UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PRÓ-REITORIA DE GRADUAÇÃO
DEPARTAMENTO DE ENGENHARIAS
CURSO DE BACHARELADO EM ENGENHARIA ELÉTRICA

JOERDSON TIAGO BATISTA DA SILVA

DESENVOLVIMENTO DE UM MODEM MULTIMODO INTEGRADO EM PROCESSO CMOS DE 130 NANÔMETROS UTILIZANDO FLUXO DE CÓDIGO ABERTO

CARAÚBAS/RN
2023

JOERDSON TIAGO BATISTA DA SILVA

DESENVOLVIMENTO DE UM MODEM MULTIMODO INTEGRADO EM PROCESSO CMOS DE 130 NANÔMETROS UTILIZANDO FLUXO DE CÓDIGO ABERTO

Monografia apresentada à Universidade Federal Rural do Semi-Árido, como exigência final para obtenção do título de Bacharel em Engenharia Elétrica

Orientador: Prof. Dr. Francisco de Assis Brito Filho

CARAÚBAS/RN

2023

S586d Silva, Joerdson Tiago Batista da.
Desenvolvimento de um modem multimodo integrado em processo CMOS de 130 nanômetros utilizando fluxo de código aberto / Joerdson Tiago Batista da Silva. – 2023.
47 f. : il.

Orientador: Francisco de Assis Brito Filho.
Monografia (graduação) – Universidade Federal Rural do Semi-árido, Curso de Engenharia Elétrica, 2023.

1. ASIC. 2. Modem multimodo. 3. Código aberto. 4. CMOS 130 nanômetros. I. Brito Filho, Francisco de Assis, orient. II. Título.

Ficha catalográfica elaborada por sistema gerador automáto em conformidade com AACR2 e os dados fornecidos pelo) autor(a).
Biblioteca Campus Caraúbas
Bibliotecária: Dalvanira Brito Rodrigues
CRB: 15/700

JOERDSON TIAGO BATISTA DA SILVA

DESENVOLVIMENTO DE UM MODEM MULTIMODO INTEGRADO EM PROCESSO CMOS DE 130 NANÔMETROS UTILIZANDO FLUXO DE CÓDIGO ABERTO

Monografia apresentada à Universidade Federal Rural do Semi-Árido, como exigência final para obtenção do título de Bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Francisco de Assis Brito Filho

Defendida em: 11 / 10 / 2023.

BANCA EXAMINADORA

FRANCISCO DE ASSIS BRITO FILHO [redacted]
Assinado de forma digital por FRANCISCO DE ASSIS BRITO FILHO:[redacted]
Dados: 2023.10.17 13:40:09 -03'00'

Prof. Dr. Francisco de Assis Brito Filho
Presidente

gov.br Documento assinado digitalmente
FRANCISCO JOSE TARGINO VIDAL
Data: 16/10/2023 12:08:51-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Franciso José Targino Vidal
Membro Examinador

gov.br Documento assinado digitalmente
JOSE GARIBALDI DUARTE JUNIOR
Data: 16/10/2023 09:13:34-0300
Verifique em https://validar.iti.gov.br

Me. José Garibaldi Duarte Júnior
Membro Examinador

Dedico este trabalho à memória do meu irmão, Gerdeson Nielyton, e do meu avô, Joaquim de Azevêdo, que sempre me incentivaram durante minha formação. A vocês, dedico meu sucesso acadêmico.

AGRADECIMENTOS

Agradeço, em primeiro lugar, a Deus, que permitiu a realização dos meus objetivos ao longo de todos os anos de estudo.

Agradeço a Jéssica Murielly, minha querida irmã, por todo o apoio durante a minha formação. Seu suporte incondicional possibilitou a realização dos meus sonhos acadêmicos.

Agradeço imensamente aos meus pais, Raimundo de Faria e Jailma Batista, pelos sacrifícios e incentivos durante a formação. Aos meus familiares, Francisca Batista, Maria Santana, Jailson Batista, Francisco de Assis e Jacson Soares, agradeço pelo carinho ao longo da minha jornada

À minha namorada, Beatriz Karine, agradeço por todo o companheirismo e incentivo durante os momentos da graduação.

Ao meu orientador, Francisco Brito, agradeço pelas orientações e contribuições durante todos os anos de pesquisa em conjunto, inclusive neste trabalho.

Aos meus bons amigos(as) Mayra, Erika, Lorena, Junior, Matheus, Kaio, Erica, Carol, Izaac e Vinicius, agradeço pelo companheirismo e troca de experiências que me permitiram crescer.

RESUMO

A crescente expansão da tecnologia de código aberto no desenvolvimento de ASICs orientado por *software* vem impactando diretamente academias e indústria. No cenário atual, o aumento na demanda de chips baseados em processos de fabricação de alto custo, torna o acesso às ferramentas de código aberto uma solução essencial para abordagens econômicas e acessíveis. Nesse contexto, o presente estudo aborda o desenvolvimento de um circuito integrado usando ferramentas automatizadas em código aberto e processo de fabricação CMOS aberto de 130 nanômetros. O projeto foi baseado em um SDR modem multimodo operante nos esquemas de modulação digital em amplitude (ASK), em frequência (FSK) e em fase (PSK). Para alcançar esse propósito, foram utilizados os conceitos de linguagem de descrição de *hardware* para implementar as etapas do fluxo do design ASIC com ferramentas de código aberto, no sistema operacional *Linux* e na plataforma *GitHub*. Os arquivos de *layout* do circuito integrado foram gerados com as regras que atendem as especificações do processo, com o *layout* final do chip apresentando dimensões de 160*x*200 μm e consumo de 1.19 mW.

Palavras-chave: ASIC. Modem multimodo. Código aberto. CMOS 130 nanômetros.

ABSTRACT

The growing expansion of open source technology in the software driven development of ASICs has directly impacted academia and industry. In the current scenario, the increase in demand for chips based on high cost manufacturing processes makes access to open source tools an essential solution for economical and accessible approaches. In this context, the present study addresses the development of an integrated circuit using open source automated tools and an open 130 nanometers CMOS manufacturing process. The project was based on a multimode SDR modem operating in digital amplitude (ASK), frequency (FSK) and phase (PSK) modulation schemes. To achieve this purpose, hardware description language concepts were used to implement the steps of the ASIC design flow with open source tools, on the Linux operating system and on the GitHub platform. The integrated circuit layout files were generated with rules that meet the process specifications, with the final chip layout having dimensions of 160*x*200 µm and consumption of 1.19 mW.

Keywords: ASIC. Multimode modem. Open source. CMOS 130 nanometers.

LISTA DE FIGURAS

Figura 1 – Arquitetura genérica de um rádio definido por *software*. 16

Figura 2 – Diagrama de bloco de um modulador. 18

Figura 3 – Portadora senoidal analógica gerada por uma LUT. 19

Figura 4 – Sinais de modulação digital para o esquema ASK. 19

Figura 5 – Sinais de modulação digital para o esquema FSK. 20

Figura 6 – Sinais de modulação digital para o esquema PSK. 21

Figura 7 – Metodologia de designs para ASICs. 22

Figura 8 – Arquitetura modem multimodo baseado em SDR. 26

Figura 9 – Fluxo de desenvolvimento ASIC em código aberto. 27

Figura 10 – Etapas de desenvolvimento do processo ASIC livre. 28

Figura 11 – Comandos de inicialização do fluxo *OpenLane*. 31

Figura 12 – Principais processos do fluxo *OpenLane*. 31

Figura 13 – Diagrama de blocos do modem multimodo após alterações. 33

Figura 14 – *Layout* GDSII visualizado no *Klayout* (sem algumas camadas). 36

Figura 15 – *Layout* GDSII visualizado no *Klayout* (com todas as camadas). 36

Figura 16 – Sinais gráficos no *GTKw*ave (a) estímulos de entrada (b) esquema ASK. 38

Figura 17 – Sinais gráficos no GTKwave (a) esquema FSK (b) esquema PSK. 39

Figura 18 – *Layout* 2D representando o chip ASIC do modem. 40

Figura 19 – Modelo do chip ASIC na placa de suporte e PCB. 41

LISTA DE TABELAS

Tabela 1 – Bibliotecas padronizadas e personalizadas de HDL. 29

Tabela 2 – Entradas e saídas do *layout* do modem de 9 bits. 35

Tabela 3 – Parâmetros do *layout* ASIC do modem de 9 bits..... 35

Tabela 4 – Entradas e saídas do chip modem de 7 bits. 37

Tabela 5 – Resultados comparativos entre as implementações. 42

LISTA DE ABREVIATURAS

ADC	*Analog-to-Digital Converter*
AM	*Amplitude Modulation*
ASIC	*Application-Specific Integrated Circuit*
ASK	*Amplitude Shift Keying*
BPSK	*Binary Phase Shift Keying*
CI	Circuito Integrado
DCA	*Digital-to-Analog Converter*
DSP	*Digital Signal Processing*
EDA	*Electronic Design Automation*
FPGA	*Field-Programmable Gate Array*
FSK	*Frequency Shift Keying*
GDSII	*Graphic Data System II*
HDL	*Hardware Description Language*
IoT	*Internet of Things*
IEEE	*Institute of Electrical and Electronics Engineers*
LUT	*Lookup Table*
PDK	*Process Design Kit*
PSK	*Phase Shift Keying*
QPSK	*Quadrature Phase Shift Keying*
RTL	*Register Transfer Level*
SDR	*Software Defined Radio*
VHDL	*VHSIC Hardware Description Language*
VLSI	*Very Large-Scale Integration*

SUMÁRIO

1 INTRODUÇÃO 14

1.1 OBJETIVOS 15

1.2 ORGANIZAÇÃO DO TRABALHO 15

2 FUNDAMENTAÇÃO TEÓRICA 16

2.1 RÁDIO DEFINIDO POR SOFTWARE 16

2.2 SISTEMA DE MODULAÇÃO DIGITAL 18

2.2.1 Modulação ASK 19

2.2.2 Modulação FSK 20

2.2.3 Modulação PSK 21

2.3 CIRCUITO INTEGRADO DE APLICAÇÃO ESPECÍFICA 22

2.4 LINGUAGEM DE DESCRIÇÃO DE HARDWARE 23

2.5 FERRAMENTAS EDA DE CÓDIGO ABERTO 24

2.5.1 Ferramenta GHDL 24

2.5.2 Ferramenta Yosys 25

2.5.3 Ferramenta OpenLane 25

2.5.4 Ferramenta Icarus Verilog 25

2.6 TRABALHOS RELACIONADOS 26

3 DESENVOLVIMENTO DO DESIGN EM ASIC 27

3.1 ASPECTOS METODOLÓGICOS 27

3.1.1 Modificações na arquitetura do modem multimodo 29

3.1.2 Fluxo de design ASIC no Linux 30

3.1.3 Fluxo de design ASIC no GitHub e evento TinyTapout 32

3.2 RESULTADOS E DISCUSSÕES 33

3.2.1 Simulação no GHDL 34

3.2.2 Síntese no Yosys 34

3.2.3 Fluxo no OpenLane 34

3.2.4 Layout físico GDSII 35
3.2.5 Design ASIC no GitHub 37
3.2.6 Fabricação do chip modem 41
3.2.7 Análises comparativas 42
4 CONSIDERAÇÕES FINAIS 43
4.1 TRABALHOS FUTUROS 44
REFERÊNCIAS 45

1 INTRODUÇÃO

O interesse crescente no processamento de sinais digitais tem impulsionado o aumento das aplicações com técnicas de modulação digital (OLIVEIRA, 2014). Nesse cenário, o Rádio Definido por *Software* (SDR) surge como uma abordagem nova para atender às demandas em constante evolução do ambiente de comunicação digital. Nos SDRs, os componentes do rádio digital são apresentados em domínio de *software*, contrastando com os rádios tradicionais que possuem componentes baseados em *hardware* (ROBERT, 2015).

Essa adaptabilidade permite ajustes nos parâmetros através de *software*, simplificando atualizações e manutenções para atender às diversas necessidades do rádio. Existem diferentes soluções que implementam as funções do SDR, como as Matrizes de Portas Programáveis em Campo (FPGA) e os Circuitos Integrados de Aplicação Específica (ASIC) (LALGE, 2015). A escolha desses *hardwares* genéricos depende da flexibilidade e do desempenho proposto para a aplicação, com cada opção oferecendo vantagens específicas.

Nos últimos anos, os *softwares* de código aberto têm impulsionado o desenvolvimento de circuitos integrados (CI) na indústria dos semicondutores através de ferramentas de código aberto (HUANG, 2019). Essas ferramentas EDA (*Electronic Design Automation*) tem apoiado o design de circuitos integrados com um fluxo digital flexível e totalmente automatizado para todas as etapas necessárias (HESHAM, 2021). Além disso, elas têm se tornado essenciais para reduzir os custos de produção, permitindo que engenheiros e designers aproveitem os recursos de código aberto disponíveis de forma colaborativa.

