UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
COORDENAÇÃO DE ENGENHARIA ELETRÔNICA
CURSO DE ENGENHARIA ELETRÔNICA

AMANDA BARRETO DO NASCIMENTO

DESENVOLVIMENTO DE UM CONTROLADOR DIGITAL PARA CONVERSORES *BUCK* VOLTADO PARA UTILIZAÇÃO EM LABORATÓRIOS DE ENSINO

TRABALHO DE CONCLUSÃO DE CURSO

TOLEDO
2018

AMANDA BARRETO DO NASCIMENTO

DESENVOLVIMENTO DE UM CONTROLADOR DIGITAL PARA CONVERSORES *BUCK* VOLTADO PARA UTILIZAÇÃO EM LABORATÓRIOS DE ENSINO

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná - UTFPR Campus Toledo, como requisito parcial para a obtenção do título de Bacharel em Engenharia Eletrônica.

Orientador: Prof. Dr. Felipe Walter Dafico Pfrimer
Universidade Tecnológica Federal do Paraná

TOLEDO
2018

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Toledo
Coordenação do Curso de Engenharia Eletrônica

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

TERMO DE APROVAÇÃO

Título do Trabalho de Conclusão de Curso Nº 70

DESENVOLVIMENTO DE UM CONTROLADOR DIGITAL PARA CONVERSORES *BUCK* VOLTADO PARA UTILIZAÇÃO EM LABORATÓRIOS DE ENSINO

por

AMANDA BARRETO DO NASCIMENTO

Esse Trabalho de Conclusão de Curso foi apresentado às **9h30 do dia 27 de Junho de 2018** como **requisito parcial** para a obtenção do título de **Bacharel em Engenharia Eletrônica**. Após deliberação da Banca Examinadora, composta pelos professores abaixo assinados, o trabalho foi considerado **APROVADO**.

Prof. Dr. Alberto Yoshihiro Nakano
Instituição

Prof. Ruahn Fuser
Instituição

Prof. Dr. Felipe Walter Dafico Pfrimer
UTFPR

Prof. Dr. Fábio Rizental Coutinho
UTFPR

O termo de aprovação assinado encontra-se na coordenação do curso

Toledo, 27 de Junho de 2018

AGRADECIMENTOS

Em primeiro lugar a Deus, não por um costume, ou por formalidade, mas por realmente ser agradecida pela vida e pelo sustento até aqui.

Aos meus pais por me guiarem até aqui com carinho e sabedoria, por serem meus maiores exemplos de vida, de estudiosos e grandes incentivadores da minha vida acadêmica. Agradeço por nunca terem medido esforços para a minha formação e por terem me dado os subsídios para que eu pudesse conquistar os meus sonhos.

Aos meus familiares sempre presentes nas minhas conquistas e essenciais no meu crescimento.

Agradeço à UTFPR, que por muitos dias se tornou minha casa, por me proporcionar crescimento pessoal e profissional, e por me permitir subir mais um degrau na minha vida acadêmica.

Aos meus professores que me acompanharam ao longo dessa jornada, em especial ao meu orientador Felipe Walter Dafico Pfrimer pela atenção e respeito dedicados a mim, e por estar sempre disposto a me receber e sanar as minhas dúvidas, desde as disciplinas anteriores ao TCC.

Aos meus amigos, os de longe, os de perto, os de ontem, os de hoje, e os de sempre, que aqui estão representados de forma genérica, mas no meu coração cada um tem um lugar reservado. Em especial, não poderia deixar de agradecer às amigas Giovana, Hikari e Taiany, que compartilharam comigo histórias, trabalhos, madrugadas de estudo e momentos de descontração ao longo da graduação e aos meus amigos Ademilson, Jéssica e Juliana, que mesmo de longe me incentivam e me encorajam a seguir em frente.

Aos que já se foram, mas que contribuíram para o que sou hoje, e de quem sentirei eternas saudades.

A menos que modifiquemos a nossa maneira de pensar, não seremos capazes de resolver os problemas causados pela forma como nos acostumamos a ver o mundo. (Albert Einstein)

RESUMO

Este projeto visa o desenvolvimento de uma ferramenta educacional voltada para o ensino de eletrônica de potência e programação de dispositivos lógicos programáveis. Trata-se de um sistema de controle para o conversor CC-CC (corrente contínua para corrente contínua) do tipo *Buck* ou *step-down*, onde a tensão de saída é menor ou igual à tensão de entrada. No âmbito da eletrônica de potência, o dispositivo proporcionará a possibilidade do aluno visualizar o comportamento da tensão de saída do conversor quando operando em malha aberta ou fechada. Visto que será utilizado um FPGA em sua construção, essa ferramenta também proporcionará à estudantes de disciplinas de linguagem de descrição de hardware (HDL), a oportunidade de verificar a estrutura da linguagem e seu funcionamento. Ademais, haverá a possibilidade de editar as constantes de um controlador PID (proporcional, integral, e derivativo) e visualizar os efeitos inerentes dos parâmetros sobre o sistema controlado.

Palavras-chave: PID Digital. Ferramenta Educacional. Conversor CC-CC. FPGA

ABSTRACT

This project develops an educational tool to help teaching power electronics and programming of programmable logic devices. It is a control system for the DC-DC converter, Buck (or step-down) topology, in which the output voltage is less than or equal to the input voltage . In the field of power electronics, the device will provide the possibility to visualize the output voltage behavior of the converter when operating in open or closed loop. Since an FPGA will be used in its construction, this tool will also provide students of hardware description language (HDL) disciplines with the opportunity to check the structure of the language and its operation. In addition, it will be possible to edit the constants of a PID controller (integral, proportional, and derivative) and visualize the inherent effects of the parameters on the controlled system.

Keywords: Digital PID. Educational Tool. DC-DC converter. FPGA

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

$\overline{CS}$	*Chip-Select Input*
$\overline{U}$/B	*Unipolar/Bipolar Input*
A/D	Analógico/Digital
AIN1A	*Primary/Positive Analog Input Channel 1*
AIN1B	*Secondary/Negative Analog Input Channel 1*
AIN2A	*Primary/Positive Analog Input Channel 2*
AIN2B	*Secondary/Negative Analog Input Channel 2*
CC-CC	Corrente Contínua - Corrente Contínua
CNVST	*Conversion-Start Input*
CPLDs	*Complex Programmable Logic Devices*
D/A	Digital/Analógico
DOUT1	*Serial-Data Output 1*
DOUT2	*Serial-Data Output 2*
FPGA	*Field-Programmable Gate Array*
GAL	*Generic Array Logic*
HDL	*Hardware Description Language*
JTAG	*Joint Test Action Group*
NMOSFET	*N-channel Metal Oxide Semiconductor Field Effect Transistor*
PAL	*Programmable Array Logic*
PC	*Personal Computer*
PCB	*Printed Circuit Boards*
PLA	*Programmable Logic Array*
PLDs	*Programmable Logic Devices*
PWM	*Pulse-Width Modulation*
REFSEL	*Reference-Select Input*
S/$\overline{D}$	*Single-Output/Dual-Output Selection Input*
SCLK	*Serial-Clock Input*
SEL	*Analog-Input Selection Input*
SPI	*Serial Peripheral Interface*
SPLDs	*Simple Programmable Logic Devices*
UPS	*Uninterruptible Power Supplies*
VHDL	*VHSIC Hardware Description Language*
VHSIC	*Very High Speed Integrated Circuits*

LISTA DE SÍMBOLOS

$e(t)$	Sinal de erro do PID no domínio contínuo
K_P	Constante da ação proporcional do PID
K_I	Constante da ação integral do PID
K_D	Constante da ação derivativa do PID
$u(t)$	Sinal de saída do PID no domínio contínuo
$u(kT_s)$	Sinal de saída do PID no domínio discreto
$e(kT_s)$	Sinal de erro no domínio discreto
$u_P(kT_s)$	Parcela proporcional do sinal de saída do PID no domínio discreto
$u_I(kT_s)$	Parcela integral do sinal de saída do PID no domínio discreto
$u_D(kT_s)$	Parcela derivativa do sinal de saída do PID no domínio discreto
$U_P(z)$	Parcela proporcional do sinal de saída do PID no domínio Z
$E(z)$	Sinal de erro do PID no domínio Z
$D_P(z)$	Função de transferência da parcela proporcional do PID no domínio Z
$U_I(z)$	Parcela integral do sinal de saída do PID no domínio Z
$D_I(z)$	Função de transferência da parcela integral do PID no domínio Z
$U_D(z)$	Parcela derivativa do sinal de saída do PID no domínio Z
$D_D(z)$	Função de transferência da parcela derivativa do PID no domínio Z
$D(z)$	Função de transferência do PID no domínio Z
T_s	Período de chaveamento do conversor *Buck*
T_{ON}	Período que a chave do conversor *Buck* está fechada
T_{OFF}	Período que a chave do conversor *Buck* está aberta
δ	Ciclo de trabalho
L	Indutor
C	Capacitor
R_L	Carga do Conversor *Buck*
v_o	Tensão instantânea de saída do conversor *Buck*
i_i	Corrente instantânea na fonte do conversor *Buck*
i_D	Corrente instantânea no diodo do conversor *Buck*
$i_{indutor}$	Corrente instantânea no indutor do conversor *Buck*
V_o	Tensão média na saída do conversor *Buck*
P_i	Potência média da entrada do conversor *Buck*
P_o	Potência média na saída do conversor *Buck*
$P_{R_{indutor}}$	Perda no indutor do conversor *Buck*
V_i	Tensão média na entrada do conversor *Buck*
I_i	Corrente média na entrada do conversor *Buck*
$R_{indutor}$	Resistência do indutor do conversor *Buck*

$I_{indutor}$	Corrente média no indutor do conversor *Buck*
$i_{indutor}$	Corrente instantânea do indutor
$v_{indutor}$	Tensão instantânea no indutor do conversor *Buck*
R_C	Resistência série do capacitor
$\mathrm{v_C}$	Tensão sobre o capacitor
$\mathbf{x}(t)$	Vetor de Estados
$\mathbf{A}$	Matriz de Estado
$\mathbf{B}$	Matriz de Entrada
$\mathbf{C}$	Matriz de Saída
$\mathbf{D}$	Matriz de Transmissão Direta
$\mathbf{z}(t)$	Vetor de Entradas
$\mathbf{y}(t)$	Vetor de Saídas
$T(s)$	Função de Transferência do Conversor *Buck*

SUMÁRIO

1 INTRODUÇÃO

A Engenharia Eletrônica está intimamente ligada aos avanços tecnológicos, e para que os novos conceitos sejam compreendidos ou concebidos, faz-se necessário que o entendimento de determinados princípios básicos estejam bem consolidados na mente dos engenheiros.

Considera-se, neste trabalho, a importância dos alunos se envolverem com o conhecimento ofertado no ambiente acadêmico, visto que o aluno é o interessado em reelaborá-lo, até que o mesmo faça sentido para sua formação acadêmica, assim como para sua formação profissional. Pressupõe-se assim, uma necessidade aumentada de se estabelecer uma associação entre a teoria e a prática, capaz de colocar professores e alunos numa dinâmica interação em busca da produção do conhecimento, na qual não se prioriza, nem se polariza teoria e prática, conforme pontua Noronha (2005)

> No conceito de Marx encontramos um homem criador, que não só recebe impressões, mas que também as elabora, as interpreta, correlaciona, antecipa acontecimentos, esboça imagens e conceitos de objetos por produzir-se, cria formas e relações simbólicas para aplicações futuras.

O homem criador, abordado em Noronha (2005), pode ser considerado como o aluno, que ao apropriar-se do conhecimento elaborado, não só o reproduz, como pode ampliá-lo, a fim de satisfazer às novas demandas. Para Marx, "O processo de trabalho [...] é atividade orientada a um fim para produzir valores de uso, apropriação do natural para satisfazer a necessidades humanas [...]." (MARX, 1985). No âmbito deste projeto, vislumbram-se "necessidades humanas" como os avanços tecnológicos. O que fazer para atender a essas necessidades que também atingem a forma de ensinar?

Para Keski (2012) "a presença de uma determinada tecnologia pode induzir profundas mudanças na maneira de organizar o ensino". Fundamentado nas ponderações feitas, o presente trabalho se propôs a desenvolver uma ferramenta educacional, um kit didático, que explora os conceitos abordados nas áreas de eletrônica de potência, controle e principalmente, programação de dispositivos lógicos programáveis (PLDs). Com a utilização desse kit, os alunos podem ver como cada componente do sistema se comporta de acordo com as configurações aplicadas, por eles mesmos. Essa ferramenta possui o propósito de auxiliar não só alunos, mas também professores, na medida que este recurso possibilita a junção entre a teoria e a prática, o que poderá promover uma maior eficiência no processo de ensino e aprendizagem.

Para o kit didático ser concretizado, foi desenvolvido um sistema de controle para um conversor *Buck*, um conversor CC-CC, que tem como característica fornecer em sua saída uma tensão de nível menor àquela que foi aplicada em sua entrada. O conversor *Buck* opera com base em um dispositivo que realiza o chaveamento do circuito de acordo com o seu ciclo de trabalho. Portanto, foi utilizado um controle que que atua nesse valor do ciclo de trabalho, de acordo com as variações da entrada e de carga que o sistema venha a sofrer, ou seja, o mesmo funciona em malha fechada ou realimentado.

O controle do sistema foi feito através de um controlador PID (Proporcional Integral Derivativo), que fornece uma resposta mais rápida, reduz o sobressinal e o erro. O PID foi modelado de forma a ser implantado em um FPGA, portanto, utilizou-se para tal, a linguagem VHDL (do inglês *VHSIC Hardware Description Language*, sendo VHSIC também oriunda do inglês *Very High Speed Integrated Circuits*).

Além do PID, fazem parte deste trabalho outros componentes descritos em VHDL, como o controle do conversor ADC, que lê e converte a tensão de saída do conversor *Buck* para digital, e o PWM, responsável por repassar o *duty cycle* e a frequência de operação para o circuito. Sendo assim, busca-se que o aluno consolide a compreensão da linguagem por meio da interação (realizando alterações nos códigos), e da observação da estrutura de todo o sistema. Ademais, objetiva-se que os códigos aqui dispostos possam ser utilizados em outras aplicações, visto que são componentes amplamente utilizados na área da Engenharia Eletrônica.

Este trabalho possui 8 capítulos, mais os Apêndices. O primeiro capítulo é a Introdução e os Capítulos 2, 3 e 4 tratam dos objetivos, da justificativa e da abordagem teórica para realização do projeto, respectivamente. Nesse último foram buscados autores de referência nas áreas tratadas, e as seguintes teorias estão expostas: controle digital, com ênfase no PID digital; conversores CC-CC, mais especificamente o conversor *Buck*; modulação de largura de pulso; e por fim, os dispositivos lógicos programáveis, nos quais o FPGA está incluso. No Capítulo 5 estão descritos os materiais que foram utilizados, e no Capítulo 6 estão discriminadas as etapas da metodologia seguida. A análise e discussão dos resultados encontram-se no Capítulo 7, enquanto que a conclusão do trabalho, com sugestões para trabalhos futuros e as considerações finais, encontra-se no Capítulo 8. Por fim, nos Apêndices, estão expostos os códigos em VHDL de todo o sistema sintetizado no FPGA.

2 OBJETIVOS

Este trabalho teve por objetivo desenvolver um kit didático que auxilie tanto professores na aplicação de aulas práticas, como alunos no entendimento de conceitos das áreas de Eletrônica de Potência e Programação de PLDs, principalmente. Dessa forma, desenvolveu-se um protótipo onde um FPGA foi utilizado para a programação de um controlador PID (Proporcional Integral e Derivativo) digital, aplicado na regulação da tensão de saída de um conversor CC-CC do tipo *Buck*.

Pode-se citar como objetivos específicos deste projeto:

- Projeto de um conversor *Buck* que opera no modo de condução contínuo (corrente no indutor nunca se anula), com base em parâmetros pré-estabelecidos;
- Construir, em uma placa de circuito impresso, o conversor *Buck*;
- Descrever, em linguagem VHDL, o controlador PID digital em um FPGA;
- Descrever em VHDL a leitura da tensão de saída do conversor *Buck* através do ADC;
- Descrever em VHDL o módulo PWM.

3 JUSTIFICATIVA

Nos ambientes de aprendizagem oferecidos pelas universidades, é preciso que teoria e pratica sejam aliadas, visto que tanto uma como a outra colaboram para o processo de apropriação do conhecimento dos alunos. De acordo com Saviani (2012), a prática deve ser tão coerente, consistente e desenvolvida quanto a teoria que a embasa, sendo assim, a prática está sendo pensada a partir da teoria.

Este trabalho consiste na criação de uma ferramenta didática que permite, através da prática, que os conceitos relacionados às áreas de controle, eletrônica analógica, eletrônica de potência e sistemas digitais sejam fixados. Com essa ferramenta pretende-se demonstrar, na prática, a estrutura de um sistema modelado em linguagem VHDL e a influência das alterações que podem vir a ser feitas no próprio código, nos parâmetros de controle, nos valores dos componentes do conversor *Buck*, e na configuração do PWM na resposta do sistema.

O autor afirma ainda que o movimento inverso é igualmente necessário, ou seja, a teoria pensada a partir da prática. Essa ferramenta educacional fornece subsídios para que o aluno, após a consolidação dos conhecimentos envolvidos neste trabalho, amplie essa teoria, dado que os códigos aqui dispostos podem ser reescritos para atender novas necessidades, ou as modelagens podem ser refeitas para que abranjam mais variáveis.

Com base nessas considerações, explica-se a motivação para este trabalho, que reside no fato de propiciar ao aluno a exploração do conhecimento apreendido em sala de aula, de uma forma livre, visual e tangível.

4 REFERENCIAL TEÓRICO

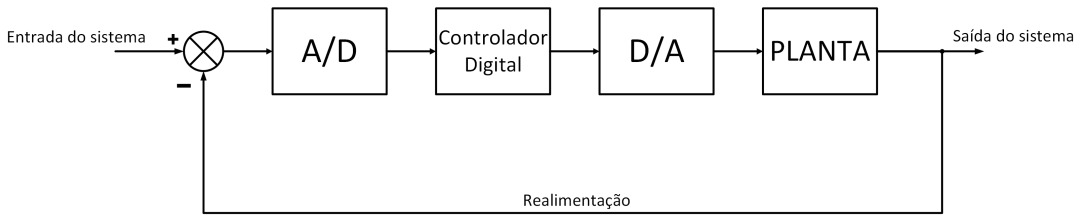
A teoria na qual o projeto se baseia abrange diversas áreas da Engenharia Eletrônica, tais como: Controle, Eletrônica Analógica, Eletrônica de Potência e Sistemas Digitais. Na Seção 4.1 serão apresentados os conceitos que envolvem controladores digitais, mais especificamente o controlador PID, amplamente utilizado para melhorar o desempenho de diversos sistemas de controle, e que será modelado e implantado em um FPGA, dispositivo que será explicado na Seção 4.4. O conversor *Buck*, sistema a ser controlado, é uma das topologias dos conversores chaveados de corrente contínua para corrente contínua (CC-CC), e será descrito na Seção 4.2. A Seção 4.3 é referente à Modulação por Largura de Pulso (PWM).

4.1 CONTROLADORES DIGITAIS

Um sistema de controle, de uma forma simples e geral, é composto por um controlador (também chamado de compensador), e uma planta (o que se deseja controlar), além de outros componentes que tornam todo o processo possível. Segundo Nise (2013), é pautado em três objetivos principais: produzir uma resposta transitória adequada, reduzir os erros no estado estacionário e proporcionar estabilidade.

Segundo Chen (1993), grande parte das plantas de controle são analógicas, conforme é o caso deste projeto, porém o controlador analógico é preterido em relação ao controlador digital, devido à acurácia, flexibilidade e confiabilidade desse último. Como pode ser observado na Figura 1, a qual representa um sistema de controle com realimentação, a comunicação entre o compensador e a planta é feita através de conversores analógico/digital (A/D) e digital/analógico (D/A), responsáveis por realizarem as conversões dos sinais analógicos em digitais e digitais em analógicos, respectivamente.

