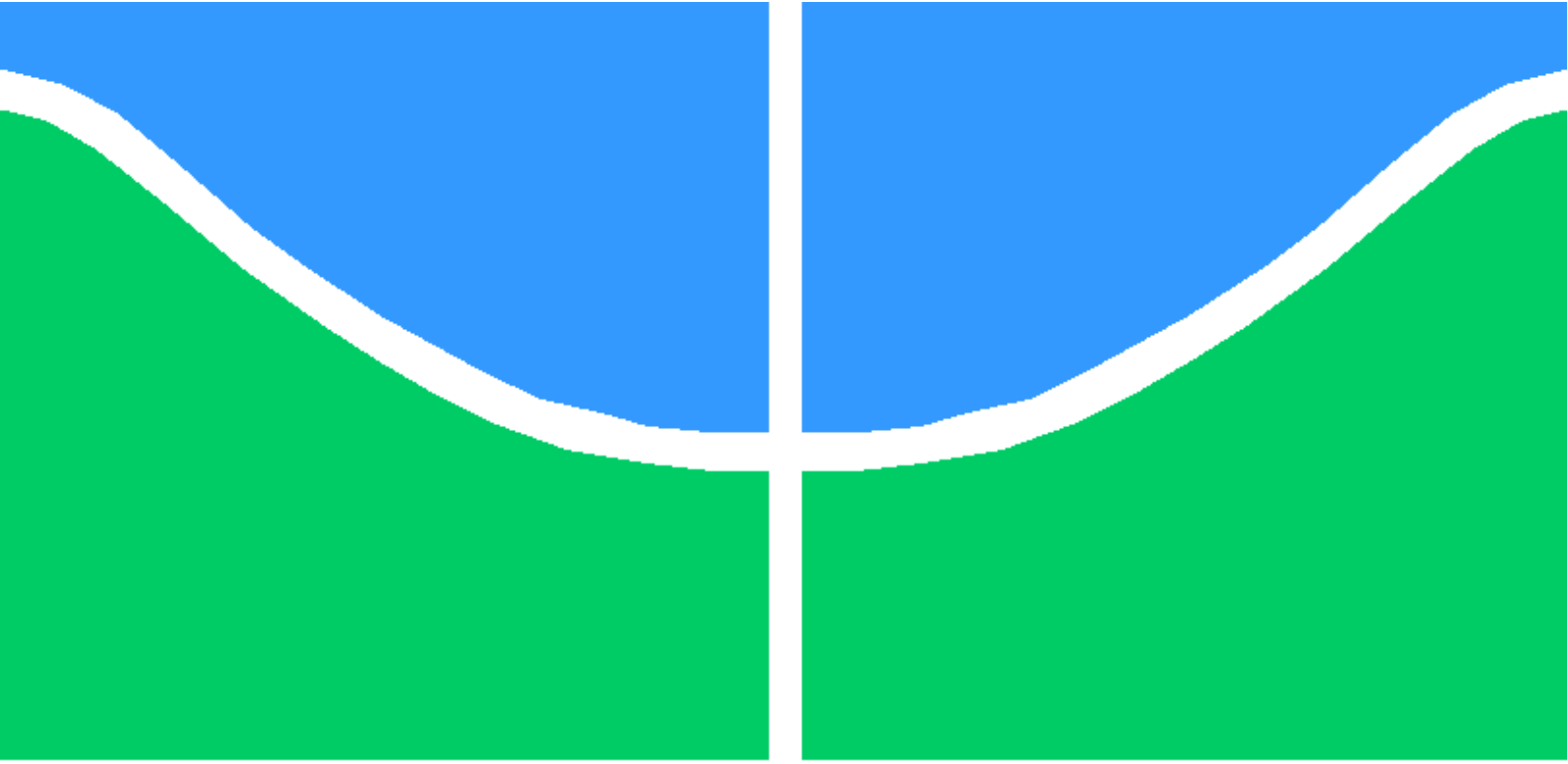

TRABALHO DE GRADUAÇÃO

DESENVOLVIMENTO DE UM DIGITAL DOWN CONVERTER (DDC) PARA UM PROTÓTIPO EMBARCADO DE RÁDIO DEFINIDO POR SOFTWARE

Rafael Schena

Brasília, julho de 2007

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia

TRABALHO DE GRADUAÇÃO

DESENVOLVIMENTO DE UM DIGITAL DOWN CONVERTER (DDC) PARA UM PROTÓTIPO EMBARCADO DE RÁDIO DEFINIDO POR SOFTWARE

Rafael Schena

Relatório submetido ao Departamento de Engenharia
Elétrica como requisito parcial para obtenção
do grau de Engenheiro Eletricista

Banca Examinadora

Prof. Adson Ferreira da Rocha - Doutor, UnB/ Dep (Orientador) _____

Prof. Eduardo Wolski - Mestre, UnB/ Dep (Co-Orientador) _____

Prof. Alexandre Ricardo Soares Romariz - Ph.D., UnB/ Dep _____

Prof. Janaína Gonçalves Guimarães - Doutora, UnB/ Dep _____

Dedicatória

Dedico aos meus pais, Paulo e Ivani, e aos meus irmãos, Gabriel, Emerson e Cristiane.

Rafael Schena

Agradecimentos

Agradeço, primeiramente, a Deus, que me deu forças, ânimo e, principalmente, saúde ao longo dos anos em que estive nesta Universidade. Agradeço a meus pais, a quem devo tudo nesta vida. Gostaria que eles soubessem que todo o meu esforço não teria razão de ser se não fosse para honrar tudo o que eles fizeram e ainda fazem por mim. Aos meus irmãos e irmã, por todo o apoio e carinho dedicados a mim. Agradeço aos grandes amigos que fiz durante estes anos em que estive nesta Universidade, cujos nomes não cito aqui pois certamente seria injusto com algum deles. Amigos estes que estiveram sempre ao lado durante a caminhada, tanto nos momentos difíceis como nos momentos felizes. Aos amigos e companheiros de trabalho do grupo de Rádio Definido por Software, em especial ao colega e amigo Franciso Augusto Garcia, que foi quem me convidou para fazer parte deste grupo, e desde então tem se mostrado sempre um grande incentivador. Ao professor Eduardo Wolski, meu orientador, com quem tive a honra de trabalhar e aprender muito durante estes anos de trabalho.

Rafael Schena

RESUMO

O objetivo central deste trabalho é o desenvolvimento de um *Digital Downconverter* para ser integrado a um protótipo embarcado de Rádio Definido por Software. Assumindo a função de realizar o papel de transladar para banda-base os sinais digitalizados entregues pelo conversor AD, para diminuir a taxa de dados entregue ao processador do terminal de rádio, o *Digital Downconverter* possibilita o processamento destes sinais de radiofreqüência por dispositivos de menor capacidade de processamento e menor consumo de potência, geralmente utilizados em sistemas embarcados. Neste trabalho, são apresentados o projeto e os testes de validação do *Digital Downconverter* implementado em FPGA.

ABSTRACT

The main goal of this work is to develop a Digital Downconverter to be incorporated to a embedded Software Defined Radio prototype. The role of a Digital Downconverter is to translate the digitized signal delivered by the analog-to-digital converter to baseband and to reduce the data rate to the radio processor, enabling the digital processing of the radiofrequency signals with devices of low-processing capacity and low-power consumption, commonly used in embedded systems. In this work, the project and validation tests of the Digital Downconverter assembled in FPGA are presented.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIMBOLOS

Siglas

ADC - *Analog-to-Digital Converter* (Conversor analógico-digital)

ASIC - *Application-Specific Integrated Circuit*

CORBA - *Common Object Request Broker*

CORDIC - *Coordinate Rotation Digital Computer*

DAC - *Digital-to-Analog Converter* (Conversor digital-analógico)

DDC - *Digital Downconverter*

DSP - *Digital Signal Processor* (Processador Digital de Sinais)

DUC - *Digital Upconverter*

ENOB - *Effective Number of Bits* (Número Efetivo de Bits)

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

FPGA - *Field Programmable Gate Array*

GPS - *Global Positioning System* (Sistema de Posicionamento Global)

IPTV - Televisão sobre IP

JTRS - *Joint Tactical Radio System* (Sistema de Rádio Tático Comum)

NCO - *Numerically Controlled Oscilator* (Oscilador Controlado Numericamente)

RDS - Rádio Definido por Software

RISC - *Reduced Instruction Set Computer*

RF - Radiofreqüência

ROM - *Read-Only Memory*

SCA - *Software Comunnication Architecture*

SOPC - *System On a Programmable Chip*

VoIP - Voz sobre IP

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

Assistiu-se, durante as duas últimas décadas, a um crescimento vertiginoso do acesso e da demanda por informação. O advento da computação pessoal, a melhoria dos serviços de telefonia fixa com a introdução da fibra óptica, o surgimento e a popularização da Internet e o desenvolvimento da telefonia móvel celular são exemplos de novos serviços que revolucionaram o modo de vida de grande parte da população mundial. Seja no trabalho ou simplesmente para entretenimento, a cada dia necessita-se de uma maior quantidade e qualidade de informações recebidas. E não só isso: há também a necessidade de poder comunicar-se com o restante de mundo em qualquer lugar e a qualquer hora. Atualmente, serviços via satélite permitem que, em qualquer região do planeta, seja possível assistir televisão, receber chamadas telefônicas, ou então obter as coordenadas geográficas do ponto onde se está por meio de um GPS, desde que observadas as condições mínimas para a recepção de um sinal de radiofreqüência.

E os avanços nas tecnologias de telecomunicações não dão sinais de que possam diminuir nos próximos anos. Muito pelo contrário: novas tecnologias, como a televisão digital de alta definição, o VoIP (transmissão de voz pela internet), a IPTV (transmissão de televisão pela internet) e o desenvolvimento dos sistemas de telefonia celular de terceira geração, acenam com mudanças que, mais uma vez, modificarão a maneira como as pessoas interagem com o mundo. Passou-se, então, a falar muito em convergência digital, isto é, a oferta integrada de serviços de telefonia celular, internet e televisão utilizando a mesma rede. E esta é uma questão que abrange tanto aspectos tecnológicos quanto jurídicos, e que deve ser amplamente debatida nos próximos anos.

Somando-se à questão da convergência de serviços oferecidos, a dificuldade em se levar esses serviços integrados ao consumidor, onde quer que ele se encontre, chega-se a um problema interessante: de um lado, a crescente demanda por serviços de comunicação móvel, e de outro, a dificuldade em integrar em um único equipamento serviços que utilizam diferentes bandas de freqüência, diferentes esquemas de modulação, diferentes codificações, entre outras tantas caracaterísticas peculiares de cada serviço. O problema poderia ser atacado agregando hardware ao equipamento, ao custo de um maior consumo de potência e de maiores dimensões físicas do produto final. Porém, a desejável portabilidade já estaria comprometida com a agregação de poucos serviços ao equipamento[1].

Neste ponto, mostra-se muito atrativa a possiblilidade de se reconfigurar o equipamento, para que este possa oferecer e ter acesso diversos serviços, sem a necessidade da incorporação de hardware ao dispositivo. Tanto melhor se este equipamento pudesse se reconfigurar remotamente, fazendo download de um software de um servidor, por exemplo. E esta é a característica central e também a principal vantagem oferecida pela tecnologia de Rádio Definido por Software: a reconfigurabilidade[2].

Esta característica de reconfigurabilidade torna o Rádio Definido por Software (RDS) interessante, não somente para a integração de serviços oferecidos comercialmente, mas também para fins militares. O RDS teve nas finalidades militares a principal motivação do seu nascimento, sendo ainda hoje uma das suas principais razões de ser[1, 2]. A construção de rádios que interagissem com diferentes interfaces aéreas, operando em diferentes modos e bandas de freqüência e que pudessem se reconfigurar, seria muito interessante para os sistemas de comunicação das Forças Armadas, para que estes possam interoperar entre si de uma forma mais barata e eficiente.

Outra vantagem oferecida pelo desenvolvimento do Rádio Definido por Software é a diminuição dos custos de produção acarretados pela produção de dispositivos de hardware mais padronizados, com suas funcionalidades podendo ser introduzidas pelo software carregado no dispositivo. Criaria-se, assim, uma economia de escala, que derrubaria os custos de produção do hardware destes sistemas, ao mesmo tempo

em que tiraria proveito do baixo custo de desenvolvimento de software[1].

1.2 DEFINIÇÃO DO PROBLEMA

A necessidade crescente da integração dos serviços de telecomunicações, sem perder em mobilidade e portabilidade fazem com que o Rádio Definido por Software esteja inserido neste processo de convergência dos serviços móveis de comunicação como uma das possíveis soluções que se apresentam.

Baseado, então, na situação explicitada no parágrafo anterior, pode-se dizer que o problema abordado neste trabalho, o desenvolvimento de um *Digital Downconverter* inserido dentro de um objetivo maior de desenvolver um protótipo embarcado de Rádio Definido por Software. Este protótipo, por definição, deve atender ao quesito de reconfigurabilidade, oferecendo assim uma solução para a integração de diversos serviços em um mesmo rádio, evitando a necessidade de agregar hardware ao equipamento.

O *Digital Downconverter* numa arquitetura de RDS é o responsável por entregar o sinal recebido pela antena e que já foi digitalizado pelos conversores AD para o dispositivo que realiza o processamento do sinal.

1.3 OBJETIVOS DO PROJETO

Neste trabalho, tem-se como objetivo principal o desenvolvimento de um *Digital Downconverter* para um protótipo embarcado de Rádio Definido por Software. Este *Digital Downconverter* apresentado neste trabalho deverá, além de desempenhar a sua função dentro da arquitetura do rádio, atender aos critérios de desempenho estabelecidos, sem restringir a capacidade de reconfiguração do protótipo. A validação do projeto implementado através de simulações também se interpõe como um objetivo necessário à obtenção do objetivo central do trabalho.

1.4 APRESENTAÇÃO DO MANUSCRITO

No capítulo 2 é feita uma revisão bibliográfica, na qual são apresentados conceitos fundamentais e arquiteturas de Rádio Definido por Software. Em seguida, no capítulo 3, a metodologia de desenvolvimento do projeto é apresentada. Resultados experimentais são discutidos no capítulo 4, seguido das conclusões no capítulo 5.

2 REVISÃO BIBLIOGRÁFICA

2.1 ARQUITETURA DE RÁDIO DEFINIDO POR SOFTWARE

2.1.1 Conceituação

O termo "Rádio Definido por Software" tem sido associado ultimamente a um grande número de tecnologias diferentes nos últimos anos e, até mesmo por ser uma tecnologia emergente, ainda não existe uma definição padronizada para o termo.

Segundo a definição de Joseph Mitola [2], "um Rádio Definido por Software (RDS) é, um rádio cuja modulação das formas de onda do canal são definidas em software. Isto é, as formas de onda são geradas como sinais digitais amostrados, convertidas de digitais para analógicas por um conversor DA de banda larga, que captura todos os canais do nó do RDS. O receptor, por sua vez, captura o sinal faz um abaixamento de freqüência, e demodula a forma de onda do canal por meio de um software que roda sobre um processador de uso geral".

Em [1], é feita uma definição mais ampla, referindo-se a RDS como todo transceptor de rádio cujos principais parâmetros são definidos em software e os aspectos fundamentais de operação do rádio possam ser reconfigurados com um simples *upgrade* deste software.

Logo, a filosofia por trás da tecnologia de RDS é, a grosso modo, transformar problemas de hardware em problemas de software, com o intuito fazer um transceptor reconfigurável. Obviamente, para atingir este objetivo, a digitalização do sinal deve ser feita o mais próximo possível da antena, de modo a realizar a maior parte do processamento em software. A figura 2.1 mostra o diagrama de blocos de um transceptor ideal definido por software. Na figura, nota-se que os conversores analógico-digital e digital-analógico estão localizados logo após a antena do rádio, sendo todo o tipo de processamento do sinal recebido e transmitido realizado através de um software rodando sobre um dispositivo reconfigurável.

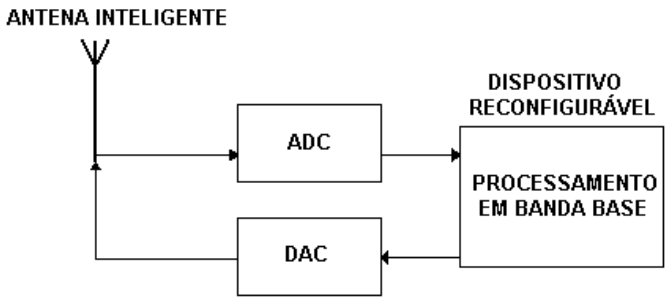

Figura 2.1: Transceptor RDS ideal

2.1.2 Arquitetura

A implementação de um modelo ideal de RDS como o apresentado na figura 2.1, como era de se esperar, esbarra em questões tecnológicas, principalmente no que diz respeito aos conversores e dispositivos de processamento. Sabe-se que, para dar suporte à reconfigurabilidade desejada para um RDS, de modo que ele possa operar em diversas faixas de freqüências, o conversor AD utilizado deve realizar a conversão a uma taxa de amostragem suficientemente alta para digitalizar banda de freqüência de alguns GHz e que seu consumo de potência esteja dentro de padrões aceitáveis. O dispositivo de processamento utilizado na arquitetura do RDS também deve ser capaz de processar a quantidade de informações necessárias para receber os dados dos conversores AD, bem como enviar dados para os conversores DA à taxa necessária. As principais limitações dos conversores AD e DA para atender a todos estes quesitos são: a taxa de amostragem, o *jitter* de abertura, o *range* dinâmico, e possíveis ruídos e distorções que são inseridos pelos conversores, como o ruído de quantização, o ruído térmico e distorções não-lineares, além de ter que atender níveis aceitáveis de consumo de potência. Limitações estas que se agravam à medida que se aumenta a taxa de amostragem do conversor, impondo maiores dificuldades ao se trabalhar em altas freqüências. E por causa destas limitações, os conversores AD e DA são, por muitas vezes, o elemento que define o *range* dinâmico, a largura de banda e o consumo de potência um RDS[3].

Assim, a necessidade de se construir um rádio que possa operar em banda-larga conflita com as limitações tecnológicas impostas pelos conversores. Torna-se necessário, para funcionamento adequado dos conversores, que seja introduzido um estágio que realize uma amplificação, controle automático de ganho, um abaixamento para uma freqüência intermediária e uma filtragem *anti-alias*. Este estágio que faz este pré-processamento é chamado de *front-end* de radiofreqüência, ou simplesmente *front-end* RF.

Após a sua digitalização, o sinal recebido deve ser entregue ao processador para que o processamento do sinal digital seja feito em software. Ocorre que, como o sinal foi digitalizado a uma freqüência intermediária, geralmente 455 kHz ou 10.7 MHz, a taxa de *throughput* necessária para a conversão analógico-digital atender ao critério de Nyquist [4] requereria uma velocidade de processamento muito alta. Uma abordagem para contornar este problema é realizar uma conversão digital para baixo (*digital downconversion*), transladando o sinal na freqüência para a banda-base, para então fazer uma re-amostragem, diminuindo ao máximo a taxa de *throughput* e, conseqüentemente, exigindo menos velocidade do processador. Na transmissão do sinal, o problema é semelhante: entregar o sinal digitalizado ao conversor digital-analógico já na freqüência intermediária exigiria muito poder de processamento. A estratégia então é fazer com que o sinal seja entregue em banda-básica e que, posteriormente, seja feita uma conversão digital para cima (*digital upconversion*), seguida de uma interpolação para aumentar a taxa de *throughput* antes da conversão DA.

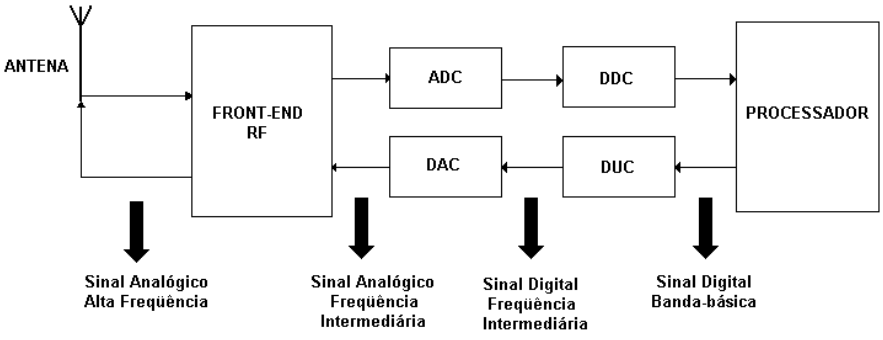

Figura 2.2: Transceptor RDS real

Diante das restrições impostas pelos conversores e pelos dispositivos de processamento, pode-se chegar a um modelo de um RDS real (figura 2.2), acrescentando os módulos *digital downconverter* (DDC)

e *digital upconverter* (DUC) e o *front-end* RF ao transceptor ideal da figura 2.1. O modelo mostrado na figura 2.2, ainda que bastante simplificado, contém todos os módulos necessários para se implementar um RDS. Obviamente, muitas complexidades e problemas práticos são encontrados quando se estuda minuciosamente cada bloco do diagrama da figura 2.2. Neste trabalho, especificamente, os esforços práticos foram concentrados na implementação do módulo DDC e comparações entre o desempenho do mesmo implementado em hardware ou em software (com a conversão para baixo sendo realizada pelo processador). Nas seções a seguir, serão descritos com maior detalhe o *front-end* RF, a conversão analógico-digital e digital-analógico, os módulos DDC e DUC, o processamento do sinal em banda-base e os dispositivos responsáveis por este processamento em um RDS.

2.1.3 O *front-end* RF

Em face das já conhecidas limitações tecnológicas encontradas nas conversões AD e DA dos sinais recebidos e transmitidos pelo terminal de rádio, a introdução de um estágio de radiofreqüência entre a antena e os conversores é necessária em qualquer implementação de RDS, como mostrado na figura 2.3. Este estágio deve fazer um processamento de radiofreqüência para, entre outras coisas, amplificar e fazer um abaixamento analógico de freqüência e uma filtragem *anti-alias* para que este possa ser digitalizado, realizar uma translação em freqüência do sinal que sai do conversor DA para que este possa ser transmitido, controlar a potência de transmissão e eliminar espúrios que possam ser transmitidos na antena do terminal, adequando o sinal transmitido ao padrão em que se deseja operar. Uma abordagem bastante abrangente sobre a adequação das especificações de um RDS aos padrões de interface aérea predominantes na Europa pode ser encontrada em [5], na parte referente à tecnologia de *front-end*.

