Processament d'àudio mitjançant una FPGA

Xavi Martí Camacho

Resum — Les FPGA són àmpliament utilitzades en molts camps i aplicacions degut a la seva capacitat de còmput i a la flexibilitat que ofereixen en ser programades. En àudio, les FPGA ofereixen una gran capacitat de processament, conversió i compressió en quant al processament de senyals (DSP), gràcies al paralel·lisme inherent en la seva arquitectura. Així, doncs, en aquest projecte s'han realitzat un conjunt de proves basades en el processament de mostres d'àudio a través de filtres digitals implementats mitjançant MATLAB en un simulador HDL (ModelSim), per determinar de manera aproximativa la capacitat computacional d'una FPGA a l'hora de processar àudio. Finalment, s'han aplicat aquests resultats a un model concret de FPGA existent al mercat, tenint en compte el hardware i característiques pròpies d'aquesta, per a aproximar-nos al rendiment que obtindriem utilitzant una FPGA real per a processar àudio.

Paraules clau—FPGA, HDL, MATLAB, filtre digital, simulador HDL, ModelSim

Abstract—FPGAs are frecuently used in various applications due to their high computing capacity and their flexibility to be programmed. In audio, FPGAs offer high processing, conversion and compression cappabilities in digital signal processing (DSP) therms, thanks to the inherent parallelism of their architecture. Therefore, this project consisted in executing a set of tests based on processing audio samples through digital filters implemented using MATLAB in a HDL simulator (ModelSim), to determinate aproximately the computing capacities of a FPGA. Finally, these results have been applied to a concrete FPGA, taking into account its own hardware and features, to get an aproximate idea of the performance we could see using a real FPGA to process audio.

Index TermsVFPGA, HDL, MATLAB, digital filter, HDL simulator, ModelSim

1 Introducció

Les FPGA permeten un alt grau de paral·lelisme en el processament de senyals degut a la seva arquitectura. En aquest projecte s'estudia el seu ús en el processament d'àudio, intentant clarificar les possibilitats que ens brinden i el potencial d'aquestes a l'hora de processar àudio a través de filtres que exigeixen certa capacitat de còmput.

El projecte s'estructura en múltiples seccions, que s'expliquen breument a continuació. En la secció d'objectius, es determinen els objectius i fites a assolir en aquest projecte. En la secció d'estat de l'art, s'expliquen els mecanismes originals de processament d'àudio, alguns mecanismes actuals i l'aparició de les FPGA en aquest camp. A l punt "4.Metodologia" s'explica la metodologia aplicada, mentre que al punt "5. Fases i implementació" es descriu detalladament els pasos seguits durant el desenvolupament i es defineixen certs conceptes necessaris. En la següent secció s'esmenten els resultats i els mètodes d'execució dels casos de prova, mentre que en la secció 7 s'extrapolen aquest resultats de manera teòrica a un model de FPGA disponible al mercat. Per últim, es parla de les línies d'investigació que han quedat obertes en aquest projecte, per a finalitzar amb la conclusió en l'últim apartat del treball.

2 Objectius

L'objectiu principal d'aquest projecte es basa en investigar i determinar de manera aproximativa la capacitat de processament d'àudio d'una FPGA a través de filtres implementats en HDL, és a dir, quina capacitat té una FPGA en realitzar una de les tasques que es realitza de manera freqüent mitjançant DSPs i algorismes. Per a la realització d'aquest objectiu, s'hauran d'assolir les següents fites:

- Implementar diversos filtres digitals amb els quals es realitzaran les diverses proves. Aquests filtres es poden agrupar de la següent manera: múltiples filtres passa-banda individuals, i un filtre d'un terç d'octava. Aquest filtre d'un terç d'octava constarà de 30 bandes (filtres passa-banda).
- Simular el comportament d'una FPGA mitjançant un simulador HDL, és a dir, un software de disseny de FPGAs i simulació de blocs. Aquest software, com ara pot ser ModelSim o Cadence Incisive, ens permetrà

- *E-mail de contacte: xavier.martic@e-campus.uab.cat*
- *Mención realitzada: Enginyeria de Computadors*
- *Treball tutoritzat per: Màrius Montón Macián*
- *Curs 2014/15*

simular els resultats que s'obtindrien en una FPGA física.
- Integrar aquests filtres, en llenguatge HDL, al simulador HDL, per tal de poder-hi processar la senyal d'àudio digital d'alta qualitat: 192KHz com a freqüència de mostreig, i 24 bits de resolució. Aquesta senyal d'àudio serà finita i, per tant, estarà formada per un nombre determinat de mostres d'àudio a processar.
- Processar la senyal d'àudio digital prèviament mencionada, a una freqüència de rellotge determinada, i obtenir-ne els resultats: el temps necessari per a processar el conjunt de mostres de la senyal, cicles de rellotge necesaris per a processar cada mostra, etc.

3 Estat de l'art

Antigament, el processament d'àudio es realitzava mitjançant circuits analògics, és a dir, amb components hardware amb els quals formaven circuits que filtraven les senyals analògiques d'entrada (modificant-ne el voltatge) per aconseguir una senyal de sortida desitjada. Això, però, implicava un alt cost degut als components del circuit, més espai necessari per implementar el circuit i més probabilitats de falla degut als múltiples components utilitzats.

Actualment, però, amb l'aparició de processadors suficientment potents, s'ha pogut replicar la funcionalitat dels circuits analògics mitjançant el processament digital de les senyals d'entrada (prèviament, però, convertides a senyals digitals). El processament digital d'àudio, té clars aventatges sobre els processament analògic: menor cost, ja que es redueix l'ús de components, un tamany més compacte, una major fiabilitat ja que un processador sol tenir una tassa de fallada molt baixa, i major integració amb altres sistemes digitals (p.ex amb els computadors).

Actualment, es pot processar àudio de manera digital en pràcticament qualsevol processador o microprocessador, tot i que n'hi ha d'especialitzats en realitzar aquesta tasca, com ara els DSP o "Processadors de senyals digitals", implementant-ne els filtres mitjançant algoritmes software.

