

Università della Calabria

**Dipartimento di Ingegneria Informatica, Modellistica,
Elettronica e Sistemistica**

Corso di Laurea Triennale in Ingegneria Informatica

Tesi di Laurea

**PROGETTAZIONE E ANALISI DI UN
MOLTIPLICATORE DI WALLACE BASATO SU FPGA**

Relatore

Prof.ssa Stefania Perri

Candidato

Giorgio Ubbriaco
Matricola 209899

Anno Accademico 2021-2022

“Non accontentarti dell’orizzonte...cerca l’infinito!”

Ai miei genitori

Indice

Indice.....	3
Indice delle Figure.....	5
1. Obiettivo della Tesi	8
2. Introduzione	9
2.1. ZedBoard™.....	9
2.1.1. FPGA	10
2.2. VHDL	11
2.3. Vivado Design Suite	12
3. Circuito Digitale.....	14
3.1. Half-Adder	17
3.2. Full-Adder.....	18
3.3. Ripple Carry Adder.....	19
3.3.1. Ripple Carry Adder Unsigned.....	20
3.3.2. Ripple Carry Adder Signed.....	21
3.4. Registro	22
4. Moltiplicatore.....	23
4.1. Moltiplicatore “Carta e Penna”	27
4.2. Moltiplicatore di Wallace.....	29
4.2.1. Moltiplicatore di Wallace a 8 bit.....	31
4.2.2. Moltiplicatore di Wallace a 16 bit.....	33
5. Design Flow	35
5.1. RTL Schematic	37

5.1.1. Moltiplicatore “Carta e Penna”	37
5.1.2. Moltiplicatore di Wallace a 8 bit.....	39
5.1.3. Moltiplicatore di Wallace a 16 bit.....	41
5.2. Sintesi	43
5.2.1. Moltiplicatore “Carta e Penna”	43
5.2.2. Moltiplicatore di Wallace a 8 bit.....	44
5.2.3. Moltiplicatore di Wallace a 16 bit.....	45
5.3. Constraint di clock	46
5.4. Implementazione.....	47
5.4.1. Moltiplicatore “Carta e Penna”	48
5.4.2. Moltiplicatore di Wallace a 8 bit.....	49
5.4.3. Moltiplicatore di Wallace a 16 bit.....	51
5.5. Simulazione	52
5.5.1. Behavioral Simulation.....	55
5.5.2. Post-Synthesis Timing Simulation.....	58
5.5.3. Post-Implementation Timing Simulation.....	60
5.6. Report	62
5.6.1. Utilizzazione delle Risorse.....	62
5.6.2. Potenza.....	65
6. Conclusioni	70
7. Bibliografia	72
8. Ringraziamenti	75

Indice delle Figure

Figura 1 ZedBoard	9
Figura 2 FPGA	10
Figura 3 <i>Inverter CMOS</i>	14
Figura 4 Caratteristica di trasferimento di un inverter CMOS	15
Figura 5 Tabella di funzionamento delle 5 regioni identificate nella caratteristica di trasferimento di un <i>inverter CMOS</i>	16
Figura 6 Tabella di verità di un Half-Adder	17
Figura 7 RTL Schematic di un Half-Adder	17
Figura 8 Tabella di verità di un Full-Adder.....	18
Figura 9 RTL Schematic di un Full-Adder.....	18
Figura 10 RTL Schematic di un Ripple Carry Adder Unsigned ad 8 bit	20
Figura 11 RTL Schematic di un Ripple Carry Adder Signed ad 8 bit	21
Figura 12 RTL Schematic di un registro ad 8 bit sensibile al fronte di salita del clock.....	22
Figura 13 Tassonomia dei moltiplicatori	24
Figura 14 Legenda della struttura del Moltiplicatore "Carta e Penna"	28
Figura 15 Struttura del Moltiplicatore "Carta e Penna"	28
Figura 16 Legenda delle strutture dei Moltiplicatori di Wallace a 8 bit e 16 bit.....	30
Figura 17 Struttura del Moltiplicatore di Wallace a 8 bit.....	31
Figura 18 Andamento dell'altezza dell'albero nel Moltiplicatore di Wallace a 8 bit.....	32
Figura 19 Struttura del Moltiplicatore di Wallace a 16 bit.....	33
Figura 20 Andamento dell'altezza dell'albero nel Moltiplicatore di Wallace a 16 bit.....	33
Figura 21 Design Flow.....	35
Figura 22 RTL Schematic del Moltiplicatore "Carta e Penna"	37

Figura 23 RTL Schematic del Moltiplicatore "Carta e Penna" con zoom all'interno dei moduli	38
Figura 24 RTL Schematic del Moltiplicatore di Wallace a 8 bit	39
Figura 25 RTL Schematic del Moltiplicatore di Wallace a 8 bit con zoom all'interno dei moduli	40
Figura 26 RTL Schematic del Moltiplicatore di Wallace a 16 bit	41
Figura 27 RTL Schematic del Moltiplicatore di Wallace a 16 bit con zoom all'interno dei moduli	42
Figura 28 Synthesis Schematic del Moltiplicatore "Carta e Penna" ...	43
Figura 29 Synthesis Schematic del Moltiplicatore di Wallace a 8 bit	44
Figura 30 Synthesis Schematic del Moltiplicatore di Wallace a 16 bit	45
Figura 31 Constraint di clock.....	47
Figura 32 Implementazione del Moltiplicatore "Carta e Penna" su device	48
Figura 33 Design Timing Summary del Moltiplicatore "Carta e Penna"	48
Figura 34 Implementazione del Moltiplicatore di Wallace a 8 bit su device	49
Figura 35 Design Timing Summary del Moltiplicatore di Wallace a 8 bit.....	50
Figura 36 Implementazione del Moltiplicatore di Wallace a 16 bit su device	51
Figura 37 Design Timing Summary del Moltiplicatore di Wallace a 16 bit.....	51
Figura 38 Behavioral Simulation del Moltiplicatore "Carta e Penna" (1)	55
Figura 39 Behavioral Simulation del Moltiplicatore "Carta e Penna" (2)	55
Figura 40 Behavioral Simulation del Moltiplicatore di Wallace a 8 bit (1)	56
Figura 41 Behavioral Simulation del Moltiplicatore di Wallace a 8 bit (2)	56

Figura 42 Behavioral Simulation del Moltiplicatore di Wallace a 16 bit (1)	57
Figura 43 Behavioral Simulation del Moltiplicatore di Wallace a 16 bit (2)	57
Figura 44 Post-Synthesis Timing Simulation del Moltiplicatore "Carta e Penna".....	58
Figura 45 Post-Synthesis Timing Simulation del Moltiplicatore di Wallace a 8 bit.....	59
Figura 46 Post-Synthesis Timing Simulation del Moltiplicatore di Wallace a 16 bit.....	59
Figura 47 Post-Implementation Timing Simulation del Moltiplicatore "Carta e Penna"	60
Figura 48 Post-Implementation Timing Simulation del Moltiplicatore di Wallace a 8 bit.....	61
Figura 49 Post-Implementation Timing Simulation del Moltiplicatore di Wallace a 16 bit.....	61
Figura 50 Report di utilizzazione delle risorse del Moltiplicatore "Carta e Penna"	63
Figura 51 Report di utilizzazione delle risorse del Moltiplicatore di Wallace a 8 bit.....	64
Figura 52 Report di utilizzazione delle risorse del Moltiplicatore di Wallace a 16 bit.....	65
Figura 53 Report di potenza del Moltiplicatore "Carta e Penna" (1) ..	67
Figura 54 Report di potenza del Moltiplicatore "Carta e Penna" (2) ..	67
Figura 55 Report di potenza del Moltiplicatore di Wallace a 8 bit (1)	68
Figura 56 Report di potenza del Moltiplicatore di Wallace a 8 bit (2)	68
Figura 57 Report di potenza del Moltiplicatore di Wallace a 16 bit (1)	69
Figura 58 Report di potenza del Moltiplicatore di Wallace a 16 bit (2)	69

1. Obiettivo della Tesi

Questo lavoro di tesi si propone l'obiettivo di progettare ed analizzare un Moltiplicatore di Wallace a 16 e 8 bit.

Nella prima fase verrà esaminata la scheda di sviluppo di riferimento, il linguaggio di descrizione dell'hardware utilizzato e la suite software impiegata.

Successivamente verranno approfonditi quelli che sono i componenti digitali considerati analizzandone la loro architettura.

Infine, verrà analizzata la struttura progettata del moltiplicatore in questione, confrontando, inoltre, le simulazioni e i report ottenuti con quelli relativi al Moltiplicatore standard “Carta e Penna”.

2. Introduzione

Al giorno d'oggi, la maggior parte dei sistemi elettronici e meccanici utilizzano schede di sviluppo¹. Basti pensare ai telefoni, ai sistemi di diagnostica medica, agli elettrodomestici e alle auto per capire la loro importanza nel mondo odierno. Il loro compito principale è quello di coinvolgere il mondo fisico, interagendo direttamente con sensori² e attuatori³.

2.1. ZedBoard™

Una piattaforma ideale sia per sviluppatori principianti sia per quelli esperti è la ZedBoard™. Essa è una scheda di sviluppo a basso costo basata sul Xilinx Zynq-7000 All Programmable SoC. È composta dal processore⁴ multi-core⁵ Corex-A9 con frequenza⁶ di funzionamento massima pari a *667 MHz*, una memoria da *512 MB* di tipologia DDR3 e 85,000 Series-7 Programmable Logic (PL) tale da essere destinata per un utilizzo in molte applicazioni. Inoltre, prevede uno slot per microSD e diverse periferiche tale da renderla una piattaforma versatile e flessibile in molti usi.

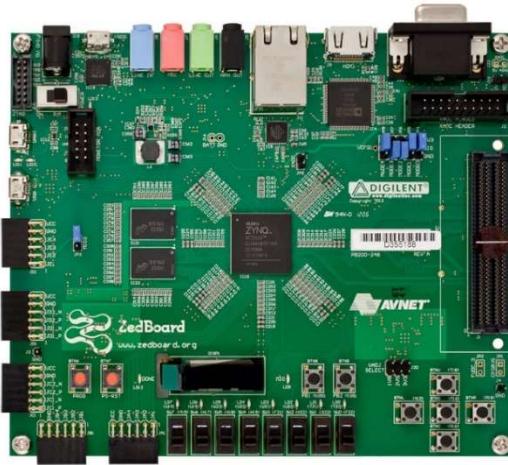


Figura 1 ZedBoard

¹ Una scheda di sviluppo è un dispositivo contenente un microprocessore, dedicato all'elaborazione dell'informazione, e tutta la logica e i componenti di supporto necessari a realizzare l'applicazione di interesse.

² Un sensore è un dispositivo che effettua una misurazione. La misura può essere di ogni genere come, ad esempio, una grandezza fisica o un parametro vitale.

³ Un attuatore è un dispositivo che converte dell'energia da una forma ad un'altra. Nello specifico, trasforma un comando, elaborato da una scheda elettronica, in un'azione fisica.

⁴ Un processore è un dispositivo hardware dedicato all'elaborazione di istruzioni, a partire da un insieme di comandi scritti in linguaggio macchina.

⁵ Un processore multi-core è un processore che presenta due o più unità di calcolo (core) sulla stessa piastra di silicio. Ogni core ha a disposizione la propria cache ed il proprio insieme di registri.

⁶ La frequenza di funzionamento definisce il numero di operazioni svolte in un determinato periodo di tempo.

2.1.1. FPGA

Oggigiorno, l'interesse relativo ai sistemi riprogrammabili tende ad aumentare sempre di più. Tanto è vero che una tra le figure più richieste nel mondo ingegneristico è quella dello sviluppatore competente sia in ambito software che hardware, capace di progettare componenti HW e SW per sistemi elettronici complessi. Nello specifico, sfruttando schede riconfigurabili, è possibile avere flessibilità ed efficienza pur mantenendo una dissipazione di potenza ed un costo moderati. Quindi, si può evincere come la richiesta di chip, che incorporano un microprocessore a cui sono cablate le logiche programmabili, sia sempre più alta. Uno tra i dispositivi logici programmabili più versatili è l'FPGA. Il *Field Programmable Gate Array* è un dispositivo logico riconfigurabile composto da un circuito integrato le cui funzionalità logiche sono programmabili tramite linguaggi descrittivi hardware come, ad esempio, VHDL e Verilog.

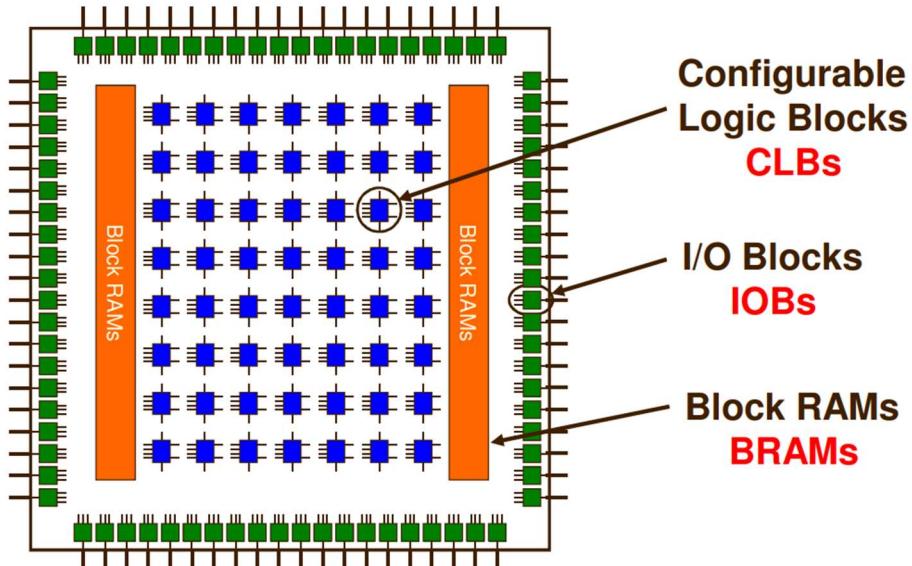


Figura 2 FPGA

Al suo interno è presente una matrice di blocchi logici configurabili, denominati CLB (*Configurable Logic Blocks*), i cui collegamenti fra loro non sono prestabiliti. Ogni blocco presenta una struttura gerarchica. Infatti, è composto da 2 o 4 *logic cell* ed ognuna di quest'ultime è composta da una o più *LUT* (*Look-up Table*). Ogni *LUT* è composta da una memoria SRAM⁷ da 16 bit e da un multiplexer⁸ a 4 ingressi tale che ognuna di esse potrà essere progettata affinché

⁷ SRAM è l'acronimo di Static Random Access Memory. È un tipo di RAM non volatile che non necessita di memory refresh, cioè non prevede la lettura periodica delle informazioni da una determinata sezione della memoria e l'immediata riscrittura delle informazioni lette nella stessa area senza apportare modifiche.

⁸ Il multiplexer, detto anche MUX, è un circuito digitale combinatorio in grado di selezionare uno fra 2^n segnali in ingresso e riportarlo in uscita. La scelta del segnale avviene tramite n input di selezione che rappresentano una determinata combinazione logica.

riproduca una funzione logica. I collegamenti delle CLB sono organizzati tramite matrici di interruttori programmabili (*Programmable Switch Matrix*). Quest'ultimi sono realizzati tramite transistori⁹ NMOS¹⁰ e ciò che viene progettato, infatti, è lo stato degli interruttori (*ON/OFF*). Inoltre, lungo il perimetro della matrice, sono presenti gli IOB (*Input Output Block*) che si occupano dell'interfacciamento input-output del circuito con l'esterno.

2.2. VHDL

Uno tra i linguaggi descrittivi dell'hardware più utilizzati è il VHDL (*Very High Speed Integrated Circuits Hardware Description Language*). Esso, sviluppato dal Dipartimento della Difesa (*DoD*) statunitense, è uno standard IEEE¹¹ utilizzato per la progettazione di sistemi elettronici digitali. Il VHDL si propone come linguaggio indipendente dall'architettura di implementazione tale da essere usato sia per la sintesi che per la simulazione del circuito progettato. Tanto è vero che, tramite la dichiarazione del modulo che si vuole simulare, è possibile creare un file di *testbench* che permette di analizzare il comportamento del circuito, cioè l'output generato, a fronte di determinati ingressi imposti al modulo.

Per quanto riguarda la progettazione generica di un componente elettronico, essa si contraddistingue in due fasi: nella prima viene descritto come *entity*, cioè si descrive la sua interfaccia, mentre nella seconda viene progettato come *architecture*, cioè si descrive la sua struttura interna.

Di notevole importanza è la possibilità di una modellazione di tipo gerarchica, cioè descrivere un componente attraverso altri sotto-moduli già progettati. Essi potranno essere connessi tra di loro tramite dei *signals*. Quest'ultimi appena citati vengono, infatti, utilizzati per modellare l'informazione che transita tra i vari moduli progettati, cioè tra le porte di ognuno e, quindi, assumere i connotati di un'entrata (*input*) o di un'uscita (*output*) di un componente logico, oppure semplicemente per realizzare una determinata funzione logica.

Una tra le caratteristiche fondamentali che lo differenziano dai linguaggi software è la concorrenzialità, cioè diverse parti di codice, dopo essere state tradotte in un

⁹ Il transistor, anche conosciuto come transistor, è un dispositivo a semiconduttore, cioè un componente elettronico che sfrutta le proprietà elettroniche dei materiali. È diventato uno dei componenti hardware più importanti e utilizzati in ambito elettronico.

¹⁰ NMOS è un transistor ad effetto di campo metallo-ossido-semiconduttore (MOSFET) a canale n. È conosciuto anche con l'acronimo NMOSFET.

¹¹ L'Institute of Electrical and Electronics Engineers è la più importante organizzazione internazionale nell'ambito ingegneristico con l'obiettivo di promuovere le scienze tecnologiche.

circuito elettronico, potranno funzionare in maniera parallela in quanto dispongono di risorse hardware dedicate.

2.3. Vivado Design Suite

La Vivado Design Suite è una suite di strumenti progettata da Xilinx per aumentare la produttività complessiva della progettazione, dell'integrazione e dell'implementazione di sistemi che utilizzano dispositivi UltraScale™, 7 series e Versal®, MPSoC Zynq® UltraScale+™ e SoC Zynq®-7000. Essa è utilizzata per la sintesi e l'analisi di progetti scritti in un linguaggio descrittivo dell'hardware di tipologia HDL. Vivado risulta essere un'evoluzione rispetto a Xilinx ISE poiché introduce funzionalità per lo sviluppo di SoC¹² e per la sintesi ad alto livello. Inoltre, la suite permette di analizzare, verificare e modificare il progetto ad ogni fase del processo di progettazione.

Per poter verificare a livello grafico se i moduli progettati sono collegati nella maniera corretta, è possibile generare l'RTL¹³ Schematic del circuito. Esso mostra la logica progettata e come l'informazione fluisce dentro e fuori dal circuito.

Inoltre, la suite prevede la possibilità di effettuare la sintesi del circuito progettato, cioè trasformare il design RTL in una rappresentazione a livello di gate. Infatti, tramite questa fase viene generato il file netlist¹⁴ che sarà utile nello stadio di implementazione. Quest'ultima, prevede il posizionamento e l'instradamento della netlist sulle risorse del dispositivo FPGA.

Molto rilevante è la possibilità di analizzare in maniera dettagliata il comportamento del circuito progettato mediante le simulazioni. Sono previste tre macro-categorie di simulazioni: la Behavioral Simulation che permette di analizzare il comportamento dei moduli RTL senza dover effettuare la sintesi o l'implementazione del circuito, la Post-Synthesis Simulation che permette di effettuare un'analisi del comportamento tenendo conto di delay del sistema approssimativi e la Post-Implementation Simulation che dà la possibilità di verificare il comportamento del circuito attraverso un'emulazione assimilabile a quella del device effettivo. Delle due appena citate, la suite predispone per ognuna due tipologie differenti: la *timing simulation* e la *functional simulation*. La prima prevede l'analisi dei segnali tenendo conto di eventuali delay o di constraint di

¹² SoC è l'acronimo di System on a Chip, cioè un circuito integrato che contiene sullo stesso chip tutti i componenti di elaborazione essenziali.

¹³ RTL è l'acronimo di Register Transfer Language.

¹⁴ Il file netlist contiene una lista relativa alle connessioni (net) elettriche di un circuito elettronico progettato.

timing (ad esempio impostando una frequenza di funzionamento desiderata), mentre la seconda prevede uno studio relativo all'aspetto funzionale del sistema progettato.

Di notevole importanza è la possibilità di accedere ai report generati automaticamente dalla suite così da analizzare gli aspetti implementativi più importanti come, ad esempio, quelli relativi alla dissipazione di potenza e al timing. Quest'ultimi parametri è possibile analizzarli dopo aver eseguito l'elaborazione RTL, la sintesi e l'implementazione.

3. Circuito Digitale

Un circuito digitale è un circuito elettronico il cui funzionamento è basato su un numero finito di livelli di tensione elettrica. I circuiti digitali integrati¹⁵, oltre ad essere classificati per la loro complessità, possono essere classificati per la tecnologia circuitale, cioè la famiglia logica digitale utilizzata per la loro realizzazione. Una tra le famiglie logiche più conosciute è la CMOS¹⁶.

Il circuito CMOS fondamentale è l'*inverter* CMOS. Tramite questa porta logica, è possibile realizzare qualsiasi altra porta logica e di conseguenza qualsiasi altro circuito.

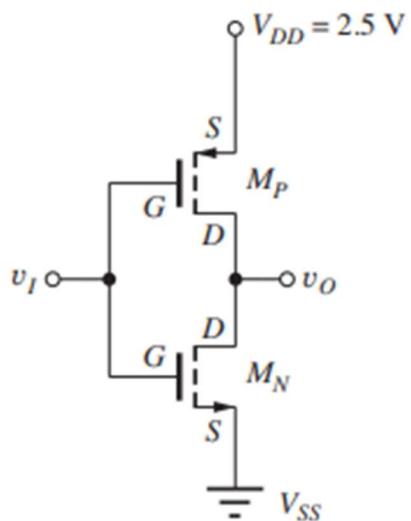


Figura 3 Inverter CMOS

La tabella di verità prevede che l'output sia il complemento dell'input del circuito citato. Infatti, considerando i due soli livelli di tensione ammessi, possiamo associare lo 0 al livello basso (*L* o *low*) e l'1 al livello alto (*H* o *high*). Nella maggior parte dei casi è consentita una tolleranza riguardo i livelli di tensione. Nel caso di tecnologia CMOS, viene considerato un range $V_{ss} \div \frac{V_{dd}}{2}$ per la rappresentazione dello 0 logico ed un range $\frac{V_{dd}}{2} \div V_{dd}$ per la rappresentazione dell'1 logico. Solitamente il valore di tensione V_{dd} per un *inverter* CMOS è un valore compreso tra +2V e +15V mentre quello di V_{ss} è un valore di tensione

¹⁵ I circuiti digitali sono costruiti mediante circuiti integrati (*IC*), cioè circuiti elettronici dove i vari transistori utilizzati sono stati formati, tramite un unico processo fisico-chimico, nello stesso istante.

¹⁶ CMOS è l'acronimo di *complementary metal-oxide semiconductor*, cioè una tecnologia utilizzata in elettronica digitale per la progettazione e la realizzazione di circuiti digitali integrati.

nullo (GND¹⁷). In questo caso, per la descrizione dell'*inverter*, si assumerà un valore di V_{DD} pari a 2.5 V.

L'*inverter CMOS* consiste in una coppia di *transistor MOS* collegati in serie tra l'alimentazione V_{DD} e il valore di tensione nullo V_{SS} . Il primo, a canale p , è collegato alla tensione di alimentazione mentre il secondo, a canale n , è collegato alla tensione nulla. In generale, nel normale funzionamento, un solo *transistor* alla volta è in conduzione. Infatti, quando l'ingresso dell'*inverter* V_{in} è al livello *high*, l'*NMOS* è in conduzione mentre il *PMOS* è spento tale che V_o sarà pari a 0 V. Invece, quando V_i è al livello *low*, l'*NMOS* è *off* mentre il *PMOS* conduce tale che la tensione di uscita V_o sarà pari alla tensione di alimentazione V_{DD} . Indicando con la variabile V_H la tensione in uscita al valore alto, cioè pari a $V_{DD} = 2.5$ V, e con la variabile V_L la tensione in uscita al valore basso, cioè pari a $V_{SS} = 0$ V, assumendo che le tensioni di soglia¹⁸ dei *transistor NMOS* e *PMOS* siano uguali in modulo e indicandole entrambe con la variabile V_T , cioè $V_{TN} = |V_{TP}| = V_T = 0.6$ V, è possibile ricavare la caratteristica di trasferimento¹⁹ dell'*inverter CMOS*.

