SAPIENZA
UNIVERSITÀ DI ROMA

Progetto di un filtro FIR equalizzatore con coefficienti riconfigurabili e simulazione su Vivado

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica

Corso di Laurea Magistrale in Ingegneria Elettronica

Studenti

Daniele D'Elia,
Alessandro Dei Giudici,
Diletta Lanini

Professore

Giuseppe Scotti

Anno Accademico 2017/2018

Progetto di un filtro FIR equalizzatore con coefficienti riconfigurabili e simulazione su Vivado.
Sapienza – Università di Roma

Versione: 31 marzo 2018

Indice

Capitolo 1

Introduzione

In questo progetto abbiamo realizzato un filtro FIR equalizzatore in grado di compensare le variazioni dell'ingresso introdotte dai canali di comunicazione in modo da ottenere la risposta in frequenza desiderata. Per questa applicazione, poiché i coefficienti del filtro non sono noti a priori ma dipendono dall'ingresso e dalla risposta desiderata, abbiamo optato per un filtro con coefficienti riconfigurabili. Inizialmente abbiamo implementato in MATLAB un algoritmo adattativo per il calcolo dei coefficienti del filtro valutando l'errore di approssimazione. Successivamente siamo passati all'implementazione del filtro in Vivado e, poiché da specifiche è richiesto che i coefficienti siano riconfigurabili, abbiamo realizzato una macchina a stati e una memoria per il caricamento di questi ultimi nel filtro.

Capitolo 2

Specifiche di progetto

Le specifiche per questo progetto sono le seguenti:

- frequenza di campionamento Fs = 250 MHz, assumendo che il clock di sistema sia alla stessa frequenza;
- produrre una risposta in frequenza del filtro equalizzatore tale che siano compensate le variazioni dell'ingresso ovvero sia rispettata la seguente equazione:

$$H_{meas} * H_{eq} = H_{rx}$$

- progettare un filtro FIR che approssimi al meglio la risposta Heq utilizzando 32, 64 e 128 tappe;
- implementare una delle 3 macro FIR con coefficienti riconfigurabili;
- effettuare la simulazione del filtro FIR in Vivado.

Per la realizzazione del filtro seguendo queste specifiche abbiamo utilizzato due software, MATLAB (un ambiente per il calcolo numerico e l'analisi statistica) e Vivado (un prodotto della Xilinx utilizzato per la sintesi e l'analisi di progetti HDL). Nei prossimi capitoli sono illustrati i vari procedimenti con cui si è realizzato il progetto; i risultati mostrati sono relativi alle simulazioni effettuate per due differenti ingressi e rispettive uscite desiderate. Per distinguere i grafici delle due simulazioni denomineremo questi ultimi in base al comportamento del filtro:

- *filtro_eq:* un filtro che compensa le variazioni dell'ingresso restituendo un'uscita prossima a 0 nel range di frequenze desiderato,
- *filtro_lp:* un filtro che compensa uno slope lineare non desiderato restituendo un'uscita passabasso.

Capitolo 3

Modelli teorici e soluzioni adottate

In questo progetto abbiamo utilizzato un filtro FIR nonostante sia i filtri IIR che i filtri FIR possano essere utilizzati per il filtraggio adattivo. Infatti, i filtri FIR sono i più utilizzati essendo intrinsecamente stabili in quanto la loro risposta in frequenza contiene unicamente zeri (forma diretta? la mettiamo?).

In primo luogo è stato necessario capire come ricavare i coefficienti del filtro in base a 2 segnali d'ingresso noti a priori. Tra i molteplici algoritmi adattativi abbiamo scelto l'algoritmo RLS (Recursive Least Squares) che si basa sul metodo dei minimi quadrati ricorsivi. Questo algoritmo permette di minimizzare l'errore tra l'ingresso e la risposta desiderata calcolando ricorsivamente i coefficienti.