Tot i disposar d'aquests processadors especialitzats, s'estan començant a utilitzar lesa FPGA per a processar tant àudio com imatges, ja que permeten un alt grau de paral·lelització dels càlculs al disposar de múltiples canals (i DSPs) i per tant disposant d'una capacitat de processament superior. En les FPGA, però, al tractar-se d'un conjunt de cèl·lules hardware programmables, s'implementen els filtres com a dissenys hardware, en comptes d'utilitzar algoritmes software.

El ús de les FPGA com a dispositius per a processar àudio, però, no ha sorgit tant de la necessitat d'obtenir un millor rendiment per a realitzar les mateixes tasques que es venien realitzant amb els DSP i certes tasques amb processadors de propòsit general, com pel sorgiment de noves tecnologies. Aquestes noves tecnologies, com ara els arrays d'altaveus, els sistemes d'acústica variable amb un gran nombre de micròfons, o el augment del nombre de canals de les taules de so i mescles, exigeixen una major capacitat de processament d'àudio, la qual s'ha vist coberta satisfactòriament amb l'ús de FPGAs.

4 Metodologia

En aquest projecte s'utilitzarà una metodologia o model incremental, és a dir, una metodologia per la qual el projecte té quatre fases (anàlisi, disseny, codificació i prova) a realitzar amb cada una de les fites importants, i amb la qual es van realitzant entregues cada cert temps. Aquestes entregues es produeixen cada cop que es realitzen increments funcionals. Amb aquest model o metodologia es poden realitzar projectes amb poc personal, i permet anar realitzarnt successives entregues de valor al client (en aquest cas al tutor del projecte) el qual pot valorar els progressos. L'arquitectura de la metodologia es basa en el pipelining, on hi ha varies fases seqüencials en les quals l'entrada d'una fase es la sortida de la fase o fita prèvia. A més, aquesta metodologia permet reduïr-ne el temps de desenvolupament inicial ja que es pot implementar funcionalitats parcials, tenint també una gran versatilitat i totes les ventatges de la metodologia seqüencial o en cascada.

5 Fases i implementació

El projecte compta amb un conjunt de fases determinades, de les quals se n'explica a posteriori la seva implementació. Les pròpies fases són les següents: implementació dels filtres, descripció del filtre en llenguatge HDL, simulació de la FPGA mitjançant un simulador HDL, preparar les mostres d'àudio per a ser processades, implementar el codi HDL del filtre en el simulador HDL, i finalment execució de la simulació i obtenció de resultats. Posteriorment a totes aquestes fases, se n'expliquen els resultats obtinguts.

5.1 Implementació dels filtres

La implementació dels filtres s'ha realitzat principalment en MATLAB. Aquest és un entorn interactiu de computació numèrica i també un llenguatge d'alt nivell que permet implementar algorismes, operar amb matrius, funcions i dades, crear interfícies d'usuari i interactuar amb altres llenguatges (C, C++, Python, Fortran,i Java). També permet realitzar simulacions mitjançant mòduls complementaris i software extern, motiu pel qual s'utilitza aquest entorn en aquest projecte. Mitjançant llenguatge d'alt nivell, doncs, s'ha escrit en MATLAB un algorisme que genera filtre d'un terç d'octava (3 bandes per cada octava) creant de manera iterativa cada una de les 30 bandes que el conformen.

```
%1/3 OCTAVE FILTER
f.BandsPerOctave = 3;
f.FilterOrder = 2;
F0 = validfrequencies(f);
Nfc = length(F0);
for i=1:Nfc,
    f.F0 = F0(i);
    Hd3Oct(i) = design(f,'butter');
end
```

Figura 1: Fragment del codi per generar les bandes del filtre d'un terç d'octava

Aquest filtre, té una freqüència central Fc de 1000Hz, a partir de la qual es distribueixen la resta de bandes. La seva freqüència de mostreig Fs2 és de 48.000Hz. Aquests dos paràmetres estan relacionats amb les freqüències sobre les que el filtre actua, de manera que el seu valor es mantindrà com a una constant. L'ordre del filtre, però, és el nombre d'elements que s'utilitzen per implementar cada banda o filtre passa-banda, augmentant doncs el nombre de càlculs a realitzar. Per tant, serà un paràmetre variable que es modificarà (amb un valor d'entre 2, 4 i 8) en les diferents execucions de les simulacions. Podem observar la incidència que té l'ordre del filtre en les freqüències de tall del propi filtre, en les Figures 2, 3 i 4.

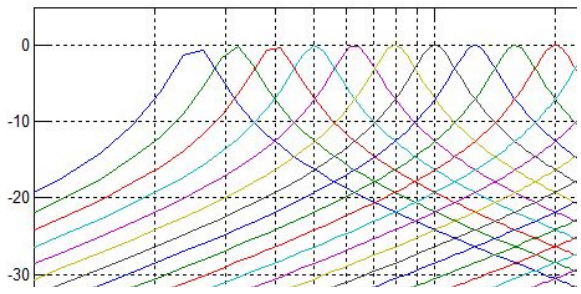

Figura 2: 10 primeres bandes del filtre d'un terç d'octava d'ordre 2 (fins a -30dB)

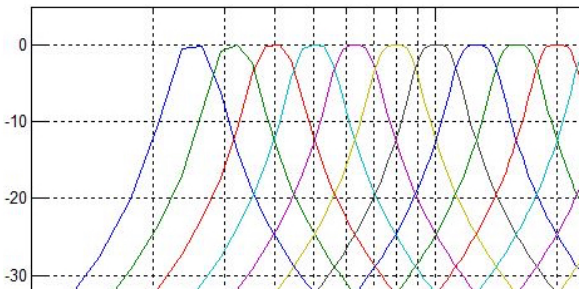

Figura 3: 10 primeres bandes del filtre d'un terç d'octava d'ordre 4 (fins a -30dB)

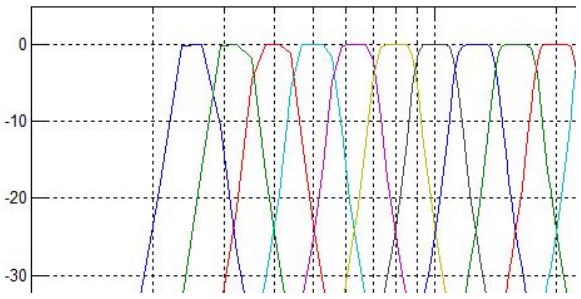

Figura 4: 10 primeres bandes del filtre d'un terç d'octava d'ordre 8 (fins a -30dB)

Posteriorment només cal especificar el tipus de disseny que s'utilitzarà. En el cas que ens ocupa, s'ha utilitzat un disseny Butterworth, ja que ens permet una resposta en freqüència plana (sense modificar) en la banda de pas, és a dir, no en modifica la senyal fins que arriba a les freqüències de tall. Es pot observar també a la Figura 1.

