UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

GIOVANI BISSANI PEDROSO

PROJETO DE DIPLOMAÇÃO

TRANSMISSÃO DE VÍDEO CODIFICADO EM H.264 VIA ETHERNET

Porto Alegre
2008

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

TRANSMISSÃO DE VÍDEO CODIFICADO EM H.264 VIA ETHERNET

Projeto de Diplomação apresentado ao Departamento de Engenharia Elétrica da Universidade Federal do Rio Grande do Sul, como parte dos requisitos para Graduação em Engenharia Elétrica.

ORIENTADOR: Prof. Dr. Altamiro Amadeu Susin

Porto Alegre
2008

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

GIOVANI BISSANI PEDROSO

TRANSMISSÃO DE VÍDEO CODIFICADO EM H.264 VIA ETHERNET

Este projeto foi julgado adequado para fazer jus aos créditos da Disciplina de "Projeto de Diplomação", do Departamento de Engenharia Elétrica e aprovado em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: Prof. Dr. Altamiro Amadeu Susin, UFRGS
Doutor pelo INPG – Grenoble, França

Banca Examinadora:

Prof. Dr. Altamiro Amadeu Susin, UFRGS
Doutor pelo Institut National Polytechnique de Grenoble – Grenoble, França

Eng. Marcos Artur Mallmann Trombetta
Graduado pela UFRGS – Porto Alegre, Brasil

Prof. Dr. Cláudio Walter, UFRGS
Doutor pelo Institut National Polytechnique de Grenoble – Grenoble, França

Porto Alegre, novembro de 2008.

DEDICATÓRIA

Dedico este trabalho a minha família, minha namorada e meus amigos, pela dedicação e apoio em todos os momentos difíceis.

AGRADECIMENTOS

Agradeço principalmente a minha família pelo apoio e confiança em mim.

Ao Prof. Orientador Altamiro Susin pelas idéias iniciais do projeto e por ter disponibilizado todos os recursos do LaPSI para a realização desse projeto.

RESUMO

O nascimento do Sistema Brasileiro de Televisão Digital impulsionou vários setores de pesquisa e desenvolvimento no Brasil. São muitas as áreas de conhecimento relacionadas, dentre elas está a codificação/decodificação de vídeo. A UFRGS possui um grupo que desenvolve soluções arquiteturais para compressão e descompressão de vídeo conforme o padrão oficial do SBTVD, o H.264. Toda a estrutura desenvolvida foi prototipada em dispositivos FPGAs, que têm características importantes como a flexibilidade, desempenho e a possibilidade de uma prototipação rápida. A motivação para esse projeto é usar esse hardware desenvolvido como base para uma porta de vídeo H.264. Para isso serão necessárias adequações ao projeto original, como a adição de uma interface de comunicação de alta velocidade, nesse caso uma interface Ethernet. Esse documento discute os pontos importantes desse projeto.

Palavras-chaves: Compressão de Vídeo, Redes Ethernet, Sockets, Sistemas Embarcados, Hardware-Software Co-design, HDL, Linguagem C.

ABSTRACT

The birth of the Brazilian System of Digital Television boosted various sectors of research and development in Brazil. There are many areas of related knowledge; one of them is the encoding and decoding of video. The UFRGS has a group that develops architectural solutions for video compression/decompression as the official SBTVD standard, H.264. The entire developed structure was prototyped in FPGA devices, which have important characteristics such as flexibility, performance and the possibility of a fast prototyping. The motivation for this project is to use that hardware developed as a basis for a video port based on H.264. It will be necessary adjustments to the original project, as the addition of a high-speed communication interface, in this case an Ethernet interface. This document discusses the important points of this project.

Keywords: Video Compression, Ethernet, Sockets, Embedded Systems, Hardware-Software Co-design, HDL, language C.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

UFRGS: Universidade Federal do Rio Grande do Sul

AVC: Advanced Video Coding

SBTVD: Sistema Brasileiro de Televisão Digital

FPGA: Field-Programmable Gate Array

DVD: Digital Video Disc

MPEG: Moving Picture Experts Group

ITU-T: International Telecommunication Union-Telecommunication

VCEG: Video Coding Experts Group

ISO: International Organization for Standardization

IEC: International Electrotechnical Commission

JVT: Joint Video Team

DCT: Discrete Cosine Transform

QP: Quantization Parameter

RTL: Register Transfer Level

ASIC: Application-Specific Integrated Circuit

PLB: Processor Local Bus

OPB: On-Chip Peripheral Bus

JTAG: Joint Test Action Group

RAM: Random-Access Memory

DDR: Double Data Rate

UART: Universal Asynchronous Receiver-Transmitter

OSI: Open Systems Interconnection

TCP: Transmission Control Protocol

UDP: User Datagram Protocol

MAC: Media Access Control

VHDL: Very High-level Design Language

IP: Intellectual Property

PHY: Physical Layer Device

FIFO: First In, First Out

1 INTRODUÇÃO

A implantação de um Sistema Brasileiro de Televisão Digital foi um evento que gerou a necessidade de um grande investimento em pesquisa e desenvolvimento. Existem muitas áreas de conhecimento associadas que podem ser exploradas. Um desses temas é a codificação de vídeo, indispensável para fins de transmissão da TV digital.

Além da pesquisa com intuito de aprimorar esse sistema, fazendo com que se atinjam maiores taxas de compressão sem perda da qualidade do vídeo, existe também um estudo sobre arquiteturas de hardware necessárias para implementação desse codec. Atingir alto desempenho de hardware pelo menor custo produtivo é a meta desse estudo.

A UFRGS possui um grupo de desenvolvimento de soluções arquiteturais para o padrão H.264/AVC (padrão oficial do SBTVD) de codificação de vídeo. A arquitetura desenvolvida foi prototipada em nível de hardware e uma placa de desenvolvimento que possui um FPGA Virtex-II Pro da Xilinx. Além do FPGA essa placa possui vários periféricos que a tornam um ambiente completo de desenvolvimento.

Aproveitar essas características descritas e o hardware já existente foi a motivação para esse projeto que tem como objetivo criar uma porta de vídeo codificado em H.264. Para isso serão necessárias algumas adequações do projeto original, sendo a mais importante delas a implantação de uma porta de comunicação de alta velocidade entre a plataforma de desenvolvimento e um computador. Esse documento descreve todas as etapas realizadas desse projeto.

2 CONTEXTO DO PROJETO

A compressão de vídeo é uma tecnologia essencial para aplicações como TV digital, DVD-Video e videoconferências. É cada vez mais importante conseguir extrair o máximo de informação do vídeo através de uma quantidade mínima de dados. E tudo isso, obviamente, sem perda significativa na qualidade. Essa busca se deve ao fato das aplicações citadas acima serem dependentes de envio e recebimento de dados.

O objetivo então é conseguir o máximo desempenho em transmissão de vídeo. Isso é possível com novas tecnologias que possibilitam o uso de altas taxas de tráfego de dados e também com uso de ferramentas compressoras, chamadas *encoders*. Essas ferramentas utilizam um algoritmo de compressão determinado por um padrão.

Dentre os padrões que existem e são utilizados atualmente, o H.264 é o mais promissor. Foi criado a partir dos padrões MPEG-2 e MPEG-4 com a intenção de fornecer grande qualidade de vídeo com uma taxa de *bitrate* muito mais baixa que os seus predecessores. Uma comparação entre H.264 e MPEG-2 pode ser vista na Figura 1. Os aspectos mais importantes do seu algoritmo serão apresentados a seguir.

Figura 1 - Comparação entre o padrão H.264 e MPEG-2

2.1 PADRÃO DE COMPRESSÃO DE VÍDEO – H.264

H.264 é um padrão para compressão de vídeo, baseado no MPEG-4 Part 10 ou AVC (*Advanced Video Coding*). O padrão foi desenvolvido pela ITU-T *Video Coding Experts Group* (VCEG) em conjunto com a ISO/IEC MPEG que formaram uma parceria conhecida por *Joint Video Team* (JVT). A versão final, formalmente chamada por ISO/IEC 14496-10), foi lançada em maio de 2003.

Um *encoder* H.264 utiliza processos de predição, transformada e codificação (Figura 2) para produzir um *bitstream* comprimido. O *decoder*, por sua vez, utiliza os processos inversos: transformada inversa, reconstrução e decodificação para produzir uma seqüência de vídeo pronta para ser exibida.

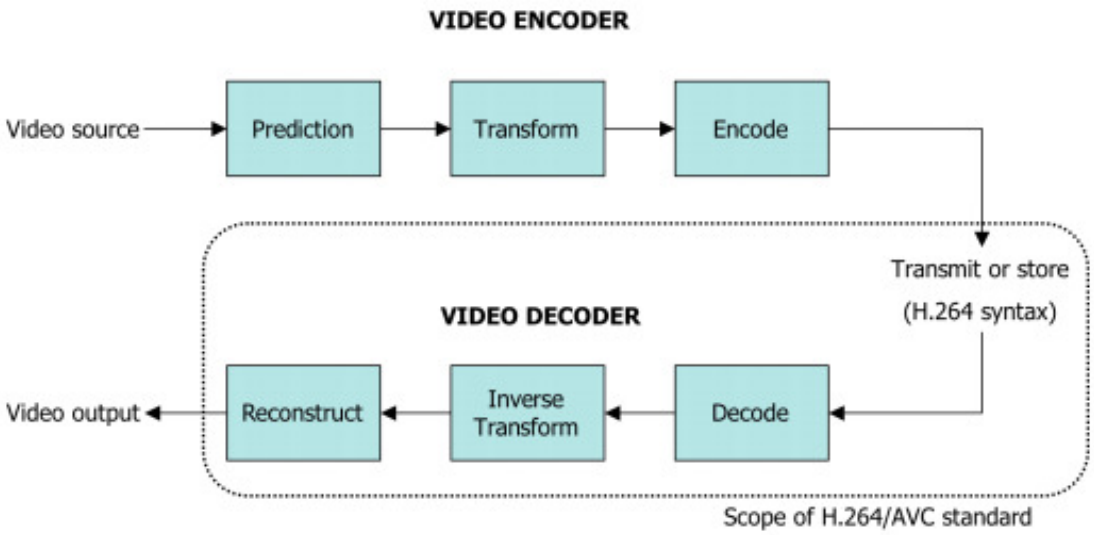

Figura 2 - Processo de compressão/descompressão de vídeo H.264

2.1.1 PROCESSOS DE COMPRESSÃO

2.1.1.1 PREDIÇÃO

O *encoder* processa um frame de vídeo em unidades chamadas macroblocos (16x16 pixels). A predição do macrobloco é feita com base em dados previamente codificados, dentro do frame atual (predição intra) ou a partir de outros frames já codificados e transmitidos (predição inter). Da subtração da predição do macrobloco corrente se forma o resíduo.

Os métodos de predição suportados pelo H.264 são mais flexíveis do que aqueles em normas anteriores, permitindo predições exatas e, portanto, uma compressão de vídeo mais eficaz. A Predição Intra usa blocos de tamanho 16x16 ou 4x4 de dentro do mesmo frame e previamente codificados para predizer o macrobloco circundante (Figura 3).