Para realizar esses projetos de CIs, um dos principais fatores a serem considerados são as regras de fabricação, conhecidas como kit de design de processo (PDK). Os PDKs possuem processos rigorosos de licenciamento e custos extremamente altos, que os torna inacessíveis a maioria das pessoas e empresas (KUMAR et al., 2021). Com a recente introdução do primeiro *open*PDK pela *Google* e *SkyWater Tecnology*, conhecido como *SkyWater* de 130 nanômetros, o cenário se abriu para os circuitos integrados de aplicações especificas com desenvolvimento em código aberto (EDWARDS, 2020). Essa iniciativa tem impulsionado significativamente a democratização do design de CIs, com inovações abertas na comunidade de semicondutores.

Nesse campo, as técnicas de modulação digital ampliam a Internet das Coisas (IoT) de forma significativa, através de aplicações que exigem flexibilidade e eficiência de recursos. O design de CIs possibilita implementar essas demandas específicas, com *hardwares* otimizados e precisos para o processamento de sinais digitais.

Esse novo cenário impacta os conceitos tradicionais associados aos chips IoT, através de baixos custos de fabricação, abordagens que melhoram a eficiência energética e dimensões reduzidas, fundamentais para aplicações associadas a processamentos digitais que exigem alto desempenho, como os transceptores baseado em SDR. Essa tendência tecnológica de soluções versáteis e ágeis impulsiona a próxima geração de dispositivos conectados.

1.1 OBJETIVOS

O objetivo geral deste trabalho é desenvolver um design de CI em processo CMOS de 130 nanômetros, a partir de um modem multimodo baseado em SDR, que opera nos esquemas básicos de modulação digital, utilizando as tecnologias de código aberto.

Além disso, definem-se os seguintes pontos como objetivos específicos:

- Realizar revisões abrangentes em literaturas relacionadas as etapas de síntese ASIC, arquiteturas SDR modem multimodo e tecnologias de código aberto;
- Implementar modificações de sintaxes nas arquiteturas que compõe o modem;
- Realizar verificações funcionais nas arquiteturas em VHDL modificadas;
- Implementar a etapa de síntese ASIC utilizando ferramentas de código aberto;
- Analisar os arquivos de *layout* e design, gerados no processo de síntese ASIC;
- Submeter os arquivos do design para processo de fabricação.

1.2 ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado em mais três capítulos, conforme descrito a seguir.

- No capítulo 2, são descritos os principais conceitos do estudo proposto, assim como os trabalhos correlatos utilizados como base teórica para a concepção;
- No capítulo 3, as etapas desenvolvidas no trabalho são apresentadas, bem como a metodologia utilizada descrita de forma detalhada. Também é exposto neste capítulo os resultados obtidos com os métodos implementados;
- No capítulo 4, são apresentadas as considerações acerca do trabalho realizado, indicando as principais conclusões observadas nos resultados, juntamente com as sugestões para os trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

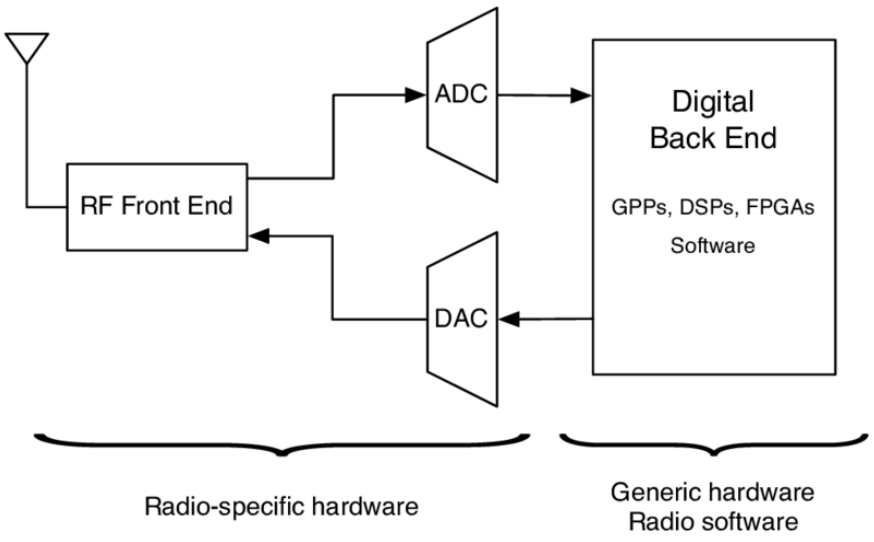
Nas próximas seções, serão abordados os conceitos essenciais para o entendimento do trabalho. Serão explanadas as definições sobre SDR seguido pelos processos de modulação de sinais. Além disso, será dada ênfase as definições sobre ASIC, seguido da linguagem VHDL e os conceitos sobre ferramentas de código aberto. Por último, serão discutidos os trabalhos que serviram de embasamento para realização desse estudo.

2.1 RÁDIO DEFINIDO POR SOFTWARE

Um Rádio Definido por *Software* pode ser definido como um sistema de comunicação, que emprega módulos baseados em *software* modificáveis ao invés dos módulos de *hardware* tradicionais (amplificadores, conversores moduladores e demoduladores) para implementar as funções básicas de rádio e processamentos de sinais. Portanto, essa arquitetura reconfigurável fornece soluções eficientes com uma quantidade mínima de *hardware*, possibilitando atender diferentes aplicações utilizando dispositivos genéricos de *hardware* para receber arquiteturas via *software* (MITOLA, 1992). Essa tecnologia surge com filosofias flexíveis e eficientes, que está sendo impulsionada pela mudança nos paradigmas dos sistemas de comunicação sem fio.

A arquitetura básica de um SDR baseado na funcionalidade de transmissão e recepção, possui as características definidas na Figura 1.

Figura 1 – Arquitetura genérica de um rádio definido por *software*.



Fonte: Moskal (2011).

O módulo de *hardware* específico corresponde ao *front end* analógico, que lida com as funções de transmissão e recepção do sistema de comunicação de radiofrequência. Dentro dos componentes do *hardware* específico do rádio, é comum observar (mas nem sempre, depende da aplicação do SDR) o Conversor Análgico-Digital (ADC) e o Conversor Digital-Analógico (DCA), responsáveis por converter os sinais enviados e recebidos das antenas de transmissão e recepção do sistema SDR (TUTTLEBEE, 2003).

O bloco de *hardware* genérico corresponde ao *back end* digital, que lida com a etapa de processamento de sinal digital após a conversão do sinal analógico para o domínio digital. O *back end* desempenha um papel fundamental no SDR, pois é nele que ocorre as funções de geração de sinais, modulação, demodulação, correções de erro e filtragem, podendo ser esses elementos baseados em *software* (ou parte em *hardware*, depende da aplicação desenvolvida) e implementados utilizando apenas um *hardware* genérico (TUTTLEBEE, 2003).

Embora o conceito de SDR tenha surgido nos anos 90, apenas recentemente, com o rápido avanço da eletrônica digital e o aumento tecnológico associado, essa tecnologia ganhou notoriedade, tornando-se amplamente popular em diversas aplicações de radiocomunicação e estabelecendo um domínio que continua a crescer com a necessidade por sistemas eficientes e flexíveis. O potencial econômico relacionado aos custos também desempenha um papel de grande relevância, uma vez que a substituição de componentes físicos por soluções baseadas em *software* resulta em reduções substanciais nos custos de implementação e manutenção dos sistemas de rádio (FORUM, 2017).

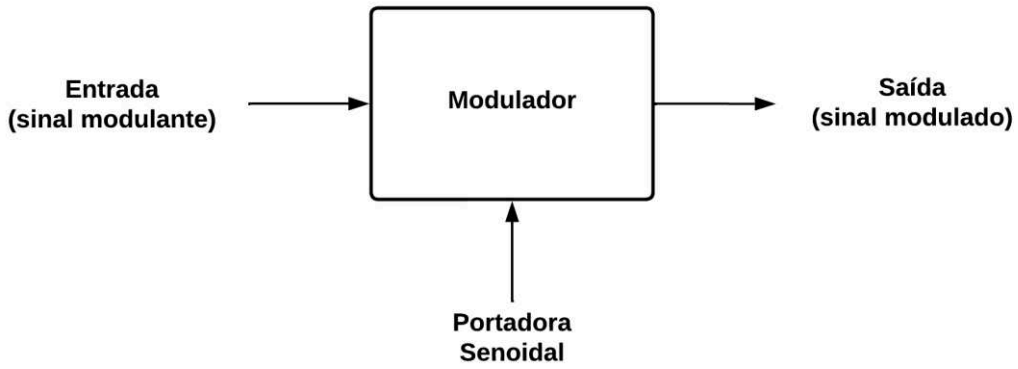
Essa crescente busca por soluções eficientes e práticas no campo de radiocomunicação tem levado empresas, centros de pesquisas e organizações a apoiarem o uso e o estudo dessas tecnologias SDR na comunicação sem fio. A *Wireless Innovation Forum*, uma corporação que é composta por mais de 100 membros pelo mundo, destaca-se nesse quesito, com incentivo na utilização dos SDR nos ambientes militares, industriais e civis (QUISPE, 2015).

Essa tecnologia revoluciona a abordagem convencional, ao assumir responsabilidades anteriormente atribuídas aos *hardwares* específicos nos sistemas de rádio. Essa flexibilidade e eficiência elevada não só viabilizam uma adaptação ágil a padrões de modulação digital, mas também a consolidam como uma solução inovadora e eficaz para os desafios dos sistemas de comunicação digitais modernos.

2.2 SISTEMA DE MODULAÇÃO DIGITAL

O modulador é um componente crucial em sistemas de comunicação, desempenhando a função de realizar o processo de modulação no transmissor, sendo responsável por alterar as características do sinal portador com base no sinal modulante (HAYKIN; MOHER, 2009). Na Figura 2 é apresentado a arquitetura de um modulador, no qual o sinal modulante representa a entrada, geralmente contendo a informação a ser transmitida, que quando combinado ao sinal da portadora, gera a informação a ser transmitida, o sinal modulado.

Figura 2 – Diagrama de bloco de um modulador.



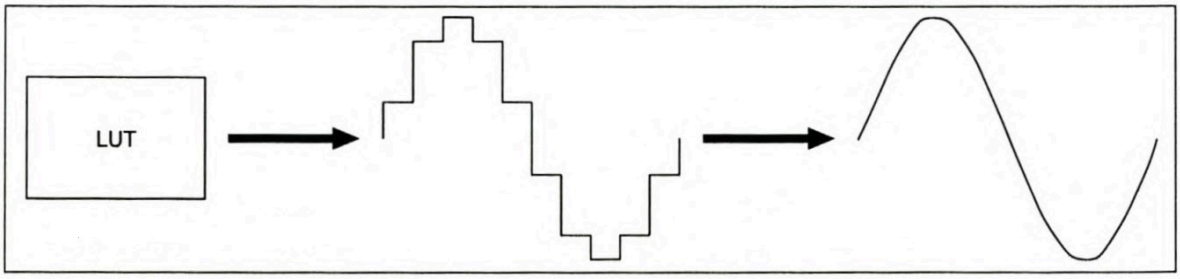
Fonte: Adaptado de Haykin e Moher (2009).

Para implementar um transmissor de rádio com um sistema SDR, é necessário utilizar um sinal portador, responsável por transportar as informações da mensagem. Uma abordagem eficiente seria a geração do sinal portador diretamente no domínio digital, pois isso permitiria maior flexibilidade e controle preciso sobre as características do sinal (CRONJE, 2004).

Para realizar isso, utiliza um conjunto de valores que representam a amplitude do sinal em vários pontos ao longo do tempo. Esse sinal torna-se digitalizado e em formato senoidal a partir de um sinal de *clock*, que sincroniza e organiza o sinal portador. Esse processo, permite uma representação precisa e eficiente do sinal analógico no domínio digital, facilitando assim sua manipulação, processamento e transmissão em sistemas baseados em SDR.

Uma *Lookup Table* (LUT) realiza esse processo armazenando todos os valores do sinal digitalizado em uma matriz, funcionando como uma espécie de memória que contém amostras representativas da amplitude do sinal em diferentes momentos do tempo (CRONJE, 2004). A Figura 3 ilustra a representação da onda senoidal gerada por uma arquitetura LUT.

Figura 3 – Portadora senoidal analógica gerada por uma LUT.



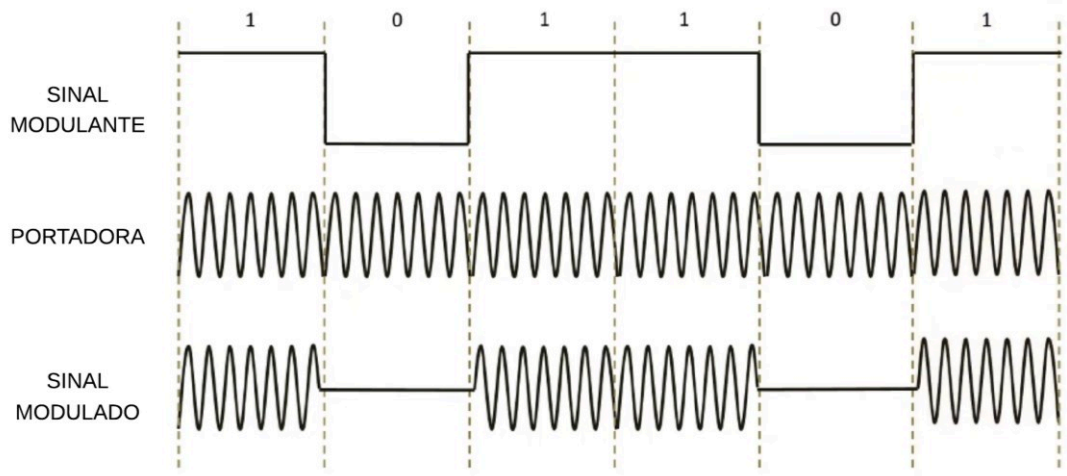
Fonte: Cronje (2004).

