

**UNIVERSIDADE DO ESTADO DE SANTA CATARINA – UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS – CCT
BACHARELADO EM ENGENHARIA ELÉTRICA**

RICARDO ZOLET

**CONTROLE DIGITAL DE INVERSOR MULTINÍVEL NPC_m IMPLEMENTADO
TOTALMENTE EM FPGA**

**JOINVILLE,SC
2014**

RICARDO ZOLET

**CONTROLE DIGITAL DE INVERSOR MULTINÍVEL NPC_m IMPLEMENTADO
TOTALMENTE EM FPGA**

Trabalho de Conclusão apresentado ao Curso de Bacharelado em Engenharia Elétrica do Centro de Ciências Tecnológicas, da Universidade do Estado de Santa Catarina, como requisito final para a obtenção do grau de Bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Joselito Anastácio
Heerdt

**JOINVILLE,SC
2014**

RICARDO ZOLET

**CONTROLE DIGITAL DE INVERSOR MULTINÍVEL NPC_m IMPLEMENTADO
TOTALMENTE EM FPGA**

Trabalho de Conclusão apresentado ao Curso de Bacharelado em Engenharia Elétrica do Centro de Ciências Tecnológicas da Universidade do Estado de Santa Catarina, como requisito final para obtenção do grau de Bacharel em Engenharia Elétrica.

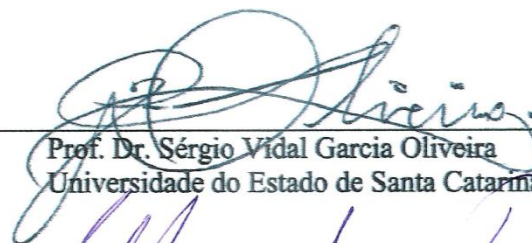
Banca Examinadora

Orientador:



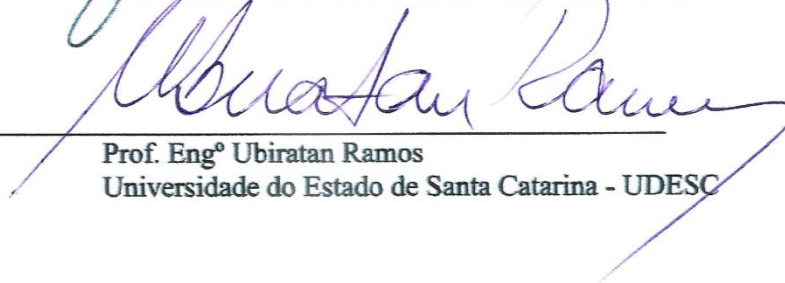
Prof. Dr. Joselito Anastácio Heerd
Universidade do Estado de Santa Catarina – UDESC

Membro:



Prof. Dr. Sérgio Vidal Garcia Oliveira
Universidade do Estado de Santa Catarina – UDESC

Membro:



Prof. Engº Ubiratan Ramos
Universidade do Estado de Santa Catarina - UDESC

Joinville, SC, 03/12/2014

AGRADECIMENTOS

A Deus por ter me dado as condições necessárias para superar as dificuldades.

À minha esposa Dieyd Zolet por todo apoio, amor e compreensão.

Aos meus pais, Ivete R. Zolet e Moacir Zolet, pelas palavras de encorajamento, de fé e por todo apoio e carinho.

Um agradecimento especial ao meu orientador, Prof. Dr. Eng^o Joselito Anastácio Heerdt por me conceder essa oportunidade, por compartilhar um pouco do seu conhecimento e pela paciência e dedicação durante a orientação.

Ao Prof. Dr. Eng^o Antônio da Silva Silveira pela grande vontade em ajudar.

Aos doutorando e mestrando de Engenharia Elétrica da UDESC: Marcos Vinicius Bressan e Gustavo Lambert, respectivamente, pelos esclarecimentos e pela cooperação.

Ao meu amigo Daniel dos Santos Matos, pela enorme contribuição neste trabalho.

A todos que direta ou indiretamente fizeram parte da minha formação.

RESUMO

ZOLET, Ricardo. **Controle digital de inversor multinível NPCm implementado totalmente em FPGA**. Trabalho de Conclusão de Curso (Bacharelado em Engenharia Elétrica – Área: Eletrônica de Potência) – Universidade do Estado de Santa Catarina. Joinville, 2014.

Este trabalho de conclusão de curso, apresenta o projeto e a implementação de um controlador PID digital, embarcado em um FPGA, para controlar a tensão de um conversor NPCm de três níveis monofásico e trifásico, avaliando a capacidade do FPGA presente no kit de desenvolvimento DE0 – Nano da fabricante Altera, em atender as demandas destas duas arquiteturas. Também é apresentado o projeto e a implementação de um controle adaptativo, método *Indirect Self Tuning*, para controlar a tensão de um conversor NPCm de três níveis monofásico, avaliando a capacidade do FPGA presente no kit, em atender as necessidades desta arquitetura. Neste contexto, é relatado também, o projeto desenvolvido para efetuar a comunicação entre o FPGA e o conversor A/D contidos no kit DE0 – Nano, além de mostrar exemplos de como utilizar alguns recursos do VHDL 2008.

Palavras-chave: FPGA, controlador PID digital, controle adaptativo, conversor NPCm, VHDL 2008

LISTA DE ILUSTRAÇÕES

Figura 1 - Arquitetura típica de um FPGA	18
Figura 2 – Estrutura de uma LE do dispositivo Cyclone IV	19
Figura 3 – Arquitetura de um bloco multiplicador no modo (18×18)	21
Figura 4 – Arquitetura de um bloco multiplicador no modo (9×9)	22
Figura 5 – Rede de <i>clock</i>	23
Figura 6 – Bloco de controle de <i>clock</i>	23
Figura 7 – Kit DE0 – Nano da fabricante Altera	25
Figura 8 - Sequência de etapas do processo de um projeto com FPGA	26
Figura 9 – Etapas principais de síntese lógica	27
Figura 10 – (a) Estrutura do conversor NPC; (b) Estrutura do conversor NPCm	32
Figura 11 - Plano $V_A \times I_A$	33
Figura 12 - Etapas de operação para o 1° e o 2° quadrantes	34
Figura 13 – Formas de onda para o 1° quadrante	35
Figura 14 – Etapas de operação do 1° e 3° quadrante	36
Figura 15 - Modulação tipo POD: (a) moduladora senoidal e portadoras ; (b) tensão de saída V_A	38
Figura 16 – Estrutura do conversor monofásico NPCm	39
Figura 17 - Forma de onda no ponto A do conversor NPCm	40
Figura 18 – Circuito equivalente da saída do conversor	40
Figura 19 - Circuito equivalente da saída do conversor em Laplace	40
Figura 20 - Circuito equivalente da saída do conversor com indicação da tensão de saída	42
Figura 21 - Estrutura de um controlador PID na versão digital	44
Figura 22 - Lugar das raízes da planta discretizada	48
Figura 23 – Resposta a entrada degrau das plantas em malha aberta	48
Figura 24 – Lugar das raízes do sistema com o controlador e a planta	49
Figura 25 – Controlador obtido no ambiente de projeto SISO	50
Figura 26 – Resposta ao degrau do sistema $CzGpwmGplanta(z)$ em malha fechada	51
Figura 27 – Diagrama de bode do sistema $CzGpwmGplanta(z)$	52
Figura 28 – Diagrama de simulação do modelo matemático, usando o Simulink	53
Figura 29 – Resultado da simulação do modelo matemático com mudança de carga	54
Figura 30 – Resultado da simulação do modelo matemático com mudança de carga (<i>zoom</i>)	54

Figura 31 – Sistema para simulação do modelo comutado, usando PSIM®: (1) Conversor NPCm com filtro LC e dispositivo de chaveamento de carga; (2) Circuito de condicionamento do sinal de tensão medido; (3) Malha de controle; (4) Circuito modulador; (5) Circuito para simulação em malha aberta; (6) Circuito para simulação em malha aberta	56
Figura 32 – Forma da tensão na saída do circuito de condicionamento.....	58
Figura 33 – Resultado da simulação do modelo comutado com mudança de carga	59
Figura 34 – Resultado da simulação do modelo comutado com mudança de carga (zoom)	59
Figura 35 – Saída do sistema controlado V _{out} e saída do sistema em malha aberta V _{MA} ..	60
Figura 36 – Resultado da simulação para a tensão no ponto V _{PWM} da Figura 31(1)	60
Figura 37 – Resultado da simulação para a tensão no ponto V _{PWM} da Figura 31.1 (zoom) ..	61
Figura 38 – Resultado da simulação para os sinais dos canais PWM	61
Figura 39 – Resultado da simulação da modulação POD (semiciclo positivo).....	62
Figura 40 – Resultado da simulação da modulação POD (semiciclo negativo).....	62
Figura 41 - Diagrama em blocos do controle adaptativo indireto	63
Figura 42 – Rastreamento do sinal de referência senoidal	69
Figura 43 – Ação do controlador mediante ao distúrbio	70
Figura 44 – Diagrama de blocos do conversor A/D do kit DE0 - Nano.....	71
Figura 45 – Conexão entre o conversor A/D, o FPGA e o conector	73
Figura 46 – Rotinas escritas em VHDL para comunicação do FPGA com o conversor A/D..	74
Figura 47 – Indicação do instante da amostragem (controlador <i>Indirect Self Tuning</i>)	78
Figura 48 – Indicação do instante da amostragem (controlador PID digital).....	78
Figura 49 – Diagrama de temporização (conversor A/D)	79
Figura 50 – Sinal aplicado ao canal IN0.....	80
Figura 51 – Resultado da conversão A/D	81
Figura 52 – Algoritmo para mudança de base (decimal para binário) da parte fracionária	82
Figura 53– Código desenvolvido no <i>software</i> MATLAB® para encontrar o número de bits necessários para representar com precisão em binário a parte fracionária de um número em decimal	83
Figura 54 – Representação gráfica dos bits necessários para representar com precisão os coeficientes da Equação (48).....	84
Figura 55 - Código desenvolvido no MATLAB® para gerar a Tabela 12.....	85
Figura 56 – Código em VHDL para a quantização dos sinais	87
Figura 57 – Código em VHDL que realiza a conversão de bases através da função “to_sfixed”	88

Figura 58 – Comportamento do parâmetro estimado $\hat{a}_1$ na simulação	89
Figura 59 – Comportamento do parâmetro estimado $\hat{a}_2$ na simulação	89
Figura 60 – Comportamento do parâmetro estimado $\hat{b}_0$ na simulação	90
Figura 61 – Comportamento do parâmetro estimado $\hat{b}_1$ na simulação	90
Figura 62 – Comportamento do parâmetro estimado $\hat{b}_1$ na simulação (discreto).....	91
Figura 63 – Comportamento do parâmetro estimado $\hat{b}_1$ na simulação (discreto - zoom)...	91
Figura 64 – Comportamento do parâmetro de sintonia s_0 na simulação.....	92
Figura 65 – Comportamento do parâmetro de sintonia s_0 na simulação (zoom).....	92
Figura 66– Representação do número de bits para representar a parte fracionária dos valores máximos e mínimos dos parâmetros estimados “a”	93
Figura 67 – Representação do número de bits para representar a parte fracionária dos valores máximos e mínimos dos parâmetros estimados “b”	93
Figura 68 – Representação do número de bits para representar a parte fracionária dos valores máximos e mínimos do parâmetro de sintonia “ s_0 ”	94
Figura 69 – Valores da matriz de covariância obtidos na simulação	94
Figura 70– Representação do número de bits para representar a parte fracionária do menor valor da matriz apresentada na Figura 69	95
Figura 71 – Código em VHDL que realiza a quantização dos sinais no projeto do controlador <i>Indirect Self Tuning</i>	96
Figura 72 - Fluxograma de modulação PWM para conversor NPC três níveis monofásico com o com o controlador.....	100
Figura 73 – Fluxograma da conversão A/D (geração do <i>clock</i> do conversor A/D e definição do canal)	101
Figura 74 – Fluxograma da conversão A/D (obtenção do valor amostrado e eliminação do <i>offset</i>)	102
Figura 75 – Fluxograma da rotina de cálculo da lei de controle	103
Figura 76 – Rotina em VHDL que executa o <i>loop</i> de controle	103
Figura 77 – Estrutura de um conversor NPCm trifásico	105
Figura 78 – Lógica escrita em VHDL para gerar a defasagem entre as fases de um sistema trifásico	106
Figura 79 – Código escrito em VHDL para a modulação da fase S, ressaltando a condição que permite a geração da mesma.....	107
Figura 80 – Detalhes do código em VHDL para controle do sistema trifásico.....	108
Figura 81 – Indicações dos instantes de amostragem para o sistema trifásico.....	108

Figura 82 – Código em VHDL para as definições dos instantes de amostragem	109
Figura 83 – Rotina em VHDL para a escolha do canal do conversor A/D	110
Figura 84 – Fluxograma do algoritmo habilita o cálculo do MQR	115
Figura 85 – Fluxograma do MQR	116
Figura 86 – Fluxograma do cálculo da lei de controle	117
Figura 87 – Vista superior do sistema: (a) Conversor NPCm 1 ϕ ; (b) Fontes auxiliares; (c) Placa de condicionamento e interface; (d) Kit DE0-Nano.	118
Figura 88 – Vista lateral do sistema: (a) Conversor NPCm 1 ϕ ; (b) Fontes auxiliares; (c) Placa de condicionamento e interface; (d) Kit DE0-Nano.....	119
Figura 89 – Instante da obtenção do valor da amostra	120
Figura 90 - Atraso entre a solicitação de amostragem e a obtenção do valor da amostra.....	120
Figura 91 - Instante do fim do cálculo da lei de controle.....	121
Figura 92 - Atraso entre o instante da amostragem e o fim do cálculo da lei de controle	121
Figura 93 – Defasagem entre a fase R e a fase S.....	122
Figura 94 – Defasagem entre a fase R e a fase T	123
Figura 95 – Instante da liberação da fase S e fase S.....	123
Figura 96 – Fase S em malha fechada	124
Figura 97 – Redução da carga	125
Figura 98 – Elevação da tensão na redução da carga	126
Figura 99– Acréscimo de carga.....	126
Figura 100 – Redução da tensão no acréscimo de carga	127
Figura 101 – Relatório referente ao <i>hardware</i> de controle projetado para o sistema monofásico	128
Figura 102 – Relatório referente ao <i>hardware</i> de controle projetado para o sistema monofásico (<i>zoom</i>)	128
Figura 103 – Relatório referente ao <i>hardware</i> de controle projetado para o sistema trifásico	129
Figura 104 – Relatório referente ao <i>hardware</i> de controle projetado para o sistema trifásico (<i>zoom</i>).....	129
Figura 105 – Relatório referente ao <i>hardware</i> de controle (Indirect Self Tuning) projetado para o sistema monofásico.....	130
Figura 106 – Relatório referente ao <i>hardware</i> de controle (Indirect Self Tuning) projetado para o sistema monofásico (<i>zoom</i>)	131
Figura 107 – Rotina que calcula o sinal de controle (Controlador com 2 polos e 2 zeros)....	132

Figura 108 – Relatório referente à rotina que calcula o sinal de controle (Controlador com 2 polos e 2 zeros).....	133
Figura 109 – Rotina que calcula o sinal de controle (Controlador com 1 polo1 e 1 zero1)...	134
Figura 110 – Relatório referente à rotina que calcula o sinal de controle (Controlador com 1 polo e 1 zero).....	135

LISTA DE TABELAS

Tabela 1 – Largura de dados do bloco M9K	20
Tabela 2 – Número de multiplicadores embutidos no dispositivo Cyclone IV	21
Tabela 3 - Características do kit de desenvolvimento DE0 – Nano	24
Tabela 4 - Resumos dos pacotes e dados usados em VHDL	29
Tabela 5 - Operadores básicos e tipos predefinidos em VHDL	29
Tabela 6 – Apoio para definição do tamanho dos campos após operações matemáticas.....	31
Tabela 7 – Tensão de saída em função do estado das chaves do conversor NPCm.....	37
Tabela 8 – Parâmetros do conversor NPCm utilizado.....	47
Tabela 9 – Descrição dos principais pinos do conversor A/D do kit DE0 - Nano	72
Tabela 10 – Identificação dos pinos do FPGA que estão conectados ao conversor A/D.....	73
Tabela 11 – Bits do Registrador de Controle (Conversor A/D)	79
Tabela 12 – Seleção do Canal de Entrada	79
Tabela 13 – Comparação entre o sinal de entrada e o sinal convertido.....	81
Tabela 14 – Peso em decimal dos bits das posições -1 à -17 com 15 casas de precisão	85

LISTA DE ABREVIATURAS E SIGLAS

ADC	Analog-to-Digital Converter
A/D	Analógico/Digital
ASICs	Application Specific Integrated Circuits
CA	Corrente Alternada
CC	Corrente Contínua
CLBs	Configurable Logic Blocks
D/A	Digital/Analógico
DHT	Distorção Harmônica Total
DSP	Digital Signal Processor
EDA	Electronic Design Automation
FPGA	Field Programmable Gate Array
FT	Função Transferência
GPIO	General Purpose Input Output
IEEE	Institute of Electrical and Electronic Engineers
IOBs	Input/Output Blocks
LEs	Logic Elements
LSB	Least Significant Bit
LUT	Lookup Table
MAC	Multiply and Accumulate
MQ	Mínimos Quadrados
MQR	Mínimos Quadrados Recursivos
MSB	Most Significant Bit
NPC	Neutral Point Clamped
NPCm	Modified Neutral Point Clamped
PID	Proporcional Integral Derivativo
PLL	Phase Locked Loop
POD	Phase Opposition Disposition
PWM	Pulse Width Modulation
SISO	Single Input Single Output
VHDL	Very High Speed Integrated Circuit Hardware Description Language
ZOH	Zero Order Hold

SUMÁRIO

1 INTRODUÇÃO	16
2 HARDWARE DE DISPOSITIVOS FPGA	18
2.1 ARQUITETURA DO FPGA.....	18
2.1.1 Blocos lógicos configuráveis.....	19
2.1.2 Blocos de memória.....	20
2.1.3 Blocos multiplicadores	20
2.1.4 Blocos DSP	22
2.1.5 Gerenciamento de <i>clock</i>	22
2.2 KIT DE DESENVOLVIMENTO DE0-NANO DA ALTERA.....	24
3 PROGRAMAÇÃO DE DISPOSITIVOS FPGA	26
3.1 ETAPAS DE UM PROJETO UTILIZANDO FPGA	26
3.2 PROGRAMAÇÃO NO FPGA	26
3.2.1 Síntese do VHDL	28
3.2.2 VHDL 2008.....	28
4 CONVERSOR NPC_m DE TRÊS NÍVEIS	32
4.1 QUADRANTES DE OPERAÇÃO DO CONVERSOR NPC _m	33
4.2 NPC _m OPERANDO COMO INVERSOR	35
4.3 ESCOLHA DA ESTRATÉGIA DE MODULAÇÃO	37
5 PROJETO DOS SISTEMAS DE CONTROLE E RESULTADOS DE SIMULAÇÃO	39
5.1 MODELAGEM DESTINADA AO CONTROLE DE TENSÃO DO CONVERSOR NPC _m MONOFÁSICO COM FILTRO LC.....	39
5.2 CONTROLADOR PID DIGITAL	43
5.3 PROJETO DO CONTROLE DE TENSÃO DO CONVERSOR NPC _m	45
5.3.1 Projeto do controlador digital no plano Z.....	46
5.4 COMPROVAÇÃO DO MODELO MATEMÁTICO	52

5.5 SINAIS RELEVANTES DO SISTEMA SIMULADO	60
5.6 CONTROLE DIGITAL ADAPTATIVO	62
5.6.1 Mínimos quadrados (MQ)	64
5.6.2 Mínimos quadrados recursivo (MQR)	66
5.6.3 PID LIM <i>self tuning</i>	67
5.6.4 Simulação do controlador <i>indirect self tuning</i>	69
6 IMPLEMENTAÇÃO DOS SISTEMAS DE CONTROLE	71
6.1 CONVERSOR A/D DO KIT DE0-NANO	71
6.1.1 Descrição do código em vhdl para a comunicação serial	74
6.1.2 Validação do código para a comunicação serial	80
6.2 QUANTIZAÇÃO DOS DADOS NO FPGA	82
6.2.1 Quantização aplicada no código que implementa o PID digital.....	82
6.2.2 Quantização aplicada no código que implementa o <i>indirect self tuning</i>	88
6.3 CONTROLE PID DIGITAL DA TENSÃO DE SAÍDA DO CONVERSOR NPCm MONOFÁSICO	98
6.3.1 Algoritmo de programação	99
6.4 CONTROLE PID DIGITAL DAS TENSÕES DE SAÍDA DO CONVERSOR NPCm TRIFÁSICO	104
6.4.1 Defasagem, amostragem das 3 fases e cálculo da lei de controle	105
6.5 IMPLEMENTAÇÃO DO <i>INDIRECT SELF TUNING</i> NO FPGA	110
6.5.1 Implementação do MQR.....	110
6.5.2 Implementação da lei de controle do PID digital recursivo	116
7 RESULTADOS EXPERIMENTAIS	118
7.1 ATRASOS	119
7.2 DEFASAGEM ENTRE AS FASES.....	122
7.3 TENSÃO DE SAÍDA EM MALHA FECHADA	124
7.4 CONSTATAÇÃO DA AÇÃO DE CONTROLE.....	125
7.5 RELATÓRIOS GERADOS PELO QUARTUS II®	127

7.6 CONSIDERAÇÕES SOBRE O CONTROLE	135
8 CONCLUSÕES GERAIS	137
REFERÊNCIAS	139
APÊNDICE A – CÓDIGO EM VHDL QUE IMPLEMENTA O CONTROLADOR PID PARA O NPC_m MONOFÁSICO	142
APÊNDICE B – CÓDIGO EM VHDL QUE IMPLEMENTA O CONTROLADOR PID PARA O NPC_m TRIFÁSICO	155
APÊNDICE C – CÓDIGO EM VHDL QUE IMPLEMENTA O MÉTODO DE CONTROLE <i>INDIRECT SELF TUNING</i>	180
APÊNDICE D – CÓDIGO PARA SIMULAÇÃO DO INDIRECT SELF TUNING NO SOFTWARE MATLAB®.....	206

1 INTRODUÇÃO

A exigência por uma energia confiável e de qualidade é crescente. No âmbito da eletrônica de potência, as topologias de conversores denominadas multiníveis se apresentam como uma solução apropriada para diversas aplicações, devido algumas vantagens que estas apresentam em relação às topologias de dois níveis. Com um maior número de níveis de tensão na saída é possível obter um sinal com uma forma mais próxima de uma senóide. Quanto maior for esta aproximação, ou seja, quanto maior o número de níveis, menor é a derivada de tensão em relação ao tempo, sobre a carga. Isto proporciona a redução do conteúdo harmônico na tensão de saída, culminando em um sinal de melhor qualidade, o que acarreta em menores perdas na carga.

O aumento no número de níveis implica em um aumento do número de chaves, que leva a uma estratégia de modulação mais complexa. Estas estratégias de maior complexidade junto com os controladores, tem sido implementada, cada vez mais com sistemas digitais, ou seja, com o uso de microcontroladores ou microprocessadores. Estes dispositivos oferecem algumas vantagens em relação aos dispositivos analógicos, como flexibilidade, já que qualquer reconfiguração pode ser realizada via *software* sem a necessidade de se alterar o *hardware*, maior confiabilidade, pois os circuitos analógicos são mais sensíveis a variações climáticas por exemplo, e redução do volume dos circuitos. Vale ressaltar, que as soluções comerciais analógicas para controle de conversores, geralmente são para as topologias ditas clássicas.

O dispositivo responsável pelo controle dos conversores, comumente utilizado na maioria dos conversores que são controlados de forma digital, é o *Digital Signal Processor* (DSP) [1]. O primeiro DSP foi lançado pela empresa Bell Labs em 1979. Porém só ficou conhecido em 1983 quando a Texas Instruments lançou o modelo TMS32010. Ele é uma alternativa interessante devido a sua alta velocidade de processamento, versatilidade quanto à linguagem de programação (C ou *Assembly*) e por sua capacidade de operação aritmética complexa com ponto flutuante [1]. Atualmente os DSPs também apresentam periféricos como: módulos PWM (*Pulse Width Modulation*), temporizadores (*timers*) de uso geral e módulos de interrupção de eventos, o que o torna bem atraente para aplicações em eletrônica de potência. Estes periféricos conferem ao DSP a possibilidade de alguma operação concorrente, já que a sua principal limitação é a operação sequencial, ou seja, as instruções são executadas uma após a outra.

A fim de superar essa dificuldade da operação sequencial, o uso do FPGA (*Field Programmable Gate Array*) para o controle de conversores passou a ser cogitado, principalmente para as topologias multiníveis com mais de uma fase, já que ele é superior ao DSP no que refere-se a operações concorrentes.

O FPGA é formado por uma matriz de blocos lógicos contidos em um único circuito integrado. Suas principais qualidades são: possibilidade de operações concorrentes, podendo ser realizadas de forma dependente ou independente uma das outras e operações com *clocks* elevados, na ordem de 50MHz. Esta tecnologia permite que algoritmos como os de controladores digitais, que necessitam de alta velocidade na realização dos cálculos, sejam executados de forma paralela a outras operações, sem perdas de rendimento [1]. Os FPGAs mais recentes possuem sistemas de frequência de *clock* de mais de 500MHz. Os principais fabricantes de FPGA atualmente são a Altera, a Xilinx e a Actel.

Os objetivos deste trabalho são:

- efetuar a comunicação do conversor A/D do kit de desenvolvimento DE0 – Nano da fabricante Altera, com o FPGA EP4CE22F17C6N da família Cyclone IV também da Altera, presente no kit;
- realizar o controle de tensão, através de um controlador PID digital, de um conversor NPCm (*Modified Neutral Point Clamped*) de três níveis monofásico e avaliar a capacidade do FPGA utilizado em atender as necessidades deste projeto;
- realizar o controle de tensão, através de um controlador PID digital, de um conversor NPCm de três níveis trifásico e avaliar a capacidade do FPGA utilizado em atender as necessidades deste projeto;
- Avaliar a capacidade do FPGA utilizado em atender a demanda por elementos lógicos de um sistema de controle adaptativo para controlar a tensão de um conversor NPCm de três níveis monofásico.
- Verificar o tempo de resposta entre a solicitação de amostragem e a atualização do sinal de controle, para o caso do controlador PID digital.

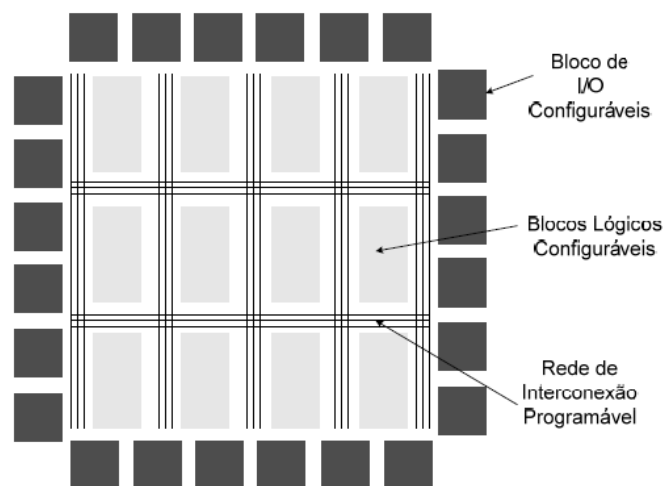
2 HARDWARE DE DISPOSITIVOS FPGA

Apresentar-se-á nesta seção, os elementos básicos inerentes a arquitetura de um FPGA, bem como o kit de desenvolvimento para FPGA utilizado neste trabalho.

2.1 ARQUITETURA DO FPGA

Os FPGAs foram desenvolvidos pela empresa Xilinx Inc. e lançados em meados da década de 1980, com a série XC2000 [2]. É um circuito integrado cuja estrutura contém uma matriz de blocos lógicos configuráveis (CLBs – do inglês, *Configurable Logic Blocks*). Estes CLBs são conectados por uma rede que é inteiramente reprogramável. Cada célula contém capacidade computacional para implementar funções lógicas e realizar roteamento para comunicação entre elas. Outro componente importante da estrutura é o bloco configurável de entrada/saída (IOB - do inglês, *Input/Output Block*). A matriz de CLBs é circundada pelos IOBs, assim o projetista pode configurar o FPGA de forma que o *hardware* inferido atenda a aplicação desejada. Esta configuração é realizada através de um *software* de desenvolvimento [1]. Na Figura 1 é exibida a arquitetura característica de um FPGA.

Figura 1 - Arquitetura típica de um FPGA



Fonte: [1]

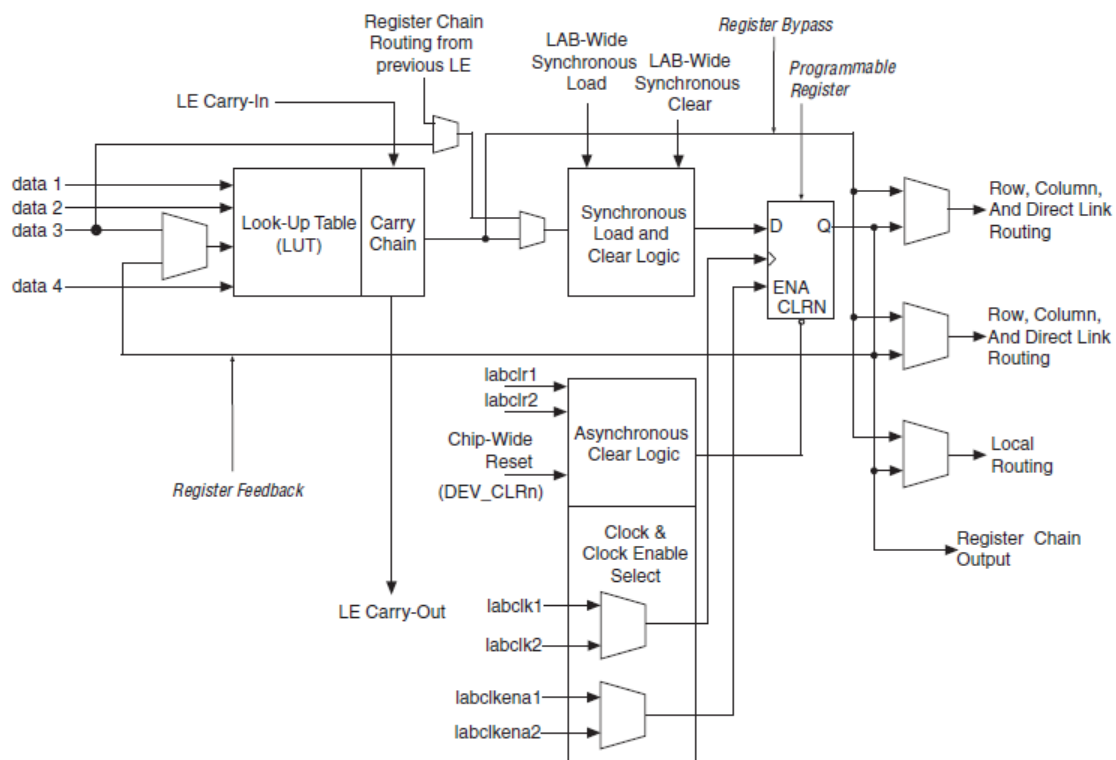
Os blocos lógicos são controlados pelas células de memória que também controlam as conexões que os blocos poderão fazer em cada aplicação. As tecnologias de memória utilizadas são: *Flash*, EPROM, SRAM, SDRAM. Ou seja, aquelas que podem ser

reprogramadas. Os FPGAs mais modernos possuem também em sua arquitetura outros blocos dedicados como blocos RAM, blocos DSP e circuitos PLL (*Phase Locked Loop*), que são úteis em aplicações de maior complexidade [2]. Outro avanço na tecnologia de FPGAs é a inclusão de um conversor A/D em alguns modelos, facilitando a sua aplicação em sistemas de controle.

2.1.1 Blocos lógicos configuráveis

Também chamados de *logic elements* (LEs), são estruturas que incluem mais de duas células lógicas [3]. O FPGA EP4CE22F17C6N utilizado neste trabalho possui 22320 LEs [3]. As LEs são as menores unidades lógicas presentes nos dispositivos da família Cyclone IV [3]. A estrutura de uma LE é apresentada na Figura 2 [3].

Figura 2 – Estrutura de uma LE do dispositivo Cyclone IV



Fonte: [3]

Esta estrutura possui uma *lookup table* (LUT) de 4 bits que pode ser configurada tanto como uma ROM (16 x 1), uma RAM ou uma função combinacional [1]. O trajeto de *carry* é incluindo para certificar-se da eficiência das operações aritméticas [1]. O *flip-flop* do tipo D presente na saída permite o registro da saída do elemento lógico. O modo *feedback* permite

que a saída do registrador alimente a LUT do mesmo LE para garantir que o registro seja empacotado com o próprio *fan out* da LUT fornecendo um outro mecanismo para um ajuste mais refinado [3].

2.1.2 Blocos de memória

Cada bloco de memória chamada de M9K do dispositivo Cyclone IV fornece 9 *kbits* de memória SRAM embutida. É possível configurar os blocos de memória M9K como porta única, uma simples porta dupla ou uma verdadeira RAM de porta dupla, bem como buffers FIFO ou ROM [3]. Estes blocos também podem ser configurados para implementar qualquer um dos comprimentos de dados da Tabela 1.

Tabela 1 – Largura de dados do bloco M9K

Mode	Data Width Configurations
Single port or simple dual port	×1, ×2, ×4, ×8/9, ×16/18, and ×32/36
True dual port	×1, ×2, ×4, ×8/9, and ×16/18

Fonte: [3]

2.1.3 Blocos multiplicadores

Os blocos multiplicadores embutidos podem implementar um multiplicador 18×18 ou dois 9×9 em um único bloco [3]. Para multiplicações maiores que 18×18 o *software* de desenvolvimento Quartus II[®], utilizado neste trabalho, cascadeia blocos multiplicadores. Não há restrições de largura dos dados inseridos nos multiplicadores, porém quanto maior for a largura de dados, mais lento é o processo de multiplicação [3]. A Tabela 2 lista o número de multiplicadores embutidos em alguns dispositivos e também o modo de multiplicador que pode ser implementado.

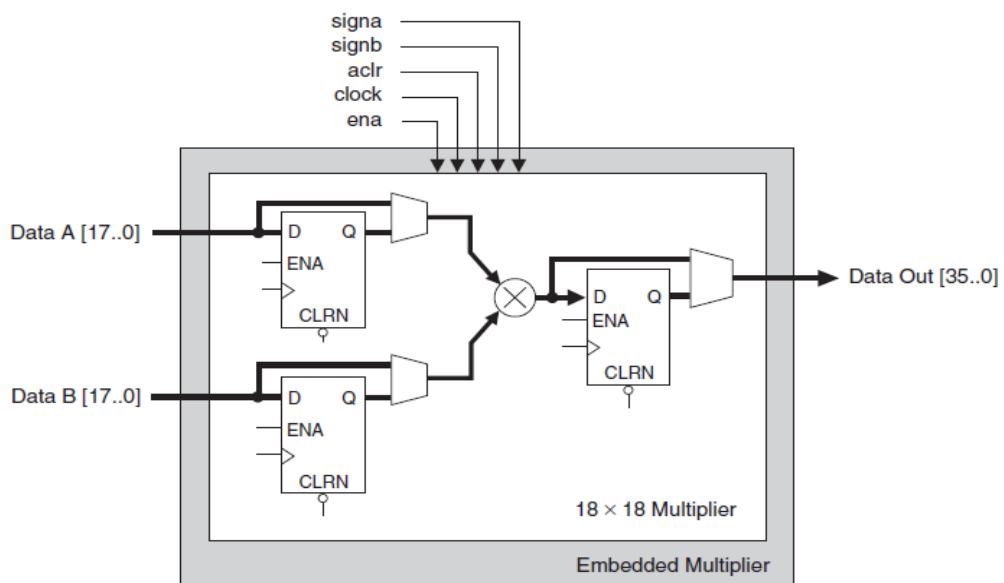
Tabela 2 – Número de multiplicadores embutidos no dispositivo Cyclone IV

Device Family	Device	Embedded Multipliers	9 × 9 Multipliers ⁽¹⁾	18 × 18 Multipliers ⁽¹⁾
Cyclone IV GX	EP4CGX15	0	0	0
	EP4CGX22	40	80	40
	EP4CGX30	80	160	80
	EP4CGX50	140	280	140
	EP4CGX75	198	396	198
	EP4CGX110	280	560	280
	EP4CGX150	360	720	360
Cyclone IV E	EP4CE6	15	30	15
	EP4CE10	23	46	23
	EP4CE15	56	112	56
	EP4CE22	66	132	66
	EP4CE30	66	132	66
	EP4CE40	116	232	116
	EP4CE55	154	308	154
	EP4CE75	200	400	200
	EP4CE115	266	532	266

Fonte: [3]

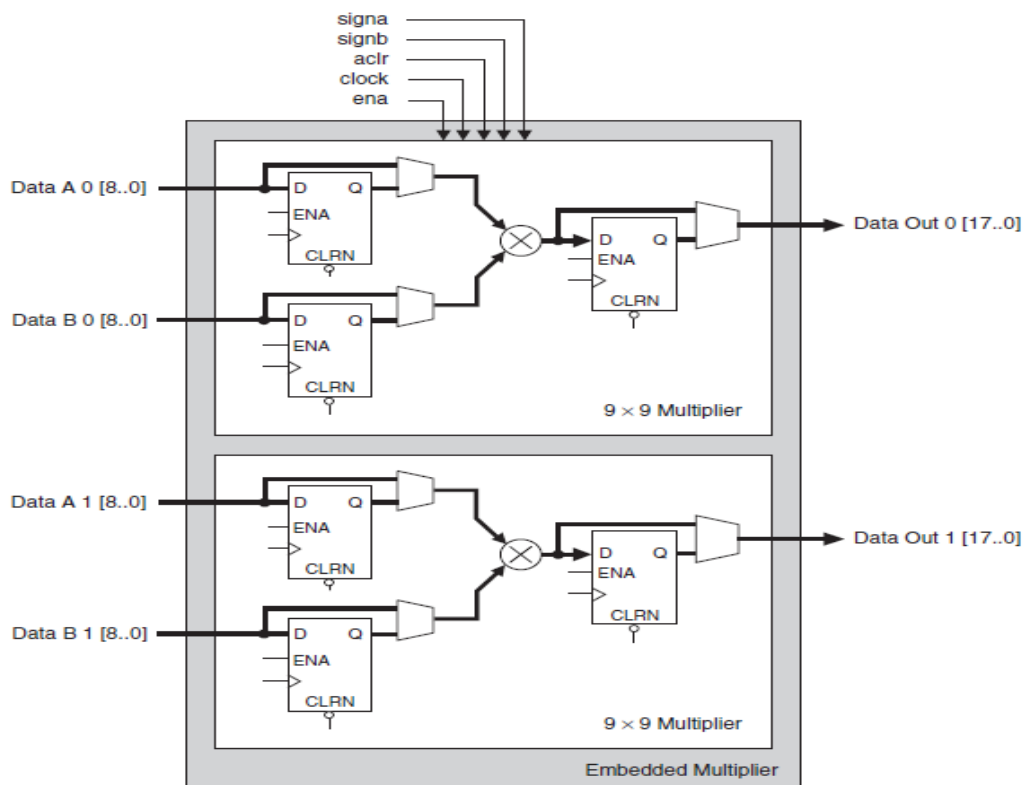
Nas Figuras 3 e 4, são apresentados a arquitetura do bloco multiplicador e os modos de multiplicador 18×18 e 9×9 respectivamente.

