

# Nota técnica SKARAB

Rafael A. Batista

LABMET & Unidade Acadêmica de Física - Universidade Federal de Campina Grande,  
Brasil.

[rafael.alves.batista@gmail.com](mailto:rafael.alves.batista@gmail.com)

## Abstract

Esta nota técnica descreve a implementação, configuração e os resultados preliminares da plataforma SKARAB no radiotelescópio BINGO. Utilizando o ecossistema CASPER, a plataforma foi configurada como um analisador de espectro em tempo real para o processamento digital de sinais. Discutimos os desafios de instalação, a validação do sistema com diferentes fontes de sinal e a infraestrutura de armazenamento de dados. Os resultados em bancada confirmam a viabilidade da SKARAB para a operação do BINGO, estabelecendo a base para futuras discussões sobre calibração e estabilidade do sistema.

## Contents

<b>1</b>	<b>Introdução</b>	<b>3</b>
<b>2</b>	<b>Descrição das SKARABs</b>	<b>3</b>
2.1	Introdução à FPGA e Computação Reconfigurável . . . . .	4
2.1.1	FPGA da SKARAB: Virtex-7 XC7VX690T-2FFG1927C . . . . .	4
2.2	Mezzanine ADC da SKARAB: TI ADC32RF45 . . . . .	6
2.3	Descrição do setup de medidas . . . . .	7
<b>3</b>	<b>Descrição das atividades</b>	<b>7</b>
3.1	Testes do fabricante . . . . .	7
3.2	Instalação de pacotes básicos . . . . .	7
3.3	A questão da decimação . . . . .	7
3.4	Requisitos de Software e Instalação . . . . .	8
3.5	Geração de Bitstream e Programação da FPGA . . . . .	9
3.5.1	Geração de Bitstream . . . . .	9
<b>4</b>	<b>Tutorial de Introdução ao SKARAB</b>	<b>9</b>
4.1	Configuração do Ambiente e Criação do Modelo . . . . .	10
4.2	Integração de Blocos e Lógica de Design . . . . .	10
4.3	Simulação e Compilação . . . . .	11
4.4	Programação e Intereração via Python . . . . .	11
4.4.1	Programação da FPGA . . . . .	12
4.5	Especrómetro (Tutorial 4) . . . . .	12
4.5.1	Digitalização (ADC) . . . . .	12
4.5.2	Transformada de Fourier (FFT) . . . . .	13
4.5.3	Cálculo de Potência e Acumulação . . . . .	13
4.6	Resultados . . . . .	13
4.7	O espetrômetro do BINGO e desafios . . . . .	14
<b>5</b>	<b>Status atual do trabalho</b>	<b>17</b>
<b>6</b>	<b>Agradecimentos</b>	<b>21</b>

<b>A Instalação e Execução do <i>casperfpga</i></b>	<b>21</b>
A.1 Instalação do <i>casperfpga</i> . . . . .	22
A.2 Execução na IDE <i>IPython</i> . . . . .	25
<b>B Instalação e Execução do <i>mlib-devel</i></b>	<b>26</b>

# 1 Introdução

O presente trabalho tem como objetivo principal descrever o equipamento, a configuração, a utilização e a análise da plataforma SKARAB no contexto do radiotelescópio BINGO. Buscou-se integrar o conhecimento teórico em processamento digital de sinais à prática de instrumentação científica aplicada à radioastronomia, promovendo o domínio das ferramentas e dos ambientes necessários para o controle e a operação da placa FPGA. Os objetivos específicos deste trabalho são:

- Compreender o funcionamento da plataforma SKARAB e sua aplicação no projeto BINGO;
- Instalar, configurar e utilizar as ferramentas de software associadas, como *casperfpga*, *mlib-devel*, Python 2.7, MATLAB e Vivado;
- Realizar testes em ambiente local com a placa SKARAB operando em modo autônomo;
- Apoiar o desenvolvimento de soluções que facilitem o controle e a análise dos dados adquiridos pelo radiotelescópio.

Como parte da instrumentação (estágio 0) do projeto BINGO, os trabalhos em torno da SKARAB são baseados nos tutoriais disponíveis pela CASPER (Collaboration for Astronomy Signal Processing and Electronics Research )<sup>1</sup>, disponíveis em sua Wikipage ([https://casper.berkeley.edu/wiki/Main\\_Page](https://casper.berkeley.edu/wiki/Main_Page)).

Os tutoriais, toolflows e demais elementos de software utilizados para a programação das SKARABs podem ser encontrados (ou listados, no caso de software proprietário) em [https://casper.berkeley.edu/wiki/Main\\_Page](https://casper.berkeley.edu/wiki/Main_Page). Os links para seções relevantes a esta nota técnica são:

- Software em geral: <https://casper.berkeley.edu/wiki/Software>
- Toolflows: <https://casper-toolflow.readthedocs.io/en/latest/>
- Tutoriais: esta página de tutoriais da CASPER não é atualizada desde 2017. Tutoriais e atualizações a partir dessa data foram transferidos um GitHub (<https://casper-toolflow.readthedocs.io/en/latest/>). Entretanto, há informações interessantes e links atualizados na página original, em <https://casper.berkeley.edu/wiki/Tutorials>

Dessa forma, esta nota técnica está organizada nas seguintes seções: a Seção 2 apresenta a descrição das placas SKARAB, incluindo comentários sobre a FPGA Virtex-7 e seus mezzaninos ADC e interfaces Ethernet QSFP+; a Seção 3 detalha as atividades desenvolvidas, abrangendo a instalação dos pacotes básicos, o processo de decimação no contexto do processamento digital de sinais durante a construção do firmware, os resultados obtidos com os primeiros espetrômetros e o objetivo final do trabalho; por fim, a Seção 4 discute o estado atual do projeto e as perspectivas futuras.

## 2 Descrição das SKARABs

A SKARAB (Square Kilometre Array Reconfigurable Application B) é uma plataforma de hardware de aquisição e processamento de dados baseada em FPGA (Field Programmable Gate Array), utilizando um Virtex-7 (XC7VX690T). Desenvolvida pela empresa sul-africana Peralex de acordo com as especificações do SKAO-SA, a SKARAB é ideal para aplicações em radioastronomia [1] e otimizada para processamento digital de

---

<sup>1</sup><https://casper.berkeley.edu>

sinais de alto desempenho, possuindo mais de 690.000 células lógicas, 3.600 blocos DSP e 53 Mb de memória RAM.

Ela integra um conversor analógico-digital (ADC) de 14 bits com taxa de amostragem de 3 GSps, permitindo amostragem livre de aliasing até 1,5 GHz, adequada para o processamento simultâneo e preciso de múltiplas faixas de frequência. O ADC suporta conexão com um conversor digital descendente (DDC) de banda dupla, que inclui até três osciladores numericamente controlados (NCOs) de 16 bits por DDC, possibilitando saltos de frequência coerentes em fase.

Além do ADC, a SKARAB possui suporte a periféricos Mezzanine para armazenamento e transmissão de dados. Um exemplo é o módulo Mezzanine Ethernet, que inclui quatro interfaces QSFP+ (Quad Small Form-factor Pluggable Plus), cada uma com taxa de até 40 GbE, resultando em largura de banda agregada de até 160 Gbps. A plataforma oferece espaço para quatro placas Mezzanine, cada uma com interface para 16 transceptores seriais de alta velocidade (10 Gb/s). Atualmente, existem três placas mezzanine para a SKARAB disponíveis no mercado:

- Módulo Mezzanine QSFP+ com quatro interfaces Ethernet de 40 Gb;
- Módulo Hybrid Memory Cube (HMC) com alta largura de banda de memória e capacidade de 4 GB por módulo;
- Placa Mezzanine ADC compatível com a interface mezzanine SKARAB, fornecendo dois chips ADC TI AD32RF45 dual 3 GSPS de 14 bits com DDC, para quatro canais A/D por placa.

Atualmente, a placa SKARAB do projeto BINGO está equipada com um único Módulo Mezzanine QSFP+ e dois Módulos Mezzanine ADC, não incluindo um Módulo Hybrid Memory Cube.

## 2.1 Introdução à FPGA e Computação Reconfigurável

Um FPGA (Field Programmable Gate Array) é um circuito integrado digital cuja principal característica é poder ser configurado pelo usuário após a fabricação. Diferente de chips com funções fixas, como microcontroladores ou ASICs, o FPGA é composto por uma matriz de blocos lógicos programáveis, memórias, e interconexões reconfiguráveis, permitindo implementar circuitos digitais personalizados em nível de hardware.

Ele é considerado reprogramável porque sua funcionalidade não é determinada permanentemente. O comportamento do FPGA é definido por um arquivo de configuração chamado bitstream, que contém os dados necessários para ajustar o estado interno dos blocos lógicos e das conexões entre eles.

Durante a programação, o bitstream é carregado na memória interna do FPGA (geralmente SRAM), configurando as portas lógicas, roteamentos e registradores de acordo com o circuito descrito em uma linguagem de descrição de hardware (como VHDL ou Verilog). Ao alterar o bitstream, é possível mudar completamente a função do hardware, por exemplo, transformar um processador digital em um filtro de sinais, e vice-versa — tudo no mesmo dispositivo físico.

Assim, o FPGA combina a flexibilidade do software com o desempenho do hardware dedicado, sendo amplamente usado em aplicações que exigem alto desempenho e possibilidade de atualização ou prototipagem rápida.

### 2.1.1 FPGA da SKARAB: Virtex-7 XC7VX690T-2FFG1927C

O **Virtex-7 XC7VX690T-2FFG1927C** é um FPGA de alto desempenho da família Virtex-7 da AMD (anteriormente Xilinx), projetado para aplicações que exigem alta capacidade de lógica, largura de banda elevada e eficiência energética. Fabricado utilizando

tecnologia de processo de 28 nm HPL (High-Performance, Low-Power) com tecnologia de porta metálica de alto-k (HKMG), este dispositivo oferece uma solução robusta para sistemas de processamento de sinais, redes de alta velocidade e prototipagem de ASICs.

