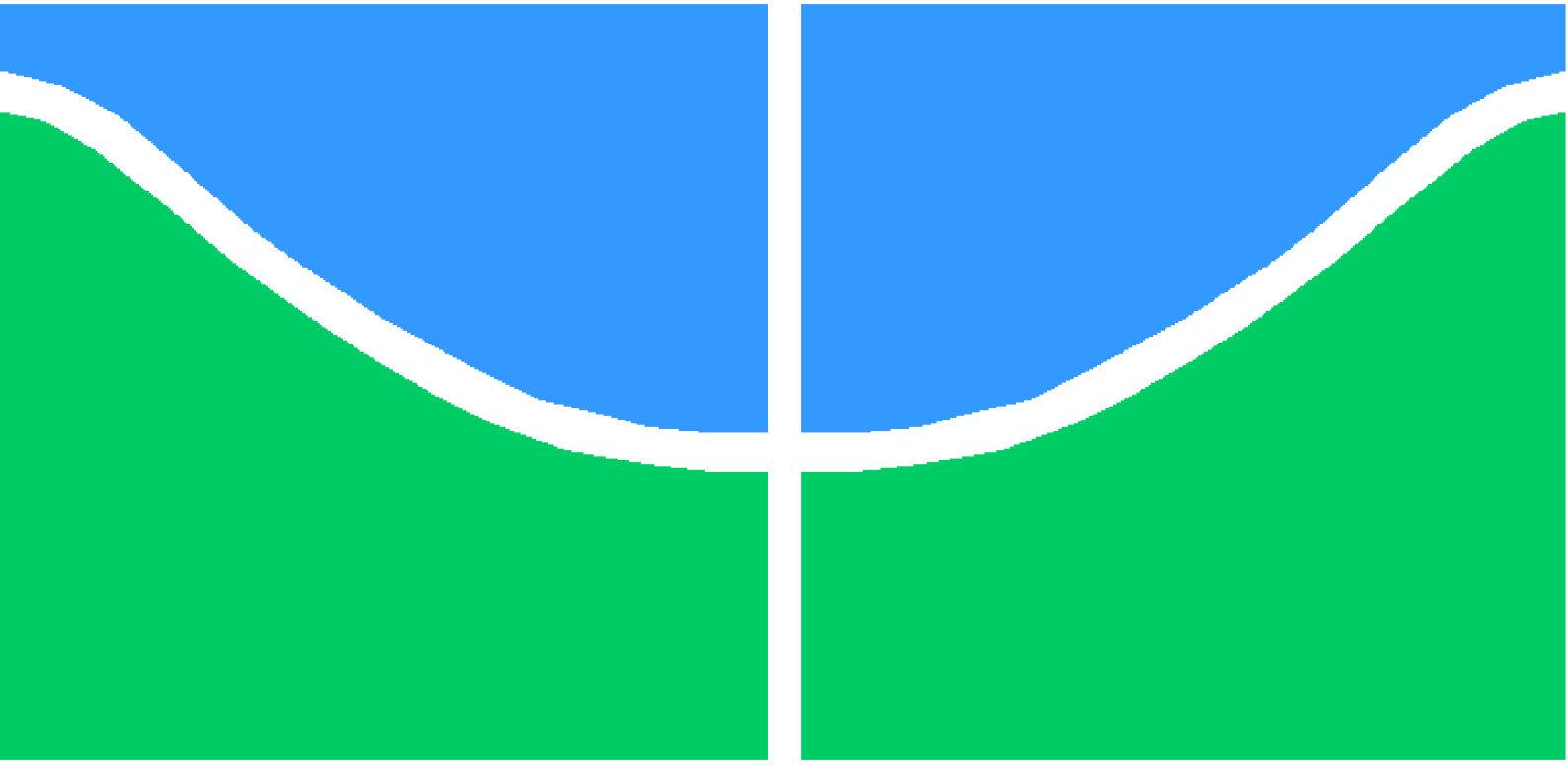


Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Engenharia Eletrônica

Laboratório Remoto Baseado em FPGA Aplicado nas Disciplinas de Prática de Eletrônica Digital 1 e 2 da Faculdade UnB Gama

Autor: Rodrigo Bonifácio de Medeiros

Orientador: Prof. Dr. Daniel Muñoz Arboleda

Brasília, DF

2018

Rodrigo Bonifácio de Medeiros

Laboratório Remoto Baseado em FPGA Aplicado nas Disciplinas de Prática de Eletrônica Digital 1 e 2 da Faculdade UnB Gama

Monografia submetida ao curso de graduação em Engenharia Eletrônica da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Eletrônica.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Prof. Dr. Daniel Muñoz Arboleda

Brasília, DF

2018

Rodrigo Bonifácio de Medeiros
Laboratório Remoto Baseado em FPGA Aplicado nas Disciplinas de Prática de Eletrônica Digital 1 e 2 da Faculdade UnB Gama/ Rodrigo Bonifácio de Medeiros. – Brasília, DF, 2018-
54 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Daniel Muñoz Arboleda

Trabalho de Conclusão de Curso 1 – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2018.

1. Palavra-chave01. 2. Palavra-chave02. I. Prof. Dr. Daniel Muñoz Arboleda. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Laboratório Remoto Baseado em FPGA Aplicado nas Disciplinas de Prática de Eletrônica Digital 1 e 2 da Faculdade UnB Gama

CDU 02:141:005.6

Rodrigo Bonifácio de Medeiros

Laboratório Remoto Baseado em FPGA Aplicado nas Disciplinas de Prática de Eletrônica Digital 1 e 2 da Faculdade UnB Gama

Monografia submetida ao curso de graduação em Engenharia Eletrônica da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Eletrônica.

Trabalho aprovado. Brasília, DF, 03 de julho de 2018 – Data da aprovação do trabalho:

Prof. Dr. Daniel Muñoz Arboleda
Orientador

Prof. Dr. Gilmar Silva Beserra
Convidado 1

Prof. Dr. Guillermo Alvarez Bestard
Convidado 2

Brasília, DF
2018

Resumo

O presente projeto visa a criação de um laboratório remoto em FPGA aplicado nas disciplinas de Prática de Eletrônica Digital 1 e Prática de Eletrônica Digital 2 da Faculdade UnB Gama a fim de mitigar o problema da falta dos kits de desenvolvimento da Basys 3 na Faculdade UnB Gama, possibilitando os alunos a trabalharem fora do ambiente acadêmico e como consequência melhorar o processo de ensino-aprendizagem, além de ser uma solução mais barata do que a aquisição de novos kits. Trata-se do desenvolvimento de uma arquitetura de hardware embarcada no FPGA e um modelo de aplicação cliente-servidor para a execução de uma interface gráfica *web*. O usuário é capaz de implementar seu projeto VHDL e interagir com a placa remotamente, a partir das chaves, botões, LEDS e *displays* de sete segmentos virtuais disponíveis na interface simulando esses componentes físicos do FPGA, além de acompanhar ao vivo a partir de uma transmissão *web* os resultados no kit Basys 3. Experimentos usando circuitos sequenciais e máquinas de estados demonstram a correta interação entre o servidor, a placa e o usuário, portanto a solução é viável e demonstrou ser facilmente replicável para ser aplicado em qualquer tipo de servidor e FPGAs.

Palavras-chaves: Laboratório Remoto. FPGA. Basys 3. Servidor *Web*. *Framework Web Flask*.

Abstract

The current project aims the establishment of a remote FGPA laboratory in the subject Digital Electronic Practice 1) and Digital Electronic Practice 2 of University of Brasilia (UnB) – Gama Campus, in order to minimize the lack of Basys 3 development kit problem at UnB Gama, making it possible for students to work outside the academic environment and, as a consequence, improve the teaching-learning process, it is also a cheaper solution rather than purchasing new kits. It is the development of a hardware architecture embedded in FPGA and a client-server model application to run a graphic web interface. The user is able to implement the VHDL project and interact with the board remotely, from switches, buttons, LEDs and displays of seven virtual segments available on the interface simulating these FGPA hardware components, as well as follow the results of Kit Basys 3 live from a web live video broadcast. Experiments using sequential circuits and state machines show the right interaction between server, board and user, thus the solution is viable and showed to be easily replicable.

Key-words: Remote Lab. FPGA. Basys 3. Web Server. Framework Web Flask.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

FPGA	*Field Programmable Gate Array*
UnB	Universidade de Brasília
FGA	Faculdade UnB Gama
PED	Prática de Eletrônica Digital
IP	Internet Protocol
MATLAB	*MATrix LABoratory*
WEB	*World Wide Web*
RELLE	*Remote Labs Leraning Environment*
GT-MRE	Grupo de trabalho em experimentação remoto
RExLab	Laboratório de experimentação remota
UFSC	Universidade Federal de Santa Catarina
IoT	*Internet of Things*
UART	*Universal Asynchronous Receiver/Transmitter*
LED	*Light Emitting Diode*
FSM	*Finite State Machine*
I/O	*Input/Output*
LUT	*look-up table*
MUX	Multiplexador
FF	*Flip-Flops*
CMOS	*Complementary Metal Oxide Semiconductor*
CLB	*Configurable Logic Blocks*
ROM	*Read Only Memory*
DSP	*Digital Signal Processing Blocks*

DCM	*Digital Clock Manager*
USB	*Universal Serial Bus*
MMCM	*Mixer Mode Clock Manager*
JTAG	*Joint Test Access Group*
VHDL	*Very High Speed Integrated Circuits Hardware Description Language*
MSI	*Medium-Scale Integration*
RTL	*Register Transfer Level*
HTML	*HyperText Markup Language*
CSS	*Cascading Style Sheet*
MVC	*Model-View-Controller*
ASCII	*American Standard Code for Information Interchange*
URL	*Uniform Resource Locator*
PWM	*Pulse Width Modulation*
DNS	*Domain Name System*

Lista de símbolos

&	Concatenar
°	Grau
Ω	ohms
Kb	*kilobits*
Hz	hertz
MHz	Megahertz
ms	milissegundos
W	watts
%	Por cento

Sumário

1 Introdução

1.1 Contextualização

No desenvolvimento do aprendizado, em especial, em disciplinas que envolvem algum tipo de método prático é necessário o uso de ferramentas físicas (MA; JEFFREY; NICKERSON, 2006). No caso particular das disciplinas de prática de sistemas digitais, que requerem o uso de placas de desenvolvimento, uma ferramenta essencial são os dispositivos *Field Programmable Gate Arrays* (FPGA), porém adquirir um kit de desenvolvimento FPGA é difícil devido ao alto custo e a dificuldade de aquisição. Por exemplo, um kit Basys 3 (família FPGA Artix-7 da Xilinx) (DIGILENT, 2016), custa 149,00 dólares (DIGILENT, 2018), sem contar a taxa de câmbio, impostos e fretes. Além da limitação de recursos financeiros, o procedimento adotado pelo sistema de compras públicas da Universidade de Brasília (UnB) é demorado para realizar a aquisição de equipamentos importados, fato que dificulta a compra dos kits de desenvolvimento FPGA.

Os fatos mencionados acima acarretam na escassez dessas ferramentas no ambiente acadêmico, atualmente, a Faculdade UnB Gama (FGA) possui treze kits de desenvolvimento da Basys 3, porém cada turma de Prática de Eletrônica Digital (PED) 1 e 2 possui em média quarenta alunos, sendo seis turmas para a primeira disciplina e duas para a segunda (MATRICULAWEB, 2018), o que faz com que um kit FPGA seja dividido muitas vezes por até quatro estudantes. Nesse contexto, o estudante encontra sérias dificuldades para trabalhar além dos horários de aula ou levar os kits FPGA para fora da universidade. Todos esses fatores dificultam o processo de ensino-aprendizagem o que atinge tanto professores como alunos. A tabela 1 mostra um resumo da descrição do problema abordado nesse projeto.

