POLITECNICO DI TORINO

Laurea Magistrale in Ingegneria Elettronica

Corso di Sistemi Digitali Integrati

Relazione Progetto San Silvestro

Professore

Maurizio Zamboni

Studenti

Cavagnetto Matteo:290149

Cicero Rosalia: 282502

Rizzello Stefano:288013

Sommario

Introduzione

L'esercitazione finale consiste nella progettazione di un'unità di elaborazione che esegua la FFT secondo l'algoritmo di Cooley Tukey.

La realizzazione della Butterfly, componente base e fondamentale, consiste nell'effettuare somme e moltiplicazioni su numeri complessi A, B e W^k basandosi sul seguente schema:

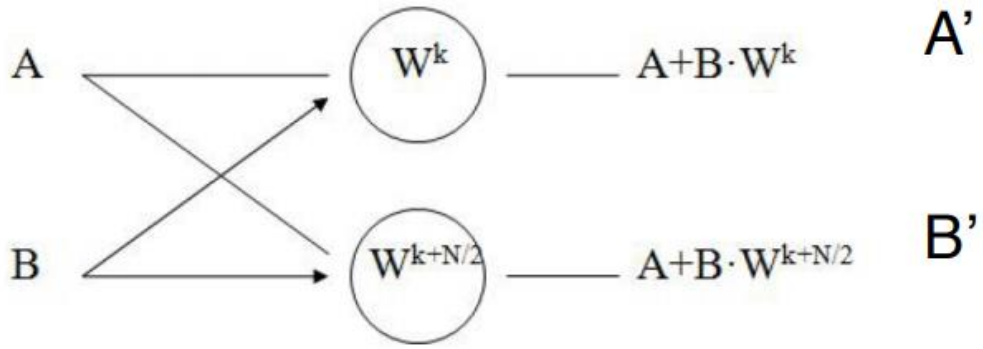


Figura 1: Struttura Butterfly

La Butterfly prenderà due valori complessi generici e ne produrrà delle uscite date da:

$A' = A + B\ x\ W^k$ e $B' = A - B\ x\ W^k$

che possono essere espresse esplicitando la parte reale e la parte immaginaria come:

$$A' = (A_R + B_R W_R - B_I W_I) + j(A_I + B_R W_I + B_I W_R)$$

$$B' = (A_R - B_R W_R + B_I W_I) + j(A_I - B_R W_I - B_I W_R)$$

I dati che riceve in ingresso saranno dati da:

$$A = A_R + jA_I$$

$$A' = A'_R + jA'_I$$

$$B = B_R + jB_I$$

$$B' = B_{,R} + jB'_I$$

Mentre i termini W^K differiscono di N/2 per cui saranno opposti.

Il risultato in uscita sarà dato quindi da dei risultati parziali, i quali identificano l'algoritmo da implementare secondo i seguenti passi:

$$M_1 = B_R W_R \qquad \Sigma_1 = A_R + M_1$$

$$M_2 = B_I W_I \qquad \Sigma_2 = \Sigma_1 - M_2 = A'r$$

$$M_3 = B_R W_I \qquad \Sigma_3 = \mathrm{A}_1 + M_3$$

$$M_4 = B_I W_R \qquad \Sigma_4 = \Sigma_3 + M_4 = A'r$$

$$M_5 = 2A_R \qquad \Sigma_5 = M_5 - \Sigma_2 = B'r$$

$$M_6 = 2A_I \qquad \Sigma_6 = M_6 - \Sigma_4 = B'r$$

Il progetto di tale struttura deve essere eseguito rispettando le seguenti specifiche:

- I dati A, B e W^k definiti in forma frazionaria (-1<dato<1) in complemento a 2 su 20 bit utilizzando una tecnica mista tra "Guard bit" e "Unconditional Block Floating Point Scaling".
- Si utilizzi un moltiplicatore con un livello di pipeline, in grado di effettuare la moltiplicazione tra due dati e la moltiplicazione di un dato per 2 tramite uno shift.
- Per le somme e le sottrazioni si ipotizzi di avere due blocchi sommatore/sottrattore in cui un segnale di controllo discrimina l'operazione da fare, e nello specifico anche il sommatore dovrà avere un livello di pipe.
- Si ipotizzi di avere le operazioni interne senza troncamento, ma che questo avvenga sul dato di uscita della Butterfly con la tecnica ROUND TO NEAREST EVEN, potendo usufruire di tutto l'hardware necessario per realizzare.
- L'unità di controllo deve essere definita mediante la tecnica della microprogrammazione con un sequencer con indirizzamento esplicito e tecnica del LATE STATUS.

Data Flow Diagram

Per la realizzazione del progetto sono state realizzate due Contro Unit: la prima CU_TOP, mostrata in Figura 2, atta a coordinare i 4 livelli di Butterfly, l'arrivo del segnale di start, la modalità in sequenza in cui può lavorare la macchina ed inoltre la fine dell'esecuzione; la seconda CU_BF, mostrata in Figura 3, si occupa invece della gestione dell'algoritmo della singola Butterfly. Quindi vengono implementate quattro CU_BF una per ogni livello di BF e una CU_TOP.

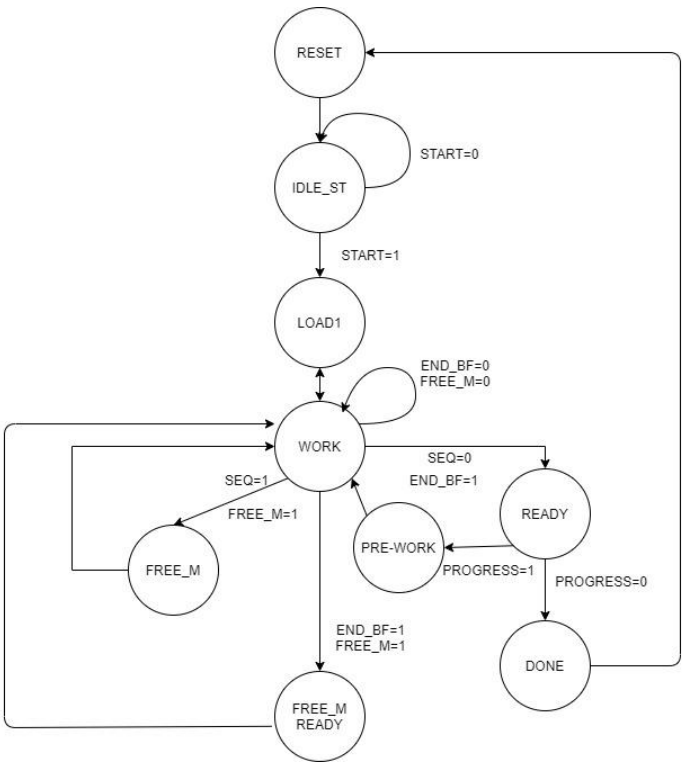


Figura 2: CU Top Level

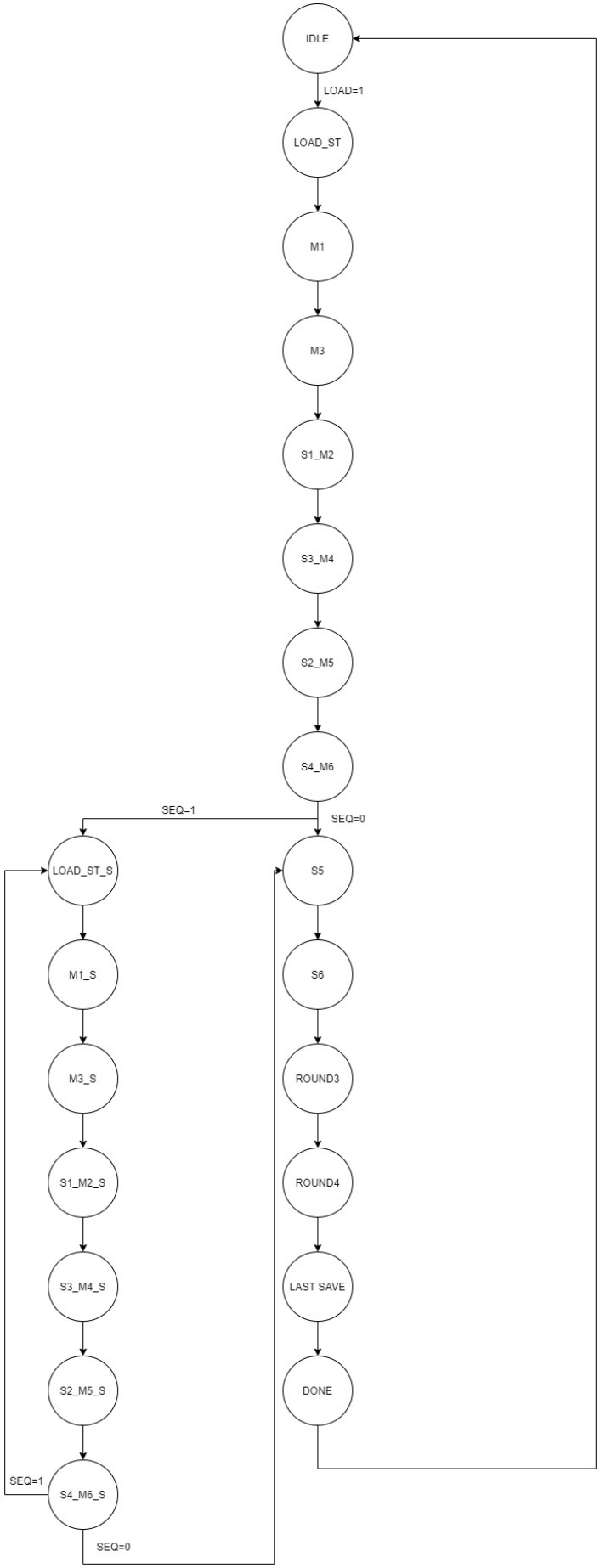


Figura 3: CU Butterfly

Nel dettaglio la CU_TOP quando evolve negli stati LOAD o FREE_M (caso per la sequenza) ha il compito di caricare i campioni all'interno delle BF di primo livello; a differenza, il caricamento dei dati tra gli altri livelli (secondo, terzo, quarto), avviene direttamente con segnali tra le CU_BF. In dettaglio il segnale END_BF del livello precedente viene usato per caricare all' interno del successivo i campioni appena elaborati. Il segnale END_BF del quarto livello viene invece letto dalla CU_TOP per andare nello stato di READY e comunica all' esterno che il dato è pronto in uscita alla fft16.

Nel caso si voglia fare delle elaborazioni in sequenza la CU_BF del primo livello giunta al sesto colpo di clock genera un segnale FREE_M per la CU_TOP, la quale, se ha anche il segnale di sequenza attivo (segnale SEQ), andrà nello stato FREE_M e provvederà a caricare nuovamente i dati nella BF di primo livello.

Quando si entra nella modalità in sequenza, la macchina eseguirà degli stati appositi al processamento di due campioni contemporaneamente. Questi sono gli stati con pedice 's', presenti nella Figura 3, nel ramo di sinistra.

Quindi, oltre a generare i segnali per l'elaborazione del nuovo dato, dovranno generare ancora i segnali per completare i calcoli del campione caricato in precedenza.

Es: M1_S dovrà generare i comandi di M1 per il nuovo dato e i comandi di S6 per il campione precedente.

I segnali attivi ad ogni stato sono rappresentati nelle tabelle.

Mentre i segnali di controllo vengono descritti successivamente.

Scheduling

Analizzate le specifiche di progetto, lo Scheduling individuato per realizzare l'algoritmo della Butterfly è il seguente:

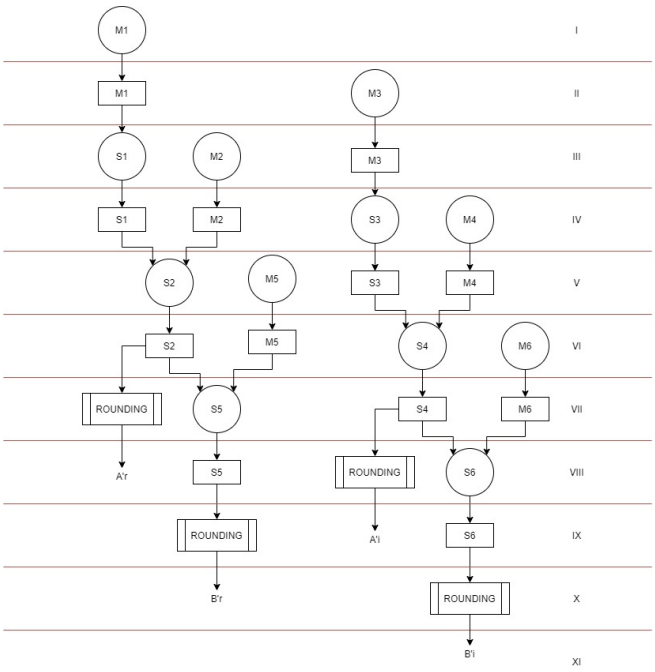


Figura 4: Scheduling

Come possibile osservare in Figura 4, al primo colpo di clock viene effettuata la moltiplicazione M1, il quale risultato come richiesto da specifica è disponibile al colpo di clock dopo ma, avendo inserito un registro di pipe tra il moltiplicatore e il sommatore, il risultato sarà disponibile per la somma al terzo colpo di clock. Lo stesso avviene per le altre moltiplicazioni.

Come è possibile osservare, non si verificheranno mai due moltiplicazioni nello stesso colpo di clock, rispettando la specifica del singolo moltiplicatore.

Il risultato della somma Σ_2 sarà una delle uscite della Butterfly, per cui avendo il risultato sul registro di pipe al VI colpo di clock, al colpo di clock successivo subirà l'arrotondamento e andrà in uscita.

Lo stesso avviene per i risultati in uscita dalla somma $\Sigma_4, \Sigma_5 \text{ ed } \Sigma_6$.

Lo Scheduling scelto è stato realizzato utilizzando la configurazione ALAP (As Late As Possible), infatti come è possibile osservare in Figura 4, si è scelto un ordine di realizzazione delle moltiplicazioni in cui dopo la prima moltiplicazione M1, viene realizzata la moltiplicazione M3, e successivamente le moltiplicazioni M2, M4, M5 ed M6, nei colpi di clock che precedono il loro impiego nelle rispettive somme. Questa soluzione è stata realizzata poiché si ha a disposizione un solo moltiplicatore, per cui si potrà avere al massimo una sola moltiplicazione per colpo di clock. Al sesto colpo di clock, il moltiplicatore sarà completamente libero e la Butterfly potrà accettare nuovi dati.

Per quanto riguarda le somme invece, pur avendo a disposizione due blocchi sommatore\sottrattore, si è scelto di utilizzarne solo uno, in cui attraverso due multiplexer si ha la possibilità di selezionare quale dato inviare al morsetto invertente per realizzare la sottrazione.

Tempo di vita delle variabili

Di seguito è riportato il diagramma del tempo di vita delle variabili.

	CK1	CK2	CK3	CK4	CK5	CK6	CK7	CK8	CK9	CK10	CK11
Ar											
Ai											
Br											
Bi											
Wr											
Wi											
M1											
M2											
M3											
M4											
M5											
M6											
Σ1											
Σ2								R	OUT		
Σ3											
Σ4									R	OUT	
Σ5										R	OUT
Σ6											R	OUT

Per la realizzazione di tale algoritmo all' interno della BF è stato scelto di salvare le variabili Ai, Ar, Bi, Br, Wi, Wr su un Register File a sei campi, mentre dal diagramma del tempo di vita delle variabili si è dedotto che sono necessari solamente un registro in cui salvare i risultati delle somme e un registro in cui salvare i risultati delle moltiplicazioni.

In tabella è stata evidenziata in rosso l'operazione di Rounding che viene eseguita sugli stati in uscita dalla Butterfly, mentre internamente si procede senza nessun troncamento. Come richiesto, viene realizzata la tecnica HALF UP ROUND TO NEAREST EVEN, che viene eseguita in un apposito colpo di clock.

Per la gestione dei valori di W_R W_I si è scelto di implementare al livello di top entity (FFT16) dei registri fissi denominati STONE, in cui vengono memorizzati solamente i valori positivi, dai quali tramite opportuna logica si è ricavato anche il valore negativo.

Data Path Butterfly

Dopo aver realizzato ed esaminato lo Scheduling e il tempo di vita delle variabili, si è proceduto con l'implementazione del progetto, andando a realizzare il Data Path dell'intera Butterfly mostrato in Figura 5.

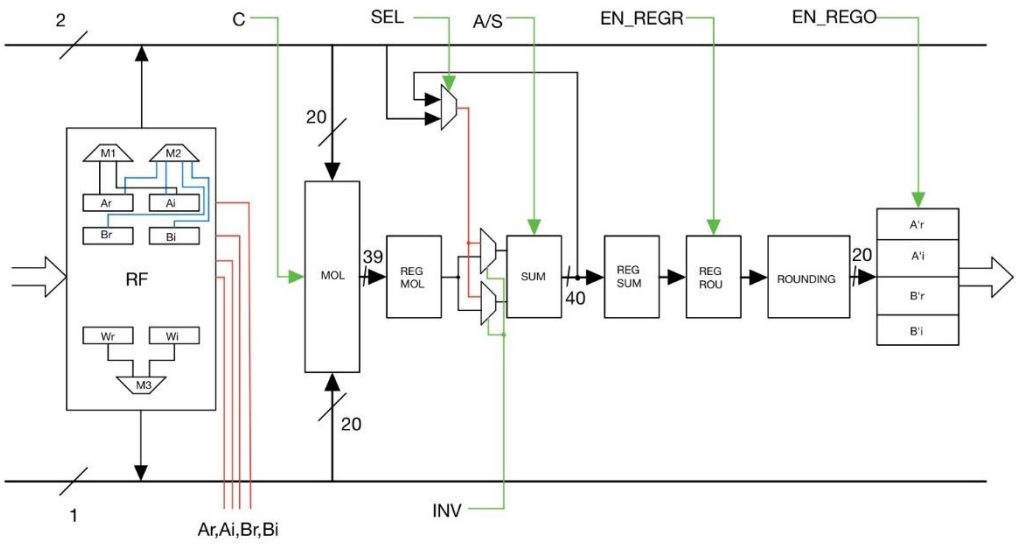


Figura 5: Data Path della singola BF

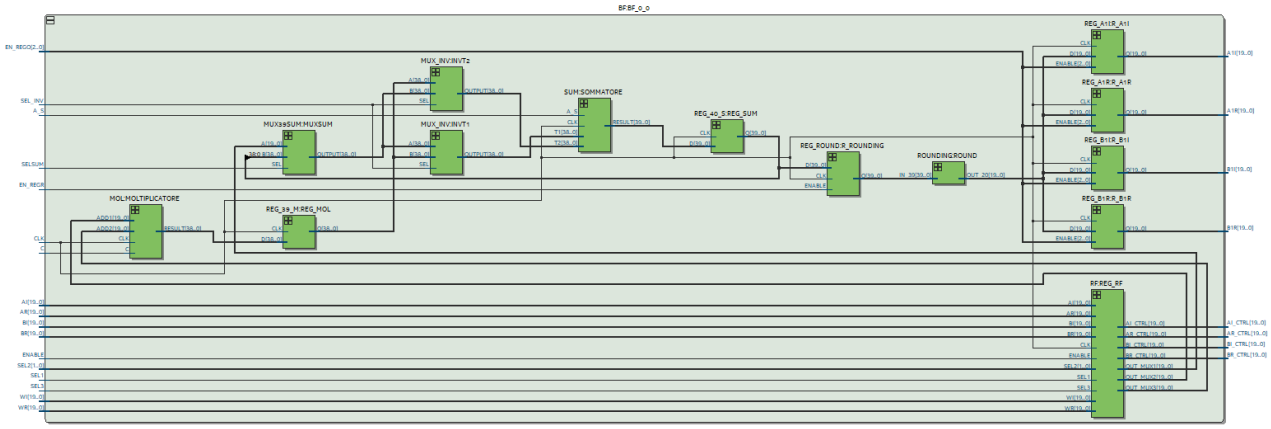


Figura 6: Datapath Butterfly

Per quanto riguarda l'ottimizzazione del numero dei BUS globali si è deciso di realizzare due BUS visibili nella parte superiore dello schema e uno nella parte inferiore, condiviso tramite MPX dai registri W_R e W_I. I BUS superiori vengono invece condivisi dai registri A_R, A_I, B_R, B_I il primo con un MPX a due vie e l'altro con un MPX a quattro vie.

Struttura dell'unità di controllo

Le figure 7 e 10 mostrano lo schema di base usato per la realizzazione delle Control Unit secondo la logica del LATE STATUS. Tale soluzione prevede di dividere le righe della memoria in più campi in

base al numero di salti richiesti dall' algoritmo (nel caso specifico 2 per CU_BF e 4 per CU_TOP) e di discriminare tali campi utilizzando i 2 bit o LSB meno significativi tramite multiplexer. Così facendo l'indirizzo di riga è noto a priori a prescindere dai salti e ciò permette di anticipare la procedura di lettura in memoria. I 2 bit meno significativi o LSB, che funzionano come selettore del MPX sono invece gestiti da un blocco di logica combinatoria che è funzione degli ingressi dalla CU.

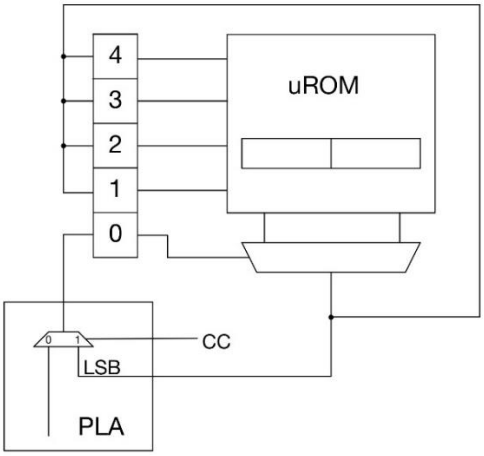


Figura 7: CU Butterfly

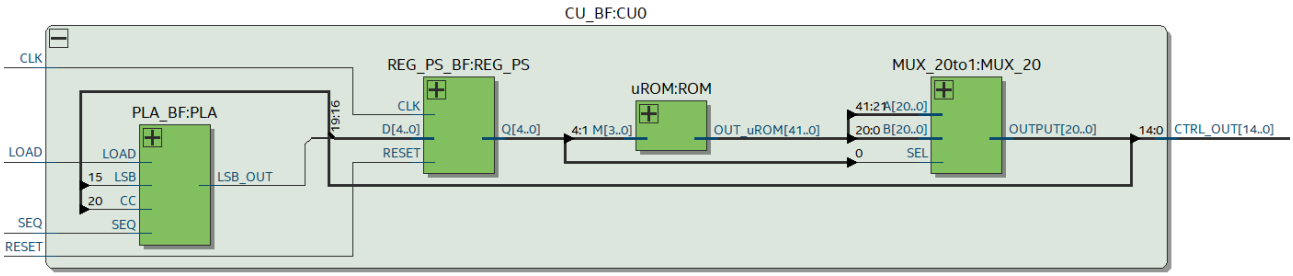


Figura 8:CU Butterfly

La Figura 8, presa dallo strumento netlist viewer di Quartus, mostra l'implementazione del blocco.

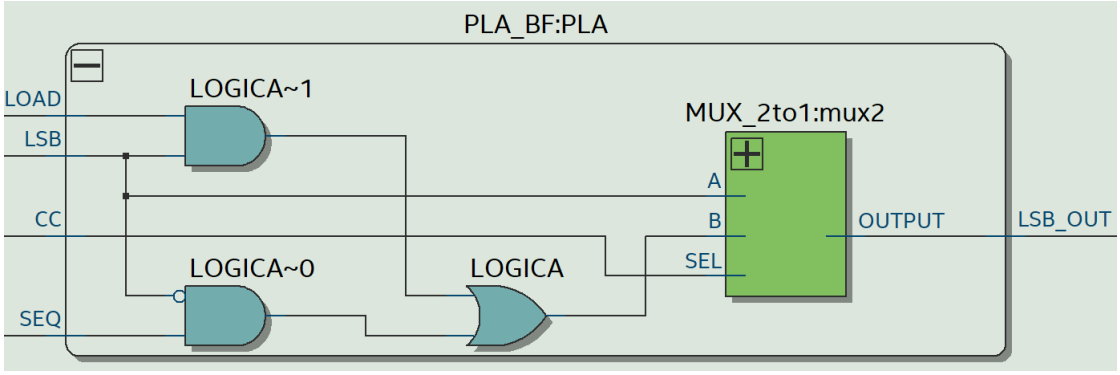


Figura 9: PLA Butterfly

L’immagine in Figura 9 mette in evidenza la logica interna del blocco PLA che è funzione degli ingressi.

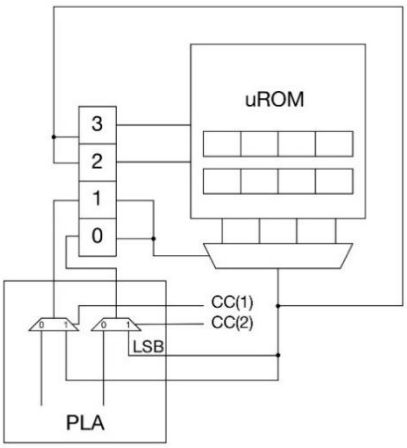


Figura 10: CU Top Level

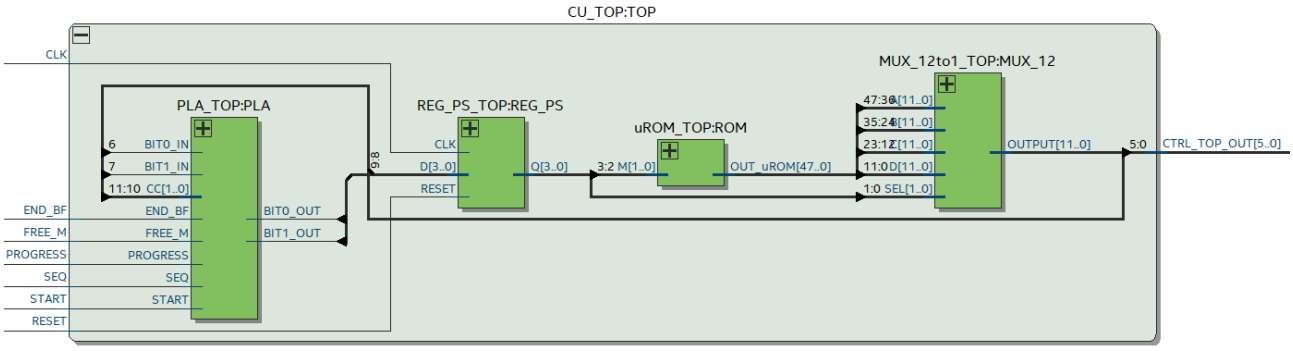


Figura 11:CU Top Level

La Figura 11 mostra l’implementazione del blocco su Quartus.

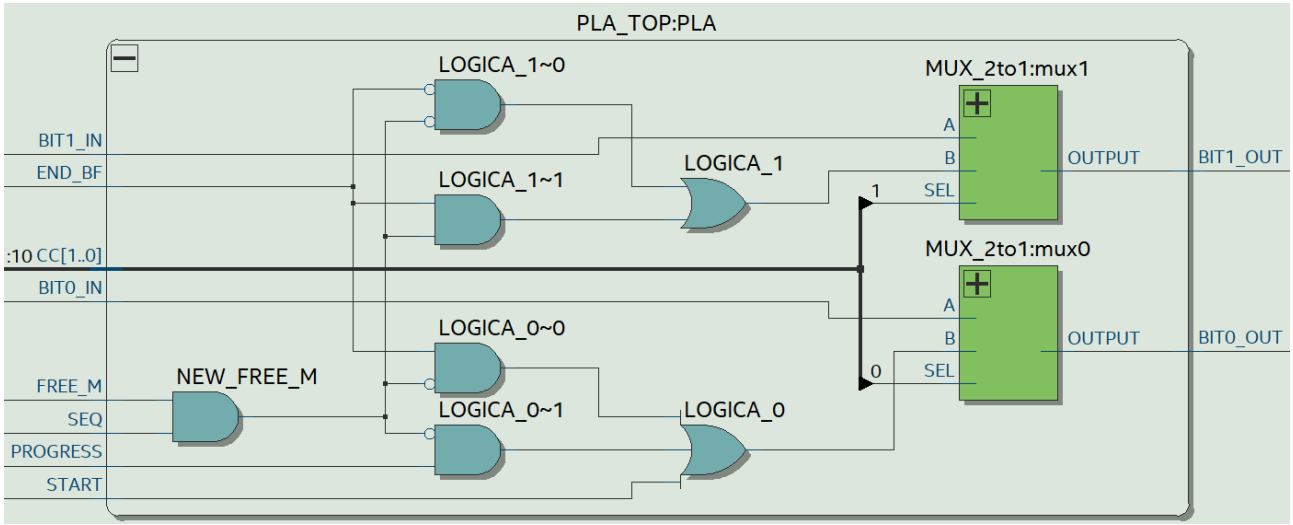


Figura 12: PLA Top Level

La Figura 12 indica la logica combinatoria funzione degli ingressi del blocco PLA della CU_TOP.