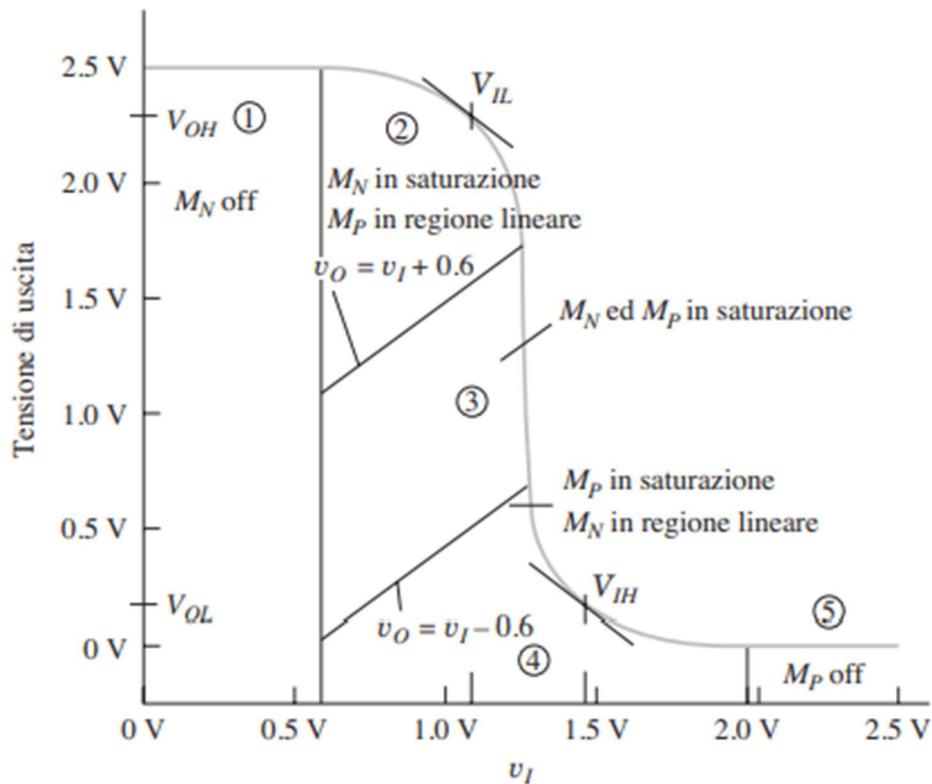


Figura 4 Caratteristica di trasferimento di un inverter CMOS

¹⁷ GND è l'acronimo di *ground* ed indica il valore di tensione nullo, cioè 0 V.

¹⁸ La tensione di soglia (*threshold voltage*) indica la minima differenza di potenziale applicabile tra *gate G* e *source S* del transistore per formare il canale.

¹⁹ La caratteristica di trasferimento di un componente elettronico indica l'insieme delle regioni in cui tale circuito presenta determinate caratteristiche di funzionamento in relazione ai parametri di tensione e/o corrente.

Si possono individuare cinque regioni di funzionamento²⁰:

Regione	Tensione di ingresso V_{in}	Tensione di uscita V_o	Transistore <i>NMOS</i>	Transistore <i>PMOS</i>
1	$V_i \leq V_{TN}$	$V_H = V_{DD}$	Interdizione	Lineare
2	$V_{TN} < V_i \leq V_o + V_{TP}$	Alta	Saturazione	Lineare
3	$V_i \approx \frac{V_{DD}}{2}$	$\frac{V_{DD}}{2}$	Saturazione	Lineare
4	$V_o + V_{TN} < V_i \leq (V_{DD} - V_{TP})$	Bassa	Lineare	Saturazione
5	$V_i \geq (V_{DD} - V_{TP})$	$V_L = 0$	Lineare	Interdizione

Figura 5 Tabella di funzionamento delle 5 regioni identificate nella caratteristica di trasferimento di un *inverter CMOS*

Pertanto, considerando una tensione di soglia $V_{TN} = 0.6 V$, si può evincere come, per un valore di tensione in ingresso V_i minore di V_{TN} , l'*inverter CMOS* risulterà essere nella regione 1 tale che la tensione di uscita, mantenuta dal *PMOSFET*, sarà pari a $V_H = 2.5 V$. Analogamente per una tensione V_i maggiore della differenza $V_{DD} - |V_{TP}|$, il transistore *PMOS* sarà interdetto e la tensione di uscita sarà pari a $V_L = 0 V$. Nelle regioni 2 e 4 si avrà, rispettivamente, una tensione di uscita alta e bassa tale che i transistori risulteranno essere il primo in regione di saturazione ed il secondo in regione lineare, e viceversa. Per quanto riguarda la regione 3, invece, essa raffigura il punto di confine tra tensione in uscita alta e bassa dove entrambi i *transistor* risulteranno essere in regione di saturazione. Pertanto, poiché uno dei due *transistor* sarà interdetto, la dissipazione di potenza a riposo per la coppia complementare di *transistor NMOS* e *PMOS* è relativamente bassa.

Inoltre, bisogna notare che, a causa della non idealità dei componenti, la commutazione dallo stato di *off* allo stato di *on*, e viceversa, non è istantanea. Tanto è vero che, solitamente si fa riferimento ad un condensatore che, tramite la capacità ad esso associata, gestisce il processo di carica e di scarica dell'intero circuito. Infatti, nel transito da un livello logico ad un altro e, quindi, nei momenti in cui entrambi i transistori si trovano in regione di saturazione, la commutazione presenta valori intermedi che rappresentano stati non stabili per l'*inverter CMOS*.

²⁰ La regione di funzionamento di un componente elettronico indica una regione in cui tale circuito presenta determinate caratteristiche in relazione ai parametri di tensione e/o corrente.

3.1. Half-Adder

L'Half-Adder, conosciuto anche come semi-sommatore, è un circuito elettronico digitale che prevede due input e due output. Esso calcola il bit di somma e di riporto considerando due bit in ingresso, senza tenere conto di un eventuale riporto in input.

A	B	S	Cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Figura 6 Tabella di verità di un Half-Adder

Dalla tabella di verità del circuito in questione si può evincere come il bit di somma può essere calcolato considerando l'operazione logica *xor* tra i due operandi, mentre il bit di riporto considerando l'operazione logica *and* sempre tra i due ingressi del semi-sommatore.

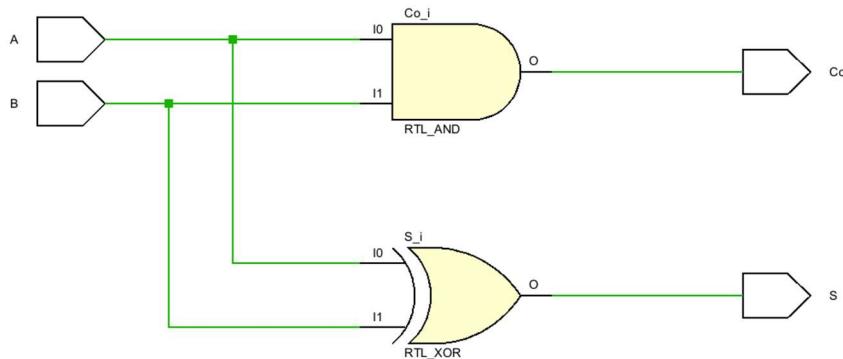


Figura 7 RTL Schematic di un Half-Adder

In questo caso, l'Half-Adder è stato utilizzato all'interno della struttura del Moltiplicatore di Wallace.

3.2. Full-Adder

Il Full-Adder, conosciuto anche come sommatore completo, è un circuito elettronico digitale che prevede tre input e due output. Esso calcola il bit di somma e di riporto considerando due bit in ingresso e tenendo conto del riporto in input.

A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Figura 8 Tabella di verità di un Full-Adder

Dalla tabella di verità del circuito in questione si può evincere un comportamento che può essere riassunto tramite due segnali: il *propagate* ed il *generate*. Il primo permette di capire se il riporto in ingresso viene propagato in uscita, mentre il secondo permette di comprendere se il riporto in uscita è stato generato dagli operandi in input senza tenere conto del riporto in ingresso.

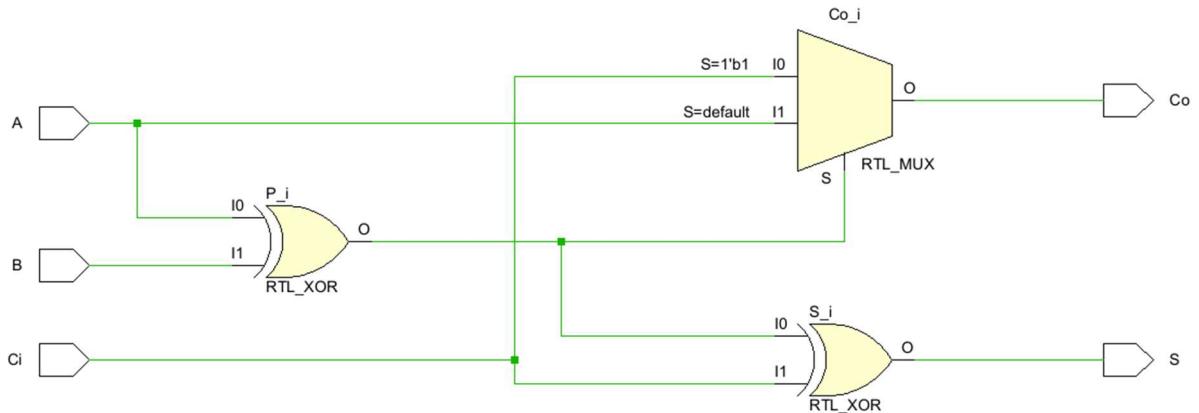


Figura 9 RTL Schematic di un Full-Adder

In questo caso, il Full-Adder è stato utilizzato all'interno della struttura del Moltiplicatore di Wallace e per la realizzazione del Ripple Carry Adder.

3.3. Ripple Carry Adder

Il Ripple Carry Adder è un circuito elettronico digitale che prevede in input due operandi ad N bit e un output ad $N + 1$ bit. Il bit aggiuntivo nella somma corrisponde all’ultimo riporto calcolato in uscita. L’RCA²¹ presenta una struttura composta da Full-Adder disposti in cascata dove il riporto in input al primo Full-Adder corrisponde al riporto in ingresso del Ripple Carry Adder. Inoltre, i riporti in uscita calcolati in ogni FA²² vengono disposti in ingresso come riporti in input ai successivi Full-Adder. Infatti, per un Ripple Carry Adder ad N bit, vengono considerati N Full-Adder in cascata. Pertanto, per calcolare una somma generica ad N bit, il delay complessivo sarà pari a $\tau(n)$ poiché per calcolare ogni bit di somma bisognerà attendere l’avvenuto calcolo del Full-Adder precedente.

In questo caso, il Ripple Carry Adder è stato utilizzato per l’implementazione del VMA²³ all’interno del Moltiplicatore di Wallace e per i sommatori in cascata del Moltiplicatore “Carta e Penna”. Nello specifico sono state utilizzate due versioni di Ripple Carry Adder: il Ripple Carry Adder Unsigned e il Ripple Carry Adder Signed. Il primo utilizzato per operazioni di somma senza segno mentre il secondo per operazioni di somma tenendo conto del segno.

Bisogna precisare che la scelta del Ripple Carry Adder all’interno di questo lavoro di tesi non è casuale. Infatti, si deve notare che il sintetizzatore della suite, in caso di espliciti constraint di progetto, sintetizza direttamente il sommatore in un Ripple Carry Adder. Questo perché le risorse previste all’interno dell’FPGA sono predisposte per un sommatore RCA. A differenza di altri sommatori, il Ripple Carry Adder è il più semplice di tutti poiché prevede dei Full-Adder in cascata con una propagazione del loro riporto. Inoltre, è il meno costoso fra i vari sommatori ad N bit esistenti e, pertanto, dissipava poca potenza. L’approccio ideale sarebbe quello di diminuire il numero di FA così da diminuire il ritardo complessivo del circuito. Infatti, considerando, ad esempio, il Carry Select Adder, si riuscirebbe a diminuire il numero dei Full-Adder da istanziare, e, quindi diminuendo il ritardo complessivo del sistema, ma rinunciando ai valori relativamente bassi di dissipazione di potenza e costo delle risorse. Un’alternativa potrebbe essere il sommatore Carry Look Ahead che permette un minor costo dal punto di vista delle risorse utilizzate ma che purtroppo, rispetto al CSA²⁴, presenta

²¹ RCA è l’acronimo di Ripple Carry Adder.

²² FA è l’acronimo di Full-Adder.

²³ VMA è l’acronimo di Vector Merging Adder, cioè un circuito elettronico digitale che permette la somma di due operandi ad N bit ottenuti mediante operazioni precedenti all’interno di un altro circuito elettronico digitale.

²⁴ CSA è l’acronimo di Carry Select Adder.

alcuni limiti fisici dal punto di vista dell'hardware da utilizzare. Pertanto, la scelta per la progettazione e l'implementazione dei vari circuiti elettronici in questo lavoro di tesi ricade verso il sommatore Ripple Carry Adder.

3.3.1. Ripple Carry Adder Unsigned

Il Ripple Carry Unsigned è composto da N sommatori completi in cascata tale che il risultato ottenuto corrisponderà al valore numerico di un *unsigned*²⁵. Tanto è vero che la sua istanziazione è ottenuta considerando un *for generate* per l'implementazione concorrente dei moduli Full-Adder in cascata.

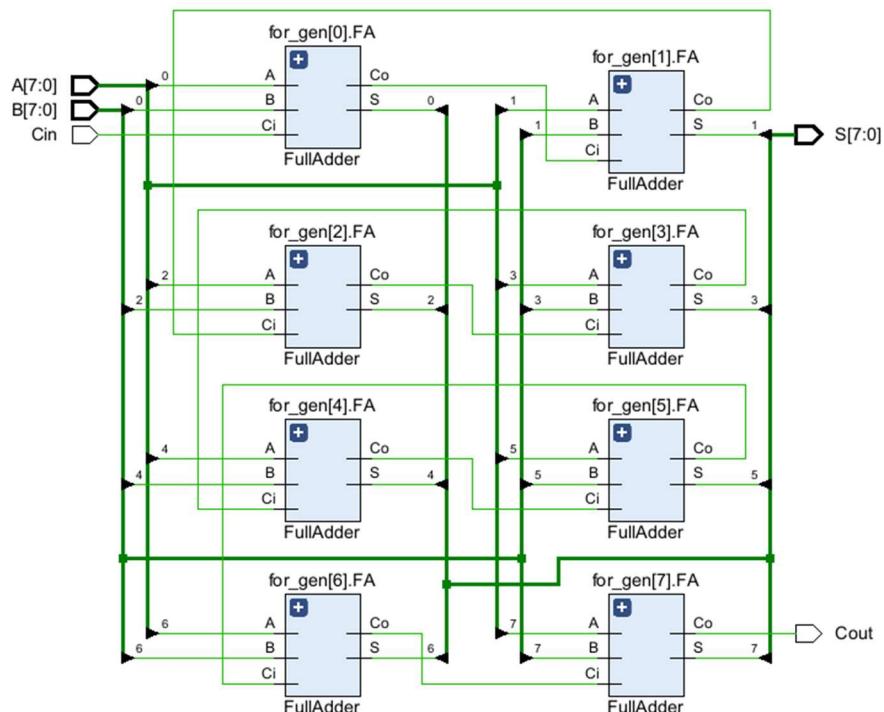


Figura 10 RTL Schematic di un Ripple Carry Adder Unsigned ad 8 bit

Il suo utilizzo è previsto in operazioni di somma quali il calcolo del complemento a due nel caso del secondo operando negativo nel Moltiplicatore di Wallace e nelle operazioni di somma intermedie del Moltiplicatore “Carta e Penna”.

²⁵ Un valore *unsigned* ad N bit è numero che è possibile codificare in un range $-2^{N-1} - 1 \div 2^{N-1}$.

3.3.2. Ripple Carry Adder Signed

Il Ripple Carry Adder Signed è composto da $N + 1$ sommatori completi in cascata tale che il risultato ottenuto corrisponderà al valore numerico di un *signed*²⁶. Tanto è vero che l' N -esimo+1 Full-Adder è ottenuto considerando gli N -esimi bit degli operandi e il riporto in uscita generato dal precedente FA. Bisogna notare che, per quanto riguarda quest'ultimo Full-Adder descritto, verrà considerato solo il bit di somma ottenuto.

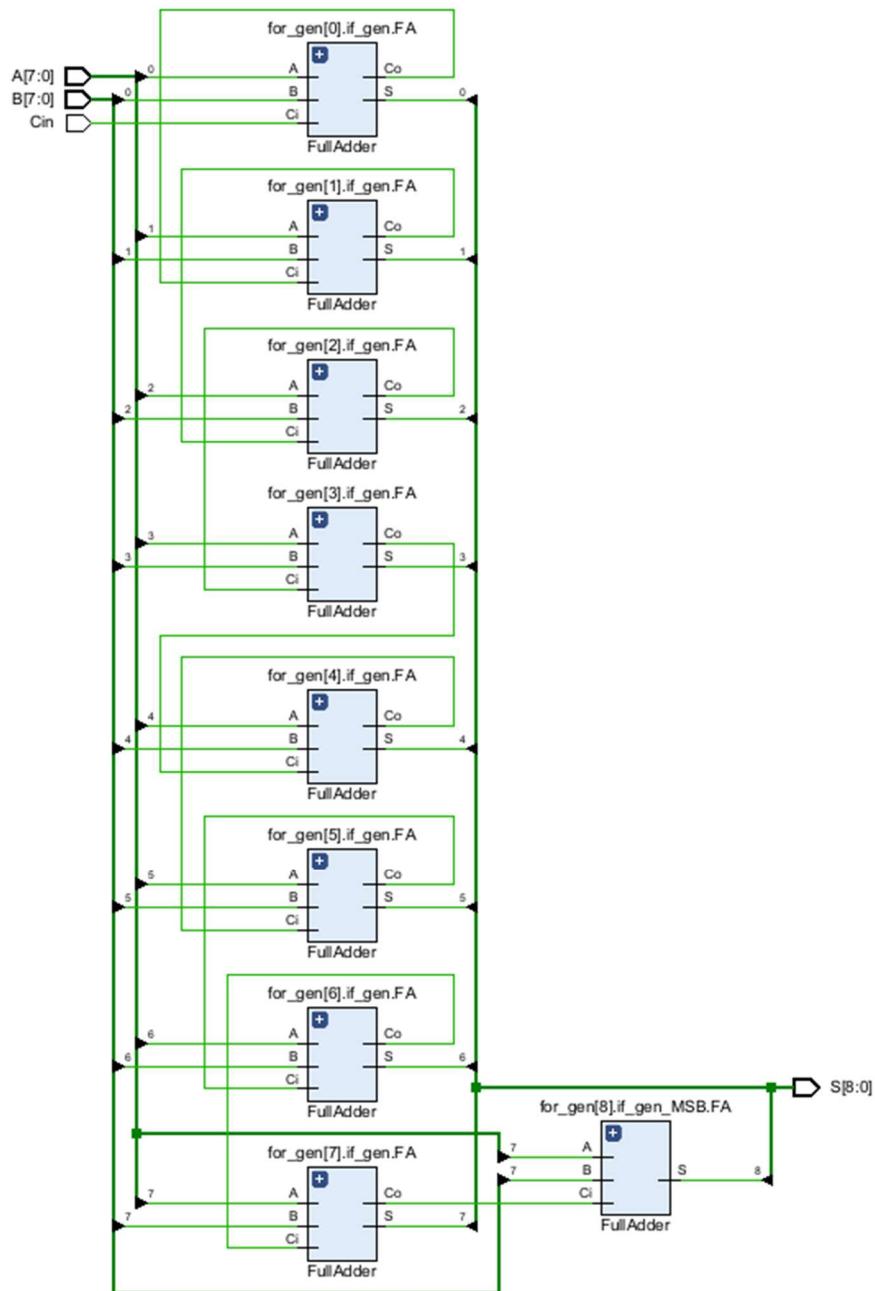


Figura 11 RTL Schematic di un Ripple Carry Adder Signed ad 8 bit

²⁶ Un valore *signed* ad N bit è numero che è possibile codificare in un range $0 \div 2^N$.

Il suo utilizzo è previsto per l'implementazione del VMA all'interno del Moltiplicatore di Wallace e per il calcolo della somma finale all'interno del Moltiplicatore “Carta e Penna”.

3.4. Registro

Il registro è un circuito elettronico digitale composto da un insieme di Flip-Flop²⁷, uno per ogni bit in input. L'insieme dei FF²⁸ lavorerà in parallelo così da garantire il soddisfacimento dei vincoli temporali²⁹. Infatti, un registro lavora correttamente se e solo se vengono rispettati i vincoli definiti in termini di tempo di setup e di hold dei quali si discuterà nel capitolo 5.

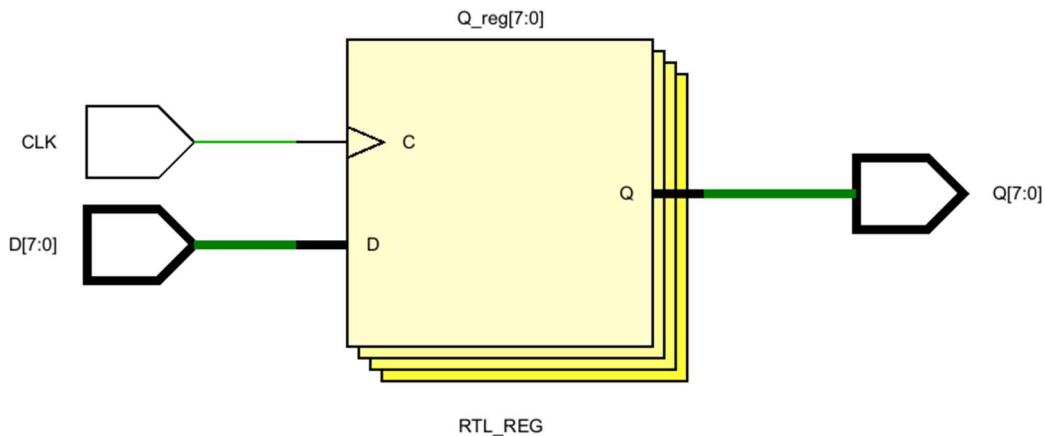


Figura 12 RTL Schematic di un registro ad 8 bit sensibile al fronte di salita del clock

In questo caso il registro progettato è sensibile ai fronti di salita del segnale di clock. Pertanto, l'ingresso generico ad N bit verrà processato in output se e solo se si è verificato un fronte di salita del clock.

Bisogna notare che, trattandosi di un componente sequenziale, la sua progettazione è stata ottenuta considerando soltanto *statement*³⁰ sequenziali.

²⁷ Il Flip-Flop è un circuito elettronico digitale composto da due input e un output. Esso permette di differenziare eventi diversi sul segnale di clock. Pertanto, il F-F è in grado di distinguere e identificare i fronti (di salita o di discesa) del clock così da progettare una determinata sincronizzazione del sistema elettronico.

²⁸ FF è l'acronimo di Flip-Flop.

²⁹ Il vincolo temporale è una condizione di frequenza di funzionamento del circuito elettronico che deve essere rispettata. Solitamente nelle suite di progettazione, esso è conosciuto anche con la denominazione di constraint di timing.

³⁰ Uno statement, all'interno dei linguaggi di programmazione, è un costrutto che permette di esplcitare alcune azioni e istruzioni da eseguire. In VHDL si possono esprimere sia statement sequenziali sia concorrenti.

Tanto è vero che viene utilizzato il costrutto *process* specificando nella sua sensitivity list³¹ quali sono i segnali da cui tale componente dipende.