E a Predição Inter utiliza uma variedade de tamanhos de blocos (valores entre 16x16 e 4x4) para a predição de pixels no frame atual em regiões similares aos frames já codificados (Figura 4).

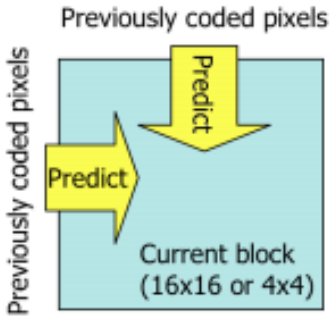

Figura 3 - Predição Intra

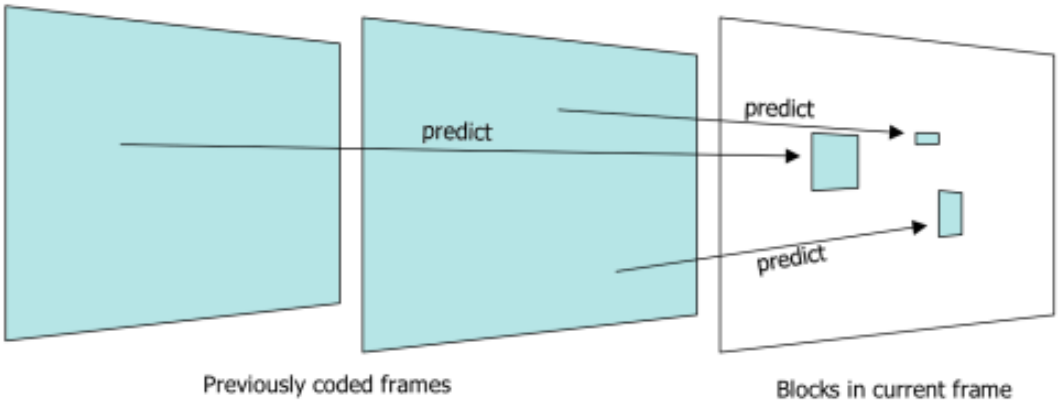

Figura 4 - Predição Inter

2.1.1.2 TRANSFORMADA E QUANTIZAÇÃO

Um bloco de amostras residuais é transformado por uma forma aproximada da Transformada de Cosseno Discreto (DCT). O resultado da transformada é um conjunto de coeficientes, cada um representando uma ponderação de um valor base de modelo padrão. Quando combinados, recriam o bloco de amostras residuais. A Figura 5 mostra como a DCT inversa cria um bloco de imagem através das ponderações indicadas pelos coeficientes resultantes da DCT.

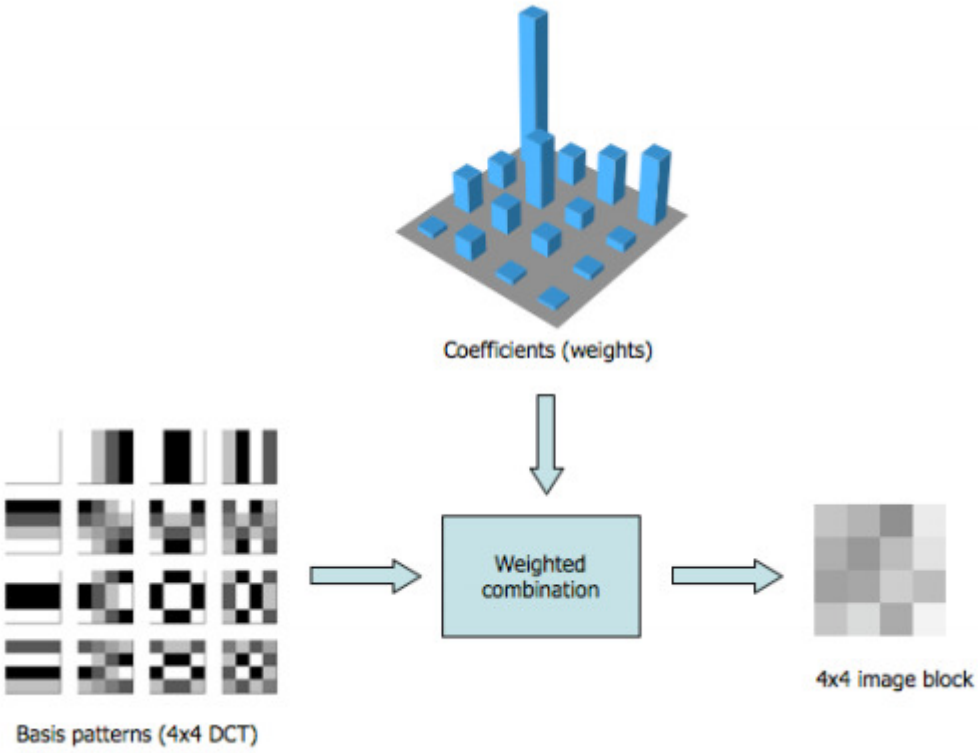

Figura 5 - Transformada inversa: combinando coeficientes ponderados com um padrão base para criar um bloco de imagem 4x4

O resultado da transformada, um bloco de coeficientes, é quantizado. Isto é, cada coeficiente é dividido por um valor inteiro. A quantização reduz a precisão dos coeficientes da transformada de acordo com um parâmetro de quantização (QP). Tipicamente, o resultado é um bloco que contém a maioria dos coeficientes nulos.

Assumindo valores maiores de QP significa que mais coeficientes são fixados em zero, resultando em alta compressão, porém com uma perda na qualidade de vídeo. Para valores menores de QP acontece o inverso, a qualidade da decodificação de vídeo aumenta à custa de uma menor taxa de compressão.

2.1.1.3 CODIFICAÇÃO

O processo de codificação do vídeo produz uma quantidade de valores que precisam ser codificados para formar um *bitstream*. Esses valores incluem:

- Coeficientes da transformada quantizados;

- Informação que permita ao decodificador recriar a predição;

- Informação sobre a estrutura de dados comprimida e as ferramentas de compressão utilizadas durante a codificação;

- Informação sobre a seqüência de vídeo completa.

Esses valores e parâmetros (elementos de sintaxe) são convertidos em códigos binários através de codificação aritmética e/ou codificação de comprimento variável. Cada um desses métodos de codificação produz uma eficiente e compacta representação binária da informação. A partir desse momento, o *bitstream* já pode ser armazenado ou transmitido.

2.1.2 PROCESSOS DE DESCOMPRESSÃO

2.1.2.1 DECODIFICAÇÃO DO *BITSTREAM*

O decodificador de vídeo recebe o *bitstream* H.264, decodifica cada um dos elementos de sintaxe e extrai as informações descritas acima (coeficientes quantizados, informação de predição, etc). Essa informação, então, é utilizada para inverter o processo de codificação e recriar uma seqüência de imagens de vídeo.

2.1.2.2 REDIMENSIONAMENTO E TRANSFORMADA INVERSA

Os coeficientes da transformada quantizados são redimensionados. Cada um é multiplicado por um valor inteiro, restaurando assim sua escala original. A transformada inversa combina um padrão de valores, ponderados pelos coeficientes redimensionados, para recriar cada bloco residual. Esses, por sua vez, são combinados gerando um macrobloco residual.

2.1.2.3 RECONSTRUÇÃO

Para cada macrobloco, o decodificador faz uma predição idêntica à que foi criada pelo *encoder*. O *decoder*, então, acrescenta a predição ao resíduo decodificado para reconstruir o macrobloco original que pode ser exibido como parte do frame de vídeo.

O padrão de compressão de vídeo H.264 é utilizado pelo Sistema Brasileiro de Televisão Digital. Esse é um sistema novo em fase de desenvolvimento, por isso existem alguns centros de pesquisa sobre diversas áreas de conhecimento relacionadas. Dentre as muitas partes desse projeto, está a concepção de codificadores e decodificadores H.264. A UFRGS possui um grupo responsável pela pesquisa de soluções arquiteturais de construção desses dispositivos em hardware.

Acerca do dispositivo decodificador, está sendo desenvolvida uma arquitetura representada pela Figura 6. Os módulos desse decodificador são: decodificador de entropia, transformadas e quantização inversas, predição intra quadros, compensação do movimento e filtro de redução do efeito de bloco. Essa arquitetura está completamente descrita em VHDL e validada em nível RTL.

A plataforma de prototipação utilizada é uma placa da Digilent, a qual contém o FPGA da Xilinx Virtex-II Pro XCVP30. Além disso, essa placa possui diversos outros ASICs

para tarefas de I/O (input/output, entrada e saída) com dispositivos físicos também para tal. É um ambiente completo de desenvolvimento. Essa placa foi eleita de acordo com as necessidades do projeto: flexibilidade, desempenho e a possibilidade de uma rápida prototipação. Na Figura 7 tem-se uma visão geral da placa, com alguns dos componentes de hardware.