1.2 Justificativa

Um projeto de laboratório remoto de FPGAs possibilita o aluno trabalhar fora do ambiente acadêmico, como no conforto de sua residência ou em outros lugares de sua preferência sem estar fisicamente perto do kit FPGA e sem precisar compartilhar o recurso com outros colegas, podendo repetir as experiências diversas vezes para facilitar a compreensão dos conceitos estudados. Por exemplo, um servidor associado a uma placa com acesso de dez minutos por aluno, em um caso extremo é capaz de ser usado por até cento e quarenta e quatro usuários em apenas um dia, otimizando eficiência de uso de um kit de desenvolvimento.

Tabela 1 – Descrição do Problema.

O PROBLEMA:	Escassez de kit de desenvolvimento FPGA nos laboratórios da FGA.
AFETA:	Os alunos e professores que cursam e lecionam principalmente as disciplinas PED 1 e PED 2.
CUJOS IMPACTOS SÃO:	Compartilhamento de um kit por vários alunos e dificuldade de acesso aos FPGAs nos horários fora de aula.
UMA SOLUÇÃO BEM SUCEDIDA:	Abre a possibilidade dos alunos trabalharem com o FPGA em outros ambientes além do acadêmico; melhora no rendimento de ensino-aprendizagem (MA; JEFFREY; NICKERSON, 2006); mitiga o problema da falta de aquisição dos kits.

Uma vantagem adicional do laboratório remoto é que professores conseguem administrar melhor a disciplina, como por exemplo, aplicando projetos que podem ser executados fora do horário de aula, ganhando tempo para lecionar outras atividades, ou realizando atividades com prazo adicional.

Como citado, o sistema de compras da universidade afeta a importação de kits de desenvolvimento FPGA. O laboratório remoto é capaz de mitigar esse problema (MA; JEFFREY; NICKERSON, 2006).

Para a solução proposta opta-se por um uso de comunicação de dados entre um FPGA e uma interface gráfica *web*, ao invés da criação de uma aplicação para celulares ou computadores, pois aplicações *webs* são mais fáceis de serem desenvolvidas com uso dos *frameworks web* e de serem acessadas sem intermédio de *download* (RODRIGUEZ-GIL et al., 2014). Toda a arquitetura de hardware embarcada no FPGA foi desenvolvida em linguagem de VHDL e projetada para o FPGA Artix-7 disponível no laboratório da FGA. Não foram usados microprocessadores visando diminuir o uso de recursos da placa. Vale ressaltar que o usuário pode fazer uso de microprocessadores (por exemplo o *Microblaze*) ou microcontroladores (por exemplo *Picoblaze*) no seu projeto de hardware.

A comunicação entre entre o FPGA e o servidor foi implementada usando uma transmissão serial através do protocolo RS-232, permitindo que o laboratório remoto emule por software as entradas e saídas sem necessidade de recursos para ativação física, como acionamento por relés. A transmissão ao vivo dos resultados proporciona uma melhor experiência aos usuários com a verificação dos seus resultados na placa física.

1.3 Objetivos

1.3.1 Objetivos Gerais

Desenvolver um protótipo funcional de um laboratório remoto baseado em FPGA usando os recursos da placa de desenvolvimento Basys 3 da Digilent voltado para os estudantes das disciplinas PED 1 e PED 2 da FGA.

1.3.2 Objetivos Específicos

Os principais objetivos específicos requeridos são:

- Desenvolvimento de uma arquitetura de hardware, que consistirá em quatro blocos: (a) *Universal Asynchronous Receiver/Transmitter* (UART) com protocolo RS-232 para troca de dados entre o servidor e o FPGA; (b) Decodificador para representar as ações das chaves virtuais; (c) Decodificador para representar os LEDs e *displays* de sete segmentos; (d) Bloco base para ser implementada a lógica do usuário;
- Desenvolvimento de uma interface gráfica *web* com: (a) Um campo de *upload* para o envio do arquivo *bitstream* do projeto; (b) Chaves, botões, LEDs e *displays* de sete segmentos virtuais; (c) Escrita em um arquivo de formato ".txt" com os estados das saídas virtuais em função do tempo e com disponibilidade de *download*;
- Sistema de transmissão de vídeo *online* do kit de desenvolvimento a partir de uma *webcam*;
- Projeto estrutural simples para integração dos componentes: kit de desenvolvimento FPGA, *webcam* e conexão ao servidor.

1.4 Requisitos

Os requisitos do laboratório remoto para FPGAs almejados neste trabalho são:

- A arquitetura de hardware deve ser a mais simples possível, com pouco consumo de recursos da Basys 3;
- A arquitetura do servidor *web* deve ser prover todas as funcionalidades citadas como: (a) Campo de *upload* para o envio do arquivo bitstream do projeto; (b) Chaves, botões, LEDs e *displays* de sete segmentos virtuais; (c) Escrita em um arquivo de formato ".txt" com os estados das saídas virtuais em função do tempo com disponibilidade de *download*. Outro requisito é possuir fácil portabilidade para execução em outras máquinas;

- O servidor deve prover tempo limite de dez minutos de uso do laboratório remoto e não permitir múltiplos acessos com aviso indicando que está ocupado;

- A escrita no arquivo ".txt" das saídas virtuais deve ser feita em um período de dez segundos após o início do experimento indicando o tempo em que cada dado foi enviado, de forma que seja possível plotar os valores do LEDs e *displays* de 7 segmentos em função do tempo;

- O manual e tutorial de instruções devem ser didáticos, com exemplos de uso, regiões na arquitetura de hardware na qual o usuário deve e não deve modificar, interação com a interface e como fazer uso do *script* de plotagem dos gráficos;

- A estrutura deve ser compacta e com baixa luminosidade facilitando a captação do vídeo.

1.5 Aspectos Metodológicos

O presente trabalho foi dividido nas seguintes etapas:

- Iniciação: Definir o escopo e requisitos do projeto com a elaboração da justificativa, referencial teórico e objetivos do laboratório remoto;

- Validação do processo: Verificar e validar um processo crucial ao projeto: A implementação automática do arquivo *bitstream* no FPGA, via terminal do sistema operacional LINUX, sem necessidade da abertura de algum aplicativo, sem esse processo o projeto não é possível de ser realizado;

- Desenvolvimento da arquitetura de hardware: Criação da arquitetura de hardware do FPGA para o laboratório remoto, nessa etapa a metodologia de desenvolvimento *bottom-up* é usada, ou seja, criação dos módulos individuais tais como: bloco de transmissão de dados UART; decodificador das chaves e botões; decodificador dos LEDs e *displays* de sete segmentos e, posteriormente a integração dos mesmos em um *topmodule* no qual deverá ficar declarada e instanciada a lógica do usuário;

- Validação da arquitetura de hardware: Simulações comportamentais referente à arquitetura de hardware e validação com diversos testes entre um computador e o FPGA. Via terminal Linux são enviados diversos comandos a placa e verificação de seu comportamento e vice-versa;

- Desenvolvimento da interface gráfica: Pesquisa do *framework web* que melhor se adequa ao projeto, bem como a criação da processo *back-end* e *front-end* da interface gráfica. A metologia de uso é a *bottom-up*;

- Validação da interface gráfica: Semelhante a validação da arquitetura de hardware, porém com a substituição do terminal Linux pela interface gráfica, os botões virtuais são acionados para verificação do comportamento da placa e o FPGA envia os estados dos LEDs e *displays* de sete segmentos para verificação do comportamento da interface gráfica e a escrita no arquivo ".txt";
- Documentação: Criação do documento final do projeto e entrega dos produtos desenvolvidos.

1.6 Organização do Trabalho

Em termos estruturais a organização deste documento é da seguinte maneira:

O capítulo 2 apresenta todas as fundamentações teóricas relacionadas ao projeto, como a funcionalidade e uso dos laboratórios remotos, descrição de hardware reconfigurável, estrutura de um *framework web* para desenvolvimento da interface gráfica e o funcionamento das disciplinas de PED 1 e PED 2.

O capítulo 3 apresenta as implementações da arquitetura de hardware do servidor e da interface gráfica;

O capítulo 4 apresenta os resultados por meio de análises de consumo de síntese, potência, *timing*, *layout* e validação dos testes. Análise dos sinais de saída da Basys 3 no Matlab.

E por fim, o capítulo 5 com as conclusões referente ao projeto, sua viabilidade, limitações e trabalhos futuros.

2 Fundamentação Teórica

2.1 Laboratórios Remotos

Segundo (MA; JEFFREY; NICKERSON, 2006) laboratórios remotos são sistemas físicos que permitem que que experimentos reais sejam acessados de um lugar remoto e possibilitam uma maneira de compartilhar habilidades e recursos especializados capazes de reduzir custo e enriquecer experiências educacionais.

Para a devida fundamentação teórica deste projeto foi necessário a busca por outros laboratórios remotos criados na comunidade acadêmica, a seguir se relacionam alguns exemplos:

- 1 - Laboratório Remoto Para Prototipação de Circuitos Digitais Utilizando Kits de Desenvolvimento FPGAs XILINX: Trabalho bem semelhante ao proposto neste projeto, consiste no envio do arquivo *bitstream* do projeto já desenvolvido pelo usuário. A interação é feita via interface gráfica *web* com chaves virtuais e resultados das saídas do FPGA apresentados na tela. Possui transmissão ao vivo e a comunicação entre o servidor e a placa é feita via protocolo RS-232 (VICENTE, 2016);

- 2 - *Remote FPGA Lab with Interactive Control and Visualisation Interface*: Laboratório remoto para FPGA, na qual faz uso de múltiplos kits com suas respectivas *webcams*. O servidor *web* fornece a interface gráfica e e aloca uma sessão para cada usuário, permitindo o envio do projeto, visualização dos LEDs, *displays* de sete segmentos e interação com as chaves e botões. A comunicação entre o servidor e o FPGA é via protocolo RS-232 (MORGAN et al., 2011);

- 3 - RELLE - *Remote Labs Learning Environment*: Desenvolvido pelo grupo de trabalho em experimentação remota (GT-MRE) do laboratório de experimentação remota (RExLab), na Universidade Federal de Santa Catarina (UFSC) consiste em uma interface gráfica *web* com a seleção de vários tipos de laboratórios em diversas área como Física, Biologia e Robótica. Usa transmissão *web* ao vivo; componentes virtuais para interação do usuário; tempo limite de acesso ao experimento; manuais e tutorias de uso (GT-MRE, 2018). O site é acessado pelo *link*: <http://relle.ufsc.br/labs>;

- 4 - *Remote laboratories to support electrical and information engineering (EIE) laboratory access for students with disabilities*: Laboratório remoto para estudantes de Engenharia Elétrica que possui alguma deficiência física. Consiste em experiências para medidas de grandezas físicas de circuitos elétricos controlados remotamente por

uma interface gráfica *web*. A comunicação entre o servidor e as ferramentas do experimento é feita via protocolo RS-232 e dispõe de transmissão ao vivo (I.GROUT, 2014);

2.2 Hardware Reconfigurável

Hardwares reconfiguráveis são circuitos integrados baseados em lógica programável, ou seja, possuem operações internas que são definidas pelo usuário, o que torna possível o desenvolvimento rápido de um projeto de sistema digital. (GARCIA et al., 2006).

2.2.1 FPGAs

A *Field Programmable Gate Array* (FPGA) é um tipo de hardware reconfigurável composto por uma matriz de duas dimensões de lógicas programáveis, lógicas fixas, recursos de roteamentos implementados na tecnologia *Complementary Metal Oxide Semiconductor* (CMOS) e blocos para a interface de entradas ou saída ou *input/output* (I/0).