No que refere-se à questão da reconfigurabilidade, a introdução do *front-end* RF, embora necessária ao funcionamento do terminal, limita a flexibilidade do mesmo. Encontram-se problemas com os filtros analógicos do *front-end*, que não mantêm uma seletividade satisfatória em uma ampla gama de freqüências. A própria antena do rádio já é um fator limitante à reconfiguração, quando se deseja uma alteração da freqüência de transmissão e recepção do sinal. A solução passa a ser, então, se possível, adicionar flexibilidade ao *front-end*, ao custo da adição de hardware para operar em faixas de freqüência distintas e também de um sistema de antenas inteligentes, que possam reconfigurar seu comprimento e sua diretividade. Mais sobre arquiteturas flexíveis de radiofreqüência e antenas inteligentes aplicados a RDS pode ser encontrado em [1].

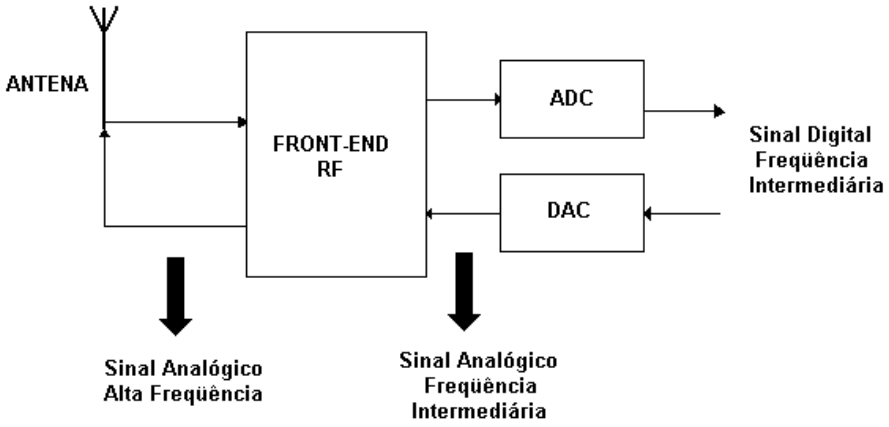

Figura 2.3: Localização do *front-end* RF

Partindo agora para uma descrição mais detalhada das funções do *front-end*, pode-se dividir as suas tarefas em duas partes: a que processa o sinal recebido pela antena e o entrega ao conversor AD e a que

processa o sinal vindo do conversor DA e o entrega à antena para que a transmissão possa ser realizada. Na parte que trata o sinal recebido, em linhas gerais, o *front-end* RF tem a função de amplificar o sinal, realizar a translação em freqüência para uma freqüência intermediária, realizar um controle automático de ganho e uma filtragem *anti-alias*, de modo que o conversor AD possa operar em uma freqüência intermediária mais baixa, não tendo o seu desempenho comprometido pela alta freqüência, tampouco pela variação da amplitude do sinal. Na parte que trata o sinal destinado à transmissão, a função do *front-end* é, basicamente, realizar a conversão do sinal para a freqüência de transmissão, filtragem para diminuir a transmissão de sinais espúrios e uma amplificação de potência, antes de entregar o sinal à antena. Em [3], nos aspectos de implementação, também se fala sobre o *front-end* RF, e tem-se uma discussão mais aprofundada em [6].

Muito embora as tarefas a serem executadas pelo *front-end* RF na transmissão e na recepção, explicitadas no parágrafo acima, devam ser realizadas em toda implementação de RDS, existe mais de uma forma de encarar este problema. E a principal diferença entre as arquiteturas do *front-end* diz respeito, principalmente, ao número de estágios utilizados para fazer as conversões de freqüência para cima ou para baixo.

Quando a conversão da freqüência de recepção para a banda base se dá em apenas um estágio, diz-se que o receptor ou transmissor possui uma arquitetura de conversão direta. Um modelo de RDS cujo *front-end* tem esta arquitetura para a recepção pode ser vislumbrado na figura 2.4. Da figura nota-se que o sinal proveniente da antena é mixado com duas senóides em quadratura geradas por um oscilador local de freqüência variável, deslocando o sinal em freqüência para a banda-base e gerando dois sinais em quadratura, e só então os sinais em quadratura são filtrados e amplificados para poder ser realizada a conversão AD. Um modelo para a arquitetura de conversão direta para a transmissão pode ser visto na figura 2.5.

Já quando a conversão para a banda-base acontece em múltiplos estágios, diz-se que o receptor possui uma arquitetura de conversões múltiplas, tal como é mostrado na figura 2.6. Observa-se, da figura 2.6, que o sinal vindo da antena é filtrado e amplificado, mixado com uma senóide gerada por um oscilador local de freqüência variável, novamente filtrado e amplificado, passa pelo conversor AD, e só então o segundo abaixamento de freqüência e a geração dos sinais em quadratura são realizados, só que desta vez digitalmente, através do DDC. Um modelo de um transmissor RDS de múltiplas conversões é mostrado na figura 2.7.

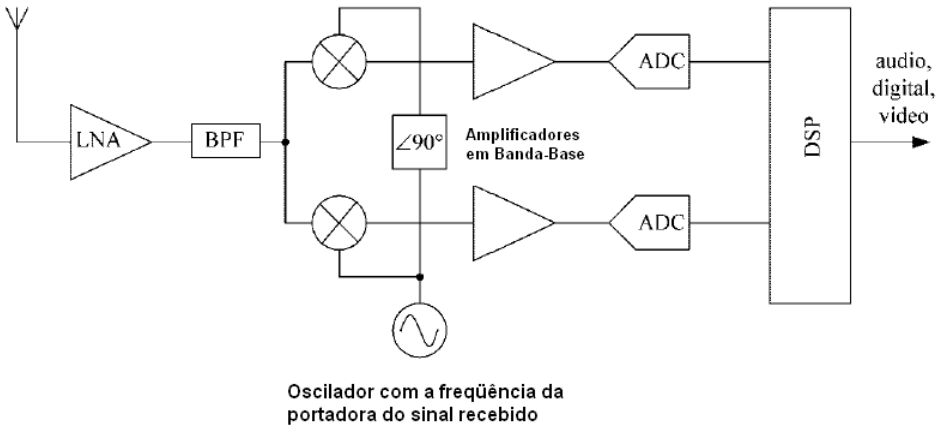

Figura 2.4: Conversão direta na recepção

Ambas as arquiteturas mostradas nas figuras 2.4 e 2.6 têm suas vantagens e desvantagens. A favor da arquitetura de conversão direta existe o fato de ser uma arquitetura de baixa complexidade, podendo inclusive ser integrada em um único *chip*, os filtros analógicos a serem implementados são mais simples e

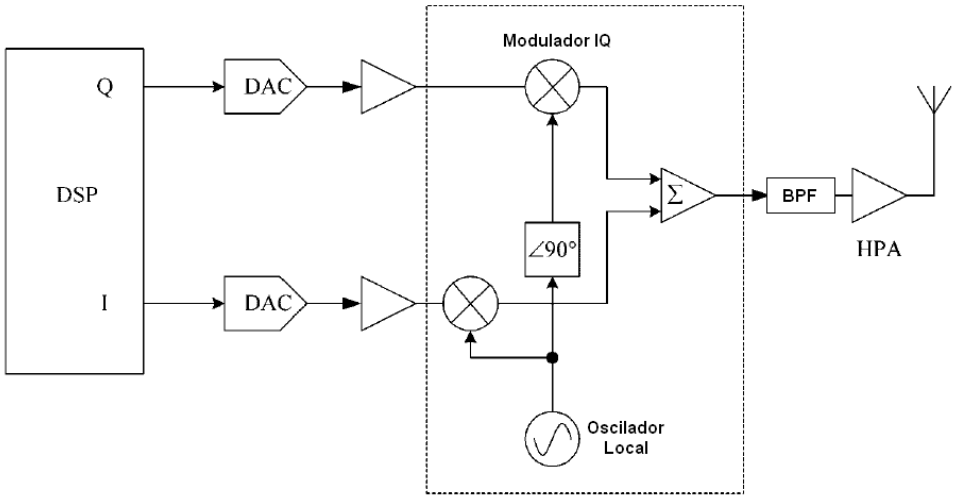

Figura 2.5: Conversão direta na transmissão

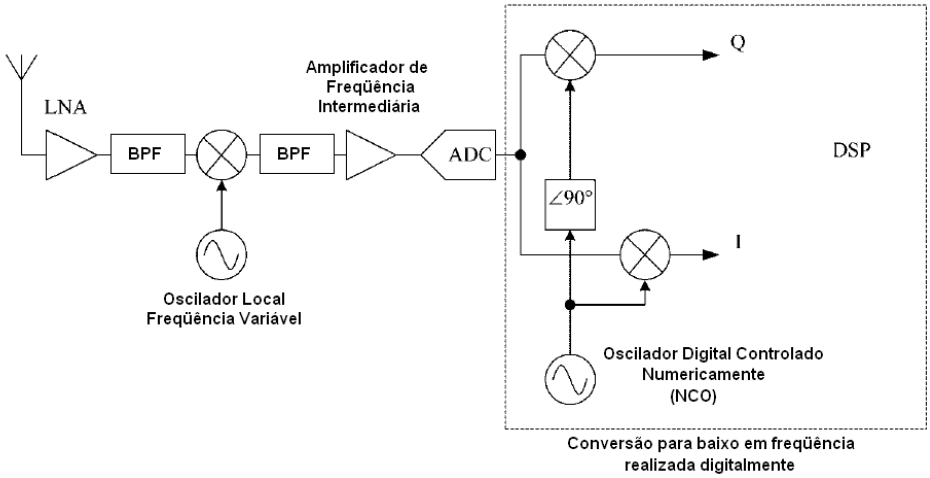

Figura 2.6: Conversão múltipla na recepção

também a supressão de freqüências imagem é mais simples do que na arquitetura de conversões múltiplas. Como desvantagens dos receptores de conversão direta, tem-se a necessidade de um oscilador que gere duas senóides precisamente em quadratura e balanceadas em amplitude em uma ampla faixa de freqüências; também os *mixers* devem estar balanceados e precisam ser capazes de operar em uma ampla faixa de freqüências. Outro problema reside no espalhamento espectral dos osciladores e o retorno destas freqüências espúrias para a antena, onde em parte serão irradiadas e em parte refletidas de volta para o receptor, causando no sinal um nível DC variante no tempo[5].

Contra o receptor de múltiplas conversões está a maior complexidade, a necessidade de vários osciladores locais e o fato de ser necessário o uso de muitos filtros de freqüência intermediária, tornando difícil a integração em um único *chip*. Porém, as vantagens da boa seletividade oferecida pelos diversos filtros de pré-seleção e canalização, o fato de a conversão de real para complexo (sinais em quadratura) ser realizada em uma freqüência intermediária fixa, juntamente com os problemas apresentados pela arquitetura de conversão direta fazem com que as arquiteturas de múltiplas conversões seja a melhor escolha para o *front-end* atualmente.

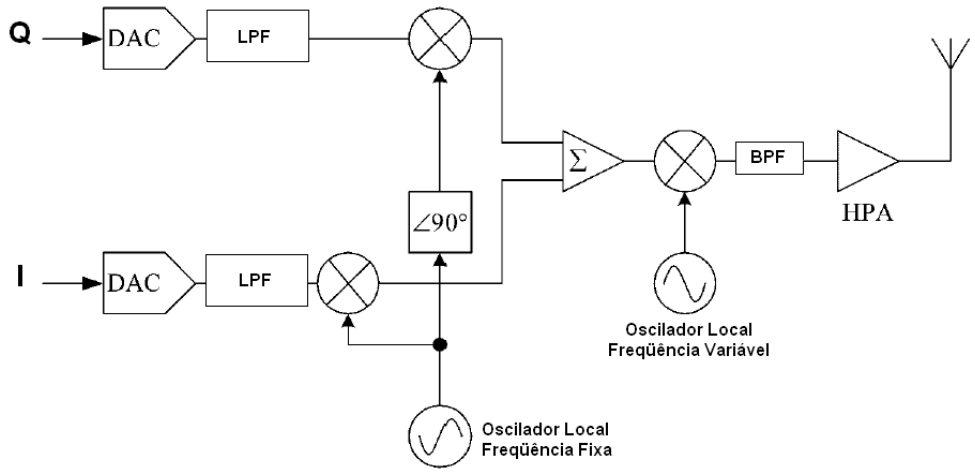

Figura 2.7: Conversão múltipla na transmissão

2.1.4 ADC e DAC

Os conversores analógico-digital e digital-analógico, são, sem dúvida, o maior ponto crítico na arquitetura de um RDS. Eles são o principal fator limitante da arquitetura, tanto em relação a custos quanto em relação ao desempenho. E quanto melhor é o seu desempenho mais próximos da antena do rádio eles podem ser situados. Por outro lado, conversores não tão bons irão requerer mais processamento sendo feito no *front-end* RF e, conseqüentemente, implicarão em uma menor flexibilidade do rádio. Um RDS ideal como o mostrado na figura 2.1 deve possuir conversores capazes de fazer a conversão do sinal diretamente na antena do rádio. Isto exigiria conversores com taxa de amostragem muito alta, largura de banda da ordem de GHz, um elevado *range* dinâmico efetivo, tudo isto evitando um consumo de potência muito elevado[3]. A tecnologia atual de conversores AD e DA não atende a todos estes requisitos.

Entretanto, o fato de os conversores serem atualmente o principal fator limitante em um RDS não significa que eles sejam um problema para a arquitetura do rádio. Pelo contrário, a evolução da tecnologia de conversores foi um dos fatores que possibilitaram o desenvolvimento do processamento digital de sinais e que possibilitam que, hoje em dia, seja possível captar sinais de radiofreqüência e processá-lo em software, bem como gerar sinais em software e transmiti-los por uma interface aérea.

2.1.4.1 Técnicas de amostragem

As especificações necessárias para os conversores utilizados em um RDS dependem substancialmente da arquitetura adotada para o rádio[5]. Na conversão AD, existem diversas técnicas de amostragem que são utilizadas em virtude da arquitetura implementada para o receptor.

Na conversão direta, a taxa de amostragem deve atender ao critério de Nyquist, sendo duas vezes maior do que a componente de maior freqüência do sinal amostrado. Cabe ressaltar a necessidade de um filtro *anti-alias* antes do conversor AD, para evitar que as freqüências superiores sejam refletidas na faixa de freqüência de interesse devido ao *alias*[4], também conhecido como dobramento espectral.

Caso os conversores AD possuam uma capacidade de amostragem maior do que a exigida pelo critério de Nyquist, pode ser usada a técnica da superamostragem, que realiza um número de amostras maior do que o necessário para diminuir a seletividade necessária do filtro *anti-alias*.

Na técnica da subamostragem (ou amostragem em freqüência intermediária), necessita-se de uma taxa de amostragem de, no mínimo, duas vezes a largura de banda do sinal amostrado. O sinal deve passar

por um filtro passa-banda bastante seletivo, uma vez que esta técnica utiliza o *alias* a seu favor, replicando várias vezes no domínio da freqüência a cada freqüência múltipla da freqüência de amostragem. A técnica da subamostragem também promove uma conversão para baixo na freqüência do sinal amostrado.

Outra técnica utilizada é a conversão em quadratura, onde o sinal é dividido em duas componentes complexas (componente em fase e em quadratura, ou I e Q, respectivamente), cada uma delas ocupando metade da banda do sinal original. Esta técnica de amostragem permite diminuir em duas vezes a taxa de amostragem do conversor, uma vez que a largura de banda do sinal foi diminuída. Em compensação, devem ser utilizados dois conversores ao invés de um único. Tal técnica de amostragem pode ser observada no receptor da figura 2.4.

2.1.4.2 Arquitetura dos conversores

Existem diferentes arquiteturas de conversores disponíveis no mercado, cada uma delas com suas vantagens e desvantagens. Conhecer tais arquiteturas é essencial para o projeto de um RDS, para saber em cada caso específico qual o tipo de conversor é o mais adequado. Entre os principais tipos de conversores pode-se citar o conversor do tipo paralelo (ou do tipo *flash*), o conversor multi-estágio e o conversor sigma-delta.

Os conversores paralelos, cuja arquitetura básica é mostrada na figura 2.8, comparam a tensão de entrada com determinados níveis de tensão distribuídos entre a referência positiva e negativa de tensão. Como se deseja a saída em código binário, é necessário que se utilize uma lógica combinacional para fazer esta codificação. A principal vantagem oferecida pelo conversor paralelo é a capacidade de realizar a conversão a uma velocidade muito grande. A maior desvantagem recai sobre o fato de o número de comparadores crescer exponencialmente com a precisão desejada na saída do conversor, sendo 10 bits o limite prático da precisão deste tipo de conversor[5]. Portanto, este tipo de conversor é ideal para aplicações que não requerem uma alta resolução na conversão, oferecendo, nestes casos, o melhor desempenho possível a um baixo custo.

Nos conversores sigma-delta, cujo modelo é mostado na figura 2.9, um comparador diz se o sinal na entrada é maior ou menor que na saída, utlizando um conversor DA na malha de realimentação para poder realizar esta comparação. Se a entrada for maior que a saída, a saída é incrementada. Caso a entrada seja menor que a saída, a saída é decrementada. Os conversores sigma-delta operam utilizando a técnica da superamostragem, e possuem a vantagem de eliminar o ruído de quantização para sinais de banda estreita[3].

Já os conversores de multi-estágio o sinal digitalizado em um estágio é convertido novamente em analógico, e, após isso, é feita uma subtração entre o sinal analógico original e o convertido. O resultado desta subtração é enviado para o próximo estágio, e assim sucessivamente. Este tipo de conversor apresenta as vantagens de se poder aumentar a precisão do conversor sem aumento exponencial no número de componentes do circuito e sem acrescentar longos delays. A desvantagem está na necessidade de uma precisão para o conversor DA do primeiro estágio maior do que a precisão do conversor AD inteiro.

O atual estado da arte dos conversores AD para sistemas *wireless* são os conversores com precisão de 14 bits, operando a mais de 100 MSamples/s, mantendo uma relação sinal-ruído típica de 75 dB. Contudo existe uma demanda crescente por conversores AD com precisão de 16 bits e taxa de amostragem de 120 MSamples/s. Em relação aos conversores DA, o estado da arte são os conversores com resolução de 14 bits com relação sinal-ruído maior que 80 dBc[5].

Em [5] e [7] tem-se uma discussão completa sobre estas arquiteturas de conversores AD e também sobre as arquiteturas de conversores DA.

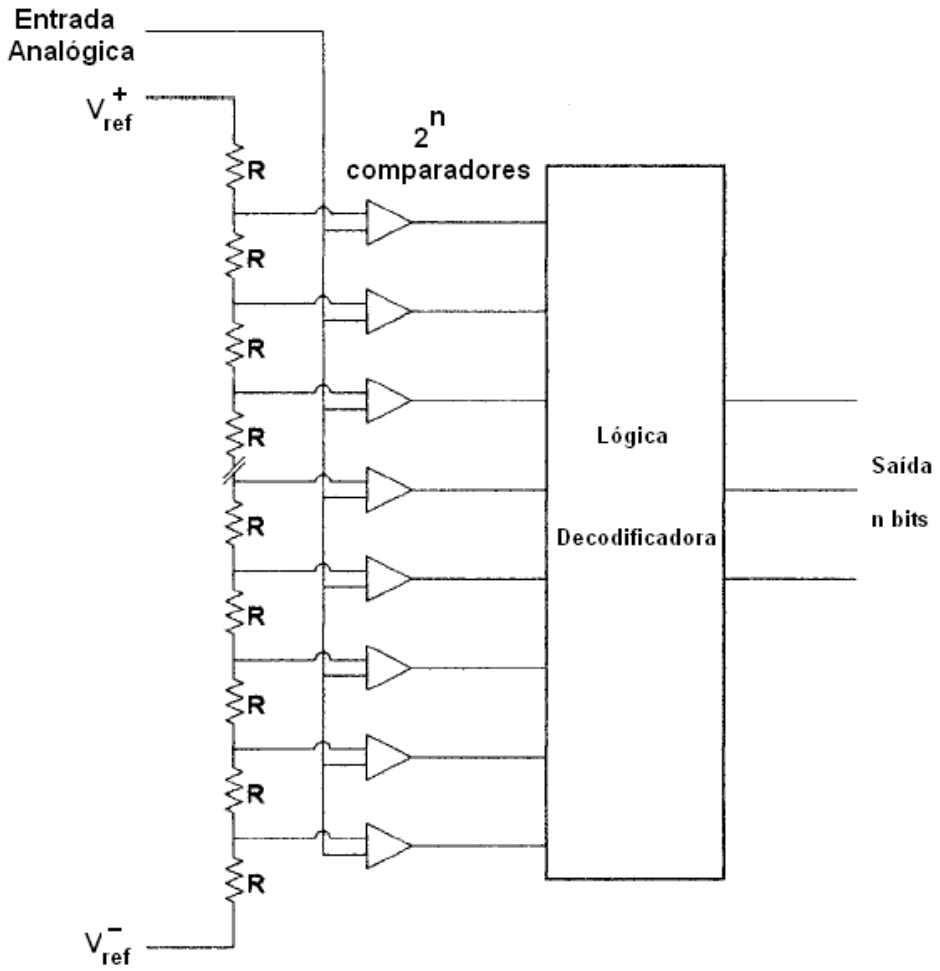

Figura 2.8: Arquitetura de um ADC *flash*

2.1.4.3 Ruídos e distorções nos conversores

As distorções que o sinal pode sofrer ao passar pelos conversores são devidas ao ruído térmico, distorções de não-linearidade diferencial, *jitter* de abertura, distorções devido a saturação, distorções harmônicas, entre outras, além do inerente ruído de quantização.

Ruído de quantização refere-se à distorção inevitável quando se representa um sinal contínuo por valores discretos. Este efeito pode ser modelado como sendo uma fonte de ruído e seu efeito sobre o sinal é tanto maior quanto menor é a resolução do conversor e a taxa de superamostragem do sinal[5].