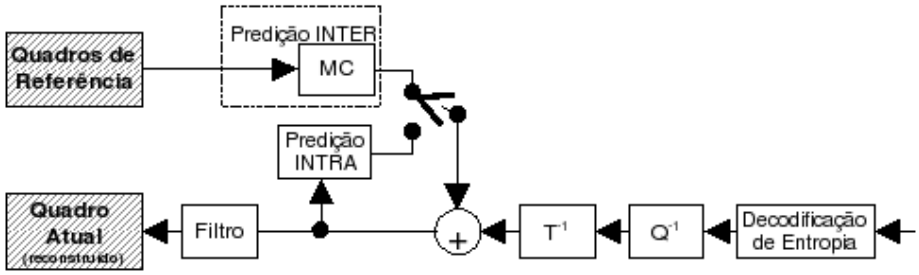

Figura 6 - Arquitetura do decodificador H.264

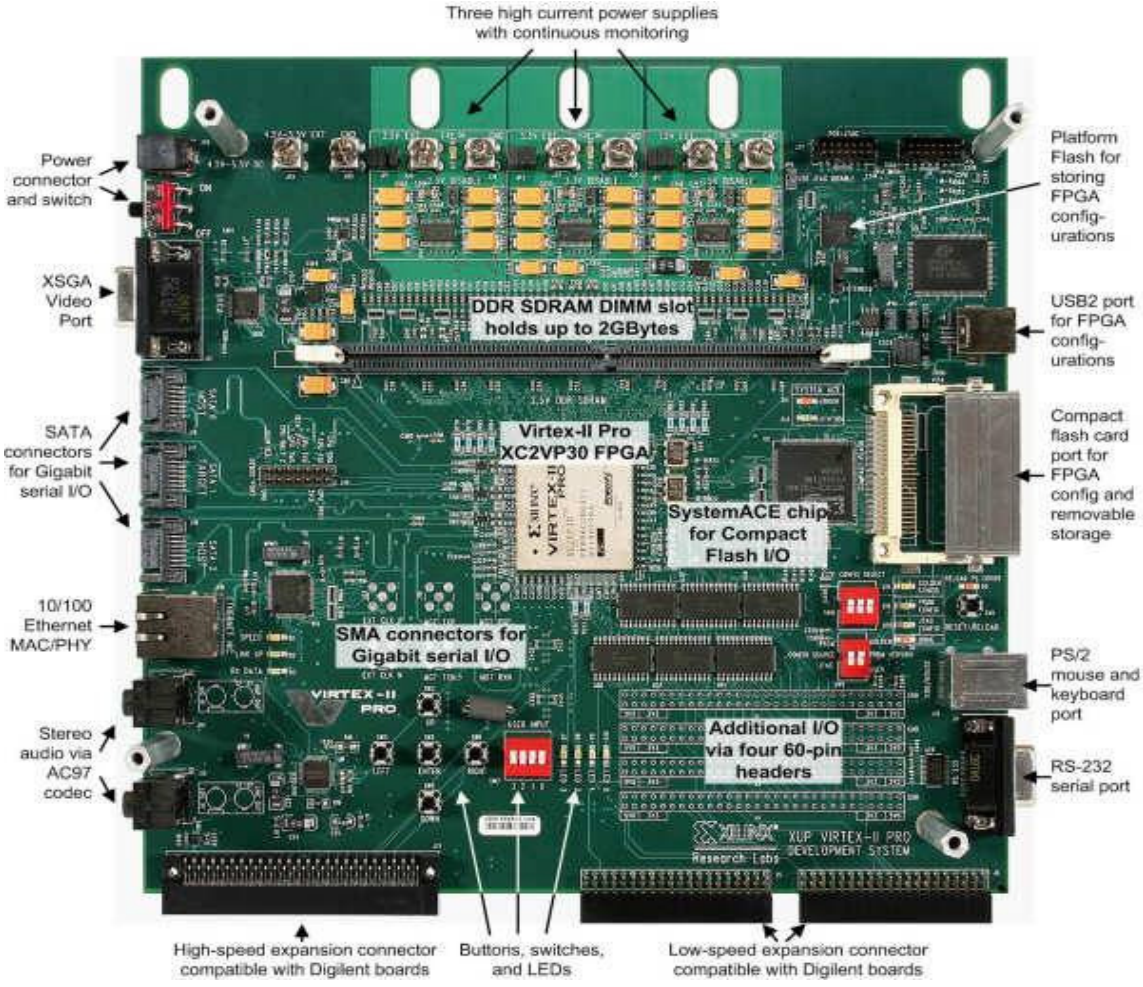

Figura 7 - Visão geral da placa XUP Virtex-II Pro utilizada

2.2 – PROJETO INTEGRAÇÃO

A fim de testar o funcionamento dos módulos do decodificador H.264 atuando em conjunto foi desenvolvido o Projeto Integração. Esse projeto consistiu na criação de uma plataforma de testes que possibilita a depuração dos módulos de hardware implementados no FPGA da placa de desenvolvimento. Além do decodificador propriamente dito, outras funcionalidades foram adicionadas à placa de forma a tornar a depuração mais eficiente.

A adição desses novos componentes foi possível através do uso da ferramenta Xilinx Platform Studio. Ela possibilita a criação de um ambiente Hardware-Software Co-Design no FPGA. Esse sistema é centralizado no processador PowerPC que existe no interior do FPGA. Barramentos para interconexão de periféricos são criados conforme a necessidade. Eles podem ser de dois tipos basicamente: Processor Local Bus (PLB), é um barramento rápido e geralmente conecta memórias de dados e de programa e periféricos de comunicação rápida com o processador; On-chip Peripheral Bus (OPB), é um barramento mais lento dedicado à conexão dos periféricos com o barramento PLB do processador.

A ferramenta ainda disponibiliza memórias de dados e programa ao processador através do barramento PLB, um bloco de reset e uma interface via JTAG para depuração da execução do software. Fica a critério do usuário então, adicionar mais periféricos de acordo com a sua necessidade. Um diagrama simplificado dessa estrutura está demonstrado na Figura 8.

O sistema relativo ao projeto integração possui o decodificador H.264 conectado ao processador através de um barramento OPB, uma interface de comunicação serial (UART) também conectada ao barramento OPB, uma interface com uma memória RAM DDR externa conectada através do barramento PLB e uma interface com o ASIC responsável pelo controle

da saída XSGA de vídeo, além dos periféricos básicos anteriormente citados. A Figura 9 mostra a representação dessa arquitetura no software Xilinx Platform Studio.

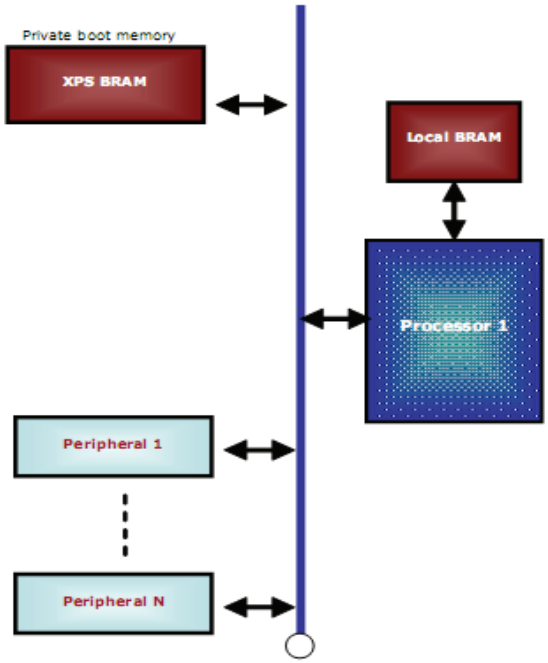

Figura 8 - Diagrama de interconexões entre o processador e seus periféricos

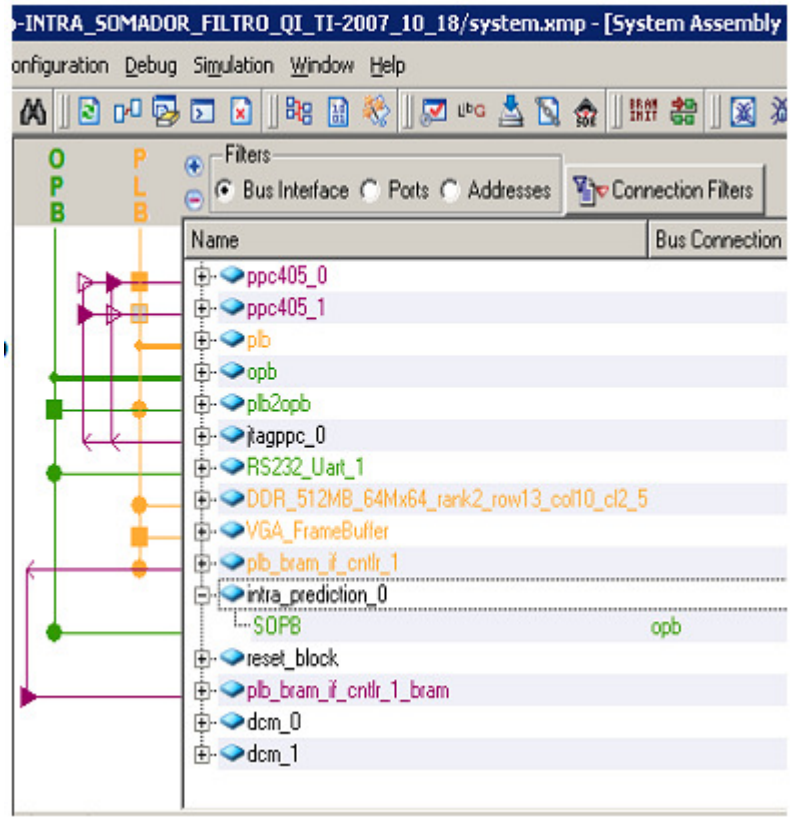

Figura 9 - Estrutura do Projeto Integração no Xilinx Platform Studio

Essa plataforma de testes funcionava da seguinte forma: o bitstream com as informações e dados para o decodificador H.264 eram transmitidos através da porta serial do PC até a placa. A interface UART da placa recebia os dados a uma taxa nominal de 115,2kbps. Então, todo esse conteúdo era armazenado na memória RAM DDR externa. Por fim, uma rotina de controle em software direcionava os dados salvos na RAM para entrada do decodificador H.264 e enviava a seqüência de vídeo, resultado da decodificação, ao ASIC que controla a porta XSGA. Conforme as ações aconteciam, através de um terminal no PC era possível fazer a depuração dos eventos.

2.3 – ESPECIFICAÇÃO DO PROJETO

Surgiu, então, a oportunidade de usar esse hardware já construído e validado como base para implementação de uma porta de vídeo codificado em H.264. A plataforma de desenvolvimento permite a flexibilidade necessária para alcançar esse objetivo. Porém, seria necessária a inclusão de funcionalidades novas na plataforma já existente, aproveitando seu hardware. A primeira e mais importante delas seria um sistema de transporte de dados mais eficiente do que o serial (RS232).

Uma porta de vídeo H.264 deveria ser capaz de transmitir qualquer vídeo codificado nesse padrão com qualquer tamanho e duração. Por isso, um fator que não pode ser limitante é o armazenamento. Logo, seria necessária uma transmissão de dados a uma taxa parecida com a qual o decodificador recebe os dados. Considerando o limite que seria representado pelo *bitrate* necessário para decodificar um vídeo em alta definição com 1920x1080 linhas de resolução que é aproximadamente 30Mbps, é possível afirmar que o uso de uma interface de comunicação Ethernet atenderia até o caso extremo, visto que ela possibilita taxas de envio e recebimento de dados entre 10Mbps e 100Mbps.

2.4 – TRANSMISSÃO DE DADOS – INTERFACE ETHERNET

Ethernet é uma tecnologia de interconexão para redes locais baseada no envio de pacotes. Ela define, dentre outras coisas, o formato desses pacotes e os protocolos utilizados. Além disso, existe também um conceito importante nessa tecnologia, que são as chamadas camadas OSI. Essa arquitetura é um modelo que divide as redes de computadores em sete camadas, de forma a se ter camadas de abstração. Cada protocolo implementa uma funcionalidade assinalada a uma determinada camada.

Protocolos de redes de computadores são regras formais de comportamento que regem comunicações em redes [Sockets, 1996]. Eles são úteis para interconexão de dados em níveis de hardware e também em níveis de software (aplicativos). Protocolos de hardware definem, por exemplo, como os dispositivos operam e exercem funções em cooperação. Protocolos de software definem de que forma se dá a comunicação entre aplicativos distintos. A Figura 10 mostra a relação entre as camadas de abstração anteriormente citadas e seus respectivos protocolos.

2.4.1 – MODELO OSI – INTERCONEXÃO DE SISTEMAS ABERTOS

Uma característica comum a todas as arquiteturas de redes é o uso de várias camadas, cada uma é responsável por um grupo de tarefas, fornecendo um conjunto de serviços bem definidos para o protocolo da camada superior. As camadas mais altas estão logicamente mais perto do usuário (chamada camada de aplicação), e lidam com dados mais abstratos, confiando em protocolos de camadas mais baixas para tarefas de menor nível de abstração. Esse modelo disposto em camadas que é utilizado em redes atualmente foi definido pela ISO

e é chamado de Modelo de Referência para Interconexões de Sistemas Abertos. A Tabela 1 abaixo resume esse modelo.