Finalment, per concloure les implementacions, cal definir una variable per cada banda definida prèviament, de manera que es pugui utilitzar aquesta variable per a ser importada a FDATool, com s'explica posteriorment. Les variables han rebut el nom d'Hd1 corresponent a la primera banda del filtre d'un terç d'octava, Hd2 corresponent a la segona banda, i successivament fins a completar-les totes amb Hd30.

5.2 Descripció dels filtres en llenguatge HDL

Un cop implementats els filtres en MATLAB, s'han hagut de descriure en llenguatge HDL per a poder ser compilats i executats pel simulador HDL. En aquest cas, MATLAB disposa d'una eina, FDAtool, que permet dissenyar varis tipus de filtres especificant-ne els seus paràmetres (ordre, freqüències de tall i central, tipus de resposta i tipus de disseny) o bé importar-ne de ja creats, i generar-ne la descripció en llenguatge hardware. No s'ha utilitzat, però, per a dissenyar el filtre d'un terç d'octava, ja que no permet especificar múltiples filtres passa banda dins d'un mateix filtre. Tampoc permet importar un filtre format per múltiples filtres, com en el cas que ens ocupa, de manera que s'han importat manualment cada una de les 30 bandes del filtre, i aquestes s'han disposat en cascada com a un únic filtre mitjançant la funcionalitat "Filter Manager" de FDATool.

Fet això, i mitjançant la pròpia eina FDATool, es pot generar tant la descripció HDL de cada una de les bandes o filtres passa-banda guardats com a variables prèviament, com la del filtre format per les 30 bandes en cascada, simulant el filtre d'un terç d'octava.

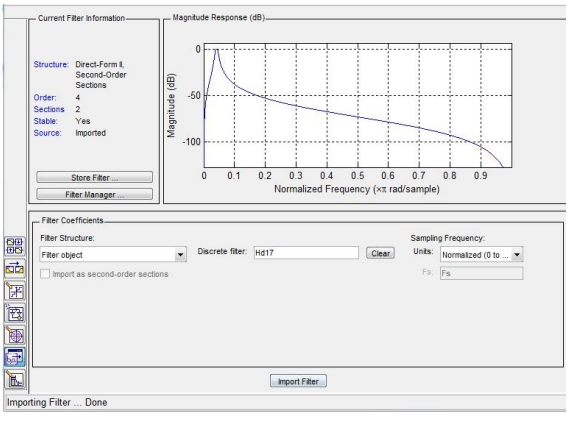

Figura 5: Importació d'un filtre passa-banda (variable Hd17 corresponent a la dissetena banda) a FDATool

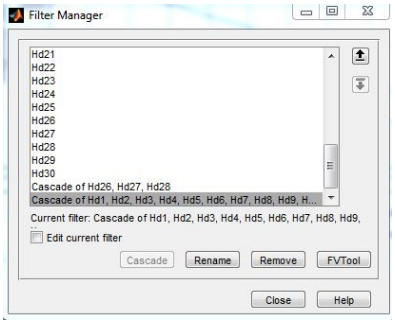

Figura 6: Concatenació de les 30 bandes en cascada

És important destacar que l'eina FDATool també ens informa de l'ordre total del filtre utilitzat així com del nombre d'etapes (filtres) de les que disposa. Com podem observar en la Figura 7, en el filtre d'un terç d'octava amb 30 bandes (filtres passa-banda) d'ordre 8 obtenim 30 seccions ("stages") i un ordre total de 240.

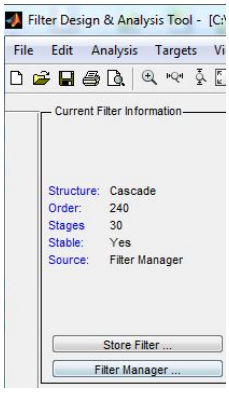

Figura 7: Informació (ordre total i etapes) del filtre en FDATool

5.3 Creació d'una senyal d'àudio discreta

La senyal d'àudio que processarem no serà una senyal continua, ja que no volem una execució contínua al llarg del temps, sinó que volem determinar el temps d'execució per a un cert nombre de mostres d'àudio. Així, la senyal estarà formada per un conjunt de mostres finit. El nombre de mostres d'àudio el determinarà la freqüència de mostreig, que és el nombre de mostres que es prenen d'una senyal d'àudio contínua (per exemple, una senyal analògica) per unitat de temps, i el tamany d'aquestes mostres el determinarà els bits de resolució. En el cas que ens ocupa, s'ha utilitzat un arxiu d'àudio de varis segons, a 24 bits i una freqüència de mostreig de 192KHz, és a dir, 192.000 mostres cada segon. S'ha retallat l'àudio a 0,5 segons, utilitzant un editor de mostres ("Ableton Live 9"), per a optimitzar els temps d'execució de les simulacions. En total, doncs, s'utilitzaran 96.000 mostres (corresponents a mig segon d'àudio) per a executar les proves.

Un cop es disposa de les mostres d'àudio, s'incorporaran com a variable del workspace de MATLAB, i posteriorment s'utilitzarà aquesta mateixa variable com a senyal d'entrada al simulador HDL, a l'hora d'executar les simulacions. Per a incorporar les mostres com a variable, s'utilitza la funció "audioread" amb l'arxiu d'origen, el nom de la variable a guardar (que hem anomenat "vivaldi") i una variable "Fs" on guardem la freqüència de mostreig. Fet això normalitzarem els valors de les mostres, és a dir, augmentarem l'amplitud de la ona augmentant-ne els valors que representen el volum en la que es reprodueix cada mostra. Cal destacar que aquest pas té un objectiu únicament funcional a l'hora d'interpretar els resultats, ja que augmentarà la visibilitat de la representació gràfica de les mostres quan es processin en el simulador HDL. Això també ens permetrà veure, de manera més diferenciada, la diferència entre la senyal d'entrada i la senyal de sortida o senyal filtrada, i per tant, la incidència que ha tingut el filtre.

```
%NORMALITZAR SENYAL ENTRADA
for j = 1:96000
    vivaldi(j,:) = vivaldi(j,:)*500000;
end
```

