UNIVERSITA' DEGLI STUDI DI PISA

Facoltà di Ingegneria

Corso di Laurea in Ingegneria Elettronica

TESI DI LAUREA

Analisi comparativa in termini di System-Gate equivalenti fra diverse tecnologie FPGA utilizzando un Processore FFT come riferimento

Relatori:

Prof. Luca Fanucci

Prof. Sergio Saponara

Candidato:

Giuseppe Morabito

Anno accademico 2006/2007

Indice

Introduzione

Capitolo 1 : Trasformazione LUT-GATE per i dispositivi ALTERA e XILINX ……..1

Introduzione

1.1 Trasformazione LUT-GATE del dispositivo FPGA ALTERA STRATIX ……...1

1.2 Trasformazione LUT-GATE del dispositivo FPGA XILINX SPARTAN-3 ……..9

Capitolo 2 : Descrizione del Processore FFT ……16

2.1 Introduzione ……...16

2.2 Descrizione dell'algoritmo ……..17

2.2.1 Algoritmo di tipo Radix …….....17

2.2.2 Algoritmo di tipo Be & Jones ……18

2.3 Descrizione dell'architettura …….23

2.3.1 Commutatore…….25

2.3.2 Butterfly…….....26

2.3.3 Moltiplicatore Complesso ……...27

2.3.4 Rom dei coefficienti di Twiddle ……28

2.3.5 Control Unit …….....28

Capitolo 3 : Simulazione funzionale Pre-Sintesi ……..30

3.1 Simulazione funzionale del processore FFT …….30

Capitolo 4 : Sintesi del processore FFT con aritmetica BFP ……...…34

Introduzione……...…….34

4.1 Sintesi del Processore FFT-BFP sul dispositivo FPGA ALTERA STRATIX ……..….35

4.2 Sintesi del Processore FFT-BFP sul dispositivo FPGA XILINX SPARTA-3 ……..42

Capitolo 5 : Raccolta dei dati e conclusioni……49

5.1 Raccolta dei dati……..49

5.1.1 Dati Stratix…….….50

5.1.2 Dati Spartan-3……..…..53

5.2 Conclusioni ed esempi……..57

BIBLIOGRAFIA……59

INTRODUZIONE

Lo scopo di questa tesi è quello di trovare uno strumento tramite il quale è possibile quantificare secondo la metrica del SYSTEM GATE la capienza della logica in un dispositivo FPGA. Il SYSTEM GATE è una metrica che deriva dai dispositivi ASIC ed è rappresentato da una cella contenente al suo interno due P-MOS e due N-MOS (vedi figura),

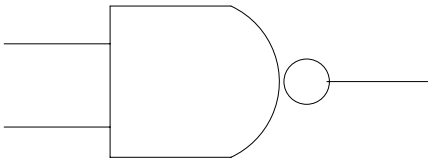


Figura 1 : Gate System Gate

Ma non può essere applicato con tale criterio ai dispositivi FPGA, infatti negli FPGA non abbiamo celle contenenti N-MOS e P-MOS, allora viene considerato Gate di Sistema il peso che ha una porta NAND a due ingressi, inoltre secondo quest'ultimo criterio possiamo definire il peso di varie porte secondo i Gate di Sistema alcune delle quali sono catalogate nella tabella 1.
Secondo queste ultime affermazioni possiamo quantificare basandoci sulla metrica dei Gate di Sistema un intero circuito logico, che a sua volta può essere usato per testare la capienza dei dispositivi FPGA, in questo modo creiamo una funzione campione che mappata tramite un tool di sintesi su un dispositivo FPGA ci da l'effettiva capienza del dispositivo, possiamo pensare che il tool di sintesi sia uno strumento di misura e la funzione che stiamo cercando potrebbe essere il meccanismo di conversione al suo interno.

PORTA LOGICA	GATE di SISTEMA (equivalenti)
NAND2	1
NANDn	(n-1)
OR2	1
ORn	(n-1)
XOR2	3
XORn	3*(n-1)

Tabella 1 : Metrica del SYSTEM GATE applicata alle porte logiche

Nel Capitolo1 sono stati scelti alcuni dispositivi FPGA sui quali effettuare la misura, è stata studiata l'architettura reperendo informazioni dai Data Sheet forniti dai costruttori e sono state effettuate sintesi di alcune funzioni logiche per avere le prime indicazioni sui criteri secondo i quali le funzioni logiche vengono mappate in maniera diversa nei diversi dispositivi.

Nel Capitolo 2 è stato descritto il Processore FFT che è la funzione che verrà usata in seguito per la valutazione di un dispositivo FPGA secondo la metrica dei Gate di Sistema.

Nel Capitolo 3 è stata effettuata una simulazione funzionale del Processore FFT per dimostrarne l'effettivo funzionamento. È importante usare una funzione che abbia un riscontro reale, l'uso di una funzione fittizia potrebbe falsare le misure questo perché una funzione che viene scritta ed usata al solo scopo di riempire un dispositivo FPGA può avere delle caratteristiche che un progettista se ne guarderebbe bene di usarle , come ad esempio un uso spropositato ed ingiustificato di risorse di Routine.

Nel Capitolo 4 sono state effettuate una serie di sintesi logiche del Processore FFT sui due dispositivi, variando i parametri che lo caratterizzano e con i dati forniti dal tool di sintesi è stata costruita una funzione che descrive la quantità di risorse logiche occupate dal Processore FFT nei due dispositivi.

Nel Capitolo 5 sono stati raccolti i dati relativi a tutte le sintesi logiche effettuate sui due dispositivi ed è stata costruita una funzione GATE_FFT(X), che descrive l'andamento dei Gate di Sistema del processore FFT al variare dei suoi parametri ed è stato effettuato il Benchmark di un altro codice VHDL usando come riferimento il Processore FFT ed il dispositivo SPARTAN-3.

Capitolo 1

Trasformazione Lut-Gate per i dispositivi Altera e Xilinx.

In questo capitolo facciamo un'analisi architetturale e una prima quantificazione della capienza secondo la metrica dei Gate di Sistema dei dispositivi FPGA che prendiamo come campione. I dispositivi considerati usano le LOOK-UP TABLE (LUT) per contenere funzioni logiche. In questo primo esame cerchiamo di trarre informazioni sulla quantità di logica che è possibile inserire nelle LUT dei diversi dispositivi usando il tool di sintesi Synplify Pro 8.1 prodotto da SYMNPLICITY. Prima di tutto dobbiamo mettere in evidenza che la capacità logica dei dispositivi FPGA dipende non solo dall'architettura del dispositivo ma anche dalle risorse di Routing che mette a disposizione.

1.1 Trasformazione LUT-GATE del dispositivo FPGA ALTERA STRATIX

Il dispositivo FPGA preso in considerazione si basa su una tecnologia HCMOS a 0.13 μm e 1.5 V, con una densità di 79040 elementi logici (LEs), 7.5 Mbits di RAM e 22 blocchi dedicati a DSP (Digital Signal Processing) con 176 moltiplicatori (9 bit x 9 bit) embedded. Nella Tabella 1.1 vengono riportate le caratteristiche principali del dispositivo[1].

[1] Il Data Sheet è stato preso dal sito www.altera.com

Feature	EP1S40	EP1S60	EP1S80
LEs	41,250	57,120	79,040
M512 RAM blocks (32 × 18 bits)	384	574	767
M4K RAM blocks (128 × 36 bits)	183	292	364
M-RAM blocks (4K × 144 bits)	4	6	9
Total RAM bits	3,423,744	5,215,104	7,427,520
DSP blocks	14	18	22
Embedded multipliers *(1)*	112	144	176
PLLs	12	12	12
Maximum user I/O pins	822	1,022	1,238

Tabella 1.1 : Caratteristiche principali del dispositivo ALTERA STRATIX

Gli elementi logici (LE) sono programmabili e la loro configurazione dipende dal tipo di logica che dobbiamo inserire (logica sequenziale o combinatoria), nella figura 1.1 è raffigurata l'architettura degli elementi logici.

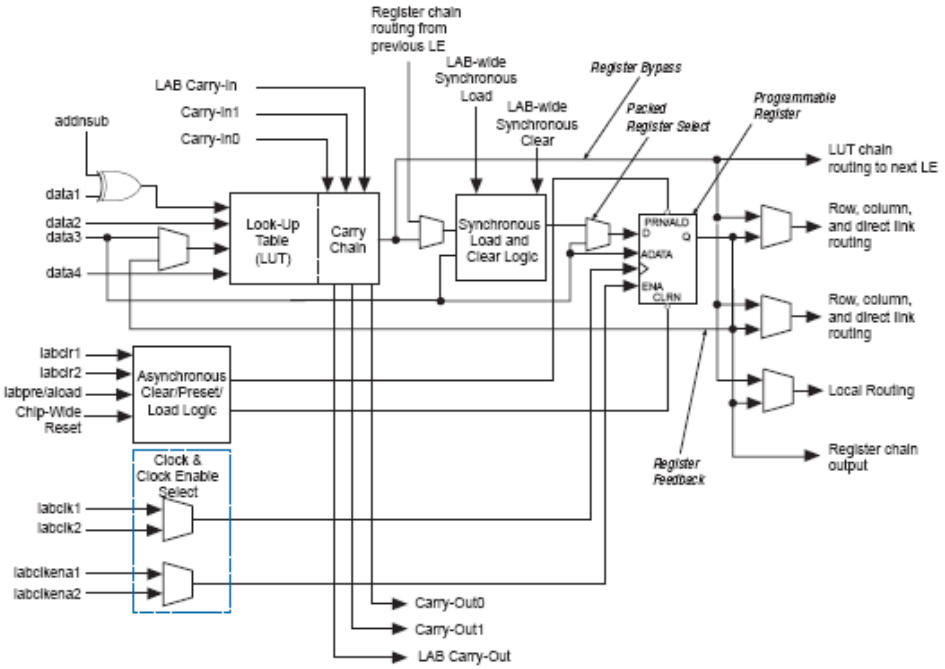


Figura 1.1 : architettura degli elementi logici (LE).

Come si vede in figura 1.1 ogni elemento logico ha un LUT a quattro ingressi e un'uscita. L'idea è quella di cercare di saturare queste LUT utilizzando delle funzioni logiche opportune.
Il primo codice VHDL preso in considerazione è parametrico e tramite i suoi parametri possiamo impostare il numero di ingressi il numero di Gate di Sistema desiderato e il numero di uscite. La definizione di Gate di Sistema ci viene in aiuto (vedi introduzione) infatti sapendo che un Gate di Sistema equivale all'area occupata da una NAND2, possiamo pensare di costruire una funzione costituita da sole porte NAND e variarne il numero tramite parametro, ovviamente questa funzione non ha nessun riscontro reale. In Figura 1.2 è riportato un pezzo del codice parametrico usato.

```
gr : for i in 0 to m-1 generate                    --genera m catene
z(i, 0) <= II(i);
        gc : for j in 0 to (n-1) generate     --genera gli n elementi della catena
         p1 : gate_4 port map ( z(i, j),  z(i, j+1)); --istanza degli n elementi
         end generate;
U(i)<=z(i,n)(0) and z(i, n)(1) and z(i,n)(2) and z(i, n)(3);
end generate;
```

Figura 1.2 : Istanza delle porte NAND

Lo schema elettrico delle porte gate_4 che vengono inserite in serie per formare una catena è raffigurato in figura 1.3 mentre le catene sono raffigurate nella sua interezza in figura 1.4. Ogni blocco equivale a 4 GATE, ogni catena ha 4 ingressi e una uscita, infatti come possiamo vedere in figura è stata inserita in uscita una NAND3.

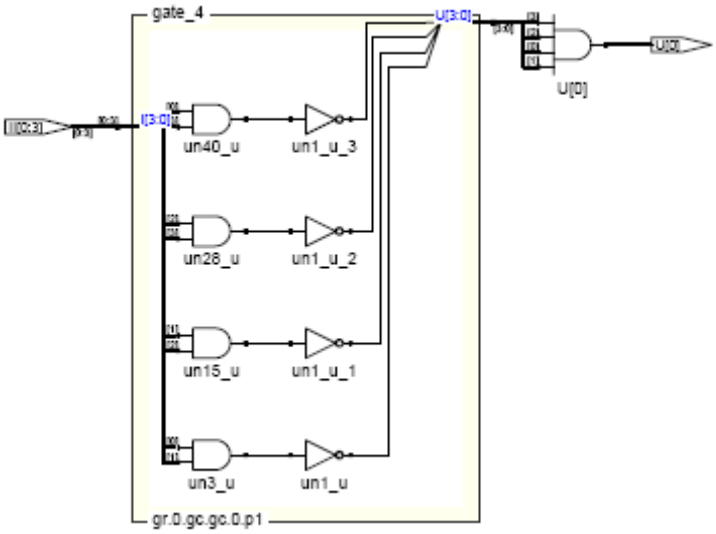


Figura 1.3 : Elemento della catena

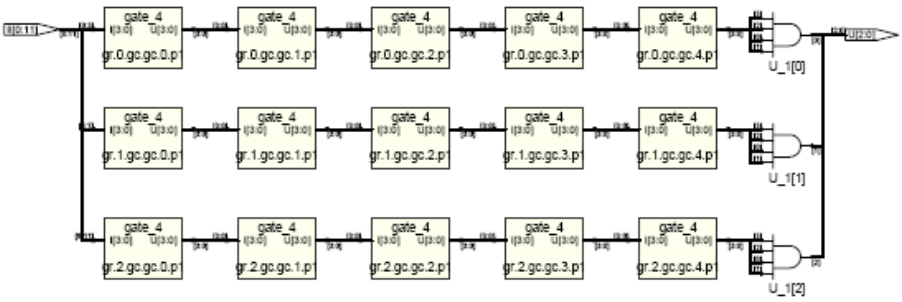


Figura 1.4 : Rappresentazione a porte del codice VHDL.

Possiamo quindi dire che dati:

- N= numero di porte per riga
- M = numero di righe
- 4*M= numero di ingressi
- M numero di uscite

Il numero di Gate Equivalenti sarà :

Gate = M(4N+3)

effettuiamo una serie di sintesi logiche[2] variando i parametri, alcune delle quali hanno prodotto i risultati raccolti nella tabella 1.2.

N	M	LUTs usati	GATE
100	1	401	403
300	3	3603	3609
400	4	6404	6412
600	5	12005	12015

Tabella 1.2 : Sintesi dalla funzione GATE

La figura 1.5 mostra una rappresentazione grafica dei dati raccolti tramite sintesi. Esaminando i dati si arriva alla conclusione che è stata mappata una porta NAND per ogni LUT.

Possiamo affermare con certezza che il risultato che abbiamo ottenuto si riferisce al caso peggiore (worst case).

Il nostro scopo è però quello di riuscire a saturare il più possibile un LUT, infatti è inverosimile che esso riesca a contenere solamente una porta NAND, quindi possiamo pensare che il risultato che abbiamo trovato prima è dovuto al tipo di architettura usata e dalla quantità di risorse di ROUTING che abbiamo impegnato, è infatti noto che la quantità dei Gate di Sistema di un dispositivo FPGA dipende anche dalle risorse di ROUTING che l'FPGA possiede, cerchiamo allora di inserire nel dispositivo uno schema elettrico molto piccolo rispetto alla grandezza dell'FPGA, in modo

[2] N.B. È stato necessario inserire un attributo nei Constraint per fare in modo che il tool di sintesi non dissolva alcun elemento.

