UNIVERSIDADE FEDERAL DO AMAZONAS
INSTITUTO DE COMPUTAÇÃO
Programa de Pós-Graduação em Informática

Metodologia de Reconfiguração de Hardware Utilizando o Sinal de TV Digital

Rodrigo Ribeiro de Oliveira

Manaus – Amazonas – Brasil
Março, 2015

Metodologia de Reconfiguração de Hardware Utilizando o Sinal de TV Digital

Tese submetida ao Programa de Pós-Graduação do Instituto de Computação da Universidade Federal do Amazonas como requisito parcial para obter o título de Doutor em Informática.

Rodrigo Ribeiro de Oliveira

Orientador: Prof. Vicente Ferreira de Lucena Júnior, Dr.

Manaus – Amazonas – Brasil

Março, 2015

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

Ribeiro de Oliveira, Rodrigo
R484m Metodologia de Reconfiguração de Hardware Utilizando o Sinal de TV Digital / Rodrigo Ribeiro de Oliveira. 2015
104 f.: il.; 31 cm.

Orientador: Vicente Ferreira de Lucena Júnior
Tese (Doutorado em Informática) - Universidade Federal do Amazonas.

1. TV Digital. 2. Reconfiguração de hardware. 3. MPEG-2. 4. SBTVD. 5. Transmissão de dados. I. Lucena Júnior, Vicente Ferreira de II. Universidade Federal do Amazonas III. Título

UNIVERSIDADE FEDERAL DO AMAZONAS

FOLHA DE APROVAÇÃO

Metodologia de Reconfiguração de Hardware Utilizando o Sinal de TV Digital

RODRIGO RIBEIRO DE OLIVEIRA

Prof. Dr. Vicente Ferreira de Lucena Júnior (Orientador).
UNIVERSIDADE FEDERAL DO AMAZONAS

Prof. Dr. Eduardo James Pereira Souto.
UNIVERSIDADE FEDERAL DO AMAZONAS

Prof. Dr. Raimundo da Silva Barreto.
UNIVERSIDADE FEDERAL DO AMAZONAS

Prof. Dr. Carlos Eduardo Pereira.
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Prof. Dr. Waldir Sabino da Silva Júnior.
UNIVERSIDADE FEDERAL DO AMAZONAS

Manaus – Amazonas, Março, 2015

“Nas grandes batalhas da vida, o primeiro passo para a vitória é o desejo de vencer.”

(Mahatma Gandhi)

Agradecimentos

Dedico meus sinceros agradecimentos:

primeiramente a Deus, por guiar e iluminar o meu caminho, dando-me coragem e persistência.

ao Professor e orientador Dr. Vicente Ferreira de Lucena Júnior, pela oportunidade, confiança, paciência e orientação.

aos meus pais, Ana e Gomercindo, pelo exemplo de fé, força e perserverança.

aos professores, Dr. Eddie Batista de Lima Filho e Dr. Lucas Carvalho Cordeiro, pela amizade, incentivo e colaboração para o aprimoramento desta pesquisa.

a todas as pessoas que direta ou indiretamente me ajudaram para a conclusão deste trabalho. Especialmente a Thales Santos, Márcia Henke, Cynthia Cardoso, Raquel Oliveira e Rossano Arnoldi, pelo apoio e incentivo.

e por fim, a todos os funcionários e professores do programa de Pós-Graduação em Informática da Universidade Federal do Amazonas.

Resumo

Esta tese de doutorado apresenta uma metodologia de reconfiguração de hardware, que utiliza o sinal da emissora de TV digital como base para atualizar módulos de receptores de TV digital. A metodologia proposta permite que núcleos de hardware sejam sinalizados, durante a geração do fluxo de transporte, transmitidos e posteriormente remontados. Desse modo, os núcleos recebidos são usados na reconfiguração de dispositivos baseados em lógica reprogramável (*field programable gate array* - FPGA), que estão integrados ao hardware de cada unidade receptora. Além disso, a sinalização de conteúdo permite a distinção entre núcleos sintetizados para FPGAs de diferentes fabricantes, o que habilita receptores de TV digital a selecionar somente núcleos de hardware relativos aos modelos de FPGA utilizados. Os resultados obtidos com os experimentos realizados, durante o desenvolvimento deste trabalho, consistem em uma prova de conceito e demonstram a viabilidade técnica do uso desta metodologia de transmissão e reconfiguração, para núcleos pré-sintetizados de hardware enviados em um ambiente de televisão digital. Fabricantes de receptores poderiam utilizar os benefícios desta metodologia para o desenvolvimento de arquiteturas reconfiguráveis, o que permitiria a incorporação de avanços tecnológicos às funções de hardware do receptor e um maior controle do ciclo de vida de produto. Como resultado, futuras revisões de normas de TV Digital não resultariam em troca de equipamento.

Palavras-chave: TV Digital, Reconfiguração de Hardware, MPEG-2, fluxo de transporte, transmissão de dados.

Abstract

This PhD thesis presents a novel hardware reconfiguration methodology, which uses the digital TV broadcast signal for reconfiguring hardware modules in digital TV receivers. The proposed methodology allows hardware cores to be signaled, during the transport stream generation step, transmitted and then reassembled. At the receiver, the recovered cores are then used to reconfigure reprogrammable field programmable gate array (FPGA) devices, which are integrated into each receiver unit. Besides, content signaling allows receivers to choose between FPGA cores synthesized for different manufacturers, which enables receivers to select broadcast hardware cores related to the employed FPGA models. The results of the performed experiments, which were carried out during the development of this work, consist in a proof of concept and show the technical feasibility of this methodology, regarding reconfiguration of pre-synthesized hardware cores through the digital TV environment. Receiver manufacturers could benefit from this methodology for developing reconfigurable architectures, which would allow the incorporation of technological advances into receiver hardware and provide better control regarding product life cycle. As a result, future revisions of DTV standards could occur without the need for device replacement.

Keywords: Digital television, Hardware Reconfiguration, MPEG-2, transport stream, data broadcasting.

Lista de Conteúdos

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

AAC	*Advanced Audio Coding*
ABERT	*Associação Brasileira de Emissoras de Rádio e Televisão*
ADIF	*Audio Data Interchange Format*
AIM	*Advanced Interconnect Matrix*
AIT	*Application Information Table*
ASCII	*American Standard Code for Information Interchange*
ASIC	*Application-Specific Integrated Circuits*
ATE	*Automated Test Equipament*
ATSC-M/H	*Advanced Television Systems Committee - Mobile/Handheld*
BAT	*Bouquet Association Table*
BER	*Bit Error Ratio*
CAT	*Conditional Access Table*
CAD	*Computer-Aided Design*
CETELI	*Centro de P&D em Tecnologia Eletrônica e da Informação*
CLB	*Configurable Logic Blocks*
CMMB	*China Multimedia Mobile Broadcasting*
CMOS	*Complementary Metal-Oxide Semiconductor*
COFDM	*Coded Orthogonal Frequency Division Multiplexing*
CPLD	*Complex Programmable Logic Device*
CRC	*Cyclic Redundancy Check*
DAVIC	*Digital Audio Visual Council*
DBPSK	*Differential Binary Phase Shift Keying*
DENC	*Digital Encoder*
DES	*Data Encryption Standard*
DQPSK	*Differential Quadrature Phase Shift Keying*
DSM-CC	*Digital Storage Media, Command and Control*
DSM-CC-OC	*Digital Storage Media - Command and Control Object Carousel*
DSM-CC-UU	*Digital Storage Media - Command and Control User to User*
DTC	*Datacasting*
DTMB	*Digital Terrestrial Multimedia Broadcast*
DTS	*Decode Time Stamp*

DVB	*Digital Video Broadcasting*
DVB-H	*DVB-Handheld*
DVB-T	*DVB-Terrestrial*
EDA	*Electronic Design Automation*
EDTV	*Enhanced Definition TeleVision*
EMM	*Entitlement Management Message*
EPG	*Electronic Programme Guide*
ETSI	*European Telecommunications Standards Institute*
FEC	*Forward Error Correction*
FPGA	*Field Programmable Gate Array*
GAL	*Generic Array Logic*
GPIO	*General Purpose Input/output*
GUI	*Graphical User Interface*
HCPLD	*High Capacity Programmable Logic Device*
HDL	*Hardware Description Language*
HDTV	*High Definition Television*
HDMI	*High-Definition Multimedia Interface*
HEAAC	*High-Efficiency Advanced Audio Coding*
HIR	*Header Instruction Register*
IOB	*Input/output Block*
ISO	*International Organization for Standardization*
ISDB-T	*Integrated Services Digital Broadcasting - Terrestrial*
ITU	*International Telecommunication Union*
LATM	*Low Overhead Audio Transport Multiplex*
LCD	*Liquid Crystal Display*
LUT	*Look Up Table*
MAC	*Media Access Control*
MPGA	*Mask Programmable Gate Array*
MHP	*Multimedia Home Platform*
MOSFET	*Metal-Oxide-Semiconductor Field Effect Transistors*
MPE	*Multi-Protocol Encapsulation*
MPEG	*Moving Picture Experts Group*
MPEG2-TS	*Moving Picture Experts Group 2 Transport Stream*
MSB	*Most Significant Bit*

MTU	*Maximum Transport Unit*
MUX	*Multiplex*
NIT	*Network Information Table*
NRE	*Non-Recurring Engineering*
OFDM	*Orthogonal Frequency Division Multiplexing*
PAT	*Program Association Table*
PES	*Packetized Elementary Streams*
PIA	*Programmable Interconnect Array*
PID	*Packet Identifier*
PIO	*Parallel Input/output*
PIOMAP	*Parallel Input/output Map*
PLA	*Programmable Logic Array*
PLL	*Phase Locked Loop*
PMT	*Program Map Table*
PROM	*Programmable Read-Only Memory*
PSI	*Program Specific Information*
PTS	*Presentation Time Stamp*
QAM	*Quadrature Amplitude Modulation*
QEF	*Quasi Error Free*
QPSK	*Quadrature Phase Shift Keying*
RSA	*Rivest, Ron; Shamir, Adi; Adleman Leonard*
RTL	*Register Transfer Level*
SBTVD	Sistema Brasileiro de TV Digital
SDR	*Scan Data Register*
SDT	*Service Description Table*
SDTV	*Standard Definition Television*
SFN	*Single Frequency Network*
SIR	*Scan Instruction Register*
SPLD	*Simple Programmable Logic Device*
STAPL	*Standard Test and Programming Language*
STB	*Set-Top Box*
STIMI	*Satellite and Terrestrial Interactive Multiservice Infrastructure*
TAP	*Test Access*
TCK	*Test Clock*

TDI	*Test Data In*
TDO	*Test Data Out*
TDR	*Trailer Data Register*
TDT	*Time and Date Table*
TMS	*Test Mode State*
TIR	*Trailer Instruction Register*
TOT	*Time Offset Table*
TRST	*Test Reset*
TTM	*Time to Market*
TS	*Transport Stream*
TVD	*TV Digital*
UHF	*Ultra-High Frequency*
URJTAG	*Universal JTAG*
VHDL	*VHSIC Hardware Description Language*
VHF	*Very-High Frequency*
VHSIC	*Very High Speed Integrated Circuits*
VLSI	*Very Large Scale Integration*

1. Introdução

A reconfiguração de hardware é amplamente utilizada em diversos segmentos da indústria de eletroeletrônicos e tem apresentado crescimentos substanciais na área de sistemas embarcados [1, 2, 3, 4, 5, 6, 7]. Estudos recentes indicam que o uso da reconfiguração de hardware pode trazer benefícios ao gerenciamento do ciclo de vida dos produtos e contribuir com a redução dos custos de projeto (*non-recurring engineering -- NRE*) [8, 9, 10, 11]. Tradicionalmente, circuitos integrados de aplicação específica (*application-specific integrated circuit* -- ASIC) oferecem o menor custo unitário, contudo, pressões crescentes para a colocação de produtos no mercado (*time to market* - TTM) aumentam exponencialmente os custos de NRE [11]. Por exemplo, uma máscara definida para um ASIC, no processo de 32/28nm, custa cerca de US$ 2M (2 milhões de dólares) (2013) [11]; por outro lado, para grandes volumes de circuitos integrados comercializados, o custo unitário do FPGA pode ser equiparado ao custo do ASIC, o que é essencialmente dado pela amortização dos custos de NRE entre os clientes de cada *integrated circuit* (IC). Tal compensação ocorre devido ao fato de que, em projetos baseados em FPGA, os custos de NRE são consideravelmente menores [10, 11].

Um receptor de sinal (set-top box - STB) é normalmente comercializado em larga escala (individualmente ou integrado a um televisor), o que acaba reforçando a necessidade de utilização da reconfiguração de hardware, para o controle do ciclo de vida do produto final, o qual está associado diretamente à arquitetura na qual o sistema foi concebido. Atualmente, alguns STBs disponíveis no mercado utilizam decodificadores de vídeo desenvolvidos em hardware, enquanto outros adotam processadores de maior capacidade e decodificadores em software [12]. Dessa forma, a preparação do sistema para o processo de reconfiguração deve ser realizada na concepção do projeto, de acordo com o escopo em que o sistema está inserido.

Um sistema de reconfiguração de hardware faz uso de lógica programável para determinar o funcionamento do FPGA, com base em um núcleo de hardware sintetizado. A síntese do núcleo de hardware é realizada através de uma ferramenta de projeto assistido por computador (*computer-aided design* – CAD), que, por sua vez, é baseada em um código com a descrição do hardware (gerado a partir de uma linguagem de descrição de hardware, ou *hardware description language – HDL,* como VHDL [13, 14, 15] e Verilog [16, 17, 18]). No contexto de um sistema de TV digital (TVD), o núcleo de hardware pode ser visto como um dado binário, que pode ser

transmitido juntamente com o conteúdo de áudio e vídeo da programação normal[1] (Figura 1). Um sistema de transmissão/recepção de TVD é composto por diversos subsistemas, que contemplam desde a preparação dos dados para transmissão até a recepção do sinal.

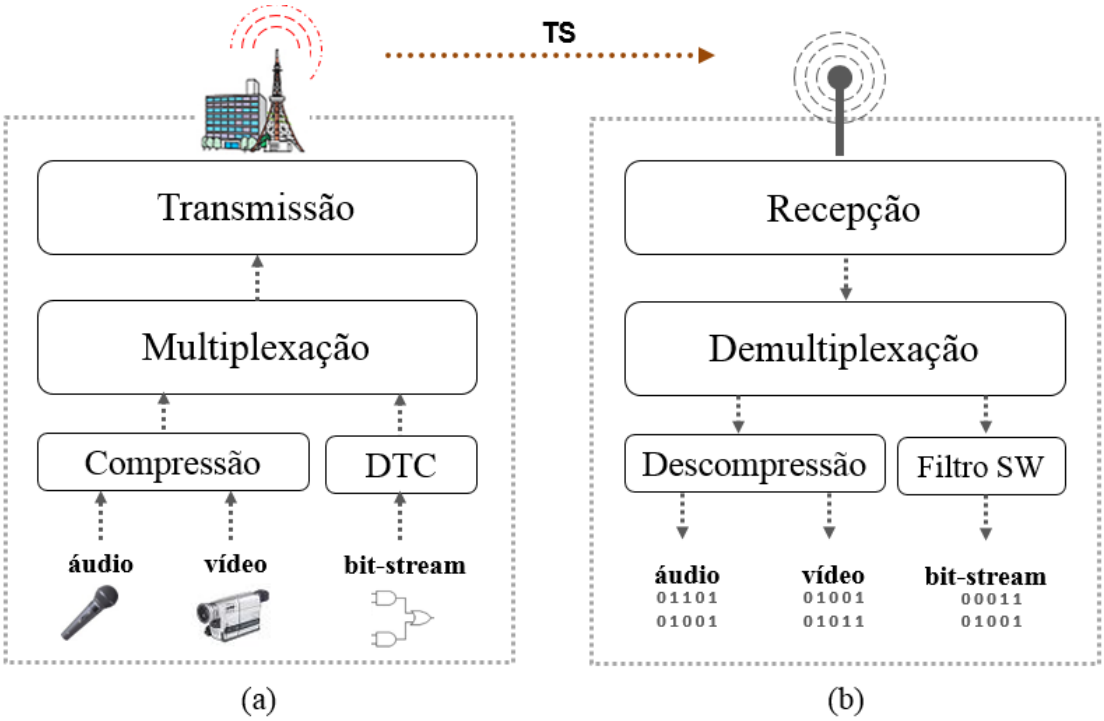

Figura 1 – (a) transmissão e (b) recepção do núcleo de hardware.

Após isso, o conteúdo enviado é filtrado e processado. O sinal recebido é composto por pacotes de áudio, vídeo e dados. No presente caso, os dados de reconfiguração de hardware são carregados através de um mecanismo de difusão de dados (*datacasting* – DTC). A Figura 1(a) descreve as três etapas básicas envolvidas na transmissão do sinal de TVD. Na etapa de compressão, os dados de áudio e vídeo são codificados com algoritmos do *moving pictures experts group* (MPEG): o vídeo é comprimido com o MPEG-4 *advanced video coding* (AVC) [19, 20, 21], também conhecido como H.264, e o áudio com o *advanced audio aoding* (AAC), considerando-se o sistema brasileiro de televisão digital (SBTVD) [22]. Após isso, o feixe de bits com a descrição de hardware é encapsulado, através de um mecanismo de DTC (ver seção 3.1.4). Na fase de multiplexação, áudio, vídeo e dados, juntamente com a descrição de hardware, são entrelaçados, formando um fluxo de transporte (*transport stream* – TS ou MPEG2-TS), o qual é necessário à transmissão de informações em um sistema de TVD. Por fim, a modulação é

[1] Programação normal da emissora de televisão contendo vídeo, áudio e demais informações necessárias para a reprodução deste conteúdo.

realizada na etapa de transmissão, quando o TS é então transmitido. A Figura 1(b) ilustra a recepção de sinal, que é primeiramente capturado e demodulado. O resultado desse processo inicial é então demultiplexado, extraindo-se pacotes de áudio, vídeo e dados, sendo que parte desses últimos é utilizada na reconfiguração do FPGA.

A reconfiguração pode ser efetuada com a interação do usuário, através de um sistema de controle residente que o notifica quando novas atualizações estiverem disponíveis, ou de forma automática, assim que o sistema identifica o conteúdo de atualização. Em qualquer dos casos, o sistema obtém o feixe de dados de reconfiguração e o grava na memória do FPGA. Após isso, o FPGA é automaticamente reiniciado, já com uma nova funcionalidade de hardware disponível na plataforma.

1.1 Definição do problema

A grande maioria dos receptores de TV Digital utiliza circuitos integrados com propósito específico, conhecidos por ASIC. As funções mais críticas de um receptor, como decodificação de vídeo, demodulação, demultiplexação e filtragem de conteúdo de transmissão, por exemplo, são geralmente realizadas em hardware, pois necessitam de maior poder de processamento. Tais funções são muitas vezes inviáveis de serem desenvolvidas na camada de software, tendo em vista que, muitas vezes, os sistemas embarcados possuem recursos limitados de processamento e memória, um fator que os diferencia de computadores de uso pessoal e estações de trabalho, que geralmente apresentam alto poder de processamento.

A utilização de ASICs cria um legado natural, gerado pela dependência física do hardware, o que acaba limitando e até mesmo impedindo atualizações e implantações de avanços tecnológicos, que poderiam ocorrer nas funções de hardware. Em relação à utilização de um ASIC, altos custos de engenharia associados ao processo completo de fabricação e projeto deste componente, para altos volumes de produção, justificam a sua substituição por um FPGA, que envolve custos de NRE e permite a quebra do legado inerente ao uso do ASIC.

Com o uso do FPGA, em receptores de TV Digital, várias funções de hardware poderiam ser simplesmente atualizadas (modificadas), quantas vezes fossem necessárias. Dessa forma, os receptores poderiam ser reconfigurados, possibilitando a implantação de diversas melhorias na codificação de sinal e permitindo a concepção de diversos projetos. Além disso, o uso do FPGA, considerando-se uma função de hardware atualizada, poderia prolongar o uso do equipamento.

Um exemplo de uma função de hardware, que poderia ser reconfigurada, é a criptografia do canal de comunicação/retorno. Utilizar diferentes métodos pode aumentar a segurança do

canal de comunicação de TV digital. Com isto, serviços que não utilizam este meio de comunicação podem vir a fazê-lo. Transações bancárias, que hoje são realizadas através da internet, constituem um exemplo viável. Outro benefício seria impedir o furto de sinal, que ocorre na área de TV por assinatura. A alternância de métodos de criptografia, através do mesmo canal de transmissão, poderia impedir e até mesmo inviabilizar o uso de plataformas que liberam (furto de sinal) os canais protegidos das operadoras de TV a cabo.

Os avanços que vem sendo constatados nas técnicas de compressão de vídeo, por exemplo, não podem ser implantados nos receptores de TV digital presentes no mercado brasileiro, em virtude do legado inerente à utilização do ASIC, no processo de decodificação de vídeo. Um exemplo é o sucessor do padrão de compressão de vídeo H.264, lançado recentemente e conhecido como *high efficiency video coding* (HEVC) [23, 24], que pode atingir taxas de compressão até 60% maiores que as apresentadas pelo seu antecessor [25]. A utilização do método de compressão de vídeo HEVC poderia reduzir consideravelmente a utilização de banda de transmissão de TV Digital, tendo em vista que grande parte do fluxo do sinal é composta por pacotes de vídeo. O mercado já disponibiliza descrições de hardware do algoritmo de compressão de vídeo HEVC, para diversas famílias de FPGA [26]. A eficiência deste novo método de compressão de vídeo é evidente e a sua utilização, no sistema de TV Digital, poderia trazer muitos benefícios. Uma alteração nas normas de TV Digital, para a incorporação deste método de compressão de vídeo, poderia ser automática, caso a função de decodificação de vídeo fosse reconfigurável. Nesse caso, os receptores acompanhariam a evolução destes métodos e poderiam se adaptar ao conteúdo transmitido pelas emissoras de TV digital.

Os dados de reconfiguração (feixe de bits) do FPGA poderiam ser inseridos no receptor, utilizando-se a porta USB ou também através da comunicação de rede. Os dados também poderiam ser armazenados no sistema de arquivos do receptor e, após isso, um gerenciador de reconfiguração controlaria o uso desses núcleos.

Outra forma interessante de enviar esses dados é através do próprio sinal de TV digital. Nesse caso, os dados seriam transmitidos pela emissora, juntamente com o áudio, o vídeo e os demais dados do sinal. Assim, todos os receptores ao alcance do sinal de transmissão seriam reconfigurados, a partir de um único meio (Figura 2). Esta última opção pode ser viabilizada com o uso da metodologia apresentada neste trabalho, que consiste na sinalização/transmissão do feixe de bits, para que este seja capturado por unidades receptoras. Essas, por sua vez, devem estar preparadas para reconhecer o conteúdo sinalizado, verificar se o conteúdo transmitido foi sintetizado para o dispositivo que está instalado no receptor, realizar a remontagem do/s núcleo/s de hardware e finalmente reconfigurar os respectivos FPGAs. Para a transmissão de núcleos de

hardware a receptores de TV digital, é necessário estabelecer uma metodologia que viabilize todo o processo, desde a transmissão até a recepção de dados. A metodologia deve contemplar a manipulação de dados para a transmissão, a recepção completa e a estrutura necessária para a reconfiguração do dispositivo FPGA. Os dados de reconfiguração precisam ser sinalizados (criação de uma nova tabela), ou seja, identificados e descritos no fluxo de bits do sinal de transmissão. Esse passo é crucial, tendo em vista a necessidade de se contextualizar o conteúdo que está sendo inserido.

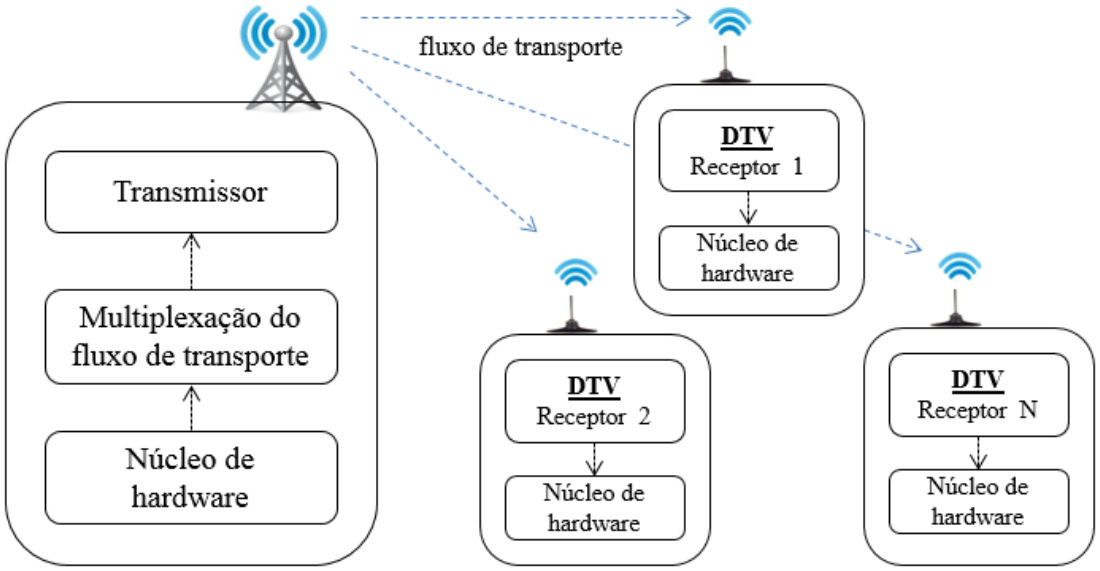

Figura 2 – Núcleo de hardware transmitido para receptores no alcance do sinal.

Tal processo permitirá que as unidades receptoras identifiquem e apliquem/descartem os dados recebidos. Tal identificação é necessária, pois diferentes tipos de dados são transmitidos e contidos no mesmo canal de transmissão. O descarte de dados pode ser necessário, caso a descrição enviada não seja compatível com o dispositivo FPGA da unidade receptora. Essa verificação é realizada a partir das informações do processo de síntese lógica do feixe de bits. Dentre essas informações, encontra-se a identificação do FPGA, incluindo nome do fabricante, família e número de série, com o objetivo de identificar o dispositivo para o qual o núcleo de hardware foi sintetizado. Essas informações são cruciais, considerando-se que receptores comerciais podem utilizar FPGAs de diversos fabricantes.

No ambiente de TV Digital, a sinalização de conteúdo é realizada através de informações inseridas em tabelas, que contém a descrição do conteúdo presente no sinal de transmissão. Tais informações mostram como extrair, do sinal de TV digital (fluxo contínuo de bits), pacotes de

áudio, vídeo, sinais de sincronismo, etc. Com a tabela de descrição do feixe de bits, o receptor já pode detectar a sinalização, extrair o feixe de bits e então realizar a reconfiguração do dispositivo FPGA. Para isso, o receptor deve estar preparado para efetuar a validação e a remontagem completa dos feixes de bits, até a reconfiguração.

1.2 Contribuições

A metodologia de reconfiguração de hardware através do sinal de TV digital, proposta neste trabalho, gera contribuições significativas às áreas de sistemas embarcados, reconfiguração de hardware e TV digital.

A concepção desta metodologia é uma das principais contribuições deste trabalho. A reconfiguração de hardware já é um procedimento consolidado e utilizado em sistemas embarcados e produtos de eletrônica de consumo. Aqui, propõe-se uma maneira de atualizar hardware de receptores, através de uma das principais características de um ambiente de TV Digital, que é a capacidade de transmissão de dados. Nesta metodologia, os dados de reconfiguração de hardware são contextualizados, permitindo que as unidades receptoras possam identificar este conteúdo, dentro de um fluxo contínuo de bits do sinal de TV Digital. A partir dessa identificação, é possível extrair e aplicar os dados obtidos, no contexto da reconfiguração de hardware.

Esta metodologia estende a norma SI em geral (*digital video broadcasting* (DVB) [27] e *integrated services digital broadcasting* (ISDB) [28]), com a definição de uma nova tabela: a *update information table* (UIT), que contém uma sintaxe composta por uma série de campos e descritores e permite sinalizar e descrever as características do núcleo de hardware multiplexado aos demais conteúdos de HDTV. Através desta sinalização, as unidades receptoras poderão identificar a existência de conteúdo de reconfiguração, no fluxo do sinal de TV, e decidir, com base nas características dispostas na sintaxe dos descritores da UIT, se o núcleo de hardware pode ser utilizado pela unidade receptora. Isso é possível porque a metodologia permite distinguir entre núcleos enviados para diferentes modelos de FPGA.

O uso desta metodologia propicia também uma "forma mais eficaz de atualização do hardware de receptores de TV Digital", em campo, considerando que uma atualização desta natureza também poderia ser realizada a partir de outros meios, como portas USB ou conexões de rede. Nesse caso, seria necessário acionar um serviço autorizado e técnicos especializados, o que envolveria custos adicionais. Assim, os fabricantes poderiam disponibilizar o arquivo de atualização para a rede autorizada, que ficaria responsável pela centralização e distribuição das

atualizações. No entanto, em uma atualização através do sinal de TV Digital, todos os receptores ao alcance da emissora seriam atualizados. A emissora poderia ficar transmitindo dados de reconfiguração ciclicamente, durante um certo período de tempo, permitindo assim a reconfiguração de receptores.

Esta metodologia também pode propiciar a criação de novos paradigmas de transmissão e recepção de dados. Dessa forma, os padrões de TV digital poderiam transmitir seus decodificadores, associados ao tipo de mídia a ser decodificado. Assim, os canais de TV digital poderiam se tornar flexíveis, adaptando-se ao tipo de mídia que está presente no sinal. Isto possibilitaria que os avanços nas técnicas de decodificação fossem incorporados ao padrão de TV digital.

Os experimentos, as métricas e os resultados relacionados à transmissão e recepção de dados, na área de TV Digital, são escassos. Sendo assim, este trabalho pode contribuir bastante com os resultados reais apresentados. Os fluxos de transporte com hardware pré-sintetizado foram gerados durante a realização de provas conceituais, que foram utilizadas para a validação da metodologia proposta. As métricas definidas para a avaliação do transporte de dados de reconfiguração de hardware, assim como o ambiente experimental, necessário para a realização de testes e experimentos, também poderão servir como base para experimentos futuros, que sejam desenvolvidos como continuidade deste trabalho.

1.3 Contexto e Motivações

O assunto desta tese de doutorado está diretamente relacionando à área de sistemas embarcados, envolvendo também a área de tecnologia de TV digital e reconfiguração de hardware. A ideia do tema surgiu a partir de resultados obtidos com testes de transmissão de dados realizados com um equipamento de radiodifusão terrestre. O objetivo desses testes era efetuar a transmissão de uma aplicação binária, compilada para um receptor específico, e posteriormente recuperá-la, em uma unidade receptora. Nesta, a aplicação deveria ser apropriadamente remontada e executada, com o objetivo de se verificar se a recepção havia sido bem-sucedida. Ao se transmitirem aplicações que utilizassem uma nova mídia, tal evolução esbarraria nos recursos disponíveis em hardware. Dessa forma, software *e* hardware deveriam ser substituídos. Assim, a arquitetura de hardware do receptor deveria usar um dispositivo de lógica programável, para abrigar a nova função de hardware.

Surgiu, então, a ideia de transmitir o núcleo de hardware a o receptor, usando o mesmo princípio utilizado para a aplicação. No entanto, a transmissão de um novo conteúdo, no

ambiente de TV digital, necessitaria de uma contextualização, para que o receptor identificasse o propósito desses dados. Além disso, existem diversos fabricantes de FPGA e cada um deles possui extensas famílias e diferentes modelos de FPGAs. Nesse caso, além do núcleo de hardware, também seria necessário sinalizar e indicar o tipo a ser transmitido. A metodologia também deveria propiciar uma forma de enviar as características do núcleo de hardware, para que o receptor soubesse como tratar este núcleo. Surgiu, então, a necessidade de se desenvolver uma metodologia que envolvesse o processo de transmissão, recepção e reconfiguração do FPGA.