## Especificações Principais

- **Células Lógicas:** 693.120
- **Módulos Lógicos Adaptáveis (ALMs):** 108.300
- **Memória Embutida:** 51,68 Mbit
- **Número de I/Os:** 600
- **Taxa de Dados Máxima:** 28,05 Gb/s
- **Número de Transceptores:** 80
- **Frequência Máxima de Operação:** 640 MHz
- **Tensão de Alimentação:** 0,97 V a 1,03 V
- **Temperatura de Operação:** 0°C a 85°C
- **Pacote:** FCBGA-1927 (45 mm x 45 mm)

## Arquitetura e Recursos

- **LUTs de 6 Entradas:** Utilizadas para lógica avançada e memória distribuída.
- **Memória RAM Distribuída:** 10.888 Kbit.
- **Memória RAM de Bloco (BRAM):** 52.920 Kbit.
- **Módulos de Gerenciamento de Clock (CMT):** Inclui PLLs e MMCMs para gerenciamento preciso de clock.
- **XADC Dual de 12 bits:** Conversores analógico-digital integrados com sensores térmicos e de alimentação.

## Aplicações Típicas

- Redes de alta velocidade (10G a 100G).
- Sistemas de radar portáteis.
- Prototipagem de ASICs.
- Processamento de sinais de alta performance.
- Aplicações em comunicações ópticas e de rádio frequência.

## Vantagens

- **Alta Densidade Lógica:** Até 2 milhões de células lógicas em variantes superiores.
- **Baixo Consumo de Energia:** Até 50% mais eficiente energeticamente em comparação com soluções de múltiplos chips.
- **Alta Largura de Banda Serial:** Suporte a transceptores de até 28,05 Gb/s.
- **Arquitetura Escalável:** Facilita a migração de designs da família Virtex-6.

Para especificações completas e datasheet, visite: [https://www.xilinx.com/support/documentation/data\\_sheets/ds183\\_Virtex\\_7\\_Data\\_Sheet.pdf](https://www.xilinx.com/support/documentation/data_sheets/ds183_Virtex_7_Data_Sheet.pdf).

## 2.2 Mezzanine ADC da SKARAB: TI ADC32RF45

ADC32RF45 é um conversor analógico-digital (ADC) de alto desempenho, dual-channel, com 14 bits de resolução, otimizado para aplicações de amostragem RF de banda larga, como comunicações 5G, radar e guerra eletrônica. Cada canal suporta taxas de amostragem de até 3,0 GSPS e frequências de entrada de até 4,0 GHz.

### Principais Especificações

- Resolução: 14 bits
- Taxa de Amostragem: 3,0 GSPS por canal
- Frequência de Entrada: DC a 4,0 GHz
- Tensão de Entrada: 1,35 Vpp diferencial
- Densidade Espectral de Ruído: -155 dBFS/Hz
- Jitter de Abertura (Aperture Jitter): 90 fs
- Isolamento entre Canais: 95 dB @ 1,8 GHz
- Consumo de Energia: 3,2 W por canal @ 3,0 GSPS
- Pacote: VQFN de 72 pinos (10 mm × 10 mm)

### Desempenho

- SNR: 60,9 dBFS @ 900 MHz, 58,8 dBFS @ 1,78 GHz
- SFDR: 67 dBc (HD2, HD3) @ 900 MHz, 66 dBc @ 1,78 GHz
- Pior Spur SFDR: 77 dBc @ 900 MHz, 75 dBc @ 1,78 GHz

### Conversor Digital Descendente (DDC)

- Até 4 DDCs por canal, modo dual-band
- 3 NCOs independentes de 16 bits por DDC
- Detectores de pico e RMS integrados
- Terminação de entrada e proteção contra sobretensão

### Interface & Sincronização

- JESD204B Subclasse 1, latência determinística
- 4 lanes por canal, até 12,5 Gbps por lane
- Sincronização multi-chip para sistemas phased-array

### Avaliação & Desenvolvimento

- ADC32RF45EVM com gerador de clock LMK04828 onboard
- Suporta clock externo e interface JESD204B
- GUI para configuração e controle

## Aplicações

- Comunicações de banda larga (5G e além)
- Sistemas de radar (phased-array, SAR)
- Guerra eletrônica e inteligência de sinais
- Testes e medições de sinais de alta frequência

Para especificações completas e datasheet, visite: <https://www.ti.com/product/ADC32RF45>.

### 2.3 Descrição do setup de medidas

## 3 Descrição das atividades

### 3.1 Testes do fabricante

Os únicos testes disponíveis para o manuseio da placa SKARAB encontram-se nos tutoriais da CASPER, que não recebem atualizações desde 2017. Entre esses tutoriais, destacam-se o tutorial introdutório (tutorial 1), que orienta a criação de um design de blocos simples para a soma de valores de registradores e o uso de blocos contadores. Em seguida, há o tutorial voltado à interface QSFP+ Ethernet (tutorial 2), que aborda sua configuração e manipulação. Por fim — e de maior relevância — encontra-se o tutorial do espectrômetro (tutorial 4 e 5), no qual o design de blocos inclui um ADC responsável pela captura de dados para processamento digital de sinais, culminando na plotagem do espectro obtido. Para acessar os tutoriais da CASPER, click no seguinte link<sup>2</sup>.

### 3.2 Instalação de pacotes básicos

A implementação da plataforma SKARAB e a aquisição de dados foram realizadas em um ambiente Python. Inicialmente, utilizou-se o sistema operacional Ubuntu 16.04 LTS, uma vez que esta era a única versão compatível com os softwares necessários para a descrição do hardware implementado na SKARAB. A identificação e compreensão dos requisitos essenciais para a correta configuração dos ambientes representaram um desafio significativo, demandando vários meses de trabalho contínuo e análise detalhada, em razão da escassa documentação disponibilizada pela colaboração CASPER. Ao final desse processo, foi possível mapear com sucesso as dependências dos pacotes *casperfpga* (veja apêndice A) e *mlib-devel* (veja apêndice B), estabelecendo uma base sólida para o desenvolvimento e operação da plataforma.

### 3.3 A questão da decimação

A decimação, em processamento de sinal digital (DSP), consiste em reduzir a taxa de amostragem de um sinal, isto é, diminuir o número de amostras por segundo. A decimação é amplamente empregada em sistemas de rádio definido por software (SDR) e aplicações de comunicação digital, pois permite concentrar o processamento apenas na banda de interesse, reduzindo significativamente a quantidade de dados e o custo computacional. Esse processo afeta diretamente o comprimento (ou largura) de banda do espectro analisado, uma vez que a largura de banda máxima de um sinal digital é proporcional à sua frequência de amostragem.

---

<sup>2</sup><https://casper-toolflow.readthedocs.io/projects/tutorials/en/latest/>

Na placa SKARAB, a decimação é implementada digitalmente dentro do bloco DDC (Digital Downconverter), executado em FPGA. Esse bloco combina operações de translação de frequência (mixagem digital), filtragem e redução da taxa de amostragem, permitindo isolar e extrair sub-bandas específicas do sinal de entrada de 3 GHz com elevada flexibilidade. A placa utiliza ADCs operando a uma taxa de amostragem de 3 GHz; entretanto, no modo DDC, os fatores de decimação programáveis disponíveis no design de blocos do Simulink são de 4, 8, 16, 32, entre outros. Assim, o maior comprimento de banda alcançável é de aproximadamente 750 MHz, correspondente ao fator de decimação igual a 4.

No contexto do projeto BINGO, um fator de decimação igual a 8 mostra-se adequado para o comprimento de banda de interesse (ver linha 7 da Tabela 1). Esse fator resulta em uma largura de banda de 375 MHz, valor que pode ser ajustado conforme a taxa de amostragem selecionada para o ADC. A Tabela 1 apresenta de forma completa a relação entre a taxa de amostragem de saída, a taxa de amostragem do ADC e do FPGA, bem como os fatores de decimação e de reamostragem (resampler) empregados no sistema.

Table 1: Taxas de Amostragem (**TA**) disponíveis no ADC da SKARAB.

<b>Ref</b>	<b>TA ADC</b>	<b>Decim. ADC</b>	<b>Decim. FPGA</b>	<b>Resamp.</b>	<b>TA Out</b>	<b>Tipo Out</b>
1.	2800	1	1	1	2800	Real
2.	2560	1	1	1	2560	Real
3.	2560	1	1	0.8	2048	Real
4.	3000	4	1	1	750	Complexo
5.	2560	4	1	1	640	Complexo
6.	2560	4	1	0.8	512	Complexo
7.	3000	8	1	1	375	Complexo
8.	2560	8	1	1	320	Complexo
9.	2560	8	1	0.8	256	Complexo
10.	3000	16	1	1	187.5	Complexo
11.	2560	16	1	1	160	Complexo
12.	2560	16	1	0.8	128	Complexo
13.	3000	16	2	1	93.75	Complexo
14.	2560	16	2	1	80	Complexo
15.	2560	16	2	0.8	64	Complexo
16.	3000	16	4	1	46.875	Complexo
17.	2560	16	4	1	40	Complexo
18.	2560	16	4	0.8	32	Complexo
19.	3000	16	8	1	23.4375	Complexo
20.	2560	16	8	1	20	Complexo
21.	2560	16	8	0.8	16	Complexo

### 3.4 Requisitos de Software e Instalação

Para o desenvolvimento e operação da plataforma SKARAB, é necessária uma infraestrutura de software específica, composta por ferramentas proprietárias e pacotes de código aberto mantidos pela colaboração CASPER. A Tabela 2 resume os principais requisitos de software e suas respectivas versões.

A instalação detalhada desses componentes é apresentada nos apêndices desta nota técnica. O Apêndice A detalha a configuração do ambiente Python 2.7 e a instalação do pacote *casperfpga*, essencial para a comunicação e programação da placa. O Apêndice

Table 2: Requisitos de software para a plataforma SKARAB.

Software / Pacote	Versão Requerida
Sistema Operacional	Ubuntu 16.04 LTS
MATLAB	R2018a
Xilinx Vivado	2019.1
Python (Controle)	2.7 (Ambiente <i>casperfpga</i> )
Python (Toolflow)	3.x (Ambiente <i>mlib-devel</i> )
<i>casperfpga</i>	Última versão compatível (Python 2.7)
<i>mlib-devel</i>	Master branch (compatível com Vivado 2019.1)

B descreve o processo de instalação do MATLAB, Vivado e do toolflow *mlib-devel*, necessários para a criação de novos firmwares.

