FABIO CORREIA KUNG
KENZO MARCELO OKAMURA
RAFAEL ANICET ZANINI

USBTV:
A TV Digital no seu computador

São Paulo
2007

FABIO CORREIA KUNG
KENZO MARCELO OKAMURA
RAFAEL ANICET ZANINI

USBTV:
A TV Digital no seu computador

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Bacharel em Engenharia.

Área de Concentração:
Engenharia de Computação

São Paulo
2007

FABIO CORREIA KUNG
KENZO MARCELO OKAMURA
RAFAEL ANICET ZANINI

USBTV:
A TV Digital no seu computador

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Bacharel em Engenharia.

Área de Concentração:
Engenharia de Computação

Orientador: Prof. Dr. Moacyr Martucci Junior

São Paulo
2007

Às nossas famílias e companheiras, pelo apoio e compreensão em mais uma etapa de nossas vidas. Aos mestres, pelo conhecimento e pela inspiração em nossos caminhos.

"Pequenas oportunidades são muitas vezes o começo de grandes empreendimentos."

Demóstenes

Político e pensador grego

(384 a.C. - 322 a.C)

Resumo

A TV digital desempenhará papel importante no processo de inclusão digital previsto pelo Governo Brasileiro. Com o intuito de contribuir para o desenvolvimento da mesma no país, este trabalho apresenta a especificação e implementação de um dispositivo de captura de TV digital via USB. São detalhados os requisitos técnicos e escolhas de projeto realizadas para a implementação do dispositivo, que foi baseado em um kit de desenvolvimento USB com uma FPGA como unidade de tratamento e processamento de sinais. O dispositivo opera em diferentes sistemas operacionais, uma vez que a portabilidade é um dos objetivos deste trabalho. Os resultados são apresentados através de aplicações que realizam a reprodução dos vídeos no computador e também sua gravação em dispositivos de armazenamento. O trabalho aborda também a integração do dispositivo ao padrão brasileiro de TV Digital, além de analisar e caracterizar os diferentes tipos de aplicações e níveis de interatividade possíveis de serem utilizados com o middleware brasileiro Ginga. Por fim, são discutidas e propostas algumas aplicações interativas que utilizam a conexão normal à Internet como canal de interatividade, expandindo assim a usabilidade dos serviços disponíveis na Internet para as aplicações de TV digital, agregando maior valor às aplicações interativas possíveis de serem desenvolvidas.

Palavras-chave: Engenharia. Engenharia da Computação. TV Digital. USB. Canal de Retorno.

Abstract

The Digital TV has appeared as the new paradigm for interaction between viewers and TV shows. Its development in the country also represents the great effort the Brazilian government has shown for digital inclusion. The aim of this project is to specify and develop a portable device for Digital TV signal capturing through the USB protocol. The technical requirements and project specifications are detailed through this paper, which includes the complete description of the necessary steps for developing the device. It is based on an USB development kit, which counts on an FPGA for dealing and processing of the digital signals. The device can be used on several operating systems, which guarantees the portability desired for this project. The main results are shown through example applications that use the device as main source for displaying and recording video streams, delivered by a transport stream generator device. This paper also describes the use of the device with the Brazilian Digital TV standard (SBTVD), besides analyzing different kinds of interactivity possible to exist on Digital TV applications running over the Brazilian middleware Ginga. In conclusion, this paper presents and discusses new applications that may use an existing connection to the Internet as the return channel, bringing the services already available on the Internet to the Digital TV world and expanding their functional capabilities and visual appeal.

Keywords: Engineering. Computer Engineering. Digital TV. USB. Return Channel.

Lista de Figuras

Sumário

Capítulo 1

INTRODUÇÃO

1.1 Objetivo

O objetivo principal deste projeto é criar um dispositivo que possibilite a captura de sinais de vídeo digital pelo computador através da interface USB e permita assistir à programação de forma interativa através do envio de dados por um canal de retorno implementado por uma conexão comum à Internet, dando suporte aos padrões brasileiros de TV digital e incentivando o desenvolvimento de novas tecnologias puramente nacionais.

1.2 Motivação

A TV Digital desempenhará em breve um dos principais papéis na difusão de informação, conteúdo e conhecimento, além de entretenimento. A convergência tecnológica percebida nos dias atuais cria demanda para a integração entre esse mundo de conteúdo digital e o principal instrumento de manipulação de informação: o computador. Neste contexto, tornam-se necessárias alternativas para a captura do sinal de TV Digital pelo computador, tanto como forma de facilitar a inclusão dos telespectadores neste novo sistema de TV, quanto melhorar a experiência de usuário. O perfil brasileiro dá preferência a alternativas simples e baratas, que acabam tendo um maior alcance.

Uma vez que o computador oferece uma ótima plataforma para desenvolvimento, o projeto também possibilita que aplicações possam ser construídas oferecendo a infraestrutura e a interatividade para que se beneficiem do sinal de TV Digital na sua totalidade.

Embora a interatividade seja uma das metas da implantação da TV Digital no Brasil, o funcionamento do canal de retorno de dados, essencial para a sua existência - pois é por esse canal que trafegam os dados do usuário - ainda não está adequadamente adaptado para o público brasileiro. Na Europa utiliza-se a transmissão por GSM, que em princípio não tem apelo comercial por tornar mais caro quanto maior a sua utilização. De fato, poucas pessoas utilizam a rede celular como forma de transmissão de dados. A transmissão dos dados pela Internet já está prevista em todos os padrões de TV Digital, porém muito pouco explorada. Deste modo, a criação de uma estrutura alternativa para a implementação deste canal de retorno é altamente justificável, e para tal, a proposta deste

projeto é o desenvolvimento desta através de uma conexão comum à Internet, seja ela por linha discada, banda larga, ou comunicação wireless (Wi-Fi ou WiMax).

Disponibilizar o sinal de TV Digital para o computador abre inúmeras portas para aplicações, manipulação e tratamento deste. Como alguns exemplos podem ser citados a gravação agendada de programas, a conversão para diversos formatos de vídeo (WMV, DivX, XVid, Mov, entre outros), a transmissão do vídeo on-line - via Internet, armazenamento dada a crescente capacidade dos discos rígidos em computadores pessoais, facilidade para criação de multimídia (CDs, DVDs). É importante novamente ressaltar que dispor o sinal de TV Digital aos computadores facilita a criação de plataformas de desenvolvimento para conteúdo interativo destinado a TV Digital.

1.3 Escopo

O projeto teve início em Janeiro de 2007 e a idéia inicial envolvia a captura de sinais analógicos de TV aberta através de uma interface USB, para a visualização da programação comum de TV no computador. O projeto evoluiu após a identificação da crescente demanda por tecnologia envolvendo TV Digital. Assim, o foco do projeto foi redirecionado para a implementação do canal de interatividade pela Internet. Este projeto desencadeou uma pesquisa profunda sobre as tecnologias atualmente existentes e toda a polêmica sobre o assunto no Brasil.

O grupo percebeu que o assunto tomava volume ao longo do ano, e que diversas equipes espalhadas pelo país estavam apostando em pesquisa e desenvolvimento técnico na área. A oportunidade de contribuir para a especificação do padrão brasileiro também incentivou o grupo a continuar o seu trabalho e a envolver-se cada vez mais nas pesquisas de TV Digital.

Com o projeto já direcionado à TV Digital, foi prevista a implementação de um receptor, sintonizador e decodificador de sinal de TV Digital para a sua captura através do dispositivo USB. Devido ao tempo e recursos limitados para a realização do projeto, houve uma redução nas características funcionais.

O projeto é caracterizado por dois grandes blocos. O primeiro deles trata mais do hardware e provê um dispositivo USB responsável pela captura do fluxo de vídeo, som e dados multiplexados no padrão MPEG-2 Transport Stream. O MPEG-2 é a tecnologia de codificação do vídeo adotada pelos três principais padrões de TV Digital: o DVB (europeu), o ATSC (americano) e o ISDB (japonês). O padrão brasileiro, conhecido como Sistema Brasileiro de TV Digital (SBTVD), definiu a adoção de uma tecnologia mais atual para a

codificação do vídeo, o MPEG-4, que é totalmente compatível com o MPEG-2 e considerado a sua evolução já que permite maior compressão. Para o transporte dos dados, o padrão adotado no Brasil continua sendo o MPEG2 Transport Stream. Por basear-se no padrão MPEG-2 Transport Stream, o projeto torna-se independente do padrão de transmissão utilizado.

O segundo bloco está relacionado ao software necessário, responsável pela leitura do fluxo MPEG-2 através da interface USB, pela apresentação do vídeo e som e pela execução de aplicativos para a TV Digital. O ambiente de execução de aplicativos é conhecido como *middleware* e deverá possibilitar a implementação de um canal de retorno através da conexão disponível com a Internet, seja ela uma conexão discada, banda larga, sem fio ou qualquer outra.

O foco do projeto é a pesquisa e desenvolvimento de uma infra-estrutura completa para a implementação do *middleware* que possibilite o canal de retorno através da Internet. Isto inclui as funcionalidades do lado cliente e do lado servidor, além da definição de padrões, estrutura e protocolos de comunicação utilizados. Este *middleware* também possibilita a criação de um poderoso ambiente de desenvolvimento, já que aplicações poderão ser desenvolvidas e testadas na mesma plataforma – o computador.

É importante enfatizar a possibilidade de expansão das funcionalidades, uma vez que o projeto visa ser independente de tecnologias de transmissão, recepção e decodificação do sinal de TV Digital. Desta forma, tantos os padrões europeus, americanos, japoneses e brasileiros poderão tirar proveito dos resultados obtidos neste trabalho.

1.4 Organização

O presente documento está dividido nas seguintes seções:

- Capítulo 2 - Aspectos Conceituais: são definidos os conceitos básicos a serem discutidos ao longo de todo o documento, o cenário atual da TV Digital no país, o estado da arte da tecnologia existente e os equipamentos utilizados durante a concepção do projeto;

- Capítulo 3 - Especificação do Projeto: nesta seção são definidos todos os requisitos necessários para o projeto, bem como o impacto destes na definição do escopo do projeto. São levantadas as características técnicas dos equipamentos e padrões que serão estudados e utilizados ao longo do projeto;

- Capítulo 4 - Metodologia: este capítulo descreve os principais métodos e técnicas utilizadas para a implementação do projeto baseado nos requisitos levantados previamente;
- Capítulo 5 - Projeto e Implementação: este capítulo descreve todas as atividades, especificações técnicas e componentes de hardware e software que foram desenvolvidos para a implementação do projeto;
- Capítulo 6 - Testes e Avaliação: este capítulo descreve os principais testes executados para a validação das funcionalidades do dispositivo USB e do software implementado, bem como a avaliação dos resultados obtidos;
- Capítulo 7 - Considerações finais: este capítulo avalia os resultados globais do projeto, analisando as dificuldades e aspectos positivos do projeto, além de ressaltar os projetos futuros que poderão se basear no presente projeto;
- Apêndice A - API FrontPanel: este apêndice relata o uso da API do kit de desenvolvimento, bem como apresenta um overview da API utilizada;
- Apêndice B - Pinagem: este apêndice relata a configuração de pinos e conexões utilizada ao longo da implementação do dispositivo USB;

Capítulo 2

ASPECTOS CONCEITUAIS

2.1 O Ambiente

2.1.1 Cenário

Iniciado em 2003 e coordenado pelo Centro de Pesquisa e Desenvolvimento em Telecomunicações (CPqD), o comitê do Sistema Brasileiro de Televisão Digital (SBTVD) foi responsável pelos estudos que definiriam o padrão a ser adotado no país. Após estudos conduzidos juntamente com universidades e companhias de comunicação, em 13 de novembro de 2005 o sistema foi apresentado pelo presente Ministro das Comunicações Hélio Costa. O sistema resultante dos estudos foi baseado no sistema ISDB-T utilizado no Japão, porém com a adição de padrões desenvolvidos no país que deveriam proporcionar alta definição e interatividade segundo as diretivas do governo.

A definição do padrão brasileiro de TV Digital e o forte compromisso do governo e de instituições privadas levam a crer que a TV Digital, incluindo o serviço de interatividade, terá ampla penetração no Brasil. Uma vez instaurados, os serviços de interatividade estarão vinculados a programas de televisão e a publicidade, exigindo assim demanda constante por produção. A eficiência deste processo será fator crítico para a redução de custos de produção e impactará na margem de lucro das produtoras e emissoras.

Todas as expectativas são de forte adoção da TV Digital pelos espectadores (conforme estudo realizado pelo CPqD [1]). A penetração da tecnologia para 2016 está estimada em 40% dos domicílios brasileiros atendidos para o pior caso e 70% de domicílios atendidos no melhor caso. Esta estimativa nem leva em consideração eventos importantes, como as Olimpíadas de 2008 em Pequim, as Eleições de 2010 e principalmente a Copa do Mundo de 2010 na África. Tais eventos têm grande chance de alavancar ainda mais a adoção, já que historicamente sempre foram um incentivo às vendas de televisores (que podem já integrar o set-top box) e ao aumento da demanda por publicidade e propaganda, que certamente serão um atrativo à interatividade. Há ainda a questão do uso da propaganda, que já está sendo utilizada como estratégia de implantação pelo governo e que promete alavancar a difusão da TV Digital trazendo a estimativa (no melhor caso) para 80% dos domicílios atendidos já em 2010.

Além de o cenário ser bastante promissor, estudos mostram que a influência dos novos serviços oferecidos TV Digital para as emissoras e produtoras de conteúdo (principais clientes do produto proposto) é sempre positiva. Especificamente, a interatividade influi positivamente sobre as receitas das emissoras e produtoras de conteúdo.

A TV Digital ainda não é uma realidade no Brasil, uma vez que só existe para assinantes de TV por assinatura. Além disso, a tecnologia utilizada pelas operadoras de TV a cabo é estrangeira, baseada nos padrões europeus de TV Digital (DVB e MHP). O projeto de TV Digital brasileiro incentiva a pesquisa de diferentes setores, tanto no meio acadêmico como no meio privado. Estas pesquisas e incentivos do governo brasileiro permitem a definição de um padrão brasileiro para transmissão e *middleware* para a execução de aplicações sobre conteúdo multimídia.

2.1.2 Plano de Transição

O governo elaborou também um plano de transição que deverá ser cumprido tanto por emissoras comerciais como públicas a fim de organizar e estabelecer um prazo para que a tecnologia de TV Digital seja implantada e esteja disponível para todos os brasileiros.

A Figura 2.1 contém um quadro onde são apresentadas as etapas do cronograma de digitalização para esse plano de transição.

O estado de São Paulo já possui a infra-estrutura e tem início das transmissões do novo padrão marcado para dezembro de 2007. Ao final do processo de transição, em 2013, todo o país terá acesso a TV Digital. Regulamentado pelo governo e também em conseqüência das perspectivas futuras de mercado, fabricantes de televisão deverão entrar em conformidade com o novo padrão sendo obrigados, também a partir de 2013, a produzir aparelhos seguindo os padrões da TV Digital brasileira.

Até atingir a conformidade com o padrão, o país pretende continuar transmitindo ambos os sinais analógicos e digitais, o chamado *simulcasting*, pelo menos até 2016. Se houver necessidade, o governo prorrogará o prazo de transição.

2.1.3 Terminal de Acesso (*set-top box*)

O usuário deverá possuir um terminal de acesso que será responsável pela recepção e decodificação do sinal para o aparelho televisor. Este equipamento, chamado *set-top box*, inicialmente será vendido como um aparelho externo; os consumidores poderão comprar e utilizar em seus televisores analógicos ou digitais. Porém, a tendência é que, com o tempo, o *set-top box* virá integrado aos novos aparelhos de televisão já adaptados a tecnologia digital.

Fases	Regiões	Emissoras	Calendário
1	As duas maiores Regiões Metropolitanas (RJ e SP)	5 maiores emissoras comerciais e a maior emissora pública (e/ou operador(es) de rede)	T_0 + 6 meses
2	RMs com mais de 2 milhões de habitantes e Brasília	5 maiores emissoras comerciais e a maior emissora pública (e/ou operador(es) de rede)	T_0 + 12 meses
3	Demais RMs, capitais e cidades com mais de 300 mil hab.	5 maiores emissoras comerciais e a maior emissora pública (e/ou operador(es) de rede)	T_0 + 24 meses
4	Cidades com mais de 100 mil habitantes	5 maiores emissoras comerciais e a maior emissora pública (e/ou operador(es) de rede)	T_0 + 36 meses
5	Cidades com mais de 100 mil habitantes	Todas as geradoras e retransmissoras (e/ou operador(es) de rede)	T_0 + 48 meses
6	Todo o país	Todas as geradoras e retransmissoras (e/ou operador(es) de rede)	T_0 + 72 meses

RM - Região Metropolitana T_0 - instante em que as primeiras freqüências são consignadas

Figura 2.1: *Quadro resumo para o plano de transição da TV Digital no Brasil*

Muito se tem especulado sobre o preço do *set-top box*. Governo e indústria divergem nos valores, porém este deve variar entre R$ 200 e R$ 800. Nos gráficos da Figura 2.2 e da Figura 2.3 observa-se o estudo feito pelo CPqD sobre o comportamento de alguns indicadores à variação do preço inicial do terminal de acesso à TV Digital.

Com relação à evolução dos preços, tem-se que quanto maior o preço inicial, maior o preço médio ao longo do período, conforme ilustrado na Figura 2.2. Em todos os casos, o preço cai mais intensamente nos primeiros anos, estabilizando por volta de oito a dez anos após o início da comercialização do terminal de acesso.

Preços mais elevados implicam em um percentual menor de domicílios atendidos, como é mostrado no gráfico da Figura 2.3. O estudo mostra que após 15 anos e com as premissas consideradas, o percentual de domicílios atendidos no caso mais favorável (a R$ 200) chega a 81%, enquanto que no caso mais desfavorável (a R$ 800) esse percentual fica em torno de 52%. É importante lembrar que, para efeito de simulação, esses valores foram divididos em 10 parcelas, segundo o procedimento de venda a prazo.

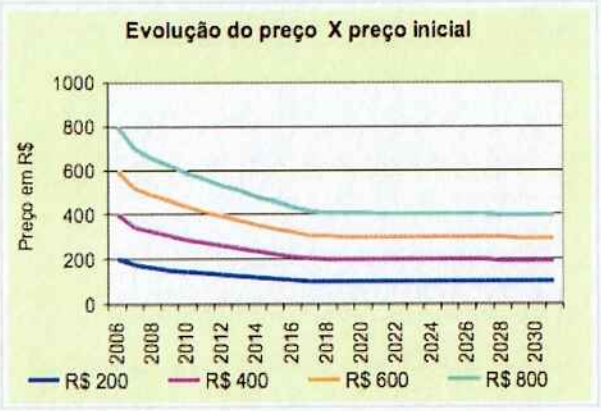

Figura 2.2: *Sensibilidade da evolução do preço ao preço inicial*

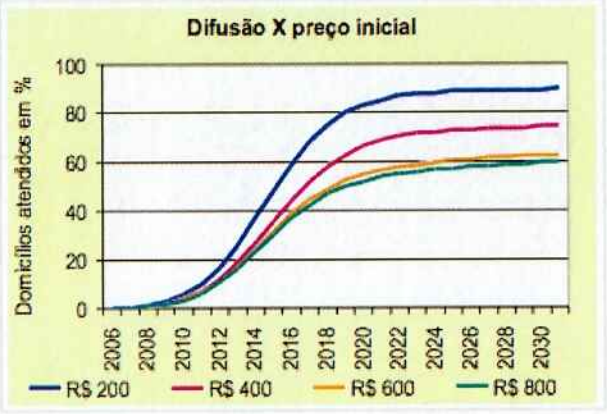

Figura 2.3: *Sensibilidade da difusão segundo o preço inicial do terminal de acesso*

Pode-se concluir que a longo prazo, mesmo com o preço elevado do terminal de acesso, a difusão da TV Digital será ampla sobre o território brasileiro e substituirá a TV analógica convencional.

2.1.4 Modelo de Serviços

Entende-se por modelo de serviços como o leque de serviços e aplicações de que um sistema de TV digital terrestre pode dispor. Para que tais serviços cheguem ao usuário é necessário que não apenas das tecnologias habilitadoras e de toda infra-estrutura de serviços subjacentes mas também do perfil de demanda dos usuários e da estratégia e capacidade de atendimento dos provedores de serviço (emissoras/programadores e outros agentes associados).

O modelo de serviços é, portanto, um reflexo do quanto será explorado das potencialidades funcionais da TV Digital, ressaltando-se a importância do conteúdo, e de sua

formatação, para que o usuário perceba valor no usufruto da nova tecnologia. Entre os principais serviços possíveis para a TV Digital, estão a interatividade, a alta definição e a portabilidade em diversos dispositivos móveis.

Neste âmbito a interatividade aparece como principal fator potencializador de inovação de conteúdo agindo também como ponto chave na percepção do usuário sobre o conteúdo.

2.1.5 Serviços Interativos

Os serviços interativos permitem uma maior participação do usuário na escolha e formatação de conteúdo, o que, no limite dessa funcionalidade, pode trazer para o mundo da televisão um novo universo de aplicações e possibilidades similares às da Internet. Entre as novas aplicações providas com interatividade local, encontram-se multicâmeras, serviços extras vinculados ao programa, portal de informação, novos formatos de publicidade e Guia Eletrônico de Programação (GEP). Dependendo do cenário, a existência do canal de retorno permite, entre outros, serviços e aplicações como cursos e jogos on-line, envio de mensagens curtas, correio eletrônico, participação em programas com respostas individualizadas, TV-Gov (declarações, prontuários, agendamentos de serviços etc.), portal de informações, noticias personalizadas, comércio eletrônico e publicidade dirigida com resposta.

Este tipo de conteúdo é novo no mercado de televisão. Apenas algumas empresas de TV por assinatura fornecem serviços de interatividade, porém de maneira muito mais restrita: uma vez instalado o terminal de acesso, os serviços de interatividade já estão definidos e não mudam de acordo com a programação.

Quando vinculados aos programas de televisão e às publicidades, os serviços de interatividade passam a exigir uma demanda constante por produção. Desta maneira, a eficiência do processo de produção reduzirá os custos de produção impactando conseqüentemente na margem de lucro das produtoras. O projeto apresentado por este documento, possibilita a criação de ferramentas de autoria de conteúdo, portanto contribui de forma indireta na melhoria deste processo de produção.

Basicamente, aplicações interativas podem ser classificadas em três diferentes níveis de interatividade, sendo que o produtor poderá misturá-los para obter o apelo comercial ou aplicativo desejado. São os níveis:

- Nível reativo: as opções de conteúdo e feedback são fornecidas pelo próprio programa, e o usuário basicamente seleciona as informações e conteúdo ao qual deseja ter acesso. Temos aqui aplicações como seleção de legendas, canais, apresentação

de informações ao vivo, informações sobre preço e característica de produtos em cena, entre outras aplicações;

- nível coativo: o usuário pode escolher a seqüência de ações, telas, controlar o ritmo e interagir com as aplicações. Temos aqui aplicações com interatividade local, ou seja, podemos ter aplicações do tipo "Quiz", jogos interativos, propagandas interativas com seleção de cenas e modificação da estrutura do vídeo e conteúdo multimídia;
- nível proativo: o usuário poderá controlar tanto a estrutura quanto o conteúdo. Temos o nível máximo de interatividade, no qual o usuário poderá demandar outros tipos de conteúdo, interagir e enviar dados de volta ao transmissor de conteúdo, acessar conteúdos multimídia em sites da Web, postar dados, acessar e-mail, aplicações de e-banking, entre outras.

Com estes três níveis de interatividade, o produtor de conteúdo poderá criar e produzir conteúdos muito mais ricos e que atinjam diretamente o seu telespectador, podendo interagir com ele e efetuar vendas e consultas de mercado durante a própria transmissão do conteúdo.

2.1.6 Plataforma brasileira para interatividade (Ginga)

Resultado de anos de pesquisas lideradas pela Pontifícia Universidade Católica do Rio de Janeiro (PUC - Rio) e pela Universidade Federal da Paraíba (UFPB), o Ginga é a camada de software intermediário (*middleware*) que permite o desenvolvimento de aplicações interativas para a TV Digital de forma independente da plataforma de hardware dos fabricantes de terminais de acesso (set-top boxes). Este padrão brasileiro reúne um conjunto de tecnologias e inovações brasileiras que o tornam a especificação de *middleware* mais avançada e, ao mesmo tempo, mais adequada à realidade do país.

Durante a elaboração deste documento, a especificação completa do GINGA esteve em processo de aprovação pública, sendo analisada por todas as entidades de pesquisas envolvidas com a produção de tecnologia para a TV Digital brasileira. A aprovação do padrão é praticamente certa.

2.2 Equipamento Utilizado

Nesta seção, são detalhados todos os equipamentos, ferramentas, software e componentes utilizados no desenvolvimento do projeto.

Equipamentos:

- Recorder Generator DVRG da Rohde & Schwarz : gerador de TS utilizado como fonte de transmissão para o dispositivo USB;
- SFU Broadcast Test System da Rohde & Schwarz: modulador para transmissão de DVB-T para recepção pelo set-top box;
- Set-top box DVB Philips: utilizado para visualização dos vídeos transmitidos em uma TV comum;
- Osciloscópio Tektronix TDS220 de 100MHz: utilizado para visualização dos sinais analógicos e digitais durante o desenvolvimento do projeto;
- MacBook, HP Pavilion, Notebook DELL: utilizados pelos integrantes do grupo para desenvolvimento do projeto;

Software:

- IntelliJ IDEA: IDE para desenvolvimento de aplicações em JAVA;
- Eclipse: IDE para desenvolvimento de aplicações em JAVA;
- Xilinx ISE 9.2i WebPack: IDE para desenvolvimento do modelo em VHDL do dispositivo e geração dos arquivos de configuração da FPGA;
- ModelSim XE III 6.2: simulador de circuitos lógicos integrado ao ambiente da Xilinx para simulação do dispositivo;

Componentes e materiais:

- Kit Opal Kelly XEM 3001v2: kit de desenvolvimento USB com FPGA Spartan-3 da Xilinx utilizado como base para o dispositivo USB;
- Conector D-subminiatura 25 pinos: utilizado para encaixe com a interface SPI do gerador de TS e interconexão com os pinos de entrada do kit de desenvolvimento;
- Cabo USB: utilizado para interconectar o kit de desenvolvimento ao computador *host*;
- Solda de estanho/chumbo: utilizada para a elaboração do conector e interconexão dos pinos no kit de desenvolvimento;

Ferramentas:

- Ferro de solda 25W: utilizado para realizar a solda de componentes;
- Alicate pela fio;
- Alicate cortador de fio;

Capítulo 3

ESPECIFICAÇÃO DO PROJETO

3.1 Funcionalidades principais

As funcionalidades principais do dispositivo são:

- Captura de vídeo digital a partir de stream MPEG2 de entrada.
- Visualização do vídeo capturado na tela do computador.
- Interface com outros programas para edição, compressão e armazenamento dos vídeos capturados em diferentes formatos (MPEG, DivX, XVid, entre outros).
- Suporte a novos módulos de aplicações para TV Digital (flexibilidade).
- Interatividade através de canal de retorno via conexão de Internet.

