Marcelo Rodrigo dos Santos Andriolli

Integração de Componentes Lógicos (FPGA) com Aplicações Computacionais via Rede de Computadores

São José – SC
Março / 2011

Marcelo Rodrigo dos Santos Andriolli

Integração de Componentes Lógicos (FPGA) com Aplicações Computacionais via Rede de Computadores

Monografia apresentada à Coordenação do Curso Superior de Tecnologia em Sistemas de Telecomunicações do Centro Federal de Educação Tecnológica de Santa Catarina para a obtenção do diploma de Tecnólogo em Sistemas de Telecomunicações.

Orientador:
Prof. André Fidalgo - ISEP

Co-orientador:
Prof. Mário de Noronha Neto

CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS DE TELECOMUNICAÇÕES
INSTITUTO FEDERAL DE SANTA CATARINA

São José – SC
Março / 2011

Monografia sob o título “Integração de Componentes Lógicos (FPGA) com Aplicações Computacionais via Rede de Computadores”, defendida por Marcelo Rodrigo dos Santos Andriolli e aprovada em 25 de março de 2011, em São José, Santa Catarina, pela banca examinadora assim constituída:

_____

Prof. André Fidalgo
Orientador– ISEP/ Porto-Portugal

_____

Prof. Mário de Noronha Neto
Co-orientador

_____

Prof. Eraldo Silveira e Silva
IFSC / SC

_____

Prof. Ederson Torresini
IFSC / SC

Lembre-se que as pessoas podem tirar tudo de você, menos o seu conhecimento. É o seu bem mais precioso. Explore; viage; descubra; conheça

Albert Einstein

Agradecimentos

Agradeço ao IFSC pelo apoio dado aos discentes desta Instituição Federal de Ensino e à Pró-Reitoria de Pesquisa e Inovação, ao ISEP pela oportunidade de realização deste projeto, ao Professor André Vaz da Silva Fidalgo pela confiança, ao Professor Gustavo Ribeiro da Costa Alves. Agradeço aos docentes do IFSC, Mário de Noronha Neto, Eraldo Silveira e Silva, Golberi de Salvador Ferreira, Jair Líbero Cadorin e ao discente Rodrigo de Souza por todo apoio e suporte dado. Agradeço ao meu Pai e minha Mãe por todo incentivo e apoio dado.

Resumo

Neste projeto são apresentados a concepção e resultados da elaboração de uma interface web amigável ao usuário que permite o controle e acesso a dispositivos sintetizados na lógica combinacional do FPGA para um Kit Spartan 3E500. A proposta é integrar a interface web a *Plataforma de Comunicação Ethernet para dispositivos embarcados em FPGAs da Xilinx* [Souza, 2010], de modo a simplificar o processo de desenvolvimento e testes de dispositivos sintetizados na lógica combinacional. Como resultado final, é apresentado uma interface web para acesso ao dispositivo alvo, bem como uma interface web dedicada para controle de passo tendo como dispositivo alvo o *Controlador de 8 bits Softcore para FPGA* [Destro, 2011].

Abstract

This project presents the design and results of developing a user friendly web interface that allows control and access to devices in the combinational logic synthesized for an FPGA Kit Spartan 3E500. The proposal is to integrate the web interface to Ethernet Communication Platform for embedded devices in Xilinx FPGAs [Souza, 2010], in order to simplify the development process and tests summarized in combinational logic devices. The end result is a web interface to access the target device, as well as dedicated web interface for pitch control as the target device with the 8-bit Controller Softcore for FPGA [Destro, 2011].

Sumário

Lista de Figuras

Acrônimos

ASIC – Application-Specific Integrated Circuit

ASP – Active Service Pages

CLI – Command Line Interface

CGI – Common Gateway Interface

DTFM – Dual Tone Multi Frequency

FPGA – Field Programmable Array

FIFO – First In Fist Out

GPIO – General Purpose Imput/Output

HTTP – Hyper Text Transfer Protocol

HTML – Hyper Text Markup Language

IHC – Interface Homem-Computador

IP – Internet Protocol

LCD – Liquid Crystal Display

MMU – Memory Management Unit

PLB - Processor Local Bus

RFC – Request for Comments

SSI – Service Side Includes

TCP/IP – Transmission Control Protocol /Internet Protocol

VHDL – VHSIC Hardware Description Lenguage

WIMP – Windows, Icons, Menus, and Pointers

WWW – World Wide Web

XPS – Xilinx Plataform Studio

Lista de Tabelas

Introdução

O projeto aqui exposto nessa monografia é um aprimoramento na *Plataforma de Comunicação Ethernet para dispositivos embarcados em FPGAs da Xilinx* [Souza, 2010]. Tal Plataforma implementada é aplicada em experimentos no Laboratório de Investigação em Sistemas de Teste (LABORIS) no Instituto Superior de Engenharia do Porto (ISEP). Esta plataforma de comunicação propõe como solução a implementação da pilha TCP/IP de forma microprocessada para o acesso, controle e testes de dispositivos sintetizados para FPGA. Esse dispositivo sintetizado para FGPA é o "alvo" da plataforma de comunicação, ou seja, com quem a plataforma irá se comunicar. De agora em diante o dispositivo sintetizado para FPGA será chamado de dispositivo alvo. Há dois requisitos que contemplam a referida plataforma para que a comunicação seja válida com o dispositivo alvo:

- Informações envias ou recebidas pela plataforma devem ter 1 Byte(limitação de hardware).
- A ligação entre o dispositvo alvo e a plataforma de comunicação deve ser conforme descrito na monografia *Plataforma de Comunicação Ethernet para dispositivos embarcados em FPGAs da Xilinx* [Souza,2010].

Como propostas de melhorias para a Plataforma de comunicação, foram definidas:

1. Desenvolvimento de uma interface web.
2. Tornar transparente o processo de concepção do Software embarcado que compõe a plataforma.
3. Prover o acesso ao *display* de LCD do kit utilizado.

A referida plataforma de comunicação possui uma interface *Command Line* que proporciona a interação entre o usuário e o dispositivo alvo, de modo a acessar, controlar e testar o mesmo remotamente através do serviço de rede TELNET. Por meio de comandos é possível enviar dados modo simples, 1 Byte por vez, ou de modo seqüencial, seqüência de

Bytes. Para envio de modo seqüencial é necessário o envio de um arquivo por TELNET para a plataforma de comunicação, contendo a seqüência de Bytes a serem enviados ao dispositivo alvo. Assim adicionando mais comandos ao processo de acesso, controle ou teste. Tal interface implementada na *Plataforma de Comunicação Ethernet para dispositivos embarcados em FPGAs da Xilinx* [Souza, 2010] mostrou-se eficiente como forma de validar o acesso, controle ou teste ao dispositivo alvo. A interface desenvolvida neste projeto permite a "tradução" de comandos em click de mouse, transferências de arquivos para a plataforma de comunicação de forma transparente e o acesso ao dispositivo alvo via navegador web.

Não encontramos nenhuma referência de interface para dispositivos lógicos programáveis em FPGA que permite o controle ou acesso remoto via *web browser* em conformidade com a norma IEEE1149.1.

Além do desenvolvimento da interface web, propôs-se tornar o processo de concepção do sistema operacional usado na plataforma de comunicação transparente ao usuário. Automatizando esse processo, seria possível ajustar facilmente algumas funcionalidades do Sistema Operacional embarcado para suprir algum requisito novo do dispositivo alvo. Outra proposta de melhoria é permitir o acesso ao *display* LCD disponível no kit Spatan3E500 de desenvolvimento da Xilinx. Assim seria possível que o dispositivo alvo acesse o display LCD para alguma funcionalidade adicional, se desejável.

1.1 *Objetivos*

Desenvolver uma interface que cumpra os seguintes requisitos:

- Permitir a execução de operações via navegador *web*:
    - Escrita da FIFO
    - Escrita da FIFO por arquivo
    - Reset da FIFO de escrita
    - Leitura da FIFO
    - Leitura da FIFO por arquivo
    - Reset da FIFO de leitura
- Transparência na transferência de arquivos para a Plataforma de Comunicação.
- Envio de arquivo para a Plataforma de Comunicação através do navegador *web*.
- *Download* de arquivo de Leitura da FIFO.

Melhorias na Plataforma de Comunicação:

- Tornar o processo de concepção do Sistema Operacional Embarcado transparente ao usuário.
- Automatizar o processo de concepção de modo a facilitar ajustes das funcionalidades do Sistema Operacional Embarcado, suprindo novos requisitos do dispositivo alvo.
- Permitir que o dispositivo alvo acesse o display LCD do kit de desenvolvimento Spartan3E500 da Xilinx .

1.2 *Organização da Monografia*

No Capítulo 2 será apresentado a Fundamentação Teórica do projeto, contendo definições das tecnologias utilizadas.

No Capítulo 3, será apresentada a parte de Metodologia de Desenvolvimento, contendo toda a parte de desenvolvimento de algoritmo, programas e diagramas esquemáticos.

Nos últimos Capítulos, 4º e 5º, são respectivamente reunidos os Resultados Obtidos e as principais Conclusões a cerca do projeto realizado.

1.3 *Cronograma de Atividades Realizadas*

A realização deste projeto dividiu-se basicamente em quatro etapas:

1- Estudo da *Plataforma de Comunicação Ethernet para dispositivos embarcados em FPGAs da Xilinx* [Souza, 2009].

2- Análise das possíveis interfaces Homem-Máquina frente às limitações de hardware e de software.

3- Estudo da RFC 3875 e do processo de concepção do Sistema Operacional Embarcado.

4- Programação e testes de validação da interface web, levantamento de requisitos para concepção do Sistema Embarcado de forma transparente ao usuário e escrita da monografia.

Etapa	**Semanas**																				
	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21
1	√	√	√	√	√	√															
2					√	√	√	√	√												
3							√	√	√	√	√										
4											√	√	√	√	√	√	√	√	√	√	√

Tabela 1: Cronograma de Atividades Realizadas

2 *Fundamentação Teórica*

Neste capítulo será apresentada a base teórica e conceitual para a análise das possíveis interfaces Homem-Máquina que poderão ser usadas no projeto frente às limitações de hardware e de software.

2.1 *Visão Geral de Interface Humano-Computador*

O Desenvolvimento e estudos de Sistemas computacionais e interfaces acessíveis vêm sendo explorados de forma crescente.Poderosas ferramentas computacionais na análise de dados coletados aliados a métodos da psicologia experimental, psicologia educacional e ergonomia auxiliam o entendimento do desempenho humano no uso de computadores e de sistemas de informação [Rocha e Baranauskas, 2003].

A interface de usuários tem mudado muito a vida das pessoas, por exemplo: na educação expandindo os horizontes em ambientes de aprendizagem; na medicina através de diagnósticos mais precisos. Por outro lado se mal concebidas, com design confuso e complexo, podem causar frustração, medo e falha por parte do usuário. A qualidade da interface determina se os usuários aceitam ou recusam um sistema [Rocha e Baranauskas, 2003].

O crescente interesse no estudo e desenvolvimento de interfaces do usuário é bastante claro em sistemas variados, desde processadores de texto à software de manipulação de imagens [Souza,C et al,1999]. Com crescimento e popularização do acesso a internet, novas mídias para comunicação também foram exploradas expandindo-se em aplicações dos mais diversos fins.