### 3.5 Geração de Bitstream e Programação da FPGA

O processo de transformar um modelo matemático em um hardware funcional na FPGA da SKARAB envolve duas etapas principais: a geração do arquivo de configuração (bitstream) e a transferência deste para a placa. É importante ressaltar que este procedimento é geral para qualquer tutorial ou design desenvolvido para a placa, consistindo nos passos fundamentais de compilação e implementação (*deployment*) do *bitstream*.

#### 3.5.1 Geração de Bitstream

A geração do bitstream é realizada através do ambiente MATLAB/Simulink utilizando o toolflow *mlib-devel*. O fluxo de trabalho segue os seguintes passos:

1. **Modelagem:** O design do firmware é construído no Simulink utilizando blocos da biblioteca CASPER e do Xilinx System Generator.
2. **Configuração:** Definição do hardware alvo (SKARAB) e parâmetros de clock.
3. **Síntese:** O comando `jasper` no MATLAB inicia o processo de compilação automática, que utiliza o Vivado em segundo plano.
4. **Arquivo FPG:** Ao final, um arquivo `.fpg` é gerado, contendo o bitstream e metadados dos registros.

## 4 Tutorial de Introdução ao SKARAB

Este tutorial, baseado na documentação oficial do CASPER<sup>3</sup>, descreve os passos fundamentais para criar, simular e compilar um design básico para a plataforma SKARAB. O objetivo é familiarizar o usuário com o fluxo de trabalho (*toolflow*) e a integração entre MATLAB/Simulink e o hardware.

---

<sup>3</sup>[https://casper-toolflow.readthedocs.io/projects/tutorials/en/latest/tutorials/skarab/tut\\_intro.html](https://casper-toolflow.readthedocs.io/projects/tutorials/en/latest/tutorials/skarab/tut_intro.html)

### Fluxo de Trabalho SKARAB: Da Modelagem à Execução

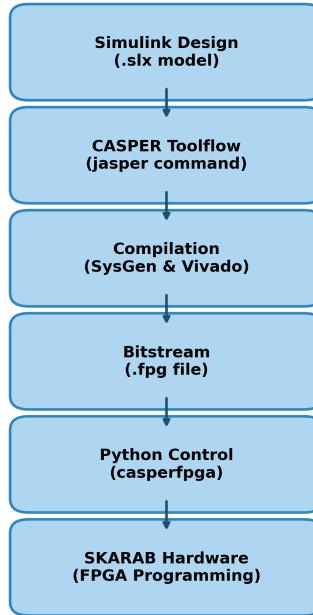


Figure 1: Fluxo de trabalho geral para a plataforma SKARAB, aplicável a qualquer design e tutorial. O processo abrange desde a modelagem em Simulink até a geração do bitstream e programação da FPGA.

#### 4.1 Configuração do Ambiente e Criação do Modelo

O primeiro passo é iniciar o MATLAB através do comando `startsg`, que carrega as bibliotecas CASPER e Xilinx necessárias. No Simulink, cria-se um novo modelo (arquivo `.slx`), observando restrições como evitar espaços ou letras maiúsculas no nome do arquivo e caminhos.

Deve-se adicionar o bloco *System Generator* e o bloco de configuração da plataforma (*XPS Config*), conforme mostrado na Fig. 2. Para a SKARAB, é essencial configurar o clock de IP do usuário (*User IP Clock Source*) para `sys_clk` em designs experimentais iniciais.

#### 4.2 Integração de Blocos e Lógica de Design

Cada design SKARAB deve conter um core de Ethernet para comunicação. Recomenda-se utilizar o template oficial que já inclui o core de 40GbE configurado (Fig. 3).

O tutorial propõe dois exercícios práticos fundamentais:

1. **LED Piscante:** Utiliza um contador de 27 bits, onde o bit mais significativo (MSB) é extraído via bloco *Slice* e enviado a um bloco *GPIO* mapeado para os LEDs da SKARAB (Fig. 4).
2. **Operação de Soma:** Demonstra o uso de registros de software (*Software Registers*) para controle bidirecional. Registros "*From Processor*" fornecem dados para um bloco *AddSub*, e um registro "*To Processor*" permite a leitura do resultado (Fig. 5).

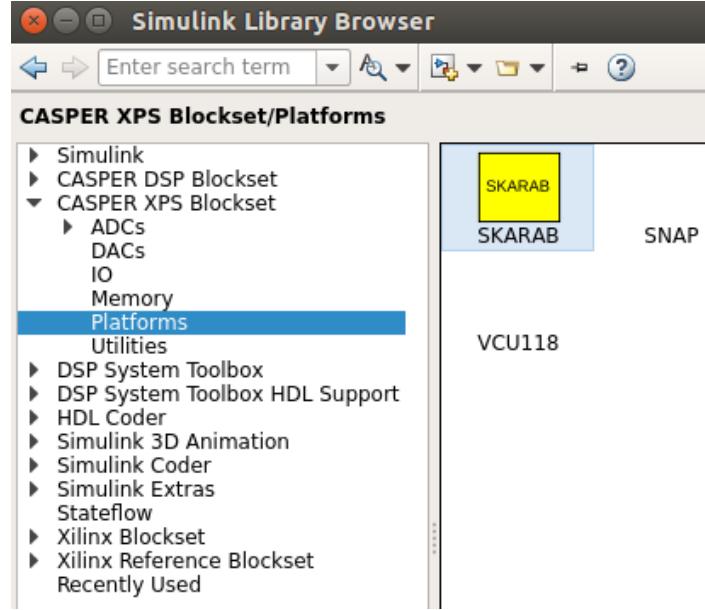


Figure 2: Seleção da plataforma SKARAB no bloco *XPS Config* da biblioteca CASPER.



Figure 3: Subsistema de comunicação Ethernet 40GbE integrado ao modelo.

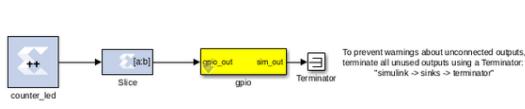


Figure 4: Lógica para controle de LED.

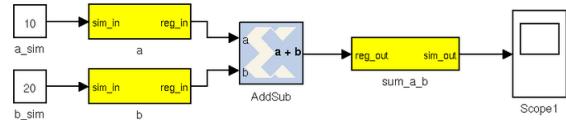


Figure 5: Circuito de soma via software.

### 4.3 Simulação e Compilação

O design pode ser simulado com precisão de clock no Simulink. Após validar o comportamento, executa-se o comando `jasper` no prompt do MATLAB. Este processo invoca o compilador da Xilinx e o Vivado para realizar a síntese e roteamento (Fig. 6), gerando o arquivo *fpg* final na pasta *outputs*.

### 4.4 Programação e Interação via Python

A programação da FPGA é realizada através da biblioteca `casperfpga`. O código abaixo ilustra o procedimento básico de carregamento do firmware e interação com os registros de hardware:

```
import casperfpga
# Conexão com a placa
fpga = casperfpga.CasperFpga('10.42.0.201')
```

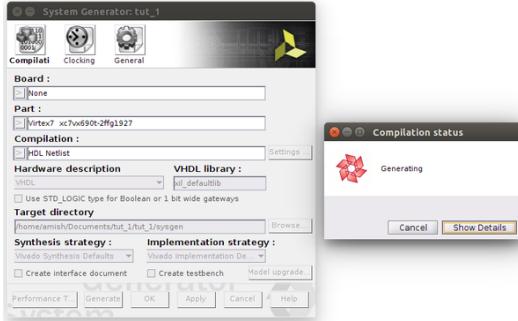


Figure 6: Processo de compilação automática via *toolflow*.

```
# Programação do bitstream
fpga.upload_to_ram_and_program('skarab_tut_intro.fpg')
# Manipulação de registros de software
fpga.write_int('reg_a', 123)
fpga.write_int('reg_b', 456)
resultado = fpga.read_int('sum_a_b')
```

#### 4.4.1 Programação da FPGA

Uma vez gerado o arquivo .fpg, a programação da SKARAB é feita via rede Ethernet (40GbE) utilizando a biblioteca Python *casperfpga*. O procedimento comum realizado no ambiente *IPython* consiste em:

1. Importar a biblioteca: `import casperfpga`.
2. Conectar à placa: `fpga = casperfpga.CasperFpga('IP_DA_SKARAB')`.
3. Carregar o firmware: `fpga.upload_to_ram_and_program('nome_do_arquivo.fpg')`.

Este processo configura instantaneamente a FPGA com o novo design, permitindo o início imediato da aquisição ou processamento de dados.

### 4.5 Espectrómetro (Tutorial 4)

Um espectrômetro digital converte sinais do domínio do tempo para o domínio da frequência, geralmente utilizando o algoritmo de Transformada Rápida de Fourier (FFT). No contexto do BINGO, o design do espectrômetro deve equilibrar a resolução temporal (necessária para detectar eventos rápidos) e a resolução espectral (para mapeamento de HI). Esta seção detalha os blocos fundamentais do design do espectrômetro de banda larga desenvolvido no ecossistema CASPER para a SKARAB.

#### 4.5.1 Digitalização (ADC)

O primeiro passo é a digitalização do sinal analógico. O bloco `skarab_adc` (Fig. 7) opera a uma taxa de amostragem de 3,0 GHz com um fator de decimação interno de 4, resultando em uma taxa efetiva de amostragem de 750 MHz. O sinal é quantizado em 16 bits (formato `fix_16_15`), representando valores entre -1 e 1. A largura de banda total disponível (Nyquist) é de 375 MHz.

