Projeto de Detectores DTMF

Andre Felippe Weber, Gabriel Cozer Cantu e
Maria Luiza Theisges *

17 de dezembro de 2016

Resumo

Este relatório apresenta os resultados da construção de um detector DTMF no Matlab, a implementação do sistema no Simulink e também como gerar o VHDL a partir do Matlab.

Palavras-chaves: DTMF, Filtros, Matlab, Simulink, VHDL.

1 Introdução

A modernização dos sistemas telefônicos fez com que os circuitos de discagem fossem modificados, deixando de serem relés eletros-mecânicos para serem dispositivos de estado sólido. Então, para reutilizar a estrutura existente, os tons do teclado começaram a ser transmitidos no mesmo par de fios usado para transportar o sinal de voz, na frequência de 300 a 3400 Hz [1].

O DTMF (do inglês *Dual-Tone-Multiple-Frequency*), é a soma de duas frequências (linha e coluna) para representar um determinado número do teclado. Esse conjunto de frequências é enviado e decodificado pela central telefônica. Porém, o problema estava em decodificar essas frequências. A solução encontrada foi a de criar um detector DTMF que separa as frequências vinda do telefone chamador, com o intuito de descobrir a linha e a coluna, encontrando o número teclado.

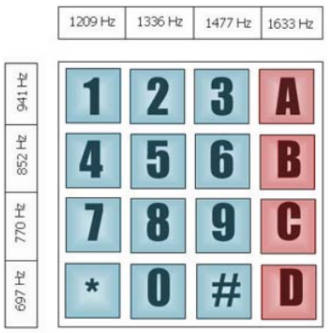

Figura 1 – Combinação de frequências do teclado DTMF completo [1]

Nesse artigo é descrito a implementação de um detector DTMF através do Matlab, bem como a sua implementação em Simulink e finalmente a conversão do cógido para VHDL. A seção 2 descreve os detalhes da implementação em Matlab, a seção 3 descreve a implementação em Simulink, e a seção 4 descreve a conversão para VHDL. Esse trabalho foi desenvolvido na disciplina de Processamento de Sinais Digitais do curso de Engenharia de Telecomunicações, do Instituto Federal de Santa Catarina- Campus São José.

*Alunos do curso de Engenharia de Telecomunicações - IFSC

2 Implementação do detector DTMF em Matlab

A especificação inicial do projeto é a de reconhecer, ao fim do sistema implementado, as teclas 0, 1, 3 e #. Outra especificação feita é de utilizar filtros passa-baixa e passa-alta do tipo Butterworth, e o filtro passa-faixa do tipo FIR.

2.1 Regras do DTMF

Para fazer a detecção correta dos números digitados pelo usuário, o sistema DTMF usa duas frequências. As frequências altas (1209 Hz, 1336 Hz, 1477 Hz e 1633 Hz) correspondem às colunas do teclado, e as frequências baixas (697 Hz, 770 Hz, 852 Hz e 941 Hz) correspondem às linhas, como pode ser observado na Figura 1. Assim, quando o usuário digita o número 1, por exemplo, as frequências 941 Hz e 1209 Hz serão detectadas pelo sistema DTMF.

2.2 Sinal DTMF no Matlab

O *software* Audacity foi utilizado para produzir sinais DTMF com uma frequência de amostragem de 44100 Hz. Alguns dos sinas gerados eram sem ruído e outros com inserção de ruído branco com até 80% da amplitude total do sinal DTMF original. Esses sinais foram lidos pelo sistema criado, com o intuito de serem filtrados e para que fosse possível a detecção de cada um dos números teclados.

2.3 Sistema DTMF no Matlab

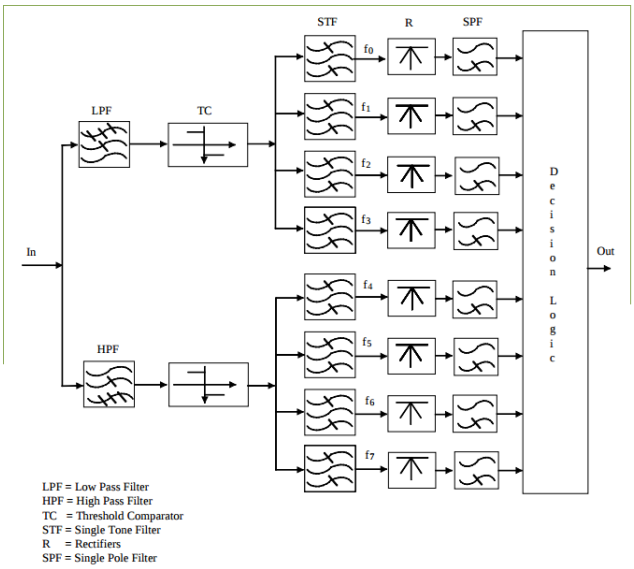

Figura 2 – Sistema DTMF

Antes de implementar o circuito da Figura 2, o sinal foi passado por um filtro IIR (*Infinite impulse response*) passa-baixa, do tipo Butterworth, com ordem 2, e frequência de corte de 2000 Hz (frequência um pouco maior do que a utilizada pelo circuito DTMF - 1633 Hz). Isto é feito para evitar que frequências indesejadas perturbem o funcionamento correto do sistema. O sinal original, com frequência de amostragem de 44100 Hz, é sub amostrado, resultando em um sinal com frequência de amostragem de 4000 Hz, que é o suficiente para que o sistema continue a interpretar o sinal corretamente.