Profissionais das mais diversas áreas têm contribuído para todo o desenvolvimento, destacam-se [Rocha e Baranauskas, 2003]:

- Designers de Software: Exploram melhores maneiras de organizar a informação graficamente por meio de linguagens de consulta e facilidades visuais para entrada, busca e saída de informações. Usam técnicas de manipulação direta, telepresença e realidade virtual mudando assim a maneira de interagir e de pensar sobre computadores.
- Projetistas de Hardware: Exploram novo design de dispositivos de apontamento e teclados, além de displays de alta resolução. Em geral, projetam sistemas com respostas rápidas para complexas manipulações trimencionais tal como entradas e saídas por voz e gestos, telas sensíveis ao toque.
- Desenvolvedores na área de tecnologia educacional: Explorando a criação de tutoriais on-line, materiais de treinamento que visam novas abordagens de discussões em grupo e ensino a distância. Designers gráficos estão envolvidos nesse caso com layout visual, já Antropólogos, Sociólogos, Filósofos e Administradores tratam do impacto organizacional, treinamento, grupos de trabalhos e mudanças sociais em geral.

2.2 *Conceito de Interface Humano-Computador*

A interface é parte do sistema computacional do qual o usuário se comunica, ou seja, aquela com a qual o usuário entra em contato para disparar as ações desejadas do sistema e receber os resultados dessas ações. Com isso o usuário interpreta para em seguida, definir suas próximas ações [Preece et al, 1994]. Também é parte de um sistema computacional com a qual uma pessoa entra em contato físico, perceptual e conceitualmente. [Moran, 1981]. Define-se como interação o processo que engloba as ações do usuário sobre a interface de um sistema, e suas interpretações sobre as respostas relevadas por esta interface. Entende-se que interface então é tanto um meio para a interação usuário-sistema quanto uma ferramenta que oferece os instrumentos para este processo comunicativo [Rocha e Baranauskas, 2003]. Assim sendo interface é o meio entre um sistema e o usuário. A Figura 1 demonstra exatamente com se dá a interação entre usuário e sistema.

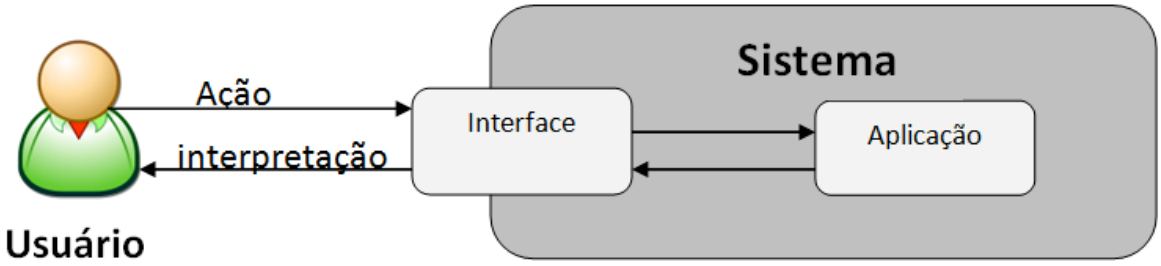

Figura 1: Interface Homem- Computador [Souza,C et al,1999].

2.3 *Estilos de Interação*

Estilo de interação é um termo genérico que inclui todas as formas que os usuários se comunicam ou interagem com sistemas computacionais [Preece et al.,1994; Shneiderman, 1998]. Destacam-se alguns estilos: Linguagem Natural, Linguagens de comando, menus, WIMP, preenchimento de formulário e manipulação direta. Os quais serão explanados a seguir.

2.4 *Linguagem Natural*

Há algumas aplicações que permitem ao usuário expressar-se utilizando a língua que ele se comunica com os outros seres humanos, ou seja, português, italiano, ou outra qualquer. Esse tipo de estilo de interação convém a usuários com pouco ou nenhum conhecimento em computação, porém não é aplicável a todos os tipos de sistemas. Sistemas de consulta a informação e sistemas baseados em conhecimento são exemplos de utilização de interfaces em linguagem natural.

Uma aplicação que oferece interface em linguagem natural precisa lidar com construções vagas, ambíguas e até gramaticalmente incorretas. Não há ainda sistemas que compreendam qualquer expressão em linguagem natural, porém há sistemas especializados que utilizam algum subconjunto de uma linguagem natural na qual o usuário deve se expressar de forma inequívoca usando frases que o sistema possa interpretar [Rocha e Baranauskas, 2003]. Para que haja a interação com aplicações em linguagem natural, podemos fornecer uma interface textual onde o usuário digita frases que expressem seus comandos ou questionamento. Outra alternativa é interfaces orientadas por menus dos quais o usuário pode selecionar cada palavra ou expressão até compor um frase desejada. Em uma aplicação em linguagem natural, tenta-se aproximar a aplicação do usuário de modo a privilegiar sua forma de comunicação. Uma

aplicação bem comum hoje em dia é o sistema de atendimento automático via telefone DTMF, onde uma gravação anuncia o número de cada opção do menu de atendimento, o usuário digita a tecla do número correspondente a opção do menu desejada. Outro exemplo de aplicação deste tipo é a chamada por voz, muito utilizadas em aparelhos de telefonia móvel.

2.5 *Linguagem de Comando*

As interfaces baseadas em linguagens de comandos proporcionam ao usuário a possibilidade de enviar instruções diretamente ao sistema através de comandos específicos [Preece et al., 1994]. Os comandos podem ser compostos por teclas de funções, por um único caracter, por abreviações ou ainda por palavras inteiras ou por uma combinação de teclas e caracteres. As linguagens de comandos podem ser consideradas poderosas pois oferecem acesso direto aos sistemas aumentando a flexibilidade na construção de comandos variando parâmetros e combinações de palavras e sentenças.

Porém toda essa flexibilidade implica em uma maior dificuldade do usuário iniciante em aprender a utilizar o sistema. Os comandos e sintaxe da linguagem precisam ser lembrados, erros de digitação são comuns mesmo para os mais experientes. Ao se projetar uma linguagem de comando, deve-se levar em consideração a organização e estrutura dos comandos, assim como os nomes e abreviações utilizadas. Algumas interfaces baseadas em comandos podem ser auxiliadas por menus que indicam quais são os diversos comandos e como eles devem ser executados. O *bash* do Linux é um exemplo de linguagem auxiliada por comandos. A Figura 2 ilustra uma tela do *bash* do Linux.

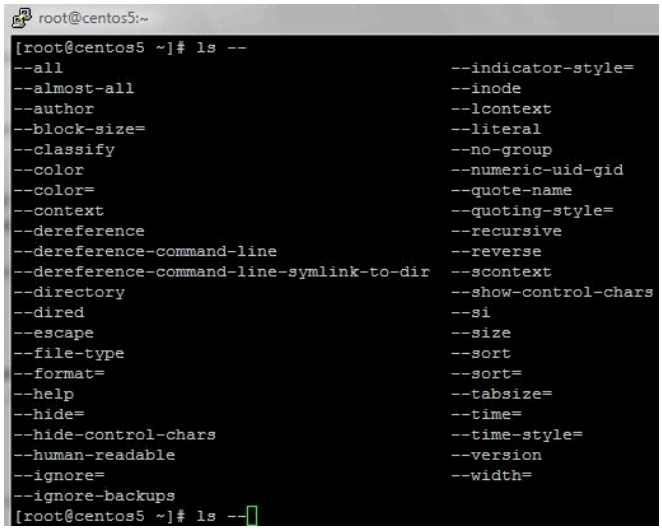

Figura 2: Interface do tipo linha de comando

2.6 *Menus*

Um menu é um conjunto de opções apresentadas na tela, no qual a seleção de uma ou mais opções resulta em uma mudança no estado da interface [Paap & Roske- Hofstrand, 1988]. “Neste estilo de interação o usuário não precisa lembrar dos itens que deseja, basta apenas reconhecer. Porém, para que esse tipo de interação seja eficiente os itens devem ser auto-explicativos” [Souza et al, 1999]. Os menus podem ser de seleção simples ou de múltipla escolha, podendo então ser utilizados para configurar um parâmetro ou disparar uma operação. “Um menu de seleção simples pode ter a forma de um grupo de botões de opção ou seja um *radio button*. Agora um menu de seleção múltipla pode ser representado por um grupo de botões de seleção ou *check Box* ” [Souza et al, 1999].

A desvantagem da interação por menus é que estes ocupam muito espaço na tela. Há várias técnicas para agrupar e apresentar menus, a mais comum é a categorização hierárquica das opções. Um menu hierárquico pode correr na forma de uma seqüência de telas, ou como um menu *pull-down*. “ O menu *pull-down* surge ao se clicar em seu título e desaparece assim que se seleciona uma das opções. Outra alternativa usada para otimizar o espaço de tela são os menus *pop-up*, na qual aparece ao se clicar em uma determinada área da tela ou elemento de interface podendo ser visível até que o usuário selecione um de seus itens ou fecha-lo” [Souza et al, 1999].

2.7 *WIMP(WINDOWS, ICONS, MENUS AND POINTERS)*

O estilo de interação WIMP, vem do inglês e quer dizer: Janelas, Ícones, Menus e Apontadores. Tal estilo de interação permite a interação com usuário através de componentes virtuais chamados widgets. É implementado com o auxilio de tecnologias de interfaces gráficas, proporcionando o uso de janelas e ou controle de entrada através do teclado e do mouse em cada uma destas janelas. O WIMP deve ser considerado como uma junção de tecnologia de hardware e software associado ao conceito de janelas e de widgets, cuja implementação permite vários estilos. WIMP pode ser considerado um estilo ou um framework de interface apoiado pela tecnologia de interfaces gráficas.

2.8 *Manipulação Direta*

"Interfaces de manipulação direta são aquelas que permitem ao usuário agir diretamente sobre os objetos da aplicação (dados ou representações de objetos do domínio) sem a necessidade de comandos de uma linguagem específica. Neste tipo de interface, os comandos são ações baseadas numa analogia entre o cursor e a mão, e as representações gráficas e os objetos do domínio. Na interação por linguagens de comandos o usuário interage indiretamente, através de comandos textuais e nomes que representem os objetos do sistema" [Souza et al, 1999].

2.9 *Formulários*

Interfaces no estilo de preenchimento de formulário são utilizadas principalmente para entrada de dados em sistemas de informação. Este estilo de interação é útil principalmente quando diferentes categorias de informação devem ser fornecidas ao sistema, principalmente quando os mesmos tipos de dados devem ser digitados repetidamente [Preece et al., 1994]. "Esse tipo de interface em geral é de fácil aprendizado. Deve-se deixar claro o tipo de dado que pode entrar em cada campo, facilitar a correção de erros de digitação e verificação dos dados digitados através de técnicas como dígitos verificadores e totalização de valores. Esse tipo de interface se popularizou em aplicações em HTML" [Souza et al, 1999].

3 *Metodologia de Desenvolvimento*

Neste capítulo será apresentada a metodologia de desenvolvimento da interface Web com base nos estudos feitos sobre interfaces Homem-Computador, apresentado no capítulo 2. O desenvolvimento desta interface *web* tem como alvo a *Plataforma de Comunicação Ethernet para dispositivos embarcados em FPGAs da Xilinx* [Souza, 2010].

3.1 *Plataforma de Comunicação*

A *Plataforma de Comunicação Ethernet para dispositivos embarcados em FPGAs da Xilinx* [Souza, 2010], provê um canal de comunicação por meio de Redes de Computadores que permite controlar dispositivos sintetizados em lógica combinacional. A plataforma de comunicação é composta por um sistema embarcado [SASS; SCHMIDT, 2010] que implementa os protocolos para comunicação via Rede de Computadores entre outras funções necessárias. Neste sistema embarcado há componentes de software e hardware. O Hardware é constituído por dispositivos sintetizados em lógica FPGA, Processador, Barramento de Dados, Controlador de Rede, bem como dispositivos externos que estão conectados ao FPGA, com Memória RAM e memória FLASH.

