UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

FABRÍCIO DA SILVA CAETANO

Processador Gráfico

Trabalho de Diplomação

Prof. Dr. Altamiro A. Susin
Orientador

Porto Alegre, dezembro de 2010

AGRADECIMENTOS

Agradeço ao Prof. Dr. Altamiro A. Susin a oportunidade de trabalhar no LaPSI, ao lado de pessoas competentes, cujas necessidades dentro do projeto pude suprir com o andamento deste trabalho. Em virtude disto, pude me aprofundar mais no entendimento do Sistema Brasileiro de TV Digital.

Agradeço à existência de um grupo de usuários LaTeX na UFRGS (UTUG, 2010), o qual forneceu o modelo deste documento.

Agradeço à minha família pela compreensão sobre o fato de não estar tão presente quando gostaria.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS 6

LISTA DE FIGURAS 7

LISTA DE TABELAS 8

RESUMO 9

ABSTRACT 10

1 INTRODUÇÃO 11
1.1 Trabalhos Relacionados 12
1.1.1 Microcontrollers for Closed Caption System 12
1.1.2 Video Decoder with Closed Caption Data on Video Output 13
1.1.3 System and Method for Displaying Closed Caption Data on a PC 13
1.1.4 Method and Apparatus for Providing Real Time Data on a Viewing Screen Concurrently with any Programing in Process 14
1.1.5 Computer System and Closed Caption Display Method 15
1.2 Legendas 15
1.3 Objetivos 15

2 METODOLOGIA 17
2.1 Formato dos Dados de Entrada 17
2.2 Ambiente de Desenvolvimento do Protótipo em Software 17
2.3 Protótipo Desenvolvido 18
2.4 Geração de vetores de testes 18
2.5 Projeto da Versão em Hardware 19
2.6 Verificação do Funcionamento em Tempo Real 19

3 DESENVOLVIMENTO DO PROTÓTIPO EM SOFTWARE 20
3.1 O Programa de Referência PRH264 20
3.2 Demultiplexador 20
3.3 Processador Gráfico 21

4 DESENVOLVIMENTO DOS COMPONENTES DE HARDWARE 26
4.1 Demultiplexador 26
4.2 Processador Gráfico 28
4.2.1 Filtro de *data_groups* 29
4.2.2 Filtro de *caption_data* 31

4.2.3 Filtro de *data_units* 31
4.2.4 Interpretador de legendas 32
4.2.5 Sobreposição 35

5 RESULTADOS 37
5.1 Síntese do Demultiplexador 37
5.2 Processador Gráfico 38
5.3 Resultados Práticos 39

6 CONCLUSÃO 41

REFERÊNCIAS 42

ANEXO 43

LISTA DE ABREVIATURAS E SIGLAS

SBTVD	Sistema Brasileiro de Televisão Digital
LaPSI	Laboratório de Processamento de Sinais e Imagens
DELET	Departamento de Engenharia Elétrica
ABNT	Associação Brasileira de Normas Técnicas
UFRGS	Universidade Federal do Rio Grande do Sul
TA	Terminal de Acesso
SO	Sistema Operacional
ARIB	Association of Radio Industries and Businesses
EIA	Electronic Intustries Association
CC	Closed Caption
OSD	On Screen Display
MPEG	Motion Picture Experts Group
TS	Transport Stream — Fluxo de Transporte
PID	Packet Identifier
PES	Packetized Elementary Stream — Fluxo Elementar em Pacotes
FSM	Finite State Machine — Máquina de Estados Finitos
FPGA	Field Programmable Gate Array
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuits
ROM	*Read Only Memory* — Memória Somente Leitura
FIFO	First In First Out
VGA	Video Graphics Array
DVI	Digital Video Interface

LISTA DE FIGURAS

Figura 1.1: Arquitetura básica do receptor, segundo NBR 15604 11

Figura 3.1: Algoritmo para retornar posição inicial dos dados em um pacote MPEG2-TS 22
Figura 3.2: Organização dos pacotes de dados de legendas 23
Figura 3.3: Algoritmo para retornar vetor com comandos *closed caption* 24

Figura 4.1: Arquitetura do sistema 26
Figura 4.2: Máquina de estados do demultiplexador 27
Figura 4.3: Grupo de dados 29
Figura 4.4: FSM que interpreta uma sequência de grupos de dados 30
Figura 4.5: Dados de comandos de legendas 31
Figura 4.6: FSM que interpreta os pacotes *caption_data* 31
Figura 4.7: Unidade de dados 32
Figura 4.8: FSM que interpreta os pacotes *data_unit* 32

Figura 5.1: Placa de FPGA XC2VP30 utilizada nos testes do projeto 37
Figura 5.2: Visão geral e área da tela utilizada para sobrepor as legendas 40

LISTA DE TABELAS

Tabela 3.1: Estatísticas sobre comandos utilizados em legendas 1SEG 24
Tabela 3.2: Estatísticas sobre comandos utilizados em legendas Full-Seg 25

Tabela 4.1: Definição da entidade Demultiplexador 29
Tabela 4.2: Definição da entidade Filtro de Legendas 30
Tabela 4.3: Comandos para legendas 33
Tabela 4.4: Página de símbolos especiais – Grupo GR3 34
Tabela 4.5: Definição da entidade Interpretador de Legendas 36

Tabela 5.1: Resultados da Síntese do Demultiplexador 38
Tabela 5.2: Resultados da Síntese do Filtro de Legendas 38
Tabela 5.3: Resultados da Síntese do Interpretador de Legendas 38
Tabela 5.4: Resultados da Síntese do Componente de Sobreposição 39

RESUMO

Este documento descreve o projeto e implementação de um processador gráfico, um interpretador de *closed caption* e demultiplexador de MPEG2-TS para uso no Sistema Brasileiro de Televisão Digital (SBTVD). O projeto será realizado em duas fases, sendo que a primeira envolve a programação de uma versão utilizando a linguagem C++ para cada um dos componentes identificados, chamados protótipos. Os protótipos são usados para teste de conceito e como fonte de dados para a testes e verificação dos componentes finais. A segunda fase trata da descrição em VHDL dos protótipos, a apresentação de suas arquiteturas e máquinas de estados finitos, além da descrição do comportamento e dos dados que cada componente será responsável por gerenciar. Ao final, tem-se testes do projeto utilizando uma placa de FPGA acoplada a um monitor para a demonstração do funcionamento.

Palavras-chave: Processador Gráfico, TV Digital, Closed Caption, Demux.

Graphic Processor

ABSTRACT

This document describes the project and implementation of a graphic processor, closed caption parser and a MPEG2-TS demultiplexer for use in the Brazilian Digital Television System (SBTVD). The project will have two phases: the first requires the programming of a software version for each one of the identified components, called prototypes. Thoses prototypes will serve for test of concept and as data source for testing purpouses and verification of the final components. The second phase has the VHDL description of the prototypes, the presentation of theirs architectures and finite state machines, besides the description of the behavior and the data that each component will be responsible. At the end, will be presented a test of this project using an FPGA board connected to a monitor to show the system working.

Keywords: Graphic Processor, DTV, Closed Caption, Demux.

1 INTRODUÇÃO

Este trabalho trata do desenvolvimento de tecnologia para Televisão Digital, mais precisamente, para o Sistema Brasileiro de Televisão Digital (SBTVD) visando a apresentação das legendas do tipo *closed caption* enviadas pelas emissoras de TV Digital Aberta, sobre o vídeo transmitido. Os componentes desenvolvidos para este sistema serão inseridos no Terminal de Acesso (TA), que consiste no equipamento conectado entre a antena e o televisor, capaz de decodificar o sinal de TV digital e fornecer ao televisor um sinal compatível à sua entrada.

Uma visão geral do diagrama de blocos do Terminal de Acesso é apresentada na Figura 1.1 e os componentes abordados pelo texto estão salientados.

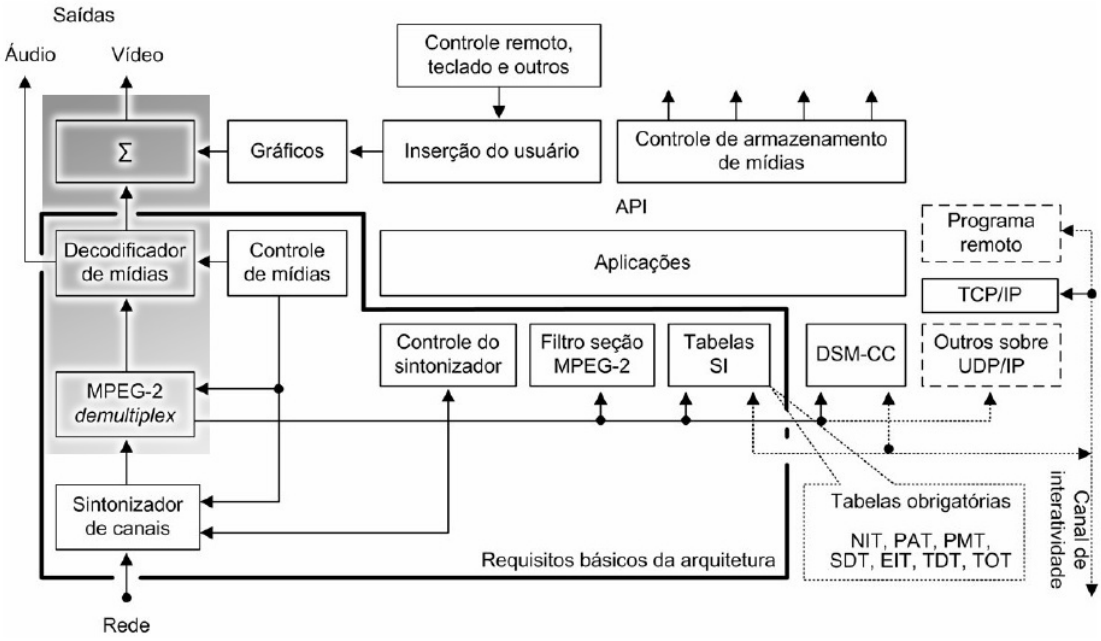

Figura 1.1: Arquitetura básica do receptor, segundo NBR 15604

De um modo geral, será contemplado todo o caminho percorrido pelos dados referentes às legendas desde sua saída do demodulador, e consequente entrada no demultiplexador, até a sua apresentação. Isso implicará apresentar o funcionamento do demultiplexador, dos filtros de pacotes, do interpretador de comandos de legendas e, ainda, a sobreposição do texto ao vídeo.

Serão abordados o desenvolvimento do demultiplexador, decodificador de *closed captions* e compositor gráfico. Demultiplexador é o bloco que recebe o fluxo completo do receptor Full-Seg e transfere cada um dos fluxos presentes para o decodificador correspondente. O decodificador de *closed captions*, que recebe os dados do demultiplexador e gera a imagem para sobreposição está presente no bloco "Decodificador de mídias". Este último é apresentado de forma bastante geral pela Figura 1.1, englobando os decodificadores de vídeo, áudio, legendas e dados relacionados à interatividade, dos quais nos limitaremos apenas ao sistema de legendas. Finalmente a sobreposição é realizada no último bloco, que também é chamado de "Compositor Gráfico", representado pela letra grega sigma maiúscula (Σ) na figura. O Processador Gráfico compreende os blocos de decodificação de legendas e compositor gráfico, consistindo em todo o *hardware* necessário entre a saída do fluxo de *closed captions* do demultiplexador, a saída do decodificador de vídeo e a saída de vídeo na qual o monitor é conectado para apresentar os resultados.

A Norma Brasileira que define os padrões da televisão digital terrestre assinala como opcional o recurso de closed caption para os *set-top-boxes*, entretanto, segundo os resultados do Censo 2000 publicados pelo IBGE (IBGE, 2001), 5,7 milhões de brasileiros tem alguma deficiência auditiva e, destes, 170 mil são declarados surdos. Tendo em mente o fato de que a televisão digital aberta deva ser um instrumento de inclusão social, segundo o próprio Presidente da República em declaração feita em 2 de dezembro de 2007, data da inauguração da TV Digital em São Paulo (TEIXEIRA, 2007), seria uma grande falha não desenvolver meios para que a parte da população com deficiência tenha como desfrutar da inclusão social com um mínimo de qualidade.

Ambientes barulhentos como restaurantes, aeroportos, bares também se beneficiariam do recurso de *closed caption*. Assim, aquele indivíduo que tem interesse na programação exibida teria como extrair boa parte da informação, mesmo com o áudio emudecido, com o volume baixo ou, ainda, outro som ambiente.

1.1 Trabalhos Relacionados

Atualmente, os sistemas que utilizam legendas baseadas em texto, pelo fato de haver suporte em *hardware* à sobreposição de camadas de imagens, grande parte das soluções são realizadas em *software*, o qual renderiza a legenda em uma imagem e esta é sobreposta ao vídeo. Esta abordagem é fortalecida pela crescente utilização do sistema operacional GNU/Linux em equipamentos embarcados, inclusive em TAs, nos quais a adição de funcionalidades pode ser realizada com a implementação de novos módulos de software ou a inclusão de aplicativos e bibliotecas já existentes.

Dos sistemas pesquisados que precederam o projeto e implementação deste trabalho, deu-se destaque aos seguintes por terem características relevantes que se assemelham ao assunto do projeto descrito neste documento.

1.1.1 Microcontrollers for Closed Caption System

Existe um trabalho chamado "MICROCONTROLLERS FOR CLOSED CAPTION SYSTEM" (SHINDO et al., 1992) o qual descreve um sistema independente para ser

usado entre a saída de um receptor e sintonizador do sinal de tv analógica e a entrada de vídeo componente de um televisor.

O sistema em questão é subdividido em duas partes: o Front End, também chamado de Data Slicer; o Sistema Microcontrolado. O Front End é o responsável por extrair do sinal de video o *bitstream* do sinal de closed caption, através da detecção de portadora, geração de sinal de clock e tem como características a detecção e ajuste automático à portadora do sinal de closed caption na linha 21. A responsabilidade de controlar a transformação do sinal de video é do Sistema Microcontrolado, que controla a geração de símbolos gráficos, como caracteres e menus.

Algumas das características do sistema são informadas, como a utilização de memória single port para reduzir o custo total do sistema, assim como área nas pastilhas de silício. Entretanto não há descrição do software que é executado. O presente trabalho difere deste pela adição do demultiplexador necessário à extração dos dados do fluxo de transporte, sem a necessidade de processar dados analógicos, além de uma arquitetura na qual a informação é sempre passada adiante, sem retorno ao sintonizador e sem a utilização de um microcontrolador. Além disso, o sistema atual permite atributos de cor diferentes para cada caracter do texto.

