Capitolo 3

Implementazione SRRC polifase

3.1 Sommario

La struttura polifase ha un ruolo preponderante nell'architettura del modulatore pertanto ogni miglioramento ad essa apportato si riversa in maniera proporzionale sulle prestazioni globali del circuito. Dopo aver descritto il progetto del filtro SRRC da essa ospitato, si passa a definire due diverse realizzazioni, la prima deriva direttamente dalla teoria e non è adatta ad una implementazione su FPGA, la seconda comporta una progettazione più elaborata ma l'implementazione è semplificata, vengono presentate entrambe al fine di mettere in evidenza le scelte progettuali.

3.2 Progetto del filtro SRRC

Le considerazioni emerse nella trattazione dell'ISI[1] hanno portato ad individuare la seguente famiglia di spettri a coseno rialzato:

$$H_{rc}(f) = \begin{cases} T & \text{se} \quad 0 \leq |f| \leq \frac{(1+\alpha)}{2T} \\ \frac{T}{2}\left[1+\cos\frac{\pi T}{\alpha}\left(|f|-\frac{(1-\alpha)}{2T}\right)\right] & \text{se} \quad \frac{(1+\alpha)}{2T} \leq |f| \leq \frac{(1-\alpha)}{2T} \\ 0 & \text{se} \quad 0 \leq |f| \geq \frac{(1-\alpha)}{2T} \end{cases} \tag{3.1}$$

per essa le specifiche prevedono un fattore di roll-off $\alpha = 0,35$ scelto come compromesso tra utilizzazione della banda e complessità della realizzazione, la risposta in frequenza viene suddivisa equamente tra trasmissione e ricezione

$$|H_{Tx}(f)| = |H_{Rx}(f)| = \sqrt{H_{rc}(f)} \tag{3.2}$$

[1] Sezione(1.3.3)

in modo da poter utilizzare il filtro in ricezione anche come filtro adattato e quindi demodulatore secondo quanto visto in Sezione(1.3.1).

Il progetto del filtro è stato effettuato col metodo del campionamento della risposta in frequenza data dalle equazioni (3.1) e (3.2), di essa si prendono $\frac{N-1}{2}$ campioni e per simmetria si ricavano i restanti $\frac{N-1}{2}$, effettuando per ciascuno di essi la IDFT[2] si ottiene un campionamento della risposta impulsiva e quindi gli N coefficienti del filtro SRRC. L'espressione della IDFT è:

$$h_d(n) = \sum_{k=0}^{N-1} H_d(k)\, e^{\jmath\left(\frac{2\pi}{N}\right)kn} \tag{3.3}$$

che per la condizione di simmetria si riduce a:

$$h_d(n) = H_d(0) + \sum_{k=0}^{\frac{N-1}{2}} H_d(k) \cos\left(\frac{2\pi}{N} kn\right) \tag{3.4}$$

L'anticausalità viene rimossa traslando la risposta impulsiva di $\frac{N-1}{2}$ campioni.

Non necessariamente un numero di campioni maggiore si traduce in un aumento dell'attenuazione del filtro in banda oscura, ad esempio il filtro SRRC interpolante 6 è stato realizzato con 39 coefficienti piuttosto che con i 42 consentiti dall'architettura in quanto per quest'ultimo valore gli estremi della risposta impulsiva hanno un'ampiezza maggiore che si traduce in un innalzamento della soglia del rumore. Partendo da questa considerazione e tenendo conto del fatto che i tre valori di interpolazione vengono implementati tutti con una unica architettura, si è individuato il n° di coefficienti ottimale nei tre casi, in particolare l'interpolazione 3 viene realizzata con 19 coefficienti, l'interpolazione 4 con 25 coefficienti e l'interpolazione 6 con 39 coefficienti.

La tecnica di progetto basata sul campionamento in frequenza ben si presta ad una eventuale compensazione della risposta in frequenza del DAC[3] dovuta al mantenimento di ordine zero:

$$H_{DAC}(f) = \frac{\sin\left(\frac{\pi f}{f_s}\right)}{\pi f} e^{-\jmath\pi\left(\frac{f}{f_s}\right)} \tag{3.5}$$

tale compensazione si realizza moltiplicando la risposta in frequenza dell'SRRC per l'inverso della risposta in frequenza del DAC per poi procedere al campionamento della risposta globale.