A distorção de saturação acontece quando a tensão na entrada do ADC excede a tensão máxima que pode ser convertida pelo ADC. Já o ruído térmico está associado às variações na movimentação dos elétrons pelo condutor devidas à maior ou menor agitação térmica dos átomos do mesmo. Está presente em todo componente eletrônico, sendo que sua variância é dada pela equação 2.1, onde k é a constante de Boltzmann, T é a temperatura na escala absoluta e R é a resistência em Ohms[5]:

$$\sigma_n^2 = \sqrt{4kTR} \tag{2.1}$$

O *jitter* de abertura é a incerteza em relação ao momento em que a amostragem é feita. Isto é, dada uma freqüência de amostragem f_s, o intervalo de tempo entre a realização de uma amostra e outra é estimado com sendo $T_s = \frac{1}{f_s}$, mas existe uma incerteza neste tempo de amostragem. Dependendo da taxa

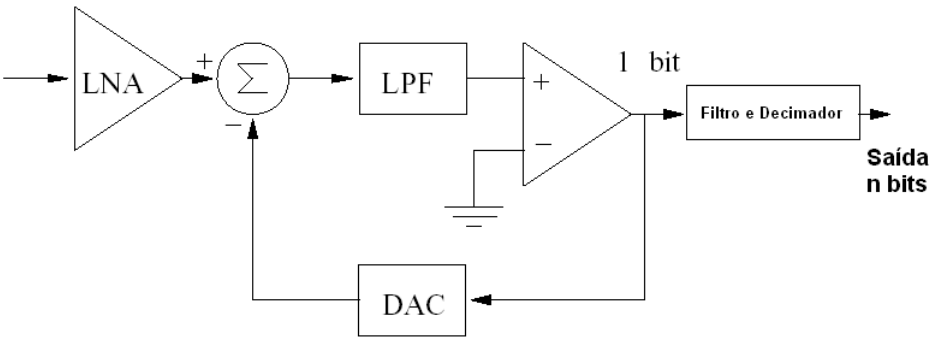

Figura 2.9: Arquitetura de um ADC sigma-delta

de variação do sinal naquele instante, um pequeno erro no tempo de amostragem pode acarretar um erro considerável na saída, tal como é mostrado na figura 2.10.

As distorções causadas pelas não-linearidades diferenciais são devidas à distribuição não-eqüitativa dos níveis de quantização dentro do intervalo de tensão que pode ser convertido. Isto ocasiona um erro na quantização, conforme mostrado na figura 2.11.

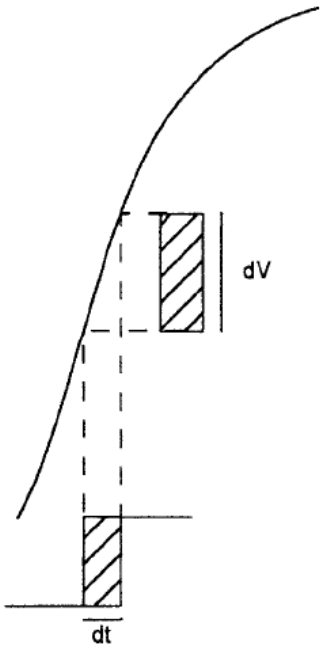

Figura 2.10: *Jitter* de abertura

Todas estas fontes de erros e distorções exercem influência sobre a resolução do conversor. O número efetivo de bits de resolução do conversor (ENOB) pode ser calculado pela equação 2.2. Se admitir-se que as outras fontes de ruído sejam desprezíveis frente à distorção do *jitter* de abertura. Neste caso, a relação sinal-ruído é dada pela equação 2.3, onde f_s é a freqüência de amostragem do conversor e $\overline{\Delta a^2}$ é a variância do *jitter* de abertura. Por 2.2 e 2.3 tem-se uma relação logarítmica entre o número efetivo de bits de resolução do conversor e a freqüência de amostragem[3]. A figura 2.12 mostra esta relação para um *jitter* de abertura de 0.5 ps.

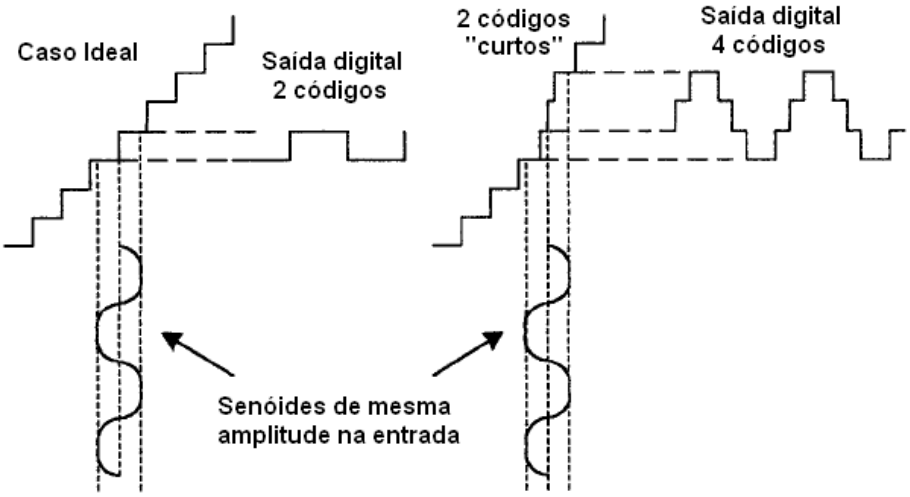

Figura 2.11: Não-linearidade diferencial

$$ENOB = \frac{SNR_{medida} - 1.76dB}{6.02} \tag{2.2}$$

$$SNR = -\log_{10}(2\pi^2 f_s^2 \overline{\Delta a^2}) \tag{2.3}$$

2.1.5 O *Digital Downconverter* (DDC)

O módulo *Digital Downconverter*, ou simplesmente DDC, é o responsável por fazer um deslocamento em freqüência do sinal digitalizado em uma freqüência intermediária para a banda-base. Ele está presente na arquitetura de um receptor de múltiplas conversões, tal como o mostrado na figura 2.6, logo após o conversor AD.

Tal conversão para baixo do sinal digitalizado é mostrada na figura 2.13. O sinal proveniente do ADC é mixado com duas senóides oscilando na freqüência intermediária e em quadratura para decompor o sinal em suas componentes complexas. Após isso passa por um filtro digital decimador, para selecionar somente a faixa de freqüência de interesse e, ao mesmo tempo, reduzir a taxa de dados encaminhados ao processador[8, 9].

As duas senóides em quadratura são geradas por meio de um NCO (*Numerically Controlled Oscilator*), comandado diretamente pelo processador do rádio. Um NCO é um bloco que, basicamente, tem como entrada um *clock* e um incremento de fase; a cada ciclo do *clock* ele incrementa o seu acumulador de fase e apresenta na saída o seno e o cosseno da fase armazenada no acumulador. Há também a possibilidade de se ter entradas para sinais digitais para modulações em freqüência ou fase das portadoras digitais geradas pelo NCO, abrindo a possibilidade de se usar o NCO como um modulador digital freqüência ou de fase. Um NCO padrão é mostrado na figura 2.14.

O funcionamento do NCO pode ser implementado em diversas arquiteturas, dependendo diretamente da disponibilidade de memória do dispositivo em que ele é implementado e da freqüência de saída necessária. Um NCO pode ser implementado através de uma *look-up table* em uma ROM, utilizando multiplicadores em hardware ou então utilizando o algoritmo CORDIC.

Dentre as arquiteturas que utilizam uma *look-up table*, tem-se a baseada em uma grande ROM e a baseada em uma pequena ROM. A que utiliza uma grande ROM armazena os valores de seno e cosseno para

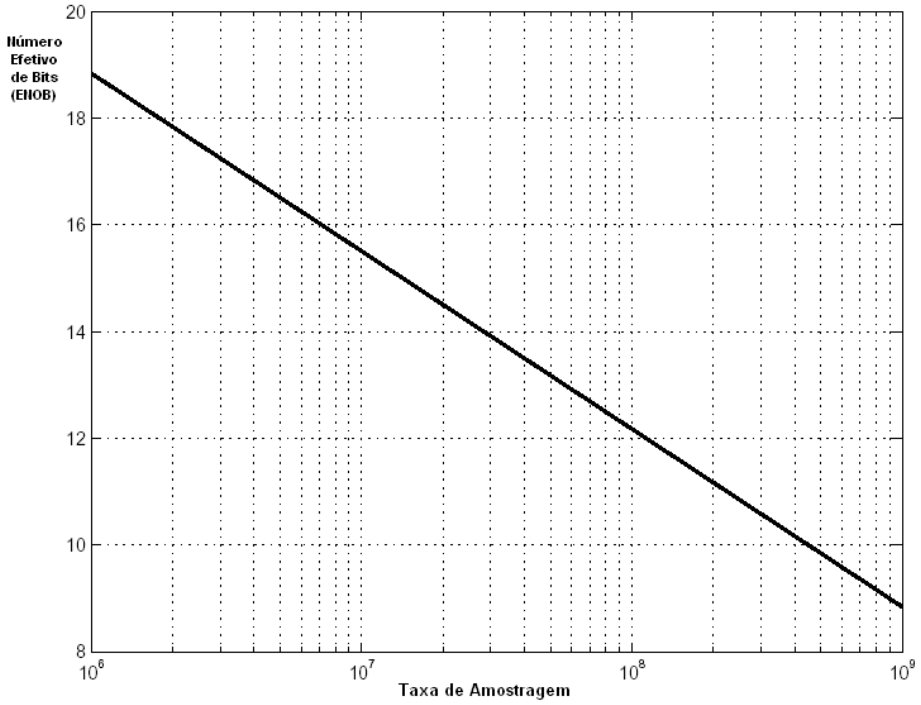

Figura 2.12: Influência de um *jitter* de abertura de 0.5 ps no número de bits efetivos de resolução em um ADC

todos os argumentos possíveis do acumulador de fase (isto é, para os 360 graus do ciclo trigonométrico), sendo a arquitetura ideal para aplicações que necessitam de um NCO muito rápido e a memória não é o fator limitante do sistema. A arquitetura baseada em uma pequena ROM armazena os valores de seno e cosseno para apenas 45 graus do ciclo trigonométrico, sendo que, para o resto do ciclo, os valores são calculados a partir de relações trigonométricas. A implementação baseada em uma pequena ROM é ideal para sistemas que ainda requerem altas freqüências de saída, mas não dispõem de muita memória. Já a arquitetura baseada em multiplicadores utiliza multiplicadores em hardware para calcular seno e cosseno através de uma série de Taylor truncada.

A freqüência do *clock*, o número de bits do acumulador e o número de bits para codificação da saída são fatores determinantes para a precisão do NCO. A freqüência em Hertz de saída das senóides geradas f_0, para um acumulador de fase de M bits, um dado incremento de fase ϕ_{inc} (número inteiro variando de 1 a 2^M), uma freqüência de *clock* f_{clk} em Hertz, é dada pela equação 2.4.

$$f_o = \frac{\phi_{inc} f_{clk}}{2^M} \tag{2.4}$$

Fazendo ϕ_{inc} igual a 1, tem-se a menor freqüência que pode ser gerada pelo NCO, que também representa a mínima diferença entre as freqüências de duas senóides geradas pelo NCO. Este valor é chamado de resolução de freqüência, e pode ser calculado pela equação 2.5.

$$f_{res} = \frac{f_{clk}}{2^M} \tag{2.5}$$

O CORDIC é um algoritmo numérico para calcular funções trigonométricas baseado em rotações fasoriais iterativas, até que se chegue tão próximo do argumento de entrada quanto necessário. É um algoritmo de excelente performance, excelente para sistemas que não dispõem de multiplicadores em hardware ou de memória interna para implementação de uma *look-up table* em ROM[5, 3].

As senóides em quadratura são, então, mixadas com o sinal digitalizado vindo do ADC por meio de multiplicadores digitais. O sinal mixado então passa por um filtro digital de resposta finita a um impulso (FIR)[10]. Tal filtro deve ser do tipo passa-baixa para eliminar o espectro do sinal centrado na freqüência de duas vezes a freqüência intermediária, resultante da mixagem do sinal. Com o sinal desejado em banda-base, é feita então uma decimação, respeitando o critério de Nyquist para evitar o dobramento espectral. Feito isso, tem-se as componentes complexas I e Q do sinal em banda-base, que a seguir são encaminhadas ao dispositivo de processamento.

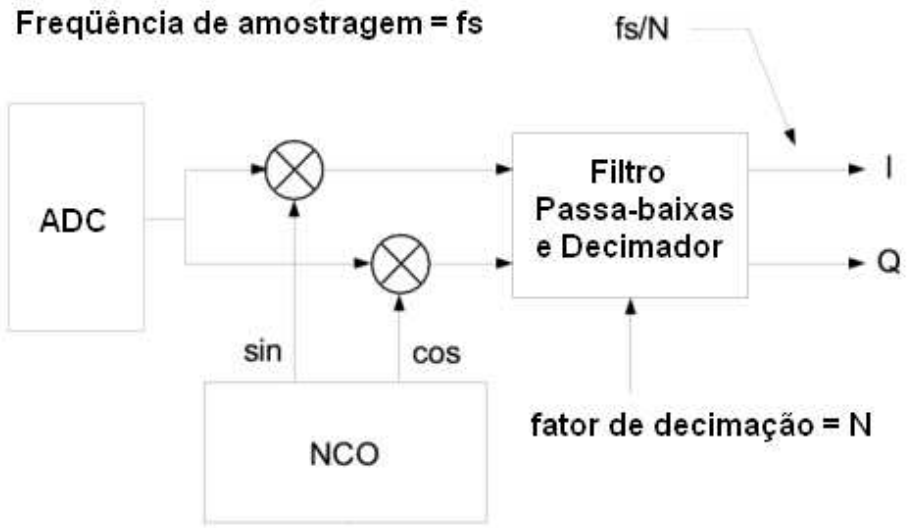

Figura 2.13: Diagrama de blocos do DDC

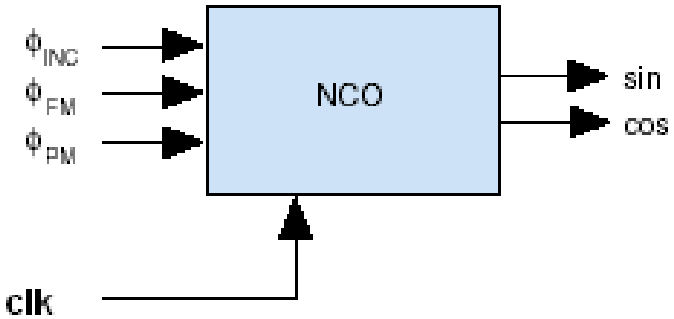

Figura 2.14: NCO padrão

2.1.6 O *Digital Upconverter* (DUC)

Em um transmissor com arquitetura de múltiplas conversões, tal como é mostrado na figura 2.7, o sinal digital a ser transmitido vem do dispositivo de processamento em banda-base e na forma complexa I e Q. Este sinal deve ser convertido novamente para analógico e passar por duas conversões para cima em

freqüência antes de ser transmitido. A primeira conversão para cima leva o sinal que sai do conversor DA da banda-base para a freqüência intermediária, e a segunda para a freqüência de transmissão.

Porém, a conversão para a freqüência intermediária pode ser realizada antes da conversão digital-analógico, aproveitando-se de uma maior facilidade para gerar senóides em quadratura digitalmente, diminuindo o hardware do *front-end* RF e levando a fronteira entre o processamento analógico e o digital mais próxima à antena. O *Digital Upconverter* (DUC) é o módulo responsável por realizar esta conversão para a freqüência intermediária digitalmente.

O DUC recebe as amostras I e Q do sinal transmitido do dispositivo de processamento, faz uma interpolação entre os pontos para obter um maior número de amostras, realiza a mixagem com duas portadoras em quadratura geradas por um NCO e subtrai Q de I para converter o sinal do formato complexo para o real[10]. Um DUC padrão é mostrado na figura 2.15. Após isso, é feita a reconstrução do sinal analógico pelo DAC para que o mesmo seja entregue ao *front-end* RF para ser transmitido. Mais sobre DUCs pode ser encontrado em [11] e [12].

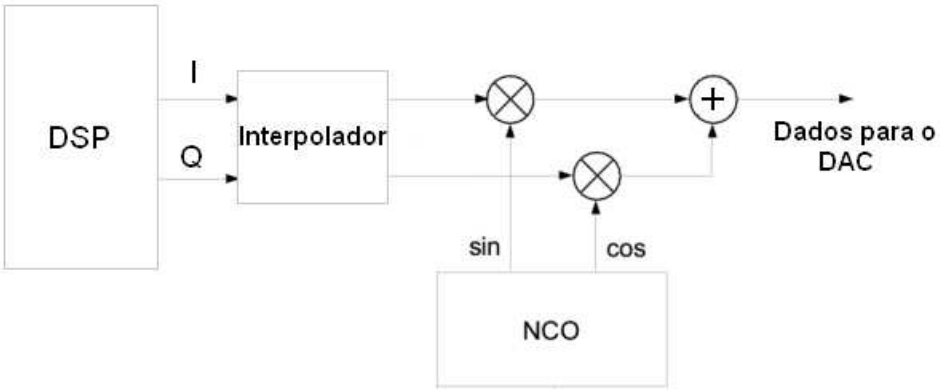

Figura 2.15: Diagrama de blocos do DUC

2.1.7 Processamento Digital de Sinais (SCA)

A grande variedade de arquiteturas de hardware e software na construção de um RDS gera o grande problema da especificidade das aplicações desenvolvidas para uma determinada plataforma. Isto é, devido às grandes diferenças entre a plataforma hardware-software em diversos terminais de RDS, as aplicações desenvolvidas para uma plataforma específica não podem ser utilizadas em uma outra plataforma.

Pensando em uma maior integração entre os sitemas militares de rádio nos EUA, o Sistema de Rádio Tático Comum, ou JTRS (*Joint Tactical Radio System*), estabeleceu uma arquitetura aberta de software que possibilita o gerenciamento e interconexão de recursos de software em um ambiente computacional distribuído[3], que foi denominada SCA (*Software Communications Architecture*). A arquitetura SCA estabelece padrões e especificações para a construção de um RDS, que fazem especificações desde a interface aérea até o desenvolvimento de aplicativos de software para que os equipamentos e programas possam ser intercambiados no sistema.

A arquitetura SCA adotou o *middleware* CORBA (*Common Object Request Broker Architecture*) para garantir a independência entre hardware e software, estabelecendo uma clara divisão entre as camadas de hardware, middleware e a camada de aplicação.

Embora existam projetos de RDS que não fazem uso da arquitetura SCA, a mesma vem sendo adotada

como padrão pelo Fórum de Rádio Definido por Software. Como a interoperabilidade entre os diversos sistemas de rádio é uma das principais razões de ser do Rádio Definido por Software, aproveitando-se dos benefícios da economia de escala gerada por isso, não há razão para crer que a arquitetura SCA não seja adotada como padrão de produção na indústria.

2.2 ELEMENTOS DE PROCESSAMENTO EM RDS

A escolha do hardware de processamento é um dos pontos críticos a serem definidos na arquitetura de um RDS. Tal escolha, além da disponibilidade de recursos, deve ser tomada em função da finalidade a que o rádio se destina. Isto é, fatores como a programabilidade, tempo de reconfiguração, capacidade de processamento, consumo de potência e custo devem ser ponderadas para se encontrar o melhor dispositivo para uma finalidade específica[3].

Entre as opções disponíveis para dispositivos de processamento em RDS tem-se processadores de uso geral, processadores digitais de sinais (DSP), FPGAs e circuitos integrados de aplicação específica (ASIC), sendo que cada opção mostra vantagens e desvantagens. Logo, deve-se encontrar uma solução de compromisso que leve à definição de qual dispositivo de processamento atende melhor às especificações do projeto. Não devem ser descartadas também soluções que se utilizem de um sistema híbrido, com processamento distribuído entre dois ou mais dispositivos distintos, aproveitando o melhor dos dois mundos.

Os processadores de uso geral são, sem dúvida, a alternativa mais barata e com maior capacidade de processamento e programabilidade. No entanto, seu desempenho em aplicações de processamento digital de sinais deixa a desejar em relação às outras alternativas disponíveis, devido, principalmente, ao fato de não possuir instruções otimizadas para estas tarefas. Quando se tenta contornar esta não-otimização com processadores que operam com *clock* mais elevado, o consumo de energia passa a ser o fator limitante e torna inviável a aplicação dos mesmos em dispositivos portáteis. Mesmo assim, existem projetos didáticos e acadêmicos de RDS que se utilizam dos processadores de uso geral e da plataforma PC, abrindo mão da portabilidade do sistema.

E quando se fala em uma arquitetura portátil para um RDS, as alternativas mais viáveis e, por conseguinte, de maior interesse, são os DSPs, FPGAs e ASICs. É feita uma descrição de FPGAs, DSPs e ASICs nas subseções 2.2.1, 2.2.2 e 2.2.3, respectivamente.

2.2.1 FPGA

O FPGA (*Field Programmable Gate Array*) consiste de um *chip* capaz de implementar funções digitais (tanto combinacionais como seqüenciais) por meio de blocos lógicos configuráveis. Estes blocos, por sua vez, assumem as funcionalidades de funções lógicas mais simples e estão organizados em forma de matriz, com seus pinos de entrada e saída conectados a chaves programáveis. Desta forma, os blocos lógicos podem se interconectar entre si e com os pinos de saída, o que permite a construção de circuitos digitais complexos[13, 14].

