Politecnico di Torino

DIPARTIMENTO DI ELETTRONICA E TELECOMUNICAZIONI
Corso di Laurea Magistrale in Ingegneria Elettronica

Sistemi Digitali Integrati
OPERAZIONE SAN SILVESTRO
Progettazione di un calcolatore di FFT

Prof. Maurizio Zamboni

Author:

Letizia Bricco (s328719)

A.A. 2023/2024

Indice

Sommario

L'obiettivo del progetto è sviluppare e testare un'unità di elaborazione basata sul processore Butterfly per il calcolo della trasformata di Fourier di un vettore di sedici numeri complessi mediante l'algoritmo FFT (*Fast Fourier Transform*).

Avendo a che fare con un'architettura integrata complessa, in fase di progetto si possono distinguere due fasi:

1. La prima parte è dedicata al design completo del processore Butterfly, di cui si devono definire l'algoritmo da implementare, la struttura della Execution Unit e della Control Unit e il timing, al fine di garantire la corretta sincronizzazione dei vari blocchi;
2. La seconda parte, invece, prevede l'assemblaggio delle unità Butterfly progettate in precedenza per la realizzazione dell'unità di calcolo effettiva; si tratta di un'operazione puramente "meccanica", in quanto le specifiche impongono di progettare il PE in modo che le varie Butterfly interfacciabili senza bisogno di alcun elemento aggiuntivo.

L'architettura del processore Butterfly è stata progettata rispettando rigorose specifiche sulla dinamica dei dati in ingresso e in uscita al *processing element*:

- Gli ingressi e le uscite sono numeri *signed fixed point*, definiti in forma frazionaria, in complemento a due (C2) e con un parallelismo di 24 bit (formato Q1.23).
- Di conseguenza, la dinamica è limitata all'intervallo $[-1, +1]$ sia in ingresso sia in uscita.
- Nello svolgimento delle operazioni interne alla Butterfly potrebbero esserci fino a 2 bit di overflow; quindi, le uscite calcolate potrebbero avere una dinamica eccessivamente grande rispetto a quella imposta.

  Per ricondursi al range $[-1, +1]$, imposto dalle specifiche, si deve utilizzare la tecnica dell'*unconditional block floating point scaling*, che consiste in un opportuno scalamento dei valori numerici ottenuti. Ai fini della progettazione dell'hardware, lo scalamento da implementare è sempre di 1 bit.
- I calcoli interni devono essere svolti *a precisione infinita*, i.e., senza effettuare alcun tipo di troncamento e, quindi, estendendo opportunamente il parallelismo delle linee per evitare *overflow*;
- L'arrotondamento deve essere effettuato solo sui dati di uscita, utilizzando la tecnica del *rounding to nearest even*.

Per il progetto della *Execution Unit*, si deve tenere conto della limitatezza delle risorse, a causa della quale alcuni blocchi di elaborazione devono essere utilizzati più volte per lo svolgimento di operazioni su dati differenti. Questo, ovviamente, renderà il progetto meno ottimizzato rispetto alle prestazioni che otterremmo con un progetto a risorse infinite, ma ha il vantaggio di contenere il costo della macchina realizzata.

La *Control Unit* va progettata come macchina microprogrammata, utilizzando la tecnica dell'indirizzamento implicito. La bassa complessità computazionale dell'algoritmo, costituito da non più di una ventina di step, consentirebbe di portare avanti il progetto anche con una semplice Macchina a Stati Finiti (FSM); ciononostante, la capacità di progettare un'architettura microprogrammata è una competenza importante nell'ottica di realizzare IP più complesse, in cui il numero di stati può facilmente essere dell'ordine delle centinaia.

La descrizione dell'architettura è stata implementata in VHDL in maniera gerarchica e il corretto comportamento è stato verificato sia mediante diverse *testbench*.

La simulazione è stata condotta utilizzando l'ambiente di sviluppo basato su Quartus-ModelSim e, per testare il blocco in maniera il più possibile completa, è stata automatizzata mediante uno script in linguaggio C++.

Per verificare la correttezza dei risultati, invece, si è utilizzato il software MATLAB®, che consente di effettuare in maniera molto rapida le operazioni sui numeri complessi.

Parte I
Progettazione

1 Introduzione

Prima di iniziare la progettazione del *processing element*, è doverosa un'introduzione matematica sul calcolo della trasformata di Fourier mediante l'algoritmo FFT.

Si consideri il segnale $t \mapsto x(t)$, definito nel dominio del tempo, e sia $\omega \mapsto X(\omega)$ la sua trasformata di Fourier. Assumendo di campionare x in N punti

$$\Big\{x[0], x[1], \dots, x[N-1]\Big\}, \tag{1.1}$$

è possibile ottenere altrettanti punti dello spettro $\Big\{X[0], X[1], \dots, X[N-1]\Big\}$ mediante la formula

$$X_N(k) = \sum_{n=0}^{N-1} x(n) W_N^k, \quad k = 0, \dots, N-1, \tag{1.2}$$

dove W_N^k prende il nome di *twiddle factor* ed è definito come segue:

$$W^k \equiv W_N^k := \exp\left(-j\,\frac{2\pi k}{N}\right) = \cos\frac{2\pi k}{N} - j\sin\frac{2\pi k}{N}, \quad k = 0, \dots, N-1. \tag{1.3}$$

Applicando alcune regole basilari dell'analisi complessa, possiamo notare la seguente proprietà dei twiddle factor, che ci sarà utile nel seguito per la derivazione dell'algoritmo:

$$W^{k+\frac{N}{2}} = \exp\left[-j\,\frac{2\pi}{N}\left(k+\frac{N}{2}\right)\right] = \underbrace{\exp\left(-j\,\frac{2\pi k}{N}\right)}_{=W^k}\underbrace{\exp\left(-j\pi\right)}_{=-1} = -W^k. \tag{1.4}$$

Questo significa che, per ottenere ognuno degli N campioni dello spettro, è necessario svolgere N operazioni. Di conseguenza, la complessità dell'algoritmo risulta essere $\mathcal{O}\left(N^2\right)$; per N grandi, il numero di operazioni da svolgere "esplode" e i tempi per ottenere i risultati diventano troppo grandi.

Per minimizzare il numero di oprazioni svolte, è possibile sostituire l'algoritmo appena presentato con l'algoritmo di **Cooley-Tookey**, che si basa sulla scomposizione ricorsiva della trasformata come somme di trasformate su opportuni sottoinsiemi dei valori di ingresso; la cardinalità di tali sottoinsiemi prende il nome di *radix* e si denota, tipicamente, con r.

In particolare, scegliendo la radix 2 è possibile scrivere ogni campione come una somma di due trasformate fatte sulla metà dei campioni:

$$X_N(k) = \sum_{n=0}^{\frac{N}{2}-1} x(2n) W_{N/2}^k + W_N^k \sum_{n=0}^{\frac{N}{2}-1} x(2n+1) W_{N/2}^k. \tag{1.5}$$

In tal modo, il costo computazionale della trasformata di Fourier risulta $\mathcal{O}\left(N \log_2 N\right)$.

Senza entrare ulteriormente nel dettaglio, denotiamo con $A, B, W \in \mathbb{C}$ gli ingressi del processore Butterfly:

$$A = \Re\{A\} + j\Im\{A\} =: A_\mathrm{r} + jA_\mathrm{i}, \tag{1.6a}$$

$$B = \Re\{B\} + j\Im\{B\} =: B_\mathrm{r} + jB_\mathrm{i}, \tag{1.6b}$$
$$W^k = \Re\{W^k\} + j\Im\{W^k\} =: W_\mathrm{r} + jW_\mathrm{i}, \tag{1.6c}$$

Utilizzando il sopracitato algoritmo di Cooley-Tukey, si dimostra che le uscite, denotate con $A', B' \in \mathbb{C}$, sono date dalle seguenti equazioni:

$$A' = A'_\mathrm{r} + jA'_\mathrm{i} = A + BW^k \tag{1.7a}$$
$$B' = B'_\mathrm{r} + jB'_\mathrm{i} = A + BW^{k+\frac{N}{2}} \equiv A - BW^k \tag{1.7b}$$

dove si è sfruttata la proprietà dei twiddle factor presentata nell'Equazione (1.4).

Sviluppiamo i calcoli per dettagliare ulteriormente le operazioni da implementare:

$$\begin{cases} A'_\mathrm{r} = \Re\Big\{(A_\mathrm{r} + jA_\mathrm{i}) + (W_\mathrm{r} + jW_\mathrm{i})(B_\mathrm{r} + jB_\mathrm{i})\Big\} = A_\mathrm{r} + B_\mathrm{r}W_\mathrm{r} - B_\mathrm{i}W_\mathrm{i} \\ A'_\mathrm{i} = \Im\Big\{(A_\mathrm{r} + jA_\mathrm{i}) + (W_\mathrm{r} + jW_\mathrm{i})(B_\mathrm{r} + jB_\mathrm{i})\Big\} = A_\mathrm{i} + B_\mathrm{r}W_\mathrm{i} + B_\mathrm{i}W_\mathrm{r} \\ B'_\mathrm{r} = \Re\Big\{(A_\mathrm{r} + jA_\mathrm{i}) - (W_\mathrm{r} + jW_\mathrm{i})(B_\mathrm{r} + jB_\mathrm{i})\Big\} = A_\mathrm{r} - B_\mathrm{r}W_\mathrm{r} + B_\mathrm{i}W_\mathrm{i} = 2A_\mathrm{r} - A'_\mathrm{r} \\ B'_\mathrm{i} = \Im\Big\{(A_\mathrm{r} + jA_\mathrm{i}) - (W_\mathrm{r} + jW_\mathrm{i})(B_\mathrm{r} + jB_\mathrm{i})\Big\} = A_\mathrm{i} - B_\mathrm{r}W_\mathrm{i} - B_\mathrm{i}W_\mathrm{r} = 2A_\mathrm{i} - A_\mathrm{i} \end{cases} \tag{1.8}$$

L'implementazione dell'algoritmo si può dunque ricondurre allo svolgimento delle seguenti operazioni:

$$\begin{cases} M_1 := B_\mathrm{r}W_\mathrm{r} \\ M_2 := B_\mathrm{i}W_\mathrm{i} \\ M_3 := B_\mathrm{r}W_\mathrm{i} \\ M_4 := B_\mathrm{i}W_\mathrm{r} \\ M_5 := 2A_\mathrm{r} \\ M_6 := 2A_\mathrm{i} \end{cases} \qquad \begin{cases} \Sigma_1 := A_\mathrm{r} + M_1 \\ \Sigma_2 := \Sigma_1 - M_2 \\ \Sigma_3 := A_\mathrm{i} + M_3 \\ \Sigma_4 := \Sigma_3 + M_4 \\ \Sigma_5 := M_5 - \Sigma_2 \\ \Sigma_6 := M_6 - \Sigma_4 \end{cases} \tag{1.9}$$

Dal momento che tornerà molto utile nel seguito, riscriviamo le operazioni separando quelle necessarie alla valutazione della parte reale (a sinistra) e immaginaria (a destra):

$$\begin{cases} M_1 = B_\mathrm{r}W_\mathrm{r} \\ M_2 = B_\mathrm{i}W_\mathrm{i} \\ M_5 = 2A_\mathrm{r} \\ \Sigma_1 = A_\mathrm{r} + M_1 \\ \Sigma_2 = \Sigma_1 - M_2 \\ \Sigma_5 = M_5 - \Sigma_2 \end{cases} \qquad \begin{cases} M_3 = B_\mathrm{r}W_\mathrm{i} \\ M_4 = B_\mathrm{i}W_\mathrm{r} \\ M_6 = 2A_\mathrm{i} \\ \Sigma_3 = A_\mathrm{i} + M_3 \\ \Sigma_4 = \Sigma_3 + M_4 \\ \Sigma_6 = M_6 - \Sigma_4 \end{cases} \tag{1.10}$$

Come possiamo notare, grazie all'algoritmo di Cooley Tookey ed alle proprietà dei numeri complessi, un algoritmo appartenentemente complicato è stato ridotto ad un insieme di 12 semplici operazioni di cui quattro moltiplicazioni, due *shift left* (moltiplicazioni per 2), tre somme e tre sottrazioni.

2 Considerazioni sulla dinamica dei dati

Come anticipato nel sommario, le specifiche di progetto impongono i seguenti vincoli:

1. Ingressi e uscite come numeri *signed fixed point*, in C2 e in formato Q1.23;
2. Il parallelismo delle linee interne va esteso opportunamente per svolgere i calcoli senza avere *overflow*;
3. Prima di inviare i risultati in uscita, essi devono essere scalati di 1 bit secondo la tecnica del *unconditional block floating point scaling* per rientrare nell'intervallo $[-1, +1]$.

Per realizzare il calcolatore di FFT completo, le Butterfly dovranno essere concatenate secondo uno schema a più livelli, in cui le uscite di ogni Butterfly diventano gli ingressi altre due, appartenenti al livello immediatamente successivo. In tale contesto, è possibile assumere che

- durante il **primo stadio**, in cui i dati provengono dall'esterno, sia possibile un *overflow* di **2 bit**;
- durante tutti gli **stadi successivi**, in cui i dati provengono dalle Butterfly dei livelli precedenti, sia possibile un *overflow* di **1 bit**.

Di conseguenza, il sistema può essere progettato come segue:

- Il blocco destinato alle operazioni di arrotondamento e scalamento implementa sempre uno scalamento di 1 bit;
- L'utilizzatore si impegna a far sì che la dinamica dei dati in ingresso al primo stadio sia contenuta nel range $[-0.5, +0.5]$: in questo modo, tenendo conto dell'*overflow* di 2 bit e dello scalamento di 1 bit, la dinamica di uscita è, come previsto, $[-1, +1]$.

3 Data Flow Diagram (DFD) e time scheduling

Il primo passo dello sviluppo del progetto è la derivazione del **Data Flow Diagram**; quello relativo al *processing element* progettato è riportato in Figura 3.2.

3.1 Progetto a risorse limitate

Il processore Butterfly deve essere progettato avendo a disposizione i seguenti blocchi hardware:

- 1 **moltiplicatore/shifter aritmetico**, con due livelli di pipeline per le operazioni di moltiplicazione e un livello di pipeline per le operazioni di *shift left*; la modalità di lavoro dipende dal valore di un opportuno segnale di selezione $M/\overline{S}$;
- 1 **sommatore** con un livello di pipeline;
- 1 **sottrattore** con un livello di pipeline;
- 1 **rounder/scaler** puramente combinatorio per le operazioni di arrotondamento e scalamento sui dati di uscita.

Per derivare il Data Flow Diagram, tuttavia, è opportuno partire dall'ipotesi di **progetto a risorse infinite**: dal momento che la soluzione ASAP (*As Soon As Possible*) richiederebbe di avere a disposizione ben 6 moltiplicatori, è possibile concentrarsi fin da subito sulla soluzione ALAP (*As Late As Possible*), mostrata in Figura 3.1:

- Lo spazio compreso tra due linee tratteggiate rappresenta un colpo di clock;

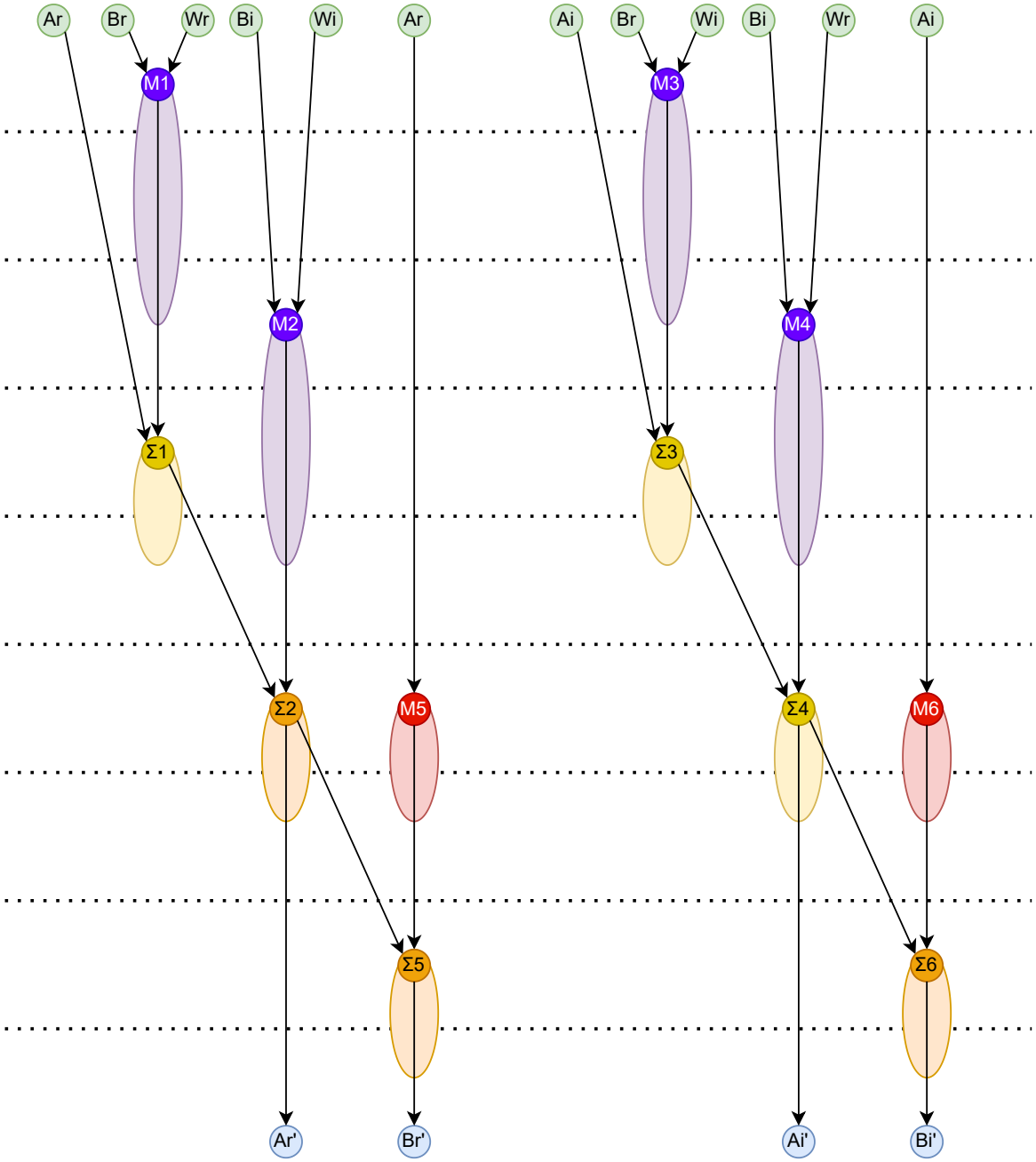


Figura 3.1: Data Flow Diagram del processore Butterfly nell'ipotesi di progetto a risorse infinite.

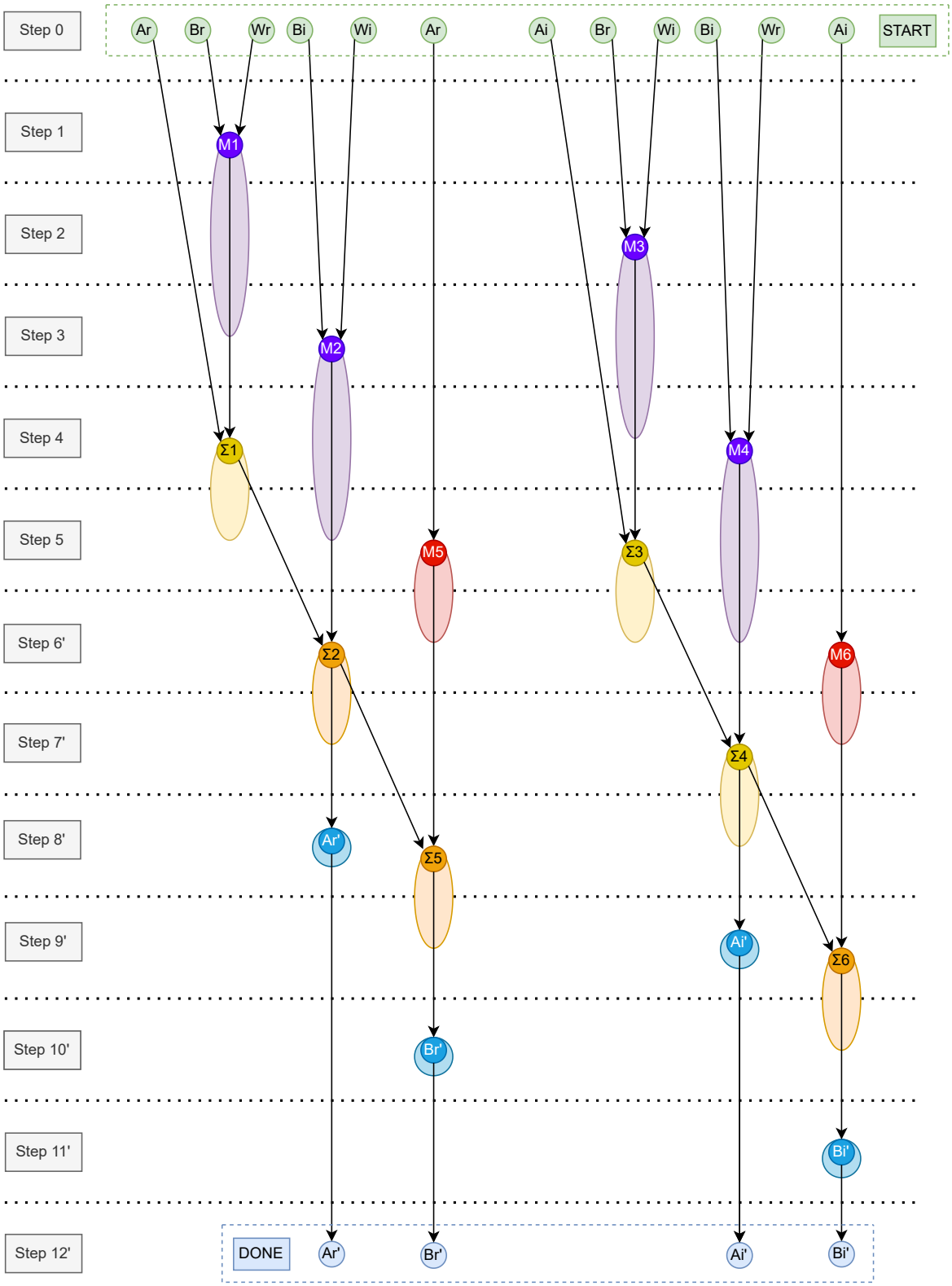


Figura 3.2: Data Flow Diagram del processore Butterfly nell'ipotesi di progetto a risorse limitate.

- Ad ogni operazione corrisponde un colore diverso: viola per le moltiplicazione, rosso per gli shift left, giallo per le somme e arancione per le sottrazioni.
- I cerchi più scuri rappresentano gli operatori, mentre gli ovali più chiari rappresentano la durata delle operazioni a seconda del numero di livelli di pipeline interni;
- Coerentemente con la filosofia di progetto ALAP, l'inizio di ogni operazione viene rimandato fino a quando il suo svolgimento non risulta strettamente necessario per procedere.

Sebbene formalmente corretto, questo scheduling non è implementabile perché richiede un numero di blocchi hardware maggiore di quelli a disposizione (2 moltiplicatori, 2 sommatori, 2 sottrattori).

Di conseguenza, il grafo in Figura 3.1 va modificato, spostando i punti di inizio delle operazioni in modo che, ad ogni step, non inizino mai due operazioni dello stesso tipo: in tal modo, si ottiene il diagramma di Figura 3.2. Nonostante il numero di step algoritmici richiesti (13) aumenti rispetto alla soluzione a risorse infinite, si tratta dell'unico modo per rispettare i *constraint* imposti.

Notiamo che, quando un operatore è disponibile all'uscita del blocco che lo ha generato, non può essere riutilizzato immediatamente ma si deve aspettare, come minimo, lo step algoritmico successivo: questo avviene perché ogni variabile viene memorizzata in un opportuno registro interno, garantendo così il sincronismo del circuito.

3.2 Analisi del tempo di vita e ottimizzazione dei registri interni

Una volta derivato il DFD, lo step immediatamente successivo è rappresentato dall'analisi del **tempo di vita delle variabili** (*variable timelife*), i.e., il tempo che intercorre tra

1. l'istante in cui un dato viene scritto nel rispettivo registro interno;
2. l'istante in cui il dato viene letto per l'ultima volta dal medesimo registro.

Se i tempi di vita di due variabili si sovrappongono, esse possono condividere lo stesso registro, ottimizzando così il numero complessivo di registri interni. All'atto pratico, l'ottimizzazione si implementa in tre step:

1. Derivazione del tempo di vita delle variabili per una Butterfly in modalità di esecuzione **isolata**[1]: a tale scopo, è possibile utilizzare primo dei due DFD riportati in Appendice; il risultato è riportato in Tabella 3.1.
2. Derivazione del DFD e del tempo di vita delle variabili per una Butterfly in modalità di esecuzione **continua**[2]: per fare ciò, si può notare che, a partire dallo step 7, il moltiplicatore non riceve più alcun dato in ingresso. Di conseguenza, da questo istante è possibile inviare alla Butterfly un nuovo blocco di dati e far partire una seconda sequenza di operazioni.

   La stesura del nuovo DFD, che si compone di 19 step algoritmici, è molto semplice: è sufficiente sovrapporre un secondo DFD al primo, sfalsato di 6 colpi di clock; il risultato è mostrato nel grafo posto in Appendice a questo documento. Per il tempo di vita delle variabili si procede come nella modalità singola e il risultato delle operazioni è mostrato in Tabella 3.2.
3. Minimizzazione del numero di registri: dalla Tabella 3.2, si vede che il numero massimo di registri occupati è pari a 9 (durante gli step 7 e 8): gli operatori possono pertanto essere riorganizzati nei registri $\mathrm{REG}_1 \div \mathrm{REG}_9$, come mostrato in Tabella 3.3.

[1] Si parla di *modalità isolata* quando alla Butterfly viene inviata una sola sequenza di dati da elaborare.
[2] Si parla di *modalità continua* quando alla Butterfly viene inviato uno stream continuo di dati.

Tabella 3.1: Tempo di vita delle variabili per un processore Butterfly in esecuzione isolata.

Operatore	1	2	3	4	5	6	7	8	9	10	11	12
W_r	W_r											
W_i	W_i											
A_r	A_r											
A_i	A_i											
B_r	B_r											
B_i	B_i											
M_1				M_1								
M_3					M_3							
Σ_1						Σ_1						
M_2						M_2						
Σ_3							Σ_3					
M_4							M_4					
M_5							M_5					
M_6								M_6				
Σ_2								Σ_2				
Σ_4									Σ_4			
Σ_5										Σ_5		
Σ_6											Σ_6	
A'_r									A'_r			
A'_i										A'_i		
B'_r											B'_r	
B'_i												B'_i

Tabella 3.2: Tempo di vita delle variabili per un processore Butterfly in esecuzione continua.

Operatore	1	2	3	4	5	6	7	8	9	10	11	12	7′	8′	9′	10′	11′	12′
W_r		W_r		W_r			W_r		W_r									
W_i		W_i		W_i			W_i		W_i									
A_r		A_r		A_r			A_r		A_r									
A_i		A_i		A_i			A_i		A_i									
B_r		B_r					B_r											
B_i		B_i		B_i			B_i		B_i									
M_1				M_1						M_1								
M_3						M_3					M_3							
Σ_1						Σ_1						Σ_1						
M_2						M_2						M_2						
Σ_3							Σ_3						Σ_3					
M_4							M_4						M_4					
M_5							M_5		M_5				M_5					
M_6									M_6				M_6					
Σ_2									Σ_2				Σ_2					
Σ_4									Σ_4				Σ_4					
Σ_5										Σ_5			Σ_5					
Σ_6											Σ_6							Σ_6
A'_r											A'_r							A'_r
A'_i											A'_i							A'_i
B'_r											B'_r							B'_r
B'_i												B'_i						B'_i
Registri occupati	6	6	5	5	3	3	9	9	8	8	7	7	3	3	3	3	4	4

Tabella 3.3: Ottimizzazione dei registri interni.

