

Ryan Wyllyan Ribeiro Inácio

**Implementação do protocolo JESD204B em
plataforma de desenvolvimento Zynq
UltraScale+ MPSoc para controle de ADC e
DAC de alta velocidade**

Brasil

Dezembro de 2024

Ryan Wyllyan Ribeiro Inácio

**Implementação do protocolo JESD204B em plataforma de
desenvolvimento Zynq UltraScale+ MPSoC para controle
de ADC e DAC de alta velocidade**

Monografia referente ao Trabalho Final de
Graduação do curso de Engenharia da Com-
putação da Universidade Federal de Itajubá

Universidade Federal de Itajubá – UNIFEI
Instituto de Engenharia de Sistemas e Tecnologia da Informação
Engenharia da Computação

Orientador: Maurílio Pereira Coutinho
Coorientador: Mateus Mendes Campos

Brasil
Dezembro de 2024

*Este trabalho é para aqueles que veem nos sistemas computacionais não apenas máquinas,
mas portais para a inovação e criatividade.*

Agradecimentos

Este trabalho não poderia ser finalizado sem a ajuda de diversas pessoas às quais presto minha homenagem:

Agradeço aos meus orientadores Maurílio Pereira Coutinho e Mateus Mendes Campos por todo o suporte durante o desenvolvimento.

Agradeço à PS Soluções por fornecer o ambiente e ferramentas necessárias para a execução deste trabalho.

Agradeço aos meus pais Sandra Cristina Ribeiro Inácio e Benedito Valdair Inácio por acreditarem que a educação seria a minha transformação de vida e me apoiarem em todos estes anos de formação.

Agradeço à minha irmã de consideração Jaynne de Cássia Fernandes pela companhia durante toda a minha jornada.

Agradeço às minhas amigas Lívia Braz Carneiro e Letícia Braz Carneiro por tantos anos de amizade.

Agradeço aos meus amigos Arlete Assunção Aguiar Ferraz e José Darcy de Castro Ferraz por todos os conselhos durante a minha formação.

Agradeço aos meus queridos amigos da União Fantástico Rangers, porque sem eles a graduação não teria sido a mesma.

Por fim, agradeço a todos os meus professores, pois é graças a eles que consegui os conhecimentos que me acompanham hoje.

“Eu acredito que às vezes são as pessoas que ninguém espera nada que fazem as coisas que ninguém consegue imaginar.” – Alan Turing

Resumo

Este trabalho tem como objetivo apresentar o projeto desenvolvido para o Trabalho Final de Graduação do curso de Engenharia de Computação da Universidade Federal de Itajubá. O foco foi alcançar uma comunicação de alta velocidade com ADCs e DACs por meio de um sistema que implementa o protocolo JESD204B, utilizando a arquitetura Zynq UltraScale+ MPSoC. Para isso, foram realizados um estudo sobre os temas mencionados, o desenvolvimento do hardware e da aplicação de controle, seguidos pelos testes necessários. Como resultado, foi alcançado um sistema capaz de receber comandos de um usuário por meio de um sistema operacional e, assim, controlar as aquisições e transmissões de sinais, as quais operam em altas taxas de comunicação, comprovando o potencial do protocolo e da arquitetura utilizados.

Palavras-chave: JESD204B. Zynq UltraScale+ MPSoC. Petalinux.

Abstract

This work aims to present the project developed for the Final Graduation Project of the Computer Engineering course at the Federal University of Itajubá. The focus was to achieve high-speed communication with ADCs and DACs through a system implementing the JESD204B protocol using the Zynq UltraScale+ MPSoC architecture. For this, a study on the mentioned topics was conducted, followed by the development of the hardware and control application, and the necessary tests. As a result, a system was achieved that can receive user commands via an operating system and control the signal acquisition and transmission processes, which operate at high communication rates, demonstrating the potential of the protocol and architecture used.

Keywords: JESD204B. Zynq UltraScale+ MPSoC. Petalinux.

Listas de ilustrações

Figura 1 – Diagrama de blocos do projeto.	20
Figura 2 – Versão JESD204.	24
Figura 3 – Versão JESD204A.	25
Figura 4 – Versão JESD204B.	25
Figura 5 – Diagrama de Blocos da placa AD-FMCDAQ2-EBZ.	26
Figura 6 – Camadas da solução HDL JESD204B.	27
Figura 7 – IP ad_ip_jesd204_tpl_adc.	28
Figura 8 – IP ad_ip_jesd204_tpl_dac.	28
Figura 9 – IP axi_adxcvr.	29
Figura 10 – Diagrama de blocos IP axi_dmac.	29
Figura 11 – Diagrama de blocos IP axi_jesd204_rx.	29
Figura 12 – Diagrama de blocos IP axi_jesd204_tx.	30
Figura 13 – Diagrama de blocos IP data_offload.	30
Figura 14 – IP util_adxcvr.	31
Figura 15 – Diagrama de blocos IP util_cpack2.	32
Figura 16 – Diagrama de blocos IP util_upack2.	32
Figura 17 – Regiões PS e PL.	33
Figura 18 – Arquitetura de Hardware do ZynqMP.	35
Figura 19 – Recursos em coluna da PL.	36
Figura 20 – Interfaces AXI PS-PL.	37
Figura 21 – Diagrama de Blocos AXI DMA.	39
Figura 22 – Diagrama de Blocos AXI GPIO.	40
Figura 23 – Diagrama de Blocos AXI Interconnect.	41
Figura 24 – Diagrama de Blocos Concat.	41
Figura 25 – Diagrama de Blocos Constant.	41
Figura 26 – Diagrama de Blocos Processor System Reset.	42
Figura 27 – Diagrama de Blocos Slice.	42
Figura 28 – ZynqMP IP Arquitetura Wrapper.	44
Figura 29 – Fluxo de desenvolvimento Vitis.	45
Figura 30 – Diagrama de blocos da camada Física.	53
Figura 31 – Diagrama de blocos da camada de <i>link</i> do ADC.	54
Figura 32 – Diagrama de blocos da camada de <i>link</i> do DAC.	55
Figura 33 – Diagrama de blocos da camada de transporte do ADC.	55
Figura 34 – Diagrama de blocos da camada de transporte do DAC.	56
Figura 35 – Diagrama de blocos do JESD204B.	57
Figura 36 – Diagrama de blocos do ADC.	57

Figura 37 – Maquina de Estados counter_ceil.	58
Figura 38 – Diagrama de blocos do DAC.	58
Figura 39 – Diagrama de blocos do PS.	59
Figura 40 – Diagrama de blocos do JESD204B e PS.	60
Figura 41 – Tela de configuração do Petalinux.	63
Figura 42 – Gerador de Ondas.	64
Figura 43 – Kit de desenvolvimento com a iW-RainboW-G30M.	65
Figura 44 – Placa AD-FMCDAQ2-EBZ.	65
Figura 45 – Osciloscópio.	66
Figura 46 – Ambiente Teste Completo.	66
Figura 47 – Diagrama de Blocos do Sistema.	67
Figura 48 – Resultados temporização do <i>hardware</i> .	69
Figura 49 – Detalhes do sistema em FPGA pelo Vivado.	69
Figura 50 – Técnicas de Síntese e Implementação do Vivado.	69
Figura 51 – Implementação do sistema em FPGA.	70
Figura 52 – Diagrama de Blocos do Fluxo do ADC.	70
Figura 53 – Ambiente Teste do ADC.	71
Figura 54 – Experimento canal 1 seno de 5 MHz e canal 2 não conectado.	71
Figura 55 – Experimento canal 1 não conectado e canal 2 seno de 5 MHz.	71
Figura 56 – Experimento canal 1 seno de 5 MHz e canal 2 seno de 5 MHz.	72
Figura 57 – Experimento canal 1 onda quadrada 12 MHz e canal 2 seno de 5 MHz.	72
Figura 58 – Experimento canal 1 seno de 5 MHz e canal 2 rampa de 1 MHz.	73
Figura 59 – Visão ampliada dos sinais obtidos pelo ADC.	73
Figura 60 – Diagrama de Blocos do Fluxo do DAC.	74
Figura 61 – Ambiente Teste do DAC.	74
Figura 62 – Experimento com <i>chirp</i> duração de 200 ns.	75
Figura 63 – Experimento com <i>chirp</i> duração de 500 ns.	75
Figura 64 – Experimento com <i>chirp</i> duração de 1 us.	75
Figura 65 – Experimento com <i>ceil</i> igual a 25000.	76
Figura 66 – Experimento com <i>ceil</i> igual a 50000.	76
Figura 67 – Experimento com <i>ceil</i> igual a 100000.	77
Figura 68 – Terminal de Comandos do Petalinux.	78
Figura 69 – Experimento com ADC e DAC - Seno 25 MHz.	78
Figura 70 – Experimento com ADC e DAC - Onda quadrada 12.5 MHz.	79
Figura 71 – Experimento com ADC e DAC - Rampa 1 MHz.	79

Lista de abreviaturas e siglas

ADC	Analog-to-Digital Converter (Conversor Analógico-Digital)
AMD	Advanced Micro Devices
ASIC	Application-Specific Integrated Circuits (Circuitos Integrados de Aplicação Específica)
AXI	Advanced eXtensible Interface (Interface Extensível Avançada)
BRAM	Bloco RAM
DAC	Digital-to-Analog Converter (Conversor Digital-Analógico)
DMA	Direct Memory Access (Acesso Direto à Memória)
FMC	FPGA Mezzanine Card (Cartão Mezanino FPGA)
FPGA	Field Programmable Gate Arrays (Matrizes de Portas Programáveis em Campo)
Gbps	Gigabits por Segundo
GPIO	General Purpose Input/Output (Entrada/Saída de Propósito Geral)
GPU	Graphics Processing Unit (Unidade de Processamento Gráfico)
GSPS	Giga Samples per Second (Giga Amostras por Segundo)
GTH	Gigabit Transceivers High-Speed (Transceptores de Alta Velocidade Gigabit)
HDL	Hardware Description Language (Linguagem de Descrição de Hardware)
IDE	Integrated Design Environment (Ambiente de Desenvolvimento Integrado)
IPs	Intellectual Properties (Propriedades Intelectuais)
I/O	Input/Output (Entrada/Saída)
JEDEC	Joint Electron Device Engineering Council (Conselho de Engenharia de Dispositivos Eletrônicos Conjuntos)
MSPS	Mega Samples per Second (Mega Amostras por Segundo)

PL	Programmable Logic (Lógica Programável)
PS	Processing System (Sistema de Processamento)
SO	Sistema Operacional
SERDES	Serializer/Deserializer (Serializador/Deserializador)
SPI	Serial Peripheral Interface (Interface Periférica Serial)
TCL	Tool Command Language (Linguagem de Comando de Ferramenta)
XSA	Xilinx Shell Archive (Arquivo de Shell Xilinx)
ZynqMP	Zynq UltraScale+ MPSoC

Sumário

1	INTRODUÇÃO	19
1.1	Objetivo	19
1.2	Justificativa	21
1.3	Estrutura do documento	21
2	REVISÃO BIBLIOGRÁFICA	23
2.1	Protocolo JESD204	23
2.1.1	JESD204	23
2.1.2	JESD204A	24
2.1.3	JESD204B	24
2.2	Framework JESD204 Analog Devices	26
2.2.1	Transporte do Receptor	27
2.2.2	Transporte do Transmissor	28
2.2.3	Controle Transceptores	28
2.2.4	Acesso Memória	28
2.2.5	Controle do <i>Link</i> do Receptor	29
2.2.6	Controle do <i>Link</i> do Transmissor	30
2.2.7	Troca de Domínios	30
2.2.8	<i>Link</i> do Receptor	31
2.2.9	<i>Link</i> do Transmissor	31
2.2.10	Transceptores	31
2.2.11	Empacotador	32
2.2.12	Blocos de Memória da FPGA	32
2.2.13	Desempacotador	32
2.3	Arquitetura Zynq UltraScale+ MPSoC	33
2.3.1	PS - <i>Processing System</i>	34
2.3.2	PL - Programmable Logic	34
2.3.3	Comunicação PS e PL	36
2.4	Vivado	38
2.4.1	AXI Direct Memory Access	39
2.4.2	AXI GPIO	40
2.4.3	AXI Interconnect	40
2.4.4	Concat	40
2.4.5	Constant	41
2.4.6	Processor System Reset	42

2.4.7	Slice	42
2.4.8	Zynq UltraScale+ MPSoc	42
2.5	Vitis	43
3	METODOLOGIA	47
3.1	Desenvolvimento <i>Hardware</i>	47
3.1.1	Estudo dos IPs	47
3.1.2	Construção dos Blocos	47
3.1.3	Construção das Camadas	47
3.1.4	União das camadas	48
3.1.5	Exportação o <i>Hardware</i>	48
3.2	Desenvolvimento do <i>Software</i>	48
3.2.1	Criação da Plataforma	48
3.2.2	Estudos dos Códigos	48
3.2.3	Reestruturação dos Códigos	48
3.2.4	Depuração	49
3.2.5	Adaptação para Sistema Operacional	49
3.3	Teste em Bancada	49
4	DESENVOLVIMENTO PRÁTICO	51
4.1	Sistema em FPGA	51
4.1.1	Camadas do JESD204B	51
4.1.1.1	Camada Física	52
4.1.1.2	Camada de <i>Link</i>	52
4.1.1.3	Camada de Transporte	54
4.1.2	Camadas de Aplicação	56
4.1.2.1	Camada Aplicação Receptor	56
4.1.2.2	Camada Aplicação Transmissor	56
4.1.3	Bloco de Processamento	59
4.1.4	Bloco Superior	60
4.2	Aplicação de controle	60
4.2.1	Códigos Estudados	61
4.2.2	Plataforma de Hardware e Reestruturação	62
4.2.3	Compilação Petalinux e Adaptação Projeto	62
4.3	Ambiente de Testes	64
4.3.1	Gerador de Ondas	64
4.3.2	Placa iw-RainboW-G30M	64
4.3.3	Placa AD-FMCDAQ2-EBZ	65
4.3.4	Osciloscópio	66
4.3.5	Ambiente de Teste Completo	66

5	EXPERIMENTOS E RESULTADOS	67
5.1	Configuração do sistema	67
5.2	Hardware em FPGA	68
5.3	Sistema <i>Bare Metal</i>	69
5.3.1	Experimentos ADC	69
5.3.2	Experimentos DAC	74
5.4	Sistema com Petalinux	77
6	CONCLUSÃO	81
	REFERÊNCIAS	83

1 Introdução

O protocolo JESD204 visa realizar a conexão entre conversores e receptores, além de proporcionar uma alta velocidade de comunicação. Sendo assim, ressalta-se que, devido aos avanços tecnológicos, evidencia-se uma melhora considerável dos recursos disponibilizados por conversores analógico-digital (ADCs) e digital-analógico (DACs). A exemplo, tem-se maiores resoluções e alta velocidade de conversão, o que implica na necessidade de interfaces robustas para processar os dados. Portanto, utiliza-se o protocolo JESD204 para realizar essa tarefa, haja vista suas significativas vantagens sobre as tecnologias utilizadas anteriormente, como CMOS e LVDS. Dentre os diferenciais deste protocolo, pode-se citar a paridade com as taxas de amostragem dos conversores, além de uma menor área de placa devido à diminuição de pinos. ([HARRIS, 2013](#))

Ademais, a versão B do protocolo JESD204 suporta taxa de dados seriais de até 12.5 Gbps (Giga bits por segundo), além de contar com 4 camadas principais: transporte, embaralhamento (opcional), *data link* e física. Ressalta-se que a camada física é responsável por transmitir e receber os dados à taxa de bits citada, sendo assim, ela é implementada por meio dos serializer/deserializer (SERDES) presentes em Field Programmable Gate Arrays (FPGAs). ([FAN, 2015](#))

Também, segundo o grupo de engenharia de Aplicações da Analog Devices ([GROUP ANALOG DEVICES, 2014](#)), interligar a saída dos conversores a FPGAs é um desafio comum de engenharia. Além de ressaltarem que, quando se trata do protocolo JESD204, há uma redução significativa dos requisitos de interligações, mas à custa de recursos sofisticados em FPGA e placas de circuito impresso.

Então, ao realizar projetos que necessitem de uma FPGA, segundo KRUPA ([KRUPA, 2020](#)) tem-se a arquitetura ZYNQ como uma boa escolha, haja vista que esta conta com uma região equivalente ao sistema de processamento de um microcontrolador e outra que equivale ao tecido lógico de uma FPGA. Sendo assim, ela possui a capacidade equivalente à de sistemas micro-controlados como os processadores da arquitetura ARM. Além de todas as vantagens de flexibilidade e recursos oriundos de uma FPGA. Logo, esta se torna uma arquitetura com um processamento, flexibilidade e paralelismo ideais para sistemas de controle complexos.

1.1 Objetivo

Este trabalho visa utilizar a arquitetura Zynq Ultrascale+ MPSoc (ZynqMP) para implementar em FPGA um hardware que contenha as camadas do protocolo JESD204B.

Para que esta estrutura possibilite a comunicação em Gbps com chips ADCs e DACs, bem como seu controle.

Também objetiva-se desenvolver uma aplicação que será responsável por configurar todas as camadas do protocolo presentes no hardware descrito, bem como configurar os parâmetros do chip de conversão. E, após todas as configurações iniciais, realizar a aquisição/transmissão de dados.

Ademais, pretende-se configurar e compilar uma distribuição do sistema operacional (SO) Petalinux, a fim de obter um SO embarcado capaz de executar a aplicação de controle desenvolvida.

Sendo assim, considerando todos os pontos explicitados, o sistema objetivado é demonstrado pelo diagrama de blocos presente na Figura 1. Vale ressaltar que todos os detalhes do sistema serão apresentados no decorrer deste documento.

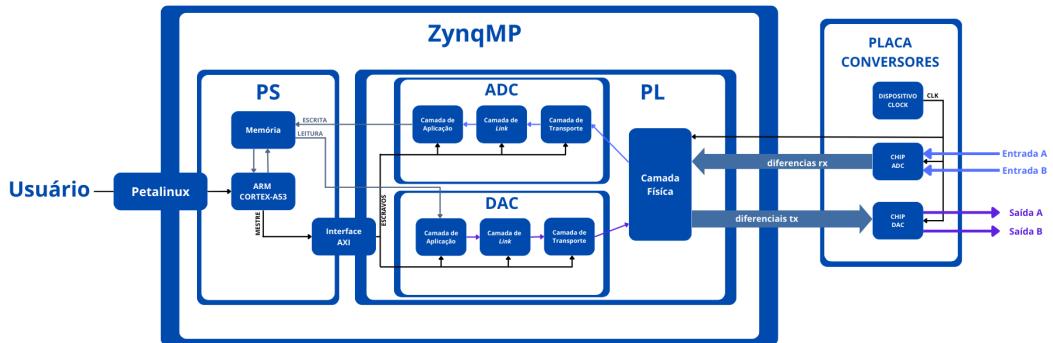


Figura 1 – Diagrama de blocos do projeto.

Fonte: Própria.

Por fim, os objetivos específicos consistem em:

- Estudar e compreender os princípios fundamentais do FPGA Zynq Ultrascale+ MPSoc, sua arquitetura heterogênea, processadores ARM integrados, linguagem de descrição de hardware (Verilog) e ferramentas de desenvolvimento (Vivado, Vitis).
- Pesquisar e analisar as especificidades do sistema de controle do ADC e DAC em FPGA, bem como as funcionalidades relacionadas a cada camada do protocolo JESD204B.
- Implementar as camadas relacionadas a comunicação e controle do ADC em FPGA ZynqMP.
- Implementar as camadas relacionadas a comunicação e controle do DAC em FPGA ZynqMP.

- Realizar a integração dos dois sistemas desenvolvidos.
- Realizar simulações detalhadas dos projetos desenvolvidos para verificar o correto funcionamento do sistema de controle do ADC e DAC.
- Utilizar ferramentas de simulação e depuração para validar os circuitos e algoritmos implementados.
- Identificar oportunidades de otimização de desempenho no sistema de controle, visando reduzir latências e aumentar eficiência energética.
- Utilizar um SO para atingir uma aplicação de controle que possibilite uma interface de fácil acesso ao usuário.

1.2 Justificativa

Quando se trata da interface de conexão com conversores de alta velocidade, o protocolo JESD204B apresenta características que atendem aos requisitos necessários para realizar o controle e comunicação com esses chips. Contudo, no que se refere à implementação do protocolo, é preciso uma estrutura com alta flexibilidade. Além disso, é fundamental um processamento compatível com a sua alta velocidade. Então, necessita-se de uma FPGA e um sistema de processamento, o que é oferecido pela arquitetura ZynqMP.

Portanto, a motivação do projeto aqui proposto encontra-se no desafio de utilizar a tecnologia Zynq Ultrascale+ MPSoC para implementar o JESD204B e tornar possível a comunicação entre uma placa de desenvolvimento e chips ADCs e DACs.

1.3 Estrutura do documento

Este documento visa detalhar o Trabalho Final de Graduação do curso de Engenharia de Computação da Universidade Federal de Itajubá, logo para um melhor entendimento dos assuntos aqui expostos, o documento será dividido em cinco capítulos, sendo estes: Revisão Bibliográfica, Metodologia, Desenvolvimento Prático, Experimento e Resultados e a Conclusão.

Desta forma, a Revisão Bibliográfica irá discutir as bases teóricas da pesquisa e as principais ferramentas utilizadas. Logo, serão apresentadas as características do funcionamento do protocolo JESD204, o *Framework* da Analog Devices que suporta a implementação do JESD204, as especificidades da arquitetura ZynqMP, as ferramentas do Vivado para criação do *hardware* em FPGA e a plataforma Vitis para desenvolvimento do *software*.

Já no capítulo de Metodologia, serão evidenciadas três etapas adotadas para o desenvolvimento do projeto aqui apresentado. Sendo estas: Desenvolvimento do *Hardware*, Desenvolvimento do *Software* e o Teste em Bancada. Vale ressaltar que o capítulo do Desenvolvimento Prático é uma complementação ao da Metodologia, haja vista que este demonstrará as atividades realizadas para o desenvolvimento do sistema em FPGA, o desenvolvimento da aplicação de controle e a montagem do ambiente de testes.

Ademais, o capítulo de Experimentos e Resultados demonstra a configuração utilizada para os experimentos, os resultados esperados, a execução dos testes realizados, bem como os sinais obtidos com o sistema desenvolvido. Por fim, o capítulo de Conclusão apresenta os comentários e discussões finais sobre os conhecimentos adquiridos com o trabalho.

2 Revisão Bibliográfica

Este capítulo visa explicitar os conceitos e ferramentas, os quais são utilizados para o desenvolvimento do projeto aqui apresentado. Logo, as próximas seções concentram-se em explicar o protocolo JESD204, o *Framework* da Analog Devices, a arquitetura Zynq Ultrascale+ MPSoC, o Vivado e o Vitis.

2.1 Protocolo JESD204

Os barramentos estão presentes desde o primeiro computador, o ENIAC, e estes foram evoluindo a partir da necessidade de maiores velocidades de comunicação. Contudo, segundo [ZHANG \(2020\)](#), tem-se um consenso na área da engenharia de que os barramentos paralelos estavam limitando a comunicação em sistemas de alta velocidade, haja vista a largura de transmissão de dados desta estrutura. Além disso, aumentar a largura de banda trata-se apenas de uma possibilidade teórica, já que esta mudança é limitada pela indústria. Desta forma, a utilização de canais seriais tornou-se a solução para a resolução desta limitação.