Tale logica è stata sintetizzata secondo le seguenti mappe:

NEW_FREE_M	END_BF	NEXT STATE	BIT 1
0	0	WORK	1
0	1	READY	0
1	0	FREE_M	0
1	1	FREE_M_READY	1

NEW_FREE_M	END_BF	START	PROGRESS	NEXT STATE	LSB(BIT 0)
0	0	0	0	DONE/IDLE	0
0	0	0	1	WORK	1
0	0	1	0	LOAD	1
0	0	1	1		
0	1	0	0	READY	1
0	1	0	1	READY	1
0	1	1	0		
0	1	1	1		
1	0	0	0		
1	0	0	1	FREE_M	0
1	0	1	0		
1	0	1	1		
1	1	0	0		
1	1	0	1	FREE_M_READY	0
1	1	1	0		
1	1	1	1		

Le righe evidenziate in Rosso indicano una combinazione degli ingressi invalida che non si potrà verificare in quanto come spiegato nel paragrafo dei controlli la macchina sarà sensibile al segnale di start solamente nello stato di IDLE e in tale stato non è possibile avere PROGRESS ad 1, che viene infatti settato nello stato di LOAD. Questo ha reso possibile una notevole semplificazione nella gestione della logica combinatoria della PLA. Inoltre, FREE_M e SEQ avendo significato solo quando sono entrambi ad 1, per ridurre il numero di variabili in ingresso, sono stati posti a prescindere in AND, in modo da gestire la Mappa con la sola variabile FREE_M. In dettaglio FREE_M indica che il livello di BF ha il moltiplicatore libero, ma solo se c'è attiva anche la richiesta di sequenza (SEQ) la CU_TOP dovrà provvedere a caricare i nuovi campioni al primo livello di BF.

La struttura delle uROM

La uROM della CU_TOP per ogni riga (0,1,2 la numero 3 non è utilizzata) presenta una configurazione a 4 campi per effettuare i salti a 4 vie che viene riportata in Figura 13 .

Indice di riga	2 LSB 00	2 LSB 01	2 LSB 10	2 LSB 11
00	IDLE	LOAD	XXXXX	RESET
01	FREEM	READY	FREEMR	WORK
10	XXXX	XXXX	DONE	PREWORK
11	XXXX	XXXX	XXXXX	XXXX

Figura 13: Struttura µROM 4 vie

In questo caso invece la uROM della CU_BF presenta solamente due campi pari e dispari in quanto sono previsti solamente salti a due vie nell'algoritmo.

Indice di riga	LSB 0	LSB 1
0000	IDLE	LOAD_ST
0001	M1	M3
0010	S1_M2	S3_M4
0011	S2_M5	S4_M6
0100	S5	LOAD_ST_S
0101	S6	ROUND3
0110	ROUND4	LAST_SAVE
0111	M1_S	M3_S
1000	S1_M2_S	S3_M4_S
1001	S2_M5_S	S4_M6_S
1010	DONE	XXXXXXX

Figura 14: Struttura della uROM CU_BF

In tabella, Figura 15, è mostrato lo schema organizzativo della riga della µROM della singola BF in cui è presente un bit per la condition code, un campo per la codifica del next state e a seguire i segnali di controllo. Le X stanno ad indicare il valore Undefined implementato su Quartus.

PRESENT STATE	CC	NEXT STATE	SEL INV	SEL SUM	SEL 1	SEL 2	SEL 3	C	A/S	EN REGS	EN REGR	EN REGO	FREE_M	END_BF
IDLE	1	00000	0	X	X	X	X	X	X	0	0	000	0	0
LOAD_ST	0	00010	0	X	X	X	X	X	X	0	0	000	0	0
M1	0	00011	0	X	X	10	0	1	X	0	0	000	0	0
M3	0	00100	0	X	X	10	1	1	X	0	0	000	0	0
S1_M2	0	00101	0	0	0	11	1	1	0	0	0	000	0	0
S3_M4	0	00110	0	0	1	11	0	1	0	0	0	000	0	0
S2_M5	0	00111	1	1	X	00	X	0	1	0	0	000	0	0
S4_M6	1	01000	0	1	X	01	X	0	0	0	1	000	1	0
S5	0	01001	1	1	X	X	X	X	1	0	1	001	0	0
LOAD_ST_S	0	01110	1	1	X	X	X	X	1	0	1	001	0	0
S6	0	01011	1	1	X	X	X	X	1	0	1	010	0	0
ROUND3	0	01100	0	X	X	X	X	X	X	0	1	011	0	0
ROUND4	0	01101	0	X	X	X	X	X	X	0	0	100	0	0
LAST SAVE	0	10100	0	X	X	X	X	X	X	1	0	100	0	0
DONE	0	00000	0	X	X	X	X	X	X	0	0	000	0	1
M1_S	0	01111	1	1	X	10	0	1	1	0	1	010	0	0
M3_S	0	10000	0	X	X	10	1	1	X	0	1	011	0	0
SI_M2_S	0	10001	0	0	0	11	1	1	0	0	0	100	0	0
S3_M4_S	0	10010	0	0	1	11	0	1	0	0	0	000	0	0
S2_M5_S	0	10010	0	1	X	00	X	0	1	0	0	000	0	1
S4_M6_S	1	01000	0	1	X	01	X	0	0	0	1	000	1	0

Figura 15: µROM singola Butterfly

Schema organizzativo µROM del componente CU_TOP:

PRESENT STATE	CC	NEXT STATE	LOAD	EN_FF	FF_VALUE	DONE	READY	RESET
RESET	00	0000	0	0	0	0	0	0
IDLE	01	0001	0	0	0	0	0	0
LOAD	00	0111	1	1	0	0	0	0
WORK	11	0100	0	0	1	0	0	0
FREE_M	00	0111	1	0	1	0	0	0
READY	01	1010	0	0	0	0	1	0
FREE_M_READY	00	0111	1	0	1	0	1	0
DONE	00	0011	0	0	0	1	0	1
PREWORK	00	0111	0	0	0	0	0	0

Figura 16: µROM CU Top Level

Criteri da rispettare

Per far sì che la macchina funzioni correttamente, bisogna rispettare dei tempi specifici per inviare i nuovi dati.

Affinché ogni segnale venga letto correttamente senza essere sovrascritto da un nuovo segnale in arrivo e affinché si possa garantire il corretto funzionamento in sequenza, è necessario che intercorrano 7 colpi di clock tra un dato e il successivo.

Se più campioni vengono inviati durante questo intervallo, infatti, l'ultimo sovrascriverà i precedenti. Mentre se un campione viene inviato oltre i 7 colpi di clock, la macchina non lo campionerà in tempo e non avverrà la lettura in sequenza. Questo perché i dati devono cambiare prima che le BF di primo livello abbiano il moltiplicatore libero (segnale FREE_M) in modo che la logica di controllo riconosca la richiesta di sequenza; dopo di che i dati devono rimanere validi e costanti per essere campionati quando la CU_TOP giunge nello stato di FREE_M.

Scelte realizzative

Nella realizzazione del progetto si è scelto di implementare all'ingresso della FFT_16 un blocco atto a ridurre la dinamica di ingresso tra -0.5 e 0.5 esclusi in modo da permettere all' utente di inviare campioni compresi tra -1 e 1. In uscita, invece, il dato deve esser moltiplicato per un fattore 32 in quanto anche al quarto livello in uscita dalle BF è prevista un'operazione di adattamento della dinamica per la BF successiva.

Segnali di controllo

Per quanto riguarda la gestione dei controlli per il segnale di START si è usata una porta OR su tutti i bit di tutti i campioni di ingresso, secondo il presupposto che l'assenza di campioni da processare sia data da tutti e quanti i campioni a 0. Inoltre, tramite una porta AND tra il segnale di start generato dalla logica combinatoria OR e un controllo SENSE, si è scelto di rendere la macchina sensibile al segnale di start solamente quando si trova nello stato di IDLE. In questo modo il segnale di start, che viene inviato alla PLA della CU_TOP una volta attivata la macchina, rimane fisso a zero. Ciò ha permesso una notevole semplificazione nella gestione della logica combinatoria per i salti a 4 vie.

Il segnale di funzionamento in sequenza è realizzato con una logica XOR tra i campioni posti in ingresso e quelli precedentemente caricati nelle BF di primo livello (questo è reso possibile usando delle porte aggiuntive delle BF denominate CTRL sui dati AR, AI, BR, BI). Ciò permette di capire se agli ingressi sono stati inviati dei nuovi dati. Inoltre, tramite una AND con il segnale di start si verifica anche che i campioni non siano tutti a zero, condizione scelta per indicare l'assenza di campioni e l'evoluzione della macchina nello stato di IDLE al completamento dell'operazione in corso.

Per la gestione del segnale di sequenza è stato inserito un componente basato su dei FF, uno per ognuna delle quattro CU_BF, dotati di segnali di set e reset. In questo modo è possibile settarli tutti e quattro ad 1 nel caso in cui ci sia l'esecuzione in sequenza, ma di rimuovere il segnale singolarmente per ogni CU_BF in modo che le BF di livello successivo possano portare a termine le operazioni rimaste in coda nei livelli di BF precedenti. In dettaglio il segnale di sequenza per il livello successivo viene azzerato solamente quando il livello di BF precedente ha terminato tutte le sue esecuzioni. Ciò è reso possibile mettendo in relazione i FF di questo componente con quelli descritti nel paragrafo successivo che generano il segnale di PROGRESS.

Per il controllo dei processi in atto nelle BF si è studiato un componente basato nuovamente su dei FF con set e reset, uno per ogni CU_BF, che vengono posti tutti ad 1 quando si inizia l'elaborazione di nuovi campioni, mentre quando un livello di BF termina l'esecuzione imposta a 0 il suo relativo FF. Le uscite dei quattro FF poste in OR permettono di generare il segnale di PROGRESS atto a

discriminare l'evoluzione della CU_TOP nello stato di DONE piuttosto che nello stato di PRE WORK, ovvero il caso in cui ci siano delle elaborazioni in coda qualora la macchina funzioni in sequenza.

In uscita è previsto un segnale di DONE per indicare che la macchina ha terminato completamente la sua esecuzione e un segnale di READY che avvisi che i dati in uscita sono validi

Es. Un'elaborazione in sequenza con 7 campioni in ingresso avrà in uscita 7 impulsi ad 1 del segnale di ready e un impulso ad 1 del segnale di DONE a termine operazione.

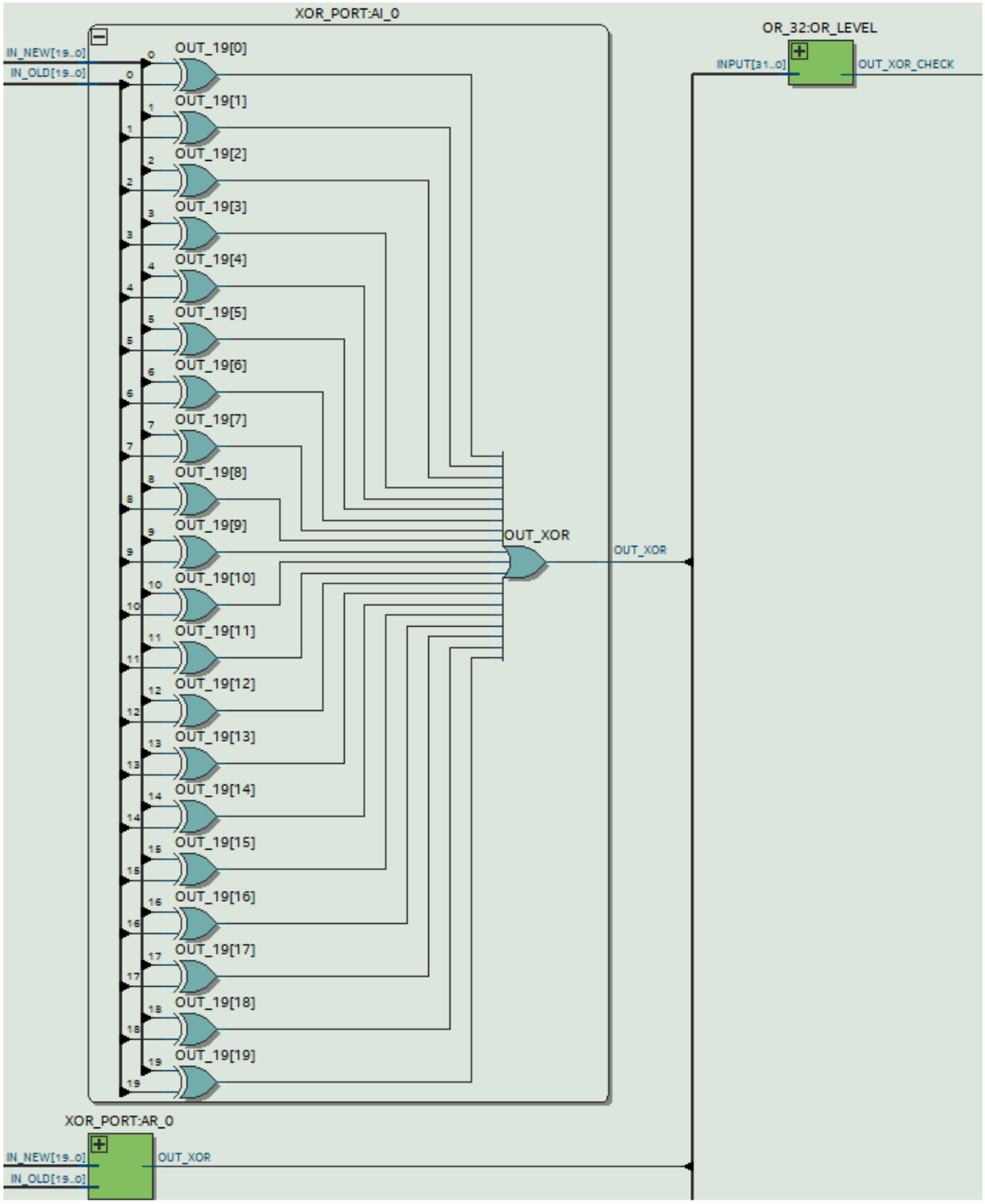


Figura 17: Sezione logica XOR per la sequenza

Codifica CU TOP

CODIFICA	STATO
RESET	0011
IDLE	0000
LOAD	0001
WORK	0111
FREE_M	0100
READY	0101
FREE_M_READY	0110
DONE	1010
PREWORK	1011

Codifica CU BF

CODIFICA	STATO
IDLE	00000
LAOD_ST	00001
M1	00010
M3	00011
S1_M2	00100
S3_M4	00101
S2_M5	00110
S4_M6	00111
S5	01000
LOAD_ST_S	01001
S6	01010
ROUND3	01011
ROUND4	01100
LAST SAVE	01101
DONE	10100
M1_S	01110
M3_S	01111
S1_M2_S	10000
S3_M4_S	10001
S2_M5_S	10010
S4_M6_S	10011

Rounding

Di seguito vengono riportati i risultati del Rounding realizzato secondo la tecnica to nearest even.

/fft_16_tb/FFT/BF_3_0/ROUND/IN_39_S	1000000000000000000100000000000000000
/fft_16_tb/FFT/BF_3_0/ROUND/OUT_20_S	10000000000000000

/fft_16_tb/FFT/BF_3_0/ROUND/IN_39_S	1111111111111111111110000000000000000
/fft_16_tb/FFT/BF_3_0/ROUND/OUT_20_S	00000000000000000

/fft_16_tb/FFT/BF_3_2/ROUND/IN_39_S	1111111111111111111100000000000000000
/fft_16_tb/FFT/BF_3_2/ROUND/OUT_20_S	11111111111111111

/fft_16_tb/FFT/BF_3_0/ROUND/IN_39_S	0000000000000000000111111111111111110
/fft_16_tb/FFT/BF_3_0/ROUND/OUT_20_S	00000000000000001

Timing

Timing esecuzione continua

Nelle figure sottostanti sono riportati in sequenza tutti gli stati che vengono svolti dalle varie control unit nel caso di lettura in sequenza.

Figura 18

In particolare, la Figura 18 mostra il passaggio dallo stato di IDLE a quello di LOAD ed il caricamento dei dati nel primo livello di BF. Essendoci un altro dato da caricare in sequenza, la BF va nello stato LOAD_ST_S e quando viene terminato il primo campione inizia a fare le sue operazioni anche la BF del secondo livello.

Figura 19

In Figura 19 si può notare che le BF del terzo e del quarto livello si attivano.

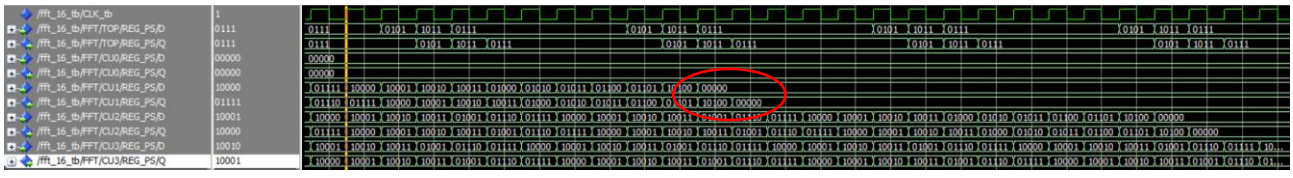


Figura 20

In Figura 20, invece, è possibile notare che le BF del primo e del secondo livello, terminati i campioni da processare, vanno nello stato di IDLE mentre quelle dei livelli successivi continuano a processare i dati rimasti.

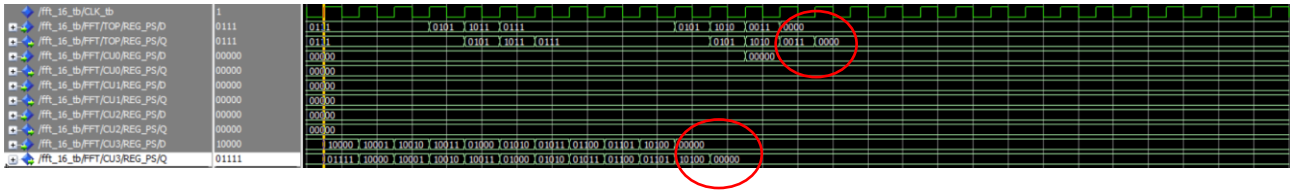


Figura 21

Infine, in Figura 21 si osserva che, terminata anche l'elaborazione dell'ultimo livello di BF la CU_TOP passa nello stato di DONE, reimposta i parametri iniziali nello stato di RESET e termina in IDLE pronta a leggere un nuovo segnale di START.

Timing esecuzione singola

Nelle seguenti figure, sono riportati in sequenza tutti gli stati che vengono svolti dalle varie control unit nel caso di lettura singola.

A differenza del caso continuo, le singole BF eseguono da LOAD a DONE l'elaborazione di un solo campione, senza passare dagli stati di sequenza.

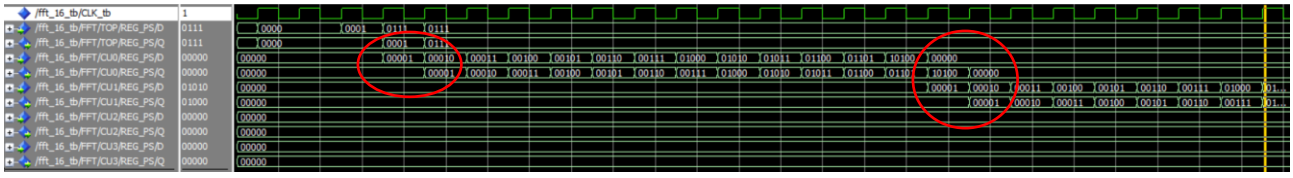


Figura 22

Inoltre, quando il livello precedente termina l'elaborazione quello successivo inizia l'elaborazione dei dati.

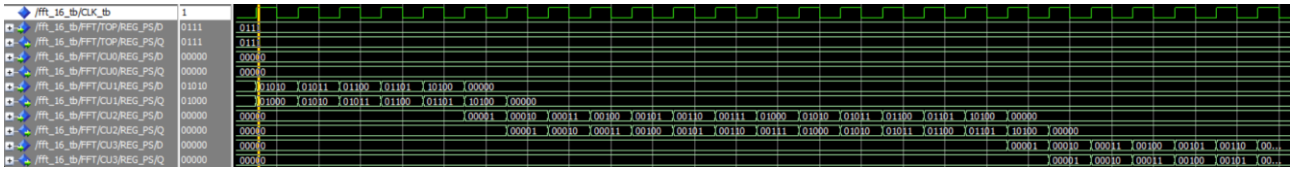
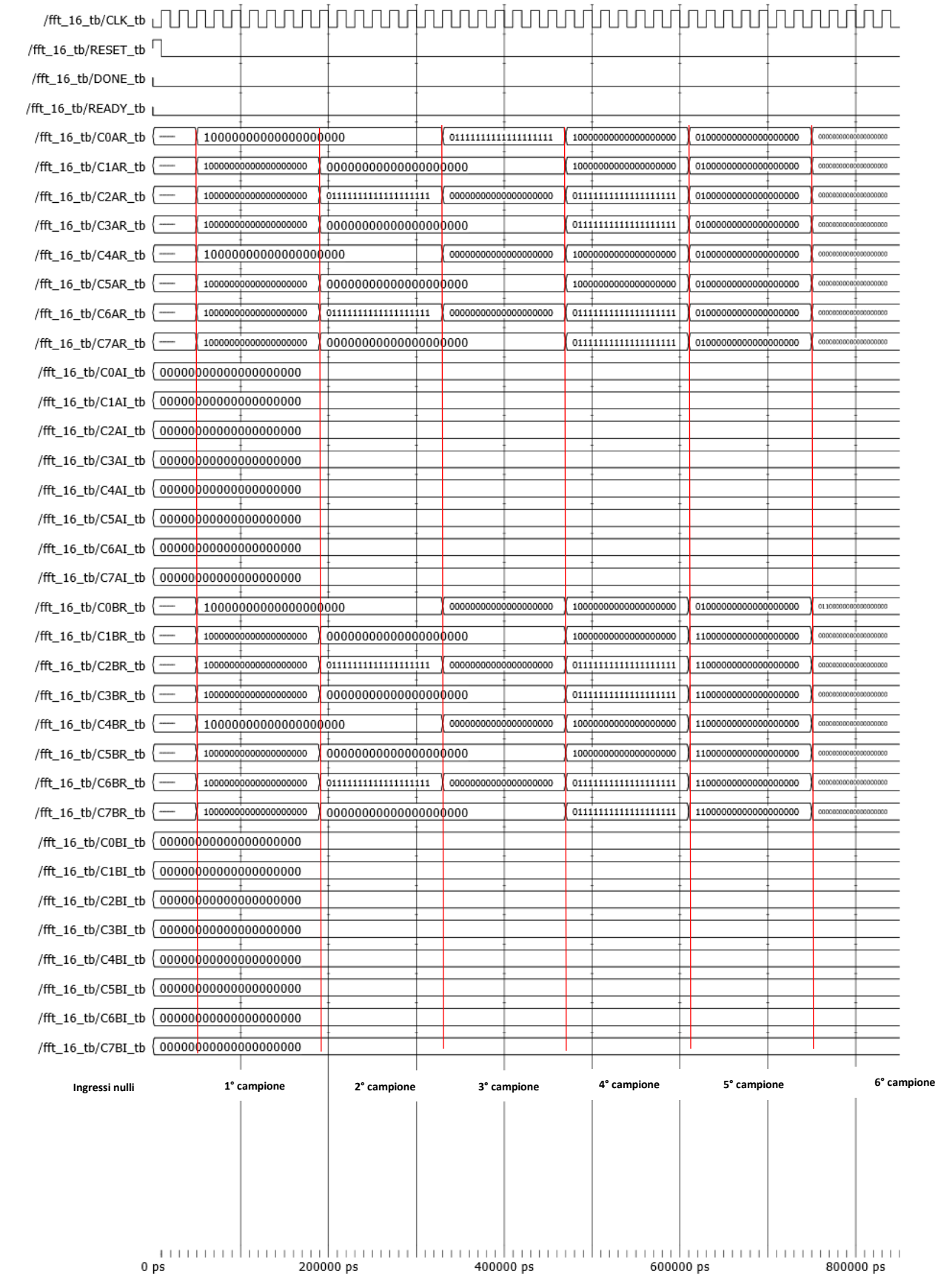


Figura 23

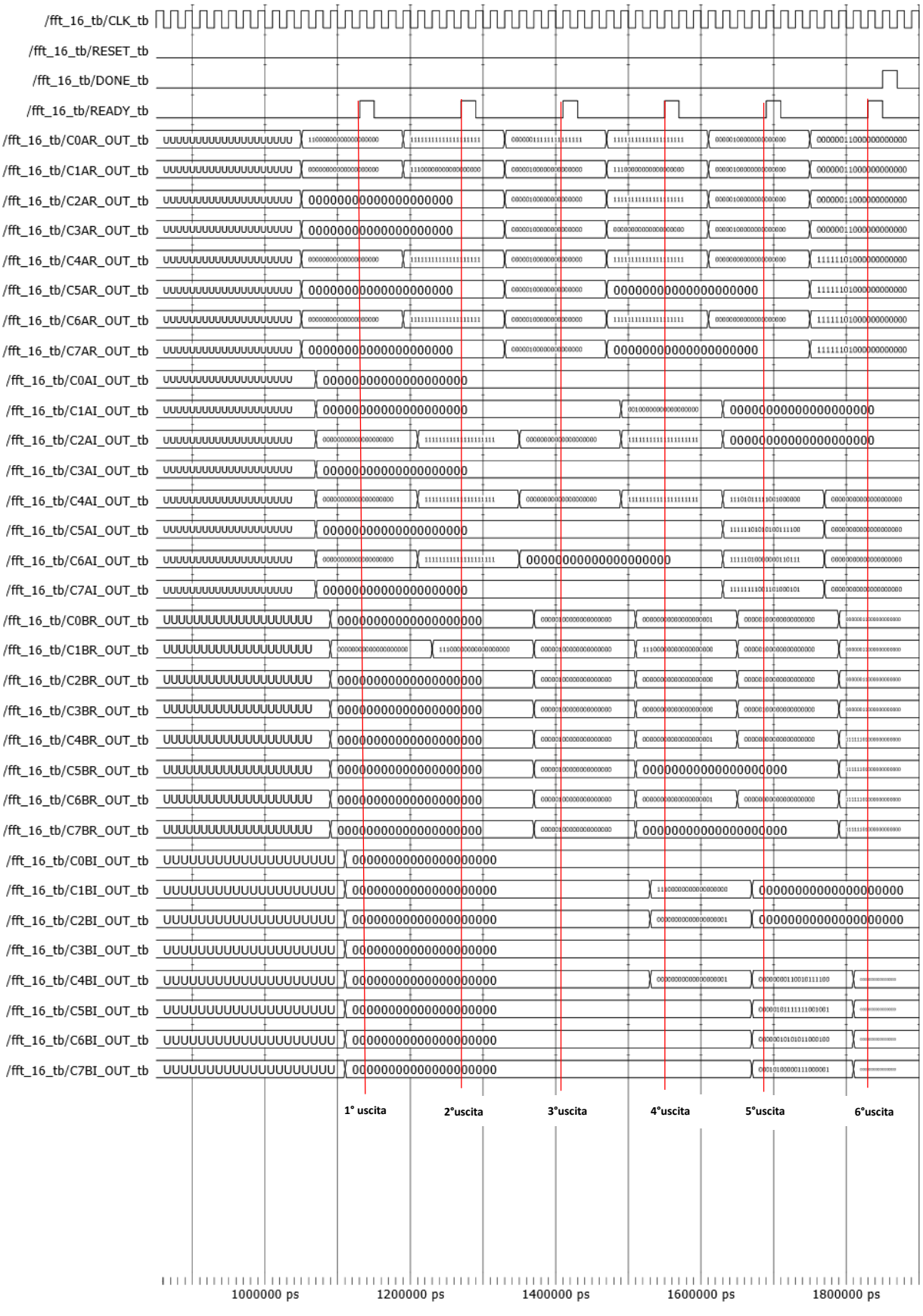
Figura 24

In Figura 24, terminata l'operazione sul quarto ed ultimo livello di BF, la CU_TOP transisce da WORK a READY e poi in DONE.

