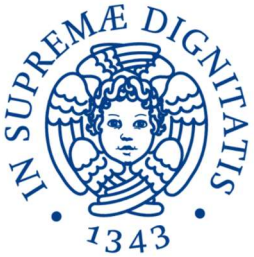


UNIVERSITÀ DI PISA

SCUOLA DI INGEGNERIA

MASTER OF SCIENCE IN COMPUTER ENGINEERING

RELAZIONE DI PROGETTO

ELECTRONICS AND COMMUNICATION SYSTEM

Progetto e sintesi di un Filtro FIR passa basso

Antonio Di Tecco

Indice

1. Filtro FIR

In tutti i sistemi dove si vuole che il transito di segnali non desiderati venga attenuato il più possibile ed i segnali voluti abbiano la minore attenuazione possibile occorre utilizzare i filtri. Il filtro a impulsi finiti (filtro FIR) è una struttura di un filtro che può essere implementata digitalmente in quasi ogni tipo di frequenza utilizzando una serie di ritardi, moltiplicatori e sommatori per generare l'output del filtro. Lo schema base del filtro è presentato in fig. 1.

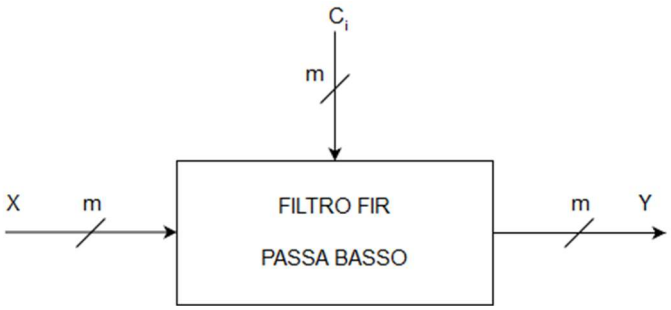


Fig.1 Esempio di filtro FIR di ordine N.

L'uscita discreta del filtro $y[n]$ è descritto dalla convoluzione $c * x[n]$. Per cui l'uscita sarà pari alla somma pesata dei valori assunti dall'ingresso al tempo corrente e a tempi precedenti. Tale operazione è descritta dalla seguente equazione:

$$y[n] = c_0 x[n] + c_1 x[n-1] + \ldots + c_N x[n-N] = \sum_{i=0}^{N} c_i x[n-i] \qquad (1)$$

Un filtro di ordine *N* è caratterizzato da *N+1* coefficienti e in genere richiede *N+1* moltiplicatori e *N* sommatori. I valori $x[n-i]$ sono comunemente chiamati "*taps*", quindi un filtro di ordine N=6 avrà 7 tap.

1.1 Applicazioni

Il filtro FIR è ampiamente utilizzato in molte applicazioni digitali, per esempio in speech processing, equalizzazione, rimozione dell'eco, del rumore e così via. Per queste applicazioni sono richiesti filtri di ordine elevato per incontrare gli stringenti vincoli in frequenza.

1.2 Architettura

Il filtro FIR può essere implementato usando differenti strutture.

1.2.1 Filtro in forma diretta

Questa è chiamata forma diretta perché è l'implementazione diretta della convoluzione.
Il numero di elementi di ritardo Δ è l'ordine del filtro, per cui questa è anche chiamata forma canonica. In fig. 2 è possibile vedere la struttura della forma diretta per un filtro N=4, dove la risposta $y[n]$ è la stessa dell'equazione (1).

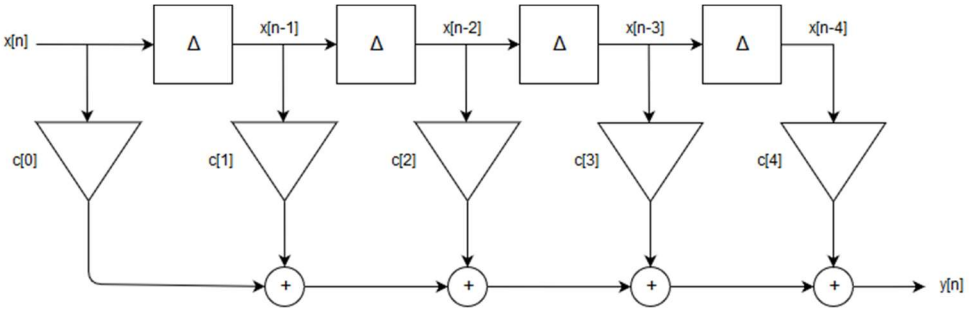


Fig.2 Esempio di architettura in forma diretta di filtro FIR di ordine 4.

1.2.2 Filtro in forma diretta a cascata

La trasposizione della forma diretta sopra mostrata è rappresentata in fig.3.

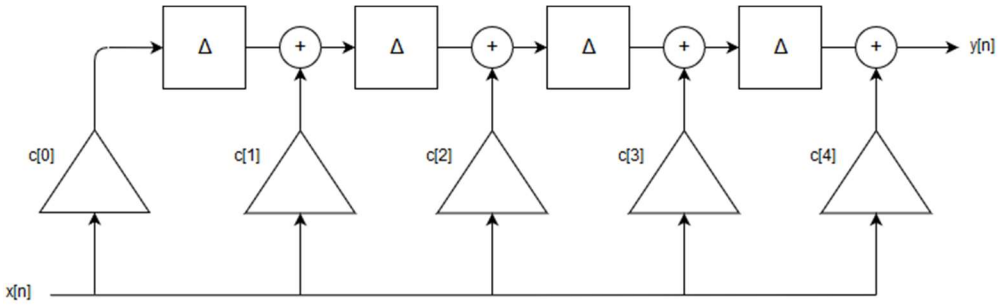


Fig.3 Esempio di architettura in forma diretta a cascata.

1.2.3 Filtro in forma a cascata

Un filtro a cascata di ordine N = 6 è mostrato in fig. 4.

