UNIVERSITÀ DI PISA

Computer Engineering

Electronics and Communication systems

Filtro IIR per applicazioni audio

Report

Leonardo Poggiani

Anno accademico: 2020/2021

Indice

Capitolo 1

Introduzione

1.1 Specifiche e requisiti

Viene richiesta la progettazione e l'implementazione di un filtro IIR per applicazioni audio a partire dalla seguente equazione differenziale:

$$y[n] = y[n-1] - \frac{1}{4}x[n] + \frac{1}{4}x[n-4]$$

Questo filtro permette di ridurre le componenti del segnale in input che hanno una frequenza uguale a metá della frequenza di campionamento.
Inoltre é necessario riferirsi ad una possibile applicazione al formato wav a 16 bit.

Figura 1.1: Schema del filtro da progettare

1.2 Filtri digitali

Nell'elaborazione dei segnali, un **filtro digitale** é un sistema che compie delle operazioni matematiche su un segnale a tempo discreto campionato per ridurre o migliorare alcuni aspetti di quel segnale. Questo va in contrasto con l'altro principale tipo di filtro, il **filtro analogico** che tipicamente é un circuito elettronico che opera su segnali analogici a tempo continuo.
Un filtro digitale solitamente consiste in un *convertitore analogico-digitale* (ADC) per campionare il segnale in input, seguito da un *microprocessore* e alcune periferiche come una memoria per immagazzinare i dati e filtrare i coefficienti. Il programma che viene eseguito sul microprocessore implementa il filtro digitale andando a svolgere le necessarie operazioni matematiche sui numeri ricevuti dal ADC.
I filtri digitali possono solitamente essere piú costosi rispetto agli equivalenti filtri analogici a causa della loro crescente complessitá ma hanno reso possibili molti progetti che sono infattibili o troppo complessi con i filtri analogici. I filtri digitali non sono soggetti alle componenti non lineari che invece complicano molto il progetto di un filtro analogico.
I filtri analogici sono realizzati con componenti elettroniche imperfette, i cui valori sono specificati attraverso delle tolleranze e che possono cambiare con la temperatura o abbassarsi con il tempo. Se l'ordine del filtro analogico cresce insieme al suo numero di componenti, l'effetto delle componenti di errore variabile sono ingranditi. Nei filtri digitali, i valori dei coefficienti sono conservati nella memoria di un computer rendendoli molto piú stabili e predicibili.
Poiché i coefficienti dei filtri digitali sono definiti, possono essere utilizzati per ottenere design molto più complessi e selettivi e in particolar modo con i filtri digitali si possono ottenere minore ondulazione della banda passante, transizione più veloce e maggiore attenuazione della banda di stop rispetto ai filtri analogici. Anche se il design si potesse ottenere usando filtri analogici, i costi di progettazione dell'equivalente filtro digitale sono molto minori. In piú si possono facilmente modificare i coefficienti di un filtro digitale per renderlo un filtro adattivo o filtro parametrico, controllato dall'utente.
Quando vengono usanti in contesti come sistemi analogici in tempo reale, i filtri digitali possono mostrare delle latenze problematiche (differenza di tempo tra l'input e la risposta) a causa delle conversioni analogico-digitale e digitale-analogico e ai filtri anti-aliasing.
Un filtro digitale é caratterizzato dalla sua *funzione di trasferimento* o in modo equivalente, dalla sua *equazione differenziale*. L'analisi matematica della sua funzione di trasferimento puó descrivere come risponderá ad ogni tipo di input.

1.3 Filtri IIR

I filtri IIR sono molto utili quando abbiamo bisogno di velocitá elevate dato che di solito richiedono un minor numero di moltiplicatori rispetto ai filtri FIR.
Un filtro Finite Impulse Response (FIR) é un filtro la cui risposta all'impulso (o la sua risposta a qualsiasi tipo di input di lunghezza finita) ha una durata finita, poiché si azzera in un tempo finito. Questo non succede nei filtri Infinite Impulse Response (IIR) che possono avere dei feedback interni e quindi continuare a rispondere all'infinito, solitamente attenuandosi.
I filtri IIR possono essere progettati in modo da avere una risposta in frequenza che é la versione discreta della risposta in frequenza di un filtro analogico. Questi filtri sono molto sensibili agli errori di quantizzazione dei coefficienti che avvengono quando si usa un numero finito di bit per rappresentare i coefficienti dei filtri.
La tipica equazione differenziale di un filtro FIR é la seguente:

$$y[n] = b_0x[n] + b_1x[n-1] + ... + b_Nx[n-N] = \sum_{i=0}^{N} b_ix[n-i]$$

Dove:

- **x[n]** é il segnale di input
- **y[n]** é il segnale di output
- **N** é l'ordine del filtro
- b_i é il valore della risposta all'impulso all'istante i-esimo per $0 <= i <= N$ di un filtro di ordine N.

Mentre la tipica equazione differenziale di un filtro IIR é la seguente:

$$y[n] = \frac{1}{a_0}(b_0x[n] + b_1x[n-1] + ... + b_Px[n-P] - a_1y[n-1] - a_2y[n-2] - ... - a_Qy[n-Q])$$

Dove:

- **P** é l'ordine del filtro feedforward
- b_i sono i coefficienti del filtro feedforward
- **Q** é l'ordine del filtro feedback
- a_i sono i coefficienti del filtro feedback.

