Universidade de Brasília

Faculdade do Gama

Implementação de Multiplexador ISDB-TB em SoC Cyclone V para aplicações em TV Digital

Victor Orfeu Merlo

TRABALHO DE CONCLUSÃO DE CURSO

ENGENHARIA ELETRÔNICA

Brasília

2025

Universidade de Brasília

Faculdade do Gama

Implementação de Multiplexador ISDB-TB em SoC Cyclone V para aplicações em TV Digital

Victor Orfeu Merlo

Trabalho de Conclusão de Curso submetido como requisito parcial para obtenção do bacharel de Engenheiro Eletrônico.

Orientador: Prof. Dr. Daniel Mauricio Muñoz Arboleda

Brasília

2025

FICHA CATALOGRÁFICA

Orfeu Merlo, Victor.

Implementação de Multiplexador ISDB-TB em SoC Cyclone V para aplicações em TV Digital / Victor Orfeu Merlo; orientador Daniel Mauricio Muñoz Arboleda. -- Brasília, 2025.

57 p.

Trabalho de Conclusão de Curso (Engenharia Eletrônica) -- Universidade de Brasília, 2025.

1. FPGA. 2. ISDB-TB. 3. Multiplexador. 4. Cyclone V. 5. HPS.I. Mauricio Muñoz Arboleda, Daniel, orient. II. Título.

Universidade de Brasília
Faculdade do Gama

Implementação de Multiplexador ISDB-TB em SoC Cyclone V para aplicações em TV Digital

Victor Orfeu Merlo

Trabalho de Conclusão de Curso submetido como requisito parcial para obtenção do bacharel de Engenheiro Eletrônico.

Prof. Dr. Daniel Mauricio Muñoz Arboleda,
UnB/FGA
Orientador

Prof. Dr. Daniel Costa Araújo,
UnB/FGA
Examinador interno

Prof. Dr. Sandro Augusto Pavlik Haddad,
UnB/FGA
Examinador interno

“Muitos bilhões de trilhões de toneladas de núcleos de hidrogênio superaquecidos explodindo se levantaram aos poucos sobre o horizonte e conseguiram parecer pequenos, frios e ligeiramente úmidos.
Há um momento em cada amanhecer no qual a luz parece flutuar e tudo parece mágico. A criação prende a respiração.”
(Douglas Adams)

Resumo

O padrão de TV Digital ISDB-TB foi desenvolvido e adotado no Brasil a partir do padrão japonês ISDB-T. A norma que descreve esse protocolo indica que os transmissores desse tipo de sinal devem unir múltiplos *Transport Streams* (TS) em um único sinal chamado *Broadcast Transport Stream* (BTS), que é efetivamente modulado e transmitido. O Multiplexador é o aparelho responsável por essa etapa do protocolo. Uma empresa nacional que desenvolve e produz esse tipo de equipamento, possui um Multiplexador implementado em uma FPGA Stratix II, da Intel, porém essa família de dispositivos está atualmente obsoleta. Dito isso, este trabalho visa trazer o Multiplexador para um *System-on-Chip* (SoC) Cyclone V, uma família de dispositivos mais recente e ainda com suporte da Intel, adicionando interfaces de comunicação com um *soft processor* Nios e o processador ARM do SoC. Isso é feito analisando-se o projeto do Multiplexador da empresa, para entender quais as conexões necessárias desse componente, e implementando gradativamente cada recurso, com o objetivo final de ter um Multiplexador que consiga se comunicar com qualquer processador através de interfaces mapeadas em memória, usando o protocolo Avalon da Intel. Verificou-se que o projeto pode ser implementado em Cyclone V, o dispositivo possui recursos suficientes para ser configurado com o Multiplexador e vários componentes aderentes. A interface Avalon é efetiva para realizar a comunicação entre o processador e o Multiplexador, além de simplificar as conexões a serem feitas quando ele for inserido em projetos diferentes. Também foi possível verificar que os demais módulos do sistema foram implementados corretamente, usando um equipamento da própria empresa, preparado para testes usando uma configuração simplificada, e um adaptador que envia um TS para o equipamento.

Palavras-chave: FPGA. ISDB-TB. Multiplexador. Cyclone V. HPS.

Abstract

The Digital TV standard ISDB-TB was developed and adopted in Brazil from the japanese standard ISDB-T. The norm that describes this protocol denotes that transmiters of this kind of signal must unite many Transport Streams (TS) in one single Broadcast Transport Stream (BTS), which is the signal actually modulated and then transmited. The Multiplexer is the equipment responsible por this stage of the protocol. A national company that develops and manufactures this kind of equipment, has a Multiplexer implemented in a Stratix II FPGA, from Intel, but this family of devices is currently obsolete. That said, this work aims to bring the Multiplexer to a System-on-Chip (SoC) Cyclone V, a more recent and still supported by Intel family of devices, adding communication interfaces with a Nios soft-processor and the SoC's ARM processor. This is done by by analysing the company's Multiplexer project, to understand what are the needed connections and grdually implementing each feature, with the final goal of having a Multiplexer that is able to communicate with any processor through Memmory Mapped interfaces, using Intel's Avalon protocol. It was verified that the design can be implemented in a Cyclone V, as the device contains enough resources to be configured with the Multiplexer and many other components. The Avalon interface can be used to efectively communicate the processor with the Multiplexer, as well as simplifying connections to be done when it is inserted into different projects. I was also possible to verify that other system modules where correctly implemented, using a national company's own equipment, prepared for tests with a simplified configuration, and a adapter that sends a TS to the equipment.

Keywords: FPGA. ISDB-TB. Multiplexer. Cyclone V. HPS.

Lista de figuras

Figura 2.1 Visão geral do sistema de transmissão ISDB-TB. 16
Figura 2.2 Diagrama de blocos dos componentes do sistema de transmissão ISDB-TB. 17
Figura 2.3 Diagrama simplificado de um ALM de uma Cyclone V, da Intel. 18
Figura 2.4 Diagrama de uma arquitetura genérica de uma FPGA. 18
Figura 2.5 Diagrama da arquitetura de uma FPGA da família Cyclone V SE. 19
Figura 2.6 Exemplo dos passos executados por uma ferramenta de síntese. 19
Figura 2.7 Diagrama do HPS em uma FPGA Cyclone V da Intel. 21
Figura 2.8 Exemplo de uma conexão entre Processador/Microcontrolador e Componentes na FPGA, através do protocolo Avalon-MM. 22
Figura 2.9 Exemplo de um sistema usando interfaces Avalon para conectar diversos componentes, dentro e fora da FPGA. 23
Figura 2.10 Exemplo da janela do Platform Designer com suas principais funcionalidades. 24
Figura 2.11 Diagrama de blocos simplificado do Signal Tap na FPGA. 24
Figura 2.12 Exemplo de janela do Signal Tap com suas principais funcionalidades. . 25
Figura 3.1 Placa de desenvolvimento SoCKit, da Arrow. 27
Figura 3.2 Adaptador de USB para ASI DTU-245, da DekTec. 29
Figura 3.3 Diagrama da implementação atual do Multiplexador. 30
Figura 3.4 Esquemático RTL (parte 1) do projeto do Multiplexador implementado em Stratix II. 32
Figura 3.5 Esquemático RTL (parte 2) do projeto do Multiplexador implementado em Stratix II. 33
Figura 3.6 Esquemático RTL (parte 3) do projeto do Multiplexador implementado em Stratix II. 34
Figura 3.7 Diagrama da arquitetura externa proposta para atualização do Multiplexador. 35
Figura 3.8 Diagrama da arquitetura interna proposta para atualização do Multiplexador. 36
Figura 4.1 Esquemático RTL (parte 1) do projeto do Multiplexador implementado em Cyclone V. 40
Figura 4.2 Esquemático RTL (parte 2) do projeto do Multiplexador implementado em Cyclone V. 41
Figura 4.3 Esquemático RTL (parte 3) do projeto do Multiplexador implementado em Cyclone V. 42

Figura 4.4 Esquemático RTL do projeto para validar encapsulamento do Multiplexador com Avalon-MM. 44
Figura 4.5 Visualização do consumo de recursos do projeto no chip Cyclone V. . . 45
Figura 4.6 Bancada no laboratório da universidade contendo computador e placa de desenvolvimento SoCkit. 46
Figura 4.7 Saída do programa em C, no terminal vindo do HPS, no primeiro teste da interface Avalon. 47
Figura 4.8 Saída do programa em C, no terminal vindo do HPS, no segundo teste da interface Avalon. 48
Figura 4.9 Bancada no laboratório da universidade contendo computador e aparelho da empresa. 50
Figura 4.10 Tela do programa StreamXpress mostrando os parâmetros identificados no TS sendo enviado ao aparelho da empresa. 51
Figura 4.11 Signal Tap mostrando os sinais internos do Multiplexador continuamente, nas primeiras 2 etapas escolhidas. 52
Figura 4.12 Signal Tap mostrando os sinais internos do Multiplexador continuamente, na última etapa escolhida. 53

Lista de tabelas

Tabela 3.1 Especificações da placa de desenvolvimento SoCKit, da Arrow. 28
Tabela 3.2 Consumo de recursos do Multiplexador implementado em Stratix II. . . . 31
Tabela 3.3 Sumário das violações de temporização no projeto implementado em Stratix II. Valores em ns. 31
Tabela 4.1 Consumo de recursos do Multiplexador implementado em Cyclone V. . . 39
Tabela 4.2 Sumário das violações de temporização no projeto em Cyclone V. Valores em ns. 39
Tabela 4.3 Consumo de recursos do Multiplexador encapsulado implementado em Cyclone V no Arrow SoCkit. 44

Lista de abreviaturas e siglas

ALM	*Adaptive Logic Block*
ASI	*Assynchronous Serial Interface*
ASIC	*Application Specific Integrated Circuit*
BTS	*Broadcast Transport Stream*
CLB	*Configurable Logic Block*
DSP	*Digital Signal Processor*
DVB	*Digital Video Broadcasting*
F2H	Ponte *FPGA-to-HPS*
FPGA	*Field Programmable Gate Array*
H2F	Ponte *HPS-to-FPGA*
HDL	*Hardware Description Language*
HPS	*Hard Processor System*
HSMC	*High Speed Mezzanine Card*
IP	*Intellectual Property*
ISDB-TB	Serviço Integrado de Transmissão Digital Terrestre Brasileiro
JTAG	*Joint Test Action Group*
LED	*Light Emiting Diode*
LWH2F	Ponte *Lightweight HPS-to-FPGA*
PAR	*Place and Route*
PIO	*Parallel Input/Ouput*
PLL	*Phase-Locked Loops*
RAM	*Random Access Memory*
RTC	*Real Time Clock*
RTL	*Register-Transfer Level*
SNR	*Signal-to-Noise Ratio*
SO	Sistema Operacional
SoC	*System-on-Chip*
SPI	*Serial Peripheral Interface*
SSH	Protocolo *Secure Shell*
TS	*Transport Stream*

Sumário

1 Introdução 14
1.1 Perguntas de Pesquisa 15
1.2 Objetivos 15
1.2.1 Objetivo Geral 15
1.2.2 Objetivos Específicos 15
1.3 Organização do Documento 15

2 Fundamentação Teorica 16
2.1 Protoloco ISDB-TB 16
2.2 Hardware Reconfigurável 17
2.2.1 FPGA 17
2.2.2 Fluxo de projeto 17
2.2.3 *Soft processors* e Nios 20
2.2.4 SoCs e HPS 20
2.3 Ferramentas específicas da Intel 20
2.3.1 Pontes entre HPS e FPGA 20
2.3.2 Interface Avalon 21
2.3.3 Platform Designer 22
2.3.4 Analisador Lógico Signal Tap 23
2.4 Trabalhos Correlatos 25

3 Metodologia Proposta 27
3.1 Materiais e Ferramentas 27
3.1.1 Materiais da empresa nacional 27
3.2 Arquitetura Atual do Multiplexador 29
3.2.1 Implementação em Stratix II 29
3.3 Arquitetura Proposta para a Atualização do Multiplexador 35
3.3.1 Remapeamento para Cyclone V 36
3.3.2 Atualização para Quartus Prime 36
3.3.3 Encapsulamento do Multiplexador com Avalon-MM 37
3.4 Planejamento Experimental 37
3.4.1 Caracterização dos circuitos 37
3.4.2 Teste do encapsulamento Avalon 37
3.4.3 Teste no equipamento da empresa 37

4 Resultados 38
4.1 Implementação do Multiplexador atual em Cyclone V 38

4.2 Encapsulamento do Multiplexador com Avalon-MM 43
4.2.1 Verificação da implementação no Arrow SoCkit 45
4.3 Testes no equipamento da empresa 48
5 Conclusões 54
Referências 55

1 Introdução

Desde 2007, o Brasil utiliza o padrão ISDB-TB, que é uma variante do padrão japonês ISDB-T para transmissão terrestre de sinais de TV Digital. Ele pôde ser adaptado ao cenário nacional pois foi desenvolvido com base em quatro principais requisitos que também se aplicam ao nosso país (Digital Broadcastig Experts Group, 2007):

a) Flexibilidade de transmissão, podendo transmitir sinais de diferentes resoluções (HDTV e SDTV), separados ou juntos, ou transmitir várias programações simultaneamente (chamado multi-programação);

b) Robustez de recepção, possuindo mecanismos para diminuir atenuações causadas por múltiplos caminhos, perdas de percurso, excesso de ruído no canal e receptor em movimento;

c) Mobilidade, sendo capaz de transmitir sinais para dispositivos móveis (chamado de sinal 1-seg);

d) Eficiência de uso do espectro de frequência, para lidar com ambientes urbanos poluídos de outras transmissões em frequências mais altas, que podem gerar interferências.

Esse padrão pode ser implementado com diversos equipamentos, sendo um deles o Multiplexador de Sinais, responsável por receber os vários fluxos de sinais de TV e uni-los em um único sinal que é efetivamente transmitido. Uma empresa nacional, que fez parte da implementação do padrão ISDB-TB, desenvolveu tal equipamento utilizando uma placa eletrônica própria com uma *Field Programmable Gate Array* (FPGA), que implementa o processo de multiplexação.

A placa também contém o *soft-processor* Nios II para controlar o Multiplexador e hospeda um servidor Web. Este providencia uma interface de usuário onde pode-se modificar as configurações e tabelas dos fluxos de transporte que estão sendo multiplexados. Esse servidor pode ser acessado por um computador convencional, conectando-o ao Multiplexador por um cabo ethernet (Ferreira, 2009).

A FPGA utilizada na placa é uma EP2S60 da família Stratix II, da Altera, agora adquirida pela Intel. Essa família de dispositivos foi lançada em 2007 e agora encontra-se descontinuada, não sendo mais fabricada pela Intel (Intel, 2021a). Com uma disponibilidade cada vez mais baixa deste chip FPGA, a empresa está procurando atualizar o *hardware* e *software* do seu multiplexador, visando também obter uma melhora no seu desempenho e usabilidade.

Pretende-se portar o projeto atual para um *System-on-Chip* (SoC) FPGA da família Cyclone V, também da Intel. Essa família de dispositivos contém um processador ARM capaz de rodar Linux. Com ele, a interface de usuário ficaria independente de um computador

externo. O controle do Multiplexador ainda se daria pelo Nios, que se comunicaria com o ARM para configurar os parâmetros do Multiplexador.

1.1 Perguntas de Pesquisa

a) É viável a implementação do Multiplexador em um SoC FPGA Cyclone V?

b) Quais as interfaces mais adequadas para realizar comunicação entre Nios, ARM e Multiplexador, nesta aplicação?

c) Qual o impacto no desempenho da estratégia de usar o Nios em conjunto com o ARM?

1.2 Objetivos

1.2.1 Objetivo Geral

Implementar o projeto do Multiplexador ISDB-TB da empresa nacional em uma plataforma mais atual baseada em SoC FPGA Cyclone V, utilizando o processador já presente no SoC para realizar a interface do usuário através do Linux.

1.2.2 Objetivos Específicos

a) Portar a solução atual de Multiplexador da empresa para o SoC Cyclone V;

b) Implementar a comunicação entre o processador e os componentes do Multiplexador através de uma interface Avalon e/ou Parallel IO;

c) Implementar a comunicação do Multiplexador com os excitadores aderentes ao padrão ISDB-TB (codificador, modulador etc).

1.3 Organização do Documento

O restante deste documento está organizado em 4 capítulos. O Capítulo 2 apresenta a fundamentação teórica, como funciona o protocolo ISDB-TB para fins desse trabalho, conceitos de *hardware* reconfigurável e pesquisas sobre atuais implementações de multiplexadores ISDB-TB. O Capítulo 3 propõe uma metodologia para a implementação desse sistema e as ferramentas a serem utilizadas. O Capítulo 4 mostra os resultados obtidos durante o desenvolvimento deste trabalho. O Capítulo 5 apresenta conclusões obtidas ao final do projeto com relação aos objetivos propostos.

2 Fundamentação Teorica

2.1 Protoloco ISDB-TB

Em 2007, o Brasil adotou para o Sistema de TV Digital o padrão ISDB (do inglês *Integrated Services Digital Broadcasting*). Desenvolvido no Japão pela *Association of Radio Industries and Businesses* (ARIB), esse padrão possui diferentes versões para transmissão por satélite (ISDB-S), por cabo (ISDB-C) e demais formas terrestres (ISDB-T).

Atualmente, no Brasil, é utilizada a versão para transmissão terrestre brasileira desse padrão, chamada ISDB-TB (ABNT, 2020). A norma define que um transmissor pode receber vários fluxos de sinais, chamados *Transport Streams* (TS), juntamente com dados referentes a eles, chamados de Tabelas. Eles devem ser combinados em um único TS chamado *Broadcast Transport Stream* (BTS), para serem modulados e transmitidos. Esses sinais são usados no formato *Assynchronous Serial Interface* (ASI). Um diagrama do transmissor é mostrado na Figura 2.1.

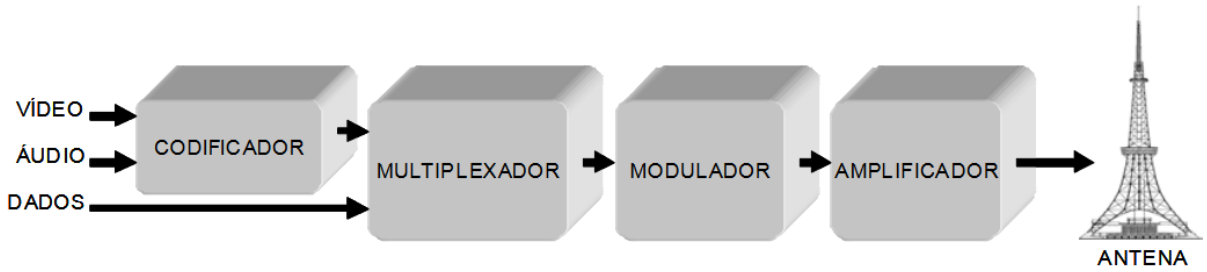


Figura 2.1 – Visão geral do sistema de transmissão ISDB-TB.
Fonte: Retirado de (ABNT, 2020).

Para cumprir os requisitos de transmitir sinais diferentes, para diversos tipos de receptores e com robustez a interferências, foi desenvolvido um esquema de Segmentação da Banda de Transmissão usando a modulação OFDM (BST-OFDM, sigla em inglês). Essa técnica permite multiplexar densamente várias portadoras, de forma que o sinal seja composto de camadas hierárquicas, transmitidas simultaneamente, que não causam interferências entre si. (Uehara; Takada; Kuroda, 1999).