4. Moltiplicatore

La moltiplicazione è una delle funzioni essenziali in un sistema elettronico digitale. Generalmente il suo uso è previsto in sistemi che si occupano di *digital signal processing*³². Tanto è vero che la moltiplicazione viene utilizzata per la realizzazione di filtri FIR³³, per le trasformate di Fourier veloci³⁴, per la convoluzione³⁵ e molto altro ancora.

Un moltiplicatore digitale è un circuito elettronico digitale che permette la moltiplicazione binaria tra due operandi ad N bit. Esso, come nella moltiplicazione standard, prevede la moltiplicazione bit per bit e, successivamente, la somma di tutti i prodotti parziali intermedi calcolati. Ovviamente, trattandosi di una moltiplicazione binaria, l'operazione di moltiplicazione è ottenuta tramite l'operazione logica *and* tra i due bit i-esimi mentre la somma dei prodotti parziali è ottenuta tramite l'ausilio di sommatori digitali.

Solitamente si cerca di progettare un moltiplicatore relativamente semplice e molto veloce così da minimizzare il ritardo e massimizzare le performance complessive del sistema. Tanto è vero che, nel corso degli anni, i moltiplicatori sono stati oggetto di ricerca riguardo la progettazione e l'implementazione della somma dei prodotti parziali dato che la moltiplicazione bit per bit prevede semplicemente le istanziazioni delle porte logiche *and*.

In genere, il flusso di esecuzione dei moltiplicatori prevede tre fasi: la generazione dei prodotti parziali, la riduzione di quest'ultimi ed una somma finale. Si evince che per moltiplicazione tra operandi con un notevole numero di bit (ad esempio, considerando operandi a 16 bit o 32 bit) sono richiesti compressori notevolmente

³¹ La sensitivity list, associata ad un componente elettronico, è una lista che contiene i segnali a cui il circuito è sensibile.

³² Il digital signal processing indica l'elaborazione numerica dei segnali, cioè una tecnica di analisi ed elaborazione digitale dei segnali elettrici.

³³ Un filtro FIR, acronimo di finite impulse response, è un sistema dinamico caratterizzato da una risposta impulsiva di durata finita.

³⁴ La trasformata di Fourier veloce, conosciuta anche come FFT, permette il calcolo effettuato dalla trasformata di Fourier standard ma tramite un algoritmo molto più efficiente rispetto all'ultima citata.

³⁵ La convoluzione è un'operazione matematica tra due funzioni che consiste nell'integrare il prodotto tra la prima e la seconda traslata di un certo valore.

complessi e articolati. Col tempo, i ricercatori hanno provato a ridurre al minimo i prodotti parziali generati. Infatti, non considerando una riduzione dei prodotti parziali e tenendo come riferimento operandi ad N bit, verrebbero generati N prodotti parziali tale che bisognerebbe calcolare N somme in cascata. Questo approccio, appena descritto, è quello alla base del Moltiplicatore “Carta e Penna”, un particolare moltiplicatore seriale. Un’evoluzione a quest’ultimo sistema digitale citato è stata la progettazione del Moltiplicatore di Wallace, un moltiplicatore ad albero di tipologia seriale-parallelo. Esso prevede una riduzione dei prodotti parziali introducendo un gran numero di sommatori di tipologia Half-Adder e Full-Adder tale da ridurre al minimo le somme in cascata da calcolare. Infatti, esso è conosciuto anche come moltiplicatore ad albero veloce: ad ogni iterazione di riduzione, l’altezza delle somme in cascata da calcolare diminuisce secondo una distribuzione esponenziale che tende al valore minimo di 2, dove l’ultima iterazione non degenera rappresenta l’unica e la sola somma ad M bit da calcolare nell’intero flusso di esecuzione. Infatti, dopo che si è ottenuta un’altezza pari a 2, le iterazioni successive risulterebbero essere degeneri poiché l’altezza rimarrebbe sempre pari a 2.

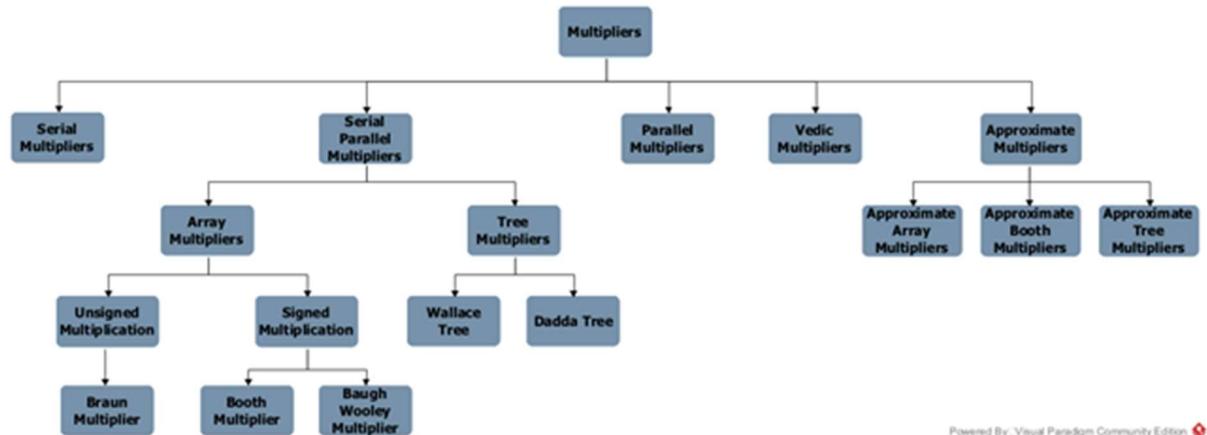


Figura 13 Tassonomia dei moltiplicatori

Osservando la tassonomia sopra schematizzata, esistono diverse tipologie di moltiplicatori digitali:

- il moltiplicatore di tipo *seriale* è il moltiplicatore più semplice da realizzare ma che presenta un delay consistente. Esso consiste nel sommare in cascata i prodotti parziali generati. Ogni somma calcolata viene salvata in un registro apposito e poi, successivamente, disposta in input per la successiva

somma. Il moltiplicatore seriale è necessario in molti ambiti applicativi come la comunicazione digitale e l'implementazione di reti neurali³⁶.

- il moltiplicatore seriale-parallelo opera su ogni bit del moltiplicatore seriale e utilizza un sommatore parallelo per l'accumulo parziale del prodotto. Esso fornisce un compromesso tra l'efficienza di un dispositivo basato su moltiplicatore parallelo e il consumo di tempo richiesto da un moltiplicatore seriale. Questo tipo di circuito è stato introdotto per accelerare applicazioni come filtri digitali, reti neurali e algoritmi di machine learning³⁷. Tra queste tipologie di moltiplicatori è presente il moltiplicatore ad array che utilizza vettori di Full-Adder ed Half-Adder tale che la somma viene eseguita sia in seriale che in parallelo. Bisogna precisare che questo tipo di circuito richiede un elevato numero di porte. I moltiplicatori ad array si dividono in due ulteriori tipologie: quelli basati su moltiplicazione *unsigned* e quelli basati su moltiplicazioni *signed*. Un esempio della prima tipologia appena citata è il moltiplicatore di Braun che presenta un'architettura composta da pochi sommatori e una serie di porte logiche *and*. Inoltre, esso è semplice da progettare e non richiede registri logici. Due esempi, invece, di moltiplicatori ad array basati su moltiplicazione *signed* sono il moltiplicatore di Booth ed il moltiplicatore di Baugh-Wooley. Il primo impiega un metodo codificato: anziché utilizzare direttamente i bit del moltiplicatore, si determina una sua codifica. Successivamente, analizzando i pesi, relativi ai *digit* generati, è possibile ottenere i prodotti parziali relativi all'operazione di moltiplicazione. Questo approccio consente la riduzione del numero di operazioni di somma da effettuare per il calcolo del prodotto. Infatti, considerando un moltiplicatore ad N bit, i prodotti parziali considerati nell'operazione di somma finale saranno $\frac{N}{2}$ anziché N . Per quanto riguarda, invece, il moltiplicatore di Baugh-Wooley, esso è un elemento fondamentale nelle applicazioni DSP³⁸. In questo circuito, il ritardo è minimo e la dissipazione di potenza è decisamente bassa.
- i moltiplicatori paralleli impiegano una strategia *divide et impera*³⁹ per ottimizzare il delay riguardo le operazioni aritmetiche. Confrontandolo con

³⁶ Una rete neurale artificiale è un modello computazionale ispirato alla rete neurale biologica.

³⁷ Il machine learning, conosciuto anche come apprendimento automatico, è un insieme di modelli e metodi riguardanti il campo dell'intelligenza artificiale.

³⁸ DSP è l'acronimo di Digital Signal Processing.

³⁹ La strategia divide et impera è una tecnica utilizzata nell'ambito informatico per la risoluzione di problemi computazionali.

un moltiplicatore seriale, il moltiplicatore parallelo presenta una maggiore velocità in termini di calcolo.

- i moltiplicatori vedici sono basati sulla matematica vedica⁴⁰ e sono impiegati in applicazioni basate sulla comunicazione e sull'elaborazione dei segnali digitali. L'applicazione delle tecniche vediche all'interno dei moltiplicatori permette il calcolo di somme e prodotti parziali in un solo passo e, inoltre, garantisce una riduzione del ritardo di propagazione.
- i moltiplicatori approssimati, utilizzati in ambito di progettazione digitale e che presentano un calcolo non esatto, sono determinanti in termini di velocità e di dissipazione di potenza. Ovviamente, essendo soggetti a variazioni rispetto al risultato esatto, richiedono un'analisi costante degli errori in modo da non avere scostamenti eccessivi rispetto al conteggio effettivo. Tra i moltiplicatori approssimati si possono ricordare il moltiplicatore approssimato di Booth, il moltiplicatore ad array approssimato ed il moltiplicatore approssimato ad albero. Il primo, confrontandolo con il moltiplicatore esatto di Booth, presenta una dissipazione di potenza minore ed introduce una minore complessità logica nella generazione dei prodotti parziali. Per quanto riguarda i moltiplicatori approssimati ad array, essi, utilizzati in applicazioni per la nitidezza delle immagini e per il riconoscimento delle cifre scritte a mano, presentano un consumo energetico minore del 63% rispetto ai moltiplicatori esatti ad array. Invece, i moltiplicatori approssimati ad albero presentano un minor delay rispetto ai corrispettivi esatti e sono applicati in contesti similari a quelli dei moltiplicatori approssimati sopracitati.
- i moltiplicatori esatti ad albero di tipologia seriale-parallelo si differenziano rispetto ai precedenti descritti poiché i prodotti parziali vengono gestiti in maniera differente. La loro compressione avviene secondo una metodologia simile a quella implementata dai sommatori Carry-Save⁴¹. Infatti, tramite l'utilizzo di Full-Adder e Half-Adder, è possibile diminuire gradualmente per iterazioni i prodotti parziali iniziali. Tanto è vero che l'obiettivo di questo tipo di tecnica è quello di diminuire il numero di prodotti parziali da N , considerando che N è il numero dei bit del

⁴⁰ La matematica vedica è un antico sistema di calcolo indiano basato su 16 formule rilevanti in molti rami della matematica come, ad esempio, calcolo, trigonometria e geometria.

⁴¹ Il Carry-Save Adder è un sommatore digitale che prevede la somma di tre operandi ad N bit. Esso si differenzia dagli altri sommatori in quanto genera in uscita due array: uno relativo ai bit di somma e l'altro relativo ai bit di riporto. Solitamente viene utilizzato per il calcolo della somma dei prodotti parziali generati da un moltiplicatore digitale.

moltiplicatore, a 2 tale da effettuare una somma a 2 operandi nella fase finale. Quindi, il delay per ogni compressione effettuata non dipenderà dai sommatori inseriti a cascata, come ad esempio succede in un moltiplicatore seriale, ma dal numero di FA e HA in parallelo e dalla somma finale. Quest'ultima può essere implementata considerando un qualsiasi sommatore ad N bit.

Per quanto riguarda il lavoro di tesi in questione, bisogna precisare che il Moltiplicatore “Carta e Penna” ed il Moltiplicatore di Wallace sono stati progettati considerando due operandi ad N bit: il primo operando *unsigned* ed il secondo *signed*. Inoltre, del primo moltiplicatore citato è stata progettata soltanto la versione ad 8 bit così da effettuare un confronto con il secondo circuito menzionato, del quale però è stata anche progettata la versione a 16 bit.

Per quanto riguarda la tipologia del secondo operando, essendo un numero *signed*, bisogna tenere conto del segno. Pertanto, l’ultimo prodotto parziale non sarà ottenuto effettuando l’*and* bit a bit tra l’ultimo bit del secondo operando e i bit del primo operando, ma bisognerà adottare un approccio diverso per tenere conto del segno del moltiplicatore. Infatti, se il secondo operando è positivo, allora l’ultimo prodotto parziale sarà pari a zero, ovviamente in formato binario. Se, invece, il moltiplicatore è negativo, allora l’ultimo prodotto parziale sarà pari al complemento a due del secondo operando in questione. Ovviamente, per questa operazione di somma, trattandosi dell’ultima somma binaria che viene effettuata, bisognerà utilizzare un Ripple-Carry Adder *signed*. Negli altri casi, per operazioni di somma ordinarie, è stato utilizzato un RCA *unsigned* dato che l’obiettivo era quello di calcolare bit di somma e bit di riporto per ogni operazione di somma tra bit.

4.1. Moltiplicatore “Carta e Penna”

Il moltiplicatore “Carta e Penna” è un circuito elettronico digitale che implementa la moltiplicazione standard tramite sommatori connessi in cascata. Pertanto, considerando operandi ad N bit, tale circuito prevede l’istanziazione di N sommatori aventi ognuno due operandi anch’essi ad N bit. Quindi, si può evincere come il delay totale del moltiplicatore, trascurando il ritardo introdotto dalle porte logiche *and* iniziali, risulti essere $\tau(N)$ proprio perché ogni sommatore successivo dovrà attendere il risultato del sommatore precedente ad esso. Questo

circuito risulta essere uno tra i moltiplicatori più semplici da progettare e implementare tramite un linguaggio descrittivo dell'hardware.

Considerando la seguente legenda:

LEGENDA											
	singolo bit che viene propagato										
	Ripple Carry Adder										
	risultato finale										
s2_4	bit in posizione 4 nel vettore somma 2										
Res_15	quindicesimo bit della somma finale										
en15	bit di estensione del segno in corrispondenza di n15 ('0' se B positivo oppure a15 negato se B negativo)										
n10	bit a10 negato oppure '0' rispettivamente per B negativo o positivo										

Figura 14 Legenda della struttura del Moltiplicatore "Carta e Penna"

è possibile definire la seguente struttura del Moltiplicatore “Carta e Penna”:

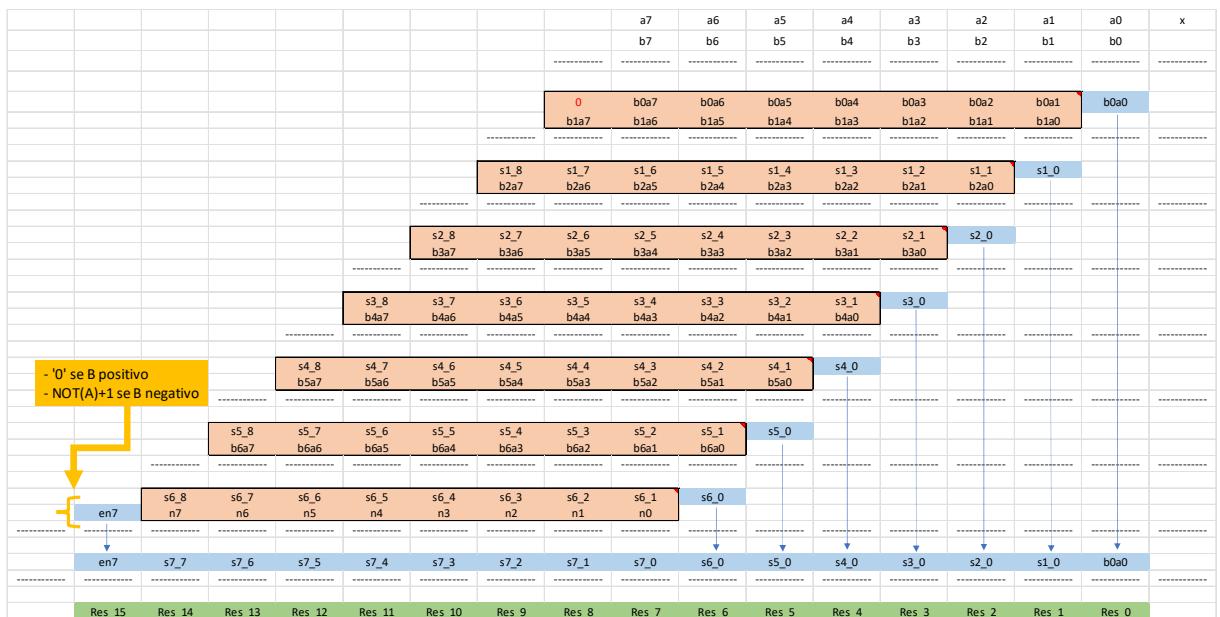


Figura 15 Struttura del Moltiplicatore "Carta e Penna"

Si può notare come il primo operando del primo sommatore presenta come bit più significativo uno 0. Questo è dovuto al fatto che il bit meno significativo del primo operando risulta essere il bit meno significativo del risultato della moltiplicazione. Pertanto, non verrà considerato nelle successive somme in cascata ma sarà direttamente propagato come bit in posizione zero del risultato. Inoltre, il secondo operando risulta essere shiftato verso sinistra di un bit essendo il secondo prodotto parziale calcolato. Tanto è vero che ogni prodotto parziale calcolato, sarà shiftato verso sinistra di un bit come d'altronde succede nella moltiplicazione standard decimale.

4.2. Moltiplicatore di Wallace

Il moltiplicatore di Wallace è un moltiplicatore ad albero veloce di tipologia seriale-parallelo. Esso, rispetto alle altre tipologie di moltiplicatore, presenta un numero di porte e di componenti digitali maggiore. Bisogna precisare che, a fronte di questa maggiore area di occupazione su chip, ci sono notevoli guadagni in termini di velocità di calcolo. Tanto è vero che esso è definito come moltiplicatore veloce ad albero poiché la struttura che lo caratterizza fa sì che l'operazione di somma, aventi come operandi i prodotti parziali calcolati, sia effettivamente più veloce rispetto agli altri tipi di moltiplicatori binari. Infatti, considerando FA e HA, è possibile ridurre notevolmente il numero degli operandi che caratterizzeranno la somma finale. Per la precisione, ad ogni terna di bit corrispondente a tre prodotti parziali differenti verrà associato un Full-Adder, mentre per ogni coppia di bit corrispondente a due prodotti parziali differenti verrà associato un Half-Adder. I corrispettivi bit di somma e di riporto verranno posti rispettivamente nella posizione corrente e nella posizione successiva nell'iterazione susseguente. Si può notare come dalla terna in input al Full-Adder viene generata una coppia di output. Pertanto, la successiva iterazione presenterà sicuramente un numero di prodotti parziali minore rispetto alla corrente e, quindi, un'altezza inferiore. Quindi, nell'iterazione finale, avendo un'altezza pari a 2, si potrà effettuare la somma conclusiva dei prodotti parziali rimanenti. Questo perché il processo di compressione produrrebbe sempre coppie in output e, pertanto, l'altezza relativa rimarrebbe pari a 2.

La somma finale è stata progettata affinché il delay generato sia minimo. Pertanto, la scelta, riguardo il VMA⁴², è ricaduta sul sommatore Ripple-Carry che per i dispositivi utilizzati risulta essere quello che è gestito nella maniera più efficiente per i motivi citati nel capitolo precedente.

Per quanto riguarda la generazione dei prodotti parziali, il moltiplicatore presenta la solita struttura adottata dagli altri moltiplicatori digitali: viene calcolato l'*and* tra il bit i-esimo del moltiplicatore e il bit j-esimo del moltiplicando.

⁴² Il vector merging adder è il sommatore utilizzato per la somma finale all'interno dei moltiplicatori digitali ad albero. Esso ha il compito di effettuare l'operazione di somma tra due operandi ottenuti tramite una compressione applicata ai prodotti parziali generati dall'operazione di moltiplicazione binaria.

Bisogna notare che il delay totale del moltiplicatore dipende dal numero di componenti logici istanziati. Generalmente, il ritardo totale associato al Moltiplicatore di Wallace è pari a:

$$n \cdot \tau(FA) + n \cdot \tau(HA) + \tau(VMA)$$

dove $\tau(VMA)$ è il ritardo associato al Vector-Merging Adder, $\tau(FA)$ è il delay associato al Full-Adder, $\tau(HA)$ è il ritardo associato all'Half-Adder ed n è il numero di iterazioni in cui sono stati utilizzati i componenti FA ed HA. Ovviamente, il ritardo associato al VMA è pari al ritardo corrispondente al sommatore utilizzato, cioè il Ripple-Carry Adder *signed*. Inoltre, considerando il Ripple-Carry Adder *unsigned* utilizzato per la gestione del segno, il delay totale del Moltiplicatore di Wallace progettato sarà pari a:

$$\begin{aligned} \tau(RCAu) + n \cdot \tau(FA) + n \cdot \tau(HA) + \tau(RCAs) = \\ = N \cdot \tau(FA) + n \cdot \tau(FA) + n \cdot \tau(HA) + M \cdot \tau(FA) \end{aligned}$$

dove $\tau(RCAu)$ è il delay associato al Ripple-Carry Adder *unsigned*, $\tau(RCAs)$ è il ritardo associato al Ripple-Carry Adder *signed*, N è il numero dei bit degli operandi del $RCAu$ ed M è il numero dei bit degli operandi del $RCAs$.

In questo lavoro di tesi, come già sopra citato, sono state progettati due Moltiplicatori di Wallace, uno a 8 bit e l'altro a 16 bit, aventi moltiplicando *unsigned* e moltiplicatore *signed*.

Per l'analisi delle strutture dei Moltiplicatori di Wallace, bisogna far riferimento alla seguente legenda:

LEGENDA	
	singolo bit
	half-adder
	full-adder
	VMA
	risultato finale
h	ALTEZZA
s	SOMMA
r	RIPORTO
fa15S	somma del quindicesimo full-adder utilizzato
fa15R	riporto del quindicesimo full-adder utilizzato
ha15S	somma del quindicesimo half-adder utilizzato
ha15R	riporto del quindicesimo half-adder utilizzato
Res_15	quindicesimo bit della somma finale
en15	bit di estensione del segno in corrispondenza di n15 ('0' se B positivo oppure a15 negato se B negativo)
n10	bit a10 negato oppure '0' rispettivamente per B negativo o positivo

Figura 16 Legenda delle strutture dei Moltiplicatori di Wallace a 8 bit e 16 bit

4.2.1. Moltiplicatore di Wallace a 8 bit

Il moltiplicatore di Wallace a 8 bit presenta 56 porte logiche *and* per la generazione dei prodotti parziali e 36 Full-Adder e 24 Half-Adder per la compressione dei PPs⁴³. Inoltre, è stato utilizzato un Ripple-Carry Adder *unsigned* per il calcolo del prodotto parziale in corrispondenza dell'ultimo bit del moltiplicatore e un Ripple-Carry Adder *signed* per il calcolo della somma finale.