Os receptores de TV digital têm diversos tipos de tarefas, que podem ser desenvolvidas em software ou hardware. No entanto, algumas destas, as consideradas mais críticas, precisam ser desenvolvidas em hardware. Um exemplo é à decodificação de vídeo, que exige um maior poder de processamento. Assim, existe a necessidade de se utilizar um hardware específico, deixando o processador livre para as demais funções desempenhadas pelo receptor. Esta necessidade, por sua vez, cria uma dependência de hardware (legado), que não permite a incorporação de avanços ao processo de decodificação de vídeo, sem a substituição direta da unidade receptora.

Do mesmo modo, outras tarefas também geram uma dependência física do hardware, o que poderia ser resolvido com o uso de dispositivos reconfiguráveis. Com estes, o hardware dos receptores de TV digital poderia ser reconfigurado, permitindo assim a implantação de novos métodos de processamento, sem considerar a troca imediata do equipamento. Com isso, seria possível prolongar o ciclo de vida dos receptores, mantendo o hardware constantemente atualizado ou inserindo avanços tecnológicos, através do processo de reconfiguração.

Outro fator motivador deste trabalho é o desafio de se enviar informação de reconfiguração através do sinal da emissora, permitindo assim que receptores, ao alcance do sinal de transmissão, possam reconfigurar seus componentes de hardware, o que evita que o dispositivo seja levado até um ponto de atualização. Com o uso desta metodologia, há uma grande redução nos custos de manutenção e suporte.

1.4 Objetivos Gerais e Específicos

Objetivo Geral

Desenvolver uma metodologia para transmissão e recepção de dados de reconfiguração de hardware, com o objetivo de reconfigurar módulos em hardware de um receptor, com base no sinal da transmissora de TV.

Objetivos específicos

- Adequar o sistema de transmissão de dados de TV digital para realizar a sinalização, a identificação e a inclusão de dados de reconfiguração de hardware, juntamente com os demais conteúdos de HDTV;
- Desenvolver uma metodologia para identificação, validação e remontagem de dados de reconfiguração de hardware;
- Adequar o sistema residente do receptor, com o objetivo de realizar uma comunicação entre a unidade receptora (*host*) e o dispositivo FPGA (*target*).

1.5 Organização do trabalho

Este trabalho está organizado como segue. O Capitulo 2 discute alguns trabalhos na mesma área da metodologia proposta. O Capítulo 3 apresenta o referencial teórico, que contempla uma revisão bibliográfica de conceitos relacionados à tecnologia de TV Digital e aos mecanismos de transmissão de dados (*datacasting*), uma comparação entre os mecanismos de transmissão de dados disponíveis no SBTVD, a codificação de canal e a modulação no SBTVD, uma descrição das arquiteturas de hardware e software das unidades receptoras de TV Digital, os conceitos básicos relacionados aos dispositivos programáveis e à tecnologia de FPGA e a síntese de hardware e os modos existentes, utilizados em esquemas de reconfiguração do FPGA. Os detalhes da metodologia de reconfiguração de hardware introduzida nesta tese de doutorado, assim como a prova de conceito e a execução de todas as etapas desta metodologia, são mostrados no Capitulo 4. Resultados de experimentos, baseados na metodologia proposta, são discutidos no Capitulo 5. No capítulo 6, as considerações finais e sugestões para trabalhos futuros são apresentadas.

2. Trabalhos Relacionados

A literatura apresenta pesquisas com arquiteturas reconfiguráveis e sistemas embarcados inteligentes, tendo a reconfiguração de hardware como núcleo para construção de sistemas mais compactos e eficientes.

Arquiteturas de sistemas embarcados, com gerenciamento de recursos de hardware, representam uma nova tendência no uso da reconfiguração de hardware. Nessas arquiteturas, funções unitárias de hardware pré-sintetizadas podem ser carregadas em um FPGA, à medida que o usuário interage com o sistema [29]. Aqui, a alocação dos módulos unitários de hardware permite a reconfiguração parcial de um FPGA, em tempo de execução. Os módulos respeitam um dado critério de alocação e são mantidos em uma fila, podendo ser carregados a partir de um gerenciador de recursos de hardware. Esse mecanismo pode melhorar a eficiência de um sistema embarcado, pois pode decidir entre executar uma tarefa em software ou hardware.

Sistemas embarcados inteligentes fazem uso de reconfiguração dinâmica parcial de hardware para reconfigurar módulos de hardware, em tempo de execução. Essa característica, similar ao sistema multitarefa de um microprocessador, permite a multiplexação de diferentes funções de hardware. Assim, os blocos de hardware funcionais (blocos lógicos) do FPGA podem ser carregados (reconfigurados) ou descarregados, de acordo com as necessidades do sistema [30]. Nesse caso, com o uso da reconfiguração dinâmica parcial, houve uma redução significativa no consumo de potência e no custo final do dispositivo.

A arquitetura reconfigurável de fluxo contínuo, apresentada por Hillenbrand [31], permite a inclusão de núcleos de processamento digital de sinais (*digital signal processing* - DSP) pré-sintetizados em código HDL, que podem ser reconfigurados em tempo de execução. Nesse caso, a utilização de núcleos pré-sintetizados minimiza o processo de síntese e permite a construção de arquiteturas com sistemas reconfiguráveis adaptativos.

Uma plataforma com a possibilidade de uso de múltiplos núcleos pré-sintetizados, gerenciados por uma unidade de processamento, proporciona um rico e flexível ambiente para programadores de aplicações, como proposto por Serres [32]. Este tipo de abordagem é chamado de arquitetura mista, no qual unidades programáveis são gerenciadas por processadores que controlam a reconfiguração de diferentes núcleos pré-sintetizados de hardware, de acordo com a demanda do sistema.

2.1 Discussão envolvendo o trabalho proposto

A principal ideia deste trabalho consiste no modo de envio do hardware pré-sintetizado. A combinação entre esta metodologia e os projetos já citados pode impulsionar a área de pesquisa em sistemas de arquiteturas reconfiguráveis, resultando em inúmeras abordagens de reconfiguráveis. Os trabalhos relacionados utilizam núcleos pré-sintetizados e realizam a reconfiguração em tempo de execução. Logo, a principal diferença é que estes mantem uma lista de núcleos pré-sintetizados, previamente armazenados no sistema de arquivos do dispositivo, e o sistema embarcado, de forma automática, gerencia a alocação dos núcleos para reconfiguração. A abordagem proposta, no entanto, pode transmitir diferentes núcleos pré-sintetizados a uma quantidade muito grande de dispositivos, que podem ser automaticamente reconfigurados.

Além disso, o esquema proposto poderia transmitir núcleos pré-sintetizados, de diversos fabricantes, e cada dispositivo seria responsável por aceitar ou rejeitar esse conteúdo. Nesse caso, os receptores poderiam utilizar dispositivos reconfiguráveis e modelos de FPGA de diferentes fabricantes. Outra funcionalidade interessante desta metodologia é que a emissora pode enviar os dados de reconfiguração de hardware ciclicamente, durante um período de tempo, dando oportunidade para que todos os receptores, ao alcance do sinal de TV Digital, sejam reconfigurados. Assim, em um ambiente de TV digital com receptores baseados em troca de módulos de hardware, os avanços que ocorressem na tecnologia, assim como o aperfeiçoamento das técnicas existentes, poderiam ser incorporados ao sistema, criando um ambiente completamente flexível.

É possível estender o uso desta metodologia a qualquer dispositivo que utilize MPEG-TS como fonte de sinal. Dispositivos móveis, como telefones celulares, poderiam aplicar esta metodologia à criação de arquiteturas e sistemas reconfiguráveis que, de alguma forma, pudessem auxiliar no desempenho e na redução dos custos de projeto. Decodificadores de TV a cabo, baseados em MPEG2-TS, podem aplicar esta metodologia à criação de dispositivos mais robustos, através de técnicas de criptografia por hardware, possibilitando um controle maior dos usuários.

A ideia de transmissão de hardware pré-sintetizado, apresentada neste trabalho, também pode ser adequada a sistemas empregados em comunicação de dados. Espera-se que esta metodologia também possa servir de referência, de modo que a reconfiguração de hardware seja utilizada no desenvolvimento de arquiteturas e sistemas embarcados adaptativos, que empreguem a reconfiguração de hardware como núcleo e possam se beneficiar da quebra do legado causada por ASICs.

3. Referencial Teórico

Este capítulo descreve conceitos básicos necessários à compreensão do tema proposto. Aqui, conceitos relacionados à tecnologia de TV Digital, ao SBTVD, às técnicas de difusão de dados oferecidas pelo padrão, à reconfiguração de hardware, aos modos de interfaceamento utilizados para controle do FPGA e às características do receptor utilizado neste trabalho serão abordados.

3.1 TV Digital

Os sistemas de TV Digital são baseados em um conjunto de normas e regulamentações, que padronizam o desenvolvimento dos diversos subsistemas de um padrão de TV digital. A regulamentação é necessária para que tanto os fabricantes de receptores/transmissores quanto emissoras possam desenvolver suas soluções, tomando por base uma referência unificada. No início dos anos 90, um consórcio de empresas de eletrônicos dos Estados Unidos (*grand aliance*) desenvolveu o padrão *advanced television systems committee* (ATSC) [33]. O objetivo deste comitê era proporcionar um sistema totalmente digital, que seria capaz de substituir o então sistema analógico *national television system committee* (NTSC). Logo em seguida, este padrão também foi adotado pela maioria dos países da América do Norte.

Com o decorrer do tempo, o ATSC sofreu algumas adaptações, como o *advanced television systems committee - mobile/handheld* (ATSC-M/H) [33], que permite que o sinal de TV digital também seja transmitido para os dispositivos móveis. Em 1993, a Europa, através de um grupo de empresas de eletroeletrônicos, começou a especificar uma família de normas para o um sistema de televisão interativa. Através dessas normas, o sistema europeu (DVB) [34] foi desenvolvido. O padrão DVB-T entrou em operação em meados de 1998 e posteriormente foi adotado por diversos outros países, conforme demonstrado na Figura 3.

Após o desenvolvimento dos padrões americano e europeu, o Japão, através de um grupo chamado *digital broadcasting experts group* (DIBEG), especificou e desenvolveu o sistema de radiofusão japonês, chamado ISDB [28]. Esse grupo é composto por membros do Conselho do Ministério das Telecomunicações do Japão e pelo grupo *association of radio industries and businesses* (ARIB). O padrão ISDB fornece serviços de áudio, vídeo e dados multimídia. Esse sistema entrou em operação, no Japão, em meados de 2003. Como os demais padrões, esse sistema é constituído por um conjunto de documentos que especifica normas, de acordo com os métodos de transmissão, modulação e compressão. Uma característica que destaca este sistema é a flexibilidade de recepção por dispositivos móveis.

Em 1998, a associação brasileira de emissoras de rádio e televisão (ABERT) e a Universidade Presbiteriana Mackenzie realizaram uma pesquisa com os padrões de TV digital existentes (ATSC, DVB e ISDB), considerando características geográficas, políticas e sociais do território brasileiro. Em meados de 2006, através de um decreto presidencial (Nº 5820), definiu-se que o sistema japonês ISDB seria a base do sistema terrestre de TV digital brasileiro. Esta decisão levou em conta a adaptação técnica do sistema à geografia do Brasil, a capacidade de interatividade e a inclusão digital da população do Brasil. O Brasil realizou algumas adaptações no padrão, substituindo o método de compressão de vídeo MPEG-2 pelo MPEG-4 AVC e definindo o MPEG-4 AAC como método de compressão de áudio. O Brasil também adicionou o *middleware* ginga (camada de interoperabilidade para a execução de aplicativos interativos, que abstrai o hardware e o sistema operacional do receptor), que havia sido desenvolvido por duas universidades brasileiras (universidade federal da paraíba -- UFPB e universidade federal do rio de janeiro -- UFRJ). O SBTVD foi adotado por grande parte dos países da américa do sul, as suas normas são revisadas constantemente e também podem sofrer alterações, como a inclusão de melhorias ou modificações com o intuito de solucionar problemas, que podem aparecer com o uso do sistema. O padrão SBTVD suporta transmissão a receptores fixos, no modo Full-Seg[2], e também a dispositivos móveis, no modo One-Seg[3].

Por fim, o padrão *digital terrestrial multimedia broadcast* (DTMB) foi desenvolvido e especificado pela China, com o objetivo de atender receptores terrestres móveis e fixos, e foi adotado por China, Hong Kong e Macau. Uma característica deste padrão é o uso de vídeo MPEG-4 AVC e áudio AAC, de modo similar ao SBTVD.

Os padrões de TV digital são distintos, no entanto, uma característica comum a todos eles, é a adoção dos métodos de compressão de áudio [36], vídeo [37] e transporte de dados criados pelo MPEG. As normas MPEG foram desenvolvidas por um grupo de trabalho da *international standards organization* (ISO), com o objetivo de padronizar a compressão de vídeo e áudio digitais, bem como a sincronização de eventos e o empacotamento de dados multimídia [38].

---

[2] A classificação *Full-Seg* é aplicada aos conversores digitais, também conhecidos por *set-top box*, e aos receptores de 13 segmentos integrados com tela de exibição, mas não exclusivos a estes. Este tipo de receptor é capaz de receber e decodificar sinais de televisão digital terrestre de alta definição e, a critério do fabricante, também receber e decodificar informações transportadas na camada "A" do fluxo de transporte.

[3] A classificação *One-Seg* é destinada aos receptores do tipo portátil, também conhecidos por "*handheld*", especialmente recomendados para telas de exibição de dimensões reduzidas, normalmente até 7 polegadas. Entre os produtos classificados como *one-seg* estão os receptores integrados com telefone celular, PDA, *dongle* e televisores portáteis. Este tipo de receptor é capaz de receber e decodificar apenas sinais de televisão digital terrestre transportado na camada "A" do fluxo de transporte e, consequentemente, apenas sinais de perfil básico, destinado aos dispositivos portáteis de recepção.

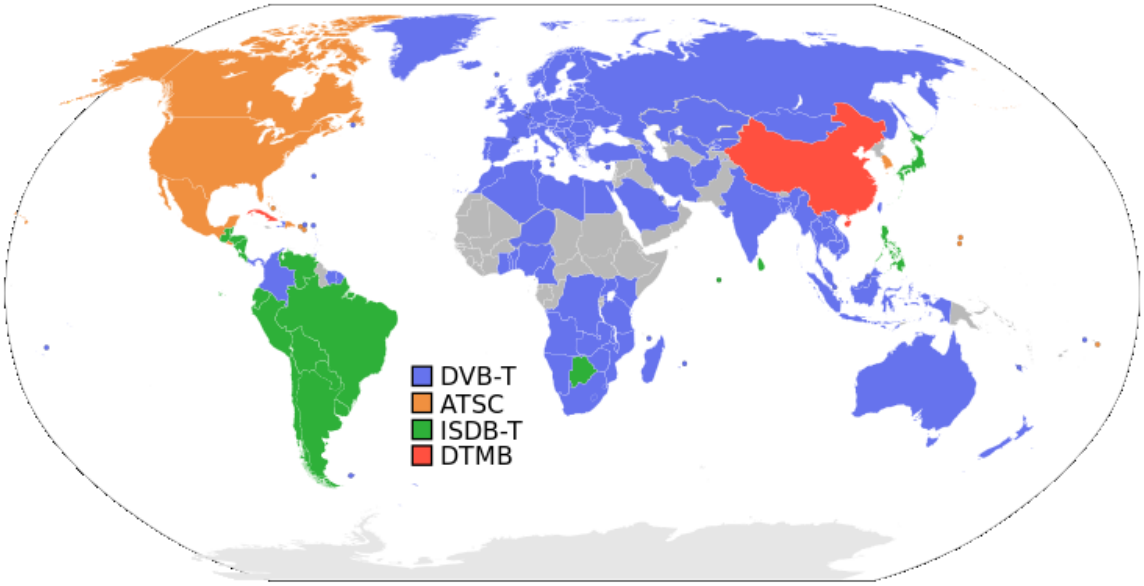

Figura 3 – Padrões de TV digital adotados pelos países, Fonte [35].

A camada de sistemas do MPEG2 consiste em um conjunto de definições, contendo informações de multiplexação de sinais. Essas definições permitem combinar sinais de áudio, vídeo e dados em um fluxo de transporte (TS). A geração do sinal de transmissão, no sistema SBTVD, consiste basicamente na codificação/empacotamento de vídeo e áudio, multiplexação MPEG-2 sistemas e codificação de canal/modulação, conforme apresentado na Figura 4.

Para a decodificação de vídeo, o padrão SBTVD utiliza o MPEG-4 AVC, também conhecido como MPEG-4 camada 10 ou H.264. Os algoritmos de compressão MPEG-2 e MPEG-4 são capazes de eliminar redundâncias contidas no sinal de vídeo, reduzindo o espaço ocupado pelo vídeo original. O papel do codificador de vídeo é a compressão dos dados de vídeo, para adequá-los à largura de banda do espectro de televisão, que é da ordem de 6MHz.

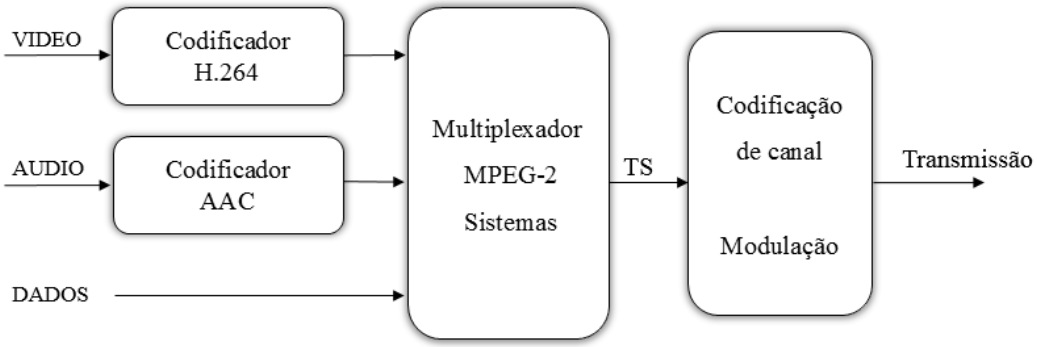

Figura 4 – Diagrama de formação do sinal de TV digital do SBTVD (Fonte [22]).

Assim como o vídeo, o áudio também é codificado. O padrão SBTVD utiliza a codificação de áudio avançada AAC [39]. Após essa codificação, os sinais de áudio e vídeo são empacotados em um fluxo de transporte, definido na norma ARIB STD-B32 [40, 41]. Além de áudio e vídeo, a área de carga dos pacotes *packetized elementary streams* (PES) também é utilizada para transportar dados [42]. O TS segue então para a codificação de canal, que reforça o sinal inserindo dados auxiliares, que aumentam a robustez do sinal. Após isso, o sinal é modulado e transmitido, através de um transmissor OFDM.

3.1.1 Multiplexação de Sinais

Na seção anterior, foi mencionado que fluxos elementares de áudio e vídeo são multiplexados, formando um único feixe de dados, antes da etapa de codificação de canal e modulação.

A norma MPEG-2 Sistemas [38] define uma estrutura para o fluxo de transporte, a partir da multiplexação de fluxos elementares de um ou mais programas, que são compostos por pacotes com tamanho fixo de 188 bytes. Um fluxo de transporte MPEG-2 pode transportar diversos programas ao mesmo tempo e também deve disponibilizar as informações necessárias para a decodificação destes, de modo que o receptor possa selecionar o programa que vai ser decodificado. A Figura 5 demonstra como múltiplos programas são multiplexados em um único fluxo de transporte.

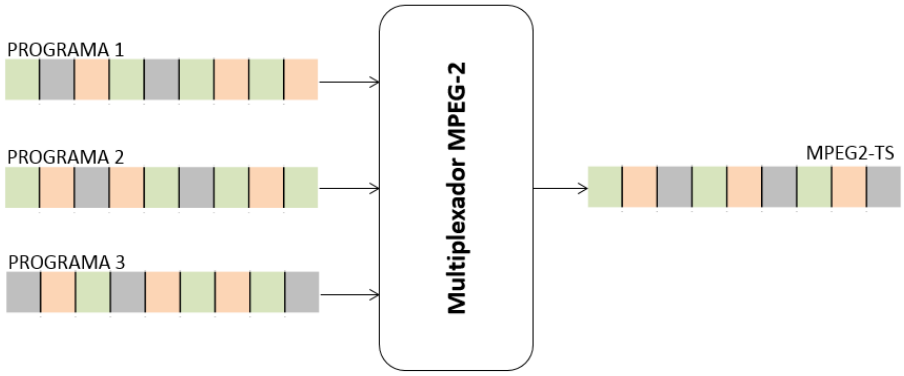

Figura 5 – Programas multiplexados em um único fluxo de transporte MPEG-2.

Programas são compostos por pacotes de vídeo, áudio e dados. Além disso, eles também são multiplexados juntamente com outras informações, como as tabelas contendo informações do conteúdo do fluxo de transporte. A tabela *program map table* (PAT) é o ponto de entrada do fluxo, é identificada pelo valor (0x00) e endereça tabelas *program map table* (PMT) associadas aos programas existentes. As PMTs, contém o *packet identifier* (PID) de cada fluxo elementar inserido no programa. É através do PID que os receptores conseguem selecionar o fluxo

elementar que desejam decodificar. Cada tabela PMT endereça um determinado programa. Além destas informações, pacotes nulos podem ser adicionados durante o processo de multiplexação, com o intuito de adequar a taxa do fluxo de transporte ao canal de transmissão.

3.1.2 Fluxo de transporte MPEG-2

O fluxo de transporte pode ser dividido em pacotes de transporte, que representam a unidade básica do fluxo de transporte. Pacotes de transporte são divididos em 3 partes (Figura 6). O cabeçalho (*Header*) contém a descrição do conteúdo. Com as informações contidas em cada cabeçalho, é possível manter o sincronismo de leitura dos pacotes e também identificar o conteúdo da área de carga elementar (*payload*). Entre os campos do cabeçalho, estão o byte de sincronismo (*sync byte*), cujo valor é 0x47 e é utilizado para indicar o início do fluxo de transporte e manter o sincronismo de leitura [43], 3 campos de 1 bit cada (*transport error indicator*, *payload unit start indicator* e *transport priority*) e os 13 bits do identificador de pacote (PID). O PID é um número utilizado para se identificar, no TS, quais pacotes correspondem a vídeo, áudio, informações de sistema (tabelas) e dados. O campo de adaptação (*adaptation field*) deve obrigatoriamente ser utilizado para transmitir informações adicionais do cabeçalho, cuja presença é opcional. Os campos de cargas elementares (*payload)* devem obrigatoriamente ser utilizados para transmitir fluxos empacotados em PES ou seções [43]. PES e/ou seções são multiplexados e distribuídos em pacotes, ao longo do fluxo de transporte, podendo estar espaçados no tempo. Durante o processo de demultiplexação de sinal, módulos específicos dos receptores de TV digital utilizam o PID extraído do cabeçalho do pacote, que permite que o demutiplexador selecione o pacote desejado. Os fluxos extraídos de cada pacote são então direcionados, em um processo contínuo, para os seus respectivos decodificadores, de modo que sejam tratados pelos sistemas residentes presentes nos receptores.

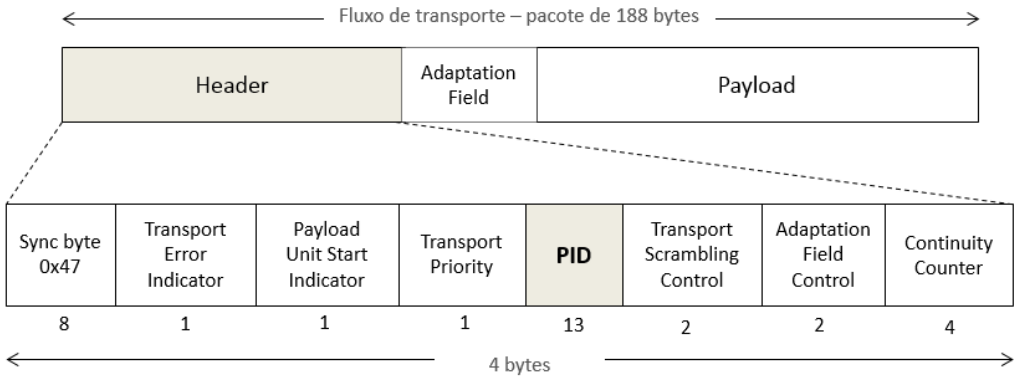

Figura 6 – Composição do pacote que formam o fluxo de transporte (Fonte [43]).

As tabelas que descrevem o conteúdo do fluxo de transporte são as primeiras a serem filtradas pelos sistemas residentes, considerando que elas informam os pontos de acesso ao conteúdo dos pacotes. Assim, cada informação presente no fluxo de transporte recebe um tratamento específico, de acordo com o contexto em que os dados devem ser inseridos.

3.1.3 Tabelas SI/PSI

A camada de sistemas MPEG-2 [38] é responsável pela integração e sincronização dos fluxos elementares de áudio e vídeo. Nessa camada, outros dados também são inseridos, assim como sinais de controle utilizados pelas várias aplicações do ambiente de TV digital. Dentre esses dados, estão as tabelas de serviço de informação (*service information* – SI) e as tabelas de informação específicas de programa (*program specific information* – PSI), que descrevem o conteúdo dos programas/serviços do fluxo de transporte. As tabelas PSI devem obrigatoriamente conter informações que permitam a configuração do receptor, de modo que este possa demultiplexar e decodificar os vários programas multiplexados no sinal de TV digital [38]. Já as tabelas SI fornecem informações sobre os serviços disponíveis, como frequência usada para transporte, categorias em que se inserem os serviços, eventos de cada serviço e informações de rede, com o objetivo de alimentar o guia eletrônico de programação (*electronic program guide* - EPG) e outras funções. As tabelas PSI (Figura 7) são divididas em: PAT, PMT, *network information table* (NIT) e *conditional access table* (CAT). A tabela PAT mostra os programas que o stream de transporte está carregando, listando todos os PIDs das tabelas PMTs associadas a cada programa, com o PID do programa ‘0’ dando acesso à tabela NIT, que contém as informações da rede. A tabela NIT é responsável por informar a organização física do agrupamento de multiplexadores/fluxos de transporte existentes em uma mesma rede e as suas características, assim como todo dado relevante sobre a sintonia dos serviços existentes. A tabela PAT é identificada pelo PID 0x00 e normalmente é a primeira tabela a ser filtrada pelos receptores de TV digital [38]. O PID 22 aponta para uma tabela PMT, que descreve o programa 1, e o PID 33 aponta para outra PMT, que contém informações sobre o programa 2. As PMTs contêm os PIDs dos fluxos elementares de áudio e vídeo e dados que estão presentes no TS. Os pacotes carregados com feixes elementares de áudio e vídeo, sinalizados por suas respectivas PMTs, são distribuídos ao longo do tempo, no feixe de transporte, como mostrado na Figura 7.

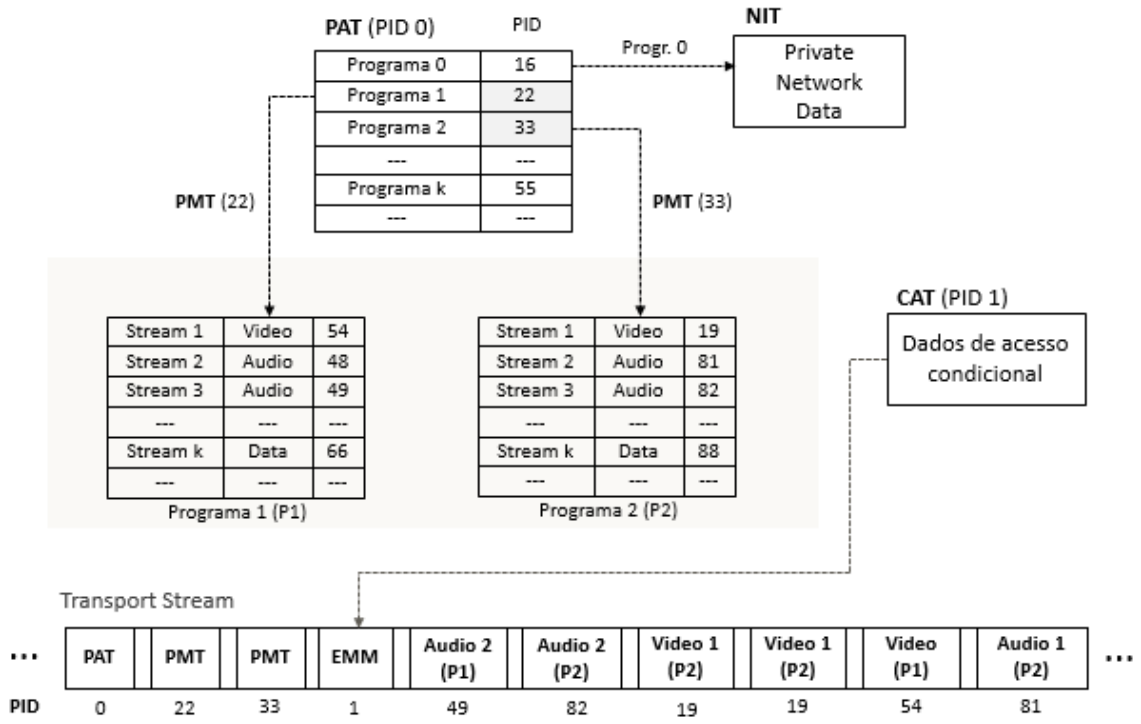

Figura 7 – Tabela PAT referenciando PMTs e NIT (Fonte [38]).

O programa descrito pela PMT com PID 22 contém um fluxo elementar de vídeo com PID 54 e de áudio: o áudio 1 (*stream* 2), com PID 48, e o áudio 2 (*stream* 3), com PID 49. A PMT com PID 33 descreve o programa 2, que é composto por um feixe elementar de vídeo, com PID 19, e dois feixes de áudio, com PIDs 81 e 82. A tabela CAT fornece informações para os sistemas de acesso condicional, ou seja, elementos *entitlement management message* (EMM[4]) individuais multiplexados [38]. É possível observar, no fluxo de transporte da Figura 7, a distribuição das tabelas em pacotes de transporte identificados pelos seus respectivos PIDs.

As normas de TV digital determinam que as tabelas que descrevem o conteúdo dos programas sejam transmitidas em caráter obrigatório. No entanto, outras tabelas opcionais, utilizadas para outros propósitos, também podem estar presentes no sinal de TV digital (Figura 8). A norma ABNT NBR 15603-2 [44] define, entre as tabelas PSI/SI existentes, as que precisam ser transmitidas em caráter obrigatório e as que podem ser transmitidas em caráter opcional. As tabelas que descrevem o conteúdo dos programas devem ser transmitidas obrigatoriamente, conforme as normas de TV digital. Além das 4 tabelas apresentadas

[4] EMM é usado para enviar os direitos para o dispositivo de controle de acesso condicional do Set-top Box. O STB vai receber os EMMs e encaminhá-los para o cartão inteligente para processamento. O cartão inteligente usa as informações nos EMMs para atualizar seu banco de dados de controle de acesso interno contendo uma lista de canais e filmes em *vídeo on demand* (VoD) que o usuário tem permissão para assistir.

anteriormente (PAT, PMT, NIT e CAT), a tabela de descrição de serviços (*service description table* – SDT), a tabela de informação de eventos (*event information table* -- EIT) e a tabela de mudança de data e horário (*time offset table* -- TOT) também devem ser transmitidas obrigatoriamente [44].

