Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia Eletrônica

# Implementação em FPGA do transceiver FMCOMMS3 para o protocolo de comunicação DVBS2X

Autor: Jhonathan Nicolas Moreira Silva
Orientador: Dr. Daniel M. Muñoz Arboleda (FGA/Unb)

Brasília, DF
2020

Jhonathan Nicolas Moreira Silva

# Implementação em FPGA do transceiver FMCOMMS3 para o protocolo de comunicação DVBS2X

Monografia submetida ao curso de graduação em Engenharia Eletrônica da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Eletrônica.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Dr. Daniel M. Muñoz Arboleda (FGA/Unb)

Brasília, DF

2020

Jhonathan Nicolas Moreira Silva

Implementação em FPGA do transceiver FMCOMMS3 para o protocolo de comunicação DVBS2X / Jhonathan Nicolas Moreira Silva. – Brasília, DF, 2020-

77 p. : il. (algumas color.) ; 30 cm.

Orientador: Dr. Daniel M. Muñoz Arboleda (FGA/Unb)

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2020.

I. Dr. Daniel M. Muñoz Arboleda (FGA/Unb) . II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Implementação em FPGA do transceiver FMCOMMS3 para o protocolo de comunicação DVBS2X

CDU 02:141:005.6

Jhonathan Nicolas Moreira Silva

# Implementação em FPGA do transceiver FMCOMMS3 para o protocolo de comunicação DVBS2X

Monografia submetida ao curso de graduação em Engenharia Eletrônica da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Eletrônica.

Trabalho aprovado. Brasília, DF, 17 de dezembro de 2020 – Data da aprovação do trabalho:

---

**Dr. Daniel M. Muñoz Arboleda**
**(FGA/Unb)**
Orientador

---

**Prof. Dr. Leonardo Aguayo**
**(ENE/FT)**
Avaliador

---

**Prof. Dr. Guillermo Alvarez Bestard**
**(FGA/Unb)**
Avaliador

Brasília, DF
2020

*Este trabalho é dedicado a todas as pessoas que,*
*em algum lugar do mundo, se esforçam para transformar esse mundo em um lugar melhor.*

# Agradecimentos

Meus agradecimentos especiais à minha mãe, meus familiares, meus amigos e colegas e aos docentes da universidade de Brasília do campus Gama cuja a dedicação e o esforço foram essenciais e inspiradores.

A todos os funcionários, servidores e terceirizados do *campus* Gama da Universidade de Brasília que trabalham assiduamente para que todas as engrenagens necessárias para a realização da minha graduação e milhares de outros alunos funcionem, colando assim engenheiros e engenheiras altamente capacitados e humanizados, aptos a trabalharem nos mais diversos nichos da tecnologia do Brasil e do mundo.

De forma mais contundente agradeço ao meu orientador e professor Dr. Daniel Munõz que de forma constante me auxiliou em diversas situações ao longo da minha graduação, e que verdadeiramente exerce o papel de docente em seu mais alto nível.

Agradeço a banca deste trabalho, individualmente ao professor Dr. Leonardo Aguayo e ao professor Dr. Guillermo A. Bestard que se disponibilizaram seu tempo para compor a banca avaliadora, e permitir assim a realização deste trabalho final de curso.

Por fim, agradeço a toda comunidade da Universidade de Brasília que faz com que essa instituição de renome internacional proporcione educação superior de altíssimo nível de forma pública e gratuita.

# Resumo

Esse trabalho apresenta um estudo detalhado da implementação em FPGAs (*Field Programmable Gate Arrays*) de um sistema operacional embarcado via PetaLinux desenvolvido para ser o *front-end* de um sistema de recepção para o protocolo de comunicação DVB-S2X integrado com o transceiver FMCOMMS3 em comparação com um DDC *(Digital Down Converter)* totalmente desenvolvido no hardware programável do FPGA. É também desenvolvido nesse documento a integração do DDC com o CFC (*Cource Frequency Correction*) conforme normatizado no padrão DVB-S2X. Por fim são feitas ponderações do desempenho entre o sistema *front-end* em SDR *(Software Defined Radio)* implementado através do PetaLinux e o circuito do DDC implementado em hardware e descrito em VHDL. Um dos componentes inerentes a qualquer padrão é o módulo de recepção do sinal, especificamente o DDC, para qual existem várias soluções implementadas tanto em hardware quanto para aplicações em SDR. A proposta central deste trabalho é desenvolver um sistema de recepção para a placa ZCU104 da empresa Xilinx integrada fisicamente com o transceiver FMCOMMS3 da empresa Analog Devices por meio do PetaLinux, em paralelo a um Digital Down converter em hardware integrado com o CFC. No caso do PetaLinux visa-se desenvolver um sistema operacional ainda não disponibilizado oficialmente pela empresa Analog Devices para o kit de desenvolvimento ZCU104. Observa-se que a empresa, até o presente momento, disponibiliza oficialmente o PetaLinux para a placa ZCU102, contudo, essa segunda é uma placa com um custo maior que a ZCU104 que possui entradas FMC que permitiria a mesma implementação a um custo menor. Em relação ao DDC visa-se explorar de forma intrínseca as especificidades do padrão DVB-S2X, como largura de banda, *Roll-off*, taxa de bits e entre outras métricas se refletindo em um baixo custo de implementação explorando a quantidade de coeficientes dos filtros, formas de ondas esperadas e resolução do oscilador acarretando em uma redução efetiva do processamento ao longo de toda cadeia da down conversão sem perda de generalidade. A generalidade irá se manter pois todos os coeficientes, largura de banda, excursão de frequência são configurados diretamente na descrição do hardware ou usando scripts desenvolvidos como modelos de referência em software que geram os trechos da descrição em hardware. Adicionalmente todos os circuitos foram sintetizados e implementados no FPGA para uma taxa de amostragem de 100 MSPS e caracterizados em termos de precisão, recursos de hardware, latência, *throughput* e consumo de energia e então comparados com as soluções existentes. Os resultados experimentais mostram que o circuito projetado e integrado com o CFC é viável e eficiente para a recepção e sincronização do sinal conforme a norma DVB-S2X, além de consumir menos recursos e tempo de execução do que os encontrados na literatura.

**Key-words**: DDC. FPGA. DVB-S2X. SDR. FPGA. PetaLinux. DSP. Embedded systems. Telecommunications.

# Abstract

This paper presents a detailed study of the implementation in FPGAs ( textit Field Programmable Gate Arrays) of an embedded operating system via PetaLinux developed to be the  textit front-end of a reception system for the communication protocol DVB- S2X integrated with the FMCOMMS3 transceiver compared to a DDC  textit (Digital Down Converter) fully developed on the FPGA's programmable hardware. It is also developed in this document the integration of the DDC with the CFC ( textit Cource Frequency Correction) as standardized in the DVB-S2X standard. Finally, performance considerations are made between the  textit front-end system in SDR  textit (Software Defined Radio) implemented through PetaLinux and the DDC circuit implemented in hardware and described in VHDL. One of the components inherent to any standard is the signal reception module, specifically the DDC, for which there are several solutions implemented both in hardware and for SDR applications. The main proposal of this work is to develop a reception system for the ZCU104 board from the company Xilinx physically integrated with the FMCOMMS3 transceiver from the company Analog devices through PetaLinux, in parallel a Digital Down converter in hardware integrated with the CFC. In the case of PetaLinux, the aim is to develop an operating system not yet officially made available by the company Analog devices for the ZCU104 development kit. It is observed that the company, until now, officially makes PetaLinux available for the ZCU102 board, however, this second one is a board with a higher cost than the ZCU104 that has FMC inputs that would allow the same implementation at a lower cost. Regarding DDC, the aim is to explore intrinsically the specifics of the DVB-S2X standard, such as bandwidth,  textit Roll-off, bit rate and other metrics, reflecting a low cost of implementation exploring the amount filter taps, expected waveforms and oscillator resolution resulting in an effective reduction in processing along the entire down conversion chain without loss of generality. The generality will remain as all the coefficients, bandwidth, frequency excursion are configured directly in the hardware description or using scripts developed as reference models in software that generate the parts of the hardware description. Additionally, all circuits were synthesized and implemented in the FPGA for a sampling rate of 100 MSPS and characterized in terms of precision, hardware resources, latency,  textit throughput and energy consumption and then compared with existing solutions. The experimental results show that the circuit designed and integrated with the CFC is feasible and efficient for the reception and synchronization of the signal according to the DVB-S2X standard, in addition to consuming less resources and execution time than those found in the literature.

**Key-words**: Digital Down Converter. FPGA. DVB-S2X. Digital Signal Processing

# Lista de ilustrações

# Lista de tabelas

# Lista de abreviaturas e siglas

| | |
|---|---|
| ADC | *Conversor Analógico Digital* |
| AWGN | *Ruído Gaussiano Branco Aditivo* |
| CIC | *Cascaded Integrator-Comb* |
| CFC | *Corretor grosso de frequência.* |
| CLB | *Bloco lógico de conexão* |
| CLK | *Clock mestre de um determinado bloco.* |
| FIFO | *algoritmo de fila simples onde o primeiro dado a entrar é o primeiro a sair.* |
| FFC | *Corretor fino de frequência.* |
| FMC | *Tipo de slot de entrada e saída ou tipo de barramento.* |
| DDC | *Digital Down Converter* |
| DDS | *Sintetizador Digital Direto* |
| DRAM | *Memória Dinâmica de Acesso Aleatório* |
| DSP | *Processamento de sinal digital* |
| DVB-S2 | *Digital Video Broadcasting - Satellite - Second Generation* |
| DVB-S2X | Extensão do DVB-S2 |
| en | *sinal de ativação de um determinado bloco* |
| FIFO | *Primeira entrada, primeira saída* |
| FIR | *Resposta ao Impulso Finito* |
| FMCOMMS3 | *Transceptor da empresa analog devices com o chip AD9361* |
| FPGA | *Matriz de portas programáveis em campo* |
| I | *Parte imaginária do sinal.* |
| IEEE | Instituto de Engenheiros Eletricistas e Eletrônicos |
| IF | *Frequência intermediária.* |

| | |
|---|---|
| IP | *Propriedade Intelectual.* |
| LUT | *Tabela de consulta.* |
| RAM | *Memória de leitura e escrita.* |
| ROM | *Memória apenas de leitura.* |
| RTL | *Troca de informações a nível de registradores.* |
| Rx | *parte do receptor.* |
| RRC | *Raiz de cosseno levantado.* |
| SDR | *Rádio Definido por Software.* |
| SNR | *Relação Sinal Ruído.* |
| Tx | *parte do transmissor.* |
| OS | *Sistema Operacional* |
| PetaLinux | *Série de ferramentas da Xilinx para o desenvolvimento do sistema operacional Linux* |
| PL | *Parte programável em hardware do FPGA.* |
| PS | *Parte programável em software do FPGA.* |
| QPSK | *Modulação por deslocamento de fase.* |
| Q | *Parte real do sinal.* |
| ULA | Unidade Lógica Aritmética |
| VFD | *Atraso Fracional Variável* |
| VHDL | *Linguagem de descrição de hardware de circuito integrado de velocidade muito alta* |
| ZIF | *Frequência intermediária zero* |
| FSBL | *Arquivo de Suporte da Placa* |
| MSE | *Erro Quadrático Médio.* |
| MSPS | *Mega amostras por segundo* |
| ILA | *Analisador lógico integrado* |
| ZCU104 | *Kit de desenviolamento ZCU104 da Xilinx* |

# Lista de símbolos

| | |
|---|---|
| $\Delta f$ | Resolução da frequência |
| $I$ | Em fase |
| $Q$ | Em Quadratura |
| $lo$ | Oscilador local |
| $K$ | Palavra de K bits para representar os múltiplos da frequência mínima. |
| $e$ | Exponencial ou número de euler. |
| $Re$ | Parte real de um número complexo. |
| $Im$ | Parte imaginaria de um número complexo. |
| $cos$ | função trigonométrica cosseno. |
| $sin$ | função trigonométrica seno. |
| $t$ | representação do tempo ou amostras na equação. |
| $IF$ | Frequência intermediária. |
| $LO$ | Frequência do oscilador local. |
| $RF$ | Frequência de radiofrequência. |
| $BW$ | Banda de frequência do sinal. |
| $mixer$ | mixer ou misturador. |
| $SNR$ | Relação sinal ruído. |
| $log$ | função matemática logaritmo. |
| $Z$ | Variável complexa amostrada. |
| $MSE$ | Erro quadrático médio |

# Sumário

# 1 Introdução

## 1.1 Receptores no Processo da Comunicação Satelital

Ao longo dos anos a comunicação via satélite *(satcom)* possibilitou a distribuição da informação da informação em nível global. Dentre os principais progressos decorrentes do uso da comunicação via satélite, pode-se destacar a telecomunicação e suas aplicações como: Sinais televisivos, internet, GPS *(Global Positioning System)* e redes móveis como: 3G, 4G e 5G. Com um avanço rápido, tornou-se necessário a implementação de protocolos ou também conhecidos como padrões de comunicação para que as demandas conseguissem ser atendidas globalmente, em paralelo com outros sinais e além de operar com uma alta performance (convenientes de cada protocolo). Um dos padrões mais utilizados no mundo é o padrão DVB-S2 que possui uma extensão, o DVB-S2X. [1].

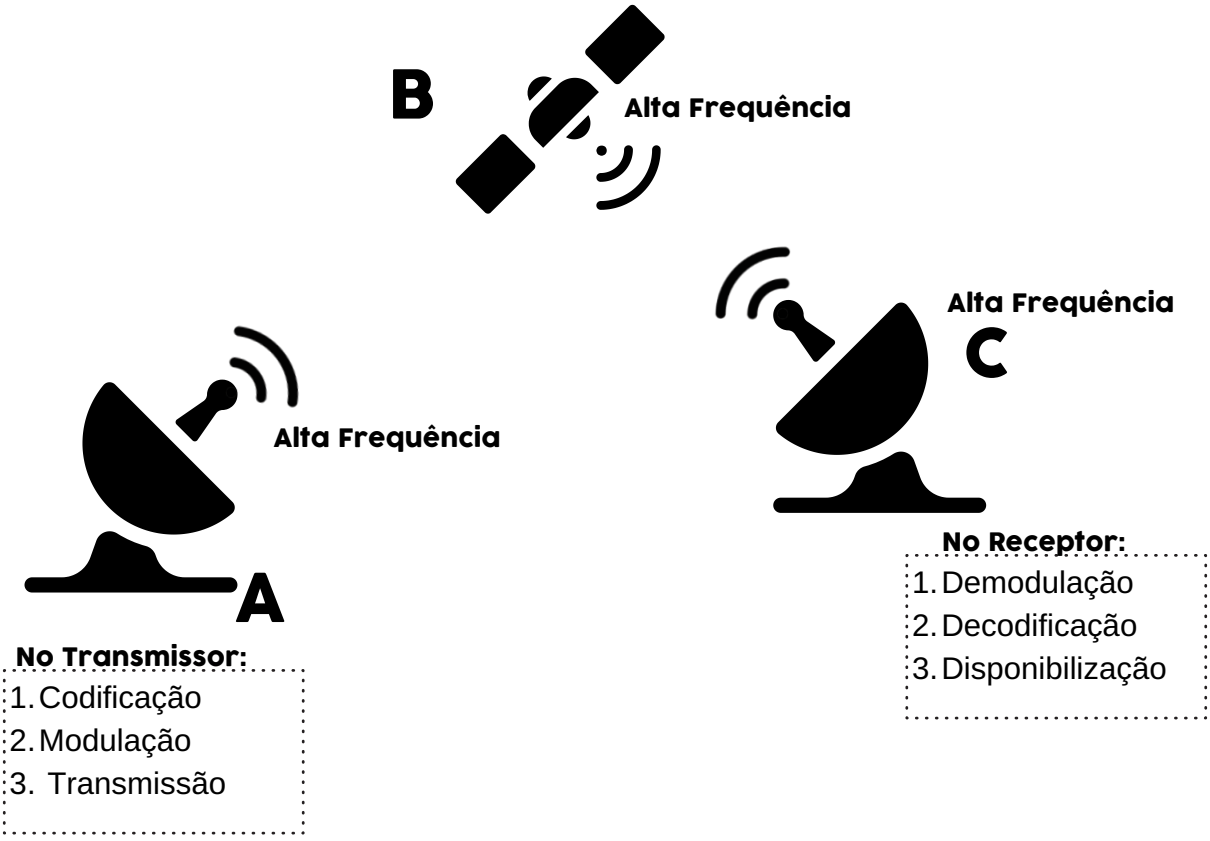



Figura 1 – Síntese da comunicação satelital abordada nesse trabalho.

A figura 1 mostra uma sintetização da comunicação satelital estudada nesse trabalho, salienta-se que na maioria dos protocolos são desenvolvidos transceptores, dispositivos que podem enviar e receber a informação, mas que para o desenvolvimento do transceptor

a cadeia do transmissor e receptor são distintas, o que é retratado na figura. Observa-se que na posição **A** a informação que se quer enviar é codificada e modulada dentro dos requisitos de cada protocolo, em especial no caso do DVB-S2X destaca-se a possibilidade de diversas modulações como por exemplo: QPSK, 8PSK, 16APSK e 32 APSK [2], em seguida é feita o processo de up conversão, onde o sinal que até então centrado em frequência zero é transladado para o que se chama de frequência intermediária, espera-se em muitos casos que essa frequência intermediária (IF) seja igual a frequência do DAC (Conversor Digital para analógico).

O Terceiro passo é a "transmissão"que entre vários processos dependentes exclusivamente de cada padrão pode-se resumir em submeter o sinal na frequência IF para uma frequência alta (na escala de GHz) e então enviar os dados para o satélite. Em **B** o satélite recebe o sinal e conforme suas especificidades retransmite o sinal para o receptor. Quando o sinal chega em **C** existem dois passos essenciais para qualquer protocolo de comunicação são eles: a demodulação e a decodificação. É na demodulação que o DDC entra como ferramenta essencial, ao chegar no receptor o sinal em alta frequência é transladado para frequência intermediária ao ser submetido ao ADC (Conversor Analógio Digital). De acordo com [2] para o protocolo DVB-S2X é necessário um bloco normatizado como *Signal Processing* que será responsável por tratar o sinal permitindo a sua decodificação.

Ao sair do conversor A/D é necessário colocar o sinal novamente em frequência zero no caso do DVB-S2X os três blocos principais por essa função são: DDC, CFC, FFC (Fine Frequency correction). O CFC é o que vai se comunicar diretamente com o DDC alterando dinamicamente a frequência que o sinal deve ser transladado para que os erros de frequência sejam minimizados, no caso do DVB-S2X não deve-se permitir uma frequência residual maior que 100 kHz [2].

## 1.2 Contextualização

O DVB-S2 é o padrão mais aceito e amplamente difundido no mercado de comunicação satelital. A extensão da geração DVB-S2 chamada DVB-S2X foi desenvolvida para satisfazer a confiabilidade dos serviços de comunicação via satélite de banda larga eficazes sobre condições hostis. Comparado ao DVB-S2 o DVB-S2X pode possui umas eficiência de até 51% a mais [1].

O DDC é o componente *front-end* do receptor DVB-S2X conforme é visto na figura 2. Este componente operando com o CFC são responsáveis por reduzir a frequência IF (frequência intermediária) a um residual de até no máximo 100 kHz mantendo todas as informações [3]. Salienta-se que o CFC aqui citado foi desenvolvido por um outro integrante do grupo [4] normatizado conforme as normas do DVB-S2X. O DDC deve ajustar os osciladores internos conforme indicado pelo módulo corretor de frequência como evidenciado

na figura 2, permitindo com que os erros frequências sejam reduzidos. *Field Programmable Gate Arrays* (FPGAs) podem ser uma solução viável para implementar DDCs, explorando o paralelismo intrínseco afim de alcançar um alto desempenho computacional [5].

No entanto, para atender a requisitos específicos da aplicação, é importante projetar uma arquitetura de hardware para o DDC parametrizável em termos de largura de banda, taxa de amostragem e largura de palavra. Também é essencial analisar como esses parâmetros afetam a ocupação de recursos, consumo de energia, latência e taxa de transferência. Os DDCs geralmente são implementados usando misturadores de frequência (*mixers*) e combinações de filtros de reamostragem, como o CIC (*Cascade Integrator Comb*), VFD (*Variable fractional Delay*), FIR ( *Finite Impulse Response*) e filtros polifásicos.