Possiamo condensare la formula come segue:

$$y[n] = \frac{1}{a_0}(\sum_{i=0}^{P} b_ix[n-i] - \sum_{j=0}^{q} a_jy[n-j])$$

Sappiamo che la funzione di trasferimento puó essere definita come:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{i=0}^{P} b_i z^{-i}}{\sum_{j=0}^{Q} b_j z^{-j}}$$

La progettazione di un filtro digitale necessita di cinque step:

- **Specifica del filtro:** Questo puó includere definire il tipo di filtro, ad esempio un filtro passabasso, l'ampiezza desiderata, la risposta in fase e le tolleranze, la frequenza di campionamento, la lunghezza dei dati in input.
- **Calcolo del coefficiente del filtro:** Il coefficiente di una funzione di trasferimento H(z) deve rispettare le specifiche stabilite. La scelta del metodo per il calcolo del coefficiente é influenzata da vari fattori.
- **Realizzazione:** Si tratta di convertire la funzione di trasferimento in un filtro adatto alle specifiche fornite.
- **Analisi:** Viene analizzato l'effetto della quantizzazione nei coefficienti del filtro e sui dati in input cosí come l'effetto che ha sulle performance l'uso di parole di lunghezza fissata.
- **Implementazione:** La produzione del codice o dell'hardware che costituisce il filtro e il suo utilizzo effettivo.

1.4 Formato audio WAV

Waveform Audio File Format è un formato di file audio standard, sviluppato da IBM e Microsoft, per memorizzare un bitstream audio sui PC. É il formato principale usato sui sistemi Microsoft Windows per l'audio non compresso. La codifica usuale del bitstream è il formato LPCM (linear pulse-code modulation).

WAV è un'applicazione del metodo di formato bitstream Resource Interchange File Format (RIFF) per la memorizzazione di dati in blocchi, e quindi è simile al formato 8SVX e AIFF usato rispettivamente sui computer Amiga e Macintosh.

Anche se un file WAV può contenere audio compresso, il formato audio WAV più comune è l'audio non compresso nel formato LPCM (linear pulse-code modulation). LPCM è anche il formato di codifica audio standard per i CD audio, che memorizzano audio LPCM a due canali campionato a 44100 Hz con 16 bit per campione. Poiché LPCM non è compresso e conserva tutti i campioni di una traccia audio, gli utenti professionali o gli esperti di audio possono usare il formato WAV con l'audio LPCM per la massima qualità audio. I file WAV possono anche essere modificati e manipolati con relativa facilità usando

un software.

Il formato WAV supporta l'audio compresso usando, su Microsoft Windows, l'Audio Compression Manager. Qualsiasi codec ACM può essere usato per comprimere un file WAV. L'interfaccia utente (UI) di Audio Compression Manager può essere accessibile attraverso vari programmi che lo utilizzano, incluso Sound Recorder in alcune versioni di Windows.

A partire da Windows 2000, è stata definita un'intestazione WAVE_FORMAT_EXTENSIBLE che specifica i dati dei canali audio multipli insieme alle posizioni degli altoparlanti, elimina l'ambiguità riguardo ai tipi di campioni e alle dimensioni dei contenitori nel formato WAV standard e supporta la definizione di estensioni personalizzate del chunk del formato.

Ci sono alcune incongruenze nel formato WAV: per esempio, i dati a 8 bit sono senza segno mentre quelli a 16 bit sono con segno, e molti chunks duplicano informazioni trovate in altri chunks.

1.5 Linear pulse code modulation (LPCM)

La modulazione a codice d'impulsi (PCM) è un metodo usato per rappresentare digitalmente segnali analogici campionati. È la forma standard di audio digitale in computer, compact disc, telefonia digitale e altre applicazioni audio digitali. In un flusso PCM, l'ampiezza del segnale analogico viene campionata regolarmente a intervalli uniformi, e ogni campione viene quantizzato al valore più vicino all'interno di una gamma di passi digitali.

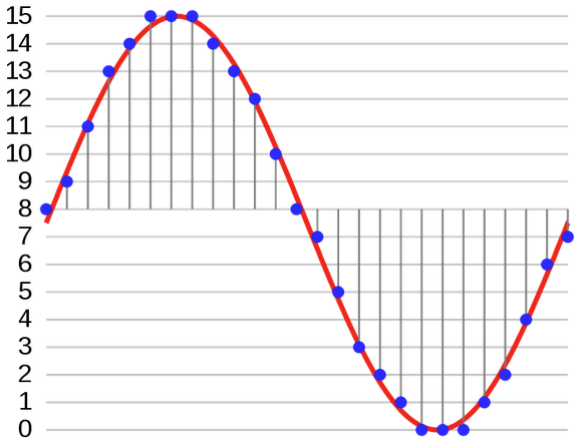
La modulazione lineare di codice a impulsi (LPCM) è un tipo specifico di PCM in cui i livelli di quantizzazione sono linearmente uniformi. Questo è in contrasto con le codifiche PCM in cui i livelli di quantizzazione variano in funzione dell'ampiezza (come con l'algoritmo A-law o l'algoritmo μ-law). Anche se PCM è un termine più generale, è spesso usato per descrivere i dati codificati come LPCM.

Un flusso PCM ha due proprietà di base che determinano la fedeltà del flusso al segnale analogico originale: la frequenza di campionamento, che è il numero di volte al secondo che i campioni sono presi; e la profondità di bit, che determina il numero di possibili valori digitali che possono essere usati per rappresentare ogni campione.

1.6 Modulazione

Nel diagramma, un'onda sinusoidale (curva rossa) è campionata e quantizzata per PCM. L'onda sinusoidale viene campionata a intervalli regolari, mostrata come linee verticali. Per

ogni campione, viene scelto uno dei valori disponibili (sull'asse y). Il processo PCM è comunemente implementato su un singolo circuito integrato chiamato convertitore analogico-digitale (ADC).



Questo produce una rappresentazione completamente discreta del segnale d'ingresso (punti blu) che può essere facilmente codificata come dati digitali per la memorizzazione o la manipolazione. Diversi flussi PCM potrebbero anche essere multiplexati in un più grande flusso di dati aggregati, generalmente per la trasmissione di più flussi su un singolo collegamento fisico. Una tecnica è chiamata time-division multiplexing (TDM) ed è ampiamente utilizzata, in particolare nel moderno sistema telefonico pubblico.