Figura 3 – Arquitetura de um bloco multiplicador no modo (18×18)



Fonte: [3]

Figura 4 – Arquitetura de um bloco multiplicador no modo (9×9)



Fonte: [3]

2.1.4 Blocos DSP

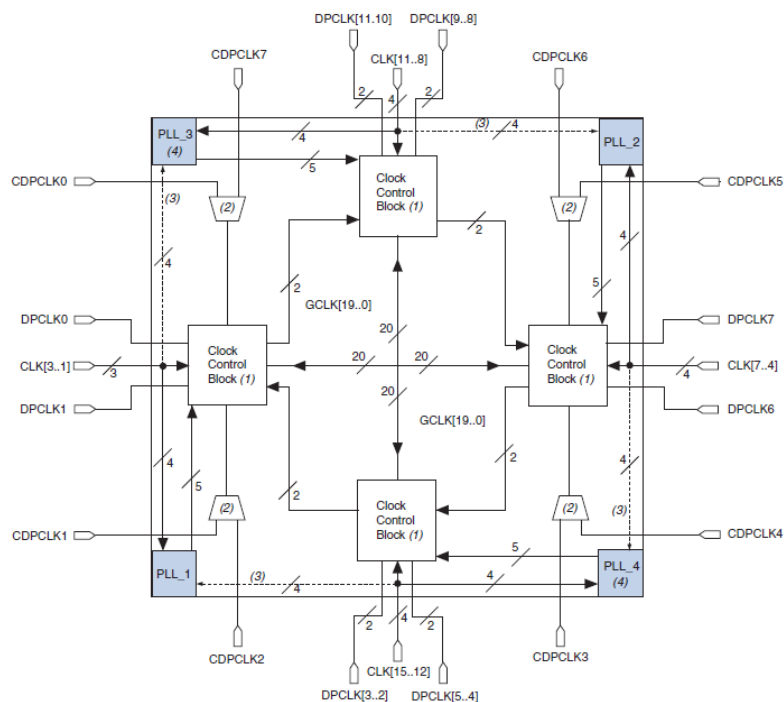
O processamento de sinais digitais tal como (filtragem FIR/IIR, FFT etc.) é executado por três elementos básicos: multiplicadores, acumuladores e registradores. Para simplificar a implementação desse tipo de aplicação e também torná-la mais rápida, foram incluídos nos FPGAs os blocos DSP, os quais consistem essencialmente de multiplicadores paralelos, circuitos MAC (*Multiply and Accumulate*) e registradores de deslocamento [1].

2.1.5 Gerenciamento de clock

O gerenciamento de *clock* conta com duas partes principais: distribuição de *clock* e manipulação de *clock*. Uma rede de distribuição de *clock* adequada é necessária para diminuir um fenômeno chamado de *clock skew* (diferença de tempo com que o sinal de *clock* chega às várias partes do circuito integrado) [1][3]. Esse tipo de rede é construída reduzindo-se tanto quanto possível as resistências e capacitâncias parasitas de suas linhas, porém sem eliminar a possibilidade de que em alguns pontos da rede seja necessário introduzir retardos intencionais

para compensar as grandes diferenças de distâncias entre as várias regiões do circuito integrado [1]. Um exemplo de rede de distribuição de *clock* pode ser observado na Figura 5.

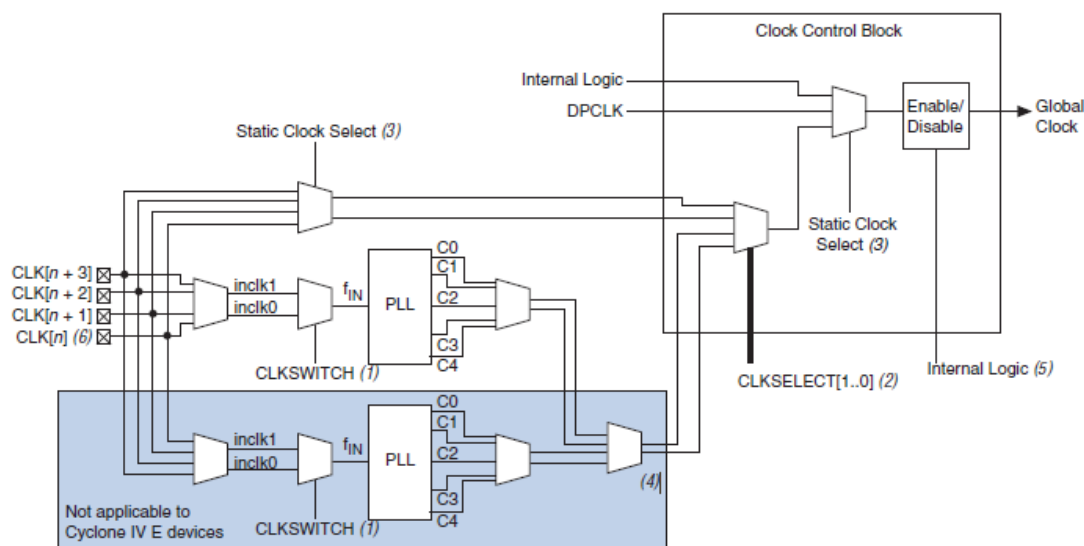
Figura 5 – Rede de *clock*



Fonte: [3]

A Figura 6 apresenta um bloco de controle de *clock*.

Figura 6 – Bloco de controle de *clock*



Fonte: [3]

A manipulação de *clock* é a outra parte fundamental do gerenciamento de *clock*. Para fazer isso, normalmente são empregados PLLs, que cumprem as quatro finalidades principais seguintes: multiplicação de *clock*, divisão de *clock*, deslocamentos de fase e filtragem de *jitter* [3]. O FPGA utilizado possui 4 PLLs.

2.2 KIT DE DESENVOLVIMENTO DE0-NANO DA ALTERA

Neste trabalho é utilizado o kit DE0-Nano, que é uma plataforma de desenvolvimento de FPGA da Altera . A placa foi projetada para a implementação de dispositivos da família Cyclone IV. A escolha deste kit está relacionada com a sua disponibilidade no laboratório. O kit possui um FPGA modelo EP4CE22F17C6N da família Cyclone IV [7]. Este FPGA atendeu a necessidade do projeto referente ao controle PID digital do conversor NPCM de três níveis monofásico e trifásico que serão apresentados nas seções 6.3 e 6.4, porém não teve a capacidade de atender a necessidade do projeto do *Indirect Self Tuning* apresentado em na seção 6.5.

A Tabela 3 apresenta as principais características do kit de desenvolvimento DE0 – Nano e a Figura 7 exibe a imagem do kit.

Tabela 3 - Características do kit de desenvolvimento DE0 – Nano

FPGA Cyclone® IV EP4CE22F17C6N • 22,320 Elementos Lógicos (LEs) • 4 PLLs • 66 Blocos Multiplicadores 18x18 • 153 Portas
Conversor A/D de 8 canais e 12 bits
Dispositivos de Memória • 32MB SDRAM • 2Kb I²C EEPROM
8 LEDs Verdes
4 Micro Chaves
2 Botões

2 Botões
Clock de 50 MHz
Acelerômetro (13-bit)
Conexão USB
2 barramentos de 40 pinos cada para conexão às I/O do circuito integrado

Fonte: produção do próprio autor

Figura 7 – Kit DE0 – Nano da fabricante Altera



Fonte: [6]

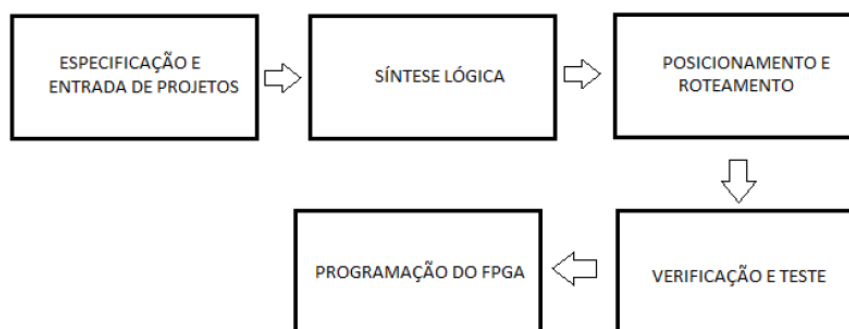
3 PROGRAMAÇÃO DE DISPOSITIVOS FPGA

Esta seção tem por objetivo, informar de forma sucinta o leitor, sobre os aspectos relacionados a um projeto utilizando FPGA, VHDL (*Very High Speed Integrated Circuit Hardware Description Language*) e VHDL 2008.

3.1 ETAPAS DE UM PROJETO UTILIZANDO FPGA

Na Figura 8 são apresentadas as etapas de projeto utilizando um FPGA. Uma descrição sobre cada etapa é apresentada em [6].

Figura 8 - Sequência de etapas do processo de um projeto com FPGA



Fonte: [6]

3.2 PROGRAMAÇÃO NO FPGA

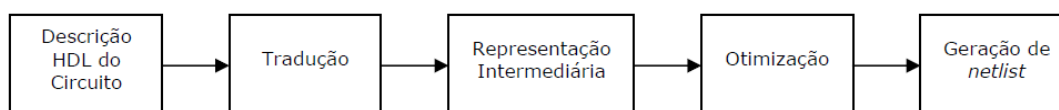
Uma alternativa à entrada esquemática de um circuito digital em um sistema de projeto auxiliado por computador é utilizar a técnica de projeto de dispositivos lógicos programáveis (PLDs) com uma ferramenta de projeto baseado em texto ou linguagem de descrição de *hardware* (HDL). Exemplos de HDLs são o AHDL (*Altera Hardware Description Language*) e os padrões VHDL e Verilog [2].

O projetista cria um arquivo de texto, seguindo certo conjunto de regras, conhecido como sintaxe da linguagem, e usa um compilador para criar dados de programação do dispositivo lógico programável. Esta descrição de *hardware* pode ser usada para gerar projetos hierárquicos, ou seja, um componente definido em uma descrição pode ser usado para gerar um *hardware* específico ou ser usado como parte de outro projeto.

Uma grande razão para o uso de HDLs é a síntese lógica. Uma descrição em HDL em conjunto com uma biblioteca de componentes é usada por uma ferramenta de síntese para a

geração automática de um circuito digital. Além disto, estas ferramentas incluem uma etapa de otimização da lógica interna do circuito gerado, antes da geração das estruturas internas de armazenamento, da lógica combinatória e da estrutura de conexão dos componentes (*netlist*) [2]. A Figura 9 abaixo apresenta um diagrama mostrando as etapas principais de síntese lógica.

Figura 9 – Etapas principais de síntese lógica



Fonte: [2]

Atualmente, as HDLs mais utilizadas são o VHDL e o Verilog. Ambas as linguagens são hoje padrões aprovados e publicados pelo IEEE (*Institute of Electrical and Electronic Engineers*), tendo portanto, várias ferramentas comerciais disponíveis. Esta padronização leva a uma grande vantagem no desenvolvimento de circuitos usando HDLs: a portabilidade. Visto que as ferramentas devem implementar as características padronizadas, fica muito fácil usar sistemas de desenvolvimento de fabricantes diferentes.

Um sistema de controle em FPGA pode ser programado de três maneiras: utilizando um ambiente de *software* EDA (*Electronic Design Automation*), utilizando programas que usam uma linguagem estruturada de descrição de *hardware*, como AHDL, VHDL ou *Verilog*, ou aplicando esses dois conceitos em uma só programação. O ambiente de projetos *Quartus II Web Edition*, desenvolvido pela Altera [4] é o *software* EDA utilizado neste trabalho. O *software* fornece ao projetista a condição para programar o FPGA, das maneiras citadas nos seguintes modos:

- editor gráfico: o projetista pode utilizar de um diagrama lógico, muito semelhante a um diagrama de blocos, desenvolvido a partir de elementos primitivos e disponibilizados pela Altera como: portas lógicas, contadores, comparadores, somadores, multiplicadores, etc.
- editor de texto: o projetista pode programar alguma função específica utilizando-se de linguagem de descrição de *hardware*.
- editor de símbolo gráfico: nesse caso, o projetista pode criar um bloco para conexão no editor gráfico a partir da programação realizada no editor de texto.

Neste trabalho será adotada a linguagem VHDL. O nome VHDL é uma sigla que significa *VHSIC Hardware Description Language*. Já o termo VHSIC é o acrônimo de *Very*

High Speed Integrated Circuit [1][2][5]. Assim podemos traduzir livremente o nome VHDL como "linguagem de descrição de *hardware* para circuitos integrados de velocidade muito alta".

3.2.1 Síntese do VHDL

A linguagem VHDL foi originalmente desenvolvida por empresas contratadas pelo governo americano e agora é um padrão requerido por todos os ASICs (*Application Specific Integrated Circuits*). Ela foi padronizada pelo IEEE em 1987 (Padrão 1076-1987 ou VHDL 87) e foi atualizada em 1993 (Padrão 1076-1993 ou VHDL 93). Os trabalhos do IEEE continuam e uma nova revisão está em desenvolvimento (VHDL 200x). Em 1999, foi lançado o padrão IEEE 1076.1, conhecido como VHDL-AMS, que oferece extensões para circuitos com sinais analógicos e digitais [1].

Todo arquivo VHDL requer ao menos duas estruturas: uma declaração de entidade (*entity*) e uma arquitetura (*architecture*). A declaração de entidade define os aspectos externos da função VHDL, isto é, os nomes das entradas e saídas e o nome da função. A arquitetura define os aspectos internos, isto é, como as entradas e saídas influem no funcionamento e como se relacionam com outros sinais internos.

3.2.2 VHDL 2008

No ano de 2008, a organização de normas *Accellera Systems Initiative* (associado ao *Institute of Electrical and Electronic Engineers* - IEEE) e a *EDA Industry Working Groups*, definiram o VHDL 4.0, que também é conhecido por VHDL 2008 [8]. O principal objetivo do VHDL 2008 é melhorar algumas rotinas antigas e estabelecer padrões para operações aritméticas de ponto fixo e de ponto flutuante conforme a norma IEEE 754 [9]. Esta versão conta com um pacote que implementa operações matemáticas em ponto fixo e ponto flutuante para operações em VHDL [9]. Estes novos recursos matemáticos do VHDL 2008 possibilitaram o projeto de controladores discretos mais precisos que os controladores construídos com representação por números inteiros [8]. Também, criou-se a possibilidade de desenvolver um projeto sob dois tipos de abordagem: utilizando ponto fixo ou utilizando ponto flutuante.

De acordo com profissionais que atuam na área, as operações em ponto flutuante no VHDL 2008 ainda são pouco estáveis. Vale ressaltar que estas opiniões não são resultados de

um trabalho científico que tenha sido submetido à uma avaliação mais criteriosa. Entretanto, elas foram consideradas na escolha da abordagem utilizada neste trabalho. Outro fator considerado, foi a demanda por recursos do FPGA. Como a implementação em ponto fixo demanda menos recurso, esta abordagem foi escolhida neste trabalho.

A Tabela 4 apresenta de forma resumida os tipos de dados mais usados em FPGA. As linhas com fundo em cinza apresentam os tipos de dados do VHDL 2008, o qual ainda está em processo de regulamentação.

Tabela 4 - Resumos dos pacotes e dados usados em VHDL

Dimensão	Tipo predefinido	Biblioteca ou pacote de origem	Valores sintetizáveis sem restrições
Escalar	boolean	Std/standar	True, false
	bit	Sdt/standar	'0', '1'
	Std_(u)logic	IEEE/std_logic_1164	Entrada: '0', '1', 'L', 'H' Saída: '0', '1', 'L', 'H', 'X', 'W', 'Z'
1D	Bit_vector	Std/standar	Idem bit
	Std_(u)logic_vector	IEEE/std_logic_1164	Idem std_(u)logic
	Unsigned, signed	IEEE/numeric_std ou IEEE/std_logic_arith	Idem std_(u)logic
1D x 1D	integer	Sdt/standar	-(231-1) a (231-1) padrão
	natural	Sdt/standar	0 a (231-1) padrão
	positive	Sdt/standar	1 a (231-1) padrão
	character	Sdt/standar	Código ASCII estendido
1D x 1D	string	Sdt/standar	Conjunto de caracteres
1D	Boolean_vector	Sdt/standar	Idem boolean
	Ufixed, sfixed (propostos)	IEEE/fixed_pkd + pacotes associados	Valores numéricos de ponto fixo
	Float (propostos)	IEEE/float_pkd + pacotes associados	Valores numéricos de ponto flutuante (IEEE 754)
1D x 1D	Integer_vector	Sdt/standar	Idem integer

Fonte: [8]

Na Tabela 5 estão os tipos de operadores e operações associadas.

Tabela 5 - Operadores básicos e tipos predefinidos em VHDL

Tipos de operadores	Operadores predefinidos	Tipos de dados sintetizáveis
Lógicos	Not, and, nand, or, nor, xor, xnor	Boolean, bit, bit_vector, std_(u)logic, std_(u)logic_vector, (un)signed
Aritméticos	+, -, *, /, **, abs, REM, mod	Integer, natural, positive, (un)signed, std_(u)logic_vector
Comparação	=, /=, >, <, >=, <=	Boolean, bit, bit_vector, integer, natural, positive, (un)signed, character, string, std_(u)logic_vector
Shift	sll, srl, sla, sra, rol, ror	Bit_vector, std_(u)logic_vector, (un)signed
Concatenação	& (também “,” e others)	Bit_vector, std_(u)logic_vector, string, (un)signed

Fonte: [8]

O VHDL 2008 esta sendo especificado como IEEE 1076-2008. Mais informações podem ser encontradas no *Standard VHDL Language Reference manual*. Porém, neste trabalho serão utilizados conceitos apresentados por [2] e [9].

O pacote desenvolvido por Bishop deve ser usado com o VHDL-93 (1993). Para o funcionamento é necessário o uso dos seguintes pacotes:

- Fixed_float_types.vhdl;
- Fixed_generic_pkg.vhdl;
- Fixed_generic_pkg-body.vhdl;
- Fixed_pkg.vhdl.

Disponíveis em www.vhdl.org/fphdl/

É muito provável que estes pacotes, desenvolvidos para serem utilizados no VHDL 2008, serão incorporados na nova versão das bibliotecas do IEEE [8]. Também é necessário, além dos pacotes citados acima, a utilização de mais dois pacotes da versão VHDL 93, a fim de proporcionar compatibilidade [8]. Segue abaixo a identificação deles:

- Fixed_float_types_c.vhdl: trata dos tipos de números com ponto fixo usados no pacote;
- Fixed_pkg_c.vhdl: versão de compatibilidade para o VHDL 93.

Ao utilizar estas bibliotecas é necessário definir previamente o domínio, ou seja, se os números serão com sinal (sfixed) ou sem sinal (ufixed) [9]. Operações matemáticas só são realizadas entre números que pertençam ao mesmo domínio. Estas bibliotecas possibilitam o uso de funções para a conversão do tipo de representação.

Exemplo:

```
x <= to_sfixed (-4.125, 6, -6); -- transforma o número "-4.125" em um número com
                                -- sinal, com 7 bits antes e 6 bits depois da vírgula.
y <= to_ufixed (4.125, 5, -6); -- transforma o número "4.125" em um número sem
                                -- sinal, com 6 bits antes e 6 bits depois da vírgula.
```

O tamanho final dos campos, antes da vírgula e depois da vírgula, dos resultados de operações matemáticas, dependem do tipo de operação efetuada e da representação escolhida. A Tabela 6 apresenta o cálculo dos campos para diferentes operações com e sem sinal.

Tabela 6 – Apoio para definição do tamanho dos campos após operações matemáticas

Operation	Result Range
$A + B$	$\text{Max}(A'_{\text{left}}, B'_{\text{left}})+1$ downto $\text{Min}(A'_{\text{right}}, B'_{\text{right}})$
$A - B$	$\text{Max}(A'_{\text{left}}, B'_{\text{left}})+1$ downto $\text{Min}(A'_{\text{right}}, B'_{\text{right}})$
$A * B$	$A'_{\text{left}} + B'_{\text{left}}+1$ downto $A'_{\text{right}} + B'_{\text{right}}$
$A \text{ rem } B$	$\text{Min}(A'_{\text{left}}, B'_{\text{left}})$ downto $\text{Min}(A'_{\text{right}}, B'_{\text{right}})$
Signed /	$A'_{\text{left}} - B'_{\text{right}}+1$ downto $A'_{\text{right}} - B'_{\text{left}}$
Signed $A \bmod B$	$\text{Min}(A'_{\text{left}}, B'_{\text{left}})$ downto $\text{Min}(A'_{\text{right}}, B'_{\text{right}})$
Signed Reciprocal(A)	$-A'_{\text{right}}$ downto $-A'_{\text{left}}-1$
$\text{Abs}(A)$	$A'_{\text{left}}+1$ downto A'_{right}
$-A$	$A'_{\text{left}}+1$ downto A'_{right}
Unsigned /	$A'_{\text{left}} - B'_{\text{right}}$ downto $A'_{\text{right}} - B'_{\text{left}} - 1$
Unsigned $A \bmod B$	B'_{left} downto $\text{Min}(A'_{\text{right}}, B'_{\text{right}})$
Unsigned Reciprocal(A)	$-A'_{\text{right}}+1$ downto $-A'_{\text{left}}$

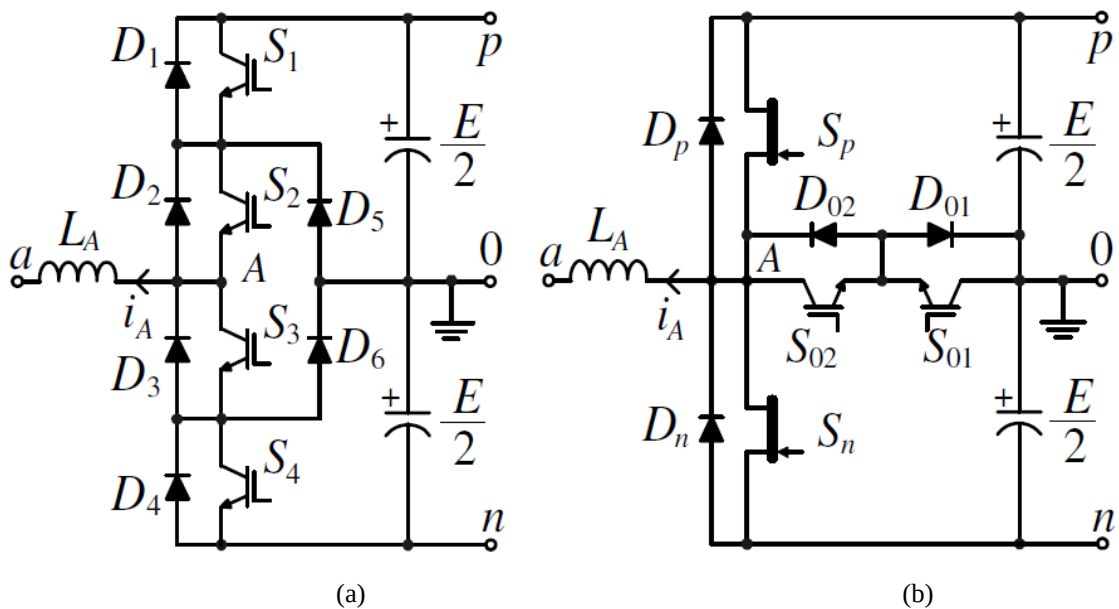
Fonte: [9]

Existem alguns comandos criados para facilitar a determinação dos tamanhos dos campos, inibindo a necessidade de cálculos repetitivos para a definição de tais tamanhos [9]. São eles: *resize*, *ufixed_high*, *ufixed_low*, *sfixed_high*, *sfixed_low*. Neste trabalho utilizou-se o comando *resize*. O comando *resize*, por padrão, proporciona um arredondamento ao invés de um truncamento, quando necessário, porém isto pode ser alterado [9]. Este comando também possui um parâmetro que define o tipo de retorno em caso de *overflow*. Por padrão, o retorno é o valor máximo que pode ser representado pelo tamanho dos campos pré definidos [9].

4 CONVERSOR NPC_m DE TRÊS NÍVEIS

A estrutura básica do conversor utilizado neste trabalho é apresentado na Figura 10(b). Trata-se de um tipo diferente de NPC (*Neutral Point Clamped*) três níveis [12]. Esta topologia foi nomeada como NPC modificado, podendo ser representado pela sigla NPC_m. O conversor de três níveis apresenta uma diminuição da derivada de tensão e redução de conteúdo harmônico na tensão de saída, em relação aos conversores de dois níveis, permitindo a redução ou eliminação de filtros passivos. O conversor NPC_m possui uma topologia bidirecional, podendo atuar como inversor ou retificador.

Figura 10 – (a) Estrutura do conversor NPC; (b) Estrutura do conversor NPC_m



Fonte – [12]

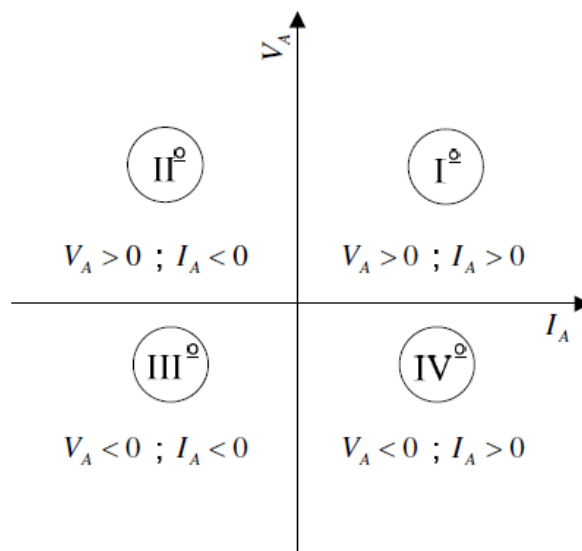
Comparando as duas estruturas (NPC – Figura 10(a) e NPC_m – Figura 10(b)) pode-se observar que a estrutura NPC possui dois diodos adicionais, D_5 e D_6 , usados para a função de grampeamento. Mas a diferença fundamental entre as duas topologias é que no conversor NPC_m os semicondutores S_p , S_n , D_p e D_n estão sujeitos à tensão total do barramento CC, enquanto que todos os outros dispositivos, das duas estruturas, bloqueiam a metade deste valor de tensão [12]. Vale ressaltar também, que devido ao potencial comum dos emissores de S_{01} e S_{02} , o conversor NPC_m necessita de somente três fontes isoladas para a alimentação dos *drivers*, enquanto que no NPC é preciso quatro fontes isoladas [12]. Um fato importante, é que mesmo com interruptores operando com diferentes níveis de bloqueio, das diferentes

topologias e diferentes quantidades de componentes, os dois sistemas apresentam as mesmas características externas. Logo se for utilizada a mesma estratégia de modulação nas duas estruturas, as tensões e correntes através de todos os seus terminais (A , p , n , 0) serão as mesmas [12].

4.1 QUADRANTES DE OPERAÇÃO DO CONVERSOR NPC_m

O conversor NPC_m de três níveis pode operar em quatro quadrantes do plano $V_A \times I_A$ apresentado na Figura 11. Neste plano, as correntes que saem pelo terminal a da Figura 10, são consideradas positivas.

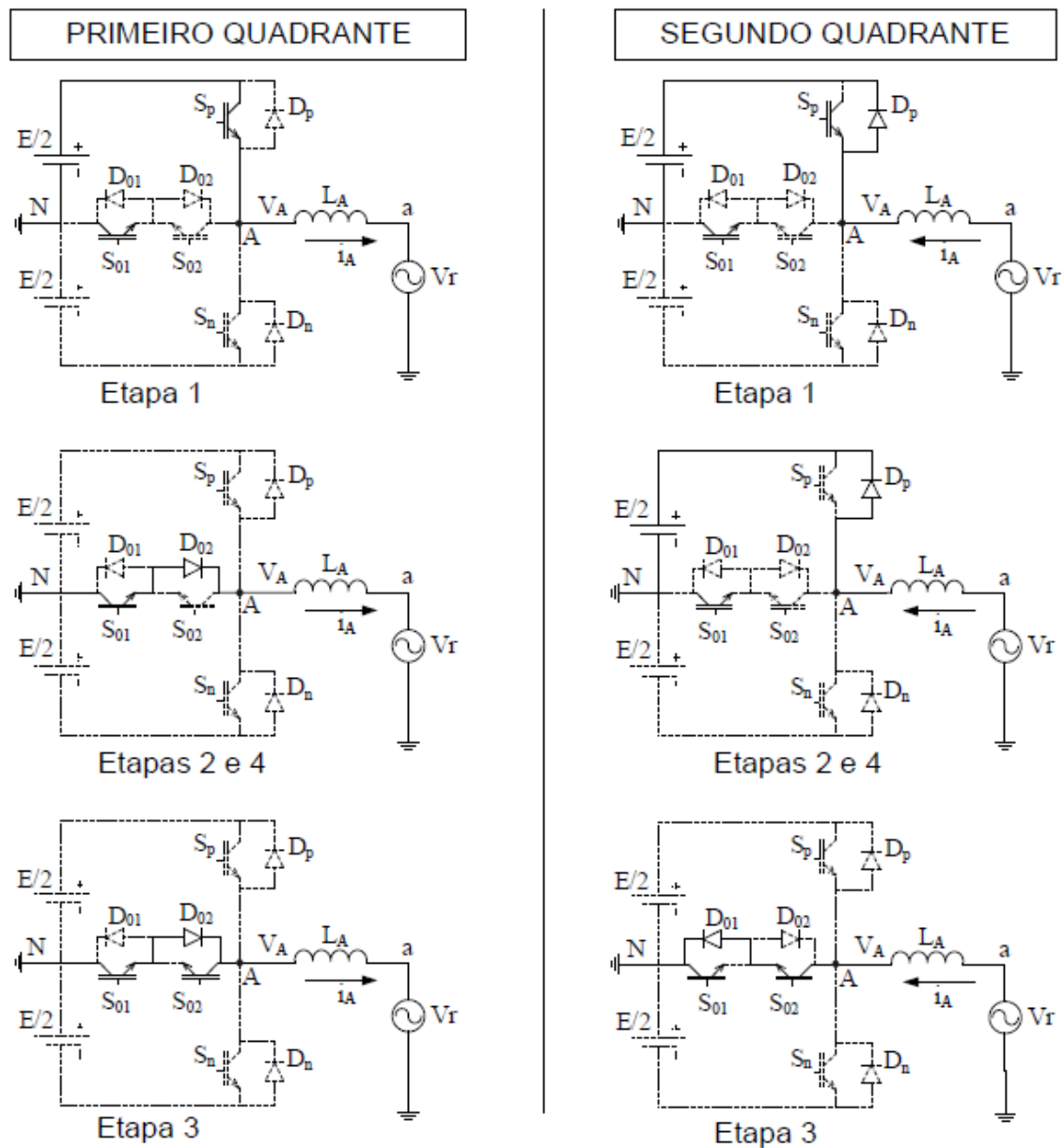
Figura 11 - Plano $V_A \times I_A$



Fonte: [12]

O conversor NPC_m atua como inversor quando opera no 1º e 3º quadrantes e pode atuar como retificador quando opera no 2º e 4º quadrantes. A Figura 12 apresenta as etapas de operação para o 1º e 2º quadrantes, em que a tensão média de saída (V_A) é positiva.

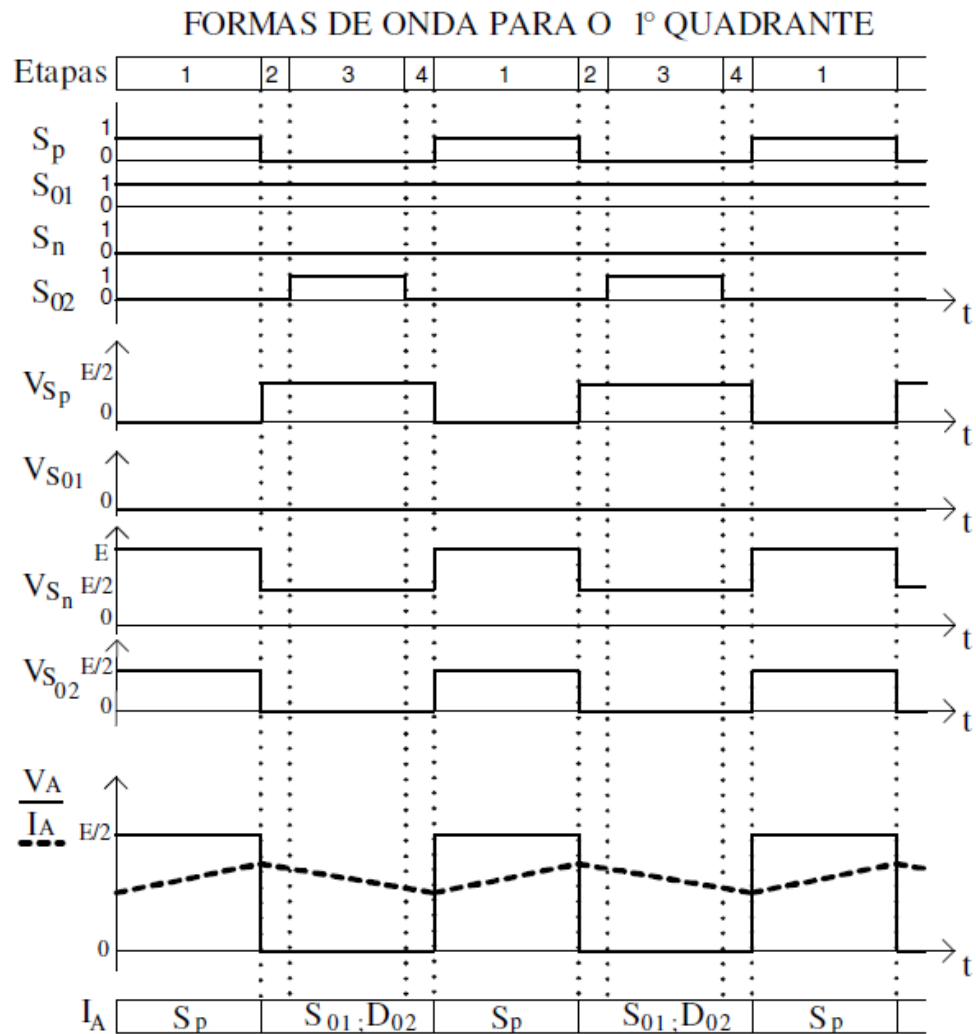
Figura 12 - Etapas de operação para o 1º e o 2º quadrantes



Fonte: [12]

A Figura 13 exibe, como exemplo, as formas de onda relativas à operação do conversor no primeiro quadrante. Através da figura pode-se observar a sequência de operação das chaves, incluindo o tempo morto, bem como a forma de onda da tensão e da corrente de saída.

