Bruno Marcos Espindola

Desenvolvimento e uso de módulos para processamento de sinais em FPGA

São José – SC

Agosto / 2011

Bruno Marcos Espindola

Desenvolvimento e uso de módulos para processamento de sinais em FPGA

Monografia apresentada à Coordenação do Curso Superior de Tecnologia em Sistemas de Telecomunicações do Instituto Federal de Santa Catarina para a obtenção do diploma de Tecnólogo em Sistemas de Telecomunicações.

Orientador:
Prof. Marcos Moecke, Dr.

CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS DE TELECOMUNICAÇÕES
INSTITUTO FEDERAL DE SANTA CATARINA

São José – SC

Agosto / 2011

Monografia sob o título *"Desenvolvimento e uso de módulos para processamento de sinais em FPGA"*, defendida por Bruno Marcos Espindola e aprovada em 12 de agosto de 2011, em São José, Santa Catarina, pela banca examinadora assim constituída:

Prof. Marcos Moecke, Dr. Eng.
Orientador

Prof. Mário de Noronha Neto, Dr. Eng.
IF-SC

Profa. Maria Cláudia de Almeida Castro, Ms. Sc.
IF-SC

"Inovação não é sinônimo de algo complicado. Ao contrário do que muitos pensam, as maiores inovações surgem de idéias simples que resolvem plenamente o problema observado. É em toda sua simplicidade que a roda, a maior invenção de todos os tempos, move o mundo moderno."

Robson Feitosa

Agradecimentos

Dedico o esforço na realização desse trabalho às pessoas que amo. Aos meus pais agradeço o incentivo aos estudos e ensinamentos durante toda minha vida. Aos amigos e namorada, agradeço todo o apoio e os inesquecíveis momentos de alegria que me proporcionaram e ainda irão proporcionar.

Agradeço ao professor Marcos Moecke pela orientação e auxílio na elaboração desse trabalho. Por final, agradeço os demais professores por todos os ensinamentos a mim passados no decorrer desses anos de curso.

Resumo

No mercado global atualmente existem diversas alternativas para o projeto de sistemas, dentre elas, os FPGAs que são dispositivos programáveis capazes de implementar circuitos digitais. Os FPGAs apresentam uma grande capacidade de processamento em função do paralelismo das operações. Este trabalho tem como resultado uma plataforma para processamento de sinais em FPGA (PS-FPGA) que possui um conjunto de módulos para: aquisição e reprodução de sinais; armazenamento e multiplicação de sinais; filtragem digital; gerador senoidal e módulos de controle. A plataforma PS-FPGA foi implementada usando dispositivos da ALTERA disponíveis no kit de desenvolvimento *DSP Development Kit - Stratix II Edition*. A PS-FPGA pode ser utilizada para o desenvolvimento de sistemas didáticos, que necessitam de um *hardware* com maior velocidade para a sua implementação devido ao intenso processamento de sinais em tempo real. Os módulos foram desenvolvidos em VHDL ou utilizando códigos disponibilizados por terceiros, denominados IP (*Intellectual Property*) tais como o NCO e o FIR Compiler. Os módulos desenvolvidos foram testados através de diferentes cenários de teste que implementam sistemas didáticos, com o objetivo de exemplificar, simplificar e incentivar a programação em dispositivos FPGA nos cursos da área de telecomunicações. Para o teste de alguns cenários foi necessário o uso de equipamentos externos como geradores de sinais arbitrários, osciloscópios e analisador de espectros. Também se utilizou o equipamento SignalTap, sintetizado no próprio FPGA, para a confirmação de sinais internos do FPGA. Os cenários implementados permitiram verificar o correto funcionamento dos módulos desenvolvidos. Dentre os módulos, o gerador de senoidal com frequência e ganho ajustável se destaca pela alta precisão no controle tanto do valor da frequência como na amplitude do sinal gerado. Em relação ao kit utilizado, percebeu-se que o transformador de RF existente junto aos dos conversores D/A e A/D impede o uso destes conversores para sinais de áudio, e sugere-se que em trabalhos futuros se utilize o CODEC de áudio disponível no kit para lidar com a faixa de áudio.

Palavras-chave: FPGA, Processamento de Sinais, VHDL, Gerador Senoidal, Filtros.

Abstract

In the global market there are actually several alternatives for the design of systems, including the FPGAs that are programmable devices capable of implementing digital circuits. FPGAs have a large processing capacity due to the parallelism of operations. This monograph has resulted in a signal processing platform using FPGA (PS-FPGA) which is a set of modules for functions such as: signals acquisition and reproduction, signals storage and multiplication, digital filtering, sinusoidal generator and control modules. The PS-FPGA platform is implemented using the Altera EP2S60 - DSP Development Kit - Stratix II Edition. This platform can be used to development educational systems which require a high-speed hardware implementation due to the intense signal processing in real time. The modules were developed in VHDL or using codes provided by third parties, called IP - Intellectual Property such as the NCO and the FIR Compiler. All the modules were tested using some test scenarios that implement educational systems in order to exemplify, simplify and encourage the use of FPGA device in telecommunications lessons. To evaluate some scenarios external equipment were required such as: arbitrary signs generators, oscilloscopes and spectrum analyzer. In some evaluations to measure signals in the FPGA device, we also have used the Signaltap equipment, which is an IP synthesized in the same device where the modules are implemented. The scenarios implemented allowed us to verify the correct operation of the developed modules. Among the modules, the sine generator with adjustable frequency and gain stands out for the high precision in controlling the value of the frequency and amplitude of the generated signal. In relation to hardware used, it was noted that the RF transformer connected to the A/D and D/A converters, prevents the use of these converters for audio signals, and suggests that in future work to use the audio codec available in the kit to deal with audio.

Keywords: FPGA, Signal Processing, VHDL, Sine Generator, Filters.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

Este trabalho tem como objetivo disponibilizar uma ferramenta didática que servirá como suporte a trabalhos futuros, possibilitando implementar em tempo real, sistemas desenvolvidos no ambiente de desenvolvimento Matlab/Simulink envolvendo o processamento de sinais. Neste capítulo serão apresentadas as principais motivações e contribuições deste trabalho. Apresenta ainda, na Seção 1.3, a organização do texto nos próximos capítulos.

1.1 Motivação e Justificativa

O projeto de equipamentos de telecomunicações nos últimos anos passou por muitas inovações, principalmente com a evolução dos circuitos de lógica programável. Dentre as possíveis soluções para projeto, a escolha da melhor alternativa é condicionada por fatores como escala de produção, tempo e custo de desenvolvimento, treinamento da equipe de desenvolvedores, e também a capacidade de processamento necessária. No mercado global atualmente existem diversas alternativas, como os DSPs (*Digital Signal Processor*), e os circuitos FPGAs (*Field Programmable Gate Arrays*), os quais apresentam uma grande capacidade de processamento, sendo que os FPGAs permitem uma maior velocidade de processamento em função do paralelismo das operações.

Os DSPs são microcontroladores especializados em processamento digital de sinal, sendo utilizados normalmente no tratamento de áudio e vídeo, assim como qualquer outra aplicação que requeira o processamento em tempo real, como automação e controle de dispositivos. Os DSPs são projetados levando-se em consideração que as operações mais frequentes no processamento digital são as de adição, multiplicação e consecutiva transferência de memória, porém o número de elementos executando essas operações é limitado. Por esse motivo, estes dispositivos precisam de muitos ciclos de relógio para calcular cada valor de saída. No caso de uma operação de filtragem, quanto maior a ordem do filtro, mais tempo levará para fazer o processamento.

Por outro lado, devido à grande flexibilidade de programação e campos de uso, os dispositivos FPGAs são uma boa escolha para a implementação de sistemas (PEDRONI, 2007). Neste

trabalho foi implementado um sistema para a aquisição, processamento e reprodução de sinais utilizando algumas ferramentas de desenvolvimento para FPGA. O resultado deste trabalho é uma plataforma para processamento de sinais em FPGA (PS-FPGA) que possui um conjunto de módulos para funções como: filtros digitais, circuito misturador, módulos para aquisição e reprodução de sinais, geradores de sinais, módulos de controle, os quais podem ser utilizados em projetos didáticos para a implementação de sistemas em tempo real. Estes podem ter sido inicialmente desenvolvidos em ambiente Matlab/Simulink e que devido ao intenso processamento de sinais necessitam de um *hardware* com maior velocidade para a sua implementação (ESPINDOLA B. M.; MOECKE, 2010).

1.2 Principais Contribuições do Trabalho

Neste trabalho, foram desenvolvidos módulos de *hardware* tipicamente utilizados em sistemas de telecomunicações que envolvem forte processamento de sinais. Os módulos desenvolvidos em VHDL (*Very High Speed Integrated Circuit Hardware Description Language*) foram utilizados para implementar diversos cenários de teste, nos quais os módulos foram conectados para implementar sistemas didáticos. Em alguns cenários utilizou-se códigos disponibilizados por terceiros, denominados IPs (*Intellectual Property*) tais como o NCO e FIR Compiler. A seguir são destacados os diferentes módulos desenvolvidos, e também os cenários de teste implementados.

- **Módulo de aquisição de sinais** - desenvolvido em VHDL que utiliza um conversor Analógico para Digital (A/D) externo ao FPGA e adapta o sinal digital para o processamento interno.
- **Módulo de reprodução de sinais** - desenvolvido em VHDL que adapta o sinal digital para a reprodução analógica através de um conversor Digital para Analógico (D/A) externo ao FPGA.
- **Módulo de armazenamento** - desenvolvido em VHDL que implementa uma memória RAM com portas de escrita e leitura de 12 bits separadas, para o armazenamento de sinais adquiridos pelo módulo de aquisição ou geradores internamente no sistema.
- **Módulo de gerador senoidal** - utiliza um IP do tipo NCO, permite implementar geradores de senoides e cossenoides com frequência ajustável através de um valor numérico.
- **Módulo de filtragem** - utiliza um IP de filtragem digital, podendo ser configurado para a resposta de frequência e também ordem do filtro de acordo com a aplicação desejada.

- **Módulo de multiplicação** - desenvolvido em VHDL e realiza a multiplicação entre dois sinais de 12 bits, pode ser usado em diversas aplicações de telecomunicações, entre elas o circuito misturador de frequências.

- **Módulo de anti-repique** - implementado com componentes discretos através do editor esquemático. É necessário para evitar o repique das chaves usadas no controle dos demais módulos.

- **Cenário de testes para aquisição e reprodução de sinais analógicos** - utilizado para realizar os testes de aquisição e reprodução de sinais analógicos.

- **Cenário de testes do gerador de sinais senoidal** - utiliza um circuito de controle para ajustar a amplitude e valor da frequência utilizada. Possibilita o ajuste da frequência por década com uma precisão de 1 Hz.

- **Cenário de testes da memória de armazenamento de sinais** - utiliza o módulo de armazenamento para realizar o armazenamento de sinais obtidos através do módulo de aquisição de sinais, ou sinais gerados internamente no FPGA.

- **Cenário de testes dos filtros** - neste cenário são utilizados sinais bitonais para o teste dos módulos de filtragem. O sinal resultante é disponibilizado externamente para analise em instrumentos de medição tais como osciloscópio e analisador de espectros.

- **Cenário de testes do misturador de frequências** - implementa um misturador de frequências utilizando um circuito multiplicador e um gerador senoidal. Neste cenário um sinal externo obtido pelo módulo de aquisição tem seu espectro de frequência deslocado para uma nova frequência central.

1.3 Organização do Texto

Esta monografia foi organizada da seguinte forma: no Capítulo 2 será apresentada a fundamentação teórica necessária para o desenvolvimento do trabalho. No Capítulo 3 será apresentada uma descrição dos módulos implementados destacando alguns aspectos teóricos dos processos envolvidos. No Capítulo 4 serão apresentados os cenários de teste utilizados para verificar o correto funcionamento dos módulos desenvolvidos, e são apresentados os resultados obtidos com a implementação. No Capítulo 5 são apresentados as principais conclusões e também são propostos possíveis melhorias ou acréscimos que podem ser incluídos em trabalhos futuros.

2 *Fundamentação Teórica*

Este capítulo apresenta a fundamentação teórica básica necessária para entender os módulos de processamento implementados. São abordados temas referentes aos circuitos FPGAs, linguagem de descrição de *hardware*, filtros digitais, técnica de modulação, o processo de aquisição e reprodução de sinais e as ferramentas e sistemas utilizados neste trabalho. Os módulos desenvolvidos tem o objetivo de facilitar futuros projetos na área de telecomunicações. Optou-se por desenvolver módulos que são mais frequentemente utilizados nesta área, entre eles: filtros digitais, gerador de senoides, circuito misturador, além de módulos para aquisição e reprodução de sinais e armazenamento.

2.1 Os Circuitos FPGAs

No mercado global atualmente existem diversas alternativas de dispositivos programáveis, dentre elas os circuitos FPGAs que possuem uma característica diferencial, a capacidade de executar processamento em paralelo. Este processamento inerentemente paralelo faz destes circuitos uma boa opção para implementação de sistemas difíceis de realizar em um processador comum por exigirem uma alta velocidade de processamento e/ou flexibilidade no número de bits.

Os FPGAs são circuitos integrados que possuem a capacidade de serem ajustados de forma a criar o *hardware* necessário para determinada aplicação. Estes dispositivos contém um grande número de elementos lógicos idênticos que implementam as funções lógicas e podem ser configurados individualmente e interconectados por uma matriz de trilhas condutoras e chaves programáveis. Cada fabricante e família de dispositivos pode ter uma arquitetura interna diferente, na qual os elementos lógicos podem conter vários recursos tais como: LUTs (*Look-Up Table*) - que são de forma geral uma SRAM, que armazena tabelas verdade para funções lógicas de n-entradas; Multiplexadores - que são dispositivos que possuem múltiplos fluxos de dados na entrada e somente um fluxo de dados na saída; Registradores - que são unidades de memória capazes de armazenar n-bits; Lógica de Carry - que são dispositivos utilizados para realizar operações de soma; e arranjos de portas lógicas. Além destes elementos lógicos básicos, os FPGAs podem possuir componentes mais sofisticados como, entre outros, multiplicadores de-

dicados, blocos de memória e PLLs (*Phase-locked loop*), de forma a conseguir otimizar o uso dos elementos lógicos e desenvolver projetos mais eficientes. No projeto desenvolvido utilizou-se um FPGA da família Stratix II (EP2S60F1020C4), o qual possui os seguintes blocos lógicos: 60440 Elementos Lógicos contendo 48352 Registradores lógicos dedicados e 48352 ALUTs (*Adaptative Look-Up Table*). O FPGA ainda possui 2.544.192 bits de memória RAM e 288 blocos de DSP, 12 circuitos PLLs e 1020 pinos externos sendo 712 para entradas/saídas do usuário (ALTERA, 2007).

