4:2	0	0	0	0	1	1	1	1	1	1	1	0	1	1	0	0	0	0
N FA	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0
N HA	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 7	1	2	2	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1

Tabella 3.12: Albero di compressione con compressori 4:2 approssimati sui primi 7 bit

bit	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STAGE 1	0	0	0	2	6	8	20	30	54	52	78	72	88	66	66	44	44	22
TAP 0-11	0	0	0	2	6	8	17	20	34	32	48	42	48	36	36	24	24	12
N 4:2	0	0	0	0	1	2	4	5	8	8	12	10	12	9	9	6	6	3
N FA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 2	0	0	0	4	6	8	14	26	38	44	52	54	61	48	45	32	29	13
N 4:2	0	0	0	1	1	2	3	6	9	11	13	13	15	12	11	8	7	3
N FA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 3	0	0	2	2	6	6	14	20	24	26	26	30	28	23	20	15	11	4
N 4:2	0	0	0	0	1	1	3	5	6	6	6	7	7	5	5	3	2	1
N FA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
N HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 4	0	0	2	4	4	8	12	12	12	14	16	16	12	13	9	7	4	1
N 4:2	0	0	0	1	1	2	3	3	3	3	4	4	3	3	2	1	1	0
N FA	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0
N HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 5	0	0	4	2	4	6	6	6	8	8	8	7	6	6	5	3	1	1
N 4:2	0	0	1	0	1	1	1	1	2	2	2	1	1	1	1	0	0	0
N FA	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
N HA	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
STAGE 6	0	2	2	2	4	4	4	4	4	4	4	3	4	4	3	1	1	1
N 4:2	0	0	0	0	1	1	1	1	1	1	1	0	1	1	0	0	0	0
N FA	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0
N HA	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 7	1	2	2	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1

Tabella 3.13: Albero di compressione con compressori 4:2 approssimati sui primi 6 bit

bit	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STAGE 1	0	0	0	2	6	8	20	30	54	52	78	72	88	66	66	44	44	22
TAP 0-11	0	0	0	2	6	8	17	20	34	32	48	42	48	36	36	24	24	12
N 4:2	0	0	0	0	1	2	4	5	8	8	12	10	12	9	9	6	6	3
N FA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 2	0	0	0	4	6	8	14	26	38	44	52	56	61	48	45	32	29	13
N 4:2	0	0	0	1	1	2	3	6	9	11	13	14	15	12	11	8	7	3
N FA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 3	0	0	2	2	6	6	14	20	24	26	28	30	28	23	20	15	11	4
N 4:2	0	0	0	0	1	1	3	5	6	6	7	7	7	5	5	3	2	1
N FA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
N HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 4	0	0	2	4	4	8	12	12	12	16	16	16	12	13	9	7	4	1
N 4:2	0	0	0	1	1	2	3	3	3	4	4	4	3	3	2	1	1	0
N FA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
N HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 5	0	0	4	2	4	6	6	6	8	8	8	7	6	6	5	3	1	1
N 4:2	0	0	1	0	1	1	1	1	2	2	2	1	1	1	1	0	0	0
N FA	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
N HA	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
STAGE 6	0	2	2	2	4	4	4	4	4	4	4	3	4	4	3	1	1	1
N 4:2	0	0	0	0	1	1	1	1	1	1	1	0	1	1	0	0	0	0
N FA	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0
N HA	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 7	1	2	2	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1

Tabella 3.14: Albero di compressione con compressori 4:2 approssimati sui primi 5 bit

	Numero compressori 4:2 approssimati	Numero compressori 4:2 esatti	Numero Full Adder	Numero Half Adder
8 bit	211	111	11	5
7 bit	174	151	11	5
6 bit	139	187	11	4
5 bit	100	229	10	4

Tabella 3.15: Numero compressori 4:2 nelle configurazioni a 8,7,6 e 5 bit

Capitolo 4

Risultati e confronti

In questo capitolo saranno presentati i risultati e i confronti delle diverse configurazioni studiate, con particolare attenzione all'area occupata, alla potenza e agli errori riscontrati per l'utilizzo dei compressori 4:2 approssimati per comprimere i prodotti parziali generati dai moltiplicatori del filtro in confronto al filtro che utilizza i soli compressori 4:2 esatti.
La disposizione dei compressori all'interno dell'albero per le diverse configurazioni è stata mostrato nella sezione 3.2.

4.1 Errori dell'aritmetica approssimata

In questa sezione possiamo iniziare a vedere come variano le metriche dell'errore in base al numero di compressori approssimati utilizzati. Si può notare come al diminuire di quest'ultimi anche i valori dell'errore tendono a diminuire proprio perchè il risultato viene approssimato di meno ogni volta che si sostituisce un intera colonna di compressori approssimati con una di compressori esatti.
Nella Tabella 4.1, Tabella 4.2, Tabella 4.3 e Tabella 4.4 vengono riportati quindi i valori di errore medio, deviazione standard, varianza ed errore quadratico medio delle rispettive configurazioni.

I valori presenti nelle tabelle sono stati ottenuti calcolando la differenza tra la somma di tutte le uscite dei moltiplicatori, che rappresenta l'ingresso dell'albero di compressione, e l'uscita del filtro, che è su 8 bit in formato $Q7.0$, partendo da un numero che e' $Q7.8$.
Per ottenere l'uscita del filtro, si parte dall'ultimo stage di compressione, dove sono presenti i bit delle ultime due righe, che vengono poi sommati con una costante calcolata dai bit non variabili ricavati durante la generazione dei prodotti parziali. Infine, per ottenere l'uscita del filtro su 8 bit, si prendono i bit della somma che

vanno dal 11° al 4° bit.
Per ottenere, invece, la somma in ingresso all'albero con lo stesso numero di bit dell'uscita del filtro, è necessario arrotondare e spostare di 8 posizioni la somma in ingresso ($Q7.8$), proprio per avere il formato in $Q7.0$.
Dalla differenza tra questi due numeri è poi possibile calcolare le diverse metriche dell'errore.