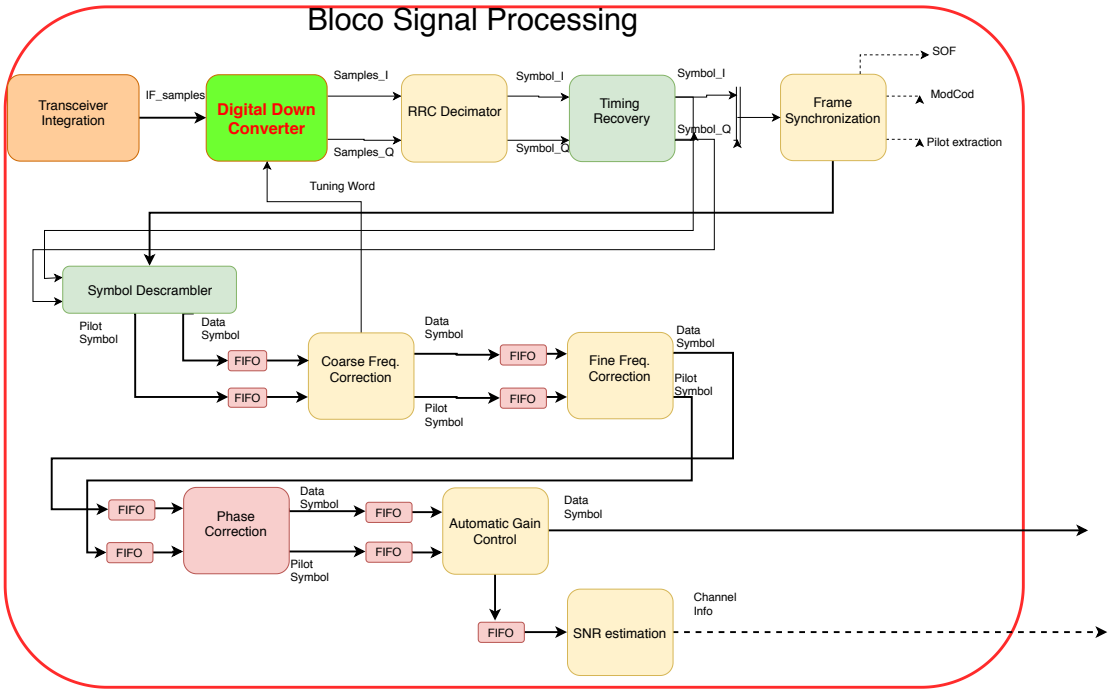



Figura 2 – Bloco *Signal Processing* do padrão DVB-S2X, evidenciando o papel do DDC.

Uma importante questão que se levanta ao pensar na implementação física da figura 2 é o fato de como vai funcionar o *transceiver* e sua integração física com o PL do FPGA. Boa parte dos transceiver disponíveis no mercado para integração com FPGAs já possuem um circuito de down conversão implementados via SDR, então abre-se uma opção de fazer com que o CFC se comunique diretamente com o sistema operacional (OS) desse SDR e, este por sua vez faça os ajustes na frequência. Nesse trabalho foi usado o *transceiver* FMCOMMS3 da empresa Analog Devices, juntamente com a placa de desenvolvimento ZCU104 da empresa Xilinx. Um dos problemas dessa integração é o fato da Analog Devices não desenvolver um sistema operacional para operar no ZCU104, o kit de desenvolvimento mais semelhante com sistema oficialmente distribuído pela Analog Devices é o ZCU102. Esse é um kit que atualmente no mercado custa 51,9% a mais que

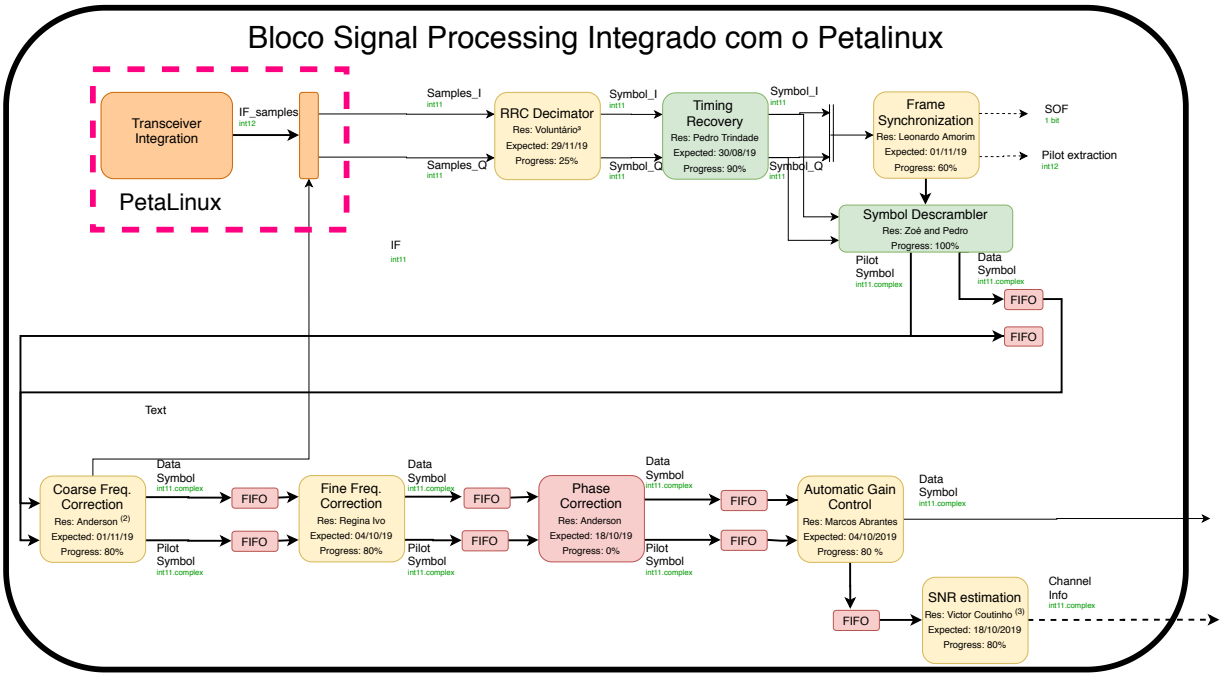



Figura 3 – Bloco *Signal Processing* do padrão DVB-S2X, evidenciando o papel do DDC utilizando o PetaLinux para a down conversão.

o ZCU104. Isto porque o ZCU102 possui recursos não disponíveis no ZCU104, contudo, esses não são necessários para implementar o transceptor no kit uma vez que essa mesma placa possui uma entrada FMC necessária para a integração com o transceiver. Diante desse cenário foi desenvolvido um sistema operacional por meio do PetaLinux para que fosse possível a integração do FMCOMMS3 com a ZCU104 utilizando o DDC interno do transceiver FMCOMMS3.

A figura 3 mostra o bloco do *signal processing* integrado com um sistema operacional construído a partir do PetaLinux. Observa-se que da mesma forma que na figura 2 o CFC continua se comunicando com o bloco responsável pela modificação da down conversão (agora o PetaLinux).

## 1.3 Objetivos

### 1.3.1 Objetivos Gerais

Implementar um Digital Down Converter em FPGA utilizando a linguagem de descrição de hardware VHDL e que atenda as métricas do padrão DVB-S2X de forma que a quantidade de amostras, resolução da frequência e tamanho da palavra amostrada sejam parametrizáveis.

Desenvolver um sistema operacional através do PetaLinux que permita a integração do *transceiver* FMCOMMS3 com o a placa de desenvolvimento ZCU104 em uma

temporização que permita a implementação do protocolo DVB-S2X no PL do FPGA.

### 1.3.2 Objetivos específicos

Espera-se alcançar os seguintes objetivos específicos:

- Modelo de referência do DDC em Matlab que permita analisar o desempenho do DDC em função da ordem dos filtros, fator de decimação, resolução da frequência, número de amostras, largura de banda e taxa de erro de bits.

- Implementação em hardware e caracterização do DDC frente ao modelo de referência desenvolvido.

- Integração do DDC ao CFC onde espera-se que o DDC seja capaz de fazer com que o desvio de frequência não seja maior que 100 kHz.

- Desenvolvimento e instalação de drives para um sistema operacional Linux através do PetaLinux para operar no SoC da placa ZCU104.

## 1.4 Contribuições do Trabalho

Sabe-se que o DDC é um circuito bastante recorrente em projetos de transmissão de informação digital sem fio, nesse sentido existem diversas arquiteturas plausíveis para a sua implementação. Porém, quando se trata de uma implementação em hardware como é o caso em FPGAs e aplicada ao padrão DVB-S2X, com ênfase em baixo consumo de hardware as opções ficam muito limitadas. Além disso o DDC descrito nesse documento possui parâmetros configurações a nível de projeto como: Número de amostras do oscilador, número de coeficiente dos filtros FIR e os próprios coeficientes dos filtros. Além de parâmetros a nível de síntese como: O fator de roll-off, tamanho das palavras de saída de cada módulo e fatores de decimação.

Até onde o levantamento do estado da arte pôde alcançar não há contribuições na comunidade científica cuja a implementação do DDC possui uma arquitetura de hardware tão parametrizável para o padrão DVB-S2X.

Ao olhar a implementação em PetaLinux a empresa fabricante do transceiver não desenvolveu um sistema operacional para a placa ZCU104, nesse sentido este trabalho se torna importante ao evidenciar essa possibilidade: Uma implementação usando o kit de desenvolvimento ZCU104 invés do ZCU102, uma economia de 51,9% para o projeto[6].

## 1.5 Estrutura do Texto

Este trabalho está dividido em 5 capítulos. Na metodologia é descrito o passo a passo para cumprir cada um dos objetivos específicos propostos e como foi feita a modelagem e descrição de cada componente que compõe o Digital Down Converter, além da metodologia adotada para o desenvolvimento do OS através do PetaLinux.

A Fundamentação Teórica, capítulo que vem a seguir, apresenta os conhecimentos técnicos que precisam ser dominados para a execução do trabalho. O terceiro capítulo, Aspectos Metodológicos e Ferramentais, descreve em mais detalhes os itens apresentados nos Objetivos Específicos do trabalho, se tratando da explicação dos passos que serão realizados para o cumprimento do objetivo final do trabalho e, além disso, como as ferramentas apresentadas na Fundamentação Teórica serão utilizadas nesses passos.

No quarto capítulo, Arquitetura e Proposta e Implementação, apresenta a arquitetura que será utilizada no projeto e explica detalhadamente como os passos citados nos objetivos específicos e metodologia foram executados para a implementação em FPGA. e implementação do OS. O penúltimo capítulo deste documento mostra os resultados atingidos e discussões. Nele são explicados as limitações do DDC. Além de mostrar o desempenho do DDC integrado já em FPGA com CFC e o nível de sucesso alcançado.

Também nesse mesmo capítulo são explicados os resultados do sistema operacional desenvolvido via PetaLinux juntamente com o transceiver FMCOMMS3, são detalhados parâmetros importantes de desempenho que são variáveis que devem ser levadas em conta ao implementar o OS juntamente com o PL do FPGA.

Ainda nesse capítulo são feitas comparações de desempenho em relação ao DDC implementado em hardware e o DDC implementado por SDR via PetaLinux.

Por fim, a Conclusão faz uma revisão dos resultados e contribuições elucidando o resultado dos dados coletados no capítulo anterior.

Foi feito um artigo científico modelo IEEE que foi submetido a convenção do LASCAS 2020 em 14 de outubro de de 2019 em relação ao desenvolvimento do DDC. O artigo está contigo nos anexos deste documento.

## 1.6 Estado da Arte

Em [7], [8] e [9] ferramentas de síntese de alto nível foram usadas para implementar DDCs em FPGAs, permitindo o ajuste de parâmetros principais do projeto, todavia, consumindo um grande número de recursos de hardware e particularmente blocos de DSPs (*Digital Signal Processing*).

Muitos outros trabalhos implementaram DDCs em FPGAs para sistemas SDR [5],

[7], [9], [10], [11] e [12], porém, os autores não mencionam quais parâmetros da arquitetura que podem ser ajustados.

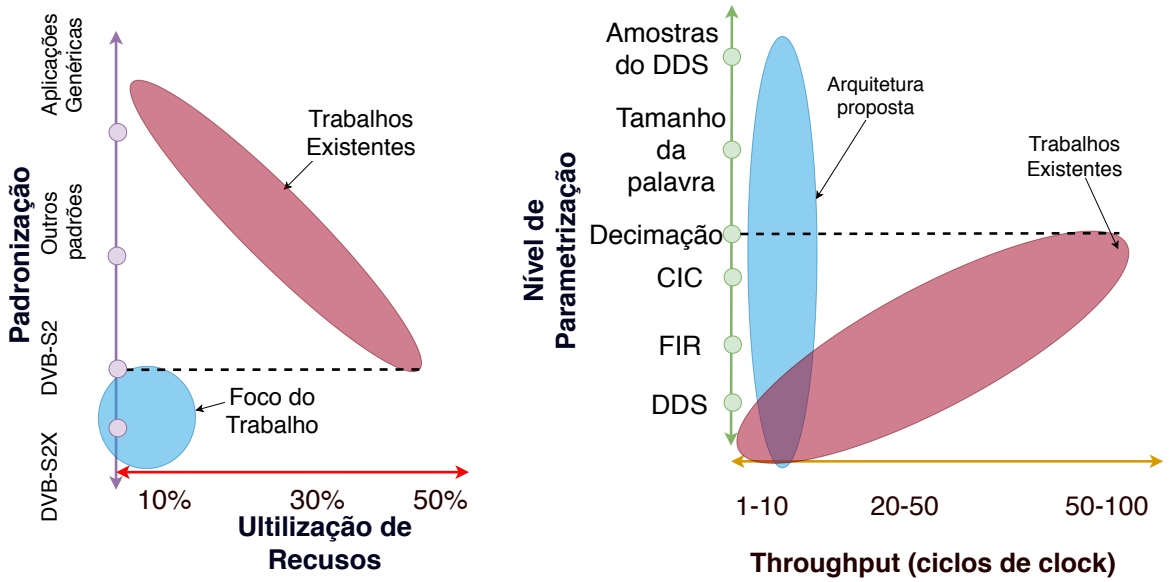
Os autores em [9] projetaram um DDC em um dispositivo *Xilinx Virtex100E* usando apenas filtros CIC economizando multiplicadores. Os autores em [9] também apresentam uma análise de como o tamanho da palavra e o número de estágios de filtragem afetam o consumo e a frequência operacional. Os autores em [5] apresentam um DDC para o padrão DVB-S2 para um dispositivo *Xilinx VirtexII Pro* que usa um DDS (*Direct Digital Synthesizer*), multiplicações complexas e 40-taps e 32-taps de filtros FIR cascateados. O DDC proposto processa sinais IF centrados em 70 MHz, roll-off de 0,35 e largura de banda de 36 MHz, produzindo na saída sinais de largura de banda de 11,75 MHz, consumindo cerca de 12277 LUTs (*Look-up tables*) e 15000 registradores.

Em [12] um DDC para aplicações de banda larga baseado nas combinações de vários filtros reamostradores foi projetado para um dispositivo *Xilinx Kintex 7* produzindo uma down conversão na saída com uma taxa de amostragem entre 1 kS/S e 225MS/s e consumindo 11646 LUTs, 18302 registradores e 106 blocos DSPs.

Recentemente os autores em [11] propuseram um modelo de implementação do DDC em FPGAs usando um processador CORDIC e filtros CIC e FIR cascateados, processando sinais de entrada de 70 MHz de largura de banda, produzindo na saída sinais de 137 KHz de largura de banda e consumindo aproximadamente 470 LUTs, 619 registradores slices e 16 blocos DSPs.

Os objetivos do presente trabalho são demostrados nas figuras 4a e 4b, evidenciando que não foram encontradas soluções já existentes que possuam um nível de parametrização tão alto para o padrão DVB-S2X.

A tabela 1 mostra a utilização dos recursos de implementações recentes no que diz respeito o DDC, é observado que nenhuma implementação é focada no padrão DVB-S2X.




(a) Padronização X Recursos.

(b) Parametrização X Desempenho.

Figura 4 – Foco do trabalho frente à outras implementações.

Tabela 1 – Utilização de recursos de algumas implementações mais recentes. (*): Valor não informado.

| **Autores** | **Componentes parametrizáveis** | **LUTs** | **Slices** | **DSPs** | **Frequência IF** | **Protocolo** |
|---|---|---|---|---|---|---|
| [5] | -* | 12277 | 15000 | -* | 70 MHz | DVB-S2 |
| [7] | -* | -* | 1350 | 24 | 440 MSPS | IEEE802.11n |
| [9] | Tamanho da palavra, nº taps e CIC | 558 | 694 | -* | 142 MHz | Genérico ou não informado |
| [12] | Tamanho da palavra, e taps | 11646 | 18302 | 106 | 1kS/s a 225MS\s | Genérico ou não informado |
| [11] | Tamanho da palavra e taps | 470 | 619 | 16 | 70 MHz | Genérico ou não informado |

# 2 Fundamentação Teórica

## 2.1 Teoria do DDC

Conforme é visto em [3] qualquer DDC tem duas premissas principais, separar os sinais ortogonais, transladar o sinal de interesse centrando na frequência zero e então atenuar todos os outros sinais permitindo assim a passagem apenas da banda de frequência que carrega a informação. Para que isso seja feito é necessário o uso de alguns componentes, dentre eles: um misturador, um DDS (*Direct Digital Synthesis*), e uma cadeia de filtros.

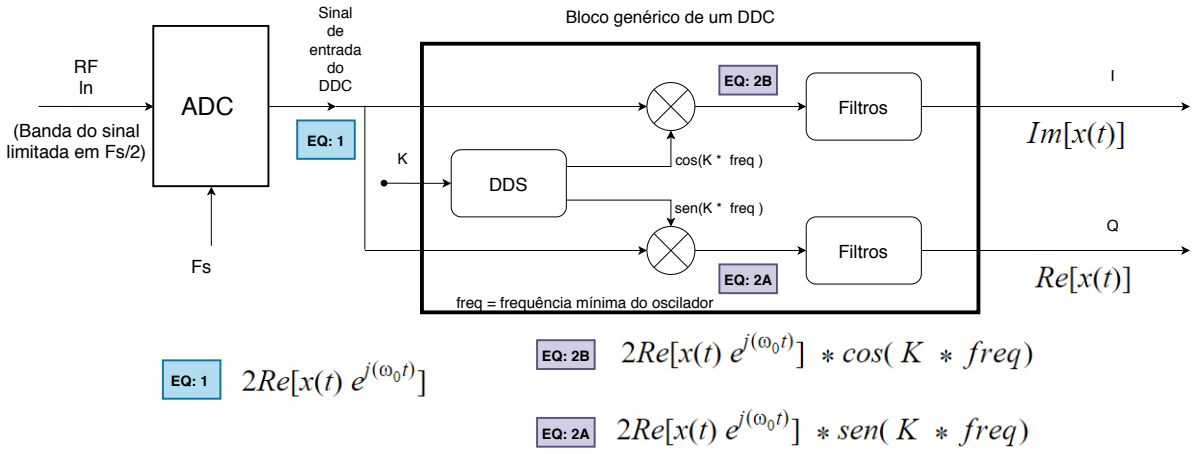



Figura 5 – Bloco genérico de um DDC dinâmico.

A figura 5 mostra de maneira simplificada o processo de transformação dos sinais dentro do DDC genérico operando dinamicamente, i.e a frequência do oscilador pode ser alterada em nível de execução. Primeiramente o sinal adquirido de um A/D operando em uma frequência $F_s$ é enviado para dois misturadores distintos. A segunda parcela de cada misturador é um sinal senoidal advindo do DDS (o oscilador), esse por sua vez é responsável por gerar dois sinais ortogonais entre si, a frequência desses sinais é um múltiplo da frequência mínima que esse oscilador pode gerar.

O múltiplo dessa frequência mínima é dada pela entrada $K$. A saída de cada misturador é o sinal da *EQ: 1* multiplicado por cada sinal do DDS resultando na *EQ: 2A* e *EQ: 2B*. Por fim esses sinais são submetidos a uma cadeia de filtros, normalmente filtros FIR (*Finite Impulse Response*). No final de toda cadeia espera-se observar $Im[x(t)]$ na saída $I$ e $Re[x(t)]$ na saída $Q$.

