INFRAESTRUTURA PARA CONTROLE DE PROJETOS EM FPGAS ATRAVÉS DO PROTOCOLO ETHERNET

Jeferson Santiago da Silva

Faculdade de Engenharia – PUCRS
Departamento de Engenharia Elétrica
jeferson.santiago@acad.pucrs.br

Fernando Gehm Moraes

Faculdade de Informática – PUCRS
Departamento Fundamentos da Computação
fernando.moraes@pucrs.br

RESUMO
Os dispositivos FPGAs (*Field Programmable Gate Array*) estão frequentemente presentes em projetos de sistemas digitais que requerem alto desempenho e um custo relativamente baixo. É comum, atualmente, nos depararmos com tal tecnologia e o presente trabalho propõe uma forma de configuração e monitoramento de dados destes dispositivos, através de uma rede Ethernet, usando um computador remoto. O objetivo deste trabalho é desenvolver, em *hardware*, os protocolos básicos de acesso à rede utilizando a pilha TCP/IP (*Transmission Control Protocol - Internet Protocol*). Uma versão alternativa semelhante ao TCP e o DHCP (*Dynamic Host Configuration Protocol*) foram desenvolvidos e validados nesse trabalho.

PALAVRAS-CHAVE: FPGA; redes de computadores; protocolos de comunicação, monitoramento.

ABSTRACT
FPGAs are often present in the design of digital systems that require high performance and a relatively low cost. It is common, nowadays, to come across this technology and this work proposes a way of configuring and monitoring data of these elements, through an Ethernet network, using a remote computer. The objective is to develop, in hardware, the basic protocols for network access using TCP/IP stack. An alternative version of TCP and DHCP protocols are developed and implemented in this work.

KEY-WORDS: FPGA, computer networks, communication protocols, monitoring.

1. INTRODUÇÃO

Com o crescente aumento na demanda por processamento, é cada dia mais frequente o uso de FPGAs para desempenhar tarefas que necessitam de alta velocidade. Os FPGAs são encontrados nas mais variadas aplicações, como sistemas de comunicação de dados, automação e aplicações espaciais.

Este trabalho visa disponibilizar uma infraestrutura de controle de projetos baseados em FPGA usando o protocolo Ethernet. A estrutura implantada é do tipo TCP/IP, a qual é caracterizada pela confiabilidade e robustez na transferência de dados. Toda arquitetura descrita no decorrer desse trabalho foi realizada em hardware reconfigurável, utilizando linguagem de descrição de hardware VHDL e prototipada em FPGA. Para diminuir os recursos de *hardware*, esse projeto é descrito basicamente utilizando FSMs (*Finite State Machine*).

O presente trabalho utilizou como base a arquitetura de rede descrita por Reinbrecht (REINBRECHT *et al*, 2009), o qual implementa as camadas da pilha TCP/IP, baseada no protocolo de transporte UDP (*User Datagram Protocol*). Este trabalho reutiliza os blocos que descrevem o MAC (*Media Access Control*) e o ARP (*Address Resolution Protocol*). As camadas de internet e transporte foram adaptadas para suportar comunicação através do protocolo TCP. Além do mais, foram adicionados outros dois módulos ao projeto, um que descreve o protocolo DHCP e outro que possibilita a descoberta do endereço IP do FPGA a partir do *host*.

O principal objetivo deste TCC é prover acesso remoto a um FPGA, que desempenha alguma tarefa específica, podendo, dessa forma, alterar parâmetros de execução e obter dados provenientes de processamento. Este projeto visa descrever uma estrutura genérica de acesso a rede que possa ser utilizada por diferentes projetos, como MPSoCs (*Multi-Processor Systems on Chip*) que podem ser configurados em tempo de execução, e assim alterar ou enviar novas tarefas aos processadores que o compõem.

É importante salientar que a estrutura que foi fornecida a este FPGA é adaptada, sendo que tanto o protocolo DHCP e o protocolo TCP foram desenvolvidos para esta aplicação específica, logo, não seguem fielmente os padrões propostos pelos seus respectivos manuais técnicos.

Nas seções seguintes apresenta-se o referencial teórico, métodos, resultados e conclusões obtidas em todas as etapas da implementação.

2. REFERENCIAL TEÓRICO

Nessa seção é apresentado o referencial teórico proposto no trabalho. Nela são apresentados o modelo de arquitetura de rede adotado e a definição dos protocolos abordados nesse trabalho. O embasamento teórico é descrito nessa ordem:

trabalhos anteriores, modelos de referência, protocolo IP, protocolo TCP e protocolo DHCP.

2.1. Trabalhos anteriores

Ao longo do desenvolvimento desse trabalho, outros projetos foram estudados e analisados, servindo como referência. Dentre os trabalhos que tratam sobre o tema abordado por esse projeto, três chamam mais atenção, pela complexidade e forma de implementação.

Reinbrecht (REINBRECHT *et al*, 2009) propôs uma estrutura de acesso remoto a um FPGA utilizando como base o protocolo de transporte UDP. Essa estrutura foi desenvolvida em *hardware* reconfigurável, simulada e prototipada em FPGA. O principal objetivo desse trabalho era fornecer acesso remoto a um MPSoC, denominado HeMPs, descrito em VHDL, executando em FPGA. HeMPs é um MPSoC composto por vários processadores interconectados por uma NoC (*Network on Chip*), os quais estão dispostos no FPGA de uma forma tal, que o acesso individual a cada processador pode ser realizado através de um protocolo que o localiza através de um sistema de coordenadas.

O modelo implantando foi desenvolvido a partir de um *IP core* (módulo de propriedade intelectual), disponibilizado pela *Xilinx*, que implementa a comunicação com um circuito externo (*phy*), ligado a uma rede do tipo Ethernet. As camadas superiores da pilha TCP/IP foram descritas em VHDL utilizando FSMs, e os dados de recepção/transmissão foram armazenados em memórias RAM (*Random Access Memory*) de dupla porta, presentes no FPGA.

Um dos grandes desafios desse trabalho foi sincronizar os dados entre os módulos de rede e a interface com a HeMPs, já que as duas estruturas não estão no mesmo domínio de relógio. Essa característica assíncrona dos relógios foi solucionada utilizando memórias RAM de duplo acesso, onde os relógios de escrita e leitura são distintos.

Esse trabalho foi validado através de simulação, utilizando o *software ModelSim*, e em seguida foi embarcado em FPGA para verificação do sistema, onde os sinais foram analisados através do *software ChipScope*.

A maior limitação da arquitetura descrita por Reinbrecht (REINBRECHT *et al*, 2009) é a vulnerabilidade na transferência de dados entre o *host* e o FPGA, pois o protocolo UDP não garante entrega dos dados. Outra limitação observada é a necessidade do FPGA em questão já possuir um endereço IP válido na rede, deixando o processo de comunicação pouco automatizado.

Hamersky (HAMERSKI, 2008), por sua vez, propõe uma estrutura de comunicação com um FPGA baseado em uma pilha TCP/IP parametrizável. Em seu trabalho é apresentada uma estrutura complexa, que implementa grande parte dos protocolos suportados pela arquitetura TCP/IP (até a camada de transporte) em VHDL e o alto da pilha (aplicação) é comandado via *software*.