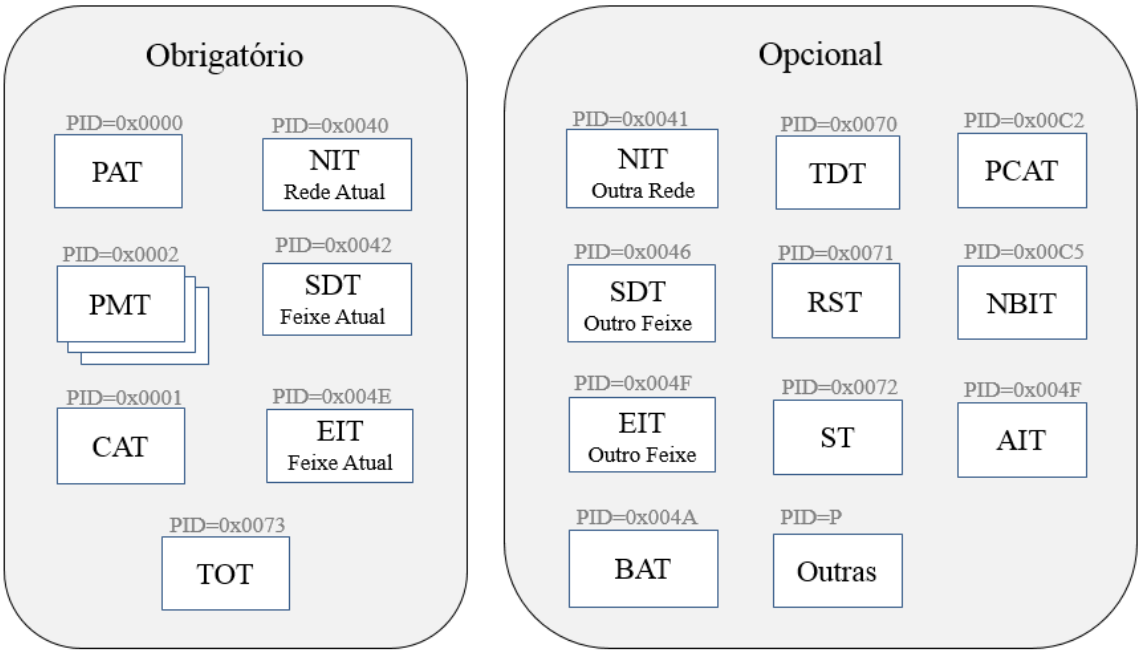

Figura 8 – Lista das tabelas PSI/SI e seus respectivos identificadores PID.

A tabela SDT informa os serviços existentes em um fluxo de transporte. A tabela EIT deve obrigatoriamente prover informações, em ordem cronológica, sobre os eventos existentes, por serviço. Por fim, a tabela TOT é responsável por informar ao receptor a hora, a data e o fuso horário.

Algumas tabelas de envio obrigatório também são de uso opcional, como as tabelas NIT, SDT e EIT, com informações sobre o feixe presente em outros TSs. As demais tabelas de uso opcional, como a tabela de data e horário (*time and date table* – TDT), a tabela de anúncio de conteúdo parcial (*partial content announcement table* – PCAT), a tabela de associação de buquê (*bouquet association table* –BAT), a tabela de informação de grupo da rede (NBIT), a tabela de preenchimento (*stuffing table* – ST), a tabela de referência a outras tabelas (*local descriptor table* – LDT), a tabela de informação de aplicação (*application information table* – AIT) e outras tabelas podem ser utilizadas para descrever diferentes conteúdos em um fluxo de transporte [44].

Conforme foi apresentado, as tabelas do sistema de DTV sinalizam e possibilitam o acesso a todo o conteúdo presente na área de carga dos pacotes de transporte. Além do mecanismo de

sinalização, técnicas de difusão de dados são utilizadas para o envio de informações aos receptores de TV digital.

3.1.4 Técnicas de transmissão de dados

Uma das principais propriedades de um sistema de TVD é a difusão de dados (*datacasting ou data broadcasting*). O *datacasting* permite que transmissoras forneçam serviços de entrega de informação a usuários que utilizam receptores adequados ao seu recebimento, desde que estejam localizados na sua área de cobertura.

O MPEG2-TS é o padrão de multiplexação utilizado pelos sistemas de TVD abertos. Dessa forma, os mecanismos de transmissão de dados na TVD fazem uso das facilidades e estruturas desse padrão para efetuar a transmissão de dados. Existem basicamente quatro tipos de mecanismos de difusão: *data piping*, *data streaming*, *multiprotocol encapsulation* e o *carousel*, sendo esse último dividido em *data carousel* e *object carrousel*.

3.1.4.1 Data Piping

O método de transmissão de dados *data piping* é o mecanismo mais simples de *datacasting*. Ele é baseado na inserção de dados brutos, diretamente na carga dos pacotes de transporte [45]. A tarefa de um encapsulador de dados desse tipo é gerar pacotes de transporte com um valor de PID conhecido pela aplicação destino, inserindo dados nos 184 bytes da carga de cada pacote, antes de enviá-los ao multiplexador. Durante o processo de recepção dos dados, o receptor precisa apenas filtrar pacotes com o valor de PID utilizado e extrair, de suas cargas, os dados transmitidos [45]. Esse mecanismo é utilizado para o transporte de dados assíncronos, não proporcionando nenhum mecanismo de temporização. O método tem como principal vantagem a simplicidade na carga de dados. Entre as desvantagens, a principal é que esta técnica não oferece nenhum mecanismo de detecção de erros.

3.1.4.2 Data Streaming

Esse mecanismo oferece suporte a serviços que requerem transmissão de dados síncronos ou assíncronos [46]. Nele, os dados são continuamente empacotados em pacotes PES, antes de serem difundidos. O suporte à apresentação de dados, de maneira síncrona, é oferecido com o uso de marcadores de tempo, no cabeçalho do PES: o *presentation time stamp* (PTS), que indica o instante em que a apresentação deve ser realizada, e o *decode time stamp* (DTS), que indica o instante de término da decodificação [46]. Nessa técnica, também é possível utilizar seções

privadas para se efetuar o encapsulamento dos dados [46]. A sintaxe das seções privadas (ver ANEXO-I) oferece um meio de manter a contiguidade física entre seções, que é fundamental para permitir a remontagem destas no mesmo formato em que foram carregadas, no mecanismo de transporte. O mecanismo de *data streaming* também é uma forma de envio de dados com baixo *overhead* (dados auxiliares e de cabeçalho). As seções permitem o uso de dois tipos de cabeçalhos: o simples e o estendido. Através da configuração de cabeçalho simples, são necessários apenas 3 bytes para cada sequência de 4093 bytes de dados; utilizando-se o cabeçalho estendido das seções privadas, é possível transmitir 4084 bytes de dados, por seção.

Este mecanismo é indicado para o transporte de dados síncronos e assíncronos e o seu uso é mais adequado a dados difundidos para todos os receptores (*broadcast*), de uma única vez [46]. Para transportar dados para um grupo de usuários, é necessário utilizar um mecanismo com suporte a protocolos de rede.

3.1.4.3 Multi Protocol Encapsulation – MPE

O MPE oferece um mecanismo para transportar datagramas em um fluxo de transporte. Ele pode ser utilizado para transportar qualquer tipo de protocolo de rede, com o uso do encapsulamento LLC/SNAP[5] [47], cobrindo *unicast* (datagramas enviados a um único receptor), *multicast* (datagramas enviados a um grupo de receptores) e *broadcast* (datagramas enviados a todos os receptores). Um endereço *media access control* (MAC) de 48 bits é utilizado para endereçar receptores [48], e a informação é transportada em seções de datagramas, que seguem o formato de seções privadas. Essas seções oferecem um formato eficiente para mapear datagramas em pacotes de TS MPEG2 e também dão suporte à filtragem com base no endereço MAC. Se o comprimento do datagrama for menor ou igual a 4080 bytes (incluindo o possível cabeçalho LLC/SNAP), o datagrama será enviado em uma única seção. No caso do endereço de protocolo de internet (*internet protocol* – IP) e do cabeçalho serem omitidos, a unidade máxima de transporte (*maximum transport unit* – MTU) deve ser configurada para aproximadamente 4080 bytes, de modo que o datagrama nunca será fragmentado. Com endereço IP e cabeçalho presentes, o MTU deve ser configurado para 4074 bytes ou menos. As seções MPE são usadas para transmitir datagramas IP no contexto do DVB-H [49, 50].

Os sistemas de difusão apresentados até aqui satisfazem a maior parte das necessidades de transporte de dados para soluções proprietárias. Entretanto, a tecnologia de TVD faz uso de

---

[5] LLC/SNAP (*Logical Link Control / Subnetwork Access Protocol*)

mecanismos mais complexos, como os carrosséis, que transportam dados de forma cíclica e estão associados ao uso do *middlleware*.

3.1.4.4 Carrosel

Os Carrosséis são protocolos de difusão de dados definidos pelo padrão DSM-CC. O nome deriva, do fato deste protocolo permitir a repetição cíclica de um determinado conjunto de dados, em um fluxo de transporte. Essa característica auxilia o receptor no acesso aos dados, visto que o mesmo, em busca de determinada informação, necessita apenas aguardar sua próxima repetição. Também é possível, a partir do receptor, adotar mecanismos de armazenamento prévio de conteúdo de um carrossel, de forma que a disponibilidade dos dados a alguma aplicação seja agilizado. Maiores detalhes sobre este mecanismo podem ser vistos em Reimers [46].

O carrossel de dados é o mecanismo mais simples definido pelo DSM-CC e proporciona uma organização lógica baseada em módulos, que são então associados em grupos. Como os dados são encapsulados nessas estruturas, essa forma de *datacasting* é recomendada para a difusão de dados, divididos em unidades de tamanho determinado (Figura 9). Porém, não há nenhuma descrição sobre qual o tipo do dado transportado por um módulo, e o seu correto tratamento é de responsabilidade da aplicação. O carrossel de dados permite também que determinados módulos sejam difundidos em intervalos de tempo diferentes, o que proporciona um mecanismo de prioridade [46].

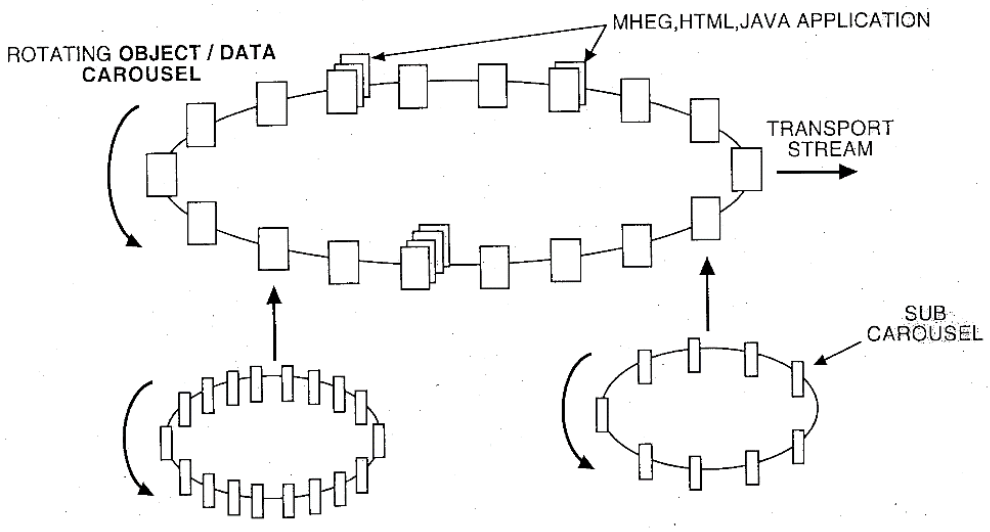

Figura 9 – Carrossel demonstrando a repetição cíclica de diferentes objetos (Fonte [48]).

Os carrosséis de objetos baseiam-se nas definições dos carrosséis de dados, porém, passam a tratar a informação na forma de objetos, que apresentam uma determinada estrutura. Para a

difusão de dados, dois objetos são importantes em um carrossel deste tipo: objetos do tipo arquivo e do tipo diretório. Com esses tipos de objetos, é possível criar um sistema de arquivos simples. Dessa forma, o terminal de acesso pode reter arquivos de um sistema que está sendo difundido em um carrossel de objetos, como se os mesmos estivessem disponíveis localmente. Devido a essa característica, o carrossel de objetos pode ser definido como um sistema de arquivos de difusão [48].

3.1.5 Comparação entre mecanismos de transmissão de dados

Para se escolher o método de transmissão mais adequado à transmissão do feixe de bits de reconfiguração do FPGA, o formato dos dados e os requisitos temporais (sincronização) associados foram levados em consideração. Com relação ao formato, os dados são classificados em: delimitados, não delimitados e *datagramas*. Os dados delimitados são aqueles que podem ser divididos em unidades de tamanho determinado, como em objetos, arquivos, etc. Por outro lado, os não delimitados são considerados como um fluxo contínuo de bits. Por fim, os datagramas correspondem à fragmentação dos dados em pacotes, seguindo algum protocolo de comunicação [51]. Quanto aos requisitos temporais, os dados são divididos em síncronos, sincronizados e assíncronos. Os dados síncronos possuem requisitos de sincronização com outros dados do mesmo fluxo, conhecido por sincronização *intra-mídia*.

Dados sincronizados são aqueles que devem ser apresentados em instantes predeterminados de tempo e em sincronismo com elementos de outras mídias como, por exemplo, fluxos de vídeo, o que é conhecido como sincronismo intermídia. Os dados assíncronos, por sua vez, não possuem requisitos temporais de sincronização. A Tabela 1 reúne as principais características dos métodos de transmissão de dados.

Os dados de reconfiguração de hardware são considerados delimitados por terem tamanho definido. Portanto, o feixe de bits (com o conteúdo de reconfiguração) poderá ser dividido em um número finito de fatias de tamanho determinado, antes de ser carregado em um mecanismo de transporte. Do ponto de vista dos requisitos temporais, os dados são assíncronos, pois o feixe de bits não precisa de nenhuma sincronização com outros tipos de dados ou elementos de outras mídias. A preocupação reside na recuperação dos dados, de forma segura, e, se possível, com a utilização de algum mecanismo de detecção de erros. Utilizando-se carrosséis, há facilidades na recuperação da informação transportada, tendo em vista que este mecanismo já possui suporte à repetição cíclica, facilitando a recuperação da informação. Porém, os carrosséis, por utilizarem

estruturas complexas, também necessitam de mecanismos de encapsulamento mais complexos, durante o processo de multiplexação do TS.

Tabela 1 - Relação entre os métodos de datacasting, tipos de dados e requisitos temporais.

REQUISITOS		DATA PIPING	DATA STREAMING		CAROUSEL	MPE
			Private Sections	**PES**		
Suporte a detecção de erros		-	OK	OK	-	OK
Suporte a protocolo de rede		-	-	-	-	OK
Suporte a unicast		-	-	-	-	OK
Suporte a multicast		-	-	-	-	OK
Suporte a broadcast		OK	OK	OK	OK	OK
Suporte a dados não delimitados	Assíncrono	OK	OK	-	-	-
	Sincronizado	-	-	OK	-	-
	Síncronos	-	-	OK	-	-
Suporte a dados delimitados	Assíncrono	-	OK	-	OK	-
	Sincronizado	-	-	-	OK	-
	Síncronos	-	-	-	-	-
Suporte a Datagramas	Assíncrono	-	OK	-	-	OK
	Sincronizado	-	-	OK	-	-
	Síncronos	-	-	OK	-	-

"OK", para o método mais apropriado e "-", para sem suporte ou menos apropriado.

Na recepção do conteúdo, as estruturas computacionais necessárias para a extração das informações das estruturas do carrossel são complexas, o que naturalmente leva a um maior consumo de recursos computacionais e espaço em memória. Os núcleos de reconfiguração de hardware são arquivos relativamente pequenos, o que sugere a utilização de mecanismos de transporte mais simples.

O mecanismo de *data streaming* via seções privadas é mais simples e tem suporte à detecção de erros, garantindo a segurança do sistema. Além disso, ele oferece suporte a dados assíncronos e delimitados e utiliza estruturas menos complexas que os carrosséis. Com o emprego do mecanismo de *data streaming* via seções privadas, o núcleo de hardware (dado delimitado) pode ser quebrado em seções de até 4096 bytes, sendo 4 bytes de código de redundância cíclica (*cyclic redundancy check* – CRC) e mais 12 bytes de cabeçalho. As seções são numeradas de acordo com a ordem de inserção das fatias (cada 4096 bytes de dados) do núcleo de hardware. Esse mecanismo também tem repetição cíclica e permite configurar a taxa

de repetição das seções (Figura 10). É possível observar que as seções podem ser repetidas em um determinado intervalo de tempo.

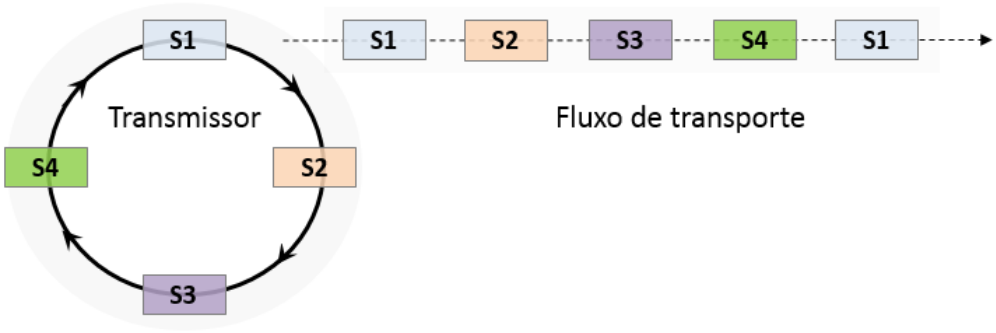

Figura 10 – Esquema de repetição cíclica de seções (S1, S2, S3 e S4).

As seções privadas podem usar a mesma taxa de repetição. Desta forma, o sistema permite que a aplicação residente do receptor consiga recuperar todas as seções que encapsulam o núcleo de hardware. Dentre os dois mecanismos mais adequados ao transporte dos dados, o método de *data streaming* através de seções privadas é o mais indicado. Entre os fatores levados em consideração, está a capacidade de envio de dados de CRC, associados aos conteúdos da carga de cada seção.

3.1.6 Código de Redundância Cíclica

Os algoritmos de detecção de erros são fundamentais para as telecomunicações. O uso destas técnicas permite que o receptor consiga determinar se um conjunto de dados, transmitido através de um determinado canal de transmissão, está corrompido [52]. Os canais de transmissão estão sujeitos a ruídos, que podem causar interferência no canal de transmissão e adicionar erros ao conteúdo transmitido. Para que o receptor consiga detectar se os dados chegaram corrompidos, é necessário que o transmissor anexe, a este conjunto de dados, um valor de redundância chamado CRC. O CRC é uma função do conjunto de dados, que posteriormente é usada pelo receptor, para comparação (*checksum*) com o CRC resultante dos dados recebidos [53, 54].

Os códigos de redundância cíclica também são conhecidos como códigos polinomiais, pois é possível escrever um padrão de bits como se fosse um polinômio. Neste padrão, os expoentes de cada polinômio representam os índices dos bits ‘1’ dentro do padrão de bits, como em

$$\mathbf{G\,(x) = x^5 + x^3 + x^2 + x^0 = (101101)_2}. \quad (1)$$

O cálculo é realizado utilizando-se aritmética módulo 2 (Figura 11). O valor resultante deste processo é um código CRC, que é anexado ao bloco de dados a ser transmitido. O receptor aplica, ao bloco de dados, o mesmo polinômio e compara o resultado do CRC obtido com o CRC enviado pelo transmissor. Se os dois valores são correspondentes, pode-se concluir que os dados foram recebidos corretamente; caso contrário, os dados estão corrompidos e alguma ação pode ser tomada.

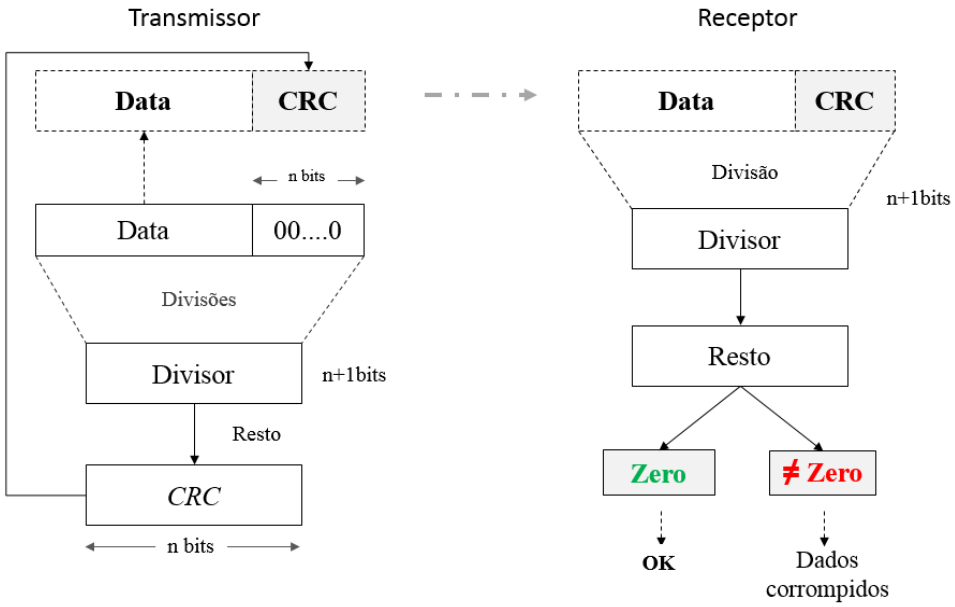

Figura 11 – Exemplificação do uso do método de CRC.

Neste trabalho, considera-se somente a "detecção de erros" para os dados de reconfiguração de hardware, ou seja, a correção de erros não é realizada. O CRC está sendo usado para garantir a integridade dos dados de reconfiguração de hardware, assim como é feito na recepção das tabelas. Como consequência, o procedimento de reconfiguração de hardware, usando o núcleo transmitido, só será realizada se o conteúdo não estiver corrompido.

Até aqui, os conceitos básicos que envolvem a sinalização e a caracterização do sinal de TV digital (fluxo de transporte), além do mecanismo de transporte mais adequado ao transporte do núcleo de hardware, foram apresentados. É possível também observar que o sinal é composto por diversos tipos de informação, incluindo áudio, vídeo e dados, e que os padrões de TV digital possuem similaridades, no que diz respeito ao uso das camadas MPEG para compactação e transporte de dados.

O fluxo de transporte, gerado no processo de multiplexação, passa então para uma nova etapa, que é a preparação do sinal para a transmissão. O padrão SBTVD utiliza técnicas de codificação de canal e modulação, que serão apresentadas na próxima Seção.

3.1.7 Codificação de canal e modulação no padrão SBTVD

A etapa de codificação de canal, no SBTVD, ocorre após a multiplexação do TS. Este processo auxilia o receptor na detecção e correção dos erros que podem ser inseridos pelo meio de transmissão. A codificação utiliza dois métodos de correção de erros concatenados (*reed-solomon* e convolucional) [34]. Além da correção de erros, técnicas de intercalação temporal são utilizadas para permitir uma redução da taxa total de erros. Esse processo tem o objetivo de aumentar a robustez do sinal de transmissão.

Para que haja transmissão hierárquica, o sistema ISDB-T define um quadro multiplex de TS, que é uma remultiplexação dos feixes que devem ser transmitidos em cada camada (Figura 12). Com essa remultiplexação predeterminada, o receptor regenera o mesmo TS. Assim, no receptor, apenas os pacotes do sinal one-seg podem ser recuperados [34]. Depois da remultiplexação, o fluxo de transporte (TS), é aplicado ao algoritmo *reed solomon* [55], que consiste em um corretor de blocos que gera os pacotes de dados do canal com base no sinal de entrada.

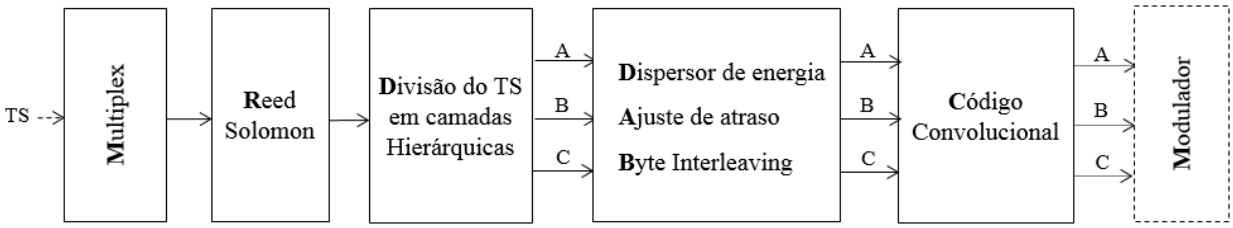

Figura 12 – Diagrama de blocos da codificação de canal no sistema SBTVD.

A cada símbolo de 188 bytes, mais 16 bytes de paridade são adicionados, os quais permitem corrigir até 8 bytes corrompidos. Para a transmissão hierárquica, o TS resultante é então dividido em um máximo de três fluxos paralelos de processamento (camadas – *layers*). Em seguida, cada TS é submetido a um módulo de dispersão de energia, que tem o objetivo de evitar a repetição de grandes sequencias de 1s ou 0s, através de um circuito conhecido por *pseudo random bit sequence* (PRBS). No mesmo módulo, também ocorre um ajuste de atraso que tem por objetivo corrigir o atraso associado ao *byte interleaving* [34]. Este ajuste visa equalizar o tempo de transmissão e recepção de todas as camadas, compensados sempre pelo lado do transmissor, o que ocorre porque as camadas possuem codificações diferentes.

Em seguida, o codificador convolucional pode efetuar o puncionamento para as seguintes taxas de compressão: 1/2, 2/3, 3/4, 5/6 ou 7/8, dado que, para uma taxa mãe de 1/2, cada bit de entrada dois bits de saída. A partir da especificação destes conjuntos de parâmetros de

transmissão, como número de segmentos, taxa de codificação interna e esquema de modulação para diferentes camadas hierárquicas, é possível obter a robustez desejada no sinal de transmissão [56].

Após a codificação de canal, os sinais de cada segmento seguem para a etapa de modulação. Os bits do sinal de entrada de cada segmento são então entrelaçados e mapeados pelo esquema definido para cada camada hierárquica (Figura 13). As camadas hierárquicas, que estão parametrizadas para diferentes configurações, precisam ser combinadas novamente para que, ao final da cadeia, sejam submetidas ao processo matemático de conversão, com base na transformada rápida inversa de Fourier (*inverse fast fourier transformer* – IFFT), gerando múltiplas portadoras.

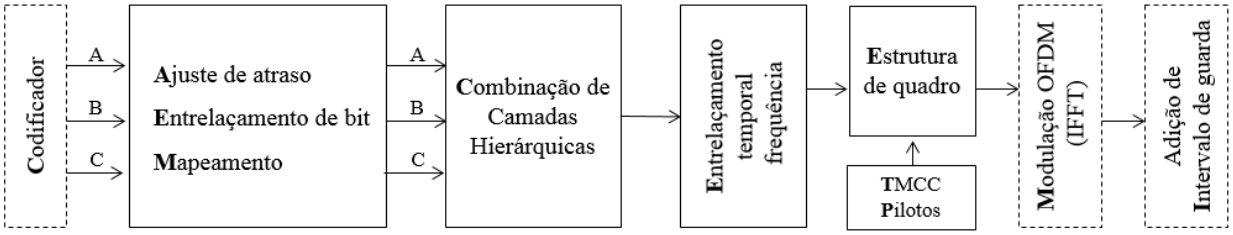

Figura 13 – Diagrama de blocos do processo de configuração para modulação das portadoras.

O sinal combinado é submetido ao entrelaçamento temporal, que ameniza o ruído impulsivo, e ao entrelaçamento em frequência, que previne a seletividade [34]. Em seguida, na estrutura de quadro são adicionados alguns sinais de frequência pilotos e sinais de controle, como o *transmission and multiplexing configuration control* (TMCC). As frequências pilotos são utilizadas durante os processos de estimação de canal e equalização, o que proporciona uma melhor qualidade do sinal em ambientes ruidosos. O TMCC transmite os parâmetros de configuração, permitindo ao receptor uma configuração automática. O sinal de saída do bloco de quadros segue para a modulação OFDM, gerando o sinal de frequência intermediária (*intermediated frequency* -- IF). A modulação OFDM proporciona multiplexação por divisão de frequências, onde múltiplos sinais são enviados em paralelo, utilizando-se múltiplas subportadoras. Em seguida, o intervalo de guarda é adicionado ao sinal, que é uma extensão cíclica do símbolo OFDM. O receptor utiliza este intervalo de guarda para eliminar as interferências existentes devido a multipercursos. Quatro tempos de intervalo de guarda foram padronizados: 1/4, 1/8, 1/16 e 1/32 da duração do símbolo. Os testes de transmissão realizados em diversas regiões do Brasil mostraram que os intervalos de guarda de 1/8 ou 1/16 se mostraram suficientes [56].

O SBTVD utiliza uma estratégia de modulação baseada em Multiplexação Ortogonal por Divisão de Frequência com segmentação de banda (*band segmented transmission-orthogonal frequency division multiplexing* – BST-OFDM). A modulação BST-OFDM consiste em dividir a banda útil de um canal dividido em 13 segmentos de 428,5 KHz, em canais de 6MHz, que podem ser agrupados em até três camadas (*layers*) distintas, seguindo o processo de transmissão hierárquica. A Figura 14 apresenta uma das organizações possíveis de divisão do canal em 13 segmentos. Considerando-se que cada camada pode ser modulada com diferentes serviços, parâmetros de transmissão diferentes podem ser configurados.

As características da modulação OFDM permitem uma maior robustez a distorções de multipercurso, que é comum em ambientes com muitos obstáculos, como os ambientes urbanos [56]. Outra característica importante da modulação OFDM é a capacidade de operar no esquema de rede de frequência única (*single frequency network* – SFN), que possibilita a transmissão do mesmo sinal por antenas diferentes utilizando a mesma frequência. A SFN permite uma maior robustez ao efeito Doppler[6] [56] associado à recepção em dispositivos móveis, o que é dado pela adequação da distância entre as estações da SFN [57, 58, 59]. Assim, três modos de transmissão foram padronizados, que indicam o número de portadoras utilizadas: modos 1, 2 e 3.

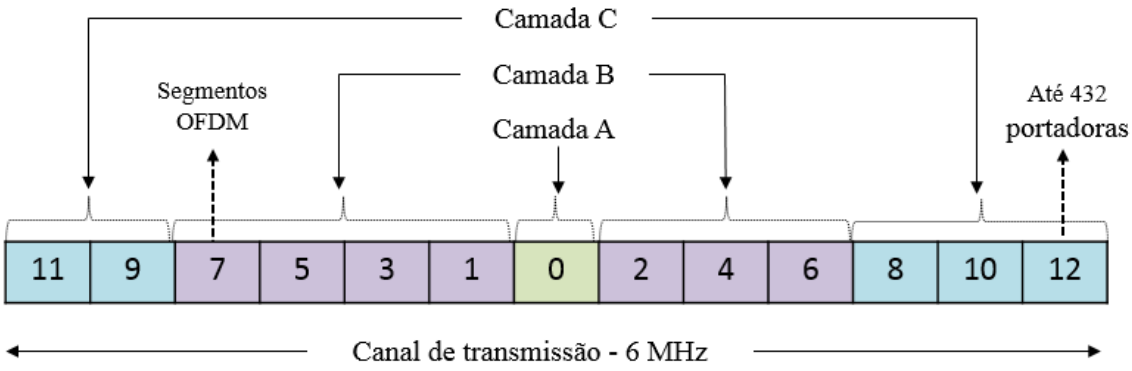

Figura 14 – Divisão do Canal de TV digital em 13 segmentos OFDM

Os conceitos relacionados com a codificação de canal e modulação, que foram apresentados neste capítulo, são necessários para a configuração dos parâmetros utilizados pelos equipamentos de modulação, durante a transmissão do sinal DTV. A escolha desses parâmetros interfere diretamente na qualidade do sinal transmitido e deve ser feita conforme o meio

[6] O Efeito Doppler é observado em objetos em movimento ao emitirem ou refletirem ondas em relação a um ponto de observação. Este nome foi atribuído em homenagem a Johann Christian Andreas Doppler, que desenvolveu esta teoria no ano de 1842.

utilizado. Durante os experimentos realizados neste trabalho, parâmetros como intervalo de guarda, número de segmentos por camada, taxa de bits e *time interleave* serão levados em consideração.