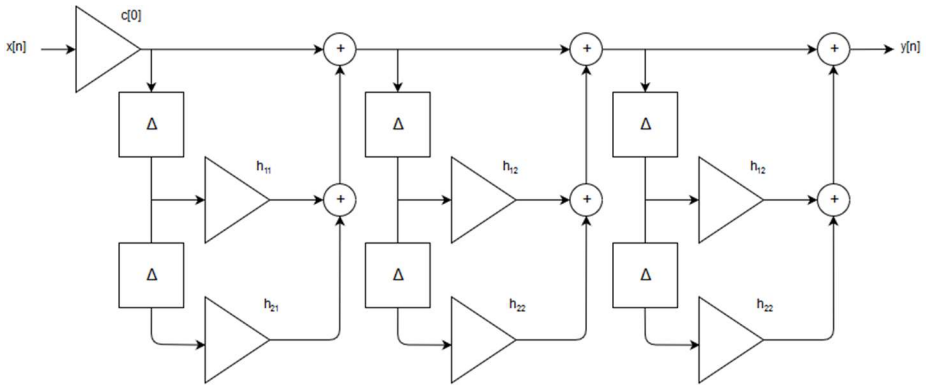


Fig.4 Esempio di architettura in forma a cascata.

Ogni ramo del filtro può essere visto come un filtro di ordine N=2 che può essere anche realizzato in forma diretta a cascata.

1.2.4 Filtro in fase lineare

La proprietà di simmetria di un filtro a fase lineare può essere utilizzata per ridurre il numero di moltiplicatori di quasi la metà del filtro in forma diretta. Si consideri il filtro di ordine N = 6 di figura 5.

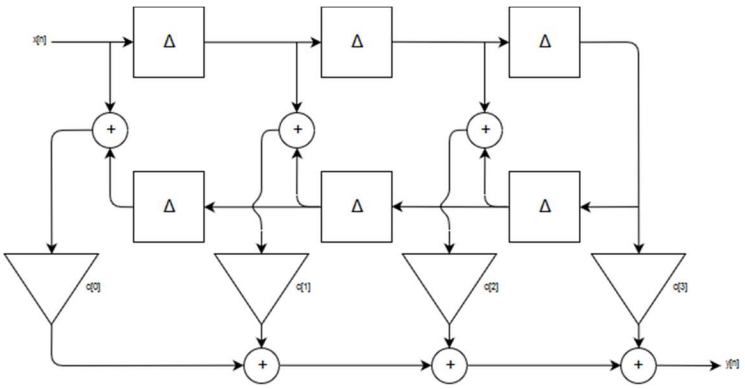


Fig.5 Esempio di architettura in fase lineare.

2. Progettazione

L'implementazione di un filtro FIR richiede tre componenti principali:

1. Moltiplicatore
2. Sommatore
3. Elemento di ritardo

Le tre componenti del filtro sono mostrate in fig. 6.

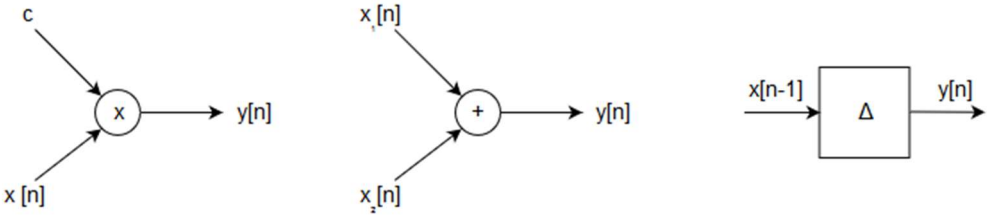


Fig.6 Componenti principali di un filtro FIR.

L'uscita del moltiplicatore è $y[n] = c \cdot x[n]$.

L'uscita del sommatore è $y[n] = x_1[n] + x_2[n]$.

L'uscita dell'elemento di ritardo è $y[n] = x[n-1]$.

3. Architettura implementata

Il filtro è stato realizzato nella forma diretta utilizzando il software Modelsim.
Nei successivi paragrafi verranno mostrati lo schema totale e dei singoli componenti.

3.1 Architettura di alto livello

Il diagramma a blocchi dell'architettura da progettare è mostrato in figura 7.

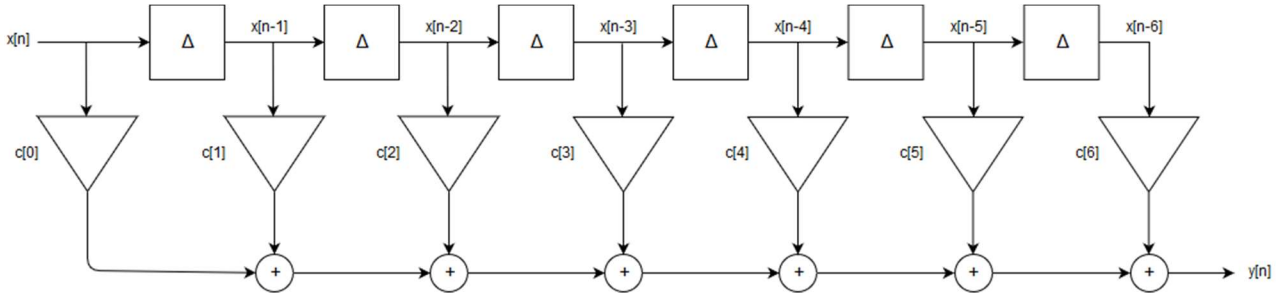


Fig.7 Architettura dello schema diretto.

Come si vede in figura il filtro è di ordine N=6 con sei elementi di ritardo, sette moltiplicatori e sei sommatori. Questa è stata riprogettata in sette moduli tap e un adder come in fig. 8. Questi moduli saranno descritti in §3.2 e §3.3.