Na Figura 2 é possível observar seis colunas de blocos, este relatório irá se referir a estas colunas para facilitar a explicação da implementação do circuito. O primeiro passo após inserir o sinal no sistema é dividir a frequências baixas das frequências altas. Isto foi feito, neste projeto, a

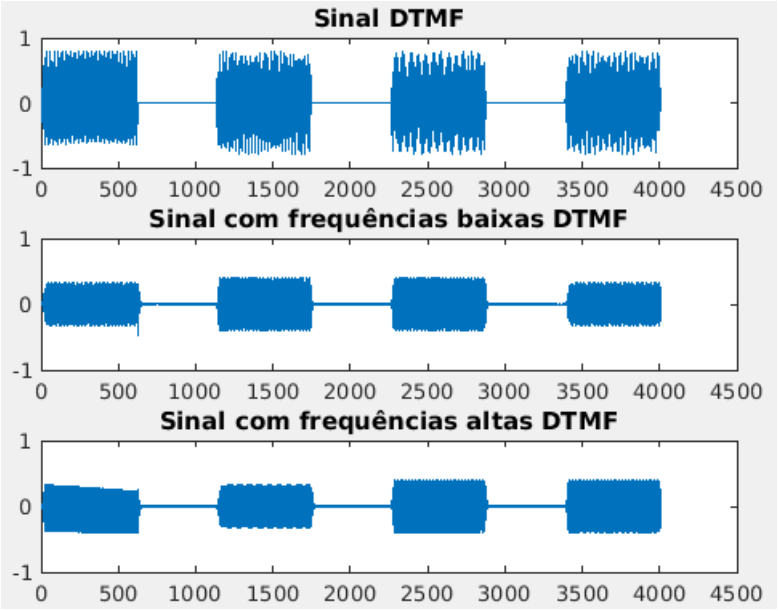

Figura 3 – Sinal DTMF de entrada e detecção das frequências baixas e altas

partir do uso de dois filtros IIR, do tipo Butterworth. O primeiro filtro, que obterá as frequências baixas, deve ser um passa-baixa com frequência de corte de 960 Hz[2]. O segundo, que é utilizado para obter as frequências altas, é uma passa-alta com frequência de corte de 1190 Hz[2].

A segunda coluna do sistema refere-se aos limitadores de amplitude do sinal de entrada. Ele é utilizado para que sinais que estejam dentro de uma faixa inferior e superior sejam excluídos da amostra original, por serem fracos. Porém não houve necessidade da implementação destes limitadores neste projeto, visto que os resultados foram satisfatórios sem o uso deles.

A terceira coluna é referente aos filtos FIR (do inglês *Finite Impulse Response*) que devem separar cada uma das oito frequências (linhas e colunas), existentes em um sistema DTMF. Os filtros passa-faixa, do tipo FIR, com janela de Hamming, tiveram suas frequências de passagem e de corte calculadas conforme as normas do ETSI (*The European Telecommunications Standards Institute*) [2], que recomenda deixar uma folga de +-(1,5% + 2 Hz) em relação a frequência nominal de cada linha e coluna. O resultado destes cálculos podem ser vistos na Tabela 1. As frequências calculadas são divididas pela metade da frequência de amostragem, pois o que se quer é apenas a parte positiva do espectro de frequências.

O objetivo, para esse projeto, era somente detectar as teclas 0, 1, 3 e #. Para isso, foi necessário implementar apenas cinco filtros, referentes às linhas 1 e 4 e às colunas 1, 2 e 3 de um teclado DTMF (Figura 1).

Tabela 1 – Cálculo das Frequências de Passagem (Fp) e *Stop* (Fs)

Frequências (Hz)	Fp1 (Hz)	Fp2 (Hz)	Fs1 (Hz)	Fs2 (Hz)
697	688,5	709,455	627,3	766,7
941	928,885	957,115	846,9	1035,1
1209	1192,865	1229,135	1088,1	1329,9
1336	1317,96	1358,04	1202,4	1469,6
1477	1456,845	1501,155	1329,3	1624,7

Na quarta coluna do sistema estão os retificadores que são implementados através do uso da função abs do Matlab, que retorna o valor absoluto de um número. Isto é feito para que haja uma representação do sinal mais forte e somente com valores positivos.