O componente desse sistema chamado Multiplexador é responsável tanto por realizar a primeira junção de vários TSs em um único BTS, que é então enviado ao Modulador para se tornar o sinal transmitido, quanto por configurar os parâmetros de transmissão do seu BTS. Na Figura 2.2 é mostrado que pode haver mais de um BTS que chega ao Modulador, e ele precisa realizar uma outra multiplexação para encaminhar ao transmissor o sinal OFDM adequado (ABNT, 2020).

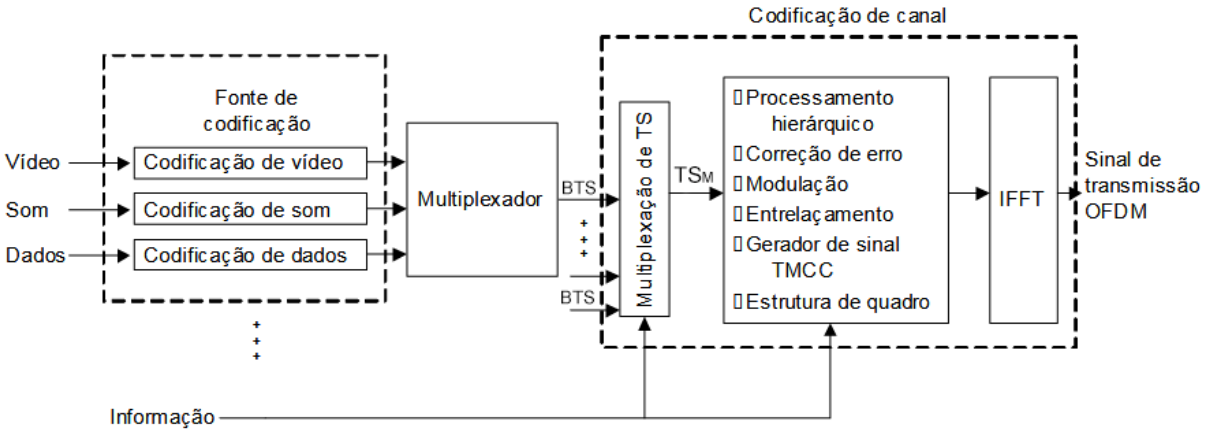


Figura 2.2 – Diagrama de blocos dos componentes do sistema de transmissão ISDB-TB.
Fonte: Adaptado de (ABNT, 2020).

2.2 Hardware Reconfigurável

2.2.1 FPGA

Uma *Field Programmable Gate Array* (FPGA) é um tipo de circuito integrado capaz de ser reprogramado diversas vezes após sua fabricação. Ela consiste em um conjunto de *Configurable Logic Blocks* (CLBs), que são blocos configuráveis capazes de implementar operações lógicas. Um exemplo de um CLB é mostrado na Figura 2.3. Esses blocos são conectados entre si por uma matriz de interconexões, também configuráveis. Juntos como mostrado na Figura 2.4, esses elementos básicos permitem a criação de configurações que realizam inúmeras operações de caráter digital.

Esses componentes possuem diferentes implementações específicas de cada empresa fabricante e de cada família de dispositivos dentro da empresa. Dado que o trabalho será implementado em dispositivo da Intel, os conceitos apresentados serão direcionados para implementações dessa empresa. Por exemplo, a Intel chama os CLBs das suas FPGAs de ALMs (*Adaptive Logic Modules*).

Complementando esses componentes, FPGAs costumam ter outros também programáveis mas com funções específicas, como blocos de memória RAM, *Digital Signal Processors* (DSPs), *Phase-Locked Loops* (PLLs) e interfaces de comunicações de alta velocidade. Algumas FPGAs possuem componentes que a caracterizam como um *System-on-Chip* (SoC), como um microcontrolador ou processador. Na Figura 2.5, pode-se ver esses componentes em uma FPGA da família Cyclone V, da Intel, juntos compondo um SoC versátil e poderoso.

2.2.2 Fluxo de projeto

Projetos em FPGA seguem um fluxo dividido em três etapas básicas: codificação, síntese e implementação. Os detalhes específicos de cada etapa mudam de acordo com a

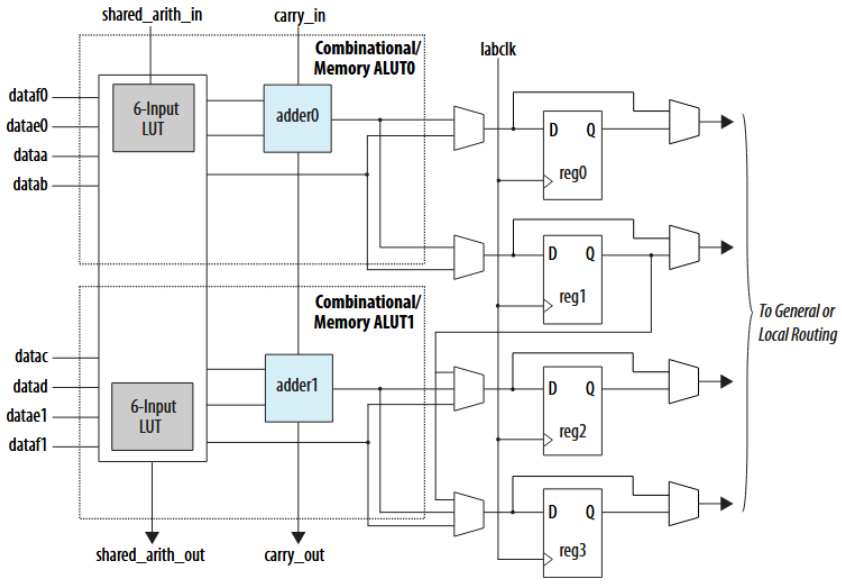


Figura 2.3 – Diagrama simplificado de um ALM de uma Cyclone V, da Intel.
Fonte: Retirado de (Altera, 2023).

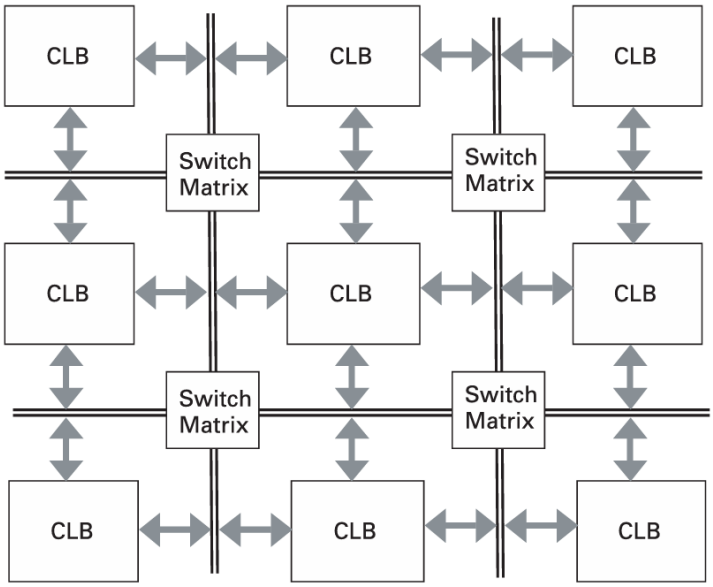


Figura 2.4 – Diagrama de uma arquitetura genérica de uma FPGA.
Fonte: Retirado de (Pedroni, 2004).

ferramenta usada, mas as ideias são similares. Cada fabricante desenvolve a ferramenta de síntese para ser usada com suas FPGAs. Da AMD tem-se o Vivado, da Intel tem-se o Quartus.

Na codificação, o projetista descreve o circuito desejado usando uma linguagem de descrição de *hardware* (HDL, sigla em inglês), geralmente VHDL ou Verilog. A partir dessa descrição, na etapa de síntese, a ferramenta infere as equações lógicas e gera um esquemático, com ligações entre os componentes necessários para realizar o circuito, independente da tecnologia. Como muitos designs envolvem registradores, diz-se que é a etapa de síntese a Nível de Transferência de Registradores (RTL, sigla em inglês). Em seguida, na implementação, a ferramenta posiciona esses componentes no *chip* escolhido e os conecta através da

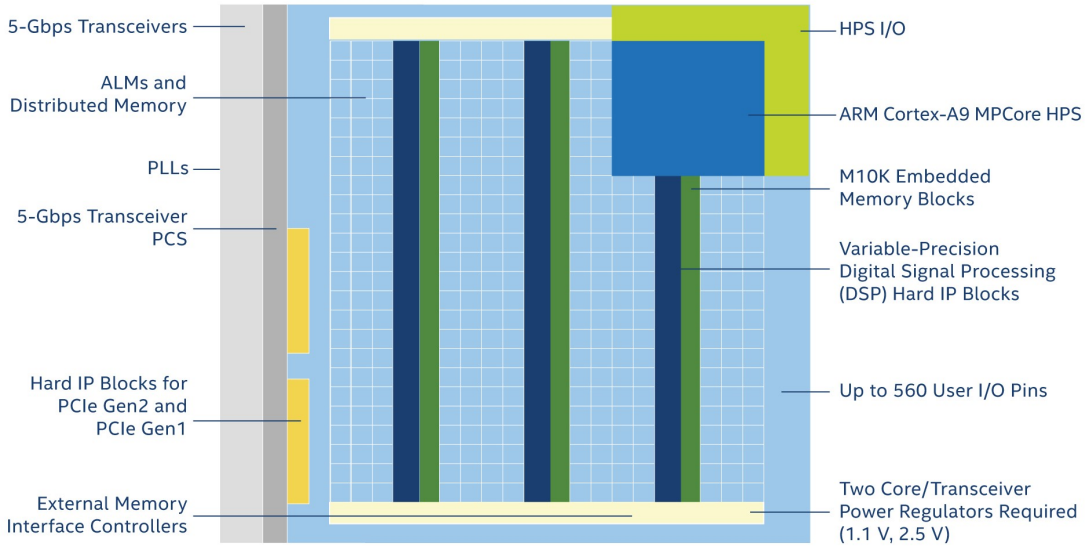


Figura 2.5 – Diagrama da arquitetura de uma FPGA da família Cyclone V SE.
Fonte: Retirado de (Intel, s.d.a).

matriz de interconexões, sendo essa etapa chamada de *Place and Route* (PAR). Na Figura 2.6 pode-se ver um exemplo simples desse processo.

Durante todo o processo, o projetista pode executar simulações do seu projeto para verificar se cumpre as funções desejadas. Automaticamente, a ferramenta de síntese também verifica se o circuito descrito é sintetizável e cumpre as restrições de temporização, tanto do próprio dispositivo como as que o usuário configurar no projeto. Com isso, na etapa de implementação, a ferramenta pode otimizar o posicionamento para diminuir consumo de recursos ou aumentar o desempenho do circuito, dois parâmetros geralmente conflitantes.

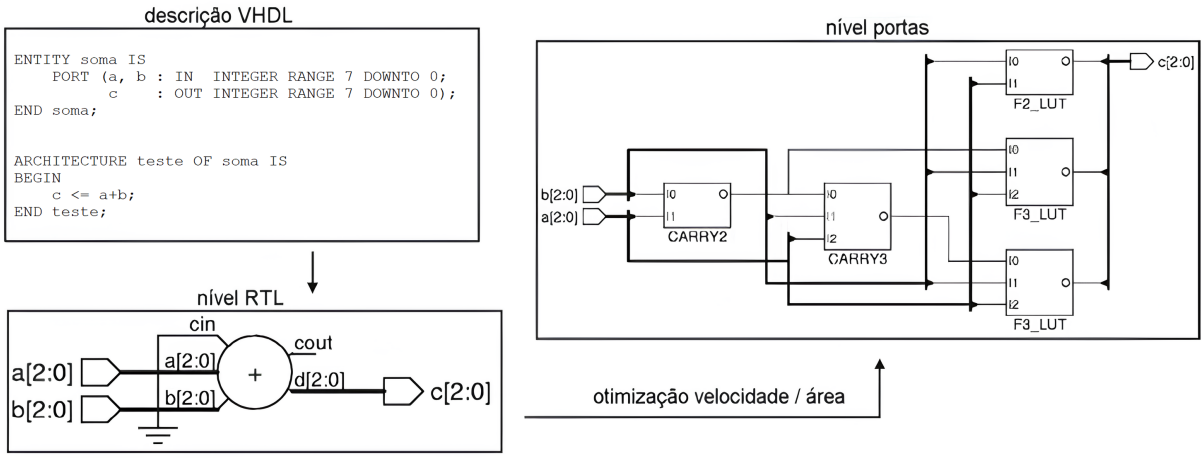


Figura 2.6 – Exemplo dos passos executados por uma ferramenta de síntese.
Fonte: Retirado de (D'Amore, 2012).

2.2.3 *Soft processors* e Nios

Os componentes de uma FPGA podem ser usados para implementar um processador, que é apenas um grande conjunto de portas lógicas e registradores do ponto de vista da FPGA. Processadores implementados dessa forma são chamados de *soft processors*, uma vez que eles podem ser personalizados para uma determinada aplicação.

A Intel disponibiliza a família de processadores Nios, que podem ser implementados em suas FPGAs. Esses *soft processors* são equivalentes a um microcontrolador, possuindo um processador e diversos periféricos e memória embarcados no mesmo *chip*. Esse microcontrolador é programado com o *toolchain* GNU C/C++ e a Intel também disponibiliza um ambiente de desenvolvimento para tal (Intel, 2023).

2.2.4 SoCs e HPS

Para projetos que precisem de ambientes de desenvolvimento mais tradicionais e acessíveis, como para um computador, e/ou performance garantida de ser melhor que um *soft processor*, existem FPGAs com esses dispositivos implementados em hardware. Eles não possuem parâmetros configuráveis como os *soft processors*, em contrapartida são otimizados para entregar maiores velocidades de processamento. Dessa forma, tem-se disponível um sistema completo em um único encapsulamento, denominado então de *System-on-Chip* (SoC).

A Intel também fabrica SoCs e disponibiliza ferramentas para desenvolver com eles. Uma das famílias de SoCs é a Cyclone V, exemplificada na Figura 2.7, que contém um processador ARM Cortex-A9, com diversos periféricos e conexões especiais com a lógica programável da FPGA. Esse sistema é denominado por eles de *Hard Processor System* (HPS) e, além da performance, é capaz de rodar uma distribuição Linux, possibilitando um desenvolvimento com softwares mais acessíveis (Intel, 2018).

2.3 Ferramentas específicas da Intel

2.3.1 Pontes entre HPS e FPGA

Para permitir a conexão entre o HPS e a FPGA do SoC, o dispositivo Cyclone V possui as chamadas pontes HPS-FPGA. Mostradas também na Figura 2.7, são três pontes: FPGA-to-HPS (F2H), HPS-to-FPGA (H2F) e Lightweight HPS-to-FPGA (LWH2F). Elas estão de acordo com as especificações do protocolo AMBA AXI 3, são unidirecionais, com direção indicada pelos nomes, e permitem a comunicação entre dispositivos mestres e escravos em cada parte do chip.

As duas primeiras pontes, F2H e H2F, são de maior desempenho, com barramentos que podem ser configurados para serem de 32, 64 ou 128 bits. A última ponte, LWH2F, possui

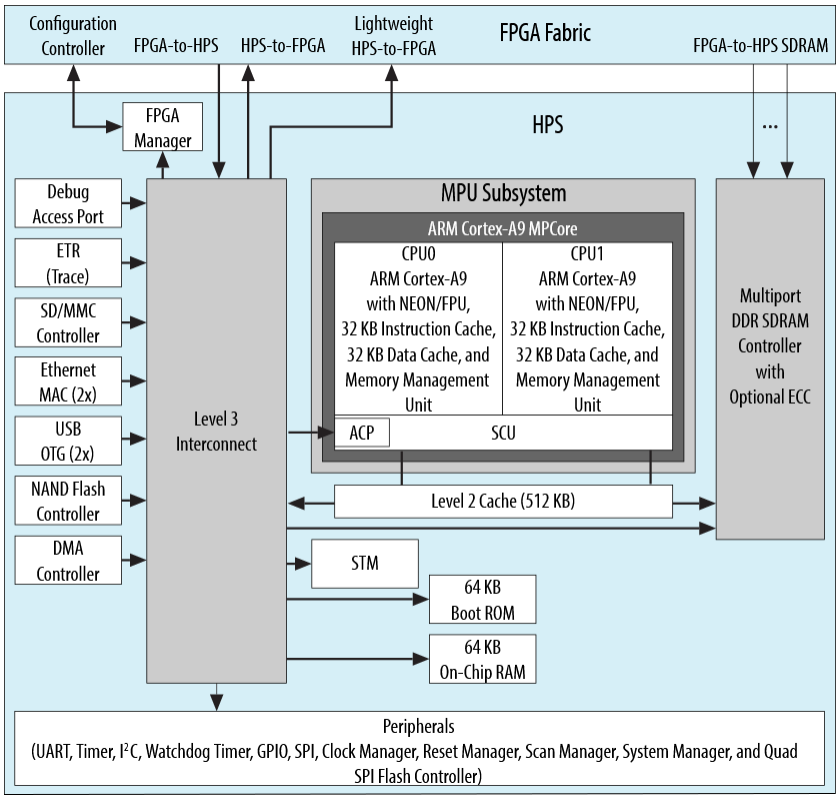


Figura 2.7 – Diagrama do HPS em uma FPGA Cyclone V da Intel.
Fonte: Retirado de (Intel, 2018).

barramento apenas de 32 bits e é recomendada para realizar configuração de registradores de controle e *status* em periféricos na FPGA (Intel, 2018).

2.3.2 Interface Avalon

Ao conectar dois componentes numa FPGA, faz-se necessário que eles usem o mesmo protocolo de comunicação. O Avalon é uma interface desenvolvida pela Altera que implementa vários protocolos de comunicação entre módulos em uma FPGA.

Um desses protocolos é denominado Avalon-Memory Mapped (Avalon-MM), que é no estilo mestre-escravo, podendo ter vários mestres e vários escravos. Cada mestre e escravo pode ser um componente que apenas lê, apenas escreve, ou ambos. Nessa interface, cada escravo recebe um endereço base único, que corresponde a um *offset* na memória acessível ao mestre. Este, por sua vez, interage com os escravos lendo ou escrevendo em um endereço a partir desse *offset* (Intel, 2022).

O Avalon implementa um outro componente chamado *Interconnect*, que realiza as transações entre mestres e escravos, assim como traduções automáticas caso haja alguma discordância entre os formatos das portas. É ele que permite a existência de vários mestres num mesmo barramento Avalon-MM. Adicionalmente, um escravo pode ter mais de uma interface Avalon, permitindo que seja acessado por vários mestres simultaneamente.

A Figura 2.8 contém um exemplo da comunicação através de Avalon-MM, onde o processador, que é o mestre nesse protocolo, "enxerga"apenas endereços de memória, e cada componente, que são os escravos, se encontra em regiões separadas dessa memória.