Neste trabalho é apresentado uma proposta de pilha desenvolvida desde o MAC, passando pelo ARP, IP, ICMP (*Internet Control Message Protocol*), UDP e TCP. Todos esses blocos foram baseados em máquinas de estados, já que é importante manter o caráter sequencial na montagem/desmontagem dos pacotes. Diferentemente de Reinbrecht (REINBRECHT *et al*, 2009), Hamersky (HAMERSKI, 2008) desenvolveu uma arquitetura que suporta múltiplas conexões, tanto TCP quanto UDP, além de possibilidade de retransmissão em caso de perda de pacotes.

Os módulos presentes no projeto, exceto o MAC, estão ligados ao barramento de memória compartilhado, onde há um árbitro que controla as operações de escrita e leitura em uma das portas da memória. O bloco do árbitro também realiza o tratamento dos dados do protocolo IP, assim cada protocolo superior implementa somente a sua efetiva funcionalidade. A outra porta da memória é ligada ao MAC, realizando a comunicação com o *phy*.

As limitações desse trabalho dizem respeito aos recursos de lógica que essa pilha irá ocupar em FPGA. Assim sendo, é necessário limitar o número de conexões simultâneas, o tamanho do *buffer* de retransmissão e o tamanho da tabela ARP.

Dollas (DOLLAS *et al*, 2005) propõe uma arquitetura de rede similar à proposta por Hamersky. Dollas descreveu vários protocolos de rede baseados na pilha TCP/IP em VHDL e prototipou o mesmo em um FPGA *Xilinx Virtex 2*. O trabalho de Dollas tem capacidade de atender até 15 conexões simultâneas, utilizando o protocolo TCP, e essa arquitetura pode estar conectada a múltiplas e distintas aplicações. Já as limitações deste trabalho estão no que diz respeito às normas de descrição dos protocolos, pois é muito difícil atender, em *hardware*, todos os requisitos que são exigidos nestas.

2.2. Modelos de referência: modelo OSI *versus* modelo TCP/IP

Os modelos de arquiteturas de rede que são considerados de referência para este trabalho são o Modelo OSI (*Open Systems Interconnection*) de sete níveis e o modelo TCP/IP, regulamentado no RFC 1180 (SOCOLOFSKY *et al*, 1991).

O modelo OSI de sete níveis foi regulamentado pelo ISO (*International Standards Organization*) e tem por objetivo padronizar o acesso a redes de computadores. Esse padrão estabelece sete níveis de abstração para o tratamento de informações provenientes da rede. Já a arquitetura TCP/IP é anterior ao modelo OSI, tendo características semelhantes a pilha OSI, onde algumas camadas não são implementadas isoladamente, estando presente em outros níveis. A pilha TCP/IP é formada por quatro níveis. A Figura 1 compara as duas arquiteturas de referência.

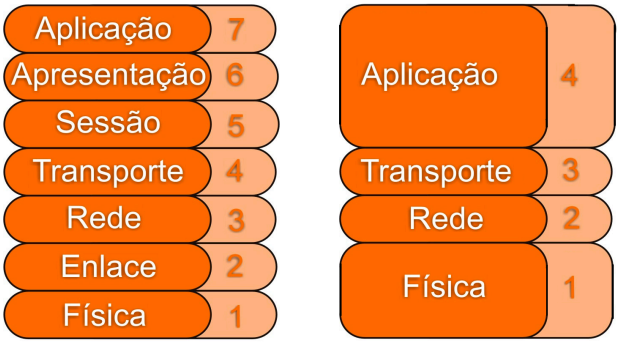

Figura 1 - Estrutura do modelo OSI em sete níveis e modelo TCP/IP.

A arquitetura implantada nesse trabalho é do tipo TCP/IP. A pilha TCP/IP é formada pelas camadas física, de rede (ou internet), de transporte e aplicação.

A camada física, segundo a pilha TCP/IP, agrupa as funcionalidades das camadas 1 e 2 da pilha OSI. A primeira camada TCP/IP é responsável pelo acesso ao meio e a atribuição dos endereços físicos, os quais podem ser obtidos através da tabela ARP.

A camada de rede é bastante similar à descrita pelo modelo OSI. Tem as funcionalidades de endereçamento lógico e roteamento dos pacotes das camadas superiores. Também é nessa camada que é executado o protocolo de controle de internet, o ICMP, o qual não pode ser enquadrado no modelo OSI.

A camada de transporte é responsável pelo estabelecimento de conexão e a montagem dos pacotes por funcionalidade, semelhantemente ao modelo OSI.

A diferença mais visível é na camada de aplicação do modelo TCP/IP, que agrupa os três protocolos do topo da pilha OSI. As camadas de apresentação e sessão são executadas dentro do protocolo de serviço requisitado pelo usuário.

2.3. Protocolo IP

O protocolo IP, regulamentado pelo RFC 791 (POSTEL, 1981), surgiu em setembro de 1981. Inicialmente preparado para fins militares, pelo programa DARPA (*Defense Advanced Research Projects Agency*). O protocolo IP até hoje é o mais usado no nível de camada de internet.

O principal objetivo do protocolo IP é dividir os dados provenientes da camada de transporte em *frames* (quadros), possuindo um mecanismo de controle de fragmentação dos dados. O protocolo IP não implementa nenhum controle de erros ou fluxo. Este controle é responsabilidade das camadas superiores. Em nível de internet há um protocolo, concorrente ao IP, que pode reportar erros, o protocolo ICMP, descrito pelo RFC 792 (POSTEL, 1981).

O quadro do protocolo IP é formado por 12 campos fixos, que tem por finalidade indicar as camadas inferiores e superiores os protocolos, máquinas e serviços envolvidos na comunicação. A Figura 2 apresenta o formato do quadro IP.

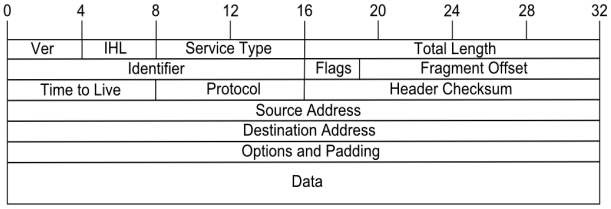

Figura 2 - Quadro IP.

2.4. Protocolo TCP

O protocolo TCP, descrito pelo RFC 793 (POSTEL, 1981) tem a mesma origem e é contemporâneo ao protocolo IP. Surgiu como concorrente ao protocolo UDP, sendo que sua principal vantagem sobre o UDP é a confiabilidade e a garantia de entrega dos pacotes da comunicação de dados.

O principal objetivo do protocolo TCP é agrupar os dados da comunicação de acordo com o serviço prestado. É nesse conceito que surge a ideia de portas, onde cada porta é responsável por um serviço no nível de aplicação.

A arquitetura de comunicação é baseada no conceito de cliente-servidor, que é fundamentado no princípio de que uma máquina (cliente) requisita dados de outra máquina (servidor), que responde a requisição com um pacote de dados.

Como citado anteriormente, o modelo TCP é mais robusto que os protocolos da camada de transporte. Dentro desse conceito, o TCP apresenta seu diferencial: o controle de fluxo, múltiplas seções, garantia de entrega de pacotes e a orientação à conexão. A Figura 3 apresenta o formato do *frame* TCP.