Posto isto, no que se refere a comunicação com conversores ADC e DAC, o problema do paralelismo é evidente quando se trata da sincronização de dados. Isto porque estes chips costumam ter precisão de 12 a 16 bits, o que dificulta que as linhas de dados sejam capturadas pelo mesmo *clock* ([ZHANG, 2020](#)). Logo, tendo em vista a tendência da substituição de estruturas paralelas por seriais, a organização internacional JEDEC (Joint Electron Device Engineering Council) lançou em 2006 o protocolo JESD204. Este protocolo adotou um par de sinais diferenciais com CML (Current Mode Logic) ao invés dos 12 a 16 bits paralelos como utilizado anteriormente ([HARRIS, 2013](#)).

A introdução do protocolo JESD204 no mercado reduziu significativamente o número de pinos para dispositivos lógicos. Tal mudança também proporcionou um roteamento mais simples, menores dispositivos ADC/DAC e um menor consumo de energia. Vale ressaltar que a adesão da indústria a este protocolo provocou o lançamento de novas versões para atender aos requisitos necessários. Sendo assim, nos tópicos seguintes serão apresentados os detalhes da versão de lançamento até a versão B que é utilizada neste projeto.

2.1.1 JESD204

Como supracitado, a versão JESD204 foi a primeira a ser lançada. Pode-se citar que seu objetivo consiste em ser um *link* de dados entre conversores e receptores como

FPGAs ou ASICs (*Application-Specific Integrated Circuits*). Esta versão conta com apenas uma via entre os dispositivos, como é apresentado pela Figura 2. Além disso, possui uma taxa de transferência no intervalo de 312,5 megabits por segundo a 3,125 gigabits por segundo. (HARRIS, 2013)

Outro ponto importante a se ressaltar é a codificação 8b/10b, ou seja, mapeia símbolos de 8 bits para símbolos de 10 bits. Esta codificação visa o equilíbrio DC e disparidade limitada, e ainda fornece mudanças de estado suficientes para permitir uma recuperação razoável do *clock*, haja vista seu *clock* embarcado. Esta funcionalidade removeu o problema associado à necessidade de uma nova linha de *clock* adicional, tendo em vista a complexidade associada ao roteamento do *clock* quando utilizadas altas taxas de transferências de dados.(ZHANG, 2020)

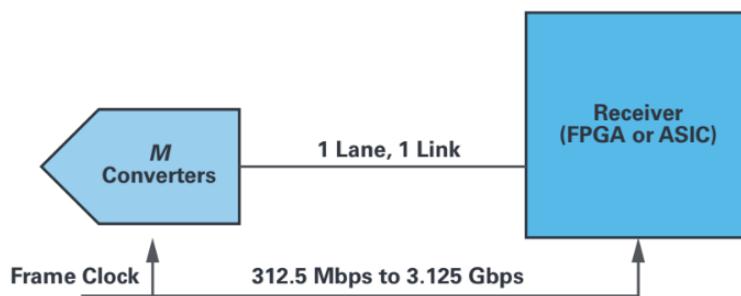


Figura 2 – Versão JESD204.

Fonte: Retirado de HARRIS (2013).

2.1.2 JESD204A

Com o aumento do uso do protocolo, a versão JESD204A foi lançada em 2008. Esta versão trouxe, como atualização, um maior número de canais entre os conversores e os dispositivos lógicos (ZHANG, 2020). Na Figura 3 é apresentado um diagrama que ilustra o suporte aos múltiplos canais.

Ressalta-se que as especificações da primeira versão foram mantidas. Mas, apesar de manter a mesma taxa de transferência de dados, a inclusão do suporte a múltiplas vias seriais alinhadas tornou possível que os conversores com maiores taxas de amostragem e maiores resoluções atingissem a máxima taxa de 3,125 Gbps. (HARRIS, 2013)

2.1.3 JESD204B

A versão JESD204B (Figura 4) foi lançada em 2011 e trouxe mudanças significativas para o protocolo. Dentre elas, tem-se o aumento da máxima taxa de transferência de dados para 12,5 Gbps e a mudança do *frame clock* para o *clock* do dispositivo como fonte principal. (ZHANG, 2020)

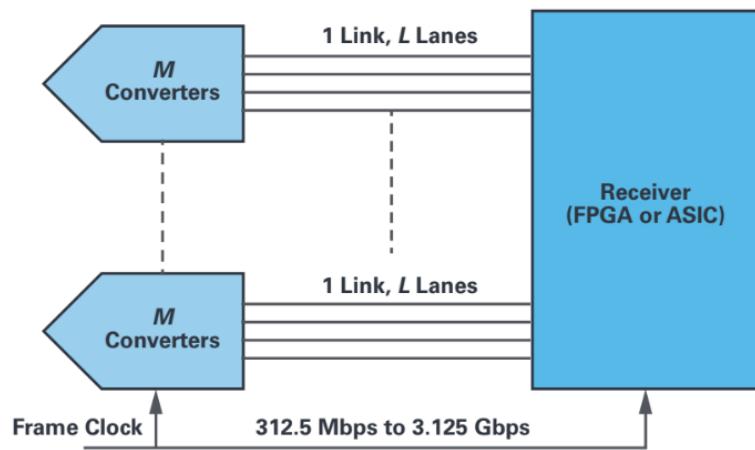


Figura 3 – Versão JESD204A.

Fonte: Retirado de [HARRIS \(2013\)](#).

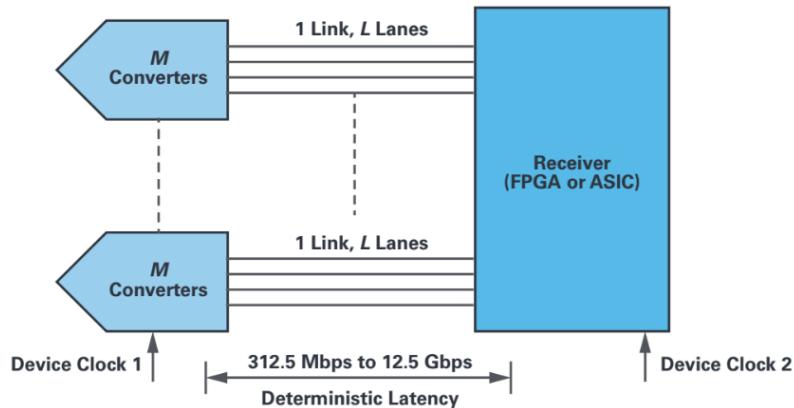


Figura 4 – Versão JESD204B.

Fonte: Retirado de [HARRIS \(2013\)](#).

Mas, a principal característica desta versão é sua capacidade de prever latências determinísticas. Isto foi possibilitado pelo auxílio de um sinal de entrada que visa sincronizar todas as vias de dados. Além disso, esta solução foi implementada de distintas maneiras, possibilitando que a versão JESD204B fosse dividida em três subclasses, sendo estas apresentadas a seguir. ([HARRIS, 2013](#))

- **Subclasse 0:** Sem suporte para latência determinística.
- **Subclasse 1:** Utiliza o sinal de SYSREF para latência determinística (recomendada para operações acima de 500 *Mega Samples Per Second (MSPS)*).
- **Subclasse 2:** Utiliza o sinal de SYNC~ para latência determinística (recomendada para operações inferiores a 500 MSPS).

2.2 Framework JESD204 Analog Devices

A empresa Analog Devices é uma participante ativa da organização JEDEC e colaboradora no desenvolvimento do protocolo JESD204. Além disso, ela é uma empresa de tecnologias e ferramentas de conversão de dados, nas quais implementa o protocolo supracitado. Tem-se como exemplo a placa AD-FMCDAQ2-EBZ (Figura 5) que conta com conversores que já possuem o JESD204B integrado, além de um chip gerador de *clock* ([Analog Devices, 2023b](#)). E o ponto importante a ser destacado é que toda esta estrutura é controlada por meio de uma FPGA. Isso porque todos os sinais de controle e dados dos chips são externalizados por um conector FMC (*FPGA Mezzanine Card*) que é conectado a uma FPGA.

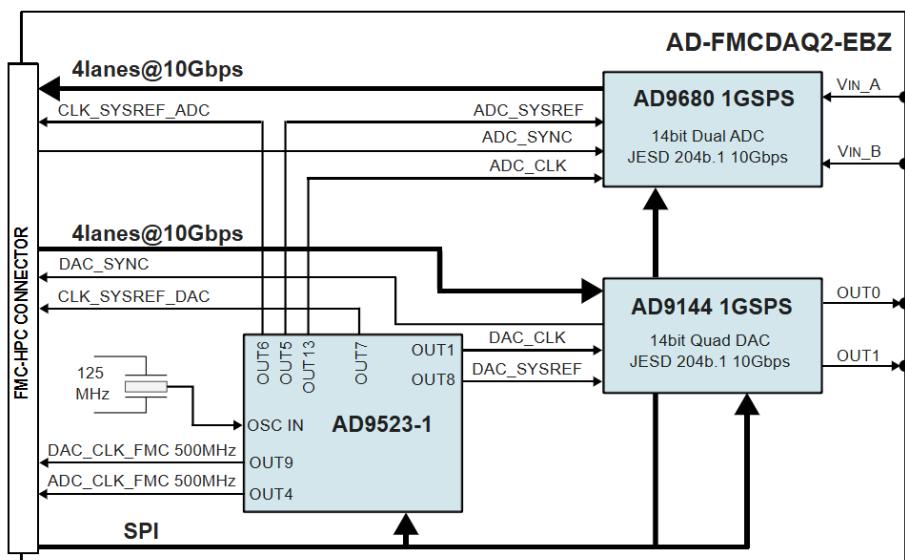


Figura 5 – Diagrama de Blocos da placa AD-FMCDAQ2-EBZ.

Fonte: Retirado de [Analog Devices \(2023b\)](#).

Logo, para que os consumidores dos produtos da Analog Devices sejam capazes de desenvolver designs de sistemas em FPGA capazes de utilizar todos os recursos disponíveis pelo protocolo JESD204B, é fornecida uma solução *full-stack* como suporte, sendo esta denominada JESD204 Interface *Framework*. ([Analog Devices, 2024h](#))

A JESD204 Interface *Framework* projetou os componentes em conjunto para fornecer uma melhor interoperabilidade. Além disso, esta cobre diferentes aspectos para auxiliar no design de sistemas, dentre eles pode-se citar:

- **Hardware:** fornece referência a rápidos protótipos de sistemas.
- **Hardware Description Language (HDL):** Intellectual Properties (IPs) para utilizar o protocolo JESD204.
- **Software:** Drivers para administrar os chips de clock, os conversores e o HDL.

Também, cabe ressaltar que o padrão JESD204B define múltiplas camadas e a solução HDL JESD204B da Analog Devices fornece um modelo de 4 camadas (Figura 6), as quais são apresentadas e detalhadas abaixo. ([HIGUERUELA, 2020](#))

- **Camada física:** responsável pela interface e configuração com os pinos *high-speed*.
- **Camada de link:** responsável por lidar com o protocolo no que se refere ao embalhamento/desembalhamento, alinhamento de faixa, substituição de caracteres e monitoramento de alinhamento.
- **Camada de transporte:** responsável pelo enquadramento e desenquadramento de dados específicos do conversor.
- **Camada de aplicação:** definida pelo usuário para fazer o processamento de sinais.

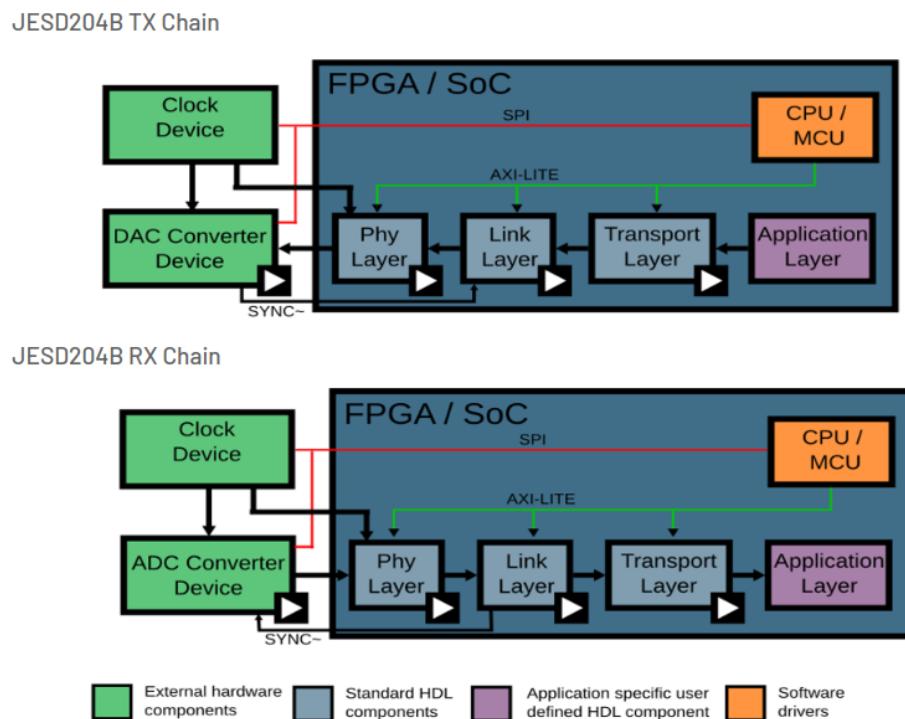


Figura 6 – Camadas da solução HDL JESD204B.

Fonte: Retirado de [Analog Devices \(2024h\)](#).

Por fim, dentre as ferramentas oferecidas, serão apresentados a seguir os IPs da Analog Devices que foram utilizados na implementação do sistema aqui descrito.

2.2.1 Transporte do Receptor

O IP ad_ip_jesd204_tpl_adc (Figura 7) implementa o nível de transporte de um receptor de dados. Tal bloco é capaz de realizar o desenquadramento dos dados e, também,

pode ser configurado em tempo de execução por meio da interface AXI. ([Analog Devices, 2024a](#))

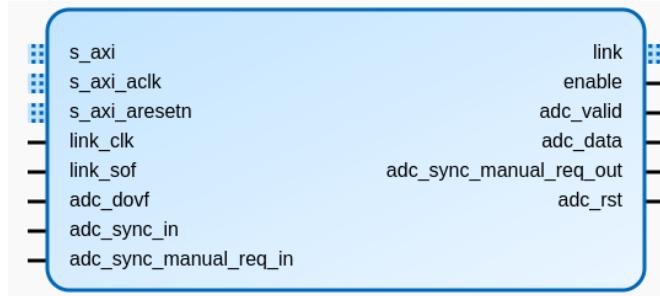


Figura 7 – IP ad_ip_jesd204_tpl_adc.

Fonte: Retirado de [Analog Devices \(2024a\)](#).

2.2.2 Transporte do Transmissor

O IP ad_ip_jesd204_tpl_dac (Figura 8) implementa o nível de transporte de um transmissor de dados. Tal bloco é capaz de realizar o enquadramento dos dados, além de conseguir gerar dados para validação da interface. Também é configurável em tempo de execução. ([Analog Devices, 2024f](#))

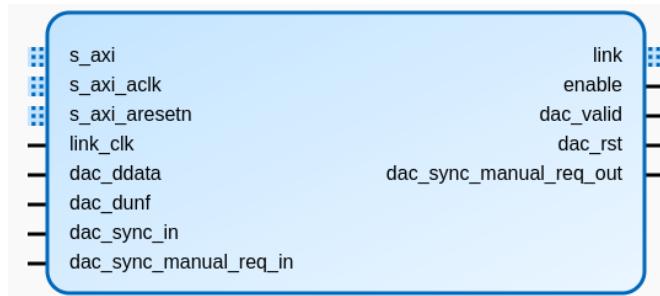


Figura 8 – IP ad_ip_jesd204_tpl_dac.

Fonte: Retirado de [Analog Devices \(2024f\)](#).

2.2.3 Controle Transceptores

O IP axi_adxcvr (Figura 9) é instanciado para possibilitar o controle dos transceptores de alta velocidade da camada física ([Analog Devices, 2024b](#)). Isto porque sua interface AXI permite a modificação dos parâmetros deste bloco.

2.2.4 Acesso Memória

O IP axi_dmac (Figura 10) é um DMA (*Direct Memory Access*) de propósito geral que possui alta velocidade e alto *throughput*. Logo, este é projetado para transferência de dados entre a memória e periféricos. ([Analog Devices, 2024c](#))

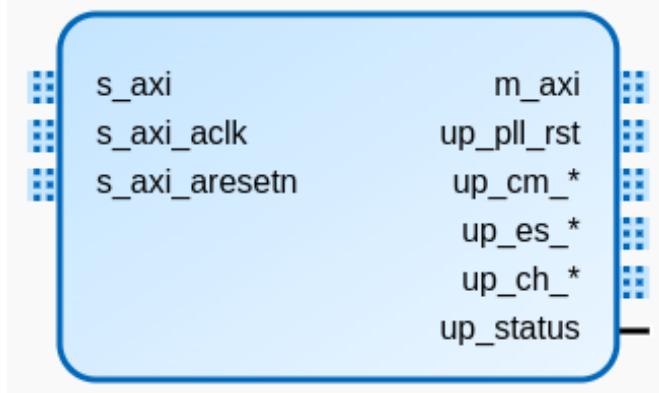


Figura 9 – IP axi_adxcvr.

Fonte: Retirado de [Analog Devices \(2024b\)](#).

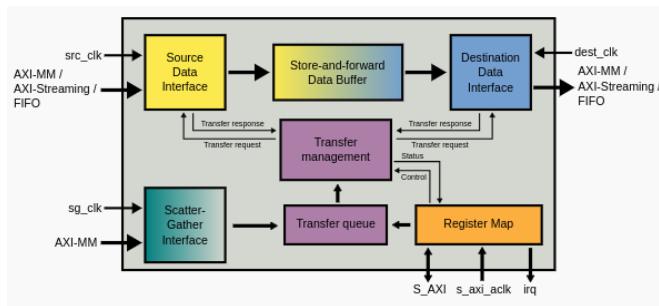


Figura 10 – Diagrama de blocos IP axi_dmac.

Fonte: Retirado de [Analog Devices \(2024c\)](#).

2.2.5 Controle do *Link* do Receptor

O IP axi_jesd204_rx (Figura 11) implementa a camada de *link* do receptor. Conta com a codificação 8B/10B, desenquadramento por *lane* e substituição de caracteres. Além do controle dos sinais SYNC~ e SYSREF, bem como da máquina de estados de *link*. ([Analog Devices, 2024i](#))

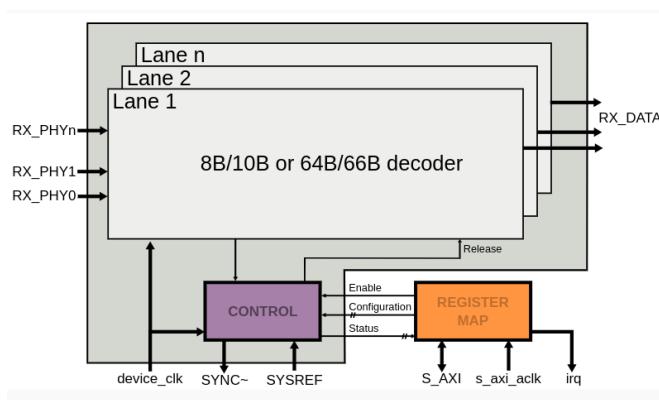


Figura 11 – Diagrama de blocos IP axi_jesd204_rx.

Fonte: Retirado de [Analog Devices \(2024i\)](#).

2.2.6 Controle do *Link* do Transmissor

O IP axi_jes204_tx (Figura 12) implementa a camada de *link* do transmissor. Conta com a codificação 8B/10B, enquadramento por *lane* e substituição de caracteres. Além do controle dos sinais SYNC~ e SYSREF, bem como da máquina de estados de *link*. (Analog Devices, 2024j)

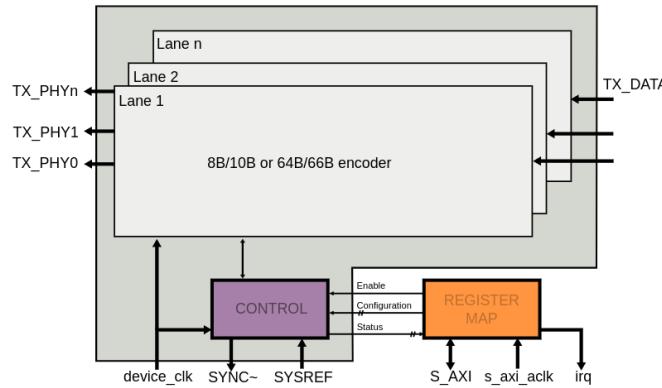


Figura 12 – Diagrama de blocos IP axi_jesd204_tx.

Fonte: Retirado de Analog Devices (2024j).

2.2.7 Troca de Domínios

O IP data_offload (Figura 13) é em essência um *buffer first-in first-out* para compartilhamento de informações entre domínios distintos de *clock*. Mas, possui alguns recursos extras e foi projetado para ser inserido entre o IP DMA e as outras camadas do JESD204. (Analog Devices, 2024g)

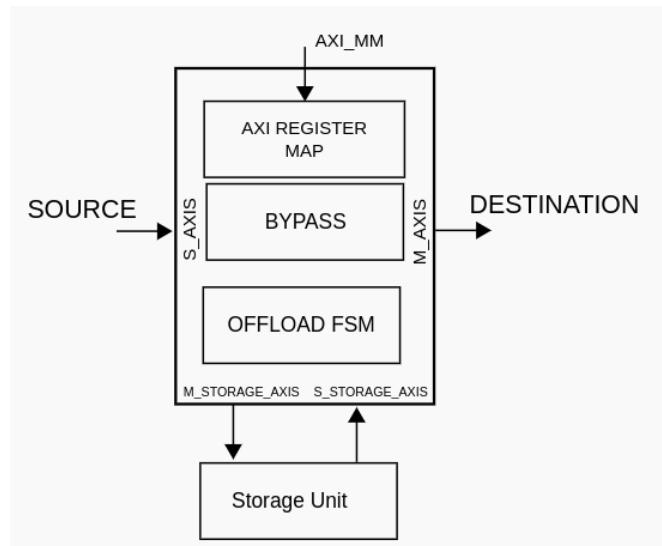


Figura 13 – Diagrama de blocos IP data_offload.

Fonte: Retirado de Analog Devices (2024g).

2.2.8 Link do Receptor

O IP jesd204_rx implementa as funcionalidades da camada de *link* do receptor. Ou seja, o IP axi_jesd204_rx recebe por meio da interface AXI os comandos para a ação a ser realizada e utiliza suas conexões com o jesd204_rx para controlá-lo e realizar a tarefa designada.

2.2.9 Link do Transmissor

O IP jesd204_tx implementa as funcionalidades da camada de *link* do transmissor. Ou seja, o IP axi_jesd204_tx recebe por meio da interface AXI os comandos para a ação a ser realizada e utiliza suas conexões com o jesd204_tx para controlá-lo e realizar a tarefa designada.

2.2.10 Transceptores

O IP util_adxcvr (Figura 14) destina-se a instanciar e configurar os *Gigabit Transceivers High-Speed* (GTHs) presentes na FPGA, além de exibir apenas as portas e atributos necessários (Analog Devices, 2024k). Logo, ele possibilita tratar um conjunto de GTHs como um único objeto, fornecendo para isto as devidas conexões que possibilitam seu controle por meio dos IPs axi_adxcvr.