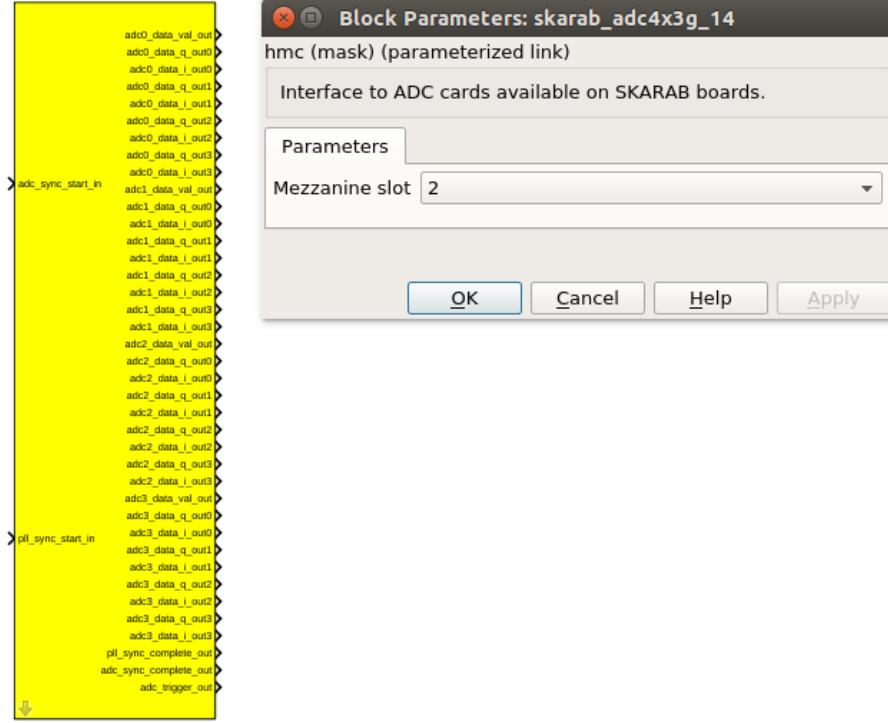


Figure 7: Bloco de interface e parâmetros do ADC da SKARAB (`skarab_adc4x3g_14`).

#### 4.5.2 Transformada de Fourier (FFT)

O bloco `fft` (Fig. 8) é o núcleo do processamento. Ele recebe fluxos complexos demultiplexados do ADC. Para a SKARAB, o design utiliza um FFT complexo projetado para processar amostras em paralelo, permitindo que o sistema opere em tempo real na frequência de clock da FPGA (187,5 MHz).

#### 4.5.3 Cálculo de Potência e Acumulação

Após a FFT, as componentes são enviadas para blocos de potência (`power`, Fig. 9), que calculam o quadrado da magnitude ( $P = re^2 + im^2$ ).

Para reduzir a taxa de dados e melhorar o sinal-ruído, os espectros são integrados temporalmente no bloco `simple_bram_vacc` (Fig. 10). Os dados acumulados são armazenados em memórias BRAM (`mem1`), que são lidas via Python.

## 4.6 Resultados

As primeiras medições usando a SKARAB foram conduzidas no Laboratório de Metrologia (LABMET) em Campina Grande - PB. O primeiro espetro obtido, ilustrado na Fig. 11, foi gerado com decimação 32, o que resulta em um comprimento de banda de 93,75 MHz, centrado em 900 MHz. A configuração básica usada para testes usando a SKARAB está na Fig. 12, composta por: (1) gerador de sinal, (2) SKARAB ADC, (3) fonte de ruído, (4) relógio atômico de rubídio com GPS e (5) analisador de espetro auxiliar.

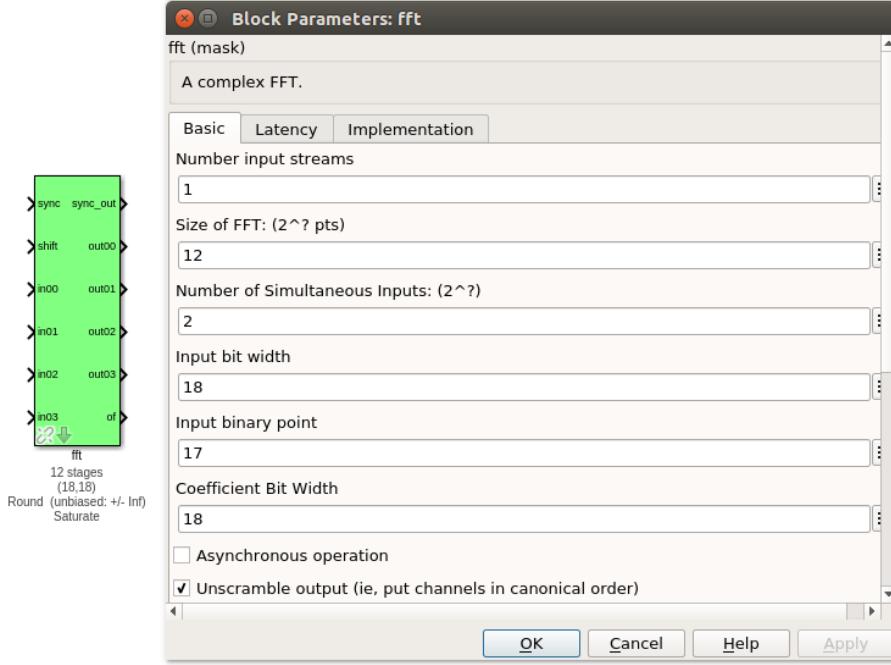


Figure 8: Parâmetros de configuração do bloco FFT ( $\text{Size of FFT}: 2^{12} = 4096$  pontos).

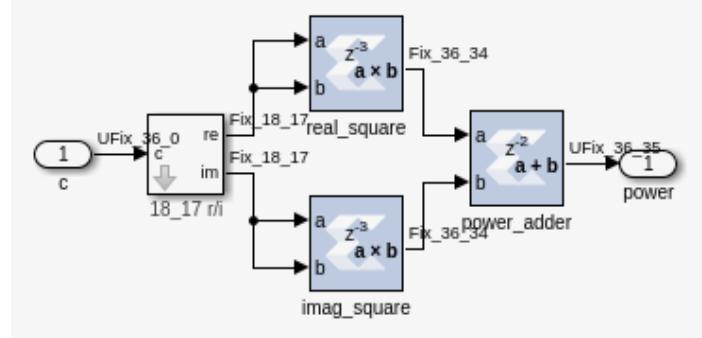


Figure 9: Implementação do cálculo de potência no domínio digital.

## 4.7 O espetrômetro do BINGO e desafios

No desenvolvimento do backend digital para o projeto BINGO, a placa SKARAB foi configurada com sucesso para operar como um espetrômetro em tempo real com largura de banda de 187,5 MHz e resolução de 32.768 bines de frequência. Esta configuração baseia-se no *toolflow* CASPER, integrando blocos de FFT (*Fast Fourier Transform*) de alto desempenho e acumuladores vetoriais. A validação do sistema incluiu a identificação da faixa dinâmica livre de espúrios (SFDR) e a verificação da integridade do sinal em condições de laboratório.

Entretanto, a implementação revelou desafios técnicos significativos. A sincronização do *clock* de amostragem entre o ADC e o processamento na FPGA exige um controle rigoroso de fase e baixos níveis de *jitter* para evitar degradação na sensibilidade do in-

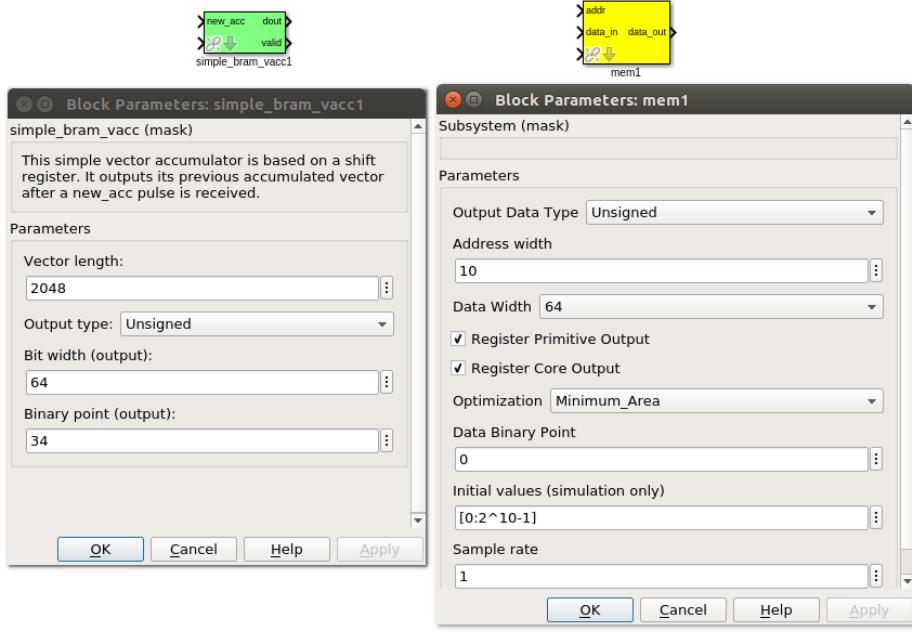


Figure 10: Acumulador vetorial em BRAM e registro de memória para leitura de dados.

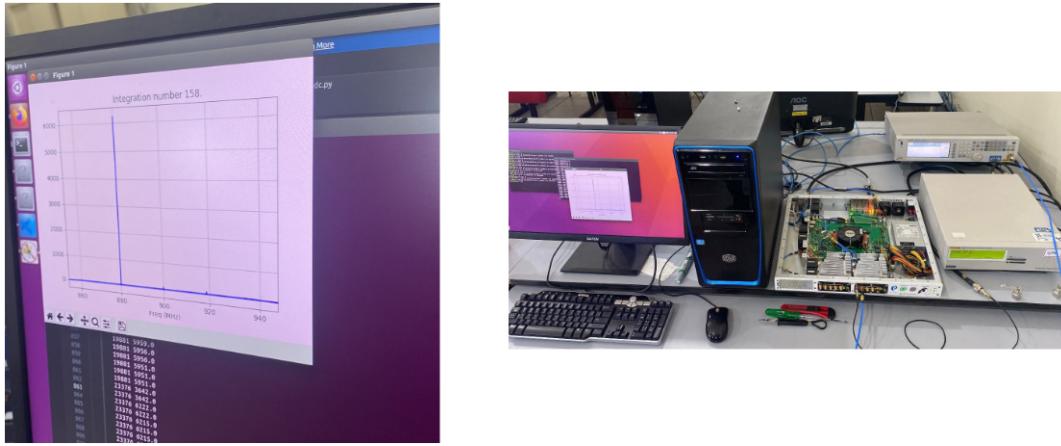


Figure 11: Primeiro espectro obtido pela SKARAB no LABMET.

strumento. Outro ponto crítico é o gerenciamento da alta taxa de transferência de dados via portas 40 GbE. Em regimes de 187,5 MHz, o fluxo de pacotes UDP deve ser otimizado para evitar perdas ou *overflow* nos *buffers* de recepção do servidor de aquisição.