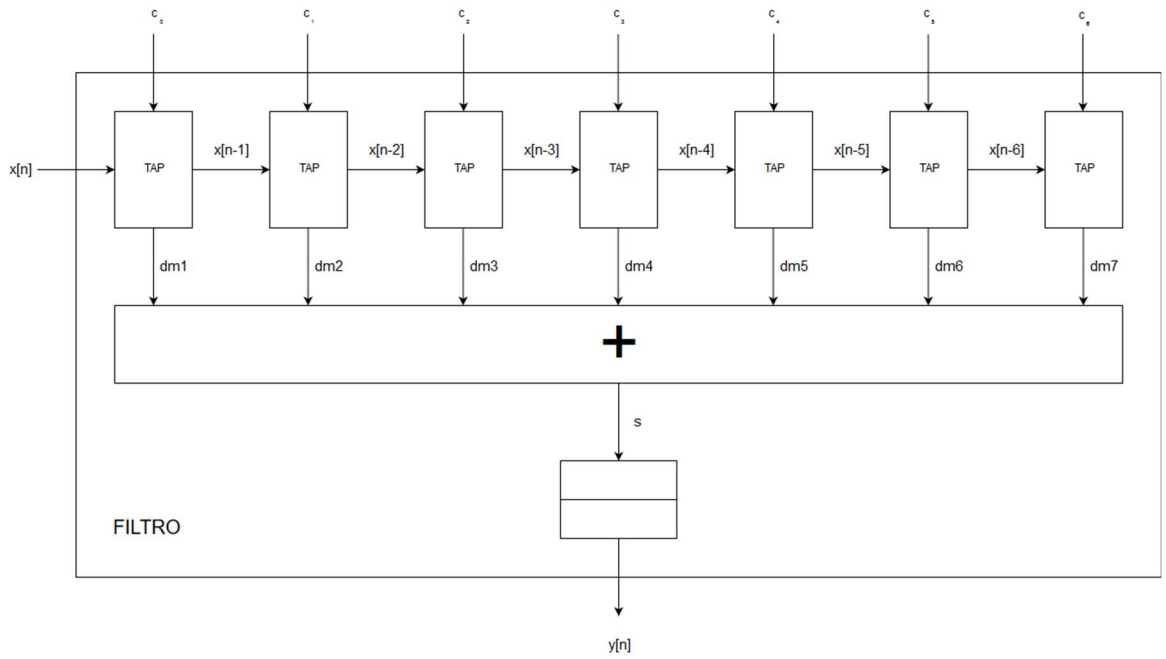


Fig.8 Organizzazione dell'architettura implementata.

3.1.1 Tap

All'interno del tap è possibile individuare due componenti base: un registro ed un moltiplicatore. L'uscita del moltiplicatore sarà un ingresso dell'adder, mentre l'uscita del registro sarà l'ingresso della successiva tap.

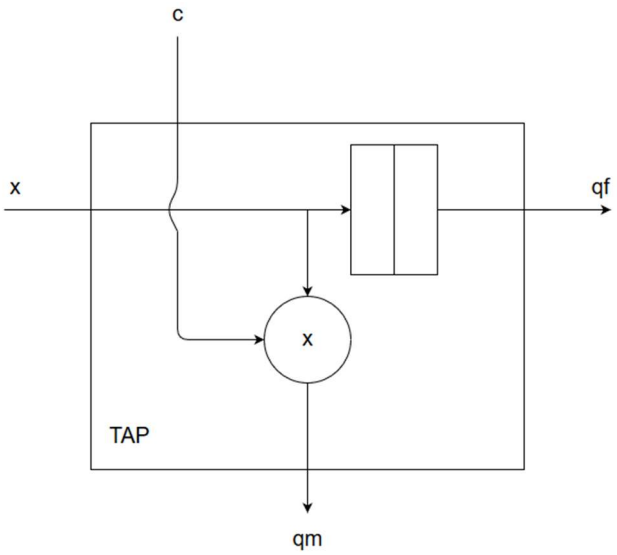


Fig.9 Schema ad alto livello del componente Tap.

```
1  entity tap is
2          generic(N: natural := 16);
3          port(
4                  x: in std_logic_vector(N-1 downto 0); -- ingresso x[n-i]
5                  c: in std_logic_vector(N-1 downto 0); -- coefficiente i-esimo del filtro
6                  qf: out std_logic_vector(N-1 downto 0);
7                  qm: out std_logic_vector(N-1 downto 0);
8                  clk: in std_logic;
9                  rst: in std_logic;
10                 en: in std_logic
11         );
12 end tap;
```

3.1.2 Adder

L'adder è il componente del filtro che genera l'output. Prende in ingresso sette prodotti in modulo e segno, li converte in complemento a due e ne esegue la somma.