Timing relativo agli ingressi



Timing relativo alle uscite



Script MATLAB per testare la FFT per ogni singolo campione

```
clc;
clear all;

campione = "inserire numero del campione da voler testare";

if campione == 1
AR = [-1,-1,-1,-1,-1,-1,-1,-1];
    elseif campione == 2
AR = [-1,0,1,0,-1,0,1,0];
    elseif campione == 3
AR = [1,0,0,0,0,0,0,0];
    elseif campione == 4
AR = [-1,-1,1,1,-1,-1,1,1];
    elseif campione == 5
AR = [0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5];
    elseif campione == 6
AR = [0,0,0,0,0,0,0,0];
end
AI = [0,0,0,0,0,0,0,0];

if campione == 1
BR = [-1,-1,-1,-1,-1,-1,-1,-1];
    elseif campione == 2
BR = [-1,0,1,0,-1,0,1,0];
    elseif campione == 3
BR = [0,0,0,0,0,0,0,0];
    elseif campione == 4
BR = [-1,-1,1,1,-1,-1,1,1];
    elseif campione == 5
BR = [0.5,-0.5,-0.5,-0.5,-0.5,-0.5,-0.5,-0.5];
    elseif campione == 6
BR = [0.75,0,0,0,0,0,0,0];
end
BI = [0,0,0,0,0,0,0,0];

A1R= [0,0,0,0,0,0,0,0];          A1I= [0,0,0,0,0,0,0,0];
B1R= [0,0,0,0,0,0,0,0];          B1I= [0,0,0,0,0,0,0,0];
A1R_temp= [0,0,0,0,0,0,0,0];     A1I_temp= [0,0,0,0,0,0,0,0];
B1R_temp= [0,0,0,0,0,0,0,0];     B1I_temp= [0,0,0,0,0,0,0,0];

WR = [1,0.92387953251129,0.70710678118655,0.38268343236509,0,-0.38268343236509,-
0.70710678118655,-0.92387953251129];
WI = [0,-0.38268343236509,-0.70710678118655,-0.92387953251129,-1,-
0.92387953251129,-0.70710678118655,-0.38268343236509];

%% bf0
for i=1:8
M1 = BR(i)*WR(1);    M2 = BI(i)*WI(1);    M3 = BR(i)*WI(1);    M4 = BI(i)*WR(1);
M5 = 2*AR(i);
M6 = 2*AI(i);        S1 = AR(i)+M1;       S2 = S1-M2;          S3 = AI(i)+M3;
S4 = S3+M4;
S5 = M5-S2;          S6 = M6-S4;

A1R_temp(i) = S2;    A1I_temp(i) = S4;    B1R_temp(i) = S5;    B1I_temp(i) = S6;
end
A1R = A1R_temp;      A1I = A1I_temp;      B1R = B1R_temp;      B1I = B1I_temp;

%% bf1
for i=1:8
if i > 4
```

```
    M1 = B1R(i)*WR(5);  M2 = B1I(i)*WI(5);  M3 = B1R(i)*WI(5);  M4 =
B1I(i)*WR(5);  M5 = 2*B1R(i-4);
    M6 = 2*B1I(i-4);    S1 = B1R(i-4)+M1;   S2 = S1-M2;         S3 = B1I(i-
4)+M3;   S4 = S3+M4;
    S5 = M5-S2;         S6 = M6-S4;
else
    M1 = A1R(i+4)*WR(1);M2 = A1I(i+4)*WI(1);M3 = A1R(i+4)*WI(1);M4 =
A1I(i+4)*WR(1);M5 = 2*A1R(i);
    M6 = 2*A1I(i);      S1 = A1R(i)+M1;     S2 = S1-M2;         S3 = A1I(i)+M3;
S4 = S3+M4;
    S5 = M5-S2;         S6 = M6-S4;
end

A1R_temp(i) = S2;   A1I_temp(i) = S4;   B1R_temp(i) = S5;   B1I_temp(i) = S6;
end
A1R = A1R_temp;     A1I = A1I_temp;     B1R = B1R_temp;     B1I = B1I_temp;

%% bf2
for i=1:8
if i > 6
    M1 = B1R(i)*WR(7);  M2 = B1I(i)*WI(7);  M3 = B1R(i)*WI(7);  M4 =
B1I(i)*WR(7);  M5 = 2*B1R(i-2);
    M6 = 2*B1I(i-2);    S1 = B1R(i-2)+M1;   S2 = S1-M2;         S3 = B1I(i-
2)+M3;   S4 = S3+M4;
    S5 = M5-S2;         S6 = M6-S4;
elseif i > 4
    M1 = A1R(i+2)*WR(3);M2 = A1I(i+2)*WI(3);M3 = A1R(i+2)*WI(3);M4 =
A1I(i+2)*WR(3);M5 = 2*A1R(i);
    M6 = 2*A1I(i);      S1 = A1R(i)+M1;     S2 = S1-M2;         S3 = A1I(i)+M3;
S4 = S3+M4;
    S5 = M5-S2;         S6 = M6-S4;
elseif i > 2
    M1 = B1R(i)*WR(5);  M2 = B1I(i)*WI(5);  M3 = B1R(i)*WI(5);  M4 =
B1I(i)*WR(5)   ;M5 = 2*B1R(i-2);
    M6 = 2*B1I(i-2);    S1 = B1R(i-2)+M1;   S2 = S1-M2;         S3 = B1I(i-
2)+M3;   S4 = S3+M4;
    S5 = M5-S2;         S6 = M6-S4;
else
    M1 = A1R(i+2)*WR(1);M2 = A1I(i+2)*WI(1);M3 = A1R(i+2)*WI(1);M4 =
A1I(i+2)*WR(1);M5 = 2*A1R(i);
    M6 = 2*A1I(i);      S1 = A1R(i)+M1;     S2 = S1-M2;         S3 = A1I(i)+M3;
S4 = S3+M4;
    S5 = M5-S2;         S6 = M6-S4;
end

A1R_temp(i) = S2;   A1I_temp(i) = S4;   B1R_temp(i) = S5;   B1I_temp(i) = S6;
end
A1R = A1R_temp;     A1I = A1I_temp;     B1R = B1R_temp;     B1I = B1I_temp;

%% bf3
for i=1:8
if i > 7
    M1 = B1R(i)*WR(8);  M2 = B1I(i)*WI(8);  M3 = B1R(i)*WI(8);  M4 =
B1I(i)*WR(8);  M5 = 2*B1R(i-1);
    M6 = 2*B1I(i-1);    S1 = B1R(i-1)+M1;   S2 = S1-M2;         S3 = B1I(i-
1)+M3;     S4 = S3+M4;
    S5 = M5-S2;         S6 = M6-S4;
elseif i > 6
    M1 = A1R(i+1)*WR(4);M2 = A1I(i+1)*WI(4);M3 = A1R(i+1)*WI(4);M4 =
A1I(i+1)*WR(4);M5 = 2*A1R(i);
    M6 = 2*A1I(i);      S1 = A1R(i)+M1;     S2 = S1-M2;         S3 = A1I(i)+M3;
S4 = S3+M4;
```

```
    S5 = M5-S2;          S6 = M6-S4;
elseif i > 5
    M1 = B1R(i)*WR(6);  M2 = B1I(i)*WI(6);  M3 = B1R(i)*WI(6);  M4 =
B1I(i)*WR(6);  M5 = 2*B1R(i-1);
    M6 = 2*B1I(i-1);    S1 = B1R(i-1)+M1;   S2 = S1-M2;         S3 = B1I(i-
1)+M3;     S4 = S3+M4;
    S5 = M5-S2;          S6 = M6-S4;
elseif i > 4
    M1 = A1R(i+1)*WR(2);M2 = A1I(i+1)*WI(2);M3 = A1R(i+1)*WI(2);M4 =
A1I(i+1)*WR(2);M5 = 2*A1R(i);
    M6 = 2*A1I(i);      S1 = A1R(i)+M1;     S2 = S1-M2;         S3 = A1I(i)+M3;
S4 = S3+M4;
    S5 = M5-S2;          S6 = M6-S4;
elseif i > 3
    M1 = B1R(i)*WR(7);  M2 = B1I(i)*WI(7);  M3 = B1R(i)*WI(7);  M4 =
B1I(i)*WR(7);  M5 = 2*B1R(i-1);
    M6 = 2*B1I(i-1);    S1 = B1R(i-1)+M1;   S2 = S1-M2;         S3 = B1I(i-
1)+M3;     S4 = S3+M4;
    S5 = M5-S2;          S6 = M6-S4;
elseif i > 2
    M1 = A1R(i+1)*WR(3);M2 = A1I(i+1)*WI(3);M3 = A1R(i+1)*WI(3);M4 =
A1I(i+1)*WR(3);M5 = 2*A1R(i);
    M6 = 2*A1I(i);      S1 = A1R(i)+M1;     S2 = S1-M2;         S3 = A1I(i)+M3;
S4 = S3+M4;
    S5 = M5-S2;          S6 = M6-S4;
elseif i > 1
    M1 = B1R(i)*WR(5);  M2 = B1I(i)*WI(5);  M3 = B1R(i)*WI(5);  M4 =
B1I(i)*WR(5);  M5 = 2*B1R(i-1);
    M6 = 2*B1I(i-1);    S1 = B1R(i-1)+M1;   S2 = S1-M2;         S3 = B1I(i)+M3;
S4 = S3+M4;
    S5 = M5-S2;          S6 = M6-S4;
else
    M1 = A1R(i+1)*WR(1);M2 = A1I(i+1)*WI(1);M3 = A1R(i+1)*WI(1);M4 =
A1I(i+1)*WR(1);M5 = 2*A1R(i);
    M6 = 2*A1I(i);      S1 = A1R(i)+M1;     S2 = S1-M2;         S3 = A1I(i)+M3;
S4 = S3+M4;
    S5 = M5-S2;          S6 = M6-S4;
end

A1R_temp(i) = S2;   A1I_temp(i) = S4;   B1R_temp(i) = S5;   B1I_temp(i) = S6;
end
A1R = A1R_temp;     A1I = A1I_temp;     B1R = B1R_temp;     B1I = B1I_temp;

A1R
A1I
B1R
B1I
```

CAMPIONE 1

INGRESSI:

```
AR = [-1,-1,-1,-1,-1,-1,-1,-1]
AI = [0,0,0,0,0,0,0,0]
BR = [-1,-1,-1,-1,-1,-1,-1,-1]
BR = [0,0,0,0,0,0,0,0]
```

USCITE:

```
A1R = [-16,0,0,0,0,0,0,0]
A1I = [0,0,0,0,0,0,0,0]
B1R = [0,0,0,0,0,0,0,0]
B1I = [0,0,0,0,0,0,0,0]
```

CAMPIONE 2

INGRESSI:

```
AR = [-1,0,1,0,-1,0,1,0]
AI = [0,0,0,0,0,0,0,0]
BR = [-1,0,1,0,-1,0,1,0]
BR = [0,0,0,0,0,0,0,0]
```

USCITE:

```
A1R = [0,-8,0,0,0,0,0,0]
A1I = [0,0,0,0,0,0,0,0]
B1R = [0,-8,0,0,0,0,0,0]
B1I = [0,0,0,0,0,0,0,0]
```

CAMPIONE 3

INGRESSI:

```
AR = [1,0,0,0,0,0,0,0]
AI = [0,0,0,0,0,0,0,0]
BR = [0,0,0,0,0,0,0,0]
BR = [0,0,0,0,0,0,0,0]
```

USCITE:

```
A1R = [1,1,1,1,1,1,1,1]
A1I = [0,0,0,0,0,0,0,0]
B1R = [1,1,1,1,1,1,1,1]
B1I = [0,0,0,0,0,0,0,0]
```

CAMPIONE 4

INGRESSI:

```
AR = [-1,-1,1,1,-1,-1,1,1]
AI = [0,0,0,0,0,0,0,0]
BR = [-1,-1,1,1,-1,-1,1,1]
BR = [0,0,0,0,0,0,0,0]
```

USCITE:

```
A1R = [0,-8,0,0,0,0,0,0]
A1I = [0,8,0,0,0,0,0,0]
B1R = [0,-8,0,0,0,0,0,0]
B1I = [0,-8,0,0,0,0,0,0]
```

CAMPIONE 5

INGRESSI:

```
AR = [0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5]
AI = [0,0,0,0,0,0,0,0]
BR = [0.5,-0.5,-0.5,-0.5,-0.5,-0.5,-0.5,-0.5]
BR = [0,0,0,0,0,0,0,0]
```

USCITE:

```
A1R = [1,1,1,1,6.661338147750939e-16,5.884182030513330e-15,-5.717648576819556e-15,-6.661338147750939e-16]
A1I = [0,0,0,0,-5.027339492125867,-6.681786379192990e-01,-1.496605762665499,-1.989123673796658e-01]
B1R = [1,1,1,1,-6.661338147750939e-16,-5.884182030513330e-15,5.717648576819556e-15,6.661338147750939e-16]
B1I = [0,0,0,0,1.989123673796662e-01,1.496605762665499,6.681786379192991e-01,5.027339492125867]
```

CAMPIONE 6

INGRESSI:

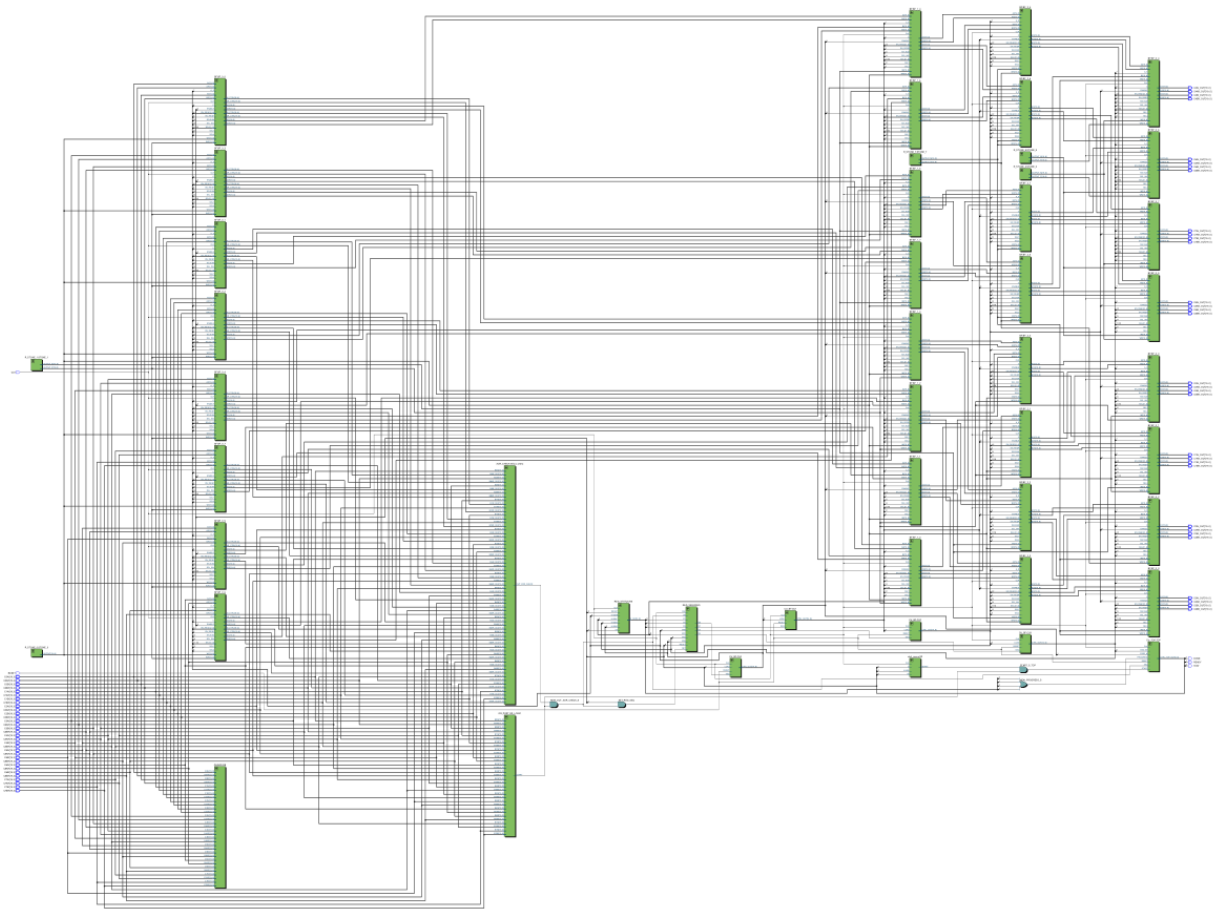
```
AR = [0,0,0,0,0,0,0,0]
AI = [0,0,0,0,0,0,0,0]
BR = [0.75,0,0,0,0,0,0,0]
BR = [0,0,0,0,0,0,0,0]
```

USCITE:

```
A1R = [0.75,0.75,0.75,0.75,-0.75,-0.75,-0.75,-0.75]
A1I = [0,0,0,0,0,0,0,0]
B1R = [0.75,0.75,0.75,0.75,-0.75,-0.75,-0.75,-0.75]
B1I = [0,0,0,0,0,0,0,0]
```

Confrontando i risultati ottenuti dallo script di Matlab e i valori in uscita dalla FFT, simulati con Modelsim, possono notarsi delle piccole incongruenze per quanto riguarda l'arrotondamento e l'errore macchina dovuto all'aritmetica finita della soluzione implementata. Nonostante ciò, i risultati ottenuti sono congruenti con quanto previsto teoricamente nella simulazione MATLAB.

Appendice



FFT_16_tb

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;


ENTITY FFT_16_tb IS
END FFT_16_tb;

ARCHITECTURE behav OF FFT_16_tb IS

SIGNAL CLK_tb,RESET_tb,DONE_tb,READY_tb:STD_LOGIC;
SIGNAL C0AR_tb,C1AR_tb,C2AR_tb,C3AR_tb,C4AR_tb,C5AR_tb,C6AR_tb,C7AR_tb:
SIGNED(19 downto 0);
SIGNAL C0AI_tb,C1AI_tb,C2AI_tb,C3AI_tb,C4AI_tb,C5AI_tb,C6AI_tb,C7AI_tb:
SIGNED(19 downto 0);
```

```
SIGNAL C0BR_tb,C1BR_tb,C2BR_tb,C3BR_tb,C4BR_tb,C5BR_tb,C6BR_tb,C7BR_tb:
SIGNED(19 downto 0);
SIGNAL C0BI_tb,C1BI_tb,C2BI_tb,C3BI_tb,C4BI_tb,C5BI_tb,C6BI_tb,C7BI_tb:
SIGNED(19 downto 0);

SIGNAL
C0AR_OUT_tb,C1AR_OUT_tb,C2AR_OUT_tb,C3AR_OUT_tb,C4AR_OUT_tb,C5AR_OUT_tb,C6AR_OUT
_tb,C7AR_OUT_tb: SIGNED(19 downto 0);
SIGNAL
C0AI_OUT_tb,C1AI_OUT_tb,C2AI_OUT_tb,C3AI_OUT_tb,C4AI_OUT_tb,C5AI_OUT_tb,C6AI_OUT
_tb,C7AI_OUT_tb: SIGNED(19 downto 0);
SIGNAL
C0BR_OUT_tb,C1BR_OUT_tb,C2BR_OUT_tb,C3BR_OUT_tb,C4BR_OUT_tb,C5BR_OUT_tb,C6BR_OUT
_tb,C7BR_OUT_tb: SIGNED(19 downto 0);
SIGNAL
C0BI_OUT_tb,C1BI_OUT_tb,C2BI_OUT_tb,C3BI_OUT_tb,C4BI_OUT_tb,C5BI_OUT_tb,C6BI_OUT
_tb,C7BI_OUT_tb: SIGNED(19 downto 0);


COMPONENT FFT_16 IS
PORT ( CLK,RESET: IN STD_LOGIC;
              DONE,READY: OUT STD_LOGIC;
       C0AR,C1AR,C2AR,C3AR,C4AR,C5AR,C6AR,C7AR: IN SIGNED(19 downto 0);
              C0AI,C1AI,C2AI,C3AI,C4AI,C5AI,C6AI,C7AI: IN SIGNED(19 downto 0);
              C0BR,C1BR,C2BR,C3BR,C4BR,C5BR,C6BR,C7BR: IN SIGNED(19 downto 0);
              C0BI,C1BI,C2BI,C3BI,C4BI,C5BI,C6BI,C7BI: IN SIGNED(19 downto 0);


C0AR_OUT,C1AR_OUT,C2AR_OUT,C3AR_OUT,C4AR_OUT,C5AR_OUT,C6AR_OUT,C7AR_OUT:
              OUT SIGNED(19 downto 0);

C0AI_OUT,C1AI_OUT,C2AI_OUT,C3AI_OUT,C4AI_OUT,C5AI_OUT,C6AI_OUT,C7AI_OUT:
              OUT SIGNED(19 downto 0);

C0BR_OUT,C1BR_OUT,C2BR_OUT,C3BR_OUT,C4BR_OUT,C5BR_OUT,C6BR_OUT,C7BR_OUT:
              OUT SIGNED(19 downto 0);

C0BI_OUT,C1BI_OUT,C2BI_OUT,C3BI_OUT,C4BI_OUT,C5BI_OUT,C6BI_OUT,C7BI_OUT:
              OUT SIGNED(19 downto 0));
END COMPONENT;

BEGIN

FFT: FFT_16 PORT MAP(CLK=>CLK_tb,RESET=>RESET_tb,DONE=>DONE_tb,READY=>READY_tb,

C0AR=>C0AR_tb,C1AR=>C1AR_tb,C2AR=>C2AR_tb,C3AR=>C3AR_tb,C4AR=>C4AR_tb,C5AR=>C5AR
_tb,C6AR=>C6AR_tb,C7AR=>C7AR_tb,
C0AI=>C0AI_tb,C1AI=>C1AI_tb,C2AI=>C2AI_tb,C3AI=>C3AI_tb,C4AI=>C4AI_tb,C5AI=>C5AI
_tb,C6AI=>C6AI_tb,C7AI=>C7AI_tb,
C0BR=>C0BR_tb,C1BR=>C1BR_tb,C2BR=>C2BR_tb,C3BR=>C3BR_tb,C4BR=>C4BR_tb,C5BR=>C5BR
_tb,C6BR=>C6BR_tb,C7BR=>C7BR_tb,
C0BI=>C0BI_tb,C1BI=>C1BI_tb,C2BI=>C2BI_tb,C3BI=>C3BI_tb,C4BI=>C4BI_tb,C5BI=>C5BI
_tb,C6BI=>C6BI_tb,C7BI=>C7BI_tb,


      C0AR_OUT=>C0AR_OUT_tb,C1AR_OUT=>C1AR_OUT_tb,C2AR_OUT=>C2AR_OUT_tb,C3AR_OUT
=>C3AR_OUT_tb,

      C4AR_OUT=>C4AR_OUT_tb,C5AR_OUT=>C5AR_OUT_tb,C6AR_OUT=>C6AR_OUT_tb,C7AR_OUT
=>C7AR_OUT_tb,
```

```
      C0AI_OUT=>C0AI_OUT_tb,C1AI_OUT=>C1AI_OUT_tb,C2AI_OUT=>C2AI_OUT_tb,C3AI_OUT
=>C3AI_OUT_tb,

      C4AI_OUT=>C4AI_OUT_tb,C5AI_OUT=>C5AI_OUT_tb,C6AI_OUT=>C6AI_OUT_tb,C7AI_OUT
=>C7AI_OUT_tb,


      C0BR_OUT=>C0BR_OUT_tb,C1BR_OUT=>C1BR_OUT_tb,C2BR_OUT=>C2BR_OUT_tb,C3BR_OUT
=>C3BR_OUT_tb,

      C4BR_OUT=>C4BR_OUT_tb,C5BR_OUT=>C5BR_OUT_tb,C6BR_OUT=>C6BR_OUT_tb,C7BR_OUT
=>C7BR_OUT_tb,


      C0BI_OUT=>C0BI_OUT_tb,C1BI_OUT=>C1BI_OUT_tb,C2BI_OUT=>C2BI_OUT_tb,C3BI_OUT
=>C3BI_OUT_tb,

      C4BI_OUT=>C4BI_OUT_tb,C5BI_OUT=>C5BI_OUT_tb,C6BI_OUT=>C6BI_OUT_tb,C7BI_OUT
=>C7BI_OUT_tb);

 clk_process: PROCESS

 BEGIN
 CLK_tb<= '0';
 wait for 10 ns;
 CLK_tb <= '1';
 wait for 10 ns;
 end process;

 --  0 : 00000000000000000000
 -- -1 : 10000000000000000000
 --  1 : 01111111111111111111
 -- 0.5: 01000000000000000000
 -- -0.5:11000000000000000000
 -- 0.75:01100000000000000000

 ingressi: PROCESS
 BEGIN
 C0AR_tb<="00000000000000000000";
 C1AR_tb<="00000000000000000000";
 C2AR_tb<="00000000000000000000";
 C3AR_tb<="00000000000000000000";
 C4AR_tb<="00000000000000000000";
 C5AR_tb<="00000000000000000000";
 C6AR_tb<="00000000000000000000";
 C7AR_tb<="00000000000000000000";
 C0AI_tb<="00000000000000000000";
 C1AI_tb<="00000000000000000000";
 C2AI_tb<="00000000000000000000";
 C3AI_tb<="00000000000000000000";
 C4AI_tb<="00000000000000000000";
 C5AI_tb<="00000000000000000000";
 C6AI_tb<="00000000000000000000";
 C7AI_tb<="00000000000000000000";
 C0BR_tb<="00000000000000000000";
 C1BR_tb<="00000000000000000000";
 C2BR_tb<="00000000000000000000";
 C3BR_tb<="00000000000000000000";
 C4BR_tb<="00000000000000000000";
 C5BR_tb<="00000000000000000000";
 C6BR_tb<="00000000000000000000";
```

```
 C7BR_tb<="00000000000000000000";
 C0BI_tb<="00000000000000000000";
 C1BI_tb<="00000000000000000000";
 C2BI_tb<="00000000000000000000";
 C3BI_tb<="00000000000000000000";
 C4BI_tb<="00000000000000000000";
 C5BI_tb<="00000000000000000000";
 C6BI_tb<="00000000000000000000";
 C7BI_tb<="00000000000000000000";
 wait for 50 ns;
 -- CAMPIONE 1
 C0AR_tb<="10000000000000000000";
 C1AR_tb<="10000000000000000000";
 C2AR_tb<="10000000000000000000";
 C3AR_tb<="10000000000000000000";
 C4AR_tb<="10000000000000000000";
 C5AR_tb<="10000000000000000000";
 C6AR_tb<="10000000000000000000";
 C7AR_tb<="10000000000000000000";
 C0AI_tb<="00000000000000000000";
 C1AI_tb<="00000000000000000000";
 C2AI_tb<="00000000000000000000";
 C3AI_tb<="00000000000000000000";
 C4AI_tb<="00000000000000000000";
 C5AI_tb<="00000000000000000000";
 C6AI_tb<="00000000000000000000";
 C7AI_tb<="00000000000000000000";
 C0BR_tb<="10000000000000000000";
 C1BR_tb<="10000000000000000000";
 C2BR_tb<="10000000000000000000";
 C3BR_tb<="10000000000000000000";
 C4BR_tb<="10000000000000000000";
 C5BR_tb<="10000000000000000000";
 C6BR_tb<="10000000000000000000";
 C7BR_tb<="10000000000000000000";
 C0BI_tb<="00000000000000000000";
 C1BI_tb<="00000000000000000000";
 C2BI_tb<="00000000000000000000";
 C3BI_tb<="00000000000000000000";
 C4BI_tb<="00000000000000000000";
 C5BI_tb<="00000000000000000000";
 C6BI_tb<="00000000000000000000";
 C7BI_tb<="00000000000000000000";
 wait for 140 ns;
 -- CAMPIONE 2
 C0AR_tb<="10000000000000000000";
C1AR_tb<="00000000000000000000";
C2AR_tb<="01111111111111111111";
C3AR_tb<="00000000000000000000";
C4AR_tb<="10000000000000000000";
C5AR_tb<="00000000000000000000";
C6AR_tb<="01111111111111111111";
C7AR_tb<="00000000000000000000";
C0AI_tb<="00000000000000000000";
C1AI_tb<="00000000000000000000";
C2AI_tb<="00000000000000000000";
C3AI_tb<="00000000000000000000";
C4AI_tb<="00000000000000000000";
C5AI_tb<="00000000000000000000";
C6AI_tb<="00000000000000000000";
C7AI_tb<="00000000000000000000";
C0BR_tb<="10000000000000000000";
```