Pertanto, è possibile definire la seguente struttura del Moltiplicatore di Wallace ad 8 bit:

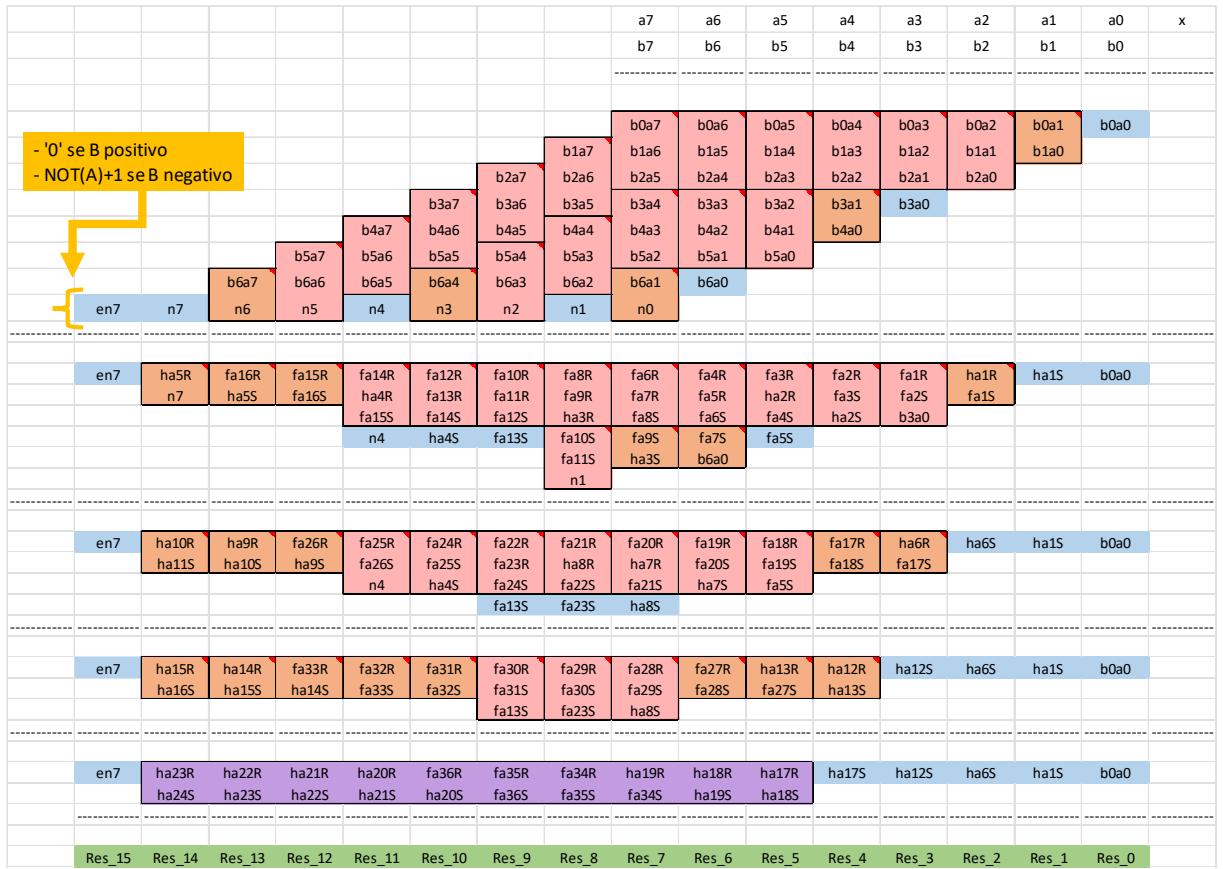


Figura 17 Struttura del Moltiplicatore di Wallace a 8 bit

Si può notare come durante le iterazioni, l'altezza sia diminuita notevolmente. Effettuando un plot riguardo l'andamento dell'altezza è possibile osservare come essa sia decrementata dal valore iniziale di 8, momento in cui sono stati generati i prodotti parziali, fino al valore di 2, momento in cui viene calcolata la somma finale tramite il Vector Merging Adder.

⁴³ PPs è l'acronimo di Partial Products.



Figura 18 Andamento dell'altezza dell'albero nel Moltiplicatore di Wallace a 8 bit

Tanto è vero che, osservando la struttura progettata, si può notare come, ad esempio, nell’iterazione 1 in corrispondenza della colonna che contiene il maggior numero di bit sono presenti due Full-Adder, i quali avendo in input una terna ognuno, generano una coppia di output ciascuno facendo diminuire l’altezza di 2 unità.

4.2.2. Moltiplicatore di Wallace a 16 bit

Il Moltiplicatore di Wallace a 16 bit presenta 240 porte logiche *and* per la generazione dei prodotti parziali e 196 Full-Adder e 81 Half-Adder per la compressione dei PPs. Inoltre, è stato utilizzato un Ripple-Carry Adder *unsigned* per il calcolo del prodotto parziale in corrispondenza dell'ultimo bit del moltiplicatore e un Ripple-Carry Adder *signed* per il calcolo della somma finale.

Pertanto, è possibile definire la seguente struttura del Moltiplicatore di Wallace ad 16 bit:

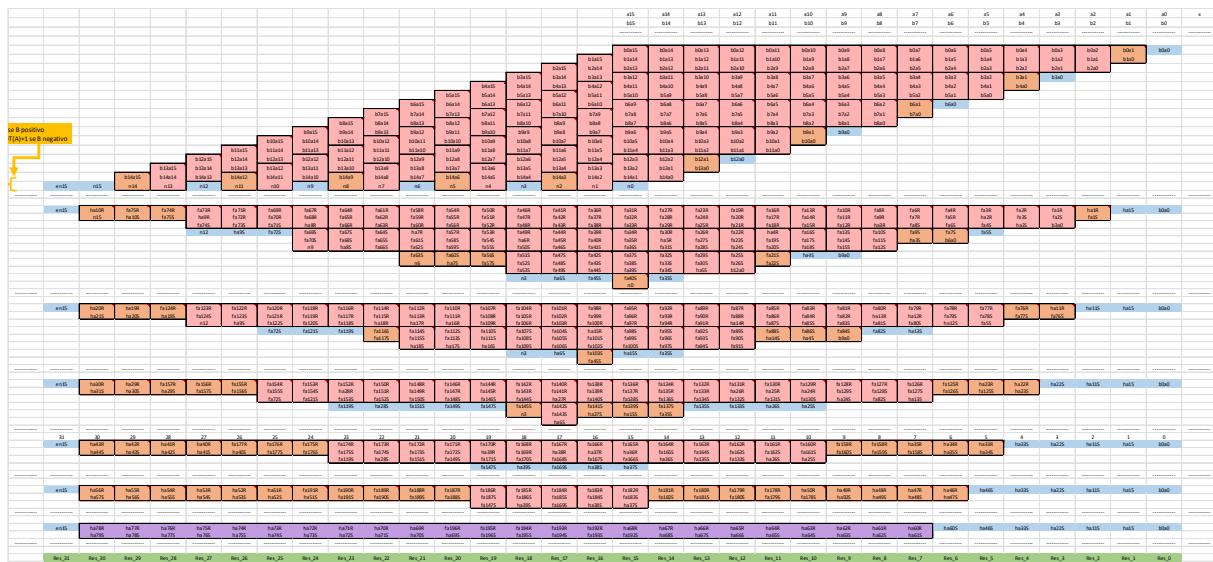


Figura 19 Struttura del Moltiplicatore di Wallace a 16 bit

Si può notare come partendo da un'altezza pari a 16, dopo 6 compressioni, il numero dei prodotti parziali è diminuito di ben 14 unità.

Effettuando il plot dell'andamento dell'altezza del moltiplicatore in questione, è possibile notare come, grazie alla presenza dei Full-Adder e degli Half-Adder, l'altezza sia diminuita notevolmente.



Figura 20 Andamento dell'altezza dell'albero nel Moltiplicatore di Wallace a 16 bit

Tanto è vero che, osservando la struttura progettata, si può notare come, ad esempio, nell'iterazione 1 in corrispondenza della colonna che contiene il maggior numero di bit sono presenti 5 Full-Adder, i quali avendo in input una terna ognuno, generano una coppia di output ciascuno facendo diminuire l'altezza di ben 5 unità.

5. Design Flow

Il flusso di progettazione per un circuito elettronico digitale su FPGA inizia con la descrizione del sistema e dei suoi moduli tramite un linguaggio di descrizione dell'hardware. Rappresentare questi moduli e, pertanto, l'intero circuito vuol dire delineare a livello logico le loro connessioni e le porte logiche di cui necessitano.

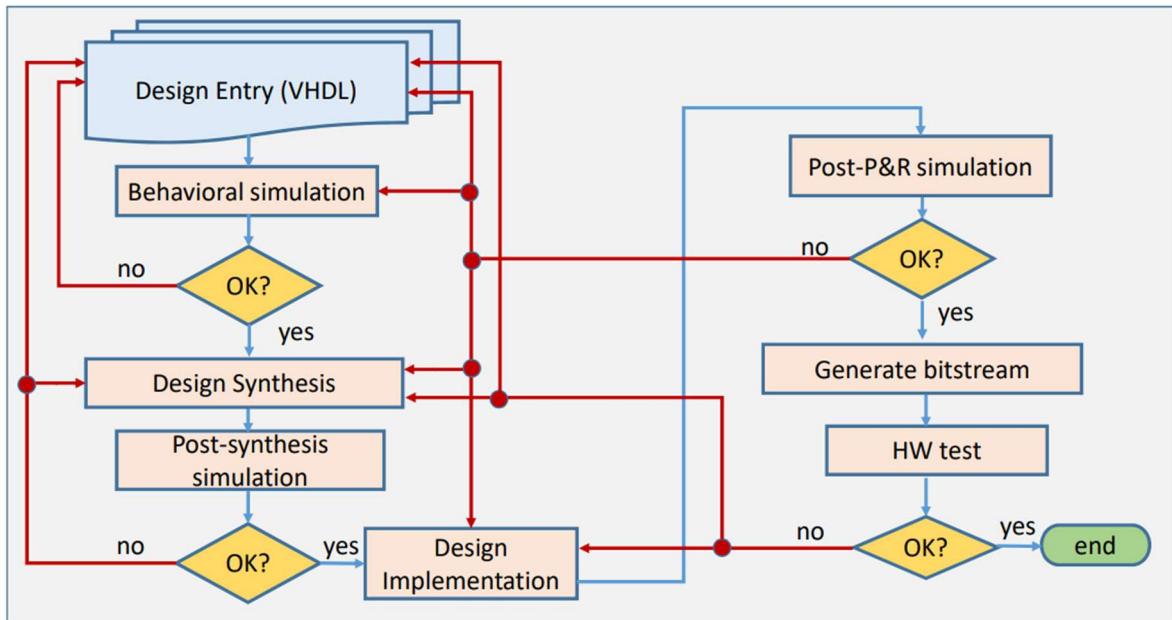


Figura 21 Design Flow

Per poter verificare se effettivamente il sistema rappresenta la funzione logica progettata, si effettua una Behavioral Simulation, cioè si analizza una simulazione relativa al comportamento che assume il circuito a fronte di determinati segnali. Se l'output del circuito non rispecchia la funzione logica desiderata, allora si esamina la struttura descritta in HDL. Se, invece, il modulo si comporta correttamente, allora si passa alla fase successiva: effettuare la sintesi del circuito. In questo caso il design RTL viene trasformato in una rappresentazione a livello di gate così da analizzare successivamente, attraverso la Post-Synthesis Simulation, il comportamento del sistema tenendo conto di delay approssimativi dei componenti che lo caratterizzano. Se la simulazione presenta imprecisioni o in casi peggiori errori non previsti, allora è il caso di esaminare nuovamente il design RTL progettato. Se, invece, la simulazione e i relativi ritardi introdotti rispecchiano le caratteristiche del sistema, allora si può procedere con l'implementazione del modulo progettato sul device di riferimento. Durante questa fase è previsto il posizionamento e l'instradamento della netlist sulle risorse del dispositivo FPGA. Ovviamente anche per l'implementazione, come

per le fasi precedenti, è prevista una simulazione ben specifica per verificare effettivamente se le proprietà del sistema progettato sono soddisfatte o meno. Pertanto, si analizza la Post-Implementation Simulation, la quale permette di verificare il comportamento del circuito elettronico digitale considerando i delay effettivi del device considerato e nel caso in cui non dovesse soddisfare le particolarità del sistema progettato, allora bisognerebbe esaminare nuovamente il design implementato come già sopra descritto. Se, invece, nella simulazione in questione le peculiarità del modulo sono soddisfatte, allora si potrebbe procedere con la generazione del bitstream e, successivamente, con il test a livello hardware sul dispositivo di riferimento.

In questo lavoro di tesi verranno analizzate tutte le fasi sopra citate per il Moltiplicatore “Carta e Penna” e per i Moltiplicatori di Wallace a 8 bit e 16 bit. Non verranno considerati, invece, la generazione del bitstream e il test a livello hardware.

5.1. RTL Schematic

La fase della generazione dell'RTL Schematic prevede, come già sopra citato, la rappresentazione a livello logico dei circuiti e delle relative connessioni. Ovviamente, ogni modulo sarà possibile ispezionarlo approfonditamente analizzando le porte logiche e i relativi collegamenti che compongono la funzione logica progettata.

5.1.1. Moltiplicatore "Carta e Penna"

Effettuando l'analisi RTL del Moltiplicatore "Carta e Penna" è possibile generare l'RTL Schematic e analizzare a livello logico la sua struttura progettata:

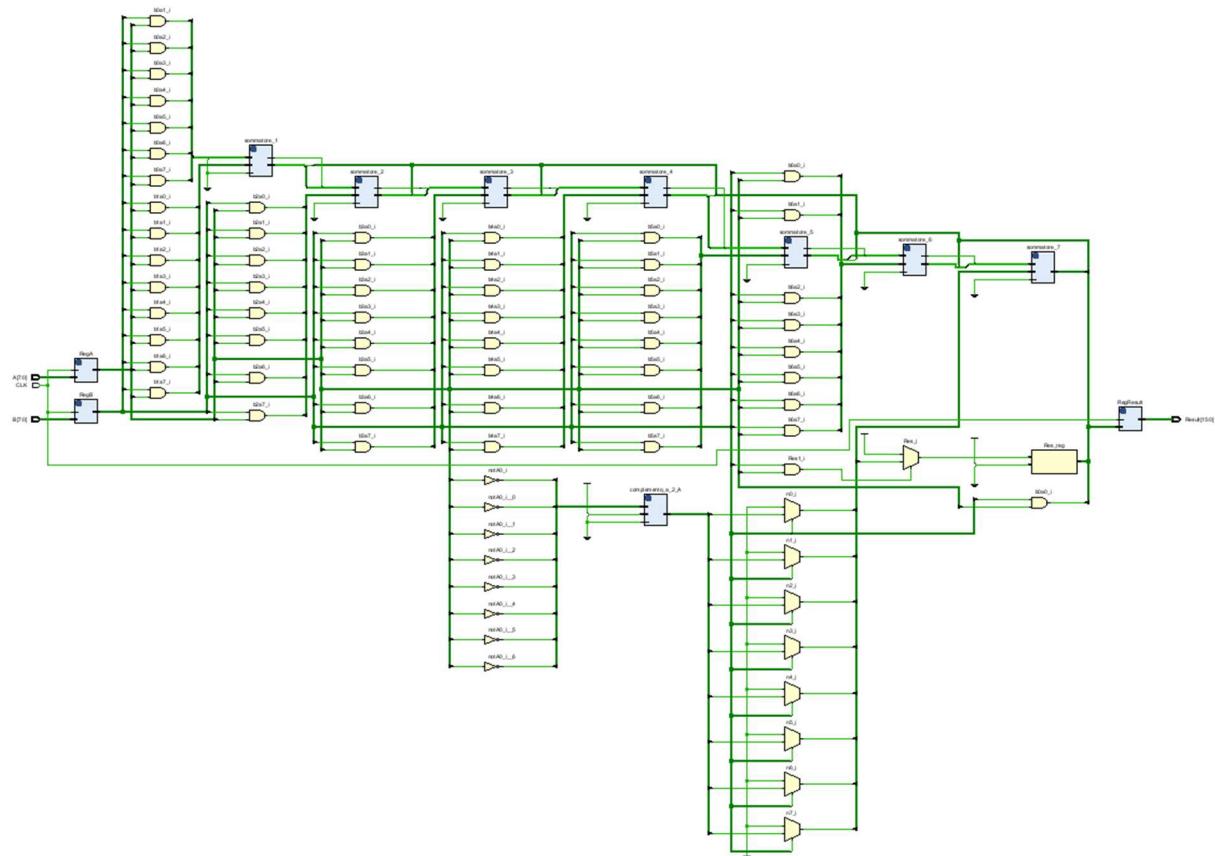


Figura 22 RTL Schematic del Moltiplicatore "Carta e Penna"

Inizialmente è possibile notare i due ingressi del sistema, il moltiplicando ed il moltiplicatore, che vengono messi in input ai due registri corrispondenti. Successivamente, tramite le porte logiche *and* vengono calcolati i prodotti parziali, i quali, attraverso le somme in cascata, genereranno il risultato finale. Ogni somma in cascata, come descritto nei capitoli precedenti, è calcolata tramite

l'ausilio di un Ripple-Carry Adder. Infine, il risultato ottenuto viene posto in un registro e, successivamente, collocato in output.

Si può effettuare uno zoom all'interno di ogni modulo progettato e inserito all'interno del sistema in questione:

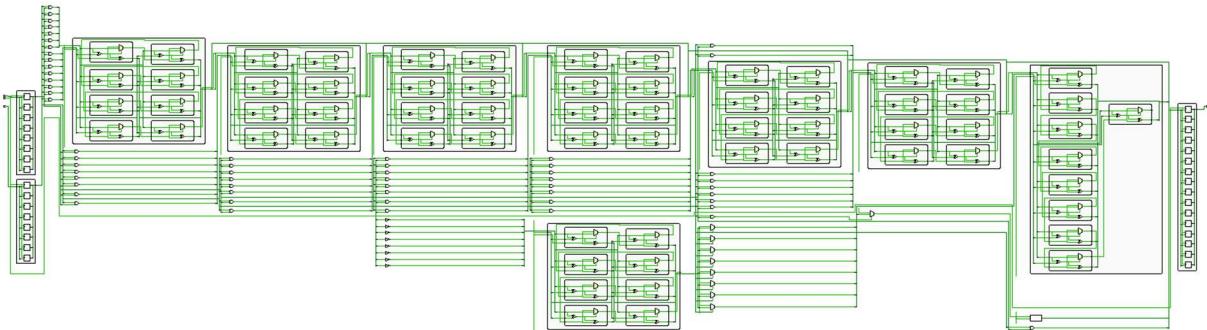


Figura 23 RTL Schematic del Moltiplicatore "Carta e Penna" con zoom all'interno dei moduli

È possibile osservare come ogni registro e Ripple-Carry Adder sia stato implementato come previsto. Nello specifico, è possibile notare come ogni RCA sia stato concretizzato tramite moduli Full-Adder come precedentemente citato.

5.1.2. Moltiplicatore di Wallace a 8 bit

Effettuando l'analisi RTL del Moltiplicatore di Wallace a 8 bit è possibile generare l'RTL Schematic e analizzare a livello logico la sua struttura progettata:

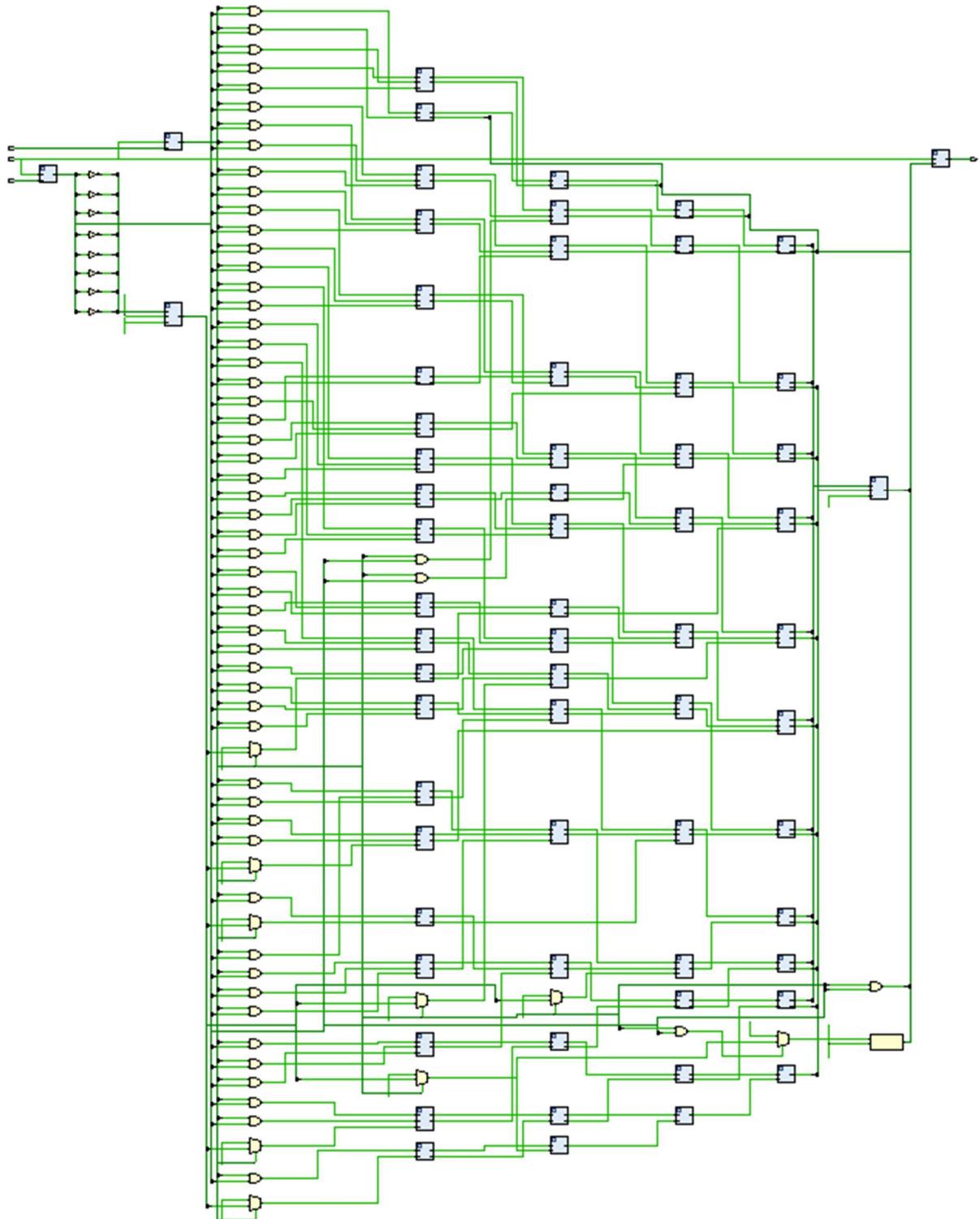


Figura 24 RTL Schematic del Moltiplicatore di Wallace a 8 bit

È possibile notare inizialmente i due ingressi del sistema, il moltiplicando ed il moltiplicatore, che vengono messi in input ai due registri corrispondenti. Successivamente, tramite le porte logiche *and* vengono calcolati i prodotti parziali, i quali, attraverso moduli Full-Adder ed Half-Adder, generano il risultato finale. Dall'RTL Schematic in questione, è possibile osservare come, ad ogni livello, il numero di circuiti FA e HA diminuisce gradualmente. Questo è dovuto, come spiegato nei capitoli precedenti, alla struttura che caratterizza il Moltiplicatore di Wallace. Inoltre, è possibile notare come la porta logica *and* corrispondente al prodotto tra il primo bit del moltiplicando ed il primo bit del moltiplicatore, venga direttamente posta, dalla suite, all'ultimo livello del moltiplicatore progettato. Questo perché, tale bit calcolato, viene direttamente propagato in output dal momento che la colonna corrispondente risulta essere di altezza pari a 1. Inoltre, è possibile osservare come sia la gestione del segno che la somma finale che caratterizza il Vector-Merging Adder siano implementati tramite l'ausilio di un Ripple-Carry Adder. Infine, il risultato ottenuto viene posto in un registro e, successivamente, collocato in output.