Registro	1	2	3	4	5	6	7	8	9	10	11	12	7′	8′	9′	10′	11′	12′
REG_1		W_r					W_r											
REG_2		W_i					W_i											
REG_3			A_r			Σ_1			A_r			Σ_1						
REG_4			A_i							A_i								
REG_5		B_r						B_r			Σ_5	Σ_6	B'_i			Σ_5	Σ_6	B'_i
REG_6			B_i						B_i			B'_r						B'_r
REG_7								M_5		A'_r				M_5			A'_r	
REG_8							Σ_3		M_6		A'_i		Σ_3		M_6			A'_i
REG_9				M_1	M_3	M_2	M_4	Σ_2	Σ_4	M_1	M_3	M_2	M_4	Σ_2	Σ_4			
Registri occupati	6	6	5	5	3	3	9	9	8	8	7	7	3	3	3	3	4	4

Tabella 4.1: Numero di porte di ingresso e uscite coinvolte nello scambio di dati ad ogni step.

Step	1	2	3	4	5	6	7	8	9	10	11	12	7′	8′	9′	10′	11′	12′
In	2	2	2	4	3	3	4	5	5	5	4	3	2	3	3	1	1	0
Out	0	0	1	1	2	3	2	2	3	3	3	3	2	2	2	2	1	0

4 Derivazione dell'architettura

4.1 Progetto e ottimizzazione

Prima di derivare l'architettura vera e propria, è opportuno definire i **simboli degli ingressi e delle uscite** dei blocchi principali presenti nel circuito:

- D_{1M} e D_{2M}: ingressi del moltiplicatore/shifter;
- Q_M e Q_{SH}: uscite del moltiplicatore/shifter;
- D_{1A} e D_{2A}: ingressi del sommatore;
- Q_A: uscita del sommatore;
- D_{1S} e D_{2S}: ingressi del sottrattore;
- Q_{SB}: uscita del sottrattore;
- D_R e Q_R: ingresso e uscita del rounder/scaler.

È inoltre necessario stabilire il **timing di ingresso dei dati** all'interno del processore; ovviamente, la sequenza di ingresso deve essere riprodotta allo stesso modo in uscita per permettere alle uscite di una Butterfly di interfacciarsi con gli ingressi di un'altra.

Per semplicità, può essere comodo scegliere di inviare **tutti e sei i dati di ingresso sullo stesso colpo di clock**, insieme ad un impulso di START che fa partire il processore; analogamente, è previsto un segnale di DONE che viene asserito per un colpo di clock in modo da segnalare che le uscite sono disponibili e possono essere campionate.

La Tabella 4.1, derivata direttamente dal DFD, mostra il numero di linee di ingresso e di uscita dei blocchi su cui transitano dei dati ad ogni step algoritmico: come vediamo, il numero massimo di linee occupate è 5 per quelle di ingresso, 3 per quelle di uscita.

In linea di principio, pertanto, sarebbe possibile progettare il *datapath* del processore inserendo tutti i registri in una struttura *register-file-like* dotata di **5 porte di uscita** e **3 porte di ingresso**.

Anche se, nel seguito, verrà utilizzato spesso il termine *register file*, va sottolineato che la struttura di memorizzazione scelta non è un register file nel vero senso del termine perché i dati contenuti nei registri risultano disponibili in uscita nel colpo di clock immediatamente successivo a quello in cui vengono scritti; per come è stato pensato, serve semplicemente a raggruppare in un unico componente tutti i registri anziché lasciarli nel datapath come registri sparsi.

Tale soluzione, tuttavia, non è molto ottimizzata: infatti, raggruppando le operazioni per tipo:

$$\begin{cases} M_1 = B_r W_r \\ M_2 = B_i W_i \\ M_3 = B_r W_i \\ M_4 = B_i W_r \end{cases} \quad \begin{cases} M_5 = 2A_r \\ M_6 = 2A_i \end{cases} \quad \begin{cases} \Sigma_1 = A_r + M_1 \\ \Sigma_3 = A_i + M_3 \\ \Sigma_4 = \Sigma_3 + M_4 \end{cases} \quad \begin{cases} \Sigma_2 = \Sigma_1 - M_2 \\ \Sigma_5 = M_5 - \Sigma_2 \\ \Sigma_6 = M_6 - \Sigma_4 \end{cases} \quad \begin{cases} A'_r = r(\Sigma_2) \\ A'_i = r(\Sigma_4) \\ B'_r = r(\Sigma_5) \\ B'_i = r(\Sigma_6) \end{cases}$$

si nota immediatamente che

1. gli ingressi del moltiplicatore/shifter (D_{1M} e D_{2M}) provengono sempre dalle porte di ingresso del PE, mentre l'uscita dello shifter Q_{SH} va sempre inviata all'ingresso D_{1S} del sottrattore;
2. l'ingresso D_{2A} del sommatore proviene sempre dall'uscita del moltiplicatore Q_M;
3. l'ingresso D_{2S} del sottrattore può provenire dalle uscite del moltiplicatore Q_M, del sommatore Q_A e del sottrattore Q_{SB}.

Queste considerazioni, in particolare i punti 2 e 3, suggeriscono di "tirare fuori" dal register file alcuni dei registri e di collegarli direttamente agli ingressi e/o alle uscite dei blocchi, mediante opportuni multiplexer.

In base allo schema riportato in tabella 3.3, la scelta più ovvia è portare fuori dal register file REG_7, REG_8 e REG_9; infatti, in tali registri vengono memorizzati i risultati di tutte le moltiplicazioni ($M_1 \div M_4$), tutti gli shift (M_5, M_6) due delle tre somme (Σ_3, Σ_4) e una delle tre sottrazioni (Σ_2).

Viceversa, siccome nei registri $REG_1 \div REG_6$ vengono memorizzati tutti i valori di ingresso, è opportuno che siano tutti riuniti in un'unica struttura, anche per una questione di "ordine mentale".

In questo modo, D_{2A}, D_{2S}, Q_M e Q_{SH} non necessitano più di alcun collegamento con il register file in quanto comunicano solo i registri sparsi REG_7, REG_8 e REG_9; di conseguenza:

- Il numero di linee di uscita del register file verso il datapath è ridotto da 5 a 4;
- Il numero di linee di ingresso nel register file dal datapath è ridotto da 3 a 2.

A questo punto, non resta che procedere ai collegamenti tra i vari blocchi del datapath; a tale scopo, può essere utile riformulare la Tabella 3.3 come mostrato nelle Tabelle 4.2, 4.3 e 4.4.

Tabella 4.2: Operatori entranti in ogni blocco e relativi registri di partenza.

Ingresso	Operatori e registri di partenza			
D_{1M}	A_r REG_3	A_i REG_4	B_r REG_5	B_i REG_6
D_{2M}	W_r REG_1	W_i REG_2		
D_{1A}	A_r REG_3	A_i REG_4	Σ_3 REG_8	
D_{2A}	M_1 REG_9	M_3 REG_9	M_4 REG_9	
D_{1S}	Σ_1 REG_3	M_5 REG_7	M_6 REG_8	
D_{2S}	M_2 REG_9	Σ_2 REG_9	Σ_4 REG_9	
D_R	Σ_2 REG_9	Σ_4 REG_9	Σ_5 REG_5	Σ_6 REG_5

Tabella 4.3: Operatori uscenti da ogni blocco e relativi registri di arrivo.

Uscita	Operatori e registri di arrivo			
Q_M	M_1 REG_9	M_2 REG_9	M_3 REG_9	M_4 REG_9
Q_{SH}	M_5 REG_7	M_6 REG_8		
Q_A	Σ_1 REG_3	Σ_3 REG_8	Σ_4 REG_9	
Q_{SB}	Σ_2 REG_9	Σ_5 REG_5	Σ_6 REG_5	
Q_R	A'_r REG_7	A'_i REG_8	B'_r REG_6	B'_i REG_5

Tabella 4.4: Per ognuno dei registri, operatori entranti ed uscenti con le rispettive porte di partenza (Q) e di arrivo (D).

$REG_\#$	Operatori entranti						Operatori uscenti					
REG_1	W_r ext						W_r D_{2M}					
REG_2	W_i ext						W_i D_{2M}					
REG_3	A_r ext	Σ_1 Q_A					A_r D_{1M} D_{1A}	Σ_1 D_{1S}				
REG_4	A_i ext						A_i D_{1M} D_{1A}					
REG_5	B_r ext	Σ_5 Q_{SB}	Σ_6 Q_{SB}	B'_i Q_R			B_r D_{1M}	Σ_5 D_R	Σ_6 D_R	B'_i ext		
REG_6	B_i ext	B'_r Q_R					B_i D_{1M}	B'_r ext				
REG_7	M_5 Q_{SH}	A'_r Q_R					M_5 D_{1S}	A'_r ext				
REG_8	Σ_3 Q_A	M_6 Q_{SH}	A'_i Q_R				Σ_3 D_{1A}	M_6 D_{1S}	A'_i ext			
REG_9	M_1 Q_M	M_3 Q_M	M_2 Q_M	M_4 Q_M	Σ_2 Q_{SB}	Σ_4 Q_A	M_1 D_{2A}	M_3 D_{2A}	M_2 D_{2S}	M_4 D_{2A}	Σ_2 D_{2S} D_R	Σ_4 D_{2S} D_R

L'architettura può essere osservata nelle immagini riportate in Appendice; nel complesso, i blocchi costituenti risultano essere i seguenti:

- 1 moltiplicatore/shifter, denotato con il label MPY_SHIFTER;
- 1 sommatore, denotato con il label ADDER;
- 1 sottrattore, denotato con il label SUBTRACTOR;
- 1 rounder/scaler, denotato con il label ROUNDER_SCALER;
- 3 registri sparsi, denotati con i label REG_7, REG_8 e REG_9;
- 1 struttura di memorizzazione, denotata con il label REG_FILE e contenente i restanti 6 registri e alcuni multiplexer prima alle porte di ingresso o dopo le porte di uscita dei registri stessi;
- 3 multiplexer davanti agli ingressi di REG_7, REG_8 e REG_9, denotati con i label mux_R7, mux_R8 e mux_R9;
- 3 multiplexer davanti alle porte di ingresso D1A del sommatore, D1S del sottrattore e DR del rounder;
- 1 multiplexer davanti alla porta di ingresso IN_AS del register file.

4.2 Parallelismo interno

L'utilizzo di numeri in forma frazionaria permette di stabilire il parallelismo interno della macchina trattando i dati come numeri interi in complemento a due. È noto che, in tali condizioni, un numero X codificato su n bit è compreso tra -2^{n-1} e $+\left(2^{n-1}-1\right)$; è pertanto possibile condurre i seguenti ragionamenti:

Moltiplicatore. Gli ingressi D_{1M} e D_{2M} provengono dall'esterno o da un'altra Butterfly; pertanto, le specifiche impongono che siano codificati su 24 bit; l'uscita Q_M, invece, è codificata su $2 \times 24 - 1 = 47$ bit in virtù delle regole dell'aritmetica *fractional point* in C2.

Shifter. Come nel caso del moltiplicatore, l'ingresso D_{1M} proviene dall'esterno ed è quindi codificato su 24 bit; dal momento che la moltiplicazione per 2 di un numero binario equivale a uno *shift left* di una posizione, per l'uscita Q_{SH} sono necessari 25 bit.

Sommatore. Gli ingressi D_{1A} e D_{2A} provengono dall'esterno, dal moltiplicatore o dal sommatore stesso. Il *worst case* è rappresentato dall'operazione

$$\Sigma_4 = \Sigma_3 + M_4 \equiv A_i + M_3 + M_4, \tag{4.1}$$

in cui vengono sommati tre numeri codificati, rispettivamente, su 24, 47 e 47 bit. Come anticipato, possiamo studiare i valori minimo e massimo di Σ_4 supponendo di avere a che fare con numeri interi:

$$\min\{\Sigma_4\} = -2^{23} - 2^{46} - 2^{46} = -2^{47} - 2^{23} < -2^{47} \tag{4.2a}$$

$$\max\{\Sigma_4\} = \left(2^{23}-1\right) + 2\left(2^{46}-1\right) = \left(2^{47}-1\right) + \left(2^{23}-2\right) > 2^{47} - 1 \tag{4.2b}$$

Segue che, per non avere overflow, sia gli ingressi sia l'uscita devono essere codificati almeno su 49 bit.

Tabella 4.5: Parallelismo dei dati in ingresso e in uscita a ogni blocco.

D_{1M}	D_{2M}	Q_M	Q_{SH}	D_{1A}	D_{2A}	Q_A	D_{1S}	D_{2S}	Q_{SB}	D_R	Q_R
24	24	47	25	49	49	49	50	50	50	50	24

Sottrattore. Gli ingressi D_{1S} e D_{2S} provengono dal moltiplicatore, dallo shifter, dal sommatore o dal sottrattore stesso. Il *worst case* è rappresentato dall'operazione

$$\Sigma_5 = M_5 - \Sigma_2 \equiv M_2 + M_5 - \Sigma_1, \tag{4.3}$$

in cui i termini sono codificati, rispettivamente, su 47, 25 e 49 bit:

$$\min\{\Sigma_5\} = -2^{48} - 2^{46} - 2^{24} < -2^{48} \tag{4.4a}$$
$$\max\{\Sigma_5\} = \left(2^{48} - 1\right) + \left(2^{46} + 2^{24} - 2\right) > 2^{48} - 1 \tag{4.4b}$$

Pertanto, il minimo parallelismo richiesto è pari a 50 bit.

Rounder. L'ingresso D_R deve essere codificato su 50 bit per poter ricevere i dati uscenti dal sottrattore; l'uscita, invece, sarà un numero a 24 bit opportunamente arrotondato secondo la tecnica *rounding to nearest even* descritta nel paragrafo 4.3.

La tabella 4.5 riassume il parallelismo degli ingressi e delle uscite di tutti i blocchi presenti nell'architettura.

4.3 Arrotondamento e scalamento

Preliminarmente, definiamo i possibili risultati dell'operazione di arrotondamento secondo la tecnica *rounding to nearest even* in base alla seguente tabella:

D_R	$r(D_R)$	f	errore
$X000X$	$X0$	A	0
$X001X$	$X0$	A	-0.25
$X010X$	$X0$	A	0.50
$X011X$	$X1$	B	0.25
$X100X$	$X1$	B	0
$X101X$	$X1$	B	-0.25
$X110X$	$X1+1$	C	0.50
$X111X$	$X1+1$	C	0.25

(4.5)

Dal momento che D_R è codificato su 50 bit in formato Q4.46, mentre il risultato dell'arrotondamento è in formato Q1.23:

- La parte più significativa del dato, il cui valore è *don't care*, è data dai bit $3 \div -22$;
- Per effettuare l'arrotondamento, bisogna analizzare i bit -23, -24 e -25 di D_R; in particolare, definiamo
$$x_1 := D_R(-23), \quad x_2 := D_R(-24), \quad x_3 := D_R(-25). \tag{4.6}$$
- La parte meno significativa del dato, che viene troncata, è data dai bit $-26 \div -46$.

Realizzando la tabella di verità della funzione di arrotondamento f, è pobbibile scriverla come somma di *minterm*:

i	x_1	x_2	x_3	m	$f(x_1, x_2, x_3)$
0	0	0	0	$\overline{x}_1\overline{x}_2\overline{x}_3$	A
1	0	0	1	$\overline{x}_1\overline{x}_2 x_3$	A
2	0	1	0	$\overline{x}_1 x_2\overline{x}_3$	A
3	0	1	1	$\overline{x}_1 x_2 x_3$	B
4	1	0	0	$x_1\overline{x}_2\overline{x}_3$	B
5	1	0	1	$x_1\overline{x}_2 x_3$	B
6	1	1	0	$x_1 x_2\overline{x}_3$	C
7	1	1	1	$x_1 x_2 x_3$	C

(4.7)

$$f(x_1, x_2, x_3) = Am_A + Bm_B + Cm_C, \tag{4.8}$$

dove

$$\begin{cases} m_A = \overline{x}_1\left(\overline{x}_2 + x_2\overline{x}_3\right) \\ m_B = x_1\overline{x}_2 + \overline{x}_1 x_2 x_3 \\ m_C = x_1 x_2 \end{cases}$$

Lo stesso risultato si ottiene utilizzando una mappa di Karnaugh:

x_1 \ x_2x_3	00	01	11	10
0	A	A	B	A
1	B	B	C	C

5 Derivazione della Control Unit

Il nucleo della Control Unit microprogrammata è rappresentato dalla **micro-ROM**, una struttura di memorizzazione che può essere letta ma non modificata; che si caratterizza come segue:

- Ogni riga rappresenta uno degli step algoritmici previsti dal DFD; pertanto, in totale sono presenti **20 righe**;
- Ogni cella è costituita da 43 bit suddivisi come segue:
  1. Bit $42 \div 15$: segnali di controllo (selettori dei multiplexer, write enable dei registri, selettore della modalità di lavoro del moltiplicatore);
  2. Bit 14: segnale di DONE;
  3. Bit $13 \div 9$ e $6 \div 2$: *jump address* J_ADD_1 e J_ADD_2;
  4. Bit $8 \div 7$ e $1 \div 0$: *condition code* CC_1 e CC_2.

L'evoluzione di stato dipende dalla modalità di accesso agli indirizzi della microROM in base ai seguenti principi:

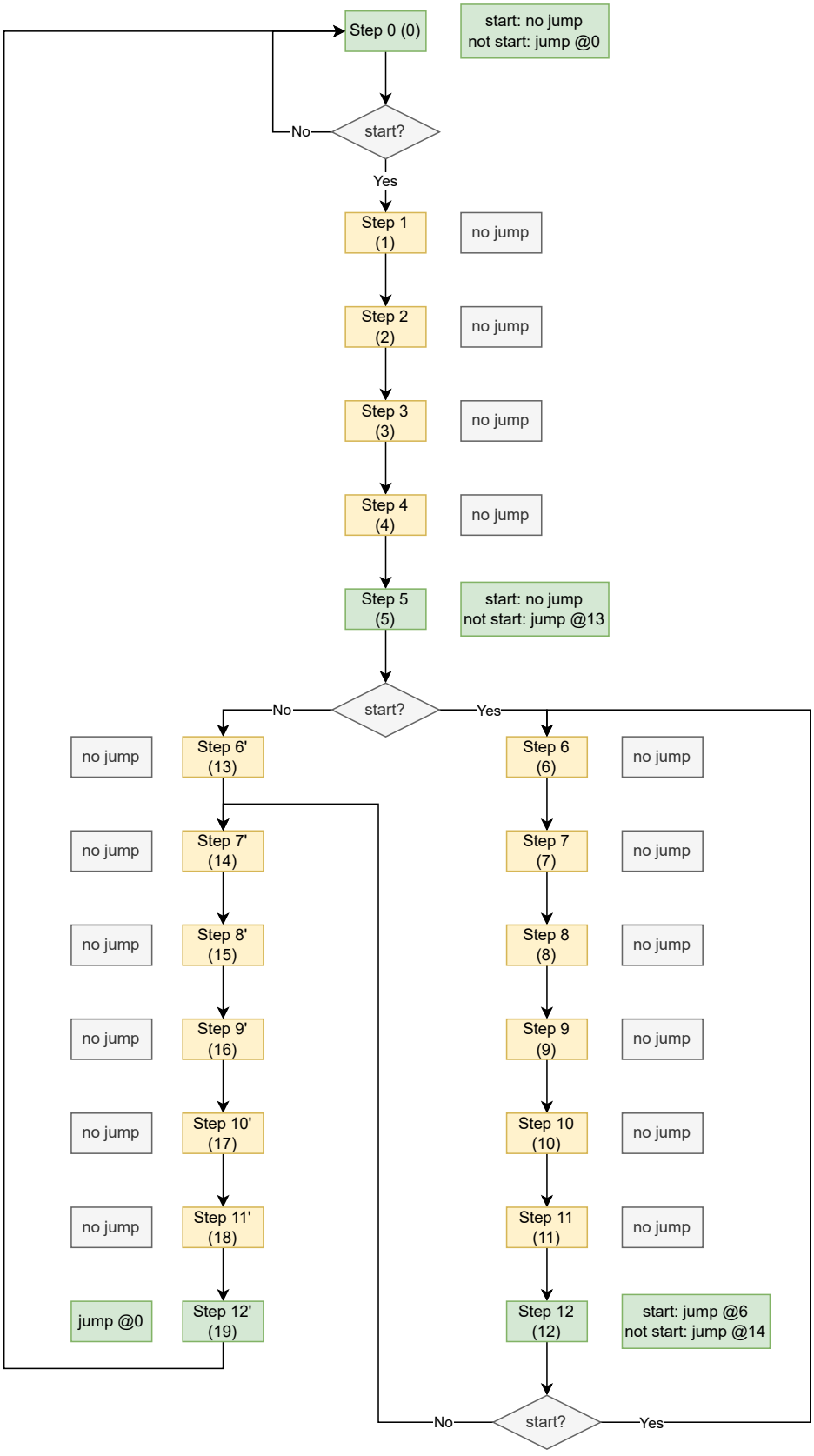


Figura 5.1: Flow chart dell'evoluzione di stato.

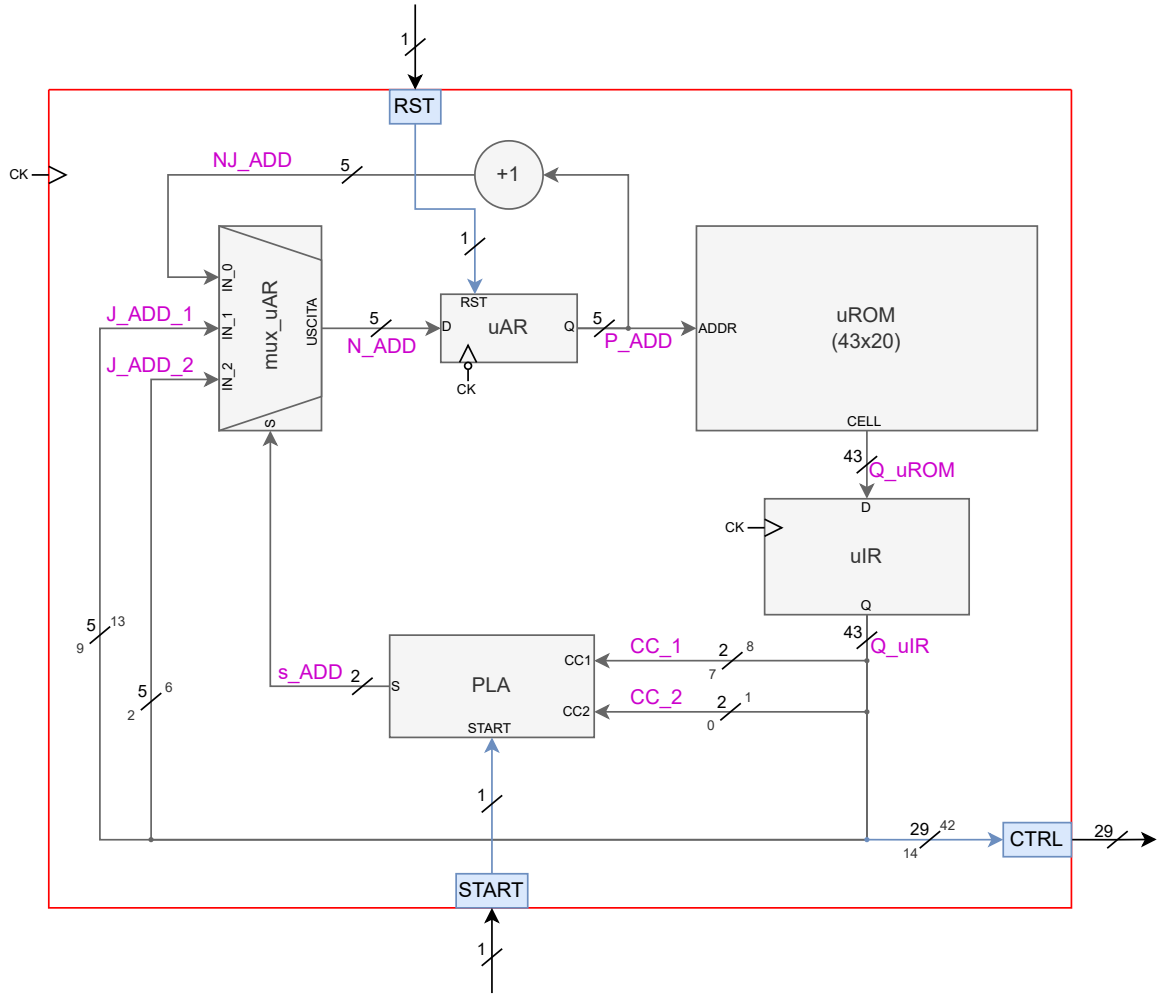


Figura 5.2: Control Unit con indirizzamento implicito. Sono indicati i parallelismi di tutte le linee e i nomi dei segnali, coerentemente con la descrizione VHDL.

- Se entrambi i condition code sono nulli, l'indirizzo dello stato futuro è quello immediatamente successivo, ottenuto incrementando l'indirizzo dello stato presente di 1;
- Se uno dei due condition code è non nullo, l'indirizzo dello stato futuro è dato da uno dei due jump address.

Per stabilire i valori dei condition code e dei jump address ad ogni step è stato utilizzato il flowchart riportato in Figura 5.1. I condition code hanno la seguente tabella di decodifica:

CC	decodifica
00	never
01	always
10	not start
11	start

(5.1)

A livello architetturale, come mostrato in Figura 5.2, oltre alla micro-ROM sono presenti i seguenti blocchi:

Micro-Instruction Register. Registro sensibile ai fronti di **salita** del clock, in cui viene memorizzata la cella di memoria selezionata dall'indirizzo dello stato presente;

Status PLA. Blocco asincrono che riceve in ingresso i condition code contenuti nel micro-IR e lo start e determina il valore del selettore di un multiplexer a due vie, incaricato di scegliere tra l'indirizzo sequenziale e i due jump address; la funzione logica implementata al suo interno è la seguente:

i	x_1	x_2	x_3	x_4	x_5	m	$f(x_1, x_2, x_3, x_4, x_5)$
0	0	0	0	0	0	$\overline{x}_1\overline{x}_2\overline{x}_3\overline{x}_4\overline{x}_5$	00
1	0	0	0	0	1	$\overline{x}_1\overline{x}_2\overline{x}_3\overline{x}_4 x_5$	00
8	0	1	0	0	0	$\overline{x}_1 x_2\overline{x}_3\overline{x}_4\overline{x}_5$	01
9	0	1	0	0	1	$\overline{x}_1 x_2\overline{x}_3\overline{x}_4 x_5$	01
16	1	0	0	0	0	$x_1\overline{x}_2\overline{x}_3\overline{x}_4\overline{x}_5$	01
17	1	0	0	0	1	$x_1\overline{x}_2\overline{x}_3\overline{x}_4 x_5$	00
28	1	1	1	0	0	$x_1 x_2 x_3\overline{x}_4\overline{x}_5$	10
29	1	1	1	0	1	$x_1 x_2 x_3\overline{x}_4 x_5$	01

Di conseguenza, f può essere scritta come somma di *minterm*:

$$f(x_1, x_2, x_3, x_4, x_5) = s_0 00 + s_1 01 + s_2 10, \tag{5.2}$$

dove

$$\begin{cases} s_0 = \overline{x}_2\overline{x}_3\overline{x}_4\left(\overline{x}_1 + x_1 x_5\right) \\ s_1 = \overline{x}_3\overline{x}_4\left(\overline{x}_1 x_2 + x_1\overline{x}_2 x_5\right) + x_1 x_2 x_3\overline{x}_4 x_5 \\ s_2 = x_1 x_2 x_3\overline{x}_4\overline{x}_5 \end{cases} \tag{5.3}$$

Micro-Address Register. Registro sensibile ai fronti di **discesa** del clock, in cui viene memorizzato l'indirizzo dello stato futuro selezionato dal mux. Il campionamento sul fronte di discesa è necessario per evitare che l'unità di controllo sia pipelinata sul controllo, il che impedirebbe la corretta evoluzione di stato.

Per costruire la micro-ROM è necessario procedere step per step e determinare i valori di tutti i segnali di controllo presenti nel circuito; lo schema utilizzato è riportato nel seguito.

