Centro de informática
Universidade Federal da Paraíba

Relatório final
Pedal de Guitarra

José Felipe Nunes da Silva
Lenildo Luan Carlos
Rodrigo da Costa Ramalho

João Pessoa, 2018

Centro de informática
Universidade Federal da Paraíba

Relatório final
Pedal de Guitarra

Relatório elaborado para o projeto final da disciplina Circuito Lógicos II, ministrada pelo Professor Eudisley Gomes dos Anjos do Centro de Informática da Universidade Federal da Paraíba.

João Pessoa, 2018

Resumo

A música é um ramo de grande rendimento para a economia, estando presente diariamente na vida das pessoas por meio de propagandas, filmes, novelas e concertos. Sendo assim, todos os dias surgem novas bandas e músicas que agradam os mais diversos gostos. No entanto, periodicamente o ramo da música se reinventa, criando estilos musicais mais inovadores, que muitas vezes utilizam de soluções tecnológicas para gerar novas melodias e sons. Dessa forma, buscando baratear a produção dessas novas melodias, aumentar a eficiência do processamento do sinais e aprender a analisar sinais usando o FPGA, buscamos gerar efeitos de áudio que podem ser adicionados ao som digitalmente e criar novas melodias. Por fim, implementamos o filtro FIR para tratar os ruídos do áudio e adicionamos os efeitos de eco e driver nos sons processados pelo FPGA.

Palavras-chave: FPGA, música, áudio, efeitos.

Lista de siglas

FPGA - Field Programmable Gate Array (Arranjo de portas programáveis em campo)

Sumário

1. Introdução

A música é um dos principais elementos da sociedade, sendo um aspecto social tão forte que pode caracterizar desde grupos sociais até culturas inteiras. Há indícios que desde a pré-história se produzia música, tendo vestígios de flautas construídas com ossos de 60.000 a.c.

Ao longo do tempo foram criados diversos tipos de instrumentos, sendo hidráulicos, como os primeiros órgãos, pneumáticos como as flautas e mecânicos, sendo de percussão, cajon, ou de cordas, violão. Porém todos esses instrumentos são acústicos, ou seja, funcionam sem a necessidade de energia elétrica. Com o advento da energia elétrica surgiram instrumentos elétricos, como a guitarra.

O conhecimento acerca da eletricidade não proporcionou somente o surgimento de novos instrumentos, mas também trouxe a possibilidade de manipular o áudio através do processamento de sinais. Com isso, surgiram vários equipamentos para instrumentos eletrônicos que mudam a saída do som, como os pedais de guitarra.

Como processamento de sinais em tempo real demanda uma quantidade alta de cálculos, e é necessário uma resposta rápida para gerar resultados satisfatórios, surgiram novas tecnologías para tratar problemas como esse. Uma delas foi o FPGA, que não só é mais eficiente que os processadores, mas também exige bem menos consumo.

O objetivo deste trabalho é desenvolver um pedal de guitarra utilizando FPGA. Com isso, pretende aprimorar os conhecimentos em verilog e aprender sobre processamento digital de áudio.

2. Metodologia

O desenvolvimento deste projeto passou por quatro etapas: levantamento de trabalhos relacionados, equipamentos necessários, programação do FPGA e testes.

2.1 Levantamento de trabalhos relacionados

Foram pesquisados artigos científicos e repositórios de projetos relacionados a processamento digital de áudio. Nessa pesquisa foram encontradas diversas técnicas para desenvolver os efeitos desejados, assim como linhas de código que nortearam nossa lógica de programação.

2.2 Equipamentos necessários

Para este projeto foi usado o kit educacional da Altera FPGA DE2 Cyclone II EP2C35F672C6, uma caixa de som amplificada da marca Meteoro, dois adaptadores de pino P2 para P10, dois cabos auxiliares P2 macho-macho e uma guitarra.

2.3 Programação do FPGA

O FPGA foi programado utilizando o ambiente de desenvolvimento integrado Quartus 2 no Ubuntu 16.04. Também foi necessário instalar o USB Blaster para a comunicação da placa de desenvolvimento com o computador.

2.4 Testes

Para realizar os testes, inicialmente, foi utilizado um cabo auxiliar P2 macho-macho conectando a saída de áudio do celular com a entrada de som da placa de desenvolvimento, foi colocado a gravação do som de uma guitarra sem efeito e verificado se o som de saída condizia com o desejado. Posteriormente foi conectado a guitarra à placa de desenvolvimento, com o auxílio de dois adaptadores P2 para P10 e um cabo auxiliar. Para captar a saída de som, foi utilizado um Headphone ou uma caixa amplificadora.