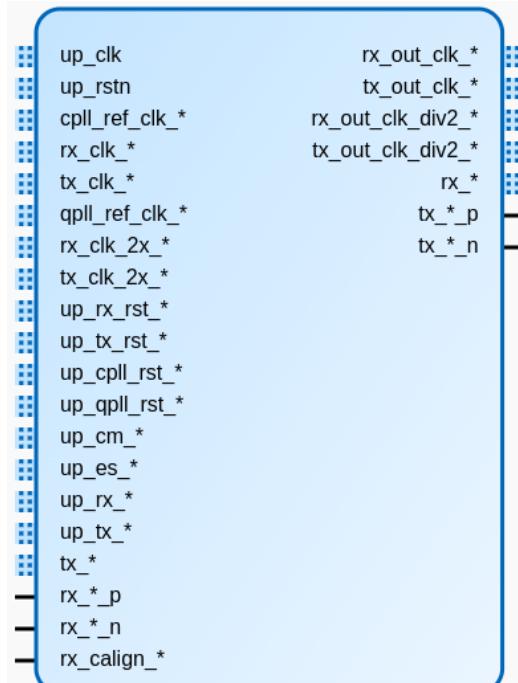


Figura 14 – IP util_adxcvr.

Fonte: Retirado de [Analog Devices \(2024k\)](#).

2.2.11 Empacotador

O IP util_cpack2 (Figura 15) é um empacotador que permite a habilitação por *software* e possibilita o uso total da largura de banda da DMA sem qualquer sobrecarga. Foi desenvolvido para funcionar com um ADC e módulos DMA. ([Analog Devices, 2024d](#))

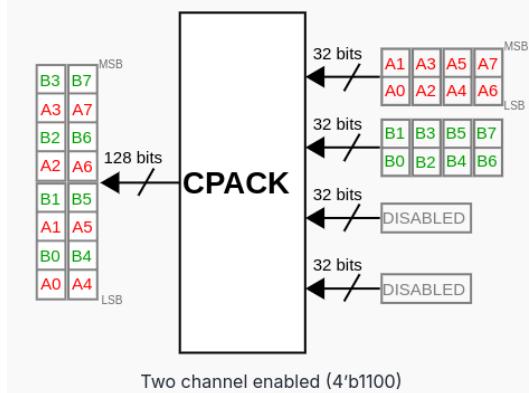


Figura 15 – Diagrama de blocos IP util_cpack2.

Fonte: Retirado de [Analog Devices \(2024d\)](#).

2.2.12 Blocos de Memória da FPGA

O IP util_do_ram é destinado a instanciar as memórias BRAMs (*Block RAM*) da FPGA. Logo, ele é um IP de complementação do data_offload, pois a partir do util_do_ram que o *buffer* é implementado.

2.2.13 Desempacotador

O IP util_upack2 (Figura 16) é um desempacotador que permite a habilitação por *software* e possibilita o uso total da largura de banda da DMA sem qualquer sobrecarga. Foi desenvolvido para funcionar com um DAC e módulos DMA. ([Analog Devices, 2024e](#))

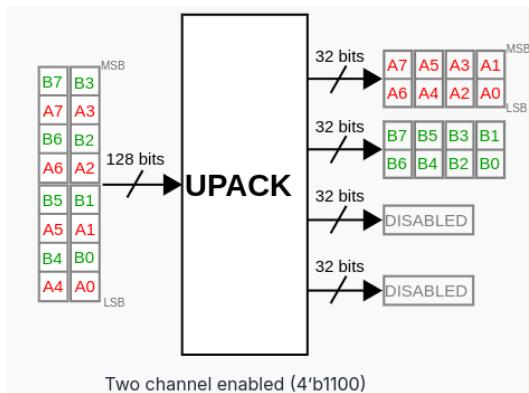


Figura 16 – Diagrama de blocos IP util_upack2.

Fonte: Retirado de [Analog Devices \(2024e\)](#).

2.3 Arquitetura Zynq UltraScale+ MPSoC

A arquitetura Zynq UltraScale+ MPSoC (ZynqMP) é a segunda geração da plataforma Zynq pertencente à empresa AMD (Advanced Micro Devices). Ressalta-se que a arquitetura UltraScale possibilita um desempenho na ordem de grandeza de Gbps de processamento inteligente, ao mesmo tempo que realiza roteamento e processamento de dados no chip. Por isso, os dispositivos que são baseados na arquitetura UltraScale são capazes de atender aos requisitos quando se trata de sistemas de alta largura de banda e alta utilização. ([AMD, 2023a](#))

Assim como a arquitetura Zynq, a ZynqMP combina duas partes (Figura 17) em um mesmo dispositivo, sendo elas: PS (*Processing System*) e PL (*User-Programmable Logic*). Deste modo, a parte PS oferece a escalabilidade de processadores de 64 bits, ao mesmo tempo que combina motores de controle em tempo real, além da possibilidade da integração desse sistema baseado em ARM com a parte programável. Já a PL promove a flexibilidade e escalabilidade de uma FPGA, além de prover o desempenho, a eficiência energética e a facilidade de uso tipicamente associados a ASICs e ASSPs (*Application-Specific Standard Products*).

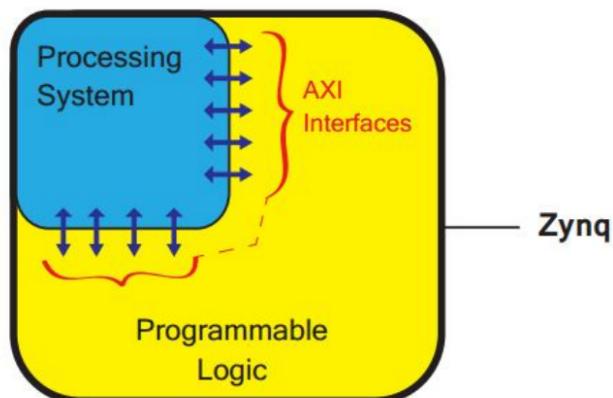


Figura 17 – Regiões PS e PL.

Fonte: Retirado de [CROCKETT \(2014\)](#).

Cabe também ressaltar que os dispositivos ZynqMP oferecem todos os benefícios promovidos pela família Zynq 7000. Mas apresentam novas características como:

- PS escalável com ajuste para potência e desempenho.
- Modo de operação de baixa potência e modo de suspensão.
- Escalonamento de potência e desempenho flexível e programável pelo usuário.
- Sistema avançado de configuração com suporte à segurança do dispositivo e do usuário.

- Suporte de conectividade estendida, incluindo PCIe, SATA e USB 3.0 no PS.
- Interface(s) avançada(s) de usuário com GPU (*Graphics Processing Unit*) e DisplayPort no PS.
- Aumento da largura de banda entre DRAM e PS-PL.
- Tráfego de memória melhorado usando reguladores avançados de QoS (Quality of Service) da ARM.
- Maior segurança e confiabilidade.

Posto isto, para o melhor entendimento do projeto aqui apresentado, as próximas subseções visam detalhar o sistema de processamento, a lógica programável e a comunicação entre essas duas partes.

2.3.1 PS - *Processing System*

O sistema de processamento consegue funcionar no modo *santdalone SoC*, ou seja, ele é capaz de funcionar sozinho sem habilitar a parte PL. Isto ocorre porque essa parte é capaz de suportar um sistema operacional, logo ela realiza o *boot* do sistema e suporta todas as partes presentes na Figura 18. Também cabe ressaltar as unidades de processamento presentes na PS, sendo elas ([AMD, 2023a](#)):

- Cortex-A53 APU é a unidade processamento de aplicação composta pela arquitetura ARM v8 64-bit quad-core ou dual-core multiprocessoing CPU.
- Cortex-R5F RPU é a unidade de processamento de tempo real composta pela arquitetura ARM v7 32-bit dual real-time processing unit with dedicated tightly coupled memory (TCM).
- Mali-400 GPU é a unidade de processamento gráfico com pixel and geometry processor e 64KB de cache L2 (disponível em dispositivos EG e EV MPSoC).
- VCU é a unidade de controle de video capaz de realizar compressão, descompressão e processamento.(disponível nos dispositivos EV MPSoC).

2.3.2 PL - Programmable Logic

A PL é a região de recursos de hardware configuráveis, ela é composta por BRAMs, portas lógicas, estruturas de clock, I/O (Input/Output) padrão e de alta faixa, DSPs (Digital Signal Processors) e LUTs (Look-Up Tables) ([AMD, 2023a](#)). Também, cabe pontuar que os recursos são organizados em layouts de colunas (Figura 19) e são combinados em

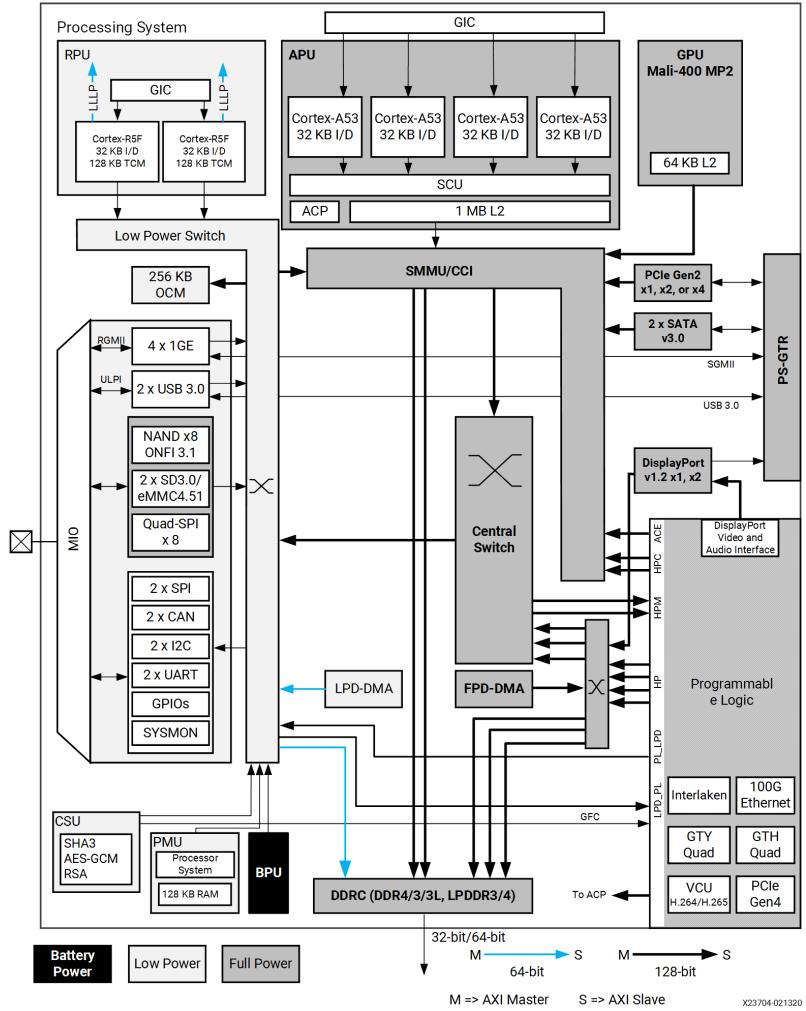


Figura 18 – Arquitetura de Hardware do ZynqMP.

Fonte: Retirado de [AMD \(2023a\)](#).

diferentes proporções para fornecer a capacidade ideal de acordo com a densidade do dispositivo, o mercado-alvo, a aplicação ou o custo do dispositivo ([AMD, 2022b](#)).

Os recursos dentro do dispositivo são divididos em regiões de *clock* segmentadas. A profundidade de uma região de *clock* é de 60 CLBs (Configurable Logic Blocks), um banco de 52 I/Os, 24 slices de DSP, 12 BRAMs ou 4 canais de transceptores. Ressalta-se que todas essas regiões possuem a mesma área, além de contarem com um roteamento de *clock* tanto vertical quanto horizontal ([AMD, 2022b](#)).

Outro ponto importante da região PL são seus controles de periféricos e unidades funcionais disponíveis. Sendo estes ([AMD, 2023a](#)):

- PCI Express Integrated
- 100G Ethernet
- DisplayPort Video and Audio Interfaces

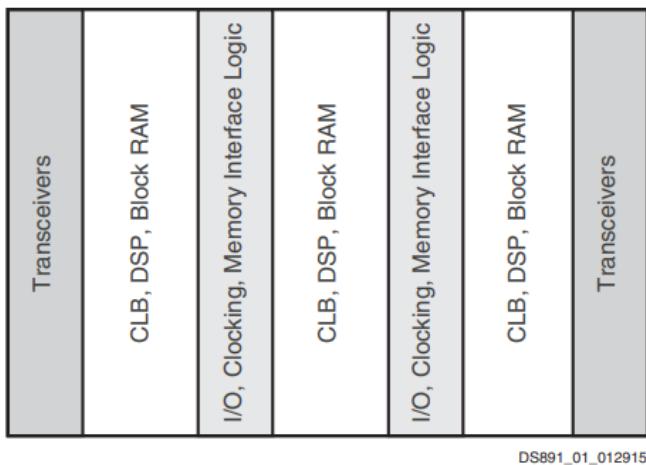


Figura 19 – Recursos em coluna da PL.

Fonte: Retirado de [AMD \(2022b\)](#).

- Interlaken
- GTH and GTY Transceivers
- PL System Monitor
- Video Codec Unit

Por fim, dentre as características supracitadas, no projeto aqui apresentado, os transceptores GTH são de suma importância. Então, destaca-se que esses transceptores são transmissores e receptores seriais e são organizados em grupos de quatro, conhecidos como um Quad de transceptor. Ademais, os transceptores GTH são eficientes em termos de energia, suportando taxas de linha de 500 Mbps a 16,375 Gbps. E, são configurados através da parte PS ([AMD, 2023a](#)).

2.3.3 Comunicação PS e PL

Como supracitado, a parte PS tem a possibilidade de funcionar sem a habilitação da PL, contudo, a arquitetura foi desenvolvida para o funcionamento de ambas partes em conjunto. Logo, a PS e a PL podem estabelecer muitas ou poucas conexões entre si. Essas conexões permitem que os designs desenvolvidos na PL sejam configurados através da PS, além de possibilitar que a PL seja capaz de acessar a memória da PS. ([AMD, 2023a](#))

A região PS se comunica com a PL usando blocos de interconexão de propósito geral que suportam uma variedade de interfaces. As interfaces possibilitam a transferência de dados entre as duas partes, bem como de outros sinais como as interrupções, *clocks* e *resets*. Além disso, elas também conectam periféricos do PS ao PL, permitindo roteamento dos I/Os do PL. Estas conexões e interfaces podem ser observadas na Figura 20.

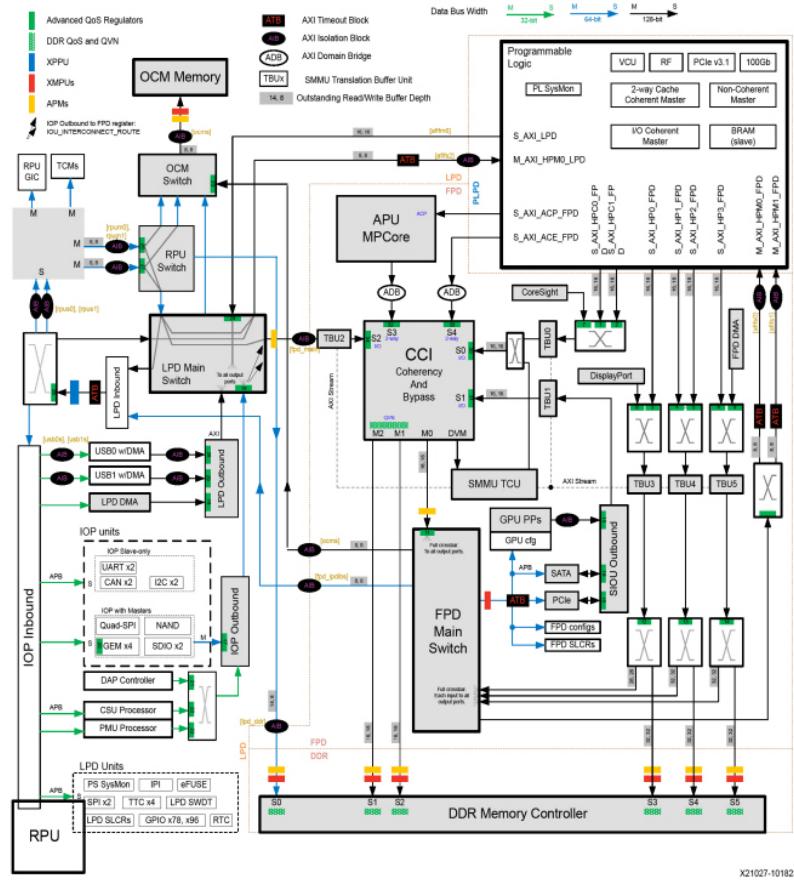


Figura 20 – Interfaces AXI PS-PL.

Fonte: Retirado de [AMD \(2023a\)](#).

E, quando se trata das interfaces AXI (*Advanced eXtensible Interface*) entre PS e PL, as principais características oferecidas são:

- Interface AXI4 de alto desempenho com suporte a *First-In First-Out* no PS.
 - Suporte a largura de barramento de dados nativo selecionável no PL (32/64/128 bits).
 - *Clocks* de leitura e escrita independentes.
 - Três interfaces suportam a coerência de I/O por meio da interconexão de cache-coerente.
- Unidade de Gerenciamento de Memória do Sistema para transações vinculadas ao PS (tradução de endereço virtual para físico).
- Caminho dedicado de baixa latência entre o domínio de baixa potência e o PL.
- Interface de porta de coerência de acelerador para coerência de I/O e alocação no cache L2 da Unidade de Processamento Acelerado.

- Extensões de coerência AXI para coerência total. Utilizável como ACE-Lite para coerência de I/O.

Ademais, existem outras interfaces que possibilitam diferentes funcionalidades, como: comunicação com a PMU (*Platform Management Unit*), interação com as interrupções do processador, comunicação dedicada do GEM (*gigabit Ethernet controller*) para o protocolo DisplayPort, etc.

2.4 Vivado

O Vivado Design Suite foi projetado pela AMD para melhorar a produtividade, isto porque atualmente os dispositivos Xilinx contam com uma variedade de recursos. Logo, estes necessitam de uma ferramenta adequada para lidar com os desafios de *design* oriundos da diversidade de tecnologias. Desta forma, o Vivado prove ferramentas de *place and route* que otimizam métricas concorrentes de design como: temporização, congestionamento, comprimento total de fios, utilização e potência. Também, pontua-se que cada etapa do processo de design pode ser analisada através do Vivado. ([AMD, 2021a](#))

Também, o Vivado possui uma linguagem de comando nativa conhecida como TCL (Tool Command Language), esta linguagem que descreve todas as ferramentas disponíveis. Ou seja, um script TCL pode conter comandos que executam o fluxo completo de design, síntese e implementação de um projeto. Mas, outra opção disponível para utilizar as ferramentas, trata-se da interface gráfica, a qual é denominada Vivado IDE (*Integrated Design Environment*). Então, por meio do Vivado IDE é possível visualizar e interagir com o design em cada etapa do projeto, como exemplo a visualização do design em vários modos. Dentre essas opções, tem-se a visualização após as seguintes etapas: elaboração do RTL (*Register-Transfer Level*), síntese e implementação. ([AMD, 2021a](#))

Posto isto, o Vivado IDE oferece uma gama de algoritmos para melhorar a performance do design, dentre eles pode-se citar ([AMD, 2021b](#)):

- *Design* em nível RTL em VHDL, Verilog e SystemVerilog;
- Integração de propriedade intelectual (IP) para núcleos;
- Simulação comportamental, funcional e de temporização com o simulador Vivado;
- Síntese Vivado;
- Implementação Vivado para *place and route*;
- Analisador de I/O serial e lógica do Vivado para depuração;
- Análise de potência do Vivado;

- Restrições de design Xilinx baseadas em SDC (XDC) para inserção de restrições de temporização;
- Análise estática de temporização;
- *Floorplanning* de alto nível;
- Modificação detalhada do *place and route*
- Geração de *bitstream*;

Como supracitado, o Vivado oferece ferramentas para integração de IPs, isto porque segundo a [AMD \(2024c\)](#) as FPGAs têm-se tornado cada vez maiores e mais complexas, tornando essencial o uso de IPs de terceiros para compor sistemas complexos. Deste modo, no Vivado é possível criar *block designs*, nos quais pode-se adicionar IPs e realizar a sua interconexão. Portanto, nas subseções seguintes serão explicitados alguns IPs estudados que foram necessários ao projeto aqui apresentado e, também, são nativos do Vivado, ou seja, produzidos pela Xilinx.

2.4.1 AXI Direct Memory Access

O IP AXI Direct Memory Access (Figura 21) também conhecido como AXI DMA provê um acesso de alta largura de banda entre a memória e um periférico que possui uma interface *AXI4-Stream*. Opcionalmente, pode-se habilitar a função *Scatter/Gather* para que o IP realize a transferência de dados sem depender do processador. Além disso, ele oferece uma interface escravo *AXI4-Lite* que possibilita configurá-lo através da memória para realizar a transferência, bem como verificar seu status. ([AMD, 2024a](#))

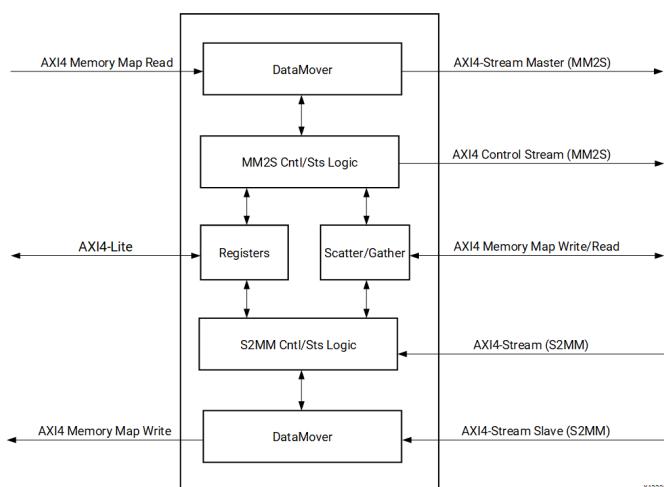


Figura 21 – Diagrama de Blocos AXI DMA.

Fonte: Retirado de [AMD \(2024a\)](#).

2.4.2 AXI GPIO

O IP AXI GPIO (Figura 22) possibilita que a interface GPIO (*General Purpose Input/Output*) seja mapeada para uma interface *AXI4-Lite*, ou seja, configurável pela memória. Também, este IP pode operar no modo de canal único ou duplo, sendo que a largura de bits de cada canal pode ser configurada independentemente. Ademais, caso necessário, pode-se habilitar os canais para gerarem interrupções quando detectada uma transição em suas respectivas entradas. (AMD, 2015a)

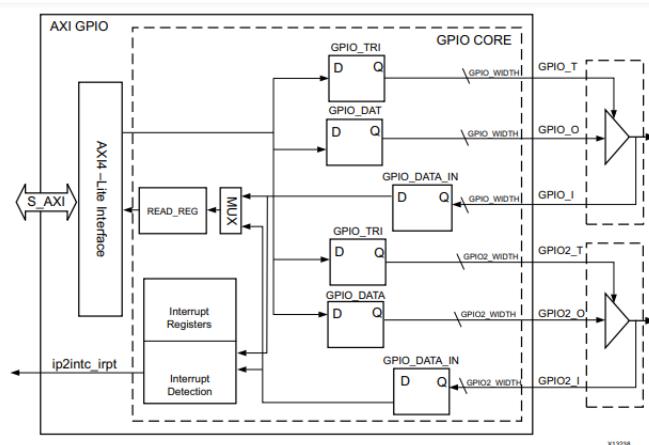


Figura 22 – Diagrama de Blocos AXI GPIO.