A programação do FPGA é feita utilizando alguma linguagem HDL (*Hardware Description Language*). As linguagens HDLs são usadas para modelar através de um *software* a operação de um *hardware* programável. Existem linguagens HDLs tais como VHDL, Verilog, Handel-C, AHDL, SDL, ISP, ABEL. Entre estas, as mais utilizadas nas industrias, e disponibilizadas nos ambientes de desenvolvimento para plataformas FPGAs são a Verilog e VHDL. No projeto foi utilizada a linguagem VHDL por ser mais facilmente encontrada na literatura e também ser a linguagem utilizada no Curso Superior de Tecnologia em Sistemas de Telecomunicações. A VHDL é uma linguagem de alto nível que permite fazer um modelamento tanto estrutural como também o modelamento comportamental. O modelamento estrutural consiste em descrever o componente a partir dos seus componentes mais básicos e das interconexões dos mesmos. Já o modelamento comportamental permite descrever um componente a partir do que se quer como resultado, não importando-se qual *hardware* será implementado para que isso seja obtido.

A configuração do FPGA é feita através de um arquivo binário gerado pelo *software* do fabricante, a partir de um determinado projeto. Esse arquivo binário contém as informações necessárias para especificar as funções de cada bloco lógico e realizar as interconexões entre eles.

2.2 Ferramentas e *Hardware* Utilizado

Para o desenvolvimento dos módulos foram utilizadas diversas ferramentas da ALTERA, e uma plataforma de *hardware* do mesmo fabricante de dispositivos lógico programáveis. O *hardware* utilizado foi o kit de desenvolvimento DSP EP2S60 (Ver foto no Anexo A), que é voltado ao desenvolvimento de diversos projetos e sistemas com processamento de sinais digitais. Ele é baseado em um FPGA EP2S60F1020C3 da família Stratix II ligado a dispositivos com: conversores A/D de 12 bits que suportam uma taxa de até 125 MHz; conversores D/A de 14 bits que suportam uma taxa de até 165 MHz; oscilador de 100 MHz; mostradores de sete-segmentos; botões do tipo contato-momentâneo (*push-button*); e Diodos Emissores de Luz (LEDs). Tanto o FPGA como os dispositivos periféricos foram utilizados para a implementação dos cenários. Além destes componentes, o kit possui conexões com saída VGA, RS-232, Compact Flash, e um CODEC 96-kHz.

Para a programação do *hardware* do FPGA foi utilizado principalmente o *software* Quartus II, que é um ambiente de desenvolvimento que permite realizar a programação do FPGA utilizando uma linguagem de descrição de *hardware*. O ambiente também permite compilar o programa, definir a conexão dos pinos do FPGA, controlar o uso dos recursos disponíveis na placa, realizar a implementação do circuito no FPGA, e carregar o arquivo de configuração obtido para o FGPA via interface JTAG[1].

Antes do teste no *hardware* optou-se por realizar a simulação com o Modelsim-Altera, *software* o qual permite a criação de um banco de testes para detectar erros de implementação dos módulos e cenários sem a necessidade de realizar síntese no FPGA, tornando mais ágil e rápido o desenvolvimento.

Alguns módulos implementados neste projeto foram desenvolvidos com IPs, que são blocos de códigos HDL com interfaces de configuração predefinidas pelo desenvolvedor. Alguns IPs são liberados para uso acadêmico, e por isso simplificam a implementação de projetos, reduzindo o tempo de desenvolvimento e testes. Outros desenvolvedores disponibilizam em parceria com os fabricantes de FPGAs estes módulos IPs vendendo as licenças de uso por implementação.

2.3 Filtros Digitais

Os sistemas práticos estão sujeitos a ruídos aleatórios e perturbações que podem dificultar a análise de um sinal ou outro procedimento. No processamento de sinal, a função de um filtro é eliminar partes indesejadas, tais como ruídos, e principalmente, extrair partes úteis fazendo o tratamento das informações no domínio da frequência, separando, classificando e medindo sinais.

Os filtros digitais são implementados utilizando elementos básicos como multiplicadores, somadores e elementos de atraso, de acordo com o algoritmo determinado pela função de transferência dos filtros e suas formas de realização, que é como a função transferência é implementada em *hardware*. Eles oferecem uma maior flexibilidade e versatilidade em relação aos filtros analógicos já que vem a ser uma tarefa fácil alterar o valor dos coeficientes de um filtro digital, mudando assim a sua função de transferência. Além disso, os filtros digitais são mais estáveis, visto que os filtros analógicos são sujeitos a variações de temperatura, variações de valores devido aos componentes usados no circuito, e outros parâmetros que dependem da aplicação e do projeto (SHENOI, 2006).

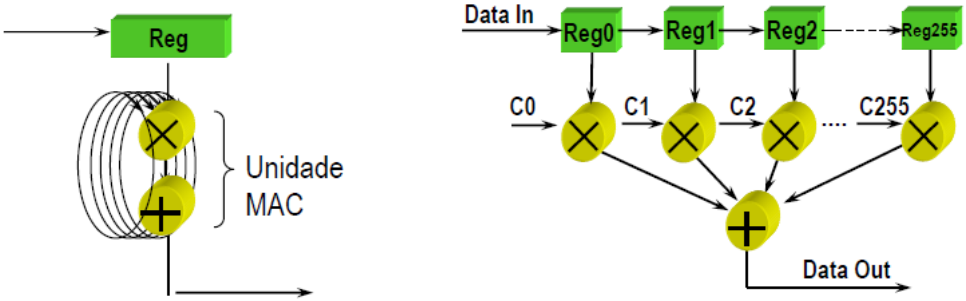
Os filtros digitais podem ser classificados quanto a duração da resposta ao impulso em: filtros de resposta ao impulso infinita (IIR) e filtros de resposta ao impulso finita (FIR). Os

[1] `http://focus.ti.com/lit/an/ssya002d/ssya002d.pdf`

filtros FIRs diferenciam-se dos IIRs por serem não recursivos, portanto a resposta de um filtro FIR depende exclusivamente do sinal de entrada atual e anteriores, sem depender dos valores de saída anteriores.

Os filtros FIRs possuem algumas vantagens em relação aos IIRs, tais como: podem ser projetados de forma a ter fase sempre linear; são sempre estáveis; as amostras da resposta ao impulso são os mesmos valores dos coeficientes da função de transferência. Já os filtros IIRs possuem a vantagem de exigirem uma menor ordem para realizarem uma determinada resposta em frequência. Devido a estas características de estabilidade e principalmente pela disponibilidade de uso de um IP, neste trabalhou optou-se pelo uso dos filtros FIR.

Em relação a implementação de um filtro digital utilizando FPGAs, pode-se utilizar uma solução usando *hardware* paralelo, no qual a cada ciclo de relógio consegue-se determinar um valor de saída sendo ideal para operação em altas taxas de amostragem e aplicações que necessitam de atraso mínimo. A Figura 2.1 apresenta a diferença de implementação de um filtro na forma paralela e serial. Note que o uso de DSPs implicaria necessariamente na implementação serial, enquanto que com FPGA pode-se selecionar o nível de paralelismo de acordo com os recursos de *hardware* disponíveis e velocidade de processamento necessária.



(a) (b)

Figura 2.1: Implementação de um filtro FIR com 256 coeficientes usando arquitetura serial (a) e paralela (b) (RAMIREZ; MCCLOUD, 2007).

2.4 Modulação e Demodulação

A difusão de rádio é feita através de ondas eletromagnéticas transmitidas pelo ar. O tamanho da antena que recebe este sinal é diretamente proporcional ao seu comprimento de onda. Portanto, para se transmitir sinais na faixa de frequência de voz seria necessário antenas de grandes proporções, na faixa de quilômetros. Com a modulação ocorre um deslocamento das frequências do sinal permitindo o uso de antenas muito menores, podendo chegar a ordem de milímetros. A

modulação também atende à necessidade de compartilhar o meio de transmissão. Isso pode ser feito modulando as mensagens em diferentes frequências, sendo possível ao receptor, selecionar qual frequência será demodulada. Com a modulação é possível que transmissores transmitam diferentes sinais em um mesmo canal sem que ocorra sobreposição e perda de informação.

Na modulação o sinal em banda base $m(t)$ é usado para alterar algum parâmetro de uma portadora de alta frequência. Normalmente ou a frequência (modulação FM), ou fase (modulação PM) ou amplitude (modulação AM) são modificados proporcionalmente ao sinal em banda base. No receptor, o sinal modulado deve passar por um processo reverso chamado demodulação, no qual a informação do sinal original é recuperada (LATHI, 1998).

2.4.1 Modulação AM

Modulação em amplitude é caracterizado pelo fato de que a amplitude A de uma portadora $A\cos(\omega_c t + \theta_c)$ varia em proporção ao sinal em banda base $m(t)$, o sinal modulante. A frequência ω_c e a fase θ_c são constantes. Se a amplitude da portadora A é diretamente proporcional ao sinal modulante $m(t)$, o sinal modulado é $m(t)\cos\omega_c t$. Este tipo de modulação simplesmente desloca o espectro de $m(t)$ para a frequência da portadora. Se $M(\omega)$ corresponde ao $m(t)$ no domínio da frequência:

$$m(t)\cos\omega_c t \Leftrightarrow \frac{1}{2}[M(\omega+\omega_c)+M(\omega-\omega_c)] \tag{2.1}$$

Note que o sinal $M(\omega-\omega_c)$ é o $M(\omega)$ deslocado de ω_c para a direita, e o sinal $M(\omega+\omega_c)$ é $M(\omega)$ deslocado de ω_c para a esquerda. Sendo assim, este processo de modulação desloca o espectro do sinal modulante de ω_c pra direita e para esquerda.

2.4.2 Misturador de Frequência

O misturador de frequência mostrado na Figura 2.2 é um circuito muito usado para deslocar uma frequência alta Fr tanto para frequências maiores que a original (*upconverter*) como para frequências menores (*downconverter*). Este circuito efetua a multiplicação de um sinal com frequência central Fr com uma e uma senoide de frequência Fo, resultando no deslocamento do espectro de frequência do sinal centrado em Fr para duas novas posições. Esse produto resulta em dois novos sinais com frequência soma: $Fr+Fo$ e frequência diferença: $Fr-Fo$. Um desses dois sinais é escolhido como sinal de frequência intermediária (FI) por meio de um filtro adequado.

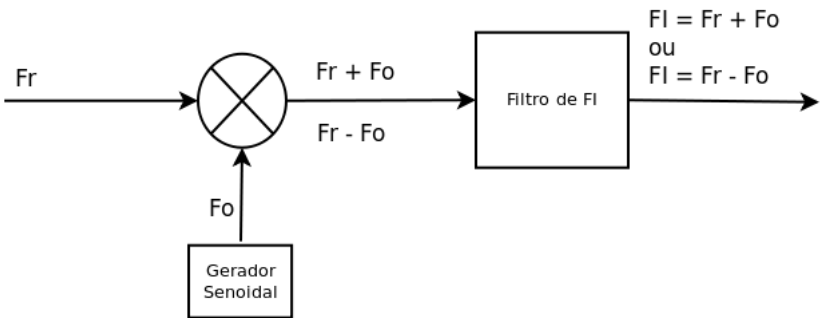


Figura 2.2: Diagrama de um misturador.

2.5 Aquisição e Reprodução de Sinais

Para o processamento digital de um sinal analógico é necessário que uma conversão de analógico para digital e vice-versa seja feita. Os dispositivos utilizados para fazer essa conversão são: o conversor A/D e o conversor D/A. Os conversores A/D são dispositivos eletrônicos que discretizam o sinal analógico, tanto na dimensão temporal (amostragem em intervalos uniformes) como na dimensão amplitude (quantização). O valor das amostras resultantes são representados através de bits. Quanto maior a quantidade de bits, menor será o erro de quantização, pois é possível representar um maior número de níveis de referência. O sinal discreto obtido deve manter fielmente a informação original. Para que isso ocorra é necessário que a taxa de amostragem utilizada seja pelo menos duas vezes maior do que a frequência máxima (f_{max}) do sinal a ser amostrado, conforme o teorema de Nyquist.

$$f_s \geq 2 \times f_{max} \tag{2.2}$$

No processo de conversão D/A cada amostra digital é transformada em um valor correspondente de tensão ou corrente. No intervalo entre as amostras, o valor anterior é normalmente mantido, resultando em uma significativa distorção de alta frequência. Através de um filtro passa baixa analógico é possível remover essas frequências e recuperar o sinal.

3 *Desenvolvimento*

Neste capítulo será apresentado como foi realizada a aquisição e a reprodução de sinais no kit, além de explicar o funcionamento dos blocos reaproveitados. Encontra-se também, a descrição dos principais blocos desenvolvidos para realizar os cenários.

3.1 Aquisição e Reprodução de Sinais

A plataforma de *hardware* utilizada no projeto é voltada ao processamento de sinais. Por este motivo, este kit já possui conversores A/D e D/A integrados na placa. Neste projeto, optou-se pelo uso desses conversores por já estarem disponíveis e por não apresentarem muita complexidade no seu uso.

3.1.1 Aquisição

No processamento de sinais analógicos através de sistemas implementados em FPGA, é necessário converter o sinal analógico de entrada para um sinal digital e, na saída do sistema, converter o sinal digital novamente para analógico. Para a conversão A/D o kit utilizado possui dois conversores (AD9433) de 12 bits. Nestes conversores, a polaridade do sinal de saída é representado no formato complemento de dois, podendo portanto variar na faixa de -2048 a 2047. O Apêndice C.1 apresenta como é feita a configuração da taxa de amostragem a ser usada pelos conversores. Em todos os cenários realizados, a frequência de amostragem utilizada nos conversores foi de 100 MHz.

3.1.2 Reprodução

Para a conversão D/A o kit utilizado possui dois conversores (DAC904) de 14 bits. Nestes conversores a polaridade do sinal de entrada deve ser representada no formato sem sinal (*unsigned*), podendo portanto variar na faixa de 0 a 16383. O Apêndice C.2 apresenta como é feita a configuração da taxa de reprodução a ser usada pelos conversores. Em todos os cenários realizados, a frequência de reprodução utilizada nos conversores foi de 100 MHz.