Figura 1 – Diagrama de blocos de um sistema de controle digital.



Fonte: Adaptado de Nise (2013).

O controlador digital do tipo PID, que será utilizado neste projeto, é um controlador amplamente empregado na indústria por reunir três formas de atuação, a proporcional, a integral, e a derivativa. Cada uma dessas tem uma função no sistema a ser controlado, conforme será explicado na Subseção 4.1.1.

4.1.1 CONTROLADOR PID

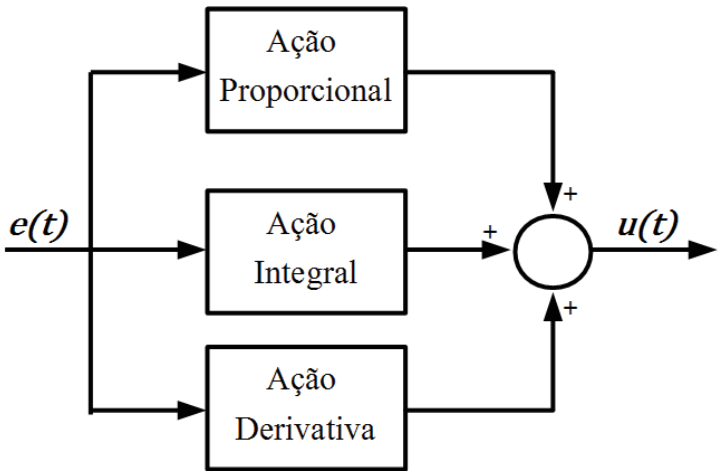
Conforme dito anteriormente, o controlador PID é composto por três ações: a proporcional, a integral e a derivativa. O controle proporcional possui a função de multiplicar o sinal de erro, $e(t)$, pela constante K_P, o que tem o efeito de tornar mais rápida a resposta do sistema. O controle integral reduz o erro no regime permanente, multiplicando a constante K_I pela integral do sinal $e(t)$ no tempo. Enquanto a ação derivativa reduz o sobressinal e as oscilações no regime transitório, através também da multiplicação de uma constante, nesse caso K_D pela derivada no tempo do sinal $e(t)$ (KUO, 1992).

Após as considerações feitas, pode-se escrever a expresão do sinal de saída, para um controlador PID de dados contínuos, da seguinte forma:

$$u(t) = K_P\, e(t) + K_I \int_0^t e(t)\, dt + K_D\, \frac{de(t)}{dt}, \tag{1}$$

onde a saída $u(t)$ é resultado da soma das ações das três parcelas do controlador PID no sinal de erro $e(t)$, conforme ilustra a Figura 2.

Figura 2 – Diagrama de blocos do controlador PID no modo contínuo



Fonte: Adaptado de Starr (2006).

Para um controlador PID digital será feita a mesma análise acima, porém considerando o domínio discreto e para isso a transformada Z será utilizada, conforme o desenvolvimento apresentado em Starr (2006). Os sinais da saída do controlador $u(t)$ e o sinal de erro $e(t)$ serão representados, no domínio discreto, por $u(kT_s)$ e $e(kT_s)$, respectivamente, onde $k = 1,2,3,...$, e T_s é o período de amostragem.

A fim de facilitar o entendimento, o sinal de saída $u(kT_s)$ terá cada uma de suas parcelas definidas separadamente, sendo elas: $u_P(kT_s)$ (ação proporcional), $u_I(kT_s)$ (ação integral) e $u_D(kT_s)$ (ação derivativa). Feitas tais considerações, a saída do controlador pode ser escrita como:

$$u(kT_s) = u_P(kT_s) + u_I(kT_s) + u_D(kT_s). \tag{2}$$

Na análise discreta, o controle proporcional continua sendo implementado por uma constante proporcional:

$$u_P(kT_s) = K_P\, e(kT_s). \tag{3}$$

Sendo $U_P(z)$ e $E(z)$ as transformadas Z de $u_P(kT_s)$ e $e(kT_s)$, respectivamente, tem-se que:

$$\frac{U_P(z)}{E(z)} = K_P. \tag{4}$$

Portanto, a função de transferência, $D_P(z)$, do controlador proporcional é:

$$D_P(z) = K_P. \tag{5}$$

Para o controle integral, a expressão no modo discreto é:

$$u_I(kT_s) = u_I(kT_s - T_s) + K_I\, e(kT_s). \tag{6}$$

É possível observar da Equação (6), que o sinal $u_I(kT_s)$ está deslocado em uma unidade de tempo de amostragem. Sendo assim, a transformada desse sinal será o sinal $U_I(z)$ multiplicado por z^{-1}:

$$U_I(z) = U_I(z)\, z^{-1} + K_I\, E(z). \tag{7}$$

Portanto, a função de transferência, $D_I(z)$, do controlador integral é:

$$\frac{U_I(z)}{E(z)} = D_I(z) = K_I\, \frac{z}{z-1}. \tag{8}$$

Para o controle derivativo, sabe-se que no domínio discreto uma diferenciação pode ser aproximada por uma equação a diferenças, ou seja,

$$\frac{de(t)}{dt} \Leftrightarrow e(kT_s) - e(kT_s - T_s), \tag{9}$$

portanto,

$$u_D(kT_s) = K_D\ \left\{e(kT_s) - e(kT_s - T_s)\right\}. \tag{10}$$

É possível observar da Equação (10) que há um sinal $e(kT_s)$ deslocado de um período de amostragem para a direita. Sendo assim, a transformada desse sinal será o sinal $E(z)$ multiplicado por z^{-1}, e $u_D(kT_s)$ torna-se $U_D(z)$:

$$U_D(z) = K_D\ \left(1 - z^{-1}\right)\ E(z). \tag{11}$$

Então, a função de transferência,$D_D(z)$, do controlador derivativo é

$$\frac{U_D(z)}{E(z)} D_D(z) = K_D\ \left(1 - z^{-1}\right). \tag{12}$$

Rearrumando os termos:

$$D_D(z) = K_D\, \frac{z-1}{z}. \tag{13}$$

A função de transferência do controlador PID, $D(z)$, será a soma de cada parcela demonstrada acima, ou seja, será a soma das Equações (5), (8) e (13), dada por:

$$D(z) = K_P + K_I \frac{z}{z-1} + K_D \frac{z-1}{z}. \tag{14}$$

Rearrumando os termos da Equação (14):

$$D(z) = \frac{(K_P + K_I + K_D) - (K_P + 2\,K_D)\; z^{-1} + K_D\, z^{-2}}{1 - z^{-1}}. \tag{15}$$

A Equação (15) não é muito útil na implementação do controlador PID digital, logo, faz-se necessário convertê-la em uma equação a diferenças, por meio da sua transformada inversa. Escrevendo-a em função da sua entrada $E(z)$ e saída $U(z)$:

$$U(z) - U(z)\, z^{-1} = E(z)[(K_P + K_I + K_D) - (K_P + 2\,K_D)\; z^{-1} + K_D\, z^{-2}]. \tag{16}$$

Considerando:

$$b_0 = K_P + K_I + K_D, \tag{17}$$

$$b_1 = -\left(K_P + 2\,K_D\right), \tag{18}$$

$$b_2 = K_D. \tag{19}$$

Aplicando a transformada Z inversa e as considerações das Equações (17), (18) e (19), na Equação (16):

$$u(kT_s) - u(kT_s - T_s) = b_0\, e(kT_s) + b_1\, e(kT_s - T_s) + b_2\, e(kT_s - 2T_s), \tag{20}$$

$$u(kT_s) = b_0\, e(kT_s) + b_1\, e(kT_s - T_s) + b_2\, e(kT_s - 2T_s) + u(kT_s - T_s). \tag{21}$$

Considerando n como o número da amostra, da Equação (21) tem-se:

$$u(n) = b_0\, e(n) + b_1\, e(n-1) + b_2\, e(n-2) + u(n-1). \tag{22}$$

Analisando a Equação (22), observa-se que para calcular a nova saída do controlador, faz-se necessário obter o sinal de saída no instante anterior, assim como os três últimos erros computados. A partir dessa equação, o PID digital pode ser implementado, pois a mesma encontra-se no domínio de tempo discreto, e seus coeficientes são facilmente calculáveis.

4.2 CONVERSORES CC-CC

Segundo Ahmed (2000), um conversor CC-CC, ou *chopper*, consiste em um dispositivo que transforma o valor de uma tensão CC de entrada em uma saída com valor ajustável. Esse circuito tem o seu funcionamento baseado na combinação de indutores e/ou capacitores, e de um dispositivo que realiza o chaveamento do circuito, o qual influencia a variação do valor médio da tensão de saída.

Pode-se citar alguns tipos básicos de conversores CC-CC que são os mais estudados: o conversor *Boost*, ou *step-up* (elevador), que fornece uma tensão de saída maior do que a inserida na entrada; o conversor *Buck-Boost*, onde a sua tensão de saída tem a polaridade invertida em relação à tensão de entrada, e sua magnitude pode ser maior, menor ou igual àquele da tensão aplicada na entrada; e o conversor *Buck*, ou *step-down* (abaixador), que terá seu funcionamento explicado, e sua topologia exposta na Subseção 4.2.1.

De acordo com Ahmed (2000) e Rashid (1999), os conversores CC-CC podem ser utilizados em diversas aplicações industriais como, por exemplo:

- controle de tração de motores CC em veículos elétricos: fornecem aceleração suave, resposta rápida e alta eficiência, ou seja, uma menor perda de energia;
- devolver energia à fonte de alimentação quando ocorre a frenagem regenerativa (processo no qual a energia cinética gerada pela frenagem é convertida em energia elétrica) em máquinas de corrente contínua;
- reguladores de tensão de uso geral;
- fontes de alimentação ininterrupta (do inglês *Uninterruptible Power Supplies* (UPS), também conhecido com *nobreak*);
- em equipamentos operados por baterias, entre outros.

4.2.1 CONVERSOR *BUCK*

Para melhor explicar o funcionamento do conversor *Buck*, nessa seção será adotada uma notação para as correntes e tensões, de acordo com os seguintes critérios:

- Correntes e tensões instantâneas são representadas por letras minúsculas, em itálico, seguidas por uma letra ou palavra subscrita que indicam a qual parte do circuito pertence a corrente ou tensão. Por exemplo, $i_o(t)$ representa a corrente instantânea na carga, $i_{indutor}(t)$ representa a corrente instantânea no indutor, $v_i(t)$ representa a tensão instantânea na entrada, entre outros.
- Correntes e tensões médias são representadas por letras maiúsculas, em itálico, seguidas por uma letra ou palavra subscrita que indicam a qual parte do circuito pertence a corrente ou tensão. Por exemplo, I_o representa a corrente média na carga, $I_{indutor}$ representa a corrente média no indutor, V_i representa a tensão média na entrada, entre outros.

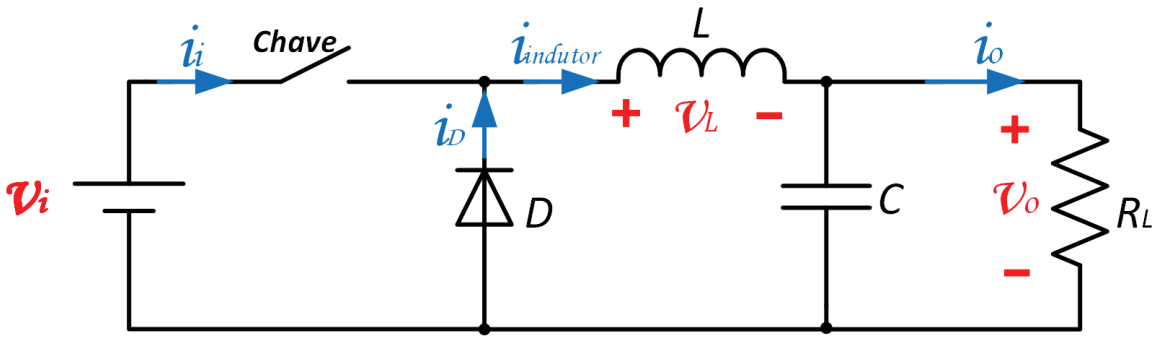
O conversor CC-CC do tipo *Buck* é um abaixador de tensão (*step-down*), ou seja, a tensão média gerada em sua saída é menor do que a que foi aplicada em sua entrada. Segundo Rashid (1999) o funcionamento do circuito de um conversor *Buck*, retratado na Figura 3, pode ser estudado em dois momentos, um quando a chave está fechada e outro quando está aberta, sendo essa chave uma forma genérica de representar um transistor, ou qualquer outra chave de potência.

O controle do chaveamento do *chopper* pode ser feito através da modulação por largura de pulso (PWM). PWM é um método no qual o tempo que a chave está fechada varia, enquanto o período T_s (soma do tempo que a chave está fechada e aberta, T_{ON} e T_{OFF}, respectivamente),

permanece constante, e será explicado com mais detalhes, na seção 4.3. A fração de tempo na qual a chave permanece aberta é denominada ciclo de trabalho, δ, que é definido por:

$$\delta = \frac{T_{ON}}{T_s}. \tag{23}$$

Figura 3 – Conversor *Buck*



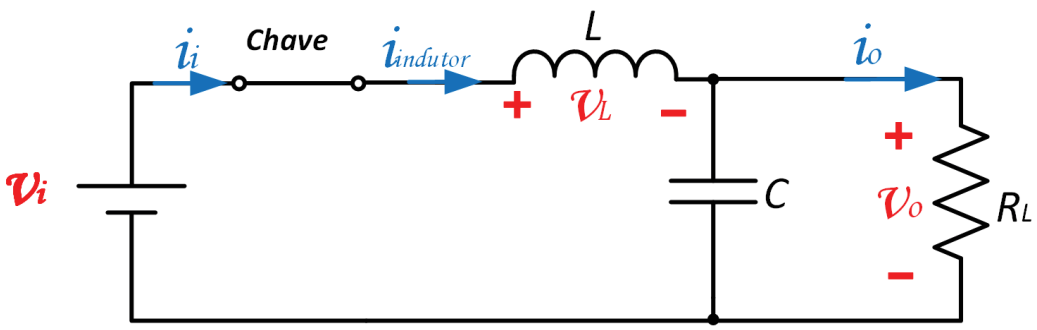
Fonte: Adaptado de Rashid (1999).

Ademais, de acordo com Ahmed (2000), esse circuito pode operar em dois modos distintos: no modo contínuo, quando a corrente no indutor nunca se anula; e no modo descontínuo, quando a mesma se anula momentaneamente dentro do período de chaveamento. Para este projeto o conversor *Buck* será modelado para operar apenas no modo de condução contínuo.

4.2.2 MODO DE CONDUÇÃO CONTÍNUO DO CONVERSOR *BUCK*

Ao observar a Figura 3, é possível deduzir que enquanto a chave estiver fechada, o diodo D ficará em corte, por estar inversamente polarizado, e o circuito equivalente é apresentado na Figura 4. Nesse circuito, a corrente na entrada crescerá exponencialmente percorrendo o indutor L, o capacitor C e a carga, representada pelo resistor R_L, onde se configura a saída do circuito.

Figura 4 – Circuito resultante do conversor *Buck* quando a chave está fechada.

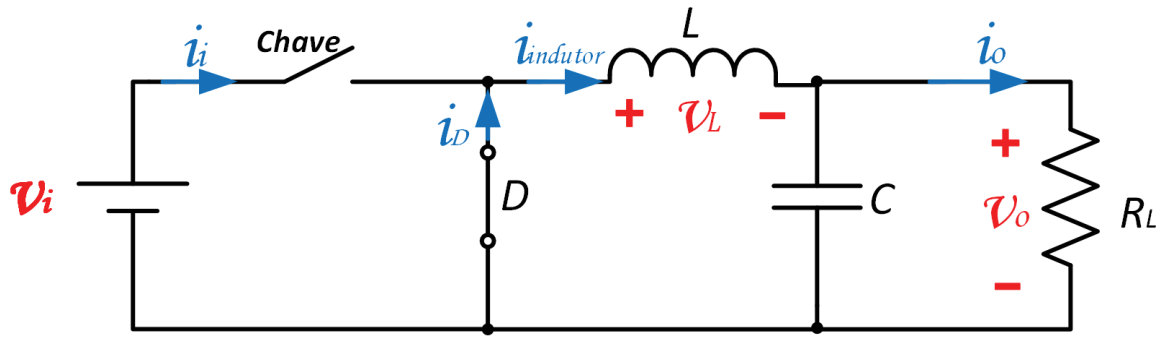


Fonte: Autoria Própria.

A partir do momento que a chave for aberta, conforme pode ser visto na Figura 5, não há fornecimento de energia proveniente da fonte de tensão para o restante do circuito, e o diodo

passará a conduzir devido à corrente armazenada no indutor, que irá continuar a fluir pelo indutor, capacitor, carga e diodo. Após um tempo, a corrente no indutor cai a um valor mínimo até que a chave ligue novamente, e assim o ciclo recomeça. Nesta análise, o diodo foi considerado como uma chave ideal, a fim de simplificar o entendimento do circuito.

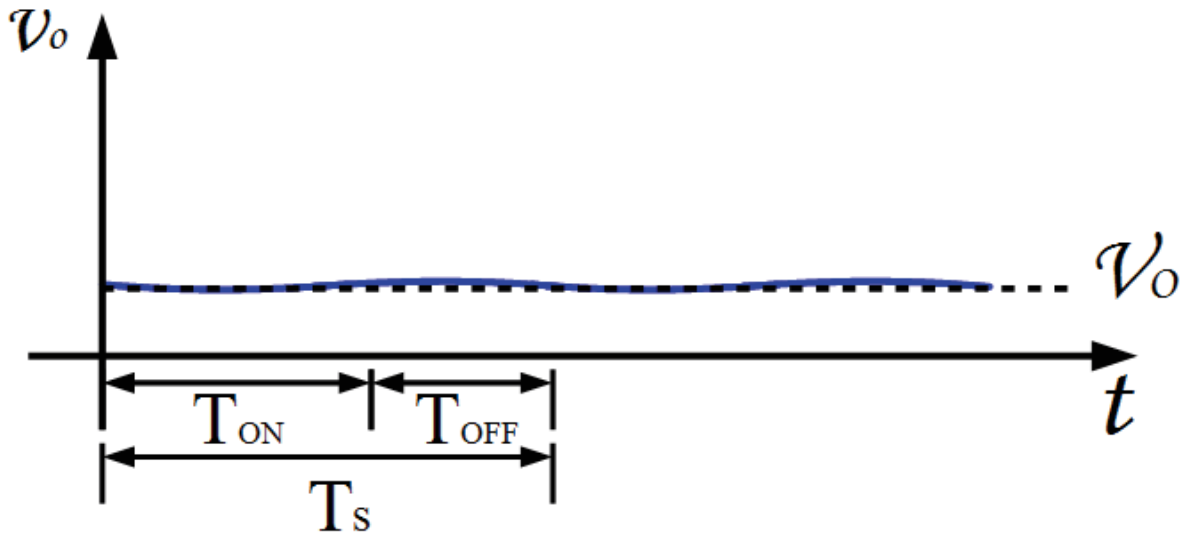
Figura 5 – Circuito resultante do conversor *Buck* quando a chave está aberta.



Fonte: Autoria Própria.

As Figuras 6, 7, 8 e 9 representam as formas de onda típicas da tensão na carga (v_o), da corrente na fonte (i_i), da corrente no diodo (i_D) e da corrente no indutor ($i_{indutor}$), respectivamente, quando o circuito opera no modo contínuo.