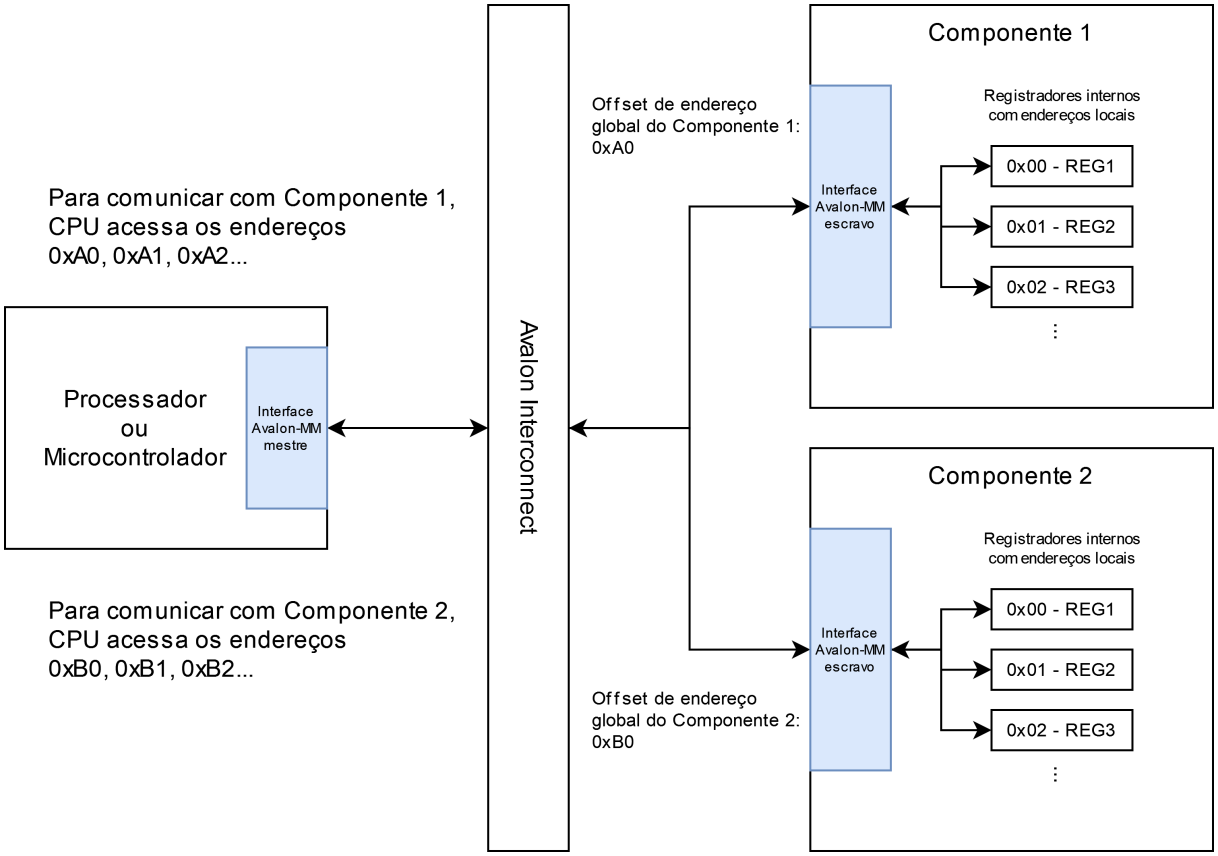


Figura 2.8 – Exemplo de uma conexão entre Processador/Microcontrolador e Componentes na FPGA, através do protocolo Avalon-MM.

A Figura 2.9 contém um exemplo de sistema que usa interfaces Avalon em diversos componentes. O *Interconnect* é representado pela seta preta com legenda "Avalon-MM", conectando os 4 mestres na parte superior com os 5 escravos na parte inferior. Essa figura também mostra as interfaces Avalon-TC, que implementa lógica *tri-state*, e Avalon-Conduit, que é uma conexão genérica direta entre módulos, podendo inclusive ser conectada a dispositivos fora da FPGA.

2.3.3 Platform Designer

Anteriormente denominado Qsys, o Platform Designer é uma ferramenta da Intel que abstrai e facilita a integração de vários sistemas para FPGAs. Exemplificado na Figura 2.10, ele contém vários recursos que ajudam o projetista a desenvolver um projeto grande mais rapidamente.

As suas principais funcionalidades são: Representação de componentes como blocos, com conexões de entradas e saídas; Gerenciamento automático dessas conexões, notificando

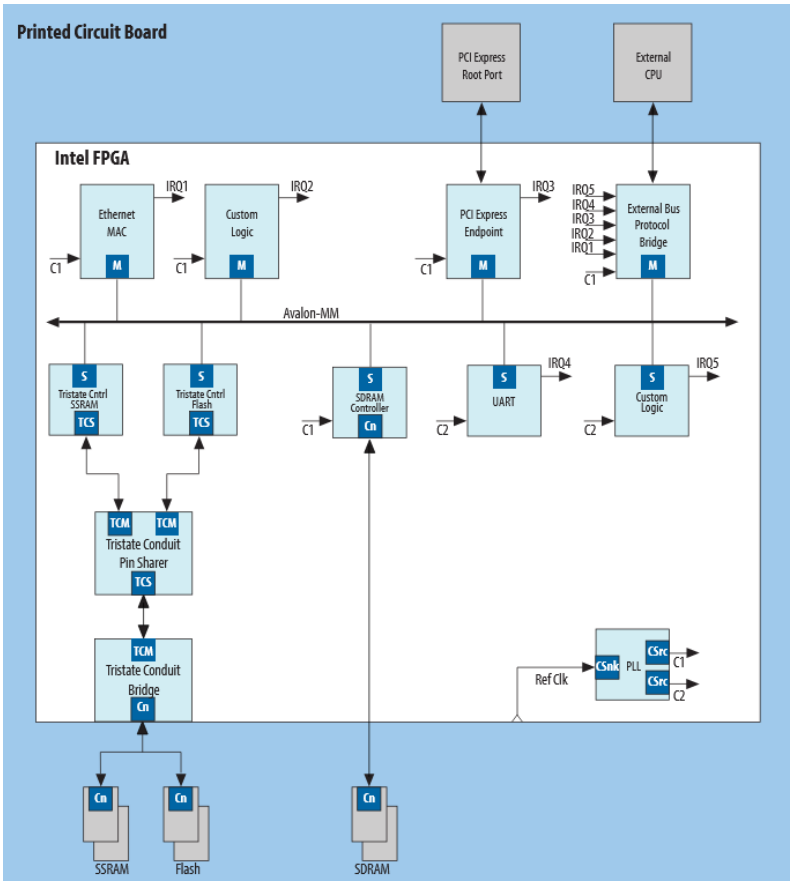


Figura 2.9 – Exemplo de um sistema usando interfaces Avalon para conectar diversos componentes, dentro e fora da FPGA.

Fonte: Retirado de (Intel, 2022).

caso haja incompatibilidades; Catálogo de Propriedades Intelectuais (IP, sigla em inglês) para inclusão no projeto; Interface gráfica para configuração desses IPs.

O catálogo de IPs inclui tanto componentes disponibilizados pela Intel, quanto componentes desenvolvidos pelo próprio projetista, que pode incluí-los nesse formato de blocos e conectá-los a outros IPs.

2.3.4 Analisador Lógico Signal Tap

O Quartus conta com a ferramenta Signal Tap, que realiza captura e mostra em tempo real sinais de dentro da FPGA, sem a necessidade de modificar o *design*. A ferramenta grava os sinais escolhidos através de sua interface gráfica e os envia para ela através do cabo usado para programar a FPGA, que geralmente usa o padrão *Joint Test Action Group* (JTAG). Esses sinais podem ser obtidos de forma contínua ou através de *triggers*, permitindo que se capture os sinais apenas em condições específicas.

A Figura 2.11 mostra um diagrama de blocos simplificado de como o Signal Tap se dispõe na FPGA quando configurado. A Figura 2.12 contém um exemplo da interface gráfica dessa ferramenta, mostrando o sinal escolhido pra ser capturado, parâmetros desse sinal e

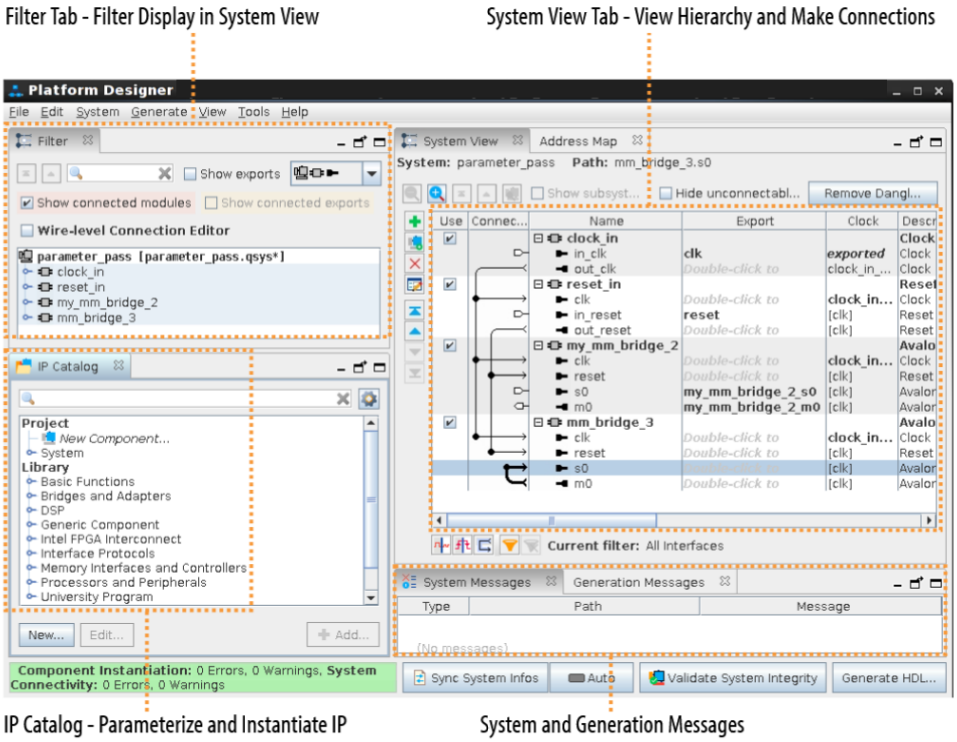


Figura 2.10 – Exemplo da janela do Platform Designer com suas principais funcionalidades.
Fonte: Retirado de (Intel, 2024b).

os *triggers* dessa captura.

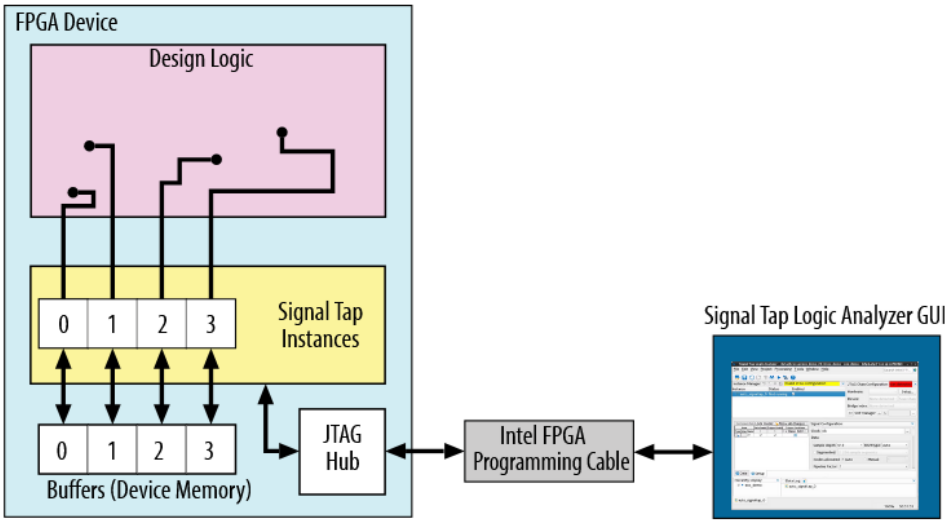


Figura 2.11 – Diagrama de blocos simplificado do Signal Tap na FPGA.
Fonte: Retirado de (Intel, 2024c).

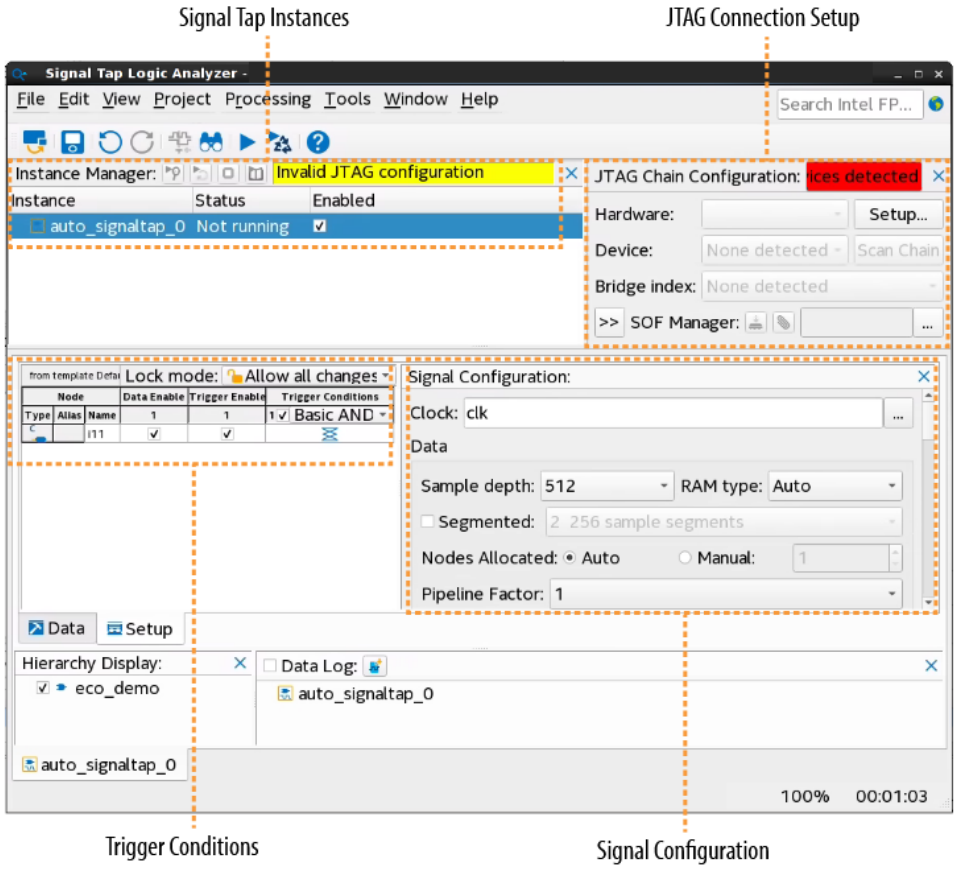


Figura 2.12 – Exemplo de janela do Signal Tap com suas principais funcionalidades.
Fonte: Retirado de (Intel, 2024c).

2.4 Trabalhos Correlatos

Existem diversas pesquisas no âmbito do protocolo ISDB-T, principalmente procurando entender aspectos de desempenho dele, para poder melhorar os algoritmos que o implementam ou desenvolver aparelhos mais otimizados.

Trabalhos como os de Nakamura *et al.* (2019) e Rabaça *et al.* (2019) analisam se o protocolo é capaz de transmitir sinais 4K ou 8K, para avaliar como o padrão poderá ser usado num futuro próximo, em que se deseja que a qualidade de vídeo sempre aumente. Ambos fazem suas pesquisas apenas em *software*.

Algumas pesquisas tentam analisar o impacto no desempenho de determinados algoritmos no padrão. Rabaça, Jerji e Akamine (2021) veem que o sinal possui maior robustez a ruído quando é usado *Layer Division Multiplexing* com 3 camadas. Oliveira, Akamine e Maciel (2016) mostram que algumas combinações de modulações e receptores resultam no aumento da Razão Sinal-Ruído (SNR, sigla em inglês) e da taxa de transmissão efetiva do padrão. Ambos também demonstram seus resultados em simulações em *software*.

Outra categoria de trabalhos é a dos que procuram transmitir outros sinais através do padrão ou mesmo modificam o sinal para fora do padrão. Valeira *et al.* (2015a) mostram que é possível converter o sinal para o formato Internet Protocol, de forma que possa ser

transmitido a um computador por cabo ethernet, mas sua implementação tornou o sinal unidirecional, sem a possibilidade do receptor enviar uma resposta ao transmissor.

Endres (2014) desenvolveu uma extensão ao programa FFMPEG, um projeto gratuito e de código aberto que contém *softwares* para lidar com áudio, vídeo e outras transmissões multimídia. Essa extensão é capaz de gerar sinais codificados no formato ISDB-T e o trabalho mostra que aparelhos receptores do padrão conseguem identificar os sinais da forma correta, tornando mais fácil a geração de sinais no padrão.

O trabalho de Akamine *et al.* (2009) demonstra que o sinal ISDB-T pode ser adaptado para o formato *Digital Video Broadcasting* (DVB), padrão muito utilizado na Europa, e vice-versa, visando aumentar o alcance do ISDB-T. Este trabalho implementa o adaptador em *software* usando C++, mas o conecta a aparelhos de transmissão e recepção de ambos padrões.

Alguns trabalhos vêm com implementações em sistemas reprogramáveis. Marchi, Cervetto e Tenorio (2011) conseguem implementar outro algoritmo de *time interleaving* usando memória DDR3 em uma placa equipada com uma FPGA Virtex 6. O algoritmo tem uma complexidade muito maior que o convencional, mas utiliza da memória de forma mais eficiente e produz os mesmos resultados, sem corromper o sinal.

Valeira *et al.* (2015b) implementaram um analisador de fluxo de transporte ISDB-T, também em FPGA, capaz de identificar erros no sinal baseados nas tabelas de dados recebidas. Esse *design* é adaptável para diversos casos, pois é possível apenas reconfigurar a FPGA para remover ou adicionar módulos de análise diferentes.

Por fim, talvez o trabalho mais completo e complexo, é o de Lima (2016), que desenvolveu um receptor ISDB-T e DVB-S2, primeiro com protótipos em FPGA e depois em Circuito Integrado de Aplicação Específica (ASIC, sigla em inglês). Seu aparelho apresenta desempenho considerado aceitável, porém não é livre de problemas, uma vez que o preço proibitivo do uso de ferramentas de *design* de ASICs fez com que certas partes do desenvolvimento fossem apressadas para diminuir esse custo.

3 Metodologia Proposta

3.1 Materiais e Ferramentas

Para testar parcialmente os *designs* desenvolvidos, foi usada a placa de desenvolvimento SoCKit, da Arrow, mostrada na Figura 3.1. Ela possui um *chip* SoC FPGA Cyclone V e diversas conexões, permitindo um bom estudo do componente e seus periféricos. A especificação detalhada encontra-se na Tabela 3.1.

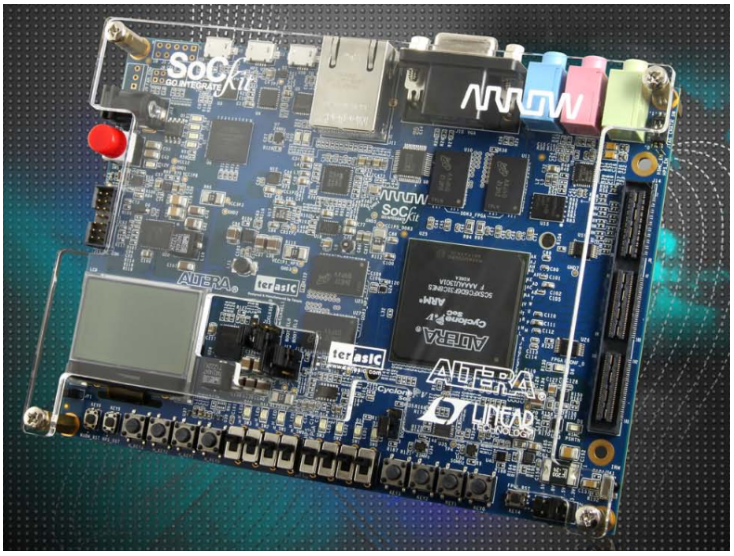


Figura 3.1 – Placa de desenvolvimento SoCKit, da Arrow.
Fonte: Adaptado de (Arrow, 2013).

