Una recensione di "FPGA Implementation of Digital Filters"

Alessandro Restelli e Daniele Paolo Scarpazza
{arestell, scarpazz}@elet.polimi.it

Politecnico di Milano
Dipartimento di Elettronica e Informazione
Dottorato in Ingegneria dell'Informazione
XVIII Ciclo - Anno Accademico 2004-2005

Lunedì 26 Aprile 2004

Sommario

Questo documento riassume e commenta il contenuto dell'articolo "FPGA Implementation of Digital Filters", ed è organizzato come segue: la sezione 1 indica le generalità dell'articolo; la sezione 2 ne riassume il contenuto con particolare riguardo alle soluzioni presentate per la realizzazione delle unità MAC, per la realizzazione di filtri FIR e IIR, e per l'introduzione del pipelining nelle architetture presentate; la sezione 3 presenta una valutazione complessiva dell'articolo in termini di originalità, tempestività, bontà delle previsioni e qualità.

1 Generalità

Titolo:	FPGA IMPLEMENTATION OF DIGITAL FILTERS
Autori:	Chi-Jui Chou, Satish Mohanakrishnan, Joseph B. Evans Telecommunications & Information Sciences Laboratory Department of Electrical & Computer Engineering, University of Kansas, USA
Apparso in:	Proceedings of the 4th International Conference on Signal Processing Applications and Technology, Sept. 28 - Oct. 1, 1993, Santa Clara, CA, pp. 80-88

2 Riassunto

2.1 Introduzione

Gli algoritmi di filtraggio digitale sono solitamente implementati su processori DSP oppure, quando i requisiti impongono elevate frequenze di campionamento, su ASIC appositamente progettati. L'autore mostra come l'implementazione di questi filtri su FPGA permetta di raggiungere prestazioni significativamente superiori a quelle basate su DSP tradizionali e, per volumi di produzione moderati, costi molto più bassi, essendo del tutto assenti i costi NRE (non-recurring engineering costs) che sono molto forti nel caso ASIC (anche decine di milioni di dollari per un ASIC nelle ultime tecnologie disponibili al momento in cui stiamo scrivendo).

Vi sono poi dei vantaggi secondari, dovuti al fatto che le FPGA possono essere riprogrammate sul posto, rendendo così possibile il miglioramento delle funzionalità offerte dal prodotto e l'adattamento a nuove condizioni operative o nuovi standard approvati nel frattempo, allungando sostanzialmente la vita utile del prodotto, cosa impossibile per DSP e ASIC.

Chiaramente vi sono anche alcuni svantaggi, limitazioni e difficoltà insite nell'adozione di una soluzione basata su FPGA:

1. la densità raggiunta dai dispositivi disponibili al momento in cui l'autore dell'articolo scriveva era appena sufficiente per realizzare moduli di modesta complessità;

2. l'architettura impone un numero significativo di vincoli sulle funzioni logiche realizzabili all'interno di ogni blocco logico e sui ritardi dell'instradamento dei segnali all'interno dell'array.

Come l'autore aveva correttamente previsto, il problema indicato al punto 1, a più di dieci anni di distanza, è stato fortemente ridimensionato dall'evoluzione tecnologica: la densità raggiunta oggi dalle FPGA è parecchio più elevata di quella di 10 anni fa. Per essere più precisi, nel lavoro esposto nell'articolo, l'autore fa uso di dispositivi della famiglia XC4000 prodotti da Xilinx: oggi il prodotto di fascia più bassa appartenente a quella famiglia (nominatamente l' XC4013XLA) ha la stessa quantità di CLB (24x24: 576) del dispositivo che nel 1993 si trovava al vertice della stessa famiglia [3]. Per di più i prodotti della famiglia XC4000 vengono ufficialmente denominati da Xilinx prodotti maturi [1], un eufemismo per indicare la strada dell'obsolescenza. Siamo infatti ben lontani dai prodotti al vertice della gamma, come le FPGA Virtex II-Pro, che contengono più di 44000 blocchi logici e fino 4 core di processori PowerPC 405 immersi nel fabric FPGA [4].