Figura 13 – Formas de onda para o 1º quadrante

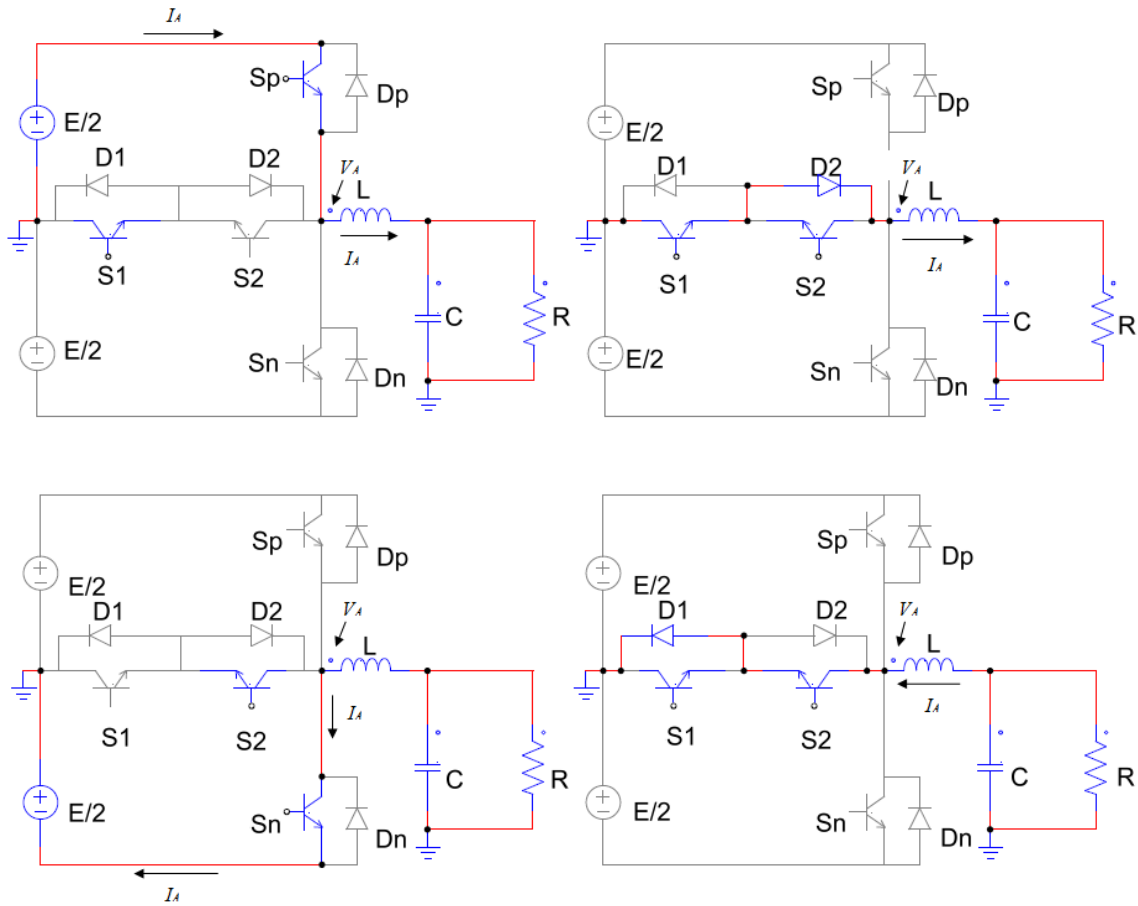


Fonte: [12]

4.2 NPCm OPERANDO COMO INVERSOR

Neste trabalho são considerados os quadrantes 1 e 3, em que o sistema opera como um inversor. É apresentado na Figura 14 as etapas de operação nestes quadrantes. As linhas em vermelho indicam o caminho percorrido pela corrente e os componentes em azul indicam estarem ativos. Nesta figura não está representado o tempo morto entre as comutações das chaves. Entretanto, nos códigos desenvolvidos em VHDL para a parte experimental o tempo morto é previsto.

Figura 14 – Etapas de operação do 1° e 3° quadrante



Fonte: produção do próprio autor

O valor da tensão de cada fonte que compõe o *Link DC* pode ser obtido a partir da Equação (1), onde m é o número de níveis e V_{cc} é a tensão total do barramento [6].

$$E/2 = \frac{V_{cc}}{m - 1} \quad (1)$$

Os níveis de tensão permitidos na saída, em cada etapa de operação apresentada na Figura 15, são $E/2$, 0 e $-E/2$. Os estados das chaves para se obter tais níveis são mostrados na Tabela 7.

Tabela 7 – Tensão de saída em função do estado das chaves do conversor NPCm

Tensão de Saída	S_p	S_n	S_{01}	S_{02}
$E/2$	1	0	1	0
0	0	0	1	1
$-E/2$	0	1	0	1

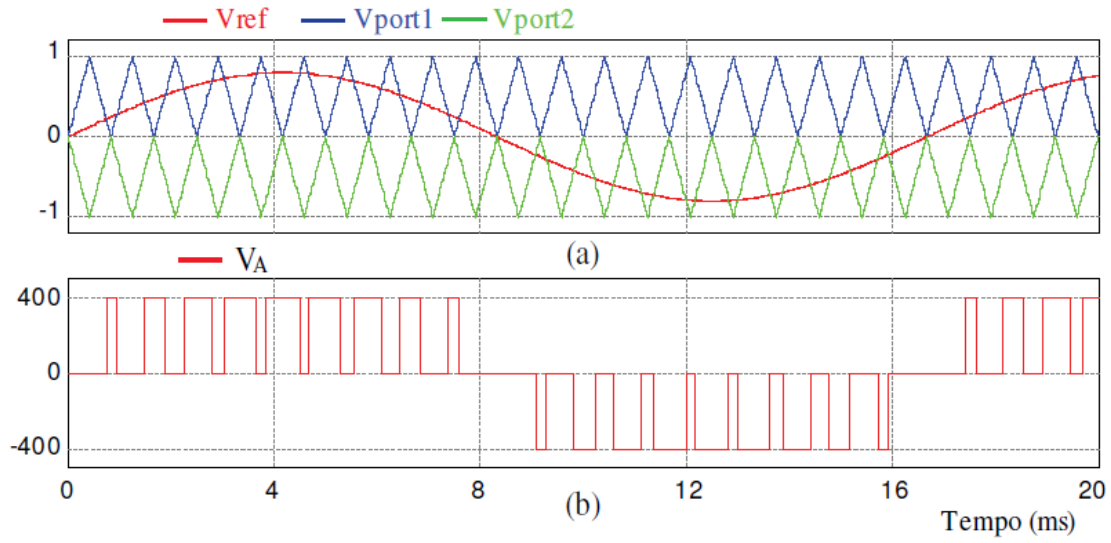
Fonte: produção do próprio autor

As chaves S_p e S_{02} atuam de forma complementar, bem como as chaves S_n e S_{01} . Enquanto S_p e S_{02} chaveiam de forma alternada, S_{01} permanece atuada. O mesmo ocorre com a chave S_{02} enquanto as chaves S_n e S_{01} chaveiam alternadamente. Os sinais que atuam estas chaves podem ser obtidos a partir de modulações PWM. Para conversores que utilizam modulação PWM, dois índices devem ser levados em consideração na análise do funcionamento dos conversores, o índice de modulação de amplitude e o índice de modulação de frequência [13][14]. Vale ressaltar que existem diversos métodos para se realizar uma modulação PWM.

4.3 ESCOLHA DA ESTRATÉGIA DE MODULAÇÃO

Os estudos realizados em [12] mostraram que a estratégia POD (*Phase Opposition Disposition*) apresentou resultados similares de DHT (Distorção Harmônica Total) e espectro harmônico da tensão de saída V_A , em comparação com outras estratégias. Assim, devido a facilidade de se implementar a estratégia de modulação PWM POD apresentada em [6], optou-se por manter o mesmo método de modulação neste trabalho. Na Figura 15 é exibido um exemplo da estratégia de modulação POD. Pode-se observar nesta figura a presença de uma moduladora senoidal e duas portadoras triangulares de mesma amplitude e frequência, deslocadas no eixo de tensão e defasadas 180° uma em relação a outra. Para conversores de três níveis são necessárias apenas duas portadoras.

Figura 15 - Modulação tipo POD: (a) moduladora senoidal e portadoras ; (b) tensão de saída VA.



Fonte: [12]

O índice de modulação de amplitude m_a é dado como a razão entre o valor de pico do sinal da moduladora A_m e o valor de pico da portadora A_p [6][12], como pode ser observado na Equação (2).

$$m_a = \frac{A_m}{A_p} \quad (2)$$

A partir da Equação (3) obtêm-se o índice de modulação de frequência, que relaciona a frequência de comutação das chaves ou da frequência da portadora com a frequência da moduladora [6][12].

$$m_f = \frac{f_{port}}{f_{mod}} \quad (3)$$

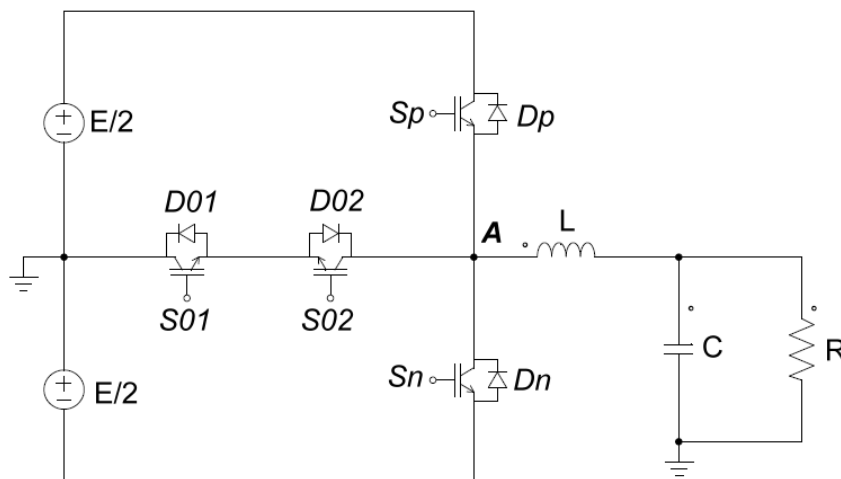
5 PROJETO DOS SISTEMAS DE CONTROLE E RESULTADOS DE SIMULAÇÃO

Apresentar-se-á nesta seção, o levantamento do modelo matemático $\Delta V/\Delta D$ do conversor NPCm, conectado a um filtro LC e a uma carga resistiva. Será abordado também, algumas definições básicas do controlador PID digital. Apresentar-se-á o projeto do controlador PID, simulações do modelo comutado e do modelo matemático, além de equações e definições relacionadas à técnica de controle adaptativo.

5.1 MODELAGEM DESTINADA AO CONTROLE DE TENSÃO DO CONVERSOR NPCm MONOFÁSICO COM FILTRO LC

A Figura 16 apresenta a estrutura monofásica do conversor NPCm que está conectada a um filtro LC. O resistor R, que representa a carga, será levado em consideração no levantamento do modelo matemático, de forma a estabelecer uma determinada região para o ponto de operação.

Figura 16 – Estrutura do conversor monofásico NPCm

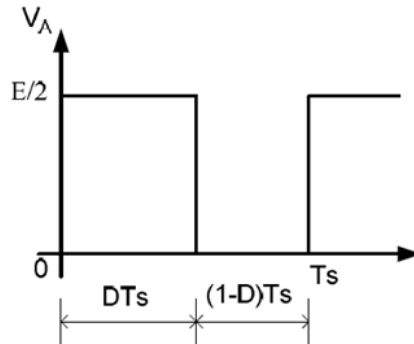


Fonte: produção do próprio autor

A forma da tensão no ponto A é mostrada na Figura 17. Pode-se obter o ganho estático através do cálculo da tensão média quase instantânea $\langle v_A \rangle$ considerando um período de comutação com tensão média positiva [12]. O ganho estático é dado pela Equação (4).

$$\frac{\langle v_A \rangle}{E/2} = d \quad (4)$$

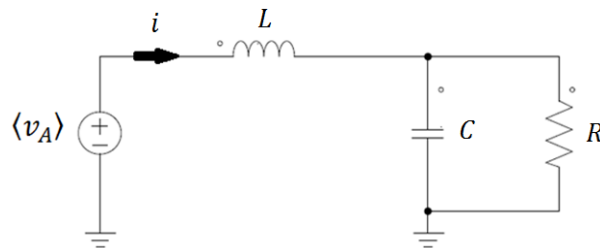
Figura 17 - Forma de onda no ponto A do conversor NPCm



Fonte: adaptado de [12]

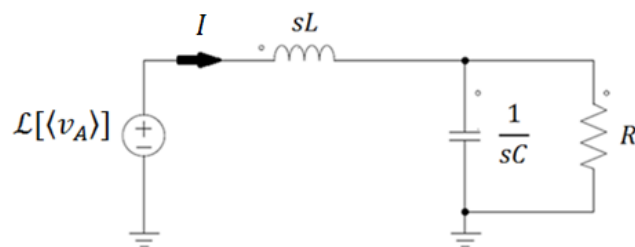
Considerando a tensão média quase instantânea $\langle v_A \rangle$ entre o ponto A e a referência zero, têm-se a malha apresentada na Figura 18. O circuito equivalente em Laplace é mostrado na Figura 19.

Figura 18 – Circuito equivalente da saída do conversor



Fonte: produção do próprio autor

Figura 19 - Circuito equivalente da saída do conversor em Laplace



Fonte: produção do próprio autor

A transformada de Laplace de $\langle v_A \rangle$ é apresentada na Equação (5). Como d varia com o tempo devido a evolução do sinal modulante e da portadora, podemos considerar d um $d(t)$.

$$\mathcal{L}[\langle v_A \rangle] = \mathcal{L}\left[\frac{E}{2}d(t)\right] = \frac{E}{2}D(s) \quad (5)$$

Efetuada a análise da malha apresentada na Figura 19 chega-se a Equação (6).

$$-\frac{E}{2}D(s) + sLI(s) + \left(\frac{R/sC}{1/sC + R}\right)I(s) = 0 \quad (6)$$

Desenvolvendo a Equação (6) têm-se:

$$\begin{aligned} sLI(s) + \left(\frac{R/sC}{1 + sRC}\right)I(s) &= \frac{E}{2}D(s) \\ sLI(s) + \left(\frac{R}{1 + sRC}\right)I(s) &= \frac{E}{2}D(s) \end{aligned} \quad (7)$$

$$I(s) \left[sL + \left(\frac{R}{1 + sRC}\right) \right] = \frac{E}{2}D(s)$$

$$I(s) \left[\frac{s^2 + 1/RC s + 1/LC}{s + 1/RC} \right] L = \frac{E}{2}D(s)$$

Considerando uma pequena perturbação na razão cíclica do conversor [15], têm-se:

$$(I(s) + \Delta I(s)) \left[\frac{s^2 + 1/RC s + 1/LC}{s + 1/RC} \right] L = (D(s) + \Delta D(s)) E/2 \quad (8)$$

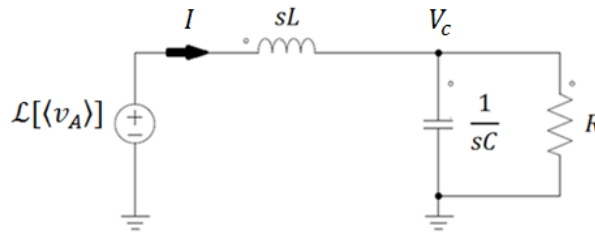
Subtraindo a Equação (8) de (7) obtêm-se:

$$\Delta I(s) \left[\frac{s^2 + 1/RC s + 1/LC}{s + 1/RC} \right] = \frac{1}{L} \Delta D(s) E/2 \quad (9)$$

$$\frac{\Delta I(s)}{\Delta D(s)} = \left[\frac{s + 1/RC}{s^2 + 1/RC s + 1/LC} \right] \left[\frac{E/2}{L} \right]$$

Entretanto neste trabalho, o interesse é pela planta de tensão do conversor. Para tal, considera-se então a tensão V_c de acordo com a Figura 20.

Figura 20 - Circuito equivalente da saída do conversor com indicação da tensão de saída



Fonte: produção do próprio autor

A tensão V_c pode ser encontrada a partir de:

$$V_c = I \left(\frac{R/sC}{R + 1/sC} \right) \quad (10)$$

$$V_c = I \left(\frac{1/C}{s + 1/RC} \right)$$

Assim:

$$I = V_c (s + 1/RC) C \quad (11)$$

$$\Delta I = \Delta V_c (s + 1/RC) C$$

Substituindo a Equação (11) em (9) têm-se:

$$\frac{\Delta V_c (s + 1/RC) C}{\Delta D(s)} = \left[\frac{s + 1/RC}{s^2 + 1/RC s + 1/LC} \right] \left[\frac{E/2}{L} \right] \quad (12)$$

$$\frac{\Delta V(s)}{\Delta D(s)} = \left[\frac{1}{s^2 + 1/RC s + 1/LC} \right] \left[\frac{E/2}{CL} \right]$$

Onde: $V_c = V(s)$.

5.2 CONTROLADOR PID DIGITAL

Por definição, o controlador contínuo PID é dado pela expressão apresentada na Equação (13) [10][11].

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d(\tau) + K_d \left(\frac{d}{dt} \right) e(t) \quad (13)$$

Onde: K_p é o ganho proporcional; K_i é o ganho integral; K_d é o ganho derivativo.

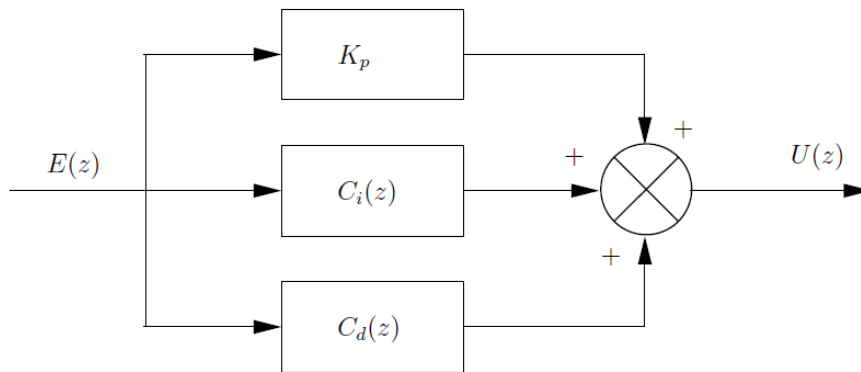
A fim de que os parâmetros do controlador PID possam ser associados aos polos e zeros da função de transferência de forma mais fácil, é conveniente a manipulação da Equação (13) de forma que os ganhos possam ser parametrizados conforme a Equação (14) [11].

$$u(t) = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \left(\frac{d}{dt} \right) e(t) \right) \quad (14)$$

Onde: T_i é o tempo integral; T_d é o tempo derivativo; $K_i = \frac{K_p}{T_i}$; $K_d = K_p T_d$.

A estrutura de um controlador PID na versão digital é apresentada na Figura 21.

Figura 21 - Estrutura de um controlador PID na versão digital



Fonte: [11]

Ou seja, de acordo com a Figura 21 temos:

$$C(z) = K_p + C_i(z) + C_d(z) \quad (15)$$

Os conceitos do controlador PID, historicamente foram definidos no domínio do tempo contínuo. Para utilizar esta técnica de controle no domínio do tempo discreto é necessário aplicar métodos que aproximam $C(z)$ de $u(t)$. Dependendo da aproximação utilizada para a integral e para a derivada, $C_i(z)$ e $C_d(z)$ podem assumir diferentes formas [10][11]. As aproximações discretas são efetuadas geralmente por três métodos bem conhecidos. São eles: *Forward differences*, *backward differences* e *tustin*. As demonstrações das aplicações dos métodos de aproximação são apresentadas em [11].

Neste trabalho serão apresentadas somente as equações dos controladores nas suas formas finais, obtidas a partir dos métodos citados.

- *Forward differences*:

$$C(z) = \frac{K_p \tau_d \left(z^2 + \left(\frac{1}{\tau_d} - 2 \right) z + \frac{\tau_i \tau_d - \tau_i + 1}{\tau_i \tau_d} \right)}{z - 1} \quad (16)$$

Este controlador possui dois zeros reais ou complexos e um polo em $z = 1$. Como a função de transferência possui mais zeros finitos do que polos finitos, a sua implementação não é realizável.

- *Backward differences:*

$$C(z) = \frac{K_p \left(1 + \frac{1}{\tau_i} + \tau_d\right) \left(z^2 - \frac{1 + 2\tau_d}{1 + \frac{1}{\tau_i} + \tau_d} z + \frac{\tau_d}{1 + \frac{1}{\tau_i} + \tau_d}\right)}{z(z - 1)} \quad (17)$$

Este controlador possui dois zeros reais ou complexos, um polo em $z = 0$ e outro em $z = 1$.

- *Tustin:*

$$C(z) = \frac{K_p \left(1 + \frac{1}{2\tau_i} + 2\tau_d\right) \left(z^2 + \frac{\frac{1}{\tau_i} - 4\tau_d}{1 + \frac{1}{2\tau_i} + 2\tau_d} z + \frac{\frac{1}{2\tau_i} + 2\tau_d - 1}{1 + \frac{1}{2\tau_i} + 2\tau_d}\right)}{(z + 1)(z - 1)} \quad (18)$$

Este controlador possui dois zeros reais ou complexos, um polo em $z = -1$ e outro em $z = 1$.

Como o controlador dado pela Equação (16) não é realizável, conclui-se a partir das Equações (17) e (18) que o controlador PID digital apresenta dois polos e dois zeros. As posições dos polos são determinadas pelo método de aproximação utilizado. As posições dos dois zeros são definidas pelos parâmetros do controlador [10][11]. Na seção 5.3 é apresentado o projeto do controlador PID digital que foi escrito em VHDL para a implementação prática.

5.3 PROJETO DO CONTROLE DE TENSÃO DO CONVERSOR NPCm

Em função das características do processo avaliado neste trabalho, ou seja, do uso de um FPGA para produzir os sinais PWM e efetuar o controle de um conversor NPCm, inevitavelmente o sistema é levado a operar no domínio do tempo discreto, considerando os efeitos da frequência de amostragem e a utilização de conversores A/D e D/A. O projeto de um controlador digital pode ser realizado diretamente no plano Z ou aplicando métodos que aproximam um controlador digital de um analógico previamente definido pelas técnicas

tradicionais da teoria de controle. Entretanto, estas aproximações adicionam erros desnecessários ao sistema [10][16].

Neste trabalho, a estrutura do sistema projetado, determina que o cálculo do erro entre o sinal de referência (gerado internamente) e o valor resultante da conversão A/D do sinal de saída da planta amostrado, bem como os cálculos da lei de controle, sejam realizados pelo *hardware* inferido no FPGA. Assim, a discretização da planta ocorre naturalmente e a opção por projetar um controlador digital no plano Z, parece uma escolha mais adequada, já que na prática o processo todo trata-se de um controlador digital atuando sobre uma planta discretizada.

5.3.1 Projeto do controlador digital no plano Z

Neste momento, vale ressaltar que o foco do trabalho é de avaliar a capacidade do FPGA EP4CE22F17C6N da família Cyclone IV E para receber uma arquitetura, capaz de realizar o controle de um conversor NPCm de três níveis, monofásico e trifásico. Assim, o trabalho não tem a pretensão de mostrar ao leitor, os detalhes matemáticos relacionados ao projeto do controlador, excetuando o caso do projeto do *Indirect Self-Tuning* que será apresentado na seção 5.6, onde a demonstração das equações do MQR é necessária para que o leitor compreenda o código escrito em VHDL. Portanto, para efetuar o projeto do controlador, a ferramenta SISOTOOL do *software* MATLAB[®] foi amplamente utilizada, com a finalidade de avaliar a estabilidade do processo e obter uma resposta aceitável na saída da planta. Neste contexto, uma flexibilidade nos valores dos pré-requisitos de projeto é admissível, já que o foco não é avaliar o desempenho do controlador projetado. Assim, definiu-se que o tempo de assentamento (t_s), deve ser pelo menos oito vezes menor do que período da fundamental ($T_s = 1/50$) do sinal de saída do conversor NPCm. A frequência de corte deve ser maior do que dez vezes a frequência da fundamental ($f = 50Hz$) e estar pelo menos uma década abaixo da frequência de chaveamento ($f_{port} = 40kHz$). A margem de fase deve ser menor do que 90° .

Os parâmetros da planta são mostrados na Tabela 8 e a função de transferência (FT) da planta com os valores dos parâmetros já inseridos é apresentada na Equação (19) .

Tabela 8 – Parâmetros do conversor NPCm utilizado

Parâmetros	Valores
Resistência (Resistor R - Carga)	100 Ω
Capacitância (Capacitor C – Filtro LC)	4,3 μF
Indutância (Indutor L – Filtro LC)	3,3 mH
Tensão do Barramento CC (Fonte E)	30 V_{cc}

Fonte: produção do próprio autor

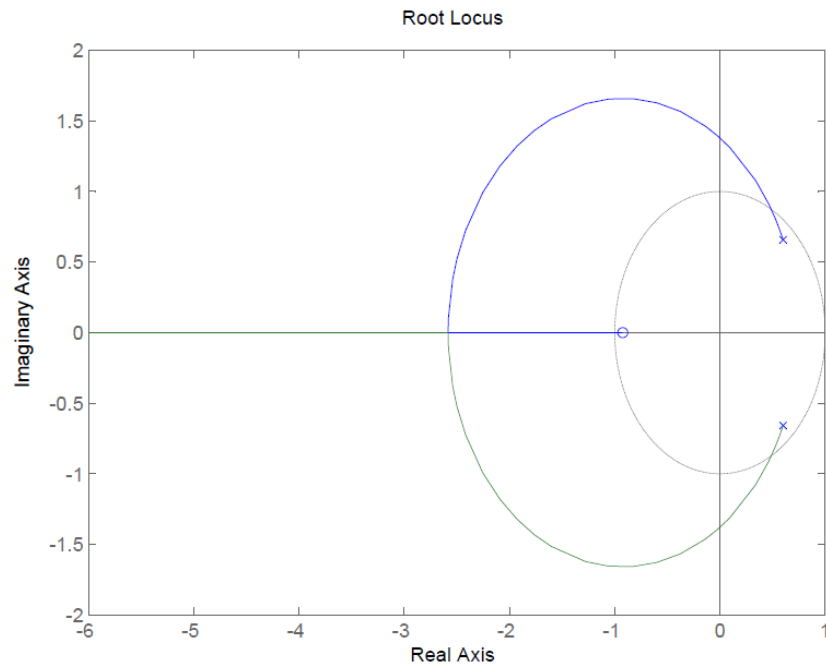
$$G_{planta}(s) = \frac{1,057 \times 10^9}{s^2 + 2326s + 7,047 \times 10^7} \quad (19)$$

A discretização da planta foi concebida através da inserção da mesma entre dois retentores de ordem zero (ZOH - *Zero Order Hold*). A frequência de amostragem (F_s) foi definida em 10kHz, logo o período de amostragem (T_s) foi estabelecido em 100 μs . No software MATLAB®, inserindo o comando “c2d” e informando a FT da planta (Equação (19)), o período de amostragem (100 μs) e o método de discretização (ZOH) obtêm-se facilmente a Equação (20).

$$G_{planta}(z) = \frac{4,622z + 4,269}{z^2 - 1,2z + 0,7925} \quad (20)$$

A Figura 22 exhibe a localização dos polos e do zero da planta dada pela Equação (20) no diagrama de lugar das raízes no plano Z.

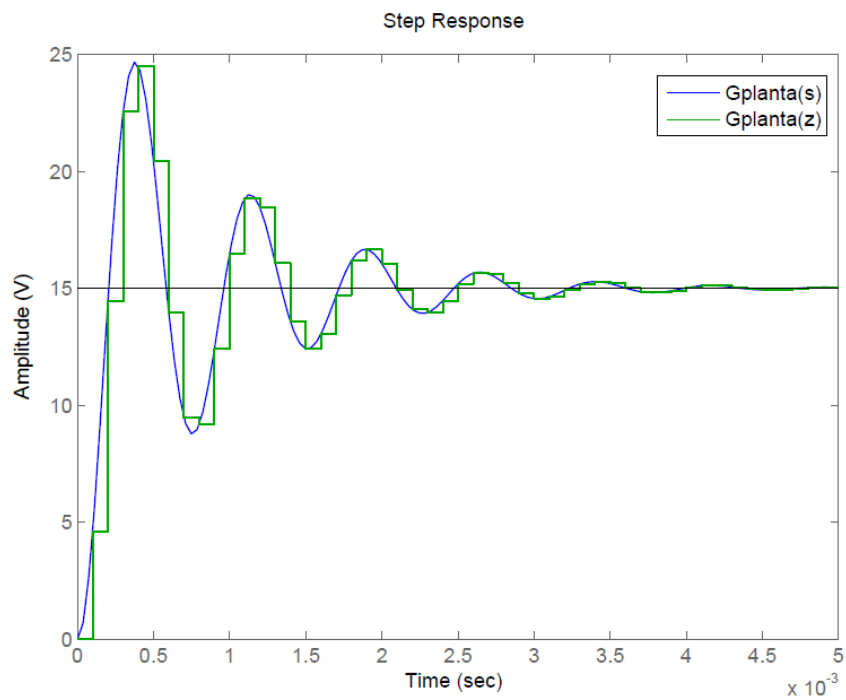
Figura 22 - Lugar das raízes da planta discretizada



Fonte: produção do próprio autor

A resposta para entrada degrau, dos sistemas dados pelas Equações (19) e (20), em malha aberta, é apresentada na Figura 23.

Figura 23 – Resposta a entrada degrau das plantas em malha aberta



Fonte: produção do próprio autor

Para projetar o controlador $C(z)$ foi utilizado o conceito de cancelamento de polos e zeros [16][17]. No editor de lugar das raízes do ambiente de projeto SISO (*Single Input Single Output*) do software MATLAB®, foram adicionados zeros complexos e conjugados a fim de cancelar os polos de $G_{planta}(z)$ e dois polos no eixo real para impor a dinâmica desejada. Um dos polos foi posicionado em $z = 1$, com o intuito de eliminar o erro de regime permanente para entrada degrau e o outro polo em $z = -0,267$ para posicionar os ramos no local desejado. O ganho do modulador PWM (G_{pwm}) também deve ser considerado [12][13].

$$G_{pwm} = \frac{1}{V_t}$$

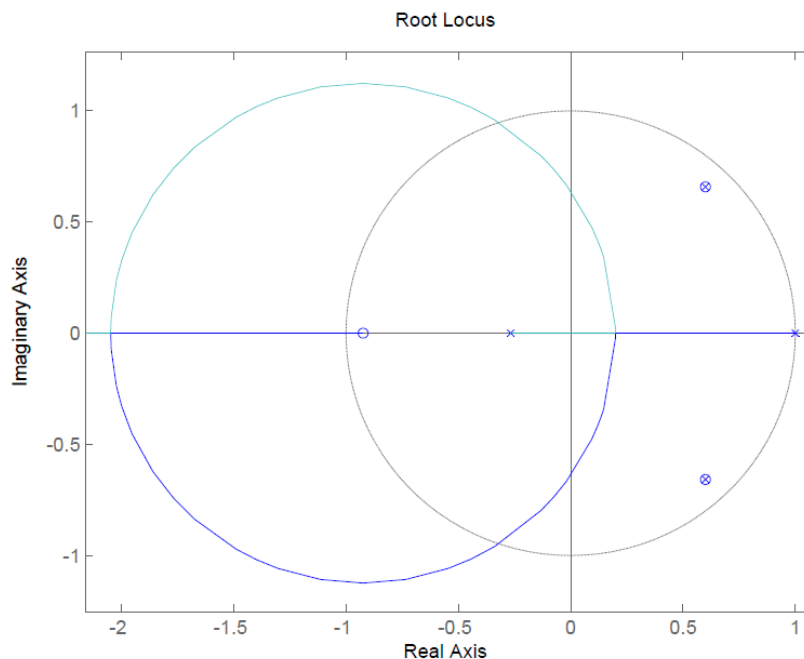
$$G_{pwm} = \frac{1}{15,7579} \quad (21)$$

$$G_{pwm} = 0,0635$$

Onde V_t é a amplitude da portadora utilizada neste projeto.

Na Figura 24 é apresentado o diagrama de lugar das raízes de $C(z)G_{pwm}G_{planta}(z)$.

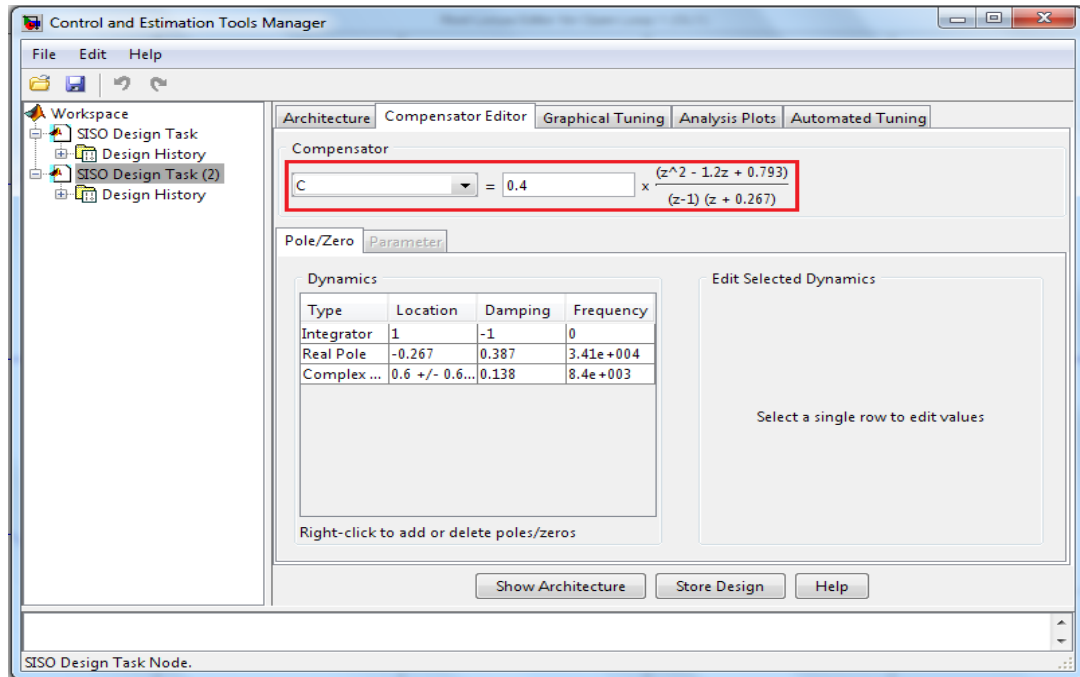
Figura 24 – Lugar das raízes do sistema com o controlador e a planta



Fonte: produção do próprio autor

A caixa com contornos em vermelho exibida na Figura 25, apresenta a FT do controlador $C(z)$ obtido com o ambiente de projeto SISO do *software* MATLAB®.

Figura 25 – Controlador obtido no ambiente de projeto SISO



Fonte: produção do próprio autor

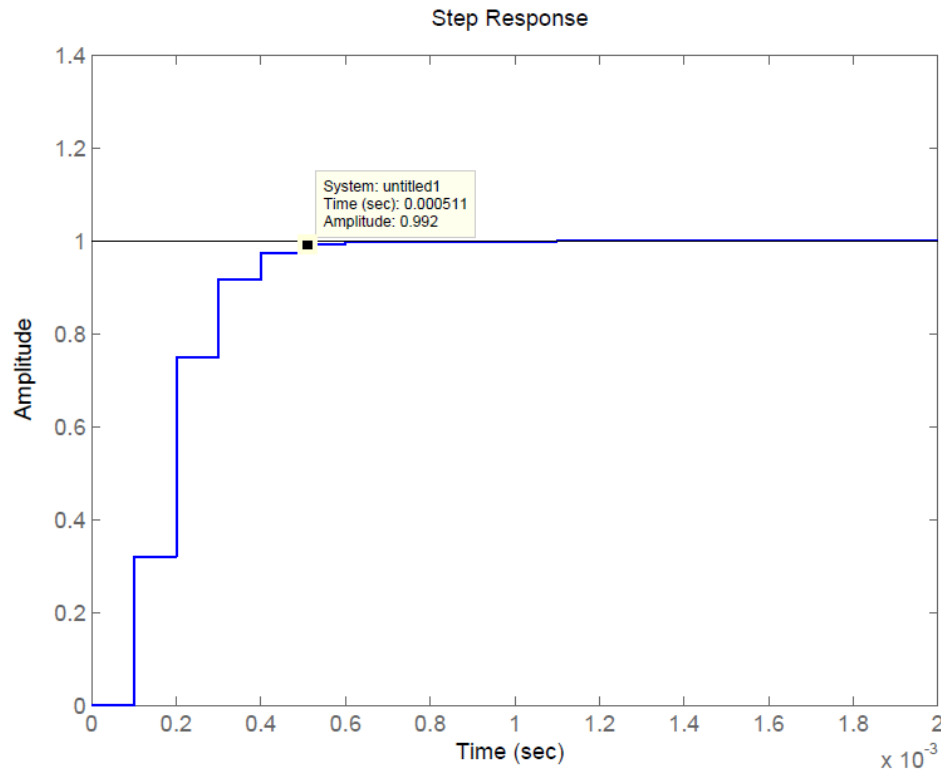
A Equação (22) apresenta a FT de $C(z)$.

$$C(z) = \frac{0,4(z^2 - 1,2z + 0,793)}{(z - 1)(z + 0,267)} \quad (22)$$

O controlador projetado se aproxima de um PID digital, de acordo com as definições apresentadas na seção 5.2. Pois como mencionado anteriormente $C(z)$ apresenta dois polos no eixo real, sendo que um deles está em $z = 1$ e dois zeros dentro do círculo unitário. A diferença, está no fato do outro polo estar localizado em uma posição do eixo real diferente de zero e -1 .