Fonte: Retirado de AMD (2015a).

2.4.3 AXI Interconnect

O IP AXI Interconnect (Figura 23) é utilizado para realizar uma mistura de AXIs mestres e escravos, isto porque os núcleos de sua infraestrutura são automaticamente inferidos e conectados. Desta forma, este bloco possibilita conectar interfaces com diferentes protocolos, largura de dados e domínio de clock. Um ponto importante a se ressaltar sobre este IP é que ele só pode ser utilizado através da sua inserção em um block design do Vivado. (AMD, 2022a)

2.4.4 Concat

O IP Concat (Figura 24) possibilita a concatenação de sinais, ou seja, ele permite a entrada de diversos barramentos e libera apenas um barramento de saída que consiste na união de todas as entradas. Vale ressaltar que o número de entradas, bem como a largura de bits de cada uma, são parâmetros configuráveis. Por fim, este IP só pode ser utilizado ao adicioná-lo no block design e sua inicialização não implicará em nenhum consumo de recursos do FPGA. (AMD, 2016a)

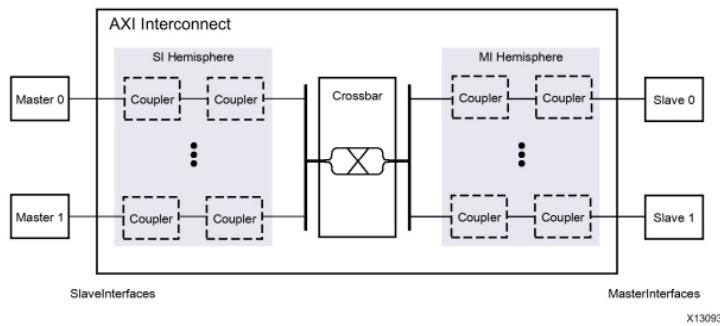


Figura 23 – Diagrama de Blocos AXI Interconnect.

Fonte: Retirado de [AMD \(2022a\)](#).

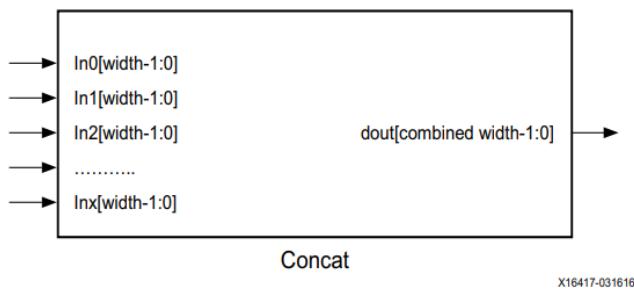


Figura 24 – Diagrama de Blocos Concat.

Fonte: Retirado de [AMD \(2016a\)](#).

2.4.5 Constant

O IP Constant (Figura 25) é utilizado para liberar um valor constante em sua saída. Este IP só pode ser utilizado através do block design, no qual também é possível configurar seus parâmetros: largura de bits do sinal de saída e o seu valor. Vale ressaltar que o valor pode ser inserido em diversos formatos, dentre eles: decimal, binário (b), octal (0) e hexadecimal (0x ou 0X). ([AMD, 2018](#))

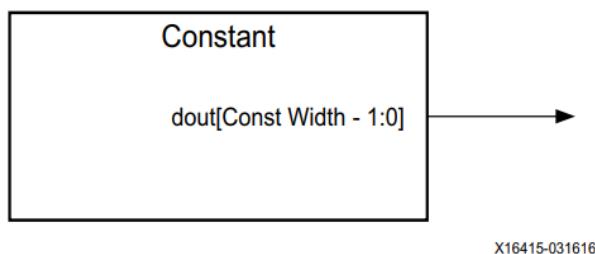


Figura 25 – Diagrama de Blocos Constant.

Fonte: Retirado de [AMD \(2018\)](#).

2.4.6 Processor System Reset

O IP Processor System Reset (Figura 26) fornece o sinal de reset para todo o design, ou seja, para todos os outros IPs presentes, até mesmo o processador. Sendo assim, esse IP considera condições internas e externas para gerar os sinais de reset como saída. Por fim, vale ressaltar que ele recebe dois parâmetros, sendo estes: largura da janela ativa de reset externo e largura da janela ativa de reset auxiliar. Tais parâmetros auxiliam na definição da largura mínima do reset externo e na detecção de transição do clock. ([AMD, 2015b](#))

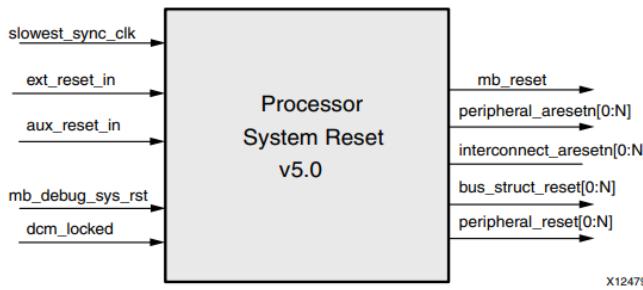


Figura 26 – Diagrama de Blocos Processor System Reset.

Fonte: Retirado de [AMD \(2015b\)](#).

2.4.7 Slice

O IP Slice (Figura 27) possibilita a seleção de um intervalo de bits dado um sinal de entrada. Logo, tem-se a largura do sinal de entrada como um parâmetro, assim como o bit mais significativo e o bit menos significativo do intervalo que se deseja retirar os bits. Estes parâmetros podem ser configurados por meio do *block design*, que é o único lugar possível para instanciar esse IP. Por fim, ressalta-se que a sua utilização no *block design* não implicará em nenhum consumo de recursos do FPGA. ([AMD, 2016b](#))

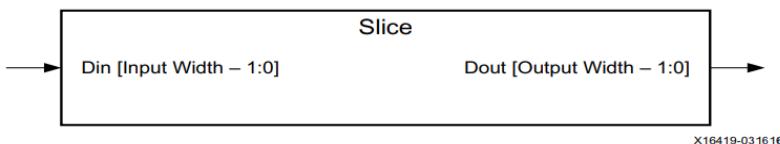


Figura 27 – Diagrama de Blocos Slice.

Fonte: Retirado de [AMD \(2016b\)](#).

2.4.8 Zynq UltraScale+ MPSoC

O IP Zynq UltraScale+ MPSoC (Figura 18) é a interface de software por volta do sistema de processamento do ZynqMP. Ou seja, ele funciona como um *wrapper* (Figura 28) para a parte PS, o que possibilita instanciar e deixar essa parte visível para a PL e para

outras lógicas externas. Deste modo, as interfaces disponibilizadas por este IP podem ser organizadas em três grupos principais: *extended multiplexed I/O* (EMIO), *programmable logic I/O* e AXI I/O. Por fim, o PCW (*PS Configuration Wizard*) que possibilita configurar todas as funcionalidades deste IP, das quais pode-se citar (AMD, 2023b):

- Habilitar/Desabilitar Periféricos de I/O
- Habilitar/Desabilitar Interfaces AXI
- Configuração de I/O Multiplexada (MIO)
- I/Os Multiplexadas Estendidas (EMIO)
- *Clocks*, interrupções e resets da PL
- *Clock* interno do PS
- Geração de Registros de Configuração de Nível de Sistema
- Configuração de SERDES de Alta Velocidade
- Configuração da DDR do PS
- Configuração de Isolamento

2.5 Vitis

O Vitis é uma plataforma unificada de software que oferece uma significativa quantidade de ferramentas no que se refere ao desenvolvimento de software embarcado como backend. Além disso, por meio da sua interface gráfica, o Vitis IDE, é possível desenvolver diversas aplicações de software para processadores embarcados da AMD, geralmente utilizando os designs de hardware oriundos do Vivado. Cabe ressaltar que o Vitis IDE é baseado na tecnologia da Eclipse Foundation, especificamente no software Eclipse Theia. Por fim, dentre os recursos disponibilizados pelo Vitis IDE têm-se (AMD, 2024b):

- Criar plataformas a partir de projetos de hardware gerados pelo AMD Vivado e gerar BSP (Board Support Package) para desenvolvimento de software.
- Criar aplicações a partir de designs de exemplo ou modelo vazio.
- Configurar e construir as plataformas e as aplicações.
- Executar, depurar ou criar perfis de aplicações no hardware.

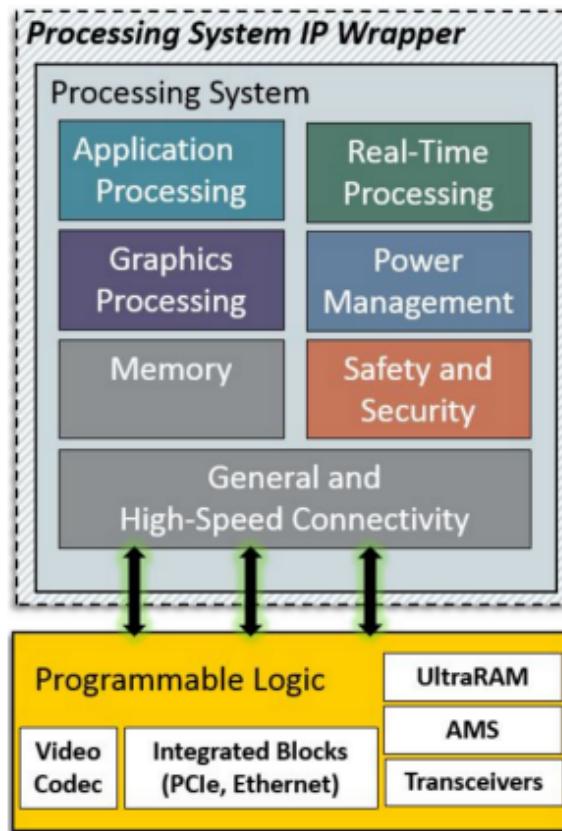


Figura 28 – ZynqMP IP Arquitetura Wrapper.

Fonte: Retirado de [AMD \(2023b\)](#).

- Gerenciar várias conexões de hardware locais ou remotas com o *Target Connection Manager*.
- Suporte avançado para dispositivos e processadores, desde MicroBlaze , Zynq 7000, Zynq AMD UltraScale+ MPSoC até AMD Versal.
- Criar imagens de inicialização.
- Configurar dispositivos.
- Programação Flash.
- Gerenciar projetos por IDE e executar ações de acordo com o tipo de componente.
- Gerenciar projetos por scripts de usuário e usar a IDE para auxiliar no procedimento de depuração.
- Controle de versão do código-fonte com git integrado.

Por fim, ressalta-se que o fluxo de criação de um projeto na plataforma Vitis (Figura 29) consiste na exportação do hardware do Vivado através de um arquivo XSA (*Xilinx*

Shell Archive), posteriormente na inclusão desse arquivo na plataforma e na definição de suas configurações (domínio). Após essas etapas, inicia-se o desenvolvimento das aplicações e, em seguida, a depuração dessas. Caso necessário, pode-se realizar a verificação a nível de sistema, quando se trata de sistemas complexos. E, por fim, com tudo validado, o Vitis é capaz de auxiliar na criação de imagens de BOOT para a inicialização do sistema.

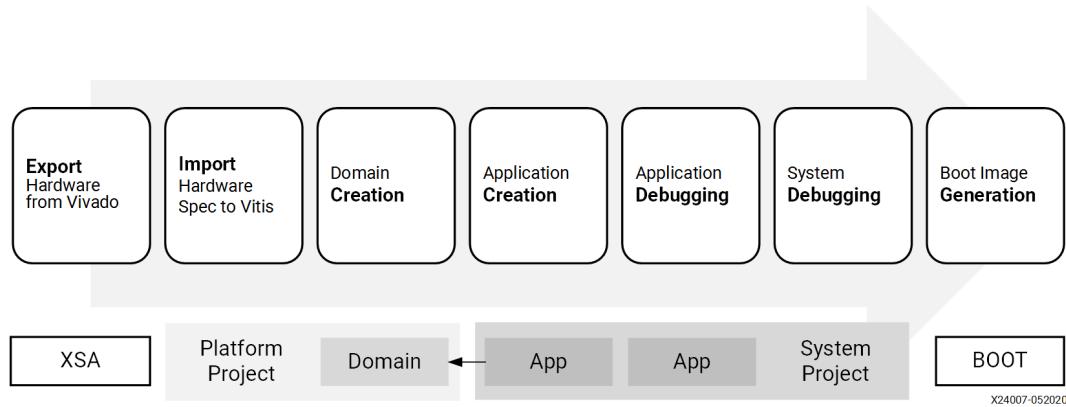


Figura 29 – Fluxo de desenvolvimento Vitis.

Fonte: Retirado de [AMD \(2024b\)](#).

3 Metodologia

Para o desenvolvimento do trabalho aqui exposto, optou-se pela divisão em três partes. A primeira parte aborda o desenvolvimento do *hardware* em FPGA. A segunda consiste no desenvolvimento do *software* para controle do sistema. E, por fim, a terceira etapa é composta pelo teste em bancada do *hardware* e *software* desenvolvidos. Sendo assim, todas as seções seguintes buscam detalhar as três partes explicitadas.

3.1 Desenvolvimento *Hardware*

Como já supracitado, o protocolo JESD204B é composto por camadas. Desta forma, esta etapa consistiu no estudo dos IPs a fim de construir, por meio do Vivado, a camada física, de link, de transporte e de aplicação. Também, é nesta etapa que foi definido o sistema topo e as *constraints* para gerar o *bitstream*. Sendo assim, pode-se dividir esta etapa nos seguintes passos.

3.1.1 Estudo dos IPs

Neste passo foi necessário realizar o estudo dos IPs da Xilinx e da Analog Devices, os quais são necessários para construir o *hardware* desejado. Logo, foi realizada a leitura da documentação fornecida pelas duas empresas citadas, a fim de compreender os parâmetros necessários para cada IP e as ligações necessárias entre cada camada do protocolo.

3.1.2 Construção dos Blocos

Para o desenvolvimento do *hardware* foi utilizado o *block design* do Vivado. Vale explicitar que os IPs da Xilinx já estavam disponíveis para serem inseridos, o que não ocorria com os IPs da Analog Devices. Portanto, nesse passo foi utilizado os arquivos Verilog fornecidos para a construção dos blocos IPs.

3.1.3 Construção das Camadas

Após realizar o desenvolvimento dos blocos IPs do passo anterior, estes foram inseridos no *block design* a fim de formar cada camada do protocolo. Ressalta-se, também, que foi criado um bloco IP para cada camada desenvolvida.

3.1.4 União das camadas

Com todas as camadas representadas por um IP foi possível criar um *block design* para conter o protocolo completo. Além disso, criou-se um arquivo Verilog que representa o sistema topo, ou seja, utiliza o bloco completo para tratar e externalizar as saídas desejadas. E, por fim, adicionou-se os arquivos XDC com as *constraints* de temporização e pinagem.

3.1.5 Exportação o *Hardware*

Com todos os passos anteriores finalizados, foi possível executar a opção “gerar *bitstream*”, a qual está presente no Vivado. E, finalmente, utilizar essa mesma plataforma para exportar o *hardware* gerado para um arquivo XSA.

3.2 Desenvolvimento do *Software*

Esta etapa consistiu na criação e depuração do *software* através da plataforma Vitis. Para isto, foi criada uma plataforma de *hardware* e códigos em linguagem C que definem uma aplicação. Desta maneira, serão apresentados nos tópicos seguintes os passos realizados.

3.2.1 Criação da Plataforma

Após ter exportado o arquivo XSA do Vivado, este foi utilizado para criar uma plataforma de *hardware* no Vitis. Esta plataforma é a maneira do código em C ter acesso aos recursos desenvolvidos na FPGA. Vale ressaltar que na plataforma, definiu-se a aplicação para funcionar em primeira instância no ambiente em modo bare metal.

3.2.2 Estudos dos Códigos

Como citado no capítulo anterior, o *framework* da Analog Devices também fornece códigos fontes. Logo, neste passo, foi realizado o estudo destes arquivos, a fim de selecionar só aqueles necessários a cada camada desenvolvida.

3.2.3 Reestruturação dos Códigos

O passo anterior evidenciou a seleção dos códigos referentes apenas a cada camada. Contudo, foram adicionados blocos personalizados no sistema, logo este passo concentrou-se em criar novos arquivos ou funções quando necessário.

3.2.4 Depuração

Como citado no primeiro passo, em primeira instância todo código desenvolvido direciona-se a uma aplicação bare metal. Deste modo, foi utilizado o modo de depuração disponibilizado pelo Vitis, a fim de validar o software para prosseguir com os passos de adaptação para o Petalinux.

3.2.5 Adaptação para Sistema Operacional

Este passo consistiu primeiramente na compilação do Petalinux personalizado pelo arquivo XSA. Em seguida, foi necessário realizar a modificação do código existente para um código capaz de executar em um sistema operacional.

3.3 Teste em Bancada

As seções anteriores apresentaram a construção do sistema, logo, para validar o seu funcionamento foi necessária a montagem de um ambiente de teste. Desta forma, os equipamentos utilizados para os testes foram:

- 1 gerador de ondas: possibilitando o envio de diferentes sinais de testes ao sistema;
- 1 placa de conversores com o JESD204B integrado: possibilitando a transição entre o domínio analógico e digital;
- 1 placa com arquitetura ZynqMP: possibilitando a implementação do *hardware* descrito e a execução do *software* de controle;
- 1 osciloscópio: possibilitando a observação dos sinais gerados pelo sistema.

4 Desenvolvimento Prático

O desenvolvimento prático visa alcançar o projeto presente no diagrama de blocos apresentado pela Figura 1. Desta forma, fica evidente o objetivo final do projeto, ou seja, que qualquer usuário seja capaz de controlar o sistema implementado na FPGA por meio do Petalinux.

Então, para que o sistema opere como desejado, a parte PS da arquitetura ZynqMP necessita dos IPs que realizam o controle do sistema e possibilitam a comunicação com a parte PL. Além disso, a parte PL conta com os IPs referentes às camadas do protocolo JESD204B, os quais permitem a comunicação em Gbps com os chips conversores.

Vale explicitar que, seguindo as especificações do protocolo, a transmissão de dados entre a placa ZynqMP e a placa com os conversores é feita por canais diferenciais. Também, outro ponto importante a ser ressaltado é a presença de fluxos de dados distintos entre o ADC e o DAC, haja vista que existem camadas e partes dedicadas a cada um destes recursos.

Ademais, no que se refere à interação com o usuário, é desejável uma maneira de fácil comunicação com o sistema. Logo, tal qualidade é buscada através da implementação de um SO juntamente com uma aplicação de configuração e controle do sistema.

Sendo assim, para atingir o projeto explicitado, este capítulo busca explicitar os procedimentos realizados para implementar as etapas apresentadas na metodologia. Logo, serão detalhados a construção do sistema em FPGA, o desenvolvimento da aplicação de controle e o ambiente de testes.

4.1 Sistema em FPGA

Esta seção apresentará o desenvolvimento do sistema que será implementado em FPGA. Logo, serão detalhados nas subseções seguintes os passos para a construção do *hardware*.

4.1.1 Camadas do JESD204B

As camadas do protocolo JESD204B são uma parte vital do sistema. Sendo assim, nos próximos tópicos serão detalhadas as suas funcionalidades e quais IPs foram utilizados como base para que cada camada se tornasse um novo IP. Ressalta-se que a organização em IPs foi realizada a partir da ferramenta *IP Packager* do Vivado.

4.1.1.1 Camada Física

O objetivo da camada física consiste em receber os dados seriais do ADC e enviar um pacote de 128 bits para a camada de *link* do receptor, além de receber um pacote de 128 da camada de *link* do transmissor e enviar dados seriais ao DAC. Desta maneira, torna-se evidente que essa camada é compartilhada entre os caminhos de aquisição e transmissão. Posto isto, a Figura 30 demonstra os IPs que foram utilizados, sendo estes:

- Concat: utilizados para unir as informações das 4 vias de dados do receptor, as quais serão enviadas para camada de *link* do receptor.
- Slice: utilizados para separar as informações oriundas da camada de *link* do transmissor para cada via serial conectada ao DAC.
- util_adxcvr: utilizado para instanciar os transceptores que realizam a comunicação serial com os chips. Ressalta-se que o parâmetro XCVR TYPE deste IP foi definido para 8 visando identificar que a placa possui GTHs do tipo GTHE4.
- axi_adxcvr: utilizados para realizar o controle do util_adxcvr, sendo um destinado ao ADC e outro destinado ao DAC.

Além disso, é importante ressaltar que o IP dos transceptores, quando gerado pela síntese, sofre realocações nas suas ligações internas visando atender o caminho de dados do receptor. Logo, os pinos físicos relacionados ao DAC não são colocados na mesma ordem, estes são alocados da seguinte forma: 0, 2, 3 e 1. Então, as ligações entre os IPs relacionadas ao DAC precisaram ser rearranjadas para que os dados sejam transmitidos corretamente pelos 4 canais diferenciais. Desta maneira, considerando a camada de *link* do transmissor como fonte e a física como recebimento, o rearranjo das ligações é dado por: (0 → 0), (1 → 2), (2 → 3) e (3 → 1).

Destaca-se que, além das ligações com as camadas de *link* já supracitadas, também estão presentes as saídas diferenciais (tx_n e tx_p) destinadas ao chip DAC, as entradas diferenciais (rx_n e rx_p) destinadas ao chip ADC e as interfaces AXIs para o controle dos IPs. Por fim, o IP criado para esta camada foi denominado phy_shared.

4.1.1.2 Camada de *Link*

A camada de *link* não é compartilhada como a camada física. Logo, primeiro será apresentada a camada do receptor e em seguida a do transmissor. Mas, antes, cabe evidenciar que ambas camadas compartilham a mesma configuração do parâmetro *Link Mode*. Tal parâmetro é definido como 1 para identificar a codificação 8B/10B que, consequentemente, determina que está sendo utilizada a versão B do protocolo JESD204. Estas camadas também são configuráveis através de suas interfaces AXI.