2.2 Realizzazione delle unità "moltiplica e accumula"

Gli autori sviluppano separatamente i singoli elementi del filtro digitale, per poi unirli in diverse configurazioni. Come viene più volte sottolineato, una delle caratteristiche più vantaggiose delle FPGA è la possibilità di confrontare le prestazioni di diverse architetture senza alcun costo di produzione e senza tempi di attesa.

L'elemento base di un filtro digitale è l'unità MAC (*multiply and accumulation unit*), un blocco funzionale in grado di eseguire il prodotto tra il dato di ingresso e il coefficiente del filtro digitale, e di sommare poi il risultato della moltiplicazione in un registro di accumulazione. Per l'implementazione della MAC gli autori sviluppano separatamente un sommatore a 16 bit ed un moltiplicatore 8x8 bit.

L'implementazione del sommatore segue in pratica un iter obbligato dal costruttore, in quanto Xilinx ha introdotto nei CLB delle lookup-table dedicate al calcolo del riporto dei blocchi sommatori.

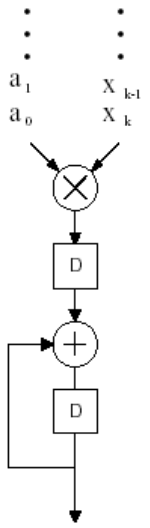

Figura 1: Struttura di una unità MAC.

Una FPGA non possiede la velocità che può avere un circuito integrato ASIC a pari tecnologia, paga infatti la completa riconfigurabilità con la presenza di un numero elevato di risorse di interconnessione, ovvero alcune net che attraversano tutta l'FPGA con i relativi driver, ed altre risorse di interconnessione locali. Poiché ogni net deve fisicamente raggiungere un numero elevato di blocchi, il suo fanout e le sue capacità parassite introducono nelle FPGA ritardi superiori rispetto a quelle di un chip ASIC. Questo però non toglie al costruttore la possibilità di realizzare particolari linee interne dedicate con driver rinforzati e capacità parassite più basse per assolvere ad alcune funzioni, come la distribuzione del clock o, appunto il calcolo e la propagazione del carry nelle addizioni.

Per quanto riguarda il moltiplicatore, l'autore sceglie di implementare una soluzione classica, di tipo ripple-carry ad array, piuttosto dispendiosa dal punto di vista dell'area, ma dalle prestazioni molto elevate. L'autore sembra però trascurare il fatto che è possibile sfruttare i fast carry logic block e realizzare il prodotto come una somma di termini.

La struttura utilizzata per il MAC completo è estremamente compatta: il sommatore a 16 bit usato come accumulatore processa solo gli 8 bit più significativi all'uscita del moltiplicatore. Per velocizzare il blocco, il sommatore ed il moltiplicatore sono stati separati da una serie di flip-flop ottenendo una configurazione a pipeline. I flip-flop introdotti non accrescono il numero di CLB necessari per il design perché sono comunque presenti nei CLB e rimarrebbero altrimenti inutilizzati. È il moltiplicatore a determinare il massimo ritardo di elaborazione (quasi 100 ns) e ad imporre una frequenza di lavoro di 10 MHz. Questa non è una limitazione significativa in termini assoluti, vista la possibilità di impiegare il pipelining anche all'interno della struttura del moltiplicatore.

A questo punto è possibile, combinando più unità MAC, realizzare filtri FIR ed IIR. Riportiamo qui di seguito in modo sintetico le configurazioni di filtri adottate dagli autori, elencandone le caratteristiche salienti.