Compressore Approssimato	Errore Medio	Deviazione Standard	Varianza	Errore Quadratico Medio
momeni	-0.55	1.31	1.71	2.01
venka	5.52	1.10	1.20	31.65
yang1	0.84	0.65	0.42	1.13
yang2	-0.19	0.93	0.86	0.90
yang3	1.05	1.09	1.18	2.28
lin	1.51	1.02	1.04	3.31
ha	3.64	0.93	0.86	14.09
akbar1	2.81	1.82	3.30	11.21
akbar2	6.46	1.33	1.76	43.43
sabetz	-0.48	1.26	1.60	1.83
ahma	1.27	1.36	1.84	3.46
strollo	1.79	0.79	0.62	3.81

Tabella 4.1: Metriche degli errori del filtro con l'utilizzo dei compressori 4:2 approssimati nei primi 8 bit dell'albero

Compressore Approssimato	Errore Medio	Deviazione Standard	Varianza	Errore Quadratico Medio
momeni	-1.01	0.80	0.63	1.66
venka	2.55	0.68	0.46	6.95
yang1	0.40	0.50	0.25	0.41
yang2	-0.10	0.59	0.35	0.36
yang3	0.49	0.64	0.41	0.66
lin	0.71	0.60	0.36	0.86
ha	1.75	0.61	0.37	3.42
akbar1	0.94	1.19	1.42	2.31
akbar2	2.99	0.76	0.58	9.49
sabetz	-0.79	0.81	0.66	1.28
ahma	0.66	0.78	0.61	1.05
strollo	0.84	0.53	0.28	0.98

Tabella 4.2: Metriche degli errori del filtro con l'utilizzo dei compressori 4:2 approssimati nei primi 7 bit dell'albero

Compressore Approssimato	Errore Medio	Deviazione Standard	Varianza	Errore Quadratico Medio
momeni	-0.47	0.54	0.29	0.51
venka	1.35	0.51	0.26	2.07
yang1	0.20	0.40	0.16	0.20
yang2	0.03	0.39	0.15	0.15
yang3	0.34	0.49	0.24	0.35
lin	0.36	0.48	0.23	0.36
ha	0.80	0.45	0.21	0.85
akbar1	-0.26	0.74	0.54	0.61
akbar2	1.57	0.53	0.28	2.74
sabetz	-0.26	0.54	0.29	0.36
ahma	0.30	0.53	0.28	0.36
strollo	0.43	0.49	0.24	0.43

Tabella 4.3: Metriche degli errori del filtro con l'utilizzo dei compressori 4:2 approssimati nei primi 6 bit dell'albero

Compressore Approssimato	Errore Medio	Deviazione Standard	Varianza	Errore Quadratico Medio
venka	0.56	0.50	0.25	0.56
yang1	0.10	0.30	0.09	0.10
yang2	0.01	0.28	0.08	0.08
yang3	0.13	0.35	0.13	0.14
lin	0.18	0.38	0.15	0.18
ha	0.40	0.49	0.24	0.40
akbar1	0.45	0.51	0.26	0.45
akbar2	0.69	0.47	0.22	0.70
sabetz	0.08	0.38	0.14	0.15
ahma	0.29	0.46	0.21	0.30
strollo	0.26	0.44	0.19	0.26

Tabella 4.4: Metriche degli errori del filtro con l'utilizzo dei compressori 4:2 approssimati nei primi 5 bit dell'albero

4.2 Potenza, area e timing dell'aritmetica approssimata

In questa sezione verranno riportati i valori di area e di potenza per le differenti architetture inserite all'interno della Tabella 4.5, Tabella 4.6, Tabella 4.7 e Tabella 4.8.

Poiché il compressore Sabetz [44] è stato progettato per fornire le migliori prestazioni in termini di area e di potenza, ma ha un tasso di errore molto alto (8/16), si è deciso di concentrarsi solo sui compressori che hanno un valore di varianza uguale o inferiore a quello del Sabetz escludendo tutti gli altri.
Nelle tabelle sono presenti anche due colonne aggiuntive che mettono a confronto l'area e la potenza con l'aritmetica esatta, ovvero con il filtro che nell'albero utilizza solo compressori 4:2 esatti (sottosezione 2.5.3), ricordando che i suoi valori di area e di potenza sono rispettivamente 14306 μm^2 (1E+04 μm^2) e 0.0731191 W (7.31E-02 W).

Compressore Approssimato	Potenza $[W]$	Area $[\mu m^2]$	Confronto Potenza	Confronto Area
strollo	6.70E-02	1.27E+04	-8.33%	-11.25%
sabetz	4.81E-02	9.26E+03	-34.20%	-35.29%
yang1	6.87E-02	1.43E+04	-6.02%	-0.27%
yang2	6.39E-02	1.36E+04	-12.55%	-5.16%
yang3	6.69E-02	1.27E+04	-8.55%	-11.36%
venka	6.11E-02	1.26E+04	-16.37%	-11.99%
lin	6.95E-02	1.29E+04	-4.88%	-10.02%
ha	6.66E-02	1.23E+04	-8.94%	-14.28%

Tabella 4.5: Valori di potenza e area del filtro con i compressori 4:2 esatti nei primi 8 bit

Compressore Approssimato	Potenza $[W]$	Area $[\mu m^2]$	Confronto Potenza	Confronto Area
strollo	6.80E-02	1.30E+04	-7.04%	-9.09%
sabetz	5.31E-02	1.03E+04	-27.38%	-27.95%
yang1	6.95E-02	1.42E+04	-5.00%	-0.79%
yang2	6.61E-02	1.36E+04	-9.57%	-4.68%
yang3	6.80E-02	1.29E+04	-7.05%	-9.64%
venka	6.37E-02	1.29E+04	-12.91%	-9.82%
lin	7.03E-02	1.32E+04	-3.79%	-8.04%
ha	6.75E-02	1.26E+04	-7.67%	-12.20%

Tabella 4.6: Valori di potenza e area del filtro con i compressori 4:2 esatti nei primi 7 bit

Compressore Approssimato	Potenza $[W]$	Area $[\mu m^2]$	Confronto Potenza	Confronto Area
strollo	6.82E-02	1.33E+04	-6.74%	-7.29%
sabetz	5.72E-02	1.13E+04	-21.74%	-21.31%
yang1	7.01E-02	1.42E+04	-4.19%	-0.79%
yang2	6.75E-02	1.38E+04	-7.75%	-3.61%
yang3	6.88E-02	1.32E+04	-5.87%	-7.44%
venka	6.55E-02	1.32E+04	-10.37%	-8.02%
lin	7.09E-02	1.32E+04	-3.00%	-8.04%
ha	6.82E-02	1.26E+04	-6.75%	-12.20%

Tabella 4.7: Valori di potenza e area del filtro con i compressori 4:2 esatti nei primi 6 bit

Compressore Approssimato	Potenza $[W]$	Area $[\mu m^2]$	Confronto Potenza	Confronto Area
strollo	6.95E-02	1.35E+04	-5.00%	-5.56%
sabetz	6.16E-02	1.21E+04	-15.73%	-15.10%
yang1	7.08E-02	1.41E+04	-3.23%	-1.21%
yang2	6.92E-02	1.38E+04	-5.36%	-3.34%
yang3	6.98E-02	1.35E+04	-4.50%	-5.51%
venka	6.79E-02	1.35E+04	-7.12%	-5.61%
lin	7.12E-02	1.37E+04	-2.65%	-4.53%
ha	6.96E-02	1.32E+04	-4.81%	-7.66%

Tabella 4.8: Valori di potenza e area del filtro con i compressori 4:2 esatti nei primi 5 bit

Nella Figura 4.1 viene riportato come varia la potenza con in base alle quattro configurazioni studiate.