[2] Inverse Discrete Fourier Transform
[3] Digital Analog Converter

3.3 Implementazione polifase SRRC

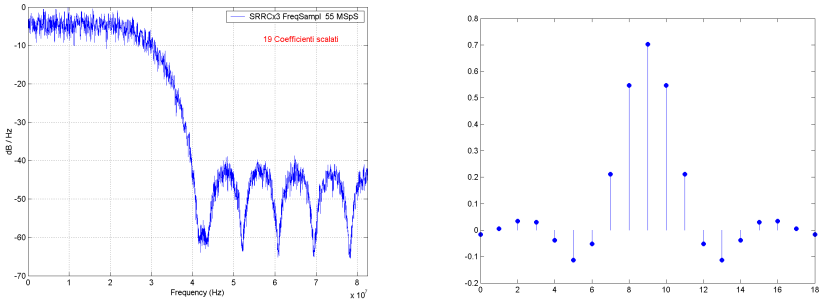
La decomposizione polifase[4] può essere implementata in maniera immediata distribuendo ciclicamente i coefficienti del filtro SRRC su un banco di N filtri FIR con N pari al valore di interpolazione desiderata. Ogni FIR utilizza 7 coefficienti pertanto per interpolare 3 i 19 coefficienti con l'aggiunta di due coefficienti nulli vengono distribuiti sui primi 3 FIR, per interpolare 4 i 25 coefficienti più 3 nulli sono applicati ai primi 4 FIR ed infine per interpolare 6 i 39 coefficienti più due nulli vengono distribuiti su tutti i FIR.

3.3.1 Modello Matlab

Lo script Matlab ***CreaCoeffsFreqSamplScaled.m*** *(Listato E.3.1)* effettua il calcolo dei coefficienti del filtro SRRC in accordo alla teoria espressa in Sezione(3.2), in particolare la risposta in frequenza descritta dall'equazione(3.1) è implementata dalla funzione ***RaisedCosineResponse.m*** *(Listato E.3.3)* . La distribuzione dei coefficienti sui diversi filtri viene eseguita tramite la funzione ***applica_polifase.m*** *(Listato E.3.2)* , essa calcola il numero dei FIR in base al valore dell'interpolazione poi associa a ciascuno di essi una riga in una matrice che viene riempita una colonna alla volta sino all'esaurimento dei coefficienti. La sequenza dati di test viene applicata ad ognuno di questi FIR tramite la funzione *filter*, di Matlab, ottenendo una matrice con un numero di righe pari al numero di FIR ed un numero di colonne pari alla dimensione del vettore di test, gli elementi di questa matrice vengono prelevati colonna per colonna modellando così il comportamento del commutatore presente nella descrizione della architettura polifase ottimizzata, di questo vettore viene calcolato lo spettro. ***CreaCoeffsFreqSamplScaled.m*** *(Listato E.3.1)* consente anche di scalare i coefficienti al fine di utilizzare al massimo la dinamica della rappresentazione in complemento a due utilizzata[5] in particolare la somma che si effettua in ogni FIR del polifase deve valere al massimo 1, essa è nota a priori in quanto sono noti i coefficienti ed il segnale d'ingresso appartiene all'insieme $(+1, -1)$. Lo stesso ***CreaCoeffsFreqSamplScaled.m*** *(Listato E.3.1)* consente il confronto tra il filtro progettato col campionamento in frequenza e quello che Matlab progetta automaticamente con la funzione *rcosine*, tale confronto mostra come le prestazioni del campionamento in frequenza siano nettamente superiori. La risposta in frequenza ed all'impulso degli SRRC per i tre data rate richiesti in specifica è rappresentata nelle Figure (3.1), (3.2) e (3.3) che evidenziano una differenza di circa 40dB tra banda passante e banda oscura.