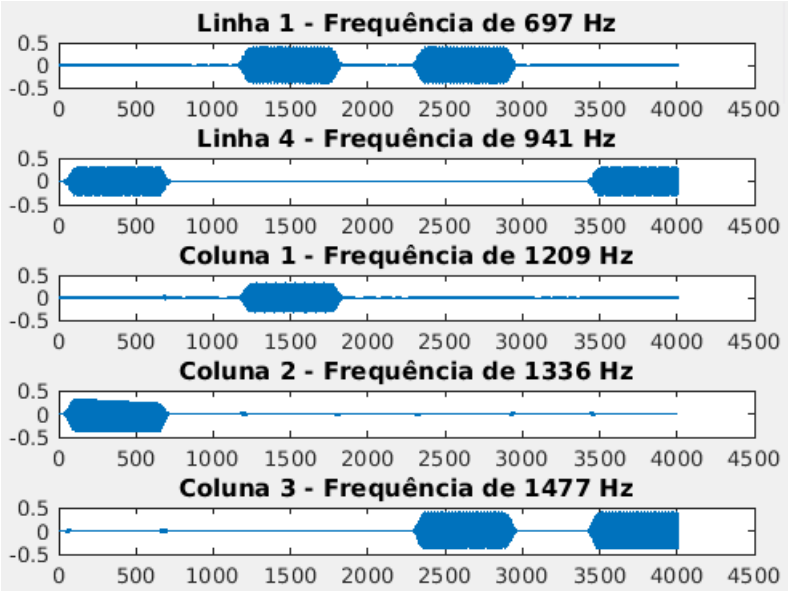

Figura 4 – Detecção das frequências das linhas 1 e 4 e colunas 1, 2 e 3

Este sinal é então repassado para a quinta coluna representada na Figura 2 composta por filtros IIR de polo único, do tipo Butterworth, que tem como objetivo detectar a envoltória dos sinais detectados na terceira coluna. Desta forma é possível traçar um limiar do que deve ser considerado como uma frequência resultante de um algarismo digitado pelo usuário.

Finalmente, foi implementada uma lógica para fazer a detecção dos algarismos digitados pelo usuário. Assim, verifica-se as frequências presentes em um determinado instante, se duas delas formarem uma par que represente uma posição do teclado DTMF, então o algarismo desta posição será armazenado em um vetor do sistema.