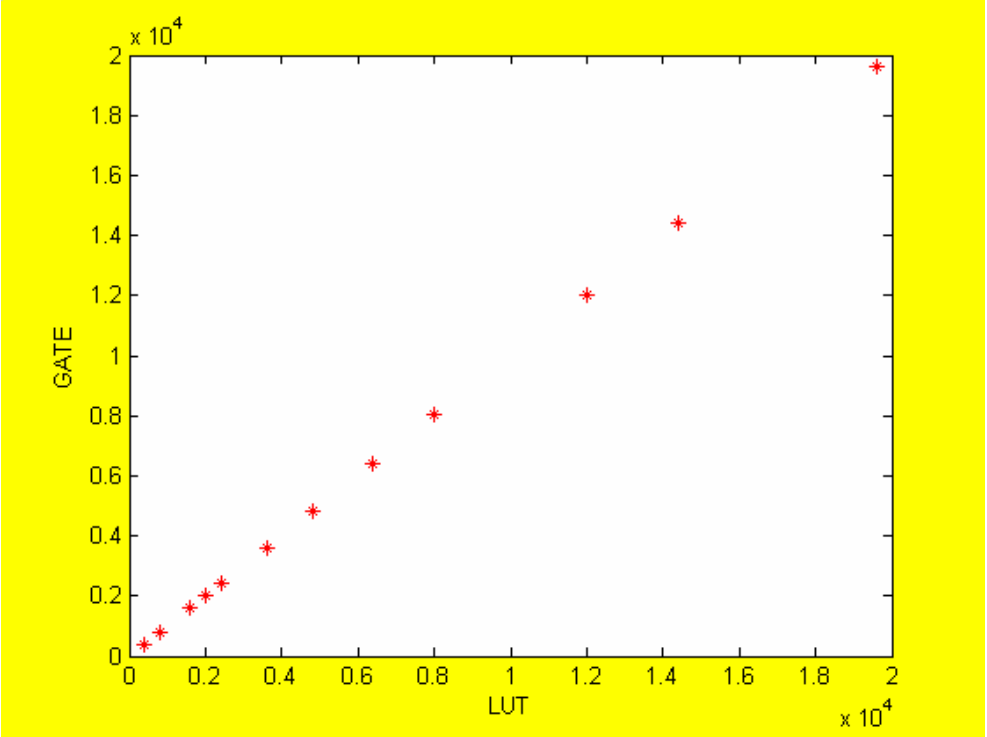


Figura 1.5 : Rappresentazione grafica dei dati della tabella 1.2

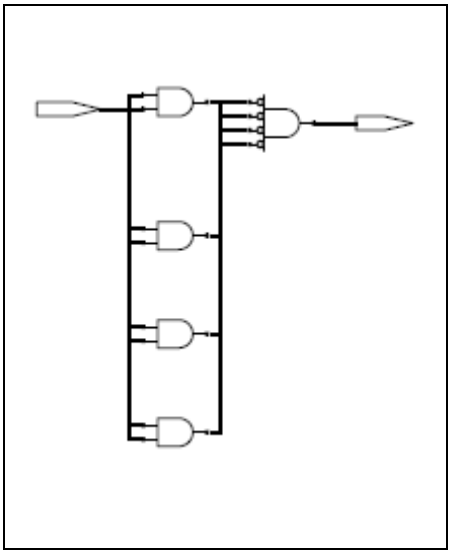
da non impegnare una grande quantità di risorse di ROUTING. Le architetture e i risultati prodotti sono riportate di seguito.

Ricordiamo che i Gate Equivalenti ad ogni architettura sono stati calcolati secondo i dati riportati nella Tabella 1.3.

Porta logica	Gate Equivalenti
NAND2	1
NANDn	n-1
OR2	1
ORn	n-1
XOR2	3
XORn	3(n-1)

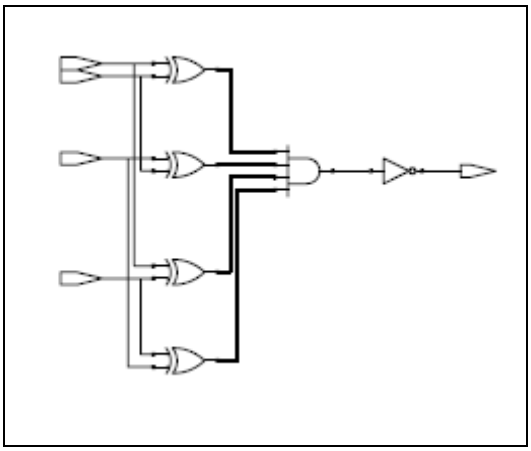
Tabella 1.3 : Gate equivalenti alle porte logiche.

A)



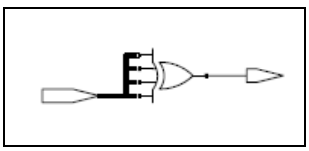
N° GATE (equiv.)	LUT (usati)	GATE/LUT
7	5	1.4

B)



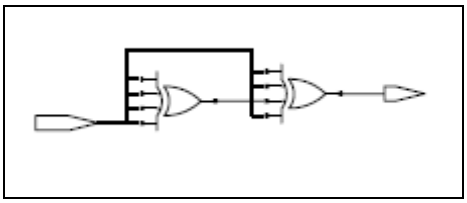
N° GATE (equiv.)	LUT (usati)	GATE/LUT
15	5	3

C)



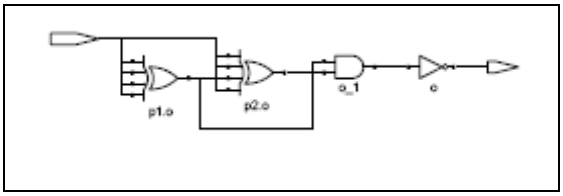
N° GATE (equiv.)	LUT (usati)	GATE/LUT
9	1	9

D)



N° GATE (equiv.)	LUT (usati)	GATE/LUT
18	2	9

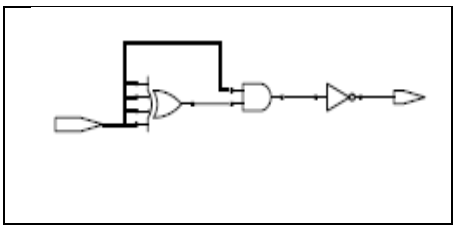
E)



N° GATE (equiv.)	LUT (usati)	GATE /LUT
19	3	6.33

F)

F)



N° GATE (equiv.)	LUT (usati)	GATE/LUT
10	2	5

Esaminando i dati troviamo che il caso C) ha prodotto il risultato maggiore e non appena proviamo ad aggiungere anche un solo Gate alla struttura (caso F)), aumenta immediatamente il numero di LUT usati. Questo però non ci assicura che in ogni LUT venga saturato con 9 Gate di Sistema indipendentemente dalla funzione che mappiamo nel dispositivo.

1.2 Trasformazione LUT-GATE del dispositivo FPGA XILINX SPARTAN-3

Il dispositivo XILINX preso in considerazione in questo paragrafo appartiene alla famiglia SPARTAN-3 ed è il dispositivo XC3S5000, le principali caratteristiche sono riportate nella tabella1.4[3]

Device	System Gates	Equivalent Logic Cells[1]	CLB Array (One CLB = Four Slices)			Distributed RAM Bits (K=1024)	Block RAM Bits (K=1024)	Dedicated Multipliers	DCMs	Maximum User I/O	Maximum Differential I/O Pairs
			Rows	Columns	Total CLBs						
XC3S50[2]	50K	1,728	16	12	192	12K	72K	4	2	124	56
XC3S200[2]	200K	4,320	24	20	480	30K	216K	12	4	173	76
XC3S400[2]	400K	8,064	32	28	896	56K	288K	16	4	264	116
XC3S1000[2]	1M	17,280	48	40	1,920	120K	432K	24	4	391	175
XC3S1500	1.5M	29,952	64	52	3,328	208K	576K	32	4	487	221
XC3S2000	2M	46,080	80	64	5,120	320K	720K	40	4	565	270
XC3S4000	4M	62,208	96	72	6,912	432K	1,728K	96	4	712	312
XC3S5000	5M	74,880	104	80	8,320	520K	1,872K	104	4	784	344

Tabella1.4 : Caratteristiche del dispositivo SPARTAN-3

Nota:

Ogni Cella Logica (Logic Cells) contiene un LOOK-UP TABLE (LUT) a 4 input e un D flip-flop, il numero equivalente di celle logiche è dato da:

"Equivalent Logic Cells" = "Total CLBs" x 8 Logic Cells/CLB x 1.125.

Nella Figura 1.6 è rappresentata l'architettura del Blocco Logico Configurabile (CLB)

[3] Il Data Sheet è stato preso dal sito www.xilinx.com

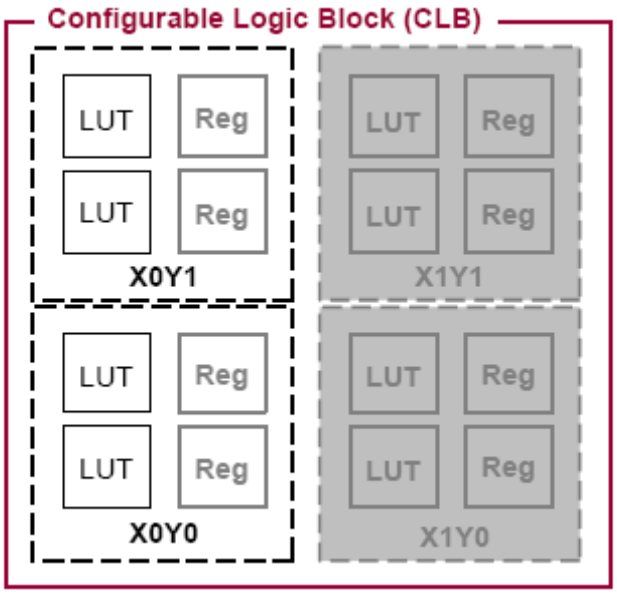


Figura 1.6 : Blocco Logico Configurabile (CLB)

Ogni CLB è costituito da quattro SLICE chiamate rispettivamente X0Y1, X0Y0, X1Y0 e X1Y1, ogni SLICE è costituita da due celle logiche, tutte le SLICE possono essere usate per l'implementazione di funzioni logiche, mentre solo le SLICE di sinistra possono essere eventualmente usate anche come RAM. L'architettura degli elementi logici (LE) all'interno della SLICE è mostrata in Figura 1.7.

Facciamo la sintesi di alcune funzioni allo scopo di saturare il più possibile i LUT e trarne delle informazioni utili al nostro scopo.

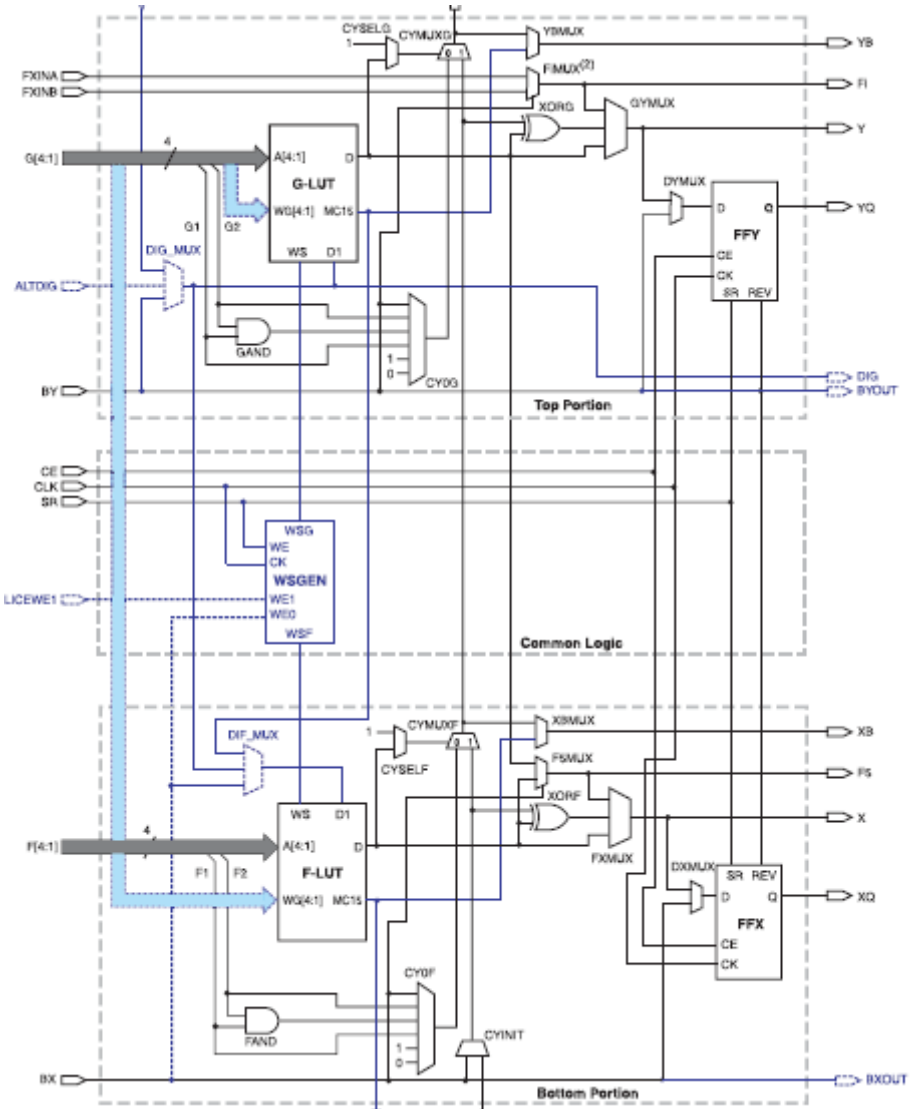


Figura 1.7 : SLICE del dispositivo FPGA SPARTAN-3

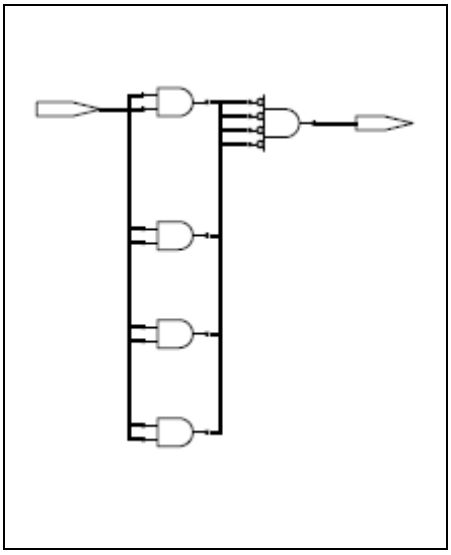
Facciamo la sintesi del codice parametrico a porte NAND (vedi Figura 1.2) e riportiamo alcuni dati nella Tabella 1.5.