A Figura 26 apresenta a resposta do sistema $C(z)G_{pwm}G_{planta}(z)$, em malha fechada, a um degrau unitário.

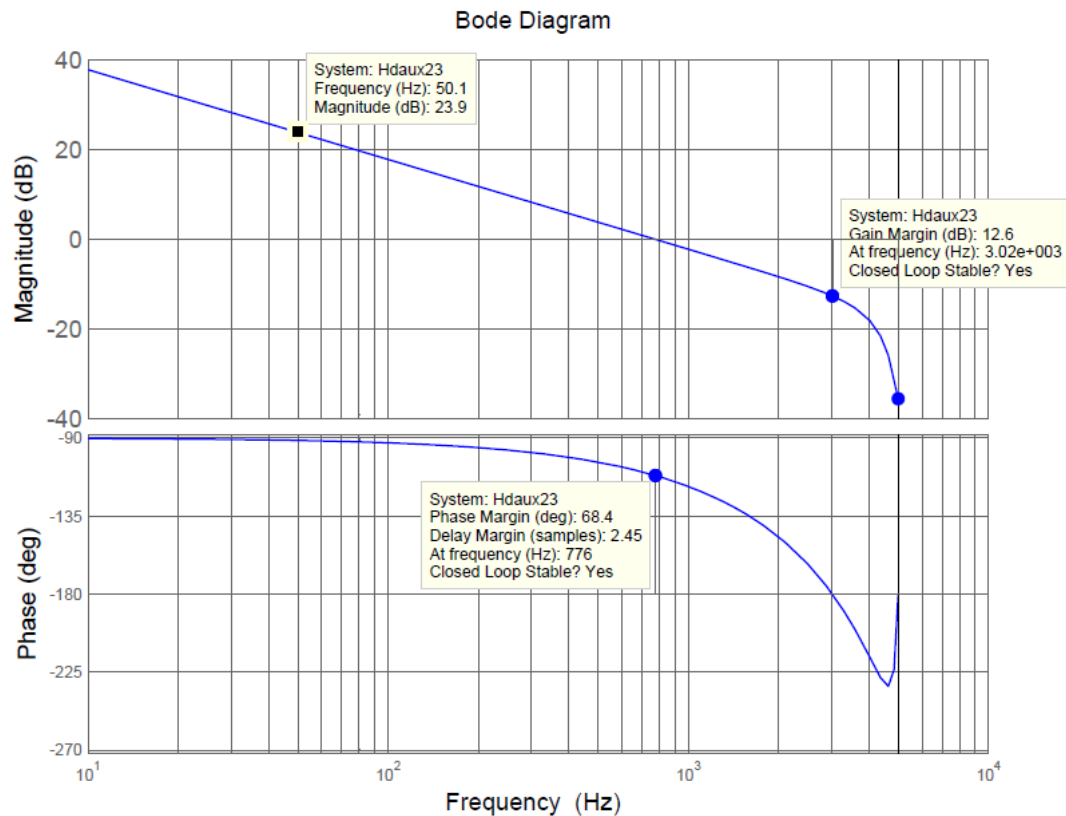
Figura 26 – Resposta ao degrau do sistema $C(z)G_{pwm}G_{planta}(z)$ em malha fechada



Fonte: produção do próprio autor

Abaixo, na Figura 27 é mostrado o diagrama de Bode do sistema $C(z)G_{pwm}G_{planta}(z)$, em malha aberta. Pode-se observar que o ponto de passagem por zero dB se dá na frequência de $776Hz$, ou seja, maior do que dez vezes [18] a frequência da fundamental do sinal de saída que é de $50Hz$. Nota-se também, que na frequência de $50Hz$ o ganho é de aproximadamente $24 dB$, estando acima do mínimo recomendado ($20dB$). A margem de fase é de $68,4^\circ$, estando um pouco afastado do valor recomendado (53°)[18], porém dentro do limite de até 90° e a margem de ganho é $12,6 dB$. Por meio do diagrama de Bode gerado pelo *software* MATLAB[®] é possível verificar a estabilidade do sistema em malha fechada. Na Figura 27, nota-se a indicação de sistema estável nos *labels*.

Figura 27 – Diagrama de bode do sistema $C(z)G_{pwm}G_{planta}(z)$



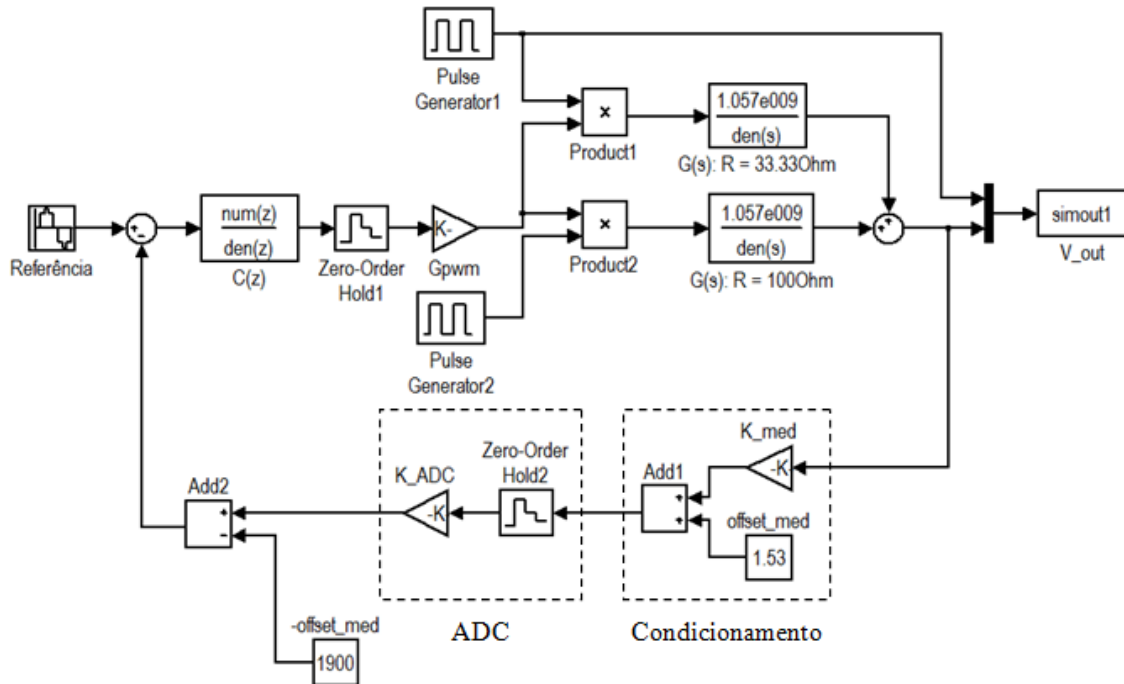
Fonte: produção do próprio autor

Vale ressaltar, que outros controladores foram projetados, resultando em uma largura de banda e margem de fase mais adequadas. Porém, a implementação na prática não foi bem sucedida e os motivos não foram analisados por não ser este o objetivo do trabalho.

5.4 COMPROVAÇÃO DO MODELO MATEMÁTICO

Para comprovar que o modelo matemático representado pela Equação (19) é coerente, uma comparação entre o modelo matemático e o modelo comutado foi realizada através das simulações feitas no *software* MATLAB® (para o modelo matemático) e no *software* PSIM® (para o modelo comutado). O modelo matemático foi simulado por meio do ambiente Simulink. Na Figura 28 segue o sistema criado para realizar a simulação do modelo matemático.

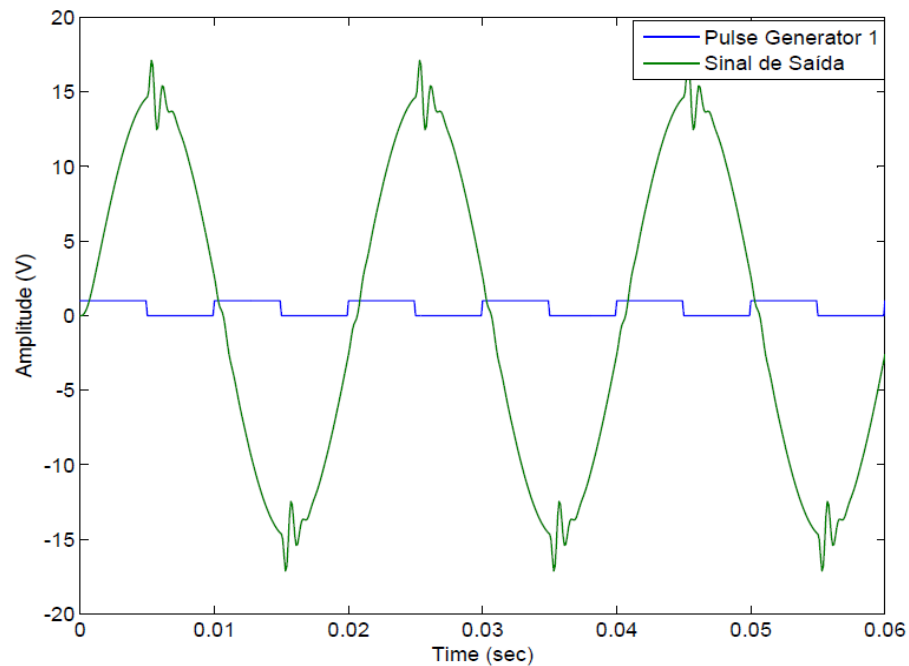
Figura 28 – Diagrama de simulação do modelo matemático, usando o Simulink



Fonte: produção do próprio autor

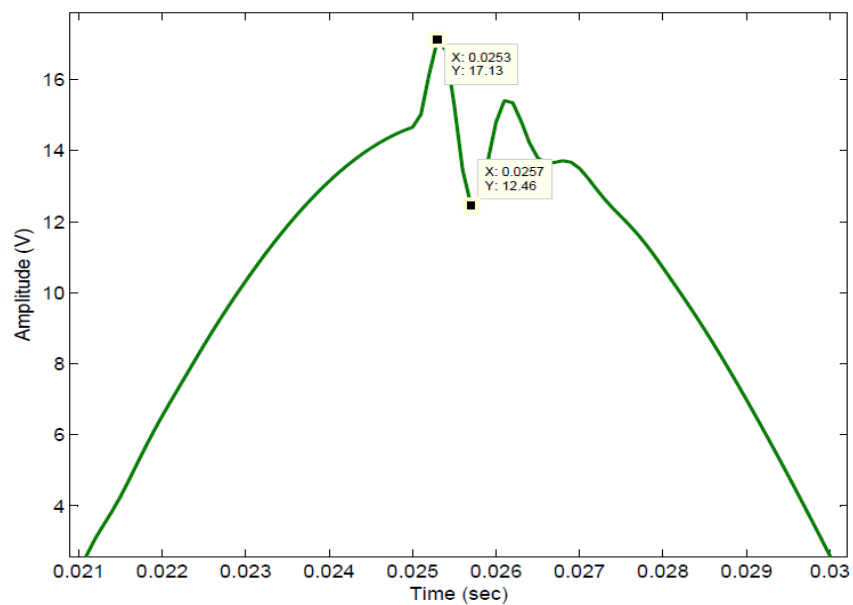
A estrutura apresentada na Figura 28 é uma aproximação da estrutura real. Através dos blocos multiplicadores *Product1* e *Product2* é possível simular a adição de uma carga e retirada da mesma na saída da planta, já que a alteração do valor da carga na saída do sistema altera o modelo da planta. Assim, foram definidas duas plantas, uma para a carga resistiva de 100Ω e outra para uma carga resistiva de $33,33\Omega$, que é resultado da associação em paralelo de dois resistores, um de 100Ω e outro de 50Ω . O sinal de saída do sistema é aplicado ao circuito de condicionamento, que atenua e atribui um *offset*, para adequá-lo aos níveis de tensão permitido pelo KIT DE0-Nano, para uso do conversor A/D. O módulo ADC (*Analog-to-Digital Converter*) realiza a amostragem através do retentor de ordem zero e atribui o ganho do conversor A/D. Na sequência do módulo ADC o *offset* é retirado, retratando o que ocorre na prática, pois foi programado esta subtração no código em VHDL. A primeira planta a operar é o sistema com uma carga resistiva de $33,33\Omega$. Na sequência, entra a planta com a carga resistiva de 100Ω provocando uma elevação da tensão na saída, como pode ser observado nas Figuras 29 e 30.

Figura 29 – Resultado da simulação do modelo matemático com mudança de carga



Fonte: produção do próprio autor

Figura 30 – Resultado da simulação do modelo matemático com mudança de carga (zoom)



Fonte: produção do próprio autor

A ondulação de tensão após a retirada de carga, chega a 4,67V.

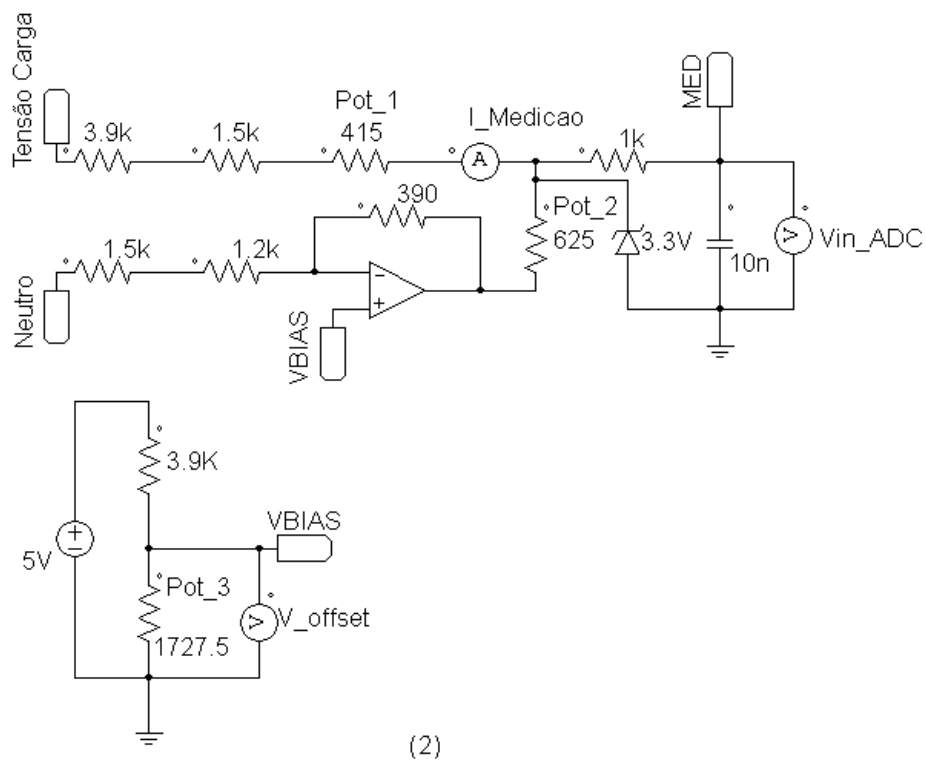
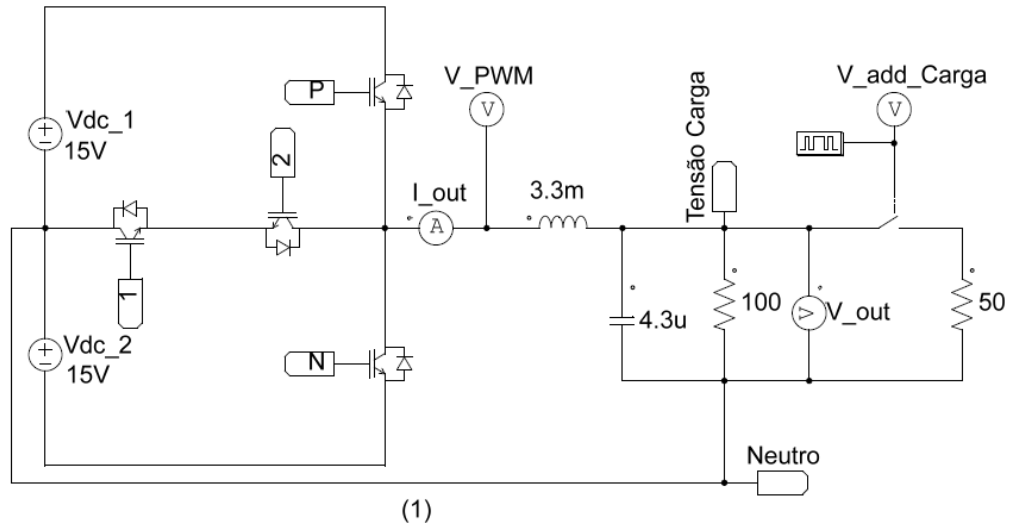
O modelo comutado é apresentado nas figuras que vão de 31(1) a 31(6). A Figura 31(1) exibe a estrutura do conversor NPCm monofásico conectado a um filtro LC e a uma carga resistiva. A adição e retirada de carga é realizada de forma periódica.

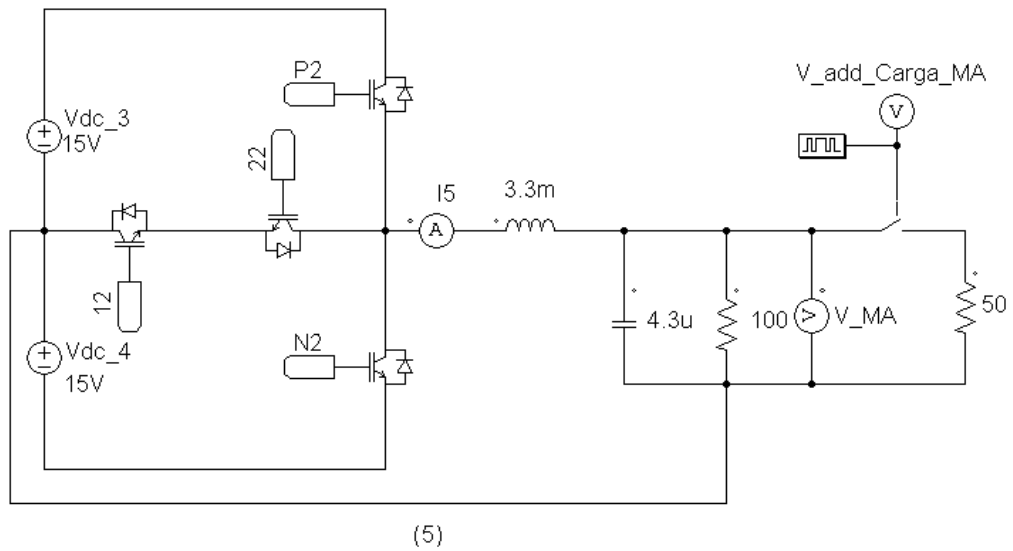
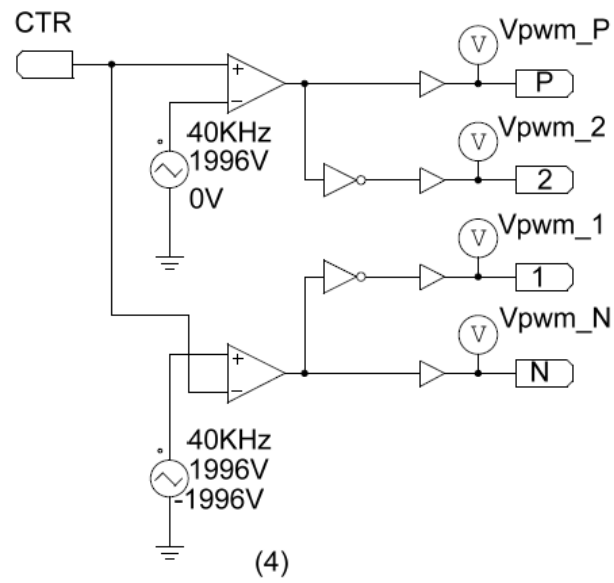
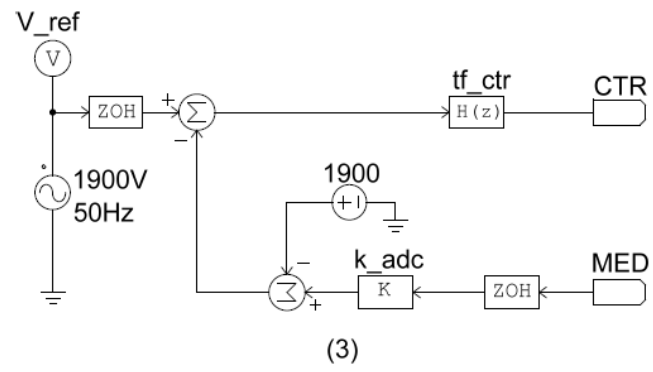
O circuito de condicionamento do sinal de saída é apresentado na Figura 31(2). Este circuito possui um amplificador operacional que permite a adição de um *offset*, deslocando o sinal de saída para valores positivos. O circuito também atenua, por meio de um divisor resistivo, o valor de pico a pico do sinal medido que idealmente é de aproximadamente $30V$, para aproximadamente $3,06V$. A saída deste circuito possui um diodo zener de $3,6V$ para limitar a magnitude do sinal de saída e proteger a entrada do conversor A/D do KIT DE0-Nano. A tensão de saída (chamada de V_{in_ADC}) deste circuito pode ser verificada na Figura 32.

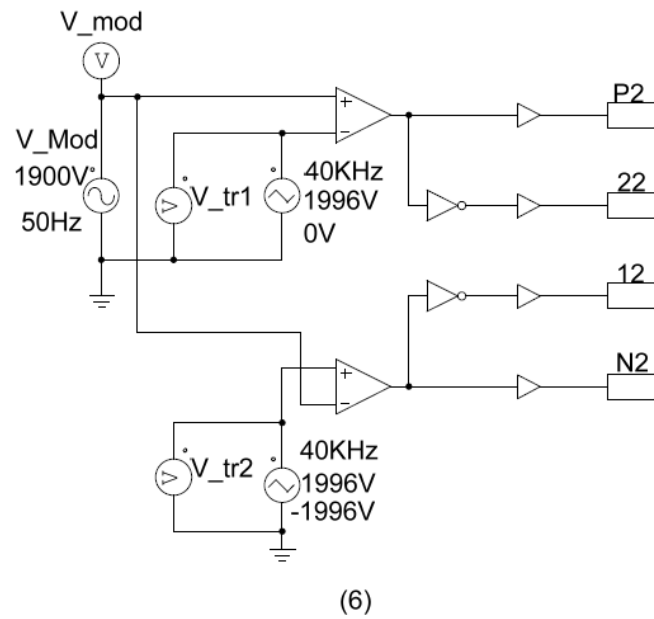
A malha de controle é exibida na Figura 31(3). O sinal de saída do circuito de condicionamento é amostrado pelo bloco ZOH. Após a amostragem é atribuído um ganho referente ao ganho do conversor A/D. A frequência de amostragem é de $10kHz$ e o ganho é de $1241,83$, que amplifica o sinal medido para um valor de pico a pico de aproximadamente 3800 . Como o conversor A/D é um conversor de $12\ bits$ e a tensão de referência proporcionada pelo KIT DE0-Nano é de $3,3V$, este valor ($3,3$) é representado por um valor máximo de 4095 em decimal ($2^{12} - 1$). Para não operar no limite, optou-se em trabalhar com um sinal de $3,06V_{pp}$, que convertido equivale a 3800 em decimal. Após o estágio de conversão é realizado uma subtração, a fim de eliminar o *offset* adquirido na etapa de condicionamento. O valor subtraído é de 1900 , ou seja, o sinal resultante é uma senoide com amplitude máxima de ± 1900 . Assim, quando o sinal de saída do conversor NPCm apresentar uma amplitude de $15V_p$, a amplitude do sinal antes do estágio de obtenção do erro, será de 1900 , que é equivalente a amplitude máxima do sinal de referência. O elemento somador determina o erro entre o sinal de referência e o sinal medido. O erro é atribuído a FT do controlador.

O modulador é apresentado na Figura 31(4). O sinal proveniente do controlador é utilizado por dois comparadores. Um dos comparadores realiza a comparação entre o sinal de controle e a portadora positiva (triangular) e o outro efetua a comparação entre o sinal de controle e a portadora negativa (triangular). A amplitude máxima das portadoras é de 1996 . O modulador fornece os sinais PWM das quatro chaves do conversor NPCm.

Figura 31 – Sistema para simulação do modelo comutado, usando PSIM®: (1) Conversor NPCm com filtro LC e dispositivo de chaveamento de carga; (2) Circuito de condicionamento do sinal de tensão medido; (3) Malha de controle; (4) Circuito modulador; (5) Circuito para simulação em malha aberta; (6) Circuito para simulação em malha aberta

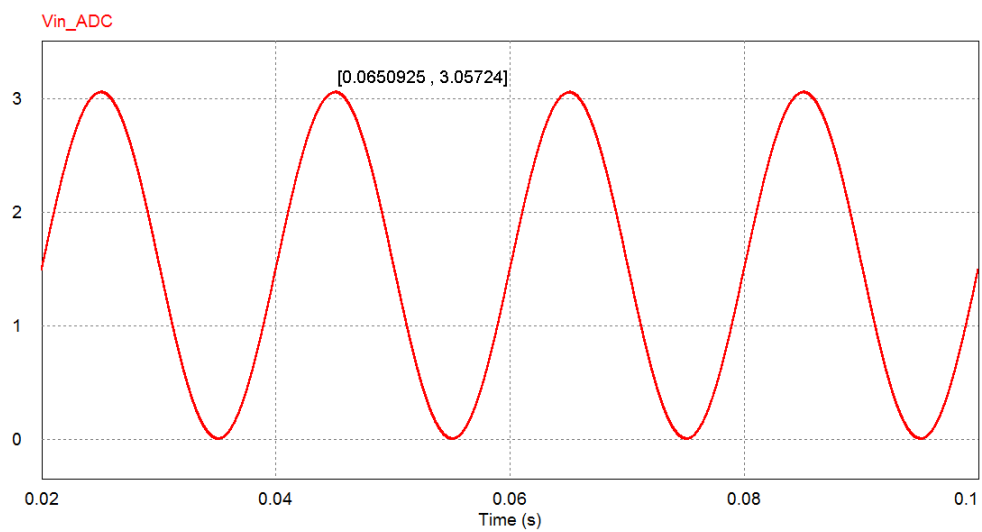






Fonte: produção do próprio autor

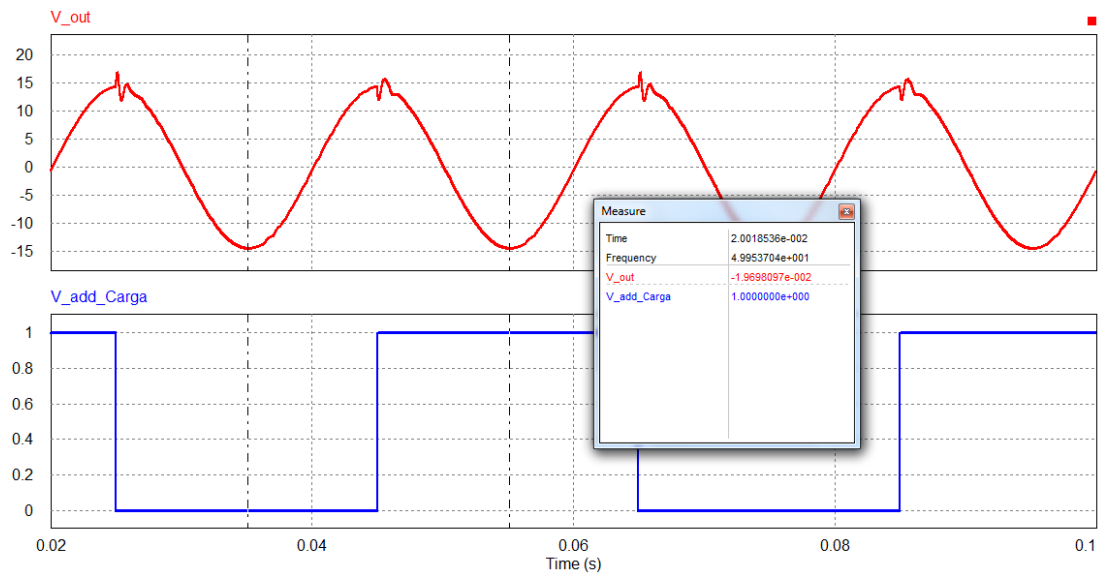
Figura 32 – Forma da tensão na saída do circuito de condicionamento



Fonte: produção do próprio autor

Simulando o sistema composto pelos circuitos apresentados nas Figuras 31(1), 31(2), 31(3) e 31(4) obtêm-se a partir da sonda de tensão chamada de V_{out} (Figura 31(1)) as Figuras 33 e 34.

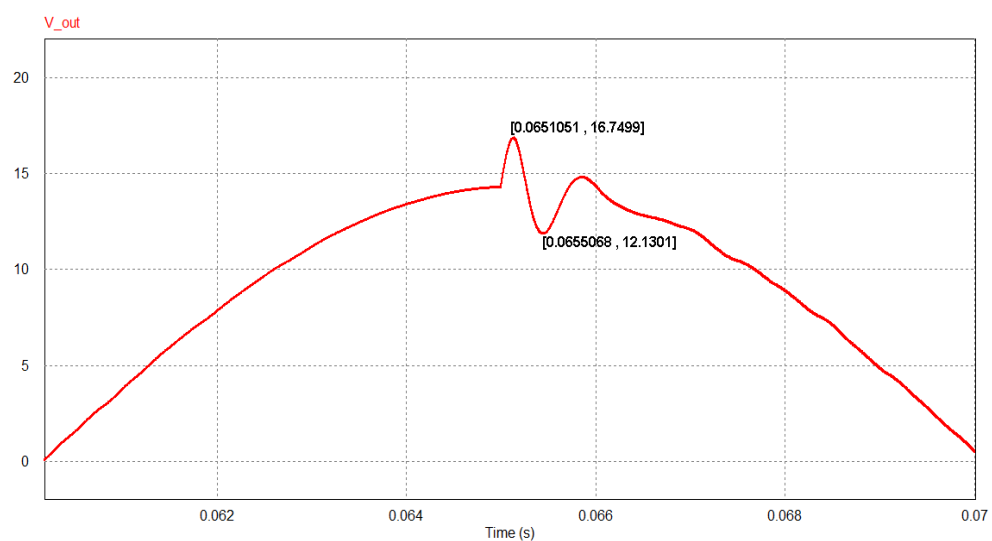
Figura 33 – Resultado da simulação do modelo comutado com mudança de carga



Fonte: produção do próprio autor

O sinal chamado de V_{add_Carga} mostra o momento da adição e retirada de carga. Quando o sinal está em nível alto, a carga de 50Ω está em paralelo com a carga de 100Ω e quando o sinal está em nível baixo, a carga na saída do conversor NPCm é de 100Ω . Na Figura 34 pode-se observar que a ondulação máxima de tensão no momento da retirada da carga de 50Ω é de aproximadamente $4,62V$.

Figura 34 – Resultado da simulação do modelo comutado com mudança de carga (zoom)

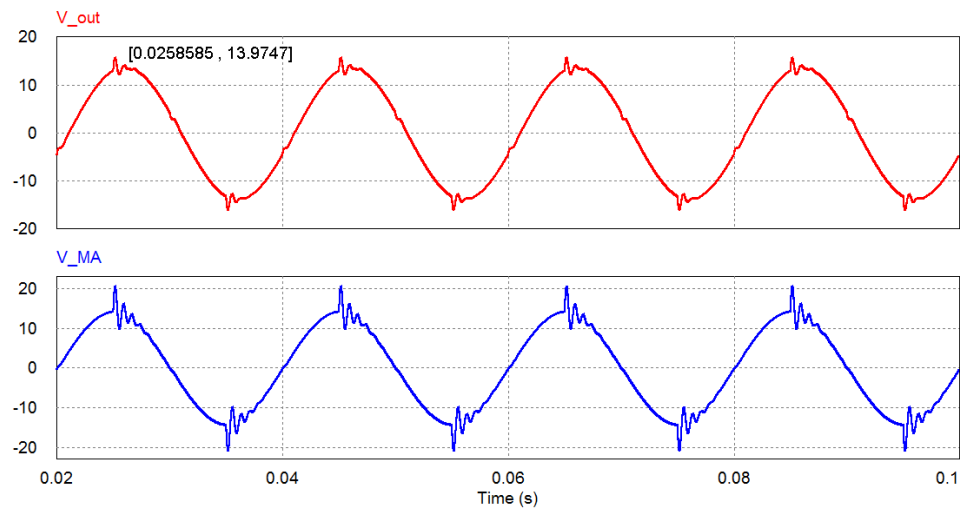


Fonte: produção do próprio autor

Comparando as simulações do modelo matemático (Figura 30) com o modelo comutado (Figura 34) conclui-se que o modelo matemático da planta é adequado, pelo menos no que se refere a um sistema ideal, e pode ser utilizado para o projeto do controlador.

Os circuitos apresentados nas Figuras 31(5) e 31(6) foram criados para a comparação dos resultados em malha aberta e malha fechada. A comparação é apresentada na Figura 35.

Figura 35 – Saída do sistema controlado V_{out} e saída do sistema em malha aberta V_{MA}

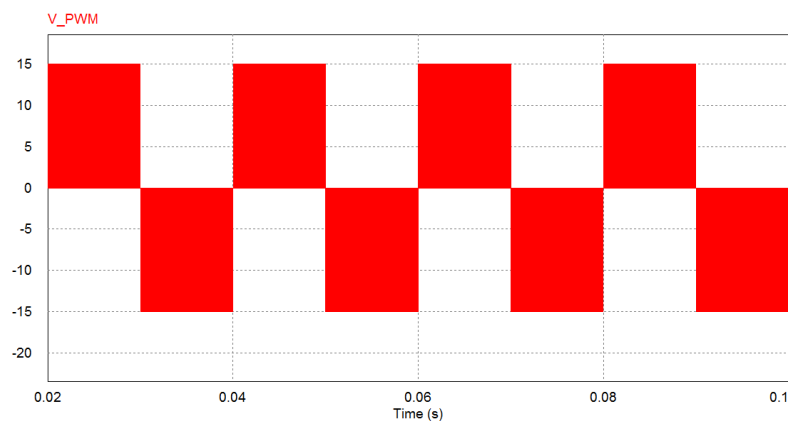


Fonte: produção do próprio autor

5.5 SINAIS RELEVANTES DO SISTEMA SIMULADO

Nas Figuras 36, 37, 38 e 39 são mostrados alguns sinais relevantes do sistema. As Figuras 36 e 37 apresentam o sinal adquirido pela sonda de tensão V_{PWM} (Figura 31(1)) que está posicionada antes do filtro LC.

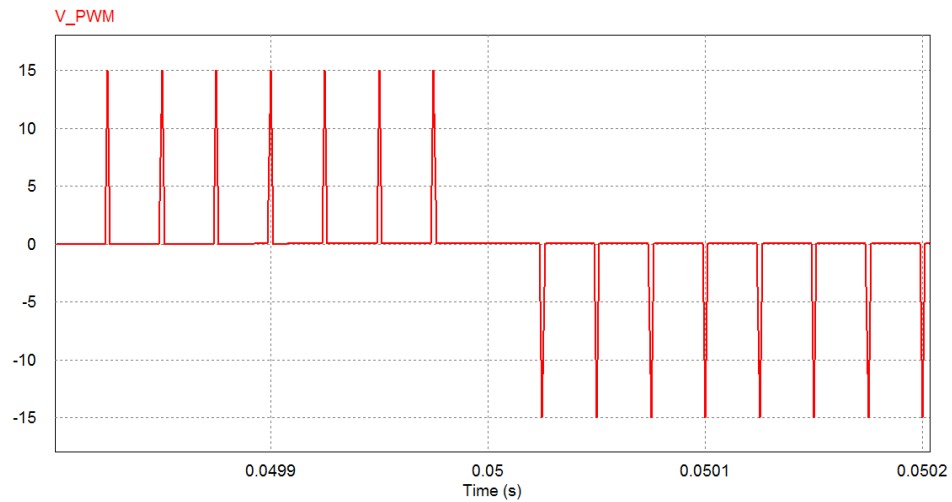
Figura 36 – Resultado da simulação para a tensão no ponto V_{PWM} da Figura 31(1)



Fonte: produção do próprio autor

Na Figura 37 é possível verificar os três níveis.