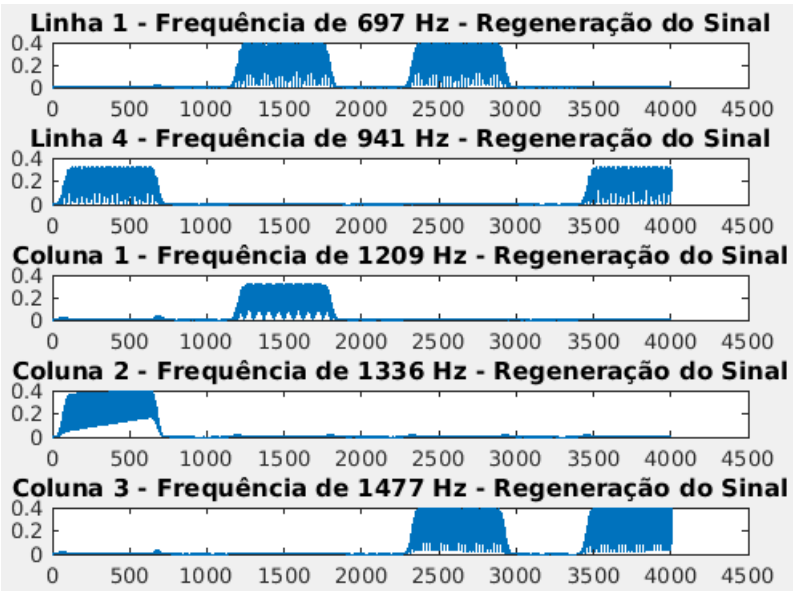

Figura 5 – Frequências detectadas e retificadas

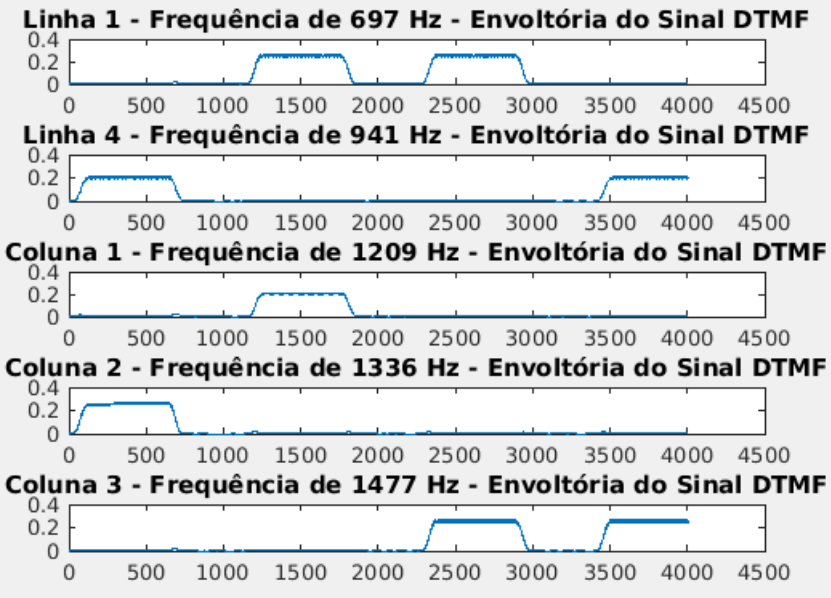

Figura 6 – Envoltória dos sinais detectados pelo sistema DTMF

3 Implementação do detector DTMF em Simulink

Após ter a implementação feita no Matlab, o passo seguinte foi a de fazer a implementação em Simulink. Para isso, foi utilizado o Simulink Library do Matlab versão 15a. O detector DTMF, após a implementação, ficou de acordo com a Figura 8.

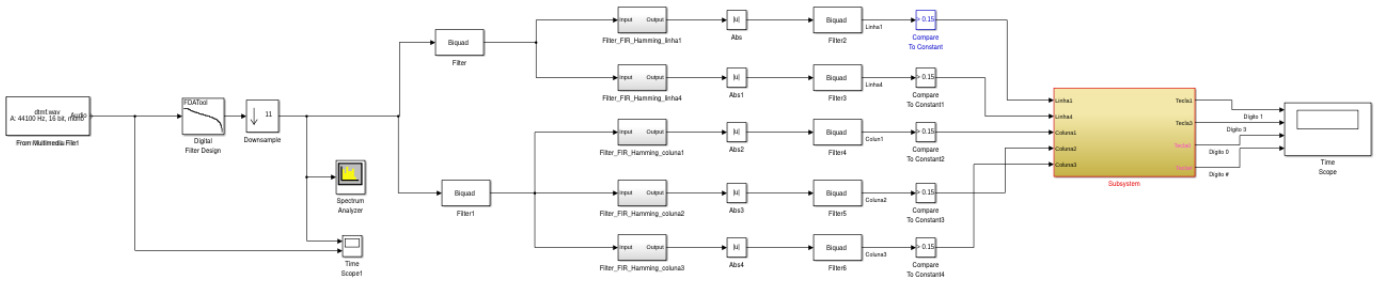

Figura 7 – Combinação de frequências do teclado DTMF completo [1]

No primeiro bloco *From Multimidia File 1* é onde o áudio, gerado através do *software* Audicity, é carregado. Ele corresponde a entrada do sistema, e tem como função representar o usuário teclando.

O segundo bloco, *Digital Filter Design*, corresponde à um filtro passa-baixa, do tipo Butterworth, com frequência de corte de 2200 Hz. Ele tem a função de eliminar frequências mais altas, como ruídos, que possam estar contidas no sinal de entrada. Com isso, permite uma melhor detecção das frequências, sendo o número reconhecido com mais facilidade.

O bloco *Downsample* realiza a subamostragem do sinal de entrada, fazendo com o que a frequência de amostragem inicial de 44100 Hz fique em 4000 Hz, o que é suficiente para fazer o reconhecimento dos números teclados.

O *Time Scope 1* e *Spectrum Analyzer* foram utilizados para mostrarem o sinal de entrada na tela. A Figura **??** mostra, no primeiro escopo de tempo, as frequências de entrada dos números 1, 2, 3, 4, 5, 6, 7, 8, 9, *, 0, #. No segundo escopo de tempo, pode-se ver as frequências já filtradas pelo

primeiro filtro e amostradas pelo Downsample. Na Figura 9 observa-se as frequências de entradas, já filtradas e amostradas, através do *Spectrum Analyzer*.

