UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

DESENVOLVIMENTO EM HDL DE UMA ALTERNATIVA DE HARDWARE DEDICADO PARA SÍNTESE DE ÁUDIO UTILIZANDO O PROTOCOLO MIDI

TRABALHO DE CONCLUSÃO DE CURSO

Isaías Bittencourt Felzmann

Santa Maria, RS, Brasil
2015

DESENVOLVIMENTO EM HDL DE UMA ALTERNATIVA DE HARDWARE DEDICADO PARA SÍNTESE DE ÁUDIO UTILIZANDO O PROTOCOLO MIDI

Isaías Bittencourt Felzmann

isaias.felzmann@ecomp.ufsm.br
ibfelzmann@gmail.com

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Computação da Universidade Federal de Santa Maria como requisito parcial para obtenção do grau de
Bacharel em Engenharia de Computação

Orientador Prof. Dr. Everton Alceu Carara

Santa Maria

2015

Universidade Federal de Santa Maria – UFSM
Centro de Tecnologia – CT
Curso de Engenharia de Computação

A Comissão Examinadora, abaixo assinada, aprova o Relatório de Trabalho de Conclusão de Curso

DESENVOLVIMENTO EM HDL DE UMA ALTERNATIVA DE HARDWARE DEDICADO PARA SÍNTESE DE ÁUDIO UTILIZANDO O PROTOCOLO MIDI

elaborado por
Isaías Bittencourt Felzmann

como requisito parcial para obtenção do grau de
Bacharel em Engenharia de Computação.

COMISSÃO EXAMINADORA:

_____
Prof. Everton Alceu Carara, Dr.
(Presidente/Orientador)

_____
Prof. Cesar Augusto Prior, Dr.
(UFSM)

_____
Prof. Leonardo Londero de Oliveira, Dr.
(UFSM)

Santa Maria, RS, 17 de dezembro de 2015.

RESUMO

Trabalho de Conclusão de Curso
Curso de Engenharia de Computação
Universidade Federal de Santa Maria

DESENVOLVIMENTO EM HDL DE UMA ALTERNATIVA DE HARDWARE DEDICADO PARA SÍNTESE DE ÁUDIO UTILIZANDO O PROTOCOLO MIDI

Autor: Isaías Bittencourt Felzmann
Orientador: Prof. Dr. Everton Alceu Carara
Santa Maria, 17 de dezembro de 2015

Um sintetizador de áudio é um sistema capaz de reproduzir notas musicais de acordo com um padrão determinado, imitando instrumentos reais. Em teoria musical, a altura de uma nota está diretamente relacionada com a frequência da harmônica fundamental, e as características (timbre) de um instrumento com a distribuição das harmônicas, todas múltiplas em frequência da primeira. O analisador *Loris* modela sons musicais decompondo-os em harmônicas, que são representadas como senoides de frequência e amplitude variáveis no tempo. Baseados nessa modelagem, propomos um sintetizador, de *hardware* dedicado, que recria os sons modelando-os a partir das múltiplas senoides, com base em comunicação feita pelo protocolo MIDI. O principal desafio é o aspecto temporal, pois a criação de cada uma das senoides envolve custosas operações de multiplicação, e é desejável a geração de milhares de senoides em alguns milionésimos de segundo, pois os sintetizadores são comumente avaliados segundo o número de notas que podem reproduzir em simultâneo. O sistema proposto, limitado pelos recursos do dispositivo de lógica programável em que foi testado, é capaz de gerar até 2048 senoides em simultâneo, o que foi o suficiente para reprodução de melodias de um único instrumento de forma correta, com boa performance e qualidade de som suficientemente fiel, embora alguns problemas de ruído sejam perceptíveis. Esse resultado é uma prova do conceito proposto e pode servir como base para trabalhos futuros para adição de funcionalidades, bem como melhoria das já desenvolvidas.

Palavras-chave: Música. Sintetizador. MIDI. Loris.

ABSTRACT

Undergraduate Final Project
Computer Engineering, Bachelor's degree
Universidade Federal de Santa Maria

DEVELOPMENT OF A DEDICATED HARDWARE AS AN ALTERNATIVE FOR AUDIO SYNTHESIS USING HDL AND THE MIDI PROTOCOL

Author: Isaías Bittencourt Felzmann
Supervisor: Prof. Dr. Everton Alceu Carara
Santa Maria, 17 December 2015

An audio synthesizer is a system able to play musical notes according to a determined pattern, mimicking real instruments. In musical theory, a note pitch is directly related to the fundamental harmonic frequency, as is an instrument characteristics (timbre) to the harmonics distribution, given that all harmonics are multiples in frequency of the first one. The *Loris* analyser models musical sounds decomposing them into harmonics, represented as frequency- and amplitude-variable in time sine waves. Based on this modelling, we propose a synthesizer, implemented in dedicated hardware, that recreates the sounds by modelling them from the multiple sine waves, based on communication using the MIDI protocol. The main challenge is the time aspect, because creating each one of the waves uses resource-consuming multiplication operations, and thousands of those should be generated in a few microseconds, because synthesizers are commonly evaluated according to the number of simultaneous notes they can play. The proposed system, limited by the programmable logic device in which it was tested, can generate up to 2048 simultaneous sine waves, enough for playing single-instrument music correctly, with good performance and realistic enough sound quality, although some noise issues are perceptible. This result is a proof of the proposed concept and may base future work on adding new features and improving the already developed ones.

Keywords: Music. Sinthesizer. MIDI. Loris.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ANEXOS

LISTA DE APÊNDICES

SUMÁRIO

INTRODUÇÃO

Um sintetizador de áudio é um sistema capaz de, com base em um conjunto de cálculos computacionais ou amostras de áudio coletadas previamente, imitar o som de instrumentos musicais reais. Geralmente, utilizam o protocolo *MIDI* (MIDI MANUFACTURES ASSOCIATION, 2015a), desenvolvido para transcrever computacionalmente partituras musicais, orientando o sintetizador sobre qual nota de qual instrumento deve ser tocada por quanto tempo.

Sintetizadores são geralmente de alto custo, principalmente quando apresentam funções mais avançadas, qualidade de som mais semelhante aos instrumentos reais e capacidade de execução de mais notas em simultâneo. Essas características aumentam a complexidade dos cálculos necessários, e, por tratar-se de um sistema desenvolvido para trabalhar em tempo real, facilmente excedem a capacidade de processadores de propósito geral mais simples, para sistemas embarcados.

Assim, foi investigada, em um projeto anterior (FELZMANN *et al.*, 2014), a possibilidade de síntese de sons em um dispositivo dedicado, a partir de modelos de som aditivos, que modelam o som em um somatório de senoides. O projeto prosseguiu até o estágio de síntese de múltiplas senoides em simultâneo, como prova de conceito, porém de maneira bastante lenta por ser controlado por um processador separado.

O presente projeto é uma continuação dessa ideia anterior, em que se espera obter uma solução mais completa e eficiente, com controle dedicado, que apresente uma boa experiência de uso e qualidade de som para composições mais simples, ou seja, que envolvem poucos instrumentos e notas em simultâneo.

1 OBJETIVOS

De um modo mais amplo, o objetivo principal deste projeto é apresentar uma solução em *hardware* dedicado para síntese de áudio, capaz de decifrar mensagens *MIDI* básicas e executar sons musicais em simultâneo. Atua como uma prova de conceito, uma tentativa de se obter uma solução eficiente e dedicada para síntese de áudio. Assim, espera-se que o desenvolvimento de um hardware dedicado auxilie em uma possível solução para melhoria na eficiência de dispositivos sintetizadores, sem a utilização de controladores de propósito geral.

Assim, o projeto envolve diversos conceitos de teoria musical, modelagem de sons, descrição de *hardware* e protocolos de comunicação exclusivos para execuções musicais, e subdivide-se em diversos objetivos mais específicos:

- Estudo da representação musical e seu significado físico;
- Estudo da representação do som a partir de componentes periódicas independentes;
- Desenvolvimento de um *hardware* para geração de uma das componentes periódicas;
- Expansão do *hardware* de geração para várias componentes em simultâneo;
- Estudo do protocolo MIDI e suas principais características;
- Desenvolvimento de um *hardware* para interpretação do protocolo e controle da geração;
- Adaptação do sistema para a possibilidade de reproduzir amostras de som pré armazenadas;
- Criação das amostras segundo a representação do som;
- União dos subsistemas para verificações funcionais finais.

2 JUSTIFICATIVA

É comum a avaliação de qualidade de sintetizadores em relação ao número de notas em simultâneo que são capazes de executar. A síntese de áudio, embora derivada de conceitos relativamente simples, envolve uma série de operações matemáticas e requer uma resposta em tempo real. Um processador moderno não teria dificuldades em executar tais operações, porém, quando se consideram controladores de mais baixo custo para dispositivos mais portáteis, as limitações de recursos podem começar a afetar a eficiência e, consequentemente, a experiência musical do ouvinte.

O trabalho aqui disposto não pretende, da maneira como é descrito, substituir soluções de sintetizadores comerciais já existentes. Os padrões MIDI (MIDI MANUFACTURES ASSOCIATION, 2015a) e General MIDI (MIDI MANUFACTURES ASSOCIATION, 2015b) definem uma série de características que o dispositivo aqui descrito não atinge. Porém, acredita-se que, ao executar a funcionalidade mais básica de um sintetizador de áudio através de um hardware dedicado, seja possível derivar possíveis soluções dedicadas para as demais características.

3 REVISÃO DE LITERATURA

A síntese de áudio MIDI envolve a tradução do protocolo MIDI (MIDI MANUFACTURES ASSOCIATION, 2015a) em eventos de instrumentos musicais, que podem ser interpretados como a execução de sons arbitrários. Tais sons podem ser matematicamente modelados para reprodução por meio de um dispositivo eletrônico. A presente seção aborda os conceitos essenciais para compreensão do mecanismo proposto. Primeiramente, são apresentados alguns conceitos de teoria musical e sua influência no sistema. Logo após, descreve-se a técnica de modelagem de som em funções seno independentes, que será reproduzida em *hardware*. Por fim, a terceira subseção resume as principais características e a operação fundamental do protocolo MIDI.

3.1 Conceitos de Teoria Musical

Esta primeira parte da revisão elenca alguns conceitos de composição e teoria musical essenciais para a compreensão das demais etapas deste projeto. É importante salientar, porém, que grande parte destes conceitos é compreendida de maneiras diferentes (como, por exemplo, as diferentes escalas musicais existentes) e, por vezes, percebida de maneiras diferentes. As definições aqui apresentadas estão contextualizadas de acordo com as convenções comuns da música ocidental e, quando aplicável, estabelecem delimitações válidas para este projeto.

3.1.1 Altura, frequência e escalas

Altura (comumente referenciada pelo termo em inglês *pitch*) é uma medida subjetiva da percepção, pelo ouvido humano, que quantifica o quão grave (ou agudo) é determinado som. Diferentemente de *frequência*, que é uma grandeza física possível de ser medida com exatidão, a altura do som depende do observador e do meio, o que motivou muitos estudos sobre como o cérebro humano percebe diferentes frequências (PLACK; OXENHAM; FAY,

2005). Para o escopo deste projeto, entretanto, é possível adotar uma correspondência direta entre altura e frequência, que será melhor compreendida nas subseções seguintes.

Assim, diferentes valores de altura são agrupados em uma convenção de *escala* musical. Existem muitas escalas musicais já desenvolvidas e utilizadas, como um retrato da música de uma época ou região. Provavelmente, a convenção mais comumente conhecida no contexto geográfico e temporal em que este trabalho está inserido é a escala em *Dó maior* (*C*, na notação anglo-saxã), aquela composta pelas notas *Dó (C)*, *Ré (D)*, *Mi (E)*, *Fá (F)*, *Sol (G)*, *Lá (A)* e *Si (B)*, nessa ordem. Essa escala, porém, não apresenta uma razão fixa de altura entre duas notas subsequentes.

Derivada da escala em *Dó maior*, a *escala cromática* é uma escala de doze notas que adiciona os tons intermediários, ou acidentes, às sete previamente citadas, de modo a formar uma razão fixa em altura (na nomenclatura de teoria musical, um espaçamento de um semitom) entre quaisquer duas notas adjacentes da escala.

A Ilustração 1, a seguir, apresenta uma maneira simples para a associação das duas escalas musicais apresentadas, situando-as nas teclas de um piano. A escala em Dó maior utiliza apenas as teclas brancas do piano, enquanto a cromática inclui também as teclas pretas. Em um teclado de piano, a razão em altura entre quaisquer duas teclas adjacentes (sejam elas brancas ou pretas) é constante, nomeadamente: de um semitom.

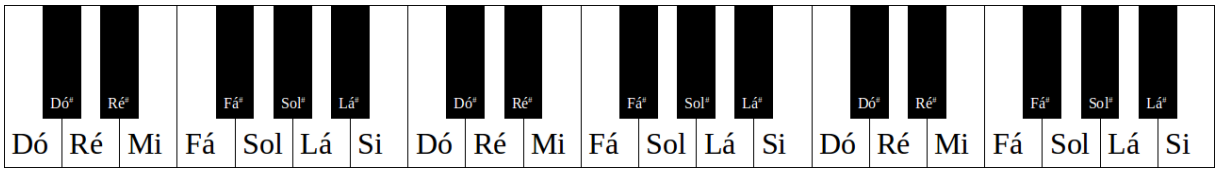


Ilustração 1: Escalas em Dó maior e cromática no teclado de um piano

A compreensão da escala cromática e da sua razão em frequência entre duas notas é essencial para a compreensão deste trabalho, pois aquela apresenta uma correspondência direta com a sequência de notas na representação MIDI, a ser abordada, e essa é a base do mecanismo para derivação de uma nota a partir de outra, de acordo com os intervalos de oitavas, conforme o conceito a seguir.

3.1.2 Oitavas e razão em frequência

Uma *oitava* é o intervalo entre uma nota e outra de mesmo nome subsequente na escala musical, ou seja, o intervalo entre duas alturas distintas em que se percebe a mesma nota, porém em uma posição de altura imediatamente mais aguda ou mais grave que a anterior. Por exemplo, na Ilustração 1, é de uma oitava a distância entre o primeiro e o segundo *Dó*. Também é definida como o intervalo entre duas alturas em que a razão de frequência entre elas é 2 (DOURADO, 2008). Tal relação entre o intervalo de uma oitava e uma razão de frequência ocorre naturalmente, a ponto de ser referenciado como "o milagre básico da música" (COOPER, 1973; LEVY; LEVARIE, 1985). De fato, segundo Levy e Levarie (1985), foi essa relação que transformou o domínio contínuo e infinito de alturas e frequências relacionadas do som em um domínio discreto e finito, possibilitando a existência das notas musicais como conhecemos hoje.

A partir da razão natural de frequências entre duas oitavas, pode-se definir matematicamente a razão em frequência de um semitom, ao considerar a escala cromática de 12 notas igualmente espaçadas em um semitom, conforme a definição anterior. Assim, a razão em frequência entre quaisquer duas notas subsequentes da escala cromática é dada pela Equação 1.

$$\frac{f_2}{f_1} = \Delta f = 2^{\frac{1}{12}} \approx 1{,}059463094 \qquad (1)$$

Equação 1: Razão em frequência de um semitom

3.1.3 Padrão de afinação A440

A razão em frequência de um semitom, definida anteriormente, permite a derivação da frequência de qualquer nota da escala cromática a partir da frequência de uma nota conhecida. Atribuir uma frequência arbitrária a uma nota conhecida é o processo que se conhece comumente por *afinação*.

A norma *ISO 16:1975* define o padrão de afinação conhecido como *A440*, onde a nota *Lá* (*A*, na notação anglo-saxã, a que se atribui o nome do padrão) médio, aquela pertencente à oitava mais central no teclado de um piano, soa uma frequência de 440Hz, padrão adotado neste projeto (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 1975). Note-se, porém, que este não é um padrão amplamente aceito, como demonstra o maestro Franz Nistl em seu *website*, que lista os diversos padrões de afinação adotados por diferentes orquestras ao redor do mundo (NISTL, 2012).

3.1.4 Timbre – fundamental e harmônicas

Timbre é definido pela *Accoustical Society of America* como o atributo que distingue dois sons distintos que possuem a mesma característica de altura e volume, o que depende do espectro de frequências, da amplitude e de características temporais (ASA STANDARDS SECRETARIAT, 1994). Em outras palavras, é o timbre que diferencia uma mesma nota (uma nota de mesma altura) executada por instrumentos diferentes.

Quando se toca determinada nota em um instrumento musical, um amplo espectro de frequências sonoras múltiplas, chamadas harmônicas, é produzido (HALLIDAY; RESNICK, 2008). A primeira e menor de todas, *harmônica fundamental*, é a frequência que está relacionada com a percepção de altura de cada indivíduo e, portanto, define a nota musical sendo executada (PIERCE, 2001). As demais harmônicas, como definido pela ASA (1994), caracterizam o timbre do instrumento de acordo com a sua amplitude e pequenas variações no domínio do tempo, ou não-linearidades.

Timbre já foi definido como "tudo o que não puder ser rotulado como altura ou volume" (MCADAMS; BREGMAN, 1979), o que inclui o envelope temporal. Esse, porém, será abordado separadamente, na subseção seguinte, devido a sua importância para representar diferentes instrumentos em um sintetizador eletrônico, objetivo deste projeto.

McAdams e Bregman (1979) ainda ressaltam que é possível construir o timbre a partir de várias harmônicas relacionadas, sem a presença de uma frequência fundamental, desde que estas harmônicas estejam situadas no espectro dentro da região de frequências que efetivamente contribuam para a formação do timbre. Por exemplo, frequências de 1000, 1250,

1500 e 1750 hertz são harmônicas de uma fundamental de 250 Hz e, se executadas paralelamente, podem ser percebidas como equivalentes em altura a um tom puro de 250 Hz, mesmo que esta fundamental não esteja presente.

Assim, apesar de ser possível identificar uma nota musical apenas a partir da onda sonora pura, que tem a frequência fundamental, são as demais harmônicas e as não-linearidades que tornam o som agradável de ser ouvido, e são essenciais para a síntese de instrumentos reais.

3.1.5 Envelope temporal – ADSR

Uma nota de um piano não soa idêntica desde o momento em que a tecla é pressionada até o momento em que o som cessa. Isso também é válido para a maioria dos instrumentos musicais. Essa variação do som percebido, elemento importante do timbre do instrumento, é chamada de *envelope temporal*. O envelope modela a variação de como o instrumento soa no decorrer da execução da nota em fases que formam o acrônimo *ADSR* – ataque, decaimento, sustentação e repouso – ilustradas na Ilustração 2 (THE EDITORS OF ENCYCLOPÆDIA BRITANNICA, 2014).

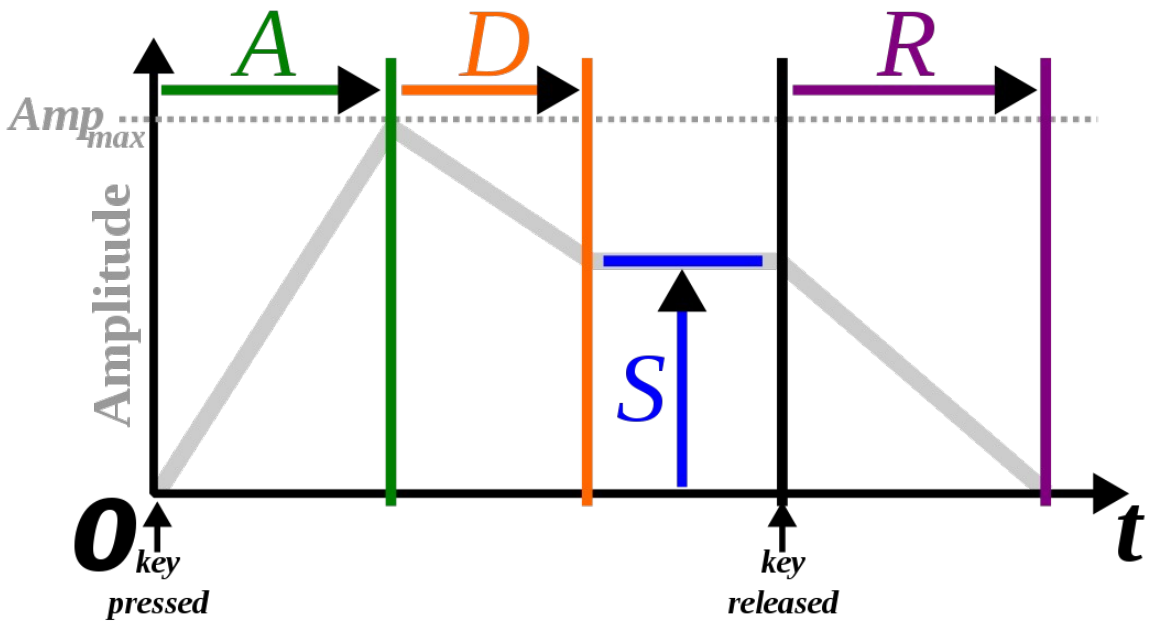


Ilustração 2: Fases ADSR do envelope temporal; extraído de (“Synthesizer”, 2015)

A fase de ataque compreende desde o momento em que a nota é iniciada (tecla pressionada no piano, corda tocada no violão, início do sopro na flauta, etc) até o momento em que o som atinge seu volume máximo. É seguida pela fase de decaimento, que é constituída por um decréscimo na intensidade do som antes de atingir o "estado estacionário" de ressonância, ou sustentação. Por fim, quando o intérprete para de executar a nota, inicia o período de repouso, que leva o som do estado de sustentação até o silêncio (THE EDITORS OF ENCYCLOPÆDIA BRITANNICA, 2014).

A ocorrência e características de cada uma dessas fases dependem das características do instrumento, ou seja, fazem parte do timbre. Alguns instrumentos, por exemplo, quase não apresentam decaimento, enquanto outros possuem um período de repouso abrupto, e outros ainda praticamente não se sustentam, características facilmente perceptíveis ao se ouvir o som produzido pelo instrumento.

3.2 Modelo aditivo de som

Segundo Halliday (2008), em seu clássico livro-texto, ondas sonoras são, genericamente, quaisquer ondas mecânicas longitudinais, ou seja, necessitam de um meio material para se propagarem e apresentam oscilações na mesma direção de propagação. Assim, uma onda sonora pode ser representada matematicamente, tal qual qualquer outro tipo de onda, por uma função periódica em termos de *amplitude* y_m, *frequência* ω e *fase* φ, conforme a Equação 2.

$$y = y_m \cdot sen(\omega t + \varphi) \tag{2}$$

Equação 2: Equação da onda como função periódica (HALLIDAY, 2008)

Ao produzirem-se sons musicais, tais ondas são transferidas para o meio de propagação (geralmente o ar) a partir de uma fonte de vibrações em instrumentos musicais (as cordas de um violão ou o tubo de ar de uma flauta, por exemplo). Embora apresentem não-linearidades, as características mais perceptíveis do resultado sonoro de um instrumento comportam-se linearmente, o que, na prática, significa que o som reproduzido pode ser

representado como um somatório de ondas sonoras, ou seja, um somatório de funções *seno* como as da Equação 2 (PIERCE, 2001). Este é o princípio de *modelos de som aditivos*, em que um som arbitrário é decomposto em múltiplas componentes senoidais (FITZ, 2007).

A Ilustração 3 exemplifica o modelo aditivo de sons por meio da decomposição em senoides do som do miado de um gato. Nela está representada a variação de frequência no domínio do tempo de múltiplas ondas senoidais extraídas do som. Embora tais senoides, se consideradas e executadas separadamente, nada se pareçam com o som modelado, todas elas executadas em simultâneo, ou, matematicamente, a soma delas, forma o som desejado.