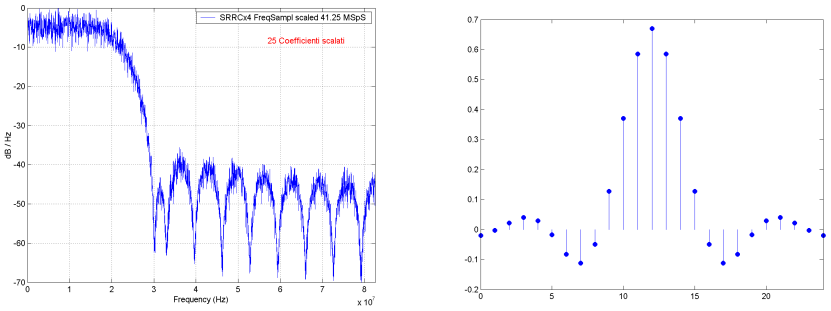
[4] Appendice(B.3.2)

[5] 12 bit di cui 1 per il segno ed i restanti per la parte dopo la virgola



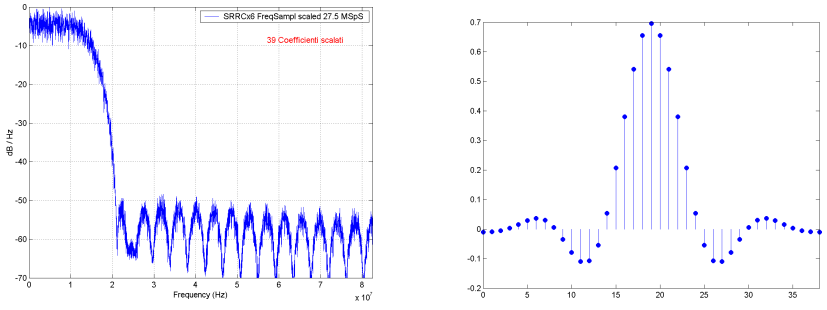
(a) Risposta in frequenza (b) Risposta all'impulso

Figura 3.1: SRRC 19 coefficienti interpolante 3



(a) Risposta in frequenza (b) Risposta all'impulso

Figura 3.2: SRRC 25 coefficienti interpolante 4



(a) Risposta in frequenza (b) Risposta all'impulso

Figura 3.3: SRRC 39 coefficienti interpolante 6

3.4 Modello VHDL

Le differenze sostanziali tra le due implementazioni VHDL del polifase risiedono nella strategia di creazione e distribuzione dei clock ridotti, generati a partire dal clock di sistema a 165MHz, e nella dislocazione della logica combinatoria.

3.4.1 Polifase standard *gated-clock*

Questa realizzazione è costituita principalmente da due blocchi:

SRRCxN: incorpora i FIR ed un multiplexer che seleziona ciclicamente, ad ogni colpo di clock, l'uscita di uno di essi.

RateAdapter: in funzione del data rate selezionato applica ad SRRCxN il giusto set di coefficienti, il clock ridotto ed una conta ciclica da 0 a N-1 essendo N il valore dell'interpolazione.