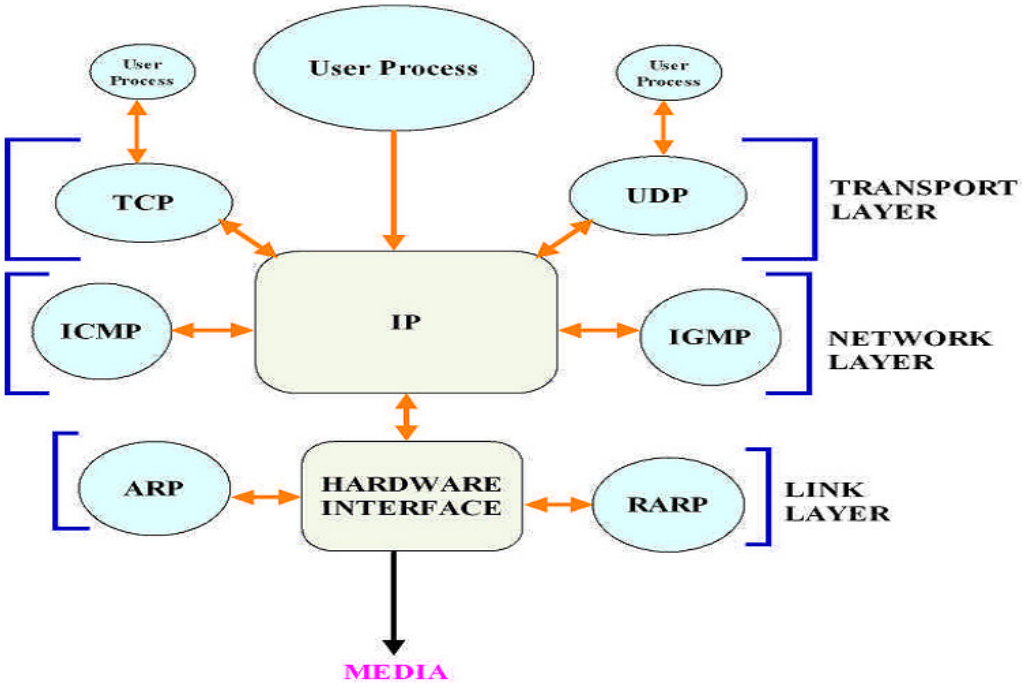

Figura 10 - Camadas OSI e seus respectivos protocolos

Tabela 1 - Tabela resumo do modelo OSI

Camadas	Estrutura	Protocolos utilizados
7 – Aplicação	Dados	DNS; FTP; TFTP; BOOTP; SNMP; LOGIN; SMTP; MIME; NFS; FINGER; TELNET; NCP; APPC; AFP; SMB
6 – Apresentação	Dados	XDR, SSL, TLS
5 – Sessão	Segmentos	TCP, ARP, RARP; SPX, NWLink, NetBIOS / NetBEUI, ATP
4 – Transporte	Segmentos	TCP, **UDP**, RTP, SCTP
3 – Rede	Pacotes	IP; ARP; RARP, ICMP; RIP; OSFP; IGMP
2 – Enlace	Frames	**Ethernet**, Token Ring, FDDI, PPP, HDLC, Q.921, Frame Relay, ATM, Fibre Channel
1 – Física	Bits	RS-232, V.35, V.34, Q.911, T1, E1, 10BASE-T,100BASE-TX , ISDN, SONET, DSL

A Camada de Aplicação é a camada de mais alta ordem do modelo OSI e, portanto, a mais próxima do usuário final. Ela faz a interface entre o protocolo de comunicação e o aplicativo que pediu ou receberá a informação através da rede. A Camada de Apresentação

fornece uma representação padrão para o envio de dados através da rede. Ela faz a conversão do formato do dado recebido pela Camada de Aplicação para um formato comum a ser usado na sua transmissão, ou seja, um formato entendido pelo protocolo usado.

A Camada de Sessão permite que duas aplicações em computadores diferentes estabeleçam uma sessão de comunicação. Nesta sessão, essas aplicações definem como será feita a transmissão de dados e coloca marcações nos dados que estão sendo transmitidos. Se porventura a rede falhar, os computadores reiniciam a transmissão dos dados a partir da última marcação recebida pelo computador receptor. A Camada de Transporte realiza detecção de erro e correções, garantindo que o aplicativo receptor receba o dado exatamente como foi mandado [Sockets, 1996].

A Camada de Rede é responsável pelo endereçamento dos pacotes, convertendo endereços lógicos (ou IP) em endereços físicos, de forma que os pacotes consigam chegar corretamente ao destino. Essa camada também determina a rota que os pacotes irão seguir para atingir o destino, baseada em fatores como condições de tráfego da rede e prioridades. A Camada de Enlace é responsável por entregar dados confiáveis para a rede física. Por último, a Camada Física que se preocupa com a transmissão e recepção de *bits* ao longo de um canal de comunicação física. **Ethernet** é um exemplo desse tipo de canal. Níveis de tensão nas linhas e pinagens entre conexões também são características dessa camada.

2.4.2 – PROTOCOLO TCP

Quando se estabelece uma interface de comunicação, a primeira exigência é de que os dados sejam enviados em seqüência, sem erros, de forma segura e livre de duplicação. Todos esses requerimentos são atendidos quando se usa o TCP. Esse protocolo atua na camada de transporte de dados. Ele tem como característica quebrar o *stream* de dados em pequenas

unidades chamadas segmentos. Para o envio desses segmentos é necessário que haja uma conexão entre a fonte e o destinatário; isso significa dizer que o TCP é um protocolo orientado a conexões. Sua robustez pode ser notada através da estrutura de seu cabeçalho na Figura 11 abaixo.

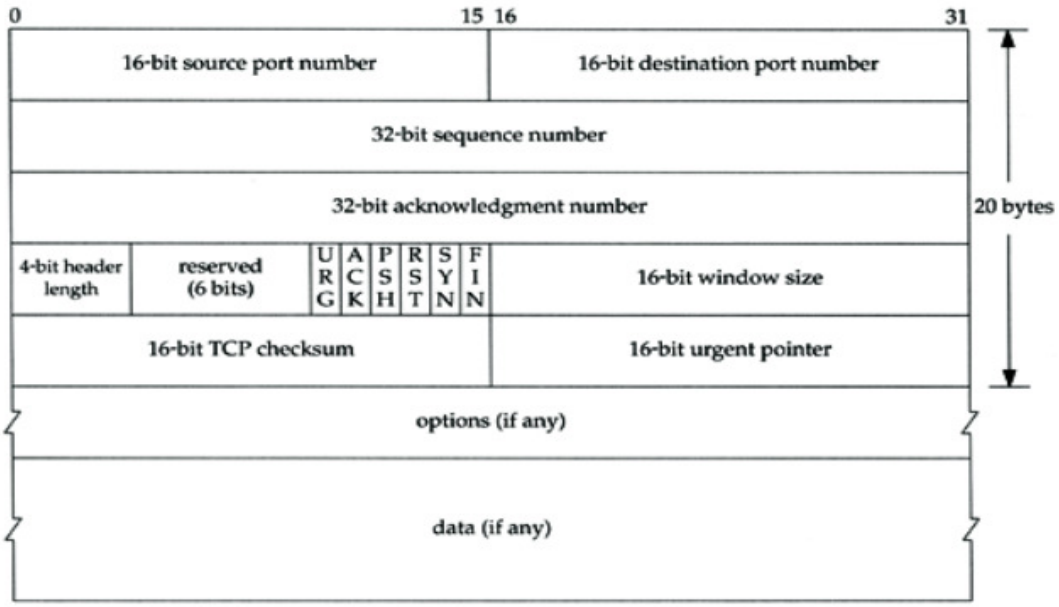

Figura 11 - Cabeçalho TCP

2.4.3 – PROTOCOLO UDP

É um simples protocolo da camada de transporte orientado a datagramas. Cada operação de envio/recebimento de dados produz exatamente um datagrama UDP, o que resulta em um datagrama IP a ser enviado/recebido. O protocolo UDP não é confiável, pois não tem nenhuma função de identificação e/ou correção de erros associada. A simplicidade do cabeçalho UDP denuncia essa sua característica na Figura 12. Caso garantias sejam necessárias, é necessária a implementação de estruturas de controle tais como *timeouts*, retransmissões, *acknowlegments*, etc. Outra característica do UDP é o fato de ser um serviço que dispensa conexão.

Apesar de ter algumas desvantagens em relação ao protocolo TCP, o UDP foi escolhido como base para a transmissão dos dados codificados em H264 para a plataforma de testes. O fato de não ser necessária nenhuma conexão entre cliente e servidor, faz com que sua implementação seja mais fácil e que a comunicação ocorra de forma mais rápida. A Figura 13 mostra um comparativo entre fluxogramas de comunicação via *socket* usando protocolos UDP e TCP.

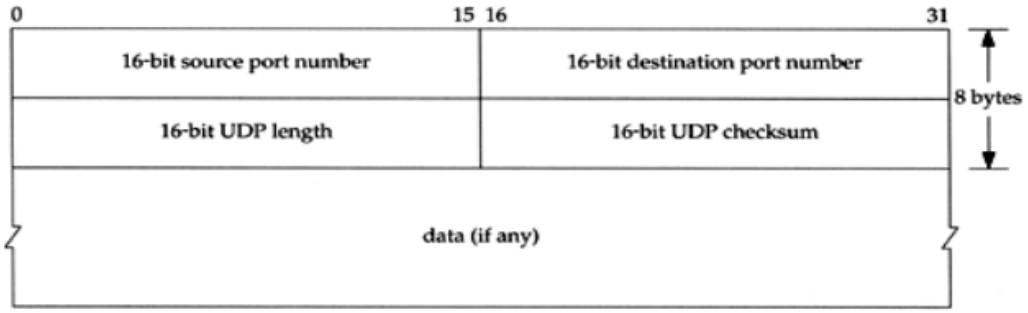

Figura 12 - Cabeçalho UDP

2.4.4 – SOCKETS

Para que efetivamente ocorra a comunicação entre máquinas mediante a pilha de protocolos TCP/IP o endereço IP é fundamental (e muito famoso), porém outros dois fatores são essenciais e precisam ser definidos: o protocolo de transporte e a porta de comunicação. A esta tríade é chamada de *socket* (ou *network socket*), cuja função é identificar univocamente uma aplicação na rede. De fato, o sistema operacional associa o *socket* a um ou mais processos existentes na memória e o utiliza como canal para envio e recebimento dos dados relacionados pela rede.