3.2 Armazenamento

Em muitas aplicações de processamento de sinais é necessário armazenar informações para o uso posterior, como por exemplo armazenar amostras de sinais, coeficientes de filtros, dados processados e períodos de uma senoide. Para sintetizar essas memórias em FPGA existem diferentes possibilidades tais como: memórias FIFO (*First In, First Out*), ROM (*Read Only Memory*) ou RAM (*Random Access Memory*). As memórias possuem variações na forma como podem ser acessadas, tendo portas únicas ou múltiplas, podendo ou não ser lidas e escritas ao mesmo tempo. A definição do tipo de memória a ser usada em cada sistema (módulo) depende de suas necessidades. Muitos circuitos de FPGA possuem blocos de memória prontos, que podem ser utilizados na síntese. Quando os blocos de memória disponíveis não são suficientes, a memória é implementada utilizando os elementos lógicos disponíveis no *chip*.

Para fazer o armazenamento de dados na saída do conversor A/D de 12 bits, foi criado um bloco de memória RAM mostrado na Figura 3.1, com as seguintes características: endereçamento de 17 bits (*address*); porta de 12 bits para escrita de dados (*data_in*) e leitura de dados (*data_out*) separadas; pino de seleção de leitura ou escrita (WR), com ativo alto para escrita. Esta memória possui 2^{17} endereços sendo suficiente para armazenar até 2,97 segundos de sinal amostrado a 44100 Hz.

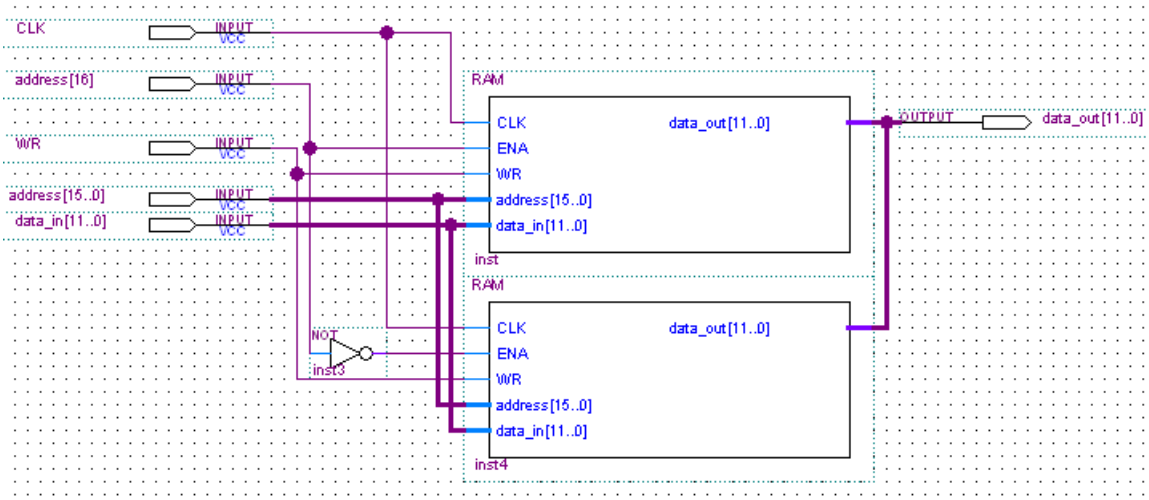


Figura 3.1: Diagrama esquemático da memória RAM.

No ambiente de trabalho da Altera, o limite máximo de implementação de uma memória RAM é de 16 bits de endereço. Por isto, para criar um bloco com 17 bits fez-se o uso de dois blocos de memória com 16 bits. Isto foi possível apenas colocando os dois blocos de RAM em paralelo e utilizando o bit de endereço mais significativo para escolher qual das memórias estaria ativa para determinado endereço. Para que as saídas possam ser ligadas em paralelo é necessário que elas sejam configuradas para alta-impedância (*tri-state*), de modo que apenas o bloco selecionado pelo endereço address[16] envie o conteúdo da memória para o barramento

data_out durante o ciclo de leitura (WR=0). A Figura 3.1 mostra a conexões entre as RAMs.

3.3 Gerador Senoidal

Para implementar um gerador senoidal em um sistema utilizando FPGA existem diversas opções. A escolha da melhor alternativa depende dos recursos disponíveis no *chip* e qualidade da onda que se busca gerar. A seguir serão detalhados os estudos realizados para a geração de senoides utilizando a ROM para armazenar um período e depois utilizando a ferramenta NCO. O primeiro método apresenta algumas limitações principalmente na definição do valor da frequência com precisão. Por outro lado o segundo método que emprega um IP, é simples de ser configurado e apresenta uma excelente resolução de frequência, utilizando no entanto uma quantidade maior de recursos de hardware do FPGA.

3.3.1 Período do Sinal Armazenado em ROM

Inicialmente foi implementado um gerador simples, no qual uma memória ROM armazena 50 amostras de um período de um sinal. Pela leitura periódica dessas amostras é gerado o sinal senoidal, cuja frequência é controlada a partir da frequência de leitura das amostras da memória. Nesta situação o sinal senoidal de 2 MHz foi o que apresentou a melhor qualidade quando se usou um relógio de 100 MHz para a leitura da memória. Para alterar a frequência do sinal senoidal utilizou-se uma divisão do relógio de 100 MHz, disponível na entrada do FPGA, através de um circuito divisor de relógio.

3.3.2 NCO Altera

Uma outra opção de gerador de *hardware* mais complexo foi implementada usando um IP denominado NCO MegaCore (ALTERA, 2011b). Este IP é um oscilador controlado numericamente (*Numerically-controlled oscillator* - NCO) que possibilita a seleção através de uma entrada numérica de uma grande faixa de valores de frequência de saída. A implementação usada no NCO pode ser com o algoritmo CORDIC (*COordinate Rotation DIgital Computer*), memória ROM grande, memória ROM pequena e baseada em multiplicadores. O Apêndice A apresenta como pode ser feito o uso deste IP utilizando o Quartus II. Nas subseções que seguem, é descrito o funcionamento do NCO, e descritas resumidamente os tipos de arquitetura e a frequência máxima que pode ser gerada pelo NCO.

Funcionamento do NCO

O funcionamento do NCO pode ser descrito através do diagrama de blocos mostrado na Figura 3.2. A forma de onda gerada pelo NCO é definida pela equação:

$$s(nT) = Asen(2\pi f_o nT) \tag{3.1}$$

onde T é o período de funcionamento do relógio; f_o é a frequência de saída não modulada com base no valor de entrada ϕ_{inc}; A é 2^{N-1}; e N é um inteiro na faixa de 10 a 32 representado a precisão de magnitude. Para determinar a frequência *fo* do sinal de saída do NCO utiliza-se o parâmetro incremento de fase (ϕ_{inc}) conforme:

$$fo = \frac{\phi inc \times f_{clk}}{2^M} \tag{3.2}$$

onde M é o número de *bits* usados para definir a precisão de fase do acumulador (parâmetro: *phase accumulator precision*) e f_{clk} a frequência do relógio de entrada no NCO.

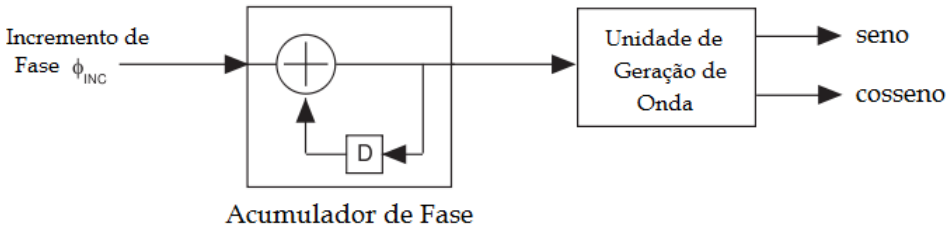


Figura 3.2: Diagrama genérico do NCO(ALTERA, 2011b).

Além desses parâmetros, outros que são importantes na configuração do NCO são: A precisão angular (parâmetro: *angular precision*) que ajusta a precisão do ângulo de fase antes da conversão de polar para cartesiano. A precisão de magnitude (parâmetro: *magnitude precision*) define o número de bits de saída, responsável por mudar a precisão que o seno/cosseno pode ser representado (ALTERA, 2011b).

O IP de NCO utilizado possui uma ferramenta de parametrização, sendo assim, fazer a mudança entre as arquiteturas é uma tarefa relativamente simples. Uma vez gerado o bloco de código do NCO, este pode ser usado como um componente descrito em VHDL, ou bloco esquemático.

Para que o NCO funcione, é necessário gerar sinais para as entradas *phi_inc*, *reset_n*, *clken* e *clk*. O *phi_inc* deve receber o valor calculado usando a Equação 3.2. As entradas *reset_n* e *clken* devem receber o valor '1' enquanto entrada *clk* recebe o sinal de relógio.

Implementação com memória ROM grande

A implementação com memória ROM grande é usada em projetos que necessitam de senoidais de alta frequência e quando pode-se utilizar grande quantidade de memória interna. Nessa arquitetura a ROM armazena os valores de amostras correspondentes aos 360º de um período do seno e cosseno sendo endereçada pela saída do acumulador de fase. Por possuir na memória interna todos os valores possíveis de saída para um determinado valor incremento, a forma de onda gerada apresenta uma grande pureza espectral. Comparado com as outras arquiteturas, a ROM grande utiliza o menor número de elementos lógicos (*Logic Elements* - LEs) quando o FPGA usado possui um grande número de blocos de memória disponíveis. Porém, quando o *chip* usado não possui estes blocos prontos, uma alta quantidade de LEs é necessária para implementar as memórias.

Implementação com memória ROM pequena

A implementação com memória ROM pequena é usada em projetos que necessitam alta frequência de sinal gerado, pouco uso de LEs (quando há blocos de memória já disponíveis no FPGA) e um tamanho de memória menor que a arquitetura com memória ROM grande. Em uma arquitetura com memória ROM pequena, o dispositivo de memória armazena apenas

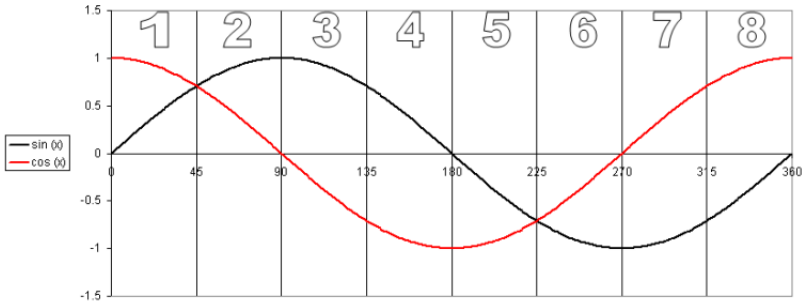


Figura 3.3: Derivação dos valores de saída da arquitetura ROM pequena(ALTERA, 2011b).

Tabela 3.1: Derivação dos valores de saída da arquitetura ROM pequena.

Posição no Círculo Unitário	Faixa para Fase x	sen(x)	cos(x)
1	$0 \leq x < \pi/4$	sen(x)	cos(x)
2	$\pi/4 \leq x < \pi/2$	cos(π/4x)	sen(π/2-x)
3	$\pi/2 \leq x < 3\pi/4$	cos(x-π/2)	-sen(x-π/2)
4	$3\pi/4 \leq x < \pi$	sen(π-x)	-cos(π-x)
5	$\pi \leq x < 5\pi/4$	-sen(x-π)	-cos(x-π)
6	$5\pi/4 \leq x < 3\pi/2$	-cos(3π/2-x)	-sen(3π/2-x)
7	$3\pi/2 \leq x < 7\pi/4$	-cos(x-3π/2)	-sen(x-3π/2)
8	$7\pi/4 \leq x < 2\pi$	-sen(2π-x)	cos(2π-x)

os valores correspondentes a 1/8 do período do sinal (45^o). Todos os outros valores de saída são derivados desses valores baseado na posição de rotação do fasor no círculo unitário como mostrado na Tabela 3.1 e Figura 3.3.

Implementação com o algoritmo CORDIC

No caso do algoritmo CORDIC, ao invés de armazenar os valores das amostras, estas são calculadas. O CORDIC além de possibilitar o cálculo de funções trigonométricas como seno e cosseno, também pode ser usado para a conversão de sinais, de polar para cartesiano e cartesiano para polar, e também implementações de Transformada Discreta de Fourier (*Discrete Fourier Transform* - DFT) e Transformada Discreta de Cosseno (*Discrete Cosine Transform* - DCT). Esse algoritmo fornece uma solução de alta performance para osciladores com precisão muito alta em sistemas com pouco uso memória interna. O algoritmo é baseado no conceito de rotação do fasor complexo por multiplicação de ângulo de fase por constantes sucessivamente menores. Em *hardware* digital, a multiplicação é feita apenas por expoente dois, por esse motivo pode ser implementado apenas com uma série de deslocamento, soma/subtração de números binários.

O algoritmo CORDIC (ANDRAKA, 1998) calcula o seno e o cosseno de um valor de fase de entrada de forma interativa deslocando o ângulo de fase para aproximar os valores da coordenada cartesiana com o ângulo de entrada. No final, a coordenada X e Y dada por um determinado ângulo representa o cosseno e seno desse ângulo, respectivamente, conforme pode ser visto na Figura 3.4.

No NCO Megacore é possível escolher entre uma arquitetura CORDIC paralela ou serial:

- A arquitetura em paralelo cria um oscilador de alta performance e precisão implementado apenas com elementos lógicos. Com essa arquitetura há um valor de saída em cada ciclo de relógio.
- A arquitetura de CORDIC em serial usa menos recursos que a arquitetura em paralelo, porém seu rendimento é reduzido por um fator igual ao valor de precisão de magnitude. Por exemplo, se o valor selecionado na precisão de magnitude for N, a taxa de saída e de Nyquist é reduzida por um valor de N. Essa arquitetura é implementada apenas com elementos lógicos e é útil se o projeto necessita de baixa frequência e formas de onda com alta precisão. Nessa arquitetura o resultado dos somadores é armazenado internamente e um novo valor é enviado para a saída somente após N ciclos de relógio.

Implementação Baseada em Multiplicadores

A arquitetura baseada em multiplicadores utiliza multiplicadores para reduzir o uso de memória. Pode ser implementado utilizando elementos lógicos ou multiplicadores dedicados.