Figura 8: Normalització de les mostres a processar

5.4 Simulador HDL

Un simulador HDL és un software que permet simular blocs lògics i dissenyar FPGAs, així com també simular el comportament d'un component descrit en llenguatge HDL i analitzar-ne la seva resposta a partir d'estímuls (senyals d'entrada). Per tant, un cop ja implementats els filtres, s'utilitzarà per compilar i executar la descripció dels filtres per simular-ne el seu comportament, i així actuar sobre la senyal d'entrada prèviament definida. Aquest pas ha constat de la instal·lació de ModelSim PE 10.4c. Durant el procés, s'ha tingut en compte l'enllaçament a MATLAB del directori on hem instal·lat ModelSim, de manera que es pugui executar des del terminal de MATLAB amb la funció "vsim". En cas contrari, s'haurà de modificar la ruta amb la comanda "vsimdir".

La versió ModelSim PE compta amb llicència d'estudiant d'aquest Simulador HDL, fet que evita una inversió econòmica en llicències. L'elecció d'aquest software, a part del cost, es justifica mitjançant la compatibilitat i integració que té amb MATLAB, com posteriorment es podrà observar.

5.5 Simulació de blocs i compilació dels filtres en ModelSim

Tant per a compilar els filtres, descrits en HDL, en ModelSim, com per encaminar la senyal d'entrada (guardada en una variable en el workspace de MATLAB) per a ser processada durant la simulació, s'ha utilitzat Simulink com a intermediari mitjançant HDL Verifier. Simulink és un entorn de diagrames de bloc que permet modelar i simular sistemes dinàmics, incorporar algorismes de MATLAB, analitzar i exportar els resultats d'una simulació, i realitzar proves en temps real mitjançant hardware (per exemple, connectant una FPGA). HDL Verifier, per altra banda, és un mòdul de MATLAB que automatitza la verificació de dissenys VHDL i Verilog mitjançant simuladors HDL externs (per exemple, ModelSim). Proporciona una interfície que enllaça MATLAB i

Simulink amb ModelSim, i ho fa de manera que permet enviar estímuls des dels primers fins el codi HDL del segon, respectivament.

D'aquesta manera, des de FDATool es pot automatitzar la generació del diagrama de blocs en Simulink, especificant-ne l'estímul.

5.5.1 Freqüència de rellotge

Un factor important a l'hora de generar el diagrama de blocs és la freqüència de rellotge a la que s'executarà la simulació. La freqüència de rellotge determina la velocitat a la que es processen les mostres, de manera que estarà directament relacionada amb els temps d'execució de manera proporcionalment inversa: a més freqüència de rellotge, més mostres es processen per cicle de rellotge i, per tant, menys temps d'execució és necessari per a processar-les. Aquest paràmetre no és especificable directament a l'hora de generar el diagrama de blocs i les descripcions HDL, sinó que s'especifica indirectament a partir dels semicicles de rellotge: per al paràmetre *clock up* s'especifica un valor temporal en nanosegons, així com també per a *clock down*. El cicle de rellotge complet serà la suma dels dos semicicles, determinant aquest la freqüència de rellotge amb la següent fórmula:

Freqüència rellotge = 1 / temps de cicle rellotge (segons)

5.5.2 Tipus d'arquitectura

Un filtre digital consta d'un conjunt de coeficients multiplicadors (amb acumulador) que modifiquen les mostres d'entrada (senyal digital) mitjançant producte i acumulació. Per tant, per a implementar un filtre digital en hardware es requereixen unitats multiplicadores (i acumuladores), i aquestes unitats multiplicadores es poden disposar de múltiples formes per a realitzar els càlculs necessaris.

Així, doncs, un altre factor determinant és el tipus d'arquitectura que tindrà el filtre, la qual s'especifica a l'hora de generar-ne el bloc en Simulink i la descripció HDL per a ModelSim amb l'eina FDATool (*Generate HDL*). L'arquitectura del filtre és el tipus d'implementació que s'utilitzarà en termes de recursos (multiplicadors) en una FPGA. Hi ha, en termes generals, tres tipus d'arquitectura: paral·lela, en sèrie, o parcialment en sèrie. L'arquitectura paral·lela utilitza tantes unitats multiplicadores com siguin necessàries (segons l'ordre del filtre), a fi d'obtenir el millor resultat possible: el mínim temps de processament. Aquesta, té un "folding factor" d'1, és a dir, cada unitat multiplicadora realitza una sola tasca assignada, però, requereix de més recursos (unitats multiplicadores) i per tant més superfície lògica de la FPGA. L'arquitectura en sèrie utilitza una sola unitat multiplicadora, amb un "folding factor" major (segons l'ordre del filtre), utilitzant-la iterativament per a realitzar tots els càlculs, de manera que s'utilitzen menys recursos de la FPGA per a realitzar la mateixa tasca, però amb un cost major en termes de temps d'execució. Finalment, l'arquitectura parcialment en sèrie es troba entre les dos prèviament mencionades: ens permet utilitzar un nombre determinat d'unitats multiplicadores de la FPGA, amb un cert "folding factor" associat a aquestes. Per exemple, si necessitem 10 unitats multiplicadores per a realitzar una tasca determinada, l'arquitectura parcialment en sèrie ens permet utilitzar 5 unitats multiplicadores amb un "folding factor" de 2: cada unitat realitzaria la tasca que es repartiria entre a 2 unitats multiplicadores, havent de realitzar doncs el doble de càlculs que en una arquitectura paral·lela.

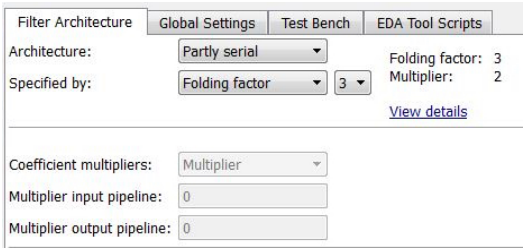

Figura 9: Arquitectura parcialment en sèrie del filtre, utilitzant 2 unitats multiplicadores amb un "folding factor" de 3

5.5.3 Diagrama de blocs i compilació

Un cop haguem completat els passos prèviament mencionats, tindrem els filtres descrits en HDL i un diagrama de blocs en Simulink que enllaça l'estímul prèviament definit ("vivaldi") amb ModelSim. El següent pas és compilar aquestes descripcions HDL dels filtres en ModelSim, executant el procés de manera automatitzada a l'hora d'iniciar ModelSim. Les comandes per automatitzar el procés de compilació i execució es mostren en la terminal de FDATool quan es genera el codi HDL i el diagrama de blocs.