3.2 Receptores de TV digital (Set-top box)

O Set-top Box, também conhecido por *integrated receiver decoder* (IRD), é um dispositivo de recepção e decodificação de sinais de televisão digital conectado a um televisor, por meio de cabos ou qualquer outro tipo de conexão, e que, para tanto, disponibiliza interfaces de saída de áudio e vídeo, sejam estas analógicas ou digitais [12]. No mercado, é possível encontrar diversos modelos de receptores, geralmente comercializados no padrão de TV Digital usado em uma dada região. No caso do Brasil, o padrão brasileiro SBTVD é utilizado, que tem por base a mesma modulação utilizada no Japão (ISDB-T). Embora existam outros padrões de TV digital, os respectivos dispositivos utilizam arquiteturas de hardware semelhantes.

3.2.1 Características da Arquitetura de Hardware

A arquitetura de hardware de um receptor de televisão digital é similar à de um computador de uso pessoal, sendo composta por processador, memórias e demais periféricos. Alguns processadores possuem múltiplos núcleos de processamento. Normalmente, as tarefas que exigem um maior poder de processamento são executadas em núcleo dedicado (*e.g.*, decodificador MPEG), ao passo que as aplicações são processadas em outra unidade. Os principais módulos da arquitetura de um receptor estão ilustrados na Figura 15.

O *frontend*, também conhecido como "tuner", é uma peça fundamental na arquitetura de hardware de um receptor. O *frontend* é responsável pela sintonia do sinal de TV digital e normalmente é dedicado a um tipo de modulação, associada a um determinado padrão de TVD (*e.g.*, DVB-T ou ISDB-T) [12]. O *frontend* é composto por um sintonizador (*phase-locked loop* – PLL) e um decodificador de canal. O PLL traz um sinal de saída, que estava no espectro de frequências de TV, para uma frequência menor, conhecida como frequência intermediária (FI). Assim, é possível utilizar o PLL para sintonizar uma dada frequência do espectro de TV Digital.

O demodulador, por sua vez, recebe o sinal modulado da saída do sintonizador (PLL) e efetua a demodulação/decodificação de canal, gerando um TS como saída. O TS gerado no processo de demodulação é então direcionado ao demultiplexador, que realiza o processo inverso da multiplexação e separa os fluxos elementares. Dessa forma, o demultiplexador filtra os pacotes com base nos seus PIDs e opcionalmente, remove o cabeçalho do pacote PES.

Geralmente, o TS é composto por 3 fluxos distintos, sendo um de dados, outro contendo pacotes PES de áudio e o último pacotes PES de vídeo. Os pacoted PES de áudio e vídeo são então direcionados [12] aos seus respectivos decodificadores (Figura 15).

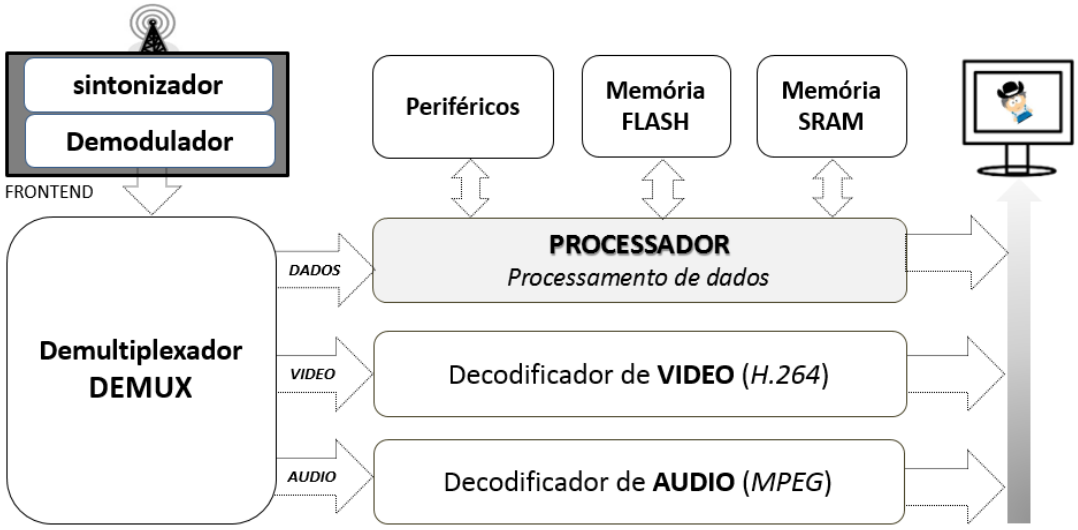

Figura 15 – Arquitetura básica de um receptor de TV digital.

O decodificador de vídeo H.264 é, normalmente, um módulo desenvolvido em hardware e composto por um processador de sinais digitais (*digital signal processor* - DSP). O DSP é responsável pela decodificação de vídeo H.264, cuja saída é então enviada ao codificador de saída de vídeo (*digital encoder* – DENC), que disponibiliza sinais em um determinado padrão. Na decodificação de áudio, o decodificador recebe um PES de áudio e o decodifica, de acordo com o formato de áudio transmitido. Então, o resultado é direcionado à interface de saída de áudio do receptor. Por fim, o fluxo de dados é direcionado ao processador, sendo tratado e manipulado pelo software residente.

Em alguns casos, o demultiplexador em hardware já disponibiliza funcionalidades que permitem fazer a filtragem de seções e pacotes PES, através de um hardware dedicado. Certamente, o demultiplexador que oferece esta funcionalidade precisa ser configurado através de uma API desenvolvida em software. Normalmente, a aplicação residente, após extrair os valores de *table_id* e PID das tabelas contendo as informações do sistema, utiliza esta informação para configurar os registradores do demultiplexador, de modo que a filtragem ocorra.

Nessa arquitetura, também é possível encontrar uma memória de armazenamento persistente (*e.g.*, memória *flash*) para o sistema de arquivos, configurações de sistema e programas residentes, e uma memória volátil (*e.g.*, *static random access memory* - SRAM), para carga e execução de software. Os demais periféricos, normalmente presentes na arquitetura de

hardware do receptor, são dedicados a tarefas específicas, principalmente aquelas que envolvem interfaces de comunicação, como controladores USB, ethernet, serial, painel frontal, controlador HDMI, receptor infravermelho, etc. O sistema embarcado é relativamente complexo e pode ser organizado seguindo uma ordem hierárquica, para uma melhor abstração do hardware do receptor.

Os fabricantes de processadores para dispositivos embarcados normalmente oferecem soluções de referência, também conhecidas por *reference designs* (plataformas de referência) ou *turnkey* (solução completa para decodificação de sinais de TV Digital). Tais soluções, mais próximas de um produto final, permitem, através de customizações e algumas adaptações, gerar o produto desejado. Estas soluções podem ser convertidas em diversos produtos que utilizem decodificação de áudio e vídeo em alta definição. A plataforma de referência, apresentada na Figura 16, é um *reference design* (placa) fornecido pelo fabricante de semicondutores NXP, que utiliza o processador de 300 MHz PNX8735 [60], juntamente com outros componentes que possibilitam a criação de uma solução completa para um receptor de TV digital.

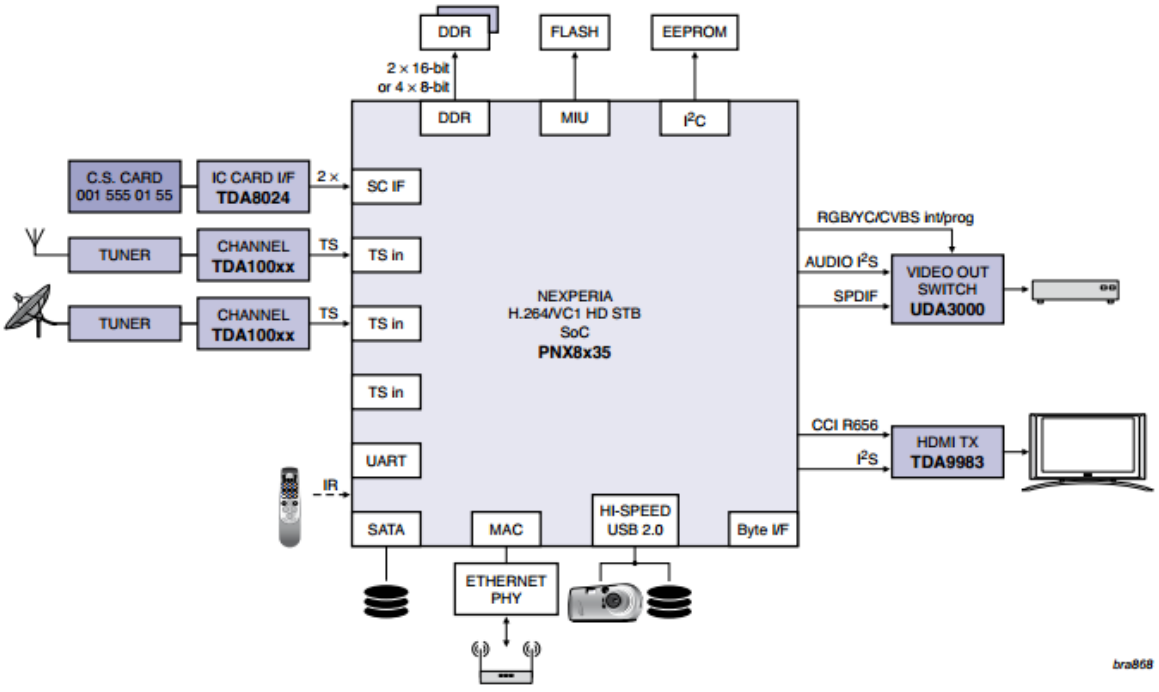

Figura 16 – Plataforma de referencia NXP STB225 (Fonte [60])

Entre os diversos componentes desta plataforma, estão os relacionados a interfaces multimídia, como o controlador de saída *high-definition multimedia interface* (HDMI) TDA9983 [61]. A interface HDMI pode transportar sinais de áudio e vídeo, do receptor ao televisor. Estes receptores geralmente disponibilizam outros tipos de saídas de vídeo como *red, green and blue*

(RGB) e *composite video broadcast signal* (CVBS), também conhecida como saída de vídeo composto, assim como saídas de áudio, *inter-IC sound, integrated interchip sound, ou IIS* (I^2S) ou saída de áudio I^2S, *sony/philips digital interface format* (SPDIF), etc. Além desses componentes, também são fornecidas interfaces para memória SRAM, *electrically-erasable programmable read-only memory* (EEPROM) e memória *flash*, assim como a interface de transferência de dados *serial advanced technology attachment* (SATA), para armazenamento externo. Entre essas interfaces, a plataforma disponibiliza entradas para TS, permitindo o uso do *tuner* (*frontend*) de diversos fabricantes. Geralmente, as plataformas vêm equipadas com um *tuner* de um fabricante específico [60] e essas interfaces, nesse caso, podem ser utilizadas para customizações.

Além da arquitetura de hardware e periféricos, apresentados nesta seção, os fabricantes também disponibilizam exemplos de aplicações embarcadas, para serem utilizadas como base para o desenvolvimento do produto desejado. Na próxima Seção, as características da arquitetura de software, utilizada em receptores de TV digital, são apresentadas com maiores detalhes.

3.2.2 Características da Arquitetura de Software

O software do receptor é organizado em camadas com diferentes níveis de abstração, que permitem o gerenciamento e o acesso a dispositivos de hardware do receptor. Durante o funcionamento normal do receptor, diversos tipos de acesso a camadas de baixo nível e hardware são realizados. Os acessos podem representar desde uma simples consulta, com o objetivo de obter o estado de um periférico, até a configuração de registradores de um determinado dispositivo. Um exemplo deste último é a configuração dos registradores do *frontend* para se realizar uma sintonia de um canal, em uma determinada frequência. Durante a instalação dos canais, a aplicação residente efetua a varredura dos sinais de TVD, configurando o *frontend* em cada uma das frequências do espectro de TV digital. Durante esse processo, caso algum sinal seja encontrado, algumas informações são extraídas (*e.g.*, serviços, PIDs de áudio e vídeo, etc.).

A organização das camadas do software de um receptor é similar à utilizada em um computador de uso pessoal. Normalmente, os sistemas embarcados desta natureza são compostos por uma camada de sistema operacional (SO), baseada em versões customizadas de um determinado sistema operacional. A Figura 17 ilustra a organização de software de um receptor de televisão digital, com base no sistema Linux. O sistema operacional (2) e o conjunto de *device drivers* (3) abstraem o hardware das demais camadas de software do dispositivo [26]. O sistema operacional do receptor administra o uso dos recursos de hardware e gerencia a execução das aplicações, no contexto do receptor. Já a camada de *device drivers* serve como mediador entre o

software e o hardware. O DirectFB (4) [62] é uma camada de abstração que disponibiliza um conjunto de bibliotecas de software, que permitem aceleração gráfica, manipulação de dispositivos de entrada (*key handler*) e oferece suporte à manipulação de diversas janelas e propriedades de controle do plano gráfico. Uma camada de controle das propriedades do som (5) para o sistema operacional Linux, chamada *advanced linux sound architecture* (ALSA), oferece suporte a interfaces de áudio.

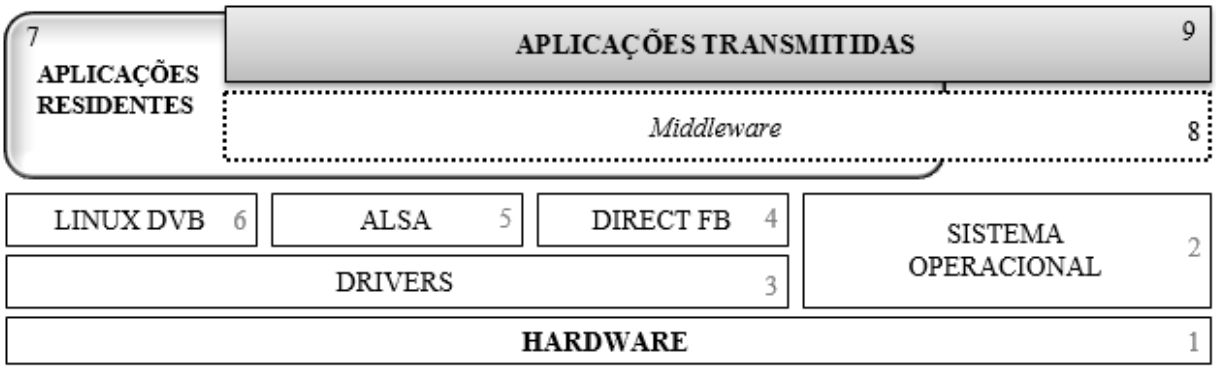

Figura 17 – Arquitetura de software baseada no Sistema Operacional Linux.

A camada de abstração Linux DVB (6) é composta por um conjunto de bibliotecas com diversos métodos, que facilitam o acesso ao demultiplexador e permitem filtrar dados que compõe o fluxo de transporte MPEG. As aplicações residentes (7), também conhecidas como aplicações nativas, são geradas para serem executadas no contexto do receptor. Estas aplicações são utilizadas para diversos fins, como gerenciar configurações do receptor, tais como resolução máxima permitida, propriedades de imagem, som, acesso a periféricos, etc. Além das aplicações de configuração, também existem aplicações para instalação de canais, que são utilizadas quando é necessário atualizar a lista de canais existentes, e aplicações de navegação entre os canais, conhecidas por *zappers*.

O *middleware* (8) é uma camada de interoperabilidade que adiciona a capacidade de interatividade ao receptor. Com uso do *middleware*, o receptor pode efetuar o download de sistemas de arquivos completos e executar os aplicativos, que são transmitidos no padrão do *middleware* utilizado. Para isso, normalmente utiliza-se uma máquina virtual, que possibilita que uma mesma aplicação transmitida seja executada em diferentes tipos de plataformas de recepção. O SBTVD utiliza o *middleware* ginga [63] no processo de interpretação e execução das aplicações interativas, as quais são transmitidas juntamente com os demais conteúdos de TV digital. Normalmente, essa camada de software não é disponibilizada pelos fornecedores de plataformas de referência. Para que as funcionalidades disponíveis no *middleware* sejam

utilizadas, é necessário primeiramente portar o *middleware* para a unidade de recepção [64, 65]. Na camada de aplicação (9), existem aplicações interativas que são transmitidas através do sinal de TV digital. Os tipos de aplicações que podem ser utilizadas nesta camada estão diretamente associados ao tipo de *middleware* utilizado.

3.2.3 Ambiente de Desenvolvimento

Os fornecedores de *reference designs* também disponibilizam o *toolchain* (*e.g.*, compilador e bibliotecas customizadas do sistema), que é necessário para a customização do sistema da plataforma [60]. Essas ferramentas auxiliam no desenvolvimento de sistemas para o contexto do *reference design* [60]. A plataforma utilizada neste projeto, por exemplo, é baseada no sistema operacional Linux, e o processo de desenvolvimento é realizado em um computador (*host*), conectado através de uma interface de rede. Dessa forma, o sistema de arquivos da plataforma fica hospedado no *host* e é carregado via *network file system* (NFS), toda vez que ela é inicializada. Esse processo é realizado através do *bootloader* (gerenciador de *boot*) da plataforma, que "busca" o sistema de arquivos de um ponto específico (*root-filesystem path*) do *host*. O sistema de arquivos é similar ao utilizado em uma distribuição Linux comercial e é composto por um conjunto de aplicações residentes, dispositivos de *drivers*, sistema operacional, arquivos de configuração, etc. As informações de Log (*debug*) são obtidas através de um sistema de comunicação serial como *kermit* ou *minicom*.

Neste capítulo, os conceitos básicos da arquitetura de hardware e software, de um receptor de TV Digital, e o respectivo ambiente de desenvolvimento foram apresentados. O receptor utilizado neste projeto disponibiliza interfaces de comunicação como porta USB, Serial, GPIO e ethernet [60], que podem ser utilizadas para o desenvolvimento do esquema de reconfiguração do FPGA. Nesse esquema, o receptor será utilizado para gerenciar a reconfiguração do FPGA. Na próxima seção, os principais conceitos relacionados aos dispositivos programáveis serão discutidos com maiores detalhes.

3.3 Dispositivos Programáveis

Os circuitos integrados não programáveis são aqueles nos quais sua lógica é inalterada, ou seja, após o processo de fabricação, não há possibilidade de se alterar o seu funcionamento interno, a não ser pela troca imediata do dispositivo ou em um novo processo de fabricação. Tais dispositivos são classificados como ASICs [66] e são largamente utilizados na indústria de sistemas eletrônicos. Os ASICs são desenvolvidos em uma altíssima escala de integração (*very large scale integration* – VLSI), que está diretamente ligada ao número de componentes físicos

(transistores) que podem ser integrados, em um determinado espaço físico [67]. Por outro lado, alguns dispositivos que utilizam lógica programável podem ser reprogramados diversas vezes [66]. Atualmente, é possível dividir esses dispositivos em duas categorias: os dispositivos programados na fábrica e os que podem ser programados em campo (Figura 18). Entre os dispositivos que são programados em fábrica, estão as memórias *read only memory* (ROM) e os módulos de *mask programmable gate array* (MPGA). Esses dispositivos são programados uma única vez, durante o processo de fabricação.

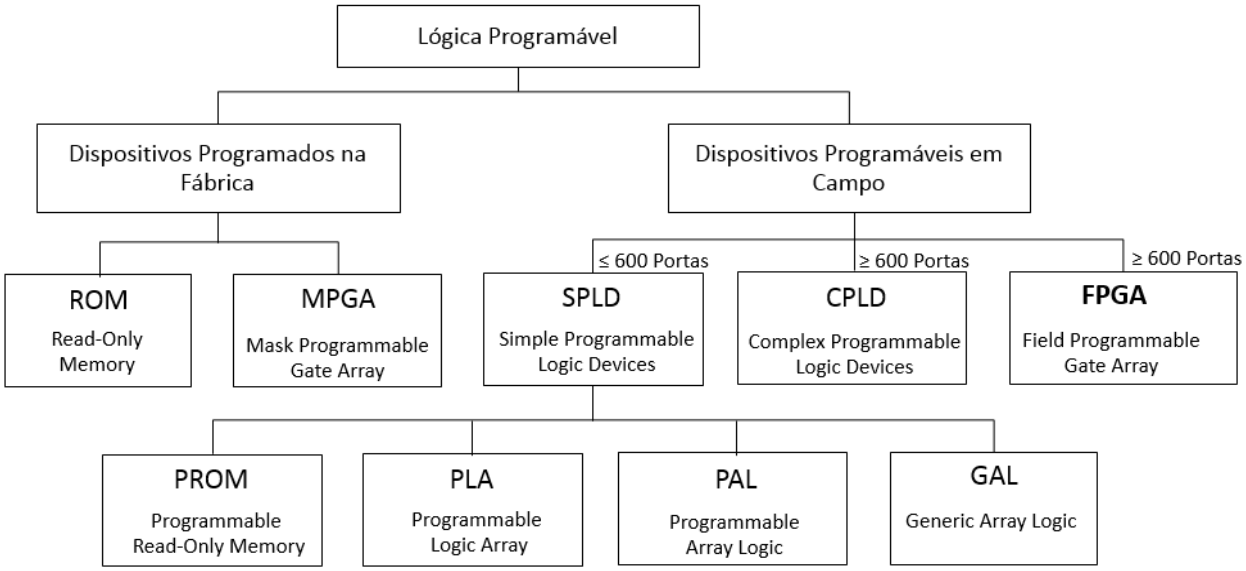

Figura 18 – Categoria dos dispositivos com lógica programável.

A ROM é utilizada como memória e o MPGA é um dispositivo que utiliza um aglomerado de transistores pré-fabricados, que podem ser customizados durante o seu processo de fabricação, com a especificação da camada de metal de interconexão. Por outro lado, há os dispositivos programáveis, que em alguns casos podem ter sua lógica alterada diversas vezes. Eles vão desde simples dispositivos lógicos, que podem ser programados em campo uma única vez, até complexos aglomerados de dispositivos lógicos, que podem ser reconfigurados quantas vezes forem necessários. Primeiramente, os com até 600 portas lógicas podem ser classificados como dispositivos programáveis de pequena capacidade e lógica simples (*simple programmable logic device* – SPLD) [69]. Os SPLDs são circuitos que possuem uma lógica interna baseada em um conjunto de portas AND e OR, denominadas arranjos lógicos, com a possibilidade do uso de

flip-flops[7] e macro células[8], nas saídas. Entre as características destes dispositivos, estão o baixo custo e um alto desempenho. As memórias ROM Programáveis (*programmable read-only memory* – PROM) podem ter sua lógica programada uma única vez. As memórias PROM utilizam duas matrizes de interconexão, sendo uma matriz fixa, composta de portas AND (*gates* AND), e uma matriz programável, composta de portas OR (Figura 19) [69]. A programação da matriz OR é dada através de elos fusíveis, que funcionam como interconexões físicas. Outro dispositivo da mesma classe dos SPLDs é a matriz programável logicamente (*programmable logic array* – PLA). Ao contrário das memórias PROMs, os PLAs são compostos de arranjos de portas AND de entrada (decodificador) e um arranjo de portas OR de saída (codificador), ambos programáveis. Os arranjos de portas AND de entrada são conectados às variáveis de entrada, através dos elos fusíveis, que estabelecem a conexão física. As funções de saída, estabelecidas pela matriz de portas OR programáveis, são obtidas através da conexão dos elos fusíveis com o produto de saída da matriz AND [69]. A matriz lógica programável (*programmable array logic* – PAL) é uma simplificação do PLA. A PAL permite a programação somente da matriz AND, diferentemente da PLA, que permite a programação das duas matrizes. Esta simplificação foi realizada para as aplicações que não necessitam que todas as combinações sejam programáveis.

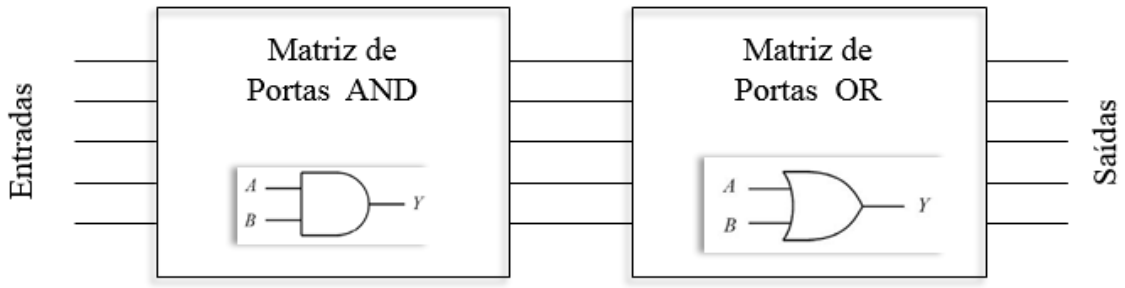

Figura 19 – Esquemas utilizados pelos SPLDs.

O último dispositivo da classe SPLD é o arranjo lógico genérico (*generic array logic* – GAL). Os dispositivos GAL possuem uma estrutura similar à usada na PAL, com a diferença de que são utilizadas macro células de saída, que podem ser configuradas como lógica combinacional ou sequencial (registradores). Os próximos dispositivos programáveis fazem

[7] O Flip-Flop, também conhecido como multivibrador biestável, é um circuito digital pulsado capaz de servir como uma memória de 1 bit.

[8] Uma macrocélula pode ser utilizada para configurar uma lógica combinacional, lógica sequencial ou até uma combinação destas duas [69].

parte dos dispositivos de lógica programável de alta capacidade (*high capacity programmable logic device* – HCPLD), com capacidade superior a 600 portas [69]. Estes dispositivos são conhecidos por *complex programmable logic device* (CPLD) e FPGA. Os CPLDs são fabricados com tecnologia *complementary metal-oxide semiconductor* (CMOS[9]) e são constituídos de várias SPLDs (blocos lógicos), conectadas através de interconexões programáveis chamadas PIA (*Programmable Interconnect Array*) ou *advanced interconnect matrix* (AIM). A diferença básica entre CPLD e FPGA reside no método de interligação dos blocos lógicos [70]. A estrutura do CPLD permite que sinais sejam interligados somente entre os blocos vizinhos lógicos, ao passo que o FPGA que permite interligações entre qualquer bloco lógico.

3.3.1 FPGA

Desde que foi concebida, a tecnologia de FPGA [71, 72, 73, 74] passou por avanços tecnológicos e atualmente seu uso já é uma realidade em sistemas embarcados e equipamentos de eletrônica de consumo. O FPGA foi criado pela Xilinx Corp., no ano de 1985, quando a empresa havia desenvolvido a lógica programável (memória), dando um próximo passo para criar uma área de circuitos com lógica programável [71]. Com o objetivo de entender a implementação do FPGA, os dispositivos programáveis que o antecederam precisaram primeiramente ser estudados.

Normalmente, as estruturas internas dos FPGAs podem variar de um fabricante para outro, ou até de uma família de mesmo fabricante para outra. No entanto, alguns elementos fundamentais do FPGA são comuns entre os vários modelos. É possível destacar três elementos fundamentais, que existem na totalidade das estruturas dos FPGAs: bloco lógico programável (*configurable logic block* – CLB), blocos de entrada e saída (*in/out block* – IOB) e matriz de comutação (*switch box* – SB) (Figura 20).

> **Bloco Lógico Programável** - Representa a unidade lógica programável de um FPGA. No interior de cada bloco lógico de FPGA, existem vários modos possíveis para a implementação de funções lógicas. O mais utilizado pelos fabricantes de FPGA (*e.g.*, altera corp.) é o bloco de memória *look up table* (LUT).

[9] CMOS é a tecnologia utilizada para a construção de circuitos integrados. Ela utiliza uma lógica de combinação de transistores de efeito de campo *metal-oxide-semiconductor field effect transistors* (MOSFET) do tipo-P e tipo-N para implementar circuitos digitais encontrados em microprocessadores, microcontroladores, dispositivos de processamento de sinais, etc.

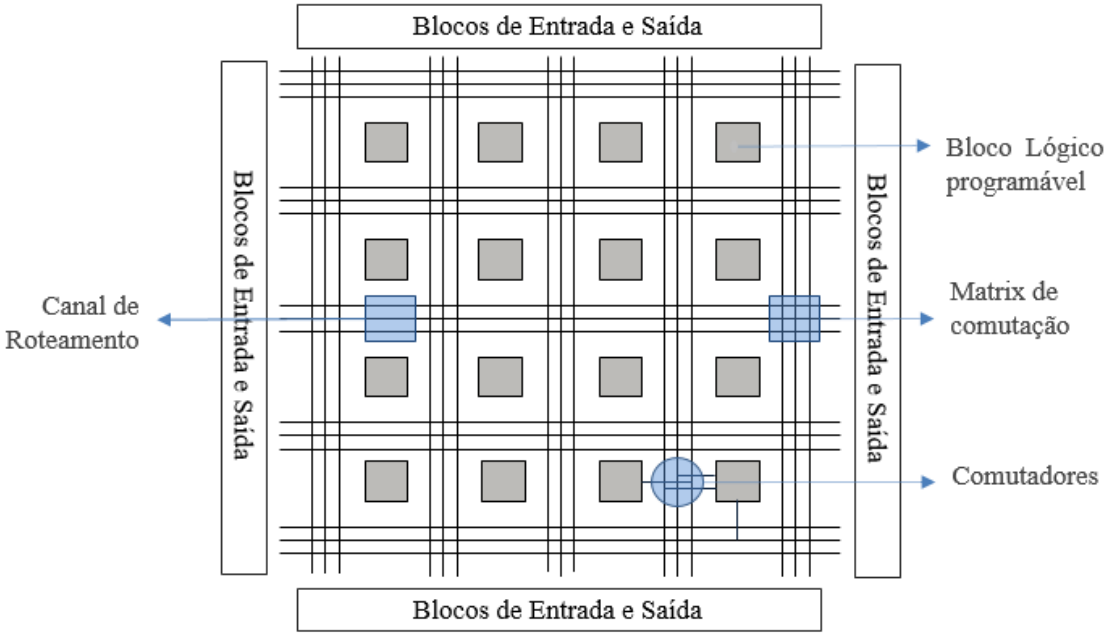

Figura 20 – Estrutura Simplificada de um FPGA.

Blocos de Entrada e Saída - Blocos de entrada e saída normalmente localizados na periferia dos FPGAs. Os blocos de entrada e saída (Figura 20) servem como interface com o ambiente externo.

Matriz de Comutação – É também conhecida por comutador (Figura 20) e representa pontos de conexão responsáveis pela interconexão entre os CLBs, através dos canais de roteamento. A arquitetura de roteamento de um FPGA é a forma pela qual os seus barramentos e as suas chaves de comutação são posicionados, com o intuito de permitir a interconexão entre os blocos lógicos. Essa arquitetura deve permitir que se obtenha um roteamento completo e, ao mesmo tempo, uma alta densidade de portas lógicas. As chaves programáveis de roteamento apresentam algumas propriedades, tais como tamanho, resistência, capacitância e tecnologia de fabricação, que afetam principalmente a velocidade e o tempo de propagação dos sinais e definem características operacionais, como volatilidade e capacidade de reprogramação [72].

As conexões físicas permitem modificar o comportamento do dispositivo reprogramável. Elas são descritas a partir do arquivo binário (feixe de bits), que define o conjunto de interconexões necessárias à configuração de um determinado circuito. Para se gerar o arquivo

binário de configuração, é necessário desenvolver e sintetizar o circuito, a partir de uma descrição de hardware (*e.g.,* HDL, *register transfer level* RTL, etc.).