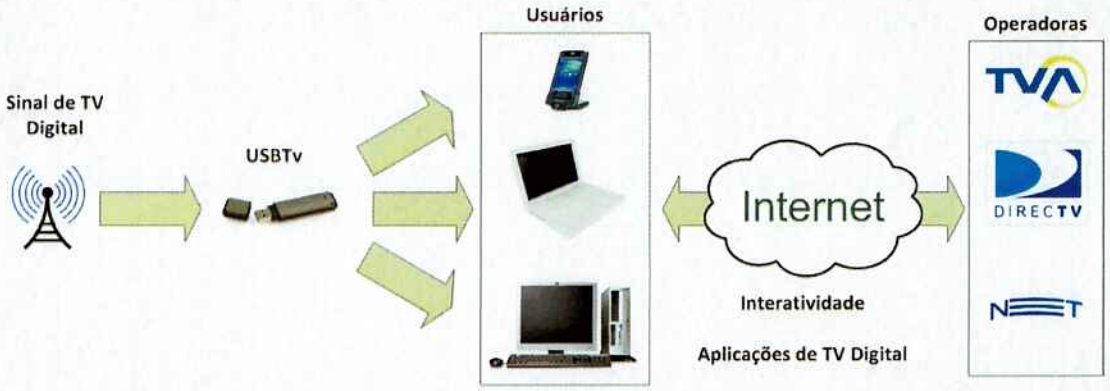

Figura 3.1: *Diagrama de blocos da aplicação principal*

3.2 Conexão USB

3.2.1 Software

Para o funcionamento do dispositivo através da conexão USB do computador faz-se necessário a criação de um driver para transferência dos dados. O desenvolvimento de drivers para plataforma Windows foi simplificado no final do ano de 2005 com a adição do

framework Windows Driver Foundation (WDF), uma camada acima e em nível mais alto que o Windows Driver Model (WDM). Tal tecnologia tem possibilitado agilidade na criação dos drivers e maior segurança através do encapsulamento de itens como a estrutura de dispositivos Plug and Play (PnP), gerenciamento de energia, requisições de I/O, eventos e interrupções.

3.2.2 Hardware

Inicialmente o hardware a ser utilizado no projeto consistia em um receptor de TV Digital por Radio Freqüência (RF), abrangendo assim todas as etapas de tratamento do sinal. Visando a simplificação, uma vez que a recepção do sinal não é o objetivo principal, o projeto foi reduzido para capturar uma saída MPEG2 Transport Stream (já digital) de um aparelho de geração de sinais, ao invés de obtê-lo a partir da demodulação e decodificação de um sinal RF. A Figura 3.2 mostra o diagrama em blocos que representa o esquema do hardware necessário.

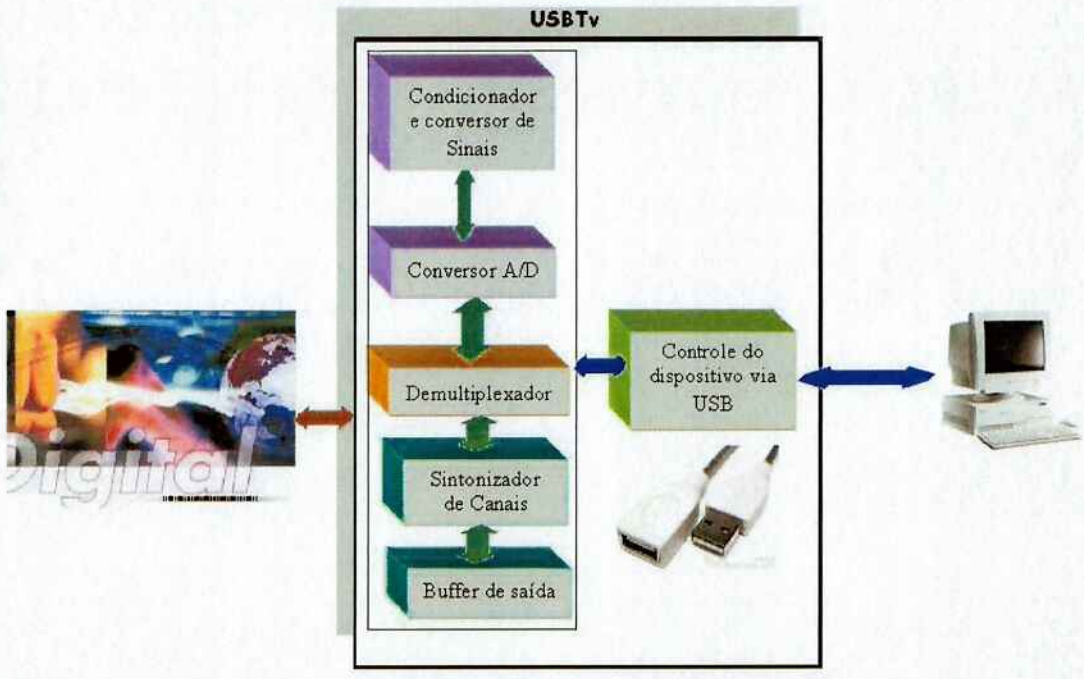

Figura 3.2: *Diagrama de blocos do hardware completo para a aquisição do sinal*

O gerador de sinais *DVRG*, da *Rohde & Schwarz*, dispõe de uma saída serial assíncrona com conector BNC a 270 Mbits/s e uma saída paralela síncrona de 25 pinos que transmitem dados em MPEG2-TS, facilitando o projeto e o desenvolvimento do sistema de

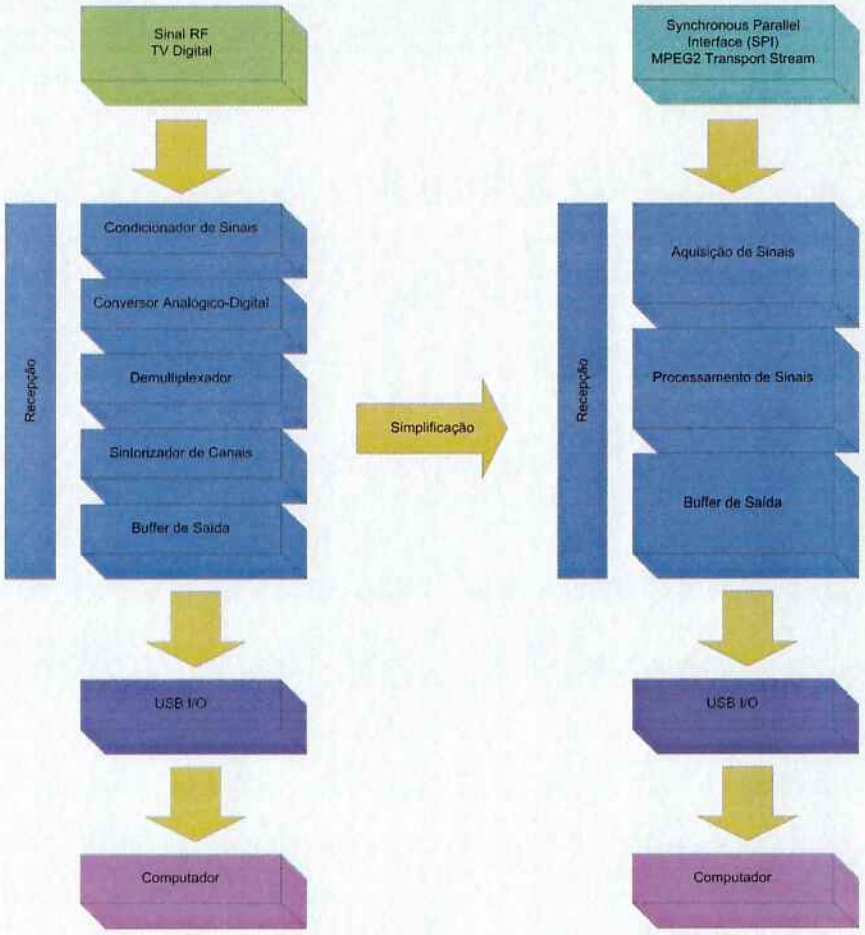

Figura 3.3: *Processo de simplificação para este projeto*

aquisição de sinais. Como o padrão USB 2.0 estabelece uma taxa de transferência de 480 Mbits/s, a viabilidade da transmissão do sinal para o computador não é comprometida.

A Figura 3.3 apresenta o diagrama de blocos do projeto original e a sua simplificação para o projeto no estágio atual.

3.3 Formato dos dados

Tecnologias de transmissão e modulação dos dados estão fora do escopo deste projeto. Não há nenhuma suposição sobre como são feitas a transmissão e a modulação dos dados. Apesar de o foco estar no modelo brasileiro de TV Digital, SBTVD, o projeto se encaixa com qualquer um dos padrões existentes para modulação e transmissão de conteúdo: o europeu DVB [13], o norte americano ATSC [10] e o japonês ISDB [14].

Para alcançar tal independência, o conteúdo será capturado no formato definido pelo padrão MPEG-2 [9], suportado por todos os padrões e tecnologias de transmissão do sinal de TV Digital.

3.3.1 O padrão MPEG-2

De uma forma geral, o padrão MPEG-2 define como informações de som, imagem e dados são codificadas, decodificadas, comprimidas, armazenadas e transmitidas. Ele também é um padrão internacional, o ISO/IEC 13818 e é divido em dez partes. São listadas abaixo as sete mais importantes:

- **Parte 1 - Sistemas**: sincronização e multiplexação de som, vídeo e dados. Também conhecida como ITU-T Rec. H.222.0.
- **Parte 2 - Vídeo**: codificação e compressão de vídeo. Também conhecida como ITU-T Rec. H.262.
- **Parte 3 - Áudio**: codificação e compressão de áudio.
- **Parte 4** - testes de compatibilidade com a especificação.
- **Parte 5** - descreve sistemas de simulação por software.
- **Parte 6** - mecanismos de proteção contra fraude (cópias e armazenamento) - DSM-CC (Digital Storage Media Command and Control).
- **Parte 7** - formato avançado para codificação e compressão de áudio, AAC (Advanced Audio Coding).

Para o projeto, a primeira parte (**Sistemas**) é a mais importante, pois define como são misturadas (ou multiplexadas) as informações de som, vídeo e dados a serem transmitidas. Esta parte ainda se divide em duas, sendo uma divisão adequada a aplicações de transmissão (*broadcast*), conhecida como *MPEG-2 Transport Stream* e uma divisão adequada a sistemas de armazenamento, conhecida como *MPEG-2 Program Stream*.

O projeto assume que os dados entram no formato MPEG-2 Transport Stream. A recepção e demodulação deste sinal estão fora do escopo.

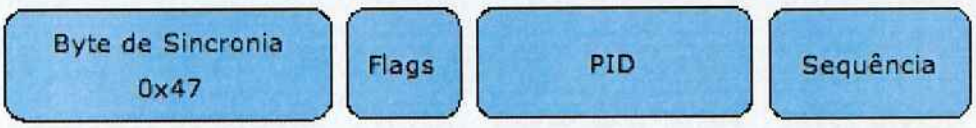

Figura 3.4: *Parte do cabeçalho dos pacotes*

3.3.2 Conteúdo como fluxo de dados: MPEG-2 Transport Stream

O padrão MPEG-2 Transport Stream é formalmente um protocolo de comunicação de áudio, vídeo e dados que tem como objetivo principal permitir a transmissão dos três componentes em um mesmo fluxo de dados, através da multiplexação no tempo. O protocolo ainda prevê recursos para sincronização entre os componentes e correção de erros no transporte sobre meios não confiáveis.

Os pacotes são a unidade básica do protocolo e geralmente têm tamanho de **188 bytes**. Um fluxo MPEG-2 é composto de pacotes pertencentes a diversos fluxos elementares, misturados pela multiplexação no tempo. A Figura 3.4 mostra a composição de uma parte do cabeçalho dos pacotes, que iniciam por um byte de sincronização de valor *0x47*.

Um dos componentes mais importantes dos cabeçalhos dos pacotes é o *Program ID* - PID, de 13 bits. Pacotes de mesmo PID formam um fluxo elementar, que pode transportar vídeo, áudio ou dados.

Um Programa é formado por um conjunto de fluxos elementares. Geralmente há um fluxo elementar para vídeo, dois para áudio (som estéreo) e um para dados, mas um programa pode ter quais fluxos elementares forem necessários. Os Programas são popularmente conhecidos como **canais de televisão** e comumente cada fluxo MPEG-2 pode carregar três deles.

Para a definição dos programas existentes em um fluxo MPEG-2, existem alguns pacotes especiais de PID 0x00. São pacotes pertencentes à PAT (*Program Association Table*), que listam os PIDs de quais pacotes contém informações sobre os programas.

Os pacotes que contém informações sobre os programas são conhecidos como PMTs (*Program Map Tables*) e definem quais são os conjuntos de PIDs que formam os Programas. Em outras palavras, os PMTs agrupam conjuntos de fluxos elementares em Programas e ainda contém metadados sobre cada um dos fluxos elementares; mostrando, por exemplo, se um fluxo elementar carrega vídeo, áudio ou dados.

3.4 Middleware

O nome vem de sua principal função: prover a interação entre as aplicações da TV Digital e o hardware do *set-top box*. Dito de outra maneira, o *middleware* é o software responsável pelo tratamento, decodificação e reprodução do áudio e do vídeo e pela execução das aplicações de TV Digital. O *middleware* é uma plataforma para a execução de aplicações de TV Digital.

Existem diversas especificações para *middlewares*. O mais famoso de todos é o *Multimedia Home Platform* (MHP) [16], definido no padrão europeu DVB [13] e baseado na tecnologia JavaTV. O MHP foi tão bem visto que acabou baseando diversas outras especificações de *middlewares* e fez com que fosse criado o *Globally Executable MHP* (GEM [17]), pedaço do MHP sem os detalhes específicos do DVB, para ser usado como base em outras tecnologias de transmissão e decodificação.

O GEM é a base que impulsiona uma das duas abordagens para *middlewares* de TV Digital; a abordagem procedural. Nesta abordagem, as aplicações de TV Digital são escritas como programas convencionais em alguma linguagem de alto nível (quase sempre em Java). Exemplos de *middlewares* que seguem esta abordagem são o europeu DVB-J (o próprio MHP, o "J" vem de Java), o norte americano OCAP (*OpenCable Application Platform* [12]), o outro norte americano ACAP (*Advanced Application Platform* [11]), o japonês ARIB B.23 [7] e o brasileiro Ginga-J (o "J" vem de Java [18]), coordenado pelo Laboratório de Aplicações de Vídeo Digital (LAViD) do Departamento de Informática (DI) da Universidade Federal da Paraíba (UFPB).

A outra abordagem para *middlewares* de TV Digital é a declarativa, onde as aplicações de TV Digital são programas declarados em linguagens derivadas do XML, como o XHTML. É um modelo de desenvolvimento parecido com o atualmente empregado na Internet, onde as páginas são geralmente declaradas em linguagem XHTML.

A abordagem declarativa ainda permite o uso de algum tipo de linguagem ou script para conseguir oferecer as mesmas funcionalidades que a abordagem procedural. Geralmente é possível descrever algumas funcionalidades específicas da aplicação de TV Digital de forma procedural através destas linguagens embutidas. Um exemplo de linguagem largamente utilizada para este fim é o ECMAScript, popularmente conhecida como JavaScript.

O pioneiro da abordagem declarativa é o padrão de *middleware* japonês definido pela organização ARIB que usa a linguagem BML (*Broadcast Markup Language*) baseada no XHTML. Outros exemplos de middlewares existentes que seguem a abordagem declarativa são o europeu DVB-HTML, o norte americano ACAP-X e o brasileiro Ginga-NCL, que usa

a Nested Context Language (NCL) e é desenvolvido pelo Departamento de Informática da Pontíficia Universidade Católica do Rio de Janeiro (PUC - Rio).

A especificação que define o middleware incluído no padrão brasileiro de TV Digital – SBTVD – é o Ginga, incluindo tanto o Ginga-J quanto o Ginga-NCL.

Para o projeto, foram consideradas duas alternativas para adoção: o europeu DVB-J/MHP como abordagem procedural, ou o brasileiro Ginga-NCL como abordagem declarativa. A escolha teve de ser feita com base nas ferramentas e implementações existentes para ambas as alternativas, tendo em vista que o projeto visa estender uma alternativa de middleware livre existente para prover as funcionalidades desejadas.

A primeira escolha foi trabalhar com o padrão europeu MHP, já que o padrão brasileiro Ginga ainda vinha sendo desenvolvido e não estava facilmente disponível para consulta. Porém durante o desenvolvimento do projeto, foi anunciado o lançamento público das especificações do Ginga e a implementação de referência do Ginga-NCL foi disponibilizada ao público através do portal Software Público, do governo federal (`http://www.softwarepublico.gov.br`).

Tais fatos fizeram com que o direcionamento da tecnologia de middleware para a interatividade fosse firmada na utilização do Ginga [18] (em oposição ao MHP), por ser parte integrante da especificação do padrão do Sistema Brasileiro de TV Digital e pela possibilidade de participar e contribuir com a implementação da especificação, por tratar-se de software livre.

Esta foi uma decisão importante, que afetou de maneira crítica o direcionamento do projeto, que originalmente estava dimensionado e baseado no padrão europeu MHP. Felizmente, todos os padrões mundiais de middleware para TV Digital têm pontos em comum, padronizados pelo GEM. Oportunamente, o padrão de middleware brasileiro adotado, também é baseado neste padrão internacional, o que permitiu o aproveitamento da maior parte do trabalho já realizado no projeto.

Dadas as desvantagens de se mudar a direção do projeto já em estágio avançado, a importante decisão foi tomada principalmente levando em consideração a potencial contribuição social e tecnológica deste Projeto de Formatura ao país. O Brasil está criando o seu próprio padrão de middleware para TV Digital, o que incentiva diretamente o desenvolvimento de tecnologia nacional. Além disso, cria-se uma reserva de mercado, já que as empresas multinacionais do setor ainda não têm a tecnologia para produção em escala, baseada no middleware brasileiro. Esta reserva abre oportunidade para novos empreendedores dentro do país.

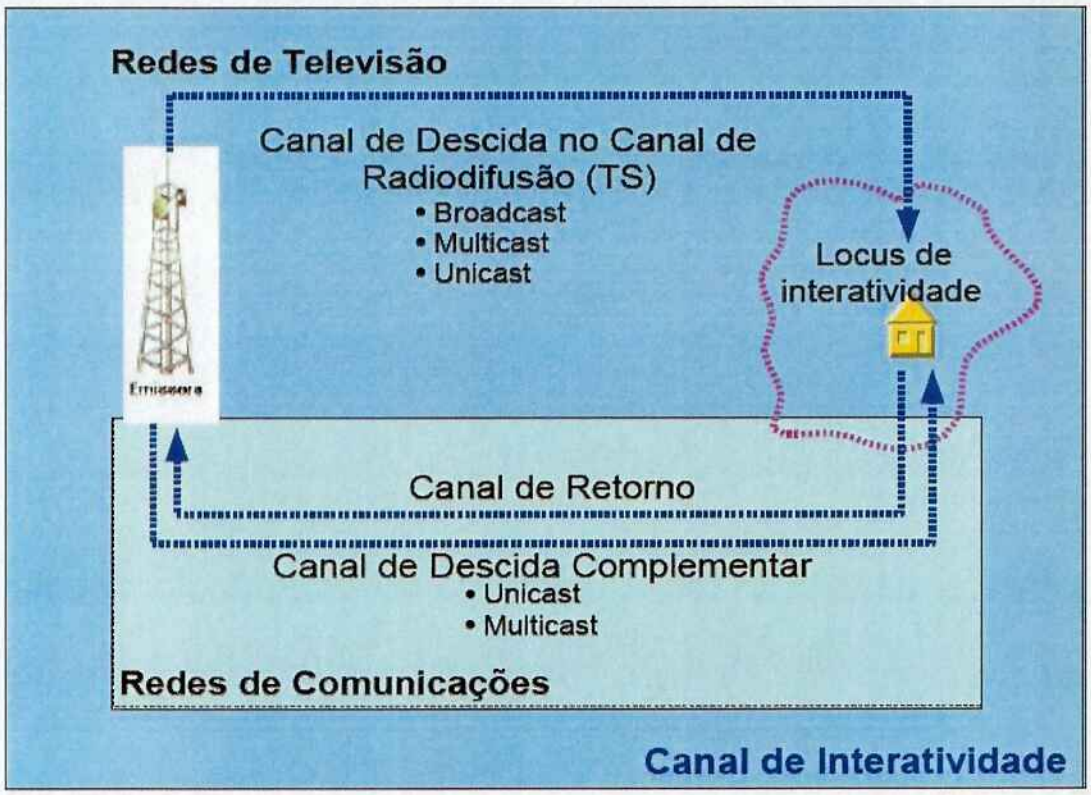

Figura 3.5: *Diagrama do Canal de Interatividade*

3.5 Canal de retorno

3.5.1 Conceito

O Canal de Retorno, ou Canal de Interatividade, é a camada responsável por viabilizar a comunicação das aplicações interativas, no terminal de acesso, com os servidores de aplicação do provedor de conteúdo. Desta forma, pode-se conceituar o Canal de Interatividade como o subsistema que permite que cada usuário, individualmente, possa interagir de forma própria com o conteúdo multimídia disponibilizado pelas emissoras, enviando e/ou recebendo informações e complementando atividades e aplicações disponíveis.

A arquitetura prevista para o Canal de Retorno tende a utilizar múltiplas tecnologias de comunicação, o que permite um aumento na capacidade de acesso, cobertura e número de usuários com acesso a conteúdos adicionais ao conteúdo normal disponibilizado.

No caso do SBTVD, o uso de diferentes meios de comunicação para a implementação do Canal de Retorno é essencial, já que a maior meta do SBTVD é a inclusão digital de regiões longínquas, onde o acesso à informação e cultura é escasso e limitado. Portanto, o canal de retorno deve possibilitar interatividade a partir dos meios disponíveis em tais

regiões, havendo assim a necessidade de se adaptar a diferentes subsistemas de comunicação, desde as redes GSM, como as redes de telefonia discada, comunicação via RF e outras.

3.5.2 Aplicações

A interatividade pode ser dividida em três níveis, como já descrito na Seção 2.1.5. Além disso, sob o ponto de vista das aplicações interativas que se utilizam do Canal de Retorno, podemos dividi-las em dois tipos de interatividade, sendo estas as aplicações de Interatividade Intermitente e Interatividade Permanente.

Aplicações de Interatividade Intermitente

Para este tipo de aplicações, a comunicação entre o telespectador e o emissor de conteúdo se dá de forma unidirecional, ou seja, o telespectador apenas envia informações para o emissor, sem esperar nenhuma resposta a cerca das informações enviadas.

Neste tipo de aplicação se enquadram o envio de votos de enquetes, pesquisas de opinião, jogos do tipo quiz, envio de mensagens SMS, entre outras aplicações. Neste caso, o usuário apenas utiliza o canal de retorno como forma de enviar informações, utilizando alguma aplicação do emissor de conteúdo para tratar os dados enviados. Assim, não é necessária a comunicação do emissor de conteúdo para o telespectador através do canal de retorno, pois todos os dados necessários já foram enviados através das aplicações de dados recebidas no terminal de acesso.

Apesar de possuírem um grau de interatividade limitado, tais aplicações já permitem que o emissor de conteúdo interaja com seus telespectadores, porém não há emissão de conteúdo individual e exclusivo para cada telespectador, há apenas a coleta de informações e tratamento das mesmas pelo emissor.

Aplicações de Interatividade Permanente

Para este tipo de aplicações, a comunicação entre o telespectador e o emissor de conteúdo passa a ser bidirecional, ampliando o nível de interatividade e permitindo a emissão de conteúdo exclusivo do emissor para o telespectador.

Tais aplicações passam a incorporar funções que poderiam existir em um computador conectado à Internet, tais como envio e recebimento de e-mails, jogos multiusuários, home banking, chat, educação à distância, compras on-line, pesquisa, etc. Este nível de inter-

Figura 3.6: *Aplicação SMS utilizando interatividade intermitente.*

atividade permite inclusive que conteúdo multimídia extra seja enviado ao telespectador, conforme suas necessidades e serviços disponíveis.

Outra aplicação acrescentada é a possibilidade de comunicação entre os telespectadores, permitindo a discussão de conteúdos e uma interação completa entre todos os receptores de conteúdo do mesmo emissor.

Como um exemplo do uso extensivo deste grau de comunicação, podemos ilustrar a seguinte situação. Um telespectador está assistindo a um evento esportivo de seu gosto, como por exemplo, uma partida de futebol. Ele se interessa em comprar a mesma chuteira que o seu jogador predileto usa. Utilizando os dados já enviados pelo emissor de conteúdo, ele sabe a marca e modelo da chuteira que o jogador está usando (interatividade local). Ele pode utilizar uma aplicação que compare o preço do produto em diferentes lojas on-line. Após isto, ele pode entrar em um fórum de discussão onde outros telespectadores comentam sobre o produto e sobre o atendimento das lojas nas quais eles compraram o produto. Utilizando uma aplicação de home banking, ele consulta o seu saldo no banco. Baseado em todas as informações, o telespectador decide comprar o produto e acessa via alguma aplicação a loja na qual o produto está mais barato. Efetua a compra e continua assistindo ao jogo, feliz em saber que em alguns dias terá a mesma chuteira que a de seu jogador favorito.

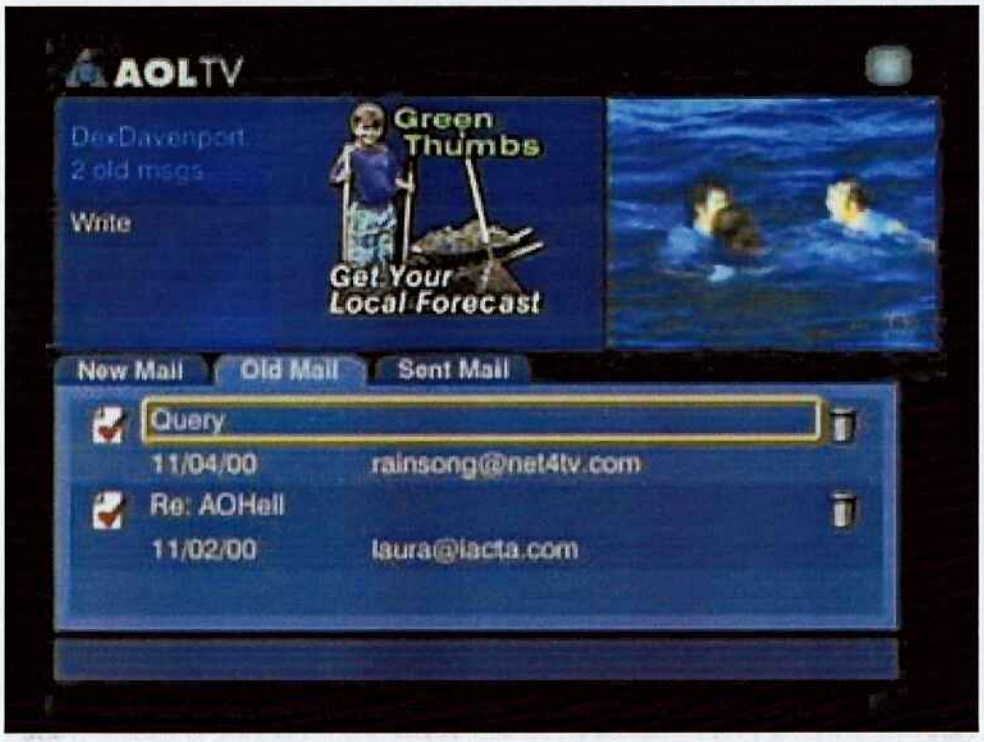

Figura 3.7: *Aplicação de e-mail com interatividade permanente.*

3.5.3 Implementação

Como foco principal do projeto, é realizada a implementação do Canal de Retorno do dispositivo USBTv através de uma conexão qualquer com a Internet, seja através de conexão discada, ADSL, cabo, sem fio, ou qualquer outra forma disponível para o usuário.