Na figura 6 é ilustrado o comportamento do DDC no domínio da frequência. Em **A** ao passa pelo ADC o sinal está centrado na frequência $IF$ definida pela equação 2.1 então

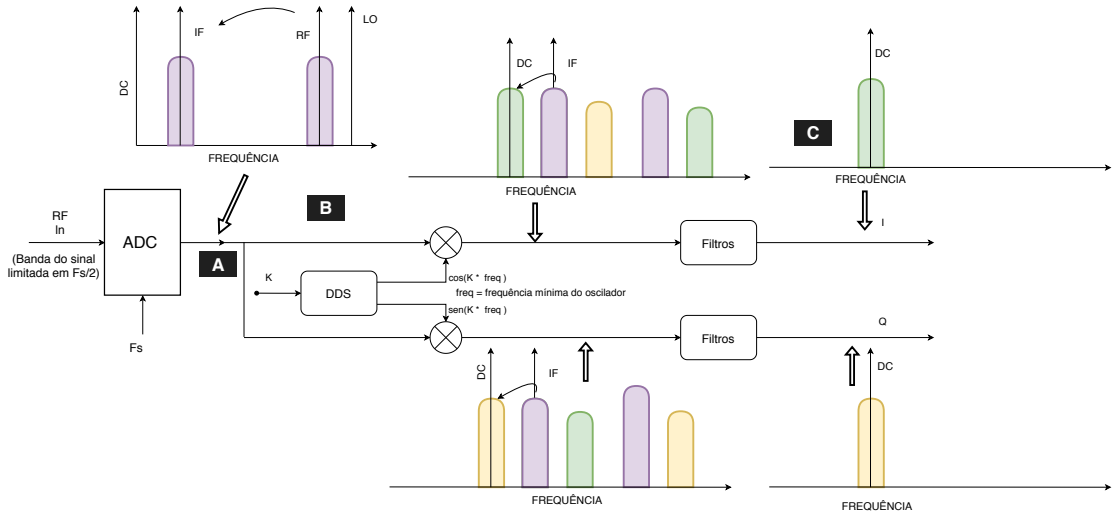



Figura 6 – Funcionamento básico do DDC. Todo sinal após o limite da largura de banda do sinal de interesse é ilustrado como ruído.

em **B** ao ser submetido pelo misturador de frequências o sinal o sinal sofre uma translação em frequência, como se espera que o DDC esteja operando próximo da frequência $IF$ o sinal não ortogonal a outra parcela da multiplicação é centrado em frequência zero. Ao fazer a mistura de frequência várias replicas do sinal original surgem em múltiplos da frequência $IF$ e $F_s$, esses por sua vez são considerados ruídos [13].

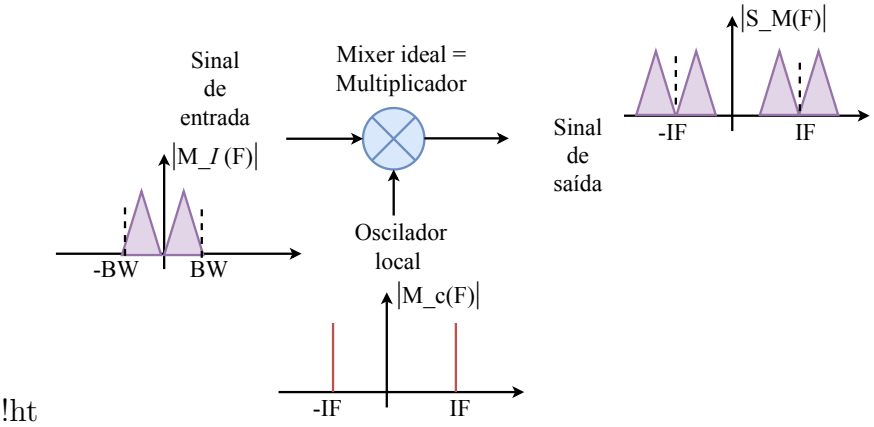
Devido a presença dos ruídos inerentes ao processo do misturador é necessário fazer uma filtragem do sinal. Nessa etapa vários filtros podem ser aplicados como: FIR, CIC, polifásicos, IR, desde que tenha um comportamento de um filtro passa baixas cuja a frequência de corte seja a banda do sinal que contem a informação útil.

$$f_{IF} = |f_{LO} - f_{RF}| \tag{2.1}$$

## 2.2 Modelagem do DDC

De acordo com [7], a arquitetura do DDC deve possuir pelo menos os seguintes blocos:

- Misturador: Para transladar a frequência.
- DDS: Para produzir o sinal necessário como parcela do misturador na translação.
- Filtros: Para atenuar os ruídos em alta frequência do sinal.
- Reamostragem: Para que a taxa do sinal seja igual ou pouco superior à largura de banda do sinal, caso contrário o sinal será processado a uma taxa mais alta que a necessária aumentando o consumo[3].




!ht

Figura 7 – Funcionamento do misturador. Um misturador ideal funciona como um multiplicador para a translação de frequência.

No caso do projeto do DDC para o para o DVB-S2X a reamostragem foi retirada já que essa por sua vez será feita pelo filtro RRC (*Root-raised cosine*) o bloco seguinte ao DDC na cadeia do *signal processing* [2].

### 2.2.1 Modelagem Matemática do Misturador

De acordo com [14] um misturador de frequências é um componente não linear que produz, a partir de duas frequências, uma nova frequência. Os misturadores são amplamente utilizados para transladar uma faixa de frequência para outra, um processo conhecido como translação de frequência [14] um receptor de down conversão no DDC é um misturador usado para transladar o sinal de uma frequência intermediária. Um misturador ideal é modelado como um multiplicador conforme é visto na figura 7. Conforme visto na equação 2.2 a multiplicação de um sinal que já está na frequência intermediária produz um sinal na banda base, e os outros sinais foram transladados para o dobro de $IF$.

Matematicamente dado a entrada:

$$e(t) = [E_0 + e_m(t)].cos(\omega_0 t)$$

A saída do oscilador é igual a :

$$e_{mix}(t) = \frac{E_0.[E_o l + e_m(t)]}{2}.cos((\omega_o l - \omega_0)t) + \frac{E_o l.[E_0 + e_m(t)]}{2}.cos((\omega_o l + \omega_0)t) \quad (2.2)$$

### 2.2.2 Modelagem Matemática do DDS

Para alimentar o misturador com um sinal na frequência $IF$ é necessário que haja um oscilador local no DDC, esse oscilador no mundo digital é conhecido como *Direct Digital Synthesizer* (DDS) ele é usado na eletrônica para gerar diversos sinais [15] no caso especifico do DDC os sinais de interesse são sempre senoidais , mais especificamente

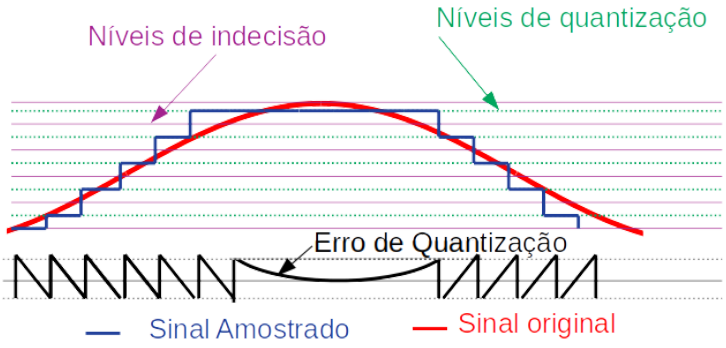



Figura 8 – Comportamento do erro de quantização em um sinal senoidal. *O erro de quantização é a diferença entre o sinal original e o sinal amostrado.*

um cosseno e um seno na frequência *IF*, existem diversos métodos que possibilitam a produção desses sinais que serão discutidos na seção de metodologia. Embora haja diversas formas de gerá-los existem prerrogativas que devem ser respeitadas independentemente da arquitetura. De acordo com [16] qualquer sinal amostrado deve respeitar a equação 2.3 conhecida na literatura como critério de Nyquist.

$$f_s \geq 2f_{max} \tag{2.3}$$

Sendo assim a frequência máxima que o DDS pode gerar deve é dada pela equação 2.4

$$f_{max} \geq f_{clk}/2 \tag{2.4}$$

Onde $f_{clk}$ é a frequência de clock do circuito digital.

Isso leva a concluir que o transceiver/transponder não pode gerar uma frequência intermediária maior que $f_{clk}/2$, caso contrário a demodulação não poderá ser feita sem perdas de informação.

Além da frequência é necessário se atentar para o ruído de quantização. De acordo com [17], a máxima relação sinal ruído de quantização é dada pela equação 2.5, onde $B$ é número de bits, salienta-se que a equação 2.5 é para o valor máximo do sinal ($Vmax$). Para um valor de $V$ diferente de $Vmax$ o SNR é definido pela equação 2.6.

$$SNR_{dB} = 6B \tag{2.5}$$

$$SNR_{dB} = 1,76 + 6,02n - 20\log\frac{V_{max}}{V} \tag{2.6}$$

$$Erro_{quantização} = |Sinal_{ref} - Sinal_{amostrado}| \tag{2.7}$$

É possível notar pela figura 8 o comportamento do erro de quantização em um sinal senoidal cuja a definição é dada pela equação 2.7. Com o conhecimento do erro produzido pelo

sinal senoidal amostrado deverá ser feita uma escolha sobre a frequência de amostragem e o tamanho da palavra binária no intuito de minimizar esses erros, respeitando as métricas de SNR definida pela norma DVB-S2X.

### 2.2.3 Modelagem Matemática do FIR

Um filtro de resposta finita (FIR) é um filtro que se estabelece em zero em valores finitos no tempo, então impulsiona a resposta, e essas características tornam o FIR a escolha ideal para aplicações sensíveis à fase, tais como, comunicações de dados, sismologia, filtros cruzados e entre outros. A figura 9 mostra a modelagem do filtro FIR onde os $C_1$, $C_2$, ..., $C_n$ representam os coeficientes do filtro e definem a sua função de transferência.

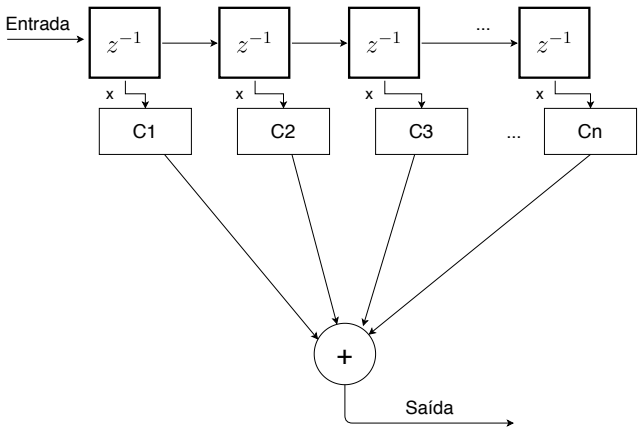



Figura 9 – Diagrama do Bloco FIR com n coeficientes.

Se tratando do filtro FIR para o digital down converter a topologia de interesse é um filtro FIR passa-baixas e devido a integração com o RRC o número de coeficientes pode ser reduzidos, uma vez que embora não seja o papel principal do RRC correrá uma atenuação nas frequências indesejadas, além da própria decimação se comportando como um filtro CIC

## 2.3 FPGAs, reconfiguração e aceleração de hardware

Um FPGA (Field Programmable Gate Array) é um sistema embarcado de hardware reconfigurável, entre outras palavras isso significa que esse hardware pode ser programado pelo usuário, usualmente por meio de uma linguagem de descrição de hardware, as mais comuns são VHDL e Verilog, para ter o comportamento e realizar as operações e tarefas projetadas [18].

No início de sua concepção nos anos 80, esses dispositivos eram de difícil reconfiguração, porém ao se perceber sua capacidade de reconfiguração e o leque de poder de processamento que um FPGA proporciona, muitos usuários começaram a investir nesse

sistema embarcado, por sua vez o acesso foi se tornando mais fácil, as linguagens de descrição mais amigáveis.

Seus desenvolvedores foram adicionando mais funcionalidades que tornavam a ferramenta ainda mais poderosa como geradores de clock, memórias randômicas de acesso remoto (DRAMs), memórias distribuídas, controladores e processadores de múltiplos núcleos no interior dos FPGAs. Além das principais desenvolvedoras construírem em massa IPs (*intellectual property*) de controladores, filtros, FIFOs, memórias, aplicações para processamento de imagem, vídeo, telecomunicações e entre outras diversas áreas de aplicação e engenharia.

Em geral, os FPGAs apresentam uma alta performance, pois pode realizar suas funções através de ULAs (Unidades Lógicas Aritméticas), como um microprocessador comum, ou de LUTs (*Look Up Tables*) que de maneira mais simplista pode ser interpretada como uma memória multiplexada [18], projetada a partir de uma memória de comprimento N e largura de 1 bit, um multiplexador de N entradas e alguns flip-flop é possível fazer uma função booleana totalmente genérica e reconfigurável de $log_2(N)$ entradas.

A figura 10 mostra como funciona uma Look-up table de 8 saídas com 3 entradas ou uma célula de SRAM. Look-Up tables podem ser conectadas por meio de CLBs (*conection lógic blocks*) e *switch blocks* (SB) para fazer funções ainda mais complexas. A figura 11

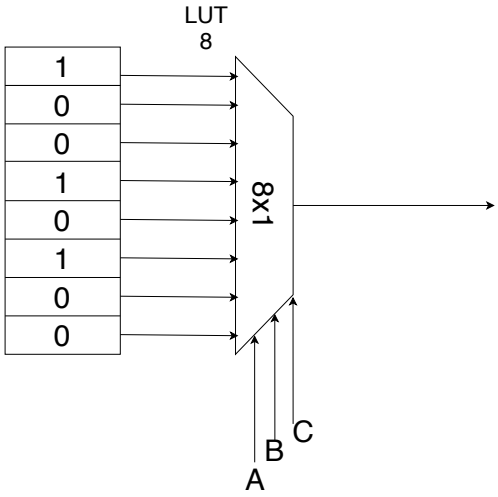



Figura 10 – LUT 8.

ilustra como funciona as interconexões dentro de um FPGA onde um bloco lógico é composto essencialmente de uma *look up table* e um flip-flop. Todos esses e outras diversas configurações de circuitos de um FPGA são responsáveis por dar a esse dispositivo uma das suas características mais poderosas, o paralelismo intrínseco, a figura 12 ilustra a diferença do processamento em um FPGA para, por exemplo, um microcontrolador, isso ocorre porque em um microcontrolador as instruções mapeadas primeiramente por software são armazenadas em uma pilha e, então o processador executa uma por vez.

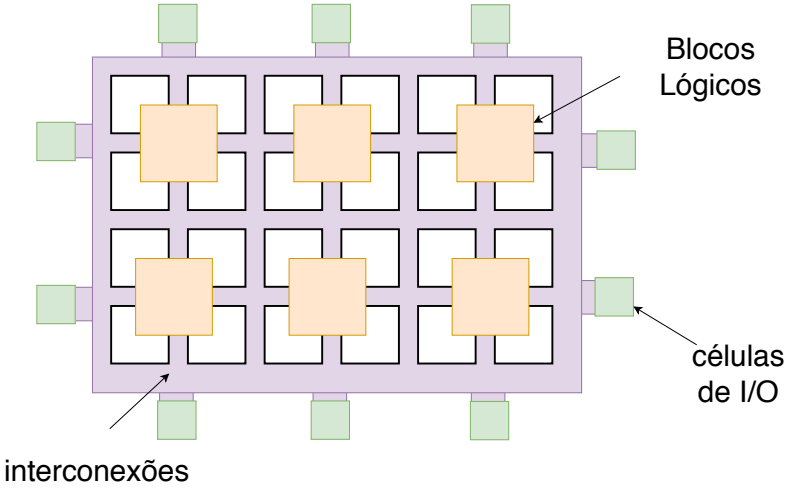



Figura 11 – Circuito simplificado das conexões internas de um FPGA.

Em um FPGA todos os processos são paralelos, pois são hardwares distintos que executam esses processos. Para o DDC isso é uma característica extremamente importante, as principais justificativas da utilização de um FPGA para o DDC são:

- I e Q são processos que devem ter respostas em paralelo, caso fossem sequenciais haveria uma perda muito alta de desempenho.

- É possível uma implementação em *pipeline*, ou seja cada bloco interno trabalha ao mesmo tempo, isso aumenta ainda mais o desempenho.

- É possível um controle em tempo real da frequência do DDS.

- As operações aritméticas(soma, subtração e multiplicação) além das operações lógicas são executadas com apenas 1 ciclo de clock.

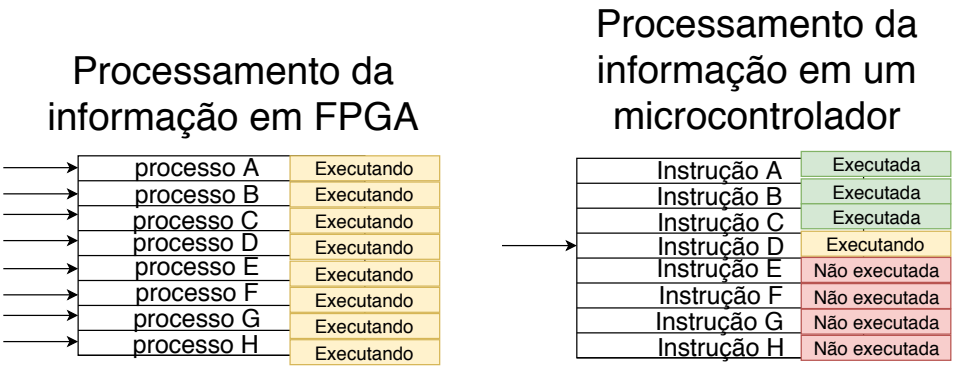



Figura 12 – Diferença do processamento em um FPGA e um microcontrolador.

## 2.4 Abordagem do PetaLinux em sistemas SDR

O pacote PetaLinux da Xilinx é uma série de ferramentas, boa parte delas em código aberto, para o desenvolvimento e implementação de sistemas operacionais, essencialmente com kernel Linux, nos FPGAs i.e SoC ou IPs dos chips da Xilinx [19].

Com o PetaLinux é possível desenvolver um sistema operacional específico para cada arquitetura da Xilinx. O PetaLinux não é, e não oferece atualmente uma distribuição Linux para se implementar no FPGA, todavia, disponibiliza uma série de ferramentas que viabilizam essa implementação como por exemplo: U-BOOT: Permite a configuração de inicialização dos sistemas operacionais, Bibliotecas e Aplicações: É possível instalar por meio de *Device Trees* aplicações e bibliotecas no OS tornando elas nativas e FSBL: Arquivo que contém essencialmente toda configuração da arquitetura do chip permitindo a criação de um kernel.

É notório que os sistemas desenvolvidos para telecomunicações operando por meio de software conhecidos como SDR são amplamente implementados no desenvolvimento de terminais e transceptores na maioria dos padrões de comunicação a nível global.

Isso ocorre porque os sistemas SDR consegue a partir da down conversão e up conversão trabalharem em uma taxa onde os processadores estão aptos a manipular os dados codificando e modulando os sinais.[12].

A aplicação do PetaLinux abre um leque de possibilidades para o desenvolvimento desses sistemas. Isso porque é possível utilizar sistemas operacionais para o tratamento dos sinais e, e concomitantemente, utilizar o PL do FPGA permitindo o a realização de operações em tempo real a uma taxa satisfatória.

A tabela 2 mostra a comparação em alguns requisitos de projeto da utilização do PS (*Programmable Software*) e PL (*Programmable Logic*) de um FPGA. Ressalta-se uma facilidade inerente do PL em operar em tempo real e explorar o paralelismo dos algoritmos uma vez que esses blocos são implementados diretamente em hardware.

Por outro lado ao usar um processador os algoritmos são implementados por instruções através de uma linguagem de programação, um facilitador para o projetista acelerando assim o desenvolvimento do projeto. Por fim, analisando em termos de integração ao se trabalhar com o PL do FPGA tem-se diversos desafios em termos de sincronismo, interface e recursos que são na maioria dos casos flexibilizados ao se trabalhar com o PS.

Tabela 2 – Diferença em algumas variáveis entre o PS e PL de um FPGA para o desenvolvimento de um projeto.

| **Comparação entre o processamento no PS e PL de um FPGA** | | |
|---|---|---|
| | **PS** | **PL** |
| **Comportamento Temporal** | Mesmo em versões bare-metal a temporização do sistema é comprometida. Ao se implementar um OS isso é ainda mais comprometido. | O sistema opera sempre em tempo real conforme sua taxa de latência e throughput. |
| **Paralelismo** | As instruções são realizadas de forma paralela apenas em sistemas de múltiplos núcleos quando corretamente projetado. | Cada bloco opera de forma independente do outro (Paralelismo intrínseco). |
| **Desenvolvimento do Projeto** | A utilização de linguagens de programação tornam o desenvolvimento mais rápido de ser implementado e testado. | O baixo nível em hardware torna o desenvolvimento mais lento. |
| **Integração** | Existe uma facilidade no desenvolvimento de software ao integrar funções e bibliotecas | A integração é lenta e precisa ser testada muitas vezes. |

# 3 Aspectos Metodológicos e Ferramentas

## 3.1 Arquitetura do DDC

Com as definições e a modelagem matemática apresentadas no capítulo anterior, a figura 13 mostra a arquitetura do DDC que será adotada.

