**UNIVERSIDADE FEDERAL DE SANTA CATARINA**
**DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS**

Ígor Assis Rocha Yamamoto

# DESENVOLVIMENTO DE ROTINAS DE PROGRAMAÇÃO EM ALTO NÍVEL PARA CONFIGURAÇÃO DE FPGAS

Orientador:
Prof. Dr. Rodolfo César Costa Flesch

Florianópolis

2016

## RESUMO

O objetivo do trabalho de iniciação científica foi desenvolver rotinas de programação de alto nível que podem ser integradas em ambiente LabVIEW para programação e configuração de FPGAs. As atividades foram realizadas com o uso da placa de ensino *NI Digital Electronics FPGA Board*, possibilitando a criação de elementos de software que exploram as ferramentas do kit de ensino, como entradas e saídas - digitais e analógicas. O material produzido com o kit pode ser utilizado no auxílio ao ensino de eletrônica digital de forma mais intuitiva, através do ambiente de desenvolvimento gráfico oferecido pelo LabVIEW, em contrapartida à programação em baixo nível.


**Palavras-chave:** FPGA, Instrumentação virtual, Programação em alto nível.

**SUMÁRIO**

# 1 INTRODUÇÃO

## 1.1 CONTEXTUALIZAÇÃO E MOTIVAÇÃO

Nos dias atuais, tornou-se cada vez mais comum o emprego de sistemas microprocessados em itens de uso cotidiano. Em comparação com alternativas anteriores, baseadas em eletrônica analógica, os sistemas microprocessados permitem uma redução da dimensão física do produto final e também uma redução significativa no tempo de desenvolvimento. Todavia, o uso de microprocessadores está limitado a aplicações que apresentam dinâmica da ordem de microssegundos, uma vez que há limites físicos para a frequência do *clock* que pode ser empregada nesse tipo de dispositivo (FLYNN, 1999). Como alternativa entre a solução analógica instantânea e a microprocessada, surgiram os FPGAs (do inglês *Field-Programmable Gate Array*), ou Arranjo de Portas Programável em Campo traduzido para o português. FPGA é um circuito integrado que pode ser configurado (programado) pelo usuário e continuar apresentando desempenho similar ao de um hardware desenvolvido especificamente para determinado fim (PARNELL; BRYNER, 2004). Apesar da versatilidade, a programação de FPGAs geralmente é realizada em muito baixo nível através das HDLs (linguagens de descrição de hardware, do inglês *Hardware Description Language*, tais como Verilog ou VHDL), o que acaba por dificultar sua ampla utilização no meio acadêmico.

## 1.2 OBJETIVOS DO TRABALHO

O trabalho realizado na iniciação científica teve como objetivo principal explorar alternativas mais atrativas para a programação e configuração de FPGAs. Através das ferramentas de software do LabVIEW, foram implementadas rotinas de programação em alto nível, utilizando os recursos e facilidades da programação gráfica. O uso do LabVIEW, em contrapartida ao modo usual de programação em HDL, permitiu a redução no tempo de criação de programas, além de torná-los mais simples e de fácil entendimento. Essas vantagens oferecidas para a manipulação de FPGAs são ideais para estudantes de engenharia focarem suas atividades nos conceitos fundamentais de eletrônica digital, deixando de lado a preocupação excessiva no aprendizado de uma nova linguagem.

## 1.3 ESTRUTURA DA PESQUISA

As atividades como bolsista de iniciação científica começaram com a capacitação no uso das ferramentas necessárias para cumprir os objetivos da pesquisa, como o aprendizado da linguagem LabVIEW e uso do kit de desenvolvimento *NI Digital Electronics FPGA Board*. Após essa etapa, foi possível criar elementos de software úteis para o uso em aulas de eletrônica digital das fases iniciais dos cursos de engenharia, como contadores e unidades lógicas e aritméticas. O período final das atividades foi destinado ao estudo de tópicos de controle preditivo e assistência a uma mestranda em sua pesquisa.

## 1.4 ESTRUTURA DO RELATÓRIO

Este relatório está decidido em 5 capítulos. No capítulo 2, é feita uma revisão de literatura, apresentando as bases teóricas, softwares e hardwares utilizados para o desenvolvimento das atividades. No capítulo 3, são discutidos os projetos elaborados com o *NI Digital Electronics FPGA Board*, NI ELVIS II e LabVIEW. No capítulo 4, é descrita a atividade realizada em conjunto com a mestranda. No capítulo 5, é feita uma avaliação geral sobre o programa de iniciação científica e resultados obtidos.

# 2 REVISÃO DE LITERATURA

Neste capítulo serão apresentados conceitos importantes para o desenvolvimento da pesquisa de iniciação científica. Também serão introduzidas as ferramentas de software e hardware utilizadas para a implementação dos projetos.

## 2.1 FPGA

FPGA (*Field-Programmable Gate Array*) é um chip de silício reprogramável (NI, 2012). Sua programação é realizada pela reconfiguração das conexões no próprio chip. O uso de FPGAs está crescendo em todos os setores industriais, devido à combinação dos melhores recursos de circuitos integrados para tarefa específica (ASICs - *application-specific integrated circuits*) e sistemas microprocessados. Entre os benefícios do FPGA, podemos destacar: maior poder de processamento em relação aos microprocessadores, prototipagem e verificação rápida, facilidade de *upgrade* em campo.

## 2.2 CONTROLE PREDITIVO BASEADO EM MODELO

Controle preditivo baseado em modelo é uma técnica avançada de controle (CAMACHO; ALBA, 2013), que tem sido usada na indústria de processos desde a década de 1980 e tem se desenvolvido consideravelmente nos últimos anos na comunidade científica. O método se fundamenta na otimização para encontrar um valor de controle a ser aplicado em cada instante de tempo, de modo a minimizar um objetivo futuro.

## 2.3 SOFTWARE UTILIZADO

### 2.3.1 LabVIEW

O LabVIEW é um ambiente gráfico para desenvolvimento de sistemas (NI, 2013a), concebido para acelerar a produtividade de engenheiros e cientistas. O desenvolvimento é feito por uma sintaxe de programação gráfica, baseada no modelo de fluxo de dados. O Lab-

VIEW é amplamente utilizado em diversas áreas da engenharia, tais como: automação, aquisição de dados e processamento de sinais, instrumentação virtual, ensino e pesquisa. O ambiente de programação permite a fácil criação de interfaces com o usuário que, aliadas aos diagramas de blocos, compõe os instrumentos virtuais (IVs). Através do módulo LabVIEW FPGA (NI, 2014), é possível configurar e programar FPGAs no mesmo ambiente de desenvolvimento.