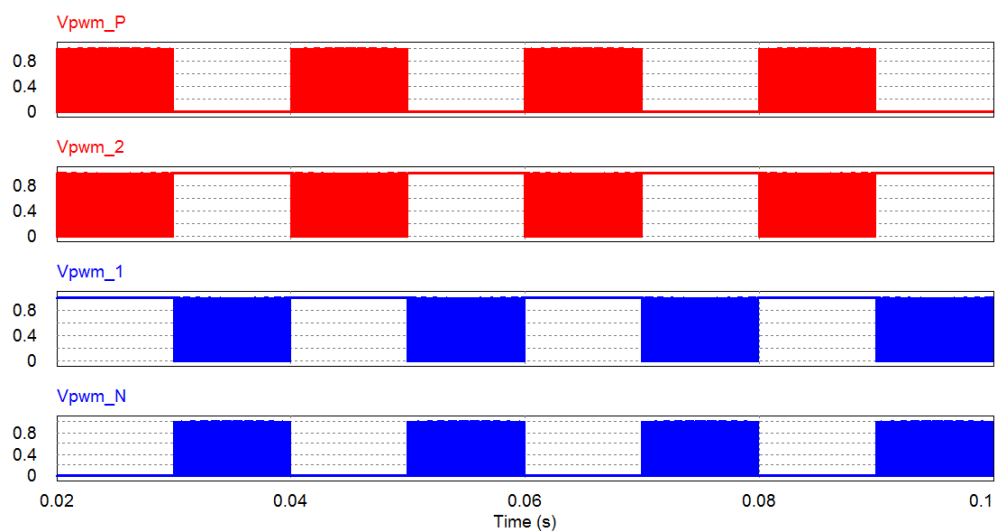
Figura 37 – Resultado da simulação para a tensão no ponto V_PWM da Figura 31.1 (zoom)



Fonte: produção do próprio autor

Os sinais dos canais PWM adquiridos pelas sondas de tensão Vpwm_P, Vpwm_2, Vpwm_1 e Vpwm_N (Figura 31(4)) são exibidos na Figura 38.

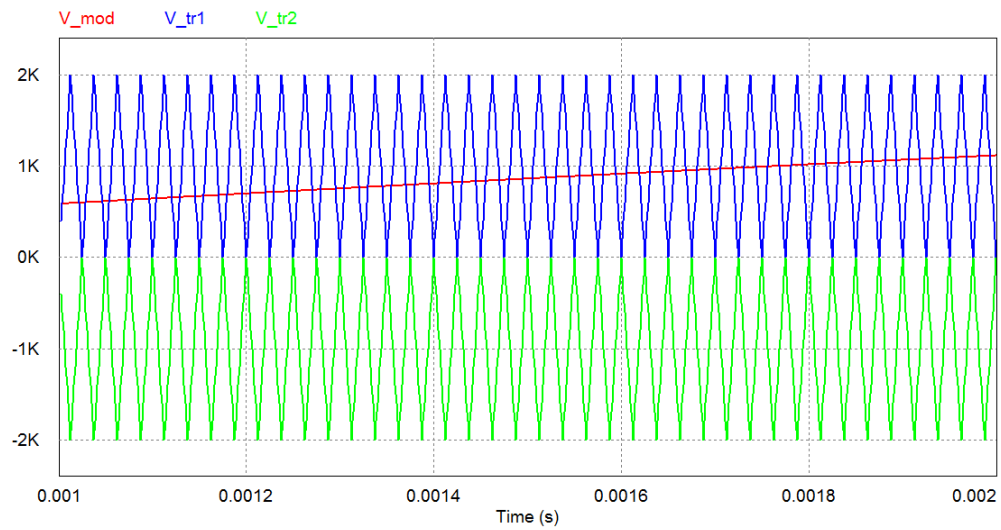
Figura 38 – Resultado da simulação para os sinais dos canais PWM



Fonte: produção do próprio autor

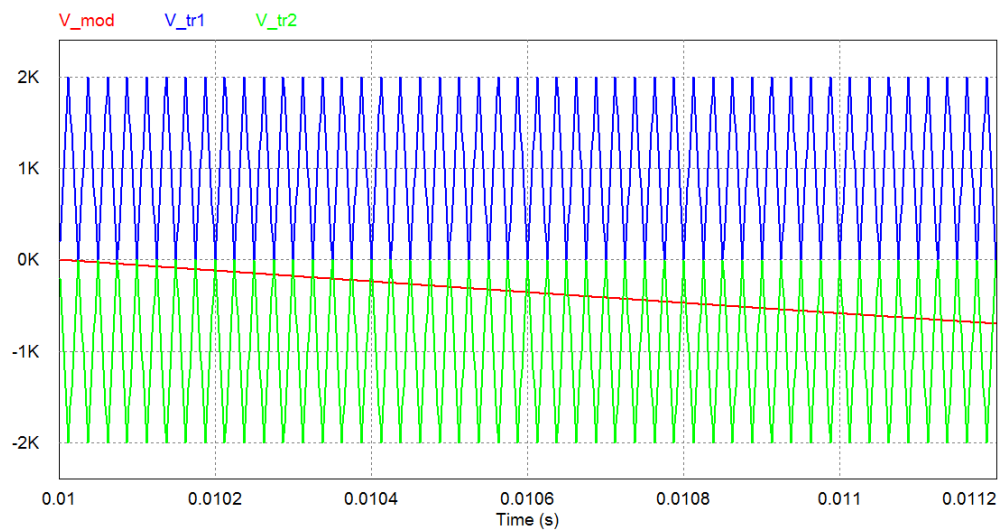
A moduladora e as duas portadoras triangulares adquiridas pelas sondas de tensão V_mod, V_tr1 e V_tr2 respectivamente (Figura 31(6)) são apresentadas na Figura 39 (parte do semiciclo positivo) e 40 (parte do semiciclo negativo).

Figura 39 – Resultado da simulação da modulação POD (semiciclo positivo)



Fonte: produção do próprio autor

Figura 40 – Resultado da simulação da modulação POD (semiciclo negativo)



Fonte: produção do próprio autor

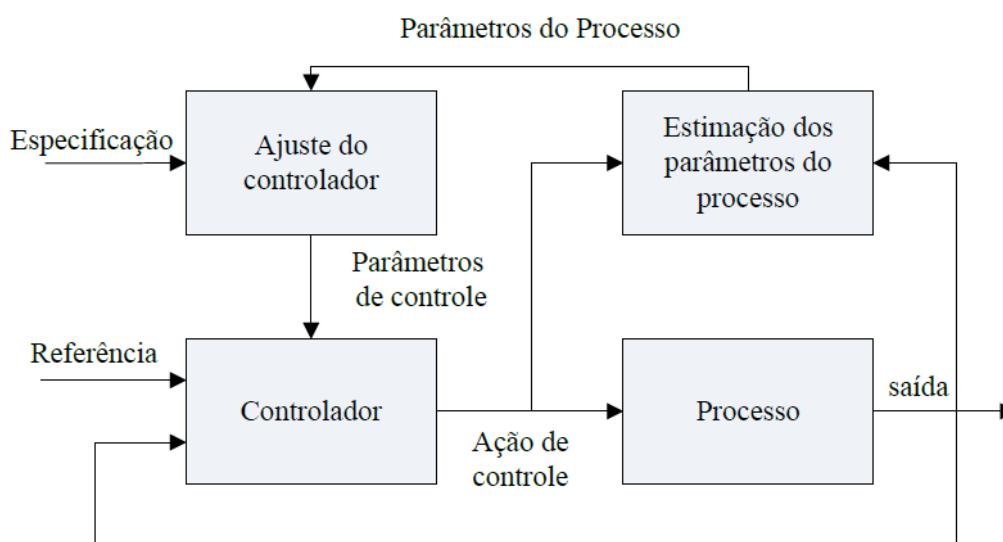
5.6 CONTROLE DIGITAL ADAPTATIVO

A maioria dos sistemas físicos reais não são lineares. Portanto ao realizar um projeto considerando o comportamento de um processo linear em torno de um ponto de operação, já devemos ter em mente que as características de desempenho e estabilidade só são válidas em torno deste ponto. A medida que as variáveis do sistema se afastam da região de operação linear o desempenho do sistema controlado é comprometido [20][21].

A principal proposta das técnicas de controle adaptativo é obter a atualização do modelo do sistema a cada período de amostragem e com isso re-sintonizar o controlador. Assim as variações nas variáveis do processo devido ao clima, aos desgastes mecânicos por exemplo, teriam um impacto menor no desempenho do sistema, já que um novo modelo é levantado a cada período de amostragem. Vale ressaltar, que após a atualização do modelo da planta é possível utilizar as mesmas técnicas lineares já conhecidas para controlar o sistema. O método de identificação online de sistemas para realizar a sintonia de controladores é denominado Controle Auto-ajustável ou Controle *Self-tuning* [20].

O projeto do controle do tipo *Self-Tuning* pode ser realizado basicamente de duas formas, direta ou indireta. Neste trabalho não será comentado o método direto, já que a técnica implementada em VHDL foi a indireta. No método *Indirect Self-Tuning* estima-se o modelo do processo e os parâmetros estimados são utilizados para o cálculo da lei de controle. Para implementar a técnica *Indirect Self-Tuning*, podemos recorrer ao método de identificação de sistemas conhecido como Método dos Mínimos Quadrados (MQ) [20][21]. Nota-se na Figura 41, a presença de uma malha externa, que estima os parâmetros da planta, os quais são utilizados para ajuste do controlador de forma recursiva. Isto é o grande diferencial em relação aos controladores estáticos [20][21].

Figura 41 - Diagrama em blocos do controle adaptativo indireto



Fonte: [21]

5.6.1 Mínimos quadrados (MQ)

O princípio dos Mínimos Quadrados foi desenvolvida por Karl Friedrich Gauss no século XXIII, com o objetivo de estudar as órbitas de corpos celestes. O conceito desenvolvido por Gauss afirma que: “a soma dos quadrados das diferenças entre o valor observado e o calculado, multiplicado por constantes que medem o grau de precisão é um mínimo” [20].

Uma possível representação de uma função de transferência de um sistema, na forma discreta, é apresentado abaixo na Equação 23:

$$\begin{aligned} y(k) + a_1 y(k-1) + \dots + a_n y(k-n) \\ = b_0 u(k-1) + \dots + b_{m-1} u(k-m) \end{aligned} \quad (23)$$

onde: a_1 até a_n e b_0 até b_m são os parâmetros do modelo na forma de uma equação a diferenças. Estes parâmetros devem ser estimados em um método de controle adaptativo. Com a finalidade de estimar estes parâmetros, a Equação (23) pode ser representada por:

$$y(k) = \varphi^T \theta \quad (24)$$

Onde θ é um vetor com elementos desconhecidos e φ é um vetor que possui as medidas de entrada e saída do sistema a cada período de amostragem:

$$\begin{aligned} \theta &= [a_1 \dots a_n \ b_0 \dots b_{m-1}]^T \\ \varphi^T &= [-y(k-1), \dots -y(k-n), \ u(k-1), \dots u(k-m)] \end{aligned} \quad (25)$$

No processo de estimação devemos considerar que os vetores θ e φ não fornecem o valor exato da saída $y(k)$ dada pela Equação (24). Deve-se considerar um erro ε decorrente da aproximação de $y(k)$ obtida com os vetores θ e φ , conhecidos como vetores de regressores [20][21]. Tal erro pode ser encontrado a partir das Equações (26) e (27).

$$\hat{y}(k) = \varphi^T \hat{\theta} \quad (26)$$

$$y(k) = \varphi^T \hat{\theta} + \varepsilon(k) \quad (27)$$

Para indicar que a variável utilizada é estimada utilizamos o operador $\hat{\cdot}$.

Assim:

$$\varepsilon(k) = y(k) - \varphi^T \hat{\theta} = y(k) - \hat{y}(k) \quad (28)$$

Utilizando a notação vetorial para reescrever a Equação (28) temos:

$$\varepsilon = Y - \Phi \hat{\theta} \quad (29)$$

Obviamente, ao se reduzir o erro, a saída estimada se aproxima do valor da saída real. Logo $\hat{\theta}$ deve ser um vetor com valores propícios para que o erro seja reduzido a um valor mínimo. O problema de aproximar a estimação à saída Y , pode ser solucionado com a minimização de um índice de desempenho que quantifica a qualidade do ajuste $\Phi \hat{\theta}$ [20][21]. Este índice de desempenho é dado pela Equação (33).

$$J_{MQ} = \sum_{i=1}^k \varepsilon(i)^2 = \varepsilon^T \varepsilon \quad (30)$$

Substituindo a Equação (29) na Equação (30) tem-se:

$$J_{MQ} = (Y - \Phi \hat{\theta})^T (Y - \Phi \hat{\theta}) \quad (31)$$

Realizando as devidas manipulações e aplicando algumas propriedades inerentes a matriz transposta obtêm-se a Equação (32).

$$J_{MQ} = Y^T Y - Y^T \Phi \hat{\theta} - \hat{\theta}^T \Phi^T Y + \hat{\theta}^T \Phi^T \Phi \hat{\theta} \quad (32)$$

Ao derivarmos parcialmente a função de custo (Equação (31)) em relação a $\hat{\theta}$ e igualarmos à zero, obtêm-se o ponto de máximo ou de mínimo. Para que o ponto seja de mínimo é preciso que a segunda derivada parcial seja maior que zero. Assim podemos minimizar o índice de desempenho [20][21].

$$\frac{\partial J_{MQ}}{\partial \hat{\theta}} = -\Phi^T Y - \Phi^T Y + 2\Phi^T \Phi \hat{\theta} = 0 \quad (33)$$

$$\frac{\partial^2 J_{MQ}}{\partial \hat{\theta}^2} = 2\Phi^T \Phi > 0 \quad (34)$$

Para qualquer valor do vetor Φ a Equação (34) sempre será satisfeita e o vetor de parâmetros estimados será:

$$\hat{\theta} = [\Phi^T \Phi]^{-1} \Phi^T Y \quad (35)$$

A Equação (35) é conhecida como estimador de mínimos quadrados clássico [20][21]. Nota-se que ela depende do vetor regressor e da saída do sistema. Pelo fato da Equação (35) depender de todos os dados do vetor de regressores e da saída, conclui-se que a estimação só poderá ser feita em bateladas. Esta necessidade, juntamente com as operações de inversão de matriz a cada período de amostragem, tornam o uso do método MQ inviável em sistemas embarcados, já que os dados são medidos e disponibilizados sequencialmente a cada período de amostragem e o custo computacional para realizar a inversão de matrizes é alto. A solução para resolver este problema é a utilização de outra técnica conhecida como Mínimos Quadrados Recursivos (MQR) [20][21].

As equações relacionadas ao método MQ foram apresentadas de forma a situar o leitor em relação aos termos utilizados tanto no método MQ quanto no método MQR e também mostrar a causa da dificuldade de se implementar o MQ em sistemas embarcados. Porém, como a proposta deste trabalho não é a de avaliar o desempenho de um tipo de controlador e sim verificar os recursos utilizados do FPGA para a implementação deste, não será apresentado aqui a dedução passo-a-passo das equações necessárias para a aplicação do MQR bem como deduções minuciosas do método PID Lim *Self Tuning*.

5.6.2 Mínimos quadrados recursivo (MQR)

O MQR tem como finalidade estimar os parâmetros de um sistema de tal maneira que esta estimação não dependa da inversão de matrizes e nem do armazenamento de todos os dados [20][21]. No domínio do tempo discreto a Equação (35) é dada por:

$$\hat{\theta}(k) = [\varphi^T(k)\varphi(k)]^{-1}\varphi(k)^T y(k) = P(k)\varphi(k)^T y(k) \quad (36)$$

Realizando as devidas manipulações que aqui serão omitidas chega-se a forma recursiva de obtenção do vetor de estimação:

$$\hat{\theta}(k) = \hat{\theta}(k-1) + L(k)[y(k) - \varphi(k)^T \hat{\theta}(k-1)] \quad (37)$$

Onde:

$$L(k) = \frac{P(k-1)\varphi(k)}{1 + \varphi^T(k)P(k-1)\varphi(k)} \quad (38)$$

$$P(k) = (I - L(k)\varphi^T(k))P(k-1) \quad (39)$$

O vetor L é conhecido como ganho do estimador e P é uma matriz não-singular que relaciona a variância entre parâmetros estimados, sendo denominada de Matriz de Covariância.

As seguintes etapas poderão ser seguidas para a implementação do algoritmo MQR para sistemas SISO:

1. Definição da Matriz de Covariância inicial;
2. Registrar a ação de controle e a saída do sistema em um instante k ;
3. Obter o ganho do estimador L ;
4. Obter o vetor de parâmetros estimados $\hat{\theta}$;
5. Atualizar a matriz de covariância P ;
6. Retornar ao passo 2 para executar a próxima iteração.

5.6.3 PID LIM *self tuning*

Após a obtenção dos parâmetros do sistema a cada período de amostragem, pode-se efetuar a sintonia de um controlador projetado para atuar em um sistema de controle *Self-Tuning*. Uma opção neste caso é o projeto de um PID Lim *Self Tuning*. A Equação (40) representa um modelo de segunda ordem de um sistema, que pode ser utilizado no projeto de um controlador PID via o método de Lim [22].

$$\frac{y(k)}{u(k)} = \frac{(b_0 + b_1 z^{-1})z^{-1}}{1 + a_1 z^{-1} + a_2 z^{-2}} = \frac{\bar{b}_0 z^{-1}}{1 + a_1 z^{-1} + a_2 z^{-2}} \quad (40)$$

Para a aplicação em um sistema de controle adaptativo a lei de controle deverá deixar de ser estática (Equação (40)) e passará a ser calculada a cada período de amostragem com base nos parâmetros estimados. Assim a Equação (40) é reescrita como:

$$\frac{y(k)}{u(k)} = \frac{(\widehat{b}_0 + \widehat{b}_1 z^{-1})z^{-1}}{1 + \widehat{a}_1 z^{-1} + \widehat{a}_2 z^{-2}} = \frac{\bar{b}_0 z^{-1}}{1 + \widehat{a}_1 z^{-1} + \widehat{a}_2 z^{-2}} \quad (41)$$

Reescrevendo o incremento do sinal de controle do PID *Self Tuning* em função dos parâmetros estimados, têm-se a Equação (42)[10][20].

$$\Delta u(k) = \frac{(1 - z_d)}{(\widehat{b}_0 + \widehat{b}_1)} [e(k) + \widehat{a}_1 e(k-1) + \widehat{a}_2 e(k-2)] \quad (42)$$

Onde z_d é um polo real discreto que deve ser selecionado de forma a proporcionar o comportamento desejado do sistema em malha fechada. Uma maneira segura de selecionar z_d é considerar sua relação com parâmetros de um modelo contínuo. O polo de um sistema contínuo localizado sobre o eixo real no semi-plano esquerdo tem seu equivalente em um sistema amostrado dado por [10]:

$$|z| = e^{-\alpha T_s} \quad (43)$$

Onde:

$$\alpha = \frac{1}{\tau} \quad (44)$$

Para um comportamento desejado em malha fechada considera-se o τ de malha fechada τ_{MF} . Assim têm-se:

$$z_d = e^{-\frac{T_s}{\tau_{MF}}} \quad (45)$$

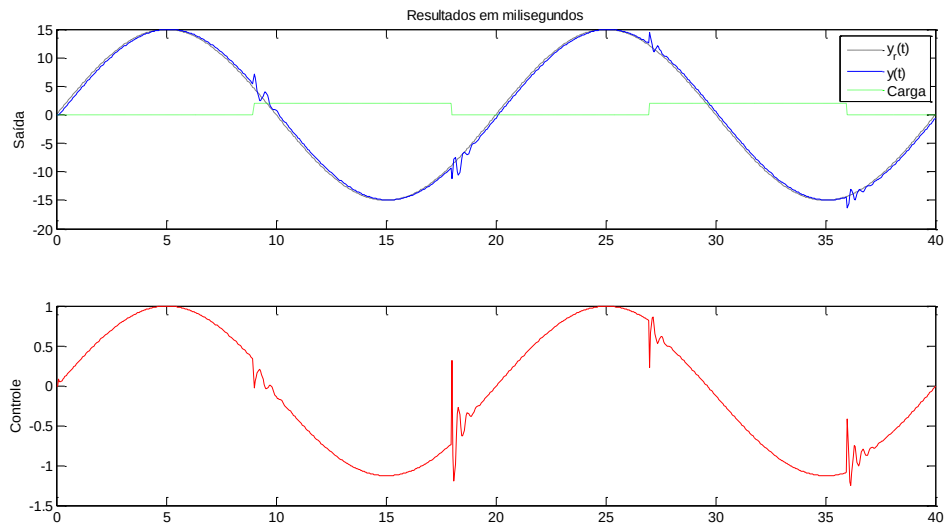
A lei de controle do PID digital incremental ou recursiva é dada por [10][20]:

$$u(k) = u(k - 1) + \Delta u(k) \quad (46)$$

5.6.4 Simulação do controlador *indirect self tuning*

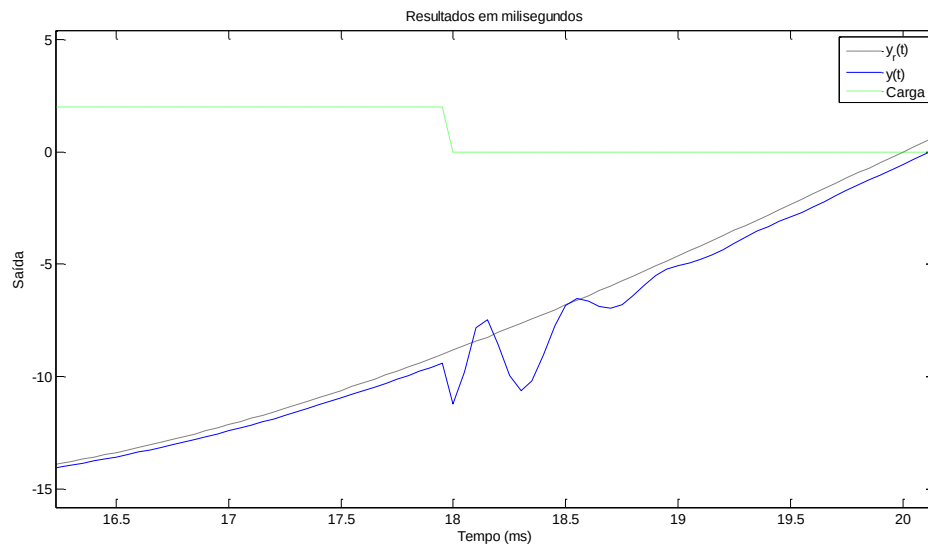
Com auxílio do *software* MATLAB[®] foi efetuado a simulação do método de controle *Indirect Self-Tuning* atuando sobre a planta dada pela Equação (19). Os parâmetros de entrada do código elaborado para a simulação são: Indutância e capacitância do filtro LC, resistência da carga, frequência de amostragem, o valor do *Link DC* e a constante de tempo da planta em malha fechada desejada. Pode-se observar na Figura 42 e 43 a ação do controlador atuando para rastrear o sinal de referência. É aplicado um degrau como um distúrbio a ser controlado.

Figura 42 – Rastreamento do sinal de referência senoidal



Fonte: produção do próprio autor

Figura 43 – Ação do controlador mediante ao distúrbio



Fonte: produção do próprio autor

O Apêndice D apresenta o código utilizado para realizar a simulação no *software* MATLAB[®]. Como o *software* MATLAB[®] resolve cálculos com matrizes, não houve a necessidade de gerar as equações a partir das matrizes, da forma que será feito para escrever o programa em VHDL no próximo tópico.

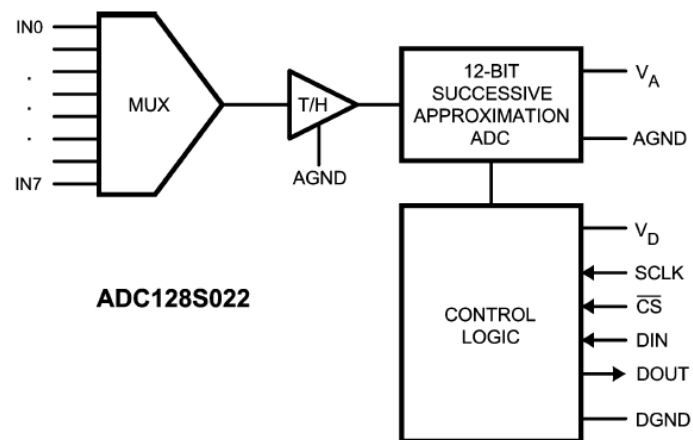
6 IMPLEMENTAÇÃO DOS SISTEMAS DE CONTROLE

Será abordado nesta seção a comunicação serial entre o conversor A/D e o FPGA do kit DE0 – Nano. Apresentar-se-á o método utilizado para quantizar os dados dos códigos em VHDL. Também será apresentada a implementação do controlador PID para os conversores monofásico e trifásico bem como o controlador *Indirect Self Tuning* para o conversor monofásico.

6.1 CONVERSOR A/D DO KIT DE0-NANO

A plataforma de desenvolvimento DE0 – Nano, conta com um conversor A/D de 12 bits com 8 canais. A Figura 44 apresenta um diagrama de blocos do conversor A/D utilizado e a Tabela 9 mostra uma descrição resumida de alguns dos pinos do conversor.

Figura 44 – Diagrama de blocos do conversor A/D do kit DE0 - Nano



Fonte: [19]

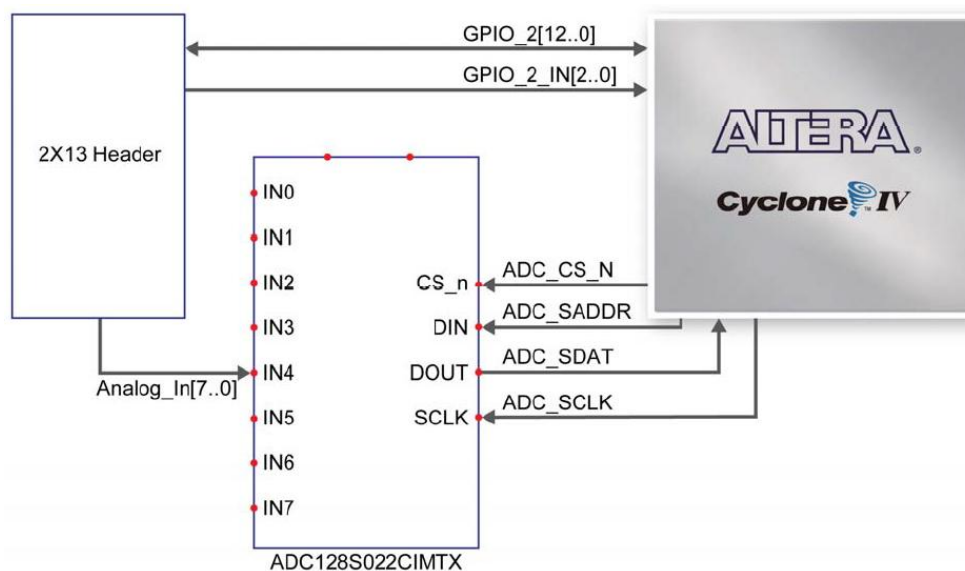
Tabela 9 – Descrição dos principais pinos do conversor A/D do kit DE0 - Nano

Símbolo do Pino	Descrição
SCLK	Entrada do sinal de clock. O desempenho do conversor A/D é garantido para um clock com frequências que estão entre 0,8MHz e 3,2MHz.
DOUT	Saída digital de dados. Os bits referentes ao valor da amostra são colocados neste pino na borda de descida do SCLK.
DIN	Entrada digital de dados. O registrador de controle do conversor A/D é configurado através deste pino na borda de subida do pino SCLK.
$\overline{CS}$	Chip select. Na borda de descida de $\overline{CS}$, um processo de conversão é iniciado. Nível alto neste pino implica no bloqueio do chip.

Fonte: adaptado de [19]

Neste trabalho, foi desenvolvido um código em VHDL, para efetuar a comunicação entre o FPGA e o conversor A/D. Através do pino DIN é possível configurar o canal onde será efetuada a amostragem do sinal analógico desejado [19]. É necessário informar ao conversor A/D o canal que se deseja, a cada amostragem. Mais a frente serão apresentados maiores detalhes sobre a configuração do canal. Após a configuração do canal, a amostragem é realizada e os doze bits referentes ao valor do sinal analógico, no instante da aquisição, são disponibilizados de forma serial no pino DOUT do conversor A/D [19], que está conectado ao pino PIN_A9 do FPGA. O valor da amostra é disponibilizado no pino DOUT, do bit mais significativo (MSB - *Most Significant Bit*) para o menos significativo (LSB - *Least Significant Bit*). Na Figura 45 e na Tabela 10 pode-se verificar a conexão entre o FPGA e o conversor A/D bem como a identificação dos pinos do FPGA.

Figura 45 – Conexão entre o conversor A/D, o FPGA e o conector



Fonte: [7]

Tabela 10 – Identificação dos pinos do FPGA que estão conectados ao conversor A/D

Signal Name	FPGA Pin No.	Description	I/O Standard
ADC_CS_N	PIN_A10	Chip select	3.3V
ADC_SADDR	PIN_B10	Digital data input	3.3V
ADC_SDAT	PIN_A9	Digital data output	3.3V
ADC_SCLK	PIN_B14	Digital clock input	3.3V

Fonte: [7]

Um *frame* serial, ou seja, um valor em binário referente ao processo de amostragem é obtido colocando o pino CS_n em nível baixo e fornecendo 16 bordas de subida no pino SCLK do conversor A/D [19]. A conversão é encerrada após uma borda de subida no pino CS_n. O sinal de clock no pino SCLK deve ter uma frequência entre 0,8MHz e 3,2MHz para garantir o desempenho do conversor A/D [19]. O KIT DE0 – Nano disponibiliza um sinal de clock com frequência de 50MHz. Atribuindo este *clock* a um divisor de frequência, pode-se obter um sinal com as características necessárias para a utilização do conversor A/D. Neste trabalho, optou-se por dividir 50MHz por 10. O sinal de 5MHz resultante é atribuído à uma lógica condicional que proporciona um sinal com uma razão cíclica de 50% e uma frequência de 2,5MHz.

6.1.1 Descrição do código em vhdl para a comunicação serial

O código escrito em VHDL para efetuar a comunicação do FPGA com o conversor A/D é apresentado na Figura 46.

Figura 46 – Rotinas escritas em VHDL para comunicação do FPGA com o conversor A/D

```

276     rotina2: process (DOUT, clk)
277
278     begin
279
280     if aux3 = '1' then--aux3 vai p/ nivel alto no momento da amostragem.
281
282         confAD <= '0';--Utilizado na configuracao do canal 0 do ADC.
283
284         if (clk'event and clk = '1')then
285
286             if Ten = 1 then--Condicao p/ carregamento das
287                 --condicoes iniciais p/ uso do ADC.
288                 CS <= '1';--Chip Select (Desabilita o Conversor A/D).
289                 aux10 <= '1';--Atua simultaneamente ao clock do ADC.
290                 --Condicao inicial p/ uso do ADC.
291                 SCLK <= '1';--Clock do Conversor A/D (Coloca nivel alto
292                 --na saida conectada ao pino SCLK do ADC).
293                 Ten:=0;--Desabilita a logica que promove o carregamento
294                 --das condicoes iniciais p/ uso do ADC.
295             end if;
296
297             K:=K+1;--Contador utilizado no Divisor de Frequencia p/
298                 --obter o clock do ADC.
299
300             if K=N then--Divisor de frequencia (50e6/10 = 5e6)
301
302                 if hab_ADC = 1 then--Condicao p/ que o ADC seja habilitado.
303                     CS <= '0';--Chip Select (Habilita o Conversor A/D).
304                 end if;
305
306                 if a5 /= 1 then-- Gera a onda quadrada p/ enviar ao ADC.
307
308                     a5:= 1;
309                     SCLK <= '1';--Clock do Conversor A/D (Coloca nivel alto na
310                     --saida conectada ao pino SCLK do ADC).
311                     aux10 <= '1';--Atua simultaneamente ao clock do ADC. Habilita
312                     --a rotina que obtem os bits da conversao.
313
314                 else a5:= 0;
315
316                     SCLK <= '0';--Clock do Conversor A/D (Coloca nivel baixo na
317                     --saida conectada ao pino SCLK do ADC).
318                     aux10 <= '0';--Atua simultaneamente ao clock do ADC. Habilita
319                     --a rotina que obtem os bits da conversao.
320

```

```

321         contADC := contADC + 1;
322         if contADC = 17 then --Permite que o ADC seja desabilitado
323             --apos a obtencao do frame.
324             CS <= '1';--Chip Select (Desabilita o Conversor A/D).
325             hab_ADC := 0;--Bloqueia a logica que permite que o ADC
326                 --seja habilitado.
327         end if;
328     end if;
329     K:=0;--Estabelece condicao inicial p/ a proxima amostragem
330     end if;
331
332
333     end if;
334
335 else
336
337     CS <= '1';--Chip Select (Desabilita o Conversor A/D).
338     Ten:=1;--Habilita a logica que promove o carregamento
339         --das condicoes iniciais p/ uso do ADC.
340     K:=0;--Estabelece condicao inicial p/ a proxima amostragem
341     a5:= 1;--Estabelece condicao inicial p/ a proxima amostragem
342     contADC := 0;--Estabelece condicao inicial p/ a proxima amostragem
343     hab_ADC := 1;--Estabelece condicao inicial p/ a proxima amostragem
344
345 end if;
346 end process rotina2;
347
348 rotina3: process (DOUT,aux10)
349
350 begin
351
352 if aux3 = '1' then--sinal eh levado a nivel alto no momento da amostragem.
353
354     if rising_edge (aux10) then--Executado na borda se subida de aux10.
355
356         if (cont10 >= 5)and(cont10 < 17) then--Permite a obtencao dos bits da
357             --conversao que sao apresentados no pino
358             --DOUT apos a quinta borda de descida do
359             --SCLK. O carregamento eh realizado na
360             --quinta borda de subida de SCLK.
361
362             if cont10 = 5 then--Permite salvar o valor do sinal de referencia
363                 --no instante da amostragem p/ comparacao posterior.
364                 senoComp := senoidel(count1);--Carrega o valor da referencia na
365                     -- variavel "senoComp".
366             end if;
367
368             if DOUT = '1' then--Permite que "ValorConv" seja incrementada
369                 --sempre que um nivel alto eh apresentado em DOUT.
370                 ValorConv:= ValorConv + peso;--Atribui a "ValorConv" o valor passado
371                     --de "ValorConv" somado ao valor de peso.
372                 peso:= peso/2;--Divide a variavel "peso" a cada borda de subida de aux10.
373             else
374                 peso:= peso/2;--Divide a variavel "peso" a cada borda de subida de aux10.
375             end if;
376         end if;
377
378         if cont10 = 2 then--Condicao p/ inicializar variaveis.
379             ValorConv:= 0;--Estabelece condicao inicial p/ a proxima amostragem.
380             peso:= 2048;--Estabelece condicao inicial p/ a proxima amostragem.
381
382         end if;
383
384     end if;

```

```

385
386     if falling_edge (aux10) then--Executado na borda de descida de aux10.
387
388         cont10:= cont10 + 1;--contador que controla a comunicacao serial entre
389             --ADC e FPGA.
390
391         if cont10 = 17 then--Permite o carregamento da variavel
392             --"ValorAtual" apos a obtencao do frame.
393             ValorAtual:=(ValorConv - 1900);--Elimina o offset adquirido no
394                 --circuito de condicionamento.
395             LiberaCalculo <= '1';--Estabelece um nivel alto no sinal "LiberaCalculo".
396
397         else
398             ValorAtual:= ValorAtual;--Enquanto a conversao nao eh finalizada,
399                 --ValorAtual eh carregado c/ o proprio valor.
400         end if;
401
402
403     end if;
404
405 else
406     LiberaCalculo <= '0';--Estabelece um nivel baixo no sinal "LiberaCalculo".
407     cont10:=0; -- Para uma atualizacao colocar zero.
408
409 end if;
410 end process rotina3;
411

```

Fonte: produção do próprio autor

Este código que conta com duas rotinas é parte dos programas apresentados nos Apêndices A, B e C. Tanto a rotina 3 quanto a rotina 4 só são executadas quando o sinal “aux3” está em nível alto. Isto ocorre no instante em que o valor da portadora é igual a zero e a condição que define a frequência de amostragem é satisfeita. No caso dos códigos apresentados neste trabalho, a frequência das portadoras é de $40kHz$ e a frequência de amostragem depende da variável “contTS”, que é incrementada sempre que o valor da portadora positiva é igual a zero. A variável “contTS” é comparada a um valor pré-estabelecido, que definirá a frequência de amostragem (F_s). Neste caso, F_s depende da frequência da portadora, que poderá ser dividida por números inteiros. A Figura 47 apresenta uma situação onde a frequência de amostragem foi definida em $20kHz$. Idealmente, o triângulo vermelho indica o momento da aquisição da amostra, porém na prática a amostragem ocorre após a configuração do canal analógico.