As técnicas de modulação digital são classificadas com base no tipo de sinal modulado e nas características específicas da modulação empregada. Existem três técnicas fundamentais para modular uma portadora de onda senoidal, sendo alterando a amplitude, a frequência ou a fase, com estas formas básicas de modulação utilizadas atualmente com os sinais digitais.

2.2.1 Modulação ASK

No esquema de modulação ASK (*Amplitude Shift Keying*), a amplitude da portadora é alterada de acordo com o sinal de entrada digital (sinal modulante). Essa modulação é análoga à *Amplitude Modulation* (AM), com diferença no sinal modulante, sendo analógico para AM, e do tipo digital (fluxo de bits), para a modulação digital (NETO, 2015). A Figura 4 ilustra um esquema genérico de modulação ASK.

Figura 4 – Sinais de modulação digital para o esquema ASK.



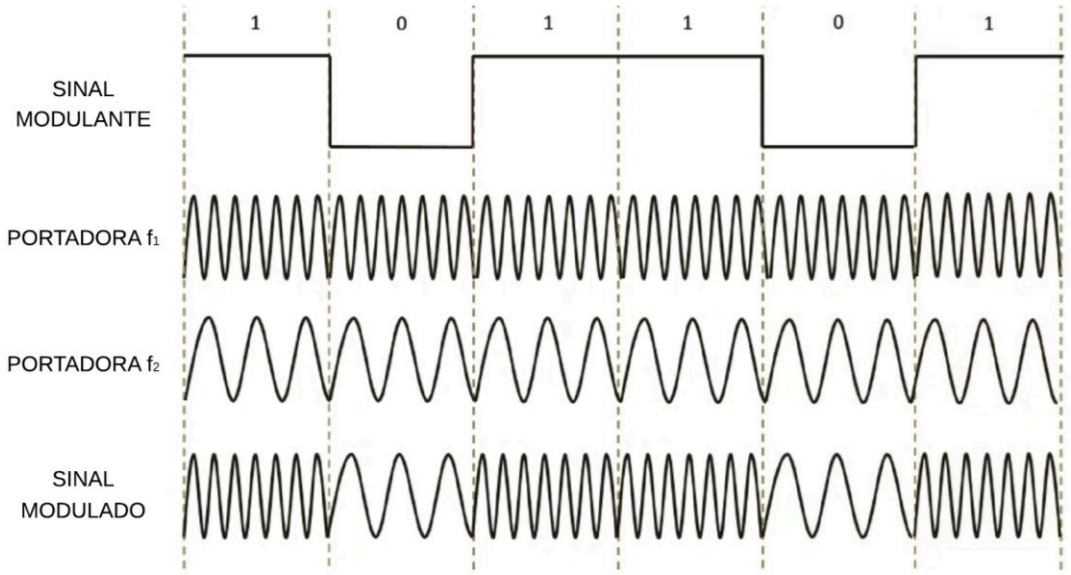
Fonte: Adaptado de Sayara (2020).

Essa técnica consiste em representar dados binários através de alterações na amplitude de uma onda portadora, ou seja, quando um bit de informação é igual a 1, a amplitude da onda portadora é alterada para um determinado valor, e quando o bit é 0, a amplitude assume outro valor diferente, mantendo-se constantes a frequência e a fase (NETO, 2015).

2.2.2 Modulação FSK

Na modulação FSK (*Frequency Shift Keying*), a frequência do sinal portador varia de acordo com as mudanças do sinal digital na onda modulante. Nesse esquema, dois estados de frequência são utilizados, com uma frequência para representar o bit 1 e outra frequência para representar o bit 0. A informação codificada é então usada para modular a frequência da onda portadora (NETO, 2015). A Figura 5 ilustra um esquema genérico de modulação FSK.

Figura 5 – Sinais de modulação digital para o esquema FSK.



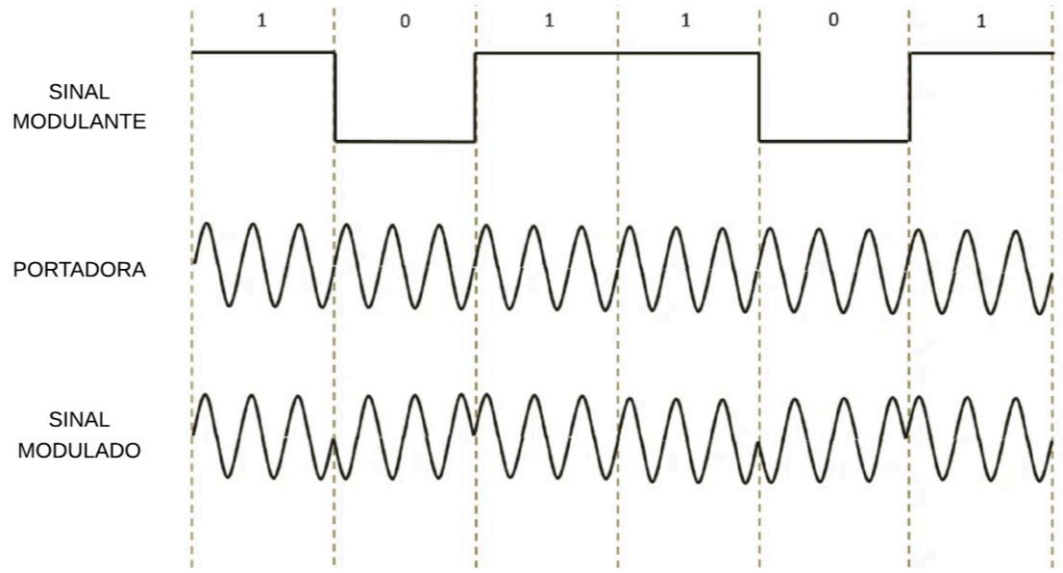
Fonte: Adaptado de Sayara (2020).

Quando o bit atual for 1, a frequência da portadora ajusta-se para refletir a frequência associada ao bit 1. Da mesma forma, quando o bit é 0, a frequência da portadora é modificada para representar a frequência correspondente ao bit 0. Nessa modulação, a amplitude e a fase são mantidas constantes (NETO, 2015).

2.2.3 Modulação PSK

A técnica de modulação PSK (*Phase Shift Keying*) envolve o chaveamento da fase da onda portadora para representar as informações digitais do sinal modulador. Nesse método, a frequência e a amplitude da portadora permanecem constantes, com a informação transmitida por meio de variações na fase em sincronia com a portadora (NETO, 2015). A Figura 6 traz a forma de onda de um esquema de modulação PSK.

Figura 6 – Sinais de modulação digital para o esquema PSK.



Fonte: Adaptado de Sayara (2020).

A modulação PSK pode ser divido em dois tipos, sendo o BPSK (*Binary Phase Shift Keying*) e o QPSK (*Quadrature Phase Shift Keying*). No primeiro método, são utilizadas duas fases distinta com separação de 180°, sendo essas fases expressas simbolicamente como 0 e 1. Dessa forma, quando o bit a ser transmitido é 0, a fase da onda portadora é mantida inalterada. Quando o bit é 1, a fase da onda portadora é invertida em 180°, criando uma transição de fase visível na forma de onda do sinal modulado (NETO, 2015).

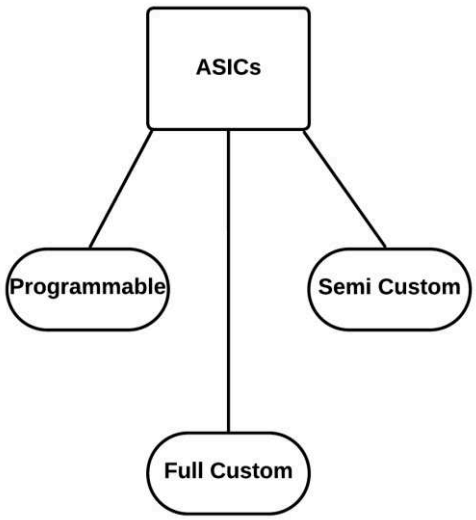
No método QPSK, quatro fases diferentes são utilizadas para representar símbolos de dois bits de dados, com quatro mudanças de fases (0°, 90°, 180° e 270°). As variações dessas fases permitem a transmissão eficiente de informações, dobrando a taxa de transferência dos dados em comparação com esquemas que utilizam apenas uma portadora. Esse método utiliza dois sinais de portadoras, que ficam 90° defasadas entre si (NETO, 2015).

2.3 CIRCUITO INTEGRADO DE APLICAÇÃO ESPECÍFICA

Um circuito integrado de aplicação específica, conhecido com ASIC, é projetado para atender às condições de tarefas específicas. Diferente dos CIs tradicionais, os ASICs possuem fabricação focada, otimizando desempenho e eficiência para as aplicações (DE MELO, 2004). Esses CIs personalizados são desenvolvidos utilizando tecnologia de semicondutores, onde os componentes são integrados em um único chip, que pode conter milhões de transistores.

Os projetos em ASICs são baseados em metodologias *bottom up* ou *top down*, sendo a primeira, relacionada às pequenas partes do projeto, como o desenho de transistores nas portas lógicas ou células analógicas, sendo amplamente utilizado nos projetos de CIs *full custom*. Na metodologia *top down*, a descrição do ASIC é preferencialmente realizada utilizando o uso de linguagens de descrição de *hardware* (HDL), embora a descrição esquemática também possa ser utilizada nesse processo (RIBAS et al., 2001). Essa abordagem *semi custom* possibilita um processo automatizado, com eficiência aprimorada e flexibilidade nas adaptações específicas, proporcionando uma abordagem dinâmica ao desenvolvimento de ASIC. A Figura 7 apresenta as principais metodologias para projetos ASICs.

Figura 7 – Metodologia de designs para ASICs.



Fonte: Autoria própria (2023).

A abordagem *programmable* geralmente está associado a utilização de um FPGA para emular um design ASIC, que é reprogramável, antes de finalizar e fabricar o design, ajudando a reduzir os riscos e os custos do desenvolvimento do CI personalizado.

Em circuitos digitais, a descrição passa por etapas de otimização e mapeamento lógico para células disponíveis nas bibliotecas. Esta síntese lógica permite optar pela construção que melhor atenda aos requisitos de desempenho, consumo de energia e área ocupada, garantindo assim uma implementação eficiente e robusta do projeto ASIC (RIBAS et al., 2001).

2.4 LINGUAGEM DE DESCRIÇÃO DE HARDWARE

Desenvolvido pelo departamento de defesa dos Estados Unidos, a linguagem VHDL, que significa *VHSIC Hardware Description Language* (Linguagem de Descrição de Hardware de Circuito Integrado Muito Avançado), surgiu com a necessidade de fornecer uma linguagem de alto nível para descrever o comportamento e a estrutura dos circuitos digitais. Desde então, tornou-se uma linguagem amplamente utilizada nas indústrias de designs de circuitos digitais complexos, como ASICs e FPGAs (D'AMORE, 2000).

Esta linguagem foi padronizada pelo Instituto de Engenheiros Elétricos e Eletrônicos (IEEE) pela primeira vez em 1987, definida como padrão VHDL-1987. Desde então, têm sido realizados aprimoramentos e atualizações para tornar a linguagem mais acessível e adaptável. Nesse processo evolutivo, surgiram dois pacotes padrões essenciais para projetos de *hardware* digital atuais, sendo o IEEE 1164 e o IEEE 1074.3 (CRUZ, 2022). Esses pacotes fornecem as bibliotecas *Std_logic_1164* e *Numeric_std*, que representam expansões significativas para as descrições de circuitos em VHDL, superando as limitações dos pacotes anteriores.

A linguagem VHDL oferece versatilidade notável, permitindo não apenas a descrição de *hardware* para a implementação em chips, mas também a criação de códigos destinados à simulação do comportamento desses circuitos, sendo possível descrever um circuito único de diversas maneiras e com vários graus de abstração (CRUZ, 2022).

Um mesmo circuito pode ser implementado através de estilos distintos de modelagem, como o comportamental e o estrutural. No estilo comportamental, é especificado o circuito em um nível mais abstrato de sistema, usando processos e atribuições de sinais. Por outro lado, no estilo estrutural, adota-se uma abordagem hierárquica, interligando os vários subsistemas para formar o circuito desejado, que representam o topo da hierarquia (CRUZ, 2022).

Essa flexibilidade na representação dos circuitos digitais oferece aos projetistas opções versáteis, que possibilita modificar a descrição projetada conforme as condições da aplicação. Ao adaptar às necessidades específicas da aplicação, essa abordagem proporciona eficiência e inovação para representar as ligações entre os módulos e os componentes dos sistemas.