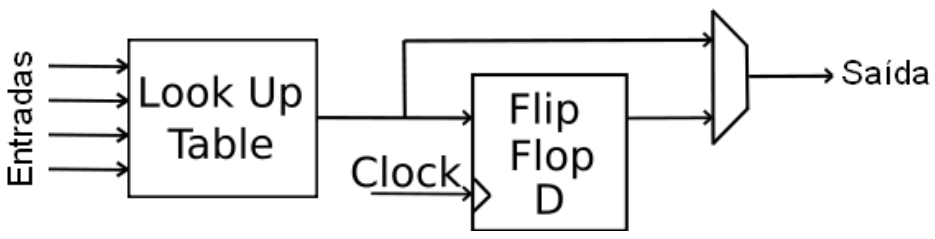

Figura 2.16: Bloco Lógico Configurável de um FPGA

Cada bloco lógico, para poder implementar funções lógicas combinacionais simples, deve receber entradas e ser capaz de retornar saídas reprogramáveis, formando uma pequena *look-up table*. Para a construção de funções seqüenciais, é necessário também um elemento de memória para armazenar estados, como um flip-flop tipo D, e uma entrada para um *clock* de sincronização. A figura 2.16 mostra a arquitetura básica de um bloco lógico de FPGA.

Com mais blocos como o da figura 2.16 interconectados é possível construir circuitos lógicos mais elaborados, como máquinas de estados, filtros digitais e até mesmo processadores. A disposição destes elementos em formato de matriz com chaves configuráveis conectando os blocos entre si permite, além de construir circuitos complexos utilizando vários blocos, que a matriz de conexões seja reconfigurada, formando um circuito diferente. A figura 2.17 mostra como estas chaves configuráveis estão dispostas em uma matriz de blocos lógicos configuráveis como é um FPGA.

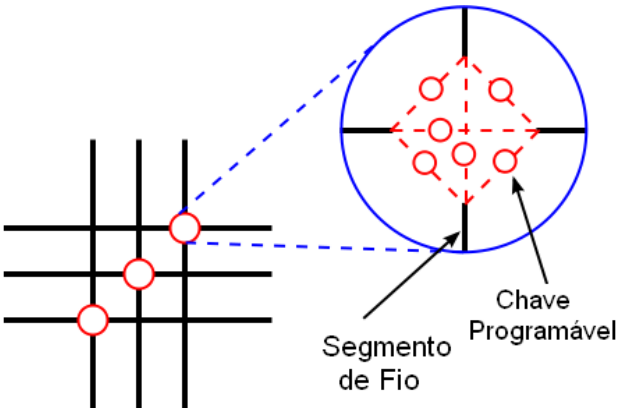

Figura 2.17: Chaves configuráveis em uma matriz de blocos lógicos configuráveis

Esta arquitetura permite que o FPGA seja capaz de executar uma reconfiguração em hardware, tornando-se um dispositivo muito atrativo para aplicações que requerem reconfigurabilidade, como RDS. Isto porque, muitas vezes é interessante que uma parte do processamento do sinal seja realizado em hardware (pela simples questão de um desempenho melhor), mas que por isso não perca a flexibilidade.

A programação de um FPGA se dá através de linguagens de descrição de hardware, como por exemplo as linguagens VHDL e Verilog. Tais linguagens permitem que o projetista descreva o circuito desejado de maneira física ou comportamental, sem a necessidade de se preocupar com a programação de cada bloco. Esta tarefa então é executada pelo sintetizador, que interpreta o código escrito pelo projetista do circuito e gera um *netlist*, um arquivo binário que será carregado no FPGA, fazendo a programação dos blocos lógicos configuráveis e das chaves programáveis.

Muitos FPGAs contêm elementos de hardware auxiliares, como multiplicadores embarcados ou blocos de memória. Os multiplicadores possuem a finalidade de otimizar a execução operações matemáticas, proporcionando ganho em tempo de execução e também economia de blocos lógicos. Em [13] encontra-se uma descrição mais detalhada sobre a tecnologia de FPGAs.

2.2.2 DSP

Um DSP (*Digital Signal Processor*) é um microprocessador projetado especificamente para executar funções de processamento digital de sinais em tempo real. Isto é, possui instruções específicas e arquitetura otimizada para realizar operações matriciais como convoluções, produto interno entre vetores ou FFT [10].

Sua arquitetura mais comum apresenta separação entre memória de dados e memória de programas (arquitetura Harvard[14]), conjunto especial de instruções para manipulações com múltiplos dados com uma única instrução (*SIMD - Single Instruction Multiple Data*), não apresentando suporte a multitarefa e

muitas vezes contando com conversores AD e DA para o processamento de sinais analógicos.

A programação de um DSP pode ser realizada tanto em C ou C++, como em linguagens de nível mais baixo como *Assembly* (valendo ressaltar, neste ponto, que cada família de DSPs possui, geralmente, seu próprio *Assembly* e suas próprias ferramentas de desenvolvimento oferecidas pelo seu fabricante). Para mudar a tarefa a ser executada pelo dispositivo basta então carregar um novo programa na memória do dispositivo, sendo este processo mais rápido que a reconfiguração de um FPGA. Porém, ao contrário do FPGA, a reconfiguração não se dá em hardware, mas em software. Conseqüentemente, o hardware de um DSP não pode ser otimizado a critério do projetista para realizar outras funções.

Contudo, o fato de os DSPs não serem otimizados para nichos ou aplicações bastante específicas é compensado pelo baixo custo destes componentes. Isto porque os DSPs são amplamente utilizados em diversos ramos da indústria de eletrônicos, não somente para aplicações em Telecomunicações. Os benefícios desta economia de escala fazem com que os DSPs sejam, por vezes, a opção mais viável para construção de um RDS[1].

2.2.3 ASIC

O principal empecilho quanto à utilização de circuitos integrados de uso específico (ASIC) em um RDS é a sua falta de flexibilidade. Ou, mais propriamente, o custo de adicionar flexibilidade ao rádio cujo processamento é baseado em ASICs. Isto porque, necessariamente, deveria haver diferentes *chips* para poder alternar entre diferentes modulações, codificações ou funcionalidades que se quisesse dar ao terminal de rádio. Deste modo, poderia ser conseguida uma reconfigurabilidade funcional do rádio (porém não na verdadeira acepção de reconfigurabilidade), ao custo da agregação de hardware e do encarecimento do terminal[3].

Mesmo assim, em um RDS, a utilização de ASICs pode ser uma alternativa viável se ele for utilizado em combinação com um dispositivo de processamento reprogramável, como um DSP, por exemplo. Devido ao seu elevado desempenho, os ASICs podem exercer funções críticas no processamento de um rádio, como por exemplo a codificação e decodificação de áudio e vídeo. Em casos como estes, a utilização de ASICs trabalhando em conjunto com outro dispositivo de processamento é mais viável economicamente do que arcar com as despesas de projeto e produção de um DSP especial para a aplicação a que se destinar[1].

2.2.4 Comparação entre FPGA, DSP e ASIC

Fazendo uma comparação entre estes três dispositivos de processamento que se apresentam mais atrativos para aplicações em RDS, verifica-se um evidente compromisso entre reconfigurabilidade e desempenho. Os DSPs oferecem a possibilidade de reprogramação rápida, atingindo a flexibilidade desejada do dispositivo de processamento. Maior flexibilidade alcançada ao custo de um pior desempenho, se comparado aos FPGAs e ASICs.

Os ASICs, por sua vez, são o extremo oposto dos DSPs na comparação entre os três tipos de dispositivos. Eles apresentam o melhor desempenho, porém oferecem pouca facilidade de reconfiguração. Um RDS com o processamento baseado em ASICs precisaria de um *chip* exclusivo para cada modo de operação caso fosse feita a opção por utilizar circuitos integrados produzidos em larga escala, o que agregaria hardware ao rádio[1]. Existe também a possibilidade de se projetar um ASIC específico para o terminal de rádio em questão, acrescentando aos custos do terminal de rádio as despesas de pesquisa e desenvolvimento do *chip*. Como conseqüência, as soluções baseadas em ASICs são geralmente mais caras.

Os FPGAs apresentam-se como um meio termo entre os DSPs e os ASICs. Seu tempo de reconfiguração é consideravelmente maior que o dos DSPs, mas oferece a grande vantagem do seu hardware ser reconfigurável. Seu desempenho é, em geral, inferior aos ASICs e superior aos DSPs. A figura 2.18 resume a comparação entre os dispositivos de processamento em questão. Em [15] esta discussão sobre os

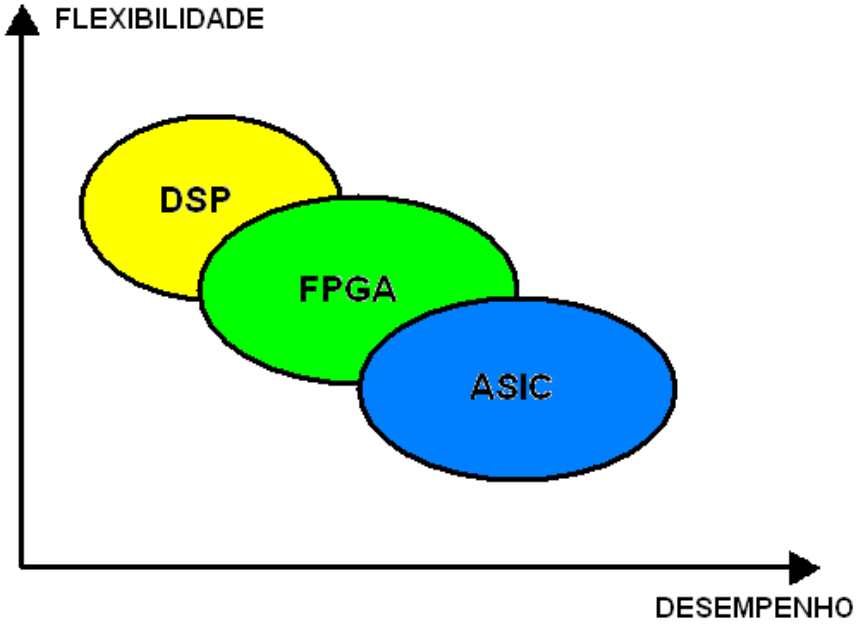

Figura 2.18: Comparação entre DSP, FPGA e ASIC

dispositivos de processamento para RDS é feita de maneira mais aprofundada.

2.2.5 Co-projeto de Hardware e Software

Geralmente, os sistemas de comunicação devem estar inseridos em sistemas maiores. Em outras palavras, é interessante ter sistemas de comunicação embarcados. O termo *sistema embarcado* refere-se a a sistemas de informação que fazem parte de um produto maior[16, 17]. E, em se tratando de sistemas embarcados, em geral não se tem dispositivos com grande capacidade de processamento e memória disponíveis. Por esta razão é tão importante o co-projeto de hardware e software, que visa buscar o ponto ótimo na distribuição de tarefas executadas em hardware e em software.

Uma dada tarefa executada em hardware é executada com uma melhor performance se comparada à mesma tarefa executada em software. O custo deste desempenho é a agregação de hardware ao sistema, a perda de flexibilidade e uma elevação do custo. E neste ponto, o co-projeto de hardware e software é essencial para resolver este *trade-off* entre desempenho e custo, buscando a melhor distribuição de funções entre o hardware e o software, sempre tendo em vista atender as especificações de desempenho do projeto e tirando máximo proveito da flexibilidade dada pelas implementações em software.

No projeto de um RDS também é interessante que seja feito um co-projeto de hardware e software, detectando as funções mais críticas do processamento e implementando-as em hardware. A implementação em hardware das partes mais críticas do processamento do sinal tem um desempenho bastante superior por se utilizar de paralelismo na execução. Sendo assim, pode-se jogar blocos de processamento para hardware e software buscando um ponto onde se estabelece um equilíbrio com o melhor desempenho possível em uma dada arquitetura.

3 DESENVOLVIMENTO DO *DIGITAL DOWNCONVERTER* (DDC)

3.1 INTRODUÇÃO

Conforme explanado no capítulo 2, em seção específica, a função exercida por um DDC em uma arquitetura de rádio digital é realizar uma conversão para baixo em freqüência, de forma digital. Muitas são as formas possíveis para se projetar e implementar um DDC, sendo que suas características de projeto devem adequá-lo às necessidades de processamento do sinal que deve ser recebido pelo rádio.

Partindo de uma arquitetura de múltiplas conversões para o receptor ao qual o DDC em questão deve se aplicar, pelas vantagens que esta arquitetura apresenta (ver capítulo 2), definiu-se que o DDC deve levar os sinais digitalizados em uma freqüência intermediária para a banda-base, mantendo a freqüência de oscilação do NCO constante[1].

Foram levadas em consideração as limitações técnicas impostas pelos conversores AD, que determinam, entre outros parâmetros de projeto do *front-end* RF, qual será a freqüência intermediária para qual o sinal será transladado na primeira conversão de freqüência, e também o número efetivo de bits em que o conversor AD pode converter as amostras do sinal analógico. Embora estes não sejam parâmetros específicos do projeto do DDC, eles devem estar definidos para que seja possível estabelecer a comunicação entre o conversor AD e o DDC.

Neste projeto de DDC, especificamente, visou-se o recebimento de sinais de voz, sendo que os coeficientes dos filtros digitais, bem como os decimadores, foram ajustados de modo a sempre atender às condições mínimas para estabelecimento deste tipo de serviço. Inclusive, os testes realizados, cujos resultados são expostos no capítulo 4, foram baseados em sinais de voz. Contudo, como será discutido neste mesmo capítulo, existe a possibilidade de se adicionar flexibilidade a este DDC, dotando-o de reconfigurabilidade em tempo de execução. Isto acrescentaria uma maior gama de aplicações ao terminal de RDS, possibilitando o recebimento e processamento de outros tipos de sinais além do sinal de voz, como vídeo e dados, por exemplo.

O presente capítulo apresenta o processo de desenvolvimento do DDC, desde os ambientes de desenvolvimento utilizados no projeto, passando pela arquitetura adotada, definição de parâmetros do projeto e detalhamento dos blocos que o compõe.

3.2 AMBIENTE DE DESENVOLVIMENTO EM HARDWARE E EM SOFTWARE

3.2.1 Ambiente de Desenvolvimento em Hardware

Para a implementação em hardware do DDC foi utilizada a placa de desenvolvimento Altera DE2, mostrada na figura 3.1. A placa DE2 possui diversas interfaces para comunicação do FPGA com outros dispositivos, o que faz dela um ambiente didático para o desenvolvimento de projetos de eletrônica digital baseados em FPGA e muito útil para prototipagem. As especificações da Altera DE2 são listadas a seguir:

- FPGA Altera Cyclone II 2C35;
- Altera Serial Configuration device - EPCS16;

[1] Exceto no caso em que a mudança da freqüência intermediária se faz necessária.

- USB Blaster (on board);
- 512-Kbyte SRAM;
- 8-Mbyte SDRAM;
- 4-Mbyte de memória Flash;
- Soquete de cartão SD;
- 4 botões;
- 18 chaves lógicas;
- 18 LEDs vermelhos;
- 9 LEDs verdes;
- Oscilador de 50 MHz;
- Oscilador de 27 MHz;
- 24-bit CD-quality audio CODEC, com entrada e saída de áudio e para microfone;
- VGA DAC (10-bit high-speed triple DACs) com conector VGA para saída;
- TV Decoder (NTSC/PAL) com conector de entrada para sinal de TV;
- Controlador Ethernet 10/100;
- Controlador USB host/device com conectores dos tipos A e B;
- RS-232 transceiver e conector de 9 pinos;
- Conector PS/2 para mouse e teclado;
- IrDA transceiver;
- Dois conectores de expansão de 40 pinos com proteção de diodo.

Na figura 3.2 é apresentado um diagrama de blocos mostrando como o FPGA da DE2 se comunica com os diversos componentes da placa.

Pelas especificações da Altera DE2, e o acesso direto pelo FPGA aos diversos componentes e interfaces da placa, conforme mostrado no diagrama da figura 3.2, vê-se que a placa oferece a possibilidade de desenvolvimento de aplicações para a área de Telecomunicações, inclusive aquelas que envolvem processamento de áudio e de vídeo.

A placa Altera DE2 é, pois, um ambiente de hardware capaz de dar suporte à implementação de um protótipo de RDS. Mesmo não sendo dotada de conversores AD e DA que possam realizar as conversões necessárias, estes conversores podem ser agregados a uma placa em que esteja implementado o *front-end* RF, comunicando-se com o FPGA através dos conectores de expansão. No FPGA seriam implementados o DDC, o DUC, as funções de processamento do sinal em banda-base e a entrada e saída de dados do sistema.

Para realizar o processamento do sinal, sem a necessidade de se reconfigurar o hardware a cada reconfiguração requerida, existe a possibilidade de se utilizar um processador embarcado no FPGA, tornando possível a reconfiguração do rádio em nível de software. Para FPGAs da Altera, existe o processador NIOS II, que apresenta uma arquitetura RISC de 32 bits. A Altera não exige pagamento de licensa de propriedade intelectual para a utilização do NIOS II, sendo que a única restrição é que ele seja utilizado somente

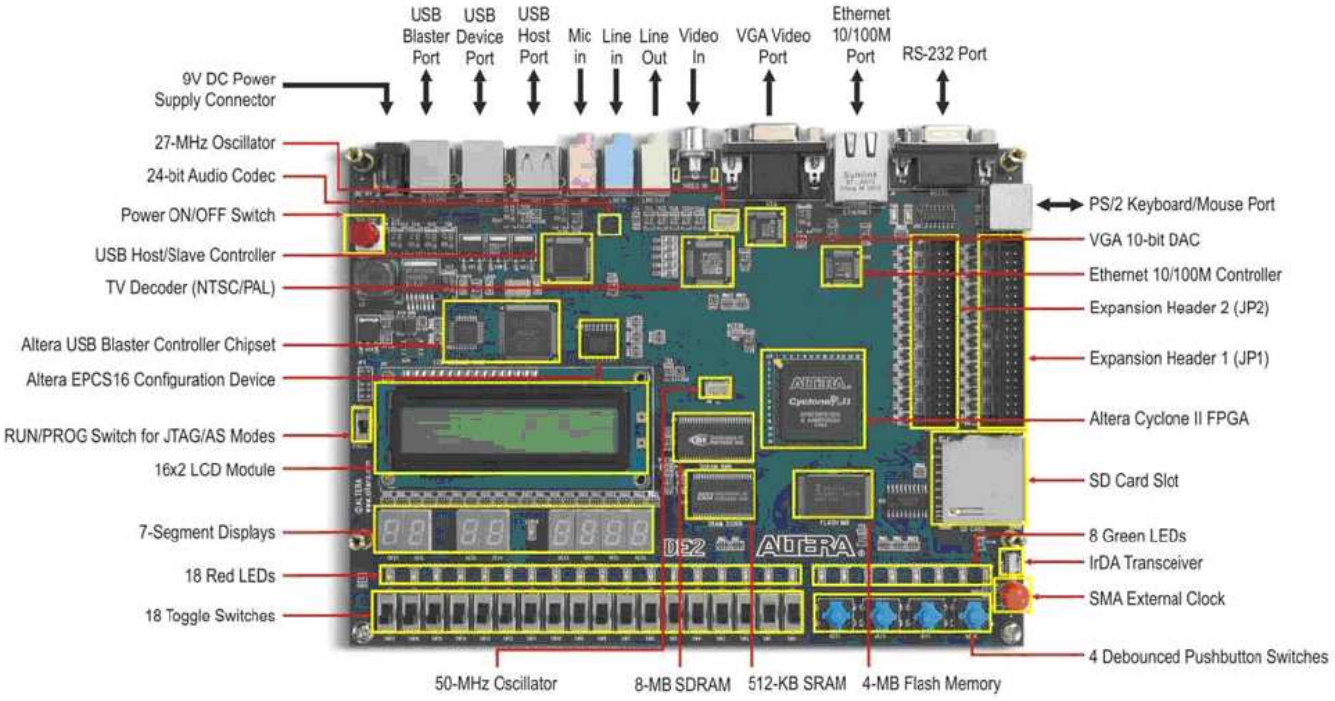

Figura 3.1: Placa de desenvolvimento Altera DE2

em FPGAs Altera. Como o NIOS II se trata de um *soft-core processor*, isto é, um processador de núcleo flexível, ele apresenta as vantagens de personalização das suas instruções e da sua comunicação com os periféricos.

3.2.2 Ambiente de Desenvolvimento em Software

Para o desenvolvimento do DDC foi utilizado o software Quartus II 7.0 da Altera, com uma licença gratuita para estudantes. O Quartus II é um ambiente completo para projetos de aplicações para chips programáveis (SOPC - *system-on-a-programmable-chip*). No ambiente do Quartus II, o projeto é elaborado em uma linguagem de descrição de hardware, como por exemplo VHDL ou Verilog, para, em seguida, ser feita a síntese destes códigos. Após a síntese, é gerado o *netlist* para programação do FPGA. No Quartus II também é possível a realização de simulações funcionais ou temporais (considerando o atraso das portas) da aplicação sintetizada. Tais simulações são muito úteis para validação do projeto realizado e convém que sejam feitas antes da programação do FPGA, uma vez que nelas podem ser verificados possíveis erros de projeto. Na figura 3.3 é mostrado o fluxograma de desenvolvimento no Quartus II.

Para simulação do funcionamento do DDC, principalmente na análise da resposta em freqüência do sinal em diversos pontos do DDC, também foi utilizado o software MATLAB 6.0, desenvolvido pela *The MathWorks*. Para isto, foi utilizada a licença do MATLAB do Departamento de Engenharia Elétrica da Universidade de Brasília, visto que se trata de um software proprietário.