1.1.2 Video Decoder with Closed Caption Data on Video Output

Existe uma patente registrada, chamada "VIDEO DECODER WITH CLOSED CAPTION DATA ON VIDEO OUTPUT" (PATTERSON, 2000), na qual é descrito um arranjo de componentes para a recepção de um fluxo de vídeo comprimido e geração de um sinal de vídeo descomprimido e dados de legendas. Ele emprega o uso de decodificador de vídeo, que descomprime e decodifica o fluxo de dados. Neste caso, o próprio decodificador de vídeo possui um bloco que insere as legendas contidas no fluxo comprimido no sinal de vídeo gerado pelo decodificador. É usada apenas uma saída de vídeo acoplada ao descompressor de vídeo, que gera um sinal de vídeo com as legendas embutidas.

É importante ressaltar que mesmo este sistema possuindo uma arquitetura bastante genérica, ele apenas modifica uma linha do vídeo para a inserção dos dados das legendas, de forma que o televisor, ao receber a imagem opte por exibir ou não as legendas. Assim sendo, a relação com o sistema de closed caption dá-se pela extração, armazenamento e repasse dos dados das legendas, enviando dois caracteres a cada linha 21, sem transformação de texto em imagem, apenas com a conversão do vídeo digital em vídeo analógico.

O que difere no presente trabalho é o fato de o sistema decodificação de legendas ser totalmente independente do vídeo e que as legendas são inseridas sobre a imagem, utilizando a mesma área visível e não como dados para serem interpretadas pelo televisor. Outro detalhe importante é que o arranjo proposto não descreve implementação dos blocos utilizados, como o demultiplexador, mas somente a comunicação e entre eles.

1.1.3 System and Method for Displaying Closed Caption Data on a PC

Outra patente registrada, chamada "SYSTEM AND METHOD FOR DISPLAYING CLOSED CAPTION DATA ON A PC MONITOR" (PRATT, 1996), na qual é descrito

o circuito e algoritmo para captura e decodificação de dados de legendas a partir de um sinal de televisão, juntamente com a imagem para o ambiente de janelas do monitor de um computador pessoal. O dispositivo de decodificação de legendas armazena a linha 21 da imagem, que no sistema analógico costuma carregar dados de legendas, e a envia para o CPU do computador. Este, então, interpreta essa linha da imagem para extrair os dois caracteres de 7 bits, verificando a paridade. Depois esses dados podem ser exibidos em uma janela separada da imagem principal, tendo a posição e tamanho definidos pelo usuário.

O sistema acima descrito tem o enfoque voltado à dificuldade que existia em transferir os dados das legendas para o sistema. Assim, ele copia a informação para dados anexados ao *framebuffer* para poderem ser recebidos pelo computador. Quando a legenda é recebida pelo computador, ele apenas copia o texto para uma janela, sem mencionar a forma de apresentação.

O trabalho desenvolvido neste documento visa a apresentação das legendas sobre o vídeo, da forma correta e não livremente e sem controle como o sistema acima propõe. Para a apresentação das legendas a posição é importante, pois foi definida pela emissora ou pelos produtores da programação de forma a facilitar o entendimento por pessoas com problemas auditivos, tendo indicações de eventos sonoros e posicionamento sobre interlocutores.

1.1.4 Method and Apparatus for Providing Real Time Data on a Viewing Screen Concurrently with any Programing in Process

A patente “METHOD AND APPARATUS FOR PROVIDING REAL TIME DATA ON A VIEWING SCREEN CONCURRENTLY WITH ANY PROGRAMING IN PROCESS” (WILLIAMS, 1997) na qual o sistema descrito utiliza canais de rádio para recebimento em massa de informações e mensagens direcionadas a aparelhos específicos a serem sobrepostas à imagem e ao vídeo. Abrange, também, a possibilidade de utilização de um sintetizador de voz para adicionar áudio ou ainda tons de alerta em caso de mensagens emergenciais para chamar a atenção do usuário da mensagem recebida. As mensagens recebidas em canais pré-definidos, tem seus cabeçalhos verificados, inclusive número de identificação, que é comparado com o identificador do aparelho, quando não for de distribuição em massa.

A seção que define a interferência do sistema no vídeo utiliza uma memória para as informações a serem exibidas (*Display RAM*), outra memória para exibição de gráficos e uma *Character ROM* usada para armazenar os símbolos a serem exibidos, correspondente aos caracteres recebidos. Há um microcontrolador que gerencia todos esses dispositivos e, inclusive, a lógica de saída para geração da imagem final. O algoritmo executado, quando do recebimento de uma mensagem, executa os seguintes passos: leitura dos caracteres; decodificação; atualização da memória de vídeo; aplicar imagem sobreposta; e ativar tons e luzes indicativas. Neste ponto nota-se a utilização da memória para realizar a operação de renderização das mensagens. O presente trabalho tem como objetivo minimizar a quantidade necessária de elementos, inclusive memória, para realizar a tarefa de sobrepor as legendas ao vídeo, de modo a provocar o menor impacto possível na área necessária.

1.1.5 Computer System and Closed Caption Display Method

A patente "COMPUTER SYSTEM AND CLOSED CAPTION DISPLAY METHOD" (TAKEUCHI, 1997) descreve a utilização de *closed captions* em mídias DVD, as quais podem ter inseridas no vídeo a informação através da linha 21. Este sistema detecta a presença dos referidos dados e informa a CPU central usando sinais de interrupção. Quando a CPU recebe os dados da linha, de um registrador de dados do decodificador de DVD, são convertidos em informação de caracteres e essa informação é escrita em uma memória gráfica ou ainda convertida em comandos para controlar a função OSD (On Screen Display), que são enviados à função OSD. Assim, há a interpretação dos dados da linha 21 e posterior escrita em uma área de memória destinada à exibição do texto, mas destacado da imagem, em uma tela de visualização e navegação para a mídia.

O sistema proposto também utiliza a escrita de dados de legenda, entretanto a abordagem utilizada é a escrita em memória de vídeo para posterior exibição. Não existe, nesse caso, uma preocupação com a sobreposição do texto ao vídeo, já que o mesmo é exibido em área diferente da tela. Apesar do tipo padrão de legendas para as mídias DVD são em formato de imagens, o vídeo pode apresentar *closed captions* codificadas diretamente na imagem. Desta forma, o projeto proposto pelo presente documento visa a apresentação das legendas sobre o vídeo, de forma direta, sem a prévia escrita em memória.

1.2 Legendas

Em uma visão simplista, legenda ou *closed caption* é um bloco de texto apresentado sobre o vídeo que é transmitido a um monitor ou televisor conectado à saída do Terminal de Acesso. Para que isso ocorra, é necessário um sistema que tenha a capacidade de extrair os dados e comandos referentes a estas informações a partir do sinal recebido do demodulador, interprete a pilha de pacotes que envolvem a sequência de comandos e os execute.

Ao contrário do sistema de legendas utilizado em mídias DVD-Vídeo, nas quais a informação é baseada em imagens, o formato das legendas do SBTVD é baseado em texto. Esse texto necessita ser convertido em imagem para poder ser apresentado. Isto requer o auxílio de mapas de símbolos para representar o texto recebido.

Esta tarefa requer o estudo das Normas Brasileiras que definem o formato dos dados para legendas, assim como o formato de pacotes para os fluxos de transportes nos quais o padrão é baseado. Por tal motivo, serão descritos desde os pacotes MPEG2-TS, que formam o *bitstream* recebido do demodulador até a sequência de comandos enviada ao Interpretador de Legendas.

1.3 Objetivos

Tem-se como objetivo a implementação de determinados componentes de *hardware* para integração no Terminal de Acesso para TV Digital, em desenvolvimento no Laboratório de Processamento de Sinais e Imagens (LaPSI), do Departamento de Engenharia Elétrica (DELET) da Escola de Engenharia da Universidade Federal do Rio Grande do

Sul. Estes componentes são o Processador Gráfico, que engloba todos os componentes que processam um fluxo de dados específico para *closed caption* desde sua saída do demultiplexador até a sua apresentação em um monitor, e, ainda, o Demultiplexador que extrai cada um dos fluxos elementares de dados a partir dos dados recebidos do demodulador, sem o qual não é possível a obtenção dos dados necessários ao funcionamento do sistema.

Esta implementação dos componentes passará por duas fases: protótipo e *hardware*. Os protótipos são desenvolvidos em *software*, na liguagem de programação C++, simulando o comportamento esperado dos componentes finais. O *hardware* será descrito em VHDL e será validado comparando-se os resultados de simulações com os resultados obtidos dos protótipos.

Espera-se que o *hardware* final tenha a capacidade de isolar o fluxo elementar de *closed captions* inserido dentro do fluxo de transporte MPEG2 recebido na entrada do demultiplexador, interpretar a sequência de comandos e gerenciar o console de texto. Ainda, este último deve poder ser sobreposto ao vídeo recebido pelo sistema, quando enviado a um monitor.

2 METODOLOGIA

O método utilizado no desenvolvimento envolve a aplicação do seguinte procedimento:

- toma-se um componente como base para o trabalho;
- pesquisa-se suas características e define-se um subconjunto a ser implementado, baseando-se em dados estatísticos sobre a análise dos dados de entrada;
- realiza-se a implementação em software do componente, para estudo do algoritmo;
- a partir do momento no qual o protótipo passa a se comportar como esperado, os resultados obtidos com ele são utilizados como vetores de teste para o projeto final, os quais serão usados como guia para o desenvolvimento e depuração do circuito.
- planeja-se uma forma de implementação em hardware, tendo em vista o desempenho esperado do sistema;
- realiza-se a implementação em hardware, e avalia-se o resultado.

2.1 Formato dos Dados de Entrada

O projeto do processador gráfico demandou a pesquisa do sistema de closed caption para o SBTVD, que é baseado no padrão Japonês (ARIB, 2000). Para o Brasil, foi escolhido o conjunto de caracteres latinos apresentados pelas Tabelas 4.3 e 4.4, de acordo com a norma NBR15606-1 (ABNT, 2007).

Também foi necessária a pesquisa do padrão MPEG2-TS, no qual baseia-se a camada de transporte do sistema. Desta forma é possível que, tendo como entrada o sinal completo demodulado do receptor, se extraia apenas as informações relevantes ao desenvolvimento deste projeto.

2.2 Ambiente de Desenvolvimento do Protótipo em Software

A determinação de um ambiente de trabalho é fundamental para a realização de um projeto, sendo que o mesmo envolve tanto um ambiente físico no qual se desenvolve,

quanto as ferramentas necessárias com as quais as idéias tomam forma e geram um produto palpável.

Como ferramentas para o desenvolvimento do código para *software* bastou um computador com o sistema operacional GNU/Linux e compilador GCC. Para a simulação, escolheu-se o *software* GHDL, simulador *open-source* de VHDL fornecido e desenvolvido pela GNU e para visualização dos sinais simulados, o *software* GTKWave Analyzer. A síntese e testes práticos foram realizados com o ambiente Xilinx ISE e a placa de FPGA XUPV2P, conectada a um monitor. Sendo necessários dados para o processamento das legendas, adquiriu-se um receptor de TV Digital Full-Seg, para a captura dos fluxos de dados vindos das emissoras locais de TV Digital Aberta.

2.3 Protótipo Desenvolvido

O protótipo do Processador Gráfico foi escrito em C++ como parte do projeto PRH264 (REFERÊNCIA), o qual será descrito no próximo capítulo, do LaPSI. Foi levada em consideração a abundância em recursos de um computador moderno para acelerar seu desenvolvimento. Assim, a imagem de sobreposição é gerada quando um pacote de closed caption é recebido e mantida em memória por toda a duração da execução do programa, sendo que seu conteúdo é atualizado a cada novo pacote deste mesmo tipo. A cada quadro de vídeo decodificado, é sobreposta essa imagem, respeitando-se as áreas transparentes.

Os pacotes de closed caption, para o protótipo, são analisados e interpretados em blocos, sendo facilmente mantida a sincronia entre os mesmos. Consequentemente, é possível a determinação do início dos dados úteis aplicando-se fórmulas sobre posições específicas da memória de um bloco.

Além do interpretador, foi necessária a implementação de *software* para geração do mapa de bits com os símbolos da fonte utilizada como referência, mapeados no padrão da norma NBR 15606-1.

2.4 Geração de vetores de testes

A partir dos protótipos desenvolvidos, gerou-se vetores de teste, os quais foram usados para validar a simulação do circuito. Os vetores de teste são desde fluxos de dados no formato MPEG2-TS, dados extraídos do demultiplexador para comparação de comportamentos, ou ainda dados específicos para testar determinadas características dos circuitos.

O comportamento dos blocos de *hardware* pode ser analisado a partir dos dados das simulações: componentes que recebem um fluxo de dados e retornam outro, tem suas saídas gravadas em arquivos para comparação binária – este é o caso do demultiplexador e do filtro de legendas; componente que atua sobre uma memória, a partir da execução de uma sequência de comandos, tem suas formas de onda gravadas e a verificação é realizada de forma manual.

A criação de arquivos com fluxos extraídos das emissoras locais foi realizada tanto com a utilização da função de gravação do software do receptor, que pode somente ser

executado em um computador com o sistema operacional Windows, quanto no ambiente de desenvolvimento, com o programa desenvolvido para este fim, utilizando a bilbiotecas DVB-API. A biblioteca DVB-API é utilizada para o controle do receptor de TV digital no SO GNU/Linux.

2.5 Projeto da Versão em Hardware

Ao contrário do que se tinha como plataforma para o protótipo, o circuito final não será implementado em um ambiente com recursos abundantes. Deste modo, foi necessário alterar a abordagem sobre o problema de modo a minimizar o seu custo. A forma que melhor se adapta ao processamento em um FPGA seria uma máquina de estados, já que a implementação de um processador específico para esta tarefa implicaria na necessidade de mais recursos que o necessário e a inserção de latências que certamente atrasariam a obtenção dos resultados, assim como cunsumiriam os parcos recursos disponíveis.

Sendo assim, o projeto final visará a descrição do *hardware* buscando a eficiência na utilização de recursos com a finalidade de se obter um circuito de baixo custo.

2.6 Verificação do Funcionamento em Tempo Real

A verificação dos protótipos foi realizada através da comparação entre os arquivos gerados, para um conjunto de arquivos de testes, pelos protótipos e por um software de referência. No caso do demultiplexador, os fluxos extraídos de um arquivo MPEG2-TS foram comparados com os mesmos extraídos com o programa DGAVCDec[1] e ainda verificação da consistência dos fluxos com decodificadores específicos, como o programa VLC[2], que pode ser usado para os testes do áudio e do vídeo. Ainda foi possível, verificar o funcionamento com o demultiplexador conectado diretamente ao receptor, processando o sinal à medida que é recebido da emissora.

No caso do Processador Gráfico a verificação do protótipo foi baseada na interpretação da norma. Não há disponível software que extraia de um fluxo elementar a sequência de comandos *closed caption* de uma forma que possa ser realizada uma verificação automática, entretanto o receptor traz consigo um software que suporta a exibição do *closed caption* de forma simplificada em uma área anexada ao vídeo, mas sem sobreposição. Assim, tem-se como comparar o resultado dos dois sistemas.