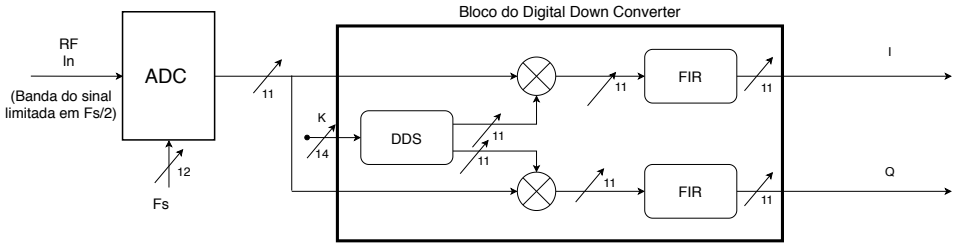



Figura 13 – Arquitetura do DDC abordada nesse trabalho.

O parâmetro $K$ é o $\Delta f$ que será enviado para o DDS. $I$ e $Q$ serão as saídas em fase e quadratura. O fluxo do sinal descreve-se pela seguinte forma:

- Primeiro o sinal é dividido em dois ramos para I e Q.
- Em seguida o ramo de I e Q são misturados pelo seno e cosseno respectivamente que são produzidos no DDS
- Após a translação em frequência o sinal passa pelo filtro FIR.

## 3.2 Ferramentas

Neste trabalho as seguintes ferramentas serão usadas utilizadas para o desenvolvimento do DDC:

- MATLAB: Nesse software foi feita a primeira concepção matemática dos modelos de cada componente do DDC, e se construiu um modelo de referência de cada componente do DDC (*Golden Model*).
- Python (*numpy, matplot, scipy*): Utlizando a linguagem de programação Python poderá ser feita a leitura dos arquivos de teste produzidos em simulação, além disso, também nessa plataforma será feita a comparação entre os modelos de referência e o modelo implementado em hardware.

- VIVADO: Ambiente para a descrição em hardware dos componentes, prosseguido da síntese, implementação em hardware, geração do bitstream e programação do chip (Zynq UltraScale+ XCZU7EV-2FFVC1156 MPSoC) do kit de desenvolvimento Xilinx ZCU104.

- PetaLinux: Série de ferramentas para o desenvolvimento do Linux embarcado que irá operar na ZCU104.

- Yocto Project: Um software colaborativo de código aberto da Linux Foundation, cujo objetivo é produzir ferramentas e processos que permitam a criação de distribuições Linux para software embarcado e IoT, independentes da arquitetura subjacente do hardware incorporado [20].

- GNURadio: Software para o estudo de caso do digital down converter para o envio e o receptação do sinal nos parâmetros DVB-S2X.

- Adalm Pluto: A placa SDR Adalm Pluto da empresa Analog Devices para enviar sinais RF em uma curta distância para o transceiver FMCOMMS3 afim de realizar simulações e testes de desempenho no canal da recepção do transceiver.

- FMCOMMS3: O transceiver que será integrado na placa ZCU104 a partir das portas FMC possibilitando a recepção dos sinais RF.

## 3.3 Metodologia para o desenvolvimento do DDC

A figura 14 ilustra a metodologia proposta para o desenvolvimento de cada componente do DDC individualmente.

A primeira parte é o levantamento do estado da arte do componente, e a identificação dos algoritmos desenvolvidos para a solução. Embora o DDC seja um circuito comumente utilizado no ambiente de telecomunicações observou-se uma carência para o padrão DVB-S2X, além de notado um alto uso de recursos dos DDCs já projetados em FPGA.

Após o estudo dos algoritmos esses foram modelados no MATLAB utilizando representação numérica em ponto flutuante do próprio ambiente, alguns desses algoritmos utilizam funções prontas da ferramenta, o que não se mostrou uma problemática, visto que muitas dessas funções seriam factíveis de implementação em hardware. Após a modelagem do algoritmo, foi feita sua validação em termos de *roll-off*, SNR, ganho e robustez. Após verificar que esses modelos estão de acordo com o padrão os mesmos foram adaptados para serem usados como modelos de referência (*golden models*) para a descrição dos componentes em VHDL.

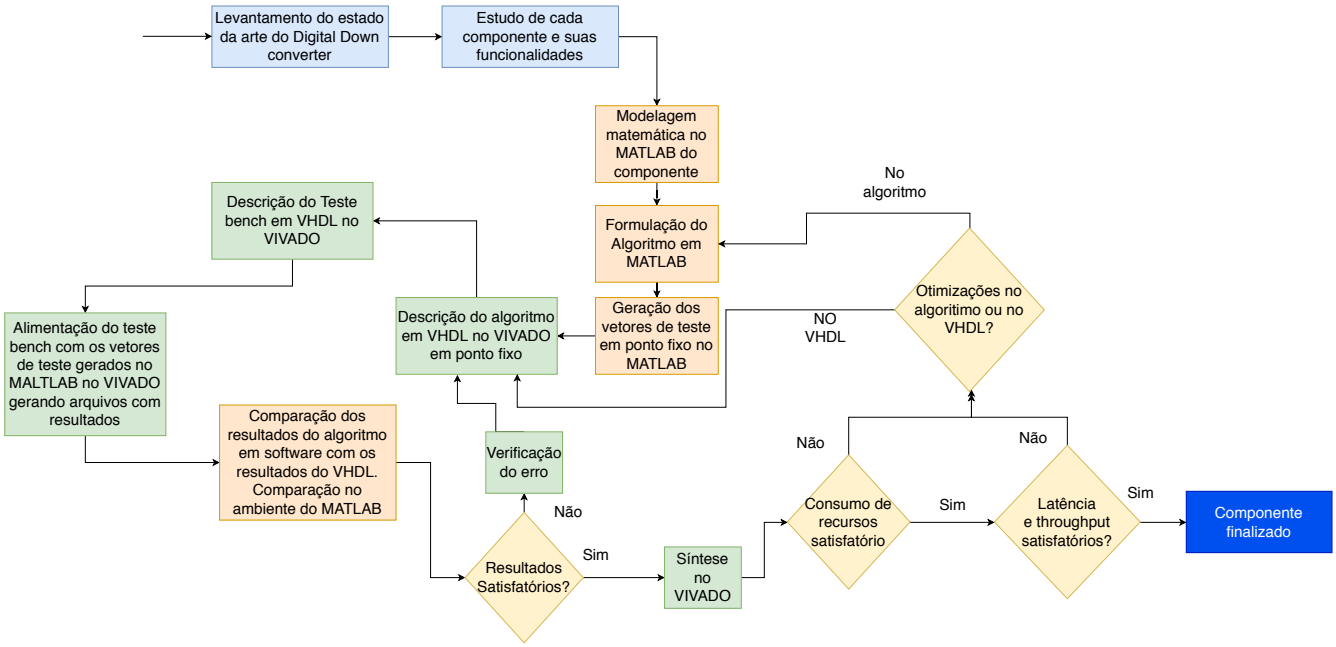



Figura 14 – Metodologia para o desenvolvimento dos componentes do Digital Down Converter.

Em particular, uma análise de precisão foi realizada para estimar os erros de quantização introduzidos pelo tamanho da palavra. Foram geradas entradas para esse *golden model* que seriam as mesmas entradas para o modelo do VHDL. A parte mais trabalhosa e desafiadora de qualquer descrição de hardware que tenha um *golden model* em software como é esse o caso, é certamente traduzir esse algoritmo para hardware utilizando a menor quantidade de recursos e com melhor latência e *throughput* possível.

Após descrição em VHDL dos módulos foram produzidos os respectivos testbenches para validar o funcionamento do algoritmo. Esse testbench é alimentado pelos vetores de teste gerados pelo *golden model* em MATLAB e, então as saídas desse testbench é armazenado em novos arquivos. Esses resultados são comparados por meio do MSE (*Mean Square Error*), onde o principal intuito é verificar se os resultados em ponto fixo feitas no VHDL para cada entrada estão de acordo com os resultados em ponto flutuante do *model golden*, caso o resultado não seja satisfatório é feita uma análise no código VHDL e corrigidas as imperfeições.

Após o processo de validação das arquiteturas projetadas é feita uma síntese lógica do circuito, onde é feita a análise dos recursos, da latência e throughput (a nível comportamental) do componente, caso os resultados não estejam satisfatórios são feitas otimizações ou no VHDL ou no algoritmo em MATLAB e toda cadeia se repete, até a finalização do componente em questão.

Após a finalização de todos os componentes do DDC, será feita a integração desses componentes e sua respectiva validação em termos de simulação comportamental.

## 3.4 Metodologia para os testes do DDC

Após finalizar o desenvolvimento dos módulos do DDC e sua respectiva integração é necessário respeitar um protocolo rígido para os testes em hardware, pois apenas esses podem garantir que houve a correta implementação do módulo.

A figura 15 ilustra a metodologia adotada para realizar os testes em hardware. Primeiramente em um ambiente de programação Python foram gerados as amostras com desvio de frequência e então carregadas para duas memórias ROMs através de um IP da Xilinx. Nota-se que as arquiteturas genéricas mostradas anteriormente para o DDC havia apenas a entrada de um sinal $(I + Q)$, porém, sabe-se que o transceiver FMCOMMS3 entrega os sinais necessariamente em fase e quadratura, i.e I e Q separados, por essa razão preferiu-se mudar a interface do DDC para duas entradas. Ao fazer a integração do DDC com o CFC que será explicada na seção 4 deste documento, a saída do DDC e do CFC foram armazenadas em memórias RAMs usando também o devido IP da Xilinx. Diante

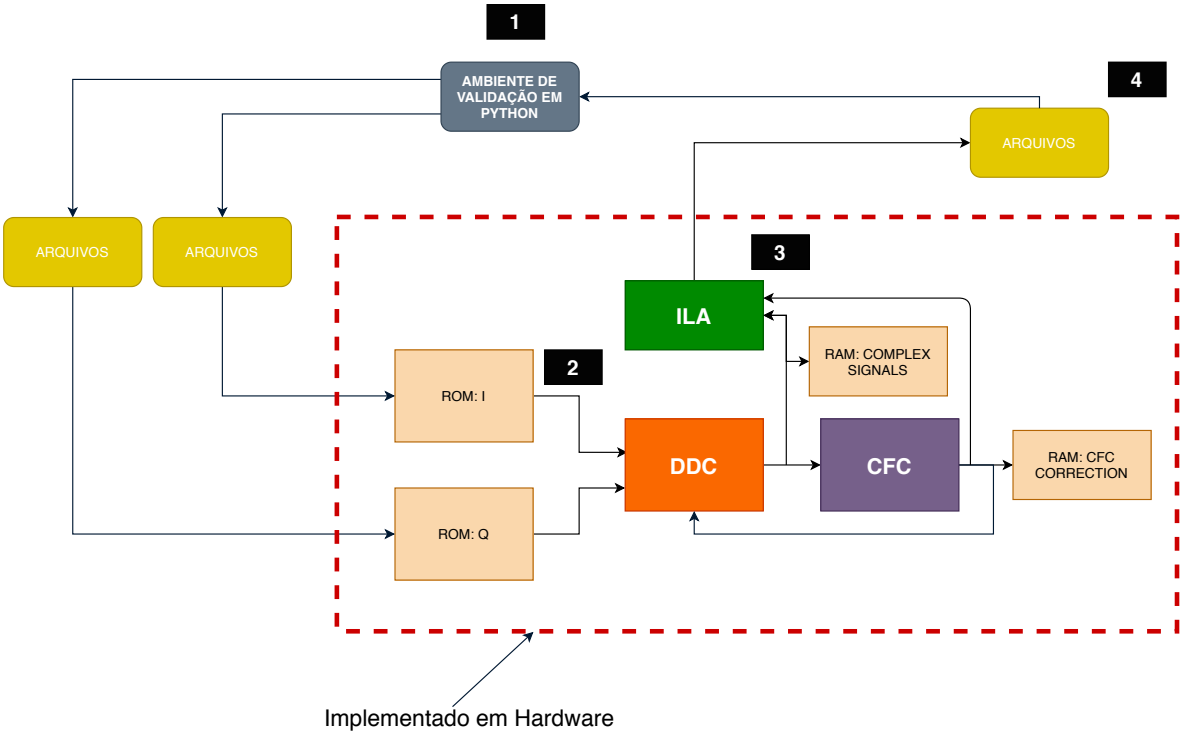



Figura 15 – Metodologia adotada para o teste em hardware do DDC.

disso foi usado mais um IP para o debug em hardware o ILA *(Integrated Logic Analyzer)* da Xilinx. Esse IP permite que o projetista consiga inserir pontas de prova em determinados caminhos de dados, i.e sinais no hardware [21]. Além de inserir pontas de prova o ILA também permite aferir valores em determinados sinais, o que não foi utilizado nesse teste uma vez que pode-se fazer uso dos *pushbuttons* do kit de desenvolvimento para o sinal de *reset* e início das operações, unicas entradas externas ao bloco de teste uma vez que I e Q já foram carregadas para memórias ROMs. O ILA permite a visualização dos sinais que

foram mapeados em tempo real, além disso é possível exportar essas informações para arquivos, (comumente de extensão .csv).

Por fim, esses arquivos são carregados para o ambiente de simulação em Python e os resultados são interpretados com seus respectivos gráficos e métricas necessárias.

## 3.5 Metodologia para o Desenvolvimento do OS usando Yocto Project e PetaLinux

Para a criação do sistema operacional que operará no SoC da ZCU104 foi necessário seguir diversos procedimentos que garantem o correto *build* e execução do OS. Muitos desses passos foram refinados ao longo de diversas tentativas, os passos aqui seguidos foram orientados tanto pela *Xilinx* quanto pela *Analog devices*, todavia, se tratando de um OS que a empresa Analog Devices não distribui, foi necessário acrescentar conhecimentos empíricos após sessões de teste. A figura 16 ilustra a metodologia abordada para a construção do OS.

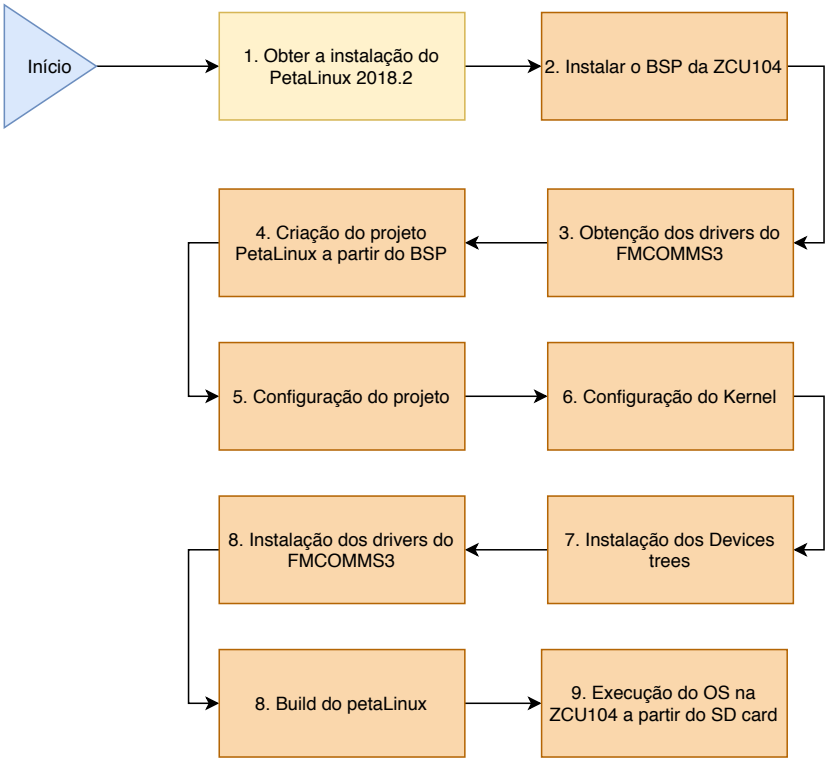



Figura 16 – Metodologia adotada para a criação do OS a partir do PetaLinux.

### 3.5.1 Obtenção do PetaLinux

Nota-se a importância de obter a versão que melhor se adapta ao sistema operacional da máquina que irá executá-lo. Ressalta-se também que o PetaLinux é uma adaptação

do *Yocto Project* um projeto amplamente utilizado para a criação de sistemas operacionais embarcados com kernel Linux.

### 3.5.2 Instalação do BSP

Uma das principais características que diferencia o PetaLinux do Yocto Project é sua capacidade de fazer pré configurações para cada dispositivo fornecido pela Xilinx. Isso é feito a partir de um arquivo chamado BSP (*Board Support Package*), um arquivo com todas as informações sobre o kit de desenvolvimento específico levando em consideração toda a arquitetura interna do SoC. Com isso é possível fazer com que o PetaLinux reconheça o contexto do *build* permitindo a utilização rápida de diversos recursos da placa por meio do device tree como:

- Utilização de portas HDMI.
- Utlização dos dispositivos de áudio.
- Interface de comunicação: I2C, SPI e AXI.
- Contexto de pinos como portas FMC (Utilizadas para integrar o transceiver) .

### 3.5.3 Obtenção dos Drives do FMCOMMS3

Um dos drivers que não tem no BSP original fornecido pela Xilinx é o drive do FMCOMMS3 que faz a interface de comunicação SPI com as portas FMCOMMS3, além de instalar importantes aplicações e funções para o controle do transceiver. Para isso foi utilizado os drivers fornecidos pela Analog devices em [22].

### 3.5.4 Instalação do BSP no PetaLinux

Ao obter o PetaLinux e o BSP da ZCU104 torna-se necessário criar um projeto para o PetaLinux que irá fazer um reconhecimento da arquitetura e dos recursos disponibilizados no kit de desenvolvimento da ZCU104.

### 3.5.5 Configuração do PetaLinux

Na configuração do PetaLinux é possível fazer configurações como: U-BOOT, Kernel, Ethernet, MAC address que são importantes para o funcionamento correto do OS integrado com o transceiver FMCOMMS3. Além disso diversas outras configurações podem ser feitas como: *Baud Rate* para comunicação serial, comunicação SPI que será usada para o transceiver FMCOMMS3. Também é importante a configuração correta dos arquivos raízes do OS como: *INITRAMFS* e *NFS*.

### 3.5.6 Configuração do Kernel

Nesse passo são feitas configurações importantes do kernel. Em um projeto genérico essa configuração não é mandatória pois o Kernel já vem configurado por padrão pelo arquivo BSP e nas configurações do PetaLinux. Todavia, como é de interesse instalar novos drives para o transceiver FMCOMMS3 é essencial fazer uma nova compilação do kernel.

### 3.5.7 Configuração do Device Tree

O device tree é essencialmente uma série de aplicações estruturada que formam a interface entre hardware e o OS. É importante a correta configuração do device tree para o PetaLinux, pois a ordem em que cada aplicação que é adicionada no device tree tem efeito no funcionamento do sistema operacional. Nesse caso é necessário modificar o device tree nativo do BSP para inserir os drivers *iio-AD9361* da Analog Devices que são usados para a comunicação com o transceiver FMCOMMS3

### 3.5.8 Build e instalação do OS na ZCU104

Após finalizar a configuração do device tree é necessário fazer o *build* do novo sistema operacional. Ao finalizar será necessário copiar esses arquivos para a pasta raiz de um SD *card* e então configurar os pinos de incialização do kit de desenvolvimento ZCU104 para realizar o boot pelo SD *card*.

# 4 Arquitetura Proposta e Implementação

A arquitetura proposta evidenciada na figura 13 foi implementada conforme proposto na seção de metodologia, integrando todos os sub-blocos do DDC. A seguir será apresentada uma descrição de cada componente.

## 4.1 Misturador

O misturador foi implementado como uma multiplicação sem o controle do ganho, ou seja, com 0dB, pois é interessante que esse controle seja feito exclusivamente pelos filtros ao longo da cadeia de processamento do DDC. A figura 17 mostra o RTL e a interface do misturador, onde *sigA* é o sinal vindo do A/D de 12 bits, *sigB* é o sinal vindo do DDS com 17 bits e *sigOut* é a saída do misturador com 11 bits. O *clk* e *reset* são respectivamente o clock master e o reset assíncrono. Vale ressaltar que essa multiplicação será implementada usando um bloco DSP no hardware do FPGA.