3. Descrição do Projeto

O projeto busca imitar um pedal de guitarra, onde os tipos de efeitos são acionados por meio dos três primeiros "*switchers",* sendo o primeiro responsável por acionar o filtro FIR, o segundo para adicionar eco à saída de áudio e, por fim, o terceiro adicionar o efeito de overdrive à saída de áudio.

3.1 *FIR - Finite Impulse Response*

O filtro FIR, sigla para *"Finite Impulse Response"*, é caracterizado por um sistema finito de somas de várias amostras de áudio, que são alteradas por escalares que modificam sua amplitude, neste tipo de filtro não é utilizado amostras de áudios anteriores ou futuros.

Este filtro pode ser representado pela seguinte expressão matemática, onde y é o resultado da implementação do filtro, o b representa os coeficientes do filtro e x é o sinal não tratado. O "n" é a ordem do filtro, tal que quanto maior seu valor, mais complexo o filtro será.

$$y[n] = \sum_{i=0}^{N} b_i x[n-i]$$

3.2 Efeitos de Áudio

Após a filtragem do sinal, iniciamos a implementação dos efeitos de áudios que são ativados quando os "*switchers*" 2 e 3 estão em nível lógico 1 e os demais em 0. Quando o "switcher" 2 está ativado, o efeito de eco é ativado.

3.2.1 Efeito de Eco

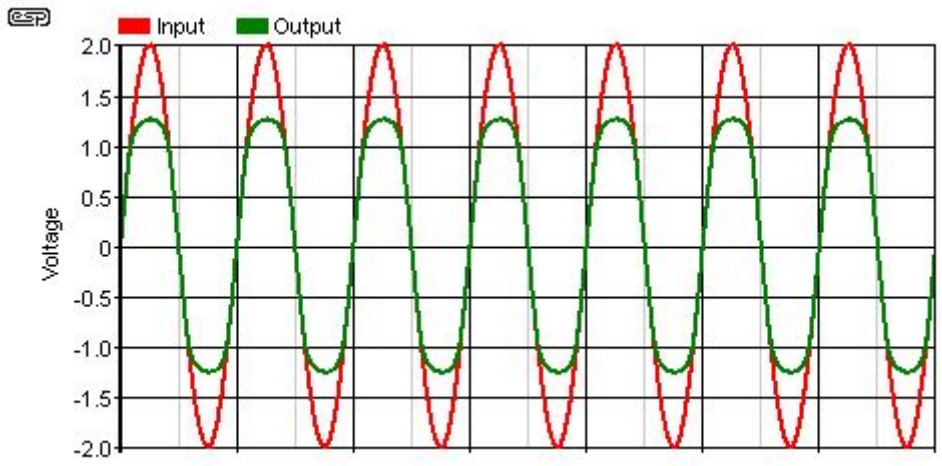
O efeito de eco ocorre quando o som é refletido em um intervalo pequeno o bastante para se distinguir do som original. Dessa forma, sua implementação na placa consiste em adicionar a amostra de áudio original com um pequeno atraso de tempo a amostra de áudio de saída um pequeno intervalo de tempo.

O efeito de eco pode ser representado pela seguinte expressão matemática, onde "y" é o sinal de áudio com efeito do eco, "x" é o sinal de áudio tratado sem o feito de áudio, "a" e "b" são coeficientes de ponderação, "n" é a quantidade de amostras de áudio e "t" é o tempo de delay necessário para gerar o efeito de eco nas amostras.

$$y_{echo}[n] = ax[n] + bx[n - t_{echo}]$$

3.2.2 Efeitos de *Overdrive*

O efeito de overdrive ocorre quando um amplificador chega na sua amplitude máxima e passa a perder precisão na saída do sinal, já que parte deste sinal é truncado. Sendo assim, nós aumentamos a amplitude da amostra o multiplicando por sete e o armazenamos em uma variável com uma quantidade de bits menor que o sinal, dessa forma, simulamos um amplificador chegando na sua amplitude máxima.



A onda vermelha se refere a amostra de áudio sem o efeito do overdrive e a onda verde é a amostra com o efeito do overdrive.

4. Execução do Projeto, Testes e Resultados

4.1 Execução do Projeto

Para a realização do projeto foi utilizado a ide Quartus II da Altera, sendo a programação da placa FPGA realizada na linguagem de descrição de hardware Verilog. Tendo como base o código que encontramos na nossa pesquisa, no qual estava implementado o filtro FIR e uma primeira versão do efeito de Eco.