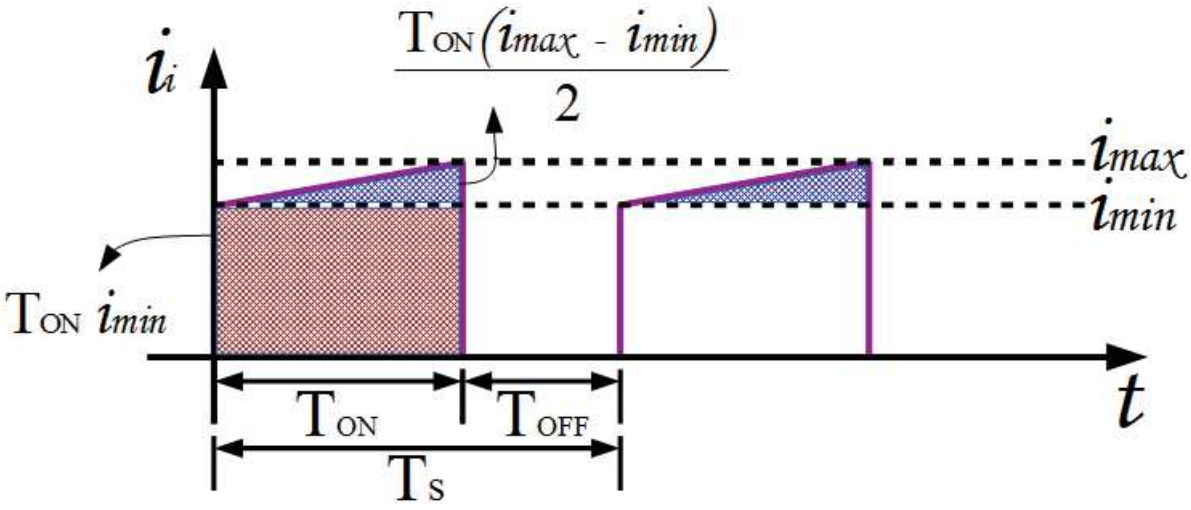
Figura 6 – Forma de onda da tensão na carga R_L, no modo contínuo.



Fonte: Autoria Própria.

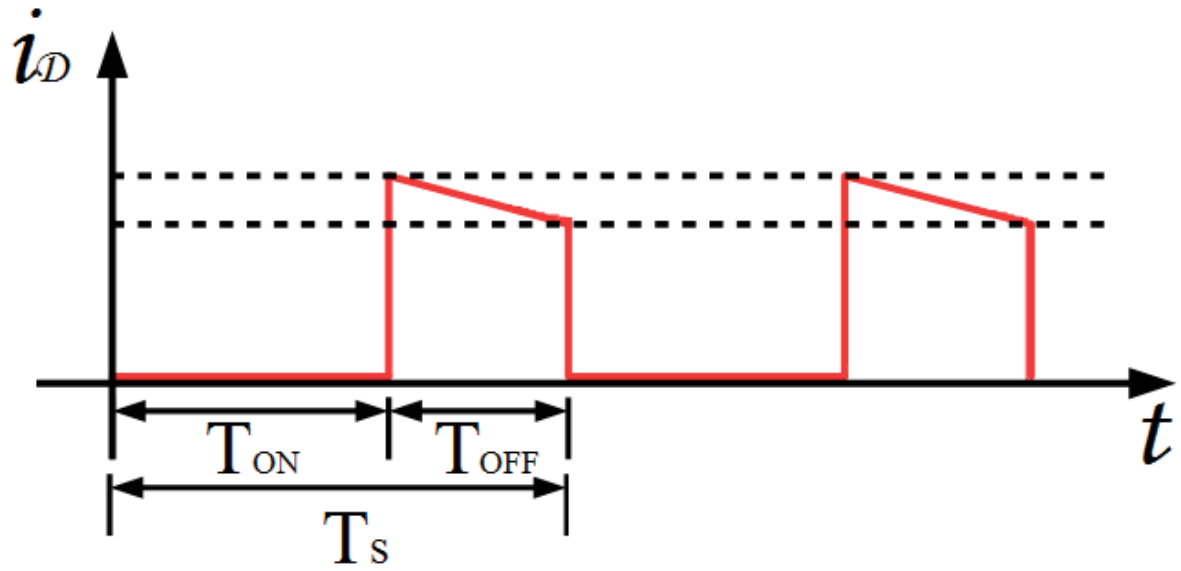
A Figura 6 apresenta o gráfico da tensão de saída v_o, onde pode ser visto que apresenta uma pequena ondulação, cuja amplitude está diretamente relacionada ao valor de C. A tensão média na saída, V_o, para um circuito com perdas no indutor e funcionando no modo contínuo,

Figura 7 – Forma de onda da corrente na fonte i_i no modo contínuo.



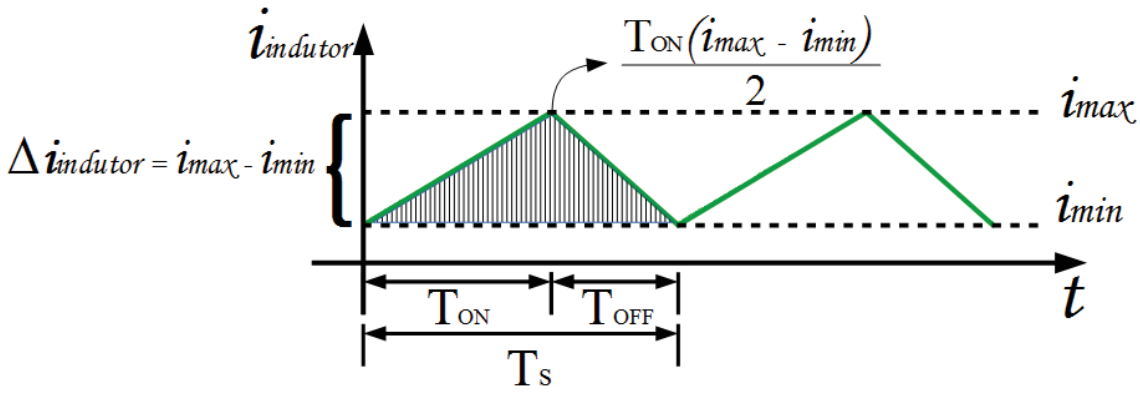
Fonte: Autoria Própria.

Figura 8 – Forma de onda da corrente no diodo i_D no modo contínuo.



Fonte: Autoria Própria.

Figura 9 – Forma de onda da corrente no indutor $i_{indutor}$ no modo contínuo.



Fonte: Autoria Própria.

pode ser deduzida a partir de:

$$P_i = P_o + P_{R_{indutor}}, \tag{24}$$

onde P_i, a potência média da entrada, é a soma da potência média na saída, representada por P_o, e da perda no indutor, $P_{R_{indutor}}$. Substituindo os termos da Equação (24) tem-se que:

$$V_i\ I_i = V_o\ I_o + R_{indutor}\ I^2_{indutor}, \tag{25}$$

sendo V_i e I_i a tensão e a corrente médias, respectivamente, na entrada; enquanto $R_{indutor}$ é a resistência do indutor, onde se configura a perda do circuito.

Através do cálculo da razão entre a área do gráfico da Figura 7 e o período T_s, é possível obter a expressão para a corrente média na entrada. Dessa forma, I_i pode ser calculada por:

$$I_i = \frac{1}{T_s}\left(\frac{i_{max} - i_{min}}{2}\ T_{ON} + i_{min}\ T_{ON}\right), \tag{26}$$

$$I_i = \frac{1}{T_s}\ \frac{i_{max} + i_{min}}{2}\ T_{ON}. \tag{27}$$

Sabe-se da Equação (23) que $T_{ON} = \delta T$, dessa forma:

$$I_i = \frac{(i_{max} + i_{min})\ \delta}{2}. \tag{28}$$

Ademais, através da análise da Figura 9 tem-se que:

$$I_{indutor} = \frac{i_{max} + i_{min}}{2}. \tag{29}$$

Portanto, substituindo a Equação (29) na Equação (28), obtém-se

$$I_i = I_{indutor}\ \delta. \tag{30}$$

A corrente média da carga será igual a do indutor ($I_{indutor}$), pois a corrente média no capacitor, considerando o regime chaveado do conversor, é sempre nula. Sendo assim, a corrente

média da entrada e da carga podem ser escritas em função de $I_{indutor}$, e a Equação (25) passa a ser:

$$V_i \; I_{indutor} \; \delta = V_o \; I_{indutor} + R_{indutor} \; I_{indutor}^2. \tag{31}$$

A partir da Figura 3, e da consideração feita acima a respeito da corrente média na carga e no indutor serem iguais, sabe-se que:

$$I_{indutor} = I_o = \frac{V_o}{R_L}. \tag{32}$$

Portanto, substituindo a Equação (32) na Equação (31), obtem-se a expressão da tensão média da saída para o modo de condução contínuo:

$$V_o = \frac{V_i \; \delta}{1 + \frac{R_{indutor}}{R_L}}. \tag{33}$$

Conforme dito anteriormente, o valor do indutor, e a frequência de chaveamento influenciam o modo no qual o conversor *Buck* estará operando. Portanto, a fim de encontrar um valor mínimo de indutância para que o conversor continue operando no modo de condução contínuo, para uma dada frequência de chaveamento ($f = \frac{1}{T_s}$), inicia-se uma análise a partir da variação de corrente no indutor ($\Delta i_{indutor}$).

A tensão $v_{indutor}$ no indutor L é calculada como:

$$v_{indutor} = L \frac{di_{indutor}}{dt}. \tag{34}$$

Como a variação da corrente é aproximadamente linear, $v_{indutor}$ pode ser considerada constante durante T_{OFF}, e igual a V_o. Dessa forma, a Equação (34) pode ser reescrita como:

$$\Delta i_{indutor} = \frac{V_o}{L} \; \Delta t. \tag{35}$$

Sabe-se, também da Figura 9, que:

$$\Delta i_{indutor} = i_{max} - i_{min}, \tag{36}$$

então,

$$i_{max} - i_{min} = \frac{V_o}{L} \; \Delta t. \tag{37}$$

Subtraindo a Equação (37) da Equação (29), e tomando $\Delta t = T_{OFF}$, ou seja, o tempo em que a chave está desligada, obtém-se:

$$i_{min} = \frac{V_o}{R_L} - \frac{V_o}{2 \; L} \; T_{OFF}. \tag{38}$$

Substituindo T_{OFF} na Equação (38), com base na Equação (23):

$$i_{min} = \frac{V_o}{R_L} - \frac{V_o}{2 \; L} \; (1 - \delta) \; T_s. \tag{39}$$

Para calcular o valor mínimo para o indutor, de forma que o circuito opere no modo contínuo, considera-se que a corrente na carga não pode se anular durante o período em que a chave estiver desligada, sendo assim, $i_{min} > 0$. Dessa forma, tem-se que:

$$L > \frac{R_L}{2} \ (1 - \delta) \ T_s. \tag{40}$$

A Inequação (40) é a expressão utilizada para calcular o valor mínimo do indutor, porém, na prática utiliza-se valores maiores do que o calculado a fim de se evitar que o sistema opere no limite do modo de condução contínuo.

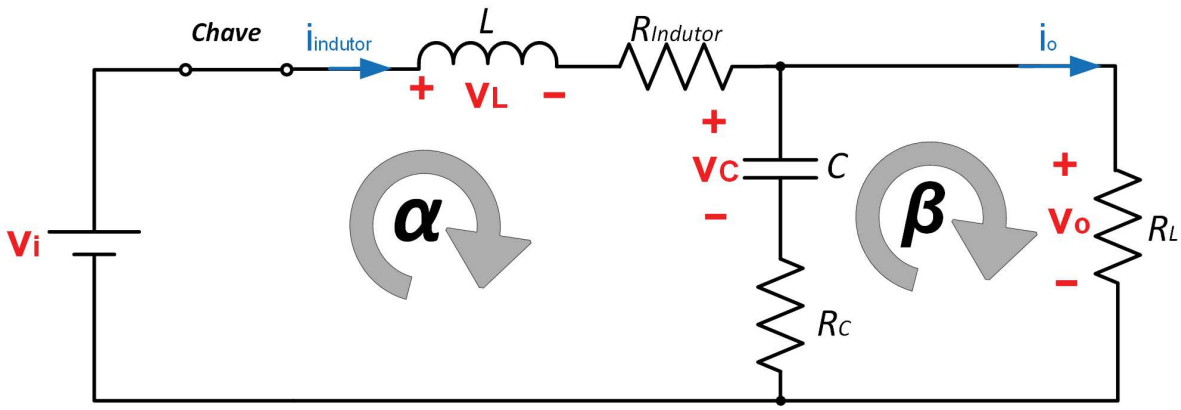
4.2.3 MODELAGEM CONVERSOR *BUCK*

Antes de iniciar a modelagem, é importante ressaltar que foi adotada a seguinte notação:

- A variável em maiúsculo, em itálico, indica a componente média. Por exemplo, V_o representa a tensão média de saída.
- A componente alternada é indicada pela letra minúscula em itálico. Por exemplo, v_o representa a componente alternada da tensão de saída
- A soma da componente média com a alternada é representada pela letra minúscula, não itálica. Por exemplo, $\mathrm{v_o}$ é a soma da tensão média com a componente alternada da saída.

Através da análise do circuito da Figura 10, e de modelagem matemática, utilizando o método das variáveis de estado, conforme Pomilio (2018), será obtida uma função de transferência com a qual o comportamento do conversor *Buck* poderá ser descrito e previsto.

Figura 10 – Conversor Buck operando com a chave fechada



Fonte: Autoria própria.

Considerando a malha α do circuito da Figura 10, onde a resistência série do capacitor é representada por R_C e $\mathrm{v_C}$ é a tensão sobre o capacitor, tem-se:

$$-\mathrm{v_i} + L \ \frac{d\mathrm{i_{indutor}}}{dt} + R_{Indutor} \ \mathrm{i_{indutor}} + \mathrm{v_C} + R_C \ C \ \frac{d\mathrm{v_C}}{dt} = 0. \tag{41}$$

Sabe-se que

$$\mathrm{v_C} + R_C \ C \ \frac{d\mathrm{v_C}}{dt} = \mathrm{v_o} = R_L \ \mathrm{i_o}, \tag{42}$$

e que pela lei dos nós

$$\mathrm{i_o} = \mathrm{i_{indutor}} - C\,\frac{d\mathrm{v_C}}{dt}. \tag{43}$$

Sendo assim, das Equações 41, 42, 43:

$$L\,\frac{d\mathrm{i_{indutor}}}{dt} - R_L\,C\,\frac{d\mathrm{v_C}}{dt} = \mathrm{v_i} - R_{Indutor}\,\mathrm{i_{indutor}} - R_L\,\mathrm{i_{indutor}}. \tag{44}$$

Da malha β do circuito da Figura 10:

$$\mathrm{v_o} - R_C\,C\,\frac{d\mathrm{v_C}}{dt} - \mathrm{v_C} = 0. \tag{45}$$

Substituindo as Equações 42 e 43 na Equação 45 tem-se:

$$(R_L + R_C)\,C\,\frac{d\mathrm{v_C}}{dt} = R_L\,\mathrm{i_{indutor}} - \mathrm{v_C}. \tag{46}$$

Após a obtenção das Equações 44 e 46, a modelagem do conversor será feita com base na teoria de espaço de estados. Segundo Ogata (2010), um estado é o menor conjunto de variáveis com as quais um sistema dinâmico pode ser descrito para qualquer instante após o início de seu funcionamento, a partir do conhecimento das mesmas no instante inicial, bem como do conhecimento da entrada para qualquer tempo. A essas variáveis dá-se o nome de variáveis de estado, componentes do vetor de estados, representado aqui por $\mathbf{x}(t)$, onde t refere-se ao tempo.

Para a análise no espaço de estados, o sistema de equações do sistema será representado na forma de matrizes, onde as matrizes **A**, **B**, **C** e **D** são denominadas matriz de estado, matriz de entrada, matriz de saída, e matriz de transmissão direta, respectivamente. As equações são na forma:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{z}(t), \tag{47}$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{z}(t), \tag{48}$$

sendo $\mathbf{z}(t)$ o vetor de entradas e $\mathbf{y}(t)$ o vetor de saídas.

Portanto, considerando essas informações, e adequando-as para o sistema do conversor Buck, tem-se que:

- As variáveis de estado são a corrente no indutor e a tensão no capacitor, representadas por $\mathbf{x_1}$ e $\mathbf{x_2}$, respectivamente, ou seja, $x_1 = \mathrm{i_{indutor}}$ e $x_2 = \mathrm{v_C}$.
- O vetor de entradas é $\mathrm{v_i}$.
- O vetor de saídas é $\mathrm{v_o}$.

Conforme abordado anteriormente, o conversor opera com a chave ligada e com a chave desligada, portanto para diferenciar, no equacionamento, tais instantes, serão utilizados os subíndices *ON* (chave ligada) e *OFF* (chave desligada) nas matrizes de estado, de entrada e de saída, sendo a matriz $\mathbf{D} = 0$ para todos os instantes.

Utilizando as Equações 47 e 48, e compreendendo os instantes nos quais a chave está ligada ou desligada:

$$\dot{\mathbf{x}} = [\mathbf{A}_{ON}\,\delta\ +\ \mathbf{A}_{OFF}\ (1\ -\ \delta)]\,\mathbf{x}\ +\ [\mathbf{B}_{ON}\,\delta\ +\ \mathbf{B}_{OFF}\ (1\ -\ \delta)]\ V_i, \tag{49}$$

$$\mathrm{v_o} = [\mathbf{C}_{ON}\,\delta\ +\ \mathbf{C}_{OFF}\ (1\ -\ \delta)]\,\mathbf{x}. \tag{50}$$

As variáveis instantâneas desse sistema: **x**, $\mathrm{v_o}$, δ e $\mathrm{v_i}$, podem ser separadas em componentes que representem perturbações (χ, v_o, d), e componentes que representam o valor médio (**X**, V_o, D e V_i) para a análise que se segue. Sendo assim:

$$\mathbf{x} = \mathbf{X} + \chi, \tag{51}$$

$$\mathrm{v_o} = V_o \ + \ v_o, \tag{52}$$

$$\delta = D \ + \ d, \tag{53}$$

$$\mathrm{v_i} = V_i + 0, \tag{54}$$

pois considera-se a tensão de entrada sem perturbações.