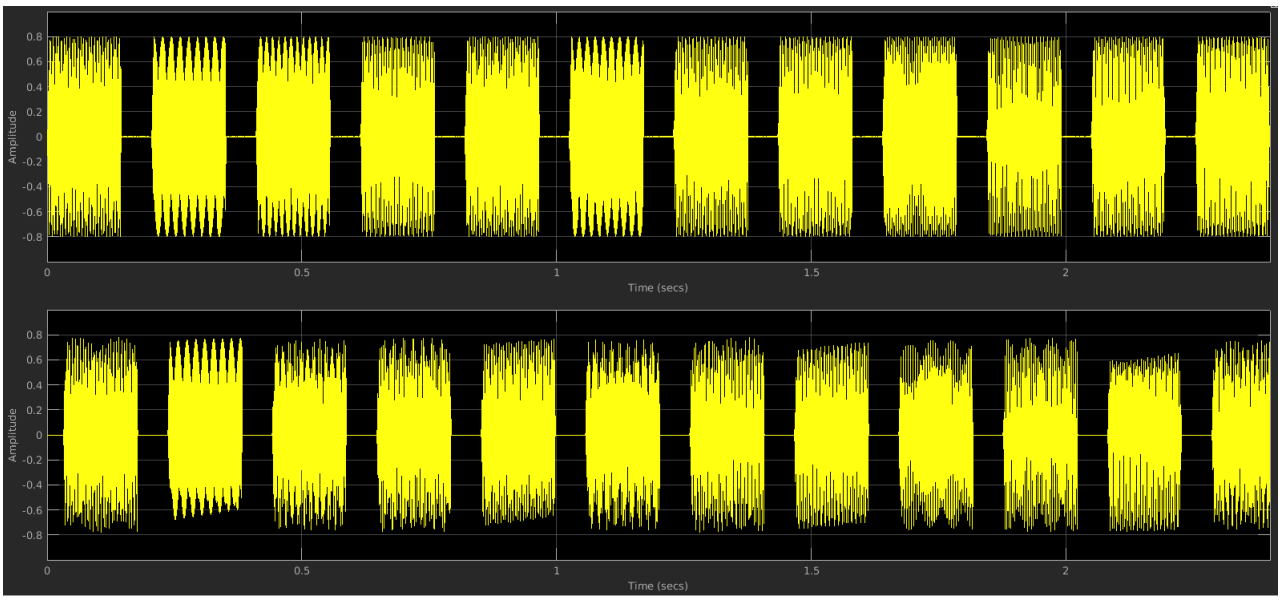

Figura 8 – Frequências de entrada do sistema

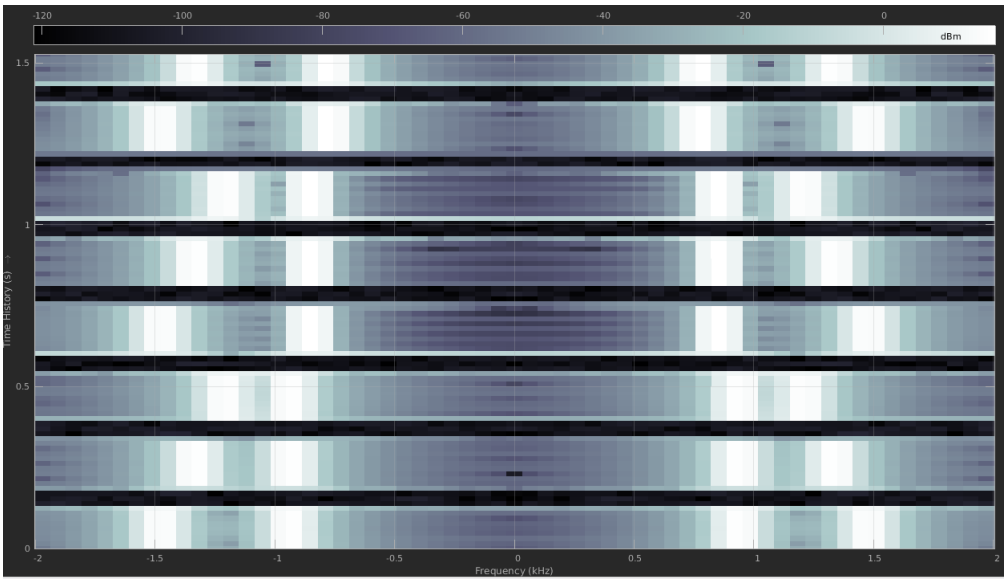

Figura 9 – Frequências de entrada do sistema após filtragem e amostragem

O bloco *Biquad* é um filtro passa-baixa, do tipo Butterworth, com ordem 10, e com frequência de corte de 960 Hz. Esse filtro foi utilizado para separar as frequências das linhas do teclado. O bloco *Biquad 1* é um filtro passa-alta, do tipo Butterworth, com ordem 10, e frequência de corte de 1190 Hz. Esse filtro foi utilizado para separar as frequências das colunas do teclado.

Após separar as frequências das linhas e das colunas, foi necessário utilizar filtros para reconhecer qual número foi teclado. Os blocos *Filter_FIR_Hamming_linha1*, *Filter_FIR_Hamming_linha4*, *Filter_FIR_Hamming_coluna1*, *Filter_FIR_Hamming_coluna2*, e *Filter_FIR_Hamming_coluna3* exercem essa função. Esses blocos são filtros passa-faixa, do tipo FIR, com janela de Hamming, e com as frequências de

Os blocos de *Abs* a *Abs4* representam os retificadores do sistema. Quando o sinal passa por esses retificadores, toda a parte negativa torna-se positiva. Após a passagem por cada um desses

blocos de *Abs*, é possível ter apenas a envoltória dos sinais.

Os filtros *Biquad* (*Filter 2* a *Filter 6*), são de pólo único, e tem a função de filtrar as frequências necessárias para o reconhecimento dos números teclados. E também, normalizam a frequência entre 0 e 1.