Para abrir e compilar o projeto atual do Multiplexador, foi usado o *software* Quartus II 13.0sp1, da Intel, usando a avaliação gratuita de 30 dias. Para o desenvolvimento de *designs* novos, foi usado o Quartus Prime Lite 18.1, também da Intel, que possui uma licença gratuita e é a mesma versão usada pela empresa nacional.

Para documentação dos códigos HDLs e do código C para *hardware* e *software* embarcado, foi adotado o padrão de documentação Doxygen versão 1.12.0. Um repositório privado no Bitbucket foi utilizado para armazenar os códigos, documentos e relatórios gerados durante o desenvolvimento do projeto.

3.1.1 Materiais da empresa nacional

Adicionalmente, a empresa enviou alguns equipamentos para testar a implementação do Multiplexador. Esses equipamentos incluem o aparelho de transmissão de sinal de TV digital, que possui a FPGA, e um DekTec DTU-245, mostrado na Figura 3.2.

Tabela 3.1 – Especificações da placa de desenvolvimento SoCKit, da Arrow.

Categoria	**Recursos**
SoC FPGA	Cyclone V SoC 5CSXFC6D6F31
	Dual-core ARM Cortex-A9 (HPS)
	110K elementos lógicos programáveis
	5,140 Kbits de memória embarcada
	6x PLLs fracionárias
	2x controladores de memória em hardware
	Transceptores 3.125G
Configuração e Debug	EPCQ256 com On-board USB Blaster II, conector USB tipo B
Memória	1GB (2x256MBx16) DDR3 SDRAM conectada à FPGA
	1GB (2x256MBx16) DDR3 SDRAM conectada ao HPS
	128MB QSPI Flash
	1x Slot para cartão Micro SD
Comunicação	1x USB 2.0 OTG, conector USB tipo AB
	1x UART-to-USB, conector micro USB tipo B
	1x Conector Ethernet 10/100/1000
Conectores	1x conector HSMC
	1x conector LTC
Display	1x DAC VGA 24-bit
	1x módulo LCD 128x64 com luz de fundo
Audio	1x CODEC 24-bit
	Conectores line-in, line-out e microphone-in
Botões e indicadores	8x botões (4x na FPGA, 4x no HPS)
	8x *switches* (4x na FPGA, 4x no HPS)
	8x LEDs (4x na FPGA, 4x no HPS)
	2x botões de *reset* do HPS
Sensores	1x Acelerômetro
	1x sensor de temperatura
Alimentação	1x Entrada 12V DC

Fonte: Adaptado de (Arrow, 2013).

O aparelho da empresa tem uma função similar à placa de desenvolvimento SoCKit, onde se tem um ambiente pronto para testarmos o projeto do Multiplexador, com *hardware* e periféricos para alimentar a FPGA, processar os sinais ASI de entrada e de saída, entre outras funções.

O DTU-245 é um adaptador de USB, que possui uma entrada ASI e uma saída ASI. Através dele, pode-se tanto enviar quanto receber um TS arbitrário (DekTec, s.d.). Com ele, é possível testar o funcionamento básico do Multiplexador no aparelho da empresa usando



Figura 3.2 – Adaptador de USB para ASI DTU-245, da DekTec.
Fonte: Retirado de (DekTec, s.d.).

sinais reais.

3.2 Arquitetura Atual do Multiplexador

Na Figura 3.3 pode-se ver um diagrama da implementação atual do Multiplexador. A FPGA é do modelo EP2S60F672, da família Stratix II. O Processador é o Nios II, embarcado na FPGA em VHDL. O Sistema Operacional e o Servidor Web utilizados são uClinux e *Basic Object Adapter*, respectivamente. Estes foram escolhidos por serem gratuitos e desenvolvidos para sistemas embarcados (Ferreira, 2009).

Os vários possíveis TSs chegam à FPGA pelas 8 entradas ASI, e o BTS confeccionado sai por uma saída também ASI. A conexão do cabo Ethernet com o computador externo permite realizar configuração e monitoramento do Multiplexador e dos fluxos que passam por ele.

Ligados às portas ASI na placa do equipamento da empresa, tanto na entrada como na saída, estão implementados circuitos que realizam essa conversão do domínio analógico para o digital. Esses circuitos operam com um sinal de *clock* que vai também para a FPGA do Multiplexador, permitindo que ambos estejam sincronizados e funcionem como esperado.

3.2.1 Implementação em Stratix II

Com acesso aos arquivos do projeto do Multiplexador, usando o *software* Quartus II versão 13.0sp1, da Intel, é possível observar a implementação na Stratix II. O projeto passa com sucesso pelas etapas de Análise, Síntese e PAR. Apesar de não apresentar erros na

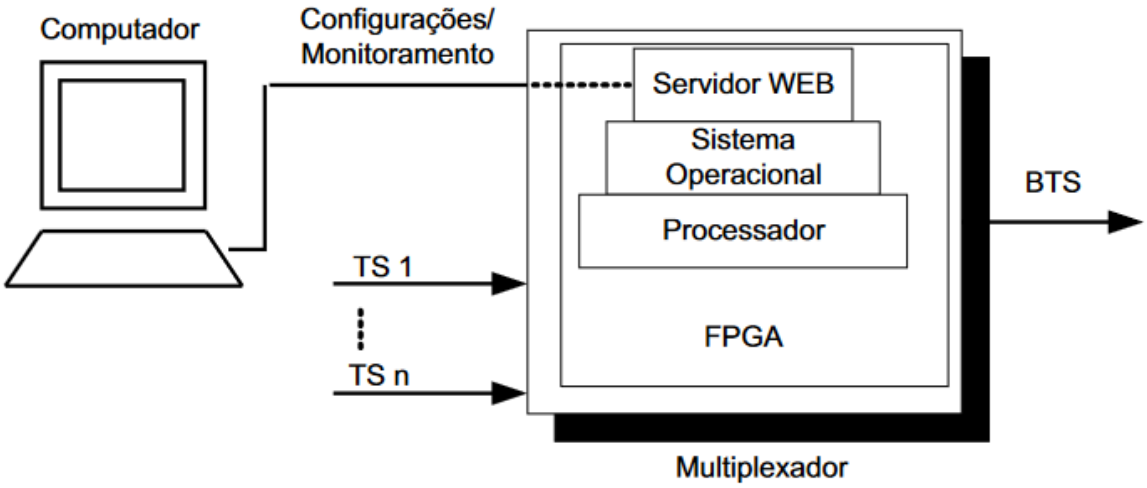


Figura 3.3 – Diagrama da implementação atual do Multiplexador.
Fonte: Retirado de (Ferreira, 2009).

etapa de Análise de Temporização, o Quartus indica algumas violações nas restrições de temporização relacionadas a *clocks* das entradas ASI.

Nas Figuras 3.4, 3.5 e 3.6, pode-se ver o esquemático RTL do projeto implementado em Stratix II. Dada a complexidade e grande quantidade de componentes do *design*, o esquemático é grande e o Quartus não permite reposicionamento dos componentes para facilitar visualização, então ele foi divido em 3 figuras. Algumas funções de cada componente descritas são suposições baseadas nos nomes e nas suas conexões, pois o projeto como um todo carece de documentação.

O módulo com muitas conexões chamado `TOP_MuxISDB_CoreMux8AsiNios_Tablev2` (Figura 3.6 a)) é o que implementa de fato o Multiplexador. O arquivo correspondente a ele tem a extensão `.bdf`, que é o tipo de arquivo do Quartus para *designs* feitos na ferramenta de diagrama de blocos (Intel, 2017). O módulo `NiosProc_MuxSatV100` (Figura 3.5 a)) é o sistema feito no Platform Designer que contém o Nios e seus periféricos de interface para conectar-se com os demais componentes do projeto.

Pode-se ver que o sistema contém diversas funcionalidades além da multiplexação, como indicações ao usuário de alarmes e contagem de tempo com um componente *Real Time Clock* (RTC). Com excessão dos IPs de *clocks* e *resets*, os demais componentes que aparecem no topo da hierarquia do RTL são IPs que relacionam os componentes do Multiplexador com componentes fora da FPGA, como os LEDs das entradas ASI e o RTC.

Na Tabela 3.2 pode-se ver o consumo de recursos do *chip* após implementação, com suas respectivas porcentagens em relação ao total disponível. Na Tabela 3.3, pode-se ver o sumário da Análise de Temporização, indicando as violações nos *clocks* das entradas ASI. Tempos negativos indicam que o sinal não cumpre com aquele requisito de temporização, estando atrasado pelo valor mostrado.

Tabela 3.2 – Consumo de recursos do Multiplexador implementado em Stratix II.

Recurso	Tipo/Quantidade
Family	Stratix II
Device	EP2S60F1020C4
Timing Models	Final
Combinational ALUTs	37.267/48.352 (77%)
Dedicated logic registers	30.913/48.352 (64%)
Total registers	30.913
Total pins	221/719 (31%)
Total virtual pins	0
Total block memory bits	1.390.436/2.544.192 (55%)
DSP block 9-bit elements	38/288 (13%)
Total PLLs	5/12 (42%)
Total DLLs	0/2 (0%)

Fonte: Adaptado do Quartus II

Tabela 3.3 – Sumário das violações de temporização no projeto implementado em Stratix II. Valores em ns.

Clock	Setup	Hold	Recovery	Removal
ip_pll_asi_1|clk[0]	-11,150	0,212	3,285	0,322
ip_pll_asi_2|clk[1]	0,543	0,202	9,380	-1,171
ip_pll_asi_2|clk[2]	1,005	0,222	2,320	-1,169

Fonte: Adaptado do Quartus II

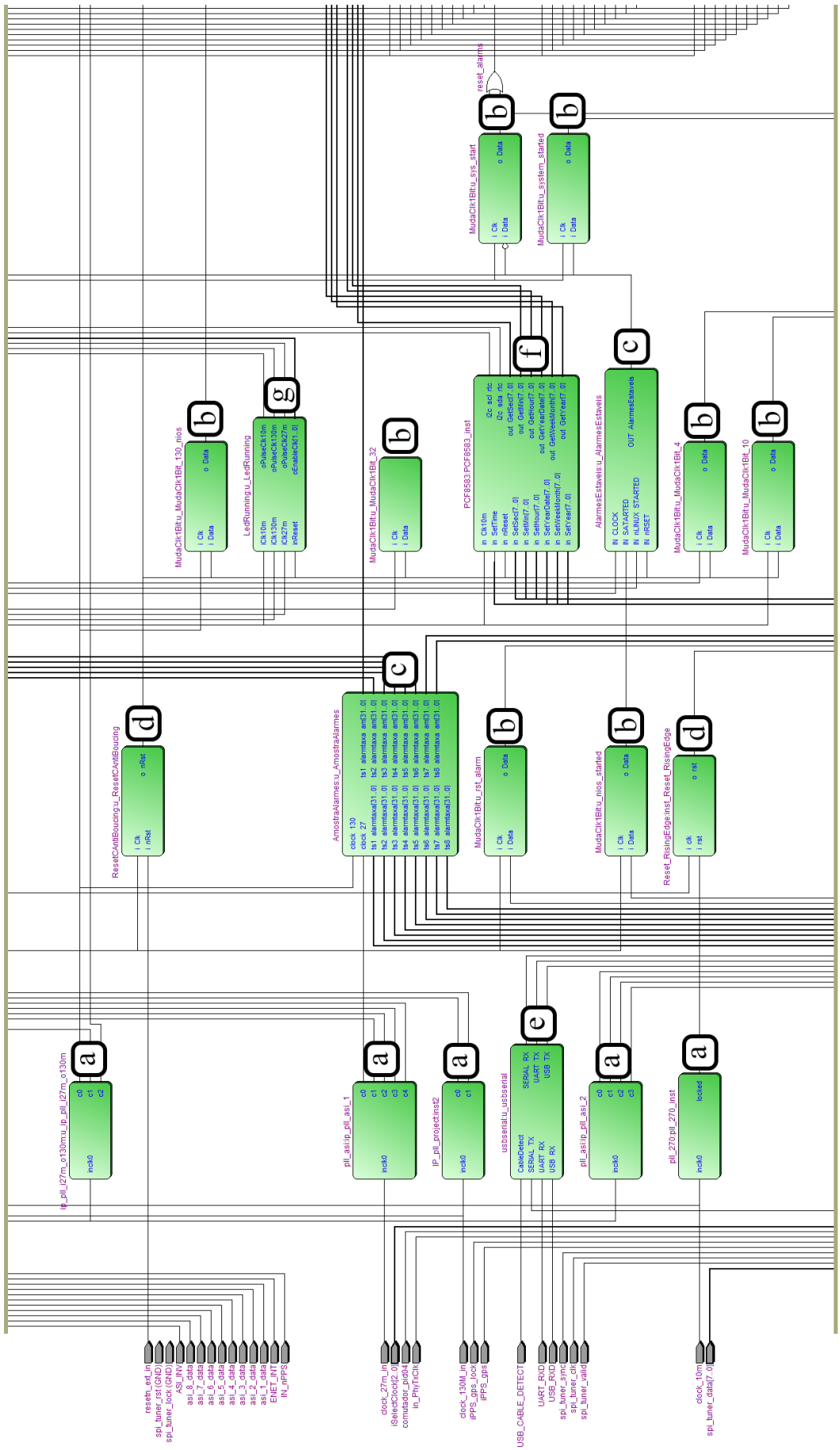


Figura 3.4 – Esquemático RTL (parte 1) do projeto do Multiplexador implementado em Stratix II. a) PLLs b) IPs de mudança de domínio de *clock* c) IPs de alarmes d) IPs de *reset* e) IP conversor de USB para Serial f) IP de comunicação com RTC fora da FPGA g) IP que controla LEDs indicadores de *clocks*.

Fonte: Adaptado do Quartus II.

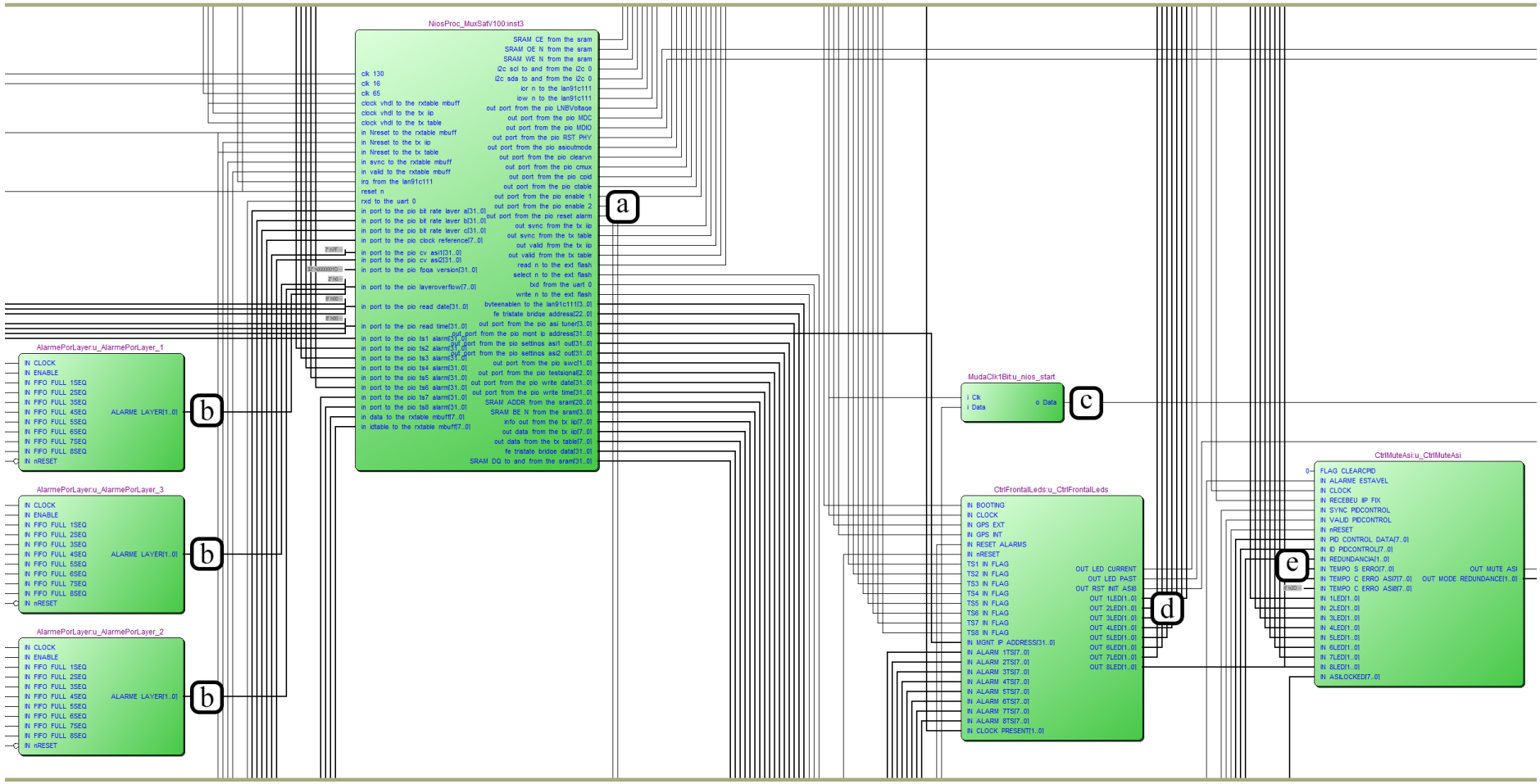


Figura 3.5 – Esquemático RTL (parte 2) do projeto do Multiplexador implementado em Stratix II. a) Sistema do Platform Designer contendo NIOS b) IPs de alarme das FIFOs de cada camada c) IP de mudança de domínio de *clock* d) IP de controle dos LEDs de cada entrada ASI e) IP de controle do ”mudo” dos alarmes.

Fonte: Adaptado do Quartus II.