```
C1BR_tb<="00000000000000000000";
C2BR_tb<="01111111111111111111";
C3BR_tb<="00000000000000000000";
C4BR_tb<="10000000000000000000";
C5BR_tb<="00000000000000000000";
C6BR_tb<="01111111111111111111";
C7BR_tb<="00000000000000000000";
C0BI_tb<="00000000000000000000";
C1BI_tb<="00000000000000000000";
C2BI_tb<="00000000000000000000";
C3BI_tb<="00000000000000000000";
C4BI_tb<="00000000000000000000";
C5BI_tb<="00000000000000000000";
C6BI_tb<="00000000000000000000";
C7BI_tb<="00000000000000000000";
 wait for 140 ns;
 -- CAMPIONE 3
 C0AR_tb<="01111111111111111111";
 C1AR_tb<="00000000000000000000";
 C2AR_tb<="00000000000000000000";
 C3AR_tb<="00000000000000000000";
 C4AR_tb<="00000000000000000000";
 C5AR_tb<="00000000000000000000";
 C6AR_tb<="00000000000000000000";
 C7AR_tb<="00000000000000000000";
 C0AI_tb<="00000000000000000000";
 C1AI_tb<="00000000000000000000";
 C2AI_tb<="00000000000000000000";
 C3AI_tb<="00000000000000000000";
 C4AI_tb<="00000000000000000000";
 C5AI_tb<="00000000000000000000";
 C6AI_tb<="00000000000000000000";
 C7AI_tb<="00000000000000000000";
 C0BR_tb<="00000000000000000000";
 C1BR_tb<="00000000000000000000";
 C2BR_tb<="00000000000000000000";
 C3BR_tb<="00000000000000000000";
 C4BR_tb<="00000000000000000000";
 C5BR_tb<="00000000000000000000";
 C6BR_tb<="00000000000000000000";
 C7BR_tb<="00000000000000000000";
 C0BI_tb<="00000000000000000000";
 C1BI_tb<="00000000000000000000";
 C2BI_tb<="00000000000000000000";
 C3BI_tb<="00000000000000000000";
 C4BI_tb<="00000000000000000000";
 C5BI_tb<="00000000000000000000";
 C6BI_tb<="00000000000000000000";
 C7BI_tb<="00000000000000000000";
 wait for 140 ns;
 -- CAMPIONE 4
 C0AR_tb<="10000000000000000000";
 C1AR_tb<="10000000000000000000";
 C2AR_tb<="01111111111111111111";
 C3AR_tb<="01111111111111111111";
 C4AR_tb<="10000000000000000000";
 C5AR_tb<="10000000000000000000";
 C6AR_tb<="01111111111111111111";
 C7AR_tb<="01111111111111111111";
 C0AI_tb<="00000000000000000000";
 C1AI_tb<="00000000000000000000";
 C2AI_tb<="00000000000000000000";
```

```
C3AI_tb<="00000000000000000000";
C4AI_tb<="00000000000000000000";
C5AI_tb<="00000000000000000000";
C6AI_tb<="00000000000000000000";
C7AI_tb<="00000000000000000000";
C0BR_tb<="10000000000000000000";
C1BR_tb<="10000000000000000000";
C2BR_tb<="01111111111111111111";
C3BR_tb<="01111111111111111111";
C4BR_tb<="10000000000000000000";
C5BR_tb<="10000000000000000000";
C6BR_tb<="01111111111111111111";
C7BR_tb<="01111111111111111111";
C0BI_tb<="00000000000000000000";
C1BI_tb<="00000000000000000000";
C2BI_tb<="00000000000000000000";
C3BI_tb<="00000000000000000000";
C4BI_tb<="00000000000000000000";
C5BI_tb<="00000000000000000000";
C6BI_tb<="00000000000000000000";
C7BI_tb<="00000000000000000000";
wait for 140 ns;
-- CAMPIONE 5
C0AR_tb<="01000000000000000000";
C1AR_tb<="01000000000000000000";
C2AR_tb<="01000000000000000000";
C3AR_tb<="01000000000000000000";
C4AR_tb<="01000000000000000000";
C5AR_tb<="01000000000000000000";
C6AR_tb<="01000000000000000000";
C7AR_tb<="01000000000000000000";
C0AI_tb<="00000000000000000000";
C1AI_tb<="00000000000000000000";
C2AI_tb<="00000000000000000000";
C3AI_tb<="00000000000000000000";
C4AI_tb<="00000000000000000000";
C5AI_tb<="00000000000000000000";
C6AI_tb<="00000000000000000000";
C7AI_tb<="00000000000000000000";
C0BR_tb<="01000000000000000000";
C1BR_tb<="11000000000000000000";
C2BR_tb<="11000000000000000000";
C3BR_tb<="11000000000000000000";
C4BR_tb<="11000000000000000000";
C5BR_tb<="11000000000000000000";
C6BR_tb<="11000000000000000000";
C7BR_tb<="11000000000000000000";
C0BI_tb<="00000000000000000000";
C1BI_tb<="00000000000000000000";
C2BI_tb<="00000000000000000000";
C3BI_tb<="00000000000000000000";
C4BI_tb<="00000000000000000000";
C5BI_tb<="00000000000000000000";
C6BI_tb<="00000000000000000000";
C7BI_tb<="00000000000000000000";
wait for 140 ns;
-- CAMPIONE 6
C0AR_tb<="00000000000000000000";
C1AR_tb<="00000000000000000000";
C2AR_tb<="00000000000000000000";
C3AR_tb<="00000000000000000000";
C4AR_tb<="00000000000000000000";
```

```
C5AR_tb<="00000000000000000000";
C6AR_tb<="00000000000000000000";
C7AR_tb<="00000000000000000000";
C0AI_tb<="00000000000000000000";
C1AI_tb<="00000000000000000000";
C2AI_tb<="00000000000000000000";
C3AI_tb<="00000000000000000000";
C4AI_tb<="00000000000000000000";
C5AI_tb<="00000000000000000000";
C6AI_tb<="00000000000000000000";
C7AI_tb<="00000000000000000000";
C0BR_tb<="01100000000000000000";
C1BR_tb<="00000000000000000000";
C2BR_tb<="00000000000000000000";
C3BR_tb<="00000000000000000000";
C4BR_tb<="00000000000000000000";
C5BR_tb<="00000000000000000000";
C6BR_tb<="00000000000000000000";
C7BR_tb<="00000000000000000000";
C0BI_tb<="00000000000000000000";
C1BI_tb<="00000000000000000000";
C2BI_tb<="00000000000000000000";
C3BI_tb<="00000000000000000000";
C4BI_tb<="00000000000000000000";
C5BI_tb<="00000000000000000000";
C6BI_tb<="00000000000000000000";
C7BI_tb<="00000000000000000000";
wait for 140 ns;
C0AR_tb<="00000000000000000000";
C1AR_tb<="00000000000000000000";
C2AR_tb<="00000000000000000000";
C3AR_tb<="00000000000000000000";
C4AR_tb<="00000000000000000000";
C5AR_tb<="00000000000000000000";
C6AR_tb<="00000000000000000000";
C7AR_tb<="00000000000000000000";
C0AI_tb<="00000000000000000000";
C1AI_tb<="00000000000000000000";
C2AI_tb<="00000000000000000000";
C3AI_tb<="00000000000000000000";
C4AI_tb<="00000000000000000000";
C5AI_tb<="00000000000000000000";
C6AI_tb<="00000000000000000000";
C7AI_tb<="00000000000000000000";
C0BR_tb<="00000000000000000000";
C1BR_tb<="00000000000000000000";
C2BR_tb<="00000000000000000000";
C3BR_tb<="00000000000000000000";
C4BR_tb<="00000000000000000000";
C5BR_tb<="00000000000000000000";
C6BR_tb<="00000000000000000000";
C7BR_tb<="00000000000000000000";
C0BI_tb<="00000000000000000000";
C1BI_tb<="00000000000000000000";
C2BI_tb<="00000000000000000000";
C3BI_tb<="00000000000000000000";
C4BI_tb<="00000000000000000000";
C5BI_tb<="00000000000000000000";
C6BI_tb<="00000000000000000000";
C7BI_tb<="00000000000000000000";
wait for 5000 ns;
END PROCESS;
```

```
 PROCESS
 BEGIN
 RESET_tb<='1';
 wait for 10 ns;
 RESET_tb<='0';
 wait for 10000 ns;
 END PROCESS;
 END behav;
```

BF.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY BF IS

PORT ( CLK, ENABLE: IN std_logic;
       AI,AR,BI,BR,WR,WI: IN SIGNED(19 downto 0);
      SEL_INV,SEL3,SEL1,SELSUM,C,A_S,EN_REGR : IN std_logic;
            SEL2 : IN std_logic_vector(1 downto 0);
            EN_REGO : IN std_logic_vector(2 downto 0);
            A1R,A1I,B1R,B1I,AI_CTRL,AR_CTRL,BI_CTRL,BR_CTRL: OUT SIGNED(19
downto 0));
END BF;

ARCHITECTURE BEHAV OF BF IS

COMPONENT RF IS
PORT ( CLK, ENABLE: IN std_logic;
       AI,AR,BI,BR,WR,WI: IN SIGNED(19 downto 0);
      SEL3,SEL1 : IN std_logic;
            SEL2 : IN std_logic_vector(1 downto 0);
            OUT_MUX1,OUT_MUX2,OUT_MUX3,AI_CTRL,AR_CTRL,BI_CTRL,BR_CTRL: OUT
SIGNED(19 downto 0));
END COMPONENT;


COMPONENT MOL IS
PORT ( C,CLK: IN std_logic;
       ADD1,ADD2: IN SIGNED(19 downto 0);
       RESULT: OUT SIGNED(38 downto 0));
END COMPONENT;

COMPONENT SUM IS
PORT ( A_S,CLK: IN std_logic;
       T1,T2: IN SIGNED(38 downto 0);
       RESULT: OUT SIGNED(39 downto 0));
END COMPONENT;

COMPONENT REG_39_M IS
PORT ( CLK: IN std_logic;
       D: IN SIGNED(38 downto 0);
       Q: OUT SIGNED(38 downto 0));
END COMPONENT;

COMPONENT REG_40_S IS
PORT ( CLK: IN std_logic;
       D: IN SIGNED(39 downto 0);
       Q: OUT SIGNED(39 downto 0));
```

```
END COMPONENT;

COMPONENT REG_ROUND IS
PORT ( CLK: IN std_logic;
              ENABLE: IN std_logic;
       D: IN SIGNED(39 downto 0);
       Q: OUT SIGNED(39 downto 0));
END COMPONENT;

COMPONENT MUX39SUM IS
PORT( A: IN SIGNED (19 downto 0);
            B: IN SIGNED(38 downto 0);
            SEL : IN std_logic;
            OUTPUT: OUT SIGNED(38 downto 0));
END COMPONENT;

COMPONENT REG_A1R IS
PORT ( CLK: IN std_logic;
              ENABLE: IN std_logic_vector(2 downto 0);
       D: IN SIGNED(19 downto 0);
       Q: OUT SIGNED(19 downto 0));
END COMPONENT;

COMPONENT REG_A1I IS
PORT ( CLK: IN std_logic;
              ENABLE: IN std_logic_vector(2 downto 0);
       D: IN SIGNED(19 downto 0);
       Q: OUT SIGNED(19 downto 0));
END COMPONENT;

COMPONENT REG_B1R IS
PORT ( CLK: IN std_logic;
              ENABLE: IN std_logic_vector(2 downto 0);
       D: IN SIGNED(19 downto 0);
       Q: OUT SIGNED(19 downto 0));
END COMPONENT;

COMPONENT REG_B1I IS
PORT ( CLK: IN std_logic;
              ENABLE: IN std_logic_vector(2 downto 0);
       D: IN SIGNED(19 downto 0);
       Q: OUT SIGNED(19 downto 0));
END COMPONENT;

COMPONENT ROUNDING IS
PORT ( IN_39: IN SIGNED(39 downto 0);
       OUT_20: OUT SIGNED(19 downto 0));
END COMPONENT;

COMPONENT MUX_INV IS
PORT( A: IN SIGNED (38 downto 0);
            B: IN SIGNED(38 downto 0);
            SEL : IN std_logic;
            OUTPUT: OUT SIGNED(38 downto 0));
END COMPONENT;


SIGNAL BUSW,BUS1,BUS2,ROUNDED: SIGNED (19 downto 0);
SIGNAL MUXSUM_OUT,MOL_RES_IN,MOL_RES_OUT,OUTMUXINVT1,OUTMUXINVT2: SIGNED(38
downto 0);
SIGNAL SUM_RES_IN,SUM_RES_OUT,REG_ROUND_OUT: SIGNED(39 DOWNTO 0);
SIGNAL NOTHING: std_logic;
```

```
BEGIN

ROUND: ROUNDING PORT MAP(IN_39=>REG_ROUND_OUT,OUT_20=>ROUNDED);

MUXSUM: MUX39SUM PORT MAP(A=>BUS1,B=>SUM_RES_OUT(38 DOWNTO 0),SEL=>SELSUM,
OUTPUT=>MUXSUM_OUT);

REG_RF: RF PORT MAP (CLK=>CLK,ENABLE=>ENABLE,AI=>AI,AR=>AR,BI=>BI,BR=>BR,WR=>WR,
WI=>WI,SEL1=>SEL1,SEL2=>SEL2,SEL3=>SEL3,AI_CTRL=>AI_CTRL,AR_CTRL=>AR_CTRL,BI_CTR
L=>BI_CTRL,BR_CTRL=>BR_CTRL,OUT_MUX1=>BUS1,OUT_MUX2=>BUS2,OUT_MUX3=> BUSW);

MOLTIPLICATORE: MOL PORT MAP(CLK=>CLK,C=>C,ADD1=>BUS2,ADD2=>BUSW,RESULT=>
MOL_RES_IN);
SOMMATORE: SUM PORT MAP(CLK=>CLK,A_S=>A_S,T1=>OUTMUXINVT1,T2=>OUTMUXINVT2,
RESULT=>SUM_RES_IN);

REG_MOL: REG_39_M PORT MAP(CLK=>CLK,D=>MOL_RES_IN,Q=>MOL_RES_OUT);
REG_SUM : REG_40_S PORT MAP(CLK=>CLK,D=>SUM_RES_IN,Q=>SUM_RES_OUT);
R_ROUNDING: REG_ROUND PORT MAP(CLK=>CLK, ENABLE=>EN_REGR,
D=>SUM_RES_OUT,Q=>REG_ROUND_OUT);
R_A1R: REG_A1R PORT MAP(CLK=>CLK,ENABLE=>EN_REGO,D=>ROUNDED(19 DOWNTO0),Q=>A1R);
R_A1I: REG_A1I PORT MAP(CLK=>CLK,ENABLE=>EN_REGO,D=>ROUNDED(19 DOWNTO0),Q=>A1I);
R_B1R: REG_B1R PORT MAP(CLK=>CLK,ENABLE=>EN_REGO,D=>ROUNDED(19 DOWNTO0),Q=>B1R);
R_B1I: REG_B1I PORT MAP(CLK=>CLK,ENABLE=>EN_REGO,D=>ROUNDED(19 DOWNTO0),Q=>B1I);
INVT1: MUX_INV PORT MAP(SEL=>SEL_INV,A=>MUXSUM_OUT,B=>MOL_RES_OUT,
OUTPUT=>OUTMUXINVT1);
INVT2: MUX_INV PORT MAP(SEL=>SEL_INV,A=>MOL_RES_OUT,B=>MUXSUM_OUT,
OUTPUT=>OUTMUXINVT2);

end behav;
```

MUX_INV.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY MUX_INV IS
PORT( A: IN SIGNED (38 downto 0);
            B: IN SIGNED(38 downto 0);
            SEL : IN std_logic;
            OUTPUT: OUT SIGNED(38 downto 0));
END MUX_INV;

ARCHITECTURE behav OF MUX_INV IS

BEGIN


with SEL  select
    OUTPUT <= A when '0',
                          B when '1',

         "00000000000000000000000000000000000000"  when others;
end Behav;
```

MUX39SUM

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY MUX39SUM IS
PORT( A: IN SIGNED (19 downto 0);                -- Q1.19
            B: IN SIGNED(38 downto 0);                          -- Q1.38
            SEL : IN std_logic;
            OUTPUT: OUT SIGNED(38 downto 0));
END MUX39SUM;

ARCHITECTURE behav OF MUX39SUM IS
SIGNAL sign_ext_a : SIGNED(18 DOWNTO 0);
SIGNAL new_a : SIGNED (38 DOWNTO 0);
BEGIN
--sign_ext_a <=(others=> A(0));                  -- estensione LSB nella parte
decimale
sign_ext_a <=(others=> '0');                     -- estensione zeri nella parte
decimale

new_a(38 downto 19) <= A;
new_a(18 downto 0) <= sign_ext_a;

with SEL  select
    OUTPUT <= new_a when '0',
                          B when '1',

         "00000000000000000000000000000000000000"  when others;
end Behav;
```

REG_39_M.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY REG_39_M IS

PORT ( CLK: IN std_logic;
       D: IN SIGNED(38 downto 0);
       Q: OUT SIGNED(38 downto 0));
END REG_39_M;

ARCHITECTURE behav OF REG_39_M IS

BEGIN

PROCESS(CLK)

BEGIN
```

```
            IF (CLK'EVENT AND CLK = '1') THEN
            Q<= D;
            END IF;
            END PROCESS;

      END behav;
```

REG_40_S.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY REG_40_S IS

PORT ( CLK: IN std_logic;
       D: IN SIGNED(39 downto 0);
       Q: OUT SIGNED(39 downto 0));
END REG_40_S;

ARCHITECTURE behav OF REG_40_S IS

BEGIN

PROCESS(CLK)

BEGIN

            IF (CLK'EVENT AND CLK = '1') THEN
            Q<= D;
            END IF;
            END PROCESS;

      END behav;
```

REG_A1I.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY REG_A1I IS

PORT ( CLK: IN std_logic;
             ENABLE: IN std_logic_vector(2 downto 0);
       D: IN SIGNED(19 downto 0);
       Q: OUT SIGNED(19 downto 0));
END REG_A1I;

ARCHITECTURE behav OF REG_A1I IS

BEGIN
```

```
PROCESS(CLK)

BEGIN

            IF (CLK'EVENT AND CLK = '1') THEN
            IF(ENABLE = "010") THEN
            Q<= D;
            END IF;
            END IF;
            END PROCESS;

      END behav;
```

REG_A1R.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY REG_A1R IS

PORT ( CLK: IN std_logic;
             ENABLE: IN std_logic_vector(2 downto 0);
       D: IN SIGNED(19 downto 0);
       Q: OUT SIGNED(19 downto 0));
END REG_A1R;

ARCHITECTURE behav OF REG_A1R IS

BEGIN

PROCESS(CLK)

BEGIN

            IF (CLK'EVENT AND CLK = '1') THEN
            IF(ENABLE = "001") THEN
            Q<= D;
            END IF;
            END IF;
            END PROCESS;

      END behav;
```

REG_B1I.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY REG_B1I IS

PORT ( CLK: IN std_logic;
              ENABLE: IN std_logic_vector(2 downto 0);
       D: IN SIGNED(19 downto 0);
       Q: OUT SIGNED(19 downto 0));
END REG_B1I;

ARCHITECTURE behav OF REG_B1I IS

BEGIN

PROCESS(CLK)

BEGIN

            IF (CLK'EVENT AND CLK = '1') THEN
            IF(ENABLE = "100") THEN
            Q<= D;
            END IF;
            END IF;
            END PROCESS;

      END behav;
```

REG_B1R

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY REG_B1R IS

PORT ( CLK: IN std_logic;
              ENABLE: IN std_logic_vector(2 downto 0);
       D: IN SIGNED(19 downto 0);
       Q: OUT SIGNED(19 downto 0));
END REG_B1R;

ARCHITECTURE behav OF REG_B1R IS

BEGIN

PROCESS(CLK)

BEGIN
```

```
            IF (CLK'EVENT AND CLK = '1') THEN
            IF(ENABLE = "011") THEN
            Q<= D;
            END IF;
            END IF;
            END PROCESS;

      END behav;
```

REG_ROUND.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY REG_ROUND IS

PORT ( CLK: IN std_logic;
             ENABLE: IN std_logic;
       D: IN SIGNED(39 downto 0);
       Q: OUT SIGNED(39 downto 0));
END REG_ROUND;

ARCHITECTURE behav OF REG_ROUND IS

BEGIN

PROCESS(CLK)

BEGIN

            IF (CLK'EVENT AND CLK = '1') THEN
            IF(ENABLE = '1') THEN
            Q<= D;
            END IF;
            END IF;
            END PROCESS;

      END behav;
```

ROUNDING.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY ROUNDING IS

PORT ( IN_39: IN SIGNED(39 downto 0);                -- Q2.38
       OUT_20: OUT SIGNED(19 downto 0));
END ROUNDING;

ARCHITECTURE behav of ROUNDING IS

SIGNAL OUT_20_S: SIGNED (19 DOWNTO 0);
SIGNAL IN_39_S: SIGNED(39 DOWNTO 0);
BEGIN


IN_39_S(39 DOWNTO 1) <= IN_39(38 DOWNTO 0);          -- sposto la virgola
IN_39_S(0) <='0';                                                    --
Q1.19

PROCESS (IN_39_S)
BEGIN


case IN_39_S(19 DOWNTO 0) is
   when "10000000000000000000" =>
     IF (IN_39_S(20) = '1') THEN
                  OUT_20_S <= IN_39_S(39 DOWNTO 20) + 1;
                  ELSE
                  OUT_20_S <= IN_39_S(39 DOWNTO 20);
                  END IF;
   when OTHERS =>
                  IF (IN_39_S(19) = '1') THEN
                  OUT_20_S <= IN_39_S(39 DOWNTO 20) + 1;
                  ELSE
                  OUT_20_S <= IN_39_S(39 DOWNTO 20);
                  END IF;
end case;
END PROCESS;


OUT_20(19) <= OUT_20_S(19);                                          --
riduco la dinamica
OUT_20(18 DOWNTO 0) <= OUT_20_S(19 DOWNTO 1);

      end behav;
```

CU_BF.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY CU_BF IS
PORT( LOAD,SEQ,CLK,RESET: IN std_logic;
```

```
            CTRL_OUT: OUT std_logic_vector(14 downto 0));
END CU_BF;

ARCHITECTURE behav OF CU_BF IS

COMPONENT MUX_20to1 IS
PORT( A,B: IN std_logic_vector(20 downto 0);
            SEL : IN std_logic;
            OUTPUT: OUT std_logic_vector(20 downto 0));
END COMPONENT;

COMPONENT REG_PS_BF IS
PORT ( CLK,RESET: IN std_logic;
       D: IN std_logic_vector(4 downto 0);
       Q: OUT std_logic_vector(4 downto 0));
END COMPONENT;


COMPONENT PLA_BF IS
PORT( LOAD,SEQ,LSB,CC : IN std_logic;
            LSB_OUT: OUT std_logic);
END COMPONENT;

COMPONENT uROM is
      generic( row : integer:= 10;
                        column : integer:= 42
                        );
      port( M: in std_logic_vector (3 downto 0);
                  OUT_uROM: out std_logic_vector (column-1 downto 0)
                  );
end COMPONENT;

SIGNAL add,NEXTS :std_logic_vector(3 downto 0);
SIGNAL AS,BS :std_logic_vector(20 downto 0);
SIGNAL CCs, LSB_MP,LSB_OUTS,lsb_sel : std_logic;

BEGIN
ROM: uROM PORT MAP (M=> add, OUT_uROM(42-1 downto 42-21) => AS, OUT_uROM(42-22
downto 0)=> BS);

PLA: PLA_BF PORT MAP( LOAD=>LOAD,SEQ=>SEQ,CC=>CCs,LSB=>LSB_MP,
LSB_OUT=>LSB_OUTS);
REG_PS: REG_PS_BF PORT MAP(RESET=>RESET,CLK=>CLK,D(0)=>LSB_OUTS,D(4downto1)
=>NEXTS,Q(4 downto 1)=> add,Q(0)=>lsb_sel);

MUX_20:MUX_20to1 PORT MAP ( A=>AS,B=>BS, SEL=>lsb_sel,OUTPUT(20)=>CCS,
OUTPUT(19 downto 16)=>NEXTS,OUTPUT(15)=>LSB_MP,OUTPUT(14 downto 0)=> CTRL_OUT);

      END BEHAV;
```

MUX_2TO1.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;
```

```
ENTITY MUX_2to1 IS
PORT( A,B,SEL : IN std_logic;
            OUTPUT: OUT std_logic);
END MUX_2to1;

ARCHITECTURE behav OF MUX_2to1 IS

BEGIN
with SEL  select
    OUTPUT <= A when '0',
                          B when '1',

         '0'  when others;
end Behav;
```

MUX_20TO1.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY MUX_20to1 IS
PORT( A,B: IN std_logic_vector(20 downto 0);
            SEL : IN std_logic;
            OUTPUT: OUT std_logic_vector(20 downto 0));
END MUX_20to1;

ARCHITECTURE behav OF MUX_20to1 IS

BEGIN
with SEL  select
    OUTPUT <= A when '0',
                          B when '1',
      "00000000000000000000" when OTHERS;
end Behav;
```

PLA_BF.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;
```

```
ENTITY PLA_BF IS
PORT( LOAD,SEQ,LSB,CC : IN std_logic;
            LSB_OUT: OUT std_logic);
END PLA_BF;

ARCHITECTURE behav OF PLA_BF IS

COMPONENT MUX_2to1 IS
PORT( A,B,SEL : IN std_logic;
            OUTPUT: OUT std_logic);
END COMPONENT;

SIGNAL LOGICA : std_logic;
BEGIN
LOGICA <= (SEQ AND NOT(LSB)) OR (LOAD AND LSB);
mux2: MUX_2to1 PORT MAP( A=> LSB, B=> LOGICA, SEL=> CC,  OUTPUT=> LSB_OUT);
      END behav;
```

REG_PS_BF.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY REG_PS_BF IS

PORT ( CLK,RESET: IN std_logic;
       D: IN std_logic_vector(4 downto 0);
       Q: OUT std_logic_vector(4 downto 0));
END REG_PS_BF;

ARCHITECTURE behav OF REG_PS_BF IS

BEGIN

PROCESS(CLK,RESET)

BEGIN
      IF (RESET = '1') THEN
            Q<="00000";
            ELSE IF (CLK'EVENT AND CLK = '1') THEN
            Q<= D;
            END IF;
            END IF;
            END PROCESS;
```

```
END behav;
```

uROM.vhd

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

entity uROM is
      generic( row : integer:= 11;
                     column : integer:= 42
                     );
      port( M: in std_logic_vector (3 downto 0);
                  OUT_uROM: out std_logic_vector (column-1 downto 0)
                  );
end uROM;

architecture Behavioral of uROM is

      type uROM_MATRIX is array (0 to 10) of std_logic_vector(column-1 downto
0);
      signal uROM_LINES: uROM_MATRIX;
      signal
r11,r12,r21,r22,r31,r32,r41,r42,r51,r52,r61,r62,r71,r72,r81,r82,r91,r92,r101,r10
2,r111 : std_logic_vector (20 downto 0);
      begin
            -- CC(1) | NA(5) | COMAND (15)


            r11 <= "1000010UUUUUUU0000000";     -- IDLE
            r12 <= "0000100UUUUUUU0000000";     -- LOAD_ST
            r21 <= "0000110UU1001U0000000";     -- M1
            r22 <= "0001000UU1011U0000000";     -- M3
            r31 <= "00010100011110000000";      -- S1_M2
            r32 <= "00011000111010000000";      -- S3_M4
            r41 <= "00011101U00U010000000";     -- S2_M5
            r42 <= "10100001U01U000000010";     -- S4_M6
            r51 <= "00101011UUUUU10100000";     -- S5
            r52 <= "00111011UUUUU10100000";     -- LOAD_ST_S
            r61 <= "00101111UUUUU10100100";     -- S6
            r62 <= "0011000UUUUUUU0101000";     -- ROUND3
            r71 <= "0011010UUUUUUU0101100";     -- ROUND4
            r72 <= "0101000UUUUUUU1010000";     -- LAST_SAVE
            r81 <= "00111111U100110100100";     -- M1_S
            r82 <= "0100000UU1011U0101000";     -- M3_S
            r91 <= "01000100011110010100";      -- S1_M2_S
            r92 <= "01001000111010001000";      -- S3_M4_S
            r101 <= "01001101U00U010000001";    -- S2_M5_S
            r102 <= "10100001U01U000000010"; -- S4_M6_S
            r111 <= "0000000UUUUUUU0000001";    -- DONE

            uROM_LINES(0) <= r11 & r12;
            uROM_LINES(1) <= r21 & r22;
```

```
            uROM_LINES(2) <= r31 & r32;
            uROM_LINES(3) <= r41 & r42;
            uROM_LINES(4) <= r51 & r52;
            uROM_LINES(5) <= r61 & r62;
            uROM_LINES(6) <= r71 & r72;
            uROM_LINES(7) <= r81 & r82;
            uROM_LINES(8) <= r91 & r92;
            uROM_LINES(9) <= r101 & r102;
            uROM_LINES(10) <= r111 & r111;

            OUT_uROM <= uROM_LINES(to_integer(unsigned(M)));

      end Behavioral;
```

CU_TOP.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY CU_TOP IS
PORT( START,PROGRESS,FREE_M,END_BF,SEQ,CLK,RESET: IN std_logic;
            CTRL_TOP_OUT: OUT std_logic_vector(5 downto 0));
END CU_TOP;

ARCHITECTURE behav OF CU_TOP IS

COMPONENT MUX_12to1_TOP IS
PORT( A,B,C,D: IN std_logic_vector(11 downto 0);
            SEL : IN std_logic_vector(1 downto 0);
            OUTPUT: OUT std_logic_vector(11 downto 0));
END COMPONENT;


COMPONENT REG_PS_TOP IS
PORT ( CLK,RESET: IN std_logic;
       D: IN std_logic_vector(3 downto 0);
       Q: OUT std_logic_vector(3 downto 0));
END COMPONENT;