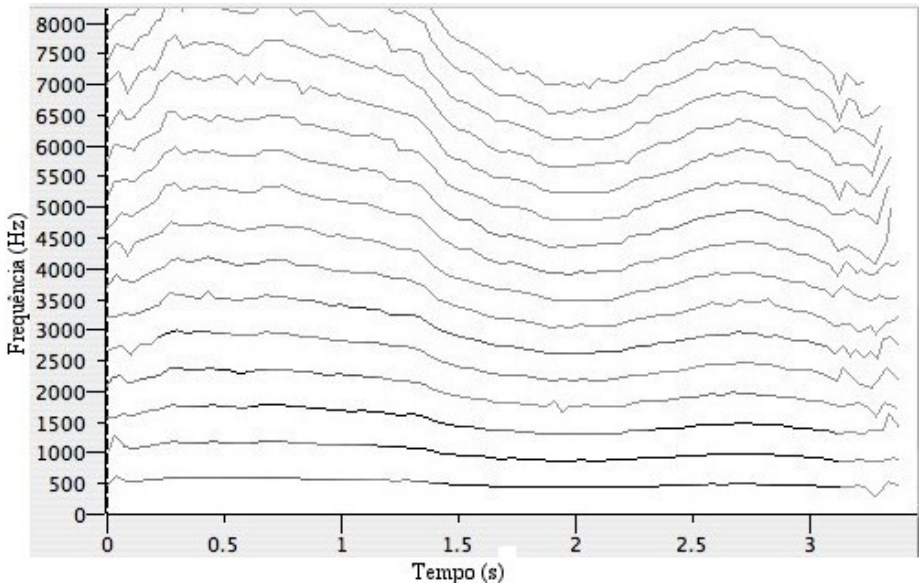


Ilustração 3: Modelo de som aditivo do miado de um gato; extraído de (FITZ, 2007)

3.2.1 Modelo aditivo aplicado a notas musicais – analisador Loris

Embora a representação de sons arbitrários possa ser de interesse para certos sons musicais (como será abordado na subseção 3.3.1), um subconjunto dessa representação é de especial interesse, que é a modelagem de notas musicais. Já foi definido anteriormente o conceito de timbre (3.1.4) a partir da parametrização das harmônicas da nota musical executada. As harmônicas nada mais são do que cada uma das componentes senoidais derivadas do modelo aditivo, parametrizadas pela frequência, amplitude e fase.

O *software* Loris (FITZ; HAKEN, [s.d.]) é um modelador e processador de som capaz de decompor, a partir de uma análise do espectro de frequências de acordo com o modelo

aditivo, notas musicais de timbre bem definido em cada uma de suas harmônicas, extraindo a parametrização de cada onda e sua variação no domínio do tempo.

A partir de interfaces em C, C++ ou Python, o programa recebe como entradas um arquivo de áudio sem compressão contendo a amostra do som a ser analisado e a frequência fundamental esperada dele, obtida a partir da nota sendo executada. O ajuste fino é feito a partir de parâmetros de resolução de frequências, tamanho de janela a ser analisado, número de harmônicas, tolerância de frequências e amplitude. O autor disponibiliza uma explicação de cada parâmetro e do impacto esperado sobre o resultado final (FITZ, 2010), o que varia de acordo com o timbre do instrumento considerado. Para melhor compreensão, os gráficos a da Ilustração 4 exemplificam a saída otimizada do programa após a análise de uma amostra de piano executando a nota Lá médio. Cada gráfico representa a frequência, subamostrada em intervalos de 4 centésimos segundo, de uma das 14 harmônicas obtidas em função do tempo, onde a amplitude é representada pela intensidade da linha, de acordo com a escala da barra à direita do respectivo gráfico.

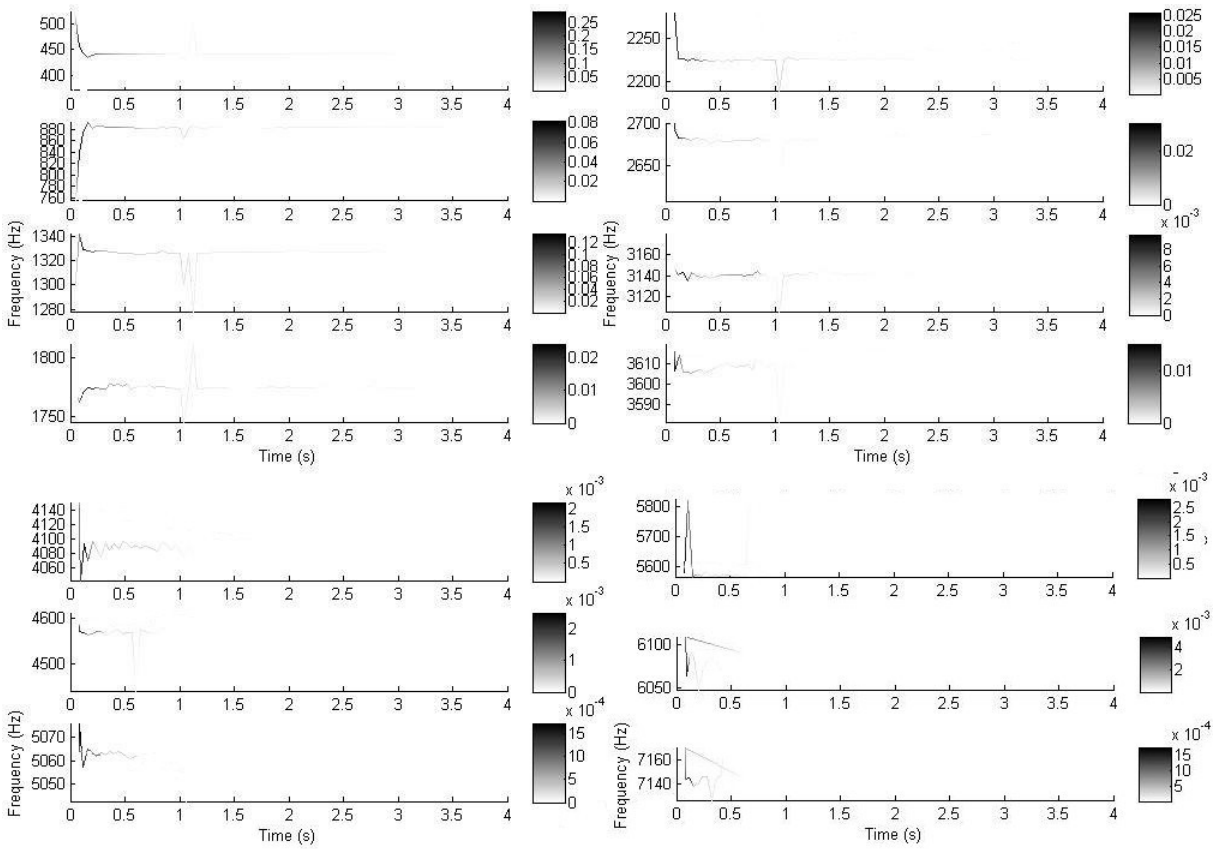


Ilustração 4: Harmônicas obtidas pelo Loris de uma amostra de piano

3.3 MIDI e General MIDI

Interface Digital de Instrumento Musical (*Musical Instrument Digital Interface – MIDI*) é um protocolo de comunicação mantido pela Associação de Fabricantes MIDI (*MIDI Manufacturers Association – MMA*), cuja especificação original data do ano de 1982 e recebe contínuas revisões e atualizações até os dias atuais (MIDI MANUFACTURES ASSOCIATION, 2015a). O padrão MIDI é utilizado para traduzir a interação do intérprete com um instrumento musical na forma de mensagens e então transmitir essas mensagens para outros dispositivos compatíveis com o padrão, onde elas serão utilizadas para controlar geradores de som (HUBER, 2007).

A especificação MIDI, porém, se limita a um protocolo de comunicação, ou seja, MIDI não é capaz de comunicar áudio ou criar sons, apenas transmitir para outro dispositivo que tipo de sons espera-se gerar, sob suas próprias especificações (HUBER, 2007). Por este motivo, foi lançado como uma extensão da especificação inicial MIDI o padrão *General MIDI* (*GM*), que define uma série de parâmetros que instrumentos e sintetizadores MIDI devem seguir para que os sons produzidos por fabricantes distintos soem similares (RUSS, 2012).

O padrão GM possui três versões distintas, o *GM1*, que é a versão criada inicialmente, mais simples e ainda popular em dispositivos de menor custo, que foi sucedido pelo *GM2* que adiciona uma série de novas propriedades para acompanhar o desenvolvimento da tecnologia como um todo – note-se que, conforme observado por HUBER (2007), a especificação MIDI, assim como o GM1, datam de uma época de que grande parte da tecnologia criada já foi esquecida – além de aumentar o número de sons e controles possíveis. O *General MIDI Lite* (GML), por sua vez, é uma versão ainda mais reduzida do GM1 concebida para dispositivos que não são capazes de suportar o padrão mais completo (MIDI MANUFACTURES ASSOCIATION, 2015b).

Esta subseção descreve um subconjunto de características do padrão GM1 consideradas no desenvolvimento deste projeto.

3.3.1 Canais

O padrão define a existência de dezesseis canais distintos, dos quais quinze, chamados “melódicos”, são alocados a timbres de instrumentos, ou seja, o sistema é capaz de representar até quinze instrumentos distintos simultaneamente. É dentro dos canais melódicos que as notas musicais são executadas, em número variável (é comum avaliar a qualidade de um sintetizador baseado no número de notas em simultâneo que ele é capaz de executar). Os canais são associados a instrumentos dinamicamente e ainda podem receber vários parâmetros de controle, que determinam as características do som final gerado, como tempos distintos de ataque e repouso, filtros de timbre e *vibrattos* aplicados (MIDI MANUFACTURES ASSOCIATION, 2015b).

O canal restante, especificamente o canal de número 10, é reservado para a execução de sons distintos de percussão. Nele, as notas musicais são associadas a sons únicos de instrumentos distintos, ao invés de a diversas alturas de um mesmo timbre como nos canais melódicos, conforme a Ilustração 5 (MIDI MANUFACTURES ASSOCIATION, 2015b). Assim, o som a ser representado nesse canal é um som arbitrário, que não segue a composição em harmônicas, demonstrada anteriormente, utilizada como base de operação do *software* Loris e, por esse motivo, não será considerado neste projeto.

3.3.2 Notas

São suportadas pelo padrão MIDI 128 notas musicais diferentes, para as quais cada fabricante de dispositivo determina suas relações quanto a altura e escalas. *General MIDI*, no entanto, associa o Lá Médio, segundo o padrão de afinação A440 (3.1.3), à nota número 69, adotando a escala cromática (GERVAIS, 2005). Assim, é possível determinar, de acordo com as definições apresentadas na subseção 3.1, a lista de possíveis notas MIDI e suas respectivas frequências de harmônica fundamental (Apêndice A).

A cada nota em execução é também atribuído um parâmetro de “velocidade”. A velocidade de uma determinada nota é um número inteiro no intervalo de 0 a 127 e pode ser entendida como a velocidade do ar soprado no tubo de uma flauta ou a força aplicada sobre a

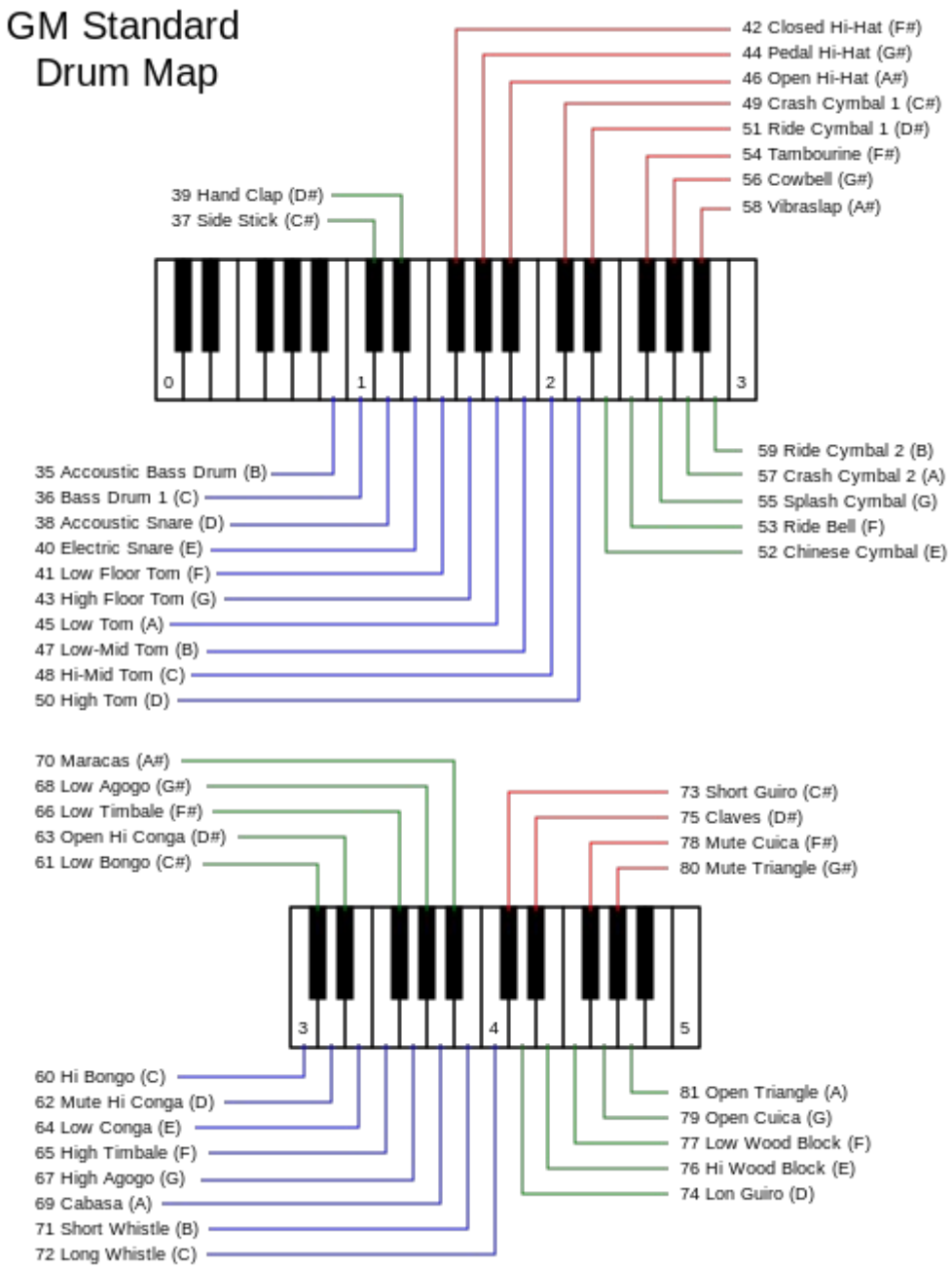


Ilustração 5: Mapeamento de sons do canal MIDI 10; extraído de (“MIDI”, 2015)

tecla de um piano, por exemplo. É evidente a relação entre a velocidade e o volume final do som gerado, e, logo, a amplitude das harmônicas, porém, o padrão MIDI não especifica como ela deve ser interpretada exatamente. Fabricantes diferentes, e até produtos diferentes, utilizam diversas implementações para este parâmetro (DANNENBERG, 2006). No escopo

deste projeto, velocidade será interpretada como uma relação direta e linear com a amplitude de cada uma das harmônicas geradas para o som de cada instrumento.

3.3.3 Programas e Instrumentos

O padrão MIDI também define 128 programas que são associados a configurações de som diferentes, conforme determinado pelo usuário ou pelo fabricante do dispositivo (HUBER, 2007). O General MIDI, por sua vez, associa diretamente cada um desses programas com um instrumento, que pode ser um instrumento musical propriamente dito ou um gerador de sons eletrônicos (GERVAIS, 2005). Os instrumentos foram assim divididos em categorias de semelhantes, conforme a Tabela 1 – a lista completa de instrumentos está disponível no Anexo A.

Tabela 1: Categorias de instrumentos MIDI

Programa	Categoria	Programa	Categoria
1—8	Pianos	65—72	Palhetas
9—16	Percussão cromática	73—80	Flautas
17—24	Órgãos	81—88	Solos sintéticos
25—32	Guitarras	89—96	Fundos sintéticos
33—40	Baixos	97—104	Efeitos sonoros sintéticos
41—48	Cordas	105—112	Instrumentos étnicos
49—56	Orquestra sinfônica	113—120	Percussão não-cromática
57—64	Metais	121—128	Efeitos sonoros

Fonte: (GERVAIS, 2005); adaptado

Cada canal melódico MIDI está associado a um instrumento de cada vez, cujo timbre será utilizado para todas as notas tocadas no respectivo canal. Assim, um sintetizador MIDI é capaz de executar até 15 instrumentos simultaneamente, além do acompanhamento de percussão no canal 10. Nesse canal apenas, não existe um único timbre característico, e portanto o programa de instrumento a ele associado é ignorado (HUBER, 2007).

3.3.4 Mensagens

A comunicação MIDI é feita por meio do envio de mensagens. Essas mensagens são transmitidas através de um protocolo de comunicação *serial* em blocos compostos por 1 *bit* de inicialização em nível lógico baixo (*start bit*), 8 *bits* de dados, sem pareamento, e 1 *bit* de parada em nível lógico alto (*stop bit*). Assume-se, assim, a linha em nível lógico alto quando não há dados sendo transmitidos. A taxa de transmissão definida no padrão é de 31 250 baud/s[1] (HUBER, 2007; SWIFT, [s.d.]).

As mensagens MIDI possuem um número variável de *bytes*, dos quais o primeiro, de *status*, é demarcado por um *bit* mais significativo em nível alto e determina o tipo de mensagem e, quando aplicável, o canal MIDI de destino da mensagem. Os demais *bytes* são de dados, ou seja, transmitem os parâmetros e informações adicionais relacionados ao tipo de mensagem. Os *bytes* de dados são identificados por um *bit* mais significativo em nível baixo (HUBER, 2007).

Das várias mensagens MIDI previstas pelo padrão, três são essenciais e implementadas neste projeto, conforme a Tabela 2 (as mensagens estão representadas em notação binária). As demais são mensagens para controle de parâmetros mais avançados, ajustes do dispositivo em si ou reservadas para uso do fabricante, e neste escopo são desconsideradas. O Anexo B contém a lista completa de possíveis mensagens MIDI e seus significados.

Tabela 2: Mensagens MIDI consideradas no projeto

Status	Dados	Mensagem
`1100cccc`	`0ppppppp`	*Program Change*: associa o instrumento *p* ao canal *c*.
`1001cccc`	`0kkkkkkk` `0vvvvvvv`	*Note On*: ativa a nota *k* com velocidade *v* no canal *c*. Quando a velocidade *v* é zero, a mensagem é interpretada como *Note Off*.
`1000cccc`	`0kkkkkkk` `0vvvvvvv`	*Note Off*: desativa a nota *k* com velocidade *v* no canal *c* [2].

Fonte: (MIDI MANUFACTURES ASSOCIATION, 2015c); adaptado

1 Huber (2007, p15) informa uma "velocidade" de 31 250 bit/s, no entanto, considerando-se o público-alvo da obra e a existência de bits de inicialização e parada, optou-se por utilizar a convenção baud/s.
2 Como a velocidade foi definida como uma relação linear com a amplitude, não há sentido no parâmetro de velocidade para o evento de *Note Off*.

4 METODOLOGIA

Esta seção contempla todo o procedimento de projeto, desenvolvimento e teste do sintetizador de áudio proposto. O sistema como um todo é composto por:

- O sintetizador, responsável por receber as mensagens MIDI através de uma interface de comunicação *serial*, conforme descrito em 3.3.4, e por gerar uma forma de onda de saída que, após ser submetida a um amplificador de áudio, possa ser executada diretamente em um alto-falante. A Ilustração 6 contém o diagrama de fluxo de dados do hardware desenvolvido.
- Um dispositivo microcontrolador auxiliar para facilitar a transferência das mensagens MIDI de um computador para o sintetizador, através de uma porta USB.
- Um computador PC equipado com múltiplas ferramentas de *software* para desenvolvimento e posterior teste do sistema.

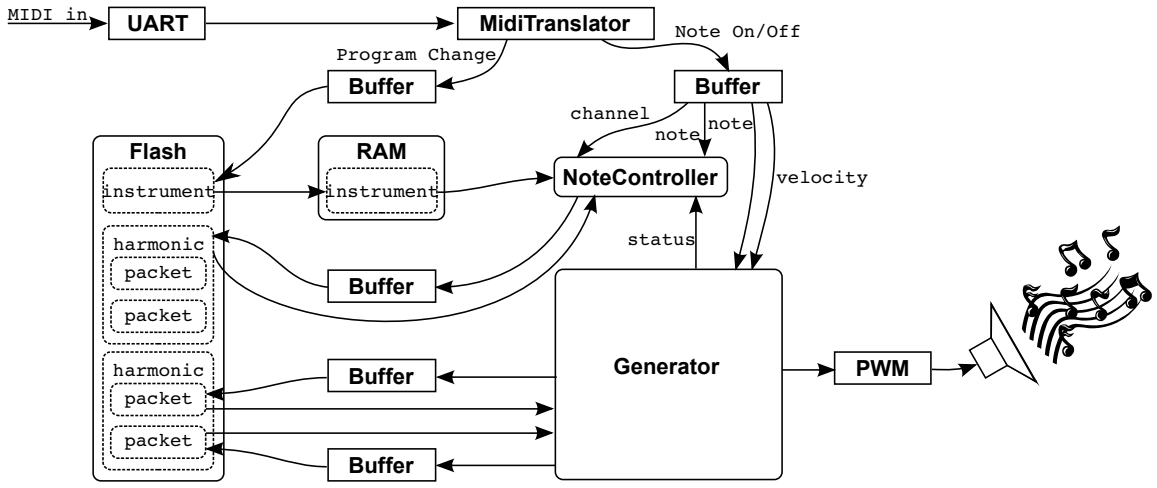


Ilustração 6: Diagrama de fluxo de dados do sintetizador

A primeira subseção a seguir apresenta cada uma das ferramentas utilizadas. A seguir, define-se funcionalmente o módulo sintetizador, com base no comportamento esperado, com a apresentação de diagramas de fluxo de dados, de blocos e de máquinas de estados do sistema. Por fim, a subseção 4.3 descreve o desenvolvimento do *software* auxiliar para amostragem de instrumentos a serem incluídos no sintetizador.

4.1 Ferramentas utilizadas

Esta subseção apresenta a série de ferramentas utilizadas para o desenvolvimento desde projeto. As duas primeiras partes contêm as características mais relevantes das placas de desenvolvimento utilizadas e ferramentas de software utilizadas como apoio. Logo em seguida introduz-se a suíte de desenvolvimento principal, a linguagem de descrição de *hardware* utilizada e, por fim, são apresentadas e relacionadas as ferramentas de *software* utilizadas para o teste final do sistema.

4.1.1 Terasic Altera DE0

A placa de desenvolvimento *Terasic Altera DE0* é uma placa baseada em uma FPGA *Altera Cyclone III EP3C16F484*. A placa contém um kit básico de entrada e saída e dispositivos de memória para suporte ao desenvolvimento. Do *hardware* da placa, destaca-se a utilização dos pinos de entrada e saída de uso geral, memória FLASH, visores de sete segmentos, LEDs e um botão. A Ilustração 7 representa a placa de desenvolvimento e o *hardware* nela contido[3] (TERASIC, 2011).