N	M	LUT2	LUT4	TOTAL LUT	GATE EQUIVALENTI
100	1	400	1	401	403
300	3	3600	3	3603	3609

Tabella 1.5 : Sintesi della funzione GATE

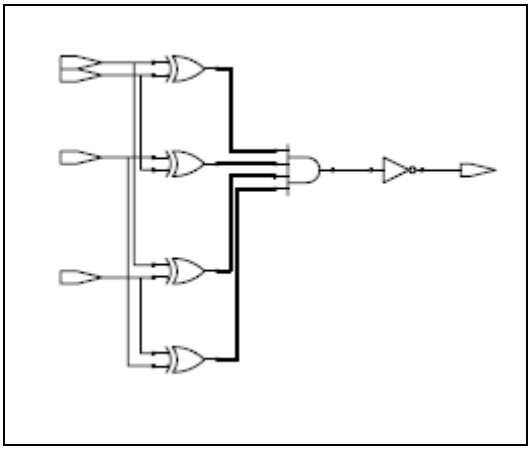
Come vediamo i risultati che abbiamo ottenuto sono identici a quelli ottenuti sul dispositivo STRATIX, abbiamo però qualche informazione in più che si riferisce ai LUT2 e ai LUT4, sappiamo quindi che l'architettura del dispositivo SPARTAN-3 permette di frazionare i LUT4 in LUT di dimensioni e con numero di ingresso inferiori, questo potrebbe permettere un riempimento maggiore dei LUT in termini di GATE di SISTEMA.
Passiamo ora alla sintesi delle altre architetture, che come abbiamo detto nel precedente capitolo sono molto piccole rispetto al dispositivo, questo per non impegnare una grande quantità di risorse di RUOTING.

A)



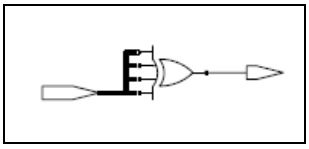
LUT2	LUT4	LUT(totali)	GATE
4	1	5	7

B)



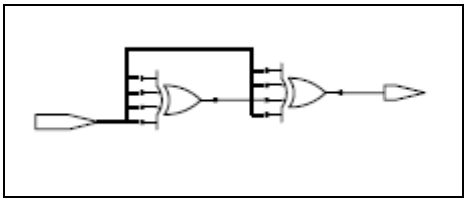
LUT2	LUT4	LUT(totali)	GATE
4	1	5	15

C)



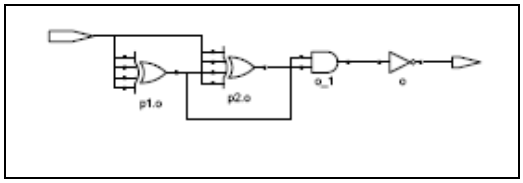
LUT4	LUT(totali)	GATE
1	1	9

D)



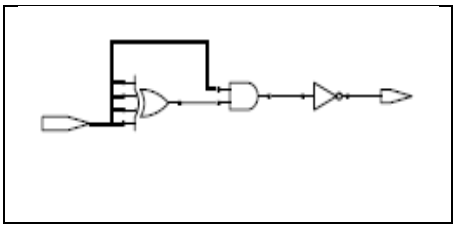
LUT4	LUT(totali)	GATE
2	2	18

E)



LUT2	LUT4	LUT(totali)	GATE
1	2	3	19

F)



LUT2	LUT4	LUT(totali)	GATE
1	1	2	10

Prendiamo in considerazione la Tabella 1.5, il numero dei LUT totali coincide con quelli usati sul dispositivo STRATIX ma il tool di sintesi ci indica il numero di LUT2 usati, allora possiamo pensare ad un LUT2 come la metà di un LUT4, con questo criterio la Tabella 1.5 si trasforma nella seguente Tabella 1.6.

N	M	LUT(presunti)	LUT(effettivi)	GATE
100	1	201	401	403
300	3	1803	3603	3609

Tabella 1.6 : analisi dei dati della tabella 1.5.

I $LUT_{presunti}$ sono stati calcolati sommando ai LUT4 la metà dei LUT2 a questo punto possiamo pensare che la differenza tra $LUT_{effettivi}$ e $LUT_{presunti}$

ci darà i "$LUT_{sprecati}$". Per quanto riguarda la sintesi delle altre architetture, il caso C) ci da il risultato maggiore cioè 9 GATE/LUT. Questo è un risultato che non ci sorprende infatti l'architettura degli Elementi Logici (LE) del dispositivo SPARTAN-3 è molto simile all'architettura degli Elementi Logici del dispositivo STRATIX.

CAPITOLO 2
DESCRIZIONE DEL PROCESSORE FFT

2.1 Introduzione

In questo capitolo desrciveremo il processore FFT soffermandoci in particolar modo sulla sua descrizione architetturale. Il processore in esame e' stato progettato in modo da implementare il calcolo della DFT/IDFT riportato di seguito:

$$X_{(k)} = \frac{1}{N}\sum_{n=0}^{N-1} x_{[n]} \cdot W_N^{nk}$$

$$W_N = e^{\pm \frac{j\,2\,\pi}{N}} \qquad (2.1)$$

$x_{[n]}$ per n=0,1,…,N-1 rappresenta il segnale di ingresso

$X_{(k)}$ per k=0,1,…,N-1 rappresenta il segnale di uscita

W_N viene definito twiddle-factor. Il segno "+" indica una operazione di IFFT mentre il segno "-" una operazione di FFT

Per implementare la (2.1) e' stato usato un algoritmo FFT (Fast Fourier Trasform) di tipo Radix, il quale consente di ridurre il numero di operazioni necessarie per il calcolo da N^2 a $N \cdot \lg_2(N)$ Il principio su cui basa questo algoritmo e' la scomposizione del calcolo della Trasformata di Fourier Discreta di una sequenza lunga N in trasformate di dimensioni piu' piccole. In particolare il tipo di algoritmo usato e' quello Mixed-Radix-2/4, che ha la caratteristica di essere composti da stadi Radix-4, tranne l'ultimo che puo'

essere Radix-2, cio' dipende dal valore di N ; infatti se N e' una potenza di 4 l'ultimo stadio e' Radix-4 altrimenti e' Radix-2.

2.2 Descrizione dell'algoritmo

2.2.1 Algoritmo di tipo radix

Come accennato nel paragrafo precedente l'algoritmo di tipo radix ci permette la scomposizione della Trasformata Discreta di Fourier di una sequenza lunga N, in una trasformata di sequenze piu' piccole basandosi sulla fattorizzazione del numero di campioni N. Infatti se N non e' un numero primo e $N = r_1 \cdot r_2$ con r_1 e r_2 entrambi maggiori di , e' possibile dividere la DFT originaria su N punti in r_1 DFT su r_2 punti o in r_2 DFT su r_1 punti, in particolare se $N = r_1 \cdot r_2 \cdots r_\nu$ con $r_1 = r_2 = \ldots = r_\nu = r$, allora e' possibile dividere la DFT su N punti in m DFT su r punti. In questo caso l'algoritmo usato e' di tipo radix-r che ha r dati di ingresso e r dati di uscita. In figura 2.1 e' riportato uno schema di calcolo dell'algoritmo FFT radix-2.

I cerchi in cui arrivano le frecce rappresentano operazioni di somma e sottrazione, i rami con i coefficienti di Twiddle rappresentano il prodotto tra il valore all'inizio del ramo e il coefficiente stesso. Come si puo' facilmente intuire il numero di stadi necessari per il calcolo della FFT di una sequenza di N numeri e' dato da $\nu = \lg_r(N)$. Come si nota dalla Figura 2.1 i dati in uscita non sono ordinati, questo implica che debbano essere riordinati all'uscita di ogni stadio prima di essere presentati allo stadio successivo.

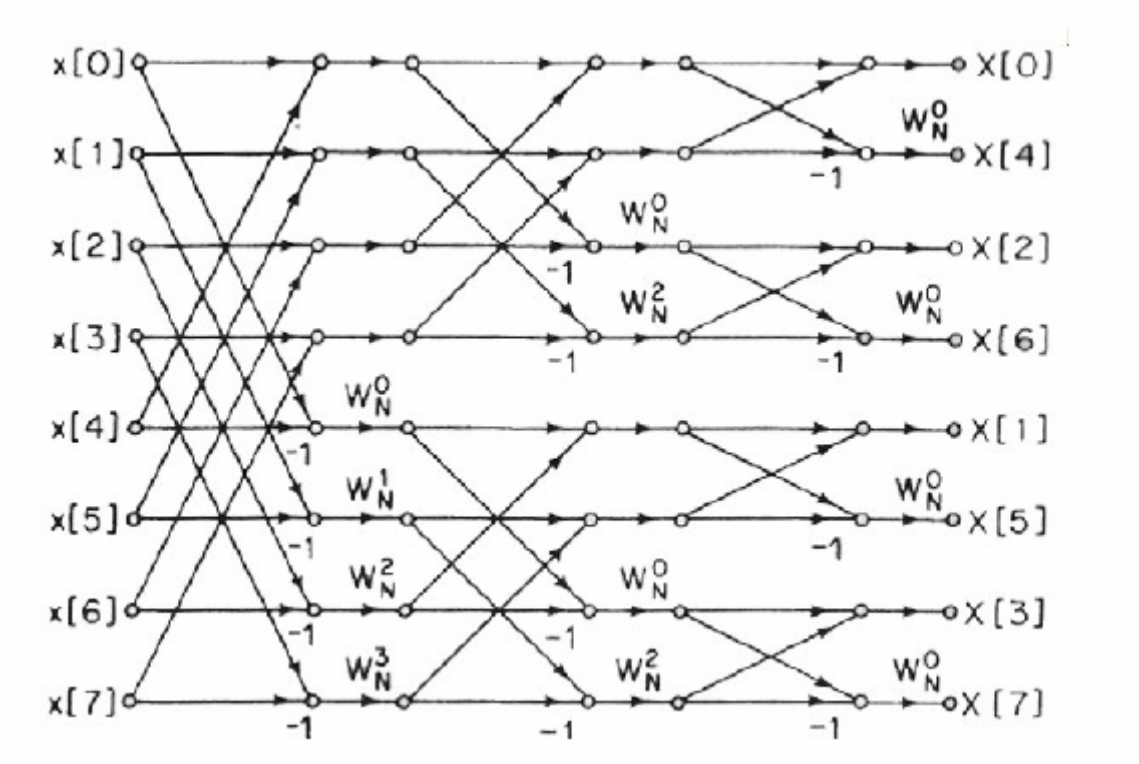


Figura 2.1: Schema di calcolo FFT radix-2 (N=8)

2.2.2 Algoritmo di tipo Bi & Jones

Come precedentemente accennato il processore FFT a cui facciamo riferimento usa un algoritmo di tipo radix, e in particolare l'algoritmo di Bi & Jones con architettura Pipeline-Cascade [Bi & Jones, 1989].

Andremo ora ad esaminare tale algoritmo scritto sotto forma di matrici.

Presa una sequenza lunga N, il calcolo della FFT di tale sequenza secondo l'algoritmo radix si basa sulla fattorizzazione di N in modo da poter calcolare la FFT di sequenze piu' piccole, in particolare avremo:

$N = r_1 \cdot r_2 \cdots r_\nu$ dove r_t e' la radice dello stadio t^{esimo} con t numero intero e ν in questo caso rappresenta il numero di radici di N e quindi il numero di stadi in cui verra' scomposta l'architettura .

Definiamo le seguenti quantita' :

$$N_t = \frac{N}{r_1 \cdot r_2 \cdots r_t} \qquad 1 \le t \le \nu - 1$$

$N_\nu = 1$

$W_M^k = e^{-j\left(\frac{2\pi k}{M}\right)}$ coefficiente di Twiddle

Detta $x_{[n]}$ la sequenza di ingresso per $0 \le n \le N-1$.

Scriviamo il vettore degli ingressi sotto forma di matrice riempiendola secondo l'ordine delle colonne.

Quindi il primo passo per t=1 e' :

$$[X]_0 = \begin{bmatrix} x_{[0]} & x_{[N_1]} & x_{[2N_1]} & \cdots & \cdots & x_{[(r_1-1)N_1]} \\ x_{[1]} & x_{[N_1+1]} & x_{[2N_1+1]} & \cdots & \cdots & x_{[(r_1-1)N_1+1]} \\ x_{[2]} & x_{[N_1+2]} & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{[N_1-1]} & x_{[2N_1-1]} & x_{[3N_1-1]} & \cdots & \cdots & x_{[r_1N_1-1]} \end{bmatrix}$$ matrice $N_1 x\ r_1$

Con $N_1 = \frac{N}{r_1}$

Effettuiamo il calcolo ${}^{m_1}x_1 = A \cdot B$ con

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & W_N^{m_1} & 0 & 0 & \cdots & 0 \\ 0 & 0 & W_N^{2m_1} & 0 & \cdots & 0 \\ 0 & 0 & 0 & W_N^{3m_1} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & W_N^{(N_1-1)m_1} \end{bmatrix}$$ matrice $N_1 x N_1,\ 0 \le m_1 \le r_1 - 1$

$$B = \left[X_{[0]} \right] \cdot \begin{bmatrix} 1 \\ W_N^{2m_1} \\ W_N^{3m_1} \\ \vdots \\ \vdots \\ W_N^{(N_1-1)m_1} \end{bmatrix} \quad \text{matrice 1 x } N_1 \quad 0 \le m_1 \le r_1 - 1$$

La diagonale della matrice A contiene i coefficienti di Twiddle del primo stadio. In uscita del primo stadio abbiamo r_1 vettori di lunghezza $N_1 : {}^0x_1, {}^1x_1, {}^2x_1, \cdots, {}^{(r_1-1)}x_1$.

Il secondo stadio ricevera' in ingresso i vettori di uscita del primo stadio ottenendo in uscita al secondo stadio r_2 vettori di lunghezza N_2 per ogni vettore d'ingresso proveniente dal primo stadio, quindi in uscita del secondo stadio abbiamo un totale di $r_1 \cdot r_2$ vettori.