Step #	Operazioni	Controlli	Done
0	Load register file Wait start (impulso di 1 CK)	WR_R1 = 1 WR_R2 = 1 WR_R3 = 1, s_R3 = 0 WR_R4 = 1 WR_R5 = 1, s_R5 = 0 (00) WR_R6 = 1, s_R6 = 0	0
1	Start $M_1 = B_r W_r$ (mpy) $D_{1M} = B_r$, leggo da REG_5 $D_{2M} = W_r$, leggo da REG_1	$M/\overline{S} = 1$ s_1M = 2 (10) s_2M = 0	0
2	Start $M_3 = B_r W_i$ (mpy) $D_{1M} = B_r$, leggo da REG_5 $D_{2M} = W_i$, leggo da REG_2 Pipe $M_1 = B_r W_r$ (mpy)	$M/\overline{S} = 1$ s_1M = 2 (10) s_2M = 1	0
3	Start $M_2 = B_i W_i$ (mpy) $D_{1M} = B_i$, leggo da REG_6 $D_{2M} = W_i$, leggo da REG_2 $Q_M = M_1$, scrivo in REG_9 Pipe $M_3 = B_r W_i$	$M/\overline{S} = 1$ s_1M = 3 (11) s_2M = 1 WR_R9 = 1, s_R9 = 0 (00)	0
4	Start $M_4 = B_i W_r$ (mpy) $D_{1M} = B_i$, leggo da REG_6 $D_{2M} = W_r$, leggo da REG_1 Start $\Sigma_1 = A_r + M_1$ $D_{1A} = A_r$, leggo da REG_3 $D_{2A} = M_1$, leggo da REG_9 $Q_M = M_3$, scrivo in REG_9 Pipe $M_2 = B_i W_i$	$M/\overline{S} = 1$ s_1M = 3 (11) s_2M = 0 s_1AS = 0, s_D1A = 1 WR_R9 = 1, s_R9 = 0 (00)	0
5	Start $M_5 = 2A_r$ (shift) $D_{1M} = A_r$, leggo da REG_3 Start $\Sigma_3 = A_i + M_3$ $D_{1A} = A_i$, leggo da REG_4 $D_{2A} = M_3$, leggo da REG_9 $Q_M = M_2$, scrivo in REG_9 $Q_A = \Sigma_1$, scrivo in REG_3 Pipe $M_4 = B_i W_r$	$M/\overline{S} = 0$ s_1M = 0 (00) s_1AS = 1, s_D1A = 1 WR_R9 = 1, s_R9 = 0 (00) s_IN_AS = 0, WR_R3 = 1, s_R3 = 1	0
6 ≡ 0′	Start $M_6 = 2A_i$ (shift) $D_{1M} = A_i$, leggo da REG_4 Start $\Sigma_2 = \Sigma_1 - M_2$ $D_{1S} = \Sigma_1$, leggo da REG_3 $D_{2S} = M_2$, leggo da REG_9 $Q_M = M_4$, scrivo in REG_9 $Q_{SH} = M_5$, scrivo in REG_7 $Q_A = \Sigma_3$, scrivo in REG_8	$M/\overline{S} = 0$ s_1M = 1 (01) s_1AS = 0, s_D1S = 2 (10) WR_R9 = 1, s_R9 = 0 (00) WR_R7 = 1, s_R7 = 0 WR_R8 = 1, s_R8 = 1 (01)	0
	Load reg file	WR_R1 = 1	

Continua nella pagina successiva

Continua dalla pagina precedente

Step #	Operazioni	Controlli	Done
	Wait start (impulso di 1 CK)	WR_R2 = 1 WR_R3 = 1, s_R3 = 0 WR_R4 = 1 WR_R5 = 1, s_R5 = 0 (00) WR_R6 = 1, s_R6 = 0	
7 ≡ 1′	Start $\Sigma_4 = \Sigma_3 + M_4$ $D_{1A} = \Sigma_3$, leggo da REG_8 $D_{2A} = M_4$, leggo da REG_9 $Q_{SH} = M_6$, scrivo in REG_8 $Q_{SB} = \Sigma_2$, scrivo in REG_9	 s_D1A = 0 WR_R8 = 1, s_R8 = 0 (00) WR_R9 = 1, s_R9 = 2 (10)	0
	Start $M_1 = B_r W_r$ (mpy) $D_{1M} = B_r$, leggo da REG_5 $D_{2M} = W_r$, leggo da REG_1	$M/\overline{S} = 1$ s_1M = 2 (10) s_2M = 0	
8 ≡ 2′	Round Σ_2 $D_R = \Sigma_2$, leggo da REG_9 $Q_R = A'_r$, scrivo in REG_7 Start $\Sigma_5 = M_5 - \Sigma_2$ $D_{1S} = M_5$, leggo da REG_7 $D_{2S} = \Sigma_2$, leggo da REG_9 $Q_A = \Sigma_4$, scrivo in REG_9	 s_DR = 1 WR_R7 = 1, s_R7 = 1 s_D1S = 1 (01) WR_R9 = 1, s_R9 = 1 (01)	0
	Start $M_3 = B_r W_i$ (mpy) $D_{1M} = B_r$, leggo da REG_5 $D_{2M} = W_i$, leggo da REG_2 Pipe $M_1 = B_r W_r$ (mpy)	$M/\overline{S} = 1$ s_1M = 2 (10) s_2M = 1	
9 ≡ 3′	Round Σ_4 $D_R = \Sigma_4$, leggo da REG_9 $Q_R = A'_i$, scrivo in REG_8 Start $\Sigma_6 = M_6 - \Sigma_4$ $D_{1S} = M_6$, leggo da REG_8 $D_{2S} = \Sigma_4$, leggo da REG_9 $Q_{SB} = \Sigma_5$, scrivo in REG_5	 s_DR = 1 WR_R8 = 1, s_R8 = 2 (10) s_D1S = 0 (00) s_IN_AS = 1, WR_R5 = 1, s_R5 = 1 (01)	0
	Start $M_2 = B_i W_i$ (mpy) $D_{1M} = B_i$, leggo da REG_6 $D_{2M} = W_i$, leggo da REG_2 $Q_M = M_1$, scrivo in REG_9 Pipe $M_3 = B_r W_i$	$M/\overline{S} = 1$ s_1M = 3 (11) s_2M = 1 WR_R9 = 1, s_R9 = 0 (00)	
10 ≡ 4′	Round Σ_5 $D_R = \Sigma_5$, leggo da REG_5 $Q_R = B'_r$, scrivo in REG_6 $Q_{SB} = \Sigma_6$, scrivo in REG_5	 s_DR = 0 WR_R6 = 1, s_R6 = 1 s_IN_AS = 1, WR_R5 = 1, s_R5 = 1 (01)	0
	Start $M_4 = B_i W_r$ (mpy) $D_{1M} = B_i$, leggo da REG_6	$M/\overline{S} = 1$ s_1M = 3 (11)	

Continua nella pagina successiva

Continua dalla pagina precedente

Step #	Operazioni	Controlli	Done
	$D_{2M} = W_r$, leggo da REG_1	s_2M = 0	
	Start $\Sigma_1 = A_r + M_1$		
	$D_{1A} = A_r$, leggo da REG_3	s_1AS = 0, s_D1A = 1	
	$D_{2A} = M_1$, leggo da REG_9		
	$Q_M = M_3$, scrivo in REG_9	WR_R9 = 1, s_R9 = 0 (00)	
	Pipe $M_2 = B_i W_i$		
11 ≡ 5′	Round Σ_6		0
	$D_R = \Sigma_6$, leggo da REG_5	s_DR = 0	
	$Q_R = B_i'$, scrivo in REG_5	WR_R5 = 1, s_R5 = 2 (10)	
	Start $M_5 = 2A_r$ (shift)	$M/\overline{S} = 0$	
	$D_{1M} = A_r$, leggo da REG_3	s_1M = 0 (00)	
	Start $\Sigma_3 = A_i + M_3$		
	$D_{1A} = A_i$, leggo da REG_4	s_1AS = 1, s_D1A = 1	
	$D_{2A} = M_3$, leggo da REG_9		
	$Q_M = M_2$, scrivo in REG_9	WR_R9 = 1, s_R9 = 0 (00)	
	$Q_A = \Sigma_1$, scrivo in REG_3	s_IN_AS = 0, WR_R3 = 1, s_R3 = 1	
	Pipe $M_4 = B_i W_r$		
12	Start $M_6 = 2A_i$ (shift)	$M/\overline{S} = 0$	1
	$D_{1M} = A_i$, leggo da REG_4	s_1M = 1 (01)	
	Start $\Sigma_2 = \Sigma_1 - M_2$		
	$D_{1S} = \Sigma_1$, leggo da REG_3	s_1AS = 0, s_D1S = 2 (10)	
	$D_{2S} = M_2$, leggo da REG_9		
	$Q_M = M_4$, scrivo in REG_9	WR_R9 = 1, s_R9 = 0 (00)	
	$Q_{SH} = M_5$, scrivo in REG_7	WR_R7 = 1, s_R7 = 0	
	$Q_A = \Sigma_3$, scrivo in REG_8	WR_R8 = 1, s_R8 = 1 (01)	
6′	Start $M_6 = 2A_i$ (shift)	$M/\overline{S} = 0$	0
	$D_{1M} = A_i$, leggo da REG_4	s_1M = 1 (01)	
	Start $\Sigma_2 = \Sigma_1 - M_2$		
	$D_{1S} = \Sigma_1$, leggo da REG_3	s_1AS = 0, s_D1S = 2 (10)	
	$D_{2S} = M_2$, leggo da REG_9		
	$Q_M = M_4$, scrivo in REG_9	WR_R9 = 1, s_R9 = 0 (00)	
	$Q_{SH} = M_5$, scrivo in REG_7	WR_R7 = 1, s_R7 = 0	
	$Q_A = \Sigma_3$, scrivo in REG_8	WR_R8 = 1, s_R8 = 1 (01)	
7′	Start $\Sigma_4 = \Sigma_3 + M_4$		0
	$D_{1A} = \Sigma_3$, leggo da REG_8	s_D1A = 0	
	$D_{2A} = M_4$, leggo da REG_9		
	$Q_{SH} = M_6$, scrivo in REG_8	WR_R8 = 1, s_R8 = 0 (00)	
	$Q_{SB} = \Sigma_2$, scrivo in REG_9	WR_R9 = 1, s_R9 = 2 (10)	
8′	Round Σ_2		0
	$D_R = \Sigma_2$, leggo da REG_9	s_DR = 1	
	$Q_R = A_r'$, scrivo in REG_7	WR_R7 = 1, s_R7 = 1	
	Start $\Sigma_5 = M_5 - \Sigma_2$		
	$D_1 = M_5$, leggo da REG_7	s_D1S = 1 (01)	

Continua nella pagina successiva

Continua dalla pagina precedente

Step #	Operazioni	Controlli	Done
	$D_{2S} = \Sigma_2$, leggo da REG_9		
	$Q_A = \Sigma_4$, scrivo in REG_9	WR_R9 = 1, s_R9 = 1 (01)	
9′	Round Σ_4		0
	$D_R = \Sigma_4$, leggo da REG_9	s_DR = 1	
	$Q_R = A'_i$, scrivo in REG_8	WR_R8 = 1, s_R8 = 2 (10)	
	Start $\Sigma_6 = M_6 - \Sigma_4$		
	$D_{1S} = M_6$, leggo da REG_8	s_D1S = 0 (00)	
	$D_{2S} = \Sigma_4$, leggo da REG_9		
	$Q_{SB} = \Sigma_5$, scrivo in REG_5	s_IN_AS = 1, WR_R5 = 1, s_R5 = 1 (01)	
10′	Round Σ_5		0
	$D_R = \Sigma_5$, leggo da REG_5	s_DR = 0	
	$Q_R = B'_r$, scrivo in REG_6	WR_R6 = 1, s_R6 = 1	
	$Q_{SB} = \Sigma_6$, scrivo in REG_5	s_IN_AS = 1, WR_R5 = 1, s_R5 = 1 (01)	
11′	Round Σ_6		0
	$D_R = \Sigma_6$, leggo da REG_5	s_DR = 0	
	$Q_R = B'_i$, scrivo in REG_5	WR_R5 = 1, s_R5 = 2 (10)	
12′			1

6 Calcolatore FFT

Una volta terminata la realizzazione del processore Butterfly, la cui descrizione in VHDL è riportata nell'appendice è possibile utilizzarlo come elemento base per la costruzione di un calcolatore di FFT.

Lo schema della macchina, nell'implementazione 16×16 è riportato in Figura 6.2; come detto, l'unica difficoltà di questa parte consiste nel comprendere come collegare e sincronizzare le 32 Butterfly che compongono l'architettura:

- Vi sono quattro livelli di gerarchia, ognuno costituito da otto Butterfly;
- Le Butterfly che lavorano con lo stesso twiddle factor si dicono appartenenti allo stesso *gruppo*;
- Il numero di gruppo presenti al livello i è pari a $2i$ (rispettivamente 1, 2, 4 e 8 gruppi);
- Ad ogni passo dell'elaborazione, si ricomincia l'ordine dei twiddle factor secondo la logica del *bit reverse order* (0, 4, 2, 6, 1, 5, 3, 7).

Inoltre, è importante sottolineare quanto segue:

- Tutte le Butterfly del primo livello lavorano con il Twiddle Factor W^0 e ricevono i dati dall'esterno;
- Le Butterfly dei livelli successivi al primo ricevono i dati dalle Butterfly del livello precedente;

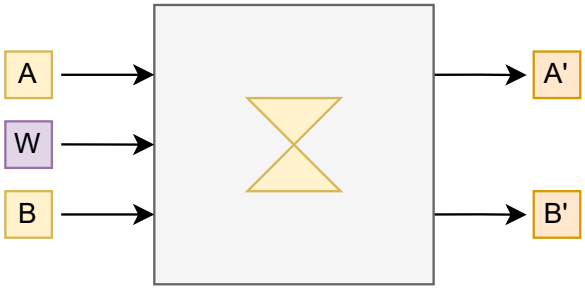


Figura 6.1: Schema semplificato delle connessioni I/O del processore Butterfly.

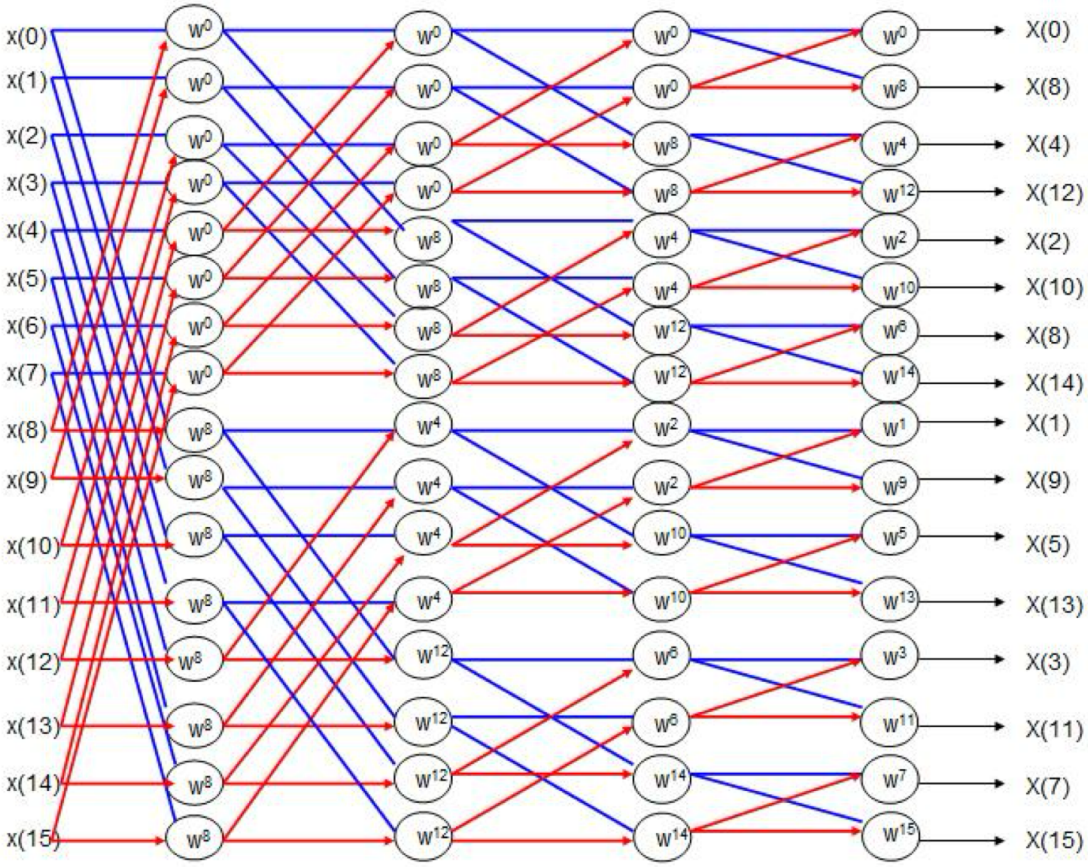


Figura 6.2: Schema calcolatore FFT 16×16.

- Lo START e il DONE sono condivisi da tutte le Butterfly appartenenti allo stesso livello: in particolare:
  - **Primo livello**: lo START è un impulso inviato dall'esterno insieme ai dati;
  - **Livelli successivi al primo**: lo START coincide con il DONE del livello precedente.

Parte II
Test

7 Test Butterfly

Il test del funzionamento del processore Butterfly è stato automatizzato con MATLAB® e C++ in modo da ottenere risultati il più possibile completi. I codici e le testbench utilizzati sono riportati nell'Appendice B.

La simulazione è suddivisa in due parti:

Prima parte. Consiste nell'esecuzione dello script MATLAB® `test.m`, all'interno del quale vengono calcolati i valori dei twiddle factor e alcune combinazioni significative dei vettori di input. L'idea è calcolare la dinamica di ingresso e di uscita delle Butterfly appartenenti al calcolatore di FFT 16×16, in modo da verificare il funzionamento del processore nelle condizioni peggiori in cui si troverà ad operare. Sono stati pertanto analizzati i seguenti casi:

1. Butterfly I livello con $W = W^0$ (L1_W0): dinamica $[-0.5, +0.5]$;
2. Butterfly II livello con $W = W^0$ (L2_W0): dinamica fissata dalle uscite di L1_W0;
3. Butterfly II livello con $W = W^4$ (L2_W4): dinamica fissata dalle uscite di L1_W0;
4. Butterfly III livello con $W = W^0$ (L3_W0): dinamica fissata dalle uscite di L2_W0;
5. Butterfly III livello con $W = W^4$ (L3_W4): dinamica fissata dalle uscite di L2_W0;
6. Butterfly III livello con $W = W^2$ (L3_W2): dinamica fissata dalle uscite di L2_W4;
7. Butterfly III livello con $W = W^6$ (L3_W6): dinamica fissata dalle uscite di L2_W4;
8. Butterfly IV livello con $W = W^0$ (L4_W0): dinamica fissata dalle uscite di L3_W0;
9. Butterfly IV livello con $W = W^4$ (L4_W4): dinamica fissata dalle uscite di L3_W0;
10. Butterfly IV livello con $W = W^2$ (L4_W2): dinamica fissata dalle uscite di L3_W4;
11. Butterfly IV livello con $W = W^6$ (L4_W6): dinamica fissata dalle uscite di L3_W4;
12. Butterfly IV livello con $W = W^1$ (L4_W1): dinamica fissata dalle uscite di L3_W2;
13. Butterfly IV livello con $W = W^5$ (L4_W5): dinamica fissata dalle uscite di L3_W2;
14. Butterfly IV livello con $W = W^3$ (L4_W3): dinamica fissata dalle uscite di L3_W6;
15. Butterfly IV livello con $W = W^7$ (L4_W7): dinamica fissata dalle uscite di L3_W6.

L'output del programma MATLAB® è una serie di file di testo contenenti i dati da fornire in ingresso alla Butterfly per verificarne il corretto comportamento;

Seconda parte. Consiste nell'esecuzione di uno script C++ che, grazie ai principi della programmazione ad oggetti, effettua opportune sostituzioni all'interno della testbench per consentire l'accesso ai file corretti per ogni simulazione, avvia automaticamente ModelSim e confronta i file di output prodotti dalle simulazioni con quelli generati da MATLAB®.

L'esecuzione del programma di collaudo ha permesso di verificare la correttezza dei valori calcolati sia in modalità "single" sia in modalità "continuous".

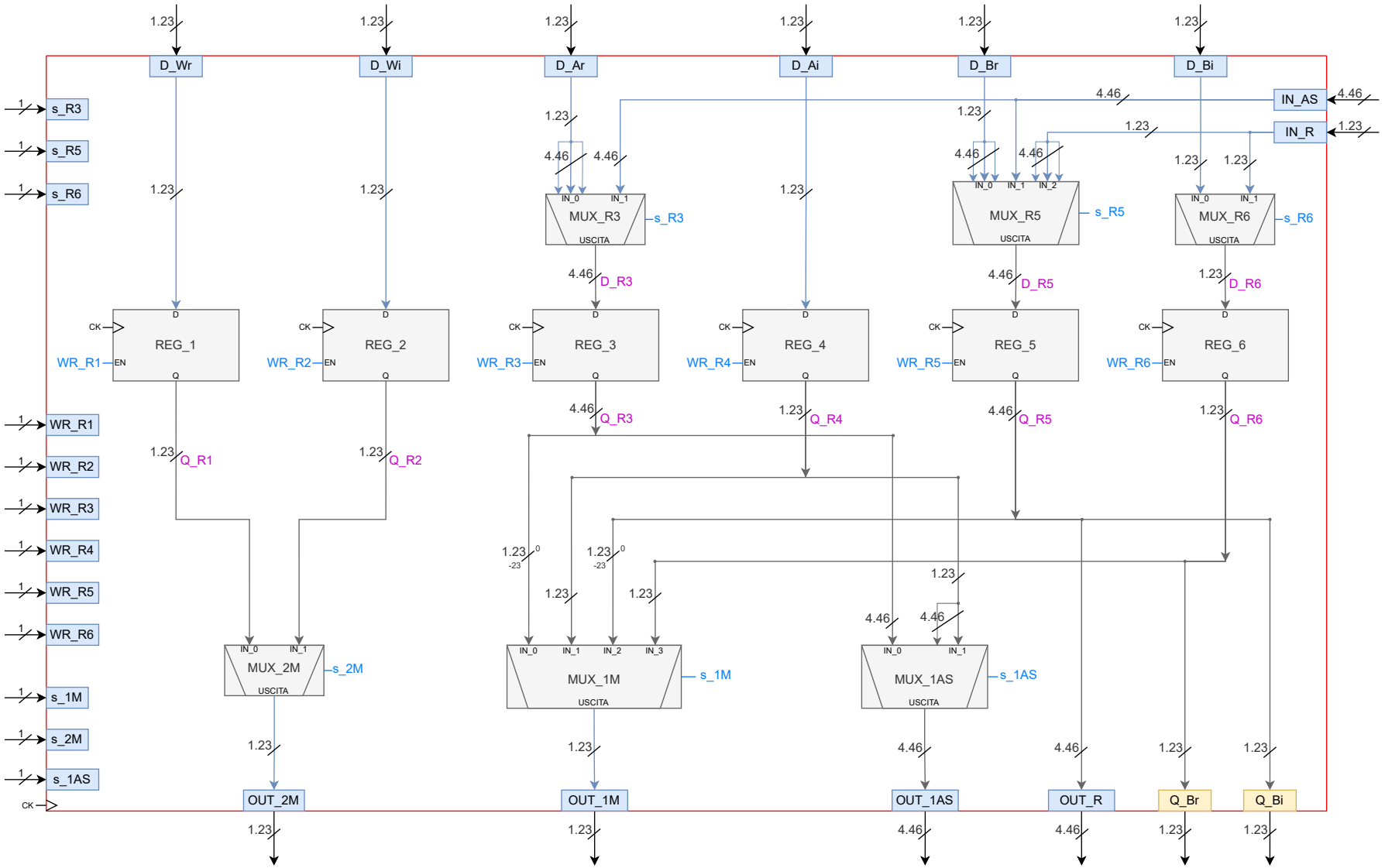
Nella simulazione automatizzata, è stata testata solo la modalità di lavoro continua, che sarà di fatto quella adottata dal calcolatore di FFT; dovendo testare anche la modalità isolata, è stata scritta una semplice testbench, in cui vengono inviati dei valori casuali all'interno della dinamica prevista; in questo modo, è stato possibile vrificare che entrambe le modalità di lavoro funzionano correttamente.

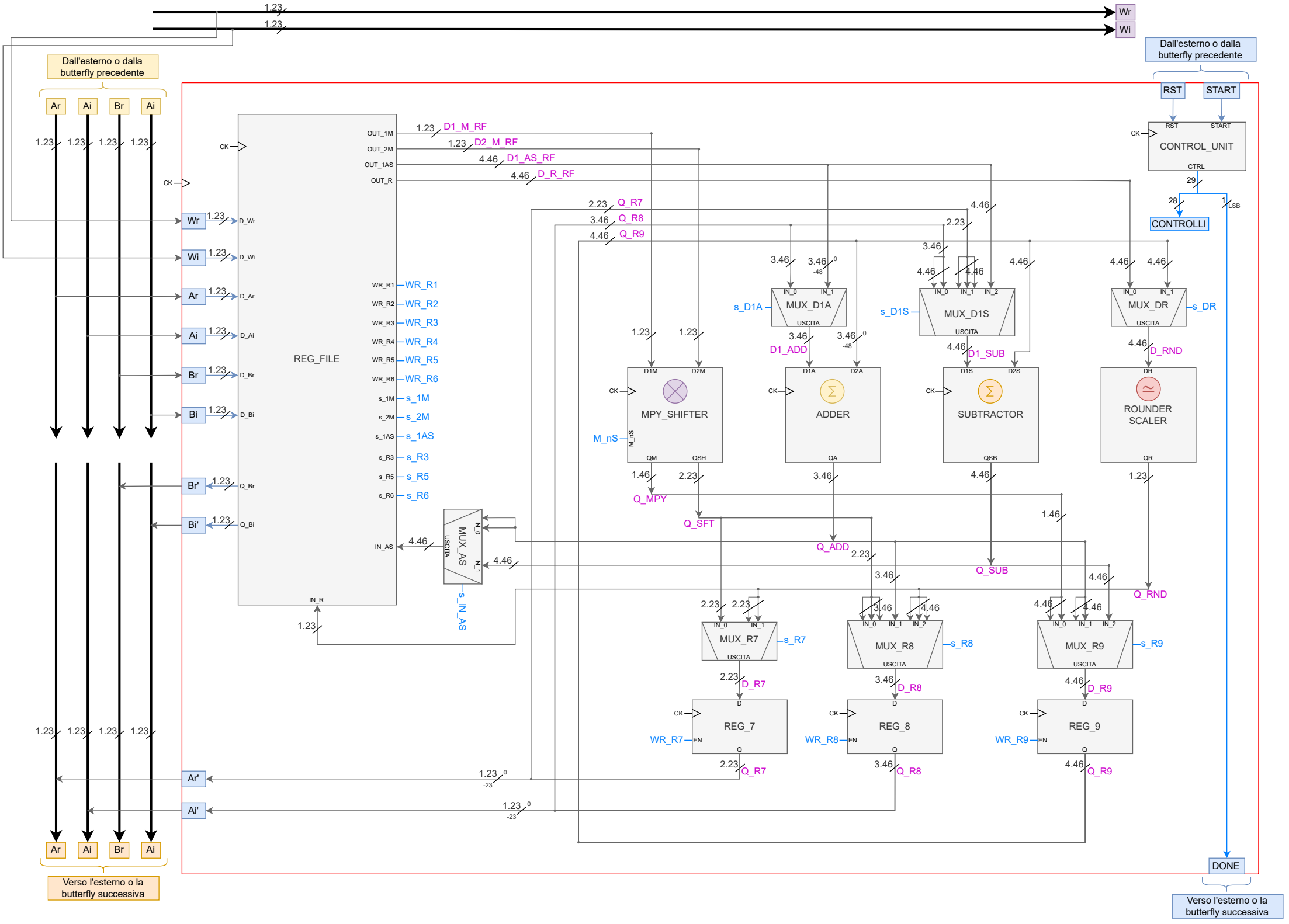
8 Test FFT

Il corretto funzionamento del calcolatore di FFT è stato testato utilizzando i vettori di ingresso riportati nel file `input_data_fft.txt`. Tutti i vettori sono costituiti da numeri ai limiti della dinamica, in modo da far lavorare l'unità di elaborazione nelle condizioni peggiori possibili.

Confrontando i risultati generati dalla testbench con quelli ottenuti mediante la funzione `fft()` di MATLAB®, è stato possibile verificare la correttezza dei risultati ottenuti.