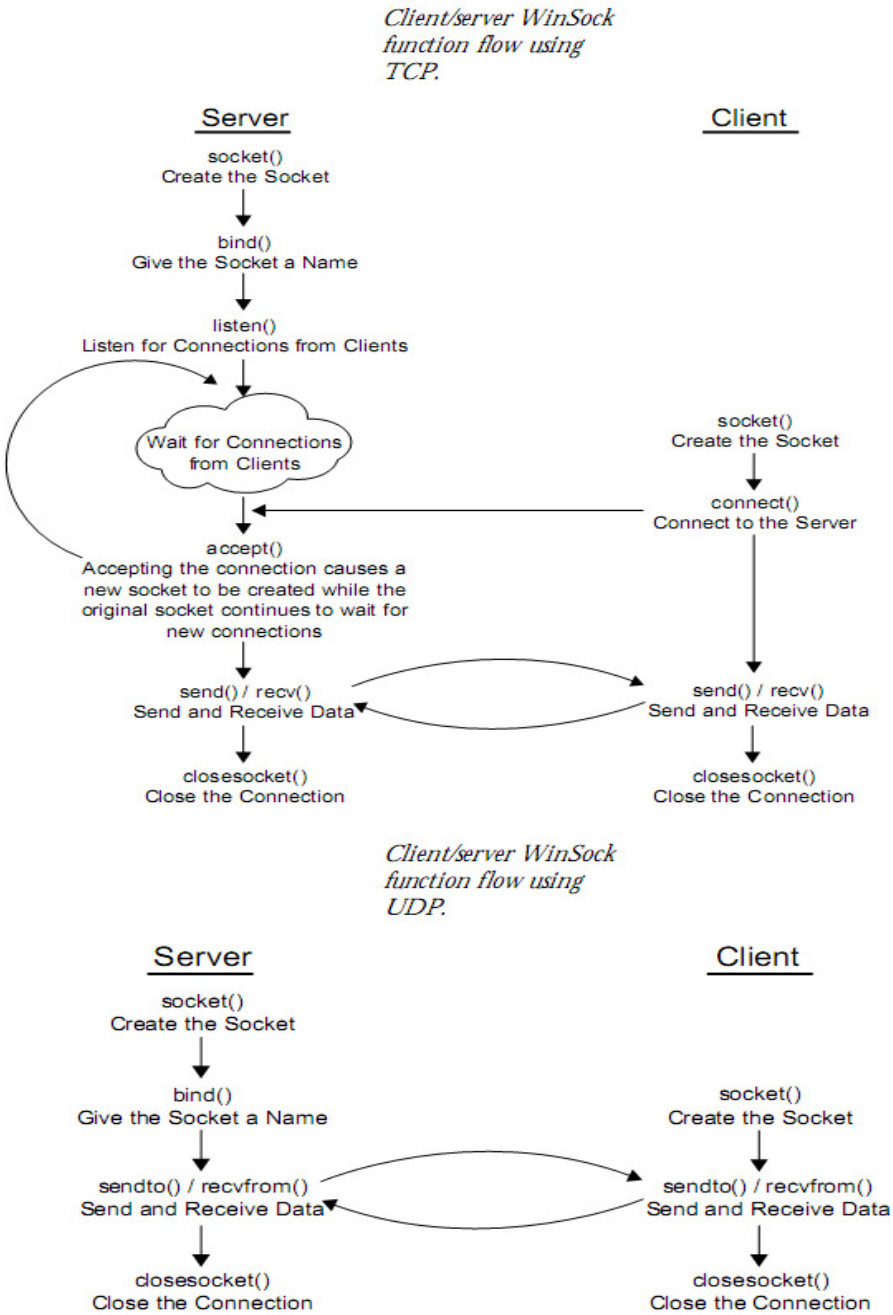

Figura 13 - Comparativo de fluxogramas de comunicação via sockets – TCP x UDP

Para criar essa abstração computacional chamada interface *socket*, são necessárias algumas classes de funções em C. Nesse projeto será usada uma biblioteca de funções denominada *WinSock*. Essa biblioteca reúne todas as diretivas necessárias para a descrição de um *socket* no sistema operacional Windows. Ela auxiliará na construção de uma estrutura no computador capaz de estabelecer uma comunicação com a plataforma de testes.

Essa ferramenta usa conceitos de edição de arquivos em sistemas operacionais para desempenhar funções de envio e recebimento de pacotes. Quando há uma requisição de um arquivo por um aplicativo, o sistema responde criando um descritor de arquivos. Esse descritor pode ser visto como uma tabela de ponteiros que apontam para estrutura de dados, um para cada arquivo; como está demonstrado na Figura 14. Isso permite ao aplicativo fazer acessos ao arquivo desejado.

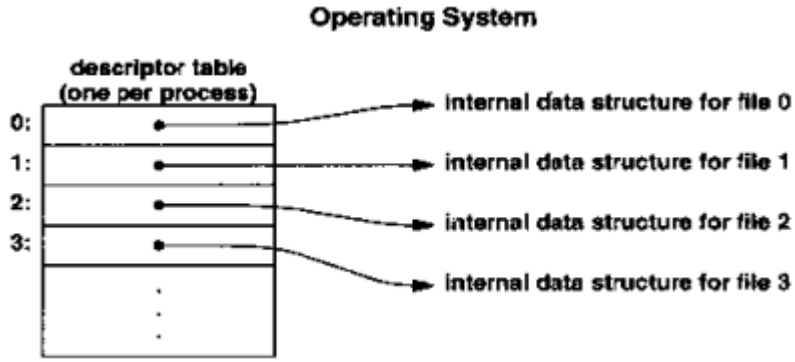

Figura 14 - Tabela descritora de arquivos

Assim como arquivos, cada *socket* ativo é identificado por um descritor de *sockets*. O sistema operacional Windows mantém uma tabela de descritores de *socket* separada para cada processo, vide Figura 15. Assim, um aplicativo trata descritores de arquivos e *sockets* da mesma maneira. Através de uma chamada de função da biblioteca *WinSock* é possível facilmente criar um *socket*.

Depois de sua criação, é necessário por parte do aplicativo, especificar detalhes para seu uso exato, visto que o *socket* é uma entidade genérica e configurável. Dentre os parâmetros de configuração, os mais importantes são: a classe de protocolos a ser utilizada e o endereço e porta de origem e o endereço e porta de destino. Na linguagem C, isso é definido em uma estrutura chamada *sockaddr_in*.

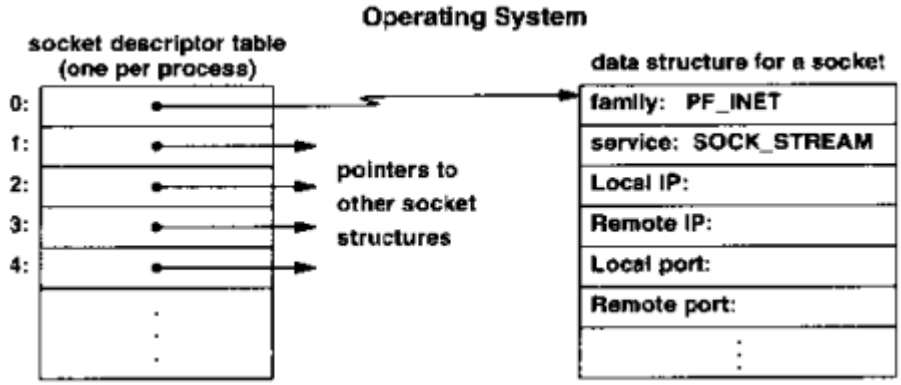

Figura 15 - Tabela descritora de sockets e estrutura de dados do socket

Após as configurações iniciais o *socket* está pronto para ser utilizado. Através de poucas funções é possível estabelecer envio e recebimento de dados utilizando protocolo UDP, como pode ser visto na Figura 13 anteriormente.

3 ANÁLISE DE ALTERNATIVAS

Para o desenvolvimento de uma interface Ethernet na plataforma do decodificador H.264 existiam duas alternativas: implementar um dispositivo MAC totalmente em hardware descrito em VHDL ou usar *IP-core* da biblioteca da ferramenta Xilinx Platform Studio, assim como a interface de comunicação serial já existente.

As vantagens da primeira opção eram inúmeras: poderia ser descrito um hardware específico e dedicado ao envio e recebimento de datagramas UDP; esse módulo poderia ser conectado diretamente à entrada do decodificador deixando-o independente de qualquer barramento ou processador. Porém a complexidade da tarefa não era compatível com o tempo disponível para execução do projeto e então foi descartada.

Restou, então, a segunda alternativa. A adição de um *IP-core* através da ferramenta é bastante simples e pouco sujeita a erros. Entretanto, existem algumas desvantagens no seu uso como a falta de flexibilidade e a necessidade de que esse periférico esteja conectado ao processador de alguma forma. Ou seja, deverá ser criada uma rotina em software para o controle das ações sobre esse periférico.

Dentre os possíveis cores da biblioteca da Xilinx, foi escolhido o OPB EthernetLite. Ele disponibiliza uma interface com conexão via barramento OPB e possui apenas funcionalidades básicas. Isso significa uma facilidade maior no seu controle via software e, ao mesmo tempo, uma menor utilização dos recursos do FPGA Virtex-II Pro.

4 MÉTODOS, PROCESSOS E DISPOSITIVOS

O primeiro passo foi adicionar o *IP-core* OPB EthernetLite ao design já existente do Projeto Integração. Para isso, foram desempenhadas as seguintes etapas: adição do novo periférico ao barramento OPB de comunicação com o processador; definição do range de endereços, através dos quais o processador PowerPC irá gerenciar o dispositivo e associar as nets de interface com o PHY (dispositivo que faz a conexão entre a camada de enlace e a camada física do modelo OSI) com os pinos de I/O do FPGA específicos para esse fim.