Si può effettuare uno zoom all'interno di ogni modulo progettato e inserito all'interno del sistema in questione:

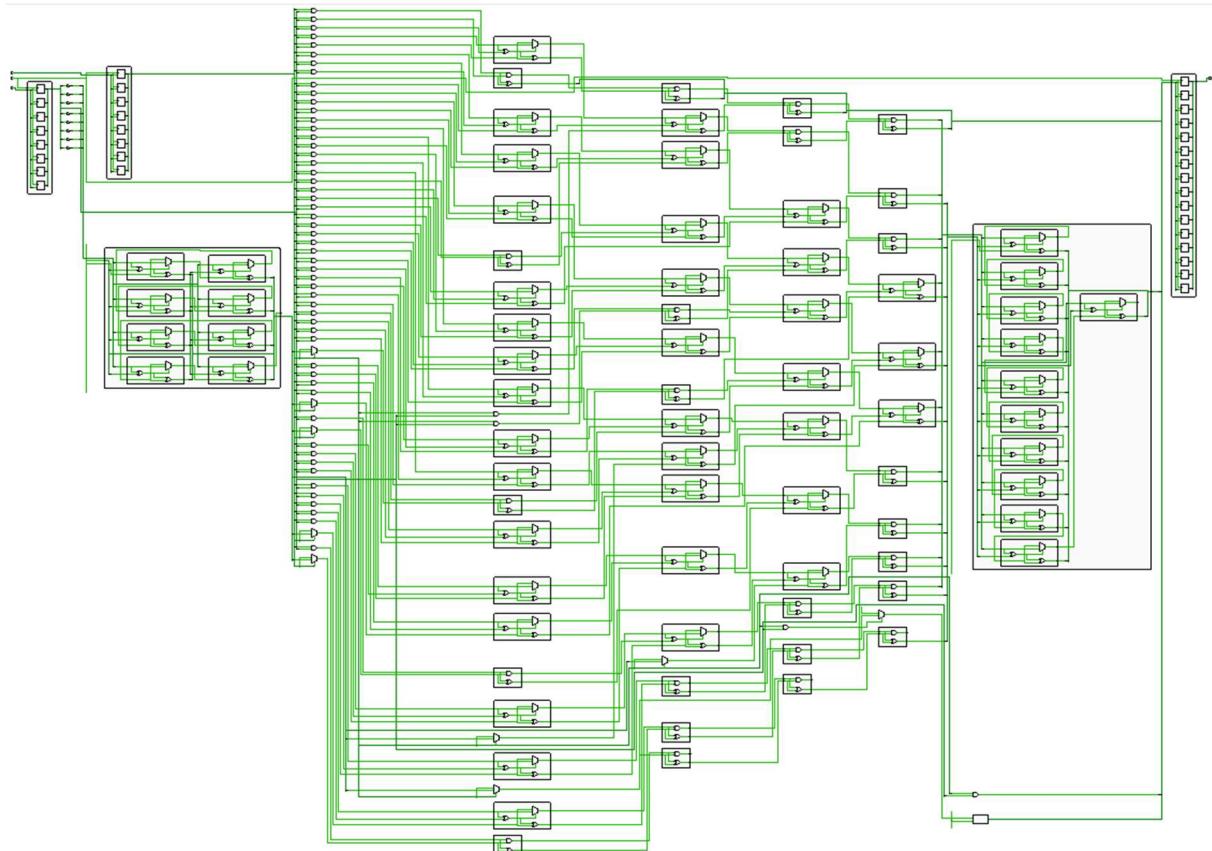


Figura 25 RTL Schematic del Moltiplicatore di Wallace a 8 bit con zoom all'interno dei moduli

È possibile osservare come i registri considerati presentano al loro interno i moduli per ogni bit in input ad essi. Inoltre, è possibile notare come ogni componente Full-Adder ed Half-Adder sia stato concretizzato correttamente.

5.1.3. Moltiplicatore di Wallace a 16 bit

Effettuando l'analisi RTL del Moltiplicatore di Wallace a 16 bit è possibile generare l'RTL Schematic e analizzare a livello logico la sua struttura progettata:

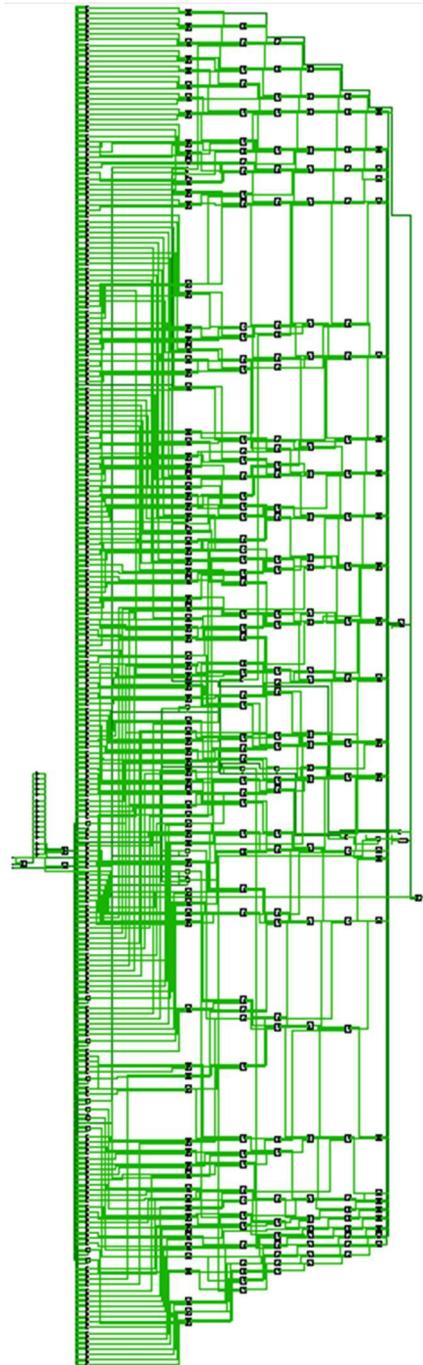


Figura 26 RTL Schematic del Moltiplicatore di Wallace a 16 bit

Inizialmente è possibile notare i due ingressi del sistema, il moltiplicando ed il moltiplicatore, che vengono messi in input ai due registri corrispondenti. Successivamente, tramite le porte logiche *and* vengono calcolati i prodotti parziali, i quali, attraverso moduli Full-Adder ed Half-Adder, generano il risultato finale. Anche in questo caso, come spiegato nel paragrafo precedente è possibile osservare come, ad ogni livello, il numero di circuiti FA e HA diminuisce gradualmente. A differenza della versione del moltiplicatore ad 8 bit, il numero delle risorse è notevolmente maggiore. Inoltre, è possibile osservare come sia la gestione del segno che la somma finale che caratterizza il Vector-Merging Adder siano implementati tramite l'ausilio di un Ripple-Carry Adder. Infine, il risultato ottenuto viene posto in un registro e, successivamente, collocato in output.

Si può effettuare uno zoom all'interno di ogni modulo progettato e inserito all'interno del sistema in questione:

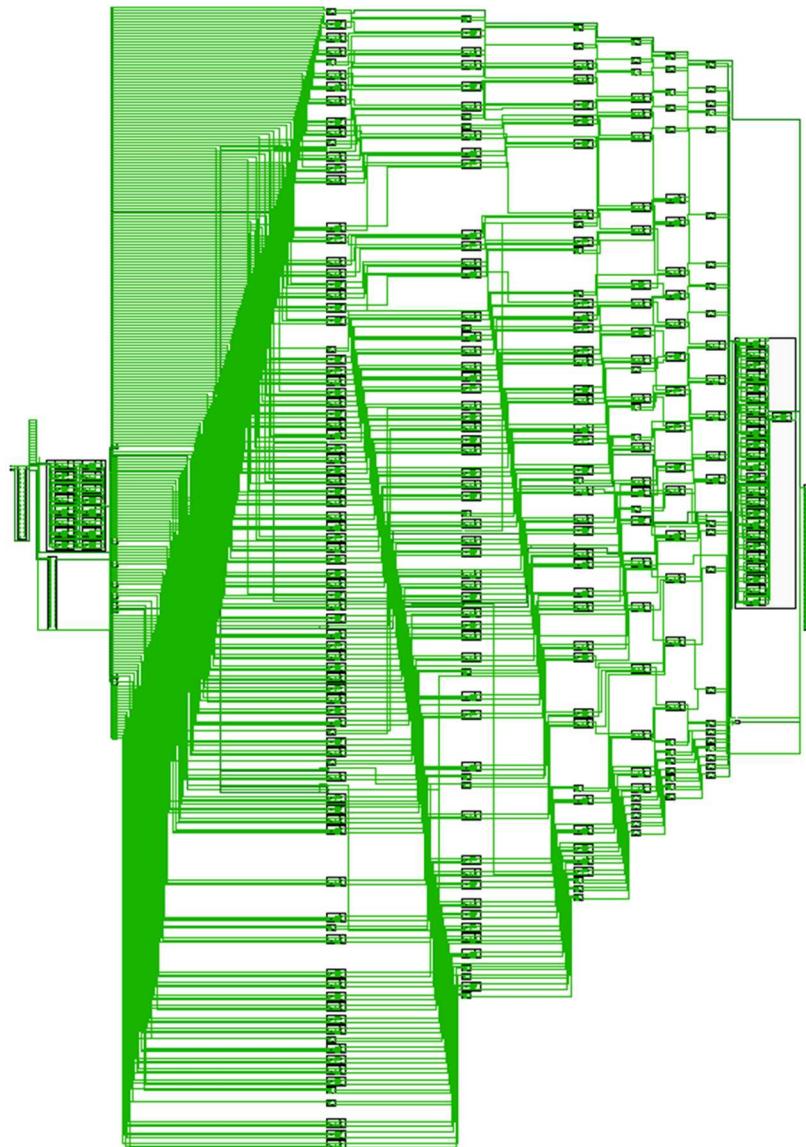


Figura 27 RTL Schematic del Moltiplicatore di Wallace a 16 bit con zoom all'interno dei moduli

È possibile osservare come i registri considerati presentano al loro interno 16 moduli ciascuno per ogni bit in input ad essi. Inoltre, è possibile notare come ogni componente Full-Adder ed Half-Adder sia stato concretizzato correttamente.

5.2. Sintesi

Il processo di sintesi, come già citato nei paragrafi precedenti, permette la trasformazione da modello RTL a rappresentazione basata a livello di gate. Nello specifico il codice scritto nel linguaggio descrittivo dell'hardware HDL viene compilato e tradotto in una netlist tramite il tool di sintesi presente all'interno della suite utilizzata.

5.2.1. Moltiplicatore “Carta e Penna”

Effettuando la sintesi del Moltiplicatore “Carta e Penna” è possibile generare lo schematico corrispondente e analizzarne la struttura sintetizzata generata:

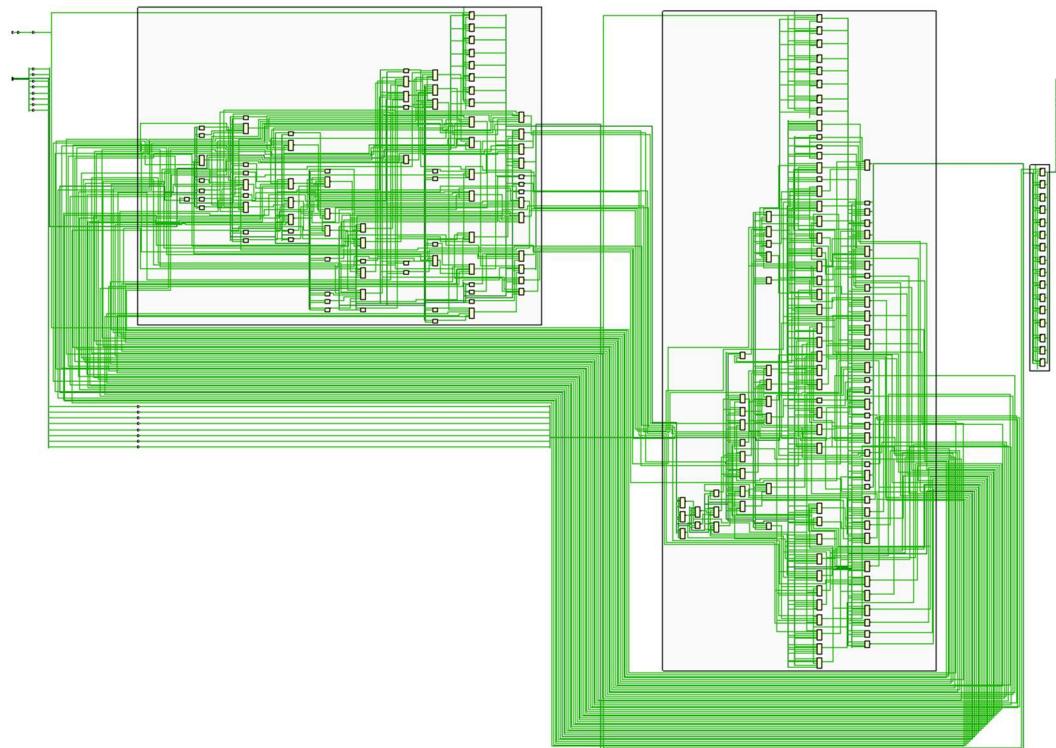


Figura 28 Synthesis Schematic del Moltiplicatore "Carta e Penna"

È possibile notare la presenza dei blocchi di tipologia FDRE che rappresentano un singolo Flip-Flop di tipologia D avente clock e reset sincrono. Questi moduli sono stati introdotti dal sintetizzatore a fronte dell'aggiunta dei registri all'interno

del circuito elettronico digitale. Inoltre, è possibile osservare la presenza di moduli LUT di varia tipologia. Ad esempio, sono presenti blocchetti a 2 input (LUT2) che permettono, quindi, la creazione di una qualche funzione logica a 2 ingressi. All'interno del design sintetizzato sono stati concretizzati anche moduli LUT a 3, 4, 5 e 6 input che permettono di istanziare una funzione logica rispettivamente a 3, 4, 5 e 6 ingressi. Infine, è possibile notare la presenza dei moduli IBUF, buffer di ingresso generico, che sono necessari per ogni pin di input del sistema. Ovviamente sono presenti anche gli OBUF, buffer di uscita generico, che sono necessari per ogni pin di uscita. Bisogna precisare che, essendo i 2 input del circuito elettronico digitale a 8 bit, saranno presenti 16 IBUF per i pin di ingresso al sistema e 16 FDRE per i 2 registri degli ingressi del sistema. Inoltre, saranno presenti 16 OBUF e 16 FDRE rispettivamente per i pin di uscita dell'unico output del circuito e per il registro corrispondente.

5.2.2. Moltiplicatore di Wallace a 8 bit

Effettuando la sintesi del Moltiplicatore di Wallace a 8 bit è possibile generare lo schematico corrispondente e analizzarne la struttura sintetizzata generata:

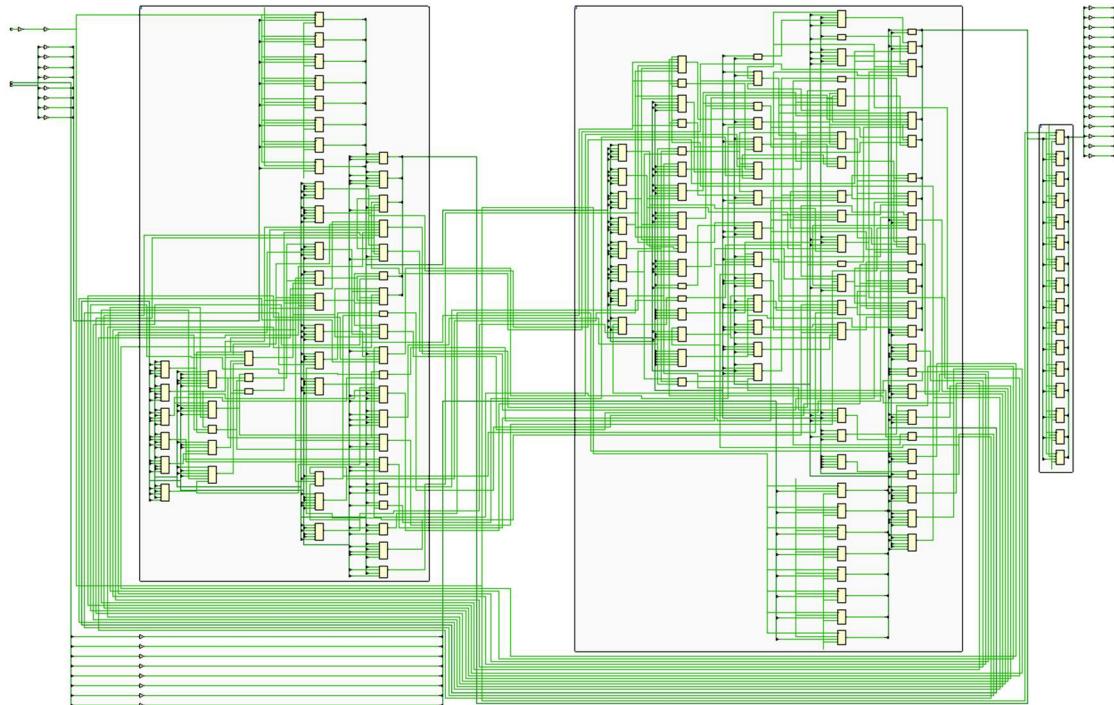


Figura 29 Synthesis Schematic del Moltiplicatore di Wallace a 8 bit

Anche in questo caso sono presenti LUT2, LUT3, LUT4, LUT5 e LUT6. Inoltre, è possibile notare la presenza dei moduli IBUF e OBUF. Bisogna precisare che, essendo i 2 input del circuito elettronico digitale a 8 bit, saranno presenti 16

IBUF per i pin di ingresso al sistema e 16 FDRE per i 2 registri corrispondenti. Invece, saranno presenti 16 OBUF e 16 FDRE rispettivamente per i pin di uscita dell'unico output del circuito e per il registro dell'uscita del sistema.

5.2.3. Moltiplicatore di Wallace a 16 bit

Effettuando la sintesi del Moltiplicatore di Wallace a 16 bit è possibile generare lo schematico corrispondente e analizzarne la struttura sintetizzata generata:

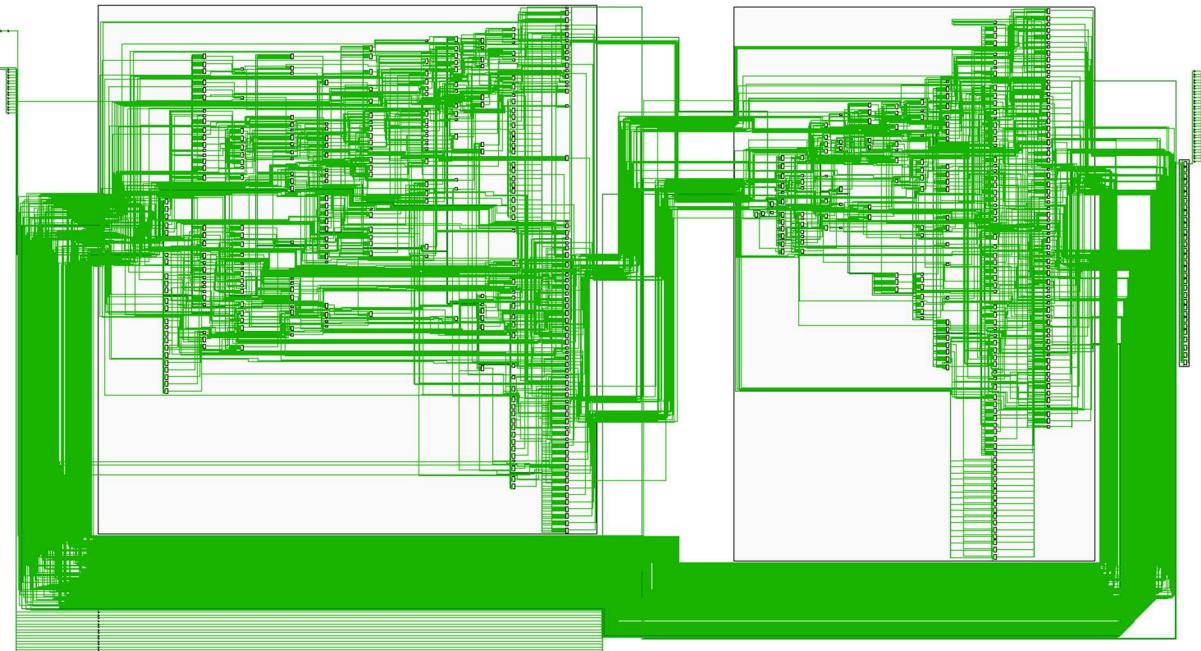


Figura 30 Synthesis Schematic del Moltiplicatore di Wallace a 16 bit

Si può osservare come, anche in questo caso, sono presenti moduli LUT a 2, 3, 4, 5 e 6 ingressi per la realizzazione delle funzioni logiche del sistema progettato. Inoltre, sono presenti moduli IBUF e OBUF per la gestione rispettivamente di input e output del circuito in questione. Ovviamente sono presenti anche i blocchi FDRE per la sintesi dei registri per i 2 ingressi e l'uscita del moltiplicatore.

Inoltre, si può notare come, rispetto allo schematico del Moltiplicatore di Wallace a 8 bit descritto nel paragrafo precedente, l'occupazione di area è estremamente maggiore. Questo è dovuto al fatto che i moduli Full-Adder ed Half-Adder, considerati nella struttura, sono superiori in quanto gli operandi considerati sono a 16 bit.

5.3. Constraint di clock

Il constraint di clock è una specifica di progetto, cioè un vincolo che si aggiunge nella gerarchia delle risorse della suite per sintetizzare il circuito in modo che lavori ad almeno una frequenza di funzionamento.

In generale, affinché i registri considerati e, pertanto, l'intero sistema funzionino correttamente, bisogna garantire che i vincoli temporali siano rispettati. Per la precisione questi vincoli possono essere identificati in termini di tempo di *setup* e tempo di *hold*. Il primo rappresenta l'anticipo minimo con cui il dato in ingresso ai Flip-Flop, presenti all'interno dei registri, può cambiare rispetto al prossimo fronte di clock attivo, cioè il tempo in cui preventivamente l'ingresso dei FF deve diventare stabile. Il tempo di *hold*, invece, rappresenta il ritardo minimo con cui il dato in ingresso ai Flip-Flop può cambiare rispetto all'ultimo fronte di clock attivo, cioè il minimo tempo tra il fronte di clock e la successiva variazione dell'ingresso. Questi vincoli temporali sono fondamentali affinché i valori di tensione considerati siano quelli nominali, 0 V o V_{CC} , anziché valori non stabili, cioè intermedi tra quelli stabili appena citati. Il problema appena descritto è di notevole importanza poiché nel caso in cui si considerino valori di tensione diversi da quelli nominali e, pertanto, considerando valori intermedi, potrebbero essere generati stati di indeterminazione nell'output del sistema. Pertanto, per far fronte a questo problema, viene stabilito un periodo di clock, cioè la cadenza con cui vengono fatti arrivare i fronti di salita del clock al circuito progettato. Affinché il periodo di clock rispecchi le specifiche appena descritte, esso viene definito nella seguente maniera:

$$T_{clock} = t_{max} + t_{setup}$$

dove T_{clock} indica il periodo di clock che verrà imposto al sistema, t_{max} è il periodo corrispondente alla massima frequenza di funzionamento del circuito progettato e t_{setup} è il vincolo temporale di *setup* che è reso disponibile, insieme al tempo di *hold*, all'interno dei *datasheet*⁴⁴. Ovviamente i vincoli temporali sono definiti affinché rispettino la seguente relazione:

$$t_{setup} \gg t_{hold}$$

Pertanto, il periodo massimo che può essere garantito da un circuito è pari a:

$$t_{max} = T_{clock} - t_{setup}$$

⁴⁴ Il *datasheet* è la documentazione relativa ad un componente elettronico o meccanico che riassume le caratteristiche principali di quest'ultimo tra cui le sue proprietà fisiche, chimiche e modalità di funzionamento.

tale che la frequenza massima a cui il sistema può funzionare è pari a:

$$f_{max} = \frac{1}{t_{max}}$$

Nello specifico è stato imposto al sistema un periodo di clock pari a 20 ns :

Name	Waveform	Period (ns)	Frequency (MHz)
MyCLK	{0.000 10.000}	20.000	50.000

Figura 31 Constraint di clock

Pertanto, la frequenza di funzionamento garantita, secondo il constraint di clock considerato, è:

$$f_{garantita} = \frac{1}{T_{constraint}} = \frac{1}{20\text{ ns}} = \frac{1}{20 \times 10^{-9}\text{ s}} = 50000000\text{ Hz} = 0.05\text{ GHz}$$

Quindi, le frequenze di funzionamento dei circuiti progettati dovranno essere almeno pari a 0.05 GHz .