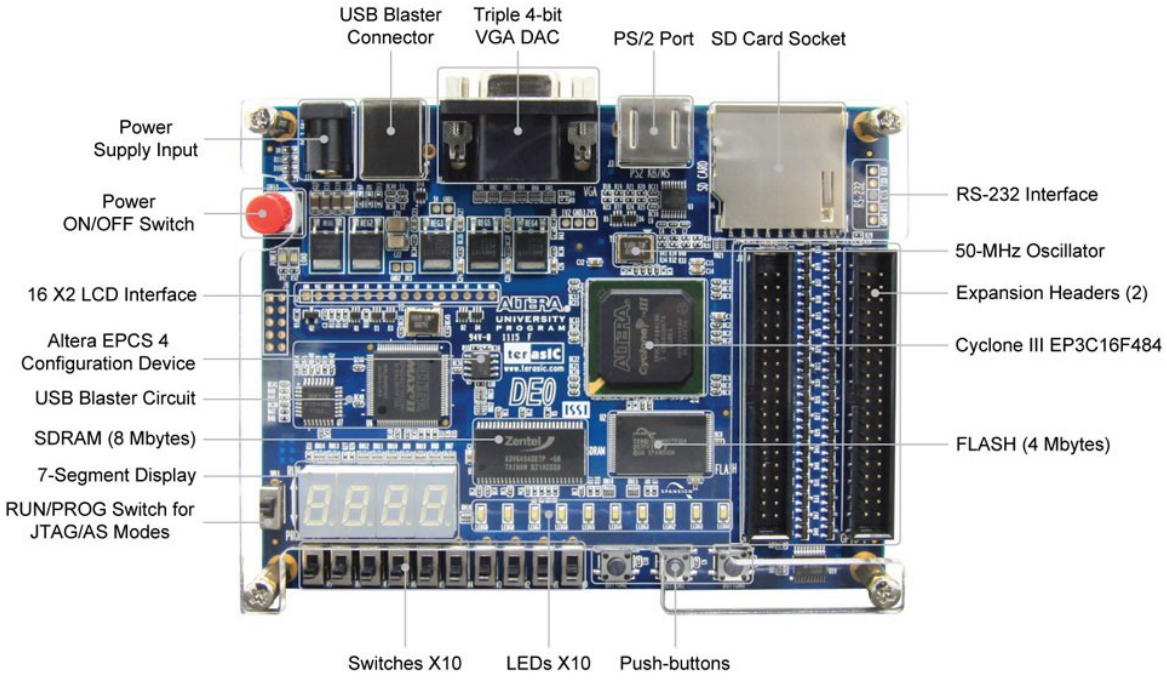


Ilustração 7: Layout da placa de desenvolvimento Terasic Altera DE0; extraído de (TERASIC, [s.d.])

O fabricante disponibiliza o *software DE0 Control Panel*, que programa a FPGA com um teste das principais funções da placa. Dentre elas, o programa permite gravar a memória externa Flash a partir de um arquivo binário. Essa ferramenta é utilizada para a gravação da amostragem dos instrumentos musicais.

Em relação à FPGA, destacam-se os 56 blocos de memória M9K, oferecendo até 9 Kbits de espaço cada. Tais blocos podem ser configurados como módulos de memória com duas portas de leitura e escrita completamente independentes, possibilidade altamente explorada no sistema proposto (ALTERA CORPORATION, 2008).

4.1.2 STMicroelectronics STM32F4-Discovery e Micropython

A placa de desenvolvimento *STM32F4-Discovery* é baseada na linha de microcontroladores *SMT32F407/417* da *STMicroelectronics*, com core *ARM Cortex-M4* (Ilustração 8) (STMICROELECTRONICS, [s.d.]). A placa é utilizada como uma ponte entre a *DE0* (4.1.1) e um computador, através da emulação de uma porta serial comum sobre a interface USB e a utilização de uma interface UART nos pinos de entrada e saída.

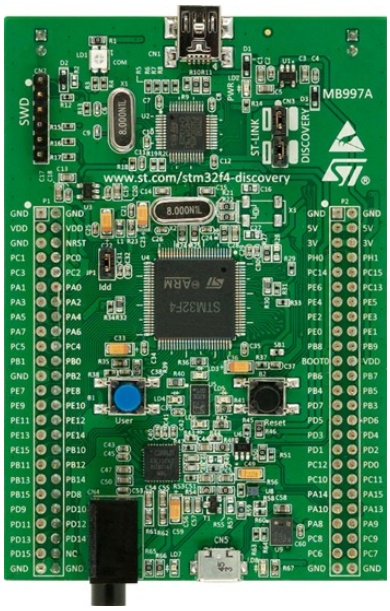


Ilustração 8: SMT32F4-Discovery; extraído de (STMICROELECTRONICS, [s.d.])

MicroPython é uma implementação da linguagem *Python* otimizada para microcontroladores. Adicionalmente a algumas bibliotecas comuns da linguagem, ela define o módulo *pyb*: um conjunto de classes e funções de baixo nível para controle do *hardware* da placa de desenvolvimento com a qual o projeto surgiu, a *Pyboard*. A *Pyboard*, por sua vez, é baseada em um microcontrolador da mesma família do utilizado pela placa *Discovery* e assim o desenvolvedor disponibiliza uma versão do projeto para ela (GEORGE, 2014).

4.1.3 Altera Quartus II e Modelsim-Altera

Quartus II é a suíte de desenvolvimento da *Altera*. Para a FPGA da placa de desenvolvimento utilizada, a *Cyclone III*, foi utilizada a versão 13.1 do *software*, que é a mais recente que a suporta. Juntamente com o *Quartus*, o fabricante disponibiliza uma versão customizada do *ModelSim*, um *software* de simulação de linguagens de descrição de *hardware*.

Faz parte da suíte de desenvolvimento um conjunto de propriedade intelectual da *Altera* para implementação de funções prontas e inferência direta de módulos do dispositivo. Dentre estas, destacam-se as funções para inferência do módulo de PLL e das memórias M9K do chip, utilizadas como memória interna (4.2.4).

4.1.4 Linguagem de descrição de hardware *SystemVerilog*

SystemVerilog é uma linguagem de descrição e verificação de *hardware* considerada a evolução da linguagem *Verilog*. É uma linguagem de sintaxe semelhante à linguagem C, fracamente tipada e que implementa muitas extensões para auxílio e parametrização da descrição do *hardware*. A linguagem dispensa a instanciação de bibliotecas e, praticamente, todos os sinais utilizados no *hardware* descrito podem ser do mesmo tipo, que aceita operações lógicas e aritméticas. Assim, o resultado final descrito na linguagem é menos prolixo que quando descrito utilizando *Verilog* e *VHDL*, além da curva de aprendizado ser menor. Por esses motivos, foi escolhida como a linguagem para descrição do sistema.

4.1.5 *Software* de apoio

Softwares de apoio são os programas de computador utilizados principalmente durante a verificação final do sistema, como, por exemplo, para edição das amostras de áudio antes de sua gravação em flash, para um fácil controle do áudio a ser executado e para interconexão entre todos os aplicativos usados.

Jack Audio Connection Kit (DAVIS, [s.d.]) é um servidor de som profissional que é o padrão *de facto* para aplicações baseadas em sistema operacional Linux. O Jack permite interconexões entre diversos outros programas por meio de portas de interface MIDI e de áudio ALSA[4]. Em conjunto com o *a2jmidid* (ARNAUDOV, [s.d.]), as portas MIDI gerenciadas pelo ALSA podem ser exportadas como portas MIDI gerenciadas pelo Jack, propiciando ainda mais possíveis interconexões.

Qsynth (CAPELA; BOWN; CANNAM, [s.d.]) é uma interface gráfica de controle para o sintetizador MIDI *FluidSynth* (FLUIDSYNTH DEVELOPERS, [s.d.]). A interface proporciona o controle de diversos parâmetros de síntese, bem como visualização dos canais sendo utilizados e instrumentos alocados. O sintetizador permite a adição de conjuntos de instrumentos a partir de inúmeras amostras disponíveis na internet. Portas ALSA MIDI ou Jack MIDI podem ser criadas para conexão do *software* com outras aplicações.

Audacity (AUDACITY DEVELOPMENT TEAM, [s.d.]) é um editor de áudio livre e multiplataforma. Ele permite várias modificações em arquivos de áudio, dentre elas a aplicação de filtros em frequência, alterações de amplitude, tempo, remoção de ruídos e de períodos de silêncio, que foram as principais ferramentas utilizadas nesse projeto. O Audacity também computa facilmente a Transformada Rápida de Fourier do áudio sendo analisado, facilitando o trabalho de identificação da nota executada, a partir do espectro de frequências.

VMPK (LOPEZ-CABANILLAS, [s.d.]) é um teclado MIDI virtual que recebe e gera eventos MIDI. Quando recebendo eventos, o *software* exibe graficamente a representação das notas sendo executadas em um teclado de piano. Também permite o controle desse teclado usando um mouse ou teclado, além da definição de parâmetros como velocidade das notas, instrumento, além de outros controles MIDI que não são utilizados no projeto. O *VMPK* cria

4 ALSA é o componente do *kernel* Linux que gerencia as interfaces de áudio e MIDI.

portas ALSA MIDI de entrada e saída que podem ser conectadas virtualmente a outros aplicativos ou a portas MIDI físicas.

MuseScore (BONTE; FROMENT; SCHWEER, [s.d.]) é um editor de partituras *WYSIWYG*[5] livre e multiplataforma. O programa lê arquivos MIDI ou de formato próprio e exibe as partituras graficamente na tela, permitindo a sua edição detalhada. O *MuseScore* possui um sintetizador integrado, porém também permite a conexão com outros aplicativos por meio de portas MIDI exportadas para o servidor Jack.

Por fim, *The Hairless MIDI to Serial Bridge* (GRATTON, [s.d.]) é um projeto de código aberto multiplataforma que cria uma ponte entre uma das portas ALSA MIDI disponíveis no sistema com uma porta de comunicação *serial*.

4.2 Hardware do módulo sintetizador

O *hardware* do sistema foi dividido em diversos módulos, com função específica. Alguns dos módulos são comuns e utilizados várias vezes no sistema principal e dentro de outros módulos. O objetivo desta subseção é apresentar cada um dos módulos e descrever o seu funcionamento e interações com os demais.

O principal dos módulos do sistema, chamado sintetizador, é responsável pelas conexões entre os demais e interface com o meio exterior, além de algumas operações mais simples que dispensam a necessidade de um módulo específico.

O módulo sintetizador é o módulo de mais alto nível do sistema, portanto, é ele que define a interface dos módulos internos entre si e com o mundo externo. Da placa *DE0*, são utilizados o circuito de *clock*, o dispositivo de memória *flash*, três pinos de entrada e saída de uso geral (um para recepção das mensagens MIDI e dois para saída de áudio), três visores de 7 segmentos, LEDs e um botão, utilizado como *reset* síncrono.

As mensagens MIDI são recebidas pelo módulo UART, responsável pela conversão de serial para paralelo. Cada uma das palavras de 8 *bits* recebidas é enviada ao módulo de tradução MIDI (*MidiTranslator* - 4.2.3), onde uma máquina de estados classifica-la-á de acordo com a mensagem MIDI da qual ela faz parte, conforme definido na subseção 3.3.4. As

5 "*What You See Is What You Get"*: a partitura exibida em tela é idêntica à partitura que seria impressa.

mensagens de *Program Change* e *Note On/Off* recebidas, únicas consideradas no sistema, são armazenadas em *buffers* circulares (4.2.4) independentes para posterior tratamento.

Os 128 primeiros endereços da memória *flash* armazenam ponteiros para o endereço de memória onde as amostras de cada instrumento MIDI estarão armazenadas. Assim, sempre que houver mensagens de *Program Change* em *buffer*, uma operação de leitura na *flash* será desencadeada, o ponteiro salvo na *flash* é recuperado e armazenado em uma memória RAM interna de 16 posições, uma para cada canal MIDI.

No caso de mensagens de *Note On/Off*, a sua existência em *buffer* desencadeia a operação da máquina de estados do módulo de controle de notas (4.2.2). Para um *Note On*, os parâmetros definidos pela mensagem MIDI – nota e velocidade – são definidos como entradas para o módulo gerador (4.2.1) e o módulo de controle começa a fazer leituras na *flash* buscando as primeiras amostras de cada uma das harmônicas do instrumento alocado ao canal alvo da mensagem, de acordo com o ponteiro armazenado durante a mensagem de *Program Change*, controlando o início da geração do som, em múltiplas senoides independentes.

Já no caso de um *Note Off*, o módulo de controle busca no gerador a informação do número de canal e nota a que cada senoide sendo gerada pertence, e as desativa uma a uma.

Após o recebimento da amostra inicial de cada harmônica, o módulo gerador passa a ler sequencialmente as amostras seguintes na *flash*, de acordo com o envelope temporal do instrumento sintetizado. O módulo gerador tem como saída a representação instantânea da soma de todas as senoides geradas. Essa representação é submetida a um modulador por largura de pulso (PWM), que atua como um conversor de digital para analógico (4.2.4). A saída do PWM é enviada diretamente aos pinos de saída de uso comum da placa.

O Apêndice B contém o diagrama de blocos mais detalhado do módulo sintetizador, com uma tabela com cada um dos sinais e barramentos, sua utilização e a qual pino da FPGA estão conectados.

É também definida uma rede de *reset* síncrono, como uma disjunção lógica do sinal de um botão externo e das saídas *full* dos dois *buffers* de entrada (conforme 4.2.4), que recebem as mensagens MIDI de *Program Change* e *Note On/Off*, com o objetivo de recuperar o sistema de um estado de exceção. Todos os módulos sequenciais do sistema possuem uma entrada de *reset* síncrono conectada à rede.

Os *clocks* do sistema são obtidos por um PLL conectado ao *clock* externo da placa. O *clock* principal, de 45 Mhz, relaciona-se com a frequência de amostragem do áudio sintetizado conforme descrito no detalhamento do módulo gerador (4.2.1). Um sinal de *clock* negado é gerado, também, para sincronização de operações que devem acontecer entre duas bordas de subida do *clock* principal. Esse sinal de *clock* negado deve ser obtido a partir do PLL ao invés de pela simples negação do *clock* principal para que a ferramenta de síntese reconheça a rede como um *clock* e efetue as otimizações necessárias. Por fim, as características dos módulos gerador e de saída exigem que a frequência de operação do segundo seja maior que a do primeiro, conforme detalhado nas subseções dos respectivos módulos (4.2.1 e 4.2.4), e para isso foi gerado um sinal de 360 Mhz.

4.2.1 Módulo gerador

O módulo gerador é responsável pela geração de múltiplas senoides em simultâneo que modelam o som a ser executado. O mecanismo de geração de múltiplas senoides é derivado a partir da multiplexação temporal do mecanismo de geração de uma única senoide com amplitude e frequência variáveis. É também o módulo gerador que controla temporalmente a variação dos parâmetros de frequência e amplitude, de acordo com o envelope ADSR (3.1.5) do instrumento modelado. Assim, o módulo gerador é o principal para a operação do sistema como um todo, sendo que os demais se adaptam às definições dele, e por esse motivo é o primeiro a ser detalhado.

A geração da onda seno é derivada a partir da característica periódica da função, bem como da aparência “simétrica” do gráfico final. Tal “simetria” pode ser demonstrada matematicamente pelas equações a seguir (Equação 3), que relacionam os valores de seno do período inteiro com os valores do primeiro quarto de período.

$$sen(x) = \left\{ \begin{array}{ccc} sen(y) & ; & 0° \le x < 90° \\ sen(90° - y) & ; & 90° \le x < 180° \\ -sen(y) & ; & 180° \le x < 270° \\ -sen(90° - y) & ; & 270° \le x < 360° \end{array} \right\} \; ; \; 0° \le y \le 90°$$

Equação 3: Relação entre os quartos de período da função seno

Assim, nota-se que um período completo de senoide pode ser obtido a partir de valores de seno para o primeiro quarto de período, apenas. Assim, foram extraídos, discretizados em intervalos igualmente espaçados e representados como números inteiros os valores de seno para um quarto de período. O resultado, exemplificado na Tabela 3 e obtido pela Equação 4, é armazenado em um módulo de memória interna (4.2.4). O exemplo utiliza uma memória de 4 *bits* de endereço e dado, portanto a amplitude máxima da onda é 15 e o número de intervalos igualmente espaçados é 16, entre 0º e 90º ($\pi/2$ rad). Devido aos intervalos de discretização serem igualmente espaçados, existe uma relação linear entre o endereço de memória e o ângulo – no caso do exemplo, um incremento de 5,625° entre endereços contíguos.

$$Valor = round\left(15 \cdot sen\left(\frac{Endereço \cdot \pi}{2 \cdot 16}\right)\right) \quad (4)$$

Equação 4: Obtenção dos valores de seno em memória

Tabela 3: Exemplo de valores de seno armazenados em memória

Endereço	Valor	Endereço	Valor	Endereço	Valor	Endereço	Valor
0	0	4	6	8	11	12	14
1	1	5	7	9	12	13	14
2	3	6	8	10	12	14	15
3	4	7	10	11	13	15	15

A leitura sequencial dos valores de seno armazenados produz a forma de onda de um quarto de período de senoide. Com base nas relações da Equação 3, definiu-se um registrador de largura dois *bits* maior que o tamanho do endereço de memória. Assim, ao incrementar-se continuamente o valor do registrador, os dois *bits* mais significativos determinam o quarto de

período do ângulo e os demais *bits* menos significativos determinam o ângulo em si. O resultado esperado está representado na Ilustração 9, onde as abcissas são a representação de ângulo, os marcadores representam o valor lido da memória e a linha o valor obtido após aplicada a relação da Equação 3. No gráfico, pode ser observado que, mesmo utilizando larguras de endereço e dado de apenas quatro *bits*, a forma de onda final já em muito se parece, visualmente, com uma senoide contínua. No *hardware* descrito, endereço e dado têm largura de 11 *bits*, valor que foi escolhido experimentalmente para possibilitar uma boa parametrização da frequência e, ao mesmo tempo, uma boa resolução do sinal.

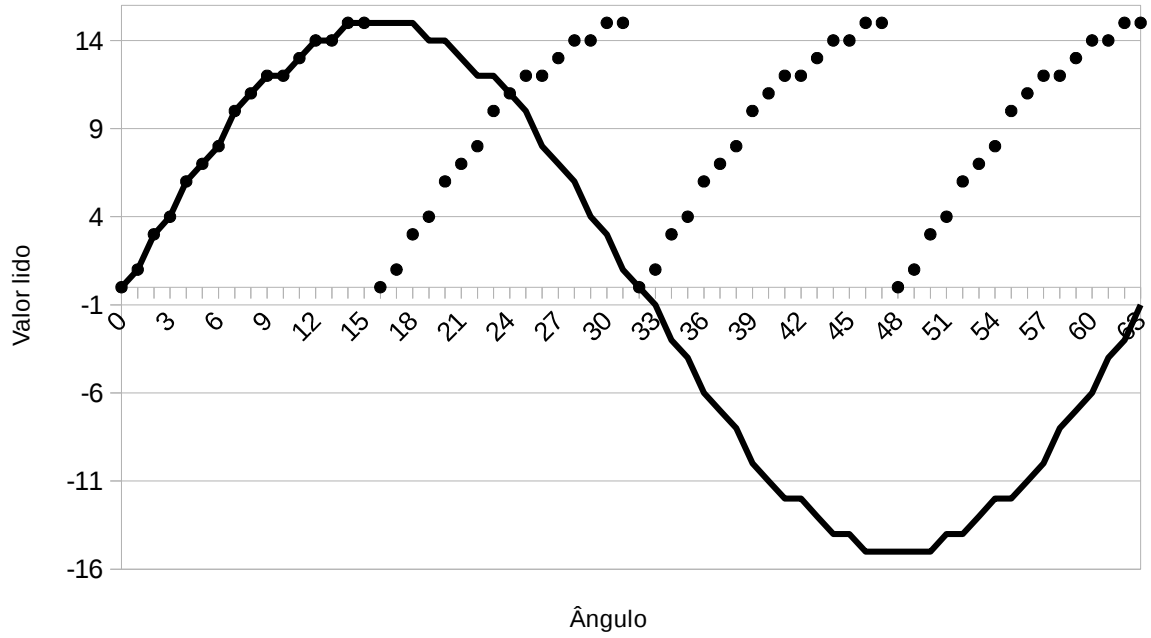


Ilustração 9: Valores de seno lidos da memória

O motivo é que o parâmetro de frequência do sinal pode ser controlado linearmente por meio do incremento do endereço de memória, ou ângulo: se a cada instante de tempo, ou período de amostragem, o endereço for incrementado duas posições, ao invés de uma, a frequência final da senoide será o dobro da anterior. Claramente, quanto maior for a frequência de saída desejada, mantendo-se fixa a taxa de amostragem, pior será a resolução da onda de saída. Para implementação do parâmetro de frequência, o registrador que armazena o ângulo foi estendido em 8 *bits*, e aceita incrementos de até 20 *bits*. Assim, os 2 *bits* mais significativos representam o quarto de período de ângulo, os 11 subsequentes são o endereço a ser lido em memória e os 8 restantes são uma indicação de precisão, conforme mostra a

Ilustração 10. A largura do dado de frequência 1 *bit* menor que a largura do ângulo também reforça o critério do Teorema de Nyquist – desta forma a frequência do sinal amostrado nunca será maior que a metade da frequência de amostragem.

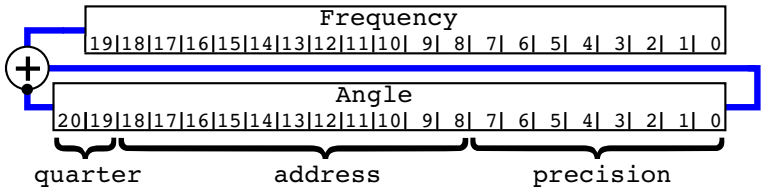


Ilustração 10: Parâmetro de frequência

O parâmetro de amplitude, por outro lado, é uma constante de multiplicação para o valor lido de memória. Na implementação, a amplitude é um número de 8 *bits* e representa uma fração do valor de seno armazenado. Um incremento de 1 no valor de amplitude, portanto, é um incremento de $\frac{1}{256} = 0{,}390625\,\%$ na amplitude, a partir do valor 0. O valor 255, porém, associa-se com 1 (100%) ao invés de com $\frac{255}{256} = 99{,}609375\,\%$. Acredita-se, porém, que a diferença ocasionada para os valores mais altos de amplitude seja imperceptível.

O diagrama de blocos da Ilustração 11 demonstra o *hardware* completo para geração de uma onda seno com amplitude e frequência variáveis. Os parâmetros são registrados a cada variação (eles se mantêm durante vários ciclos de operação do sistema) e a saída do circuito é a representação binária do sinal desejado. A malha de *clock* está omissa no diagrama, entretanto, pode-se perceber que a frequência do *clock* desejado corresponde à taxa de amostragem do sinal final.