L'algoritmo RLS è costituito dall'insieme delle seguenti equazioni:

$$\mathbf{k}(n) = \frac{\lambda^{-1}\mathbf{P}(n-1)\mathbf{u}(n)}{1+\lambda^{-1}\mathbf{u}^H(n)\mathbf{P}(n-1)\mathbf{u}(n)}$$
$$y(n) = \mathbf{w}^T(n-1)\mathbf{u}(n)$$
$$e(n) = d(n) - y(n)$$
$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mathbf{k}(n)e(n)$$
$$\mathbf{P}(n) = \lambda^{-1}\mathbf{P}(n-1) - \lambda^{-1}\mathbf{k}(n)\mathbf{u}^H(n)\mathbf{P}(n-1)$$

con le variabili utilizzate descritte nella seguente tabella:

Variabili	Spiegazione
n	indice temporale corrente
u(n)	vettore dei campioni di ingresso
P(n)	matrice di correlazione inversa
k(n)	vettore del guadagno
w(n)	vettore delle stime delle tappe del filtro
y(n)	uscite filtrate
e(n)	errore stimato
d(n)	risposta desiderata
λ	forgetting factor

Per valutare la bontà del filtro utilizziamo l'errore di approssimazione definito come la differenza tra la risposta in frequenza in uscita dal filtro e quella desiderata, ovvero come parte reale della trasformata veloce di Fourier di e(n).

Infine abbiamo tracciato un diagramma ASM (Algorithmic State Machine) per la realizzazione di una macchina a stati che si occupa di caricare i coefficienti dalla memoria al filtro. Il diagramma è il seguente:

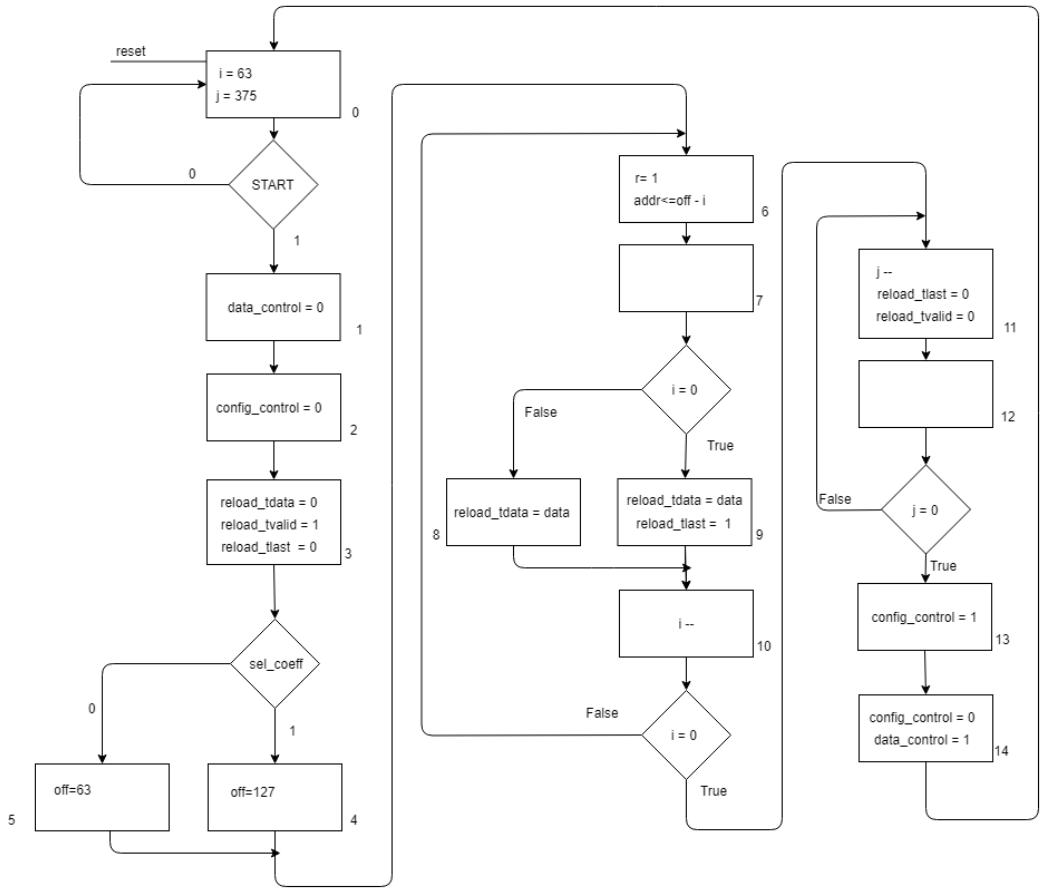

Figura 3.1. Diagramma ASM del modulo per il caricamento dei coefficienti

Al segnale di start, il filtro viene bloccato per tutta la durata del caricamento dei coefficienti. Dopo aver deciso il set di coefficienti tramite l'ingresso sel_coeff, si effettua il caricamento dalla memoria di un coefficiente alla volta. **I salti condizionali sono stati gestiti sempre confrontando la variabile i con il valore 0 in modo da evitare la creazione di comparatori hardware ma utilizzando il bit di segno dell'ALU.** Infine si sblocca il filtro e la macchina a stati resta in attesa del prossimo ciclo di caricamento.

Capitolo 4

Script Matlab

Nella prima parte di questo progetto abbiamo realizzato un algoritmo in ambiente MATLAB (presente in appendice) per la generazione dei coefficienti del filtro, relativi ai due ingressi di test. Tali coefficienti permettono di ottenere, dato un determinato segnale di ingresso, una risposta di uscita desiderata. Nelle specifiche era richiesto di generare tre set di coefficienti a differenti tappe, ovvero 32, 64 e 128. In seguito alla generazione abbiamo valutato il set migliore in base all'errore di approssimazione. L' ingresso e l'uscita desiderata corrispondente sono stati forniti attraverso dei file in formato txt, in cui sono riportati i valori del modulo in dB in funzione della frequenza. Inoltre, nel file del segnale di uscita desiderato, è espresso anche il valore della fase in funzione della frequenza. Abbiamo quindi creato delle strutture MATLAB in cui poter riportare i valori. Di seguito sono illustrati i successivi passi di progetto.

Interpolazione dell'asse delle frequenze

Poiché l'ingresso e l'uscita desiderata del filtro presentavano un range di frequenze differente, abbiamo proceduto con un'interpolazione lineare dell'asse delle frequenze, in modo che presentassero lo stesso numero di valori.

Risposta impulsiva dei set di misura

Avendo ottenuto la risposta in frequenza della misura interpolata, abbiamo ricavato la risposta impulsiva, in quanto l'algoritmo RLS opera nel dominio dell'indice. In primo luogo abbiamo convertito il modulo in lineare; poiché MATLAB utilizza array positivi mentre la risposta in frequenza interpolata presenta frequenze sia positive che negative, è stato necessario riportare tutto l'asse delle frequenze in unico array con indici positivi. Abbiamo così ottenuto una risposta in frequenza simmetrica rispetto al valore centrale dell'array, corrispondente all'ultimo valore della risposta in frequenza positiva. Infine abbiamo ottenuto la risposta impulsiva applicando la trasformata di Fourier inversa all'array. Abbiamo applicato lo stesso procedimento anche alla risposta in frequenza desiderata. Nelle figure seguenti la risposta impulsiva della misura è confrontata con la rispettiva risposta impulsiva desiderata per i due ingressi; si può notare come le risposte impulsive non corrispondano alle rispettive risposte desiderate.