## 2.4 HARDWARE UTILIZADO

### 2.4.1 *NI Digital Electronics FPGA Board*

A placa de desenvolvimento *NI Digital Electronics FPGA Board* foi criada com o objetivo de auxiliar educadores no ensino de eletrônica digital (NI, 2009). Ela possui o FPGA *Xilinx Spartan-3E* integrado, permitindo sua configuração através de linguagens de baixo nível (Verilog e VHDL) com o software Xilinx ISE tools ou pelo ambiente gráfico do LabVIEW. A placa possui área para prototipação e diversos recursos para acessar sinais do mundo real, ideal para educadores explorarem conceitos no ensino. A placa também permite plena integração com a plataforma NI ELVIS.

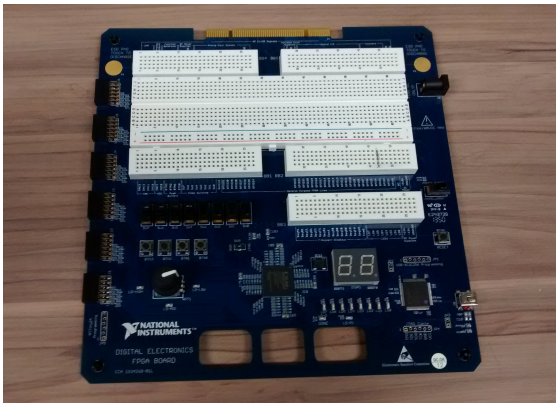


Figura 1 – *NI Digital Electronics FPGA Board*

### 2.4.2 NI ELVIS II Series

NI ELVIS II Series(*National Instruments Educational Laboratory Virtual Instrumentation Suite II Series*) é um ambiente de prototipação e design baseado em LabVIEW, dirigido ao meio acadêmico e científico (NI, 2011c). O NI ELVIS II fornece acesso a diversos instrumentos muito usados em atividades de laboratório, como osciloscópio, mulitímetro digital, gerador de função, leitura e escrita de sinais digital.


Figura 2 – *NI ELVIS II*

### 2.4.3 NI myRIO

O NI myRIO é um dispositivo de hardware para projetos embarcados, destinado a estudantes (NI, 2013b). Seu intuito é ajudá-los a projetar e construir sistemas complexos de engenharia com maior rapidez e acessibilidade. O dispositivo conta com um sistema em um chip (SoC, do inglês *System-on-a-chip*) completamente programável *Xilinx Zynq-7010*, incluindo um processador *dual-core ARM Cortex-A9* de tempo real, integrado a um FPGA *Artix-7*. Toda a programação do dispositivo pode ser feita no ambiente de desenvolvimento gráfio do LabVIEW.


Figura 3 – *NI myRIO*

# 3 PROJETOS DE ELETRÔNICA DIGITAL

Este capítulo contém os objetivos, especificações e implementações de quatro projetos de eletrônica digital, seguido de uma sessão discutindo a validação dos mesmos. Os projetos, implementados em LabVIEW, utilizam os recursos da placa de ensino *NI Digital Electronics FPGA Board* e são testados com o auxílio da plataforma NI ELVIS II. A base para formulação dos projetos é fornecida pela *National Instruments*, em formato de tutoriais (NI, 2011b).

## 3.1 *MINORITY GATE*

### 3.1.1 Objetivo

O objetivo do projeto é criar um circuito combinacional em alto nível, trabalhando alguns conceitos fundamentais de eletrônica digital: álgebra booleana, tabela verdade, mapa de Karnaugh, composição de portas lógicas. O exercício também introduz a construção de projetos hierárquicos pelo LabVIEW, através das sub-VIs.

### 3.1.2 Especificação do Projeto

O circuito *Minority Gate* contém três entradas (A, B e C) e uma saída (F). Ele funciona da seguinte forma: a saída será verdadeira sempre que o número de entradas verdadeiras for menor que o de entradas falsas. O projeto foi desenvolvido usando o conceito de hierarquia, começando com as portas de baixo nível (implementadas através das portas lógicas universais NAND e NOR, por motivos didáticos) até a composição final da *Minority Gate*.

### 3.1.3 Implementação

De acordo com a especificação do projeto, foi construída a tabela verdade (Tabela 1) contendo as entradas A,B e C e a saída F. Após a criação da tabela, foi utilizada a técnica para simplificação de expressões booleanas por mapa de Karnaugh (Figura 4). Foram obtidas duas funções minimizadas da saída F: a primeira sob a forma de minitermos

e a segunda, maxitermos. Essa última foi utilizada como base para a execução do projeto.

| Entradas | | | Saída |
|---|---|---|---|
| A | B | C | F |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

Tabela 1 – Tabela Verdade - Minority Gate

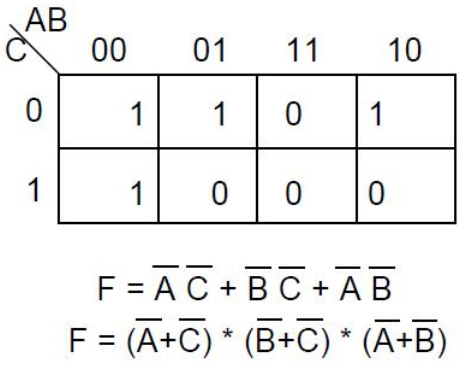


Figura 4 – *Mapa de Karnaugh - Minority Gate*

A implementação do circuito combinacional em linguagem LabVIEW teve três níveis hierárquicos: no primeiro (Figura 5), foi feita a composição das portas OR e AND a partir das portas universais NAND e NOR; no segundo (Figura 6), as portas criadas anteriormente foram utilizadas para compor a *Minority Gate*; e no topo da hierarquia (Figura 7), a *Minority Gate* foi usada para receber as entradas dos interruptores (SW0-2) e fornecer a saída para o LED0.