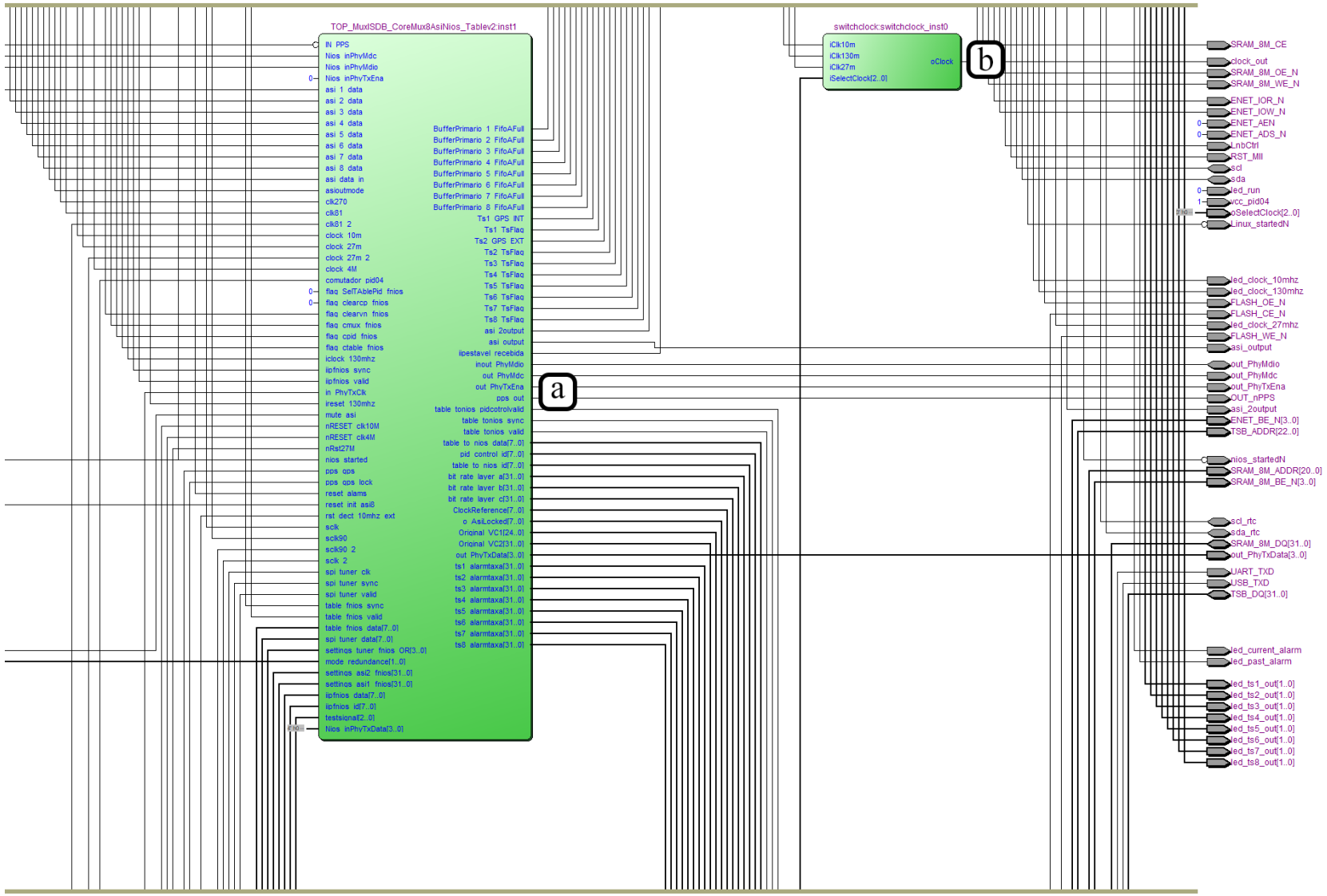


Figura 3.6 – Esquemático RTL (parte 3) do projeto do Multiplexador implementado em Stratix II. a) Multiplexador b) IP que controla qual *clock* vai para LED de indicar *clock*.

Fonte: Adaptado do Quartus II.

3.3 Arquitetura Proposta para a Atualização do Multiplexador

Pretende-se atualizar o Multiplexador para tecnologias mais recentes. Isso envolve tanto mudar o *chip* FPGA usado, quanto mudar o *software* onde o projeto será desenvolvido. Nas Seções de 3.3.1 a 3.3.3, são explicados os principais objetivos dessa atualização. Nas Figuras 3.7 e 3.8 podem ser vistos, respectivamente, os diagramas externo e interno da arquitetura proposta para essa atualização.

O diagrama externo mostra como o novo componente do Multiplexador se conecta ao resto do sistema. Através de sua interface Avalon, ele se comunica com o processador, neste caso o HPS, para ler e escrever parâmetros de configuração disponíveis. Através de um protocolo de comunicação desenvolvido pela empresa, o Multiplexador se comunica com componentes que fazem alguns processamentos com as tabelas contidas nos TSs. Esses componentes foram deixados fora do encapsulamento pois já possuíam interfaces Avalon independentes do Multiplexador.

O diagrama interno mostra como esse encapsulamento Avalon foi organizado. A interface Avalon lê e escreve em registradores internos do Multiplexador. Anteriormente, esses registradores eram implementados juntos do processador e se conectavam ao Multiplexador através de IPs *Parallel Input/Output* (PIO), que deixavam registradores disponíveis para serem acessados de forma paralela.

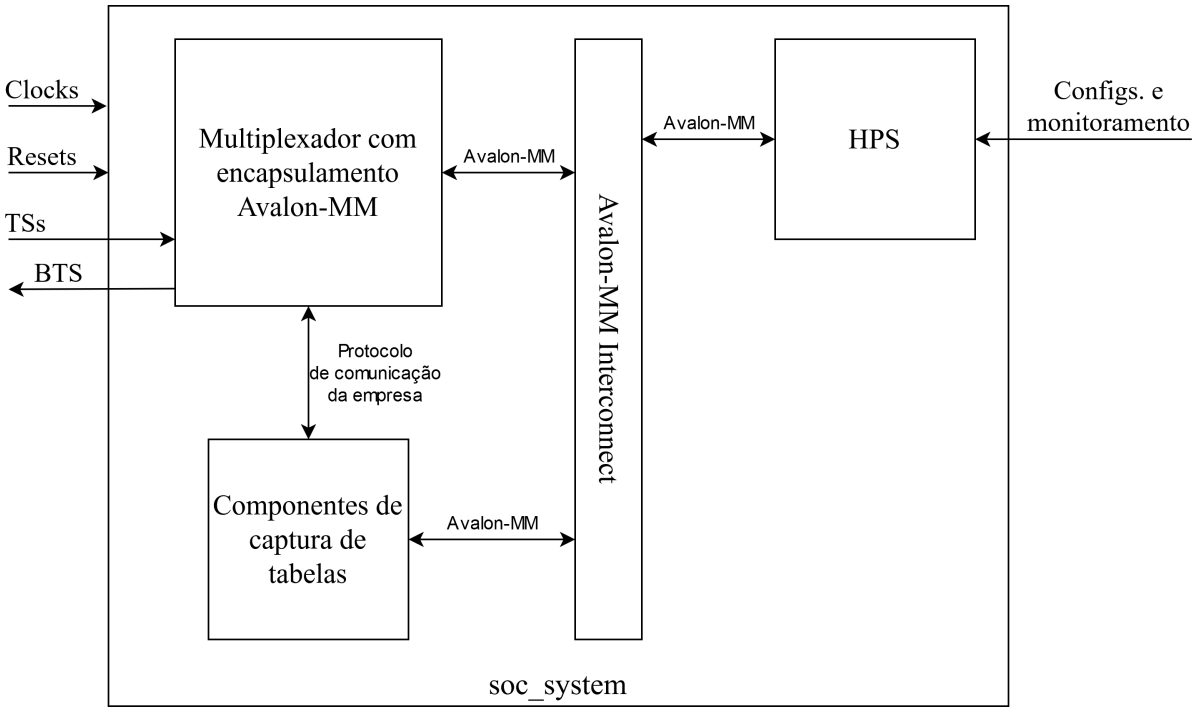


Figura 3.7 – Diagrama da arquitetura externa proposta para atualização do Multiplexador.

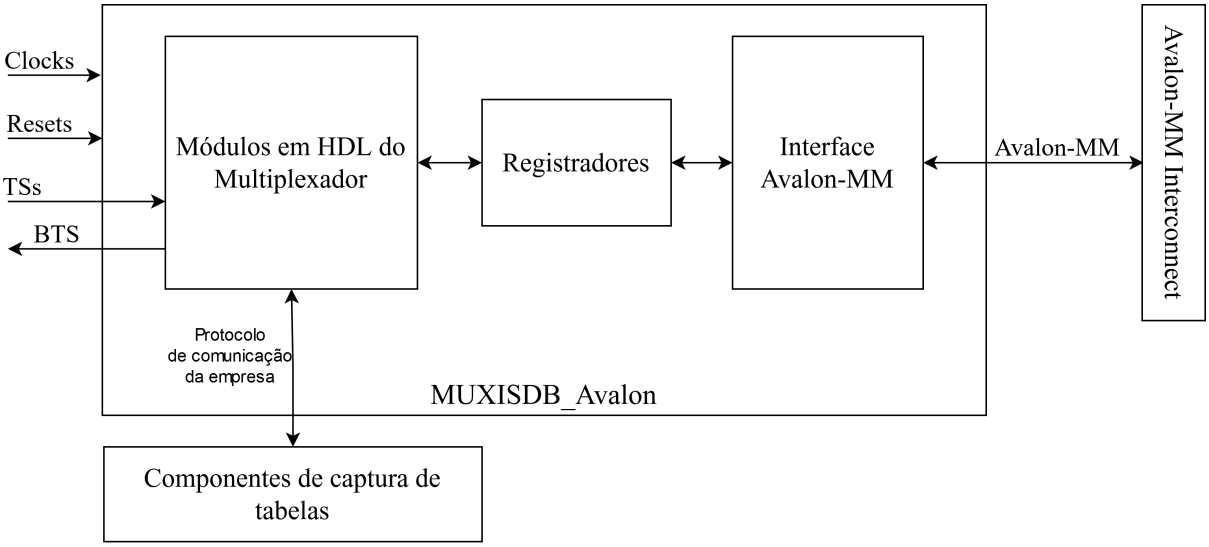


Figura 3.8 – Diagrama da arquitetura interna proposta para atualização do Multiplexador.

3.3.1 Remapeamento para Cyclone V

A empresa deseja substituir o dispositivo atual por um da família Cyclone V, também da Intel. Dispositivos dessa família possuem um processador ARM Cortex A9, chamado HPS, e diversos periféricos que o complementam, como conexões especiais com a seção de lógica programável do *chip* (Intel, 2018).

Além de ser uma família de dispositivos mais atual, e com isso possuir performance teoricamente melhor, o processador é capaz de rodar uma distribuição de Linux embarcada para ele, de forma que toda configuração e processamento podem ser feitos pelo próprio aparelho, através de uma simples conexão SSH via cabo ethernet, por exemplo.

3.3.2 Atualização para Quartus Prime

O projeto atual da empresa foi desenvolvido no *software* Quartus II, da Intel. A última versão desse software foi lançada em 2013, tornando-o uma ferramenta relativamente antiga. A Intel ainda não o tornou obsoleto, mas recomenda na própria página de *downloads* a se utilizar a versão mais recente do produto (Intel, s.d.b).

O modelo mais recente do Quartus é denominado Quartus Prime, com a última versão sendo lançada em 2024 (Intel, 2024a), até o momento de elaboração deste trabalho. Esse programa é o indicado pela Intel para ser usado em novos projetos com suas FPGAs. Utilizando ele, pode-se garantir que o Multiplexador terá suporte com as ferramentas atuais da fabricante do Cyclone V, evitando problemas de obsolescência para a empresa.

3.3.3 Encapsulamento do Multiplexador com Avalon-MM

Com o intuito de facilitar manutenção, modificação e comunicação com o Multiplexador, sua nova arquitetura inclui um encapsulamento com a interface Avalon-MM. Esse protocolo foi escolhido pela sua relativa simplicidade de implementação no lado do escravo e por ser o que vários outros IPs usados pela empresa já usam.

Assim, o Multiplexador pode ser integrado a outros componentes usando o Platform Designer, além de ser possível comunicar com ele tanto com o HPS, com o Nios, ou com outros IPs com a mesma interface, a depender das necessidades futuras da empresa.

3.4 Planejamento Experimental

3.4.1 Caracterização dos circuitos

Os *designs* propostos serão primeiramente avaliados de acordo com os resultados obtidos pelo Quartus. Os critérios de caracterização dos circuitos obtidos estão baseados no consumo de recursos de hardware, consumo de memória, temporização e frequência de operação, e consumo de energia. Serão modificadas algumas diretivas de implementação, como tamanho de memórias, conexões entre módulos e/ou restrições de temporização, observando seus efeitos no consumo de recursos e nas análises de temporização.

3.4.2 Teste do encapsulamento Avalon

Pretende-se testar o funcionamento do encapsulamento Avalon-MM através da leitura e escrita dos registradores criados nessa interface. Esses registradores são uma adição ao projeto e não dependem totalmente do Multiplexador para funcionarem. Como essa interface é mapeada em memória no HPS, os valores desses registradores podem ser lidos e escritos através de *software*, de forma que um programa em C pode configurar os registradores e validar o funcionamento da interface.

3.4.3 Teste no equipamento da empresa

Como a empresa enviou um de seus aparelhos como uma plataforma para testar essa nova implementação do Multiplexador, pretende-se configurar esse aparelho com o *design* desenvolvido neste trabalho.

Usando a ferramenta Signal Tap, é possível observar em tempo real sinais escolhidos de dentro do Multiplexador, podendo-se analisar o comportamento e o tráfego de dados ao longo de diversos componentes de Multiplexador. Esse experimento é de grande ajuda na análise do *design* quando ele não produz os resultados esperados, ou seja, não apresenta os TSs corretamente multiplexados na saída de BTS.

4 Resultados

4.1 Implementação do Multiplexador atual em Cyclone V

Como uma análise preliminar para averiguar se é possível trazer o *design* atual para Cyclone V, foi feita a mudança no *chip* no projeto do Quartus II e rodadas as etapas de Análise, Síntese e PAR. Nas Figuras 4.1, 4.2 e 4.3, pode-se ver que o esquemático RTL é igual ao mostrado nas Figuras 3.4, 3.5 e 3.6, salvo posicionamento gráfico dos componentes. Isso é o esperado, umas vez que esse esquemático é uma abstração das conexões dos módulos em HDL, que não mudou entre *chips*.

Na Tabela 4.1, pode-se ver que o *design* foi implementado de fato no Cyclone V, porém não de forma completa. Ao trocar de dispositivo, o Quartus apaga todas as conexões das portas do *design* com os pinos, uma vez que esses mudam. Isso ocorre pois os dispositivos possuem quantidades de pinos diferentes, com funções diferentes, e é papel do projetista indicar à ferramenta qual a função desejada de cada um. Então o programa não é capaz de realizar as análises de temporização completamente, indicando que o modelo de temporização é preliminar.

Na Tabela 4.2, pode-se ver que apareceram mais problemas de temporização em relação à Tabela 3.3, e todos eles relacionados aos PLLs que geram *clocks* conectados às entradas ASI. Tempos negativos indicam que o sinal não cumpre com aquele requisito de temporização.

Essa implementação, no entanto, não é totalmente equivalente à feita em Stratix II. Uma biblioteca interna utilizada pelo Nios, de um controlador do protocolo *Serial Peripheral Interface* (SPI), não possui implementação em Cyclone V. Foi necessário substituir o código do controlador por um que apenas emula seu comportamento a nível de simulação.

Tabela 4.1 – Consumo de recursos do Multiplexador implementado em Cyclone V.

Recurso	Tipo/Quantidade
Family	Cyclone V
Device	5CSEMA6F31C7
Timing Models	Preliminary
Logic utilization (in ALMs)	23.365/41.910 (56%)
Total registers	33.050/166.036 (20%)
Total pins	221/457 (48%)
Total block memory bits	1.389.788/5.662.720 (25%)
Total DSP blocks	79/112 (71%)
Total PLLs	5/6 (83%)
Total DLLs	0/2 (0%)

Fonte: Adaptado do Quartus II

Tabela 4.2 – Sumário das violações de temporização no projeto em Cyclone V. Valores em ns.

Clock	Setup	Hold	Recovery	Removal
inst2|generic_pll1|divclk	-35,911	-0,119	-7,800	0,277
ip_pll_asi_1|generic_pll1|divclk	-37,042	-2,384	-5,211	-0,274
ip_pll_asi_1|generic_pll2|divclk	-0,581	0,121	8,733	0,258
ip_pll_asi_2|generic_pll2|divclk	-0,693	-0,031	3,696	-0,490
ip_pll_asi_2|generic_pll3|divclk	1,085	0,172	-2,348	-0,387
ip_pll_asi_2|generic_pll4|divclk	0,753	2,264	-3,885	1,028
u_ip_pll_i27m_o130m|generic_pll1|divclk	-5,337	-2,123	-8,447	0,693
u_ip_pll_i27m_o130m|generic_pll2|divclk	-3,088	-2,042	-7,086	-0,766
u_ip_pll_i27m_o130m|generic_pll3|divclk	42,504	0,175	-0,826	-0,119

Fonte: Adaptado do Quartus II

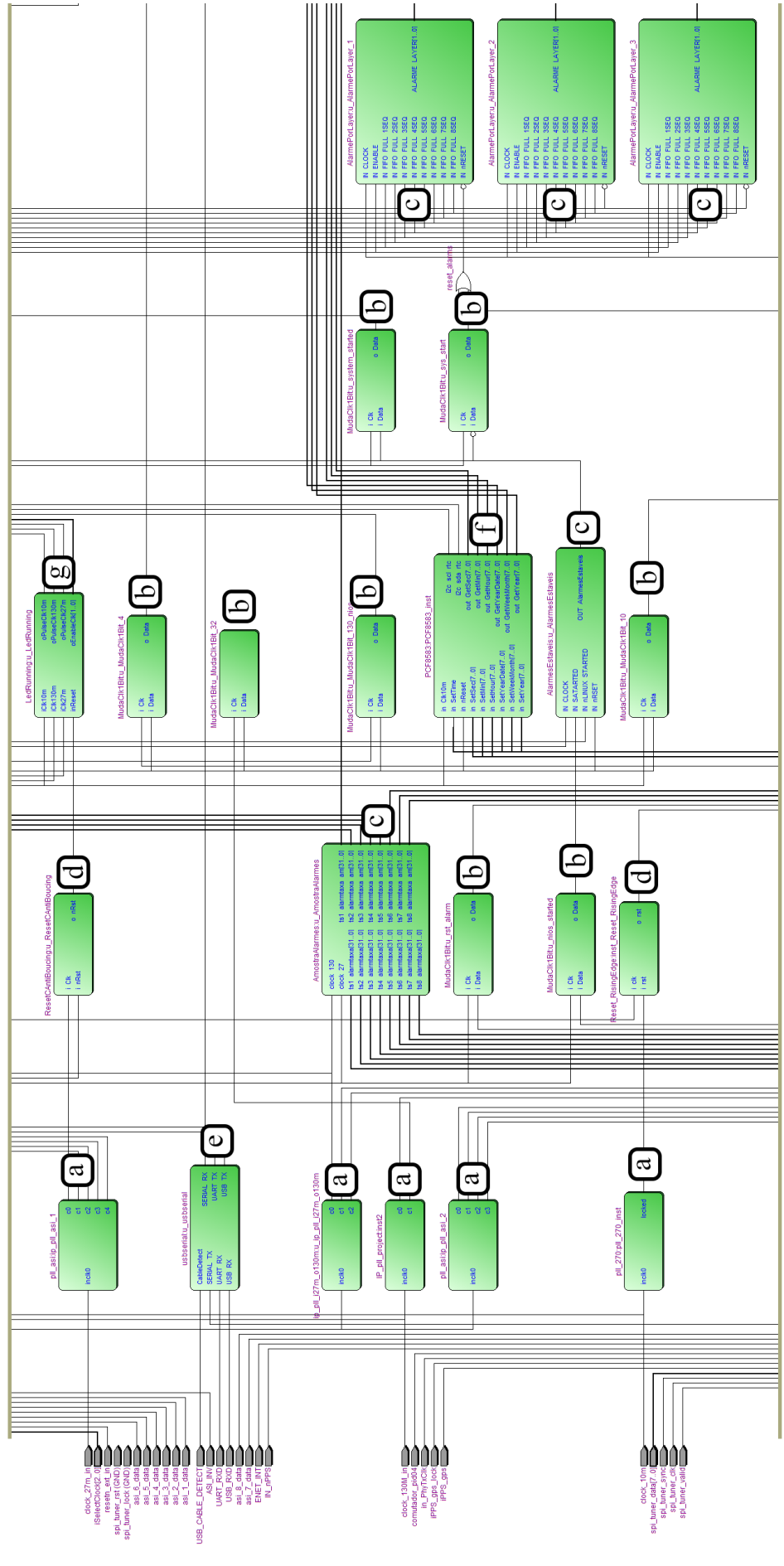


Figura 4.1 – Esquemático RTL (parte 1) do projeto do Multiplexador implementado em Cyclone V. a) PLLs b) IPs de mudança de domínio de *clock* c) IPs de alarmes d) IPs de *reset* e) IP conversor de USB para Serial f) IP de comunicação com RTC fora da FPGA g) IP que controla LEDs indicadores de *clocks*.