2.3 Filtri FIR

Sono proposte due configurazioni per realizzare filtri FIR. La prima è la forma canonica invertita (figura 2), in cui sono presenti blocchi di somma e moltiplicazione in cascata attraverso l'uso di pipeline. Nella forma canonica invertita l'ultimo dato che entra nel filtro viene moltiplicato in un solo ciclo di clock per tutti i coefficienti (taps) del filtro. I risultati di questi prodotti sono il contributo all'uscita del filtro che l'ingresso nell'istante n dovrà avere negli istanti $n-1, n-2, n-3$, etc... dunque solo il termine relativo all'istante n può confluire all'uscita sommato alla somma dei termini precedenti. Per ogni tap $n-x$ del filtro è presente un registro che memorizza la somma dei termini fino a $n-x$ e la cui uscita confluisce in un sommatore che aggiunge alla somma il termine $n-x+1$, che a sua volta confluisce in un successivo registro posto in cascata. L'architettura richiede un numero di sommatori, registri e moltiplicatori (dunque di MAC) pari al numero di taps del filtro, dunque è dispendiosa in termine di area, ma ha il vantaggio di avere il throughput più elevato, essendo esso imposto dal tempo di elaborazione dell'elemento più lento (in genere il moltiplicatore) e non dal numero di taps.

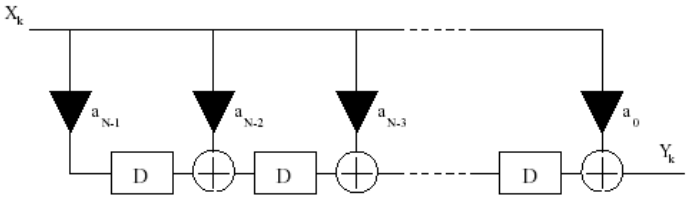

Figura 2: Forma canonica invertita di filtri FIR

Per snellire la struttura si possono però sostituire ai moltiplicatori dei semplici shift register a patto di utilizzare come pesi del filtro soltanto potenze di due. Questa prima architettura di filtro mette immediatamente in luce il vantaggio che le FPGA hanno sui DSP. I DSP, nonostante la loro struttura adatta all'elaborazione dei segnali, rimangono comunque macchine di tipo sequenziale in cui le unità aritmetiche sono riutilizzate in modo ciclico per eseguire operazioni complesse. Nelle FPGA è possibile rendere più rapidi i tempi di elaborazione aumentando l'occupazione di risorse hardware. Sebbene in linea di principio questo sia possibile in misura arbitraria, non lo è nella pratica, poiché le risorse sono limitate. Anche nelle FPGA si deve poter trovare un'architettura che permetta un compromesso tra tempo di elaborazione e occupazione di area. Gli autori propongono la struttura[1] che rappresentiamo qui nelle figure 3 e 4.

La struttura è ricavata quadruplicando l'implementazione di una classica unità MAC, che

[1]Obiezione: a parere nostro non è chiaro perché sia necessaria la seconda fila di ritardi (indicata in figura dal rettangolo a bordi tratteggiati), infatti il ritardo complessivo dei sommatori a monte e a valle di tali registri è comunque inferiore al ritardo del moltiplicatore (22.5 ns per il sommatore contro 100 ns per il moltiplicatore). Si avrebbe una latenza minore eliminandoli, a meno che non si voglia introdurre pipelining anche nel moltiplicatore.

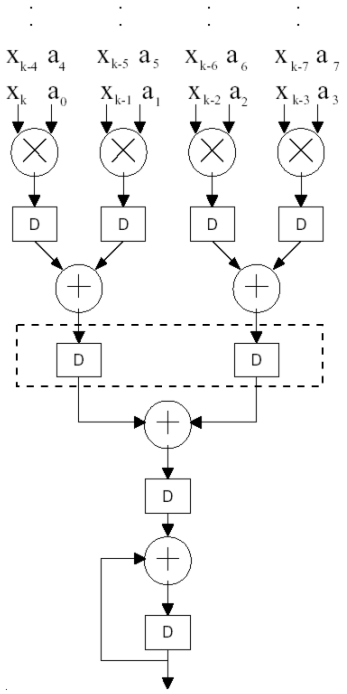

Figura 3: Struttura proposta per implementare un filtro FIR in FPGA.

normalmente si ha all'interno dei DSP. Questo rende il calcolo dell'uscita del filtro 4 volte più rapida.