Finalmente, há a verificação do sistema como um todo através da utilização da placa de FPGA

[1]Disponibilizado pelo *site* <http://www.videohelp.com/tools/DGAVCDec>
[2]Disponibilizado pelo *site* <http://www.videolan.org/vlc/>

3 DESENVOLVIMENTO DO PROTÓTIPO EM SOFTWARE

Os protótipos são trechos de códigos-fonte criados com o intuito de teste de conceito e rápida obtenção de resultados, que serão usados para validação dos componentes de *hardware* finais. O *software* desenvolvido não tem compromisso com eficiência, mas sim com a correção dos resultados gerados, os quais devem ser compatíveis com o sistema estudado e passíveis de utilização no procedimento de teste dos componentes de *hardware*.

3.1 O Programa de Referência PRH264

O PRH264 é um *software* desenvolvido no LaPSI, cujo propósito é servir de base para os desenvolvimento de protótipos em *software* de componentes do decodificador de vídeo no padrão H.264, o qual é utilizado pelo SBTVD. Este padrão possui um *software* de referência, disponibilizado pelo Instituto Fraunhofer, o qual é chamado de JM[1] e que está na sua versão 17.2 na data da escrita deste documento. Além disso, o PRH264 permite a gravação dos dados processados para vários de seus componentes, sendo controlado diretamente pela sua tela principal, auxiliando na depuração e desenvolvimento dos componentes.

Como o PRH264 foi totalmente pensado para ser a versão em software de um decodificador de vídeo compatível com o *software* de referência, sua implementação não levava em conta a possibilidade de entradas diferentes de um fluxo H.264. Logo, foi necessária a criação de uma função de entrada de dados nova que suportasse a leitura de fluxos de transporte no padrão MPEG2. Essa função foi a base do Demultiplexador, que mesmo mantendo o comportamento externo de retornar o fluxo de vídeo, tem a capacidade de transferir os demais fluxos de dados para os devidos decodificadores.

3.2 Demultiplexador

O protótipo do demultiplexador também foi implementado como parte do projeto PRH264. Entretanto, como o comportamento básico do programa era apenas decodificar fluxos de vídeo no padrão H.264, foi necessária uma intromissão nas rotinas de leitura para que fosse possível desviar o fluxo de dados com legendas para o subsistema responsável, enquanto fosse transparente ao decodificador de vídeo a presença do demul-

[1] Disponibilidado através do *site* <http://iphome.hhi.de/suehring/tml/>.

tiplexador. O algoritmo para efetuar esta tarefa, é apresentado na Figura 3.1 escrito em pseudo-C.

Esta proposta vem do fato de o receptor enviar para o programa blocos de dados de 188 octetos, já sincronizados. Portanto, pode-se ler os bits específicos das posições de memória para se calcular a posição do início dos dados, em função da presença, ou não, dos vários cabeçalhos e de suas estensões. Cada cabeçalho adiciona o seu tamanho à posição de início dos dados.

Outro protótipo do demultiplexador foi escrito em forma de máquina de estados para se testar o desempenho em processadores que seriam implementados em *hardware*. O desempenho sobre outros processadores (Plasma e Leon, ambos com arquitetura MIPS) em FPGA foi insatisfatório, havendo a necessidade da implementação em *hardware* do próprio algoritmo.

Este novo algoritmo tem como entrada uma função que retorna um octeto do fluxo de transporte e como saída uma função que recebe o valor do octeto a ser gravado e o PID do fluxo elementar. Com estes dois dados, faz-se a seleção do arquivo de saída para gravação, sendo que a gravação de dados só é chamada quando os dados lidos fazem parte apenas do fluxo elementar de um determinado PID. A execução do demux é realizada lendo-se o arquivo de entrada até que o mesmo chegue ao final. Inicialmente é verificado o sincronismo dos pacotes, após trata-se os cabeçalhos e finalmente se transfere os dados para o fluxo elementar apropriado.

A máquina de estados para esta última versão será detalhada mais profundamente no próximo capítulo, no qual servirá de base para o desenvolvimento do demultiplexador em *hardware* descrito em VHDL.

3.3 Processador Gráfico

O protótipo do processador gráfico trata de códigos adicionados ao atual projeto PRH264. Desenvolvido na linguagem C++, recebe o fluxo de dados vindos do demultiplexador, desenha sobre uma imagem *bitmap*, a qual é sobreposta à imagem vinda do decodificador de vídeo H.264.

Os dados vindos do demultiplexador são em blocos de tamanho arbitrário, mas menores do que 188 octetos. Destes blocos são extraídos o fluxo de *data_units*, que são os comandos para o interpretador de *closed captions*. Entretanto há alguns níveis na hierarquia de pacotes, que devem ser respeitados. A organização destes pacotes de dados é mostrada na Figura 3.2.

Os dados recebidos são *data_groups*, grupos de dados com cabeçalho de cinco octetos, carga de dados e rabeira de dois octetos. O cabeçalho contém informações e identificador de grupo, versão, dados de gerenciamento do fluxo de grupos de dados e tamanho da carga transportada, em número de octetos. A rabeira possui dois octetos formando o valor para verificação de erros do tipo CRC16.

A carga de dados transportada pode ser tanto do tipo *caption_management_data* (dados de gerenciamento de legendas), quanto *caption_statement_data* (dados de comandos

```
// parâmetros:
// packet    -- pacote MPEG2-TS (188 bytes)
// retorno: >= 0 em caso de sucesso
//          -1 em caso de falha
int retornaInicioDosDados(uchar *packet){
  int pos, pid, stream_id;

  // cálculo da posição do payload, levando em consideração o A.F.
  pos = ((packet[3]&0x20)?5+packet[4]:4);

  // montar o valor do PID
  pid = ((packet[1]<<8) | packet[2]) & 0x1fff;
  if(packet[1]&0x40)
  {
      // Payload Unit Start Indicator == 1
      // é início de pacote PES

      if(!((packet[pos] == 0) && (packet[pos+1] == 0) &&
           (packet[pos+2] == 1)))
      {
          // não possui cabeçalho PES
          if((pid == PID_PAT) || (pid == PID_NIT) ||
             (pid == PID_PMT) || ((pid & 0x1fff) == 0x1fc8))
          {
              // é uma tabela
              return 5;
          }
          return -1;
      }
      else
      {
          // pos está no início do cabeçalho do PES
          stream_id = packet[pos+3];

          // pula o cabeçalho PES
          pos += 9 + ts_pack[pos+8];
          if(stream_id == 0xBD || stream_id == 0xBF)
          {
              // syncronized_PES_data, Asynchronous_PES_data
              // possuem cabeçalho também
              if((ts_pack[pos] & 0xfe) == 0x80)
                  pos += (ts_pack[pos+2] & 0x0f) + 3;
          }
      }
  }
  return pos;
}
```

Figura 3.1: Algoritmo para retornar posição inicial dos dados em um pacote MPEG2-TS

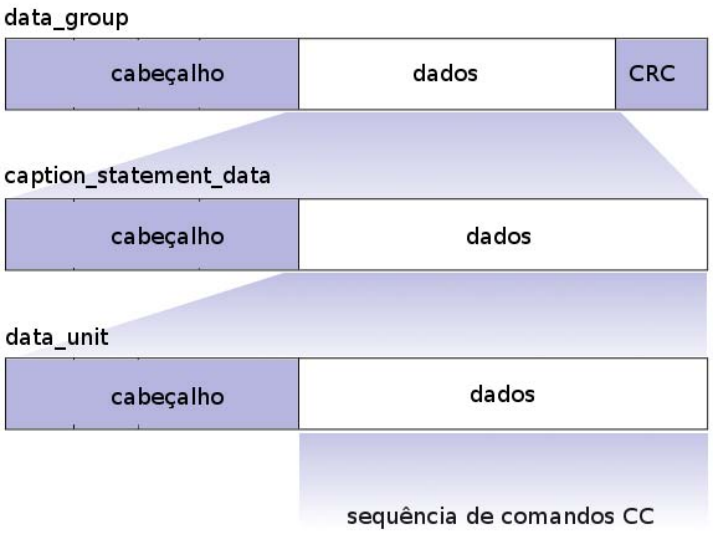

Figura 3.2: Organização dos pacotes de dados de legendas

de legendas). Nos dados de gerenciamento podem ser encontradas informações sobre a quantidade de linguagens disponíveis, modo de apresentação, código de três letras da linguagem de acordo com a ISO 639, formato e direção de escrita na tela, além dos modos de rolagem. Nos dados dos comandos, tem-se apenas informação sobre temporização da apresentação da legenda transportada.

Para a extração dos comandos que serão interpretados e gerarão a imagem para sobreposição, optou-se por ler os dados dos cabeçalhos e calcular o valor da posição de início dos dados.Realizando-se este procedimento para cada cabeçalho lido, ao final tem-se o valor de início do fluxo de comandos e seu tamanho, em octetos. O algoritmo para efetuar esta tarefa, é apresentado na Figura3.3 escrito em pseudo-C.

A interpretação dos comandos é realizada a partir de um laço que varre o vetor com os comandos, repassando a posição corrente do vetor a uma seleção que verifica se o valor é um comando ou dado e executa os passos necessários para cada caso.

Grande parte dos comandos apenas mudam o estado atual das variáveis utilizadas para o desenho na imagem de sobreposição, como definir a cor do texto, a cor de fundo, transparências, o tamanho dos caracteres, a posição atual para inserção de texto, entre outros. Alguns comandos como o CS, por exemplo, determina que a imagem de sobreposição seja reinicializada para transparente, o que necessita de mais tempo e uma grande quantidade de acessos à memória.

Com base na análise realizada sobre um fluxo de legendas recebidas com mais de duas horas de duração, compreendendo um programa de auditório e um filme, tem-se as taxas de comandos utilizados, apresentadas na Tabela 3.1:

O mesmo trecho foi, também, analisado o fluxo de legendas voltadas ao vídeo com alta definição, cujas taxas são apresentadas na Tabela 3.2. Como os dados são coletados a partir da presença ou não do comando em uma legenda, faz-se necessário informar que o comando CSI pode ser desdobrado em vários outros. Neste caso ele informa o

```
// parâmetros:
// in_data    -- data_group
// in_size    -- tamanho do data_group
// out_data   -- data_unit_data_bytes
// out_size   -- data_unit_size
bool retornaComandosCC(uchar *in_data, int in_size,
                       uchar *out_data, int *out_size){
  int inicio = 6;  // posição mínima para iniciar os dados
  int data_group_id;

  // daata_group_id, desconsiderando o grupo
  data_group_id = (in_data[0] >> 2) & 0x1f;

  // verifica se o TMD do caption_data = '10' (binário)
  if( in_data[5] & 0xc0 == 0x80 ) // caption_data.TMD = 10?
    inicio += 5;                  // STM (offset no tempo)

  if(data_group_id != 0){
    inicio += 3; // caption_data.data_unit_loop_length

    if(in_data[inicio] == 0x1f) {       // US: unit separator
      if(in_data[inicio+1] == 0x20){    // parâmetro CC
        out_data = (in_data[inicio+2]<<16) |
            (in_data[inicio+3]<<8) | (in_data[inicio+4]);
        inicio += 5;
        out_data = in_data + inicio;
        return true;
      }
    }
  }

  return false;
}
```

Figura 3.3: Algoritmo para retornar vetor com comandos *closed caption*

Comando	**Ocorrências**	**Taxa**
CS (Limpar tela)	2678	100,00%
WHF (Texto na cor branca)	2678	100,00%
MSZ (Caracter de tamanho médio)	2678	100,00%
APR (Nova linha)	2028	75,73%
SS3 (Shift Simples)	70	2,61%
APF (Avanço para frente)	37	1,38%

Tabela 3.1: Estatísticas sobre comandos utilizados em legendas 1SEG

Comando	**Ocorrências**	**Taxa**
CS (Limpar tela)	2316	100,00%
CSI (Comando estendido)	2316	100,00%
APS (Definir posição do texto)	2316	100,00%
SSZ (Caracter de tamanho pequeno)	2316	100,00%
COL (Definir cor)	2316	100,00%
APR (Nova linha)	1893	81,74%
SS3 (Shift Simples)	70	3,02%
APF (Avanço para frente)	46	1,99%

Tabela 3.2: Estatísticas sobre comandos utilizados em legendas Full-Seg

posicionamento do console sobre o vídeo, a largura e altura da área ocupada pelo console para as legendas, o tamanho e espaçamento entre símbolos e a quantidade de caracteres por linha e por coluna. Tem-se a impressão de que no início de cada legenda, a emissora envia um bloco padrão de comandos para definir o ambiente e, após, envia os comandos específicos para a legenda. Esta seria a única explicação para se ter 100% de presença em alguns dos comandos.

4 DESENVOLVIMENTO DOS COMPONENTES EM HARDWARE

O desenvolvimento do projeto foi dividido em componentes menores para reduzir o escopo de cada componente, definindo interfaces entre eles de modo a permitir a implementação interna de cada um não influenciar na implementação dos demais.

A Figura 4.1 apresenta o caminho de dados através do sistema, até sua saída para um monitor. O decodificador de vídeo foi pontilhado para simbolizar o fato de que ele está fora do escopo deste projeto, mas mesmo assim é necessário informar sua localização.

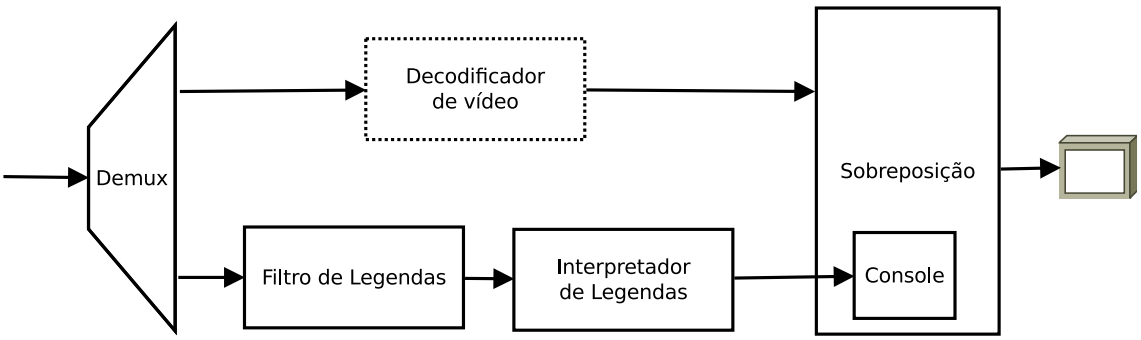

Figura 4.1: Arquitetura do sistema

4.1 Demultiplexador

O componente demultiplexador foi um tanto mais complexo de se desenvolver o *hardware* pelo fato do protótipo ser muito baseado em sequências de passos. Isso acabou por tornar-se um transtorno em função do paralelismo inerente das operações em *hardware* na transcrição da máquina de estados.