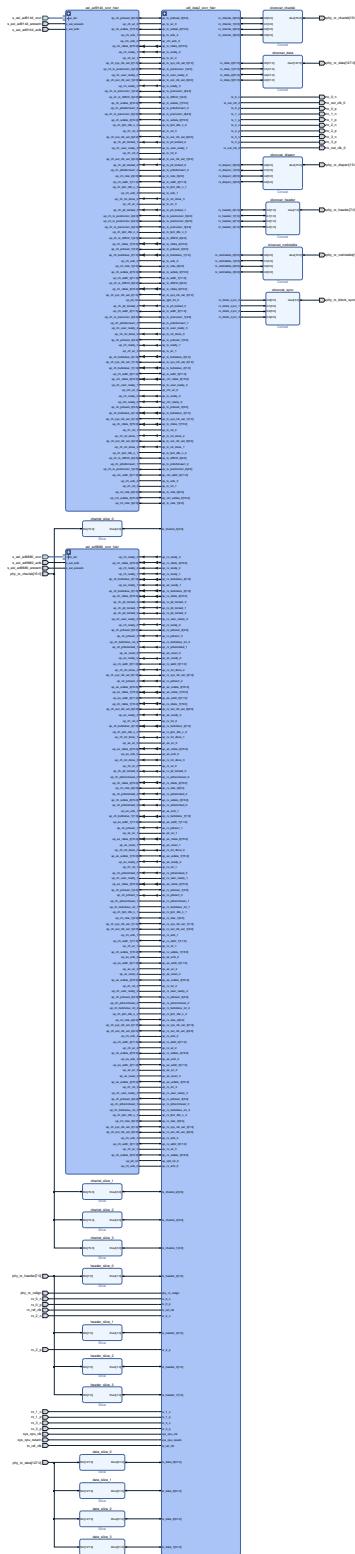


Figura 30 – Diagrama de blocos da camada Física.

Fonte: Própria.

Deste modo, a camada de *link* do receptor (Figura 31) recebe os pacotes de bits da camada física, os quais contam com os dados das amostras, além de outros sinais que permitem a realização de verificações de erros. Após as análises, os dados são repassados para a camada de transporte. Para sua construção, foram utilizados os IPs jesd204_rx e axi_jesd204_rx. O IP criado para esta camada foi denominado axi_ad9680_jesd.



Figura 31 – Diagrama de blocos da camada de *link* do ADC.

Fonte: Própria.

Por fim, a camada de *link* do transmissor (Figura 32) recebe da camada de transporte os dados das amostras. Então, após o recebimento, esta envia os dados para a camada física, além de informações adicionais que permitem verificar erros de comunicação. Para sua construção, foram utilizados os IPs jesd204_tx e axi_jesd204_tx. O IP criado para esta camada foi denominado axi_ad9144_jesd.

4.1.1.3 Camada de Transporte

A camada de transporte também não é compartilhada, desta maneira, primeiro serão apresentados os detalhes da camada referente ao receptor e, em seguida, do transmissor. Mas, antes, ressalta-se que estas camadas são configuráveis através de suas interfaces AXI.

Então, a camada de transporte do receptor (Figura 33) é responsável por receber os dados da camada de *link* realizar o desenquadramento dos dados e disponibilizá-los à camada de aplicação. Vale ressaltar que o desenquadramento consiste na separação das amostras dependendo da sua precisão e número de canais, os quais são parâmetros de configuração.

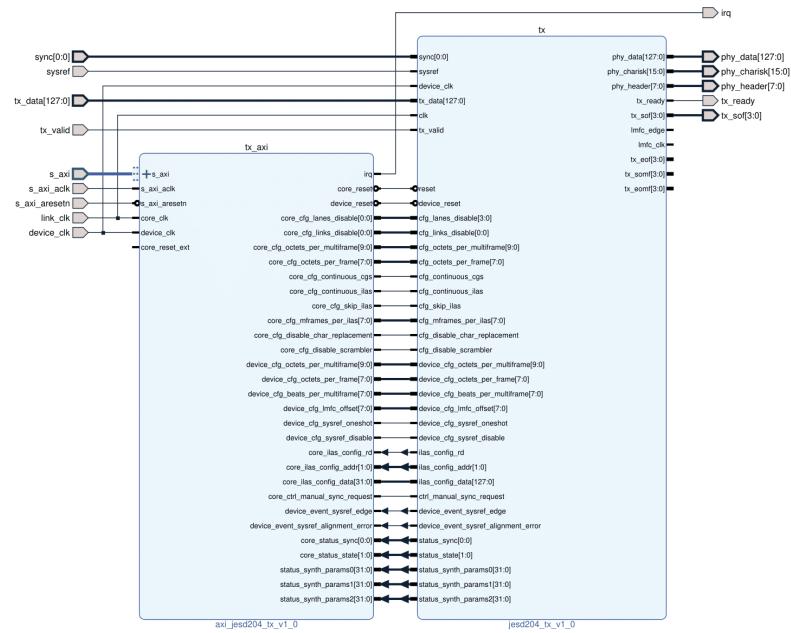


Figura 32 – Diagrama de blocos da camada de *link* do DAC.

Fonte: Própria.

guração desta camada. Para sua construção, foram utilizados os IPs ad_ip_jesd204_tpl_adc e Slices. Já o IP criado para esta camada foi denominado axi_ad9680_tpl.

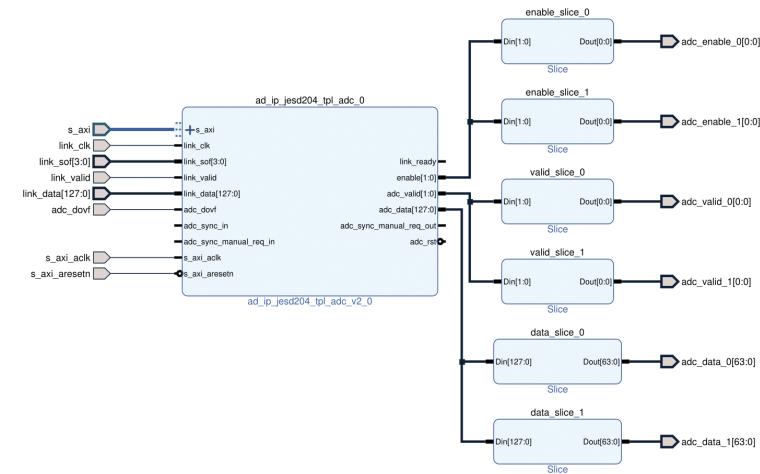


Figura 33 – Diagrama de blocos da camada de transporte do ADC.

Fonte: Própria.

Por fim, a camada de transporte do transmissor (Figura 34) consiste em receber os dados da camada de aplicação, realizar o enquadramento e repassá-los à camada de *link*. Também, ressalta-se que o enquadramento depende da precisão das amostras e do número de canais para decidir a maneira de realizar a união dos dados. Para sua construção, foram utilizados os IPs ad_ip, jesd204_tpl_dac, Concat e Slices. Já o IP criado para esta camada

foi denominado axi_ad9144_tpl.

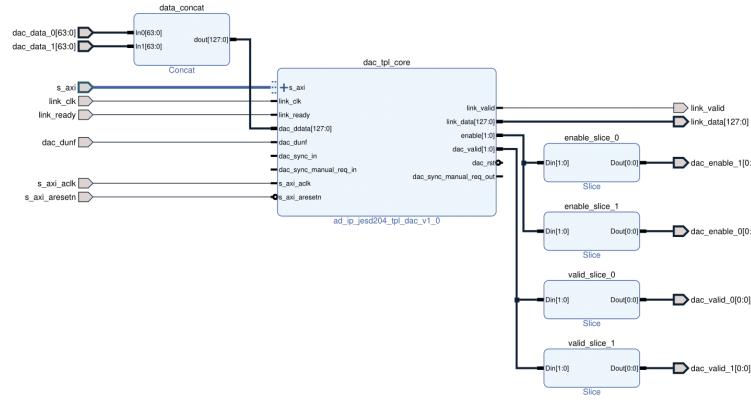


Figura 34 – Diagrama de blocos da camada de transporte do DAC.

Fonte: Própria.

4.1.2 Camadas de Aplicação

A camada de aplicação, diferente das outras camadas do JESD204B, pode ser projetada de acordo com o objetivo que se deseja alcançar com o sistema em questão. Deste modo, visando atingir o sistema desejado, as camadas já apresentadas foram unidas às camadas de aplicação, como pode ser observado pela Figura 35. Portanto, serão apresentadas a seguir a camada de aplicação criada para o ADC e para o DAC.

4.1.2.1 Camada Aplicação Receptor

Para lidar com os dados recebidos com a aquisição de um sinal, o caminho projetado na camada de aplicação do receptor baseou-se no envio dos dados para um empacotador, seguido de um *buffer* e, por último, enviá-los para a memória.

Deste modo o empacotador utilizado foi o IP util_cpack2. O *buffer* foi criado a partir da união dos IPs data_offload e util_do_ram, vale ressaltar que utilizou-se deste artifício para que o DMA conseguisse operar em um menor *clock*. E, por fim, o DMA utilizado foi o IP axi_dmac.

Sendo assim, a Figura 36 apresenta a camada de aplicação explicitada, juntamente com a camada de *link* e a de transporte do receptor.

4.1.2.2 Camada Aplicação Transmissor

A camada de aplicação do transmissor consiste em um DMA que envia os dados para um desempacotador. Em seguida, o desempacotador já envia os dados para as outras camadas do JESD204B. Neste caso não foi necessário um *buffer* pois o DMA opera no

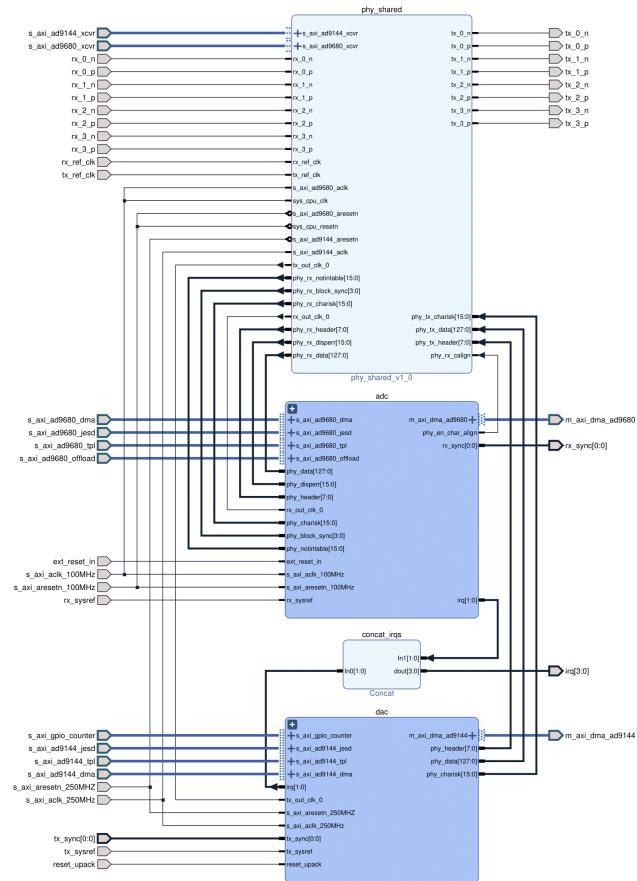


Figura 35 – Diagrama de blocos do JESD204B.

Fonte: Própria

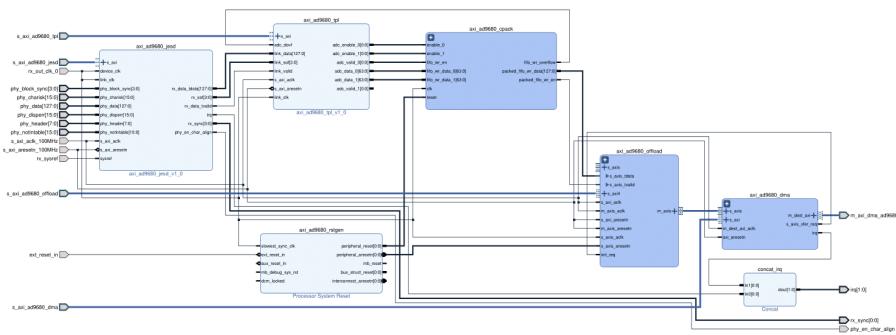


Figura 36 – Diagrama de blocos do ADC.

Fonte: Própria.

mesmo domínio de *clock* que o restante dos IPs. Desta maneira, ressalta-se que o DMA utilizado foi o AXI Direct Memory Access e o desempacotador o util_upack2.

Contudo, esta camada de aplicação possui alguns blocos adicionais. Sendo estes um AXI GPIO e um bloco personalizado counter_ceil. Então, explicita-se que o GPIO foi adicionado para enviar e receber dados do contador. Já o contador trata-se de uma Máquina de Moore desenvolvida a fim de obter uma repetição programável do sinal. Logo,

a máquina de estados pode ser observada na Figura 37 e a lógica de seus estados é dada por:

- IDLE: é o estado inicial ao habilitar o *reset* na máquina de estados; permanece nesse estado enquanto a entrada start é nula, já quando esta entrada vai para 1 provoca a transição para o estado COUNTING.
- COUNTING: é o estado que incrementa o contador, ou seja, a cada pulso de clock uma unidade é adicionada no registro de contagem. E, quando este atinge o *ceil* definido a máquina transita para o estado de DONE.
- DONE: é o estado que indica que a contagem atingiu o valor desejado, ou seja, diferente dos outros estados, este define a saída *done* em 1. A máquina fica parada nesse estado enquanto não receber um sinal de start, e ao receber esse sinal volta para o estado COUNTING.

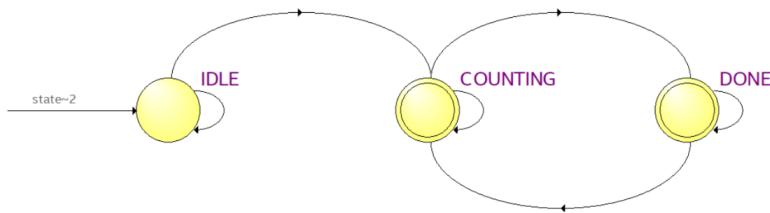


Figura 37 – Maquina de Estados counter_ceil.

Fonte: Própria.

Sendo assim, a Figura 38 apresenta a camada de aplicação explicitada, juntamente com a camada de *link* e a de transporte do transmissor.

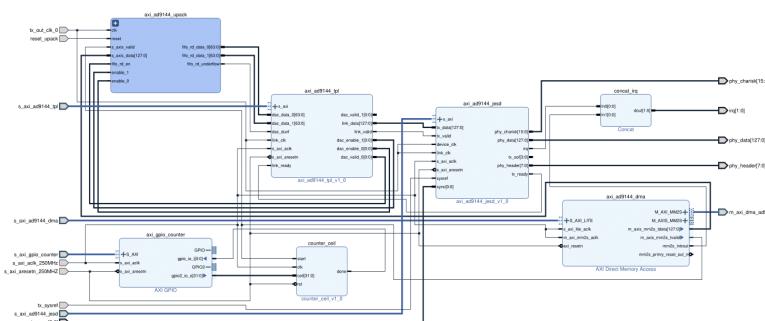


Figura 38 – Diagrama de blocos do DAC.

Fonte: Própria.

4.1.3 Bloco de Processamento

Como já apresentado anteriormente, a arquitetura ZynqMP divide-se na parte PL e PS. As camadas anteriores compõem a PL, desta maneira, a seguir são apresentados os IPs que constituíram a parte PS (Figura 39), sendo estes:

- AXI Interconnect
- Concat
- Constant
- Processor System Reset
- Zynq Ultrascale+ MPSoC

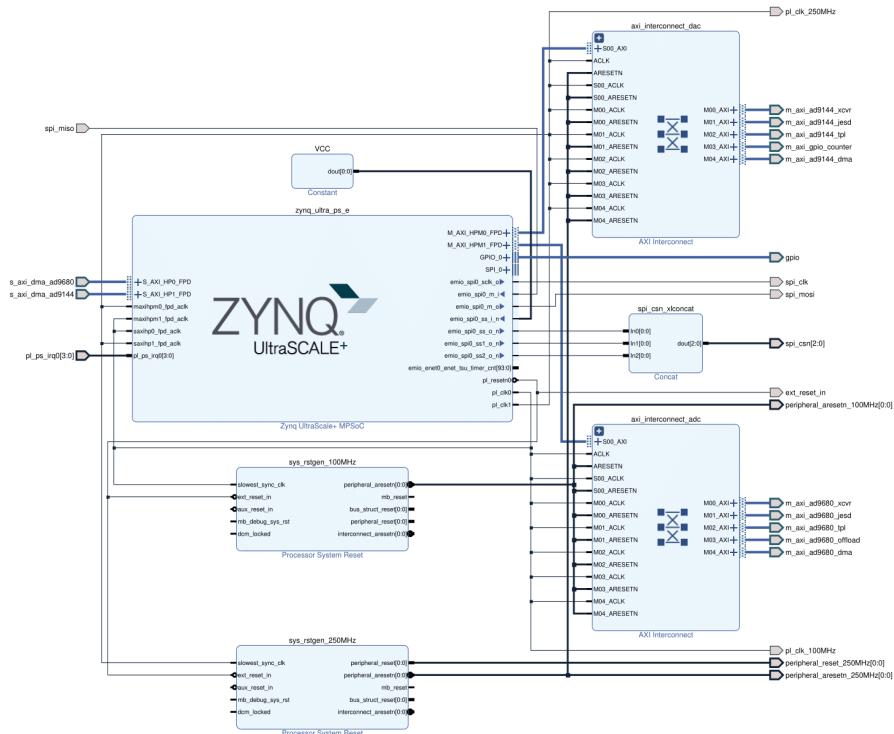


Figura 39 – Diagrama de blocos do PS.

Fonte: Própria.

Ademais, vale salientar que os IPs Processors System Reset têm como objetivo gerar os sinais de reset referentes aos dois domínios de *clock* presentes no sistema e os AXI Interconnect visam organizar as ligações das interfaces AXI com as camadas desenvolvidas. Além disso, no Zynq Ultrascale+ MPSoc foram feitas as seguintes configurações:

- Habilitação de duas interfaces AXI mestres, uma para controle das camadas do receptor e outra para o transmissor.

- Habilitação de duas interfaces AXI escravas, uma para receber as amostras do sinal do receptor e outra para enviar as amostras do sinal para o transmissor.
- GPIO para controle de sinais extras dos chips de conversão.
- SPI para configuração dos chips de conversão.
- Duas saídas de *clock*, uma a 100 MHz para os blocos do receptor e outra a 250 MHz para os blocos do transmissor.

4.1.4 Bloco Superior

Deste modo, com todos os blocos apresentados anteriormente, foi realizada a conexão de todas as partes desenvolvidas, resultando no diagrama de blocos da Figura 40.

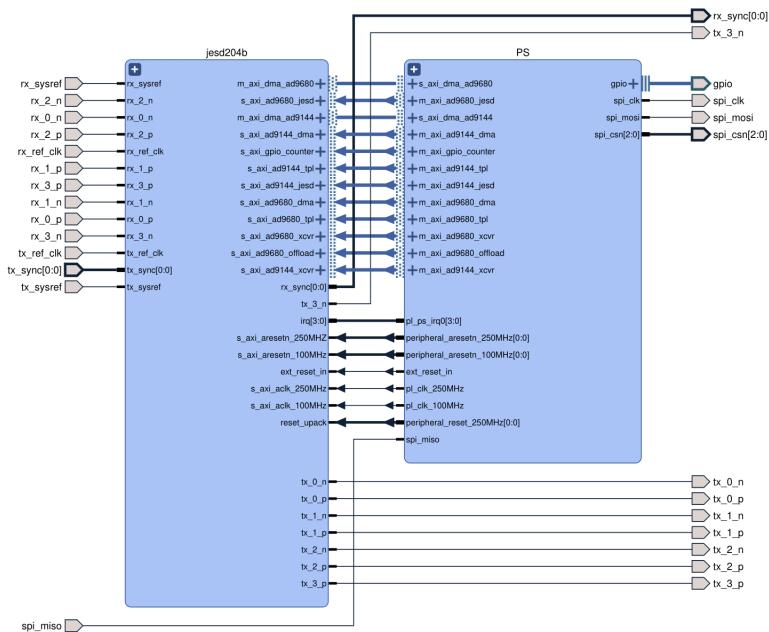


Figura 40 – Diagrama de blocos do JESD204B e PS.

Fonte: Própria.

Além disso, foi definido um arquivo Verilog que engloba o bloco apresentado para manipular as entradas e saídas do sistema. Também, foram adicionadas as *constraints* de tempo e dos pinos referentes à placa utilizada. Por fim, gerou-se o *bitstream* do *hardware* aqui descrito e a sua exportação para um arquivo XSA.

4.2 Aplicação de controle

Esta seção volta-se a apresentar os detalhes dos passos realizados para o desenvolvimento do *software* da aplicação de controle do *hardware* desenvolvido.

4.2.1 Códigos Estudados

Como já supracitado, a Analog Devices fornece uma gama de ferramentas para trabalhar com o protocolo JESD204B. Sendo assim, serão apresentados a seguir os códigos da Analog Devices que foram utilizados como base para o desenvolvimento da aplicação *bare metal*.

- ad9144.h e ad9144.c: realizam a configuração do conversor DAC por meio do SPI;
- ad9523.h e ad9523.c: realizam a configuração do *clock device* através do SPI;
- ad9680.h e ad9680.c: realizam a configuração do conversor ADC por meio do SPI;
- adc_core.h e adc_core.c: realizam a configuração de alguns parâmetros da camada de transporte do ADC por meio do AXI;
- dac_buffer.h e dac_buffer.c: realizam a gravação na memória de um sinal presente em um vetor, dado um endereço inicial recebido como parâmetro;
- dac_core.h e dac_core.c: realizam a configuração de alguns parâmetros da camada de transporte do DAC por meio do AXI;
- dmac_core.h e dmac_core.c: realizam a configuração dos parâmetros da recepção e transmissão dos dados pelo AXI DMA;
- fmdaq2.c: arquivo principal, que chama a função de todos os outros para executar as configurações e, assim, realizar os testes de integridade dos dados, a aquisição e transmissão de sinais;
- jesd_core.h e jesd_core.c: realizam a configuração de alguns parâmetros da camada de link por meio do AXI;
- platform_drivers.h e platform_drivers.c: realizam as comunicações por meio do SPI e do GPIO;
- xcvr_core.h e xcvr_core.c realizam a configuração de alguns parâmetros da camada física por meio do AXI;
- xilinx_qpll.h e xilinx_qpll.c: realizam a configuração da QPLL presente na camada física;
- xilinx_xcavr_channel.h e xilinx_xcavr_channel.c: realizam a configuração da PLL presente na camada física.

4.2.2 Plataforma de Hardware e Reestruturação

Como citado na seção de Sistema em FPGA, gerou-se um arquivo XSA com o *hardware* descrito. Então, utilizou-se deste arquivo para criar uma Plataforma de Hardware no Vitis, o que possibilitou a criação de uma aplicação com os códigos estudados. Desta maneira, tornou-se viável a compilação e reestruturação dos códigos, logo cita-se as seguintes mudanças realizadas:

- Arquivo JESD204B: para melhor organização do controle do protocolo o arquivo fmcdaq2.c foi transformado no jesd204b.c, além da modificação do nome, o conteúdo presente neste arquivo foi organizado em funções de inicialização do protocolo, inicialização da aquisição e inicialização da transmissão.
- Arquivo de DMA: como o arquivo dmac_core.c era capaz de controlar apenas o IP da Analog Devices, foram adicionadas funções para controlar o IP da Xilinx, além de separar funções para configurar os IPs e outras para inicializar a transferência de dados.
- Arquivo *buffer* do DAC: o arquivo dac_buffer.c realizava a gravação de um sinal senoidal na memória com uma defasagem entre os canais, então, modificou-se o algoritmo para que deste arquivo guarde sinais personalizados na memória sem nenhuma defasagem entre os canais.
- Arquivo principal: adicionou-se um arquivo main.c que inicializa o protocolo e entra em *loop* infinito para realizar a aquisição e transmissão de sinal como desejado no projeto do sistema.