3.3.2 Linguagens de Descrição de *Hardware*

As HDLs permitem descrever o comportamento e a funcionalidade de um circuito, através de uma descrição textual contendo propriedades e a sintaxe da linguagem HDL utilizada [75]. Essa descrição contém um conjunto de instruções e operadores lógicos utilizados para descrever circuitos digitais, em um alto nível de abstração. A complexidade de um circuito digital pode ser vista em diferentes níveis de abstração: quanto maior é o nível da representação de um circuito, mais legível e mais próximo da compressão humana ela se encontra (Figura 21).

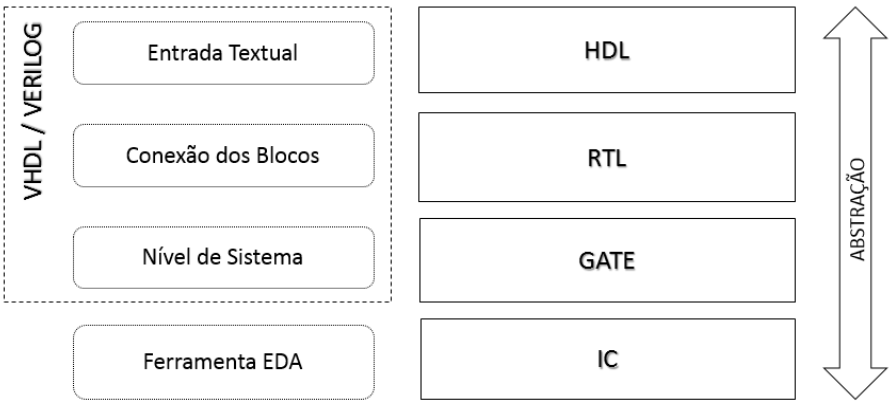

Figura 21 – Níveis de abstração dos circuitos digitais

As linguagens HDL ocupam o topo dessa hierarquia de representação. Nesse ponto, as especificidades do baixo nível ficam transparentes, dando ao projetista uma visão mais ampla durante o desenvolvimento do projeto. As linguagens VHDL e Verilog são as representações mais utilizadas por projetistas de circuitos integrados. A descrição em nível RTL consiste em uma descrição funcional formada por blocos que descrevem as operações lógicas e as transferências de dados entre os registradores, ou seja, este é um nível mais baixo de representação. As ferramentas EDA costumam converter o formato HDL para o RTL. Logo abaixo, está o nível estrutural representado por portas lógicas. Nesse formato, o circuito é descrito através de combinações entre portas lógicas, o que representa um maior nível de detalhe do circuito e, desta forma, torna-se pouco legível para projetistas de circuitos integrados. As ferramentas EDA podem gerar esta representação através da conversão dos níveis de abstração superiores. Na base dessa hierarquia, está o circuito propriamente dito, ou seja, o próprio IC que é constituído por milhares de transistores e conexões físicas.

3.3.3 Síntese de hardware

O procedimento de síntese de hardware inicia a partir de um projeto de circuito, no qual alguns fatores importantes, como a análise dos requisitos de especificação e a família e o modelo de FPGA mais adequado para o projeto, devem ser levados em consideração, com o objetivo de uma melhor otimização do processo de síntese e evitar retrabalhos futuros [75, 76, 77].

O desenvolvimento de um projeto para FPGA segue um fluxo de desenvolvimento de projeto distribuído em etapas distintas. Normalmente, essas etapas de projeto são automatizadas por ferramentas de projeto assistido por computador (*computer aided design* – CAD) conhecidas por ferramentas de automação eletrônica de projeto (*electronic design automation* – EDA). Um sistema típico de desenvolvimento de projetos, com ferramentas desta natureza, é dado em um fluxo de projeto sequencial, através da interconexão de várias ferramentas de software [75]. A Figura 22 ilustra um fluxo de um projeto de síntese de hardware, composto por 7 etapas distintas. A entrada do projeto é uma descrição textual do circuito em desenvolvimento, no formato de uma linguagem HDL (1). No processo de síntese, as ferramentas EDA utilizadas compilam e analisam o código HDL (*design*), convertendo esta descrição em descrições com baixo nível de abstração. Nesse processo, se a descrição HDL não tiver nenhum erro de sintaxe, esta é convertida para uma representação RTL [76, 77].

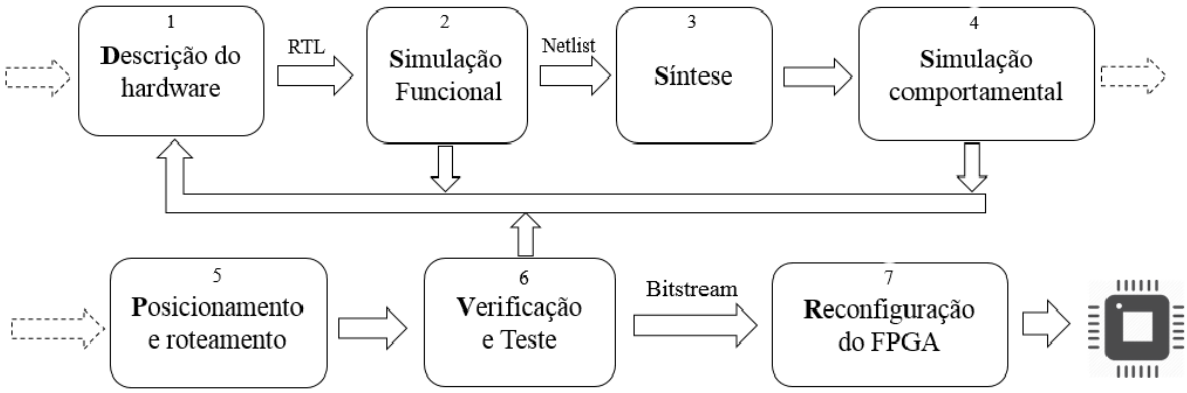

Figura 22 – Fluxo de projeto de um design de FPGA

Uma simulação RTL é então realizada, com o objetivo de verificar se as operações de transferência de dados estão de acordo com o sistema. Se alguma incoerência for detectada nesta simulação, o ciclo de projeto retorna para a fase (1), onde a descrição HDL será otimizada, de modo a resolver as incoerências detectadas na simulação funcional. A simulação no nível RTL costuma ser mais fácil de compreender, quando comparada a descrições com um nível menor de

abstração. Se essa simulação apresentar um comportamento satisfatório, a descrição RTL é transformada em uma *Netlist*[10] e o fluxo do projeto segue para a síntese lógica (3). A síntese lógica é composta por duas fases distintas: otimização lógica, que consiste em minimizar as equações booleanas, e mapeamento de tecnologia, que realiza a conversão das equações em módulos da tecnologia-alvo.

Com a descrição em portas lógicas, já é possível efetuar o posicionamento e o roteamento do circuito. No entanto, antes de partir para essa etapa, é desejável realizar a simulação comportamental do circuito (4). Para as etapas de simulação, que compõem esse fluxo de projeto, são utilizadas ferramentas EDA, que simulam o comportamento dos dispositivos físicos. Na simulação comportamental, as instruções presentes da descrição (declarações) são executadas, de forma a simular o funcionamento dos *testbenches* (bancos de ensaio). Estes são executados através sinais nas entradas primárias do circuito, o que é seguido da verificação das suas respostas, através das saídas primárias, visando a detecção de possíveis erros comportamentais [74]. Caso erros ou problemas no comportamento sejam detectados, é necessário voltar para a etapa 1 do projeto e modificar o código HDL, de modo a se corrigir o problema.

Após a simulação comportamental, o próximo passo é o posicionamento/roteamento (5). No primeiro, o posicionamento de todos os componentes eletrônicos, circuitos e elementos lógicos será decidido, em uma quantidade de espaço geralmente limitada. O roteamento consiste na ligação de todas as conexões físicas necessárias entre os componentes previamente posicionados, respeitando-se as regras e limitações do processo de fabricação do dispositivo alvo (FPGA). Após o posicionamento e roteamento, uma análise temporal (6) pode ser realizada. Caso sejam constatados problemas nessa análise, o fluxo retorna ao passo 1; caso contrário, a implementação do projeto está completa e o resultado deste processo é um arquivo de reconfiguração (núcleo de hardware) do dispositivo alvo. O próximo e último passo é a reconfiguração (7) do FPGA, que consiste em se utilizar uma ferramenta de programação de FPGA, para o envio do núcleo de hardware.

Em um ambiente de desenvolvimento, a reconfiguração do FPGA é realizada através de uma conexão física (cabo de gravação) entre a unidade de desenvolvimento (*host*) e a placa de desenvolvimento (dispositivo alvo – *target*), com o auxílio de uma ferramenta EDA específica para reconfiguração. Os modos de configuração são baseados em diferentes formatos de

[10] Netlist descreve a lista de componentes (portas lógicas, *flip-flops*, etc) e interconexões entre os elementos que fazem parte da descrição de um circuito digital.

arquivos. Os arquivos proprietários definem seus próprios formatos, no entanto, existem alguns formatos que são suportados por grande parte dos dispositivos de FPGA.

3.3.4 Arquivos de Reconfiguração de Hardware

Os arquivos de configuração do FPGA possuem formatos proprietários, que variam de um fabricante para outro. As ferramentas EDA utilizadas durante o projeto para FPGA podem gerar arquivos em diferentes formatos. Normalmente, o arquivo de reconfiguração gerado com o *toolchain* fornecido pelo fabricante de FPGA é compatível somente com os dispositivos do próprio fabricante. No entanto, existem formatos padrões, que podem ser utilizados para a reconfiguração de diversos FPGAs e diferentes kits de desenvolvimento. Normalmente, as ferramentas EDA convertem seus formatos proprietários em formatos universais, como é o caso dos formatos *serial vector file* (SVF) [78] e *standard test and programming language* (STAPL), sendo este último também conhecido por linguagem JAM (criada pelos engenheiros da fabricante altera).

O SVF é compatível com a maioria dos fabricantes de FPGA do mercado. Na Tabela 2, é possível observar a relação entre os fabricantes de FPGA disponíveis no mercado e os tipos de arquivos de configuração suportados por suas respectivas ferramentas EDA.

Tabela 2 - Fabricantes e formatos de arquivos suportados.

Fabricante	Ferramenta de Design	Formato dos arquivos de reconfiguração	Arquivos de reconfiguração padrão
XILINX	ISE	.jed, .bit, .rbt, .isc, .nky, .mcs, .exo, .mpm, .svf, .xsvf	.stapl, .svf
ALTERA	QUARTUS II	.pof, .sof, .jam, .jed	.jbc, .jam, .svf
LATTICE	ISP VM	.jed	.svf, .stapl
ACTEL	LIBERO	.pdb, .stp, .isc	.stp, .svf
ATMEL	AVRSVF	.hex	.svf
SILICON LAB.	HEX2SVF	.hex	.svf
CYPRESS	ISR	.jed	.stapl, .svf

A maioria dos dispositivos comercializados pelos fabricantes de FPGA mencionados oferece suporte aos formatos padrão SVF e STAPL [79]. Esses formatos de arquivo são compatíveis com a comunicação JTAG, que permite realizar diversas operações com FPGAs ou circuitos que utilizem a tecnologia *boundary scan* [79, 80]. Essa tecnologia, que é empregada na grande maioria dos circuitos integrados microeletrônicos, é utilizada para testar o circuito integrado e também para intermediar o processo de reconfiguração do FPGA. Assim como o SVF, o formato STAPL é uma boa opção para ser utilizado como formato padrão de saída para o

núcleo de hardware. O formato STAPL é mais avançado que o SVF, pois permite o uso de expressões condicionais, porém, o SVF é suportado por todas as ferramentas disponíveis no mercado, o que permite desenvolver uma solução para o uso em dispositivos fornecidos pela maioria dos fabricantes de FPGA.

Os diferentes formatos de arquivos apresentados permitem realizar a reconfiguração do FPGA, a partir de um determinado modo de reconfiguração. Os FPGAs suportam vários modos de reconfiguração e cada modelo tem suas próprias características físicas de programação. Os modos de reconfiguração de FPGA existentes serão discutidos com maiores detalhes na próxima seção.

3.4 Modos de Reconfiguração do FPGA

As ferramentas de projeto utilizadas para o desenvolvimento e síntese do núcleo utilizam esquemas proprietários para a reconfiguração do FPGA. A ferramenta utilizada nesse processo de reconfiguração está anexada ao kit de desenvolvimento fornecido pelo fabricante, fazendo parte das ferramentas utilizadas durante o fluxo de projeto (normalmente, não é disponibilizada individualmente). Dessa forma, para se gerenciar o processo de reconfiguração, é necessário utilizar um esquema de reconfiguração *standalone*.

Os FPGAs suportam alguns modos de reconfiguração, divididos em modos ativos, passivos e *joint test action group* (JTAG). Para a melhor compreensão de cada um destes modos, é necessário entender os conceitos de comunicação utilizados por cada um deles.

3.4.1 Configuração em Modo Ativo

A configuração em modo ativo considera o uso de um dispositivo FPGA interconectado a um dispositivo de configuração serial (EPCS). O dispositivo EPCS é, na verdade, uma memória *flash* que pode armazenar permanentemente os dados utilizados para realizar a reconfiguração do FPGA [81]. O FPGA é um dispositivo baseado em uma memória SRAM e, dessa forma, os dados de reconfiguração são voláteis e necessitam ser carregados cada vez que sistema é iniciado. O modo de reconfiguração ativo funciona na forma ativo serial (*active serial* – AS), onde os dados são transmitidos de forma serial, e também o modo ativo paralelo (*active paralell* – AP), onde os dados são enviados de forma paralela. No modo AS (Figura 23), o FPGA produz o sinal de *clock* e coordena a reconfiguração, através da leitura do núcleo de hardware, diretamente da memória EPCS e no momento da inicialização do sistema [81]. Uma característica deste modo é que o FPGA não fornece os endereços de leitura para o dispositivo de memória externa.

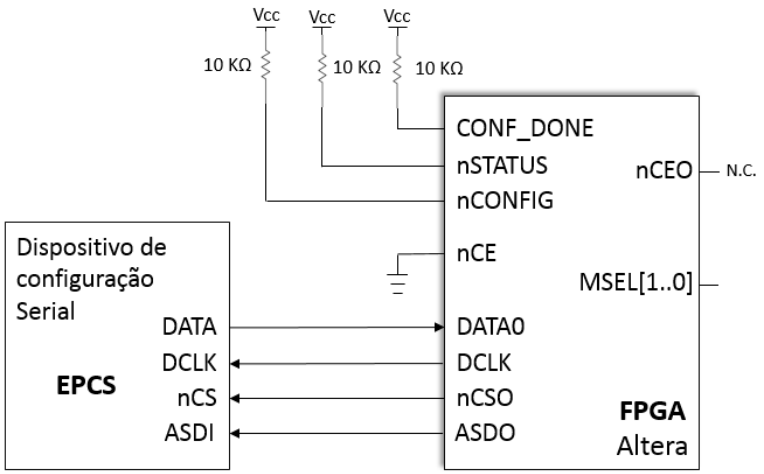

Figura 23 – Modo de configuração AS (Fonte [81]).

Neste modo, o FPGA atua como mestre e habilita o dispositivo EPCS, inserindo um sinal de nível baixo no pino nCS. Assim, o FPGA envia as instruções e endereços para a memória EPCS, através do sinal ASDI. O sinal de clock é gerado pelo próprio FPGA, que será reconfigurado. A memória EPCS responde às instruções, enviando os dados de reconfiguração através do pino de dados DATA0, na borda de descida (*falling edge*) do sinal de clock DCLK. Os dados ficarão armazenados no FPGA, na próxima borda de descida do ciclo de clock do sinal DCLK. O modo AP é similar ao modo de reconfiguração AS, com a principal diferença na forma de envio dos dados, que neste caso utiliza uma comunicação paralela.

3.4.2 Configuração em Modo Passivo

No modo de reconfiguração passivo, um dispositivo microprocessado controla a reconfiguração do FPGA. Os dados de reconfiguração podem ser transferidos no modo passivo serial (*passive serial* – PS) ou no modo passivo paralelo (*passive paralell* – PP). No modo PS, os dados de reconfiguração são transferidos bit a bit. No entanto, no modo paralelo já é possível transferir cadeias de bits a uma taxa de 8, 16 ou até 32 bits [81]. Neste modo de reconfiguração, uma memória controlada por um dispositivo microprocessado é utilizada. É possível observar que, neste tipo de reconfiguração, o microprocessador controla a transferência dos dados e também gera o sinal de clock DCLK (Figura 24). Os dados são transferidos da memória externa para o FPGA, através do sinal data0. O dado de reconfiguração é configurado no *latch*[11] do FPGA, na borda de descida do sinal DCLK. A cada ciclo de clock, 1 bit de dado de reconfiguração é

[11] Um *latch* é um circuito sequencial biestável constituído por portas lógicas capaz de armazenar um bit de informação.

transferido, considerando-se o modo serial. Ao final da reconfiguração, o sinal bidirecional conf_done fica em nível alto, informando ao microprocessador que ao processo de reconfiguração finalizou. Os modos de reconfiguração ativo (serial/paralelo) e passivo (serial/paralelo) dependem diretamente do modelo de FPGA utilizado e podem ter variações de um fabricante para outro. Dessa forma, exigem esquemas de reconfiguração que levam em consideração as especificidades do modelo de FPGA utilizado.

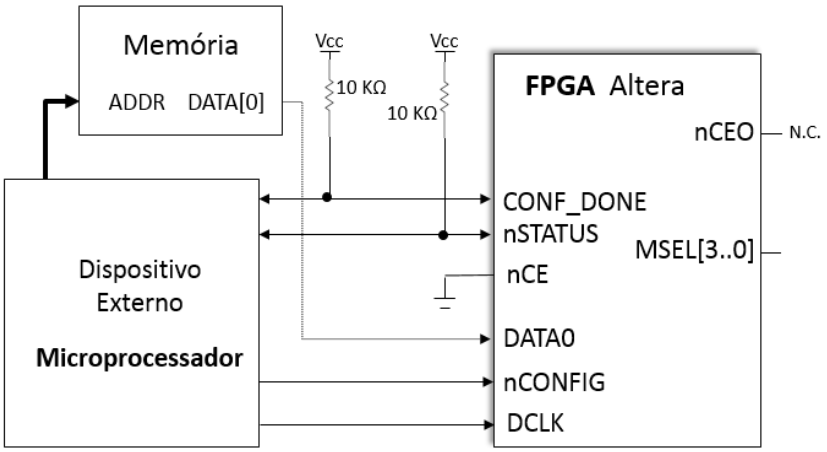

Figura 24 – Modo de configuração PS (Fonte [81]).

Neste trabalho, a escolha de um esquema de reconfiguração com maior portabilidade entre as famílias de FPGA disponíveis no mercado foi considerada. O modo de reconfiguração, através da interface JTAG, é suportado pela grande maioria dos FPGAs e será discutido com maiores detalhes na próxima Seção.

3.4.3 Configuração em Modo JTAG

A tecnologia JTAG surgiu em meados de 1980, juntamente com a necessidade de melhorar o acesso à lógica interna e a testabilidade dos circuitos microeletrônicos. A alta escala de integração dos circuitos estava tornando o teste, através dos testadores tradicionais, inviável, considerando-se o difícil acesso físico as conexões internas do circuito. As restrições de espaço e a dificuldade do acesso físico, entre milhares de interconexões dos circuitos microeletrônicos, fizeram com que o custo de detecção de falhas aumentasse drasticamente, enquanto a capacidade de detecção de falhas diminuía gradativamente. Em 1980, o grupo de pesquisa JTAG desenvolveu um método para testar interconexões de circuitos integrados, que foi padronizado como IEEE Std. 1149.1-1990 [80]. Esse método permitia acesso às interconexões dos circuitos, o que resultou em testes e validações mais simples. Em 1993, uma nova revisão ao padrão IEEE Std. 1149.1 foi introduzida (intitulada 1149.1ª), contendo mais detalhes, correções e

melhoramentos. Acompanhando o aperfeiçoamento do padrão, a *boundary-scan description language* (BSDL) foi adicionada em 1994 e, desde então, o padrão tem sido adotado pela indústria de dispositivos microeletrônicos [81, 82, 83].

Os circuitos microeletrônicos, que utilizam a tecnologia *boundary-scan* (BS), usam células do tipo BS conectadas diretamente aos pinos do circuito microeletrônico. As células BS são compostas basicamente por um multiplexador e um *latch*, conectados a cada pino do dispositivo. Com a utilização das células BS, é possível capturar dados dos pinos ou sinais lógicos internos do circuito, bem como forçar dados para a realização dos testes. Os dados capturados são serialmente deslocados (*shifted*) para fora do circuito e podem ser comparados externamente com os resultados esperados. Eles também podem ser deslocados serialmente para dentro das células BS, permitindo a "injeção" de estímulos de teste de uma forma mais simples. O controle do fluxo de dados é chamado de *scan path* ou *scan chain* [80]. A Figura 25 apresenta os principais elementos da tecnologia JTAG, que utiliza um conjunto de sinais para a operação das células BS.

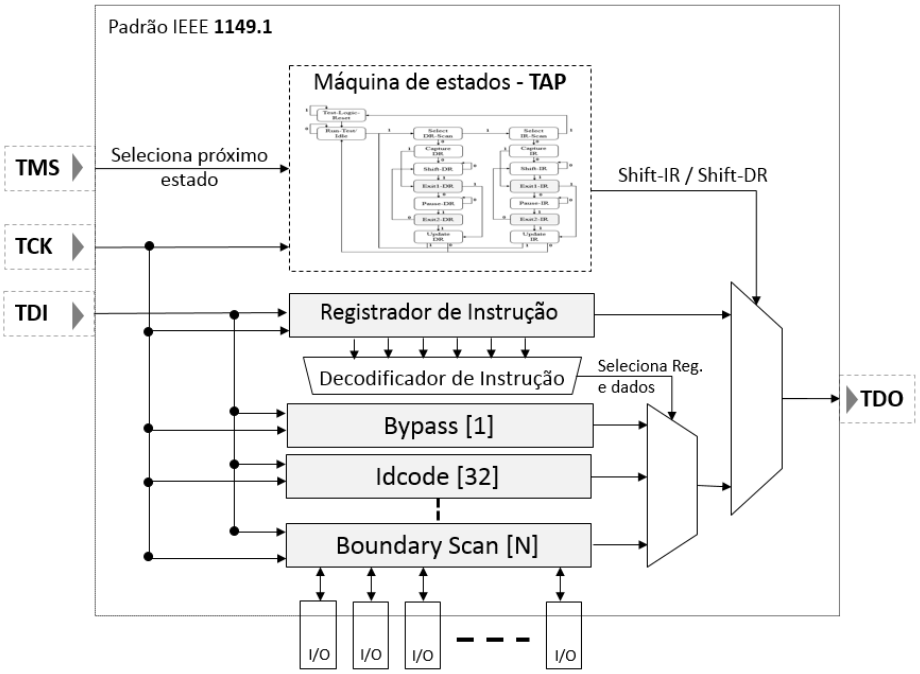

Figura 25 – Componentes de conformidade física JTAG padrão IEEE 1149.1 (Fonte [80]).

A *test access port* (TAP), por exemplo, é uma máquina de estados que gerencia o fluxo de entrada e saída de dados do circuito. Ela utiliza três sinais de entrada e um sinal de saída, de modo a gerenciar o fluxo de dados seriais utilizados durante o teste ou a reconfiguração do circuito. Entre os sinais, estão o *test clock* (TCK), o *test mode state* (TMS), o *test data in* (TDI) e

o *test data out* (TDO). Além desses sinais, há um sinal opcional que pode ser utilizado para reiniciar a TAP, chamado de *test reset* (TRST).

O TCK é um sinal gerado pelo *clock* interno do sistema e é utilizado para sincronizar as operações realizadas entre os estados da TAP. Já o sinal TMS é amostrado na borda de subida do sinal TCK e define qual é o próximo estado da TAP. O sinal TDI representa os dados que estão sendo inseridos (*input*) para a realização do teste do dispositivo. Este sinal é amostrado na borda ascendente do TCK, de acordo com o estado corrente da máquina de estados. Por fim, o sinal TDO representa os dados que estão sendo deslocados para fora do dispositivo, ou seja, os dados de saída, que são capturados na borda de descida do sinal TCK, de acordo com o estado corrente da máquina de estados. Existem dois tipos de registradores associados às células BS: cada dispositivo compatível tem um registrador de instrução e dois ou mais registradores de dados. O registrador de instrução armazena a instrução atual, que é utilizada pelo controlador da TAP para decidir o que fazer com os sinais que são recebidos.

É comum dizer que o conteúdo do registrador de instrução irá definir qual sinal de registrador de dados deve ser transmitido. Os registradores de dados são divididos em três registradores de dados primários: o bypass, o idcodes e o BS. Outros registradores de dados podem estar presentes, mas eles não precisam ser utilizados no padrão JTAG. O BS é o principal registrador e é utilizado para mover os dados dos pinos de entrada e saída (E/S) dos dispositivos para a lógica interna da célula BS. O registrador *bypass* é formado por um único bit e é utilizado para conectar o registrador TDI (que mantém os dados de entrada) diretamente ao registrador TDO (representa a saída da máquina de estados TAP). Essa conexão proporcionada pelo *bypass* permite que outros dispositivos sejam testados, com um *overhead* mínimo. E por fim, o registrador *idcode* contém o código de identificação (*ID code*) e o número de revisão do dispositivo. Esse dado é utilizado para que o sistema possa "conectar" o dispositivo à descrição BSD, o que fornece detalhes físicos das células BS utilizadas no dispositivo alvo.

Para a reconfiguração do FPGA, os arquivos de núcleos de hardware, nos formatos proprietários, podem ser encapsulados em padrões com suporte a JTAG. O formato SVF especifica um conjunto de comandos e uma sintaxe, de modo a implementar uma comunicação entre os sistemas de controle e os dispositivos físicos BS/JTAG. As características do formato SVF são detalhadas na próxima seção.

3.4.3.1 *Serial Vector File*

A sintaxe do arquivo SVF [78] é definida no padrão IEEE 1149.1 [80] e permite realizar diversas operações, através de uma interface JTAG. Em geral, as operações consistem na varredura (*scan*)

e controle dos movimentos entre os estados da máquina TAP. O SVF é utilizado como um formato de troca (comunicação) entre os programas, que geram padrões de entrada e saída JTAG, e os dispositivos, que utilizam uma interface física JTAG.

O SVF utiliza um formato de texto *american standard code for information interchange* (ASCII), com a extensão svf, sendo empregado por equipamentos de teste automático (*automated test equipament* – ATE), para a programação e verificação de um ou diversos dispositivos presentes em uma cadeia JTAG. A especificação SVF é constituída por um conjunto de comandos e instruções [84], utilizados no controle de dispositivos conectados através de uma interface JTAG. O anexo II apresenta a descrição dos comandos definidos na especificação SVF. Além da sintaxe formada por diversos comandos de controle e operações lógicas, o formato SVF utiliza uma nomenclatura própria para representar os estados da máquina de estados TAP. Os nomes do padrão SVF tem correspondência direta com os nomes dos estados utilizados no padrão IEEE 1149.1 (anexo III).

O SVF utiliza instruções que podem fazer com que a execução permaneça em um estado da TAP, por um determinado tempo, ou também dar saltos de execução entre um estado e outro. Dessa forma, é possível serializar dados através da lógica interna do circuito, com o objetivo de testar ou configurar o dispositivo. O arquivo de reconfiguração SVF é manipulado pela unidade de processamento, que vai realizar a decodificação (*parse*) do conteúdo do arquivo SVF e enviar os comandos e dados de teste ou reconfiguração, através do canal de comunicação física existente entre o *host* e *target*.

Em resumo, os conceitos abordados até aqui são necessários para o entendimento das atividades desenvolvidas em cada uma das etapas contempladas nesta metodologia. Os assuntos foram abordados de uma forma mais abrangente, com o objetivo principal de propiciar uma visão ampla da tecnologia e dos conceitos utilizados no desenvolvimento deste trabalho. As etapas que contemplam o uso desta metodologia serão discutidas com maiores detalhes, no próximo capítulo.

4. Metodologia de Reconfiguração de Hardware Utilizando o Sinal de TV Digital

A metodologia de reconfiguração de hardware proposta, através do sinal de TV Digital, contempla as etapas necessárias para a completa reconfiguração do dispositivo FPGA utilizado no receptor de TV digital. A metodologia considera transmissão do feixe de bits com o hardware pré-sintetizado, remontagem na unidade receptora e, por fim, reconfiguração do FPGA interligado ao receptor. Dessa forma, a metodologia foi subdividida em três etapas distintas: empacotamento do fluxo de hardware, remontagem do fluxo de hardware e reconfiguração do FPGA (Figura 26).

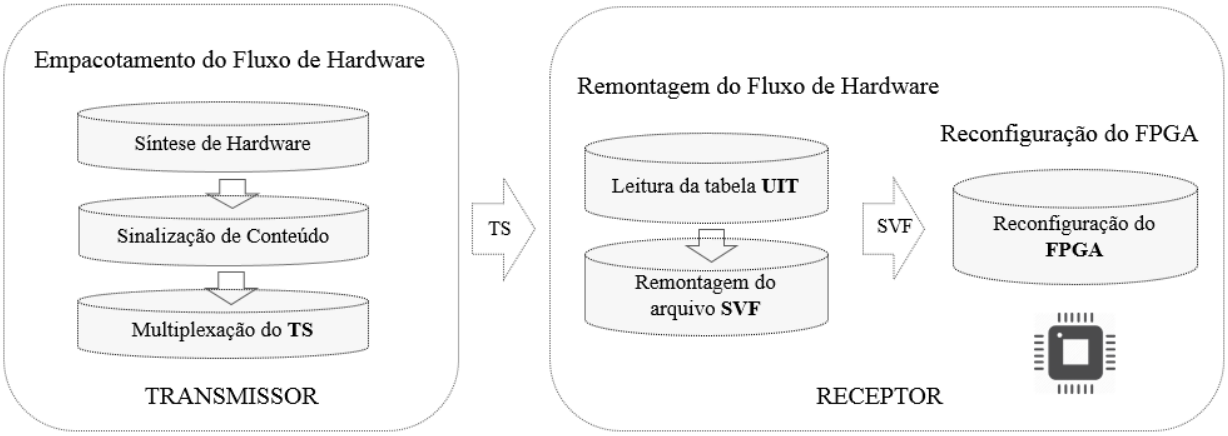

Figura 26 – Etapas envolvidas na metodologia de reconfiguração de hardware.

1. **Empacotamento do fluxo de hardware**: O empacotamento do conteúdo ocorre no transmissor e contempla os todos passos necessários, que vão desde a síntese do hardware até a multiplexação do conteúdo gerado em um TS. O processo inicia com a geração do núcleo (arquivo SVF), através de uma ferramenta de síntese de hardware para FPGA. O conteúdo do arquivo SVF é quebrado em fatias de dados hexadecimal e carregado em seções privadas, mantendo-se a sua ordem sequencial. Esse conteúdo de hardware é então descrito e sinalizado com a inserção de uma nova tabela de TV digital, chamada de tabela de informação de atualização, ou UIT, que sinaliza o conteúdo de hardware carregado, identifica as seções através de seus respectivos PID e table_id (TID) e descreve as características do hardware transmitido. O produto dessa etapa é um arquivo TS que carrega um feixe de hardware pré-sintetizado, sinalizado e multiplexado aos demais conteúdos de TV Digital (*e.g.,* áudio, vídeo e dados).