Per quanto riguarda il calcolo dei tempi di *setup* e di *hold* e della frequenza massima di funzionamento, essi verranno calcolati e commentati, per ogni implementazione eseguita, nei paragrafi successivi. In particolare, verranno esaminati il WNS⁴⁵, TNS⁴⁶, il WHS⁴⁷ ed il THS⁴⁸.

5.4. Implementazione

Dopo che il sistema è stato sintetizzato, si procede con l'implementazione del circuito sul dispositivo di riferimento. L'implementazione, come già citato nei paragrafi precedenti, prevede il posizionamento e l'instradamento della netlist sulle risorse del dispositivo FPGA. Nello specifico, il processo mappa il circuito sintetizzato sul chip FPGA.

⁴⁵ WNS è l'acronimo di Worst Negative Slack. Esso corrisponde al peggior slack di tutti i percorsi di temporizzazione per l'analisi del ritardo massimo. Tale *slack* può essere sia positivo che negativo.

⁴⁶ TNS è l'acronimo di Total Negative Slack. Esso corrisponde alla somma di tutte le violazioni WNS, se si considera solo la peggiore violazione di ciascun *endpoint* del percorso di temporizzazione. Nello specifico, il suo valore può essere 0 ns quando tutti i vincoli di temporizzazione sono soddisfatti per l'analisi del ritardo massimo oppure negativo quando sono presenti delle violazioni.

⁴⁷ WHS è l'acronimo di Worst Hold Slack. Esso corrisponde al peggior slack di tutti i percorsi di temporizzazione per l'analisi del ritardo minimo. Tale *slack* può essere sia positivo che negativo.

⁴⁸ THS è l'acronimo di Total Hold Slack. Esso corrisponde alla somma di tutte le violazioni WHS, se si considera solo la peggiore violazione di ciascun *endpoint* del percorso temporale. Nello specifico, il suo valore può essere 0 ns quando tutti i vincoli temporali sono soddisfatti per l'analisi del ritardo minimo oppure negativo quando sono presenti delle violazioni.

5.4.1. Moltiplicatore “Carta e Penna”

Effettuando l’implementazione del Moltiplicatore “Carta e Penna” è possibile generare lo schematico corrispondente e analizzarne la struttura sintetizzata generata:

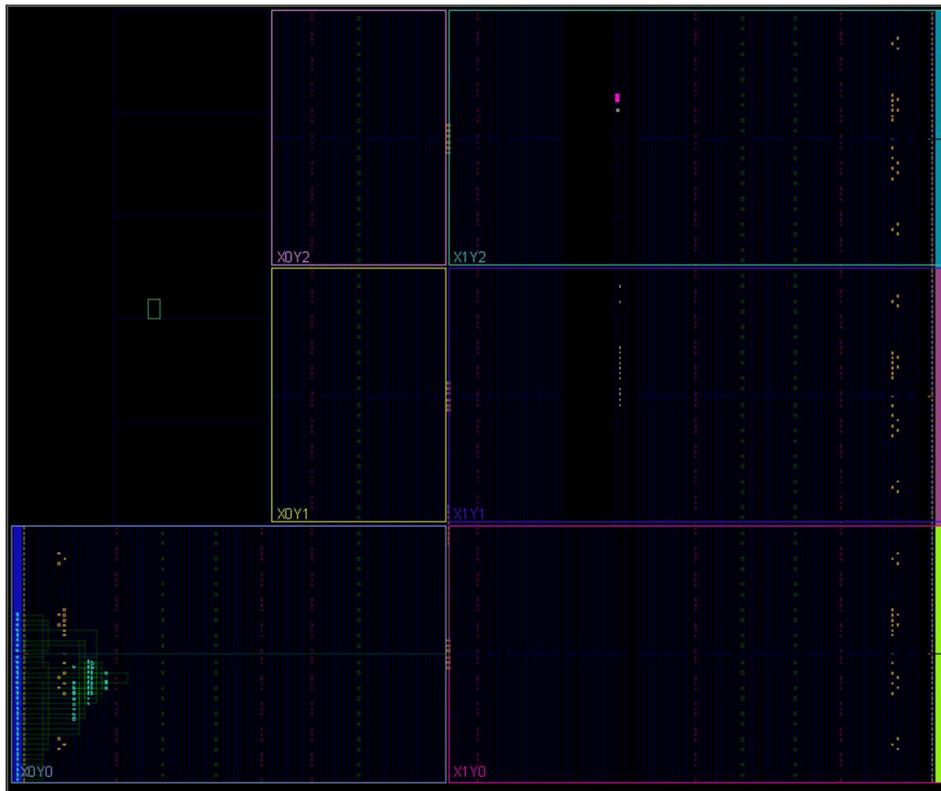


Figura 32 Implementazione del Moltiplicatore "Carta e Penna" su device

È possibile notare come il circuito sintetizzato relativo al Moltiplicatore “Carta e Penna” sia stato implementato su device.

Per quanto riguarda i vincoli temporali, sono stati generati i seguenti tempi di *setup* e *hold*:

Setup	Hold
Worst Negative Slack (WNS): 9.294 ns	Worst Hold Slack (WHS): 0.260 ns
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns

Figura 33 Design Timing Summary del Moltiplicatore "Carta e Penna"

Si può evincere come la relazione tra t_{setup} e t_{hold} sia garantita. Infatti:

$$t_{setup} = 9.294 \text{ ns} \gg t_{hold} = 0.260 \text{ ns}$$

Inoltre, si può notare come il periodo corrispondente alla massima frequenza di funzionamento è pari a:

$$t_{max} = T_{clock} - t_{setup} = 20 \text{ ns} - 9.294 \text{ ns} = 10.706 \text{ ns}$$

Pertanto, la massima frequenza a cui il dispositivo in questione può funzionare è pari a:

$$\begin{aligned} f_{max} &= \frac{1}{t_{max}} = \frac{1}{10.706 \text{ ns}} = \frac{1}{10.706 \times 10^{-9} \text{ s}} = 93405566.97 \text{ Hz} \\ &= 0.09340556697 \text{ GHz} \end{aligned}$$

Si può osservare come la frequenza massima di funzionamento, ottenuta per un circuito di tipologia Moltiplicatore “Carta e Penna” ad 8 bit, implementato su device, è maggiore della frequenza di funzionamento garantita dal constraint di clock considerato.

5.4.2. Moltiplicatore di Wallace a 8 bit

Effettuando l’implementazione del Moltiplicatore di Wallace a 8 bit è possibile generare lo schematico corrispondente e analizzarne la struttura sintetizzata generata:

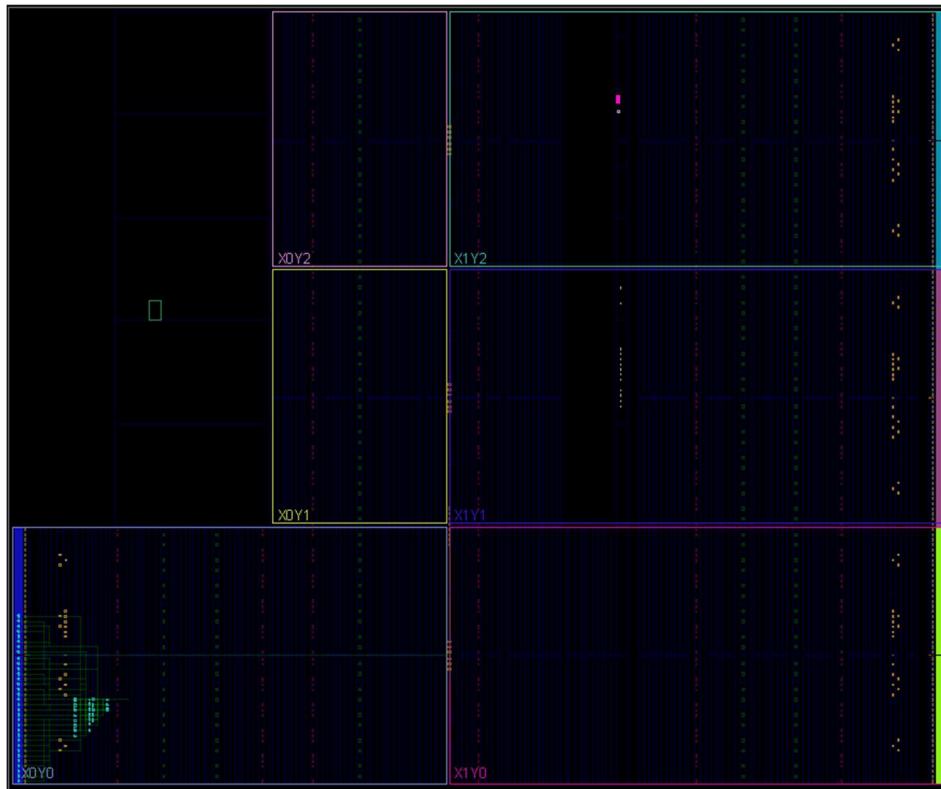


Figura 34 Implementazione del Moltiplicatore di Wallace a 8 bit su device

È possibile notare come il circuito sintetizzato relativo al Moltiplicatore di Wallace a 8 bit sia stato implementato su device.

Per quanto riguarda i vincoli temporali, sono stati generati i seguenti tempi di *setup* e *hold*:

Setup	Hold
Worst Negative Slack (WNS): 12.414 ns	Worst Hold Slack (WHS): 0.431 ns
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns

Figura 35 Design Timing Summary del Moltiplicatore di Wallace a 8 bit

Si può evincere come la relazione tra t_{setup} e t_{hold} sia garantita. Infatti:

$$t_{setup} = 12.414 \text{ ns} \gg t_{hold} = 0.431 \text{ ns}$$

Inoltre, si può notare come il periodo corrispondente alla massima frequenza di funzionamento è pari a:

$$t_{max} = T_{clock} - t_{setup} = 20 \text{ ns} - 12.414 \text{ ns} = 7.586 \text{ ns}$$

Pertanto, la massima frequenza a cui il dispositivo in questione può funzionare è pari a:

$$f_{max} = \frac{1}{t_{max}} = \frac{1}{7.586 \text{ ns}} = \frac{1}{7.586 \times 10^{-9} \text{ s}} = 131821777 \text{ Hz} = 0.131821777 \text{ GHz}$$

Si può osservare come la frequenza massima di funzionamento, ottenuta per un circuito di tipologia Moltiplicatore di Wallace ad 8 bit, implementato su device, è maggiore della frequenza di funzionamento garantita dal constraint di clock considerato.

5.4.3. Moltiplicatore di Wallace a 16 bit

Effettuando l'implementazione del Moltiplicatore di Wallace a 16 bit è possibile generare lo schematico corrispondente e analizzarne la struttura sintetizzata generata:

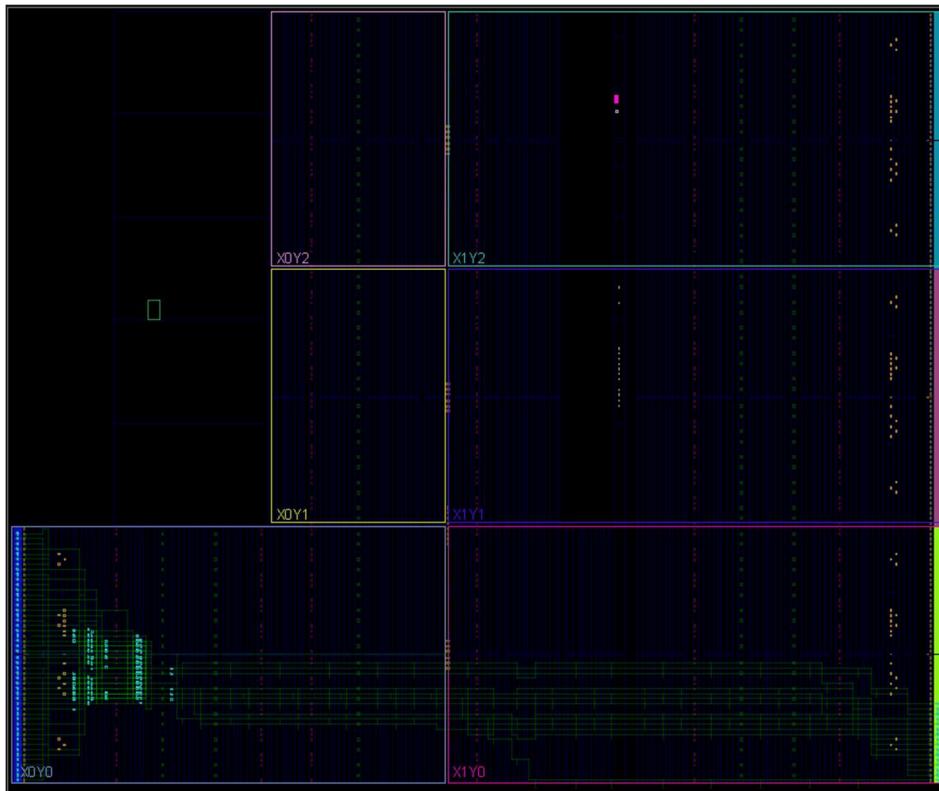


Figura 36 Implementazione del Moltiplicatore di Wallace a 16 bit su device

È possibile notare come il circuito sintetizzato relativo al Moltiplicatore di Wallace a 16 bit sia stato implementato su device. Inoltre, si può osservare come l'occupazione di area è maggiore di quello descritto nel precedente paragrafo. Questo perché il numero dei bit degli operandi è 16 e, pertanto, come già descritto nei capitoli precedenti, il numero delle risorse utilizzate, in termini di Full-Adder e Half-Adder, è superiore rispetto al circuito ad 8 bit.

Per quanto riguarda i vincoli temporali, sono stati generati i seguenti tempi di *setup* e *hold*:

Setup	Hold
Worst Negative Slack (WNS): 7.650 ns	Worst Hold Slack (WHS): 0.253 ns
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns

Figura 37 Design Timing Summary del Moltiplicatore di Wallace a 16 bit

Si può evincere come la relazione tra t_{setup} e t_{hold} sia garantita. Infatti:

$$t_{setup} = 7.650 \text{ ns} \gg t_{hold} = 0.253 \text{ ns}$$

Inoltre, si può notare come il periodo corrispondente alla massima frequenza di funzionamento è pari a:

$$t_{max} = T_{clock} - t_{setup} = 20 \text{ ns} - 7.650 \text{ ns} = 12.35 \text{ ns}$$

Pertanto, la massima frequenza a cui il dispositivo in questione può funzionare è pari a:

$$f_{max} = \frac{1}{t_{max}} = \frac{1}{12.35 \text{ ns}} = \frac{1}{12.35 \times 10^{-9} \text{ s}} = 80971659.92 \text{ Hz} = 0.08097165992 \text{ GHz}$$

Si può osservare come la frequenza massima di funzionamento, ottenuta per un circuito di tipologia Moltiplicatore di Wallace ad 16 bit, implementato su device, è maggiore della frequenza di funzionamento garantita dal constraint di clock considerato.

5.5. Simulazione

La simulazione è una fase molto importante all'interno del design flow di progettazione perché permette di esaminare come si comporta il circuito progettato a fronte di determinati input di riferimento.

Nella suite considerata è presente una GUI che permette di analizzare approfonditamente il comportamento del sistema attraverso una finestra *Waveform*⁴⁹. Questo strumento permette allo sviluppatore di effettuare un vero e proprio debug del circuito descritto nelle fasi precedenti di progettazione. All'interno di questa finestra è possibile visualizzare i segnali di ingresso al sistema e le relative uscite. Pertanto, si può esaminare l'andamento nel tempo di questi segnali. Inoltre, è possibile cambiare il formato di visualizzazione dei *signals* scegliendo tra varie tipologie: binario, decimale, esadecimale e altri ancora. Un aspetto fondamentale della simulazione in Vivado è la possibilità di condurre una simulazione per un periodo di tempo definito dallo sviluppatore: dai pochi picosecondi fino anche a qualche secondo. Ovviamente condurre una simulazione equivalente ad un periodo di tempo pari ad un secondo richiede una quantità di calcolo non indifferente.

⁴⁹ La finestra di tipologia Waveform è una rappresentazione, permessa dalla suite di riferimento, che illustra l'andamento dei segnali nel tempo.

Per effettuare una simulazione è necessario scrivere un file di testbench, cioè un file in cui viene istanziato il componente da simulare e dove vengono definiti i segnali in ingresso al circuito così da visualizzare, successivamente nella finestra Waveform, l'andamento delle sue uscite.

La Vivado Design Suite permette di effettuare diverse tipologie di simulazione. Per ogni fase progettuale è associata una simulazione ben specifica. Dopo aver descritto il sistema tramite il linguaggio di descrizione dell'hardware, è possibile verificare il comportamento del circuito tramite una Behavioral Simulation. Successivamente, dopo aver effettuato la sintesi e l'implementazione del circuito, è possibile analizzare il comportamento ed il ritardo del circuito tramite rispettivamente la Post-Synthesis Timing Simulation e la Post-Implementation Timing Simulation che permettono di analizzare l'andamento degli output generati considerando i delay introdotti dal dispositivo. Bisogna precisare che, dopo aver effettuato la sintesi e l'implementazione del circuito su device, è possibile analizzare due ulteriori simulazioni: la Post-Synthesis Functional Simulation e la Post-Implementation Functional Simulation. Quest'ultime appena citate permettono di analizzare il comportamento del sistema dopo aver sintetizzato ed implementato quest'ultimo sul dispositivo di riferimento.

Durante questo progetto di tesi sono stati considerati due ingressi per ogni circuito: il primo *unsigned* a n bit ed il secondo *signed* a m bit. Pertanto, l'output generato dal sistema è di tipologia *signed* a $n + m$ bit. Inoltre, bisogna precisare che l'approccio utilizzato è quello di una simulazione esaustiva, cioè un'analisi che prevede la verifica del comportamento del circuito a fronte di tutti i possibili ingressi che possono essere utilizzati per ogni sistema. Pertanto, tramite una doppia procedura di loop, sono stati considerati tutti i valori fattibili che potrebbero essere considerati come input validi per ogni circuito progettato. Infatti, bisogna precisare che per un valore *unsigned* ad n bit è stato considerato il seguente range di valori:

$$0 \div 2^n - 1$$

mentre per un valore *signed* ad n bit è stato considerato il seguente intervallo di valori:

$$-2^{n-1} \div 2^{n-1} - 1$$

Inoltre, è stato preso in considerazione un certo ritardo iniziale per puntualizzare il fatto che inizialmente il circuito si trova in uno stato in cui gli operandi risultano

ancora non definiti, cioè *uninitialized*. Pertanto, è stato considerato un delay iniziale pari a

$$2 \cdot T_{CLK} + 100\text{ ns}$$

dove T_{CLK} è il periodo di clock considerato ed è pari a 20 ns , cioè pari al constraint di clock precedentemente descritto. Quindi, il ritardo iniziale sarà pari a 140 ns .

Inoltre, è stata garantita una variazione del segnale di clock ogni $\frac{T_{CLK}}{2}$, cioè ogni 10 ns , così da garantire la verifica di ogni output del sistema. Bisogna ricordare che, l'uscita del circuito è processata in output ogni volta che viene a verificarsi un fronte di salita del segnale di clock. Infatti, questo sincronismo è garantito dai registri che scandiscono l'ingresso dei valori nel sistema e processano in uscita il valore ottenuto a fronte dei calcoli effettuati dal circuito.

In aggiunta, sono stati considerati due segnali ulteriori a fini di debugging⁵⁰ del sistema: *TrueResult* ed *Error*. Il primo identifica il vero valore che dovrebbe assumere l'uscita del circuito considerato, mentre il secondo indica la differenza tra il valore vero e quello assunto dal sistema.

⁵⁰ Il debugging, nell'ambito informatico e dello sviluppo software, indica l'attività che consiste nell'individuazione e nella correzione di uno o più bug (errori) rilevati all'interno del software.

5.5.1. Behavioral Simulation

La Behavioral Simulation, come già descritto nel precedente paragrafo, è una simulazione che permette di analizzare il comportamento del circuito senza tener conto di delay relativi alla logica e al dispositivo considerato. Tanto è vero che è definita con il termine di simulazione comportamentale che identifica proprio il fine ultimo di questa analisi.

5.5.1.1. Moltiplicatore “Carta e Penna”

Il moltiplicatore seriale “Carta e Penna” progettato presenta un moltiplicando ad 8 bit *unsigned* ed un moltiplicatore ad 8 bit *signed*. Pertanto, l’output del sistema è di tipologia *signed* a 16 bit.

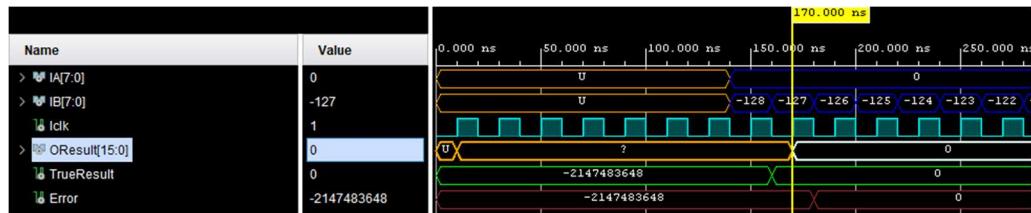


Figura 38 Behavioral Simulation del Moltiplicatore "Carta e Penna" (1)

È possibile osservare come il primo output generato del sistema si ottiene dopo 170 ns di funzionamento. Questo perché, come già descritto nel paragrafo precedente, sono stati considerati 140 ns di ritardo iniziale e, inoltre, ulteriori 20 ns per ogni uscita generata. Inoltre, devono essere considerati ulteriori 10 ns poiché in corrispondenza dei 160 ns appena descritti, è possibile osservare la presenza di un fronte di discesa del segnale di clock. Pertanto, affinché venga processata un’uscita dal sistema, il circuito attende ulteriori $T_{CLK}\text{ ns}$ affinché si verifichi un nuovo fronte di salita del segnale di clock.

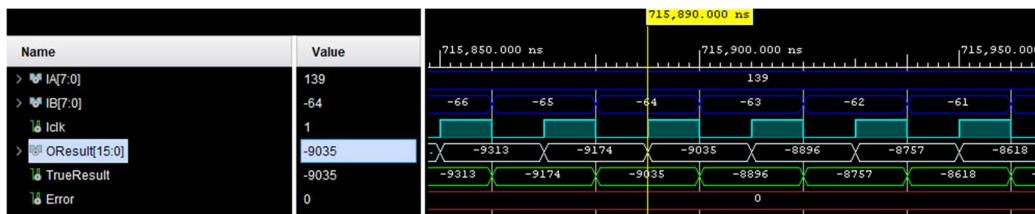


Figura 39 Behavioral Simulation del Moltiplicatore "Carta e Penna" (2)

È possibile notare come, in corrispondenza di un moltiplicando ad 8 bit *unsigned* pari a 139 e di un moltiplicatore ad 8 bit *signed* pari a -65 , l’output generato è corretto, cioè pari a -9035 . Tanto è vero che, l’errore calcolato è pari a 0. Inoltre,

è possibile osservare come l'uscita del sistema è stata generata successivamente al calcolo effettuato dal segnale di debugging. Questo perché l'output del circuito è stato processato in corrispondenza del fronte di salita del segnale di clock disponibile in corrispondenza di $715,890.000\text{ ns}$.

5.5.1.2. Moltiplicatore di Wallace a 8 bit

Il moltiplicatore di Wallace a 8 bit progettato presenta un moltiplicando ad 8 bit *unsigned* ed un moltiplicatore ad 8 bit *signed*. Pertanto, l'output del sistema è di tipologia *signed* a 16 bit.