Por fim, o Vitis também possibilitou utilizar o modo de depuração para validar os códigos gerados. Os resultados obtidos serão apresentados em um capítulo posterior.

4.2.3 Compilação Petalinux e Adaptação Projeto

Assim como para criar uma Plataforma, utilizou-se o arquivo XSA para criar um projeto personalizado a fim de compilar uma distribuição do sistema operacional Petalinux. Desta forma, por meio da interface do menu de configuração do *kernel* (Figura 41), foram definidas as configurações desejadas para o sistema, como habilitação de *drivers* SPI e habilitação de servidor de compartilhamento de arquivos. Por fim, foram gerados os arquivos necessários para realizar o *boost*, os quais foram adicionados em um cartão de memória que foi inserido na placa para realização dos testes.

No que se refere ao código da aplicação, este necessitou passar por algumas mudanças para se adaptar ao novo ambiente de execução. Deste modo, pode-se citar as seguintes adaptações:

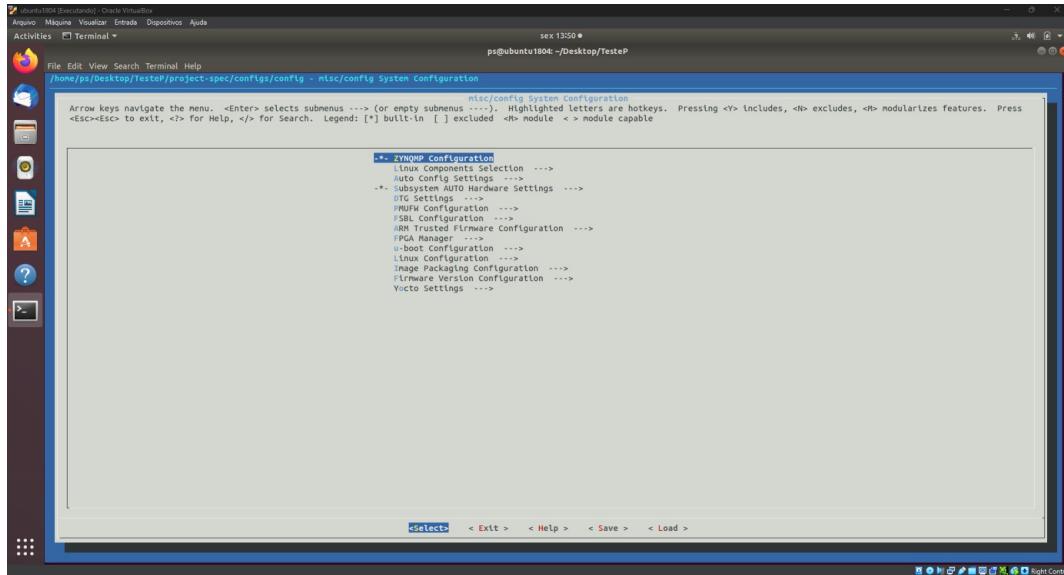


Figura 41 – Tela de configuração do Petalinux.

Fonte: Própria.

- Endereços AXI: No modo *bare metal* o Vitis possibilita o acesso direto ao arquivo xparameters.h gerado na plataforma de *hardware*, dessa forma, todos os endereços AXI presentes no *hardware* podem ser acessados através desse arquivo. Já no projeto da aplicação para Petalinux não há como acessar o arquivo citado. Logo, foram criadas definições para todos os endereços de memória.
- Mapeamento de Memória: No modo *bare metal* era possível acessar diretamente os endereços físicos da memória, contudo com um SO em operação tal ação não é permitida devido ao gerenciamento de memória. Logo, necessitou-se criar funções para abrir o sistema de arquivos, alocar memória, desalocar memória e realizar a leitura ou escrita da memória.
- Funções do SPI: As funções associadas à comunicação SPI estavam totalmente dependentes da plataforma de *hardware* presente no projeto *bare metal*, isso levou a modificação completa da função de inicialização e da função de leitura e escrita, as quais agora operam com *drivers* SPI disponibilizados pelo Petalinux.
- Funções do GPIO: As funções da comunicação com GPIO dependiam da plataforma de *hardware*, então, elas foram alteradas para funcionar com o mapeamento de memória supracitado.
- Funções de Arquivos: No modo *bare metal* utilizava-se dos recursos presentes no Vitis para importar e exportar dados de sinais adquiridos ou a serem transmitidos. Desta forma, como o Petalinux apresenta as vantagens de um SO as transferências de dados podem ser realizadas por meio de arquivos. Posto isso, foram criadas funções para realizar a escrita e leitura de sinais em arquivos.

Por fim, vale ressaltar que o Vitis é capaz de se conectar à placa a partir da ferramenta *TCF Agent* e, desta forma, transferir um arquivo executável com a aplicação desenvolvida. Tal arquivo possui a extensão .elf e pode ser executado a partir da sua invocação pela linha de comando do Petalinux.

4.3 Ambiente de Testes

Para a validação do *hardware* e *software* desenvolvidos, foi montado um ambiente de teste que conta com um gerador de ondas, a placa com arquitetura ZynqMP, a placa com os chips conversores e um osciloscópio. Portanto, a seguir serão apresentados os detalhes dos dispositivos utilizados.

4.3.1 Gerador de Ondas

O gerador de ondas utilizado para fornecer os sinais para os testes do sistema é o modelo AFG1022 da Tektronix (Figura 42). Este gerador possui uma largura de banda de 25 MHz, até 10 Volts de pico a pico de amplitude de saída e conta com duplo canal ([Grupo Zilocchi, 2024](#)).



Figura 42 – Gerador de Ondas.

Fonte: Retirado de [Grupo Zilocchi \(2024\)](#).

4.3.2 Placa iW-RainboW-G30M

O dispositivo que contém a FPGA utilizada para implementar o sistema desenvolvido é a placa iW-RainboW-G30M presente no kit de desenvolvimento (Figura 43) da empresa iWave, o qual conta com sistema em módulo Zynq Ultrascale+ MPSoc ZU7 ([iWave, 2022](#)). Outro ponto importante é que esta placa possui os microprocessadores que irão executar o SO desenvolvido.



Figura 43 – Kit de desenvolvimento com a iW-RainboW-G30M.

Fonte: Retirado de [iWave \(2022\)](#).

4.3.3 Placa AD-FMCDAQ2-EBZ

No que se refere aos chips conversores, o dispositivo utilizado é a placa AD-FMCDAQ2-EBZ (Figura 44) da Analog Devices. Vale ressaltar que o chip ADC em questão é o AD9680, o qual é capaz de converter amostras a uma taxa de 1 GSPS e com precisão de 14 bits. Já o chip DAC é o AD9144, o qual tem capacidade de operação de 2,8 GSPS e precisão de 16 bits. Por fim, o chip gerador de *clock* é o AD9523 com saída de 1 GHz, além de possuir outros componentes de gerenciamento de energia. ([Analog Devices, 2023a](#))

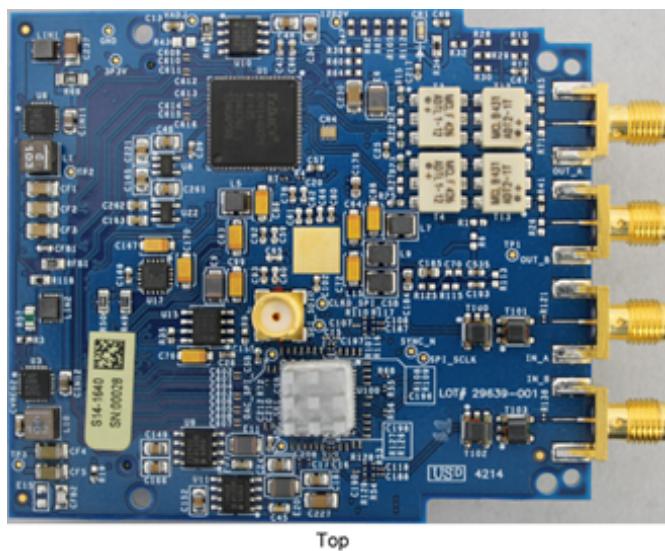


Figura 44 – Placa AD-FMCDAQ2-EBZ.

Fonte: Retirado de [Analog Devices \(2023a\)](#).

4.3.4 Osciloscópio

O osciloscópio utilizado para apresentar os sinais gerados pelo sistema foi o modelo MO-3104 da Minipa (Figura 45). Este osciloscópio possui uma largura de banda de 100 MHz, contém 4 canais e é capaz de atingir uma taxa de amostragem de 1 GSPS ([Minipa, 2024](#)).



Figura 45 – Osciloscópio.

Fonte: Retirado de [Minipa \(2024\)](#).

4.3.5 Ambiente de Teste Completo

Deste modo, o ambiente de teste completo é em base à união dos componentes supracitados. Cabe apenas ressaltar que a comunicação da placa iW-rainboW-G30M com a placa AD-FMCDAQ2-EBZ é realizada por meio dos conectores FMC.

Também, é válido pontuar que, como a placa dos chips de conversões possui dois canais, estes foram testados em conjunto e isoladamente. Portanto, a Figura 46 apresenta a configuração do ambiente de teste explicitado com ambos os canais conectados.



Figura 46 – Ambiente Teste Completo.

Fonte: Própria.

5 Experimentos e Resultados

Este capítulo busca apresentar os resultados adquiridos com o trabalho aqui exposto. Cabe ressaltar que o *hardware* e *software* apresentados no capítulo de Desenvolvimento Prático já compõem o resultado objetivado para este trabalho.

Ademais, busca-se, primeiramente, apresentar mais detalhes sobre a configuração do ambiente de testes e os resultados esperados. Então, nas próximas seções demonstra-se os detalhes do *hardware* implementado, os resultados obtidos na depuração do sistema bare metal, focando separadamente no ADC e no DAC. E, por fim, exibe-se o sistema operando com o Petalinux embarcado.

5.1 Configuração do sistema

Explicitou-se em capítulos anteriores os equipamentos utilizados para montar o ambiente de testes. Contudo, a Figura 47 apresenta um diagrama de blocos com tais componentes, mas, adiciona um detalhamento sobre o caminho projetado para os dados, durante a aquisição e transmissão de um sinal.

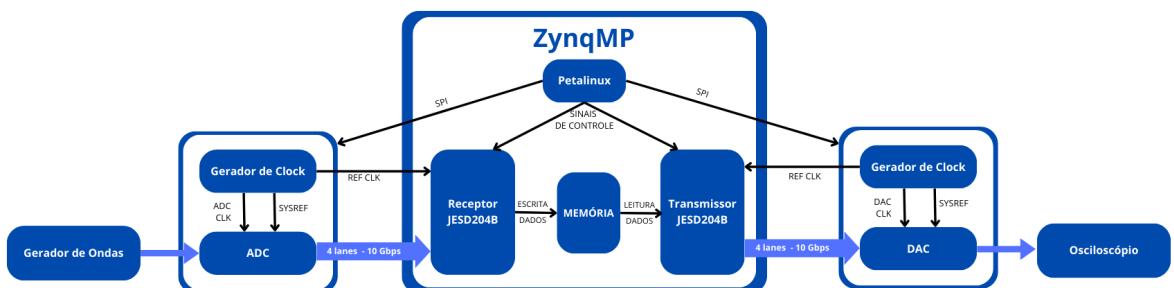


Figura 47 – Diagrama de Blocos do Sistema.

Fonte: Própria.

Logo, o caminho projetado consiste na emissão do sinal através do gerador de ondas, aquisição pelo chip ADC e armazenamento na memória da placa ZynqMP, por meio do protocolo JESD204B. Posteriormente, o JESD204B envia o sinal presente na memória para o chip DAC, o qual transmite essas informações convertidas para o osciloscópio.

Sendo assim, cabe ressaltar que as depurações realizadas para o *software bare metal* são feitas com distintos fluxos. Ou seja, primeiramente valida-se apenas o caminho de dados do receptor, esperando obter os sinais do gerador na memória da placa. E, em seguida, verifica-se isoladamente o caminho de dados do transmissor, esperando obter no osciloscópio os sinais presentes na memória da placa.

Por fim, quando considerado o sistema completo, cabe à aplicação presente no Petalinux controlar os chips de conversão e de *clock*, por meio das conexões SPI, além de configurar as camadas do protocolo JESD204B. Já no que se refere ao fluxo de dados, este foi projetado para que o sistema seja capaz de capturar um sinal e, logo em seguida, transmiti-lo. Sendo assim, espera-se que ao inserir o sinal do gerador de ondas no sistema, este mesmo sinal seja observado no osciloscópio conectado à saída do sistema.

5.2 Hardware em FPGA

Quando se trata de sistemas em FPGA, é vital para o correto funcionamento que os requisitos de temporização sejam cumpridos pelo sistema criado. Isso ocorre porque foram definidas *constraints* de *clock* por meio de um arquivo XDC, como apresentado a seguir:

```

1 create_clock -name rx_ref_clk -period 2.00 [get_ports
    rx_ref_clk_p]
2 create_clock -name tx_ref_clk -period 2.00 [get_ports
    tx_ref_clk_p]
3 create_clock -name spi_clk -period 50.0 [get_pins
    jesd204_teste_i/PS/zynq_ultra_ps_e_0/inst/PS8_i/
    EMIOSPIOSCLK0]
4 create_generated_clock -name rx_div_clk [get_pins
    jesd204_teste_i/jesd204b/phy_shared/inst/jesd204_teste_i/
    util_daq2_xcvr_hier/util_daq2_xcvr/inst/i_xch_0/
    i_gthe4_channel/RXOUTCLK]
5 create_generated_clock -name tx_div_clk [get_pins
    jesd204_teste_i/jesd204b/phy_shared/inst/jesd204_teste_i/
    util_daq2_xcvr_hier/util_daq2_xcvr/inst/i_xch_0/
    i_gthe4_channel/TXOUTCLK]
```

Logo, se os requisitos de tempo não forem atendidos, erros podem ser introduzidos nos dados presentes no sistema. Então, a Figura 48 apresenta os resultados obtidos pelo Vivado em relação aos possíveis atrasos. Portanto, como é possível observar, todos os tempos ficaram positivos e atendem a todos os requisitos do sistema.

Além disso, a Figura 49 apresenta o número de *flip-flops*, *look up tables*, BRAMs, blocos DSP e potência utilizados. Também, a Figura 50 demonstra as técnicas do Vivado utilizadas para realizar a síntese e implementação, as quais não possuem restrições específicas, mas buscam atender às *constraints* definidas.

Por fim, a Figura 51 apresenta uma visão do dispositivo, a qual destaca as partes da FPGA que foram usadas durante a implementação realizada pelo Vivado. Também,

Design Timing Summary								
Setup		Hold			Pulse Width			
Worst Negative Slack (WNS):	0,038 ns	Worst Hold Slack (WHS):	0,010 ns		Worst Pulse Width Slack (WPWS):	0,500 ns		
Total Negative Slack (TNS):	0,000 ns	Total Hold Slack (THS):	0,000 ns		Total Pulse Width Negative Slack (TPWS):	0,000 ns		
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0		Number of Failing Endpoints:	0		
Total Number of Endpoints:	76882	Total Number of Endpoints:	76882		Total Number of Endpoints:	30968		

All user specified timing constraints are met.

Figura 48 – Resultados temporização do *hardware*.**Fonte:** Própria.

Name	Status	LUT	FF	BRAM	DSP	Total Power	Part
✓ synth_1 (active)	synth_design Complete!	12	8	0	0		xczu7ev-fbvb900-1-e
✓ impl_1	write_bitstream Complete!	24512	28880	264.5	16	5.494	xczu7ev-fbvb900-1-e

Figura 49 – Detalhes do sistema em FPGA pelo Vivado.

Fonte: Própria.

Name	Part	Report Strategy	Run Strategy
✓ synth_1 (active)	xczu7ev-fbvb900-1-e	Vivado Synthesis Default Reports (Vivado Synthesis 2022)	Vivado Synthesis Defaults (Vivado Synthesis 2022)
✓ impl_1	xczu7ev-fbvb900-1-e	Vivado Implementation Default Reports (Vivado Implementation 2022)	Vivado Implementation Defaults (Vivado Implementation 2022)

Figura 50 – Técnicas de Síntese e Implementação do Vivado.

Fonte: Própria.

cabe ressaltar que as colunas que se destacam na imagem são os blocos BRAMs que foram utilizados.

5.3 Sistema Bare Metal

Como comentado anteriormente para a primeira versão do sistema de controle, foi desenvolvido um código independente de um SO. Logo, como pode ser observado na parte inferior esquerda da Figura 46, o cabo JTAG (*Joint Test Action Group*) é conectado na placa para que o Vitis possa programar a FPGA e executar a aplicação desenvolvida. Logo, os detalhes dos experimentos serão apresentados a seguir.

5.3.1 Experimentos ADC

Os testes da parte do sistema voltada para o ADC (Figura 52) consistiram na configuração do protocolo e gravação das amostras adquiridas do gerador de funções na memória da placa iW-Rainbow-G30M. Logo, o ambiente de teste pode ser reduzido à configuração presente na Figura 53.

Desta maneira, foram testados cada canal da placa AD-FMCDAQ2-EBZ separadamente e em conjunto. Além de utilizar diferentes combinações de ondas do gerador de sinais.

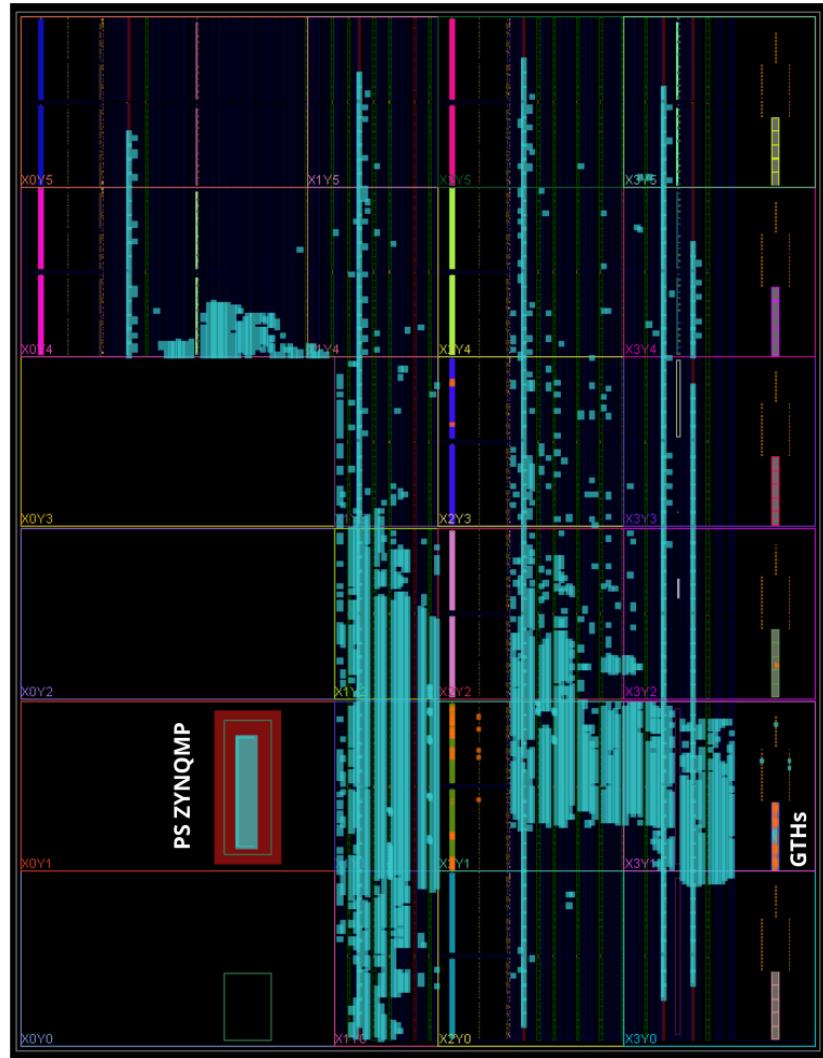


Figura 51 – Implementação do sistema em FPGA.

Fonte: Própria.

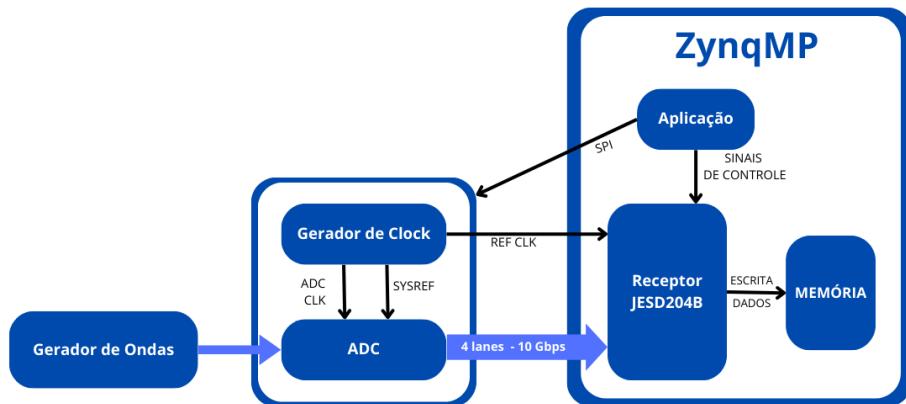


Figura 52 – Diagrama de Blocos do Fluxo do ADC.

Fonte: Própria.



Figura 53 – Ambiente Teste do ADC.

Fonte: Própria.

Posto isto, os resultados das amostras presentes na memória do sistema são apresentados por representações gráficas, as quais estão presentes nas Figuras 54, 55, 56, 57 e 58.

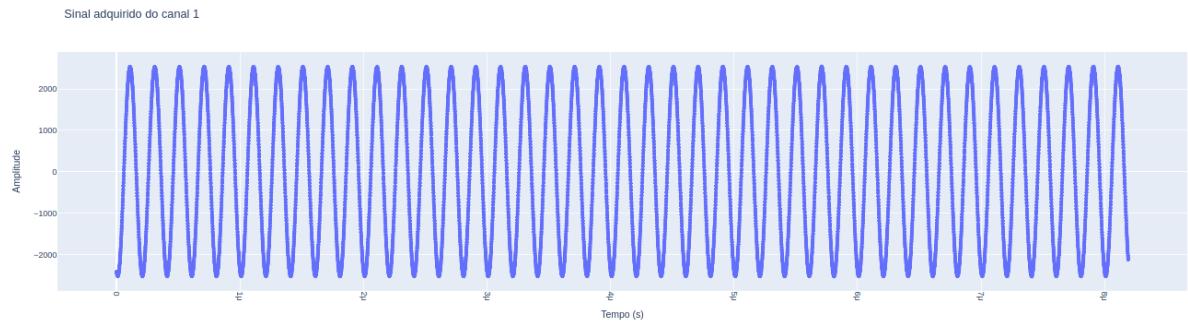


Figura 54 – Experimento canal 1 seno de 5 MHz e canal 2 não conectado.