Os dispositivos sintetizados em lógica FPGA são implementados por meio da ferramenta da Xilinx chamada XPS. Essa ferramenta utiliza os *Intellectual Property Cores* (*IPcore*), que são unidades lógicas reutilizáveis. Desenvolvido em HDL e licenciado pelo fabricante, os *IPcores* podem ser usado como blocos lógicos em projetos de *chip* ASIC ou FPGA . Os dispositivos externos fazem parte do Kit Spartan3E500. A Tabela 2 mostra os dispositivos sintetizados em lógica FPGA que fazem parte do sistema embarcado e estão ligados ao barramento de dados do processador (PLB).

Nome do dispositivo	IPcore	Função no Sistema Embarcado
microblaze_0	microblaze 7.20.d	Microprocessador do Sistema
FLASH	xps mch emc 3.01.a	Responsável por escrever e ler na memória Flash
Mdm_0	mdm 1.00.g	Depurador, não utilizado pelo sistema embarcado
Ethernet MAC	xps ethernetlite 3.01.a	Controlador de rede MAC
xps gpio_in	xps gpio 2.00.a	Responsável por receber dados do FIFO para GPIO
xps gpio out	xps gpio 2.00.a.	Responsável por enviar dados ao GPIO para FIFO
xps_intc_0	xps intc 2.00.a	Controlador de interrupção
Xps_timer_0	xps timer 1.01.b	Temporizador do barramento PLB
RS232 DCE	xps uartlite 1.01.a	Terminal de usuário do sistema embarcado

Tabela 2: IPcores do sistema embarcado conectados ao PLB[Souza, 2010]

A Tabela 3 mostra os circuitos impressos que compõe também o sistema embarcado e estão presentes no kit Spartan 3E500.

Nome do CI	Tipo	Função no Sistema Embarcado
XC3S500E	FPGA	Sintetizar grande parte do sistema embarcado
32MB Micron DDR SDRAM	Memória DDR SDRAM	Memória RAM utilizada pelo Microblaze
16MB Numonyx StrataFlash	Memória Flash	Memória ROM onde ficar armazenado a imagem do Petalinux e o Bitstream
SMSC LAN83C185 Ethernet PHY	Ethernet PHY	Interface com o meio analógico da Ethernet

Tabela 3: Circuitos Integrados do sistema embarcado que integram o Kit Spartan 3E500[Souza, 2010]

Para um sistema embarcado executar as funções da plataforma de comunicação, é necessário um software que execute as funções cabíveis da pilha TCP/IP e gerencie a comunicação com o dispositivo requerente. Um sistema operacional que possui nativamente a pilha TCP/IP e os *drivers* necessários para controlar o sistema embarcado, que suporta a

arquitetura microprocessada, é o Petalinux [PETALOGIX, 2009]. Assim sendo, esse foi o sistema operacional usado no sistema embarcado que compõe a plataforma de comunicação. A Figura 3 mostra o sistema embarcado como um todo, os IPcores que compõe o dispositivo sintetizado em lógica FPGA.

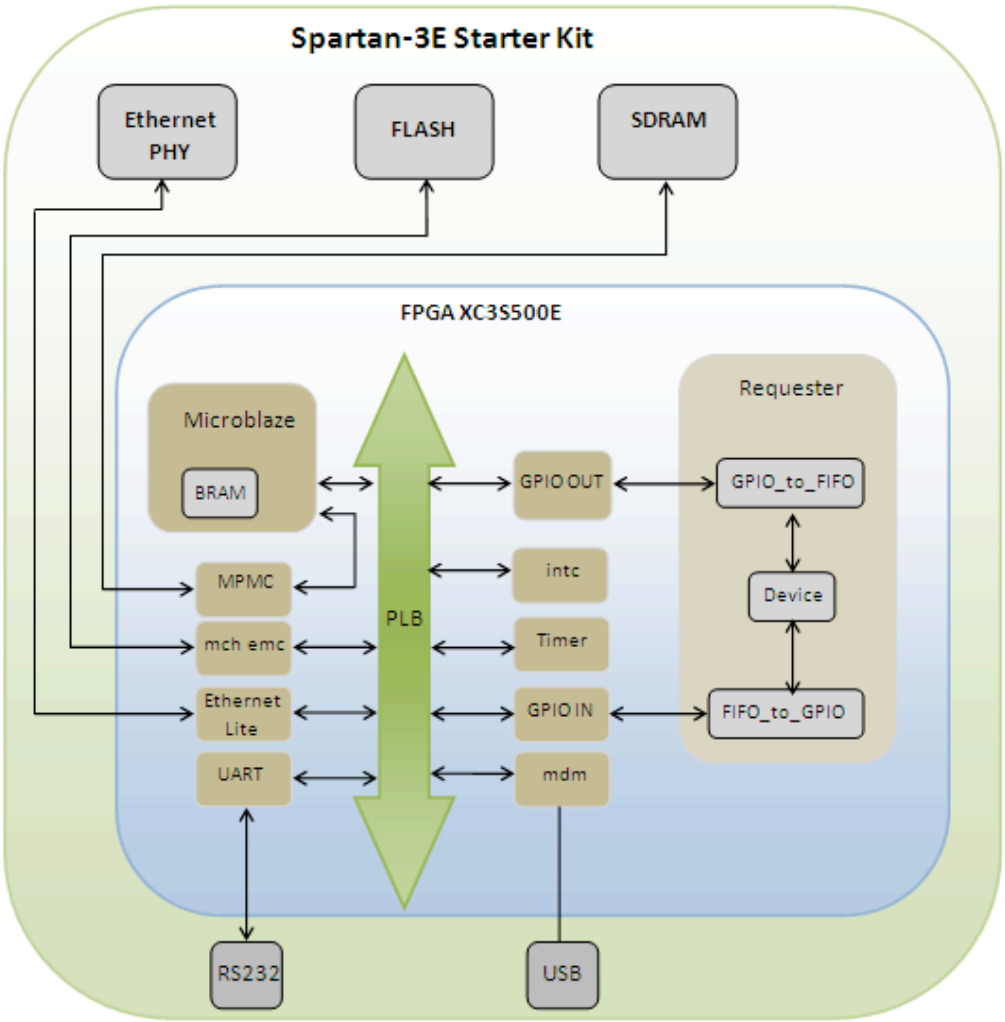

Figura 3: Sistema Embarcado como um todo e dispositivo sintetizado em FPGA[Souza, 2010]

Basicamente a plataforma de comunicação é interligada ao módulo VHDL chamado *Requester* no qual é instanciado e interconectado ao módulo GPIO_to_FIFO, o módulo FIFO_to_GPIO e o *Device,* como mostrado na Figura 3. Assim o módulo VHDL *requester* representa o dispositivo requerente. No módulo VHDL GPIO_to_FIFO está instanciado a FIFO de entrada e implementado o sincronizador, já no módulo FIFO_to_GPIO está

instanciado a FIFO de saída e implementado o sincronizador. O sincronizador implementado no módulo VHDL FIFO_to_GPIO e GPIO_to_FIFO é utilizado para evitar incompatibilidade entre o clock utilizado pelo sistema embarcado e a FIFO, caso o dispositivo que esteja tentando executar uma leitura ou escrita não faça o mesmo período de clock da FIFO. Assim a FIFO com o sincronizador permite a interligação do *Device* de forma assíncrona ao relógio da plataforma de comunicação, dando maior flexibilidade no desenvolvimento do *Device*.

O controle de acesso ao *Requester* é implementado por software executado em linha de comando sob Petalinux, através dele é feito a escrita na GPIO_OUT e leitura na GPIO_IN. O *Requester* também pode ser controlado remotamente por linha de comando através do serviço de rede TELNET, proporcionando assim também a interação entre usuário e dispositivo alvo por interface de linha de comando.

3.2 *Requisitos funcionais da Interface*

A interface desenvolvida neste projeto deve prover as seguintes funcionalidades:

1- A interface deve ser acessada via navegador Web.

2- A interface deve permitir a execução de operações de: escrita da FIFO, escrita da FIFO por arquivo, Reset da FIFO de escrita, leitura da FIFO, leitura da FIFO por arquivo, Reset da FIFO de leitura.

3- A interface deve permitir o envio de arquivos para a Plataforma de comunicação de forma através do navegador web.

4- A interface deve permitir o download de arquivo de Leitura da FIFO.

3.3 *Restrições de Software da Plataforma de Comunicação*

O software usado na Plataforma de Comunicação é uma distribuição de Sistema Operacional Linux para Sistemas Embarcados, o Petalinux [PETALOGIX, 2009]. O Petalinux não só oferece as funções cabíveis para estabelecer a comunicação entre a plataforma de comunicação e o dispositivo alvo, bem como ferramentas e utilitários que dão suporte a vários serviços e aplicativos de rede tais como: Servidor Web, Servidor FTP, TELNET, PING etc..

No Petalinux essas ferramentas fazem parte de um pacote de ferramentas chamado

BusyBox[Bruce Perens,1996] ,uma versão mínima do *GNU Core Utilities* usado no Linux para arquitetura não embarcada. O BusyBox[Bruce Perens,1996] oferece várias ferramentas e utilitários comuns ao *GNU Core Utilities* porém mais leves e com um número menor de opções. No Petalinux há suporte a dois servidores Web, o Boa webserver [Phillips, 2005] e o THTTPD websever [ACME Laboratories,2003]. A Tabela 4 a baixo apresenta as algumas especificações de ambos servidores web:

	Segurança			Hots Virtual	Conteúdo Dinâmico				
Servidor	Autenticação Básica	Autenticação criptografada	HTTP Segura		CGI	Fast CGI	Java Servelet	SSI	ASP.Net
BOA	Não	Não	Não	Sim	Sim	Não	Não	Não	Não
THTTPD	Sim	Não	Não	Sim	Sim	Sim	Sim	Sim	Não

Tabela 4: Comparação entre Servidores Web [ACME Laboratories,2003]

No Petalinux, na versão 0.40, não há suporte a plataforma Java, sendo assim o desenvolvimento de páginas web dinâmica limita-se ao uso de *CGI* ou *Fast CGI* . Devido a maior estabilidade e desempenho em Linux para sistemas embarcados optou-se por utilizar o *Boa webserver* [Phillips, 2005]. Uma das maneiras de implementar *CGI* em servidores web para Linux é através de *shellscript*. Porém há incompatibilidade na implementação de *CGI* em *shellscript* no Petalinux, tal incompatibilidade gera uma mensagem de erro na requisição HTTP do tipo: *500 Internal Server Error - The server encountered an internal error and could not complete your request.*

Devido a essa incompatibilidade na implementação de *CGI* em *shellscript* no Petalinux em ambos Servidores HTTP (BOA e THTTPD), optou-se pelo uso de um projeto de código aberto em liguagem C disponível no Petalinux na versão 0.40, que implementa o CGI. O *cgi_generic* [Bay, 2000] permite então a implementação de páginas dinâmicas com o uso de CGI. A composição do projeto de código aberto pode ser observado na Tabela 5.

cgi.c	Programa principal
cgivars.c	Captura as informações do form HTML em conformidade com a RFC3875
cgivars.h	Header do código ***cgivars.c***
htmllib.c	Implementa algumas tag HTML assim facilitando a programação da página HTML em *C*
htmllib.h	Header do código ***htmllib.c***
Makefile	Makefile dos códigos
template.c	Processa as informações recebidas do form HTML capturadas por **cgivars.c** e gera a resposta para requisição feita pelo browser