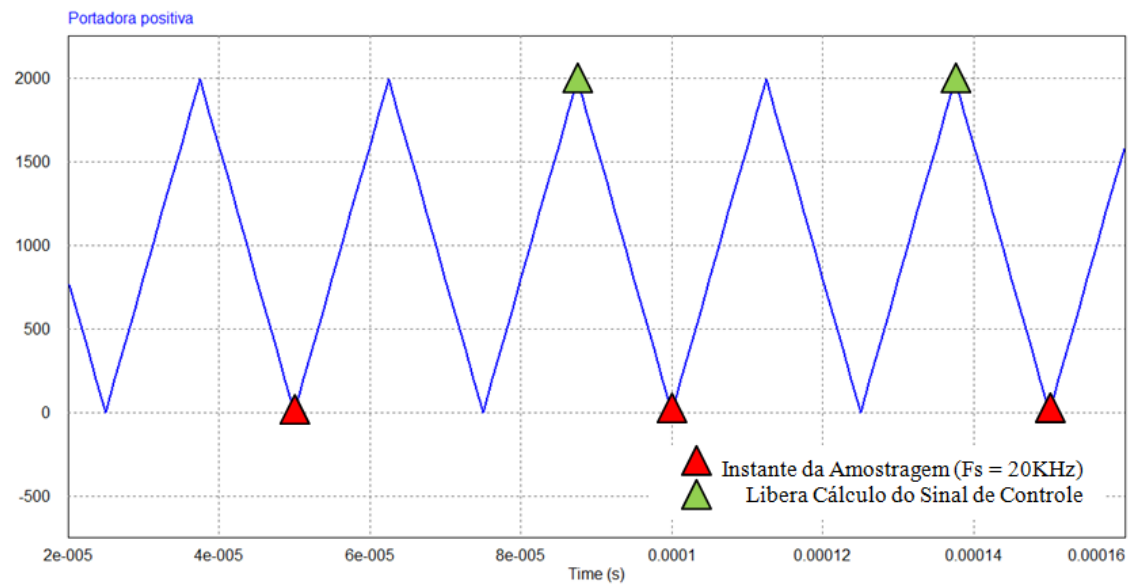
O canal analógico é definido depois de quatro bordas de subida no pino SCLK, o que gera um atraso na aquisição da amostra. O tempo de atraso depende da frequência do sinal que é enviado ao pino SCLK. Como mencionado anteriormente, a frequência do clock do conversor A/D foi definida em $2,5MHz$. Logo, o tempo de atraso é igual $1,6\mu s$ ($4 \times 1/2,5 \times 10^6$). Vale ressaltar que um *frame* é constituído após um atraso de $6,4\mu s$ ($16 \times 1/2,5 \times 10^6$), já que são necessários dezesseis bordas de subida no pino SCLK para colocar o último *bit* da conversão no pino DOUT.

Com o sinal “aux3” em nível alto as rotinas 2 e 3 poderão ser executadas. As variáveis “K” e “a5” são responsáveis pelo condicionamento do clock do KIT DE0 – Nano, gerando o clock do conversor A/D. O sinal “aux10” atua simultaneamente ao sinal de clock enviado ao conversor A/D. O sinal “aux10” é colocado em nível alto antes do pino CS_n ser colocado em nível baixo. Assim, no momento que o pino CS_n é levado a nível baixo, a primeira transição que ocorre no sinal “aux10” é obrigatoriamente, de nível alto para nível baixo (borda de descida). Como a variável “a5” é iniciada com um valor igual a 1, a execução do código leva ao resultado esperado, ou seja, primeira transição de “aux10” é uma borda descida. O código identifica a borda de descida de “aux10” e incrementa a variável “cont10”. Após cinco bordas de descida do sinal “aux10” que equivalem a quatro bordas de subida do mesmo sinal, o canal do conversor A/D já terá sido configurado e o *bit* mais significativo (MSB) referente a amostra, estará disponível no pino DOUT. O código também identifica a borda de subida de “aux10”. Assim na primeira transição *down-up* após as cinco bordas de descida, o *bit* que está no pino DOUT permite, caso o seu valor seja igual a 1, que a variável “ValorConv” receba o valor da variável “peso”, que inicialmente tem um valor igual a 2048, somado ao próprio valor da variável “ValorConv” que é inicializada com o valor zero.

A cada *bit* que é disponibilizado no pino DOUT, a variável “peso” é dividida por 2. Este processo de divisão permite que um peso em decimal seja atribuído ao *bit*.

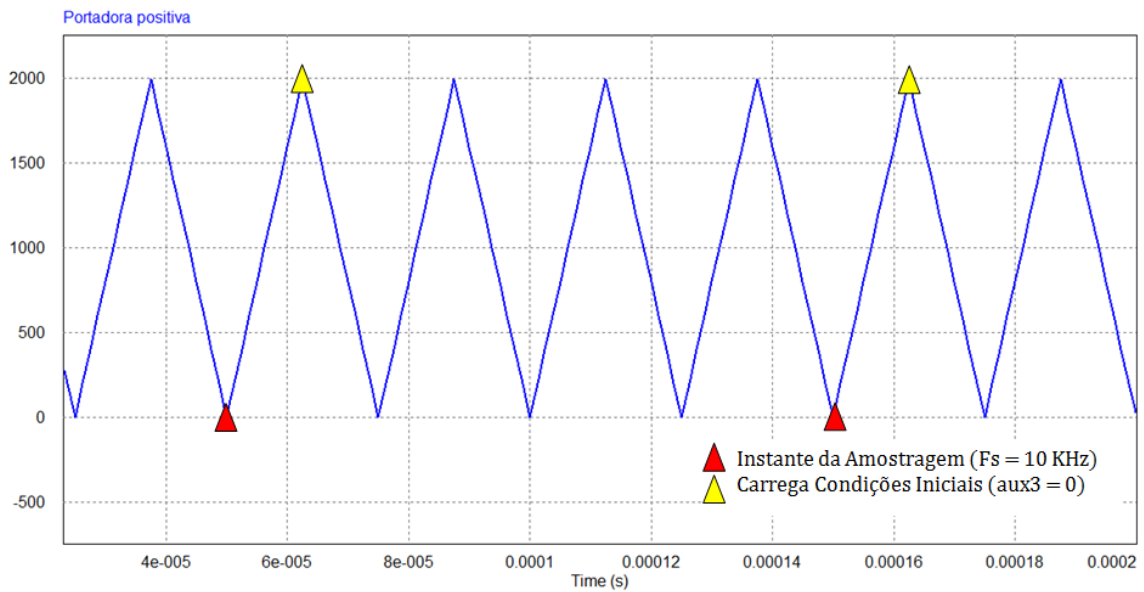
Após 17 transições *up-down* no sinal “aux10” o valor de “ValorConv” é subtraído de um valor em decimal (1900), para retirar o *offset* incorporado no circuito de condicionamento (Figura 31(2)) e atribuído a variável “ValorAtual”. Neste momento, “ValorAtual” possui o valor em decimal e com o ganho do conversor A/D, da amostra do sinal analógico. Após esta atualização de “ValorAtual” o sinal “LiberaCalculo” é colocado em nível alto possibilitando a realização dos cálculos inerentes ao tipo de controlador que foi implementado no código em questão. A variável “cont10” continuará sendo incrementada até que o sinal “aux3” seja levado a nível baixo, porém devido a lógica condicional implementada, os valores de “ValorConv” e “ValorAtual” não são alterados até que ocorra a próxima amostragem. O sinal “aux3” é colocado em nível baixo quando um valor pré-definido da portadora positiva é alcançado. O triângulo verde da Figura 47 indica este momento no código que implementa o controlador *Indirect Self Tuning*. A Figura 48 mostra o caso para código apresentado no Apêndice A, onde F_s é de 10kHz.

Figura 47 – Indicação do instante da amostragem (controlador *Indirect Self Tuning*)



Fonte: produção do próprio autor

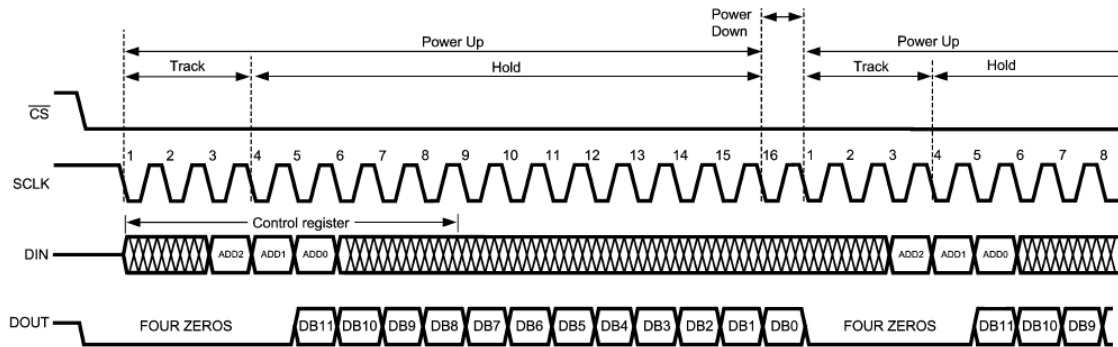
Figura 48 – Indicação do instante da amostragem (controlador PID digital)



Fonte: produção do próprio autor

A Figura 49 apresenta o diagrama de temporização dos sinais envolvidos na comunicação com o conversor A/D.

Figura 49 – Diagrama de temporização (conversor A/D)



Fonte: [19]

Neste projeto, a escolha do canal analógico é realizada através do valor que é atribuído ao pino de saída chamado de “confAD” (nome dado ao pino PIN_B10; Ver Figura 45 e Tabela 10). Colocando o valor zero em “confAD”, após quatro bordas de subida em SCLK o canal IN0 é ativado. De acordo com a Tabela 11, os bits 0, 1, 2, 6 e 7 do registrador responsável pela escolha do canal analógico não são relevantes, podendo ter qualquer nível lógico. A Tabela 12 apresenta os valores que devem ser atribuídos aos bits 3, 4 e 5 para realizar a escolha de um determinado canal. Como o canal IN0 é escolhido atribuindo nível lógico baixo aos três bits relevantes do registrador, basta deixar “confAD” sempre em zero.

Tabela 11 – Bits do Registrador de Controle (Conversor A/D)

Bit 7 (MSB)	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DONTC	DONTC	ADD2	ADD1	ADD0	DONTC	DONTC	DONTC

Fonte: [19]

Tabela 12 – Seleção do Canal de Entrada

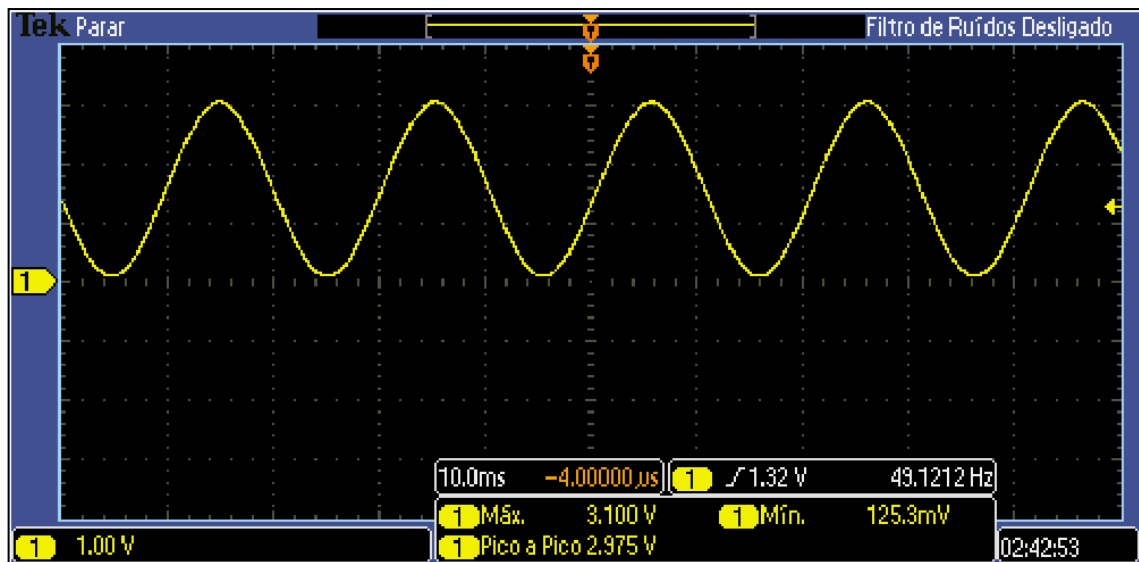
ADD2	ADD1	ADD0	Input Channel
0	0	0	IN0 (Default)
0	0	1	IN1
0	1	0	IN2
0	1	1	IN3
1	0	0	IN4
1	0	1	IN5
1	1	0	IN6
1	1	1	IN7

Fonte: [19]

6.1.2 Validação do código para a comunicação serial

A validação do código desenvolvido para a comunicação entre o FPGA e o conversor A/D foi realizada com a utilização de um gerador de sinais e um osciloscópio capaz de efetuar a leitura do nível lógico de uma word (16 bits) presente em 16 pinos de algum dispositivo. Criando vetores de sinais ao invés de variáveis é possível disponibilizar nos pinos dos conectores GPIOs (*General Purpose Input Output*) do KIT DE0 – Nano os 12 bits referentes ao valor da amostragem. Através do osciloscópio é possível registrar a evolução de cada um dos bits em um intervalo de tempo e gerar um arquivo (.csv) com estes dados. Aplicando o sinal apresentado na Figura 50 no canal IN0, obteve-se o sinal mostrado na Figura 51, que foi extraído dos pinos do conector (GPIOs) do KIT DE0 – Nano.

Figura 50 – Sinal aplicado ao canal IN0



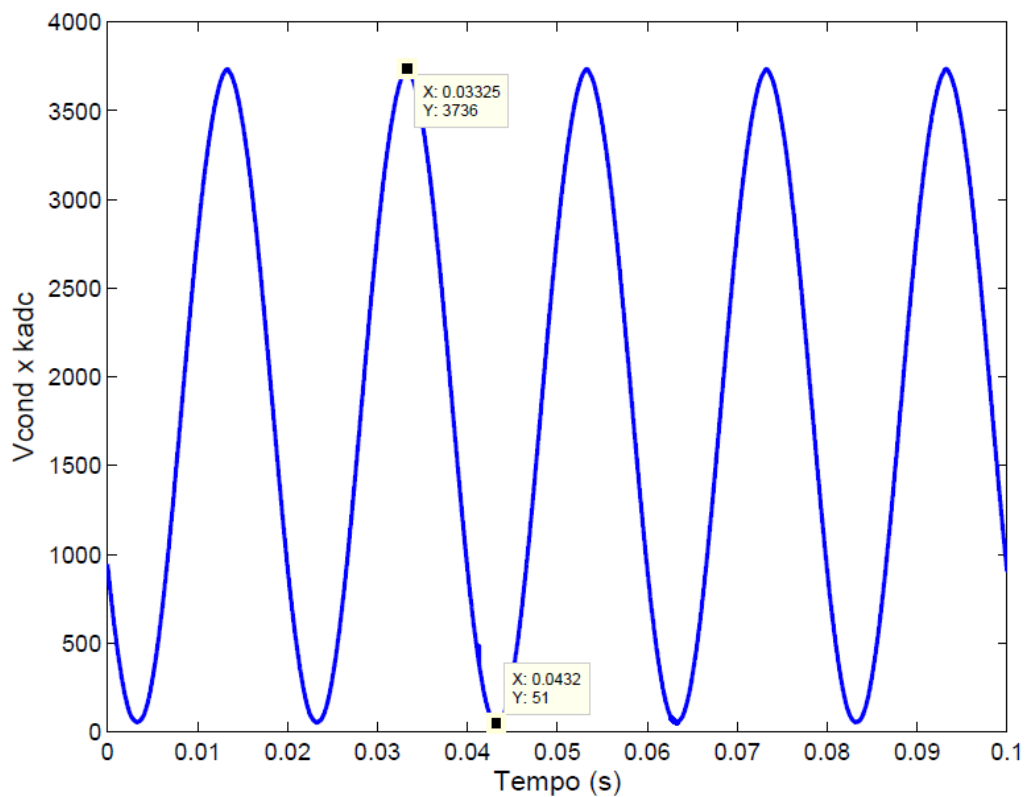
Fonte: produção do próprio autor

Com auxílio do *software* MATLAB[®], os dados obtidos com o osciloscópio foram tratados para gerar a Figura 51. O valor de cada amostra que compõe a curva da Figura 51 possui o ganho do conversor A/D incorporado ao valor real. Nota-se que o valor máximo é de 3736 e o valor mínimo é de 51, que equivalem a 3,01V e 41,09mV respectivamente, conforme a Equação (47).

$$V_{inadc} = \frac{(V_{cond} \times k_{adc}).(3,3)}{(2^{12} - 1)} \quad (47)$$

Onde V_{inadc} é o valor em tensão na entrada do conversor A/D, V_{cond} é o valor em tensão proveniente do circuito de condicionamento e $kadc$ é o ganho do conversor A/D. As variáveis V_{inadc} e V_{cond} possuem sempre o mesmo valor. Entretanto, a Equação (47) foi definida para fornecer o valor em tensão do sinal de entrada do conversor A/D, a partir dos valores obtidos no gráfico da Figura 51. O termo da Equação (47) de valor 3,3 é referente ao valor da tensão de referência colocada no conversor A/D, decorrente da configuração de *hardware* do KIT DE0 – Nano.

Figura 51 – Resultado da conversão A/D



Fonte: produção do próprio autor

O sinal proveniente da conversão apresenta uma atenuação em relação ao sinal de entrada conforme a Tabela 13.

Tabela 13 – Comparação entre o sinal de entrada e o sinal convertido

	Entrada (V)	Saída (V)	Atenuação (V)
Valor Máximo	3,10	3,01	90×10^{-3}
Valor Mínimo	$125,30 \times 10^{-3}$	$41,09 \times 10^{-3}$	$84,21 \times 10^{-3}$

Fonte: produção do próprio autor

6.2 QUANTIZAÇÃO DOS DADOS NO FPGA

Para definir o número de bits, a fim de representar tanto a parte inteira quanto a parte fracionária (utilizando o VHDL 2008), de cada variável ou sinal relacionados ao algoritmo de modulação, conversão A/D e cálculo da lei de controle, adotou-se as estratégias apresentadas a seguir.

6.2.1 Quantização aplicada no código que implementa o PID digital

Em primeiro lugar, foi determinado a quantidade de bits necessários, para representar com precisão a parte fracionária dos coeficientes da Equação (48). Uma forma de se obter o equivalente em binário, da parte fracionária de um número representado na base decimal é mostrada a seguir. No exemplo abaixo, a parte fracionária do número $1,625_D$ representado em decimal é convertida para a base binária.

Figura 52 – Algoritmo para mudança de base (decimal para binário) da parte fracionária

$0,625 \times 2 = 1,25 \Rightarrow$ *toma – se a parte inteira (1) **MSB***
 $0,25 \times 2 = 0,5 \Rightarrow$ *toma – se a parte inteira (0)*
 $0,5 \times 2 = 1 \Rightarrow$ *toma – se a parte inteira (1) **LSB***

Fonte: produção do próprio autor

O equivalente em binário é representado da esquerda (**MSB**) para a direita (**LSB**). Logo:

$$0,625_D = 0,101_B$$

O algoritmo descrito na Figura 52 foi implementado no *software* MATLAB[®] para gerar o número de *bits* necessários para representar de forma precisa os coeficientes da Equação (48). Na Figura 53 é exibido o código escrito no *software* MATLAB[®].

Figura 53– Código desenvolvido no *software* MATLAB® para encontrar o número de bits necessários para representar com precisão em binário a parte fracionária de um número em decimal

```

clear all; close all; clc;

x=0.733;
n=1;
while x~=1
    x = x*2;
    if x >= 1
        k0_b(n) = 1;
    else
        k0_b(n) = 0;
    end
    if x > 1
        x = x-1;
    end
    n = n+1;
end

x=0.267;
n=1;
while x~=1
    x = x*2;
    if x >= 1
        k1_b(n) = 1;
    else
        k1_b(n) = 0;
    end
    if x > 1
        x = x-1;
    end
    n = n+1;
end

x=0.4;
n=1;
while x~=1
    x = x*2;
    if x >= 1
        k2_b(n) = 1;
    else
        k2_b(n) = 0;
    end
    if x > 1
        x = x-1;
    end
    n = n+1;
end

x=0.48;
n=1;
while x~=1
    x = x*2;
    if x >= 1
        k3_b(n) = 1;
    else
        k3_b(n) = 0;
    end
    if x > 1
        x = x-1;
    end
    n = n+1;
end

x=0.3172;
n=1;
while x~=1
    x = x*2;
    if x >= 1
        k4_b(n) = 1;
    else
        k4_b(n) = 0;
    end
    if x > 1
        x = x-1;
    end
    n = n+1;
end

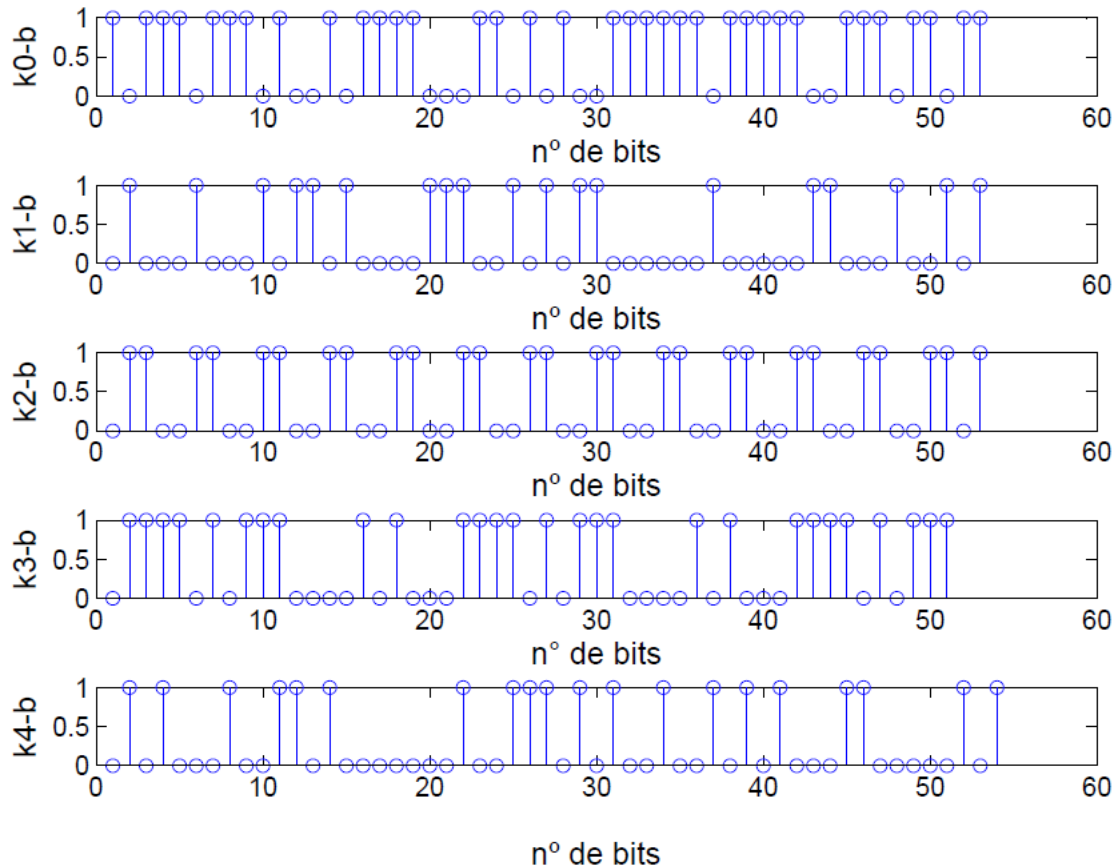
subplot(511),
    stem(k0_b);ylabel('k0-
b');xlabel('nº de bits');
subplot(512),
    stem(k1_b);ylabel('k1-
b');xlabel('nº de bits');
subplot(513),
    stem(k2_b);ylabel('k2-
b');xlabel('nº de bits');
subplot(514),
    stem(k3_b);ylabel('k3-
b');xlabel('nº de bits');
subplot(515),
    stem(k4_b);ylabel('k4-
b');xlabel('nº de bits');

```

Fonte: produção do próprio autor

Com o código apresentado na Figura 53 foi gerado os gráficos da Figura 54. Estes gráficos representam de forma discreta o número de *bits* necessários para representar com precisão a parte fracionária dos coeficientes da Equação (48).

Figura 54 – Representação gráfica dos bits necessários para representar com precisão os coeficientes da Equação (48)



Fonte: produção do próprio autor

Nota-se a partir da Figura 54, que a quantidade de *bits* para representar com precisão a parte fracionária dos coeficientes da Equação (48), fica entre 50 e 54 *bits*. De acordo com [3] não há restrição, com relação ao tamanho das variáveis ou sinais, para o uso de multiplicadores. Porém, como mencionado na seção 2.1.3, quanto maior o número de bits, maior é o tempo gasto para efetuar as multiplicações. Assim, procurou-se atribuir um número de bits que proporcionasse uma aproximação razoável dos valores reais. Como o FPGA utilizado, conta com 66 multiplicadores de 18×18 , optou-se por atribuir para o caso dos coeficientes, 18 *bits*. Entretanto, é interessante checar a precisão das aproximações. Para tal, a Tabela 14 foi gerada no *software* MATLAB® a partir do código apresentado na Figura 55. Esta tabela exibe o peso em decimal dos dezessete primeiros bits à direita da vírgula. Vale ressaltar, que devido ao fato dos coeficientes terem sido definidos como um tipo com sinal, no código em VHDL, 1 *bit* deve ser reservado para indicar se o valor é positivo ou negativo. Restando assim 17 *bits* para representar o valor.

Figura 55 - Código desenvolvido no MATLAB® para gerar a Tabela 12

```
clear all; close all; clc;

for k=1:1:17
y(k)=(1/2^k);
end
```

Fonte: produção do próprio autor

Tabela 14 – Peso em decimal dos bits das posições -1 à -17 com 15 casas de precisão

$1/2^n$	Parte Fracionária
$1/2^1$	0.500000000000000
$1/2^2$	0.250000000000000
$1/2^3$	0.125000000000000
$1/2^4$	0.062500000000000
$1/2^5$	0.031250000000000
$1/2^6$	0.015625000000000
$1/2^7$	0.007812500000000
$1/2^8$	0.003906250000000
$1/2^9$	0.001953125000000
$1/2^{10}$	0.000976562500000
$1/2^{11}$	0.000488281250000
$1/2^{12}$	0.000244140625000
$1/2^{13}$	0.000122070312500
$1/2^{14}$	0.000061035156250
$1/2^{15}$	0.000030517578125
$1/2^{16}$	0.000015258789063
$1/2^{17}$	0.000007629394531

Fonte: produção do próprio autor

A partir da Figura 54 e da Tabela 14 foram gerados os resultados abaixo. De acordo com as regras de arredondamento dadas em [9], verificou-se a precisão das aproximações decorrentes da escolha do número de 17 *bits* para representar os coeficientes da Equação (48).

$$0,733_D = 0,1011101110100101 \mathbf{1}_{LSB} | \mathbf{1}_{MSB} 10001101010011111101111100111011011_B$$

$$0,733_D = 0,10111011101001 \mathbf{1} 00_B = 0,733001708984375_D$$

$$0,267_D = 0,0100010001011010 \mathbf{0}_{LSB} | \mathbf{0}_{MSB} 01110010101100000010000011000100101_B$$

$$0,267_D = 0,0100010001011010 \mathbf{0}_B = 0,266998291015625_D$$

$$0,4_D = 0,0110011001100110 \mathbf{0}_{LSB} | \mathbf{1}_{MSB} 1001100110011001100110011001101_B$$

$$0,4_D = 0,0110011001100110 \mathbf{0}_B = 0,399993896484375_D$$

$$0,48_D = 0,0111101011100001\mathbf{0}_{LSB}|\mathbf{1}_{MSB}000111101011100001010001111010111_B$$

$$0,48_D = 0,01111010111000010_B = 0,479995727539063_D$$

$$0,3172_D = 0,0101000100110100\mathbf{0}_{LSB}|\mathbf{0}_{MSB}000100111010100100101010001100000101_B$$

$$0,3172_D = 0,01010001001101000_B = 0,317199707031250_D$$

Para o sinal que recebe os valores do vetor que representa a moduladora (também utilizado como sinal de referência), chamado no código em VHDL de “refComp_b”, foram atribuídos 12 *bits* para representá-lo. Neste caso são 11 *bits* para representar o valor e mais 1 *bit* de sinal. A amplitude máxima do sinal modulante é de 1900 e todos os valores do vetor são inteiros, como $(2^{11} - 1)$ é igual a 2047, 11 *bits* são suficientes para representá-lo. Para o sinal que recebe o valor proveniente do conversor A/D sem o *offset*, chamado de “valaatual_b”, também foram atribuídos 12 *bits*, já que os valores são inteiros e a magnitude é a mesma de “refComp_b”. De acordo com a Tabela 6, o sinal que recebe o resultado da subtração entre “refComp_b” e “valaatual_b” deve ter 13 *bits* para representá-lo. Logo, o sinal chamado de “ek_b” é quantizado em 13 *bits* (13 *bits* para o valor e mais 1 *bit* para o sinal). Como “ek1_b” e “ek2_b” recebem os valores passados de “ek_b”, também foram atribuídos 13 *bits* para representar estes dois sinais. Para determinar o tamanho do sinal de controle chamado de “uk_b” no código em VHDL, seria necessário avaliar cada operação matemática e levar em conta a Tabela 6. Porém, como o cálculo do sinal de controle é recursivo, o tamanho do sinal aumentaria sempre que “uk_b” fosse calculado, já que “uk_b” depende de valores passados atribuídos à “uk1_b” e “uk2_b”. Na primeira vez que “uk_b” é calculado, o valor obtido é o produto entre o sinal de erro “ek_b” e o coeficiente “k2_b”. Neste momento, de acordo com a Tabela 6 o tamanho de “uk_b” deveria ser de 14 *bits* para a parte inteira e 17 *bits* para a parte fracionária. Como mencionado acima, o tamanho deste sinal aumentaria indefinidamente, entretanto, foi utilizado a função *resize* de [9] para formatar “uk_b”, mantendo constante o tamanho deste sinal. Assim, foi atribuído de forma arbitrária, 18 *bits* para representar a parte inteira, 18 *bits* para representar a parte fracionária de “uk_b” e mais 1 *bit* de sinal, ou seja, um tamanho maior do que o necessário para representar o valor inicial dos primeiros cálculos de “uk_b”. Os sinais “uk1_b” e “uk2_b” também foram representados com 36 *bits* e mais 1 *bit* de sinal, já que eles recebem valores passados de “uk_b”. As Figuras 56 e 57 mostram partes do código em VHDL pertencente ao programa apresentado no Apêndice A. Da linha 45 à linha 58 são realizadas as declarações dos sinais citados acima.

Figura 56 – Código em VHDL para a quantização dos sinais

```

45     signal k0_b : sfixed(0 downto -17);--recebe o coeficiente a1.
46     signal k1_b : sfixed(0 downto -17);--recebe o coeficiente a2.
47     signal k2_b : sfixed(0 downto -17);--recebe o coeficiente b0.
48     signal k3_b : sfixed(0 downto -17);--recebe o coeficiente b1.
49     signal k4_b : sfixed(0 downto -17);--recebe o coeficiente b2.
50
51     signal refComp_b : sfixed(11 downto 0);--recebe o vetor senoide(+/-2047).
52     signal valatual_b : sfixed(11 downto 0);--recebe saida convertida(+/-2047).
53     signal ek_b : sfixed(12 downto 0);--recebe o erro;malha fechada(+/-4095).
54     signal ek1_b : sfixed(12 downto 0);--armazena o erro e(k-1).
55     signal ek2_b : sfixed(12 downto 0);--armazena o erro e(k-2).
56     signal uk_b : sfixed(18 downto -18);--sinal de controle.
57     signal uk1_b : sfixed(18 downto -18);--sinal de controle no instante u(k-1).
58     signal uk2_b : sfixed(18 downto -18);--sinal de controle no instante u(k-2).

```

Fonte: produção do próprio autor

Da linha 416 à linha 431 são feitas as conversões de base (decimal para binário). A função “to_sfixed” apresentada em [9], retorna um valor em binário e quantizado (de acordo com a Figura 56), além de considerar 1 *bit* para sinal. Todos os sinais foram declarados como “sfixed” a fim de facilitar a escrita do código em VHDL, já que a biblioteca [9] não prevê operações matemáticas, como multiplicação, entre tipos diferentes e conversões teriam que ser feitas previamente.

Figura 57 – Código em VHDL que realiza a conversão de bases através da função “to_sfixed”

```

416     k0_b <= to_sfixed(0.733,k0_b);--Coeficiente a1. Atribui a k0_b
417         --um valor ja convertido p/ um tipo
418         --pto fixo c/ sinal.
419     k1_b <= to_sfixed(0.267,k1_b);--Coeficiente a2. Atribui a k1_b um
420         --valor ja convertido p/ um tipo pto
421         --fixo c/ sinal.
422     k2_b <= to_sfixed(0.4,k2_b);--Coeficiente b0. Atribui a k2_b um valor
423         --ja convertido p/ um tipo pto fixo c/ sinal.
424     k3_b <= to_sfixed(0.48,k3_b);--Coeficiente b1. Atribui a k3_b um valor
425         --ja convertido p/ um tipo pto fixo c/ sinal.
426     k4_b <= to_sfixed(0.3172,k4_b);--Coeficiente b2. Atribui a k4_b um valor
427         --ja convertido p/ um tipo pto fixo c/ sinal.
428     refComp_b <= to_sfixed(senoComp,refComp_b);--Atribui a refComp_b o valor da
429         --referencia convertido p/ um
430         --tipo pto fixo c/ sinal.
431     valatual_b <= to_sfixed(ValorAtual,valatual_b);--Atribui a valatual_b o
432         --valor da amostragem convertido
433         --p/ um tipo pto fixo c/ sinal.
434
435
436     if rising_edge (LiberaCalculo) then
437
438         ek2_b <= ek1_b;--Atribui ek1_b a ek2_b p/ que na proxima
439             --amostragem o calculo do sinal de controle
440             --conte com o valor do erro de duas amostragem passada.
441         uk2_b <= uk1_b;--Atribui uk1_b a uk2_b p/ que na proxima
442             --amostragem o calculo do sinal de controle
443             --conte com o valor do sinal de controle obtido
444             --a duas amostragem passada.
445         ek1_b <= ek_b;--Atribui ek_b a ek1_b p/ que na proxima amostragem
446             --o calculo do sinal de controle conte com o valor
447             --anterior do erro.
448         uk1_b <= uk_b;--Atribui uk_b a uk1_b p/ que na proxima amostragem
449             --o calculo do sinal de controle conte com o valor
450             --anterior do sinal de controle.
451
452         ek_b <= refComp_b - valatual_b;--Atribui a "ek_b" a diferenca entre
453             --o valor de referencia e o valor
454             --amostrado.
455
456
457     --*****
458     --Calculo do sinal de controle.
459
460     uk_b <= resize(k0_b*uk1_b + k1_b*uk2_b + k2_b*ek_b - k3_b*ek1_b + k4_b*ek2_b,uk_b);

```

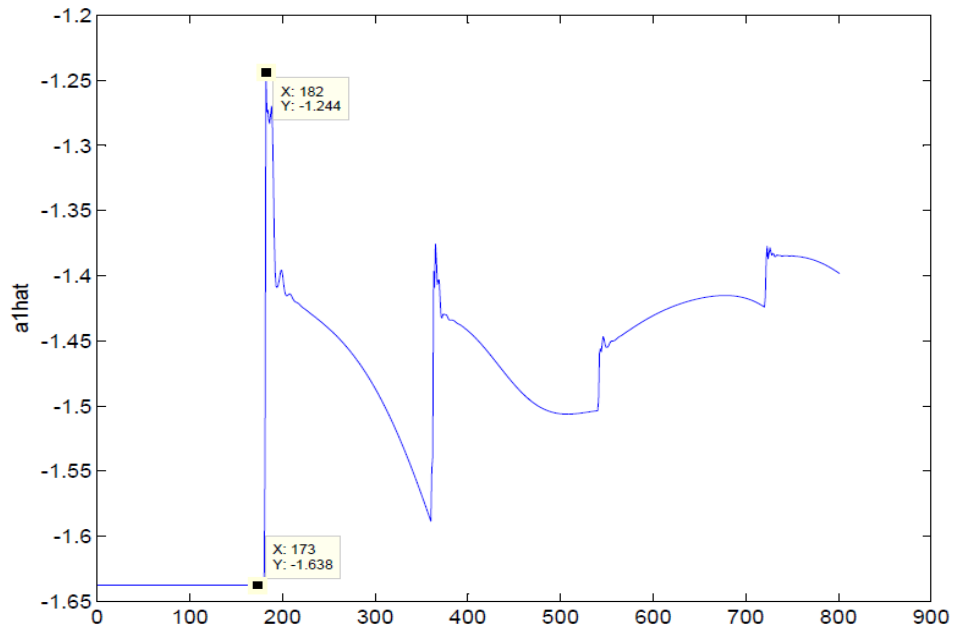
Fonte: produção do próprio autor

6.2.2 Quantização aplicada no código que implementa o *indirect self tuning*

As equações que implementam o MQR, que serão apresentadas na seção 6.5.1, possuem uma quantidade grande de operações matemáticas como produto e soma. Assim, após a elaboração do código em VHDL, optou-se inicialmente por atribuir de forma arbitrária o número de *bits* para representar os sinais e desconsiderar a Tabela 6, a fim de verificar o resultado da compilação do código. Como o relatório do Quartus II® apresentou uma necessidade de elementos lógicos muito maior que o número disponível no FPGA utilizado, não foi realizado um estudo mais minucioso para quantizar os sinais envolvidos no código que implementa o *Indirect Self Tuning*. Entretanto, para que o tamanho dos sinais fosse relativamente coerente, utilizou-se o resultado da simulação feita no *software* MATLAB®, para verificar o menor e o maior valor em módulo, dos parâmetros estimados e do parâmetro

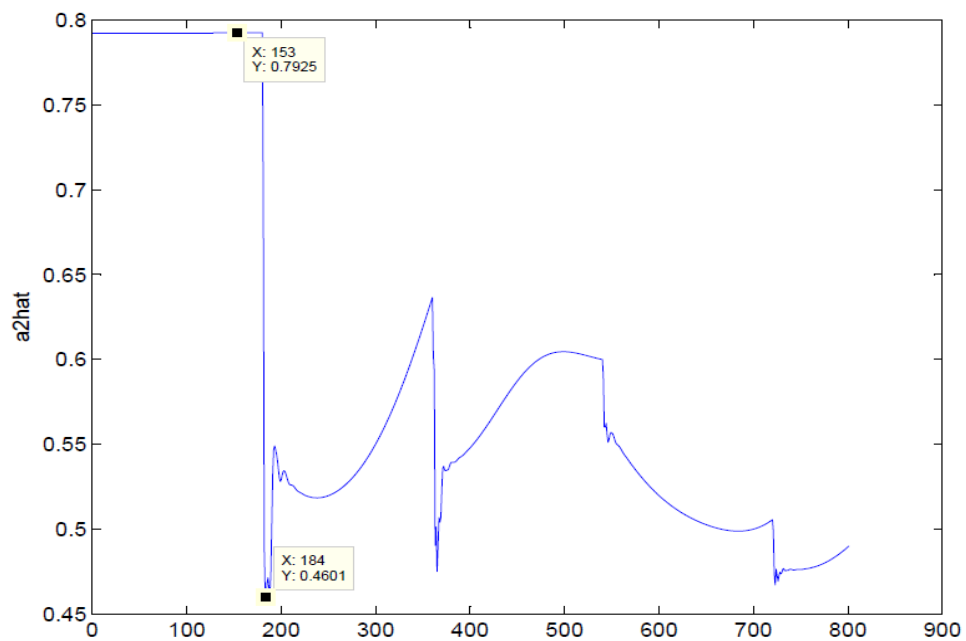
de sintonia “s0”. Das Figura 58 à 68 são apresentados os comportamentos dos parâmetros estimados e do parâmetro de sintonia “s0” durante a simulação.