In generale se il numero di moltiplicazioni in parallelo per ogni ciclo di clock è M e il filtro FIR è a N taps, per un periodo di clock pari a T secondi la frequenza di elaborazione, dunque di campionamento dei dati in ingresso, è

$$f = \frac{1}{T\lceil N/M \rceil}. \tag{1}$$

2.4 Filtri IIR

È noto che i filtri di tipo IIR permettono, grazie alla retroazione dell'uscita, di realizzare funzioni di trasferimento con poli e zeri, che, a parità di prestazioni in termini di risposta in frequenza, utilizzano un numero inferiore di coefficienti (taps). Questo riduce notevolmente le risorse necessarie, sia in termini di occupazione di area, che in termini di tempo di elaborazione. Così gli autori propongono un filtro completo con architettura ARMA (un filtro autoregressivo seguito in cascata da un filtro a media mobile) la cui struttura è illustrata in figura 5.

Nella figura è evidenziato con una linea tratteggiata il punto che dovrebbe rappresentare il collo di bottiglia del filtro, essendo l'unico blocco di elaborazione che contiene due sommatori ed un moltiplicatore al posto che un sommatore ed un moltiplicatore. Tuttavia ancora una volta la

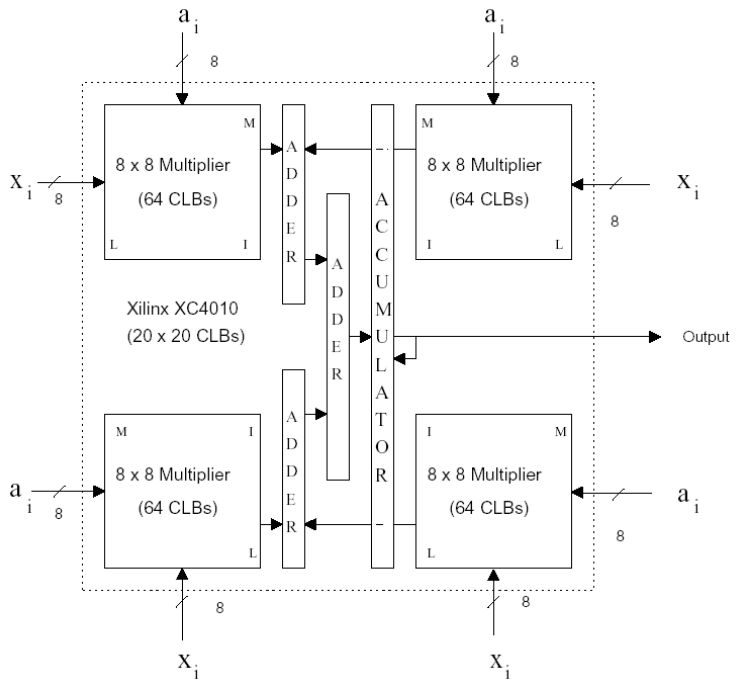

Figura 4: Implementazione del filtro FIR in FPGA Xilinx XC4010.

presenza delle *fast carry logic* permette di realizzare un sommatore a due stadi con una latenza di appena 23 ns, che è molto vicina a quella di un singolo sommatore. Il ritardo introdotto dalla presenza del secondo sommatore può così essere inferiore ai ritardi di interconnessione cosicché non si può nemmeno localizzare il percorso critico nel blocco suddetto.