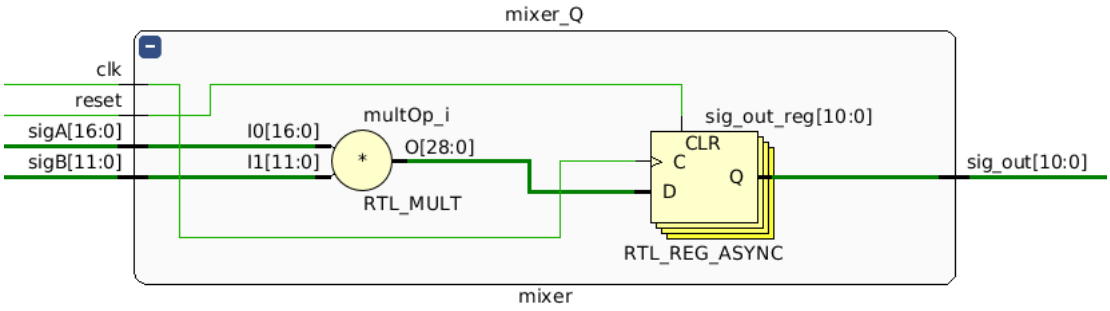



Figura 17 – Interface RTL do misturador.

## 4.2 DDS

Para o projeto do DDS primeiro foi feito no MATLAB o cálculo de um quarto do período da função seno, onde o número de amostras deve ser igual a um quarto da profundidade da memória que, para essa arquitetura foi de 16384, além disso o processo de quantização foi de 11bits em ponto fixo (2 bits da parte inteira e 9 bits da parte fracionária). A figura 18 mostra o gráfico das amostras calculadas no MATLAB que serão carregadas para o conversor fase/amplitude (memória ROM). Como pode ser visto será necessário apenas um quarto do período da senoide na memória, pois os outros três quartos é possível reconstruir a partir de translações do primeiro, economizando assim o uso de memória ROM. Considerando que a frequência do clock *master* do projeto é de 100 MHz, a menor frequência que poderá ser alcançada é $f_{min} = \frac{f_{clk}}{65536} = 1525,87\ Hz = 1,528\ kHz$.

Tabela 3 – Relação dos parâmetros do DDS conforme a profundidade da memória.

| **N** | **Número de amostras em um ciclo senoidal** | **Número de amostras armazenadas em memória** | **Tamanho da Memória ROM** | $f_{min}$ **(Hz)** |
|---|---|---|---|---|
| 10 | 1024 | 256 | 352 B | 97656,25 |
| 11 | 2048 | 512 | 704 B | 48828,13 |
| 12 | 4096 | 1024 | 1,408 KB | 24414,06 |
| 13 | 8192 | 2048 | 2,816 KB | 12207,03 |
| **14** | 16384 | 4096 | 5,632 KB | 6103,52 |
| **15** | 32768 | 8192 | 11,264 KB | 3051,76 |
| **16** | 65536 | 16384 | 22,528 KB | 1525,88 |
| **17** | 131072 | 32768 | 45,056 KB | 762,94 |
| **18** | 262144 | 65536 | 90,112 KB | 381,47 |
| **19** | 524288 | 131072 | 180,224 KB | 190,73 |
| **20** | 1048576 | 262144 | 360,448 KB | 95,37 |

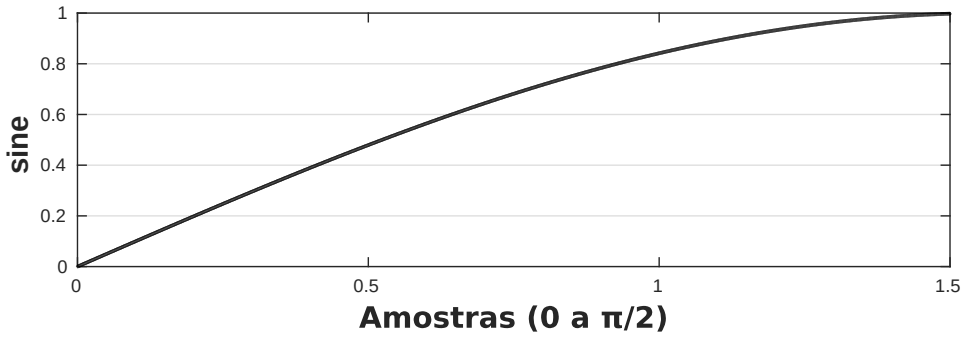



Figura 18 – Amostras que serão carregadas para o conversor de fase/amplitude.

De acordo com [7] não é adequado que a frequência do DDS para essa arquitetura seja maior que um terço da frequência do clock, isto respeita com uma margem grande a equação 2.3 conhecida como critério de Nyquist. Portanto, a maior frequência que esse DDS consegue alcançar sem perda significativa de SNR é $f_{max} = \frac{100\ MHz}{3} = 33.33\ MHz$. Por fim sabe-se que a resolução do DDS é igual a frequência mínima, logo $\Delta f = 1,5287kHz$. A tabela 3 mostra como esses parâmetros variam conforme N (o tamanho em bits do endereçamento da memória). O número de amostras em um ciclo senoidal representa o DDS funcionando na frequência mínima onde não há decimação, já a quantidade de amostras armazenadas em memória e o tamanho da ROM são sempre constantes para cada valor de $N$.

A codificação do acumulador de fase e do conversor fase/amplitude em VHDL foi feita considerando todos os parâmetros de integração das variáveis a nível de síntese lógica, como tamanho da palavra e tamanho da memória. Para a implementação da função cosseno, conforme especificado, foi feita uma inicialização de $\frac{Amostras}{4}$ da mesma memória no acumulador de fase, aplicando assim um desfasagem equivalente a $\frac{\pi}{4}$.

É interessante ressaltar que há um *trade-off* entre o tamanho da memória e a resolução do DDS. Considerando o protocolo DVB-S2X sabe-se que o CFC não pode permitir um residual de frequência maior que *100 kHz* [2], portanto, com uma resolução de $1,525kHz$ espera-se atender os critérios da norma.

A figura 19 mostra o resultado RTL da descrição em VHDL do DDS onde *en, clk, e reset* são respectivamente, o bit de *enable* do circuito, o clock master de 100 MHz e o reset assíncrono.

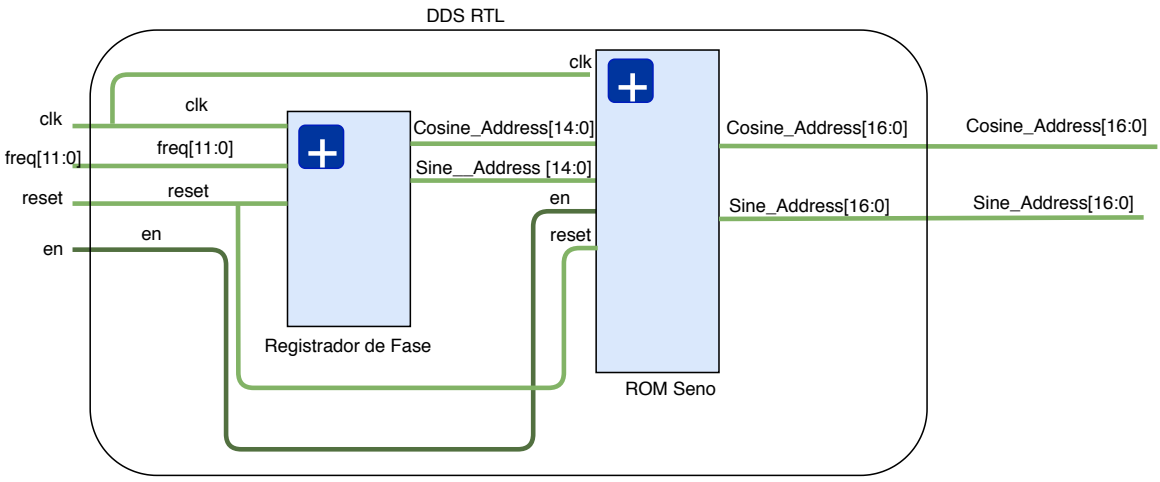



Figura 19 – Interface RTL do DDS.

## 4.3 Filtro FIR

No projeto do filtro FIR foi implementado o diagrama da figura 9. Para os somadores foram feitos agrupamentos de somadores de 2 entradas. Vale ressaltar que os DSPs da Xilinx são projetados para efetuarem uma multiplicação e uma soma em 1 ciclo de clock. Neste trabalho, para um filtro FIR de 16 coeficientes foi usado usando ao total 32 DSPs (16 para fase e 16 para quadratura), um resultado satisfatório se comparados às implementações de hardware em [5] [8] que usam também 32 coeficientes. A frequência de corte desse filtro foi projetada para 1MHz, sendo possível observar na figura 20 que a frequência portadora (10 MHz) e suas multiplicações são atenuadas preservando a informação em 1 MHz.

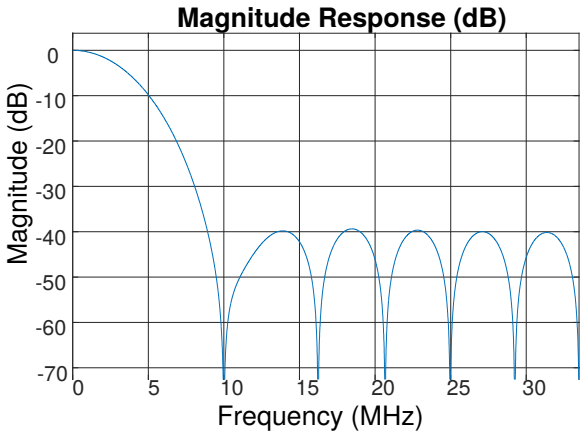



Figura 20 – Resposta da função de transferência do filtro FIR com 16 coeficientes.

A figura 21 mostra a interface RTL do projeto em VHDL do filtro. Vale ressaltar que embora os trucamentos gerem erros conforme discutido na seção anterior, eles são aceitáveis uma vez que a informação na banda de frequência desejada sofre pouca interferência.

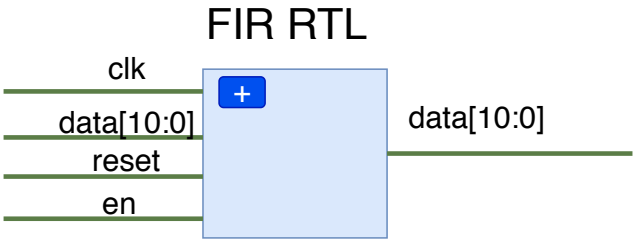



Figura 21 – Interface RTL do filtro FIR.

## 4.4 Integração do DDC

Após a descrição e validação de cada bloco do DDC. Foi necessário instanciar dois blocos para FIR e misturador. A figura 22 mostra o RTL final do DDC. O parâmetro $K$ foi chamado de *freq* na arquitetura e será integrado ao CFC.

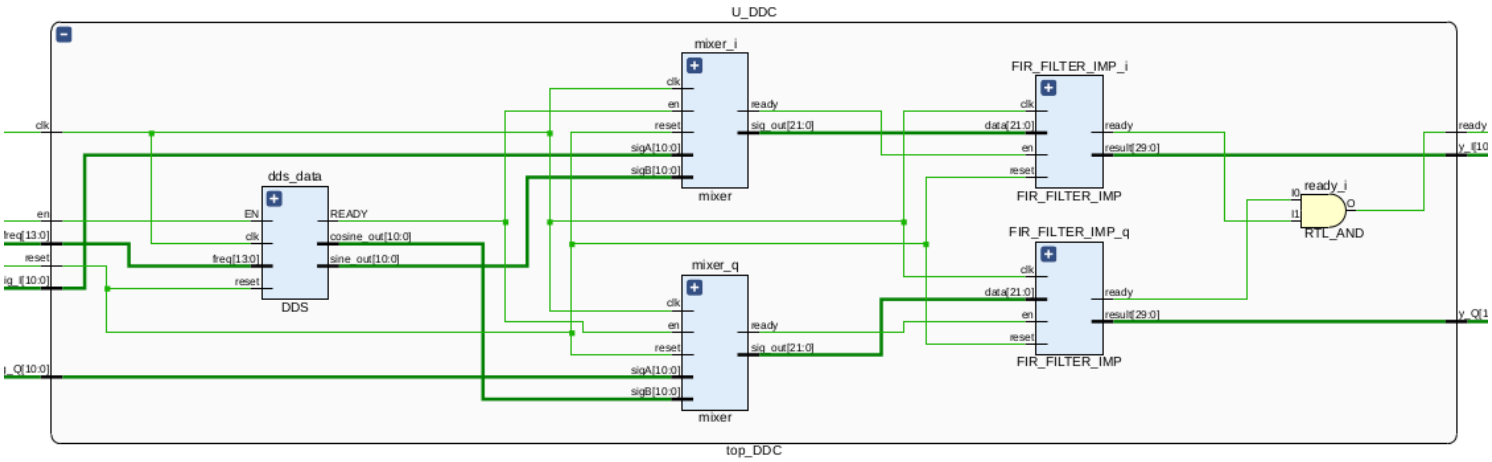



Figura 22 – Interface RTL do DDC.

## 4.5 Integração do DDC com o CFC

Um dos problemas mais evidentes ao fazer a integração do bloco do DDC com o CFC foi a interface, i.e o tamanho das palavras binárias e seu respectivo formato. O CFC possui uma saída para representar o desvio de frequência em ponto fixo de 25 bits sendo 22 fracionários. Por outro lado o DDC possui uma entrada inteira de 14 bits para representar múltiplos da frequência mínima, conforme discutido na seção de metodologia.

Por essa razão foi projetado um bloco nomeado de interpretador. Com um funcionamento combinacional ele interpreta a frequência da saída do CFC e converte para múltiplos da frequência mínima do DDC, tornando assim a implementação factível.

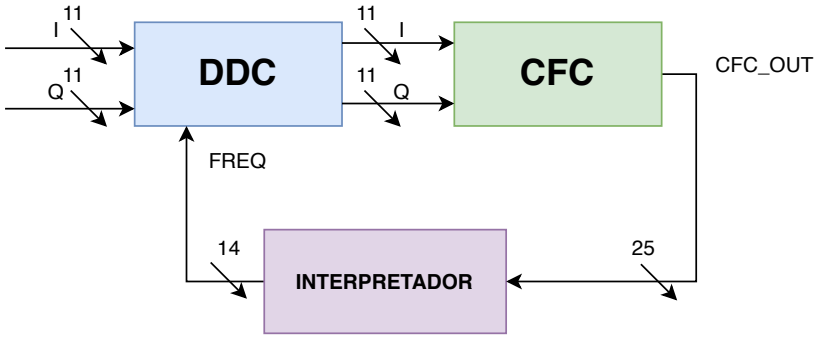



Figura 23 – Arquitetura proposta para a integração do DDC com CFC.

Observa-se na figura 23 o bloco interpretador recebendo a entrada de 25 bits do CFC, e na saída para o DDC uma palavra de 14 bits, respeitando dessa forma a interface dos dois blocos. Ressalta-se que o interpretador é apenas um bloco combinacional, mais precisamente um decodificador que traduz a frequência do CFC para múltiplos inteiros da frequência mínima do DDC. Embora omitido na figura 23 conforme informado na seção de metodologia todos os blocos possuem sinais de *enable* e *ready* conectados entre

si, permitindo assim que o CFC não calcule um desvio de frequência sem anteriormente ter sido corrigido pelo DDC, e o DDC receba um desvio de frequência relativo a uma amostra que já foi corrigida anteriormente. Isso é de suma importância para que a malha fechada DDC-INTERPRETADOR-CFC consiga convergir. A figura 24 mostra o resultado da arquitetura RTL implementada com todos os sinais de controle.

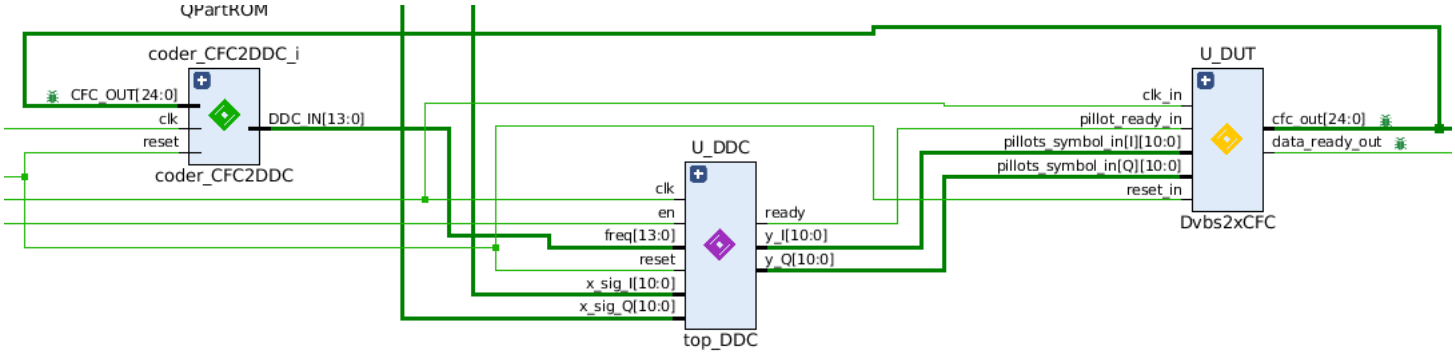



Figura 24 – RTL da malha DDC e CFC implementada.

## 4.6 Implementação do ambiente de teste e simulação DDC-CFC

Conforme proposto na seção anterior, duas memórias ROMs e uma memória RAM fora usadas para fazer a integração dos blocos. Nota-se na figura 25 que todos os sinais de controle foram usados para integrar os blocos.

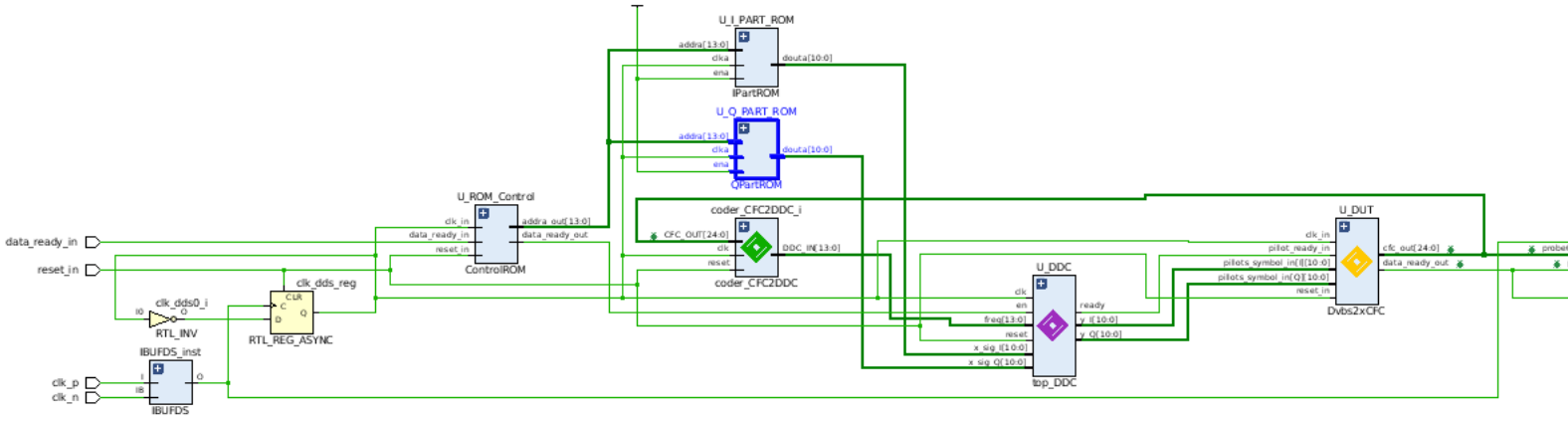



Figura 25 – RTL do DDC integrado com o CFC.

## 4.7 Implementação e teste do PetaLinux

Para que fosse possível testar a implementação do PetaLinux foi necessário um ambiente de prototipagem. Foi usado um kit de desenvolvimento SDR Adalm Pluto da

Analog Devices que enviará os arquivos de testes para o transceiver FMCOMMS3 integrado no ZCU104 por meio das portas FMC.

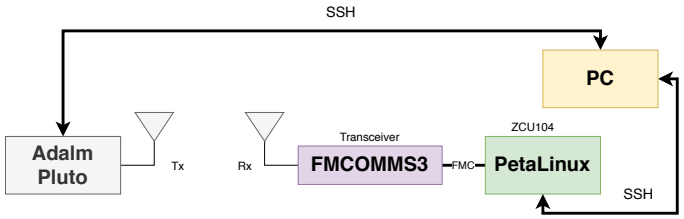



Figura 26 – Ambiente de prototipagem do PetaLinux.