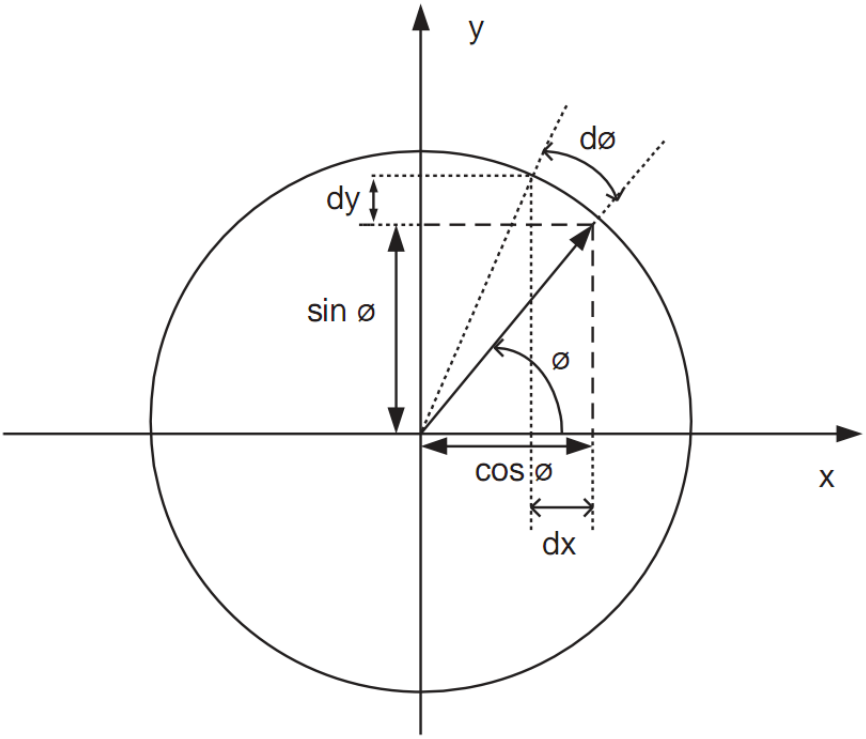


Figura 3.4: Rotação CORDIC para o calculo de seno e cosseno(ALTERA, 2011b).

Quando especificado saída dupla no NCO, é possível obter duas saídas defasadas 90º entre si, o seno e o cosseno. Porém, nesta arquitetura a saída dupla reduz o padrão de saída por fator de dois, produzindo um valor somente a cada dois ciclos de relógio.

Frequência Máxima do NCO

A frequência senoidal máxima que o NCO pode gerar é delimitada pelo critério de Nyquist, sendo então a metade da frequência do relógio de entrada, podendo ainda ser limitada a valores menores em algumas arquiteturas. Se um novo valor na saída do NCO é produzido a cada ciclo de relógio, como por exemplo a CORDIC em paralelo, a taxa máxima é a de Nyquist. Se no entanto a arquitetura escolhida necessita de ciclos adicionais de relógio para calcular os valores, como a CORDIC em serial, a frequência máxima de Nyquist deve ser dividida pelo número de ciclos necessários para gerar um novo valor de saída.

Nos módulos testados através dos cenários 4.2 e 4.4 do Capítulo 4, foram utilizados o NCO com ROM pequena pois buscava-se fazer o uso de um gerador com alta frequência de sinal gerado e um menor gasto de elementos lógicos, aproveitando os blocos de memória disponíveis no kit.

3.4 Filtros

Em muitos sistemas de telecomunicações é frequente o uso de utilizam filtros para eliminar frequências indesejadas, para fazer análise de sinais e para extrair partes úteis. Os filtros IIR possuem um custo computacional de implementação menor pois necessitam menos coeficientes para realizar uma determinada resposta em frequência e consequentemente menos multiplicadores e memórias. Por outro lado, os filtros FIR podem possuir fase linear e são estáveis tornando-os atrativos para serem usados em um grande número de sistemas. Por esse motivo, optou-se por filtros FIR neste trabalho, utilizando o FIR Compiler que será apresentado a seguir.

3.4.1 FIR Compiler Altera

O FIR Compiler Altera (ALTERA, 2011a) é uma ferramenta que agiliza e facilita a implementação de filtros FIR em FPGA. Utilizando esta ferramenta é possível projetar e implementar filtros FIR com algumas técnicas de janelamento. Também é possível importar os coeficientes, proporcionando que o projeto seja feito em outra ferramenta, como o Fdatool do Matlab, utilizando apenas os coeficientes resultantes para criar o filtro no FIR Compiler. O filtro gerado pode ser usado como um componente descrito em VHDL ou, no caso do ambiente Quartus II, o bloco esquemático. O Apêndice B apresenta o procedimento usado para configurar e utilizar este IP.

Devido a facilidade oferecida pelo ambiente gráfico do FIR Compiler, foi adotado este IP para o projeto de filtros neste trabalho. Para a configuração e uso correto desse IP na síntese de filtros, alguns parâmetros são essenciais: os coeficientes desejados para o filtro; a quantidade de bits usada na representação dos coeficientes e sinais; a velocidade que o filtro fará o processamento e a arquitetura utilizada na implementação.

Os coeficientes da função de transferência são os números que definem o comportamento de um filtro e a sua resposta em frequência. Para o cálculo dos coeficientes foi utilizado o IP FIR Compiler, no qual especificando o tipo de resposta em frequência, ordem do filtro e frequências de corte, os coeficientes são obtidos diretamente (ver Apêndice B). Definidos os valores dos coeficiente, é necessário definir a quantidade de bits usada para representar cada coeficiente. O filtro ficará mais perto do que foi projetado quanto maior for a quantidade de bits utilizada na representação. Porém, a quantidade de bits está diretamente relacionada com o custo de *hardware* do filtro, e por isso deve-se buscar a melhor relação entre a quantidade de bits e o custo.

A escolha da arquitetura a ser usada é fundamental para limitar a complexidade (uso de recursos) e determinar a velocidade do processamento (frequência máxima) do filtro. A arquitetura de processamento em paralelo é a mais rápida pois calcula um valor de saída a cada

ciclo de relógio, utilizando uma maior quantidade de recursos lógicos. Por outro lado, as arquiteturas em série utilizam menos recursos, porém o sinal de saída tem um maior atraso no processamento, que é tipicamente medido em ciclos de relógio, conforme pode ser observado na Tabela 3.2. A escolha da estrutura de implementação deve ser feita baseando-se na quantidade de recursos disponíveis e necessidade específica do filtro.

Tabela 3.2: Relação entre o custo de *hardware* e velocidade para filtros implementados com arquitetura paralela e serial para um FPGA do tipo Stratix II utilizando o IP FIR Compiler.

Filtro de ordem	Paralelo		Serial	
	Células Lógicas	Tempo de Atraso	Células Lógicas	Tempo de Atraso
20	1385	1 ciclo de relógio	694 (50%)	15 ciclos de relógio
50	3133	1 ciclo de relógio	1236 (39%)	16 ciclos de relógio
100	5843	1 ciclo de relógio	2096 (36%)	16 ciclos de relógio
200	11608	1 ciclo de relógio	3869 (33%)	16 ciclos de relógio

3.5 Misturador de Frequência

O misturador foi implementado em VHDL através de um circuito multiplicador (Figura 3.5). Este bloco multiplicador possui duas entradas, uma de 12 bits e outra de 14 bits. Na entrada de 12 bits é ligado o sinal modulado vindo do conversor A/D. Na entrada de 14 bits é ligado um NCO que está fazendo a geração da frequência pela qual o sinal modulado será multiplicado. Ao fazer uma multiplicação de 12 bits com 14 bits, tem-se como resultado um sinal com 26 bits. Como o conversor D/A suporta apenas 14 bits de entrada, fez-se necessário eliminar os bits menos significativos do sinal produto. Desta forma, a saída do bloco multiplicador corresponde aos 14 bits mais significativos do produto dos dois sinais ligados na entrada do bloco.

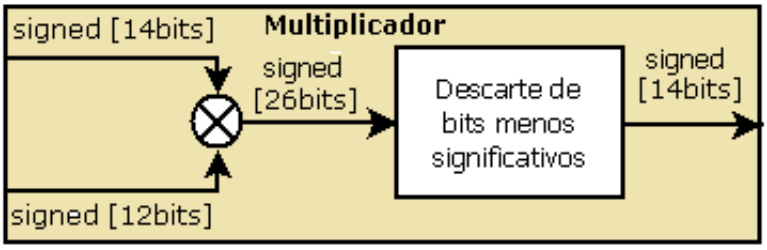


Figura 3.5: Diagrama do multiplicador implementado.

3.6 Circuito de Anti-Repique

As chaves mecânicas de contado momentâneo(*push-botton*) utilizadas ao serem pressionados produzem repiques que resultam em várias transições de sinais se não forem tratadas. Para

tal foi utilizado um circuito anti-repique (*debouncing*), implementado conforme mostrado na Figura 3.6. Utiliza-se um bloco contador para gera um sinal de relógio de 10ms, o qual é usado como relógio dos flip-flops D que estão conectados as chaves. O tempo de 10ms foi determinado de modo a eliminar o repique mecânico.

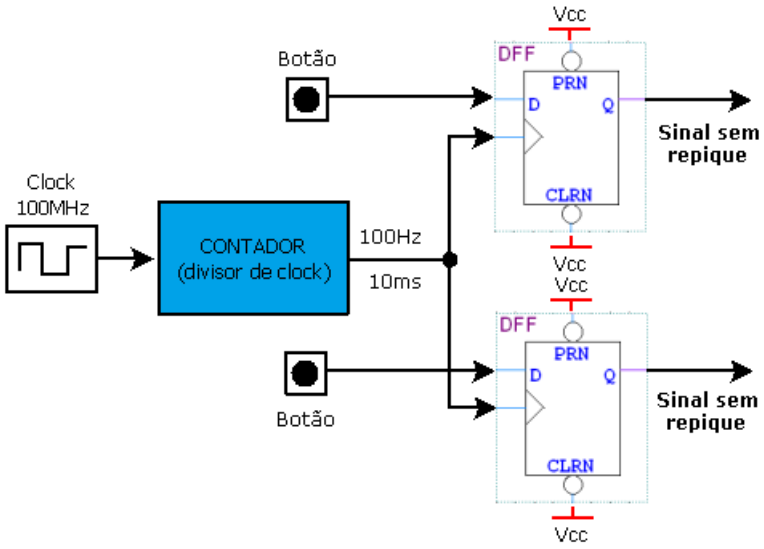


Figura 3.6: Diagrama do circuito anti-repique.

4 *Cenários de Teste*

Para teste dos módulos desenvolvidos neste trabalho, foram utilizados diversos cenários que visam "simular" circuitos utilizados em aplicações reais na área de telecomunicações. Os cenários foram primeiro simulados usando a ferramenta Modelsim e uma vez comprovado o seu funcionamento, os circuitos foram sintetizados no FPGA. Para efetuar as medições foram utilizados geradores externos de sinais, osciloscópios e analisadores de espectro, conforme pode ser visto na Figura 4.1. Em alguns cenários, optou-se pela sintetização em FPGA de um equipamento de medição denominado SignalTap, que é um analisador lógico integrado ao Quartus II que utiliza o próprio FPGA e a interface JTAG para capturar e apresentar em tempo real qualquer sinal interno no *chip*.

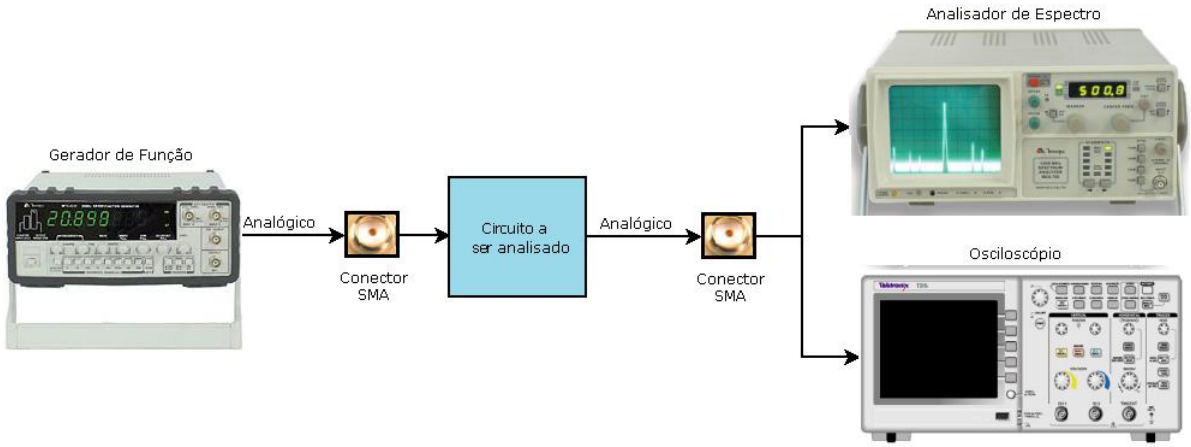


Figura 4.1: Bancada de testes usada para análise dos cenários sob teste.

4.1 Aquisição e Reprodução de Sinais Analógicos

Para testar a aquisição e reprodução de sinais no kit, foi utilizado um cenário básico mostrado na Figura A.1. Neste caso, um gerador senoidal gera o sinal a ser enviado para o conversor D/A. O sinal analógico na saída do conversor D/A é conectado através de um cabo coaxial a entrada do conversor A/D. O sinal digital obtido na saída do conversor A/D que é do tipo *signed* de 12 bits é convertido para *unsigned* de 14 bits pelo bloco conversor "*converter unsigned* " e

em seguida convertido para analógico pelo outro conversor D/A do kit, como pode ser visto na Figura A.1.