Com o intuito de assegurar o caráter combinacional do circuito (a saída F em determinado instante de tempo depende apenas das entradas A,B e C nesse mesmo instante), a implementação do circuito

foi realizada através da estrutura temporal do LabVIEW *Single-Cycle Timed Loop*, garantindo assim que todas as operações lógicas sejam feitas em um único ciclo de *clock* do FPGA.

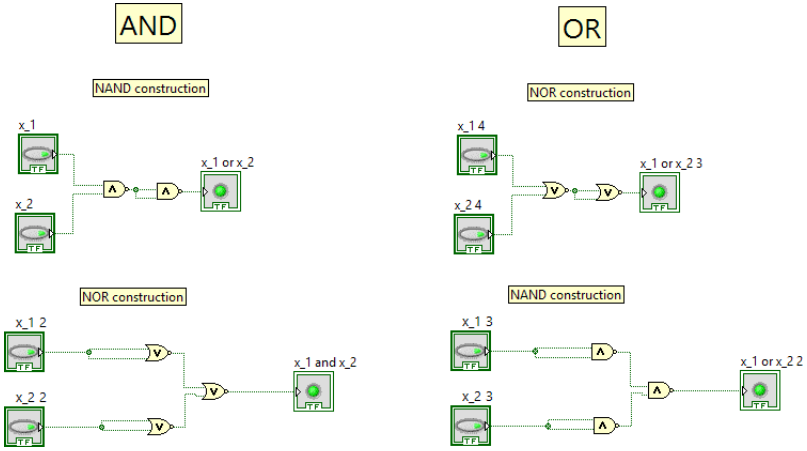


Figura 5 – *Construção das portas AND e OR a partir das portas universais NAND e NOR*

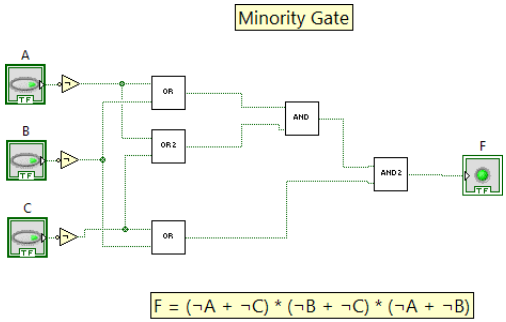


Figura 6 – *Construção da Minority Gate a partir das portas AND e OR*

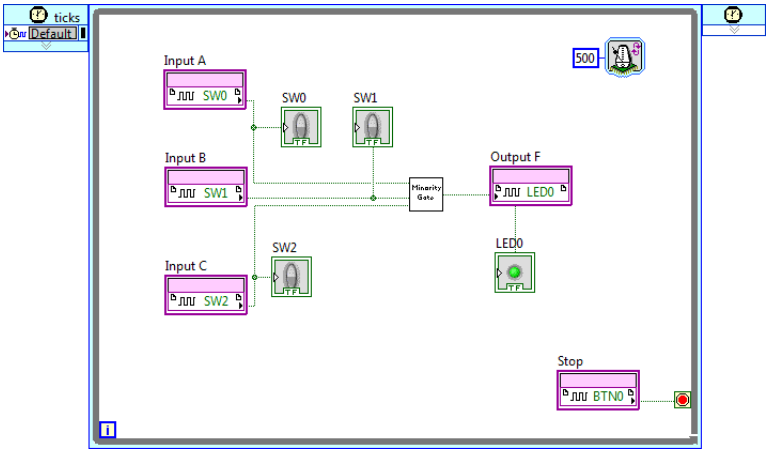


Figura 7 – *Topo de hierarquia do circuito combinacional*

## 3.2 DETECTOR DE SEQUÊNCIA

### 3.2.1 Objetivo

O objetivo deste projeto é explorar o conceito de máquinas de estado finito (FSM, do inglês *Finite State Machine*) através da criação de um detector de sequência.

### 3.2.2 Especificação do Projeto

O problema foi definido em projetar um detector para a sequência de bits 1000. O sistema contém duas entradas: a primeira com os dados da sequência (X) e a outra que aciona a parada de execução (*Stop*). Existe uma saída indicando a detecção da sequência. O sistema opera de maneira a realizar uma contínua detecção da sequência 1000 na entrada X. O programa só termina de executar quando a entrada de parada é acionada.

### 3.2.3 Implementação

Primeiramente, foi desenhada a máquina de estados que representa o comportamento do sistema de detecção, conforme mostra a Figura 8. Ela é composta por 5 estados, apelidados de S0, S1, S2, S3 e S4 com seus respectivos significados na tabela da Figura 8. S0 (*Idle*) representa o estado em que nenhum elemento da sequência foi detectado ainda; S1 (*One Detected*) é o estado em que o primeiro elemento da sequência (1) foi detectado; S2 ($1^{st}$ *Zero*), S3 ($2^{nd}$ *Zero*) e S4 ($3^{rd}$ *Zero*) representam a detecção dos elementos subsequentes (000). Somente no estado S4, a saída de detecção da sequência é indicada.

O código em diagrama de blocos, criado no LabVIEW, pode ser visto na Figura 9. A entrada X usa o interruptor 0 (SW0) da placa de ensino, com o envio de dados controlado pela transição de subida do botão 0 (BTN0). O LED0 indica a saída do sistema. Como pode ser observado, o projeto em LabVIEW tornou a implementação dos estados do sistema demasiadamente simples, sendo composta por um bloco de estrutura de casos (*case structure*).