1.7 Demodulazione

L'elettronica coinvolta nella produzione di un segnale analogico accurato dai dati discreti è simile a quella usata per generare il segnale digitale. Questi dispositivi sono convertitori digitale-analogico (DAC). Producono una tensione o una corrente (a seconda del tipo) che rappresenta il valore presentato sui loro ingressi digitali. Questa uscita verrebbe poi generalmente filtrata e amplificata per essere utilizzata.

Per recuperare il segnale originale dai dati campionati, un demodulatore può applicare la procedura di modulazione al contrario. Dopo ogni periodo di campionamento, il demodulatore legge il valore successivo e transiziona il segnale di uscita al nuovo valore. Come risultato di queste transizioni, il segnale conserva una quantità significativa di energia ad alta frequenza dovuta agli effetti di imaging. Per rimuovere queste frequenze indesiderate, il demodulatore passa il segnale attraverso un filtro di ricostruzione che sopprime l'energia al di fuori della gamma di frequenza prevista (maggiore della frequenza di Nyquist $f_s/2$).

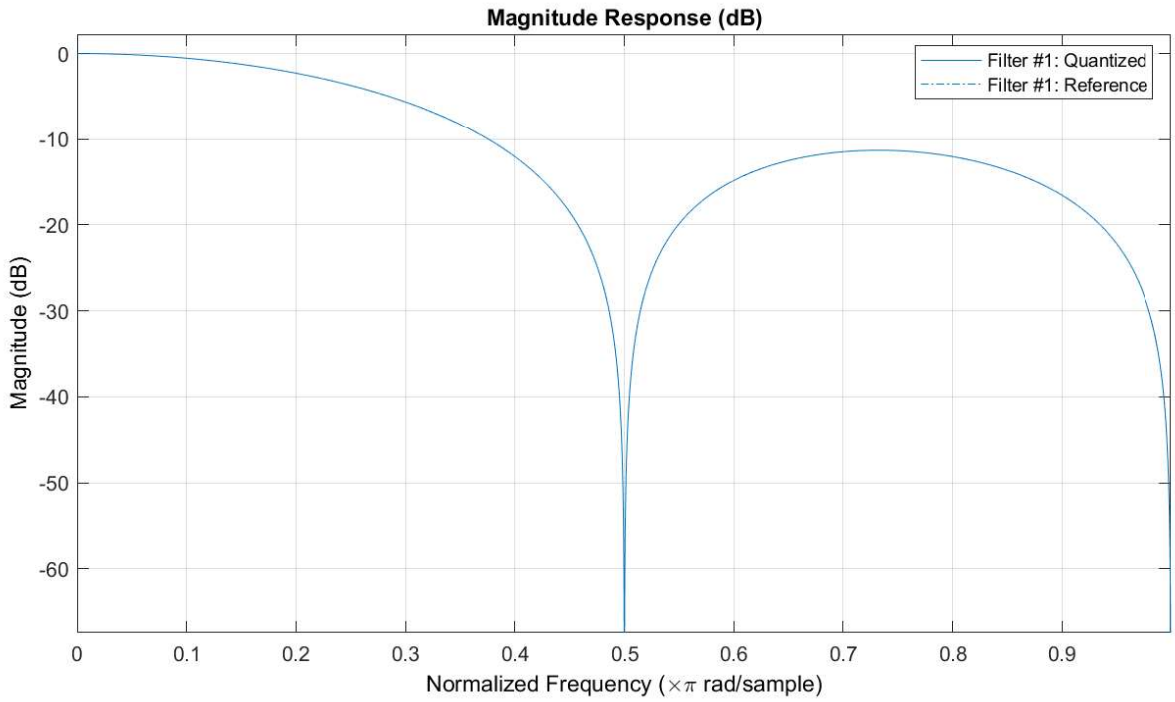
Capitolo 2

Design

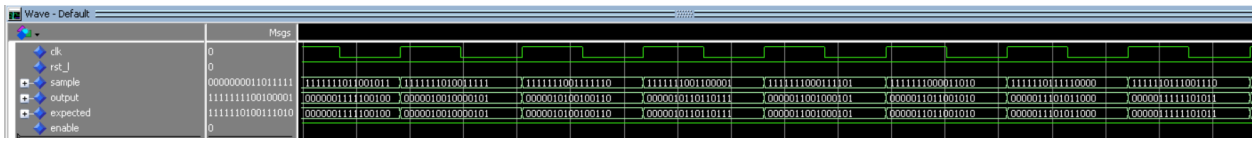
Partendo dall'equazione differenziale si puó derivare la funzione di trasferimento a tempo discreto:

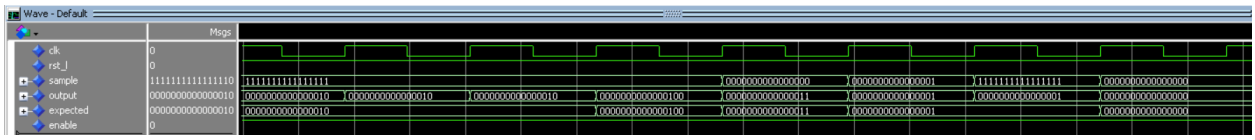
$$H(z) = \frac{Y(z)}{X(z)} = \frac{\frac{1}{4}z^{-4} - \frac{1}{4}}{1 - z^{-1}}$$

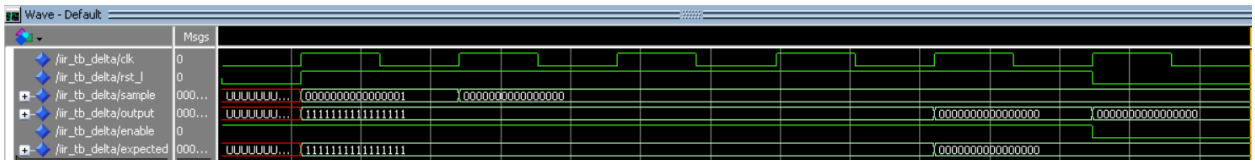
Il design é stato parzialmente svolto anche in Matlab per ottenere i dati necessari alla creazione dei testbench per verificare che il filtro progettato rispettasse le specifiche richieste.
Il filtro in Matlab é stato progettato con un filtro a tempo discreto in forma diretta II, usando il comando *dfilt.df2* . La scelta dei coefficienti é stata fatta rispettando le esigenze progettuali del filtro reale e ne é stata analizzata la risposta in frequenza.