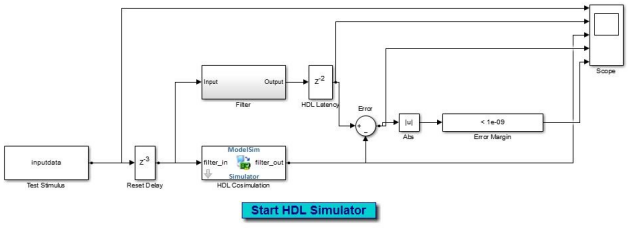

Figura 10: Diagrama de blocs en Simulink

En la Figura 10 es mostra el diagrama de blocs generat, on Inputdata és l'estímul ("vivaldi") que s'enviarà a ModelSim per a ésser processat.

5.6 Execució de la simulació

Primer de tot, caldrà determinar en nanosegons el temps d'execució de la simulació. Simulink proposa una estimació del temps d'execució necessari per a completar el processat de totes les mostres d'àudio, en base el nombre de mostres, la freqüència de rellotge definida i el tipus d'implementació dels filtres. Tot i així, augmentarem lleugerament (de l'ordre d'unitats de miler) el temps d'execució proposat per Simulink per obtenir cert marge d'error. Pel que fa a la freqüència de rellotge, la majoria de proves s'han executat amb un cicle de rellotge de 10ns (5ns clock up, 5ns clock down). Per tant, s'han executat amb una freqüència de 100MHz, fàcilment assolible pels models de FPGA actuals.

Quan s'executa la simulació, l'estímul inicial ("vivaldi") s'envia al mòdul Filter, que actua com a model ideal de comportament o "behavioral model" dels filtres generats, traient per sortida el resultat esperat (estímul inicial processat a través del filtre). L'estímul inicial s'envia també a ModelSim, on es processa a través dels filtres compilats i especificats en HDL, i finalment s'envia també a Scope. En ModelSim podrem observar gràficament l'entrada "filter_in", que és el nostre estímul "vivaldi", i la sortida "filter_out", que és la senyal resultant de processar l'estímul a través dels filtres en ModelSim. A part de les representacions gràfiques, també obtenim el temps d'execució necessari per a processar totes les mostres, situant el cursor vertical al final de la representació gràfica. Automàticament, el cursor es situarà sobre l'instant posterior a l'última mostra processada, mostrant en nanosegons a la part inferior el temps d'execució.

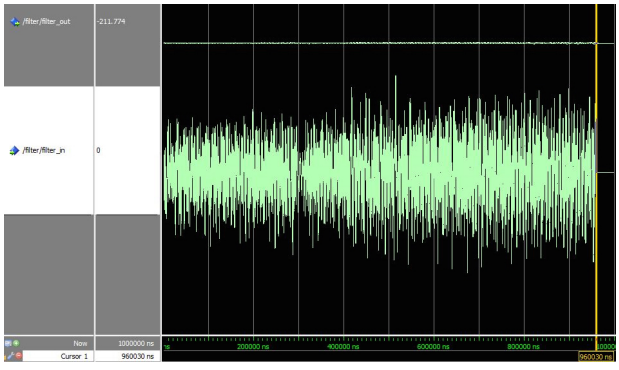

Figura 11: Representació gràfica dels senyals i temps d'execució de la simulació en ModelSim

Per a determinar el nombre de mostres processades per cicle de rellotge, es pot afegir en ModelSim la senyal "clk" (corresponent als cicles de rellotge) i incrementar l'escala de representació del senyal (zoom), de manera que puguem visualitzar les mostres una a una, i verificar els cicles de rellotge utilitzats per a cada una d'aquestes.

Per altra banda, scope, que ens permet visualitzar de manera gràfica cada un dels estímuls generats durant la simulació, rep cada un dels tres estímuls anteriorment esmentats (l'estímul inicial, l'estímul processat a través del mòdul Filter, i l'estímul processat a través de ModelSim), i n'estima el marge d'error entre els dos estímuls processats, mostrant-ne els resultats de manera gràfica. Se'n pot modificar a consciència l'escala de les unitats, a l'hora de visualitzar els resultats.

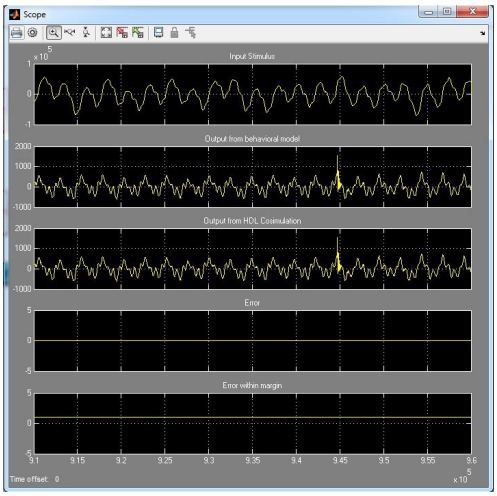

Figura 12: Visualització gràfica en el mòdul Scope dels estímuls generats en l'execució de la simulació

6 EXECUCIONS I RESULTATS

6.1 Resultats obtinguts

Per a realitzar les diferents proves s'han tingut en compte els paràmetres anteriorment descrits, i que influeixen directament en els temps d'execució: freqüència de rellotge, ordre del filtre/s, tipus d'arquitectura i nombre de mostres totals a processar (directament relacionades amb la freqüència de mostreig). S'ha fet èmfasis, però, en el tipus d'arquitectura dels filtres i en l'ordre d'aquests, ja que els temps d'execució obtinguts amb una freqüència de mostreig determinada i amb un nombre concret de mostres són fàcilment extrapolables donats uns altres valors determinats. Finalment, s'han considerat també realitzar les proves en un sol filtre passa-banda (una banda) i en el filtre d'un terç d'octava complet (30 bandes en cascada).