Figura 58 – Comportamento do parâmetro estimado $a1hat$ na simulação



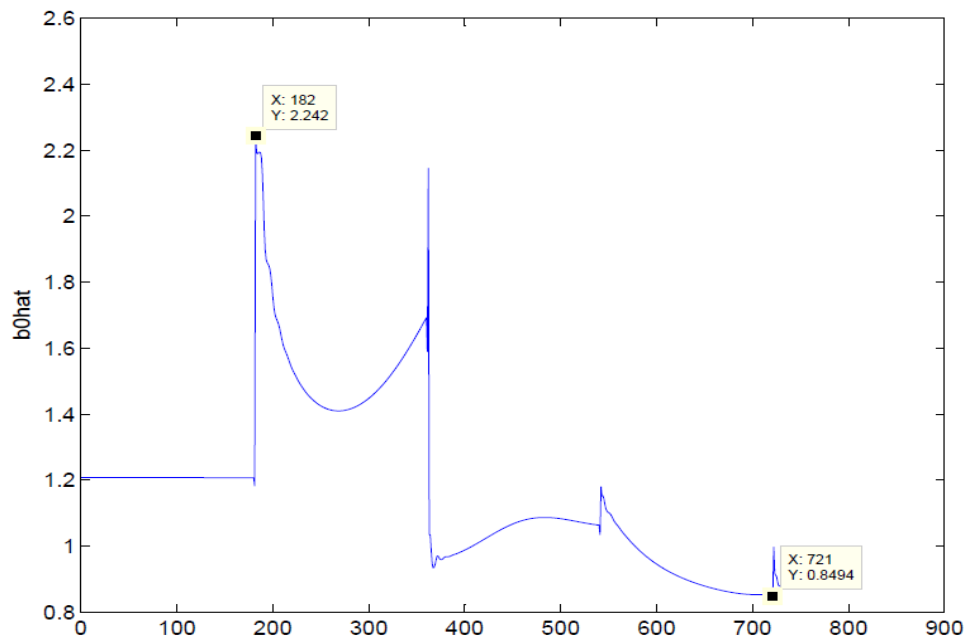
Fonte: produção do próprio autor

Figura 59 – Comportamento do parâmetro estimado $a2hat$ na simulação



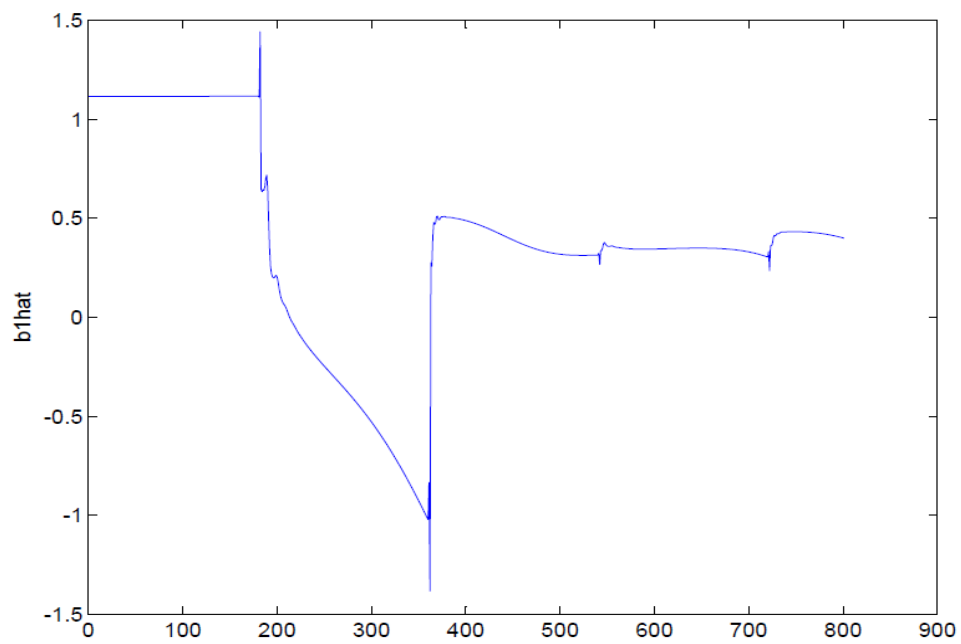
Fonte: produção do próprio autor

Figura 60 – Comportamento do parâmetro estimado $b_0\hat{}$ na simulação



Fonte: produção do próprio autor

Figura 61 – Comportamento do parâmetro estimado $b_1\hat{}$ na simulação



Fonte: produção do próprio autor

Figura 62 – Comportamento do parâmetro estimado $b1hat$ na simulação (discreto)

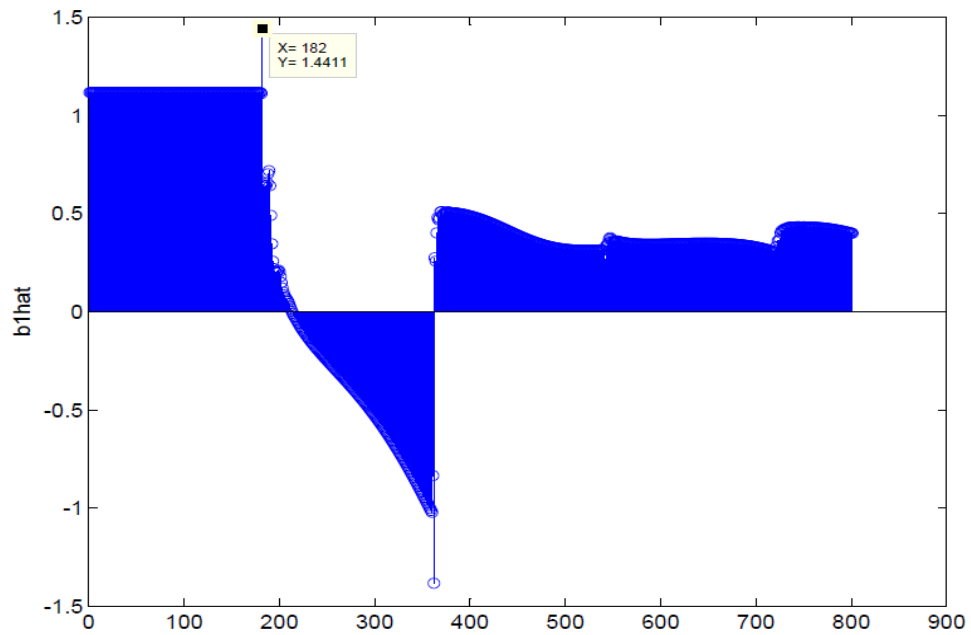


Figura 63 – Comportamento do parâmetro estimado $b1hat$ na simulação (discreto - zoom)

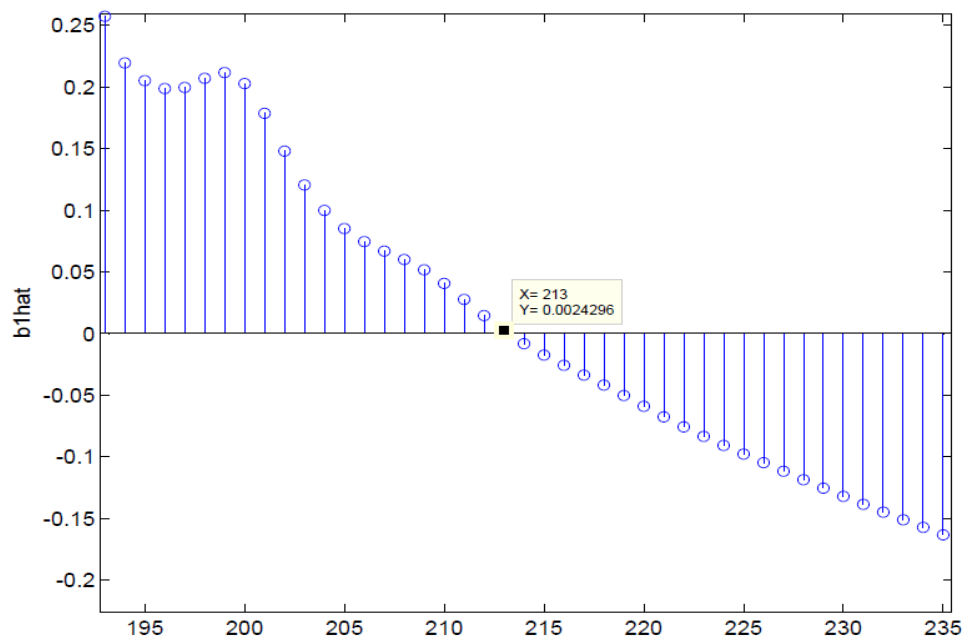


Figura 64 – Comportamento do parâmetro de sintonia s_0 na simulação

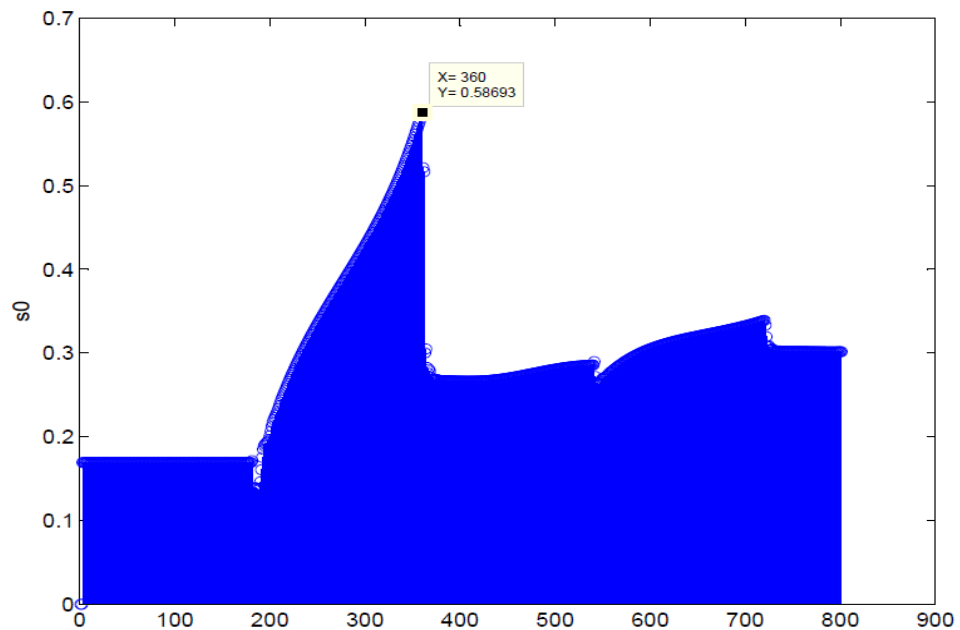
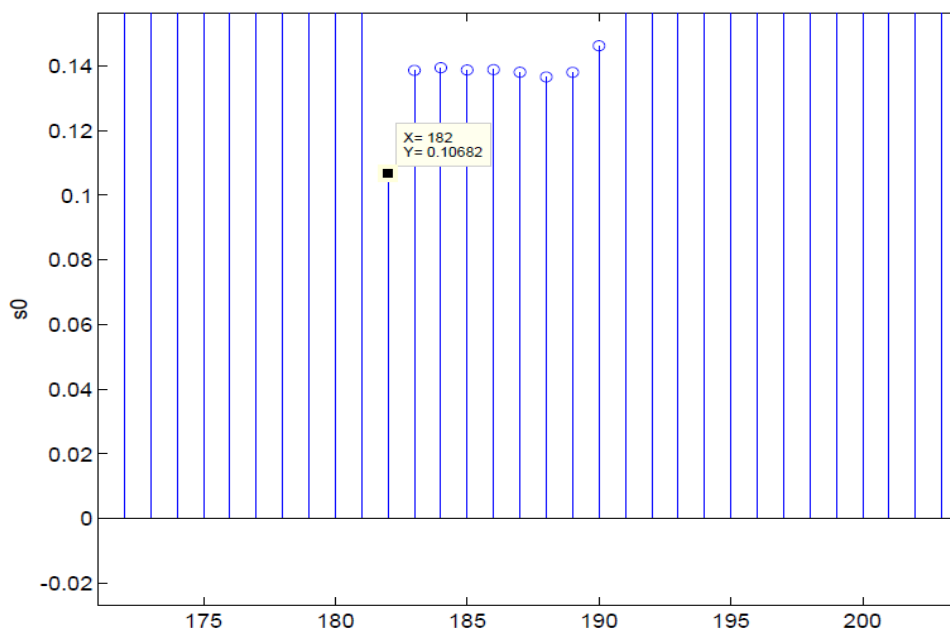
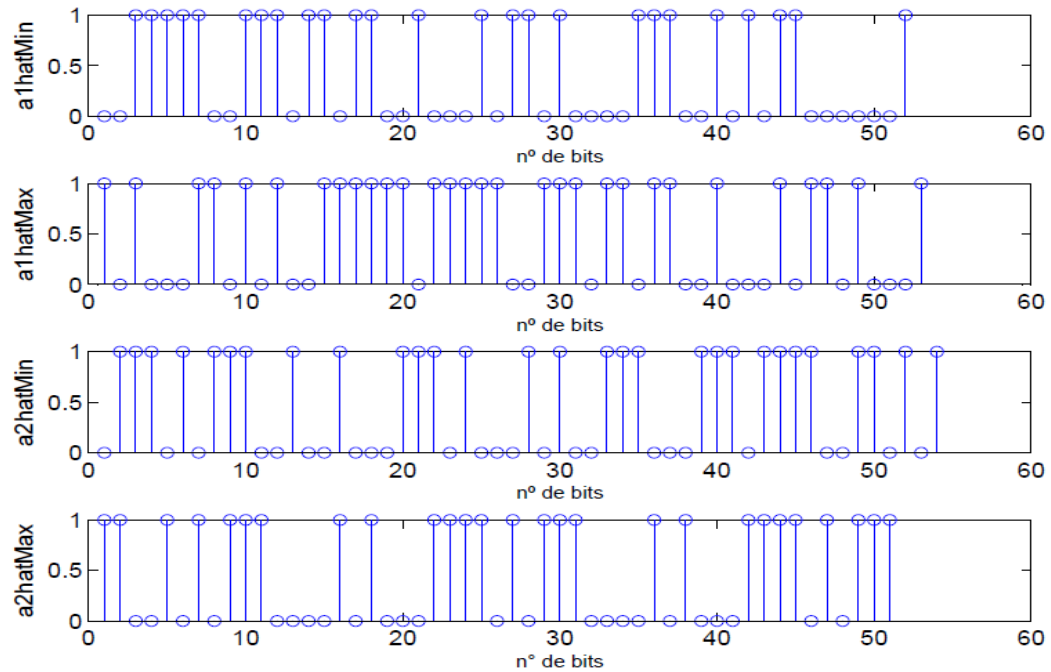


Figura 65 – Comportamento do parâmetro de sintonia s_0 na simulação (zoom)



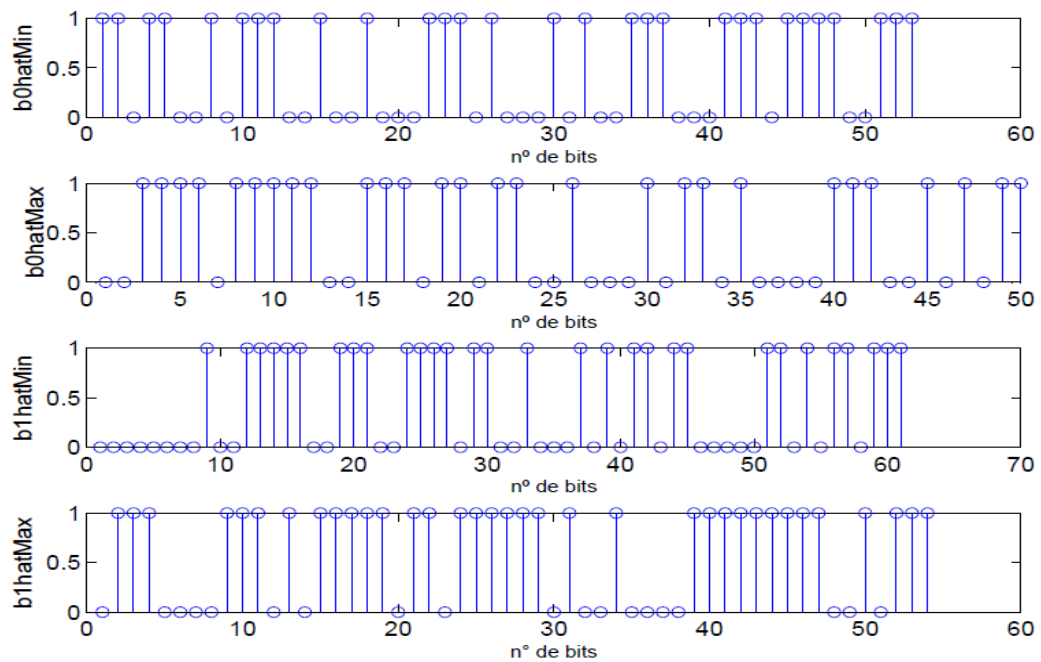
Aplicando os valores obtidos nos gráficos acima no programa apresentado na Figura 53, obtêm-se as Figuras 66, 67 e 68.

Figura 66– Representação do número de bits para representar a parte fracionária dos valores máximos e mínimos dos parâmetros estimados “a”



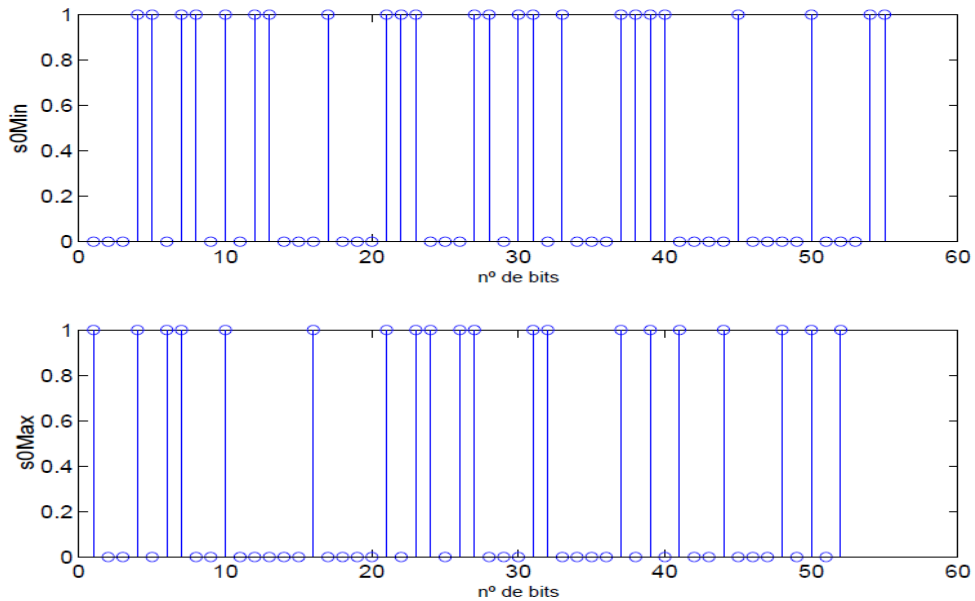
Fonte: produção do próprio autor

Figura 67 – Representação do número de bits para representar a parte fracionária dos valores máximos e mínimos dos parâmetros estimados “b”



Fonte: produção do próprio autor

Figura 68 – Representação do número de bits para representar a parte fracionária dos valores máximos e mínimos do parâmetro de sintonia “s0”



Fonte: produção do próprio autor

Da Figura 58 à 65 é possível observar que a parte inteira dos sinais representados tem um valor pequeno, que pode ser representado em binário por uma quantidade mínima de *bits*. Já o número de *bits* para representar com precisão a parte fracionária, fica entre 50 e 62 *bits*, como pode ser visto nas Figuras 66, 67 e 68. Entretanto, como mencionado em 6.2.1, consegue-se uma precisão boa para a parte fracionária com 17 *bits*. Assim, de forma arbitrária porém coerente, foi determinado 4 *bits* para a parte inteira e 17 *bits* para a parte fracionária dos parâmetros estimados e do parâmetro de sintonia.

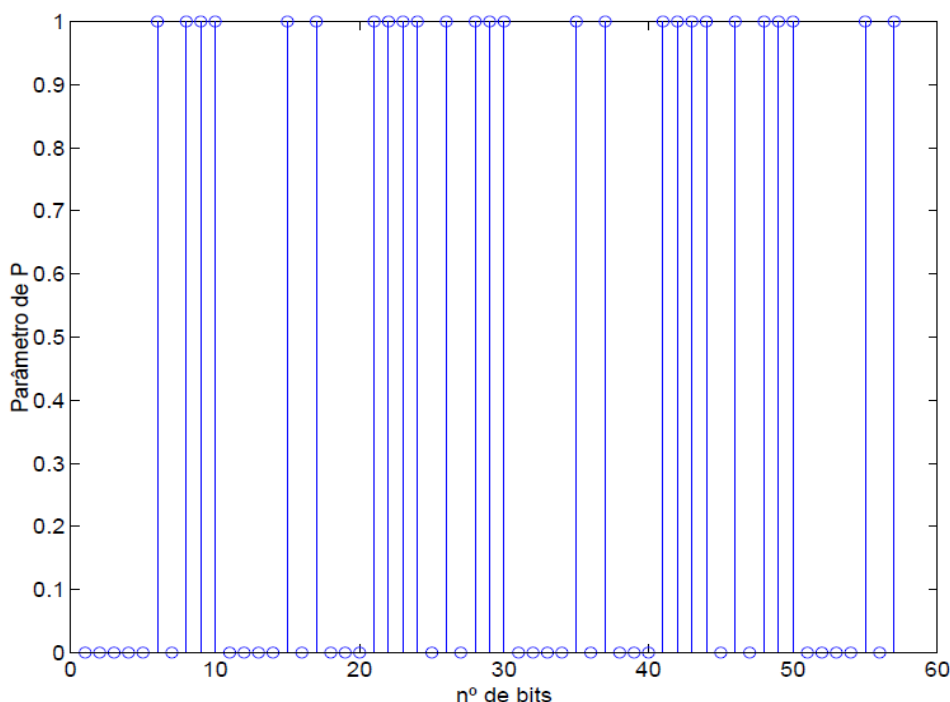
A Figura 69 exibe os valores finais da matriz de covariância P, obtidos com a simulação. Inicialmente esta matriz é uma matriz diagonal de valor igual a 10. A tendência é que estes valores da matriz diminuam ao longo do processo de controle. Assim, é necessário uma quantidade de *bits* para representar o maior valor da parte inteira e uma quantidade de *bits* para representar o menor valor da parte fracionária. Atribuindo o menor valor em módulo, observado na Figura 69, ou seja 0.0225 no código da Figura 53, obtêm-se o gráfico da Figura 70.

Figura 69 – Valores da matriz de covariância obtidos na simulação

$$\begin{bmatrix} 0.0248 & -0.0233 & -0.0381 & 0.0583 \\ -0.0233 & 0.0225 & 0.0392 & -0.0515 \\ -0.0381 & 0.0392 & 0.2841 & -0.2684 \\ 0.0583 & -0.0515 & -0.2684 & 0.3661 \end{bmatrix}$$

Fonte: produção do próprio autor

Figura 70– Representação do número de bits para representar a parte fracionária do menor valor da matriz apresentada na Figura 69



Fonte: produção do próprio autor

Como mencionado anteriormente, consegue-se uma precisão razoável com 17 *bits* para a parte fracionária. Logo, para a matriz de covariância, foi adotado 4 *bits* para representar o valor máximo da parte inteira e 17 *bits* para a parte fracionária. Aos sinais de erro e controle, “ek_b” e “uk_b” respectivamente, bem como os sinais que recebem os valores passados de “ek_b” e “uk_b” como “ek1_b”, “ek2_b”, “uk1_b” e “uk2_b”, foram atribuídos os mesmos números de *bits* utilizados no código apresentado em 6.2.1, já que a magnitude dos sinais são semelhantes. Da mesma forma, aos sinais “refComp_b” e “valaatual_b” mencionados em 6.2.1, foram determinados 12 *bits* para representá-los. Os parâmetros “phi” recebem os valores de “uk_b” e “yk”, logo, o tamanho de “phi” foi definido em 36 *bits* para os parâmetros que recebem “uk1_b” e “uk2_b” e 12 *bits* para os parâmetros que recebem yk1 e yk2, já que yk recebe o valor de “valaatual_b”. Ao ganho do estimador dado pelos parâmetros chamados no programa em VHDL de L1K, L2K, L3K e L4K foram atribuídos 18 *bits* para a parte inteira e 18 *bits* para parte fracionária, já que eles são calculados a partir dos parâmetros “phi”. Nas Figura 71 é exibida a parte do código do Apêndice C, onde é efetuado a quantização dos sinais.

Figura 71 – Código em VHDL que realiza a quantização dos sinais no projeto do controlador *Indirect Self Tuning*

```

33     signal yk1 : sfixed(11 downto 0);--sinal de saida no instante y(k-1).
34     signal yk2 : sfixed(11 downto 0);--sinal de saida no instante y(k-2).
35
36     signal phi1 : sfixed(12 downto 0);--Compoe o vetor de regressores.
37     signal phi2 : sfixed(12 downto 0);--Compoe o vetor de regressores.
38     signal phi3 : sfixed(18 downto -18);--Compoe o vetor de regressores.
39     signal phi4 : sfixed(18 downto -18);--Compoe o vetor de regressores.
40
41     --*****
42     --Compoe o vetor de ganho do estimador.
43
44     signal L1K : sfixed(18 downto -18);
45     signal L2K : sfixed(18 downto -18);
46     signal L3K : sfixed(18 downto -18);
47     signal L4K : sfixed(18 downto -18);
48
49     --*****
50     --Auxiliares p/ calculo do ganho do estimador.
51
52     signal L1Kaux1 : sfixed(18 downto -18);
53     signal L1Kaux2 : sfixed(18 downto -18);
54     signal L1Kaux3 : sfixed(18 downto -18);
55     signal L1Kaux4 : sfixed(18 downto -18);
56     signal L1Kaux5 : sfixed(18 downto -18);
57
58     signal L2Kaux1 : sfixed(18 downto -18);
59     signal L2Kaux2 : sfixed(18 downto -18);
60     signal L2Kaux3 : sfixed(18 downto -18);
61     signal L2Kaux4 : sfixed(18 downto -18);
62     signal L2Kaux5 : sfixed(18 downto -18);
63
64     signal L3Kaux1 : sfixed(18 downto -18);
65     signal L3Kaux2 : sfixed(18 downto -18);
66     signal L3Kaux3 : sfixed(18 downto -18);
67     signal L3Kaux4 : sfixed(18 downto -18);
68     signal L3Kaux5 : sfixed(18 downto -18);
69
70     signal L4Kaux1 : sfixed(18 downto -18);
71     signal L4Kaux2 : sfixed(18 downto -18);
72     signal L4Kaux3 : sfixed(18 downto -18);
73     signal L4Kaux4 : sfixed(18 downto -18);
74     signal L4Kaux5 : sfixed(18 downto -18);
75
76     --*****
77     --Compoe o vetor de parametros estimados.
78
79     signal theta1K : sfixed(4 downto -17);
80     signal theta2K : sfixed(4 downto -17);
81     signal theta3K : sfixed(4 downto -17);
82     signal theta4K : sfixed(4 downto -17);
83
84     --*****
85     --Compoe o vetor de parametros estimados no instante (k-1) .
86
87     signal theta1K1 : sfixed(4 downto -17);
88     signal theta2K1 : sfixed(4 downto -17);
89     signal theta3K1 : sfixed(4 downto -17);
90     signal theta4K1 : sfixed(4 downto -17);
91
92     --*****
93     signal aux1_theta : sfixed(18 downto -18);
94

```

```

95  --*****
96  --Compoe a matriz de covariancia.
97
98  signal P1K : sfixed(4 downto -17);
99  signal P2K : sfixed(4 downto -17);
100 signal P3K : sfixed(4 downto -17);
101 signal P4K : sfixed(4 downto -17);
102 signal P5K : sfixed(4 downto -17);
103 signal P6K : sfixed(4 downto -17);
104 signal P7K : sfixed(4 downto -17);
105 signal P8K : sfixed(4 downto -17);
106 signal P9K : sfixed(4 downto -17);
107 signal P10K : sfixed(4 downto -17);
108 signal P11K : sfixed(4 downto -17);
109 signal P12K : sfixed(4 downto -17);
110 signal P13K : sfixed(4 downto -17);
111 signal P14K : sfixed(4 downto -17);
112 signal P15K : sfixed(4 downto -17);
113 signal P16K : sfixed(4 downto -17);
114
115  --*****
116  --Compoe a matriz de covariancia no instante (k-1).
117
118  signal P1K1 : sfixed(4 downto -17);
119  signal P2K1 : sfixed(4 downto -17);
120  signal P3K1 : sfixed(4 downto -17);
121  signal P4K1 : sfixed(4 downto -17);
122  signal P5K1 : sfixed(4 downto -17);
123  signal P6K1 : sfixed(4 downto -17);
124  signal P7K1 : sfixed(4 downto -17);
125  signal P8K1 : sfixed(4 downto -17);
126  signal P9K1 : sfixed(4 downto -17);
127  signal P10K1 : sfixed(4 downto -17);
128  signal P11K1 : sfixed(4 downto -17);
129  signal P12K1 : sfixed(4 downto -17);
130  signal P13K1 : sfixed(4 downto -17);
131  signal P14K1 : sfixed(4 downto -17);
132  signal P15K1 : sfixed(4 downto -17);
133  signal P16K1 : sfixed(4 downto -17);
134
135  --*****
136  --Recebem os parametros estimados.
137
138  signal k0_b : sfixed(4 downto -17);
139  signal k2_b : sfixed(4 downto -17);
140  signal k3_b : sfixed(4 downto -17);
141  signal k4_b : sfixed(4 downto -17);
142
143  --*****
144
145  signal s0 : sfixed(12 downto -12);--utilizado p/ calcular a lei de
146  --de controle.
147
148  signal refComp_b : sfixed(11 downto 0);--recebe o vetor senoide(+/-2047).
149  signal valatual_b : sfixed(11 downto 0);--recebe saida convertida(+/-2047).
150  signal ek_b : sfixed(12 downto 0);--recebe o erro;malha fechada(+/-4095).
151  signal ek1_b : sfixed(12 downto 0);--armazena o erro e(k-1).
152  signal ek2_b : sfixed(12 downto 0);--armazena o erro e(k-2).
153  signal uk_b : sfixed(18 downto -18);--sinal de controle.
154  signal uk1_b : sfixed(18 downto -18);--sinal de controle no instante u(k-1).
155  signal uk2_b : sfixed(18 downto -18);--sinal de controle no instante u(k-2).
156
157  signal saida_b : sfixed(18 downto -18);--recebe o sinal de controle.
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198

```

Fonte: produção do próprio autor

6.3 CONTROLE PID DIGITAL DA TENSÃO DE SAÍDA DO CONVERSOR NPC_m MONOFÁSICO

Aplicando o método de recorrência temporal apresentado em [10], na Equação (22) chega-se a equação a diferenças dado pela Equação (48).

$$\begin{aligned}
 C(z) &= \frac{0,4(z^2 - 1,2z + 0,793)}{(z - 1)(z + 0,267)} \\
 C(z) &= \frac{0,4 z^2 - 0,48 z + 0,3172}{z^2 - 0,733 z - 0,267} \times \frac{z^{-2}}{z^{-2}} \\
 C(z) &= \frac{0,4 - 0,48 z^{-1} + 0,3172z^{-2}}{1 - 0,733 z^{-1} - 0,267z^{-2}} \\
 \frac{U(z)}{E(z)} &= \frac{0,4 - 0,48 z^{-1} + 0,3172z^{-2}}{1 - 0,733 z^{-1} - 0,267z^{-2}} \\
 U(z)(1 - 0,733 z^{-1} - 0,267z^{-2}) &= E(z)(0,4 - 0,48 z^{-1} + 0,3172z^{-2}) \\
 u(k) &= 0,733u(k - 1) + 0,267u(k - 2) + 0,4e(k) \\
 &\quad - 0,48e(k - 1) + 0,3172e(k - 2)
 \end{aligned} \tag{48}$$

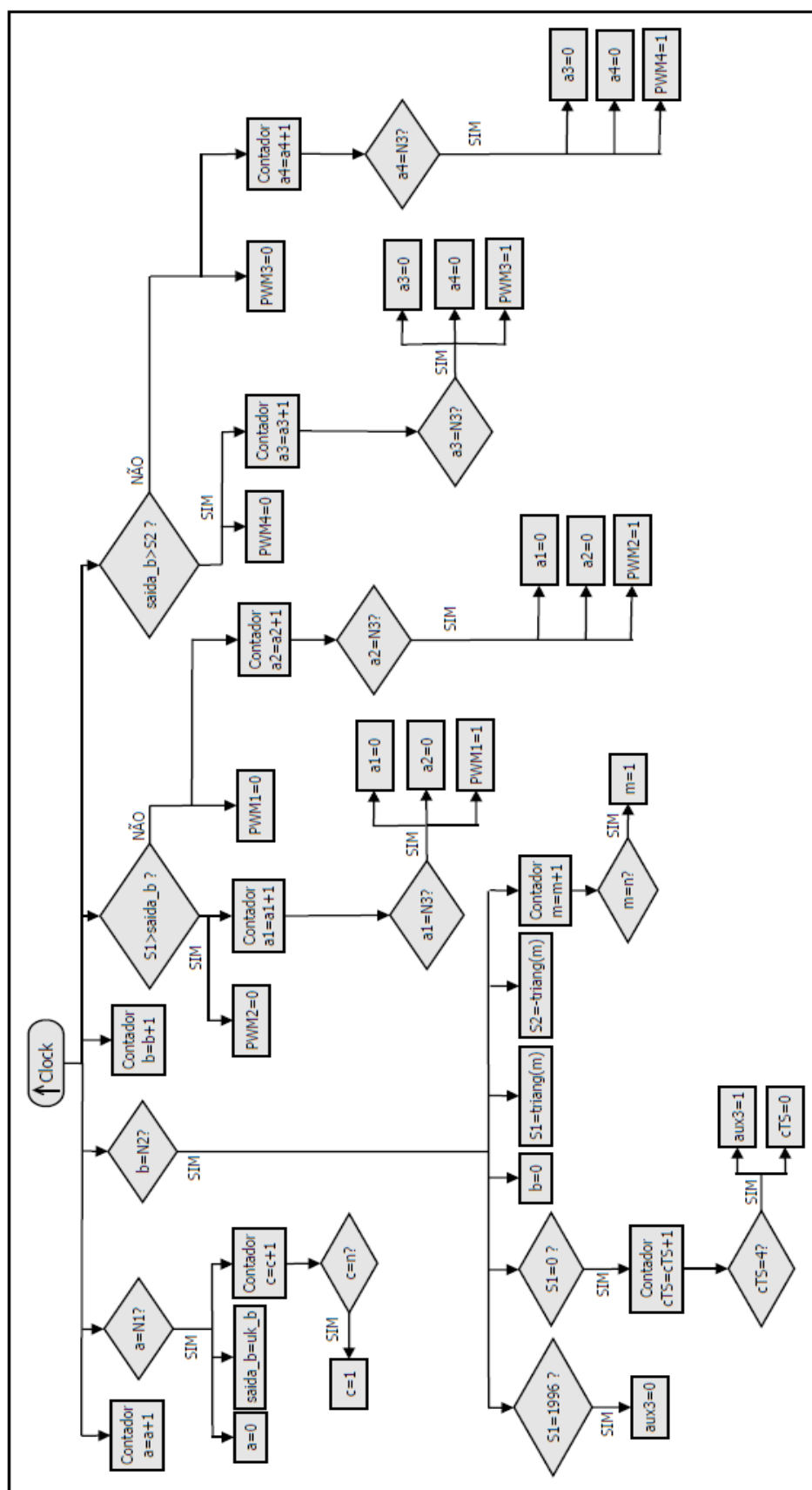
A Equação (48) pode ser escrita em VHDL e o *hardware* para realizar o cálculo de $u(k)$ pode ser inferido no FPGA. O loop de controle consiste em:

1. Ler o conversor A/D;
2. Atualizar $e(k - 2)$;
3. Atualizar $u(k - 2)$;
4. Atualizar $e(k - 1)$;
5. Atualizar $u(k - 1)$;
6. Calcular $e(k)$;
7. Calcular $u(k)$;
8. Atribuir $u(k)$ ao modulador;
9. Aguardar o próximo instante de amostragem.