Como já previsto no SBTVD, o canal de retorno deve ser de fácil implementação e permitir o uso de diferentes tecnologias. Supondo que, para a maior parte dos usuários de computador, já existe uma conexão com a Internet, nada melhor do que utilizá-la como o canal de retorno, permitindo assim uma integração total das aplicações de TV Digital com as aplicações disponíveis para um computador comum através da Internet.

Assim sendo, o projeto deve analisar e implementar a camada de interface entre as aplicações de TV Digital e os serviços disponíveis na Internet, inicialmente para o *middleware* escolhido, porém expansível para os demais tipos de *middleware* existentes.

O uso da Internet como canal de retorno viabiliza tanto a interatividade intermitente como a permanente, ampliando-as ainda mais, pois permite a interação com os sistemas já existentes de conteúdo multimídia na Internet. Poderíamos citar, por exemplo, uma aplicação de TV digital que permitisse a busca por vídeos no YouTube similares ao conteúdo que está sendo atualmente transmitido pelo emissor de conteúdo e, através do canal de retorno, carregar e apresentar o vídeo simultaneamente à transmissão de TV. Ou ainda,

poderíamos prever uma integração com os serviços já disponibilizados via Web Services de lojas como Amazon, permitindo o reuso das tecnologias já existentes na Internet para o uso com aplicações de TV Digital.

Assim, o projeto deve viabilizar e formalizar o canal de retorno através da Internet, analisando quais os protocolos de comunicação mais adequados a serem utilizados e realizando sua implementação como modelo de referência e prova de conceito.

3.6 Planejamento e Métodos

Todas as tarefas desenvolvidas pelos integrantes do grupo podem ser divididas em três grandes conjuntos de tarefas, as quais são desenvolvidas por cada um dos integrantes do grupo de acordo com seu grau de interesse pelo assunto.

Foram previstas etapas de integração das tarefas desenvolvidas individualmente, para que o grupo como um todo possa entender e acompanhar o funcionamento de cada uma das partes do projeto.

Cada grupo de tarefas representa problemas distintos, necessitando para tanto o uso de diferentes abordagens para sua resolução. A seguir estão detalhados cada um dos conjuntos de tarefas, assim como os métodos e técnicas que serão utilizados para o seu desenvolvimento.

3.6.1 TV Digital e Middleware

Este conjunto de tarefas engloba todas as etapas de pesquisa e desenvolvimento necessários para o entendimento e uso das atuais tecnologias e plataformas de desenvolvimento para a TV Digital.

São necessários o entendimento dos formatos de dados, fluxos, codificação, modulação, empacotamento e transmissão do sinal multimídia em fluxo de dados, bem como o entendimento profundo do middleware a ser utilizado como plataforma para as aplicações de TV Digital.

Fica de responsabilidade deste conjunto de tarefas o entendimento e uso adequado das ferramentas de desenvolvimento, simulação e execução de aplicações, além da geração e/ou simulação da transmissão de TV Digital a ser captada e apresentada pelo dispositivo USBTv.

Sinal de TV Digital

Esta tarefa consiste em obter e utilizar dispositivos de geração, transmissão e recepção de sinal de TV Digital para o uso como entrada para o dispositivo de captura. Isto tem como função principal simular o sinal recebido por um telespectador transmitido por um provedor de TV Digital, baseado nos padrões disponíveis atualmente no mercado. É dada prioridade ao formato escolhido pelas especificações do padrão brasileiro de TV Digital (SBTVD), porém outros padrões poderão ser utilizados, desde que se obtenha como produto final o MPEG-2 Transport Stream para o uso com o dispositivo de captura USBTv.

Complementação do Middleware

Esta tarefa consiste em testar e analisar os diferentes padrões de *middleware* previstos para o SBTVD (Ginga-J e Ginga-NCL) e verificar qual padrão oferece melhor suporte e viabilidade técnica para o desenvolvimento das aplicações de TV Digital com suporte ao canal de retorno pela Internet.

Inicialmente foram abordados os padrões GINGA-NCL e o MHP, sendo analisadas as suas especificações e ferramentas disponíveis no mercado para o desenvolvimento e simulação de aplicações em cada um dos *middlewares*.

3.6.2 Captura do Sinal via USB

Este conjunto de tarefas engloba todas as etapas de pesquisa e desenvolvimento necessárias para a implementação das camadas de hardware e software, que permitam o envio do sinal de TV Digital no formato MPEG-2 Transport Stream como fluxo serial de dados para o uso nas plataformas de aplicações de TV Digital no computador, através de uma porta USB.

Este conjunto é a parte do projeto que permite que o mundo da TV Digital possa ser aproveitado, assistido e utilizado em qualquer computador que possua interface USB. Para tanto, deverá implementar não só o hardware mas também os *drivers* e componentes de software necessários para o uso do sinal de TV Digital no computador.

Este conjunto de funcionalidades pode ser simulado ao longo do projeto para o desenvolvimento das outras etapas, permitindo o paralelismo do trabalho. Existem no mercado softwares que geram arquivos no formato MPEG-2 Transport Stream, que podem ser utilizados como sinal de entrada para as plataformas de *middleware* e para o desenvolvimento e testes do canal de retorno. Porém, o projeto deve implementar tal interface de hardware

para que os objetivos de portabilidade e usabilidade especificados no projeto sejam completamente alcançados.

Hardware USB

Esta tarefa consiste em especificar, detalhar e implementar o hardware necessário para a captura do MPEG-2 Transport Stream e seu envio em formato serial para o computador através do padrão USB.

Como método, pode ser citado o uso de diagramas de blocos funcionais, que a cada nível detalham os requisitos específicos de cada módulo, detalhando suas entradas e saídas. A arquitetura é detalhada até a obtenção de módulos compatíveis com componentes existentes no mercado, de forma a facilitar a implementação do hardware necessário.

Está aberta a possibilidade do uso de kits de desenvolvimento, que forneçam suporte ao tratamento dos dados e comunicação USB, além de ser necessária a completa especificação e elaboração do circuito lógico e físico dedicados para a aplicação USBTv, permitindo assim uma possível implementação comercial do projeto.

Software USB

Esta tarefa consiste em especificar, detalhar e implementar os drivers e aplicativos necessários para o tratamento da entrada de dados USB e sua correta formatação para o uso do sinal como entrada para aplicações de TV Digital.

Inicialmente está previsto o desenvolvimento de drivers para o Sistema Operacional Windows, uma vez que já existem bibliotecas padrão que facilitam a implementação da comunicação USB. Para o desenvolvimento é utilizado o WDK (*Windows Driver Kit*), que já possui extensões para a implementação de drivers USB.

São utilizadas as ferramentas e padrões de desenvolvimento estabelecidos por órgãos internacionais, como os padrões definidos [19].

Caso possível, podem ser desenvolvidos também drivers e aplicações para outros sistemas operacionais, como o Linux e o Mac OS X.

3.6.3 Canal de Retorno pela Internet

Este conjunto de tarefas engloba todas as etapas de pesquisa e desenvolvimento necessárias para a implementação da interface e protocolos de comunicação que possibilitem o canal de interatividade via conexão com a Internet.

Tais tarefas devem apresentar como resultado final um arcabouço que permita que qualquer aplicação de TV Digital consiga acessar serviços disponibilizados na Internet por emissores de conteúdo, de preferência independentemente do padrão de *middleware* utilizado.

Dentro destas tarefas, está também a migração de algum serviço disponível na Internet como serviço para aplicações de TV Digital, utilizando os protocolos e bibliotecas desenvolvidas para fazer a comunicação e interfaces necessárias como canal de retorno. Ou seja, deve ser criada uma aplicação de exemplo que utilize o canal de retorno implementado para executar alguma tarefa atualmente só disponível para aplicações de computadores.

Como métodos de desenvolvimento, podem ser citados o uso das camadas do modelo OSI, sendo definidas as interfaces, entradas e saídas (protocolos) para cada uma das camadas necessárias para a implementação do canal de retorno via Internet.

São utilizados os padrões da UML para documentação do software a ser desenvolvido, sendo utilizadas ferramentas abertas de modelagem de software (como, por exemplo, a ferramenta ArgoUML para modelagem UML) e dadas as características da equipe e do projeto, optou-se por uma metodologia ágil e iterativa de desenvolvimento, baseada em resultados e metas parciais.

Capítulo 4

METODOLOGIA

4.1 Metodologia Ágil

Como metodologia de processo de desenvolvimento de software o grupo adotou métodos ágeis, utilizando como base para gerenciamento de projetos a metodologia *SCRUM*.

Esta metodologia consiste em um processo incremental e iterativo, no qual desenvolvedores e interessados têm uma relação muito próxima e versões intermediárias são entregues assim que determinadas funcionalidades são completadas.

São realizadas pequenas reuniões diárias para avaliação do andamento das equipes no desenvolvimento de cada funcionalidade, havendo assim uma troca de experiências, problemas e soluções entre os integrantes da equipe. Ao final de prazos pré-estabelecidos (para este projeto o prazo foi de uma semana), são entregues versões das funcionalidades para que os interessados possam testá-las, sugerir modificações e propor melhorias. Desta forma, é garantida a qualidade e prazos para funcionalidade e, ao final de todo o processo iterativo, o produto é finalizado com todas as suas funcionalidades testadas e aprovadas pelos interessados.

A Figura 4.1 exibe o fluxo do processo de desenvolvimento iterativo adotado.

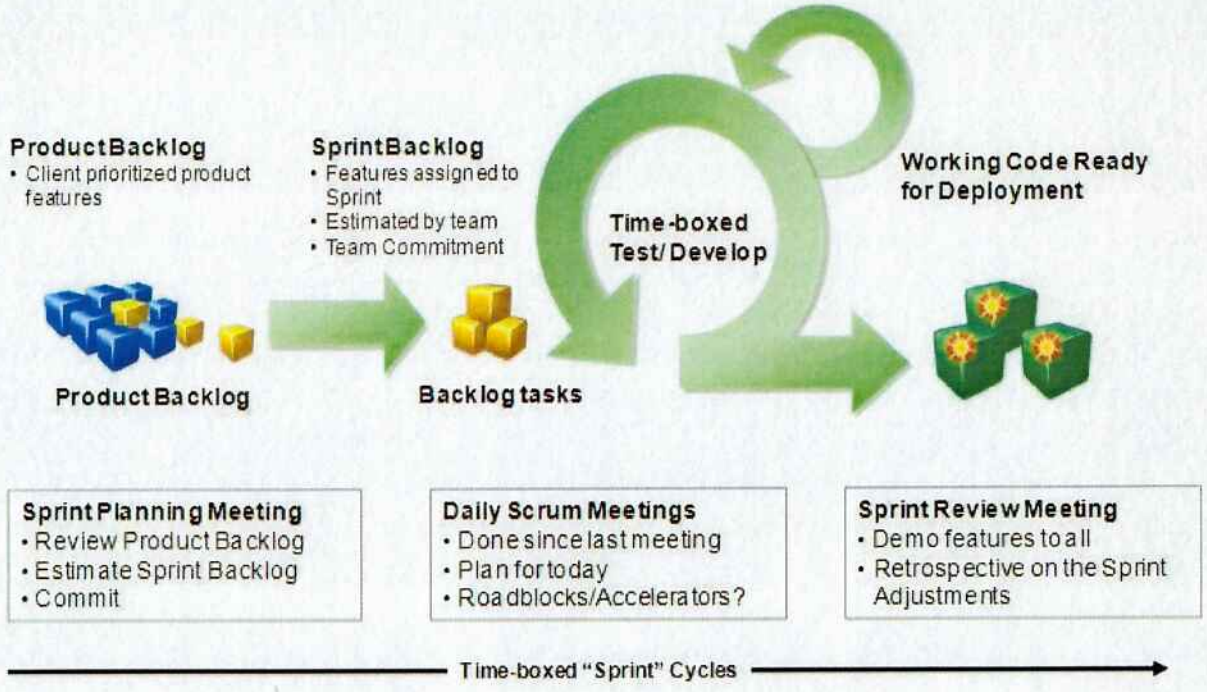

Figura 4.1: *Descrição da metodologia de desenvolvimento ágil Scrum*

4.2 Desenvolvimento do Hardware

Como metodologia para o desenvolvimento do hardware, também foi adotado o modelo incremental e iterativo, no qual a cada iteração novas funcionalidades foram sendo agregadas ao dispositivo em desenvolvimento. Assim, foi possível gerar versões intermediárias do dispositivo, a medida das quais ele se tornava mais complexo e apto a realizar completamente determinadas funcionalidades.

Como modelo de arquitetura de desenvolvimento de sistemas digitais, foi utilizado o modelo de máquina de estados com "datapath", no qual existe uma máquina de estados que controla as entradas, saídas e transições da unidade de controle, porém com interligação direta entre os diferentes componentes da arquitetura para a realização do fluxo de dados.

Assim, foi possível utilizar todas as técnicas comumente utilizadas no desenvolvimento de sistemas digitais [31] baseados em máquinas de estados, no qual é possível acompanhar o estado atual do dispositivo e o andamento do fluxo de dados ao longo das transições entre estados. Desta forma, foi possível validar o sistema com simulações de sua lógica interna, antes mesmo de tê-lo implementado no hardware físico.

Desta forma, foi possível garantir em ambiente de desenvolvimento e simulação o funcionamento lógico do dispositivo, validando-se inclusive o fluxo de dados entre os diferentes componentes desenvolvidos.

Para o design e avaliação individual de cada componente, foram utilizadas as metodologias de projeto lógico digital descritas em [32], no qual são definidas técnicas para desenvolvimento de memórias e componentes lógicos digitais.

Capítulo 5

PROJETO E IMPLEMENTAÇÃO

5.1 Dispositivo USB

Baseando-se nas especificações dos requisitos funcionais e não funcionais para o dispositivo de captura de vídeo digital através da porta de comunicação USB, foi possível elaborar uma arquitetura que implementasse todo o tratamento do sinal de TV Digital até a comunicação serial com o computador de destino dos dados.

A arquitetura resume-se a componentes de hardware que efetuam o tratamento dos sinais transmitidos pela interface SPI do Gerador de Transport Stream, tornando-os compatíveis para o tratamento lógico digital. Em seguida, os dados são amostrados e armazenados em unidades de memória organizadas como um FIFO, e por fim lidas pela interface USB pelo computador host.

A Figura 5.1 ilustra a arquitetura adotada, representando os principais blocos utilizados para a implementação do dispositivo de captura USB. As demais seções irão detalhar o funcionamento de cada bloco individualmente, bem como os componentes lógicos e de hardware, além padrões de interfaces utilizados na implementação de tal arquitetura.

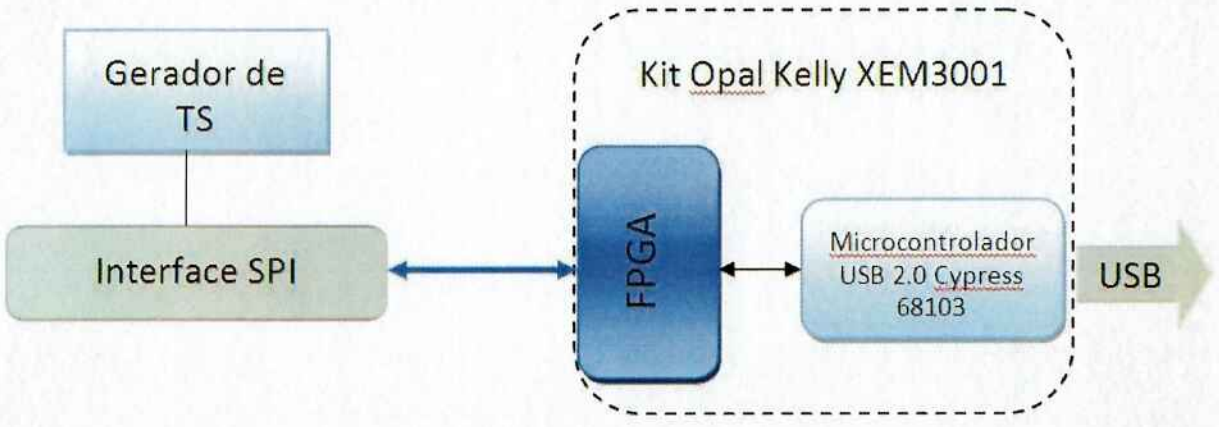

Figura 5.1: *Visão geral da arquitetura do dispositivo USB*

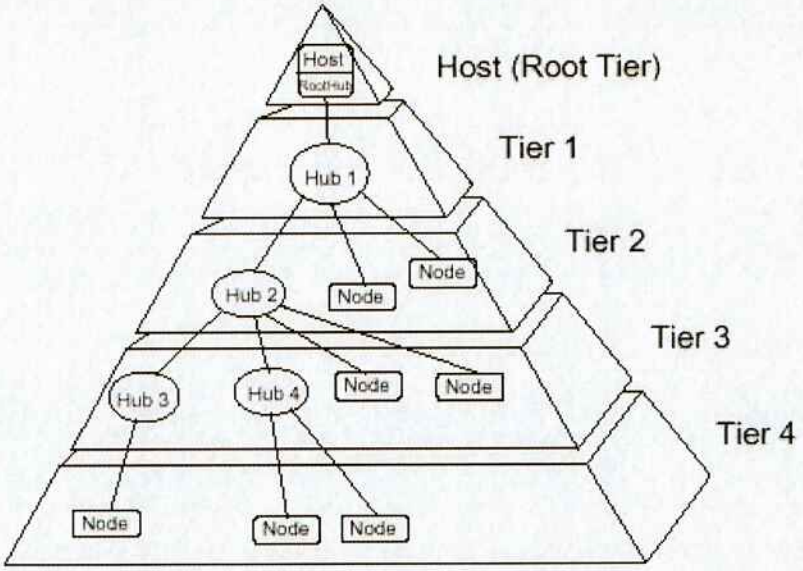

Figura 5.2: *Topologia da arquitetura USB*

5.1.1 Arquitetura USB

Para desenvolver um dispositivo de captura USB, é necessário entender a arquitetura e o funcionamento do padrão USB, para que o dispositivo a ser interconectado possa implementar a interface de comunicação com o dispositivo host.

De acordo com [19], um sistema USB é descrito por três áreas fundamentais: a interconexão, os dispositivos e o host. A interconexão descreve como os dispositivos USB são conectados e se comunicam com o host, isto inclui a definição da topologia do barramento, as relações inter-camadas, os modelos de fluxo de dados e a listagem USB. O host é o dispositivo controlador, do qual se iniciam todas as transações. Os dispositivos são itens periféricos que recebem ou enviam dados ao host. Em geral, o host é um computador, e os dispositivos são periféricos tais como mouses, teclados, webcams, entre outros.

A topologia do barramento USB é a topologia conhecida por tiered-star (estrela disposta em camadas/níveis). Esta topologia permite que cada nó possa representar uma função ou outro hub no qual se interliguem mais dispositivos.

Todas as transações do barramento envolvem a transmissão de até três pacotes. Cada transação se inicia pelo controlador USB do host, que envia um pacote USB descrevendo o tipo e a direção da transação, o endereço do dispositivo USB e o número do ponto final (endpoint). O ponto final de um dispositivo é a porção endereçável dele, que é a fonte de informação em um fluxo de comunicações entre o host e o próprio. O número de endpoint

é um valor de 4 bits entre 0H e FH, inclusive, associado a um ponto final de um periférico USB.

A arquitetura USB compreende quatro tipos básicos de transferências de dados:

- **Transferência de Controle**: Usada para configurar um dispositivo no instante de sua conexão e pode ser usada para outros propósitos específicos, incluindo controle de outros pipes no dispositivo;
- **Transferência do tipo Bulk**: Gerada e consumida em grandes quantidades e simultaneamente. Possui uma ampla e dinâmica latitude em transmissões de reserva;
- **Transferência de Interrupção**: Usada para caracteres ou coordenadas com percepções humanas ou características de respostas regenerativas;
- **Transferência Isossíncrona de Dados**: Ocupa uma quantidade pré-negociável da banda de transmissão do barramento, com a distribuição de pulsos. Chamada também de transferência de correntes em tempo real (streaming real-time transfers);

Estes tipos de transferência são implementados por componentes simples no kit de desenvolvimento escolhido para o projeto. Maiores detalhes sobre os tipos de transferência e como elas são utilizadas estão descritos nas próximas seções deste documento.

5.1.2 Gerador de Transport Stream

Como o objetivo do projeto é a captura do sinal de TV Digital, foram utilizadas as instalações e equipamentos de TV Digital disponíveis no Laboratório de Sistemas Abertos da Escola Politécnica da USP. O equipamento é um gerador e gravador de Transport Stream, baseado no padrão europeu de TV Digital (DVB). Portanto, foram necessárias adaptações aos padrões de Transport Stream, de forma que vídeos gerados e obtidos pelo grupo pudessem ser transmitidos adequadamente pelo equipamento.

O DTV Recorder Generator DVRG da Rohde & Schwarz possui diversas interfaces de entrada e saída de Transport Stream, sendo responsável pela sincronização e geração dos sinais de clock e sincronização do sinal nas diferentes saídas. Os padrões mais utilizados para realizar a interface entre dispositivos de geração e transmissão de Transport Stream são a interface SPI (Synchronous Parallel Interface) e a interface ASI (Asynchronous Serial Interface), detalhadas posteriormente neste documento.

O gerador pode ser controlado via acesso remoto, pelo qual os arquivos de vídeo são carregados e codificados no padrão proprietário do equipamento. Devido a restrições em

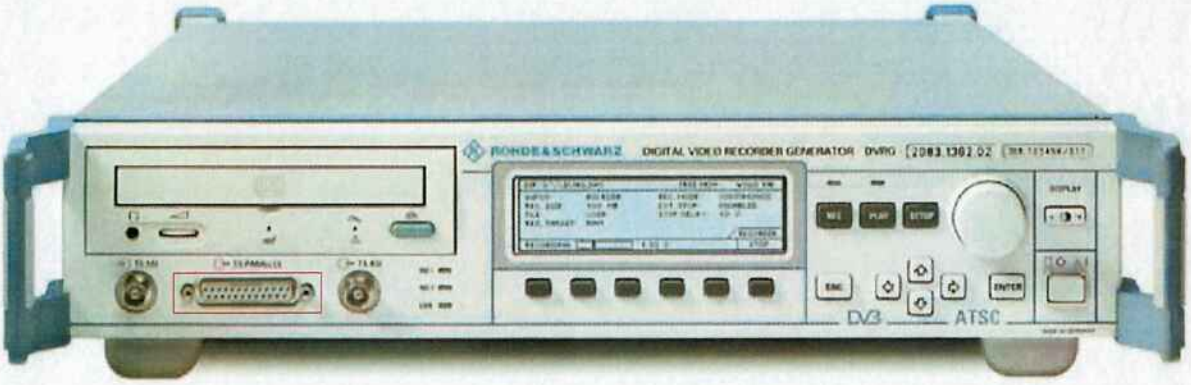

Figura 5.3: *Gerador de TS da Rohde & Schwarz com interface SPI*

relação à capacidade do equipamento ao gerar os sinais de sincronia, é necessário gerar arquivos de Transport Stream sem variação de taxa de bits (bitrate constante), além de outras especificações que permitam ao equipamento transmitir os pacotes contidos no arquivo.

5.1.3 Interfaces e Padrões

Conforme mencionado anteriormente, os padrões mais utilizados para a transmissão são os padrões definidos na especificação técnica do padrão europeu DVB [23]. Nesta especificação são definidas duas interfaces para transmissão dos pacotes de TS, cada uma com diferentes taxas de transmissão e formato dos dados. A seguir serão detalhadas as duas interfaces, bem como a utilização da interface SPI na implementação do dispositivo de captura USB.

Interface ASI

A interface ASI é uma interface cujo objetivo é ter uma grande taxa de transmissão de dados, uma vez que pode ser implementada sob diferentes meios físicos de transmissão, seja via cabo coaxial, fibra ótica ou outros. Por se tratar de um canal serial, todos os bits são transmitidos em série, havendo necessidade de uma organização dos pacotes de forma que seja possível sincronizar o sinal e obter informações sobre o clock e bitrate do vídeo sendo transmitido.

Para tanto, a interface ASI está organizada em três camadas. Conforme pode ser observado pela Figura 5.4 A camada dois é a responsável pela definição dos pacotes de TS (188 ou 204 bytes), bem como a inserção do byte de sincronia (0x47) para sincronização dos pacotes. A camada um é a responsável pela codificação dos dados no formato 8B/10B,

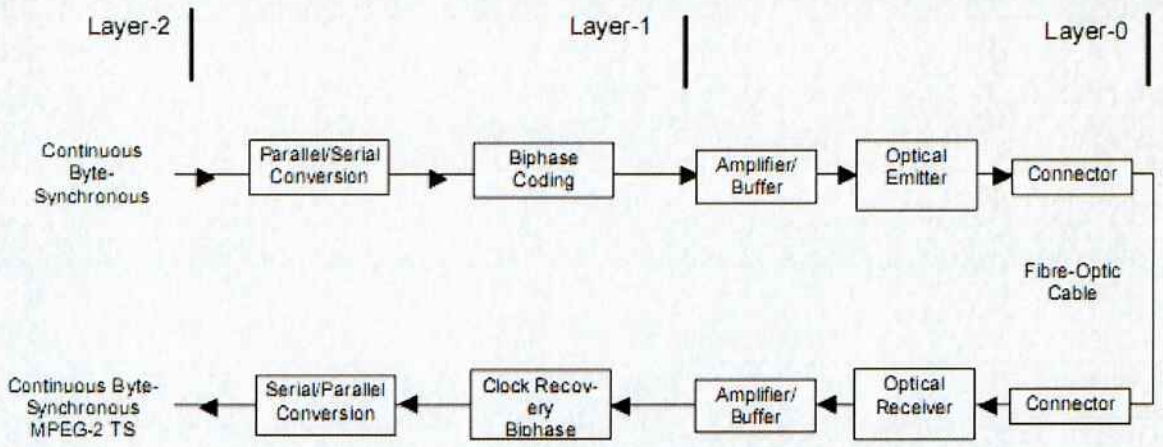

Figura 5.4: *Camadas da interface ASI*

de forma a inserir bits de redundância para correção de erros. Finalmente, a camada zero é a física, responsável pela transmissão e recepção dos bits.

A interface prevê uma taxa constante de 270 Mbps, e caso a fonte não possua dados a serem transmitidos nesta taxa, devem ser inseridos pacotes extra de sincronia de forma a manter a taxa de bits constante. Devido a esta taxa relativamente alta de operação, o grupo optou pelo padrão SPI detalhado a seguir, de forma que a freqüência de leitura dos dados fosse menor, por se tratar de uma interface paralela ao invés de serial como a ASI.

Interface SPI

A interface SPI baseia-se na transmissão paralela dos bytes, além de acrescentar bits de sincronia e clock de forma a facilitar a transmissão e recepção dos bits de informação. Como a interface utiliza o padrão LVDS para a transmissão física dos dados, são necessários dois fios para a transmissão de um único sinal lógico. Assim, a interface utiliza um conector D-subminiatura de 25 pinos [26], destinando 16 pinos para a transmissão do byte de dados, dois para o sinal de clock, dois para o sinal de sincronia e outros dois para o bit de sinalização da validade do byte, conforme pode ser observado na Figura 5.5.