2.5 FERRAMENTAS EDA DE CÓDIGO ABERTO

O mercado global das EDAs (*Eletronics Designs Automations*) tem sido dominado por três grandes indústrias do setor de semicondutores (*Cadence*, *Synopsys* e *Siemens* EDA), que controlam atualmente mais de 80% do mercado de EDAs (LI, 2022). Essas ferramentas EDAs são indispensáveis para a indústria VLSI (*Very Large Scale Integration*), pois possibilitam a concepção, análise e otimização eficientes de CIs complexos, acelerando significativamente o processo de design e contribuindo para a tecnologia de semicondutores (KHAN, 2021).

As ferramentas EDA representam um conjunto de recursos de *software* empregados no processo de desenvolvimento de circuitos eletrônicos, como o CI. Essas ferramentas oferecem um suporte indispensável para criação, análise e aprimoramento dos projetos com alto número de transistores, que seria impossível de gerenciar sem um processo de automação sofisticado (SCHEFFER et al., 2018). Porém, essas ferramentas comerciais não são acessíveis ao público devido aos custos elevados associados, tornando o acesso restrito a profissionais e instituições financeiramente estruturados.

A democratização do desenvolvimento de circuitos eletrônicos tem se destacado com as iniciativas de código aberto e as ferramentas EDA gratuitas. Essas ferramentas possuem os códigos fontes disponíveis para que a comunidade possa compreender, modificar e aprimorar suas funcionalidades, impulsionando a inovação colaborativa (LUCK et al., 2022).

Em 2020, as empresas *Google* e *SkyWater Technology* lançaram o primeiro PDK com código aberto estabelecido em ASIC, com nó de processo de 130 nm (ROCKET, 2021). Esse processo se refere a ferramentas, arquivos e informações necessários para projetar CIs usando a tecnologia e as regras de fabricação da *SkyWater*, com processo CMOS de 130 nanômetros.

Em conjunto com esse processo PDK, as ferramentas de código aberto são usadas para otimizar e facilitar diversas etapas cruciais do design ASIC. Entre essas ferramentas notáveis estão o *GHDL*, *Yosys* e *OpenLane*.

2.5.1 Ferramenta GHDL

O GHDL é uma ferramenta de código aberto utilizada para simulação de *hardware* em linguagem VHDL, que permite testar e verificar o comportamento de circuitos digitais antes da implementação física. Essa ferramenta suporta várias funcionalidades, incluindo simulação de forma rápida e precisa, depuração eficiente e análise detalhada de sinais, possibilitando que seja possível identificar e corrigir problemas no inicial do design (TGINGOLD, 2020).

2.5.2 Ferramenta Yosys

O *Yosys* é uma ferramenta aberta de síntese *verilog* HDL, que converte linguagens de alto nível, como VHDL, para linguagens de baixo nível equivalente, em formato *verilog*. Essa ferramenta apresenta um conjunto básico de *scripts* de síntese, que desempenham três papéis distintos. Primeiramente, o *front end*, responsáveis por ler os arquivos de entrada, geralmente em formato *verilog*. Em seguida, os *Passes*, que transforma o design na memória, permitindo otimizações e ajustes. Por fim, o *back end*, que grava o design feito na memória em diversos formatos disponíveis, inclusive o *verilog* (WOLF, 2016).

2.5.3 Ferramenta OpenLane

O *OpenLane* é uma ferramenta de código aberto dedicada à síntese física de projetos em RTL (*Register Transfer Level*), capaz de proporcionar fluxos automatizados nas etapas do design de CI de código aberto (GHAZY; SHALAN, 2020). Essa ferramenta visa simplificar e otimizar o complexo processo de design dos chips, permitindo abordagens mais acessíveis. O fluxo abrangente executa as fases de implementação, desde a descrição RTL até a geração dos arquivos GDS (*Graphic Data System*), essenciais para o processo de fundição.

2.5.4 Ferramenta Icarus Verilog

O *Icarus Verilog* é uma ferramenta de código aberto, que compila e simula linguagens de descrição de *hardware verilog*, sendo utilizado principalmente em verificações de circuitos digitais. Ele funciona analisando a sintaxe *verilog* da descrição, verificando as conectividades e gerando um modelo de simulação do circuito digital correspondente, possibilitando validar o funcionamento dos circuitos digitais antes da implementação física (STEVE ICARUS, 2009).

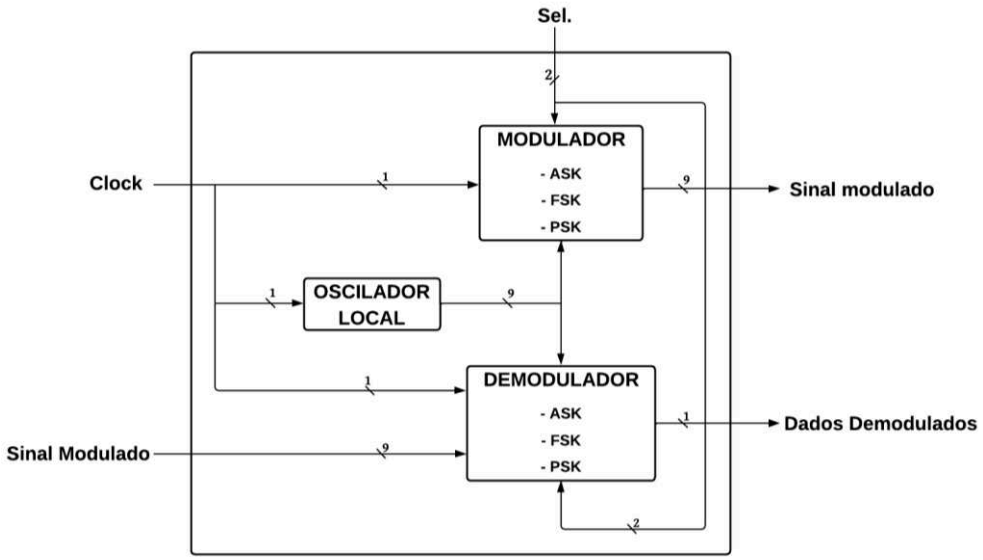
Em conjunto com essa ferramenta, o *GTKWave* é utilizado para depuração de projetos de design digital baseados em *hardware* descritos em *verilog*. O *GTKWave* é uma ferramenta de código aberto que visualiza os sinais digitais, proporcionando uma análise aprofundada dos circuitos implementados (FINKELSTEIN et al., 2011). Essa ferramenta é útil para entender o comportamento dos circuitos digitais projetados e identificar os possíveis problemas durante o desenvolvimento, tornando o processo de depuração mais eficiente e preciso.

2.6 TRABALHOS RELACIONADOS

Os modens desenvolvidos por rádio definido por *software* são tendências recentes nos campos de comunicação sem fio. Esse cenário oferece oportunidades para as comunidades de pesquisas aprimorarem e criarem maneiras mais eficientes de aproveitar a utilização de FPGA e ASIC para implementar módulos SDR (HATAI; CHAKRABARTI, 2009).

No estudo apresentado por Duarte Júnior (2018), um modem baseado em SDR, com as características de modulação e demodulação para amplitude (ASK), frequência (FSK) e fase (PSK) foi implementado em FPGA. Essa flexibilidade proporcionada pelo design com FPGA permitiu uma adaptação eficiente às demandas variáveis dos padrões de comunicação sem fio, enquanto a implementação em VHDL assegurou uma descrição precisa e modular do modem. A arquitetura do trabalho descreve três módulos principais, ilustrado na Figura 8.

Figura 8 – Arquitetura modem multimodo baseado em SDR.



Fonte: Duarte Júnior (2018).

Em estudos realizados por Lalge (2015), foi implementado um modem PSK em FPGA com soluções SDR para reduzir o uso de *hardware* e o consumo de energia, utilizando técnica de reconfiguração parcial. O modem permitiu abordagens dinâmicas e eficientes das variáveis de comunicações modernas, através das técnicas de reconfiguração parcial.

A partir dos estudos descritos, a síntese em ASIC representa um passo significativo em direção às aplicações práticas dessas inovações no cenário de comunicações sem fio.

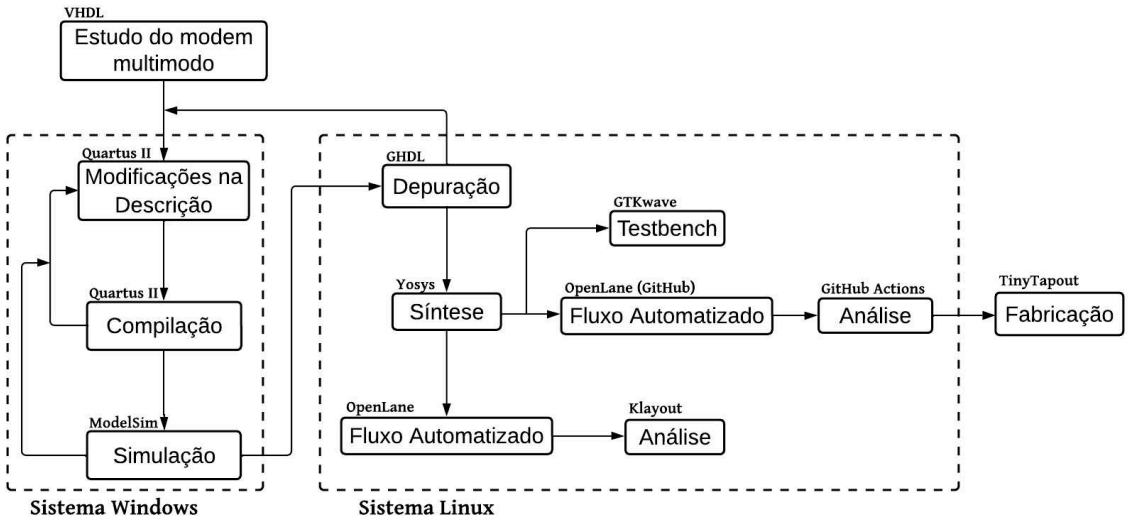
3 DESENVOLVIMENTO DO DESIGN EM ASIC

Neste capítulo, serão detalhadas as abordagens empregadas no design ASIC proposto para um SDR modem multimodo, utilizando ferramentas de código aberto. Além disso, serão apresentados os resultados obtidos por meio desse processo de fluxo livre.

3.1 ASPECTOS METODOLÓGICOS

Este trabalho concentrou os estudos no desenvolvimento de um design ASIC orientado por *software SkyWater* de código aberto, utilizando arquiteturas programáveis e o conceito de rádio definido por *software* para implementar um *layout* físico de ASIC baseado no projeto de modem multimodo construído por Duarte Júnior (2018), que realiza processos de modulação e demodulação nos esquemas ASK, FSK e PSK. O diagrama apresentado na Figura 9, resume as principais etapas utilizadas no desenvolvimento deste trabalho.

Figura 9 – Fluxo de desenvolvimento ASIC em código aberto.



Fonte: Autoria própria (2023).

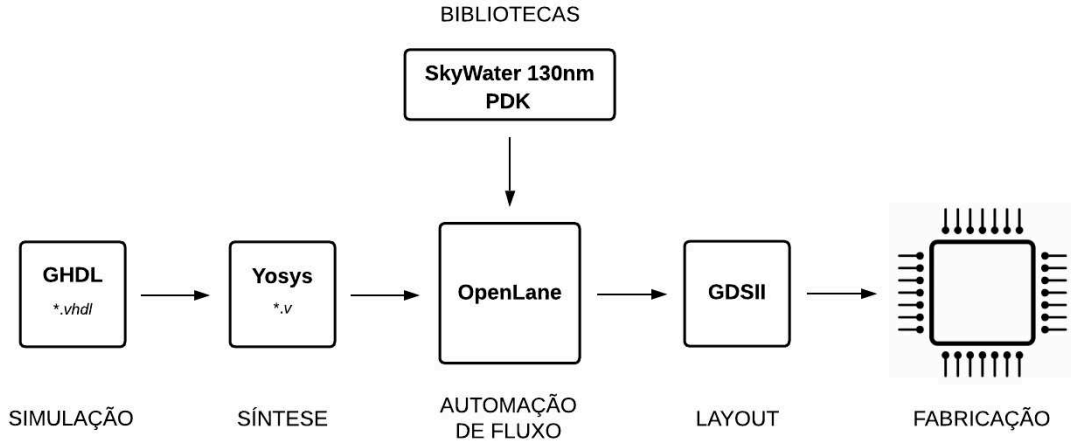
Para implementar o design do CI, foi utilizado o projeto de Duarte Júnior (2018) como ponto de partida no desenvolvimento, estudando as descrições em linguagem de *hardware* do modem e os métodos utilizados de forma detalhada. Nesse contexto, também foram estudados os conceitos de rádio definido por *software* aplicados nas arquiteturas do modem, assim como todos os módulos descritos em linguagem VHDL.

O modem multimodo foi projetado para implementações em FPGA e no design ASIC, modificações foram realizadas no código de descrição, utilizando o *software Quartus II* como ambiente de desenvolvimento dessas alterações. O funcionamento foi mantido sem alterações, igual ao apresentado na descrição inicial, sem alterar os três esquemas básicos de modulação, utilizando o *software ModelSim* para simular o funcionalmente após as etapas de mudanças.