Essa associação é feita através de um arquivo de *constraints*, que nada mais é do que uma lista de restrições utilizada pela ferramenta de síntese. Além de associar os pinos com as nets, essa lista contém informações sobre o tipo de sinal que será utilizado. As seguintes linhas foram adicionadas ao arquivo de *constraints* do Projeto Integração:

```
Net opb_ethernetlite_0_PHY_slew1_pin LOC=B3;
Net opb_ethernetlite_0_PHY_slew1_pin IOSTANDARD = LVTTL;
Net opb_ethernetlite_0_PHY_slew1_pin SLEW = SLOW;
Net opb_ethernetlite_0_PHY_slew1_pin DRIVE = 8;
Net opb_ethernetlite_0_PHY_slew2_pin LOC=A3;
Net opb_ethernetlite_0_PHY_slew2_pin IOSTANDARD = LVTTL;
Net opb_ethernetlite_0_PHY_slew2_pin SLEW = SLOW;
Net opb_ethernetlite_0_PHY_slew2_pin DRIVE = 8;
Net opb_ethernetlite_0_PHY_rst_n_pin LOC=G6;
Net opb_ethernetlite_0_PHY_rst_n_pin IOSTANDARD = LVTTL;
Net opb_ethernetlite_0_PHY_rst_n_pin SLEW = SLOW;
Net opb_ethernetlite_0_PHY_rst_n_pin DRIVE = 8;
Net opb_ethernetlite_0_PHY_crs_pin LOC=C5;
Net opb_ethernetlite_0_PHY_crs_pin IOSTANDARD = LVTTL;
Net opb_ethernetlite_0_PHY_col_pin LOC=D5;
Net opb_ethernetlite_0_PHY_col_pin IOSTANDARD = LVTTL;
Net opb_ethernetlite_0_PHY_tx_data_pin<3> LOC=C2;
Net opb_ethernetlite_0_PHY_tx_data_pin<3> IOSTANDARD = LVTTL;
Net opb_ethernetlite_0_PHY_tx_data_pin<3> SLEW = SLOW;
Net opb_ethernetlite_0_PHY_tx_data_pin<3> DRIVE = 8;
Net opb_ethernetlite_0_PHY_tx_data_pin<2> LOC=C1;
Net opb_ethernetlite_0_PHY_tx_data_pin<2> IOSTANDARD = LVTTL;
Net opb_ethernetlite_0_PHY_tx_data_pin<2> SLEW = SLOW;
Net opb_ethernetlite_0_PHY_tx_data_pin<2> DRIVE = 8;
Net opb_ethernetlite_0_PHY_tx_data_pin<1> LOC=J8;
Net opb_ethernetlite_0_PHY_tx_data_pin<1> IOSTANDARD = LVTTL;
Net opb_ethernetlite_0_PHY_tx_data_pin<1> SLEW = SLOW;
Net opb_ethernetlite_0_PHY_tx_data_pin<1> DRIVE = 8;
Net opb_ethernetlite_0_PHY_tx_data_pin<0> LOC=J7;
Net opb_ethernetlite_0_PHY_tx_data_pin<0> IOSTANDARD = LVTTL;
```

```
Net opb_ethernetlite_0_PHY_tx_data_pin<0> SLEW = SLOW;
Net opb_ethernetlite_0_PHY_tx_data_pin<0> DRIVE = 8;
Net opb_ethernetlite_0_PHY_tx_en_pin LOC=C4;
Net opb_ethernetlite_0_PHY_tx_en_pin IOSTANDARD = LVTTL;
Net opb_ethernetlite_0_PHY_tx_en_pin SLEW = SLOW;
Net opb_ethernetlite_0_PHY_tx_en_pin DRIVE = 8;
Net opb_ethernetlite_0_PHY_tx_clk_pin LOC=D3;
Net opb_ethernetlite_0_PHY_tx_clk_pin IOSTANDARD = LVTTL;
Net opb_ethernetlite_0_PHY_rx_er_pin LOC=J2;
Net opb_ethernetlite_0_PHY_rx_er_pin IOSTANDARD = LVTTL;
Net opb_ethernetlite_0_PHY_rx_clk_pin LOC=M8;
Net opb_ethernetlite_0_PHY_rx_clk_pin IOSTANDARD = LVTTL;
Net opb_ethernetlite_0_PHY_dv_pin LOC=M7;
Net opb_ethernetlite_0_PHY_dv_pin IOSTANDARD = LVTTL;
Net opb_ethernetlite_0_PHY_rx_data_pin<0> LOC=K6;
Net opb_ethernetlite_0_PHY_rx_data_pin<0> IOSTANDARD = LVTTL;
Net opb_ethernetlite_0_PHY_rx_data_pin<1> LOC=K5;
Net opb_ethernetlite_0_PHY_rx_data_pin<1> IOSTANDARD = LVTTL;
Net opb_ethernetlite_0_PHY_rx_data_pin<2> LOC=J1;
Net opb_ethernetlite_0_PHY_rx_data_pin<2> IOSTANDARD = LVTTL;
Net opb_ethernetlite_0_PHY_rx_data_pin<3> LOC=K1;
Net opb_ethernetlite_0_PHY_rx_data_pin<3> IOSTANDARD = LVTTL;
```

4.1 – OPB ETHERNETLITE MEDIA ACCESS CONTROLLER

Esse módulo de propriedade intelectual (IP) da Xilinx pode ser dividido em dois blocos: o EthernetMAC e o OPB IPIF. O primeiro faz a interface com o dispositivo PHY e provém funcionalidades para o envio e recebimento de dados através da rede Ethernet, já o segundo adapta o bloco EthernetMAC permitindo acessibilidade através do barramento OPB [Xilinx, 2006]. O diagrama da Figura 16 demonstra a disposição dos blocos citados acima no contexto desse projeto e suas interconexões.

Através de um *driver* escrito na linguagem C (XEmacLite), é possível gerenciar o módulo OPB EthernetLite. Esse *driver* descreve funções básicas para configuração, verificação de status, envio e recebimento de dados. Então, basta referenciar os arquivos que descrevem esse *driver* ao código fonte do processador PowerPC para que se crie a possibilidade de um controle das ações do módulo via software.

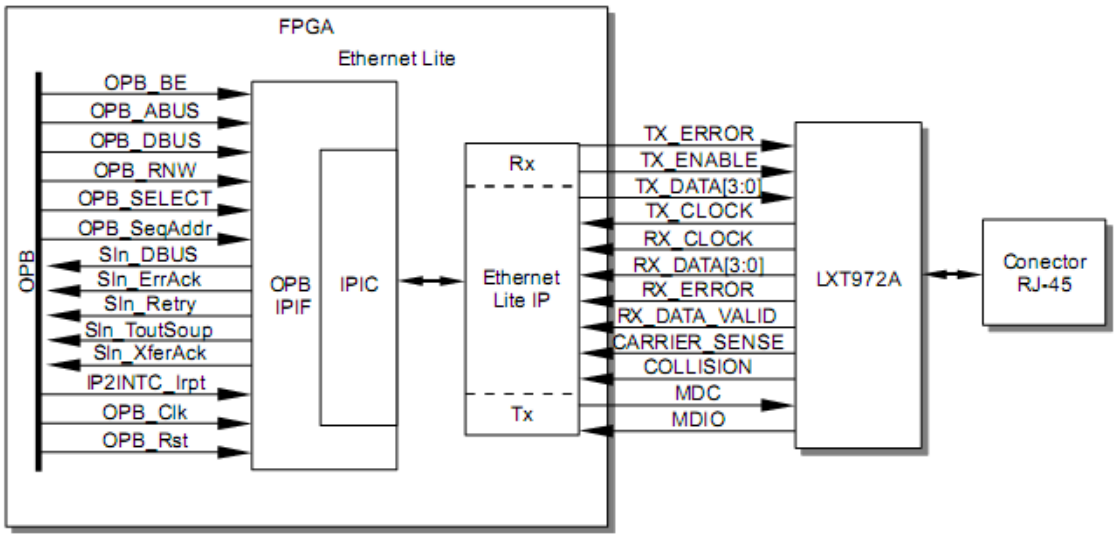

Figura 16 - Composição do IP-core OPB EthernetLite

Porém, o driver XEmacLite dá suporte apenas a camadas de baixo nível de abstração, como a camada de enlace (referência ao modelo OSI) por exemplo. Ou seja, seria necessário um esforço manual a fim de fazer a adequação para uma camada de mais alto nível, como a camada de transporte onde se definem os protocolos UDP e TCP. Fez-se necessário então, o uso de uma biblioteca que agrega funcionalidades de mais alto nível chamada XilNet.

4.2 – LIB XILNET

Essa é uma biblioteca de funções de uso em redes descrita na linguagem C, que é especifica para processadores embarcados. A *XilNet* possui suporte para uso de *sockets*, como anteriormente citado, um elemento computacional indispensável para aplicações de alto nível em redes. A sintaxe e estrutura de suas funções e constantes faz com que a *XilNet* se assemelhe bastante com a *WinSock*. Além disso, a compatibilidade entre elas é quase que total, o que as certifica para atuar na comunicação entre a placa e o computador.

4.3 – INFORMAÇÕES COMPLEMENTARES

Nesse processo de adição de componentes ao projeto que já existia, é importante levar em consideração nesse tipo de prototipação a utilização de recursos do FPGA. Ele é um dispositivo que tem características importantes como flexibilidade e desempenho, entretanto seus recursos não são ilimitados.

As informações que seguem dizem respeito ao Projeto Integração na estrutura original:

```
Device Utilization Summary:

   Number of BUFGMUXs                        9 out of 16     56%
   Number of DCMs                            3 out of 8      37%
   Number of External IOBs                 136 out of 556    24%
      Number of LOCed IOBs                 136 out of 136   100%

   Number of JTAGPPCs                        1 out of 1     100%
   Number of MULT18X18s                      4 out of 136     2%
   Number of PPC405s                         1 out of 2      50%
   Number of RAMB16s                        75 out of 136    55%
   Number of SLICEs                      12532 out of 13696  91%
```

Dentre esses valores, destaca-se a utilização de SLICEs do FPGA que fica em 91%. Um SLICE é a menor unidade lógica configurável de um FPGA. Ou seja, de maneira geral pode-se dizer que o projeto ocupa 91% da área utilizável do dispositivo FPGA, o que representa um valor considerável.

Com a adição do *IP-core* OPB EthernetLite no design, esses valores de utilização de recursos do FPGA mudaram e serão exibidos na seqüência:

```
Device Utilization Summary:

   Number of BUFGMUXs                        9 out of 16     56%
   Number of DCMs                            3 out of 8      37%
   Number of External IOBs                 153 out of 556    27%
      Number of LOCed IOBs                 153 out of 153   100%

   Number of JTAGPPCs                        1 out of 1     100%
   Number of MULT18X18s                      4 out of 136     2%
```

```
Number of PPC405s                          1 out of 2      50%
Number of RAMB16s                         77 out of 136    56%
Number of SLICEs                       12899 out of 13696  94%
```

Foram consumidos apenas 3% dos SLICEs, 2 Block RAMs (são elementos de memória no interior do FPGA) e 17 pinos de I/O para conexão com o dispositivo PHY. Esses valores encontram-se dentro da faixa prevista pelo Data Sheet do IP-core adicionado.

A partir desse momento, tornava-se possível a comunicação da placa via rede Ethernet com qualquer outro elemento disponível na rede. No entanto, outras ações ainda precisavam ser realizadas. Era necessária uma estrutura de memória para compatibilizar a taxa de recepção de dados pela plataforma e a taxa de utilização dos mesmos pelo processo de decodificação H.264.

Existiam duas soluções passíveis de implementação: poderia se usar uma arquitetura com dois processadores (o FPGA Virtex-II Pro possui dois processadores PowerPC internos) trabalhando em paralelo ou então, com apenas um processador, implementar uma FIFO, o que acarretaria uma mudança severa na rotina de controle do decodificador H.264. Essas duas propostas serão mais bem descritas nos capítulos a seguir.

4.4 – SOLUÇÃO 1: PARALELISMO ENTRE PROCESSADORES

A grande vantagem da escolha de utilização de dois processadores seria um aumento considerável no desempenho da futura porta de vídeo codificado em H.264. Um processador ficaria totalmente dedicado a recepção e controle de fluxo dos dados recebidos pela interface Ethernet e o outro teria a função apenas de controlar o funcionamento do sistema de decodificação de vídeo.

Porém, algumas questões precisariam ser resolvidas. A primeira e mais importante é a questão da utilização de recursos do FPGA. Para a implementação de uma arquitetura com

duplo processamento seria necessária uma estrutura que suportasse essa nova disposição. Um novo barramento PLB precisaria ser adicionado, visto que os dois processadores não conseguem compartilhar o mesmo. A utilização dos elementos de memória seria maior, pois os dois processadores necessitam memória de programa e de dados individual. Talvez, alguns periféricos precisem ser compartilhados, como por exemplo, a interface de comunicação serial (UART). Para isso seriam necessárias estruturas *bridges* que fazem a interconexão entre diferentes barramentos e também um padrão de sincronização para que os processadores não acessem os periféricos compartilhados ao mesmo tempo [Xilinx, 2007]. A Figura 17 exemplifica a arquitetura descrita acima.