Conforme a Figura 5.6, a sincronização do sinal é feita pelo bit de sincronia (PSYNC), cuja função é sinalizar o início de cada pacote de 188 ou 204 bytes que está sendo transmitido. A sincronia também é garantida pelo sinal de clock, que é um sinal cuja freqüência varia conforme a taxa de dados do TS sendo transmitido. Assim, um TS de bitrate de 20Mbps irá gerar um sinal de clock de aproximadamente 2,5MHz.

O bit de validade (DVALID) serve para sinalizar a validade dos bytes transmitidos, especificamente quando a transmissão utilizar pacotes de tamanho variável ou de 204 bytes.

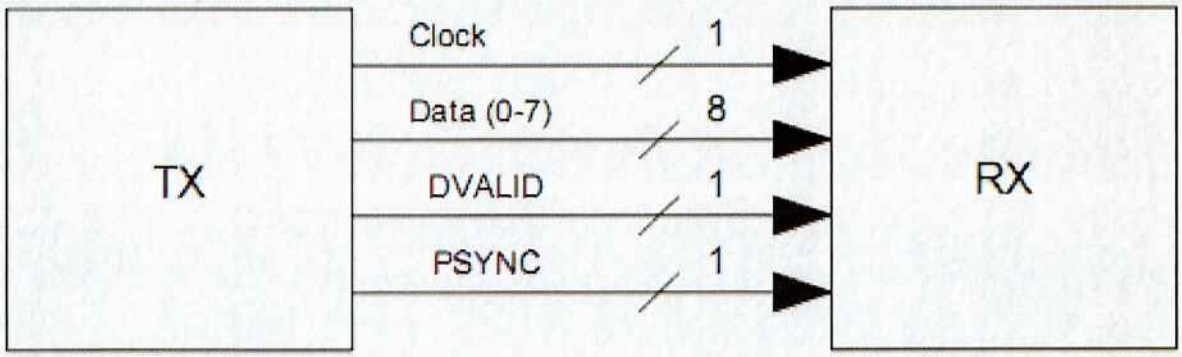

Figura 5.5: *Interface de transmissão paralela (SPI)*

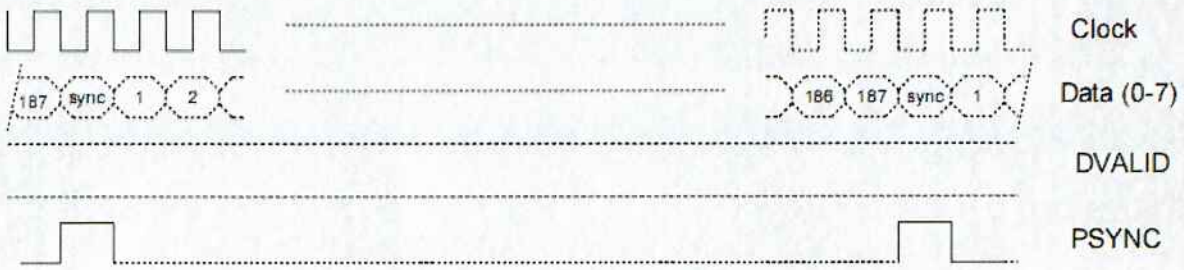

Figura 5.6: *Formato de transmissão de pacotes de 188 bytes*

Este sinal irá então ser ativado toda vez que bytes vazios forem transmitidos para completar a quantidade total de 204 bytes.

Esta foi a interface escolhida para o projeto do dispositivo de captura USB, uma vez que podemos aproveitar a interface paralela para obtermos taxas de transmissão altas com uma baixa freqüência de clock, algo limitante no caso de uso de FPGAs e da interface USB, cuja maior freqüência de operação é 48MHz. Foi possível assim utilizar a grande quantidade de pinos de I/O disponíveis no Kit de Desenvolvimento para a implementação da interface paralela entre a FPGA e a porta paralela SPI. Além disso, o sinal de clock e bit de sincronia existente nessa interface permite um controle maior na recepção dos dados.

Desta forma, o dispositivo desenvolvido consegue capturar vídeos com taxa variável de bitrate, podendo capturar vídeos com taxas de bitrate de até 300Mbps. Durante os testes do dispositivo, foram utilizados arquivos de TS com taxas de 7, 19 e até 49 Mbps, demonstrando o funcionamento correto do dispositivo para diferentes taxas de transmissão de dados.

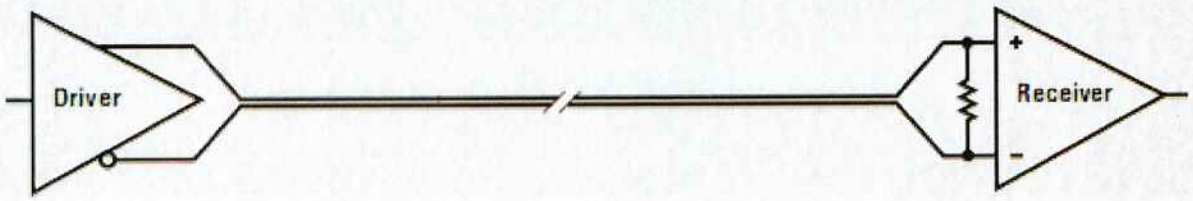

Figura 5.7: *Interconexão entre transmissor e receptor dos sinais*

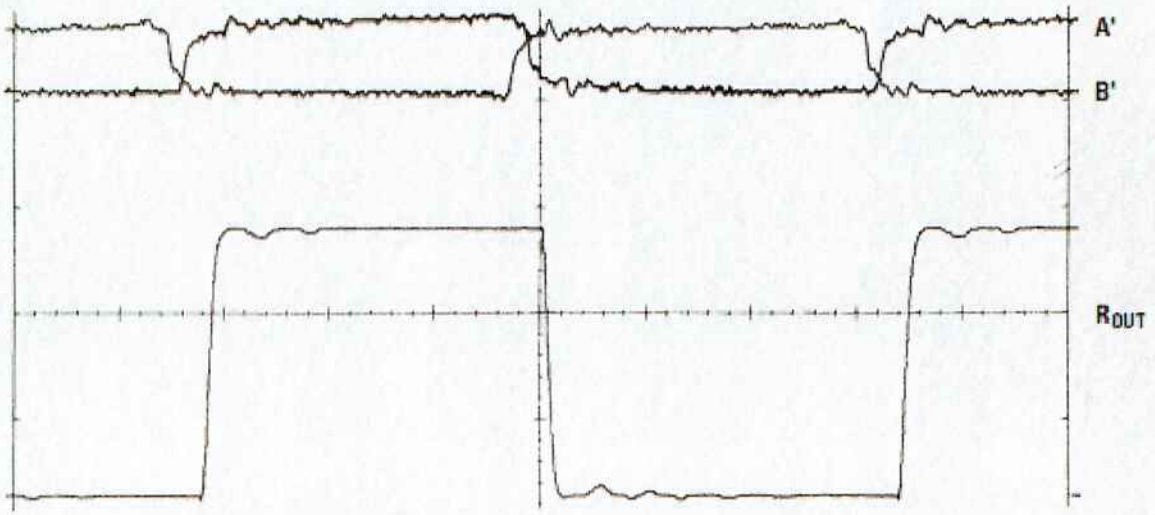

Figura 5.8: *Geração de sinal lógico a partir dos sinais analógicos*

Padrão LVDS

Ambas as interfaces de transmissão definidas anteriormente utilizam o padrão elétrico *Low-Voltage Differential Signaling* (LVDS), padrão que utiliza técnicas de circuitos analógicos de alta velocidade para prover transferência de dados de vários gigabits sobre um condutor de cobre.

Baseando-se em uma dupla de condutores, o padrão utiliza sinais de baixíssima amplitude (cerca de 180 milivolts) que reduzem a quantidade de interferência causada pela irradiação eletromagnética dos sinais. Além disso, o padrão permite a redução da potência dissipada no circuito de transmissão, uma vez que trabalha com sinais de base constante, havendo somente a variação de milivolts para a modulação dos sinais lógicos.

O sinal lógico é obtido a partir da comparação entre os dois sinais analógicos, gerando um sinal digital "1" caso a diferença seja positiva, e "0" caso contrário. O padrão foi desenvolvido principalmente para aplicações de TV Digital, que necessitam de grandes taxas de transmissão e baixo consumo de energia e simplicidade nos meios de transmissão.

Além disso, o padrão também é suportado pela maioria das FPGAs existentes no mercado, uma vez que disponibiliza uma forma de barramento de alta velocidade e baixo consumo. O padrão é utilizado também para comunicação entre unidades de processamento e memórias, uma vez que permite taxas de até gigabits e pode ser utilizado sem maiores problemas em relação à interferência eletromagnética.

5.1.4 Kit de Desenvolvimento

Para o desenvolvimento do dispositivo USB, o grupo levantou os requisitos funcionais e não-funcionais que o dispositivo deveria ter e chegou-se a conclusão que um Kit de Desenvolvimento USB seria o ideal para a implementação do projeto, uma vez que a maioria dos Kits disponíveis no mercado são altamente flexíveis e permitem que projetos sejam desenvolvidos rapidamente.

Assim, após a pesquisa de diversos modelos e fabricantes, concluiu-se que o melhor kit para o desenvolvimento do projeto era o kit XEM 3001 da Opal Kelly, empresa americana que desenvolve kits para desenvolvimento com FPGAs. Este modelo possui todos os componentes necessários para o desenvolvimento do projeto, além de apresentar um dos menores custos no mercado. Apesar de seu custo reduzido, o grupo teve grandes problemas em importá-lo, uma vez que o processo de desembaraço alfandegário é demorado e caro. As taxas de importação aumentaram em 100% o valor do Kit, algo que fora ignorado inicialmente nas projeções dos custos do projeto.

O kit integra diversas funcionalidades, entre elas um controlador USB 2.0 da Cypress, um gerador de clock, a FPGA Spartan-3 da Xilinx como dispositivo de processamento e pinos para entrada e saída de sinais conectados diretamente aos pinos de I/O da FPGA. Na Figura 5.10 é possível verificar a interligação dos componentes disponíveis no kit XEM 3001.

O controlador USB 2.0 é o ideal para a aplicação em questão, uma vez que para a captura do TS é necessária uma alta taxa de transmissão de dados entre o dispositivo e o computador host. Além disto, o controlador já está interconectado aos pinos de configuração da FPGA, sendo que o software incluído no kit permite uma rápida transferência dos arquivos de programação da FPGA do computador host para o dispositivo.

A FPGA da família Spartan-3 da Xilinx utilizada é do modelo XC3S400-4PQ208, que já possui internamente 16 blocos de RAM para o desenvolvimento de componentes de memória, algo crítico para o projeto, uma vez que é necessário o armazenamento temporário da informação no dispositivo, até a efetuação da leitura por parte do host. Além disso, as FPGAs desta família são destinadas justamente ao processamento de sinais

Figura 5.9: *Kit de Desenvolvimento XEM3001 da Opal Kelly*

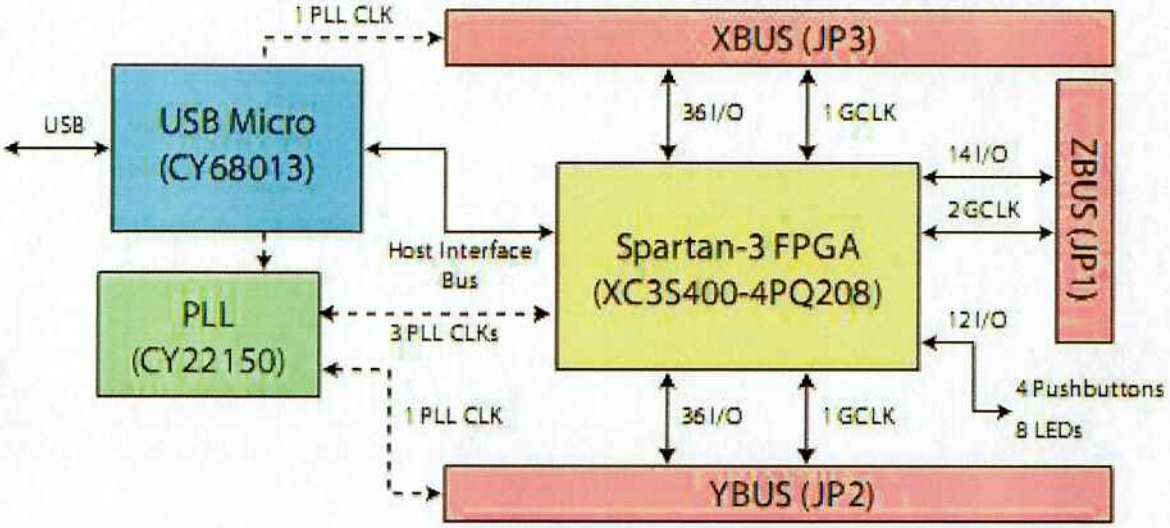

Figura 5.10: *Diagrama dos blocos funcionais do Kit de Desenvolvimento XEM3001*

multimídia, devida a alta freqüência de operação e flexibilidade nos padrões dos sinais de entrada e saída. Devido ao suporte a entradas LVDS, foi possível interconectar a interface SPI do gerador de TS diretamente aos pinos de I/O da FPGA, algo que reduziu bastante o tempo de desenvolvimento do projeto e garantiu melhores resultados.

Além disso, o kit possui drivers e APIs simples que permitem executar funções básicas de intercomunicação USB entre o computador host e o dispositivo. Desta forma, foi possível utilizar parte destas funcionalidades no desenvolvimento do projeto, adaptando os comandos de controle e leitura de dados via USB para utilizar ao máximo os recursos do kit. É possível obter maiores informações sobre as interfaces do kit em [25].

Por possuir também um tamanho reduzido, o kit garante um dos objetivos do projeto, que é o desenvolvimento de um dispositivo pequeno e portátil, que permita que o usuário o transporte e utilize de maneira simples e intuitiva. Além disso, o kit possui drivers para diferentes sistemas operacionais, algo que garante a portabilidade do dispositivo entre diferentes sistemas operacionais. Durante o desenvolvimento, o projeto foi testado com o Microsoft Windows XP e o Apple Mac OS X 10.4.

5.1.5 Implementação VHDL

Esta seção apresenta a arquitetura que foi utilizada na implementação lógica em VHDL do dispositivo de captura de vídeo USB. O maior desafio do grupo foi desenvolver uma arquitetura que permitisse que os sinais analógicos da interface SPI fossem corretamente amostrados pelo computador host via interface USB, algo nada trivial, uma vez que as freqüências de transmissão de dados pela interface SPI e pela interface USB são bastante distintas, sendo que a interface USB trabalha melhor com transferência de grandes blocos de informação.

Adicionalmente, foi necessário integrar e adaptar a arquitetura aos componentes de interface do kit, garantindo a comunicação entre a FPGA e o controlador USB, o gerador de clock e os pinos de entrada e saída disponíveis no kit. Além disso, foi necessário desenvolver toda a lógica de controle e leitura do dispositivo pelo host, uma vez que todo o controle do dispositivo deve ser feito pela interface USB.

A Figura 5.11 apresenta o diagrama de interconexão entre os componentes desenvolvidos. A partir dos sinais de entrada analógicos, foram utilizados módulos para conversão dos sinais positivo e negativo analógico para um sinal digital. Assim, todos os sinais de dados, clock e bits de sincronização estão interconectados a módulos de conversão LVDS. Estes sinais estão interconectados ao módulo FIFO desenvolvido especialmente para a aplicação e à unidade de controle, que por sua vez está interconectada aos sinais de cont-

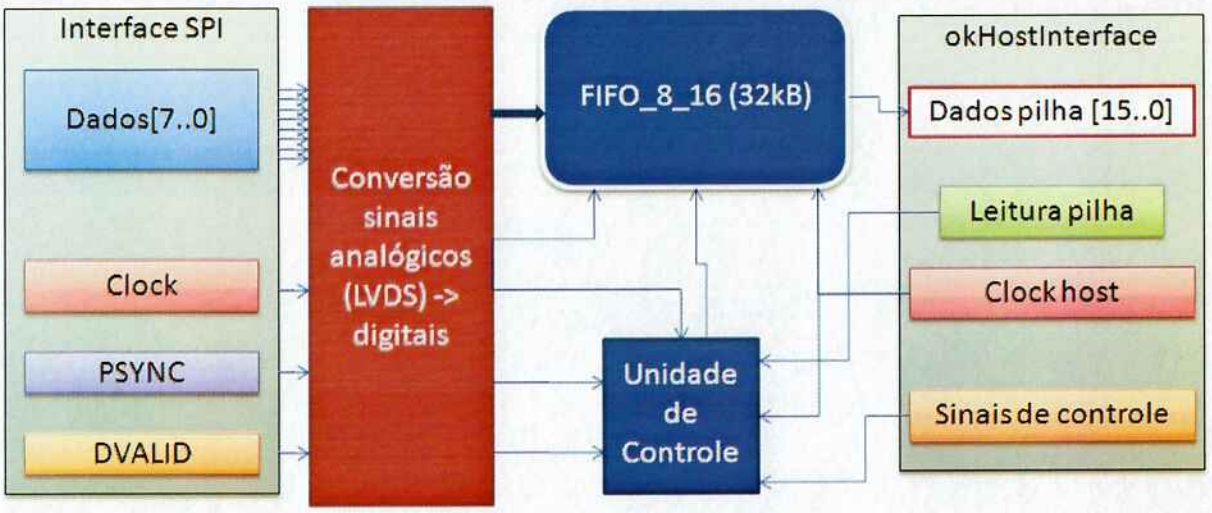

Figura 5.11: *Visão geral da arquitetura lógica desenvolvida em VHDL*

role disponíveis pela interface USB do dispositivo, dando acesso aos sinais de controle do computador host. A seguir, estão detalhados os funcionamentos de cada um dos módulos desenvolvidos.

Como IDE de desenvolvimento, foi utilizada a versão gratuita da IDE de desenvolvimento da Xilinx, o Xilinx ISE 9.2i WebPack, disponível no site da Xilinx. Esta IDE possui todas as ferramentas para edição, simulação e geração do código de execução a ser carregado no FPGA para a execução da lógica implementada. Apesar de ser bastante flexível, foi necessário simular a lógica do projeto sem considerar os sinais de entrada como analógicos, pois não há suporte para tal tipo de sinal no ambiente de simulação. Assim, foi possível simular somente o tratamento lógico dos dados de entrada e saída dos módulos desenvolvidos, sendo que cada unidade foi testada separadamente e depois foram realizadas as simulações do módulo integrado.

Unidade FIFO

Inicialmente, baseando-se nos recursos do dispositivo, foi implementada uma arquitetura que utilizava os módulos de memória RAM existentes na FPGA como módulos de memória simples, nos quais os endereços de memória eram controlados pela unidade de controle e era disparado um trigger ao host USB toda vez que metade da memória RAM estivesse completa. Assim, toda vez que metade da memória disponível (16kB) fosse escrita, a unidade de controle ficaria responsável por notificar o host e fazer a requisição de um ciclo de leitura. Assim o host ficaria inativo até receber tal requisição. Esta arquitetura não funcionou corretamente, uma vez que cada ciclo de ativação de leitura do host demor-

ava muito para ser iniciado, e assim triggers sucessivos eram perdidos, fazendo com que parte do vídeo amostrado fosse descartado.

Para corrigir tal problema, foi necessário projetar uma arquitetura que mantivesse o ciclo de leitura do host sempre ativo, de forma que não houvesse a necessidade de acionar a leitura baseada em triggers. Assim, foi criada uma arquitetura na qual o host efetua um loop infinito, no qual a cada iteração é realizada uma operação de leitura. Porém, desta forma, era necessário notificar o host quanta informação de fato deveria ser lida, e implementar uma unidade de controle que ficasse responsável por corrigir continuamente os endereços de leitura e escrita.

Para tanto, foi desenvolvida uma unidade de memória do tipo FIFO (First In First Out), de forma que os dados escritos a partir da interface SPI digital fossem escritos seqüencialmente na memória, permitindo posteriormente que o host obtivesse os dados também na seqüência correta. A unidade FIFO foi desenvolvida baseada nos blocos de memória disponíveis na FPGA Spartan-3 da Xilinx, de forma a utilizar totalmente os 16 blocos de 2kB disponíveis internamente à FPGA, totalizando uma unidade FIFO de 32kB. Além disso, como a largura das palavras de entrada é de oito bits e a interface de pilha disponível pela interface USB é de palavras de 16 bits, foi necessário criar uma unidade FIFO com entrada de 8bits e saída de 16, algo que afetou o tratamento do endereçamento interno do FIFO.

Além disso, a unidade foi criada de forma a trabalhar com dois clocks independentes, um para a escrita e outro para a leitura. Dessa forma, é possível escrever e ler dados em freqüências diferentes, desde que esta diferença respeite os limites de memória existentes no módulo. Este módulo portanto funciona como um buffer dos dados lidos, armazenando-os temporariamente para que a interface USB possa lê-los em forma de bloco posteriormente. A Figura 5.12 apresenta o diagrama lógico do FIFO desenvolvido para o projeto.

Pelo diagrama pode-se perceber a existência dos sinais de entrada e saída, sinais de controle de leitura e escrita, além de um contador, que informa quantas palavras estão armazenadas internamente e disponíveis para serem lidas pelo host. Este sinal é especialmente importante, uma vez que notifica o host sobre o número de bytes que ele pode requisitar em uma operação de leitura. Com este sinal, é possível ler continuamente a partir do host, aumentando assim a taxa de leitura e otimizando a utilização do FIFO.

Unidade de Controle

A unidade de controle desenvolvida é a unidade responsável pela sincronização dos sinais e controle dos sinais de leitura e escrita no FIFO, e controle da leitura de dados pela pilha da interface USB. Sua implementação foi baseada em um diagrama de estados,

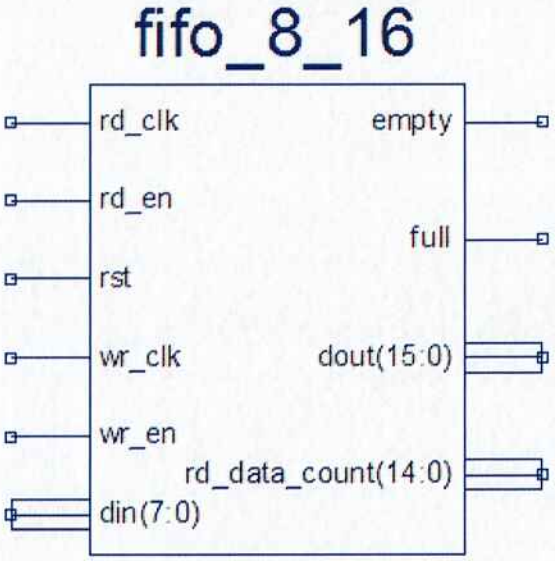

Figura 5.12: *Diagrama lógico da unidade FIFO desenvolvida*

que inicia a escrita dos dados na FIFO caso um flag START seja disparado, interrompendo assim que o flag STOP for ativado. Além disso, esta unidade filtra os pacotes de sincronia emitidos pela interface SPI, baseando-se para tanto no sinal de entrada PSYNC.

O sinal de entrada DVALID não está sendo efetivamente utilizado para o controle, uma vez que a saída do gerador de TS está configurada para utilizar a transmissão síncrona de pacotes de apenas 188 bytes. Desta forma, o sinal DVALID nunca estará ativo, pois não é necessário reconhecer bytes inválidos de preenchimento de pacotes.

A unidade de controle é responsável também por ativar os sinais de leitura da FIFO, baseando-se para isso nos sinais de controle de pilha disponibilizados pelo componente okHostInterface. Assim, a unidade de controle garante que o sinal RD-EN, responsável por permitir a leitura, esteja ativo de acordo com a requisição do host. Este controle está sincronizado com o clock de leitura do host, que funciona a uma taxa de 48MHz. Além disso, a unidade de controle efetua o tratamento de disparo de FLAGs entre o dispositivo e o host, de forma a notificar ações de leitura e quantidade de palavras disponíveis para serem lidas.

Unidade okHostInterface

Esta unidade é a responsável por efetuar o controle da interface USB disponível no kit. Ela está disponível juntamente com a documentação original do Kit e com a documentação da API baseada nos drivers do mesmo. Assim, através dos sinais de controle disponíveis

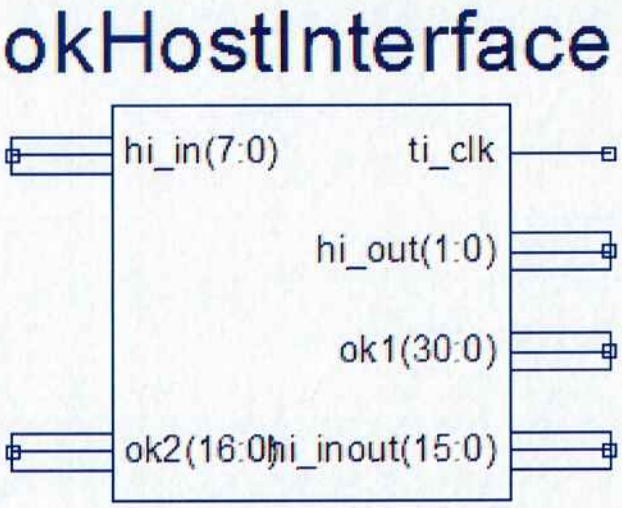

Figura 5.13: *Diagrama lógico do módulo de interface com o controlador USB*

nessa unidade, é possível efetuar operações simples de leitura de blocos de memória, bem como o disparo de flags entre o dispositivo e o host.

Esta interface define Endpoints do protocolo USB, que são como endereços do dispositivo que realizam a comunicação direta com a API disponível do kit. Existem três tipos de Endpoints para o dispositivo, citados a seguir:

- **TriggerIn e TriggerOut**: são endpoints que realizam o disparo de flags entre o dispositivo e o host. Estes triggers são sincronizados e atualizados de acordo com chamadas via API, disponibilizando informações de até 16bits. Este endpoint é o responsável pelo disparo das flags de START e STOP que controlam o início e parada da captura dos dados da interface SPI pelo dispositivo;

- **WireIn e WireOut**: são endpoints que realizam interligações virtuais entre o dispositivo e o host. Também oferecem um barramento de 16 bits e são atualizados com chamadas a funções da API, sincronizados com o clock do host. Este tipo de endpoint é utilizado para transmitir a informação de quantas palavras estão disponíveis para leitura a partir do sinal de contador do FIFO;

- **PipeIn e PipeOut**: são endpoints que permitem a leitura e escrita de blocos de memória entre o dispositivo e o host. Estes sinais oferecem um barramento de 16 bits também, sendo sincronizados com o clock do host, fornecendo flags que notificam o desejo do host de se comunicar com o dispositivo. Através deste módulo é possível

transferir blocos inteiros de memória armazenados na FIFO, uma vez que os sinais de leitura sejam devidamente controlados pela unidade de controle;

Com estes blocos funcionais simples, é possível desenvolver aplicações bastante complexas utilizando a devida lógica de controle. Assim, a partir de um endereço de endpoint, é possível ativar e ler informações a partir do código do usuário, realizando a atualização de flags e leitura de blocos da pilha.