Dividindo o código em módulos, para melhor legibilidade e manutenção do projeto, foram definidos, além dos módulos já existentes como os do CODEC de Áudio, 3 módulos, o do filtro FIR responsável por tratar o sinal retirando os ruídos indesejados, o módulo do efeito de eco, que recebe uma amostra de áudio, guarda e soma à saída final com um delay, fazendo com que a mesma amostra de áudio se repita n vezes num determinado espaço de tempo, e o módulo do overdrive que recebe o sinal, aumenta a sua amplitude o bastante para distorcer o som e trunca o seu pico superior e inferior, porém diminui o ganho do circuito para que o som do resultado seja audível.

O sinal recebido é sonoro, ou seja, analógico, porém ao passar pelo módulo CODEC da FPGA, é discretizado e pode ser tratado digitalmente. As amostras tornam-se, para o verilog, vetores de bits, que são manipulados alterando seus valores, com deslocamento de bits, ou alterando o tamanho desses vetores, depois de recebidos, ou seja, truncando o sinal.

Para a implementação do filtro FIR, o sample de entrada tem o tamanho de 16 bits, é o bastante para que o ruído possa ser detectado e eliminado, deslocando a parte constante da amostra e somando à anterior, no fim esse sample tratado é concatenado com o anterior e o resultado é um som limpo.

Após discretizada, a amostra de sinal se torna um vetor de números binários, guardando um pedaço desse vetor e o adicionando à saída com um atraso, foi possível produzir o efeito de eco. Onde pode variar o tamanho da amostra de som que “ecoa”, o tempo que que ela leva para se repetir e a quantidade de vezes que ela se repete.

Para produzir o efeito de overdrive, a amostra captada tem originalmente 16 bits, e uma amplitude que foi aumentada em 10 vezes, com a truncagem do vetor no qual é alocada a amostra, esse ganho na amplitude é reduzido porém o sinal continua distorcido, porém audível, foi preciso fazer ele passar por um segundo FIR para eliminar os ruídos indesejados e reproduzir apenas o som com a distorção proposital.

4.2 Testes

Para testar o projeto durante o seu desenvolvimento, utilizamos como entrada hora áudios gravados no celular, hora microfone, e por fim uma guitarra, para a saída, utilizamos inicialmente fones de ouvido sterio e posteriormente uma caixa de som amplificada.

Os testes foram necessários para saber se o som que estava sendo reproduzido era o desejado e entender o porquê do som ser reproduzido daquela maneira, fazendo modificações no código. Com os testes, foi possível deduzir o “tamanho” ideal do eco, a amplitude e o ganho do efeito de overdrive, o quanto o vetor precisava ser truncado e notar que a amostra precisava passar por um segundo FIR depois de passar pelo overdrive.

4.3 Resultados

Ao final dos testes, obtivemos um módulo de filtro do tipo FIR capaz de eliminar os ruídos presentes no sinal recebido pela FPGA e devolvê-lo limpo, um de efeito de eco, que recebe um sinal limpo e retorna o mesmo sinal porém repetindo parte dele e um terceiro módulo que capaz de aplicar um efeito de overdrive no sinal de entrada, ou seja, distorcer o som porém de forma que seja audível e que dê para distinguir bem o som.

5. Conclusões

Com a realização desse projeto, concluímos que é possível produzir um pedal de efeitos sonoros para guitarras utilizando um kit FPGA, programado através da IDE Quartus II, e a linguagem de descrição de hardware Verilog. Foi possível obter resultado semelhante às marcas de pedais presentes no mercado, porém com maior custo benefício, devido a alta capacidade de processamento da FPGA e a possibilidade de ter vários efeitos em um só equipamento compacto.

6. Referências

LAL, Vipin. **Synthesiable Verilog code for a 4 tap FIR Filter.** 2015. Disponível em: <http://verilogcodes.blogspot.com/2015/11/synthesiable-verilog-code-for-4-tap-fir.html>. Acesso em: 7 jun 2018.

SCHULTZ, Richard. **FPGA Implementation of Audio Effects - An EE 552 Student Application Note.** Disponível em: <http://www.ece.ualberta.ca/~elliott/ee552/studentAppNotes/2003_w/misc/Audio_Effects/FPGA-effects.pdf>. Acesso em: 8 jun. 2018.

IOWEGIAN INTERNATIONAL. **FIR Filter Basics.** Disponível em:

<http://dspguru.com/dsp/faqs/fir/basics/>. Acesso em: 7 jun. 2018.

ROD ELLIOTT (ESP). **Soft Clipping.** Disponível em: <http://sound.whsites.net/articles/soft-clip.htm>. Acesso em: 08 jun. 2018.

WICKERT, Mark. **FIR Filter.** Disponível em: <http://www.eas.uccs.edu/~mwickert/ece2610/lecture_notes/ece2610_chap5.pdf>. Acesso em: 08 jun. 2018.