Os próximos passos para o espectrômetro incluem a expansão da largura de banda para 375 MHz (utilizando dual-ADC) e a redução do número de bins para 8.192 para otimizar o uso de recursos de memória BRAM na FPGA. Além disso, estamos desenvolvendo blocos de IP para o gerenciamento de metadados via protocolo SPEAD, garantindo a rastreabilidade temporal dos dados através de *timestamps* de alta precisão sincronizados via sinal PPS (*Pulse Per Second*).

No que diz respeito ao receptor analógico, a corneta Uirapuru foi montada e testada, e a caracterização do amplificador de baixo ruído (LNA) e dos filtros já foi concluída.

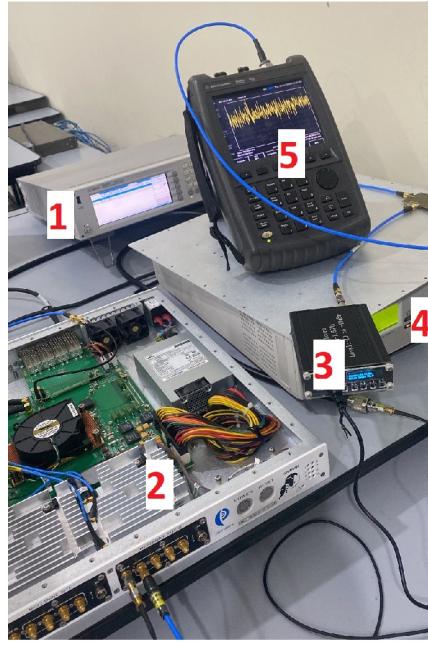


Figure 12: Configuração experimental utilizada nos testes da placa SKARAB, composta por: (1) gerador de sinal, (2) SKARAB ADC, (3) fonte de ruído, (4) relógio atômico de rubídio com GPS e (5) analisador de espectro auxiliar.

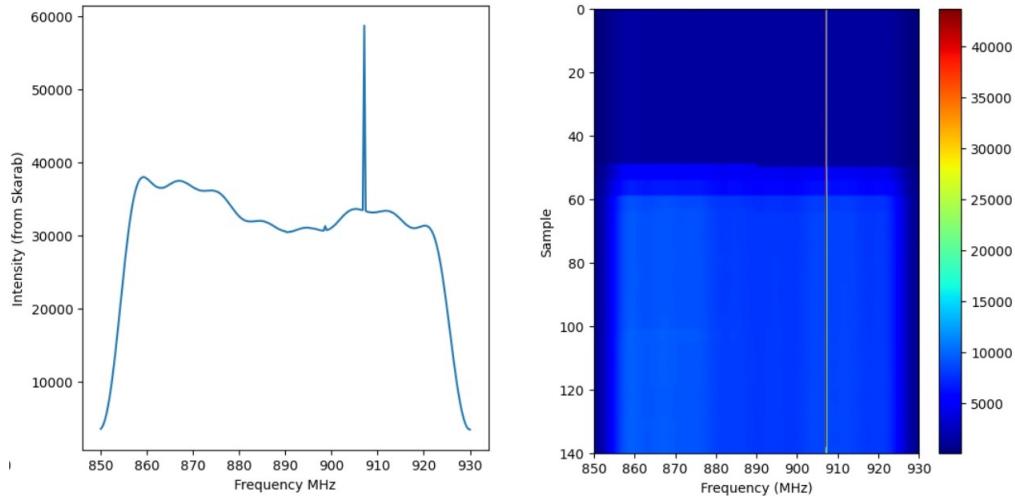


Figure 13: Espectro e waterfall de um sinal contínuo (tom) em 908 MHz com potência de -90 dBm. Após um determinado tempo da inicialização, foi adicionada uma fonte de ruído para observação do efeito sobre o sinal.

Os próximos passos envolvem aprimorar as medições de temperatura de ruído, refinar os métodos de calibração e realizar testes de estabilidade uma vez integrado ao sistema SKARAB.

Os principais desafios estão relacionados à mitigação das flutuações e instabilidades

de ganho por meio de técnicas como os radiômetros Dicke e de pseudocorrelação, bem como garantir a estabilidade e o desempenho de longo prazo do receptor.

O objetivo principal é usar os módulos ADC do SKARAB para digitalizar quatro entradas de RF cada—duas polarizações ortogonais e duas cargas de referência ColFET—enquanto empregamos a comutação Dicke para reduzir as flutuações de ganho.

No que diz respeito à aquisição de dados e geração de arquivos FITS, uma estrutura de cabeçalho FITS já foi projetada, e os dados de espectro foram armazenados com sucesso no formato FITS. Os próximos passos visam incorporar o suporte completo a metadados via pacotes UDP (por exemplo, tempo de sincronização, marcações temporais (timestamps), contadores de heap e de pacotes), bem como a caracterização da temperatura do sistema ( $T_{sys}$ ) e da temperatura de carga de calibração ( $T_{cal}$ ). Os principais desafios incluem o desenvolvimento de software capaz de integrar dados ordenados por tempo (TOD), metadados SPEAD e marcações temporais de alta precisão em arquivos FITS, juntamente com o estabelecimento de um sistema de banco de dados para o gerenciamento de dados em grande escala.

Table 3: Conquistas, Próximos Passos e Desafios do BINGO (Status em 15/01/2026).

Categoría	Principais Conquis-tas	Próximos Passos	Desafios em Curso
Receptor Analógico	<ul style="list-style-type: none"> <li>Montagem da corneta Uirapuru.</li> <li>LNA e filtros testados.</li> <li>S21 medido no LAB-MET.</li> </ul>	<ul style="list-style-type: none"> <li>Temperatura de ruído.</li> <li>Calibração refinada.</li> <li>Estabilidade back-end.</li> </ul>	<ul style="list-style-type: none"> <li>Mitigação de instabilidade.</li> <li>Estabilidade em campo.</li> </ul>
Backend (SKARAB)	<ul style="list-style-type: none"> <li>187,5 MHz / 32k bines OK.</li> <li>SFDR e integridade validados.</li> <li>Toolflow CASPER funcional.</li> </ul>	<ul style="list-style-type: none"> <li>Upgrade 375 MHz dual-ADC.</li> <li>Otimização para 8k bines.</li> <li>Blocos IP para SPEAD.</li> </ul>	<ul style="list-style-type: none"> <li>Sincronização PPS/Phase.</li> <li>Throughput UDP 40 GbE.</li> <li>Multi-board sync.</li> </ul>
Dados (FITS)	<ul style="list-style-type: none"> <li>Cabeçalho FITS OK.</li> <li>Escrita de espectros.</li> <li>Pipeline inicial.</li> </ul>	<ul style="list-style-type: none"> <li>TOD + UDP metadados.</li> <li>Tsys/Tcal automático.</li> </ul>	<ul style="list-style-type: none"> <li>Sincronização de metadados.</li> <li>Redução Big Data.</li> </ul>

A Tabela 3 resume as principais conquistas, os próximos passos planejados e os desafios em curso para o projeto UIRAPURU/BINGO-ABDUS até 15 de janeiro de 2026. As atividades estão organizadas em três categorias: receptor analógico, placa backend digital (SKARAB) e aquisição de dados com geração de arquivos FITS.

No geral, este roteiro destaca progressos significativos em todos os subsistemas, ao mesmo tempo que identifica os desafios técnicos que devem ser abordados para garantir a confiabilidade, estabilidade e escalabilidade do projeto UIRAPURU/BINGO.

## 5 Status atual do trabalho

Começamos esta seção apresentando o estado atual do protótipo do receptor analógico do BINGO. A corneta Uirapuru (horn), ilustrada na Fig. 14, é uma antena corrugada desenvolvida anteriormente [2]. Medindo aproximadamente 1,7 metros de diâmetro e 4,3



Figure 14: Corneta Uirapuru no LABMET, localizado em Campina Grande-PB, Brasil.

metros de comprimento, ela apresenta uma estrutura corrugada composta por 128 anéis coaxiais de liga de alumínio.

Esta corneta servirá como um elemento fundamental do radiotelescópio BINGO, que emprega uma arquitetura de "prato único, muitas cornetas" ("single-dish, many-horns"). O sistema completo está planejado para incluir dois espelhos de 40 metros e aproximadamente 28 antenas cornetas Uirapuru.

A corneta (horn) é conectada aos componentes de front-end, que incluem a transição, o polarizador e os magic-tees. Nesta etapa, as duas polarizações circulares são convertidas em polarizações lineares, mantendo o seu isolamento e independência.

Esses sinais são então alimentados no receptor Uirapuru, que é um radiômetro simples, como ilustrado no diagrama de blocos esquemático na Fig. 15. Aqui, as polarizações circulares de mão esquerda e de mão direita (LHCP e RHCP) são encaminhadas, cada uma, para um acoplador híbrido, onde são combinadas.

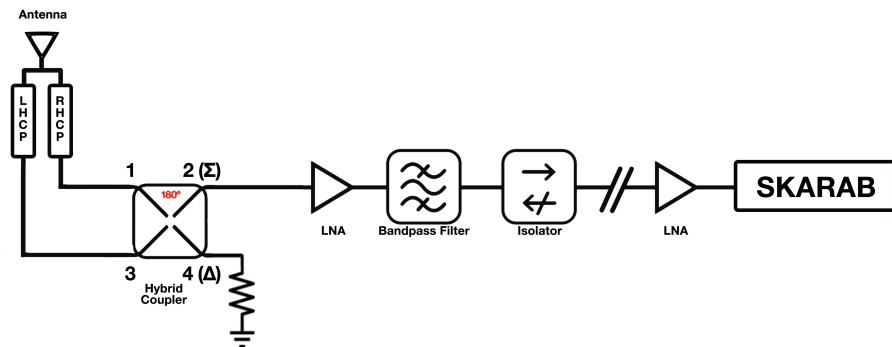


Figure 15: Esquema do receptor do Uirapuru.

O sinal passa então por um Amplificador de Baixo Ruído (LNA), modelo WBA0913AS<sup>4</sup> da WanTcom, que fornece um ganho de 38 dB—amplificando o sinal significativamente enquanto introduz um ruído adicional mínimo. Em seguida, um filtro passa-banda tipo cavidade define a faixa de frequência operacional do BINGO, de 980 MHz a 1260 MHz. As frequências de corte do filtro são determinadas pelas suas características de ressonância. Após isso, o sinal é roteado através de um Isolador, modelo UIY 3434A960T1260SF<sup>5</sup>, para proteger o sistema de interferências externas. Os componentes centrais do backend analógico do protótipo do receptor Uirapuru são mostrados na Fig.16. Como parte de um esforço de caracterização mais detalhado, o parâmetro de transmissão  $S_{21}$  foi medido, conforme ilustrado na Fig.17—um passo que não era comumente incluído em avaliações anteriores. Os resultados demonstram que o receptor tem um desempenho confiável em toda a banda de frequência alvo do BINGO, de 980–1260 MHz.