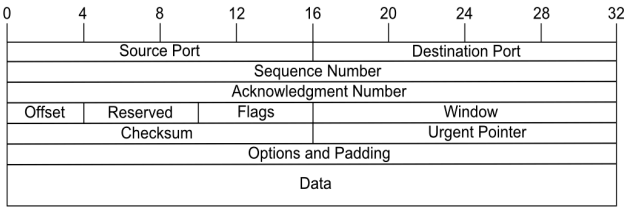

Figura 3 - Frame TCP.

Sendo um protocolo orientado a conexão, o TCP tem um mecanismo próprio para o início da comunicação, que é conhecido como *three way hand shake* (PETERSON, 2003), que consiste no envio de sinais de sincronização entre as máquinas cliente e servidor, através de *flags* de controle localizados no cabeçalho do protocolo. Assim como na conexão, também há um mecanismo de desconexão, caracterizando assim as duas vias da orientação à comunicação.

2.5. Protocolo DHCP

O protocolo DHCP, regulamentado pelo RFC 2131 (DROMS, 1997) é um protocolo de aplicação que tem por objetivo prover parâmetros de configuração em redes TCP/IP de modo automático. Entre os parâmetros estão configuração de endereço IP, servidores DNS (*Domain Name System*), endereço IP do roteador a qual a máquina solicitante está

conectada, entre outros.

A arquitetura de comunicação usada pelo DHCP é do tipo cliente-servidor, onde o cliente requisita parâmetros ao servidor e o mesmo responde a esta requisição.

Segundo Comer (COMER, 2005), o DHCP pode assumir três configurações básicas de comunicação. A primeira é a configuração manual, em que o servidor atribui parâmetros ao cliente de acordo com uma tabela que relaciona os endereços MAC com a configuração desejada pelo cliente, se o mesmo se encontrar na tabela. A segunda é chamada de configuração automática, onde o servidor atribui endereços IP ao cliente dentro de uma faixa de endereços existentes. A terceira é conhecida como configuração dinâmica e é semelhante a automática, porém o cliente mantém aquela configuração durante um tempo determinado e vencendo esse período o endereço IP pode ser repassado a outra máquina. A Figura 4 apresenta o quadro DHCP.

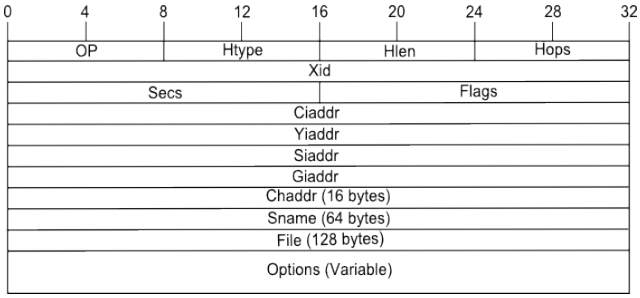

Figura 4 - Frame DHCP.

3. METODOLOGIA DE PESQUISA

Para o desenvolvimento desse trabalho foi adotada uma metodologia de pesquisa baseada no modelo experimental ilustrado pela Figura 5.

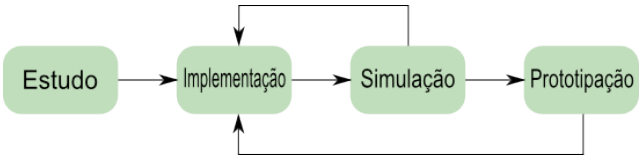

Figura 5 - Fluxo de projeto.

O modelo consiste em quatro etapas distintas durante a fase de desenvolvimento, as quais incluem: estudo, implementação, simulação e prototipação.

3.1. Estudo

O estudo nesse trabalho basicamente esteve ligado ao aprofundamento no conhecimento dos protocolos de rede envolvidos nos objetivos, e já listados nos capítulos anteriores, a definição da arquitetura adotada, bem como a familiarização com a estrutura de prototipação que será utilizada para validação do *hardware*.

Como a maioria das arquiteturas conhecidas que implementam as funcionalidades da pilha TCP/IP são desenvolvidas em *software*, houve também nesse trabalho um estudo comparativo entre a arquitetura adotada neste e outras que existem em *software*, como por exemplo a proposta de Rhoads (RHOADS, 2009), com seu *soft-core* Plasma (processador com arquitetura MIPS). Sendo que ao final desta fase, a estrutura escolhida foi baseada em *hardware* reconfigurável utilizando FPGAs e os motivos ficarão evidentes no capítulo que em é tratado o desenvolvimento do projeto.

Uma abordagem levada em consideração durante essa fase foi a definição dos objetivos do trabalho e como este seria implementado. A Figura 6 facilita esse entendimento e ilustra a arquitetura como um todo, em um digrama conhecido como visão *top-down* do projeto.

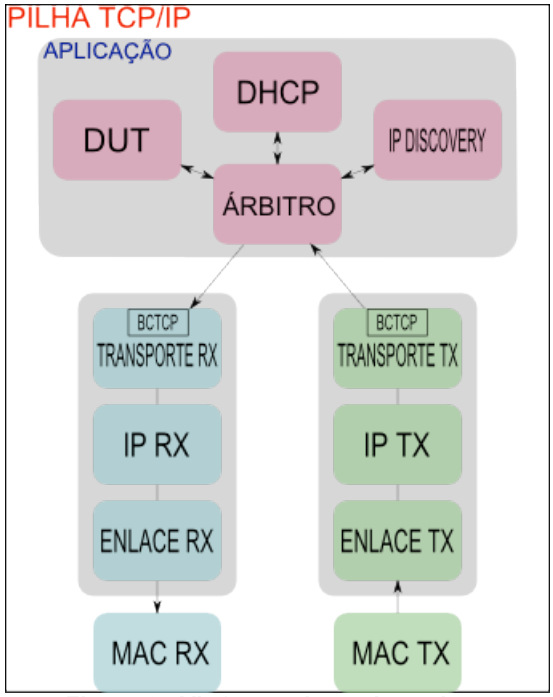

Figura 6 - Visão top-down do projeto.

Na camada de aplicação da pilha TCP/IP estão localizados os seguintes blocos: DUT (*design under test*), DHCP, *IP discovery* e o árbitro.

O DUT é usado para emular o circuito que o usuário vai utilizar no topo da pilha. Neste projeto o DUT tem por finalidade a validação dos blocos que compõem a arquitetura TCP/IP. O DHCP implementa o cliente DHCP, que executa no FPGA. O *IP Discovery* é o circuito que implementa a descoberta automática de endereço IP, que é requisitado por parte do *host*.

O árbitro está localizado entre a camada de aplicação e de transporte, e tem por objetivo controlar a aplicação que irá comunicar com o *host*, onde o DHCP tem maior prioridade, seguido do DUT e o por último o *IP Discovery*.

O corpo da pilha TCP/IP é composto pela camada de transporte RX/TX, que implementa o protocolo UDP, sendo que o protocolo BCTCP (Baixo Consumo TCP), desenvolvido neste trabalho, está encapsulado no UDP. Abaixo dele está camada IP RX/TX, que descreve o circuito da camada de internet. A base da pilha é o enlace RX/TX, que implementa a camada de enlace na pilha TCP/IP.