In figura 6 è possibile vedere l'implementazione di un filtro del secondo ordine a parametri variabili in una FPGA Xilinx XC4013, con la seguente funzione di trasferimento:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-1}}{2^S - a_1 z^{-1} - a_2 z^{-1}}$$

Notiamo al denominatore la presenza del termine 2^S. Allo schema ideale di figura 5 è stato infatti aggiunto un blocco shift register comandato da un bus a 3 bit. Lo shift register serve a dividere per una potenza di 2 il dato che deve ricircolare nel blocco AR del filtro. Questo accorgimento permette di usare la notazione in virgola fissa garantendo che i poli del filtro abbiano modulo minore di uno, condizione di stabilità asintotica per il filtro. Oltre ai coefficienti del filtro a_n e b_n a 8 bit, è quindi necessario fornire all'FPGA l'esponente S del divisore a 3 bit. Partendo dalla struttura IIR appena vista, che supporta una velocità di campionamento di ben 7 MHz, gli autori implementano sulla stessa FPGA un filtro del quarto ordine mettendo in cascata due filtri del secondo ordine. In questo caso i coefficienti sono fissi. Questo accorgimento semplifica notevolmente l'hardware e riduce l'occupazione di CLB perché ciascuno dei 10 moltiplicatori della struttura è realizzato con semplici shifter e sommatori. La nuova velocità massima di campionamento è di 10 MHz.

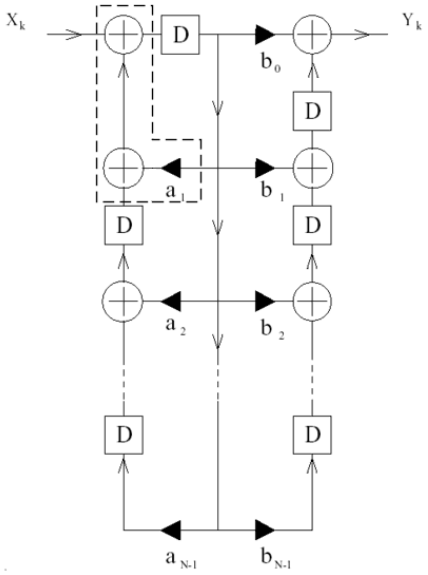

Figura 5: Struttura del filtro IIR ARMA proposto.

2.5 Tecniche di Pipelining

La trattazione si conclude prospettando la realizzazione dei filtri FIR partendo da un'unità MAC dotata di pipeline. Frammentare l'esecuzione di un'operazione di prodotto e accumulazione ha senso soltanto se è possibile introdurre nella struttura un dato da elaborare per ogni colpo di clock, cosa che può avvenire soltanto con i filtri FIR. Gli autori, avendo precedentemente introdotto un filtro FIR con quattro unità MAC, ma modesto uso di pipelining (figure 3 e 4), mostrano come sia possibile realizzare un'unica unità MAC in grado, grazie all'uso di pipeline, di funzionare con un clock 8 volte superiore. Si possono confrontare le due situazioni utilizzando la formula 1.

Nell'architettura di figura 3 vengono forniti al filtro 4 coefficienti per ogni colpo di clock, e per un filtro a N taps la frequenza di sampling è

$$f_s = \frac{1}{100ns\lceil \frac{N}{4} \rceil} = \left\lfloor \frac{40}{N} \right\rfloor MHz \ . \tag{2}$$

Con una sola unità MAC dotata di pipeline il periodo del clock passa da 100 ns a meno di 12.5 ns. Il filtro accetta un solo coefficiente per ogni colpo di clock, dunque per ottenere un dato in uscita è necessario attendere $N + 1$ cicli. La frequenza di sampling è perciò

$$f_s = \frac{1}{12.5ns(N+1)} = \frac{80}{N+1} MHz. \tag{3}$$

In pratica si ha un miglioramento delle prestazioni approssimativamente di un fattore due. Gli autori sottolineano però come sia più complesso il floorplanning con l'aumento degli stadi di pipeline. La frammentazione della struttura dei moltiplicatori riduce notevolmente i ritardi dovuti alla