A figura 1 mostra a arquitetura de alto nível de um FPGA: O bloco *Programmable interconnect* possui interconexões programáveis por software e implementadas em hardware. O bloco de I/0 é constituído de circuitos para acesso de periféricos externos, como os LEDs, chaves, *displays* entre outros. (INSTRUMENTS, 2013)

Os blocos lógicos são formados por *Configurable Logic Blocks* (CLBs) e a figura 2 mostra a arquitetura desses blocos para as séries 7 da Xilinx, cada CLB é formado por 2 *slices* independentes, na qual *p*ossui: (XILINX, 2016):

- 4 *logic-function generator* ou *look-up tables* (LUTs) de 6 entradas 2 saídas para a formação da lógica booleana;
- 8 elementos de armazenamento da saída do LUT: *flip-flops* (FFs) do tipo D, sensíveis ao nível de subida ou descida do *clock*;
- Multiplexadores (MUX) de diversos propósitos;
- *Carry* lógico para a melhora na performance de funções aritméticas como somadores, subtratores e comparadores. Para funções mais complexas como: multiplicações e filtros, um slice possui um um bloco especializado para esse propósito, os *Digital Signal Processor* (DSPs).

Os itens listados são responsáveis pela implementação das funções lógicas, aritméticas e de *Read only memory* (ROM). Alguns slices possuem capacidade de armazenar dados

na memória *Random Access Memory* (RAM) e em registradores de 32 bits. A figura 3 mostra um exemplo de implementação de um multiplexador de 16:1 em uma slice, observe o uso das LUTs na entrada dos sinais, multiplexadores nas saídas da LUTs e as saídas armazenada em um FF ativo pela borda de *clock* .

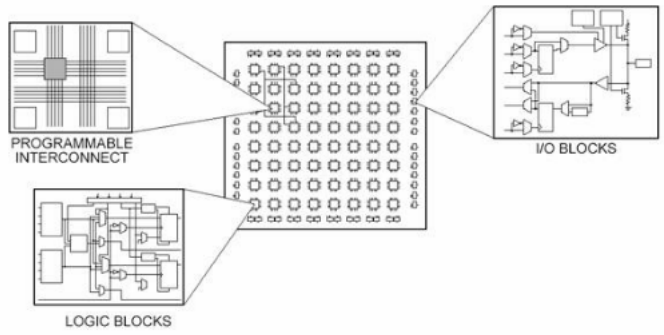


Figura 1 – Arquitetura de alto nível de um FPGA (INSTRUMENTS, 2013).

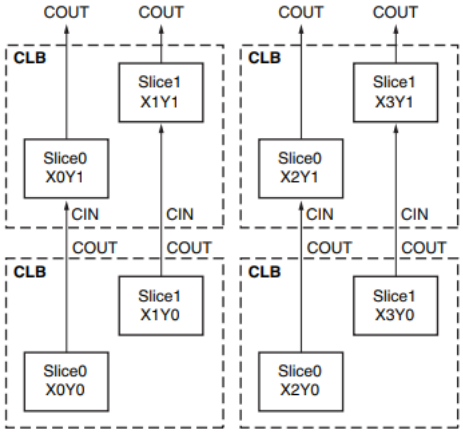


Figura 2 – Arquitetura de uma CLB (XILINX, 2016).

E para concluir o as funcionalidades básicas a tabela 2 enumera as vantagens e desvantagens no uso de um FPGA para desenvolvimento de projetos de sistemas embarcados.

Tabela 2 – Vantagens e desvantagens no uso do FPGA (GARCIA et al., 2006).

VANTAGENS:	Maior velocidade de processamento em relação ao software; prototipagem de circuitos integrados; redundância para sistemas críticos e paralelização de processos .
DESVANTAGENS:	Alto custo de aquisição; alto consumo de potência para funcionamento.

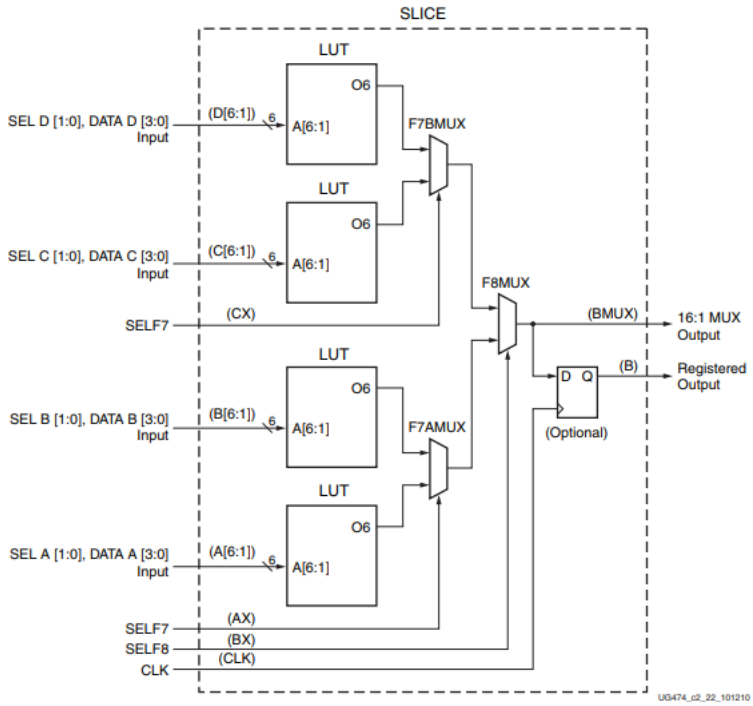


Figura 3 – Implementação de um MUX 16:1 em uma *slice* (XILINX, 2016).

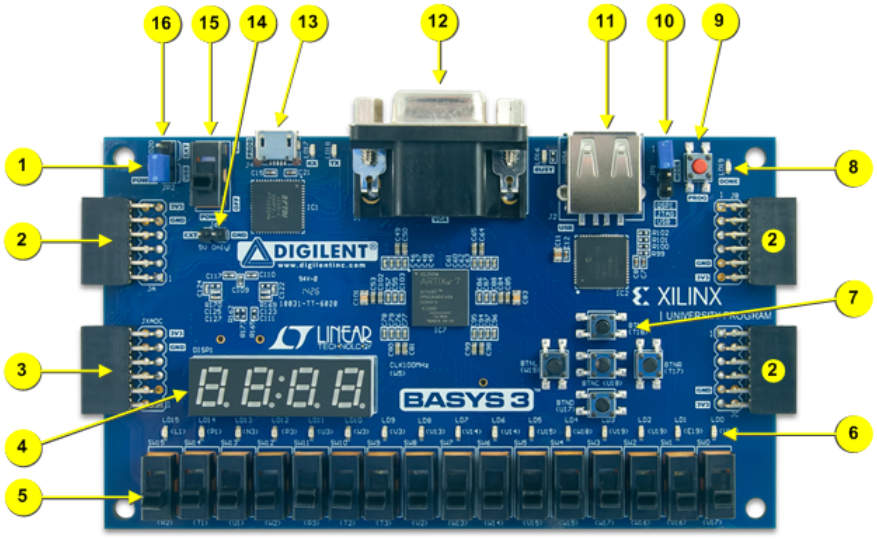
2.2.2 Arquitetura Artix-7 - XC7A35T

O kit Basys 3 utiliza um FPGA da família Artix-7 (dispositivo XC7A35T) da Xilinx. A arquitetura desse modelo utiliza 33200 células lógicas; 5200 *slices*, sendo que cada *slice* possui 4 LUTs e 8 FFs com 400 *Kilobits* (Kb) de memória distribuída ; 90 DSP48E1 *Slices* com 25x18 multiplicadores, um somador e um acumulador; memória RAM de 1800 Kb distribuídos em 100 blocos de 18 Kb ou 50 de 36 Kb; 5 *Mixed-Mode Clock Manager* MMCM para geração de diferentes sinais de *clock* e 250 pinos de I/0 (XILINX, 2018).

2.2.3 Kit de Desenvolvimento Basys 3

A figura 4 mostra a disposição dos componentes da Basys 3 com circuito integrado do FPGA ARTIX-7 no centro do Kit. Para a disciplina de PED 1 e 2, os essenciais são: 1 - LED indicativo da placa ligada ligada; 4-*Display* de sete segmento; 5 - Chaves, 6 - LEDs; 7 - Botões; 8 - Led indicativo de FPGA programada; 9 - Botão de *reset* para configuração inicias do FPGA; 13 - Porta USB para alimentação/comunicação UART/programação pela *Joint Test Access Group* (JTAG) e 15 - Chave de liga e desliga (DIGILENT, 2016).

A figura 5 detalha as conexões dos periféricos, como chaves,botões, LEDs e *display* de sete segmentos, fator importante para o bom uso do kit da Basys 3. As resistências mostradas possuem valor de 330 Ω.



Callout	Component Description	Callout	Component Description
1	Power good LED	9	FPGA configuration reset button
2	Pmod connector(s)	10	Programming mode jumper
3	Analog signal Pmod connector (XADC)	11	USB host connector
4	Four digit 7-segment display	12	VGA connector
5	Slide switches (16)	13	Shared UART/ JTAG USB port
6	LEDs (16)	14	External power connector
7	Pushbuttons (5)	15	Power Switch
8	FPGA programming done LED	16	Power Select Jumper

Figura 4 – Caracterização do kit de desenvolvimento Basys 3 (DIGILENT, 2016).

2.3 Prática de Eletrônica Digital 1 e 2

As disciplinas PED 1 e PED 2 são lecionadas na Faculdade UnB Gama em conjunto com as disciplinas de Teoria de Eletrônica Digital 1 e 2 e são detalhadas nas seções a seguir. Ambas as turmas usam o mesmo kit de desenvolvimento Basys 3.

2.3.1 Prática de Eletrônica Digital 1

A disciplina é composta por seis turmas de até quarenta alunos cada, com aulas de duração de duas horas uma vez na semana, a sua ementa é composta por (MATRICULAWEB, 2018):

- Sistemas de numeração e códigos;
- Portas lógicas e álgebra booleana;
- Circuitos lógicos combinacionais;
- *Very High Speed Integrated Circuits Hardware Description Language* (VHDL);

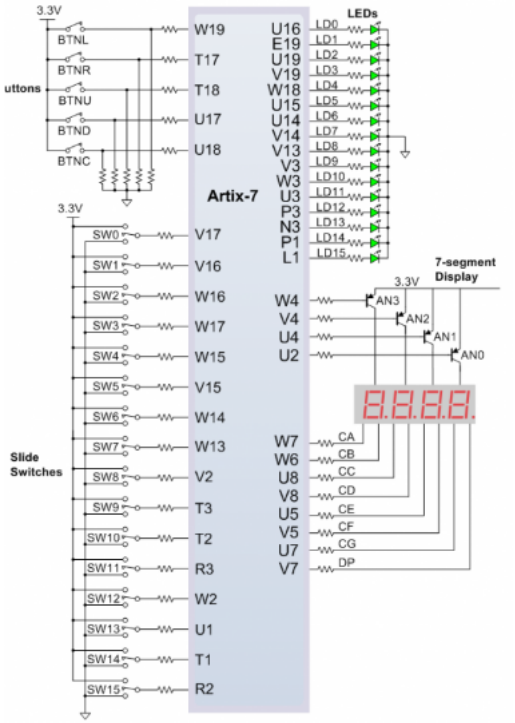


Figura 5 – Conexão das chaves,botões, LEDs e *display* de sete segmentos na Artix-7 (DIGILENT, 2016).

- Aritmética digital: Operações e circuitos;
- Circuitos lógicos *Medium-Scale Integration* (MSI);
- Princípios de sistemas sequenciais.