O MAC RX/TX é um circuito específico do FPGA (*hard-core*), que pode ser configurado através do

software CoreGen, contido no pacote ISE da fabricante *Xilinx*.

3.2. Implementação

A etapa de implementação, por se tratar de um trabalho tendo por arquitetura alvo dispositivos FPGAs, é basicamente composto pela codificação em nível RTL (*Register Transfer Level*), neste caso VHDL.

A codificação tem por objetivo realizar a descrição de circuitos lógicos que compõe os blocos que serão utilizados para prover a comunicação baseada no modelo TCP/IP. É nessa fase que é escolhida a estrutura de memória que será utilizada e a definição das máquinas de estado que irão compor os módulos funcionais do projeto.

Além da codificação em VHDL, foi necessário nesta etapa utilizar uma linguagem de alto nível, desenvolvida em C#, para prover uma interface gráfica ao usuário e, assim estabelecer a conexão com o FPGA através de um *host*.

3.3. Simulação

A simulação, em um trabalho baseado em lógica programável, é fundamental para validações parciais dos circuitos, principalmente no que se refere à correção de erros sintáticos e lógicos presentes na descrição VHDL, poupando tempo durante a fase de prototipação.

Além da correção de erros, uma simulação realizada através de um *testbench*, é útil para validação dos módulos de *hardware* do projeto, emulando-os através de um ambiente muito próximo ao que será encontrado no mundo real.

Todos os módulos deste trabalho foram simulados para posterior prototipação e o ambiente escolhido para simulação foi o *software ModelSim* da fabricante *Mentor Graphics*.

3.4. Prototipação

A fase de prototipação, certamente é a mais longa durante o projeto de circuitos digitais baseados em FPGAs. Nesta fase são encontradas as não idealidades presentes na simulação. Problemas de temporização (*timing*) são os mais comuns nesta etapa do projeto.

A placa alvo da prototipação é fabricada pela *Hitech Global* e contém um FPGA *Virtex5lxt330*, da fabricante *Xilinx* (XILINX, 2010), que conta com 51.840 slices e 324 blocos de memória de 36 kbits. A placa ainda possui dois PHY Ethernet, memória DDR2 e barramentos SATA e PCI-*e*.

Para este trabalho os problemas mais frequentemente encontrados durante a prototipação foram durante a comunicação com o servidor de DHCP devido à falta de informações sobre a configuração da máquina, porém com auxílio de um *software* de monitoramento de rede chamado *Wireshark*, e o analisador de sinais internos do FPGA, *ChipScope Pro* (XILINX, 2009), foi possível verificar os erros e então estabelecer a comunicação DHCP.

4. DESENVOLVIMENTO DO PROJETO

As seções a seguir, descrevem os módulos em hardware que foram desenvolvidos no decorrer deste trabalho.

4.1. Desenvolvimento em hardware do Cliente DHCP

Por se tratar de um protocolo de nível de aplicação na pilha TCP/IP (SOCOLOFSKY *et al*, 1991), a forma de implementação mais comum para um cliente DHCP é em *software*.

Rhoads (RHOADS, 2009) propõe uma estrutura de um cliente DHCP, para uma pilha TCP/IP embarcada em FPGA, descrito em *software*, por meio de um processador *soft-core* embarcado em FPGA chamado Plasma. Tal método foi analisado durante o período de estudo deste trabalho, porém foi descartada devido à complexidade que seria agregada ao projeto, visto que seria necessário um *kernel* de controle e de adaptações nos drivers de acesso à rede Ethernet, além do fato de ser custoso no nível de recursos de lógica (memória, *flip-flops*) do FPGA.

Visando redução de complexidade e recursos de lógica, optou-se por desenvolver em hardware uma estrutura simplificada que comporte grande parte das funcionalidades de um cliente DHCP. A arquitetura adotada nesse projeto é baseada em de máquina de estados, de modo um reduzir o consumo de memória e área ocupada do FPGA. Por se tratar de uma estrutura simplificada, este cliente DHCP não comporta todos os requisitos descritos por Droms (DROMS, 1997). A renovação automática de endereço IP e a atualização dos parâmetros de conexão não são implementados neste trabalho.

4.1.1. DESCRIÇÃO DA ARQUITETURA

Para implementar este cliente DHCP foram necessárias três máquinas de estados. Uma delas controla a operação DHCP, ou seja, tem a finalidade de prover as etapas básicas de comunicação descritas pelo RFC 2131 (DROMS, 1997), outra FSM é responsável pela montagem do frame de transmissão, e a terceira realiza a retirada dos dados do pacote DHCP recebido. A Figura 7 apresenta as etapas de comunicação entre o FPGA e o servidor DHCP.

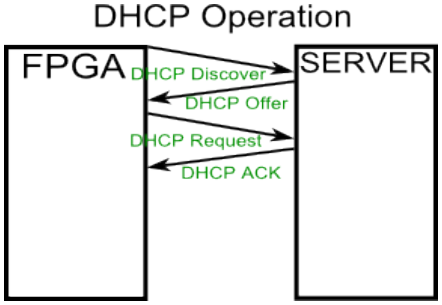

Figura 7 - Fluxo de dados da comunicação.

A seguir descreve-se sucintamente as máquinas de estado que foram desenvolvidas para implementar o cliente DHCP.

- ***Máquina de Controle***

A máquina de controle de operações DHCP, ilustrada na Figura 8, é composta de quatro estados. A característica desta máquina de estados é que ela controla tanto as operações de transmissão de mensagens DHCP quanto à recepção de pacotes DCHP. Os estados dessa máquina são descritos abaixo:

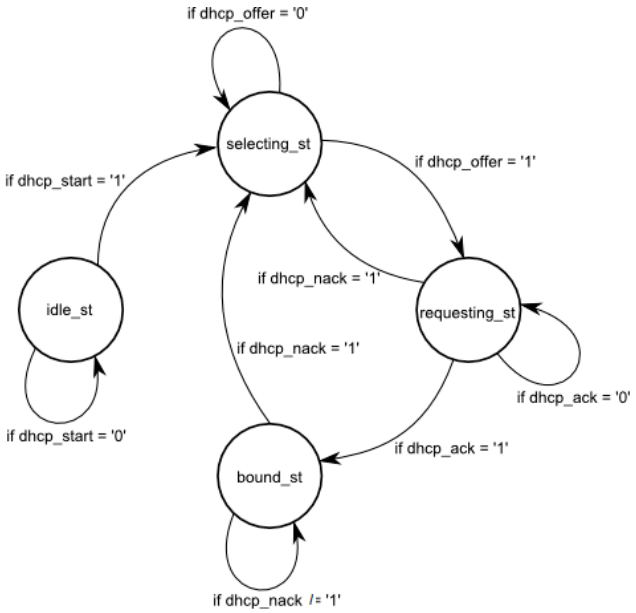

Figura 8 - Diagrama de transição de estados da FSM de controle.