Fonte: Adaptado do Quartus II.

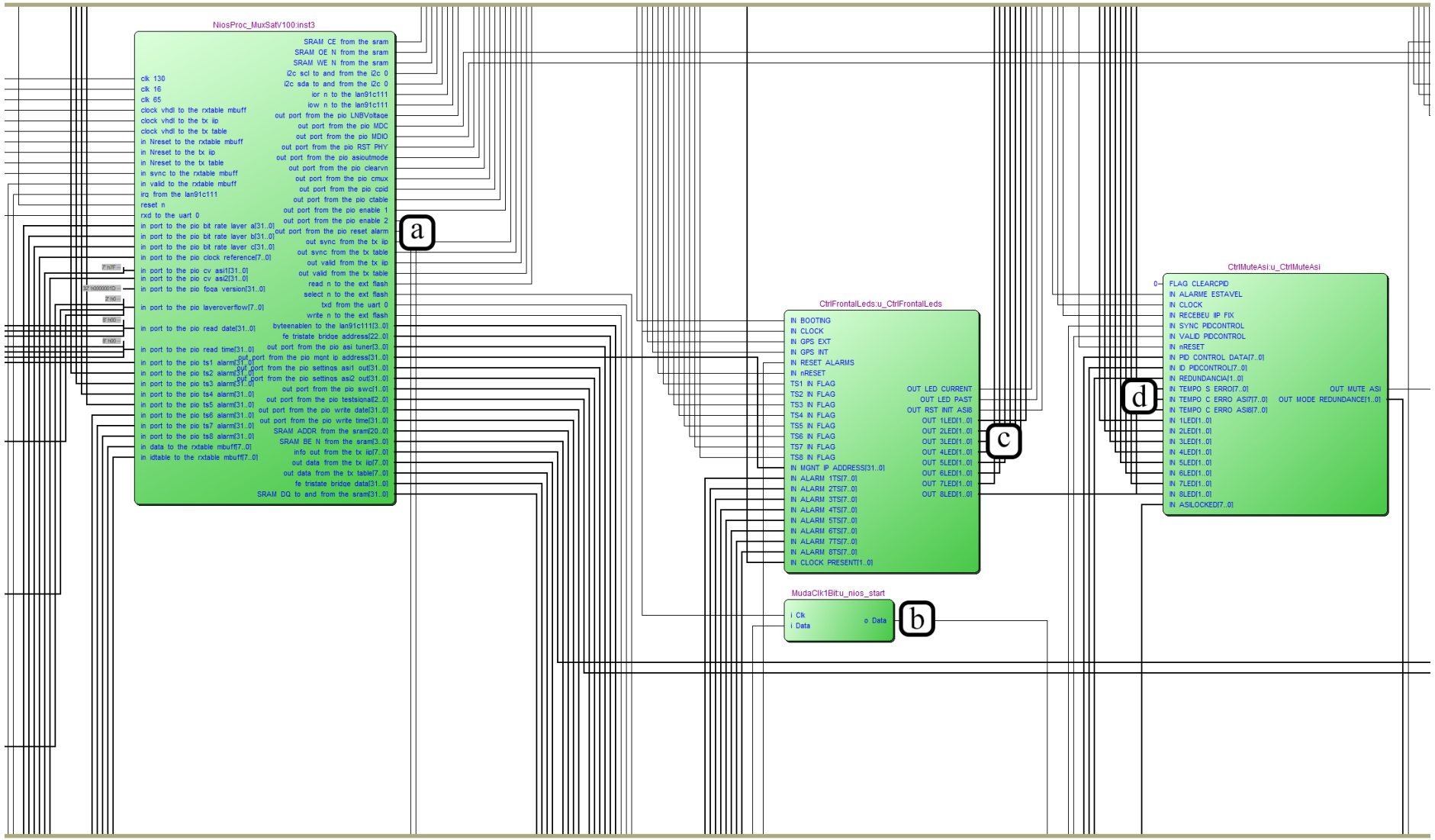


Figura 4.2 – Esquemático RTL (parte 2) do projeto do Multiplexador implementado em Cyclone V. a) Sistema do Platform Designer contendo NIOS b) IP de mudança de domínio de *clock* c) IP de controle dos LEDs de cada entrada ASI d) IP de controle do ”mudo” dos alarmes.

Fonte: Adaptado do Quartus II.

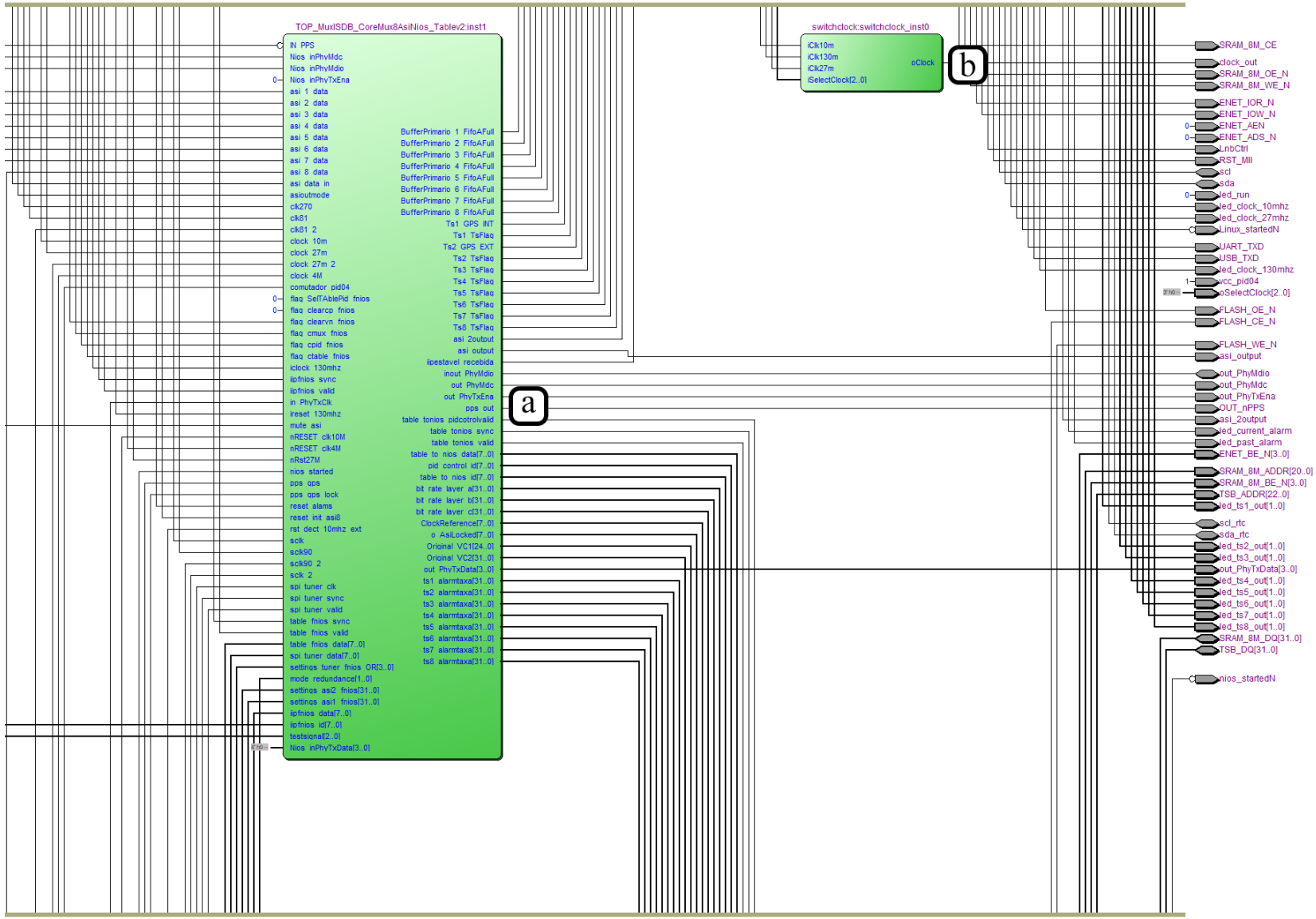


Figura 4.3 – Esquemático RTL (parte 3) do projeto do Multiplexador implementado em Cyclone V. a) Multiplexador b) IP que controla qual *clock* vai para LED de indicar *clock*.

Fonte: Adaptado do Quartus II.

4.2 Encapsulamento do Multiplexador com Avalon-MM

O encapsulamento foi feito em VHDL, seguindo a arquitetura mostrada na Figura 3.8, em um arquivo que contém o IP principal do Multiplexador e diversos outros IPs da própria empresa que comunicam com o Multiplexador e manipulam dados vindos ou que vão para ele. A interface Avalon-MM foi implementada através de dois processos VHDL, um de leitura e um de escrita, que respectivamente leem e escrevem em registradores internos do projeto. Esses registradores, por sua vez, estão conectados aos diversos IPs desse arquivo, da forma mais similar possível ao *design* original.

O projeto original possui IPs que realizam alguns processamentos nos sinais ASI para extrair ou inserir tabelas ao sinal. Esses IPs já possuem interfaces Avalon para se comunicar com o processador e também possuem uma interface criada pela empresa para se comunicarem com o Multiplexador. Foi decidido deixar esses IPs fora do encapsulamento, como mostrado na Figura 3.8, para que se conectassem ao mesmo barramento Avalon do Multiplexador, e foram mantidas as conexões da interface da empresa através de portas do tipo conduítes.

Para validar esse encapsulamento, foi criado um projeto no Platform Designer chamado `soc_system`, que contém o sistema com o Multiplexador, HPS e IPs de captura de tabelas, como mostrado na Figura 3.7. Esse sistema foi então incluído em um arquivo VHDL que contém IPs existentes no projeto original, sendo eles PLLs, componentes que geram *clocks* nas diversas frequências requeridas pelo Multiplexador, e componentes de troca de domínio de *clock* e geração de *resets*, criados pela própria empresa.

Na Figura 4.4 encontra-se o esquemático RTL desse projeto de validação. Apesar da difícil visualização causada pela forma como o Quartus gera seus esquemáticos, é possível diferenciar os principais componentes. O bloco verde maior, destacado em vermelho, é o sistema que contém o Multiplexador, HPS e demais IPs. Os blocos verdes menores, destacados em laranja, são as PLLs, componentes de mudanças de *clocks* e componentes que realizam *resets*. Os blocos azuis na parte inferior da figura, destacados em azul, são a implementação de um contador que faz um LED da placa piscar a uma frequência fixa de 1 Hz, para verificar visualmente que a FPGA foi configurada corretamente. Esse projeto foi implementado na placa de desenvolvimento SoCkit, da Arrow.

Na Tabela 4.3 encontra-se o consumo de recursos desse projeto. Pode-se ver que a quantidade de recursos é similar aos da Tabela 4.1, indicando que o Multiplexador, o componente comum entre ambas implementações, é responsável pela maior parte da utilização. Apesar do modelo de temporização ser indicado como "final"nessa tabela, ele não representa a análise de *timing* completa do projeto. Como o Arrow SoCkit não possui os mesmos pinos que o projeto original requer, esses pinos do *design* não estão associados a pinos físicos da FPGA, impossibilitando que seja feita uma análise de temporização completa.

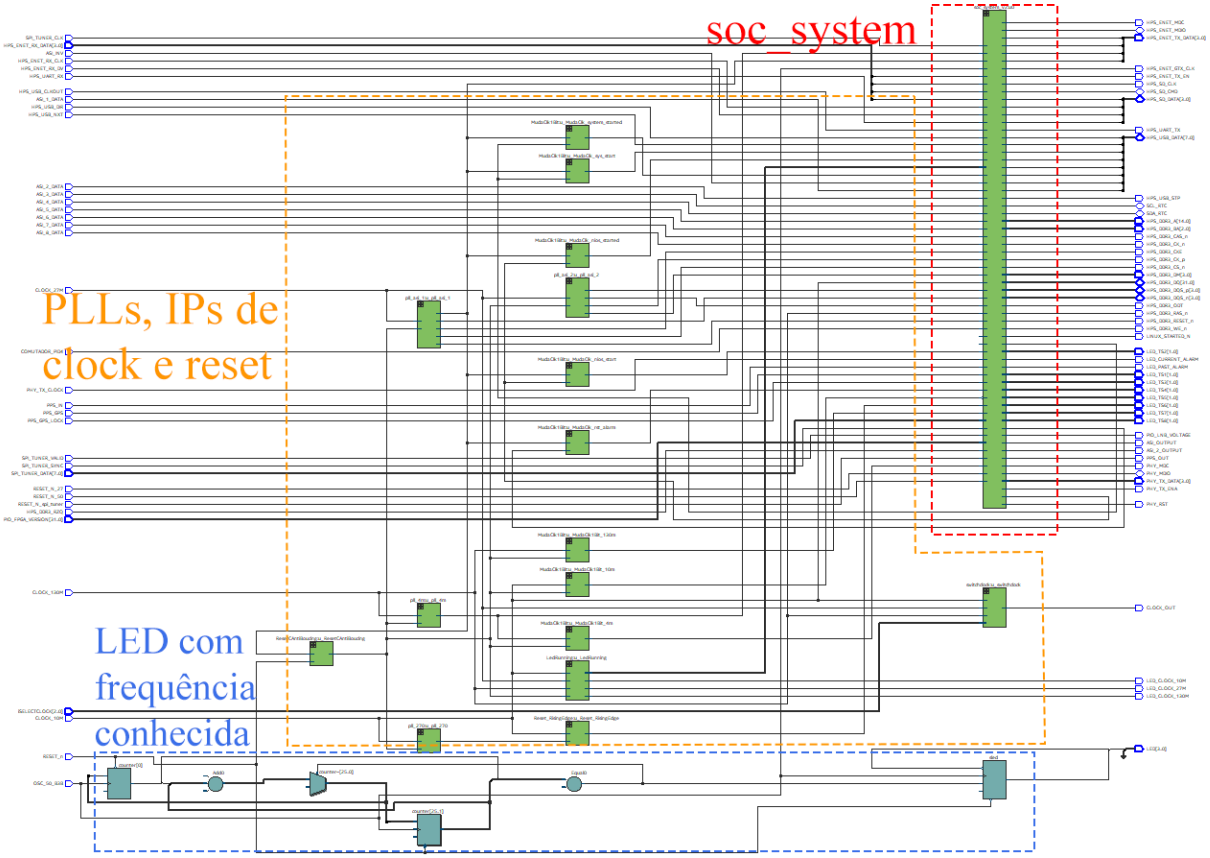


Figura 4.4 – Esquemático RTL do projeto para validar encapsulamento do Multiplexador com Avalon-MM.

Fonte: Adaptado do Quartus Prime 18.1

Tabela 4.3 – Consumo de recursos do Multiplexador encapsulado implementado em Cyclone V no Arrow SoCkit.

Recurso	**Tipo/Quantidade**
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	20.358/32.070 (63%)
Total registers	27.473/128.300 (21%)
Total pins	215/457 (47%)
Total block memory bits	1.205.710/4.065.280 (30%)
Total DSP blocks	76/87 (87%)
Total PLLs	4/6 (67%)
Total DLLs	1/4 (25%)

Fonte: Adaptado do Quartus Prime 18.1

Na Figura 4.5 pode-se ver a disposição dos recursos utilizados no chip. Em vermelho está destacado o `soc_system`, o componente que inclui a conexão com o HPS e os IPs do Multiplexador. Em azul escuro estão os IPs de *clock* e *reset*. Em laranja estão as PLLs

instanciadas nas regiões específicas delas, que contém *hardware* dedicado para tal. Em azul acinzentado são os demais componentes do projeto.

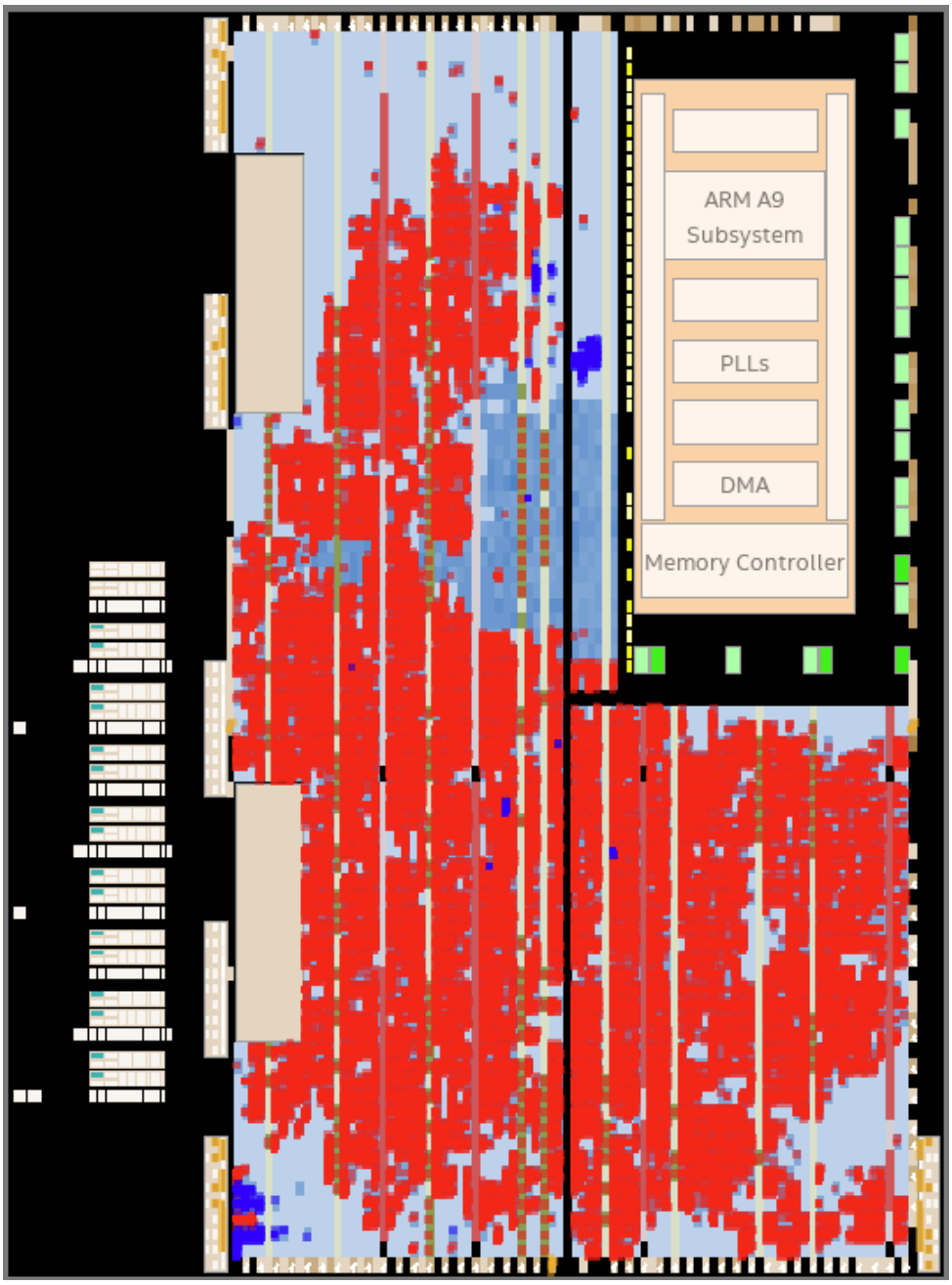


Figura 4.5 – Visualização do consumo de recursos do projeto no chip Cyclone V. Em vermelho, o `soc_system` contendo Multiplexador. Em azul escuro, IPs de *clock* e *reset*. Em laranja, PLLs em *hardware*. Em azul acinzentado, demais componentes.