Transformando as Equações 44 e 46 para a forma matricial:

$$\begin{pmatrix} \dot{\mathrm{x}}_1 \\ \dot{\mathrm{x}}_2 \end{pmatrix} = \begin{pmatrix} -\frac{R_L\ (R_C+R_{Indutor}) \ + \ R_C\ R_{Indutor}}{L\ (R_C \ + \ R_L)} & -\frac{R_L}{L\ (R_C \ + \ R_L)} \\ -\frac{R_L}{C\ (R_C \ + \ R_L)} & -\frac{1}{C\ (R_C \ + \ R_L)} \end{pmatrix} \begin{pmatrix} \mathrm{x}_1 \\ \mathrm{x}_2 \end{pmatrix} + \begin{pmatrix} \frac{1}{L} \\ 0 \end{pmatrix} V_i \tag{55}$$

Para a saída:

$$\mathrm{v_o} = R_L\ (x_1 - C\ \dot{x}_2), \tag{56}$$

$$\mathrm{v_o} = R - L\ \left(x_1 - \frac{R_L\ x_1 \ - \ x_2}{R_C \ + \ R_L} \right) \tag{57}$$

$$\mathrm{v_o} = \frac{R_L\ R_C\ x_1 \ + \ R_L\ x_2}{R_C \ + \ R_L} \tag{58}$$

A matriz de saída **C** é igual para todos os instantes, assim como a matriz de estados **A**, enquanto que a matriz $\mathbf{B} = \mathbf{B}_{ON}\delta$, pois $\mathbf{B}_{OFF} = 0$ devido ao desligamento da chave. Portanto:

$$\mathbf{C} = \mathbf{C}_{ON} = \mathbf{C}_{OFF} = \begin{pmatrix} \frac{R_L\ R_C}{R_C \ + \ R_L} & \frac{R_L}{R_C \ + \ R_L} \end{pmatrix} \tag{59}$$

Considerando que $R_L >> R_C$, é possível simplificar as matrizes. Sendo assim, tem-se:

$$\mathbf{A} = \mathbf{A}_{ON} = \mathbf{A}_{OFF} = \begin{pmatrix} -\frac{(R_C \ + \ R_{Indutor})}{L} & -\frac{1}{L} \\ -\frac{1}{C} & -\frac{1}{C\ R_L} \end{pmatrix} \tag{60}$$

$$\mathbf{C} = \mathbf{C}_{ON} = \mathbf{C}_{OFF} \approx \begin{pmatrix} R_C & 1 \end{pmatrix} \tag{61}$$

$$\mathbf{B} = \begin{pmatrix} \frac{1}{L} \\ 0 \end{pmatrix} \ \delta \tag{62}$$

Substituindo as Equações 51, e 53 na Equação 49, considerando $\dot{\mathbf{X}} = 0$, e desconsiderando algumas parcelas de χ e d por serem muito pequenas:

$$\dot{\chi} = \mathbf{AX} \ + \ \mathbf{B}V_i \ + \mathbf{A}\chi \ + \ [(\mathbf{A}_{ON} \ - \ \mathbf{A}_{OFF})\ \mathbf{X} \ + \ (\mathbf{B}_{ON} \ - \ \mathbf{B}_{OFF})\ V_i]\ d. \tag{63}$$

De maneira análoga, porém analisando a saída:

$$V_o + v_o = \mathbf{C}\mathbf{X} \ + \ \mathbf{C}\chi \ + \ [(\mathbf{C}_{ON} \ - \ \mathbf{C}_{OFF})\,\mathbf{X}\,]\,d. \tag{64}$$

Em regime permanente, as perturbações e os termos dependentes do tempo são nulos, portanto das Equações (63) e (64) obtem-se a relação:

$$\frac{V_o}{V_i} = -\mathbf{C}\,\mathbf{A}^{-1}\,\mathbf{B}, \tag{65}$$

sendo $\mathbf{A}^{-1}$ a matriz inversa de **A**.

$$\mathbf{A}^{-1} = \begin{pmatrix} -\frac{L}{R_L + R_C + R_{Indutor}} & \frac{C\ R_L}{R_L + R_C + R_{Indutor}} \\ -\frac{L\ R_L}{R_L + R_C + R_{Indutor}} & \frac{C\ R_L\ (R_C + R_{Indutor})}{R_L + R_C + R_{Indutor}} \end{pmatrix}$$

Portanto,

$$-\frac{V_o}{V_i} = -\begin{pmatrix} R_C & 1 \end{pmatrix} \begin{pmatrix} -\frac{L}{R_L + R_C + R_{Indutor}} & \frac{C\ R_L}{R_L + R_C + R_{Indutor}} \\ -\frac{L\ R_L}{R_L + R_C + R_{Indutor}} & \frac{C\ R_L\ (R_C + R_{Indutor})}{R_L + R_C + R_{Indutor}} \end{pmatrix} \begin{pmatrix} \frac{1}{L} \\ 0 \end{pmatrix} D \tag{66}$$

$$\frac{V_o}{V_i} = \frac{(R_L \ + \ R_C)}{R_L \ + \ R_C \ + \ R_{Indutor}}\,D, \tag{67}$$

$$\frac{V_o}{V_i} \cong D. \tag{68}$$

Da componente alternada das Equações 63 e 64, tem-se:

$$\dot{\chi} = \mathbf{A}\chi \ + \ [(\mathbf{A}_{ON} \ - \ \mathbf{A}_{OFF})\,\mathbf{X} \ + \ (\mathbf{B}_{ON} \ - \ \mathbf{B}_{OFF}) \ V_i]\,d, \tag{69}$$

$$v_o = \mathbf{C}\chi \ + \ [(\mathbf{C}_{ON} \ - \ \mathbf{C}_{OFF})\,\mathbf{X}\,]\,d. \tag{70}$$

Aplicando a transformada de Laplace às Equações 69 e 70:

$$s\chi(s) = \mathbf{A}\chi\,(s) + \ [(\mathbf{A}_{ON} \ - \ \mathbf{A}_{OFF})\,\mathbf{X} \ + \ (\mathbf{B}_{ON} \ - \ \mathbf{B}_{OFF}) \ V_i]\,d(s), \tag{71}$$

$$v_o(s) = \mathbf{C}\chi(s) \ + \ [(\mathbf{C}_{ON} \ - \ \mathbf{C}_{OFF})\,\mathbf{X}\,]\,d(s). \tag{72}$$

ou ainda,

$$\chi(s) = [s\mathbf{I} \ - \ \mathbf{A}]^{-1} \ [(\mathbf{A}_{ON} \ - \ \mathbf{A}_{OFF})\,\mathbf{X} \ + \ (\mathbf{B}_{ON} \ - \ \mathbf{B}_{OFF}) \ V_i]\,d(s), \tag{73}$$

$$\chi = \frac{v_o(s) \ - \ [(\mathbf{C}_{ON} \ - \ \mathbf{C}_{OFF})\,\mathbf{X}\,] \ d(s)}{\mathbf{C}}, \tag{74}$$

onde **I** é a matriz identidade de mesma ordem da matriz **A**.

Das Equações 73 e 74, obtêm-se a função de transferência:

$$\mathrm{T}(s) = \frac{v_o(s)}{d(s)} = \mathbf{C}\,[s\mathbf{I} \ - \ \mathbf{A}]^{-1} \ [(\mathbf{A}_{ON} \ - \ \mathbf{A}_{OFF})\,\mathbf{X} \ + \ (\mathbf{B}_{ON} \ - \ \mathbf{B}_{OFF}) \ V_i] + (\mathbf{C}_{ON} \ - \ \mathbf{C}_{OFF})\,\mathbf{X}. \tag{75}$$

O vetor **X** é descrito como:

$$\mathbf{X} = \begin{pmatrix} I_{indutor} \\ V_C \end{pmatrix} = \begin{pmatrix} \frac{DV_i}{R_L} \\ DV_i \end{pmatrix}. \tag{76}$$

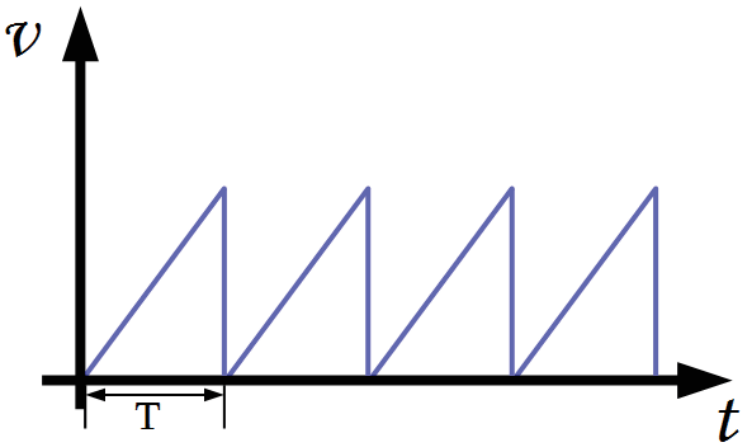
Portanto, aplicando os valores, encontrados anteriormente, das matrizes **A**, **B** e **C**, e do vetor **X** na Equação 75, tem-se a função de transferência do conversor *Buck*, $T(s)$:

$$\mathrm{T}(s) = \frac{v_o(s)}{d(s)} = V_i \cdot \frac{1 \ + \ s \ R_C \ C}{L \ C \ \left[s^2 \ + \ s \left(\frac{1}{R_L \ C} + \frac{R_C \ + \ R_I ndutor}{L} \right) \ + \ \frac{1}{L \ C} \right]}. \tag{77}$$

4.3 MODULAÇÃO POR LARGURA DE PULSOS

Do inglês *Pulse-Width Modulation*, PWM é um método utilizado para controle de velocidade de motores, de luminosidade, de servo motores, entre outros. Segundo Rashid (1999), a modulação consiste na comparação entre dois sinais de tensão, o de referência, e o da portadora. O sinal de referência, é uma imagem do sinal que se espera na saída, portanto no caso dos conversores CC-CC será um sinal de tensão contínua; para a portadora, nos conversores CC-CC geralmente utiliza-se um sinal "dente de serra" (Figura 11) e ela é a responsável pela definição da frequência de chaveamento, e do ciclo de trabalho δ. Essa comparação, realizada por um modulador, resulta em um sinal de acionamento alternado, com frequência fixa e largura de pulso variável, que depende da amplitude do sinal da portadora (Figura 12).

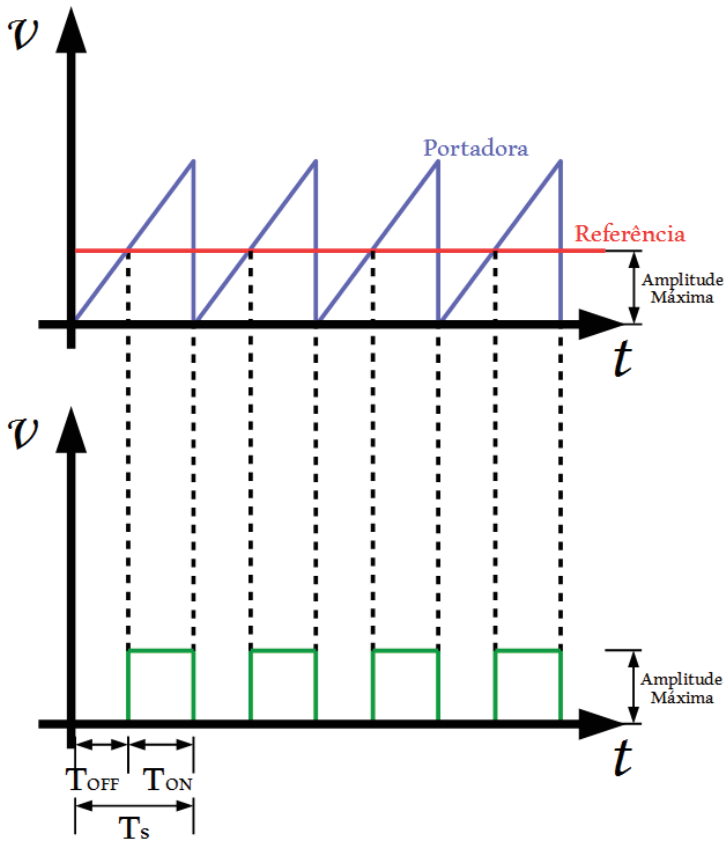
Figura 11 – Sinal "dente de serra".



Fonte: Autoria Própria.

Esses mesmos resultados podem ser obtidos utilizando circuitos digitais (microcontroladores, microprocessadores, entre outros), visto que os mesmos possuem *hardware* dedicado para a geração de sinais PWM *on-chip*, facilitando a implementação.

Figura 12 – Sinal modulado.



Fonte: Adaptado de Rashid (1999).

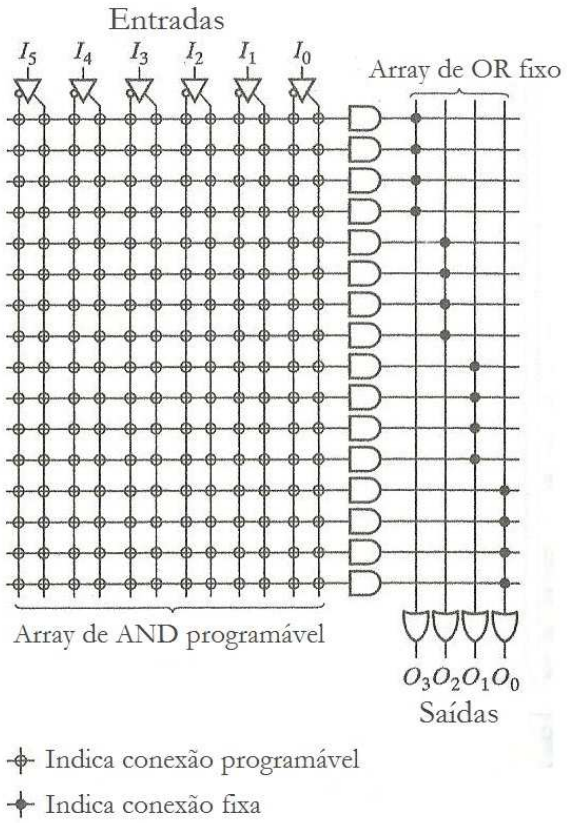
4.4 DISPOSITIVOS LÓGICOS PROGRAMÁVEIS

Dispositivos Lógicos Programáveis (PLDs) são uma combinação de circuitos combinacionais lógicos que podem ser programados, a fim de atender às necessidades do programador. Os primeiros PLDs pertencem a uma classe denominada PLDs simples (SPLDs) e são chamados *programmable array logic* (PAL) ou *programmable logic array* (PLA), dependendo do seu esquema de programação[1], e inicialmente possuíam apenas portas lógicas convencionais, porém depois passaram a contar com *flip-flops* em cada saída do circuito. Nas Figuras 13 e 14 estão representadas as arquiteturas dos PALs e dos PLAs, respectivamente. Posteriormente, foram acrescentadas macrocélulas a cada saída do PLD, que continham, além do *flip-flop*, portas lógicas e multiplexadores; *Generic array logic* (GAL) foi o nome dado à essa nova estrutura (PEDRONI, 2010).

PALs, PLAs e GALs, todos classificados como SPLDs, foram substituídos pelos PLDs complexos (CPLDs). Os CPLDs, representados na Figura 15, são dispositivos compostos por vários PLDs fabricados em um único invólucro, que se comunicam por meio de um barramento

[1]Enquanto em um PAL as ligações AND são programáveis e as ligações OR são fixas, no PLA ambas são programáveis

Figura 13 – Arquitetura PAL



Fonte: Adaptado de Rabaey, Chandrakasan e Nikolić (2003).

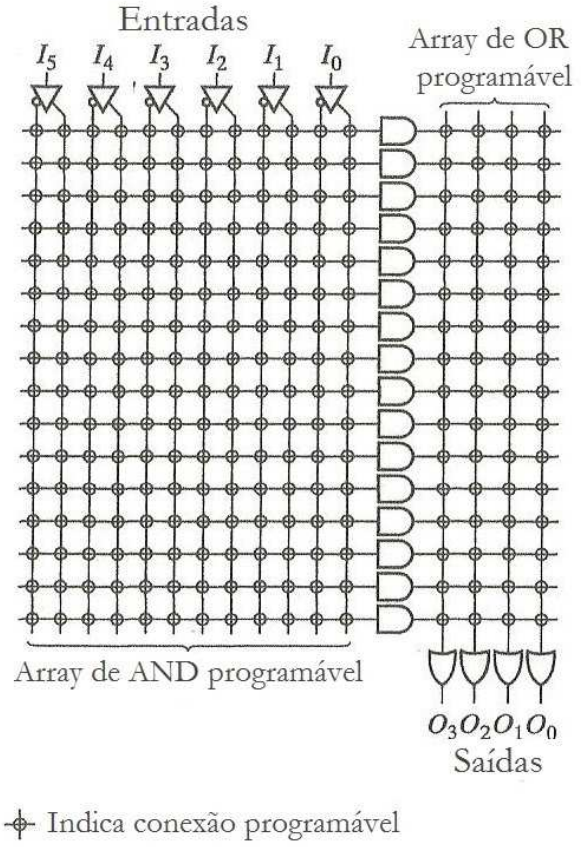
de interconexões, e possuem drivers de I/O mais modernos, suporte para *Joint Test Action Group* (JTAG) e um grande número de pinos de I/O disponíveis (PEDRONI, 2010).

Por fim, vieram as matrizes de portas programáveis, ou de acordo com a sigla em inglês, *field-programmable gate array* (FPGA), que possuem arquitetura, tecnologia, características, forma, custo e desempenho diferente de seus antecessores, os CPLDs, e serão explicados na subseção 4.4.1.

4.4.1 FPGA

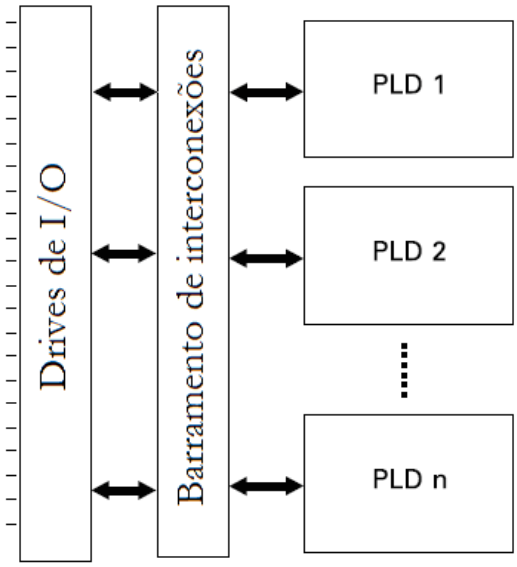
O FPGA é um dispositivo que pertence a uma subcategoria de dispositivos lógicos programáveis (PLDs). A Figura 16 representa uma visão abstrata da estrutura de um FPGA, onde podem ser vistos os blocos lógicos (na Figura 16 escritos como *logic cell*), e as chaves programáveis (tradução livre do inglês *Programmable Switch*), ou matriz de interconexões. Segundo Chu (2008) e Hauck e DeHon (2007), cada bloco lógico pode executar simples funções lógicas, enquanto as chaves programáveis podem ser configuradas para interconectar os diversos blocos. Para programar as suas aplicações, o FPGA, assim como os CPLDs, pode utilizar a linguagem VHDL.

Figura 14 – Arquitetura PLA



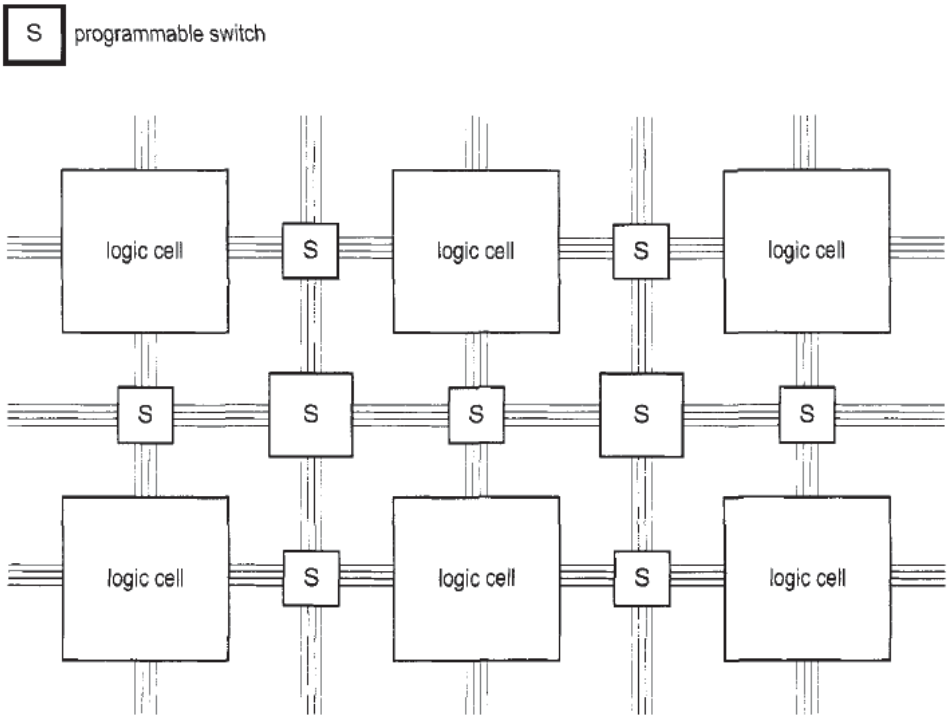
Fonte: Adaptado de Rabaey, Chandrakasan e Nikolić (2003).

Figura 15 – Arquitetura Básica de um CPLD



Fonte: Adaptado de Pedroni (2010).

Figura 16 – Estrutura do FPGA



Fonte: Retirado de Chu (2008).