É stato possibile anche reperire in rete due semplici script che permettessero di prendere in ingresso due file audio in formato WAV e convertirli in file .txt.
In questo modo é possibile leggere un file di tipi WAV come un array di int16, convertirlo in valori 16 bit signed fixed e successivamente passarli come input al filtro. L'output sará salvato sia in formato WAV che in formato testuale, per essere dato come input ai testbench che andremo a svolgere.


Wave - Default
Msgs
clk
rst_l
sample
output
expected
enable


Wave - Default
Msgs
clk
rst_l
sample
output
expected
enable


Wave - Default
Msgs
/iir_tb_delta/clk
/iir_tb_delta/rst_l
/iir_tb_delta/sample
/iir_tb_delta/output
/iir_tb_delta/enable
/iir_tb_delta/expected

Capitolo 3

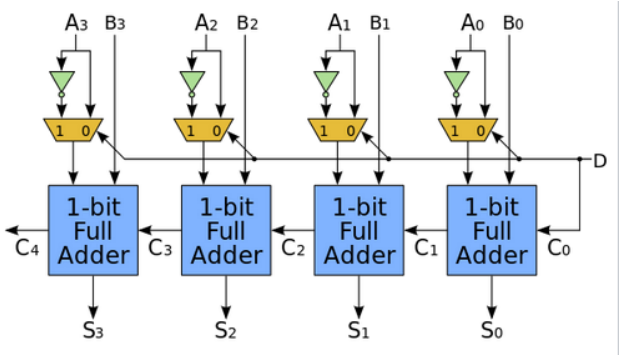
Implementazione

L'implementazione del filtro IIR é visibile nel file IIR.vhd. Verranno riportati nella documentazione solo i passaggi principali del codice.

Di seguito é stata riportata la struttura dell'entitá principale.

```
entity IIR is
        generic (Nbit : natural := 8);
        port (
        clk     :       in      std_logic;
        rst_l   :       in      std_logic;
        x       :       in      std_logic_vector(Nbit-1 downto 0);
        y       :       out     std_logic_vector(Nbit-1 downto 0)
        );
end IIR;
```

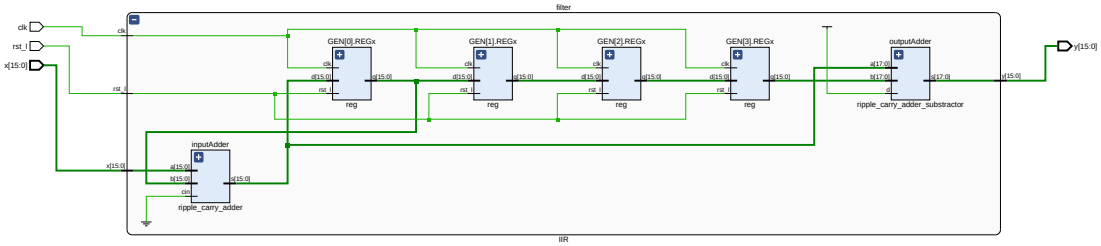
Durante tutta la fase di implementazione si é pensato di fornire una struttura generica, anche se non direttamente specificato nei requisiti, al fine di permettere eventuali aggiornamenti nel codice in futuro.
All'ingresso abbiamo un ripple carry adder di dimensione N.
La moltiplicazione per i coefficienti é ottenuta tramite segnali dato che puó essere calcolata semplicemente come un right shift di 2 bit. Dopo lo shift i segnali sono su N + 2 bit e entrano nel adder-subtractor. Un adder-subtractor é un circuito capace di sommare o sottrarre numeri in base al valore del segnale di controllo.



In questo caso l'adder subtractor é consigliato per risparmiare un adder e di conseguenza ridurre la complessitá. Infatti sarebbero serviti due adder, uno per effettuare la moltiplicazione (da 2-complementare) e uno per la somma finale.

```
entity full_adder_substractor is
        port (
                a       :       in      std_logic;
                b       :       in      std_logic;
                d       :       in      std_logic;
                cin     :       in      std_logic;
                s       :       out std_logic;
                cout    :       out std_logic
        );
end full_adder_substractor;
```

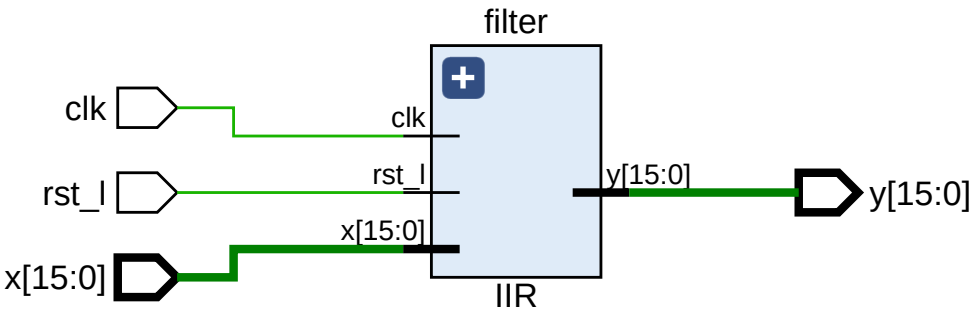
L'ingresso **a** sará preso cosí come é se **d** é uguale a zero, mentre verrá invertito se **d** é uguale a uno, in quanto stiamo richiedendo una sottrazione.