2. **Remontagem do fluxo de hardware**: A remontagem do fluxo de hardware é realizada pela aplicação residente na unidade receptora. Após sintonizar um canal, a aplicação primeiramente identifica se existe algum conteúdo de hardware sinalizado, através da tabela UIT. Após isso, o analisador da UIT extrai as informações do hardware transmitido. De posse dessas informações, é possível verificar se o conteúdo foi transmitido para o FPGA usado no receptor, o que é seguido pela remontagem das seções privadas, através do PID e do TID extraídos da tabela UIT. Ao final dessa etapa, o arquivo SVF estará remontado no sistema de arquivos do receptor.

3. **Reconfiguração do FPGA**: A reconfiguração do FPGA é realizada ao término da etapa anterior e consiste basicamente na utilização do arquivo SVF para reconfigurar o FPGA respectivo. Assim, o módulo de reconfiguração do sistema residente, instalado no *host*, primeiramente verifica se a interface de gravação (cabo de conexão física) do dispositivo FPGA está conectada e, em seguida, identifica as características do FPGA da unidade receptora e configura o canal de comunicação JTAG, com as informações do arquivo BSD. Após isso, inicia-se a análise do arquivo SVF que foi remontado, o que resulta em um processo de leitura e reconfiguração do dispositivo alvo. Esse processo só é finalizado quando o sinal TDO, da interface de comunicação JTAG, apresentar o nível alto. Nesse ponto, o FPGA está reconfigurado com uma nova funcionalidade de hardware.

Ao decorrer deste capítulo, as três etapas resumidas aqui serão apresentadas com mais detalhes, para o completo entendimento da metodologia proposta neste trabalho.

4.1 Etapa de empacotamento do fluxo de hardware

A etapa de empacotamento do núcleo de hardware consiste na multiplexação dos dados de reconfiguração em um TS e inicia com um núcleo pré-sintetizado para um dispositivo alvo (FPGA). A geração do núcleo de hardware, é discutida com mais detalhes na seção 3.3.3; aqui, considera-se apenas o resultado deste processo. A etapa de empacotamento inicia no passo 1 da Figura 27, onde é necessário coletar informações do processo de síntese, para preencher os campos da tabela UIT, e também utilizar o conteúdo do arquivo SVF como entrada do particionador. As informações referentes ao tipo de dispositivo FPGA, para o qual o projeto foi sintetizado, são reunidas na tabela UIT. Esta tabela contém toda informação necessária à

identificação do fluxo de reconfiguração de hardware, bem como a descrição do dispositivo de FPGA para qual o fluxo de bits foi sintetizado. Esse processo, também conhecido por sinalização de conteúdo, permite que receptores identifiquem quando uma atualização de hardware está sendo transmitida e também verifiquem, a partir das informações contidas na tabela UIT, se o conteúdo transmitido corresponde às características do dispositivo utilizado na unidade de recepção. É necessário propiciar, aos sistemas residentes das unidades receptoras, um meio de identificar os diversos conteúdos de dados que compõe o sinal de TV Digital. Dessa forma, é muito importante contextualizar os dados, de modo que que as unidades receptoras os utilizem de forma correta. Se os dados não forem identificados, eles podem ser vistos apenas como fluxo de dados binários presentes no sinal de TV digital, sem um propósito específico.

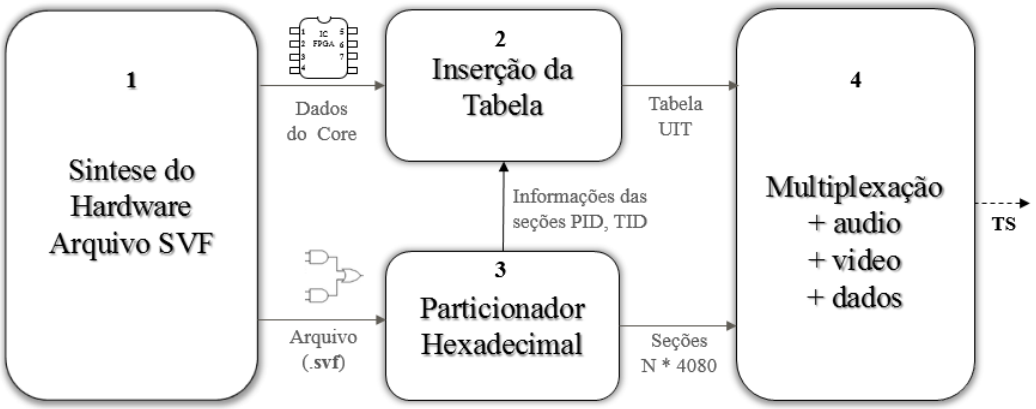

Figura 27 - Etapas para o empacotamento do fluxo de hardware.

Assim, é necessário primeiramente sinalizar, através a inserção da tabela e também da descrição da finalidade para a qual os dados estão sendo transmitidos. A Tabela 3 apresenta os campos da tabela UIT. A construção da tabela UIT foi baseada na tabela AIT, que faz parte do padrão de TV Digital e é utilizada na sinalização e descrição de aplicativos, que também podem ser transmitidos no padrão de TV digital. Na primeira coluna da tabela, o nome do campo e a ordem em que o campo deve ser filtrado são apresentados. Na segunda coluna, o número de bits utilizado na representação é mostrado. Logo em seguida, na terceira coluna, o identificador, que determina o tipo de dado carregado neste campo, é indicado. Os identificadores *unsigned integer most significant bit first* (uimsbf) e *bit string, left bit first* (bslbf) identificam o conteúdo que é armazenado no campo [38] e, por fim, a última coluna (valor) indica o conteúdo dos campos com valores padrões. Os campos vazios não têm valores padrões e dependem necessariamente do conteúdo que será inserido, durante o processo de multiplexação do TS. O identificador da tabela UIT é representado pelo campo TID de 8 bits que é disponibilizado na segunda linha da

tabela. Normalmente, as demais tabelas do padrão de TV digital também são identificadas por esse campo. Geralmente, filtros de seção presentes em receptores utilizam o campo *table_id* para localizar e filtrar tabelas no sinal de TV digital. Os demais campos são idênticos aos utilizados na tabela AIT [85], com exceção dos campos *fpga_core_number, hw_core_flag* e *update_loop_length.*

Tabela 3 - Update Information Table – UIT

Sintaxe dos dados da tabela	Taxa de bits (No. De bits)	Mnemônico	Valor
update_information_table(){			
table_id	8	uimsbf	0x91
section_syntax_indicator	1	bslbf	
hw_core_flag	1	bslbf	
reserved	2	bslbf	
section_length	12	uimsbf	
fpga_core_number	16	uimsbf	
reserved	2	bslbf	
version_number	5	uimsbf	
current_next_indicator	1	bslbf	
section_number	8	uimsbf	
last_section_number	8	uimsbf	
reserved_future_use	4	uimsbf	
common_descriptors_length	12	uimsbf	
for(i=0; i<N; i++){			
descriptor()			
}			
reserved_future_use	4	bslbf	
update_loop_length	12	uimsbf	
for(i=0; i<N; i++){			
update_hardware_identifier()			
reserved_future_use	8	uimsbf	
reserved_future_use	4	bslbf	
hw_descriptors_loop_length	12	uimsbf	
for(i=0; i<N; i++){			
descriptor()			
}			
}			
CRC_32	32	uimsbf	
}			

O campo de 16 bits *fpga_core_number* indica o número de núcleos de hardware (*cores*) que estão presentes (sendo transmitidos) no fluxo de transporte. A sintaxe da tabela UIT permite endereçar diversos núcleos de hardware, com a descrição do FPGA para o qual foram sintetizados. Essa característica permite que receptores possam ser equipados com FPGAs de

múltiplos fabricantes e também famílias distintas do mesmo fabricante. O *hw_core_flag é um campo* de 2 bits utilizado para indicar uma atualização de hardware. Embora a presença da tabela UIT já indique a existência de conteúdo de atualização, esse campo é utilizado para reforçar a sinalização e a comprovação da existência do hardware, para casos em que a tabela for transmitida sem o endereçamento de conteúdo de hardware. Em alguns casos, radiodifusores apenas transmitem informações/referências no sinal de TV digital, sem realmente adicionar o conteúdo relacionado, com o objetivo de alocar descritores/identificadores e indicar que aquele conteúdo pode ser disponibilizado, em algum momento. O valor deste "0x1" indica a presença de uma atualização de hardware.

A Tabela 4 expande o identificador *update_hardware_identifier* (), que contém os campos necessários à descrição das características de cada núcleo de hardware transmitido. O *update_hardware_identifier* () é caracterizado por um *loop*, onde cada iteração representa a descrição de um núcleo de hardware, de acordo com o número de núcleos informados pelo campo *fpga_core_number*. Esse identificador pode descrever o conteúdo de núcleos de múltiplos fabricantes e diferentes famílias de FPGA. O primeiro campo, *fpga_core_size*, contém o tamanho, em bytes, do núcleo de hardware transmitido no sinal (*e.g.,* 900845 bytes).

Tabela 4 - Sintaxe do campo *update_hardware_identifier* ()

Sintaxe dos dados do identificador	Taxa de bits (No. de bits)	Mnemônico	Valor
`update_hardware_identifier(){`			
`  fpga_core_size`	32	uimsbf	
`  fpga_core_version`	16	uimsbf	
`  fpga_core_module_name()`			
`  fpga_core_device_info()`			
`  fpga_section_identifier()`			
`  fpga_synthesis_results ()`			
`}`			

O campo *fpga_core_version* é um campo numérico de 16 bits, cujo objetivo é guardar a versão de atualização do núcleo de hardware transmitido. Além disso, com esse identificador, o receptor pode gerenciar a atualização de hardware do receptor. Os próximos campos são representados por descritores, com sintaxe definida, e trazem informações sobre as características do FPGA para qual o feixe de bits foi sintetizado. O primeiro descritor é o *fpga_core_module_name ()*, que é apresentado na Tabela 5 e é identificado pelo primeiro campo, o *descriptor_tag* com o valor "0x01". O próximo campo *descriptor_length* é um campo numérico de 8 bits contendo o tamanho, em bytes, do campo *reserved_future_use*, que representa

dados de caracteres de 24 bits e é reservado para uso futuro. Esse campo foi colocado para manter a sintaxe do descritor correspondente da tabela AIT. O campo *hardware_name_length* é um campo de 8 bits contendo o tamanho, em bytes, do campo *fpga_core_module_name*, que consiste em um campo com uma sequência de caracteres de 16 bytes contendo o nome do módulo de hardware transmitido (*e.g.,*16x2 lcd message).

Tabela 5 - Sintaxe do descritor *fpga_core_module_name* ()

Sintaxe dos dados do descritor	No.bits	Mnemônico	Valor
fpga_core_module_name(){			
descriptor_tag	8	uimsbf	0x01
descriptor_length	8	uimsbf	
for(i=0; i<N; i++){			
reserved_future_use	24	bslbf	
hardware_name_length	8	uimsbf	
for(i=0; i<N; i++){			
fpga_core_module_name	nx8	bslbf	
}			
}			
}			

O segundo descritor *fpga_core_device_info ()* é apresentado na Tabela 6 e contém a descrição do dispositivo para qual o feixe de bits foi sintetizado. Nessa descrição, é possível encontrar o nome do fabricante, a família do FPGA e o *part number* (número de identificação do dispositivo). O sistema residente do receptor compara essas informações às do FPGA instalado. Esse descritor é identificado pelo campo *descriptor_tag* com valor 0x03. O campo *fpga_core_device_info* é um campo que armazena uma sequencia de caracteres de 27 bytes contendo as informações sobre o dispositivo alvo para qual o feixe de bits foi sintetizado (*e.g.,* altera stratix ep1s10f780c6)

Tabela 6 - Sintaxe do descritor *fpga_core_device_info ()*

Sintaxe dos dados do descritor	No.bits	Mnemônico	Valor
fpga_core_device_info(){			
descriptor_tag	8	uimsbf	0x03
descriptor_length	8	uimsbf	
for(i=0; i<N; i++){			
fpga_core_device_info	nx8	bslbf	
}			

O terceiro descritor, seguindo-se a ordem da Tabela 4, é o *fpga_core_device_info ()*, que fornece o ponto de acesso para a filtragem das seções privadas (ver anexo I) do fluxo binário de

reconfiguração. Esse descritor é representado pelo valor 0x05 no primeiro campo de 8 bits *decriptor_tag*, apresentado na Tabela 7. O próximo campo do descritor é o *decriptor_length*, de 8 bits, que é utilizado para armazenar o tamanho, em bytes, dos próximos campos do *loop*. Entre os campos necessários para identificar seções que carregam o núcleo de hardware, está o campo *remount_core_pid*, de 32 bits, que contém o *packet identifier* (PID) dos pacotes que compõem as seções. Em seguida, o campo *remount_core_section_tid*, de 16 bits, contém o valor de identificação da seção, ou seja, o TID da seção privada. Com esses valores, o filtro de seção pode ser configurado para filtrar todas as seções necessárias. Para casos em que o tamanho do núcleo de hardware ultrapassa 256 seções, que é o limite máximo permitido pela norma, o campo de 8 bits *remount_priority* deve ser utilizado, de modo a se determinar a ordem de remontagem entre os grupos de seções. Por padrão, este valor é inicializado em '0', representando o primeiro grupo de seções identificados por seus respectivos PID e TID. Para o segundo grupo de seções, representado pelos seus respectivos valores de PID e TID, o valor deste campo é incrementado para '1', de modo que a ordem de leitura das seções seja respeitada, até que todo o conteúdo do feixe de bits tenha sido carregado.

Tabela 7 - Sintaxe do descritor *fpga_section_identifier ()*

Sintaxe dos dados do descritor	No.bits	Mnemônico	Valor
fpga_section_identifier(){			
descriptor_tag	8	Uimsbf	0x05
descriptor_length	8	Uimsbf	
for(i=0; i<N; i++){			
remount_core_pid	32	uimsbf	
remount_core_section_tid	16	uimsbf	
remount_priority	8	uimsbf	
}			
}			

O último descritor da Tabela 4 é o *fpga_synthesis_results()*, que fornece informações sobre o processo de síntese, de acordo com o dispositivo FPGA utilizado. Esses valores são obtidos através dos resultados do processo de síntese de hardware, fornecidos a partir das ferramentas de projeto de FPGA disponibilizadas pelos fabricantes. O descritor, apresentado na Tabela 8, é identificado pelo valor 0x09, o que é disponibilizado pelo primeiro campo de 8 bits, o *descriptor_tag*. O próximo campo é o *descriptor_length_elements,* de 8 bits, que contém o tamanho, em bytes, do campo que está dentro do *loop fpga_used_logic_elements* e consiste em uma sequência de caracteres de 255 bytes. Ele contém informações sobre a quantidade de

elementos lógicos que foram utilizados, durante a síntese do núcleo de hardware, considerando-se o total de elementos disponíveis no dispositivo alvo (*e.g.*, 222 / 10,570 2%).

Tabela 8 – Sintaxe do descritor *fpga_ synthesis_results ()*

Sintaxe dos dados do descritor	No.bits	Mnemônico	Valor
fpga_synthesis_results(){			
descriptor_tag	8	uimsbf	0x09
descriptor_length_elements	8	uimsbf	
descriptor_length_pins	8	uimsbf	
for(i=0; i<N; i++){			
fpga_used_logic_elements	nx8	bslbf	
fpga_used_logic_pins	nx8	bslbf	
}			

No exemplo apresentado, 222 elementos, de um total de 10,570, foram utilizados, ou seja, apenas 2% do total de elementos lógicos disponíveis nesse modelo de FPGA. O próximo campo é o *descriptor_length_pin*, de 8 bits, que contém o tamanho, em bytes, do próximo campo do *loop* de caracteres. O campo *fpga_used_logic_pins* representa uma sequência de caracteres de 255 bytes, que informa a quantidade de pinos de I/O utilizados na síntese do feixe de bits, considerando o total disponível no dispositivo alvo (*e.g.*, 8 / 427 2%). No exemplo apresentado, a síntese utilizou 2% do total de pinos disponíveis no dispositivo.

O particionamento do núcleo de hardware é realizado em conjunto com o preenchimento da tabela UIT, de acordo com a sintaxe que foi apresentada anteriormente. O particionamento ocorre na etapa 3 da Figura 27, onde o arquivo SVF é convertido em hexadecimal e quebrado em seções de 4080 bytes. Esse tamanho é o máximo permitido pela área de carga da seção privada [38], onde cada fatia do arquivo SVF será carregada. O campo *private_data_byte*, da sintaxe da seção privada apresentado na Tabela 9, representa a área de carga de cada seção privada. Nessa estratégia, é imprescindível manter a ordem de inserção do conteúdo, o que é obtido através do campo *section_number*. A ordem das seções é fundamental para que a coerência dos dados seja mantida. Se pelo menos uma única seção estiver invertida, o núcleo de hardware ficará corrompido, o que resultará em uma falha na reconfiguração do FPGA. Dessa forma, o papel do particionador é assegurar a contiguidade numérica das seções, durante a inserção do conteúdo.

Outra informação importante é o número da última seção, que é mantida pelo campo de 8 bits *last_section_number*. Tal campo permite uma contagem máxima de 256 seções, para cada grupo de seções que utiliza os mesmos TID e PID. Cada grupo de 256 seções permite uma carga de 1044480 bytes (4080*256). Caso o tamanho do núcleo de hardware ultrapasse este valor, é necessário inserir mais um grupo de seções, com diferentes identificadores PID e TID, e assim

sucessivamente. A estratégia utilizada para manter a ordem de remontagem dos grupos de seções é dada pelo campo *remount_priority*, do descritor *fpga_section_identifier ()* presente na tabela UIT, que mantem uma lista dos identificadores (PIDs e TIDs) de cada grupo de seções privadas (Tabela 7).

Na Tabela 9, é possível ver que a última coluna contém valores fixos e também limites de intervalo permitido, o que depende do campo. Os campos mais utilizados são o *table_id*, que permite valores no intervalo de 0x80 até 0xFE, o campo *section_number*, que informa o número da seção privada, e o campo *last_section_number*, que informa o número da última seção que contém carga do núcleo de hardware. O campo *private_data_byte* contém o conteúdo do núcleo de hardware da seção, com tamanho de até 4080 bytes de valores hexadecimais.

Tabela 9 - Sintaxe da seção privada com valores *default* em alguns campos

Sintaxe dos dados do descritor	No.bits	Mnemônico	Valor
private_section(){			
table_id	8	uimsbf	[0x80 até 0xFE]
section_syntax_indicator	1	bslbf	1
private_indicator	1	bslbf	1
reserved	2	bslbf	0x3
private_section_length	12	uimsbf	4089
section_syntax_indicator=1			
table_id_extension	16	uimsbf	0x0001
reserved	2	bslbf	0x3
version_number	5	uimsbf	[0 até 31]
current_next_indicator	1	bslbf	1
section_number	8	uimsbf	0
last_section_number	8	uimsbf	220
private_data_byte	nx8 bits	bslbf	{núcleo de hardware} 4080 bytes 0x21436F7079726967687 420284329203139393120D 32303131204167465726 120436F72706F72617469 6F6E0D0A21596F7572207 57365206F6620416C7465 6F6620416F66204162041 ...
CRC_32	32	rpchof	0x6F484DA5
}			

As ferramentas de manipulação de TS adicionam um valor de checagem para cada seção privada, de acordo com os dados inseridos na carga das seções. O campo de 32 bits *CRC_32* é utilizado para que o sistema residente do receptor possa identificar erros. Tal verificação é

fundamental para que o sistema de recepção possa validar o conteúdo transmitido. No exemplo apresentado na Tabela 9, o campo *CRC_32* está carregando o valor *0x6F484DA5*, que será utilizado pelo receptor para o *checksum* (ver capitulo 3.1.6).

Após a carga do conteúdo nas seções privadas e a geração da tabela UIT, com as informações referentes ao conteúdo transportado, a etapa de multiplexação do fluxo de transporte é iniciada. Na multiplexação (etapa 4 da Figura 27), as seções privadas e a tabela de descrição do conteúdo de hardware (UIT) são multiplexadas a os demais conteúdos de HDTV (áudio, vídeo e dados), formando um único fluxo de transporte. Ao final desse processo, é possível utilizar um modulador terrestre ISDB-T para a transmissão do fluxo de transporte, em um determinado canal de TV digital.

4.2 Etapa de Remontagem do Fluxo de Hardware

A etapa de remontagem do núcleo de hardware ocorre do lado do receptor de TV digital, onde o fluxo de hardware contido no sinal de TV precisa ser extraído e utilizado no processo de reconfiguração. Nesse sentido, a recepção destes dados contempla o desenvolvimento de subsistemas de software responsáveis pela identificação e remontagem do núcleo de hardware. Esta etapa inicia após a identificação do conteúdo sinalizado com o filtro da tabela UIT. O sistema residente, executando no contexto do receptor (*cross-compiled*), deve receber o fluxo de dados e filtrar as seções de seus respectivos PIDs e TIDs, para então extrair os dados de reconfiguração.

Após a completa extração dos dados, é necessário realizar a persistência destes, em memória permanente. Primeiramente, é necessário que a aplicação sintonize um canal de TV digital. Assim, a aplicação residente permitirá que o usuário navegue entre os canais de TV digital, utilizando o navegador (*zapper*) da aplicação residente. O *zapper* percorre o espectro de frequência, em busca de sinais existentes e, nesse processo, aciona o *tuner (frontend)* do receptor. Ao encontrar um sinal, o *tuner* trava na respectiva frequência e inicia o processo de demodulação, que tem como resultado um TS, como mostrado na Figura 28 (passo 1). No passo 2, o sistema residente realiza tentativas de filtrar a tabela periódica UIT, com o objetivo de identificar a existência de algum fluxo de hardware. Se a tabela UIT não existe, o sistema sinaliza que não há conteúdo de reconfiguração de hardware. Por outro lado, se a tabela UIT é identificada, o sistema verifica (passo 3), primeiramente, o estado do campo *hw_core_flag* presente na sintaxe da tabela UIT (ver seção 4.1). Se ele estiver com o valor 0x0 (*off*), então nenhum hardware está sendo transmitido. Nesse caso, o sistema simplesmente ignora a filtragem

de dados. Por outro lado, se *hw_core_flag* contiver o valor 0x1 (*on*), isso significa a existência de dados de reconfiguração.

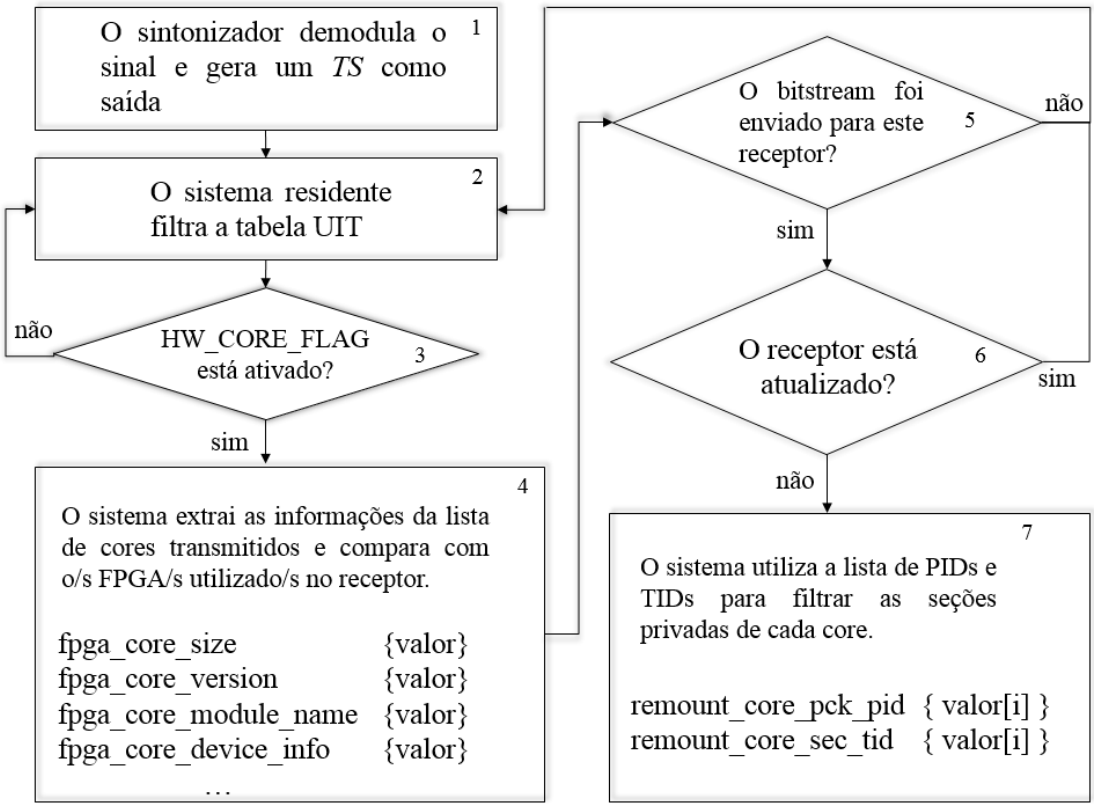

Figura 28 – Fluxograma de extração dos dados da UIT e comparação com o modelo de FPGA do receptor.

Neste caso, o sistema residente continua fazendo o *parser* da tabela UIT, em busca (passo 4) dos demais campos com informações referentes ao núcleo de hardware, como o tamanho do núcleo (*fpga_core_size*), a versão do núcleo transmitido (*fpga_core_version,* ex. 0x12), o nome do módulo a ser reconfigurado (*fpga_core_module_name*), e as informações do dispositivo para o qual o núcleo de hardware foi sintetizado (*e.g.*, ALTERA STRATIX EP1S10F780C6), que são disponibilizadas pelo campo *fpga_core_device_info*. Além disso, o sistema extrai, do descritor *fpga_section_identifier()*, a lista de identificadores (PID e TID) e informações necessárias à filtragem de seções privadas, que estão multiplexadas no conteúdo do sinal. Após a captura dos dados, o sistema segue para o passo 5, onde verifica se o núcleo de hardware foi enviado ao receptor. A identificação é realizada através da comparação entre os dados recebidos (descritor *fpga_core_device_info* da tabela UIT) e os do FPGA instalado no receptor, como Fabricante, Família e Modelo (*part number*). Vale ressaltar que o receptor pode aceitar uma lista de módulos para reconfiguração.

Caso os dados recebidos não sejam compatíveis com as características do FPGA instalado no receptor, o sistema mantém o ciclo de verificação da tabela UIT. Entretanto, se os dados forem compatíveis, a linha de execução do sistema vai para o passo 6, onde compara a versão de atualização (*fpga_core_version*) à versão do receptor. Se já houve uma atualização, o sistema mantém o ciclo de verificação de dados sinalizados com a leitura da tabela UIT; caso contrário, o sistema segue a linha de execução dada pelo passo 7. Tais valores representam os pontos de acesso ao fluxo de hardware contido no sinal de TV. Com a lista de PIDs e TIDs, os filtros de seção do receptor podem ser reconfigurados, o que incia o processo de filtragem das seções privadas. A Figura 29 ilustra o processo de remontagem das seções privadas.

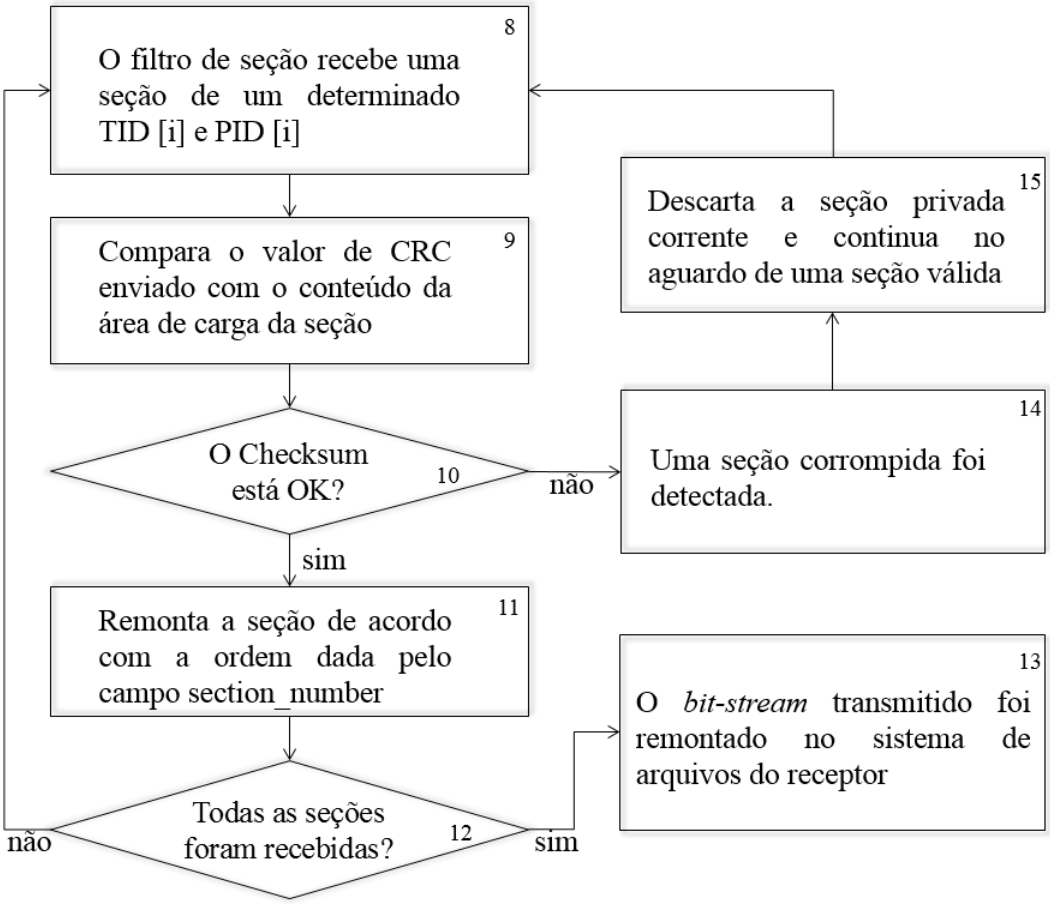

Figura 29 – Fluxograma descrevendo a remontagem do núcleo de hardware

No passo 8, o filtro de seções do receptor retorna uma dada seção associada ao PID e TID correntes. Após isso, no passo 9, é realizado o *checksum* (validação), com o objetivo de detectar erros no conteúdo extraído da área de carga da seção. Esta verificação é possível com a extração do valor do campo *CRC_32*, da sintaxe da seção privada. Este valor deve corresponder ao obtido pelo algoritmo de detecção de erros, cuja comparação ocorre no passo 10. Se os valores

estiverem corrompidos, a linha de execução segue pelo passo 14 e a seção é descartada, no passo 15, enquanto o sistema fica no aguardo de uma seção válida (passo 8). As seções são repetidas a uma determinada taxa utilizada no processo de multiplexação do fluxo de transporte e, dessa forma, o sistema tem autonomia para descartar seções corrompidas e aguardar seções que estejam corretas. Entretanto, se a seção estiver correta, a linha de execução segue para o passo 11 e a seção é remontada, de acordo com a contiguidade dada pelo campo *section_number*. Este processo é repetido até que todas as seções da lista de PIDs e TIDs tenham sido remontadas, em um único arquivo. Para garantir que todo o conteúdo do núcleo de hardware foi remontado, o sistema utiliza como base o tamanho, em bytes, do núcleo de hardware informado pelo campo *fpga_core_size*, que faz parte da lista de informações disponibilizadas pela tabela UIT, durante o processo de sinalização de conteúdo. O arquivo remontado no sistema de arquivos pode então ser usado para reconfigurar o FPGA do receptor.

4.3 Etapa de Reconfiguração do Dispositivo Alvo

Para a realização da etapa de reconfiguração do dispositivo alvo, é necessário que a conexão física e o modo de programação do FPGA já tenham sido previamente estabelecidos. A conexão física faz parte da implementação necessária para se realizar a prova de conceito da metodologia de reconfiguração de hardware proposta e é discutida, com mais detalhes, na próxima seção.