2.3.2 Prática de Eletrônica Digital 2

A disciplina é composta por duas turmas de até quarenta alunos cada, com aulas de duração de duas horas uma vez na semana, a sua ementa é composta por (MATRICULAWEB, 2018):

- Sistemas de numeração e códigos;
- Descrição de circuitos sequenciais em VHDL;
- Descrição de FSMs em VHDL;
- Memórias;
- Projeto em nível de transferência de registradores ou *Register Transfer Level* (RTL);
- Arquitetura de microprocessadores.

2.4 *Framework Web* - Flask

Framework é um aplicativo semi-completo que pode pode ser utilizado para produzir aplicações personalizadas, baseado no paradigma de software de orientação a objeto, resulta na modularidade, reusabilidade e extensibilidade, ou seja, esconde detalhes de implementações complicadas por trás de interfaces estáveis, essas interfaces são definidas como componentes genéricos para uso do desenvolvedor (FAYAD; SCHMIDT, 1997). O *mini Framework Web* FLASK, possui um núcleo robusto com as funcionalidades básicas que aplicativos *Web* necessitam, é possível e frequente uso de pacotes adicionais. Escrito na linguagem de alto nível Python, é compatível com *framework front-end* Bootstrap, aplicação com a linguagem de marcação de *HyperText Markup Language* (HTML) para a estruturação da página *web*, a linguagem *Cascading Style Sheets* (CSS) para estilizar os elementos da interface, e a linguagem Javascript para definir ações nos elementos criados (GRINBERG, 2014).

Baseado na arquitetura *Model-View-Controller* (MVC) mostrada na figura 6, na qual divide uma aplicação nessas três partes interconectadas, sendo os módulos *Controller* e *Model* correspondente ao *Back end* e a *View* ao *front end*. Basicamente, o módulo *Controller* é responsável pelo envio de comandos ao *Model* que por sua vez o armazena em um banco de dados e notifica os módulos sobre uma possível ação para atualizar seu estados. A *view* é responsável pela a apresentação dos resultados de acordo com os estados dos dois blocos anteriores, e está intimamente ligada ao *framework bootstrap* (GUPTA, 2010).

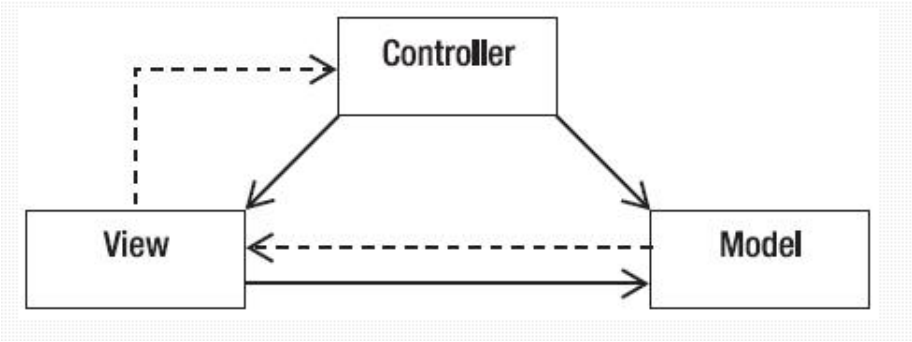


Figura 6 – Arquitetura MVC do *framework web* Flask.

3 Implementação

3.1 Arquitetura de Hardware Embarcada

Para o desenvolvimento da arquitetura de hardware na Basys 3 foi utilizado o pacote de software Vivado Design Suite 2016.4 e a linguagem VHDL (SASS; SCHMIDT, 2010). No processo de implementação foi utilizada a metologia *bottom-up*, na qual cada bloco é desenvolvido individualmente e, posteriormente, adicionados em uma arquitetura geral chamada de ***top_wrapper***, que faz alusão à invólucro, mostrada na figura 7. Cada módulo interno é detalhado nas seções posteriores. Todos os blocos são fixos, com exceção do ***user***, que é destinado para a lógica do projeto do usuário. O arquivo *constraints* para uso das entradas e saídas da Basys 3 também é fixo, essa escolha faz com que o uso do laboratório remoto seja simplificado e o usuário desenvolva apenas o código VHDL. Os arquivos fontes se encontram no *link*: <https://gitlab.com/damuz/labremotofpga>.

Neste projeto o *clock* de uso é do sistema interno da placa Basys 3 com frequência de 100 Megahertz (MHz), doravante chamado de *clk*, que é comum a todos os blocos internos a fim de sincronizar os processos na borda de subida. Outro sinal comum é o *reset*, ativo em nível lógico alto, que coloca a arquitetura nas condições iniciais de processamento.

O módulo ***serialcom*** é responsável pela transmissão dos dados ao servidor através do sinal *tx* e recebimento pelo sinal *rx*, esses dados são uma palavra binária representando um caractere no formato *American Standard Code for Information Interchange* (ASCII (INJOSOFTAB, 2018)). O sinal *rx* que é responsável por enviar os dados simulados dos botões e chaves na interface gráfica é recebido e armazenado no sinal *data_out*. Esse sinal é decodificado no módulo ***deco_sw_bts*** e, posteriormente, armazenados nos sinais de saídas *sw* e *buttons*, representando os dados das chaves e botões, respectivamente. Após esse processamento o módulo ***user*** recebe essas palavras para serem usadas de acordo com o projeto desenvolvido pelo usuário.

Todas as saídas do ***user*** são processadas no ***deco_leds_7seg*** que é responsável pelo envio ao servidor dos dados do registrador *data_in*. Os sinais de saídas *led*, *an*, *seg* e *dp*, representam os dados dos LEDs, anodos, *displays* e ponto decimal do sete segmentos, respectivamente. Esses dados são apresentados nas saídas físicas da Basys 3, nas saídas virtuais da interface gráfica e, posteriormente, armazenados em um arquivo ".txt".

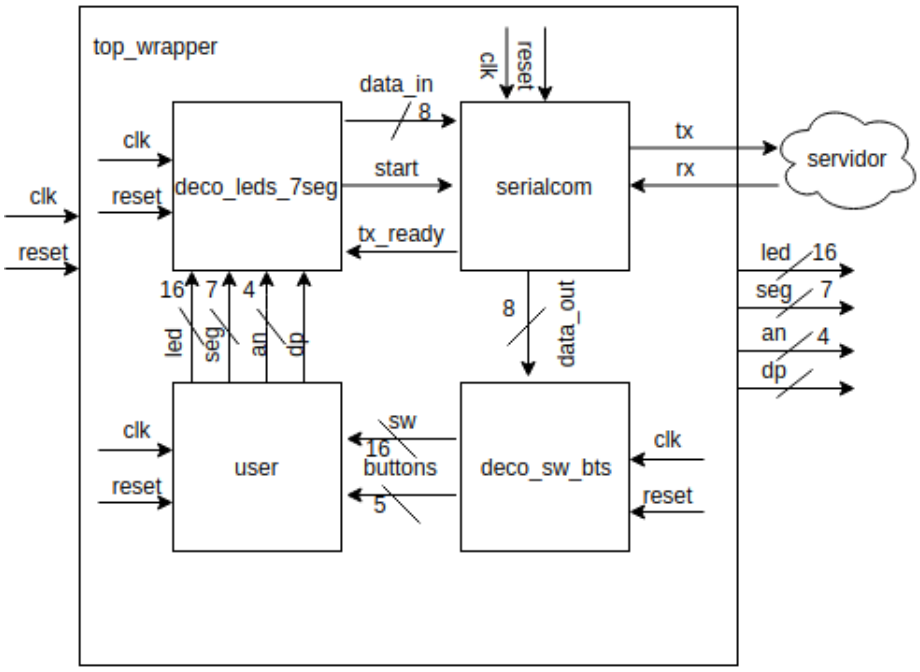


Figura 7 – Arquitetura geral de hardware desenvolvida na Basys 3.

3.1.1 Módulo *serialcom*

Para a comunicação serial no módulo ***serialcom*** foi modificado um código VHDL cedido pelo Prof. Dr. Daniel Maurício Muñoz Arboleda. A função é implementar um protocolo UART ou RS-232 com *bit rate* de 9600 sem paridade, a figura 8 mostra sua estrutura interna.

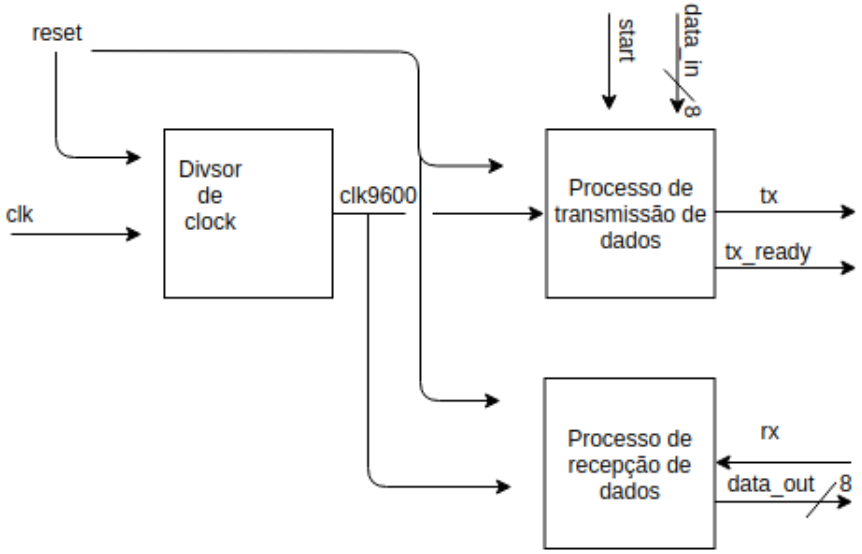


Figura 8 – Arquitetura do bloco *serialcom*.

Observe-se que que há um módulo divisor de *clock* que transforma a frequência interna para a frequência do UART. Há dois processos independentes, sincronizados pela

borda de subida do sinal *clk9600*, que são responsáveis pela transmissão e recepção de dados. O primeiro recebe como entrada o sinal *start*, quando o estado é nível alto, inicia-se a transmissão no sinal *tx* recebidos pelo *data_in* de 8 *bits*. Após concluído o processo, *tx_ready* indica o término com o sinal ativo em nível alto.

O segundo processo é o de recepção, no qual o sinal *rx* recebe serialmente os dados externos e os paraleliza em um sinal de 8 *bits* chamado de *data_out*.

3.1.2 Módulo *deco_leds_7seg*

A figura 9 mostra a estrutura interna do módulo ***deco_leds_7seg***, composto por um contador e um MUX. Sua base de funcionamento parte do fato de que existem quatro entradas, ou saídas do ***top_wrapper***, os sinais *led_lsb* e *led_msb*. Estes representam os estados dos 8 *bits* menos e mais significativos, respectivamente, oriundos do sinal *led* do módulo ***user***, esse partilhamento em dois foi necessário, pois o sinal de saída *data_out* possui 8 *bits* e o sinal *led* possui 16. O sinal de *seg&dp* é o estado do *display* de sete segmentos concatenado com o estado do ponto decimal. A entrada *an&1111* indica se os *displays* estão ligados ou desligados, como há apenas quatro deles na Basys 3 é necessário concatenar com a palavra binária "1111" para completar os 8 *bits* da palavra de transmissão.