Fonte: Própria

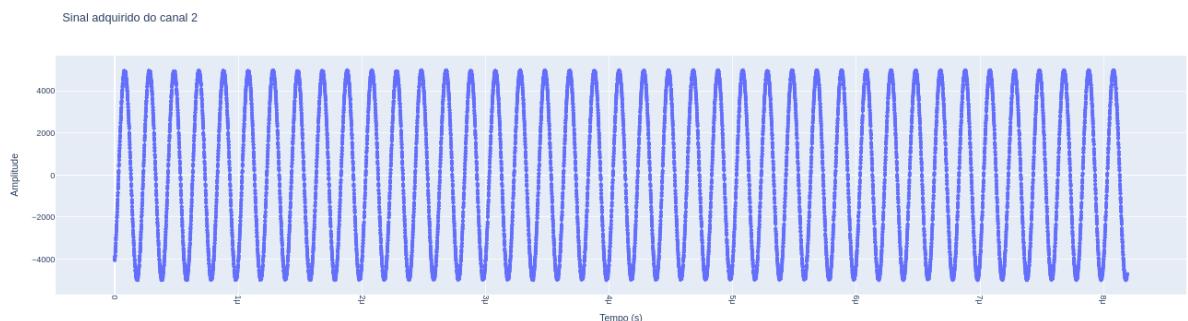


Figura 55 – Experimento canal 1 não conectado e canal 2 seno de 5 MHz.

Fonte: Própria

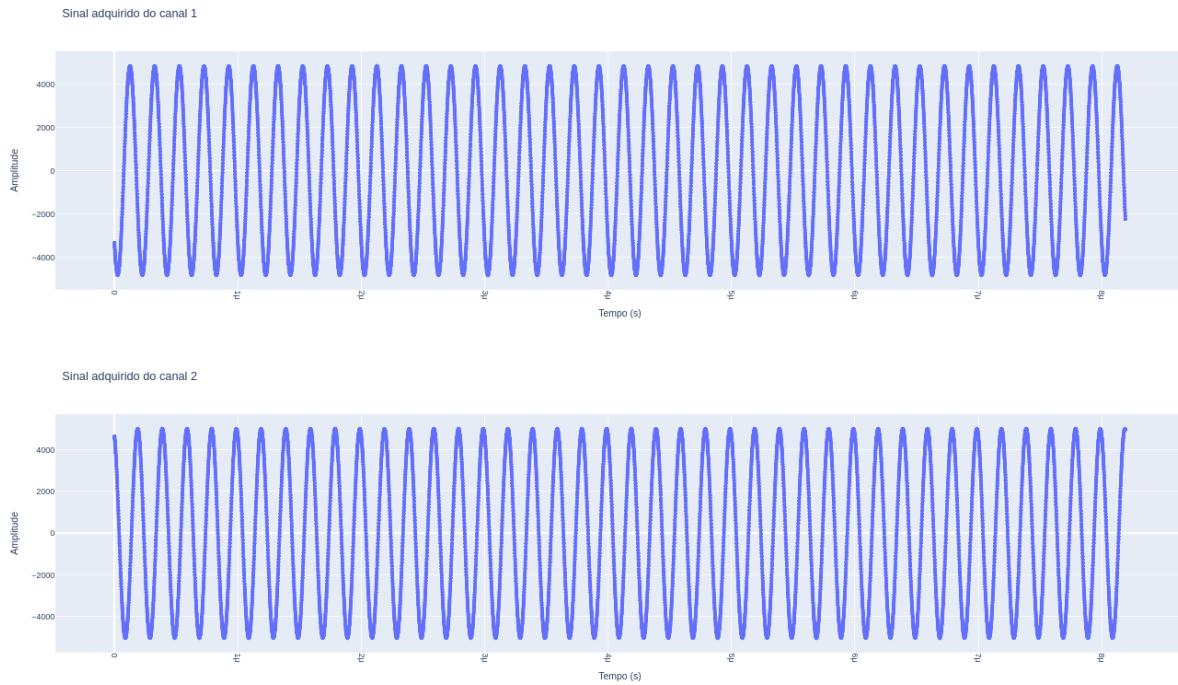


Figura 56 – Experimento canal 1 seno de 5 MHz e canal 2 seno de 5 MHz.

Fonte: Própria

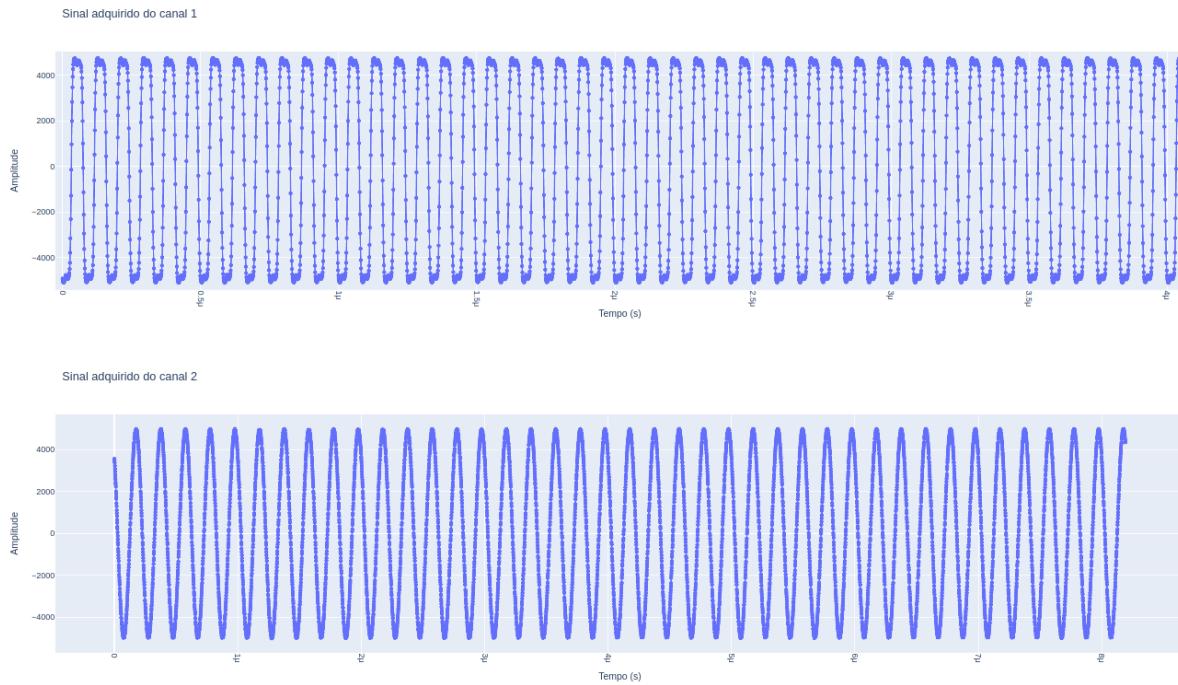


Figura 57 – Experimento canal 1 onda quadrada 12 MHz e canal 2 seno de 5 MHz.

Fonte: Própria

Ressalta-se que o sistema operou a 1 GSPS na aquisição de todas as combinações dos canais e ondas testados, como pode ser observado pela Figura 59. Esta apresenta uma visão ampliada da onda quadrada de 12 MHz e do seno de 5 MHz, demonstrando que um ciclo de cada uma dos sinais tem, respectivamente, 83 e 200 amostras.

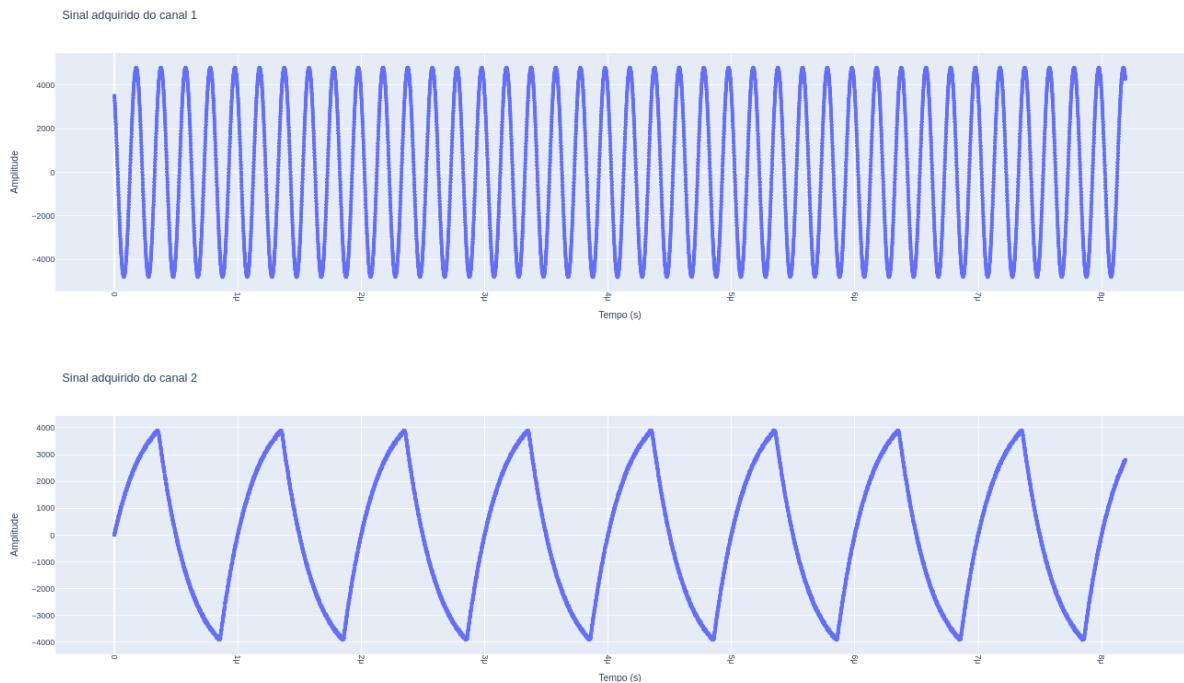


Figura 58 – Experimento canal 1 seno de 5 MHz e canal 2 rampa de 1 MHz.

Fonte: Própria

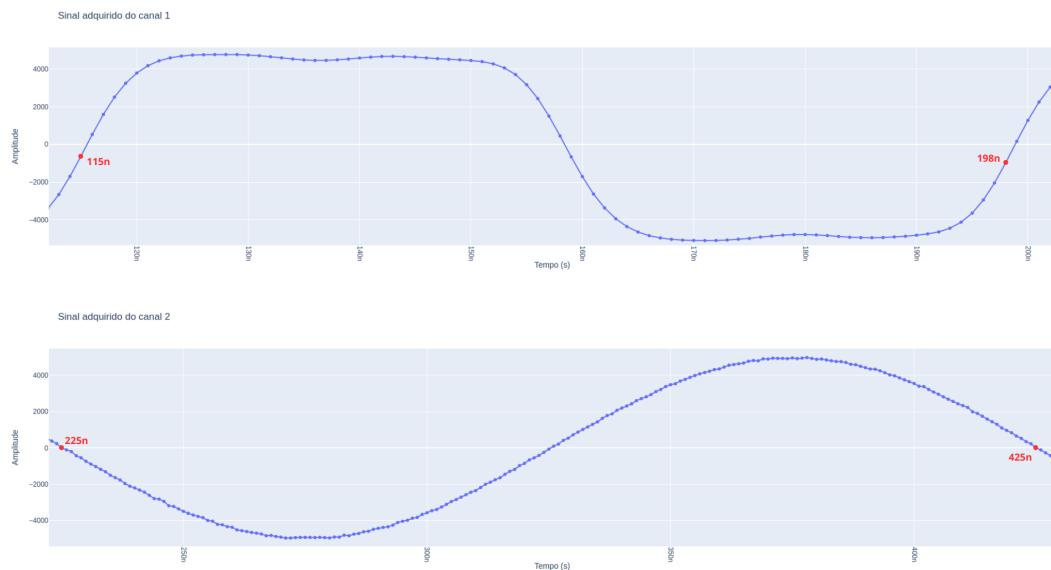


Figura 59 – Visão ampliada dos sinais obtidos pelo ADC.

Fonte: Própria

Portanto, a partir das ondas apresentadas, é possível observar que a comunicação e o controle do chip ADC, por meio do protocolo JESD204B, ocorreram como o esperado. Haja vista que as ondas emitidas pelo gerador de sinais mantiveram suas características nas amostras retiradas da memória do sistema.

5.3.2 Experimentos DAC

Os testes da parte do sistema voltada para o DAC (Figura 60) consistiram na configuração do protocolo e leitura das amostras na memória da placa iW-Rainbow-G30M para transmiti-las ao osciloscópio. Logo, o ambiente de teste pode ser reduzido à configuração presente na Figura 61.

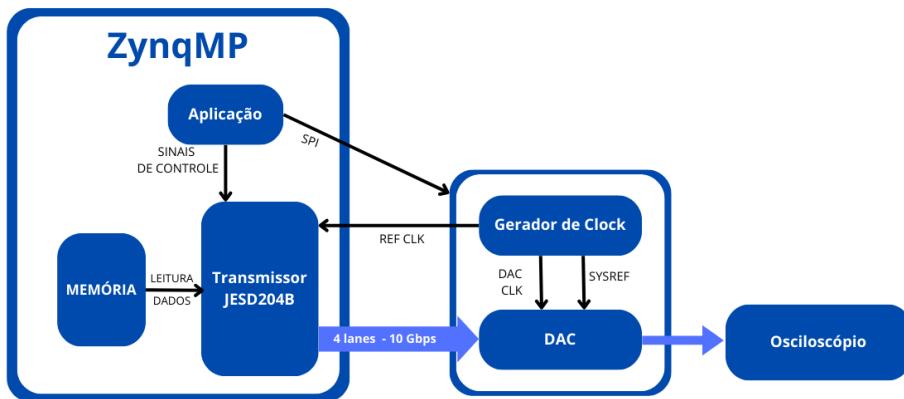


Figura 60 – Diagrama de Blocos do Fluxo do DAC.

Fonte: Própria.

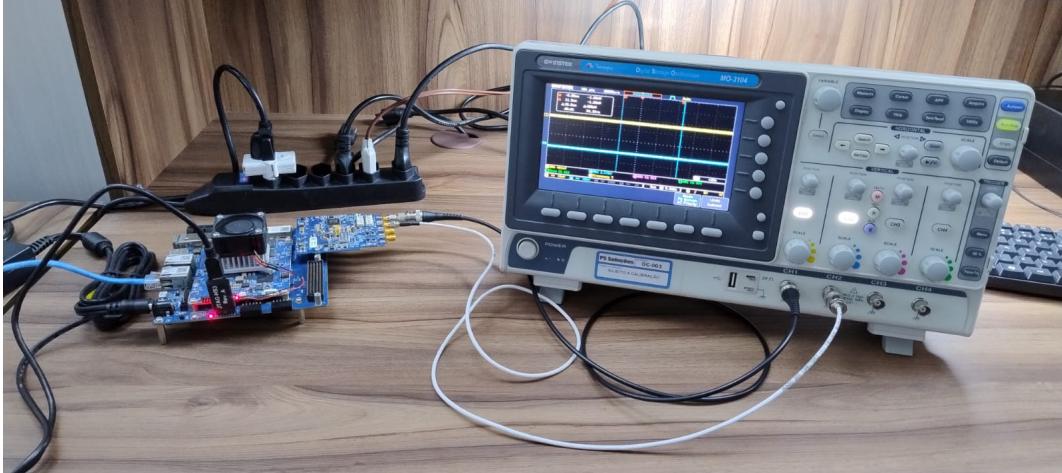
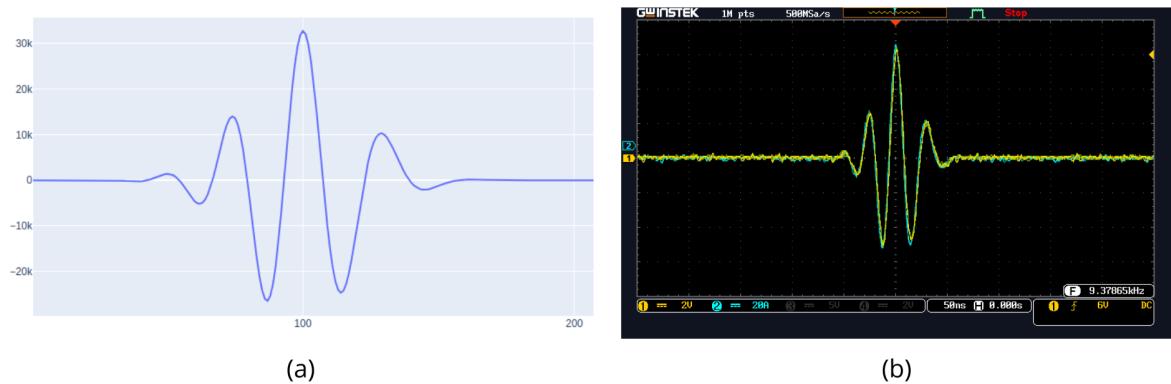


Figura 61 – Ambiente Teste do DAC.

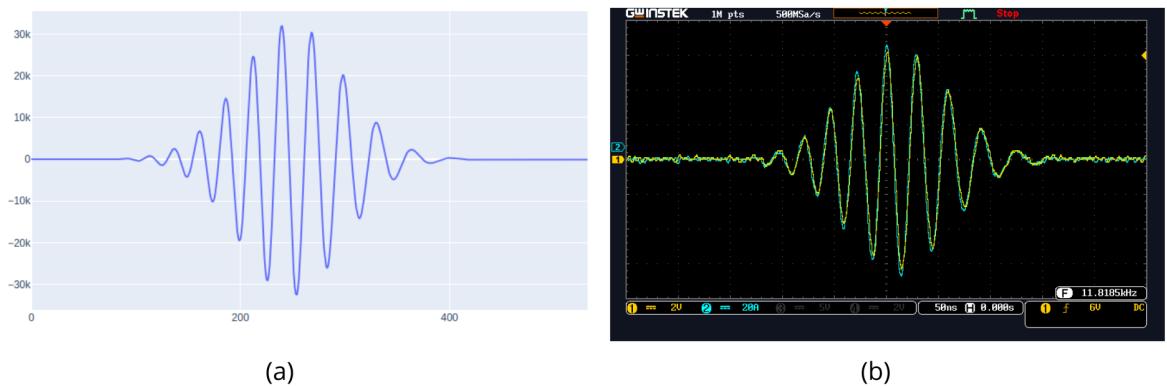
Fonte: Própria.

Sendo assim, foram testados em paralelo os dois canais da placa AD-FMCDAQ2-EBZ, alterando apenas as amostras presentes na memória. Este procedimento visou o envio de três *chirps*, cuja frequência varia linearmente de 1 MHz a 100 MHz, com duração distinta para cada sinal. As Figuras 62, 63 e 64 apresentam os sinais *chirps* inseridos na memória ("Esperado") e aqueles emitidos pelo DAC ("Obtido").

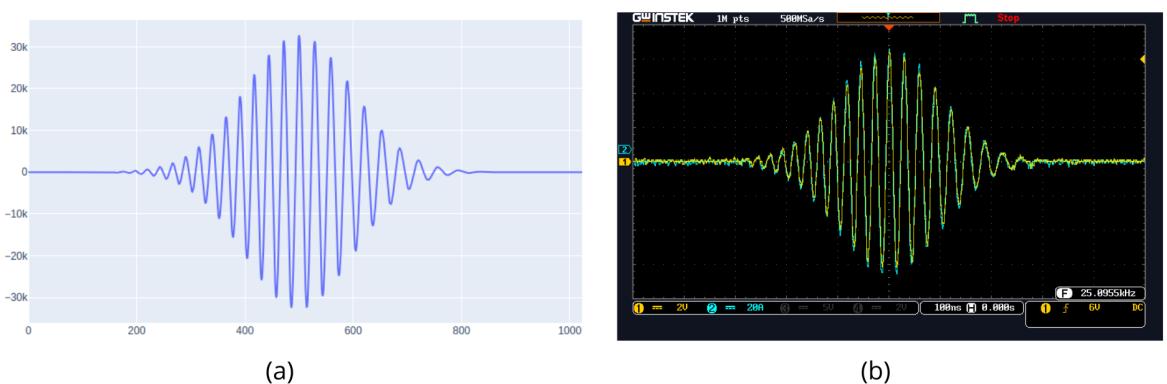
Ressalta-se que o sistema operou a 1 GSPS na transmissão de todos os sinais testados. Portanto, a partir das ondas apresentadas, é possível observar que a comunicação

Figura 62 – Experimento com *chirp* duração de 200 ns.

(a) Esperado | (b) Obtido
Fonte: Própria.

Figura 63 – Experimento com *chirp* duração de 500 ns.

(a) Esperado | (b) Obtido
Fonte: Própria.

Figura 64 – Experimento com *chirp* duração de 1 us.

(a) Esperado | (b) Obtido
Fonte: Própria.

e o controle do chip DAC, por meio do protocolo JESD204B, ocorreram como o esperado, haja vista que todos os sinais obtidos são semelhantes aos esperados.

Outro ponto importante testado foi o funcionamento do bloco contador, o qual permite programar a repetição do sinal transmitido. Vale lembrar que o contador está conectado a um *clock* de 250 MHz, o que fornece uma contagem de 4 ns a cada *clock*. Sendo assim, foram realizados testes para três valores diferentes de *ceil*, os quais estão presentes nas Figuras 65, 66 e 67.



Figura 65 – Experimento com *ceil* igual a 25000.

Fonte: Própria.

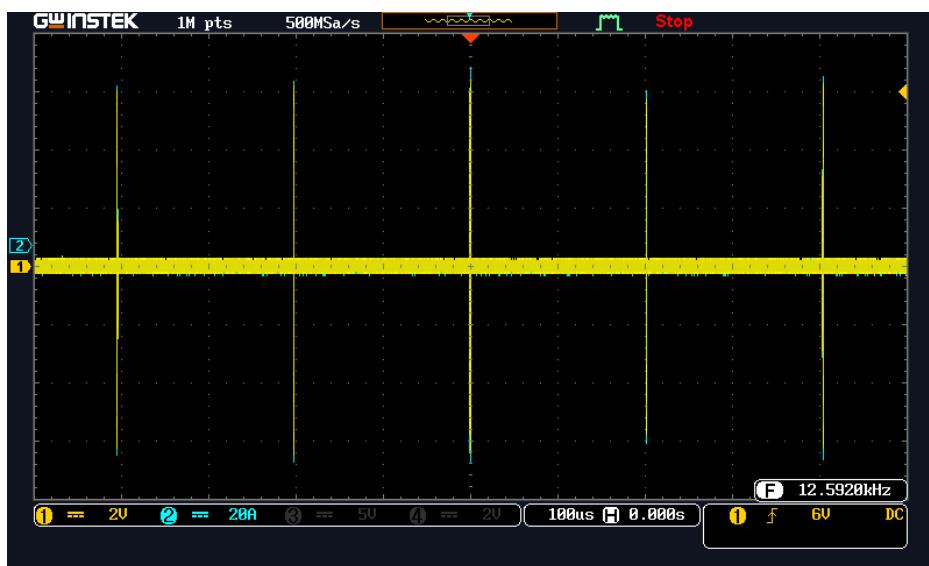


Figura 66 – Experimento com *ceil* igual a 50000.

Fonte: Própria.