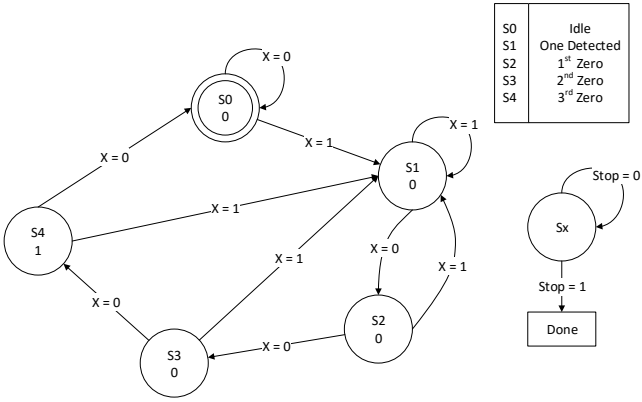


Figura 8 – Máquina de estados do detector de sequência

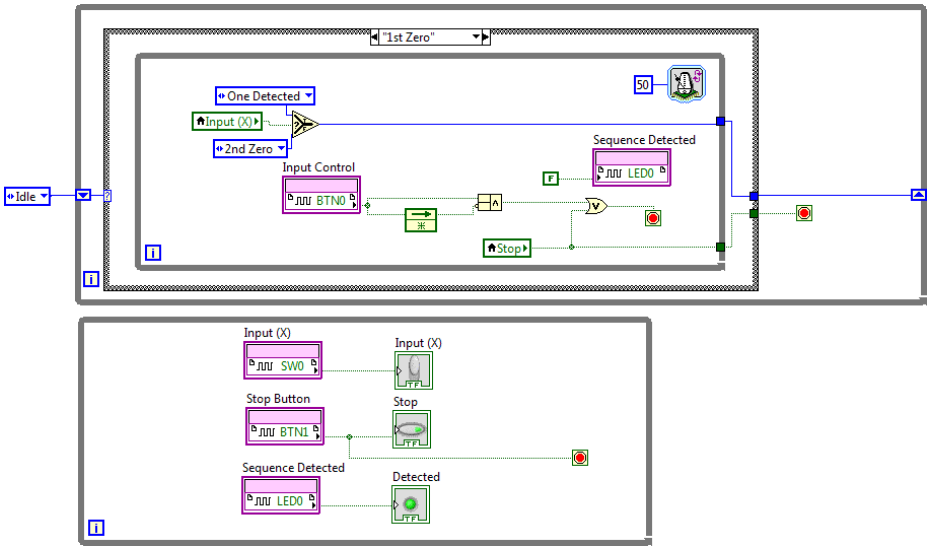


Figura 9 – Detector de sequência

## 3.3 CONTADOR PROGRAMÁVEL

### 3.3.1 Objetivo

O objetivo desta atividade é fazer uma introdução ao projeto de circuitos sequências em alto nível. Com o desenvolvimento de um contador programável em LabVIEW, o estudante de eletrônica digital pode praticar os conceitos vistos em aula.

### 3.3.2 Especificação do Projeto

O contador programável (Figura 10) possui as seguintes funcionalidades:

1. zerar (*reset*) a contagem atual;
2. segurar (*hold*) a contagem atual;
3. carregar (*load*) a entrada de contagem paralela de 4 bits(*Parallel Count Input*);

4. incrementar/decrementar em 1 bit;
5. exibir a saída da contagem atual;
6. exibir a saída de fim de contagem (*terminal count*).

Os sinais de entrada, que definem as funcionalidades do contador, seguem uma ordem de prioridade. A prioridade das entradas é definida de acordo com a numeração da lista de funcionalidades apresentada anteriormente, sendo zerar a função de maior precedência e incrementar/decrementar, a de menor.

O fim de contagem é indicado (lógico 1) quando, em modo de operação incremental, a contagem chega em 15; ou, em modo decremental, chega em 0.

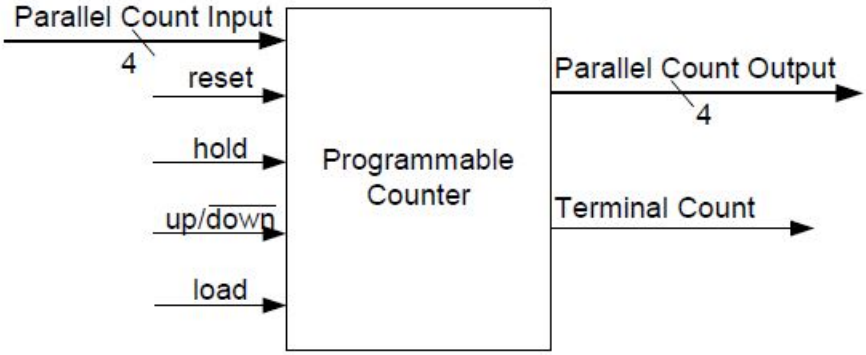


Figura 10 – Projeto - Contador programável de 4 bits

### 3.3.3 Implementação

A implementação do contador em LabVIEW foi feita com blocos de condições (*select blocks*) ligados em cascata, de acordo com a precedência da respectiva funcionalidade. Cada condição é controlada por um sinal de entrada: o inerruptor 0 (SW0) controla a função incrementar/decrementar; o botão 2 (BTN2) controla a função segurar; o botão 1 (BTN1), a função carregar; o botão 0 (BTN0), a função zerar.

O número da contagem de saída é representado nos LEDs (LED0-3) e o término da contagem no LED4. A lógica de saturação do contador é feita com uma estrutura de caso (falso, quando a contagem é

decremental; verdadeiro, quando incremental). Além disso, o botão 3 (BTN3) é responsável pela parada de execução e os interruptores de 1 a 4 (SW1-4) formam a entrada de contagem paralela.