DFD Butterfly - modalità di esecuzione continua
Step 0
Step 1
Step 2
Step 3
Step 4
Step 5
Step 6
Step 7
Step 8
Step 9
Step 10
Step 11
Step 12
Ar
Br
Wr
Bi
Wi
Ar
Ai
Br
Wi
Bi
Wr
Ai
START
M1
M3
M2
M4
Σ1
M5
Σ3
M6
Σ2
Σ4
Ar'
Σ5
Ai'
Σ6
Br'
Bi'
DONE
Ar'
Br'
Ai'
Bi'
Step 0'
Step 1'
Step 2'
Step 3'
Step 4'
Step 5'
Step 6'
Step 7'
Step 8'
Step 9'
Step 10'
Step 11'
Step 12'


D_Wr
D_Wi
D_Ar
D_Ai
D_Br
D_Bi
IN_AS
IN_R
s_R3
s_R5
s_R6
WR_R1
WR_R2
WR_R3
WR_R4
WR_R5
WR_R6
s_1M
s_2M
s_1AS
CK
REG_1
REG_2
REG_3
REG_4
REG_5
REG_6
MUX_R3
MUX_R5
MUX_R6
MUX_2M
MUX_1M
MUX_1AS
USCITA
D_R3
D_R5
D_R6
Q_R1
Q_R2
Q_R3
Q_R4
Q_R5
Q_R6
OUT_2M
OUT_1M
OUT_1AS
OUT_R
Q_Br
Q_Bi


Dall'esterno o dalla butterfly precedente
Ar
Ai
Br
Ai
1.23
Wr
Wi
RST
START
CONTROL_UNIT
CTRL
29
28
1
LSB
CONTROLLI
CK
REG_FILE
OUT_1M
OUT_2M
OUT_1AS
OUT_R
D1_M_RF
D2_M_RF
D1_AS_RF
D_R_RF
D_Wr
D_Wi
D_Ar
D_Ai
D_Br
D_Bi
Q_Br
Q_Bi
WR_R1
WR_R2
WR_R3
WR_R4
WR_R5
WR_R6
s_1M
s_2M
s_1AS
s_R3
s_R5
s_R6
IN_AS
IN_R
MUX_AS
s_IN_AS
Q_R7
Q_R8
Q_R9
MUX_D1A
s_D1A
D1_ADD
MUX_D1S
s_D1S
D1_SUB
MUX_DR
s_DR
D_RND
MPY_SHIFTER
M_nS
Q_MPY
ADDER
Q_ADD
SUBTRACTOR
Q_SUB
ROUNDER SCALER
Q_RND
Q_SFT
MUX_R7
s_R7
MUX_R8
s_R8
MUX_R9
s_R9
D_R7
D_R8
D_R9
REG_7
REG_8
REG_9
WR_R7
WR_R8
WR_R9
Ar'
Ai'
DONE
Verso l'esterno o la butterfly successiva

A Descrizione dell'hardware

A.1 Component Execution Unit

A.1.1 Registro di pipe *sfixed*

```
--*******************************************************************************
--* Registro di pipe per numeri signed fixed point
--*******************************************************************************

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.fixed_pkg.all;

entity pipe_sfixed is
    generic (
        M : integer := 0; -- MSB
        N : integer := 24 -- lunghezza numero [bit]
    );
    port (
        ck : in std_logic;                    -- clock
        d  : in sfixed(M downto (M - N + 1)); -- ingresso
        q  : out sfixed(M downto (M - N + 1)) -- uscita
    );
end entity pipe_sfixed;

architecture structure of pipe_sfixed is

begin
    CK_process : process (ck)
    begin
        if (ck'event and ck = '1') then q <= d;
        end if;
    end process CK_process;

end architecture structure;
```

A.1.2 Registro *sfixed* con enable

```
--*******************************************************************************
--* Registro per memorizzazione di numeri signed fixed point
--*******************************************************************************

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.fixed_pkg.all;

entity register_sfixed is
    generic (
        M : integer := 0; -- MSB
        N : integer := 24 -- lunghezza numero [bit]
    );
    port (
        ck : in std_logic;                    -- clock
```

```
        en : in std_logic;                    -- enable (attivo alto)
        d  : in sfixed(M downto (M - N + 1)); -- ingresso
        q  : out sfixed(M downto (M - N + 1)) -- uscita
    );
end entity register_sfixed;

architecture structure of register_sfixed is

begin
    CK_process : process (ck)
    begin
        if (ck'event and ck = '1') then
            if (en = '1') then
                q <= d;
            end if;
        end if;
    end process CK_process;
end architecture structure;
```

A.1.3 Multiplexer *sfixed* a due vie

```
--*****************************************************************************
--* Multiplexer a due vie con ingressi e uscita su N bit signed fixed point
--* s=0: out_mux = IN_0
--* s=1: out_mux = IN_1
--*****************************************************************************

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.fixed_pkg.all;

entity mux2to1_sfixed is
    generic (
        M : integer; -- MSB
        N : integer  -- lunghezza numero [bit]
    );
    port (
        s          : in std_logic;                    -- selettore a 1 bit
        IN_0, IN_1 : in sfixed(M downto (M - N + 1)); -- input a N bit Q(M+1).(N-M-1)
        uscita     : out sfixed(M downto (M - N + 1)) -- output a N bit Q(M+1).(N-M-1)
    );
end entity mux2to1_sfixed;

architecture structure of mux2to1_sfixed is
begin
    uscita <= IN_0 when s = '0' else
        IN_1;
end structure;
```

A.1.4 Multiplexer *sfixed* a quattro vie

```
--*****************************************************************************
--* Multiplexer a quattro vie con ingressi e uscita a N bit signed fixed point
--* s=00: out_mux = IN_0 (0)
--* s=01: out_mux = IN_1 (1)
```

```
--* s=10: out_mux = IN_2 (2)
--* s=11: out_mux = IN_3 (3)
--********************************************************************************

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.fixed_pkg.all;

entity mux4to1_sfixed is
    generic (
        M : integer; -- MSB
        N : integer  -- lunghezza numero [bit]
    );
    port (
        s                       : in std_logic_vector(1 downto 0); -- selettore a 2 bit
        IN_0, IN_1, IN_2, IN_3 : in sfixed(M downto (M - N + 1)); -- input a N bit Q(M+1).(N-M-1)
        uscita                  : out sfixed(M downto (M - N + 1)) -- output a N bit Q(M+1).(N-M-1)
    );
end entity mux4to1_sfixed;

architecture structure of mux4to1_sfixed is
begin
    uscita <=
        IN_0 when s = "00" else --0
        IN_1 when s = "01" else --1
        IN_2 when s = "10" else --2
        IN_3;                   --3
end structure;
```

A.1.5 Sommatore *sfixed* pipelinato

```
--********************************************************************************
--* Sommatore di numeri signed fixed point con un livello di pipeline
--********************************************************************************

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.fixed_pkg.all;

entity adder_pipe_sfixed is
    generic (
        M : integer := 2; -- MSB
        N : integer := 49 -- lunghezza numero [bit]
    );
    port (
        CK       : in std_logic;                    -- clock
        D1A, D2A : in sfixed(M downto (M - N + 1)); -- ingressi (default Q3.46)
        QA       : out sfixed(M downto (M - N + 1)) -- uscita (default Q3.46)
    );
end entity adder_pipe_sfixed;

architecture structure of adder_pipe_sfixed is

```

```
    -- somma non pipelinata
    signal QA_async : sfixed(M downto (M - N + 1));

    -- registro di pipe
    component pipe_sfixed is
        generic (
            M : integer := 0;
            N : integer := 24
        );
        port (
            ck : in std_logic;
            d  : in sfixed(M downto (M - N + 1));
            q  : out sfixed(M downto (M - N + 1))
        );
    end component;

begin

    -- N.B. Q2.46 + Q2.46 = Q3.46
    QA_async <= D1A((M - 1) downto (M - N + 1)) + D2A((M - 1) downto (M - N + 1));

    PIPE : pipe_sfixed
    generic map(M => M, N => N)
    port map(
        ck => CK,
        d  => QA_async, -- default Q3.46
        q  => QA        -- default Q3.46
    );

end architecture structure;
```

A.1.6 Sottrattore *sfixed* pipelinato

```
--*******************************************************************************
--* Sottrazione di numeri signed fixed point con un livello di pipeline
--*******************************************************************************

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.fixed_pkg.all;

entity subtractor_pipe_sfixed is
    generic (
        M : integer := 3; -- MSB
        N : integer := 50 -- lunghezza numero [bit]
    );
    port (
        CK       : in std_logic;                     -- clock
        D1S, D2S : in sfixed(M downto (M - N + 1)); -- ingressi (default Q4.46)
        QSB      : out sfixed(M downto (M - N + 1)) -- uscita (default Q4.46)
    );
end entity subtractor_pipe_sfixed;

architecture structure of subtractor_pipe_sfixed is

    -- differenza non pipelinata
```

```
25      signal QSB_async : sfixed(M downto (M - N + 1));
26
27      -- registro di pipe
28      component pipe_sfixed is
29          generic (
30              M : integer := 0;
31              N : integer := 24
32          );
33          port (
34              ck : in std_logic;
35              d  : in sfixed(M downto (M - N + 1));
36              q  : out sfixed(M downto (M - N + 1))
37          );
38      end component;
39
40  begin
41
42      -- N.B. Q3.46 - Q3.46 = Q4.46
43      QSB_async <= D1S((M - 1) downto (M - N + 1)) - D2S((M - 1) downto (M - N + 1));
44
45      PIPE : pipe_sfixed
46      generic map(M => M, N => N)
47      port map(
48          ck => CK,
49          d  => QSB_async, -- default Q4.46
50          q  => QSB        -- default Q4.46
51      );
52
53  end architecture structure;
```

A.1.7 Moltiplicatore/shifter pipelinato con ingressi fractional point

```
1   --*****************************************************************************
2   --* Moltiplicatore/shifter di numeri sfixed fractional point
3   --* Due livelli di pipeline per il moltiplicatore (M_nS=1)
4   --* Un livello di pipeline per lo shifter (M_nS=0)
5   --*****************************************************************************
6
7   library ieee;
8   use ieee.std_logic_1164.all;
9   use ieee.numeric_std.all;
10  use ieee.fixed_pkg.all;
11
12  entity mpy_shifter_pipe_sfixed is
13      generic (N : integer := 24); -- lunghezza numero [bit]
14      port (
15          CK       : in std_logic;                      -- clock
16          M_nS     : in std_logic;                      -- operation mode (1 mpy, 0 shift)
17          D1M, D2M : in sfixed(0 downto (-N + 1));      -- ingressi (default Q1.23)
18          QM       : out sfixed(0 downto (-2 * N + 2)); -- uscita moltiplicatore (default
            Q1.46)
19          QSH      : out sfixed(1 downto (-N + 1))      -- uscita shifter (default Q2.23)
20      );
21  end entity mpy_shifter_pipe_sfixed;
22
23  architecture structure of mpy_shifter_pipe_sfixed is
24
```

```
25      -- definizione segnali interni
26      signal QM_async  : sfixed(1 downto (-2 * N + 2)); -- moltiplicazione non pipelinata
27      signal QM_pipe1  : sfixed(0 downto (-2 * N + 2)); -- moltiplicazione dopo 1 stadio
        di pipe
28      signal QSH_async : sfixed(1 downto (-N + 1));     -- shift non pieplinato
29
30      -- registro di pipe
31      component pipe_sfixed is
32          generic (
33              M : integer := 0;
34              N : integer := 24
35          );
36          port (
37              ck : in std_logic;
38              d  : in sfixed(M downto (M - N + 1));
39              q  : out sfixed(M downto (M - N + 1))
40          );
41      end component;
42
43  begin
44
45      -- process per scegliere tra moltiplicazione e shift in funzione di M_nS
46      -- la sensitivity list deve contenere sia il segnale di controllo, sia gli ingressi
        del blocco!!!
47      MPY_SHIFT_PROCESS: process(M_nS, D1M, D2M)
48      begin
49          if (M_nS = '1') then -- multiply (N.B. Q1.23 * Q1.23 = Q2.46)
50              QM_async  <= D1M * D2M;
51              QSH_async <= (others => '0');
52          else -- shift
53              QM_async <= (others => '0');
54              QSH_async <= D1M & '0';
55          end if;
56      end process MPY_SHIFT_PROCESS;
57
58      PIPE_1_MPY : pipe_sfixed -- primo stadio di pipeline mpy
59      generic map(M => 0, N => 2*N-1)
60      port map(
61          ck => CK,
62          d  => QM_async(0 downto (-2 * N + 2)),
63          q  => QM_pipe1
64      );
65
66      PIPE_2_MPY : pipe_sfixed -- secondo stadio di pipeline mpy
67      generic map(M => 0, N => 2*N-1)
68      port map(
69          ck => CK,
70          d  => QM_pipe1,
71          q  => QM
72      );
73
74      PIPE_SHIFT : pipe_sfixed -- stadio di pipeline shift
75      generic map(M => 1, N => N+1)
76      port map(
77          ck => CK,
78          d  => QSH_async,
79          q  => QSH
80      );
81
```

```
82  end architecture structure;
```

A.1.8 Rounder

```
1   --*****************************************************************************
2   --* Blocco HW per arrotondamento e scalamento di numeri signed fixed point
3   --* Arrotondamento in forma Q1.23 secondo la tecnica del rounding to nearest even
4   --* Scalamento secondo la tecnica del "Unconditional Block Floating Point Scaling"
5   --* Si assume che lo scalamento da implementare sia sempre di 1 bit
6   --*****************************************************************************
7
8   library ieee;
9   use ieee.std_logic_1164.all;
10  use ieee.numeric_std.all;
11  use ieee.fixed_pkg.all;
12
13  entity rounder_sfixed is
14      generic (
15          M : integer := 3; -- MSB
16          N : integer := 50 -- lunghezza numero [bit]
17      );
18      port (
19          DR : in sfixed(M downto (M - N + 1)); -- ingresso (default Q4.46)
20          QR : out sfixed(0 downto -23)         -- uscita Q1.23
21      );
22  end entity rounder_sfixed;
23
24  architecture structure of rounder_sfixed is
25
26      -- definizione segnali interni
27      signal x1, x2, x3 : std_logic;            -- bit -23, -24 e -25 di DR
28      signal mA, mB, mC : std_logic;            -- minterm
29      signal A, B, C, f : sfixed(3 downto -23); -- Q4.23
30      signal LSB_one    : sfixed(2 downto -23); -- Q2.23
31
32  begin
33
34      x1 <= DR(-23);
35      x2 <= DR(-24);
36      x3 <= DR(-25);
37
38      -- somme di minterm ottenute dalla K-map
39      mA <= (not x1) and ((not x2) or (x2 and (not x3)));
40      mB <= (x1 and (not x2)) or ((not x1) and x2 and x3);
41      mC <= x1 and x2;
42
43      -- possibili risultati del rounding to nearest even in funzione di x1, x2 e x3
44      A       <= DR(3 downto -23);
45      B       <= DR(3 downto -22) & '1';
46      LSB_one <= (-23 => '1', others => '0');
47      C       <= DR(2 downto -23) + LSB_one;
48
49      -- process per scegliere il valore di f tra A, B e C
50      -- f = (mA * A) + (mB * B) + (mC * C)
51      ROUND_PROCESS : process (mA, mB, mC, A, B, C)
52      begin
53          if (mA = '1') then
```

```
            f <= A;
        elsif (mB = '1') then
            f <= B;
        elsif (mC = '1') then
            f <= C;
        end if;
    end process ROUND_PROCESS;

    -- eliminazione dei bit 3 e 2 + scalamento di 1 bit per ricondursi alla forma Q1.23
    QR <= f(1 downto -22);

end architecture structure;
```

A.2 Component Control Unit

A.2.1 Registro sensibile ai fronti di salita

```
--*******************************************************************************
--* Registro sensibile ai fronti di salita del clock
--*******************************************************************************

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity reg_rising is
  generic (N : integer);
  port (
    clk : in std_logic;
    d   : in std_logic_vector(N - 1 downto 0);
    q   : out std_logic_vector(N - 1 downto 0)
  );
end entity reg_rising;

architecture structure of reg_rising is
begin

  CK_process : process (clk)
  begin
    if (clk'event and clk = '1') then
      q <= d;
    end if;
  end process CK_process;

end structure;
```

A.2.2 Registro sensibile ai fronti di discesa

```
--*******************************************************************************
--* Registro sensibile ai fronti di discesa del clock
--*******************************************************************************

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

```

```
entity reg_falling is
  generic (N : integer);
  port (
    clk : in std_logic;
    rst : in std_logic;
    d   : in std_logic_vector(N - 1 downto 0);
    q   : out std_logic_vector(N - 1 downto 0)
  );
end entity reg_falling;

architecture structure of reg_falling is
begin

  CK_process : process (clk, rst)
  begin
    if (rst = '1') then
      q <= (others => '0');
    elsif (clk'event and clk = '0') then
      q <= d;
    end if;
  end process CK_process;

end structure;
```

A.2.3 Multiplexer a due vie

```
--*******************************************************************************
--* Multiplexer a due vie con ingressi e uscita su N bit
--* s=0: out_mux = IN_0
--* s=1: out_mux = IN_1
--*******************************************************************************

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity mux2to1 is
  generic (N : integer := 16);
  port (
    s          : in std_logic;                          -- selettore a 1 bit
    IN_0, IN_1 : in std_logic_vector(N - 1 downto 0); -- input a N bit
    uscita     : out std_logic_vector(N - 1 downto 0) -- output a N bit
  );
end mux2to1;

architecture structure of mux2to1 is
begin
  uscita <= IN_0 when s = '0' else
    IN_1;
end structure;
```

A.3 Processore Butterfly

A.3.1 Register file

```
--*******************************************************************************
--* Register file per memorizzazione di ingressi e risultati parziali
--*******************************************************************************

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.fixed_pkg.all;

entity register_file is
    port (
        CK  : in std_logic;
        --* segnali di controllo ***************************************************
        WR_R1 : in std_logic;
        WR_R2 : in std_logic;
        WR_R3 : in std_logic;
        WR_R4 : in std_logic;
        WR_R5 : in std_logic;
        WR_R6 : in std_logic;
        ------------------------------------------------------------------------
        s_R3 : in std_logic;
        s_R5 : in std_logic_vector(1 downto 0);
        s_R6 : in std_logic;
        ------------------------------------------------------------------------
        s_1M  : in std_logic_vector(1 downto 0);
        s_2M  : in std_logic;
        s_1AS : in std_logic;
        --* porte di ingresso *****************************************************
        D_Wr : in sfixed(0 downto -23); -- Q1.23
        D_Wi : in sfixed(0 downto -23); -- Q1.23
        ------------------------------------------------------------------------
        D_Ar : in sfixed(0 downto -23); -- Q1.23
        D_Ai : in sfixed(0 downto -23); -- Q1.23
        D_Br : in sfixed(0 downto -23); -- Q1.23
        D_Bi : in sfixed(0 downto -23); -- Q1.23
        ------------------------------------------------------------------------
        IN_AS : in sfixed(3 downto -46); -- Q4.46
        IN_R  : in sfixed(0 downto -23); -- Q1.23
        --* porte di uscita ******************************************************
        Q_Br : out sfixed(0 downto -23); -- Q1.23
        Q_Bi : out sfixed(0 downto -23); -- Q1.23
        ------------------------------------------------------------------------
        OUT_1M  : out sfixed(0 downto -23); -- Q1.23
        OUT_2M  : out sfixed(0 downto -23); -- Q1.23
        OUT_1AS : out sfixed(3 downto -46); -- Q4.46
        OUT_R   : out sfixed(3 downto -46)  -- Q4.46
    );
end entity register_file;

architecture structure of register_file is

    --*******************************************************************************
    --* Definizione segnali interni
    --*******************************************************************************
```

```
55
56      signal D_Ar_50 : sfixed(3 downto -46); -- D_Ar Q3.46 (IN_0 mux_R3)
57      signal D_Br_50 : sfixed(3 downto -46); -- D_Br Q3.46 (IN_0 mux_R5)
58      signal IN_R_50 : sfixed(3 downto -46); -- IN_R Q3.46 (IN_2 mux_R5)
59      ------------------------------------------------------------------------------
60      signal D_R3 : sfixed(3 downto -46); -- ingresso REG_3 Q3.46 (uscita mux_R3)
61      signal D_R5 : sfixed(3 downto -46); -- ingresso REG_5 Q3.46 (uscita mux_R5)
62      signal D_R6 : sfixed(0 downto -23); -- ingresso REG_6 Q1.23 (uscita mux_R6)
63      ------------------------------------------------------------------------------
64      signal Q_R1    : sfixed(0 downto -23); -- uscita REG_1 Q1.23 (IN_0 mux_2M)
65      signal Q_R2    : sfixed(0 downto -23); -- uscita REG_2 Q1.23 (IN_1 mux_2M)
66      signal Q_R3    : sfixed(3 downto -46); -- uscita REG_3 Q4.46 (IN_0 mux_1AS)
67      signal Q_R3_24 : sfixed(0 downto -23); -- uscita REG_3 Q1.23 (IN_0 mux_1M)
68      signal Q_R4    : sfixed(0 downto -23); -- uscita REG_4 Q1.23 (IN_1 mux_1M)
69      signal Q_R4_50 : sfixed(3 downto -46); -- uscita REG_4 Q4.46 (IN_1 mux_1AS)
70      signal Q_R5    : sfixed(3 downto -46); -- uscita REG_5 Q4.46 (out OUT_R)
71      signal Q_R5_24 : sfixed(0 downto -23); -- uscita REG_5 Q1.23 (IN_2 mux_1M, out Q_Bi)
72      signal Q_R6    : sfixed(0 downto -23); -- uscita REG_6 Q1.23 (IN_3 mux_1M, out Q_Br)
73      ------------------------------------------------------------------------------
74      signal zero : sfixed(3 downto -46);
75
76      --*****************************************************************************
77      --* Dichiarazione component
78      --*****************************************************************************
79
80      -- registro
81      component register_sfixed is
82          generic (
83              M : integer := 0; -- MSB
84              N : integer := 24 -- lunghezza numero [bit]
85          );
86          port (
87              ck : in std_logic;
88              en : in std_logic;
89              d  : in sfixed(M downto (M - N + 1));
90              q  : out sfixed(M downto (M - N + 1))
91          );
92      end component;
93
94      -- mux a due vie
95      component mux2to1_sfixed is
96          generic (
97              M : integer; -- MSB
98              N : integer  -- lunghezza numero [bit]
99          );
100         port (
101             s          : in std_logic;
102             IN_0, IN_1 : in sfixed(M downto (M - N + 1));
103             uscita     : out sfixed(M downto (M - N + 1))
104         );
105     end component;
106
107     -- mux a 4 vie
108     component mux4to1_sfixed is
109         generic (
110             M : integer; -- MSB
111             N : integer  -- lunghezza numero [bit]
112         );
113         port (
```

```
            s                       : in std_logic_vector(1 downto 0);
            IN_0, IN_1, IN_2, IN_3 : in sfixed(M downto (M - N + 1));
            uscita                  : out sfixed(M downto (M - N + 1))
        );
    end component;

begin

    -- vettore di zeri
    zero <= (others => '0');
    -- estensioni da Q1.23 a Q4.46
    D_Ar_50 <= D_Ar(0) & D_Ar(0) & D_Ar(0) & D_Ar & zero(-24 downto -46);
    D_Br_50 <= D_Br(0) & D_Br(0) & D_Br(0) & D_Br & zero(-24 downto -46);
    IN_R_50 <= IN_R(0) & IN_R(0) & IN_R(0) & IN_R & zero(-24 downto -46);
    Q_R4_50 <= Q_R4(0) & Q_R4(0) & Q_R4(0) & Q_R4 & zero(-24 downto -46);
    -- riduzioni da Q4.46 a Q1.23
    Q_R3_24 <= Q_R3(0 downto -23);
    Q_R5_24 <= Q_R5(0 downto -23);

    --*************************************************************************************
    --* Istanze component
    --*************************************************************************************

    mux_R3 : mux2to1_sfixed
    generic map(M => 3, N => 50) -- Q4.46
    port map(
        s      => s_R3,
        IN_0   => D_Ar_50,
        IN_1   => IN_AS,
        uscita => D_R3
    );

    mux_R5 : mux4to1_sfixed
    generic map(M => 3, N => 50) -- Q4.46
    port map(
        s      => s_R5,
        IN_0   => D_Br_50,
        IN_1   => IN_AS,
        IN_2   => IN_R_50,
        IN_3   => zero,
        uscita => D_R5
    );

    mux_R6 : mux2to1_sfixed
    generic map(M => 0, N => 24) -- Q1.23
    port map(
        s      => s_R6,
        IN_0   => D_Bi,
        IN_1   => IN_R,
        uscita => D_R6
    );

    REG_1 : register_sfixed
    generic map(M => 0, N => 24) -- Q1.23
    port map(
        ck => CK,
        en => WR_R1,
        d  => D_Wr,
        q  => Q_R1
```

```
    );

    REG_2 : register_sfixed
    generic map(M => 0, N => 24) -- Q1.23
    port map(
        ck => CK,
        en => WR_R2,
        d  => D_Wi,
        q  => Q_R2
    );

    REG_3 : register_sfixed
    generic map(M => 3, N => 50) -- Q4.46
    port map(
        ck => CK,
        en => WR_R3,
        d  => D_R3,
        q  => Q_R3
    );

    REG_4 : register_sfixed
    generic map(M => 0, N => 24) -- Q1.23
    port map(
        ck => CK,
        en => WR_R4,
        d  => D_Ai,
        q  => Q_R4
    );

    REG_5 : register_sfixed
    generic map(M => 3, N => 50) -- Q4.46
    port map(
        ck => CK,
        en => WR_R5,
        d  => D_R5,
        q  => Q_R5
    );

    REG_6 : register_sfixed
    generic map(M => 0, N => 24) -- Q1.23
    port map(
        ck => CK,
        en => WR_R6,
        d  => D_R6,
        q  => Q_R6
    );

    mux_1M : mux4to1_sfixed
    generic map(M => 0, N => 24) -- Q1.23
    port map(
        s      => s_1M,
        IN_0   => Q_R3_24,
        IN_1   => Q_R4,
        IN_2   => Q_R5_24,
        IN_3   => Q_R6,
        uscita => OUT_1M
    );

    mux_2M : mux2to1_sfixed
```

```
    generic map(M => 0, N => 24) -- Q1.23
    port map(
        s      => s_2M,
        IN_0   => Q_R1,
        IN_1   => Q_R2,
        uscita => OUT_2M
    );

    mux_1AS : mux2to1_sfixed
    generic map(M => 3, N => 50) -- Q4.46
    port map(
        s      => s_1AS,
        IN_0   => Q_R3,
        IN_1   => Q_R4_50,
        uscita => OUT_1AS
    );

    Q_Br  <= Q_R6;
    Q_Bi  <= Q_R5_24;
    OUT_R <= Q_R5;

end architecture structure;
```

A.3.2 Micro-ROM

```
--*******************************************************************************
--* Micro ROM per la microprogrammazione della control unit
--* Ogni riga è costituita da 36 bit
--* Bit 35-7: segnali di controllo che governano il funzioanmento del circuito
--* Bit 6-2: jump address
--* Bit 1-0: condition code
--*******************************************************************************

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity uROM_butterfly is
    port (
        ADDR : in integer range 0 to 19;
        CELL : out std_logic_vector(42 downto 0)
    );
end entity uROM_butterfly;

architecture structure of uROM_butterfly is

    type rom is array (0 to 19) of std_logic_vector(42 downto 0);
    constant micro_rom : rom := (
        "11111100001000000000000000000000100000000",
        "00000000001000000000001000000000000000000",
        "00000000001010000000001000000000000000000",
        "00000000001110001000001000000000000000000",
        "00000000001100001000001100000000000000000",
        "00100010000001001000000100000011011000000",
        "11111100000100111001000010000000000000000",
        "00000000001000011000101000000000000000000",
        "00000000001010101100011001100000000000000",
```

```
        "00001000101110011010001000110000000000000000",
        "00001100111110000100000110001000000000000000",
        "00101011000001001000000100000000000000000000",
        "00000000000100111001000010001001101101110110",
        "00000000000100111001000010000000000000000000",
        "00000000000000011000100000000000000000000000",
        "00000000000000101100010001100000000000000000",
        "00001000100000010010000000110000000000000000",
        "00001100110000000000000000010000000000000000",
        "00001001000000000000000000000000000000000000",
        "00000000000000000000000000010000001000000000"

    );

begin

    CELL <= micro_rom(ADDR);

end architecture structure;
```