A figura 26 mostra como foi feita essa comunicação. Por meio de *scripts* SSH depois de instalar interfaces AXIs no IP do computador que irá enviar as instruções para o PetaLinux e receber os arquivos, também foi feita uma comunicação SSH com a placa Adalm Pluto que irá enviar um sinal senoidal para a verificação dos testes.

Por meio das antenas RF da Adalm Pluto será enviado o sinal senoidal em uma taxa de 69 MSPS e o transceiver receberá nessa mesma taxa, salvando o sinal já em fase e quadratura em um arquivo de texto que será acessado pelo computador, para posterior comparação.

# 5 Resultados e Discussões

## 5.1 Caracterização do Circuito

Foi feita a implementação física do down-converter proposto em uma placa de desenvolvimento ZCU104 (dispositivo Xilinx ZynqU+) com duas memórias ROMs de 11 bits apresentando 10000 vetores de teste de um sinal de entrada e integrado com blocos RAMs e o ILA conforme discutido anteriormente. Foram usados diversos vetores de testes com desvios de frequências variados de 0,02 a 0,2 em frequência normalizada. Foi utilizado a constelação QPSK com os critérios prescritos na norma DVB-S2X.

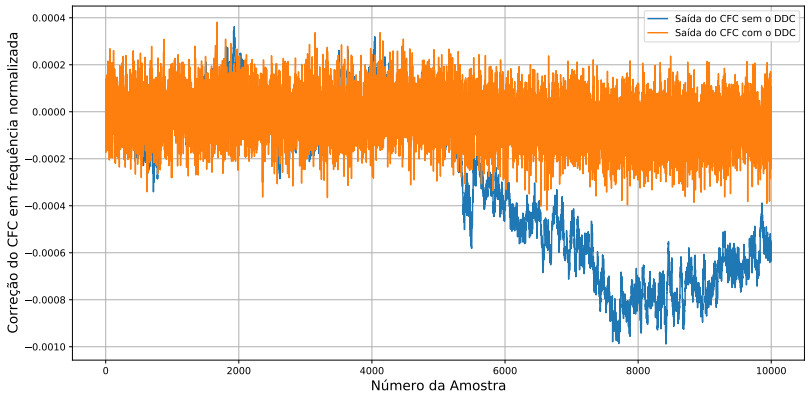
É possível observar na figura 27.A que mesmo quando o desvio de frequência é na ordem de 0,02 em relação a frequência normalizada o DDC ainda consegue corrigir o desvio de frequência informado pelo CFC. Também é observado nas figuras 27.B e 27.C que conforme o desvio de frequência aumenta o DDC consegue com êxito corrigir grande parte do desvio de frequência.

## 5.2 MSE, Correlação Cruzada e Diferença Absoluta dos sinais

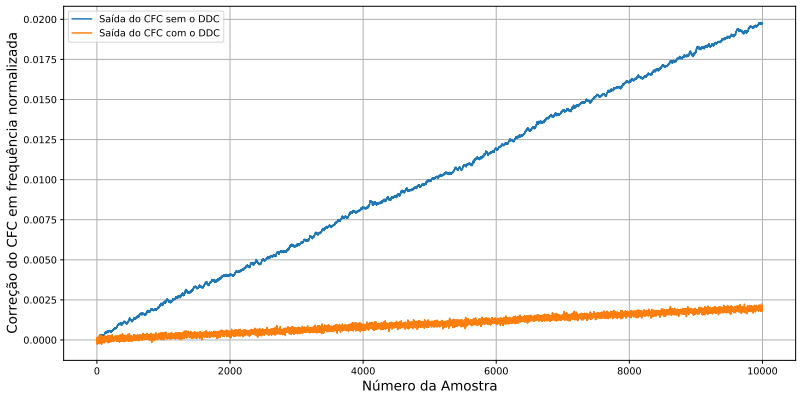
Para a comparação entre os sinais da saída do CFC com o DDC e da saída do CFC sem o DDC foram feitos os cálculos do MSE, correlação cruzada e da diferença absoluta. A figura 28 mostra o gráfico da diferença absoluta entre os sinais pelo desvio de frequência. A tabela 4 mostra que o MSE entre os sinais são baixos, isso implica uma baixa relação numérica entre os sinais. O corrobora a atuação do DDC no CFC. O índice de correlação cruzada mostra que os sinais embora tenha amplitudes diferentes, possuem um nível de semelhança. Já a diferença absoluta alta evidenciada na tabela 4 indica que embora os sinais possam ser semelhantes possuem uma diferença de amplitude considerável.

## 5.3 Análise da Constelação

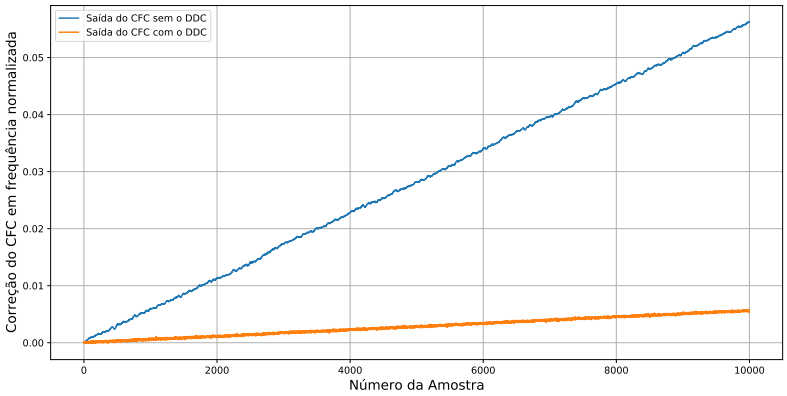
A diferença do desvio de frequência com e sem o DDC corrigindo a frequência também é observada na constelação dos sinais. Para o teste foi usado a modulação QPSK com uma taxa de código de 1/4. É válido ressaltar que os blocos de *time recovery* e *fine frequency correction* ainda devem ser integrados na cadeia do signal processing para que a constelação seja satisfatória no padrão DVB-S2X.




A. Resultado da saída do CFC com o desvio de frequência de 0,02 a 0,04.




B. Resultado da saída do CFC com o desvio de frequência de 0,04 a 0,06.




C. Resultado da saída do CFC com o desvio de frequência de 0,06 a 0,08.

Figura 27 – Resultado do desvio de frequência do CFC entre 0,02 a 0,08 com a presença e ausência do DDC.

## 5.4 Utilização de Recursos

A tabela 5 mostra a utilização dos recursos implementando o DDC e o CFC para um dispositivo Xilinx XCZU7 EV no kit de desenvolvimento ZCU104. O DDS tem o uso

Tabela 4 – Resultados do MSE, Correlação cruzada e Diferença absoluta dos sinais conforme desvio de frequência.

| **Desvio de frequência** | **MSE** | **Correlação Cruzada** | **Diferença Absoluta** |
|---|---|---|---|
| 0,00 a 0,02 | 0,00000017 | 0,00019942 | 2,42137225 |
| 0,02 a 0,04 | 0,00010839 | 0,13389381 | 90,47120129 |
| 0,04 a 0,06 | 0,00043249 | 0,53414340 | 181,10994563 |
| 0,06 a 0,08 | 0,00086361 | 1,06607940 | 254,76079820 |
| 0,08 a 0,10 | 0,00134213 | 1,65683231 | 319,21673658 |
| 0,10 a 0,12 | 0,00171180 | 2,11378868 | 359,34120863 |
| 0,12 a 0,14 | 0,00185748 | 2,29314816 | 373,65364618 |
| 0,14 a 0,16 | 0,00177270 | 2,18825012 | 364,83688081 |
| 0,16 a 0,18 | 0,00157380 | 1,94296712 | 345,46596876 |
| 0,18 a 0,2 | 0,00110345 | 1,36295165 | 288,85296319 |

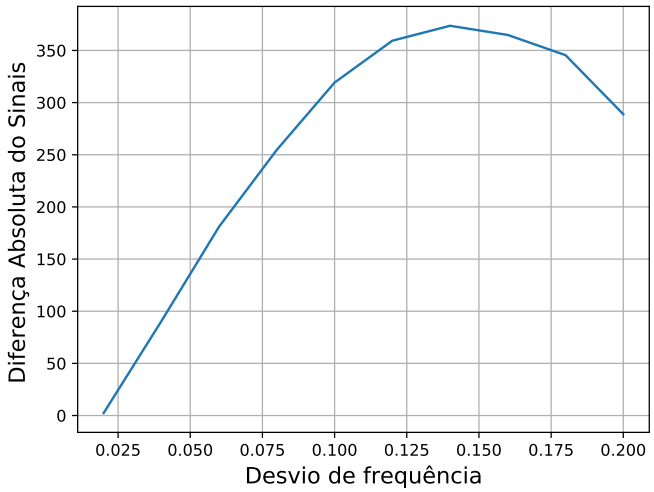
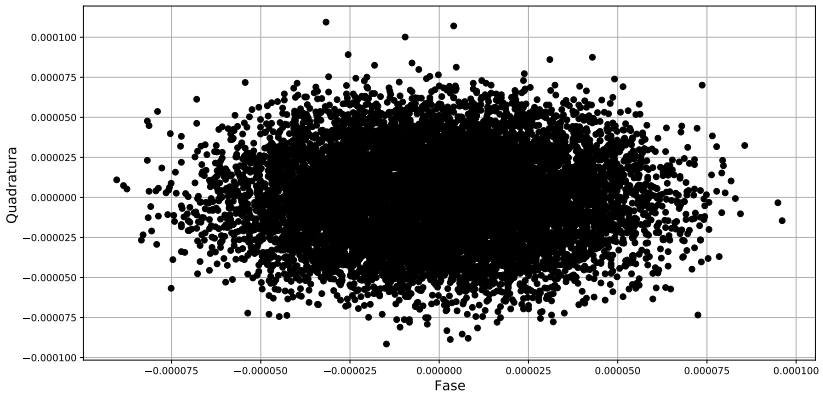


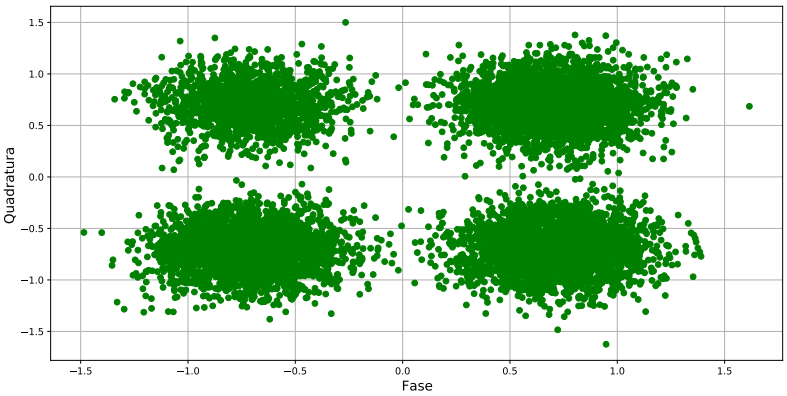

Figura 28 – Diferença absoluta da saída do CFC com e sem o DDC.

mais significativo de LUTs (0,9%) devido ao grande número de amostras que o conversor de fase/amplitude tem para armazenar (ROM). Todos os estágios dos filtro FIR são registrados, consumindo 170 *slices registers* cada. Os misturadors usam dois blocos DSPs (fase e quadratura), enquanto os FIRs usam um DSP para cada coeficiente resultando em 32 DSPs (16 para fase e 16 para quadratura), permitindo uma taxa de saída de uma amostra por ciclo de clock. Após algumas otimizações de síntese o DDC requiriu um total de 34 blocos DSPs.

A figura 29 mostra o layout do circuito, o que demonstra que o DDC proposto foi efetivamente mapeado no dispositivo Xilinx XCZU7 EV juntamente com o CFC, e permite observar a distribuição de cada componente no dispositivo em questão. Os misturadores não são visíveis pois são essencialmente 1 DSP com a saída registrada.




A. Constelação sem o DDC.




B. Constelação com o DDC

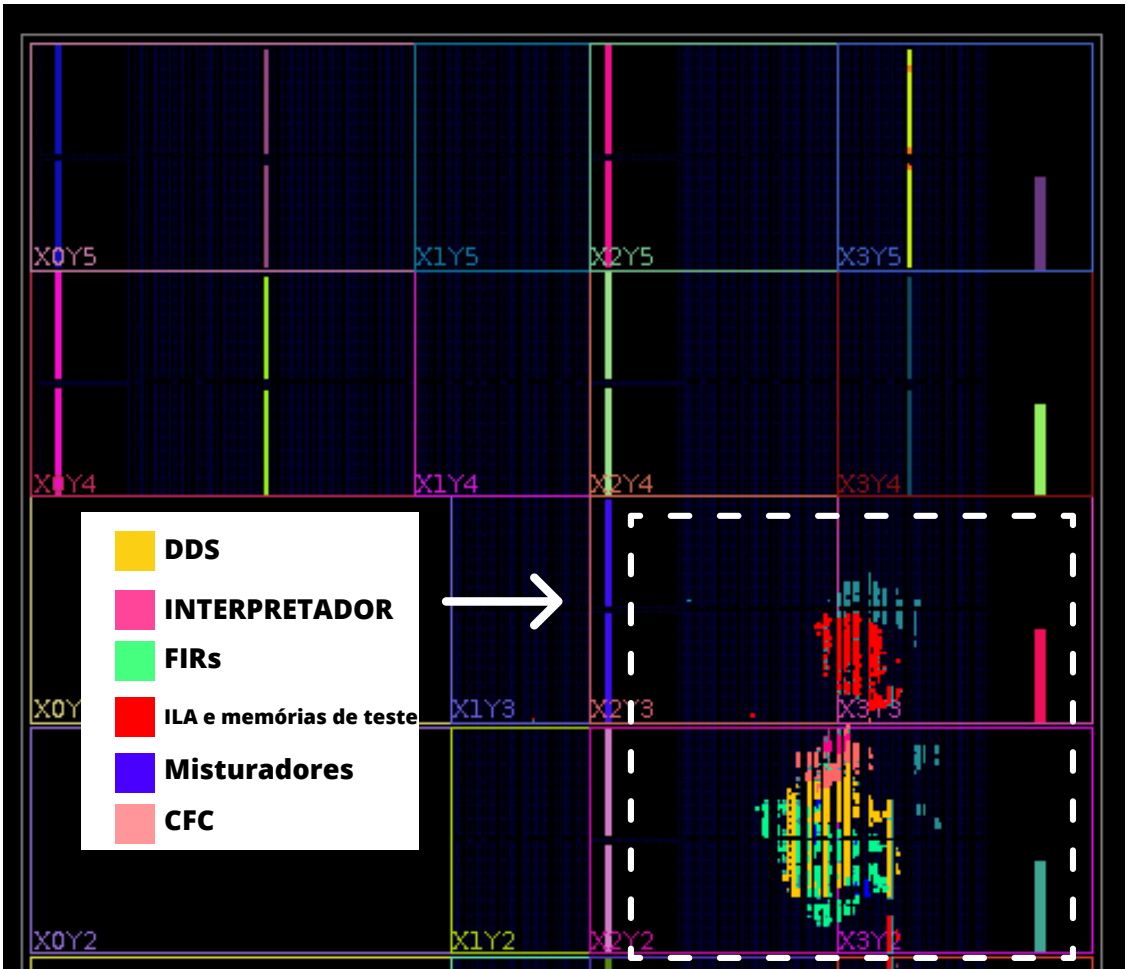


Tabela 5 – Utilização dos recursos do DDC implementado em hardware juntamente com o CFC no chip XCZU7 EV na placa ZCU104 .

| **Desvio de frequência** | ***Slice* LUTs (máx: 230400)** | **DSPs (máx: 1728)** | ***Slice* Regs (máx: 460800)** | **Blocos de BRAM (máx: 312)** |
|---|---|---|---|---|
| DDS | 2072 (0,9%) | 0 | 58 (0,01%) | 0 |
| FIR | 1192 (0,52%) | 32 (1,85%) | 340 (0,07%) | 0 |
| MISTURADOR | 50 (0,02%) | 2 (0,12%) | 2 (0,0004%) | 0 |
| INTERPRETADOR | 221 (0,10%) | 0 | 14 (0,003%) | 0 |
| CFC [4] | 459 (0,20%) | 3 (0,17%) | 367 (0,0796%) | 0 |
| ILA* | 847 (0,37%) | 0 | 1216 (0,2639%) | 1 (0,032%) |
| Memórias para o teste* | 24 (0,01%) | 0 | 12 (0,0026%) | 8 (2,564%) |
| **Total** | **4865 (4,22%)** | **37 (2,14%)** | **2009 (0,436%)** | **9 (2,884%)** |

(*): Circuitos que não fazem parte da arquitetura final do projeto, mas são importantes para a obtenção dos resultados, possibilitando a injeção de dados e a captura deles após toda cadeia de processamento.

Em termos de tempo de execução, o DDC proposto possui uma latência de 19 ciclos de clock que, operando a uma frequência de 100MHz, representam 190 ns. Como o DDC foi implementado usando uma arquitetura pipeline, após a latência inicial, a taxa de saída é de uma amostra por ciclo de clock, correspondente a 100 MSPS. A Figura 30 mostra o consumo de energia estimado do circuito (confiança de baixo nível). O total de 0,634 Watts é dissipado pelo DDC, 97% para potência estática e 3% de potência dinâmica, consumidos principalmente pelos circuitos lógicos e registradores.

## 5.5 PetaLinux: Implementação e resultados de transmissão em *loop*

A figura 31 mostra o ambiente de teste projetado no software GNU Radio para verificar o sistema de recepção do transceiver FMCOMMS3 integrado na ZCU104 a partir do PetaLinux desenvolvido. Foi usado um demodulador para o padrão DVB-S2X nativo do software que está conectado a uma fonte de número randômicos de 0 a 255 com pelo menos um mil amostras. O sinal complexo será enviado pelo transceiver da placa Adalm Pluto com algumas configurações padrões evidenciadas na tabela 6.

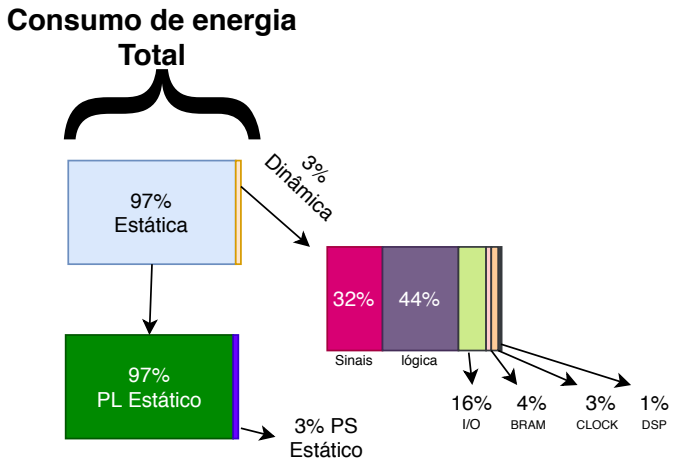



Figura 30 – Estimação do consumo energético do DDC integrado com o CFC.

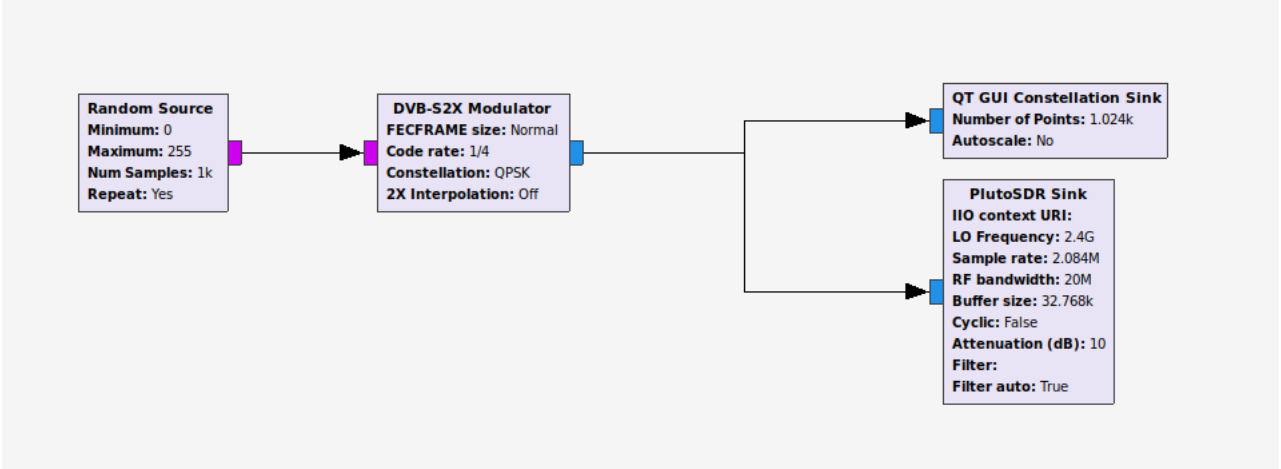



Figura 31 – Ambiente de Teste desenvolvido no GNU Radio com a Adalm Pluto para testar a recepção do PetaLinux.