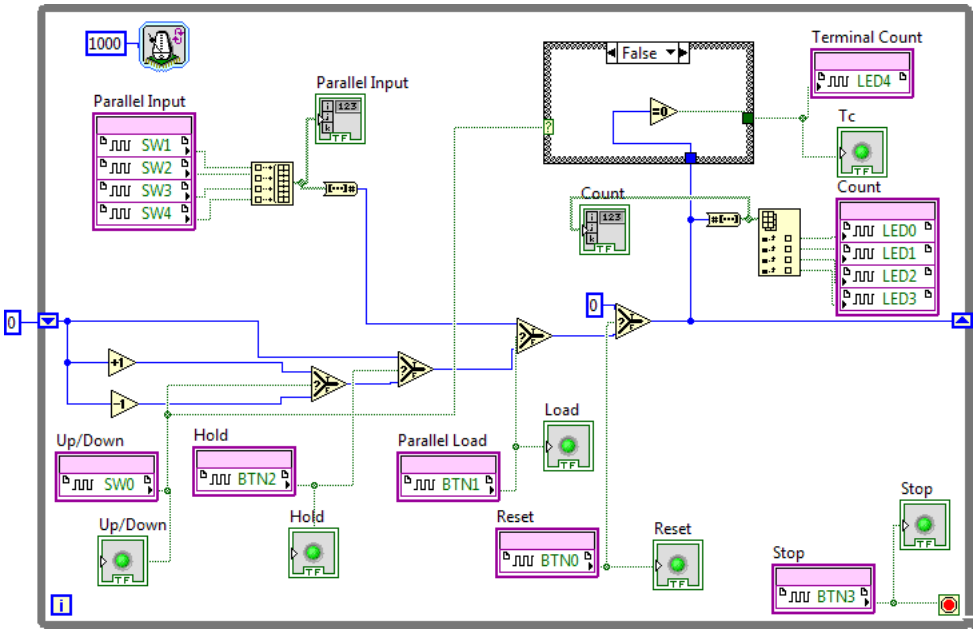


Figura 11 – Contador programável de 4 bits

## 3.4 USO DE MEMÓRIA ROM E RAM

### 3.4.1 Objetivo

O objetivo deste projeto é mostrar como a memória do FPGA pode ser utilizada como ROM e RAM. Na primeira parte do projeto, um exemplo de uso da memória do FPGA como ROM é implementado; enquanto na segunda parte, a memória RAM é explorada.

### 3.4.2 Memória ROM

#### 3.4.2.1 Especificação do Projeto

O exemplo consiste em fazer uma busca de conteúdo em uma memória ROM. Esta possui o tamanho $16x8$, inicializada com os inteiros de 8 bits: 12, 46, 23, 78, 90, 11, 22, 44, 66, 88, 100, 128, 90, 55, 34, 45. O programa no FPGA procura o número desejado através de uma entrada de 8 bits. A busca se inicia quando um botão é pressionado e termina no momento em que o conteúdo procurado é encontrado (indicando sucesso) ou quando o fim da memória é detectado (indicando falha). O fim de busca é indicado em uma saída.

#### 3.4.2.2 Implementação

Um bloco de memória foi criado no FPGA para ser tratado como ROM. O endereçamento foi feito do número 0 ao 15, com os valores inicializados em cada endereço (Figura 12). O acesso à memória ROM foi realizado a partir do bloco pronto de leitura de memória do LabVIEW FPGA (Figura 13).

A entrada do número desejado de 8 bits foi implementada através dos interruptores 0 a 7(SW0-7). O controle do início da busca é feito através do botão 0 (BTN0). O LED0 indica a falha da procura, o LED1 mostra o sucesso da busca e o LED2 indica o fim do processo.

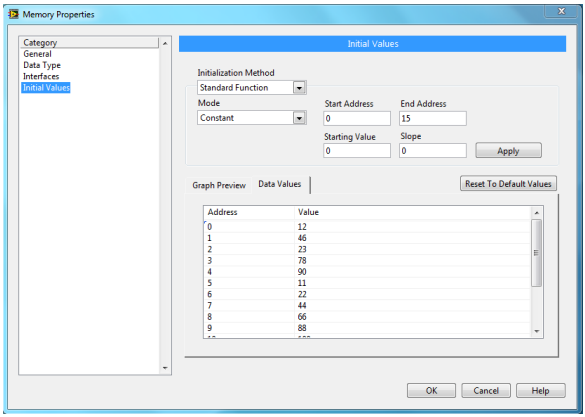


Figura 12 – Criação da memória ROM em LabVIEW FPGA

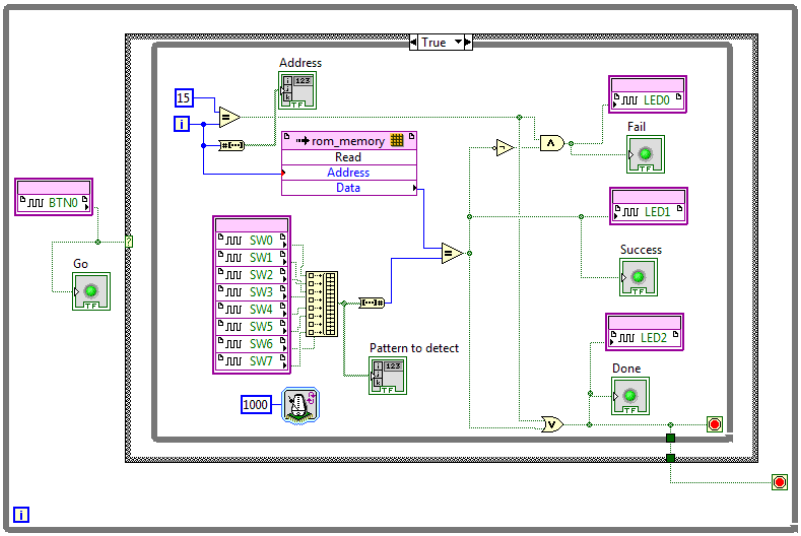


Figura 13 – Uso de memória ROM

### 3.4.3 Memória RAM

#### 3.4.3.1 Especificação do Projeto

O exemplo consiste em criar um bloco de memória $32x8$ no FPGA a ser tratado como RAM. Os valores iniciais de cada endereço da memória são todos nulos. O programa contém duas formas de interação com o usuário: escrita e leitura de dados.

A escrita de dados na memória tem início com a ativação de um botão. Uma entrada de 8 bits é fornecida e armazenada no primeiro endereço da memória. A partir de então, cada endereço de memória seguinte receberá o valor do conteúdo anterior acrescido de 3. Uma saída indica o termino da escrita.

A leitura de dados é realizada ao pressionar-se um botão. Todos os conteúdos dos endereços de memória são indicados em uma saída de 8 bits, no intervalo de 1s para visualização dos dados. Uma saída indica o término da leitura.

Os processos de escrita e leitura são independentes e a escrita pode ocorrer inúmeras vezes. Além disso, um botão de parada termina a execução do programa a qualquer momento.

#### 3.4.3.2 Implementação

Um bloco de memória foi criado no FPGA para ser tratado como RAM. Os valores correspondentes a cada endereço de 0 a 31 foram inicializados em zero. Os acessos para leitura e escrita na memória foram realizados com os blocos prontos para o LabVIEW FPGA.

O diagrama de blocos do programa (Figura 15) é dividido em três *while loops* independentes: o primeiro possui a lógica de escrita; o segundo, a de leitura e o terceiro lê continuamente o estado do botão de parada (BTN2).