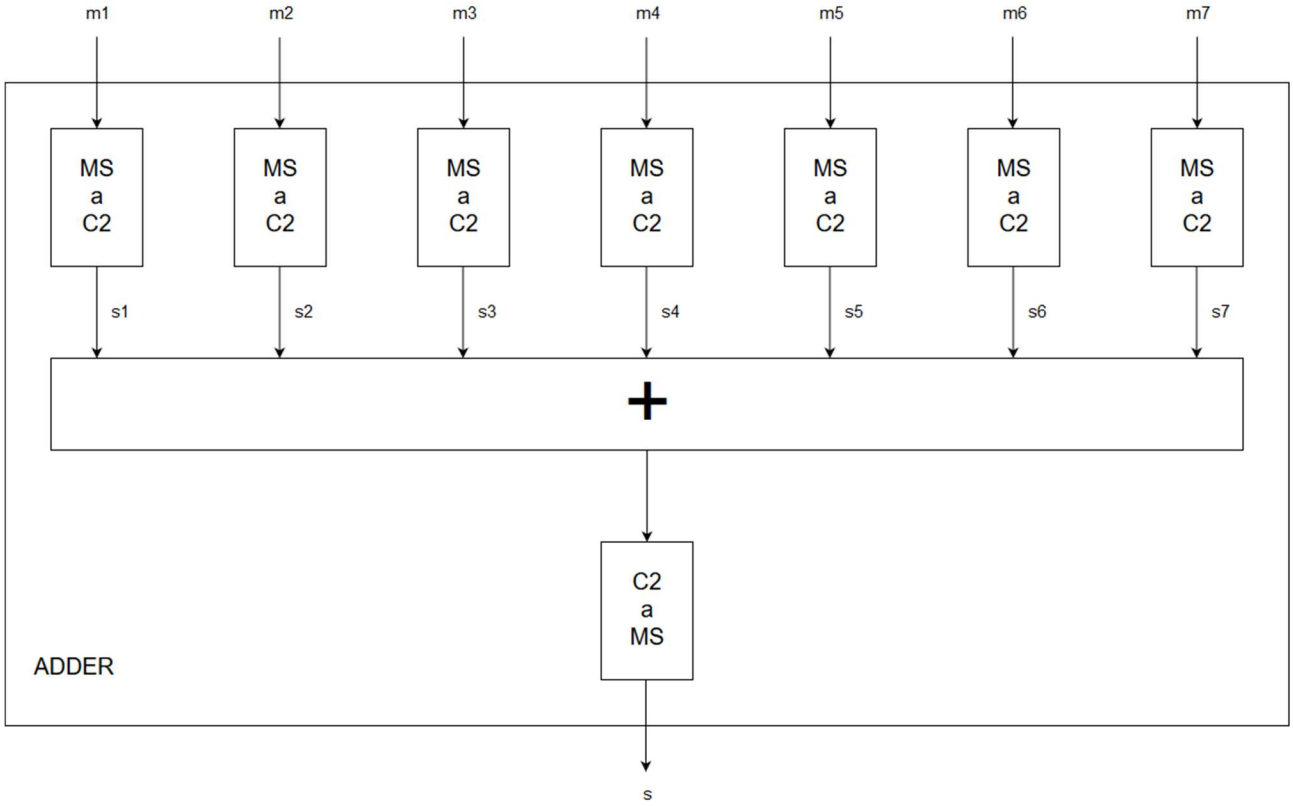


Fig.10 Schema ad alto livello del componente Tap.

```
1  entity adder is
2          generic(N: integer := 16);
3          port(
4                  m1, m2, m3, m4, m5, m6, m7: in std_logic_vector(N-1 downto 0);
5                  s: out std_logic_vector(N-1 downto 0)
6          );
7  end adder;
```

4. Test plan e Test bench

4.1 Test plan

Per pianificare la fase di analisi e verifica del dispositivo si è scelto innanzitutto la codifica dell'informazione. Con il vincolo di progetto che i dati in ingresso e in uscita dal componente fossero su 16 bit, è stata considerata una codifica modulo e segno in virgola fissa, con il bit più significativo b_{15} per il segno e i restanti $b_{14}..b_0$ per rappresentare la parte frazionaria dell'informazione, rappresentando valori nell'insieme D:

$$-1+\frac{1}{2^{N-1}} \leq a \leq 1-\frac{1}{2^{N-1}} \tag{2}$$

Usando il software MATLAB si è calcolato il valore del LSB come:

$$LSB = \frac{1}{2^{N-1}} = 3.051 \cdot 10^{-5} \tag{3}$$

Con LSB è stato possibile quantizzare i 63 campioni di test generati con una funzione sinusoidale, i coefficienti del filtro e calcolare l'errore di quantizzazione su ogni valore.

```
1  % Calcolo dei valori xq in ingresso al sistema
2  alfa = 0:0.1:2*pi;
3
4  x = sin(alfa);
5
6  xi = round(x/LSB); xq = xi * LSB;
7
8  ex = abs(xq - x); exm = mean(ex);     % Errore di quantizzazione
9  exassoluto = var(ex);
10 exrelativo = exassoluto/exm;
11 expercentuale = exrelativo * 100;
12 extot = sum(ex);
13
14 % Coefficienti c del filtro quantizzati
15 c = [0.0135,0.0785,0.2409,0.3344,0.2409,0.0785,0.0135]; % Coefficienti assegnati
16
17 ci = round(c/LSB); cq = ci * LSB; % Coefficienti quantizzati
18
19 ec = abs(cq - c); ecm = mean(ec);     % Errore di quantizzazione
20 ecassoluto = var(ec);
21 ecrelativo = ecassoluto/ecm;
22 ecpercentuale = ecrelativo * 100;
23 ectot = sum(ec);
```

L'errore di quantizzazione totale sui campioni di test è $4.1163 \cdot 10^{-4}$, con un errore medio di $6.5337 \cdot 10^{-6} \pm 1.9180 \cdot 10^{-11}$, mentre per i coefficienti $6.3184 \cdot 10^{-5}$, con un errore medio di $9.0262 \cdot 10^{-6} \pm 6.3129 \cdot 10^{-12}$.

A questo punto non resta che calcolare i valori di convoluzione ideali e quantizzati:

```
1  y = conv(x, c); % Uscita ideale
2
3  yq = conv(xq, cq); % Uscita quantizzata
4
5  ey = abs(yq – y); eym = mean(ey);     % Errore di quantizzazione
6  eyassoluto = var(ey);
7  eyrelativo = ecassoluto/eym;
8  eypercentuale = eyrelativo * 100;
9  eytot = sum(ey);
```

L'errore totale dei campioni atteso dal filtro è $6.6657 \cdot 10^{-4}$, con un errore medio di $9.6604 \cdot 10^{-6} \pm 3.8588 \cdot 10^{-11}$.

4.2 Test bench

Per validare il dispositivo è stato realizzato un test bench su Modelsim. Il test bench consiste di cinque prove con differenti subset di campioni di test e l'esito di ciascun test è stato confrontato con il risultato della convoluzione quantizzata generata da MATLAB. Il confronto è evidenziato in una tabella sotto ad ogni prova. Le prime tre colonne descrivono l'output del filtro in codifiche differenti, l'ultima il prodotto della convoluzione calcolato su MATLAB.

Prima prova In ingresso al filtro è stato trasmesso un solo campione $x_q = 0.1$.