4.3.1 Esquema de reconfiguração *host/target*

O desenvolvimento do módulo de reconfiguração teve por base referências disponíveis na literatura [86, 87, 88, 89, 90]. O módulo estabelece comunicação física com a placa FPGA, através de um cabo de programação JTAG disponibilizado pelo fabricante. O cabo de programação é equipado com o dispositivo *future technology devices international* (FTDI), para a conversão de sinais USB para Serial [91]. Dessa forma, é possível utilizar a porta USB disponível no receptor de referência para a realização da comunicação física. As referências utilizadas neste trabalho [85, 86, 87, 91] trazem exemplos de implementação relativos a drivers e comunicação USB. Assim, é possível manter compatibilidade com interfaces usadas em placas de FPGA de diferentes modelos.

Há código de referência para a leitura de arquivos BSD, que contém as características das células e interconexões da estrutura BS do FPGA. Assim, para utilizar outro modelo de FPGA, é necessário configurar os parâmetros BS com as informações disponibilizadas pelo arquivo BSD, do dispositivo utilizado. Os arquivos BSD de cada modelo de FPGA, que utilizam a tecnologia *boundary scan*, são disponibilizados para download nos sites dos fabricantes de FPGA. A Figura

30 ilustra o diagrama do esquema de comunicação entre o receptor de TV digital e a placa FPGA. Uma aplicação residente (APP) gerencia o processo de reconfiguração do FPGA, utilizando o arquivo remontado na seção anterior. Os detalhes da aplicação residente são discutidos na seção 4.4.

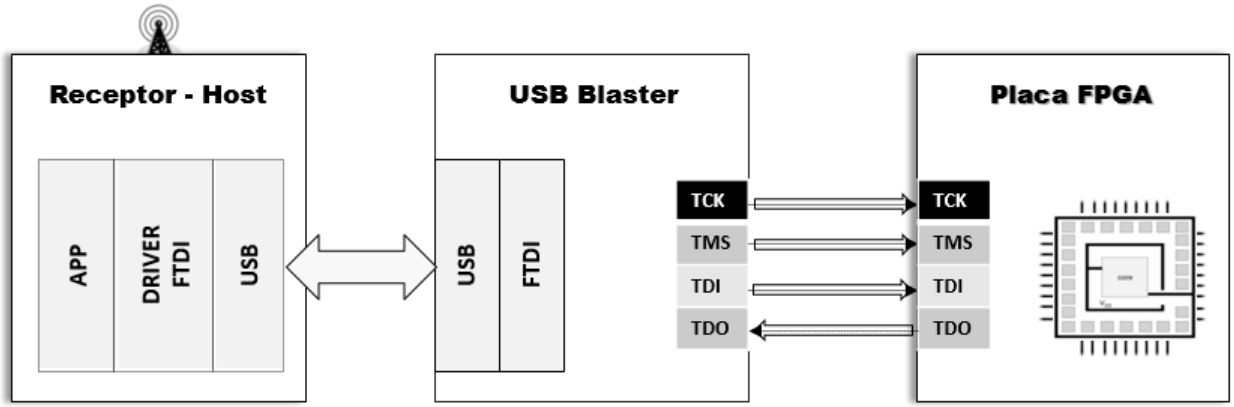

Figura 30 – Comunicação entre *host/target* através do cabo de conexão física USB *blaster*

O primeiro passo do módulo de reconfiguração é estabelecer comunicação com a placa de FPGA através do cabo de comunicação do fabricante, Altera USB Blaster. Dessa forma, as bibliotecas de link dinâmico *libusb.so*, *libcompat.so e libftdi.so* foram integradas no sistema de arquivos do receptor (*buildroot*). A *libusb.so* disponibiliza uma interface com métodos para desenvolver a comunicação com a porta USB. A *libcompat.so* é uma dependência necessária à instalação da *libftdi* que, por fim, disponibiliza uma interface de comunicação com o dispositivo FTDI, que foi empregado na comunicação USB-serial do cabo de comunicação. As bibliotecas integradas ao sistema do receptor foram customizadas, de modo a resolver problemas de dependências não disponibilizadas na versão do sistema Linux da plataforma de teste. O cabo de comunicação converte os sinais enviados, pelo *host,* para sinais compatíveis com JTAG e recebe sinais do *target* através da mesma interface.

Com a conexão física estabelecida, o procedimento de reconfiguração consiste basicamente em se efetuar o parser do arquivo SVF, que, por sua vez, permite reconfigurar o FPGA, através do modo JTAG. Dessa forma, a leitura do arquivo só termina com a reconfiguração completa do FPGA e, ao término desta, o sinal no pino TDO (ver capitulo 3.4.3) vai para o nível alto ‘1’, indicando ao sistema que o processo de reconfiguração chegou ao fim.

Em resumo, todas as etapas da metodologia de reconfiguração de hardware, através do sinal de TV digital, foram apresentadas. Como prova de conceito, um sistema embarcado foi desenvolvido, no contexto de um receptor de TV digital, com o objetivo de validar a

metodologia e todas as etapas envolvidas nesse processo. O sistema embarcado funciona como um sistema de navegação (*zapper*) entre os canais de TV, ou seja, o objetivo é replicar o comportamento de um sistema de decodificação de TV digital. A prova de conceito visa demonstrar a viabilidade técnica desta solução, em equipamentos de uso comercial.

4.4 Sistema Embarcado de Reconfiguração de Hardware

O sistema embarcado desenvolvido neste trabalho foi utilizado como prova de conceito (*prof of concept* – PoC) e tem por objetivo demonstrar o uso da metodologia proposta, no que diz respeito à etapa de remontagem do fluxo de dados, que foi detalhada na seção 4.2. Em resumo, o sistema visa demonstrar a viabilidade técnica dos procedimentos de remontagem e reconfiguração de FPGA, em um dispositivo comercial.

O sistema foi desenvolvido para o receptor de referência utilizado neste trabalho, que é uma plataforma de referência (*reference design*), modelo STB225, fornecido pelo fabricante de semicondutores NXP. A plataforma oferece o suporte necessário à manipulação e ao tratamento do sinal de TV digital. As ferramentas e o ambiente de desenvolvimento, fornecidos pelo fabricante, foram necessários à compilação cruzada do código fonte da aplicação, que utiliza a linguagem C, para rotinas de baixo nível, e Qt/C++ para controle de aplicação, tratamento de eventos, comunicação entre processos (*inter-process communication* – IPC), interação com usuário e desenvolvimento da interface gráfica (*general user interface* – GUI). O usuário interage com o sistema através do controle remoto da plataforma e possibilita navegar entre os canais de TV digital, simulando o comportamento de um *zapper*. A sintonia de canal é realizada com o auxílio do hardware demodulador (*frontend*). O sistema de demodulação realiza a sintonia do canal, quando há um sinal transmitido. Quando o sistema sintoniza um canal, o *frontend* passa a produzir um TS na saída do demodulador. A aplicação utiliza 5 etapas (lock, es/play, uit/hw, crc/rmt e rec/dev) para demonstrar os passos necessários até a completa reconfiguração do FPGA, o que está ilustrado na GUI da Figura 31. O primeiro estado é o *lock*, que indica o momento em que um canal de TVD foi sintonizado. Essa alteração é visualizada na GUI, com a mudança da cor do identificador, que no momento da sintonia passa a ser azul (o normal é cinza). A partir desse ponto, o sistema entra automaticamente no estado es/play (*elementary stream/play*), que consiste na reprodução dos feixes elementares de áudio e vídeo multiplexados no TS. A aplicação realiza então a filtragem das tabelas (*e.g.,* PAT, PMT, etc) de PSI, obtendo acesso aos pacotes que carregam os feixes elementares. Para acessar a tabela PMT, primeiramente o conteúdo da tabela PAT (ver a sintaxe do método *parsePAT ()* – anexo IV) é analisado.

Figura 31 – Interface gráfica (GUI) da aplicação residente HW/REC

Com os identificadores, a aplicação direciona os fluxos de áudio e vídeo para seus respectivos decodificadores, reproduzindo o vídeo na área de reprodução de mídias da GUI. Após a inicialização da decodificação de áudio e vídeo, o sistema passa para a próxima etapa, que é a verificação da existência da tabela UIT, o que é representado, na GUI, por uit/hw (*table/hardware*). Se a tabela estiver presente, a aplicação atualiza a GUI, mudando a cor do identificador de cinza para azul. Nessa etapa, o sistema faz a análise da tabela UIT e extraí as informações necessárias para dar prosseguimento no processo de reconfiguração. As informações extraídas da UIT são então apresentadas no lado direito da GUI, abaixo do identificador "--'UIT' *Hardware details* --". A etapa crc/rmt (*checksum/remounting*) realiza a verificação de CRC e remonta as *seções privadas*, que transportam o conteúdo de hardware. Esse processo só termina quando todas as seções relacionadas ao conteúdo sinalizado são remontadas, e o seu andamento é ilustrado através das barras de progresso (*check sum progress* e *remount progress*). Os resultados dos processos de *checksum* e remontagem são apresentados abaixo do identificador --'RMT' *Hardware details* --. O último passo é o rec/dev (*reconfiguration/device*), que acontece assim que a remontagem do arquivo SVF é finalizada. Nessa etapa, a comunicação com a interface física JTAG será estabelecida e os dados de reconfiguração do FPGA serão enviados. Inicialmente, o andamento da análise do arquivo SVF é apresentado na barra de

progresso "*Parsing SVF/RAW*" e, após isso, a reconfiguração é iniciada, enquanto as instruções JTAG são extraídas. O andamento desse processo é visualizado na barra de progresso "*FPGA reconfiguration*", ilustrada na GUI no momento em que estiver realizando a reconfiguração. Esse procedimento é finalizado com a reconfiguração completa do FPGA, cujos resultados são apresentados abaixo do identificador da GUI "--'REC' *Hardware details* --".

No próximo capítulo, os experimentos e também os resultados obtidos com a transmissão, a recepção e a reconfiguração de núcleos de hardware, de acordo com a metodologia descrita nos capítulos anteriores, são apresentados.

5. Experimentos e Resultados

Os experimentos foram realizados com base na metodologia proposta e contemplam as três etapas apresentadas no capítulo anterior. O propósito deles é medir o desempenho do sistema e também demonstrar a viabilidade técnica da metodologia, em um ambiente real de transmissão e utilizando um canal de TV digital terrestre e um receptor comercial. Como prova de conceito, alguns projetos de circuitos foram gerados para o FPGA, que está conectado ao receptor.

Para a etapa de empacotamento do fluxo de hardware, exemplos de circuitos (*designs*) pré-sintetizados serão utilizados, segundo os critérios de empacotamento apresentados na seção 4.1 do capítulo anterior. Com relação às etapas de remontagem do fluxo de hardware e reconfiguração do FPGA, apresentadas nas seções 4.2 e 4.3 do capítulo anterior, respectivamente, o sistema embarcado de reconfiguração de hardware apresentado na seção 4.4, do capítulo anterior, será utilizado.

5.1 Experimentos sinalizando hardware para o receptor alvo

Os núcleos utilizados nestes experimentos foram sintetizados para um dispositivo da Altera, modelo *Stratix* (EP1S10F780C6) [92], que está conectado ao receptor de referência. Para a síntese destes circuitos, a ferramenta de CAD Quartus-II versão 11.0 foi utilizada, a qual é disponibilizada para o desenvolvimento de projetos para dispositivos do fabricante Altera. O Quartus-II aceita projetos de entrada nos formatos VHDL, Verilog ou nível RTL. O resultado da síntese é gerado em diversos formatos de arquivos, como o formato para uso em modo JTAG, com um arquivo SVF. Após o ciclo de síntese, os projetos são verificados diretamente no hardware da placa de FPGA.

A Figura 32 demonstra o projeto de um contador de 4 bits, que apresenta o resultado da contagem utilizando 4 diodos emissores de luz (*light emitting diode* - LED), disponibilizados na placa de desenvolvimento. O resultado do processo de síntese foi um arquivo SVF (Ex01.svf).

Após a síntese, é necessário fazer a carga do conteúdo do arquivo SVF em um fluxo de transporte. Esse procedimento segue as regras de empacotamento apresentadas na seção 4.1. Assim, o conteúdo do arquivo SVF é empacotado em seções privadas e multiplexado juntamente com a inclusão de uma tabela UIT, que sinaliza e disponibiliza as informações do conteúdo de hardware. O resultado desse processo é um arquivo TS (Ex01.ts), que transporta o conteúdo de hardware pré-sintetizado do arquivo SVF.

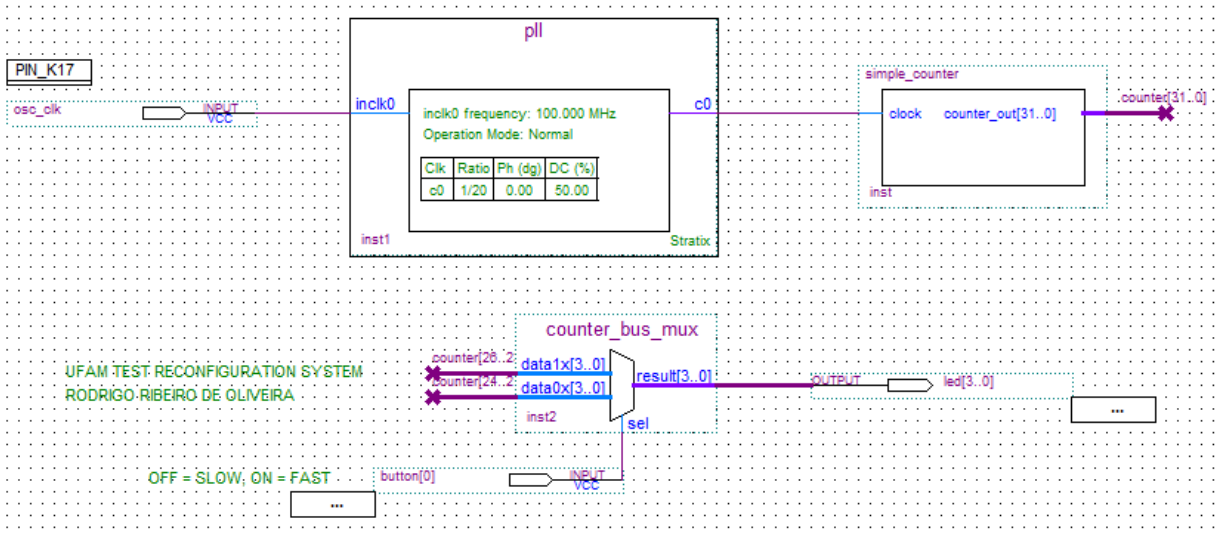

Figura 32 – Contador LED de 4-bit (Ex01.svf).

As características do Ex01.ts são apresentadas na Tabela 10, onde a coluna $CORE_{SINALIZAÇÃO\text{-}INFO}$ contém informações sobre o processo de sinalização, que ficam armazenadas na tabela UIT. Nessa coluna, é possível encontrar o tamanho, em bytes, do núcleo de hardware, o nome do fabricante do FPGA, a família de FPGA para qual o núcleo de hardware foi sintetizado, o modelo de FPGA (*Part Number*) e o nome do módulo de hardware transmitido.

A coluna $TS_{MULTIPLEXAÇÃO\text{-}INFO}$, contém informações sobre o processo de multiplexação, apresentando o tamanho e a taxa de dados do fluxo de transporte, sendo este último dado pelo campo "tx. de dados do TS", o tamanho do fluxo elementar de vídeo é informado pelo campo "vídeo (tamanho)", a taxa de compressão do vídeo é dada pelo campo "vídeo (tx. bits)" e, por fim, o campo "*stream loop time*" demonstra o tempo, em segundos, até o reinício do ciclo de transmissão do fluxo de transporte. Para o processo de multiplexação, a ferramenta de geração de fluxo de transporte Rohde&Schwarz - *advanced stream combiner*, versão 03.10, foi utilizada.

Após o encapsulamento do hardware no fluxo de transporte, já é possível realizar a modulação e transmissão desses dados, através de um determinado canal de TV digital.

Tabela 10 - Característica do exemplo Ex01.ts.

TS_{NAME}	$CORE_{SINALIZAÇÃO\text{-}INFO}$		$TS_{MULTIPLEXAÇÃO\text{-}INFO}$	
Ex01.ts	TAMANHO	900.845 bytes	TAMANHO	70,806 MB
	FABRICANTE	ALTERA	TX. DE DADOS DO TS	1.571.374 Bit/s
	FAMILIA	STRATIX	VIDEO (TAMANHO)	10.063.944 bytes
	MODELO	EP1S10F780C6	VIDEO (TX. BITS)	1.429 Mbit/s
	MÓDULO	4-BITS BCD-CNT	*STREAM LOOP TIME*	56.360 s

Para que a modulação do fluxo de transporte seja realizada, utiliza-se um modulador USB-2 multi padrão modelo Dectek DTU-215 [93]. Este modulador terrestre funciona com o auxílio de um computador e conectado através da porta USB, com alcance aproximado de 50m. O fabricante Dectek fornece ferramentas de software utilizadas em manipulação, edição e reprodução (*player*) de TS, chamadas *StreamXpress*. O pacote *StreamXpress* reproduz um TS de cada vez e gerencia a configuração dos parâmetros do modulador (no padrão ISDB-T), durante o processo de transmissão. A configuração do modulador é importante, tendo em vista que *tuners* de receptores brasileiros estão preparados para demodular sinais configurados de acordo com o SBTVD. A Tabela 11 apresenta os valores que foram utilizados durante a configuração do *StreamXpress*. A faixa de frequência utilizada deve estar dentro dos limites usados na TV Digital e é importante não configurar o modulador em canais usados por emissoras comerciais (interferência cocanal).

Tabela 11 - Parâmetros de modulação utilizados durante os testes.

Padrão da Modulação	**ISDB-T**		
Frequência (MHz)	485.143 (Canal 15)		
Taxa de bits de saída (bps)	29.958.294		
Intervalo de guarda	1/8		
Camadas	**A**	**B**	**C**
Segmentos	13	0	0
Modulação	64QAM	64QAM	64QAM
Code Rate	¾	3/4	3/4
Time Interleave	2	2	2
Taxa (bps)	18.255.835	0	0
Taxa selecionada (bps)	9.178.912	0	0

Os primeiros experimentos foram realizados para definir a taxa de repetição cíclica mais adequada à transmissão dos dados de reconfiguração de hardware, com o uso desta metodologia. A taxa de repetição cíclica influencia diretamente o processo de remontagem de seções, considerando-se que ela pode determinar o desempenho do sistema de remontagem. Dessa forma, 8 TSs, multiplexados com diferentes taxas de repetição cíclica das seções privadas, foram utilizados. Todos os exemplos foram baseados no Ex01.ts, mudando-se somente a taxa de repetição. A taxa de dados do fluxo de transporte foi de 1.571.374 bit/s para todos os exemplos. A Tabela 12 apresenta os valores médios alcançados com os experimentos realizados, em diferentes taxas. Na coluna $TS_{NAME,}$ o nome do TS é apresentado, na segunda coluna, a taxa de repetição cíclica, usada na multiplexação de seções privadas de cada TS, é informada e, na

última coluna, o tempo de remontagem (RMT) do arquivo de reconfiguração, no sistema de arquivos do receptor, é mostrado.

A partir dos resultados obtidos com os experimentos, é possível estabelecer o intervalo de taxa de repetição cíclica que mais se adapta ao esquema de remontagem do núcleo de hardware. Dessa forma, notou-se que a remontagem, para uma taxa de repetição de 500ms, teve o menor desempenho na captura de dados. Nessa taxa, há um maior descarte de seções, por parte do sistema de remontagem. Tal descarte ocorre quando o sistema consegue capturar um número menor de seções, durante cada ciclo de repetição. Embora, nessa taxa, seja possível enviar mais seções em um dado período de tempo, o *overhead* de leitura do esquema de remontagem provoca um menor rendimento. Isso faz com que mais ciclos, para a captura de todas as seções, sejam necessários.

Tabela 12 - Experimento usando diferentes taxas de repetição cíclica.

TS_{NAME}	Taxa de repetição cíclica das seções privadas	RMT
Ex01_500ms.ts	500 ms	29,20 s
Ex01_750ms.ts	750 ms	17,76 s
Ex01_1000ms.ts	1000 ms	9,87 s
Ex01_1250ms.ts	1250 ms	12,70 s
Ex01_1500ms.ts	1500 ms	13,60 s
Ex01_1750ms.ts	1750 ms	14,26 s
Ex01_2000ms.ts	2000 ms	14,84 s

Com a taxa de repetição configurada em 750ms, houve uma melhora significativa no desempenho da remontagem, acompanhada por um menor descarte de seções. No entanto, taxas no intervalo entre 1000 e 1500ms apresentaram o melhor desempenho médio, para o tempo de remontagem do núcleo de hardware.

As taxas acima desse intervalo apresentam valores próximos, mas já com uma tendência de aumento do tempo de remontagem. Para os demais experimentos, a taxa de repetição cíclica de seções privadas foi fixada em 1000ms e seguiu a mesma estratégia de projeto usada no Ex01.ts.

Os próximos experimentos têm por objetivo medir o desempenho do sistema, até o processo de reconfiguração do FPGA. Tais testes foram voltados para a validação de todas as etapas da metodologia, desde a preparação do fluxo de transporte até a completa recepção e reconfiguração do dispositivo alvo. Duas medidas de desempenho foram adotadas: o tempo de remontagem (*remounting time* – RMT), que consiste na soma entre tempo de cálculo de *checksum* e tempo de remontagem do arquivo SVF (núcleo de hardware), com a recepção de todas as seções privadas necessárias ao processo de reconfiguração, e o tempo de reconfiguração (*reconfiguration time* – RCT), que representa o tempo total para a reconfiguração do FPGA. Os

valores de RMT e RCT, apresentados nos experimentos, correspondem a valores médios resultantes de várias amostras. O RCT também considera o tempo de análise do arquivo remontado (*/. Svf) e a comunicação JTAG *host*/*target*, inicia no instante em que o sistema começa a fazer a análise do arquivo SVF, armazenado no sistema de arquivos do receptor, e termina com a reconfiguração total do FPGA.

O primeiro circuito (Ex01.svf), apresentado na Figura 32, foi utilizado para determinar a taxa de repetição cíclica das seções privadas. A Tabela 13 apresenta as características do experimento Ex01.ts, a uma taxa de 1000ms, com resultados de RMT e RCT.

Tabela 13 – Experimento com o núcleo de hardware - contador de 4 bits

TSNAME	**CORE**SINALIZAÇÃO-INFO		**TS**MULTIPLEXAÇÃO-INFO		**R**MT	**R**CT
Ex01.ts	TAMANHO	900.845 bytes	TAMANHO	70,806 MB	9.87s	29.12s
	FABRICANTE	ALTERA	TX. REPETIÇÃO (SEÇÃO)	1000 ms		
	FAMILIA	STRATIX	TX. DE DADOS DO TS	1.571.374 Bit/s		
	MODELO	EP1S10F780C6	VIDEO (TAMANHO)	10.063.944 bytes		
	MÓDULO	4-BITS BCD-CNT	VIDEO (TX. BITS)	1.429 Mbit/s		
			STREAM LOOP TIME	56.360 s		

O segundo circuito (Ex02.svf) foi desenvolvido para escrever caracteres em um mostrador de cristal líquido **(***liquid crystal display -- LCD***),** que está disponível na placa de desenvolvimento do FPGA [92]. Esse circuito utiliza as duas linhas do mostrador de 16 caracteres (16x2) e imprime mensagens em cada uma delas, além de trocar tais mensagens entre as linhas, utilizando uma temporização. Os resultados desse teste estão apresentados na Tabela 14.

Tabela 14 - Experimento com o núcleo de hardware – mensagem em LCD 16x2

TSNAME	**CORE**SINALIZAÇÃO-INFO		**TS**MULTIPLEXAÇÃO-INFO		**R**MT	**R**CT
Ex02.ts	TAMANHO	900.945 bytes	TAMANHO	70,819 MB	10.89s	29.44s
	FABRICANTE	ALTERA	TX. REPETIÇÃO (SEÇÃO)	1000 ms		
	FAMILIA	STRATIX	TX. DE DADOS DO TS	1.571.374 Bit/s		
	MODELO	EP1S10F780C6	VIDEO (TAMANHO)	10.063.944 bytes		
	MÓDULO	16x2 LCD MESSAGE	VIDEO (TX. BITS)	1.429 Mbit/s		
			STREAM LOOP TIME	56.360 s		

O terceiro circuito (*Ex03.svf*) é um conversor *binary code decimal* (BCD) de 4 bits. Esse circuito recebe um código binário de entrada, através das chaves de comutação disponibilizadas na placa do FPGA (Sw0, Sw1, Sw2, Sw3) [92], e escreve o resultado da conversão em um display de 7 segmentos também disponível na placa de desenvolvimento. Os resultados desse teste estão detalhados na Tabela 15.

Tabela 15 – Experimento com o núcleo de hardware – conversor BCD de 4 bits

TSNAME	CORESINALIZAÇÃO-INFO		TSMULTIPLEXAÇÃO-INFO		RMT	RCT
Ex03.ts	TAMANHO	900.923 bytes	TAMANHO	70,806 MB	9.95s	29.25s
	FABRICANTE	ALTERA	TX. REPETIÇÃO (SEÇÃO)	1000 ms		
	FAMILIA	STRATIX	TX. DE DADOS DO TS	1.571.374 Bit/s		
	MODELO	EP1S10F780C6	VIDEO (TAMANHO)	10.063.944 bytes		
	MÓDULO	BCD 7 SEG	VIDEO (TX. BITS)	1.429 Mbit/s		
			STREAM LOOP TIME	56.360 s		

O quarto circuito (*Ex04.svf*) é um contador para display de 7 segmentos. Esse circuito utiliza o sinal de clock, da placa de desenvolvimento, para realizar uma contagem decimal, através do display de 7 segmentos. Os resultados desse teste estão detalhados na Tabela 16.

Tabela 16 - Experimento com o núcleo de hardware – contador de 7 segmentos.

TSNAME	CORESINALIZAÇÃO-INFO		TSMULTIPLEXAÇÃO-INFO		RMT	RCT
Ex04.ts	TAMANHO	900.951 bytes	TAMANHO	70,818 MB	10.25s	29.26s
	FABRICANTE	ALTERA	TX. REPETIÇÃO (SEÇÃO)	1000 ms		
	FAMILIA	STRATIX	TX. DE DADOS DO TS	1.571.374 Bit/s		
	MODELO	EP1S10F780C6	VIDEO (TAMANHO)	10.063.944 bytes		
	MÓDULO	CNT 7 SEG	VIDEO (TX. BITS)	1.429 Mbit/s		
			STREAM LOOP TIME	56.360 s		

Alguns exemplos têm tamanhos similares, em bytes, e apresentaram resultados diferentes de RMT. Essas variações são dadas em virtude de fatores como condições do meio de transmissão e também do ponto de início da leitura das seções privadas (aleatório). A leitura das seções inicia assim que as informações da tabela UIT e a lista de PIDs e TIDs são extraídas, não sendo possível determinar o número da seção presente nesse instante. Por exemplo, em um total de 200 seções, a remontagem pode iniciar em qualquer seção no intervalo de 0 até 200. Se o ponto inicial de leitura for na seção 0, o experimento pode ser mais rápido, porém, se o início da leitura acontecer em uma seção intermediária, o tempo de remontagem pode sofrer variações. Essa aleatoriedade faz com que os resultados sejam diferentes entre os experimentos. Independentemente dessas variações, o núcleo de hardware será remontado, com a leitura de todas as seções. Os resultados de RCT podem ser otimizados com a integração do FPGA diretamente ao projeto da placa do receptor, o que otimizaria a comunicação entre sistema e FPGA. No entanto, essa atividade demandaria um novo leiaute.

5.2 Experimentos sinalizando hardware para outros receptores

Os experimentos utilizando núcleos de hardware sintetizados para diferentes fabricantes de FPGAs visam validar o comportamento do receptor, utilizando a metodologia de reconfiguração

de hardware. Desta forma, foram gerados fluxos de transporte com UITs sinalizando conteúdos para dispositivos de FPGA diferentes do utilizado. O sistema de teste deve ignorar conteúdos sinalizados para outros dispositivos de FPGA e, ao mesmo tempo reproduzir os fluxos elementares de áudio e vídeo multiplexados no TS. A Tabela 17 apresenta o caso de uso com a transmissão de núcleos de hardware para diferentes dispositivos. A coluna ES indica se os fluxos elementares foram reproduzidos com sucesso e a coluna IG indica informa se o conteúdo foi ignorado. O resultado é dado por (V), se o teste obteve sucesso, e (**N**), caso este falhe.

Tabela 17 - Caso de uso 1 (TstCase1.ts).

TS$_{NAME}$	**CORE$_{SINALIZAÇÃO-INFO}$**		**TS$_{MULTIPLEXAÇÃO-INFO}$**		**ES**	**IG**
TstCase1.ts	TAMANHO	900.945 bytes	TAMANHO	70,819 MB	V	V
	FABRICANTE	XILINX	TX. REPETIÇÃO (SEÇÃO)	1000 ms		
	FAMILIA	SPARTAN	TX. DE DADOS DO TS	1.571.374 Bit/s		
	MODELO	XC3S500E	VIDEO (TAMANHO)	10.063.944 bytes		
	MÓDULO	16x2 LCD MSG	VIDEO (TX. BITS)	1.429 Mbit/s		

Os resultados obtidos com o TstCase1.ts, apresentados na Tabela 17, indicaram que a atualização de hardware, enviada a um modelo de FPGA XC3S500E do fabricante Xilinx, da família Spartan e com *part number* (PN), não foi aceito após a extração do conteúdo da tabela UIT. Nesse mesmo processo, é possível constatar que o fluxo elementar de vídeo foi reproduzido, como esperado. Essa validação demonstra a capacidade da metodologia em selecionar somente o conteúdo enviado para a unidade receptora e que núcleos de diferentes fabricantes podem ser transmitidos, o que possibilita reconfigurar somente as unidades receptoras que são compatíveis com o conteúdo do sinal.

Assim como o experimento anterior, os resultados obtidos com o TstCase2.ts, mostrados na Tabela 18, indicaram que a atualização de hardware enviada a um modelo de FPGA Cyclone-IV PN. EP4CGX110, do fabricante altera, também não foi utilizado para o processo de reconfiguração do FPGA instalado no receptor. Além disso, o fluxo elementar de vídeo foi reproduzido, como esperado.

Tabela 18 - Caso de uso 2 (TstCase2.ts).

TS$_{NAME}$	**CORE$_{SINALIZAÇÃO-INFO}$**		**TS$_{MULTIPLEXAÇÃO-INFO}$**		**ES**	**IG**
TstCase2.ts	TAMANHO	900.923 bytes	TAMANHO	70,817 MB	V	V
	FABRICANTE	ALTERA	TX. REPETIÇÃO (SEÇÃO)	1000 ms		
	FAMILIA	CYCLONE-IV	TX. DE DADOS DO TS	1.571.374 Bit/s		
	MODELO	EP4CGX110	VIDEO (TAMANHO)	10.063.944 bytes		
	MÓDULO	BCD 7 SEG	VIDEO (TX. BITS)	1.429 Mbit/s		

Assim como o experimento anterior, os resultados obtidos com o TstCase3.ts, resumidos na Tabela 19, também indicaram que a atualização de hardware enviada a um modelo de FPGA Smartfusion do fabricante Actel, PN. A2F200M3F, não foi utilizada e o fluxo elementar de vídeo também foi reproduzido, como esperado.

Tabela 19 - Caso de uso 3 (TstCase3.ts).