A.3.3 Status PLA

```
--*****************************************************************************
--* Status PLA processore Butterfly
--*****************************************************************************

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity status_PLA_butterfly is
    port (
        START : in std_logic;
        CC1   : in std_logic_vector(1 downto 0); -- condition code 1
        CC2   : in std_logic_vector(1 downto 0); -- condition code 2
        s     : out std_logic_vector(1 downto 0) -- selettore mux
    );
end entity status_PLA_butterfly;

architecture structure of status_PLA_butterfly is

    signal x1, x2, x3, x4, x5 : std_logic;
    signal s0, s1, s2         : std_logic;

begin

    x1 <= CC1(1);
    x2 <= CC1(0);
    x3 <= CC2(1);
    x4 <= CC2(0);
    x5 <= START;

    s0 <= (not x2) and (not x3) and (not x4) and ((not x1) or (x1 and x5));
    s1 <= ((not x3) and (not x4) and (((not x1) and x2) or (x1 and (not x2) and (not
    x5)))) or (x1 and x2 and x3 and (not x4) and x5);
    s2 <= x1 and x2 and x3 and (not x4) and (not x5);

```

```
    ADD_PROCESS : process (s0, s1, s2)
    begin
        if (s0 = '1') then
            s <= "00";
        elsif (s1 = '1') then
            s <= "01";
        elsif (s2 = '1') then
            s <= "10";
        end if;
    end process ADD_PROCESS;

end architecture structure;
```

A.3.4 Control Unit microprogrammata

```
--*****************************************************************************
--* CU processore Butterfly con tecnica della microprogrammazione
--*****************************************************************************

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity CU_butterfly is
    port (
        CK    : in std_logic;
        START : in std_logic;
        RST   : in std_logic;
        CTRL  : out std_logic_vector(28 downto 0)
    );
end entity CU_butterfly;

architecture structure of CU_butterfly is

    --*************************************************************************
    --* Definizione segnali interni
    --*************************************************************************

    signal CC_1       : std_logic_vector(1 downto 0);  -- condition code 1
    signal J_ADD_1    : std_logic_vector(4 downto 0);  -- jump address 1
    signal CC_2       : std_logic_vector(1 downto 0);  -- condition code 2
    signal J_ADD_2    : std_logic_vector(4 downto 0);  -- jump address 2
    signal NJ_ADD     : std_logic_vector(4 downto 0);  -- no jump address
    signal s_ADD      : std_logic_vector(1 downto 0);  -- selettore next address
    signal Q_uROM     : std_logic_vector(42 downto 0); -- uscita uROM
    signal Q_uIR      : std_logic_vector(42 downto 0); -- uscita uIR
    signal P_ADD      : std_logic_vector(4 downto 0);  -- present address
    signal N_ADD      : std_logic_vector(4 downto 0);  -- next address
    signal PS_int     : integer range 0 to 20;         -- indirizzo uROM
    signal NJ_ADD_int : integer range 1 to 20;         -- indirizzo uROM + 1

    --*************************************************************************
    --* Dichiarazione component
    --*************************************************************************

    -- micro rom butterfly
    component uROM_butterfly is
```

```
43          port (
44              ADDR : in integer range 0 to 19;
45              CELL : out std_logic_vector(42 downto 0)
46          );
47      end component;
48
49      -- registro che campiona sul fronte di salita del clock
50      component reg_rising is
51          generic (N : integer);
52          port (
53              clk : in std_logic;
54              d   : in std_logic_vector(N - 1 downto 0);
55              q   : out std_logic_vector(N - 1 downto 0)
56          );
57      end component;
58
59      -- registro che campiona sul fronte di discesa del clock
60      component reg_falling is
61          generic (N : integer);
62          port (
63              clk : in std_logic;
64              rst : in std_logic;
65              d   : in std_logic_vector(N - 1 downto 0);
66              q   : out std_logic_vector(N - 1 downto 0)
67          );
68      end component;
69
70      -- mux a 4 vie
71      component mux4to1 is
72          generic (N : integer := 16);
73          port (
74              s                       : in std_logic_vector(1 downto 0);     -- selettore a
                2 bit
75              IN_0, IN_1, IN_2, IN_3 : in std_logic_vector(N - 1 downto 0); -- input a N
                bit
76              uscita                  : out std_logic_vector(N - 1 downto 0) -- output a N
                bit
77          );
78      end component;
79
80      -- status PLA
81      component status_PLA_butterfly is
82          port (
83              START : in std_logic;
84              CC1   : in std_logic_vector(1 downto 0); -- condition code 1
85              CC2   : in std_logic_vector(1 downto 0); -- condition code 2
86              s     : out std_logic_vector(1 downto 0) -- selettore mux
87          );
88      end component;
89
90  begin
91
92      PS_int     <= to_integer(unsigned(P_ADD));
93      NJ_ADD_int <= PS_int + 1;
94      NJ_ADD     <= std_logic_vector(to_unsigned(NJ_ADD_int, 5));
95
96      CTRL    <= Q_uIR(42 downto 14);
97      J_ADD_1 <= Q_uIR(13 downto 9);
98      CC_1    <= Q_uIR(8 downto 7);
```

```
99      J_ADD_2 <= Q_uIR(6 downto 2);
100     CC_2    <= Q_uIR(1 downto 0);
101
102     --*****************************************************************************
103     --* Istanze component
104     --*****************************************************************************
105
106     uROM : uROM_butterfly
107     port map(
108         ADDR => PS_int,
109         CELL => Q_uROM
110     );
111
112     uIR : reg_rising
113     generic map(N => 43)
114     port map(
115         clk => CK,
116         d   => Q_uROM,
117         q   => Q_uIR
118     );
119
120     mux_uAR : mux4to1
121     generic map(N => 5)
122     port map(
123         s      => s_ADD,
124         IN_0   => NJ_ADD,  -- no jump
125         IN_1   => J_ADD_1, -- jump 1
126         IN_2   => J_ADD_2, -- jump 2
127         IN_3   => "00000",
128         uscita => N_ADD
129     );
130
131     uAR : reg_falling
132     generic map(N => 5)
133     port map(
134         clk => CK,
135         rst => RST,
136         d   => N_ADD,
137         q   => P_ADD
138     );
139
140     PLA : status_PLA_butterfly
141     port map(
142         START => START,
143         CC1   => CC_1,
144         CC2   => CC_2,
145         s     => s_ADD
146     );
147
148 end architecture structure;
```

A.3.5 Top level

```
1   --*****************************************************************************
2   --* Processore Butterfly
3   --*****************************************************************************
4
```

```
5  library ieee;
6  use ieee.std_logic_1164.all;
7  use ieee.numeric_std.all;
8  use ieee.fixed_pkg.all;
9
10 entity butterfly is
11     port (
12         CK    : in std_logic;
13         RST   : in std_logic;
14         START : in std_logic;
15         DONE  : out std_logic;
16         --* porte di ingresso *****************************************************
17         Ar : in sfixed(0 downto -23);
18         Ai : in sfixed(0 downto -23);
19         Br : in sfixed(0 downto -23);
20         Bi : in sfixed(0 downto -23);
21         Wr : in sfixed(0 downto -23);
22         Wi : in sfixed(0 downto -23);
23         --* porte di uscita ********************************************************
24         Ar_primo : out sfixed(0 downto -23);
25         Ai_primo : out sfixed(0 downto -23);
26         Br_primo : out sfixed(0 downto -23);
27         Bi_primo : out sfixed(0 downto -23)
28     );
29 end entity butterfly;
30
31 architecture structure of butterfly is
32
33     --*****************************************************************************
34     --* Definizione segnali interni
35     --*****************************************************************************
36
37     signal CTRL : std_logic_vector(28 downto 0);
38     ------------------------------------------------------------------------------
39     signal WR_R1 : std_logic;
40     signal WR_R2 : std_logic;
41     signal WR_R3 : std_logic;
42     signal WR_R4 : std_logic;
43     signal WR_R5 : std_logic;
44     signal WR_R6 : std_logic;
45     signal s_R3  : std_logic;
46     signal s_R5  : std_logic_vector(1 downto 0);
47     signal s_R6  : std_logic;
48     signal s_1M  : std_logic_vector(1 downto 0);
49     signal s_2M  : std_logic;
50     signal s_1AS : std_logic;
51     ------------------------------------------------------------------------------
52     signal WR_R7 : std_logic;
53     signal WR_R8 : std_logic;
54     signal WR_R9 : std_logic;
55     signal s_R7  : std_logic;
56     signal s_R8  : std_logic_vector(1 downto 0);
57     signal s_R9  : std_logic_vector(1 downto 0);
58     ------------------------------------------------------------------------------
59     signal M_nS : std_logic;
60     ------------------------------------------------------------------------------
61     signal s_D1A   : std_logic;                    -- selettore mux_D1A
62     signal s_D1S   : std_logic_vector(1 downto 0); -- selettore mux_D1S
63     signal s_DR    : std_logic;                    -- selettore mux_DR
```

```
64      signal s_IN_AS : std_logic;                    -- selettore mux_IN_AS
65      ------------------------------------------------------------------------------
66      signal D1_M_RF  : sfixed(0 downto -23); -- D1M da reg file Q1.23
67      signal D2_M_RF  : sfixed(0 downto -23); -- D2M da reg file Q1.23
68      signal D1_AS_RF : sfixed(3 downto -46); -- D1A/D1S da reg file Q4.46
69      signal D_R_RF   : sfixed(3 downto -46); -- DR da reg file Q4.46 (IN_0 mux_DR)
70      ------------------------------------------------------------------------------
71      signal D1_ADD : sfixed(2 downto -46); -- D1A Q3.46 (uscita mux_D1A)
72      signal D1_SUB : sfixed(3 downto -46); -- D1S Q4.46 (uscita mux_D1S)
73      signal D_RND  : sfixed(3 downto -46); -- DR Q4.46 (uscita mux_DR)
74      ------------------------------------------------------------------------------
75      signal Q_MPY    : sfixed(0 downto -46); -- QM Q1.46
76      signal Q_MPY_50 : sfixed(3 downto -46); -- QM Q4.46 (IN_0 mux_R9)
77      signal Q_SFT    : sfixed(1 downto -23); -- QSH Q2.23 (IN_0 mux_R7)
78      signal Q_SFT_49 : sfixed(2 downto -46); -- QSH Q3.46 (IN_0 mux_R8)
79      signal Q_ADD    : sfixed(2 downto -46); -- QA Q3.46 (IN_1 mux_R8)
80      signal Q_ADD_50 : sfixed(3 downto -46); -- QA Q4.46 (IN_1 mux_R9)
81      signal Q_SUB    : sfixed(3 downto -46); -- QSB Q4.46 (IN_2 mux_R9)
82      signal Q_RND    : sfixed(0 downto -23); -- QR Q1.23 (IN_R reg file)
83      signal Q_RND_25 : sfixed(1 downto -23); -- QR Q2.23 (IN_1 mux_R7)
84      signal Q_RND_49 : sfixed(2 downto -46); -- QR Q3.46 (IN_2 mux_R8)
85      signal Q_AS     : sfixed(3 downto -46); -- IN_AS Q4.46 (uscita mux_AS)
86      ------------------------------------------------------------------------------
87      signal D_R7 : sfixed(1 downto -23); -- ingresso REG_7 Q2.23 (uscita mux_R7)
88      signal D_R8 : sfixed(2 downto -46); -- ingresso REG_8 Q3.46 (uscita mux_R8)
89      signal D_R9 : sfixed(3 downto -46); -- ingresso REG_9 Q4.46 (uscita mux_R9)
90      ------------------------------------------------------------------------------
91      signal Q_R7    : sfixed(1 downto -23); -- uscita REG_7 Q2.23
92      signal Q_R7_50 : sfixed(3 downto -46); -- uscita REG_7 Q4.46 (IN_1 mux_D1S)
93      signal Q_R8    : sfixed(2 downto -46); -- uscita REG_8 Q3.46 (IN_0 mux_D1A)
94      signal Q_R8_50 : sfixed(3 downto -46); -- uscita REG_8 Q4.46 (IN_0 mux_D1S)
95      signal Q_R9    : sfixed(3 downto -46); -- uscita REG_9 Q4.46 (IN_1 mux_DR, D2S)
96      signal Q_R9_49 : sfixed(2 downto -46); -- uscita REG_9 Q3.46 (D2A sommatore)
97      ------------------------------------------------------------------------------
98      signal zero : sfixed(3 downto -46);
99
100     --*****************************************************************************
101     --* Dichiarazione component
102     --*****************************************************************************
103
104     -- registro
105     component register_sfixed is
106         generic (
107             M : integer := 0; -- MSB
108             N : integer := 24 -- lunghezza numero [bit]
109         );
110         port (
111             ck  : in std_logic;
112             en  : in std_logic;
113             d   : in sfixed(M downto (M - N + 1));
114             q   : out sfixed(M downto (M - N + 1))
115         );
116     end component;
117
118     -- mux a due vie
119     component mux2to1_sfixed is
120         generic (
121             M : integer; -- MSB
122             N : integer  -- lunghezza numero [bit]
```

```
        );
        port (
            s          : in std_logic;
            IN_0, IN_1 : in sfixed(M downto (M - N + 1));
            uscita     : out sfixed(M downto (M - N + 1))
        );
    end component;

    -- mux a quattro vie
    component mux4to1_sfixed is
        generic (
            M : integer; -- MSB
            N : integer  -- lunghezza numero [bit]
        );
        port (
            s                       : in std_logic_vector(1 downto 0);
            IN_0, IN_1, IN_2, IN_3 : in sfixed(M downto (M - N + 1));
            uscita                  : out sfixed(M downto (M - N + 1))
        );
    end component;

    -- sommatore
    component adder_pipe_sfixed is
        generic (
            M : integer := 2; -- MSB
            N : integer := 49 -- lunghezza numero [bit]
        );
        port (
            CK       : in std_logic;
            D1A, D2A : in sfixed(M downto (M - N + 1)); -- Q3.46
            QA       : out sfixed(M downto (M - N + 1)) -- Q3.46
        );
    end component;

    -- sottrattore
    component subtractor_pipe_sfixed is
        generic (
            M : integer := 3; -- MSB
            N : integer := 50 -- lunghezza numero [bit]
        );
        port (
            CK       : in std_logic;
            D1S, D2S : in sfixed(M downto (M - N + 1)); -- Q4.46
            QSB      : out sfixed(M downto (M - N + 1)) -- Q4.46
        );
    end component;

    -- moltiplicatore/shifter
    component mpy_shifter_pipe_sfixed is
        generic (N : integer := 24); -- lunghezza numero [bit]
        port (
            CK       : in std_logic;
            M_nS     : in std_logic;
            D1M, D2M : in sfixed(0 downto (-N + 1));      -- Q1.23
            QM       : out sfixed(0 downto (-2 * N + 2)); -- Q1.46
            QSH      : out sfixed(1 downto (-N + 1))      -- Q2.23
        );
    end component;

```

```
182     -- arrotondatore
183     component rounder_sfixed is
184         generic (
185             M : integer := 3; -- MSB
186             N : integer := 50 -- lunghezza numero [bit]
187         );
188         port (
189             DR : in sfixed(M downto (M - N + 1)); -- Q4.46
190             QR : out sfixed(0 downto -23)         -- Q1.23
191         );
192     end component;
193 
194     -- register file
195     component register_file is
196         port (
197             CK  : in std_logic;
198             --* segnali di controllo ************************************************
199             WR_R1 : in std_logic;
200             WR_R2 : in std_logic;
201             WR_R3 : in std_logic;
202             WR_R4 : in std_logic;
203             WR_R5 : in std_logic;
204             WR_R6 : in std_logic;
205             ---------------------------------------------------------------------------
206             s_R3 : in std_logic;
207             s_R5 : in std_logic_vector(1 downto 0);
208             s_R6 : in std_logic;
209             ---------------------------------------------------------------------------
210             s_1M  : in std_logic_vector(1 downto 0);
211             s_2M  : in std_logic;
212             s_1AS : in std_logic;
213             --* porte di ingresso ***************************************************
214             D_Wr : in sfixed(0 downto -23); -- Q1.23
215             D_Wi : in sfixed(0 downto -23); -- Q1.23
216             ---------------------------------------------------------------------------
217             D_Ar : in sfixed(0 downto -23); -- Q1.23
218             D_Ai : in sfixed(0 downto -23); -- Q1.23
219             D_Br : in sfixed(0 downto -23); -- Q1.23
220             D_Bi : in sfixed(0 downto -23); -- Q1.23
221             ---------------------------------------------------------------------------
222             IN_AS : in sfixed(3 downto -46); -- Q4.46
223             IN_R  : in sfixed(0 downto -23); -- Q1.23
224             --* porte di uscita ****************************************************
225             Q_Br : out sfixed(0 downto -23); -- Q1.23
226             Q_Bi : out sfixed(0 downto -23); -- Q1.23
227             ---------------------------------------------------------------------------
228             OUT_1M  : out sfixed(0 downto -23); -- Q1.23
229             OUT_2M  : out sfixed(0 downto -23); -- Q1.23
230             OUT_1AS : out sfixed(3 downto -46); -- Q4.46
231             OUT_R   : out sfixed(3 downto -46)  -- Q4.46
232         );
233     end component;
234 
235     -- control unit
236     component CU_butterfly is
237         port (
238             CK    : in std_logic;
239             START : in std_logic;
240             RST   : in std_logic;
```

```
241             CTRL  : out std_logic_vector(28 downto 0)
242         );
243     end component;
244 
245 begin
246 
247     zero <= (others => '0');
248 
249     -- estensioni e riduzioni
250     Q_MPY_50 <= Q_MPY(0) & Q_MPY(0) & Q_MPY(0) & Q_MPY;             -- da Q1.46 a Q4.46
251     Q_SFT_49 <= Q_SFT(1) & Q_SFT & zero(-24 downto -46);           -- da Q2.23 a Q3.46
252     Q_ADD_50 <= Q_ADD(2) & Q_ADD;                                  -- da Q3.46 a Q4.46
253     Q_RND_25 <= Q_RND(0) & Q_RND;                                  -- da Q1.23 a Q2.23
254     Q_RND_49 <= Q_RND(0) & Q_RND(0) & Q_RND & zero(-24 downto -46); -- da Q1.23 a Q3.46
255     Q_R7_50  <= Q_R7(1) & Q_R7(1) & Q_R7 & zero(-24 downto -46);   -- da Q2.23 a Q4.46
256     Q_R8_50  <= Q_R8(2) & Q_R8;                                    -- da Q3.46 a Q4.46
257     Q_R9_49  <= Q_R9(2 downto -46);                                -- da Q4.46 a Q3.46
258 
259     -- segnali di controllo
260     WR_R1   <= CTRL(28);
261     WR_R2   <= CTRL(27);
262     WR_R3   <= CTRL(26);
263     WR_R4   <= CTRL(25);
264     WR_R5   <= CTRL(24);
265     WR_R6   <= CTRL(23);
266     s_R3    <= CTRL(22);
267     s_R5    <= CTRL(21 downto 20);
268     s_R6    <= CTRL(19);
269     s_1M    <= CTRL(18 downto 17);
270     s_2M    <= CTRL(16);
271     s_1AS   <= CTRL(15);
272     WR_R7   <= CTRL(14);
273     WR_R8   <= CTRL(13);
274     WR_R9   <= CTRL(12);
275     s_R7    <= CTRL(11);
276     s_R8    <= CTRL(10 downto 9);
277     s_R9    <= CTRL(8 downto 7);
278     M_nS    <= CTRL(6);
279     s_D1A   <= CTRL(5);
280     s_D1S   <= CTRL(4 downto 3);
281     s_DR    <= CTRL(2);
282     s_IN_AS <= CTRL(1);
283     DONE    <= CTRL(0);
284 
285     Ar_primo <= Q_R7(0 downto -23); -- Q1.23
286     Ai_primo <= Q_R8(0 downto -23); -- Q1.23
287 
288     --*****************************************************************************
289     --* Istanze component
290     --*****************************************************************************
291 
292     REG_FILE : register_file
293     port map(
294         CK      => CK,
295         WR_R1   => WR_R1,
296         WR_R2   => WR_R2,
297         WR_R3   => WR_R3,
298         WR_R4   => WR_R4,
299         WR_R5   => WR_R5,
```

```
        WR_R6   => WR_R6,
        s_R3    => s_R3,
        s_R5    => s_R5,
        s_R6    => s_R6,
        s_1M    => s_1M,
        s_2M    => s_2M,
        s_1AS   => s_1AS,
        D_Wr    => Wr,        -- Q1.23
        D_Wi    => Wi,        -- Q1.23
        D_Ar    => Ar,        -- Q1.23
        D_Ai    => Ai,        -- Q1.23
        D_Br    => Br,        -- Q1.23
        D_Bi    => Bi,        -- Q1.23
        IN_AS   => Q_AS,      -- Q4.46
        IN_R    => Q_RND,     -- Q1.23
        Q_Br    => Br_primo, -- Q1.23
        Q_Bi    => Bi_primo, -- Q1.23
        OUT_1M  => D1_M_RF,  -- Q1.23
        OUT_2M  => D2_M_RF,  -- Q1.23
        OUT_1AS => D1_AS_RF, -- Q4.46
        OUT_R   => D_R_RF    -- Q4.46
    );

    MPY_SHIFTER : mpy_shifter_pipe_sfixed
    generic map(N => 24)
    port map(
        CK   => CK,
        M_nS => M_nS,
        D1M  => D1_M_RF, -- Q1.23
        D2M  => D2_M_RF, -- Q1.23
        QM   => Q_MPY,   -- Q1.46
        QSH  => Q_SFT    -- Q2.23
    );

    ADDER : adder_pipe_sfixed
    generic map(M => 2, N => 49)
    port map(
        CK  => CK,
        D1A => D1_ADD,  -- Q3.46
        D2A => Q_R9_49, -- Q3.46
        QA  => Q_ADD    -- Q3.46
    );

    SUBTRACTOR : subtractor_pipe_sfixed
    generic map(M => 3, N => 50)
    port map(
        CK  => CK,
        D1S => D1_SUB, -- Q4.46
        D2S => Q_R9,   -- Q4.46
        QSB => Q_SUB   -- Q4.46
    );

    ROUNDER_SCALER : rounder_sfixed
    generic map(M => 3, N => 50)
    port map(
        DR => D_RND, -- Q4.46
        QR => Q_RND  -- Q1.23
    );

```

```
359     mux_D1A : mux2to1_sfixed
360     generic map(M => 2, N => 49)
361     port map(
362         s      => s_D1A,
363         IN_0   => Q_R8,                    -- Q3.46
364         IN_1   => D1_AS_RF(2 downto -46), -- Q3.46
365         uscita => D1_ADD                   -- Q3.46
366     );
367
368     mux_D1S : mux4to1_sfixed
369     generic map(M => 3, N => 50)
370     port map(
371         s      => s_D1S,
372         IN_0   => Q_R8_50,  -- Q4.46
373         IN_1   => Q_R7_50,  -- Q4.46
374         IN_2   => D1_AS_RF, -- Q4.46
375         IN_3   => zero,
376         uscita => D1_SUB -- Q4.46
377     );
378
379     mux_DR : mux2to1_sfixed
380     generic map(M => 3, N => 50)
381     port map(
382         s      => s_DR,
383         IN_0   => D_R_RF, -- Q4.46
384         IN_1   => Q_R9,   -- Q4.46
385         uscita => D_RND   -- Q4.46
386     );
387
388     mux_R7 : mux2to1_sfixed
389     generic map(M => 1, N => 25)
390     port map(
391         s      => s_R7,
392         IN_0   => Q_SFT,    -- Q2.23
393         IN_1   => Q_RND_25, -- Q2.23
394         uscita => D_R7      -- Q2.23
395     );
396
397     mux_R8 : mux4to1_sfixed
398     generic map(M => 2, N => 49)
399     port map(
400         s      => s_R8,
401         IN_0   => Q_SFT_49, -- Q3.46
402         IN_1   => Q_ADD,    -- Q3.46
403         IN_2   => Q_RND_49, -- Q3.46
404         IN_3   => zero(2 downto -46),
405         uscita => D_R8 -- Q3.46
406     );
407
408     mux_R9 : mux4to1_sfixed
409     generic map(M => 3, N => 50)
410     port map(
411         s      => s_R9,
412         IN_0   => Q_MPY_50, -- Q4.46
413         IN_1   => Q_ADD_50, -- Q4.46
414         IN_2   => Q_SUB,    -- Q4.46
415         IN_3   => zero,
416         uscita => D_R9 -- Q4.46
417     );
```

```
418
419     REG_7 : register_sfixed
420     generic map(M => 1, N => 25)
421     port map(
422         ck  => CK,
423         en  => WR_R7,
424         d   => D_R7,
425         q   => Q_R7
426     );
427
428     REG_8 : register_sfixed
429     generic map(M => 2, N => 49)
430     port map(
431         ck  => CK,
432         en  => WR_R8,
433         d   => D_R8,
434         q   => Q_R8
435     );
436
437     REG_9 : register_sfixed
438     generic map(M => 3, N => 50)
439     port map(
440         ck  => CK,
441         en  => WR_R9,
442         d   => D_R9,
443         q   => Q_R9
444     );
445
446     mux_AS : mux2to1_sfixed
447     generic map(M => 3, N => 50)
448     port map(
449         s      => s_IN_AS,
450         IN_0   => Q_ADD_50, -- Q4.46
451         IN_1   => Q_SUB,    -- Q4.46
452         uscita => Q_AS      -- Q4.46
453     );
454
455     CONTROL_UNIT : CU_butterfly
456     port map(
457         CK    => CK,
458         RST   => RST,
459         START => START,
460         CTRL  => CTRL
461     );
462
463 end architecture structure;
```