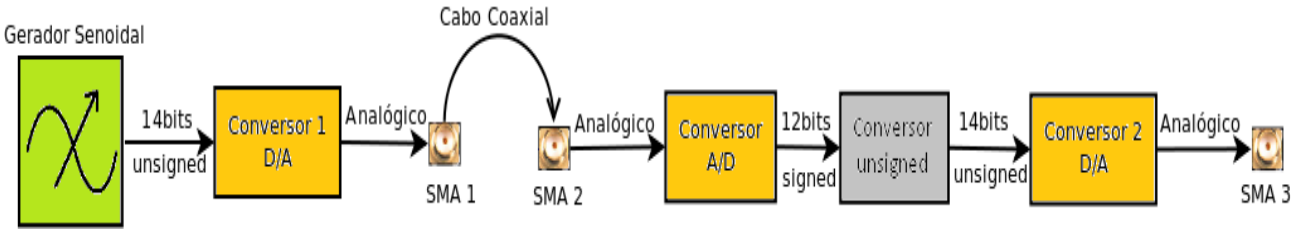
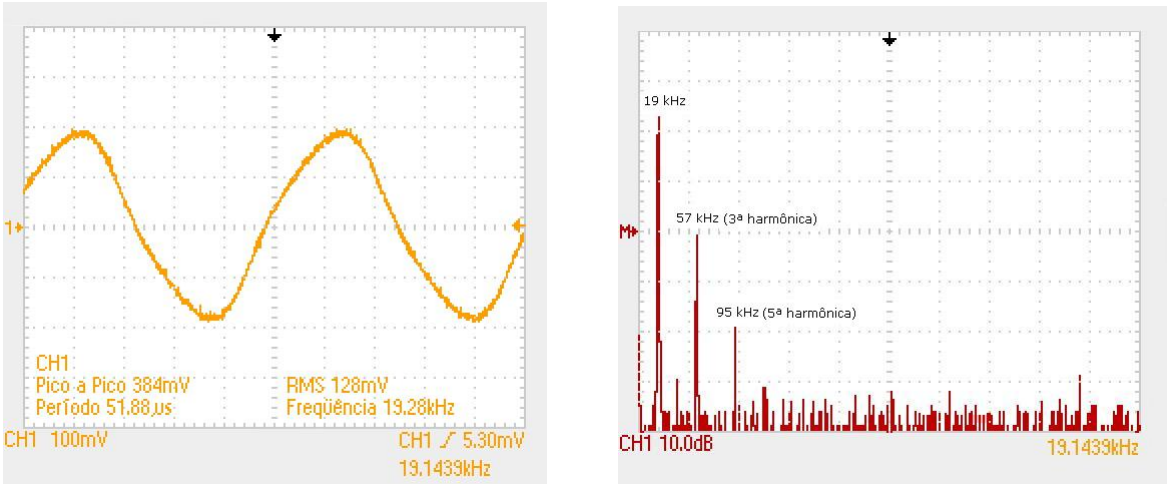


Figura 4.2: Cenário de testes de aquisição e reprodução de sinais.

Optou-se pela geração interna do sinal, para primeiro testar a reprodução do sinal através do conversor D/A e depois testar a aquisição do sinal via conversor A/D. A nova conversão D/A tem por objetivo medir externamente o sinal fazendo o tratamento correto do tipo de representação numérica das amostras. A Tabela 4.1 apresenta o gasto computacional do circuito implementado.

Tabela 4.1: *Hardware* utilizado no cenário de aquisição e reprodução.

Componente	Disponíveis	Utilizados	Percentagem de uso
ALUTs	48352	827	2%
Registradores Lógicos Dedicados	48352	1031	2%
Bits de Memória RAM	2544192	335872	13%
Blocos DSP	288	2	1%



(a) Sinal no domínio do tempo

(b) FFT do sinal

Figura 4.3: Captura de tela do osciloscópio de um sinal de 19 kHz na saída do conversor D/A.

Nos testes realizados, percebeu-se que frequências menores que 30 kHz sofrem forte atenuação, superiores a 12 dB, e junto com o aparecimento de harmônicas se torna inviável a aquisição e a reprodução de sinais na faixa entre 0 e 30 kHz, como pode ser visto para a frequência de 19 kHz na Figura 4.3. O espectro mostra a presença da 3ª e 5ª harmônica, as quais não são geradas

no NCO, mas surgem devido a distorção causada. O motivo dessa distorção é a existência de transformadores de RF ADT1-1WT na entrada dos conversores A/D e também na saída dos conversores D/A, conforme pode ser visto nas Figuras 4.4 e 4.5. Estes transformadores tem resposta de frequência plana apenas na faixa entre 0,4 MHz a 800 MHz(MINI-CIRCUITS,). Originalmente pretendia-se utilizar estes conversores na faixa de áudio (de 20 Hz a 20 kHz), mas devido a esse problema isso não foi possível. Por esse motivo, os cenários de teste usados trabalham com sinais de frequências na faixa de 30 kHz a 6,25 MHz.

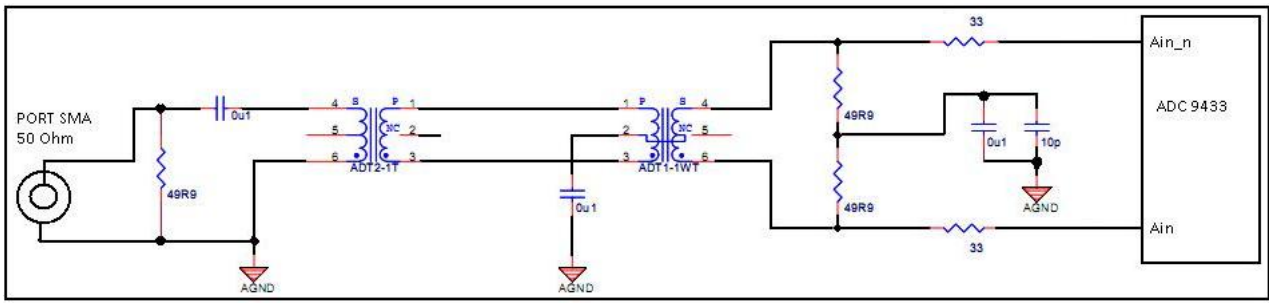


Figura 4.4: Digrama esquemático da entrada do conversor A/D.

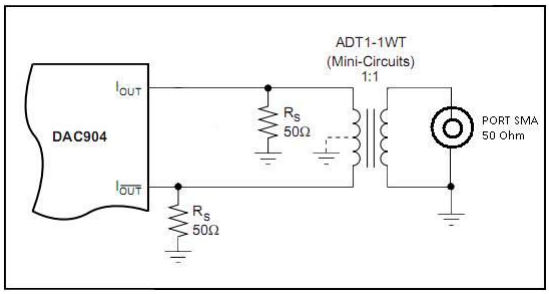


Figura 4.5: Digrama esquemático da saída do conversor D/A.

4.2 Memória de Armazenamento de Sinais

O circuito de armazenamento de sinal foi implementado de forma que um sinal do conversor A/D ou do gerador senoidal fosse sub-amostrado a uma taxa de 44,1 k amostras por segundo. Ao acionar o botão que ativa a escrita, 2 segundos de sinal são armazenados na memória para posterior processamento. O sinal armazenado corresponde a 88.200 amostras de 12 bits. Quando o botão de leitura é acionado, os dados armazenados são lidos e enviados para o conversor D/A que converte o sinal para analógico. O *hardware* utilizado (ver Tabela 4.2) para a implementação do cenário de armazenamento de sinais foi de 537 (1%) ALUTs, e 62% dos blocos de memória disponíveis no FPGA.

O circuito mostrado na Figura 4.6, foi sintetizado usando componentes em VHDL, os quais foram transformados em blocos e interligados usando o diagrama esquemático da ferramenta

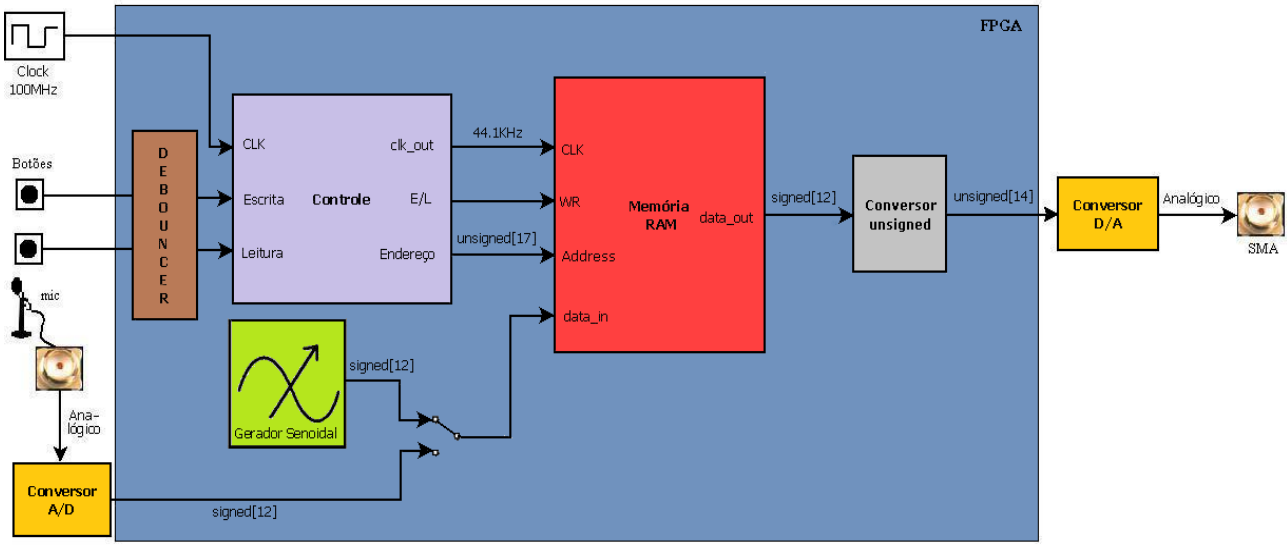


Figura 4.6: Diagrama do circuito de armazenamento de sinal.

Quartus II. A função de cada bloco implementado é descrita a seguir. Nessa descrição apenas os detalhes mais importantes são apresentados.

Tabela 4.2: *Hardware* utilizado no cenário de armazenamento de sinal.

Componente	Disponíveis	Utilizados	Percentagem de uso
ALUTs	48352	537	1%
Registradores Lógicos Dedicados	48352	439	1%
Bits de Memória RAM	2544192	1582080	62%
Blocos DSP	288	4	1%

- **Bloco de Controle**: Este bloco controla a leitura e escrita na memória RAM. Quando um dos botões é pressionado, um sinal de *clk_44* (clk_out) com frequência de 44,1 kHz é enviado para a entrada de relógio da memória RAM. A cada ciclo de *clk_44*, o valor de endereço inicializado com 00 é incrementado até chegar ao endereço máximo da memória 88200, retornando em seguida ao estado de espera. Ambos botões quando acionados ativam o processo de geração de endereços sequencias descrito acima, diferenciando apenas o valor que é enviado para o sinal E/L. Ao acionar o botão de escrita, o sinal E/L recebe o valor '1', e ao acionar o botão de leitura recebe o valor '0'.

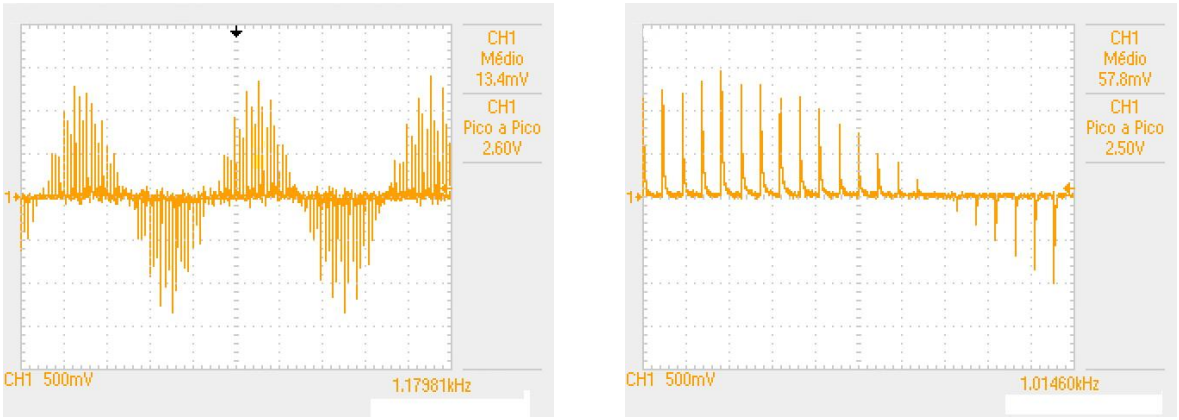
- **Circuito de anti-repique**: Utilizado para eliminar o repique de chave nos botões.

- **Conversor *Unsigned***: Este bloco recebe o sinal do tipo inteiro positivo ou negativo representado em complemento dois e o converte para inteiro positivo, pois é esse o formato do sinal de entrada do conversor D/A.

- **Sinal de entrada**: O sinal de entrada possui uma taxa de amostragem de 100 M amostras por segundo, podendo vir de um gerador interno ou através do conversor A/D.

Resultados Obtidos

A memória RAM armazenou e reproduziu corretamente os dois segundos de um sinal gerado pelo NCO, porém o transformador de RF presente na saída do conversor D/A produziu o decaimento do valor inicial de cada amostra reproduzida na taxa de 44,1 k. Por esse motivo, o sinal de saída do conversor comportou-se como impulsos, como pode ser visto nas figuras 4.7(a) e 4.7(b). Para comprovar que o sinal conectado na entrada do conversor D/A estava correto, foi realizado a simulação utilizando o Modelsim, conforme pode ser visto na Figura 4.8, e a captura do sinal no *chip* FPGA utilizando o SignalTap, Figura 4.9.



(a) Sinal reproduzido

(b) Zoom do sinal reproduzido mostrando o decaimento do valor da amostra.

Figura 4.7: Sinal medido na saída do conversor D/A com osciloscópio.

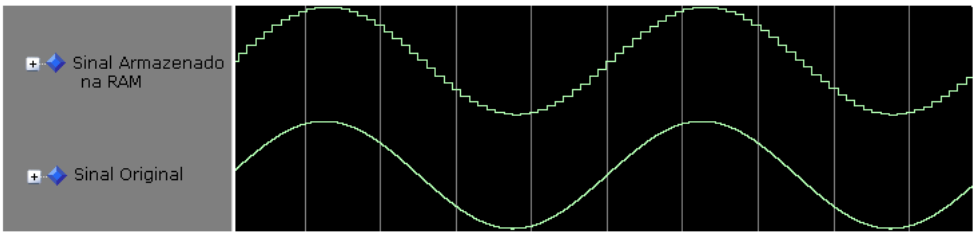


Figura 4.8: Sinal simulado obtido com o Modelsim.