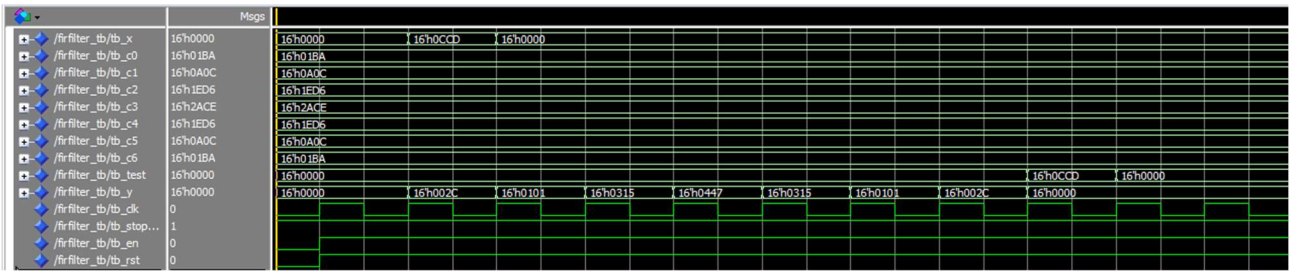


Fig. 10 Prima prova di Modelsim. Osservando il segnale tb_x è possibile vedere gli ingressi del filtro ad ogni ciclo di clock, invece guardando il segnale tb_y si nota la convoluzione prodotta dal filtro.

HEX	BIN	DEC	MATLAB
0000	0000000000000000	0	0
002C	0000000000101100	0.0013	0.0013
0101	0000000100000001	0.0078	0.0078
0315	0000001100010101	0.0240	0.0240
0447	0000010001000111	0.0334	0.0334
0315	0000001100010101	0.0240	0.0240
0101	0000000100000001	0.0078	0.0078
002C	0000000000101100	0.0013	0.0013

Seconda prova In ingresso al filtro sono trasmessi i primi 7 campioni x_q generati dalla funzione sinusoidale.

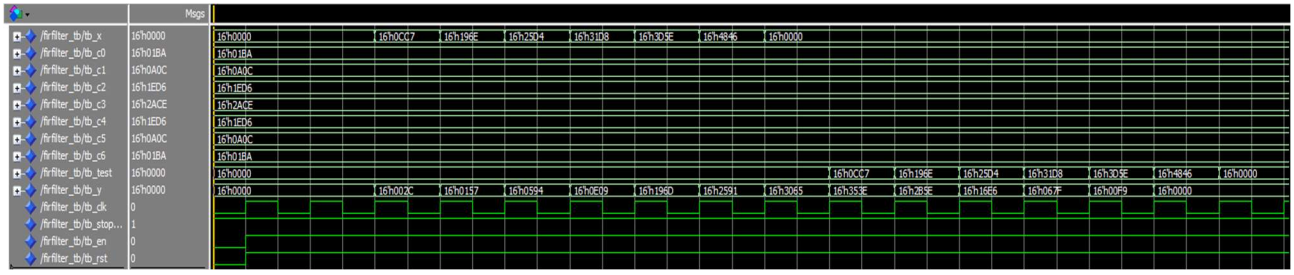


Fig. 11 Seconda prova di Modelsim.

HEX	BIN	DEC	MATLAB
0000	0000000000000000	0	0
002C	0000000000101100	0.0013	0.0013
0157	0000000101010111	0.0104	0.0105
0594	0000010110010100	0.0436	0.0436
0E09	0000111000001001	0.1096	0.1096
196D	0001100101101101	0.1986	0.1987
2591	0010010110010001	0.2935	0.2935
3065	0011000001100101	0.3781	0.3781
353E	0011010100111110	0.4160	0.4160
2B5E	0010101101011110	0.3388	0.3388
16E6	0001011011100110	0.1789	0.1789
067F	0000011001111111	0.0507	0.0507
00F9	0000000011111001	0.0076	0.0076

Terza prova In ingresso al filtro vengono trasmessi i 7 campioni x_q in prossimità del picco della funzione seno.

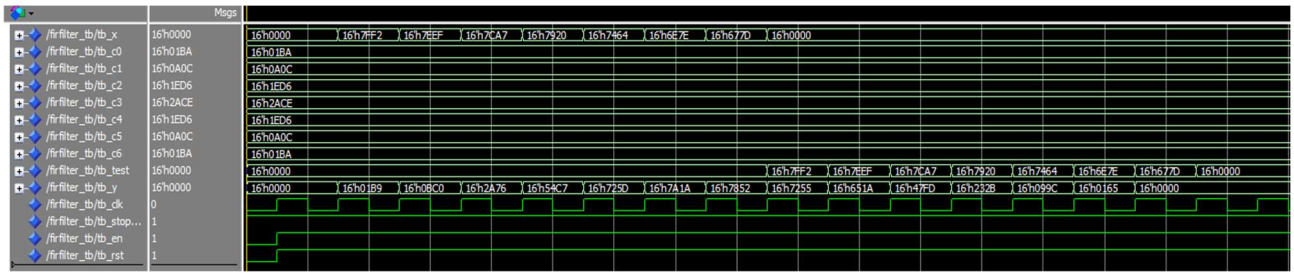


Fig. 12 Terza prova di Modelsim.

HEX	BIN	DEC	MATLAB
0000	0000000000000000	0	0
01B9	0000000110111001	0.0134	0.0134
0BC0	0000101111000000	0.0918	0.0918
2A76	0010101001110110	0.3317	0.3318
54C7	0101010011000111	0.6623	0.6624
725D	0111001001011101	0.8934	0.8936
7A1A	0111101000011010	0.9539	0.9540
7852	0111100001010010	0.9400	0.9401
7255	0111001001010101	0.8932	0.8933
651A	0110010100011010	0.7899	0.7899
47FD	0100011111111101	0.5624	0.5625
232B	0010001100101011	0.2747	0.2748
099C	0000100110011100	0.0751	0.0751
0165	0000000101100101	0.0109	0.0109

Quarta prova Vengono trasmessi in ingresso i primi 7 campioni x_q negativi generati dalla funzione seno.

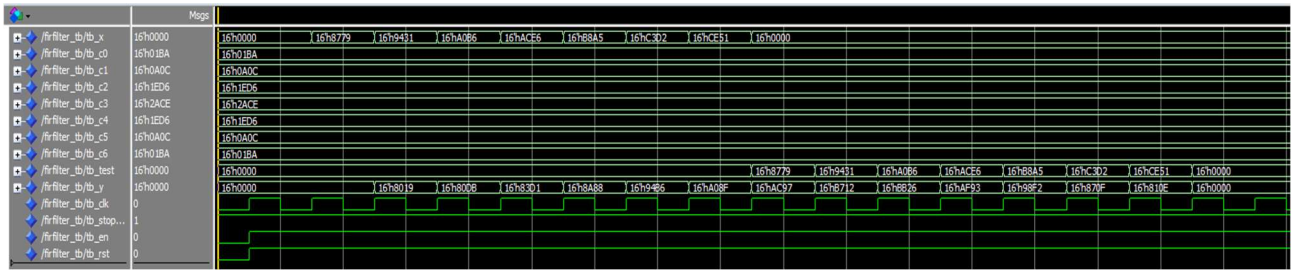


Fig. 13 Quarta prova di Modelsim.