COMPONENT PLA_TOP IS
PORT( START,PROGRESS,FREE_M,END_BF,SEQ: IN std_logic;
      BIT0_IN,BIT1_IN: IN std_logic;
            CC: IN std_logic_vector(1 downto 0);
            BIT0_OUT,BIT1_OUT: OUT std_logic);
END COMPONENT;

COMPONENT uROM_TOP is
      generic( row : integer:= 3;
                        column : integer:= 48
```

```
                        );
      port( M: in std_logic_vector (1 downto 0);
                  OUT_uROM: out std_logic_vector (column-1 downto 0)
                  );
end COMPONENT;

SIGNAL add,lsb_sel:std_logic_vector(1 downto 0);
SIGNAL AS,BS,CS,DS :std_logic_vector(11 downto 0);
SIGNAL CCS: STD_LOGIC_VECTOR(1 downto 0);
SIGNAL BIT3,BIT0_MP,BIT1_MP,BIT2,BIT0_OUTS,BIT1_OUTS : std_logic;

BEGIN
ROM: uROM_TOP PORT MAP (M=> add,OUT_uROM(47 downto 36) => AS, OUT_uROM(35 downto
24)=> BS,
                                                        OUT_uROM(23 downto
12) => CS, OUT_uROM(11 downto 0)=> DS);

PLA: PLA_TOP PORT MAP(
SEQ=>SEQ,CC=>CCs,BIT0_IN=>BIT0_MP,BIT1_IN=>BIT1_MP,BIT0_OUT=>BIT0_OUTS,

BIT1_OUT=>BIT1_OUTS,START=>START,PROGRESS=>PROGRESS,FREE_M=>FREE_M,
                                          END_BF=>END_BF);
REG_PS: REG_PS_TOP PORT MAP(
CLK=>CLK,RESET=>RESET,D(0)=>BIT0_OUTS,D(1)=>BIT1_OUTS,D(2)=>BIT2,D(3)=>BIT3,
                                                  Q(3 downto 2)=>
add,Q(1)=>lsb_sel(1),Q(0)=>lsb_sel(0));

MUX_12:  MUX_12to1_TOP PORT MAP ( A=>AS,B=>BS,C=>CS,D=>DS,SEL=>lsb_sel,OUTPUT(11
DOWNTO 10)=>CCS,

OUTPUT(9)=>BIT3,OUTPUT(8)=>BIT2,OUTPUT(7)=>BIT1_MP,OUTPUT(6)=>BIT0_MP,
                                                              OUTPUT(5
downto 0)=> CTRL_TOP_OUT);
      END BEHAV;
```

MUX_12TO1_TOP.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY MUX_12to1_TOP IS
PORT( A,B,C,D: IN std_logic_vector(11 downto 0);
            SEL : IN std_logic_vector(1 downto 0);
            OUTPUT: OUT std_logic_vector(11 downto 0));
END MUX_12to1_TOP;

ARCHITECTURE behav OF MUX_12to1_TOP IS

BEGIN
with SEL  select
  OUTPUT <= A when "00",
                          B when "01",
```

```
                        c when "10",
                        D when "11",

        "000000000000"  when others;

end Behav;
```

PLA_TOP.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY PLA_TOP IS
PORT( START,PROGRESS,FREE_M,END_BF,SEQ: IN std_logic;
      BIT0_IN,BIT1_IN: IN std_logic;
            CC: IN std_logic_vector(1 downto 0);
            BIT0_OUT,BIT1_OUT: OUT std_logic);
END PLA_TOP;

ARCHITECTURE behav OF PLA_TOP IS

COMPONENT MUX_2to1 IS
PORT( A,B,SEL : IN std_logic;
            OUTPUT: OUT std_logic);
END COMPONENT;

SIGNAL LOGICA_0,LOGICA_1,NEW_FREE_M : std_logic;

BEGIN

NEW_FREE_M <= FREE_M AND SEQ;

LOGICA_0 <= START OR (NOT(NEW_FREE_M) AND END_BF) OR (NOT(NEW_FREE_M) AND
PROGRESS) ; -- C + A' * B + A' * D
LOGICA_1 <= (NOT(NEW_FREE_M) AND NOT(END_BF)) OR (NEW_FREE_M AND END_BF);
                        -- A' * B' + A * B

mux0: MUX_2to1 PORT MAP( A=> BIT0_IN, B=> LOGICA_0, SEL=> CC(0),  OUTPUT=>
BIT0_OUT);
mux1: MUX_2to1 PORT MAP( A=> BIT1_IN, B=> LOGICA_1, SEL=> CC(1),  OUTPUT=>
BIT1_OUT);

      END behav;
```

REG_PS_TOP.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY REG_PS_TOP IS

PORT ( CLK,RESET: IN std_logic;
       D: IN std_logic_vector(3 downto 0);
       Q: OUT std_logic_vector(3 downto 0));
END REG_PS_TOP;

ARCHITECTURE behav OF REG_PS_TOP IS

BEGIN

PROCESS(CLK,RESET)

BEGIN
      IF (RESET = '1') THEN
            Q<= "0011";
            ELSE IF (CLK'EVENT AND CLK = '1') THEN
            Q<= D;
            END IF;
            END IF;

            END PROCESS;

      END behav;
```

uROM_TOP.VHD

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

entity uROM_TOP is
      generic( row : integer:= 3;
                     column : integer:= 48
                     );
      port( M: in std_logic_vector (1 downto 0);
                  OUT_uROM: out std_logic_vector (column-1 downto 0)
                  );
end uROM_TOP;

architecture Behavioral of uROM_TOP is

      type uROM_MATRIX is array (0 to 2) of std_logic_vector(column-1 downto 0);
      signal uROM_LINES: uROM_MATRIX;
      signal r0011,r0000,r0001,r0111,r0100,r0101,r0110,r1010,r1011,rnull :
std_logic_vector (11 downto 0);
      begin
            -- CC(1) | NA(4) | COMAND (12)


            r0011 <= "000000000001";      -- RESET
            r0000 <= "010001000000";      -- IDLE
            r0001 <= "000111110000";      -- LOAD
            r0111 <= "110100001000";      -- WORK
            r0100 <= "000111101000";      -- FREE_M
            r0101 <= "011010000010";      -- READY
            r0110 <= "000111101010";      -- FREE_M-READY
            r1010 <= "000011000100";      -- DONE
            r1011 <= "000111000000";      -- PREWORK
            rnull <= "UUUUUUUUUUUU";      -- NULL

            uROM_LINES(0) <= r0000 & r0001 & rnull & r0011;
            uROM_LINES(1) <= r0100 & r0101 & r0110 & r0111;
            uROM_LINES(2) <= rnull & rnull & r1010 & r1011;


            OUT_uROM <= uROM_LINES(to_integer(unsigned(M)));

      end Behavioral;
```

MOL.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY MOL IS

PORT ( C,CLK: IN std_logic;
       ADD1,ADD2: IN SIGNED(19 downto 0);
       RESULT: OUT SIGNED(38 downto 0));
END MOL;
```

```
ARCHITECTURE behav OF MOL IS

COMPONENT MOLTI IS
PORT ( C: IN std_logic;
       ADD1,ADD2: IN SIGNED(19 downto 0);
       RESULT: OUT SIGNED(39 downto 0));
END COMPONENT;

COMPONENT REG_MOL IS
PORT ( CLK: IN std_logic;
       D: IN SIGNED(38 downto 0);
       Q: OUT SIGNED(38 downto 0));
END COMPONENT;

SIGNAL partial: SIGNED(39 DOWNTO 0);

BEGIN



moltiplicatore: MOLTI PORT MAP(C=>C,ADD1=>ADD1,ADD2=>ADD2,RESULT=>partial);
reg: REG_MOL PORT MAP( CLK=>CLK,D=>partial(38 DOWNTO 0),Q=> RESULT);


      END behav;
```

MOLTI.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY MOLTI IS

PORT ( C: IN std_logic;
       ADD1,ADD2: IN SIGNED(19 downto 0);
       RESULT: OUT SIGNED(39 downto 0));
END MOLTI;

ARCHITECTURE behav OF MOLTI IS

SIGNAL sign_ext: SIGNED(19 DOWNTO 0);

BEGIN
sign_ext <=(others=> ADD1(0));

PROCESS(C,ADD1,ADD2,sign_ext)

BEGIN
      IF (C= '1') THEN                                          -- ADD1*ADD2
      RESULT <= ADD1 * ADD2;
      END IF;
      IF (C = '0') THEN                                         -- ADD1*2
```

```
      RESULT(38 DOWNTO 20) <= ADD1(18 DOWNTO 0);
      RESULT(19 DOWNTO 0) <= sign_ext;
      RESULT(39) <= ADD1(0);
      END IF;
END PROCESS;

      END behav;
```

REG_MOL.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY REG_MOL IS

PORT ( CLK: IN std_logic;
       D: IN SIGNED(38 downto 0);
       Q: OUT SIGNED(38 downto 0));
END REG_MOL;

ARCHITECTURE behav OF REG_MOL IS

BEGIN


PROCESS(CLK)

BEGIN

            IF (CLK'EVENT AND CLK = '1') THEN
            Q<= D;
            END IF;
            END PROCESS;

      END behav;
```

OR_PORT.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
```

```
USE ieee.numeric_std.all;

ENTITY OR_PORT IS

PORT (
       B0AR,B1AR,B2AR,B3AR,B4AR,B5AR,B6AR,B7AR: IN SIGNED(19 downto 0);
             B0AI,B1AI,B2AI,B3AI,B4AI,B5AI,B6AI,B7AI: IN SIGNED(19 downto 0);
             B0BR,B1BR,B2BR,B3BR,B4BR,B5BR,B6BR,B7BR: IN SIGNED(19 downto 0);
             B0BI,B1BI,B2BI,B3BI,B4BI,B5BI,B6BI,B7BI: IN SIGNED(19 downto 0);
       START: OUT std_logic);
END OR_PORT;

ARCHITECTURE behav OF OR_PORT IS

SIGNAL B0ARS,B1ARS,B2ARS,B3ARS,B4ARS,B5ARS,B6ARS,B7ARS: STD_LOGIC_VECTOR(19
downto 0);
SIGNAL B0AIS,B1AIS,B2AIS,B3AIS,B4AIS,B5AIS,B6AIS,B7AIS: STD_LOGIC_VECTOR(19
downto 0);
SIGNAL B0BRS,B1BRS,B2BRS,B3BRS,B4BRS,B5BRS,B6BRS,B7BRS: STD_LOGIC_VECTOR(19
downto 0);
SIGNAL B0BIS,B1BIS,B2BIS,B3BIS,B4BIS,B5BIS,B6BIS,B7BIS: STD_LOGIC_VECTOR(19
downto 0);

SIGNAL      START0,START1,START2,START3,START4,START5,START6,START7,START8,
                  START9,START10,START11,START12,START13,START14,START15,

      START16,START17,START18,START19,START20,START21,START22,START23,

      START24,START25,START26,START27,START28,START29,START30,START31:
STD_LOGIC;

 BEGIN

B0ARS<= std_logic_vector(B0AR);
B1ARS<= std_logic_vector(B1AR);
B2ARS<= std_logic_vector(B2AR);
B3ARS<= std_logic_vector(B3AR);
B4ARS<= std_logic_vector(B4AR);
B5ARS<= std_logic_vector(B5AR);
B6ARS<= std_logic_vector(B6AR);
B7ARS<= std_logic_vector(B7AR);

B0AIS<= std_logic_vector(B0AI);
B1AIS<= std_logic_vector(B1AI);
B2AIS<= std_logic_vector(B2AI);
B3AIS<= std_logic_vector(B3AI);
B4AIS<= std_logic_vector(B4AI);
B5AIS<= std_logic_vector(B5AI);
B6AIS<= std_logic_vector(B6AI);
B7AIS<= std_logic_vector(B7AI);

B0BIS<= std_logic_vector(B0BI);
B1BIS<= std_logic_vector(B1BI);
B2BIS<= std_logic_vector(B2BI);
B3BIS<= std_logic_vector(B3BI);
B4BIS<= std_logic_vector(B4BI);
B5BIS<= std_logic_vector(B5BI);
B6BIS<= std_logic_vector(B6BI);
B7BIS<= std_logic_vector(B7BI);

B0BRS<= std_logic_vector(B0BR);
B1BRS<= std_logic_vector(B1BR);
```

```
B2BRS<= std_logic_vector(B2BR);
B3BRS<= std_logic_vector(B3BR);
B4BRS<= std_logic_vector(B4BR);
B5BRS<= std_logic_vector(B5BR);
B6BRS<= std_logic_vector(B6BR);
B7BRS<= std_logic_vector(B7BR);



START0<= B0ARS(19) OR B0ARS(18) OR B0ARS(17) OR B0ARS(16) OR B0ARS(15) OR
B0ARS(14) OR B0ARS(13) OR                 B0ARS(12) OR B0ARS(11) OR B0ARS(10) OR
B0ARS(9) OR B0ARS(8) OR B0ARS(7) OR B0ARS(6) OR              B0ARS(5) OR
B0ARS(4) OR B0ARS(3) OR B0ARS(2) OR B0ARS(1) OR B0ARS(0);

START1<= B1ARS(19) OR B1ARS(18) OR B1ARS(17) OR B1ARS(16) OR B1ARS(15) OR
B1ARS(14) OR B1ARS(13) OR                 B1ARS(12) OR B1ARS(11) OR B1ARS(10) OR
B1ARS(9) OR B1ARS(8) OR B1ARS(7) OR B1ARS(6) OR              B1ARS(5) OR
B1ARS(4) OR B1ARS(3) OR B1ARS(2) OR B1ARS(1) OR B1ARS(0);

START2<= B2ARS(19) OR B2ARS(18) OR B2ARS(17) OR B2ARS(16) OR B2ARS(15) OR
B2ARS(14) OR B2ARS(13) OR                 B2ARS(12) OR B2ARS(11) OR B2ARS(10) OR
B2ARS(9) OR B2ARS(8) OR B2ARS(7) OR B2ARS(6) OR              B2ARS(5) OR
B2ARS(4) OR B2ARS(3) OR B2ARS(2) OR B2ARS(1) OR B2ARS(0);

START3<= B3ARS(19) OR B3ARS(18) OR B3ARS(17) OR B3ARS(16) OR B3ARS(15) OR
B3ARS(14) OR B3ARS(13) OR                 B3ARS(12) OR B3ARS(11) OR B3ARS(10) OR
B3ARS(9) OR B3ARS(8) OR B3ARS(7) OR B3ARS(6) OR              B3ARS(5) OR
B3ARS(4) OR B3ARS(3) OR B3ARS(2) OR B3ARS(1) OR B3ARS(0);

START4<= B4ARS(19) OR B4ARS(18) OR B4ARS(17) OR B4ARS(16) OR B4ARS(15) OR
B4ARS(14) OR B4ARS(13) OR                 B4ARS(12) OR B4ARS(11) OR B4ARS(10) OR
B4ARS(9) OR B4ARS(8) OR B4ARS(7) OR B4ARS(6) OR              B4ARS(5) OR
B4ARS(4) OR B4ARS(3) OR B4ARS(2) OR B4ARS(1) OR B4ARS(0);

START5<= B5ARS(19) OR B5ARS(18) OR B5ARS(17) OR B5ARS(16) OR B5ARS(15) OR
B5ARS(14) OR B5ARS(13) OR                 B5ARS(12) OR B5ARS(11) OR B5ARS(10) OR
B5ARS(9) OR B5ARS(8) OR B5ARS(7) OR B5ARS(6) OR              B5ARS(5) OR
B5ARS(4) OR B5ARS(3) OR B5ARS(2) OR B5ARS(1) OR B5ARS(0);

START6<= B6ARS(19) OR B6ARS(18) OR B6ARS(17) OR B6ARS(16) OR B6ARS(15) OR
B6ARS(14) OR B6ARS(13) OR                 B6ARS(12) OR B6ARS(11) OR B6ARS(10) OR
B6ARS(9) OR B6ARS(8) OR B6ARS(7) OR B6ARS(6) OR              B6ARS(5) OR
B6ARS(4) OR B6ARS(3) OR B6ARS(2) OR B6ARS(1) OR B6ARS(0);

START7<= B7ARS(19) OR B7ARS(18) OR B7ARS(17) OR B7ARS(16) OR B7ARS(15) OR
B7ARS(14) OR B7ARS(13) OR                 B7ARS(12) OR B7ARS(11) OR B7ARS(10) OR
B7ARS(9) OR B7ARS(8) OR B7ARS(7) OR B7ARS(6) OR              B7ARS(5) OR
B7ARS(4) OR B7ARS(3) OR B7ARS(2) OR B7ARS(1) OR B7ARS(0);

--AIS

START8<= B0AIS(19) OR B0AIS(18) OR B0AIS(17) OR B0AIS(16) OR B0AIS(15) OR
B0AIS(14) OR B0AIS(13) OR                 B0AIS(12) OR B0AIS(11) OR B0AIS(10) OR
B0AIS(9) OR B0AIS(8) OR B0AIS(7) OR B0AIS(6) OR              B0AIS(5) OR
B0AIS(4) OR B0AIS(3) OR B0AIS(2) OR B0AIS(1) OR B0AIS(0);

START9<= B1AIS(19) OR B1AIS(18) OR B1AIS(17) OR B1AIS(16) OR B1AIS(15) OR
B1AIS(14) OR B1AIS(13) OR                 B1AIS(12) OR B1AIS(11) OR B1AIS(10) OR
B1AIS(9) OR B1AIS(8) OR B1AIS(7) OR B1AIS(6) OR              B1AIS(5) OR
B1AIS(4) OR B1AIS(3) OR B1AIS(2) OR B1AIS(1) OR B1AIS(0);
```

```
START10 <= B2AIS(19) OR B2AIS(18) OR B2AIS(17) OR B2AIS(16) OR B2AIS(15) OR
B2AIS(14) OR B2AIS(13)        OR B2AIS(12) OR B2AIS(11) OR B2AIS(11) OR B2AIS(9)
OR B2AIS(8) OR B2AIS(7) OR B2AIS(6) OR           B2AIS(5) OR B2AIS(4) OR
B2AIS(3) OR B2AIS(2) OR B2AIS(1) OR B2AIS(0);

START11<= B3AIS(19) OR B3AIS(18) OR B3AIS(17) OR B3AIS(16) OR B3AIS(15) OR
B3AIS(14) OR B3AIS(13)              OR B3AIS(12) OR B3AIS(11) OR B3AIS(10) OR
B3AIS(9) OR B3AIS(8) OR B3AIS(7) OR B3AIS(6) OR              B3AIS(5) OR
B3AIS(4) OR B3AIS(3) OR B3AIS(2) OR B3AIS(1) OR B3AIS(0);

START12<= B4AIS(19) OR B4AIS(18) OR B4AIS(17) OR B4AIS(16) OR B4AIS(15) OR
B4AIS(14) OR B4AIS(13)              OR B4AIS(12) OR B4AIS(11) OR B4AIS(10) OR
B4AIS(9) OR B4AIS(8) OR B4AIS(7) OR B4AIS(6) OR              B4AIS(5) OR
B4AIS(4) OR B4AIS(3) OR B4AIS(2) OR B4AIS(1) OR B4AIS(0);

START13<= B5AIS(19) OR B5AIS(18) OR B5AIS(17) OR B5AIS(16) OR B5AIS(15) OR
B5AIS(14) OR B5AIS(13)              OR B5AIS(12) OR B5AIS(11) OR B5AIS(10) OR
B5AIS(9) OR B5AIS(8) OR B5AIS(7) OR B5AIS(6) OR              B5AIS(5) OR
B5AIS(4) OR B5AIS(3) OR B5AIS(2) OR B5AIS(1) OR B5AIS(0);


START14<= B6AIS(19) OR B6AIS(18) OR B6AIS(17) OR B6AIS(16) OR B6AIS(15) OR
B6AIS(14) OR B6AIS(13)              OR B6AIS(12) OR B6AIS(11) OR B6AIS(10) OR
B6AIS(9) OR B6AIS(8) OR B6AIS(7) OR B6AIS(6) OR              B6AIS(5) OR
B6AIS(4) OR B6AIS(3) OR B6AIS(2) OR B6AIS(1) OR B6AIS(0);

START15 <= B7AIS(19) OR B7AIS(18) OR B7AIS(17) OR B7AIS(16) OR B7AIS(15) OR
B7AIS(14) OR B7AIS(13)              OR B7AIS(12) OR B7AIS(11) OR B7AIS(10) OR
B7AIS(9) OR B7AIS(8) OR B7AIS(7) OR B7AIS(6) OR              B7AIS(5) OR
B7AIS(4) OR B7AIS(3) OR B7AIS(2) OR B7AIS(1) OR B7AIS(0);

-- BIS

START16 <= B0BIS(19) OR B0BIS(18) OR B0BIS(17) OR B0BIS(16) OR B0BIS(15) OR
B0BIS(14) OR B0BIS(13)              OR B0BIS(12) OR B0BIS(11) OR B0BIS(10) OR
B0BIS(9) OR B0BIS(8) OR B0BIS(7) OR B0BIS(6) OR              B0BIS(5) OR
B0BIS(4) OR B0BIS(3) OR B0BIS(2) OR B0BIS(1) OR B0BIS(0);

START17<= B1BIS(19) OR B1BIS(18) OR B1BIS(17) OR B1BIS(16) OR B1BIS(15) OR
B1BIS(14) OR B1BIS(13)              OR B1BIS(12) OR B1BIS(11) OR B1BIS(10) OR
B1BIS(9) OR B1BIS(8) OR B1BIS(7) OR B1BIS(6) OR              B1BIS(5) OR
B1BIS(4) OR B1BIS(3) OR B1BIS(2) OR B1BIS(1) OR B1BIS(0);

START18<= B2BIS(19) OR B2BIS(18) OR B2BIS(17) OR B2BIS(16) OR B2BIS(15) OR
B2BIS(14) OR B2BIS(13)              OR B2BIS(12) OR B2BIS(11) OR B2BIS(10) OR
B2BIS(9) OR B2BIS(8) OR B2BIS(7) OR B2BIS(6) OR              B2BIS(5) OR
B2BIS(4) OR B2BIS(3) OR B2BIS(2) OR B2BIS(1) OR B2BIS(0);

START19<= B3BIS(19) OR B3BIS(18) OR B3BIS(17) OR B3BIS(16) OR B3BIS(15) OR
B3BIS(14) OR B3BIS(13)              OR B3BIS(12) OR B3BIS(11) OR B3BIS(10) OR
B3BIS(9) OR B3BIS(8) OR B3BIS(7) OR B3BIS(6) OR              B3BIS(5) OR
B3BIS(4) OR B3BIS(3) OR B3BIS(2) OR B3BIS(1) OR B3BIS(0);

START20<= B4BIS(19) OR B4BIS(18) OR B4BIS(17) OR B4BIS(16) OR B4BIS(15) OR
B4BIS(14) OR B4BIS(13)              OR B4BIS(12) OR B4BIS(11) OR B4BIS(10) OR
B4BIS(9) OR B4BIS(8) OR B4BIS(7) OR B4BIS(6) OR              B4BIS(5) OR
B4BIS(4) OR B4BIS(3) OR B4BIS(2) OR B4BIS(1) OR B4BIS(0);

START21<= B5BIS(19) OR B5BIS(18) OR B5BIS(17) OR B5BIS(16) OR B5BIS(15) OR
B5BIS(14) OR B5BIS(13)              OR B5BIS(12) OR B5BIS(11) OR B5BIS(10) OR
B5BIS(9) OR B5BIS(8) OR B5BIS(7) OR B5BIS(6) OR              B5BIS(5) OR
B5BIS(4) OR B5BIS(3) OR B5BIS(2) OR B5BIS(1) OR B5BIS(0);
```

```
START22<= B6BIS(19) OR B6BIS(18) OR B6BIS(17) OR B6BIS(16) OR B6BIS(15) OR
B6BIS(14) OR B6BIS(13)              OR B6BIS(12) OR B6BIS(11) OR B6BIS(10) OR
B6BIS(9) OR B6BIS(8) OR B6BIS(7) OR B6BIS(6) OR              B6BIS(5) OR
B6BIS(4) OR B6BIS(3) OR B6BIS(2) OR B6BIS(1) OR B6BIS(0);

START23<= B7BIS(19) OR B7BIS(18) OR B7BIS(17) OR B7BIS(16) OR B7BIS(15) OR
B7BIS(14) OR B7BIS(13)              OR B7BIS(12) OR B7BIS(11) OR B7BIS(10) OR
B7BIS(9) OR B7BIS(8) OR B7BIS(7) OR B7BIS(6) OR              B7BIS(5) OR
B7BIS(4) OR B7BIS(3) OR B7BIS(2) OR B7BIS(1) OR B7BIS(0);


--BRS

START24<= B0BRS(19) OR B0BRS(18) OR B0BRS(17) OR B0BRS(16) OR B0BRS(15) OR
B0BRS(14) OR B0BRS(13)              OR B0BRS(12) OR B0BRS(11) OR B0BRS(10) OR
B0BRS(9) OR B0BRS(8) OR B0BRS(7) OR B0BRS(6) OR              B0BRS(5) OR
B0BRS(4) OR B0BRS(3) OR B0BRS(2) OR B0BRS(1) OR B0BRS(0);

START25<= B1BRS(19) OR B1BRS(18) OR B1BRS(17) OR B1BRS(16) OR B1BRS(15) OR
B1BRS(14) OR B1BRS(13)              OR B1BRS(12) OR B1BRS(11) OR B1BRS(10) OR
B1BRS(9) OR B1BRS(8) OR B1BRS(7) OR B1BRS(6) OR              B1BRS(5) OR
B1BRS(4) OR B1BRS(3) OR B1BRS(2) OR B1BRS(1) OR B1BRS(0);

START26<= B2BRS(19) OR B2BRS(18) OR B2BRS(17) OR B2BRS(16) OR B2BRS(15) OR
B2BRS(14) OR B2BRS(13)              OR B2BRS(12) OR B2BRS(11) OR B2BRS(10) OR
B2BRS(9) OR B2BRS(8) OR B2BRS(7) OR B2BRS(6) OR              B2BRS(5) OR
B2BRS(4) OR B2BRS(3) OR B2BRS(2) OR B2BRS(1) OR B2BRS(0);

START27<= B3BRS(19) OR B3BRS(18) OR B3BRS(17) OR B3BRS(16) OR B3BRS(15) OR
B3BRS(14) OR B3BRS(13)              OR B3BRS(12) OR B3BRS(11) OR B3BRS(10) OR
B3BRS(9) OR B3BRS(8) OR B3BRS(7) OR B3BRS(6) OR              B3BRS(5) OR
B3BRS(4) OR B3BRS(3) OR B3BRS(2) OR B3BRS(1) OR B3BRS(0);

START28<= B4BRS(19) OR B4BRS(18) OR B4BRS(17) OR B4BRS(16) OR B4BRS(15) OR
B4BRS(14) OR B4BRS(13)              OR B4BRS(12) OR B4BRS(11) OR B4BRS(10) OR
B4BRS(9) OR B4BRS(8) OR B4BRS(7) OR B4BRS(6) OR              B4BRS(5) OR
B4BRS(4) OR B4BRS(3) OR B4BRS(2) OR B4BRS(1) OR B4BRS(0);

START29<= B5BRS(19) OR B5BRS(18) OR B5BRS(17) OR B5BRS(16) OR B5BRS(15) OR
B5BRS(14) OR B5BRS(13)              OR B5BRS(12) OR B5BRS(11) OR B5BRS(10) OR
B5BRS(9) OR B5BRS(8) OR B5BRS(7) OR B5BRS(6) OR              B5BRS(5) OR
B5BRS(4) OR B5BRS(3) OR B5BRS(2) OR B5BRS(1) OR B5BRS(0);

START30<= B6BRS(19) OR B6BRS(18) OR B6BRS(17) OR B6BRS(16) OR B6BRS(15) OR
B6BRS(14) OR B6BRS(13)              OR B6BRS(12) OR B6BRS(11) OR B6BRS(10) OR
B6BRS(9) OR B6BRS(8) OR B6BRS(7) OR B6BRS(6) OR              B6BRS(5) OR
B6BRS(4) OR B6BRS(3) OR B6BRS(2) OR B6BRS(1) OR B6BRS(0);

START31<= B7BRS(19) OR B7BRS(18) OR B7BRS(17) OR B7BRS(16) OR B7BRS(15) OR
B7BRS(14) OR B7BRS(13)              OR B7BRS(12) OR B7BRS(11) OR B7BRS(10) OR
B7BRS(9) OR B7BRS(8) OR B7BRS(7) OR B7BRS(6) OR              B7BRS(5) OR
B7BRS(4) OR B7BRS(3) OR B7BRS(2) OR B7BRS(1) OR B7BRS(0);




START<=  START0 OR START1 OR START2 OR START3 OR START4 OR START5 OR START6 OR
START7 OR START8 OR
                 START9 OR START10 OR START11 OR START12 OR START13 OR START14
OR START15 OR
```