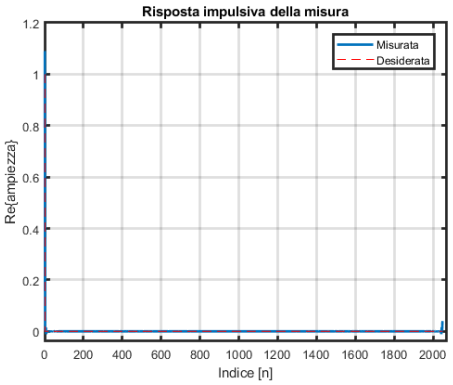

Figura 4.1. Risposta impulsiva per il filtro_eq

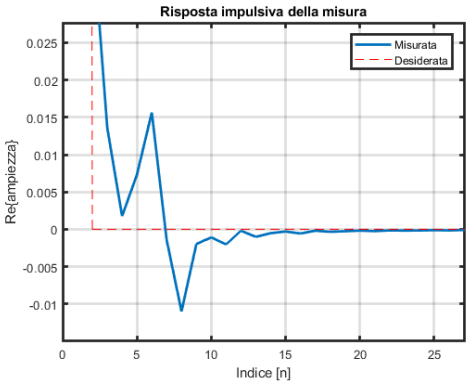

Figura 4.2. Dettaglio risposta impulsiva per il filtro_eq

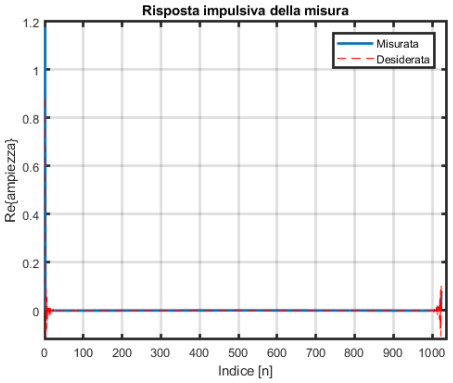

Figura 4.3. Risposta impulsiva per il filtro_lp

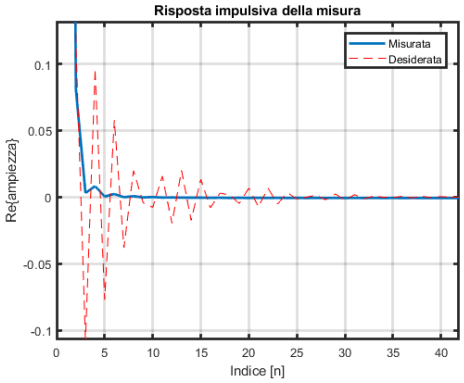

Figura 4.4. Dettaglio risposta impulsiva per il filtro_lp

Algoritmo RLS

Per generare i coefficienti abbiamo utilizzato la funzione dsp.RLSFilter dedicata ai filtri. Fornendo alla funzione la risposta impulsiva della misura e dell'uscita desiderata, abbiamo ottenuto l'uscita filtrata, i coefficienti del filtro e l'errore di approssimazione. In particolare nel nostro script abbiamo generato tre set di coefficienti per ogni ingresso, come richiesto dalle specifiche. Dopo aver generato i coefficienti, abbiamo utilizzato la funzione MATLAB dfilt.dffir per ottenere nel dominio tempo-discreto la risposta impulsiva finita in forma diretta, essendo un filtro FIR.

Figure risposta impulsiva e in frequenza del filtro per i 3 set

Nelle figure seguenti sono mostrati i grafici della risposta impulsiva e della risposta in frequenza del filtro.