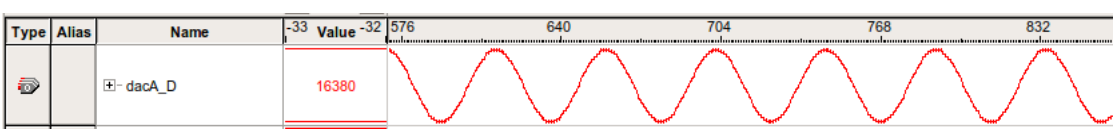


Figura 4.9: Sinal medido internamente no FPGA na entrada do conversor D/A com SignalTap.

4.3 Gerador de Sinais Senoidal

Com o intuito de implementar um gerador com frequência e amplitude ajustável e boa qualidade espectral, foi desenvolvido um circuito para controlar o funcionamento do IP NCO, utilizando os 4 botões disponíveis no kit. Estes botões são usados para controlar tanto a frequência quanto a amplitude de saída do gerador.

Todos os tipos de arquiteturas de NCO foram testadas, obtendo êxito com as arquiteturas baseada na ROM pequena, no algorítimo CORDIC e na arquitetura baseada em multiplicadores. A arquitetura ROM grande foi a única que mostrou-se impossível de implementar neste kit por necessitar de mais memória interna do que disponível no FPGA. Na Tabela 4.3, é possível notar a diferença de recursos utilizados pelas arquiteturas. Os parâmetros do NCO foram configurados de tal modo que o sinal de saída pudesse ser conectado a entrada do conversor D/A, mantendo uma boa precisão no valor em Hertz da frequência do gerador. Para isto, os parâmetros configurados na precisão de fase do acumulador (*phase accumulator precision*), precisão angular (*angular precision*) e precisão de magnitude (*magnitude precision*) foram 32, 16 e 14 respectivamente. As opções *frequency modulation input* e *Phase Modulation Input* não foram selecionadas.

Tabela 4.3: *Hardware* utilizado para criar o gerador senoidal.

Componente	Cordic	ROM Pequena	Baseada em Multiplicadores
ALUTs	2295	1077	958
Registradores Lógicos Dedicados	1727	420	333
Bits de Memória RAM	0	212 992	10752
Blocos DSP	4	4	12

Como indicado na Equação 3.2, é possível controlar a frequência gerada mudando o valor de entrada do sinal phi_inc(ϕinc)

$$\phi inc = \frac{fo}{f_{clk}} \times 2^M \tag{4.1}$$

Dessa forma, obtêm-se uma frequência de 6,5MHz neste kit quando: $\phi inc = \frac{6.25\times10^6}{100\times10^6} \times 2^{32} = 268435456$

A Equação 4.1 indica que para aumentar a frequência de saída do NCO é necessário aumentar o phi_inc da quantidade mostrada na Tabela 4.4. Para aumentar 1 Hz a frequência gerada, basta somar aproximadamente 43 ao valor que já havia no *phi_inc*. Para diminuir 10 Hz basta diminuir 429 no *phi_inc*, e assim por diante. É em cima deste conceito que é feito o controle de frequência na saída do NCO neste gerador.

Tabela 4.4: Relação entre o incremento do valor da frequência de saída do gerador e incremento do phi_inc.

Δfo	$\Delta\, phi\_inc$
1 Hz	43
10 Hz	429
100 Hz	4295
1 kHz	42950
10 kHz	429497
100 kHz	4294967
1 MHz	42949673

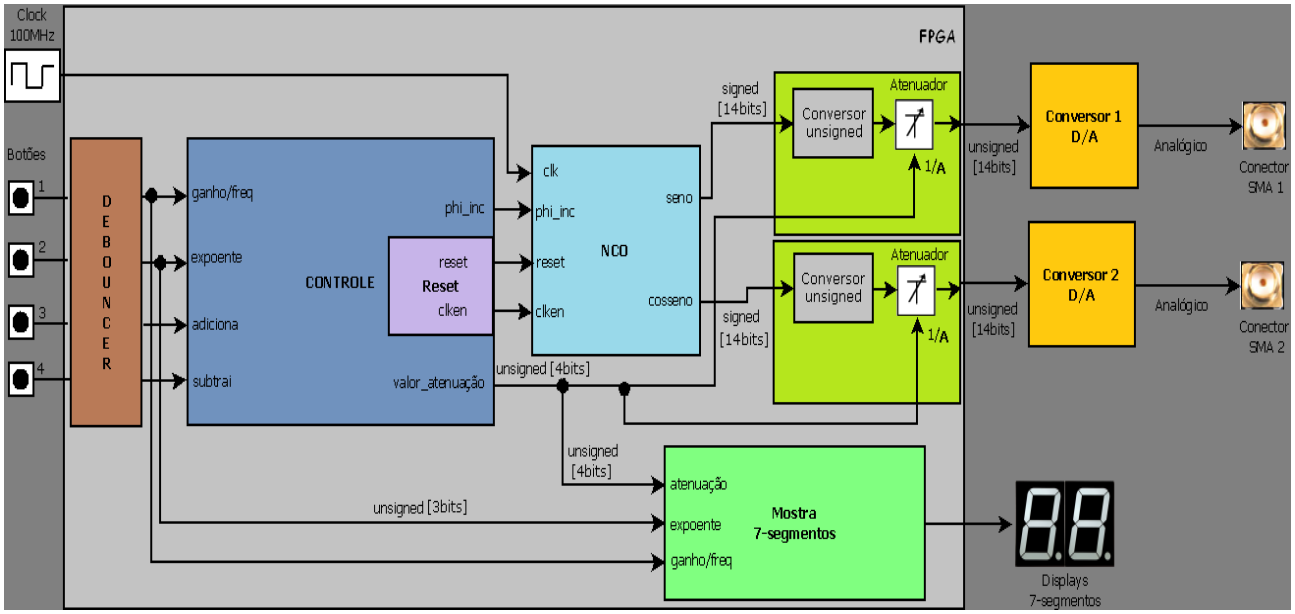


Figura 4.10: Diagrama do gerador de sinais senoidal.

O circuito implementado

Com o objetivo de transformar o gerador em algo prático, de uso simples, utilizamos os 4 botões (chaves *push-button*) junto com os dois mostradores (displays) de sete segmentos para controlar o ganho e a frequência (ver Figura 4.10). Adotamos um sistema de ajuste de frequência semelhante a usada em alguns geradores de frequência comerciais. O botão 1 quando acionado, seleciona a função ganho ou frequência (sinal ganho/freq). O botão 2 quando acionado, altera o expoente na base 10 do valor de frequência que será mudado. O valor do expoente decrementado ou incrementado varia de 0 até 6. Se por exemplo, o valor do expoente for 0, o valor que será mudado quando o Botão 3 ou o Botão 4 for acionado será 10^0 =1 Hz. Se o valor for 6, será mudado $10^6 = 1$ MHz. A dupla de botões 3 e 4 é usada para aumentar (3) ou diminuir (4) o valor da frequência de acordo com a base escolhida no botão 2, ou o ganho indicado no mostrador.

Para indicar o valor do expoente e do ganho, foram usados dois mostradores de sete-

segmentos. Quando o valor do sinal ganho/freq for 0 (habilita alterar a frequência), será mostrado nos *displays* o valor que está no expoente. Quando o valor do sinal ganho/freq for 1 (habilita alterar o ganho) o valor do ganho dividido por 10 será mostrado nos *displays*. O valor do ganho varia de 1 até 10, sempre uma unidade por vez. Esse valor dividido por 10, multiplica a saída do NCO, alterando a amplitude do sinal gerado.

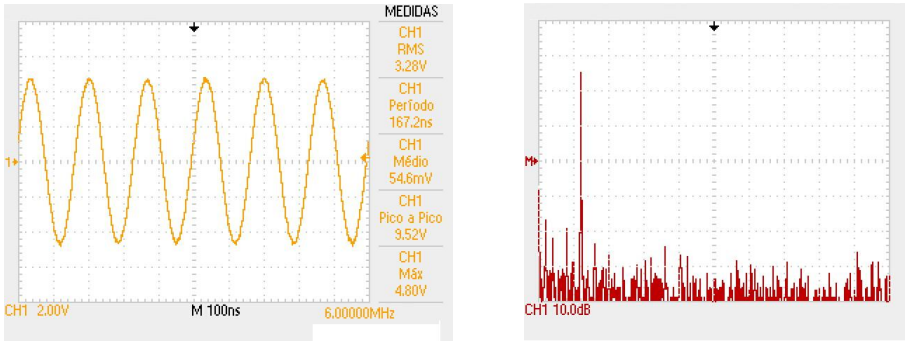
Se por exemplo, um sinal inicial possui 2 MHz e ganho de 1 e se quer alterá-lo para 970 kHz com ganho de 0,6, é possível realizar o seguinte procedimento: Ativar o Botão 2 até que o valor do expoente seja 6 (10^6 = 1 MHz). Ativar o Botão 4 uma vez, para que seja subtraído 1MHz na frequência gerada, resultando em um sinal de 1 MHz. Ativar novamente o Botão 2 até que o valor do expoente seja 4 (10^4= 10 kHz). Ativar o Botão 4 três vezes, para que seja subtraído 30 kHz na frequência gerada, resultando o sinal de 970 kHz. Agora basta mudar o valor do ganho acionando o botão 1 para alterar para função ganho, e acionar o botão 4 até que o valor 0,6 apareça nos displays.

O sistema foi implementado utilizando o diagrama esquemático da ferramenta Quartus. Cada bloco foi implementado ou diretamente em VHDL (bloco de Controle, Conversão para *Unsigned*, Mostra_7segmentos, Circuito de anti-repique) ou usando IPs do MegaWizard da Altera (NCO). Os blocos implementados com suas funções e funcionamento são descritos a seguir.

O bloco Controle realiza a programação do NCO através dos sinais *phi_inc*, *reset_n* e *clken*, além de atuar sobre o bloco de ganho, no qual a amplitude de saída é definida. O bloco Conversão para *unsigned* muda a representação das amostras do tipo inteiro em complemento dois (*signed*) para inteiro positivo (*unsigned*). O bloco Mostra_7segmentos recebe os sinais de expoente e ganho e os mostra nos *displays* conforme o valor correspondente ao sinal ganho/freq. O Circuito de anti-repique é utilizado para eliminar o repique de chave nos botões. O NCO é um IP implementado conforme parâmetros descritos anteriormente.

Resultados Obtidos

O gerador mostrou-se funcional e com uma ótima precisão de frequência e de estabilidade na saída do conversor D/A. O sistema de controle atendeu ao que se buscava ser feito, sendo possível, de uma maneira rápida, ajustar a frequência do sinal gerado com uma precisão realmente alta e pouca oscilação para a faixa de 30 kHz à 6,25 MHz. A Figura 4.11 apresenta um sinal gerado de 6 MHz e seu espectro.



(a) Sinal no domínio do tempo (b) FFT do sinal

Figura 4.11: Captura de tela do osciloscópio de um sinal 6 MHz.

4.4 Filtro FIR

O cenário realizado é mostrado na Figura 4.12. Na entrada do filtro a ser testado é possível conectar a soma de duas senoides com frequências diferentes, como pode ser visto na Figura 4.13. Tanto o sinal de entrada como a saída do filtro são conectados aos conversores D/A para possibilitar a sua medição externa comparando os dois sinais e consequentemente o funcionamento do filtro.

Neste circuito, dois botões foram usados como chave para os multiplexadores que enviam a senoidal ou um apenas zero para o bloco somador. O resultado disso é poder ativar ou desativar a saída dos geradores de frequência.

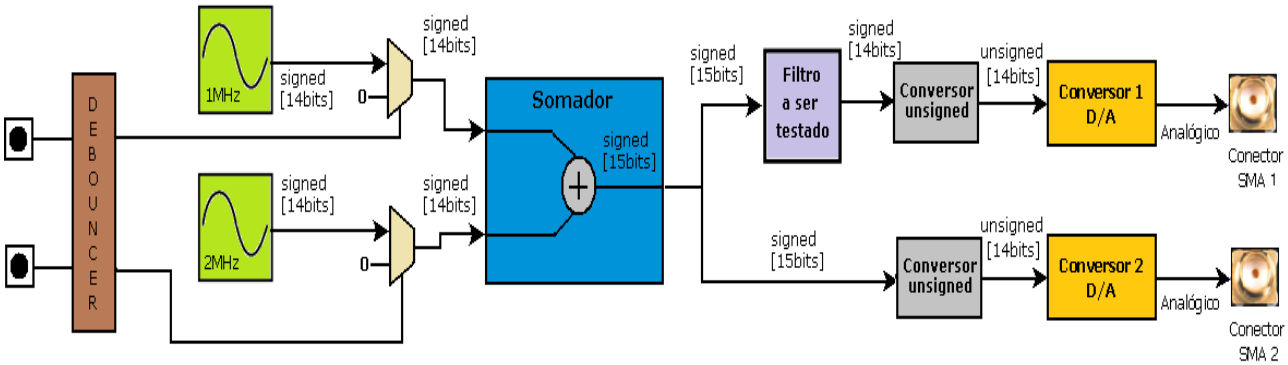
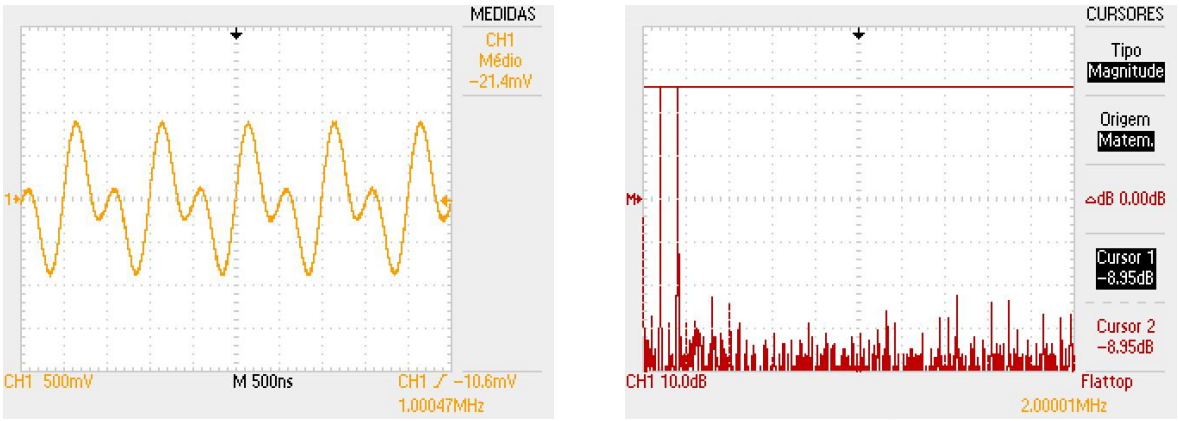


Figura 4.12: Cenário de testes dos filtros.