Tabela 5: Programas que pertencem ao pacote de dados cgi_generic

O código *template.c* é um modelo inicial que exemplifica como se dá a programação em C para CGI. Baseado nesse modelo inicial será adaptado e desenvolvido a lógica para interface web para acesso a *FIFO.*

3.4 *Escolha da Interface Homem-Computador*

No capítulo 2 foram apresentados conceitos e tipos de IHC. A escolha do uso ou não de algum tipo de IHC no projeto aqui desenvolvido em questão, se restringe a requisitos funcionais apresentados na sessão 3.2 e tecnológicos apresentados na sessão 3.3 . Se tratando de interface Homem-Computador, optou-se pelo uso do modelo de formulários. O uso de formulários em páginas web além de ser amigável ao usuário, apresenta o estilo de interação útil, principalmente quando diferentes categorias de informação devem ser fornecidas ao sistema, principalmente quando os mesmos tipos de dados devem ser digitados repetidamente [Preece et al., 1994]. Interfaces web podem ser concebidas com o uso de *forms* HTML. O uso de *forms* possibilita interação através de diferentes objetos de interação. Sendo eles: *checkboxes*, *radio buttons*, *menus*, *text input*, *file select*, *hidden controls*, *object controls* . Buscou-se harmonia na forma em que estes objetos de interação estão dispostos na página web. Para o acesso a FIFO têm-se seis opções: *writefifo*, *writereset, writefifofile, readfifo, readreset, readfifofile.*

Devido ao número de opções de acesso optou-se pelo uso do *form menulist*, sendo esta opção mais harmônica se comparado com o *radio buttons* ou *checkboxes*. Dependendo da opção escolhida é necessário informar o valor que será escrito na FIFO, assim cabe o uso de um *text input*. A interface também deveria enviar arquivos e por esse motivo usou-se no desenvolvimento da mesma o botão *select file*. Para que sejam enviados os dados usou-se o *botton*. Os objetos de interação foram dispostos na pagina web compondo a interface conforme mostrado na Figura 4.

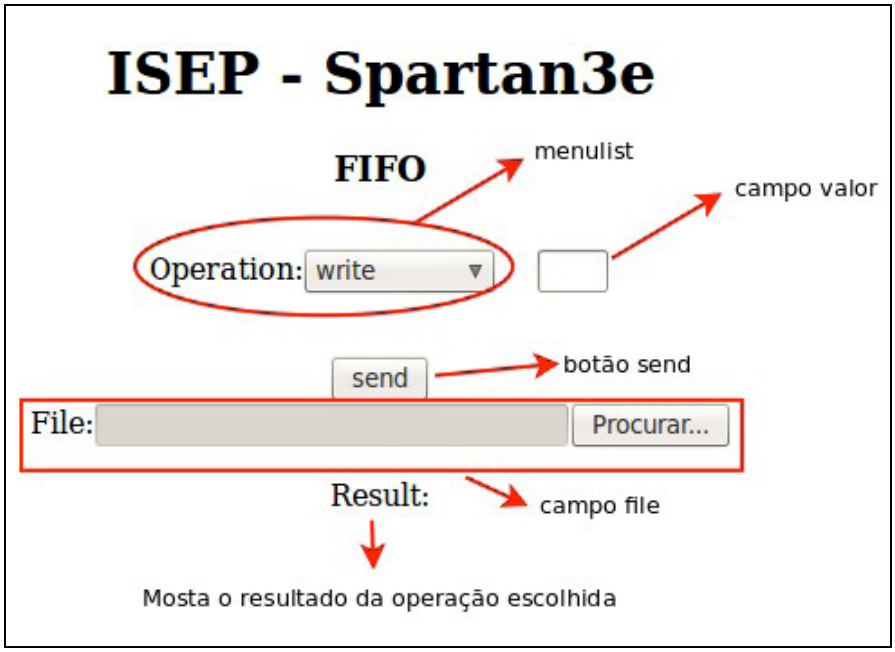

Figura 4: Interface web

No menulist é escolhido o método em que é acessado a FIFO.Há dois métodos de acessar a FIFO, o primeiro é o modo simples , ou seja, o acesso é feito na escolha de quatro operações no menulist da interface web, são elas: *writefifo, writereset, readfifo, readreset*. Neste modo só é permitido a escrita de um dado de 1byte por vez. O segundo método é modo seqüencial por meio de arquivo, ou seja, o acesso é feito na escolha de duas operações no menulist da interface web, são elas: writefifofile e readfifofile. Através do modo seqüencial é possível escrever uma seqüência de dados de 1byte ao invés de um dado por vez como no modo simples.

No campo valor, é digitado o dado a ser escrito na FIFO, ou seja, esse campo só será preenchido na escolha da opção *writefifo*. No campo file é selecionado o arquivo que contém os dados a serem escritos na FIFO, a seleção desse arquivo só é necessária quando forem escolhidas as opções *writefifofile ou readfifofile.*

O botão *send* é usado para enviar as informações. Em *result* é mostrado a reposta da operação escolhida. Os procedimentos para o acesso a FIFO de modo simples e modo seqüencial estão descritos no Apêndice C.O uso da interface Homem-Computador do tipo formulário HTML aliado ao CGI então compõe a interface web dinâmica. A interação entre a interface web dinâmica, plataforma de comunicação e *Requester* estão exemplificadas na Figura 5.

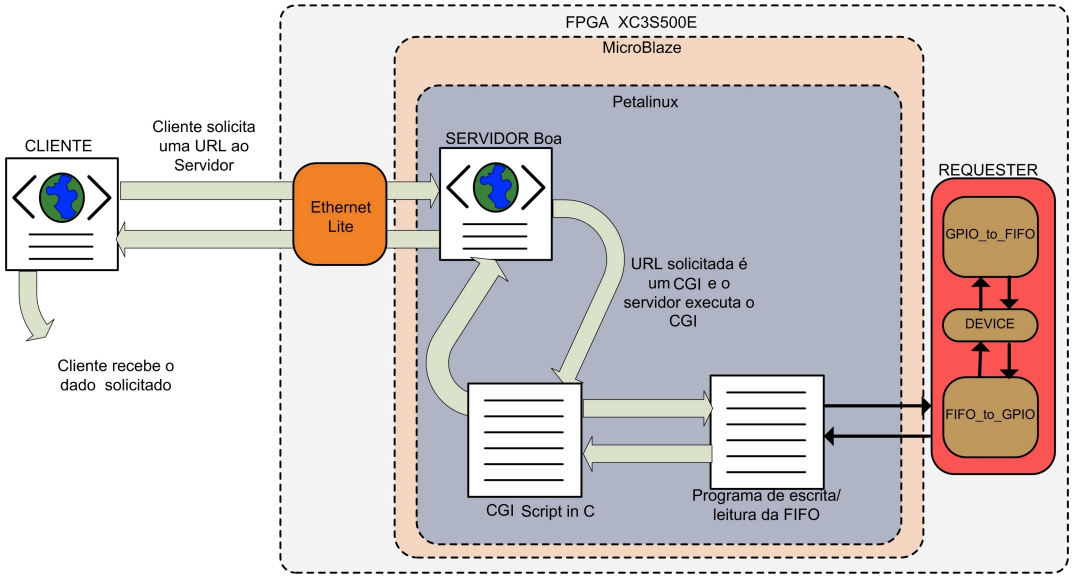

Figura 5 : Interação entre interface web dinâmica e a plataforma de comunicação

A Plataforma de comunicação está conectada a rede de computadores por meio do IPcore Ethernet Lite que compõe o sistema embarcado usado. Via *web browser* então é acessada a interface web que está hospedada no *Boa webserver* [Phillips, 2005] que compõe plataforma de comunicação. Após a escolha das opções de acesso a *FIFO*, é enviado ao servidor os dados. O servidor então recebe a requisição e executa o *CGI script* em C, que por sua vez processa as informações recebidas do *browser*, por exemplo, leitura da *FIFO*. O *CGI script* em C gera a resposta para requisição feita pelo *browser*, por exemplo, o valor lido da *FIFO*. Tal resposta é entregue ao webserver que a envia ao *browse*r assim completando o fluxo pedido/resposta em conformidade com o protocolo HTTP.

3.5 *Programando a Interface web*

Antes de ser iniciada a programação da interface web é necessário realizar o procedimento descrito no Apêndice A para que seja habilitado o *cgi_generic* [Bay, 2000] no Petalinux. Logo após faz-se necessário realizar o procedimento descrito no Apêndice B.

A programação da interface web divide-se em duas partes, a primeira é a programação web em HTML e a segunda parte é a programação para o processamento da informação enviada pelo usuário através da página web, conforme o conteúdo do código *template.c* no cd com os códigos usados no projeto. Na primeira parte da programação da interface é constituída por *forms* HTML. Através desses *forms* HTML o usuário interage com a plataforma de comunicação por meio de *menulist*, botões e campo de inserção de texto ao invés de linha de comandos. A Figura 6 ilustra a primeira parte da programação web da interface web.

```
53 centerText();
54 printf("<H3>FIFO</H3>");
55 /*Form HTML*/
56 printf("<FORM ENCTYPE=\"%s\" ACTION=\"%s\" METHOD=POST>", "multipart/formdata","/
cgibin/
cgi");
57 printf("<TABLE CELLPADDING=10>");
58 printf("<TR><TD>");
59 printf("Operation:");
60 printf("<SELECT NAME=\"operation\">");
61 printf("<OPTION>write");
62 printf("<OPTION>writereset");
63 printf("<OPTION>writefifofile");
64 printf("<OPTION>read");
65 printf("<OPTION>readreset");
66 printf("<OPTION>readfifofile");
67 printf("</TD> <TD>");
68 printf("</SELECT>");
69 printf("<INPUT TYPE=\"text\" name=\"valor\" maxlength=\"2\" size=\"2\">");
70 printf("</TD></TR>");
71 printf("<TR><TD></TD> <TD></TD></TR>");
72 printf("</TABLE>");
73 printf("<INPUT TYPE=submit VALUE=\"send\">");
74 printf("<BR>");
75 printf("File:");
76 printf("<INPUT TYPE=\"file\" name=\"ficheiro\" maxlength=\"50\" size=\"30\">");
77 printf("</FORM>");
78 printf("Result:");
79 centerTextFooter();
```

Figura 6 : Trecho de código para gerar HTML

Nota-se que a programação HTML em C é um pouco diferente da tradicional. Para que as tags HTML possam ser interpretadas de maneira correta pelo browser usa-se a função *printf* que imprime na saída padrão (stdout). Nesse caso a saída padrão tem como alvo não o *bash* ou *prompt* e sim um browser que irá interpretar todas as tags HTML escrita desta forma.

O método de pedido CGI e a forma como os dados do formulário HTML são codificados são definidos conforme o trecho do código mostrado a cima, como:

```
printf("<FORM ENCTYPE=\"%s\" ACTION=\"%s\"
METHOD=POST>", "multipart/formdata","/
cgibin/
cgi");
```

Figura 7 : Definição de Método

O método de pedido CGI definido é *post* e forma de codificação definida é *multipart/form-data*. A codificação do tipo *multipart/form-data* permite o envio de um arquivo, se não fosse necessário o envio de um arquivo esse parâmetro não seria definido. A segunda parte do código é referente a filtragem das informações recebidas do programa cgi.c através da variável postvars[]. Nessa variável há informações relacionadas ao *menulist* operation com a opção selecionada pelo usuário, o conteúdo do campo valor e do campo ficheiro. O fluxograma da Figura 8 descreve exatamente o funcionamento da segunda parte do código *template.c*.