- *idle_st*: estado inicial da máquina de estados de controle do cliente DCHP. Aguarda uma requisição de endereço por parte do sistema embarcado e dispara os sinais para descoberta de servidores DHCP, enviando um pacote DHCPDISCOVER, passando então ao próximo estado, *selecting_st*. Não havendo solicitação permanece em *idle_st*.
- *selecting_st*: estado de seleção da máquina de estados de controle do cliente DHCP. Aguarda os sinais de recepção do protocolo. Caso haja um servidor disponível (DCHPOFFER), aciona os sinais de requisição de endereço IP, envia uma mensagem DHCPREQUEST, passando para o *requsting_st*. Em caso de *timeout*, retransmite um pacote de descoberta de servidor (DHCPDISCOVER), permanecendo em *selecting_st*.
- *requesting_st*: estado de requisição de endereços da máquina de estados de controle do cliente DHCP. Aguarda os sinais de recepção do protocolo. Em caso de aceitação de solicitação de endereço (DCHPACK), o sinal dhcp_ack é ativado, o indicador de existência de IP válido é ativado, e a máquina passa ao estado *bound_st*. Se o pacote recebido for DCHPOFFER (pacote DHCPREQUEST se perdeu) ou houver *timeout*, envia-se novamente o pacote DHCPREQUEST, permanecendo em *requesting_st*. Em caso de não aceitação da solicitação (DCHCPNACK), a *flag* dhcp_nack é ativada e a máquina retorna ao estado *selecting_st*, para encontrar outro servidor DHCP.
- *bound_st*: estado de "agradecimento" na máquina de estados de controle do cliente DHCP. Permanece nesse estado enquanto houver um IP válido para a seção. Caso seja um pacote DCHCPNACK, o sinal *dhcp_nack* é ativado e a máquina retorna ao estado *selecting_st*, para iniciar uma nova procura por servidor DHCP.

- ***Máquina de transmissão DHCP***

A máquina que controla a operação de transmissão de mensagens DHCP, ilustrada na Figura 9, é composta por quatro estados. O objetivo dessa máquina de estados é formar o frame DHCP, inserindo o cabeçalho e as opções do protocolo DHCP. Na sequência descreve-se os estados dessa FSM.

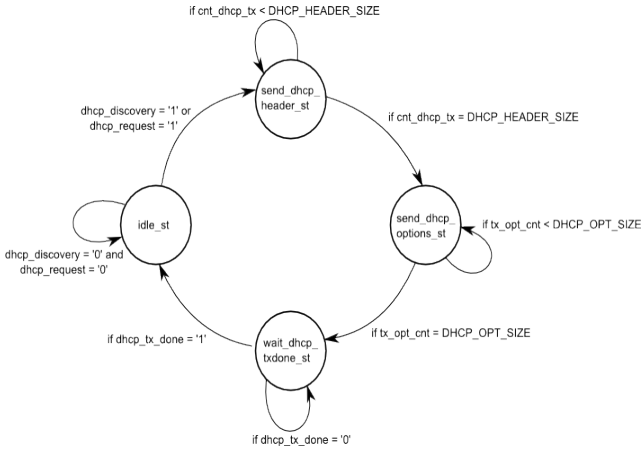

Figura 9 - Diagrama de transição de estados da FSM de transmissão.

- *idle_st*: estado inicial da máquina de estados de transmissão do cliente DHCP. Verifica se há requisição de envio de *frame* DHCP, através das *flags* dhcp_discovery e dhcp_request. Se houver, passa ao estado *send_dhcp_header_st*. Caso contrário, permanece em *idle_st*.
- *send_dhcp_header_st*: estado de envio do cabeçalho da máquina de estados do cliente DHCP. Monta o *header* do *frame* DHCP, passando ao estado *send_dhcp_options_st* após o termino da montagem do cabeçalho. Durante o preenchimento do cabeçalho do frame, a máquina permanece no estado *send_dhcp_header_st*.
- *send_dhcp_options_st*: estado de envio das opções da máquina de estados do cliente DHCP. Insere as opções no frame DHCP, passando ao estado *wait_txdone_st* ao final da inserção. Durante o preenchimento das opções, a máquina permanece no *send_dhcp_options_st*.
- *wait_dhcp_txdone_st*: estado de espera de conclusão de envio da máquina de estados do

cliente DHCP. Aciona o sinal de indicação de fim de quadro e informa a camada de transporte o tamanho do *frame* que será enviado. Permanece no *wait_dhcp_txdone_st* até o recebimento do sinal que informa que a camada de transporte recebeu o frame DHCP, passando então ao estado *idle_st*.

- ***Máquina de recepção DHCP***

A máquina que controla a recepção de quadros DHCP, ilustrada na Figura 10, é composta por três estados. Essa FSM tem por finalidade ler o *buffer* de recepção preenchido pela camada de transporte e analisar a opções do protocolo DHCP, conferir o cabeçalho do *frame* recebido, descartando-o em caso de inconformidade nos dados do header. Na sequência são descritos os estados dessa FSM.

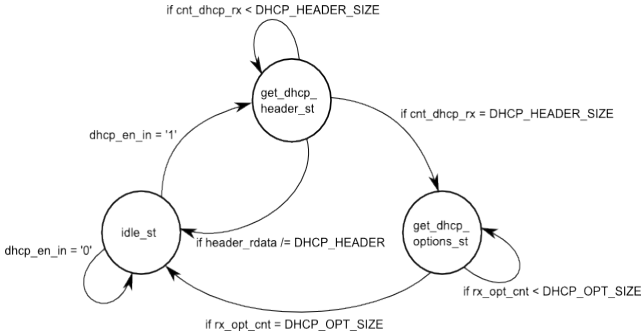

Figura 10 - Diagrama de transição de estados da FSM de recepção.

- *idle_st*: estado inicial da máquina de recepção do cliente DHCP. Verifica se há *frame* disponível para leitura no *buffer* de recepção. Se houver, aciona do sinal de habilitação de leitura do buffer e passa ao estado *get_dhcp_header_st*. Do contrário, permanece em *idle_st*.
- *get_dhcp_header_st*: estado de extração do cabeçalho da máquina do cliente DHCP. Realiza a leitura do *buffer*, conferindo os dados do header DHCP. Após a leitura ser concluída com sucesso a FSM passa para estado *get_dhcp_options_st*. Se houver inconsistência nos dados do *header*, o sinal de habilitação de leitura do *buffer* é colocado em nível baixo retornando ao estado *idle_st*. Durante a leitura do cabeçalho do frame, a máquina permanece no *get_dhcp_header_st*.
- *get_dhcp_options_st*: estado de extração de dados das opções da máquina de estados do cliente DHCP. Realiza a leitura do buffer de recepção, retornando ao estado *idle_st* ao final da leitura ou em caso opções não suportadas pelo protocolo DHCP, colocando o sinal de habilitação de leitura do *buffer* em nível baixo. Durante a leitura do campo opções, a máquina permanece no *get_dhcp_options_st*.

4.2. Desenvolvimento em hardware do *IP Discovery*

O bloco *IP Discovery* foi elaborado com objetivo de prover os parâmetros de rede do FPGA ao *host* que realizará a comunicação com o FPGA. A maior vantagem do *IP Discovery* é que o host não precisa conhecer nenhuma característica do FPGA, desde que possua o programa de acesso à rede, desenvolvido para esta aplicação.