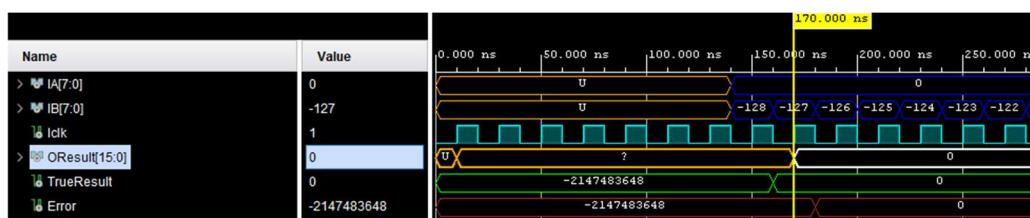


Figura 40 Behavioral Simulation del Moltiplicatore di Wallace a 8 bit (1)

È possibile osservare come, anche in questo caso, il primo output generato del sistema si ottiene dopo 170 ns di funzionamento. Inoltre, analizzando un ulteriore finestra temporale della simulazione, è possibile notare come il circuito progettato genera correttamente gli output previsti:



Figura 41 Behavioral Simulation del Moltiplicatore di Wallace a 8 bit (2)

È possibile notare come, in corrispondenza di un moltiplicando ad 8 bit *unsigned* pari a 23 e di un moltiplicatore ad 8 bit *signed* pari a -93 , l'output generato è corretto, cioè pari a -2139 . Tanto è vero che, l'errore calcolato è pari a 0. Inoltre, è possibile osservare come l'uscita del sistema è generata successivamente al calcolo effettuato dal segnale di debugging. Questo perché l'output del circuito è stato processato in corrispondenza del fronte di salita del segnale di clock disponibile in corrispondenza di $1,435,070.000\text{ ns}$.

5.5.1.3. Moltiplicatore di Wallace a 16 bit

Il moltiplicatore di Wallace a 16 bit progettato presenta un moltiplicando ad 16 bit *unsigned* ed un moltiplicatore ad 16 bit *signed*. Pertanto, l'output del sistema è di tipologia *signed* a 32 bit.

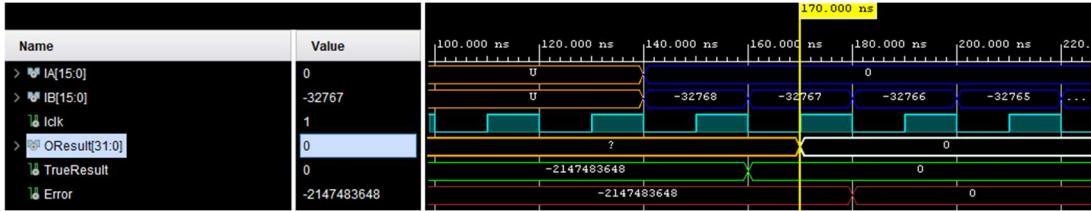


Figura 42 Behavioral Simulation del Moltiplicatore di Wallace a 16 bit (1)

È possibile osservare come, anche in questo caso, il primo output generato del sistema si ottiene dopo 170 ns di funzionamento. Inoltre, analizzando un ulteriore finestra temporale della simulazione, è possibile notare come il circuito progettato genera correttamente gli output previsti:

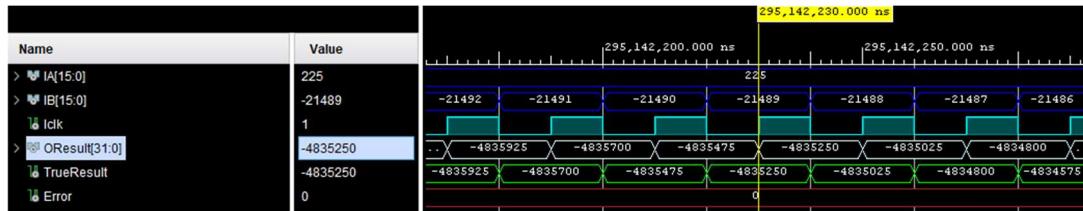


Figura 43 Behavioral Simulation del Moltiplicatore di Wallace a 16 bit (2)

È possibile notare come, in corrispondenza di un moltiplicando ad 16 bit *unsigned* pari a 225 e di un moltiplicatore ad 16 bit *signed* pari a -21490 , l'output generato è corretto, cioè pari a -4835250 . Tanto è vero che, l'errore calcolato è pari a 0. Inoltre, è possibile osservare come l'uscita del sistema è generata successivamente al calcolo effettuato dal segnale di debugging. Questo perché l'output del circuito è stato processato in corrispondenza del fronte di salita del segnale di clock disponibile in corrispondenza di $295,142,230.000\text{ ns}$.

5.5.2. Post-Synthesis Timing Simulation

La Post-Synthesis Simulation, come già citato nei capitoli precedenti, permette di effettuare un'analisi del comportamento tenendo conto di delay del sistema approssimativi. Bisogna ricordare che, essa è possibile analizzarla solo dopo aver sintetizzato il circuito progettato.

5.5.2.1. Moltiplicatore “Carta e Penna”

Considerando il moltiplicatore seriale “Carta e Penna” progettato, avente un moltiplicando ad 8 bit *unsigned* ed un moltiplicatore ad 8 bit *signed*, l’output generato dal sistema in questione sarà di tipologia *signed* a 16 bit.

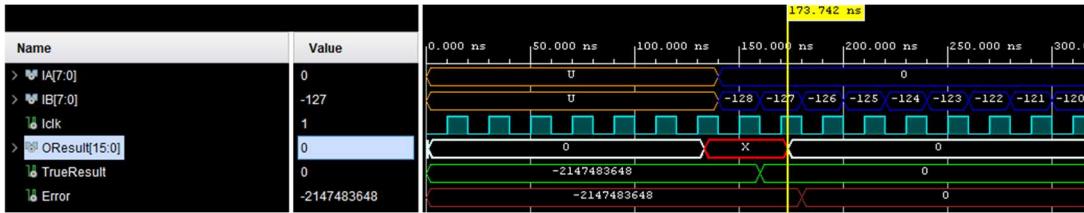


Figura 44 Post-Synthesis Timing Simulation del Moltiplicatore "Carta e Penna"

È possibile notare che, rispetto alla Behavioral Simulation del circuito, analizzata nei paragrafi precedenti, la Post-Synthesis Timing Simulation sta considerando i ritardi approssimativi riguardo la sintesi effettuata dalla suite. Tanto è vero che, il primo risultato idoneo si ha in corrispondenza di 173.742 ns che risulta essere, quindi, maggiore rispetto al risultato ottenuto nella simulazione comportamentale in corrispondenza di 170 ns . Infatti, si può notare che questo delay viene introdotto a causa dello stato logico *unknown* dell’uscita in quanto il circuito sintetizzato sta ancora elaborando gli ingressi del sistema.

5.5.2.2. Moltiplicatore di Wallace a 8 bit

Il moltiplicatore di Wallace a 8 bit progettato presenta un moltiplicando ad 8 bit *unsigned* ed un moltiplicatore ad 8 bit *signed*. Pertanto, l'output del sistema è di tipologia *signed* a 16 bit.

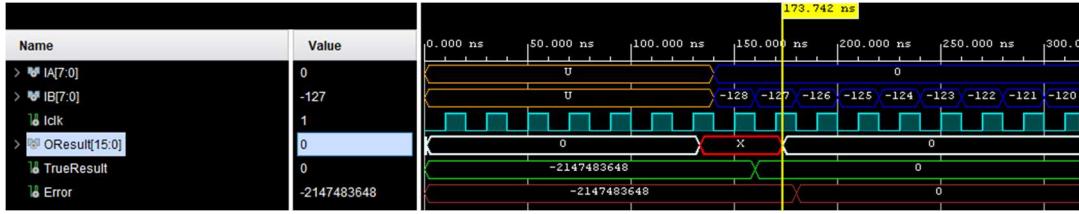


Figura 45 Post-Synthesis Timing Simulation del Moltiplicatore di Wallace a 8 bit

È possibile notare che, rispetto alla Behavioral Simulation del circuito, analizzata nei paragrafi precedenti, la Post-Synthesis Timing Simulation sta considerando i ritardi approssimativi riguardo la sintesi effettuata dalla suite. Tanto è vero che, il primo risultato idoneo si ha in corrispondenza di 173.742 ns che risulta essere, quindi, maggiore rispetto al risultato ottenuto nella simulazione comportamentale in corrispondenza di 170 ns . Infatti, si può notare che questo delay viene introdotto a causa dello stato logico *unknown* dell'uscita in quanto il circuito sintetizzato sta ancora elaborando gli ingressi del sistema.

5.5.2.3. Moltiplicatore di Wallace a 16 bit

Il moltiplicatore di Wallace a 16 bit progettato presenta un moltiplicando ad 16 bit *unsigned* ed un moltiplicatore ad 16 bit *signed*. Pertanto, l'output del sistema è di tipologia *signed* a 32 bit.

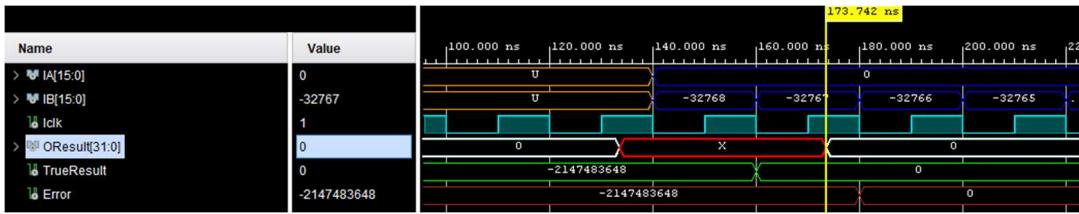


Figura 46 Post-Synthesis Timing Simulation del Moltiplicatore di Wallace a 16 bit

Anche in questo caso è possibile notare che, rispetto alla Behavioral Simulation del circuito, analizzata nei paragrafi precedenti, la Post-Synthesis Timing Simulation sta considerando i ritardi approssimativi riguardo la sintesi effettuata dalla suite. Tanto è vero che, il primo risultato idoneo si ha in corrispondenza di 173.742 ns che risulta essere, quindi, maggiore rispetto al risultato ottenuto nella simulazione comportamentale in corrispondenza di 170 ns .

5.5.3. Post-Implementation Timing Simulation

La Post-Implementation Timing Simulation, come già descritto nei capitoli precedenti, permette di effettuare un'analisi del comportamento tenendo conto dei ritardi effettivi introdotti dopo aver implementato il circuito progettato sul dispositivo di riferimento.

5.5.3.1. Moltiplicatore “Carta e Penna”

Considerando il moltiplicatore seriale “Carta e Penna” progettato, avente un moltiplicando ad 8 bit *unsigned* ed un moltiplicatore ad 8 bit *signed*, l’output generato dal sistema in questione sarà di tipologia *signed* a 16 bit.

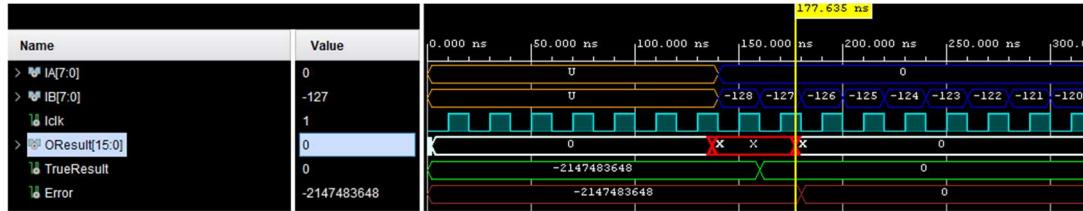


Figura 47 Post-Implementation Timing Simulation del Moltiplicatore "Carta e Penna"

È possibile notare che, rispetto alla Post-Synthesis Timing Simulation, descritta nel paragrafo precedente, la Post-Implementation Timing Simulation presenta un delay maggiore. Infatti, in questo caso i ritardi considerati sono quelli effettivi introdotti dal dispositivo di riferimento. Tanto è vero che, il primo risultato idoneo si ha in corrispondenza di 177.635 ns che risulta essere, quindi, maggiore rispetto al risultato ottenuto nella Post-Synthesis Timing Simulation in corrispondenza di 173.742 ns.

5.5.3.2. Moltiplicatore di Wallace a 8 bit

Il moltiplicatore di Wallace a 8 bit progettato presenta un moltiplicando ad 8 bit *unsigned* ed un moltiplicatore ad 8 bit *signed*. Pertanto, l'output del sistema è di tipologia *signed* a 16 bit.

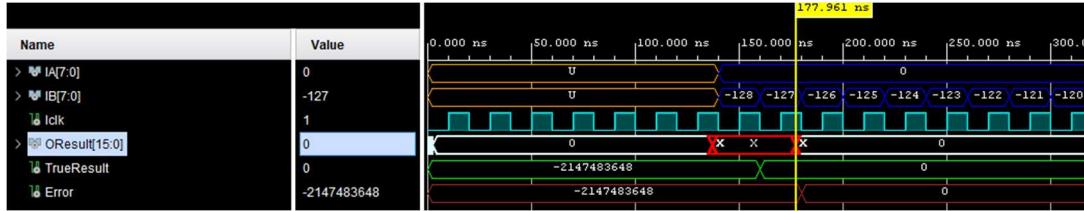


Figura 48 Post-Implementation Timing Simulation del Moltiplicatore di Wallace a 8 bit

Anche in questo caso è possibile notare che, rispetto alla Post-Synthesis Timing Simulation, descritta nel paragrafo precedente, la Post-Implementation Timing Simulation presenta un delay maggiore. Tanto è vero che, il primo risultato idoneo si ha in corrispondenza di 177.961 ns che risulta essere, quindi, maggiore rispetto al risultato ottenuto nella Post-Synthesis Timing Simulation in corrispondenza di 173.742 ns .

5.5.3.3. Moltiplicatore di Wallace a 16 bit

Il moltiplicatore di Wallace a 16 bit progettato presenta un moltiplicando ad 16 bit *unsigned* ed un moltiplicatore ad 16 bit *signed*. Pertanto, l'output del sistema è di tipologia *signed* a 32 bit.

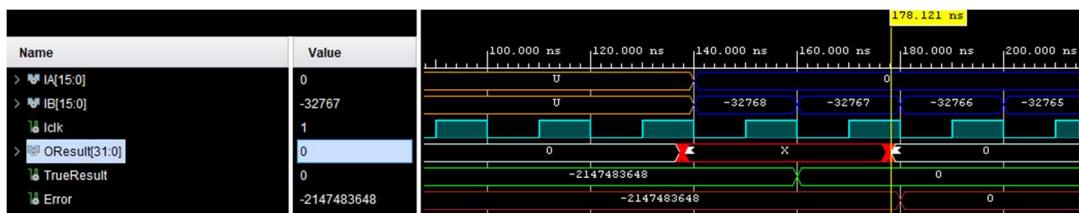


Figura 49 Post-Implementation Timing Simulation del Moltiplicatore di Wallace a 16 bit

Anche in questo caso è possibile notare che, rispetto alla Post-Synthesis Timing Simulation, descritta nel paragrafo precedente, la Post-Implementation Timing Simulation presenta un delay maggiore. Tanto è vero che, il primo risultato idoneo si ha in corrispondenza di 178.121 ns che risulta essere, quindi, maggiore rispetto al risultato ottenuto nella Post-Synthesis Timing Simulation in corrispondenza di 173.742 ns . Si può osservare che il delay appena ottenuto risulta essere maggiore di quello ottenuto dopo l'implementazione su dispositivo del Moltiplicatore di Wallace a 8 bit. Questo perché il numero di componenti

sintetizzati e implementati su device è estremamente maggiore rispetto all'ultimo circuito citato.

5.6. Report

La Vivado Design Suite, come già citato nei capitoli precedenti, permette di generare e visualizzare, dopo aver eseguito la simulazione e/o l'implementazione del circuito progettato, i report relativi al timing, all'utilizzazione delle risorse, alla dissipazione di potenza e altro ancora. Quelli relativi al timing sono stati già analizzati nel paragrafo relativo all'implementazione dei sistemi progettati. Inoltre, bisogna precisare che in questo lavoro di tesi verranno analizzati solo i report di utilizzazione delle risorse relativi all'implementazione. In aggiunta, bisogna ricordare che i report relativi alla dissipazione di potenza è possibile generarli direttamente solo dopo aver effettuato l'implementazione del circuito su device e che, inoltre, essi sono stati generati dopo aver applicato il constraint di clock, citato nei paragrafi precedenti, al sistema progettato.

5.6.1. Utilizzazione delle Risorse

I report relativi all'utilizzazione delle risorse permettono di analizzare quante risorse sono state utilizzate per la sintesi e/o l'implementazione del dispositivo di riferimento. Pertanto, è possibile effettuare un confronto tra i sistemi, così da verificare quale risulta essere più conveniente dal punto di vista del costo o per analizzare possibili ottimizzazioni da effettuare.

5.6.1.1. Moltiplicatore “Carta e Penna”

Considerando il moltiplicatore seriale di tipologia “Carta e Penna”, avente due ingressi ad 8 bit rispettivamente *unsigned* e *signed*, un’uscita a 16 bit *signed*, 56 porte logiche *and* e 8 operazioni di somma in cascata realizzate mediante Ripple-Carry Adder, il report relativo all’utilizzazione delle risorse, generato dalla suite dopo aver effettuato l’implementazione del circuito in questione, è il seguente:

Resource	Utilization	Available	Utilization %
LUT	141	53200	0.27
FF	32	106400	0.03
IO	33	200	16.50
BUFG	1	32	3.13

Figura 50 Report di utilizzazione delle risorse del Moltiplicatore "Carta e Penna"

Si può notare come il numero di LUT utilizzate è davvero minimo, cioè pari al 0.27% delle 53200 presenti all’interno del dispositivo di riferimento. Inoltre, sono stati utilizzati 32 Flip-Flop, 16 dei quali impiegati per i registri degli ingressi del sistema e i rimanenti 16 per il registro dell’uscita del circuito in questione. Si può osservare che le risorse IO⁵¹ utilizzate sono 33 su un totale di 200 disponibili. Si deve ricordare che le risorse input-output, appena citate, corrispondono ai buffer IOB⁵², cioè i pin utilizzati per l’implementazione del circuito sul dispositivo. Tali pin corrispondono agli ingressi e alle uscite del sistema progettato. Inoltre, è possibile notare come sia stato utilizzato un BUFG⁵³, cioè un Global Clock Simple Buffer.

⁵¹ IO è l’acronimo di Input/Output. Esso indica l’insieme delle interfacce disponibili per la gestione degli ingressi (input) e delle uscite (output) di un sistema.

⁵² IOB è l’acronimo di Input/Output Buffer. Esso indica il numero di pin che sono stati utilizzati per il dispositivo.

⁵³ BUFG è l’acronimo di Global Clock Simple Buffer. Esso è un buffer ad alto fanout che collega i segnali alle risorse di routing globali. Solitamente essi sono impiegati in reti di clock ad alto fanout.

5.6.1.2. Moltiplicatore di Wallace a 8 bit

Considerando il moltiplicatore di Wallace a 8 bit, avente due ingressi ad 8 bit rispettivamente *unsigned* e *signed* ed un'uscita a 16 bit *signed*, 56 porte logiche *and*, 36 Full-Adder, 24 Half-Adder e 2 Ripple-Carry Adder, il report relativo all'utilizzazione delle risorse, generato dalla suite dopo aver aver effettuato l'implementazione del circuito in questione, è il seguente:

Resource	Utilization	Available	Utilization %
LUT	102	53200	0.19
FF	32	106400	0.03
IO	33	200	16.50
BUFG	1	32	3.13

Figura 51 Report di utilizzazione delle risorse del Moltiplicatore di Wallace a 8 bit

Si può osservare come il numero di LUT utilizzate sia minore di quelle utilizzate per il moltiplicatore seriale. Infatti, bisogna ricordare che la struttura ad albero di Wallace, oltre ad ottimizzare il ritardo per il calcolo dei prodotti parziali, prevede un miglioramento anche della struttura implementata. Tanto è vero che, ad esempio, il Moltiplicatore di Wallace utilizza solo 36 Full-Adder a differenza del Moltiplicatore “Carta e Penna” che utilizza 56 FA all'interno dei Ripple-Carry Adder adottati per le somme in cascata. Per quanto riguarda, invece, il numero di Flip-Flop, risorse IO e BUFG è il medesimo del moltiplicatore seriale precedentemente descritto poiché entrambi i circuiti prevedono due ingressi a 8 bit ciascuno ed un'uscita a 16 bit.

5.6.1.3. Moltiplicatore di Wallace a 16 bit

Considerando il moltiplicatore di Wallace a 16 bit, avente due ingressi ad 16 bit rispettivamente *unsigned* e *signed* ed un’uscita a 32 bit *signed*, 240 porte logiche *and*, 196 Full-Adder, 81 Half-Adder e 2 Ripple-Carry Adder, il report relativo all’utilizzazione delle risorse, generato dalla suite dopo aver aver effettuato l’implementazione del circuito in questione, è il seguente:

Resource	Utilization	Available	Utilization %
LUT	424	53200	0.80
FF	64	106400	0.06
IO	65	200	32.50
BUFG	1	32	3.13

Figura 52 Report di utilizzazione delle risorse del Moltiplicatore di Wallace a 16 bit

Si può notare come il numero di LUT utilizzate risulta essere praticamente il quadruplo dei moltiplicatori descritti precedentemente. Infatti, basta pensare al numero di FA e HA utilizzati per la gestione dei prodotti parziali. Inoltre, è possibile osservare come il numero di Flip-Flop sia raddoppiato rispetto ai circuiti precedenti. Questo perché i due ingressi considerati sono a 16 bit ciascuno mentre l’uscita considerata è a 32 bit. Tanto è vero che, le risorse input-output risultano essere pari a 65 rispetto alle 33 previste per i moltiplicatori precedentemente analizzati.

5.6.2. Potenza

Ogni circuito elettronico digitale richiede una certa quantità di potenza per funzionare. La dissipazione di potenza dipende dal numero di porte logiche, dalla potenza di commutazione, dalla frequenza di funzionamento e da molto altro ancora. Bisogna ricordare che la potenza dissipata di un circuito cresce linearmente con la frequenza di funzionamento del sistema. Infatti, essi sono legati dalla seguente relazione:

$$P_{dissipata} = E_{totale\ dissipata} \cdot f_{funzionamento}$$

Pertanto, per circuiti che lavorano a maggiori frequenze e, quindi, a maggiori velocità di funzionamento, corrisponde una potenza di dissipazione superiore.

Facendo riferimento alle porte che compongono i circuiti integrati, solitamente il calcolo della $P_{dissipata}$ si effettua sulla base della tensione di alimentazione V_{CC}

e della corrente I_{CC} . Bisogna precisare che, il valore della corrente drenata dall'alimentazione non è costante ma dipende dallo stato logico in cui le porte si trovano. Infatti, essa sarà pari a I_{CCH} quando l'uscita della porta è al valore alto mentre sarà pari a I_{CCL} quando l'uscita della porta è al valore basso. Quindi, il calcolo della potenza dissipata può essere ottenuto considerando il valore medio della corrente I_{CC} :

$$P_{dissipata} = V_{CC} \cdot I_{CC} = V_{CC} \cdot \frac{I_{CCH} - I_{CCL}}{2}$$

In generale, la potenza dissipata di un circuito elettronico digitale può essere di due tipologie: potenza statica dissipata e potenza dinamica dissipata. La prima corrisponde alla potenza dissipata in condizioni in cui l'uscita del circuito si trova ad un valore costante, mentre la seconda corrisponde alla potenza dissipata durante le commutazioni dell'uscita.

La Vivado Design Suite, dopo aver effettuato l'implementazione del circuito elettronico digitale progettato sul dispositivo di riferimento, permette di analizzare la Total On-Chip Power. Essa è possibile calcolarla tramite la seguente relazione:

$$\text{Total On - Chip Power} = \text{Static} + \text{Design Dynamic}$$

dove *Static* è ottenuta tramite la seguente somma:

$$\text{Static} = \text{Device Static} + \text{Design Static}$$

Nello specifico la Device Static indica la potenza di dispersione del transistor quando il dispositivo è alimentato ma non configurato, la Design Static indica la potenza quando il dispositivo è stato configurato e non vi è alcuna attività di commutazione (*switching*) e, infine, la Design Dynamic indica la potenza media derivante dalla logica e dall'attività di commutazione.

Inoltre, la suite permette di analizzare ulteriori parametri fondamentali quali la Junction Temperature⁵⁴, il Thermal Margin⁵⁵ e l'Effective ϑJA ⁵⁶.

5.6.2.1. Moltiplicatore “Carta e Penna”

Considerando il Moltiplicatore “Carta e Penna”, dopo aver effettuato l’implementazione del circuito in questione sul dispositivo di riferimento, è possibile analizzare i report di potenza relativi.