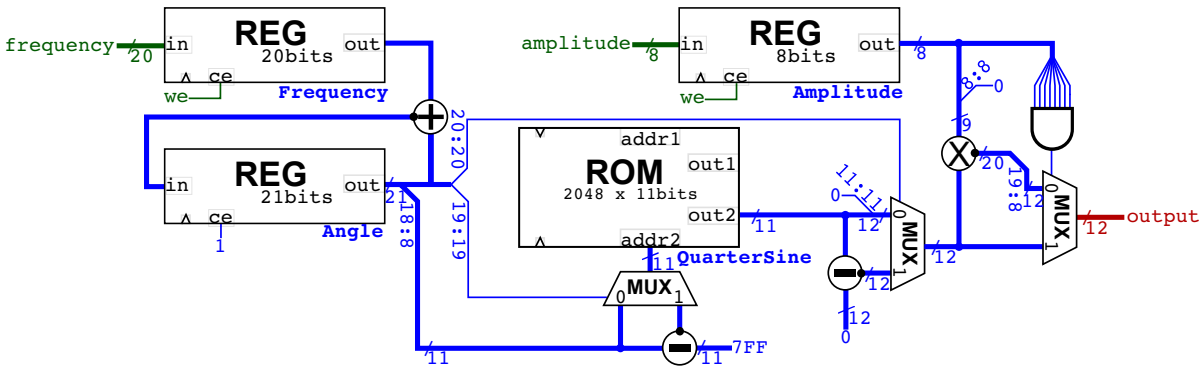


Ilustração 11: Diagrama de blocos de um gerador de onda seno

O *hardware* proposto baseia-se no fato de que o *bit* mais significativo do ângulo, quando em nível lógico alto, representa univocamente um valor de seno negativo, enquanto o segundo *bit* mais significativo representa um ângulo no segundo ou quarto quadrantes. Assim, seguindo o exemplo anterior, os valores lidos da memória, marcadores do gráfico da Ilustração 9, são ajustados segundo a Equação 3 pelos dois multiplexadores mais à esquerda, obtendo como resultado os valores representados pela linha no gráfico.

É importante salientar que, como estão armazenados apenas os valores positivos de seno, o barramento deve ser estendido em 1 *bit* para que os valores negativos também sejam representados. Além disso, o valor de amplitude, que é sempre positivo (uma amplitude negativa representaria uma mudança na fase no sinal, que será adicionada como parâmetro adicional em etapas posteriores) também deve ser estendido, para que a multiplicação de números com sinal retorne o resultado esperado.

A geração de uma única senoide, entretanto, não é o suficiente para simulação de instrumentos musicais. Várias são necessárias, e, quantas mais forem geradas, maiores serão a qualidade do som sintetizado e o número de notas em simultâneo que ele é capaz de executar.

A primeira concepção para geração de múltiplas senoides foi a paralelização do hardware apresentado e a posterior soma de todos os sinais de saída. É possível logo perceber que essa alternativa aumenta linearmente a área do circuito gerado e logo excederia o número de elementos lógicos, aritméticos e de memória da FPGA. Por exemplo, a *Cyclone III* presente na placa *DE0* possui 56 dispositivos multiplicadores de 18x18 *bits*, limitando a 56 entidades a paralelização. Mesmo com a possibilidade de se utilizar os dispositivos de memória M9K para operações de multiplicação, isso aumentaria o número de dispositivos multiplicadores em paralelo para 112, o que não é o suficiente, principalmente considerando-se que o uso de multiplicadores não se limita a um por senoide – o parâmetro MIDI de velocidade também desencadeia uma operação de multiplicação, assim como a frequência final do sinal pode ser definida como a multiplicação de uma frequência amostrada por uma constante de adequação à altura da nota desejada (ALTERA CORPORATION, 2008).

Além disso, seria um desperdício no aspecto temporal. A taxa de amostragem *de facto* para aplicações de áudio digital não profissional, tipicamente 44,1 Khz, é mais de 1000 vezes menor que a frequência do *clock* externo da *DE0* (TERASIC, 2011) e mais de 6000 vezes menor que a frequência máxima de operação dos multiplicadores e dispositivos M9K da

Cyclone III utilizada (ALTERA CORPORATION, 2008). Assim, a simples paralelização do sistema de gerador único não é uma alternativa viável.

Para resolver o problema, propôs-se uma expansão do gerador único, utilizando multiplexação temporal: os registradores que armazenam o valor de frequência, ângulo e amplitude são substituídos por memórias, onde cada endereço de memória representa o valor do parâmetro em um oscilador independente[6]. Incluíram-se registradores auxiliares para controlar o "endereço" do oscilador ativo em cada momento para cada estágio do fluxo, endereço esse que será incrementado a cada ciclo de *clock*. Assim, o módulo gerador é composto por múltiplos osciladores secundários, cada um lendo os parâmetros de geração de sua senoide a partir de um endereço de memória interna. Os sinais de saída são acumulados e, a cada fluxo completo do sistema, o valor resultante é liberado para saída.

Como as memórias M9K, utilizadas como memória interna (4.2.4), possuem duas portas independentes, uma pode ser utilizada para a leitura sequencial dos parâmetros e a outra para definição de novos parâmetros na memória. A ocorrência de operações de leitura e escrita em simultâneo, no mesmo endereço, provenientes de portas diferentes, é uma exceção conhecida e gera uma leitura de valor desconhecido (ALTERA CORPORATION, 2008). Para evitar este tipo de exceção, a porta utilizada para escrita utiliza o *clock* negado gerado no PLL do sistema (4.2). O diagrama da Ilustração 12 apresenta o esboço inicial da expansão do *hardware* da Ilustração 11 em múltiplos osciladores.

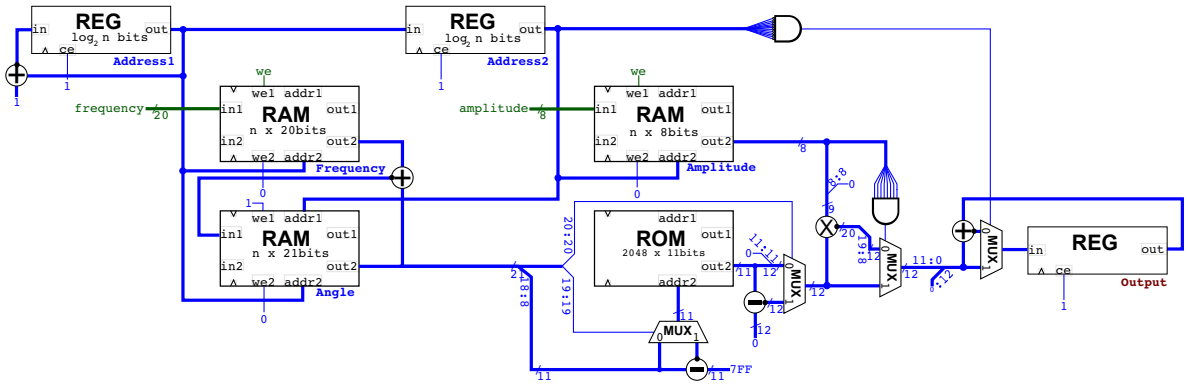


Ilustração 12: Expansão de um gerador de senoide única em múltiplas senoides

6 Embora o mecanismo de funcionamento básico seja o mesmo, para evitar confusão entre o módulo gerador e seus "geradores" internos, esses serão denominados "osciladores".

Além da frequência e da amplitude, outros parâmetros são necessários para o ajuste das senoides. Primeiramente, cada uma das harmônicas possui um parâmetro de fase, que nada mais é que uma constante somada ao ângulo. Pela modelagem das harmônicas segundo o Loris (3.2.1), a fase se mantém constante durante todo o período de execução da harmônica, diferentemente da frequência e da amplitude, que variam com o tempo. Por ser uma função periódica, não faz sentido a fase ser maior que um período completo da onda e, assim, a largura do valor desse parâmetro é de 13 *bits*, representando quatro vezes o número de endereços de memória, ou de segmentos de ângulo, armazenados.

O protocolo MIDI ainda define os parâmetros de nota e velocidade. A velocidade tem o mesmo efeito que a amplitude na forma de onda final, e é aplicada como uma multiplicação da mesma maneira. Entretanto, ao invés de multiplicar a onda de saída, como a amplitude, a velocidade multiplica a própria amplitude, gerando um produto que representa a amplitude final da onda. Embora nesse momento a ordem dos fatores não altere o produto, o cálculo da amplitude final será utilizado em etapas posteriores para o ajuste proporcional da onda gerada.

Em relação ao parâmetro nota, sabe-se que o número da nota MIDI tem uma relação direta com a altura da nota executada[7] e, logo, com a frequência da harmônica principal. Como as frequências das demais harmônicas são múltiplas da fundamental, o número da nota MIDI determina uma constante de multiplicação a partir de um ponto de referência. Definindo-se que serão armazenados em memória os valores de frequência de todas as harmônicas para a nota MIDI de número 0, a menor frequência possível, e considerando-se a relação de frequência entre notas consecutivas estabelecida na subseção 3.1.2, a constante de multiplicação de frequência que relaciona qualquer altura com a amostrada é a da Equação 5, onde *n* é o número MIDI da nota.

$$FrMul = 2^{\frac{n}{12}}$$

Equação 5: Fator de multiplicação de frequência

Como são definidas pelo MIDI 128 notas diferentes, o valor máximo dessa constante de multiplicação é de, aproximadamente, 1534,25 Hz. Assim, as 128 constantes de

7 Deve-se evitar confusão entre a nota e o parâmetro de nota. A nota é o conceito de teoria musical – Lá médio, por exemplo – enquanto o parâmetro é o número de 7 *bits* que o protocolo MIDI usa para representá-la.

multiplicação (uma para cada nota MIDI) são definidas como números de 19 *bits*, 11 para a parte inteira e 8 para a mantissa, multiplexados de acordo com um parâmetro que representa a nota, de 7 *bits*. A Ilustração 14 representa a porção do *hardware* do gerador que incrementa o ângulo segundo os parâmetros de nota, frequência e fase.

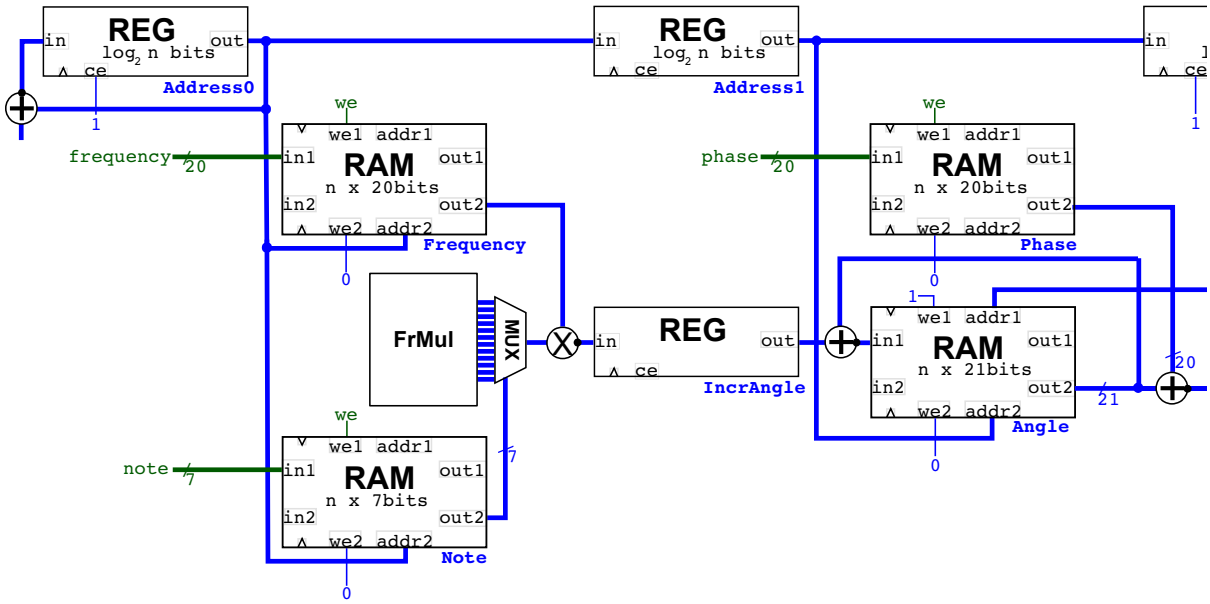


Ilustração 13: Adição dos parâmetros de nota e fase no gerador

A multiplicação da constante de nota pela frequência resulta em um número cuja parte inteira é de 23 *bits*, o que facilmente extrapola o critério do Teorema de Nyquist. Sabe-se que a frequência final da onda gerada não pode ter uma largura maior que 12 *bits*, então se qualquer um dos 11 *bits* mais significativos deste produto estiver em nível lógico alto, o ângulo não é incrementado e, posteriormente, o oscilador correspondente será temporariamente desativado (não representado no diagrama da Ilustração 14).

A desativação de um oscilador instrui o gerador a não somar a respectiva onda no registrador de saída. Em geral, um gerador é desativado quando a nota cuja harmônica ele estava gerando termina, ou seja, ao fim do tempo de repouso do envelope ADSR (3.1.5). As fases do envelope e a variação de frequência e amplitude são determinadas de acordo com as amostras de instrumento, salvas na memória *flash*. As informações sobre cada variação, doravante denominadas pacotes, são salvas como os atributos de frequência e amplitude a serem aplicadas, o endereço de memória do próximo pacote e o intervalo de tempo até a

próxima variação. Assim, a organização em memória de cada amostra de instrumento é semelhante a uma lista encadeada.

Quando o sistema recebe uma mensagem MIDI de ativação de nota (*Note On* – 3.3.4), o módulo de controle de notas gera requisições de leitura na *flash* para os endereços do primeiro pacote de cada uma das harmônicas que o instrumento associado ao canal MIDI ativado utilizará. Esses pacotes, após lidos, são enviados ao módulo gerador para ativação dos osciladores. O módulo também se encarrega de controlar o tempo para gerar novos pedidos de leitura em memória quando o tempo do pacote houver terminado, de acordo com a informação de intervalo entre variações consecutivas. A cada término de operação de leitura, os parâmetros obtidos são aplicados nos osciladores, subsequentemente até que a informação de próximo endereço a ser lido seja zero, convencionado como fim do período de execução da nota.

Como as leituras na *flash* são mais lentas que a operação normal do módulo gerador, fica claro que é necessário a inclusão de *buffers* para tratar a possibilidade de muitas leituras consecutivas, provenientes de osciladores diferentes, além de um módulo de controle para as leituras na *flash* provenientes de diferentes partes do sistema. Mais detalhes sobre o módulo de controle de leituras na *flash*, bem como o módulo de controle de notas citado anteriormente, estão disponíveis nas respectivas subseções (4.2.4 e 4.2.2).

Para controle do tempo, foi adicionado um registrador que é incrementado a cada intervalo do período de amostragem. O valor desse registrador representa um instante de tempo dentro de um intervalo pré-fixado, de pouco menos de 1,5 segundo, de acordo com a sua largura de dados, 16 *bits*, e a frequência de amostragem convencionada, 44,1 Khz. Assim, de maneira semelhante aos demais parâmetros, foi incluído um módulo de memória interna para armazenar o instante de tempo em que o próximo pacote deve ser lido. Esse valor é atualizado a cada recebimento de pacote, e recebe o valor do registrador de tempo naquele instante somado ao intervalo de tempo determinado pelo pacote. Durante o processo de amostragem dos instrumentos, portanto, cuidado deve ser tomado para que o intervalo de tempo entre dois pacotes não seja maior que o valor máximo representado por 16 *bits*.

Sempre que o instante de tempo lido da memória interna for idêntico ao valor do registrador do controle de tempo, o módulo gerador gerará um sinal de saída para requisição de uma leitura na *flash* no endereço do próximo pacote, que também é salvo em memória,

desde que este não seja igual a zero. O valor do próximo endereço igual a zero propaga para os próximos estágios de geração da senoide a informação de que o oscilador em questão está desativado e, portanto, seu sinal de saída não deve ser somado na onda final. Como o módulo que controla o ligamento e desligamento de notas não é o gerador, o estado de cada um dos osciladores é propagado também para o módulo de controle.

Como apenas uma porta da memória que armazena os valores de seno está utilizada, todos os demais elementos do gerador foram replicados, de modo a utilizá-la plenamente. A possibilidade de gerar duas senoides simultaneamente também permite a utilização de um *clock* de menor frequência sem alterar a taxa de amostragem ou, ainda, a elevação da taxa de amostragem sem alterar o *clock*. Isso porque, já que o valor de saída é integralmente gerado a cada fluxo completo do sistema, o período de amostragem da onda é o produto entre o número de ciclos de *clock* necessários para ler os parâmetros de todos os osciladores e o período do *clock* utilizado. A posterior síntese do sistema sintetizador como um todo demonstrou que o maior número de osciladores que a FPGA poderia abrigar, considerando apenas potências de dois, é de 2048 osciladores. Assim, com base nesse número, definiu-se o *clock* do sistema em 45 MHz, totalizando uma taxa de amostragem de 43.945,3125 Hz, suficientemente próxima do objetivo, 44,1 Khz.

Por fim, quanto mais senoides forem geradas e somadas, maior será a amplitude da onda resultante. Esse, porém, não é o comportamento esperado do sistema. Portanto, a amplitude da onda resultante deve ser ajustada proporcionalmente de acordo com as amplitudes das ondas individuais. A representação da soma de todas as senoides, além disso, deve ter a sua largura em *bits* estendida para evitar *overflow*.

O sistema utiliza o valor da amplitude final de cada senoide, que é o produto da velocidade com a amplitude da onda obtido anteriormente, para obter a soma de todas as amplitudes e a máxima amplitude aplicada a um oscilador único. Assim, para cada instante de amostragem, a onda de saída final é obtida pela razão entre a soma de todas as senoides e a soma de todas as amplitudes, o que ajusta a onda de volta a um inteiro de 12 *bits*, multiplicada pela maior amplitude entre os osciladores.

Como as operações de divisão, principalmente, e multiplicação finais são muito custosas, que tipicamente se estendem por vários ciclos de *clock*, os operandos e o resultado são registrados a cada fluxo completo do sistema gerador, ou a cada período de amostragem.

Além da forma de onda de saída, que será enviada para o módulo de saída (4.2.4) para conversão de digital para analógica, o gerador ainda cria uma saída auxiliar que representa o número de osciladores ativos a cada ciclo. Essa saída é utilizada apenas para exibição nos visores de sete segmentos da *DE0*, como uma representação de o quanto os recursos do sistema estão sendo utilizados. A própria representação binária da onda de saída também é enviada diretamente aos LEDs da placa, para uma melhor visão da amplitude final instantânea da saída.

4.2.2 Módulo de controle de notas

O módulo de controle de notas é responsável pelo controle do gerador no que diz respeito a ligamento e desligamento de notas, conforme instruído pelos eventos de mensagens MIDI recebidas (*Note On* e *Note Off* – 3.3.4). O módulo utiliza alguns dispositivos de memória interna (4.2.4), que armazenam o canal MIDI associado a cada oscilador do gerador, o estado de cada oscilador, segundo propagado pelo gerador, o estado definido pelo controle e o endereço da *flash* que contém o primeiro pacote (que contém informações dos parâmetros usados pelo gerador, conforme 4.2.1) do período de repouso, segundo o envelope ADSR do instrumento amostrado (3.1.5). O controle é feito por uma máquina de estados, conforme o diagrama de estados simplificado da Ilustração 14. Um diagrama de estados mais detalhado está disponível no Apêndice C.

O sistema registra dois endereços de osciladores, um de busca e um de parada. A cada ciclo de *clock*, o endereço de busca é enviado ao módulo gerador para obtenção do parâmetro de nota do oscilador em questão. Das memórias internas, são obtidas as informações de estado obtida do gerador, estado definido pelo próprio controle, canal e endereço de repouso. Duas indicações de estado são necessárias pois a desativação de uma nota, pelo controle, não significa a desativação de todos os osciladores a ela associados. A desativação dos osciladores propriamente dita é controlada pelo módulo gerador, ao final do período de repouso.

As mensagens MIDI são recebidas pelo módulo de um *buffer* (4.2.4) de entrada. A existência de dados no *buffer* desencadeia a operação da máquina de estados e, ao término de cada operação, é gerada uma mensagem para o *buffer* reconhecendo a leitura de um dado.

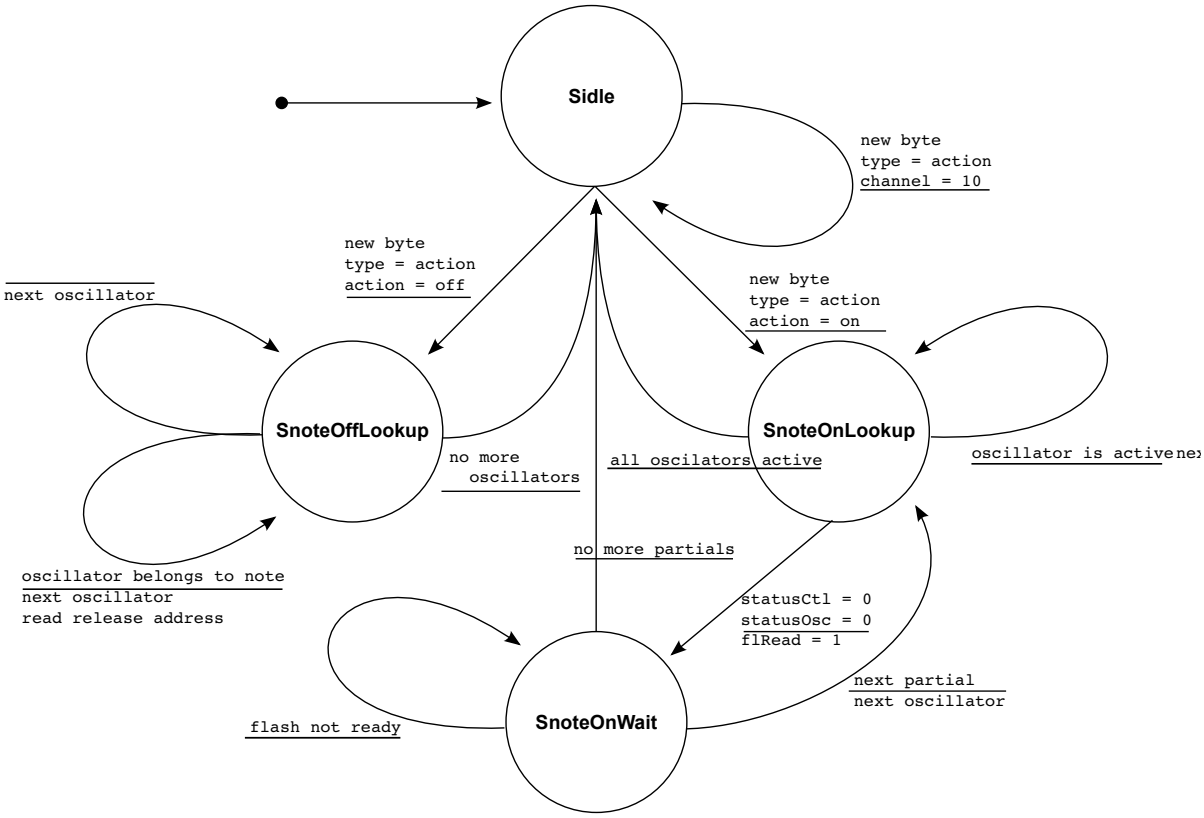


Ilustração 14: Diagrama de estados do controle de notas

Quando uma mensagem de *Note On* é recebida, o sistema inicia uma busca por osciladores desativados. Quando o primeiro oscilador desativado é encontrado, o sistema lê da *flash* o pacote inicial da primeira harmônica do instrumento associado ao canal, a partir do endereço armazenado em memória durante a execução anterior de uma mensagem de *Program Change* (4.2). O primeiro pacote de uma harmônica contém, além dos parâmetros utilizados pelo gerador para criação da senoide, o endereço de repouso daquela harmônica, o endereço de memória inicial da próxima (ou zero, caso não haja uma próxima) e a fase da senoide correspondente. Quando a leitura na *flash* termina, o parâmetro de fase é enviado diretamente ao módulo gerador, juntamente com nota e velocidade, obtidos diretamente do *buffer* de entrada, e demais parâmetros, obtidos diretamente da *flash*. O endereço de repouso é armazenado para o posterior desligamento da nota e o processo se repete enquanto o endereço de início da próxima harmônica for diferente de zero. A busca também pode terminar anormalmente quando todos os osciladores já estiverem ativos, condição controlada pelo endereço de parada.