Com relação aos fundamentos sobre fluxo de projeto e fabricação em ASIC, pesquisas foram realizadas em literaturas internacionais a respeito da construção de chips utilizando os conceitos focados no *SkyWater* PDK de 130 nm aberto. Essa tecnologia recente oferece forte contraste com o modelo tradicional de código fechado, com uma abordagem fora do estado da arte e um fluxo baseado em ferramentas EDA livres que facilitam o processo de design ASIC.

O *GHDL*, *Yosys* e o *OpenLane* foram as ferramentas de código aberto instaladas para implementar, no sistema *Linux*, as etapas de geração do *layout* de fabricação. O design VHDL do modem foi submetido aos processos de simulação, síntese e automação de fluxo por meio dessas ferramentas, seguindo a sequência de desenvolvimento apresentada na Figura 10. Após concluir esses processos, os arquivos GDS foram verificados na ferramenta *Klayout* e por fim, analisados através dos relatórios gerados na síntese, que finalizou essa implementação.

Figura 10 – Etapas de desenvolvimento do processo ASIC livre.



Fonte: Autoria própria (2023).

Para submeter o design modem a etapa de fabricação, foi realizado um processo ASIC utilizando a plataforma *GitHub*. Nesse segundo método, a ferramenta *OpenLane* foi executada nos servidores do *GitHub* com um fluxo com verificações automatizadas. O *GHDL* e o *Yosys* continuam funcionais no sistema *Linux*, sendo implementado nesta etapa um fluxo de trabalho misto com essas ferramentas de código aberto e os recursos do *GitHub Actions*.

A descrição VHDL do modem foi modificada novamente no ambiente do *Quartus II* e *ModelSim*, utilizando métodos que mantivessem os esquemas de modulação sem alterações. A sequência de desenvolvimento ASIC foi similar ao apresentado na Figura 10, com os arquivos da descrição sendo submetidos a processos de simulação, síntese e fluxo automatizado através das ferramentas de código aberto. Antes de iniciar o fluxo *OpenLane* no *GitHub*, foi realizado um *testbench* com esses arquivos após a síntese no *Yosys*, utilizando as ferramentas *GTKwave* e *Icarus Verilog* para implementar verificações no *hardware* digital.

Concluído esse design, os arquivos de fundição foram analisados através dos relatórios apresentados pelo *GitHub Actions*. Finalizando, o design ASIC modem foi submetido a etapa de fabricação através do evento internacional *TinyTapout*, que em colaboração com a empresa *Efabless*, produzirá o CI personalizado em processo CMOS de 130 nanômetros.

3.1.1 Modificações na arquitetura do modem multimodo

As modificações iniciais foram realizadas na arquitetura de conversão A/D e D/A, que não são necessárias no processo ASIC. Utilizando os *softwares Quartus II* e *ModelSim*, foram removidos esses blocos através de alterações no código VHDL, analisando todas as conexões internas e externas dessas arquiteturas para reescrever as partes relacionadas aos conversores sem alterar a semântica de funcionamento do modem.

Com relação a sintaxe do modem, foram implementadas correções nas bibliotecas e na descrição das arquiteturas. Para a linguagem VHDL, as ferramentas de código aberto *GHDL*, *Yosys* e *OpenLane* seguem os padrões definidos pelo IEEE, e no processo ASIC esses padrões são seguidos, de modo que as bibliotecas utilizadas em projeto devem estar de acordo com as especificações do IEEE. A tabela 1 apresenta as bibliotecas que foram utilizadas e removidas para padronizar a descrição VHDL do modem multimodo.

Tabela 1 – Bibliotecas padronizadas e personalizadas de HDL.

Padrão IEEE	**Biblioteca**
Sim	std_logic_1164
Sim	numeric_std
Não	std_logic_arith
Não	std_logic_unsigned
Não	std_logic_misc

Fonte: Autoria própria (2023).

As bibliotecas *std_logic_1164* e *numeric_std* foram utilizadas para corrigir as sintaxes dos 32 arquivos VHDL que compõe a descrição modem. Esses problemas ocorrem quando as bibliotecas personalizadas são removidas das arquiteturas, sendo necessário adequar algumas sintaxes antigas para o padrão aceito pelas bibliotecas do IEEE, utilizando os dados *std_logic* e *unsigned*. A semântica do código foi mantida a mesma, sem mudanças no funcionamento. Essas modificações foram implementadas nos *softwares Quartus II* e *ModelSim*, e analisadas através da ferramenta *GHDL*, que depurou com as especificações do IEEE.

As duas etapas com modificações apresentadas foram implementadas no design ASIC desenvolvido no sistema *Linux*, que utilizou o modem multimodo com largura de 9 bits nos três esquemas de modulação. Para o processo ASIC desenvolvido na plataforma *GitHub*, foi realizado uma terceira modificação com o propósito de submissão a etapa de fabricação.

A resolução do modem, originalmente de 9 bits, foi alterada para funcionar com 7 bits, através de mudanças na descrição do bloco oscilador local. Foi utilizado os *softwares Quartus II* e *ModelSim* para reconfigurar a nova matriz de células, endereçadas para os valores de uma senoide de 7 bits com amplitude máxima de 64 bits e resolução mínima de 30 pontos.

Na arquitetura modem, a nova senoide foi modificada nos 6 osciladores presentes nos blocos ASK, FSK e PSK de modulação e demodulação. As sintaxes precisaram ser adequadas para a nova resolução de 7 bits, através de correções nas entradas e saídas das arquiteturas.

3.1.2 Fluxo de design ASIC no Linux

O fluxo de design ASIC, implementado em ambiente *Linux,* se baseia nas ferramentas *GHDL*, *Yosys* e *OpenLane* em conjunto com o PDK de código aberto de processo *SkyWater* de 130 nm. As etapas do desenvolvimento foram definidas de acordo com essas ferramentas, iniciando com os arquivos VHDL submetidos ao processo de simulação através da ferramenta *GHDL*, que depurou os problemas de lógica dos 32 blocos do modem de 9 bits.

Após a depuração, foi realizada a síntese da descrição de alto nível do modem para sua representação lógica equivalente de nível mais baixo, no formato *verilog*. Essa etapa utilizou a ferramenta *Yosys* no processo, que sintetizou individualmente os 32 blocos modem VHDL em representações lógicas específicas de *hardware*. Os arquivos sintetizados foram integrados no repositório de design *OpenLane* no *Linux*, permitindo o uso automatizado do fluxo *OpenLane* por meio dos comandos apresentados na Figura 11.

Figura 11 – Comandos de inicialização do fluxo *OpenLane*.

```
1) export PDK_ROOT=/edatools/pdks

2) docker run --rm -it -v $(pwd):/openLANE_flow -v $PDK_ROOT:$PDK_ROOT -e PDK_ROOT=$PDK_ROOT -u
   $(id -u $USER):$(id -g $USER) openlane:rc6

3) ./flow.tcl -design modem -tag modem-9bits
```

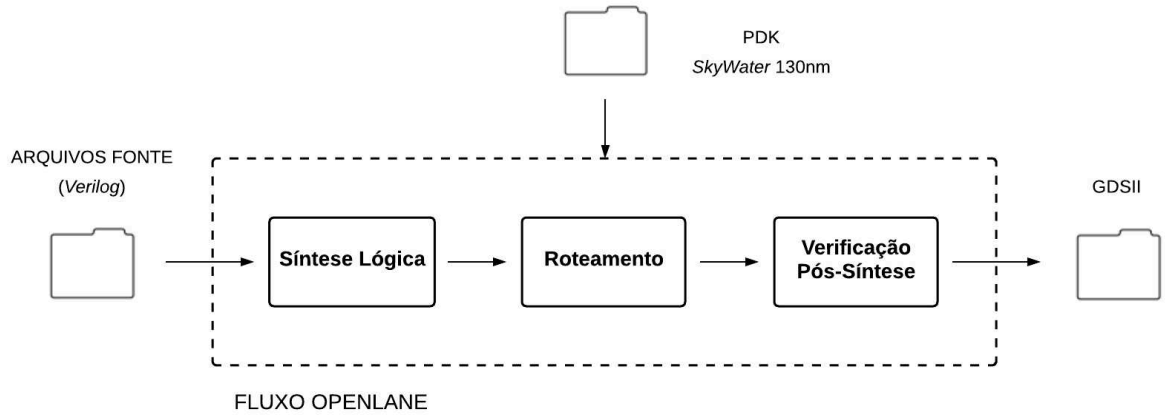
Fonte: Autoria própria (2023).

O primeiro comando permitiu a ferramenta acessar as informações específicas do PDK (bibliotecas, regras de design e recursos de fabricação sky130). O segundo comando iniciou o ambiente *OpenLane* e o último, executou o fluxo automatizado com os arquivos do modem, permitindo as principais etapas de síntese, roteamento e verificação fossem executadas.

Quando inicia o fluxo *OpenLane*, uma série de etapas são executadas automaticamente para transformar o design lógico em *layout* físico pronto para fabricação em CI. Essas etapas incluem a síntese de planejamento automático em termos de portas lógicas, o roteamento das células no espaço do chip e a verificação das regras de fabricação como as principais do fluxo *OpenLane*, sendo apresentadas na Figura 12.

O *OpenLane*, associado ao *SkyWater* 130 nm, seguiu as informações contidas no PDK para guiar o processo de síntese e *layout* do design ASIC, com as características do processo de fabricação. Após a conclusão das etapas de processo, o design foi convertido em um *layout* físico completo (GDSII) que atende as regras e as especificações do PDK.

Figura 12 – Principais processos do fluxo *OpenLane*.



Fonte: Autoria própria (2023).

3.1.3 Fluxo de design ASIC no GitHub e evento TinyTapout

O projeto *TinyTapout*, atualmente em sua 4ª edição, surgiu como uma oportunidade de submeter o design ASIC do modem a um processo real de fabricação de chip. Para submeter o design modem, foi implementado algumas etapas através da plataforma *GitHub*. Esse método também utilizou as ferramentas *GHDL*, *Yosys* e *OpenLane* em conjunto com o processo PDK *SkyWater* de 130 nanômetros de código aberto.

O *TinyTapout* forneceu o *template* para o processo ASIC, especificando os terminais de entrada e saída em 8 bits cada. Dessa forma, foi utilizado o modem com resolução de 7 bits nesse processo ASIC. As ferramentas *GHDL* e *Yosys* continuam sendo executadas no sistema *Linux* igual ao método anterior, com os 32 arquivos VHDL submetidos a etapa de depuração e em seguida, ao processo de síntese em *verilog*, respectivamente.

Os arquivos *verilog* do modem foram integrados no repositório *template* do *GitHub*. Para conectar os sinais de E/S do modem ao *template*, foi descrito um módulo de interface com todas as conexões externas e associados aos correspondentes sinais de E/S do *template*. Um *testbench* foi realizado utilizando as ferramentas *GTKwave* e *Icarus Verilog* para verificar a funcionalidade dos arquivos *verilog* no *template*.

Para gerar os estímulos na entrada do módulo de interface, empregou-se um *script* em *python* com o ambiente de teste proposto e um módulo superior conectando com os arquivos do modem. Para todas os testes, a entrada *sel* foi selecionada de acordo com a implementação da modulação proposta, o *clock* definido com período de 2 ms e para o *reset*, estabelecido um pulso sincronizante de duração igual a 4 ms no início de cada simulação.

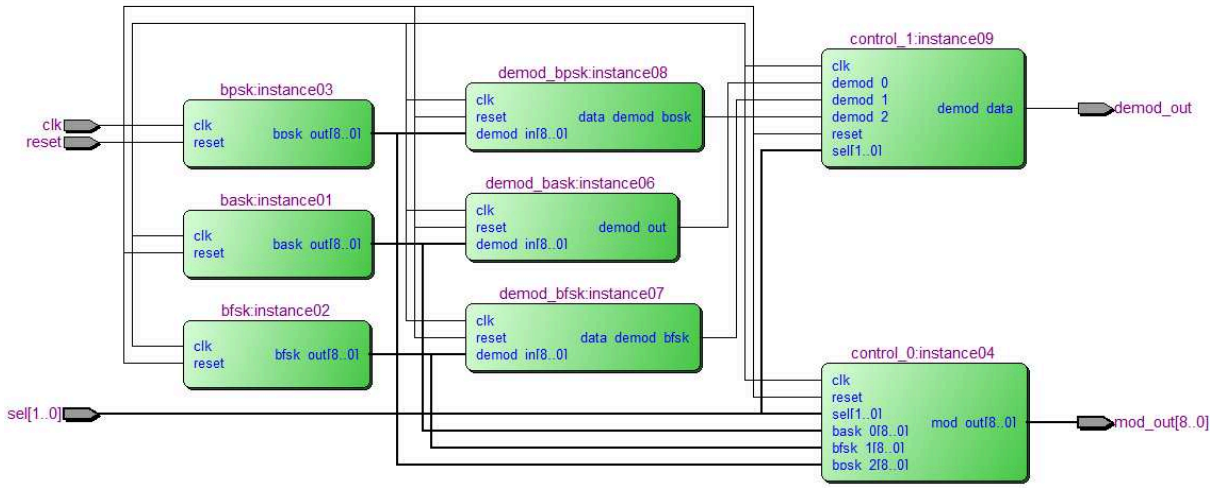
Após essa etapa, foi executado o fluxo *OpenLane* no servidor do *GitHub*, que hospeda os arquivos no repositório do design modem e realiza o processo ASIC automaticamente. As etapas seguidas no fluxo foram similares ao apresentado da Figura 10. Após concluir o fluxo, o design, convertido em *layout* físico de fabricação, deve atender as regras e as especificações do PDK *SkyWater* 130 nanômetros de código aberto.