O início do processo de escrita é controlado pelo botão 0 (BTN0). A entrada de dados de 8 bits foi implementada com os interruptores (SW0-7). A cada segundo os dados são escritos na memória, mostrados no painel de controle (Figrua 14) e acrescidos de três unidades até que o fim da memória seja atingido.

O processo de leitura de dados da memória é controlado pelo botão 1 (BTN1). A saída de 8 bits dos valores lidos é indicada nos LEDs (LED0-7) e no painel de controle a cada segundo até que o fim da memória seja atingido.

O botão de parada (BTN2) pode ser acionado em qualquer momento do programa. Uma vez ativado, duas variáveis locais do botão de parada podem ser lidas no interior dos processos de escrita ou leitura, interrompendo a execução.

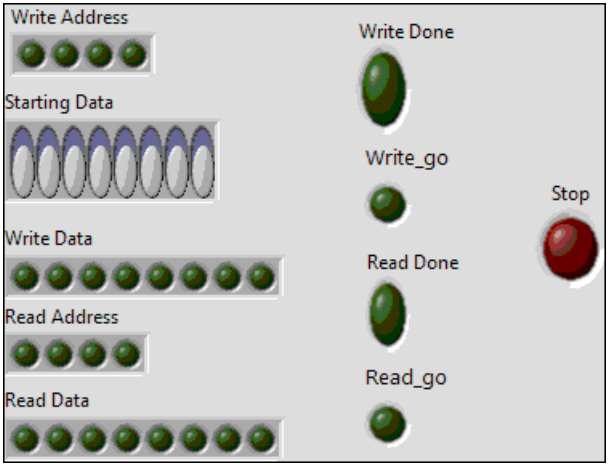


Figura 14 – Uso de memória RAM

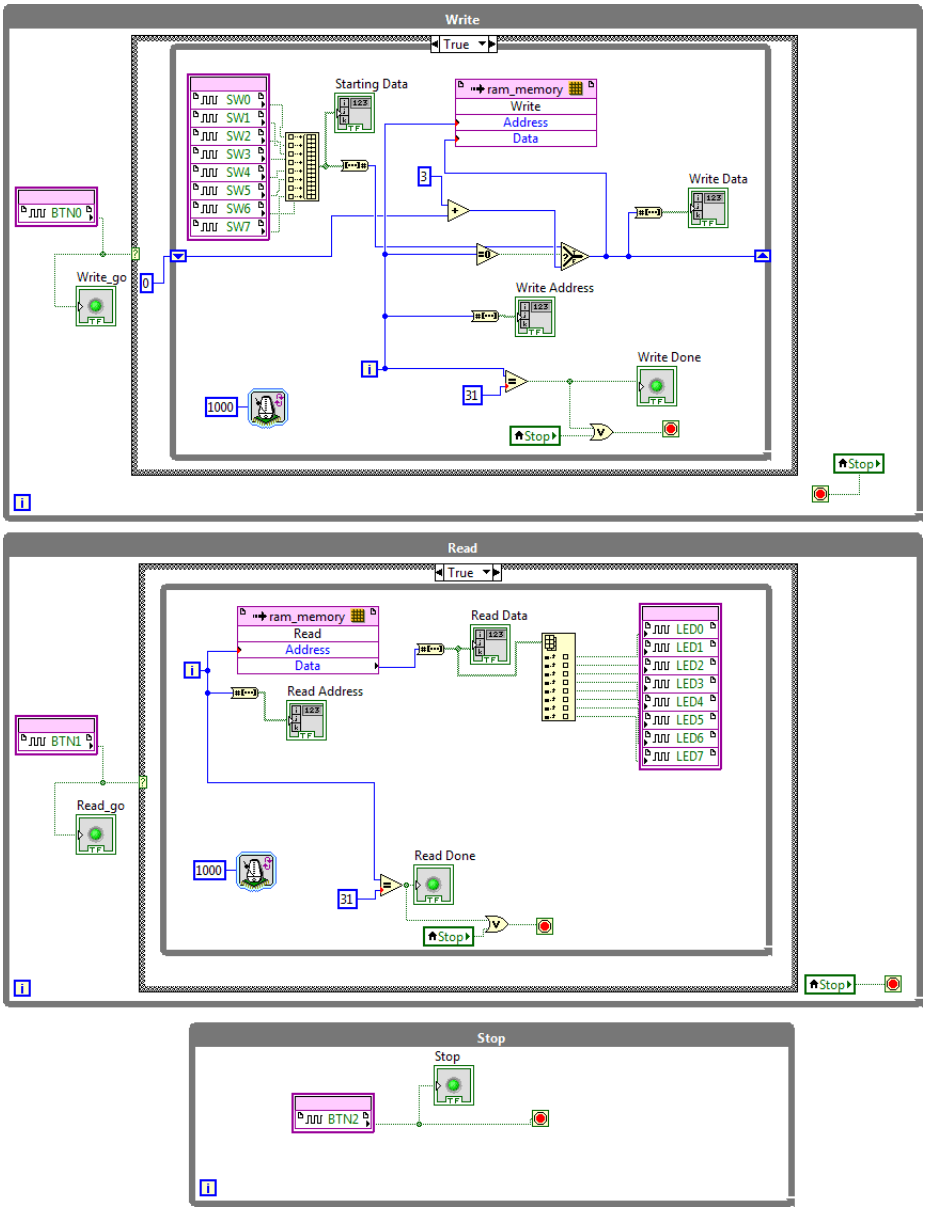


Figura 15 – Uso de memória RAM

## 3.5 VALIDAÇÃO DOS PROJETOS

Após a implementação de cada projeto, foi estabelecida uma metodologia para a validar os resultados. A primeira etapa da avaliação do sistema foi realizada na própria placa de ensino. A segunda etapa de testes foi feita com o auxílio da plataforma NI ELVIS II.

### 3.5.1 Placa *NI Digital Electronics FPGA Board*

Todos os projetos apresentados anteriormente foram implementados explorando os recursos fornecidos pela placa de ensino *NI Digital Electronics FPGA Board*. Assim, a primeira etapa de validação dos projetos consistiu em testar os códigos dos programas utilizando diretamente as entradas e saídas da placa. Todas as combinações de entradas, com suas respectivas saídas esperadas foram anotadas e testadas manualmente.