As instruções de leitura na *flash* não são enviadas diretamente ao controlador da *flash*, e sim armazenadas em um *buffer* circular, que armazena o endereço a ser lido e o oscilador que receberá os parâmetros lidos. O motivo disso é que a busca desencadeada pela mensagem de *Note Off* não aguarda pelos dados lidos da *flash*. A busca lê os valores de canal, nota e velocidade de todos os osciladores ativos e, caso coincidam com os contidos na mensagem MIDI recebida, desencadeia a leitura da *flash* do pacote inicial do período de repouso (salvo durante o *Note On*), a menos que esse endereço seja igual a zero. O endereço de repouso igual a zero significa que a harmônica em questão não possui período de sustentação e, portanto, entrará (ou entrou) em repouso naturalmente.

4.2.3 Módulo de tradução MIDI

O módulo de tradução MIDI recebe os *bytes* de dados das mensagens MIDI enviadas ao sintetizador e os agrupa e classifica quanto ao tipo de mensagem, considerando apenas *Note On*, *Note Off* e *Program Change* (3.3.4). As mensagens MIDI são recebidas diretamente da interface UART do sistema (4.2.4) e são geradas saídas com o número do canal onde será aplicada a operação da mensagem (*channel*), e os parâmetros de instrumento (*instrument*), nota (*note*) e velocidade (*velocity*), de acordo com o tipo de mensagem. As mensagens tratadas são encaminhadas a dois *buffers* circulares (4.2.4) independentes, um para as mensagens de troca de instrumentos e outro para ligamento e desligamento de notas. A Ilustração 15 demonstra a máquina de estados de controle do módulo. Uma visão mais detalhada do diagrama de estados está disponível no Apêndice D.

A transição entre estados é controlada pela entrada, recebida da UART, de que um novo *byte* de dados está disponível. As mensagens são classificadas entre troca de programas e operação de notas de acordo com os *bits* mais significativos, conforme a Tabela 2. Como o *bit* mais significativo dos *bytes* de dados de mensagens MIDI é sempre zero, essa condição também é utilizada para controlar a transição entre estados. Uma mensagem recebida como *Note On* que possui um parâmetro de velocidade igual a zero é interpretada, após o último estado, como um *Note Off*.

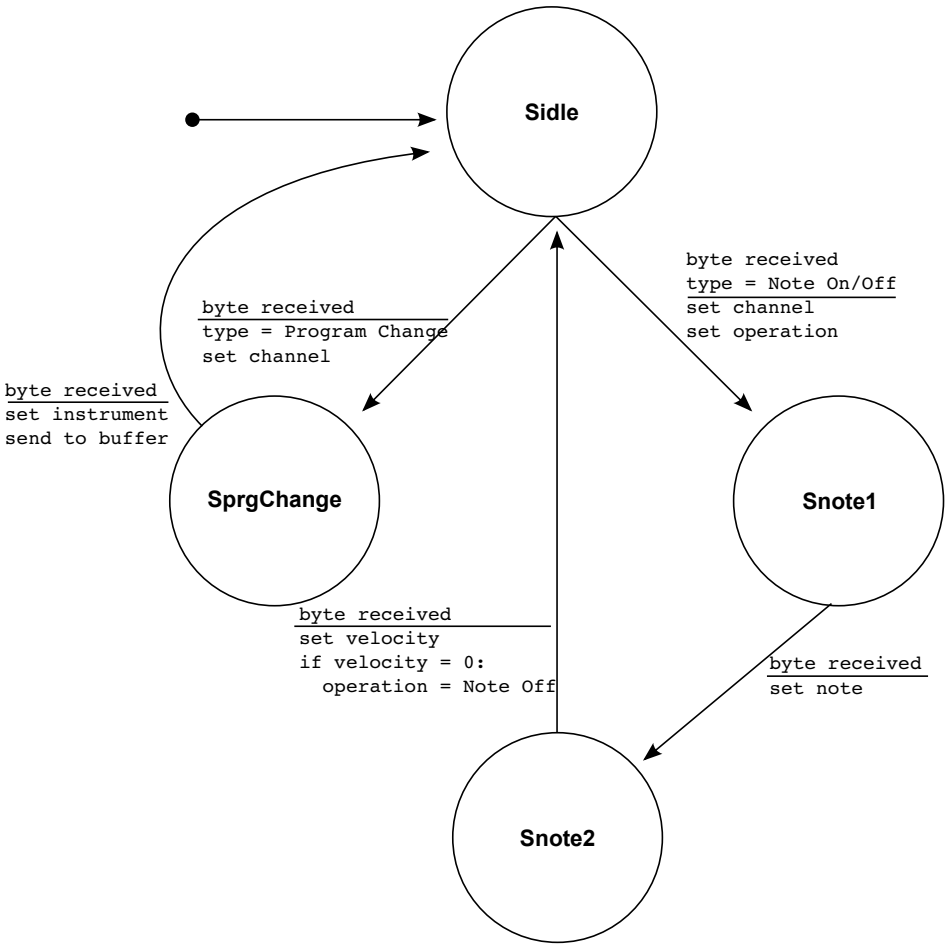


Ilustração 15: Diagrama de estados do tradutor de mensagens MIDI

4.2.4 Módulos auxiliares

Alguns módulos projetados para o sistema, embora essenciais para o funcionamento correto como um todo, não estão diretamente relacionados à síntese do áudio ou ao sequenciamento MIDI. Esses são considerados módulos auxiliares, e são aqui apresentados em menor nível de detalhamento, apenas de acordo com o comportamento em alto nível de abstração e a interação esperada com os demais módulos do sistema. São considerados módulos auxiliares as memórias internas, *buffers*, o módulo de saída, a interface UART e o controlador de memória *flash*.

A FPGA *Cyclone III* possui 56 módulos de memória M9K. Estes módulos podem ser configurados como memórias síncronas de duas portas, independentes. Como o sistema do

sintetizador necessita processar grande quantidade de dados, decidiu-se utilizar os blocos de memória para armazenagem dos parâmetros e resultados parciais. Um módulo de memória foi implementado através da inferência direta de uma memória de duas portas, com o auxílio de uma função disponibilizada pelo fabricante. A função permite a inicialização da memória a partir de um arquivo em formato próprio, aspecto importante para alguns segmentos do sistema que exigem valores iniciais. A Ilustração 16 representa um módulo de memória interna nos demais diagramas de blocos. Para cada uma das duas portas, os sinais *in* e *out* representam a entrada e saída de dados, respectivamente, *addr* é o endereço a ser lido/escrito e *we* o sinal de permissão de escrita. O módulo é parametrizado para receber tamanhos de palavras de dados e endereços customizados para cada necessidade.

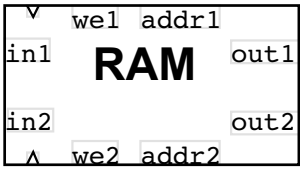


Ilustração 16: Representação do módulo de memória interna

Também foi descrito um módulo de *buffer* circular para armazenamento temporário de operações. Como as amostras de instrumentos são armazenadas em memória *Flash*, por exemplo, e esta memória é mais lenta que o restante do sistema, o *buffer* é necessário para possibilitar o armazenamento dos próximos endereços a serem explorados. O *buffer* implementa entradas de permissão de escrita (*we*) e reconhecimento de leitura (*read*) para controle, e gera saídas auxiliares que indicam o seu estado, se vazio (*empty*) ou cheio (*full*). O sinal de *reset* síncrono do *buffer* força o seu esvaziamento. A Ilustração 17 apresenta a representação do módulo de *buffer* nos demais diagramas de blocos do sistema, enquanto o Apêndice E contém o diagrama de blocos mais detalhado do módulo.

Ilustração 17: Representação do módulo de buffer circular

O módulo de saída é um conversor de digital para analógico na forma de um dispositivo de modulação de largura de pulso (PWM). O PWM adquire a representação binária da onda de saída do gerador (4.2.1) e gera dois sinais de saída diferenciais que, após amplificação, podem ser reproduzidos em um dispositivo alto-falante comum. Porém, para que o contador do PWM seja capaz de atingir todos os níveis possíveis da onda de saída dentro do período de amostragem, a frequência do *clock* do dispositivo de saída foi elevada, em relação ao restante do sintetizador, para 360 MHz. Além disso, duas adaptações foram feitas no conceito principal de um dispositivo PWM. Primeiramente, a forma de onda de saída do gerador não é comparada com o contador interno do PWM diretamente, mas sim com a sua representação binária "invertida", substituindo o *bit* menos significativo pelo mais significativo e assim subsequentemente. Isso em nada altera a largura do pulso de saída dentro do período de amostragem, porém aumenta consideravelmente o número de variações do sinal de saída, consequentemente elevando a frequência das componentes geradas pela característica de onda quadrada da saída do PWM. As componentes de alta frequência derivadas do PWM tradicional podem ser bastante audíveis como "zumbidos" e ruídos desagradáveis. A segunda adaptação é um controle de nível DC no sinal de saída. As saídas diferenciais do PWM são registradas em um registrador de deslocamento de duas posições, e aplicadas diretamente aos pinos de saída apenas se ambas as posições forem idênticas. Caso sejam diferentes, as duas posições se anulam e a saída é definida em alta impedância. O nível DC do sinal também é bastante audível, principalmente quando não há som sendo executado e o circuito amplificador de saída é mais simples.

A interface UART do sintetizador foi descrita para receber as mensagens MIDI no formato de *bytes* de dados, conforme as definições da subseção 3.3.4. O módulo foi definido como uma máquina de estados que aguarda sequencialmente o tempo de um *bit* e adquire o valor da linha, sempre na metade do tempo de *bit*. Sempre que há uma transição na linha de entrada, o contador de tempo interno se ressincroniza com a linha. Além disso, para evitar problemas de metaestabilidade, a entrada da linha é duplamente registrada antes de entrar no sistema. As saídas geradas são o último *byte* recebido e um pulso indicando quando um novo dado está disponível, que são utilizadas pelo módulo de tradução MIDI (4.2.3) para controle do sistema como um todo.

Por fim, as amostras de instrumentos do sistema são todas obtidas de uma memória *flash* externa, e os dados são lidos por módulos distintos. Por isso, foi definido um módulo

controlador de memória *flash*, que define quatro portas de leitura de dados, organizadas por prioridade. A primeira porta, de largura de 16 *bits*, é utilizada para leitura do endereço inicial da primeira harmônica de cada instrumento, durante as operações de *Program Change*. As duas portas seguintes, de largura de 96 *bits*, são utilizadas pelo módulo gerador (4.2.1) para leitura dos pacotes de amostras. São necessárias duas portas para isso pois o gerador pode ler dois endereços simultaneamente. Por fim, a quarta porta de leitura, de 112 *bits*, faz a leitura do primeiro pacote de uma harmônica, que também contém o endereço da próxima harmônica, endereço de repouso e parâmetro de fase, e é lido pelo módulo de controle de notas (4.2.2). Essa é a porta de menor prioridade para evitar que uma ou mais harmônicas sejam incorretamente "reativadas" por requisições em *buffer* do gerador conflitantes com uma requisição de desativação do respectivo oscilador pelo controle.

4.3 Amostragem dos instrumentos

As amostras de instrumentos utilizadas foram obtidas da página dos Estúdios de Música Eletrônica da Universidade de Iowa (FRITTS, [s.d.]). A página possui uma coletânea de diversos instrumentos tocando diferentes notas em diferentes velocidades, também variando outros parâmetros como *vibrattos* em instrumentos de sopro. Após o *download* das amostras, foi utilizado o *software* Audacity (4.1.5) para edição do áudio: redução de ruído, eliminação de períodos de silêncio, filtragem de altas e baixas frequências, adequação de ganho de amplitude e remoção de um dos canais (o Loris não suporta audio *stereo*).

O áudio resultante é analisado utilizando o Loris (3.2.1) por um *software* escrito em linguagem Python. A saída gerada pelo Loris contém frequências e amplitudes em alta precisão distribuídos em intervalos de tempo igualmente espaçados, portanto há espaço para algumas otimizações. Primeiramente, a amplitude da onda é ajustada proporcionalmente, de modo que o pico máximo de amplitude seja unitário.

Logo após, são eliminados todos os "pacotes" que contém uma variação em frequência, amplitude e tempo que não pode ser representada em relação ao anterior, devido às larguras de dados definidas para esses parâmetros. As harmônicas que não contêm pacotes então são eliminadas.

O próximo passo é percorrer todos os pacotes em busca de intervalos de tempo que extrapolem o maior tempo capaz de ser representado pelo gerador (4.2.1), de aproximadamente 1,5 segundos. Caso tais intervalos sejam encontrados, são adicionados pacotes intermediários, interpolando linearmente os parâmetros de frequência e amplitude.

Por fim, o *software* recebe parâmetros que contêm um instante de tempo em que o som do instrumento está dentro do período de sustentação do envelope ADSR (3.1.5) e uma tolerância de amplitude de sustentação. O sistema então percorre a onda analisada a partir do instante selecionado e define os tempos de início e fim do período de sustentação, de acordo com a tolerância, criando a lista encadeada de pacotes de acordo: a execução dos pacotes de sustentação será feita repetidamente até o recebimento de uma mensagem de *Note Off*. Caso os parâmetros de instante e tolerância de sustentação sejam omitidos, o programa considera que o instrumento não se sustenta e preenche o endereço de repouso com zero (conforme explicitado em 4.2.2).

Por fim, todos os dados obtidos das amostras são convertidos para suas representações binárias de acordo com as larguras de dados pré-definidas para cada um dos módulos do sistema, e agrupados em um arquivo binário. A gravação da memória *flash* a partir desse arquivo é feita utilizando o *software DE0 Control Panel* (4.1.1).

4.4 Verificação final

Para verificação final do sistema, foi montado o ambiente de testes do diagrama da Ilustração 18. O *software* MuseScore é utilizado para leitura de partituras e arquivos MIDI. A execução da partitura é redirecionada, utilizando o Jack (com o auxílio do a2jmidid) para o teclado virtual VMPK para uma visualização gráfica da execução das notas. Alternativamente, as notas podem ser tocadas manualmente a partir do próprio VMPK.

Por utilizar portas ALSA MIDI, o teclado virtual pode ser conectado diretamente à ponte estabelecida pelo Hairless MIDI to Serial Bridge. A ponte envia as mensagens MIDI através da porta *serial* virtual estabelecida sobre a conexão USB com a placa de desenvolvimento DISCOVERY, onde um *firmware* em *Python* rodando sobre o *MicroPython* lê as mensagens recebidas e as redireciona para uma das interfaces UART do

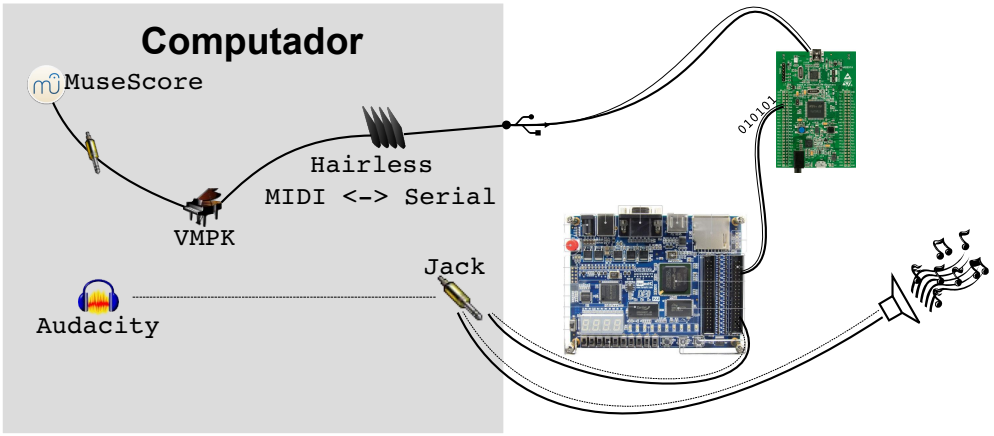


Ilustração 18: Ambiente de testes para verificação final

microcontrolador. O transmissor dessa interface UART está fisicamente conectado com um dos pinos de entrada e saída da placa de desenvolvimento DE0.

O módulo de UART do sintetizador recebe a entrada de linha a partir desse pino de entrada e saída, desencadeando todo o fluxo de controle e síntese de áudio do sistema principal. A saída diferencial do PWM, conectada diretamente a pinos de entrada e saída da placa de desenvolvimento, foi soldada a um conector P2 fêmea juntamente com uma referência de terra. Um cabo P2 comum conecta a saída do PWM diretamente à entrada de microfone da placa de som do computador utilizado.

No computador, o Jack novamente faz uma conexão virtual entre a entrada de microfone e a saída de alto-falantes, proporcionando a saída de áudio. Alternativamente, os alto-falantes também podem ser conectados diretamente ao P2 fêmea soldado à placa, porém para facilitar a posterior análise do áudio sintetizado e evitar constantes conexões e desconexões do alto falante, a saída foi conectada ao computador.

O editor Audacity pode ser utilizado para comparação visual entre a onda sintetizada e a amostra obtida. Além disso, o sintetizador FluidSynth pode ser conectado à saída do teclado virtual ou à saída do MuseScore para proporcionar uma comparação entre o som sintetizado em *software* por um sintetizador reconhecido e o gerado pelo *hardware*.

5 RESULTADOS E DISCUSSÃO

Todos os módulos de *hardware* propostos foram testados por meio de simulações, quando possível, e posterior síntese para gravação na FPGA utilizada, para testes por inspeção. A simulação de todas a capacidade do dispositivo proposto, porém, não é viável com os recursos disponíveis. As simulações obtidas cobrem pequenas frações de segundo para demonstração de funcionamento do *hardware* segundo as características mais básicas, enquanto a operação do sistema pode se desenvolver durante longos período de tempo, executando muito mais operações em simultâneo que uma simulação em um computador comum seria capaz de suportar. Apenas para efeito de exemplificação, uma simulação do sintetizador completo que foi executada durante aproximadamente 20 minutos mal pôde representar a recepção de duas mensagens MIDI, uma de *Program Change* e uma de *Note On*, e os primeiros instantes de síntese da senoide.

Por esse motivo, grande parte dos testes foram efetuados após síntese do *hardware*, por inspeção visual, com o auxílio dos LEDs e visores de sete segmentos, ou auditiva. Esta seção descreve os resultados dos testes, destacando, quando aplicável, problemas encontrados e possíveis soluções ou melhorias.

5.1 Verificação do módulo gerador

A simulação do módulo gerador permite verificar a correta geração das senoides independentes. Primeiramente, o conceito inicial do módulo gerador (Ilustração 11) foi simulado. No resultado, demonstrado na Ilustração 19, pode-se perceber a correta geração de uma senoide com frequência e amplitude variáveis. O mesmo *hardware* foi sintetizado e gravado na FPGA e, com o auxílio do módulo de saída (4.2.4), as senoides geradas foram redirecionadas como áudio para alto-falantes, utilizando os botões da FPGA para determinar os parâmetros de entrada. Como resultado, pode-se ouvir tons senoidais puros, mesmo para frequências mais elevadas (que acabam representadas mais “quadradas”, devido às perdas de resolução, nas simulações) comprovando assim o correto funcionamento do sistema proposto.

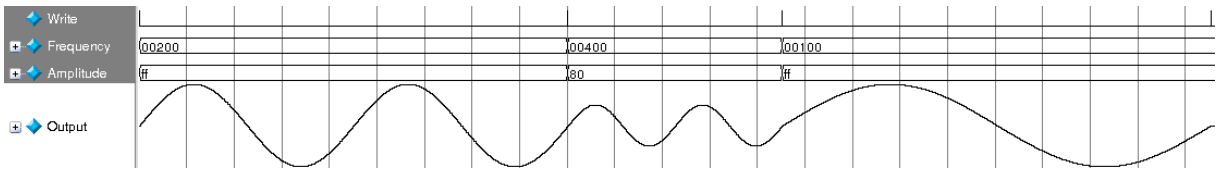


Ilustração 19: Simulação inicial do módulo gerador

O módulo simulador final, após adicionados os demais parâmetros e a capacidade de controle de tempo, também foi simulado na ferramenta. Na simulação, o comportamento das leituras da memória *flash* foi reproduzido, e as variações de frequência e amplitude foram então controladas pelo próprio módulo gerador. O resultado está reproduzido na Ilustração 20. No teste, foram utilizados os primeiros parâmetros do teste anterior, para efeitos de comparação. O resultado evidencia que o módulo gerador é capaz de controlar a variação temporal dos parâmetros.

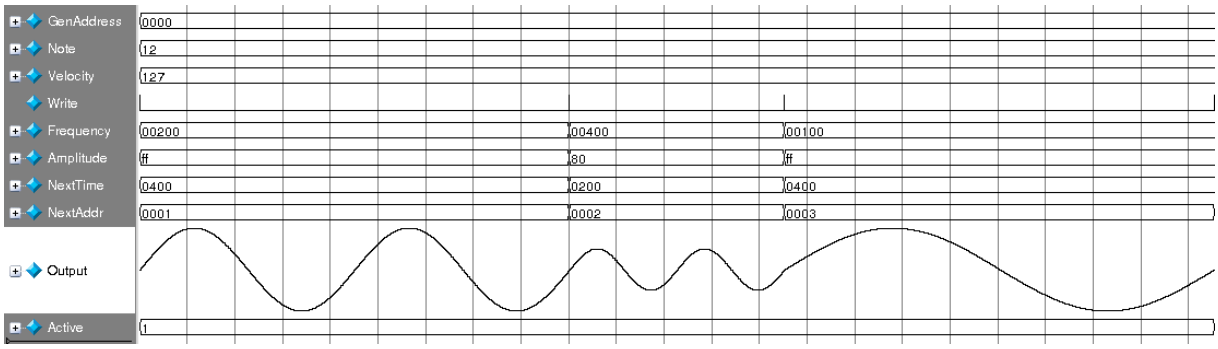


Ilustração 20: Simulação do módulo gerador completo

A geração de várias senoides em simultâneo foi verificada após síntese e gravação na FPGA, anteriormente à adição do controle temporal, com o auxílio de um processador interno, de propriedade intelectual da *Altera*, para gerar os parâmetros das senoides. A execução de tons senoidais puros apresenta características bastante distintas e audíveis, sendo portanto possível verificar-se quando várias senoides estão sendo geradas em simultâneo. O uso do processador interno também possibilitou a síntese dos primeiros sons musicais obtidos, porém de modo bastante lento. Além disso, foi esse teste inicial de geração de múltiplas ondas em simultâneo que motivou a evolução do sistema para um gerador capaz de controlar também o aspecto temporal da variação dos parâmetros.

5.2 Verificação do módulo de tradução MIDI

A verificação do módulo de tradução MIDI foi feita diretamente na placa de desenvolvimento, com o auxílio da interface UART, dos *softwares* de teclado virtual e ponte MIDI-*serial* e da placa DISCOVERY. As saídas do módulo foram associadas com os LEDs e visores de sete segmentos da placa, de modo a fornecer um método visual de verificação. Os visores mostram o canal MIDI em que o último evento foi recebido e, em caso de mensagem de *Program Change*, o número do instrumento ou a velocidade, em caso de ação de notas. Os LEDs foram associados às notas Dó a Si da quarta oitava (notação MIDI 60 a 71), excluídos os acidentes. A Ilustração 21 exemplifica o conceito (a foto foi escurecida para uma melhor visualização da informação nos visores).