Riportiamo di seguito le matrici relative al secondo stadio:

$$\left[X \right]_1^{m_1} = \begin{bmatrix} {}^{m_1}x_{1_{[0]}} & {}^{m_1}x_{1_{[N_2]}} & {}^{m_1}x_{1_{[N_2]}} & \cdots & \cdots & {}^{m_1}x_{1_{[(r_2-1)N_2]}} \\ {}^{m_1}x_{1_{[1]}} & {}^{m_1}x_{1_{[N_2+1]}} & {}^{m_1}x_{1_{[2N_2-1]}} & \cdots & \cdots & {}^{m_1}x_{1_{[(r_2-1)N_2+1]}} \\ {}^{m_1}x_{1_{[2]}} & {}^{m_1}x_{1_{[N_2+2]}} & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ {}^{m_1}x_{1_{[N_2-1]}} & {}^{m_1}x_{1_{[2N_2-1]}} & \cdots & \cdots & \cdots & {}^{m_1}x_{1_{[r_2N_2-1]}} \end{bmatrix} \quad \text{matrice } N_2 \text{ x } r_2$$

$$ {}^{m_1 m_2}x_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & W_N^{m_2} & 0 & 0 & \cdots & 0 \\ 0 & 0 & W_N^{2m_2} & 0 & \cdots & 0 \\ 0 & 0 & 0 & W_N^{3m_2} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & W_N^{(N_2-1)m_2} \end{bmatrix} \cdot \left[X\right]_1^{m_1} \cdot \begin{bmatrix} 1 \\ W_{r_2}^{m_2} \\ W_{r_2}^{2m_2} \\ W_{r_2}^{3m_2} \\ \vdots \\ W_{r_2}^{(N_2-1)m_2} \end{bmatrix} $$

$$ 0 \le m_1 \le r_1 - 1 \text{ e } 0 \le m_2 \le r_2 - 1 $$

Per il passo t generico abbiamo:

$$ \left[X\right]_{t-1}^{m_1 m_2 .. m_{t-1}} = \begin{bmatrix} {}^{m_1 m_2 .. m_{t-1}}x_{t-1_{[0]}} & {}^{m_1 m_2 .. m_{t-1}}x_{t-1_{[N_2]}} & {}^{m_1 m_2 .. m_{t-1}}x_{t-1_{[N_2]}} & \cdots & \cdots & {}^{m_1 m_2 .. m_{t-1}}x_{t-1_{[(r_2-1)N_2]}} \\ {}^{m_1 m_2 .. m_{t-1}}x_{t-1_{[1]}} & {}^{m_1 m_2 .. m_{t-1}}x_{t-1_{[N_2+1]}} & {}^{m_1 m_2 .. m_{t-1}}x_{t-1_{[2N_2-1]}} & \cdots & \cdots & {}^{m_1 m_2 .. m_{t-1}}x_{t-1_{[(r_2-1)N_2+1]}} \\ {}^{m_1 m_2 .. m_{t-1}}x_{t-1_{[2]}} & {}^{m_1 m_2 .. m_{t-1}}x_{t-1_{[N_2+2]}} & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ {}^{m_1 m_2 .. m_{t-1}}x_{t-1_{[N_2-1]}} & {}^{m_1 m_2 .. m_{t-1}}x_{t-1_{[2N_2-1]}} & \cdots & \cdots & \cdots & {}^{m_1 m_2 .. m_{t-1}}x_{t-1_{[r_2 N_2-1]}} \end{bmatrix} $$

Matrice di dimensione N_1 x r_1

$$ {}^{m_1 m_2 .. m_t}x_t = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & W_{N_{t-1}}^{m_t} & 0 & 0 & \cdots & 0 \\ 0 & 0 & W_{N_{t-1}}^{2m_t} & 0 & \cdots & 0 \\ 0 & 0 & 0 & W_{N_{t-1}}^{3m_t} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & W_{N_{t-1}}^{(N_1-1)m_t} \end{bmatrix} \cdot \left[X\right]_{t-1}^{m_1 m_2 .. m_{t-1}} \cdot \begin{bmatrix} 1 \\ W_{r_t}^{m_t} \\ W_{r_t}^{2m_t} \\ W_{r_t}^{3m_t} \\ \vdots \\ W_{r_t}^{(N_2-1)m_t} \end{bmatrix} $$

$$ 2 \le t \le \nu - 1 \text{ e } 0 \le m_t \le r_t - 1 $$

L'ultimo passo, per t=ν, ci permette di ottenere la sequenza di uscita $X_{(k)}$

$$X_{(k)} = \left({}^{m_1 m_2 \ldots m_{\nu-1}} x_{\nu-1} \right)^T \cdot \begin{bmatrix} 1 \\ W_{r_\nu}^{m_\nu} \\ W_{r_\nu}^{2m_t} \\ W_{r_\nu}^{3m_\nu} \\ \vdots \\ W_{r_\nu}^{(r_t-1)m_\nu} \end{bmatrix}$$

Il valore di k per $0 \le k \le N-1$ dipende da r_t e m_t nel seguente modo:

$k = r_1 r_2 \ldots r_{\nu-1} m_\nu + r_1 r_2 \ldots r_{\nu-2} m_{\nu-1} + \ldots\ldots + r_1 m_2 + m_1$

con $1 \le t \le \nu - 1$ e $0 \le m_t \le r_t - 1$.

La figura seguente rappresenta l'algoritmo radix-4

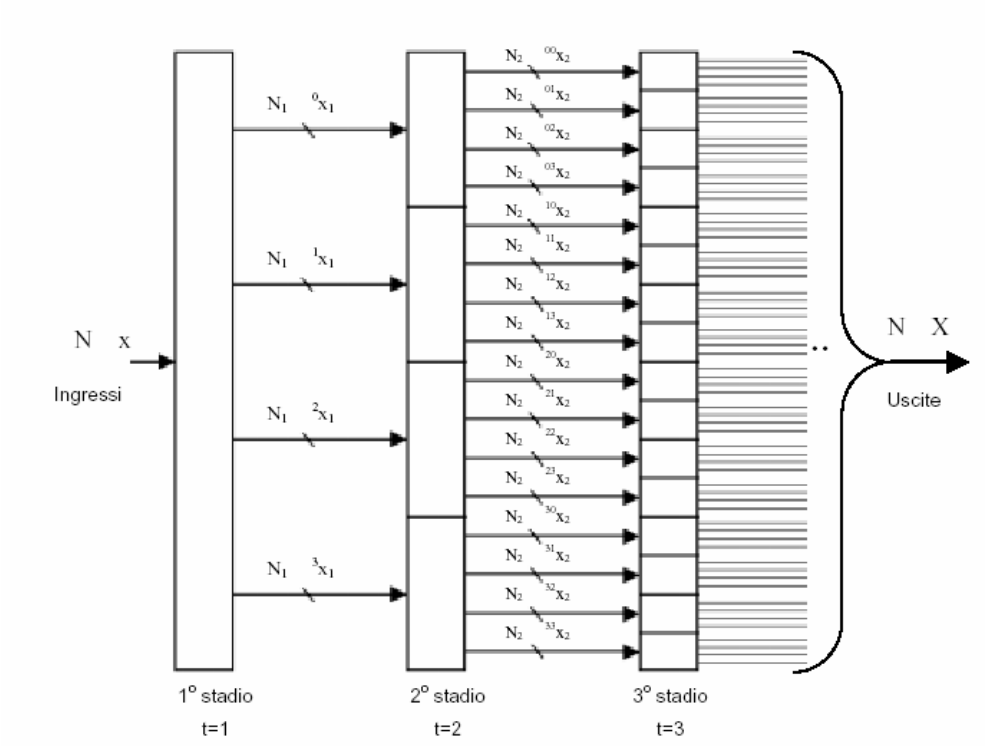


Figura 2.2 : rappresentazione grafica dell'algoritmo radix-4

Con $r_1 = r_2 = ... = r_v = 4$, $2 \le t \le \nu - 1$ e $0 \le m_t \le 3$

Avremo $N_1 = \frac{N}{4}$, $N_2 = \frac{N_1}{4} = \frac{N}{16}$

In generale sappiamo che il numero di stadi necessari e' :

$\nu = \ell og_4 (N)$.

2.3 Descrizione dell'architettura

In questo paragrafo parleremo dell'architettura del processore partendo dalla sua interfaccia esterna rappresentata in figura 2.3 , descrivendo poi ogni singolo blocco che lo costituisce.

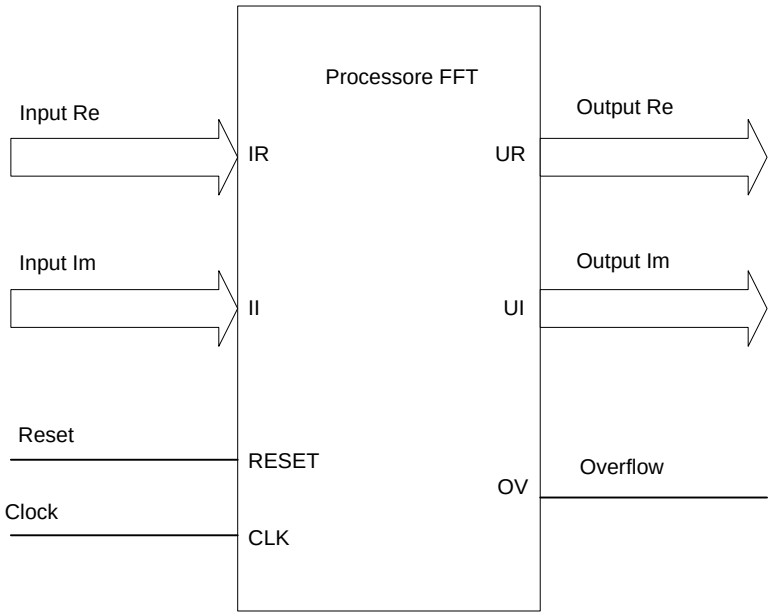


Figura 2.3 : Interfaccia del processore FFT

Gli ingressi IR e II costituiscono rispettivamente la parte reale e la parte immaginaria della sequenza di ingresso mentre CLK e RESET sono i piedini del clock e del reset globale sincrono del processore. In uscita abbiamo UR e UI che sono rispettivamente la parte reale e la parte immaginaria della sequenza di uscita mentre OV segnala eventuali overflow.

I blocchi principali sono:

- Commutatore
- Butterfly
- Moltiplicatore complesso
- Rom dei coefficienti di twiddle
- Unita' di controllo

Lo scema architetturale del processore e' rappresentato in figura 2.4 . Come si vede l'architettura e' di tipo cacscade, il numero di stadi che la costituiscono e' dato da $\nu = \ell og_4(N)$ e sono tutti radix-4 tranne l'ultimo che puo' essere radix-2 nel caso in cui N e' una potenza di 2, in questo caso l'algoritmo usato e' di tipo mixed-radix-4\2.

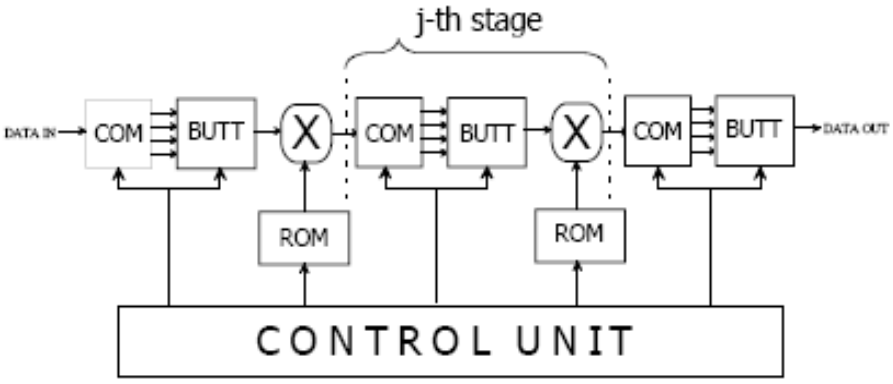


Figura 2.4 : Schema architetturale del processore FFT

2.3.1 Commutatore

Come possiamo osservare dalla figura 2.4 il Commutatore costituisce l'ingresso di ogni singolo stadio, esso prende un dato complesso e fornisce in uscita 2 o 4 dati complessi a seconda se lo stadio di cui fa parte e' di tipo radix-2 o radix-4. In figura 2.5 e' riportato lo schema a blocchi del commutatore radix-2 mentre in figura 1.6 e' riportato lo schema a blocchi del commutatore radix-4.

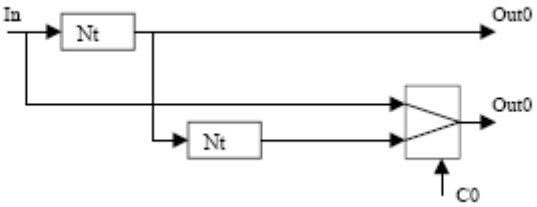


Figura 2.5 : Commutatore radix-2

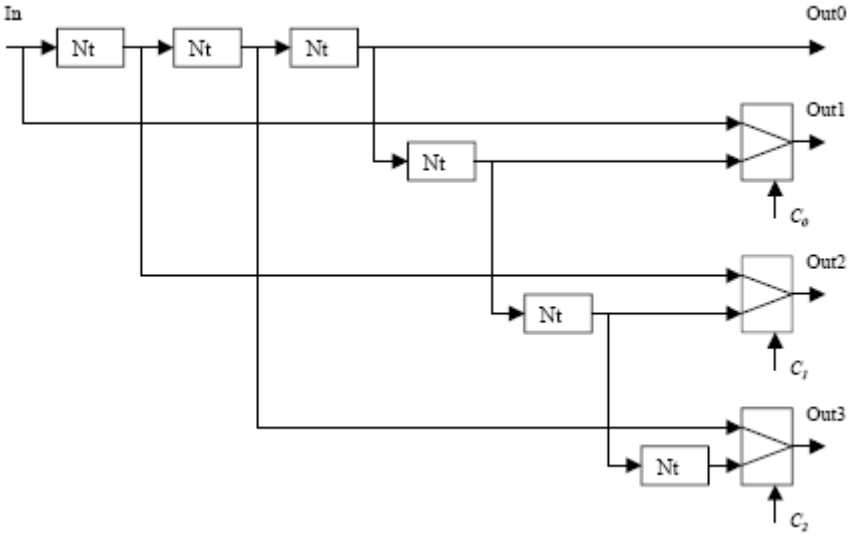


Figura 2.6 : Commutatore radix-4

La funzione principale del commutatore e' quella di riordinare i dati e presentarli con l'ordine esatto e dopo un certo periodo di ritardo alla Butterfly. Le unita' N_t sono shift-register e costituiscono gli elementi di ritardo. I mux sono direttamente comandati dalla Control Unit in maniera ciclica.

2.3.2 Butterfly

Prendiamo in esame una Butterfly radix-4, la sua funzione e' quella di compiere operazioni di somma e sottrazione tra le otto componenti dei 4 vettori complessi che riceve in ingresso, successivamente effettua un arrotondamento del risultato e presenta in uscita la parte reale e la parte immaginaria del vettore risultante che servira' al moltiplicatore complesso a valle.

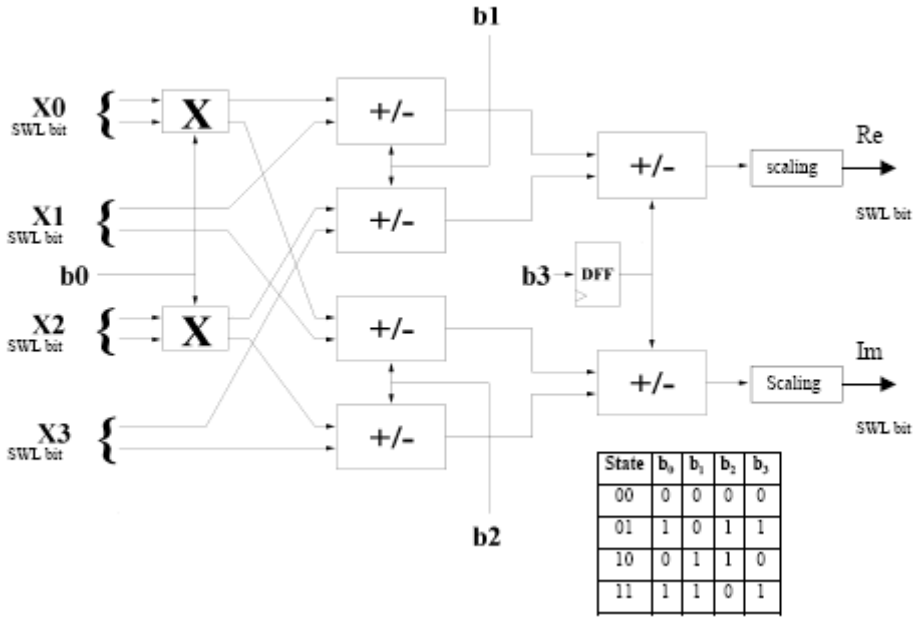


Figura 2.7 : Butterfly radix-4

La figura 2.7 oltre allo schema a blocchi della Butterfly radix-4 mostra anche la tabella di verita' della logica di controllo. I blocchi chiamati X altro non sono che degli Switch.