```
                    START16 OR START17 OR START18 OR START19 OR START20 OR START21
OR START22 OR START23 OR
                    START24 OR START25 OR START26 OR START27 OR START28 OR START29
OR START30 OR START31;
      end behav;
```

FLIPFLOP_STATUS.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY FlipFlop_status IS

PORT (D, CLK,set,enable,RESET : IN std_logic;
       Q: OUT std_logic);
END  FlipFlop_status;

ARCHITECTURE behav OF FlipFlop_status IS
signal new_d : std_logic;
BEGIN
new_d<= NOT(D);
PROCESS(CLK,set,enable,RESET)

BEGIN

            IF (CLK'EVENT AND CLK = '1') THEN
            IF (enable = '1') THEN
            Q<= new_d;

            ELSE IF (set='1') then
            Q<='1';

            ELSE IF (RESET = '1') THEN
            Q <= '0';
            END IF;
            END IF;
            END IF;
            END IF;

            END PROCESS;

      END behav;
```

REG_STATUS.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY REG_STATUS IS

PORT ( CLK,set,enable0,enable1,enable2,enable3,RESET : IN std_logic;
       Q_OUT: OUT std_logic_vector(3 downto 0));
END  REG_STATUS;

ARCHITECTURE BEHAV OF REG_STATUS IS

COMPONENT FlipFlop_status IS
PORT (D, CLK,set,enable,RESET : IN std_logic;
       Q: OUT std_logic);
END  COMPONENT;


BEGIN

ff0: FlipFlop_status PORT MAP(CLK=>CLK,set=>set,enable=>enable0,Q=>Q_OUT(0),D=>
enable0,RESET=>RESET);
ff1: FlipFlop_status PORT MAP(CLK=>CLK,set=>set,enable=>enable1,Q=>Q_OUT(1),D=>
enable1,RESET=>RESET);
ff2: FlipFlop_status PORT MAP(CLK=>CLK,set=>set,enable=>enable2,Q=>Q_OUT(2),D=>
enable2,RESET=>RESET);
ff3: FlipFlop_status PORT MAP(CLK=>CLK,set=>set,enable=>enable3,Q=>Q_OUT(3),D=>
enable3,RESET=>RESET);


      end behav;
```

MUX_4_RF.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY MUX_4_RF IS
PORT( AR,AI,BR,BI: IN SIGNED(19 downto 0);
            SEL : IN std_logic_vector(1 downto 0);
            OUTPUT: OUT SIGNED(19 downto 0));
END MUX_4_RF;

ARCHITECTURE behav OF MUX_4_RF IS

BEGIN
with SEL  select
```

```
    OUTPUT <= AR when "00",
                          AI when "01",
                          BR when "10",
                          BI when "11",

          "00000000000000000000"  when others;
      end Behav;
```

MUX_20TO1_RF.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY MUX_20to1_RF IS
PORT( A,B: IN SIGNED(19 downto 0);
            SEL : IN std_logic;
            OUTPUT: OUT SIGNED(19 downto 0));
END MUX_20to1_RF;

ARCHITECTURE behav OF MUX_20to1_RF IS

BEGIN
with SEL  select
    OUTPUT <= A when '0',
                          B when '1',

          "00000000000000000000"  when others;
      end Behav;
```

REG_BF_AB.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY REG_BF_AB IS

PORT ( CLK, ENABLE: IN std_logic;
       D: IN SIGNED(19 downto 0); -- va cambiato il formato e conversioni
       Q: OUT SIGNED(19 downto 0));
END REG_BF_AB;

ARCHITECTURE behav OF REG_BF_AB IS

BEGIN

PROCESS(CLK)

BEGIN

            IF (CLK'EVENT AND CLK = '1') THEN
```

```
            IF ( ENABLE = '1') THEN
            Q<= D;
            END IF;
            END IF;
            END PROCESS;

      END behav;
```

REG_BF_W.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY REG_BF_W IS

PORT ( CLK: IN std_logic;
       D: IN SIGNED(19 downto 0); -- va cambiato il formato e conversioni
       Q: OUT SIGNED(19 downto 0));
END REG_BF_W;

ARCHITECTURE behav OF REG_BF_W IS

BEGIN

PROCESS(CLK)

BEGIN

            IF (CLK'EVENT AND CLK = '1') THEN
            Q<= D;
            END IF;
            END PROCESS;

      END behav;
```

RF.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY RF IS
```

```
PORT ( CLK, ENABLE: IN std_logic;
       AI,AR,BI,BR,WR,WI: IN SIGNED(19 downto 0); -- va cambiato il formato e
conversioni
      SEL3,SEL1 : IN std_logic;
            SEL2 : IN std_logic_vector(1 downto 0);
            OUT_MUX1,OUT_MUX2,OUT_MUX3,AI_CTRL,AR_CTRL,BI_CTRL,BR_CTRL: OUT
SIGNED(19 downto 0));
END RF;

ARCHITECTURE BEHAV OF RF IS

COMPONENT REG_BF_W IS
PORT ( CLK: IN std_logic;
       D: IN SIGNED(19 downto 0); -- va cambiato il formato e conversioni
       Q: OUT SIGNED(19 downto 0));
END COMPONENT;

COMPONENT REG_BF_AB IS
PORT ( CLK, ENABLE: IN std_logic;
       D: IN SIGNED(19 downto 0); -- va cambiato il formato e conversioni
       Q: OUT SIGNED(19 downto 0));
END COMPONENT;

COMPONENT MUX_20to1_RF IS
PORT( A,B: IN SIGNED(19 downto 0);
            SEL : IN std_logic;
            OUTPUT: OUT SIGNED(19 downto 0));
END COMPONENT;

COMPONENT MUX_4_RF IS
PORT( AR,AI,BR,BI: IN SIGNED(19 downto 0);
            SEL : IN std_logic_vector(1 downto 0);
            OUTPUT: OUT SIGNED(19 downto 0));
END COMPONENT;

SIGNAL ARQ,AIQ,BRQ,BIQ,WRQ,WIQ : SIGNED(19 downto 0);

BEGIN
MUX_2BA : MUX_4_RF PORT MAP(AR=>ARQ,AI
=>AIQ,BR=>BRQ,BI=>BIQ,SEL=>SEL2,OUTPUT=>OUT_MUX2);
MUX_3W : MUX_20to1_RF PORT MAP (A=>WRQ,B=>WIQ,SEL=>SEL3,OUTPUT=>OUT_MUX3);
MUX_1A : MUX_20to1_RF PORT MAP (A=>ARQ,B=>AIQ,SEL=>SEL1,OUTPUT=>OUT_MUX1);
R_AR : REG_BF_AB PORT MAP (CLK=>CLK,ENABLE=>ENABLE,D=>AR,Q=>ARQ);
R_AI : REG_BF_AB PORT MAP (CLK=>CLK,ENABLE=>ENABLE,D=>AI,Q=>AIQ);
R_BR : REG_BF_AB PORT MAP (CLK=>CLK,ENABLE=>ENABLE,D=>BR,Q=>BRQ);
R_BI : REG_BF_AB PORT MAP (CLK=>CLK,ENABLE=>ENABLE,D=>BI,Q=>BIQ);
R_WR : REG_BF_W  PORT MAP (CLK=>CLK,D=>WR,Q=>WRQ);
R_WI : REG_BF_W  PORT MAP (CLK=>CLK,D=>WI,Q=>WIQ);

AI_CTRL<= AIQ;
AR_CTRL<= ARQ;
BI_CTRL<= BIQ;
BR_CTRL<= BRQ;
      END BEHAV;
```

START_SENSE.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY start_sense IS

PORT (D, CLK,set,enable : IN std_logic;
       SENSE : OUT std_logic);
END  start_sense;

ARCHITECTURE behav OF start_sense IS
signal new_d : std_logic;
BEGIN

new_d<=not(D);
PROCESS(CLK,set,enable)


BEGIN
            IF (CLK'EVENT AND CLK = '1') THEN
            IF (enable = '1') THEN
            SENSE <= new_d;
            ELSE IF (set='1') then
            SENSE <='1';
            END IF;
            END IF;
            END IF;

            END PROCESS;

      END behav;
```

REG_SUM.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY REG_SUM IS

PORT ( CLK: IN std_logic;
       D: IN SIGNED(39 downto 0);
       Q: OUT SIGNED (39 downto 0));
END REG_SUM;

ARCHITECTURE behav OF REG_SUM IS
```

```
BEGIN

PROCESS(CLK)

BEGIN

            IF (CLK'EVENT AND CLK = '1') THEN
            Q<= D;
            END IF;
            END PROCESS;

      END behav;
```

SUB_ADDER.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY SUB_ADDER IS

PORT ( A_S: IN std_logic;
       T1,T2: IN SIGNED(38 downto 0);
       RESULT: OUT SIGNED(39 downto 0));
END SUB_ADDER;

ARCHITECTURE behav OF SUB_ADDER IS

SIGNAL sign_ext_t1,sign_ext_t2: SIGNED(39 DOWNTO 0);

BEGIN

sign_ext_t1(39) <= T1(38);                                          --passo
nel formato Q2.38
sign_ext_t1(38 DOWNTO 0) <= T1(38 DOWNTO 0);

sign_ext_t2(39) <= T2(38);                                          --passo
nel formato Q2.38
sign_ext_t2(38 DOWNTO 0) <= T2(38 DOWNTO 0);

PROCESS(A_S,sign_ext_t1,sign_ext_t2)

BEGIN

            IF (A_S = '0') THEN
            RESULT(39 DOWNTO 0) <= sign_ext_t1+sign_ext_t2;
            END IF;
            IF (A_S ='1') THEN
            RESULT(39 DOWNTO 0) <= sign_ext_t1-(sign_ext_t2);
            END IF;
            END PROCESS;
```

```
      END behav;
```

SUM.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY SUM IS

PORT ( A_S,CLK: IN std_logic;
       T1,T2: IN SIGNED(38 downto 0);
       RESULT: OUT SIGNED(39 downto 0));
END SUM;

ARCHITECTURE behav OF SUM IS


COMPONENT REG_SUM IS
PORT ( CLK: IN std_logic;
       D: IN SIGNED(39 downto 0);
       Q: OUT SIGNED (39 downto 0));
END COMPONENT;

COMPONENT SUB_ADDER IS
PORT ( A_S: IN std_logic;
       T1,T2: IN SIGNED(38 downto 0);
       RESULT: OUT SIGNED(39 downto 0));
END COMPONENT;

SIGNAL partial : SIGNED(39 downto 0);

BEGIN

SOMMATORE: SUB_ADDER PORT MAP(A_S=>A_S,T1=>T1,T2=>T2,RESULT=>PARTIAL);
REG: REG_SUM PORT MAP (CLK=>CLK,D=>partial,Q=>RESULT);

      END behav;
```

OR_32.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY OR_32 IS

PORT ( INPUT: IN std_logic_vector(31 downto 0);
       OUT_XOR_CHECK: OUT std_logic);
END OR_32;

ARCHITECTURE behav OF OR_32 IS

BEGIN

OUT_XOR_CHECK <= INPUT(31) OR INPUT(30) OR INPUT(29) OR INPUT(28) OR INPUT(27)
OR INPUT(26)                                    OR INPUT(25) OR INPUT(24) OR
INPUT(23) OR INPUT(22) OR INPUT(21) OR INPUT(20)
OR INPUT(19) OR INPUT(18)  OR INPUT(17) OR INPUT(16) OR INPUT(15) OR INPUT(14)
OR
                                INPUT(13) OR INPUT(12) OR INPUT(11) OR INPUT(10)
OR INPUT(9) OR INPUT(8) OR                          INPUT(7) OR INPUT(6) OR
INPUT(5) OR INPUT(4) OR INPUT(3) OR INPUT(2)                              OR
INPUT(1) OR INPUT(0);
      END behav;
```

XOR_CHECK.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY XOR_CHECK IS

PORT (
       B0AR,B1AR,B2AR,B3AR,B4AR,B5AR,B6AR,B7AR: IN SIGNED(19 downto 0);
             B0AI,B1AI,B2AI,B3AI,B4AI,B5AI,B6AI,B7AI: IN SIGNED(19 downto 0);
             B0BR,B1BR,B2BR,B3BR,B4BR,B5BR,B6BR,B7BR: IN SIGNED(19 downto 0);
             B0BI,B1BI,B2BI,B3BI,B4BI,B5BI,B6BI,B7BI: IN SIGNED(19 downto 0);


B0AR_OLD,B1AR_OLD,B2AR_OLD,B3AR_OLD,B4AR_OLD,B5AR_OLD,B6AR_OLD,B7AR_OLD:
             IN SIGNED(19 downto 0);

B0AI_OLD,B1AI_OLD,B2AI_OLD,B3AI_OLD,B4AI_OLD,B5AI_OLD,B6AI_OLD,B7AI_OLD:
             IN SIGNED(19 downto 0);

B0BR_OLD,B1BR_OLD,B2BR_OLD,B3BR_OLD,B4BR_OLD,B5BR_OLD,B6BR_OLD,B7BR_OLD:
             IN SIGNED(19 downto 0);
```

```
B0BI_OLD,B1BI_OLD,B2BI_OLD,B3BI_OLD,B4BI_OLD,B5BI_OLD,B6BI_OLD,B7BI_OLD:
              IN SIGNED(19 downto 0);

              OUT_XOR_CHECK: OUT std_logic);

END XOR_CHECK;

ARCHITECTURE behav OF XOR_CHECK IS

COMPONENT XOR_PORT IS
PORT ( IN_NEW,IN_OLD: IN SIGNED(19 downto 0);
       OUT_XOR: OUT std_logic);
END COMPONENT;

COMPONENT OR_32 IS
PORT ( INPUT: IN std_logic_vector(31 downto 0);
       OUT_XOR_CHECK: OUT std_logic);
END COMPONENT;

SIGNAL INPUT_S: std_logic_vector(31 downto 0);
BEGIN

AR_0: XOR_PORT  PORT MAP(IN_NEW=>B0AR,IN_OLD=>B0AR_OLD,OUT_XOR=>INPUT_S(31));
AR_1: XOR_PORT  PORT MAP(IN_NEW=>B1AR,IN_OLD=>B1AR_OLD,OUT_XOR=>INPUT_S(30));
AR_2: XOR_PORT  PORT MAP(IN_NEW=>B2AR,IN_OLD=>B2AR_OLD,OUT_XOR=>INPUT_S(29));
AR_3: XOR_PORT  PORT MAP(IN_NEW=>B3AR,IN_OLD=>B3AR_OLD,OUT_XOR=>INPUT_S(28));
AR_4: XOR_PORT  PORT MAP(IN_NEW=>B4AR,IN_OLD=>B4AR_OLD,OUT_XOR=>INPUT_S(27));
AR_5: XOR_PORT  PORT MAP(IN_NEW=>B5AR,IN_OLD=>B5AR_OLD,OUT_XOR=>INPUT_S(26));
AR_6: XOR_PORT  PORT MAP(IN_NEW=>B6AR,IN_OLD=>B6AR_OLD,OUT_XOR=>INPUT_S(25));
AR_7: XOR_PORT  PORT MAP(IN_NEW=>B7AR,IN_OLD=>B7AR_OLD,OUT_XOR=>INPUT_S(24));

AI_0: XOR_PORT  PORT MAP(IN_NEW=>B0AI,IN_OLD=>B0AI_OLD,OUT_XOR=>INPUT_S(23));
AI_1: XOR_PORT  PORT MAP(IN_NEW=>B1AI,IN_OLD=>B1AI_OLD,OUT_XOR=>INPUT_S(22));
AI_2: XOR_PORT  PORT MAP(IN_NEW=>B2AI,IN_OLD=>B2AI_OLD,OUT_XOR=>INPUT_S(21));
AI_3: XOR_PORT  PORT MAP(IN_NEW=>B3AI,IN_OLD=>B3AI_OLD,OUT_XOR=>INPUT_S(20));
AI_4: XOR_PORT  PORT MAP(IN_NEW=>B4AI,IN_OLD=>B4AI_OLD,OUT_XOR=>INPUT_S(19));
AI_5: XOR_PORT  PORT MAP(IN_NEW=>B5AI,IN_OLD=>B5AI_OLD,OUT_XOR=>INPUT_S(18));
AI_6: XOR_PORT  PORT MAP(IN_NEW=>B6AI,IN_OLD=>B6AI_OLD,OUT_XOR=>INPUT_S(17));
AI_7: XOR_PORT  PORT MAP(IN_NEW=>B7AI,IN_OLD=>B7AI_OLD,OUT_XOR=>INPUT_S(16));

BR_0: XOR_PORT  PORT MAP(IN_NEW=>B0BR,IN_OLD=>B0BR_OLD,OUT_XOR=>INPUT_S(15));
BR_1: XOR_PORT  PORT MAP(IN_NEW=>B1BR,IN_OLD=>B1BR_OLD,OUT_XOR=>INPUT_S(14));
BR_2: XOR_PORT  PORT MAP(IN_NEW=>B2BR,IN_OLD=>B2BR_OLD,OUT_XOR=>INPUT_S(13));
BR_3: XOR_PORT  PORT MAP(IN_NEW=>B3BR,IN_OLD=>B3BR_OLD,OUT_XOR=>INPUT_S(12));
BR_4: XOR_PORT  PORT MAP(IN_NEW=>B4BR,IN_OLD=>B4BR_OLD,OUT_XOR=>INPUT_S(11));
BR_5: XOR_PORT  PORT MAP(IN_NEW=>B5BR,IN_OLD=>B5BR_OLD,OUT_XOR=>INPUT_S(10));
BR_6: XOR_PORT  PORT MAP(IN_NEW=>B6BR,IN_OLD=>B6BR_OLD,OUT_XOR=>INPUT_S(9));
BR_7: XOR_PORT  PORT MAP(IN_NEW=>B7BR,IN_OLD=>B7BR_OLD,OUT_XOR=>INPUT_S(8));

BI_0: XOR_PORT  PORT MAP(IN_NEW=>B0BI,IN_OLD=>B0BI_OLD,OUT_XOR=>INPUT_S(7));
BI_1: XOR_PORT  PORT MAP(IN_NEW=>B1BI,IN_OLD=>B1BI_OLD,OUT_XOR=>INPUT_S(6));
BI_2: XOR_PORT  PORT MAP(IN_NEW=>B2BI,IN_OLD=>B2BI_OLD,OUT_XOR=>INPUT_S(5));
BI_3: XOR_PORT  PORT MAP(IN_NEW=>B3BI,IN_OLD=>B3BI_OLD,OUT_XOR=>INPUT_S(4));
BI_4: XOR_PORT  PORT MAP(IN_NEW=>B4BI,IN_OLD=>B4BI_OLD,OUT_XOR=>INPUT_S(3));
BI_5: XOR_PORT  PORT MAP(IN_NEW=>B5BI,IN_OLD=>B5BI_OLD,OUT_XOR=>INPUT_S(2));
BI_6: XOR_PORT  PORT MAP(IN_NEW=>B6BI,IN_OLD=>B6BI_OLD,OUT_XOR=>INPUT_S(1));
BI_7: XOR_PORT  PORT MAP(IN_NEW=>B7BI,IN_OLD=>B7BI_OLD,OUT_XOR=>INPUT_S(0));

OR_LEVEL: OR_32 PORT MAP(INPUT=>INPUT_S,OUT_XOR_CHECK=>OUT_XOR_CHECK);

END BEHAV;
```

XOR_PORT.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY XOR_PORT IS

PORT ( IN_NEW,IN_OLD: IN SIGNED(19 downto 0);
       OUT_XOR: OUT std_logic);
END XOR_PORT;

ARCHITECTURE behav OF XOR_PORT IS

SIGNAL NEW_S,OLD_S : std_logic_vector(19 downto 0);
SIGNAL OUT_19 : std_logic_vector(19 downto 0);
BEGIN


NEW_S<= std_logic_vector(IN_NEW);
OLD_S<= std_logic_vector(IN_OLD);

OUT_19(19) <= NEW_S(19) XOR OLD_S(19);
OUT_19(18) <= NEW_S(18) XOR OLD_S(18);
OUT_19(17) <= NEW_S(17) XOR OLD_S(17);
OUT_19(16) <= NEW_S(16) XOR OLD_S(16);
OUT_19(15) <= NEW_S(15) XOR OLD_S(15);
OUT_19(14) <= NEW_S(14) XOR OLD_S(14);
OUT_19(13) <= NEW_S(13) XOR OLD_S(13);
OUT_19(12) <= NEW_S(12) XOR OLD_S(12);
OUT_19(11) <= NEW_S(11) XOR OLD_S(11);
OUT_19(10) <= NEW_S(10) XOR OLD_S(10);
OUT_19(9) <= NEW_S(9) XOR OLD_S(9);
OUT_19(8) <= NEW_S(8) XOR OLD_S(8);
OUT_19(7) <= NEW_S(7) XOR OLD_S(7);
OUT_19(6) <= NEW_S(6) XOR OLD_S(6);
OUT_19(5) <= NEW_S(5) XOR OLD_S(5);
OUT_19(4) <= NEW_S(4) XOR OLD_S(4);
OUT_19(3) <= NEW_S(3) XOR OLD_S(3);
OUT_19(2) <= NEW_S(2) XOR OLD_S(2);
OUT_19(1) <= NEW_S(1) XOR OLD_S(1);
OUT_19(0) <= NEW_S(0) XOR OLD_S(0);

OUT_XOR<= OUT_19(19) OR OUT_19(18) OR OUT_19(17) OR OUT_19(16) OR OUT_19(15) OR
OUT_19(14) OR                 OUT_19(13) OR OUT_19(12) OR OUT_19(11) OR
OUT_19(10) OR OUT_19(9) OR OUT_19(8) OR
                  OUT_19(7) OR OUT_19(6)  OR OUT_19(5) OR OUT_19(4) OR OUT_19(3)
OR OUT_19(2) OR
                  OUT_19(1) OR OUT_19(0);


END behav;
```

FF_SEQ.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY FF_SEQ IS

PORT (CLK,D,SET,RESET,ENABLE: IN STD_LOGIC;
            Q: OUT STD_LOGIC);
END FF_SEQ;

ARCHITECTURE behav OF FF_SEQ IS

BEGIN

PROCESS(CLK,SET,RESET,ENABLE)
BEGIN

      IF (CLK'EVENT AND CLK = '1') THEN
            IF (ENABLE = '1') THEN
            Q <= D;
            ELSE IF (SET = '1') THEN
            Q <= '1';
            ELSE IF (RESET = '1') THEN
            Q <= '0';
            END IF;
            END IF;
            END IF;
            END IF;

END PROCESS;

      END behav;
```

GUARD.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;
```

```
ENTITY GUARD IS

PORT ( C0AR,C1AR,C2AR,C3AR,C4AR,C5AR,C6AR,C7AR: IN SIGNED(19 downto 0);
             C0AI,C1AI,C2AI,C3AI,C4AI,C5AI,C6AI,C7AI: IN SIGNED(19 downto 0);
             C0BR,C1BR,C2BR,C3BR,C4BR,C5BR,C6BR,C7BR: IN SIGNED(19 downto 0);
             C0BI,C1BI,C2BI,C3BI,C4BI,C5BI,C6BI,C7BI: IN SIGNED(19 downto 0);


C0AR_GUARD,C1AR_GUARD,C2AR_GUARD,C3AR_GUARD,C4AR_GUARD,C5AR_GUARD,C6AR_GUARD,C7A
R_GUARD:
             OUT SIGNED(19 downto 0);

C0AI_GUARD,C1AI_GUARD,C2AI_GUARD,C3AI_GUARD,C4AI_GUARD,C5AI_GUARD,C6AI_GUARD,C7A
I_GUARD:
             OUT SIGNED(19 downto 0);

C0BR_GUARD,C1BR_GUARD,C2BR_GUARD,C3BR_GUARD,C4BR_GUARD,C5BR_GUARD,C6BR_GUARD,C7B
R_GUARD:
             OUT SIGNED(19 downto 0);

C0BI_GUARD,C1BI_GUARD,C2BI_GUARD,C3BI_GUARD,C4BI_GUARD,C5BI_GUARD,C6BI_GUARD,C7B
I_GUARD:
             OUT SIGNED(19 downto 0));
END GUARD;

ARCHITECTURE behav OF GUARD IS

BEGIN

C0AR_GUARD(19) <= C0AR(19); C0AR_GUARD(18 DOWNTO 0) <= C0AR(19 DOWNTO 1);
C1AR_GUARD(19) <= C1AR(19); C1AR_GUARD(18 DOWNTO 0) <= C1AR(19 DOWNTO 1);
C2AR_GUARD(19) <= C2AR(19); C2AR_GUARD(18 DOWNTO 0) <= C2AR(19 DOWNTO 1);
C3AR_GUARD(19) <= C3AR(19); C3AR_GUARD(18 DOWNTO 0) <= C3AR(19 DOWNTO 1);
C4AR_GUARD(19) <= C4AR(19); C4AR_GUARD(18 DOWNTO 0) <= C4AR(19 DOWNTO 1);
C5AR_GUARD(19) <= C5AR(19); C5AR_GUARD(18 DOWNTO 0) <= C5AR(19 DOWNTO 1);
C6AR_GUARD(19) <= C6AR(19); C6AR_GUARD(18 DOWNTO 0) <= C6AR(19 DOWNTO 1);
C7AR_GUARD(19) <= C7AR(19); C7AR_GUARD(18 DOWNTO 0) <= C7AR(19 DOWNTO 1);

C0AI_GUARD(19) <= C0AI(19); C0AI_GUARD(18 DOWNTO 0) <= C0AI(19 DOWNTO 1);
C1AI_GUARD(19) <= C1AI(19); C1AI_GUARD(18 DOWNTO 0) <= C1AI(19 DOWNTO 1);
C2AI_GUARD(19) <= C2AI(19); C2AI_GUARD(18 DOWNTO 0) <= C2AI(19 DOWNTO 1);
C3AI_GUARD(19) <= C3AI(19); C3AI_GUARD(18 DOWNTO 0) <= C3AI(19 DOWNTO 1);
C4AI_GUARD(19) <= C4AI(19); C4AI_GUARD(18 DOWNTO 0) <= C4AI(19 DOWNTO 1);
C5AI_GUARD(19) <= C5AI(19); C5AI_GUARD(18 DOWNTO 0) <= C5AI(19 DOWNTO 1);
C6AI_GUARD(19) <= C6AI(19); C6AI_GUARD(18 DOWNTO 0) <= C6AI(19 DOWNTO 1);
C7AI_GUARD(19) <= C7AI(19); C7AI_GUARD(18 DOWNTO 0) <= C7AI(19 DOWNTO 1);

C0BR_GUARD(19) <= C0BR(19); C0BR_GUARD(18 DOWNTO 0) <= C0BR(19 DOWNTO 1);
C1BR_GUARD(19) <= C1BR(19); C1BR_GUARD(18 DOWNTO 0) <= C1BR(19 DOWNTO 1);
C2BR_GUARD(19) <= C2BR(19); C2BR_GUARD(18 DOWNTO 0) <= C2BR(19 DOWNTO 1);
C3BR_GUARD(19) <= C3BR(19); C3BR_GUARD(18 DOWNTO 0) <= C3BR(19 DOWNTO 1);
C4BR_GUARD(19) <= C4BR(19); C4BR_GUARD(18 DOWNTO 0) <= C4BR(19 DOWNTO 1);
C5BR_GUARD(19) <= C5BR(19); C5BR_GUARD(18 DOWNTO 0) <= C5BR(19 DOWNTO 1);
C6BR_GUARD(19) <= C6BR(19); C6BR_GUARD(18 DOWNTO 0) <= C6BR(19 DOWNTO 1);
C7BR_GUARD(19) <= C7BR(19); C7BR_GUARD(18 DOWNTO 0) <= C7BR(19 DOWNTO 1);

C0BI_GUARD(19) <= C0BI(19); C0BI_GUARD(18 DOWNTO 0) <= C0BI(19 DOWNTO 1);
C1BI_GUARD(19) <= C1BI(19); C1BI_GUARD(18 DOWNTO 0) <= C1BI(19 DOWNTO 1);
C2BI_GUARD(19) <= C2BI(19); C2BI_GUARD(18 DOWNTO 0) <= C2BI(19 DOWNTO 1);
C3BI_GUARD(19) <= C3BI(19); C3BI_GUARD(18 DOWNTO 0) <= C3BI(19 DOWNTO 1);
C4BI_GUARD(19) <= C4BI(19); C4BI_GUARD(18 DOWNTO 0) <= C4BI(19 DOWNTO 1);
C5BI_GUARD(19) <= C5BI(19); C5BI_GUARD(18 DOWNTO 0) <= C5BI(19 DOWNTO 1);
```

```
C6BI_GUARD(19) <= C6BI(19); C6BI_GUARD(18 DOWNTO 0) <= C6BI(19 DOWNTO 1);
C7BI_GUARD(19) <= C7BI(19); C7BI_GUARD(18 DOWNTO 0) <= C7BI(19 DOWNTO 1);



      END behav;
```

R_STONE_0.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY R_STONE_0 IS

PORT(
      OUTPUT: OUT SIGNED(19 DOWNTO 0));
END R_STONE_0;


ARCHITECTURE BEHAV OF R_STONE_0 IS
SIGNAL VALORE :SIGNED (19 DOWNTO 0);
BEGIN
VALORE<= "00000000000000000000";
OUTPUT<=VALORE;
      END BEHAV;
```