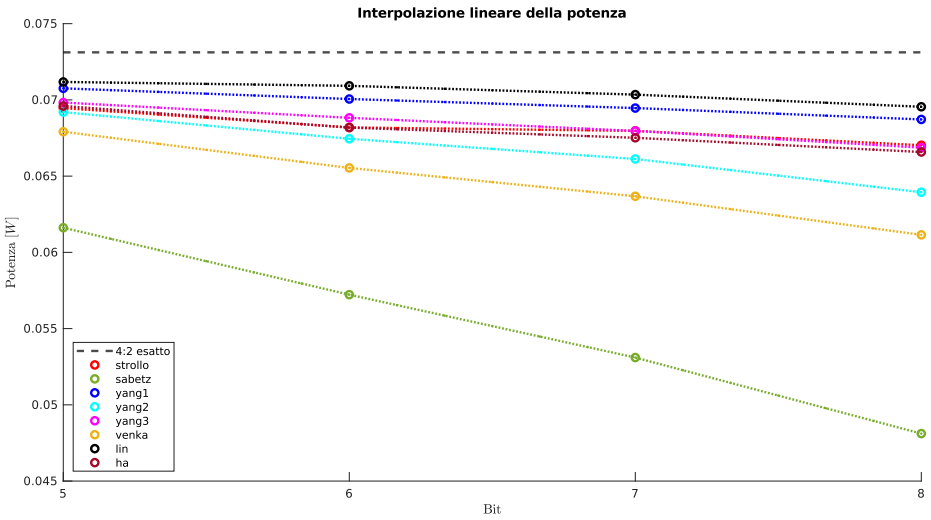


Figura 4.1: Potenza del filtro con compressori 4:2 approssimati rispettivamente a 8,7,6,5 bit

Stessa cosa viene fatta per l'area, come nella Figura 4.2.

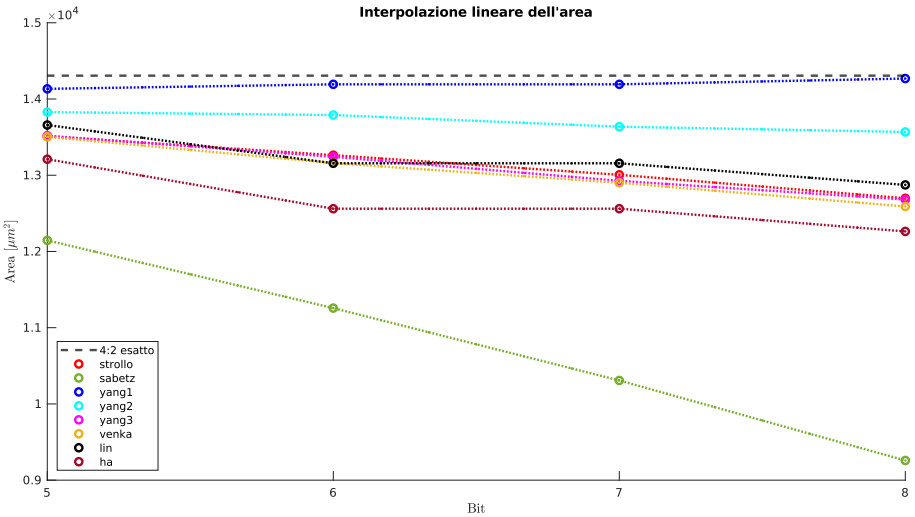


Figura 4.2: Area del filtro con compressori 4:2 approssimati rispettivamente a 8,7,6,5 bit

Essendoci 5 stage nell'albero di compressione, tra i 4 valori di aria e di potenza, per le rispettive architetture, sono stati interpolati altri 5 valori per simulare l'aggiunta o la rimozione di un intera colonna di compressori approssimati nel singolo stage. E' possibile vedere le potenze (Figura 4.3) e l'aree (Figura 4.4) che si dovrebbero avere anche nelle situazioni intermedie.

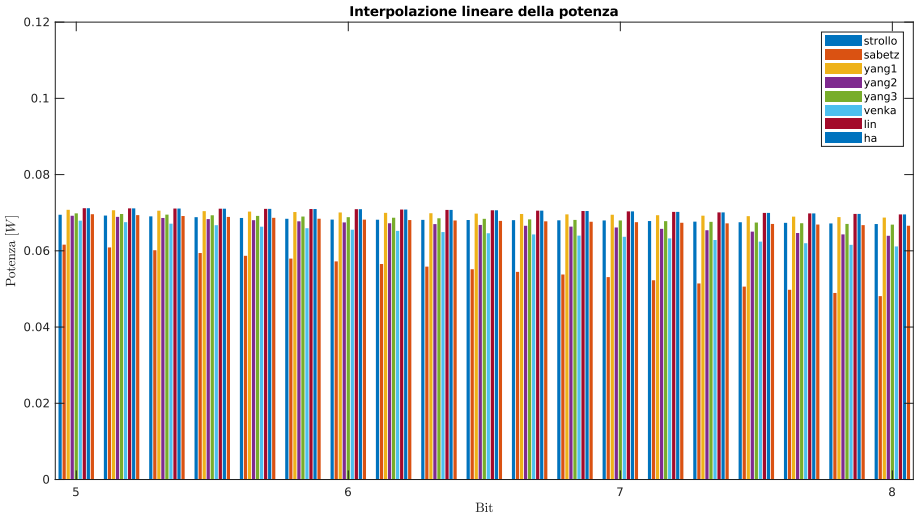


Figura 4.3: Potenza di tutti i compressori 4:2 approssimati studiati tra a 8 e 5 bit

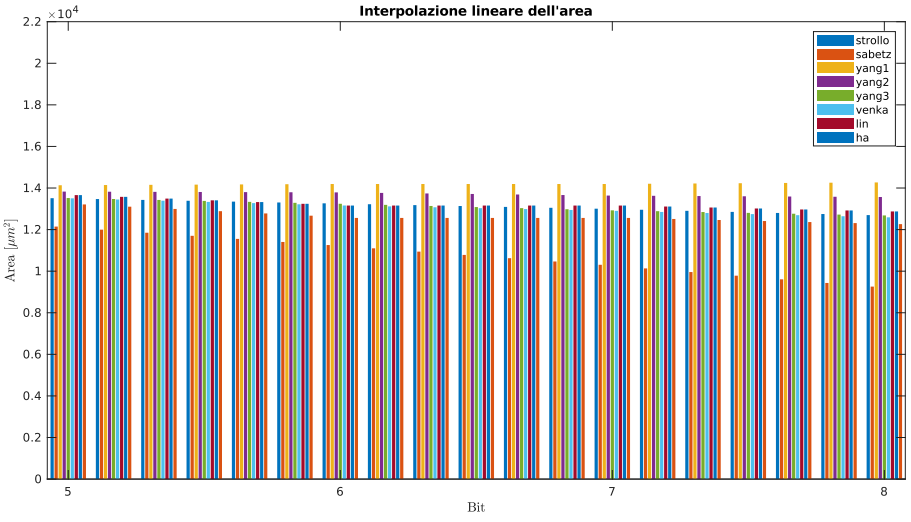


Figura 4.4: Area di tutti i compressori 4:2 approssimati studiati tra a 8 e 5 bit

Per quanto riguarda il timing, non si è avuto nessun miglioramento rispetto al filtro di partenza, in quanto, i critical path non passano mai attraverso i compressori approssimati ma passano attraverso i compressori esatti che portano sempre ad avere slack 0 con lo stesso valore di clock ed incertezza inseriti nella sintesi.