4.4.2 LINGUAGEM VHDL

VHDL é uma linguagem de descrição de hardware, que pode ser utilizada tanto para descrição como para síntese, e simulação de circuitos. Desenvolvida na década de 1980 pelo Departamento de Defesa dos Estados Unidos, foi ganhando ao longo dos anos novas versões, onde foram feitas alterações a fim de adicionar facilidades à linguagem.

Na linguagem, os comandos são executados de forma concorrente e, para que sejam executados sequencialmente, devem estar dentro de estruturas denominadas *PROCESS*, *FUNCTION* ou *PROCEDURE*, todas essas sendo palavras reservadas da linguagem. Uma característica importante da linguagem VHDL, é que, para serem utilizados, bibliotecas e pacotes (ou *libraries* e *packages*) devem ser declarados. As bibliotecas armazenam as informações compiladas, enquanto que os pacotes fornecem um meio para definir recursos de uso comum, como constantes, funções e declarações de tipos.

Para o procedimento de verificação da compatibilidade do código desenvolvido, com o que foi especificado no projeto, cria-se um outro código, também em VHDL, a fim de que este defina estímulos de teste. Outro aspecto da linguagem é que a mesma não é *case-sensitive*, ou seja, não difere comando em letras maiúsculas ou minúsculas.

Um projeto em VHDL é composto por dois blocos fundamentais: a declaração da entidade (ou *entity*) e arquitetura (ou *architecture*). Entidade é onde as entradas e saídas configuradas estão relacionadas e especificadas. Enquanto que a arquitetura é onde todo o funcionamento

do circuito está descrito, onde são feitas as interligações entre os componentes que foram devidamente declarados (D'AMORE, 2012).

5 MATERIAIS

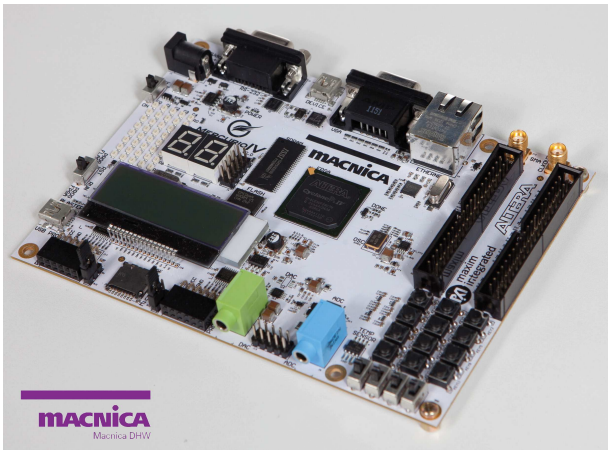
Este projeto consiste no controle digital de um conversor *Buck* utilizando FPGA, portanto para ser realizado, foram utilizados o Kit Mercurio® IV (5.1) e o conversor *Buck* (5.2), com seus componentes, descritos na sequência.

5.1 MERCURIO® IV DEVKIT

O Kit Mercurio® IV (Figura 17) é uma placa de desenvolvimento criada pela Macnica DHW com base no FPGA Cyclone® IV EP4CE30F23 FPGA da Altera, que possui cerca de 30 mil elementos lógicos. Além de outros componentes, o kit é equipado com um conversor A/D, modelo MAX1379, fabricado pela Maxim Integrated, que possui as seguintes especificações, segundo o seu *datasheet* (Integrated (2009)):

- 12 bits de resolução;
- 1,25 Msps de taxa de amostragem;
- Possibilidade de duas amostras simultâneas;
- Referencal interno ou externo;
- 3-*Wire Serial Interface* [2].

Figura 17 – Mercurio® IV DevKit.



Fonte: Macnica (MACNICA. . . , 2017).

5.2 CONVERSOR *BUCK*

Para a construção do conversor *Buck*, foram utilizados: capacitor de 10 μF, indutor de 200 mH e resistor de 560 Ω, bem como o diodo *Schottky* 1N5819. Conforme visto na subseção

[2] Um método de comunicação serial no qual um dado pode ser enviado ao mesmo tempo que um outro dado é recebido.

4.2.1, o conversor *Buck* possui ainda uma chave de potência. Neste trabalho, um transistor de efeito de campo metal - óxido - semicondutor de canal n (NMOSFET) possui a função da chave, e foi utilizado o IRF630. Para que a potência necessária para ativar o NMOSFET fosse alcançada, utilizou-se o driver IR2102.

Os pulsos gerados pela modulação PWM no Kit Mercurio® IV são repassados ao circuito através do acoplador óptico[3] 6N137. Esse sistema conta ainda com um regulador de tensão, LN7805, para manter a tensão de alimentação constante e dentro dos limites desejados.

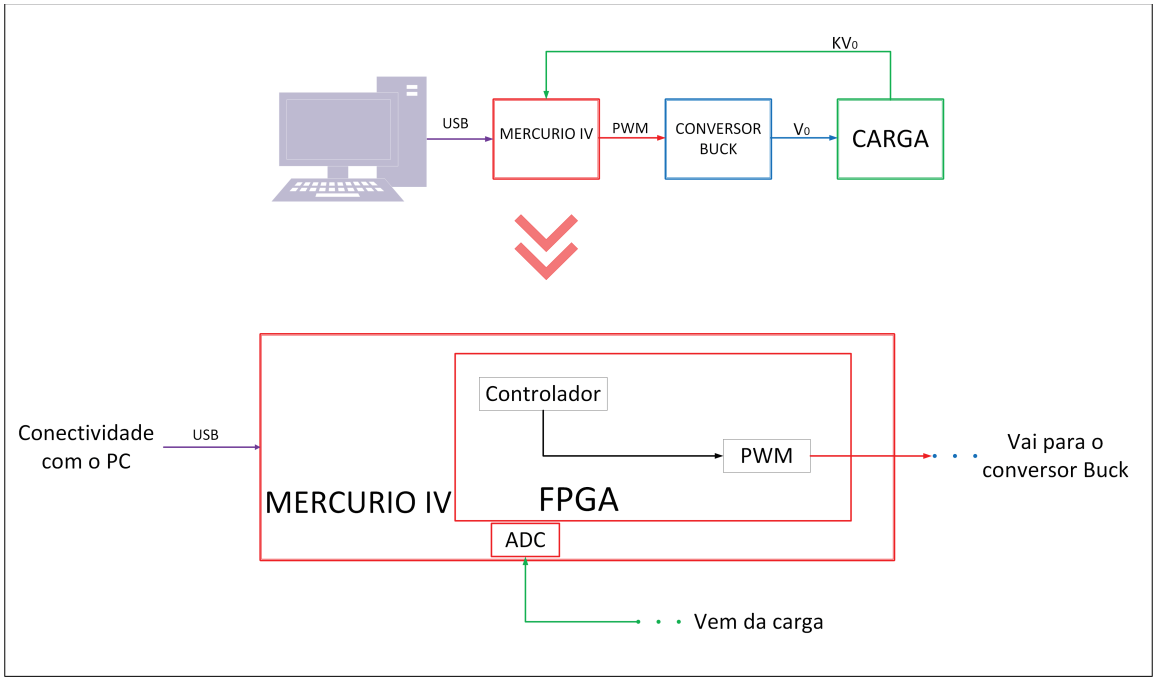
A montagem desse circuito está especificada no próximo capítulo, na Seção 6.4.

[3]Dispositivos que transmitem sinais de um circuito para o outro sem ligação elétrica

6 METODOLOGIA

O projeto foi desenvolvido conforme o esquema da Figura 18, tendo o computador como primeiro elemento, seguido pelo Kit Mercurio® IV, composto de mais partes importantes ao projeto (FPGA, ADC), e por fim, o Conversor *Buck* com a carga. Cada componente do esquema, com suas respectivas funções, será exposto ao longo desse capítulo.

Figura 18 – Diagrama Geral.



Fonte: Elaborado pela Autora.

6.1 PC

O primeiro item, o computador, ou do inglês *Personal Computer* (PC), além de alimentar o kit de desenvolvimento (Mercurio® IV), deverá programar e configurar o FPGA e o ADC, através do *software* Intel® Quartus® Prime. O conector USB é o responsável por acoplar o kit e o computador.

6.2 A/D

O outro componente do Kit Mercurio® IV, relevante à este projeto, é o conversor A/D (na Figura 18 representado como ADC). Como é um sistema realimentado, o conversor A/D possui a função de receber o sinal oriundo da carga, ou seja, da saída do conversor *Buck*, que é analógico, e o converte para digital, repassando-o ao controlador PID. A descrição do

comportamento do conversor A/D foi feita em linguagem VHDL, assunto abordado na Subseção 6.3.1.

6.3 FPGA

No FPGA, parte componente do Kit Mercurio® IV, foram implantados os códigos em VHDL do conversor A/D, do controlador PID e do PWM, que serão apresentados nas subseções subsequentes. Na última subseção está descrito o código principal que coordena todos esses componentes anteriores. Para a confecção dos códigos foi utilizado o *software* Intel® Quartus® Prime.

6.3.1 CÓDIGO CONTROLADOR DO CONVERSOR A/D

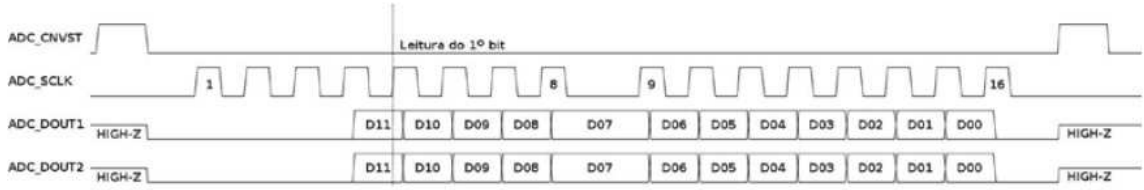
O conversor A/D se comunica com o FPGA Cyclone IV através de um barramento serial SPI, e o seu controle é feito através de alguns sinais digitais de entrada, enviados para os seus pinos, citados a seguir:

- REFSEL: Seleciona qual referencial será utilizado, o referencial interno (‘0’) ou externo (‘1’);
- $\overline{\text{U}}$/B: Seleciona o modo Unipolar (‘0’) ou Bipolar (‘1’). Se o modo for o bipolar, as entradas analógicas serão diferenciais;
- DOUT1 e DOUT2: São as saídas de dado da comunicação serial;
- SCLK: É o clock do dispositivo, o qual define o tempo de conversão;
- CNVST: Determina o início da conversão. A conversão inicia na borda de descida de CNVST;
- $\overline{\text{CS}}$: Ativa (‘0’) ou desativa (‘1’) a interface serial;
- S/$\overline{\text{D}}$: Seleciona se a entrada será *single* (‘1’) ou *dual* (‘0’). Se for *single*, o dado do segundo canal irá para DOUT1 após o dado do primeiro canal ter sido totalmente enviado. Se for *dual*, o dado do primeiro canal irá para DOUT1 e o dado do canal 2 irá para DOUT2 ao mesmo tempo;
- SEL: Se estiver em nível baixo, seleciona os canais primários, AIN1A e AIN2A, e em nível alto, seleciona os secundários, AIN2B e AIN2B. Se $\overline{\text{U}}$/B estiver em nível alto, ou seja, no modo bipolar, SEL é ignorado.

Após a descrição dos pinos, e visualização da Figura 19, que representa como a leitura dos bits é feita no modo dual, torna-se possível compreender como o código do conversor A/D foi feito. Embora a figura retrate os dois canais, e o ADC esteja configurado para o modo *dual*, este projeto utiliza apenas um canal para leitura.

Primeiramente, quando a borda de subida do *clock* (neste projeto possui o valor de 50 MHz, e *duty cycle* de 50%, produzido pelo Kit Mercurio® IV) é identificada, verifica-se se o conversor está ligado no barramento SPI. Em caso positivo, e se há um bit a ser transferido ao *buffer*, é feita a contagem de pulsos para dividir a frequência do *clock* de entrada para um valor

Figura 19 – Forma de onda do conversor AD no modo dual.



Fonte: Macnica (MACNICA. . . , 2017).

previamente definido pelo usuário, o que irá resultar no sinal de *clock* do dispositivo, enviado para o SCKL. Após a divisão da frequência, confere-se a contagem de uma variável auxiliar. Se essa contagem for igual a 16 (no modo *dual*), que é o número de pulsos do SCLK necessários para completa conversão (sendo 4 pulsos de latência inicial e mais 12 para o envio total do resultado da conversão), o resultado da conversão analógica é enviado para a saída da *entity*.

As configurações de cada pino e a descrição do funcionamento em linguagem VHDL encontram-se disponíveis no Apêndice A. Importante ressaltar que o código é disponibilizado pela Macnica DHW (MACNICA. . . , 2017), e que foi feita uma adaptação do mesmo, para que se adequasse às necessidades do projeto.

6.3.2 CÓDIGO PID

O controlador digital PID, modelado em VHDL e implantado no FPGA, recebe os dados provenientes do usuário (as constantes do controlador, que podem ser alteradas no próprio código), e do ADC (leitura da saída do Conversor *Buck*) e realiza a compensação do sistema para que a resposta seja a mais próxima possível da desejada.

O código do controlador PID, apresentado no Apêndice B, foi construído baseado na descrição da Equação 22 em VHDL, a qual utiliza números não inteiros para o seu cálculo, podendo estes serem representados por ponto fixo ou ponto flutuante. Na representação por ponto fixo, a parte inteira e fracionária possuem, cada uma, um número fixo de bits, enquanto que na representação por ponto flutuante, esses bits são variáveis.

Neste projeto optou-se pela representação em ponto fixo, que não é um tipo padrão da linguagem VHDL. Portanto para que isso fosse possível, foram utilizados os pacotes fixed_float_types e fixed_pkg. Tais pacotes permitem a definição de dois novos tipos: “ufixed”, o ponto fixo sem sinal, e “sfixed”, o ponto fixo com sinal; e também de algumas novas funções, sendo as utilizadas neste projeto: “ to_ufixed” e “to_sfixed” (responsáveis pela conversão para o tipo “ufixed” e “sfixed”, respectivamente), e a função “resize”, utilizada para fixar o tamanho da saída (BISHOP,).

Conforme dito anteriormente, o código do controlador é a descrição em VHDL da Equação 22, portanto foram criados vetores do tipo “sfixed” para os sinais de erro ($e(n)$, $e(n-1)$ e $e(n-2)$), o sinal de saída do controlador ($u(n)$ e $u(n-1)$), e temp, com três posições para

receber os valores da multiplicação das constantes (b_0, b_1, b_2) pelo valor dos erros no instante atual e nos instantes anteriores.

O erro atual, $e(n)$, recebe a diferença entre o valor de referência e o valor da tensão lido e convertido pelo ADC. A cada pulso de *clock*, esse valor do erro atual é deslocado, portanto se torna primeiramente $e(n-1)$ e depois $e(n-2)$, enquanto novos erros vão sendo computados. A saída recebe a soma da multiplicação dos erros pelas suas respectivas constantes, e da saída anterior, que passou pelo mesmo processo de deslocamento dos erros.

Ao final é feita uma adequação do valor da saída para que fique entre os valores de 0 e 100. Esses valores representam a porcentagem enviada ao PWM para definir o *duty cycle* (δ).

6.3.3 PWM

Após a atuação do controlador PID um novo valor numérico é gerado, portanto para que a compensação seja repassada ao circuito do conversor Buck faz-se necessária uma nova conversão para retomar a forma analógica. Sendo assim, a modulação PWM é a responsável por repassar ao conversor *Buck* novos valores relacionados ao seu tempo de chaveamento.

O código do PWM utiliza-se de uma frequência dividida, produzida pelo componente Prescaler, apresentado no Apêndice D. O Prescaler recebe o sinal de *clock*, e o valor da relação entre a frequência de entrada e saída (denominada no código como PRE). Quando uma variável, utilizada para contar as bordas de subida do *clock* de entrada, se equivale ao valor da metade de PRE, o sinal de *clock* da saída assume um valor oposto ao qual estava antes. Sendo assim, o período do *clock* de saída é PRE vezes maior do que o *clock* de entrada e o seu *duty cycle* é de 50%.

Após a divisão da frequência, verifica-se a borda de subida do sinal, advindo do Prescaler. Em caso positivo, decrementa-se um contador, que foi carregado com o valor de $100 - duty$, sendo *duty* a variável que recebe o valor do *duty cycle* na ordem de 0 a 100. Quando o contador é zerado, o PWM vai para nível alto, e outro contador é carregado com o valor de *duty*, e também decrementado, a cada pulso do *clock*. No momento em que este segundo contador é zerado, o PWM vai para nível baixo e o primeiro contador é recarregado para que o processo ocorra novamente. No Apêndice C encontra-se a descrição do PWM em VHDL.

6.3.4 CÓDIGO PRINCIPAL

A entidade do código principal possui como entradas: o sinal de *clock*, proveniente do Kit Mercurio® IV e que será repassado aos componentes; o sinal de *reset*, também repassado aos componentes; e o ADC_DOUT que é o sinal correspondente ao bit lido pelo canal 1 do ADC. As saídas são todas referentes à configuração do ADC.

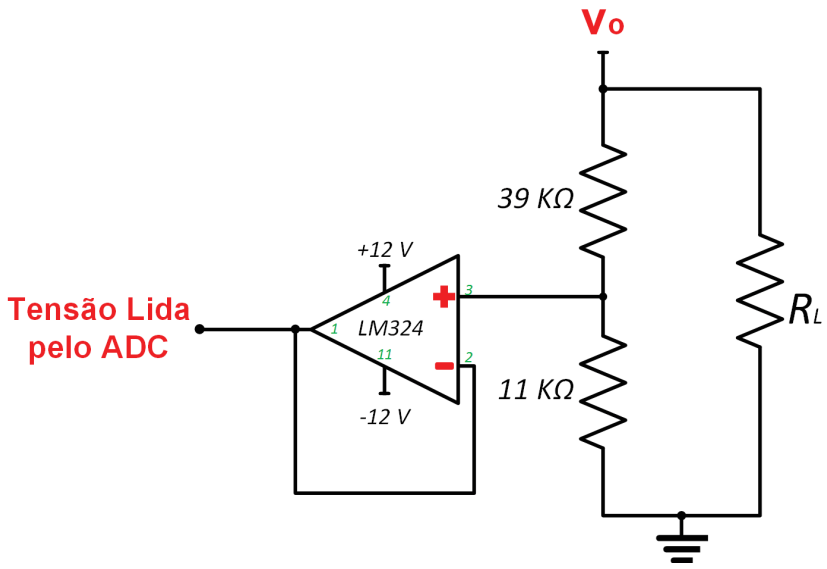
Na arquitetura do código principal, as constantes, a tensão de entrada e a tensão de referência são declaradas e podem ser alteradas. Além disso, os bits recebidos pelo ADC são convertidos para um valor em Volts, o qual será enviado para o componente referente ao controlador PID, onde o erro do sistema é calculado e multiplicado pelas constantes, gerando

uma saída do tipo sfixed. Portanto, para esta ser repassada ao PWM, deve ser convertida para o tipo inteiro. Após o bloco de conversões, os componentes são instanciados com seus respectivos sinais. O código principal encontra-se no Apêndice E.

6.4 CONVERSOR *BUCK*

Acoplado ao Kit Mercurio® IV está o Conversor *Buck* que, conforme explicado na seção 4.2.1, faz o rebaixamento da tensão aplicada à sua entrada e a fornece à carga. Essa tensão, chamada de V_o, alimenta a carga e é enviada ao conversor A/D do Mercurio® IV, após ser multiplicada por um fator K (sendo $K < 1$), a fim de adequar o sinal às especificações do conversor A/D. Esse fator K, na prática, é um divisor resistivo, projetado para que com a entrada entre 0 e 15 V, a saída fique no intervalo de 0 a 3.3 V, pois é a faixa de operação do ADC. Devido ao fato de a impedância de entrada do ADC ser baixa, na saída do divisor resistivo, foi colocado um LM324 que atua como um *buffer*, ou seguidor de tensão. A Figura 20 apresenta o divisor resistivo, com o valor das resistências utilizadas, e também o LM324 com as suas ligações identificadas (números em verde).