Vários das operações, nas quais eram testados índices ou somados *offsets*, necessitaram ajustes de modo a proporcionar o resultado esperado. Após os ajustes necessários, foi possível iniciar a extração dos fluxos elementares a partir do fluxo de transporte.

A versão final possui vinte estados, nos quais há uma combinação entre a interpretação dos pacotes TS,PES e PES (as)síncronos de forma a entregar apenas a informação realmente necessária para os decodificadores, além de minimizar o atraso entre a entrada e a saída para apenas um ciclo de relógio. Esta combinação aumentou o nível de com-

plexidade do circuito ao tratar de três níveis diferentes.

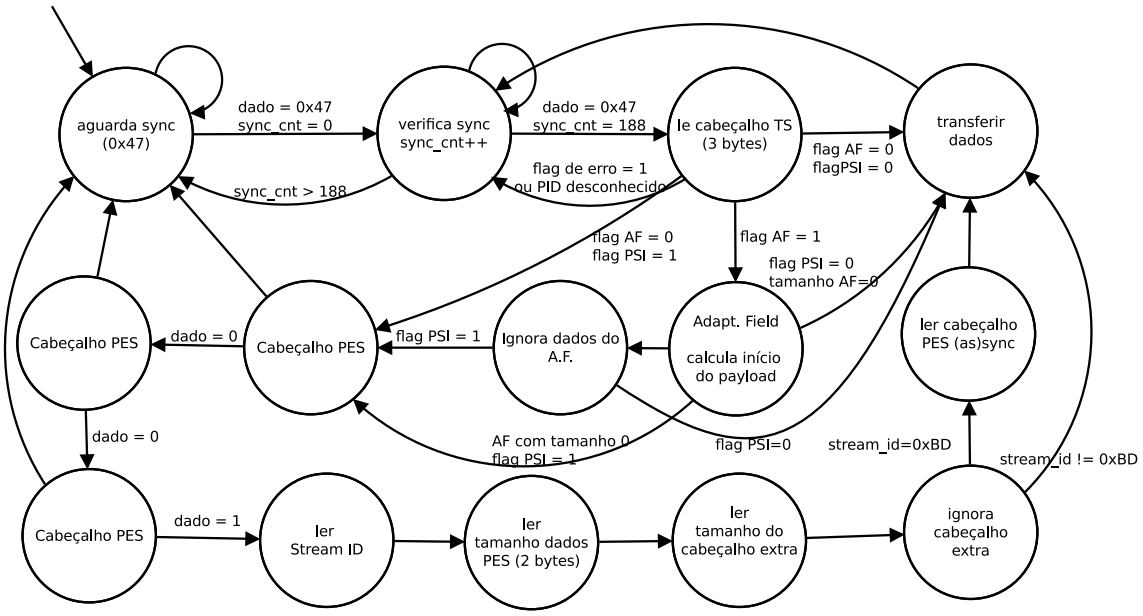

Figura 4.2: Máquina de estados do demultiplexador

A máquina de estados do demultiplexador é apresentada na Figura 4.2. O estado inical tem a finalidade de aguardar da entrada um octeto com o valor 0x47, chamado de sincronismo, que indica o início de um pacote de transporte. Recebido este valor, o contador sync_count recebe o valor 1 (um) e passa-se ao próximo estado que tem a finalidade de verificar o sincronismo e o tamanho do pacote. O valor do contador sync_count é incrementado a cada novo octeto lido da entrada. Quando é recebido outro octeto com o valor 0x47 e o valor de sync_count é 188, sync_count recebe o valor 1 novamente e passa-se ao próximo estado para a leitura do cabeçalho de transporte. Caso sync_count passe de 188, retorna-se ao estado inicial para recomeçar o sincronismo dos pacotes. Este estado também é utilizado quando chega-se à conclusão de que o restante dos dados do pacote de transporte atual não serão utilizados.

A leitura do cabeçalho de transporte é realizada em um único estado, em três ciclos diferentes. No terceiro ciclo decide-se qual será o próximo estado em função dos *flags* presentes no primeiro dos três octetos. Se o pacote apresenta erros que não puderam ser corrigidos pelo demodulador, retorna-se ao estado de verificação de sincronismo para aguardar o próximo pacote, sem transferir os dados deste. Se este pacote de transporte carrega o início de um pacote do fluxo elementar (PES), segue-se para a interpretação do deste no estado “Cabeçalho PES”. Caso exista um campo de adaptação (AF), ele é lido seguindo-se ao estado “Adapt. Field”. Ainda há a possibilidade dos dados fazerem parte do meio de um pacote PES, sendo assim, apenas necessário seguir para o estado de transferência de dados e aguardar o final do pacote.

O estado “Adapt. Field” tem por finalidade reconhecer o tamanho do campo de adaptação para que seja possível acessar os dados seguintes. Normalmente o campo de adaptação é utilizado para preencher o pacote de transporte quando os dados do fluxo elementar são insuficientes. Isso ocorre geralmente na última parte que completa um pacote PES, em função da fragmentação dos dados que podem não ter um tamanho múltiplo de 184. Reconhecido o tamanho do campo de adaptação, segue-se ao próximo estado que

lê da entrada a quantidade de octetos presentes no campo de adaptação. Caso o tamanho informado seja zero, tem-se duas opções: se tem-se o início de um pacote PES, segue-se à leitura deste cabeçalho, do contrário segue-se à transferência dos dados.

A leitura do cabeçalho dos pacotes PES é realizada em três estados, os quais leem os três octetos de sinalização de início de pacote, que são [0x00, 0x00, 0x01]. Caso essa sequência não seja lida, percebe-se um erro e o controle é transferido ao estado de verificação de sincronismo. Validada a sequência, lê-se o identificador de fluxo (*stream_id*), que identifica o conteúdo do fluxo elementar, de acordo com a faixa de valores na qual se encontra:

0xBD Fluxo de *closed captions*

0xC0 - 0xDF Fluxo de Áudio

0xE0 - 0xEF Fluxo de Vídeo

Após a leitura do *stream_id*, os próximos dois octetos formam o tamanho do pacote PES, que deve ser lido. Em seguida, há três octetos da extensão do cabeçalho PES. Nestes dados há a informação sobre a existência de mais campos de cabeçalho, inclusive o tamanho que estas novas informações ocupam, esta última presente no terceiro octeto. Tendo-se a quantidade de dados da continuação do cabeçalho, pode-se seguir a leitura dos dados de entrada até que todo o cabeçalho tenha sido lido. Em seguida, sendo um fluxo de áudio ou vídeo, passa-se ao estado de transferência de dados. Caso contrário, há ainda o cabeçalho PES (as)síncrono com mais informações a serem tratadas.

O cabeçalho PES (as)síncrono é interpretado em 4 estados, sendo 2 estados para a leitura do identificador de dados (*data_identifier*) e do identificador de fluxo privado (*private_stream_id*). O terceiro estado é utilizado para a leitura da quantidade de dados privados e o quarto estado é utilizado para ler estes dados privados do fluxo de entrada.

A definição da entidade do Demultiplexador em VHDL é descrita na Tabela 4.1. Além da interface padrão de filtro de dados, foi adicionada a saída que informa o PID dos dados repassados. Desta forma, a seleção do decodificador pode ser realizada externamente ao demultiplexador, já que os PIDs serão informados através do processador central do sistema pela leitura das tabelas PAT e PMT.

4.2 Processador Gráfico

O Processador gráfico em *software* possui características, como a dependência da sequência das operações, as quais não são muito bem-vindas quando descritas em *hardware*, em função do seu paralelismo inerente. Por tal motivo, foi necessário dividi-lo em cinco partes, as quais possuem funções específicas.

A arquitetura básica dos blocos filtrantes apresenta entradas de sinal de relógio, reset, dados com oito bits de largura e habilitação. As saídas são dados, também com oito bits de largura, e sinal de validade dos dados. Assim, o sinal de dados válidos de um bloco pode ser utilizado para habilitar o bloco seguinte e o fluxo de informação se mantém estável.

```
ENTITY Demultiplexador IS
    PORT    (
        clk_i            : IN STD_LOGIC;
        rst_i            : IN STD_LOGIC;

        -- input interface
        input_i          : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
        en_i             : IN STD_LOGIC;

        -- output interface
        pid_o            : OUT STD_LOGIC_VECTOR (12 DOWNTO 0);
        data_o           : OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
        data_valid_o     : OUT STD_LOGIC
    );
END Demultiplexador;
```

Tabela 4.1: Definição da entidade Demultiplexador

Quando um bloco do início do caminho de dados está interpretando um cabeçalho, os demais estarão desabilitados aguardando novos dados válidos.

Como todos os blocos operam na mesma frequência, a taxa de transferência de dados entre eles é sempre a mesma, entretanto a taxa de dados válidos por unidade de tempo em cada interconexão de bloco é reduzida ao longo do caminho dos dados até o interpretador de *closed captions*.

A definição da entidade do Filtro de Legendas em VHDL é descrita na Tabela 4.2, que apresenta uma interface padrão de filtro de dados, cuja largaura é de oito bits. A entrada de habilitação é utilizada para informar que existem novos dados na entrada, que é lida a cada ciclo de relógio. Este filtro é formado pelos filtros de *data_groups*, de *caption_data* e de *data_units* em sequência, tendo a saída conectada ao Interpretador de Legendas.

4.2.1 Filtro de *data_groups*

A implementação do filtro de *data_groups* foi realizada com base na sua estrutura de dados, apresentada na Figura 4.3.

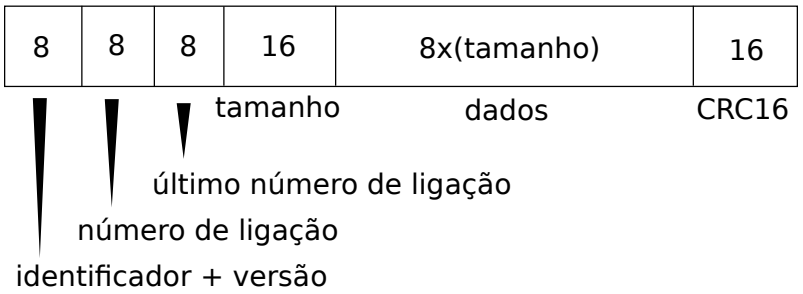

Figura 4.3: Grupo de dados

A máquina de estados necessária para interpretar essa entrutura de dados é bastante

```
ENTITY cc_filter IS
    PORT    (
        clk_i           : IN STD_LOGIC;
        rst_i           : IN STD_LOGIC;

        -- input interface
        input_i         : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
        en_i            : IN STD_LOGIC;

        -- output interface
        data_valid_o    : OUT STD_LOGIC;
        data_o          : OUT STD_LOGIC_VECTOR (7 DOWNTO 0)
    );
END cc_filter;
```

Tabela 4.2: Definição da entidade Filtro de Legendas

simples e está representada na Figura 4.4. São utilizados três estados para percorrer os octetos do cabeçalho. Em seguida, obtém-se o tamanho da carga de dados do pacote *data_group* em dois estados. Esse valor será decrementado uma vez por octeto lido da entrada, para os próximos dados, que representa a carga de dados. Neste estado repassa-se os dados de entrada para a saída até o contador voltar a ser zero. Então lê-se os dois octetos que formam o CRC16 e volta-se ao estado inicial.

A identificação de um *data_group* pode ter o valor nas faixas 0x00 até 0x08 e 0x20 até 0x28. Os valores 0x00 e 0x20 informam que a carga de dados apresenta dados de gerenciamento. Os demais valores representam dados de legendas para cada uma das oito possíveis linguagens.

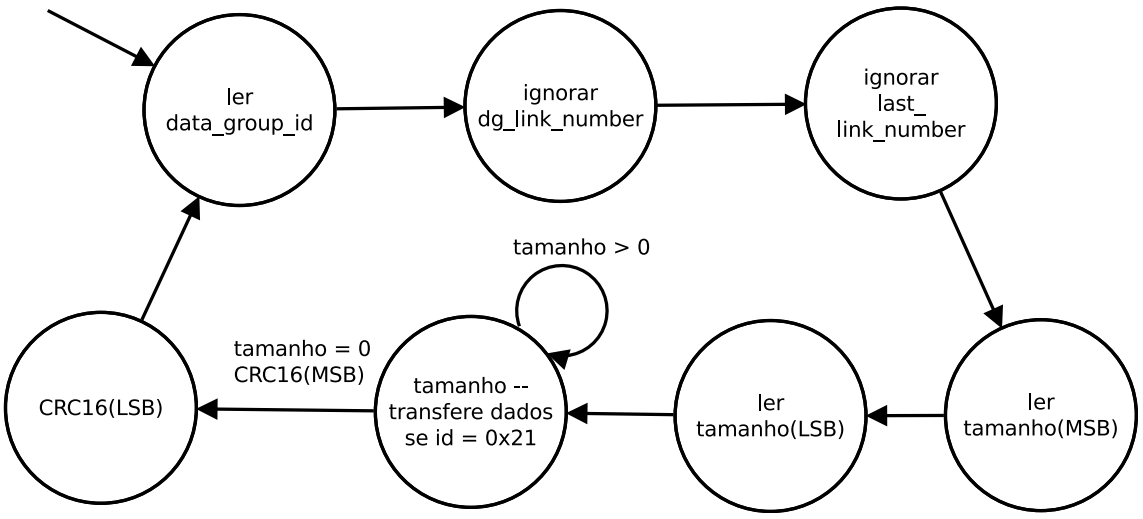

Figura 4.4: FSM que interpreta uma sequência de grupos de dados

4.2.2 Filtro de *caption_data*

A estrutura interpretada por este filtro contém os comandos das legendas que serão exibidas na tela, encapsuladas por estruturas chamadas *data_units*. Portanto, somente são recebidos os pacotes com identificação 0x01 ou 0x21, da primeira linguagem disponível, caso exista. Há, também a possibilidade de que o usuário escolha qual das legendas presentes será exibida. A estrutura destes pacotes são apresentadas na Figura 4.5.

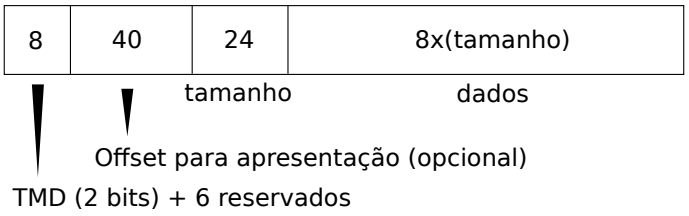

Figura 4.5: Dados de comandos de legendas

A máquina de estados é representada na Figura 4.6. O estado inicial lê o primeiro octeto, dos quais os dois bits mais significativos formam o campo TMD (Time Control Mode). Se o valor deste campo é '01' ou '10', segue um campo extra chamado STM (Presentation Start-Time), o qual tem o tamanho de 36 bits formando um número BCD de 9 dígitos, o qual indica o instante de tempo em que os dados devem ser apresentados, seguidos de 4 bits reservados. Em seguida lê-se em três estados o tamanho do bloco de dados, simplificados no diagrama por um único estado. Finalmente o estado no qual os dados de entrada são transferidos para a saída é alcançado. A cada octeto lido da entrada o registrador com o tamanho dos dados é decrementado. Quando seu valor alcança zero, há uma transição para o estado inicial e o cliclo recomeça para a interpretação de um novo *caption_statement_data*.