Os blocos de *Compare To Constant* a *Compare To Constant 4*, realizam a seguinte verificação:

- Se houver uma frequência normalizada acima de 0,15, então existe um número teclado na determinada linha, ou coluna, verificada naquele instante;
- Caso contrário, nenhum número foi teclado naquela determinada linha ou coluna em questão.

No bloco *Subsystem* é onde está a lógica de reconhecimento do número teclado. A Figura 10 mostra a estrutura interna do bloco.

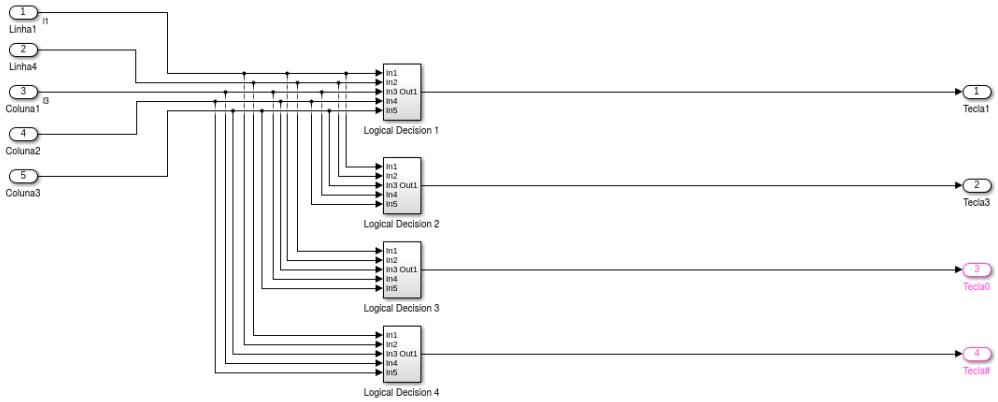

Figura 10 – Lógica para reconhecimento dos números teclados

Cada uma das entradas de linhas e colunas vai para os blocos de *Logical Decision*. Dentro de cada um desses blocos há uma lógica implementada, podendo ser vista através da Figura 11.

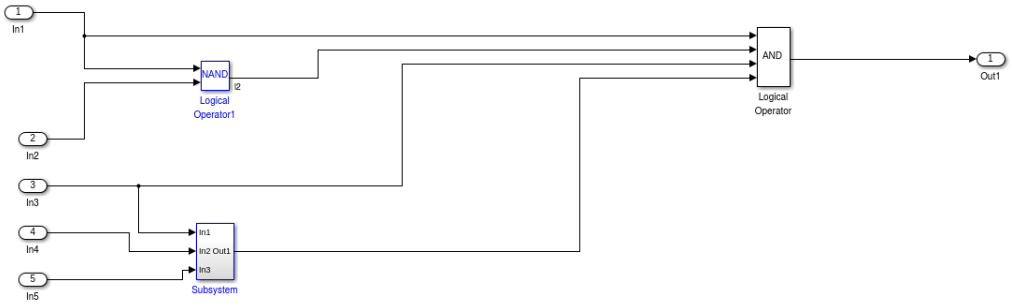

Figura 11 – Lógica para reconhecimento das frequências das linhas do teclado

Para fazer o reconhecimento das frequências existentes nas colunas foi implementada a lógica que pode ser vista conforme a Figura 12.

Por último, mais um *Time Scope* foi colocado, dessa vez no fim do sistema, para verificar as frequências de saída. Com isso, é possível reconhecer os números teclados, certificando-se que o sistema funciona de forma correta. A figura 13 mostra o resultado final.