R_STONE_1.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY R_STONE_1 IS

PORT(
      OUTPUT_P,OUTPUT_N: OUT SIGNED(19 DOWNTO 0));
END R_STONE_1;


ARCHITECTURE BEHAV OF R_STONE_1 IS
SIGNAL VALORE :SIGNED (19 DOWNTO 0);
BEGIN
VALORE<= "01111111111111111111";
OUTPUT_P<=VALORE;
```

```
OUTPUT_N<= NOT VALORE;
      END BEHAV;
```

R_STONE_3.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY R_STONE_3 IS

PORT(
      OUTPUT_P,OUTPUT_N: OUT SIGNED(19 DOWNTO 0));
END R_STONE_3;


ARCHITECTURE BEHAV OF R_STONE_3 IS
SIGNAL VALORE :SIGNED (19 DOWNTO 0);
BEGIN
--VALORE<= "00110000111100000000";
VALORE<= "00110000111110111100";
OUTPUT_P<=VALORE;
OUTPUT_N(19 downto 3)<= NOT VALORE(19 downto 3);
OUTPUT_N(2 downto 0)<= VALORE(2 DOWNTO 0);
      END BEHAV;
```

R_STONE_7.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY R_STONE_7 IS

PORT(
      OUTPUT_P,OUTPUT_N: OUT SIGNED(19 DOWNTO 0));
END R_STONE_7;


ARCHITECTURE BEHAV OF R_STONE_7 IS
SIGNAL VALORE :SIGNED (19 DOWNTO 0);
BEGIN  --
--VALORE<= "01011010100000000000";
VALORE<= "01011010100000101000";
OUTPUT_P<=VALORE;
```

```
OUTPUT_N(19 downto 4)<= NOT VALORE(19 downto 4);
OUTPUT_N(3 downto 0)<= VALORE(3 DOWNTO 0);
      END BEHAV;
```

R_STONE_9.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY R_STONE_9 IS

PORT(
      OUTPUT_P,OUTPUT_N: OUT SIGNED(19 DOWNTO 0));
END R_STONE_9;


ARCHITECTURE BEHAV OF R_STONE_9 IS
SIGNAL VALORE :SIGNED (19 DOWNTO 0);
BEGIN
--VALORE<= "01110110010000000000";
VALORE<= "01110110010000011011";
OUTPUT_P<=VALORE;
OUTPUT_N(19 downto 1)<= NOT VALORE(19 downto 1);
OUTPUT_N(0)<= VALORE(0);
      END BEHAV;
```

REG_SEQ.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY REG_SEQ IS

PORT (CLK,D0,D1,D2,D3,SET,RESET,EN0,EN1,EN2,EN3: IN STD_LOGIC;
            Q0,Q1,Q2,Q3: OUT STD_LOGIC);
END REG_SEQ;

ARCHITECTURE behav OF REG_SEQ IS
```

```
COMPONENT FF_SEQ IS
PORT (CLK,D,SET,RESET,ENABLE: IN STD_LOGIC;
            Q: OUT STD_LOGIC);
END COMPONENT;


BEGIN

FF0: FF_SEQ PORT MAP(CLK=>CLK,D=>D0,SET=>SET,RESET=>RESET,ENABLE=>EN0,Q=>Q0);
FF1: FF_SEQ PORT MAP(CLK=>CLK,D=>D1,SET=>SET,RESET=>RESET,ENABLE=>EN1,Q=>Q1);
FF2: FF_SEQ PORT MAP(CLK=>CLK,D=>D2,SET=>SET,RESET=>RESET,ENABLE=>EN2,Q=>Q2);
FF3: FF_SEQ PORT MAP(CLK=>CLK,D=>D3,SET=>SET,RESET=>RESET,ENABLE=>EN3,Q=>Q3);