Total On-Chip Power:	0.112 W
Junction Temperature:	26.3 °C
Thermal Margin:	58.7 °C (4.9 W)
Effective ϑJA :	11.5 °C/W

Figura 53 Report di potenza del Moltiplicatore "Carta e Penna" (1)

Si può notare come la potenza totale on-chip del sistema in questione risulta essere pari a 0.112 W. Bisogna osservare che la suite non segnala alcun problema relativo alla potenza totale dissipata dal circuito e alla temperatura di giunzione.

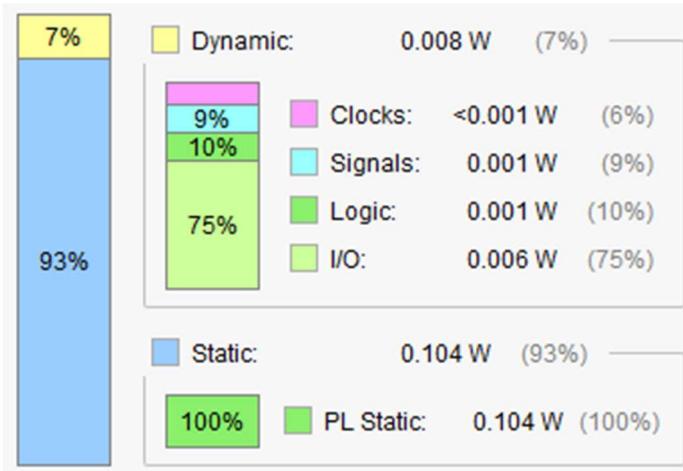


Figura 54 Report di potenza del Moltiplicatore "Carta e Penna" (2)

Si può osservare come la potenza dissipata risulta essere per il 93% di tipo statica e per il restante 7% dinamica. Nello specifico si può notare come il 75% della

⁵⁴ La Junction Temperature è la temperatura di giunzione, cioè la temperatura reale relativa ad un semiconduttore. Essa dipende dalla potenza totale del dispositivo, dal sistema di raffreddamento, dalla selezione della scheda e dalle condizioni ambientali. Poiché la temperatura di giunzione è direttamente proporzionale alla potenza totale, essa varia all’aumentare della potenza dinamica. È molto importante specificare la giusta temperatura di giunzione per stimare con precisione la potenza statica legata al dispositivo.

⁵⁵ Il Thermal Margin è il margine di temperatura consentito dal dispositivo di riferimento. Esso è negativo quando la temperatura di giunzione stimata supera il valore massimo specificato.

⁵⁶ L’Effective ϑJA , conosciuta anche come Effective Thermal Resistance To Air ϑJA , è un coefficiente che definisce come la potenza viene dissipata dal silicio del dispositivo all’ambiente.

potenza dinamica è dovuta alla potenza dissipata dalle risorse input-output mentre la restante percentuale è relativa alla logica generale e ai segnali del circuito.

5.6.2.2. Moltiplicatore di Wallace a 8 bit

Considerando il Moltiplicatore di Wallace a 8 bit, dopo aver effettuato l'implementazione del circuito in questione sul dispositivo di riferimento, è possibile analizzare i report di potenza relativi.

Total On-Chip Power:	0.112 W
Junction Temperature:	26.3 °C
Thermal Margin:	58.7 °C (4.9 W)
Effective θ_{JA}:	11.5 °C/W

Figura 55 Report di potenza del Moltiplicatore di Wallace a 8 bit (1)

Si può notare come anche in questo caso la potenza on-chip dissipata è pari a $0.112 W$. Pertanto, in termini di dissipazione di potenza, i circuiti progettati, Moltiplicatore “Carta e Penna” a 8 bit e Moltiplicatore di Wallace a 8 bit, non presentano differenze.

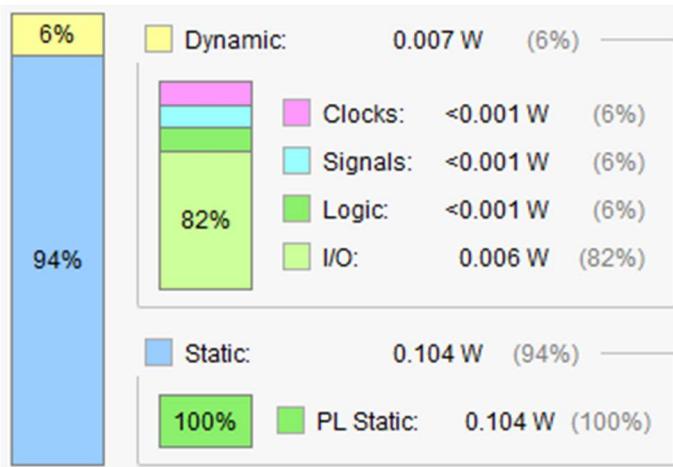


Figura 56 Report di potenza del Moltiplicatore di Wallace a 8 bit (2)

Nello specifico, si può osservare un lievissimo miglioramento, in termini di dissipazione di potenza, da parte del Moltiplicatore di Wallace a 8 bit. Infatti, sommando i contributi ottenuti riguardo la potenza statica e dinamica del circuito, essa risulta essere pari a $0.111 W$. Pertanto, si può pensare ad un'approssimazione effettuata direttamente dalla suite sui millesimi della potenza dissipata calcolata.

5.6.2.3. Moltiplicatore di Wallace a 16 bit

Considerando il Moltiplicatore di Wallace a 16 bit, dopo aver effettuato l'implementazione del circuito in questione sul dispositivo di riferimento, è possibile analizzare i report di potenza relativi.

Total On-Chip Power:	0.125 W
Junction Temperature:	26.4 °C
Thermal Margin:	58.6 °C (4.9 W)
Effective θ_{JA}:	11.5 °C/W

Figura 57 Report di potenza del Moltiplicatore di Wallace a 16 bit (1)

Si può osservare come la potenza on-chip dissipata è pari a 0.125 W , cioè maggiore rispetto ai circuiti precedentemente citati. D'altronde il moltiplicatore in questione presenta due operandi a 16 bit, un'uscita a 32 bit e un numero maggiore di funzioni logiche implementate rispetto al Moltiplicatore “Carta e Penna” e al Moltiplicatore di Wallace a 8 bit.

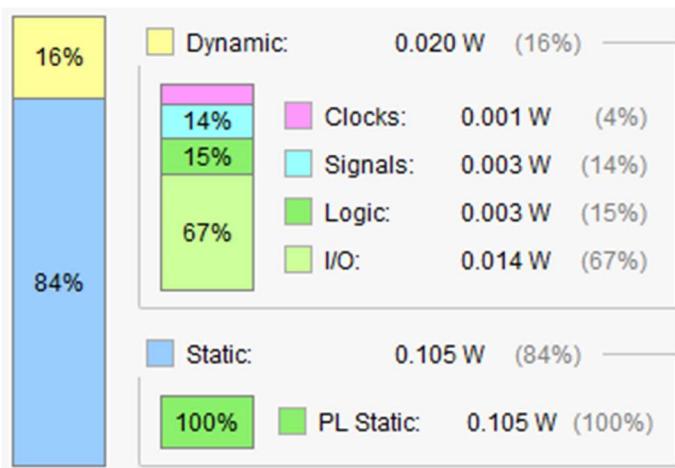


Figura 58 Report di potenza del Moltiplicatore di Wallace a 16 bit (2)

Si può osservare come l'84% della potenza dissipata totale sia di tipologia statica mentre il 16% rimanente dinamica. Nello specifico, il 67% della potenza dinamica dissipata è relativa alle risorse input-output utilizzate e la rimanente percentuale è legata ai segnali e alla logica utilizzata per la realizzazione del circuito.

6. Conclusioni

In conclusione, si può evincere come i moltiplicatori presentino, al giorno d'oggi, un ruolo fondamentale nelle applicazioni elettroniche digitali. Attualmente, i ricercatori di tutto il mondo hanno come obiettivo la riduzione al minimo del ritardo relativo alla generazione dei prodotti parziali mantenendo sempre una determinata occupazione di area sul chip e un valore limitato di dissipazione di potenza.

Proprio per questo motivo, l'obiettivo della tesi è stato quello di analizzare, progettare e mettere a confronto una struttura standard della moltiplicazione binaria e una struttura più complessa ma che permette di effettuare il medesimo calcolo. Si è notato come il moltiplicatore seriale “Carta e Penna”, trattandosi di un circuito che adotta un approccio sequenziale, risulta essere più semplice da implementare pur mantenendo un ritardo non indifferente. Proprio per questo motivo è stata progettata e analizzata la struttura del moltiplicatore seriale-parallelo di Wallace che ha permesso di esaltarne il minore delay pur presentando una struttura piuttosto articolata. Infatti, nel quarto paragrafo del quinto capitolo, si è potuto notare come quest'ultimo circuito presenti un'occupazione di area sul chip maggiore rispetto al moltiplicatore seriale sopra citato. Bisogna ricordare, però, che la dissipazione di potenza, confrontando il circuito seriale e quello seriale-parallelo ad 8 bit, risulta essere praticamente la medesima. Quindi, si può affermare che il moltiplicatore di Wallace sia riuscito a diminuire il delay complessivo relativo alla generazione dei prodotti parziali mantenendo una potenza dissipata uguale al sistema seriale e rinunciando, quindi, alla struttura semplice che presenta il moltiplicatore standard.

Come già affermato nel quarto capitolo, i moltiplicatori digitali sono fondamentali al giorno d'oggi nell'ambito del *digital signal processing* e delle reti neurali. Nello specifico, è molto importante analizzare e progettare nuove strutture di moltiplicatori binari che permettono la medesima operazione ma con costi e ritardi contenuti. Tanto è vero che, già come descritto nei capitoli precedenti, oggigiorno, stanno prendendo il sopravvento i moltiplicatori approssimati che permettono di mantenere i parametri sopra citati relativamente bassi pur mantenendo una soglia di errore limitata rispetto al risultato esatto.

Inoltre, bisogna ricordare che le suite utilizzate attualmente risultano essere di notevole importanza. Tanto è vero che, senza di esse, il processo di progettazione e simulazione del circuito risulterebbe essere più lungo e, soprattutto, più incline

agli errori. Infatti, basta pensare ai possibili collegamenti errati tra moduli o a possibili input non conformi alle interfacce dei componenti. Proprio per questo motivo, le suite permettono, attraverso strumenti di debugging e analisi, di progettare interfacce e modelli attraverso linguaggi di descrizione dell'hardware standardizzati che concedono allo sviluppatore di delineare nei minimi particolari i moduli progettati a priori. Bisogna sottolineare l'importanza della possibilità di effettuare simulazioni dei circuiti descritti all'interno della suite. Esse, effettivamente, permettono di analizzare gli ingressi che vengono utilizzati per l'analisi del sistema e l'evoluzione sia di possibili segnali intermedi utilizzati sia degli output generati dal circuito. Inoltre, di notevole importanza è la possibilità di impostare dei constraint di vario genere che permettono il soddisfacimento di determinati vincoli imposti durante la progettazione. Quest'ultimi, solitamente, vengono introdotti dai progettisti affinché sia possibile far funzionare il sistema secondo alcuni parametri o modalità così da garantire prestazioni ottimali e costi relativamente bassi. Pertanto, pensare di sviluppare oggigiorno nuovi sistemi elettronici, anche se complessi, risulta essere più agevole rispetto agli anni precedenti tale da garantire tempi molto più veloci sia dal punto di vista della progettazione che della simulazione.

7. Bibliografia

- [1] S. Perri, *FPGA*.
- [2] AVNET, «AVNET,» [Online]. Available: https://www.avnet.com/wps/wcm/connect/onesite/7ae0f288-1cc5-4283-9e85-300c5401b680/GS-AES-Z7EV-7Z020-G-V7-1.pdf?MOD=AJPERES&CACHEID=ROOTWORKSPACE.Z18_NA5A1I41L0ICD0ABNDMDDG0000-7ae0f288-1cc5-4283-9e85-300c5401b680-obW1aLr
- [3] A. Silberschatz, P. B. Galvin e G. Gagne, *Sistemi Operativi - Concetti ed esempi*, Decima edizione a cura di, M. Riccardo, A cura di, Pearson, 2019.
- [4] Intel Corporation, «Cos'è la velocità di clock?,» [Online]. Available: [https://www.intel.it/content/www/it/it/gaming/resources/cpu-clock-speed.html#:~:text=La%20frequenza%20definisce%20il%20numero,milioni%20di%20cicli%20al%20secondo.\)](https://www.intel.it/content/www/it/it/gaming/resources/cpu-clock-speed.html#:~:text=La%20frequenza%20definisce%20il%20numero,milioni%20di%20cicli%20al%20secondo.))
- [5] Wikipedia, L'enciclopedia libera, «Processore,» [Online]. Available: <https://it.wikipedia.org/wiki/Processore>.
- [6] Wikipedia, L'enciclopedia libera, «Attuatore,» [Online]. Available: <https://it.wikipedia.org/wiki/Attuatore>.
- [7] Wikipedia, L'enciclopedia Libera, «Microprocessore,» [Online]. Available: <https://it.wikipedia.org/wiki/Microprocessore>.
- [8] Wikipedia, L'enciclopedia libera, «Field Programmable Gate Array,» [Online]. Available: https://it.wikipedia.org/wiki/Field_Programmable_Gate_Array.
- [9] cadence, «What is hardware software co-design and how can it benefit you or your business,» [Online]. Available: <https://resourcespcb.cadence.com/blog/2019-what-is-hardware-software-co-design-and-how-can-it-benefit-you-or-your-business>.
- [10] FARE ELETTRONICA, «Schede di sviluppo: cosa sono e come funzionano,» [Online]. Available: <https://farelettronica.it/schede-di-sviluppo/>.
- [11] Xilinx, «Vivado Design Suite User Guide,» [Online]. Available: https://www.xilinx.com/content/dam/xilinx/support/documents/sw_manuals/xilinx2022_1/ug910-vivado-getting-started.pdf.
- [12] Xilinx, «Vivado Design Suite User Guide - Getting Started,» [Online]. Available: https://www.xilinx.com/content/dam/xilinx/support/documents/sw_manuals/xilinx2021_2/ug910-vivado-getting-started.pdf.
- [13] Wikipedia, L'enciclopedia libera, «Institute of Electrical and Electronics Engineers,» [Online]. Available: https://it.wikipedia.org/wiki/Institute_of_Electrical_and_Electronics_Engineers.
- [14] Wikipedia, L'enciclopedia libera, «Xilinx Vivado,» [Online]. Available: https://en.wikipedia.org/wiki/Xilinx_Vivado.
- [15] Wikipedia, L'enciclopedia libera, «SRAM,» [Online]. Available: <https://it.wikipedia.org/wiki/SRAM>.

- [16] Wikipedia, L'enciclopedia libera, «Memory refresh,» [Online]. Available: https://it.wikipedia.org/wiki/Memory_refresh.
- [17] M. Albasini, «multiplexer,» [Online]. Available: <https://www.marcoalbasini.com/2020/11/multiplexer/>.
- [18] R. C. Jaeger e T. N. Blalock, Microelettronica, Quinta Edizione a cura di, P. Bardella, A cura di, Mc Graw Hill, 2018.
- [19] Fastweb, «Com'è fatto e come funziona un transistor,» [Online]. Available: <https://www.fastweb.it/fastweb-plus/digital-magazine/com-e-fatto-e-come-funziona-un-transistor/>.
- [20] F. Crupi, *MOSFET*.
- [21] Wikipedia, L'enciclopedia libera, «Institute of Electrical and Electronics Engineers,» [Online]. Available: https://it.wikipedia.org/wiki/Institute_of_Electrical_and_Electronics_Engineers.
- [22] outofbit, «Che cos'è un SoC?,» [Online]. Available: <https://www.outofbit.it/che-cose-un-soc/>.
- [23] Wikipedia, L'enciclopedia libera, «Netlist,» [Online]. Available: <https://it.wikipedia.org/wiki/Netlist>.
- [24] Wikipedia, L'enciclopedia libera, «Circuito digitale,» [Online]. Available: https://it.wikipedia.org/wiki/Circuito_digitale.
- [25] S. Petrizzelli, *Capitolo 7 - Famiglie logiche*.
- [26] Wikipedia, L'enciclopedia libera, «Circuito integrato,» [Online]. Available: https://it.wikipedia.org/wiki/Circuito_integrato.
- [27] Università degli Studi di Napoli Federico II (UniNa), *Invertitore CMOS*.
- [28] Wikipedia, L'enciclopedia libera, «Tensione di soglia,» [Online]. Available: https://it.wikipedia.org/wiki/Tensione_di_soglia.
- [29] S. Immareddy e A. Sundaramoorthy, «A survey paper on design and implementation of multipliers for digital system applications,» *Artificial Intelligence Review*.
- [30] Università degli Studi di Padova (Unipd), *Lezione_5_colori*.
- [31] Real Digital, «Synthesize, Implementation, and Generate Bitstream,» [Online]. Available: <https://www.realdigital.org/doc/bd6a53089056fc9e2888deabdfcb2a66>.
- [32] Xilinx, *Vivado Design Suite User Guide - Implementation*.
- [33] Xilinx, *RTL and Technology Schematic Viewers Tutorial*.
- [34] Xilinx, *Vivado Design Suite User Guide - Synthesis*.
- [35] Xilinx, *Vivado Design Suite Tutorial - Logic Simulation*.
- [36] Xilinx, *Vivado Design Suite User Guide - Power Analysis and Optimization*.
- [37] Xilinx, *Vivado Design Suite 7 Series FPGA and Zynq-7000 SoC Libraries Guide*.
- [38] Xilinx, *Xilinx Power Estimator User Guide*.

- [39] Wikipedia, l'enciclopedia libera, «Elaborazione numerica dei segnali,» [Online]. Available: https://it.wikipedia.org/wiki/Elaborazione_numerica_dei_segnali.
- [40] Wikipedia, l'enciclopedia libera, «Finite impulse response,» [Online]. Available: https://it.wikipedia.org/wiki/Finite_impulse_response.
- [41] Wikipedia, l'enciclopedia libera, «Convoluzione,» [Online]. Available: <https://it.wikipedia.org/wiki/Convoluzione>.
- [42] Wikipedia, l'enciclopedia libera, «Rete neurale artificiale,» [Online]. Available: https://it.wikipedia.org/wiki/Rete_neurale_artificiale.
- [43] Wikipedia, l'enciclopedia libera, «Apprendimento automatico,» [Online]. Available: <https://it.wikipedia.org/wiki/ApprendimentoAutomatico>.
- [44] Wikipedia, l'enciclopedia libera, «Divide et impera (informatica),» [Online]. Available: [https://it.wikipedia.org/wiki/Divide_et_impera_\(informatica\)](https://it.wikipedia.org/wiki/Divide_et_impera_(informatica)).
- [45] Wikipedia, l'enciclopedia libera, «Debugging,» [Online]. Available: <https://it.wikipedia.org/wiki/Debugging>.

8. Ringraziamenti

Un ringraziamento speciale ai miei genitori che mi hanno sostenuto durante tutte le scelte. Vi dedico ogni mio traguardo. Siete le persone più importanti della mia vita senza le quali non sarei la persona che sono tutt'ora. Cercherò sempre di rendervi orgogliosi. Sono felice di somigliarvi in tutto e per tutto. La tenacia, l'allegria, la sincerità, la generosità e l'onestà sono i principali valori della vita che mi avete trasmesso da quando sono nato e che sono fiero di perseguire ogni giorno. Vi ringrazio per l'infinita pazienza e per ogni volta che mi avete sostenuto sia nei momenti di gioia che nei momenti di tristezza. Quando mi vedevi perso, ho trovato in voi la forza di rialzarmi e di andare avanti. Queste mie parole sono il nulla rispetto al sostegno che mi avete dimostrato durante questo percorso. Spero di potervi rendere sempre più orgogliosi. Io sarò sempre al vostro fianco e sappiate di poter contare sempre sul mio aiuto.

Grazie a mio nonno Costantino sempre presente e informato su tutto ciò che succede: mio grande consigliere e guida di tutti i giorni.

Grazie a mia nonna Maria (*nonnina*) che, anche se da lassù, rimani il mio punto di riferimento. Fin dall'infanzia hai cercato di consigliarmi nel migliore dei modi seguendo costantemente sani principi. Grazie a te non mi sono mai accontentato ma ho cercato di fare sempre di più.

Grazie ai miei nonni Amilcare e Maria che, anche se non ci sono più da alcuni anni, hanno sempre creduto in me insegnandomi a non arrendermi mai davanti alle difficoltà. Oggi, sareste orgogliosi di me.

Grazie ai miei zii Antonio, Maria Carmela, Tommaso e Anna Maria e ai miei cugini Alessia, Martina e Fabrizio per essermi stati accanto e avermi sostenuto durante il mio percorso di studi, incoraggiandomi e consigliandomi nel migliore dei modi. Un ringraziamento particolare a zio Antonio per la sua infinita pazienza e immensa disponibilità per la rilegatura della tesi.

Grazie alle mie zie Ilde e Ida per avermi fatto compagnia nei lunghi pomeriggi invernali durante la mia infanzia.

Grazie a Fabrizio e Igar, miei amici sinceri di vita da sempre, per aver condiviso insieme momenti di gioia e per essermi stati accanto nei momenti di difficoltà. Grazie per volermi bene per quello che sono e senza i quali non sarei quello che sono. Sono sicuro che riuscirete a realizzare i vostri sogni.

Grazie a Daniela e Francesca, compagne di liceo e tutt'oggi amiche vicine, per il sostegno che mi avete dimostrato e che mi manifestate ogni giorno. Senza di voi le serate e le feste sarebbero una noia mortale. Auguro a Daniela di diventare la miglior designer di interni e a Francesca la migliore tatuatrice che io conosca.

Grazie a Niccolò e a Mariachiara, miei amici fedeli e grandi *ascoltatori* delle mie preoccupazioni, per essermi stati vicino nei miei momenti più bui. Auguro a Niccolò di diventare il miglior preparatore atletico e a Mariachiara la migliore biotecnologa.

Grazie a Francesca, mia compagna di banco dalle elementari, per avermi sempre dimostrato la sua amicizia sincera. Ormai insieme ai tuoi genitori, Emanuela e Nicola, siete parte integrante della famiglia. Auguro a te e Francesco di realizzare i vostri sogni.

Grazie a Liliana, mia amica leale e sincera, per avermi supportato sempre. Anche se ci vediamo raramente, quelle poche volte riescono a compensare tutti i momenti persi. Ti auguro di diventare la migliore professoressa di letteratura inglese.

Grazie a Giuseppe, mio vicino di banco delle superiori e di casa, per esserci sostenuti a vicenda, nella buona e nella cattiva sorte, durante i momenti di sconforto e nei momenti di gioia che hanno caratterizzato la nostra amicizia. Grazie anche a Matteo e Marco per le belle serate trascorse insieme. A tutti voi auguro di raggiungere i migliori traguardi.

Grazie a Ludovica, mia collega di università e amica sincera, per avermi sostenuto durante i miei esami e per essersi sempre preoccupata anche delle più piccole cose. Ti auguro di diventare il miglior ingegnere informatico che io conosca.

Grazie a Vladimiro, Giuseppe, Roberta, Antonella e a tutti gli altri miei colleghi di università per tutti i vostri consigli e per tutte le volte che abbiamo studiato insieme. Un particolare ringraziamento va a Vladimiro per le serate trascorse insieme e per tutti i momenti che abbiamo condiviso. Auguro a voi tutti il meglio.

Grazie alla Prof.ssa Perri, mio relatore, per la sua continua disponibilità, per avermi seguito costantemente e per avermi permesso di trattare un argomento interessante durante il mio lavoro di tesi. Inoltre, grazie ai suoi preziosi consigli ho scelto quello che sarà il mio percorso di studi dei prossimi anni.

Infine, il ringraziamento più grande va a me stesso, per il duro lavoro, l'impegno e la costanza che ho avuto durante questi tre anni. Dopo un lungo periodo buio, fatto di momenti bassi, sono riuscito finalmente a raggiungere il mio obiettivo. Ricordando tutti i sacrifici e la grinta che ci ho messo, posso dire di essere fiero di me stesso.

Per aspera ad astra!