4.3 Requisiti del sistema

Il Bit Error Ratio è una misura della qualità della trasmissione di dati in un sistema di comunicazione digitale. Indica la frazione di bit trasmessi che sono stati ricevuti in modo errato. Il BER è un parametro importante per valutare l'affidabilità e la qualità di un sistema di comunicazione digitale.

Per l'OIF-CEI-05.2 [48] la specifica per il SerDes 112 Gps LR prevede un BER $< 10^{-4}$, mentre 802.3-KR [49] prevede BER $< 2.4 * 10^{-}4$.
Si consideri che i valori limite di BER stabiliti dalla specifica si riferiscono ad un errore random perfettamente gaussiano, che corrompe alcuni simboli che verranno poi corretti dal FEC. Gli errori, causati prevalentemente dal rumore bianco, saranno per lo più casuali. Per soddisfare sia la specifica sia le esigenze aggiuntive del cliente, che richiede un margine di sicurezza, è stato determinato che il margine sul parametro 112 deve assicurare un BER non superiore a 10^{-6} in ogni condizione operativa.

Per conoscere il valore del rumore che porta il sistema ad avere un BER di 10^{-6} si è usato un modello che introduce un rumore sugli ingressi. Questi hanno lo stesso pattern di ingressi del filtro, valori in PAM4 $[-3, -1,1,3]$ che vengono poi convoluti con i coefficienti $h_0 = 15$ e $h_1 = 4.5$ del DFE. In uscita dal filtro i livelli saranno concentrati attorno ai valori aspettati $[-45, -15,15,45]$ e compresi tra -64 e 63. Una volta passati attraverso il DFE vengono confrontati i valori della sua uscita con i valori senza rumore in ingresso al filtro. Da questo confronto si visualizza la curva del BER e si trova l'SNR che lo porta ad avere un valore pari a 10^{-6}, come si può notare nella Figura 4.5.

Nella figura sono riportati i due valori dell'SNR per BER $= 1 * 10^{-4}$ e BER $= 1 * 10^{-6}$, rispettivamente $-12.0867dB$ e $-9.9399dB$.

Per calcolare la varianza del rumore e confrontarla con le metriche di errore delle aritmetiche approssimate e' stata ritenuta trascurabile power penalty di $0.2dB$ nel punto in cui il BER è pari a 10^{-6}.
Aggiungendo $0.2dB$ all'SNR, il sistema avrebbe un BER di $1.6 * 10^{-6}$, il quale, nel nostro caso, rappresenta ancora un errore trascurabile in uscita.
Come riportato anche nel grafico gli $0.2dB$ dell'SNR per il quale il BER vale $1 * 10^{-6}$ è 0.4661.
Per confrontare le configurazioni che utilizzano l'aritmetica approssimata, si può utilizzare l'errore quadratico medio come parametro di errore, poiché l'utilizzo della varianza richiederebbe l'inserimento di un circuito aggiuntivo per centrare l'errore in zero, il quale potrebbe compromettere le performance del design.

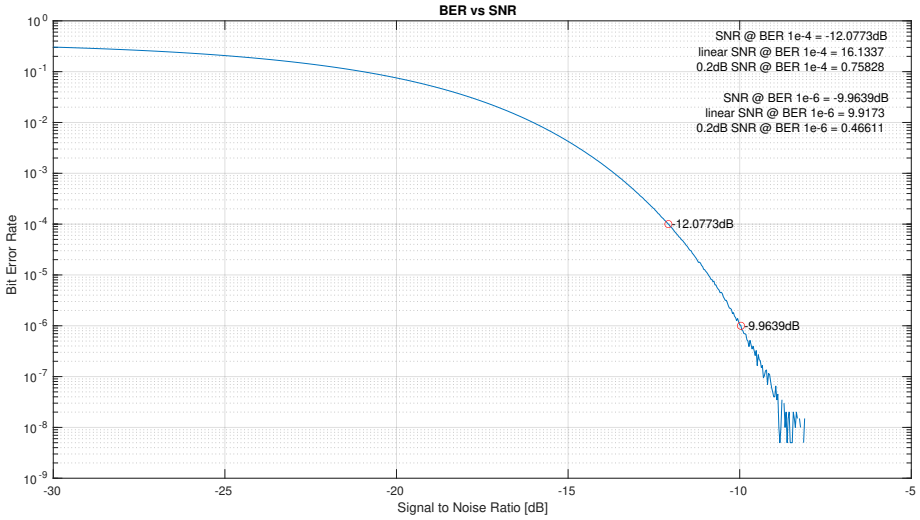


Figura 4.5: Bit Error Rate vs Signal to Noise Ratio

4.4 Riduzione di area

Per visualizzare la diminuzione di area ottenibile con le diverse configurazioni esaminate, si è scelto di rappresentare graficamente tale riduzione per diversi valori di errore quadratico medio. Attraverso l'uso di uno script su Matlab, sono stati interpolati i valori dell'errore quadratico medio e dell'aree ottenute, per calcolare la percentuale di riduzione rispetto alla soluzione con i compressori 4:2 esatti, in corrispondenza dei punti in cui l'errore quadratico medio assume i valori [0.2 0.3 0.4 0.5 0.6]. Questi risultati sono mostrati nella Figura 4.6.

4.5 Riduzione di potenza

La stessa procedura è stata applicata anche per la riduzione di potenza e i risultati sono riportati nellaFigura 4.7.

Come possiamo notare il compressore che all'interno dell'albero porta a dei risultati migliori rispetto a tutti gli altri è il Sabetz, con una differrenza superiore al 10% nei confronti anche del secondo miglior compressore approssimato. Nella Tabella 4.9 sono riportati tutti i valori della figura precedente.