3.4.1.1 SRRCxN

Nello schema a blocchi in Figura(3.4), associato al file ***srrc_x_n.vhd***

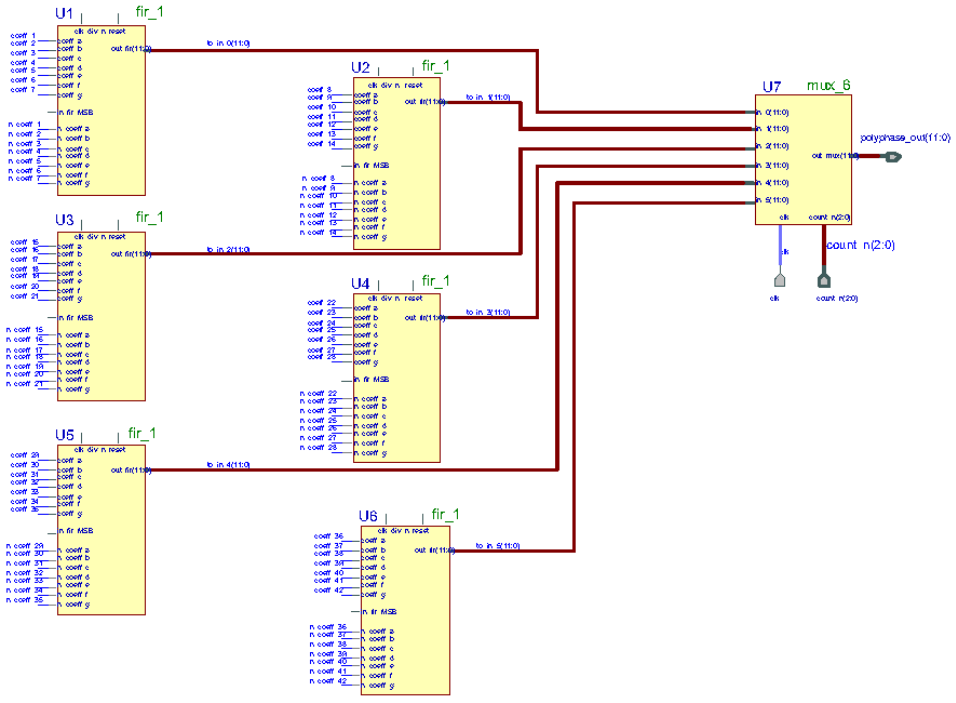


Figura 3.4: SRRC interpolante 3, 4, 6

(Listato F.2.10) , si ha che il multiplexer descritto in ***mux_6.vhd*** *(Listato F.2.15)* opera alla frequenza di clock mentre ciascuno dei 6 FIR descritto da ***fir_1.vhd*** *(Listato F.2.11)* e rappresentato in Figura(3.5) utilizza la

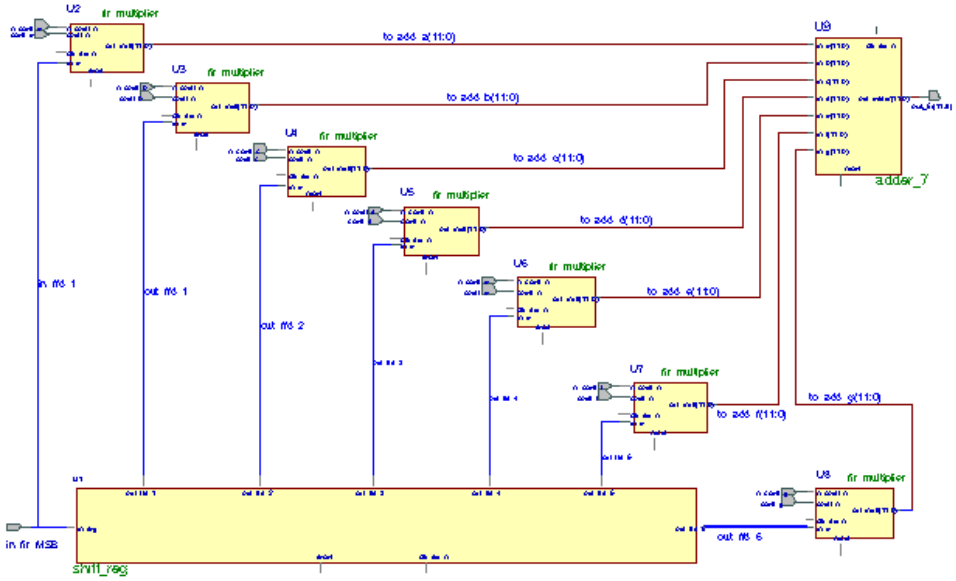


Figura 3.5: FIR i_{esimo}

stessa temporizzazione dei dati in ingresso.