3.3 PARÂMETROS DO PROJETO

Em linhas gerais, a arquitetura do DDC implementado segue o modelo simplificado mostrado na figura 2.13. Na figura 3.4 vê-se o diagrama de blocos do DDC comunicando-se com o processador NIOS II. Neste esquema, o DDC é enxergado como periférico pelo processador, que deve controlar seus parâmetros

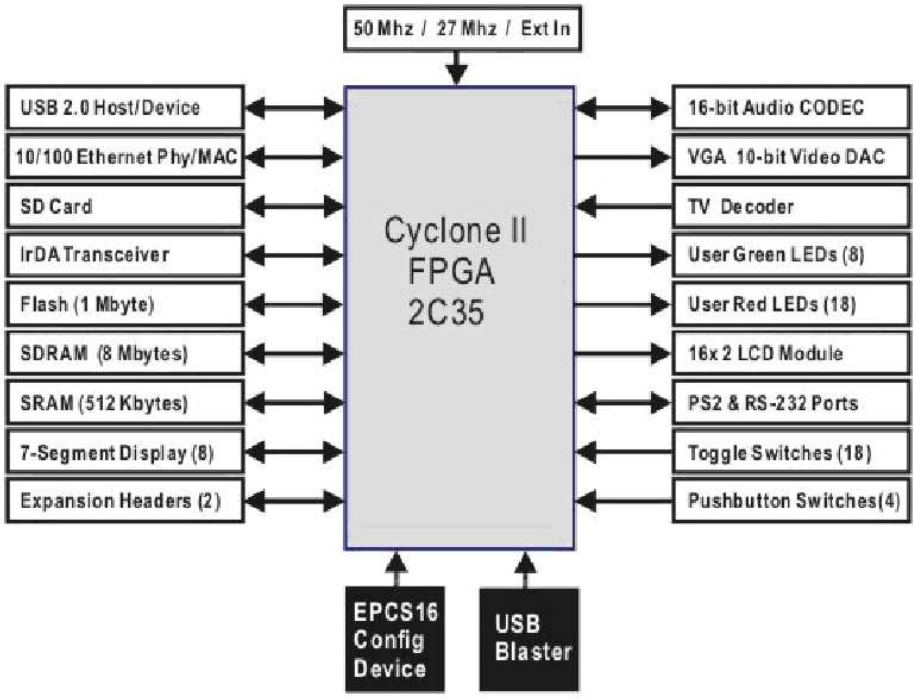

Figura 3.2: Diagrama de blocos da Altera DE2

de entrada, principalmente o incremento de fase do NCO, e receber as componentes I e Q do sinal já em banda-base para ser processado.

Porém, antes de se iniciar o projeto a partir do rascunho, para que ao final ele se comporte tal como o modelo da figura 2.13, é necessário estabelecer parâmetros que servirão de base para a arquitetura do dispositivo. Os parâmetros gerais que foram estabelecidos para o projeto do DDC apresentado neste trabalho estão resumidos na tabela 3.1. Ao longo desta seção são dadas as explicações para a adoção destes parâmetros para o projeto em questão.

Um dos parâmetros que deve ser definido é o número de bits do barramento de entrada do DDC. Esta decisão recai imediatamente sobre a escolha do conversor AD utilizado no protótipo de RDS. Uma resolução de 14 bits a 50 MSamples/s foi adotada para o conversor AD, uma vez que esta resolução atende com sobras às finalidades do DDC e já é encontrada em conversores AD fabricados em larga escala.

A freqüência intermediária assumida para o receptor de múltiplas conversões, no qual o DDC implementado deve se inserir, foi a de 455 kHz. Este valor é um dos valores padronizados para as freqüências intermediárias em receptores AM super-heteródinos[4, 18]. Já para receptores FM, a freqüência intermediária adotada é a de 10.7 MHz. O estabelecimento destes valores padronizados dizem respeito aos osciladores a cristal disponíveis para a construção de tais receptores, necessários para a conversão em freqüência analógica. No caso de um RDS, a escolha da freqüência intermediária está muito ligada às limitações que serão encontradas na construção do *front-end* RF. Para o DDC, basta que se conheça o valor desta freqüência intermediária, e duas senóides desta freqüência em quadratura serão geradas digitalmente. No caso de uma mudança da freqüência intermediária, basta que se mude o parâmetro de entrada correspondente ao incremento de fase em cada ciclo do NCO.

Para o NCO, foi definido que este iria gerar as senóides digitais com uma resolução de 14 bits, operando com um *clock* de 50 MHz, um dos *clocks* disponíveis na placa Altera DE2. O número de bits utilizado no acumulador de fase é 32 bits, o que significa que é esperado um número binário de 32 bits para representar o incremento de fase das senóides a cada ciclo do *clock*. Mais detalhes sobre a definição dos parâmetros do NCO serão apresentados na próxima seção.

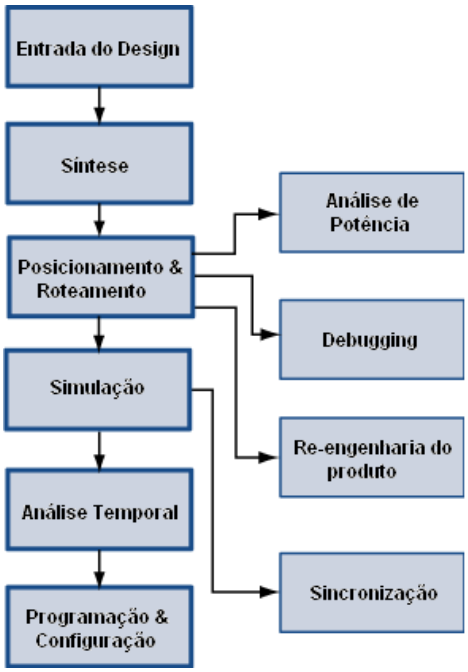

Figura 3.3: Fluxograma do desenvolvimento de um projeto no Quartus II

Os *mixers* digitais utilizados no DDC são multiplicadores digitais implementados em hardware. Um multiplica os 14 bits do sinal digitalizado entregue pelo conversor AD pelo cosseno digital gerado pelo NCO, enquanto o outro multiplica o mesmo sinal digitalizado pelo seno digital gerado pelo NCO. Como a multiplicação se dá em tempo real nos *mixers*, para que não ocorra um aumento da taxa de *throughput* após os *mixers* que não representa um aumento na taxa de digitalização do sinal, é necessário que haja sincronismo entre o sinal na saída do conversor AD e as senóides geradas pelo NCO. Caso não haja este sincronismo entre estes sinais, além de gerar um aumento da taxa de *throughput* que não reflete um aumento real da quantidade de informações processadas, poderá haver diferença nos tempos de duração dos dados que serão repassados ao filtro passa-baixas. Tal situação é mostrada na figura 3.5. É utilizado um barramento de 28 bits (número máximo de bits necessário para representar o resultado da multiplicação de dois números de 14 bits) na saída do *mixer*.

Os filtros passa-baixa digital devem, então, receber uma entrada de 28 bits e realizar a filtragem da maneira devida. Foi definido que a saída de cada filtro passa-baixa deve ser representada por números de 32 bits. Isto porque, como o filtro passa-baixa já pode executar uma decimação, o uso de outro decimador cascateado com o filtro passa-baixa torna-se opcional. Neste caso, o filtro passa-baixa entregaria os 32 bits das componentes complexas I e Q do sinal em banda-base diretamente ao processador. No caso do uso dos decimadores, estes recebem 32 bits na entrada, e reduzem a taxa de *throughput* por um fator de decimação N, escolhendo um a cada N dados que lhe são enviados. Mais detalhes da implementação do filtro passa-baixa digital e do decimador serão dados na próxima seção.

Definidos estes parâmetros de arquitetura do DDC, acrescidos de sinais de validação para entrada e saída do DDC, *reset* e *enable*, pode-se utilizá-lo como uma "caixa-preta" tal como a mostrado na figura 3.6.

Tabela 3.1: Parâmetros gerais de projeto do DDC

ADC	50 MSamples/s Resolução: 14bits
Barramento de entrada do DDC	14 bits
NCO	Algoritmo: CORDIC Resolução da saída: 14 bits Acumulador de fase: 32 bits 50 MSamples/s
Mixers	Barramentos de entrada: 14 bits Barramentos de saída: 28 bits
Filtros digitiais FIR passa-baixa	Barramentos de entrada: 28 bits Barramentos de saída: 32 bits Freqüência de corte: 100kHz Número de coeficientes: 128 Fator de decimação: 128
Decimadores	Barramentos de entrada: 32 bits Barramentos de saída: 32 bits Fator de decimação: 9
Barramento de saída do DDC	32 bits

3.4 DETALHES DOS COMPONENTES

3.4.1 *MegaCores* Altera

Os componentes descritos nesta seção fazem parte dos *MegaCores*, blocos que implementam funcões utilizadas em processamento digital de sinais de propriedade intelectual da Altera. Entre estas funções, encontram-se, por exemplo, filtros digitais, decimadores e interpoladores, NCOs, blocos para cálculo de FFTs, codificadores e detectores de erros.

Os *MegaCores* Altera são parametrizáveis, isto é, o projetista do sistema SOPC pode personalizá-los com as características que forem de seu interesse antes que de gerar os códigos das funções desejadas em Verilog ou VHDL. Após gerado o código destas funções em uma destas linguagens de descrição de hardware, elas podem ser integradas a um projeto maior, para que seja feita a síntese do *netlist* que será carregado no FPGA.

A Altera disponibiliza estas funções para que os projetistas de sistemas SOPC possam testá-las em simulações e no FPGA. No entanto, o *netlist* gerado utilizando algum *MegaCore* é limitado em tempo. Após testado, caso o *MegaCore* seja aprovado e seja de interesse a sua utilização em um produto comercial, deve-se entrar em contato com a Altera para a aquisição de uma licença para a utilização da sua propriedade intelectual.

3.4.2 NCO

A definição dos parâmetros do NCO é muito importante em aspectos qualitativos da geração das senóides em quadratura para a decomposição do sinal digitalizado proveniente do conversor AD. Isto ocorre porque estes parâmetros influenciam diretamente na pureza espectral das senóides geradas. Parâmetros como o algoritmo utilizado para a geração destas funções trigonométricas (ver seção referente ao NCO no capítulo 2), número de bits do acumulador de fase e o número de bits utilizado para representação da saída exercem efeitos sobre o nível de ruído de quantização na saída do NCO.

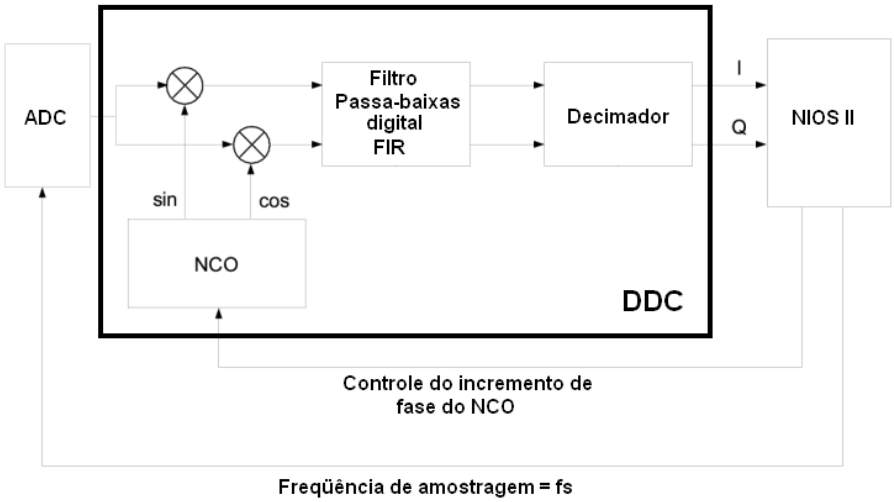

Figura 3.4: Diagrama de blocos do DDC comunicando-se com o NIOS II

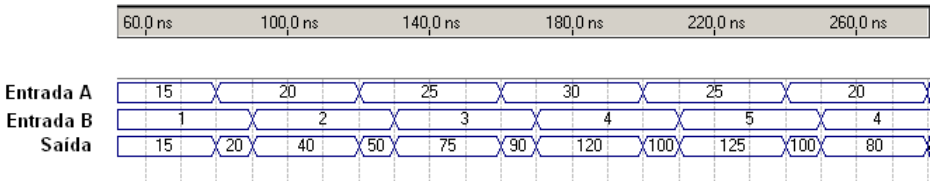

Figura 3.5: Formas de onda de entradas não sincronizadas no mixer

A escolha do algoritmo para a geração das senóides requer uma solução de compromisso entre a utilização de recursos do FPGA e a pureza espectral das senóides geradas. Pode-se também, ao escolher um algoritmo que economize recursos da pastilha, compensar a perda de desempenho variando os outros parâmetros do NCO, tais como a precisão do acumulador de fase ou a precisão da representação da amplitude de saída das senóides. A tabela 3.2 mostra os valores dos parâmetros do NCO utilizado na implementação do DDC. Infere-se da tabela 3.2 que o algoritmo CORDIC não utiliza bits de memória nem multiplicadores embarcados, economizando elementos que podem ser mais úteis se utilizados para outras funções, até mesmo pelo processador do rádio. O custo desta economia de memória e multiplicadores é um aumento do número de elementos lógicos na implementação. Neste projeto, optou-se pela utilização do algoritmo CORDIC, devido à intenção de, posteriormente, implementar mais dispositivos dentro da mesma pastilha de FPGA, como por exemplo um processador flexível.

A tabela 3.3 indica a escolha dos parâmetros do NCO utilizado na implementação do DDC. Com os 32 bits utilizados para a precisão do acumulador de fase, calcula-se a resolução de freqüência do NCO através da fórmula 2.5, obtendo como resultado uma resolução de freqüência de 0.0116 Hz. A máxima freqüência da senóide que pode ser gerada no NCO é definida, pelo critério de Nyquist, como metade da freqüência do *clock*. No caso de utilizarmos um *clock* de 50 MHz, a máxima freqüência obtida é de 25 MHz.

Para os parâmetros definidos na tabela 3.3, com o incremento de fase ajustado para gerar senóides de 455 kHz, a resposta em freqüência do NCO é mostrada na figura 3.7 e a resposta no tempo é mostrada na figura 3.8. O incremento de fase para que se pudesse obter uma senóide a freqüência desejada foi obtido através da equação 2.4. Para obter uma senóide de freqüência de 455 kHz, o incremento de fase deve ser de 39084202, representando-o como um número inteiro. Para uma freqüência de 10.7 MHz, outro dos valores padronizados para freqüências intermediárias, o incremento de fase deve ser de 919123001.

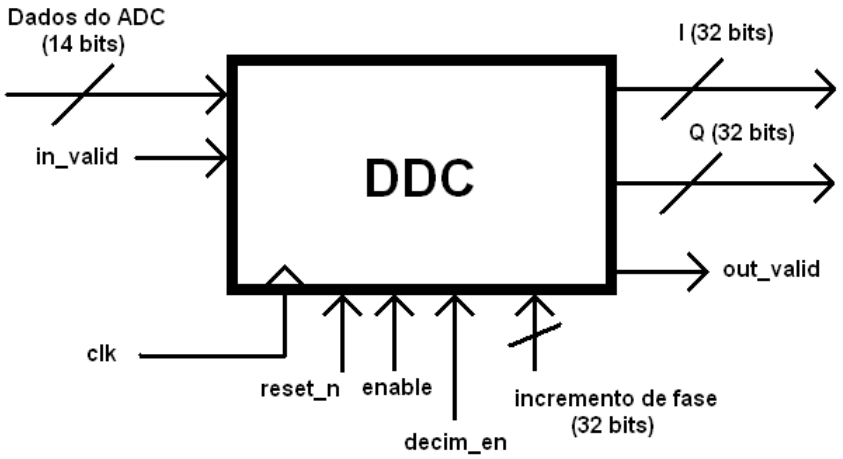

Figura 3.6: Entradas e saídas do DDC projetado

Tabela 3.2: Comparação dos usos dos recursos do FPGA por cada um dos algoritmos para implementação do NCO

Algoritmo	Número de Elementos Lógicos utilizados	Número de bits de memória utilizados	Multiplicadores utilizados
Pequena ROM	331	212992	0
Grande ROM	150	1835008	0
CORDIC	1787	0	0
Série de Taylor	236	10572	8

3.4.3 Filtro FIR Passa-Baixa

Após a mixagem do sinal digitalizado, proveniente do conversor AD, com as senóides em quadratura, geradas pelo NCO, faz-se necessária uma filtragem para selecionar somente o sinal desejado, já em banda-base. A definição dos parâmetros do filtro de resposta finita ao impulso (FIR - *Finite Impulse Response*) deve levar em conta a banda de freqüência do sinal que se deseja processar, atenuando as freqüências superiores indesejáveis.

Como o DDC proposto neste trabalho visa aplicações em sinais de voz, que ocupam a banda de 0 a 4 kHz, a filtragem realizada deve manter os sinais nesta faixa de freqüências. Contudo, tendo em vista a flexibilidade visada por um RDS, não seria interessante colocar neste estágio um filtro passa-baixa com freqüência de corte igual a 4 kHz, pois este se aplicaria somente a sinais de voz, ou a sinais limitados a esta banda de freqüências. A cada mudança de aplicação do rádio seria necessária uma alteração num componente de hardware do rádio, o DDC, o que dificultaria a reconfiguração do rádio. A estratégia mais prudente para não comprometer toda a capacidade de reconfiguração do terminal, neste caso, é a de aumentar a freqüência de corte do filtro passa-baixa de modo a permitir a utilização do DDC para mais tipos de aplicações, deixando que filtragens mais seletivas sejam feitas em software.

Os parâmetros definidos para o filtro passa-baixa estão mostrados na tabela 3.4. A arquitetura de aritmética distribuída e totalmente paralela, apesar de gastar mais elementos lógicos na implementação, tira vantagem do paralelismo que uma implementação em hardware possibilita e garante o melhor desempenho possível deste que é o estágio que exige maior esforço computacional do DDC. Uma freqüência de corte de 100 kHz oferece a possibilidade de demodulação de sinais de FM de banda-larga, abrindo a possibilidade

Tabela 3.3: Parâmetros do NCO utilizado no DDC

Algoritmo	CORDIC
Número de bits do Acumulador de fase	32
Número de bits de precisão da amplitude	14
Freqüência do *clock*	50 MHz
Máxima freqüência da senóide gerada	25 MHz
Resolução de freqüência	0.0116 Hz

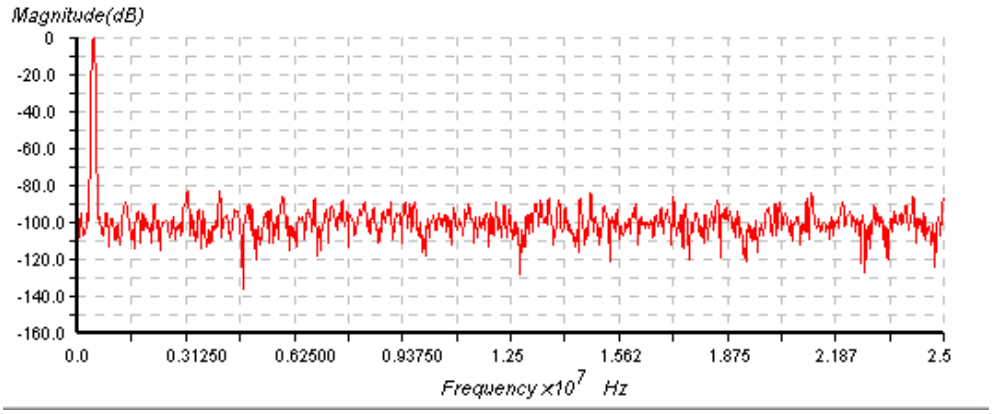

Figura 3.7: Resposta em freqüência do NCO utilizado no DDC

de se trabalhar com sinais em toda a banda de áudio, não se limitando somente a sinais de voz. A resposta em freqüência do filtro FIR passa-baixa é mostrada na figura 3.9.

Uma opção viável para aplicações que envolvam vídeo ou dados, que exigem uma banda de freqüências maior do que 100 kHz, é adicionar outro conjunto de coeficientes para o filtro FIR passa-baixa, com a freqüência de corte necessária para tal aplicação. Os *MegaCores* dos filtros FIR permitem a adição de um ou mais conjuntos de coeficientes para o filtro, com a seleção de qual conjunto se deseja utilizar através de um sinal de entrada. No caso de uma desejável troca de aplicação, seria possível selecionar qual dos filtros deveria ser utilizado por meio de um software que se comunicasse com o DDC.

Embora exista um componente dedicado especialmente a realizar a decimação do sinal, parte desta decimação pode ser realizada no filtro passa-baixa. Isto porque a taxa de *throughput* pode (e deve) ser reduzida, após a translação em freqüência do sinal desejado para a banda-base. No caso em que se trabalha com o sinal de maior largura de banda possível para este DDC, isto é, 100 kHz em banda-base, o critério de Nyquist estabelece uma taxa mínima de 200 kSamples/s para que a reconstrução do sinal seja possível. Ou seja, como a taxa de *trhoughput* poderia ser reduzida de 50 MSamples/s para 200 kSamples/s, poder-se-ia realizar uma decimação por um fator de 250. Como na saída do filtro passa-baixa já é realizada uma decimação por 128, não poderia haver outra decimação após o filtro FIR passa-baixa, uma vez que o menor fator de decimação possível é 2, e nesse caso a decimação total do sinal seria de 256, valor maior do que o estabelecido como máximo pelo teorema da amostragem.

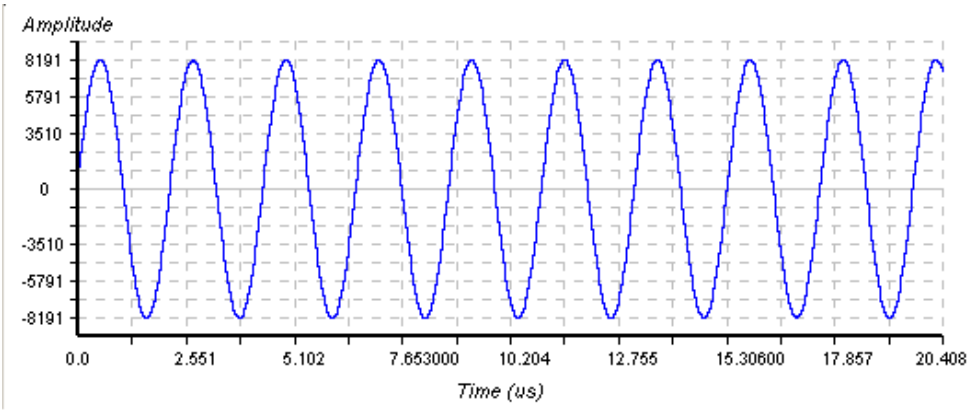

Figura 3.8: Resposta no tempo do NCO utilizado no DDC

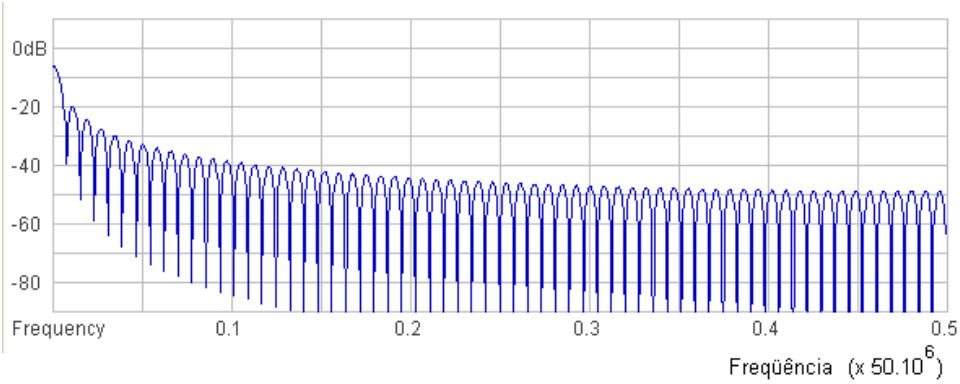

Figura 3.9: Resposta em freqüência do filtro FIR passa-baixa utilizado no DDC

3.4.4 Decimador

O estágio final antes de se entregar os dados digitalizados para serem processados em software pelo processador do rádio é a decimação. A operação de decimação por um fator N é simplesmente a escolha de 1 em cada N valores de entrada, reduzindo por N a taxa de *throughput*, tal como é mostrado na figura 3.10.