A.4 Calcolatore FFT

```
1 --*****************************************************************************
2 --* Calcolatore FFT 16x16
3 --*****************************************************************************
4
5 library ieee;
6 use ieee.std_logic_1164.all;
7 use ieee.numeric_std.all;
8 use ieee.fixed_pkg.all;
```

```
9
10  entity FFT_16 is
11      port (
12          CK    : in std_logic;
13          RST   : in std_logic;
14          START : in std_logic;
15          DONE  : out std_logic;
16          --* porte di ingresso ********************************************************
17          Xr_0, Xi_0   : in sfixed(0 downto -23); -- X_0
18          Xr_1, Xi_1   : in sfixed(0 downto -23); -- X_1
19          Xr_2, Xi_2   : in sfixed(0 downto -23); -- X_2
20          Xr_3, Xi_3   : in sfixed(0 downto -23); -- X_3
21          Xr_4, Xi_4   : in sfixed(0 downto -23); -- X_4
22          Xr_5, Xi_5   : in sfixed(0 downto -23); -- X_5
23          Xr_6, Xi_6   : in sfixed(0 downto -23); -- X_6
24          Xr_7, Xi_7   : in sfixed(0 downto -23); -- X_7
25          Xr_8, Xi_8   : in sfixed(0 downto -23); -- X_8
26          Xr_9, Xi_9   : in sfixed(0 downto -23); -- X_9
27          Xr_10, Xi_10 : in sfixed(0 downto -23); -- X_10
28          Xr_11, Xi_11 : in sfixed(0 downto -23); -- X_11
29          Xr_12, Xi_12 : in sfixed(0 downto -23); -- X_12
30          Xr_13, Xi_13 : in sfixed(0 downto -23); -- X_13
31          Xr_14, Xi_14 : in sfixed(0 downto -23); -- X_14
32          Xr_15, Xi_15 : in sfixed(0 downto -23); -- X_15
33          --* porte di uscita ***********************************************************
34          Yr_0, Yi_0   : out sfixed(0 downto -23); -- Y_0
35          Yr_1, Yi_1   : out sfixed(0 downto -23); -- Y_1
36          Yr_2, Yi_2   : out sfixed(0 downto -23); -- Y_2
37          Yr_3, Yi_3   : out sfixed(0 downto -23); -- Y_3
38          Yr_4, Yi_4   : out sfixed(0 downto -23); -- Y_4
39          Yr_5, Yi_5   : out sfixed(0 downto -23); -- Y_5
40          Yr_6, Yi_6   : out sfixed(0 downto -23); -- Y_6
41          Yr_7, Yi_7   : out sfixed(0 downto -23); -- Y_7
42          Yr_8, Yi_8   : out sfixed(0 downto -23); -- Y_8
43          Yr_9, Yi_9   : out sfixed(0 downto -23); -- Y_9
44          Yr_10, Yi_10 : out sfixed(0 downto -23); -- Y_10
45          Yr_11, Yi_11 : out sfixed(0 downto -23); -- Y_11
46          Yr_12, Yi_12 : out sfixed(0 downto -23); -- Y_12
47          Yr_13, Yi_13 : out sfixed(0 downto -23); -- Y_13
48          Yr_14, Yi_14 : out sfixed(0 downto -23); -- Y_14
49          Yr_15, Yi_15 : out sfixed(0 downto -23)  -- Y_15
50      );
51  end entity FFT_16;
52
53  architecture structure of FFT_16 is
54
55      --*****************************************************************************
56      --* Definizione segnali interni
57      --*****************************************************************************
58
59      signal Wr_0, Wi_0 : sfixed(0 downto -23); -- W_0
60      signal Wr_1, Wi_1 : sfixed(0 downto -23); -- W_1
61      signal Wr_2, Wi_2 : sfixed(0 downto -23); -- W_2
62      signal Wr_3, Wi_3 : sfixed(0 downto -23); -- W_3
63      signal Wr_4, Wi_4 : sfixed(0 downto -23); -- W_4
64      signal Wr_5, Wi_5 : sfixed(0 downto -23); -- W_5
65      signal Wr_6, Wi_6 : sfixed(0 downto -23); -- W_6
66      signal Wr_7, Wi_7 : sfixed(0 downto -23); -- W_7
67      ------------------------------------------------------------------------------
```

```
68      signal DL1_SL2 : std_logic; -- DONE livello 1, START livello 2
69      signal DL2_SL3 : std_logic; -- DONE livello 2, START livello 3
70      signal DL3_SL4 : std_logic; -- DONE livello 3, START livello 4
71      ------------------------------------------------------------------------------
72      signal ArO00_ArI08, AiO00_AiI08 : sfixed(0 downto -23); -- A' bfy 00, A bfy 08
73      signal BrO00_ArI12, BiO00_AiI12 : sfixed(0 downto -23); -- B' bfy 00, A bfy 12
74      signal ArO01_ArI09, AiO01_AiI09 : sfixed(0 downto -23); -- A' bfy 01, A bfy 09
75      signal BrO01_ArI13, BiO01_AiI13 : sfixed(0 downto -23); -- B' bfy 01, A bfy 13
76      signal ArO02_ArI10, AiO02_AiI10 : sfixed(0 downto -23); -- A' bfy 02, A bfy 10
77      signal BrO02_ArI14, BiO02_AiI14 : sfixed(0 downto -23); -- B' bfy 02, A bfy 14
78      signal ArO03_ArI11, AiO03_AiI11 : sfixed(0 downto -23); -- A' bfy 03, A bfy 11
79      signal BrO03_ArI15, BiO03_AiI15 : sfixed(0 downto -23); -- B' bfy 03, A bfy 15
80      signal ArO04_BrI08, AiO04_BiI08 : sfixed(0 downto -23); -- A' bfy 04, B bfy 08
81      signal BrO04_BrI12, BiO04_BiI12 : sfixed(0 downto -23); -- B' bfy 04, B bfy 12
82      signal ArO05_BrI09, AiO05_BiI09 : sfixed(0 downto -23); -- A' bfy 05, B bfy 09
83      signal BrO05_BrI13, BiO05_BiI13 : sfixed(0 downto -23); -- B' bfy 05, B bfy 13
84      signal ArO06_BrI10, AiO06_BiI10 : sfixed(0 downto -23); -- A' bfy 06, B bfy 10
85      signal BrO06_BrI14, BiO06_BiI14 : sfixed(0 downto -23); -- B' bfy 06, B bfy 14
86      signal ArO07_BrI11, AiO07_BiI11 : sfixed(0 downto -23); -- A' bfy 07, B bfy 11
87      signal BrO07_BrI15, BiO07_BiI15 : sfixed(0 downto -23); -- B' bfy 07, B bfy 15
88      ------------------------------------------------------------------------------
89      signal ArO08_ArI16, AiO08_AiI16 : sfixed(0 downto -23); -- A' bfy 08, A bfy 16
90      signal BrO08_ArI18, BiO08_AiI18 : sfixed(0 downto -23); -- B' bfy 08, A bfy 18
91      signal ArO09_ArI17, AiO09_AiI17 : sfixed(0 downto -23); -- A' bfy 09, A bfy 17
92      signal BrO09_ArI19, BiO09_AiI19 : sfixed(0 downto -23); -- B' bfy 09, A bfy 19
93      signal ArO10_BrI16, AiO10_BiI16 : sfixed(0 downto -23); -- A' bfy 10, B bfy 16
94      signal BrO10_BrI18, BiO10_BiI18 : sfixed(0 downto -23); -- B' bfy 10, B bfy 18
95      signal ArO11_BrI17, AiO11_BiI17 : sfixed(0 downto -23); -- A' bfy 11, B bfy 17
96      signal BrO11_BrI19, BiO11_BiI19 : sfixed(0 downto -23); -- B' bfy 11, B bfy 19
97      signal ArO12_ArI20, AiO12_AiI20 : sfixed(0 downto -23); -- A' bfy 12, A bfy 20
98      signal BrO12_ArI22, BiO12_AiI22 : sfixed(0 downto -23); -- B' bfy 12, A bfy 22
99      signal ArO13_ArI21, AiO13_AiI21 : sfixed(0 downto -23); -- A' bfy 13, A bfy 21
100     signal BrO13_ArI23, BiO13_AiI23 : sfixed(0 downto -23); -- B' bfy 13, A bfy 23
101     signal ArO14_BrI20, AiO14_BiI20 : sfixed(0 downto -23); -- A' bfy 14, B bfy 20
102     signal BrO14_BrI22, BiO14_BiI22 : sfixed(0 downto -23); -- B' bfy 14, B bfy 22
103     signal ArO15_BrI21, AiO15_BiI21 : sfixed(0 downto -23); -- A' bfy 15, B bfy 21
104     signal BrO15_BrI23, BiO15_BiI23 : sfixed(0 downto -23); -- B' bfy 15, B bfy 23
105     ------------------------------------------------------------------------------
106     signal ArO16_AiI24, AiO16_AiI24 : sfixed(0 downto -23); -- A' bfy 1 A bfy 24
107     signal BrO16_AiI25, BiO16_AiI25 : sfixed(0 downto -23); -- B' bfy 1 A bfy 25
108     signal ArO17_BiI24, AiO17_BiI24 : sfixed(0 downto -23); -- A' bfy 1 B bfy 24
109     signal BrO17_BiI25, BiO17_BiI25 : sfixed(0 downto -23); -- B' bfy 1 B bfy 25
110     signal ArO18_AiI26, AiO18_AiI26 : sfixed(0 downto -23); -- A' bfy 1 A bfy 26
111     signal BrO18_AiI27, BiO18_AiI27 : sfixed(0 downto -23); -- B' bfy 1 A bfy 27
112     signal ArO19_BiI26, AiO19_BiI26 : sfixed(0 downto -23); -- A' bfy 1 B bfy 26
113     signal BrO19_BiI27, BiO19_BiI27 : sfixed(0 downto -23); -- B' bfy 1 B bfy 27
114     signal ArO20_AiI28, AiO20_AiI28 : sfixed(0 downto -23); -- A' bfy 2 A bfy 28
115     signal BrO20_AiI29, BiO20_AiI29 : sfixed(0 downto -23); -- B' bfy 2 A bfy 29
116     signal ArO21_BiI28, AiO21_BiI28 : sfixed(0 downto -23); -- A' bfy 2 B bfy 28
117     signal BrO21_BiI29, BiO21_BiI29 : sfixed(0 downto -23); -- B' bfy 2 B bfy 29
118     signal ArO22_AiI30, AiO22_AiI30 : sfixed(0 downto -23); -- A' bfy 2 A bfy 30
119     signal BrO22_AiI31, BiO22_AiI31 : sfixed(0 downto -23); -- B' bfy 2 A bfy 31
120     signal ArO23_BiI30, AiO23_BiI30 : sfixed(0 downto -23); -- A' bfy 2 B bfy 30
121     signal BrO23_BiI31, BiO23_BiI31 : sfixed(0 downto -23); -- B' bfy 2 B bfy 31
122
123     --*****************************************************************************
124     --* Dichiarazione component
125     --*****************************************************************************
126
```

```
127     component butterfly is
128         port (
129             CK    : in std_logic;
130             RST   : in std_logic;
131             START : in std_logic;
132             DONE  : out std_logic;
133             --* porte di ingresso ******************************************************
134             Ar : in sfixed(0 downto -23);
135             Ai : in sfixed(0 downto -23);
136             Br : in sfixed(0 downto -23);
137             Bi : in sfixed(0 downto -23);
138             Wr : in sfixed(0 downto -23);
139             Wi : in sfixed(0 downto -23);
140             --* porte di uscita *********************************************************
141             Ar_primo : out sfixed(0 downto -23);
142             Ai_primo : out sfixed(0 downto -23);
143             Br_primo : out sfixed(0 downto -23);
144             Bi_primo : out sfixed(0 downto -23)
145         );
146     end component;
147 
148 begin
149 
150     Wr_0 <= "01111111111111111111111";
151     Wr_1 <= "01110110010000110101111";
152     Wr_2 <= "01011010100000100111001";
153     Wr_3 <= "00110000111110111000101";
154     Wr_4 <= "00000000000000000000000";
155     Wr_5 <= "11001111000001000011010";
156     Wr_6 <= "10100101011111011000110";
157     Wr_7 <= "10001001101111100101000";
158 
159     Wi_0 <= "00000000000000000000000";
160     Wi_1 <= "11001111000001000011010";
161     Wi_2 <= "10100101011111011000110";
162     Wi_3 <= "10001001101111100101000";
163     Wi_4 <= "10000000000000000000000";
164     Wi_5 <= "10001001101111100101000";
165     Wi_6 <= "10100101011111011000110";
166     Wi_7 <= "11001111000001000011010";
167 
168     --*************************************************************************************
169     --* Istanze component
170     --*************************************************************************************
171 
172     --* LIVELLO 1 ***************************************************************************
173     BUTTERFLY_00 : butterfly
174     port map(
175         CK => CK, RST => RST, START => START, DONE => DL1_SL2,
176         -- porte di ingresso
177         Ar => Xr_0, Ai => Xi_0, Br => Xr_8, Bi => Xi_8,
178         Wr => Wr_0, Wi => Wi_0,
179         -- porte di uscita
180         Ar_primo => ArO00_ArI08, Ai_primo => AiO00_AiI08,
181         Br_primo => BrO00_ArI12, Bi_primo => BiO00_AiI12
182     );
183 
184     BUTTERFLY_01 : butterfly
185     port map(
```

```
186         CK => CK, RST => RST, START => START, DONE => DL1_SL2,
187         -- porte di ingresso
188         Ar => Xr_1, Ai => Xi_1, Br => Xr_9, Bi => Xi_9,
189         Wr => Wr_0, Wi => Wi_0,
190         -- porte di uscita
191         Ar_primo => Ar001_ArI09, Ai_primo => Ai001_AiI09,
192         Br_primo => Br001_ArI13, Bi_primo => Bi001_AiI13
193     );
194
195     BUTTERFLY_02 : butterfly
196     port map(
197         CK => CK, RST => RST, START => START, DONE => DL1_SL2,
198         -- porte di ingresso
199         Ar => Xr_2, Ai => Xi_2, Br => Xr_10, Bi => Xi_10,
200         Wr => Wr_0, Wi => Wi_0,
201         -- porte di uscita
202         Ar_primo => Ar002_ArI10, Ai_primo => Ai002_AiI10,
203         Br_primo => Br002_ArI14, Bi_primo => Bi002_AiI14
204     );
205
206     BUTTERFLY_03 : butterfly
207     port map(
208         CK => CK, RST => RST, START => START, DONE => DL1_SL2,
209         -- porte di ingresso
210         Ar => Xr_3, Ai => Xi_3, Br => Xr_11, Bi => Xi_11,
211         Wr => Wr_0, Wi => Wi_0,
212         -- porte di uscita
213         Ar_primo => Ar003_ArI11, Ai_primo => Ai003_AiI11,
214         Br_primo => Br003_ArI15, Bi_primo => Bi003_AiI15
215     );
216
217     BUTTERFLY_04 : butterfly
218     port map(
219         CK => CK, RST => RST, START => START, DONE => DL1_SL2,
220         -- porte di ingresso
221         Ar => Xr_4, Ai => Xi_4, Br => Xr_12, Bi => Xi_12,
222         Wr => Wr_0, Wi => Wi_0,
223         -- porte di uscita
224         Ar_primo => Ar004_BrI08, Ai_primo => Ai004_BiI08,
225         Br_primo => Br004_BrI12, Bi_primo => Bi004_BiI12
226     );
227
228     BUTTERFLY_05 : butterfly
229     port map(
230         CK => CK, RST => RST, START => START, DONE => DL1_SL2,
231         -- porte di ingresso
232         Ar => Xr_5, Ai => Xi_5, Br => Xr_13, Bi => Xi_13,
233         Wr => Wr_0, Wi => Wi_0,
234         -- porte di uscita
235         Ar_primo => Ar005_BrI09, Ai_primo => Ai005_BiI09,
236         Br_primo => Br005_BrI13, Bi_primo => Bi005_BiI13
237     );
238
239     BUTTERFLY_06 : butterfly
240     port map(
241         CK => CK, RST => RST, START => START, DONE => DL1_SL2,
242         -- porte di ingresso
243         Ar => Xr_6, Ai => Xi_6, Br => Xr_14, Bi => Xi_14,
244         Wr => Wr_0, Wi => Wi_0,
```

```
245         -- porte di uscita
246         Ar_primo => ArO06_BrI10, Ai_primo => AiO06_BiI10,
247         Br_primo => BrO06_BrI14, Bi_primo => BiO06_BiI14
248     );
249 
250     BUTTERFLY_07 : butterfly
251     port map(
252         CK => CK, RST => RST, START => START, DONE => DL1_SL2,
253         -- porte di ingresso
254         Ar => Xr_7, Ai => Xi_7, Br => Xr_15, Bi => Xi_15,
255         Wr => Wr_0, Wi => Wi_0,
256         -- porte di uscita
257         Ar_primo => ArO07_BrI11, Ai_primo => AiO07_BiI11,
258         Br_primo => BrO07_BrI15, Bi_primo => BiO07_BiI15
259     );
260 
261     --* LIVELLO 2 ***************************************************************************
262     BUTTERFLY_08 : butterfly
263     port map(
264         CK => CK, RST => RST, START => DL1_SL2, DONE => DL2_SL3,
265         -- porte di ingresso
266         Ar => ArO00_ArI08, Ai => AiO00_AiI08, Br => ArO04_BrI08, Bi => AiO04_BiI08,
267         Wr => Wr_0, Wi => Wi_0,
268         -- porte di uscita
269         Ar_primo => ArO08_ArI16, Ai_primo => AiO08_AiI16,
270         Br_primo => BrO08_ArI18, Bi_primo => BiO08_AiI18
271     );
272 
273     BUTTERFLY_09 : butterfly
274     port map(
275         CK => CK, RST => RST, START => DL1_SL2, DONE => DL2_SL3,
276         -- porte di ingresso
277         Ar => ArO01_ArI09, Ai => AiO01_AiI09, Br => ArO05_BrI09, Bi => AiO05_BiI09,
278         Wr => Wr_0, Wi => Wi_0,
279         -- porte di uscita
280         Ar_primo => ArO09_ArI17, Ai_primo => AiO09_AiI17,
281         Br_primo => BrO09_ArI19, Bi_primo => BiO09_AiI19
282     );
283 
284     BUTTERFLY_10 : butterfly
285     port map(
286         CK => CK, RST => RST, START => DL1_SL2, DONE => DL2_SL3,
287         -- porte di ingresso
288         Ar => ArO02_ArI10, Ai => AiO02_AiI10, Br => ArO06_BrI10, Bi => AiO06_BiI10,
289         Wr => Wr_0, Wi => Wi_0,
290         -- porte di uscita
291         Ar_primo => ArO10_BrI16, Ai_primo => AiO10_BiI16,
292         Br_primo => BrO10_BrI18, Bi_primo => BiO10_BiI18
293     );
294 
295     BUTTERFLY_11 : butterfly
296     port map(
297         CK => CK, RST => RST, START => DL1_SL2, DONE => DL2_SL3,
298         -- porte di ingresso
299         Ar => ArO03_ArI11, Ai => AiO03_AiI11, Br => ArO07_BrI11, Bi => AiO07_BiI11,
300         Wr => Wr_0, Wi => Wi_0,
301         -- porte di uscita
302         Ar_primo => ArO11_BrI17, Ai_primo => AiO11_BiI17,
303         Br_primo => BrO11_BrI19, Bi_primo => BiO11_BiI19
```

```
304     );
305
306     BUTTERFLY_12 : butterfly
307     port map(
308         CK => CK, RST => RST, START => DL1_SL2, DONE => DL2_SL3,
309         -- porte di ingresso
310         Ar => Br000_ArI12, Ai => Bi000_AiI12, Br => Br004_BrI12, Bi => Bi004_BiI12,
311         Wr => Wr_4, Wi => Wi_4,
312         -- porte di uscita
313         Ar_primo => Ar012_ArI20, Ai_primo => Ai012_AiI20,
314         Br_primo => Br012_ArI22, Bi_primo => Bi012_AiI22
315     );
316
317     BUTTERFLY_13 : butterfly
318     port map(
319         CK => CK, RST => RST, START => DL1_SL2, DONE => DL2_SL3,
320         -- porte di ingresso
321         Ar => Br001_ArI13, Ai => Bi001_AiI13, Br => Br005_BrI13, Bi => Bi005_BiI13,
322         Wr => Wr_4, Wi => Wi_4,
323         -- porte di uscita
324         Ar_primo => Ar013_ArI21, Ai_primo => Ai013_AiI21,
325         Br_primo => Br013_ArI23, Bi_primo => Bi013_AiI23
326     );
327
328     BUTTERFLY_14 : butterfly
329     port map(
330         CK => CK, RST => RST, START => DL1_SL2, DONE => DL2_SL3,
331         -- porte di ingresso
332         Ar => Br002_ArI14, Ai => Bi002_AiI14, Br => Br006_BrI14, Bi => Bi006_BiI14,
333         Wr => Wr_4, Wi => Wi_4,
334         -- porte di uscita
335         Ar_primo => Ar014_BrI20, Ai_primo => Ai014_BiI20,
336         Br_primo => Br014_BrI22, Bi_primo => Bi014_BiI22
337     );
338
339     BUTTERFLY_15 : butterfly
340     port map(
341         CK => CK, RST => RST, START => DL1_SL2, DONE => DL2_SL3,
342         -- porte di ingresso
343         Ar => Br003_ArI15, Ai => Bi003_AiI15, Br => Br007_BrI15, Bi => Bi007_BiI15,
344         Wr => Wr_4, Wi => Wi_4,
345         -- porte di uscita
346         Ar_primo => Ar015_BrI21, Ai_primo => Ai015_BiI21,
347         Br_primo => Br015_BrI23, Bi_primo => Bi015_BiI23
348     );
349
350     --* LIVELLO 3 ************************************************************************
351     BUTTERFLY_16 : butterfly
352     port map(
353         CK => CK, RST => RST, START => DL2_SL3, DONE => DL3_SL4,
354         -- porte di ingresso
355         Ar => Ar008_ArI16, Ai => Ai008_AiI16, Br => Ar010_BrI16, Bi => Ai010_BiI16,
356         Wr => Wr_0, Wi => Wi_0,
357         -- porte di uscita
358         Ar_primo => Ar016_AiI24, Ai_primo => Ai016_AiI24,
359         Br_primo => Br016_AiI25, Bi_primo => Bi016_AiI25
360     );
361
362     BUTTERFLY_17 : butterfly
```

```
363     port map(
364         CK => CK, RST => RST, START => DL2_SL3, DONE => DL3_SL4,
365         -- porte di ingresso
366         Ar => ArO09_ArI17, Ai => AiO09_AiI17, Br => ArO11_BrI17, Bi => AiO11_BiI17,
367         Wr => Wr_0, Wi => Wi_0,
368         -- porte di uscita
369         Ar_primo => ArO17_BiI24, Ai_primo => AiO17_BiI24,
370         Br_primo => BrO17_BiI25, Bi_primo => BiO17_BiI25
371     );
372 
373     BUTTERFLY_18 : butterfly
374     port map(
375         CK => CK, RST => RST, START => DL2_SL3, DONE => DL3_SL4,
376         -- porte di ingresso
377         Ar => BrO08_ArI18, Ai => BiO08_AiI18, Br => BrO10_BrI18, Bi => BiO10_BiI18,
378         Wr => Wr_4, Wi => Wi_4,
379         -- porte di uscita
380         Ar_primo => ArO18_AiI26, Ai_primo => AiO18_AiI26,
381         Br_primo => BrO18_AiI27, Bi_primo => BiO18_AiI27
382     );
383 
384     BUTTERFLY_19 : butterfly
385     port map(
386         CK => CK, RST => RST, START => DL2_SL3, DONE => DL3_SL4,
387         -- porte di ingresso
388         Ar => BrO09_ArI19, Ai => BiO09_AiI19, Br => BrO11_BrI19, Bi => BiO11_BiI19,
389         Wr => Wr_4, Wi => Wi_4,
390         -- porte di uscita
391         Ar_primo => ArO19_BiI26, Ai_primo => AiO19_BiI26,
392         Br_primo => BrO19_BiI27, Bi_primo => BiO19_BiI27
393     );
394 
395     BUTTERFLY_20 : butterfly
396     port map(
397         CK => CK, RST => RST, START => DL2_SL3, DONE => DL3_SL4,
398         -- porte di ingresso
399         Ar => ArO12_ArI20, Ai => AiO12_AiI20, Br => ArO14_BrI20, Bi => AiO14_BiI20,
400         Wr => Wr_2, Wi => Wi_2,
401         -- porte di uscita
402         Ar_primo => ArO20_AiI28, Ai_primo => AiO20_AiI28,
403         Br_primo => BrO20_AiI29, Bi_primo => BiO20_AiI29
404     );
405 
406     BUTTERFLY_21 : butterfly
407     port map(
408         CK => CK, RST => RST, START => DL2_SL3, DONE => DL3_SL4,
409         -- porte di ingresso
410         Ar => ArO13_ArI21, Ai => AiO13_AiI21, Br => ArO15_BrI21, Bi => AiO15_BiI21,
411         Wr => Wr_2, Wi => Wi_2,
412         -- porte di uscita
413         Ar_primo => ArO21_BiI28, Ai_primo => AiO21_BiI28,
414         Br_primo => BrO21_BiI29, Bi_primo => BiO21_BiI29
415     );
416 
417     BUTTERFLY_22 : butterfly
418     port map(
419         CK => CK, RST => RST, START => DL2_SL3, DONE => DL3_SL4,
420         -- porte di ingresso
421         Ar => BrO12_ArI22, Ai => BiO12_AiI22, Br => BrO14_BrI22, Bi => BiO14_BiI22,
```

```
422             Wr => Wr_6, Wi => Wi_6,
423             -- porte di uscita
424             Ar_primo => Ar022_AiI30, Ai_primo => Ai022_AiI30,
425             Br_primo => Br022_AiI31, Bi_primo => Bi022_AiI31
426         );
427
428         BUTTERFLY_23 : butterfly
429         port map(
430             CK => CK, RST => RST, START => DL2_SL3, DONE => DL3_SL4,
431             -- porte di ingresso
432             Ar => Br013_ArI23, Ai => Bi013_AiI23, Br => Br015_BrI23, Bi => Bi015_BiI23,
433             Wr => Wr_6, Wi => Wi_6,
434             -- porte di uscita
435             Ar_primo => Ar023_BiI30, Ai_primo => Ai023_BiI30,
436             Br_primo => Br023_BiI31, Bi_primo => Bi023_BiI31
437         );
438
439         --* LIVELLO 4 ************************************************************************
440         BUTTERFLY_24 : butterfly
441         port map(
442             CK => CK, RST => RST, START => DL3_SL4, DONE => DONE,
443             -- porte di ingresso
444             Ar => Ar016_AiI24, Ai => Ai016_AiI24, Br => Ar017_BiI24, Bi => Ai017_BiI24,
445             Wr => Wr_0, Wi => Wi_0,
446             -- porte di uscita
447             Ar_primo => Yr_0, Ai_primo => Yi_0,
448             Br_primo => Yr_8, Bi_primo => Yi_8
449         );
450
451         BUTTERFLY_25 : butterfly
452         port map(
453             CK => CK, RST => RST, START => DL3_SL4, DONE => DONE,
454             -- porte di ingresso
455             Ar => Br016_AiI25, Ai => Bi016_AiI25, Br => Br017_BiI25, Bi => Bi017_BiI25,
456             Wr => Wr_4, Wi => Wi_4,
457             -- porte di uscita
458             Ar_primo => Yr_4, Ai_primo => Yi_4,
459             Br_primo => Yr_12, Bi_primo => Yi_12
460         );
461
462         BUTTERFLY_26 : butterfly
463         port map(
464             CK => CK, RST => RST, START => DL3_SL4, DONE => DONE,
465             -- porte di ingresso
466             Ar => Ar018_AiI26, Ai => Ai018_AiI26, Br => Ar019_BiI26, Bi => Ai019_BiI26,
467             Wr => Wr_2, Wi => Wi_2,
468             -- porte di uscita
469             Ar_primo => Yr_2, Ai_primo => Yi_2,
470             Br_primo => Yr_10, Bi_primo => Yi_10
471         );
472
473         BUTTERFLY_27 : butterfly
474         port map(
475             CK => CK, RST => RST, START => DL3_SL4, DONE => DONE,
476             -- porte di ingresso
477             Ar => Br018_AiI27, Ai => Bi018_AiI27, Br => Br019_BiI27, Bi => Bi019_BiI27,
478             Wr => Wr_6, Wi => Wi_6,
479             -- porte di uscita
480             Ar_primo => Yr_6, Ai_primo => Yi_6,
```

```
481         Br_primo => Yr_14, Bi_primo => Yi_14
482     );
483
484     BUTTERFLY_28 : butterfly
485     port map(
486         CK => CK, RST => RST, START => DL3_SL4, DONE => DONE,
487         -- porte di ingresso
488         Ar => ArO20_AiI28, Ai => AiO20_AiI28, Br => ArO21_BiI28, Bi => AiO21_BiI28,
489         Wr => Wr_1, Wi => Wi_1,
490         -- porte di uscita
491         Ar_primo => Yr_1, Ai_primo => Yi_1,
492         Br_primo => Yr_9, Bi_primo => Yi_9
493     );
494
495     BUTTERFLY_29 : butterfly
496     port map(
497         CK => CK, RST => RST, START => DL3_SL4, DONE => DONE,
498         -- porte di ingresso
499         Ar => BrO20_AiI29, Ai => BiO20_AiI29, Br => BrO21_BiI29, Bi => BiO21_BiI29,
500         Wr => Wr_5, Wi => Wi_5,
501         -- porte di uscita
502         Ar_primo => Yr_5, Ai_primo => Yi_5,
503         Br_primo => Yr_13, Bi_primo => Yi_13
504     );
505
506     BUTTERFLY_30 : butterfly
507     port map(
508         CK => CK, RST => RST, START => DL3_SL4, DONE => DONE,
509         -- porte di ingresso
510         Ar => ArO22_AiI30, Ai => AiO22_AiI30, Br => ArO23_BiI30, Bi => AiO23_BiI30,
511         Wr => Wr_3, Wi => Wi_3,
512         -- porte di uscita
513         Ar_primo => Yr_3, Ai_primo => Yi_3,
514         Br_primo => Yr_11, Bi_primo => Yi_11
515     );
516
517     BUTTERFLY_31 : butterfly
518     port map(
519         CK => CK, RST => RST, START => DL3_SL4, DONE => DONE,
520         -- porte di ingresso
521         Ar => BrO22_AiI31, Ai => BiO22_AiI31, Br => BrO23_BiI31, Bi => BiO23_BiI31,
522         Wr => Wr_7, Wi => Wi_7,
523         -- porte di uscita
524         Ar_primo => Yr_7, Ai_primo => Yi_7,
525         Br_primo => Yr_15, Bi_primo => Yi_15
526     );
527
528 end architecture structure;
```

B Test processore Butterfly

B.1 Testbench a scopo di *debug*

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.fixed_pkg.all;
use std.textio.all;
use ieee.std_logic_textio.all;

entity tb_butterfly_debug is
end tb_butterfly_debug;

architecture behavioral of tb_butterfly_debug is

    -- definizione segnali interni
    signal clock  : std_logic            := '0';
    signal reset  : std_logic            := '0';
    signal start  : std_logic            := '0';
    signal done   : std_logic            := '0';
    signal Ar_in  : sfixed(0 downto -23) := (others => '0');
    signal Ai_in  : sfixed(0 downto -23) := (others => '0');
    signal Br_in  : sfixed(0 downto -23) := (others => '0');
    signal Bi_in  : sfixed(0 downto -23) := (others => '0');
    signal Wr_in  : sfixed(0 downto -23) := (others => '0');
    signal Wi_in  : sfixed(0 downto -23) := (others => '0');
    signal Ar_out : sfixed(0 downto -23);
    signal Ai_out : sfixed(0 downto -23);
    signal Br_out : sfixed(0 downto -23);
    signal Bi_out : sfixed(0 downto -23);