2.3.3 Moltiplicatore complesso

Il moltiplicatore complesso realizza il prodotto tra il vettore complesso proveniente dalla Butterfly e il coefficiente di Twiddle proveniente dalle Rom. Esso e' costituito da 4 moltiplicatori reali e due sommatori.

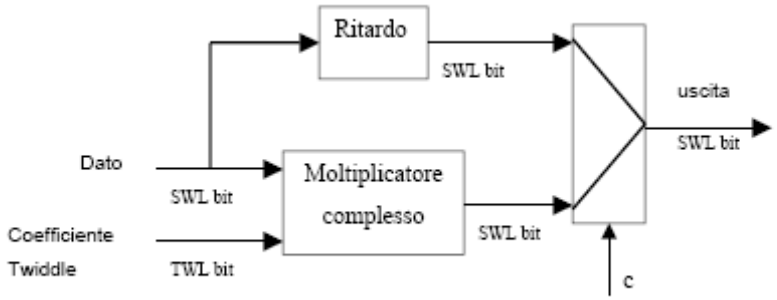


Figura 2.8 : Moltiplicatore complesso

Il blocco di ritardo e il multiplexer sono stati inseriti nel moltiplicatore complesso per evitare la moltiplicazione per il coefficiente di Twiddle con parte immaginaria nulla, in tal modo si evita di introdurre inutilmente nei dati un errore di approssimazione che invece abbiamo in tutte le altre moltiplicazioni con i coefficienti diversi da1.

2.3.4 Rom dei coefficienti di Twiddle

Ogni stadio contiene una ROM le cui dimensioni dipendono dallo stadio stesso nel seguente modo:
Sia t il generico stadio con $0 \le t \le \nu - 1$ e ν numero di stadi

La ROM del t^{esimo} stadio e' costituita da $N_{t-1} = \frac{N}{4^{t-1}}$ celle ciascuna delle quali contiene un coefficiente di Twiddle codificato su TWL bit (Twiddle Word Length), la Rom del t^{esimo} stadio e' quindi costituita da $(\frac{N}{4^{t-1}} \cdot TWL)$ bit. Ogni ROM (tranne la prima) e' ottenuta prelevando un elemento ogni 4 dalla ROM dello stadio precedente, in questo modo abbiamo che la ROM del primo stadio contiene N coefficienti, quella del secondo stadio ne contiene $\frac{N}{4}$, quella del terzo stadio ne contiene $\frac{N}{16}$ e cosi' via.

2.3.5 Control Unit

Il ruolo dell'unita' di controllo e' quello di sincronizzare tra loro i blocchi che costituiscono ciascuno stadio e contemporaneamente tutti gli stadi tra loro. In realta' l'unita' di controllo e' costituita da tante piccole unita' ognuna delle quali controlla in modo indipendente dalle altre il singolo stadio a cui e' dedicata (vedi figura 2.9). La ROM dei coefficienti di Twiddle e' inserita nelle unita' di controllo dello stadio e gli indirizzi vengono generati da un contatore a $\ell og_2\left(\frac{N}{4^{t-1}}\right)$ bit. I due bit piu' significativi del contatore costituiscono il segnale "state" che comanda la logica di controllo della Butterfly e del commutatore (vedi figura 2.6 e 2.7)

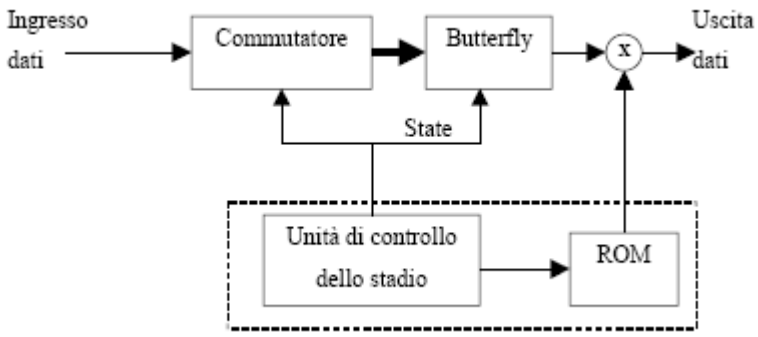


Figura 2.9 : Unita' di controllo di uno stadio generico

Capitolo 3

Simulazione funzionale pre-sintesi

3.1 Simulazione funzionale del processore FFT

Per completare le descrizione del processore FFT riportiamo la simulazione funzionale descrivendo il tipo di verifica e il programma C++ che e' stato usato. Il programma C++ e' in grado di generare due file "test.in" e "test.out" contenenti entrambi dei vettori complessi codificati in binario, il file "test.in" contiene i vettori che verranno forniti dal simulatore in ingresso al processore FFT. In uscita al processore avremo la trasformata della sequenza di ingresso contenuta nel file "test.in" che verra' poi confrontata con i corrispondenti vettori complessi contenuti nel file "test.out" che corrispondono ai vettori trasformati della sequenza contenuta nel file "test.in". il programma C++ genera anche dei file di codice VHDL, che sono il file Parameters che contiene i parametri del codice del processore, il file Rom_pack che e' il file in cui viene generata la Rom la cui grandezza dipende dai parametri del codice, e il file Rom che contiene i coefficienti di Twiddle. In figura 2.1 e' riportato lo schema di verifica del processore, la variabile boleana OK viene generata dal confronto tra i data in uscita del processore con i dati contenuti nel file "test.out" e restituisce il valore "true" in corrispondenza dei due valori.

Il software usato per la simulazione e' Active-HDL 4.2, in figura 2.2 e' riportata la pagina iniziale della simulazione, il Clock generato ha una frequenza di 50 MHz, il processore viene inizializzato con un reset al primo ciclo di Clock.

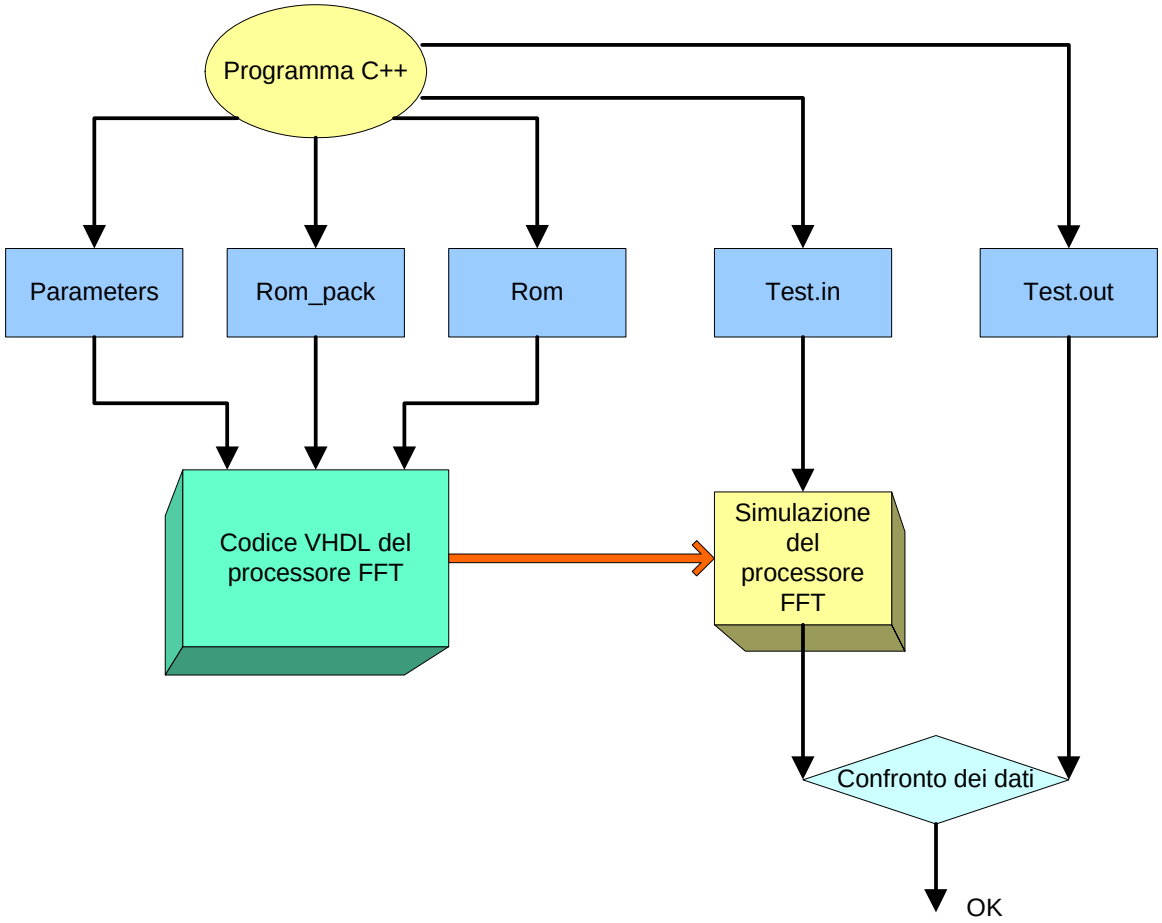


Figura 3.1 : Schema della simulazione funzionale del processore FFT

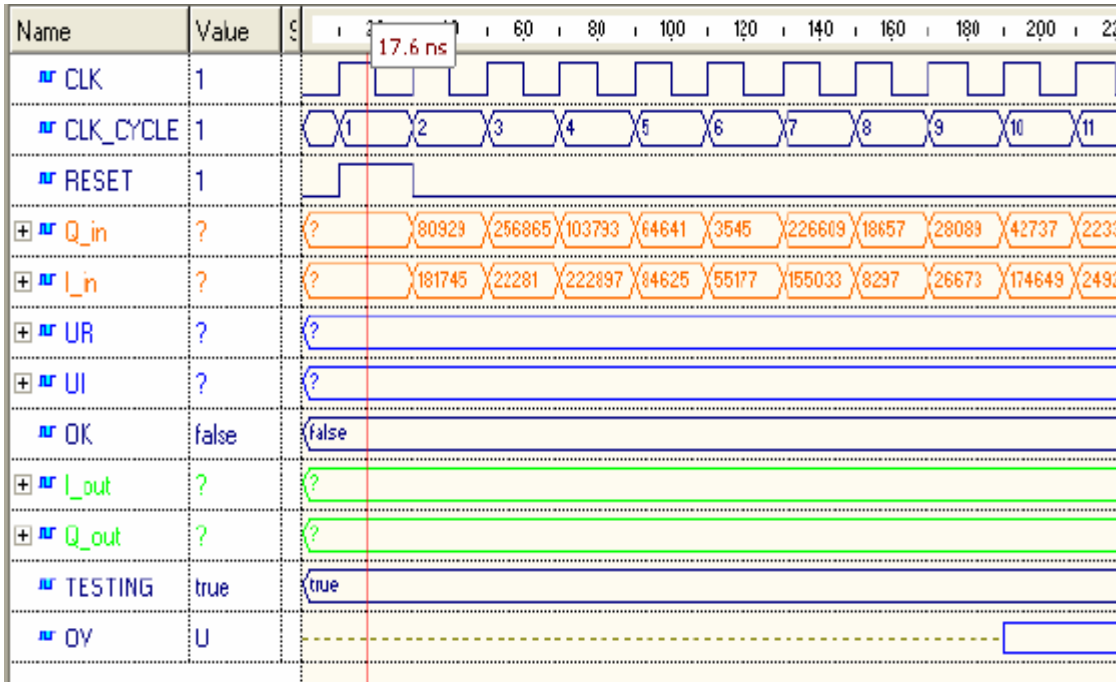


Figura 3.2 : Simulazione funzionale del processore FFT
(Reset del processore)

La variabile CLK_CYCLE conta i cicli di Clock, in questo modo possiamo quantificare la latenza del processore. Le variabili Q_in e I_in sono i dati letti dal file test.in e vengono presentati in ingresso al processore e sono rispettivamente parte reale e parte immaginaria dal dato. Le variabili UR e UI sono parte reale e parte immaginaria della uscite del processore mentre I_out e Q_out sono i dati letti dal file test.out che verranno confrontati con le uscite del processore. La variabile OV e' un'uscita del processore che assume valore OV=1 se si e' verificato un overflow, notiamo che in figura 3.2 OV non ha alcun valore perche' non ci sono dati in uscita. La variabile buleana TESTING assume il valore "true" quando inizia la simulazione e diventa "false" non appena l'ultimo dato del processore e' stato presentato in uscita ed e' stato effettuato il confronto. La variabile OK ha il valore "false" che sta ad indicare che ancore non risono dati in uscita dal processore da confrontare.

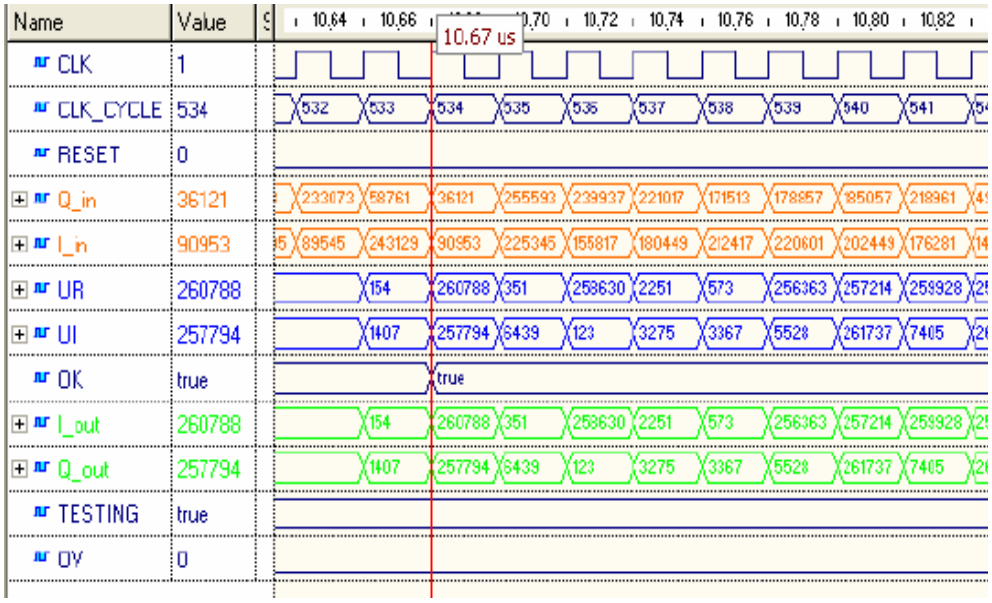


Figura 3.3 : Arrivo dei dati in uscita del processore e inizio del confronto.

Nella figura 3.2 viene mostrato l'inizio del confronto dei dati del processore descritto in VHDL con i dati del modello C++. La variabile OK assume il valore "true" che indica che i dati del processore sono corrispondenti con i dati del file test.out. Altra cosa molto importante che possiamo dedurre e' la latenza del processore che e' data dalla variabile CLK_CYCLE=532.