Para indicar essas entradas no *data_in* é necessário multiplexá-las em um tempo determinado. Para isso é usado um contador, cujo valor de *preset* foi configurado em 1.5 milissegundos (ms), na qual o valor é baseado na taxa de transmissão do módulo ***serialcom***. Quando o contador chega no tempo estabelecido o sinal *s_mux* é somado em 1 para a alternância na saída do multiplexador. A figura 10 mostra a estrutura de dados dos sinais de saída, na qual cada dado possui um identificador representado pelos sinais com prefixo "id", esses sinais são necessários para indicar qual dado será enviado ao servidor na contagem seguinte. Portanto os valores dos sinais de saída Basys 3 são enviados a cada 2 contagens (3 milissegundos) e para um determinado estado de saída ser atualizado são necessários 8 contagens (12 milissegundos).

A entrada *tx_ready* e a saída *start* são para controle da transmissão e são usados para evitar perda de dados, ou seja, quando o módulo ***serialcom*** está ocupado com *tx_ready* em nível alto, o sinal *start* está em nível baixo para não tentar iniciar uma transmissão.

3.1.3 Módulo *deco_sw_bts*

A arquitetura do módulo ***deco_sw_bts***, mostrada na figura 11, faz uso de um decodificador síncrono. Ele recebe a entrada *data_out* do módulo ***serialcom*** e indica os estados das saídas *sw* e *buttons*, para isso cada palavra binária pré-determinada indica qual

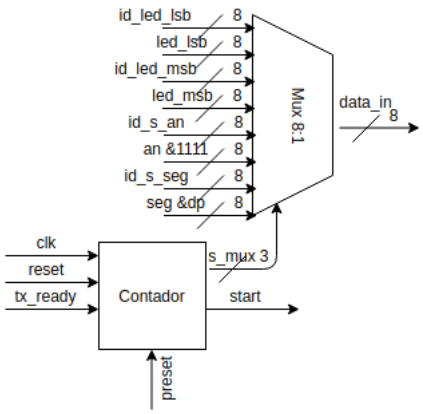


Figura 9 – Arquitetura do módulo *deco_leds_7seg*.

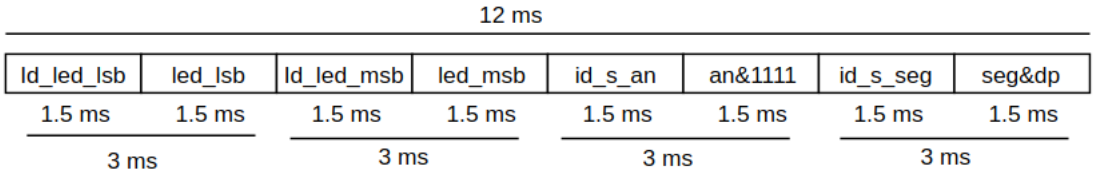


Figura 10 – Estrutura da transmissão de dados dos sinais de saída da Basys 3.

chave ou botão é acionado e seu respectivo estado lógico. Vale destacar que a arquitetura de hardware do laboratório remoto não limita a criação de novos botões e chaves, portanto é possível a criação desses componentes além da quantidade que existe na placa física.

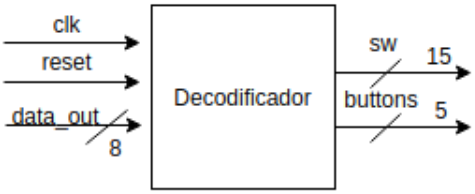


Figura 11 – Arquitetura do módulo *deco_sw_bts*.

3.1.4 Módulo *user*

A entidade do módulo ***user*** é mostrada na figura 12. Ele recebe os valores de entrada de *sw* e *buttons* provenientes das chaves e botões virtuais e a partir do processamento feito pela arquitetura do usuário, modificam-se as saídas *led*, *seg*, *an* e *dp* para serem apresentados nas unidades de saídas da Basys 3 e serem decodificadas, enviadas para a interface gráfica e armazenadas no arquivo ".txt".

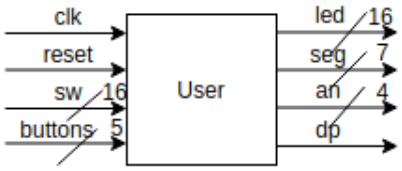


Figura 12 – Arquitetura do módulo *user*.

3.2 Servidor e Estrutura da Interface Gráfica

A rede da aplicação é baseada no modelo cliente-servidor (TANENBAUM, 2003), a figura 13 mostra a arquitetura do laboratório remoto para essa estrutura. Todas as tarefas são fornecidas por um servidor, e o cliente as chama com requisições (por URLs - *Uniform Resource Locator*) por meio da interface *web*. A comunicação entre os dois lados é baseada no protocolo TCP/IP de uma rede de *internet*. O servidor executa as tarefas em função dos dados trocados entre a Basys 3 e as imagens captadas da *webcam*. As seções seguintes detalham como o lados servidor/cliente são desenvolvidos com auxílio do *framework* Flask (GRINBERG, 2014).

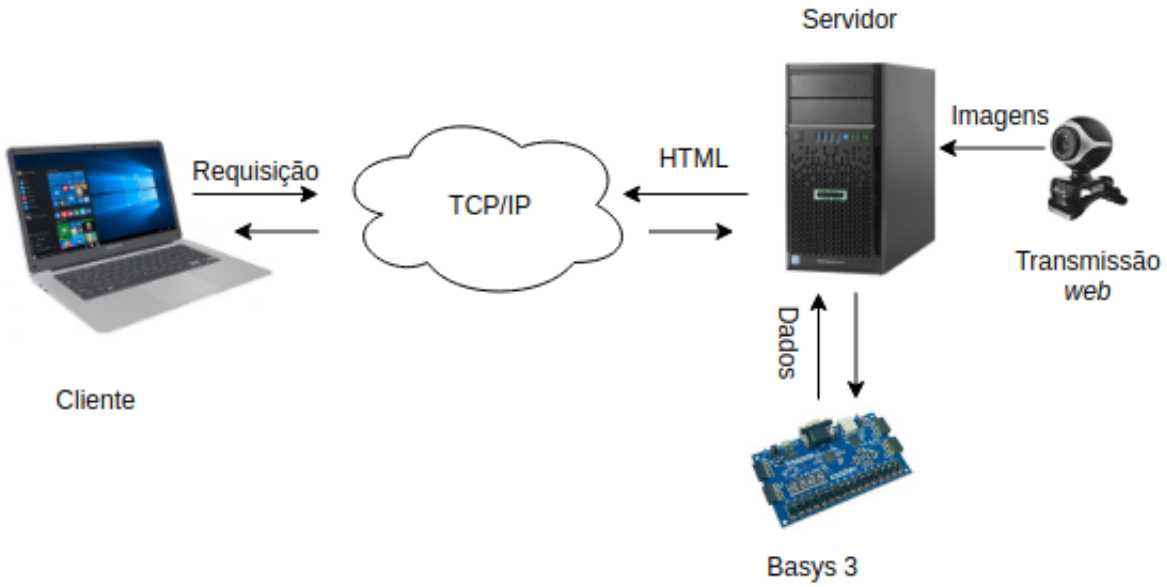


Figura 13 – Modelo cliente-servidor do laboratório remoto. Imagens meramente ilustrativas.

3.2.1 Servidor

Para a implementação do modelo nos lados cliente-servidor, toda a aplicação é separada em uma máquina virtual com auxílio de um simulador de ambientes chamado de Virtualenv (BICKING, 2018), isso é necessário para não haver incompatibilidade de

bibliotecas Python usadas no projeto com as bibliotecas instaladas no computador, além de facilitar a portabilidade para outras máquinas.

De acordo com a estrutura MVC do Flask, o lado servidor é construído a partir dos módulos *Controllers* e *Models*, porém como não há banco de dados no modelo do laboratório remoto, o último é abstraído. As tarefas designadas para o módulo *Controllers* são:

- Comunicação com a Basys 3: É estabelecida a comunicação RS-232 com a Basys 3 a partir do módulo pySerial (LIECHTI, 2014), na qual fornece diversas funções para acesso a porta serial, como leitura e escrita de dados.
- *Upload* e implementação da arquivo *bitstream* na Basys 3: O Flask se encarrega do recebimento do arquivo, enquanto o Adept (DIGILENT, 2005), pacote da Digilent que faz comunicação entre o sistema operacional e a placa Basys 3 sem necessidade de abrir uma aplicação, implementa o arquivo de configuração no FPGA;
- Transmissão ao vivo da *webcam* na interface gráfica: Faz uso do OpenCV (OPENCV-TEAM, 2018), pacote de multiplataforma para aplicações de visão computacional. No laboratório remoto é usado o módulo de vídeo I/O em Python em conjunto com o Flask;
- Tempo limite de acesso de 10 minutos, bloqueio de múltiplos acessos: Desenvolvido no ambiente Flask;
- Escrita no arquivo ".txt" e disponibilidade de *download*: Desenvolvido no ambiente Flask, na qual o tempo de gravação no arquivo é de 10 segundos após a implementação do *bitstream* no FPGA. O servidor escreve os valores dos sinais do módulo ***deco_leds_7seg***, com seu respectivo tempo de transmissão no arquivo ".txt". A figura 14 mostra a estrutura desse arquivo, na qual o ciclo de escrita se repete necessariamente nessa ordem e o número de amostras varia em torno de 3333.

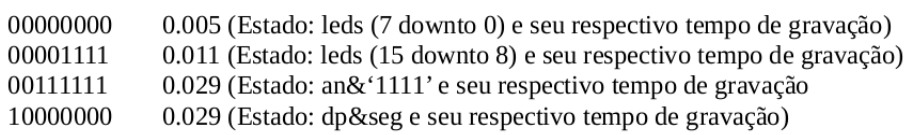


Figura 14 – Estrutura do arquivo ".txt", com estados dos LEDs, anodos e *displays* de sete segmentos com seus respectivos tempo de escrita.

Para a comunicação entre o servidor/cliente no protocolo TCP/IP é necessário atribuir uma porta (para transferir HTML são usadas portas de começo 80 (IANA, 2018)) e um endereço IP à aplicação do servidor para que o usuário consiga achá-lo na rede e fazer as requisições necessárias.

3.2.2 Interface gráfica

Toda a estrutura das páginas *web*, bem como a parte estética são desenvolvidas nesse módulo, com uso do HTML, CSS e Javascript.

A interface gráfica é desenvolvida no bloco *View*, que contém uma pasta *Static* com todos os arquivos estáticos da aplicação, na qual possui o Bootstrap. Uma outra pasta contém os *templates* que são páginas HTML de retorno quando um usuário faz uma requisição. Como por exemplo, quando um usuário aperta o botão, uma URL é chamada e a tarefa do servidor é executada, retornando uma ação, ou uma página HTML.

Algumas ações importantes são desenvolvidas com auxílio do Javascript: Não atualização da página HTML quando uma variável muda de valor ou um botão é acionado e a não permissão de *upload* de arquivos com nome diferentes de *top_wrapper.bit*, esse fato é necessário para a implementação correta do arquivo *bitstream* na Basys 3.

As figura 15 mostra a página inicial *web* no laboratório remoto. O botão *Contato* redireciona para a URL com a página da figura 16. O botão *Faça o Upload* abre a página da figura 17. O botão *Manual* retorna um arquivo de formato "pdf" com o manual do laboratório remoto que se encontra no *link*: <https://gitlab.com/damuz/labremotofpga>. Caso o usuário use o botão *Início* ele retorna à página inicial.

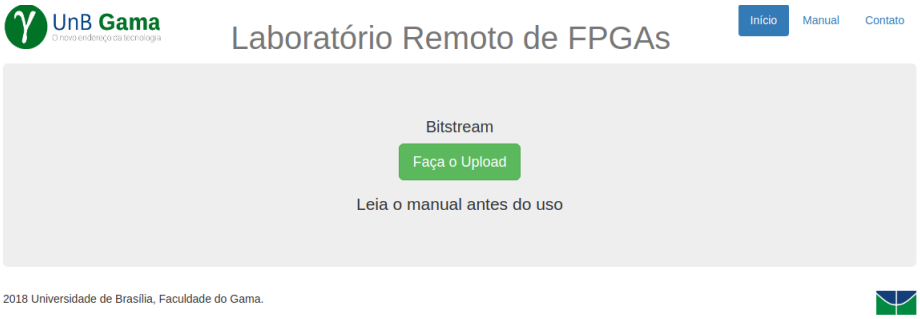


Figura 15 – Página *web* inicial do laboratório remoto.