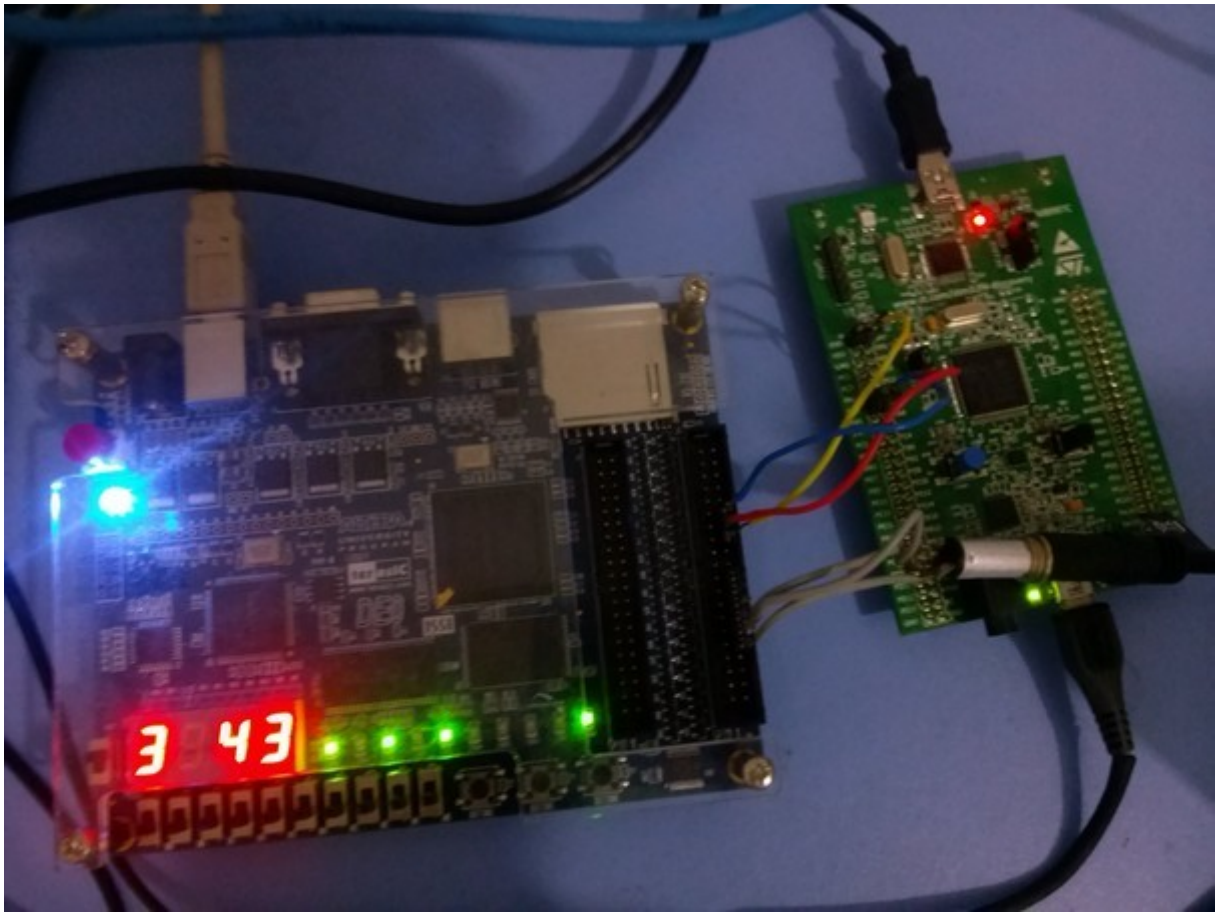


Ilustração 21: Exemplo de verificação do módulo de tradução MIDI

No exemplo da Ilustração 21, foram pressionadas as teclas Dó, Mi e Sol no teclado virtual, simultaneamente. O LED aceso mais à direita indica que a última mensagem recebida foi de *Note On*. No visor, pode-se verificar que foi escolhido o canal MIDI 4 (os canais MIDI são definidos de 1 a 16, mas na placa são exibidos os números, em hexadecimal, de 0 a F) e a velocidade 67 (43, também em hexadecimal).

As mensagens de *Note Off* e *Program Change* foram verificadas de maneira semelhante. Além disso, o sistema foi conectado ao editor de partituras MuseScore para experimentação no aspecto temporal, com múltiplos eventos acontecendo ao mesmo tempo e em vários canais. Os atrasos, evidentemente existentes, são, porém, imperceptíveis na execução em tempo real do sistema.

5.3 Verificação do módulo de controle de notas

A verificação do módulo de controle de notas foi feita por meio de exaustivas simulações da operação do módulo em conjunto com gerador, controlador de *flash*, tradutor MIDI e *buffers* associados. A simulação do controle em conjunto com os demais módulos, apesar de muito mais demorada que a simulação do módulo isoladamente, abrange muito mais condições do sistema, e possibilitou a correção de falhas que seriam praticamente impossíveis de identificar no sistema após sintetizado.

Para simulação, foi elaborado um modelo de memória *flash* contendo valores de parâmetros arbitrários. As mensagens MIDI foram inseridas manualmente no módulo tradutor, e foi possível então verificar o fluxo de dados entre os módulos. Foi observada a correta ativação e desativação de osciladores de acordo com as mensagens enviadas e as informações em memória. Devido aos aspectos temporais envolvidos, porém, as simulações abrangeram um ou dois eventos de nota, simulando instrumentos que utilizam poucas harmônicas e poucas variações de parâmetros.

Devido ao tamanho das simulações e número de sinais envolvidos, considera-se que a ilustração delas com gráficos em nada agregam para a compreensão do conceito. Por esse motivo, esses não estão inclusos neste relatório.

5.4 Verificação do sintetizador completo

Para verificação final, foi executado o procedimento de modelagem de um piano executando a nota Lá médio, a partir de amostras de áudio obtidas de um banco de dados (FRITTS, [s.d.]), conforme a subseção 4.3. Após adequação da amostra, amostragem e gravação na *flash*, testou-se o sistema com o teclado virtual e editor de partituras. O sistema sintetizador foi projetado para executar os sons a uma taxa de amostragem de, aproximadamente, 44,1 KHz (4.2.1), o que é o padrão *de facto* de taxa de amostragem para aplicações de áudio, também utilizado nos áudios obtidos do banco de dados. O Anexo C apresenta os detalhes da amostra obtida, conforme informados na origem.

O som do piano foi executado corretamente e de acordo com o som da amostra, com poucas diferenças audíveis. Um pouco de ruído de fundo é perceptível, o som sintetizado se estende por menos tempo e termina mais bruscamente, além de apresentar um instante de pico, durante o período de ataque, mais longo e "explosivo". Essas diferenças também são evidenciadas visivelmente nos gráficos da Ilustração 22, obtidos com o Audacity.

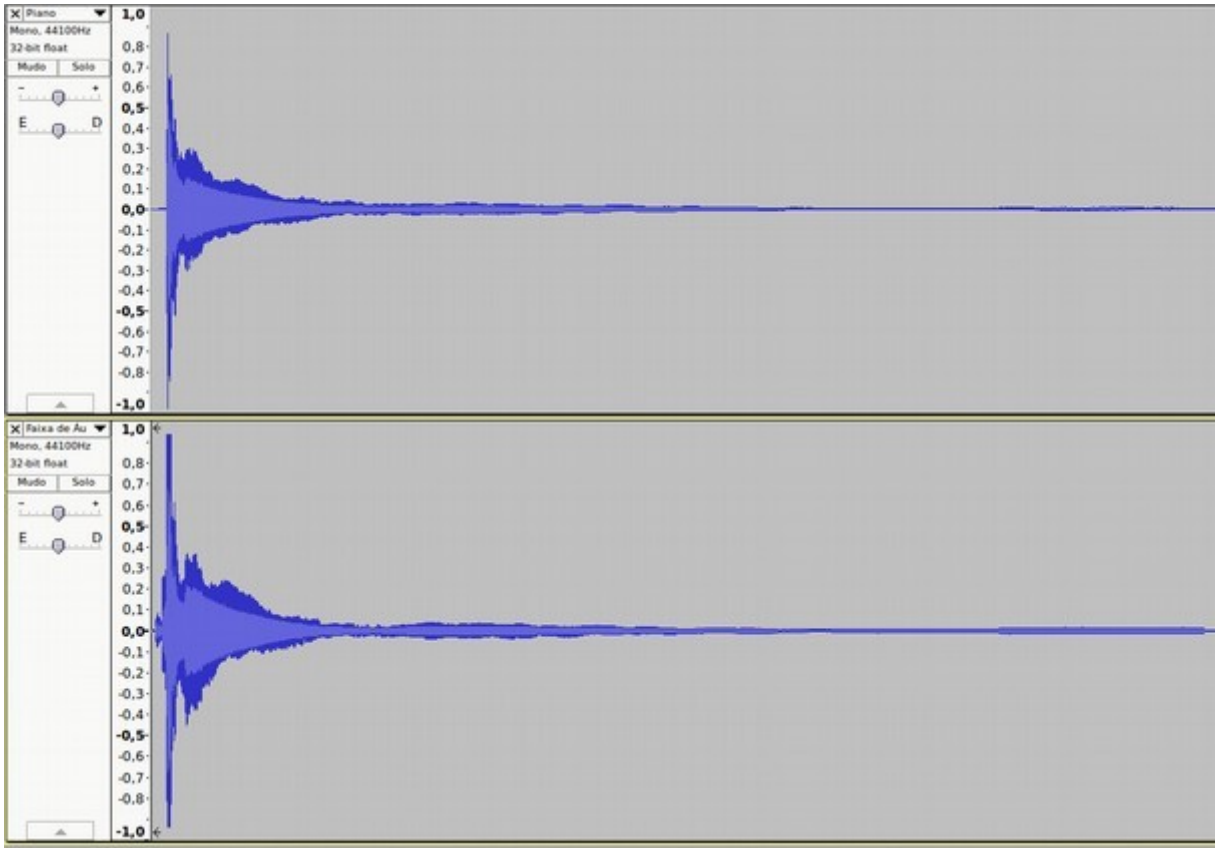


Ilustração 22: Formas de onda da amostra de piano (acima) e do piano sintetizado (abaixo)

O ruído de fundo é constante, mesmo quando nenhuma nota está sendo executada pelo sintetizador, e perceptível principalmente se amplificado. Porém, deve-se considerar que enquanto a amostra de piano original foi gravada em ambiente controlado, utilizando equipamentos de áudio profissionais, o som sintetizado foi gravado por uma placa de som convencional, conectada por um cabo não blindado a um conector exposto, com soldas caseiras.

Quanto às demais diferenças, elas podem ser explicadas pela eliminação de partes da amostra durante o processo de amostragem. Além disso, vários parâmetros de ajuste fino são definidos pelo *Loris* e não só podem, como deveriam, ser utilizados para melhorar o processo, no caso de uma aplicação menos experimental do sistema proposto.

As ilustrações 23 e 24, a seguir, contém a análise do espectro de frequências utilizando a Transformada Rápida de Fourier, também obtido com o Audacity. É perceptível a semelhança entre os espectros, principalmente nas componentes de menor frequência.

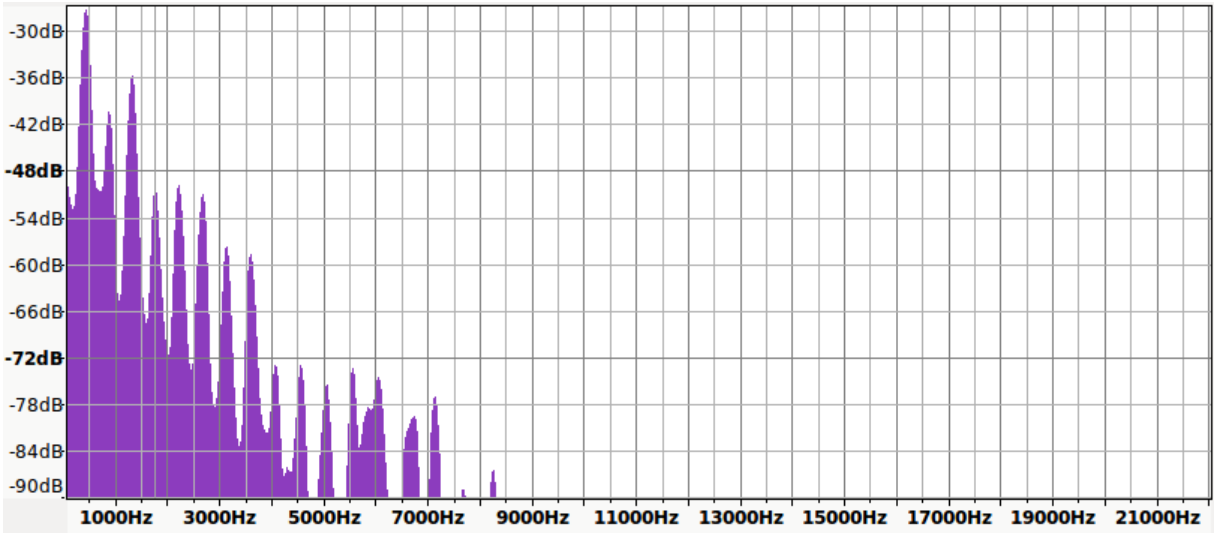


Ilustração 23: Espectro de frequências da amostra de piano

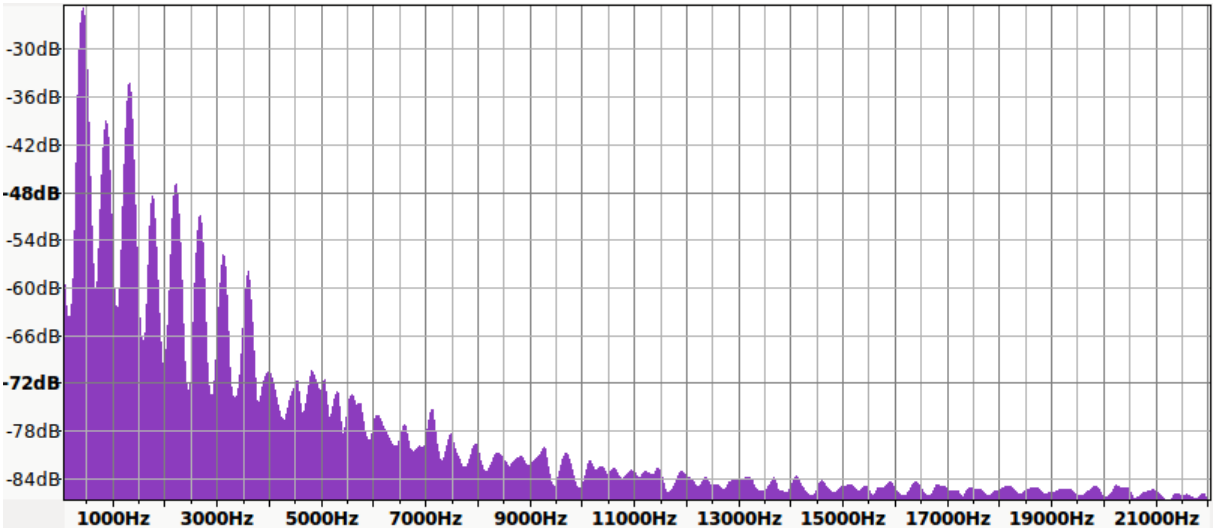


Ilustração 24: Espectro de frequências do piano sintetizado

O som sintetizado também foi comparado com o som obtido pelo sintetizador FluidSynth. Embora ambos possam ser identificados como sons de piano, eles não se parecem. O som obtido com o sintetizador proposto, de certo modo e avaliando apenas o aspecto básico, lembra mais o som de um piano real que o do *software*. É necessário destacar, porém, que o FluidSynth permite alterar a fonte utilizada para síntese, e que outras podem apresentar características diferentes. As ilustrações 25 e 26 contêm, respectivamente, a forma de onda do som gerado pelo FluidSynth e a análise de frequências.

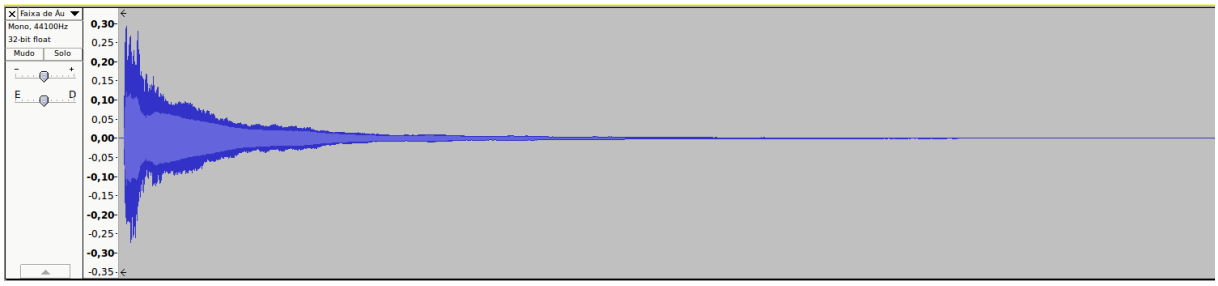


Ilustração 25: Forma de onda de um som de piano sintetizado por software

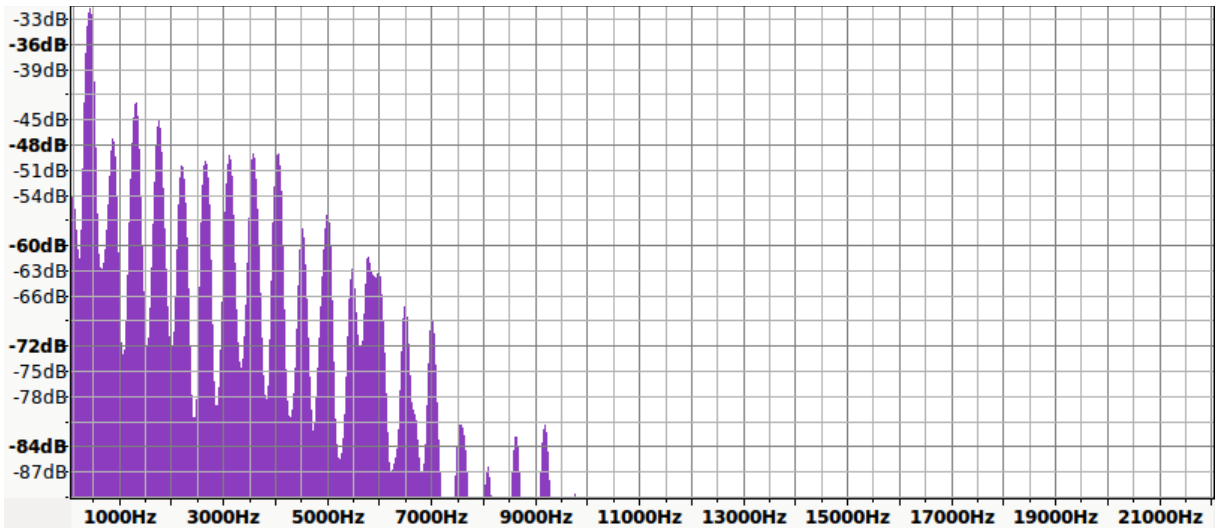


Ilustração 26: Espectro de frequências de um som de piano sintetizado por software

O gráfico da forma de onda da Ilustração 25 demonstra diferenças em relação aos demais principalmente durante o início do período de ataque, bem menos evidente. O restante do período da nota se assemelha nas componentes mais densas. Quanto ao espectro de frequências, nota-se a maior existência de componentes de maior frequência, podendo

explicar a percepção de ser o mesmo instrumento, mas ainda assim diferente. De fato, o som proveniente do *software* parece mais agudo, apesar de estar executando a mesma nota.

5.5 Verificação de performance

A última verificação realizada foi feita com o auxílio do MuseScore. Arquivos MIDI de melodias foram obtidos da internet para execução no sintetizador e avaliação da performance. A execução ocorreu normalmente, sem atrasos perceptíveis, mesmo com um alto número de notas executando ao mesmo tempo e execuções mais rápidas. Dois problemas foram, porém, identificados.

O primeiro problema está relacionado a ruído. Principalmente, mas não exclusivamente, quando o número de osciladores utilizados é mais elevado, são ouvidos ruídos como “cliques” no som. Embora mais experimentação seja necessária para identificar a origem dos ruídos, acredita-se que estejam relacionados com a operação de divisão, executada no gerador, para adequação proporcional da amplitude.

Essa hipótese é baseada no fato de que, durante a fase de desenvolvimento, devido a um erro cometido, as entradas do *hardware* de divisão não eram corretamente registradas, de modo que o tempo disponibilizado para a operação de divisão era menor que o projetado. Esse erro resultou em exatamente o mesmo tipo de ruído, porém, ele acontecia em intervalos muito mais próximos e durante a execução de muito menos notas simultâneas.

A operação de divisão foi descrita simplesmente utilizando-se o operador matemático de divisão. Durante o processo de síntese, uma função de propriedade do fabricante é automaticamente gerada para a operação. Essa função permite o ajuste de parâmetros para uma melhor adequação ao projeto, além de que outras funções, sequenciais e possivelmente mais eficientes, estão disponíveis no pacote. Assim, sugere-se a alteração do *hardware* de divisão como tentativa de solução do problema.

O segundo problema observado é a ocorrência eventual de travamentos do sistema como um todo. Quando o travamento ocorre, o *buffer* de entrada de notas fica cheio, indicando que a máquina de estados do controle de notas não está lendo novas mensagens do *buffer*. Os principais módulos para investigação do problema são o próprio controle de notas e

o controlador da *flash*, pois o controle de notas propositalmente aguarda a leitura dos pacotes da *flash* antes de prosseguir com as operações.

Esse aspecto de exceção ocorre apenas eventualmente e geralmente após alguns minutos de execução de uma melodia, de forma que a simulação é extremamente inviável. Nesse sentido, sugere-se o uso de um analisador lógico externo para visualização dos sinais de controle das máquinas de estado, em busca das condições que levam a exceção. Como maneira de contornar o problema, a rede de *reset* síncrona do sistema foi conectada às saídas dos *buffers* de entrada que indicam estado cheio, de forma que, quando o travamento ocorre, o sistema automaticamente se recupera.

5.6 Recursos utilizados

A Tabela 4 descrimina a utilização dos recursos disponíveis, tanto da FPGA em termos de número de elementos, como espaço utilizado da *flash* e alocação de osciladores no módulo gerador.

Tabela 4: Utilização dos recursos disponíveis

Recurso	Utilização absoluta	Utilização relativa
FPGA		
Elementos lógicos	3 969	26%
Funções combinacionais	3 767	24%
Registradores de lógica dedicada	813	5%
Pinos de entrada e saída	114	33%
Bits de memória	306 576	59%
Multiplicadores	24	21%
PLLs	1	25%
Sintetizador		
Uso da *flash*	24,3 KB	19%[8]
Número de osciladores		
Nota única de piano	14	0,7%
Executando melodia	> 592	> 29%

8 Devido à largura de endereço definida, a utilização da *flash* se limita a 128 KB.

Analisando os recursos utilizados da FPGA, percebe-se claramente que o fator limitante ao projeto foi a quantidade de dispositivos de memória M9K disponíveis. De fato, as memórias são amplamente utilizadas no sistema, e prevê-se a inclusão de novas funcionalidades utilizá-las-iam ainda mais. Não são incomuns, porém, FPGAs que incluem muito mais dispositivos de memória que a utilizada.