O início da comunicação é caracterizado com o envio, em *broadcast*, de um pacote pelo host de descoberta de IP, com o campo de dados que identificam o pacote coerente com a requisição desejada. O FPGA recebe esse pacote e responde a requisição com seus parâmetros de configuração de rede. Se neste pacote todos os bits da *flag* de habilitação de comunicação estiverem em nível alto (0xFFFF), o host não poderá conectar-se, pois alguma outra máquina já estará comunicando-se com o FPGA, caso contrário (0x0000), o host está apto a iniciar o processo de comunicação. A Figura 11 ilustra esta comunicação.

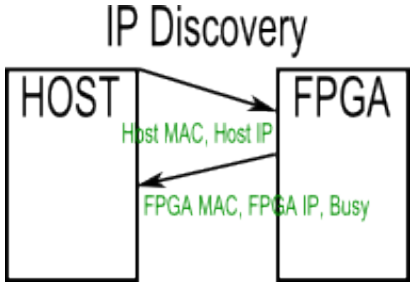

Figura 11 - Comunicação baseado no IP Discovery.

Este bloco de comunicação foi implementado utilizando a porta 12 do protocolo UDP e esta tem a função de selecionar como os dados recebidos da rede serão tratados. A comunicação utiliza 24 bytes do campo de dados do UDP, sendo:

- *Source MAC Address*: 6 bytes - endereço físico (MAC) da máquina de origem. Tem a finalidade de simplificar o processo, eliminando a etapa ARP (descoberta de MAC).
- *Source IP Address*: 4 bytes - endereço lógico (IP) da máquina de origem. Tem a finalidade de eliminar o armazenamento deste dado no FPGA.
- *Project Identifier*: 4 bytes - identificador do projeto. *String* codificada em ASCII para decodificação do pacote recebido.
- *FPGA Identifier*: 4 bytes - identificador de FGPA. *String* codificada em ASCII para decodificação do pacote recebido.
- *User Identifier*: 4 bytes - identificador de usuário. *String* codificada em ASCII para decodificação do pacote recebido.
- *Busy Flag*: 2 bytes - identificador de FPGA ocupado. O objetivo deste campo é informar o *host* que o FPGA está apto ou não a estabelecer conexão.

4.3. Desenvolvimento em hardware do protocolo BCTCP

O protocolo BCTCP, executado na camada de transporte, foi desenvolvido com intuito ser robusto a erros, porém sem incorrer no custo do protocolo de transporte TCP.

O BCTCP executa dentro do protocolo de transporte UDP e foi desenvolvido a fim de minimizar o custo da comunicação TCP convencional. Outra questão levada em consideração nessa implementação, foi o baixo consumo do BCTCP em relação ao protocolo TCP, o qual teria de ser implementado com uma FSM composta com um número elevado de estados e memórias (HAMERSKY, 2008) para atender o RFC 793 (POSTEL, 1981).

O controle de fluxo e erros do protocolo BCTCP é baseado no cálculo do *checksum* do pacote UDP e no número de sequência de pacote. Para essa implementação foram incluídos oito bytes no início do campo de dados do protocolo UDP. A Figura 12 ilustra o formato do *frame* BCTCP.

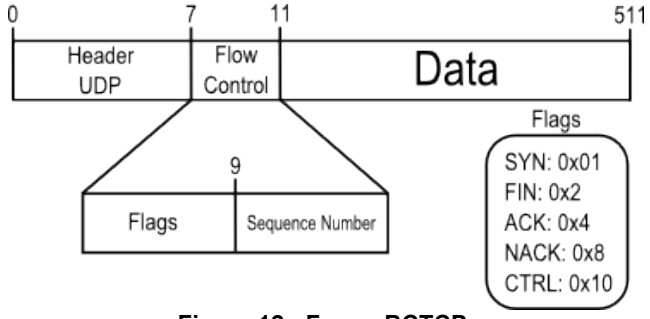

Figura 12 - Frame BCTCP.

4.3.1. AS ETAPAS DA COMUNICAÇÃO BCTCP

A Figura 13 apresenta o fluxo de dados do protocolo desenvolvido.

- *Synchonizing* (sincronismo): essa etapa tem por finalidade estabelecer a comunicação entre o *host* e o FPGA. Nela são estabelecidos os números de sequência inicias da comunicação, para que se torne possível encontrar uma falha no caso de perda de pacotes. O *host* envia um pacote de requisição de conexão (SYN) e o FPGA responde a requisição (SYN + ACK), caso não esteja conectado a outra máquina.
- *Host to FPGA* (fluxo de dados do host para o FPGA): este passo descreve o protocolo dos dados (sentido *host*-FPGA) e requisições de retransmissão (FPGA-*host*). De modo a minimizar o custo na comunicação, o host envia uma sequência de pacotes (burst) (ACK + data), enquanto o FPGA os processa, somando uma unidade ao sinal que armazena o próximo número de sequência esperado. Em caso de erro de checksum ou perda de pacotes o FPGA envia uma mensagem de controle (CTRL + NACK), com o número de sequência esperado e com o campo de dados nulo, caracterizando assim a requisição de reenvio de pacote.
- *FPGA to host* (fluxo de dados do FPGA para o *host*): esta etapa tem por objetivo estabelecer um mecanismo de controle de fluxo no sentido FPGA-*host*. Como o envio de dados pelo FPGA não segue um padrão determinístico, a cada pacote enviado (ACK + data), o FPGA espera um pacote de controle (CTRL + ACK) para garantir que o pacote foi recebido pelo *host*. Se houver perda de pacotes ou erro de *checksum* o host envia um pacote informando que ocorreu um erro (CTRL + NACK) e o FPGA retransmite o pacote.
- *Finalizing* (finalização): o objetivo deste passo é prover um mecanismo de desconexão conhecido tanto pelo FPGA, quanto pelo *host*. O *host* envia um pacote de finalização de seção (FIN) e o FPGA responde a esta requisição (FIN + ACK), estando novamente livre para que outra máquina possa conectar-se.

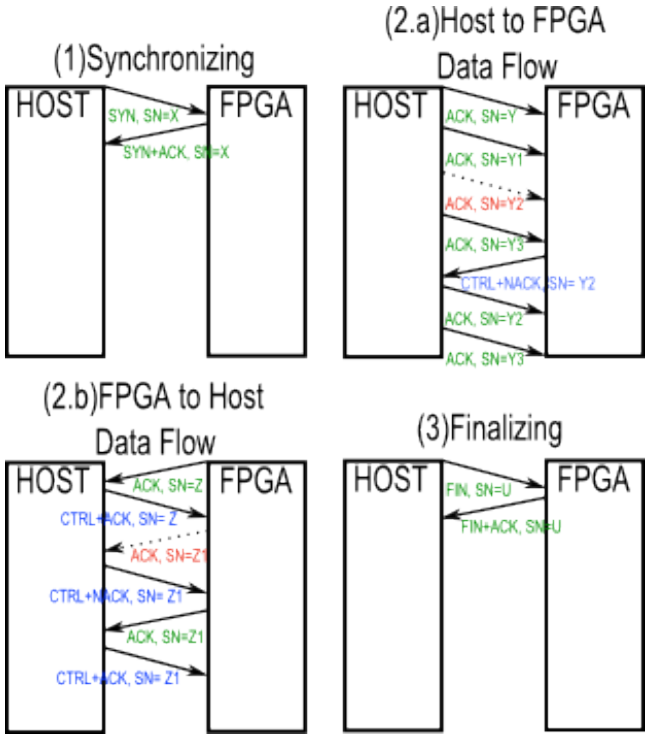

Figura 13 - Fluxo de dados da comunicação BCTCP.