Capitolo 4

Sintesi del processore FFT con aritmetica BFP

INTRODUZIONE

In questo capitolo verrà trattata la sintesi del processore FFT con aritmetica BFP su più dispositivi FPGA. L'obiettivo che ci siamo prefissati è quello di valutare secondo la metrica dei System Gate una funzione logica tramite la quale si possa quantificare, mediante sintesi logica, le risorse logiche utilizzate in un dispositivo FPGA. Il motivo che ha portato alla scelta del processore FFT come funzione utile al nostro scopo è proprio la natura di tipo parametrica del codice VHDL tramite il quale è stato descritto, infatti come evidenziato nel primo capitolo, variando i suoi parametri varia la quantità di risorse logiche necessarie per la mappatura su un dispositivo programmabile,mantenendo inalterata l'architettura. Nella Figura 4.1 è riportato il file Parameters del codice VHDL che viene generato insieme ad altri file da un programma C++ che ci permette di variare i parametri che caratterizzano il processore. Il primo passo è quello di effettuare una serie di sintesi logiche su uno stesso dispositivo variando di volta in volta i parametri in modo da poter osservare ad ogni sintesi la quantità di risorse logiche utilizzate ed estrapolare una funzione che possa descriverne l'andamento.

```
library IEEE;
package PARAMETERS  is
  constant N : integer :=256;  -- Number of samples.
  constant NST : integer :=4;  -- Number of stages.
  constant SWL : integer :=19;  -- System  Word Length
  constant TWL : integer :=14;  -- Twiddle Word Length
  constant OWL : integer :=18;  -- Output  Word Length
  constant DWL : integer :=18;  -- Data Word Length
  constant RSH : integer :=1;  -- Right Shifts
  constant MIX : integer :=0;  -- Mixed Radix
  constant FFT : integer :=1;
end PARAMETERS;
```

Figura 4.1: Parametri del processore FFT.

4.1 Sintesi del Processore FFT-BFP sul dispositivo FPGA ALTERA STRATIX

La sintesi logica del processore FFT con aritmetica BFP è stata effettuata con il tool Synplify Pro 8.1 prodotto da SYNPLICITY. Il primo dispositivo che è stato scelto per la sintesi è un FPGA ALTERA serie STRATIX EP1S80 che come vedremo è in grado di contenere il processore in questione. I vincoli che sono stati imposti in fase di ottimizzazione sono:

- Segnale di clock di 20ns corrispondente ad una frequenza di 50 Mhz
- Minima area occupate.

Nella tabella 4.2 sono riportati i dati relativi alle sintesi logiche del processore FFT con aritmetica BFP, il tool di sintesi ci fornirà dati dettagliati ma concentreremo la nostra attenzione sulle risorse di logica combinatoria che sono state mappate sul dispositivo FPGA, tralasceremo quindi il numero di celle di memoria occupate per il semplice motivo che non è necessario quantificarle in quanto la dimensione di memoria di ogni dispositivo FPGA è riportata sui Data Sheet forniti dal produttore. Daremo quindi particolare importanza al numero di risorse logiche usate che come descritto nel Capitolo 1 vengono mappate nel dispositivo attraverso le LUT (LOOK-UP TABLE).

SWL	TWL	OWL	DWL	LUTs usati
21	17	20	20	81742
19	14	18	18	71961
17	12	16	16	63510
14	10	13	13	51510
10	10	9	9	36396

Tabella 4.2 : Risultati della sintesi del processore FFT-BFP al variare dei valori SWL, TWL, OWL e DWL.

La tabella 4.2 si riferisce ad un processore in cui N è stato fissato 256 e il numero di stadi NST pari a quattro.

Quello che dobbiamo fare a questo punto è trovare una funzione caratteristica in grado di descrivere la quantità logica usata al variare dei parametri, a tal proposito dobbiamo fare alcune considerazioni sui parametri per capire quali tra questi influisce maggiormente sulla logica occupata nel dispositivo. Il parametro TWL (Twiddle Word Lengh) determina la lunghezza dei coefficienti di Twiddle che andranno poi inseriti in una ROM,

e mandati in ingresso al moltiplicatore complesso per essere moltiplicati per SWL (vedi capitolo 1) che è il parametro dominante rispetto gli altri in quanto determina i Data Path interni . Per quanto riguarda i coefficienti OWL e DWL, che assumono lo stesso valore, sono legati al valore di SWL secondo la relazione:

$SWL_{minima} = OWL + RSH$ ponendo quindi RSH = 1

Otteniamo:

$SWL_{minima} = OWL + 1.$

Come notiamo dalla tabella 4.2 è stata presa sempre una SWL_{minima} questo perché una piccola variazione di questo parametro determina una grande variazione della quantità di risorse logiche necessarie affinché il processore possa essere mappato. Nella Figura 4.2 vengono rappresentati i LUTs usati al variare di SWL.

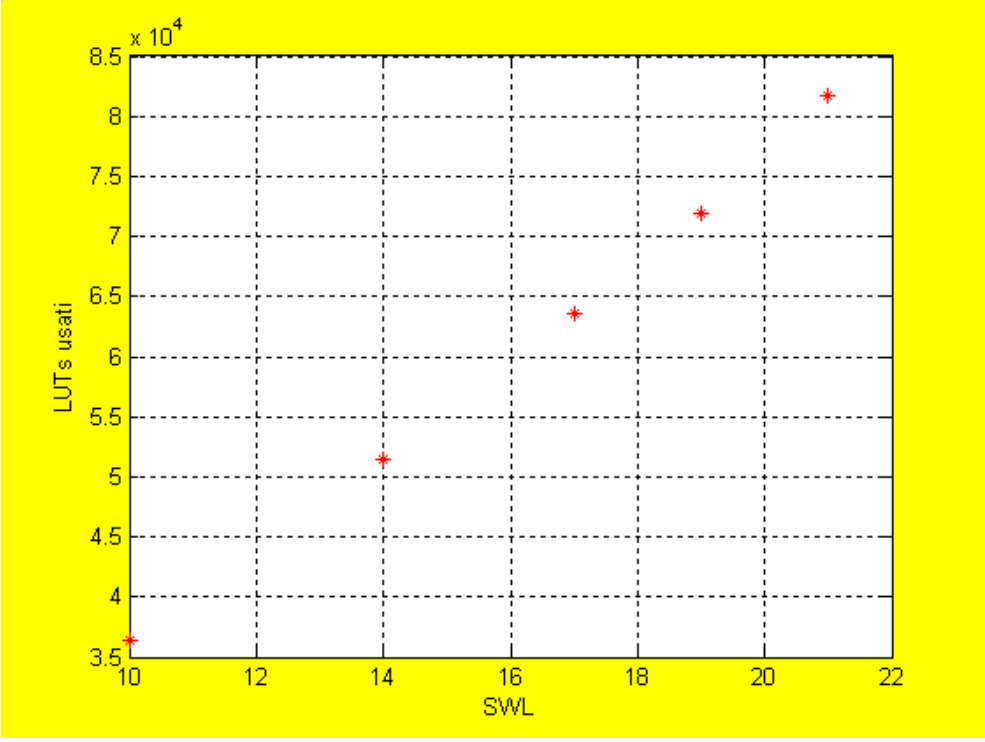


Figura 4.2 : Rappresentazione grafica dei LUTs usati.

Costruiamo il polinomio di interpolazione[1] sui punti:

SWL=X	LUTs
10	36396
14	51510
17	63510
19	71961
21	81742

Otteniamo[2]:

$LUT(X) = 1.4374\ X^4 - 84.742\ X^3 + 1877.8\ X^2 - 14549\ X + 64484$

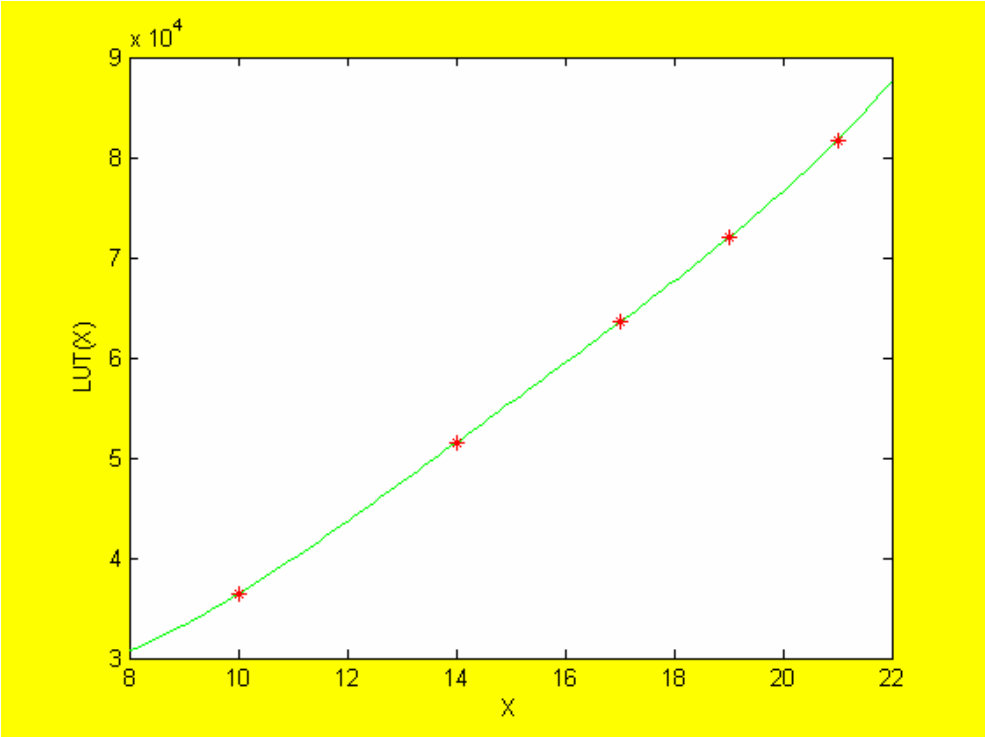


Figura 4.3 : Polinomio di interpolazione

L'andamento del polinomio interpolante è mostrato nella Figura 4.3.

[1] Tutte le elaborazioni e i grafici sono stati effettuati con Matlab

[2] Preso N come il numero di punti da interpolare e n il grado del polinomio interpolante, se n=N-1, il polinomio di interpolazione soddisfa esattamente le condizioni di interpolazione, se invece n<N-1 allora Matlab interpolerà nel senso dei minimi quadrati.

L'aver calcolato il polinomio di interpolazione per i punti considerati non ci assicura che vada bene per ogni X nell'intervallo considerato. Sono state effettuate le sintesi nei punti intermedi di ogni tratto della caratteristica per semplicità ne riportiamo solo i risultati del primo tratto della curva per X Є [10; 14], negli altri tratti di curva abbiamo ottenuto gli stessi risultati.

SWL	LUTs
10	36396
11	40119
12	43932
13	47641
14	51510

Nella Figura 4.4 riportiamo i punti ottenuti tramite sintesi logica:

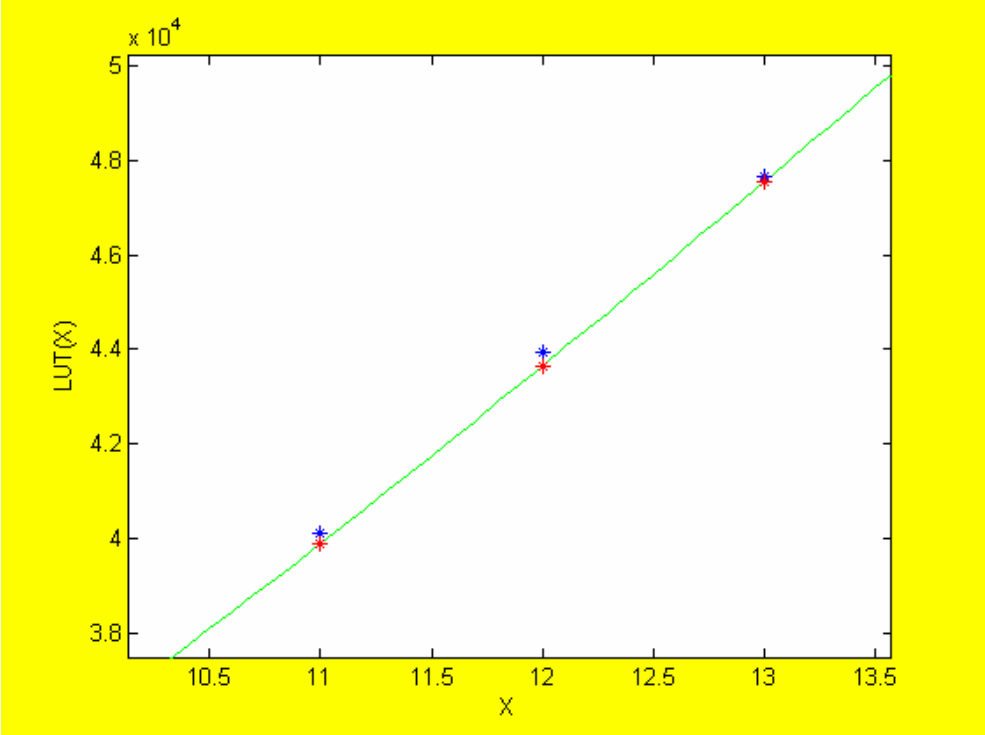


Grafico 4.4 : valori intermedi

Gli atsrerischi di colore blu si riferiscono al valore misurato, mentre quelli rossi si riferiscono al valore previsto.

Nella tabella 4.3 sono riportati :

- F(X) valore ottenuto mediante sintesi logica
- P(X) valore del polinomio di interpolazione in quel punto
- Є scarto tra valore reale e valore previsto

X	F(X)	P(X)	Є
11	40119	39899	220 (0.54%)
12	43932	43655	277 (0.63%)
13	47641	47551	90 (0.19%)

Tabella 4.3 : Valore reale, valore stimato e scarto.

La percentuale riportata nella tabella 4.3 è riferita al valore reale e come possiamo notare è nettamente inferiore all'1%[3], che può ritenersi contenuto, quindi possiamo dire che P(x) descrive in maniera soddisfacente la funzione F(X).

Un'altra verifica che possiamo fare è mantenere invariati i valori di DWL, OWL e TWL e vedere come variano le risorse logiche usate al variare di SWL.

X	LUTs
10	36396
11	37148
12	37866
13	38588
14	39331

Tabella 4.4 : Sintesi logiche effettuate per SWL non minima (SWL>OWL+RSH).

[3] Lo stesso risultato è stato ottenuto negli altri tratti della curva.

La Tabella 4.4 si riferisce alle sintesi effettuate con i seguenti parametri:

- OWL=DWL=9
- TWL=10
- X=SWL =[10, 11, 12, 13, 14]

Riportiamo in un grafico tali risultati.