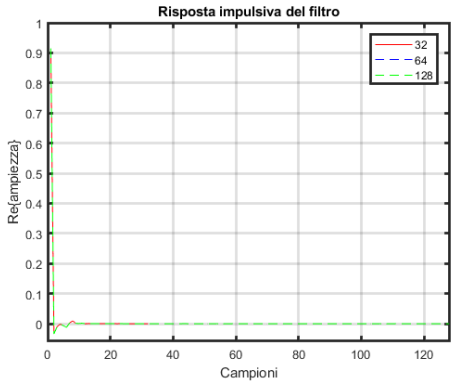

Figura 4.5. Risposta impulsiva per il filtro_eq

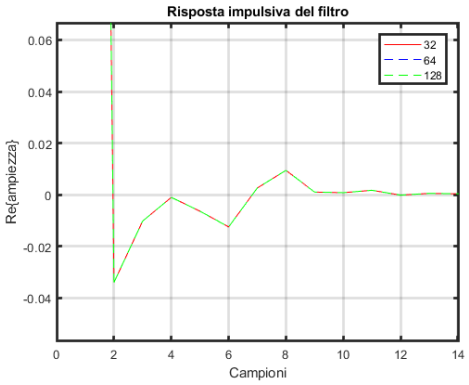

Figura 4.6. Dettaglio risposta impulsiva per il filtro_eq

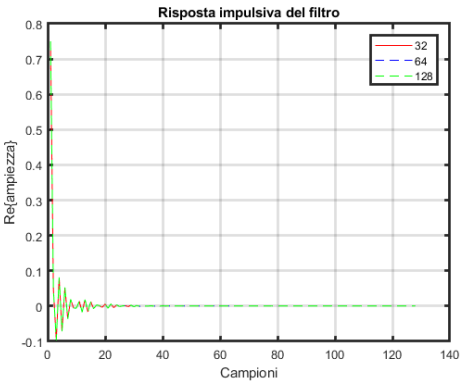

Figura 4.7. Risposta impulsiva per il filtro_lp

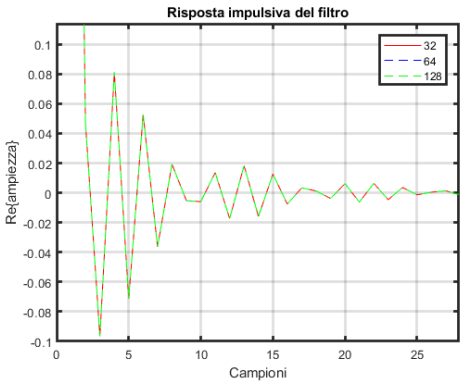

Figura 4.8. Dettaglio risposta impulsiva per il filtro_lp

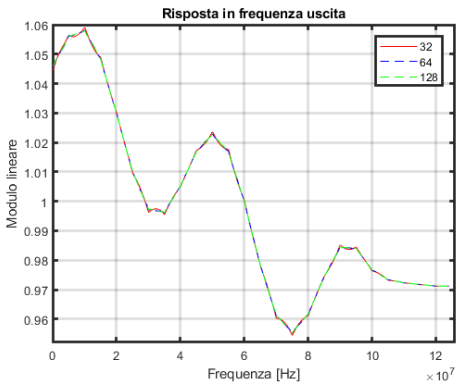

Figura 4.9. Risposta in frequenza per il filtro_eq

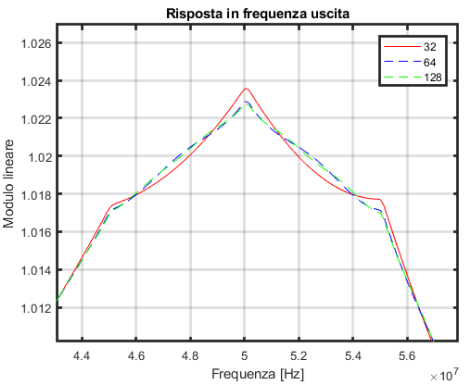

Figura 4.10. Dettaglio risposta in frequenza per il filtro_eq

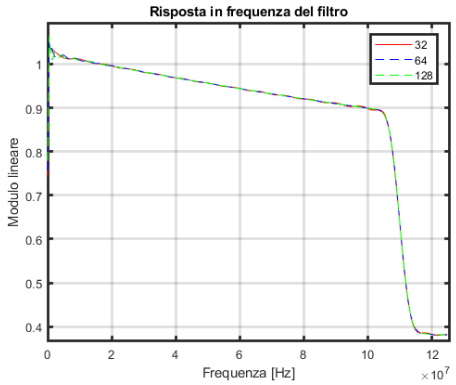

Figura 4.11. Risposta in frequenza per il filtro_lp

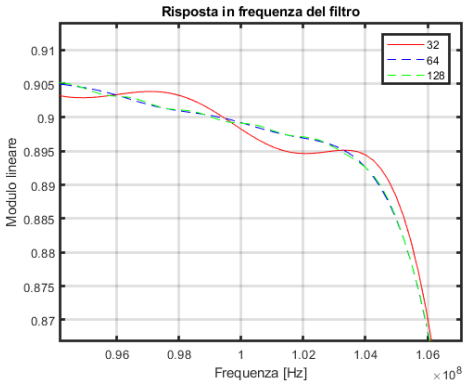

Figura 4.12. Dettaglio risposta in frequenza per il filtro_lp

Come si può notare per ogni simulazione sia la risposta impulsiva che la risposta in frequenza hanno andamenti ragionevoli, ovvero un impulso in tempo e una risposta quasi piatta per l'equallizzatore e un sinc in tempo e una risposta passabasso in frequenza per lpf. Negli ingrandimenti delle immagini possiamo apprezzare le differenze delle risposte con varie tappe, notando che i filtri a 64 e a 128 tappe presentano andamenti