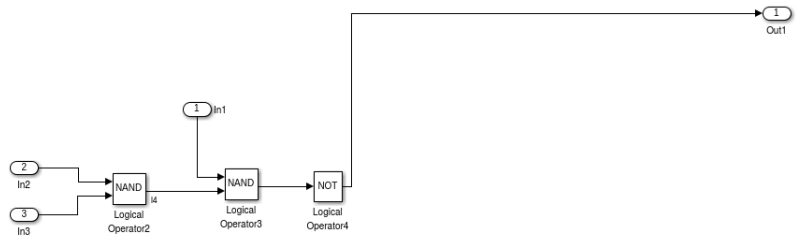

Figura 12 – Lógica para reconhecimento das frequências das colunas do teclado

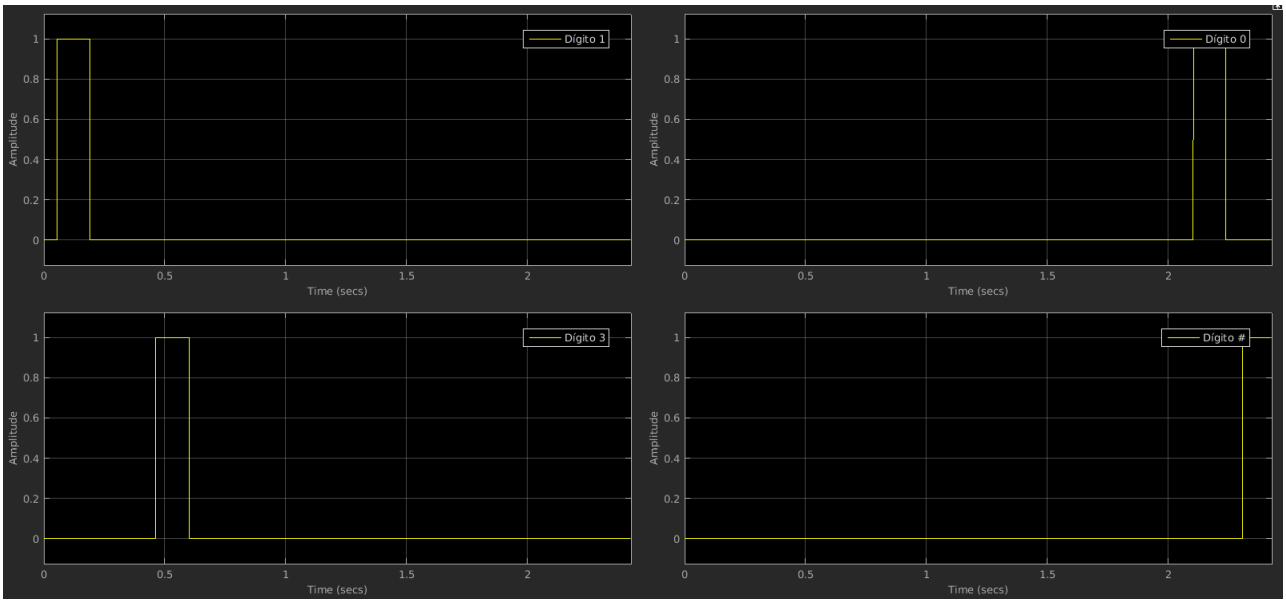

Figura 13 – Resultado final do sistema de detector DTMF

4 Implementação do detector DTMF em VHDL

A ferramenta *HDL Coder* permite realizar a portabilidade e sintetização de VHDL e Verilog, através de códigos Matlab ou sistemas implementados no Simulink. A geração do *HDL Code* pode ser utilizada para a programação de FPGAs ou protótipos ASIC [3].

A terceira etapa desse projeto é a conversão do sistema de detecção DTMF, implementado no *Simulink*, para HDL Coder, onde o sistema será convertido em uma série de códigos VHDL. Para auxiliar na portabilidade, o *Simulink* dispõe uma aplicação chamada *WorkFlow Advisor* que, dentre as suas funções, realiza:

- Verificação de compatibilidade entre o modelo gerado no Simulink e o HDL Coder;
- Geração do código VHDL, juntamente com o *testbench*;
- Realização de análise e síntese;
- Levantamento dos recursos utilizados para implementação em hardware.

Apesar de ser uma ferramenta eficaz na geração de códigos HDL, através de modelos *Simulink*, não são todas as funcionalidades do *Simulink* que podem ser convertidas em *HDL Code*. O sistema implementado, e mostrado na Figura 8, sofreu algumas alterações para ser compatível para conversão em códigos VHDL.

O primeiro filtro do sistema, com frequência de corte de 2200 Hz, foi implementado através da ferramenta *fdatool*, porém esta ferramenta não pode ser sintetizada em *HDL Coder*. Para que esse filtro pudesse ser sintetizado, ele foi realizado através do *fdatool*, e sua matriz SOS (do inglês, *Second Order Section*) e os valores escalares foram utilizados para a geração de um filtro Biquad.

Os filtros FIR, também implementados através da ferramenta *fdatool*, foram realizados e importados para o *Simulink*. Porém, os mesmos não são sintetizados para códigos VHDL. Dessa mesma forma, os filtros FIR foram realizados no *fdatool*, e os coeficientes são utilizados para a geração de um Discrete FIR Filter, os quais também podem ser sintetizados em *HDL Coder*.

Os demais componentes do sistema, entre eles o *Downsample*, os retificadores, os comparadores, o analisador de espectro e os escopos de tempo são sintetizados em *HDL Coder*, e não necessitam de alterações.

Realizadas as alterações de compatibilidade, todo o sistema, exceto o *From Multimidia File* e o *Time Scope* final, foi colocado em um único subsistema, afim de deixar o código em VHDL mais organizado.

Porém, a saída do bloco *From Multimidia File* é do tipo *double*, e, quando gerado o código em VHDL, as variáveis as entradas e saídas do sistema recebem tipos Reais. Variáveis de tipo Real não são utilizadas em sistemas que serão implementados em FPGAs, apenas são utilizadas para simulações em máquina. Para tal, na saída do bloco *From Multimidia File*, um bloco de conversão é utilizado, conforme mostra a Figura 14. A função deste bloco é converter um sinal de entrada do tipo *double*, para um sinal de saída do tipo Ponto Fixo, para que na hora da conversão em código VHDL, os tipos de variáveis utilizadas sejam do tipo *Standart Logic*, e com isso aptas a implementação em uma FPGA.