Então, como cada divisão do osciloscópio utilizado corresponde a 100 μ s e cada sinal amarelo é uma repetição do *chirp* apresentado anteriormente, é possível observar que

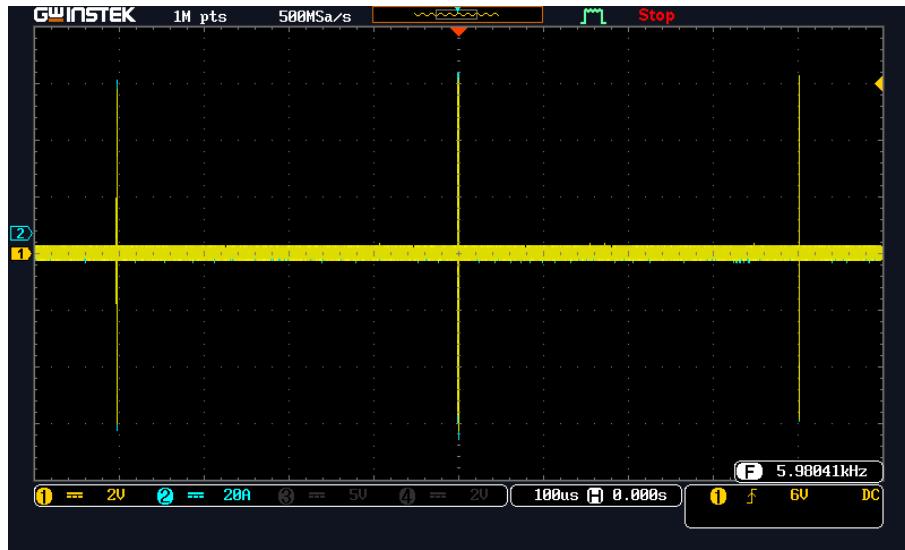


Figura 67 – Experimento com *ceil* igual a 100000.

Fonte: Própria.

o tempo de 100 μ s ($25000 * 4$ ns) na Figura 65, o de 200 μ s ($50000 * 4$ ns) na Figura 66 e o de 400 μ s ($100000 * 4$ ns) na Figura 67 correspondem aos respectivos valores inseridos no *ceil*. Validando, deste modo, a máquina de estados construída.

5.4 Sistema com Petalinux

Como apresentado na seção anterior, o sistema é capaz de operar em um ambiente *standalone*. Deste modo, esta seção busca demonstrar os testes realizados do sistema completo, ou seja, com SO, DAC e ADC operando em conjunto.

Sendo assim, o ambiente de teste utilizado para este modo já foi explicado anteriormente e apresentado pela Figura 46. Deste modo, os testes basearam-se em chamar, por meio da linha de comando (Figura 68), o arquivo executável da aplicação desenvolvida.

Então, o algoritmo invocado inicializa o protocolo JESD204B e executa um *loop* interrumpido de captura e transmissão dos sinais. Vale relembrar que a captura envolve escrever o sinal na memória e a transmissão envia o mesmo sinal que foi armazenado, a partir da leitura da memória.

Outro ponto importante evidenciado pela Figura 68 são as saídas de estado da aplicação. Porque essas saídas demonstram o *status* retornado da configuração do sistema, bem como o resultado dos testes de verificação de cada fluxo de dado. Portanto, essas saídas podem ser utilizadas como uma maneira de identificar problemas.

Como os ambos canais já foram verificados nos experimentos anteriores, para esse modo de operação só foi utilizado um canal. Contudo, vale ressaltar que foram aplicadas diferentes formas de ondas, como é apresentado pelas Figuras 69, 70 e 71.

```

TesteP:~$ sudo /run/media/mmcblk0pl/app.elf 1
-----parameters-----
qnt_samples:          4096
cycles:               4
name_signal_file:    signal200.txt (waited path:/run/media/mmcblk0pl/)
-----running-init-----
ADC 1000 MSPS; DAC 1000 MSPS
AD9523: SPI write-verify succeed (0xAD95) !
AD9680: PLL is locked! Everything OK
xcvr_setup XCVR (0x8200D000) initialization succeed!
xcvr_setup XCVR (0x82007000) initialization succeed!
AD9144: PLL is locked! Everything OK
Tx link is enabled
Measured Link Clock: 100 MHz
Link status: DATA
SYSREF captured: Yes
Rx link is enabled
Measured Link Clock: 250 MHz
Link status: DATA
SYSREF captured: Yes
adc_setup ADC core initialized (1000 MHz).
dac_setup dac core initialized (400 MHz).
ad9144_status : No problems with AD9144!
ad9144_short_pattern_test : all tests OK! .
ad9144_datapath_prbs_test : all tests OK! .
ad9144_datapath_prbs_test : all tests OK! .
AD9680_TEST_PN9 success executed!
AD9680_TEST_PN23 success executed!
ceil:0x393870
dac configured in dma mode.
jesd204b configured
-----running-ADC_DAC_ADC-----

```

Figura 68 – Terminal de Comandos do Petalinux.

Fonte: Própria.

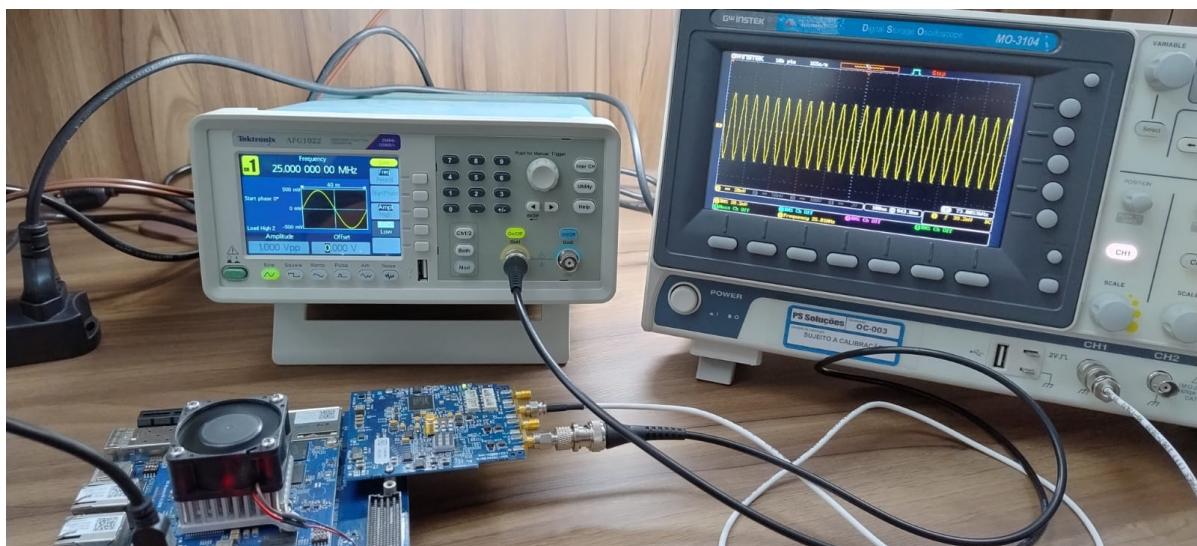


Figura 69 – Experimento com ADC e DAC - Seno 25 MHz.

Fonte: Própria.

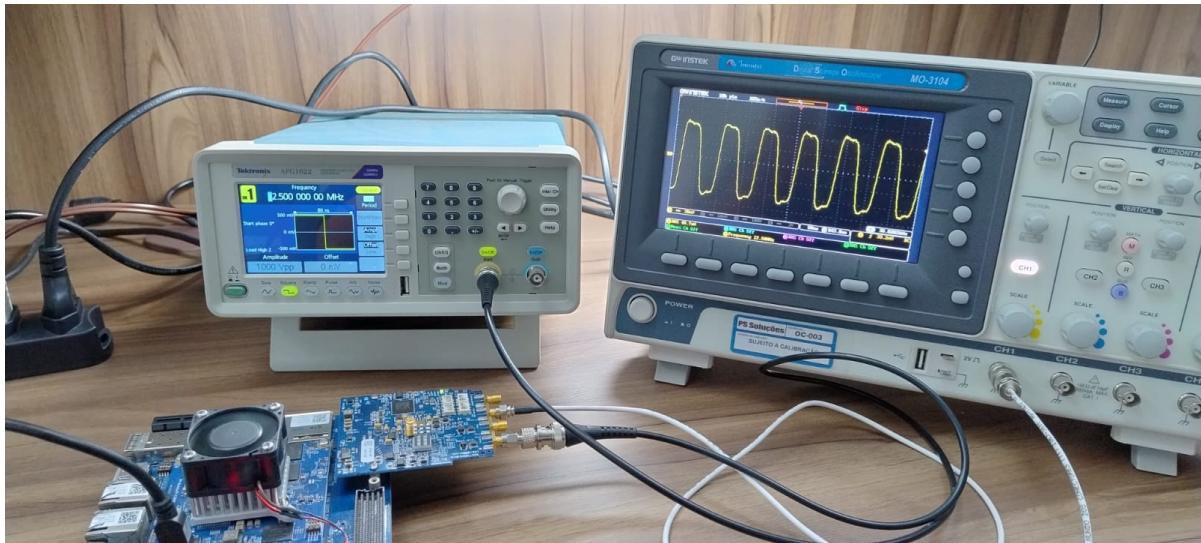


Figura 70 – Experimento com ADC e DAC - Onda quadrada 12.5 MHz.

Fonte: Própria.

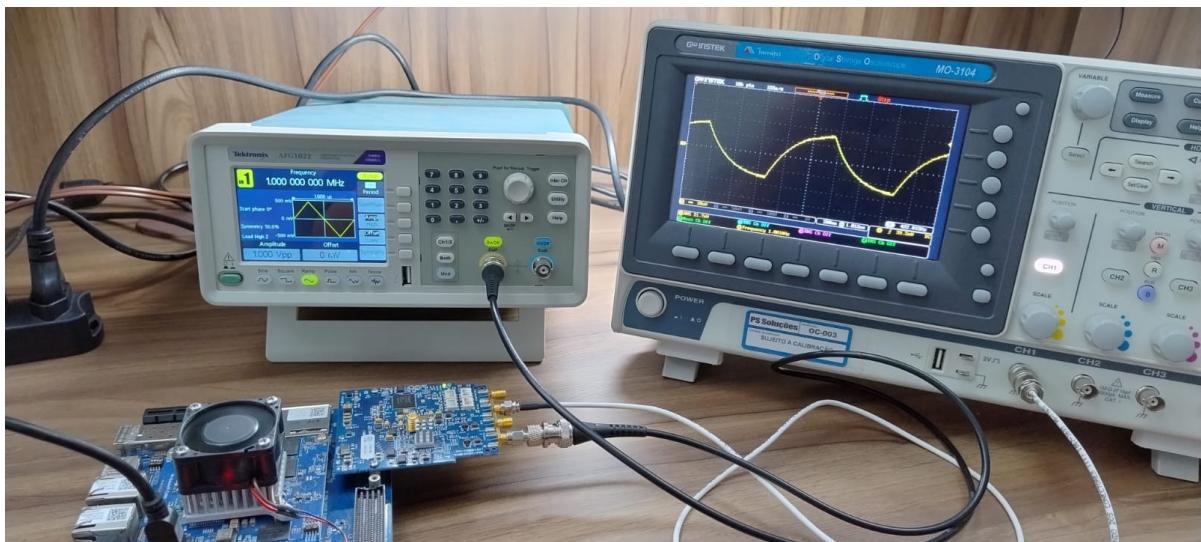


Figura 71 – Experimento com ADC e DAC - Rampa 1 MHz.

Fonte: Própria.

Portanto, como demonstrado pelos experimentos realizados, o sistema opera como o esperado. Haja vista que todas as ondas fornecidas pelo gerador de funções mantiveram suas características quando observadas no osciloscópio, mesmo após percorrerem o sistema. Também, cabe explicitar que o sistema realizou a aquisição e transmissão dos sinais a 1 GSPS, demonstrando, desta maneira, a robustez do protocolo JESD204B em operar a comunicação de alta velocidade.

6 Conclusão

Com este documento conclui-se o projeto realizado para o Trabalho Final de Graduação do curso de Engenharia de Computação da Universidade Federal de Itajubá. Ressalta-se que o projeto desenvolvido consistiu na implementação de um *hardware* compatível com o protocolo JESD204B e de um *software* de controle para esse sistema.

Além disso, foi definida a metodologia de trabalho a ser utilizada e as etapas de criação do projeto, que foram divididas em três partes principais, sendo estas: desenvolvimento do *hardware*, desenvolvimento do *software* e testes do sistema.

Quanto ao desenvolvimento prático, observa-se que os resultados obtidos por depurações mostraram-se coerentes do ponto de vista comportamental do sistema, comprovando o funcionamento lógico deste. Também, os testes em bancada permitiram observar o comportamento real do sistema, além de demonstrar que a resposta obtida foi coerente com o esperado, validando, desta forma, o projeto aqui exposto.

Outro ponto importante a se destacar é a robustez demonstrada pelo protocolo JESD204B, no que se refere à comunicação em alta velocidade, lembrando que era este um dos objetivos iniciais do sistema. Além disso, vale salientar o significativo potencial apresentado pela arquitetura ZynqMP, já que esta possibilitou uma abordagem mista entre processadores clássicos e FPGAs.

Por fim, é importante explicitar o uso extensivo de referências a manuais técnicos da AMD relacionados a chips ZynqMP e os IPs desta empresa, bem como da documentação da Analog Devices a cerca do protocolo JESD204.

Referências

AMD. *AXI GPIO v2.0 LogiCORE IP Product Guide*. 2015. Disponível em: <<https://docs.amd.com/v/u/en-US/pg144-axi-gpio>>. Acesso em: 12 set. 2024. Citado na página 40.

AMD. *Processor System Reset Module v5.0 Product Guide*. 2015. Disponível em: <<https://docs.amd.com/v/u/en-US/pg164-proc-sys-reset>>. Acesso em: 12 set. 2024. Citado na página 42.

AMD. *LogiCORE IP Concat v2.1 Product Brief*. 2016. Disponível em: <<https://docs.amd.com/v/u/en-US/pb041-xilinx-com-ip-xlconcat>>. Acesso em: 13 set. 2024. Citado 2 vezes nas páginas 40 e 41.

AMD. *LogiCORE IP Slice v1.0 Product Brief*. 2016. Disponível em: <<https://docs.amd.com/v/u/en-US/pb042-xilinx-com-ip-xlslice>>. Acesso em: 13 set. 2024. Citado na página 42.

AMD. *LogiCORE IP Constant v1.1 Product Brief*. 2018. Disponível em: <<https://docs.amd.com/v/u/en-US/pb040-xilinx-com-ip-xlconstant>>. Acesso em: 13 set. 2024. Citado na página 41.

AMD. *Vivado Design Suite User Guide: Getting Started*. 2021. Disponível em: <<https://docs.amd.com/r/2021.2-English/ug910-vivado-getting-started/What-is-the-Vivado-Design-Suite>>. Acesso em: 03 set. 2024. Citado na página 38.

AMD. *Vivado Design Suite User Guide: Using the Vivado IDE*. 2021. Disponível em: <<https://docs.amd.com/r/2021.2-English/ug893-vivado-ide/Introduction>>. Acesso em: 06 set. 2024. Citado na página 38.

AMD. *AXI Interconnect v2.1 LogiCORE IP Product Guide*. 2022. Disponível em: <<https://docs.amd.com/r/en-US/pg059-axi-interconnect>>. Acesso em: 10 set. 2024. Citado 2 vezes nas páginas 40 e 41.

AMD. *Zynq® UltraScale+™ MPSoC Data Sheet: Overview*. 2022. Disponível em: <<https://docs.amd.com/v/u/en-US/ds891-zynq-ultrascale-plus-overview>>. Acesso em: 26 ago. 2024. Citado 2 vezes nas páginas 35 e 36.

AMD. *Zynq UltraScale+ Device Technical Reference Manual*. 2023. Disponível em: <<https://docs.amd.com/r/en-US/ug1085-zynq-ultrascale-trm/Zynq-UltraScale-Device-Technical-Reference-Manual>>. Acesso em: 21 ago. 2024. Citado 5 vezes nas páginas 33, 34, 35, 36 e 37.

AMD. *Zynq UltraScale+ MPSoC Processing System Product Guide*. 2023. Disponível em: <<https://docs.amd.com/r/en-US/pg201-zynq-ultrascale-plus-processing-system/Overview>>. Acesso em: 23 ago. 2024. Citado 2 vezes nas páginas 43 e 44.

AMD. *AXI DMA LogiCORE IP Product Guide*. 2024. Disponível em: <https://docs.amd.com/r/en-US/pg021_axi_dma/Core-Overview>. Acesso em: 11 set. 2024. Citado na página 39.

AMD. *Vitis Unified Software Platform Documentation: Embedded Software Development*. 2024. Disponível em: <<https://docs.amd.com/r/en-US/ug1400-vitis-embedded/Getting-Started-with-Vitis>>. Acesso em: 16 set. 2024. Citado 2 vezes nas páginas 43 e 45.

AMD. *Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator*. 2024. Disponível em: <<https://docs.amd.com/r/en-US/ug994-vivado-ip-subsystems/Getting-Started-with-Vivado-IP-Integrator>>. Acesso em: 09 set. 2024. Citado na página 39.

Analog Devices. *AD-FMCDAQ2-EBZ*. 2023. Disponível em: <<https://www.analog.com/en/resources/evaluation-hardware-and-software/evaluation-boards-kits/eval-ad-fmcdaq2-ebz.html#eb-documentation>>. Acesso em: 21 nov. 2024. Citado na página 65.

Analog Devices. *AD-FMCDAQ2-EBZ User Guide*. 2023. Disponível em: <<https://wiki.analog.com/resources/eval/user-guides/ad-fmcdaq2-ebz>>. Acesso em: 12 ago. 2024. Citado na página 26.

Analog Devices. *ADC JESD204B/C Transport Peripheral*. 2024. Disponível em: <https://analogdevicesinc.github.io/hdl/library/jesd204/ad_ip_jesd204_tpl_adc/index.html>. Acesso em: 20 nov. 2024. Citado na página 28.

Analog Devices. *AXI ADXCVR*. 2024. Disponível em: <https://analogdevicesinc.github.io/hdl/library/axi_adxcvr/index.html>. Acesso em: 20 nov. 2024. Citado 2 vezes nas páginas 28 e 29.

Analog Devices. *AXI DMAC*. 2024. Disponível em: <https://analogdevicesinc.github.io/hdl/library/axi_dmac/index.html>. Acesso em: 20 nov. 2024. Citado 2 vezes nas páginas 28 e 29.

Analog Devices. *Channel CPACK Utility*. 2024. Disponível em: <https://analogdevicesinc.github.io/hdl/library/util_pack/util_cpack2.html>. Acesso em: 20 nov. 2024. Citado na página 32.

Analog Devices. *Channel UPACK Utility*. 2024. Disponível em: <https://analogdevicesinc.github.io/hdl/library/util_pack/util_upack2.html>. Acesso em: 20 nov. 2024. Citado na página 32.

Analog Devices. *DAC JESD204B/C Transport Peripheral*. 2024. Disponível em: <https://analogdevicesinc.github.io/hdl/library/jesd204/ad_ip_jesd204_tpl_dac/index.html>. Acesso em: 20 nov. 2024. Citado na página 28.

Analog Devices. *Data Offload*. 2024. Disponível em: <https://analogdevicesinc.github.io/hdl/library/data_offload/index.html>. Acesso em: 20 nov. 2024. Citado na página 30.

Analog Devices. *JESD204 Interface Framework*. 2024. Disponível em: <<https://wiki.analog.com/resources/fpga/peripherals/jesd204>>. Acesso em: 15 ago. 2024. Citado 2 vezes nas páginas 26 e 27.

Analog Devices. *JESD204B/C Link Receive Peripheral*. 2024. Disponível em: <https://analogdevicesinc.github.io/hdl/library/jesd204/axi_jesd204_rx/index.html>. Acesso em: 20 nov. 2024. Citado na página 29.

Analog Devices. *JESD204B/C Link Transmit Peripheral*. 2024. Disponível em: <https://analogdevicesinc.github.io/hdl/library/jesd204/axi_jesd204_tx/index.html>. Acesso em: 20 nov. 2024. Citado na página 30.

Analog Devices. *Núcleo UTIL_AXDCVR para dispositivos Xilinx*. 2024. Disponível em: <https://analogdevicesinc.github.io/hdl/library/xilinx/util_adxcvr/index.html#util-adxcvr>. Acesso em: 20 nov. 2024. Citado na página 31.

CROCKETT, L. H. *The Zynq Book, Embedded Processing with the ARM Cortex-A9 on the Xilinx Zynq-7000 All Programmable SoC*. Department of Electronic and Electrical Engineering University of Strathclyde Glasgow, Scotland UK: Strathclyde Academic Media, 2014. Citado na página 33.

FAN, H. Quickly implement jesd204b. *Analog Dialogue*, v. 49, n. 1, 2015. Disponível em: <<https://www.analog.com/media/en/analog-dialogue/volume-49/number-1/articles/quickly-implement-JESD204B.pdf>>. Acesso em: 06 ago. 2024. Citado na página 19.

GROUP ANALOG DEVICES, I. A. E. *Interfacing FPGAs to an ADC Converter's Digital Data Output*. 2014. Analog Devices JESD204B Survival Guide. Disponível em: <<https://www.analog.com/media/en/technical-documentation/technical-articles/JESD204B-Survival-Guide.pdf>>. Acesso em: 08 ago. 2024. Citado na página 19.

Grupo Zilocchi. *Tektronix AFG1022- Gerador de Função Arbitrária de 25MHz, 02 canais*. 2024. Disponível em: <https://www.zilocchi-loja.com.br/tektronix-afg1022-gerador-de-funcao-arbitraria-de-25mhz-02-canais-5-anos-de-garantia?srsltid=AfmBOoqlkwKzeKphiqGGR7IpnC_G2kGlT5hr71y8aoOVguNjF7qBzc0>. Acesso em: 21 nov. 2024. Citado na página 64.

HARRIS, J. *What Is JESD204 and Why Should We Pay Attention to It?* 2013. Analog Devices JESD204B Survival Guide. Disponível em: <<https://www.analog.com/media/en/technical-documentation/technical-articles/JESD204B-Survival-Guide.pdf>>. Acesso em: 05 ago. 2024. Citado 4 vezes nas páginas 19, 23, 24 e 25.

HIGUERUELA, A. P. *Design of high-speed data acquisition systems based on the JESD204 interface with Xilinx ZynqMP devices for MicroTCA platforms*. 2020. 82 f. Grado en Ingeniería Electrónica de Comunicaciones, 2023. Citado na página 27.

iWave. *iWave iW-Rainbow-G30D Hardware User's Manual*. 2022. Disponível em: <<https://www.manualslib.com/manual/2623009/Iwave-Iw-Rainbow-G30d.html#manual>>. Acesso em: 21 nov. 2024. Citado 2 vezes nas páginas 64 e 65.

KRUPA, G. P. *Sistema embarcado com arquitetura ZYNQ, firmware e software, para controle de sistema ultrassônico*. 2020. 84 f. Monografia (Trabalho de Conclusão de Curso) - Engenharia da Computação, Universidade Federal de Itajubá, Itajubá, 2020. Citado na página 19.

Minipa. *PROPOSTA TÉCNICA OSCILOSCOPIO DIGITAL SÉRIE: MO-3104*. 2024. Disponível em: <http://www.minipa.com.br/images/proposta_tecnica/MO-3104-1300-BR.pdf>. Acesso em: 21 nov. 2024. Citado na página 66.

ZHANG, F. *High-speed Serial Buses in Embedded Systems*. Springer, 2020. Disponível em: <<https://link.springer.com/book/10.1007/978-981-15-1868-3>>. Citado 2 vezes nas páginas 23 e 24.