quasi completamente sovrapposti mentre il filtro a 32 tappe presenta un andamento più approssimato. Per questi motivi il filtro a 64 tappe risulta il compromesso migliore per la realizzazione del filtro.

confronto errori di approssimazione

Avendo generato i set di coefficienti abbiamo dovuto decidere quale dei tre utilizzare per implementare il filtro con cui operare la successiva simulazione in Vivado.

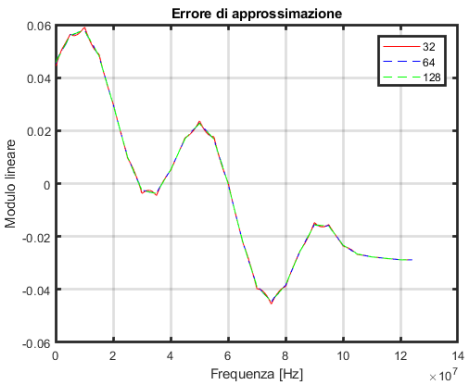

Figura 4.13. Errore di approssimazione per il filtro_eq

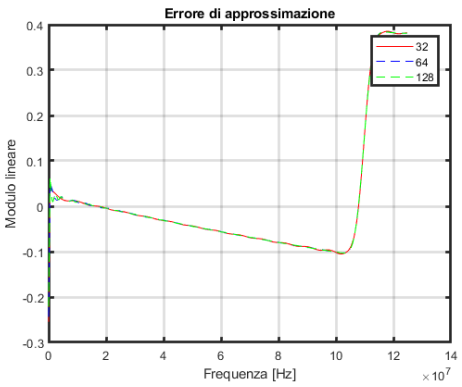

Figura 4.14. Errore di approssimazione per il filtro_lpf

Come è possibile apprezzare dalle precedenti figure l'errore di approssimazione si mantiene minore di 0,1 in lineare per il filtro_eq mentre per il filtro_lpf si mantiene minore di 0,5. Il filtro_lpf ha permesso di effettuare un test più approfondito dell'algoritmo RLS da cui è risultato un errore di approssimazione maggiore rispetto

al primo test ma comunque accettabile. Per ognuna delle implementazioni del filtro (32, 64 e 128 tappe) non si apprezza alcuna differenza degli errori, per cui abbiamo scelto di implementare il filtro a 64 tappe dati i motivi illustrati nel paragrafo precedente.