Quanto aos recursos do sintetizador, fica claro que são necessários mais osciladores e mais espaço de armazenamento na *flash*. O uso da *flash* pode ser estendido pela adaptação do sistema para suportar uma maior largura de endereço, o que tem um impacto direto na utilização de memória interna da FPGA. O aumento do número de osciladores, além de aumentar o utilização de memória interna, também exigiria o aumento na frequência do *clock* utilizado para que a frequência de amostragem seja mantida. Seria possível aumentar o *clock* do sistema para até 360 MHz sem extrapolar os requisitos temporais dos dispositivos internos da FPGA (ALTERA CORPORATION, 2008), o que possibilitaria um número de 16384 osciladores, contra os atuais 2048.

6 CONCLUSÃO

A qualidade do som sintetizado, bem como os baixos recursos utilizados e a boa possibilidade de expansão, demonstram que o sistema proposto é capaz de reproduzir sons musicais e pode, se ampliado, ser uma alternativa a outros sistemas existentes. Não foram observados problemas de performance durante a execução de melodias, e os problemas de ruídos e eventuais travamentos percebidos parecem facilmente resolvíveis se submetidos a investigação adicional.

Considerando-se possível a utilização de uma FPGA com mais recursos, principalmente no que diz respeito a capacidade de memória interna, é evidente, pela taxa de utilização dos demais recursos do dispositivo, a possibilidade de adição de novas funcionalidades. Sugere-se, por exemplo, a inclusão de parâmetros de ruído. Juntamente com frequência e amplitude, o Loris modela uma amplitude de ruído dos sons analisados, possivelmente produzindo sons mais fiéis, e pode ser utilizado como um fator de multiplicação de um gerador de números pseudoaleatórios, devidamente filtrados em frequência.

Também poderia ser adicionada a capacidade de responder às mensagens MIDI de desativar todas as notas e desativar todo o som, que poderiam ser implementadas utilizando os módulos já existentes do controle de notas, por exemplo. Experimentação pode ser feita com a multiplicação das formas de onda finais dos osciladores por ondas triangulares, além disso, para simulação e síntese de efeitos de *vibratto* nos instrumentos.

Por fim, também existe possibilidade de melhoria na interface do sistema com o usuário. Em se alterando o controlador de memória *flash* para adição da funcionalidade de escrita, por exemplo, pode-se utilizar uma interface *serial* auxiliar, ou até mesmo as próprias mensagens MIDI, para possibilitar a gravação de amostras na *flash* e edição das existentes, através de um *software* dedicado. As bibliotecas do Loris disponibilizam a possibilidade de interface com múltiplas linguagens de programação, o que possibilitaria uma interface gráfica responsiva e bem definida, e o protocolo MIDI prevê mensagens proprietárias dos fabricantes, proporcionando uma alternativa de comunicação, embora lenta, com o dispositivo.

7 REFERÊNCIAS BIBLIOGRÁFICAS

ALTERA CORPORATION. **Cyclone III Device Handbook, volume 1**.

ARNAUDOV, N. a2jmidid. [S.l.], [s.d.]. Disponível em: <http://home.gna.org/a2jmidid/>. Acesso em: 2 dez. 2015.

ASA STANDARDS SECRETARIAT. **Acoustical Terminology ANSI S1.1-1994 (ASA 111-1994)**.

AUDACITY DEVELOPMENT TEAM. Audacity. [S.l.], [s.d.]. Disponível em: <http://audacityteam.org/>. Acesso em: 2 dez. 2015.

BONTE, T.; FROMENT, N.; SCHWEER, W. MuseScore. [S.l.], [s.d.]. Disponível em: <https://musescore.org/>. Acesso em: 2 dez. 2015.

CAPELA, R. N.; BOWN, R.; CANNAM, C. Qsynth - Qt GUI Interface for FluidSynth. [S.l.], [s.d.]. Disponível em: <http://qsynth.sourceforge.net/>. Acesso em: 2 dez. 2015.

COOPER, P. **Perspectives in music theory: an historical-analytical approach**. [S.l.]: Dodd, Mead, 1973.

DANNENBERG, R. B. The Interpretation of MIDI Velocity. *In*: INTERNATIONAL COMPUTER MUSIC CONFERENCE, 2006, San Francisco, CA. **Anais eletrônicos**... San Francisco, CA: The International Computer Music Association, 2006. p. 193–196. Disponível em: <http://www.cs.cmu.edu/~./rbd/papers/velocity-icmc2006.pdf>.

DAVIS, P. JACK Audio Connection Kit. [S.l.], [s.d.]. Disponível em: <http://www.jackaudio.org/>. Acesso em: 2 dez. 2015.

DOURADO, H. A. **Dicionário de termos e expressões da música**. 2. ed. São Paulo: Editora 34, 2008.

FELZMANN, I; LEONARD, P; CLARKE, C. **Development of an audio synthesizer in a FPGA**. Summer project report. Bath, UK: University of Bath, 2014. Não publicado.

FITZ, K. Sound Modelling and Morphing. [S.l.], 28 ago. 2007. Disponível em: <http://www.cerlsoundgroup.org/Kelly/soundmorphing.html>. Acesso em: 18 ago. 2015.

_____. Sound modelling using Loris. [S.l.], 20 mar. 2010. Disponível em: <http://www.cerlsoundgroup.org/Loris/LorisModelingInPython.html>. Acesso em: 28 out. 2015.

FITZ, K.; HAKEN, L. Loris. [S.l.], [s.d.]. Disponível em: <http://sourceforge.net/projects/loris/>. Acesso em: 19 ago. 2015.

FLUIDSYNTH DEVELOPERS. FluidSynth | Software synthesizer based on the SoundFont 2 specifications. [S.l.], [s.d.]. Disponível em: <http://www.fluidsynth.org/>.

FRITTS, L. Musical Instrument Samples. [S.l.], [s.d.]. Disponível em: <http://theremin.music.uiowa.edu/MIS.html>. Acesso em: 20 ago. 2015.

GEORGE, D. MycroPython - Python for microcontrollers. [S.l.], 2014. Disponível em: <http://micropython.org/>. Acesso em: 23 nov. 2015.

GERVAIS, R. **MIDI Power!, Second Edition: The Comprehensive Guide**. 2. ed. [S.l.]: Cengage Learning, 2005.

GRATTON, A. The Hairless MIDI to Serial Bridge. [S.l.], [s.d.]. Disponível em: <http://projectgus.github.io/hairless-midiserial/>. Acesso em: 2 dez. 2015.

HALLIDAY, D.; RESNICK, R. **Fundamentos de física: gravitação, ondas e termodinâmica**. 8ª. ed. Rio de Janeiro: Grupo Gen - LTC, 2008. V. 2.

HUBER, D. M. **The MIDI Manual - A Practical Guide to MIDI in the Project Studio**. 3. ed. [S.l.]: Focal Press/Elsevier, 2007.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **Acoustics -- Standard tuning frequency (Standard musical pitch)**. Disponível em: <http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=3601>. Acesso em: 28 out. 2015.

LEVY, E.; LEVARIE, S. **A Theory of Harmony**. [S.l.]: State University of New York Press, 1985.

LOPEZ-CABANILLAS, P. VMPK. [S.l.], [s.d.]. Disponível em: <http://vmpk.sourceforge.net/>. Acesso em: 2 dez. 2015.

MCADAMS, S.; BREGMAN, A. Hearing Musical Streams. **Computer Music Journal**, dez. 1979. v. 3, n. 4, p. 26–43;60.

MIDI. *In*: **Wikipedia, the free encyclopedia**. [S.l.]: [s.n.], 2015. Disponível em: <https://en.wikipedia.org/wiki/MIDI>. Acesso em: 30 out. 2015.

MIDI MANUFACTURES ASSOCIATION. The Complete MIDI 1.0 Detailed Specification. [S.l.], 2015a. Disponível em: <http://www.midi.org/techspecs/midispec.php>. Acesso em: 29 out. 2015.

_____. General MIDI 1, 2 and Lite Specifications. [S.l.], 2015b. Disponível em: <http://www.midi.org/techspecs/gm.php>. Acesso em: 29 out. 2015.

_____. MIDI Messages. [S.l.], 2015c. Disponível em: <http://midi.org/techspecs/midimessages.php>. Acesso em: 30 out. 2015.

MIDI MANUFACTURES ASSOCIATION; ASSOCIATION OF MUSICAL ELETRONICS INDUSTRY. **MIDI IMPLEMENTATION CHART V2 INSTRUCTIONS**. Disponível em: <http://www.midi.org/techspecs/midi_chart-v2.pdf>. Acesso em: 29 out. 2015.

NISTL, F. Franz Nistl - Klaviermachermeister. [S.l.], 2012. Disponível em:

<http://www.nistl.com/>. Acesso em: 28 out. 2015.

PIERCE, J. Sound Waves and Sine Waves. **Music, cognition, and computerized sound: an introduction to psychoacoustics**. 1. ed. Cambridge, Mass.: MIT Press, 2001.

PLACK, C. J.; OXENHAM, A. J.; FAY, R. R. **Pitch: Neural Coding and Perception**. [S.l.]: Springer, 2005.

RUSS, M. **Sound Synthesis and Sampling**. [S.l.]: Taylor & Francis, 2012.

STMICROELECTRONICS. **STM32F4DISCOVERY Discovery kit with STM32F407VG MCU**. Disponível em: <http://www.st.com/web/catalog/tools/FM116/SC959/SS1532/PF252419?sc=internet/evalboard/product/252419.jsp>. Acesso em: 23 nov. 2015.

SWIFT, A. An introduction to MIDI. [S.l.], [s.d.]. Disponível em: <http://www.doc.ic.ac.uk/~nd/surprise_97/journal/vol1/aps2/>. Acesso em: 30 out. 2015.

SYNTHESIZER. *In*: **Wikipedia, the free encyclopedia**. [S.l.]: [s.n.], 2015. Disponível em: <https://en.wikipedia.org/wiki/Synthesizer>. Acesso em: 28 out. 2015.

TERASIC. **DE0 User Manual**.

_____. Altera DE0 Board. [S.l.], [s.d.]. Disponível em: <http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=165&No=364&PartNo=3>. Acesso em: 23 nov. 2015.

THE EDITORS OF ENCYCLOPÆDIA BRITANNICA. envelope. *In*: **Encyclopædia Britannica Online**. [S.l.]: [s.n.], 2014. Disponível em: <http://www.britannica.com/science/envelope-sound>. Acesso em: 28 out. 2015.

Apêndice A Relações entre notas e frequências

MIDI #	Note	Frequency	MIDI #	Note	Frequency	MIDI #	Note	Frequency
0	Dó$_0$ C	8,1758 Hz	43	Sol$_3$ G	97,9989 Hz	86	Ré$_7$ D	1 174,6591 Hz
1	Dó$_0^{\#}$ D$_b$	8,6620 Hz	44	Sol$_3^{\#}$ A$_b$	103,8262 Hz	87	Ré$_7^{\#}$ E$_b$	1 244,5079 Hz
2	Ré$_0$ D	9,1770 Hz	45	Lá$_3$ A	110,0000 Hz	88	Mi$_7$ E	1 318,5102 Hz
3	Ré$_0^{\#}$ E$_b$	9,7227 Hz	46	Lá$_3^{\#}$ B$_b$	116,5409 Hz	89	Fá$_7$ F	1 396,9129 Hz
4	Mi$_0$ E	10,3009 Hz	47	Si$_3$ B	123,4708 Hz	90	Fá$_7^{\#}$ G$_b$	1 479,9777 Hz
5	Fá$_0$ F	10,9134 Hz	48	Dó$_4$ C	130,8128 Hz	91	Sol$_7$ G	1 567,9817 Hz
6	Fá$_0^{\#}$ G$_b$	11,5623 Hz	49	Dó$_4^{\#}$ D$_b$	138,5913 Hz	92	Sol$_7^{\#}$ A$_b$	1 661,2188 Hz
7	Sol$_0$ G	12,2499 Hz	50	Ré$_4$ D	146,8324 Hz	93	Lá$_7$ A	1 760,0000 Hz
8	Sol$_0^{\#}$ A$_b$	12,9783 Hz	51	Ré$_4^{\#}$ E$_b$	155,5635 Hz	94	Lá$_7^{\#}$ B$_b$	1 864,6550 Hz
9	Lá$_0$ A	13,7500 Hz	52	Mi$_4$ E	164,8138 Hz	95	Si$_7$ B	1 975,5332 Hz
10	Lá$_0^{\#}$ B$_b$	14,5676 Hz	53	Fá$_4$ F	174,6141 Hz	96	Dó$_8$ C	2 093,0045 Hz
11	Si$_0$ B	15,4339 Hz	54	Fá$_4^{\#}$ G$_b$	184,9972 Hz	97	Dó$_8^{\#}$ D$_b$	2 217,4610 Hz
12	Dó$_1$ C	16,3516 Hz	55	Sol$_4$ G	195,9977 Hz	98	Ré$_8$ D	2 349,3181 Hz
13	Dó$_1^{\#}$ D$_b$	17,3239 Hz	56	Sol$_4^{\#}$ A$_b$	207,6523 Hz	99	Ré$_8^{\#}$ E$_b$	2 489,0159 Hz
14	Ré$_1$ D	18,3540 Hz	57	Lá$_4$ A	220,0000 Hz	100	Mi$_8$ E	2 637,0205 Hz
15	Ré$_1^{\#}$ E$_b$	19,4454 Hz	58	Lá$_4^{\#}$ B$_b$	233,0819 Hz	101	Fá$_8$ F	2 793,8259 Hz
16	Mi$_1$ E	20,6017 Hz	59	Si$_4$ B	246,9417 Hz	102	Fá$_8^{\#}$ G$_b$	2 959,9554 Hz
17	Fá$_1$ F	21,8268 Hz	60	Dó$_5$ C	261,6256 Hz	103	Sol$_8$ G	3 135,9635 Hz
18	Fá$_1^{\#}$ G$_b$	23,1247 Hz	61	Dó$_5^{\#}$ D$_b$	277,1826 Hz	104	Sol$_8^{\#}$ A$_b$	3 322,4376 Hz
19	Sol$_1$ G	24,4997 Hz	62	Ré$_5$ D	293,6648 Hz	105	Lá$_8$ A	3 520,0000 Hz
20	Sol$_1^{\#}$ A$_b$	25,9565 Hz	63	Ré$_5^{\#}$ E$_b$	311,1270 Hz	106	Lá$_8^{\#}$ B$_b$	3 729,3101 Hz
21	Lá$_1$ A	27,5000 Hz	64	Mi$_5$ E	329,6276 Hz	107	Si$_8$ B	3 951,0664 Hz
22	Lá$_1^{\#}$ B$_b$	29,1352 Hz	65	Fá$_5$ F	349,2282 Hz	108	Dó$_9$ C	4 186,0090 Hz
23	Si$_1$ B	30,8677 Hz	66	Fá$_5^{\#}$ G$_b$	369,9944 Hz	109	Dó$_9^{\#}$ D$_b$	4 434,9221 Hz
24	Dó$_2$ C	32,7032 Hz	67	Sol$_5$ G	391,9954 Hz	110	Ré$_9$ D	4 698,6363 Hz
25	Dó$_2^{\#}$ D$_b$	34,6478 Hz	68	Sol$_5^{\#}$ A$_b$	415,3047 Hz	111	Ré$_9^{\#}$ E$_b$	4 978,0317 Hz
26	Ré$_2$ D	36,7081 Hz	69	Lá$_5$ A	440,0000 Hz	112	Mi$_9$ E	5 274,0409 Hz
27	Ré$_2^{\#}$ E$_b$	38,8909 Hz	70	Lá$_5^{\#}$ B$_b$	466,1638 Hz	113	Fá$_9$ F	5 587,6517 Hz
28	Mi$_2$ E	41,2034 Hz	71	Si$_5$ B	493,8833 Hz	114	Fá$_9^{\#}$ G$_b$	5 919,9108 Hz
29	Fá$_2$ F	43,6535 Hz	72	Dó$_6$ C	523,2511 Hz	115	Sol$_9$ G	6 271,9270 Hz
30	Fá$_2^{\#}$ G$_b$	46,2493 Hz	73	Dó$_6^{\#}$ D$_b$	554,3653 Hz	116	Sol$_9^{\#}$ A$_b$	6 644,8752 Hz
31	Sol$_2$ G	48,9994 Hz	74	Ré$_6$ D	587,3295 Hz	117	Lá$_9$ A	7 040,0000 Hz
32	Sol$_2^{\#}$ A$_b$	51,9131 Hz	75	Ré$_6^{\#}$ E$_b$	622,2540 Hz	118	Lá$_9^{\#}$ B$_b$	7 458,6202 Hz
33	Lá$_2$ A	55,0000 Hz	76	Mi$_6$ E	659,2551 Hz	119	Si$_9$ B	7 902,1328 Hz
34	Lá$_2^{\#}$ B$_b$	58,2705 Hz	77	Fá$_6$ F	698,4565 Hz	120	Dó$_{10}$ C	8 372,0181 Hz
35	Si$_2$ B	61,7354 Hz	78	Fá$_6^{\#}$ G$_b$	739,9888 Hz	121	Dó$_{10}^{\#}$ D$_b$	8 869,8442 Hz
36	Dó$_3$ C	65,4064 Hz	79	Sol$_6$ G	783,9909 Hz	122	Ré$_{10}$ D	9 397,2726 Hz
37	Dó$_3^{\#}$ D$_b$	69,2957 Hz	80	Sol$_6^{\#}$ A$_b$	830,6094 Hz	123	Ré$_{10}^{\#}$ E$_b$	9 956,0635 Hz
38	Ré$_3$ D	73,4162 Hz	81	Lá$_6$ A	880,0000 Hz	124	Mi$_{10}$ E	10 548,0818 Hz
39	Ré$_3^{\#}$ E$_b$	77,7817 Hz	82	Lá$_6^{\#}$ B$_b$	932,3275 Hz	125	Fá$_{10}$ F	11 175,3034 Hz
40	Mi$_3$ E	82,4069 Hz	83	Si$_6$ B	987,7666 Hz	126	Fá$_{10}^{\#}$ G$_b$	11 839,8215 Hz
41	Fá$_3$ F	87,3071 Hz	84	Dó$_7$ C	1 046,5023 Hz	127	Sol$_{10}$ G	12 543,8540 Hz
42	Fá$_3^{\#}$ G$_b$	92,4986 Hz	85	Dó$_7^{\#}$ D$_b$	1 108,7305 Hz			

Apêndice B Detalhes de implementação do módulo Sintetizador

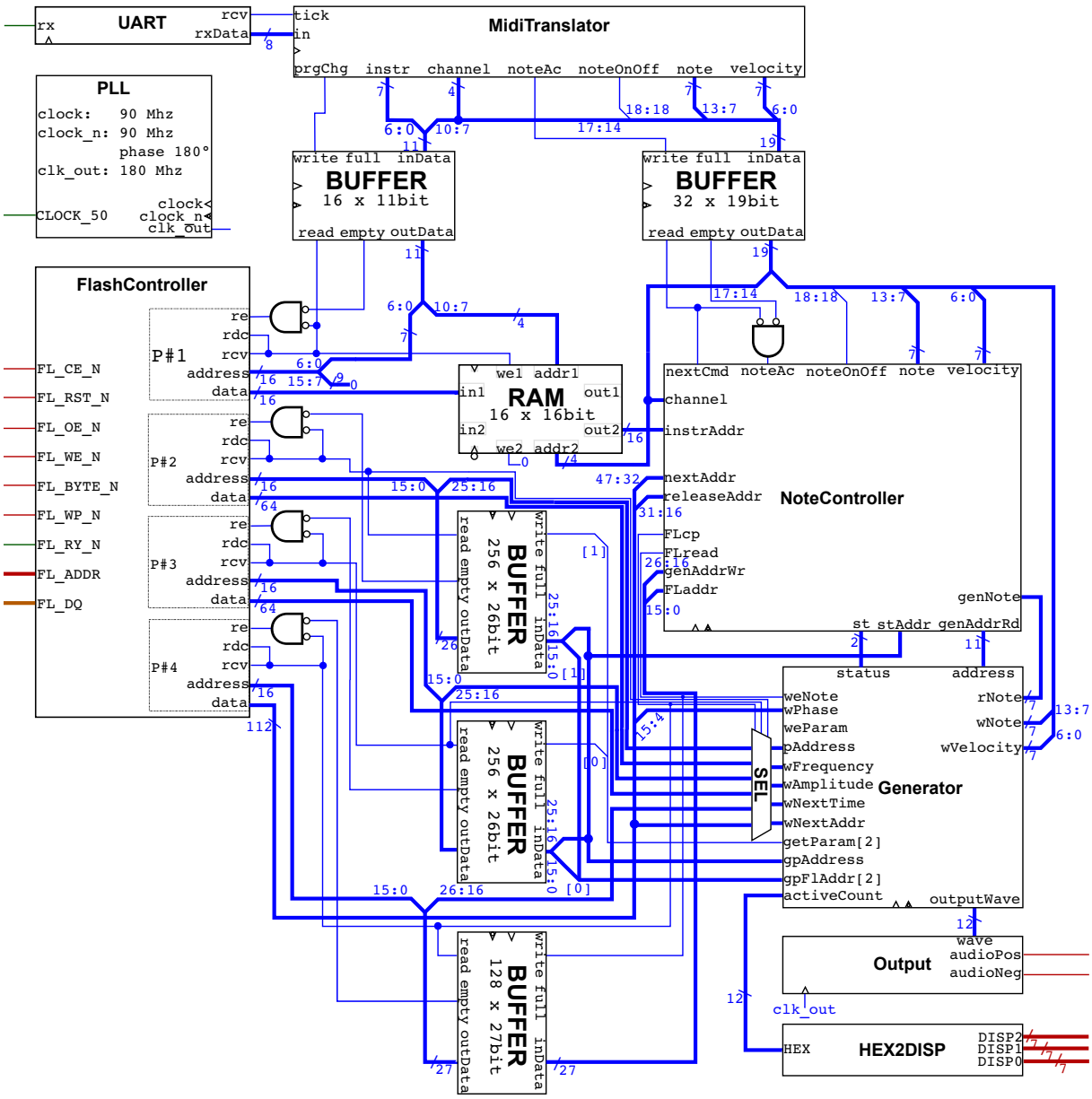


Diagrama de blocos detalhado do sistema

Sinal	Pino	Sinal	Pino	Sinal	Pino
CLOCK_50	PIN_G21	LEDG[9]**	PIN_B1	FL_ADDR[19]	PIN_P1
rx	PIN_AB17	LEDG[8]**	PIN_B2	FL_ADDR[20]	PIN_P3
audioPos	PIN_T12	LEDG[7]**	PIN_C2	FL_ADDR[21]	PIN_R2
audioNeg	PIN_R12	LEDG[6]**	PIN_C1	FL_DQ[0]	PIN_R7
DISP0[0]	PIN_E11	LEDG[5]**	PIN_E1	FL_DQ[1]	PIN_P8