Unidade TSReader

Esta unidade é a representação global do dispositivo, mostrando os seus pinos de entrada e saída globais utilizados na comunicação com a interface SPI. Este é o módulo principal desenvolvido em VHDL, que integra os demais blocos de FIFO, unidade de controle e okHostInterface, realizando as interligações necessárias entre os módulos.

Este módulo interliga também os pinos físicos de comunicação do kit com os pinos de entrada e saída da FPGA, de acordo com a pinagem disponível em [25]. Na Figura 5.14 temos a representação do diagrama lógico do módulo, na qual podemos verificar a existência dos pinos de entrada analógicos (pinos p e n), e a saída para os leds e interface do controlador USB.

O funcionamento básico do dispositivo se resume às seguintes etapas:

- Início da leitura: o host envia um trigger sinalizando a flag START para que o dispositivo inicie a captura dos bytes pela interface de saída digital dos sinais analógicos da interface SPI já convertidos;

- Escrita da informação: sincronizado com o clock de escrita da interface SPI, os bytes são escritos na memória FIFO, incrementando o contador de palavras disponíveis para leitura;

- Rotina de leitura: periodicamente o host atualiza os sinais de WireOut para avaliar o sinal de quantidade de palavras disponíveis. O host então executa a rotina de leitura de blocos de informação da pilha, passando como parâmetro a quantidade de bytes que ele deseja ler para que os sinais de requisição de leitura sejam ativados. A unidade de controle trata estes sinais e ativa o sinal de leitura da memória FIFO. A leitura é então sincronizada com o clock do host, que se mantém ativo durante todo o tempo.

  Após ter lido a quantidade desejada de bytes, os sinais de pilha são atualizados e a unidade de controle realiza os ajustes na memória FIFO para interromper o processo

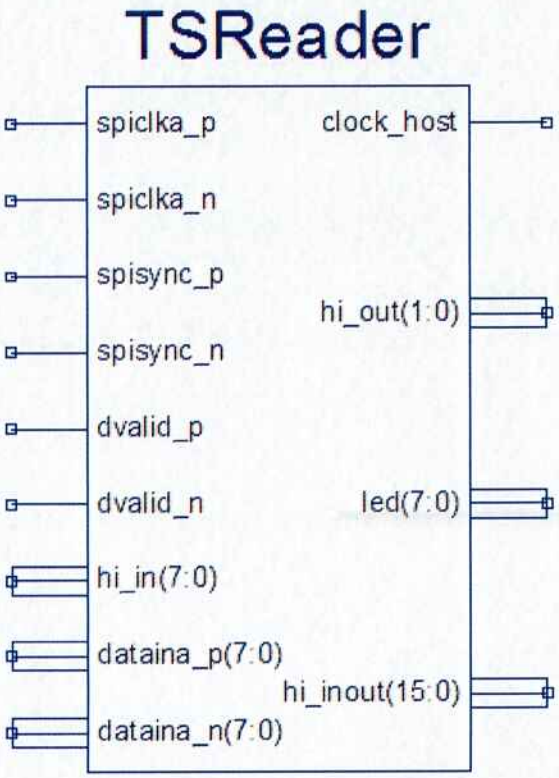

Figura 5.14: *Diagrama lógico do principal módulo do dispositivo*

de leitura. A cada palavra lida, o contador de palavras disponíveis é decrementado de duas unidades, uma vez que o barramento de saída é de 16 bits, e as palavras de entrada são de apenas 8 bits;

- Parada da leitura: o host envia outro trigger sinalizando a flag STOP para que o dispositivo pare com a captura dos bytes. Este sinal também é controlado pela unidade de controle, e atualiza os estados atuais do dispositivo para que este deixe de capturar os bytes vindos da interface SPI;

Este processo é executado indefinidamente, até que sejam efetuadas operações de parada ou reinício de leitura dos dados. Baseada nesta implementação, é possível utilizar freqüência de entrada de até 100MHz de clock, o que garante a transmissão de vídeo a uma taxa de até 100MBps. Porém, como a taxa de leitura do host não ultrapassa os 48MBps, o gargalo do dispositivo continua sendo a leitura do host via interface USB. Porém, com as taxas obtidas, foi possível transmitir praticamente todos os arquivos de vídeo disponíveis no laboratório, fossem eles de alta resolução ou não. Isto demonstra a viabilidade comercial do projeto, uma vez que ele pode ser adaptado para sua redução de custos e apresenta excelente desempenho para captura de vídeo, inclusive para vídeos de alta resolução.

5.2 Lendo os dados do dispositivo USB

A leitura dos dados via USB é feita de forma síncrona, ou seja, o programa cliente (leitor) requisita uma quantidade de bytes ao dispositivo de hardware e fica travado, esperando a resposta.

A leitura de um byte por vez é proibitiva e para minimizar o *overhead* de comunicação, a leitura dos dados é feita em blocos de tamanho variável. Como já exposto, o dispositivo de *hardware* dispõe os dados através de um *buffer*, implementado com uma fila do tipo *First In First Out* (FIFO). Desta forma, periodicamente o programa cliente pergunta ao buffer FIFO quantos bytes estão disponíveis para leitura e requisita exatamente esta quantidade de bytes. O esquema pode ser visto na Figura 5.15

Contínua e periodicamente, os bytes são então lidos e despachados para demultiplexação e decodificação; operações estas que são extremamente custosas. O programa cliente não pode ficar travado esperando os bytes serem processados para só então requisitar mais bytes, já que isto pode fazer com que o *buffer* do *hardware* estoure e dados sejam perdidos.

Figura 5.9: *Kit de Desenvolvimento XEM3001 da Opal Kelly*

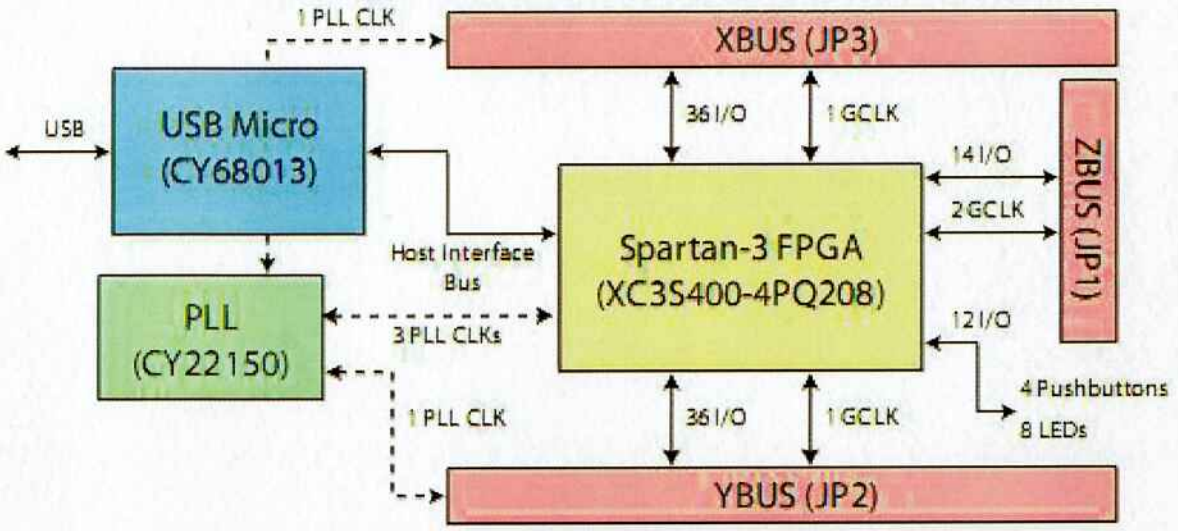

Figura 5.10: *Diagrama dos blocos funcionais do Kit de Desenvolvimento XEM3001*

multimídia, devida a alta freqüência de operação e flexibilidade nos padrões dos sinais de entrada e saída. Devido ao suporte a entradas LVDS, foi possível interconectar a interface SPI do gerador de TS diretamente aos pinos de I/O da FPGA, algo que reduziu bastante o tempo de desenvolvimento do projeto e garantiu melhores resultados.

Além disso, o kit possui drivers e APIs simples que permitem executar funções básicas de intercomunicação USB entre o computador host e o dispositivo. Desta forma, foi possível utilizar parte destas funcionalidades no desenvolvimento do projeto, adaptando os comandos de controle e leitura de dados via USB para utilizar ao máximo os recursos do kit. É possível obter maiores informações sobre as interfaces do kit em [25].

Por possuir também um tamanho reduzido, o kit garante um dos objetivos do projeto, que é o desenvolvimento de um dispositivo pequeno e portátil, que permita que o usuário o transporte e utilize de maneira simples e intuitiva. Além disso, o kit possui drivers para diferentes sistemas operacionais, algo que garante a portabilidade do dispositivo entre diferentes sistemas operacionais. Durante o desenvolvimento, o projeto foi testado com o Microsoft Windows XP e o Apple Mac OS X 10.4.

5.1.5 Implementação VHDL

Esta seção apresenta a arquitetura que foi utilizada na implementação lógica em VHDL do dispositivo de captura de vídeo USB. O maior desafio do grupo foi desenvolver uma arquitetura que permitisse que os sinais analógicos da interface SPI fossem corretamente amostrados pelo computador host via interface USB, algo nada trivial, uma vez que as freqüências de transmissão de dados pela interface SPI e pela interface USB são bastante distintas, sendo que a interface USB trabalha melhor com transferência de grandes blocos de informação.

Adicionalmente, foi necessário integrar e adaptar a arquitetura aos componentes de interface do kit, garantindo a comunicação entre a FPGA e o controlador USB, o gerador de clock e os pinos de entrada e saída disponíveis no kit. Além disso, foi necessário desenvolver toda a lógica de controle e leitura do dispositivo pelo host, uma vez que todo o controle do dispositivo deve ser feito pela interface USB.

A Figura 5.11 apresenta o diagrama de interconexão entre os componentes desenvolvidos. A partir dos sinais de entrada analógicos, foram utilizados módulos para conversão dos sinais positivo e negativo analógico para um sinal digital. Assim, todos os sinais de dados, clock e bits de sincronização estão interconectados a módulos de conversão LVDS. Estes sinais estão interconectados ao módulo FIFO desenvolvido especialmente para a aplicação e à unidade de controle, que por sua vez está interconectada aos sinais de cont-

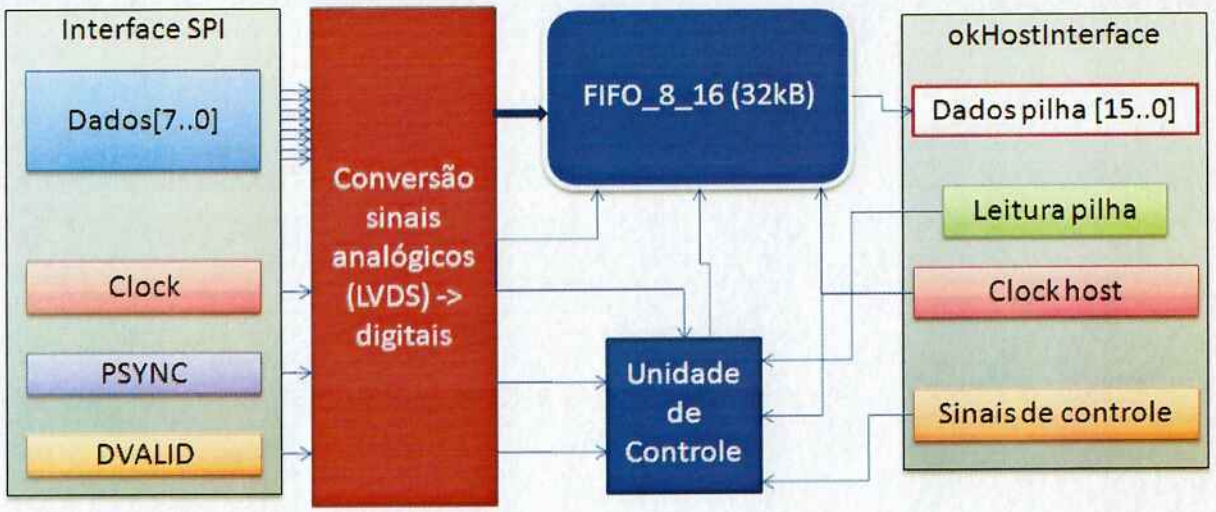

Figura 5.11: *Visão geral da arquitetura lógica desenvolvida em VHDL*

role disponíveis pela interface USB do dispositivo, dando acesso aos sinais de controle do computador host. A seguir, estão detalhados os funcionamentos de cada um dos módulos desenvolvidos.

Como IDE de desenvolvimento, foi utilizada a versão gratuita da IDE de desenvolvimento da Xilinx, o Xilinx ISE 9.2i WebPack, disponível no site da Xilinx. Esta IDE possui todas as ferramentas para edição, simulação e geração do código de execução a ser carregado no FPGA para a execução da lógica implementada. Apesar de ser bastante flexível, foi necessário simular a lógica do projeto sem considerar os sinais de entrada como analógicos, pois não há suporte para tal tipo de sinal no ambiente de simulação. Assim, foi possível simular somente o tratamento lógico dos dados de entrada e saída dos módulos desenvolvidos, sendo que cada unidade foi testada separadamente e depois foram realizadas as simulações do módulo integrado.

Unidade FIFO

Inicialmente, baseando-se nos recursos do dispositivo, foi implementada uma arquitetura que utilizava os módulos de memória RAM existentes na FPGA como módulos de memória simples, nos quais os endereços de memória eram controlados pela unidade de controle e era disparado um trigger ao host USB toda vez que metade da memória RAM estivesse completa. Assim, toda vez que metade da memória disponível (16kB) fosse escrita, a unidade de controle ficaria responsável por notificar o host e fazer a requisição de um ciclo de leitura. Assim o host ficaria inativo até receber tal requisição. Esta arquitetura não funcionou corretamente, uma vez que cada ciclo de ativação de leitura do host demor-

ava muito para ser iniciado, e assim triggers sucessivos eram perdidos, fazendo com que parte do vídeo amostrado fosse descartado.

Para corrigir tal problema, foi necessário projetar uma arquitetura que mantivesse o ciclo de leitura do host sempre ativo, de forma que não houvesse a necessidade de acionar a leitura baseada em triggers. Assim, foi criada uma arquitetura na qual o host efetua um loop infinito, no qual a cada iteração é realizada uma operação de leitura. Porém, desta forma, era necessário notificar o host quanta informação de fato deveria ser lida, e implementar uma unidade de controle que ficasse responsável por corrigir continuamente os endereços de leitura e escrita.

Para tanto, foi desenvolvida uma unidade de memória do tipo FIFO (First In First Out), de forma que os dados escritos a partir da interface SPI digital fossem escritos seqüencialmente na memória, permitindo posteriormente que o host obtivesse os dados também na seqüência correta. A unidade FIFO foi desenvolvida baseada nos blocos de memória disponíveis na FPGA Spartan-3 da Xilinx, de forma a utilizar totalmente os 16 blocos de 2kB disponíveis internamente à FPGA, totalizando uma unidade FIFO de 32kB. Além disso, como a largura das palavras de entrada é de oito bits e a interface de pilha disponível pela interface USB é de palavras de 16 bits, foi necessário criar uma unidade FIFO com entrada de 8bits e saída de 16, algo que afetou o tratamento do endereçamento interno do FIFO.

Além disso, a unidade foi criada de forma a trabalhar com dois clocks independentes, um para a escrita e outro para a leitura. Dessa forma, é possível escrever e ler dados em freqüências diferentes, desde que esta diferença respeite os limites de memória existentes no módulo. Este módulo portanto funciona como um buffer dos dados lidos, armazenando-os temporariamente para que a interface USB possa lê-los em forma de bloco posteriormente. A Figura 5.12 apresenta o diagrama lógico do FIFO desenvolvido para o projeto.

Pelo diagrama pode-se perceber a existência dos sinais de entrada e saída, sinais de controle de leitura e escrita, além de um contador, que informa quantas palavras estão armazenadas internamente e disponíveis para serem lidas pelo host. Este sinal é especialmente importante, uma vez que notifica o host sobre o número de bytes que ele pode requisitar em uma operação de leitura. Com este sinal, é possível ler continuamente a partir do host, aumentando assim a taxa de leitura e otimizando a utilização do FIFO.

Unidade de Controle

A unidade de controle desenvolvida é a unidade responsável pela sincronização dos sinais e controle dos sinais de leitura e escrita no FIFO, e controle da leitura de dados pela pilha da interface USB. Sua implementação foi baseada em um diagrama de estados,

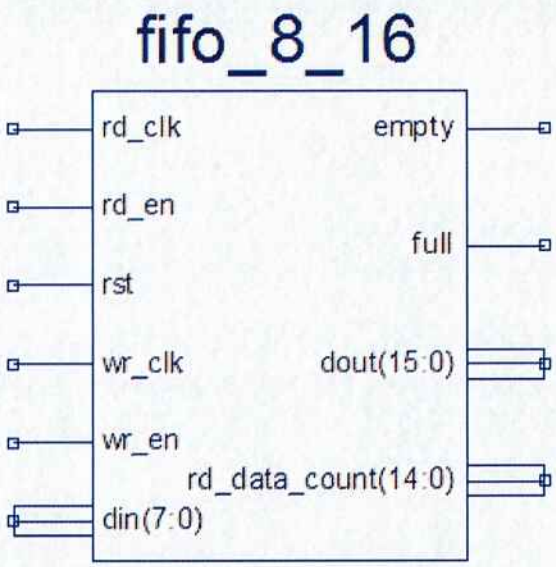

Figura 5.12: *Diagrama lógico da unidade FIFO desenvolvida*

que inicia a escrita dos dados na FIFO caso um flag START seja disparado, interrompendo assim que o flag STOP for ativado. Além disso, esta unidade filtra os pacotes de sincronia emitidos pela interface SPI, baseando-se para tanto no sinal de entrada PSYNC.

O sinal de entrada DVALID não está sendo efetivamente utilizado para o controle, uma vez que a saída do gerador de TS está configurada para utilizar a transmissão síncrona de pacotes de apenas 188 bytes. Desta forma, o sinal DVALID nunca estará ativo, pois não é necessário reconhecer bytes inválidos de preenchimento de pacotes.

A unidade de controle é responsável também por ativar os sinais de leitura da FIFO, baseando-se para isso nos sinais de controle de pilha disponibilizados pelo componente okHostInterface. Assim, a unidade de controle garante que o sinal RD-EN, responsável por permitir a leitura, esteja ativo de acordo com a requisição do host. Este controle está sincronizado com o clock de leitura do host, que funciona a uma taxa de 48MHz. Além disso, a unidade de controle efetua o tratamento de disparo de FLAGs entre o dispositivo e o host, de forma a notificar ações de leitura e quantidade de palavras disponíveis para serem lidas.

Unidade okHostInterface

Esta unidade é a responsável por efetuar o controle da interface USB disponível no kit. Ela está disponível juntamente com a documentação original do Kit e com a documentação da API baseada nos drivers do mesmo. Assim, através dos sinais de controle disponíveis

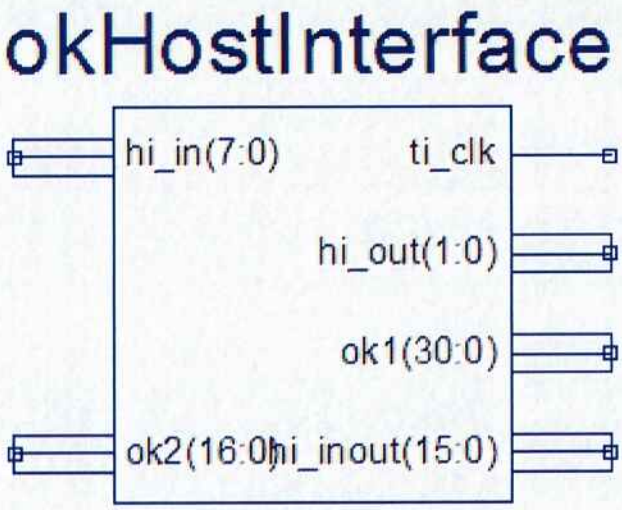

Figura 5.13: *Diagrama lógico do módulo de interface com o controlador USB*

nessa unidade, é possível efetuar operações simples de leitura de blocos de memória, bem como o disparo de flags entre o dispositivo e o host.

Esta interface define Endpoints do protocolo USB, que são como endereços do dispositivo que realizam a comunicação direta com a API disponível do kit. Existem três tipos de Endpoints para o dispositivo, citados a seguir:

- **TriggerIn e TriggerOut**: são endpoints que realizam o disparo de flags entre o dispositivo e o host. Estes triggers são sincronizados e atualizados de acordo com chamadas via API, disponibilizando informações de até 16bits. Este endpoint é o responsável pelo disparo das flags de START e STOP que controlam o início e parada da captura dos dados da interface SPI pelo dispositivo;

- **WireIn e WireOut**: são endpoints que realizam interligações virtuais entre o dispositivo e o host. Também oferecem um barramento de 16 bits e são atualizados com chamadas a funções da API, sincronizados com o clock do host. Este tipo de endpoint é utilizado para transmitir a informação de quantas palavras estão disponíveis para leitura a partir do sinal de contador do FIFO;

- **PipeIn e PipeOut**: são endpoints que permitem a leitura e escrita de blocos de memória entre o dispositivo e o host. Estes sinais oferecem um barramento de 16 bits também, sendo sincronizados com o clock do host, fornecendo flags que notificam o desejo do host de se comunicar com o dispositivo. Através deste módulo é possível

transferir blocos inteiros de memória armazenados na FIFO, uma vez que os sinais de leitura sejam devidamente controlados pela unidade de controle;

Com estes blocos funcionais simples, é possível desenvolver aplicações bastante complexas utilizando a devida lógica de controle. Assim, a partir de um endereço de endpoint, é possível ativar e ler informações a partir do código do usuário, realizando a atualização de flags e leitura de blocos da pilha.

Unidade TSReader

Esta unidade é a representação global do dispositivo, mostrando os seus pinos de entrada e saída globais utilizados na comunicação com a interface SPI. Este é o módulo principal desenvolvido em VHDL, que integra os demais blocos de FIFO, unidade de controle e okHostInterface, realizando as interligações necessárias entre os módulos.

Este módulo interliga também os pinos físicos de comunicação do kit com os pinos de entrada e saída da FPGA, de acordo com a pinagem disponível em [25]. Na Figura 5.14 temos a representação do diagrama lógico do módulo, na qual podemos verificar a existência dos pinos de entrada analógicos (pinos p e n), e a saída para os leds e interface do controlador USB.

O funcionamento básico do dispositivo se resume às seguintes etapas:

- Início da leitura: o host envia um trigger sinalizando a flag START para que o dispositivo inicie a captura dos bytes pela interface de saída digital dos sinais analógicos da interface SPI já convertidos;

- Escrita da informação: sincronizado com o clock de escrita da interface SPI, os bytes são escritos na memória FIFO, incrementando o contador de palavras disponíveis para leitura;

- Rotina de leitura: periodicamente o host atualiza os sinais de WireOut para avaliar o sinal de quantidade de palavras disponíveis. O host então executa a rotina de leitura de blocos de informação da pilha, passando como parâmetro a quantidade de bytes que ele deseja ler para que os sinais de requisição de leitura sejam ativados. A unidade de controle trata estes sinais e ativa o sinal de leitura da memória FIFO. A leitura é então sincronizada com o clock do host, que se mantém ativo durante todo o tempo.

  Após ter lido a quantidade desejada de bytes, os sinais de pilha são atualizados e a unidade de controle realiza os ajustes na memória FIFO para interromper o processo

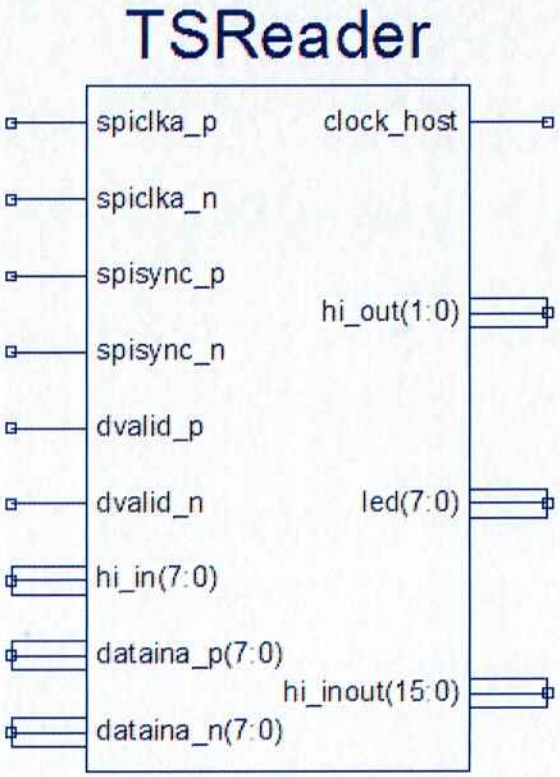

Figura 5.14: *Diagrama lógico do principal módulo do dispositivo*

de leitura. A cada palavra lida, o contador de palavras disponíveis é decrementado de duas unidades, uma vez que o barramento de saída é de 16 bits, e as palavras de entrada são de apenas 8 bits;

- Parada da leitura: o host envia outro trigger sinalizando a flag STOP para que o dispositivo pare com a captura dos bytes. Este sinal também é controlado pela unidade de controle, e atualiza os estados atuais do dispositivo para que este deixe de capturar os bytes vindos da interface SPI;

Este processo é executado indefinidamente, até que sejam efetuadas operações de parada ou reinício de leitura dos dados. Baseada nesta implementação, é possível utilizar freqüência de entrada de até 100MHz de clock, o que garante a transmissão de vídeo a uma taxa de até 100MBps. Porém, como a taxa de leitura do host não ultrapassa os 48MBps, o gargalo do dispositivo continua sendo a leitura do host via interface USB. Porém, com as taxas obtidas, foi possível transmitir praticamente todos os arquivos de vídeo disponíveis no laboratório, fossem eles de alta resolução ou não. Isto demonstra a viabilidade comercial do projeto, uma vez que ele pode ser adaptado para sua redução de custos e apresenta excelente desempenho para captura de vídeo, inclusive para vídeos de alta resolução.

5.2 Lendo os dados do dispositivo USB

A leitura dos dados via USB é feita de forma síncrona, ou seja, o programa cliente (leitor) requisita uma quantidade de bytes ao dispositivo de hardware e fica travado, esperando a resposta.

A leitura de um byte por vez é proibitiva e para minimizar o *overhead* de comunicação, a leitura dos dados é feita em blocos de tamanho variável. Como já exposto, o dispositivo de *hardware* dispõe os dados através de um *buffer*, implementado com uma fila do tipo *First In First Out* (FIFO). Desta forma, periodicamente o programa cliente pergunta ao buffer FIFO quantos bytes estão disponíveis para leitura e requisita exatamente esta quantidade de bytes. O esquema pode ser visto na Figura 5.15

Contínua e periodicamente, os bytes são então lidos e despachados para demultiplexação e decodificação; operações estas que são extremamente custosas. O programa cliente não pode ficar travado esperando os bytes serem processados para só então requisitar mais bytes, já que isto pode fazer com que o *buffer* do *hardware* estoure e dados sejam perdidos.