Així, a cada prova realitzada s'ha hagut de repetir el procés que comprèn des de la implementació dels filtres en alt nivell a MATLAB, canviant-ne l'ordre d'aquests, fins a l'execució de la simulació en Simulink i ModelSim.

Les proves s'han realitzat amb tres tipus d'ordre dels filtres: 2, 4 i 8. Per a cada ordre s'han executat els test implementant els tres tipus d'arquitectura de filtre, dins dels quals s'han tingut en compte tant l'execució d'una sola banda o filtre passa-banda, com també del filtre d'un terç d'octava complet. Aquest últim, però, només ens ha permès fer-ne una execució en una arquitectura paral·lela, de manera que no s'han pogut testejar els casos d'una arquitectura en sèrie i parcialment en sèrie. Totes aquestes proves s'han realitzat a 100MHz de freqüència de rellotge, amb l'àudio a 192KHz i 24 bits, mentre que per altra banda s'ha realitzat una breu prova a 96KHz i 24 bits, on l'únic factor que varia és el nombre de mostres a processar, i a 192KHz i 16 bits, on varia el tamany de la mostra.

Els resultats obtinguts a 192KHz i 24 bits, per a una sola banda del filtre, s'observen en la següent figura:

	Ordre 2	Ordre 4	Ordre 8
Arq. Paral·lela	960.030 ns	960.030 ns	960.030 ns
Arq. Parcial Sèrie	2.880.090 ns	5.760.180 ns	11.520.360 ns
Arq. Sèrie	5.760.180 ns	11.520.360 ns	23.040.720 ns

Figura 13: Temps d'execució (en nanosegons) d'àudio a 192KHz i 24 bits en un filtre passa-banda

Els resultats obtinguts a 192KHz i 24 bits, per al filtre d'un terç d'octava, s'observen en la següent figura:

	Ordre 2	Ordre 4	Ordre 8
Arq. Paral·lela	960.030 ns	960.030 ns	960.030 ns

Figura 14: Temps d'execució (en nanosegons) d'àudio a 192KHz i 24 bits en un filtre d'un terç d'octava

Els resultats obtinguts a 96KHz i 24 bits, per a una sola banda del filtre, s'observen en la següent figura:

	Ordre 2	Ordre 4	Ordre 8
Arq. Paral·lela	480.030 ns	480.030 ns	480.030 ns
Arq. Parcial Sèrie	1.440.090 ns	2.880.090 ns	5.760.180 ns
Arq. Sèrie	2.880.090 ns	5.760.180 ns	11.520.360 ns

Figura 15: Temps d'execució (en nanosegons) d'àudio a 96KHz i 24 bits en un filtre passa-banda

Finalment, els resultats obtinguts a 192KHz i 16 bits han sigut idèntics als obtinguts a 192KHz i 24 bits en una sola banda.

Cal precisar que en totes les execucions amb arquitectures parcialment en sèrie no s'ha utilitzat un folding factor concret, sinó que s'han utilitzat un nombre concret de multiplicadors en cada cas (2 multiplicadors exactament), i el folding factor (el nombre d'execucions que ha hagut de realitzar cada multiplicador) ha vingut donat a partir d'aquests 2 i segons l'ordre del filtre.

6.2 Anàlisi dels resultats

Principalment podem observar que els factors més determinants són el tipus d'arquitectura en que s'ha implementat el filtre, l'ordre del filtre, i el nombre de mostres a processar.

A partir de la informació donada per FDATool, es pot observar que cada ordre del filtre necessita 3 unitats multiplicadores per a obtenir el màxim rendiment, és a dir, per a obtenir el temps d'execució mínim en una arquitectura de filtre en paral·lel. Així, doncs, per a un ordre total de 2 es necessiten 6 unitats multiplicadores en paral·lel. Aquest cas es correspondria a una sola banda del filtre, ja que en un filtre d'un terç d'octava complet d'ordre 2, l'ordre total s'eleva a 60: un ordre 2 de cada filtre passa-banda que el composa, per 30 filtres passa-banda en total. Per tant, en aquest últim cas es necessiten 180 unitats multiplicadores en paral·lel. Si elevéssim l'ordre dels filtres a 8, en una banda es necessitarien 24 unitats multiplicadores en paral·lel per a una sola banda, i un total de 720 unitats multiplicadores en un filtre d'un terç d'octava, en arquitectura paral·lela. Els mateixos resultats són extrapolables a qualsevol ordre de filtre que s'utilitzi.

Pel que fa als resultats obtinguts, 960.030ns és el temps mínim d'execució que es pot obtenir amb cicles de rellotge de 10ns, és a dir, a 100MHz. Curiosament, aquest valor és idèntic tant en un sol filtre passa-banda com en un filtre d'un terç d'octava complet, però es justifica pel fet que el simulador HDL utilitza tantes unitats multiplicadores com necessiti en una implementació en paral·lel: al ser una simulació per software, no té límit físic de superfície lògica (i per tant, d'unitats de càlcul). En una FPGA física, però, arribariem a un límit a partir del qual, al no disposar de més unitats de càlcul, penalitzaria els temps d'execució en un filtre d'un terç d'octava. No podem observar les diferències tampoc entre una sola banda i 30 bandes en conjunt, ja que només s'han pogut implementar en paral·lel, de manera que queda observar com afecten les arquitectures del filtre en els resultats.

Utilitzant una arquitectura parcialment en sèrie amb 2 unitats multiplicadores, el folding factor varia segons l'ordre del filtre. En una banda d'ordre 2, utilitzant 2 multiplicadors, obtenim un folding factor de 3, és a dir, 6 unitats multiplicadores necessàries en una arquitectura paral·lela es converteixen en 2 unitats multiplicadores. Aquestes dos úniques unitats han de realitzar totes les operacions que abans es repartien entre 6 unitats, i per tant s'acaben produint el triple d'operacions en cada unitat. Això permet reduïr les unitats lògiques de la FPGA necessaries per al càlcul, però, influeïx en el temps d'execució: cada unitat realitza el triple d'operacions, de manera que el temps d'execució acaba essent el triple del temps d'execució en arquitectura paral·lela. Per tant, de 960.030ns de temps d'execució passem a obtenir un resultat de 2.880.890ns. El mateix patró lògic s'aplica amb la resta de casos de prova, segons l'ordre del filtre, tal i com es pot observar en les figures 13 i 15.