filter
clk
rst_i
x[15:0]
y[15:0]
GEN[0].REGx
GEN[1].REGx
GEN[2].REGx
GEN[3].REGx
reg
outputAdder
ripple_carry_adder_substractor
inputAdder
ripple_carry_adder
IIR

Capitolo 4

Sintesi

Prima di procedere con la sintesi é necessario scrivere un wrapper per il filtro, in quanto é stato progettato con una dimensione di tipo *generic* per avere una maggiore flessibilitá.

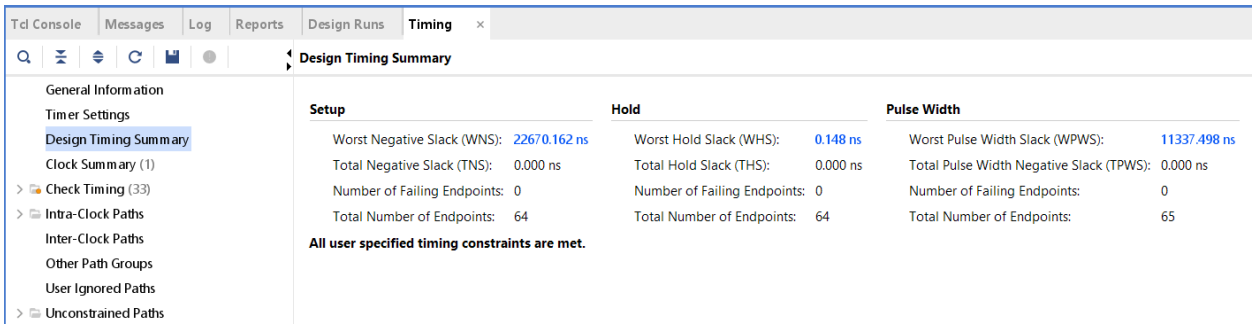


La frequenza di funzionamento del filtro deve essere di 44100 Hz, quindi il periodo di clock deve essere:

$$t_{ck} = \frac{1}{44100} \approx 22676ns$$

É stato di conseguenza creato un clock constraint che rispettasse questo valore e si é

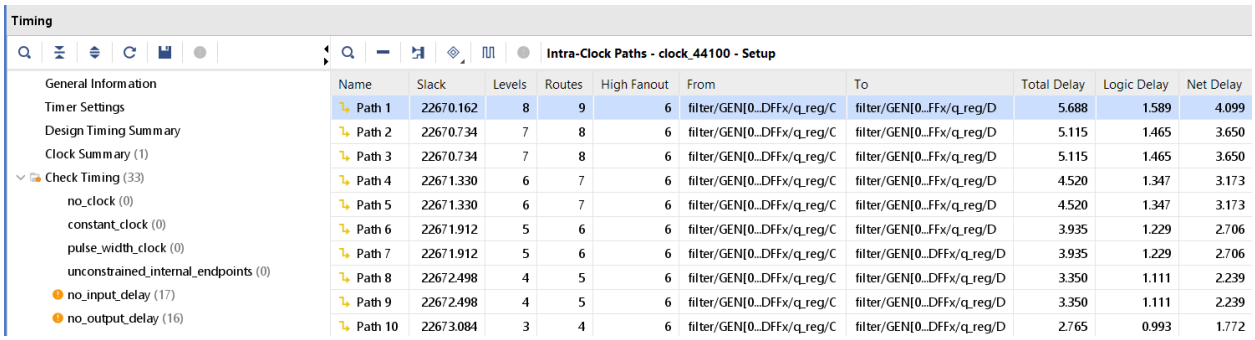
analizzato il timing report generato:



Come possiamo vedere il Worst Negative Slack (WNS) é pari a 22670.162 ns e questo ci porta a concludere che la massima frequenza di funzionamento del nostro circuito é di:

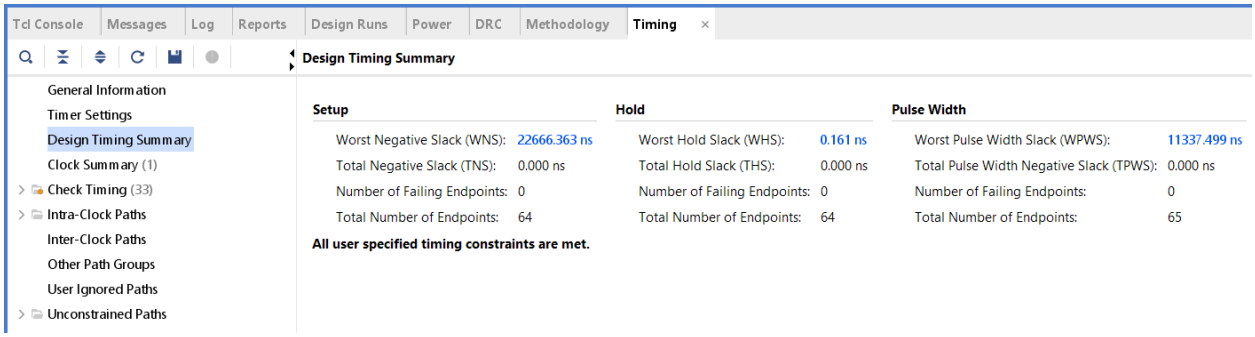
$$f_{max} = \frac{1}{(22676 - 22670.162) \cdot 10^{-9}} \approx 171MHz$$

Il path critico é evidenziato dalla figura seguente:



4.1 Implementazione

La fase di implementazione é terminata senza errori e ha permesso il calcolo del delay reale dovuto alla rete di interconnessione. In particolare riportiamo ancora il timing report generato:



Dopo l'implementazione siamo in grado di analizzare il ritardo dovuto alla rete di interconnesione e calcolare quindi in maniera più accurata la frequenza di funzionamento massima.