    -- dichiarazione UUT
    component butterfly is
        port (
            CK    : in std_logic;
            RST   : in std_logic;
            START : in std_logic;
            DONE  : out std_logic;
            --* porte di ingresso ******************************************************
            Ar : in sfixed(0 downto -23);
            Ai : in sfixed(0 downto -23);
            Br : in sfixed(0 downto -23);
            Bi : in sfixed(0 downto -23);
            Wr : in sfixed(0 downto -23);
            Wi : in sfixed(0 downto -23);
            --* porte di uscita ********************************************************
            Ar_primo : out sfixed(0 downto -23);
            Ai_primo : out sfixed(0 downto -23);
            Br_primo : out sfixed(0 downto -23);
            Bi_primo : out sfixed(0 downto -23)
        );
    end component;

begin

    -- istanza UUT
    uP_BUTTERFLY : butterfly
```

```
55      port map(
56          CK       => clock,
57          RST      => reset,
58          START    => start,
59          DONE     => done,
60          Ar       => Ar_in,
61          Ai       => Ai_in,
62          Br       => Br_in,
63          Bi       => Bi_in,
64          Wr       => Wr_in,
65          Wi       => Wi_in,
66          Ar_primo => Ar_out,
67          Ai_primo => Ai_out,
68          Br_primo => Br_out,
69          Bi_primo => Bi_out
70      );
71
72      -- process per la generazione del clock
73      clk_process : process
74      begin
75          wait for 50 ns;
76          clock <= not clock;
77      end process clk_process;
78
79      trial : process
80      begin
81
82          reset <= '1';
83          wait for 2 ns;
84          reset <= '0';
85
86          -- Faccio partire il processore
87
88          -- single
89          wait for 100 ns;
90          start <= '1';
91          Wr_in <= "01111111111111111111111";
92          Wi_in <= "00000000000000000000000";
93          Ar_in <= to_sfixed(0.5, 0, -23);
94          Ai_in <= to_sfixed(0.5, 0, -23);
95          Br_in <= to_sfixed(0.5, 0, -23);
96          Bi_in <= to_sfixed(0.5, 0, -23);
97          wait for 100 ns;
98          start <= '0';
99          wait for 1500 ns;
100
101         -- single
102         start <= '1';
103         Wr_in <= "01111111111111111111111";
104         Wi_in <= "00000000000000000000000";
105         Ar_in <= to_sfixed(0.1, 0, -23);
106         Ai_in <= to_sfixed(0.1, 0, -23);
107         Br_in <= to_sfixed(0.2, 0, -23);
108         Bi_in <= to_sfixed(0.2, 0, -23);
109         wait for 100 ns;
110         start <= '0';
111         wait for 1500 ns;
112
113         -- continuous
```

```
114             start <= '1';
115             Wr_in <= "01111111111111111111111";
116             Wi_in <= "00111111111111111111111";
117             Ar_in <= to_sfixed(0.1, 0, -23);
118             Ai_in <= to_sfixed(0.2, 0, -23);
119             Br_in <= to_sfixed(0.3, 0, -23);
120             Bi_in <= to_sfixed(0.4, 0, -23);
121             wait for 100 ns;
122             start <= '0';
123             wait for 400 ns;
124             start <= '1';
125             Wr_in <= "01111111111111111111111";
126             Wi_in <= "00111111111111111111111";
127             Ar_in <= to_sfixed(0.1, 0, -23);
128             Ai_in <= to_sfixed(0.1, 0, -23);
129             Br_in <= to_sfixed(0.2, 0, -23);
130             Bi_in <= to_sfixed(0.2, 0, -23);
131             wait for 100 ns;
132             start <= '0';
133             wait for 1500 ns;
134         end process trial;
135 
136     end behavioral;
```

B.2 Automatizzazione della simulazione con C++

B.3 Generazione di vettori di I/O mediante script MATLAB

```
1  close all
2  clearvars
3  clc
4  format long
5  
6  N = 16; % numero di campioni
7  Wr_vec = zeros(1, N/2); % vettore delle parti reali
8  Wi_vec = zeros(1, N/2); % vettore delle parti immaginarie
9  W_vec = zeros(1, N/2); % vettore dei twiddle factors completi
10 q = quantizer([24,23]); % quantizzatore (24 bit di cui 23 frazionari)
11 
12 %% Calcolo dei twiddle factors
13 for j=1:N/2
14     k = j - 1;
15     x = 2 * pi * k / N;
16     W_vec(j) = cos(x) - 1i*sin(x);
17     Wr_vec(j) = real(W_vec(j));
18     Wi_vec(j) = imag(W_vec(j));
19 end
20 
21 %% Ingressi butterfly primo livello, dinamica -0.5/+0.5
22 dinamica_L1_W0 = [-0.5 0.5];
23 ingressi_L1_W0 = worstCaseInput(dinamica_L1_W0, q); % in C2
24 writematrix(ingressi_L1_W0)
25 
26 %% Butterfly primo livello con W = W0
27 Wr = Wr_vec(1); % 1
28 Wi = Wi_vec(1); % 0
29 [risultati_L1_W0, dinamica_L2_W0_W4] = butterflyRange(Wr, Wi, dinamica_L1_W0); % uscite
   livello 1 (W0) e relativa dinamica
```

```
ingressi_L2_W0_W4 = worstCaseInput(dinamica_L2_W0_W4, q); % ingressi ai limiti della dinamica livello 2 (W0-W4)

writematrix(ingressi_L2_W0_W4)
writematrix(risultati_L1_W0,'risultati_L1_W0_matlab.txt','Delimiter','space')

%% Butterfly secondo livello con W = W0
Wr = Wr_vec(1); % 1
Wi = Wi_vec(1); % 0
[risultati_L2_W0, dinamica_L3_W0_W4] = butterflyRange(Wr, Wi, dinamica_L2_W0_W4); % uscite livello 2 (W0) e relativa dinamica
ingressi_L3_W0_W4 = worstCaseInput(dinamica_L3_W0_W4, q); % ingressi ai limiti della dinamica livello 3 (W0-W4)

writematrix(ingressi_L3_W0_W4)
writematrix(risultati_L2_W0,'risultati_L2_W0_matlab.txt','Delimiter','space')

%% Butterfly secondo livello con W = W4
Wr = Wr_vec(5); % 0
Wi = Wi_vec(5); % -1
[risultati_L2_W4, dinamica_L3_W2_W6] = butterflyRange(Wr, Wi, dinamica_L2_W0_W4); % uscite livello 2 (W4) e relativa dinamica
ingressi_L3_W2_W6 = worstCaseInput(dinamica_L3_W2_W6, q); % ingressi ai limiti della dinamica livello 3 (W2-W6)

writematrix(ingressi_L3_W2_W6)
writematrix(risultati_L2_W4,'risultati_L2_W4_matlab.txt','Delimiter','space')

%% Butterfly terzo livello con W = W0
Wr = Wr_vec(1); % 1
Wi = Wi_vec(1); % 0
[risultati_L3_W0, dinamica_L4_W0_W4] = butterflyRange(Wr, Wi, dinamica_L3_W0_W4); % uscite livello 3 (W0) e relativa dinamica
ingressi_L4_W0_W4 = worstCaseInput(dinamica_L4_W0_W4, q); % ingressi ai limiti della dinamica livello 4 (W0-W4)

writematrix(ingressi_L4_W0_W4)
writematrix(risultati_L3_W0,'risultati_L3_W0_matlab.txt','Delimiter','space')

%% Butterfly terzo livello con W = W4
Wr = Wr_vec(5); % 0
Wi = Wi_vec(5); % -1
[risultati_L3_W4, dinamica_L4_W2_W6] = butterflyRange(Wr, Wi, dinamica_L3_W0_W4); % uscite livello 3 (W4) e relativa dinamica
ingressi_L4_W2_W6 = worstCaseInput(dinamica_L4_W2_W6, q); % ingressi ai limiti della dinamica livello 4 (W2-W6)

writematrix(ingressi_L4_W2_W6)
writematrix(risultati_L3_W4,'risultati_L3_W4_matlab.txt','Delimiter','space')

%% Butterfly terzo livello con W = W2
Wr = Wr_vec(3); % 0.707106781186548
Wi = Wi_vec(3); % -0.707106781186548
[risultati_L3_W2, dinamica_L4_W1_W5] = butterflyRange(Wr, Wi, dinamica_L3_W2_W6); % uscite livello 3 (W2) e relativa dinamica
ingressi_L4_W1_W5 = worstCaseInput(dinamica_L4_W1_W5, q); % ingressi ai limiti della dinamica livello 4 (W1-W5)

writematrix(ingressi_L4_W1_W5)
```

```
78  writematrix(risultati_L3_W2,'risultati_L3_W2_matlab.txt','Delimiter','space')
79
80  %% Butterfly terzo livello con W = W6
81  Wr = Wr_vec(7); % -0.707106781186548
82  Wi = Wi_vec(7); % -0.707106781186548
83  [risultati_L3_W6, dinamica_L4_W3_W7] = butterflyRange(Wr, Wi, dinamica_L3_W2_W6); %
    uscite livello 3 (W6) e relativa dinamica
84  ingressi_L4_W3_W7 = worstCaseInput(dinamica_L4_W3_W7, q); % ingressi ai limiti della
    dinamica livello 4 (W3-W7)
85
86  writematrix(ingressi_L4_W3_W7)
87  writematrix(risultati_L3_W6,'risultati_L3_W6_matlab.txt','Delimiter','space')
88
89  %% Butterfly quarto livello con W = W0
90  Wr = Wr_vec(1); % 1
91  Wi = Wi_vec(1); % 0
92  [risultati_L4_W0, dinamica_uscita_L4_W0] = butterflyRange(Wr, Wi, dinamica_L4_W0_W4); %
    uscite livello 4 (W0) e relativa dinamica
93
94  writematrix(risultati_L4_W0,'risultati_L4_W0_matlab.txt','Delimiter','space')
95
96  %% Butterfly quarto livello con W = W4
97  Wr = Wr_vec(5); % 0
98  Wi = Wi_vec(5); % -1
99  [risultati_L4_W4, dinamica_uscita_L4_W4] = butterflyRange(Wr, Wi, dinamica_L4_W0_W4); %
    uscite livello 4 (W4) e relativa dinamica
100
101 writematrix(risultati_L4_W4,'risultati_L4_W4_matlab.txt','Delimiter','space')
102
103 %% Butterfly quarto livello con W = W2
104 Wr = Wr_vec(3); % 0.707106781186548
105 Wi = Wi_vec(3); % -0.707106781186548
106 [risultati_L4_W2, dinamica_uscita_L4_W2] = butterflyRange(Wr, Wi, dinamica_L4_W2_W6); %
    uscite livello 4 (W2) e relativa dinamica
107
108 writematrix(risultati_L4_W2,'risultati_L4_W2_matlab.txt','Delimiter','space')
109
110 %% Butterfly quarto livello con W = W6
111 Wr = Wr_vec(7); % -0.707106781186548
112 Wi = Wi_vec(7); % -0.707106781186548
113 [risultati_L4_W6, dinamica_uscita_L4_W6] = butterflyRange(Wr, Wi, dinamica_L4_W2_W6); %
    uscite livello 4 (W6) e relativa dinamica
114
115 writematrix(risultati_L4_W6,'risultati_L4_W6_matlab.txt','Delimiter','space')
116
117 %% Butterfly quarto livello con W = W1
118 Wr = Wr_vec(2); % 0.923879532511287
119 Wi = Wi_vec(2); % -0.382683432365090
120 [risultati_L4_W1, dinamica_uscita_L4_W1] = butterflyRange(Wr, Wi, dinamica_L4_W1_W5); %
    uscite livello 4 (W1) e relativa dinamica
121
122 writematrix(risultati_L4_W1,'risultati_L4_W1_matlab.txt','Delimiter','space')
123
124 %% Butterfly quarto livello con W = W5
125 Wr = Wr_vec(6); % -0.382683432365090
126 Wi = Wi_vec(6); % -0.923879532511287
127 [risultati_L4_W5, dinamica_uscita_L4_W5] = butterflyRange(Wr, Wi, dinamica_L4_W1_W5); %
    uscite livello 4 (W5) e relativa dinamica
128
```

```
writematrix(risultati_L4_W5,'risultati_L4_W5_matlab.txt','Delimiter','space')

%% Butterfly quarto livello con W = W3
Wr = Wr_vec(4); % 0.382683432365090
Wi = Wi_vec(4); % -0.923879532511287
[risultati_L4_W3, dinamica_uscita_L4_W3] = butterflyRange(Wr, Wi, dinamica_L4_W3_W7); % uscite livello 4 (W3) e relativa dinamica

writematrix(risultati_L4_W3,'risultati_L4_W3_matlab.txt','Delimiter','space')

%% Butterfly quarto livello con W = W7
Wr = Wr_vec(8); % -0.923879532511287
Wi = Wi_vec(8); % -0.382683432365090
[risultati_L4_W7, dinamica_uscita_L4_W7] = butterflyRange(Wr, Wi, dinamica_L4_W3_W7); % uscite livello 4 (W7) e relativa dinamica

writematrix(risultati_L4_W7,'risultati_L4_W7_matlab.txt','Delimiter','space')


%% Funzione per calcolare le 4 uscite (A', B') di una singola butterfly dati i 6 ingressi (A, B, W)
function [Ar_primo, Ai_primo, Br_primo, Bi_primo] = butterfly(Ar, Ai, Br, Bi, Wr, Wi)
% Moltiplicazioni
M1 = Br * Wr;
M2 = Bi * Wi;
M3 = Br * Wi;
M4 = Bi * Wr;
M5 = 2 * Ar;
M6 = 2 * Ai;

% Somme e sottrazioni
sigma_1 = Ar + M1;
sigma_2 = sigma_1 - M2;
sigma_3 = Ai + M3;
sigma_4 = sigma_3 + M4;
sigma_5 = M5 - sigma_2;
sigma_6 = M6 - sigma_4;

Ar_primo = sigma_2;
Ai_primo = sigma_4;
Br_primo = sigma_5;
Bi_primo = sigma_6;

% Arrotondamenti per riportare la dinamica a -1/+1
if abs(Ar_primo) >= 1
    while abs(Ar_primo) >= 1
        Ar_primo = Ar_primo / 2;
    end
else
    Ar_primo = Ar_primo / 2;
end

if abs(Ai_primo) >= 1
    while abs(Ai_primo) >= 1
        Ai_primo = Ai_primo / 2;
    end
else
    Ai_primo = Ai_primo / 2;
end
```

```
185
186 if abs(Br_primo) >= 1
187     while abs(Br_primo) >= 1
188         Br_primo = Br_primo / 2;
189     end
190 else
191     Br_primo = Br_primo / 2;
192 end
193
194 if abs(Bi_primo) >= 1
195     while abs(Bi_primo) >= 1
196         Bi_primo = Bi_primo / 2;
197     end
198 else
199     Bi_primo = Bi_primo / 2;
200 end
201
202 end
203
204 %% Funzione per calcolare la dinamica di uscita della butterfly al variare della
    dinamica di ingresso e del twiddle factor W
205 % @return output_data_real = matrice con i risultati corrispondenti ad ingressi ai
    limiti della dinamica
206 % @return output_range = dinamica di uscita della butterfly
207 % I dati uscita verranno usati per il confronto con i risultati della simulazione,
    quindi conviene generare una matrice di numeri reali 16x4
208     function [output_data_real, output_range] = butterflyRange(Wr, Wi, range)
209         % calcolo le uscite con tutte le combinazioni di ingressi
210         output_data_real = zeros(16,4);
211         i = 1;
212         for n1=1:2
213             for n2=1:2
214                 for n3=1:2
215                     for n4=1:2
216                         Ar = range(n1);
217                         Ai = range(n2);
218                         Br = range(n3);
219                         Bi = range(n4);
220                         [Ar_primo, Ai_primo, Br_primo, Bi_primo] = butterfly(Ar, Ai, Br,
                            Bi, Wr, Wi);
221                         output_data_real(i,1) = Ar_primo;
222                         output_data_real(i,2) = Ai_primo;
223                         output_data_real(i,3) = Br_primo;
224                         output_data_real(i,4) = Bi_primo;
225                         i = i + 1;
226                     end
227                 end
228             end
229         end
230
231         % ciclo sugli elementi della matrice e sostituisco con degli zeri tutti i numeri
            inferiori in modulo a 1e-12
232         for i=1:16
233             for j=1:4
234                 if abs(output_data_real(i,j)) < 1e-12
235                     output_data_real(i,j) = 0;
236                 end
237             end
238         end
```

```
239
240         % definisco il nuovo range
241         output_range = [min(output_data_real,[],"all") max(output_data_real,[],"all")];
242     end
243
244 %% Funzione per calcolare tutte le combinazioni di ingressi ai limiti della dinamica
245 % La matrice risultante verrà data in pasto a Modelsim, quindi è conveniente generare
    una matrice 4x16 e convertirne gli elementi in C2 con la funzione dec2bin
246     function output_data = worstCaseInput(input_range, quantizer)
247         output_data_real = zeros(4,16);
248         i = 1;
249         for n1=1:2
250             for n2=1:2
251                 for n3=1:2
252                     for n4=1:2
253                         output_data_real(1,i) = input_range(n1);
254                         output_data_real(2,i) = input_range(n2);
255                         output_data_real(3,i) = input_range(n3);
256                         output_data_real(4,i) = input_range(n4);
257                         i = i + 1;
258                     end
259                 end
260             end
261         end
262         output_data = num2bin(quantizer, output_data_real);
263     end
```

B.4 Testbench automatizzata

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4  use ieee.fixed_pkg.all;
5  use std.textio.all;
6  use ieee.std_logic_textio.all;
7
8  entity tb_butterfly is
9  end tb_butterfly;
10
11 architecture behavioral of tb_butterfly is
12
13     -- definizione segnali interni
14     signal clock       : std_logic            := '0';
15     signal reset       : std_logic            := '0';
16     signal start       : std_logic            := '0';
17     signal done        : std_logic            := '0';
18     signal Ar_in       : sfixed(0 downto -23) := (others => '0');
19     signal Ai_in       : sfixed(0 downto -23) := (others => '0');
20     signal Br_in       : sfixed(0 downto -23) := (others => '0');
21     signal Bi_in       : sfixed(0 downto -23) := (others => '0');
22     signal Wr_in       : sfixed(0 downto -23) := (others => '0');
23     signal Wi_in       : sfixed(0 downto -23) := (others => '0');
24     signal Ar_out      : sfixed(0 downto -23);
25     signal Ai_out      : sfixed(0 downto -23);
26     signal Br_out      : sfixed(0 downto -23);
27     signal Bi_out      : sfixed(0 downto -23);
28     signal Ar_out_real : real;
```

```
29      signal Ai_out_real : real;
30      signal Br_out_real : real;
31      signal Bi_out_real : real;
32
33      -- definizione file di I/O
34      file file_INPUT_W  : text;
35      file file_INPUT_AB : text;
36      file file_OUTPUT   : text;
37
38      -- dichiarazione UUT
39      component butterfly is
40          port (
41              CK    : in std_logic;
42              RST   : in std_logic;
43              START : in std_logic;
44              DONE  : out std_logic;
45              --* porte di ingresso *****************************************************
46              Ar : in sfixed(0 downto -23);
47              Ai : in sfixed(0 downto -23);
48              Br : in sfixed(0 downto -23);
49              Bi : in sfixed(0 downto -23);
50              Wr : in sfixed(0 downto -23);
51              Wi : in sfixed(0 downto -23);
52              --* porte di uscita ******************************************************
53              Ar_primo : out sfixed(0 downto -23);
54              Ai_primo : out sfixed(0 downto -23);
55              Br_primo : out sfixed(0 downto -23);
56              Bi_primo : out sfixed(0 downto -23)
57          );
58      end component;
59
60  begin
61
62      -- istanza UUT
63      uP_BUTTERFLY : butterfly
64      port map(
65          CK       => clock,
66          RST      => reset,
67          START    => start,
68          DONE     => done,
69          Ar       => Ar_in,
70          Ai       => Ai_in,
71          Br       => Br_in,
72          Bi       => Bi_in,
73          Wr       => Wr_in,
74          Wi       => Wi_in,
75          Ar_primo => Ar_out,
76          Ai_primo => Ai_out,
77          Br_primo => Br_out,
78          Bi_primo => Bi_out
79      );
80
81      -- conversione dei risultati in numeri reali
82      Ar_out_real <= to_real(Ar_out);
83      Ai_out_real <= to_real(Ai_out);
84      Br_out_real <= to_real(Br_out);
85      Bi_out_real <= to_real(Bi_out);
86
87      -- process per la generazione del clock
```

```
88      clk_process : process
89      begin
90          wait for 50 ns;
91          clock <= not clock;
92      end process clk_process;
93
94      -- process per lettura da file e calcoli
95      calc_process : process
96
97          variable v_ILINE_W  : line;      -- riga file input W
98          variable v_ILINE_Ar : line;
99          variable v_ILINE_Ai : line;
100         variable v_ILINE_Br : line;
101         variable v_ILINE_Bi : line;
102         variable v_OLINE    : line;      -- riga file output
103         variable v_SPACE    : character; -- carattere spazio
104
105         variable v_Ar : sfixed(0 downto -23);
106         variable v_Ai : sfixed(0 downto -23);
107         variable v_Br : sfixed(0 downto -23);
108         variable v_Bi : sfixed(0 downto -23);
109         variable v_Wr : sfixed(0 downto -23);
110         variable v_Wi : sfixed(0 downto -23);
111
112     begin
113         -- Apro file di input in modalità di lettura
114         file_open(file_INPUT_W, "twiddle_W0.txt", read_mode);
115         file_open(file_INPUT_AB, "ingressi_L1_W0.txt", read_mode);
116         file_open(file_OUTPUT, "risultati_L1_W0_tb.txt", write_mode);
117
118         -- Reset macchina
119         reset <= '1';
120         wait for 2 ns;
121         reset <= '0';
122
123         -- Leggo la coppia di twiddle factor da usare per la simulazione
124         readline(file_INPUT_W, v_ILINE_W);
125         read(v_ILINE_W, v_Wr);    -- Wr
126         read(v_ILINE_W, v_SPACE); -- spazio
127         read(v_ILINE_W, v_Wi);    -- Wi
128
129         while not endfile(file_INPUT_AB) loop
130             -- Ar --
131             readline(file_INPUT_AB, v_ILINE_Ar);
132             read(v_ILINE_Ar, v_Ar);
133             -- Ai --
134             readline(file_INPUT_AB, v_ILINE_Ai);
135             read(v_ILINE_Ai, v_Ai);
136             -- Br --
137             readline(file_INPUT_AB, v_ILINE_Br);
138             read(v_ILINE_Br, v_Br);
139             -- Bi --
140             readline(file_INPUT_AB, v_ILINE_Bi);
141             read(v_ILINE_Bi, v_Bi);
142
143             -- Passo le variabili ai corrispondenti segnali per poterle usare nei
                calcoli
144             Wr_in <= v_Wr;
145             Wi_in <= v_Wi;
```

```
            Ar_in <= v_Ar;
            Ai_in <= v_Ai;
            Br_in <= v_Br;
            Bi_in <= v_Bi;

            -- Faccio partire il processore
            wait for 100 ns;
            start <= '1';
            wait for 100 ns;
            start <= '0';
            wait for 1150 ns;

            -- Scrivo in output_results.txt
            write(v_OLINE, Ar_out_real);
            write(v_OLINE, v_SPACE);
            write(v_OLINE, Ai_out_real);
            write(v_OLINE, v_SPACE);
            write(v_OLINE, Br_out_real);
            write(v_OLINE, v_SPACE);
            write(v_OLINE, Bi_out_real);
            writeline(file_OUTPUT, v_OLINE);

            wait for 350 ns;

        end loop;

        -- Closing In/Out files
        file_close(file_input_W);
        file_close(file_input_AB);
        file_close(file_OUTPUT);

    end process;

end behavioral;
```

B.4.1 Classe Simulation

```
#include <iostream>
#include <fstream>    // per file processing
#include <sstream>    // per trattare le stringhe come stream di dati
#include <filesystem> // per verificare l'esistenza dei file
#include <cstdlib>    // per usare comandi shell
#include <string>     // per manipolazione stringhe
#include <cstring>    // per manipolazione stringhe C-like
#include <vector>     // per manipolazione vettori
#include <cmath>      // per funzioni matematiche
#include <iomanip>    // per formattazione I/O

#include "Tools.hpp"
#include "Simulation.hpp"

using namespace std;

// Costruttore esplicito
Simulation::Simulation(unsigned int c)
    : correct{c} {}