I moltiplicatori sono in genere la parte più delicata nelle implementazioni di algoritmi DSP[6] su FPGA, laddove possibile è bene non utilizzarli o semplificarli al massimo, nel modulatore QPSK il segnale d'ingresso ai FIR può assumere soltanto i valori[7] $+1$ e -1 pertanto ogni moltiplicatore può essere sostituito con una descrizione VHDL che ad ogni colpo del clock ridotto presenta in uscita il valore del coefficiente o il suo negato. La negazione di un numero rappresentato in complemento a due si realizza invertendone tutti i bit e sommando 1, necessita pertanto di un sommatore che è un altro dei circuiti critici dei DSP, per evitarlo dato il numero esiguo di coefficienti si è scelto di fornire alla descrizione VHDL ***fir_multiplier.vhd*** *(Listato F.2.13)* che di fatto sostituisce il moltiplicatore, sia il coefficiente che il suo negato. La somma delle uscite dei moltiplicatori viene effettuata in ***adder_7.vhd*** *(Listato F.2.12)* , un sommatore la cui complessità limita le prestazioni del modulatore sia perché è costituito da 7 ingressi a 12bit sia perché lo schema del modulatore in Figura(2.6) ne richiede 12.

Gli elementi di ritardo tipici dei FIR sono realizzati con un registro a

[6] Digital Signal Processing

[7] due valori si rappresentano con 1 solo bit anche se in complemento a due a $+1$ corrisponde 01 e a -1 corrisponde 11

scorrimento SIPO[8] il cui VHDL è in ***shift_reg.vhd*** *(Listato F.2.14)* , tale soluzione è migliore rispetto al mettere in cascata 7 descrizioni FFD[9] in quanto segnala chiaramente al sintetizzatore VHDL che essi debbono esser posti quanto più vicini possibile tra di loro.

3.4.1.2 Rate Adapter

Nello schema a blocchi in Figura(3.6), associato al file ***rate_adapter.vhd***

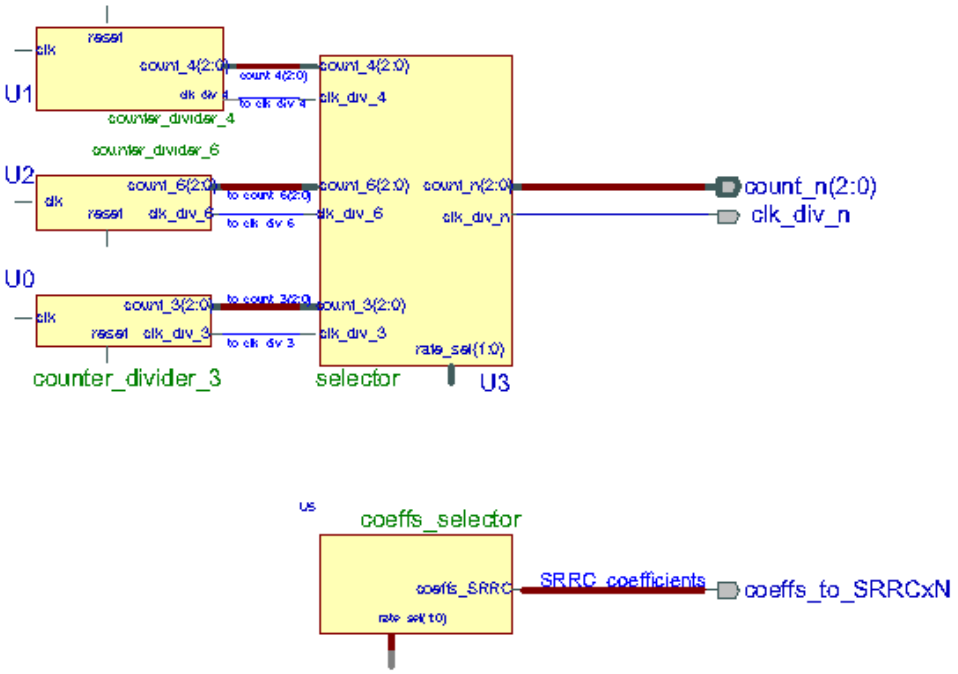


Figura 3.6: Rate Adapter

(Listato F.2.3) , si ha che ***selector.vhd*** *(Listato F.2.9)* a seconda del data rate impostato tramite *rate_sel* seleziona uno tra i contatori/divisori descritti da ***counter_divider_3.vhd*** *(Listato F.2.5)* , ***counter_divider_4.vhd*** *(Listato F.2.6)* e ***counter_divider_6.vhd*** *(Listato F.2.8)* , tra essi il più critico è senza dubbio il contatore/divisore per 3 in quanto utilizza entrambe i fronti del clock e pertanto è come se operasse ad una velocità di clock doppia rispetto agli altri due.