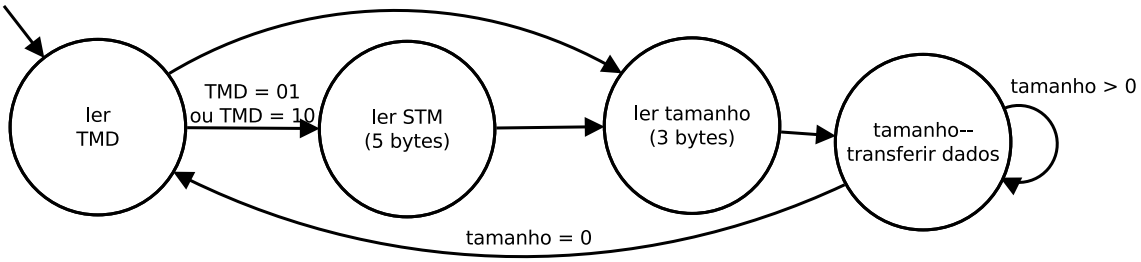

Figura 4.6: FSM que interpreta os pacotes *caption_data*

4.2.3 Filtro de *data_units*

Os pacotes denominados *data_units* são semelhantes aos *caption_data*, tendo um cabeçalho que contém um separador como octeto inicial, um parâmetro e o tamanho dos dados que seguem. A estrutura é apresentada na Figura 4.7.

Finalmente, a carga de dados desse pacote pode ser repassada integralmente ao interpretador de legendas, sendo uma sequência de comandos específicos para a geração da imagem de sobreposição.

A máquina de estados que interpreta um pacote *data_unit* é apresentada na Figura 4.8, sendo que seu estado inicial faz a leitura do separador, cujo valor deve se 0x1F. Quando

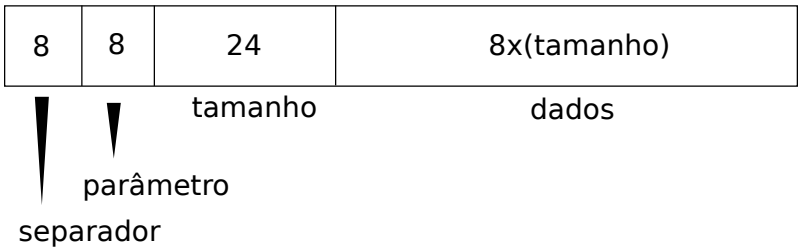

Figura 4.7: Unidade de dados

um separador é encontrado, passa-se ao segundo estado que verifica se o parâmetro tem o valor 0x20, que representa uma legenda do tipo *closed_caption*. Este parâmetro definirá se a informação será transferida ou não, posteriormente através de um bit de ativação, armazenado em um único *flip-flop*. Em seguida lê-se os 3 octetos que formam o tamanho do bloco de dados em três estados distintos, os quais foram agrupados em um único estado no diagrama. Finalmente, alcança-se o estado de transferência de dados, que decrementa o contador de tamanho e utiliza como sinalização de validade de dados o valor do bit de ativação. Deste modo evita-se basear a espera do próximo bloco *data_unit* através da comparação do octeto de entrada com o valor 0x1F, pois esse mesmo valor poderia ocorrer dentro do bloco de dados, gerando um comportamento incoerente.

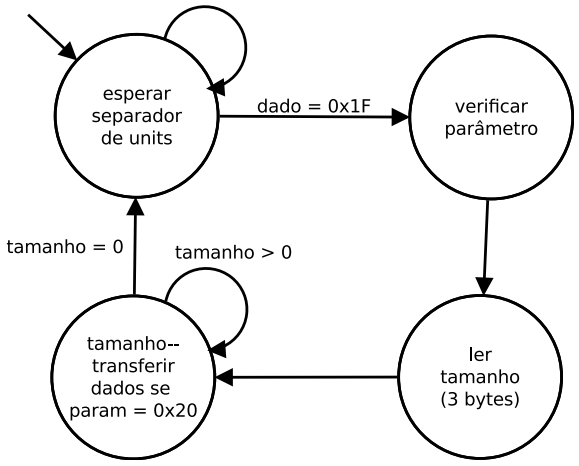

Figura 4.8: FSM que interpreta os pacotes *data_unit*

4.2.4 Interpretador de legendas

O interpretador de legendas é um componente de *hardware* que tem como entradas um fluxo de comandos para geração de legendas e como saída endereço e dados para a gravação em uma memória que representará um console de texto, o qual será sobreposto ao vídeo decodificado.

Este interpretador busca comando após comando no fluxo de dados, executando ações específicas para cada valor, operando de forma semelhante a um processador CISC. Os comandos são apresentados na Tabela 4.3.

Os comandos que possuem uma semântica associada estão nas colunas com fundo cinza, ou seja, de 0x00 a 0x1f e de 0x80 a 0x9f. Os demais valores representam o caracter

Tabela 4.3: Comandos para legendas

	0x	1x	2x	3x	4x	5x	6x	7x	8x	9x	Ax	Bx	Cx	Dx	Ex	Fx
x0	NUL		SP	0	@	P	`	p	BKF	COL	10/0	°	À	Đ	à	õ
x1			!	1	A	Q	a	q	RDF	FLC	¡	±	Á	Ñ	á	ñ
x2			"	2	B	R	b	r	GRF	CDC	¢	²	Â	Ò	â	ò
x3			#	3	C	S	c	s	YLF	POL	£	³	Ã	Ó	ã	ó
x4			$	4	D	T	d	t	BLF	WMM	€	Ž	Ä	Õ	ä	ô
x5			%	5	E	U	e	u	MGF	MACRO	¥	µ	Å	Õ	å	õ
x6		PAPF	&	6	F	V	f	v	CNF		Š	¶	Æ	Ö	æ	ö
x7	BEL		'	7	G	W	g	w	WHF	HLC	§	·	Ç	×	ç	÷
x8	APB	CAN	(	8	H	X	h	x	SSZ	RPC	š	ž	È	Ø	è	ø
x9	APF	SS2	)	9	I	Y	i	y	MSZ	SPL	©	¹	É	Ù	é	ù
Xá	APD		*	:	J	Z	j	z	NSZ	STL	ª	º	Ê	Ú	ê	ú
xB	APU	ESC	+	;	K	[	k	{	SZX	CSI	«	»	Ë	Û	ë	û
xC	CS	APS	,	<	L	\	l	|			¬	Œ	Ì	Ü	ì	ü
xD	APR	SS3	-	=	M	]	m	}		TIME	ÿ	œ	Í	Ý	í	ý
xE	LS1	RS	.	>	N	^	n	~			®	Ÿ	Î	Þ	î	þ
xF	LS0	US	/	?	O	_	o	DEL			¯	¿	Ï	ß	ï	15/15

que será inserido na posição corrente do cursor no console de texto.

Os comandos que alteram a posição do cursor são:

APB Voltar uma posição, similar à função da tecla *Backspace*.

APF Avançar uma posição, similar à função da tecla Espaço.

APD Avançar uma posição abaixo.

APU Avançar uma posição acima.

APR Nova linha, similar à função da tecla *Enter*.

APS Define nova posição, requer dois parâmetros para indicar linha e coluna.

Conjunto de comandos que alteram a página de símbolos atual:

LS0 Invoca troca de página de símbolos.

LS1 Invoca troca de página de símbolos.

SS2 Invoca troca de página de símbolos para o próximo comando somente.

SS3 Habilita a página de símbolos especiais para o próximo comando somente, apresentada na Tabela 4.4.

Conjunto de comandos que alteram a cor dos próximos símbolos:

BKF Texto preto.

RDF Texto vermelho.

GRF Texto verde.

Tabela 4.4: Página de símbolos especiais – Grupo GR3

	0x	1x	2x	3x	4x	5x	6x	7x
x0				¤	…			
x1			♪	¦	▉			
x2				¨	‘			
x3				´	’			
x4				¸	“			
x5				¼	”			
x6				½	•			
x7				¾	™			
x8					⅛			
x9					⅜			
xA					⅝			
xB					⅞			
xC								
xD								
xE								
xF								

YLF Texto amarelo.

BLF Texto azul.

MGF Texto magenta.

CNF Texto ciano.

WHF Texto branco.

COL Comando composto, depende dos parâmetros. Pode definir tanto a cor do texto, quanto a cor do plano de fundo, inclusive transparência.

Conjunto de comandos que alteram o tamanho dos caracteres:

SSZ Tamanho pequeno para os caracteres.

MSZ Tamanho médio para os caracteres.

NSZ Tamanho normal para os caracteres.

SZX Define o tamanho dos caracteres, de acordo com os parâmetros que acompanham o comando.

Os demais comandos de um octeto são pouco utilizados, dado que em todos os fluxos analisados durante o presente ano não houve ocorrência dos mesmos. O comando CSI (Introdutor de Sequência de Controle) é bastante complexo, derivando vários sub-comandos os quais determinam as características das legendas usadas no sistema de alta definição.

Para tal tarefa, é necessário manter registradores com o valor da posição atual do cursor, das cores ativas para texto e fundo e tamanho dos caracteres. A máquina de estados gerencia boa parte dos comandos simples em apenas um estado. Os demais estados são utilizados para varreduras no console, limpeza da tela, interpretação de parâmetros e comandos mais complexos.

Além disso, como existem operações que são executadas em mais de um ciclo de relógio, é necessário, dentro do interpretador, uma fila FIFO que possa armazenar os dados seguintes enquanto uma operação está em andamento. Geralmente o primeiro comando de uma sequência é o CS, ou limpar a tela, e ele força a escrita em todas as posições de memória. Isso faz com que seja necessário acumular os demais comandos da sequência. Considerando o tamanho de um único pacote como tendo 188 octetos e o tamanho mínimo dos cabeçalhos inseridos por cada camada subsequente, tem-se 156 octetos como máxima sequência de comandos, pois 32 perdem-se em cabeçalhos. Destes 32 octetos considera-se 4 para o cabeçalho TS, 9 para o PES, 3 para o PES (as)síncrono, 7 para o *data_group*, 4 para o *caption_data* e 5 para o *data_group*. Dados os tempos para preenchimento da fila, e atrasos decorrentes das operações mais complexas, tem-se como tempo necessário para o console apresentar a legenda completa em torno de

$$t = \frac{tamanho_{fila} + ciclos_{CS} + comandos}{f_{OSC}}$$

Tomando como f_{OSC} o valor de 2,5MHz, o tamanho da fila de 156 octetos, 160 ciclos (4 linhas $\times$ 40 colunas) para o comando CS e os demais 155 comandos, tem-se $t = \frac{471}{2500000} \approx 0,19ms$, como o tempo máximo de processamento de uma legenda. Uma legenda tão longa necessitaria de um tempo de leitura bastante longo, devendo ser de alguns segundos sem novas informações. Na prática, recebendo-se três pacotes quaisquer entre uma legenda e outra é o suficiente para o bom funcionamento do sistema, apesar de a primeira legenda não ter tempo suficiente de ser exibida ou percebida pelo usuário.

A definição da entidade do Interpretador de Legendas em VHDL é descrita na Tabela 4.5, tendo como entrada um fluxo de comandos devidamente filtrado. A saída é conectada à memória do console, que é acessada diretamente através do endereçamento e da sinalização de escrita.

4.2.5 Sobreposição

A sobreposição de vídeo é alcançada com a leitura da memória do console de texto ao mesmo tempo em que o vídeo é recebido do decodificador. Assim, gera-se apenas um pixel por vez, minimizando a quantidade de memória necessária à tarefa.

O componente que realiza a sobreposição das legendas ao vídeo necessita, também, de uma ROM, a qual armazenará a representação gráfica dos símbolos que serão visualizados ao aplicar a legenda sobre o vídeo. Para reduzir a quantidade de memória necessária, os grupos de caracteres são mapeados para uma sequência mais compacta de endereçamento, na qual as faixas de valores nas quais encontram-se comandos também são utilizadas para dados. Deste modo, utiliza-se menos de 256 símbolos diferentes, de acordo com as colunas brancas não vazias das Tabelas 4.3 e 4.4. Logo, como tem-se 12 colunas de caracteres nas duas primeiras páginas do conjunto latino de caracteres e 3 colunas no grupo GR3, de caracteres especiais, totalizando 15 colunas. Cada símbolo é constituído de uma imagem binária de 8 *pixels* de largura, por 16 *pixels* de altura, sendo que a organização em memória mapeia cada uma das 16 linhas de 8 *pixels* de cada símbolo para um octeto. Com este procedimento, cada símbolo consome 16 octetos na memória, necessitando de apenas 3808 octetos para todo o mapa de caracteres, que será armazenado em uma *Block RAM* no FPGA.

```
ENTITY cc_parser IS
    generic (col_adrs_bits : positive:= 6;
             row_adrs_bits : positive:= 2); -- valor padrão
    PORT    (
        clk_i         : IN std_logic;
        rst_i         : IN std_logic;

        -- input interface
        input_i       : IN std_logic_vector (7 DOWNTO 0);
        en_i          : IN std_logic;

        -- output interface
        col_address_o : OUT std_logic_vector(col_adrs_bits-1 downto 0);
        row_address_o : OUT std_logic_vector(row_adrs_bits-1 downto 0);
        data_o        : OUT std_logic_vector(15 downto 0);
        write_o       : OUT std_logic
    );
END cc_parser;
```

Tabela 4.5: Definição da entidade Interpretador de Legendas

5 RESULTADOS

Os resultados de síntese dos componentes foram obtidos com o auxílio da ferramenta Xilinx ISE versão 10.1 para o dispositivo XC2VP30 (2vp30ff896) mostrado na Figura 5.1, a partir dos arquivos VHDL. Cada um dos componentes foram sintetizados isoladamente para a obtenção de resultdos mais localizados. Evitando-se, assim, mesclar dados do sistema antes existente às informações dos novos módulos criados.