En quant a l'arquitectura en sèrie, es produeix una situació similar, però en aquest cas el nombre d'unitats multiplicadores és el mínim: 1. Així, una sola unitat multiplicadora ha de realitzar totes les operacions que abans s'executaven entre varies unitats. El resultat doncs, dependrà de la relació entre el total d'unitats multiplicadores (d'una arquitectura paral·lela del filtre) i 1. Un exemple s'observa en la figura 13, amb un ordre 8: en paral·lel es necessiten 24 unitats multiplicadores, i en sèrie una sola unitat ha de realitzar els càlculs que es repartien entre aquestes 24, de manera que el temps d'execució mínim de 960.030ns s'eleva 24 vegades fins a obtenir un total de 23.040.720 ns.

Finalment, en relació a la freqüència de rellotge, s'estima una relació inversament proporcional amb el temps d'execució, o bé directament proporcional amb el nombre de mostres processades: a més freqüència de rellotge, més mostres es processen per cicle de rellotge i, per tant, menys temps d'execució total per a processar-les. Tot i que el nombre de mostres que es processen a cada cicle depèn del tipus d'arquitectura del filtre, de manera òptima processant 1 mostra per cada cicle en una implementació paral·lela, podem simplificar afirmant que: respecte els resultats obtinguts amb 100MHz, a una freqüència de 200MHz, es processen el doble de mostres (i es

redueixen els temps d'execució a la meitat), i a 50MHz es produeïx l'efecte invers.

6.3 Implementació alternativa

Fins ara s'ha considerat una implementació tradicional d'un filtre d'un terç d'octava: 30 etapes (bandes o filtres passa-banda) en cascada o sèrie, i aquests disposats amb la seva pròpia arquitectura. Per tant, cada banda disposa dels seus propis recursos hardware assignats (multiplicadors). Es proposa, però, una alternativa, que implicaria un diferent utilització dels recursos d'una FPGA per a processar l'àudio. Aquesta, consta d'una sola banda o filtre passa-banda utilitzada iterativament per a simular les 30 bandes d'un filtre d'un terç d'octava. Així, es simula un filtre d'un terç d'octava complet, utilitzant únicament el hardware necessari per a implementar una sola etapa del filtre.

Les proves realitzades amb una sol filtre passa-banda, processant un conjunt de mostres d'àudio a 192KHz amb una durada total de 0,5 segons (un total de 96.000 mostres), mostren que s'ha necessitat un temps d'execució situat entre 960.030ns i 23.040.720 ns, segons l'ordre del filtre i el tipus d'arquitectura d'aquest. Per tant, aquest àudio es pot processar múltiples vegades fins a arribar al temps real que aquest ocupa durant la seva reproducció. En 0,5 segons (500.000.000ns) de temps en mostres, hi ha marge per processar aquest conjunt de mostres d'àudio aproximadament 520 vegades (amb una arquitectura paral·lela i un ordre 2 de filtre), és a dir, es poden arribar a processar a través de 520 bandes de filtre sense sobrepassar la capacitat de processament de la FPGA. En el cas més pessimista, de tots els que hem estudiat, es poden arribar a processar aquests 0,5 segons de mostres un total de 21 vegades amb una implementació en sèrie i d'ordre 8. En aquest cas, no es podria arribar a simular un filtre d'un terç d'octava, de manera que s'hauria de reduïr l'ordre del filtre (afectant a les freqüències de tall) o be utilitzar una implementació parcialment en sèrie: amb 2 multiplicadors (folding factor 12), es podrien processar les mostres fins a aproximadament 43 vegades, permetent simular les 30 bandes del filtre.

7. Aplicació dels resultats a un model de FPGA del mercat

Segons les especificacions del fabricant, la gama de FPGA's Spartan 6 disposen de 180 DSP slices, cada un amb un multiplicador (i acumulador) capaç d'operar a 250MHz. Per tant, disposa d'un total de 180 multiplicadors.

Així, doncs, segons els resultats obtinguts anteriorment, on no es disposava d'un nº determinat de multiplicadors que limitessin la capacitat computacional de la FPGA, podem calcular de manera aproximada la capacitat de processament d'àudio a 192Khz d'una Spartan 6.

A 100MHz, cada banda d'ordre 2 requeria de 6 multiplicadors amb una implementació en paral·lel, per a obtenir un temps de processament de 960.030ns. A 250MHz, però, es processen 2,5 cops més mostres cada cicle de rellotge (abans es processava 1 mostra per cicle, ara 2,5), és a dir, es tardarien 384.012ns a processar 96.000 mostres.

Seria possible, doncs, processar 30 bandes (el filtre d'un terç d'octava sencer) en una implementació en paral·lel ja que cada banda necessita 6 multiplicadors en aquest tipus d'implementació, necessitant 180 multiplicadors dels que ja disposa la gama de FPGAs Spartan 6. Per tant, es poden processar 0,5s d'àudio a través d'un filtre d'un terç d'octava d'ordre 2 en implementació paral·lela amb una latència d'aproximadament 0,384ms.

També és possible processar 2 inputs d'àudio a 192KHz simultàniament canviant-ne la implementació: utilitzant un folding factor de 2, és a dir, utilitzant 3 unitats multiplicadores per cada banda, es necessitarien un total de 90 unitats multiplicadores per cada àudio a processar (i per tant, podent processar 2 àudios simultàniament fins a utilitzar les 180 unitats multiplicadores de què disposen les Spartan 6). El temps necessari per a processar el total de mostres, però, seria el doble que utilitzant una implementació paral·lela, és a dir, 768.024ns en total, ja que cada unitat multiplicadora hauria de realitzar el doble de càlculs.

Una altra possible implementació seria processar un sol arxiu d'àudio amb un folding factor de 2 a través d'un filtre d'un terç d'octava d'ordre 4, ja que requereix el doble d'elements multiplicadors. No es podria realitzar una implementació en paral·lel ja que al tractar-se d'un filtre amb un ordre total de 120 es necessitarien 360 unitats multiplicadores de les que no disposa la FPGA.