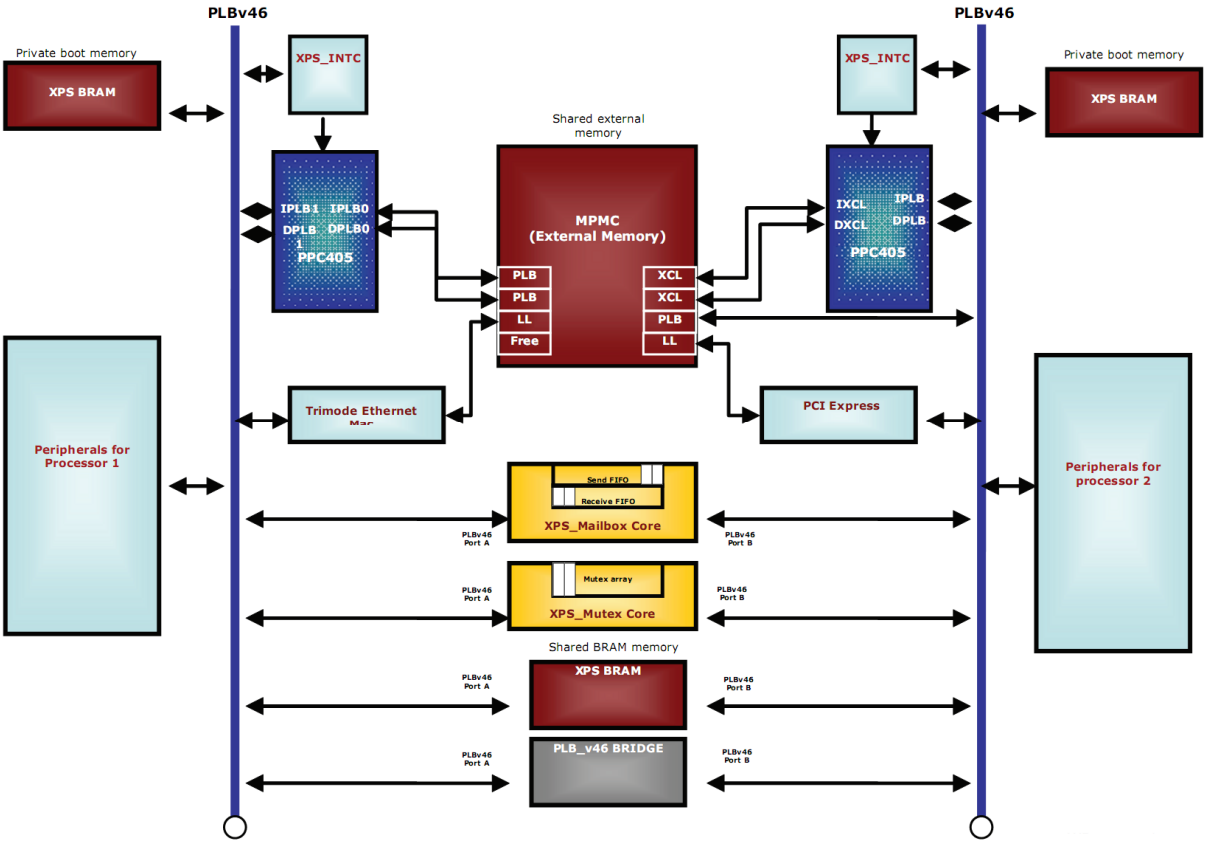

Figura 17 - Arquitetura de um sistema com processamento dual

4.5 – UTILIZAÇÃO DE UMA FIFO E APENAS UM PROCESSADOR

A escolha por essa proposta reduz todos os problemas com a utilização de recursos do FPGA relatados anteriormente. Entretanto, sua implementação não é nada trivial, uma vez que seria necessária a alteração da rotina de software que controla o decodificador H.264. Muito provavelmente seja necessário o uso de um sistema baseado em interrupções no processador. Esses sinais de interrupção seriam gerados pela FIFO quando algum dos seus limites fosse ultrapassado. Então, ao perceber essa sinalização, o processador prioriza a rotina de atendimento a essa interrupção o que pode causar uma descontinuidade no processo de decodificação de vídeo.

Além disso, o nível de complexidade da rotina que faz o controle do decodificador é um aspecto a ser levado em consideração. Isso acontece, pois os módulos que compõem o decodificador H.264 não foram dispostos em uma estrutura denominada *pipeline*. Isto é, diferente do que é mostrado na Figura 6. Assim, a rotina de controle precisa incorporar características individuais de cada módulo, tornando-a muito mais complexa. A Figura 18 demonstra a estrutura do decodificador H.264.

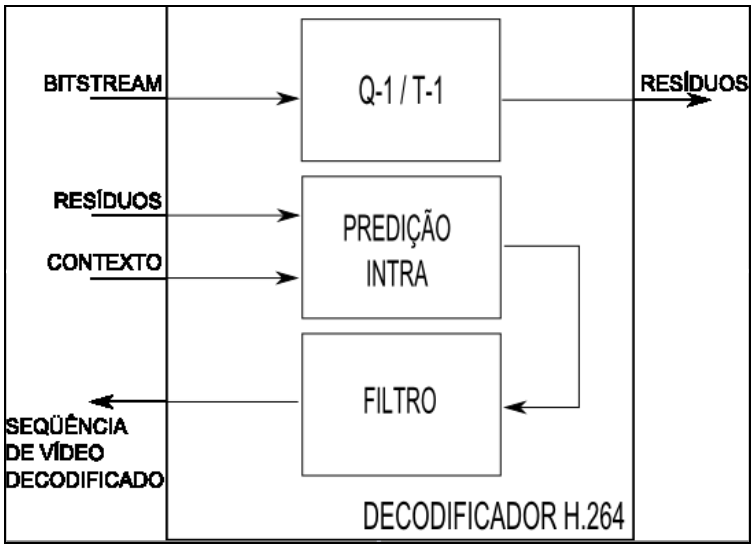

Figura 18 - Estrutura do decodificador H.264

Dentre as alternativas de implementação, foi escolhida a que representava uma arquitetura com processamento dual. Entretanto, utilizou-se uma estrutura mais simples que a exemplificada na Figura 17. Os elementos do Projeto Integração original foram todos mantidos e alguns novos periféricos adicionados como mostra a figura a seguir.

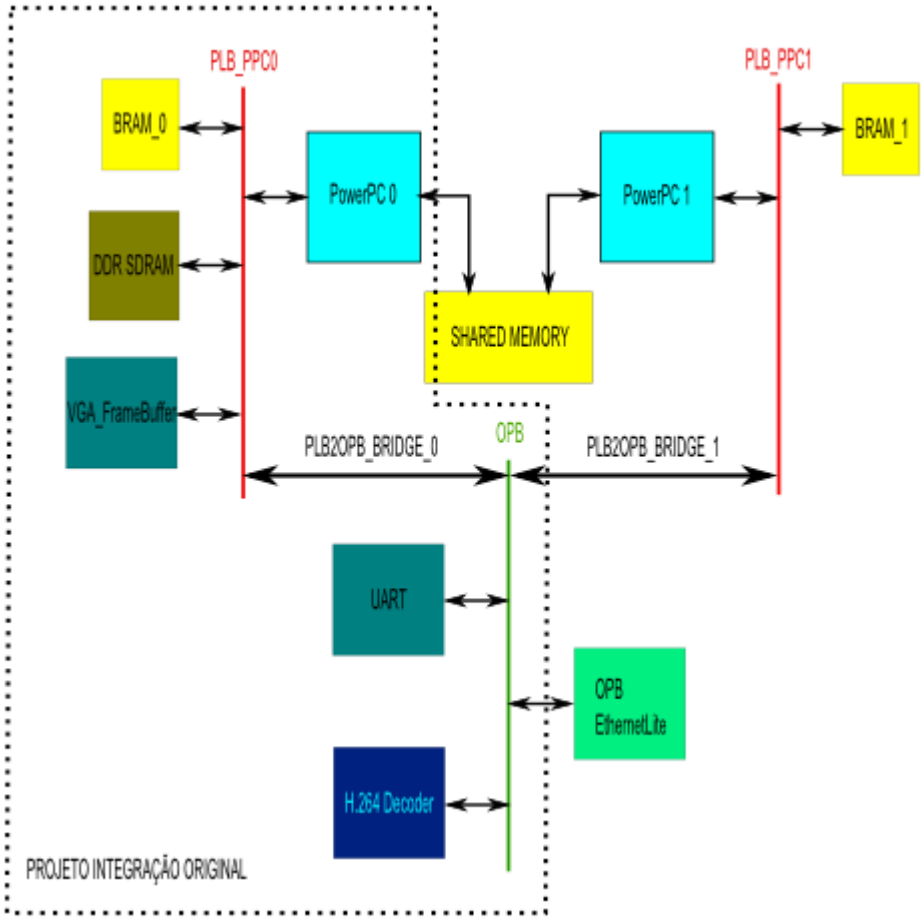

Figura 19 - Diagrama de blocos final do projeto

Nessa fase do projeto fez-se a opção de apenas validar essa nova arquitetura proposta e não mais implementar uma porta de vídeo de fato. Essa decisão foi tomada, pois a complexidade envolvida no controle do H.264 Decoder já ultrapassava os limites da abordagem desse projeto. Porém, a validação dessa nova arquitetura é muito importante, porque ela garante uma excelente plataforma para desenvolvimento de projetos futuros de alto desempenho, como a porta de vídeo codificado em H.264.

5 RESULTADOS ALCANÇADOS

Depois de feita a integração final do sistema, alguns aspectos merecem destaque e serão apresentados a seguir.

5.1 – UTILIZAÇÃO DOS RECURSOS DO FPGA

Depois da inclusão de todos os periféricos necessários no projeto, os resultados de utilização de recursos são esses:

```
Device Utilization Summary:

   Number of BUFGMUXs                        8 out of 16     50%
   Number of DCMs                            3 out of 8      37%
   Number of External IOBs                 153 out of 556    27%
      Number of LOCed IOBs                 153 out of 153   100%

   Number of JTAGPPCs                        1 out of 1     100%
   Number of MULT18X18s                      4 out of 136     2%
   Number of PPC405s                         2 out of 2     100%
   Number of RAMB16s                       109 out of 136    80%
   Number of SLICEs                      13694 out of 13696  99%
```

5.2 – VALIDAÇÃO DA INTERFACE ETHERNET

A validação da interface Ethernet de comunicação foi feita da seguinte forma. A plataforma de desenvolvimento foi conectada ao computador através de um cabo “cross”, que possui os pinos TX/RX invertidos nas extremidades, o que permite uma comunicação direta via Ethernet dos dois dispositivos. Com o auxílio da linguagem de programação C foi desenvolvida uma aplicação que era capaz de mandar uma quantidade de datagramas UDP variável de tamanho também variável para outro elemento identificado da rede. Esse aplicativo só enviava o datagrama seguinte depois que recebesse uma mensagem de confirmação do correto recebimento do pacote anterior.

A rotina de software do processador PowerPC embarcado, apenas inicializava os dispositivos da placa e monitorava a rede na espera por datagramas. Quando o recebimento de um pacote era bem sucedido, uma mensagem de confirmação era enviada ao elemento da rede remetente.