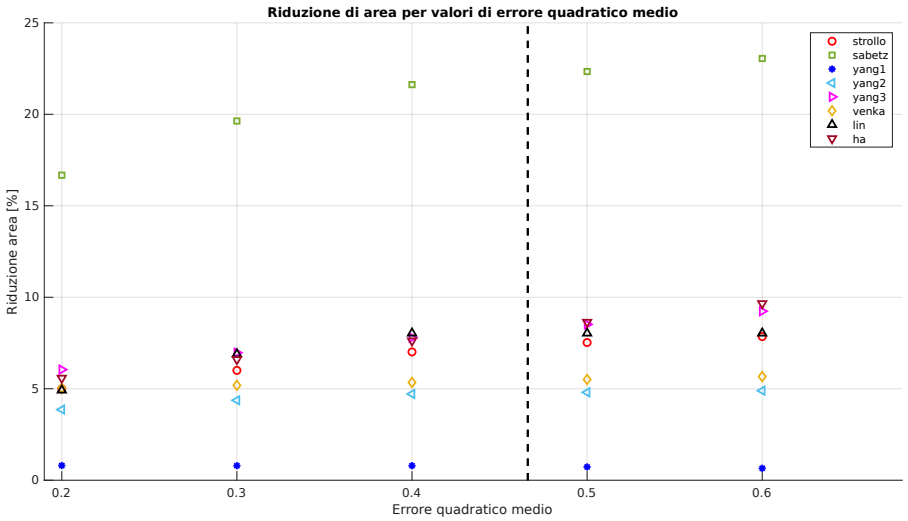


Figura 4.6: Riduzione di area dei compressori 4:2 approssimati nei corrispettivi valori di errore quadratico medio

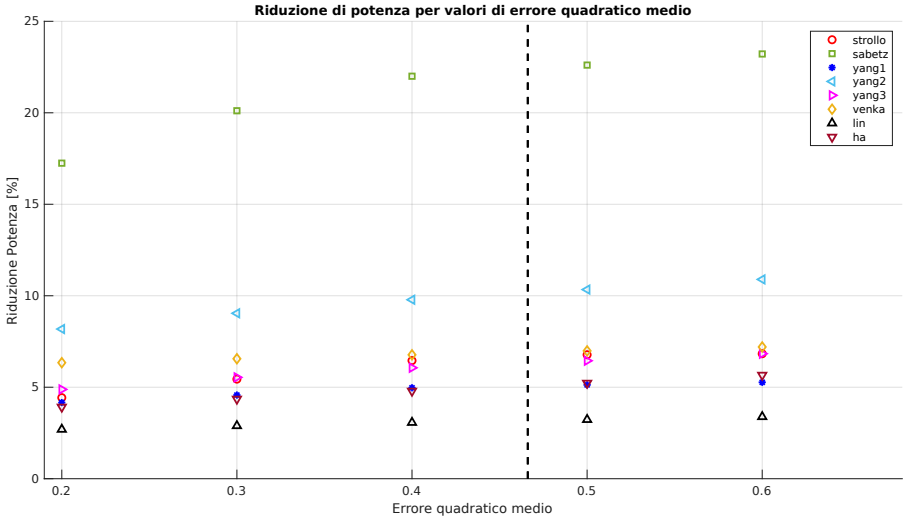


Figura 4.7: Riduzione di potenza dei compressori 4:2 approssimati nei corrispettivi valori di errore quadratico medio

4.6 Soluzione migliore

Per confermare i risultati ottenuti tramite l'utilizzo di interpolazioni lineari dei valori ricavati dalle configurazioni nei quali i compressori approssimati sono stati

	0.2	0.3	0.4	0.5	0.6
strollo	4.43%	5.44%	6.46%	6.78%	6.83%
sabetz	17.25%	20.11%	22.00%	22.61%	23.22%
yang1	4.16%	4.57%	4.97%	5.13%	5.27%
yang2	8.18%	9.04%	9.78%	10.34%	10.89%
yang3	4.88%	5.54%	6.06%	6.45%	6.83%
venka	6.34%	6.56%	6.77%	6.99%	7.20%
lin	2.69%	2.89%	3.07%	3.23%	3.38%
ha	3.92%	4.36%	4.79%	5.23%	5.66%

Tabella 4.9: Riduzione di potenza per i rispettivi errori quadratici medi per ogni configurazione

inseriti nei primi 8, 7, 6 e 5 bit, è stato modificato l'albero del filtro che utilizza i compressori Sabetz, in modo da ottenere un errore quadratico medio simile a 0.4661. Ciò ha portato alla progettazione di un albero con 155 compressori 4:2 approssimati, 170 compressori 4:2 esatti, 11 full adder e 5 half adder, come riportato nella Tabella 4.11. Nella colonna del sesto bit, il primo stage utilizza compressori esatti, nel secondo stage ci sono 9 compressori esatti e 4 compressori approssimati, mentre dal terzo stage fino all'ultimo sono tutti compressori approssimati. La configurazione di partenza prevedeva l'utilizzo di compressori approssimati fino al settimo bit (colonna 6), al quale sono stati sostituiti i primi compressori approssimati con quelli esatti, partendo dall'alto, al fine di diminuire l'errore quadratico medio del filtro, come si può notare dalla Tabella 4.10.

	Numero Compressori Approssimati	Errore Medio	Deviazione Standard	Varianza	Errore Quadratico Medio	Potenza [W]	Riduzione Potenza
7 bit	174	-0.79	0.81	0.66	1.28	5.31E-02	-27.38%
intermedio	155	-0.14	0.66	0.44	0.46	5.42E-02	-25.88%
6 bit	139	-0.26	0.54	0.29	0.36	5.72E-02	-21.74%

Tabella 4.10: Valori metriche di errore e potenza delle configurazioni con il Sabetz

In questo caso, l'errore quadratico medio assume il valore di 0.4624, mentre la potenza è pari a 0.0541923 W, il che comporta una riduzione del 25.88% rispetto all'albero con soli compressori esatti. Il valore di 0.4661 può essere approssimato a 0.5, e dalla Tabella 4.9 si evince che il moltiplicatore che utilizza i Sabetz con questo errore quadratico medio presenta una riduzione del 22.61%, che risulta essere molto vicina al valore calcolato. Tuttavia, poiché i valori che conducono a tale riduzione di potenza nel grafico sono stati interpolati, potrebbero esserci degli errori o

bit	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STAGE 1	0	0	0	2	6	8	20	30	54	52	78	72	88	66	66	44	44	22
TAP 0-11	0	0	0	2	6	8	17	20	34	32	48	42	48	36	36	24	24	12
N 4:2	0	0	0	0	1	2	4	5	8	8	12	10	12	9	9	6	6	3
N FA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 2	0	0	0	4	6	8	14	26	38	44	52	54	61	48	45	32	29	13
N 4:2	0	0	0	1	1	2	3	6	9	11	13	13	15	12	11	8	7	3
N FA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 3	0	0	2	2	6	6	14	20	24	26	26	30	28	23	20	15	11	4
N 4:2	0	0	0	0	1	1	3	5	6	6	6	7	7	5	5	3	2	1
N FA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
N HA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 4	0	0	2	4	4	8	12	12	12	14	15	16	12	13	9	7	4	1
N 4:2	0	0	0	1	1	2	3	3	3	3	3	4	3	3	2	1	1	0
N FA	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0
N HA	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
STAGE 5	0	0	4	2	4	6	6	6	8	8	8	7	6	6	5	3	1	1
N 4:2	0	0	1	0	1	1	1	1	2	2	2	1	1	1	1	0	0	0
N FA	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
N HA	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
STAGE 6	0	2	2	2	4	4	4	4	4	4	4	3	4	4	3	1	1	1
N 4:2	0	0	0	0	1	1	1	1	1	1	1	0	1	1	0	0	0	0
N FA	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0
N HA	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STAGE 7	1	2	2	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1