4.3.2. DESCRIÇÃO DA ARQUITETURA

O desenvolvimento desse protocolo utilizou uma arquitetura UDP já existente (REINBRECHT *et al*, 2009), onde foi incluído o mecanismo de controle de fluxo e erros na máquina de estados de recepção. A Figura 14 apresenta um diagrama da implementação desta etapa.

O controle de fluxo, apresentado na Figura 14, utiliza-se de flags e do sequenciamento de pacotes contidos no *"header"* do protocolo BCTCP, desta forma, dependendo do valor recebido nesses campos, o mecanismo aciona o bloco de transmissão para que este repasse a informação ao *host*.

A análise inicial é realizada pela leitura das *flags*. Estas indicam a operação desejada, as quais podem ser: sincronismo, finalização de seção, envio de dados e solicitação de retransmissão. Após esta análise é comparado o número de sequência recebido com o esperado pelo FPGA e verificado o *checksum* do pacote e em caso de erro são gerados sinais de controle (*ret_en/tx_ctrl_en*) repassados ao *TX Engine*. No bloco de transmissão esses sinais controlam um *mux*, que seleciona a origem dos dados, que são provenientes das aplicações, da FIFO (*first-in, first-out*) de retransmissão e dos pacotes de controle de seção.

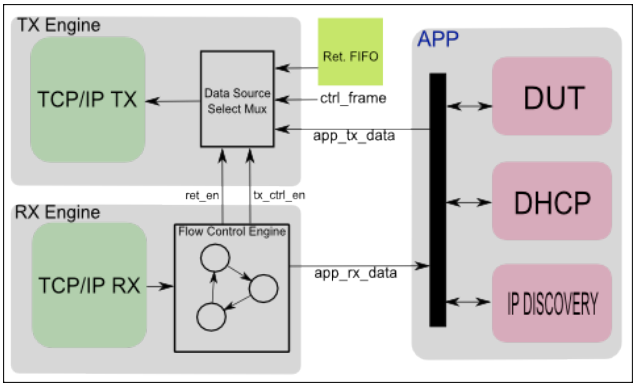

Figura 14 - Arquitetura BCTCP.

A FIFO de retransmissão é carregada quando a aplicação decide enviar um pacote para o *host*, sendo que seus ponteiros de escrita e leitura são reinicializados quando a FIFO não está em operação, de modo à sempre haver o pacote válido para a enésima solicitação de retransmissão. Já os pacotes de controle são os pacotes de sincronismo, solicitação de retransmissão e finalização de seção, que não possuem dados de processamento válidos, mas são de fundamental importância para o correto funcionamento do protocolo BCTCP.

5. RESULTADOS E DISCUSSÕES

Para validação da estrutura de rede proposta, inicialmente foram realizadas simulações do projeto, visando garantir a funcionalidade da solução apresentada. Os resultados da simulação são apresentados na Figura 15.

A Figura 15 apresenta os seguintes resultados: em (1) o FPGA envia um pacote DHCP *Discover* em *broadcast*; (2) o FPGA recebe um DHCP *Offer* do servidor; (3) o FPGA envia um DHCP *Request* em *broadcast*; (4) há resposta do servidor com um DHCP ACK; (5) a *flag* de indicação de IP válido é ativada no FPGA. As etapas (6) e (7) apresentam a comunicação IP *Discovery*, onde primeiramente o *host* envia um pacote de descoberta de IP e em seguida o FPGA responde essa requisição. As etapas (8) e (9) correspondem ao mecanismo de sincronismo do protocolo BCTCP. Em (10) a indicação de *host* conectado é acionada. Por fim em (11) e (12) ocorre o mecanismo de desconexão do protocolo BCTCP.

Após a simulação, a etapa seguinte foi a prototipação em FPGA do projeto, para comprovar que a arquitetura proposta estava funcionando corretamente. Para tal foi utilizado o *kit* de prototipação citado no capítulo 3.4. Os resultados foram verificados utilizando o *softwares Wireshark* e *ChipScope Pro*, e os recursos de FPGA utilizados neste trabalho são apresentados na Tabela 1, retirados dos relatórios de implementação do *software PlanAhead*. Na mesma tabela é apresentado o consumo de área do projeto original (REINBRECHT, 2009).

Tabela 1 – Comparação de consumo de área deste projeto em relação ao projeto original proposto por Reinbrecht (REINBRECHT *et al*, 2009)

Utilização Lógica		
	Trabalho Atual	**Trabalho de Referência**
Slice Registers	1.927	1.412
Utilizados como *Flip-Flops*	1.899	1.385
Utilizados como *Latches*	27	27
Slice LUTs	5.315	4.403
Utilizados como lógica	4.274	3.524
Utilizados como memória	997	862
Distribuição Lógica		
Ocupação de *Slices*	1.803	1.513
Utilização de *IO*		
Bounded IOBs	22	22
Componentes Específicos (*hard-cores*)		
BUFG BUFGCTRLs	8	5
Utilzados como BUFGs	8	5
DCM_ADVs	2	2
TEMACs	1	1

O atual trabalho apresentou um consumo um pouco superior, utilizando e ocupando cerca 500 *slices* a mais, porém incluiu novas funcionalidades ao projeto, como o controle de erros e o cliente DHCP, sendo que além dos módulos citados acima, o árbitro de escrita e leitura na pilha TCP/IP e o *IP Discovery* contribuíram para o aumento no consumo de área pelo atual projeto.

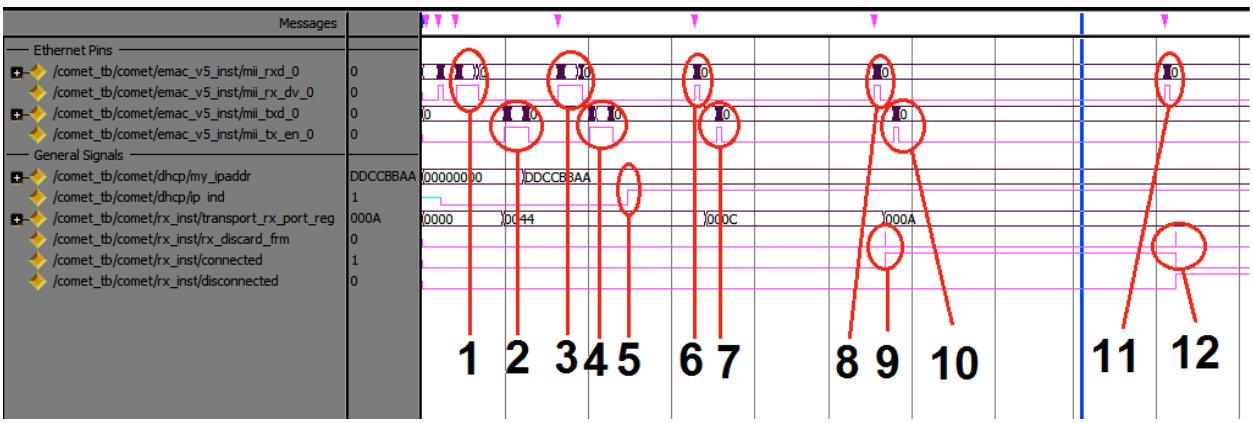

Figura 15 - Simulação Funcional.