Capitolo 5

Implementazione su Vivado

Come si può notare dal seguente schematico, il progetto finale è composto da 5 blocchi: un generatore di impulsi, un generatore di clock (non visibile nello schematico), un filtro FIR riconfigurabile, una macchina a stati e una ROM. Ogni componente è stato scritto in VHDL.

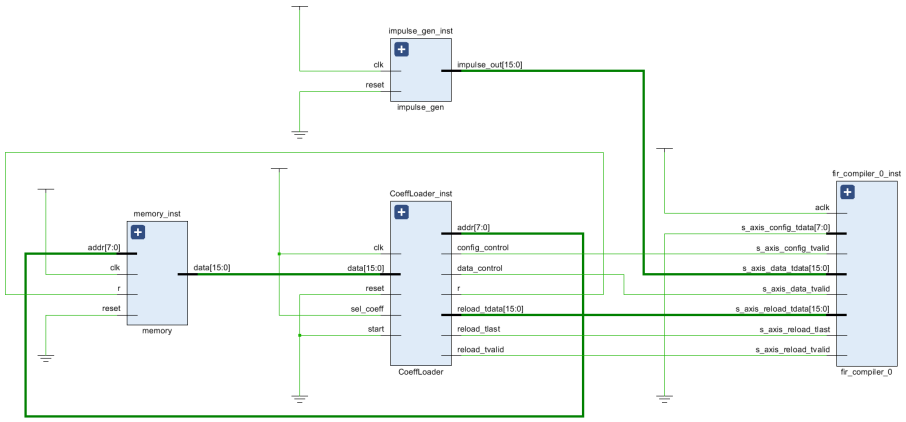

Figura 5.1. Schematico del filtro progettato

ROM

Nella memoria abbiamo inserito i due set di coefficienti ricavati tramite MATLAB. La memoria è solamente leggibile in quanto l'unica operazione da effettuare è prelevare i coefficienti e mandarli in ingresso al filtro FIR.

Generatore di impulsi

Serve a generare periodicamente un impulso digitale da inviare in ingresso al filtro. E' costituito da un contatore che viene incrementato ad ogni ciclo di clock. Quando esso raggiunge il valore 0xFE0 in uscita viene trasmesso il valore 0x7FFF. In tutti gli altri istanti l'uscita è pari a 0.

Macchina a stati

Per tradurre l'ASM di partenza in una macchina a stati abbiamo utilizzato una forma generale di template a 3 processi dove:

- un processo specifica un'unità combinatoria che calcola lo stato prossimo del sistema;
- un processo sensibile al fronte di clock specifica l'assegnazione dello stato prossimo allo stato presente;
- un processo sensibile al fronte di clock specifica l'assegnazione di registri e porte di uscita con risultati di operazioni svolte durante il corrente ciclo di clock.

La macchina a stati si occupa di gestire la trasmissione dei coefficienti al filtro disabilitandolo tramite il segnale data_tvalid, caricare il nuovo set di coefficienti e infine riabilitare il filtro. In particolare, il caricamento del nuovo set avviene tramite i segnali reload_tvalid e reload_tdata, mentre l'attivazione del nuovo set viene effettuata tramite il segnale config_tvalid. Quest'ultimo da datasheet dev'essere attivato dopo un numero di cicli pari a

$$2 \cdot \#coefficienti \cdot \#cicli\_per\_coefficiente. \tag{5.1}$$

Nel nostro caso sarebbero sufficienti circa 320 cicli: per avere un buon margine attendiamo 375 cicli prima di abilitare il filtro.

Generatore di clock

Abbiamo realizzato il generatore del segnale di clock mediante la IP Clocking Wizard. Come da specifiche abbiamo impostato la frequenza di clock a 250 MHz con l'impostazione BUFG in modo da far arrivare il clock a tutti i componenti in contemporanea.