Figura 20 – Divisor resistivo e *Buffer*.



Fonte: Autoria Própria.

Para um melhor funcionamento do conversor Buck, foram acrescentadas algumas etapas anteriores à ele, que estão enumeradas e explicadas abaixo. O Apêndice F representa o esquemático do circuito completo, dividido em blocos, simbolizando as etapas, devidamente identificados.

O circuito completo foi montado em uma PCB (do inglês *Printed Circuit Boards*), que terá sua confecção detalhada na Subseção 6.4.1.

1. Circuito de Isolação

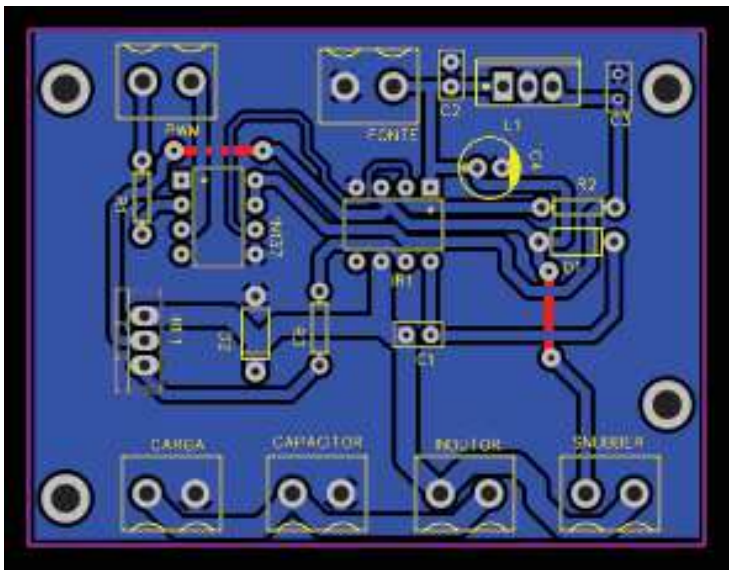
Para repassar o sinal advindo do PWM ao restante do circuito foi utilizado o acoplador óptico, também denominado optoacoplador, 6N137. Um acoplador óptico permite que um circuito seja controlado por outro sem que haja contato elétrico entre eles, pois seu funcionamento é baseado na transferência de sinais através de energia luminosa no interior do CI.

2. Regulador de Tensão
   Após o bloco do optoacoplador está o bloco do regulador de tensão que possui a função de manter a tensão nos limites exigidos para o funcionamento do circuito. No projeto foi utilizado o regulador LN7805 que está recebendo 15 V em sua entrada e fornece 5 V.
3. Etapa de Potência
   Antecedendo imediatamente o Conversor *Buck*, foi colocado um *driver* de potência, um dispositivo responsável por amplificar a potência que recebe em sua entrada a fim de que seja correspondente às especificações do MOSFET e ele possa ser acionado. Para esse projeto, o IR2102, um circuito integrado, foi utilizado como driver.

6.4.1 MONTAGEM DA PCB

O processo para a montagem do circuito na PCB iniciou com a criação do esquemático, e o *design* da PCB, utilizando-se para tal uma ferramenta *online* chamada *EasyEDA*. A Figura 21 é o resultado do *design* desse projeto, com os componentes posicionados, e as trilhas fazendo as ligações entre os mesmos. Após o design, a placa foi confeccionada através do processo de transferência térmica.

Figura 21 – Design da PCB.



Fonte: Acervo Próprio.

Para permitir que o estudante verifique a influência dos diferentes valores do indutor, do capacitor e da carga no resultado, foram colocados bornes de 2 vias para que a troca dos

componentes possa ser feita de maneira fácil, permitindo mais de uma configuração para o conversor *Buck*.

Além disso, foi incluído um borne para um circuito opcional, denominado *Snubber*, responsável por limitar os transientes de alta tensão, ou *spikes*, que ocorrem na comutação da carga, protegendo o MOSFET de danos e minimizando interferências causadas por esses transientes.

7 ANÁLISE E DISCUSSÃO DOS RESULTADOS

Após a modelagem do conversor *Buck*, descrita na Seção 4.2, a Equação 77 foi obtida. Para fins de exemplo, foram adotados, tanto para simulações, quanto para a prática, os seguintes valores para os componentes:

- $C = 10\ \mu$F ;
- $L = 200$ mH (Obedecendo o critério explicitado na Seção 4.2, para que o conversor opere no Modo de Condução Contínuo, tomando como parâmetros a frequência de 1 kHz e o ciclo de trabalho de 50%);
- $R_L = 560\ \Omega$.

Ainda, para as constantes do PID, utilizando o método de sintonia manual, foram adotados: $K_P = 75{,}65$, $K_I = 1000$ e $K_D = 2{,}1 \cdot 10^{-6}$.

Neste Capítulo serão apresentados os resultados alcançados por este trabalho, sendo as Seções 7.1 e 7.2 referentes à simulação, e à implementação prática do circuito, respectivamente.

7.1 SIMULAÇÃO

Foram simulados os resultados do sistema em malha aberta 7.1.1 e em malha fechada 7.1.2, ou seja, com a atuação do controlador PID.

7.1.1 MALHA ABERTA

A fim de verificar o comportamento da tensão de saída do conversor Buck, em malha aberta, foram considerados $V_i = 15$ V, $d(s) = 0{,}5$ e $R_C = R_{indutor} = 3\ \Omega$. Portanto, após a inserção dos valores, e rearrumação dos termos da Equação 77, obteve-se a Equação 78:

$$v_o(s) = \frac{112{,}5s\ +\ 3{,}75 \cdot 10^6}{s^2\ +\ 208{,}6s\ +\ 5 \cdot 10^5}. \tag{78}$$

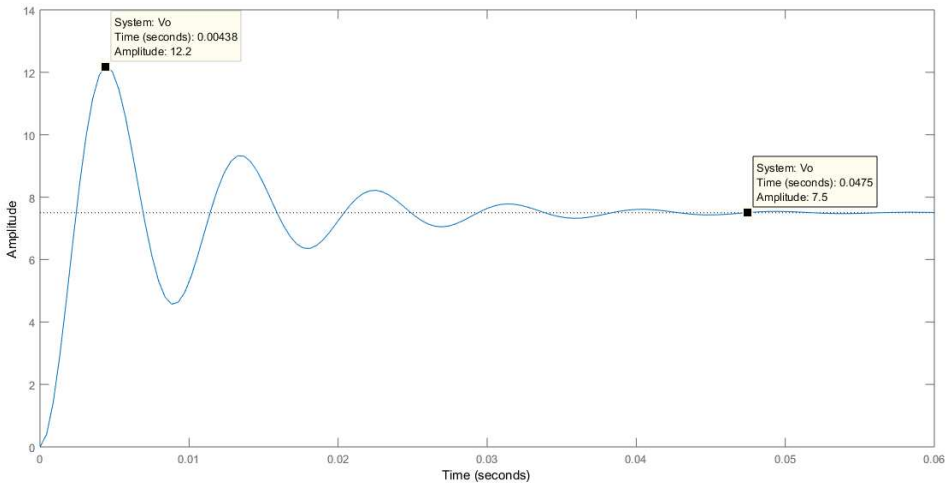
Conforme a Figura 22, verifica-se que em malha aberta, a tensão de saída possui um sobressinal de 61%, e tempo de acomodação de $47{,}5$ ms.

7.1.2 MALHA FECHADA

A simulação para malha fechada foi feita utilizando a diagramação gráfica por blocos, exposta na Figura 23, onde estão representados o controlador PID, a função de transferência do Conversor *Buck* modelada, e a tensão de referência, que é o valor de $7{,}5$ V desejado para a tensão de saída.

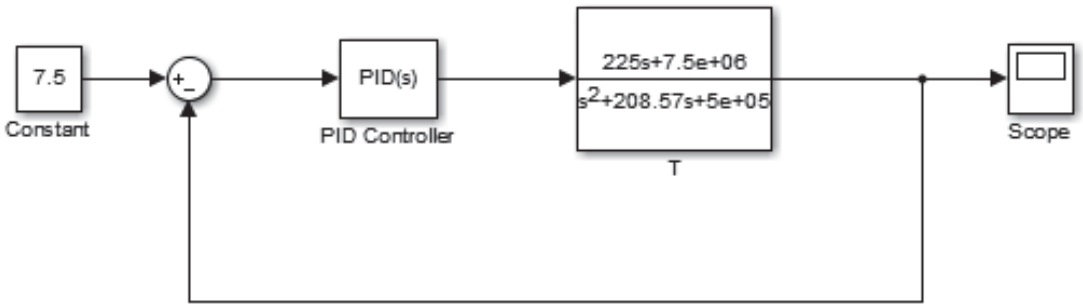
O transitório da resposta encontra-se na Figura 24, na qual pode ser observado que o valor do sobressinal foi de $10{,}4$ V, e o tempo de acomodação foi de aproximadamente $0{,}8$ ms.

Figura 22 – Tensão de saída em malha aberta (Simulação)



Fonte: Autoria Própria.

Figura 23 – Diagrama de Blocos no Simulink



Fonte: Elaborado pela Autora.

Enquanto que na Figura 25 está representada a resposta no estado estacionário com o valor de 7,5 V, exatamente o valor esperado.

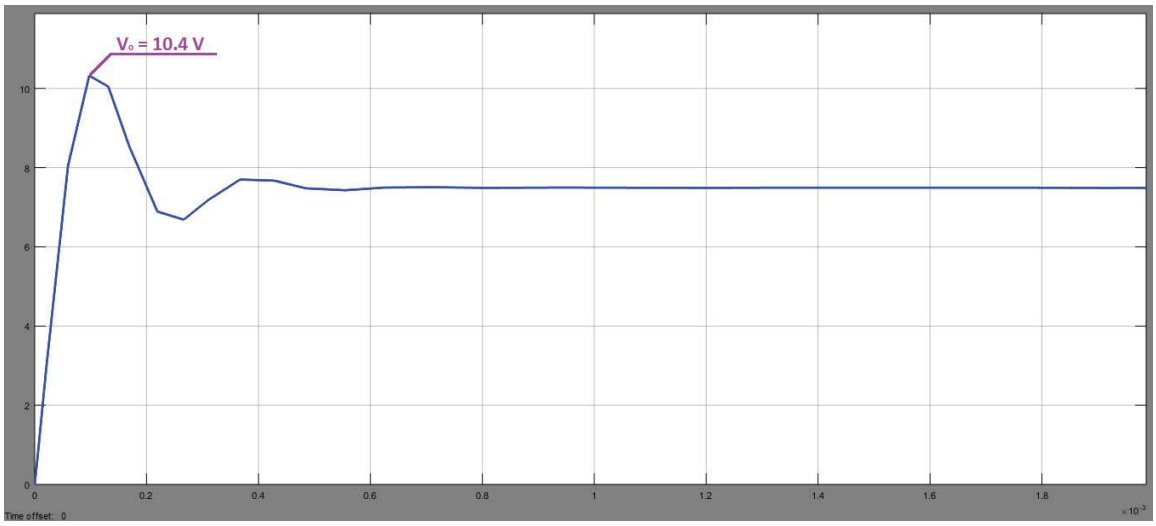
Comparando os resultados de malha aberta com os de malha fechada simulados, é possível verificar através das Figuras 28, 29 e 30 que:

- O sobressinal foi reduzido de 12,2 V para 10,4 V.
- O tempo de acomodação foi reduzido de 47,5 ms para 0,8 ms.
- O erro em regime estacionário se manteve muito próximo de zero nas duas situações.

7.2 IMPLEMENTAÇÃO PRÁTICA

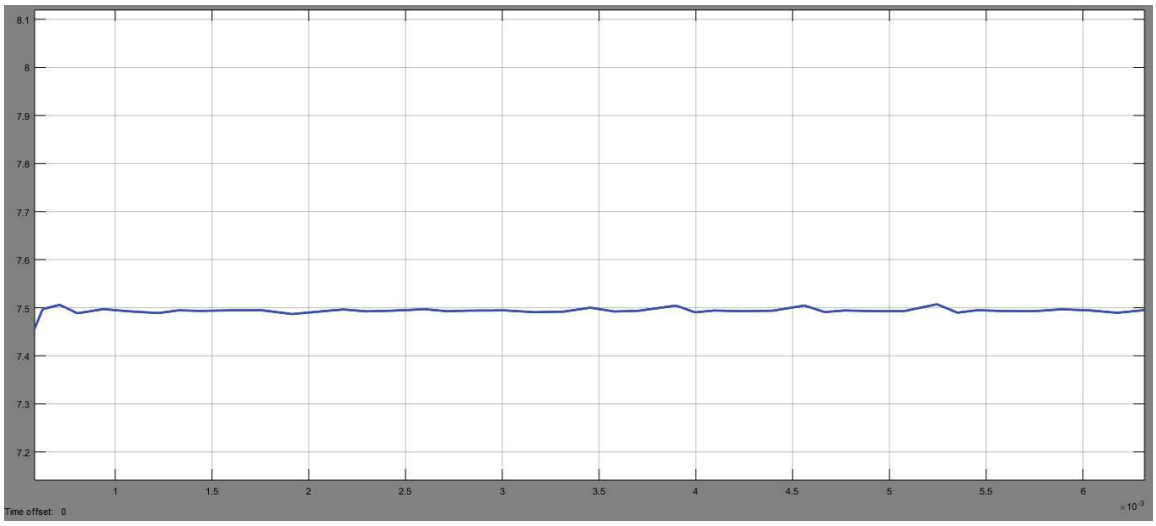
Nesta seção serão apresentados os resultados da implementação prática em malha aberta (7.2.1), e em malha fechada (7.2.2), ou seja, com a atuação do controlador PID.

Figura 24 – Sobressinal em Malha Fechada (Simulação)



Fonte: Elaborado pela Autora.

Figura 25 – Regime Estacionário em Malha Fechada (Simulação)

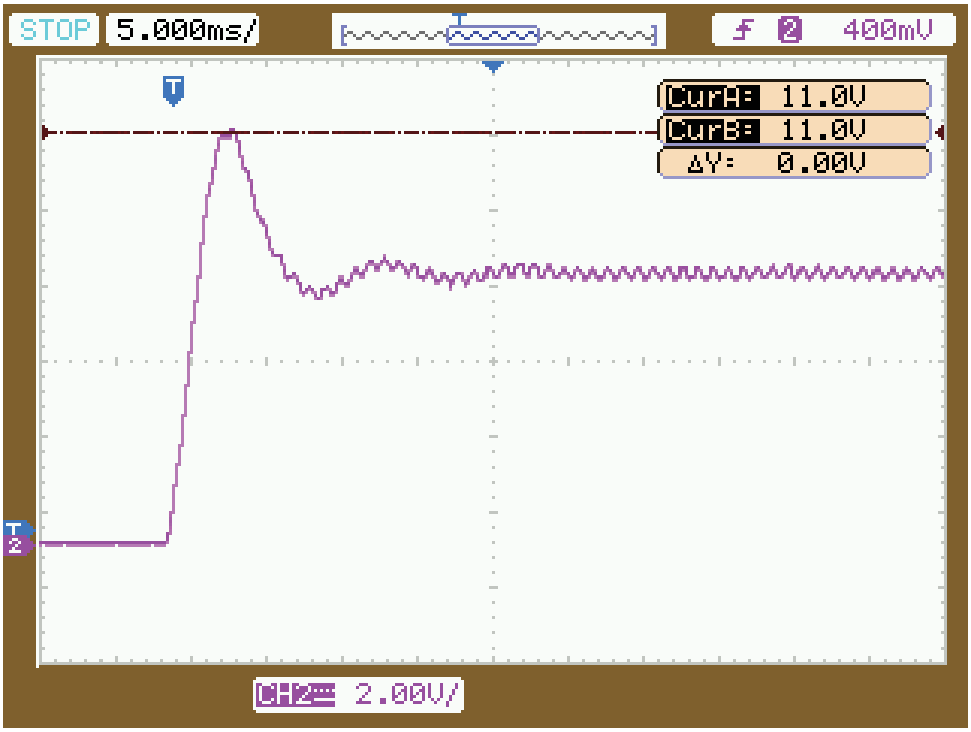


Fonte: Elaborado pela Autora.

7.2.1 MALHA ABERTA

Para o PWM configurado com frequência de 1 kHz e $\delta = 0{,}5$ e tensão de entrada de 15 V aplicados no circuito, o comportamento da tensão de saída em malha aberta pode ser visto nas Figuras 26 e 27.

Figura 26 – Sobressinal em Malha Aberta (Prática)



Fonte: Autoria Própria.

Pode-se observar que o sinal apresenta um sobressinal de 11 V (Figura 26), e um tempo de acomodação de 28 ms (Figura 27), que diferem um pouco dos valores da simulação em malha aberta. Para explicar essa divergência de valores, os motivos possíveis são: as simplificações feitas para algumas variáveis durante a modelagem, as quais no modelo real continuam atuando; e a precisão do osciloscópio utilizado para a captação das ondas na prática.

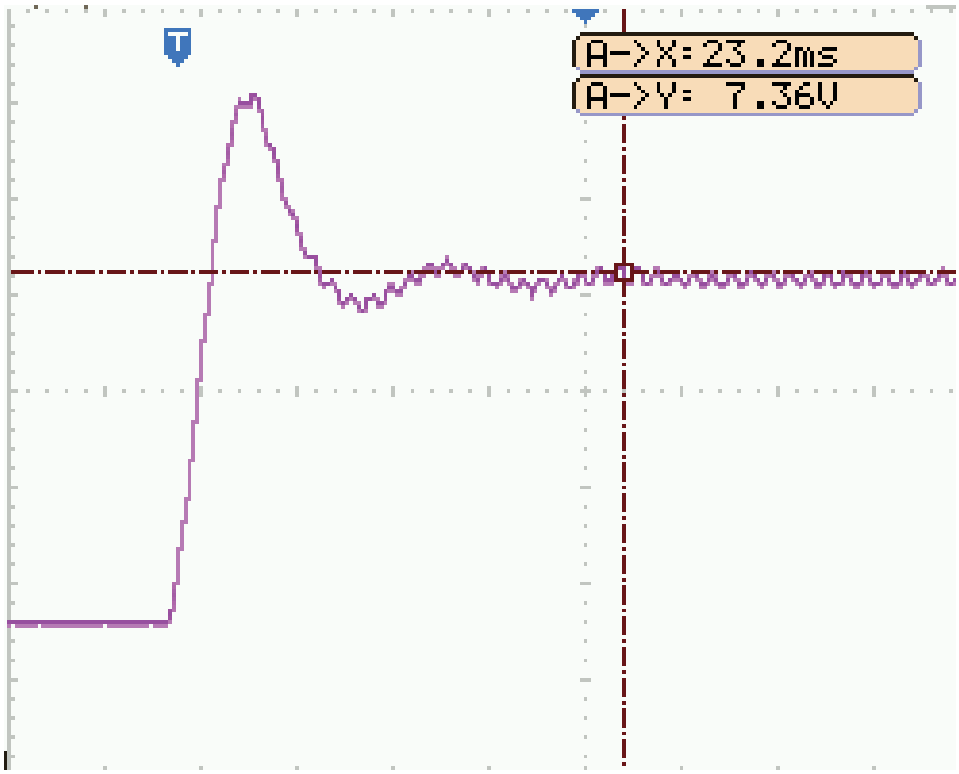
7.2.2 MALHA FECHADA

Com os valores de K_P K_I e K_D citados acima e de acordo com as Equações 17, 18 e 19, os parâmetros do controlador são: $b_0 = 1075{,}65$, $b_1 = -75{,}65$, $b_2 = 0{,}00147$.