Fonte: Retirado do Quartus Prime 18.1

4.2.1 Verificação da implementação no Arrow SoCkit

Apesar desse projeto possuir muitas entradas e saídas (elementos azuis nas laterias esquerda e direita da Figura 4.4), a maioria delas não pode ser verificada na placa SoCkit pois não há formas de conectar esses pinos. A maioria dos pinos da FPGA disponíveis se

encontram em um conector High Speed Mezzanine Card (HSMC), que necessita de um cabo ou adaptador específico. Então, para testar no kit de desenvolvimento, o projeto foi simplificado removendo grande parte da sua pinagem, mantendo apenas os pinos necessários para o funcionamento básico do sistema, como *clocks* disponíveis na placa, conexões permitindo comunicação entre HPS e Multiplexador, e o LED de frequência conhecida.

Com o SoCkit conectado ao computador através de um cabo ethernet, é possível enviar arquivos ao Sistema Operacional (SO) da placa usando o comando `scp` nativo do Linux. Usando um programa de comunicação serial, neste caso o Putty, é possível interagir com a linha de comando do HPS, que é conectada ao computador através de um cabo micro-USB. Na Figura 4.6 pode-se ver a bancada onde esse teste foi efetuado.

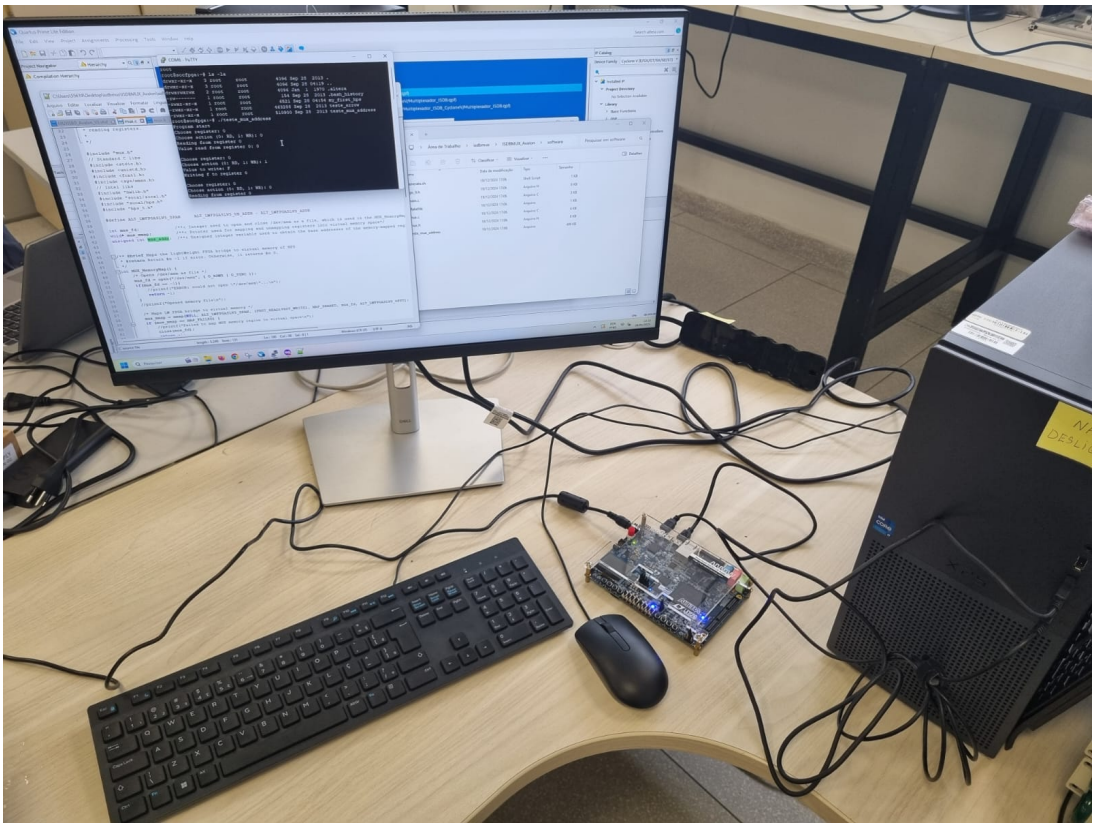


Figura 4.6 – Bancada no laboratório da universidade contendo computador e placa de desenvolvimento SoCkit.

Foi desenvolvido um programa em C que realiza a comunicação com a interface Avalon. Esse programa, que roda no próprio HPS, foi compilado em um computador do laboratório da universidade, com o SO Linux, usando ferramentas de compilação cruzada, pois a arquitetura do HPS é diferente da arquitetura dos computadores do laboratório e o Linux do HPS não veio com compiladores próprios.

O SO do HPS foi adquirido no site da fabricante da placa (Terasic, s.d.), pois ele já foi preparado para funcionar no SoCkit, considerando o *hardware* disponível na placa pro Linux acessar através do HPS.

Esse programa consiste em um *loop* que pede para o usuário escolher um registrador da interface Avalon, dá a opção de leitura e escrita desse registrador, mostra o valor atual do registrador se escolher ”leitura” ou escreve nele se escolher ”escrita”. Na Figura 4.7 pode-se ver as operações feitas no registrador com endereço `0x00`, que possui 1 bit de largura. Apesar do valor de escrita ser `F` em hexadecimal (que possui o valor `1111` em binário), apenas o bit menos significativo foi escrito ao registrador.

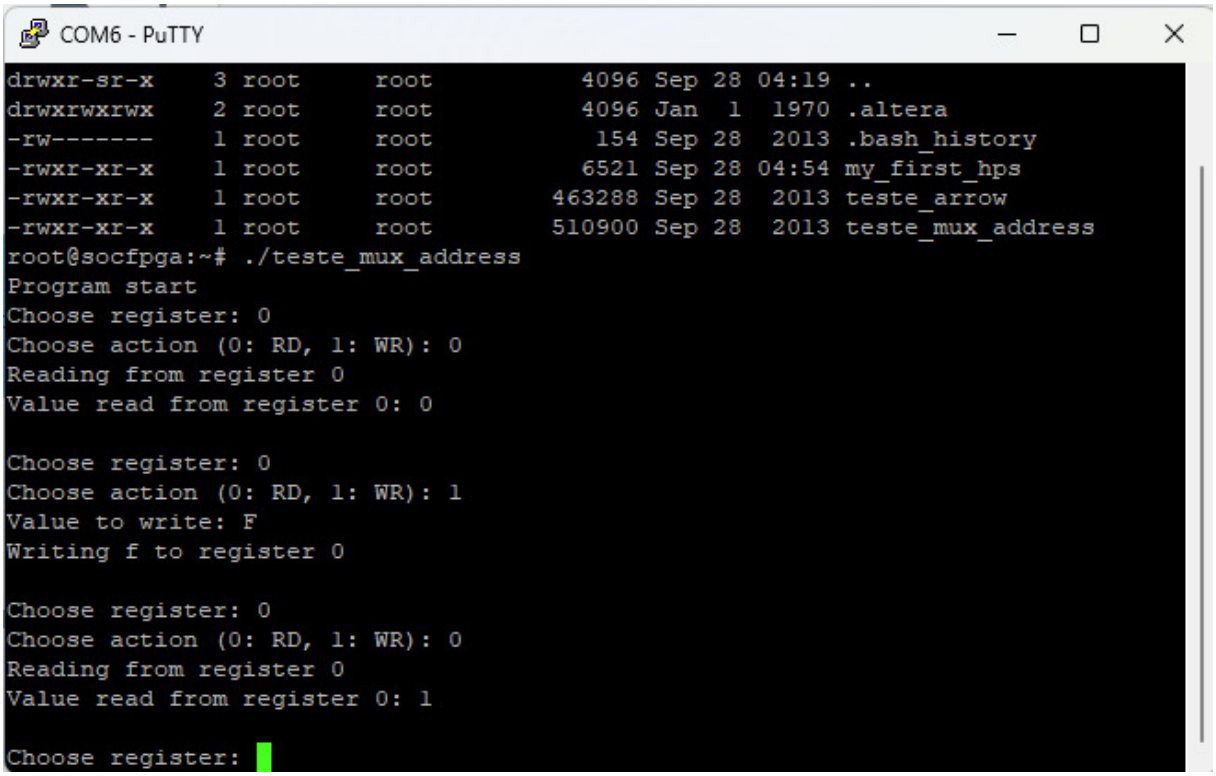


Figura 4.7 – Saída do programa em C, no terminal vindo do HPS, no primeiro teste da interface Avalon.

Na Figura 4.8 pode-se ver as operações feitas em outros registradores. O endereço `0x0B` é de um registrador de 32 bits, então ao escrever `F`, esse valor completo é armazenado e lido de volta pela interface. O endereço `0x04` é de um registrador de 2 bits, então ao escrever `F`, apenas os dois últimos bits são escritos e a leitura retorna o valor `3`, que é `11` em binário.

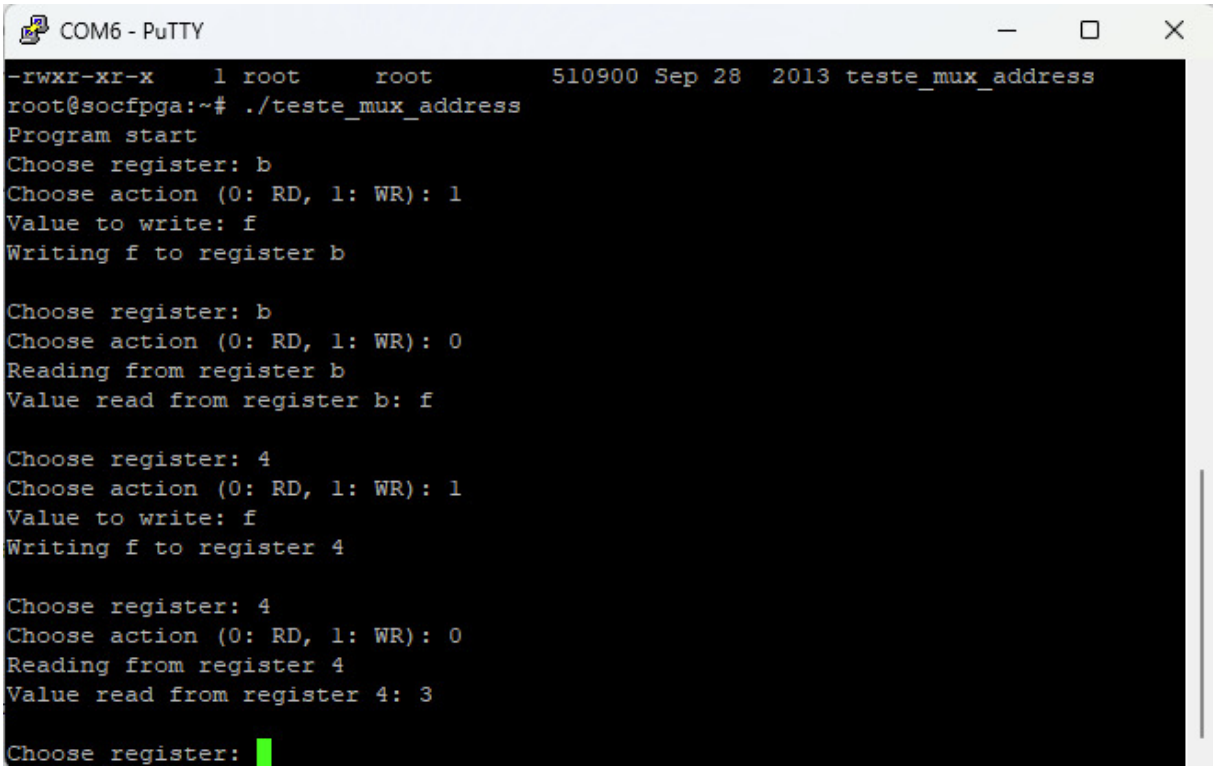


Figura 4.8 – Saída do programa em C, no terminal vindo do HPS, no segundo teste da interface Avalon.

4.3 Testes no equipamento da empresa

Os testes descritos nesta seção não estavam previstos no cronograma original do projeto. Contudo, visto o desenvolvimento rápido e efetivo até então, a empresa incentivou a realizá-los para ver até onde seria possível testar no laboratório da universidade.

Para realizar testes no aparelho enviado pela empresa, são necessários alguns recursos que realizam a interface e controle com o *hardware* do equipamento. Para isso, a empresa desenvolveu e disponibilizou um projeto no Quartus Prime chamado "MUX_POC", que consiste nos componentes mínimos na FPGA para o funcionamento do aparelho. Esses componentes incluem vários módulos em HDL que controlam periféricos fora do chip, como o microcontrolador Nios, o SO Linux que a empresa adequou para a placa do equipamento, vários programas nesse SO e o *firmware* do microcontrolador.

Como o Multiplexador e seus IPs aderentes foram encapsulados em Avalon, a inclusão dele nesse *design* consistiu em adicioná-los ao projeto do Platform Designer, conectá-los ao barramento Avalon já existente e conectar os demais sinais necessários, como *clocks*, *resets*, *ethernet* etc. Após essa conexão, foi criada uma configuração no Signal Tap, com alguns sinais internos do Multiplexador selecionados para serem monitorados.

Dada a falta de documentação do projeto, foi necessário analisar o esquemático RTL do Multiplexador detalhadamente, considerando a arquitetura descrita por Juliano (Ferreira, 2009), para escolher que sinais monitorar para verificar o funcionamento do projeto.

Decidiu-se monitorar os sinais ASI em 3 etapas:

1. Após o componente `ProcessaTs`, pois mostraria que os TSs estão sendo recebidos e processados pelos módulos na entrada;
2. Após passarem pelo `MuxDataBufferPrimario` de cada entrada, pois mostraria que os TSs estão sendo colocados nesses *buffers* corretamente antes de serem multiplexados;
3. Após o componente `MuxTableAndData` e antes do `BufferSecundario`, pois mostraria que os TSs estão tendo tabelas inseridas neles e são multiplexados para cada camada do padrão ISDB-TB (camadas A, B e C).

Por recomendação da empresa, o Signal Tap inicia a captura de amostras desses sinais nas bordas de subida do sinal que indica saída válida no módulo `ProcessaTs`.

Em seguida, a compilação desse projeto precisou ser feita pela empresa, pois ele utiliza recursos do Quartus apenas disponíveis em licenças pagas do programa, que não se obteve acesso no laboratório da universidade. São dois principais recursos:

- Logic Lock: O Quartus permite travar componentes em uma localização fixa na FPGA, de forma que a ferramenta de PAR sempre os posiciona naquele local, gerando resultados mais previsíveis em cada iteração do projeto (Intel, 2024d).
- Incremental Compilation: Partes do projeto podem ser separadas de forma que não são recompiladas quando não há mudanças nelas, permitindo que se itere mais rapidamente por modificações pontuais no projeto (Intel, 2021b).

O Logic Lock foi usado em alguns componentes do `MUX_POC` para garantir que ficassem próximos às entradas e saídas que lhe foram designados. A Incremental Compilation foi usada por ser um projeto relativamente grande, envolvendo vários engenheiros o desenvolvendo simultaneamente, e com isso o trabalho de um não afeta significamente o de outro.

Na Figura 4.9 pode-se ver como a bancada foi disposta para a realização desse teste. Ao aparelho da empresa foram conectados o DTU-245 em uma entrada ASI e um adaptador USB Blaster na porta JTAG para realizar o monitoramento do Signal Tap. Neste, foi feito o monitoramento contínuo, que mostra novas amostras capturadas automaticamente.

A empresa disponibilizou também alguns arquivos de TSs para testes, que possuem conteúdos arbitrários, como vários *streams* de vídeo, áudio e tabelas que os identificam. Na Figura 4.10, pode-se ver a tela do programa StreamXpress, que foi usado para enviar um desses TSs para o aparelho da empresa através do DTU-245. Esse programa faz uma análise do TS e mostra os parâmetros identificados do seu conteúdo.

A Figura 4.11 mostra a tela do Signal Tap durante o monitoramento contínuo nos sinais das primeiras duas etapas, no `ProcessaTs` e `MuxDataBufferPrimario`. Pode-se ver que o TS está sendo recebido na entrada, porém não está aparecendo no `MuxDataBufferPrimario`. Isso pode ter acontecido porque esse TS é lido rapidamente do *buffer*, não dando tempo do Signal Tap registrar os sinais e o *buffer* aparentar estar sempre vazio.



Figura 4.9 – Bancada no laboratório da universidade contendo computador e aparelho da empresa.

A Figura 4.12 mostra a tela do Signal Tap com os sinais da última etapa escolhida, no `MuxTableAndData` e `BufferSecundario`. Pode-se ver que não há mudança nos sinais de dados do TS, apenas nos sinais de contadores de *bytes*. Como a contagem aparenta estar funcionando, com os valores subindo sequencialmente, isso valida que os componentes também foram implementados corretamente.

A ausência de sinal do TS pode ocorrer por vários motivos, sendo um deles o fato de que não estão sendo enviados sinais de controle do processador para o Multiplexador através da interface Avalon, então ele não tem uma instrução de qual *buffer* deve receber o sinal de entrada e nada é direcionado para a saída.

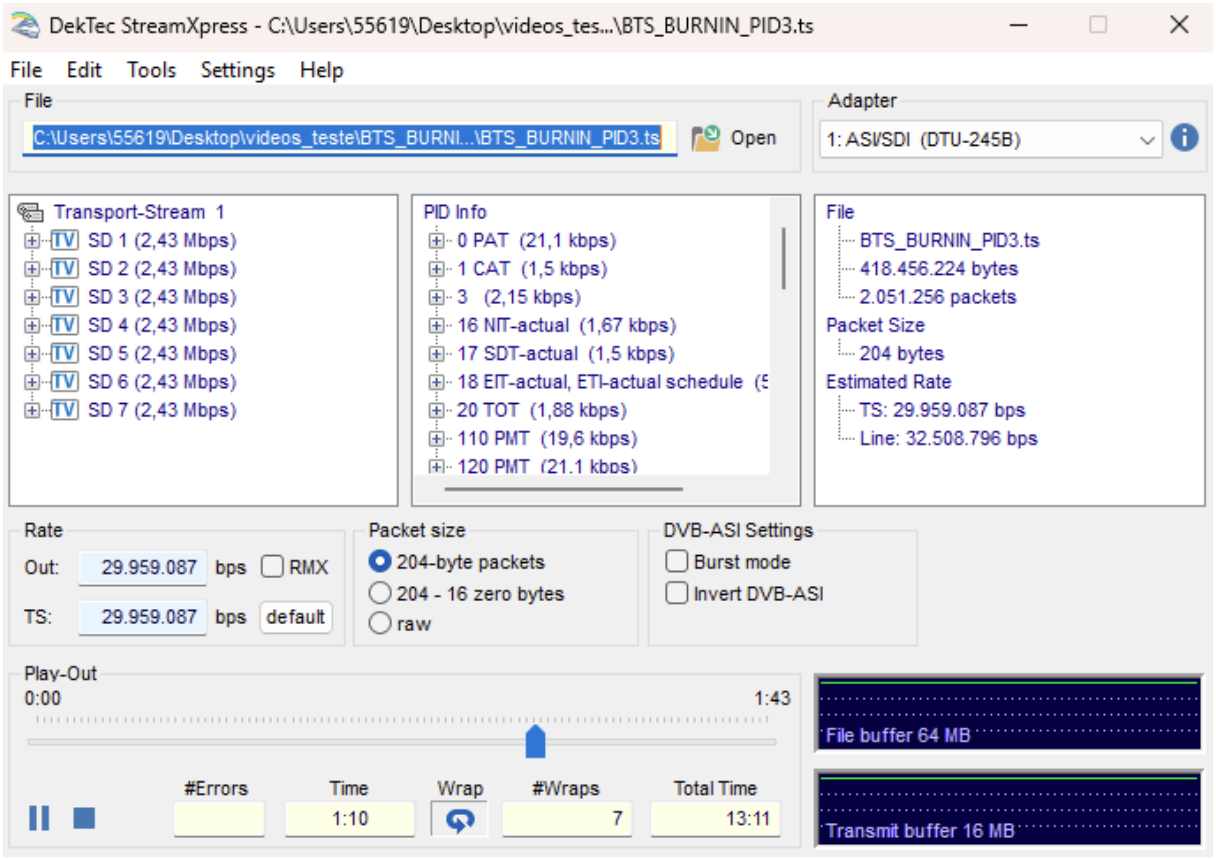


Figura 4.10 – Tela do programa StreamXpress mostrando os parâmetros identificados no TS sendo enviado ao aparelho da empresa.