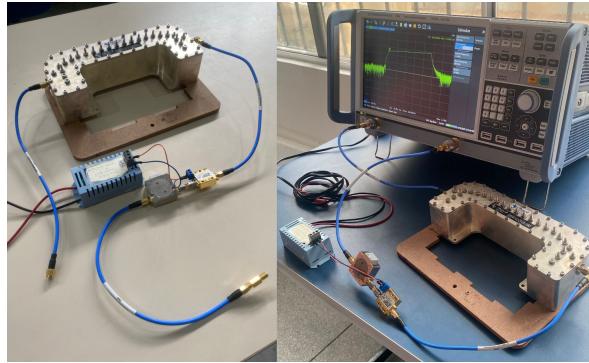


Figure 16: Painel esquerdo: Os principais componentes do backend do receptor Uirapuru, incluindo um filtro passa-banda de 980–1260 MHz, um amplificador de baixo ruído (LNA) com ganho de 38 dB e um isolador. Painel direito: A medição do parâmetro  $S_{21}$  do protótipo do backend analógico do receptor usando um analisador vetorial de redes (VNA) Rohde & Schwarz ZNB.

Outro componente chave do receptor Uirapuru é o acoplador híbrido (veja a Fig.18). Este dispositivo permite a combinação de sinais, produzindo saídas de soma ( $\Sigma$ ) e diferença ( $\Delta$ ), como ilustrado no esquema da Fig.15. Realizamos várias medições, incluindo a análise do parâmetro  $S$  do híbrido, mostrada na Fig. 19. O isolamento é observado nas configurações de portas cruzadas (por exemplo, portas 1–3 e 2–4), enquanto a transmissão ocorre predominantemente entre pares de portas adjacentes, como 1–2, 2–3, 3–4 e 1–4. Portanto, temos as portas de entrada 2 e 4 e as portas de saída soma  $\Sigma$  (porta 3 no painel esquerdo da Fig.18) e a porta de diferença  $\Delta$  (porta 1 no painel esquerdo da Fig.18). Enquanto isso, o parâmetro  $S_{11}$  permanece consistente em todas as portas. Realizamos as medições—including a análise do parâmetro  $S$  dos LNAs, filtro passa-banda tipo cavidade, isolador e híbrido, conforme mostrado nas Figs. 16 e 18.

Finalmente, quando conectamos o híbrido à parte central do receptor, conforme apresentado no esquema da Fig.15, os sinais passam por outro LNA (Amplificador de Baixo Ruído) da WanTcom antes de chegarem ao nosso conversor analógico-digital (ADC) e ao sistema de processamento digital de sinais, que é um SKARAB da Peralex<sup>6</sup>.

Em primeiro lugar, para testar a capacidade de detecção do nosso sistema, a Fig.20

<sup>4</sup><https://www.wantcominc.com/DataSheets/WBA/WBA0913AS.pdf>

<sup>5</sup>UIY – 3434A960T1260SF

<sup>6</sup><https://www.peralex.com/radio-astronomy/>

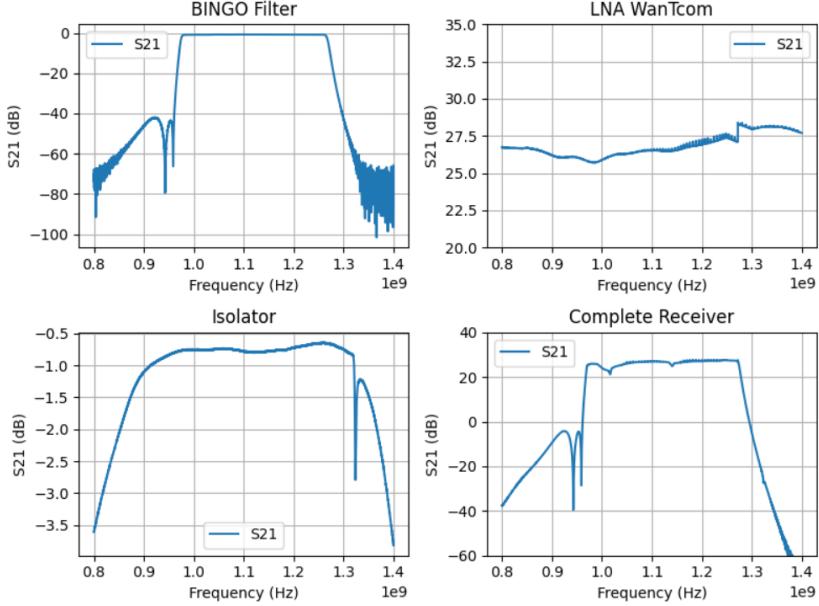


Figure 17: Medições do parâmetro  $S_{21}$  na faixa entre 0,8 e 1,4 GHz para os seguintes componentes: filtro BINGO, LNA WanTcom, isolador e o receptor completo com todos os componentes combinados.

exibe o equipamento básico utilizado para as medições. Os testes envolveram sinais controlados manualmente gerados por um transmissor montado em um drone (Fig.20a) com um rádio portátil (Fig.20b) para produzir Interferência de Radiofrequênci (RFI) intencional. Para o transmissor montado no drone, a detecção de medição do terceiro harmônico do sinal de 400 MHz (ou seja, 1200 MHz) foi realizada usando a corneta Uirapuru (Fig.20c e d).

A Figura 21 ilustra a configuração da medição. Na Fig.21a, a corneta Uirapuru é mostrada ao lado do drone—destacado com uma seta vermelha—voando a uma altitude de 35 metros com o transmissor a bordo. A Fig.21b fornece uma vista de cima da configuração do Uirapuru. Como visto na Fig. 21c (em baixo), o terceiro harmônico do sinal transmitido, centrado em 1200 MHz, é claramente visível tanto no espectro quanto no gráfico waterfall. Além disso, um sinal fraco da banda GPS L5 em 1176 MHz também é detectado. Apesar do grande número de canais FFT utilizados ( $2^{15}$ ), a frequência central do firmware apresenta um comportamento irregular nesta faixa de frequência, provavelmente devido a artefatos de processamento FFT ou sinais de sincronização. No entanto, essas anomalias não afetam a integridade dos resultados gerais. A potência exibida é o logaritmo de base 10 da saída não calibrada do SKARAB (portanto, não possui uma unidade física). A frequência corresponde a uma largura de banda de 187,5 MHz, com frequência central em 1120 MHz.

Após a realização de testes utilizando sinais controlados manualmente a partir de um transmissor montado em um drone, começamos a observar sinais comuns no céu detectados pelo Uirapuru.

Conforme mencionado no parágrafo anterior, a Fig. 22 mostra alguns sinais típicos observados durante um dia inteiro de monitoramento. A potência é representada como o logaritmo de base 10 da saída não calibrada, e a frequência abrange uma largura de banda de 187,5 MHz centrada em 1120 MHz, com correspondência na escala de decibel (dB).

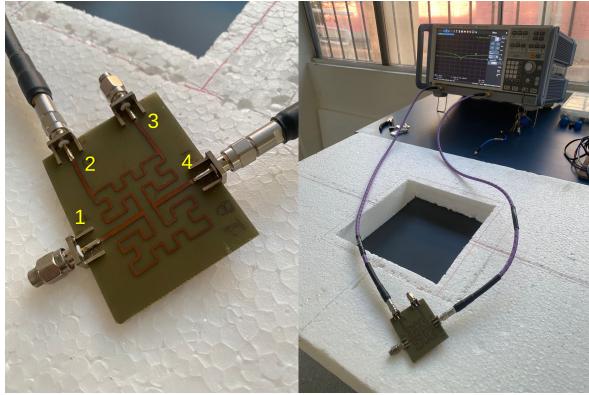


Figure 18: À esquerda: O híbrido fabricado na Universidade Federal de Campina Grande, onde as portas 1, 2, 3 e 4 são indicadas pelos números amarelos. À direita: A medição do parâmetro  $S_{21}$  do híbrido usando o VNA Rohde & Schwarz. As portas de entrada devem ser necessariamente alternadas—ou apenas pares ou apenas ímpares, segundo esta numeração. O isolamento está presente apenas entre estas portas específicas. Portanto, temos as portas de entrada 2 e 4 e as portas de saída soma  $\Sigma$  (porta 3) e a porta de diferença  $\Delta$  (porta 1).

Notavelmente, um forte sinal da banda GPS L5 em 1176 MHz é claramente visível, juntamente com sinais menores em torno de 1200 MHz. Estes sinais são caracterizados por uma aparência suave e um desvanecimento gradual, indicando a relativa aproximação e afastamento da fonte do sinal do ponto de vista da corneta Uirapuru.

A Figura 23 apresenta um dia completo de observação do Uirapuru, correspondente a 9 de maio de 2025. A potência exibida é o logaritmo de base 10 da saída não calibrada do Skarab, com correspondência na escala de decibel (dB).

A frequência abrange uma largura de banda de 187,5 MHz, variando de 1026,25 MHz a 1213,75 MHz, e o período de observação cobre de 00:00 a 23:59 do mesmo dia. A figura revela a passagem de satélites GPS, com sinal visível na banda L5. Adicionalmente, entre 10:00 e 15:00—aproximadamente o meio-dia—é observado um provável trânsito solar em toda faixa de frequência.

## 6 Agradecimentos

O progresso detalhado neste trabalho foi significativamente impulsionado pelo ambiente colaborativo da colaboração BINGO, a quem expresso minha gratidão. Agradecimentos especiais são dirigidos às equipes técnicas do LABMET, sob a coordenação do Prof. Amilcar Queiroz, e ao INPE pelo apoio contínuo e por disponibilizar a infraestrutura de teste e a expertise essenciais. Estendo também meu reconhecimento à FAPESQ-PB pelo seu valioso apoio à pesquisa e ao desenvolvimento no estado da Paraíba.

