Simulação de um efeito não-linear de distorção de guitarra elétrica utilizando a GPU

Jonathan W. Guimarães[1], **Wagner M. Nunan Zola** [1]

[1]Departamento de Informática – Universidade Federal do Paraná (UFPR)
Curitiba – PR – Brasil

jonathanguimaraes@ufpr.br,wagner@inf.ufpr.br

Resumo. *O trabalho propõe-se a implementar um modelo de simulação de distorção não-linear do sinal da guitarra elétrica chamado Waveshape, que divide esse filtro em três sub-filtros: ganho, overdrive e passa-baixas. Implementamos o algoritmo serial executado na CPU e uma versão paralela na GPU, usando CUDA.*

1. Introdução

A guitarra é um dos instrumentos mais tocados e apreciados ao redor do mundo. Pedais de efeito, que são analógicos eletrônicos, que recebem o sinal da guitarra como entrada e entregam um sinal processado na saída [Sunnerberg 2019]. A distorção é um tipo de efeito que ceifa o pico do sinal original, na altura de um parâmetro variável que determina o quão distorcido será o sinal de saída. Os melhores pedais de distorção analógica utilizam válvulas eletrônicas, que aplicam um filtro não-linear no sinal produzindo harmônicos que se incorporam ao timbre do instrumento. Esse trabalho consistiu na simulação digital de um efeito de distorção de guitarra. Foi feito um algoritmo linear, na CPU e um algoritmo paralelo na GPU, com o intuito de apresentar baixa latência, para futuro uso em tempo real.

2. Fundamentos teóricos e metodologia

Podemos considerar que a válvula dos amplificadores de guitarra é um elemento de produção de distorção não-linear [Sunnerberg 2019]. Um dos problemas em construir um modelo digital que a represente está na subjetividade do que é considerado bom para o ouvido do guitarrista em questão. Independente dessa subjetividade, existem dois parâmetros que permitem ao músico controlar o som da melhor maneira para ele, a saber, *wildness* e *sharpness* (chamados de *Drive* e *Tone*) [Chang 2011]. O modelo de simulação de distorção escolhido utiliza esses parâmetros juntamente com o sinal da guitarra para realizar a distorção. Para isso utiliza-se de 3 fases de processamento do sinal: um ganho inicial, seguido do corte da onda (produzindo o *overdrive*) e um filtro passa-baixas, a fim de evitar as altas frequências indesejadas produzidas pelo corte do sinal. Para cada uma dessas etapas foi feito um algoritmo para execução CPU e outro para a GPU. A programação em GPU pode ser vista como um meio para tornar algoritmos de processamento de áudio digital mais eficiente, devido ao alto grau de paralelismo proporcionado por esses dispositivos [Lourenço 2009].

2.1. Implementação e detalhes

O algoritmo de ganho, primeira etapa do processo, consistiu em multiplicar o sinal de entrada por um parâmetro com o intuito de modificar a amplitude desse sinal no domínio

do tempo.
A segunda etapa do filtro consistiu na distorção digital *Waveshape*, uma relação entre senóides aplicadas ao sinal de entrada, conforme as equações [Chang 2011]:

$$x' = ((1 + k) * x)/(1 + k * abs(x)), \tag{1}$$

onde

$$k = (2 * a)/(1 - a); a = sin(((drive + 1)/101) * (pi/2)); \tag{2}$$

A última etapa do algoritmo é o filtro passa-baixas. Para isso, realizamos a transformada de Fourier do sinal (FFT) [Moreland and Angel 2003] , filtrando as frequências altas a partir do parâmetro *Tone*, controlado pelo instrumentista, retornando com o sinal para o domínio do tempo com a transformada inversa.
Para os algoritmos em CPU foram utilizados laços percorrendo todo o sinal . O algoritmo para GPU foi escrito utilizando a linguagem CUDA, da Nvidia. Foi utilizado um *grid* unidimensional de 32 *threads* com total de *threads* de acordo com o tamanho do número de amostragens do sinal de entrada.
O ambiente de execução dos algoritmos possuía um processador I7 de sétima geração e uma placa Nvidia 940MX.

3. Resultados e discussão

Foi criado um programa que lê o arquivo de áudio de guitarra previamente gravado e armazenado em disco e o envia para ambos os fluxos de execução: um responsável pelos três filtros dentro da CPU e o outro pela vesão paralela em GPU. Ao fazer isso ele armazenou o tempo de execução de cada uma das partes. A CPU executou as três partes da distorção, respectivamente ganho, overdrive e passa-baixas em 0.35 segundos, 0.44 segundos e 0.8 segundos. Já a GPU fez essas partes em 0.05 segundos, 0.06 segundos e 0.1 segundos. No total A CPU levou 1.59 segundos, enquanto a GPU demorou 0.21 segundos. Podemos ver um ganho de velocidade de 7.56 vezes, ao compararmos os algoritmos em GPU com os algoritmos em CPU.

4. Conclusão

Esse trabalho evidenciou a infinidade de usos da GPU para algoritmos de processamento paralelo. Podemos ver que o processamento do sinal de áudio utilizando a GPU foi mais rápido obtendo uma aceleração 7.56, quando comparado com o processamento sequencial em CPU. Percebemos que é possível entregar sinal de áudio com qualidade, baixa latência e em tempo real utilizando a GPU. Isso amplia o universo dos simuladores digitais de áudio, dando a eles uma capacidade maior de processamento, ao consider um mesmo intervalo de tempo de reprodução o que viabiliza o uso desses algoritmos para aplicações de áudio em tempo real.

Referências

Chang, C.-H. (2011). A guitar overdrive/distortion effect of digital signal processing.

Lourenço, L. H. A. (2009). Processamento paralelo de áudio em gpu.

Moreland, K. and Angel, E. (2003). The fft on a gpu.

Sunnerberg, T. D. (2019). Analog musical distortion circuits for electric guitars.