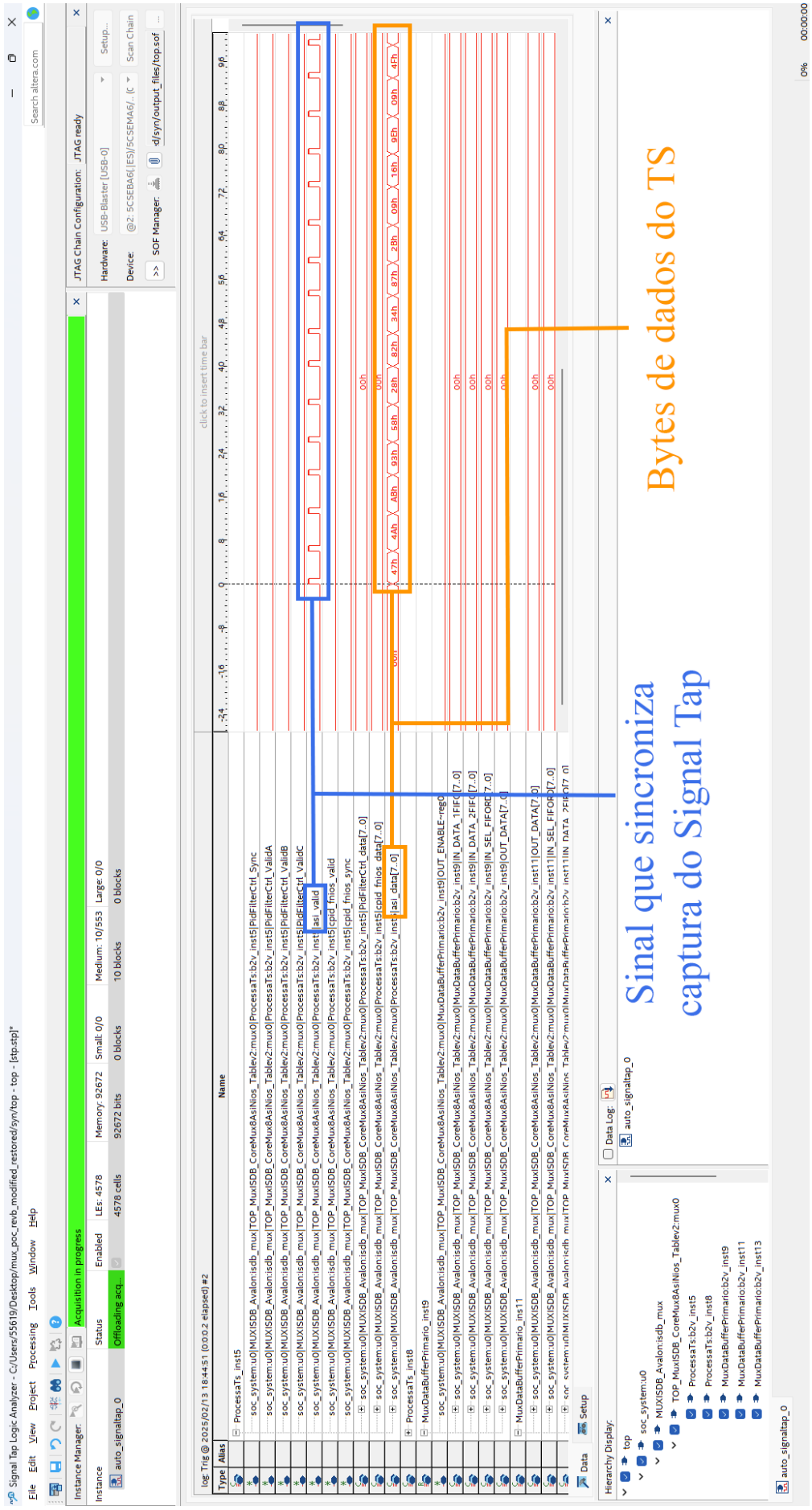


Figura 4.11 – Signal Tap mostrando os sinais internos do Multiplexador continuamente, nas primeiras 2 etapas escolhidas.

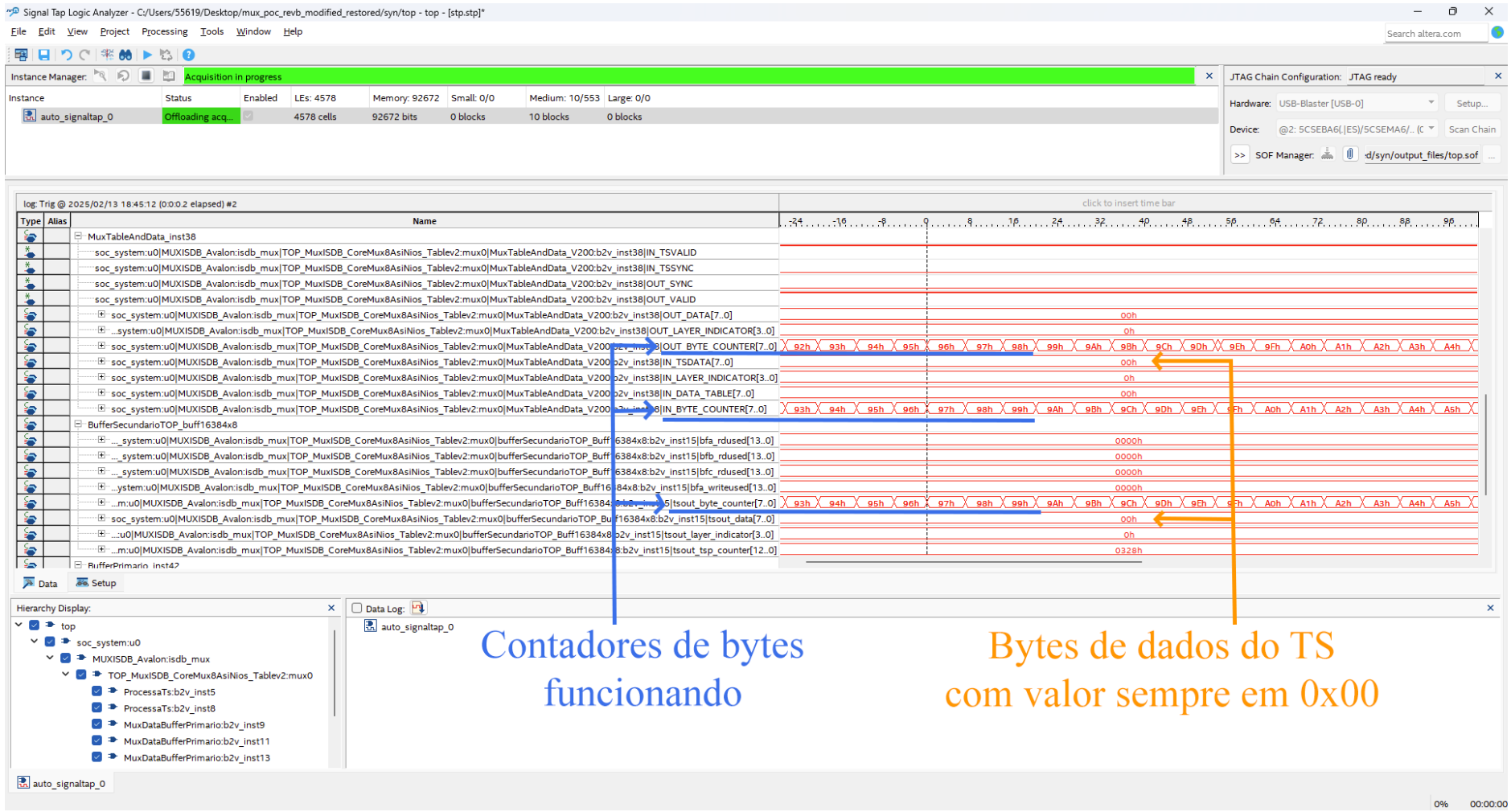


Figura 4.12 – Signal Tap mostrando os sinais internos do Multiplexador continuamente, na última etapa escolhida.

5 Conclusões

Neste trabalho, verificou-se que é possível implementar o projeto original do Multiplexador ISDB-TB no *chip* Cyclone V da Intel/Altera. Foi utilizado o protocolo Avalon para realizar um mapeamento em memória dos IPs internos do Multiplexador, com o intuito de configurá-lo através do processador ARM Cortex A9. Devido a alguns IPs internos do Multiplexador ISDB-TB não estarem disponíveis para a família Cyclone V, foi necessário realizar modificações do projeto. O projeto modificado utiliza 63% de Look-Up Tables, 21% de Flip-flops, 30% de blocos de memória e 87% de blocos DSPs, como visto na Tabela 4.3, o que indica ser uma boa escolha, capaz de ter o *design* implementado e ainda ter espaço para alguns recursos adicionais.

O encapsulamento do Multiplexador em uma interface Avalon se mostrou funcional e efetivo para configuração, manutenção e modificação do projeto. Isso pôde ser verificado nos testes de bancada mostrados nas Figuras 4.7 e 4.8. Com essa interface padronizada, é possível conectá-lo tanto a um processador como a um mestre Avalon qualquer, que o configure e o controle da forma que a empresa desejar.

As modificações do encapsulamento ocorreram apenas em sinais de configuração e controle, enquanto os sinais de dados foram mantidos inalterados, conectados a portas do projeto que vão para pinos da FPGA. Assim, os caminhos de dados dos TSs não sofreram alterações e pôde-se verificar parcialmente esse funcionamento, como mostrado nas Figuras 4.11 e 4.12.

Dada a complexidade e carência de documentação desse projeto, que possui códigos desenvolvidos desde 2006, a empresa se mostrou muito satisfeita com esses resultados. Durante o processo de desenvolvimento foi utilizado o padrão de documentação Doxygen tanto para o hardware embarcado quando para o software embarcado no HPS.

Os testes de integração do Multiplexador com encapsulamento Avalon-MM com o projeto `MUX_POC` permitiram demonstrar que o Multiplexador pode ser levado para um novo equipamento, requerendo para isso algumas etapas de desenvolvimento, a saber:

a. Verificação detalhada da integridade dos TSs através de um analisador de PIDs;
b. Desenvolvimento de um código em C para configurar através do HPS as diversas camadas de multiplexação e IPs internos;
c. Integração do Multiplexador ao *hardware* específico.

Referências

AKAMINE, C.; IANO, Y.; VALEIRA, G. de M.; BEDICKS, G. Re-Multiplexing ISDB-T BTS Into DVB TS for SFN. **IEEE Transactions on Broadcasting**, v. 55, n. 4, p. 802–809, 2009. Citado na p. 26.

ALTERA CORPORATION. **Cyclone V Device Handbook**: Volume 1: Device interfaces and integration. San Jose, CA, 2023. Disponível em: https://www.intel.com/content/www/us/en/docs/programmable/683375. Acesso em: 6 jun. 2024. Citado na p. 18.

Arrow. **SOCKIT - Arrow Development Tools**. 2013. Disponível em: https://www.arrow.com/en/products/sockit/arrow-development-tools. Acesso em: 7 jan. 2025. Citado nas pp. 27 e 28.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 15601**: Televisão digital terrestre — Sistema de transmissão. Rio de Janeiro, 2020. Citado nas pp. 16 e 17.

D'AMORE, R. **VHDL - Descrição e Síntese de Circuitos Digitais**. [*S.l.*]: LTC, 2012. ISBN 9788521620549. Citado na p. 19.

DekTec. **DTU-245 - FantASI ASI/SD-SDI Input and Output for USB-2**. s.d. Disponível em: https://www.dektec.com/Products/Usb/DTU-245/. Acesso em: 15 fev. 2025. Citado nas pp. 28 e 29.

Digital Broadcastig Experts Group. **Features of ISDB-T system**. 2007. Disponível em: https://www.dibeg.org/techp/feature/. Acesso em: 05 mai. 2024. Citado na p. 14.

ENDRES, L. P. **Development of an MPEG2 Multiplexer compliant with SBTVD Digital TV Standard**. Dissertação (Trabalho de Conclusão de Curso) — Universidade Federal do Rio Grande do Sul, Porto Alegre, RS, BR, June 2014. Disponível em: https://lume.ufrgs.br/handle/10183/105054. Citado na p. 26.

FERREIRA, J. S. **Uma proposta de implementação do Multiplexador do sistema ISDB-TB.** Dissertação (Dissertação de Mestrado) — Instituto Nacional de Telecomunicações, 2009. Citado nas pp. 14, 29, 30 e 48.

Intel. **Glossary - Block Design File (.bdf) Definition**. 2017. Disponível em: https://www.intel.com/content/www/us/en/programmable/quartushelp/17.0/reference/glossary/def_bdf.htm. Acesso em: 24 jun. 2024. Citado na p. 30.

Intel. **Cyclone V Device Overview**. 2018. Disponível em: https://www.intel.com/content/www/us/en/docs/programmable/683694/current/cyclone-v-device-overview.html. Acesso em: 30 mai. 2024. Citado nas pp. 20, 21 e 36.

Intel. **PRODUCT DISCONTINUANCE NOTIFICATION - PDN2041**. US, 2021. Disponível em: https://media.digikey.com/pdf/PCNs/Intel/PDN2041_Rev2.0.0.pdf. Acesso em: 11 mai. 2024. Citado na p. 14.

Intel. **Quartus® Prime Standard Edition User Guide: Design Compilation**. 2021. Disponível em: https://www.intel.com/content/www/us/en/docs/programmable/683283/18-1/about-incremental-compilation.html. Acesso em: 01 mar. 2025. Citado na p. 49.

Intel. **Quartus Prime Design Software**. 2024. Disponível em: https://www.intel.com/content/www/us/en/products/details/fpga/development-tools/quartus-prime.html. Acesso em: 30 mai. 2024. Citado na p. 36.

Intel. **Quartus Prime Pro Edition User Guide: Platform Designer**. 2024. Disponível em: https://www.intel.com/content/www/us/en/docs/programmable/683609/24-3/creating-a-system-with.html. Acesso em: 05 dez. 2024. Citado na p. 24.

Intel. **Quartus® Prime Pro Edition User Guide: Debug Tools**. 2024. Disponível em: https://www.intel.com/content/www/us/en/docs/programmable/683819/24-3/faq.html. Acesso em: 25 jan. 2025. Citado nas pp. 24 e 25.

Intel. **Quartus® Prime Pro Edition User Guide: Design Optimization**. 2024. Disponível em: https://www.intel.com/content/www/us/en/docs/programmable/683641/24-3/defining-placement-constraints.html. Acesso em: 01 mar. 2025. Citado na p. 49.

Intel. **FPGA e FPGA SoC Cyclone V**. s.d. Disponível em: https://www.intel.com.br/content/www/br/pt/products/details/fpga/cyclone/v/article.html. Acesso em: 8 jun. 2024. Citado na p. 19.

Intel. **FPGA Software Download Center**. s.d. Disponível em: https://www.intel.com/content/www/us/en/collections/products/fpga/software/downloads.html?s=Newest&f:guidetmD240C377263B4C70A4EA0E452D0182CA=%5BQuartus%C2%AE%20II%20Software%5D. Acesso em: 30 mai. 2024. Citado na p. 36.

INTEL CORPORATION. **Avalon Interface Specifications**. [*S.l.*], 2022. Disponível em: https://www.intel.com/content/www/us/en/docs/programmable/683091/22-3/. Acesso em: 18 jun. 2024. Citado nas pp. 21 e 23.

INTEL CORPORATION. **Nios II Processor Reference Guide**. [*S.l.*], 2023. Disponível em: https://www.intel.com/content/www/us/en/docs/programmable/683836/current/eol.html. Acesso em: 10 jun. 2024. Citado na p. 20.

LIMA, E. R. D. **Architecture and algorithms for the implementation of digital wireless receivers in FPGA and ASIC: ISDB-T and DVB-S2 cases**. Tese (Doutorado) — Universitat Politècnica de València, Valência, Espanha, 2016. Disponível em: http://hdl.handle.net/10251/61967. Citado na p. 26.

MARCHI, E. J.; CERVETTO, M. A.; TENORIO, M. L. A DDR3 memory based time interleaving FPGA implementation for ISDB-T standard. *In*: **2011 VII Southern Conference on Programmable Logic (SPL)**. [*S.l.: s.n.*], 2011. p. 1–5. Citado na p. 26.

NAKAMURA, M.; SATO, A.; MIYASAKA, H.; AASAKURA, S.; SHITOMI, T.; SHIRAI, N.; TAKEUCHI, T.; OKANO, M.; TSUCHIDA, K.; NARIKIYO, Y.; MURAYAMA, K. A

Study on the Transmission System of an Advanced ISDB-T. *In*: **2019 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)**. [*S.l.: s.n.*], 2019. p. 1–5. Citado na p. 25.

OLIVEIRA, G. H. Maranhão Garcia de; AKAMINE, C.; MACIEL, Y. P. Implementation of ISDB-T LDM broadcast system using LDPC codes. *In*: **2016 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)**. [*S.l.: s.n.*], 2016. p. 1–4. Citado na p. 25.

PEDRONI, V. A. **Circuit Design with VHDL**. Londres, Inglaterra: MIT Press, 2004. ISBN 9780262162241. Citado na p. 18.

RABAçA, R.; OLIVEIRA, G.; JERJI, F.; AKAMINE, C. Implementation of an 8K-DTV broadcast system using SDR. **SET INTERNATIONAL JOURNAL OF BROADCAST ENGINEERING**, v. 2019, p. 10–15, 08 2019. Citado na p. 25.

RABAçA, R. S.; JERJI, F.; AKAMINE, C. Implementation of a 3-layer LDM Broadcast System Backward-compatible with ISDB-T B. **Radioengineering**, v. 30, n. 2, 2021. Citado na p. 25.

Terasic. **SoCKit - the Development Kit for New SoC Device**. s.d. Disponível em: https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=167&No=816&PartNo=4. Acesso em: 30 jan. 2025. Citado na p. 46.

UEHARA, M.; TAKADA, M.; KURODA, T. Transmission scheme for the terrestrial ISDB system. **IEEE Transactions on Consumer Electronics**, v. 45, n. 1, p. 101–106, 1999. Citado na p. 16.

VALEIRA, G. de M.; AKAMINE, C.; HORTA, E. L.; YAMADA, F.; MOTOYAMA, R. IP Transmission over ISDB-TB. **SET International Journal of Broadcast Engineering**, v. 1, n. 4, 2015. Citado na p. 25.

VALEIRA, G. de M.; BEDICKS, G.; AKAMINE, C.; HORTA, E. L.; STOLFI, G. Transport stream analysis of a isdb-t signal using an embedded and reconfigurable system. **IEEE Transactions on Broadcasting**, v. 61, n. 1, p. 30–38, 2015. Citado na p. 26.