Da mesma forma que o cenário anterior, este também foi implementado utilizando o diagrama esquemático da ferramenta Quartus II. Cada bloco foi implementado ou diretamente em VHDL (Multiplexadores, Somador, Circuito de anti-repique, Conversor para *unsigned*) ou usando IPs do MegaWizard da Altera (Gerador Senoidal, Filtro FIR). Os blocos implementos com suas funções e funcionamento são descritos a seguir.

O cenário possui dois geradores senoidal implementados utilizando o NCO Altera com arquitetura ROM pequena, de forma a permitir gerar duas frequências diferentes. Os Multiplexadores são usados para ativar ou desativar a saída dos geradores quando os botões são acionados. O bloco Somador recebe os dois sinais de 14 bits vindos dos geradores e os soma resultando um sinal de 15 bits. O Circuito de anti-repique (*debouncer*) foi utilizado para eliminar o repique de chave nos botões. O bloco Conversor para *unsigned* muda a representação das amostras do tipo inteiro em complemento de dois (*signed*) para inteiro positivo (*unsigned*). Além disso, o bloco Conversor para *unsigned*, elimina o bit menos significativo do sinal de 15 bits vindo diretamente do somador.

Foram testados filtros passa baixa, passa alta e um rejeita faixa. Os filtros foram implementados utilizando o FIR Compiler e projetados utilizando uma janela retangular. O bloco de Filtro passa baixa possui ordem 149 e frequência de corte em 1,1 MHz. O bloco de Filtro passa alta possui ordem 149 e frequência de corte em 1,5 MHz. O rejeita faixa possui ordem 199 e frequências de corte em 0,6 MHz e 1,25 MHz. O funcionamento do filtro passa baixa de 1,1 MHz pode ser verificado comparando o sinal de entrada do filtro (ver Figura 4.13) com o sinal de saída do filtro (Figura 4.14). Tanto no sinal no domínio do tempo como na FFT calculada pelo osciloscópio pode-se ver que a frequência de 2MHz está atenuada (26 dB), a frequência de 1 MHz é transmitida pelo filtro sem perda. Analisando a Tabela 4.5 é possível notar que ocorre uma diferença no número de elementos gastos até mesmo nos filtros de mesma ordem.



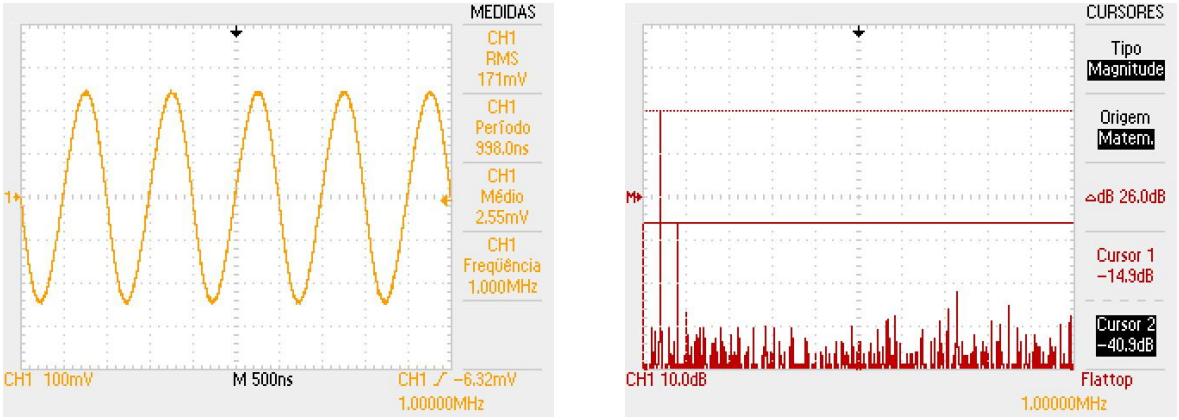
(a) Sinal no domínio do tempo

(b) FFT do sinal

Figura 4.13: Captura de tela do osciloscópio de um sinal com duas frequências.

Tabela 4.5: *Hardware* utilizado para testar os filtros FIR.

Componente	Rejeita-Faixa 200	Passa-Baixa 150	Passa-Alta 150
ALUTs	9178	8274	6403
Registradores Lógicos Dedicados	10288	8699	7488
Bits de Memória RAM	426191	426245	426155



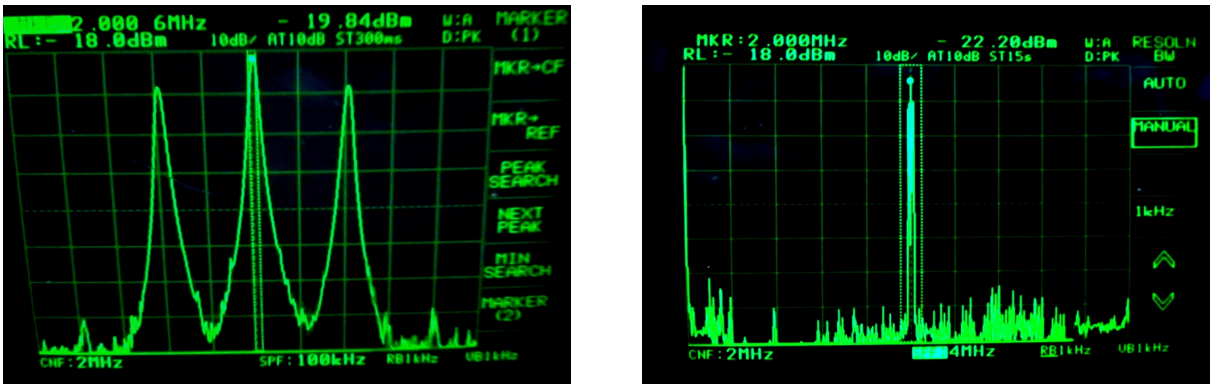
(a) Sinal no domínio do tempo

(b) FFT do sinal

Figura 4.14: Captura de tela do osciloscópio de um sinal filtrado por um passa baixa de ordem 149 com frequência de corte em 1,1 MHz .

4.5 Misturador de Frequência

Para testar o circuito misturador de frequências foi utilizado o cenário mostrado na Figura 4.16. Neste, o misturador recebe um sinal modulado AM/DSB através de um conversor A/D, e o sinal do gerador senoidal interno. O sinal modulado é uma senoide de 20 kHz sobre uma portadora de 2 MHz (ver Figura 4.15) proveniente do gerador de função Minipa MFG-4201A. O gerador interno ajustável produz uma senoide com frequência inicial de 1 MHz. Na saída do multiplicador, são obtidos os sinais $Fr + Fo = 3MHz$ e $Fr - Fo = 1MHz$, visto que $Fr = 2MHz$ e $Fo = 1MHz$. Neste cenário escolheu-se a FI em 1 MHz, utilizando um passa-baixas e um passa-faixa para realizar a eliminação do sinal na faixa de 3 MHz. Estes dois filtros foram utilizados apenas com o intuito de testa-los neste cenário, pois na prática apenas um filtro é necessário.



(a) Janela de observação de 100 kHz.

(b) Janela de observação de 4 MHz

Figura 4.15: Sinal senoidal de 20kHz modulado em AM/DSB por portadora de 2 MHz.

O cenário mostrado na Figura 4.16 foi implementado utilizando o diagrama esquemático da ferramenta Quartus II. Cada bloco foi implementado ou diretamente em VHDL (Multiplicador,

Conversor para *unsigned*) ou usando IPs do MegaWizard da Altera (Gerador de Sinais Senoidal, Filtros FIR). Este sinal modulado foi conectado ao conversor A/D. Os principais blocos implementados com suas funções e funcionamento são descritos a seguir.

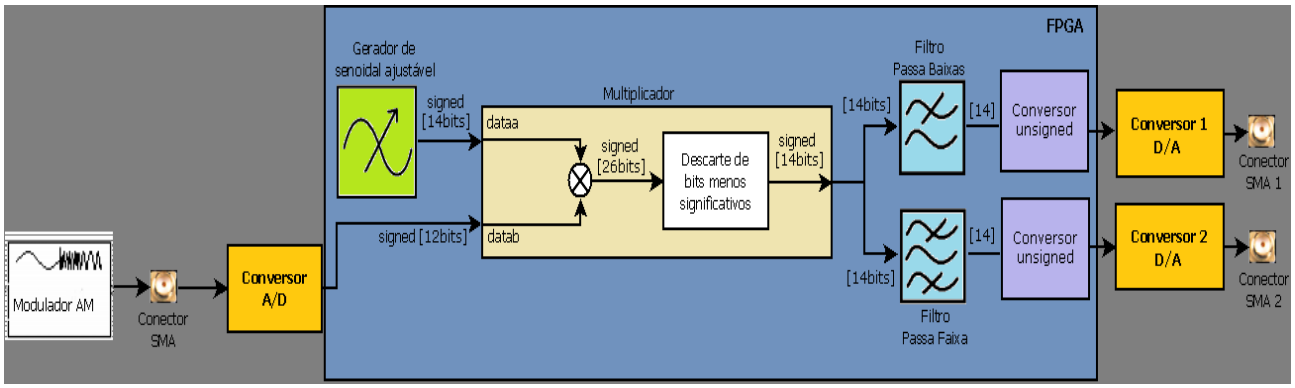


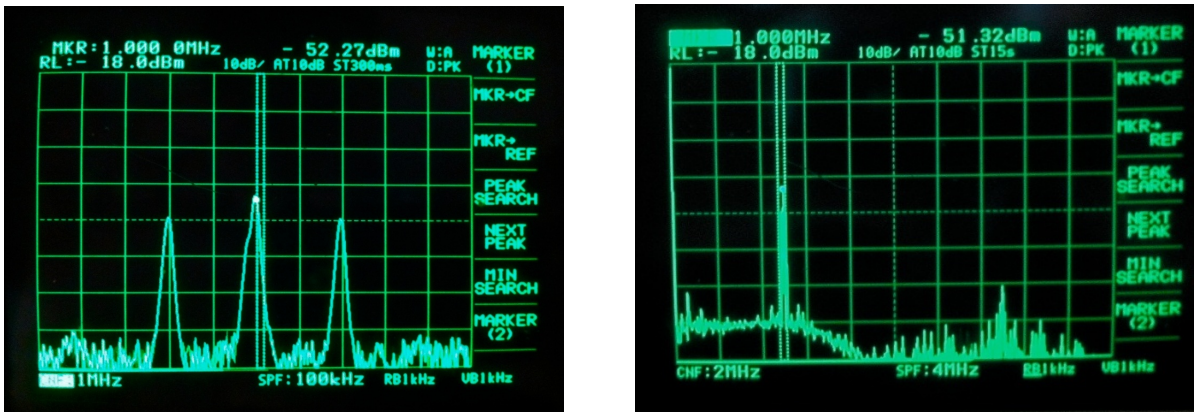
Figura 4.16: Diagrama do misturador de frequência.

No diagrama que ilustra o cenário de testes do misturador de frequência (ver Figura 4.16), a função de cada bloco é: O Gerador de Sinais Senoidal gera um sinal que multiplica o sinal modulado; O bloco Multiplicador realiza a multiplicação do sinal modulado com a senoide do gerador; O bloco Conversão para *unsigned* muda a representação das amostras do tipo inteiro em complemento dois (*signed*) para inteiro positivo (*unsigned*). Os filtros foram implementados utilizando o FIR Compiler e projetados usando uma janela retangular. O bloco de filtro passa baixa possui ordem 149 e frequência de corte em 1,5 MHz. O bloco de filtro passa faixa possui ordem 349 e frequências de corte em 0,8 MHz e 1,2 MHz.

Analisando a Tabela 4.6 é possível notar que os filtros implementados, utilizaram um grande número de registradores lógicos e ALUTs. Nota-se na Figura 4.17(b) que a componente de 3 MHz existente na saída do misturador de frequência foi fortemente atenuada na saída do filtro passa baixa. Ao mesmo tempo nota-se que o conteúdo do sinal modulado permanece com as duas bandas laterais e a portadora com (18dBm) como no sinal de entrada (ver Figura 4.15).

Tabela 4.6: *Hardware* utilizado para implementar o misturador.

Componente	Disponíveis	Utilizados	Percentagem de uso
ALUTs	48352	25406	53%
Registradores Lógicos Dedicados	48352	26550	55%
Bits de Memória RAM	2544192	213474	8%
Blocos DSP	288	2	1%



(a) Janela de observação de 100 kHz.

(b) Janela de observação de 4 MHz

Figura 4.17: Sinal modulado em AM/DSB deslocado pelo misturador para FI de 1MHz.

5 *Conclusões e Trabalhos Futuros*

Este trabalho teve como objetivo criar uma plataforma para processamento de sinais em dispositivos FPGA (PS-FPGA), sendo então uma ferramenta didática que servirá como suporte a trabalhos futuros, possibilitando implementar em tempo real, sistemas desenvolvidos no ambiente de desenvolvimento Matlab/Simulink envolvendo o processamento de sinais. Além disso, visou-se o processo de desenvolvimento dos módulos que constituem a plataforma, optando por diferentes abordagens sendo utilizada a linguagem VHDL, IPs, diagrama esquemático e sinais de entrada e saída externos à plataforma. Os módulos foram testados em cenários que são frequentemente utilizados na área de telecomunicações, fazendo o uso de ferramentas de análise tais como osciloscópio e analisador de espectro.

Foi possível notar durante o desenvolvimento do projeto, a grande vantagem de se utilizar um dispositivo FPGA, que é a sua flexibilidade, pois em um mesmo dispositivo, foi possível realizar diferentes cenários obtendo os resultados esperados. Com a plataforma para processamento de sinais desenvolvida, foi possível implementar em ambiente prático, sistemas que no meio acadêmico são normalmente apenas simulados.

Durante o trabalho houve uma dificuldade inicial em relação ao uso do conversor D/A e A/D devido à existência de transformadores de RF ADT1-1WT que impedem o uso desses conversores para sinais de áudio. Porém, utilizando sinais em uma faixa de frequência adequada foi possível realizar as implementações e testes com êxito. Ao final do projeto, percebe-se que os objetivos foram alcançados, pois tanto a abordagem utilizada como o produto e possibilidades de interferência externa podem ser vistas através dos diversos cenários desenvolvidos.