Filtro FIR riconfigurabile

L'ultimo elemento del sistema è il filtro che abbiamo realizzato tramite la IP FIR Compiler. Abbiamo scelto di quantizzare ingresso e uscita con 16 bit e abbiamo impostato il filtro in modo che utilizzi 64 tappe con coefficienti riconfigurabili.

Capitolo 6

Verifica dei risultati e conclusioni

Per verificare il corretto funzionamento del filtro abbiamo realizzato un testbench nel quale effettuiamo il caricamento dei due set di coefficienti consequenzialmente.

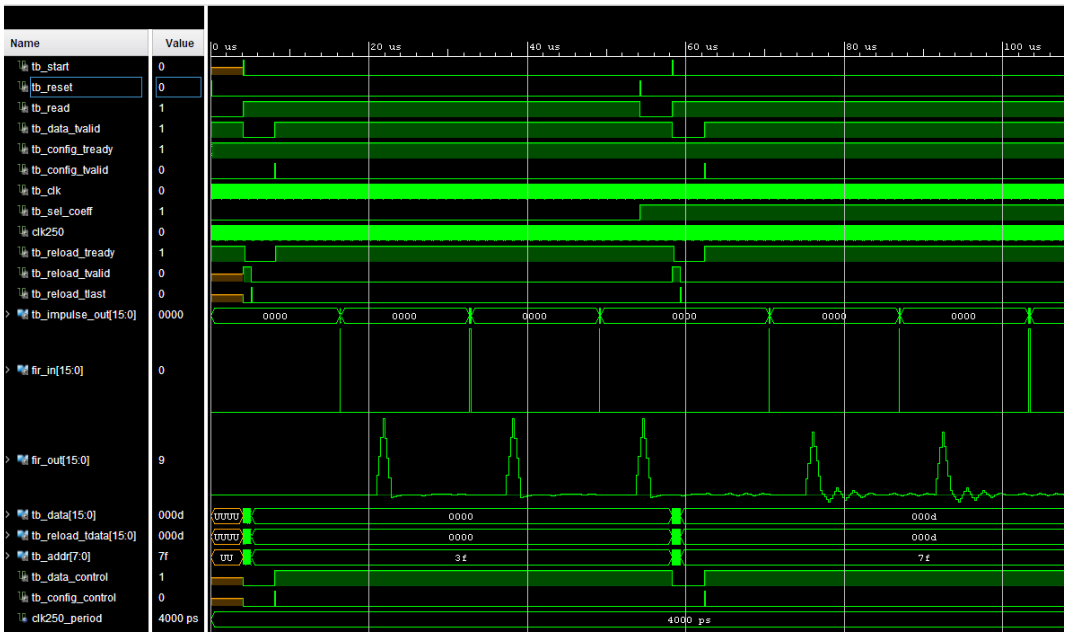

Figura 6.1. Simulazione in Vivado del filtro realizzato

Come possiamo notare dallo screenshot di seguito, le risposte impulsive risultano corrette. La scelta del set dei coefficienti in uso è determinato dal segnale tb_sel_coeff. Tramite fir_in è possibile monitorare l'invio degli impulsi al filtro tramite i quali si ottiene la rispettiva risposta impulsiva fir_out. E' possibile apprezzare due risposte impulsive diverse: una relativa al filtro_eq e l'altra al filtro_lp. La prima rappresenta un impulso in quanto, nel dominio della frequenza, si ottiene una risposta costante e unitaria; la seconda è un sinc in quanto, nel dominio della frequenza, si ottiene un andamento approssimabile ad un rect.

I risultati ottenuti sono in accordo con le specifiche del progetto.

Riferimenti

- *Mauro Olivieri*, Elementi di progettazione dei sistemi VLSI, volume II,
- *V.K. Ingle J.G. Proakis*, Digital Signal Processing using Matlab,
- *Xilinx Vivado Design Suite*, LogiCORE IP Product Guide FIR Compiler v7.2.
- Matlab Guide, https://it.mathworks.com/help/dsp/ref/dsp.rlsfilter-system-object.html