A escolha do fator de decimação N depende da banda ocupada pelo sinal em banda-base. Esta escolha deve ser tal que o critério de Nyquist seja sempre respeitado, o que garantirá que o sinal possa ser reconstruído em software. Como margem de segurança, e para diminuir a seletividade necessária no filtro de reconstrução, o sinal pode ser representado com uma taxa de amostragem maior do que a taxa mínima ditada pelo critério de Nyquist. Como trabalha-se com a possibilidade de receber sinais que ocupem uma banda de até 100 kHz em banda-base, e o sinal está sendo enviado a uma taxa de 50 MSamples/s, pode-se realizar uma decimação por um fator de até 250. Como já foi feita uma decimação por 128 no filtro FIR passa-baixa, não poderia se realizar mais uma decimação, visto que a menor decimação é por um fator 2, e que neste caso, totalizaria uma decimação por 256, violando o critério de Nyquist. Neste caso, os dados devem ser entregues diretamente da saída do filtro passa-baixa ao processador.

Porém, no caso em que se trabalha com sinais de menor banda de freqüência, pode-se realizar uma decimação adicional. No caso de sinais de áudio em banda-base, que ocupam a banda de 0 a 20 kHz, para satisfazer o critério de Nyquist deve-se ter uma taxa de 40 kSamples/s no mínimo, o que possibilitaria uma decimação máxima por um fator de 1250. Como uma decimação por 128 já foi feita no filtro passa-baixa, pode-se fazer uma decimação por um fator de 9, sem comprometer a reconstrução do sinal.

Tabela 3.4: Parâmetros do filtro FIR passa-baixa utilizado no DDC

Tipo	FIR Passa-baixa
Freqüência de corte	100 kHz
Número de coeficientes	37
Freqüência de amostragem	50 MHz
Estrutura	Aritmética distribuída totalmente paralela
Número de bits da entrada	28
Número de bits da saída	32
Fator de decimação	128

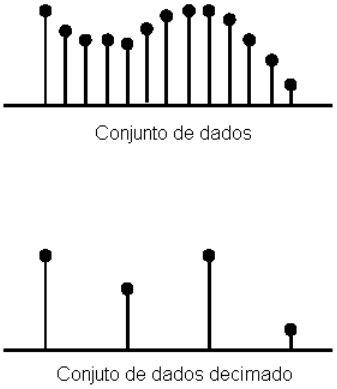

Figura 3.10: Decimação

No DDC projetado neste trabalho foram colocados os blocos decimadores de fator de decimação 9, cascateados com os filtros FIR passa-baixas. O uso dos decimadores, no entanto, deve ser restrito a aplicações que permitam uma decimação por tal fator sem infringir o teorema da amostragem. Para ativação do decimador, foi colocado um sinal de ativação do decimador como entrada do DDC, permitindo o controle da decimação pelo dispositivo de processamento.

3.5 RECURSOS DO FPGA UTILIZADOS PELO DDC

Após a implementação do DDC descrito neste capítulo na linguagem de descrição de hardware Verilog, foi realizada a síntese do mesmo para o FPGA Altera Cyclone II 2C35, o FPGA da placa de desenvolvimento Altera DE2. A tabela 3.5 mostra os recursos utilizados pelo DDC implementado.

Tabela 3.5: Recursos do FPGA utilizados pelo DDC

	Utilizados	**Disponíveis no Cyclone II**	**% Utilizada**
Elementos Lógicos	13768	33216	41%
Bits de Memória	960	483840	< 1%
Multiplicadores	4	70	6%
PLLs	0	4	0%

4 RESULTADOS OBTIDOS

4.1 SIMULAÇÃO DO DDC

4.1.1 Disposições gerais sobre as simulações realizadas

Após a conclusão da síntese dos códigos escritos em linguagem de descrição de hardware, o passo seguinte para a validação do projeto sintetizado é a execução das simulações. É por meio das simulações que se verifica o comportamento esperado do circuito recém-implementado, podendo detectar alguns erros de projeto sem a necessidade de efetuar o carregamento do *netlist* no FPGA para a verificação de tais erros.

Foram realizados três tipos de simulação do DDC implementado: a simulação funcional do projeto, a simulação temporal e a simulação em MATLAB. A primeira foi realizada no ambiente de desenvolvimento Quartus II e nela pôde-se verificar as formas de onda da saída, numa simulação das respostas das portas lógicas do FPGA a determinadas entradas, sem levar em consideração o atraso em cada porta. Já a segunda, também realizada com auxílio do software Quartus II, fornece o mesmo tipo de análise, porém considerando o atraso das portas lógicas, o que permite que se verifique a influência de tais atrasos no funcionamento do projeto.

Nas simulações funcional e lógica, devido à indisponibilidade de um conversor AD, emulou-se um sinal vindo do conversor AD, através de uma senóide digital de freqüência 458 kHz. Este sinal emula um sinal da faixa de voz modulado em AM-SSB, já convertido para a freqüência intermediária de 455 kHz e digitalizado à taxa de 50 MSamples/s.

A simulação em MATLAB foi realizada a partir de um modelo do DDC implementado na linguagem de script do MATLAB. Tal tipo de simulação foi necessária devido à impossibilidade em se conseguir uma análise em domínio da freqüência dos sinais de saída do DDC a partir das simulações realizadas no Quartus II. Tal análise permite verificar a correção matemática das operações realizadas pelo DDC sobre o sinal que vem do conversor AD.

Na simulação do MATLAB pôde-se emular um sinal de entrada mais similar a um sinal prático. Com o algoritmo para a obtenção da transformada rápida de Fourier (FFT) oferecido pelo MATLAB, pôde-se observar o espectro de freqüências do sinal em cada ponto do DDC, e como a conversão de freqüência é feita.

4.1.2 Simulação Funcional

Através da simulação funcional pôde-se verificar se as funções lógicas presentes no DDC estavam realizando corretamente as operações desejadas.

A primeira situação passível de análise é a passagem do sinal pelos *mixers*. A multiplicação do sinal digitalizado entregue ao DDC pelas senóides geradas pelo NCO deve ter um atraso muito pequeno, limitado ao atraso das portas que o compõem, uma vez que os *mixers*, conforme explicitado no capítulo 3, são multiplicadores combinacionais. Na figura 4.1 observa-se as formas de onda de saída do NCO, com alguns de seus sinais internos sendo exibidos para facilitar a análise neste tipo de simulação.

Na figura 4.1 é mostrado o sinal emulado como entrada de dados do DDC, chamado de *signal_from_adc*. Também são exibidos os valores do seno e do cosseno gerados pelo NCO (*cosseno_nco* e *seno_nco*), bem como os resultados da mixagem entre o sinal digitalizado e as senóides, que servem de entrada para o filtro passa-baixa, e na figura 4.1 são chamados de *in_filter_i* e *in_filter_q*. Tomando arbitrariamente o intervalo de $33.71\mu s$ a $33.73\mu s$, tempo correspondente a um período de *clock*, pode-se verificar se a operação reali-

zada pelo mixer está correta:
$signal\_from\_adc \times cosseno\_nco = 5722 \times 8095 = 46319590$
$signal\_from\_adc \times seno\_nco = 5722 \times 1234 = 7060948$
Nota-se, então, que o sinal entregue aos filtros passa-baixa corresponde realmente ao resultado da multiplicação entre o sinal que vem do conversor AD e as senóides em quadratura.

O estágio posterior corresponde à filtragem digital dos sinais mixados. Como a simulação funcional do Quartus II fornece apenas as formas de onda das saídas do projeto implementado, fica difícil a verificação da operação de filtragem. Para que esta verificação seja mais simples, é necessária uma ferramenta que forneça o espectro de freqüências dos sinais na entrada e saída do filtro. E esta é justamente a função das simulações em MATLAB neste trabalho, exibidas nesta seção, após as simulações temporais.

Contudo, a simulação temporal permite que se verifique se a decimação por um fator 128 na saída dos filtros passa-baixa está sendo feita de maneira correta. Para isso, basta verificar os momentos em que as saídas dos filtros passa-baixa são atualizadas. Nas figuras 4.2 e 4.3 estes momentos das atualizações das saídas dos filtros são mostrados, com as saídas dos filtros sendo denotadas por *in_decimator_i* e *in_decimator_q*, para os filtros dos ramos que darão origem a I e Q, respectivamente.

Na figura 4.2 é mostrado o início de uma das saídas dos filtros passa-baixa, mais precisamente no instante $33.63\mu s$. Estas saídas se mantêm até o instante $36.19\mu s$, mostrado na figura 4.3. Durante o intervalo de tempo entre estas duas atualizações das saídas dos filtros, transcorreram-se $2.56\mu s$. Como os dados mixados são entregues pelos *mixers* a cada ciclo de *clock*, isto é, a cada 20 ns, constata-se que os filtros passa-baixa mantém uma saída para cada 128 entradas, realizando a decimação de maneira correta.

A mesma estratégia pode ser adotada para a verificação do funcionamento dos decimadores. Nas figuras 4.4 e 4.5 são mostradas duas atualizações subseqüentes das saídas dos decimadores, denominadas I e Q, e que também são os barramentos de saída de dados do DDC. As saídas dos decimadores são mantidas constantes durante o período de $31.19\mu s$ até $54.23\mu s$, ou seja, durante um período de $23.04\mu s$. Como os dados são entregues ao DDC a cada 20 ns, tem-se uma saída para cada 1152 entradas, reduzindo a taxa de *throughput* por um fator de 1152. Como se definiu uma decimação por um fator de 128 na saída dos filtros passa-baixa, os decimadores estão realizando uma decimação por 9, conforme estabelecido no projeto do DDC.

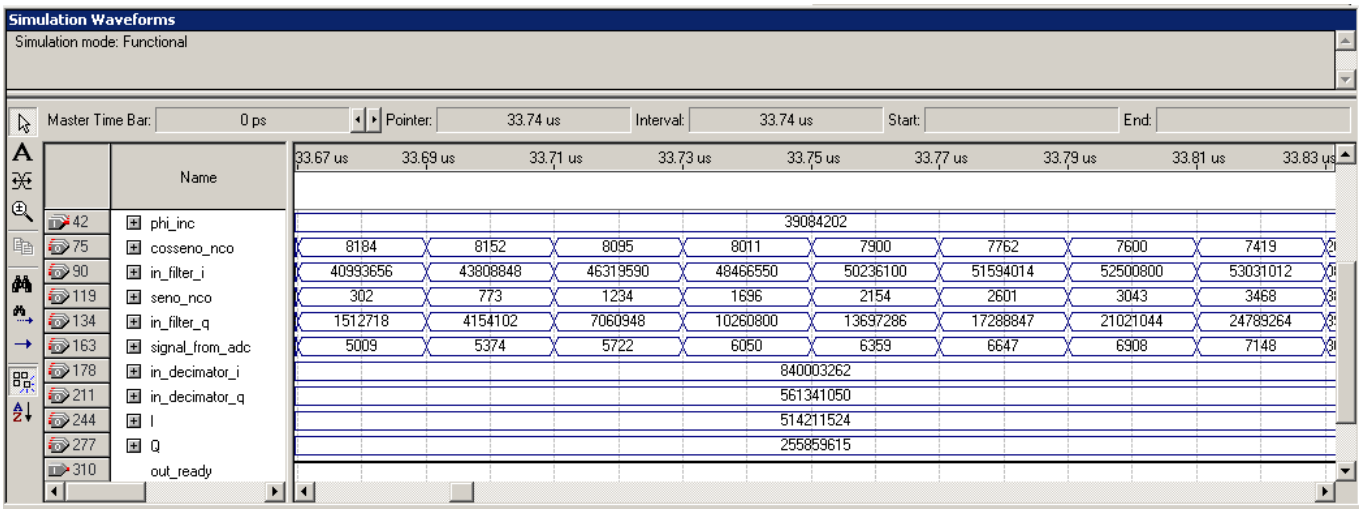

Figura 4.1: Comportamento dos *mixers* mostrado pela simulação funcional

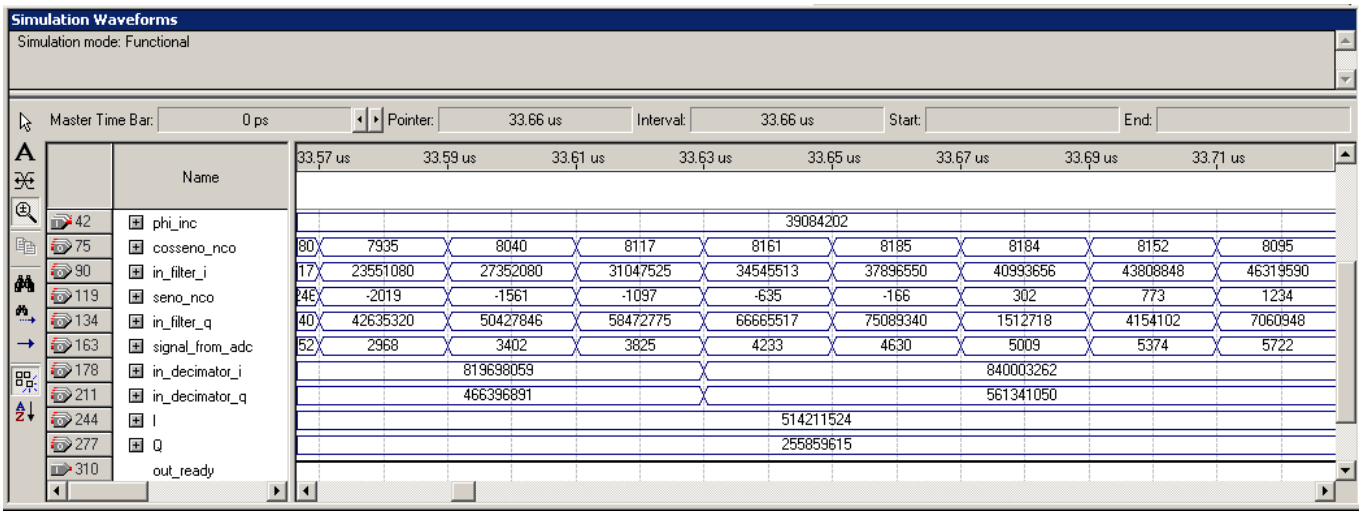

Figura 4.2: Instante de tempo em que as saídas dos filtros passa-baixa são atualizadas

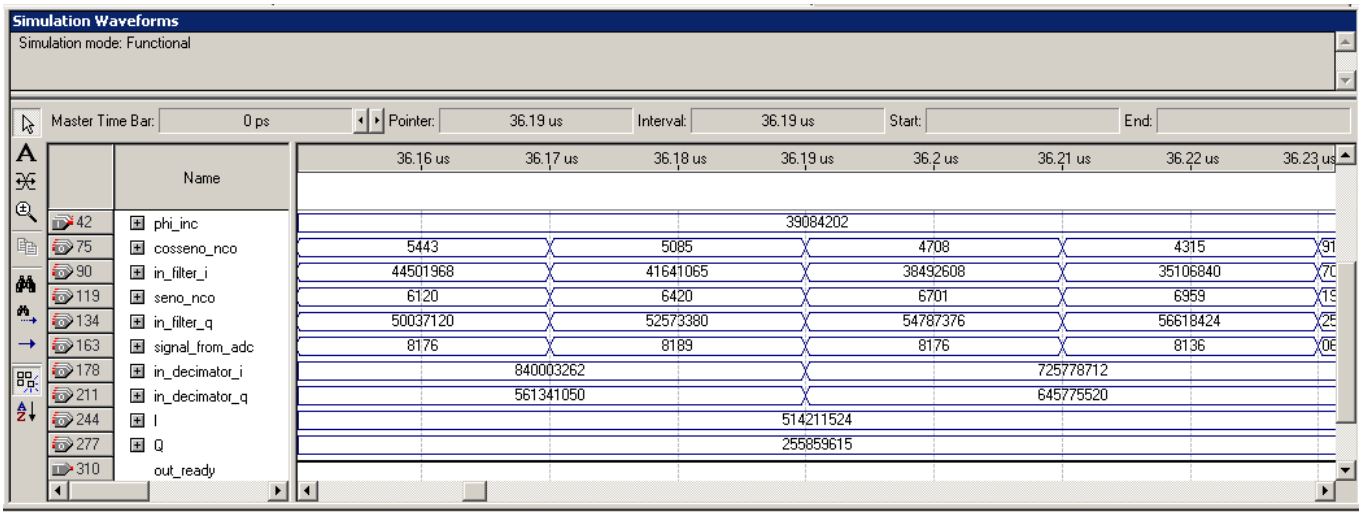

Figura 4.3: Atualização subseqüente da saída dos filtros passa-baixa, em relação ao instante mostrado na figura 4.2.

4.1.3 Simulação Temporal

As simulações temporais simulam as formas de onda das saídas do circuito projetado, nos mesmos moldes das simulações funcionais, com a diferença de que levam em consideração o atraso registrado nas portas lógicas do FPGA. Através deste tipo de simulação é que se verifica a influência do atraso das portas lógicas nas formas de onda das saídas do circuito. Dependendo do tempo necessário para a estabilização do sinal de saída é que se chega à sincronização necessária para o funcionamento correto do circuito.

Na figura 4.6 é mostrado o atraso causado pelas portas lógicas no sinal de saída dos *mixers*. Nota-se, da figura 4.6, que a atualização dos sinais de saída dos *mixers*, *in_filter_i* e *in_filter_q* está cerca de 2 nanossegundos atrasada em relação à atualização dos dados que entram no mixer, tempo este que representa o atraso imposto pelas portas lógicas dos *mixers*.

Para a análise do desempenho e do atraso do sinal em cada um dos filtros, é necessário mais do que considerar o tempo de atraso das portas. Isto se deve ao fato de os filtros FIR não serem circuitos combinacionais, onde as saídas dependem somente do valor das entradas em um dado instante. É necessário, portanto, um certo período de processamento entre o instante de entrada da amostra no filtro e o instante

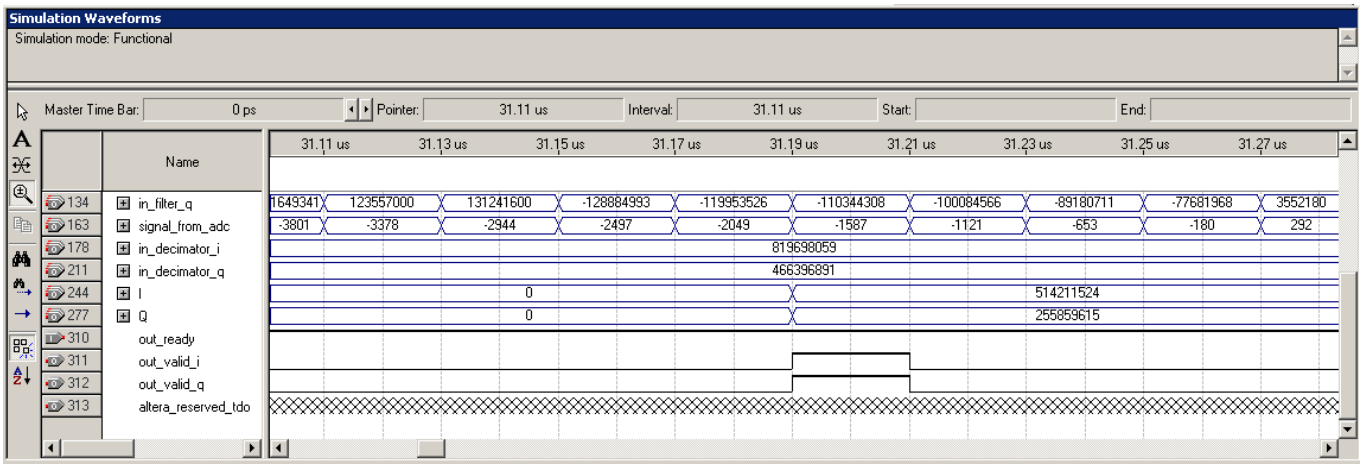

Figura 4.4: Instante de tempo em que as saídas dos decimadores são atualizadas

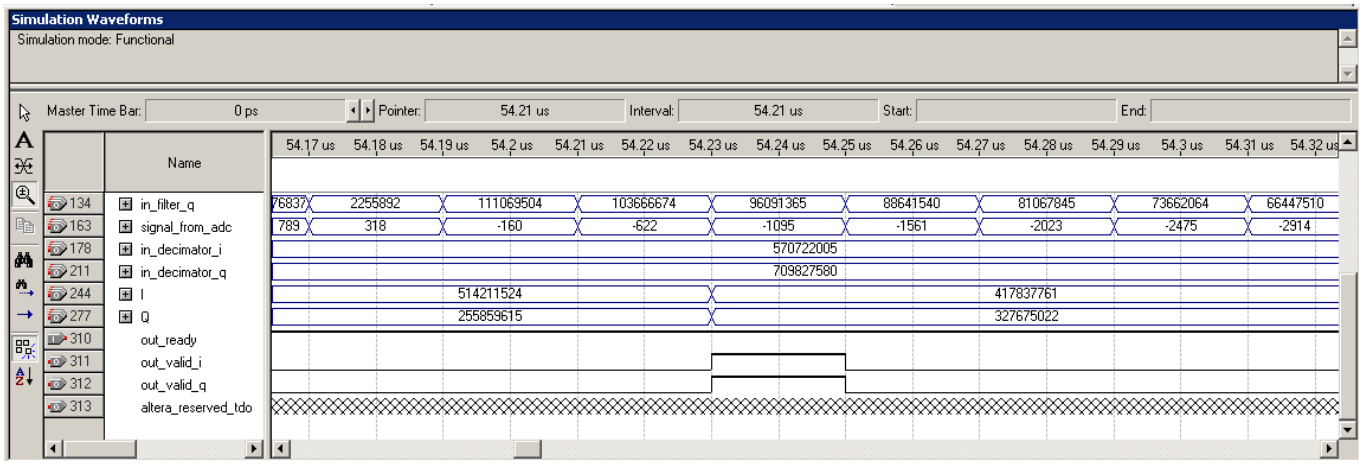

Figura 4.5: Atualização subseqüente da saída dos decimadores, em relação ao instante mostrado na figura 4.4.

em que a saída é repassada ao decimador. Na figura 4.7 é exibido o momento na simulação em que o DDC começa a receber dados do gerador de sinal que emula o comportamento do conversor AD. Este instante servirá de base para a análise do tempo gasto para o processamento do sinal nos filtros.