Tabela 6 – Configurações para o teste na Adalm Pluto e no FMCOMMS3.

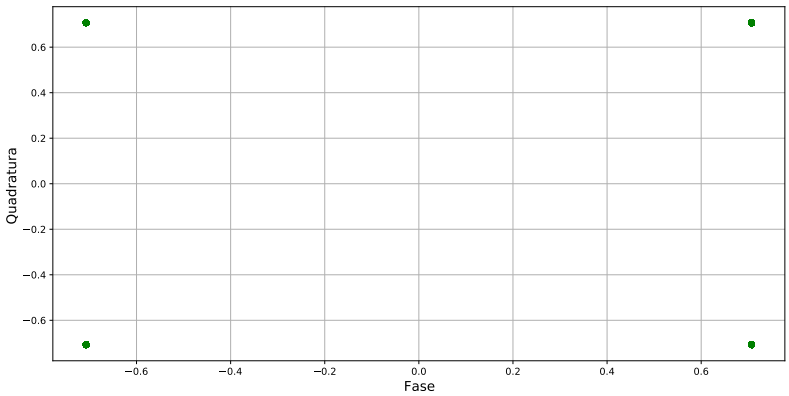
| | **Configurações na Adalm Pluto (Rx)** | **Configurações no FMCOMMS3 pelo PetaLinux (Tx)** |
|---|---|---|
| Frequência LO | 2,4G | 2,4G |
| Taxa de Amostragem | 2,084 MSPS | 2,2 MSPS |
| Banda de RF | 20M | 60M |

### 5.5.1 Análise dos dados recebidos

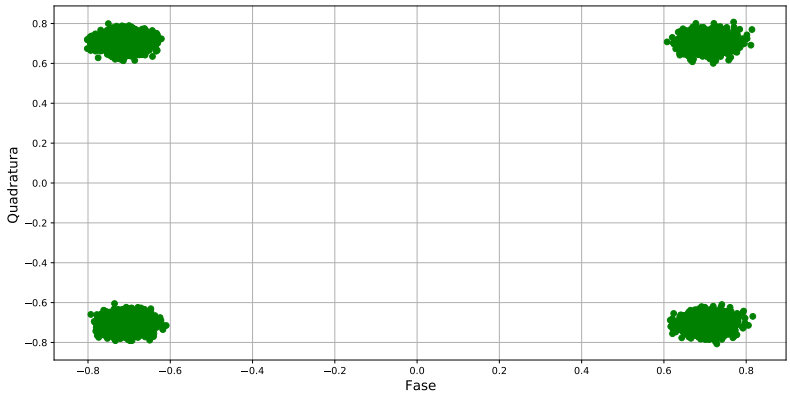
A figura 32.A mostra a constelação produzida no GNU Radio onde praticamente não há ruído para o sinal que será enviado. A figura 32.B mostra o sinal recebido pelo PetaLinux. Existe um certo nível de ruído, porém vale lembrar que não há nenhum bloco do signal processing integrado com o PetaLinux , i.e não foi feita nenhuma correção de frequência, fase, tempo entre outros procedimentos inerentes ao DVB-S2X, ainda assim

nota-se um grau de sucesso elevado na recepção do sinal.

Na figura 33 foi feito o envio do sinal com a mesma taxa de código, porém, agora com uma constelação 8PSK onde se evidencia um baixo nível de ruído.




A. Constelação QPSK enviada pelo GNU Radio.




B. Constelação QPSK recebida no PetaLinux.

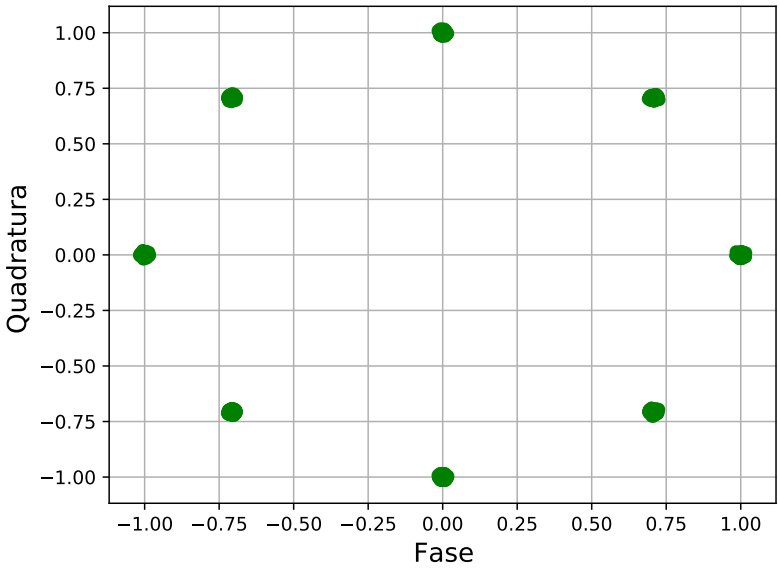
Figura 32 – Comparação das constelações QPSK.

Nas figuras 34 foi feita a correlação cruzada entre a parte real e a parte imaginária dos sinais. É possível a caracterização de um ruído gaussiano em ambos os sinais, o que de fato é esperado uma vez que a fonte geradora do sinal original era um ruído branco.

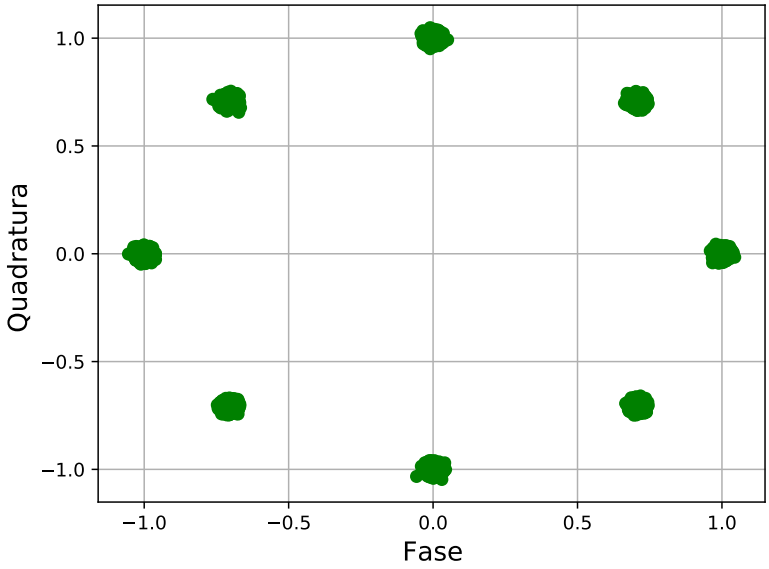
É possível notar que a parte real possui uma suavidade maior em determinadas amostras da correlação, porém é perceptível que na parte imaginária há menos resultados nulos entre as amostras do sinais, o que evidencia a correlação dos sinais.

A correlação cruzada permite validar que os sinal enviado possui uma semelhança matemática com o sinal recebido, observando na figura 34.A e 34.B O formato de uma curva gaussiana entre os sinais.

É observado que ambos sinais são centrados em *10000* devido ao número de amostras, evidenciando que não há uma defasagem significativa entre os sinais uma vez que o




A. Constelação 8PSK enviada pelo GNU Radio.

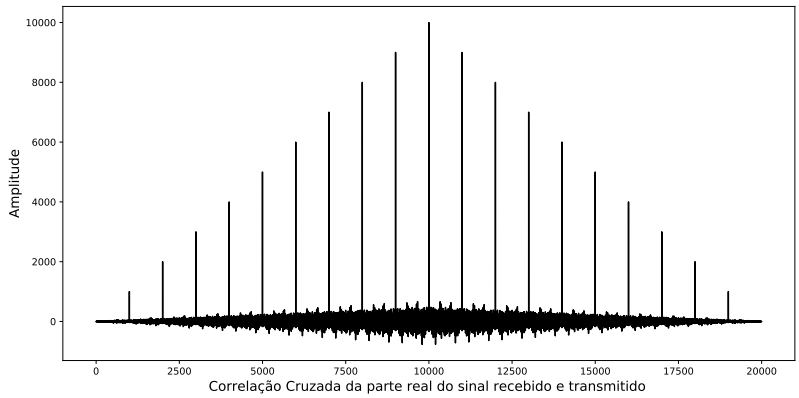



B. Constelação 8PSK recebida no PetaLinux.

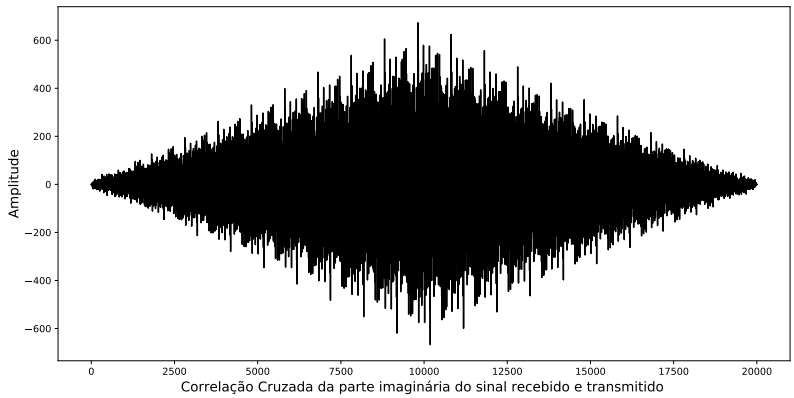
Figura 33 – Comparação das constelações 8PSK.

sinal como todo tem *10000* amostras.

Também é observado uma diferença significativa entre os sinal real e imaginário, isso é devido ao próprio bloco de pilotos que foram usados para gerarem as amostras.




A. Correlação Cruzada entre os sinais reais enviados e recebidos no teste.




B. Correlação Cruzada entre os sinais imaginários enviados e recebidos no teste.

Figura 34 – Correlação cruzada entre os sinais reais e imaginários enviados e recebidos.

# 6 Conclusão

## 6.1 Conclusões

Este trabalho propõe uma arquitetura FPGA parametrizável de um Digital Down Converter (DDC) para canais de satélite. O DDC proposto usa um DDS, misturador digital e filtros FIR em cascata com baixa latência e consumo de recursos de hardware. A arquitetura é parametrizada em termos de resolução de frequência, largura de palavra, tamanho da amostra do DDS, ordem dos filtros e ordem dos estágios de decimação. Um estudo de caso para o padrão DVB-S2X foi implementado em um dispositivo Xilinx ZCU104, atingindo latência de 19 ciclos de clock, taxa de saída de 1MSPS e consumo de 3535 LUTs, 781 *slice registers* e 34 blocos de DSP. O desempenho foi avaliado por simulações numéricas e implementação física utilizando a ferramenta de debug em hardware da Xilinx ILA *(Integrated Logic Analyzer)*, com vetores de testes gerados para o padrão DVB-S2X com desvio de frequência de 0,02 a 0,2 em relação a frequência nominal. A arquitetura DDC proposta utiliza menos slices registers em comparação com [2], [7], [8]. Os resultados obtidos com a integração do CFC (*coarse frequency correction)* normatizado pelo padrão DVB-S2X mostram que o DDC proposto nesse trabalho atende a norma em todos os quesitos de ruído de frequência e fase. Mostra-se também que o DDC é capaz de corrigir uma faixa de frequência de 1,5 kHz a 33 MHz, sendo que a faixa de frequência exigida é limitada pelo CFC a cerca de 10 MHz pelo fato do próprio módulo CFC não conseguir atuar com um desvio de frequência maior que 10 MHz.

Nesse mesmo trabalho foi feito o projeto de um sistema operacional baseado no PetaLinux para se tornar o bloco *front-end* do sistema de comunicação. O PetaLinux que foi desenvolvido opera na placa de desenvolvimento ZCU104 integrado com o transceiver FMCOMMS3 a partir das porta FMC presentes no dispositivo. Foram feitos diversos testes de transmissão a partir do software GNU Radio utilizando o modulador DVB-S2X presente na ferramenta. Os testes de transmissão e recepção mostraram que o sistema operacional foi capaz de receber um sinal modulado para o protocolo DVB-S2X, se mostrando uma solução adequada e economicamente mais viável do que utilizando a placa de desenvolvimento ZCU102 oficialmente distribuída pela Analog Devices para a integração com o FMCOMMS3.

Evidenciou-se pela comparação do sinal enviado e recebido, tanto pelas constelações quanto pela correlação cruzada que a informação enviada conseguiu ser recebida com uma banda de frequência adequada pelo transceiver FMCOMMS3 integrado na ZCU104 operando o sistema operacional desenvolvido.

# Referências

1 KOEN, W. DVB-S2X demystified. 2014. Disponível em: <https://www.newtec.eu/frontend/files/userfiles/files/Whitepaper%20{DVB}_S2X.p>. Citado 2 vezes nas páginas 23 e 24.

2 EUROPEAN TELECOMMUNICATIONS STANDARDS INSTITUTE. *DVBS2X 301*: Digital video broadcasting (DVB); second generation DVB interactive satellite system(DVB- RCS2); part 2 lower layers for satellite standard. [S.l.], 2014. 15 p. Citado 4 vezes nas páginas 24, 33, 51 e 67.

3 T.HOLLIS; R.WEIR. The theory of digital down conversion. 2003. Citado 3 vezes nas páginas 24, 31 e 32.

4 SALLES, A. Implementação em fpga dos módulos correção grosseira de frequência e correção de fase aderentes ao protocolo dvb-s2x, trabalho de conclusão de curso em engenharia eletrônica fga/unb. 2020. Citado 2 vezes nas páginas 24 e 61.

5 SAVVOPOULOS, P.; ANTONAKOPOULOS, T. An IF digital down-converter for software radio DVB-S2 receivers. In: . [S.l.: s.n.], 2008. p. 1043 – 1046. Citado 4 vezes nas páginas 25, 28, 29 e 30.

6 KIT, Z. U. M. Z. E. Disponível em: https://www.xilinx.com/products/boards-and-kits/zcu104.html. nov.2020. Citado na página 27.

7 LUO, F.-L. Digital front-end in wireless communications and broadcasting. Circuits and signal processing. 01 2011. Citado 6 vezes nas páginas 28, 29, 30, 32, 50 e 67.

8 OBRADOVIC, V. et al. Practical implementation of digital down conversion for wideband direction finder on FPGA. *Scientific Technical Review*, v. 66, p. 40–46, 01 2016. Citado 2 vezes nas páginas 28 e 67.

9 GIRAU, G. et al. FPGA digital down converter IP for SDR terminals. In: . [S.l.: s.n.], 2002. v. 2, p. 1010 – 1014 vol.2. ISBN 0-7803-7576-9. Citado 3 vezes nas páginas 28, 29 e 30.

10 Changrui, W. et al. Design and FPGA implementation of flexible and efficiency digital down converter. In: *IEEE 10th INTERNATIONAL CONFERENCE ON SIGNAL PROCESSING PROCEEDINGS*. [S.l.: s.n.], 2010. p. 438–441. Citado na página 29.

11 DATTA, D.; MITRA, P.; DUTTA, H. S. FPGA implementation of high performance digital down converter for software defined radio. *Microsystem Technologies*, Springer, p. 1–10, 2019. Citado 2 vezes nas páginas 29 e 30.

12 LIU, X. et al. Design and FPGA implementation of a reconfigurable digital down converter for wideband applications. *IEEE Transactions on very large scale integration (vlsi) systems*, IEEE, v. 25, n. 12, p. 3548–3552, 2017. Citado 3 vezes nas páginas 29, 30 e 38.

13 LATHI, B. P. *Modern Digital and Analog Communication Systems*. [S.l.]: Oxford University Press, Inc., 1998. Citado na página 32.

14 RHYNE, G. W.; STEER, M. B.; BATES, B. D. Frequency-domain nonlinear circuit analysis using generalized power series. *IEEE Transactions on Microwave Theory and Techniques*, IEEE, v. 36, n. 2, p. 379–387, 1988. Citado na página 33.

15 KROUPA, V. F. *A Digital Frequency Synthesizer*. [S.l.]: Wiley-IEEE Press, 1999. Citado na página 33.

16 DORF, R. C.; BISHOP, R. H. *Modern control systems*. [S.l.]: Pearson, 2011. Citado na página 34.

17 WIDROW, B.; KOLLÁR, I. Quantization noise. *Cambridge University Press*, v. 2, p. 5, 2008. Citado na página 34.

18 HAUCK, S.; DEHON, A. *Reconfigurable computing: the theory and practice of FPGA-based computation*. [S.l.]: Elsevier, 2010. v. 1. Citado 2 vezes nas páginas 35 e 36.

19 XILINX. PetaLinux Tools overview demystified. 2014. Disponível em: <https://www.xilinx.com/content/xilinx/en/products/design-tools/embedded-software/petalinux-sdk.html>. Citado na página 38.

20 FOUNDATION, L. The Linux Foundation Announces Yocto Project Steering Group and Release 1.0 demystified. 2020. Disponível em: <https://www.linuxfoundation.org/press-release/2011/04/the-linux-foundation-announces-yocto-project-steering-group-and-release-1-0/>. Citado na página 42.

21 XILINX. LogiCORE IP Product Guide. 2016. Disponível em: <https://www.xilinx.com/support/documentation/ip_documentation/ila/v6_1/pg172-ila.pdf>. Citado na página 44.

22 ANALOG. AD9361 high performance, highly integrated RF Agile Transceiver™ Linux device driver. 2019. Disponível em: <https://wiki.analog.com/resources/tools-software/linux-drivers/iio-transceiver/ad9361>. Citado na página 46.

# Anexos

# .1 Artigo submetido ao LASCAS 2020

# Design and FPGA Implementation of a Parameterized DDC for Satellite Channels

Jhonathan Silva*, Zoé Magalhães† and Daniel M. Muñoz*
†Autotrac Comércio e Telecomunicações S/A
*Electronics Engineering Program, University of Brasilia
Brasília, DF, Brazil
Email: jnicolas@aluno.unb.br, zoe.magalhaes@autotrac.com.br, damuz@unb.br

***Abstract*—This work presents a design of a Digital Down Converter (DDC) for the DVB-S2X protocol, mapped on Field Programmable Gate Arrays (FPGAs), using a fixed-point arithmetic representation. The proposed architecture is portable to different signal's bandwidth and frequency range, the hardware's coefficients can be configured directly on hardware description code or using Matlab scripts that generate the snippets of hardware description code. In order to enable the online correction of frequency errors on received signal, the DDC structure has an oscillator with run-time configurable frequency. This circuits were synthesized for a Xilinx Zynq device for a sample rate of 100 MSPS and a frequency step of 3 kHz and characterized in terms of precision, hardware resources, latency, throughput, and energy consumption.**

***Index Terms*—Digital Down Converter, Digital Signal Processing, FPGAs, DVBS2X**

## I. Introduction

DVB-S2 is the most accepted and widely spread standard in the satellite market. The extension of the generation DVB-S2 specs, DVB-S2X, was developed for satisfying reliable and effective broadband satellite communication services even under 'hostile' conditions. Compared to DVB-S2, efficiency gains up to $51\%$ can be achieved with DVB-S2X [1].

Digital Down-converter (DDC) is the signal processing front-end of the DVB-S2X receiver. This component is responsible for reducing the frequency IF (Intermediate Frequency) to ZIF (Zero IF) retaining all information [2]. Also, the DDC must adjust internal oscillators as indicated by the frequency corrector module, allowing error frequencies to be reduced [3]. Field Programmable Gate Arrays (FPGAs) can be a feasible solution to implement DDCs, exploring the intrinsic parallelism to achieve high-performance computing applications [4]. However, to meet specific application requirements, it is important to design a parameterizable hardware architecture of the DDC in terms of bandwidth, sample rate, and word-width. It is also essential to analyze how those parameters impact the area and power consumption, latency, and throughput.

DDCs usually are implemented using mixers and combinations of resampling filters such as CIC (Cascade Integrator Comb), VFD (Variable fractional Delay), FIR (Finite Impulse Response), and polyphase filters. High-level synthesis tools have been used to implement DDCs on FPGAs, allowing the main design parameter to be adjusted; however, consuming a large number of hardware resources and DSP blocks [5], [6], [7].