TSNAME	**COREsinalização-info**		**TSmultiplexação-info**		**ES**	**IG**
TstCase3.ts	TAMANHO	900.845 bytes	TAMANHO	70,806 MB	V	V
	FABRICANTE	ACTEL	TX. REPETIÇÃO (SEÇÃO)	1000 ms		
	FAMILIA	SMARTFUSION	TX. DE DADOS DO TS	1.571.374 Bit/s		
	MODELO	A2F200M3F	VIDEO (TAMANHO)	10.063.944 bytes		
	MÓDULO	4-BITS BCD-CNT	VIDEO (TX. BITS)	1.429 Mbit/s		

Os experimentos com modelos de FPGA diferentes dos utilizados apresentaram o resultado esperado, com a reprodução da mídia associada a cada fluxo elementar e a rejeição do conteúdo sinalizado para um FPGA diferente do utilizado no receptor.

6. Considerações Finais

A metodologia apresentada neste trabalho demonstrou a possibilidade de transmissão de hardware pré-sintetizado através de um canal aberto de transmissão de televisão digital. Os experimentos realizados com a transmissão e recepção de circuitos pré-sintetizados, em um fluxo de transporte, são a prova conceitual necessária à metodologia proposta. O trabalho também demonstrou que existe viabilidade técnica para o desenvolvimento de receptores com base nesta metodologia. Dessa forma, vale salientar que a metodologia pode ser utilizada por receptores de TV Digital que sejam voltados para a atualização de hardware em campo. Em um mercado horizontal, como é o caso dos fabricantes de componentes eletrônicos, esta atualização pode não ser bem vista, no entanto, em um mercado vertical, como o de provedores de TV por assinatura, esta possibilidade de atualização do hardware seria extremamente interessante.

O processo de sinalização de núcleos de hardware, através da tabela UIT, provê ferramentas para que receptores selecionem somente núcleos de hardware enviados à sua própria unidade, o que permite que indústrias desse nicho de mercado utilizem dispositivos de FPGA de qualquer fabricante. Os experimentos realizados, com a sinalização de diferentes núcleos de hardware, demonstraram a eficiência da metodologia proposta, durante a seleção de núcleos de hardware válidos para a unidade receptora. A utilização do método de transmissão de dados "*data streaming* com seções privadas", para a carga do núcleo de hardware, possibilitou eficiência durante a recuperação dos dados na unidade receptora. Além disso, a distribuição do núcleo de hardware em grupos de seções, sinalizados pelos seus respectivos PIDs e TIDs na tabela UIT, amplia o tamanho de carga máximo permitida com o método de transmissão de dados via seções privadas.

Os resultados obtidos mostraram que a taxa de repetição cíclica escolhida influencia diretamente o tempo de remontagem dos núcleos de hardware. Taxas de repetição menores podem gerar maior descarte de seções e, com isso, aumentar o tempo de remontagem. Foi estabelecido um intervalo empírico, onde taxas de repetição cíclicas obtém um melhor rendimento, com relação ao processo de remontagem de seções. O formato de arquivo SVF adiciona a sintaxe necessária à realização da reconfiguração de hardware, através da tecnologia BS/JTAG, o que faz com que o núcleo de hardware praticamente dobre de tamanho. O processo de remontagem pode ser mais rápido, caso núcleos de hardware, em formatos proprietários, sejam utilizados durante o processo de reconfiguração do FPGA.

7. Trabalhos Futuros

A reconfiguração de hardware, utilizando o sinal de TV Digital, tem um aspecto inovador e pode servir de base para projetos de sistemas embarcados com atualização de hardware. É possível imaginar uma série de arquiteturas e formas de atualização de hardware, que antes seriam inviáveis, considerando-se as dificuldades de se atualizar dispositivos em campo. Certamente, essas novas arquiteturas deverão ser planejadas, de acordo com o contexto de cada tipo de aplicação. Entre os trabalhos que poderiam ser desenvolvidos, estão:

- Atualização do módulo de decodificação de vídeo H.264, o que permitiria que receptores de TVD incorporassem avanços tecnológicos nas normas, como é o caso do decodificador H.265;
- Uso e atualização de módulos baseados em criptografia em hardware, com o objetivo de aumentar a segurança do canal de TV Digital. Nesse caso, o canal de TV digital poderia ser utilizado para transações seguras, com uma confiabilidade maior que a outros meios de comunicação;
- Desenvolvimento de receptores universais adaptativos ao contexto do demodulador, que reconfigurariam o módulo de demodulação para a filtragem de qualquer provedor de TV a cabo.

Referências

[1] GARCIA, Philip, *et al*. An overview of reconfigurable hardware in embedded systems. **EURASIP Journal on Embedded Systems**, v.1, p.13-13, 2006.

[2] CHALLA, Sanjay. How FPGAs, Multicore CPUs are changing embedded design. **EDN Network**, 10 April 2013. Disponível em http://www.edn.com/design/systems-design/4411864/How-FPGAs-and-multicore-CPUs-are-changing-embedded- design Acesso em: 8 dez. 2014.

[3] PATEL, Parimal. Embedded systems design using FPGA. *In*: **19th International Conference on VLSI DESIGN, 2006. Held jointly with 5th International Conference on Embedded Systems and Design**, Hyderabad: IEEE, 2006.

[4] FIGULI, Peter *et al*. A heterogeneous SoC architecture with embedded virtual FPGA cores and runtime Core Fusion. *In*: **NASA/ESA Conference on Adaptive Hardware and Systems (AHS)**, IEEE, 2011. p. 96-103.

[5] SIOZIOS, K.; SOUDRIS, D.; HUBNER, M. A framework for customizing virtual 3-D reconfigurable platforms at run-time parallel & distributed processing. *In*: **Symposium Workshops & PhD Forum (IPDPSW)**, IEEE, 2014. p. 183-188.

[6] BOMEL, Pierre; MARTIN, Kevin; DIGUET, J.-P. Virtual UARTs for reconfigurable multi-processor architectures parallel and distributed processing. *In*: **Symposium Workshops & PhD Forum (IPDPSW) 27th International**, IEEE, 2013. p. 252-259.

[7] SIDIROPOULOS, Harry *et al*. A platform-independent runtime methodology for mapping multiple applications onto FPGAs through resource virtualization. *In*: **23rd International Conference on Field Programmable Logic and Applications (FPL)**, IEEE, 2013. p. 1-4.

[8] KARUNAKAR REDDY, Basireddy; SABBAVARAPU, Srinivas; ACHARYYA, Amit. A new VLSI IC design automation methodology with reduced NRE costs and time-to-market using the NPN class representation and functional symmetry. *In*: **International Symposium on Circuits and Systems (ISCAS)**, IEEE, Melbourne, Australia, 2014. p. 177-180.

[9] REDDY, Basireddy Karunakar *et al*. A novel and unified digital ic design and automation methodology with reduced NRE cost and time-to-market. *In*: **International Symposium on Electronic System Design (ISED)**, IEEE, Singapore, 2013. p. 36-40.

[10] LEONG, Philip Heng Wai. Recent trends in FPGA architectures and applications. *In*: **4th IEEE International Symposium on Electronic Design, Test and Applications (DELTA)**, IEEE, 2008. p. 137-141.

[11] MAN-HO HO *et al*. Architecture and design flow for a highly efficient structured ASIC. **IEEE Transaction on Very Large Scale Integration (VLSI) Systems**, v. 21, p. 424-433, 2013.

[12] ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR 15604: **Televisão digital terrestre - Receptores**, 2007.

[13] IEEE DESIGN AUTOMATION STANDARDS COMMITTEE *et al*. STD 1076-2008, IEEE Standard VHDL Language Reference Manual. **IEEE,** New York, USA, 2008.

[14] ASHENDEN, Peter J. The Designer's Guide to VHDL. **Morgan Kaufmann Publishers Inc**., San Francisco, 1996.

[15] SYNARIO DESIGN AUTOMATION. **VHDL Reference Manual**. Redmond, Washington, 1997. 144p.

[16] IEEE COMPUTER SOCIETY. **IEEE Standard for Verilog Hardware Description Language, IEEE STD 1364-2005**. 2006.

[17] PALNITKAR, Samir. **Verilog HDL: a guide to digital design and synthesis**. Prentice Hall Professional, 2003.

[18] OPEN VERILOG INTERNATIONAL. **Verilog-A Language Reference Manual Analog Extensions to Verilog HDL**. USA v. 1.0, 1996. 170p.

[19] YOSHIDA, Daisuke et al. Highly efficient H. 264/AVC codec technology for high definition consumer applications. *In*: **International Conference on Consumer Electronics (ICCE). Digest of Technical Papers**, IEEE, 2008. p. 1-2.

[20] WIEGAND, Thomas et al. Overview of the H. 264/AVC video coding standard. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 13, n. 7, p. 560-576, 2003.

[21] INTERNATIONAL TELECOMMUNICATION UNION. **Recommendation H.264. Advanced Video Coding for Generic Audiovisual Services**. 2012.

[22] ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR 15601: **Televisão digital terrestre – Sistema de transmissão**. 1. ed. 2007.

[23] SULLIVAN, Gary J. et al. Overview of the high efficiency video coding (HEVC) standard. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 22, n. 12, p. 1649-1668, 2012.

[24] BOSSEN, Frank et al. HEVC complexity and implementation analysis. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 22, n. 12, p. 1685-1696, 2012.

[25] KOUMARAS, Harilaos; KOURTIS, M.; MARTAKOS, Drakoulis. Benchmarking the encoding efficiency of H.265/HEVC e H.264/AVC. In: **Future Network & Mobile Summit, 2012**. IEEE, 2012. p. 1-7.

[26] VANGUARD VIDEO. **Vanguard Video Demonstrates H.265/HEVC Technology Running on Xilinx FPGA Hardware**, LAS VEGAS, 2013.

[27] ETSI, E. N. Digital Video Broadcasting (DVB); specification for service information (SI) in DVB systems. **European Standard**, v. 1, 2010.

[28] ARIB STANDARD. Transmission system for digital terrestrial television broadcasting. **Association of Radio Industries and Businesses**, 2009.

[29] LIN, Chih-Tung; HORNG, Shi-Jinn; HUANG, Yao-Lin. Hardware resource manager for reconfiguration system. *In*: **International Symposium on Biometrics and Security Technologies (ISBAST)**, IEEE, 2012. p. 59-65.

[30] ECHANOBE, Javier et al. Dynamic partial reconfiguration in embedded systems for intelligent environments. *In*: **8th International Conference on Intelligent Environments (IE)**, IEEE, 2012. p. 109-113.

[31] HILLENBRAND, Dominic et al. RIVER: Reconfigurable pre-synthesized-streaming architecture for signal processing on FPGAS. In: **Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International**, IEEE, 2012. p. 397-400.

[32] SERRES, Olivier; NARAYANA, Vikram K.; EL-GHAZAWI, Tarek. An architecture for reconfigurable multi-core explorations. *In*: **International Conference on Reconfigurable Computing and FPGAs**, IEEE, 2011. p. 105-110.

[33] ATSC. ATSC Digital Television Standard: Part 1 – Digital Television System. **Advanced Television Systems Committee, Inc.,** 2009.

[34] ETSI, E. N. Digital video broadcasting (DVB); framing structure, channel coding and modulation for digital terrestrial television. **European Standard**, v. 1, 2009.

[35] CASSIA, Fernando. Japanese-Brazilian digital TV standard conquers South America. **Tech Eye Net**, 2010. Disponível em: <http://news.techeye.net/hardware/japanese-brazilian-digital-tv-standard-conquers-south-america>. Acesso em: 05 dez. 2014.

[36] INTERNATIONAL STANDARD. ISO/IEC 14496: **Information technology – Coding of audio-visual objects – Part 3: Audio**. 2005.

[37] INTERNATIONAL STANDARD. ISO/IEC 14496: **Information technology – Coding of audio-visual objects – Part 3: Advanced Video Coding**. 2005.

[38] INTERNATIONAL STANDARD. ISO/IEC 13818-1: **Information Technology – Generic Coding of Moving Pictures and Associated Audio Information - Part 1: Systems.** Geneva, Switzerland, 3th ed. 2007.

[39] ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR 15602-2: **Codificação de vídeo, áudio e multiplexação. Parte 2: Codificação de áudio**. 1. ed. 2007.

[40] ARIB STANDARD. Video coding, audio coding, and multiplexing specifications for digital broadcasting. In: ARIB STD-B32. Part 1: Video Signals and Coding Scheme. **Association of Radio Industries and Businesses**, Version 2.1-E1, 2007.

[41] ARIB STANDARD. Video coding, audio coding, and multiplexing specifications for digital broadcasting. In: ARIB STD-B32. Part 2: Audio Signals and Coding Scheme. **Association of Radio Industries and Businesses**, Version 2.1-E1, 2007.

[42] ARIB STANDARD. Video coding, audio coding, and multiplexing specifications for digital broadcasting. In: ARIB STD-B32. Part 3: Multiplexing Scheme of Transmission Signals. **Association of Radio Industries and Businesses**, Version 2.1-E1, 2007.

[43] ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR 15602-3: **Codificação de**

vídeo, áudio e multiplexação. Parte 3: Sistemas de multiplexação de sinais. 1. ed. 2007.

[44] ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 15603-2: Multiplexação e serviços de informação (SI). Parte 2: Estrutura de dados e definições da informação básica de SI.** 1. ed. 2007.

[45] ETSI, E. N. Digital video broadcasting (DVB); DVB specification for data broadcasting. **European Standard**, v. 1, p. 1, 2000.

[46] REIMERS, Ulrich (Ed.). **DVB: the family of international standards for digital video broadcasting**. Springer Science & Business Media, 2005.

[47] NATARAJAN, Balajee. Study of transmission efficiency of IP packets over ATM using LLC/SNAP encapsulation. *In*: **The 9th Asia-Pacific Conference on Communications (APCC)**. IEEE, 2003. p. 892-895.

[48] ETSI, E. N. Digital Video Broadcasting (DVB); Implementation guidelines for Data Broadcasting. **European Standard**, v. 1, 2003.

[49] FURHT, Borko; AHSON, Syed A. (Ed.). **Handbook of mobile broadcasting: DVB-H, DMB, ISDB-T, and mediaflo**. CRC Press, 2008.

[50] INTERNATION TELECOMUNICATION UNION, Broadcasting of Multimedia and Data Applications for Mobile Reception by Handheld Receivers, Recomendation BT.1833, ITU-R, 2014.

[51] PICCIONI, Carlos Alexandre *et al*. **Modelo e implementação de um serviço de datacasting para televisão digital**. 2005.

[52] CASTAGNOLI, Guy; BRAUER, Stefan; HERRMANN, Martin. Optimization of cyclic redundancy-check codes with 24 and 32 parity bits. **IEEE Transactions on Communications.** v. 41, n. 6, p. 883-892, 1993.

[53] BAICHEVA, Tsonka S. Determination of the best CRC codes with up to 10-bit redundancy. **IEEE Transactions on Communications**, v. 56, n. 8, 2008, p. 1214-1220.

[54] KOOPMAN, Philip. 32-bit cyclic redundancy codes for internet applications. *In*: **International Conference on Dependable Systems and Networks, 2002. DSN 2002. Proceedings**. IEEE, 2002. p. 459-468.

[55] WICKER, Stephen B. **Error control systems for digital communication and storage**. Englewood Cliffs: Prentice hall, 1995.

[56] DA SILVA MELLO, L. A. R. et al. Measurements of propagation loss and multipath delay profiles at 593 MHz in a dense urban area in Brazil. *In*: **The Second European Conference on Antennas and Propagation (EUCAP)**. IET, 2007. p. 1-3.

[57] BEDICKS, Gunnar et al. Results of the ISDB-T system tests, as part of digital TV study carried out in Brazil. **IEEE Transactions on Broadcasting**. v. 52, n. 1, p. 38-44, 2006.

[58] ANGUEIRA, Pablo *et al*. DTV (COFDM) SFN signal variation field tests in urban environments for portable outdoor reception. **IEEE transactions on broadcasting**, v. 49, n. 1, p. 81-86, 2003.

[59] HENDRANTORO, Gamantyo et al. Urban radio propagation measurement for digital TV broadcast in Jakarta, Indonesia. **IEEE Broadcast Technology Newsletter**, 2007.

[60] NXP. Create affordable IP and hybrid DTV set-top boxes with highest HD picture quality, NXP Nexperia HD IP STB development kit STB225, DSA00339603.pdf. Denver, USA, 2007. Disponível em: <www.nxp.com>. Acesso em: 17 dez. 2014.

[61] NXP. **Move to a higher definition with TDA998x HDMI transmitters, NXP HDMI transmitters TDA9981, TDA9983 and TDA9984**. Netherlands, 2007. Disponível em: <www.nxp.com>. Acesso em: 17 dez. 2014.

[62] BORE, Chris. **Direct FB: Linux API for video mixing**. Bores Signal Processing, 1th ed. 2014, 390p.

[63] SOARES, Luiz Fernando Gomes; RODRIGUES, Rogério Ferreira; MORENO, Márcio Ferreira. Ginga-NCL: the declarative environment of the Brazilian digital TV system. **Journal of the Brazilian Computer Society**, v. 12, n. 4, p. 37-46, 2007.

[64] SOUZA FILHO, Guido Lemos de; LEITE, Luiz Eduardo Cunha; BATISTA, Carlos Eduardo Coelho Freire. Ginga-j: The procedural middleware for the brazilian digital TV system. **Journal of the Brazilian Computer Society**, v. 12, n. 4, p. 47-56, 2007.

[65] PIESING, Jon. The DVB multimedia home platform (MHP) and related specifications. **Proceedings of the IEEE**, v. 94, n. 1, p. 237-247, 2006.

[66] KUON, Ian; ROSE, Jonathan. Measuring the gap between FPGAs and ASICs. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,** v. 26, n. 2, p. 203-215, 2007.

[67] WESTE, Neil HE; ESHRAGHIAN, Kamran. Principles of CMOS VLSI design: a systems perspective. **NASA STI/Recon Technical Report A**, v. 85, p. 47028, 1985.

[68] ABRAHAM, Carlos; CHIAO, Sun. The FPGA to MPGA ASIC conversion process. In: **Proceedings of the 40th Midwest Symposium on Circuits and Systems**. IEEE, 1997. p. 1426-1429.

[69] GROUT, Ian. **Digital systems design with FPGAs and CPLDs**. Newnes, Burlington, USA, 2011. 784p.

[70] BROWN, Stephen; ROSE, Jonathan. FPGA and CPLD architectures: A tutorial. **IEEE design & test of computers**, v. 13, n. 2, p. 42-57, 1996.

[71] HAUCK, Scott. The roles of FPGAs in reprogrammable systems. **Proceedings of the IEEE**, v. 86, n. 4, p. 615-638, 1998.

[72] MONMASSON, Eric; CIRSTEA, Marcian N. FPGA design methodology for industrial control systems - A review. **IEEE Transactions on Industrial Electronics,** v. 54, n. 4, p. 1824-1842, 2007.

[73] KUON, Ian; TESSIER, Russell; ROSE, Jonathan. FPGA architecture: Survey and challenges. **Foundations and Trends in Electronic Design Automation**, v. 2, n. 2, p. 135-253, 2008.

[74] AHMED, Syed Zahid et al. Survey of new trends in industry for programmable hardware: FPGAs, MPPAs, MPSoCs, structured ASICs, eFPGAs and new wave of innovation in FPGAs. *In*: **International Conference on Field Programmable Logic and Applications (FPL)**, IEEE, 2010. p. 291-297.

[75] ELES, Petru; KUCHCINSKI, Krzysztof; PENG, Zebo. **System Synthesis with VHDL: a transformational approach**. Kluwer Academic Publishers, Norwell, MA, 1998.

[76] CHANG, Kou-Chuan. **Digital design and modeling with VHDL and synthesis**. IEEE Computer Society Press, Los Alamitos, CA, 1996.

[77] CHU, Pong P. **RTL hardware design using VHDL: coding for efficiency, portability, and scalability**. John Wiley & Sons, 2006.

[78] INTERTECH, ASSET. Inc., **Serial Vector Format Specification**. 1999.

[79] PARKER, Kenneth P. (Ed.). **The Boundary - Scan Handbook Third Edition**. Springer Science & Business Media, 2003.

[80] IEEE STANDARD. IEEE Standard for Test Access Port and Boundary-Scan Architecture. **IEEE Std**., n. 1149.1, 2013.

[81] ALTERA. NIOS Development Board, Stratix Edition Data Sheet. **Altera Corporation**, 2003.

[82] STRUBEL, Martin. I**mplementing JTAG debugging solutions for custom hardware**. 2012.

[83] DOMINIC, R. **Design and Implementation of an On-Chip Debug Solution for Embedded Target Systems based on the ARM7 and ARM9 Family**. Ph.D. thesis, University of Applied Sciences Augsburg, Germany, 2005.

[84] BRIDGFORD B.; CAMMON J. **SVF and XSVF File Formats for Xilinx, Xilinx Corporation, application not: Xilinx Devices**. 2007. Disponível em: <http://www.pldtool.com/pdf/xapp503_svf_xsvf.pdf>. Acesso em: 10 nov. 2014.

[85] ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 15606-3: Digital terrestrial television - Data coding and transmission specification for digital broadcasting - Part 3: Data transmission specification.** 1. ed. 2007.

[86] **URJTAG, Universal JTAG Library, server and tools, SourceForge**. Disponível em: <http://urjtag.org/>. Acesso em: 10 jan. 2014.

[87] **OPEN JTAG Project**. 2014. Disponível em: <http://openjtag.org>. Acesso em: 05 jan. 2014.

[88] ZOU, Hong; HUANG, Jiye; GAO, Mingyu. The application of virtual JTAG technology in FPGA design and debugging. In: **Electrical and Control Engineering (ICECE), 2011 International Conference on**. IEEE, 2011. p. 2637-2640.

[89] BAANG, Shen Xu; HAI, Liang Song. Design and implementation of a JTAG boundary-scan interface controller. *In*: **Proceedings of the Second Asian Test Symposium**, IEEE, 1993. p. 215-218.

[90] ROBINSON, Gordon D. Why 1149.1 (JTAG) really works. *In*: **Proceedings of the Electro/94 International. Combined Volumes**, IEEE, 1994. p. 749-754.

[91] FTDI CHIP. **FT232R - USB UART IC**. Disponível em: <www.ftdichip.com>. Acesso em: 17 jan. 2014.

[92] ALTERA. NIOS Hardware development tutorial for the NIOS development board, Stratix Edition Data Sheet. **Altera Corporation**, 2003.

[93] DEKTEC. **Quality Tools for Digital – TV Professionals DTU-215 USB-2 VHF/UHF Modulator**. Disponível em:<http://www.dektec.com/products/USB/ DTU-215/>. Acesso em: 17 dez. 2014.

ANEXO I – definições da norma ISO/IEC 13818-1

Sintaxe dos dados da tabela	Taxa de bits (No. de bits)	Mnemônico
private_section() {		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
private_indicator	1	bslbf
reserved	2	bslbf
private_section_length	12	uimsbf
if (section_syntax_indicator == '0') {		
for (i=0; i < N; i++) {		
private_data_byte	8	bslbf
}		
}		
else {		
table_id_extension	16	uimsbf
reserved	2	bslbf
version_number	5	uimsbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
for (i=0; i < private_section_length-9; i++) {		
private_data_byte	8	bslbf
}		
CRC_32	32	rpchof
}		
}		

ANEXO II – Especificação SVF

A lista de comandos, bem como a descrição de cada comando permitido na sintaxe da especificação SVF são listados na tabela abaixo.

Comando	DESCRIÇÃO
ENDDR	Especifica estado END como *default* para as operações de varredura de DR.
ENDIR	Especifica estado END como *default* para as operações de varredura de IR.
FREQUENCY	Especifica a frequência de teste máxima para as operações de barramento (*bus*) IEEE 1149.1.
HDR	*Header Data Register*, especifica o padrão do cabeçalho que precederá o início das próximas operações de varredura do registrador DR.
HIR	*Header Instruction Register*, especifica o padrão do cabeçalho que precederá o início das próximas operações de varredura do registrador IR.
PIO	*Parallel Input/Output*, especifica o padrão de teste paralelo.
PIOMAP	*Parallel Input/Output Map*, mapeia as posições das colunas dos pinos de entrada e saída PIO para um pino lógico.
RUNTEST	Força as operações de barramento (*bus*) IEEE 1149.1 a executar um estado por um número específico de *clocks* ou por período específico de tempo.
SDR	*Scan Data Register*, executa uma varredura no registrador de dados (DR) IEEE 1149.1.
SIR	*Scan Instruction Register*, executa uma varredura no registrador de instrução (IR) IEEE 1149.1.
STATE	Força o barramento (*bus*) IEEE 1149.1 para um estado específico.
TDR	*Trailer Data Register*, especifica o padrão de transporte (*trailer*) que é anexado ao final da próxima operação de varredura (*scan*) do registrador de dados (DR).
TIR	*Trailer Instruction Register*, especifica o padrão de transporte (*trailer*) que é anexado ao final da próxima operação de varredura (*scan*) do registrador de instrução (IR).
TRST	*Test ReSeT*, sinal opcional que controla a reinicialização (*reset*) do teste.

ANEXO III - IEEE 1149.1/SVF

A tabela abaixo ilustra o nome dos estados lógicos da máquina de estados TAP no padrão IEEE 1149.1 e a correspondência no formato SVF.

Estados TAP def. IEEE 1149.1	Estados TAP formato SVF	Estados TAP def. IEEE 1149.1	Estados TAP formato SVF
Test-Logic-Reset	RESET	Update-DR	DRUPDATE
Run-Test/Idle	IDLE	Select-IR-Scan	IRSELECT
Select-DR-Scan	DRSELECT	Capture-IR	IRCAPTURE
Capture-DR	DRCAPTURE	Shift-IR	IRSHIFT
Shift-DR	DRSHIFT	Exit1-IR	IREXIT1
Exit1-DR	DREXIT1	Pause-IR	IRPAUSE
Pause-DR	DRPAUSE	Exit2-IR	IREXIT2
Exit2-DR	DREXIT2	Update-IR	IRUPDATE

ANEXO IV – Sintaxe do parser de tabelas

No trecho de código apresentado na tabela abaixo o parâmetro “buf” do método *parsePAT* (linha 1) contém uma “fatia” em bytes do *transport stream*. O primeiro campo *table_id* é o identificador da tabela PAT com valor fixo em 0x00. A sintaxe do algoritmo extrai todos os valores dos campos da tabela PAT. Na iteração da linha 29 os PIDs da tabela PMT listados na tabela PAT são extraídos e mantidos em uma estrutura de dados.

```
1: void Tables::parsePAT(const uint8 t  *buf)
2: {
3:   pat.table_id                  = (uint8_t)buf[0];
4:   pat.section_syntax_indicator  = (uint8_t)(buf[1] >> 7 & 0x01);
5:   pat.zero                      = (uint8 t)(buf[1] >> 6 & 0x01);
6:   pat.reserved 0                = (uint8 t)(buf[1] >> 4 & 0x03);
7:   pat.section length            = (int32 t)((((uint32 t)(buf[1]&0x0f)<<8)|(uint32 t)buf[2]));
8:   pat.transport_stream_id       = (int32_t)(((uint32_t )b uf[3]<<8)|(uint32_t)buf[4]);
9:   pat.reserved_1                = (uint8_t)(buf[5] >> 6 & 0x03);
10:  pat.version_number            = (uint8_t)(buf[5] >> 1 & 0x1f);
11:  pat.current next indicator    = (uint8 t)(buf[5] & 0x01);
12:  pat.section number            = (uint8 t)buf[6];
13:  pat.last section number       = (uint8 t)buf[7];
14:
15:  section_length = (int32_t)(pat.section_length - PAT_FIXED_FIELDS);
16:  idx = 0;
17:  nit idx = 0;
18:
19:  while (section_length > 0)
20:  {
21:    pat.numProgram[idx].program_number = (int32_t)(((uint32_t)buf[8]<<8)|(uint32_t )buf[9]);
22:
23:    if(pat.numProgram[idx].program number == 0)
24:    {
25:       pat.nit_pid = (int32_t)(((uint32_t )(buf[10]&0x1f)<<8)|(uint32_t)buf[11]);
26:    }
27:    else
28:    {
29:       pat.numProgram[idx].pmt pid=(int32 t)(((uint32 t)(buf[10]&0x1f)<<8)|(uint32 t)buf[11]);
30:       idx++;
31:    }
32:    pat.numProgram[idx].reserved_1 = (uint8_t)((buf[10]>>5)&0x07);
33:    buf += 4;
34:    section length -= 4;
35:  }
36:
37:  pat.crc_32 = (int32_t)((buf[0]<<24)|(buf[1]<<16)|(buf[2]<<8)|(uint32_t)buf[3]);
38:  pat.num programs = idx;
39: }
```

ANEXO V – Sintaxe da tabela UIT

FPGA_CORE_SIZE	Campo numérico de 32-bit que contém o tamanho em bytes do núcleo transmitido no sinal. Exemplo: 900845 bytes.
FPGA_CORE_VERSION	Campo numérico de 16-bit contendo o código de atualização do núcleo que está sendo transmitido. Com este código o receptor pode gerenciar a atualização de *hardware* do sistema do dispositivo. Exemplo: 0x12
FPGA_CORE_MODULE_NAME	Campo consistindo de uma sequencia de caracteres de 16 bytes contendo o nome do módulo de *hardware* transmitido. Exemplo: "16X2 LCD MESSAGE"
FPGA_CORE_DEVICE_INFO	Campo consistindo de uma sequência de caracteres de 27 bytes contendo as informações do dispositivo alvo para qual o núcleo de *hardware* foi sintetizado. Exemplo: "ALTERA STRATIX EP1S10F780C6"
REMOUNT_CORE_PID	Campo numérico de 32-bit que contém o *Packet Identifier* (PID) endereçando os pacotes no fluxo de transporte que transportam o núcleo de *hardware*. Exemplo: 0x0016.
REMOUNT_CORE_SECTION_TID	Campo numérico de 16-bit com o valor do identificador de seção *table_id* (TID) que endereça as seções privadas que estão transportando o conteúdo de *hardware*. Exemplo: 0x80
REMOUNT _PRIORITY	Campo de 8-bit que representa a ordem de prioridade da remontagem. Este campo vai ser necessário quando um core ultrapassar o limite permitido em grupos de seção que é de 256*4080 = ~1Mb. Neste caso este valor vai ser utilizado para priorizar a ordem de remontagem de cada grupo com mesmo PID e TID iniciando em 0x00 e assim sucessivamente.

ANEXO VI – Publicações

Artigos Publicados

1. OLIVEIRA, R. R. ; CORDEIRO, L. C. ; Lucena Jr, V. F. "**Hardware Reconfiguration Based on Broadcasted Digital TV Signal**," In: IEEE International Conference on Consumer Electronics (ICCE), 2015, Las Vegas. 2015 IEEE International Conference on Consumer Electronics. Las Vegas, USA: IEEE, 2015. v. ICCE15. p. 624-626.

2. OLIVEIRA, R. R., FILHO, E. B., LUCENA Jr, V. F. "**Reconfigurable Systems for Digital TV Environment,**" In: *10th European Interactive TV Conference, Adjunct Proceedings of the 10th European Interactive TV Conference - EUROITV*, v. 1. Berlin, Germany, 2012. p. 233-236.

3. OLIVEIRA, R. R.; FILHO, E. B.; Lucena Jr, V. F.; "**A Framework for Hardware Reconfiguration Using the Digital TV Signal**,**"** In: *8th FPGAWorld Conference. New York: ACM v. 1. p. 1-5. doi>10.1145/2157871.2157877*, Munich, Germany, 2011.

4. OLIVEIRA, R. R., FILHO, E. B., Lucena Jr, V. F., "**Metodologia de Reconfiguração de Hardware utilizando o Sinal de TV Digital**, ". In: *VI CITA, Congresso Ibero Americano de Telemática.* Gramado, Rio Grande do Sul. Cadernos de Informática UFRGS, Porto Alegre, RS. v. 6. 2011. p. 9-14.

Artigos Submetidos

1. OLIVEIRA, R. R., CORDEIRO L. C., FILHO, E. B., LUCENA Jr, V. F., "**Hardware Reconfiguration Scheme Based on the Digital TV Signal** ," In: *IEEE Transactions on Circuits and Systems for Video Technology.*