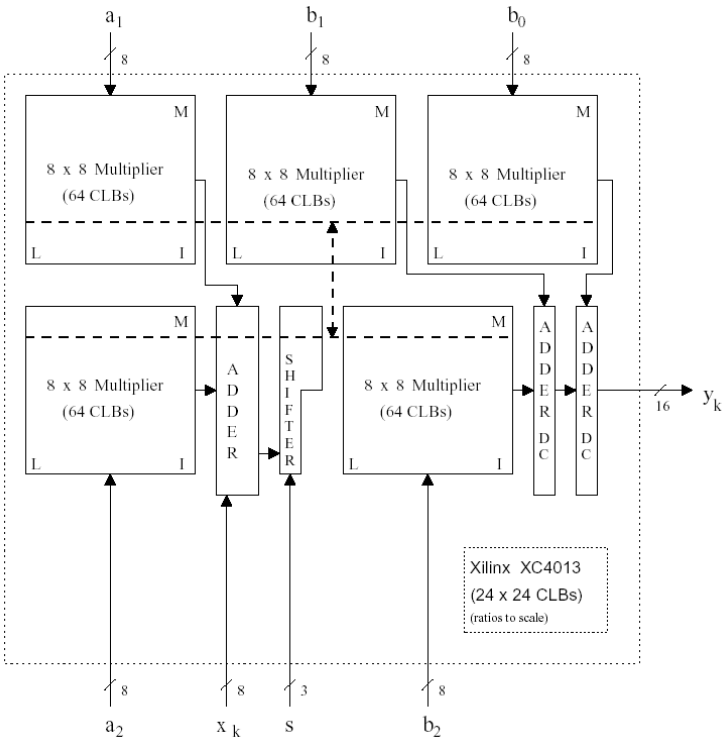

Figura 6: Implementazione di un filtro del secondo ordine su di una FPGA Xilinx XC4013.

logica combinatoria, tanto che possono diventare dominanti i ritardi dovuti alle interconnessioni (routing delays). Questi ultimi dipendono notevolmente dalla disposizione dei blocchi funzionali nell' FPGA.

3 Valutazione dell'articolo

3.1 Originalità e tempestività

Per poter fornire una valutazione dell'articolo che sia obiettiva e priva di anacronismi ed errori di prospettiva, una giusta contestualizzazione storica è necessaria. Per questo motivo è necessario valutare l'articolo nel quadro delle pubblicazioni scientifiche e tecniche contemporanee e successive a quella considerata.

A nostro parere l'articolo in questione manifesta doti notevoli di originalità, infatti intendiamo dimostrare che il lavoro apre strade a cui nemmeno Xilinx, il produttore dei dispositivi usati nel lavoro, aveva pensato dettagliatamente. Per fare ciò citiamo documentazione ufficiale di Xilinx (articoli, guide e *application note*). Nel 1992, anno in cui è il lavoro descritto è presumibilmente stato svolto, Xilinx non aveva ancora reso disponibile alcuna application note che evidenziasse l'attitudine delle proprie FPGA verso applicazioni DSP. Solo nel 1993, anno di pubblicazione del-

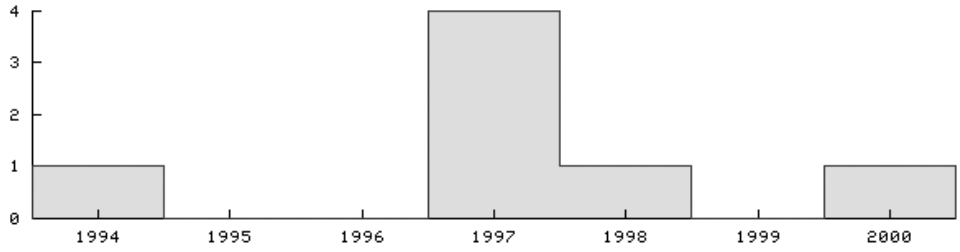

Figura 7: Distribuzione negli anni delle citazioni attive [2]