Tabella 4.11: Albero di compressione con compressori 4:2 approssimati Sabetz per avere l'errore quadratico medio minore di 0.49313

approssimazioni che potrebbero far sì che questi risultino solo approssimativamente corrispondenti tra di loro, sebbene molto simili.

Capitolo 5

Conclusioni

In questo lavoro è stata svolta un'analisi dettagliata del comportamento di un filtro FIR, con particolare attenzione all'utilizzo di compressori 4:2 approssimati, al fine di valutare se l'inserimento di tali compressori all'interno dell'albero delle somme possa portare a un miglioramento delle prestazioni del filtro, in termini di riduzione della potenza e dell'area. Per raggiungere questo obiettivo, sono stati condotti diversi test e confronti tra diverse configurazioni del filtro, al fine di individuare la soluzione ottimale per l'applicazione specifica presa in esame.
Nel corso della ricerca sono stati esaminati diversi moltiplicatori con diversi livelli di approssimazione, con lo scopo di valutare quale soluzione possa garantire il miglior compromesso tra potenza e accuratezza. Tra le diverse opzioni prese in esame, il compressore Sabetz si è dimostrato il più efficace. Tuttavia, per valutare l'errore introdotto dai compressori approssimati e' stata ritenuta trascurabile power penalty di $0.2dB$ nel punto in cui il BER è pari a 10^{-6} che causa una degradazione del BER a $1.6 \cdot 10^{-6}$. Tale valore rappresenta un limite oltre il quale l'errore diventa rilevante, pertanto tutte le configurazioni che presentano un errore quadratico medio inferiore a tale valore possono essere utilizzate in base alle specifiche richieste, al fine di garantire un compromesso accettabile tra potenza e accuratezza e di minimizzare l'errore introdotto dai compressori approssimati.

La soluzione ottimale, ritenuta trascurabile una power penalty di $0.2dB$, è il filtro approssimato che utilizza 155 compressori 4:2 approssimati Sabetz. Questo design mi porta ad avere una riduzione di potenza del 25.88% rispetto al filtro che utilizza i compressori esatti all'interno dell'albero.

Per quanto riguarda le possibili evoluzioni future della ricerca, si potrebbe valutare l'utilizzo di altri compressori approssimati, al fine di ampliare la gamma di soluzioni prese in esame. In particolare, sarebbe interessante esaminare il comportamento del compressore 3:2 approssimato, per confrontarlo con il filtro che utilizza

l'algortimo MBE e l'albero di compressione con i contatori 3:2 esatti (ovvero i full adder). Tale confronto potrebbe fornire ulteriori informazioni sulle prestazioni del filtro e sulla capacità di garantire un compromesso accettabile tra potenza e accuratezza.

Bibliografia

[1] *Cisco Global Cloud Index: Forecast and Methodology, 2016–2021.* Cisco, San Jose, CA, USA, 2018 (cit. a p. 1).

[2] Cadence PCB Solutions. *SerDes Design: High Speed Electronic Challenges.* URL: https://resources.pcb.cadence.com/blog/2020-serdes-design-high-speed-electronic-challenges (cit. a p. 3).

[3] *NRZ vs. PAM4 Modulation Techniquess.* URL: https://community.fs.com/article/nrz-vs-pam4-modulation-techniques.html (cit. a p. 4).

[4] V. Stojanovic e M. Horowitz. «Modeling and analysis of high-speed links». In: *Proceedings of the IEEE 2003 Custom Integrated Circuits Conference, 2003.* 2003, pp. 589–594. DOI: 10.1109/CICC.2003.1249467 (cit. a p. 6).

[5] Pavan Kumar Hanumolu, Gu-Yeon Wei e Un-Ku Moon. «Equalizers for high-speed serial links». In: *International Journal of High Speed Electronics and Systems* 15.02 (2005), pp. 429–458. DOI: 10.1142/S0129156405003259 (cit. a p. 6).

[6] Mahdi Moshfegh, Alireza Rahmati e Abolfazl Hajisami. «Optimized Decision Feedback Equalizer and Comparison with MLSE Algorithm for GSM Channel». In: mag. 2011, pp. 482–487. DOI: 10.1109/UKSIM.2011.98 (cit. a p. 7).

[7] Matteo Pisati et al. «A 243-mW 1.25–56-Gb/s Continuous Range PAM-4 42.5-dB IL ADC/DAC-Based Transceiver in 7-nm FinFET». In: *IEEE Journal of Solid-State Circuits* 55.1 (2020), pp. 6–18. DOI: 10.1109/JSSC.2019.2936307 (cit. a p. 8).

[8] C.R. Baugh e B.A. Wooley. «A Two's Complement Parallel Array Multiplication Algorithm». In: *IEEE Transactions on Computers* C-22.12 (1973), pp. 1045–1047. DOI: 10.1109/T-C.1973.223648 (cit. alle pp. 12–15, 17, 32).

[9] C. S. Wallace. «A Suggestion for a Fast Multiplier». In: *IEEE Transactions on Electronic Computers* EC-13.1 (1964), pp. 14–17. DOI: 10.1109/PGEC.1964.263830 (cit. alle pp. 12, 23).

[10] L. Dadda. «Some schemes for parallel multipliers». In: *Alta Frequenza* 34 (1965), pp. 349–356. URL: https://cir.nii.ac.jp/crid/1572543024443237760 (cit. alle pp. 12, 23).

[11] L. Dadda. «On parallel digital multipliers». In: *Reprinted from Alta Frequenza* 45 (1976), pp. 547–580. URL: https://cir.nii.ac.jp/crid/1570572701507815168 (cit. alle pp. 12, 23).

[12] Raja Desella. «Design and Implementation of Advanced Modified Booth Encoding Multiplier». In: *International Journal of Engineering Science* 2 (ago. 2013), pp. 60–68 (cit. alle pp. 12, 21).