### 3.5.2 NI ELVIS II

Dependendo do projeto analisado, torna-se difícil ou até mesmo impraticável testar todas as possibilidades de entrada e saída gerada do sistema. A plataforma NI ELVIS II permite que cada combinação possível seja testada de forma automatizada. O programa *Simple FPGA Tester* (Figura 16), desenvolvido e disponibilizado pela *National Instruments*, foi usado para validar os resultados através do NI ELVIS II.

O *Simple FPGA Tester* deve ser configurado com os parâmetros indicados no painel de controle: nome do dispositivo (Elvis II/II+ *Device Name*), número de entradas (*Number of Inputs*), número de saídas (*Number of Outputs*), arquivo textual com as entradas a serem testadas (*Input List File*), tempo de atraso entre cada teste de entrada (*Wait time after Writing FPGA in Milliseconds*) e arquivo textual com as saídas esperadas (*Expected Output File*).

Após a configuração com os parâmetros desejados, o *Simple FPGA Tester* é executado junto ao programa principal. Uma mensagem é gerada no painel de controle indicando sucesso caso as saídas esperadas coincidam com as observadas; ou falha caso as saídas não coincidam. No painel de controle também é possível ver quais entradas geraram as saídas observadas em comparação com as esperadas,

facilitando o processo de identificação dos erros.

Pequenas modificações no código em LabVIEW dos projetos foram realizadas para integrar a placa de ensino com o NI ELVIS II. Para que a placa recebesse os sinais de entrada gerados pelo NI ELVIS II e este lesse os sinais de saídas gerados pela placa, foi necessário trocar os blocos referentes as entradas e saídas da placa (BTNx, SWx, LEDx) por blocos referentes as linhas de E/S de propósito geral (GPIO *lines*).

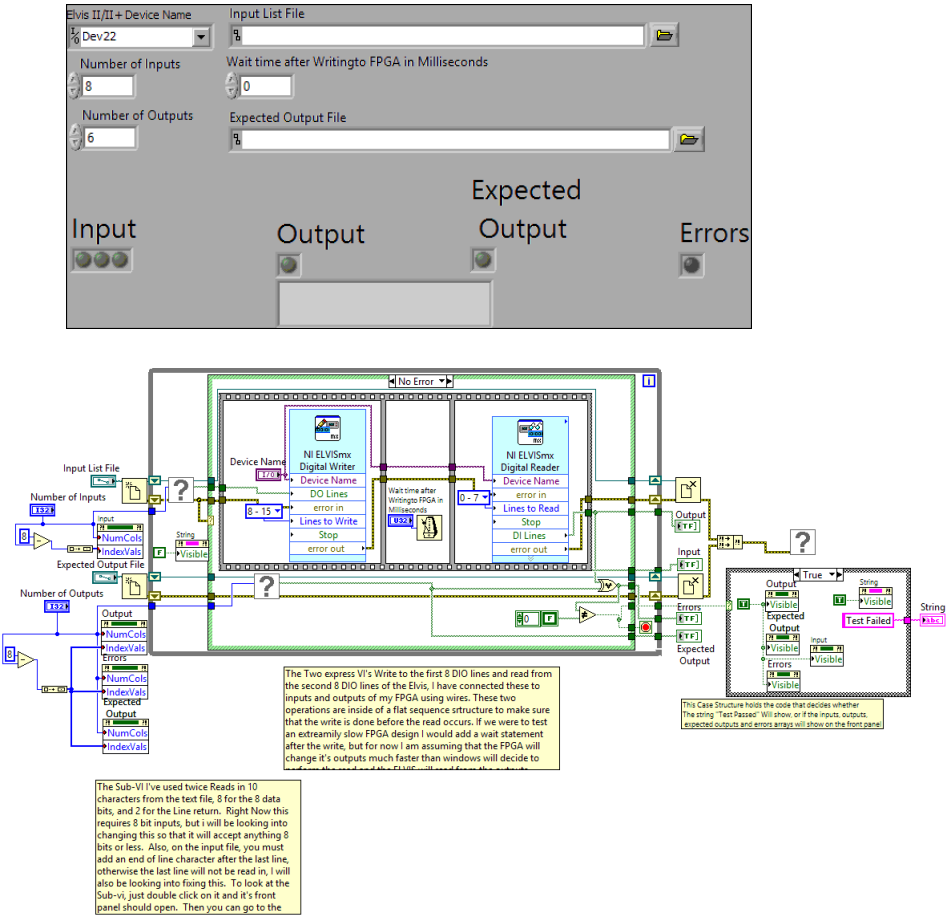


Figura 16 – *Simple FPGA Tester*

# 4 ATIVIDADES EM CONJUNTO À MESTRANDA

O período final da bolsa de iniciação científica foi destinado ao estudo introdutório de controle preditivo e apoio à pesquisa da mestranda Silvane Schons. Os estudos foram guiados por aulas expositivas iniciais da disciplina de controle preditivo. A assistência à mestranda foi prestada auxiliando em alguns aspectos da implementação de um projeto envolvendo a criação de um controle PID em FPGA e aplicação da técnica de controle preditivo.

## 4.1 PROJETO DE CONTROLE COM O NI MYRIO

Este projeto tem como objetivo desenvolver um controlador em duas camadas: implementação de um PID em FPGA e desenvolvimento de um Controlador Preditivo Generalizado (GPC, do inglês Generalized Predictive Control) em um sistema operacional de tempo real.

O dispositivo NI myRIO provê as ferramentas necessárias para este projeto (FPGA e sistema operacional de tempo real), permitindo o completo desenvolvimento em LabVIEW.

O controlador preditivo no sistema operacional de tempo real funciona como um controlador de alto nível que guia para onde o controlador PID de baixo nível deve conduzir o processo. O controle preditivo fornece uma referência a ser seguida pelo PID no FPGA a cada intervalo de tempo pré-estabelecido. O PID, então, deve ser capaz de executar rapidamente.

Nas Figuras 17 e 18 estão parte do processo de implementação do controlador em linguagem gráfica. O PID é criado a partir de um dos blocos prontos para uso em LabVIEW FPGA. O controlador preditivo é realizado com base nas equações para o modelo GPC (CLARKE; MOHTADI; TUFFS, 1987), utilizando o *Quadratic Programming VI* para efetuar a otimização não-linear.