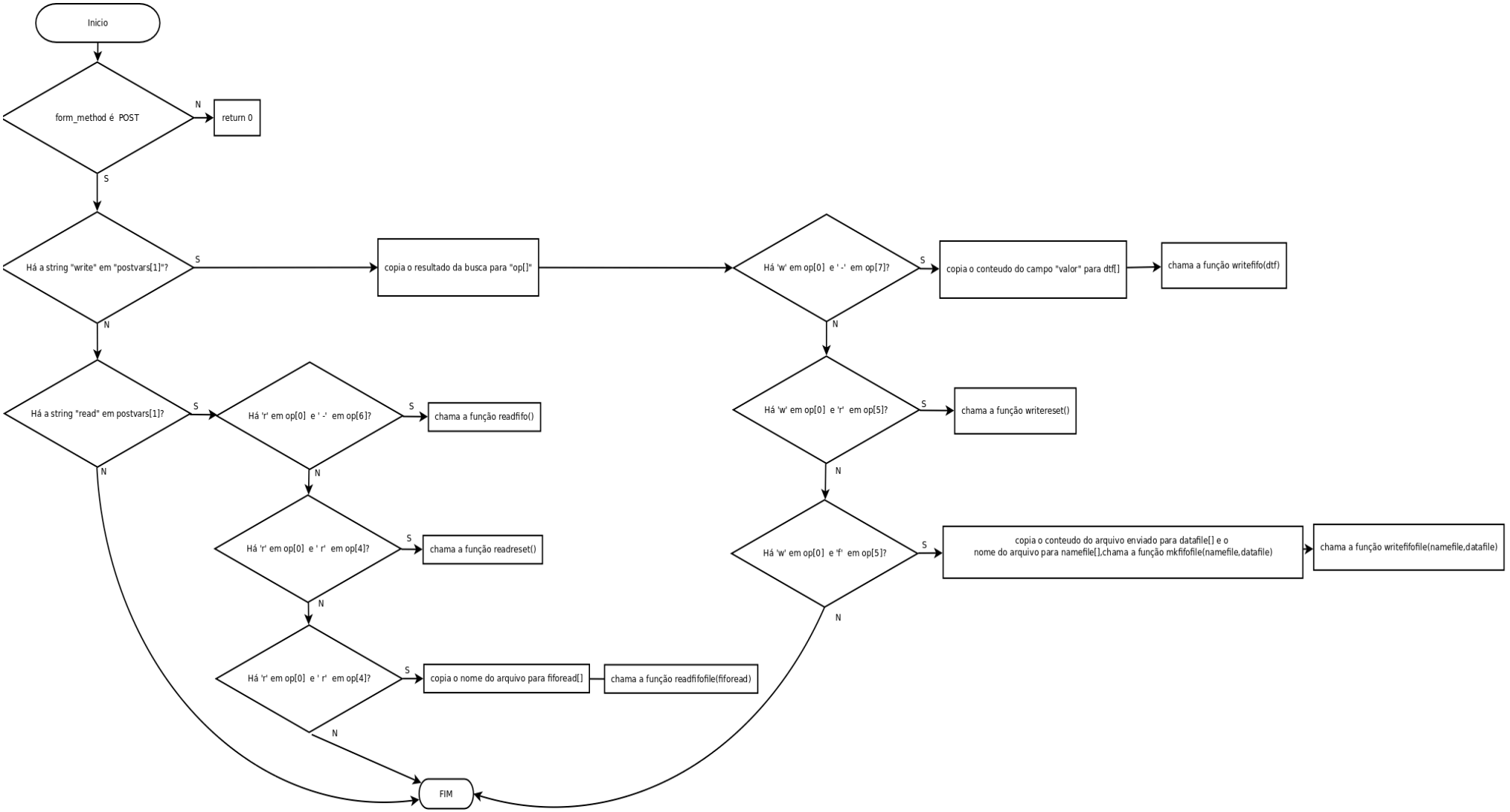

Figura 8: Fluxograma de funcionamento do código template.c

Basicamente é feita a busca da palavra "*write*" e ou "*read*" dentro da string postvars[1], se houver ocorrência da palavra "write" e ou "read" então o resultado da busca é copiado para um variável auxiliar chamada op[]. Após a cópia é feito uma verificação dentro de op[] para então identificar a operação escolhida. De acordo com a opção escolhida outros dados são buscados dentro de postvars[]. Só depois que os dados necessários são coletados, é que então é chamado um programa em C que acessa a FIFO de acordo com a opção escolhida pelo usuário.

3.6 *Interação entre os códigos do pacote cgi_generic*

A interação entre os códigos do pacote *cgi_generic* [Bay, 2000] e os códigos de escrita e leitura na FIFO inicia-se a partir do momento em que o servidor web recebe a requisição da página web feita pelo usuário no *web browser* e então executa o CGI, conforme mostra a Figura 9.

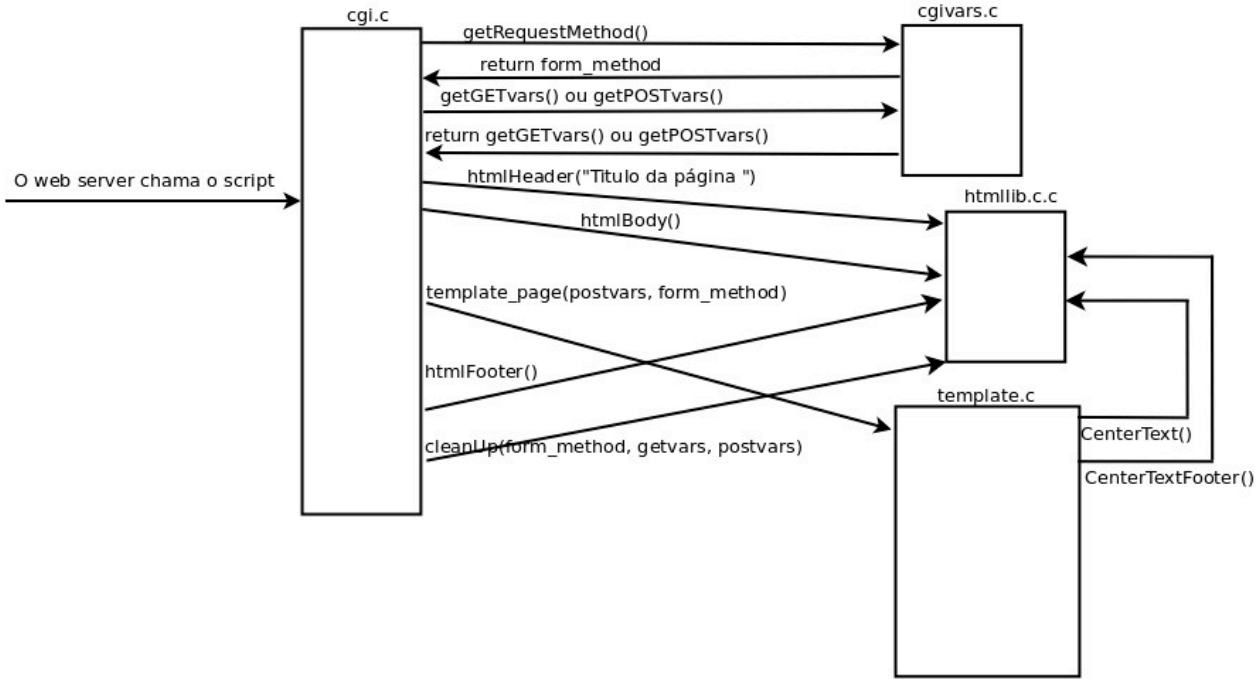

Figura 9: Interação entre os códigos do pacote cgi_generic

No programa cgi.c é requisitado através da função getRequestMethod(), implementado pelo código cgivars.c em conformidade com a RFC3875, que obtém o método usado no CGI . O método usado é retornado pela função getRequestMethod() ao cgi.c. De acordo com

método usado é então chamada a função getGETvars() ou getPOSTvars() que também são implementadas por cgivars.c. A função getGETvars() ou getPOSTvars() retornam ao cgi.c as informações colhidas do form HTML. Na seqüência é chamada a função htmlHeader(char *title) que é implementada pelo código htmllib.c, e insere a Tag HTML do título na pagina.

A próxima função a ser chamada dentro de cgi.c, que também pertencente a htmllib.c é a função htmlBody() a qual insere a tag HTML indicando o inicio do corpo da página. Logo após, é chamada a função template_page(postvars, form_method) tendo como argumentos o método usado e o conteúdo do form HTML. Essa função é implementada pelo código template.c cuja tarefa é processar os dados passados pelo form HTML provendo então acesso a FIFO de acordo com a opção selecionada, conforme mostrado no digrama de funcionamento da segunda parte do código template.c.

Só depois do acesso a FIFO é que retorna-se a execução ao código cgi.c e é chamada a função htmlFooter(), implementada pelo código htmllib.c . Essa função insere o final do corpo da página e do HTML. Para finalizar é então chamada a função cleanUp(form_method, getvars,postvars) , implementada pelo código cgivars.c,a qual "limpa" as variáveis que são passadas como argumento e libera a memória.

3.7 *Concebendo uma Interface Web dedicada para comando de motor de passo*

Visando a integração entre *Controlador de 8 bits Softcore para FPGA* [Destro, 2011] e a FIFO conectada a *Plataforma de Comunicação Ethernet para dispositivos embarcados em FPGAs da Xilinx* [Souza, 2010] iniciou-se o desenvolvimento de uma nova interface web. O acesso a FIFO por meio da nova interface web é regido pelo protocolo de funcionamento em conformidade com o *Controlador de 8 bits Softcore para FPGA* [Destro, 2011]. Tal controlador está conectado a um motor de passo, possibilitando assim o controle do mesmo via rede de computadores. Faz-se então necessário interligar a FIFO ao controlador, este procedimento está descrito em forma de roteiro no Apêndice C .

3.8 *Funcionamento da interface web para comando de motor de passo*

A interface web para comando de motor de passo tem um funcionamento baseado na interface para acesso a FIFO. O programa divide-se em duas partes. A primeira parte é constituída por programação HTML, segunda parte é a programação para o processamento da informação enviada pelo usuário através da página web, conforme o protocolo do controlador ligado a ela. O fluxograma de Figura 10 descreve o funcionamento do código *template.c* para comando do motor.