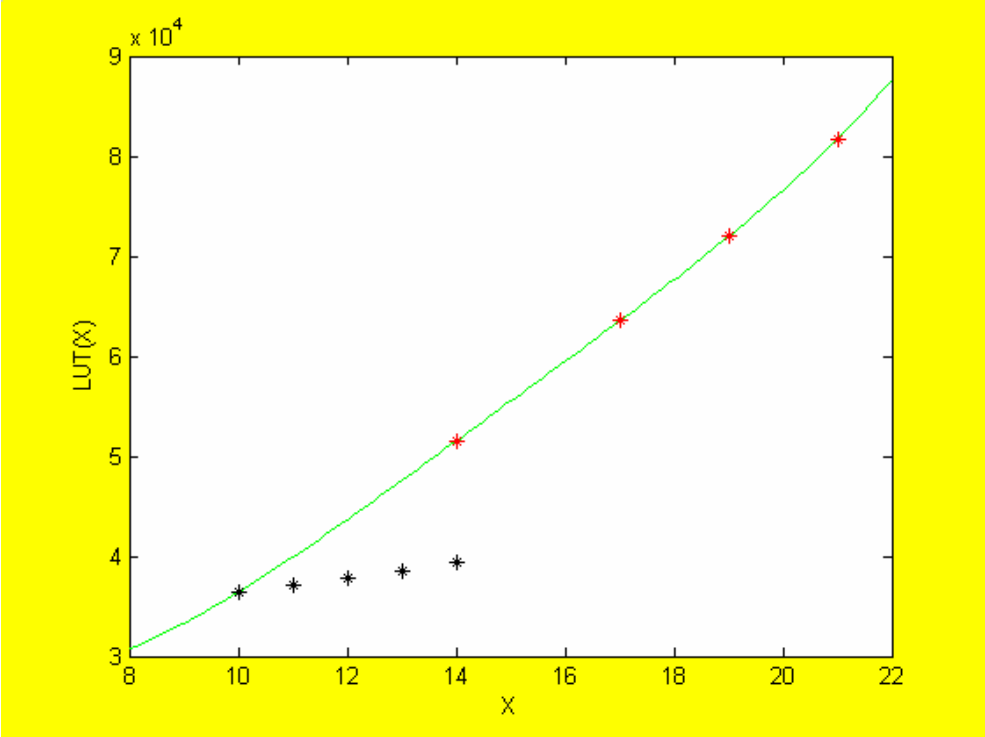


Figura 4.5 : valori misurati con SWL non minima

I punti di colore nero si riferiscono ad una SWL non minima, mentre la curva di colore verde è la rappresentazione grafica di P(X).

Come possiamo vedere i valori trovati con SWL non minima sono molto contenuti, una funzione calcolata con questi punti avrebbe una pendenza troppo piccola per poter coprire un accettabile range di osservazione

4.2 Sintesi del Processore FFT-BFP sul dispositivo FPGA XILINX SPARTAN-3

La sintesi logica del processore FFT con aritmetica BFP viene ora effettuata sul dispositivo XILINX SPARTAN-3, il tool di sintesi è sempre il Synplify Pro 8.1 prodotto da SYNPLICITY e i vincoli imposto in fase di ottimizzazione sono:

- Segnale di clock di 20ns corrispondente ad una frequenza di 50 Mhz
- Minima area occupata.

I risultati delle sintesi logiche sono riportate nella Tabella 4.5, questa volta però grazie alla flessibilità del dispositivo in uso il tool di sintesi ci fornisce dati dettagliati relativi all'uso dei LOOK-UP TABLE, infatti vengono riportati i LUT_{totali} , e poi anche LUT1, LUT2, LUT3 e i LUT4. Possiamo pensare ai LUT1, LUT2 e LUT3 come parti di LUT4 che contengono logica e considerarli rispettivamente come:

- LUT1 = ¼ LUT4;
- LUT2 = ½ LUT4;
- LUT3 = ¾ LUT4.

SWL	TWL	DWL OWL	TOTAL LUT	PARTIAL LUT	
21	17	20	35173	**LUT1**	567
				LUT2	2312
				LUT3	23032
				LUT4	9131
19	14	18	30140	**LUT1**	529
				LUT2	1877
				LUT3	17688
				LUT4	9491
17	12	16	26864	**LUT1**	492
				LUT2	1515
				LUT3	16151
				LUT4	8275
14	10	13	21209	**LUT1**	457
				LUT2	1042
				LUT3	14034
				LUT4	5317
10	10	9	15556	**LUT1**	357
				LUT2	843
				LUT3	9416
				LUT4	4581

Tabella 4.5 : Sintesi del processore FFT-BFP sul dispositivo FPGA SPARTAN-3.

Supponiamo di dividere i LUT in due tipologie:

1) LUT fisici
2) LUT logici

I LUT fisici sono i LUT che abbiamo effettivamente usato per mappare il Processore FFT sul dispositivo SPARTAN-3, mentre i LUT logici sono la parte di LUT fisici che contengono logica, cioè sono il numero di LUT4 che avrebbero potuto contenere la logica necessaria a mappare il Processore FFT sul dispositivo SPARTAN-3 se tutti i LUT fossero stati riempiti completamente.
Il numero di LUT logici sarà dato da:

$$LUT_{logici} = ¼ (LUT1 + 2 * LUT2 + 3 * LUT3) + LUT4. \quad (4.1)$$

I LUT_{fisici} sono i LUT totali che sono stati mappati dal tool di sintesi e sono riportati nella Tabella 4.4.
Usando la (4.1) e la Tabella 4.4 possiamo scrivere la Tabella 4.5 dove vengono riportati i dati relativi alle sintesi logiche del Processore FFT con quella particolare configurazione dei suoi parametri.
Nella Tabella 4.5 vengono riportati anche i LUT_{waste} che si riferiscono al numero di LUT logici che sono stati sprecati nella sintesi logica.

$$LUT_{waste} = LUT_{fisici} - LUT_{logici} \quad (4.2)$$

SWL	LUT_{fisici}	LUT_{logici}	LUT_{waste}
21	35173 (52%)	27703	7470
19	30140 (45%)	23828	6312
17	26864 (40%)	21269	5595
14	21209 (31%)	16478	4731
10	15556 (21 %)	12154	3402

Tabella 4.5 : Anali dei report delle sintesi del Processore FFT-BFP

La Figura 4.6 riporta le risorse usate nel dispositivo SPARTAN-3 XC3S5000 per SWL = 21.

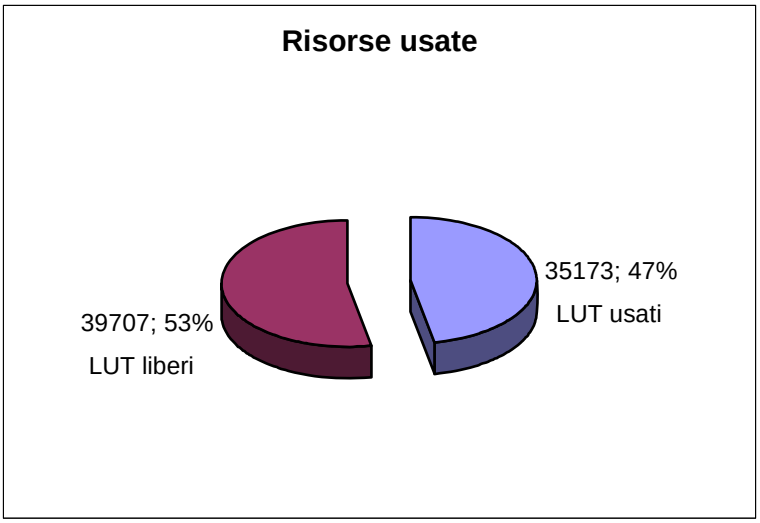


Figura 4.6 : Risorse usate per SWL =21.

Ricordiamo che il dispositivo XILINX SPARTAN-3 XC3S500 possiede 74880 Elementi Logici (LE) e che ogni Elemento Logico possiede a sua volta un LUT4 (vedi capitolo 1).

La Figura 4.7 ci mostra la ripartizione dei LUT_{fisici} in LUT_{logici} e LUT_{waste}.

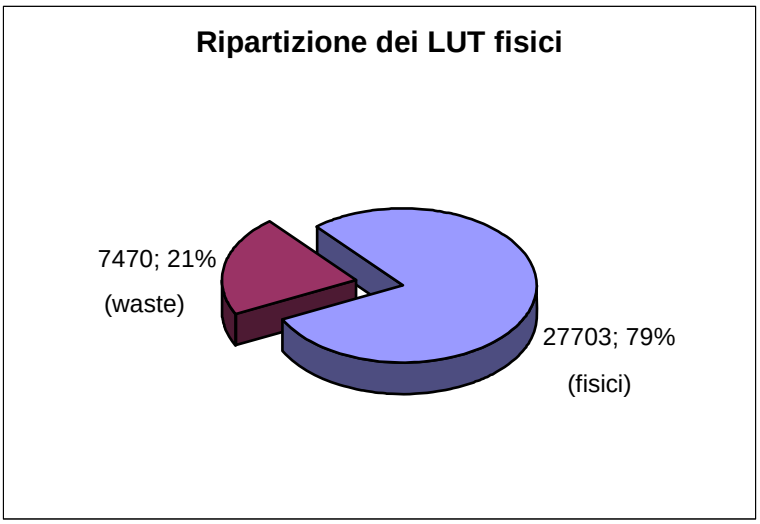


Figura 4.7 : Ripartizione dei LUT_{fisici} per SWL = 21.

Calcoliamo ora la caratteristica di occupazione delle risorse logiche del Processore FFT-BFP nel dispositivo che stiamo considerando.

Nella Figura 4.8 sono riportati i dati della Tabella 4.5 riguardanti i LUT_{fisici} (punti di colore rosso) e i LUT_{logici} (punti di colore blu). Nella Figura 4.9 sono riportati i grafici dei polinomi di interpolazione che descrivono l'andamento dei LUT_{fisici} e dei LUT_{logici} che chiameremo rispettivamente LUT_{logici} (X) (caratteristice di colre verde) e LUT_{fisici}(X) (caratteristica di colore nero) dove è stato posto per semplicità SWL = X .

La differenza delle due caratteristiche rappresenta la caratteristica dei LUT_{waste} al variare di X.

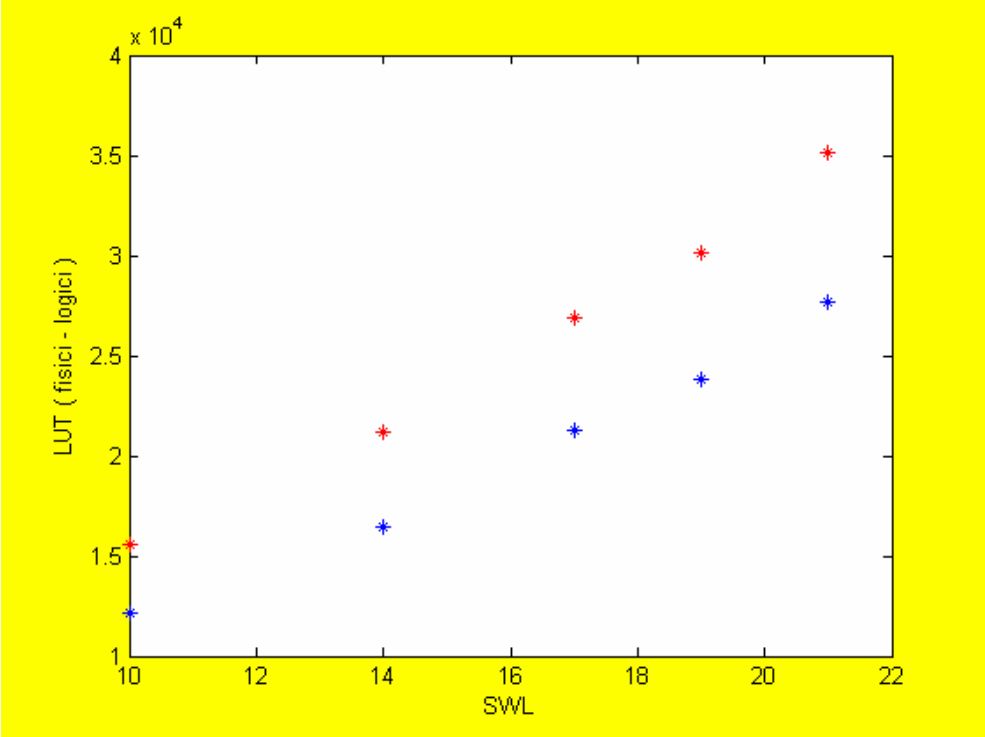


Figura 4.8 : Rappresentazione grafica dei LUT_{fisici} e dei LUT_{logici}.

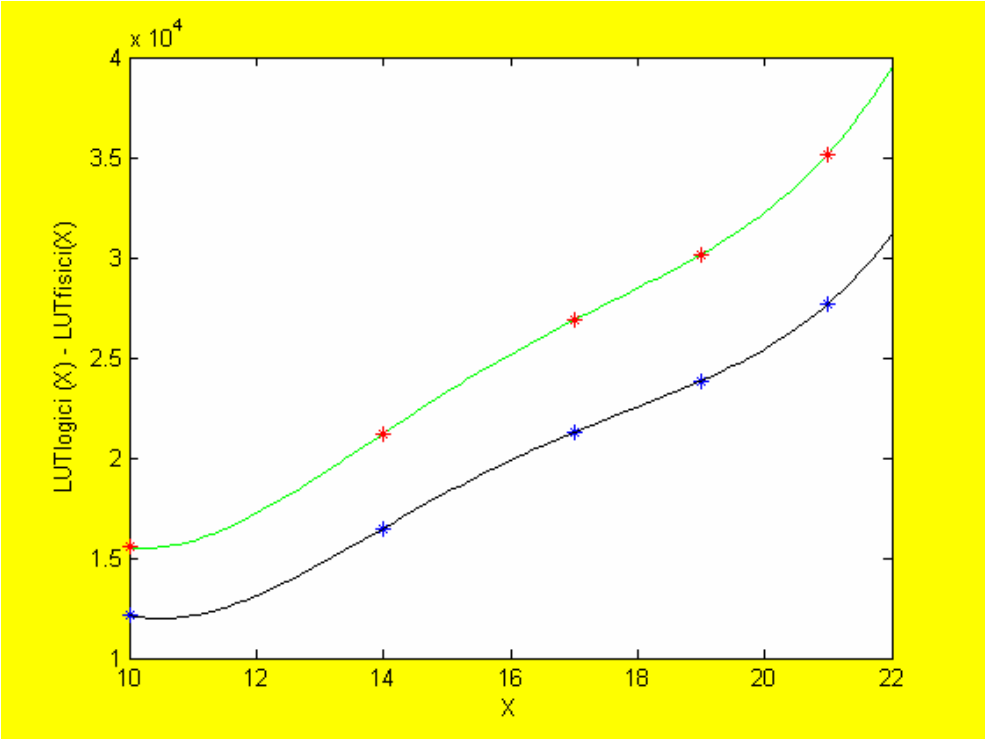


Figura 4.9 : Polinomio di interpolazione $LUT_{fisici}(X)$ e $LUT_{logici}(X)$.

La caratteristica di colore verde si riferisce a:

$$LUT_{ficisi}(X) = 4{,}6736\ X^4 - 293{,}39\ X^3 + 6801{,}3\ X^2 - 67100\ X + 2{,}5308 * 10^5$$

La caratteristica di colore nero si riferisce a:

$$LUT_{logici}(X) = 4{,}347\ X^4 - 276{,}07\ X^3 + 6467{,}3\ X^2 - 64650\ X + 2{,}4452 * 10^5.$$

Capitolo 5

Raccolta dei dati e conclusioni

In questo capitolo verrà caratterizzato completamente, dal punto di vista dei Gate di Sistema, il Processore FFT con aritmetica BFP. Caratterizzare completamente vuol dire conoscere il valore della funzione in termini di Gate di Sistema a prescindere dal dispositivo su cui essa verrà in futuro sintetizzata, anzi proprio tramite sintesi logiche del Processore FFT sui dispositivi FPGA potremmo conoscere i Gate di Sistema Equivalenti del dispositivo relativi alle risorse logiche, una quantità che spesso non viene menzionata nei Data Sheet dei dispositivi o se anche così fosse potrebbe risultare non veritiera e del tutto inaffidabile.