Para atender aos requisitos estabelecidos no *template*, o processo de CI foi submetido a verificações de integridades e conformidades, utilizando o *GitHub Actions*. Essa ferramenta verificou três etapas do processo ASIC desenvolvido, definidas em qualidade dos arquivos do modem, na qualidade do *testbench* realizado e nos arquivos GDSII. Após essas verificações, o processo ASIC do modem de 7 bits foi concluído.

3.2 RESULTADOS E DISCUSSÕES

Com os arquivos da implementação modem, foram realizadas modificações no código VHDL utilizando a plataforma *Quartus II*. As primeiras alterações ocorreram nas arquiteturas auxiliares compostas por três blocos (gerador de dados, conversores A/D e D/A). Esses blocos foram removidos corretamente, com a descrição VHDL sendo compilada sem erros. A Figura 13 apresenta o diagrama de blocos do modem em nível RTL, gerado pelo *Quartus II*.

Figura 13 – Diagrama de blocos do modem multimodo após alterações.



Fonte: Autoria própria (2023).

Após remover os blocos auxiliares, foram analisadas 32 arquiteturas remanescentes do modem. Essa etapa exigiu análises repetidas para identificar e corrigir erros devidos conexões flutuantes que não foram identificadas pelo *Quartus II*, comum em projetos longos. Concluído as correções, foi compilado sem erros o código com as modificações, validando as conexões utilizadas no código. Além disso, no *software ModelSim* foi simulado as respostas gráficas do sistema e observado um funcionamento igual ao apresentado no código original, validando as primeiras modificações realizadas no modem multimodo.

Após isso, foram instaladas as ferramentas de código aberto *GHDL*, *Yosys* e *OpenLane* no sistema *Linux*, utilizando repositórios do *GitHub*. Para o *OpenLane*, algumas modificações ocorreram antes e depois do processo de instalação, pois foi customizado e utilizado apenas os pacotes necessários no processo ASIC. Assim, foram instaladas corretamente as ferramentas e iniciado a etapa de desenvolvimento ASIC através do sistema *Linux* e da plataforma *GitHub*.

3.2.1 Simulação no GHDL

Antes de iniciar o processo ASIC, foi validado a sintaxe e a semântica dos arquivos do modem de 9 bits. Foram analisados 32 arquivos com a ferramenta *GHDL* e observado erros relacionados às bibliotecas personalizadas utilizadas na descrição, que não são aceitas pelo *GHDL* por não fazerem parte do padrão IEEE. Dessa forma, essas bibliotecas do padrão IEEE foram utilizadas para corrigir esses erros, alterando todas as sintaxes declaradas na descrição VHDL para o padrão aceito e mantendo a semântica do modem sem alterações.

As modificações propostas foram válidas através dos *softwares QuartusII* e *ModelSim*, com a nova descrição compilada sem erros e apresentando simulação funcional correta. Após essas análises, todos os arquivos modem foram simulados no *GHDL* sem erros, validando a sintaxe e a semântica do design VHDL com as bibliotecas padronizadas do IEEE.

3.2.2 Síntese no Yosys

A ferramenta de código aberto *Yosys* proporcionou uma conversão precisa do código VHDL para o *verilog*, garantindo que a descrição de alto nível do design fosse corretamente traduzida para um formato de nível inferior, pronto para as etapas do fluxo de design ASIC. A conversão ocorreu sem erros, e reforça a padronização correta com as ferramentas de código aberto, em termos de sintaxe, apresentada na etapa anterior.

3.2.3 Fluxo no OpenLane

Durante a execução do fluxo *OpenLane*, foram observados alguns erros relacionados a otimização do espaço no chip. Esses erros foram corrigidos modificando duas configurações dos arquivos fonte do *OpenLane*, associados a ocupação da área do núcleo, em relação à área total disponível, e a densidade de posicionamento dos componentes utilizados no design.

Esses ajustes são relativos para cada projeto, e para o modem, os requisitos de design e otimização foram definidos através dos resultados apresentados nos relatórios automáticos do fluxo *OpenLane*. Como resultado, o fluxo *OpenLane* concluiu corretamente todas as etapas de execução, com os arquivos de *layout* físico GDSII do modem de 9 bits sendo gerados com as regras de design e as características do processo de fabricação.

3.2.4 Layout físico GDSII

Esse *layout* GDSII representa a implementação física do chip modem em um formato que pode ser utilizado na etapa de fabricação. Todos os elementos de E/S foram definidos bit a bit e a Tabela 2 apresenta as entradas e saídas definidas para o *layout* do chip modem.

Tabela 2 – Entradas e saídas do *layout* do modem de 9 bits.

Pino	Entrada	Saída
0	clock	mod_out_0
1	reset	mod_out_1
2	sel_0	mod_out_2
3	sel_1	mod_out_3
4	none	mod_out_4
5	none	mod_out_5
6	none	mod_out_6
7	none	mod_out_7
8	none	mod_out_8
9	none	demod_out

Fonte: Autoria própria (2023).

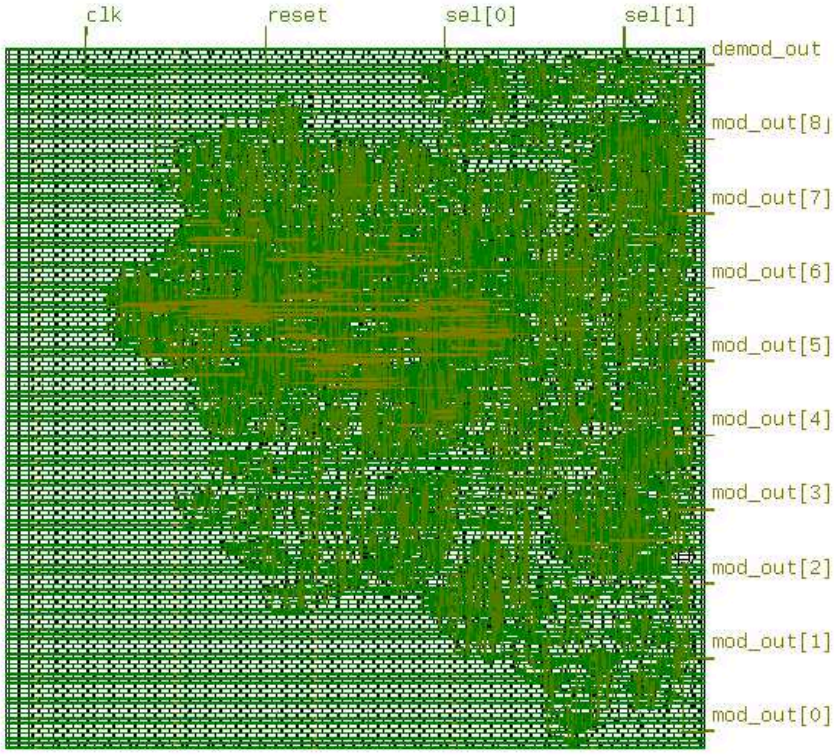
As Figuras 14 e 15 mostram o *layout* físico do modem visualizado através do *Klayout*. Esse design de 384*x*384 µm ocupou 46.7% das dimensões com 2270 células lógicas (portas logicas básicas *AND*, *NAND*, *OR*, *NOR, XOR* e *XNOR*, *buffers*, *flip flops* e multiplexadores), planejadas para serem compatíveis com o processo tecnológico de 130 nm. Em resumo, esses resultados representam um design ASIC válido e pronto para fabricação, pois foram atendidos todos os critérios de qualidade e integridade exigidos durante do o fluxo *OpenLane*. A Tabela 3 apresenta o resumo dos principais parâmetros obtidos do *layout* físico do modem.

Tabela 3 – Parâmetros do *layout* ASIC do modem de 9 bits.

Parâmetros	ASIC modem
Tecnologia	130 nm
Dimensões	384*x*384 µm
Área do chip	0.147 μm^2
Área ocupada	0.068 μm^2
Células	2270

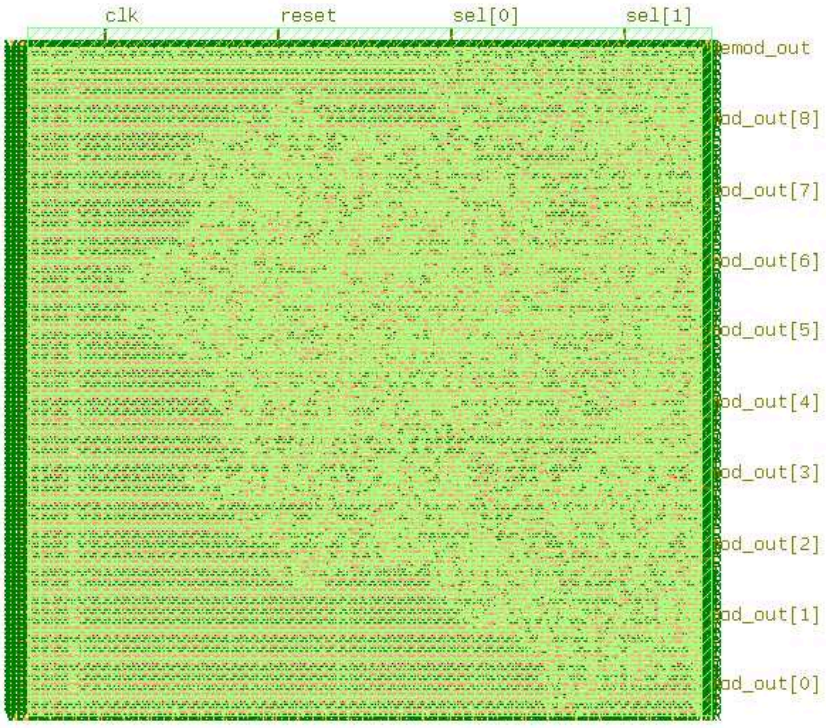
Fonte: Autoria própria (2023).

Figura 14 – *Layout* GDSII visualizado no *Klayout* (sem algumas camadas).



Fonte: Autoria própria (2023).

Figura 15 – *Layout* GDSII visualizado no *Klayout* (com todas as camadas).



Fonte: Autoria própria (2023).

3.2.5 Design ASIC no GitHub

Os arquivos em VHDL do modem foram modificados para operar com modulação de 7 bits. Essas modificações ocorreram em todos os blocos da arquitetura, e ao final, o sistema modem funcionou corretamente através dos *softwares Quartus II* e *ModelSim*, com uma etapa de compilação sem erros e resultados funcionais equivalentes aos do modem de 9 bits.

Com essa validação, foram simulados sem erros os arquivos VHDL do modem através da ferramenta *GHDL* no sistema *Linux*, validando a sintaxe e a semântica do código descrito. Para a ferramenta *Yosys* os resultados também foram corretos, visto que todos os 32 arquivos foram convertidos para formato equivalente *verilog* sem apresentar erros. Após concluir essas etapas, foram adicionados os arquivos em *verilog* ao repositório no *GitHub* com o *template*.

Validando a descrição do modem com o *template*, o módulo de interface que descreve as conexões externas foi aceito no *GitHub Actions*, apresentando verificação correta no teste de funcionalidade. Os relatórios fornecidos pelo *GitHub* apresentaram resultados corretos para essa etapa, visto que os terminais de E/S do modem foram representados nos terminais de E/S do *template*, com a indicação correta de aterramento apresentada para os sinais bidirecionais. A Tabela 4 apresenta as entradas e saídas definidas para o design ASIC do modem.

Tabela 4 – Entradas e saídas do chip modem de 7 bits.