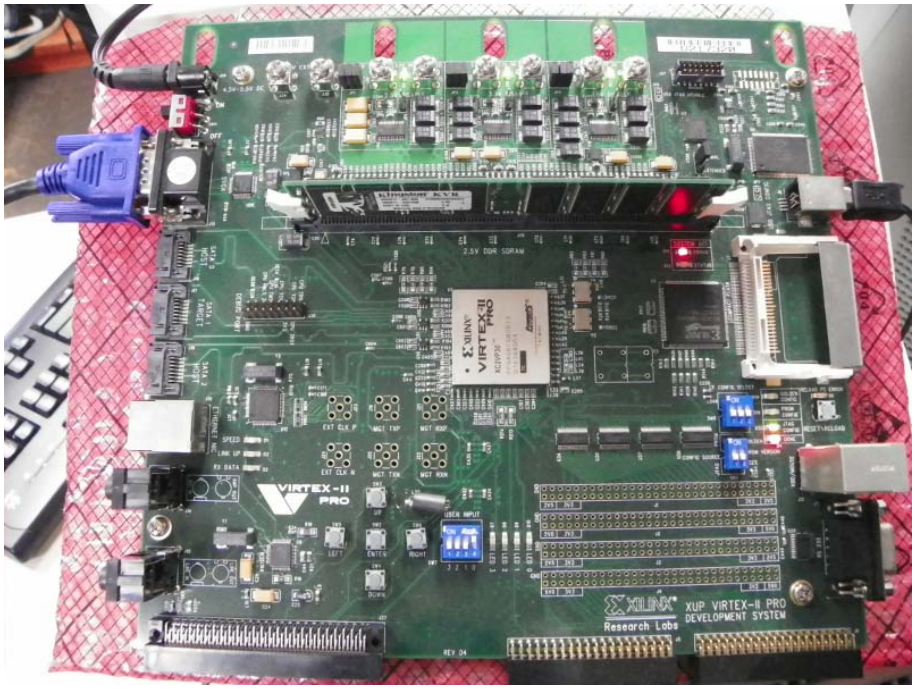

Figura 5.1: Placa de FPGA XC2VP30 utilizada nos testes do projeto

5.1 Síntese do Demultiplexador

A Tabela 5.1 apresenta os resultados de síntese do Demultiplexador desenvolvido, sendo que a frequência máxima de operação reportada foi de 221,870MHz. Este valor é quase noventa vezes superior ao mínimo necessário para o sistema. Notou-se uma certa otimização por parte do programa de síntese, pois alguns dos sinais que não eram utilizados, mas descritos em função da legibilidade do código, foram removidos. Isso reduziu a quantidade de *flip-flops* usados no circuito final.

Utilização Lógica	Usados	Disponíveis	Utilização
Número de Slices	89	13696	0%
Número de Slice Flip Flops	55	27392	0%
Número de 4 input LUTs	170	27392	0%
Número de bonded IOBs	33	556	5%
Número de GCLKs	1	16	6%
Frequência Máxima	-	221,870MHz	-

Tabela 5.1: Resultados da Síntese do Demultiplexador

5.2 Processador Gráfico

O Processador Gráfico foi subdividido em três blocos distintos: Filtro de Legendas, Interpretador de Legendas e Sobreposição. As Tabelas 5.2 até 5.4 apresentam os resultados das sínteses destes componentes.

O Filtro de Legendas gerou uma grande quantidade de *flip-flops* em função do gerenciamento dos três processos internos em *pipeline*. Os 20 sinais de entrada e saída correspondem a oito bits de entrada e oito de saída para dados, *clock*, *reset* e habilitador e sinalização de saída válida.

Utilização Lógica	Usados	Disponíveis	Utilização
Número de Slices	157	13696	1%
Número de Slice Flip Flops	108	27392	0%
Número de 4 input LUTs	304	27392	1%
Número de bonded IOBs	20	556	3%
Número de GCLKs	1	16	6%
Frequência Máxima	-	233,111MHz	-

Tabela 5.2: Resultados da Síntese do Filtro de Legendas

Utilização Lógica	Usados	Disponíveis	Utilização
Número de Slices	63	13696	0%
Número de Slice Flip Flops	34	27392	0%
Número de 4 input LUTs	117	27392	0%
Número de bonded IOBs	36	556	6%
Número de GCLKs	1	16	6%
Frequência Máxima	-	313,927MHz	-

Tabela 5.3: Resultados da Síntese do Interpretador de Legendas

O Componente de Sobreposição utilizaria a frequência de operação mais alta, já que o mesmo é vinculado à taxa de envio de *pixels* ao monitor, sendo um *pixel* por ciclo de relógio. Considerando 30 imagens de 1920x1080 pixels por segundo, teríamos a necessidade de verificar 62,208 milhões de *pixels* por segundo, limite bem inferior ao máximo suportado, mesmo levando em consideração os intervalos de sincronismo horizontal e vertical do monitor.

Utilização Lógica	Usados	Disponíveis	Utilização
Número de Slices	120	13696	0%
Número de Slice Flip Flops	70	27392	0%
Número de 4 input LUTs	218	27392	0%
Número de bonded IOBs	50	556	8%
Número de BRAMs	2	136	1%
Número de GCLKs	1	16	6%
Frequência Máxima	-	241,608MHz	-

Tabela 5.4: Resultados da Síntese do Componente de Sobreposição

5.3 Resultados Práticos

Embora o decodificador de vídeo seja um circuito grande demais para ser comportado pela placa de FPGA disponível e utilizada, o demultiplexador e o processador gráfico podem ter seus funcionamentos verificados. Então, será utilizado como fonte de vídeo um gerador de padrões, cujo tamanho físico é mais modesto. Há também o fato de o decodificador de vídeo não ser compreendido pelo escopo deste documento. Deste modo, o gerador de padrões substitui o acesso à memória DDR do TA, que armazena os macroblocos decodificados e que, depois, são transformados em linhas de *pixels* (LOP). Essas LOP são, então, enviadas para a saída de vídeo.

Para a realização dos testes práticos foi necessária a implementação de um componente gerador de fluxo de transporte que reproduz um bloco de dados carregado em um vetor para a entrada do demultiplexador. Este componente possui apenas um contador, que é incrementado a cada ciclo de relógio e é usado para acessar as posições do vetor de dados. A saída de dados é, então, o valor da posição do vetor endereçada pelo contador. Não é utilizada um fluxo em tempo real para os testes, pois a entrada de dados do TA como um todo seria gerenciada por um processador central e o circuito responsável pelas legendas terá o comportamento de um periférico.

Assim, apresenta-se o circuito tendo como entrada um fluxo de legendas extraído do canal 34 de Porto Alegre (Canal virtual 12.1, RBS TV POA), como exemplificação de funcionamento. Foi utilizada a placa de FPGA, mostrada na Figura 5.1, conectada ao monitor (Figura 5.2 - detalhe no canto superior esquerdo) interpretando o fluxo de dados descrito anteriormente. Ao fundo, nota-se uma sequência de quadrados coloridos, os quais são resultado da saída do gerador de padrões. A figura maior é uma aproximação com mais detalhes da região da tela alterada pelo Processador Gráfico.

A depuração do circuito foi realizada de forma iterativa. Cada iteração da depuração do circuito, que compreende sequências de troca do fluxo de transporte em teste, síntese e carregamento do *bitstream* para o FPGA, consumiu entre 10 e 30 minutos. Foram necessárias 30 iterações, devido a necessidade de correção da descrição em VHDL antes de se executar uma nova síntese, caso o comportamento do circuito se desviasse do comportamento correto. O comportamento correto é definido pela contenção de todos os símbolos em suas respectivas posições, com a utilização das informações de cor e transparência, sem o avanço sobre o espaço reservado aos símbolos adjacentes. As correções visavam principalmente a sincronização da ativação da janela do console de texto sobre o vídeo, sua posição e a apresentação dos símbolos dentro dessa janela.

Figura 5.2: Visão geral e área da tela utilizada para sobrepor as legendas

Isto foi necessário em função da quantidade de informação contida em uma única tela, cuja área real é maior do que a área visível, pois apresenta, também, espaço (traduzindo-se em tempo) para os sinais de sincronismo vertical e horizontal. Dessa forma, uma tela de 640×480 *pixels* utiliza uma grade de 800×520 *pixels*, um aumento de mais de 35%. Em função de limitações da placa de FPGA utilizada, que possui apenas saída de vídeo do tipo VGA, com conector D-SUB, não foi possível alcançar uma resolução FullHD de 1920×1080 *pixels*, situação que requer pelo menos um conector DVI e *hardware* específico para o seu controle.

Para os demais componentes, a depuração sobre os resultados das simulações já foram suficientes para que o circuito sintetizado se comportasse da forma desejada. Neste caso, isso foi possível de acordo com o formato dos dados de entrada, tanto para o fluxo de transporte, como para os fluxos específicos para as legendas, nos quais a observação das formas de ondas geradas pelas simulações continham informação relevante e sucinta para inspeções de forma manual do comportamento dos componentes. Estas inspeções manuais foram realizadas nos pontos em que o resultado gerado pelo componente simulado divergia do resultado obtido pelo protótipo. Dessa forma foi possível minimizar a interferência manual, automatizando o processo.

Em cada uma das inspeções fazia-se necessário verificar, primeiro os dois resultados divergentes e a entrada que os gerou. Após, com a dedução de qual versão do componente (*software* ou *hardware*) resultou em erro, analisa-se o comportamento deste componente para localização da transição de estado problemática. Durante a fase de depuração, os as entidades em VHDL ganharam uma porta extra para exteriorizar estados internos e acelerar a localização dos problemas através dos *testbenches* realizados, consumindo os arquivos com fluxos para teste gravados das emissoras e produzindo novos arquivos com o resultado da execução. Com a localização e correção do problema, realiza-se novas simulações e o processo de depuração é repetido até que os resultados simulados e gerados pelos protótipos sejam iguais.

6 CONCLUSÃO

Concluindo, pode-se observar através dos resultados obtidos que o sistema proposto tem capacidade de apresentar as legendas do tipo *closed caption* do Sistema Brasileiro de Televisão Digital de forma satisfatória, sendo um complemento ao sistema de Terminal de Acesso em desenvolvimento no laboratório LaPSI. Os valores obtidos de frequência máxima de operação são superiores às necessidades do Terminal de Acesso, sendo viável sua utilização.

Felizmente foi possível a implementação de todos os módulos propostos, com o tempo disponível, inclusive os testes de funcionamento. Das ferramentas mais importantes utilizadas, estão o simulador GHDL e a ferramenta de visualização GTKWave que aceleraram o desenvolvimento, além da geração dos vetores de teste e comparações.

A partir da experiência com o desenvolvimento de *hardware*, nota-se um grande distância entre escrever um algoritmo para ser executado em um processador e escrever um algoritmo para ser transformado em componente de *hardware*. O determinismo do software e a ordem das operações facilitam muito mais a maneira de pensar. O paralelismo do circuito final faz com que uma tarefa relativamente simples torne-se complexa demais, já que todos os componentes trabalham juntos e ao mesmo tempo, é necessário um controle muito maior.

REFERÊNCIAS

UTUG. **Página do grupo de usuários TEX da UFRGS**. Disponível em: <http://www.inf.ufrgs.br/utug>. Acesso em: novembro 2010.

IBGE (2001). ***Censo Demográfico***. 2000. Instituto Brasileiro de Geografia e Estatística.

TEIXEIRA, M. Lula inaugura TV digital em São Paulo. **ESTADÃO .COM.BR**. 2007. São Paulo, dezembro 2007. Disponível em <http://www.estadao.com.br/noticias/economia,lula-inaugura-tv-digital-em-sao-paulo,89399,0.htm>. Acesso em dez. 2010.

PATTERSON, J. T. **Video Decoder with Closed Caption Data on Video Output**. 2000. 18f. Patent 6,018,396 – United States Patent and Trademark Office, San Jose, CA.

PRATT, K. A.; YONKER, M. A.; XU, F. L. **System and Method for Displaying Closed Caption Data on a PC Monitor**. 1996. 7f. Patent 5,543,850 – United States Patent and Trademark Office, San Jose, CA.

WILLIAMS, M. C.; PRICE, T. W. **Method and Apparatus for Providing Real Time Data on a Viewing Screen Concurrently with any Programing in Process**. 1997. 20f. Patent 5,701,161 – United States Patent and Trademark Office, Indianapolis, IN.

TAKEUCHI, Y; ISHIBASHI, Y. **Computer System and Closed Caption Display Method**. 2001. 22f. Patent 6,297,797 – United States Patent and Trademark Office, Tokyo, JP.

SHINDO, H. et al. (1992). **Microcontrollers for closed caption system**. 2000. IEEE Transactions on Consumer Electronics, 38(3):268–273.

ARIB (2000). ***STD B-24 - Data Coding and Transmission Specification for Digital Broadcasting***, volume 1. Association of Radio Industries and Business (ARIB).

ABNT (2007). ***NBR 15606 - Televisão digital terrestre - Codificação de dados e especificações de transmissão para radiodifusão digital***. ABNT.

ISO/IEC (2000). ***ISO 13818 - Information technology - Generic coding of moving pictures and associated audio information***.

ANEXO

Processador Gráfico

Fabrício S. Caetano, Prof. Dr. Altamiro A. Susin (Orientador)

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

fscaetano@gmail.com

Abstract. *This paper describes the project of a graphic processor, closed-caption parser and demultiplexer for use in the Brazilian Digital Television System (SBTVD).*

Resumo. *Este artigo descreve o projeto de um processador gráfico, interpretador de closed-caption e demultiplexador para uso no Sistema Brasileiro de Televisão Digital(SBTVD).*

1. Introdução

1.1. Contexto

A Universidade Federal do Rio Grande do Sul - UFRGS, em parceria com algumas universidades de renome brasileiras desenvolvem o projeto de um Set-Top-Box com tecnologia nacional. No momento, o projeto engloba a recepção do sinal enviado pelas emissoras de acordo com o Sistema Brasileiro de Televisão Digital, demultiplexação dos fluxos de dados, decodificação do áudio e vídeo e apresentação. No entanto, o sistema não dispõe de recursos como interpretação de legendas e closed-caption, assim como demais recursos de sobreposição de informações ao vídeo e demultiplexação do fluxo de transporte proveniente do demodulador.

O desenvolvimento será realizado no Laboratório de Processamento de Sinais e Imagens – LaPSI – do Departamento de Engenharia Elétrica – DELET – da Escola de Engenharia da Universidade Federal do Rio Grande do Sul.

1.2. Motivação

A Norma Brasileira que define os padrões da televisão digital terrestre assinala como opcional o recurso de closed caption para os set-top-boxes, entretanto, segundo os resultados do Censo 2000 publicados pelo IBGE [IBGE 2001], 5,7 milhões de brasileiros tem alguma deficiência auditiva e, destes, 170 mil são declarados surdos. Tendo em mente o fato de que a televisão digital aberta deva ser um instrumento de inclusão social, segundo o próprio Presidente da República em declaração feita em 2 de dezembro de 2007, de acordo com o Jornal Estado de São Paulo [2], seria uma grande falha não desenvolver meios para que a parte da população com deficiência tenha como desfrutar com um mínimo de qualidade da inclusão social.

Ambientes barulhentos como restaurantes, aeroportos, bares também se beneficiariam do recurso de closed-caption. Assim, aquele que tem interesse na programação exibida teria como extrair boa parte da informação, mesmo com o volume baixo ou outro som ambiente.