Sinal	Pino	Sinal	Pino	Sinal	Pino
DISP0[1]	PIN_F11	LEDG[4]**	PIN_F2	FL_DQ[2]	PIN_R8
DISP0[2]	PIN_H12	LEDG[3]**	PIN_H1	FL_DQ[3]	PIN_U1
DISP0[3]	PIN_H13	LEDG[2]**	PIN_J3	FL_DQ[4]	PIN_V2
DISP0[4]	PIN_G12	LEDG[1]**	PIN_J2	FL_DQ[5]	PIN_V3
DISP0[5]	PIN_F12	LEDG[0]**	PIN_J1	FL_DQ[6]	PIN_W1
DISP0[6]	PIN_F13	FL_ADDR[0]	PIN_P7	FL_DQ[7]	PIN_Y1
DISP1[0]	PIN_A13	FL_ADDR[1]	PIN_P5	FL_DQ[8]	PIN_T5
DISP1[1]	PIN_B13	FL_ADDR[2]	PIN_P6	FL_DQ[9]	PIN_T7
DISP1[2]	PIN_C13	FL_ADDR[3]	PIN_N7	FL_DQ[10]	PIN_T4
DISP1[3]	PIN_A14	FL_ADDR[4]	PIN_N5	FL_DQ[11]	PIN_U2
DISP1[4]	PIN_B14	FL_ADDR[5]	PIN_N6	FL_DQ[12]	PIN_V1
DISP1[5]	PIN_E14	FL_ADDR[6]	PIN_M8	FL_DQ[13]	PIN_V4
DISP1[6]	PIN_A15	FL_ADDR[7]	PIN_M4	FL_DQ[14]	PIN_W2
DISP2[0]	PIN_D15	FL_ADDR[8]	PIN_P2	FL_DQ[15]	PIN_Y2
DISP2[1]	PIN_A16	FL_ADDR[9]	PIN_N2	FL_BYTE_N	PIN_AA1
DISP2[2]	PIN_B16	FL_ADDR[10]	PIN_N1	FL_CE_N	PIN_N8
DISP2[3]	PIN_E15	FL_ADDR[11]	PIN_M3	FL_OE_N	PIN_R6
DISP2[4]	PIN_A17	FL_ADDR[12]	PIN_M2	FL_RST_N	PIN_R1
DISP2[5]	PIN_B17	FL_ADDR[13]	PIN_M1	FL_RY	PIN_M7
DISP2[6]	PIN_F14	FL_ADDR[14]	PIN_L7	FL_WE_N	PIN_P4
Buffer0.full*	PIN_D13	FL_ADDR[15]	PIN_L6	FL_WP_N	PIN_T3
Buffer2.full*	PIN_B15	FL_ADDR[16]	PIN_AA2	_reset***	PIN_F1
Buffer2.full*	PIN_A18	FL_ADDR[17]	PIN_M5		
Buffer3.full*	PIN_G16	FL_ADDR[18]	PIN_M6		

* Buffer0: *Buffer* entre controle de notas e *flash*;

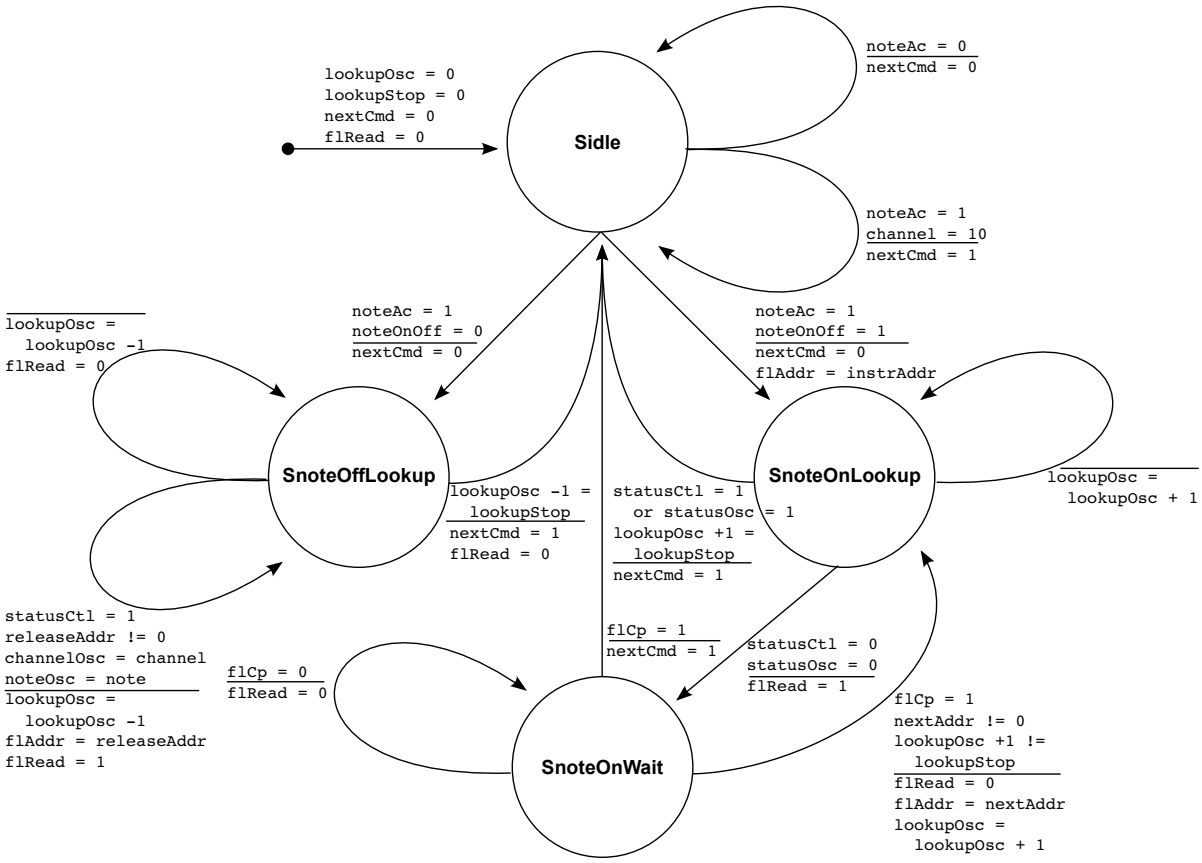
Buffer1 e Buffer2: *Buffers* entre gerador e *flash*;

Buffer3: *Buffer* entre tradutor MIDI e controle de notas.

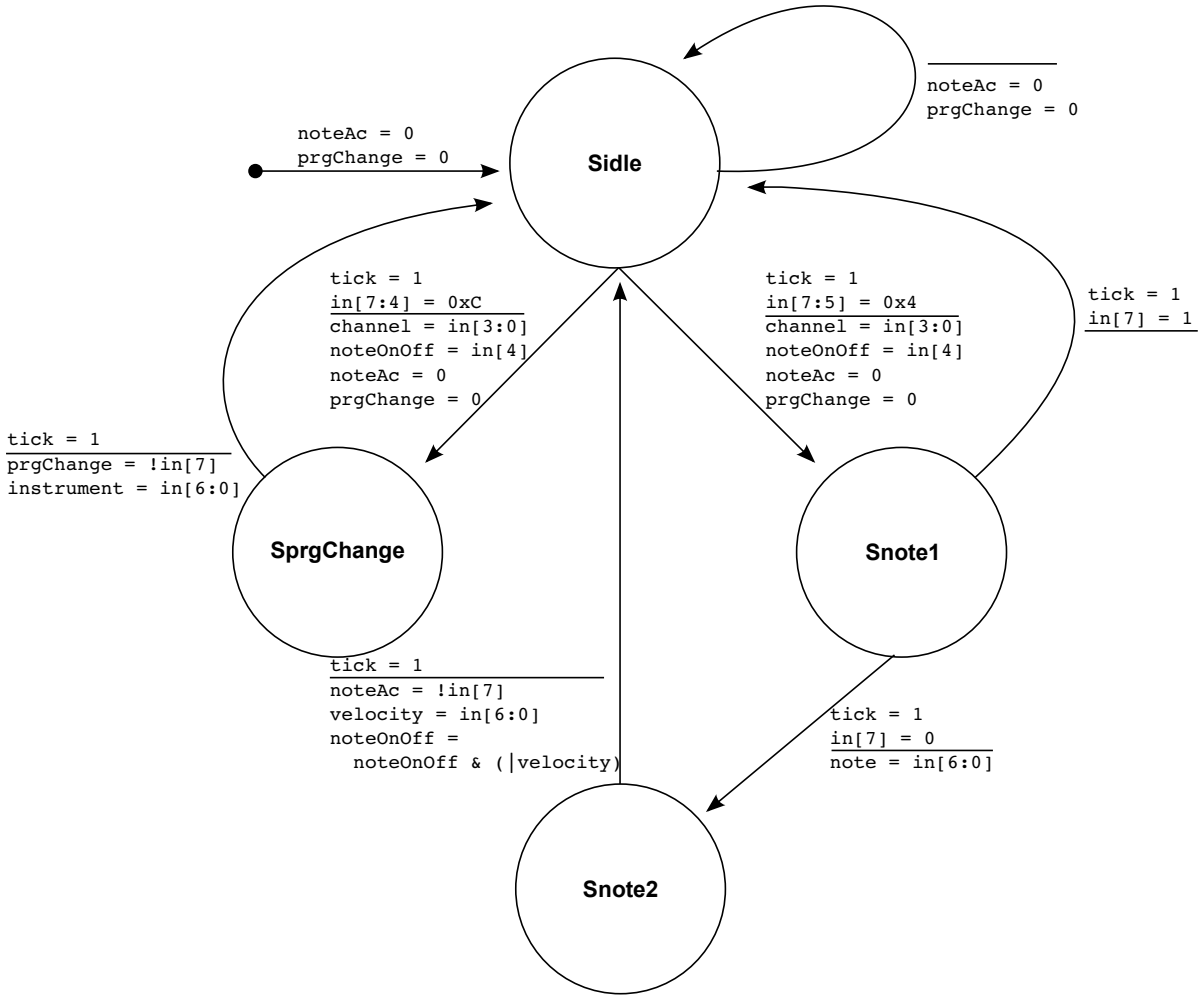
** LEDs conectados a *outputWave* para representação de amplitude.

*** Botão de reset síncrono do sistema.

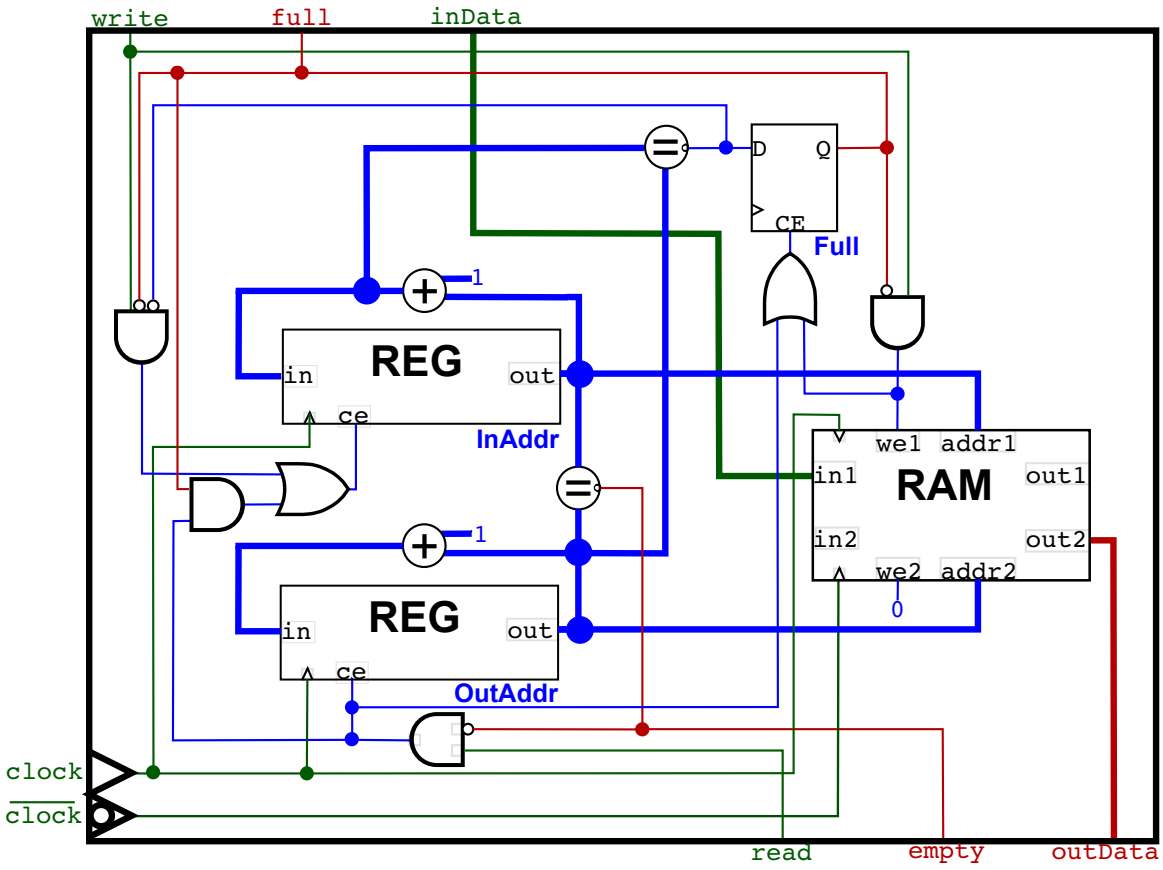
Apêndice C Diagrama de estados do controle de notas



Apêndice D Diagrama de estados do tradutor MIDI



Apêndice E Diagrama de blocos do módulo de *buffer* circular



Anexo A Lista de instrumentos General MIDI Level 1 (GM1)

PC#	Instrument Name	PC#	Instrument Name	PC#	Instrument Name
1.	Acoustic Grand Piano	44.	Contrabass	87.	Lead 7 (fifths)
2.	Bright Acoustic Piano	45.	Tremolo Strings	88.	Lead 8 (bass + lead)
3.	Electric Grand Piano	46.	Pizzicato Strings	89.	Pad 1 (new age)
4.	Honky-tonk Piano	47.	Orchestral Harp	90.	Pad 2 (warm)
5.	Electric Piano 1	48.	Timpani	91.	Pad 3 (polysynth)
6.	Electric Piano 2	49.	String Ensemble 1	92.	Pad 4 (choir)
7.	Harpsichord	50.	String Ensemble 2	93.	Pad 5 (bowed)
8.	Clavi	51.	SynthStrings 1	94.	Pad 6 (metallic)
9.	Celesta	52.	SynthStrings 2	95.	Pad 7 (halo)
10.	Glockenspiel	53.	Choir Aahs	96.	Pad 8 (sweep)
11.	Music Box	54.	Voice Oohs	97.	FX 1 (rain)
12.	Vibraphone	55.	Synth Voice	98.	FX 2 (soundtrack)
13.	Marimba	56.	Orchestra Hit	99.	FX 3 (crystal)
14.	Xylophone	57.	Trumpet	100.	FX 4 (atmosphere)
15.	Tubular Bells	58.	Trombone	101.	FX 5 (brightness)
16.	Dulcimer	59.	Tuba	102.	FX 6 (goblins)
17.	Drawbar Organ	60.	Muted Trumpet	103.	FX 7 (echoes)
18.	Percussive Organ	61.	French Horn	104.	FX 8 (sci-fi)
19.	Rock Organ	62.	Brass Section	105.	Sitar
20.	Church Organ	63.	SynthBrass 1	106.	Banjo
21.	Reed Organ	64.	SynthBrass 2	107.	Shamisen
22.	Accordion	65.	Soprano Sax	108.	Koto
23.	Harmonica	66.	Alto Sax	109.	Kalimba
24.	Tango Accordion	67.	Tenor Sax	110.	Bag pipe
25.	Acoustic Guitar (nylon)	68.	Baritone Sax	111.	Fiddle
26.	Acoustic Guitar (steel)	69.	Oboe	112.	Shanai
27.	Electric Guitar (jazz)	70.	English Horn	113.	Tinkle Bell
28.	Electric Guitar (clean)	71.	Bassoon	114.	Agogo
29.	Electric Guitar (muted)	72.	Clarinet	115.	Steel Drums
30.	Overdriven Guitar	73.	Piccolo	116.	Woodblock
31.	Distortion Guitar	74.	Flute	117.	Taiko Drum
32.	Guitar harmonics	75.	Recorder	118.	Melodic Tom
33.	Acoustic Bass	76.	Pan Flute	119.	Synth Drum
34.	Electric Bass (finger)	77.	Blown Bottle	120.	Reverse Cymbal
35.	Electric Bass (pick)	78.	Shakuhachi	121.	Guitar Fret Noise
36.	Fretless Bass	79.	Whistle	122.	Breath Noise
37.	Slap Bass 1	80.	Ocarina	123.	Seashore
38.	Slap Bass 2	81.	Lead 1 (square)	124.	Bird Tweet
39.	Synth Bass 1	82.	Lead 2 (sawtooth)	125.	Telephone Ring
40.	Synth Bass 2	83.	Lead 3 (calliope)	126.	Helicopter
41.	Violin	84.	Lead 4 (chiff)	127.	Applause
42.	Viola	85.	Lead 5 (charang)	128.	Gunshot
43.	Cello	86.	Lead 6 (voice)		

Extraído de: MIDI MANUFACTURERS ASSOCIATION. **General MIDI Level 1 Sound Set**. Disponível em: <http://www.midi.org/techspecs/gm1sound.php>. Acesso em: 6 dez. 2015.

Anexo B Lista de mensagens MIDI

Status D7----D0	Data Byte(s) D7----D0	Description
Channel Voice Messages [nnnn = 0-15 (MIDI Channel Number 1-16)]		
1000nnnn	0kkkkkkk 0vvvvvvv	Note Off event. This message is sent when a note is released (ended). (kkkkkkk) is the key (note) number. (vvvvvvv) is the velocity.
1001nnnn	0kkkkkkk 0vvvvvvv	Note On event. This message is sent when a note is depressed (start). (kkkkkkk) is the key (note) number. (vvvvvvv) is the velocity.
1010nnnn	0kkkkkkk 0vvvvvvv	Polyphonic Key Pressure (Aftertouch). This message is most often sent by pressing down on the key after it "bottoms out". (kkkkkkk) is the key (note) number. (vvvvvvv) is the pressure value.
1011nnnn	0ccccccc 0vvvvvvv	Control Change. This message is sent when a controller value changes. Controllers include devices such as pedals and levers. Controller numbers 120-127 are reserved as "Channel Mode Messages". (ccccccc) is the controller number (0-119). (vvvvvvv) is the controller value (0-127).
1100nnnn	0ppppppp	Program Change. This message sent when the patch number changes. (ppppppp) is the new program number.
1101nnnn	0vvvvvvv	Channel Pressure (After-touch). This message is most often sent by pressing down on the key after it "bottoms out". This message is different from polyphonic after-touch. Use this message to send the single greatest pressure value (of all the current depressed keys). (vvvvvvv) is the pressure value.
1110nnnn	0lllllll 0mmmmmmm	Pitch Bend Change. 0mmmmmmm This message is sent to indicate a change in the pitch bender (wheel or lever, typically). The pitch bender is measured by a fourteen bit value. Center (no pitch change) is 2000H. Sensitivity is a function of the transmitter. (lllllll) are the least significant 7 bits. (mmmmmm) are the most significant 7 bits.
Channel Mode Messages (See also Control Change, above)		
1011nnnn	0ccccccc 0vvvvvvv	Channel Mode Messages. This the same code as the Control Change (above), but implements Mode control and special message by using reserved controller numbers 120-127. The commands are:
		All Sound Off. When All Sound Off is received all oscillators will turn off, and their volume envelopes are set to zero as soon as possible. c = 120, v = 0: All Sound Off
		Reset All Controllers. When Reset All Controllers is received, all controller values are reset to their default values. (See specific Recommended Practices for defaults). c = 121, v = x: Value must only be zero unless otherwise allowed in a specific Recommended Practice.
		Local Control. When Local Control is Off, all devices on a given channel will respond only to data received over MIDI. Played data, etc. will be ignored. Local Control On restores the functions of the normal controllers. c = 122, v = 0: Local Control Off c = 122, v = 127: Local Control On
		All Notes Off. When an All Notes Off is received, all oscillators will turn off. c = 123, v = 0: All Notes Off (See text for description of actual mode commands.) c = 124, v = 0: Omni Mode Off c = 125, v = 0: Omni Mode On c = 126, v = M: Mono Mode On (Poly Off) where M is the number of channels (Omni Off) or 0 (Omni On) c = 127, v = 0: Poly Mode On (Mono Off) (Note: These four messages also cause All Notes Off)
System Common Messages		
11110000	0iiiiiii [0iiiiiii 0iiiiiii] 0ddddddd --- --- 0ddddddd 11110111	System Exclusive. This message type allows manufacturers to create their own messages (such as bulk dumps, patch parameters, and other non-spec data) and provides a mechanism for creating additional MIDI Specification messages. The Manufacturer's ID code (assigned by MMA or AMEI) is either 1 byte (0iiiiiii) or 3 bytes (0iiiiiii 0iiiiiii 0iiiiiii). Two of the 1 Byte IDs are reserved for extensions called Universal Exclusive Messages, which are not manufacturer-specific. If a device recognizes the ID code as its own (or as a supported Universal message) it will listen to the rest of the message (0ddddddd). Otherwise, the message will be ignored. (Note: Only Real-Time messages may be interleaved with a

Status D7----D0	Data Byte(s) D7----D0	Description
		System Exclusive.)
11110001	0nnndddd	MIDI Time Code Quarter Frame. nnn = Message Type dddd = Values
11110010	0lllllll 0mmmmmmm	Song Position Pointer. This is an internal 14 bit register that holds the number of MIDI beats (1 beat= six MIDI clocks) since the start of the song. l is the LSB, m the MSB.
11110011	0sssssss	Song Select. The Song Select specifies which sequence or song is to be played.
11110100		Undefined. (Reserved)
11110101		Undefined. (Reserved)
11110110		Tune Request. Upon receiving a Tune Request, all analog synthesizers should tune their oscillators.
11110111		End of Exclusive. Used to terminate a System Exclusive dump (see above).
System Real-Time Messages		
11111000		Timing Clock. Sent 24 times per quarter note when synchronization is required (see text).
11111001		Undefined. (Reserved)
11111010		Start. Start the current sequence playing. (This message will be followed with Timing Clocks).
11111011		Continue. Continue at the point the sequence was Stopped.
11111100		Stop. Stop the current sequence.
11111101		Undefined. (Reserved)
11111110		Active Sensing. This message is intended to be sent repeatedly to tell the receiver that a connection is alive. Use of this message is optional. When initially received, the receiver will expect to receive another Active Sensing message each 300ms (max), and if it does not then it will assume that the connection has been terminated. At termination, the receiver will turn off all voices and return to normal (non- active sensing) operation.
11111111		Reset. Reset all receivers in the system to power-up status. This should be used sparingly, preferably under manual control. In particular, it should not be sent on power-up.

Extraído de: MIDI MANUFACTURERS ASSOCIATION. **MIDI Messages**. Disponível em: <http://www.midi.org/techspecs/midimessages.php>. Acesso em: 6 dez. 2015.

Anexo C Detalhes da amostra de piano obtida para teste

Instrument	Piano
Model	Steinway & Sons model B
Performer	Evan Mazunik
Date	November 5 & 27, 2001
Location	2017 Voxman Music Building
Technician	Michael Cash
Distance	Left mic 8" above center bass strings Right mic 8" above center treble strings
Microphone	Neumann KM 84
Mixer	Mackie 1402-VLZ
Recorder	Panasonic SV-3800 DAT
Format	16-bit, 44.1 kHz, stereo
Comments	stereo, non-anechoic recording

Extraído de: UNIVERSITY OF IOWA ELECTRONIC MUSIC STUDIOS. **Piano**. Disponível em: <http://theremin.music.uiowa.edu/MISpiano.html>. Acesso em: 18 dez. 2015.