Foram realizados testes onde eram enviados pacotes com informação correspondente a vídeo codificado em H.264. No total foram enviados 2537 pacotes com 1024 bytes de dados do aplicativo cliente no computador (10.1.25.222) em direção ao endereço de IP 10.1.25.200, o qual representava a placa de desenvolvimento. Foi usado um software de monitoramento de rede para certificar o envio e recebimento correto dos pacotes. Na Figura 20 é possível observar os resultados do monitoramento.

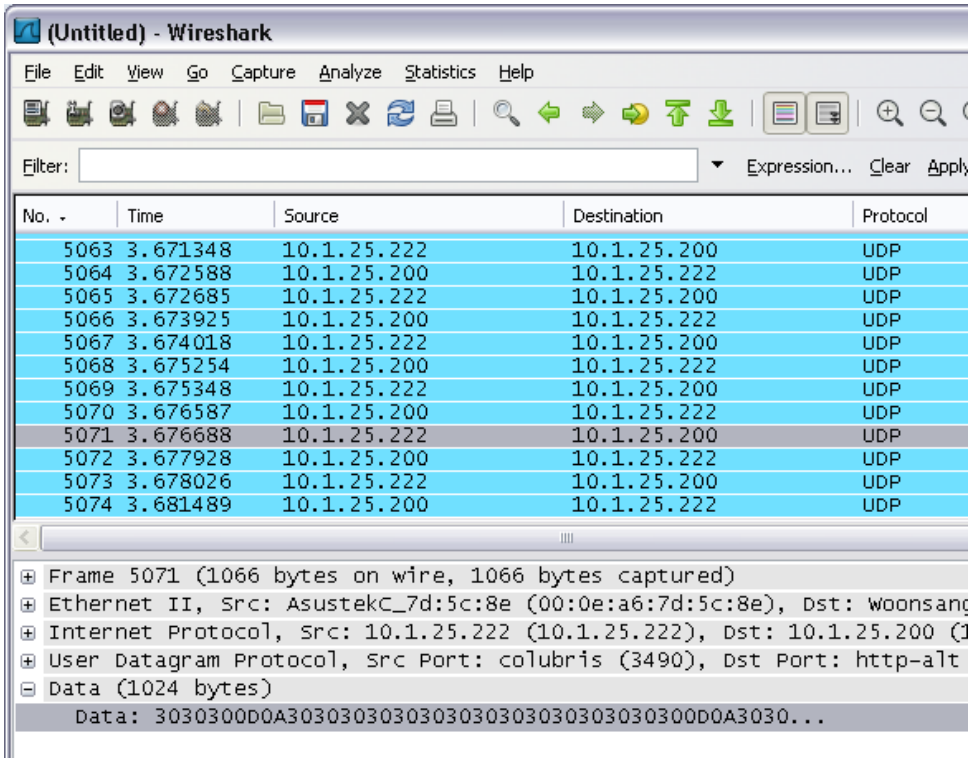

Figura 20 - Demonstração do pacote com 1024 bytes enviado

Apesar de a figura mostrar um número superior a 5000 pacotes, deve se desconsiderar os relacionados com as mensagens de confirmação de recebimento. Então, o tempo gasto para

o envio de 2537 pacotes com 1024 bytes de informação foi de aproximadamente 3,68s, o que resultou em uma taxa média de 5,6Mbps.

5.3 – VALIDAÇÃO DO USO DA MEMÓRIA COMPARTILHADA

A validação desse quesito foi feita através de depuração do software via interface serial. Os dados que chegavam pela interface Ethernet eram armazenados na memória compartilhada e automaticamente lidos por uma rotina descrita no segundo processador. Assim, comparando com a informação original era possível verificar alguma falha no acesso à memória, o que não foi verificado. A figura a seguir demonstra a execução dessa verificação.

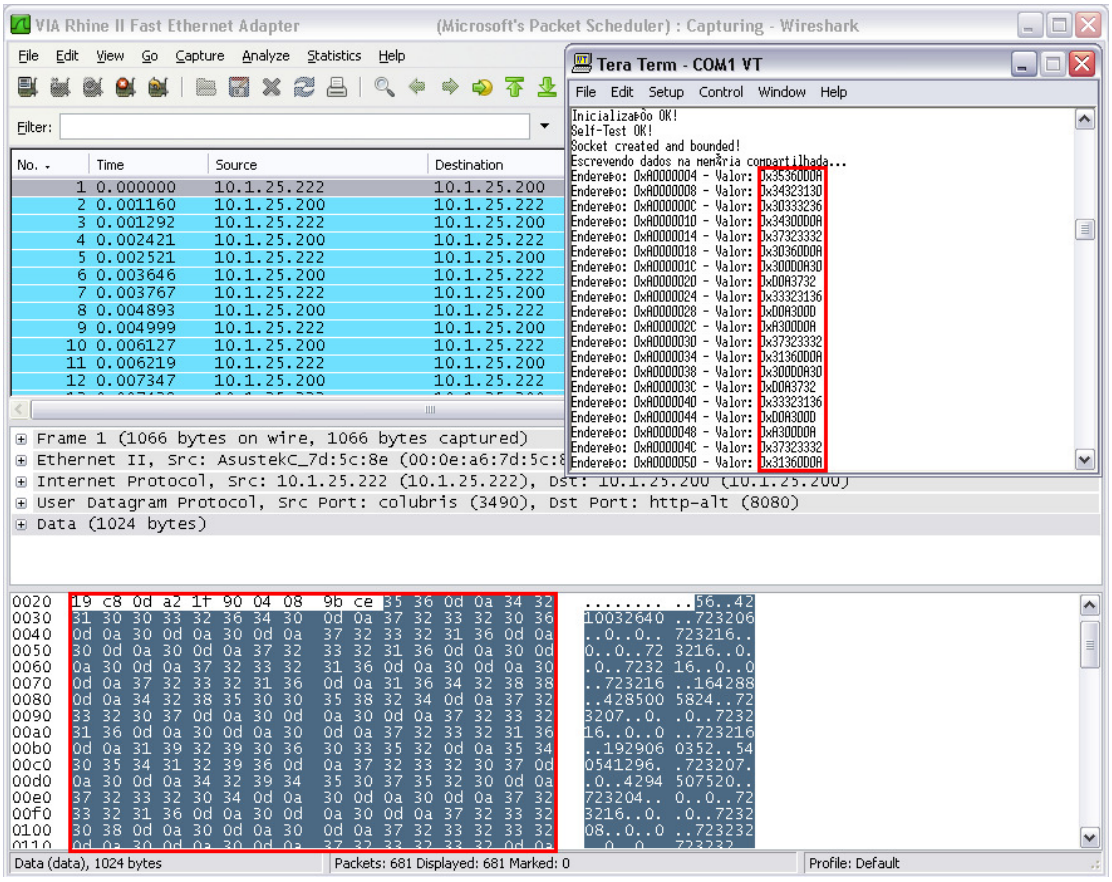

Figura 21 - Verificação do uso da memória compartilhada

6 CONCLUSÃO

Apesar do grande esforço, não foi possível atingir o objetivo inicial desse projeto. Algumas dificuldades encontradas ao longo de sua execução foram decisivas, porém o resultado atingido foi bastante satisfatório. Pode-se destacar a eficiência da comunicação via rede Ethernet entre a placa e o computador. O uso de sockets realmente foi imprescindível, pois tornou muito simples a descrição de um sistema de transmissão de dados através de aplicativos.

Além disso, o uso de uma arquitetura com duplo processamento proporciona uma grande eficiência e uma possibilidade para o desenvolvimento de aplicações com alto desempenho. Futuramente, contando com uma estrutura de decodificação que favoreça, isto é, que precise menos intervenções externas para o seu funcionamento, é perfeitamente possível a construção dessa porta de vídeo. O que será um fator que agregará muito para o desenvolvimento arquitetural do decodificador, uma vez que se transformará em uma excelente plataforma de testes.

Essa arquitetura com processamento dual também é importante para a resolução da questão do controle no fluxo de dados entre a recepção da porta Ethernet e a entrada do decodificador H.264. Pode-se fazer uso de uma memória compartilhada entre os processadores como um pequeno buffer para a adequação das taxas de recebimento de dados e de entrada do decodificador H.264. Também é possível o uso de um controle adaptativo. Como há uma comunicação bidirecional entre cliente (PC) e servidor (placa protótipo), pode-se fazer com que o tamanho dos pacotes enviados seja inversamente proporcional ao nível de ocupação da memória "buffer".

Nesse caso, a memória exerce uma função semelhante a de uma FIFO. O servidor, na mensagem de confirmação de recebimento do frame, informaria ao aplicativo do cliente a situação corrente de ocupação da memória compartilhada. Então, com esse conhecimento, o

aplicativo redimensionaria o tamanho dos pacotes enviados adaptando a taxa de transmissão. No momento em que a memória for pouco utilizada, transmitem-se dados a uma taxa maior (usando pacotes de maior tamanho) e na situação inversa utiliza-se uma taxa menor fazendo com que o nível de ocupação dessa memória atinja valores aceitáveis através do consumo de informações do decodificador.

Esse projeto foi interessante no ponto de vista de aprendizado, pois permite ao aluno desenvolver temas de sua preferência e que nem sempre foram amplamente abordados durante a graduação. Por isso, pode-se concluir que o Projeto de Diplomação correspondeu plenamente às expectativas.

7 REFERÊNCIAS BIBLIOGRÁFICAS

[Sockets, 1996] *Bonner, Pat, "Network Programming with Windows Sockets",* Prentice-Hall, Upper SaddleRiver, NJ, 1996, ISBN: 0-13-230152-0

[Stevens, 2000] *D. E. Comer and D.L. Stevens, "Internetworking with TCP/IP Vol. III Client-Server Programming and Applications-Windows Sockets Version",* Prentice-Hall, 2000, ISBN: 0138487146

[TCPIP, 1996] *W. Richard Stevens, Addison-Wesley, "TCP/IP Illustrated, Volume 1: The Protocols"*, 1996, ISBN 0-201-63346-9.

[Richardson, 2003] *I. Richardson.* "*H.264 and MPEG4 Video Compression"*. John Wiley and Sons, 2003.

[Xilinx, 2004] XILINX, INC, *"XUP Virtex-II Pro development system",* 2004, 38p. Disponível em <http://www.xilinx.com/univ/XUPV2P/Documentation/EXTERNAL_REV_C_SCHEMATICS.pdf>. Acesso em setembro de 2008.

[Xilinx, 2005] XILINX, INC, *"Xilinx University Program - Virtex-II Pro Development System"*, 2005, 138p. Disponível em <http://www.xilinx.com/univ/XUPV2P/Documentation/XUPV2P_User_Guide.pdf>. Acesso em setembro de 2008.

[Xilinx, 2006] XILINX, INC, *"OPB Ethernet Lite Media Access Controller (v1.01b) DataSheet"*, 2006, 23p. Disponível em <http://www.xilinx.com/support/documentation/ip_documentation/opb_ethernetlite.pdf>. Acesso em setembro de 2008.

[Xilinx, 2007] XILINX, INC, *"Designing Multiprocessor Systems in Platform Studio",* 2007, 18p. Disponível em <www.xilinx.com/support/documentation/white_papers/wp262.pdf>. Acesso em outubro de 2008.