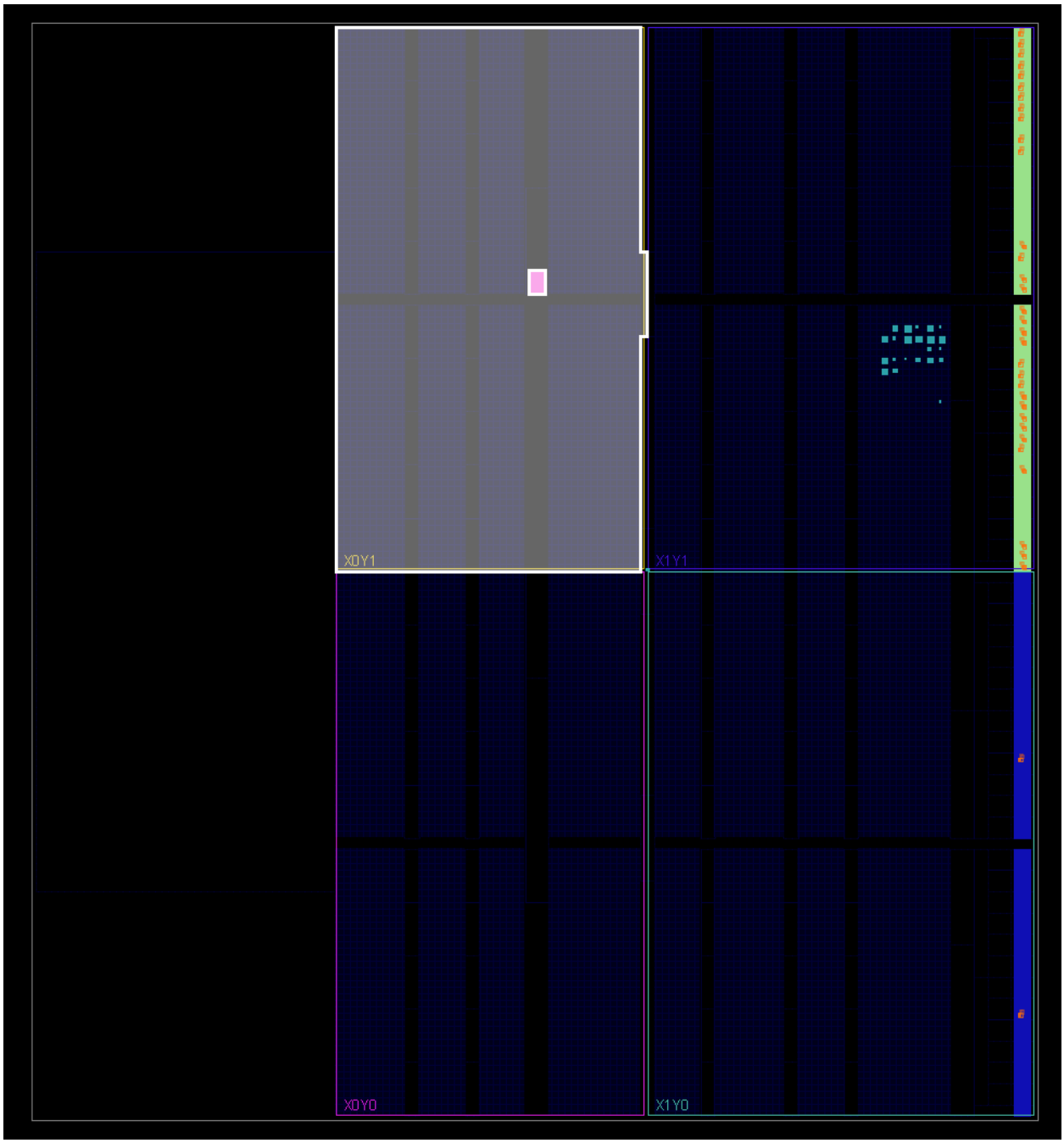
$$f_{max} = \frac{1}{(22676 - 22666.363) \cdot 10^{-9}} \approx 104MHz$$