Da figura 4.7, vê-se que o ínicio da entrega das amostras para o DDC ocorre no instante $t = 620ns$. Nesta análise, pode-se considerar desprezível o atraso causado pelos *mixers*. Na figura 4.8 é mostrado o instante em que os filtros passa-baixa entregam as primeiras amostras filtradas aos decimadores. Nota-se que os filtros passa-baixa liberam as primeiras amostras na sua saída, aproximadamente, no instante $t = 5.48\mu s$, o que representa uma diferença de $4.86\mu s$ entre os instantes da chegada da primeira amostra e a emissão da primeira saída.

A figura 4.9, por sua vez, mostra o instante em que os decimadores emitem a primeira resposta às entradas que começaram a lhes ser entregues no instante mostrado na figura 4.8. Da figura 4.9, tem-se que o início da emissão de amostras se dá, aproximadamente, no instante de tempo $t = 31.2\mu s$. Isto representa um intervalo de tempo de $26.34\mu s$ entre o instante em que chega a primeira amostra à entrada dos decimadores e o instante em que é emitida a primeira saída.

No entanto, cabe a ressalva de que, mesmo sendo grande a diferença entre os tempos verificados entre a chegada da primeira entrada e a emissão da primeira saída para os filtros FIR e os decimadores, os decimadores não representam o bloco que exige maior esforço computacional. Este papel, pela própria complexidade das operações envolvidas, cabe ao filtro FIR. O tempo maior para a emissão das saídas do

decimador deve-se ao fato de que que a taxa de dados que chegam à entrada deles é 128 vezes menor do que a taxa de dados na entrada do filtro passa-baixa. De fato, o tempo que os filtros FIR demoram para liberar 9 amostras estabelece um limite mínimo para o tempo de processamento dos decimadores.

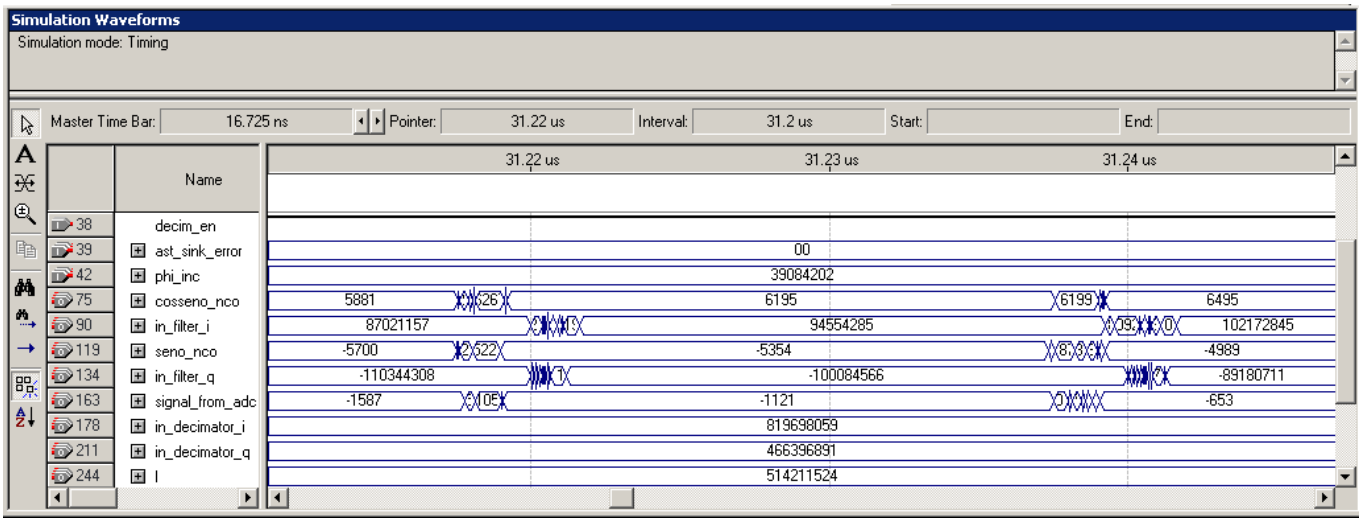

Figura 4.6: Atraso causado pelas portas lógicas na saída dos *mixers*

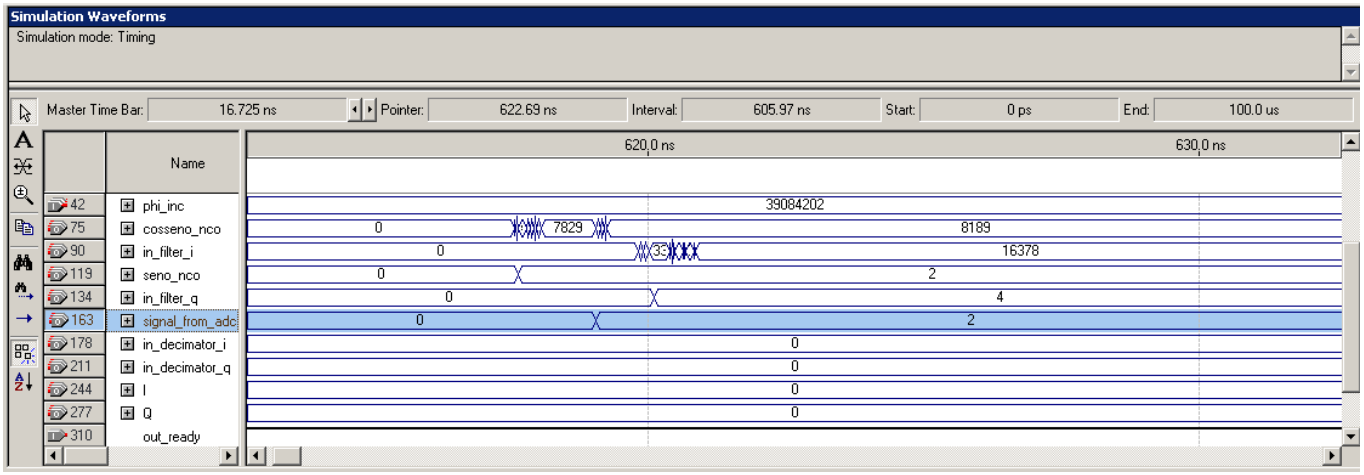

Figura 4.7: Instante em que o DDC começa a receber dados (t = 620 ns).

4.1.4 Desempenho da implementação em hardware do DDC

Um importante parâmetro para a caracterização do desempenho do DDC é o tempo estimado para o processamento do sinal. Via de regra, devido ao paralelismo no processamento, implementações em hardware têm desempenhos satisfatórios no que diz respeito ao tempo de processamento. A partir das simulações temporais executadas, buscou-se determinar o tempo de processamento de uma amostra única que entra no DDC, pois este é um dado que se reflete diretamente no atraso introduzido no sinal pelo DDC.

Com base nas figuras 4.7 a 4.9, foi montada a tabela 4.1, que mostra os instantes de tempo em que chega a primeira amostra ao DDC e os instantes em que os filtros FIR passa-baixa e os decimadores respondem a estas amostras, que passam a chegar à entrada do DDC a uma taxa de 50 MSamples/s.

Os tempos mostrados na tabela 4.1 refletem apenas o tempo de resposta do DDC ao início da entrada de dados, servindo como parâmetro para estimar, no pior caso, o tempo de processamento de uma amostra de dados que passa pelo DDC. Contudo, devido ao fato da implementação em hardware valer-se de um

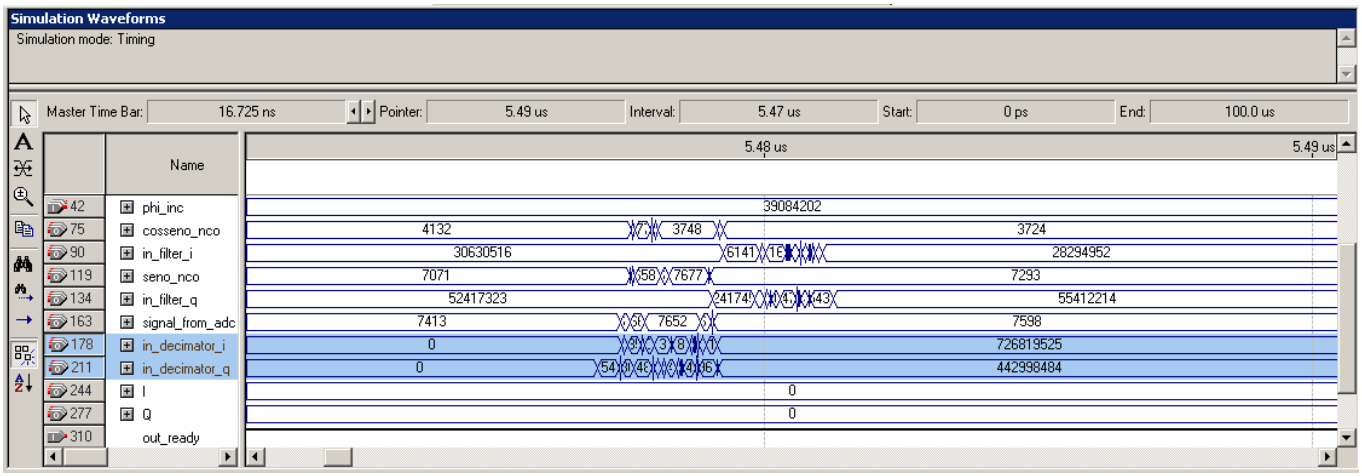

Figura 4.8: Instante em que os filtros FIR passa-baixa repassam as primeiras amostras filtradas aos decimadores.

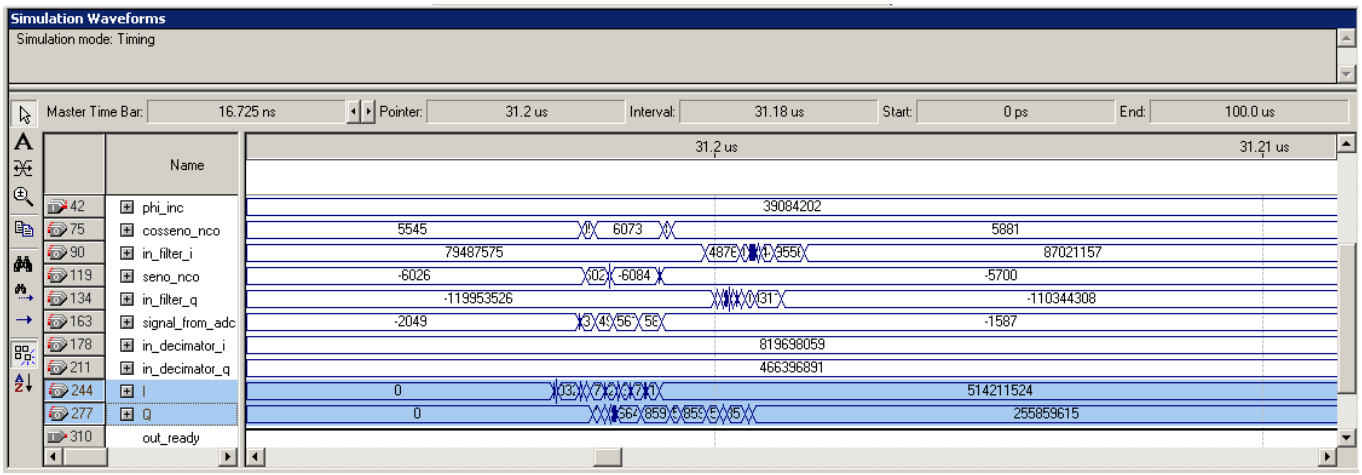

Figura 4.9: Instante em que os decimadores emitem os primeiros resultados.

processamento paralelo dos dados que chegam ao DDC, o tempo médio de processamento é bem menor do que o mostrado na tabela 4.1.

Observando o instante de atualização da saída do DDC mostrado na figura 4.10, tem-se que, no instante dado por $t = 54.24\mu s$, já foram processadas, pelo menos, 1152 amostras do sinal vindo do conversor AD. Isto porque, no filtro FIR realiza-se uma decimação por um fator de 128 e, em seguida, outra decimação por um fator de 9, fazendo com que o tempo de duração de cada componente complexa I e Q do sinal seja equivalente ao tempo de 1152 amostras que vêm do conversor AD. Neste caso, calculando o tempo médio de processamento para estas 1152 amostras:

$$\bar{t} = \frac{54.24 - 0.62}{1152} = 4.655 \times 10^{-2} \mu s = 46.55 ns$$

Esta grande diferença entre o tempo de processamento de uma amostra deve-se ao paralelismo do processamento em hardware, não referindo-se somente à simultaneidade de processamento das amostras I e Q, mas do processamento contínuo e ininterrupto das amostras que chegam à entrada do DDC. Desta forma, os dados começam a ser processados assim que chegam ao DDC, não tendo que esperar que o processamento completo de uma amostra seja terminado para que só então o próximo dado seja tratado, como aconteceria em uma implementação em software.

Tabela 4.1: Medida dos tempos de resposta do DDC à entrada de dados provenientes do conversor AD.

Evento	Instante de tempo
Início do recebimento de dados	$t = 620ns$
Primeira resposta dos filtros FIR	$t = 5.48\mu s$
Primeira resposta dos decimadores	$t = 31.2\mu s$
Primeira resposta do DDC	$t = 31.2\mu s$

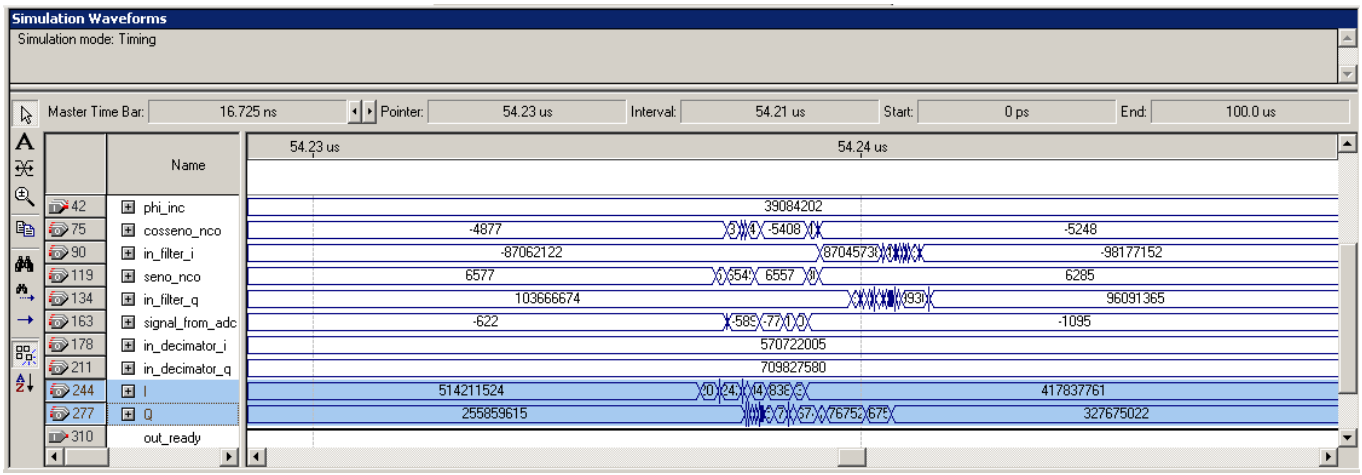

Figura 4.10: Instante subseqüente de atualização das saídas em relação ao instante mostrado na figura 4.9

4.1.5 Simulação em MATLAB

A simulação do circuito implementado em MATLAB tem em vista apenas fazer uma análise espectral das operações realizadas pelo DDC, verificando o espectro obtido para o sinal em vários pontos do circuito, desde o ponto de entrada até a saída.

Para isto, foi implementado um modelo em MATLAB do DDC projetado, a partir dos modelos gerados automaticamente para os blocos Altera *MegaCores* utilizados na construção do DDC. Neste modelo o sinal é processado em blocos de 2048 amostras.

Para as simulações realizadas, foi utilizado como um sinal de entrada um sinal banda-passante de largura de banda de aproximadamente 200 kHz centrado em uma freqüência intermediária de 10.7 MHz, emulando um típico sinal de FM comercial. Esta é uma mudança significativa de sinal em relação às simulações funcionais e temporais realizadas e deveu-se a dois fatores: a maior facilidade em se gerar sinais no MATLAB, e à necessidade de testar o desempenho do DDC em condições que utilizassem a sua máxima largura de banda.

Sejam os pontos enumerados de 1 a 7, no DDC mostrado na figura 4.12. O ponto 1, na entrada do DDC, diz respeito ao sinal vindo do conversor AD, e o seu espectro é mostrado na figura 4.11. Os pontos 2 e 3 referem-se ao seno e cosseno digitais gerados pelo NCO, representados no domínio da freqüência, respectivamente, nas figuras 4.13 e 4.14. Os pontos 4 e 5 são os sinais complexos em fase e em quadratura, resultantes da mixagem entre o sinal vindo do conversor AD e as senóides geradas pelo NCO, e seus espectros de freqüências são mostrados, respectivamente, nas figuras 4.15 e 4.16. Os pontos 6 e 7 são as saídas em quadratura do DDC, e estão representadas no domínio da freqüência nas figuras 4.17 e 4.18.

Analisando as figuras de 4.11 a 4.17, verifica-se que o DDC está realizando corretamente as operações que dele se espera: a translação em freqüência do sinal da freqüência intermediária para a banda-base e em seguida realizando uma filtragem, que permitem que a taxa de *throughput* seja reduzida. Percebe-se,

contudo, que os sinais em quadratura na saída, mostrados nas figuras 4.17 e 4.18 sofreram uma distorção pelos filtros FIR passa-baixa, e que os sinais decimados provavelmente necessitem passar por um filtro equalizador. Esta distorção acontece devido ao filtro FIR passa-baixa não apresentar ganho constante em toda a sua largura de faixa. Como o sinal emulado ocupa toda a banda passante do filtro, a distorção por ele imposta ficou evidente. Contudo, em sinais de banda estreita, esta distorção seria praticamente imperceptível.

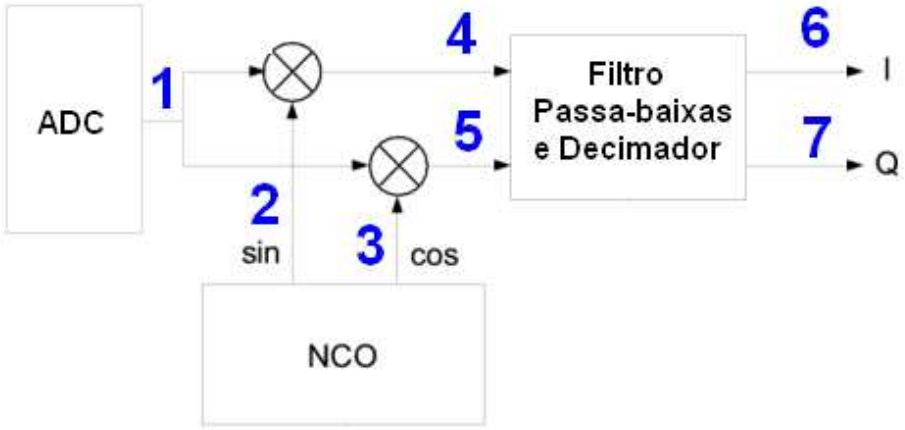

Figura 4.11: Pontos do DDC onde foram feitas análises do espectro de freqüências dos sinais.

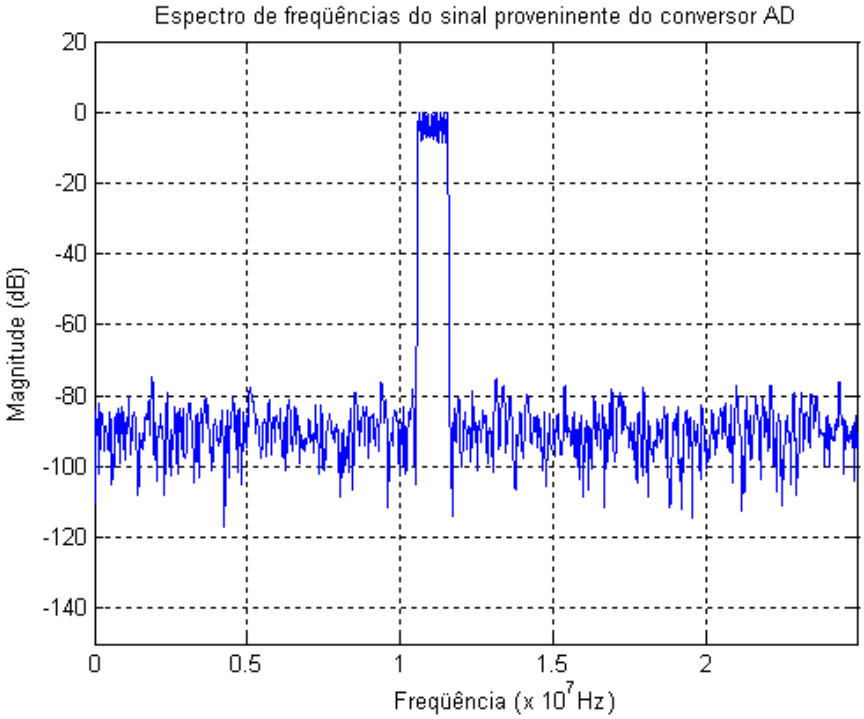

Figura 4.12: Espectro de freqüências do sinal proveninente do conversor AD (Ponto 1 da figura 4.11).