Figura 16 – Página *web* para contatos do laboratório remoto.

A página da figura 17 fornece ao usuário a opção de *upload* do arquivo *bitstream*, caso essa ação seja efetuada com sucesso ela retorna a página da figura 18 que possui as principais funcionalidades do laboratório remoto: a transmissão *web*; o botão de *download*

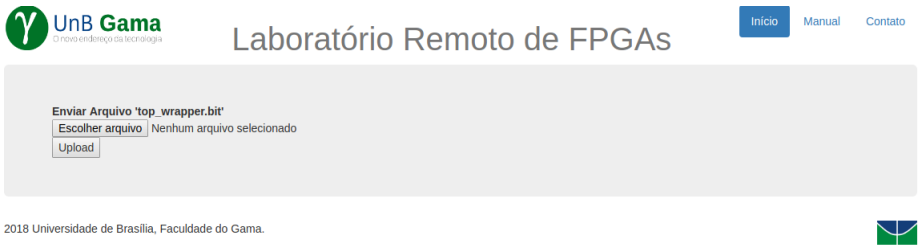


Figura 17 – Página *web* para *upload* do arquivo *bitstream*.

para acesso ao arquivo dos estados das saídas da Basys 3 no formato ".txt" ; chaves e botões virtuais, em azul e verde, respectivamente.

A figura 19 mostra o aviso caso haja acessos múltiplos ao servidor. Quando o tempo limite de usuário se esgota ele tem o retorno à página inicial para desocupar o laboratório.

Figura 18 – Página *web* contendo as principais funcionalidades do laboratório remoto.

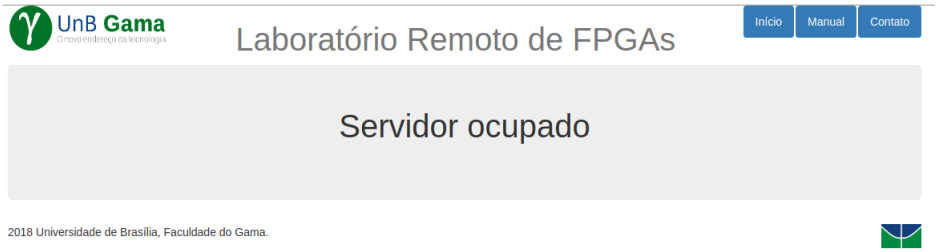


Figura 19 – Página *web* com aviso de ocupado caso haja acesso múltiplos ao servidor.

4 Resultados

A fim de caracterizar a solução proposta está seção apresenta as análises de consumo de recursos, estimativa do consumo de potência, análise de *timing*, *layout* e validação dos testes do laboratório remoto. A caracterização de circuitos de uso nas disciplinas na PED 1 e PED 2, como um contador de 1 segundo sequencial e uma máquina de estado detector de sequência, também são detalhados. Na última seção foi feita uma análise dos estados das saídas LEDs e displays de 7 segmentos a partir do arquivo ".txt"

4.1 Caracterização da Arquitetura Proposta ou Exemplo 1: Leitura das Chaves e Escrita nos LEDs

Para caraterizar os recursos que a arquitetura de hardware do laboratório consome e o espaço do FPGA destinado ao usuário, foi criado um exemplo mais simples possível no bloco ***user***, referente à leitura das chaves e escrita nos LEDs da Basys 3.

A tabela 3 mostra o consumo dos recursos da Basys 3. Não há utilização de LUTs de memória, blocos RAM, DSPs nem *latch* e o consumo de LUTs lógico e FFs é menor que 1%, fator que vai de encontro ao requisito de pouco uso do recursos do FPGA. Os 30% dos pinos de I/O serão sempre constantes, pois como o arquivo *constraint* é fixo, sempre haverá uso dos LEDs, anodos, os 4 *displays* de sete segmentos e os sinais de transmissão e recepção de dados do protocolo serial RS-232. Vale destacar que as chaves e botões são virtuais e não consomem pinos I/0.

Tabela 3 – Consumo de recursos dos componentes essenciais para o funcionamento da arquitetura de hardware proposta na Basys 3.

Recurso	**Consumo**	**Disponível**	**Consumo %**
LUT lógico	118	20800	0.56
LUT de memória	0	9600	0.00
Registrador como FF	103	41600	0.25
Registrador como *Latch*	0	41600	0.00
Blocos RAM	0	50	0.00
DSPs	0	90	0.00
IO	32	106	30.19

A figura 20 mostra a *layout* do exemplo 1 na Basys 3, o ***serialcom*** é representado pela cor azul, o ***deco_leds_seg*** em vermelho e o ***deco_sw_bts*** em verde, como o módulo ***user*** é muito pequeno ele é abstraído da imagem.

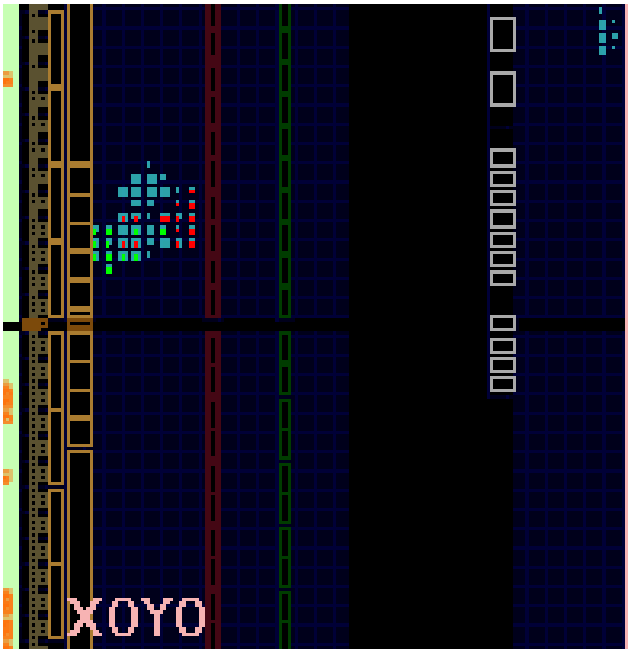


Figura 20 – *Layout* na Basys 3 dos componentes essenciais para o funcionamento da arquitetura de hardware proposta.

A figura 21 detalha o consumo de potência da arquitetura de hardware proposta do exemplo 1 para uma temperatura ambiente de 25.3° C, apesar de ser uma estimativa de baixa confiabilidade, pelo fato da temperatura sempre ter uma variação, ela mostra uma ideia de quanto de potência o circuito consome. Com um uso total de 0.007 W, sendo que 0.068 W cerca de 98%, apenas para manter a Basys 3 com os circuitos em funcionamento. Os outros 2% são dissipados para potência dinâmica como *clocks*, com uso de 57% dos 0.002 W, *signals* com uso de 16%, *logic* com 22% e o restante para os pinos de I/O.

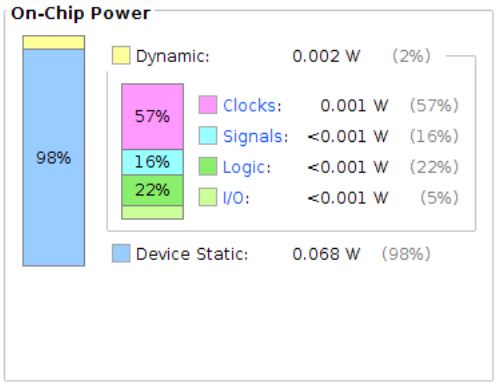


Figura 21 – Consumo de potência do componentes essenciais para o funcionamento da arquitetura de hardware proposta na Basys 3.

A tabela 4 mostra a análise de *timing* para os máximos *delays* de *setup* e de *hold*, o *clock* máximo de operação do exemplo 1 é o inverso do pior caso, o *delay* de *setup*.

Como não há restrições de *timing* e o *clock* de uso é de 100 MHz, o projeto funciona perfeitamente.

Tabela 4 – Análise de *timing* dos componentes essenciais para o funcionamento da arquitetura de hardware proposta na Basys 3.

***Delay* máximo de setup**:	5.822 ns
Delay* máximo de *hold:	0.117 ns
***Clock* máximo de operação**:	171 MHz

A figura 22 mostra o caminho crítico de *setup*, na qual a fonte é o registrador[15] e o destino o registrador [20] do contador do ***deco_led_7seg***.

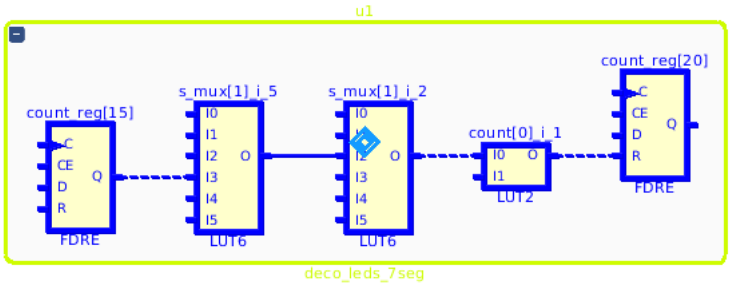


Figura 22 – Caminho crítico de *setup* dos componentes essenciais para o funcionamento da arquitetura de hardware proposta na Basys 3.

4.2 Exemplo 2: Contador de 1 Segundo

O exemplo sequencial é um contador de um segundo, na qual os resultados incrementados da contagem são mostrados em 2 *displays* de sete segmentos e nos 4 *bits* menos significativo dos primeiros 8 LEDs. O resultado decrementado é apresentado nos 4 *bits* menos significativos dos últimos 8 LEDs. Um vídeo tutorial desse exemplo para o laboratório remoto se encontra no *link*: <https://gitlab.com/damuz/labremotofpga>

A tabela 5 mostra o consumo dos recursos da Basys 3, na qual se observa um aumento significativo de FFs pela adição da lógica sequencial, uma diminuição de LUTs pelo fato do ferramenta de síntese otimizar o uso desses elementos em relação ao exemplo 1. Não há uso de LUT de memória, *latchs*, blocos de memória e nem DSPs. Como esperado, o consumo de I/O é o mesmo do exemplo anterior.

A figura 23 mostra o *layout* do exemplo na Basys 3, observe que agora há a adição, em amarelo, do módulo ***user***.

A figura 24 mostra que há um aumento do consumo de potência, com um total de 0.093 W, e o uso da potência dinâmica tem um aumento em relação à análise anterior, pelo fato do estado do *display* está sempre em mudança pela contagem, por isso 87% são

Tabela 5 – Consumo dos recursos da Basys 3 para o contador de 1 segundo.