## A Instalação e Execução do *casperfpga*

O pacote casperfpga é responsável pelo primeiro contato com a SKARAB. A partir dele, é possível utilizar arquivos .fpg já existentes e testar as funcionalidades da placa via Python 2.7, por meio do terminal do computador. Os testes realizados tiveram como objetivo verificar o desempenho do Python 2.7 com a SKARAB, e os resultados foram satisfatórios. Para utilizar a plataforma, é necessário seguir alguns passos para instalar

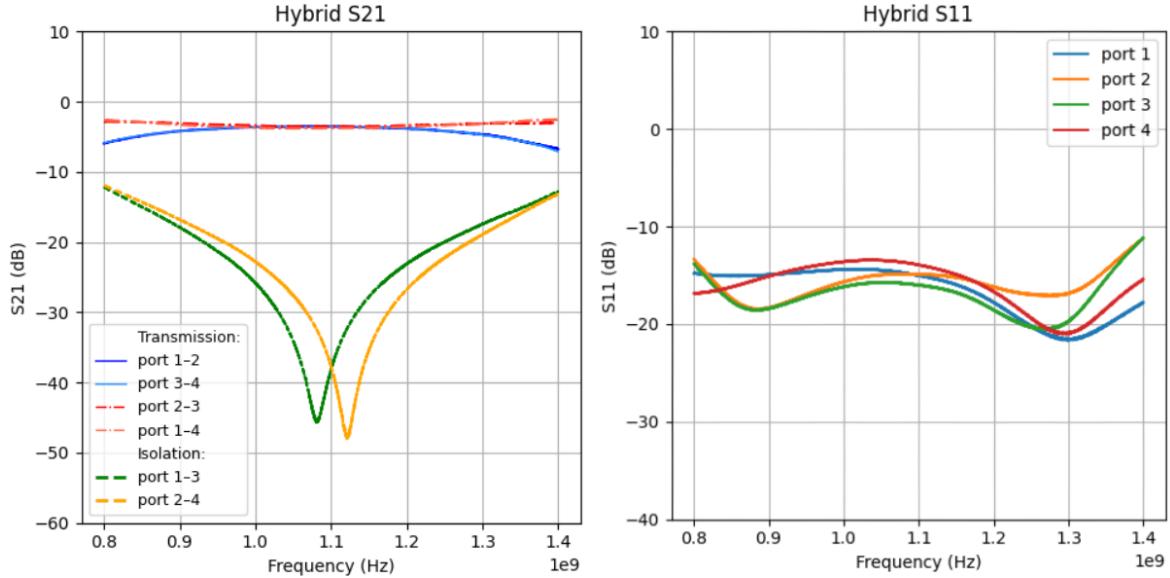


Figure 19: Parâmetros  $S_{21}$  e  $S_{11}$  do acoplador híbrido. As respostas de  $S_{21}$  para as portas 1–2 e 3–4 (ambas mostradas em azul) exibem comportamento semelhante, assim como as respostas para as portas 2–3 e 1–4 (ambas mostradas em vermelho).

o casperfpga em um ambiente Python 2.7. A seguir, são apresentados os procedimentos para instalar o casperfpga do zero no Ubuntu 16.04.

### A.1 Instalação do *casperfpga*

1. Instale o Python 2.7:

```
sudo apt install python2.7
```

2. Instale o `virtualenv` para criar ambientes virtuais:

```
sudo apt install virtualenv
```

3. Crie um ambiente virtual:

```
virtualenv -p python2 cfpca_venv
```

4. Ative o ambiente virtual:

```
source cfpca_venv/bin/activate
```

5. Instale o `git`:

```
sudo apt install git
```

6. Clone o repositório da CASPER:

```
git clone https://github.com/casper-astro/casperfpga
```



Figure 20: a) Drone com transmissor de rádio Baofeng BF-777s, b) Transmissor de rádio Retevis RT85, c) Drone sobrevoando a Uirapuru, d) Uirapuru vista de cima.

7. Entre no repositório da CASPER:

```
cd casperfpga
```

8. Instale alguns requerimentos:

```
sudo apt install build-essential && sudo apt install python-dev
```

9. Instale os requerimentos do `casperfpga`:

```
pip install -r requirements.txt
```

10. Entre na pasta `progska`:

```
cd progska
```

11. Compile o `progska`:

```
make
```

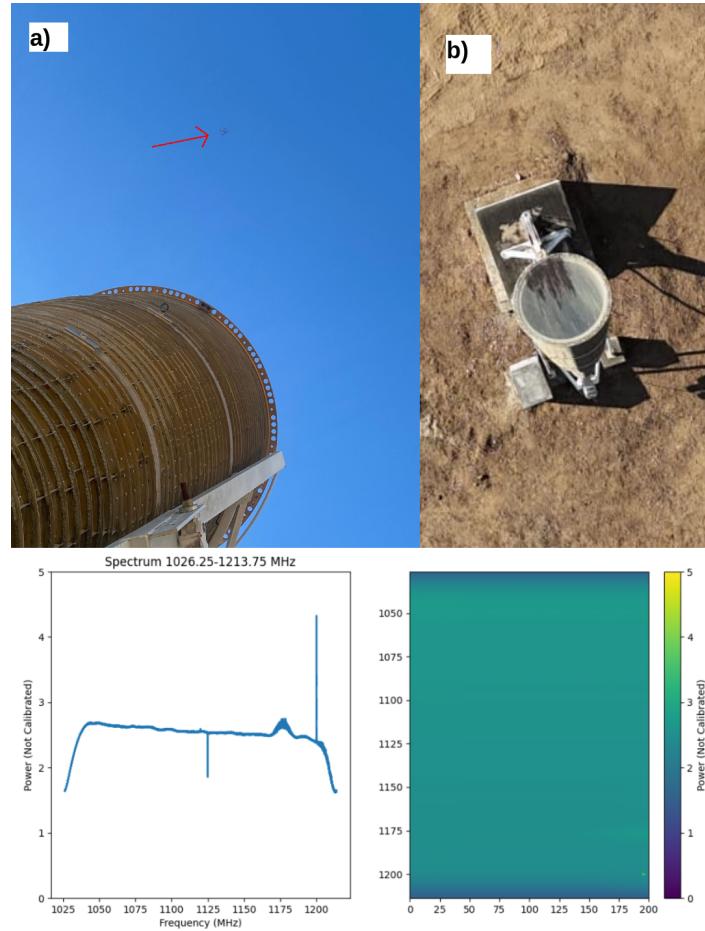


Figure 21: a) Uirapuru e drone com transmissor de rádio Baofeng BF-777s (destacado pela seta vermelha), b) Uirapuru vista de 35 metros acima. c) (em baixo) Apresentamos o espectro e o waterfall do sinal transmitido pelo Baofeng BF-777s acoplado ao drone.

12. Volte para a pasta **casperfpga**:

```
cd ..
```

13. Instale o pacote **casperfpga**:

```
pip install .
```

14. Saia da pasta **casperfpga**:

```
cd ..
```

15. Instale o **matplotlib**:

```
pip install matplotlib
```

16. Instale ferramentas de rede:

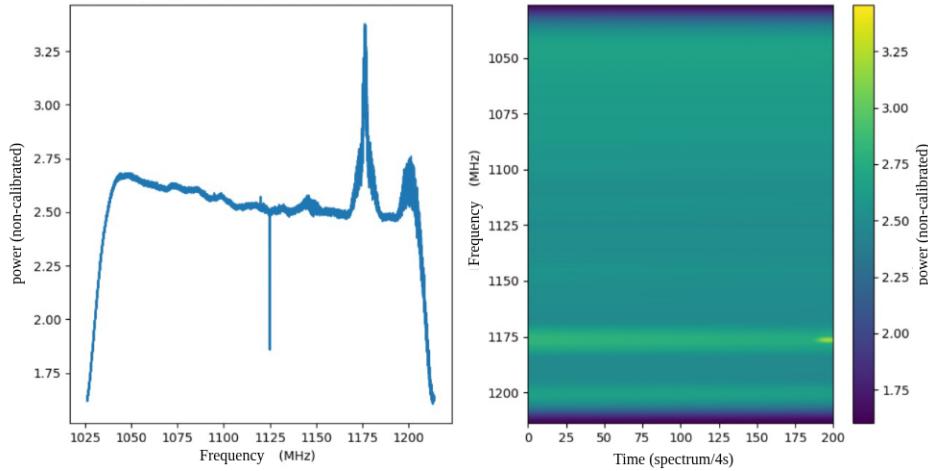


Figure 22: Detecção de Sinais GPS L5 (1176 MHz) e 1200 MHz. O painel esquerdo mostra o espectro de potência com um forte sinal GPS L5 em 1176 MHz e um sinal mais fraco próximo a 1200 MHz. O painel direito exibe o correspondente waterfall plot (diagrama de cachoeira), onde o sinal L5 aparece fraco no início e aumenta em intensidade.

```
sudo apt install net-tools
```

17. Vá em configurações de internet (*Network Settings*) e configure:
  - “Compartilhado para outros computadores” (*Shared to other computers*) para o IPv4;
  - “Ignore” para o IPv6.
18. Configure o MTU para 9000 em “Identidade” (*Identity*) e reinicie a conexão 40GbE após esses passos.
19. Para testar, cheque o IP da SKARAB com conexão 40GbE:

```
arp -a
```

No exemplo, observamos:

```
(10.42.0.201) at 06:50:02:0f:02:01 [ether] on enp1s0f0
```

## A.2 Execução na IDE *IPython*

Após todos esses passos, a SKARAB pode ser operada por meio da IDE *IPython*:

1. Execute o *IPython* (IDE utilizada para operar a SKARAB):

```
ipython
```

(sempre execute com o ambiente Python ativado).