Após a aplicação do controle PID, e comparando com os resultados de malha aberta, na prática, é possível verificar através das Figuras 28, 29 e 30 que:

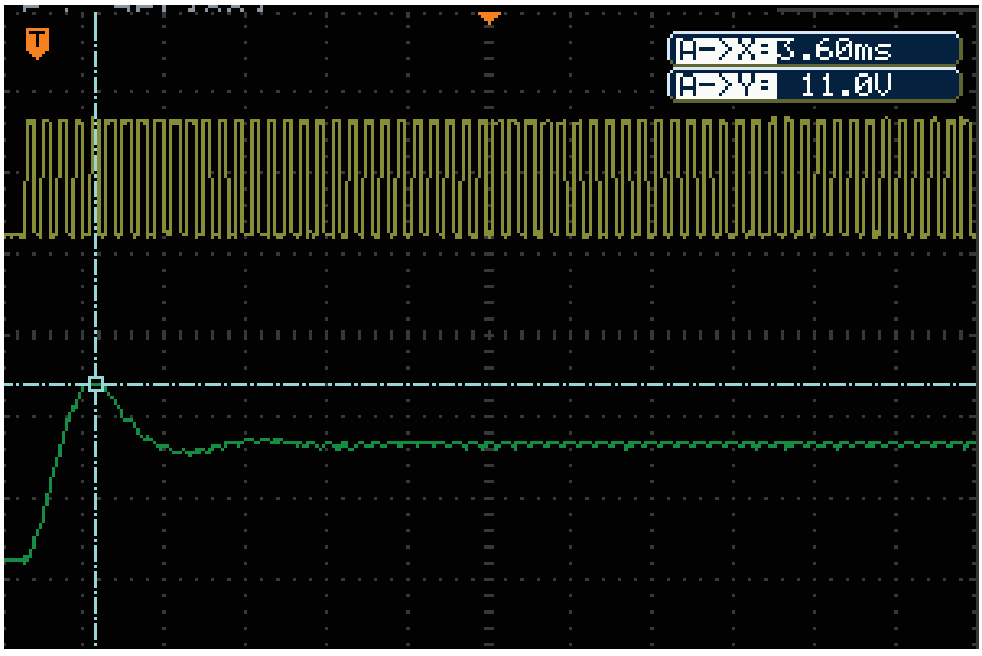
- O sobressinal se manteve.
- O tempo de acomodação foi reduzido de 23.2 ms para $16{,}8$ ms.
- O erro em regime estacionário foi reduzido de $1{,}86\%$ ($7{,}36$ V em malha aberta) para $1{,}33\%$ ($7{,}4$ V em malha fechada), tendo como referência a tensão de $7{,}5$ V esperada.

Figura 27 – Tempo de Acomodação em Malha Aberta(Prática)



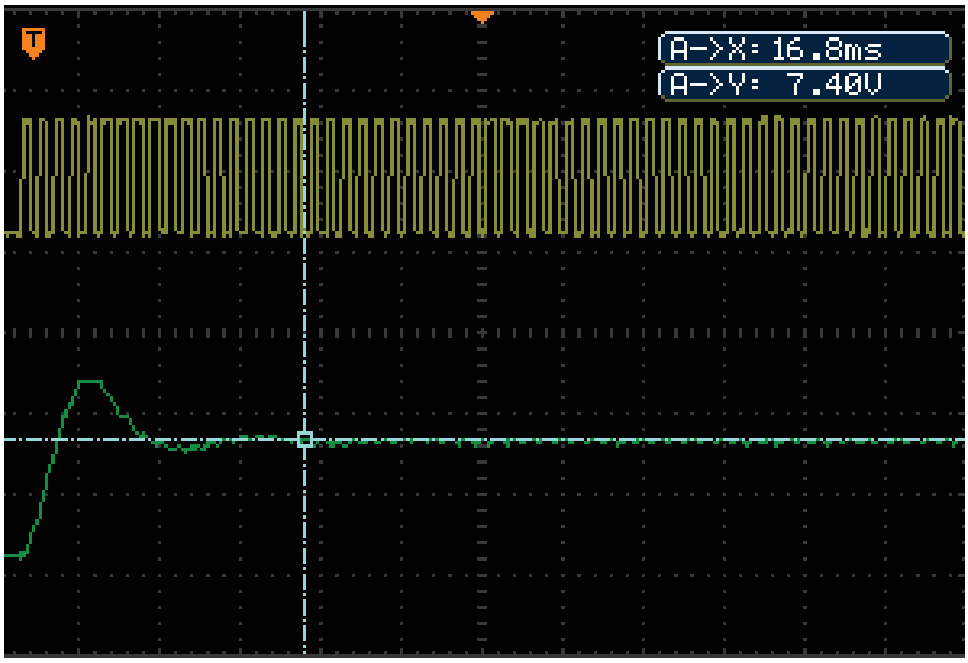
Fonte: Autoria Própria.

Figura 28 – Sobressinal em Malha Fechada (Prática)



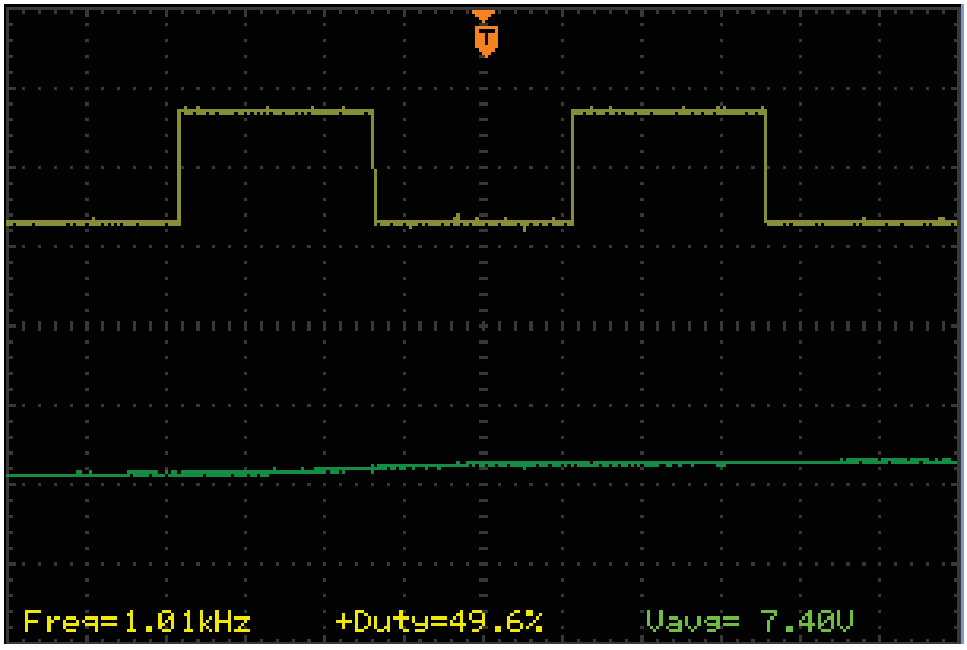
Fonte: Autoria Própria.

Figura 29 – Tempo de Acomodação em Malha Fechada (Prática)



Fonte: Autoria Própria.

Figura 30 – Regime Estacionário em Malha Fechada (Prática)



Fonte: Autoria Própria.

Considerando os resultados de malha fechada na prática, conclui-se que o sobressinal, o tempo de acomodação e o erro em regime estacionário foram maiores na prática em comparação com os resultados simulados. Os motivos para a divergência dos valores são os já citados na Subseção 7.2.1, referentes às simplificações feitas na modelagem matemática e à precisão do osciloscópio.

8 CONCLUSÃO

A Engenharia Eletrônica é composta por diversas áreas, as quais apesar de abordarem teorias em campos diferentes, podem ser conciliadas proporcionando inúmeros projetos, conforme o presente trabalho se propôs a fazer. Sendo assim, a Seção 8.1 apresenta uma sugestão para extensão deste projeto, de forma que explore mais a área de Controle Digital, que não foi o foco dessa ferramenta educacional.

Por último, a Seção 8.2 traz as considerações finais deste trabalho.

8.1 TRABALHOS FUTUROS

Para trabalhos futuros, propõe-se o desenvolvimento de uma interface na qual os alunos possam inserir as constantes do controlador PID, e visualizar o efeito das mesmas na tensão de saída. Para que isso seja possível, seria utilizado o chip FT245RL, disponível no Kit Mercurio® IV, responsável por gerenciar a comunicação USB e disponibilizar as informações, captadas da interface, para o FPGA.

Ademais, um outro componente necessário seria o Nios® II, um microprocessador, inserido no FPGA com um sistema de processamento equivalente a um microcontrolador. As atribuições desse microprocessador incluiriam o acesso de memórias (onde ficariam armazenadas as constantes inseridas pelo usuário), gerenciamento de todo o sistema, processamento das informações, e transferência de dados com o Kit.

8.2 CONSIDERAÇÕES FINAIS

Ao final deste projeto, os resultados obtidos foram satisfatórios. O modelo matemático do conversor *Buck* descreveu, de forma confiável, o seu comportamento real, mesmo que o sobressinal de malha aberta da simulação não tenha sido compatível com o sobressinal da leitura da tensão em malha aberta no circuito real, conforme citado na Seção 7.2.

Para o sistema em malha fechada, o controlador PID compensou o sistema da maneira esperada, estipulando em sua saída um *duty cycle* condizente com os parâmetros passados e melhorando a resposta transitória e a resposta no estado estacionário. O PWM repassou ao circuito o *duty cycle* recebido pelo PID e também manteve a sua frequência em 1 kHz, de acordo com as contantes configuradas no código.

Não houve a aplicação dessa ferramenta de ensino nas disciplinas envolvidas, porém a mesma cumpre o objetivo ao qual foi proposto, visto que todos os componentes estão funcionando do modo esperado, e que portanto os alunos poderão verificar quais recursos foram utilizados e compreendê-los. Tal processo os estimulará a reproduzir os conceitos apreendidos e criar novas aplicações para os componentes abordados neste projeto.

Espera-se que esse projeto tenha sido um incentivo para que mais kits didáticos, com diferentes configurações e que envolvam outras disciplinas possam ser produzidos.

Outrossim, que seja utilizado por professores como complemento de suas aulas e por outros alunos como um auxílio para a compreensão não só da programação em VHDL e da configuração de um FPGA, como dos parâmetros do controlador, do funcionamento de um conversor *Buck*, entre outros.

REFERÊNCIAS

AHMED, A. **Eletrônica de potência**. São Paulo, Brasil: Pearson Prentice Hall, 2000. Citado 3 vezes nas páginas 8, 9 e 10.

BISHOP, D. **Fixed point package user's guide**. [S.l.]. Citado na página 29.

CHEN, C.-T. **Analog and Digital Control System Design**: Transfer-function, state-space, and algebraic methods. New York, USA: Oxford University Press, 1993. Citado na página 5.

CHU, P. P. **FPGA prototyping by VHDL examples**. USA: John Wiley & Sons, Inc, 2008. Citado 2 vezes nas páginas 21 e 23.

D'AMORE, R. **VHDL**: Descrição e síntese de circuitos digitais. Rio de Janeiro, Brasil: LTC Editora, 2012. Citado na página 24.

HAUCK, S.; DEHON, A. **Reconfigurable computing: the teory and pratice of FPGA-based computation**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007. Citado na página 21.

INTEGRATED, M. **Dual, 12-Bit, 1.25Msps Simultaneous-Sampling ADCs with Serial Interface**. 2009. Https://datasheets.maximintegrated.com/en/ds/MAX1377-MAX1383.pdf. Acesso em abr. 2018. Citado na página 25.

KESKI, V. M. **Educação e tecnologias: O novo ritmo da informação.** São Paulo, Brasil: Papirus, 2012. Citado na página 1.

KUO, B. C. **Digital Control Systems**. New York, USA: Oxford University Press, 1992. Citado na página 6.

MACNICA DHW. 2017. Http://www.macnicadhw.com.br/products/mercurion-4-devkit-board. Acesso em mar. 2017. Citado 2 vezes nas páginas 25 e 29.

MARX, K. **O Capital: Crítica da Economia Política**. São Paulo, Brasil: Abril Cultural, 1985. v. 1. Citado na página 1.

NISE, N. S. **Engenharia de Sistemas de Controle**. Rio de Janeiro, Brasil: LTC, 2013. Citado na página 5.

NORONHA, O. M. Práxis e educação. **Revista HISTEDBR On-line**, São Paulo, Brasil, p. 89, 2005. Citado na página 1.

OGATA, K. **Engenharia de Controle Moderno**. São Paulo, Brasil: Pearson Prentice Hall, 2010. Citado na página 16.

PEDRONI, V. **Eletrônica Digital Moderna e VHDL**. Rio de Janeiro, Brasil: Campus Elsevier, 2010. 648 p. Citado 3 vezes nas páginas 20, 21 e 22.

POMILIO, J. A. **Modelagem de Fontes Chaveadas: Método das Variáveis de Estado**. 2018. Disponível em: <http://www.dsce.fee.unicamp.br/~antenor/pdffiles/it505/CAP8.pdf>. Acesso em: junho de 2018. Citado na página 15.

RABAEY, J. M.; CHANDRAKASAN, A.; NIKOLIĆ, B. **Digital Integrated Circuits**: A design perspective. Upper Saddle River, NJ USA: Pearson Education Inc., 2003. 761 p. Citado 2 vezes nas páginas 21 e 22.

RASHID, M. H. **Eletrônica de potência**: Circuitos, dispositivos e aplicações. São Paulo, Brasil: Makron Books, 1999. Citado 4 vezes nas páginas 9, 10, 19 e 20.

SAVIANI, D. **Pedagogia Histórico-Crítica: primeiras aproximações**. Campinas, Brasil: Autores Associados, 2012. Citado na página 4.

STARR, G. P. **Introduction to Applied Digital Control**. USA: Department of Mechanical Engineering The University of New Mexico, 2006. Citado na página 6.

Apêndices

APÊNDICE A – CÓDIGO DO CONVERSOR A/D

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_unsigned.all;
4  use ieee.fixed_float_types.all;
5  use ieee.fixed_pkg.all;
6
7  entity ADC is
8  generic (PRE:integer:=500); -- Prescaler para reduzir o clock da entrada de 50 MHz
   para 100 kHz
9  port(
10       clk_i : in  std_logic;  -- Recebe o clock que será enviado ao ADC.
11       rst_i : in  std_logic;  -- Recebe o comando para resetar o ADC.
12       sd_i  : in  std_logic;  -- Recebe o comando para selecionar saída Single/Dual.
13       ub_i  : in  std_logic;  -- Recebe o comando para selecionar entrada
         Unipolar/Bipolar.
14       csn_i : in  std_logic;  -- Recebe o comando para habilitar chip no barramento
         SPI.
15
16       ADC_DOUT_i   : in  std_logic;   -- Saída do dado Serial 1. Dado é recebido na
         borda de subida de SCLK.
17       ADC_CNVST_o  : out std_logic;   -- Indica se uma nova conversão deve ser
         iniciada. A conversão inicia na borda de descida.
18       ADC_CS_N_o   : out std_logic;   -- Em nível baixo hablita a interface serial.
19       ADC_SEL_o    : out std_logic;   -- Seleciona a entrada analógica primária ou
         secundária.
20       ADC_UB_o     : out std_logic;   -- Seleciona o modo Unipolar ou bipolar.
21       ADC_SD_o     : out std_logic;   -- Seleção entre single (conversão do canal 2
         somente após terminar o canal 1) e dual conversão paralela).
22       ADC_REFSEL_o : out std_logic;   -- Define o referencial interno ou externo para
         ADC
23       ADC_SCLK_o   : out std_logic);  -- Clock do ADC que configura o tempo de
         conversão.
24       data_1_o     : out ufixed (11 downto 0); --Resultado da conversão do sinal
         analógico para digital
25 end ADC;
26
27 architecture comportamento of ADC is
28
29 signal count        : integer range 0 to 16:= 0;  -- Variável auxiliar para contar
   os pulsos do SCLK e controlar a conversão.
30 signal preescaler   : integer range 0 to PRE:= 0; -- Inteiro com range de 0 a PRE,
   para dividir o clock.
31 signal sclk         : std_logic := '0';           -- Clock com preescaler.
32 signal busy         : std_logic := '0';           -- Flag que indica o estado do
   buffer.
33 signal data_1_buffer : ufixed (11 downto 0);      -- Buffer da recepção.
34
35 begin
36 ADC_SD_o <= sd_i;     -- A saída ADC_SD_o recebe o valor informado pela entrada sd_i
37 ADC_UB_o <= ub_i;     -- A saída ADC_UB_o recebe o valor informado pela entrada ub_i
38 ADC_REFSEL_o <= '0'; -- A saída ADC_REFSEL_o seleciona um referencial externo para o
   ADC
39 ADC_SCLK_o <= sclk;  -- A saída ADC_SCLK_o recebe o valor de sclk
```