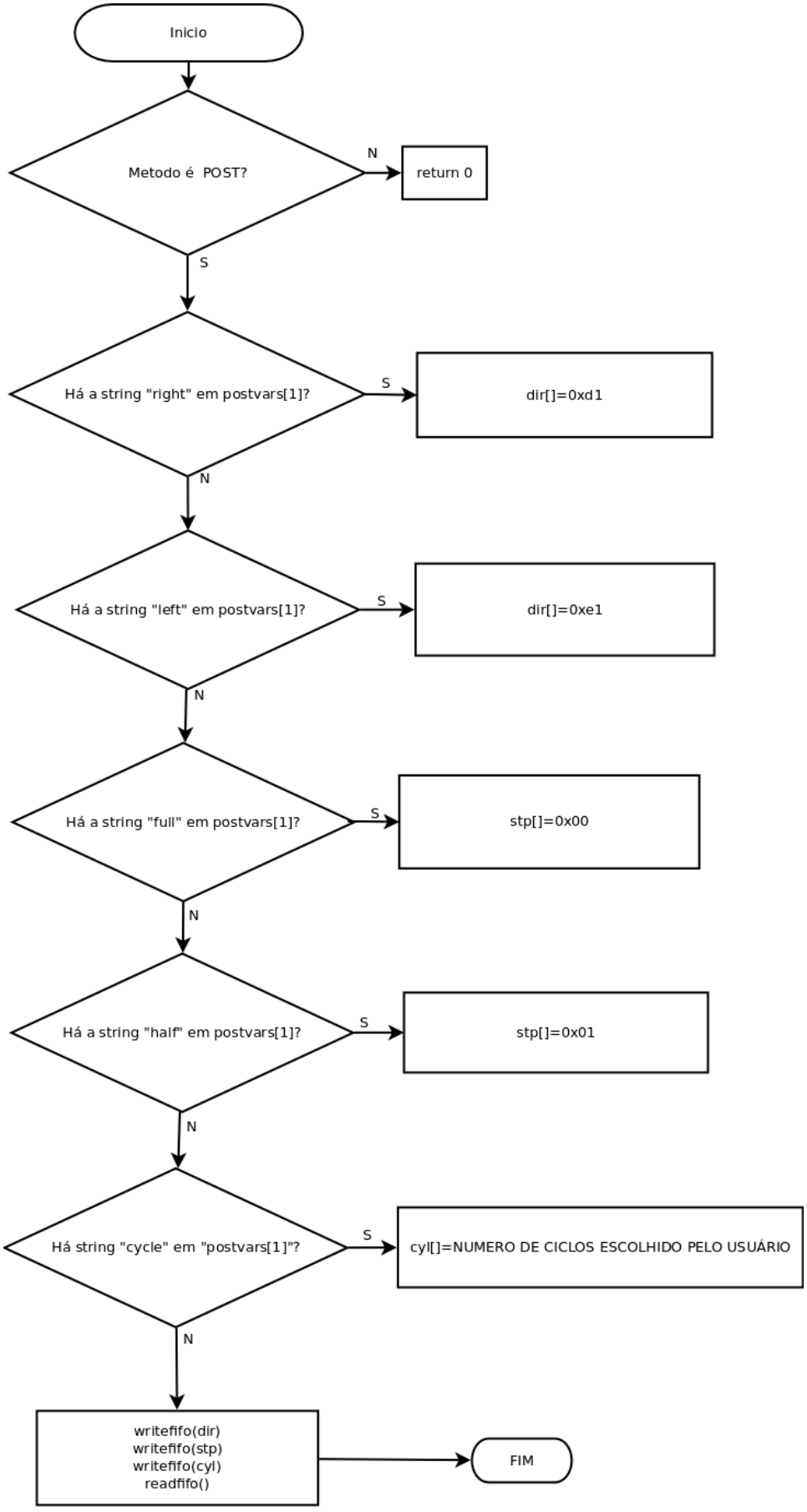

Figura 10: Fluxograma de funcionamento da interface web para motor de passo

Basicamente é feita uma busca das strings *right*,*left*,*ful cyclel*,*half cycle* de modo a filtrar a opção escolhida pelo usuário. De acordo com a escolha é definido o valor que será escrito na FIFO como pode ser observado na Tabela 5.

Valor a ser escrito na GPIO_to_FIFO	Função
0x1D	Motor anda para direita
0x1E	Motor anda para esquerda
0x00	Meio passo
0x01	Passo completo
Valor a ser lido na FIFO_to_GPIO	
0xd1	Fim de curso Direito
0xe2	Fim de curso esquerdo
0x01	Operação bem sucedida sem fim de curso

Tabela 6 : Protocolo de funcionamento do controlador

Depois de filtradas as informações, é feito a escrita dos dados seqüenciais na FIFO respeitando o protocolo de funcionamento do controlador. Esta seqüência de escrita corresponde a três comandos, respectivamente: direção do passo, tipo do passo e quantidade de passos a serem executados. O protocolo de comunicação com o controlador é bidirecional, ou seja, não se dá num único sentido, todo envio de seqüência de comandos tem uma resposta gerada pelo controlador, esta reposta pode ser de execução bem sucedida ou de fim de curso conforme descrito na Tabela 5. A Figura 11 exemplifica o comando do motor de passo através da interface web tendo como resposta o dado de fim de curso Direito. Já a Figura 12 exemplifica o comando do motor de passo tendo como resposta o dado de uma operação bem sucedida.

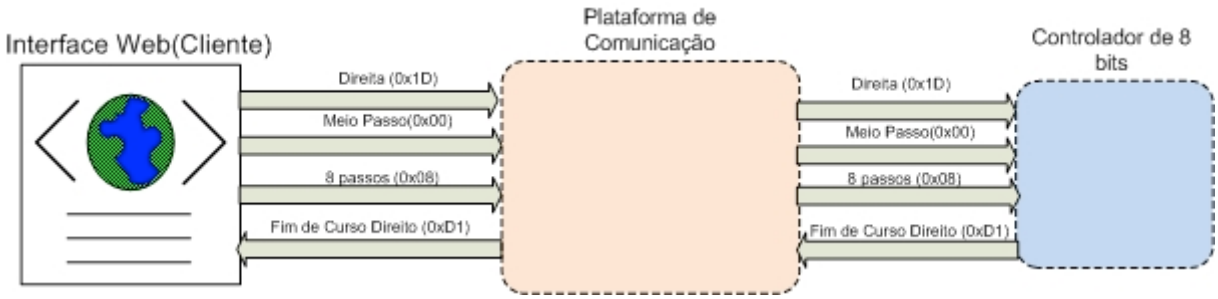

Figura 11: Comunicação dado de resposta Fim de Curso Direito

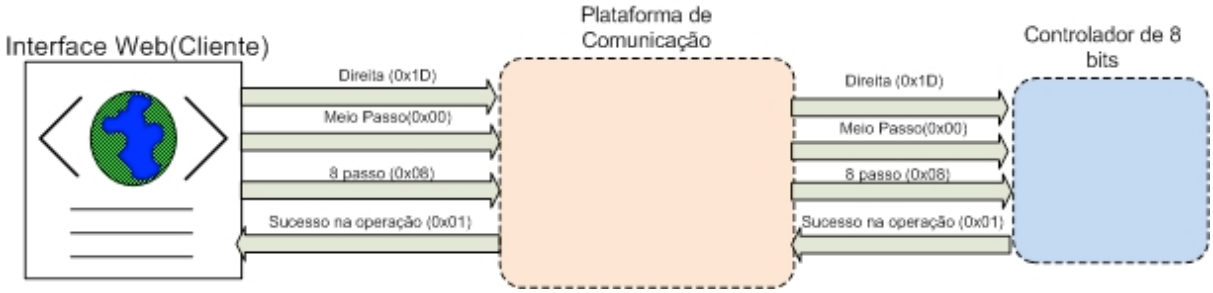

Figura 12: Comunicação dado de resposta Sucesso na operação

3.9 *Funcionamento da interface web para comando de motor de passo*

A interface web para controle do motor de passo é constituída por dois *menulist*, *Direction* e *Step* , também constituída por um campo *Cycles* e botão send conforme a Figura 13.

Figura 13: Interface para controle ao motor de passo

No *menulist Direction* é escolhida a direção em que o motor irá executar o passo, já no *menulis*t *Step* é escolhido o tipo de passo. No campo *Cycle*, o usuário deve colocar a quantidade de passo. Após a escolher a direção, tipo de passo e a quantidade de passo, basta só dar um *click* no botão *send* e os dados serão enviados. Na seqüência será recebida a confirmação de passo bem sucedido ou de fim de curso.

4 *Análise dos Resultados Obtidos*

Após a concepção e desenvolvimento de uma interface web que agregue simplicidade e flexibilidade no controle de dispositivos sintetizados na lógica combinacional do FPGA, iniciou-se a fase de testes e validação do projeto. A primeira fase de testes foi aplicado sobre a interface que acessa a *FIFO*. Através de um *browser* foi possível acessar via rede de computadores a interface web hospedada na plataforma de comunicação para o Kit Spartan3E500. Tal acesso permitiu, por meio de *clicks* do *mouse* ou por envio de arquivos de texto, escrever e ler dados de 8 bits na *FIFO*. Na Figura 14 é feita a operação de escrita da *FIFO*, como exemplo, é escrito o dado 0F.

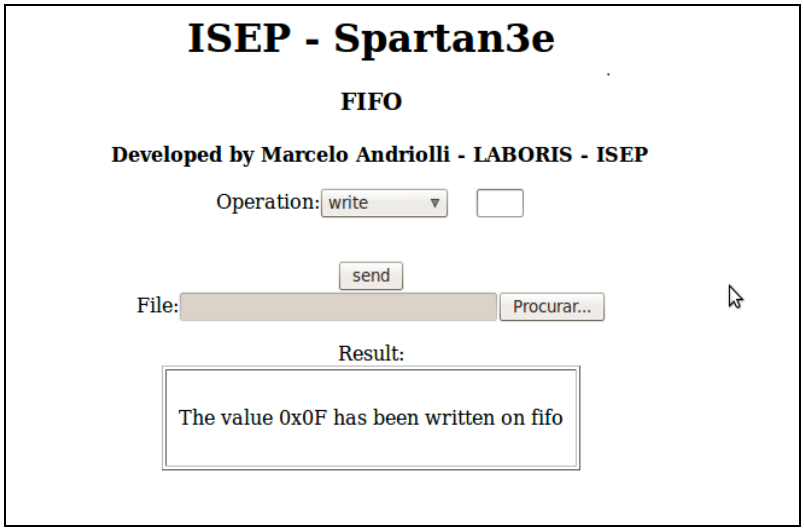

Figura 14: Escrita na FIFO via interface web

Já na Figura 15 é executada a operação de leitura do dado escrito anteriormente, ou seja, é lido o valor 0F.

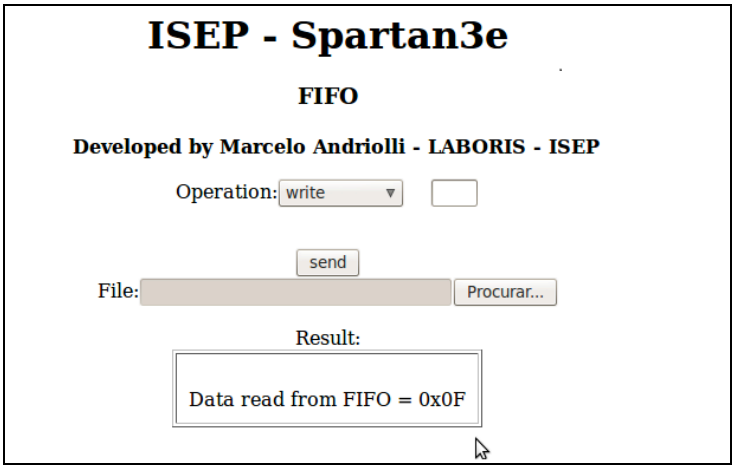

Figura 15: Leitura da FIFO via interface Web

Nos testes feitos com a interface web para controle de motor de passo, através da interface web foi possível controlar o motor de passo com eficiência. A única limitação encontrada, é relacionado a leitura por arquivo na FIFO. O download do arquivo de leitura não foi possível diretamente através da interface web. Tal limitação se dá por conta de restrições de leitura e escrita imposta pelo sistema de arquivos utilizados no sistema operacional embarcado, o Petalinux [PETALOGIX, 2009]. Na Figura 16 é feito o comando para mover o motor para direita, devido ao motor ter atingido o fim de curso é enviado o dado correspondente ao fim de curso.