A Tabela 2 descreve a estimativa de consumo de potência do trabalho proposto. Para tal foi utilizado o *software Xilinx XPower Analyzer.*

Tabela 2 – Estimativa de consumo de potência do projeto desenvolvido

Síntese do Consumo de Potência Intra-Chip	
	Trabalho Atual (mW)
Clocks	90,37
Lógica	0,89
Sinais	0,70
IOs	1,41
DCMs	159,87
TEMACs	0
Quiescente	3.678,59
Total	3.931,83
Síntese do Consumo de Potência	
Quiescente	3.678,59
Dinâmica	175,47
Total	3.931,83

6. CONCLUSÃO

Este trabalho apresentou um mecanismo de acesso e controle de projetos em FPGA usando uma rede simples, do tipo Ethernet, onde foi desenvolvido um mecanismo de controle de erros e fluxo, um cliente DHCP e um bloco responsável pela transferência de parâmetros do configuração de rede do FPGA ao *host* que deseja conectar-se.

Esta arquitetura foi desenvolvida de uma forma genérica, para que futuramente possa ser integrada a outros projetos que utilizam comunicação Ethernet.

Observando os resultados obtidos, é possível afirmar que a arquitetura proposta neste trabalho mostrou-se viável, tendo em vista seu baixo consumo de área e complexidade em relação a outros trabalhos nesta área.

Os maiores desafios no desenvolvimento deste trabalho estavam ligado a fase de prototipação. Os softwares *Wireshark o ChipScope Pro* foram úteis para depurar o projeto. Outra dificuldade foi o desenvolvimento do software de comunicação no *host*, descrito em C#, devido ao pouco conhecimento desta linguagem de programação.

Outro grande desafio encontrado no decorrer deste projeto esteve ligado a um problema encontrado na sincronização do *phy* Ethernet do *kit* de prototipação utilizado e a rede, o qual levava muito tempo até sincronizar e não havia nenhum procedimento padrão descrito no manual da placa de prototipação e *datasheet* do *chip* do *phy*, porém uma vez sincronizado este não apresentava mais problemas enquanto a placa estivesse conectada a fonte de alimentação.

7. REFERÊNCIAS BIBLIOGRÁFICAS

COMER, D. E. Internetworking with TCP/IP. Vol. 1, 5th Edition, Prentice-Hall, 2005.

DOLLAS, A; ERMIS, I; KOIDIS, I; ZISIS I; KACHRIS C. An Open TCP/IP Core for Reconfigurable Logic**. Proceedings of the 13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'05)**. IEEE Computer Society, 2005. Disponível em: **http://www.computer.org/portal/web/csdl/doi/10.1109/FCCM.2005.20**. Acesso em: 26/03/2011.

DROMS, R. Dynamic Host Configuration Protocol. **RFC 2131**. Bucknell University, Computer Science Department. Lewisburg, Pennsylvania, USA, 1997. Disponível em: **http://tools.ietf.org/pdf/rfc2131.pdf**. Acesso em 26/03/2011.

HAMERSKI, J. C. Desenvolvimento de uma arquitetura parametrizável para processamento da pilha TCP/IP em hardware. **Dissertação de mestrado - Programa de Pós-graduação em Computação - UFRGS**, Porto Alegre, 2008. Disponível em: **http://www.lume.ufrgs.br/handle/10183/15310**. Acesso em 25/03/2011.

PETERSON, L. L.; DAVIE, B. S. Computer Networks: A Systems Approach. 3rd edition, Morgan-Kaufman, 2003.

POSTEL, J. B. User Datagram Protocol. **RFC 768**. University of Southern California, Information Sciences Institute, Marina del Rey, California. USA, 1980. Disponível em: **http://tools.ietf.org/pdf/rfc768.pdf**. Acesso em 26/03/2011.

POSTEL, J. B. Internet Protocol. **RFC 791**. University of Southern California, Information Sciences Institute, Marina del Rey, California, USA, 1981. Disponível em: **http://tools.ietf.org/pdf/rfc791.pdf**. Acesso em 26/03/2011.

POSTEL, J. B. Internet Control Message Protocol. **RFC 792**. University of Southern California, Information Sciences Institute, Marina del Rey, California, USA, 1981. Disponível em: **http://tools.ietf.org/pdf/rfc792.pdf**. Acesso em 26/03/2011.

POSTEL, J. B. Transmission Control Protocol. **RFC 793**. University of Southern California, Information Sciences Institute, Marina del Rey, California, USA, 1981. Disponível em: **http://tools.ietf.org/pdf/rfc793.pdf**. Acesso em 26/03/2011.

REINBRECHT, C. R. W; SCARTEZZINI, G; ROSA, T. R. Desenvolvimento de um ambiente de execução de aplicações embarcadas para a plataforma multiprocessada HeMPs. **Trabalho de Conclusão de Curso – Engenharia de Computação - PUCRS**, Porto Alegre, 2009. Disponível em: **http://revistaseletronicas.pucrs.br/ojs/index.php/graduacao/article/viewFile/6046/4353**. Acesso em 25/03/2011.

RHOADS, S. "PLASMA Processor". Disponível em: **http://www.opencores.org/?do=project&who=mips.** Acesso em: 27/03/2011.

SOCOLOFSKY, T. J. KALE, C. J. A TCP/IP Tutorial. **RFC 1180**. Spider Systems Limited. Edinburgh, UK, 1991. Disponível em: **http://tools.ietf.org/pdf/rfc1180.pdf**. Acesso em 26/03/2011.

XILINX INC. Virtex-5 FPGA User Guide. UG190 (v5.3). Maio, 2010. Disponível em: **http://www.xilinx.com/support/documentation/user_guides/ug190.pdf**. Acesso em: 26/03/2011.

XILINX INC. ChipScope Pro 11.1 Software and Cores User Guide. UG029 (v11.1). April, 2009. Disponível em: **http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/chipscope_pro_sw_cores_11_1_ug029.pdf**. Acesso em: 26/03/2011.

Jeferson Santiago da Silva é estudante de graduação de Engenharia Elétrica na Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS). Atualmente é desenvolvedor de *hardware* e lógica programável na Datacom Telemática. Suas principais áreas de atuação são: arquiteturas reconfiguráveis, dispositivos programáveis e redes de comunicação.

Fernando Gehm Moraes é graduado em Engenharia Elétrica pela Universidade Federal do Rio Grande do Sul (UFRGS) em 1987. Recebeu o título de mestre em Computação pela mesma universidade em 1990. Obteve o título de doutor em Microeletrônica pelo *Laboratoire d´Informatique, Robotique et Microélectronique* de Montpellier (LIRMM), França, 1994. Atualmente é professor titular da Faculdade de Informática da Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS). Suas áreas de pesquisa incluem: microeletrônica, arquiteturas reconfiguráveis, NoCs (*Networks on Chip*) e SoC (*System on Chip*).