Several previous works implemented DDCs in FPGAs for SDR (Software Defined Radio) systems [4], [5], [7], [8], [9], [10]; however, authors do not mention which architecture parameters can be adjusted. Authors in [7] designed a DDC on a Xilinx Virtex100E device using only CIC filters, saving multipliers, and reducing the computational overhead. Authors in [7] also present an analysis of how the word-width and number of filter stages affect resource consumption and the operational frequency. Authors in [4] present a DDC for the DVB-S2 standard for a Xilinx VirtexIIPro device, which makes use of a DDS (Direct Digital Synthesizer), complex multiplications and 40-taps and 32-taps cascaded FIR filters. The proposed DDC processes IF signals centered at 70 MHz, roll-off of 0.35, and 36 MHz bandwidth, producing output signals of 11.75 MHz bandwidth, consuming around 12277 LUTs and 15000 slice registers. In [10], a DDC for wideband applications based on a combination of several resampling filters was designed for a Xilinx Kintex7 device, producing a down-converted output with a sampling rate between 1 kS/s to 225 MS/s and consuming 11656 LUTs, 18302 slice registers, and 106 DSP blocks, is presented. Recently, authors in [9] proposed a model for implementing DDCs on FPGAs using a CORDIC processor and cascaded CIC and FIR filters, processing input signals of 70 MHz bandwidth, producing output signals of 137 kHz bandwidth and consuming around 470 LUTs, 619 slice registers, and 16 DSP blocks. As the best of our knowledge, there are no contributions in the scientific community implementing parameterizable hardware architectures of the DDCs for DVB-S2X standard.

This paper presents a parameterizable DDC design implemented on Xilinx FPGAs for the DVB-S2X standard. The proposed design aims to be applicable for different input signal characteristics and is based on the well-known architecture composed of a DDS, mixers, and cascaded CIC and FIR decimator filters [5]. To enable the coarse frequency correction recommended for DVB-S2X receivers, the oscillator frequency is run-time configurable. The flexibility is also achieved by the development of a hardware description with configurable parameters that affect the inner DDC components, as presented in Table I. The proposed architecture was synthesized on a Xilinx Zynq7000 devices, and some experiments were con-

ducted for a 100 MSPS, 1 MHz bandwidth, and 10 MHz IF. Simulations and characterizations demonstrate that the proposed DDC is efficient in terms of latency, throughput, and resource consumption.

Table I
PARAMETERIZABLE PROPERTIES OF THE PROPOSED DDC.

| Architecture | Parameter |
|---|---|
| *DDS* | Frequency (in real-time), word-width, frequency resolution, size word sample, size number sample. |
| *CIC* | Order of integrator and comb stage, decimation factor, delay factor. |
| *FIR* | Taps: number and precision. |

## II. BACKGROUND

Figure 1 depicts the main component used by the DDC implemented in this work. It is composed of mixers, cascades CIC and FIR filters, a DDS, and decimators [5]. The Mixers multiply the IF input signal by a sinusoidal signals with the IF carrier frequency [11]. This sinusoidal signal is generated by the DDS, which adjusts the frequency according to a feedback input coming from a baseband signal processing [3] [5]. After the translation, it is necessary to remove the carrier signals using the CIC and FIR filters. The former is a multiplier-less filter that performs a sampling rate conversion (SRC) and enables to reduce the number of taps of the FIR filter [7] [5]. The later is used to meet roll-off requirements and improve phase and frequency aspects, as well as to attenuate the carrier. In the DVBS2X standard, a root raised cosine filter must be applied after the DDC allowing for correction and improvements of the signal spectrum in baseband [3].

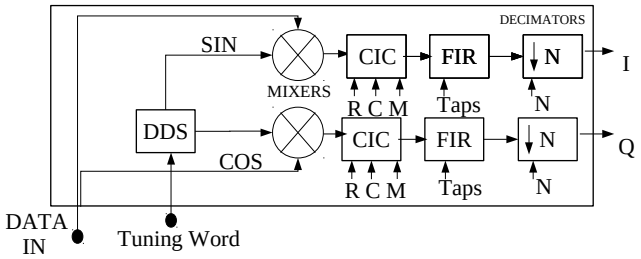



Figure 1. DDC Block Diagram. *Where: R = factor decimator, C = Integration and comb order, M = delay factor of comb filter, Taps = Taps value of FIR, N = factor of final decimation.*

Mathematically, the signal after the A/D conversion is defined by the equation 1, where $I$ and $Q$ are the signals in-phase and quadrature [11], respectively. After the mixers, the signals are denoted by equations 2 and 3, where the last two terms are the passband signals centered in $2\pi f_c$. These terms must be suppressed by the cascade filters, preserving only the I and Q signals [11]. After filtering, a decimation is performed, centering the signal in a frequency $Fs/N$, where $N$ is the factor of decimation.

$$Data_{in} = Acos(2\pi f_c t + \psi) = I(t)cos(2\pi f_c t) + Q(t)sin(2\pi f_c t) \quad (1)$$

$$I(t)' = I(t) + I(t)cos(2\pi f_c t) + Q(t)sin(2\pi f_c t) \quad (2)$$

$$Q(t)' = Q(t) - Q(t)cos(2\pi f_c t) + I(t)sin(2\pi f_c t) \quad (3)$$

## III. DESIGN OF DDC

All components of the DDC were firstly designed in Matlab using a floating-point arithmetic representation. After validation in software, each subsystem block was manually implemented in VHDL using fixed-point arithmetic and exploring the intrinsic parallelism of the algorithms. The hardware architectures were developed in a parameterizable way, including the design parameters listed in Table I, but also trying to achieve the right balance between hardware resources consumption, latency, and throughput. Other essential parameters the designer must deal with are related to the DVB-S2X application, for instance, the sample rate, word width of the ADC, and signal bandwidth. A particular case, using a 12-bits ADC at 100 MSPS, an output signal of 11 bits, and a bandwidth up to 5 MHz, was modeled and implemented in VHDL as proof of concept.

### A. Mixer

A digital mixer is simply an arithmetic function that multiplies the incoming digital source signals $I$ and $Q$, with carrier waveforms generated by DDS [5]. Since the DDS signal and the output were modeled to have 11 bits, it was necessary to truncate the resulting word to fit 11 bits. This truncation is allowed because mathematically, the mixers only serve to translate the frequency and do not necessarily add more precision in the signal. The mixers were designed to work in $0db$ since the gain will be controlled only by filters.

### B. DDS

Direct digital synthesis (DDS) is a method to generate a time-varying signal in digital form from a stored sinusoid waveform in a look-up table (LUT) [12]. Figure 2 shows the general architecture of the DDS.

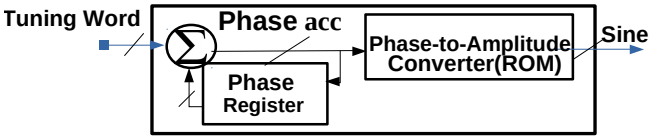



Figure 2. Block diagram of the DDS. The ROM mapped samples only between 0 and $\pi/2$. To implement the cosine signal a simple initialization equivalent to $samplesize/4$ was applied to the phase accumulator.

In general, the larger the number of samples, the greater the accuracy. However, to optimize the LUT size, the frequency resolution should be estimated according to the application. In the DVB-S2X standard, the DDS will receive from the frequency recovery module the $\Delta f$ parameter, which modifies the carrier frequency [2]. According to [13] the frequency recovery tolerates residual normalized carrier frequencies offset up to $3.5 \times 10^{-4}$. The DDS frequency resolution can be modeled as $\Delta f = (f_s/2)/2^B = f_s/2^{B+1}$, where $B$ is the number of bits of the phase register and $f_s$ is the sampling rate or operational frequency of the DDS. Thus, for $f_s = 100$MHz and $\Delta f = 3.1$kHz, the word size of the phase register must be implemented using $B = 12$ bits.

### C. CIC Filter

The cascaded integrator comb (CIC) is a multiplier-free filter that works as a low pass filter and enables to reduce the DSPs usage [5]. Firstly, the filter was designed in Matlab, where numerical simulations were conducted to choose the decimation factor $N$, the number of cascaded integration and comb stages $n$ (order of the filter), and the comb delay $M$, as depicted in Fig. 3. Figure 4 shows the frequency response of the filter for a decimator factor of $N = 2$, an integrator, and comb of third-order ($n = 3$) with two comb factor delay ($M = 2$). The integration, comb, and decimation blocks were designed individually in VHDL using the $N$, $n$, and $M$ factors as generic parameters, allowing different configurations to be obtained after the logical synthesis step.

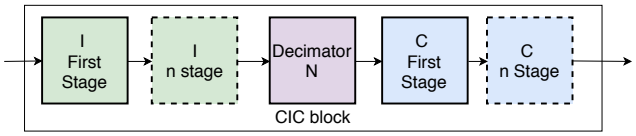



Figure 3. CIC Block. The decimator block in this filter is the same block shown in figure 1.

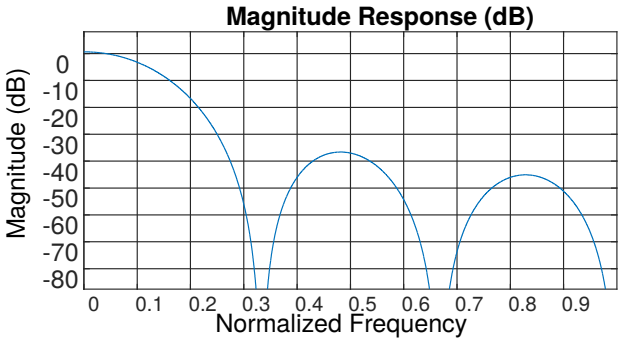



Figure 4. Frequency response of the CIC filter.

### D. FIR Filter

Although the CIC filter is efficient, a more robust and controlled filter process is required to be able to handle roll-off factors and filter cutoff frequency more easily. For this purpose, a finite impulse response (FIR) was designed in Matlab, and after simulation, the selected topology was implemented in VHDL. The frequency response of the designed filter is shown in Fig. 5 with a rectangular window.

In this work, an 8-taps FIR filter was used, saving some DSPs if compared with hardware implementations in [5] [8]. The cutoff frequency of this filter was designed to $1MHz$, and it is possible to note in Fig. 5 that the carrier frequency ($10MHz$) and its multiplies are almost totally attenuated.

## IV. Results

The proposed down-converter was mapped on a Zybo B development board (Xilinx Zynq 7010 device), which was interfaced with a 12-bits ROM contained 5000 samples of the input signal and 2 BRAMs to store the $I$ and $Q$ output signals. Several tests were designed using input signals with no noise, frequency noise, and phase noise. The input signal corresponds to the modulation of a 1MHz sine and square waves for the $I$ and $Q$ signals, respectively. The DDS frequency was fixed

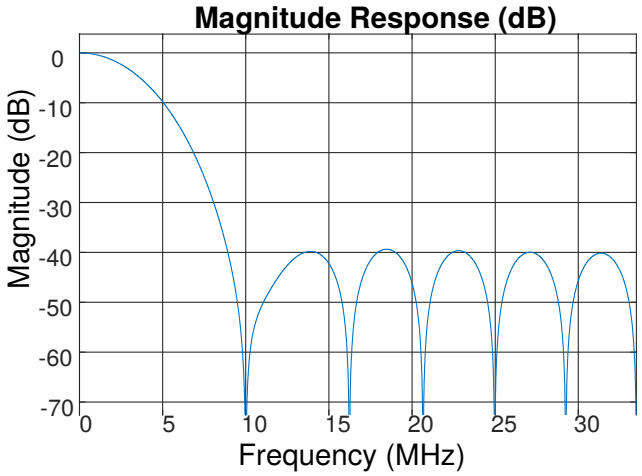



Figure 5. Frequency response of the FIR filter.

in 10.68MHz. The output signals of each component were collected, and the results were compared with the golden model implemented in Matlab, as shown in Fig. 6.

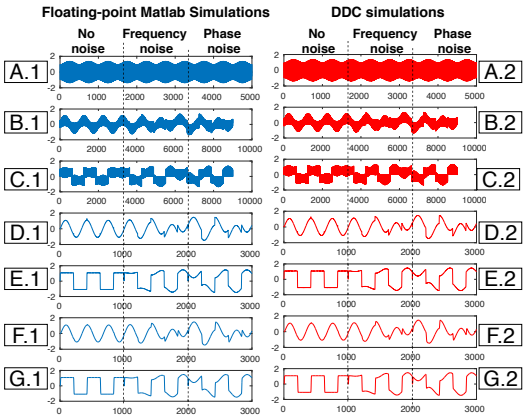



Figure 6. Results of simulation. A: data in. B: $I$ Mixer output. C: $Q$ Mixer output. D: $I$ CIC output. E: $Q$ CIC output. F: $I$ FIR output. G: $Q$ FIR output.

Table II shows the mean squared error of the obtained outputs of each component with respect to the reference model. It is possible to note that all the components have an MSE of around $10^{-6}$, which are expected because of the truncation errors imposed by the arithmetic representation.

Table II
MSE between DDC HW results and Matlab implementation

| Architecture | I MSE | Q MSE |
|---|---|---|
| *Mixer output* | 5.2602e-06 | 5.4475e-06 |
| *CIC output* | 5.4717e-06 | 5.4717e-06 |
| *FIR/DDC output* | 5.6190e-06 | 3.6287e-06 |
| *Data in* | 8.9297e-06 | 8.9297e-06 |

A frequency analysis of the output signals was performed using a Discrete Fourier Transform, and the SNR of the $I$ and $Q$ signals were estimated, as shown in figure 7. There are low noises in DDC outputs because the DDS frequency was not precisely the carrier frequency used in the golden model, that created the input signals.

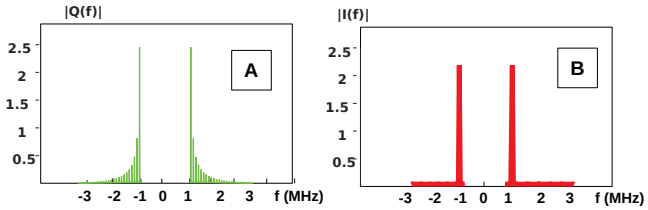



Figure 7. Frequency analysis of output signals. A: Quadrature (Q) with $SNR_{dB} = 54.95$ . B: in-Phase (I) with $SNR_{dB} = 50.24$.

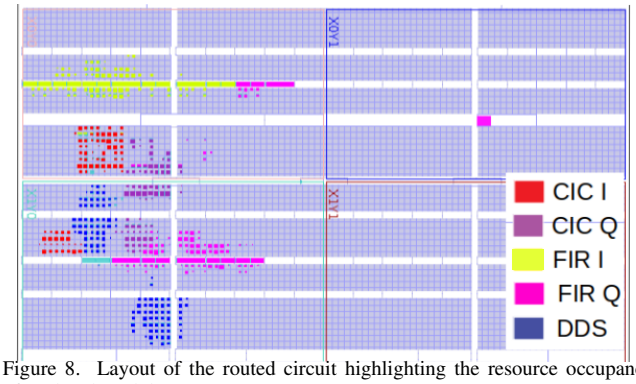



Figure 8. Layout of the routed circuit highlighting the resource occupancy of each submodule.

Table III shows the resource consumption of the proposed down-converter, which was implemented using pipeline architecture. The DDS has the most significant usage of LUTs (2.1%) because of the large number of samples that the phase-amplitude converter has to storage. All the stages of the CIC and FIR filters are registered, thus consuming 450 and 440 slices registers, respectively. The mixers use two DSP blocks, whereas the FIRs use one DSPs for each tap, allowing an output rate of one sample per clock cycle. After some synthesis optimizations, the DDC consumes a total of 16 DSP blocks.

Table III
RESOURCES CONSUMPTION (XILINX XC7Z010 DEVICE)

| Architecture | *Slice LUTs (Max:17600)* | *DSPs (Max:80)* | *Slice Regs (Max:35200)* |
|---|---|---|---|
| **DDC** | 860 (4.89%) | 16 (17.52%) | 931 (2.64%) |
| **DDS** | 372 (2.11%) | - | 39 (0.11%) |
| **Mixers** | 20 (0.12%) | 2 (0.03%) | 2(0.03%) |
| **CICs** | 278 (1.56%) | - | 450 (1.28%) |
| **FIRs** | 196 (1.2%) | 14 (17.50%) | 440 (1.26%) |

Figure 8 shows the layout of the circuit, which demonstrates that the proposed down-converter was effectively mapped on the Xilinx Zynq 7010 device. In terms of execution time, the proposed DDC has a latency of 16 clock cycles that, operating at a clock frequency of 100MHz, represents 160 ns. Since the DDC was implemented using a pipeline architecture, after the initial latency, the output rate is one sample per clock cycle, corresponding to 1MSPS. Figure 9 reports the estimated power consumption of the circuit (low-level confidence). A total of 0.213 Watts is dissipated by the DDC, 40% for static power, and 60% of dynamic power, mainly consumed by logic and signals.

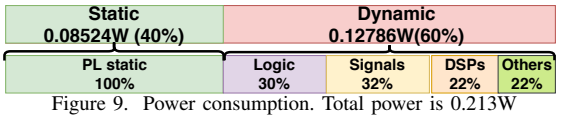



Figure 9. Power consumption. Total power is 0.213W

## V. DISCUSSION AND CONCLUSIONS

This paper proposes a parameterizable FPGA architecture of a Digital Down Converter (DDC) for satellite channels. The proposed DDC uses a DDS and cascaded CIC and FIR filters with low latency and hardware resources consumption. The architecture is parameterized in terms of frequency resolution, word-width, sample size of the DDS, the order of CIC and FIR filters, and the order of the decimator stages. A case study for the DVB-S2X standard was implemented on a Xilinx Zynq 7010 device achieving 16 clock cycle latency, 1MSPS output rate, and a consumption of 880 slice LUTs, 931 slice registers, and 16 DSP blocks. The performance was evaluated by numerical simulations using input signals with frequency and phase noise, demonstrating the correctness of the proposed parameterized DDC model, operating at a clock frequency of 100MHz, and up to 10MHz bandwidth.

The proposed DDC architecture uses less slices registers as compared with [7], [4], [5], [6]. It consumes the same number of DSP blocks and around 300 more slice registers as compared with [9], which makes use of a CORDIC processor instead of a DDS mapped on LUTs. Future works will integrate the DDC with symbol timing recovery and coarse frequency correction modules for the DVB-S2X standard.

## REFERENCES

[1] Willems, Koen, DVB-S2X Demystified, Business Insight on DVB-S2X, Newtec, p. 1-11, 2014.
[2] Hollis T. Weir R. The Theory of Digital Down Conversion, 2003.
[3] EUROPEAN TELECOMMUNICATIONS STANDARDS INSTITUTE. ETSI EN 301 545-2 V1.2.1: Digital Video Broadcasting (DVB); Second Generation DVB Interactive Satellite System(DVB- RCS2); Part 2 Lower Layers for Satellite Standard. ETSE, 2014.
[4] Savvopoulos, Panayiotis Antonakopoulos, Theodore. An IF digital down-converter for software radio DVB-S2 receivers, IEEE Int. Conf. on Electronics, Circuits and Systems, p. 1043-1046, 2007.
[5] Luo, F.-L. Digital front-end in wireless communications and broadcasting: circuits ad signal processing. Cambridge University Press, 2011.
[6] Obradović, Vuk Okiljevic, Predrag Kozic, Nadica Ivkovic, Dejan. Practical implementation of digital down conversion for wideband direction finder on FPGA. Scientific Technical Review. vol. 66, p. 40-46, 2016.
[7] Girau, G. Martina, Maurizio Molino, Andrea Terreno, A. Vacca, Fabrizio. FPGA digital down converter IP for SDR terminals, IEEE Int. Conf. on Signals, Systems and Computers, p. 1010-1014, vol. 2, 2002.
[8] Changrui, Wu et al. Design and FPGA Implementation of Flexible and Efficiency Digital Down Converter. IEEE Int, Conf. on Signal Processing, p. 438-441, 2010.
[9] Datta, D. Mitra P., and Dutta, H. FPGA implementation of high performance digital down converter for software defined radio, Microsystems Technologies, Aug, 2019.
[10] Liu, X., Yan, Xin-Xin, Wang Ze-Ke, Deng, Qing-Xu, Design and FPGA Implementation of a Reconfigurable Digital Down Converter for Wideband Applications, IEEE Trans. on VLSI Systems, vol. 25, 12, p. 3548-3552, 2017.
[11] Lathi, B. P., & Ding, Z. (2019). Modern digital and analog communication systems. New York: Oxford University Press. DVB-S2".
[12] MURPHY, Eva; SLATTERY, Colm. All About Direct Digital Synthesis. Ask The Application Engineer, Analog Dialogue, v.38, p.1-5, 2004.
[13] Casini, Enrico et al. DVB-S2 modem algorithms design and performance over typical satellite channels. Int. J. Satellite Communications Networking,p. 281-318, 2004.