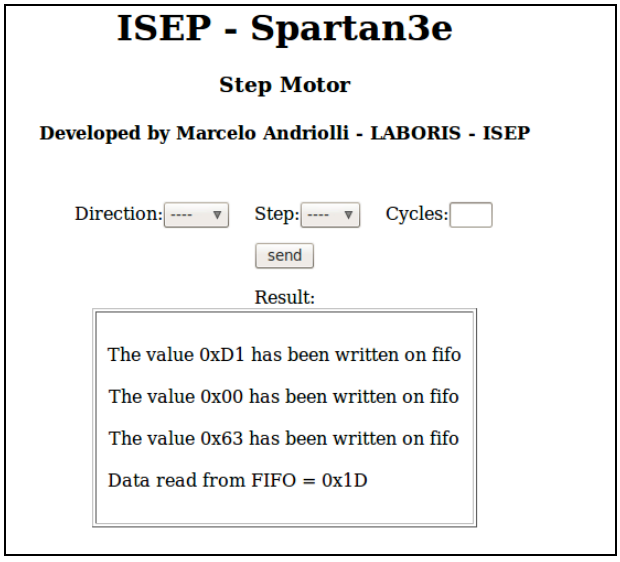

Figura 16: Controle de motor de passo com fim de curso

Já na Figura 17, é feito o comando para mover o motor de passo para esquerda, como não houve o fim de curso então a operação foi executada sem fim de curso, ou seja, com sucesso.

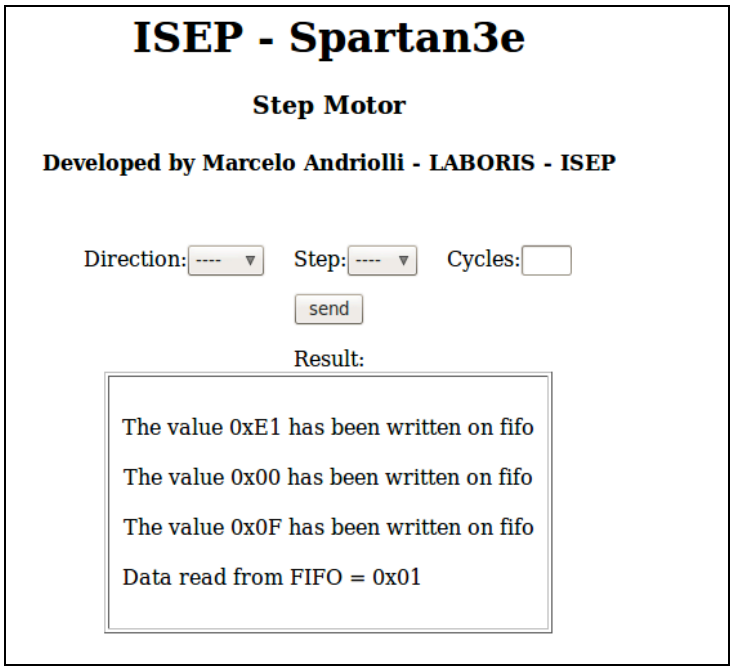

Figura 17: Controle de motor de passo sem fim de curso

5 *Conclusões e trabalhos futuros*

Como propostas de melhorias para a *Plataforma de Comunicação Ethernet para dispositivos embarcados em FPGAs da Xilinx* [Souza, 2010], inicialmente foram definidas: Desenvolvimento de uma interface web, tornar transparente o processo de concepção do Software embarcado que compõe a plataforma e prover o acesso ao *display* de LCD do kit utilizado. A interface desenvolvida neste projeto permite a "tradução" comandos em *click* de mouse, transferências de arquivos para a plataforma de comunicação de forma transparente e o acesso ao dispositivo alvo via navegador web, cumprindo os requisitos assim desejados para esta interface. Para o objetivo proposto neste trabalho, não havia sido encontrada na literatura nenhuma interface para dispositivos lógicos programáveis em FPGA que permite o controle ou acesso remoto via web browser em conformidade com a norma IEEE1149.1.

Por se tratar de uma interface experimental a mesma possui limitações, são elas: Difícil Manutenção, Solicitações de dados Síncrono, Clientes concorrentes e Download de forma transparente de arquivos de leitura. A interface foi desenvolvida em linguagem C , tal escolha se deu pela limitação do sistema operacional usado na Plataforma de Comunicação descrita no Capítulo 3. Outra opção para o desenvolvimento da mesma interface, seria o uso de *Shell Script*, o tornando a manutenção de sistema mais simples. Com ela seria possível usar algumas ferramentas oferecida pelo sistema operacional simplificando algumas tarefas que em linguagem C seriam complexas. A compilação de uma alteração na interface web feita em liguagem C acarretaria numa nova compilação do sistema operacional sendo necessário fazer o *boot* do sistema operacional a cada alteração. Atualmente a solicitação de dados através da interface web é síncrona, ou seja, a leitura de um dado enviado do dispositivo alvo para a plataforma de comunicação se dá por meio de uma requisição da interface web , feita através do envio de um formulário HTML pelo botão *send* com a opção *read* no menulist *operation* selecionada, para o servidor web. É necessário que a interface solicite a leitura para que então seja apresentado no *browser* o dado enviado. Outra limitação

da plataforma é a concorrência entre clientes, ou seja dois usuários acessando a interface web para controlar ou comandar o mesmo dispositivo alvo. Não foi feito nenhum tratamento de hardware ou software para concorrência entre clientes, sendo assim as ações feitas por um usuário sobrepõe uma ação feita por outro usuário. Uma das propostas de melhoria previa a transferência de arquivos de forma transparente ao usuário. Tal melhoria foi implementada permitindo o envio de arquivo através da interface web, porém o download do mesmo não é possível de forma transparente por conta de restrições de modificação de permissão de acesso de alguns diretórios durante a configuração do sistema de arquivos usado no Sistema Operacional usado pela Plataforma de Comunicação.

Durante o desenvolvimento da interface web ocorreram limitações referentes ao sistema operacional, fazendo com que fosse gasto mais tempo do que o estimado inicialmente no desenvolvimento da interface web. Notou-se que durante esse processo, que prover o acesso ao *display* de LCD do kit utilizado seria uma particularidade do dispositivo alvo que não seria explorada nos testes com a interface web, sendo assim o acesso ao *display* de LCD não está disponível.

Melhorias na interface web, como o tratamento de clientes concorrente e requisições de dados assíncronas, tornariam a interface web mais robusta e dinâmica. Bem como o desenvolvimento de uma ferramenta para tornar transparente o processo de concepção do Software embarcado baseado em shell script com zenity/GNOME e download transparente de arquivos de leitura, seguem como sugestões de trabalhos futuros.

Referências Bibliograficas

[Preece, 1994]. Peece, J.; Rogers, Y.; Sharp, H.; Benyon,D.; Holland, S.; Careu,T. (1994) Human-Computers Interaction. Reading, MA. Addison-Wesley.

[Preece et al.,1994] - “Human-Computer Interaction”. Preece et al.'94,AddisonWesley

[Moran, 1981]. Moran, T. (1981) “The Command Language Grammars: a represetantion for the user interface of interactive computer systems. International Journal of Man- Machine Studies, 15, 3-50.

[Shneiderman, 1998] . Shneiderman, B. (1998) Designing the User Interface, 3rd Edition. Reading, MA: Addison Wesley.

[Souza,C et al,1999]. Souza,C.; Leite, J.;Prates, Projeto de Interfaces de Usuário,Perspectivas Cognitivas e Semióticas(1999)

[Tanenbaum, A. S., 2002]. Tanenbaum, A. S. Computer networks. 4. ed. [S.l.]: Prentice Hall, 2002

[Phillips, 2005]. Paul Phillips,Boa Webserver,2005

[Souza, 2010]. Souza,N. Rodrigo, Plataforma de Comunicação Ethernet para dispositivos embarcados em FPGAs da Xilinx,2010

[Destro, 2011]. Destro, C. Mario, Controlador de 8 bits Softcore para FPGA,2011

[Rocha e Baranauskas, 2003]. Rocha, V. Heloísa ; Baranauskas. C. Maria Cecília, Design Avalia de Interfaces Humano-Computador,2003

[PETALOGIX, 2009]. http://www.petalogix.com/products/petalinux - Acessado dia 16 de Novembro de 2010

[Tanenbaum, A. S., 1999]. Tanenbaum, A. S. Modern Operating Systems, 2nd ed. Prentice Hall,1999.

[ACME Laboratories,2003] . http://acmelab.com/ - Acessado dia 07 de Janeiro de 2011

Apêndice A. Habilitando o pacote cgi_generic no Petalinux

Para habilitar o pacote cgi_generic no Petalinux , acesse o diretório do Petalinux com o comando a baixo:

$cd petalinuxv0.40final/

Na sequencia execute os script ./settings.sh:

$cd source ./settings.sh

Entre no diretório da distribuição do Petalinux:

$cd $PETALINUX/software/petalinuxdist

Acesse o menu de configuração:

$cd make menuconfig

No menu principal conforme a Figura 18, vá em Kernel/Library/Defaults Selection.

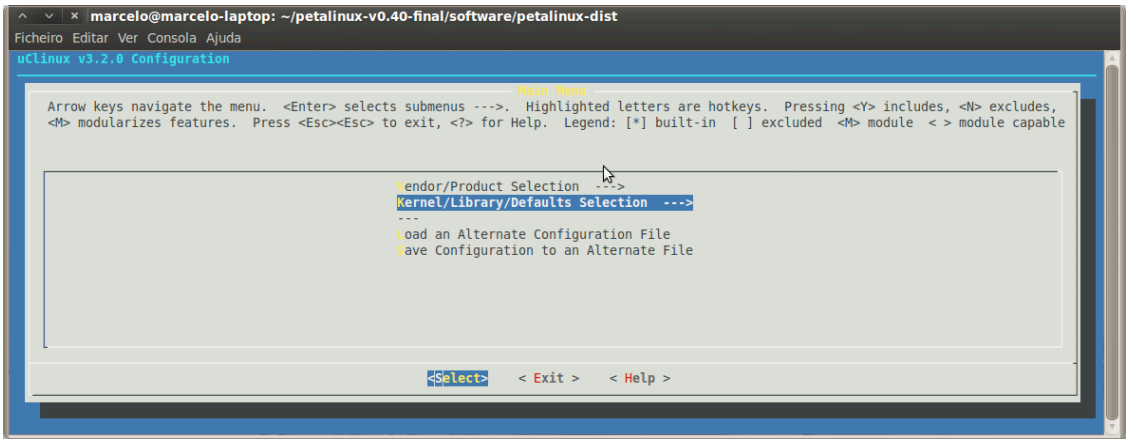

Figura 18 : Menu Principal do Petalinux

Marque a Opção Customize Vendor/User Settings conforme a Figura 19, saia e salve as alterações.