[13] Kelly Liew Suet Swee e Lo Hai Hiung. «Performance comparison review of Radix-based multiplier designs». In: *2012 4th International Conference on Intelligent and Advanced Systems (ICIAS2012)*. Vol. 2. 2012, pp. 854–859. DOI: 10.1109/ICIAS.2012.6306134 (cit. a p. 20).

[14] Shiann-Rong Kuang, Jiun-Ping Wang e Cang-Yuan Guo. «Modified Booth Multipliers With a Regular Partial Product Array». In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 56.5 (2009), pp. 404–408. DOI: 10.1109/TCSII.2009.2019334 (cit. alle pp. 21, 22, 32).

[15] Zhijun Huang e M.D. Ercegovac. «High-performance low-power left-to-right array multiplier design». In: *IEEE Transactions on Computers* 54.3 (2005), pp. 272–283. DOI: 10.1109/TC.2005.51 (cit. a p. 21).

[16] K.A.C. Bickerstaff, M. Schulte e E.E. Swartzlander. «Reduced area multipliers». In: *Proceedings of International Conference on Application Specific Array Processors (ASAP '93)*. 1993, pp. 478–489. DOI: 10.1109/ASAP.1993.397168 (cit. alle pp. 24–26).

[17] Swartzlander. «Merged Arithmetic». In: *IEEE Transactions on Computers* C-29.10 (1980), pp. 946–950. DOI: 10.1109/TC.1980.1675482 (cit. a p. 25).

[18] Chip-Hong Chang, Jiangmin Gu e Mingyan Zhang. «Ultra low-voltage low-power CMOS 4-2 and 5-2 compressors for fast arithmetic circuits». In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 51.10 (2004), pp. 1985–1997. DOI: 10.1109/TCSI.2004.835683 (cit. a p. 29).

[19] Sreehari Veeramachaneni, Kirthi M Krishna, Lingamneni Avinash, Sreekanth Reddy Puppala e M.B. Srinivas. «Novel Architectures for High-Speed and Low-Power 3-2, 4-2 and 5-2 Compressors». In: *20th International Conference on VLSI Design held jointly with 6th International Conference on Embedded Systems (VLSID'07)*. 2007, pp. 324–329. DOI: 10.1109/VLSID.2007.116 (cit. a p. 30).

[20] Ardalan Najafi, Babak Mazloom-nezhad e Amir Najafi. «Low-power and high-speed 4-2 compressor». In: *2013 36th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. 2013, pp. 66–69 (cit. a p. 30).

[21] Honglan Jiang, Cong Liu, Naman Maheshwari, Fabrizio Lombardi e Jie Han. «A comparative evaluation of approximate multipliers». In: *2016 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*. 2016, pp. 191–196. DOI: 10.1145/2950067.2950068 (cit. a p. 41).

[22] Parag Kulkarni, Puneet Gupta e Milos Ercegovac. «Trading Accuracy for Power with an Underdesigned Multiplier Architecture». In: *2011 24th Internatioal Conference on VLSI Design*. 2011, pp. 346–351. DOI: 10.1109/VLSID.2011.51 (cit. a p. 41).

[23] Semeen Rehman, Walaa El-Harouni, Muhammad Shafique, Akash Kumar, Jorg Henkel e Jörg Henkel. «Architectural-space exploration of approximate multipliers». In: *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 2016, pp. 1–8. DOI: 10.1145/2966986.2967005 (cit. a p. 41).

[24] G. A. Gillani, M. A. Hanif, B. Verstoep, S. H. Gerez, M. Shafique e A. B. J. Kokkeler. «MACISH: Designing Approximate MAC Accelerators With Internal-Self-Healing». In: *IEEE Access* 7 (2019), pp. 77142–77160. DOI: 10.1109/ACCESS.2019.2920335 (cit. a p. 41).

[25] Srinivasan Narayanamoorthy, Hadi Asghari Moghaddam, Zhenhong Liu, Taejoon Park e Nam Sung Kim. «Energy-Efficient Approximate Multiplication for Digital Signal Processing and Classification Applications». In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 23.6 (2015), pp. 1180–1184. DOI: 10.1109/TVLSI.2014.2333366 (cit. a p. 41).

[26] Soheil Hashemi, R. Iris Bahar e Sherief Reda. «DRUM: A Dynamic Range Unbiased Multiplier for approximate applications». In: *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 2015, pp. 418–425. DOI: 10.1109/ICCAD.2015.7372600 (cit. a p. 41).

[27] Shaghayegh Vahdat, Mehdi Kamal, Ali Afzali-Kusha e Massoud Pedram. «TOSAM: An Energy-Efficient Truncation- and Rounding-Based Scalable Approximate Multiplier». In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 27.5 (2019), pp. 1161–1173. DOI: 10.1109/TVLSI.2018.2890712 (cit. a p. 41).

[28] Georgios Zervakis, Kostas Tsoumanis, Sotirios Xydis, Dimitrios Soudris e Kiamal Pekmestzi. «Design-Efficient Approximate Multiplication Circuits Through Partial Product Perforation». In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 24.10 (2016), pp. 3105–3117. DOI: 10.1109/TVLSI.2016.2535398 (cit. a p. 41).

[29] Weiqiang Liu, Jiahua Xu, Danye Wang, Chenghua Wang, Paolo Montuschi e Fabrizio Lombardi. «Design and Evaluation of Approximate Logarithmic Multipliers for Low Power Error-Tolerant Applications». In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 65.9 (2018), pp. 2856–2868. DOI: 10.1109/TCSI.2018.2792902 (cit. a p. 41).

[30] Mohammad Saeed Ansari, Bruce F. Cockburn e Jie Han. «A Hardware-Efficient Logarithmic Multiplier with Improved Accuracy». In: *2019 Design, Automation and Test in Europe Conference and Exhibition (DATE)*. 2019, pp. 928–931. DOI: 10.23919/DATE.2019.8714868 (cit. a p. 41).

[31] Nicola Petra, Davide De Caro, Valeria Garofalo, Ettore Napoli e Antonio Giuseppe Maria Strollo. «Design of Fixed-Width Multipliers With Linear Compensation Function». In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 58.5 (2011), pp. 947–960. DOI: 10.1109/TCSI.2010.2090572 (cit. a p. 41).

[32] Davide De Caro, Nicola Petra, Antonio Giuseppe Maria Strollo, Fabio Tessitore e Ettore Napoli. «Fixed-Width Multipliers and Multipliers-Accumulators With Min-Max Approximation Error». In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 60.9 (2013), pp. 2375–2388. DOI: 10.1109/TCSI.2013.2245252 (cit. a p. 41).

[33] Mohsen Imani, Daniel Peroni e Tajana Rosing. «CFPU: Configurable floating point multiplier for energy-efficient computing». In: *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*. 2017, pp. 1–6. DOI: 10.1145/3061639.3062210 (cit. a p. 41).