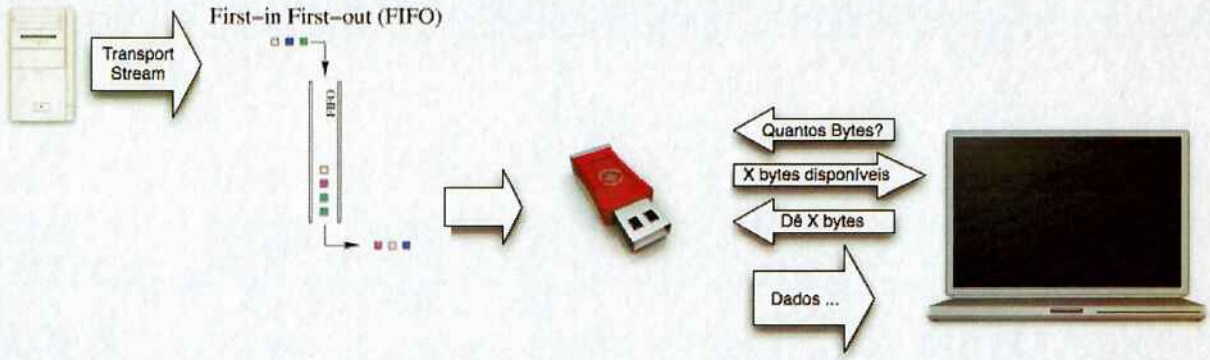

Figura 5.15: *Requisição de bytes ao FIFO pelo programa cliente*

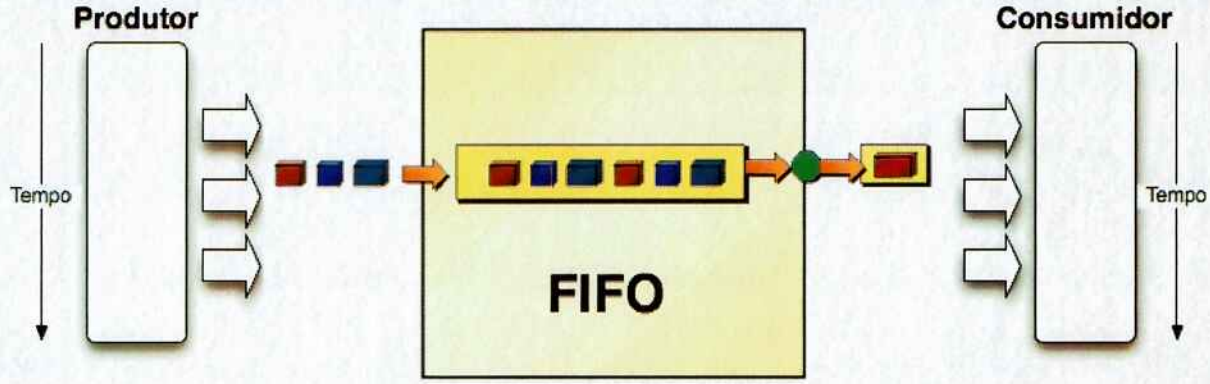

Figura 5.16: *Duas* threads *(produtora e consumidora) manipulando o* buffer

De modo a contornar este problema, a necessidade de tornar o programa concorrente ficou evidente. A concorrência leva o sistema a um nível de complexidade bem superior, mas neste caso foi necessária, já que a leitura dos dados oriundos do hardware não pode ser interrompida pelo processamento dos bytes. O programa foi dividido em duas linhas de execução (*threads*), uma responsável pela leitura contínua dos bytes via USB e outra responsável pelo processamento e demultiplexação do vídeo.

Esta divisão em duas linhas de execução é um padrão conhecido como **produtor/-consumidor**, onde uma linha de execução faz o papel de *produtora* de dados, e a outra *consome* os dados sendo responsável pelo tratamento. Os dados são transferidos através de um buffer compartilhado entre todas as linhas de execução, e para este caso foi utilizado mais um buffer implementado com uma fila do tipo FIFO, como esquematizado na Figura 5.16.

O padrão **produtor/consumidor** tem a interessante característica de permitir um número indeterminado de produtores e/ou consumidores (também conhecidos como *workers*). Para este projeto foi necessário apenas um produtor e um consumidor.

O ponto crítico em deixar o programa concorrente, foi justamente a implementação do buffer. Como existem duas linhas de execução o manipulando, pode-se dizer que é um recurso compartilhado e portanto deve ter o acesso sincronizado entre todas as *threads*, a fim de evitar os clássicos problemas de concorrência também conhecidos como condições de corrida.

Felizmente, a plataforma Java tem facilidades para programação concorrente e acesso sincronizado a recursos compartilhados. Para este fim, foi utilizada a palavra chave *synchonized* da linguagem Java nos pontos críticos de concorrência do programa. Este é um recurso da linguagem que garante o acesso exclusivo ao recurso compartilhado, através de semáforos e/ou travas (*locks*, *mutex*) e evita as condições de corrida. A desvantagem é que tem um *overhead* associado, portanto o acesso exclusivo não deve ser usado indiscriminadamente; apenas nos pontos críticos.

O *buffer* FIFO foi implementado como uma lista ligada e com acesso sincronizado nos pontos críticos de concorrência. Teoricamente tem capacidade infinita de armazenamento, porém na prática está limitado ao tamanho máximo do *heap* de objetos da máquina virtual Java que executa o programa. O tamanho padrão (32 MB) para o *heap* da máquina virtual Java mais popular, da *Sun Microsystems*, se mostrou suficiente nos diversos testes executados.

Neste ponto do projeto, foi desenvolvida uma aplicação que lê o fluxo e salva em um arquivo. Há uma infinidade de usos para a aplicação, como a gravação agendada de programas e a conversão para outros formatos de vídeo. A escrita dos bytes no arquivo pode ser mais lenta que a leitura do buffer em alguns momentos de carga elevada do computador host, porém tal problema não afeta o funcionamento da aplicação justamente pela existência do buffer FIFO, que se mostrou bastante eficiente.

O buffer desacopla a leitura do stream a partir da porta USB da escrita em um arquivo, possibilitando que cada um opere no seu ritmo, sem afetar o outro.

5.3 Demultiplexação e decodificação

O grupo iniciou esta tarefa com o objetivo de implementar um demultiplexador completo para pacotes do *MPEG Transport Stream*, que seriam então passados a um decodificador e reprodutor de vídeo compatível com Java. Inicialmente foi previsto o uso do *Java Media Framework* (JMF) associado à extensão FOBS (`http://fobs.sourceforge.net`), que permite um número maior de formatos e compressão de vídeo.

Nesta etapa do desenvolvimento, foi crucial o estudo e entendimento completo do formato dos pacotes *MPEG Transport Stream* (TS). Tipicamente, um pacote TS tem 188 bytes, é iniciado com o byte de sincronia *0x47*, contém um identificador conhecido como *Program ID* (**PID**) e encapsula um pacote do tipo *Packetized Elementary Stream* (PES).

Diversos pacotes TS com o mesmo PID formam um programa. O programa pode ser então composto por diversos *Elementary Streams* (ES), que representam as informações básicas do programa, como canais de áudio, vídeo e metadados. Para serem inclusos em pacotes TS, cada um dos *Elementary Streams* precisa ser quebrado em pacotes e para isso foi definido o padrão *Packetized Elementary Stream* (PES), que basicamente carrega informações de seqüencia e continuidade.

Durante o desenvolvimento do demultiplexador, o grupo teve contato com o *VLC media player* [21], um fantástico reprodutor de mídias capaz de demultiplexar e decodificar todos os padrões especificados pelo Sistema Brasileiro de TV Digital.

Um estudo sobre o código fonte do VLC (que é livre), permitiu concluir que seria uma alternativa viável para reprodução do *Transport Stream* lido através do dispositivo de hardware. O VLC possui o módulo *ts-demux*, que o torna capaz de entender e demultiplexar completamente os pacotes TS, e o módulo *libdvbspi*, que proporciona o processamento das tabelas do padrão europeu DVB-MHP, adotado também pelo padrão brasileiro.

O grande problema do uso do VLC é que não é escrito em Java e sim em C. Desta forma, precisa ser iniciado como um processo externo e isto uma grande dificuldade para passar os bytes lidos do hardware, já que exige a complicada comunicação entre processos (IPC). Há uma alternativa sendo desenvolvida pelo time do VLC conhecida como jVLC, que nada mais é do que uma forma facilitada de iniciar o VLC usando Java, através da *Java Native Interface* (JNI).

Mesmo o jVLC não resolve a dificuldade de comunicação entre processos, visto que não fornece uma interface simplificada para prover os bytes do vídeo ao VLC. A solução encontrada pelo grupo foi utilizar um modo do VLC que lê os dados da sua entrada padrão. O programa em Java que lê os bytes do hardware USB inicia um processo VLC externo e passa os bytes lidos através de sua entrada padrão. Esta técnica é largamente utilizada pelos sistemas operacionais baseados em UNIX e é conhecida como ***pipe*** entre processos. A solução se mostrou muito satisfatória nos diversos sistemas operacionais testados: Windows XP, Mac OS X 10.4 e Linux.

Figura 5.17: *Reprodução de um* Transport Stream *lido através do USBTv no Ginga*

5.4 Adequando ao Ginga

A última tarefa relativa à exibição do vídeo no computador através da porta USB foi integrar a exibição do *Transport Stream* lido pelo dispositivo USBTv ao middleware para a TV Digital brasileira: o Ginga.

O processo para embutir o reprodutor de *Transport Streams* no Ginga começou com um estudo profundo sobre a implementação do Ginga e da linguagem *Nexted Context Language* (NCL) desenvolvida pelo grupo Telemídia, da PUC-RJ. O Ginga-NCL, como é popularmente conhecido tem uma arquitetura flexível para a inclusão de novos tipos de mídia suportadas e novos reprodutores para a exibição destas mídias.

O primeiro passo foi estudar como funciona o suporte aos formatos de vídeo atuais. O Ginga-NCL permite associar um tipo de arquivo (*mime type*) a uma classe responsável pela sua exibição (*Player*, ou *Adapter*) de forma simples, através de um arquivo de configuração: *mimedefs.ini*, que se encontra na pasta *gingaNclConfig/players*. Um exemplo é o suporte a vídeos do tipo MPEG-2 através do *Java Media Framework*; para tal existe no arquivo de configuração a seguinte entrada: *video/mpeg=pacote.JmfVideoPlayerAdapter*.

O suporte completo ao USBTv no Ginga, foi incluso em duas etapas. O objetivo inicial foi tornar o Ginga capaz de reproduzir *Transport Streams* salvos no computador (um

```
<body>

  <!--+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  !  PONTO DE ENTRADA:
  !  indica o componente onde o programa inicia
  !+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++-->

  <port id="pInicio" component="video1"/>

  <!--+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  !  MÍDIAS:
  !  define o local dos arquivos de mídia e as associa com seus descritores
  !+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++-->
  <media type="video/mpeg-ts" id="video1" src="file://MARIA-TS.ts" descriptor="dVideo1"/>
```

Figura 5.18: *Definição de mídia video/mpeg-ts em NCL para ser reproduzida com o VLC*

```
<media type="video/usbtv" id="video1" descriptor="dVideo1"/>
```

Figura 5.19: *Definição de mídia video/usbtv em NCL*

arquivo no formato MPEG-TS disponível no sistema de arquivos). Na etapa posterior foi adicionado o suporte de leitura através da porta USB para a reprodução ao vivo de *Transport Streams*.

O primeiro novo tipo de mídia criado foi o *video/mpeg-ts* e a ele foi associada à classe Java *VlcPlayerAdapter*, responsável por ler os bytes do arquivo e iniciar um processo do VLC para reproduzi-lo. A Figura 5.18 mostra a definição de uma mídia deste tipo em um documento NCL. Nesta etapa, grande parte da integração com o Ginga foi resolvida, já que o VLC passa a ser iniciado através da execução de um documento NCL e de dentro do middleware para TV Digital. Além disso, configurações do video como tamanho e volume do som, passam a ser definidas no documento NCL e interpretadas pelo próprio Ginga.

O segundo novo tipo de mídia criada, finalizando a integração do USBTv com o Ginga, foi o *video/usbtv*. A este tipo de mídia foi associada uma classe desenvolvida pelo grupo (*UsbTvPlayerAdapter*), responsável por iniciar o processo do VLC, pela leitura dos dados através da interface USB, e pelo controle da concorrência e armazenamento em buffer já discutido. Para fazer com que um *Transport Stream* sendo lido pelo dispositivo USBTv seja exibido em uma aplicação para TV Digital, basta agora incluir uma mídia do tipo *video/usbtv* no documento NCL, como mostra a Figura 5.19.

Figura 5.20: *O popular jogo Space Invaders*

5.5 Aplicação com Interatividade Local

De modo a explorar as capacidades e possibilidades do Ginga, foi implementada uma aplicação para a TV Digital que explora a interatividade local (sem canal de retorno).

O conhecido e antigo jogo *Space Invaders* (Figura 5.20), originalmente criado no Japão, foi portado para rodar sobre o middleware para TV Digital como forma de exploração do Ginga e com o objetivo de aprofundamento nas especificações para a interatividade.

Os aspectos sobre a apresentação como posicionamento e tamanho, além de toda a interação com o usuário (resposta às teclas do controle remoto) foram definidos através de um documento NCL. O restante do funcionamento do jogo foi desenvolvido usando a parte procedural do middleware, em Java.

Um dos problemas enfrentados pela comunidade de desenvolvedores do Ginga é a falta de exemplos de aplicações para a TV Digital que misturem aspectos da parte declarativa (Ginga-NCL) e da parte procedural (Ginga-Java). O *Space Invaders* para a TV Digital poderá ser fornecido como exemplo de integração entre as duas partes do Ginga.

Um documento NCL pode incluir mídias do tipo *application/x-ginga-NCLet*, que são classes Java a serem utilizadas pela aplicação. Caso o arquivo da mídia tenha a extensão *.class*, não é obrigatória a definição explícita do tipo e se a classe em questão ainda implementar a interface *javax.tv.xlet.Xlet*, o Ginga cuida do seu ciclo de vida. Há ainda outra forma de embutir código procedural em um documento NCL, através de mídias *application/x-ginga-NCLua*, que devem conter código na linguagem Lua [22].

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ncl id="invaders" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">
  <head>
    <descriptorBase>
      <descriptor id="gameDescriptor" region="gameRegion">
        <descriptorParam name="x-classpath" value="./"/>
      </descriptor>
    </descriptorBase>
  </head>

  <body>
    <port id="pInicio" component="game"/>

    <media id="game"
      src="br/usp/poli/usbtv/invaders/SpaceInvaders.class"
      descriptor="gameDescriptor" />
  </body>
</ncl>
```

Figura 5.21: *Exemplo de uso de classes Java em um documento NCL*