1.3. Objetivos

De acordo com os blocos construtivos já implementados, pelos grupos de pesquisa e desenvolvimento das universidades anteriormente citadas e, ainda, com as necessidades do sistema como um todo, o objetivo principal deste trabalho é o projeto de um compositor gráfico.

O compositor gráfico é o componente de hardware responsável pela composição da sequência de imagens que serão exibidas pelo televisor. Estas imagens são compostas pela saída do decodificador de vídeo, o plano de legendas e demais símbolos gráficos referentes aos comandos dados pelo usuário através do controle remoto, ou ainda informações sobre o estado do receptor. É parte integrante do compositor gráfico, a interpretação do stream de closed-caption, caso exista.

1.4. Metas

As metas deste projeto foram alcançadas com sucesso, de modo que os objetivos foram concluídos ao final do período de planejamento.

1.5. Protótipos

Para testar os conceitos aqui apresentados em primeira instância, será construída uma versão em software do interpretador de closed-caption, que será usada no futuro como base para a descrição da versão final em VHDL ou outra linguagem de descrição de hardware.

1.6. Resultados Esperados

Espera-se que, ao final do projeto, se tenha um componente de hardware capaz de interpretar um stream elementar de closed-caption, se tenha como saída o plano de imagem com o texto e posicionamento corretos.

2. Estudo Bibliográfico

De acordo com a pesquisa de mercado realizada, há alguns trabalhos relacionados a este, que foram implementados. No entanto, nenhum deles aborda com detalhe o processamento dos dados do sistema de closed caption. Dentre estes trabalhos, apenas um aborda um sistema similar ao sistema no qual este trabalho é baseado.

2.1. Trabalho relacionado

Existe um trabalho chamado “MICROCONTROLLERS FOR CLOSED CAPTION SYSTEM” [Shindo et al. 1992] o qual descreve um sistema independente para ser usado entre a saída de um receptor e sintonizador do sinal de tv analógica e a entrada de vídeo componente de um televisor.

O sistema em questão é subdividido em duas partes: o Front End, também chamado de Data Slicer; o Sistema Microcontrolado. O Front End é o responsável por extrair do sinal de video o bitstream do sinal de closed caption, através da detecção de portadora, geração de sinal de clock e tem como características a detecção e ajuste automático à portadora do sinal de closed caption na linha 21. A responsabilidade de controlar a transformação do sinal de video é do Sistema Microcontrolado, que possui a geração de caracteres, funções de geração de menus.

Algumas das características do sistema são informadas, como a utilização de memória single port para reduzir o custo total do sistema, assim como área nas pastilhas de silício. Entretanto não há descrição do software que é executado.

3. Descrição da Informação

A informação gerada pelo demodulador, proveniente do sinal recebido, possui as características de um sistema de comunicação multicamadas, semelhante aos protocolos usados na internet. Esta abordagem facilita a transmissão audiovisual tanto pelas antenas de emissoras de TV, quanto seu roteamento em redes IP.

O fluxo de dados passa por diversas camadas de hardware e software desde que sai do demodulador. Cada camada é responsável por abstrair parte da complexidade do sistema de comunicação como um todo. A camada do Transport Stream é responsável por multiplexar todos os fluxos de dados. O Packetized Elementary Stream é responsável por criar pacotes concisos do fluxo elementar.

TS $\rightarrow$ PES $\rightarrow$ PES independente $\rightarrow$ ES

3.1. MPEG2-TS

No Sistema Brasileiro de Televisão Digital Terrestre, que é baseado no sistema ISDB-T, o Transport Stream (fluxo de transporte) utilizado é do tipo MPEG2-TS, padronizado pela norma ISO/IEC 13818, Parte 1. O fluxo de transporte é definido como uma sequência de pacotes de formato e tamanho fixo, desenvolvido para ser utilizado em meios sujeitos a erros e perdas de dados. A sintaxe do pacote de transporte é dada pela Tabela 1 padronizada pela ISO 13818 [ISO/IEC 2000].

Assim, pode-se determinar o sincronismo facilmente tomando-se um buffer com tamanho pelo menos duas vezes maior do que um pacote e procurar dois sync_bytes separados pelo tamanho do pacote. Através do campo de adaptação, obtém-se algumas as informações de temporização para sincronizar o sistema como um todo: a apresentação do vídeo, áudio e closed captions e demais informações. Muitas vezes o campo de adaptação é utilizado apenas para preencher o espaço do pacote, pois há poucos dados para transmitir naquele instante.

O identificador de pacote é chamado de PID. Cada fluxo de dados recebe um PID específico. São definidos alguns PIDs para o transporte de informações da emissora, tais como o nome da rede (NIT), lista de programas de um canal (PAT), tabelas de programas (PMT) com indicações de quais os PIDs (vídeo, áudios, closed-caption, dados) fazem parte de um programa.

No caso específico deste projeto, assim como no caso do vídeo e do áudio, o conteúdo do payload (carga) representado pela sequência de N data_bytes, será um PES. Em alguns casos, são recebidas tabelas: PAT, PMT, NIT, entre outras, as quais devem ser interpretadas para extrair os dados necessários ao funcionamento do demultiplexador.

O valor do campo sync_byte é sempre 0x47. A indicação de que o pacote pode estar corrompido e que o receptor não teve condições de recuperar com os algoritmos de correção de erro é feita setando o campo transport_error_indicator em ‘1’. O campo transport_priority, quando carrega o dado ‘1’, informa que o pacote deve ser tratado com prioridade. Para o sinal de televisão digital aberta, o campo transport_scrambling_control

Sintaxe	Núm. Bits
transport_packet() {	188×8
sync_byte	8
transport_error_indicator	1
payload_unit_start_indicator	1
transport_priority	1
PID	13
transport_scrambling_control	2
adaptation_field_control	2
continuity_counter	4
if(adaptation_field_control = '10' or adaptation_field_control = '11') {	
adaptation_field()	
}	
if(adaptation_field_control = '01' or adaptation_field_control = '11') {	
for(i=0; i<N; i++) {	
data_byte	
}	
}	
}	

Tabela 1. Sintaxe dos pacotes do MPEG2-TS, segundo ISO 13818-1

deve ser carregar '00' para indicar que os dados não são embaralhados, o que é obrigatório em uma transmissão aberta. A presença de um campo de adaptação e de uma carga de dados e dada pelo valor do campo adaptation_field_control. O bit mais significativo informa a presença do campo de adaptação e o bit menos significativo informa a presença de uma carga de dados. Finalmente, o campo continuity_counter traz a informação de um contador, que é usado pelo receptor para identificar perdas de pacotes. Cada PID possui um continuity_counter independente dos demais.

3.2. PES

PES é o fluxo de pacotes que contém o fluxo elementar. Um pacote PES pode se estender por vários pacotes TS, sendo informado pelo campo payload_unit_start_indicator do TS o início de um pacote PES ou sua continuação. O último pacote TS que forma um pacote PES apresenta um campo de adaptação de tamanho variável para preencher o espaço não utilizado. O fluxo de closed caption pode ser transmitido por qualquer tipo de PES, mas é preferido o uso de PES independente síncrono.

A sintaxe de um pacote PES é demasiadamente extensa para ser apresentada, contudo o cabeçalho simples é formado por três campos: packet_start_code_prefix de 24 bits (e não 6, como afirmada pela NBR 15602-3 2007), carregando o valor 0x000001; stream_id de 8 bits, que identifica o tipo de conteúdo que o pacote carrega e, finalmente PES_packet_length com 16 bits, que informa o tamanho do pacote PES. De acordo com o valor do stream_id, um cabeçalho estendido pode ser requerido.

Vídeo e áudio são exemplos de dados que requerem a extensão do cabeçalho PES. Esta extensão contém informações que cada pacote carrega são relacionados a taxa de dados, permissão de cópia, relógios para sincronização e apresentação, tamanho do

Sintaxe	Núm. Bits
[a]synchronous_PES_data() {	
data_identifier	8
private_stream_id	8
reserved_for_future_use	4
PES_data_packet_header_length	4
for(i=0; i <N1; i++) {	
PES_data_private_data_byte	8
}	
for(i=0; i <N2; i++) {	
[a]synchronous_PES_data_byte	8
}	
}	

Tabela 2. Sintaxe dos pacotes do PES independentes, segundo NBR 15606-3

cabeçalho estendido, entre outros dados que são reservados e obrigatoriamente devem ter seu valor com todos os bits em '1'.

No caso preferido pela norma, a carga de dados do pacote PES é um pacote PES independente síncrono e sua estrutura é relativamente simples, além de utilizar a versão mínima do cabeçalho.

3.3. PES Independente

Um pacote PES independente pode ser síncrono ou assíncrono. Apesar de serem diferentes, pois o síncrono é reconhecido pelo PES como private_stream_1 e o assíncrono como private_stream_2. A estrutura de ambos é idêntica e dada pela Tabela 2, de acordo com NBR15606 [ABNT 2007].

O campo data_identifier tem o valor 0x80 para os pacotes sincronos e 0x81 para os assíncronos. O campo private_stream_id não é utilizado e seu valor é 0xff. O campo PES_data_packet_header_length informa quantos bytes PES_data_private_data_byte deverão ser lidos antes de se chegar à carga de dados. Curiosamente, um pacote PES independente não carrega a informação da quantidade de bytes [a]synchronous_PES_data_byte.

3.4. Fluxo de Closed Caption

A norma brasileira é bastante sucinta em relação à sintaxe dos pacotes que carregam o fluxo de closed caption a ponto de não fornecer tal descrição. Entretanto, como é baseada na norma japonesa STD B-24 [ARIB 2000], tem-se uma quantidade satisfatória de informação.

A norma japonesa recomenda que dentro dos pacotes PES independentes sejam transmitidos pacotes chamados data_groups, de forma que seja possível transmitir em um único fluxo elementar múltiplas línguas e modos de apresentação. Cada data_group apresenta um campo denominado data_group_id, que, de acordo com seu valor, sendo 0x00 ou 0x20, determina que os dados recebidos pertencem à configuração e gerenciamento ou apresentação das diversas línguas de legendas disponíveis, até um total de 8 por programa,

Sintaxe	Núm. Bits
data_group() {	
data_group_id	6
data_group_version	2
data_group_link_number	8
last_data_group_link_number	8
data_group_size	16
for(i=0;i <N;i++) {	
data_group_data_byte	8
}	
CRC_16	16
}	

Tabela 3. Sintaxe dos pacotes data_group, segundo ARIB STD B-24

Sintaxe	Núm. Bits
caption_data() {	
TMD	2
Reserved	6
if(TMD=‘01’ or TMD=‘10’) {	
STM	36
Reserved	4
}	
data_unit_loop_length	24
for(i=0;i <N;i++) {	
data_unit()	8
}	
}	

Tabela 4. Sintaxe dos pacotes caption_data, segundo ARIB STD B-24

caso o valor esteja nas faixas 0x01 até 0x08 ou 0x21 até 0x28. A sintaxe dos data_groups é apresentada na Tabela 3.

Quando data_group_id tem o seu valor nas faixas 0x01 até 0x08 ou 0x21 até 0x28, o pacote formado pelos dados presentes no grupo data_group_data_byte pode ser interpretado pela descrição do caption_management_data. Pelo fato da norma não ser muito clara em relação a estas informações, apresentando indentações que sugerem a possibilidade de alguns campos serem opcionais, é necessária uma breve investigação experimental para familiarização com os dados de gerenciamento. Estes dados englobam relógio de apresentação, número de linguagens, formato e tamanho de tela em pixels, direção de escrita e codificação dos caracteres.

Quando o data_group não possui dados de gerenciamento, ele possui comandos e sua sintaxe é determinada pela estrutura caption_data apresentada na Tabela 5. O campo TMD informa o controle da temporização de exibição, STM informa o instante de tempo para início da exibição. O campo data_unit_loop_length informa a quantidade de bytes que devem ser lidos como data_unit().

Sintaxe	Núm. Bits
data_unit() {	
unit_separator	8
data_unit_parameter	8
data_unit_size	24
for(i=0;i <N;i++) {	
data_unit_data_byte	8
}	
}	

Tabela 5. Sintaxe dos pacotes data_unit, segundo ARIB STD B-24

A estrutura data_unit, finalmente, contém as informações que serão relevantes na construção da imagens que serão sobrepostas ao vídeo proveniente do decodificador de vídeo. O campo unit_separator sempre contém o valor 0x1f e este valor é reservado para separar as várias data_units em sequência. Data_unit_parameter, quando a estrutura carrega dados de closed caption, contem o valor 0x20. Outros valores são possíveis para representar sons, gráficos geométricos, mapas de cores e caracteres especiais DRCS. Data_unit_size contém a quantidade de bytes data_unit_data_bytes e, estes últimos, são a sequência de comandos closed caption que o processador gráfico tem interesse e necessita.

4. Descrição do Projeto

O projeto final será o compositor gráfico, composto pelo sistema necessário pela interpretação dos dados do closed-caption e geração da imagem que será sobreposta ao vídeo decodificado. A seguir, na Figura 1 é apresentada a visão geral da arquitetura do receptor e em cada componente será informado o local, nesta arquitetura, apropriado para sua localização.

4.1. Processador Gráfico

O processador gráfico consiste no componente de hardware responsável pela composição final da imagem que será exibida pelo televisor. Logicamente, este componente é separado dos demais, tendo apenas a responsabilidade de verificar a cor de um pixel específico da imagem de sobreposição e, de acordo com o valor da cor, decidir se repassa ao televisor a imagem de sobreposição ou a imagem vinda do decodificador de vídeo. É apresentado na Figura 1 como o bloco contendo a letra grega sigma.

Levando-se em consideração o hardware do receptor, armazenar uma imagem extra inteira comprometeria o projeto, aumentando suas necessidades de largura de banda para o acesso à memória, inclusive aumentando a quantidade de memória necessária no receptor. Assim, decidiu-se gerar apenas uma fatia horizontal da imagem de sobreposição à medida que o decodificador envia o vídeo para a saída.