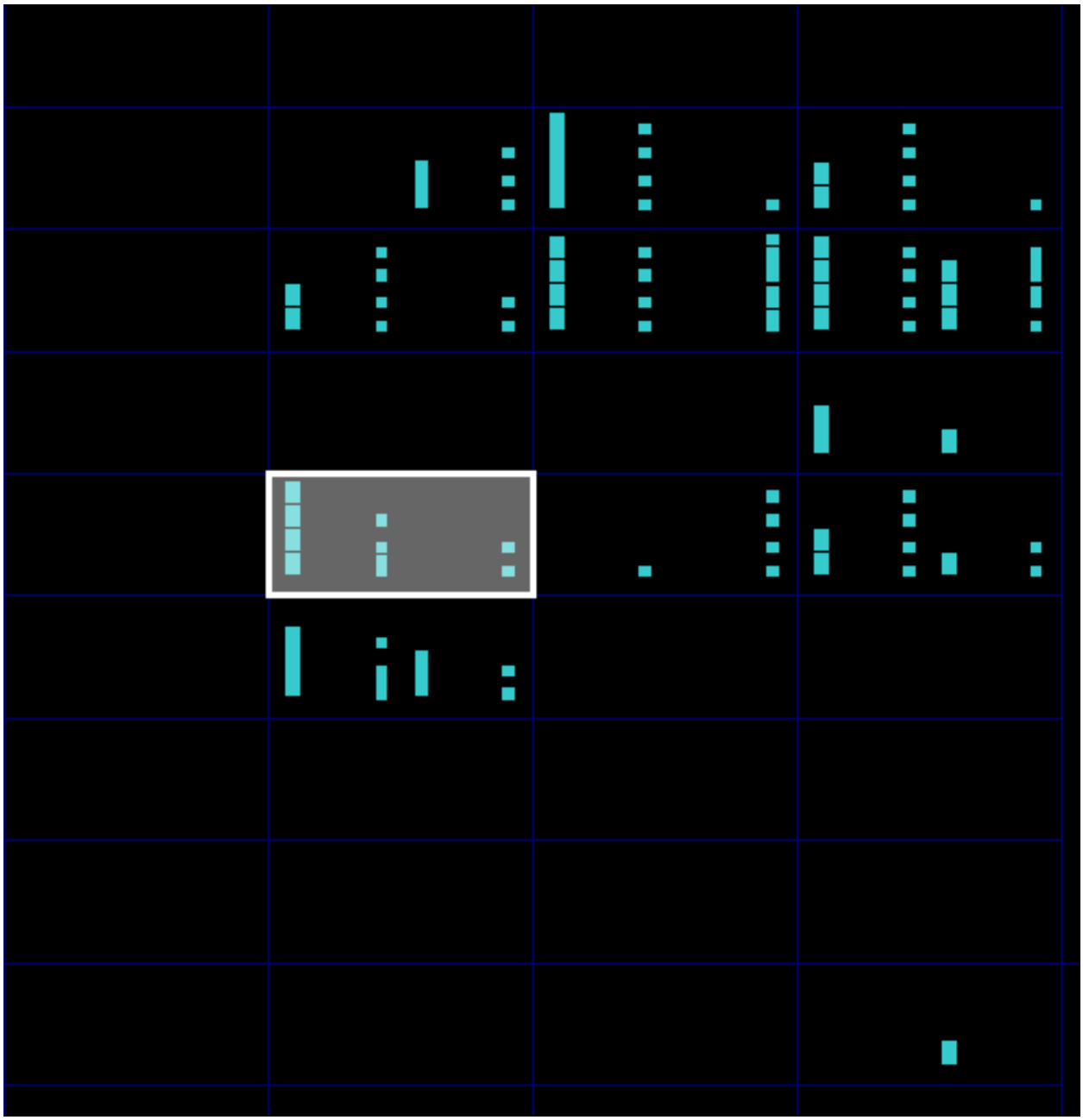
Tcl Console | Messages | Log | Reports | Design Runs | Timing | Power | Methodology | DRC | Package Pins | I/O Ports

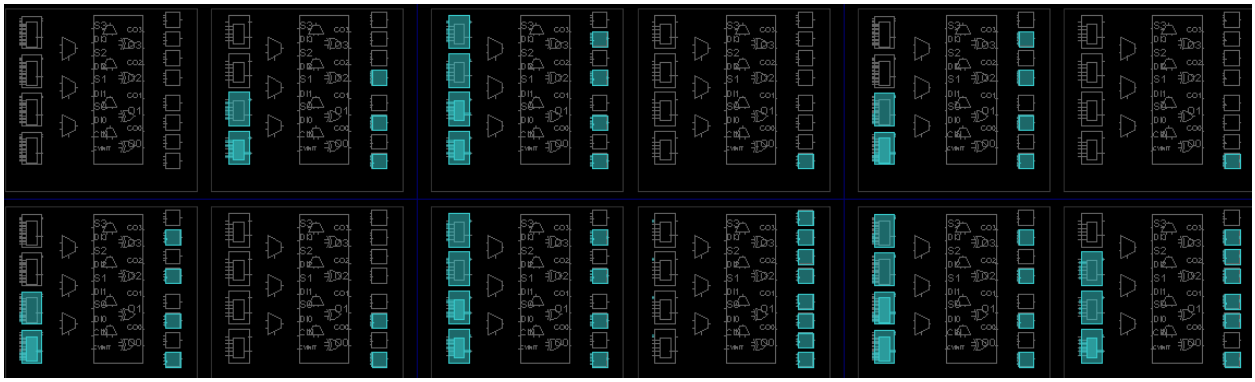
Intra-Clock Paths - clk44100 - Setup

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock
Path 1	22666.615	7	8	8	filter/GEN[0...DFFx/q_reg/C	filter/GEN[0...FFx/q_reg/D	9.218	2.242	6.976	22676.0	clk44100	clk44100
Path 2	22666.920	8	9	8	filter/GEN[0...DFFx/q_reg/C	filter/GEN[0...FFx/q_reg/D	9.051	2.366	6.685	22676.0	clk44100	clk44100
Path 3	22667.275	6	7	8	filter/GEN[0...DFFx/q_reg/C	filter/GEN[0...FFx/q_reg/D	8.417	2.144	6.273	22676.0	clk44100	clk44100
Path 4	22667.564	7	7	8	filter/GEN[0...DFFx/q_reg/C	filter/GEN[0...FFx/q_reg/D	8.406	2.242	6.164	22676.0	clk44100	clk44100
Path 5	22668.244	6	7	8	filter/GEN[0...DFFx/q_reg/C	filter/GEN[0...FFx/q_reg/D	7.666	2.118	5.548	22676.0	clk44100	clk44100
Path 6	22669.104	5	6	8	filter/GEN[0...DFFx/q_reg/C	filter/GEN[0...DFFx/q_reg/D	6.743	1.766	4.977	22676.0	clk44100	clk44100
Path 7	22669.121	5	6	8	filter/GEN[0...DFFx/q_reg/C	filter/GEN[0...FFx/q_reg/D	6.749	1.766	4.983	22676.0	clk44100	clk44100
Path 8	22670.443	4	5	8	filter/GEN[0...DFFx/q_reg/C	filter/GEN[0...DFFx/q_reg/D	5.426	1.412	4.014	22676.0	clk44100	clk44100
Path 9	22671.064	4	5	8	filter/GEN[0...DFFx/q_reg/C	filter/GEN[0...DFFx/q_reg/D	4.788	1.412	3.376	22676.0	clk44100	clk44100
Path 10	22671.762	3	4	8	filter/GEN[0...DFFx/q_reg/C	filter/GEN[0...DFFx/q_reg/D	4.091	1.058	3.033	22676.0	clk44100	clk44100