La selezione di quale banco di coefficienti utilizzare viene effettuata in ***coeffs_selector.vhd*** *(Listato F.2.4)* basandosi sul valore di *rate_sel*, si tratta di una descrizione non efficiente in quanto il sintetizzatore la implementa

[8] Serial In Parallel Out
[9] Flip Flop D

con un multiplexer le cui dimensioni divengono molto grandi al crescere del numero di coefficienti.

Trattandosi di una architettura *gated-clock*, si ha che i clock ridotti vengono generati con elaborazioni combinatorie sul clock di sistema, ciò è da evitare in quanto un semplice rumore, termico o ambientale, può dar luogo a problemi di corsa critica, per questo motivo viene preferita la tecnica *clock-enable* nella quale tutti i blocchi che richiedono il clock ridotto operano in realtà col clock di sistema ma dispongono di un segnale di abilitazione *clock-enable* pilotato tramite della logica combinatoria anch'essa temporizzata dal clock di sistema, in tal modo non è nemmeno più necessario che i clock ridotti abbiano un duty-cycle del 50%.

3.4.1.3 Risultati sperimentali

Il VHDL di questo progetto è stato ampliamente testato ed ha rappresentato un punto di partenza verso una conoscenza più approfondita delle tematiche della sintesi tuttavia non è stata effettuata alcuna implementazione su FPGA in quanto l'eccessiva sensibilità del *gated-clock* lo rende inadatto ad applicazioni spaziali dove i circuiti sono soggetti a radiazioni, stress meccanici e termici.

3.4.2 ROM Polifase *clock-enable*

La Figura(3.5) è il punto di partenza per una semplificazione sostanziale del polifase e quindi del modulatore, essa evidenzia come ognuno degli addendi che giungono al sommatore può assumere solo due valori, quello del coefficiente o il suo negato, pertanto il numero di combinazioni possibili al suo ingresso è $2^7 = 128$, un numero finito e tutto sommato limitato, di qui l'idea di inserire in una ROM i risultati delle somme corrispondenti a queste 128 combinazioni, eliminando in tal modo il grosso sommatore che limita la massima frequenza di clock applicabile al modulatore.

Ogni FIR viene pertanto realizzato con una ROM ed un registro a scorrimento nel quale fluiscono i simboli da trasmettere, le uscite in parallelo del registro individuano una locazione di memoria nella quale è memorizzato su 12 bit il risultato della somma. Un polifase che interpoli N si realizza con una unica ROM avente nelle prime 128 locazioni le somme per il primo FIR, nelle successive 128 le somme per il secondo FIR e così via sino alle somme per il FIR N_{esimo}. Il multiplexer che ha il compito di iterare ciclicamente sulle uscite dei FIR si ottiene aggiungendo dei bit alla ROM e facendoli pilotare direttamente dal contatore/divisore che quindi in ogni periodo di clock va a selezionare un diverso banco di 128 locazioni di memoria corrispondente ad uno degli N FIR.