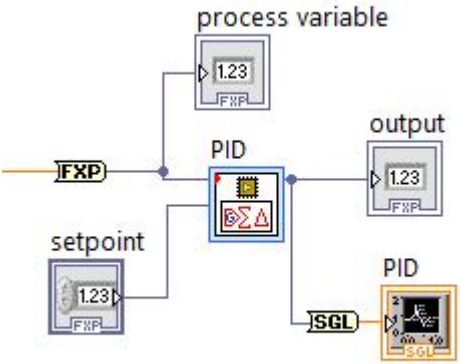


Figura 17 – Bloco Controlador PID em FPGA

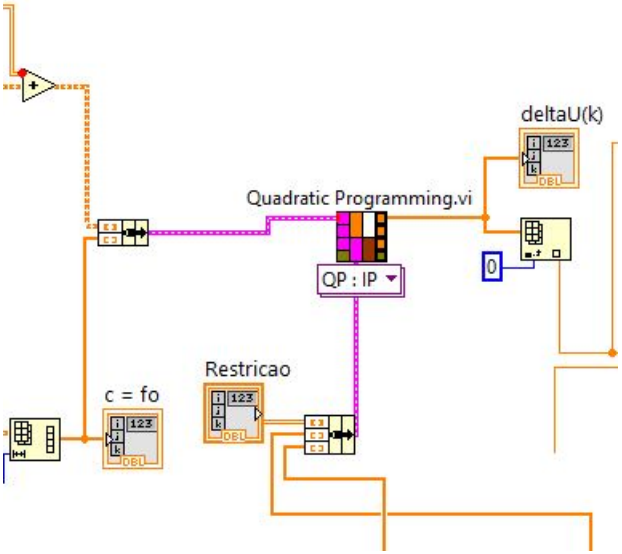


Figura 18 – *Quadratic Programming VI* dentro do GPC em LabVIEW

# 5 CONCLUSÃO

## 5.1 SOBRE OS PROJETOS

Os projetos desenvolvidos durante o período de vigência da bolsa de iniciação científica ocorreram adequadamente. As atividades realizadas com a placa *NI Digital Electronics FPGA Board* mostram-se úteis para a aprendizagem da linguagem de programação gráfica do LabVIEW. Através da exploração de conceitos de eletrônica digital, foi possível criar elementos de software em alto nível que podem ser usados no ensino da disciplina.

As atividades em apoio à mestranda, deram seguimento ao conhecimento adquirido na primeira parte da iniciação científica, utilizando ferramentas similares para seu desenvolvimento (hardware com FPGA embutido, integrado ao ambiente LabVIEW). Apesar do tópico avançado, foi possível acompanhar um pouco do trabalho realizado pela mestranda e aprender os fundamentos básicos de controle preditivo.

## 5.2 SOBRE A INICIAÇÃO CIENTÍFICA

As atividades de iniciação científica proporcionaram o desenvolvimento de várias habilidades úteis. O conhecimento adquirido no ambiente de pesquisa trouxe resultados benéficos para a graduação, atuando de forma complementar ao que as atividades acadêmicas oferecem. O trabalho prático em laboratório foi uma ótima experiência de aprendizado, propiciando contato com diversos equipamentos e softwares utilizados na engenharia. O convívio com outros graduandos, mestrandos e doutorandos também foi proveitoso para troca de experiências.

A oportunidade de inserção no meio científico foi possível graças a bolsa concedida pela CAPES (Coordenação de Aperfeiçoamento de Pessoal do Nível Superior) através do programa Jovens Talentos para a Ciência e graças ao programa para bolsistas do PIBIC (Programa Institucional de Bolsas de Iniciação Científica) da CNPQ (Conselho Nacional de Desenvolvimento Científico e Tecnológico). O trabalho de pesquisa foi realizado com a orientação do professor Rodolfo César Costa Flesch, que forneceu todo o amparo necessário no decorrer das atividades. O auxílio dos colegas bolsistas em atividade no LIN (Laborátorio de Instumentação) do DAS (Departamento de Automação e Sistemas) foi fundamental para o desenvolvimento dos projetos.

# REFERÊNCIAS

CAMACHO, E. F.; ALBA, C. B. *Model Predictive Control*. London: Springer, 2013. 405 p.

CLARKE, D. W.; MOHTADI, C.; TUFFS, P. S. Generalized predictive control - part i. the basic algorithm. *Pergamon Journals Ltd.*, v. 23, n. 2, p. 137–148, 1987.

FLYNN, M. J. Basic issues in microprocessor architecture. *Journal of Systems Architecture*, v. 45, n. 1, p. 939–948, 1999.

NI. *NI Digital Eletronics FPGA Board: User Manual*. Austin, Texas, 2009.

NI. *Example Programs for the NI Digital Electronics FPGA Board and NI ELVIS II*. Março 2011. National Instruments. <http://www.ni.com/example/8857/en>. Acessado em Setembro de 2015.

NI. *LabVIEW Courseware for the NI Digital Electronics FPGA Board and NI ELVIS II*. Março 2011. National Instruments White Paper. <http://www.ni.com/white-paper/8856/en>. Acessado em Agosto de 2015.

NI. *Where to Start with the NI ELVIS II Series*. Austin, Texas, 2011.

NI. *FPGA Fundamentals*. Maio 2012. National Instruments White Paper. <http://www.ni.com/white-paper/6983/en>. Acessado em Setembro de 2015.

NI. *LabVIEW: Getting Started with LabVIEW*. Austin, Texas, 2013.

NI. *NI myRIO: Design Real Systems, Fast*. Official Manual for the "Learn to Design Real Systems Fast with NI myRIO" Hands-On Workshop, 2013.

NI. *Getting Started With LabVIEW FPGA*. Novembro 2014. National Instruments. <http://www.ni.com/tutorial/14532/en>. Acessado em Julho 2015.

PARNELL, K.; BRYNER, R. *Comparing and Contrasting FPGA and Microprocessor System Design*

*and Development*. Julho 2004. Xilinx White Paper. <http://www.xilinx.com/support/documentation/whitepapers/wp213.pdf>. Acessado em Maio 2016.