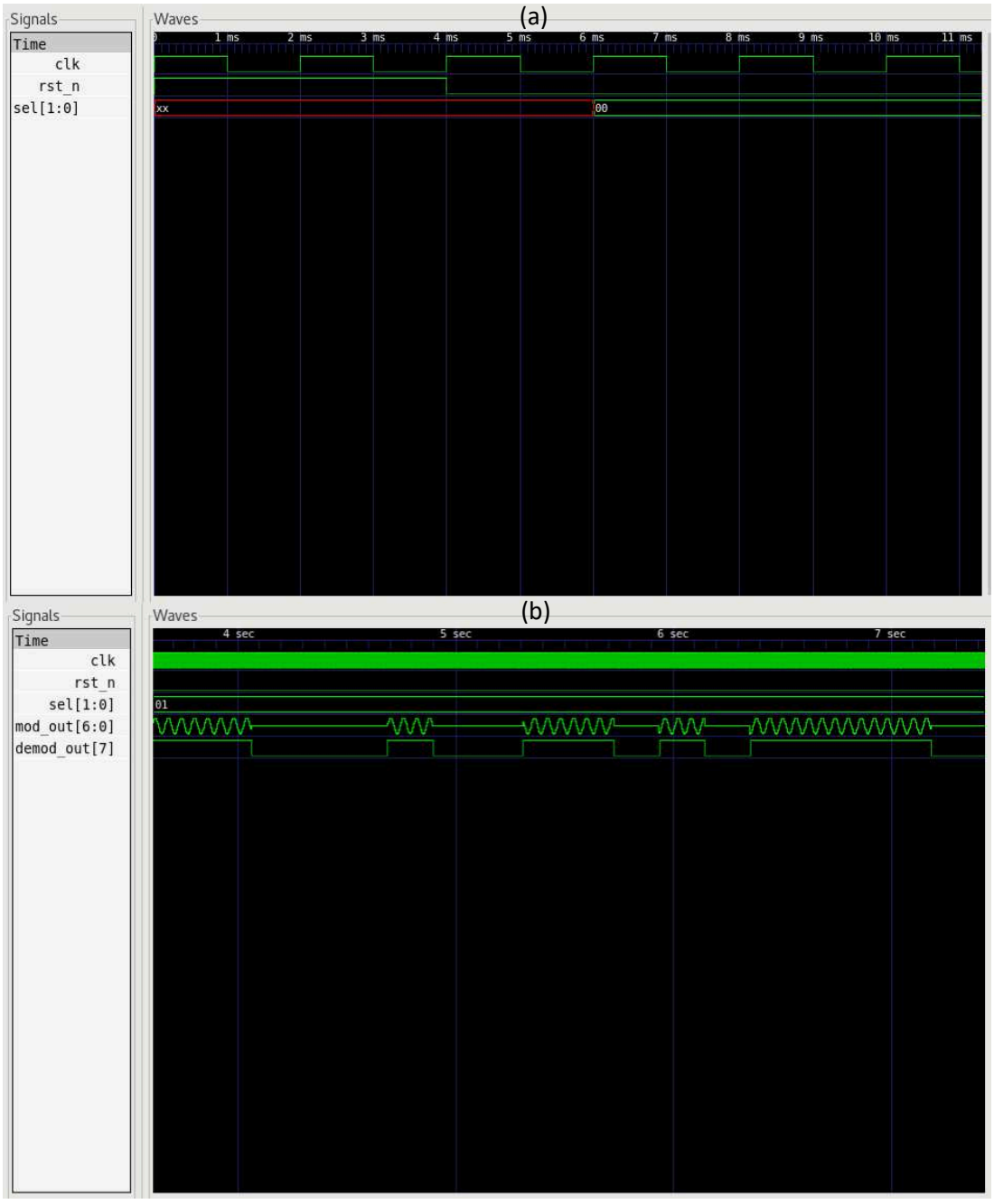
Pino	Entrada	Saída
0	clock	mod_out_0
1	reset	mod_out_1
2	sel_0	mod_out_2
3	sel_1	mod_out_3
4	none	mod_out_4
5	none	mod_out_5
6	none	mod_out_6
7	none	demod_out

Fonte: Autoria própria (2023).

Para testar a funcionalidade dos arquivos *verilog* no *template*, um *testbench* foi criado através da ferramenta *GTKwave*. As Figuras 16 e 17 ilustram as ondas gráficas obtidas para os sinais de estímulos, modulados (forma analógica) e demodulados (forma binária) através dos esquemas ASK, FSK e PSK do modem multimodo de 7 bits.

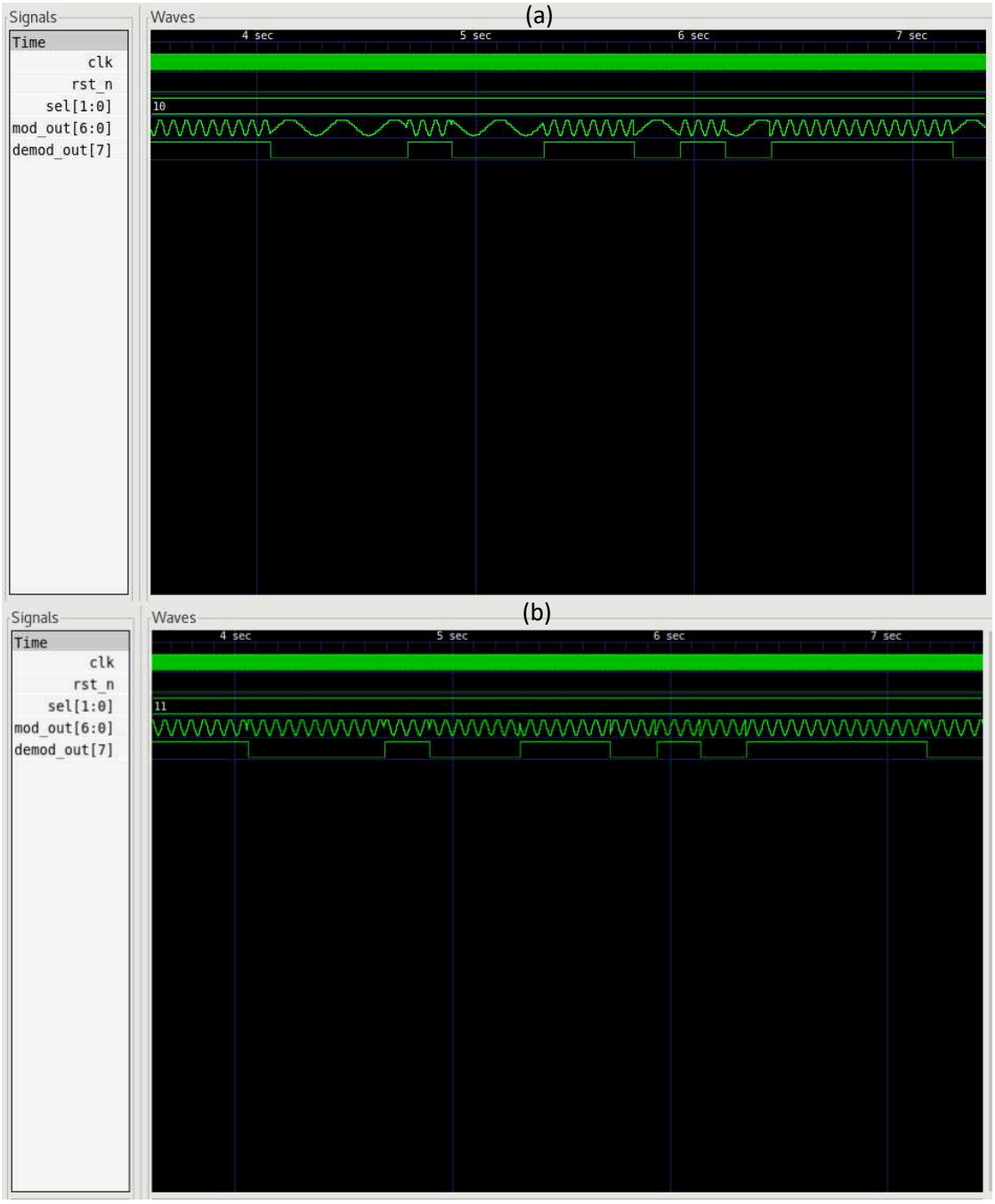
Na modulação ASK, nota-se o formato senoidal da onda apresentando trechos nulos, e nas modulações FSK e PSK, suas formas de ondas apresentaram senoides com frequência reduzida e com inversão de fase, respectivamente. Essas formas gráficas são coerentes com os estudos desenvolvidos no *Quartus II* e *ModelSim*, e indicam que os esquemas de modulação do modem estão funcionando corretamente após adaptação no *template*. Para a demodulação, observou-se o funcionamento correto para todos os esquemas.

Figura 16 – Sinais gráficos no *GTKw*ave (a) estímulos de entrada (b) esquema ASK.



Fonte: Autoria própria (2023).

Figura 17 – Sinais gráficos no GTKwave (a) esquema FSK (b) esquema PSK.



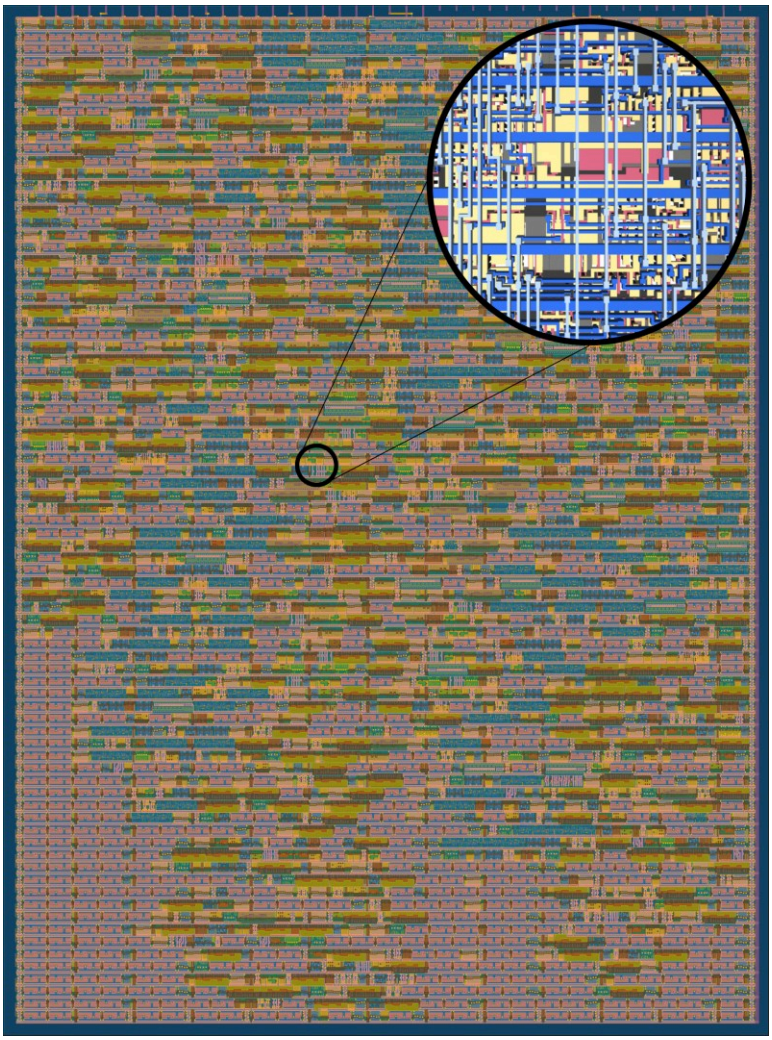
Fonte: Autoria própria (2023).

Essas respostas gráficas obtidas no *testbench* validam a descrição criada para conectar a interface do modem com os sinais do *TinyTapout*, visto que as modulações e demodulações, para os esquemas ASK, FSK e PSK, funcionam corretamente com todos os cenários testados. Após isso, o fluxo *OpenLane* foi executado automaticamente na plataforma *GitHub*, gerando os arquivos GDSII do design ASIC modem, com *layout* ocupando 1*x*2 *tiles* de espaço.

Desse modo, foi possível concluir e validar as principais verificações exigidas para a submissão, sendo duas obrigatórias e uma opcional. As ações obrigatórias estão associadas ao modem com o *template*, que gerou os arquivos de design ASIC válidos. A ação opcional está associada ao *testbench*, que não foi exigido no *template*, mas sua realização foi fundamental para verificar o design do modem funcionando conforme o esperado. Dessa forma, o design ASIC atendeu aos requisitos especificados pelo *TinyTapout* e foi submetido para fabricação.

A Figura 18 mostra a representação do chip modem em *layout* 2D, gerado através do *GitHub Actions*. As dimensões finais do chip foram de 160*x*200 µm e desse espaço, 52.6% foi ocupado com o roteamento de 1943 células lógicas, estruturadas principalmente por portas logicas básicas (*AND*, *OR*, *NAND* e *NOR*), *flip flops*, *buffers* e multiplexadores, que serão fabricados em nó tecnológico de 130nm. O consumo energético do chip foi estimado em 1.19 mW, com 79.7% dessa potência consumida internamente pelos elementos do design.

Figura 18 – *Layout* 2D representando o chip ASIC do modem.