2. Dentro da IDE *IPython*, importe o módulo `casperfpga`:

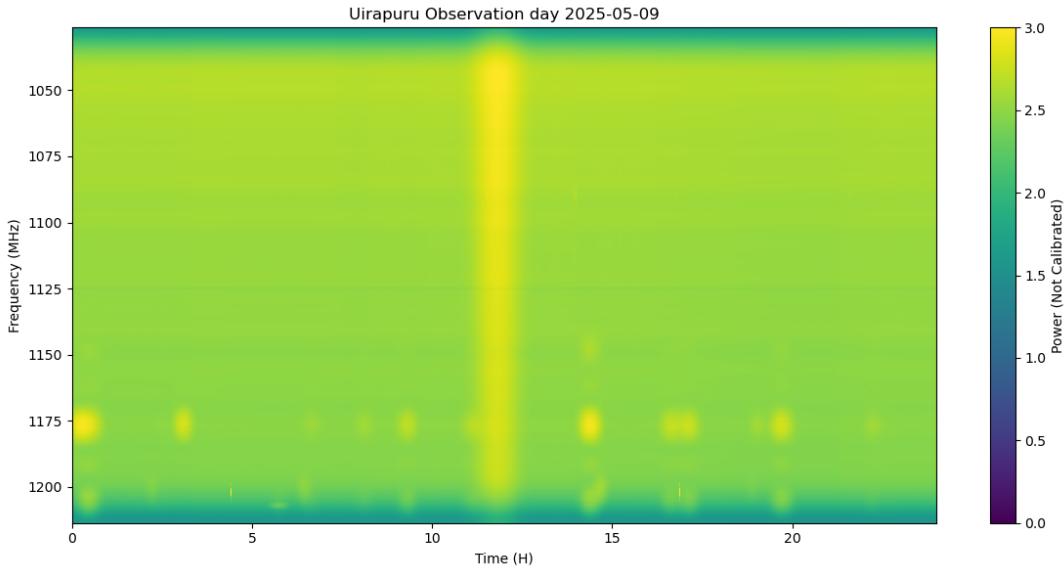


Figure 23: TOD (Dados Ordenados por Tempo) de observação do Uirapuru para o dia 2025-05-09 em 24 horas. No eixo  $y$  são apresentados os canais de frequência em MHz, no eixo  $x$  é apresentado o tempo em horas, e o logaritmo de base 10 da intensidade (potência não calibrada) é apresentado pela barra de cores, onde definimos o valor máximo como três para investigar as informações principais.

```
import casperfpga
```

3. Verifique a versão instalada do `casperfpga`:

```
casperfpga.__version__
```

(observe que são dois sublinhados “\_” antes e depois da palavra `version`, sem espaços).

4. Estabeleça a conexão com a SKARAB:

```
fpga = casperfpga.CasperFpga('10.42.0.201')
```

5. Faça o upload do seu arquivo `.fpg` para a SKARAB:

```
fpga.upload_to_ram_and_program('file.fpg')
```

## B Instalação e Execução do *mlib-devel*

Responsável pela criação de arquivos ".fpg" de autoria própria através do MATLAB R2018a e do Vivado 2019.1 no Ubuntu 16.04. Isso nos permite programar a SKARAB para realização de diversas atividades. Para isso, seguimos com os seguintes passos.

1. Instalação de Dependências Básicas:

Antes de iniciar a instalação do Vivado, é essencial garantir que o sistema possua bibliotecas e pacotes fundamentais. Esses pacotes são necessários para assegurar que o software funcione corretamente, sem conflitos de bibliotecas ou dependências ausentes.

- (a) Biblioteca padrão C++

- Comando:

```
sudo apt-get install libstdc++6
```

Explicação: A biblioteca C++ é necessária para a execução de diversos componentes do Vivado, que dependem de código escrito nesta linguagem.

- (b) Biblioteca GTK+ 2.0

- Comando:

```
sudo apt-get install libgtk2.0-0
```

Explicação: A GTK+ é usada para criar interfaces gráficas e é crucial para garantir que o Vivado e outras ferramentas que usam GUI funcionem corretamente.

- (c) Ferramentas de desenvolvimento dpkg

- Comando:

```
sudo apt-get install dpkg-dev
```

Explicação: O dpkg é o sistema de gerenciamento de pacotes do Ubuntu e suas ferramentas de desenvolvimento são necessárias para compilar ou instalar alguns dos pacotes do software Xilinx.

- (d) Instalação de Python e Pip

- Comando:

```
sudo apt install python3-pip
```

```
sudo apt install python-dev
```

Explicação: Python é frequentemente utilizado para scripts de automação no ambiente de desenvolvimento, e o Pip permite gerenciar pacotes e bibliotecas python.

- (e) Bibliotecas de terminal e ncurses

- Comando:

```
sudo apt install libtinfo5 libncurses5
```

Explicação: Essas bibliotecas são usadas para lidar com a interface de linha de comando e são exigidas por vários programas no ambiente Linux.

- (f) Adicionar suporte ao Qtf4 para a interface gráfica

- Comando:

```
sudo add-apt-repository ppa:rock-core/qt4
```

Explicação: Algumas ferramentas gráficas do Vivado utilizam o framework Qt4, tornando a instalação dessas bibliotecas necessária para uma correta exibição de janelas.

- (g) Instalar bibliotecas Qt4

- Comando:

```
sudo apt update  
sudo apt install libqtcore4 libqtdgui4
```

Explicação: As bibliotecas Qt4 são responsáveis por fornecer os componentes visuais da interface gráfica do usuário.

2. Instalação do **MATLAB R2018a**:

- Baixe o MATLAB R2018a na página da Mathworks.
- Após baixar o MATLAB R2018a, vá para o caminho onde está o instalador:

```
cd /caminho/para/matlab/download/matlab_R2018a_glnxa64.zip
```

- Crie uma pasta para o MATLAB:

```
mkdir matlab_R2018a
```

- Descompacte o .zip:

```
matlab_R2018a_glnxa64.zip -d matlab_R2018a
```

- Entre na pasta:

```
cd matlab_R2018a
```

- Instale o MATLAB

```
sudo ./install
```

- Após a instalação, execute o MATLAB. Digite o seguinte comando:

```
.matlab
```

dentro da pasta /path/to/MATLAB/R2018a/bin  
(no nosso caso: /usr/local/MATLAB/R2018a/bin/).

3. Instalação do **Vivado 2019.1**: Após garantir que o sistema possui todas as dependências básicas, a instalação do Vivado pode ser feita com segurança. Seguem os passos:

- Baixe .zip para instalação do Vivado 2019.1 no site da Xilinx. Após isso descompacte o arquivo no formato .tar.gz: - Comando:

```
tar -xvzf Xilinx_Vivado_SDK_2019.1_0524_1430.tar.gz
```

- Entre na pasta descompactada. - Comando:

```
cd Xilinx_Vivado_SDK_2019.1_0524_1430
```

- Inicie o instalador. - Comando:

```
sudo ./xsetup
```

- Marque a opção "Vivado HL System Edition"

- Também marque a opção: check "Vivado Design Suite" on "Design Tools"

- Após a instalação completa, execute o Vivado para testar com o comando

```
sudo ./vivado
```

dentro da pasta /tools/Xilinx/Vivado/2019.1/bin

- (g) Gere uma licença Vivado, a opção gratuita tem duração de 30 dias. Para inserir a licença, clique em "Help" e "Manage License" dentro do Vivado.
- (h) Outros comandos podem ser necessários para executar o Vivado adequadamente:
  - i. sudo cp -r ~/Downloads/Xilinx.lic /home/bingo/.Xilinx/
  - ii. sudo cp -r ~/Downloads/Xilinx.lic /tools/Xilinx/Vivado/2019.1/data/sysgen/hwcosim\_compiler/pp\_ethernet/
  - iii. sudo cp -r ~/Downloads/Xilinx.lic /tools/Xilinx/Vivado/2019.1/data/ip/core\_licenses/

#### 4. Instalação do **Vitis**

O Vitis é a plataforma unificada da Xilinx para o desenvolvimento de sistemas heterogêneos. Para instalá-lo:

No Vivado, vá até o menu "Help" e selecione "Add Design Tools or Devices" para integrar o Vitis ao ambiente de desenvolvimento.

#### 5. Configuração do Ambiente Python

- (a) Instalar o pacote python3-venv:

```
sudo apt install python3-venv
```

- (b) Criar e ativar um ambiente virtual

```
python3 -m venv casper_venv (para criar)
```

```
source casper_venv/bin/activate (para ativar)
```

Explicação: O ambiente virtual garante que todas as bibliotecas específicas para o projeto estejam isoladas.

#### 6. Instalação do toolflow *mlib-devel*

O mlib-devel é um toolbox do MATLAB/Simulink utilizado para instanciar blocos e pacotes desenvolvidos pela colaboração CASPER, facilitando o projeto e a integração de sistemas digitais em FPGAs.

- (a) Clonar o repositório:

```
git clone -b master https://github.com/peralex/mlib-devel
```

- (b) Entre na pasta mlib-devel:

```
cd mlib-devel
```

- (c) Adicione os seguintes requerimentos no arquivo requirements.txt

```
numpy < 1.9
colorlog
pyyaml
odict
#xml2vhdl requirements
lxml==4.3.0
pyyaml==3.13
-e
git+http://github.com/casper-astro/
xml2vhdl#egg=xml2vhdl_ox-0.2.2-py3.5.egg
&subdirectory=scripts/python/xml2vhdl-ox
```

- (d) Antes de instalar os requerimentos, instale o python3-dev:

```
sudo apt install python3-dev
```

- (e) Instale agora os requerimentos para o mlib\_devel:  
`pip3 install -r requirements.txt`

7. Configure o arquivo startsg.local.example. Coloque as seguintes linhas dentro do arquivo

```
export XILINX_PATH=/tools/Xilinx/Vivado/2019.1
export MATLAB_PATH=/usr/local/MATLAB/R2018a
export PLATFORM=lin64
export JASPER_BACKEND=vivado
export LD_PRELOAD=${LD_PRELOAD}:"/usr/lib/x86_64-linux-gnu/libexpat.so"
export CASPER_PYTHON_VENV_ON_START=/home/bingo/casper_venv
```

8. Execute o mlib\_devel usando o ambiente python3:

```
./startsg startsg.local.example
```

No nosso caso, temos a seguinte expressão no terminal:

```
(casper_venv) bingo@bingo:~/mlib-devel$ ./startsg startsg.local.example
```

## References

- [1] Melis, A., Cabras, A., Comoretto, G., Concu, R., Fiorentini, M., Ladu, A., Maccaferri, A., Migoni, C., Murgia, M., Pilia, M. & Others The SKARAB Board in the Framework of Single-Dish Radio Astronomy. (2024),
- [2] Wuensche, C., Abdalla, E., Abdalla, F. & Others Baryon Acoustic Oscillations from Integrated Neutral Gas Observations: an instrument to observe the 21cm hydrogen line in the redshift range  $0.13 < z < 0.45$ —status update. *Anais Da Academia Brasileira De Ciências*. **93** pp. e20201096 (2021)