Finalment, cal tenir en compte la implementació alternativa d'un filtre de terç d'octava de la que es feia menció en l'apartat "6.3 Implementació alternativa", amb una Spartan 6. El nombre de filtres implementables en una Spartan 6 dependria del tipus d'arquitectura del filtre passa-banda, i de l'ordre d'aquest filtre, però posant un exemple senzill, es podrien implementar fins a 30 filtres d'un terç d'octava d'arquitectura paral·lela(amb cada etapa d'ordre 2), al necessitar 6 multiplicadors per banda, i disposar-ne de 180.

8. Futures línies d'investigació

Una possible millora seria realitzar un aprofundiment en la utilització dels recursos de la FPGA, determinant exactament quins recursos s'estan utilitzant i el percentatge de recursos disponibles durant la execució de les simulacions, entre d'altres, de manera física. Això es podria assolir mitjançant una co-simulació en hardware amb una FPGA real (FPGA-in-the-loop), extraient les dades de recursos utilitzats mitjançant l'entorn del vendor de la FPGA. Aquests recursos, a més, no només tindrien en compte els mòduls físics de processament (multiplicadors, acumuladors), sinó també la memòria, els bits d'operació d'aquests mòduls, i altres elements comb les Look Up Tables, entre d'altres.

Per altra banda, una altra via d'investigació podria centrar-se en l'efectivitat d'utilitzar un sol filtre passa-banda per a simular un filtre d'octava o d'un terç d'octava. S'ha vist en aquest projecte que és factible realitzar-ho en termes d'eficiència i temps d'execució, però no en ter-

mes de funcionalitat: l'execució es realitza amb una mateixa banda 30 vegades, de manera que s'està simulant la càrrega computacional que demana un filtre d'un terç d'octava, però no la seva funcionalitat al no afectar a la resta de bandes de freqüència. Així, doncs, caldria veure el cost, en termes de overheat, que generaria modificar els coeficients que descriuen el filtre passa-banda per a poder assolir una banda diferent. Si aquest cost implícit a cada una de les 30 bandes, afegit en termes de temps al temps de processament de les mostres, es manté dins la capacitat de processar mostres de la FPGA, seria factible implementar un filtre d'un terç d'octava amb una mínima superfície lògica d'aquesta: la necessària per a implementar un sol filtre passa-banda. Per tant, es podria implementar amb una sola FPGA múltiples filtres i elements de processament per a processar de manera paral·lela múltiples conjunts de mostres d'àudio diferents, tot i que amb una latència més elevada que utilitzant la màxima superfície lògica de la FPGA per a calcular un sol conjunt de mostres.

Per últim, es podria aprofundir en relació a la utilització d'una FPGA com a dispositiu per a processar àudio de manera contínua. Normalment, s'utilitzen DSPs a l'hora de processar àudio de manera contínua (un clar exemple està en les interfícies d'àudio externes, utilitzades tant per reproduir com per capturar àudio), generant una latència del ordre d'unitats i de desenes de milisegons, en l'àudio considerat actualment com a estàndard: 48.000Hz de freqüència de mostreig i 16 bits de resolució . En aquest projecte s'ha treballat amb una finestra de 96.000 mostres i 192.000Hz de freqüència de mostreig, processant-les en pocs milisegons. En cas d'utilitzar finestres molt més reduïdes (entre 64 i 2048 mostres), com és el cas de les interfícies d'àudio modernes, caldria veure quina latència s'obtindria i quants canals d'àudio és capaç de processar, tenint en compte també les latències generades pels busos de transferència que enllacen amb l'origen de les mostres.

9. Conclusions

En aquest projecte s'ha intentat determinar, de manera aproximativa, el potencial de les FPGA a l'hora de processar àudio, documentant-ne les fases i posteriorment els resultats obtinguts, fent-ne un breu anàlisis. Això s'ha realitzat mitjançant simulacions, i posteriorment se n'han extrapolat els resultats a una FPGA del mercat segons el hardware del que disposa. Tot i així, només s'ha visualitzat una petita fracció, quedant doncs com a futures millores una experimentació amb FPGAs físiques, realitzant-ne exhaustives proves per a determinar-ne sense marge d'error la seva capacitat de processar senyals, i aprofundint sobre la importància de cada un dels elements hardware d'aquestes en a l'hora de processar àudio.

Agraïments

L'autor vol utilitzar aquesta secció per agraïr al seu tutor, Màrius Montón, la direcció i supervisió d'aquest projecte.

Bibliografia

1. HDL Verifier [en línia]. [Consulta: Abril de 2015]. Autor: MathWorks. Nautick, Massachusetts, EE.UU: The MathWorks, Inc. Disponible a internet: http://es.mathworks.com/products/hdl-verifier/
2. Fdatool [en línia]. [Consulta: Abril de 2015]. Autor: MathWorks. Nautick, Massachusetts, EE.UU: The MathWorks, Inc. Disponible a internet: http://es.mathworks.com/help/signal/ref/fdatool.html
3. Mentor Modelsim. [en línia]. [Consulta: Abril de 2015]. EE.UU: Mentor Graphics, Disponible a internet: http://www.mentor.com/products/fv/modelsim/
4. Modelsim Tutorial. [en línia]. [Consulta: Maig de 2015]. Tietotekniikan laitos - Department of Pervasive Computing. Disponible a internet: http://www.tkt.cs.tut.fi/tools/public/tutorials/mentor/modelsim/getting_started/gsms.html
5. Field Programmable Gate Array (FPGA) [en línia]. [Consulta: Maig de 2015]. Autor: Xilinx. EE.UU: Xilinx Inc. Disponible a internet: http://www.xilinx.com/training/fpga/fpga-field-programmable-gate-array.htm
6. All about FPGAs. Autor: Bob Zeidman, EETimes [en línia]. [Consulta: Maig de 2015]. Disponible a internet: http://www.eetimes.com/document.asp?doc_id=1274496
7. MATLAB. Autor: MathWorks. [en línia]. [Consulta: Maig de 2015]. Autor: MathWorks. Nautick, Massachusetts, EE.UU: The MathWorks, Inc. Disponible a internet: http://es.mathworks.com/products/matlab/
8. Hardware Description Language. [en línia]. [Consulta: Maig de 2015]. Autor: Texas A&M University, Computer Science and Engineering Department, EEUU. Disponible a internet: faculty.cs.tamu.edu/ejkim/Courses/cpsc350/slide8.ppt