```

```
/*****************************************************************************
Modifica testbench
In totale ci sono 15 implementazioni:
Implementazioni 0, 1, 3, 7: W = W0
Implementazioni 2, 4, 8: W = W4
Implementazioni 5, 9: W = W2
Implementazioni 6, 10: W = W6
Implementazione 11: W = W1
Implementazione 12: W = W5
Implementazione 13: W = W3
Implementazione 14: W = W7
*****************************************************************************/
void Simulation::changeTB(unsigned int implementazione, string tb)
{
    string tFileName_iniziale;
    string iFileName_iniziale;
    string oFileName_iniziale;
    string tFileName_finale;
    string iFileName_finale;
    string oFileName_finale;

    switch (implementazione)
    {
    case 0: // L1 W0 (non sostituitsco nulla, i nomi dei file sono già corretti)
        break;
    case 1: // L2 W0 (non sostituisco file W, sostituisco file input, sostituisco file output)
        system(("sed -i -e 's/ingressi_L1_W0.txt/ingressi_L2_W0_W4.txt/' " + tb).c_str());
        system(("sed -i -e 's/risultati_L1_W0_tb.txt/risultati_L2_W0_tb.txt/' " + tb).c_str());
        break;
    case 2: // L2 W4 (sostituisco file W, non sostituisco file input, sostituisco file output)
        system(("sed -i -e 's/twiddle_W0.txt/twiddle_W4.txt/' " + tb).c_str());
        system(("sed -i -e 's/risultati_L2_W0_tb.txt/risultati_L2_W4_tb.txt/' " + tb).c_str());
        break;
    case 3: // L3 W0 (sostituisco file W, sostituisco file input, sostituisco file output)
        system(("sed -i -e 's/twiddle_W4.txt/twiddle_W0.txt/' " + tb).c_str());
        system(("sed -i -e 's/ingressi_L2_W0_W4.txt/ingressi_L3_W0_W4.txt/' " + tb).c_str());
        system(("sed -i -e 's/risultati_L2_W4_tb.txt/risultati_L3_W0_tb.txt/' " + tb).c_str());
        break;
    case 4: // L3 W4 (sostituisco file W, non sostituisco file input, sostituisco file output)
        system(("sed -i -e 's/twiddle_W0.txt/twiddle_W4.txt/' " + tb).c_str());
        system(("sed -i -e 's/risultati_L3_W0_tb.txt/risultati_L3_W4_tb.txt/' " + tb).c_str());
        break;
    case 5: // L3 W2 (sostituisco file W, sostituisco file input, sostituisco file output)
        system(("sed -i -e 's/twiddle_W4.txt/twiddle_W2.txt/' " + tb).c_str());
        system(("sed -i -e 's/ingressi_L3_W0_W4.txt/ingressi_L3_W2_W6.txt/' " + tb).c_str());
        system(("sed -i -e 's/risultati_L3_W4_tb.txt/risultati_L3_W2_tb.txt/' " + tb).c_str());
```

```
67          break;
68      case 6: // L3 W6 (sostituisco file W, non sostituisco file input, sostituisco file
        ↪ output)
69          system(("sed -i -e 's/twiddle_W2.txt/twiddle_W6.txt/' " + tb).c_str());
70          system(("sed -i -e 's/risultati_L3_W2_tb.txt/risultati_L3_W6_tb.txt/' " +
            ↪ tb).c_str());
71          break;
72      case 7: // L4 W0 (sostituisco file W, sostituisco file input, sostituisco file
        ↪ output)
73          system(("sed -i -e 's/twiddle_W6.txt/twiddle_W0.txt/' " + tb).c_str());
74          system(("sed -i -e 's/ingressi_L3_W2_W6.txt/ingressi_L4_W0_W4.txt/' " +
            ↪ tb).c_str());
75          system(("sed -i -e 's/risultati_L3_W6_tb.txt/risultati_L4_W0_tb.txt/' " +
            ↪ tb).c_str());
76          break;
77      case 8: // L4 W4 (sostituisco file W, non sostituisco file input, sostituisco file
        ↪ output)
78          system(("sed -i -e 's/twiddle_W0.txt/twiddle_W4.txt/' " + tb).c_str());
79          system(("sed -i -e 's/risultati_L4_W0_tb.txt/risultati_L4_W4_tb.txt/' " +
            ↪ tb).c_str());
80          break;
81      case 9: // L4 W2 (sostituisco file W, sostituisco file input, sostituisco file
        ↪ output)
82          system(("sed -i -e 's/twiddle_W4.txt/twiddle_W2.txt/' " + tb).c_str());
83          system(("sed -i -e 's/ingressi_L4_W0_W4.txt/ingressi_L4_W2_W6.txt/' " +
            ↪ tb).c_str());
84          system(("sed -i -e 's/risultati_L4_W4_tb.txt/risultati_L4_W2_tb.txt/' " +
            ↪ tb).c_str());
85          break;
86      case 10: // L4 W6 (sostituisco file W, non sostituisco file input, sostituisco file
        ↪ output)
87          system(("sed -i -e 's/twiddle_W2.txt/twiddle_W6.txt/' " + tb).c_str());
88          system(("sed -i -e 's/risultati_L4_W2_tb.txt/risultati_L4_W6_tb.txt/' " +
            ↪ tb).c_str());
89          break;
90      case 11: // L4 W1 (sostituisco file W, sostituisco file input, sostituisco file
        ↪ output)
91          system(("sed -i -e 's/twiddle_W6.txt/twiddle_W1.txt/' " + tb).c_str());
92          system(("sed -i -e 's/ingressi_L4_W2_W6.txt/ingressi_L4_W1_W5.txt/' " +
            ↪ tb).c_str());
93          system(("sed -i -e 's/risultati_L4_W6_tb.txt/risultati_L4_W1_tb.txt/' " +
            ↪ tb).c_str());
94          break;
95      case 12: // L4 W5 (sostituisco file W, non sostituisco file input, sostituisco file
        ↪ output)
96          system(("sed -i -e 's/twiddle_W1.txt/twiddle_W5.txt/' " + tb).c_str());
97          system(("sed -i -e 's/risultati_L4_W1_tb.txt/risultati_L4_W5_tb.txt/' " +
            ↪ tb).c_str());
98          break;
99      case 13: // L4 W3 (sostituisco file W, sostituisco file input, sostituisco file
        ↪ output)
100         system(("sed -i -e 's/twiddle_W5.txt/twiddle_W3.txt/' " + tb).c_str());
101         system(("sed -i -e 's/ingressi_L4_W1_W5.txt/ingressi_L4_W3_W7.txt/' " +
            ↪ tb).c_str());
102         system(("sed -i -e 's/risultati_L4_W5_tb.txt/risultati_L4_W3_tb.txt/' " +
            ↪ tb).c_str());
103         break;
104     case 14: // L4 W7 (sostituisco file W, non sostituisco file input, sostituisco file
        ↪ output)
```

```
105             system(("sed -i -e 's/twiddle_W3.txt/twiddle_W7.txt/' " + tb).c_str());
106             system(("sed -i -e 's/risultati_L4_W3_tb.txt/risultati_L4_W7_tb.txt/' " +
                ↪  tb).c_str());
107             break;
108         default:
109             break;
110         }
111     }
112
113     // Esecuzione simulazione
114     void Simulation::run(string fileCompilazione)
115     {
116         system(("vsim -c -do " + fileCompilazione).c_str()); // lancio la simulazione
117     }
118
119     // Riporto la TB alle condizioni di partenza
120     void Simulation::restoreTB(string tb)
121     {
122         system(("sed -i -e 's/twiddle_W7.txt/twiddle_W0.txt/' " + tb).c_str());
123         system(("sed -i -e 's/ingressi_L4_W3_W7.txt/ingressi_L1_W0.txt/' " + tb).c_str());
124         system(("sed -i -e 's/risultati_L4_W7_tb.txt/risultati_L1_W0_tb.txt/' " +
        ↪  tb).c_str());
125     }
126
127     // Controlla la correttezza dei risultati
128     unsigned int Simulation::report(string risultati_tb, string risultati_matlab)
129     {
130         string line_tb, line_matlab; // righe dei due file
131         int cnt_tb = 0;              // contatore di riga del file dei vettori di ingresso
132         int cnt_matlab = 0;          // contatore di riga del file dei risultati
133         int tot_correct = 0;         // numero totale di righe corrette
134
135         // apro i file in lettura (niente controllo perché so che esistono)
136         ifstream tbF(risultati_tb);
137         ifstream matlabF(risultati_matlab);
138
139         while ((tbF.good() && matlabF.good()))
140         {
141             // estraggo una riga da ognuno dei due file
142             if (getline(tbF, line_tb) && getline(matlabF, line_matlab))
143             {
144                 // separo i campi della riga della TB
145                 istringstream iss_tb(line_tb);
146                 vector<string> fields_tb;
147                 string field_tb;
148                 while (getline(iss_tb, field_tb, ' '))
149                 {
150                     fields_tb.push_back(field_tb);
151                 }
152
153                 // separo i campi della riga di MATLAB
154                 istringstream iss_matlab(line_matlab);
155                 vector<string> fields_matlab;
156                 string field_matlab;
157                 while (getline(iss_matlab, field_matlab, ' '))
158                 {
159                     fields_matlab.push_back(field_matlab);
160                 }
161
```

```
            // confronto i risultati
            float tolleranza = 1e-6;
            float diff_0 = abs(stof(fields_tb[0]) - stof(fields_matlab[0]));
            float diff_1 = abs(stof(fields_tb[1]) - stof(fields_matlab[1]));
            float diff_2 = abs(stof(fields_tb[2]) - stof(fields_matlab[2]));
            float diff_3 = abs(stof(fields_tb[3]) - stof(fields_matlab[3]));

            // se i risultati ottenuti sono uguali (a meno della tolleranza), incremento
            ↪  il numero di righe corrette
            if (diff_0 <= tolleranza && diff_1 <= tolleranza &&
                diff_2 <= tolleranza && diff_3 <= tolleranza)
            {
                tot_correct++;
            }
            // se i risultati sono diversi, esco dal ciclo (non ho più bisogno di
            ↪  controllare le righe restanti)
            else
            {
                break;
            }

            // incremento i contatori di riga
            cnt_tb++;
            cnt_matlab++;
        }
    }

    // chiudo i file
    tbF.close();
    matlabF.close();

    // stabilisco il valore del flag che mi dice se la simulazione è andata a buon fine
    if ((cnt_tb == cnt_matlab) && (tot_correct == cnt_matlab))
        correct = 1;
    else
        correct = 0;

    return correct;
}
```

B.4.2 Main

```
#include <iostream>
#include <fstream>    // per file processing
#include <sstream>    // per trattare le stringhe come stream di dati
#include <filesystem> // per verificare l'esistenza dei file
#include <cstdlib>    // per usare comandi shell
#include <string>     // per manipolazione stringhe
#include <cstring>    // per manipolazione stringhe C-like
#include <vector>     // per manipolazione vettori
#include <cmath>      // per funzioni matematiche
#include <iomanip>    // per formattazione I/O

#include "Tools.hpp"
#include "Simulation.hpp"

using namespace std;
```

```
16
17  int main(int argc, char **argv)
18  {
19
20      int ret = 0;          // variabile per il return
21      Simulation Simulator; // oggetto della classe Simulation per l'automatizzazione
        ↪  della simulazione
22
23      /*****************************************************************************/
24      /******** Inizializzazione degli oggetti necessari alla gestione dei file *********/
25
26      const string tbFileName = "tb_butterfly.vhd"; // testbench
27      const string compileFileName = "compile.do";  // file con le info per la simulazione
28
29      /*****************************************************************************/
30
31      // check esistenza testbench
32      if (!filesystem::exists(tbFileName))
33      {
34          cerr << "Errore! La testbench " << tbFileName << " non esiste." << endl;
35          ret = 1;
36      }
37
38      // check esistenza file per la compilazione
39      if (!filesystem::exists(compileFileName))
40      {
41          cerr << "Errore! Il file per la compilazione " << compileFileName << " non
            ↪  esiste." << endl;
42          ret = 1;
43      }
44
45      // simulazione automatizzata
46      for (int i = 0; i < 15; i++)
47      {
48          cout << endl
49               << "*************************************************************************"
                 ↪  << endl;
50          cout << "Inizio Iterazione " << i + 1 << endl
51               << endl;
52          Simulator.changeTB(i, tbFileName);
53          Simulator.run(compileFileName);
54          cout << endl
55               << "Fine Iterazione " << i + 1 << endl;
56          cout << "*************************************************************************"
            ↪  << endl
57               << endl;
58      }
59
60      // ripristino della testbench di partenza
61      Simulator.restoreTB(tbFileName);
62
63      // confronto TB e MATLAB
64      int w; // twiddle factor
65      int l; // livello
66      string matlab;
67      string testbench;
68      int simulazioni_corrette = 0;
69      for (w = 0; w < 8; w++)
70      {
```

```
71          if (w == 0)
72          {
73              for (l = 1; l <= 4; l++)
74              {
75                  matlab = "risultati_L" + to_string(l) + "_W" + to_string(w) +
                    ↪  "_matlab.txt";
76                  testbench = "risultati_L" + to_string(l) + "_W" + to_string(w) +
                    ↪  "_tb.txt";
77                  if (Simulator.report(testbench, matlab))
78                      simulazioni_corrette++;
79              }
80          }
81          else if (w == 4)
82          {
83              for (l = 2; l <= 4; l++)
84              {
85                  matlab = "risultati_L" + to_string(l) + "_W" + to_string(w) +
                    ↪  "_matlab.txt";
86                  testbench = "risultati_L" + to_string(l) + "_W" + to_string(w) +
                    ↪  "_tb.txt";
87                  if (Simulator.report(testbench, matlab))
88                      simulazioni_corrette++;
89              }
90          }
91          else if (w == 2 || w == 6)
92          {
93              for (l = 3; l <= 4; l++)
94              {
95                  matlab = "risultati_L" + to_string(l) + "_W" + to_string(w) +
                    ↪  "_matlab.txt";
96                  testbench = "risultati_L" + to_string(l) + "_W" + to_string(w) +
                    ↪  "_tb.txt";
97                  if (Simulator.report(testbench, matlab))
98                      simulazioni_corrette++;
99              }
100         }
101         else
102         {
103             l = 4;
104             matlab = "risultati_L" + to_string(l) + "_W" + to_string(w) + "_matlab.txt";
105             testbench = "risultati_L" + to_string(l) + "_W" + to_string(w) + "_tb.txt";
106             if (Simulator.report(testbench, matlab))
107                 simulazioni_corrette++;
108         }
109     }
110
111     if (simulazioni_corrette == 15)
112     {
113         cout << endl
114              << "OK! Tutte le simulazioni Modelsim hanno prodotto gli stessi risultati
                 ↪  previsti da MATLAB." << endl;
115         cout << "Verosimilmente il processore Butterfly funziona! :)" << endl
116              << endl;
117     }
118     else
119     {
120         cout << endl
121              << "Non tutte le simulazioni Modelsim hanno prodotto gli stessi risultati
                 ↪  previsti da MATLAB." << endl;
```

```
122             cout << "Il processore Butterfly non funziona. :(" << endl
123                  << endl;
124         }
125
126         return ret; // punto di uscita dal programma
127     }
```

C Test calcolatore FFT

C.1 Testbench

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.fixed_pkg.all;
use std.textio.all;
use ieee.std_logic_textio.all;

entity tb_fft is
end tb_fft;

architecture behavioral of tb_fft is

    -- definizione segnali interni
    signal clock                : std_logic            := '0';
    signal reset                : std_logic            := '0';
    signal start                : std_logic            := '0';
    signal done                 : std_logic            := '0';
    signal Xr_in_0, Xi_in_0     : sfixed(0 downto -23) := (others => '0');
    signal Xr_in_1, Xi_in_1     : sfixed(0 downto -23) := (others => '0');
    signal Xr_in_2, Xi_in_2     : sfixed(0 downto -23) := (others => '0');
    signal Xr_in_3, Xi_in_3     : sfixed(0 downto -23) := (others => '0');
    signal Xr_in_4, Xi_in_4     : sfixed(0 downto -23) := (others => '0');
    signal Xr_in_5, Xi_in_5     : sfixed(0 downto -23) := (others => '0');
    signal Xr_in_6, Xi_in_6     : sfixed(0 downto -23) := (others => '0');
    signal Xr_in_7, Xi_in_7     : sfixed(0 downto -23) := (others => '0');
    signal Xr_in_8, Xi_in_8     : sfixed(0 downto -23) := (others => '0');
    signal Xr_in_9, Xi_in_9     : sfixed(0 downto -23) := (others => '0');
    signal Xr_in_10, Xi_in_10   : sfixed(0 downto -23) := (others => '0');
    signal Xr_in_11, Xi_in_11   : sfixed(0 downto -23) := (others => '0');
    signal Xr_in_12, Xi_in_12   : sfixed(0 downto -23) := (others => '0');
    signal Xr_in_13, Xi_in_13   : sfixed(0 downto -23) := (others => '0');
    signal Xr_in_14, Xi_in_14   : sfixed(0 downto -23) := (others => '0');
    signal Xr_in_15, Xi_in_15   : sfixed(0 downto -23) := (others => '0');
    signal Yr_out_0, Yi_out_0   : sfixed(0 downto -23);
    signal Yr_out_1, Yi_out_1   : sfixed(0 downto -23);
    signal Yr_out_2, Yi_out_2   : sfixed(0 downto -23);
    signal Yr_out_3, Yi_out_3   : sfixed(0 downto -23);
    signal Yr_out_4, Yi_out_4   : sfixed(0 downto -23);
    signal Yr_out_5, Yi_out_5   : sfixed(0 downto -23);
    signal Yr_out_6, Yi_out_6   : sfixed(0 downto -23);
    signal Yr_out_7, Yi_out_7   : sfixed(0 downto -23);
    signal Yr_out_8, Yi_out_8   : sfixed(0 downto -23);
    signal Yr_out_9, Yi_out_9   : sfixed(0 downto -23);
    signal Yr_out_10, Yi_out_10 : sfixed(0 downto -23);
    signal Yr_out_11, Yi_out_11 : sfixed(0 downto -23);
    signal Yr_out_12, Yi_out_12 : sfixed(0 downto -23);
    signal Yr_out_13, Yi_out_13 : sfixed(0 downto -23);
    signal Yr_out_14, Yi_out_14 : sfixed(0 downto -23);
    signal Yr_out_15, Yi_out_15 : sfixed(0 downto -23);

    -- definizione file di I/O
    file file_INPUT  : text;

    -- dichiarazione UUT
```

```
55      component FFT_16 is
56          port (
57              CK    : in std_logic;
58              RST   : in std_logic;
59              START : in std_logic;
60              DONE  : out std_logic;
61              --* porte di ingresso ******************************************************
62              Xr_0, Xi_0   : in sfixed(0 downto -23); -- X_0
63              Xr_1, Xi_1   : in sfixed(0 downto -23); -- X_1
64              Xr_2, Xi_2   : in sfixed(0 downto -23); -- X_2
65              Xr_3, Xi_3   : in sfixed(0 downto -23); -- X_3
66              Xr_4, Xi_4   : in sfixed(0 downto -23); -- X_4
67              Xr_5, Xi_5   : in sfixed(0 downto -23); -- X_5
68              Xr_6, Xi_6   : in sfixed(0 downto -23); -- X_6
69              Xr_7, Xi_7   : in sfixed(0 downto -23); -- X_7
70              Xr_8, Xi_8   : in sfixed(0 downto -23); -- X_8
71              Xr_9, Xi_9   : in sfixed(0 downto -23); -- X_9
72              Xr_10, Xi_10 : in sfixed(0 downto -23); -- X_10
73              Xr_11, Xi_11 : in sfixed(0 downto -23); -- X_11
74              Xr_12, Xi_12 : in sfixed(0 downto -23); -- X_12
75              Xr_13, Xi_13 : in sfixed(0 downto -23); -- X_13
76              Xr_14, Xi_14 : in sfixed(0 downto -23); -- X_14
77              Xr_15, Xi_15 : in sfixed(0 downto -23); -- X_15
78              --* porte di uscita *********************************************************
79              Yr_0, Yi_0   : out sfixed(0 downto -23); -- Y_0
80              Yr_1, Yi_1   : out sfixed(0 downto -23); -- Y_1
81              Yr_2, Yi_2   : out sfixed(0 downto -23); -- Y_2
82              Yr_3, Yi_3   : out sfixed(0 downto -23); -- Y_3
83              Yr_4, Yi_4   : out sfixed(0 downto -23); -- Y_4
84              Yr_5, Yi_5   : out sfixed(0 downto -23); -- Y_5
85              Yr_6, Yi_6   : out sfixed(0 downto -23); -- Y_6
86              Yr_7, Yi_7   : out sfixed(0 downto -23); -- Y_7
87              Yr_8, Yi_8   : out sfixed(0 downto -23); -- Y_8
88              Yr_9, Yi_9   : out sfixed(0 downto -23); -- Y_9
89              Yr_10, Yi_10 : out sfixed(0 downto -23); -- Y_10
90              Yr_11, Yi_11 : out sfixed(0 downto -23); -- Y_11
91              Yr_12, Yi_12 : out sfixed(0 downto -23); -- Y_12
92              Yr_13, Yi_13 : out sfixed(0 downto -23); -- Y_13
93              Yr_14, Yi_14 : out sfixed(0 downto -23); -- Y_14
94              Yr_15, Yi_15 : out sfixed(0 downto -23)  -- Y_15
95          );
96      end component;
97
98  begin
99
100     -- istanza UUT
101     FFT_calculator : FFT_16
102     port map(
103         CK    => clock,
104         RST   => reset,
105         START => start,
106         DONE  => done,
107         Xr_0 => Xr_in_0, Xi_0 => Xi_in_0,
108         Xr_1 => Xr_in_1, Xi_1 => Xi_in_1,
109         Xr_2 => Xr_in_2, Xi_2 => Xi_in_2,
110         Xr_3 => Xr_in_3, Xi_3 => Xi_in_3,
111         Xr_4 => Xr_in_4, Xi_4 => Xi_in_4,
112         Xr_5 => Xr_in_5, Xi_5 => Xi_in_5,
113         Xr_6 => Xr_in_6, Xi_6 => Xi_in_6,
```

```
        Xr_7 => Xr_in_7, Xi_7 => Xi_in_7,
        Xr_8 => Xr_in_8, Xi_8 => Xi_in_8,
        Xr_9 => Xr_in_9, Xi_9 => Xi_in_9,
        Xr_10 => Xr_in_10, Xi_10 => Xi_in_10,
        Xr_11 => Xr_in_11, Xi_11 => Xi_in_11,
        Xr_12 => Xr_in_12, Xi_12 => Xi_in_12,
        Xr_13 => Xr_in_13, Xi_13 => Xi_in_13,
        Xr_14 => Xr_in_14, Xi_14 => Xi_in_14,
        Xr_15 => Xr_in_15, Xi_15 => Xi_in_15,
        Yr_0 => Yr_out_0, Yi_0 => Yi_out_0,
        Yr_1 => Yr_out_1, Yi_1 => Yi_out_1,
        Yr_2 => Yr_out_2, Yi_2 => Yi_out_2,
        Yr_3 => Yr_out_3, Yi_3 => Yi_out_3,
        Yr_4 => Yr_out_4, Yi_4 => Yi_out_4,
        Yr_5 => Yr_out_5, Yi_5 => Yi_out_5,
        Yr_6 => Yr_out_6, Yi_6 => Yi_out_6,
        Yr_7 => Yr_out_7, Yi_7 => Yi_out_7,
        Yr_8 => Yr_out_8, Yi_8 => Yi_out_8,
        Yr_9 => Yr_out_9, Yi_9 => Yi_out_9,
        Yr_10 => Yr_out_10, Yi_10 => Yi_out_10,
        Yr_11 => Yr_out_11, Yi_11 => Yi_out_11,
        Yr_12 => Yr_out_12, Yi_12 => Yi_out_12,
        Yr_13 => Yr_out_13, Yi_13 => Yi_out_13,
        Yr_14 => Yr_out_14, Yi_14 => Yi_out_14,
        Yr_15 => Yr_out_15, Yi_15 => Yi_out_15
    );

    -- process per la generazione del clock
    clk_process : process
    begin
        wait for 50 ns;
        clock <= not clock;
    end process clk_process;

    -- process per lettura da file e calcoli
    calc_process : process

        variable v_ILINE : line;      -- riga file input
        variable v_OLINE : line;      -- riga file output
        variable v_SPACE : character; -- carattere spazio

        variable v_Xr_in_0, v_Xi_in_0   : sfixed(0 downto -23);
        variable v_Xr_in_1, v_Xi_in_1   : sfixed(0 downto -23);
        variable v_Xr_in_2, v_Xi_in_2   : sfixed(0 downto -23);
        variable v_Xr_in_3, v_Xi_in_3   : sfixed(0 downto -23);
        variable v_Xr_in_4, v_Xi_in_4   : sfixed(0 downto -23);
        variable v_Xr_in_5, v_Xi_in_5   : sfixed(0 downto -23);
        variable v_Xr_in_6, v_Xi_in_6   : sfixed(0 downto -23);
        variable v_Xr_in_7, v_Xi_in_7   : sfixed(0 downto -23);
        variable v_Xr_in_8, v_Xi_in_8   : sfixed(0 downto -23);
        variable v_Xr_in_9, v_Xi_in_9   : sfixed(0 downto -23);
        variable v_Xr_in_10, v_Xi_in_10 : sfixed(0 downto -23);
        variable v_Xr_in_11, v_Xi_in_11 : sfixed(0 downto -23);
        variable v_Xr_in_12, v_Xi_in_12 : sfixed(0 downto -23);
        variable v_Xr_in_13, v_Xi_in_13 : sfixed(0 downto -23);
        variable v_Xr_in_14, v_Xi_in_14 : sfixed(0 downto -23);
        variable v_Xr_in_15, v_Xi_in_15 : sfixed(0 downto -23);

    begin
```

```
173         -- Apro file di I/O
174         file_open(file_INPUT, "input_data_fft.txt", read_mode);
175 
176         -- Reset macchina
177         reset <= '1';
178         wait for 2 ns;
179         reset <= '0';
180 
181         while not endfile(file_INPUT) loop
182 
183             -- Leggo da file di input
184             readline(file_INPUT, v_ILINE);
185             read(v_ILINE, v_Xr_in_0); -- get first input
186             read(v_ILINE, v_SPACE);   -- read in the space character
187             read(v_ILINE, v_Xi_in_0); -- get second input
188 
189             readline(file_INPUT, v_ILINE);
190             read(v_ILINE, v_Xr_in_1); -- get first input
191             read(v_ILINE, v_SPACE);   -- read in the space character
192             read(v_ILINE, v_Xi_in_1); -- get second input
193 
194             readline(file_INPUT, v_ILINE);
195             read(v_ILINE, v_Xr_in_2); -- get first input
196             read(v_ILINE, v_SPACE);   -- read in the space character
197             read(v_ILINE, v_Xi_in_2); -- get second input
198 
199             readline(file_INPUT, v_ILINE);
200             read(v_ILINE, v_Xr_in_3); -- get first input
201             read(v_ILINE, v_SPACE);   -- read in the space character
202             read(v_ILINE, v_Xi_in_3); -- get second input
203 
204             readline(file_INPUT, v_ILINE);
205             read(v_ILINE, v_Xr_in_4); -- get first input
206             read(v_ILINE, v_SPACE);   -- read in the space character
207             read(v_ILINE, v_Xi_in_4); -- get second input
208 
209             readline(file_INPUT, v_ILINE);
210             read(v_ILINE, v_Xr_in_5); -- get first input
211             read(v_ILINE, v_SPACE);   -- read in the space character
212             read(v_ILINE, v_Xi_in_5); -- get second input
213 
214             readline(file_INPUT, v_ILINE);
215             read(v_ILINE, v_Xr_in_6); -- get first input
216             read(v_ILINE, v_SPACE);   -- read in the space character
217             read(v_ILINE, v_Xi_in_6); -- get second input
218 
219             readline(file_INPUT, v_ILINE);
220             read(v_ILINE, v_Xr_in_7); -- get first input
221             read(v_ILINE, v_SPACE);   -- read in the space character
222             read(v_ILINE, v_Xi_in_7); -- get second input
223 
224             readline(file_INPUT, v_ILINE);
225             read(v_ILINE, v_Xr_in_8); -- get first input
226             read(v_ILINE, v_SPACE);   -- read in the space character
227             read(v_ILINE, v_Xi_in_8); -- get second input
228 
229             readline(file_INPUT, v_ILINE);
230             read(v_ILINE, v_Xr_in_9); -- get first input
231             read(v_ILINE, v_SPACE);   -- read in the space character
```

```
232             read(v_ILINE, v_Xi_in_9); -- get second input
233 
234             readline(file_INPUT, v_ILINE);
235             read(v_ILINE, v_Xr_in_10); -- get first input
236             read(v_ILINE, v_SPACE);    -- read in the space character
237             read(v_ILINE, v_Xi_in_10); -- get second input
238 
239             readline(file_INPUT, v_ILINE);
240             read(v_ILINE, v_Xr_in_11); -- get first input
241             read(v_ILINE, v_SPACE);    -- read in the space character
242             read(v_ILINE, v_Xi_in_11); -- get second input
243 
244             readline(file_INPUT, v_ILINE);
245             read(v_ILINE, v_Xr_in_12); -- get first input
246             read(v_ILINE, v_SPACE);    -- read in the space character
247             read(v_ILINE, v_Xi_in_12); -- get second input
248 
249             readline(file_INPUT, v_ILINE);
250             read(v_ILINE, v_Xr_in_13); -- get first input
251             read(v_ILINE, v_SPACE);    -- read in the space character
252             read(v_ILINE, v_Xi_in_13); -- get second input
253 
254             readline(file_INPUT, v_ILINE);
255             read(v_ILINE, v_Xr_in_14); -- get first input
256             read(v_ILINE, v_SPACE);    -- read in the space character
257             read(v_ILINE, v_Xi_in_14); -- get second input
258 
259             readline(file_INPUT, v_ILINE);
260             read(v_ILINE, v_Xr_in_15); -- get first input
261             read(v_ILINE, v_SPACE);    -- read in the space character
262             read(v_ILINE, v_Xi_in_15); -- get second input
263 
264             -- Passo le variabili ai corrispondenti segnali per poterle usare nei
                calcoli
265             Xr_in_0  <= v_Xr_in_0;
266             Xi_in_0  <= v_Xi_in_0;
267             Xr_in_1  <= v_Xr_in_1;
268             Xi_in_1  <= v_Xi_in_1;
269             Xr_in_2  <= v_Xr_in_2;
270             Xi_in_2  <= v_Xi_in_2;
271             Xr_in_3  <= v_Xr_in_3;
272             Xi_in_3  <= v_Xi_in_3;
273             Xr_in_4  <= v_Xr_in_4;
274             Xi_in_4  <= v_Xi_in_4;
275             Xr_in_5  <= v_Xr_in_5;
276             Xi_in_5  <= v_Xi_in_5;
277             Xr_in_6  <= v_Xr_in_6;
278             Xi_in_6  <= v_Xi_in_6;
279             Xr_in_7  <= v_Xr_in_7;
280             Xi_in_7  <= v_Xi_in_7;
281             Xr_in_8  <= v_Xr_in_8;
282             Xi_in_8  <= v_Xi_in_8;
283             Xr_in_9  <= v_Xr_in_9;
284             Xi_in_9  <= v_Xi_in_9;
285             Xr_in_10 <= v_Xr_in_10;
286             Xi_in_10 <= v_Xi_in_10;
287             Xr_in_11 <= v_Xr_in_11;
288             Xi_in_11 <= v_Xi_in_11;
289             Xr_in_12 <= v_Xr_in_12;
```

```
290             Xi_in_12 <= v_Xi_in_12;
291             Xr_in_13 <= v_Xr_in_13;
292             Xi_in_13 <= v_Xi_in_13;
293             Xr_in_14 <= v_Xr_in_14;
294             Xi_in_14 <= v_Xi_in_14;
295             Xr_in_15 <= v_Xr_in_15;
296             Xi_in_15 <= v_Xi_in_15;
297 
298             -- Faccio partire il processore
299             wait for 100 ns;
300             start <= '1';
301             wait for 100 ns;
302             start <= '0';
303             wait for 7000 ns;
304 
305         end loop;
306 
307         -- Closing In/Out files
308         file_close(file_INPUT);
309 
310     end process;
311 
312 end behavioral;
```