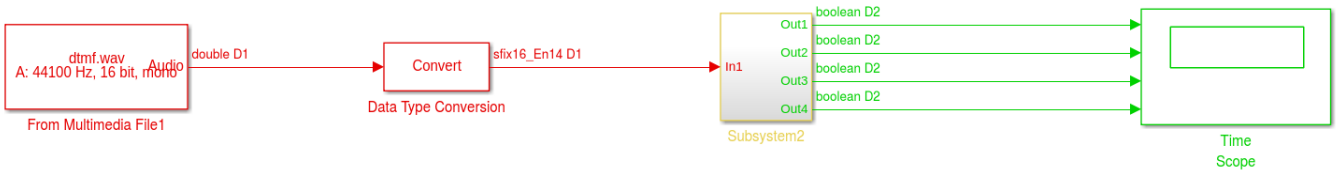

Figura 14 – Sistema de detecção DTMF no Simulink, pronto para ser convertido para HDL Code

A Figura 15 mostra o sistema que está encapsulado dentro do subsistema principal. O primeiro subsistema é referente a primeira filtragem com frequência de corte de 2200 Hz, seguido do processo de *Downsample*. O segundo bloco é referente a dois filtros, um passa-baixas e outro passa-alta, responsável por separar as frequências das linhas do teclado DTMF das frequências das colunas. Os próximos 5 subsistemas realizam a detecção das frequências pré-determinadas, frequências da linha 1, linha 4, coluna 1, coluna 2 e coluna 3. O último subsistema é o responsável pela detecção lógica das frequências DTMF.

Realizando a implementação dessa maneira, o código VHDL principal do Subsistema 2 contará apenas com a instanciação de componentes, referentes aos subsistema mostrados na Figura 15, deixando assim o código mais organizado e de fácil compreensão.

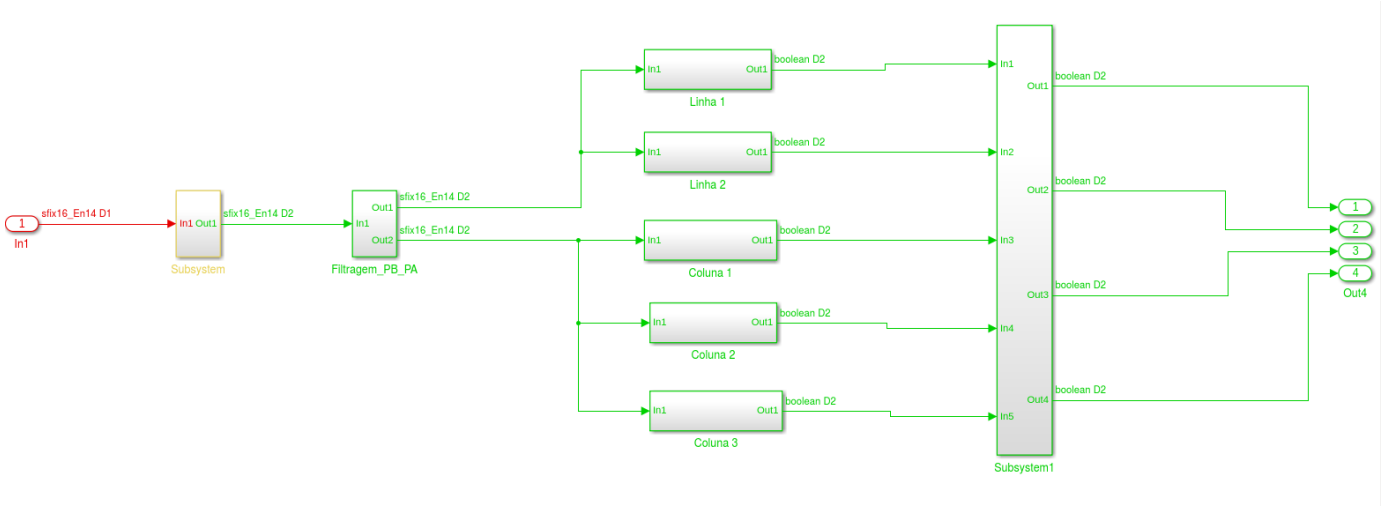

Figura 15 – Implementação do Subsistema 2

5 Referências Bibliográficas

[1] FERREIRA, João Roberto S. Gandara. Artigo Técnico: Como funcionam os tons DTMF. 2006. Disponível em: <http://www.cram.org.br/wordpress/?p=1325>. Acesso em: 16 dez. 2016.

[2] DES/ATA-005076-3: Specification of Dual Tone Multi-Frequency (DTMF) Transmitters and Receivers; Part 3: Receivers. European Telecommunications Standards Institute 2000, 2000.

[3] Math Works. HDL Coder. Disponível em: <https://www.mathworks.com/products/hdl-coder/features.htmlautomating-fpga-design>. Acesso em: 17 dez. 2016.