HEX	BIN	DEC	MATLAB
0000	0000000000000000	0	0
8019	1000000000011001	$-7.3242 \cdot 10^{-4}$	$-7.8748 \cdot 10^{-4}$
80DB	1000000011011011	-0.0067	-0.0067
83D1	1000001111010001	-0.0298	-0.0299
8A88	1000101010001000	-0.0823	-0.0823
94B6	1001010010110110	-0.1618	-0.1619
A08F	1010000010001111	-0.2543	-0.2544
AC97	1010110010010111	-0.3483	-0.3485
B712	1011011100010010	-0.4302	-0.4303
BB26	1011101100100110	-0.4621	-0.4622
AF93	1010111110010011	-0.3716	-0.3717
98F2	1001100011110010	-0.1949	-0.1950
870F	1000011100001111	-0.0551	-0.0552
810E	1000000100001110	-0.0082	-0.0083

Quinta prova Infine vengono trasmessi gli ultimi tre campioni positivi e i primi quattro negativi del test set.

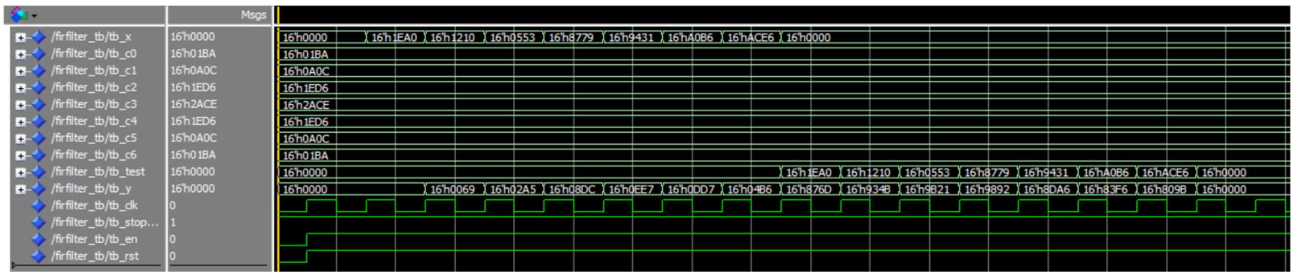


Fig. 14 Quinta prova di Modelsim.

HEX	BIN	DEC	MATLAB
0000	0000000000000000	0	0
0069	0000000001101001	0.0032	-0.0032
02A5	0000001010100101	0.0206	0.0207
08DC	0000100011011100	0.0692	0.0693
0EE7	0000111011100111	0.1164	0.1165
0DD7	0000110111010111	0.1081	0.1081
04B6	0000010010110110	0.0368	0.0368
876D	1000011101101101	-0.0580	-0.0580
934B	1001001101001011	-0.1507	-0.1507
9B21	1001101100100001	-0.2119	-0.2120
9892	1001100010010010	-0.1920	-0.1920
8DA6	1000110110100110	-0.1066	-0.1067
83F6	1000001111110110	-0.0309	-0.0310
809B	1000000010011011	-0.0047	-0.0047

5. Sintesi logica

I file di descrizione hardware usati su Modelsim sono stati importati su VIVADO dove è stata fatta la sintesi del circuito.

5.1 RTL

Lo schema generato in fig. 15 coincide con il design di fig. 8. L'importazione è stata completata senza warning.

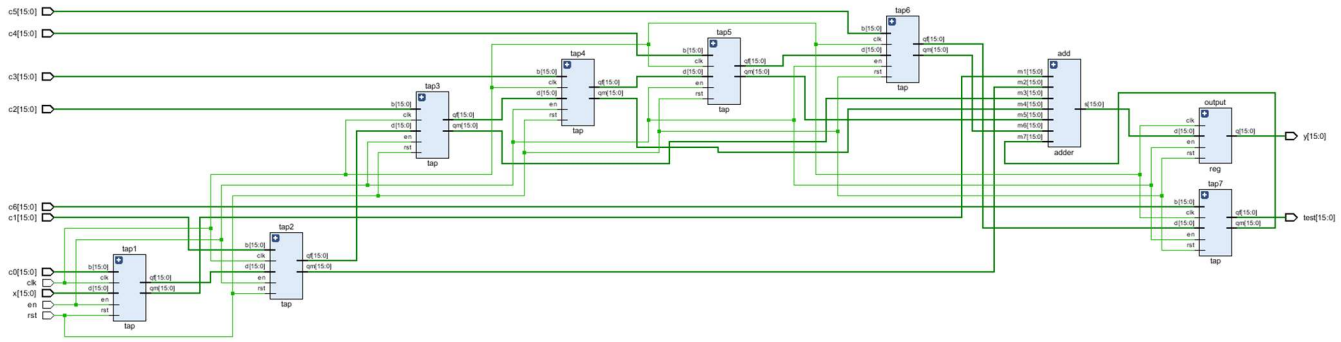


Fig. 15 Schema RTL del filtro.

È interessante osservare come sono descritti i componenti adder e MSaC2 di §3.3.