Sulla base delle precedenti considerazioni il polifase che consente di implementare i 3 data rate richiesti dalle specifiche diviene quello in Fi-

gura(3.7), cui corrisponde il file ***srrc_x_n.vhd*** *(Listato F.3.3)* , in esso

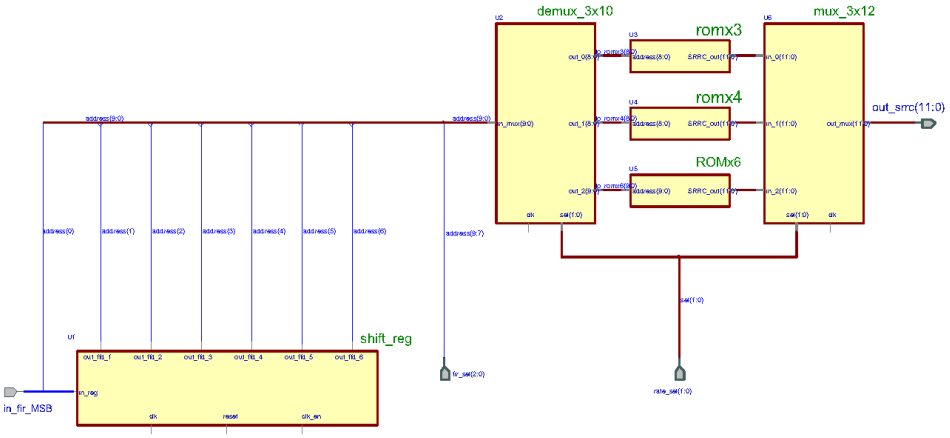


Figura 3.7: SRRCxN versione ROM

il segnale *rate_sel* imposta il data rate e conseguentemente il multiplexer descritto in ***mux_3x12*** *(Listato F.3.5)* insieme al demultiplexer ***demux_3x10.vhd*** *(Listato F.3.4)* seleziona la ROM da utilizzare tra quelle descritte in ***ROMx3.vhd*** *(Listato F.3.6)* , ***ROMx4.vhd*** *(Listato F.3.7)* e ***ROMx6.vhd*** *(Listato F.3.8)* .

I valori da inserire nelle ROM sono determinati da ***CreaROM.m*** *(Listato E.3.4)* a partire dal file contenente i coefficienti scalati, esso calcola tutte le possibili somme per ogni FIR, le pone in cascata, implementando così un intero polifase, e produce delle descrizioni della ROM in uno dei 3 formati:

COSTANTE: è il formato utilizzato per piccole ROM, viene sintetizzato infatti come un multiplexer le cui dimensioni possono divenire rilevanti.

CASE: è adatto a ROM di medie dimensioni in quanto il costrutto VHDL CASE viene riconosciuto dal sintetizzatore XST che lo implementa sulla RAM distribuita in ogni CLB[10].

BlockRAM: viene utilizzato per ROM di dimensioni molto grandi implementate tramite la BlockRAM[11] presente nella FPGA Virtex, è una soluzione che consente velocità di clock inferiori rispetto alla ROM distribuita nelle CLB, tuttavia è decisamente più agevole il Floorplanning[12]. Il file di testo prodotto può essere applicato al *Core Generator* per ottenere la BlockRAM inizializzata.

[10] Configurable Logic Block
[11] Appendice(C.3.4.2)
[12] Appendice(D.6)

La produzione e distribuzione dei clock ridotti a partire dal clock di sistema a 165MHz per i motivi precedentemente espressi non avviene più secondo la tecnica *gated-clock* ma secondo la *clock-enable*. I 3 contatori/-divisori utilizzati nel precedente progetto vengono soppiantati dal contatore/divisore programmabile descritto in ***counter.vhd*** *(Listato F.3.2)* , esso produce sia la conta ciclica sino ad $N-1$ che la produzione di un impulso di abilitazione ogni N colpi di clock destinato alle descrizioni che debbono operare a velocità ridotta. Apparentemente, per quello che riguarda i vincoli sulle temporizzazioni, il *clock-enable* introduce un abbassamento della massima frequenza di clock in quanto circuiti ai quali prima si applicava il vincolo del clock ridotto ora si vedono applicato il vincolo del clock di sistema che è più stringente, tuttavia per essi si può settare una specifica denominata *multi-cycle* che comunica al sintetizzatore di rilassare le loro temporizzazioni di un fattore N.