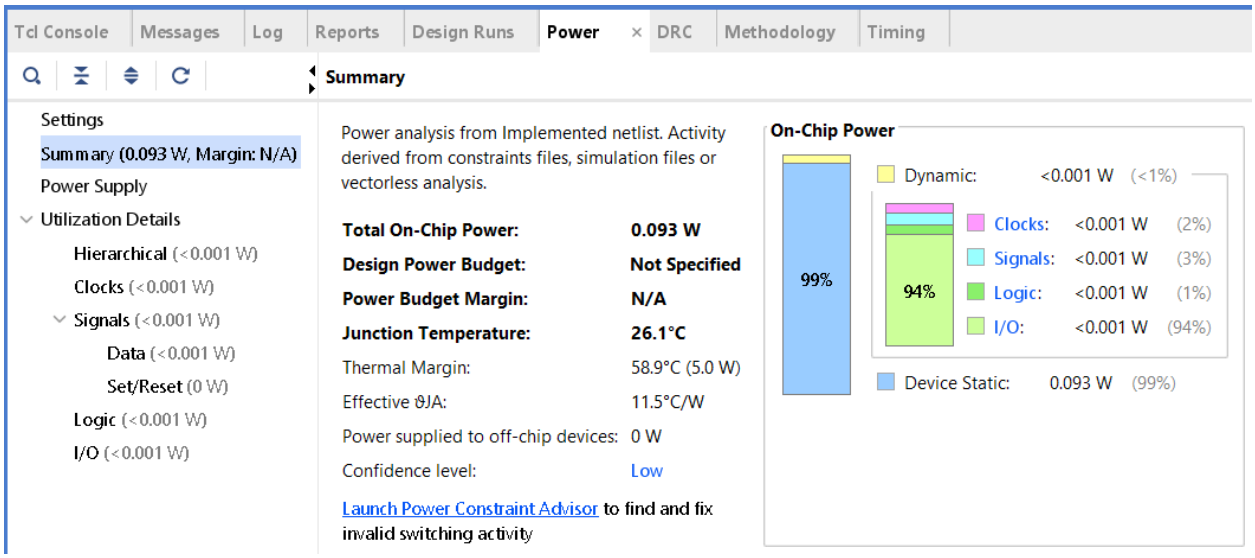
A seguire degli screen che mostrano l'area occupata dal filtro realizzato.


X0Y1
X1Y1
X0Y0
X1Y0



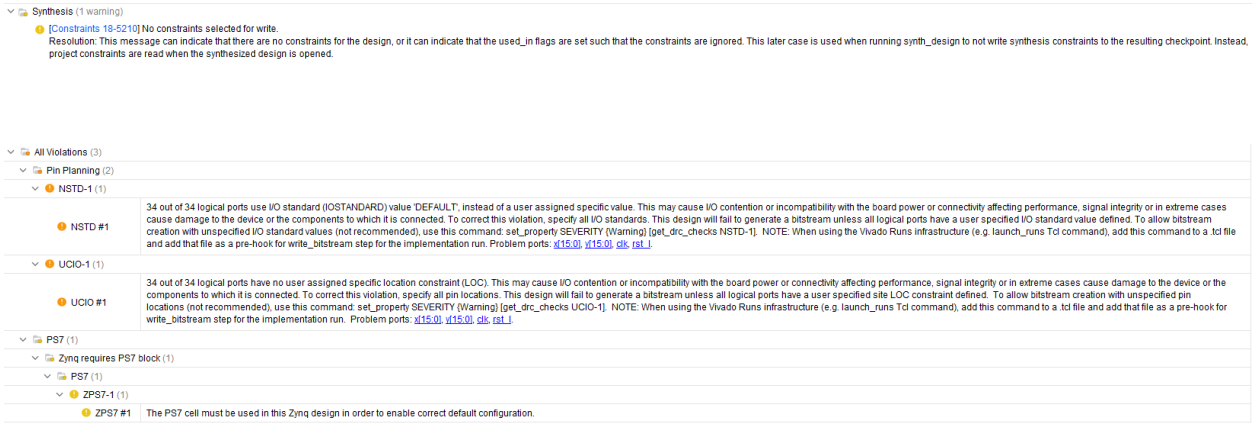


Il power report è stato generato usando una tensione di 3.3 V:



4.2 Warning ed errori

Il processo di implementazione e sintesi non ha portato a errori rilevanti o warning. Gli unici warning rilevati sono stati quelli già discussi durante le ore di lezione, dovuti al fatto che non era richiesto l'assegnamento dei pin di input e output (non era necessaria la creazione del bitstream).



Capitolo 5

Conclusioni

In conclusione, è stato realizzato un filtro IIR per applicazioni audio con una struttura modulare e generica in rispetto dei requisiti definiti dalle specifiche.
Si sarebbero potuti tentare altri tipi di implementazione, volti a rendere il filtro più generico o efficiente ma questo esulava dall'obbiettivo del lavoro effettuato.
Le verifiche effettuate attraverso i testbench hanno mostrato come il filtro progettato sia in grado di funzionare con file WAV reali e produrre una traccia filtrata come output.