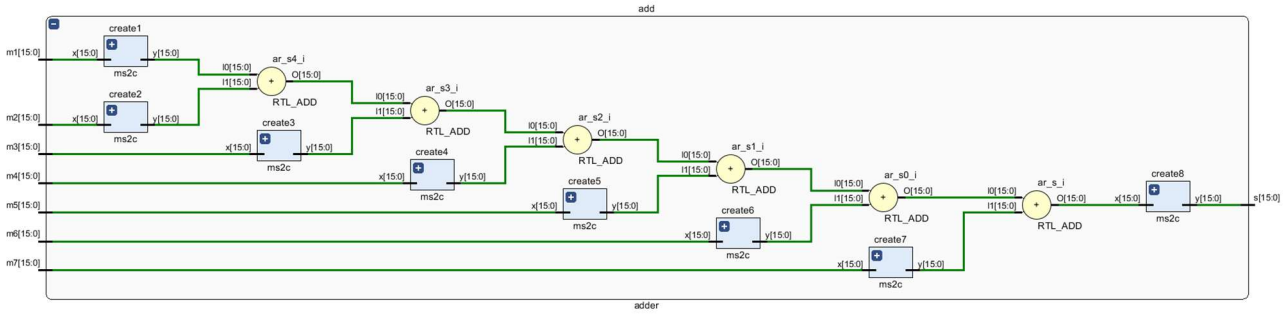


Fig. 16 Schema RTL del componente adder.

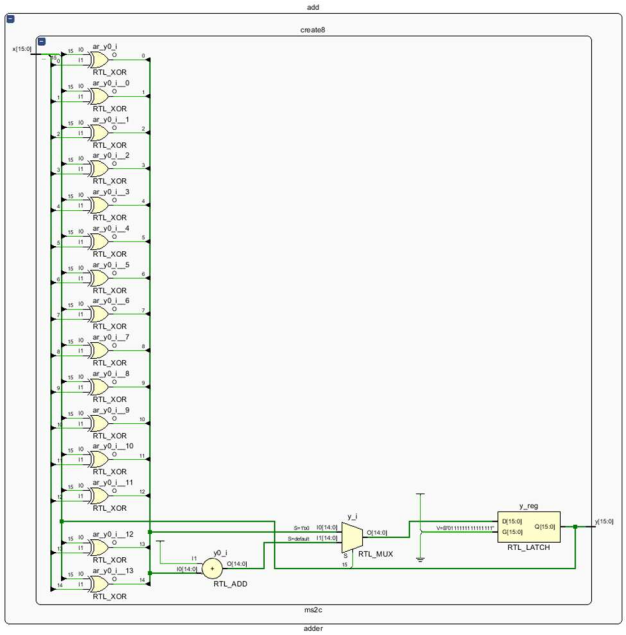


Fig. 17 Schema RTL del componente MSaC2.

5.2 Sintesi

Per studiare il comportamento del circuito viene collegato il filo di $clock$ del simulatore al cavo clk del filtro. La sintesi è stata avviata con un periodo di clock $T_{clock} = 20\ ns$. L'esito della sintesi termina senza warning importanti.

Temporizzazione Nella scheda report timing i vincoli risultano incontrati.

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	6,103 ns	Worst Hold Slack (WHS):	0,158 ns	Worst Pulse Width Slack (WPWS):	9,500 ns
Total Negative Slack (TNS):	0,000 ns	Total Hold Slack (THS):	0,000 ns	Total Pulse Width Negative Slack (TPWS):	0,000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	112	Total Number of Endpoints:	112	Total Number of Endpoints:	129

All user specified timing constraints are met.

Fig. 18 Timing report di VIVADO.

Osservando il valore dello slack (WNS) si può individuare il percorso critico del filtro, che va dal registro di output al registro del componente tap, come mostrato nelle figure 19 e 20.

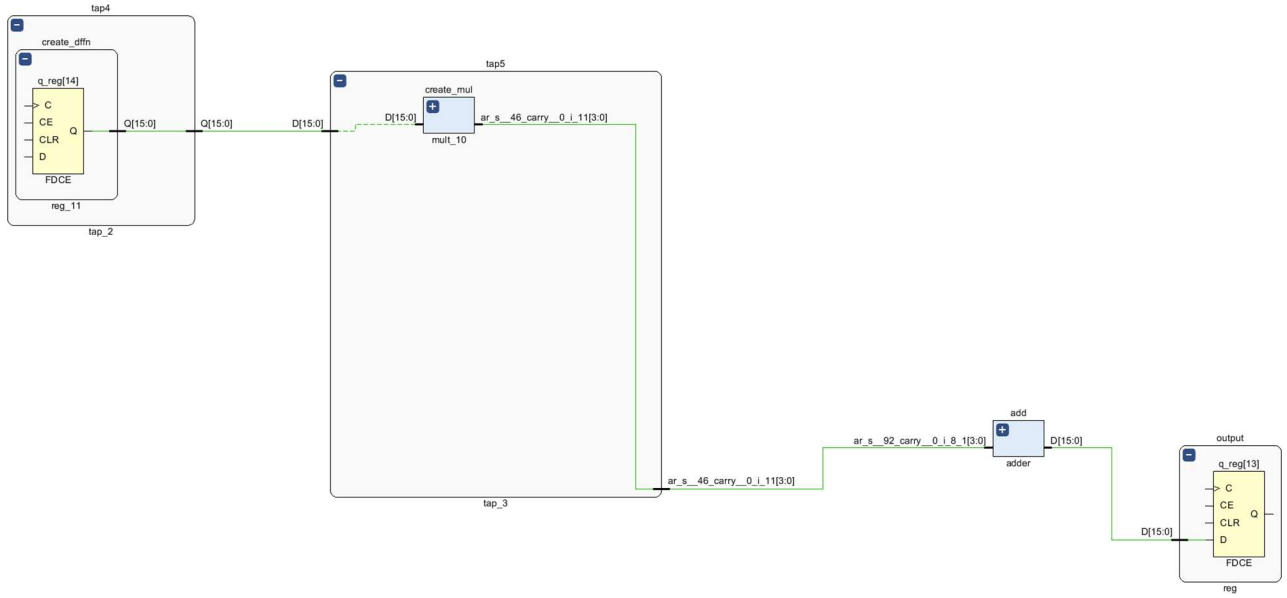


Fig. 19 Percorso critico del filtro.