```
<media id="game" src="br/usp/poli/usbtv/invaders/SpaceInvaders.class">
  <property name="start"/>
  <property name="fire"/>
  <property name="leftPressed"/>
  <property name="rightPressed"/>
</media>
```

Figura 5.22: *Expondo métodos da classe como propriedades em um documento NCL*

As mídias podem conter um conjunto de classes Java, contanto que indiquem a classe inicial a ser executada através do atributo *src* do nó *media* e especifiquem o *classpath* contendo todas as outras classes a serem utilizadas, através de descritores NCL. Um exemplo pode ser visto na Figura 5.21

Para mídias que contenham classes Java também é possível definir propriedades desta classe (no padrão *JavaBeans*, com accessors e modificadores) para que sejam manipuladas pelos elementos NCL. No *Space Invaders*, alguns métodos da classe foram expostos como propriedades (Figura 5.22) para serem chamados em eventos durante a execução do documento NCL (como o pressionamento de uma tecla do controle remoto).

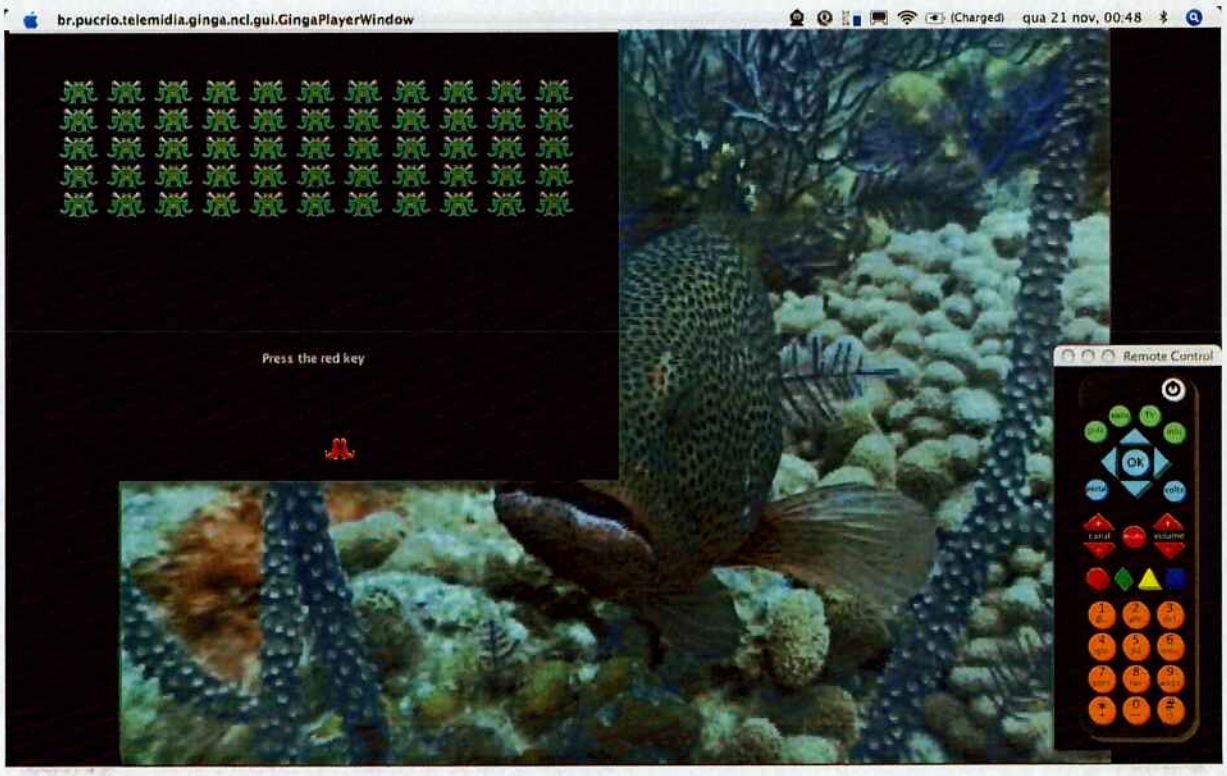

Figura 5.23: *Space Invaders rodando no Ginga, junto com a exibição de um TS*

A parte procedural - Ginga-Java - ainda não está publicamente disponível devido a restrições sobre uso de software de terceiros, porém a parte em Java da aplicação aqui descrita não faz uso de nenhum recurso específico do padrão brasileiro, podendo ser utilizada em qualquer middleware de TV Digital que siga os padrões do Java TV (inclusos todos aqueles que seguem o GEM). Assim que o middleware Ginga-Java for disponibilizado, aplicações Java com mais recursos poderão ser desenvolvidas com maior simplicidade.

5.6 Uso do canal de retorno

Um dos grandes desafios da implantação da interatividade completa no Sistema Brasileiro de TV Digital é a definição do canal de retorno. A tecnologia já oferece diversas alternativas, porém ainda não se sabe exatamente qual é a mais viável para a situação brasileira.

Algumas empresas de TV por assinatura já fornecem set-top boxes com um modem interno, que possibilita o uso de um canal de retorno via conexão discada. Muito se fala do uso das redes de telefonia celular como o GPRS para transferência de dados e conexões permanentes através de Wi-Fi, Ethernet ou WiMax começam a surgir como opções promissoras. Além disso, há um cenário de convergência tecnológica no país; muitas fornecedo-

Figura 5.24: *Produtos relacionados a emagrecimento, em um filme sobre obesidade*

ras de TV a cabo e Internet prometem integrar em um só aparelho o set-top box e um modem (ADSL ou cable modem).

O **USBTv** traz a TV Digital para dentro dos computadores de uso pessoal, que naturalmente já possuem uma conexão a Internet. Tal fato não pode ser ignorado, já que a constante conectividade que pode ser associada aos computadores traz uma enorme facilidade para a implementação do canal de retorno, através da Internet.

De modo a explorar esta possibilidade, foi desenvolvida uma aplicação interativa para a TV Digital, compatível com o middleware brasileiro, Ginga. A aplicação busca aproveitar a conexão provida pelo canal de retorno para acessar serviços disponíveis na Internet. Neste caso específico, informações sobre o vídeo sendo exibido (metadados) são utilizadas para a consulta de produtos relevantes no popular site de comércio eletrônico Mercado Livre (`http://www.mercadolivre.com.br`).

Foram feitos testes com a aplicação rodando junto a um vídeo sobre obesidade. A aplicação exalta a opção de interatividade exibindo um ícone com a cor da tecla do controle remoto adequada. A qualquer momento enquanto este ícone estiver sendo exibido, o espectador pode usar o seu controle remoto para ativar a pesquisa de produtos relacionados a emagrecimento. Uma foto da aplicação pode ser vista na Figura 5.24.

Para a consulta dos dados no Mercado Livre, a aplicação se beneficia do suporte ao protocolo HyperText Transfer Protocol (HTTP) previsto no padrão brasileiro de TV Digital. De fato, tal suporte já está previsto no GEM, porém mesmo que o set-top box não o suporte, poderia ser facilmente implementado pela própria aplicação, já que o set-top box fornece no mínimo o suporte ao protocolo TCP.

Uma requisição HTTP é feita ao serviço oferecido pelo Mercado Livre. A esta requisição, está associado um parâmetro que indica quais serão as palavras relativas à pesquisa. O resultado é um texto contendo informações sobre produtos e categorias relacionados às palavras da pesquisa, no formato XML.

A aplicação então processa este XML, exibindo as informações em um formato adequado a realidade da TV. Nesta etapa do trabalho houve desafios interessantes. O processamento do XML teve de ser feito com muita cautela, levando sempre em conta as restrições impostas pelo set-top box, que costuma ser um hardware simples e com baixo poder de processamento.

Com isto em mente, o processamento do XML teve de ser feito utilizando apenas as APIs disponíveis em um ambiente de TV Digital (especificadas primariamente no padrão Java TV, que o padrão brasileiro Ginga adota). Além disso, para que o processamento fosse extremamente rápido e leve, a equipe baseou-se no padrão *XML Pull Parsing* [27], implementado pela biblioteca XPP3 [28]. Tal biblioteca é adequada para ser usada em pequenos dispositivos já que é rápida, leve e eficiente.

Como o Ginga-Java, responsável por definir como será feito o carregamento de bibliotecas de terceiros por aplicações de TV Digital ainda não foi publicamente lançado, o uso de uma biblioteca leve como o XPP3 minimiza o risco, já que pode ser facilmente integrada à aplicação e tem o código fonte disponível. Desta forma, caso necessário, não precisará mais ser carregada como uma biblioteca (arquivo JAR) extra.

A apresentação dos dados na tela trouxe ainda outros grandes desafios. O primeiro relacionado à usabilidade de aplicações para a TV Digital, que exige atenção especial já que tem restrições de processamento gráfico e o controle remoto é a única interface para interação com o usuário. Cores, fontes e disposição dos elementos na tela foram cuidadosamente estudados para levar uma boa experiência ao usuário e não fazer com que a aplicação destoe daquilo que o usuários está habituado a ver em uma TV.

Nas primeiras versões da aplicação, todo o carregamento dos dados era feito de forma síncrona. A aplicação ficava travada, esperando até que todos os dados fossem carregados. Isto fez com que a experiência do usuário fosse terrível, já que a aplicação não respondia prontamente aos seus comandos e nunca indicava o que estava acontecendo.

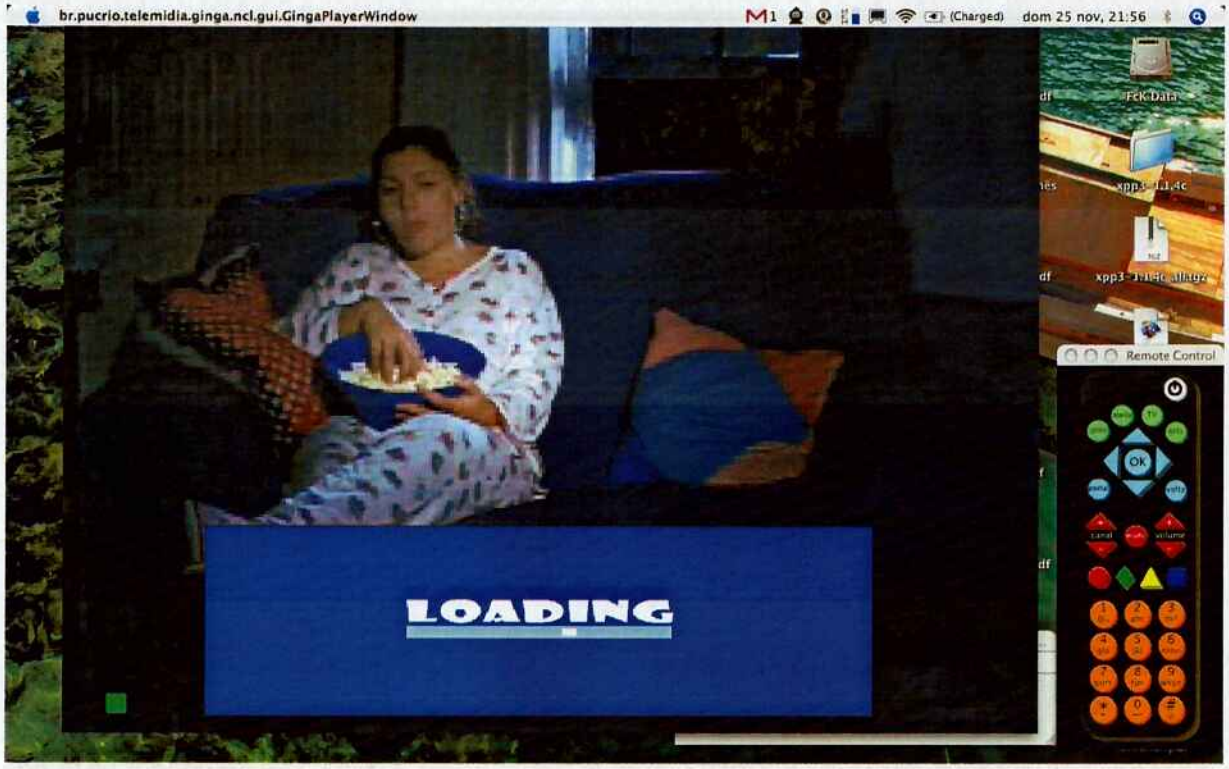

Figura 5.25: *Dados sendo carregados de forma assíncrona*

O carregamento assíncrono dos dados ficou evidente. Mais uma vez, foi utilizada a programação concorrente e desta vez para melhorar a experiência do usuário. O carregamento dos dados é feito em uma linha de execução (thread) separada; a thread principal da aplicação fica exibindo uma animação indicando que os dados estão sendo carregados, provendo feedback ao usuário, como mostra a Figura 5.25.

Assim que os dados terminam de ser carregados, a thread que os carrega substitui a imagem de carregamento por um painel que mostra dados sobre os produtos relacionados aos metadados incluídos no vídeo.

No XML de retorno do Mercado Livre estão também inclusos endereços para imagens associadas a cada um dos produtos. A aplicação utiliza novamente o canal de retorno através da Internet para carregar cada uma dessas imagens. Isto trouxe mais um grande problema a ser enfrentado já que as imagens costumam ter tamanhos relativamente grandes se comparadas ao texto XML com a resposta. O tempo para carregamento de todas as imagens fazia com que demorasse muito até que o usuário pudesse ver as informações sobre os produtos.

Desta forma, a concorrência novamente precisou ser empregada. Usando o mesmo raciocínio de carregamento assíncrono o texto da resposta é exibido primeiro e diversas

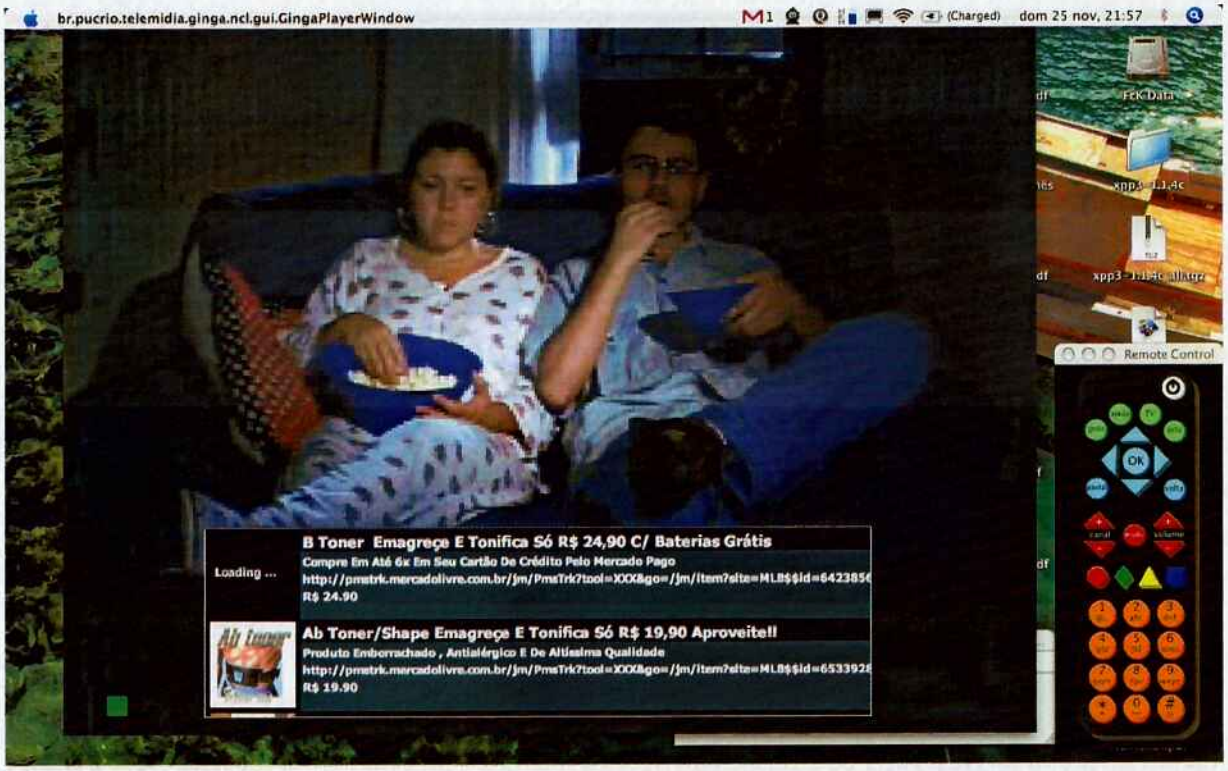

Figura 5.26: *Imagens sendo carregadas pouco a pouco por threads separadas*

threads separadas vão carregando as imagens sob demanda, que pouco a pouco vão aparecendo. O usuário pode ir analisando o resultado da pesquisa enquanto espera as imagens serem carregadas. O exemplo das imagens sendo carregadas sob demanda pode ser visto na Figura 5.26.

Assim como na aplicação *Space Invaders*, para esta aplicação de consulta ao Mercado Livre foram empregados conceitos de ambas as partes do middleware brasileiro Ginga; declarativa e procedural. Esta aplicação também pode servir de exemplo de como integrar as funcionalidades providas pelas duas abordagens e quais tarefas cada uma delas se mostra mais adequada a resolver.

Capítulo 6

TESTES E AVALIAÇÃO

6.1 Hardware

Durante o desenvolvimento do projeto de hardware, foram utilizadas IDEs de simulação de circuitos lógicos, para validar se a lógica da unidade de controle estava correta, e se os sinais estavam sendo corretamente lidos e escritos na memória FIFO. Porém, não foi possível simular os efeitos dos sinais LVDS como entrada do dispositivo, algo que dificultou a simulação e testes do dispositivo como um todo.

Para tanto, foram colocados alguns pinos de saída com sinais intermediários da lógica interna do dispositivo, para avaliar se os sinais analógicos estavam gerando corretamente os sinais de CLOCK, PSYNC e DVALID, bem como os sinais dos dados de entrada. Para verificar se os sinais estavam sendo gerados corretamente, foi utilizado um osciloscópio fornecido pelo Departamento de Computação e Sistemas Digitais, de forma a verificar se as formas de onda e freqüências estavam de acordo com o esperado.

Durante esta fase, foram utilizados 3 diferentes tipos de TS com diferentes bitrates para a validação dos sinais e freqüências que estavam sendo transmitidas ao dispositivo. A figura 6.1 ilustra as diferentes características de cada um dos vídeos utilizados durante os testes.

Após a fase de implementação, foi necessário validar se a taxa de leitura do host estava de acordo com o esperado. Nesta fase, gastou-se grande parte do tempo de desenvolvimento do projeto até que fosse encontrada uma seqüência de chamadas da API que permitisse que o host pudesse ler os dados na freqüência adequada, evitando assim perda de pacotes do TS. Para tanto, também foram utilizados pinos de debug no dispositivo, que

Nome do Arquivo	Formato	FPS	Dimensão	Bitrate	Programas	Canais de Áudio
Shark_ATSC_420_59.ts	ATSC(59Hz)	59	704 x 480	19.155Mbps	1	1
flowerga.ts	DVB(25Hz)	25	720 x 576	19.155Mbps	3	3
trans_qu.ts	DVB(25Hz)	25	720 x 576	19.155Mbps	3	3
MHP_INTRO.ts	DVB(25Hz)	25	720 x 576	5Mbps	1	1
SHARK_DVB_1280_720_59HZ.ts	DVB(59Hz)	59	1280 x 720	19.155Mbps	1	1
Shark_ATSC_720_29.ts	ATSC(29Hz)	29	1280 x 720	19.155Mbps	1	1
SHARK_DVB_1920_1080_29HZ.ts	DVB(29Hz)	29	1920 x 1080	19.155Mbps	1	1
Shark_ATSC_720_59.ts	ATSC(59Hz)	59	1280 x 720	19.155Mbps	1	1
maria.ts	DVB(25Hz)	25	720 x 576	7Mbps	1	1

Figura 6.1: *Arquivos de Transport Streams utilizados nos testes*

Vídeo	Pacotes Esperados	Pacotes Aceitos	% Pacotes Aceitos	Bytes Aceitos	Pacotes perdidos por descontinuidade	% Pacotes perdidos por descontinuidade
Shark_ATSC_720_59.ts	696.188	566.254	81,34	106.455.752	2902	0,42
flowerga.ts	662.740	575.292	86,81	108.154.896	1815	0,27
trans_qu.ts	428.232	378.234	88,32	71.107.992	59	0,01
MHP_INTRO.ts	205.083	183.652	89,55	34.526.576	212	0,10
SHARK_DVB_1280_720_59HZ.ts	469.820	425.884	90,65	80.066.192	715	0,15
Shark_ATSC_720_29.ts	590.097	538.967	91,34	101.325.796	1621	0,27
SHARK_DVB_1920_1080_29HZ.ts	342.758	320.566	93,53	60.266.408	908	0,26
Shark_ATSC_420_59.ts	418.561	394.634	94,28	74.191.192	567	0,14
maria.ts	836.555	813.440	97,24	152.926.720	1287	0,15

Figura 6.2: *Resultado dos testes efetuados*

permitiam avaliar a freqüência com que o sinal de leitura do host estava ativo. Isto permitiu avaliar os problemas de leitura que ocorreram no início de desenvolvimento do projeto, o que acarretava com falhas na visualização do vídeo por perdas de pacotes.

Para validar a leitura dos vídeos, foram utilizados novamente os vídeos apresentados na figura 6.1, que permitiram avaliar o correto funcionamento do dispositivo para vídeos de diferentes taxas de transmissão. Foram testadas tanto a aplicação de gravação do TS em um dispositivo de armazenamento, como a aplicação de visualização simultânea do vídeo amostrado.

Para avaliar quantitativamente a qualidade do vídeo amostrado, foi utilizada a aplicação de gravação de vídeo. Assim, foi possível salvar o vídeo amostrado em um arquivo e posteriormente analisá-lo, de forma a avaliar corretamente as características obtidas na amostragem do vídeo. Inicialmente, utilizou-se o *TSReaderLite* [33] para verificar os pacotes, porém a qualidade da informação não pareceu adequada.

Assim, foi criada uma aplicação que verifica o tamanho dos pacotes e da sua correta seqüencia dentro do TS, fornecendo os resultados obtidos na figura 6.2. Os resultados contemplam os testes para cada tipo distinto de arquivo amostrado. Nestes resultados, o número de pacotes aceitos é a quantidade de pacotes com tamanhos válidos. A quantidade de pacotes esperados foi calculada baseada na quantidade de bytes do arquivo obtido a partir da amostra, algo que contempla não só os pacotes de informação de vídeo, mas também todos os pacotes de identificação dos programas e sincronia do vídeo.

A quantidade de bytes aceitos é apenas a multiplicação do número de pacotes aceitos por 188, que no caso é o tamanho em bytes de cada pacote. Os pacotes perdidos por descontinuidade foram calculados baseados no número de seqüencia dos pacotes, o que é de certa forma impreciso, uma vez que não se pode detectar perdas de mais de 15 pacotes consecutivos para um mesmo Program ID (PID). Pacotes de vários PIDs diferentes são intercalados entre si e com pacotes nulos (de PID 0x1FFF) em um Transport Stream. Portanto, a detecção de pacotes perdidos por descontinuidade não leva em consideração a perda de pacotes nulos, uma vez que nem todos utilizam o número de seqüencia.

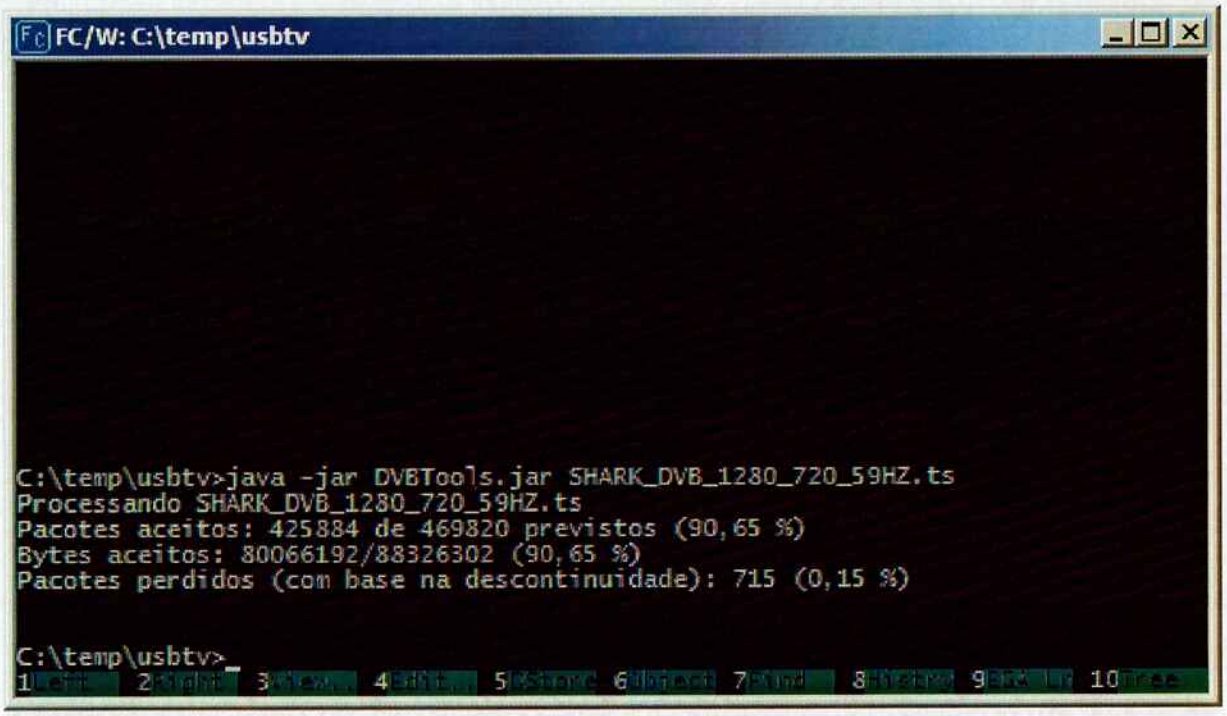

Figura 6.3: *Aplicação desenvolvida para efetuar testes nos vídeos amostrados*

Em todos os casos, a perda de pacotes não afetou a qualidade do vídeo, uma vez que somente parte destes pacotes perdidos é de fato dado útil para a visualização do vídeo. Os resultados demonstram que, para TS com maiores taxas de resolução, bitrate e número de programas dentro do mesmo TS, a perda de pacotes é mais elevada. Porém, para vídeos com taxas menores, a perda é praticamente insignificante, não afetando em nada a qualidade do vídeo exibido.

6.2 Aplicação com Interatividade Local

Para a validação da aplicação com interatividade local, o jogo *Space Invaders* foi executado na presença das diversas mídias possíveis, como vídeos simples salvos no disco rígido, vídeos lidos a partir de Transport Streams também salvos no disco e por fim junto a Transport Streams lidos através do dispositivo USBTv.

Além disso, o Space Invaders foi extensivamente jogado, para validar todas as situações possíveis.

Os testes foram executados em três dos principais sistemas operacionais do mercado: Windows XP, Mac OS X 10.4 e Ubuntu Linux 7.10.

6.3 Aplicação com Canal de Retorno

A aplicação que faz a consulta no Mercado Livre através do canal de retorno implementado sobre a conexão comum à Internet pôde ser validada comparando as pesquisas feitas com a mesma palavra chave tanto no próprio site Mercado Livre, quanto através da aplicação rodando dentro do Ginga.

Os produtos exibidos são sempre os mesmos, já que a consulta é feita por um serviço WEB que o site Mercado Livre oferece. Assim como anteriormente, estes testes foram executados nos três principais sistemas operacionais: Windows XP, Mac OS X 10.4 e Ubuntu Linux 7.10, garantindo a alta portabilidade.

Capítulo 7

CONSIDERAÇÕES FINAIS

7.1 Análise dos Resultados

O projeto obteve como resultado um dispositivo USB completamente funcional, que possibilita a captura de streams de vídeo digital de alta resolução e taxa de dados, sem perda de qualidade, seja para a gravação ou visualização imediata do vídeo. O dispositivo final é compatível e passível de ser utilizado sob qualquer sistema operacional, algo que garante a portabilidade desejada inicialmente no projeto.

Com o sucesso de sua integração como fonte de vídeo para as aplicações baseadas no middleware Ginga, o dispositivo expande as fronteiras de usabilidade do padrão brasileiro de TV digital, uma vez que disponibiliza um canal eficaz e portátil para sua visualização e aplicação. Ficam ainda pendentes as tarefas de demultiplexação de Transport Streams na estrutura do padrão brasileiro, contendo aplicações Ginga em forma de pacotes TS. Tal aplicação não foi possível de ser realizada dentro do escopo do projeto, uma vez que não havia a disponibilidade de um multiplexador eficiente para a geração de tal formato de transmissão de vídeo. O grupo estimula que trabalhos futuros englobem tal aplicação, de forma a validar o uso do dispositivo para a captura de streams no formato SBTVD que incluam aplicações interativas.

A recepção do sinal de TV Digital RF puro esteve fora do escopo deste projeto, que trata apenas do sinal MPEG2 Transport Stream, já demodulado. Futuros projetos de pesquisa poderão contemplar a parte de recepção e demodulação do sinal oferecido pelas transmissoras, que chega a casa dos espectadores. Desta forma, o projeto pode se tornar comercialmente viável, tornando o computador uma potencial alternativa a set-top boxes.

Quanto à análise e execução de aplicações interativas, fica clara a necessidade de uma atenção especial quanto a interface homem máquina e usabilidade. Foi comprovado na prática, que as aplicações devem responder prontamente as ações dos espectadores e prover feedback constante. Para tal, técnicas de concorrência, carregamento e processamento assíncronos podem ser amplamente exploradas.

Este projeto, também pôde expor alguns meios de utilização conjunta das funcionalidades oferecidas por ambas as abordagens para o middleware brasileiro para a TV Digital, Ginga. Aplicações interativas para a TV Digital completas podem ser escritas tanto com o

Ginga-NCL, quanto com o Ginga-Java. A integração, pontos de conflito e complementaridade entre os dois ainda é pouco documentada e pouco explorada pela comunidade.

As aplicações desenvolvidas durante este projeto levam a crer que a parte declarativa do Ginga é poderosa na definição dos elementos pertencentes a aplicação, as interações entre eles no espaço e no tempo, como reagirão a eventos (comandos das teclas do controle remoto, por exemplo) e a sincronia temporal e espacial. A parte procedural se mostra mais adequada para a definição do comportamento de cada um dos elementos na reação a estímulos e na especificação de suas tarefas e algoritmos.

Com o intuito de ajudar no estabelecimento de uma arquitetura padrão para a implementação de um canal de retorno e a viabilização da interatividade propiciada pela TV Digital, este grupo tem se empenhado nas pesquisas sobre as tecnologias adotadas no mundo e as em curso no país. A generalização das soluções propostas visa a fácil adaptação para quaisquer tecnologias que venham a ser adotadas no Sistema Brasileiro de TV Digital.

É desejo deste grupo que possam ser apresentadas as idéias deste projeto em convenções da área, fomentando a discussão e a análise das direções escolhidas, contribuindo assim para o desenvolvimento tecnológico do país. Como os padrões de TV Digital definidos no país ainda não são realidade, este trabalho pode não ser compatível com eventuais adaptações das especificações, porém são buscadas soluções que mais se adéqüem à realidade brasileira.

7.2 Trabalhos Futuros

O início do desenvolvimento da TV Digital no país abriu portas para todo o tipo de pesquisa, seja ela na área de equipamentos, frameworks ou aplicações destinadas a fazer crescer o uso e produção de conteúdo para a TV Digital. O projeto em questão estimula a pesquisa de novos dispositivos que permitam uma integração ainda maior com o padrão brasileiro de TV Digital, possibilitando talvez sua produção em larga escala com fins comerciais.

Em relação às aplicações interativas, é possível explorar ainda mais profundamente o uso da Internet como fonte de aplicações e serviços a serem reutilizados pela TV Digital. Uma vez que o acesso à Internet é cada vez mais amplo no país, a integração dos seus serviços já existentes agrega às aplicações de TV Digital um grande valor, expandindo assim aplicações comumente utilizadas na Internet para a TV Digital.

Este trabalho também proporciona a base para o desenvolvimento de novas formas de implementação do canal de retorno, pois este é um tópico ainda em discussão e desenvolvimento pela equipe responsável pelo middleware brasileiro GINGA. O canal de retorno e as aplicações interativas ainda não foram completamente estabelecidas para o padrão brasileiro, e até mesmo nos padrões internacionais o uso da interatividade em aplicações de TV Digital é muito restrita. Ficam abertas, portanto, diversas possibilidades de pesquisa na área.

Referências Bibliográficas

[1] BRASIL. Centro de Pesquisas e Desenvolvimento em Telecomunicações (CPqD). **MODELO DE REFERÊNCIA**: Sistema Brasileiro de Televisão Digital Terrestre. Versão PD.30.12.36A.0002A/RT-08-AB. [Campinas]: CPqD, 13 fev. 2006. (Relatório Técnico, Cliente: FUNTTEL, OS: 40539) Disponível em <http://sbtvd.cpqd.com.br/> (seção de Divulgação). Acesso em 12 ago. 2007.

[2] BRASIL. Centro de Pesquisas e Desenvolvimento em Telecomunicações (CPqD). **VISÃO DE LONGO PRAZO DA ECONÔMIA**. Versão PD.30.12.36A.0002A/RT-01-AA. [Campinas]: CPqD, 25 mai. 2004. (Relatório Técnico, Cliente: FUNTTEL, OS: 40539) Disponível em <http://sbtvd.cpqd.com.br/> (seção de Divulgação). Acesso em 10 ago. 2007.

[3] BRASIL. Centro de Pesquisas e Desenvolvimento em Telecomunicações (CPqD). **ARQUITETURA DE REFERÊNCIA**: Sistema Brasileiro de Televisão Digital Terrestre. Versão PD.30.12.34A.0001A/RT-13/AA. [Campinas]: CPqD, 10 fev. 2006. (Relatório Técnico, Cliente: FUNTTEL, OS: 40541) Disponível em <http://sbtvd.cpqd.com.br/> (seção de Divulgação). Acesso em 12 ago. 2007.

[4] BRASIL. Centro de Pesquisas e Desenvolvimento em Telecomunicações (CPqD). **ESPECIFICAÇÃO TÉCNICA DE REFERÊNCIA**: Sistema Brasileiro de Televisão Digital Terrestre. Versão PD.30.12.34A.0001A/RT-14/AA. [Campinas]: CPqD, 10 fev. 2006. (Relatório Técnico, Cliente: FUNTTEL, OS: 40544) Disponível em <http://sbtvd.cpqd.com.br/> (seção de Divulgação). Acesso em 12 ago. 2007.

[5] BRASIL. Centro de Pesquisas e Desenvolvimento em Telecomunicações (CPqD). **PLANO DE DESENVOLVIMENTO**: Sistema Brasileiro de Televisão Digital Terrestre. Versão PD.30.12.34A.0001A/RT-15/AA. [Campinas]: CPqD, 10 fev. 2006. (Relatório Técnico, Cliente: FUNTTEL, OS: 40544) Disponível em <http://sbtvd.cpqd.com.br/> (seção de Divulgação). Acesso em 12 ago. 2007.

[6] BRASIL. Centro de Pesquisas e Desenvolvimento em Telecomunicações (CPqD), Fundacão Padre Urbano Thiesen. **CARTOGRAFIA AUDIOVISUAL BRASILEIRA DE 2005**: um estudo quali-quantitativo de TV e cinema.

[Campinas]: CPqD, 2006. (Relatório de Pesquisa, Coordenadora: Prof. Dra. Cosette Castro) Disponível em <http://sbtvd.cpqd.com.br/> (seção de Divulgação). Acesso em 10 ago. 2007.

[7] ARIB Standard for Digital Broadcasting. Padrões para o sistema de TV Digital japonês. Disponível em <http://www.dibeg.org/aribstd/ARIBSTD.htm>. Acesso em 10 out. 2007.

[8] OLIVEIRA, Carina Teixeira de. **UM ESTUDO SOBRE OS PADRÕES DE *MIDDLEWARE* PARATELEVISAO DIGITAL INTERATIVA**. Fortaleza: CEFET-CE, 2007. Disponível em <http://www.gta.ufrj.br/~carina/artigos/MonografiaCarina.pdf>. Acesso em 06 out. 2007.

[9] FAIRHURST, Gorry. **MPEG-2 Overview**. Aberdeen: Department of Engineering – University of Aberdeen, 2001. Disponível em <http://erg.abdn.ac.uk/research/future-net/digital-video/mpeg2.html>. Acesso em 14 set. 2007.

[10] Advanced Television Systems Comittee Inc (ATSC). Padrões para o sistema de TV Digital norte americano. Disponível em <http://www.atsc.org>. Acesso em 13 set. 2007.

[11] Advanced Common Application Platform (ACAP). Padrões para interatividade do sistema de TV Digital norte americano. Disponível em <http://www.acap.tv>. Acesso em 13 set. 2007.

[12] OpenCable Platform (OCAP). Padrões da OpenCable para o sistema de TV Digital norte americano. Disponível em <http://www.opencable.com/ocap>. Acesso em 13 set. 2007.

[13] Digital Video Broadcasting (DVB). Padrões para o sistema de TV Digital europeu. Disponível em <http://www.dvb.org>. Acesso em 13 set. 2007.

[14] JAPÃO. Japan Broadcasting Corporation (NHK). **Outline of the Specification for ISDB-T**. [S.l.]: 1999. Disponível em <http://www.nhk.or.jp/strl/open99/de-2/shosai-e.html>. Acesso em 23 set. 2007.

[15] Sistema Brasileiro de TV Digital (SBTVD). Padrões para o sistema de TV Digital brasileiro. Disponível em <http://sbtvd.cpqd.com.br>. Acesso em 12 set. 2007.

[16] Digital Video Broadcasting Multimedia Home Platform (DVB-MHP). Padrões para interatividade do sistema de TV Digital europeu. Disponível em <http://www.mhp.org>. Acesso em 12 set. 2007.

[17] Globally Executable Multimedia Home Platform (GEM). Conjunto comuns de todos os padrões de TV Digital mundiais. Disponível em <http://www.mhp.org/mhp_technology/gem>. Acesso em 12 set. 2007.

[18] Projeto Ginga. Página do projeto do *middleware* brasileiro. Disponível em <http://www.softwarepublico.gov.br/dotlrn/clubs/ginga>. Acesso em 12 out. 2007.

[19] Universal Serial Bus (USB). Página dos padrões para a porta serial de dados. Disponível em <http://www.usb.org>. Acesso em 23 ago. 2007.

[20] GIANSANTE, M.; OGUSHI, C.M.I; MENEZES, E.; BONADIA, G.C.; GEROLAMO, G.P.B.; RIOS J.M.; PORTO, P.C.S.; HOLANDA, G.M.; DALL'ANTONIA, J.C. **Cadeia de Valor**: Projeto Sistema Brasileiro de Televisão Digital. Versão AB PD.30.12.36A.002A/RT-02-AB. Campinas: CPqD, 2004, 95 p. (Relatório Técnico, Cliente: Funttel, atividade 1236, OS: 40539).

[21] VideoLAN - VLC media player. *Free Software and Open Source video streaming solution for every OS*. Disponível em <http://www.videolan.org>. Acesso em 02 nov. 2007.

[22] The Programming Language Lua. Lua programming language official website. Disponível em <http://www.lua.org>. Acesso em 18 nov. 2007.

[23] European Committee for Electrotechnical Standardization (CENELEC). **European Standard EN 50083-9** (Interfaces for CATV/SMATV headends and similar professional equipment for DVB/MPEG-2 transport streams) Disponível em <http://www.bjpace.com.cn/data/tec/tec-DVB/DVBBlueBooksStandards/SpecificationsandStandards/interfacing/dvb-pi/En50083_9.pdf>. Acesso em 15 nov. 2007.

[24] The International Engineering Consortium (IEC). **LVDS Tutorial** Disponível em <http://www.iec.org/online/tutorials/low_voltage/>. Acesso em 15 nov. 2007.

[25] Opal Kelly. **XEM3001v2 User's Manual** Disponível em <http://www.opalkelly.com/library/XEM3001v2-UM.pdf>. Acesso em 10 out. 2007.

[26] Entry in Wikipedia, the free encyclopedia. **D-subminiature** Disponível em <http://en.wikipedia.org/wiki/D-subminiature>. Acesso em 10 out. 2007.

[27] XML Pull Parsing. **Standards for general Pull parsing** Disponível em <http://www.xmlpull.org>. Acesso em 18 nov. 2007.

[28] MXP1: Xml Pull Parser 3rd Edition (XPP3). **Uma das principais implementações dos padrões do *XML Pull Parsing*** Disponível em <http://www.extreme.indiana.edu/xgws/xsoap/xpp/mxp1/>. Acesso em 18 nov. 2007.

[29] International Organization for Standardization (ISO) & International Electrotechnical Commission (IEC). **International Standard ISO/IEC 13818 Part 1**. Information technology — Generic coding of moving pictures and associated audio information: Systems, second edition. ISO/IEC 13818-1:2000(E).

[30] Digital Video Broadcasting Project. **DVB Document A086 Rev. 6**. Transport of MPEG 2 Transport Stream (TS) Based DVB Services over IP Based Networks, 2007.

[31] HENNESSY, J.L.; PATTERSON, D.A. **Arquitetura de Computadores**: Uma abordagem quantitativa. Tradução da 3ª Edição Americana. Tradução de Vandenberg D. de Souza. Rio de Janeiro: Editora Campus, 2003, 828 p.

[32] FREGNI, E.; SARAIVA, A.M. **Projeto Lógico Digital**: Conceitos e Prática. São Paulo: Editora Edgard Blücher, 1995, 498 p.

[33] TSREADER Lite. **MPEG-2 Transport Stream Analysis and Recording** Disponível em <http://www.coolstf.com/tsreader/>. Acesso em 18 nov. 2007.

Glossário

- *ADSL*: acrônimo de *Asymmetric Digital Subscriber Line*, é uma forma de transmissão de dados de alta velocidade utilizando linhas telefônicas comuns, em freqüências maiores que os seres humanos conseguem escutar.
- *BNC*: é um tipo de conector cujo nome vem de seus criadores: "bayonet Neil-Concelman". Sua grande característica é o sistema de trava, tipo twist-lock (gira e trava), que possibilita grande segurança na conexão. É bastante utilizado nos equipamentos profissionais de vídeo.
- *Broadcast*: é um modo de difusão de sinais em que é transmitido o mesmo conteúdo para todos os receptores. Numa transmissão de TV, por exemplo, todas as pessoas sintonizadas no mesmo canal assistem ao mesmo programa. Em Internet, o termo é usado muitas vezes para designar o envio de uma mensagem par todos os membros de um grupo, em vez da remessa para membros específicos.
- *Buffer*: é uma área de armazenamento que compensa diferentes velocidades de fluxos de dados ou temporizações de eventos, ao transferir dados de um dispositivo para outro.
- *CD*: acrônimo de Compact Disc, é um padrão de armazenamento óptico para dados digitais. Conversor A/D: um Conversor Analógico-Digital é componente de um sistema responsável por converter dados analógicos para digitais através da amostragem de um sinal contínuo e sua posterior discretização gerando valores numéricos digitais.
- *Classpath*: é um argumento passado para a Java Virtual Machine indicando onde procurar classes e pacotes para carregamento dinâmico.
- *CPqD*: Centro de Pesquisa e Desenvolvimento em Telecomunicações.
- *DivX*: é um formato de compactação de vídeo criado pela DivxNetworks Inc.
- *Driver*: um driver é um componente de software responsável por estabelecer a comunicação entre hardware e software, provendo comandos para enviar e receber dados de um dispositivo instalado.
- *DVD*: acrônimo de Digital Versatile Disc, é a geração seguinte ao CD, possibilitando um armazenamento maior de dados.

- *ECMAScript*: é uma linguagem de programação baseada em scripts, padronizada pela Ecma International na especificação ECMA-262. A linguagem é bastante usada em tecnologias para Internet, sendo esta base para a criação do JavaScript/JScript e também do ActionScript.

- *FIFO*: acrônimo para First In, First Out (que em português significa primeiro a entrar, primeiro a sair) refere-se a estruturas de dados do tipo fila onde os elementos vão sendo colocados e retirados (ou processados) por ordem de chegada.

- *GEM*: acrônimo de *Globally Executable MHP* [17], é uma parte do MHP independente dos padrões de transmissão europeus, criado com a finalidade de padronizar partes de todos os padrões de *middleware* mundiais e possibilitar a criação de aplicações que funcionem em qualquer *middleware* que seja compatível com o GEM.

- *GSM*: acrônimo de Global System for Mobile Communications é um dos principais padrões para telefonia móvel existente.

- *Heap*: Heap de objetos é o nome dado a parte da memória do computador que contém a estrutura de dados responsável por armazenar todos os objetos durante a execução da máquina virtual Java.

- *HTTP*: acrônimo para HyperText Transfer Protocol (Protocolo de Transferência de Hipertexto), utilizado para transferência de dados na rede mundial de computadores, a World Wide Web.

- *IPC (Inter-Process Communication)*: é o grupo de mecanismos que permite aos processos transferirem informação entre si. Entre estes mecanismos podem ser citados pipes, filas de mensagens e memória compartilhada.

- *Java*: é uma linguagem de programação orientada a objeto desenvolvida na década de 90 pelo programador James Gosling, na empresa Sun Microsystems.

- *JavaBeans*: são componentes reutilizáveis de software escritos em linguagem Java, e que seguem algumas convenções de modo a permitir que ferramentas possam utilizá-los e manipulá-los.

- *JavaTV*: é uma biblioteca Java que contempla a maior parte dos recursos necessários para a operação de sistemas receptores de TV digital, simplificando assim o desenvolvimento de softwares, uma vez que os programadores de aplicativos podem se voltar ao tema principal da aplicação em desenvolvimento.

- *JNI (Java Native Interface)*: é um padrão de programação que permite que a máquina virtual da linguagem Java acesse bibliotecas construídas com o código nativo de um sistema.

- *Lista ligada*: é uma estrutura de dados linear e dinâmica composta por células que apontam para o próximo elemento da lista.

- *Metodologia (Processo) de Desenvolvimento Ágil*: metodologias de desenvolvimento ágil foram pensadas de forma a minimizar riscos no desenvolvimento de software através períodos mais curtos de lançamento, chamados iterações, que tipicamente levam de uma quatro semanas. Métodos ágeis prezam mais a comunicação face-a-face que a documentação, como forma de acelerar o processo de desenvolvimento.

- *MHP*: acrônimo de *Multimedia Home Platform* [16], é um padrão aberto de *middleware* para TV Digital, adotado principalmente pelos países europeus. Está incluso em outro padrão maior, o *Digital Video Broadcasting* (DVB) [13] que agrupa todas as características da tecnologia de TV Digital européia.

- *Middleware* (conforme [18]): é a camada de software intermediário que permite o desenvolvimento de aplicações interativas para a TV Digital de forma independente da plataforma de hardware dos fabricantes de terminais de acesso (*set-top boxes*).

- *MPEG*: acrônimo para Moving Picture Experts Group, é o grupo de trabalho da Organização Internacional para Padronização (ISO) para o desenvolvimento de padrões para vídeo e áudio digitais.

- *MPEG2*: é um padrão de compressão e codificação de vídeo para difusão e comunicações, bem como para armazenamento em meios diversos, tais quais os ópticos.

- *MPEG-TS*: MPEG Transport Stream (TS, TP, ou MPEG-TS) é um protocolo de comunicação para transmissão de áudio, vídeo e dados, especificado pelo padrão ISO/IEC 13818-1. Permite multiplexar vídeo e áudio digital sincronizando a saída. Possui correção de erro e transporte, e é usado para difusão de aplicações como DVB e ATSC.

- *MOV*: formato de vídeo criado pela Apple, é um container para imagem, diversas trilhas, efeitos e textos. Sua base foi aprovada pela ISO como padrão para MPEG4 Part. 14.

- *Multiplexação no tempo*: a multiplexação no tempo consiste na transmissão de dois ou mais sinais ou fontes de bits simultaneamente através de um único canal pela

divisão do tempo em pequenos compartimentos de tamanhos fixos, onde são transmitidos alternadamente um pouco de cada sinal.

- *Overhead*: Em computação overhead é geralmente considerado qualquer processamento ou armazenamento em excesso, seja de tempo de computação, de memória, de largura de banda ou qualquer outro recurso que seja requerido para ser utilizado ou gasto para executar uma determinada tarefa.
- *Pipe (UNIX)*: é o redirecionamento da saída padrão de um programa para a entrada padrão de outro.
- *Set-top Box (STB)*: é o termo que descreve um equipamento que se conecta a um televisor e a uma fonte externa de sinal, transformando este sinal em conteúdo no formato que possa ser apresentado em uma tela.
- *SBTVD* [15]: acrônimo para Sistema Brasileiro de TV Digital.
- *SMS*: acrônimo para Short Message Service. Tecnologia amplamente utilizada em telefonia celular para a transmissão de mensagens de texto curtas.
- *SOAP*: acrônimo de *Service Oriented Architecture Protocol*, é um protocolo de troca de mensagens em formato XML.
- *Thread*: é uma forma de um processo dividir a si mesmo em duas ou mais tarefas que podem ser executadas simultaneamente.
- *UML*: acrônimo de Unified Modelling Language, é um padrão gráfico de especificação para modelagem de objetos, e uma ferramenta importante no processo de desenvolvimento de software.
- *USB*: acrônimo de Universal Serial Bus, é uma especificação para interfaces de comunicação serial de dados. É padronizado pelo USB Implementers Forum (USB-IF), que possui membros como Apple Inc., Hewlett-Packard, NEC, Microsoft e Intel.
- *W3C*: acrônimo de World Wide Web Consortium, é o principal órgão de padronização para a Web (Internet).
- *Web Services*: é definido pelo W3C como um sistema de software projetado para dar suporte à comunicação interoperável entre duas máquinas utilizando uma rede. Essa comunicação é feita através de mensagens XML utilizando-se servidores web, em um padrão denominado SOAP.

- *Wi-Fi*: é uma marca registrada pertencente à Wireless Ethernet Compatibility Alliance (WECA), e é a abreviatura de *wireless fidelity*, sendo uma tecnologia de interconexão entre dispositivos sem fio, usando o protocolo IEEE 802.11.

- *WiMax*: acrônimo de *Worldwide Interoperability for Microwave Access*, é o nome comercial para o padrão IEEE 802.16, que especifica uma interface sem fio para redes metropolitanas, agregando conhecimentos e recursos mais recentes ao padrão Wi-Fi visando melhor performance de comunicação.

- *WMV*: *Windows Media Video* é o nome para uma série de formatos de vídeo compactados criados pela Microsoft.

- *XHTML*: acrônimo para *eXtensible HyperText Markup Language*, é uma linguagem de marcação com as mesmas marcas do HTML, porém com uma sintaxe mais rigorosa pois é baseada no XML, e dessa podem ser validadas com bibliotecas XML.

- *XML*: acrônimo para *eXtensible Markup Language*, é uma especificação para uma linguagem de marcação de uso geral, permitindo que possam ser criados novas linguagens.

- *XviD*: formato de compactação de vídeo competidor direto do DivX. Enquanto o DivX é um formato proprietário, o XVid é livre e de código aberto, disponível para diferentes plataformas.

- *YouTube*: é um site na Internet que permite que seus usuários carreguem, assistam e compartilhem vídeos em formato digital. Foi fundado em fevereiro de 2005 por três pioneiros do PayPal, um famoso site da internet ligado a gerenciamento de doações. Foi comprada em 9 de Outubro de 2006 pelo Google, pela quantia de US$1,65 bilhões em ações.

Apêndice A

API FrontPanel

A.1 Utilização da API

A partir da API FrontPanel disponibilizada com o kit de desenvolvimento, foi possível implementar as rotinas de leitura em blocos dos dados e leitura dos triggers e valores definidos no dispositivo. A API facilitou o desenvolvimento das aplicações cliente, uma vez que disponibiliza os métodos tais como UpdateTriggerOuts(), UpdateWireOuts(), e ReadFromPipeOut(), que realizam a transferência de dados conforme os endpoints definidos no dispositivo. Para maiores detalhes da API disponível, a mesma está anexada a seguir.

No software desenvolvido, a seqüência de chamadas à API para a configuração dinâmica do dispositivo com a lógica a ser executada pela FPGA está ilustrada na figura A.1, e está detalhada a seguir:

- **Instanciação do objeto**: ao ser instanciado, o objeto inicia as bibliotecas da API, já realizando a verificação de dispositivos atualmente conectados ao computador host;
- **OpenBySerial()**: realiza a abertura da comunicação com o dispositivo cujo identificação é aquela passada como parâmetro. No caso do dispositivo desenvolvido, a identificação de série é "IotORXnsdt";
- **ResetFPGA()**: chamada utilizada para resetar todos os sinais da FPGA, limpando a área de memória destinada ao programa a ser executado pela FPGA. Esta rotina deve ser chamada para que configurações antigas sejam descarregadas da FPGA;
- **LoadDefaultPLLConfiguration()**: utilizada para utilizar as configurações padrões de geração de clock e do controlador USB disponível no kit de desenvolvimento;
- **ConfigureFPGA()**: utilizada para carregar o arquivo de configuração com a lógica desenvolvida para o dispositivo de captura USB. Pode-se perceber que a configuração da FPGA pode ser realizada dinamicamente, algo que facilita o desenvolvimento, uma vez que diferentes configurações da FPGA podem ser executadas e simuladas, sem haver necessidade de reconfiguração do software desenvolvido, tornando assim o desenvolvimento do hardware e software desacoplados;