```
40 ADC_CS_N_o <= csn_i; -- A saída ADC_CS_N_o recebe o valor informado pela entrada
   csn_i
41
42
43 process (clk_i, rst_i)
44 begin
45    if rst_i = '0' then  -- Se receber comando para resetar:
46       data_1_o    <= (others => '0'); -- Limpa as saídas digitais do ADC.
47       ADC_CNVST_o <= '0';  -- Liga conversão.
48       ADC_SEL_o   <= '0';  -- Seleciona os canais primários, AIN1A ou AIN2A.
49       sclk        <= '0';  -- Força uma borda de descida para sclk.
50       count       <=  0 ;  -- Reinicia count.
51
52    elsif rising_edge (clk_i) then   -- Se reset = 1, e a borda de subida do clock
      for identificada:
53       if csn_i = '0' then        -- Se ADC estiver habilitado no barramento SPI:
54          if busy = '0' then      -- Se o buffer está vazio:
55             ADC_CNVST_o <= '1'; -- Desliga a conversão.
56             busy <= '1';        -- Atualiza flag. Buffer em uso.
57          elsif busy = '1' then  -- Se o buffer está em uso, ou seja, se há um bit a
            ser transferido para o buffer:
58             if preescaler < PRE then         -- Se preescaler < PRE:
59                preescaler <= preescaler + 1; -- Incrementa preescaler.
60             end if;
61             if preescaler = PRE then  -- Se preescaler = PRE:
62                preescaler <= 0;       -- Reinicia preescaler.
63                sclk <= '1';           -- Gera uma borda de subida para sclk.
64                if count < 16 then     -- Se count < 16 (ainda não completou a
                  leitura do canal):
65                   count <= count + 1; -- Incrementa o contador
66                   data_1_buffer (11 downto 1) <= data_1_buffer(10 downto 0); --
                     Rotaciona os bits do buffer 1
67                   data_1_buffer(0) <= ADC_DOUT_i;  -- Carrega o buffer 1 com o
                     último bit recebido do canal AIN1A.
68                elsif count = 16 then         -- Se recebeu todos os bits da entrada
                  primária AIN1A.
69                   data_1_o <= data_1_buffer; -- Manda data_1_buffer para a saída
                     data_1_o.
70                   ADC_SEL_o <= '0'; -- Seleciona o canal primário AIN1A
71                   count <= 0;       -- Zera o contador
72                   busy <= '0';      -- Indica que o buffer está vazio novamente.
73                end if;
74             elsif preescaler  = PRE/2 then -- Se preescaler = PRE/2:
75             ADC_CNVST_o <= '0';            -- Religa a conversão.
76             sclk <= '0';                   -- Gera uma borda de descida para sclk.
77             end if;
78          end if;
79       end if;
80    end if;
81 end process;
82
83 end comportamento;
```

APÊNDICE B – CÓDIGO DO CONTROLADOR PID

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.fixed_float_types.all; -- Pacotes que possibilitam utilizar números não
   inteiros.
4  use ieee.fixed_pkg.all;
5
6  entity pid_control is
7  port(clk        : in std_logic;                -- Sinal de Clock
8       rst        : in std_logic;                -- Sinal de Reset
9       v_ref      : in sfixed  (7 downto -24);   -- Tensão de Referência
10      v_adc      : in sfixed  (7 downto -24);   -- Tensão de saída do conversor, lida
        pelo ADC
11      b0, b1, b2 : in sfixed  (12 downto -19); -- constantes do controlador
12      duty       : out sfixed (8 downto -23)); -- ciclo de trabalho 0 até 1
13
14 end entity;
15
16 architecture main of pid_control is
17
18 type erro_t is array(0 to 2) of sfixed(8 downto -24);  -- Define o tipo erro_t como
   um array de 3 posições do tipo sfixed para armazenar os dois últimos erros e o erro
   atual
19
20 type u_t is        array(0 to 1) of sfixed(7 downto -24);     -- Define o tipo u_t
   como um array de 2 posições do tipo sfixed para armazenar a saída anterior e a
   atual.
21
22 type temp_t is array(0 to 2) of sfixed(11 downto -24); -- Define o tipo temp_t como
   um array de 3 posições do tipo sfixed para armazenar a multiplicação das constantes
   pelos erros computados
23
24
25 signal erro : erro_t := (to_sfixed(0, 8, -24), to_sfixed(0, 8, -24), to_sfixed(0, 8,
   -24)); -- Declara erro como o tipo erro_t e inicializa todas as posições de erro com
   0
26
27 signal u    : u_t := (to_sfixed(0, 7, -24), to_sfixed(0, 7, -24)); -- Declara u como
   tipo u_t e inicializa todas as posições de u com 0
28
29 signal temp        : temp_t := (to_sfixed(0, 11, -24), to_sfixed(0, 11, -24),
   to_sfixed(0, 11, -24)); -- Declara temp como tipo temp_t e inicializa todas as
   posições de temp com 0
30
31
32 begin
33
34    erro(0) <= v_ref - v_adc; -- Define erro(0) como a diferença entre a tensão de
      referência e a tensão lida pelo ADC
35
36    temp(0) <= resize(b0*erro(0), temp(0)'high, temp(0)'low); -- recebe a
      multiplicação de b0 pelo erro no instante n (instante atual)
37
```

```
38    temp(1) <= resize(b1*erro(1), temp(1)'high, temp(1)'low); -- recebe a
      multiplicação de b1 pelo erro no instante n-1
39
40    temp(2) <= resize(b2*erro(2), temp(2)'high, temp(2)'low); -- recebe a
      multiplicação de b2 pelo erro no instante n-2
41
42    u(0) <= resize(temp(0) + temp(1) + temp(2) + u(1), u(0)'high, u(0)'low); -- A
      saída atual recebe a soma de todas as posições de temp, mais a saída computada
      anteriormente.
43
44
45    shift_reg: process(clk, rst, erro, u)
46    begin
47       if rst = '0' then                    -- Se o reset foi ativado
48          erro(1) <= to_sfixed(0, erro(1)); -- Zera o erro(1)
49          erro(2) <= to_sfixed(0, erro(2)); -- Zera o erro(2)
50          u(1)    <= to_sfixed(0, u(1));    -- Zera a saída anterior
51       else
52          if rising_edge(clk) then -- Se detectada a borda de subida
53             erro(1) <= erro(0);   -- erro(1) recebe o erro anterior
54             erro(2) <= erro(1);   -- erro(2) recebe o erro de dois instantes
               anteriores
55             u(1) <= u(0);         -- u(1) recebe a saída anterior
56          end if;
57       end if;
58    end process;
59
60    saturador: process(u(0))
61    variable duty_v        :        sfixed(8 downto -23) := to_sfixed(0, 8, -23); --
      Variável auxiliar para delimitação do duty cycle
62    begin
63       duty_v := resize(u(0) * to_sfixed(10, u(0)'high, u(0)'low), duty'high,
         duty'low); -- Multiplicar o valor da saída para que o valor de duty fique na
         ordem de 0 a 100%
64
65       if duty_v > to_sfixed(100, duty_v) then -- Se duty_v ultrapassar o valor
         máximo de 100%, então
66          duty_v := to_sfixed(100, duty_v);       -- duty_v recebe o valor 100%
67       elsif duty_v < to_sfixed(0, duty_v) then -- Se duty_v for menor que 0, então
68          duty_v := to_sfixed(0, duty_v);       -- duty_v recebe o valor 0
69       else
70          null;
71       end if;
72       duty <= duty_v; -- A saída duty recebe a variável auxiliar duty_v
73    end process;
74
75 end architecture;
```

APÊNDICE C – CÓDIGO DO PWM

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4
5  entity PWM is
6  generic (PRE:integer:=500); -- Divisor de Frequência
7
8  port( duty     : in integer;      -- Duty Cycle de entrada
9        clock_in : in std_logic;    -- Sinal de clock de entrada
10       reset    : in std_logic;    -- Sinal de Reset
11       PWM_out  : out std_logic); -- Sinal de PWM de saída.
12
13 end PWM;
14
15
16 architecture pwm_signal of PWM is
17
18 component Prescaler    -- Declaração do componente Prescaler
19 generic (PRE:integer); -- Valor para dividir a frequência
20
21 port( clock_in:in std_logic;             -- Sinal de clock de entrada (50 MHz)
22       reset: in std_logic;                -- Sinal de Reset
23       prescaler_signal:buffer std_logic); -- Sinal de clock dividido
24 end component;
25
26 signal pre_in : std_logic; -- Sinal auxiliar para receber o sinal de clock dividido
27
28 begin
29    instance_label : Prescaler -- Instanciação do componente Prescaler
30    generic map (PRE => PRE)   -- Recebe o valor da entity
31    port map (clock_in         => clock_in, -- O clock do prescaler recebe o clock da
      entity do PWM
32              reset            => reset,    -- O sinal de reset do prescaler recebe o
                reset do PWM
33              prescaler_signal => pre_in);  -- O prescaler_signal é enviado para o
                sinal auxiliar pre_in
34
35
36
37         process(pre_in, reset, duty)
38
39    variable cont1 : integer := 100 - duty; -- Variável para auxiliar na contagem dos
      pulsos para ciclo off
40    variable cont2 : integer := 0;          -- Variável para auxiliar na contagem dos
      pulsos para o ciclo on
41    variable flag : std_logic:= '0';        -- Flag para indicar o início e fim do
      ciclo ativo:'0' -> Ciclo on e '1'-> ciclo off.
42
43         begin
44
45       if(reset = '0') then -- Se houver o reset
46          cont1 := 0;       -- Zera o cont1
47          cont2 := 0;       -- Zera o cont2
```

```
48          PWM_out <= '0';    -- A saída do PWM fica em nível baixo
49
50       elsif rising_edge (pre_in) then -- Se for identificada a borda de subida do
         sinal pre_in
51          if(cont1 > 0) then           -- Se o cont1 ainda estiver carregado
52             cont1 := cont1 - 1;       -- Decrementa a variável
53          end if;
54          if(cont1 = 0 and flag = '0') then -- Se acabou a contagem de cont1 e ainda
            não começou a contagem de cont2
55             PWM_out <='1';                 -- A saída vai para nível alto
56             cont2:= duty;                  -- A variável cont2 recebe o valor do
               ciclo de trabalho
57             flag := '1';                   -- A flag em nível alto indica o final do
               ciclo off e início do ciclo on
58          end if;
59          if(cont2 > 0) then                -- Se o cont2 ainda estiver carregado.
60             cont2 := cont2 - 1;            -- Decrementa a variável.
61          end if;
62          if(cont2 = 0 and flag = '1') then -- Se cont2 e cont1 forem iguais a zero.
63             PWM_out <='0';                 -- A saída vai para nível baixo
64             cont1:= 100 - duty;            -- Recarrega o contador do ciclo off
65             flag := '0';                   -- Zera a flag para indicar o fim do
               ciclo on e início do ciclo off
66          end if;
67       end if;
68    end process;
69
70
71 end pwm_signal;
```

APÊNDICE D – CÓDIGO DO PRESCALER

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4
5  entity Prescaler is
6  generic (PRE:integer); -- Prescaler para reduzir o clock da entrada de 50 MHz
7
8  port(        clock_in         : in std_logic;    -- Sinal de clock de entrada
9        reset            : in std_logic;    -- Sinal de reset
10       prescaler_signal : out std_logic); -- Sinal de saída, com período N vezes
         maior.
11 end Prescaler;
12
13
14 architecture divide of Prescaler is
15 signal temp : std_logic:= '0'; -- Sinal auxiliar para receber o sinal que será
   repassado para a saída.
16
17 begin
18    process(clock_in, reset)
19    variable cont : integer := 0; -- Variável para contar quantas vezes a borda de
      subida do clock de entrada foi identificada.
20
21    begin
22       if(reset = '0') then -- Se houver o reset
23          cont := 0;         -- Zera o contador
24          temp <= '0';       -- Zera o sinal auxiliar
25
26       elsif rising_edge(clock_in) then -- Se borda de subida
27          cont := cont + 1;             -- O contador é incrementado
28          if cont = PRE/2 then          -- Se o contador de borda de subida for igual
            a metade da relação entre o período do sinal de saída e o de entrada
29             temp <= not temp;          -- A variável auxiliar recebe o valor oposto
               ao anterior.
30             cont := 0;                 -- Zera o contador.
31          end if;
32       end if;
33    end process;
34    prescaler_signal <= temp; -- O sinal de prescaler recebe a variável auxiliar.
35
36 end architecture;
```

APÊNDICE E – CÓDIGO PRINCIPAL

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.fixed_float_types.all; -- Pacotes que possibilitam utilizar números não
   inteiros.
4  use ieee.fixed_pkg.all;
5
6  entity Projeto_Final is
7  generic(PRE : integer := 500);
8          port(
9           clock       : in std_logic;     -- Sinal de Clock, gerado pelo Kit
           Mercurio         reset       : in std_logic;     -- Sinal de Reset
10          PWM_out     : out std_logic;    -- PWM a ser enviado ao circuito
11
12          ----------Referente ao ADC----------------------------
13          ADC_DOUT    : in  std_logic;          -- Saída do dado Serial 1. Dado é
           recebido na borda de subida de SCLK.
14          ADC_CNVST   : out std_logic;          -- Indica se uma nova conversão deve
           ser iniciada.
15          ADC_CS_N    : out std_logic;         -- Em nível baixo habilita a interface
           serial.
16          ADC_SEL     : out std_logic;         -- Seleciona a entrada analógica
           primária ou secundária.
17          ADC_UB      : out std_logic;         -- Seleciona o modo Unipolar ou bipolar.
18          ADC_SD      : out std_logic;         -- Seleção entre single e dual.
19          ADC_REFSEL : out std_logic;          -- Define o referencial interno ou
           externo para ADC
20          ADC_SCLK    : out std_logic);         -- Clock do ADC que configura o tempo
           de conversão.
21
22 end entity;
23
24 architecture main of Projeto_Final is
25 -----------------Declaração do Componente ADC-------------------------
26 component ADC
27 generic (PRE:integer);
28 port(
29          clk_i : in std_logic; -- Recebe o clock que será enviado ao ADC.
30          rst_i : in std_logic; -- Recebe o comando para resetar o ADC.
31          sd_i  : in std_logic; -- Recebe o comando para selecionar saída
           Single/Dual.
32          ub_i  : in std_logic; -- Recebe o comando para selecionar entrada
           Unipolar/Bipolar.
33          csn_i : in std_logic; -- Recebe o comando para habilitar chip no barramento
           SPI.
34
35
36      data_1_o      : out ufixed(11 DOWNTO 0); -- Resultado da conversão do sinal
       analógico para digital
37
38
39       ADC_DOUT_i    : in  std_logic;  -- Saída do dado Serial 1. Dado é recebido na
        borda de subida de SCLK.
```

```
40        ADC_CNVST_o   : out std_logic;  -- Indica se uma nova conversão deve ser
          iniciada.
41        ADC_CS_N_o    : out std_logic;  -- Em nível baixo hablita a interface serial.
42        ADC_SEL_o     : out std_logic;  -- Seleciona a entrada analógica primária ou
          secundária.
43        ADC_UB_o      : out std_logic;  -- Seleciona o modo Unipolar ou bipolar.
44        ADC_SD_o      : out std_logic;  -- Seleção entre single e dual.
45        ADC_REFSEL_o  : out std_logic;  -- Define o referencial interno ou externo para
          ADC
46        ADC_SCLK_o    : out std_logic); -- Clock do ADC que configura o tempo de
          conversão.
47  end component;
48
49  -----------------Declaração do Componente PID-------------------------
50  component pid_control
51          port( clk      : in std_logic;                -- Sinal de Clock
52           rst       : in  std_logic;               -- Sinal de Reset
53           v_ref     : in  sfixed (7 downto -24);  -- Tensão de Referência
54           v_adc     : in  sfixed (7 downto -24);  -- Tensão de saída do conversor,
             lida pelo ADC
55           b0, b1, b2: in  sfixed (12 downto -19); -- constantes do controlador
56           duty      : out sfixed (7 downto -24)); -- ciclo de trabalho 0 até 1
57  end component;
58
59  --------------------Declaração do Componente PWM----------------------
60  component PWM
61  generic (PRE:integer); -- Divisor de Frequência
62
63  port( duty     : in integer;     -- Duty Cycle de entrada
64        clock_in : in std_logic;   -- Sinal de clock de entrada
65        reset    : in std_logic;   -- Sinal de Reset
66        PWM_out  : out std_logic); -- Sinal de PWM de saída.
67
68  end component;
69  -------------------Declaração do Componente PRESCALER-----------------
70  component Prescaler    -- Declaração do componente Prescaler
71  generic (PRE:integer); -- Valor respassado para dividir a frequência.
72
73  port( clock_in          : in std_logic;      -- Sinal de clock de entrada (50 MHz)
74        reset             : in std_logic;      -- Sinal de Reset
75        prescaler_signal  : buffer std_logic); -- Sinal de clock dividido
76
77  end component;
78  ----------------Fim das declarações dos componentes-----------------------
79
80  ---------------Declaração dos Sinais Auxiliares -------------------------
81  signal amostra_original : ufixed(11 downto 0) := "000000000000"; -- Sinal para
    receber o valor lido do ADC
82  signal auxiliar0        : integer;               -- Sinal auxiliar para converter a
    amostra_original para o tipo inteiro
83  signal auxiliar1        : sfixed(16 downto -16); -- Sinal auxiiar para conversão
    para o tipo sfixed.
84  signal auxiliar2        : sfixed(16 downto -16); -- Sinal auxiliar para converter o
    sinal auxiliar para o tipo s_fixed
85  signal amostra_adaptada : sfixed(8 downto -23);  -- Amostra que recebe o valor da
    tensão com o divisor resistivo e o restaura para se obter o valor da tensão real
86  signal duty_pid         : sfixed(7 downto -24);  -- Sinal que recebe o duty que vem
    do conversor PID
```

```
87  signal duty_pwm         : integer;                -- Sinal que recebe o duty que será
    enviado ao PWM
88  signal Vin              : sfixed (8 downto -23); -- Tensão de entrada
89  signal v_ref            : sfixed(8 downto -23);  -- Tensão de Referência, vinda do
    PC
90  signal b0, b1, b2       : sfixed(12 downto -19); -- Constantes do controlado PID
91  signal pre_in           : std_logic;              -- Sinal auxiliar para receber o
    sinal de clock dividido
92
93  begin
94
95  process(pre_in) --lista de sensibilidade
96     begin
97        if rising_edge (pre_in) then
98           duty_pwm <= to_integer(duty_pid); -- Converte o sinal que vem do PID para o
             tipo inteiro
99           Vin <= to_sfixed(15, Vin); --Convertendo a tensão de entrada para o tipo
             sfixed
100          v_ref <= to_sfixed(7.5, v_ref); --Convertendo a tensão de referência para o
             tipo sfixed
101
102          b0 <= to_sfixed(1075.65, b0); -- A constante b0 do PID recebe o seu valor
103          b1 <= to_sfixed(-75.65, b1);  -- A constante b1 do PID recebe o seu valor
104          b2<= to_sfixed(0.00147, b2);  -- A constante b2 do PID recebe o seu valor
105
106          auxiliar0 <= to_integer(amostra_original);          -- Sinal auxiliar para
             converter a amostra_original do tipo ufixed para o tipo inteiro
107          auxiliar1 <= to_sfixed(auxiliar0, 16, -16);         -- Sinal auxiliar
             converter para o tipo sfixed.
108          auxiliar2 <= resize(auxiliar1/4095, 16, -16);              -- Sinal
             auxiliar para dividr o valor advindo da leitura do ADC (devidamente
             convertido para o tipo desejado) por 4095 (2^12 - 1)
109          amostra_adaptada <= resize((auxiliar2*15), 8, -23); -- Recebe o sinal
             multiplicado pela tensão de entrada, para que o valor da tensão possa ser
             restaurado para Volts
110
111 end if;
112 end process;
113
114 ---------Instanciação do Componente ADC----------------
115 Inst_ADC: ADC
116 generic map (PRE => PRE)
117         port map
118         (
119      clk_i => clock, -- Recebe o sinal de clock.
120      rst_i => reset, -- Recebe o sinal de reset .
121      sd_i  => '0',   -- Selecionado para modo dual
122      ub_i  => '0',   -- Selecionado para entrada unipolar
123      csn_i => '0',   -- Selecionado para habilitar ADC no barramento SPI.
124
125      data_1_o => amostra_original,        -- A variável amostra_original recebe o
         valor lido do ADC
126
127      ADC_DOUT_i   => ADC_DOUT,   -- Recebe a entrada ADC_OUT da entidade top-level.
128      ADC_CNVST_o  => ADC_CNVST,  -- Envia para a saída ADC_CNVST da entidade
         top-level.
129      ADC_CS_N_o   => ADC_CS_N,   -- Envia para a saída ADC_CS_N da entidade
         top-level.
```

```
130       ADC_SEL_o     => ADC_SEL,     -- Envia para a saída ADC_SEL da entidade
          top-level.
131       ADC_UB_o      => ADC_UB,      -- Envia para a saída ADC_UB da entidade top-level.
132       ADC_SD_o      => ADC_SD,      -- Envia para a saída ADC_SD da entidade top-level.
133       ADC_REFSEL_o         => ADC_REFSEL, -- Envia para a saída ADC_REFSEL da entidade
          top-level.
134       ADC_SCLK_o    => ADC_SCLK);  -- Envia para a saída ADC_SCLK da entidade
          top-level.
135
136   ---------Instanciação do Componente PID----------
137   Inst_PID : pid_control
138           port map(
139                clk => pre_in,  -- recebe o sinal de clock que vem do Prescaler
140                rst => reset,   -- recebe o reset
141                v_ref => v_ref, -- recebe a tensão de referência
142                v_adc => amostra_adaptada, -- recebe a tensão lida do ADC
143                b0 => b0,          -- recebe as constantes
144                b1 => b1,
145                b2 => b2,
146                duty => duty_pid); -- recebe o sinal que será enviado como duty cycle
                   para o PWM
147
148
149
150   --------Instanciação do Componente PWM-------------
151   Inst_PWM: PWM
152   generic map(PRE => PRE)
153           port map (
154                 clock_in => clock,   -- recebe o sinal de clock de 50 MHz
155                 duty     => duty_pwm,-- Recebe o duty advindo do PID, convertido para
                    um valor inteiro na ordem de 0 a 100.
156                 reset    => reset,   -- Recebe o valor de reset
157                 PWM_out  => PWM_out);-- Envia o valor de PWM
158
159   ------Instanciação do Componente PRESCALER---------------
160   instance_label : Prescaler         -- Instanciação do componente Prescaler
161   generic map (PRE => PRE)
162   port map (
163              clock_in         => clock,   -- O clock do prescaler recebe o clock da
               entity do PWM
164              reset            => reset,   -- O sinal de reset do prescaler recebe o
               reset do PWM
165              prescaler_signal => pre_in); -- O prescaler_signal é enviado para o sinal
               auxiliar pre_in
166
167   end architecture;
```

APÊNDICE F – ESQUEMÁTICO DO CIRCUITO DO CONVERSOR *BUCK*