[34] Weiqiang Liu, Liangyu Qian, Chenghua Wang, Honglan Jiang, Jie Han e Fabrizio Lombardi. «Design of Approximate Radix-4 Booth Multipliers for Error-Tolerant Computing». In: *IEEE Transactions on Computers* 66.8 (2017), pp. 1435–1441. DOI: 10.1109/TC.2017.2672976 (cit. a p. 41).

[35] Weiqiang Liu, Tian Cao, Peipei Yin, Yuying Zhu, Chenghua Wang, Earl E. Swartzlander e Fabrizio Lombardi. «Design and Analysis of Approximate Redundant Binary Multipliers». In: *IEEE Transactions on Computers* 68.6 (2019), pp. 804–819. DOI: 10.1109/TC.2018.2890222 (cit. a p. 41).

[36] Vasileios Leon, Konstantinos Asimakopoulos, Sotirios Xydis, Dimitrios Soudris e Kiamal Pekmestzi. «Cooperative Arithmetic-Aware Approximation Techniques for Energy-Efficient Multipliers». In: *2019 56th ACM/IEEE Design Automation Conference (DAC)*. 2019, pp. 1–6 (cit. a p. 41).

[37] Antonio Giuseppe Maria Strollo, Ettore Napoli, Davide De Caro, Nicola Petra e Gennaro Di Meo. «Comparison and Extension of Approximate 4-2 Compressors for Low-Power Approximate Multipliers». In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 67.9 (2020), pp. 3021–3034. DOI: 10.1109/TCSI.2020.2988353 (cit. a p. 42).

[38] Amir Momeni, Jie Han, Paolo Montuschi e Fabrizio Lombardi. «Design and Analysis of Approximate Compressors for Multiplication». In: *IEEE Transactions on Computers* 64.4 (2015), pp. 984–994. DOI: 10.1109/TC.2014.2308214 (cit. a p. 42).

[39] Suganthi Venkatachalam e Seok-Bum Ko. «Design of Power and Area Efficient Approximate Multipliers». In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 25.5 (2017), pp. 1782–1786. DOI: 10.1109/TVLSI.2016.2643639 (cit. a p. 43).

[40] Zhixi Yang, Jie Han e Fabrizio Lombardi. «Approximate compressors for error-resilient multiplier design». In: *2015 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)*. 2015, pp. 183–186. DOI: 10.1109/DFT.2015.7315159 (cit. alle pp. 45, 59).

[41] Chia-Hao Lin e Ing-Chao Lin. «High accuracy approximate multiplier with error correction». In: *2013 IEEE 31st International Conference on Computer Design (ICCD)*. 2013, pp. 33–38. DOI: 10.1109/ICCD.2013.6657022 (cit. a p. 49).

[42] Minho Ha e Sunggu Lee. «Multipliers With Approximate 4–2 Compressors and Error Recovery Modules». In: *IEEE Embedded Systems Letters* 10.1 (2018), pp. 6–9. DOI: 10.1109/LES.2017.2746084 (cit. alle pp. 51, 58, 62).

[43] Omid Akbari, Mehdi Kamal, Ali Afzali-Kusha e Massoud Pedram. «Dual-Quality 4:2 Compressors for Utilizing in Dynamic Accuracy Configurable Multipliers». In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 25.4 (2017), pp. 1352–1361. DOI: 10.1109/TVLSI.2016.2643003 (cit. a p. 52).

[44] Farnaz Sabetzadeh, Mohammad Hossein Moaiyeri e Mohammad Ahmadinejad. «A Majority-Based Imprecise Multiplier for Ultra-Efficient Approximate Image Multiplication». In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 66.11 (2019), pp. 4200–4208. DOI: 10.1109/TCSI.2019.2918241 (cit. alle pp. 55, 71).

[45] Mohammad Ahmadinejad, Mohammad Moaiyeri e Farnaz Sabetzadeh. «Energy and Area Efficient Imprecise Compressors for Approximate Multiplication at Nanoscale». In: *AEU - International Journal of Electronics and Communications* 110 (ago. 2019), p. 152859. DOI: 10.1016/j.aeue.2019.152859 (cit. a p. 56).

[46] Antonio Giuseppe Maria Strollo, Ettore Napoli, Davide De Caro, Nicola Petra e Gennaro Di Meo. «Comparison and Extension of Approximate 4-2 Compressors for Low-Power Approximate Multipliers». In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 67.9 (2020), pp. 3021–3034. DOI: 10.1109/TCSI.2020.2988353 (cit. alle pp. 57, 61, 62).

[47] Darjn Esposito, Antonio Giuseppe Maria Strollo, Ettore Napoli, Davide De Caro e Nicola Petra. «Approximate Multipliers Based on New Approximate Compressors». In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 65.12 (2018), pp. 4169–4182. DOI: 10.1109/TCSI.2018.2839266 (cit. a p. 62).

[48] *IA Title: Common Electrical I/O (CEI) - Electrical and Jitter Interoperability agreements for 6G+ bps, 11G+ bps, 25G+ bps, 56G+ bps and 112G+ bps I/O*. URL: https://www.oiforum.com/wp-content/uploads/OIF-CEI-05.2.pdf (cit. a p. 77).

[49] «IEEE Standard for Ethernet Amendment 4: Physical Layer Specifications and Management Parameters for 100 Gb/s, 200 Gb/s, and 400 Gb/s Electrical Interfaces Based on 100 Gb/s Signaling». In: *IEEE Std 802.3ck-2022 (Amendment to IEEE Std 802.3-2022 as amended by IEEE Std 802.3dd-2022, IEEE Std 802.3cs-2022, and IEEE Std 802.3db-2022)* (2022), pp. 1–316. DOI: 10.1109/IEEESTD.2022.9999414 (cit. a p. 77).

Ringraziamenti

Desidero esprimere la mia profonda gratitudine a coloro che hanno reso possibile il mio coinvolgimento in questo lavoro, creando un'atmosfera di serenità e motivazione che ha arricchito notevolmente la mia esperienza professionale.

Vorrei ringraziare i miei relatori, il professor Maurizio Martina e il professor Guido Masera, per avermi introdotto a questa nuova realtà e per aver fornito preziose risposte che hanno facilitato il mio percorso professionale.

Ringrazio Piero Bonifacino e Mauro Natuzzi per avermi aperto le porte di questo settore e per avermi incoraggiato a superarmi giorno dopo giorno, rendendo ogni tappa di questo percorso non solo piacevole ma anche ricca di emozioni, fino al conseguimento di questo importante traguardo. Desidero inoltre ringraziare calorosamente i miei colleghi per la loro inestimabile collaborazione e il costante supporto, che ha trasformato ogni sfida in una preziosa occasione di apprendimento e sviluppo personale.