5.1 Raccolta dei dati

Riportiamo i dati raccolti nei Capitoli 1 e 4 relativi alle sintesi logiche distinguendoli in base al dispositivo usato. Chiameremo i dati relativi al dispositivo FPGA ALTERA STRATIX EP1S80 “Dati Stratix”, ricordiamo inoltre che tale dispositivo possiede 79040 Elementi Logici e quindi altrettanti LUT a quattro ingressi e una uscita.
Chiameremo i dati relativi al dispositivo FPGA XILINX SPARTAN-3 XC3S500 “Dati Spartan-3” anche per questo dispositivo ricordiamo che possiede 74880 Elementi Logici e altrettanti LUT a quattro ingressi e una uscita. Un primo paragone che possiamo fare è sulla loro architettura degli Elementi Logici che può sembrare somigliante (vedi Capitolo1), hanno la stessa tipologia di LOOK-UP TABLE e il loro numero è dello stesso ordine di grandezza.

5.1.1 Dati Stratix

N	M	LUTs usati	GATE
100	1	401	403
300	3	3603	3609
400	4	6404	6412
600	5	12005	12015

Tabella 5.1 : Sintesi della funzione parametrica a porte NAND

Da questa tabella deduciamo che il numero di Gate di Sistema mappati per ogni LUT è circa 1 Gate/Lut. Dobbiamo inoltre dire che non è stato possibile distinguere i LUT usati in LUT_{logici} e LUT_{fisici} perché il dispositivo permette la ripartizione dei LUT solo se gli Elementi Logici vengono usati in ARITMETIC MODE[1] e non per la semplice implementazione di logica. Mentre abbiamo trovato che il massimo che un LUT può contenere è 9 Gate che si riferisce al Caso C) nel Capitolo 1 paragrafo 1.1.

Funzione	**GATE/LUT**
Parametrica a porte NAND	1
Caso A)	1.4
Caso B)	3
Caso C)	9
Caso E)	6.33
Caso F)	5
Media	4.28

Tabella 5.2 : Capacità dei LUT del dispositivo STRATIX[2]

[1] Vedi Data Sheet del dispositivo nel sito www.altera.com
[2] Vedi Capitolo 1 paragrafo 1.1

La Tabella 5.2 riporta i GATE/LUT minimi, massimi e la media aritmetica tra le due quantità.
La sintesi del Processore FFT ha prodotto i dati riportati nella Tabella 5.3 dove sono stati riportati i soli $LUT_{fisici.}$

SWL=X	LUT_{fisici}
10	36396
14	51510
17	63510
19	71961
21	81742

Tabella 5.3 : sintesi del Processore FFT-BFP su STRATIX

La caratteristica di occupazione del Processore FFT-BFP è :

$$LUT\_STR(X) = 1.4374\ X^4 - 84.742\ X^3 + 1877.8\ X^2 - 14549\ X + 64484$$

Il suo andamento e riportato in Figura 5.1

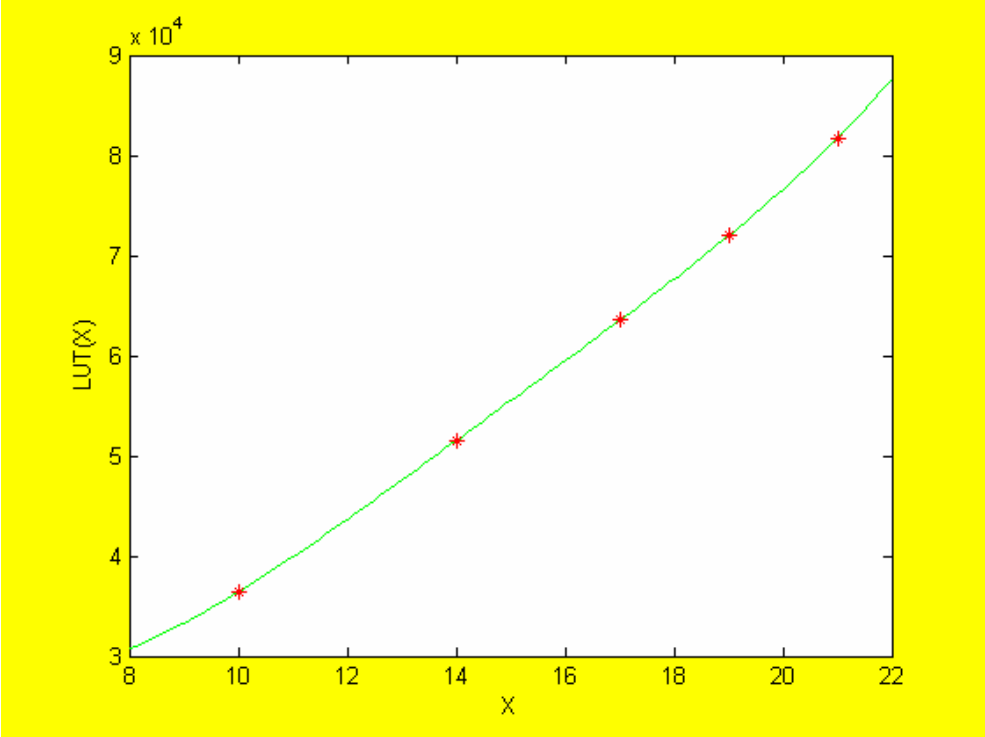


Figura 5.1 : Caratteristica di occupazione del Processore FFT su STRATIX.

Le informazioni che abbiamo raccolto dalle sintesi effettuate sul dispositivo STRATIX purtroppo non sono così dettagliate da poter essere usate per caratterizzare completamente il Processore FFT , infatti conosciamo con sicurezza la caratteristica di occupazione del Processore in termini di LUT sul dispositivo STRATIX, ma pensando ai LUT come un contenitore di logica, non possiamo dire fino a che punto tale contenitore sia stato riempito.

Usando la media dei GATE/LUT riportati nella Tabella 5.2 otteniamo:[3]

$$SG\_FFT\_STR(X) = \left(\frac{GATE}{LUT}\right)_{MEDI} LUT\_STR(X) \qquad (5.1)$$

[3] SG_FFT_STR(X) è la funzione SYSTEM GATE equivalenti del PROCESSORE FFT sul dispositivo STRATIX

5.1.2 Dati SPARTAN-3

La sintesi sul dispositivo SPARTAN-3 ci ha fornito molte informazioni che possono essere usate per caratterizzare in maniera completa il Processore FFT. Costruiamo una tabella usando i dati del Capitolo1.

FUNZIONE	**GATE/LUT$_{logici}$**	**GATE/LUT$_{fisici}$**
Parametrica a porte NAND	2	1
Caso A)	2.33	1.4
Caso B)	5	3
Caso C)	9	9
Caso E)	7.6	6.33
Caso F)	6.67	5

Tabella 5.4 : Capacità dei LUT del dispositivo SPARTAN-3

Facendo un primo confronto con la Tabella 5.2 ci accorgiamo immediatamente che i GATE/LUT$_{fisici}$ coincidono, mentre i GATE/LUT$_{logici}$, questa volta riferendoci al solo dispositivo SPARTAN-3, sono nettamente maggiori, questo perché abbiamo considerato che i LUT$_{logici}$ fossero la parte dei LUT$_{fisici}$ che contengono logica (vedi Capitolo 4).
Nella Tabella 5.5 sono riportati i dati relativi alle sintesi logiche del Processore FFT-BFP sul dispositivo SPARTAN-3.

X	**LUT$_{fisici}$**	**LUT$_{logici}$**	**LUT$_{waste}$**
10	15556 (21 %)	12154	3402
14	21209 (31%)	16478	4731
17	26864 (40%)	21269	5595
19	30140 (45%)	23828	6312
21	35173 (52%)	27703	7470

Tabella 5.5 : sintesi del Processore FFT sul dispositivo SPARTAN-3

LA Tabella 5.6 riporta i LUT_{fisici} delle sintesi logiche nei due dispositivi e il loro rapporto.

X	LUT_{fisici} SPARTAN-3	LUT_{fisici} STRATIX	STRATIX / SPARTAN-3
10	15556	36396	2.34
14	21209	51510	2.43
17	26864	63510	2.36
19	30140	71961	2.39
21	35173	81742	2.32

Tabella 5.6 : Confronto tra le sintesi del Processore FFT.

Il dispositivo SPARTAN-3 usa circa il 60% in meno dei LUT rispetto ai LUT usati nel dispositivo STRATIX, ma entrambe i dispositivi possono contenere al massimo 9 GATE/LUT. Possiamo quindi affermare che il dispositivo SPARTAN-3 ha una flessibilità maggiore del dispositivo STRATIX nel mappare la logica .

In figura 5.2 è riportato l'andamento delle caratteristiche di occupazione dei LUT_{ficisi} (caratteristica di colore verde) e dei LUT_{logici} (caratteristica di colore nero) descritte dalle equazioni (5.2) e (5.3).

$$LUT_{ficisi}(X) = 4{,}6736\ X^4 - 293{,}39\ X^3 + 6801{,}3\ X^2 - 67100\ X + 2{,}5308 * 10^5 \quad (5.2)$$

$$LUT_{logici}(X) = 4{,}347\ X^4 - 276{,}07\ X^3 + 6467{,}3\ X^2 - 64650\ X + 2{,}4452 * 10^5 \quad (5.3)$$

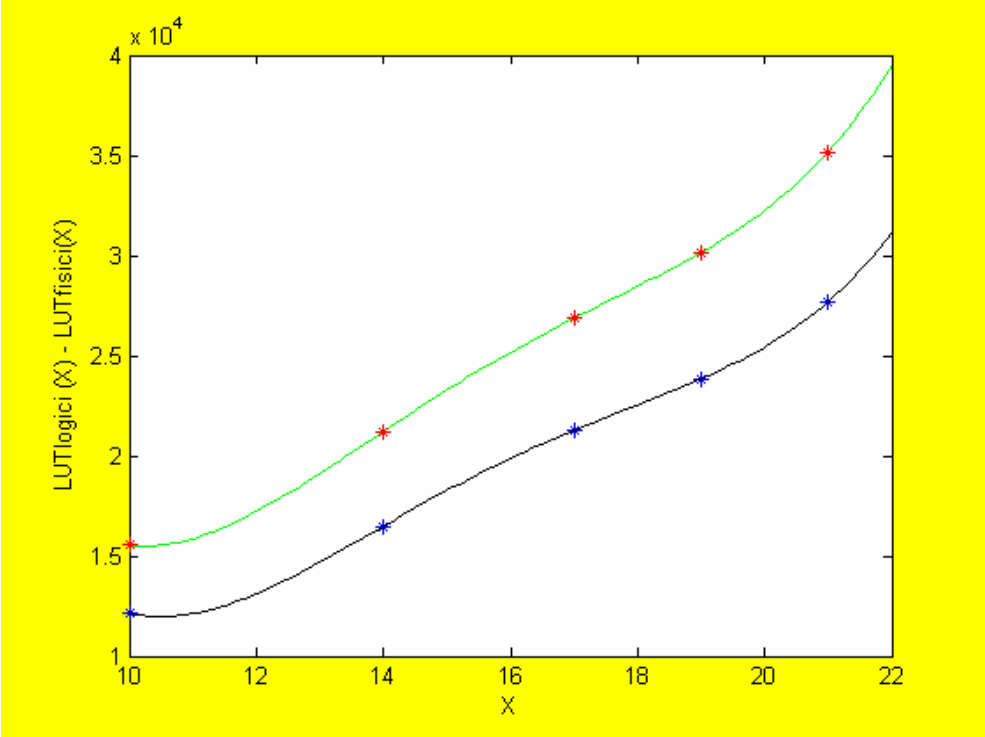


Figura 5.2 : Caratteristica di occupazione del Processore FFT sul dispositivo SPARTAN-3.

Chiamiamo Ω il numero di GATE/LUT del dispositivo SPARTAN-3 otteniamo:

$$GATE\_FFT(X) = \Omega * LUT_{logici}(X) \qquad (5.4)$$

$$X \in N\ [10, 21]$$

Ponendo $\Omega = 9$ nell'equazione (5.4) riusciamo a calcolare il numero di GATE di SISTEMA del dispositivo SPARTAN-3.

Infatti ponendo X=21 dalla Tabella 5.5 abbiamo che i LUT_{fisici}=35173 pari al 52% che è la percentuale di spazio che il processore FFT ha effettivamente occupato nel dispositivo STRATIX, mentre i LUT_{logici}=27703 che moltiplicati per Ω otteniamo i GATE di SISTEMA del Processore FFT per X=21.
Riassumendo :

Ω = 9 Gate/Lut massimi

$LUT_{fisici}(21)$=35173 pari al 52% dei LUT totali del dispositivo SPARTAN-3

$LUT_{logici}(21)$ =27703

SYSTEM-GATE
SPARTAN-3 = Ω * $LUT_{logici}(21)$ / 0.52 = 479475

È questo il risultato a cui volevamo arrivare[4]

[4] I GATE di SISTEMA che abbiamo calcolato si riferiscono solo alla logica infatti non abbiamo considerato la memoria inclusa nel dispositivo.

5.2 Conclusioni ed esempi

A questo punto abbiamo tutti gli strumenti per effettuare la misura su un dispositivo FPGA secondo la METRICA del SYSTEM-GATE, riferendoci solamente alla logica che è possibile implementare nel dispositivo stesso tramite LOOK-UP TABLE nel seguente modo:

1) Effettuare la sintesi logica del Processore FFT con il Tool Synplify Pro 8.1 sul dispositivo in esame.
2) Calcolare tramite le funzioni (5.3) e (5.4) il valore dei GATE di SISTEMA del Processore FFT
3) Calcolare i GATE di SISTEMA del dispositivo FPGA in esame usando i dati della sintesi logica del Processore FFT (punto 1)

Per esempio per il dispositivo STRATIX abbiamo:
poniamo **x=19**

LUT_{USATI}=71961 $LUT_{dispositivo}$= 79040

GATE_FFT(19) = Ω * LUT_{logici}(19) = 214452

GATE_STRATIX = GATE_FFT(19) * (79040/71961)=235549

Possiamo misurare anche i GATE di SISTEMA relativi di una funzione qualsiasi.

È stato preso il codice VHDL MiniMips dalla rete[5] ed è stata effettuata la sintesi logica sul dispositivo SPARTAN-3 che ha prodotto i seguenti risultati:

LUT1	LUT2	LUT3	LUT4	LUTtotali
112	386	1482	1495	3731

Tramite la funzione (4.1) sono stati calcolati i LUT_{logici}:

LUT_{logici} = ¼ (LUT1 + 2 * LUT2 + 3 * LUT3) + LUT4=2827

I GATE equivalenti del Processore MiniMips risultano:

GATE_MiniMips = Ω LUT_{logici} = 25452

Con Ω= 9 GATE/LUT massimi (vedi capitolo 1).

[5] www.openceres.org

BIBLIOGRAFIA

COMPLETE DATA SHEET STRATIX.
www.altera.com

COMPLETE DATA SHEET SPARTAN-3.
www.xilinx.com

[Saponara et al,2005] S. Saponara, L. Serafini, L. Fanucci, P. Terreni, Automated design of FFT/IFFT processor for advenced telecom applications, International Symposium on signal, circuits and system, Jul 2005, pag 103-106, vol1.

[Fanucci el al,2002] L. Fanucci, M. Forliti, P. Terreni, FAST: FFT ASIC Automated Synthesis, Integration, the VLSI Journaol, dec.2002, vol. 33 num. 1-2, pag. 23-37.