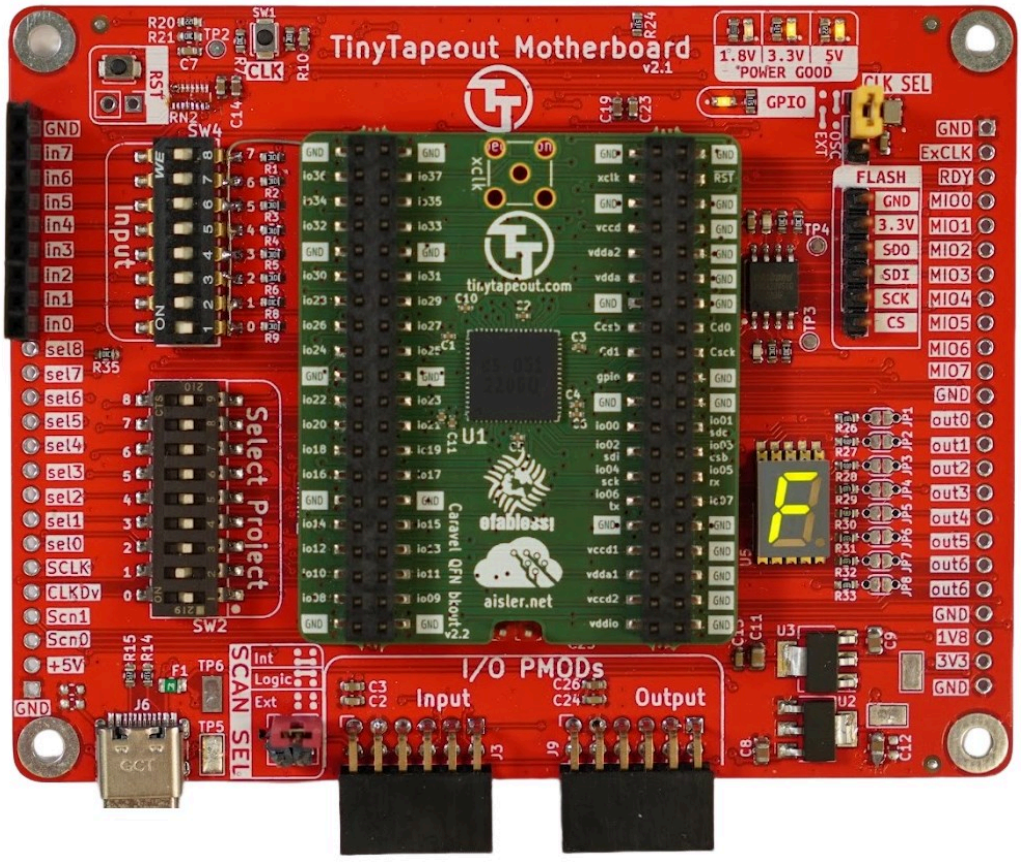
Fonte: Autoria própria (2023).

3.2.6 Fabricação do chip modem

O design ASIC foi submetido a etapa de fabricação através do evento *TinyTapout*, que utilizará nó tecnológico de 130 nm para produzir um kit (chip e PCB) a um custo de US$160. Para alcançar esse preço acessível, o *TinyTapout* utilizou o serviço *shuttle*, que compartilha os custos da fabricação utilizando um único *wafer* de silício para comportar os múltiplos projetos da 4ª edição. A Figura 19 mostra o modelo do kit que será produzido, com o chip pré-soldado em uma placa de suporte (em verde) e a PCB (em vermelho) com as entradas e saídas.

O chip será produzido com todos os designs enviados na quarta edição do *TinyTapout*, e para selecionar o design do modem, deverá utilizar os controles externos da PCB para rotear os pinos físicos do chip às entradas e saídas do modem multimodo. A placa PCB contará com três níveis de alimentação (1.8 V, 3.3 V e 5 V), espaço para encaixe de memória, display de 7 segmento e diversos pinos de E/S, com essa placa possuindo dimensões de 104.5x81.0 mm. A placa que recebe o chip possui dimensões de 46.0x52.0 mm.

Figura 19 – Modelo do chip ASIC na placa de suporte e PCB.



Fonte: TinyTapout (2023).

3.2.7 Análises comparativas

O desenvolvimento em ASIC para o modem com funcionamento de 9 e 7 bits mostrou características importantes do processo. Em ambos os casos, os designs foram submetidos as mesmas ferramentas de código aberto com o PDK de 130 nanômetros. Na tabela 5, observa-se que os parâmetros obtidos no modem de 7 bits foram mais eficientes quando comparados com o modem de 9 bits. O fluxo *OpenLane* realizado no *GitHub* seguiu características específicas estabelecidas pelo evento *TinyTapout*, tornando-se um diferencial para essa otimização.

As dimensões menores do *layout* ASIC de 7 bits ocorre devido a menor quantidade de células no roteamento, implicando em um consumo baixo da área ocupada. Outro importante fator são as alterações realizadas no modem de 7 bits, que mesmo sendo poucas, influenciam os resultados final do *layout*. O ASIC de 9 bits possui resultados aceitáveis para as primeiras sínteses, podendo ser melhor otimizado através de um entendimento maior nas configurações da ferramenta *OpenLane*.

Tabela 5 – Resultados comparativos entre as implementações.

Parâmetros	**ASIC modem 9 bits**	**ASIC modem 7 bits**
Tecnologia	130 nm	130 nm
Dimensões	384*x*384 µm	160*x*200 µm
Área do chip	0.147 μm^2	0.032 μm^2
Área ocupada	0.068 μm^2	0.016 μm^2
Células	2270	1943

Fonte: Autoria própria (2023).

4 CONSIDERAÇÕES FINAIS

Neste trabalho, foi proposto um desenvolvimento de um modem multimodo integrado em processo de 130 nanômetros utilizando fluxo de código aberto. A pesquisa permitiu atingir com sucesso os objetivos específicos relacionados a modificação, implementação, validação e avaliação de todas as etapas construtivas do projeto ASIC personalizado.

A utilização da linguagem VHDL possibilitou modificar a arquitetura do SDR modem, de forma flexível e configurável, através dos ambientes *Quartus II* e *ModelSim*. Os blocos que compõe a descrição do modem, em nível de sintaxe, foram adequados para os padrões aceitos pelo IEEE e válidos para a implementação nas ferramentas de código aberto. As compilações e simulações viabilizaram as modificações empregadas com uma semântica funcional.

Os processos de simulação, síntese e automação de fluxo das etapas implementadas no design ASIC validaram o uso das ferramentas *GHDL*, *Yosys* e *OpenLane* de código aberto. Os arquivos VHDL do modem foram simulados e sintetizados de forma adequada, indicando que o comportamento do design está de acordo com as conformidades estabelecidas pelo IEEE. O design passou com sucesso por todas as etapas do processo de CI automatizado, com o *layout* físico do chip compatível às regras de fabricação CMOS de 130 manômetros.

O design de *hardware* do modem apresentou validação funcional nos terminais de E/S através de *testbenches*, com os esquemas de modulação e demodulação funcionando de forma adequada nos testes, indicando que o circuito foi preservado após o processo de síntese.

Após a implementação do modem em ASIC, análises apresentaram um *layout* de chip com dimensões de 160*x*200 μm e alimentação de 1.8 V com consumo energético de 1.19 mW. Esses resultados demonstram adequação para uso em aplicações de sensoriamento na internet das coisas, visto que, a eficiência energética e o tamanho do *layout* são fatores críticos para os dispositivos de comunicação sem fio em IoT.

O trabalho foi submetido para fabricação através do evento *TinyTapout*, que enviará à fundição de semicondutores o projeto, finalizando a implementação física do chip. Com isso, o desenvolvimento do modem multimodo em ASIC foi bem sucedido, atingindo os objetivos estipulados para o trabalho de forma satisfatória e contribuindo para o recente movimento de democratização de projetos de circuitos integrados em código aberto.

4.1 TRABALHOS FUTUROS

Como continuidade deste trabalho, seguem abaixo algumas sugestões que podem ser adotadas, realçando-se que muitos outros tópicos poderiam naturalmente ser incluídos.

- Realizar testes experimentais com a PCB do modem multimodo, analisando os processos de modulação e demodulação em bancada laboratorial;
- Realizar estudos comparativos entres as implementações FPGA e ASIC;
- Analisar o desempenho do modem multimodo com outras arquiteturas SDR já empregados no âmbito acadêmico e de mercado;
- Implementar aplicações na área de sensoriamento da internet das coisas.

REFERÊNCIAS

ALAM, Syed Anas et al. Open-Source Chip Design in Academic Education. In: 2022 IEEE Nordic Circuits and Systems Conference (NorCAS). IEEE, 2022. p. 1-6.

CRONJE, Johannes Jacobus. Software Architecture Design of a Software Defined Radio System. 2004. Tese de Doutorado. Stellenbosch: Stellenbosch University.

CRUZ, Eduardo et al. Sistemas digitais reconfiguráveis: FPGA e VHDL. Alta Books, 2022.

D'AMORE, Roberto. VHDL: Descrição E Síntese de Circuitos Digitais. Grupo Gen-LTC, 2000.

DE MELO, Wellington Romeiro. Estudo do Fluxo de Projeto de Circuitos Integrados Digitais de Aplicação Específica (ASICS) Aplicado a um CI Monitor de Velocidade. 2004. Tese de Doutorado. UNIVERSIDADE ESTADUAL DE CAMPINAS.

DUARTE JÚNIOR, José Garibaldi et al. Desenvolvimento de um modem multimodo baseado em fpga para aplicações em rádio definido por software. 2018.

EDWARDS, R. Timothy. Google/SkyWater and the Promise of the Open PDK. In: Workshop on Open-Source EDA Technology. 2020.

FINKELSTEIN, U. et al. GTKWave 3.3 Wave Analyzer User's Guide. 2011.

FORUM, W. I. Software Defined Radio (SDR) forum: technical definitions., 2017. Disponível em: http://www.wirelessinnovation.org/Introduction_to_SDR. Acesso em: 18 de agosto de 2023.

GHAZY, Ahmed; SHALAN, Mohamed. Openlane: The open-source digital asic implementation flow. In: Proc. Workshop on Open-Source EDA Technol. (WOSET). 2020.

HATAI, Indrani. lndrajit Chakrabarti," FPGA Implementation of a Digital FM Modem. In: International Conference on Information and Multimedia Technology Conference. 2009.

HESHAM, Sarah et al. Digital ASIC Implementation of RISC-V: OpenLane and Commercial Approaches in Comparison. In: 2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS). IEEE, 2021. p. 498-502.

HUANG, Tsung-Wei et al. Essential building blocks for creating an open-source eda project. In: Proceedings of the 56th Annual Design Automation Conference 2019. 2019. p. 1-4.

KUMAR, Gaurav; CHATTERJEE, Baibhab; SEN, Shreyas. OpenSerDes: an open source process-portable all-digital serial link. In: 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2021. p. 386-391.

KHAN, Saif M.; MANN, Alexander; PETERSON, Dahlia. The semiconductor supply chain: Assessing national competitiveness. Center for Security and Emerging Technology, v. 8, n. 8, 2021.

LALGE, Archana M.; SHRIVASTAV, Anjali; BHANDARI, Sheetal U. Implementing PSK modems on FPGA using partial reconfiguration. In: 2015 International Conference on Computing Communication Control and Automation. IEEE, 2015. p. 917-921.

LI, Weihui; WEN, Haoyu; DUAN, Peiji. Key technologies and international trends in EDA field of digital IC design: a patent analysis. In: SHS Web of Conferences. EDP Sciences, 2022. p. 01020.

LUCK, Christian et al. Industrial Experience with Open-Source EDA Tools. In: Proceedings of the 2022 ACM/IEEE Workshop on Machine Learning for CAD. 2022. p. 143-143.

MITOLA, J. SoftwareRadios: Survey, Critical Evaluation and Future Directions, proc. National Telesystems Conference. 1992.

MOSKAL, Jakub Jerzy. Interfacing a reasoner with heterogeneous self-controlling software. 2011. Tese de Doutorado. Northeastern University.

NETO, V. S. Sistemas de Comunicação-Serviços Modulação e Meios de Transmissão. São Paulo: Editora Érica, v. 1, 2015.

OLIVEIRA, Robert Kley Lageano de. Otimização de sistemas de comunicação: modulações digitais e antenas. 2014.

QUISPE, Segundo Gerardo Gamarra. Análise e implementação de um sistema de comunicações sem-fio por portadora única utilizando rádio definido por software= Analysis and implementation of a wireless communication system on an unique carrier using a software defined radio. 2015. Tese de Doutorado.

RIBAS, Renato Perez; REIS, Andre Inacio; LUBASZEWSKI, Marcelo Soares. Concepção de circuitos e sistemas integrados. Revista de informática teórica e aplicada. Porto Alegre. Vol. 8, n. 1 (set. 2001), p. 7-21, 2001.

ROBERT, Max et al. Software frameworks for SDR. Proceedings of the IEEE, v. 103, n. 3, p. 452-475, 2015.

ROCKET55. Open Source ASICs Take a Giant Leap Forward with the First Ever Open Foundry PDK., 2021. Disponível em: https://www.skywatertechnology.com/open-source-asics-take-a-giant-leap-forward-with-first-ever-open-foundry-pdk/. Acesso em 29 de agosto de 2023.

SAYARA, Anika. Amplitude Shift Keying (ASK) || Digital to Analog Signal (Part 2). Youtube, 2020. Disponível em: https://www.youtube.com/watch?v=Ma-hAQsbYFI. Acesso em 01 setembro 2023.

SCHEFFER, Louis; LAVAGNO, Luciano; MARTIN, Grant (Ed.). EDA for IC system design, verification, and testing. CRC press, 2018.

STEVE ICARUS. Icarus Verilog Repository. Disponível em: <https://github.com/steveicarus/iverilog\>. Acesso em: 18 setembro de 2023.

TGINGOLD. ghdl-yosys-plugin. 2020. Disponível em: https://github.com/ghdl/. Último acesso em: 2 de setembro de 2023.

TINYTAPOUT. Disponível em: https://tinytapeout.com/specs/pcb/. Acessado 28 de setembro de 2023.

TUOMI, Ilkka et al. The Future of Semiconductor Intellectual Property Architectural Blocks in Europe. European Communities, 2009.

TUTTLEBEE, Walter HW (Ed.). Software defined radio: enabling technologies. John Wiley & Sons, 2003.

WOLF, Clifford. Yosys open synthesis suite. 2016.