      END behav;
```

FFT_16.VHD

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

ENTITY FFT_16 IS

PORT ( CLK,RESET: IN STD_LOGIC;
             DONE,READY: OUT STD_LOGIC;
       C0AR,C1AR,C2AR,C3AR,C4AR,C5AR,C6AR,C7AR: IN SIGNED(19 downto 0);
             C0AI,C1AI,C2AI,C3AI,C4AI,C5AI,C6AI,C7AI: IN SIGNED(19 downto 0);
             C0BR,C1BR,C2BR,C3BR,C4BR,C5BR,C6BR,C7BR: IN SIGNED(19 downto 0);
             C0BI,C1BI,C2BI,C3BI,C4BI,C5BI,C6BI,C7BI: IN SIGNED(19 downto 0);


C0AR_OUT,C1AR_OUT,C2AR_OUT,C3AR_OUT,C4AR_OUT,C5AR_OUT,C6AR_OUT,C7AR_OUT:
             OUT SIGNED(19 downto 0);

C0AI_OUT,C1AI_OUT,C2AI_OUT,C3AI_OUT,C4AI_OUT,C5AI_OUT,C6AI_OUT,C7AI_OUT:
             OUT SIGNED(19 downto 0);

C0BR_OUT,C1BR_OUT,C2BR_OUT,C3BR_OUT,C4BR_OUT,C5BR_OUT,C6BR_OUT,C7BR_OUT:
             OUT SIGNED(19 downto 0);

C0BI_OUT,C1BI_OUT,C2BI_OUT,C3BI_OUT,C4BI_OUT,C5BI_OUT,C6BI_OUT,C7BI_OUT:
             OUT SIGNED(19 downto 0));
```

```
END FFT_16;

ARCHITECTURE behav OF FFT_16 IS

COMPONENT BF IS
PORT ( CLK, ENABLE: IN std_logic;
       AI,AR,BI,BR,WR,WI: IN SIGNED(19 downto 0); -- va cambiato il formato e
conversioni
      SEL_INV,SEL3,SEL1,SELSUM,C,A_S,EN_REGR : IN std_logic;
            SEL2: IN std_logic_vector(1 downto 0);
            EN_REGO: IN std_logic_vector(2 downto 0);
            A1R,A1I,B1R,B1I,AI_CTRL,AR_CTRL,BI_CTRL,BR_CTRL: OUT SIGNED(19
downto 0));
END COMPONENT;

COMPONENT OR_PORT IS
PORT ( B0AR,B1AR,B2AR,B3AR,B4AR,B5AR,B6AR,B7AR: IN SIGNED(19 downto 0);
             B0AI,B1AI,B2AI,B3AI,B4AI,B5AI,B6AI,B7AI: IN SIGNED(19 downto 0);
             B0BR,B1BR,B2BR,B3BR,B4BR,B5BR,B6BR,B7BR: IN SIGNED(19 downto 0);
             B0BI,B1BI,B2BI,B3BI,B4BI,B5BI,B6BI,B7BI: IN SIGNED(19 downto 0);
       START: OUT std_logic);
END COMPONENT;

COMPONENT XOR_CHECK IS
PORT ( B0AR,B1AR,B2AR,B3AR,B4AR,B5AR,B6AR,B7AR: IN SIGNED(19 downto 0);
             B0AI,B1AI,B2AI,B3AI,B4AI,B5AI,B6AI,B7AI: IN SIGNED(19 downto 0);
             B0BR,B1BR,B2BR,B3BR,B4BR,B5BR,B6BR,B7BR: IN SIGNED(19 downto 0);
             B0BI,B1BI,B2BI,B3BI,B4BI,B5BI,B6BI,B7BI: IN SIGNED(19 downto 0);


B0AR_OLD,B1AR_OLD,B2AR_OLD,B3AR_OLD,B4AR_OLD,B5AR_OLD,B6AR_OLD,B7AR_OLD:
             IN SIGNED(19 downto 0);

B0AI_OLD,B1AI_OLD,B2AI_OLD,B3AI_OLD,B4AI_OLD,B5AI_OLD,B6AI_OLD,B7AI_OLD:
             IN SIGNED(19 downto 0);

B0BR_OLD,B1BR_OLD,B2BR_OLD,B3BR_OLD,B4BR_OLD,B5BR_OLD,B6BR_OLD,B7BR_OLD:
             IN SIGNED(19 downto 0);

B0BI_OLD,B1BI_OLD,B2BI_OLD,B3BI_OLD,B4BI_OLD,B5BI_OLD,B6BI_OLD,B7BI_OLD:
             IN SIGNED(19 downto 0);

             OUT_XOR_CHECK: OUT std_logic);
END COMPONENT;

COMPONENT R_STONE_0 IS
PORT(
      OUTPUT: OUT SIGNED(19 DOWNTO 0));
END COMPONENT;

COMPONENT R_STONE_1 IS
PORT(
      OUTPUT_P,OUTPUT_N: OUT SIGNED(19 DOWNTO 0));
END COMPONENT;

COMPONENT R_STONE_3 IS
PORT(
      OUTPUT_P,OUTPUT_N: OUT SIGNED(19 DOWNTO 0));
END COMPONENT;

COMPONENT R_STONE_7 IS
PORT(
```

```
      OUTPUT_P,OUTPUT_N: OUT SIGNED(19 DOWNTO 0));
END COMPONENT;

COMPONENT R_STONE_9 IS
PORT(
      OUTPUT_P,OUTPUT_N: OUT SIGNED(19 DOWNTO 0));
END COMPONENT;

COMPONENT CU_BF IS
PORT( LOAD,SEQ,CLK,RESET: IN std_logic;
            CTRL_OUT: OUT std_logic_vector(14 downto 0));
END COMPONENT;

COMPONENT CU_TOP IS
PORT( START,PROGRESS,FREE_M,END_BF,SEQ,CLK,RESET: IN std_logic;
            CTRL_TOP_OUT: OUT std_logic_vector(5 downto 0));
END COMPONENT;

COMPONENT start_sense IS
PORT (D, CLK,set,enable : IN std_logic;
       SENSE : OUT std_logic);
END  COMPONENT;

COMPONENT REG_STATUS IS
PORT ( CLK,set,enable0,enable1,enable2,enable3,RESET : IN std_logic;
       Q_OUT: OUT std_logic_vector (3 downto 0));
END  COMPONENT;

COMPONENT REG_SEQ IS
PORT (CLK,D0,D1,D2,D3,SET,RESET,EN0,EN1,EN2,EN3: IN STD_LOGIC;
            Q0,Q1,Q2,Q3: OUT STD_LOGIC);
END COMPONENT;

COMPONENT GUARD IS
PORT ( C0AR,C1AR,C2AR,C3AR,C4AR,C5AR,C6AR,C7AR: IN SIGNED(19 downto 0);
             C0AI,C1AI,C2AI,C3AI,C4AI,C5AI,C6AI,C7AI: IN SIGNED(19 downto 0);
             C0BR,C1BR,C2BR,C3BR,C4BR,C5BR,C6BR,C7BR: IN SIGNED(19 downto 0);
             C0BI,C1BI,C2BI,C3BI,C4BI,C5BI,C6BI,C7BI: IN SIGNED(19 downto 0);


C0AR_GUARD,C1AR_GUARD,C2AR_GUARD,C3AR_GUARD,C4AR_GUARD,C5AR_GUARD,C6AR_GUARD,C7A
R_GUARD:
             OUT SIGNED(19 downto 0);

C0AI_GUARD,C1AI_GUARD,C2AI_GUARD,C3AI_GUARD,C4AI_GUARD,C5AI_GUARD,C6AI_GUARD,C7A
I_GUARD:
             OUT SIGNED(19 downto 0);

C0BR_GUARD,C1BR_GUARD,C2BR_GUARD,C3BR_GUARD,C4BR_GUARD,C5BR_GUARD,C6BR_GUARD,C7B
R_GUARD:
             OUT SIGNED(19 downto 0);

C0BI_GUARD,C1BI_GUARD,C2BI_GUARD,C3BI_GUARD,C4BI_GUARD,C5BI_GUARD,C6BI_GUARD,C7B
I_GUARD:
             OUT SIGNED(19 downto 0));
END COMPONENT;

-- SEGNALI CTRL TOP OUT
SIGNAL FF_VALUE_S,EN_FF_S: STD_LOGIC;
SIGNAL RESET_OUT: STD_LOGIC;
SIGNAL PROGRESS_S: STD_LOGIC_VECTOR (3 downto 0);
```

```
SIGNAL EN_REGS0,EN_REGS1,EN_REGS2,EN_REGS3,FREE_M_S: STD_LOGIC;

SIGNAL NEW_PROGRESS_S : STD_LOGIC;


-- SEGNALI PER LA CU DELLE BF USCITA
SIGNAL SEL3_S0,SEL1_S0,SELSUM_S0,C_S0,A_S_S0,EN_REGR_S0,ENABLE_S0,SEL_INV_S0 :
STD_LOGIC;
SIGNAL SEL3_S1,SEL1_S1,SELSUM_S1,C_S1,A_S_S1,EN_REGR_S1,ENABLE_S1,SEL_INV_S1 :
STD_LOGIC;
SIGNAL SEL3_S2,SEL1_S2,SELSUM_S2,C_S2,A_S_S2,EN_REGR_S2,ENABLE_S2,SEL_INV_S2 :
STD_LOGIC;
SIGNAL SEL3_S3,SEL1_S3,SELSUM_S3,C_S3,A_S_S3,EN_REGR_S3,ENABLE_S3,SEL_INV_S3 :
STD_LOGIC;

SIGNAL SEL2_S0: STD_LOGIC_VECTOR(1 DOWNTO 0);
SIGNAL SEL2_S1: STD_LOGIC_VECTOR(1 DOWNTO 0);
SIGNAL SEL2_S2: STD_LOGIC_VECTOR(1 DOWNTO 0);
SIGNAL SEL2_S3: STD_LOGIC_VECTOR(1 DOWNTO 0);

SIGNAL EN_REGO_S0: STD_LOGIC_VECTOR(2 DOWNTO 0);
SIGNAL EN_REGO_S1: STD_LOGIC_VECTOR(2 DOWNTO 0);
SIGNAL EN_REGO_S2: STD_LOGIC_VECTOR(2 DOWNTO 0);
SIGNAL EN_REGO_S3: STD_LOGIC_VECTOR(2 DOWNTO 0);

--SEGNALI PER LA CU TOP INGRESSO
SIGNAL START_S,START_S_TOP, OUT_XOR_CHECK_S,NEW_OUT_XOR_CHECK_S : STD_LOGIC;
SIGNAL END_BF_TOP,FREE_M_TOP : STD_LOGIC;

-- SONO I SEGNALI DA MANDARE AI REGISTRI WR WI
SIGNAL  P_0,P_1,P_3,P_7,P_9 : SIGNED (19 DOWNTO 0);
SIGNAL  N_1,N_3,N_7,N_9 :SIGNED (19 DOWNTO 0);

-- SEGNALI PER CONTROLLO XOR (da mettere tra step 0 e XOR_CHECK
SIGNAL AI0_CTRL,AI1_CTRL,AI2_CTRL,AI3_CTRL,AI4_CTRL,AI5_CTRL,AI6_CTRL,AI7_CTRL :
SIGNED (19 DOWNTO 0);
SIGNAL AR0_CTRL,AR1_CTRL,AR2_CTRL,AR3_CTRL,AR4_CTRL,AR5_CTRL,AR6_CTRL,AR7_CTRL :
SIGNED (19 DOWNTO 0);
SIGNAL BR0_CTRL,BR1_CTRL,BR2_CTRL,BR3_CTRL,BR4_CTRL,BR5_CTRL,BR6_CTRL,BR7_CTRL :
SIGNED (19 DOWNTO 0);
SIGNAL BI0_CTRL,BI1_CTRL,BI2_CTRL,BI3_CTRL,BI4_CTRL,BI5_CTRL,BI6_CTRL,BI7_CTRL :
SIGNED (19 DOWNTO 0);

-- SEGNALI PER COLLEGARE UNA BF ALLA SUCCESSIVA
SIGNAL S01_0_AR,S01_1_AR, S01_2_AR, S01_3_AR, S01_4_AR, S01_5_AR, S01_6_AR,
S01_7_AR:SIGNED(19 DOWNTO 0);
SIGNAL S01_0_AI,S01_1_AI, S01_2_AI, S01_3_AI, S01_4_AI, S01_5_AI, S01_6_AI,
S01_7_AI:SIGNED(19 DOWNTO 0);
SIGNAL S01_0_BR,S01_1_BR, S01_2_BR, S01_3_BR, S01_4_BR, S01_5_BR, S01_6_BR,
S01_7_BR:SIGNED(19 DOWNTO 0);
SIGNAL S01_0_BI,S01_1_BI, S01_2_BI, S01_3_BI, S01_4_BI, S01_5_BI, S01_6_BI,
S01_7_BI:SIGNED(19 DOWNTO 0);

SIGNAL S12_0_AR,S12_1_AR, S12_2_AR, S12_3_AR, S12_4_AR, S12_5_AR, S12_6_AR,
S12_7_AR:SIGNED(19 DOWNTO 0);
SIGNAL S12_0_AI,S12_1_AI, S12_2_AI, S12_3_AI, S12_4_AI, S12_5_AI, S12_6_AI,
S12_7_AI:SIGNED(19 DOWNTO 0);
SIGNAL S12_0_BR,S12_1_BR, S12_2_BR, S12_3_BR, S12_4_BR, S12_5_BR, S12_6_BR,
S12_7_BR:SIGNED(19 DOWNTO 0);
SIGNAL S12_0_BI,S12_1_BI, S12_2_BI, S12_3_BI, S12_4_BI, S12_5_BI, S12_6_BI,
S12_7_BI:SIGNED(19 DOWNTO 0);
```

```
SIGNAL S23_0_AR,S23_1_AR, S23_2_AR, S23_3_AR, S23_4_AR, S23_5_AR, S23_6_AR,
S23_7_AR:SIGNED(19 DOWNTO 0);
SIGNAL S23_0_AI,S23_1_AI, S23_2_AI, S23_3_AI, S23_4_AI, S23_5_AI, S23_6_AI,
S23_7_AI:SIGNED(19 DOWNTO 0);
SIGNAL S23_0_BR,S23_1_BR, S23_2_BR, S23_3_BR, S23_4_BR, S23_5_BR, S23_6_BR,
S23_7_BR:SIGNED(19 DOWNTO 0);
SIGNAL S23_0_BI,S23_1_BI, S23_2_BI, S23_3_BI, S23_4_BI, S23_5_BI, S23_6_BI,
S23_7_BI:SIGNED(19 DOWNTO 0);

-- GUARD BIT
SIGNAL
C0AR_GUARD_S,C1AR_GUARD_S,C2AR_GUARD_S,C3AR_GUARD_S,C4AR_GUARD_S,C5AR_GUARD_S,C6
AR_GUARD_S,C7AR_GUARD_S : SIGNED(19 DOWNTO 0);
SIGNAL
C0AI_GUARD_S,C1AI_GUARD_S,C2AI_GUARD_S,C3AI_GUARD_S,C4AI_GUARD_S,C5AI_GUARD_S,C6
AI_GUARD_S,C7AI_GUARD_S : SIGNED(19 DOWNTO 0);
SIGNAL
C0BR_GUARD_S,C1BR_GUARD_S,C2BR_GUARD_S,C3BR_GUARD_S,C4BR_GUARD_S,C5BR_GUARD_S,C6
BR_GUARD_S,C7BR_GUARD_S : SIGNED(19 DOWNTO 0);
SIGNAL
C0BI_GUARD_S,C1BI_GUARD_S,C2BI_GUARD_S,C3BI_GUARD_S,C4BI_GUARD_S,C5BI_GUARD_S,C6
BI_GUARD_S,C7BI_GUARD_S : SIGNED(19 DOWNTO 0);


SIGNAL SENSE_S: STD_LOGIC;

SIGNAL SEQ_CU0,SEQ_CU1,SEQ_CU2,SEQ_CU3: STD_LOGIC;

SIGNAL SET_FALSO: STD_LOGIC;

SIGNAL SET_REG_SEQ: STD_LOGIC;

BEGIN


SET_REG_SEQ <= FF_VALUE_S AND NEW_OUT_XOR_CHECK_S;
NEW_OUT_XOR_CHECK_S <= OUT_XOR_CHECK_S AND START_S;
SET_FALSO <= RESET_OUT AND FF_VALUE_S;
START_S_TOP <= SENSE_S AND START_S;
NEW_PROGRESS_S <= PROGRESS_S(3) OR PROGRESS_S(2) OR PROGRESS_S(1) OR
PROGRESS_S(0);

-- BF DELLO STEP 0 COLLEGATE DIRETTAMENTI AGLI INGRESSI DELLA FFT 16
BF_0_0: BF PORT
MAP(AI=>C0AI_GUARD_S,AR=>C0AR_GUARD_S,BR=>C0BR_GUARD_S,BI=>C0BI_GUARD_S,
A1R=>S01_0_AR,A1I=>S01_0_AI,B1R=>S01_0_BR,B1I=>
      S01_0_BI,SEL3=>
SEL3_S0,SEL1=>SEL1_S0,SELSUM=>SELSUM_S0,C=>C_S0,A_S=>A_S_S0,EN_REGR=>
                              EN_REGR_S0,ENABLE=>ENABLE_S0,SEL2=>
SEL2_S0,EN_REGO=>EN_REGO_S0,WR=>P_1,WI=>P_0,CLK=>CLK,AI_CTRL=>AI1_CTRL,AR_CTRL=>
AR0_CTRL,BI_CTRL=>BI0_CTRL,BR_CTRL=>BR0_CTRL,SEL_INV=>SEL_INV_S0);
BF_0_1: BF PORT
MAP(AI=>C1AI_GUARD_S,AR=>C1AR_GUARD_S,BR=>C1BR_GUARD_S,BI=>C1BI_GUARD_S,
A1R=>S01_1_AR,A1I=>S01_1_AI,B1R=>S01_1_BR,B1I=>
      S01_1_BI, SEL3=>
SEL3_S0,SEL1=>SEL1_S0,SELSUM=>SELSUM_S0,C=>C_S0,A_S=>A_S_S0,EN_REGR=>
                              EN_REGR_S0,ENABLE=>ENABLE_S0,SEL2=>
SEL2_S0,EN_REGO=>EN_REGO_S0,WR=>P_1,WI=>P_0,
CLK=>CLK,AI_CTRL=>AI0_CTRL,AR_CTRL=>AR1_CTRL,BI_CTRL=>BI1_CTRL,BR_CTRL=>BR1_CTRL
,SEL_INV=>SEL_INV_S0);
BF_0_2: BF PORT
MAP(AI=>C2AI_GUARD_S,AR=>C2AR_GUARD_S,BR=>C2BR_GUARD_S,BI=>C2BI_GUARD_S,
```

```
A1R=>S01_2_AR,A1I=>S01_2_AI,B1R=>S01_2_BR,B1I=>
      S01_2_BI,SEL3=>
SEL3_S0,SEL1=>SEL1_S0,SELSUM=>SELSUM_S0,C=>C_S0,A_S=>A_S_S0,
                        EN_REGR=>EN_REGR_S0,ENABLE=>ENABLE_S0,SEL2=>
SEL2_S0,EN_REGO=>EN_REGO_S0,WR=>P_1,WI=>P_0,CLK=>CLK,AI_CTRL=>AI2_CTRL,AR_CTRL=>
AR2_CTRL,BI_CTRL=>BI2_CTRL,BR_CTRL=>BR2_CTRL,SEL_INV=>SEL_INV_S0);
BF_0_3: BF PORT
MAP(AI=>C3AI_GUARD_S,AR=>C3AR_GUARD_S,BR=>C3BR_GUARD_S,BI=>C3BI_GUARD_S,
A1R=>S01_3_AR,A1I=>S01_3_AI,B1R=>S01_3_BR,B1I=>
      S01_3_BI,SEL3=>
SEL3_S0,SEL1=>SEL1_S0,SELSUM=>SELSUM_S0,C=>C_S0,A_S=>A_S_S0,EN_REGR=>
                              EN_REGR_S0,ENABLE=>ENABLE_S0,SEL2=>
SEL2_S0,EN_REGO=>EN_REGO_S0,WR=>P_1,WI=>P_0,
CLK=>CLK,AI_CTRL=>AI3_CTRL,AR_CTRL=>AR3_CTRL,BI_CTRL=>BI3_CTRL,BR_CTRL=>BR3_CTRL
,SEL_INV=>SEL_INV_S0);
BF_0_4: BF PORT
MAP(AI=>C4AI_GUARD_S,AR=>C4AR_GUARD_S,BR=>C4BR_GUARD_S,BI=>C4BI_GUARD_S,
A1R=>S01_4_AR,A1I=>S01_4_AI,B1R=>S01_4_BR,B1I=>
      S01_4_BI, SEL3=>
SEL3_S0,SEL1=>SEL1_S0,SELSUM=>SELSUM_S0,C=>C_S0,A_S=>A_S_S0,EN_REGR=>
                              EN_REGR_S0,ENABLE=>ENABLE_S0,SEL2=>
SEL2_S0,EN_REGO=>EN_REGO_S0,WR=>P_1,WI=>P_0,
CLK=>CLK,AI_CTRL=>AI4_CTRL,AR_CTRL=>AR4_CTRL,BI_CTRL=>BI4_CTRL,BR_CTRL=>BR4_CTRL
,SEL_INV=>SEL_INV_S0);
BF_0_5: BF PORT
MAP(AI=>C5AI_GUARD_S,AR=>C5AR_GUARD_S,BR=>C5BR_GUARD_S,BI=>C5BI_GUARD_S,
A1R=>S01_5_AR,A1I=>S01_5_AI,B1R=>S01_5_BR,B1I=>
      S01_5_BI, SEL3=>
SEL3_S0,SEL1=>SEL1_S0,SELSUM=>SELSUM_S0,C=>C_S0,A_S=>A_S_S0,
                        EN_REGR=>EN_REGR_S0,ENABLE=>ENABLE_S0,SEL2=>
SEL2_S0,EN_REGO=>EN_REGO_S0,WR=>P_1,WI=>P_0,CLK=>CLK,AI_CTRL=>AI5_CTRL,AR_CTRL=>
AR5_CTRL,BI_CTRL=>BI5_CTRL,BR_CTRL=>BR5_CTRL,SEL_INV=>SEL_INV_S0);
BF_0_6: BF PORT
MAP(AI=>C6AI_GUARD_S,AR=>C6AR_GUARD_S,BR=>C6BR_GUARD_S,BI=>C6BI_GUARD_S,
A1R=>S01_6_AR,A1I=>S01_6_AI,B1R=>S01_6_BR,B1I=>
      S01_6_BI, SEL3=>
SEL3_S0,SEL1=>SEL1_S0,SELSUM=>SELSUM_S0,C=>C_S0,A_S=>A_S_S0,EN_REGR=>
                              EN_REGR_S0,ENABLE=>ENABLE_S0,SEL2=>
SEL2_S0,EN_REGO=>EN_REGO_S0,WR=>P_1,WI=>P_0,
CLK=>CLK,AI_CTRL=>AI6_CTRL,AR_CTRL=>AR6_CTRL,BI_CTRL=>BI6_CTRL,BR_CTRL=>BR6_CTRL
,SEL_INV=>SEL_INV_S0);
BF_0_7: BF PORT
MAP(AI=>C7AI_GUARD_S,AR=>C7AR_GUARD_S,BR=>C7BR_GUARD_S,BI=>C7BI_GUARD_S,
A1R=>S01_7_AR,A1I=>S01_7_AI,B1R=>S01_7_BR,B1I=>
      S01_7_BI, SEL3=>
SEL3_S0,SEL1=>SEL1_S0,SELSUM=>SELSUM_S0,C=>C_S0,A_S=>A_S_S0,EN_REGR=>
                              EN_REGR_S0,ENABLE=>ENABLE_S0,SEL2=>
SEL2_S0,EN_REGO=>EN_REGO_S0,WR=>P_1,WI=>P_0,
CLK=>CLK,AI_CTRL=>AI7_CTRL,AR_CTRL=>AR7_CTRL,BI_CTRL=>BI7_CTRL,BR_CTRL=>BR7_CTRL
,SEL_INV=>SEL_INV_S0);

-- BF STEP 1

BF_1_0: BF PORT MAP(AI=>S01_0_AI,AR=>S01_0_AR,BR=>S01_4_AR,BI=>S01_4_AI,
A1R=>S12_0_AR,A1I=>S12_0_AI,B1R=>
      S12_0_BR,B1I=>S12_0_BI,SEL3=>SEL3_S1,SEL1=>SEL1_S1,SELSUM=>SELSUM_S1,C=>C_
S1,A_S=>
      A_S_S1,EN_REGR=>EN_REGR_S1,ENABLE=>ENABLE_S1,SEL2=>
SEL2_S1,EN_REGO=>EN_REGO_S1,WR=>P_1,WI=>P_0, CLK=>CLK,SEL_INV=>SEL_INV_S1);
BF_1_1: BF PORT MAP(AI=>S01_1_AI,AR=>S01_1_AR,BR=>S01_5_AR,BI=>S01_5_AI,
A1R=>S12_1_AR,A1I=>S12_1_AI,B1R=>
      S12_1_BR,B1I=>S12_1_BI,SEL3=>SEL3_S1,SEL1=>
```

```
      SEL1_S1,SELSUM=>SELSUM_S1,C=>C_S1,A_S
      =>    A_S_S1,EN_REGR=>EN_REGR_S1,ENABLE=>ENABLE_S1,SEL2=>
SEL2_S1,EN_REGO=>EN_REGO_S1,WR=>P_1,WI=>P_0, CLK=>CLK,SEL_INV=>SEL_INV_S1);
BF_1_2: BF PORT MAP(AI=>S01_2_AI,AR=>S01_2_AR,BR=>S01_6_AR,BI=>S01_6_AI,
A1R=>S12_2_AR,A1I=>S12_2_AI,B1R=>
      S12_2_BR,B1I=>S12_2_BI,SEL3=>SEL3_S1,SEL1=>SEL1_S1,SELSUM=>SELSUM_S1,C=>C_
S1,A_S=>
      A_S_S1,EN_REGR=>EN_REGR_S1,ENABLE=>ENABLE_S1,SEL2=>
SEL2_S1,EN_REGO=>EN_REGO_S1,WR=>P_1,WI=>P_0, CLK=>CLK,SEL_INV=>SEL_INV_S1);
BF_1_3: BF PORT MAP(AI=>S01_3_AI,AR=>S01_3_AR,BR=>S01_7_AR,BI=>S01_7_AI,
A1R=>S12_3_AR,A1I=>S12_3_AI,B1R=>
      S12_3_BR,B1I=>S12_3_BI,SEL3=>SEL3_S1,SEL1=>SEL1_S1,SELSUM=>SELSUM_S1,C=>C_
S1,A_S=>
      A_S_S1,EN_REGR=>EN_REGR_S1,ENABLE=>ENABLE_S1,SEL2=>
SEL2_S1,EN_REGO=>EN_REGO_S1,WR=>P_1,WI=>P_0, CLK=>CLK,SEL_INV=>SEL_INV_S1);
BF_1_4: BF PORT MAP(AI=>S01_0_BI,AR=>S01_0_BR,BR=>S01_4_BR,BI=>S01_4_BI,
A1R=>S12_4_AR,A1I=>S12_4_AI,B1R=>
      S12_4_BR,B1I=>S12_4_BI,SEL3=>SEL3_S1,SEL1=>
      SEL1_S1,SELSUM=>SELSUM_S1,C=>C_S1,A_S=>
      A_S_S1,EN_REGR=>EN_REGR_S1,ENABLE=>ENABLE_S1,SEL2=>
SEL2_S1,EN_REGO=>EN_REGO_S1,WR=>P_0,WI=>N_1, CLK=>CLK,SEL_INV=>SEL_INV_S1);
BF_1_5: BF PORT MAP(AI=>S01_1_BI,AR=>S01_1_BR,BR=>S01_5_BR,BI=>S01_5_BI,
A1R=>S12_5_AR,A1I=>S12_5_AI,B1R=>
      S12_5_BR,B1I=>S12_5_BI,SEL3=>SEL3_S1,SEL1=>SEL1_S1,SELSUM=>SELSUM_S1,C=>C_
S1,A_S=>
      A_S_S1,EN_REGR=>EN_REGR_S1,ENABLE=>ENABLE_S1,SEL2=>
SEL2_S1,EN_REGO=>EN_REGO_S1,WR=>P_0,WI=>N_1, CLK=>CLK,SEL_INV=>SEL_INV_S1);
BF_1_6: BF PORT MAP(AI=>S01_2_BI,AR=>S01_2_BR,BR=>S01_6_BR,BI=>S01_6_BI,
A1R=>S12_6_AR,A1I=>S12_6_AI,B1R=>
      S12_6_BR,B1I=>S12_6_BI,SEL3=>SEL3_S1,SEL1=>SEL1_S1,SELSUM=>SELSUM_S1,C=>C_
S1,A_S=>
      A_S_S1,EN_REGR=>EN_REGR_S1,ENABLE=>ENABLE_S1,SEL2=>
SEL2_S1,EN_REGO=>EN_REGO_S1,WR=>P_0,WI=>N_1, CLK=>CLK,SEL_INV=>SEL_INV_S1);
BF_1_7: BF PORT MAP(AI=>S01_3_BI,AR=>S01_3_BR,BR=>S01_7_BR,BI=>S01_7_BI,
A1R=>S12_7_AR,A1I=>S12_7_AI,B1R=>
      S12_7_BR,B1I=>S12_7_BI,SEL3=>SEL3_S1,SEL1=>SEL1_S1,SELSUM=>SELSUM_S1,C=>C_
S1,A_S=>
      A_S_S1,EN_REGR=>EN_REGR_S1,ENABLE=>ENABLE_S1,SEL2=>
SEL2_S1,EN_REGO=>EN_REGO_S1,WR=>P_0,WI=>N_1, CLK=>CLK,SEL_INV=>SEL_INV_S1);

-- BF STEP 2

BF_2_0: BF PORT MAP(AI=>S12_0_AI,AR=>S12_0_AR,BR=>S12_2_AR,BI=>S12_2_AI,
A1R=>S23_0_AR,A1I=>S23_0_AI,B1R=>
      S23_0_BR,B1I=>S23_0_BI,SEL3=>SEL3_S2,SEL1=>SEL1_S2,SELSUM=>SELSUM_S2,C=>C_
S2,A_S=>
      A_S_S2,EN_REGR=>EN_REGR_S2,ENABLE=>ENABLE_S2,SEL2=>
SEL2_S2,EN_REGO=>EN_REGO_S2,WR=>P_1,WI=>P_0, CLK=>CLK,SEL_INV=>SEL_INV_S2);
BF_2_1: BF PORT MAP(AI=>S12_1_AI,AR=>S12_1_AR,BR=>S12_3_AR,BI=>S12_3_AI,
A1R=>S23_1_AR,A1I=>S23_1_AI,B1R=>
      S23_1_BR,B1I=>S23_1_BI,SEL3=>SEL3_S2,SEL1=>SEL1_S2,SELSUM=>SELSUM_S2,C=>C_
S2,A_S=>
      A_S_S2,EN_REGR=>EN_REGR_S2,ENABLE=>ENABLE_S2,SEL2=>
SEL2_S2,EN_REGO=>EN_REGO_S2,WR=>P_1,WI=>P_0, CLK=>CLK,SEL_INV=>SEL_INV_S2);
BF_2_2: BF PORT MAP(AI=>S12_0_BI,AR=>S12_0_BR,BR=>S12_2_BR,BI=>S12_2_BI,
A1R=>S23_2_AR,A1I=>S23_2_AI,B1R=>
      S23_2_BR,B1I=>S23_2_BI,SEL3=>SEL3_S2,SEL1=>SEL1_S2,SELSUM=>SELSUM_S2,C=>C_
S2,A_S=>
      A_S_S2,EN_REGR=>EN_REGR_S2,ENABLE=>ENABLE_S2,SEL2=>
SEL2_S2,EN_REGO=>EN_REGO_S2,WR=>P_0,WI=>N_1, CLK=>CLK,SEL_INV=>SEL_INV_S2);
BF_2_3: BF PORT MAP(AI=>S12_1_BI,AR=>S12_1_BR,BR=>S12_3_BR,BI=>S12_3_BI,
A1R=>S23_3_AR,A1I=>S23_3_AI,B1R=>
```

```
      S23_3_BR,B1I=>S23_3_BI,SEL3=>SEL3_S2,SEL1=>SEL1_S2,SELSUM=>SELSUM_S2,C=>C_
S2,A_S=>
      A_S_S2,EN_REGR=>EN_REGR_S2,ENABLE=>ENABLE_S2,SEL2=>
SEL2_S2,EN_REGO=>EN_REGO_S2,WR=>P_0,WI=>N_1, CLK=>CLK,SEL_INV=>SEL_INV_S2);
BF_2_4: BF PORT MAP(AI=>S12_4_AI,AR=>S12_4_AR,BR=>S12_6_AR,BI=>S12_6_AI,
A1R=>S23_4_AR,A1I=>S23_4_AI,B1R=>
      S23_4_BR,B1I=>S23_4_BI,SEL3=>SEL3_S2,SEL1=>SEL1_S2,SELSUM=>SELSUM_S2,C=>C_
S2,A_S=>
      A_S_S2,EN_REGR=>EN_REGR_S2,ENABLE=>ENABLE_S2,SEL2=>
SEL2_S2,EN_REGO=>EN_REGO_S2,WR=>P_7,WI=>N_7, CLK=>CLK,SEL_INV=>SEL_INV_S2);
BF_2_5: BF PORT MAP(AI=>S12_5_AI,AR=>S12_5_AR,BR=>S12_7_AR,BI=>S12_7_AI,
A1R=>S23_5_AR,A1I=>S23_5_AI,B1R=>
      S23_5_BR,B1I=>S23_5_BI,SEL3=>SEL3_S2,SEL1=>SEL1_S2,SELSUM=>SELSUM_S2,C=>C_
S2,A_S=>
      A_S_S2,EN_REGR=>EN_REGR_S2,ENABLE=>ENABLE_S2,SEL2=>
SEL2_S2,EN_REGO=>EN_REGO_S2,WR=>P_7,WI=>N_7, CLK=>CLK,SEL_INV=>SEL_INV_S2);
BF_2_6: BF PORT MAP(AI=>S12_4_BI,AR=>S12_4_BR,BR=>S12_6_BR,BI=>S12_6_BI,
A1R=>S23_6_AR,A1I=>S23_6_AI,B1R=>
      S23_6_BR,B1I=>S23_6_BI,SEL3=>SEL3_S2,SEL1=>SEL1_S2,SELSUM=>SELSUM_S2,C=>C_
S2,A_S=>
      A_S_S2,EN_REGR=>EN_REGR_S2,ENABLE=>ENABLE_S2,SEL2=>
SEL2_S2,EN_REGO=>EN_REGO_S2,WR=>N_7,WI=>N_7, CLK=>CLK,SEL_INV=>SEL_INV_S2);
BF_2_7: BF PORT MAP(AI=>S12_5_BI,AR=>S12_5_BR,BR=>S12_7_BR,BI=>S12_7_BI,
A1R=>S23_7_AR,A1I=>S23_7_AI,B1R=>
      S23_7_BR,B1I=>S23_7_BI,SEL3=>SEL3_S2,SEL1=>SEL1_S2,SELSUM=>SELSUM_S2,C=>C_
S2,A_S=>
      A_S_S2,EN_REGR=>EN_REGR_S2,ENABLE=>ENABLE_S2,SEL2=>
SEL2_S2,EN_REGO=>EN_REGO_S2,WR=>N_7,WI=>N_7, CLK=>CLK,SEL_INV=>SEL_INV_S2);

-- BF STEP 3 (ULTIMO)

BF_3_0: BF PORT MAP(AI=>S23_0_AI,AR=>S23_0_AR,BR=>S23_1_AR,BI=>S23_1_AI,
A1R=>C0AR_OUT,A1I=>C0AI_OUT,B1R=>
      C0BR_OUT,B1I=>C0BI_OUT,SEL3=>SEL3_S3,SEL1=>SEL1_S3,SELSUM=>SELSUM_S3,C=>C_
S3,A_S=>
      A_S_S3,EN_REGR=>EN_REGR_S3,ENABLE=>ENABLE_S3,SEL2=>
SEL2_S3,EN_REGO=>EN_REGO_S3,WR=>P_1,WI=>P_0, CLK=>CLK,SEL_INV=>SEL_INV_S3);
BF_3_1: BF PORT MAP(AI=>S23_0_BI,AR=>S23_0_BR,BR=>S23_1_BR,BI=>S23_1_BI,
A1R=>C1AR_OUT,A1I=>C1AI_OUT,B1R=>
      C1BR_OUT,B1I=>C1BI_OUT,SEL3=>SEL3_S3,SEL1=>SEL1_S3,SELSUM=>SELSUM_S3,C=>C_
S3,A_S=>
      A_S_S3,EN_REGR=>EN_REGR_S3,ENABLE=>ENABLE_S3,SEL2=>
SEL2_S3,EN_REGO=>EN_REGO_S3,WR=>P_0,WI=>N_1, CLK=>CLK,SEL_INV=>SEL_INV_S3);
BF_3_2: BF PORT MAP(AI=>S23_2_AI,AR=>S23_2_AR,BR=>S23_3_AR,BI=>S23_3_AI,
A1R=>C2AR_OUT,A1I=>C2AI_OUT,B1R=>
      C2BR_OUT,B1I=>C2BI_OUT,SEL3=>SEL3_S3,SEL1=>SEL1_S3,SELSUM=>SELSUM_S3,C=>C_
S3,A_S=>
      A_S_S3,EN_REGR=>EN_REGR_S3,ENABLE=>ENABLE_S3,SEL2=>
SEL2_S3,EN_REGO=>EN_REGO_S3,WR=>P_7,WI=>N_7, CLK=>CLK,SEL_INV=>SEL_INV_S3);
BF_3_3: BF PORT MAP(AI=>S23_2_BI,AR=>S23_2_BR,BR=>S23_3_BR,BI=>S23_3_BI,
A1R=>C3AR_OUT,A1I=>C3AI_OUT,B1R=>
      C3BR_OUT,B1I=>C3BI_OUT,SEL3=>SEL3_S3,SEL1=>SEL1_S3,SELSUM=>SELSUM_S3,C=>C_
S3,A_S=>
      A_S_S3,EN_REGR=>EN_REGR_S3,ENABLE=>ENABLE_S3,SEL2=>
SEL2_S3,EN_REGO=>EN_REGO_S3,WR=>N_7,WI=>N_7, CLK=>CLK,SEL_INV=>SEL_INV_S3);
BF_3_4: BF PORT MAP(AI=>S23_4_AI,AR=>S23_4_AR,BR=>S23_5_AR,BI=>S23_5_AI,
A1R=>C4AR_OUT,A1I=>C4AI_OUT,B1R=>
      C4BR_OUT,B1I=>C4BI_OUT,SEL3=>SEL3_S3,SEL1=>SEL1_S3,SELSUM=>SELSUM_S3,C=>C_
S3,A_S=>
      A_S_S3,EN_REGR=>EN_REGR_S3,ENABLE=>ENABLE_S3,SEL2=>
SEL2_S3,EN_REGO=>EN_REGO_S3,WR=>P_9,WI=>N_3, CLK=>CLK,SEL_INV=>SEL_INV_S3);
```

```
BF_3_5: BF PORT MAP(AI=>S23_4_BI,AR=>S23_4_BR,BR=>S23_5_BR,BI=>S23_5_BI,
A1R=>C5AR_OUT,A1I=>C5AI_OUT,B1R=>
      C5BR_OUT,B1I=>C5BI_OUT,SEL3=>SEL3_S3,SEL1=>SEL1_S3,SELSUM=>SELSUM_S3,C=>C_
S3,A_S=>
      A_S_S3,EN_REGR=>EN_REGR_S3,ENABLE=>ENABLE_S3,SEL2=>
SEL2_S3,EN_REGO=>EN_REGO_S3,WR=>N_3,WI=>N_9, CLK=>CLK,SEL_INV=>SEL_INV_S3);
BF_3_6: BF PORT MAP(AI=>S23_6_AI,AR=>S23_6_AR,BR=>S23_7_AR,BI=>S23_7_AI,
A1R=>C6AR_OUT,A1I=>C6AI_OUT,B1R=>
      C6BR_OUT,B1I=>C6BI_OUT,SEL3=>SEL3_S3,SEL1=>SEL1_S3,SELSUM=>SELSUM_S3,C=>C_
S3,A_S=>
      A_S_S3,EN_REGR=>EN_REGR_S3,ENABLE=>ENABLE_S3,SEL2=>
SEL2_S3,EN_REGO=>EN_REGO_S3,WR=>P_3,WI=>N_9, CLK=>CLK,SEL_INV=>SEL_INV_S3);
BF_3_7: BF PORT MAP(AI=>S23_6_BI,AR=>S23_6_BR,BR=>S23_7_BR,BI=>S23_7_BI,
A1R=>C7AR_OUT,A1I=>C7AI_OUT,B1R=>
      C7BR_OUT,B1I=>C7BI_OUT,SEL3=>SEL3_S3,SEL1=>SEL1_S3,SELSUM=>SELSUM_S3,C=>C_
S3,A_S=>
      A_S_S3,EN_REGR=>EN_REGR_S3,ENABLE=>ENABLE_S3,SEL2=>
SEL2_S3,EN_REGO=>EN_REGO_S3,WR=>N_9,WI=>N_3, CLK=>CLK,SEL_INV=>SEL_INV_S3);

OR_LOGIC:OR_PORT PORT
MAP(START=>START_S,B0AR=>C0AR,B1AR=>C1AR,B2AR=>C2AR,B3AR=>C3AR,B4AR=>C4AR,B5AR=>
C5AR,                                                  B6AR=>C6AR,B7AR=>C7AR,

B0AI=>C0AI,B1AI=>C1AI,B2AI=>C2AI,B3AI=>C3AI,B4AI=>C4AI,B5AI=>C5AI,B6AI=>C6AI,B7A
I
=>C7AI,B0BR=>C0BR,B1BR=>C1BR,B2BR=>C2BR,B3BR=>C3BR,B4BR=>C4BR,B5BR=>C5BR,B6BR=>

C6BR,B7BR=>C7BR,B0BI=>C0BI,B1BI=>C1BI,B2BI=>C2BI,B3BI=>C3BI,B4BI=>C4BI,B5BI=>C5B
I,                                                     B6BI=>C6BI,B7BI=>C7BI);


SEQ_LOGIC: XOR_CHECK PORT
MAP(OUT_XOR_CHECK=>OUT_XOR_CHECK_S,B0AR=>C0AR_GUARD_S,B1AR=>C1AR_GUARD_S,B2AR=>C
2AR_GUARD_S,B3AR=>C3AR_GUARD_S,
      B4AR=>C4AR_GUARD_S,B5AR=>C5AR_GUARD_S,B6AR=>C6AR_GUARD_S,B7AR=>C7AR_GUARD_
S,B0AI=>C0AI_GUARD_S,B1AI=>C1AI_GUARD_S,B2AI=>C2AI_GUARD_S,B3AI=>

      C3AI_GUARD_S,B4AI=>C4AI_GUARD_S,B5AI=>C5AI_GUARD_S,B6AI=>C6AI_GUARD_S,B7AI
=>C7AI_GUARD_S,B0BR=>C0BR_GUARD_S,B1BR=>C1BR_GUARD_S,B2BR=>C2BR_GUARD_S,

B3BR=>C3BR_GUARD_S,B4BR=>C4BR_GUARD_S,B5BR=>C5BR_GUARD_S,B6BR=>C6BR_GUARD_S,B7BR
=>C7BR_GUARD_S,B0BI=>C0BI_GUARD_S,B1BI=>C1BI_GUARD_S,B2BI=>

      C2BI_GUARD_S,B3BI=>C3BI_GUARD_S,B4BI=>C4BI_GUARD_S,B5BI=>C5BI_GUARD_S,B6BI
=>C6BI_GUARD_S,B7BI=>C7BI_GUARD_S,B0AR_OLD=>AR0_CTRL,

      B1AR_OLD=>AR1_CTRL,B2AR_OLD=>AR2_CTRL,B3AR_OLD=>AR3_CTRL,B4AR_OLD=>AR4_CTR
L,
      B5AR_OLD=>AR5_CTRL,B6AR_OLD=>AR6_CTRL,B7AR_OLD=>AR7_CTRL,B0AI_OLD=>AI0_CTR
L,B1AI_OLD=>
      AI1_CTRL,B2AI_OLD=>AI2_CTRL,B3AI_OLD=>AI3_CTRL,B4AI_OLD=>AI4_CTRL,B5AI_OLD
=>AI5_CTRL,
      B6AI_OLD=>AI6_CTRL,B7AI_OLD=>AI7_CTRL,B0BR_OLD=>BR0_CTRL,B1BR_OLD=>BR1_CTR
L,B2BR_OLD=>
      BR2_CTRL,B3BR_OLD=>BR3_CTRL,B4BR_OLD=>BR4_CTRL,B5BR_OLD=>BR5_CTRL,B6BR_OLD
=>BR6_CTRL,
      B7BR_OLD=>BR7_CTRL,B0BI_OLD=>BI0_CTRL,B1BI_OLD=>BI1_CTRL,B2BI_OLD=>BI2_CTR
L,B3BI_OLD=>
      BI3_CTRL,B4BI_OLD=>BI4_CTRL,B5BI_OLD=>BI5_CTRL,B6BI_OLD=>BI6_CTRL,B7BI_OLD
=>BI7_CTRL);

STONE_0:    R_STONE_0 PORT MAP(OUTPUT=>P_0);
```

```
STONE_1: R_STONE_1 PORT MAP(OUTPUT_P=>P_1,OUTPUT_N=>N_1);
STONE_3: R_STONE_3 PORT MAP(OUTPUT_P=>P_3,OUTPUT_N=>N_3);
STONE_7: R_STONE_7 PORT MAP(OUTPUT_P=>P_7,OUTPUT_N=>N_7);
STONE_9: R_STONE_9 PORT MAP(OUTPUT_P=>P_9,OUTPUT_N=>N_9);

-- CU BF
CU0 : CU_BF PORT MAP
(LOAD=>ENABLE_S0,SEQ=>SEQ_CU0,CLK=>CLK,RESET=>RESET,CTRL_OUT(14)=>SEL_INV_S0,

CTRL_OUT(13)=>SELSUM_S0,CTRL_OUT(12)=>SEL1_S0,CTRL_OUT(11 downto 10)=>SEL2_S0,

CTRL_OUT(9)=>SEL3_S0,CTRL_OUT(8)=>C_S0,CTRL_OUT(7)=>A_S_S0,CTRL_OUT(6)=>EN_REGS0
,
                                                CTRL_OUT(5)=>EN_REGR_S0,CTRL_OUT(4
DOWNTO 2)=>EN_REGO_S0,CTRL_OUT(1)=>FREE_M_TOP,
                                                CTRL_OUT(0)=>ENABLE_S1);
CU1 : CU_BF PORT MAP
(LOAD=>ENABLE_S1,SEQ=>SEQ_CU1,CLK=>CLK,RESET=>RESET,CTRL_OUT(14)=>SEL_INV_S1,

CTRL_OUT(13)=>SELSUM_S1,CTRL_OUT(12)=>SEL1_S1,CTRL_OUT(11 downto 10)=>SEL2_S1,

CTRL_OUT(9)=>SEL3_S1,CTRL_OUT(8)=>C_S1,CTRL_OUT(7)=>A_S_S1,CTRL_OUT(6)=>EN_REGS1
,
                                                CTRL_OUT(5)=>EN_REGR_S1,CTRL_OUT(4
DOWNTO 2)=>EN_REGO_S1,CTRL_OUT(1)=>FREE_M_S,
                                                CTRL_OUT(0)=>ENABLE_S2);
CU2 : CU_BF PORT MAP
(LOAD=>ENABLE_S2,SEQ=>SEQ_CU2,CLK=>CLK,RESET=>RESET,CTRL_OUT(14)=>SEL_INV_S2,

CTRL_OUT(13)=>SELSUM_S2,CTRL_OUT(12)=>SEL1_S2,CTRL_OUT(11 downto 10)=>SEL2_S2,

CTRL_OUT(9)=>SEL3_S2,CTRL_OUT(8)=>C_S2,CTRL_OUT(7)=>A_S_S2,CTRL_OUT(6)=>EN_REGS2
,
                                                CTRL_OUT(5)=>EN_REGR_S2,CTRL_OUT(4
DOWNTO 2)=>EN_REGO_S2,CTRL_OUT(1)=>FREE_M_S,
                                                CTRL_OUT(0)=>ENABLE_S3);
CU3 : CU_BF PORT MAP
(LOAD=>ENABLE_S3,SEQ=>SEQ_CU3,CLK=>CLK,RESET=>RESET,CTRL_OUT(14)=>SEL_INV_S3,

CTRL_OUT(13)=>SELSUM_S3,CTRL_OUT(12)=>SEL1_S3,CTRL_OUT(11 downto 10)=>SEL2_S3,

CTRL_OUT(9)=>SEL3_S3,CTRL_OUT(8)=>C_S3,CTRL_OUT(7)=>A_S_S3,CTRL_OUT(6)=>EN_REGS3
,
                                                CTRL_OUT(5)=>EN_REGR_S3,CTRL_OUT(4
DOWNTO 2)=>EN_REGO_S3,CTRL_OUT(1)=>FREE_M_S,
                                                CTRL_OUT(0)=>END_BF_TOP);

--CU TOP
TOP : CU_TOP PORT MAP
(START=>START_S_TOP,PROGRESS=>NEW_PROGRESS_S,FREE_M=>FREE_M_TOP,END_BF=>END_BF_T
OP,

SEQ=>NEW_OUT_XOR_CHECK_S,CLK=>CLK,RESET=>RESET,

CTRL_TOP_OUT(5)=>ENABLE_S0,CTRL_TOP_OUT(4)=>EN_FF_S,CTRL_TOP_OUT(3)=>FF_VALUE_S,

CTRL_TOP_OUT(2)=>DONE,CTRL_TOP_OUT(1)=>READY,CTRL_TOP_OUT(0)=>RESET_OUT);

-- SENSE START
SS : start_sense PORT
MAP(D=>EN_FF_S,CLK=>CLK,set=>RESET_OUT,enable=>EN_FF_S,SENSE=>SENSE_S);
```

```
-- REG STATUS
RS : REG_STATUS PORT MAP
(CLK=>CLK,set=>ENABLE_S0,enable0=>EN_REGS0,enable1=>EN_REGS1,enable2=>EN_REGS2,

enable3=>EN_REGS3,Q_OUT=>PROGRESS_S,RESET=>RESET_OUT);
-- REG SEQ
RSEQ : REG_SEQ PORT MAP
(CLK=>CLK,D0=>NEW_OUT_XOR_CHECK_S,D1=>PROGRESS_S(0),D2=>PROGRESS_S(1),D3=>PROGRE
SS_S(2),

SET=>SET_REG_SEQ,RESET=>RESET_OUT,EN0=>ENABLE_S1,EN1=>ENABLE_S2,

      EN2=>ENABLE_S3,EN3=>END_BF_TOP,Q0=>SEQ_CU0,Q1=>SEQ_CU1,Q2=>SEQ_CU2,Q3=>SEQ
_CU3);

GB : GUARD PORT MAP
(C0AR=>C0AR,C1AR=>C1AR,C2AR=>C2AR,C3AR=>C3AR,C4AR=>C4AR,C5AR=>C5AR,C6AR=>C6AR,C7
AR=>C7AR,

      C0AI=>C0AI,C1AI=>C1AI,C2AI=>C2AI,C3AI=>C3AI,C4AI=>C4AI,C5AI=>C5AI,C6AI=>C6
AI,C7AI=>C7AI,

      C0BR=>C0BR,C1BR=>C1BR,C2BR=>C2BR,C3BR=>C3BR,C4BR=>C4BR,C5BR=>C5BR,C6BR=>C6
BR,C7BR=>C7BR,

      C0BI=>C0BI,C1BI=>C1BI,C2BI=>C2BI,C3BI=>C3BI,C4BI=>C4BI,C5BI=>C5BI,C6BI=>C6
BI,C7BI=>C7BI,


      C0AR_GUARD=>C0AR_GUARD_S,C1AR_GUARD=>C1AR_GUARD_S,C2AR_GUARD=>C2AR_GUARD_S
,C3AR_GUARD=>C3AR_GUARD_S,

      C4AR_GUARD=>C4AR_GUARD_S,C5AR_GUARD=>C5AR_GUARD_S,C6AR_GUARD=>C6AR_GUARD_S
,C7AR_GUARD=>C7AR_GUARD_S,

      C0AI_GUARD=>C0AI_GUARD_S,C1AI_GUARD=>C1AI_GUARD_S,C2AI_GUARD=>C2AI_GUARD_S
,C3AI_GUARD=>C3AI_GUARD_S,

      C4AI_GUARD=>C4AI_GUARD_S,C5AI_GUARD=>C5AI_GUARD_S,C6AI_GUARD=>C6AI_GUARD_S
,C7AI_GUARD=>C7AI_GUARD_S,

      C0BR_GUARD=>C0BR_GUARD_S,C1BR_GUARD=>C1BR_GUARD_S,C2BR_GUARD=>C2BR_GUARD_S
,C3BR_GUARD=>C3BR_GUARD_S,

      C4BR_GUARD=>C4BR_GUARD_S,C5BR_GUARD=>C5BR_GUARD_S,C6BR_GUARD=>C6BR_GUARD_S
,C7BR_GUARD=>C7BR_GUARD_S,

      C0BI_GUARD=>C0BI_GUARD_S,C1BI_GUARD=>C1BI_GUARD_S,C2BI_GUARD=>C2BI_GUARD_S
,C3BI_GUARD=>C3BI_GUARD_S,

      C4BI_GUARD=>C4BI_GUARD_S,C5BI_GUARD=>C5BI_GUARD_S,C6BI_GUARD=>C6BI_GUARD_S
,C7BI_GUARD=>C7BI_GUARD_S);

      END BEHAV;
```