```
public boolean InitializeDevice() {
    // Create an instance of the device.
    m_dev = new okFrontPanel();

    // Open the device for communication
    ErrorCode code = m_dev.OpenBySerial("lotORXnsdt");
    if (code.swigValue() != ErrorCode.NoError.swigValue()) {
        System.out.println("Error while opening device. Abort !!");
        return false;
    }

    // Reset previous configurations
    m_dev.ResetFPGA();

    // Setup the PLL from defaults.
    m_dev.LoadDefaultPLLConfiguration();

    // Configure the FPGA with the TSReader binaries
     code = m_dev.ConfigureFPGA("tsreader.fifo.bit");

}
```

Figura A.1: *Inicialização e configuração dinâmica do dispositivo via software*

Já para a seqüência de leitura dos dados do dispositivo, são utilizadas chamadas à API responsáveis pela comunicação direta com os Endpoints definidos previamente. A seqüência de chamadas está ilustrada na figura A.2, e detalhada a seguir:

- **ActivateTriggerIn()**: ao ser chamada, esta função ativa um Endpoint do tipo Trigger, que será um flag setado no dispositivo. Esta chamada é feita para inicializar a memória RAM utilizado no módulo FIFO e para iniciar a captura de dados pelo dispositivo. Esta chamada também é feita para setar os flags de parada de leitura do dispositivo;

- **UpdateWireOuts()**: realiza a leitura de valores fixos em um Endpoint do tipo WireOut a partir do dispositivo. Esta chamada serve para obter do dispositivo a quantidade de endereços que deverão ser lidos pelo host. Pode-se perceber que a quantidade é multiplicada por 2, uma vez que cada endereço lido pelo host retorna 2 bytes, já que o barramento de saída da pilha é de 16 bits. Isto garante que a chamada seguinte de leitura da pilha tente ler a quantidade correta de bytes, ao invés da metade de bytes disponíveis para leitura;

- **ReadFromPipeOut()**: esta chamada à API passa como parâmetros o Endpoint da pilha do dispositivo, a quantidade de bytes que deverão ser lidos da pilha e um buffer de bytes como saída. Esta chamada é a responsável por disparar os flags de leitura de pilha no dispositivo, permitindo assim que a unidade de controle comande a memória FIFO para leitura dos dados nela armazenados;

```
        // Reset the RAM address pointer.
        m_dev.ActivateTriggerIn(ram_reset, (short) 0);
        m_dev.ActivateTriggerIn(start, (short) 0);
        byte[] buf;
        while (true) {
            int transfer_length = 0;

            m_dev.UpdateWireOuts();
            transfer_length = m_dev.GetWireOutValue(can_read_wire)*2;

            buf = new byte[transfer_length];

            int retorno = m_dev.ReadFromPipeOut(pipe, transfer_length, buf);

            synchronized (bytebuf) {
                bytebuf.addLast(buf);
            }
        }
    }

    private void StopTSReader() {
        if (m_dev != null) {
            if (m_dev.IsOpen()) {
                m_dev.ActivateTriggerIn(stop, (short) 0);
            }
            return;
        }
        return;
    }
```

Figura A.2: *Seqüência de chamadas à API para leitura de dados do dispositivo*

A.2 Overview

Overview (Version 3.0.0) The FrontPanel API provides a powerful C++ interface to Opal Kelly USB-based FPGA boards. The library is able to communicate with any Opal Kelly device supported by the USB-FPGA driver, including the XEM3001, XEM3005, and XEM3010.

To build an application using this API, you need to include the file okCUsbFrontPanel.h in files that access the API, and also add the library okFrontPanel.lib to be linked with your project. The library was built under Microsoft Visual Studio .NET (Version 7.1) with the Multi-threaded DLL setting.

To use the library, you create an instance of okCUsbFrontPanel which encapsulates communication with the USB driver and provides access to FrontPanel endpoints. Call GetDeviceCount to determine how many XEMs are attached to the USB. You can use the GetDeviceListXXX methods to determine specific information about the boards attached, then call OpenBySerial to open one of them by referring to its serial number. (If no serial number is provided, this method opens the first available device.)

ConfigureFPGA is used to download a configuration file to the FPGA. Sending data to Wire In endpoints is done with calls to SetWireInValue followed by a call to UpdateWireIns to simultaneously update all Wire Ins. ActivateTriggerIn is used to trigger the XEM.

The API uses polling to query the values of output endpoints such as Wire Outs and Trigger Outs. The methods UpdateWireOuts and UpdateTriggerOuts perform these queries. Once the wire and trigger values have been updated by these methods, you can retrieve the values using the methods GetWireOutValue and IsTriggered.

FrontPanel API Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

okCEvent A base class encapsulating the concept of a generic event coming from the XEM device

okCEventHandler The base class used to create FrontPanel event handlers in your own code

okCPLL22150 This is mainly a container class which holds the appropriate configuration parameters for a Cypress 22150 PLL

okCPLL22393 This is mainly a container class which holds the appropriate configuration parameters for a Cypress 22393 PLL

okCTriggerOutEvent Encapsulates a Trigger Out event coming from the XEM

okCUsbFrontPanel This class encapsulates the functionality of an Opal Kelly FrontPanel-enabled USB device including FPGA configuration, PLL configuration, and FrontPanel endpoint access

okCWireOutEvent Encapsulates a Wire Out event coming from the XEM

Here is a list of all documented class members with links to the class documentation for each member:

- a -

ActivateTriggerIn() : okCUsbFrontPanel

AddEventHandler() : okCUsbFrontPanel

- c -

classifyDevice() : okCUsbFrontPanel

ConfigureFPGA() : okCUsbFrontPanel

ConfigureFPGAFromMemory() : okCUsbFrontPanel

ConnectHandler() : okCEventHandler

- d - doReadFromBlockPipeOut() : okCUsbFrontPanel

doWriteToBlockPipeIn() : okCUsbFrontPanel

- e -

enable16Bit() : okCUsbFrontPanel

EnableAsynchronousTransfers() : okCUsbFrontPanel

- f -

fpgaToFrontPanelError() : okCUsbFrontPanel

- g -

GetAddress() : okCEvent

GetBoardModel() : okCUsbFrontPanel

GetDeviceCount() : okCUsbFrontPanel

getDeviceID() : okCUsbFrontPanel

GetDeviceID() : okCUsbFrontPanel

GetDeviceListModel() : okCUsbFrontPanel

GetDeviceListSerial() : okCUsbFrontPanel

GetDeviceMajorVersion() : okCUsbFrontPanel

GetDeviceMinorVersion() : okCUsbFrontPanel

GetDiv1Divider() : okCPLL22150

GetDiv1Source() : okCPLL22150

GetDiv2Divider() : okCPLL22150

GetDiv2Source() : okCPLL22150

GetEepromPLL22150Configuration() : okCUsbFrontPanel

GetEepromPLL22393Configuration() : okCUsbFrontPanel

GetLastTransferLength() : okCUsbFrontPanel

GetMask() : okCTriggerOutEvent

GetOutputDivider() : okCPLL22393

GetOutputFrequency() : okCPLL22393, okCPLL22150

GetOutputSource() : okCPLL22393, okCPLL22150

GetPLL22150Configuration() : okCUsbFrontPanel

GetPLL22393Configuration() : okCUsbFrontPanel

GetPLLFrequency() : okCPLL22393

GetPLLP() : okCPLL22393

GetPLLQ() : okCPLL22393

GetProgrammingInfo() : okCPLL22393, okCPLL22150

GetReference() : okCPLL22393, okCPLL22150

GetSerialNumber() : okCUsbFrontPanel

GetValue() : okCWireOutEvent

GetVCOFrequency() : okCPLL22150

GetVCOP() : okCPLL22150

GetVCOQ() : okCPLL22150

GetWireOutValue() : okCUsbFrontPanel

- h -
Has16BitHostInterface() : okCUsbFrontPanel
- i -
InitFromProgrammingInfo() : okCPLL22393, okCPLL22150
IsFrontPanel3Supported() : okCUsbFrontPanel
IsFrontPanelEnabled() : okCUsbFrontPanel
IsHighSpeed() : okCUsbFrontPanel
isI2CAddressRestricted() : okCUsbFrontPanel
IsOpen() : okCUsbFrontPanel
IsOutputEnabled() : okCPLL22393, okCPLL22150
IsPLLEnabled() : okCPLL22393
IsTriggered() : okCUsbFrontPanel
- l -
LoadDefaultPLLConfiguration() : okCUsbFrontPanel
- o -
okCPLL22150() : okCPLL22150
okCPLL22393() : okCPLL22393
okCUsbFrontPanel() : okCUsbFrontPanel
okMilliSleep() : okCUsbFrontPanel
OpenBySerial() : okCUsbFrontPanel
openBySerial() : okCUsbFrontPanel
- p -
ProcessTriggerOuts() : okCEventHandler
ProcessWireOuts() : okCEventHandler
- r -
ReadFromBlockPipeOut() : okCUsbFrontPanel
ReadFromPipeOut() : okCUsbFrontPanel
ReadI2C() : okCUsbFrontPanel
ResetFPGA() : okCUsbFrontPanel
resetValues() : okCUsbFrontPanel
- s -
SetAddress() : okCEvent
SetBTPipePollingInterval() : okCUsbFrontPanel
SetCrystalLoad() : okCPLL22393, okCPLL22150
SetDeviceID() : okCUsbFrontPanel

SetDiv1() : okCPLL22150
SetDiv2() : okCPLL22150
SetEepromPLL22150Configuration() : okCUsbFrontPanel
SetEepromPLL22393Configuration() : okCUsbFrontPanel
SetMask() : okCTriggerOutEvent
SetOutputDivider() : okCPLL22393
SetOutputEnable() : okCPLL22393, okCPLL22150
SetOutputSource() : okCPLL22393, okCPLL22150
SetPLL22150Configuration() : okCUsbFrontPanel
SetPLL22393Configuration() : okCUsbFrontPanel
SetPLLLF() : okCPLL22393
SetPLLParameters() : okCPLL22393
SetReference() : okCPLL22393, okCPLL22150
SetTimeout() : okCUsbFrontPanel
SetValue() : okCWireOutEvent
SetVCOParameters() : okCPLL22150
SetWireInValue() : okCUsbFrontPanel
- u -
UnregisterAll() : okCUsbFrontPanel
UpdateTriggerOuts() : okCUsbFrontPanel
UpdateWireIns() : okCUsbFrontPanel
UpdateWireOuts() : okCUsbFrontPanel
- w -
WriteI2C() : okCUsbFrontPanel
WriteToBlockPipeIn() : okCUsbFrontPanel
WriteToPipeIn() : okCUsbFrontPanel
- -
okCUsbFrontPanel() : okCUsbFrontPanel

Apêndice B

Pinagem

B.1 Conector SPI

A partir da API FrontPanel disponibilizada com o kit de desenvolvimento, foi possível implementar as rotinas de leitura em blocos dos dados e leitura dos triggers e valores definidos no dispositivo. A API facilitou o desenvolvimento das aplicações cliente, uma vez que disponibiliza os métodos tais como UpdateTriggerOuts(), UpdateWireOuts(), e ReadFromPipeOut(), que realizam a transferência de dados conforme os endpoints definidos no dispositivo. Para maiores detalhes da API disponível, a mesma está anexada a seguir.

Pin	Signal line	Pin	Signal line
1	Clock A	14	Clock B
2	System Gnd	15	System Gnd
3	Data 7 A(MSB)	16	Data 7 B
4	Data 6 A	17	Data 6 B
5	Data 5 A	18	Data 5 B
6	Data 4 A	19	Data 4 B
7	Data 3 A	20	Data 3 B
8	Data 2 A	21	Data 2 B
9	Data 1 A	22	Data 1 B
10	Data 0 A	23	Data 0 B
11	DVALID A	24	DVALID B
12	PSYNC A	25	PSYNC B
13	Cable Shield		

Figura B.1: *Configuração dos 25 pinos do conector D-subminiatura na interface SPI*

B.2 FPGA e Kit de Desenvolvimento

Nesta seção são detalhadas as pinagens envolvidas com os pinos de entrada e saída da FPGA e os pinos do kit de desenvolvimento, bem como sua interligação com o conector SPI construído pelo grupo.

A figura B.2 representa a interconexão dos pinos da FPGA Spartan-3 nos pinos do kit de desenvolvimento. Já a figura B.3 representa a interconexão entre os pinos do conector SPI com os pinos do kit de desenvolvimento (JP3). É importante verificar que o pino de DEBUG foi utilizado durante a fase de testes para analisar os sinais lógicos através de um osciloscópio.

JP3	Connection	JP3	Connection	JP3	Connection	JP3	Connection	JP3	Connection
1	DGND	11	I/O 148	21	I/O 138	31	DGND	41	+3.3VDD
2	DGND	12	I/O 147	22	I/O 137	32	DGND	42	+3.3VDD
3	I/O 156	13	I/O 146	23	I/O 135	33	I/O 124	43	I/O 114
4	I/O 155	14	I/O 144	24	I/O 133	34	I/O 123	44	I/O 113
5	I/O 154	15	I/O 143	25	I/O 132	35	I/O 122	45	I/O 111
6	I/O 152	16	I/O 141	26	I/O 131	36	I/O 120	46	I/O 109
7	I/O 150	17	I/O 140	27	I/O 130	37	I/O 119	47	I/O / GCLK4 180
8	I/O 149	18	I/O 139	28	I/O 128	38	I/O 117	48	SYS CLK 4
9	+3.3VDD	19	DGND	29	I/O 126	39	I/O 116	49	DGND
10	+3.3VDD	20	DGND	30	I/O 125	40	I/O 115	50	DGND

Figura B.2: *Relação entre os pinos da FPGA e do kit de desenvolvimento*

Pino JP3	Fio SPI	Pino JP3	Fio SPI
3	14	18	18
4	1	21	5
5	----	22	17
6	23	23	4
7	10	24	16
8	22	25	3
11	9	26	----
12	21	33	----
13	8	34	25
14	20	35	12
15	7	36	24
16	19	37	11
17	6	38	DEBUG

Figura B.3: *Relação entre os fios do conector SPI e pinos do kit de desenvolvimento*