Recurso	**Consumo**	**Disponível**	**Consumo %**
LUT lógico	93	20800	0.45
LUT de memória	0	9600	0.00
Registrador como FF	120	41600	0.29
Registrador como *Latch*	0	41600	0.00
Blocos RAM	0	50	0.00
DSPs	0	90	0.00
I/O	32	106	30.19

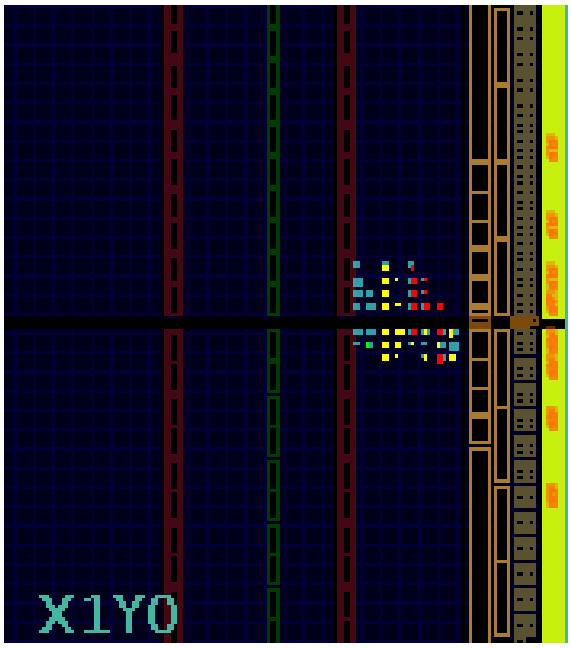


Figura 23 – *Layout* na Basys 3 do contador de 1 segundo.

dissipados para os I/0. Ainda assim grande parte da potência de consumo é para manter os circuitos da Basys 3 em funcionamento, cerca de 0.072W.

A tabela 6 mostra a análise de *timing*, demonstrando que não foram evidenciados restrições de *setup* e *hold* e, dado que *clock* máximo de operação é maior que 100 MHz, o circuito funciona perfeitamente.

Tabela 6 – Análise de *timing* do circuito contador de 1 segundo.

Delay* máximo de *setup:	4.829 ns
Delay* máximo de *hold:	0.178 ns
***Clock* máximo de operação**:	207 MHz

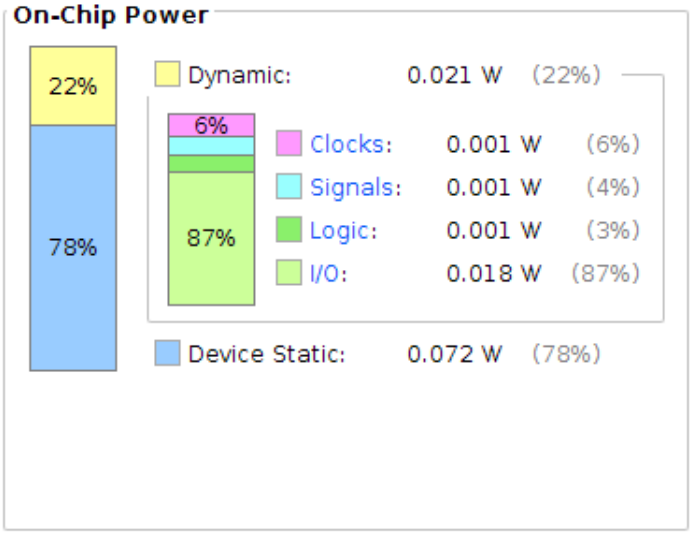


Figura 24 – Consumo de potência do circuito contador de 1 segundo.

4.3 Exemplo 3: FSM detector de sequência

O detector de sequência é uma máquina de estado que detecta a sequência "110" de uma entrada qualquer e acende um LED caso a sequência esteja correta. O exemplo usado é de uma das aulas de Teoria de Eletrônica Digital 2 lecionada pelo Prof. Dr. Gilmar Silva Beserra.

A tabela 7 mostra o consumo dos recursos da Basys 3. É possível observar que há diminuição de LUTs e FFs, pelo fato da ferramenta de síntese otimizar o uso desses elementos em relação ao exemplo 1. Não há uso de LUT de memória, *latchs*, blocos de memória e nem DSPs. Como esperado, o consumo de I/O é o mesmo do exemplo 1.

Tabela 7 – Consumo dos recursos da Basys 3 para a FSM detector de sequência.

Recurso	**Consumo**	**Disponível**	**Consumo %**
LUT lógico	79	20800	0.38
LUT de memória	0	9600	0.00
Registrador como FF	91	41600	0.22
Registrador como *Latch*	0	41600	0.00
Blocos RAM	0	50	0.00
DSPs	0	90	0.00
I/O	32	106	30.19

A figura 25 mostra o *layout* do exemplo na Basys 3.

A figura 26 mostra que há um aumento do consumo de potência, em relação ao exemplo 1, com um total de 0.074 W. A potência dinâmica, cerca de 0.002, é dissipada em grande parte para *clock* e *signals*. Ainda assim grande parte da potência de consumo é pra manter os circuitos da Basys 3 em funcionamento, cerca de 0.072W.

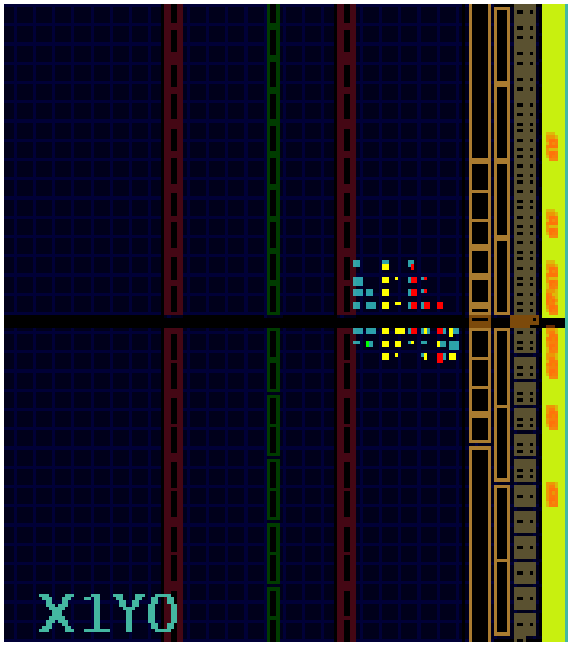


Figura 25 – *Layout* na Basys 3 para a FSM detector de sequência.

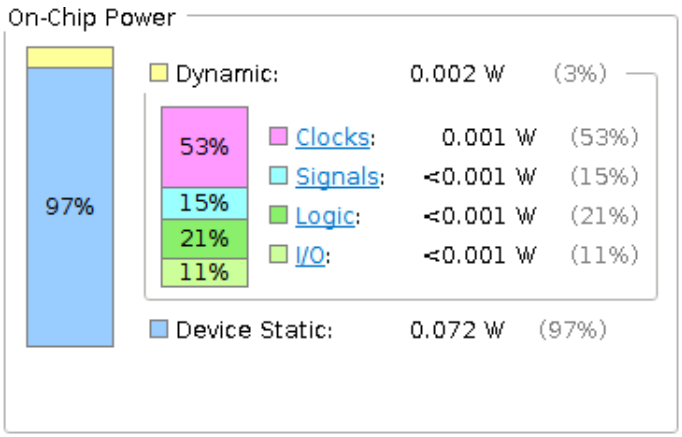


Figura 26 – Consumo de potência para a FSM detector de sequência.

A tabela 8 mostra a análise de *timing*, demonstrando que não foram evidenciados restrições de *setup* e *hold* e, dado que o *clock* máximo de operação é maior que 100 MHz, o circuito funciona perfeitamente.

Tabela 8 – Análise de *timing* para a FSM detector de sequência.

Delay* máximo de *setup:	5.811 ns
Delay* máximo de *hold:	0.178 ns
***Clock* máximo de operação**:	172 MHz

4.4 Análise dos estados das saídas dos LEDs e *Displays* de 7 segmentos a partir do arquivo ".txt"

Para caracterizar as saídas dos LEDs e *displays* de 7 segmentos da Basys 3 a partir do arquivo ".txt", foi criado um *script* no Matlab (MATHWORKS, 2018) para plotar os estados desses sinais em função do tempo, definido até 10 segundos, o arquivo se encontra no link: <https://gitlab.com/damuz/labremotofpga>.

O caso de teste foi o contador de 1 segundo do exemplo 2.

O gráfico à esquerda da figura 27 mostra o estado, em decimal, dos 8 *bits* menos significativos dos LEDs da Basys 3 em função do tempo, dado em segundos. O da direita corresponde aos 8 *bits* mais significativos. Pode-se observar no primeiro gráfico que a contagem está incrementando o valor do sinal até 10, porém esse incremento não ocorre de 1 em 1 segundo, esse fato é explicado pelo fato de haver atraso na transmissão e o sistema operacional do servidor ser um não determinístico e, portanto a escrita no arquivo não ocorrer no mesmo momento em que o dado é recebido. O mesmo ocorre para o gráfico à direita que está decrementado o valor de 15 até 0, como definido no exemplo. Apesar dessa limitação os gráficos retratam bem o comportamento dos sinais para o contador de 1 segundo.

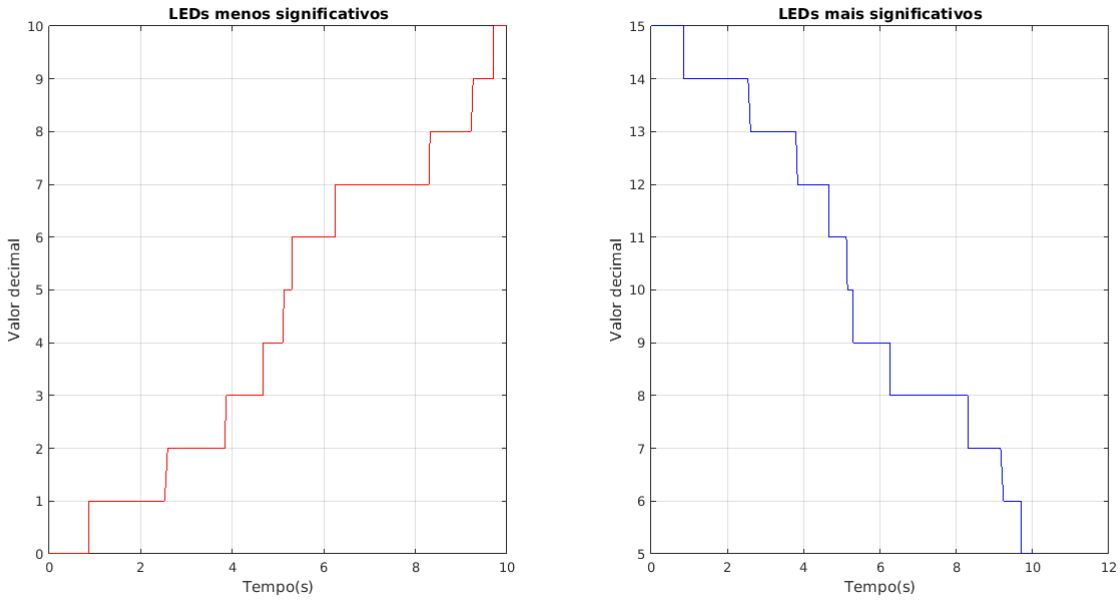


Figura 27 – Gráficos dos estados dos LEDs mais e menos significativos da Basys 3 em função do tempo.

figura 28 mostra os gráficos dos estados, em decimal, dos 4 *display* de segmentos da Basys 3 em função do tempo, dado em segundos. No *script* foi definido que um *display* desligado corresponde a -1 no gráfico, e um valor desconhecido como -2, como por exemplo

a disposição dos sete segmentos não formar nenhum caractere hexadecimal conhecido. A análise do gráfico retrata bem, apesar da limitação citada anteriormente, o comportamento destes elementos, com o *display* 1 e 2 desligados e os *displays* 3 e 4 variando o valor de 0 a 10