3.4.2.1 Risultati sperimentali

Per la versione ROM *clock-enable* del polifase sono state effettuate le due seguenti verifiche sperimentali:

VHDL: si è applicata alla descrizione VHDL del polifase la medesima sequenza applicata al polifase Matlab, gli spettri risultanti sono praticamente coincidenti come illustra la Figura(3.8) riferita ad un clock di

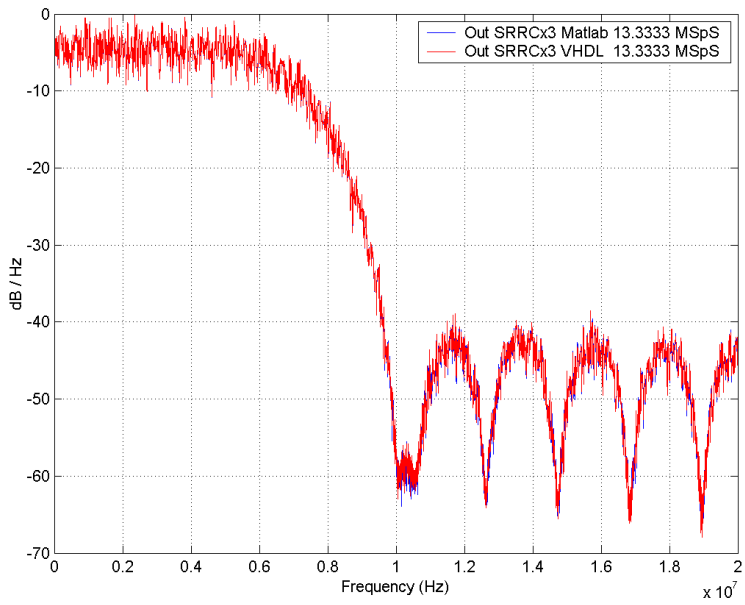


Figura 3.8: Confronto Polifase VHDL – Polifase Matlab

sistema di 40MHz generata tramite ***PolyphasePSDVHDLvsPSD-Matlab.m*** *(Listato E.3.5)* .

FPGA: l'implementazione su FPGA è stata abbastanza elaborata per via della necessità di adattare il ritmo dei dati provenienti dal *pattern-generator* con il ritmo del segnale di abilitazione generato a partire dal clock di sistema, la soluzione trovata viene descritta in Sezione(4.3.2), qui ci si limita a riportare i risultati ottenuti tramite ***PolyphasePSDFPGAvsPSDMatlab.m*** *(Listato E.3.7)* in termini di confronto spettrale col modello Matlab, essi sono graficati in Figura(3.9) che

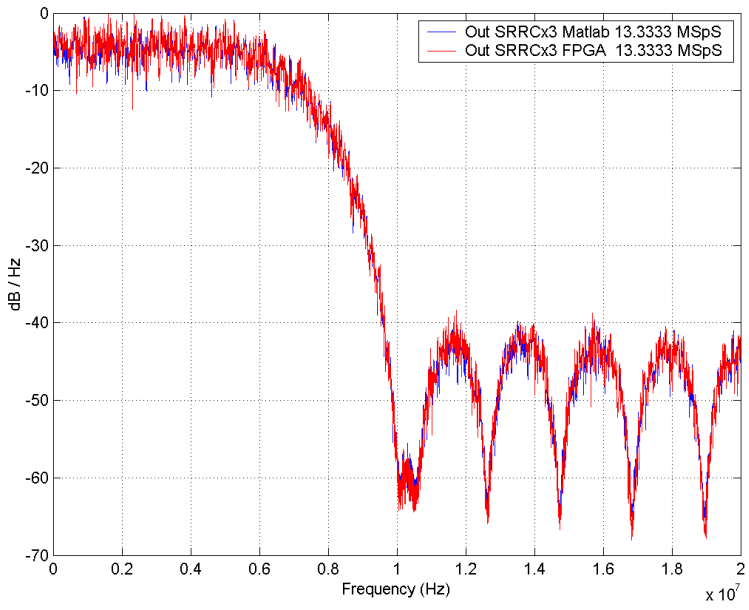


Figura 3.9: Confronto Polifase FPGA – Polifase Matlab

ricalca quella ottenuta per il modello VHDL.