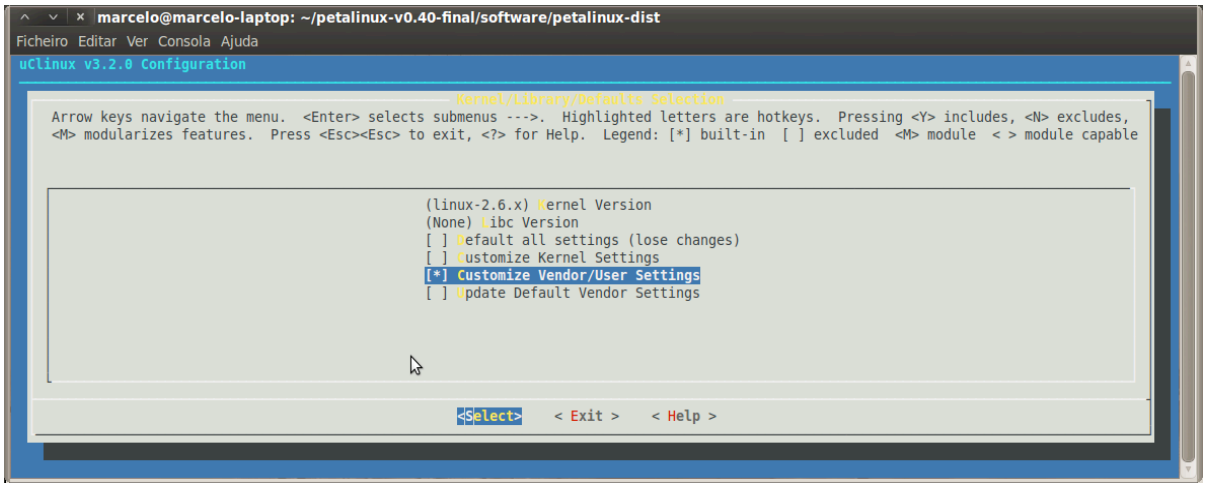

Figura 19: Menu Kernel/Library/Defaults Selection

Na seqüência irá aparecer outro menu de configurações, vá em Miscellaneous Configuration conforme a Figura 20.

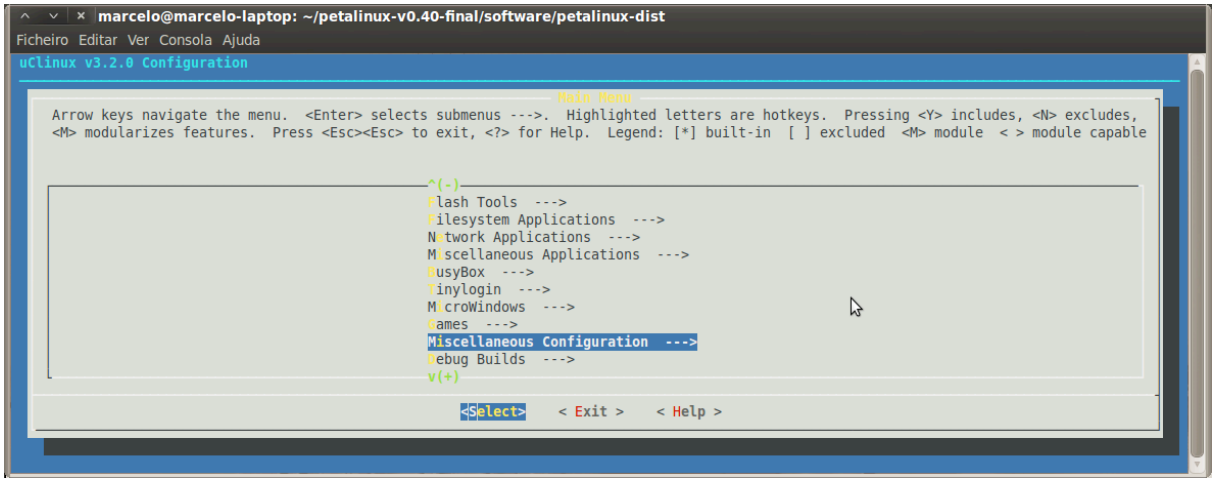

Figura 20 : Miscellaneous Configuration

Marque a opção generic_cgi conforme Figura 21, volte ao menu saia e salve as alterações.

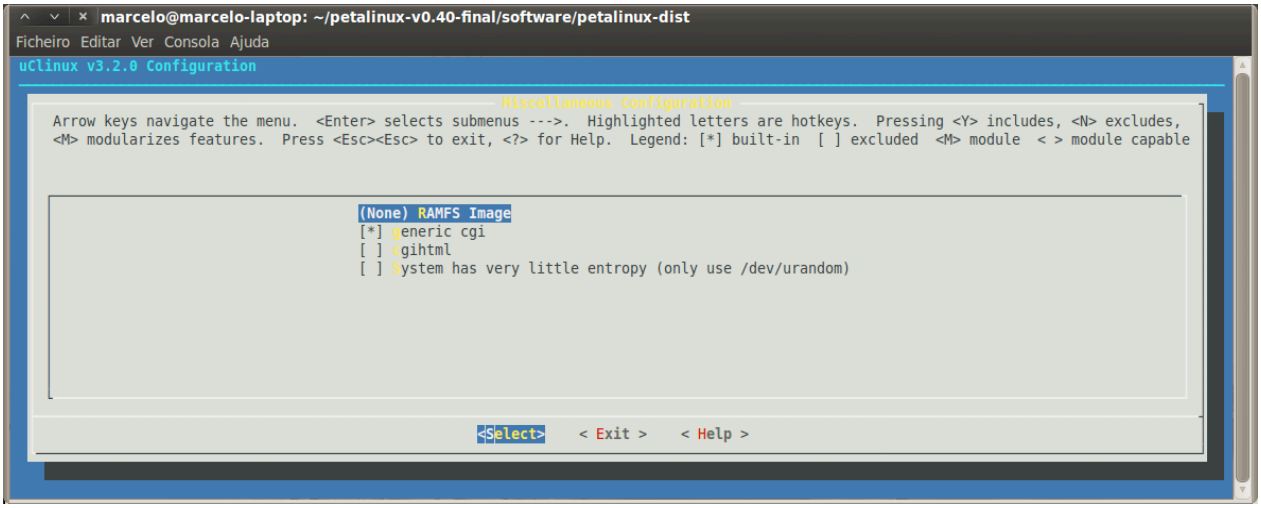

Figura 21 : Marcando cgi_generic

Logo depois de marcar a opção gci_generic, saia do menuconfig e salve as alterações. Basta agora compilar o Petalinux com o seguinte comando:

```
$yes "" | make oldconfig dep all
```

Após a compilação bem sucedida do Petalinux o pacote cgi_generic estará disponível para ser usado no diretório …../petalinux-v0.40-final/software/petalinux-dist/usr/cgi_generic/

Apêndice B. Interligação entre códigos em C para acesso a FIFO e gci_generic

Para integrar os códigos em C para acesso a FIFO ao pacote de códigos cgi_generic[Bay,2000] é necessário criar arquivos com o nome: mkfifofile.c, mkfifofile.h, readfifo.c, readfifofile.c, readfifofile.h, readfifo.h, readreset.c, readreset.h, readreset.h, readreset.h, writefifo.c, writefifofile.c, writefifofile.h, writefifo.h, writereset.c,writereset.h. Crie dentro do diretório ../Petalinux-v0.40-final/software/petalinux-dist/usr/cgi_generic/. Para cria-los digite o comando a baixo:

$ touch <nomedo_arquivo>

Após ter criado os arquivos citados a cima, edite-os e copie o conteúdo de acordo com o nome de cada código presente no cd dentro do diretório Códigos/C Codes/. Edite o arquivo Makefile de modo que fique como abaixo mostrado:

```
EXEC = cgi

OBJS = cgi.o cgivars.o htmllib.o template.o writefifo.o
readfifo.o writereset.o readreset.o mkfifofile.o
writefifofile.o readfifofile.o


all: $(EXEC)

romfs:
	$(ROMFSINST) $(ROOTDIR)/vendors/Generic/httpd
/home/httpd
	$(ROMFSINST) -d $(EXEC) /home/httpd/cgi-bin/cgi


$(EXEC): $(OBJS)
	$(CC) $(LDFLAGS) -o $@ $(OBJS) $(LDLIBS)


clean:
	-rm -f $(EXEC1) *.elf *.gdb *.o

$(OBJS): cgivars.h htmllib.h template.h writefifo.h
readfifo.h writereset.h readreset.h writefifofile.h
mkfifofile.h
```

Edite o #include dos seguintes arquivos: readfifo.c, readfifofile.c, readreset.c, , writefifo.c, writefifofile.c, writereset.c. Altere o #include a baixo que está presente nos arquivos a cima:

```
#include</home/marcelo/petalinuxv0.40final/software/linux2.6.xpetalogix/include/linux/xgpio_ioctl.h>
/* Linux Xilinx GPIO library for ioctl functions */
```

Coloque o caminho absoluto do diretório do qual localiza-se a biblioteca xgpio_ioctl.h conforme a baixo:

```
#include</home/diretório_do_usuário/petalinuxv0.40final/software/linux2.6.xpetalogix/include/linux/xgpio_ioctl
.h> /* Linux Xilinx GPIO library for ioctl functions */
```

Altere também o arquivos htmllib.c substituindo o arquivo original pelo do arquivo que está no cd no …./Códigos/C Codes/ .

Apêndice C. Procedimento de Acesso a FIFO via interface

Escrita de modo simples:

1 – Selecione a no *menulist Operation* a opção *writefifo*.

2 – Digite no campo valor o valor em hexadecimal a ser escrito na FIFO ex: 0F

3 – Pressione o botão *send* para que os dados sejam enviados

Perceba que após ter sido feito o procedimento a cima será mostrada em *result* o valor que foi escrito na FIFO.

Leitura de modo simples:

1 – Selecione a no *menulist Operation* a opção *readfifo*.

2 – Pressione o botão *send* para seja feita a leitura

Perceba que após ter sido feito o procedimento a cima será mostrada em *result* o valor que foi lido na FIFO.

Escrita de modo seqüencial:

1 – Selecione a no *menulist Operation* a opção *writefifofile*.

2 – Pressione o botão procurar para procurar o arquivo a ser enviado, depois de encontrado selecione o mesmo.

3 – Pressione o botão *send,* para enviar o arquivo.

Perceba que após ter sido feito o procedimento a cima será mostrada em result o conteúdo do arquivo e o estado da FIFO após a escrita. Citações de outros trabalhos apresentam uma forma para incluir tabelas no documento. O Capítulo 4 apresenta as conclusões deste trabalho além de apresentar os trabalhos futuros.

Leitura de modo sequencial:

1 – Selecione a no *menulist Operation* a opção *readfifofile*.

2 – Pressione o botão procurar para procurar o mesmo arquivo envidado no procedimento de escrita de forma indireta, ou seja, *writefifofile*. Esse procedimento faz-se necessário para que seja informado o nome do arquivo que será salvo os dados lidos na FIFO. Exemplo: se você enviou para escrita com arquivos na FIFO o arquivo escritafifo.txt terá que selecionar o mesmo arquivo no procedimento de leitura de forma indireta, ou seja, *readfifofile*.

3 – Pressione o botão send para ser feita a leitura da FIFO. Perceba que após ter sido feito o procedimento a cima, será mostrado em result os dados que foram lidos na FIFO de escrita pela FIFO de leitura e salvados no arquivo que é mostrado mais a baixo. É possível também obter o arquivo, bastar acessar via ftp. Para isso abra o console ou na linha de comando do Linux e digite o comando:

```
$ ftp <ip_configurado_para_o_kit>
```

Após isso faço o login, o usuário é root e senha root. Após o login digite o comando:

```
ftp> cd tmp/
```

Agora execute o comando a baixo para baixar o arquivo:

```
ftp> get <nome_do_arquivo>
```

Exemplo:

```
ftp> get readfile_FIFO.txt
```

O arquivo será baixado para o diretório corrente do usuário logado

Apêndice D. Interligação do Controlador com a FIFO

Para interligar o controlador a FIFO, basta copiar todos os arquivos existentes no cd dentro do diretório *VHDL/Integração Controlador de 8Bits/* para o diretório do projeto no ISE. Após isso faça a verificação de sintaxe do projeto.