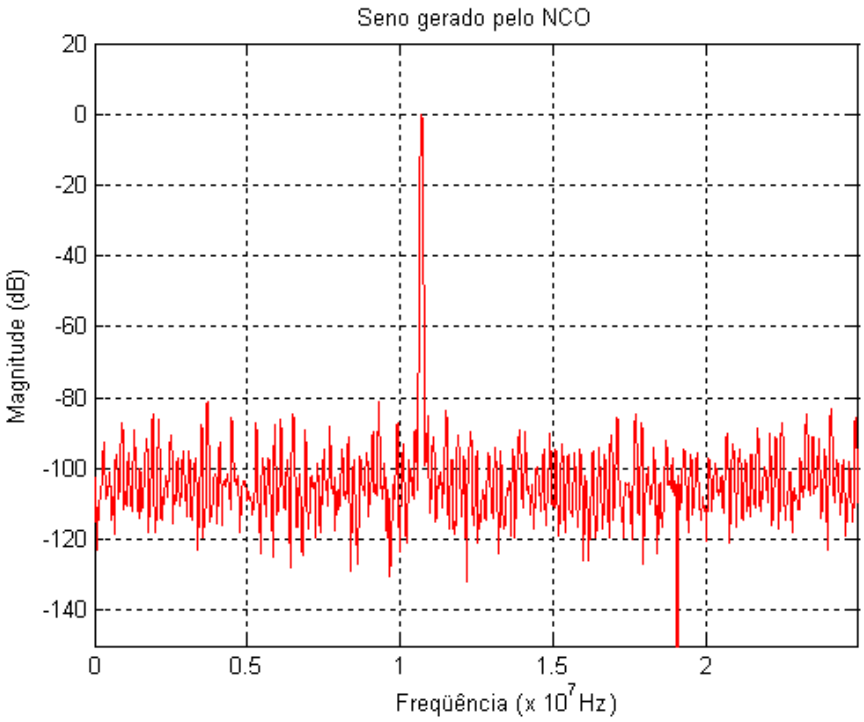

Figura 4.13: Espectro de freqüências do seno gerado pelo NCO (Ponto 2 da figura 4.11).

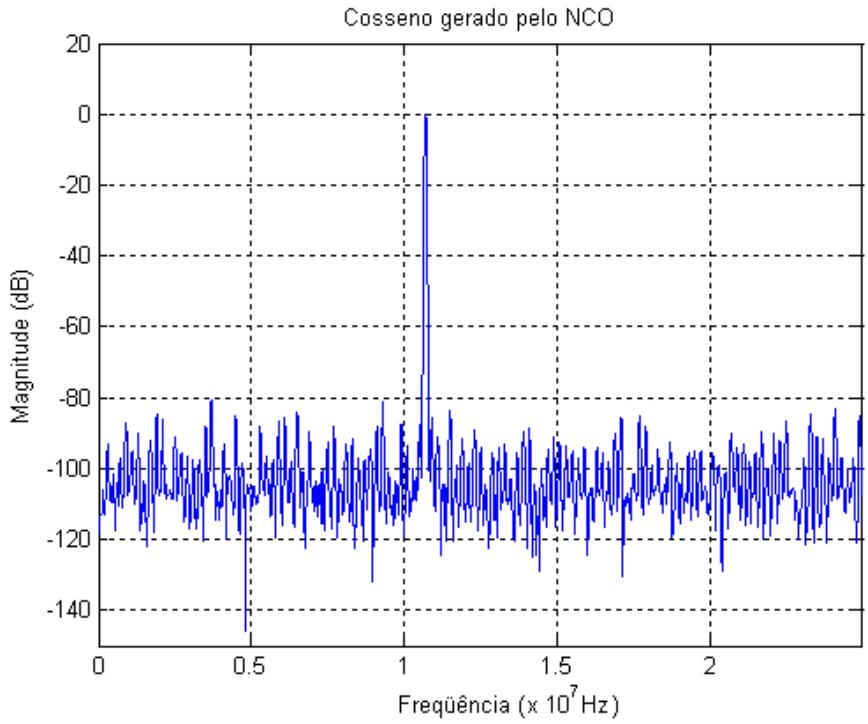

Figura 4.14: Espectro de freqüências do cosseno gerado pelo NCO (Ponto 3 da figura 4.11).

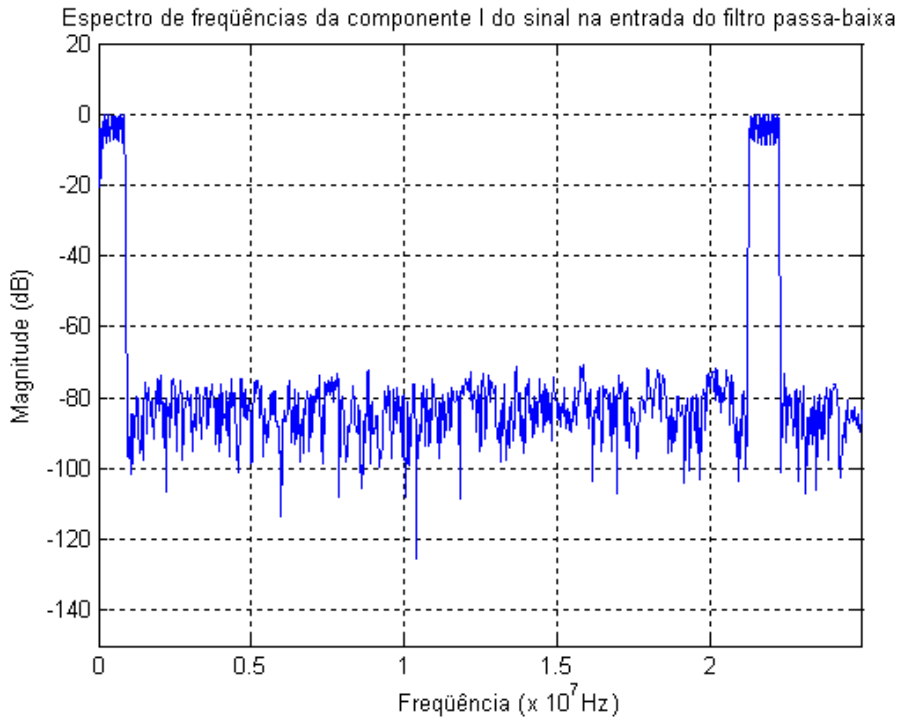

Figura 4.15: Espectro de freqüências da componente complexa em fase (I) do sinal na entrada do filtro passa-baixa (Ponto 5 da figura 4.11).

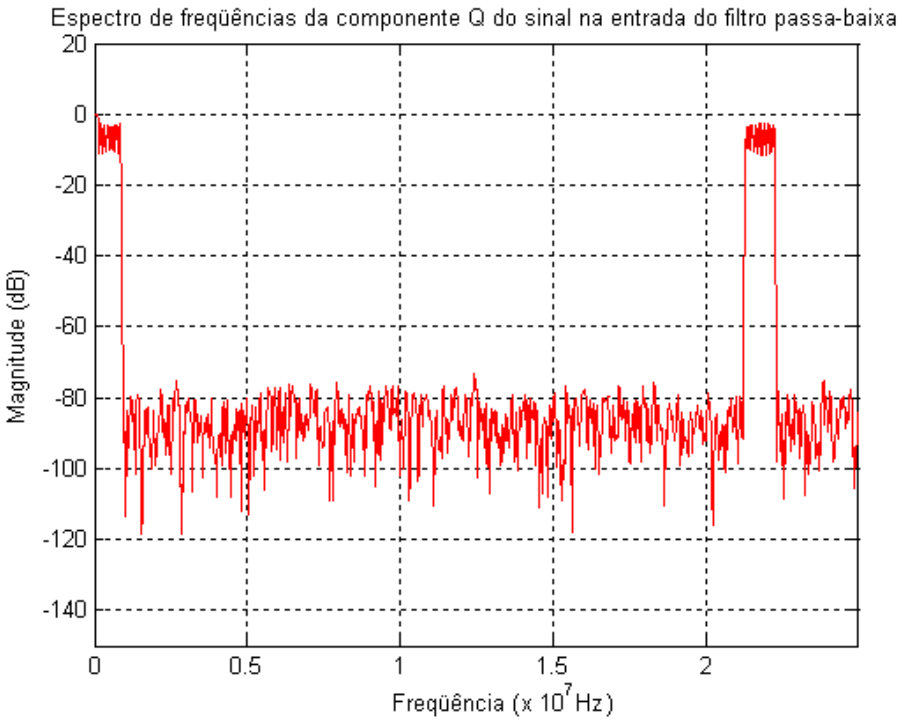

Figura 4.16: Espectro de freqüências da componente complexa em quadratura (Q) do sinal na entrada do filtro passa-baixa (Ponto 4 da figura 4.11).

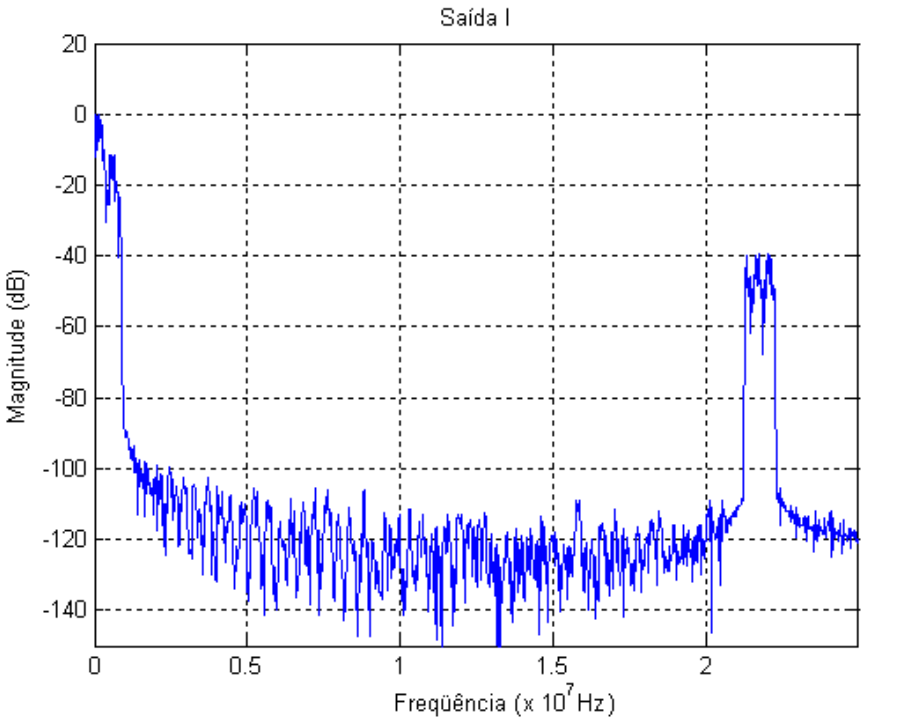

Figura 4.17: Espectro de freqüências da componente complexa em fase (I) na saída do DDC (Ponto 6 da figura 4.11).

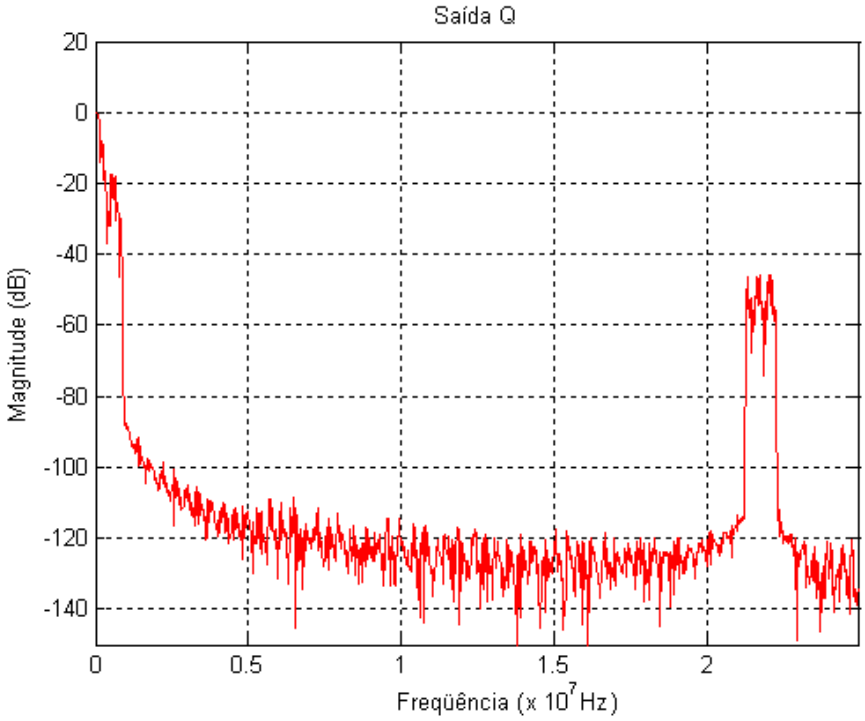

Figura 4.18: Espectro de freqüências da componente complexa em quadratura (Q) na saída do DDC (Ponto 7 da figura 4.11).

5 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho, foi projetado e desenvolvido um *Digital Downconverter* (DDC) para um protótipo embarcado de Rádio Definido por Software. A metodologia utilizada no transcorrer deste trabalho, apresentada e discutida em detalhes ao longo do texto, partiu da busca por alcançar o objetivo principal, que era obter um DDC que cumprisse a sua função de reduzir a taxa de dados entregue ao dispositivo de processamento do protótipo de RDS e que, ao mesmo tempo, não limitasse significativamente a capacidade de reconfiguração do terminal de rádio no qual ele esteja inserido.

Estabelecida a arquitetura do DDC a ser implementado, com as entradas que deve receber e as saídas que deve fornecer, bem como as operações que deve realizar sobre o sinal digitalizado, partiu-se então para o processo de desenvolvimento do dispositivo, que fez uso dos *MegaCores* Altera nos blocos que compõem o DDC. Durante todo o processo de desenvolvimento, foram feitos ajustes de parâmetros que tinham em vista melhora de desempenho, economia de recursos da pastilha do FPGA, buscando sempre um ponto de equilíbrio entre o desempenho do dispositivo e a possibilidade de integração com outros dispositivos dentro do mesmo FPGA.

Tendo em vista as especificações a serem atendidas, pode-se concluir que o objetivo principal do trabalho foi alcançado, com a implementação em FPGA do DDC proposto.

A etapa seguinte à implementação do DDC foi a da realização dos testes de validação do dispositivo. Nesta fase, foram realizadas simulações do *design* proposto, pois devido ao fato de o protótipo de RDS no qual o DDC desenvolvido deve se inserir estar ainda em fase inicial de desenvolvimento, as simulações com a emulação dos sinais de entrada oriundos do ADC eram os meios de que se dispunha para verificar a funcionalidade do produto final.

No ambiente de desenvolvimento Quartus II foram realizadas as simulações funcionais e temporais do DDC. Descritas em detalhes no capítulo 4, estas simulações foram muito importantes na verificação das operações realizadas em cada estágio por qual o sinal passava, na verificação do atraso imposto pelas portas lógicas e, talvez como resultados mais importantes, a estimativa do tempo necessário para processar uma amostra dos dados, que corresponderia ao *delay* introduzido pelo DDC na recepção do sinal.

Contudo, por mostrarem apenas as formas de onda dos sinais de saída, a análise que as simulações funcionais e temporais do Quartus II não se mostrava suficiente para validação do DDC. Como, por definição, o funcionamento de um DDC está vinculado à realização de uma translação em freqüência do sinal digitalizado para a banda-base, seria necessária também uma análise espectral do sinal processado em diversos pontos do DDC. Para realização de tal análise, foi utilizado um modelo computacional do DDC em MATLAB, através do qual pôde-se obter a representação no domínio da freqüência de sinais em vários pontos do DDC.

Os resultados obtidos nas simulações refletiram o que se desejava obter na validação do funcionamento do DDC: um tempo de processamento que não introduz um atraso perceptível no processamento do sinal, e a realização da operação de conversão digital de freqüência de maneira correta.

O próximo passo a ser dado na linha natural de continuação deste trabalho é a etapa que consiste em escrever um *device driver* para o DDC implementado, de modo que ele possa ser comandado por um processador implementado na mesma pastilha de FPGA, como por exemplo o NIOS II. A obtenção desta comunicação do DDC com o NIOS, através do *device driver*, representaria o estabelecimento das condições necessárias para a plena validação do projeto apresentado neste trabalho. Através desta comunicação, os dados entregues pelo DDC ao processador poderiam, por exemplo, ser armazenados em arquivos de dados para análise posterior em MATLAB. Analisando então estes dados gravados pelo NIOS em arquivos de dados, seria possível o ajuste fino de alguns parâmetros de projeto, de modo a obter um melhor funcionamento do DDC.

Outro importante resultado a ser obtido após o estabelecimento da comunicação entre o DDC e um processador implementado no FPGA, é a comparação entre o desempenho desta implementação em hardware do DDC com uma implementação do mesmo processamento realizado por ele em software. Neste caso, por ambas as implementações utilizarem o mesmo dispositivo de processamento, estabelecer-se-ia uma comparação justa entre o desempenho em hardware e o desempenho em software da função desejada, sendo que o resultado desta comparação poderia trazer a comprovação da necessidade da implementação em hardware do DDC.

Para a finalização de um protótipo de Rádio Definido por Software, ainda seria necessário acrescentar as duas extremidades do terminal: um *front-end* RF e o processamento em software dos dados entregues pelo DDC ao processador. Além do processamento do sinal, tanto o DDC, quanto o conversor AD e o *front-end* devem ser, de alguma forma, controlados por parâmetros definidos em software, de modo que a reconfigurabilidade almejada para um RDS seja alcançada. Neste ponto, também seria interessante a implantação da arquitetura SCA, a fim de agregar também ao protótipo as características de escalabilidade e interoperabilidade.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] KENINGTON, P. B. *RF and baseband techniques for software defined radio*. [S.l.]: Artech House, 2005.

[2] MITOLA, J. *Software Radio Architecture*. [S.l.]: Wiley, 2000.

[3] ISOMäKI, P.; AVESSTAH, N. *An Overview of Software Defined Radio Technologies*. [S.l.], 2004. Disponível online em *www.tucs.fi/publications/attachment.php?fname=TR652.pdf*.

[4] LATHI, B. P. *Modern Digital and Analog Communications Systems*. [S.l.]: Oxford University Press, 1998.

[5] TUTTLEBEE, W. *Software Defined Radio: Enabling Technologies*. [S.l.]: Wiley, 2002.

[6] PAQUELET, S.; MOY, C.; AUBERT, L.-M. Rf front-end considerations for sdr ultra-wideband communications systems. *RF Design Magazine - Wireless Communication and Radio Frequency Circuit Design for RF Engineers*, v. 1, July 2004, On page(s): 44 - 51, 2004.

[7] ALLEN, P. E.; HOLBERG, D. R. *CMOS Analog Circuit Design*. [S.l.]: Oxford University Press, 2002.

[8] LI, Z.; MA, Q.; QI, R. Design of a programmable digital down-converter structure. *Electrical and Computer Engineering, 2003. IEEE CCECE 2003. Canadian Conference on*, v. 1, May 2003 Page(s): 535 - 538, 2003.

[9] SJöSTRöM, U.; CARLSSON, M.; HöRLIN, M. Design and implementation of a digital downconverter chip. In: *The European Association for Signal and Image Processing*. [S.l.: s.n.], 1996.

[10] SHENOI, B. A. *Introduction to Digital Signal Processing and Filter Design*. [S.l.]: Wiley, 2005.

[11] GIRAU, G. et al. Fpga digital down converter ip for sdr terminals. *Signals, Systems and Computers, 2002. Conference Record of the Thirty-Sixth Asilomar Conference on*, v. 2, November 2002, On page(s): 1010- 1014, 2002.

[12] GENTILE, K. Digital upconverter ic tames complex modulation. *Microwaves and RF*, v. 1, August 2000, 2000.

[13] WAKERLY, J. F. *Digital Design - Principles and Practices*. [S.l.]: Prentice Hall, 2000.

[14] ZELENOVSKY, R.; MENDONçA, A. *Eletrônica Digital*. [S.l.]: MZ Editora, 2004.

[15] AHLQUIST, G. C.; RICE, M.; NELSON, B. Error control coding in software radios: an fpga approach. *Personal Communications, IEEE*, v. 6, August 1999, On page(s): 35 - 39, 1999.

[16] MARWEDEL, P. *Embedded System Design*. [S.l.]: Springer, 2006.

[17] JERRAYA, A. A. Long term trends for embedded system design. *Digital System Design, 2004. DSD 2004. Euromicro Symposium on*, v. 1, On page(s): 20- 26, 2004.

[18] HAYKIN, S. S. *Communication Systems*. [S.l.]: Wiley, 2000.

ANEXOS

I. DESCRIÇÃO DO CONTEÚDO DO CD

O conteúdo do CD anexo a esta monografia está dividido em três diretórios, da seguinte maneira:

- Diretório DDC: nele estão contidos os códigos do DDC escritos nas linguagens de descrição de hardware Verilog e VHDL, bem como os vetores de entrada para as simulações funcionais e temporais, descritas no capítulo 4.
- Diretório MATLAB: nele estão contidos os modelos MATLAB utilizados para a simulação do DDC, descritas no capítulo 4.
- Diretório Monografia: nele estão contidos os arquivos do presente manuscrito em formato digital.