l'articolo in questione, Xilinx rende disponibile una prima application note [5] che spiega come implementare su FPGA semplici filtri FIR a coefficienti costanti. È necessario attendere la fine del 1994 per vedere una semplice application note che descriva un filtro FIR a 8 bit e 16 tap (ancora una volta a coefficienti costanti), sviluppato proprio su dispositivi della serie XC4000, con prestazioni di 5 MSample/s circa [6], e il 1995 per vedere a riguardo un paio di articoli significativi e di respiro sufficientemente ampio: il primo [7] mostra abbastanza genericamente in quali contesti e condizioni sia vantaggioso l'uso di FPGA rispetto a DSP; il secondo [8] espone invece dettagliatamente due casi di studio in cui le FPGA sono state usate per realizzare un decodificatore Viterbi e un filtro FIR. Le architetture implementate sono descritte con precisione, e i vantaggi (sia prestazionali che economici) sono analizzati in una ampia varietà di condizioni possibili.

È superfluo aggiungere che dal 1995 a oggi il filone di ricerca si è consolidato e ha proseguito il proprio sviluppo con notevoli risultati: il trattamento digitale dei segnali su FPGA è diventata una soluzione di riferimento anche in ambito video [8,9]. Il consolidamento di questo tipo di soluzione è così elevato che nel 2002 Xilinx rende disponibile un prodotto software denominato System Generator for DSP (oggi giunto alla versione 6.1), che permette di completare tutte le fasi di progettazione, simulazione, implementazione e verifica di un sistema DSP basato su FPGA all'interno di un unico ambiente di lavoro di tipo MATLAB/Simulink.

Per quanto riguarda la letteratura scientifica, una analisi dettagliata degli articoli indicizzati rivela che non vi sono pubblicazioni che propongano esplicitamente e nel dettaglio l'uso di FPGA in ambito DSP antecedenti o contemporanee a quella considerata. Ve ne è una pressoché simultanea [10], ad opera dello stesso autore, ed un certo numero, ad opera dello stesso gruppo di lavoro, che completano il filone di ricerca negli anni immediatamente successivi [11,12].

3.2 Bontà delle previsioni

Nell'articolo in questione, gli autori esprimono una collezione di previsioni, riguardanti i seguenti temi:

- col passare del tempo l'evoluzione dei processi di fabbricazione permette di ottenere densità sempre maggiori e ritardi sempre minori;
- la disponibilità in futuro di FPGA dotate di maggiore densità rende i vincoli di progetto, relativi per esempio alla ampiezza della parola dati, sempre meno stringenti.