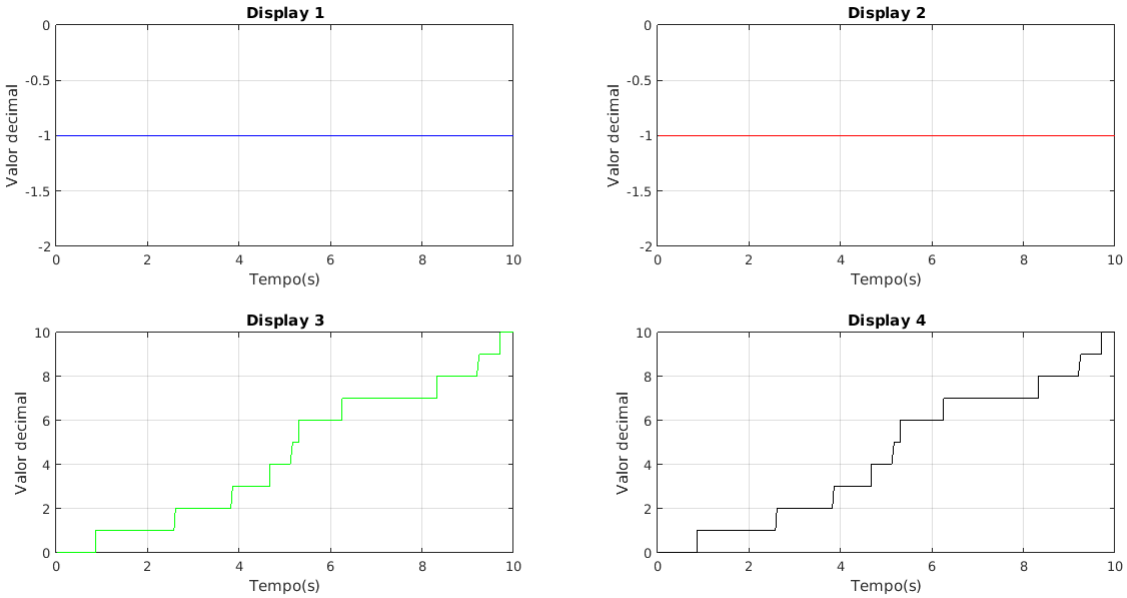


Figura 28 – Gráficos dos estados dos 4 *displays* de sete segmentos da Basys 3 em função do tempo

5 Conclusões

A solução da arquitetura de hardware, sem restrições de *timing*, faz com que o usuário tem cerca de 99 % de disponibilidade dos recursos do FPGA para desenvolvimento de sua lógica de projeto.

O servidor *web* faz uso da comunicação serial de protocolo RS-232 para envio e recebimento de dados à interface gráfica e Basys 3. A restrição de comunicação é baseado no *bit rate* do módulo ***serialcom*** que é de 9600 bits/s, portanto há um limite de dados/segundo que podem ser transmitidos. Esse fato, em conjunto com o contador implementado no módulo ***deco_leds_7seg*** que é de 1.5 milissegundos para um sinal ser multiplexado e enviado, faz com que um determinado dado demore 8 contagens (12 milissegundos) para ser atualizado, portanto a frequência máxima de um sinal de saída sem perda de dados no servidor, é de 83 Hz.

A interface gráfica provê as funcionalidades de interação com o usuário, como o envio do arquivo *bitstream*, o uso de chaves e botões virtuais, arquivo ".txt" com os estados dos sinais de saída e transmissão *web* da placa Basys 3. Essa transmissão com auxílio do OpenCV (OPENCVTEAM, 2018) pode ocorrer com certos atrasos, dependo da qualidade e velocidade de conexão da internet entre o usuário e o servidor.

A escrita dos dados no arquivo ".txt" não é ao mesmo tempo do recebimento do sinal, pois há o tempo de atraso de transmissão e o sistema operacional não ser determinístico, esses fatos devem ser levados em consideração pelo usuário na interpretação dos gráficos das saídas dos LEDs e *displays* em função do tempo.

A estrutura das arquiteturas do servidor e de hardware faz com que o o laboratório remoto seja replicável para outros servidores e FPGAs e, a partir dos exemplos de uso, como o contador de 1 segundos e o detector de sequência, o projeto mostrou-se ser viável para os projetos dos alunos da FGA das disciplinas de PED 1 e PED 2.

5.1 Limitações

As principais limitações do projeto, já discutidas, são:

- Não ser possível a criação de uma projeto estrutural em VHDL, pois isso requer a construção de um *top-level* e portanto o usuário teria que modificar o ***top_wrapper*** e o laboratório remoto não funcionaria da maneira adequada;
- A frequência máxima que pode ser usada nos sinais de saída da Basys 3 é de 83 Hz;

- Atraso na escrita dos dados de saída no arquivo ".txt";
- Atraso na transmissão ao vivo da Basys 3;
- Não atualização dos valores dos estados dos LEDs, *displays* de 7 segmentos e anodos virtuais em elementos ilustrativos na interface gráfica.

5.2 Trabalhos Futuros

A partir do desenvolvimento do laboratório remoto, alguns trabalhos podem ser desenvolvidos a fim de garantir melhorias nas soluções propostas deste projeto.

- Garantir robustez e segurança da infraestrutura do servidor, como no uso de chaves SSH e *firewalls*, para evitar ataques cibernéticos;
- Alocar um IP e porta pública da rede de internet da UnB para o servidor poder ser acessado fora do ambiente da FGA;
- Atribuir um Domain Name System (DNS) para traduzir o IP do servidor;
- Diminuir o tempo de atualização dos dados dos LEDs, *displays* de sete segmentos e anodos do módulo ***deco_leds_7seg***. A atualização é de 12 milissegundos, pois cada dado é precedido do envio de um "id". Uma solução para diminuir esse tempo é enviar um "id" para indicar o início e outro para término da transmissão. Essa solução diminui a atualização para seis contagens (9.0 milissegundos) e aumenta para 111 Hz a máxima frequência de saída sem perda de informações no servidor, além de garantir robustez caso haja perda de informação na transmissão;
- Incluir no arquivo ".txt" e no *script* do Matlab o estado das chaves e botões virtuais em função do tempo;
- Criação de um contador em hardware para o tempo ser enviado, antes do estado da saída, ao servidor para a escrita no arquivo ".txt". Esse contador garante mais fidelidade que o sistema operacional do servidor no registro do tempo de envio de cada dado.
- Desocupar o servidor quando a janela de *browser* da interface gráfica do usuário é fechada, sem ter que esperar pelo término de 10 minutos de uso;
- Atualização dos dados das saídas na interface gráfica por meio de elementos ilustrativos semelhantes aos da placa física;

- Indicar o tempo decrementando de uso do laboratório remoto;
- Desenvolvimento de exemplos de uso para microcontroladores *Picoblaze*;
- Colocar o protótipo do laboratório remoto em funcionamento em um sala reservada na FGA;
- Aplicação de formulários para os alunos de PED 1 e PED2 avaliarem o laboratório remoto em seus projetos.

Referências

BICKING, I. *Virtualenv Documentation*. 2018. Disponível em: <https://virtualenv.pypa.io/en/latest/>. Citado na página 37.

DIGILENT. *Adept 2 Reference Manual*. 2005. Disponível em: <https:/reference.digilentinc.com/reference/software/adept/reference-manual>. Citado na página 38.

DIGILENT. *Basys 3^{TM} FPGA Board Reference Manual*. Pulmann, USA, 2016. 19 p. Citado 5 vezes nas páginas 9, 19, 28, 29 e 30.

DIGILENT. *Basys 3 Artix-7 FPGA Trainer Board: Recommended for Introductory Users*. 2018. Disponível em: <https://store.digilentinc.com/basys-3-artix-7-fpga-trainer-board-recommended-for-introductory-users>. Citado na página 19.

FAYAD, M.; SCHMIDT, D. Object-oriented application frameworks. v. 40, p. 32,24, 10 1997. Citado na página 31.

GARCIA, P. et al. An overview of reconfigurable hardware in embedded systems. *EURASIP Journal on Embedded Systems*, v. 2006, n. 1, p. 056320, Sep 2006. ISSN 1687-3963. Disponível em: <https://doi.org/10.1155/ES/2006/56320>. Citado 3 vezes nas páginas 11, 26 e 27.

GRINBERG, M. *Flask Web Development: Developing Web Applications with Python*. 1st. ed. [S.l.]: O'Reilly Media, Inc., 2014. ISBN 1449372627, 9781449372620. Citado 2 vezes nas páginas 31 e 37.

GT-MRE. *RELLE - Remote Labs Learning Environment*. 2018. Disponível em: <http://relle.ufsc.br/labs>. Citado na página 25.

GUPTA, P. Spring web mvc framework for rapid open source j2ee application development: a case study. Singhania University, India, 2010. Citado na página 31.

IANA. *Service Name and Transport Protocol Port Number Registry*. 2018. Disponível em: <https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml>. Citado na página 38.

I.GROUT. Remote laboratories to support electrical and information engineering (eie) laboratory access for students with disabilities. In: *2014 25th EAEEIE Annual Conference (EAEEIE)*. [S.l.: s.n.], 2014. p. 21–24. Citado na página 26.

INJOSOFTAB. *ASCII Code - The extended ASCII table*. 2018. Disponível em: <https://www.ascii-code.com/>. Citado na página 33.

INSTRUMENTS, N. *Fundamentos da tecnologia FPGA*. 2013. Disponível em: <http://www.ni.com/white-paper/6983/pt/>. Citado 3 vezes nas páginas 9, 26 e 27.

LIECHTI, C. *pySerial's documentation*. 2014. Disponível em: <https://pythonhosted.org/pyserial/>. Citado na página 38.

MA, J.; JEFFREY; NICKERSON, V. Hands-on, simulated, and remote laboratories: A comparative literature review. *ACM Comput. Surv*, v. 38, p. 7, 2006. Citado 3 vezes nas páginas 19, 20 e 25.

MATHWORKS. *MATLAB*. 2018. Disponível em: <https://www.mathworks.com/products/matlab.html>. Citado na página 47.

MATRICULAWEB. *Fluxo Engenharia Eletrônica da Universidade de Brasília*. 2018. Disponível em: <https://matriculaweb.unb.br/graduacao/curso_dados.aspx?cod=1601>. Citado 3 vezes nas páginas 19, 29 e 30.

MORGAN, F. et al. Remote fpga lab with interactive control and visualisation interface. In: . [S.l.: s.n.], 2011. p. 496–499. Citado na página 25.

OPENCVTEAM. *OpenCV library*. 2018. Disponível em: <https://opencv.org/>. Citado 2 vezes nas páginas 38 e 49.

RODRIGUEZ-GIL, L. et al. Graphic technologies for virtual, remote and hybrid laboratories: Weblab-fpga hybrid lab. In: *2014 11th International Conference on Remote Engineering and Virtual Instrumentation (REV)*. [S.l.: s.n.], 2014. p. 163–166. Citado na página 20.

SASS, R.; SCHMIDT, A. G. *Embedded Systems Design with Platform FPGAs: Principles and Practices*. 1st. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2010. ISBN 9780080921785, 9780123743336. Citado na página 33.

TANENBAUM, A. *Redes de computadores*. CAMPUS - RJ, 2003. ISBN 9788535211856. Disponível em: <https://books.google.com.br/books?id=0tjB8FbV590C>. Citado na página 37.

VICENTE, C. *Laboratório Remoto Para Prototipação De Circuitos Digitais Utilizando Kits de Desenvolvimento FPGAS XILINX*. 2016. Bacharelado em ciência da computação), (Centro Uinversitário Eurípides de Marilia), Marilia, Brasil. Citado na página 25.

XILINX. *7 Series FPGAs Configurable Logic Block*. California, USA, 2016. 74 p. Citado 4 vezes nas páginas 9, 26, 27 e 28.

XILINX. *7 Series FPGAs Data Sheet: Overview*. California, USA, 2018. 18 p. Citado na página 28.