Foi desenvolvido um protótipo em software que realiza as funções deste projeto, que pode ser visto na Figura 2. O programa recebe um fluxo MPEG2-TS via TCP/IP, demultiplexando os pacotes, interpretando a tabela PAT para identificação dos programas, escolhendo a tabela PMT contendo o programa 1SEG e extraindo os PID dos fluxos de

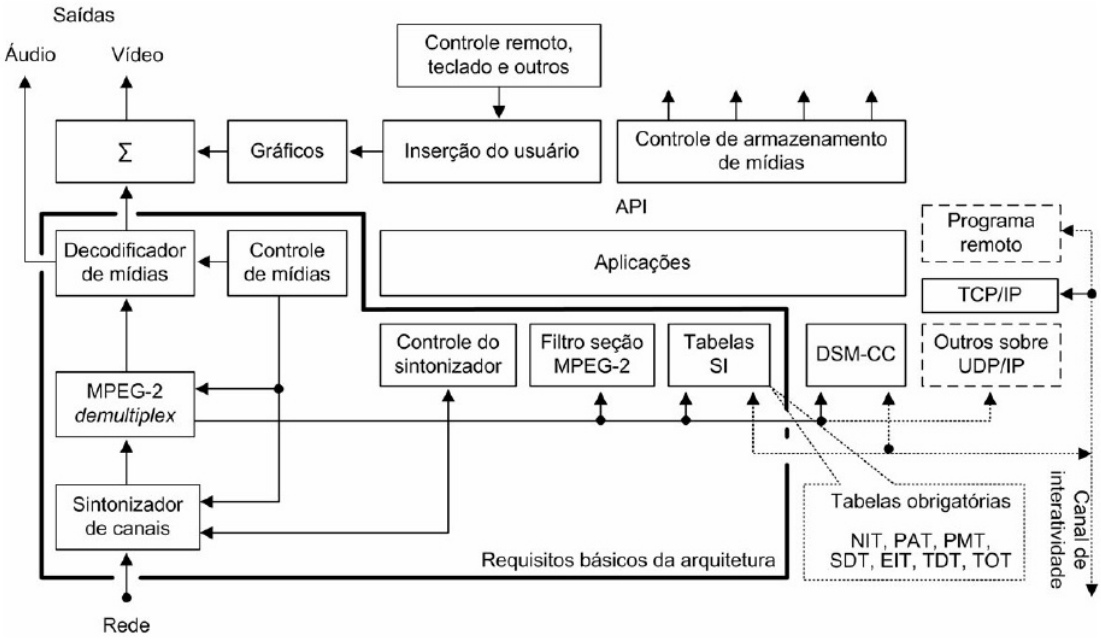

Figura 1. Arquitetura básica do receptor, segundo NBR 15604

vídeo, áudio e closed caption. O fluxo de vídeo e closed caption são repassados para o decodificador H.264 e interpretador de closed caption. Neste último é gerada a imagem completa para sobreposição.

4.2. Demultiplexador

O demultiplexador é responsável pela seleção e encaminhamento dos pacotes de estrada do stream recebido do demodulador, que são do tipo MPEG2-TS descritos no Capítulo 3.1 e apresentado na Figura 1 como MPEG2-2 demultiplex.

O processo de demultiplexação pode ser realizado de várias formas, duas delas serão abordadas a seguir.

4.2.1. Protótipo

A versão mais simples, que possui um custo elevado em memória, usada como protótipo para verificar o funcionamento dos componentes do sistema, dá-se pela leitura de blocos de dados de um fluxo de transporte. Sincroniza-se o início dos pacotes através da busca de dois bytes de sincronização afastados de 188 bytes, quando esta condição é satisfeita, dá-se início à leitura de blocos de 188 bytes, sendo o primeiro byte o de sincronização e seu índice é zero.

Após a sincronização, são interpretados os cabeçalhos MPEG2-TS e extraídos os identificadores de pacote (PID). Se o PID tem o valor esperado para um pacote de vídeo, áudio, closed caption ou o valor de uma tabela PAT ou PMT, o restante do pacote é analizado para se encontrar o início dos dados que o demultiplexador deve fornecer aos demais decodificadores e interpretadores do sistema.

A princípio, considera-se o início dos dados como sendo o de índice quatro.

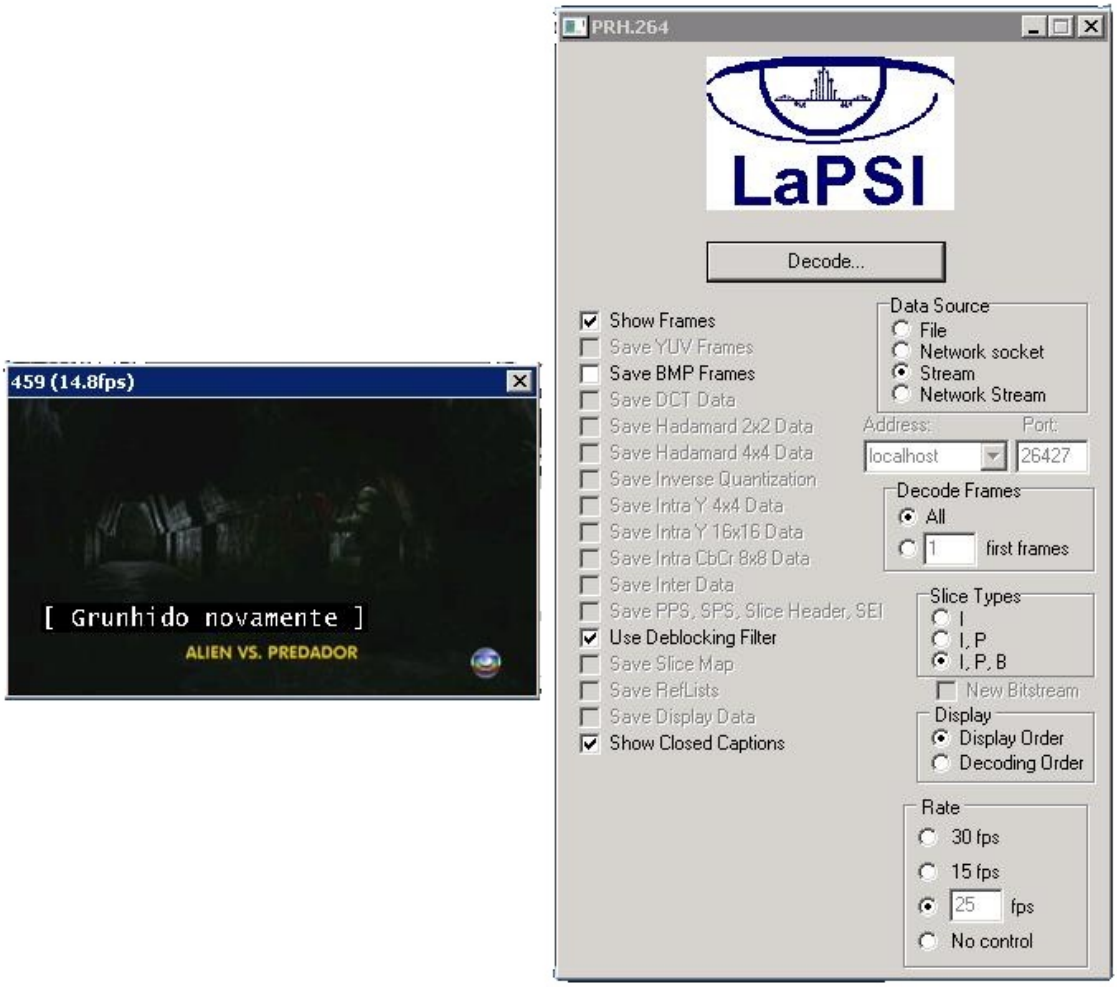

Figura 2. Programa PRH264, desenvolvido no LaPSI com protótipo de closed caption e sobreposição

No caso de um pacote conter o campo de adaptação, informado pelo campo adaptation_field_control, seu tamanho é extraído do primeiro byte do campo de adaptação. Assim, tem-se o início dos dados no índice cinco somado do tamanho do campo de adaptação.

Se o campo payload_unit_start_indicator for igual a '1', são verificados os primeiros três bytes do bloco de dados e caso seu valor seja 0x000001, admite-se que é o início de um cabeçalho PES. Caso contrário, são enviados para a saída apropriada desde o primeiro destes três bytes até o final do pacote MPEG2-TS, pois trata-se de dados ou tabelas.

Identificado o início de um cabeçalho PES, o byte seguinte contém o valor do identificador de fluxo (stream_id). De acordo com o valor deste identificador, é possível descobrir se o cabeçalho PES é estendido ou não. Com o cabeçalho estendido, é necessário analizar o byte de índice 8 do cabeçalho PES, no qual é especificado o tamanho do restante deste cabeçalho.

No caso do pacote conter os dados do sistema de closed caption, o início dos dados

coincide com o início do cabeçalho PES Independente Síncrono. Os quatro primeiros bits do terceiro byte deste cabeçalho informa a quantidade de bytes privados. Os dados que o demultiplexador deve repassar para o interpretador de closed captions encontram-se nos synchronous_PES_data_bytes.

4.2.2. Máquina de estados finitos

O processo de demultiplexação mais econômico é caracterizado por uma máquina de estados que tem como entrada um byte do fluxo de transporte e várias saídas possíveis, orientadas a byte. Além das entradas que define o componente como uma máquina de estados finitos, são necessárias entradas para parâmetros, já que o demultiplexador não tem como saber a priori o valor exato dos PIDs dos pacotes de áudio, vídeo, closed caption, tabelas com PID informados na PAT, entre outros. Uma das saídas é conectada ao decodificador de vídeo, fornecendo um fluxo elementar de vídeo H.264 consistido por uma sequência de NALs. As outras saídas são conectadas ao decodificador de áudio, ao interpretador de closed captions, interpretadores de tabelas e demais fluxos de dados.

Cada estado desta máquina consome um byte da entrada e estão separados em grupos diferentes: interpretação do cabeçalhos MPEG2-TS, PES, PES Independente e repasse dos dados para as saídas vinculadas aos PIDs dados como parâmetros. São um total de 20 estados, sendo que a vários apenas consomem dados e avançam para o estado seguinte, outros acumulam ou montam os valores dos campos que informam os tamanhos dos cabeçalhos.

4.3. Interpretador de Closed Caption

O interpretador de closed caption é o componente de hardware que gera a imagem para sobreposição em tempo real. Ele é reponsável por receber um fluxo de data_groups e gerar a fatia da imagem de sobreposição que será usada pelo Processador Gráfico. Na arquitetura básica do receptor, este interpretador é englobado pelo bloco Decodificador de mídias.

Entre a recepção dos data_groups e a geração da imagem, há ainda o processo de abertura dos pacotes para remoção das informações sem valor para a tarefa, como cabeçalhos intermediários e closed captions de línguas não selecionadas. Assim, dos data_groups extrai-se os caption_management_data para configuração do console do módulo gerador de imagens e, após, os caption_data para extração dos data_units. Dos data_units extrai-se os data_unit_data_byte que finalmente serão interpretados para geração da imagem.

A partir do ponto no qual se obtém um fluxo de data_unit_data_byte, o mesmo é utilizado como um fluxo de instruções para um processador de 8 bits. As intruções de valor nas faixas 0x00 a 0x1f e 0x80 a 0x9f, ressaltadas na Tabela 6, são mais complexas e serão detalhadas a seguir. As demais instruções apenas copiam um valor, o próprio, para a posição corrente da área de console. A Tabela 7 apresenta os caracteres usados quando o processador está no modo shift.

Das intruções complexas, tem-se o avanço da posição atual nas direções acima, abaixo, esquerda, direita e nova linha, respectivamente com as seguintes intruções APU,

APD, APB, APF e APR. Existe também a opção de avanço parametrizado com PAPF e a definição da linha e da coluna do ponto corrente com APS tendo dois bytes como parâmetros.

A instrução CS atribui a todos os caracteres do console o caracter espaço com fundo transparente, realizando a operação de limpar a tela. O modo shift é acionado com as instruções LS0, LS1, SS2 e SS3, sendo que LS0(1) tem a função de ligar e desligar o modo shift, enquanto SS2(3) habilitam o modo shift apenas para o próximo caracter escrito.

A instrução CAN tem como objetivo preencher a linha até o final com o caracter espaço, sendo que o fundo é da cor de fundo e não transparente. RS e US são reservados como separadores de informação.

A coluna de instruções 8x de 0x80 a 0x87 definem a cor do texto como sendo preto, vermelho, verde, amarelo, azul, magenta, ciano e branco. SSZ, MSZ e NSZ definem o temanho do corpo dos caracteres exibidos como pequeno, médio e normal. O tamanho pequeno faz com que os caracteres tenham metade da altura e metade da altura. O tamanho médio faz com que os caracteres apenas tenham metade da altura. SZX é usado para definir tamanhos especiais e possui um byte de parâmetro.

	0x	1x	2x	3x	4x	5x	6x	7x	8x	9x	Ax	Bx	Cx	Dx	Ex	Fx
x0	NUL		SP	0	@	P	`	p	BKF	COL	10/0	°	À	Ð	à	õ
x1			!	1	A	Q	a	q	RDF	FLC	¡	±	Á	Ñ	á	ñ
x2			"	2	B	R	b	r	GRF	CDC	¢	²	Â	Ò	â	ò
x3			#	3	C	S	c	s	YLF	POL	£	³	Ã	Ó	ã	ó
x4			$	4	D	T	d	t	BLF	WMM	€	Ž	Ä	Ô	ä	ô
x5			%	5	E	U	e	u	MGF	MACRO	¥	µ	Å	Õ	å	õ
x6		PAPF	&	6	F	V	f	v	CNF		Š	¶	Æ	Ö	æ	ö
x7	BEL		'	7	G	W	g	w	WHF	HLC	§	·	Ç	×	ç	÷
x8	APB	CAN	(	8	H	X	h	x	SSZ	RPC	š	ž	È	Ø	è	ø
x9	APF	SS2	)	9	I	Y	i	y	MSZ	SPL	©	¹	É	Ù	é	ù
Xá	APD		*	:	J	Z	j	z	NSZ	STL	ª	º	Ê	Ú	ê	ú
xB	APU	ESC	+	;	K	[	k	{	SZX	CSI	«	»	Ë	Û	ë	û
xC	CS	APS	,	<	L	\	l	|			¬	Œ	Ì	Ü	ì	ü
xD	APR	SS3	-	=	M	]	m	}		TIME	ÿ	œ	Í	Ý	í	ý
xE	LS1	RS	.	>	N	^	n	~			®	Ÿ	Î	Þ	î	þ
xF	LS0	US	/	?	O	_	o	DEL			¯	¿	Ï	ß	ï	15/15

Tabela 6. Conjunto de caracteres latinos, segundo NBR 15606-1

	0x	1x	2x	3x	4x	5x	6x	7x
x0				¤	…			
x1			♪	¦	▉			
x2				¨	‘			
x3				´	’			
x4				¸	“			
x5				¼	”			
x6				½	•			
x7				¾	™			
x8					⅛			
x9					⅜			
xA					⅝			
xB					⅞			
xC								
xD								
xE								
xF								

Tabela 7. Conjunto de caracteres especiais como G3, segundo NBR 15606-1

Referências

ABNT (2007). *NBR 15606 - Televisão digital terrestre - Codificação de dados e especificações de transmissão para radiodifusão digital*. ABNT.

ARIB (2000). *STD B-24 - Data Coding and Transmission Specification for Digital Broadcasting*, volume 1. Association of Radio Industries and Business (ARIB).

IBGE (2001). *Censo Demográfico 2000*. Instituto Brasileiro de Geografia e Estatística.

ISO/IEC (2000). *ISO 13818 - Information technology - Generic coding of moving pictures and associated audio information*.

Shindo, H. et al. (1992). Microcontrollers for closed caption system. *IEEE Transactions on Consumer Electronics*, 38(3):268–273.