O resultado deste projeto é uma plataforma didática que permite explorar em aulas tanto os conceitos como circuitos de processamento de sinais em tempo real. No entanto a PS-FPGA está ainda em fase embrionária necessitando diversas melhorias e adição de módulos. Por isso como sugestão de continuidade deste trabalho são propostos alguns temas que utilizam como base atual da plataforma PS-FPGA, tais como:

- Criação de um bloco para controle do CODEC de áudio disponível no kit para aquisição e reprodução de sinais na faixa de áudio, evitando assim a distorção causada pelos transformadores de RF quando os conversores A/D e D/A são usados;

- Implementação em FPGA de ferramentas de análise como o osciloscópio e analisador de espectros utilizando alguns módulos já desenvolvidos como o misturador, filtros seletivos e aquisição de sinais. Neste caso seria necessário desenvolver adicionalmente um módulo para a apresentação dos dados na tela através da interface VGA;

- Criação de módulos adicionais que implementam funções como: correlação, convolução, interpolação, filtros IIR. Estes forneceriam a base para outras aplicações em processamento de sinais;

- Transformar o gerador de sinais obtido em um equipamento real, acrescentando circuitos adicionais de saída, chaves e mostradores auxiliares. Neste caso possivelmente a migração para um kit de menor custo (DE0-nano) ou a produção de uma placa de circuito impresso sejam necessárias devido a questão de custo final do gerador.

Certamente este projeto não conclui o tema pesquisa, mas serve como uma plataforma inicial para todos os temas acima propostos, podendo no futuro através do acréscimo de novos módulos resultar em um produto didático a ser usado em salas de aula para o ensino de telecomunicações.

APÊNDICE A – Uso do NCO

O NCO Altera pode ser adicionado à um projeto no DSP Builder, ou então no Quartus II utilizando a ferramenta *MegaWizard*. Esse segundo caso pode ser feito da seguinte maneira:

1. Abra o MegaWizard no Quartus II.
2. Na janela que abrir selecione a opção: *Create a new custom megafunction variation* e aperte *Next*.
3. O NCO pode ser encontrado na aba *DSP/Signal Generation*. Selecione o NCOv10.0 e o tipo de HDL que se deseja usar. Após dar um nome para o arquivo que será gerado, clique em *Next*.
4. Aparecerá agora uma janela com cinco botões. Clique no botão: *Step1 Parameterize*. Uma janela nova irá aparecer. Após fazer todas as parametrizações desejadas, clique em *Finish*.
5. Na janela com os cinco botões, clique em *Step 2 Set Up Simulation*. Caso seja necessário fazer a simulação, marcar a opção *Generate Simulation Model*. Importante ressaltar que os arquivos gerados no caso de simulação, NÃO SÃO SINTETIZÁVEIS.
6. Basta agora clicar no *Step 3 Generate*. Os arquivos serão gerados. Após ler o relatório, clique em *exit*.
7. Uma janela aparecerá no Quartus II perguntando se deseja adicionar os arquivos ao projeto. Adicione.

APÊNDICE B – O uso do FIR Compiler v10.1

O FIR Compiler pode ser adicionado à um projeto no Quartus II utilizando a ferramenta MegaWizard da seguinte maneira:

1. Abra o MegaWizard no Quartus II.
2. Na janela que abrir selecione a opção: *Create a new custom megafunction variation* e aperte *Next*.
3. O FIR Compiler v10.1 pode ser encontrado na aba DSP/Filters. Selecione o FIR Compiler v10.1 e o tipo de HDL que se deseja usar. Após dar um nome para o arquivo que será gerado, clique em *Next*.
4. Aparecerá agora uma janela com cinco botões. Clique no botão: *Step1 Parameterize*. Uma janela nova irá aparecer. Nesta janela será feita a parametrização do filtro. Primeiramente é necessário ajustar a resposta desejada para o filtro. Isso é possível clicando em *Edit Coefficient Set*. Uma terceira janela irá abrir, e nela encontram-se as opções de filtros, tipos de janelas, frequência de amostragem do sinal de entrada e as frequências de corte desejadas. Para importar os coeficientes de um arquivo, basta marcar a opção *Imported Coefficient Set* e escolher o arquivo onde os coeficientes desejados estão salvos. Este arquivo deve conter em cada linha o valor do coeficiente sem espaços, podendo o valor ser representado na forma de ponto-flutuante, ponto-fixo ou notação científica. Após realizar as parametrizações necessárias, basta clicar em *Ok*.
5. Os próximos passos correspondem ao tratamento dos sinais de entrada e saída. Em *Input Specification*, é possível configurar o número de canais, a quantidade de bits de entrada e a forma de representação deste sinal. Em *Ouput Specification* é possível escolher a quantidade de *bits* que será mantida na saída, podendo especificar se os bits descartados serão apenas eliminados, arredondados ou saturados. O sinal de saída pode ser representado no formato *signed* ou *signed* fracional.
6. A opção *Structure* configura qual tipo de arquitetura será usada. Realizado todo esse procedimento, clique em *Finish* para salvar as alterações.

7. Na janela com os cinco botões, clique em *Step 2 Set Up Simulation*. Caso seja necessário fazer a simulação, marcar a opção *Generate Simulation Model*. Importante ressaltar que os arquivos gerados no caso de simulação, NÃO SÃO SINTETIZÁVEIS. Também é possível gerar arquivos de teste para o Matlab e Testbench.

8. Basta agora clicar no *Step 3 Generate*. Os arquivos serão gerados. Após ler o relatório, clique em *exit*.

9. Uma janela aparecerá no Quartus II perguntando se deseja adicionar os arquivos ao projeto. Adicione.

APÊNDICE C – Configuração dos Conversores

C.1 Taxa de amostragem dos conversores A/D

A frequência de amostragem a ser usada pelo conversor é configurável através de *jumpers* encontrados no kit, que devem ser colocados entre os pinos indicados na Tabela C.1. O *jumper* J3 configura o relógio do conversor *adc_A* enquanto o *jumper* J4 configura o relógio do conversor *adc_B*.

Tabela C.1: Controle de frequência de amostragem do conversor A/D.

Pinos	Origem do relógio
1 e 2	Pinos PLLs B18 e D18
3 e 4	Oscilador de 100 MHz positivo
5 e 6	Oscilador de 100 MHz negativo

C.2 Taxa de reprodução dos conversores D/A

A frequência que as amostras são reproduzidas pelo conversor é configurável através de *jumpers* do kit, que devem ser colocados entre os pinos indicados na Tabela C.2. O *jumper* J18 configura o relógio do conversor *dac_A* enquanto o *jumper* J19 configura o relógio do conversor *dac_B*.

Tabela C.2: Controle de frequência de amostragem do conversor D/A.

Pinos	Origem do relógio
1 e 2	Pinos PLLs B15 e C16
3 e 4	Pinos PLLs C15 e D16
5 e 6	Oscilador de 100 MHz positivo
7 e 8	Entrada externa pelo pino J12

ANEXO A – Kit DSP EP2S60

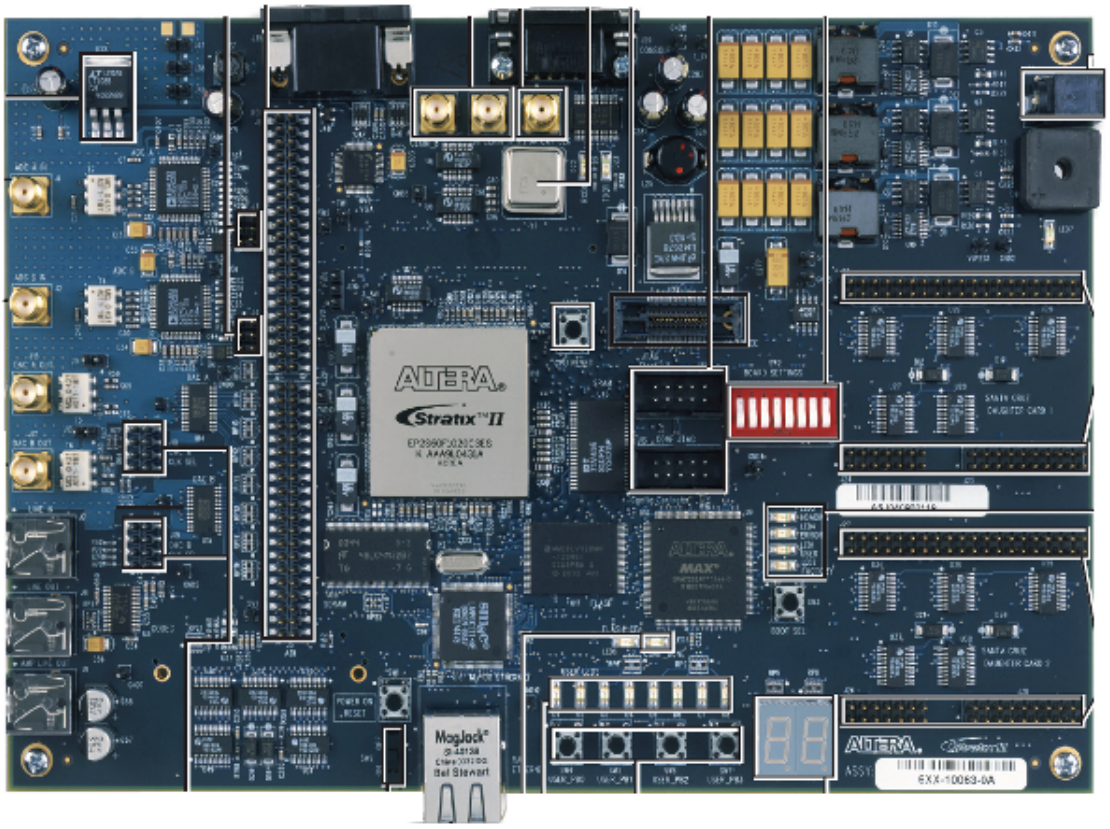


Figura A.1: Kit de desenvolvimento DSP EP2S60 - Stratix II Edition (ALTERA, 2004).

Lista de Abreviaturas

A/D Analógico/Digital

ABEL Linguagem de Expressão Booleana Avançada (*Advanced Boolean Expression Language*)

AHDL Linguagem de Descrição de Hardware Altera (*Altera Hardware Description Language*)

ALUT Tabela Adaptativa de Consulta (*Adaptative Look-Up Table*)

AM Modulação em Amplitude (*Amplitude Modulation*)

CORDIC (*COordinate Rotation DIgital Computer*)

D/A Digital/Analógico

DCT Transformada Discreta de Cosseno (*Discrete Cosine Transform*)

DFT Transformada Discreta de Fourier (*Discrete Fourier Transform*)

DSP Processador Sinais Digitais (*Digital Signal Processor)*

FIFO Memória onde o primeiro dado a entrar será o primeiro a sair. (*First In, First Out*)

FIR Resposta ao Impulso Finita (*Finite Impulse Response*)

FM Modulação em Frequência (*Frequency Modulation*)

FPGA Arranjo de Portas Programável em Campo (*Field-programmable gate array*)

HDL Linguagem de Descrição de Hardware (*Hardware Description Language*)

IIR Resposta ao Impulso Infinita (*Infinite Impulse Response*)

IP Propriedade Intelectual (*Intellectual Property*)

ISP Processador de Conjunto de Instruções (*Instruction Set Processor*)

LE Elementos Lógicos (*Logic Element*)

LED Diodo Emissor de Luz (*Light Emitting Diode*)

LUT Tabela de Consulta (*Look-Up Table*)

NCO Oscilador Controlado Numericamente (*Numerically-controlled oscillator)*

PLL Malha de Captura de Fase (*Phase-locked loop*)

PM Modulação em Fase (*Phase Modulation*)

RAM Memória de Acesso Aleatório (*Random Access Memory*)

RF Rádio Frequência

ROM Memória Apenas de Leitura (*Read Only Memory*)

VGA *Video Graphics Array*

VHDL Linguagem de Descrição de Hardware de Circuitos Integrados de Muito Alta Velocidade (*Very High Speed Integrated Circuit Hardware Description Language*)

Referências Bibliográficas

ALTERA. *Stratix II EP2S60 DSP Development Board*. [S.l.], 2004. Disponível em <http://www.datasheetcatalog.org/datasheets2/39/3908991_1.pdf> acesso em: julho, 2011.

ALTERA. *Datasheet Stratix II FPGA Family*. [S.l.], 2007. Disponível em <http://www.altera.com/literature/hb/stx2/stx2_sii51001.pdf> acesso em: julho, 2011.

ALTERA. *FIR Compiler - User Guide*. [S.l.], 2011. Disponível em <http://www.altera.com/literature/ug/fircompiler_ug.pdf> acesso em: julho, 2011.

ALTERA. *NCO MegaCore Function - User Guide*. [S.l.], 2011. Disponível em <http://www.altera.com/literature/ug/ug_nco.pdf> acesso em: julho, 2011.

ANDRAKA, R. *A survey of CORDIC algorithms for FPGA based computers*. New York, NY, USA, 1998. 191–200 p. (FPGA '98). Disponível em: <http://doi.acm.org/10.1145/275107.275139>.

ESPINDOLA B. M.; MOECKE, M. *Resumo: Plataforma em FPGA para Processamento de Voz com Wavelet*. [S.l.], 2010. Disponível em <http://jornadacientificasul.blogspot.com/2010/12/anais-da-iii-jornada-ja-estao.html> acesso em: julho, 2011.

LATHI, B. P. *Modern Digital and Analog Communication*. Terceira edição. [S.l.]: Oxford University Press, 1998.

MINI-CIRCUITS. *Datasheet RF Transformer*. [S.l.]. Disponível em <http://www.minicircuits.com/pdfs/ADT1-1WT.pdf> acesso em: julho, 2011.

PEDRONI, V. A. *Digital Electronics and Design with VHDL*. [S.l.]: The MIT Press, 2007.

RAMIREZ, S. R.; MCCLOUD, S. *Use ESL synthesis techniques to replace dedicated DSPs with FPGAs*. [S.l.], 2007. Disponível em <http://www.eetimes.com/design/embedded/4006823/Use-ESL-synthesis-techniques-to-replace-dedicated-DSPs-with-FPGAs> acesso em: julho, 2011.

SHENOI, B. A. *Introduction to Digital Signal Processing and Filter Design*. [S.l.]: Wiley-Interscience, 2006.