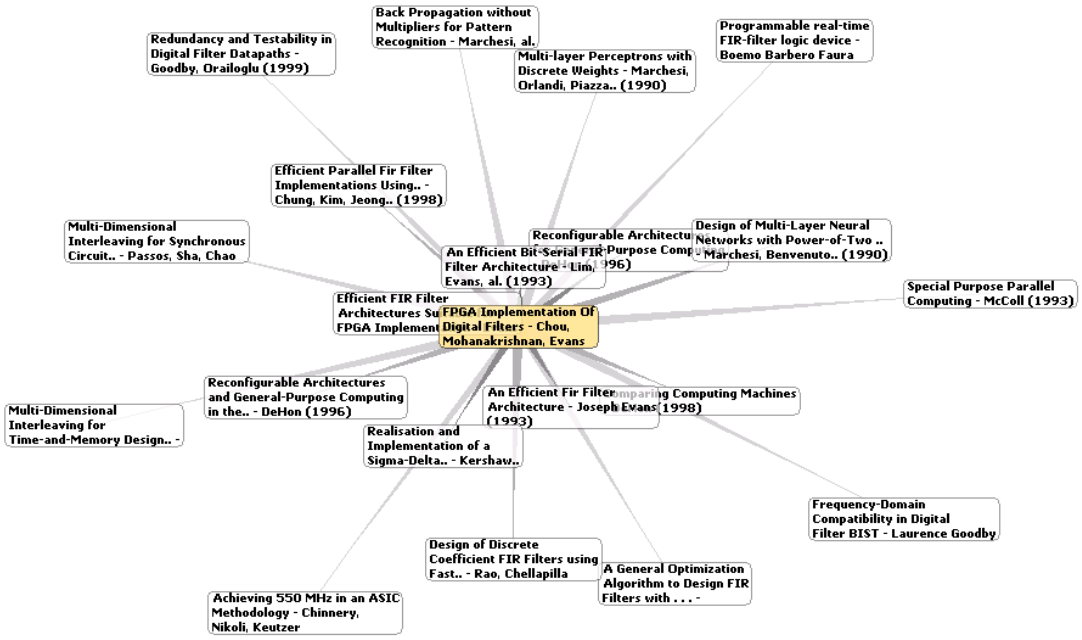

Figura 8: Mappa delle citazioni attive e passive [2]

Come già dettagliato nella prima sezione di questo documento, è evidente che entrambe le previsioni sono state ampiamente verificate.

Considerando che per la crescita delle densità dei dispositivi le operazioni manuali nel flusso di progettazione diventano sempre meno attuabili, e i tool di EDA sempre più necessari e determinanti per la fattibilità di un progetto, gli autori potevano forse sbilanciarsi nel fare qualche previsione anche a riguardo dell'evoluzione del software per la progettazione, che pure ha subito un fortissimo sviluppo nei dieci anni che separano il paper dai nostri giorni.

3.3 Conclusioni

Riteniamo che l'articolo sia di interesse per una porzione rilevante delle comunità di ricerca sull'elaborazione digitale dei segnali, sulle architetture dei calcolatori, sui sistemi embedded e sugli strumenti EDA. Il paper ha stimolato un ragionevole numero lavori successivi, in parte ad opera dello stesso gruppo di lavoro, in parte di altri, come testimoniato dalle almeno 9 citazioni in altri lavori di differenti gruppi di ricerca, distribuite dal 1994 fino ai nostri giorni (vedere fig. 7). Le idee presentate sono originali e creative, e tecnicamente motivate. Da un punto di vista formale, l'articolo è ben fatto: titolo e riassunto sintetizzano in maniera efficace il contenuto, la bibliografia è completa, il linguaggio impiegato è appropriato, e l'uso delle figure esplicativo, e sufficiente ma non sovrabbondante. L'organizzazione complessiva e l'ordine di presentazione degli argomenti sono ben strutturati.

Bibliografia

1. Sito web Xilinx `http://www.xilinx.com` ;
2. Citeseer - Scientific Literature Digital Library - sito web:
   `http://citeseer.ist.psu.edu/chou93fpga.html`;
3. Xilinx, *XC4000XLA/XV Field Programmable Gate Arrays - Product Specification*, publication DS015 (v1.3), October 18, 1999;
4. Xilinx, *Virtex-II Platform FPGAs: Introduction and Overview*, publication DS031-1 (v1.9) September 26, 2002;
5. Xilinx, *Constant Coefficient Multipliers for the XC4000E*, publication XAPP054 (v1.0) 1993;
6. Xilinx, *16-Tap, 8-Bit FIR Filter Applications Guide* (v1.01) November 21, 1994;
7. Steven K. Knapp, Xilinx, *Using Programmable Logic to Accelerate DSP Functions*, 1995;
8. Xilinx, *An Inverse Discrete Cosine Transform (IDCT) Implementation in Virtex for MPEG Video Applications*, XAPP208 (v1.1) December 29, 1999;
9. Xilinx, *Virtex-EM FIR Filter for Video Applications*, XAPP241 (v1.1) October 3, 2000;
10. Joseph B. Evans, *An efficient FIR filter architecture*, Proceedings of ISCAS 1993, pages 627-630;
11. Joseph B. Evans, *Efficient FIR Filter Architectures Suitable for FPGA Implementation*, IEEE Transactions on Circuits and Systems, 41(7):490–493, July 1994;
12. Yong Ching Lim, Joseph B. Evans, et al., *An Efficient Bit-Serial FIR Filter Architecture*, in Circuits, Systems and Signal Processing, May 1995;