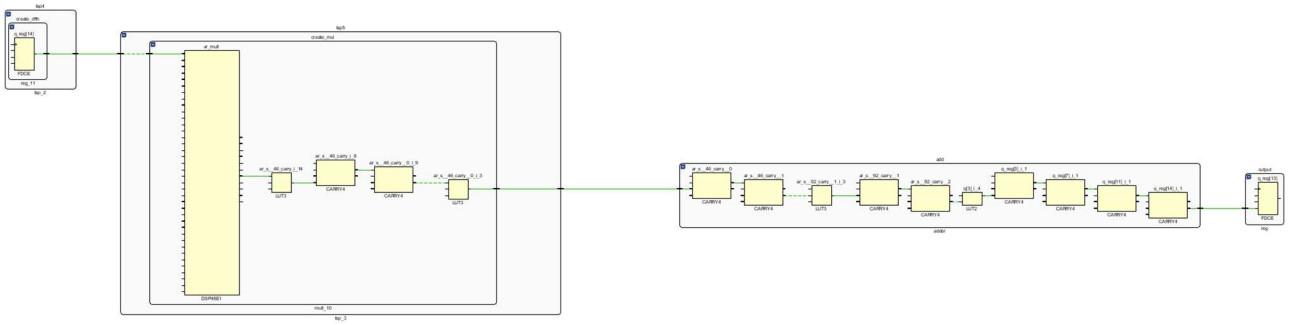


Fig. 20 Percorso critico del filtro.

Tenendo in considerazione il valore slack del timing report è possibile prevedere quanto più veloce può andare il circuito, infatti sapendo dalla teoria che:

$$T_{clock} \geq T_{setup} + T_{p-logic} + T_{c-q} + T_{slack} \tag{4}$$

ed essendo la somma $T_{setup} + T_{p-logic} + T_{c-q} = K$ fissata dall'architettura, è possibile calcolare la frequenza massima di clock. Riscrivendo la (4) come equazione:

$$T_{clock} = K + T_{slack} \tag{5}$$

E scrivendo il periodo minimo di clock come:

$$T_{clockmin} = K + T_{slackmin} \tag{6}$$

Raccogliendo le due equazioni per K:

$$T_{clockmin} - T_{slackmin} = T_{clock} - T_{slack} \tag{7}$$

E poiché $T_{slackmin} = 0$, ne deriva sostitudendo con i valori di fig. 18:

$$T_{clockmin} = T_{clock} - T_{slack} = 20 - 6.103 = 13.897\ ns \tag{8}$$

Da cui si ottiene la massima frequenza di clock:

$$f_{clockmax} = \frac{1}{T_{clockmin}} = 71.958\ Mhz \tag{9}$$

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	0,000 ns	Worst Hold Slack (WHS):	0,158 ns	Worst Pulse Width Slack (WPWS):	6,448 ns
Total Negative Slack (TNS):	0,000 ns	Total Hold Slack (THS):	0,000 ns	Total Pulse Width Negative Slack (TPWS):	0,000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	112	Total Number of Endpoints:	112	Total Number of Endpoints:	129

All user specified timing constraints are met.

Fig. 21 Report con massima frequenza di clock.

Risorse Nella scheda report utilization sono visibili le risorse necessarie per la realizzazione del filtro. Bisogna notare che servono più pin di I/O di quelli presenti sul dispositivo Zynq 7000, quindi l'implementazione su questo dispositivo non è possibile.

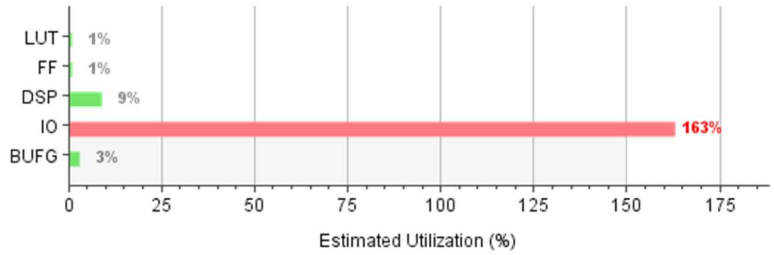
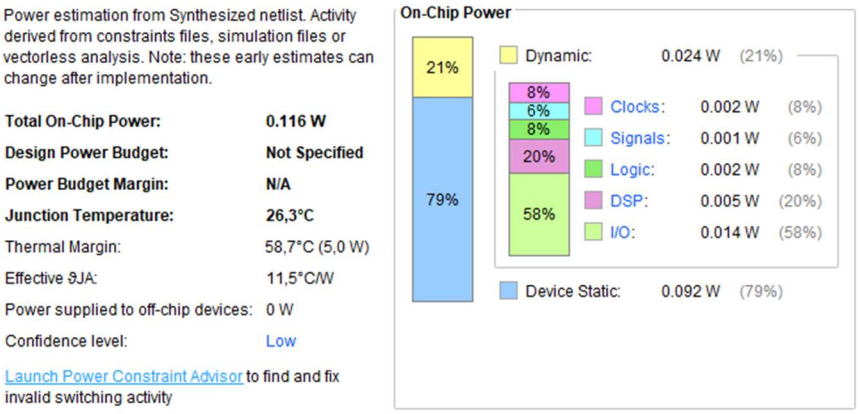


Fig. 22 Report con massima frequenza di clock.

Resource	Estimation	Available	Utilization %
LUT	216	17600	1.23
FF	128	35200	0.36
DSP	7	80	8.75
IO	163	100	163.00
BUFG	1	32	3.13

Fig. 23 Report con massima frequenza di clock.

Potenza Nella scheda report power è possibile ottenere una stima della potenza dissipata pari a $0.116\ W$, dove il 79% è potenza statica.



6. Conclusioni

Al termine di questo lavoro è stato progettato e sintetizzato un filtro FIR in forma diretta con un errore di quantizzazione medio ideale dell'ordine di 10^{-6}.
Purtroppo non è stato possibile implementare questo filtro sul dispositivo Zynq 7000 perché i pin di I/O sul dispositivo non sono sufficienti, tuttavia è possibile una descrizione alternativa dove in ingresso al filtro si hanno solo 16 pin per tutti i coefficienti c_i e non $16 \cdot 7$. Per realizzarlo si rivelerà essenziale implementare una procedura che permetta il caricamento dei coefficienti in 7 registri del filtro e successivamente avviare la convoluzione.