6.3.1 Algoritmo de programação

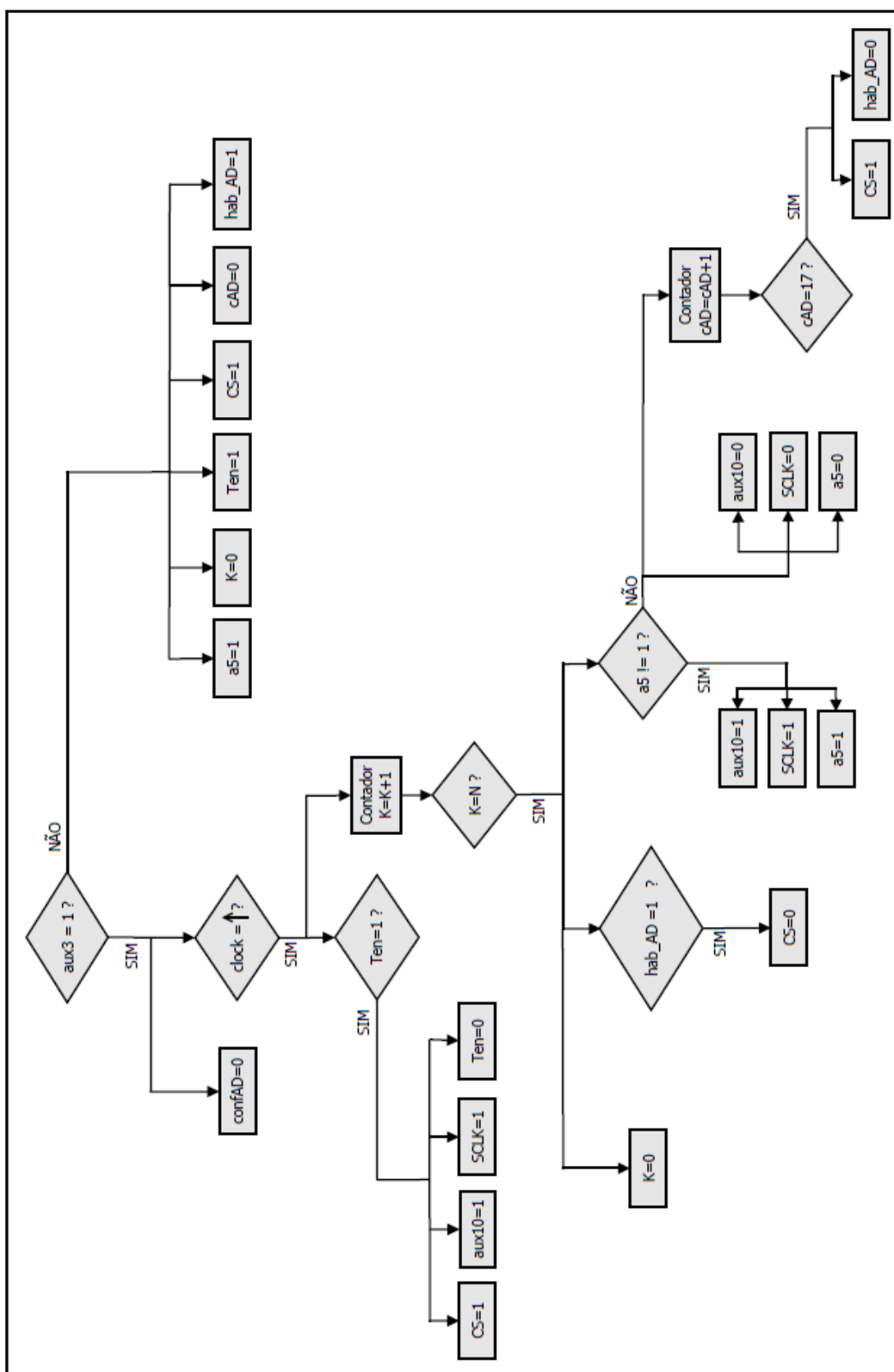
O código escrito em VHDL, para efetuar o controle por meio de um PID Digital, da tensão de saída de um conversor NPCm monofásico de três níveis é composto por três módulos que realizam tarefas distintas. São eles: Modulação PWM, Conversão A/D e Cálculo da Lei de Controle. O algoritmo responsável pela modulação PWM é apresentado em [6] para um conversor NPC, equivalente ao NPCm no que se refere à modulação. Na seção 6.1 foi detalhado o funcionamento do módulo referente à conversão A/D. O algoritmo criado para efetuar o cálculo de $u(k)$, deve executar o loop de controle citado acima a partir do passo 2, já que o primeiro passo é realizado pelo módulo de conversão A/D. Os fluxogramas apresentados nas Figuras 72, 73, 74 e 75 ilustram o algoritmo completo que efetua a modulação PWM, a conversão A/D e o cálculo da lei de controle.

Figura 72 - Fluxograma de modulação PWM para conversor NPC três níveis monofásico com o controlador



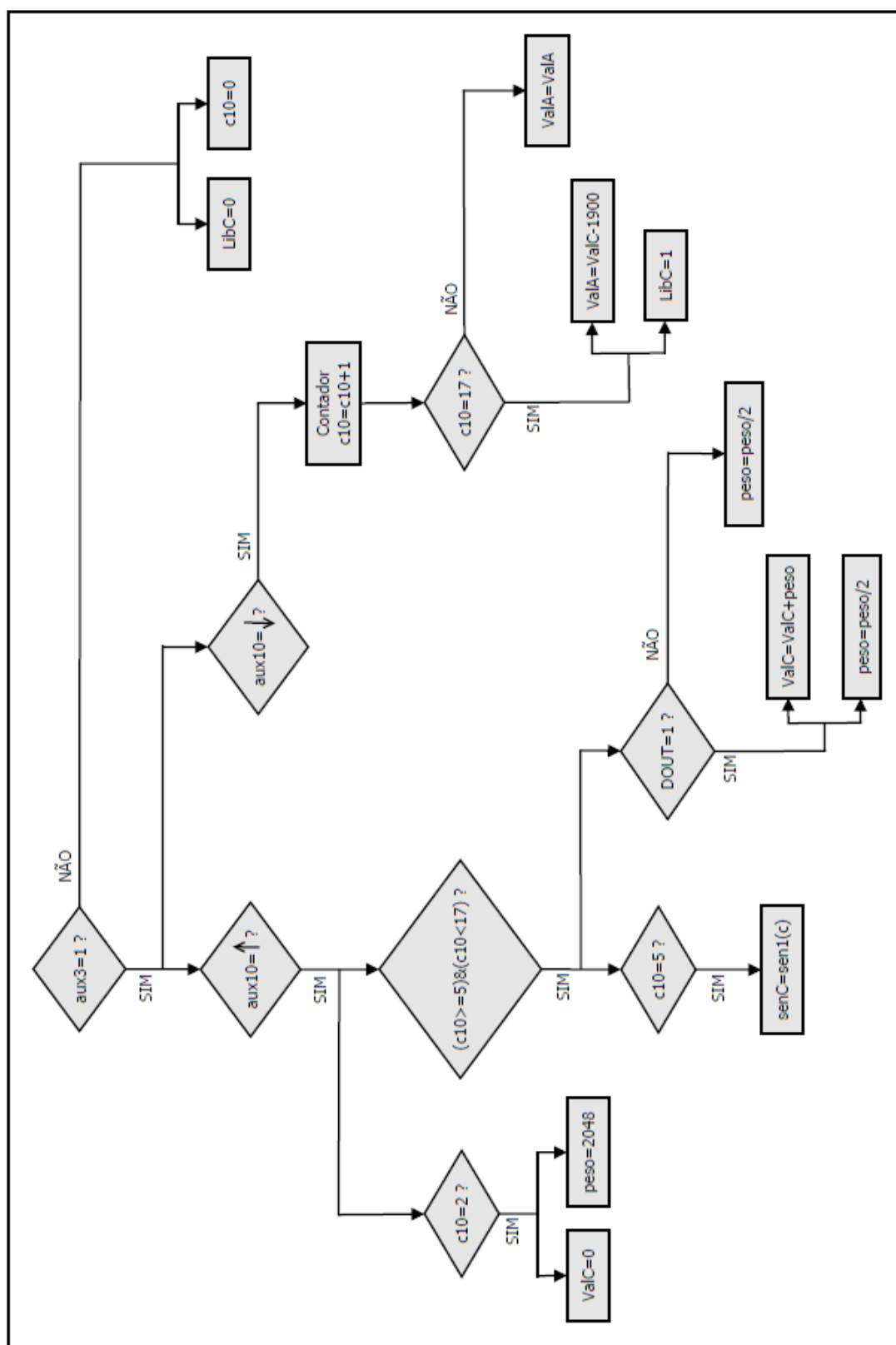
Fonte: baseado em [6]

Figura 73 – Fluxograma da conversão A/D (geração do *clock* do conversor A/D e definição do canal)



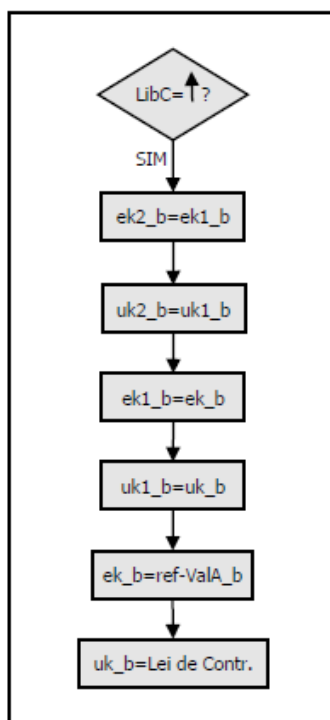
Fonte: produção do próprio autor

Figura 74 – Fluxograma da conversão A/D (obtenção do valor amostrado e eliminação do *offset*)



Fonte: produção do próprio autor

Figura 75 – Fluxograma da rotina de cálculo da lei de controle



Fonte: produção do próprio autor

Na Figura 76 segue a rotina programada em VHDL, responsável pela execução do loop de controle citado acima a partir do passo 2. O passo 1 é mostrado na seção 6.1. O código completo é apresentado no Apêndice A.

Figura 76 – Rotina em VHDL que executa o *loop* de controle

```

412     rotina4: process (LiberaCalculo)
413
414     begin
415
416     --*****
417     --*****Projeto (C=4.3u e fs = 10KHz)*****
418
419
420     k0_b <= to_sfixed(0.733,k0_b);--Coeficiente a1. Atribui a k0_b
421                                     --um valor ja convertido p/ um tipo
422                                     --pto fixo c/ sinal.
423     k1_b <= to_sfixed(0.267,k1_b);--Coeficiente a2. Atribui a k1_b um
424                                     --valor ja convertido p/ um tipo pto
425                                     --fixo c/ sinal.
426     k2_b <= to_sfixed(0.4,k2_b);--Coeficiente b0. Atribui a k2_b um valor
427                                     --ja convertido p/ um tipo pto fixo c/ sinal.
428     k3_b <= to_sfixed(0.48,k3_b);--Coeficiente b1. Atribui a k3_b um valor
429                                     --ja convertido p/ um tipo pto fixo c/ sinal.
430     k4_b <= to_sfixed(0.3172,k4_b);--Coeficiente b2. Atribui a k4_b um valor
431                                     --ja convertido p/ um tipo pto fixo c/ sinal.
432     refComp_b <= to_sfixed(senoComp,refComp_b);--Atribui a refComp_b o valor da
433                                     --referencia convertido p/ um
434                                     --tipo pto fixo c/ sinal.
435     valatual_b <= to_sfixed(ValorAtual,valatual_b);--Atribui a valatual_b o
  
```

```

436                                     --valor da amostragem convertido
437                                     --p/ um tipo pto fixo c/ sinal.
438
439
440     if rising_edge (LiberaCalculo) then
441
442         ek2_b <= ek1_b;--Atribui ek1_b a ek2_b p/ que na proxima
443                     --amostragem o calculo do sinal de controle
444                     --conte com o valor do erro de duas amostragem passada.
445         uk2_b <= uk1_b;--Atribui uk1_b a uk2_b p/ que na proxima
446                     --amostragem o calculo do sinal de controle
447                     --conte com o valor do sinal de controle obtido
448                     --a duas amostragem passada.
449         ek1_b <= ek_b;--Atribui ek_b a ek1_b p/ que na proxima amostragem
450                     --o calculo do sinal de controle conte com o valor
451                     --anterior do erro.
452         uk1_b <= uk_b;--Atribui uk_b a uk1_b p/ que na proxima amostragem
453                     --o calculo do sinal de controle conte com o valor
454                     --anterior do sinal de controle.
455
456         ek_b <= refComp_b - valatual_b;--Atribui a "ek_b" a diferenca entre
457                                     --o valor de referencia e o valor
458                                     --amostrado.
459
460
461     --*****
462     --Calculo do sinal de controle.
463
464     uk_b <= resize(k0_b*uk1_b + k1_b*uk2_b + k2_b*ek_b - k3_b*ek1_b + k4_b*ek2_b,uk_b);
465
466     --*****
467
468     end if;
469
470     end process rotina4;
471
472 end ADConverter;

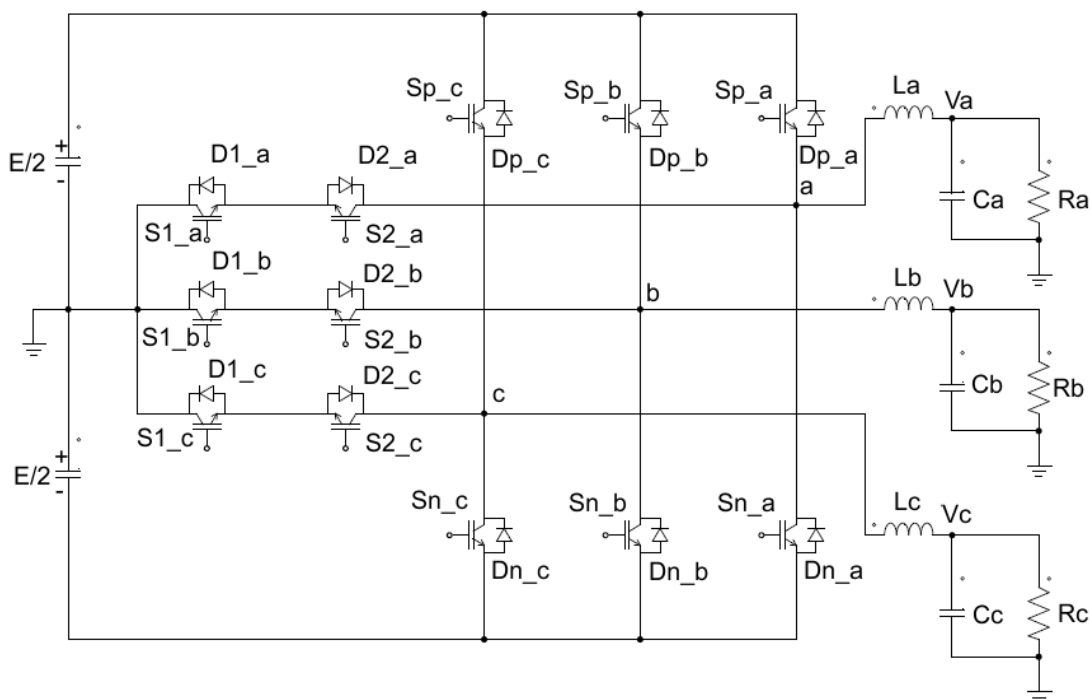
```

Fonte: produção do próprio autor

6.4 CONTROLE PID DIGITAL DAS TENSÕES DE SAÍDA DO CONVERSOR NPCm TRIFÁSICO

O controle da tensão de saída do conversor NPCm trifásico, topologia exibida na Figura 77, pode ser realizado de forma independente para cada fase, como se fossem três conversores monofásicos. Esta característica está relacionada ao fato, do ponto comum entre as duas fontes $E/2$, estar conectado em uma das extremidades do filtro LC de cada fase. Portanto, o código escrito em VHDL para controlar essas tensões é apenas uma adaptação do código apresentado no Apêndice A, sendo que tanto as rotinas para calcular os sinais de controle como os parâmetros das leis de controle são apenas réplicas do que é apresentado neste código.

Figura 77 – Estrutura de um conversor NPCM trifásico



Fonte: produção do próprio autor

Uma das adaptações, refere-se a estratégia adotada para realizar a amostragem do sinal de saída de cada fase utilizando o conversor A/D do KIT DE0 – Nano. Este conversor A/D não permite que se efetue a amostragem das três fases simultaneamente. O código apresentado no Apêndice A, foi replicado para atender o número de chaves da estrutura trifásica. A introdução de uma lógica para realizar a defasagem entre as três fases também foi necessária.

6.4.1 Defasagem, amostragem das 3 fases e cálculo da lei de controle

A fim de gerar uma defasagem de 120° entre as fases, foi inserido duas lógicas condicionais que quando satisfeitas habilitam os incrementos dos contadores de índice do vetor que descreve a função seno, possibilitando a geração de mais dois sinais modulantes.

As senoides geradas internamente apresentam um valor de pico de 1900.

$$Amplitude = Amplitude_{MAX} \sin(\theta) \quad (49)$$

De acordo com a Equação (49), para um ângulo igual a 120° e uma amplitude máxima de 1900, a amplitude será de aproximadamente 1645. O valor mais próximo de 1645 que está

presente no vetor que descreve a senoide no código do Apêndice B é 1644. O valor 1644 está localizado na posição de índice 209 do vetor. Da mesma forma, para um ângulo igual a 240° a amplitude será de aproximadamente -1645 . O valor mais próximo de -1645 está localizado na posição de índice 417. Portanto, a condição para que a moduladora que representa a segunda fase seja reproduzida é que o índice da primeira fase esteja com um valor igual a 209 e para que o sinal da terceira moduladora seja reproduzido, o índice da primeira fase deve ter um valor igual a 417. As Figuras 78 e 79 mostram partes do código que está presente no Apêndice B. Entre as linhas 262 e 273 está a lógica citada acima.

Figura 78 – Lógica escrita em VHDL para gerar a defasagem entre as fases de um sistema trifásico

```

253         if a=N1 then
254
255             saida_R <= uk_b_R;--Atribui a variavel "saida_R" o valor do
256                 --sinal de controle "uk_b_R".
257
258             --saida_R := senoide1(count_R);
259
260             count_R:=1+count_R;--Incrementa 1 na variavel "count1".
261
262             if count_R = 209 then--logica p/ promover a defasagem de 120
263                 --graus da fase S.
264                 ref_S := 1;--habilita o process que gera a modulacao PWM
265                     --p/ a fase S.
266                 defasag <= '1';--p/ verificar a defasagem entre as fases
267             end if;
268             if count_R = 417 then--logica p/ promover a defasagem de 240
269                 --graus da fase T.
270                 ref_T := 1;--habilita o process que gera a modulacao PWM
271                     --p/ a fase T.
272                 defasag <= '0';--p/ verificar a defasagem entre as fases
273             end if;
274             -----
275             if count_R>625 then--Limita count1 em 625 (tamanho do vetor "senoide1")
276                 count_R:=1;
277             end if;
278             a:=0;--zera a variavel "a" p/ reiniciar o processo.
279         end if;

```

Fonte: produção do próprio autor

Na linha 399 da Figura 79 está a condição que permite ou não a geração da moduladora referente a fase S, também chamada neste trabalho de “segunda fase”, a partir do contador nomeado de “count_S”. O contador “count_S” executa a varredura do vetor “senoide1”. Quando a condição para a amostragem da fase S é satisfeita, a variável chamada de “senoComp_S” armazena o valor do vetor nomeado de “senoide1” cujo índice é dado por “count_S”. O mesmo procedimento é adotado para a fase T.

Figura 79 – Código escrito em VHDL para a modulação da fase S, ressaltando a condição que permite a geração da mesma

```

397         if clk='1' then
398
399             if ref_S = 1 then
400
401                 aS:=1+aS;
402                 bS:=1+bS;
403
404                 if aS=N1 then
405
406                     saida_S <= uk_b_S;--Atribui a variavel "saida_S" o
407                     --valor do sinal de controle "uk_b_S".
408                     count_S:=count_S+1;
409                     if count_S>625 then--Limita count_S em 625
410                         --(tamanho do vetor "senoide1")
411                         count_S:=1;
412                     end if;
413
414                     aS:=0;
415                 end if;
416
417
418                 if saida_S > saidat1 then--Comparacao entre o sinal de controle
419                     --ea portadora positiva(Modulacao da fase S)
420
421                     pwms2<='0';--Coloca em nivel logico baixo a saida "pwms2"
422                     a1S:=1+a1S;--Incrementa 1 na variavel "a1S" a cada pulso de clock,
423                     --estabelecendo um contador.
424                     if a1S=N3 then--Logica utilizada p/ gerar o tempo morto
425                         --entre as chaves.
426                         pwms1<='1';--Coloca em nivel logico alto a saida "pwms1"
427                         a1S:=0;
428                         a2S:=0;
429                     end if;
430                 else
431                     pwms1<='0';--Coloca em nivel logico baixo a saida "pwms1"
432                     a2S:=1+a2S;--Incrementa 1 na variavel "a2S" a cada pulso de clock,
433                     --estabelecendo um contador.
434                     if a2S=N3 then--Logica utilizada p/ gerar o tempo morto
435                         --entre as chaves.
436                         pwms2<='1';--Coloca em nivel logico alto a saida "pwms2"
437                         a2S:=0;
438                         a1S:=0;
439                     end if;
440                 end if;
441                 if saida_S > saidat2 then--Comparacao entre o sinal de controle ea
442                     --portadora negativa(Modulacao da fase S).
443
444                     pwms4<='0';--Coloca em nivel logico baixo a saida "pwms4"
445                     a3S:=1+a3S;--Incrementa 1 na variavel "a3S" a cada pulso de clock,
446                     --estabelecendo um contador.
447                     if a3S=N3 then--Logica utilizada p/ gerar o tempo morto
448                         --entre as chaves.
449                         pwms3<='1';--Coloca em nivel logico alto a saida "pwms3"
450                         a3S:=0;
451                         a4S:=0;
452                     end if;
453                 else
454                     pwms3<='0';--Coloca em nivel logico baixo a saida "pwms3"
455                     a4S:=1+a4S;--Incrementa 1 na variavel "a4S" a cada pulso de clock,
456                     --estabelecendo um contador.
457                     if a4S=N3 then--Logica utilizada p/ gerar o tempo morto
458                         --entre as chaves.
459                         pwms4<='1';--Coloca em nivel logico alto a saida "pwms4".
460                         a3S:=0;
461                         a4S:=0;
462                     end if;
463                 end if;
464             end if;
465         end if;
466     end if;
467
468 end if;
469
470 end process rotina7;

```

Fonte: produção do próprio autor

A Figura 80 mostra as três variáveis que recebem os valores das amostras de cada fase.

Figura 80 – Detalhes do código em VHDL para controle do sistema trifásico

```

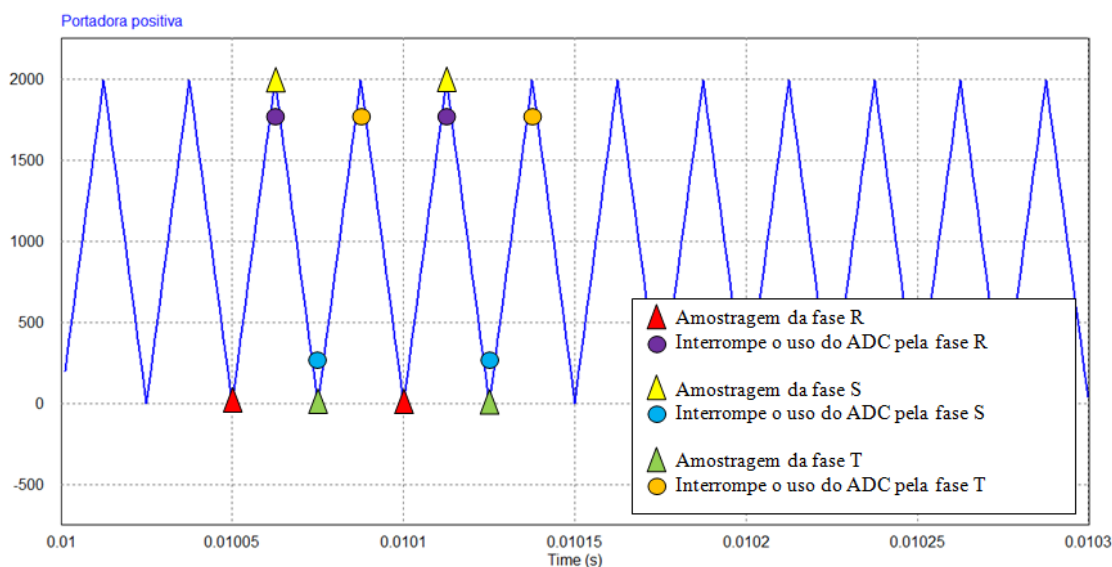
671         if cont10 = 5 then --Permite salvar o valor do sinal de referencia no instante
672             --da amostragem p/ comparacao posterior.
673             senoComp_R := senoidel(count_R); --Carrega a referencia da fase R na variavel
674             --"senoComp_R".
675             senoComp_S := senoidel(count_S); --Carrega a referencia da fase S na variavel
676             --"senoComp_S".
677             senoComp_T := senoidel(count_T); --Carrega a referencia da fase T na variavel
678             --"senoComp_T".
679         end if;

```

Fonte: produção do próprio autor

Como mencionado na seção 6.1, o conversor A/D do KIT DE0 – Nano disponibiliza 8 canais de entrada analógica. Configurando uma entrada para cada fase obtêm-se o *hardware* necessário para realizar a amostragem das três fases. Entretanto, os canais devem ser configurados em instantes diferentes, pois existe uma única saída serial [19]. Na metodologia adotada para realizar a amostragem das três fases, procurou-se manter o princípio de amostrar no momento em que não ocorre um chaveamento. A Figura 81 mostra o instante em que ocorre cada amostragem e também a interrupção do uso do conversor A/D por cada fase.

Figura 81 – Indicações dos instantes de amostragem para o sistema trifásico



Fonte: produção do próprio autor

Da linha 290 à linha 335 do código exibido na Figura 82, têm-se a lógica utilizada para definir os instantes de amostragem e interrupções ilustradas na Figura 81.

Figura 82 – Código em VHDL para as definições dos instantes de amostragem

```

290         if hab_amostrT = 1 then--Condicao p/ habilitar a amostragem da fase T
291             --no instante predefinido.
292             if saidat1 = 0 then--Define o instante da Amostragem da fase T.
293                 auxT3<='1';--Habilita o "process" responsavel pelo ADC (p/ fase T).
294                 hab_amostrT := 0;--zera a variavel "hab_amostrT" p/
295                     --reiniciar o processo.
296             end if;
297         end if;
298
299         if (saidat1 = 1760)then--Condicao p/ interromper o uso do ADC na fase T.
300             auxT3<='0';--Interrompe o uso do ADC na fase T.
301         end if;
302
303         --*****
304
305
306         if saidat1 = aux1 then --Condicao p/ habilitar a amostragem da fase R
307             --no instante predefinido.
308             contTS := contTS + 1;
309             if contTS = 2 then--Amostra a cada 2 ciclos da portadora.
310                 auxR3<='1';--Habilita o "process" responsavel pelo ADC (p/ fase R).
311                 hab_amostrT := 1;--Inicializa a variavel usada p/ possibilitar a
312                     --sequencia de amostragem.
313                 contTS := 0;
314             end if;
315         end if;
316
317         if (saidat1 = 1760)then--Condicao p/ interromper o uso do ADC na fase R.
318             auxR3<='0';--Interrompe o uso do ADC na fase R.
319         end if;
320
321         --*****
322         if hab_amostrT = 1 then--Condicao p/ habilitar a amostragem da fase S no
323             --instante predefinido.
324             if saidat1 = 1997 then--Define o instante da Amostragem da fase S.
325                 auxS3<='1';--Habilita o "process" responsavel pelo ADC (p/ fase S).
326             end if;
327         end if;
328
329         if (saidat1 = 300)then--Condicao p/ interromper o uso do ADC na fase S.
330             auxS3<='0';--Interrompe o uso do ADC na fase S.
331         end if;
332
333         if count2>625 then--Limita count2 em 625 (tamanho do vetor "triangular1").
334             count2:=1;--Atribui 1 a count2, pois o indice do vetor inicia em 1.
335         end if;

```

Fonte: produção do próprio autor

A amostragem das fases R, S e T são realizadas a partir das entradas analógicas IN0, IN7 e IN6 respectivamente. A Figura 83 mostra a parte do código, que está no Apêndice B, que efetua a escolha do canal.

Figura 83 – Rotina em VHDL para a escolha do canal do conversor A/D

```

594         if (auxR3 = '1') then --Condicao p/ selecionar o canal
595             --IN0 do ADC.
596             confAD <= '0'; --Seleciona canal IN0. Saida conectada
597             --ao pino DIN do ADC.
598         end if;
599
600         if (auxS3 = '1') then --Condicao p/ selecionar o
601             --canal IN7 do ADC.
602             confAD <= '1'; --Seleciona canal IN7
603         end if;
604
605         if (auxT3 = '1') then --Condicao p/ selecionar o
606             --canal IN6 do ADC.
607         cont_AD_T := cont_AD_T + 1;
608             if cont_AD_T <= 4 then --Define qual nivel logico sera
609                 --colocado no pino DIN na subida
610                 --do SCLK.
611                 confAD <= '1'; --Seleciona canal IN6 do ADC.
612             else
613                 confAD <= '0'; --Seleciona canal IN6 do ADC.
614             end if;
615         end if;

```

Fonte: produção do próprio autor

Os parâmetros da lei de controle e o cálculo do sinal de controle são os mesmos utilizados no controle do conversor NPCm de três níveis monofásico apresentado na seção 6.3. A única diferença é que no programa desenvolvido para o conversor NPCm trifásico, existem três rotinas ao invés de uma, que calculam um sinal de controle para cada fase.

6.5 IMPLEMENTAÇÃO DO *INDIRECT SELF TUNING* NO FPGA

6.5.1 Implementação do MQR

O vetor φ composto pelos dados medidos e aplicados no sistema e o vetor $\hat{\theta}$ composto pelos parâmetros estimados serão definidos por:

$$\varphi(k) = \begin{bmatrix} \varphi_{1k} \\ \varphi_{2k} \\ \varphi_{3k} \\ \varphi_{4k} \end{bmatrix} = \begin{bmatrix} -y(k-1) \\ -y(k-2) \\ u(k-1) \\ u(k-2) \end{bmatrix} \quad (50)$$

$$\hat{\theta}(k) = \begin{bmatrix} \theta_{1k} \\ \theta_{2k} \\ \theta_{3k} \\ \theta_{4k} \end{bmatrix} = \begin{bmatrix} \hat{a}_1 \\ \hat{a}_2 \\ \hat{b}_0 \\ \hat{b}_1 \end{bmatrix} \quad (51)$$

$$\hat{\theta}(k-1) = \begin{bmatrix} \theta 1_{k-1} \\ \theta 2_{k-1} \\ \theta 3_{k-1} \\ \theta 4_{k-1} \end{bmatrix} \quad (52)$$

As Equações (53), (54) e (55) definem a matriz de covariância P e o ganho L .

$$P(k) = \begin{bmatrix} P1_k & P2_k & P3_k & P4_k \\ P5_k & P6_k & P7_k & P8_k \\ P9_k & P10_k & P11_k & P12_k \\ P13_k & P14_k & P15_k & P16_k \end{bmatrix} \quad (53)$$

$$P(k-1) = \begin{bmatrix} P1_{k-1} & P2_{k-1} & P3_{k-1} & P4_{k-1} \\ P5_{k-1} & P6_{k-1} & P7_{k-1} & P8_{k-1} \\ P9_{k-1} & P10_{k-1} & P11_{k-1} & P12_{k-1} \\ P13_{k-1} & P14_{k-1} & P15_{k-1} & P16_{k-1} \end{bmatrix} \quad (54)$$

$$L(k) = \begin{bmatrix} L1_k \\ L2_k \\ L3_k \\ L4_k \end{bmatrix} \quad (55)$$

Como não é possível escrever um código em VHDL que seja sintetizável em *hardware* para resolver cálculos com matrizes, deve-se a partir da Equação (38) obter uma equação para cada parâmetro de ganho L [21]. Para facilitar a representação gráfica das equações, o denominador será atribuído a uma variável com prefixo “den” e o numerador a uma variável com o prefixo “num”.

$$\begin{bmatrix} L1_k \\ L2_k \\ L3_k \\ L4_k \end{bmatrix} = \frac{\begin{bmatrix} P1_{k-1} & P2_{k-1} & P3_{k-1} & P4_{k-1} \\ P5_{k-1} & P6_{k-1} & P7_{k-1} & P8_{k-1} \\ P9_{k-1} & P10_{k-1} & P11_{k-1} & P12_{k-1} \\ P13_{k-1} & P14_{k-1} & P15_{k-1} & P16_{k-1} \end{bmatrix} \begin{bmatrix} \varphi 1_k \\ \varphi 2_k \\ \varphi 3_k \\ \varphi 4_k \end{bmatrix}}{1 + [\varphi 1_k \ \varphi 2_k \ \varphi 3_k \ \varphi 4_k] \begin{bmatrix} P1_{k-1} & P2_{k-1} & P3_{k-1} & P4_{k-1} \\ P5_{k-1} & P6_{k-1} & P7_{k-1} & P8_{k-1} \\ P9_{k-1} & P10_{k-1} & P11_{k-1} & P12_{k-1} \\ P13_{k-1} & P14_{k-1} & P15_{k-1} & P16_{k-1} \end{bmatrix} \begin{bmatrix} \varphi 1_k \\ \varphi 2_k \\ \varphi 3_k \\ \varphi 4_k \end{bmatrix}}$$

$$numL1_k = P1_{k-1}\varphi 1_k + P2_{k-1}\varphi 2_k + P3_{k-1}\varphi 3_k + P4_{k-1}\varphi 4_k$$

$$\begin{aligned}
denL_k = 1 + & [\varphi 1_k (P1_{k-1} \varphi 1_k + P2_{k-1} \varphi 2_k + P3_{k-1} \varphi 3_k + P4_{k-1} \varphi 4_k) \\
& + \varphi 2_k (P5_{k-1} \varphi 1_k + P6_{k-1} \varphi 2_k + P7_{k-1} \varphi 3_k + P8_{k-1} \varphi 4_k) \\
& + \varphi 3_k (P9_{k-1} \varphi 1_k + P10_{k-1} \varphi 2_k + P11_{k-1} \varphi 3_k + P12_{k-1} \varphi 4_k) \\
& + \varphi 4_k (P13_{k-1} \varphi 1_k + P14_{k-1} \varphi 2_k + P15_{k-1} \varphi 3_k + P16_{k-1} \varphi 4_k)] \\
L1_k = & \frac{numL1_k}{denL_k}
\end{aligned} \tag{56}$$

$$numL2_k = P5_{k-1} \varphi 1_k + P6_{k-1} \varphi 2_k + P7_{k-1} \varphi 3_k + P8_{k-1} \varphi 4_k$$

$$L2_k = \frac{numL2_k}{denL_k} \tag{57}$$

$$numL3_k = P9_{k-1} \varphi 1_k + P10_{k-1} \varphi 2_k + P11_{k-1} \varphi 3_k + P12_{k-1} \varphi 4_k$$

$$L3_k = \frac{numL3_k}{denL_k} \tag{58}$$

$$numL4_k = P13_{k-1} \varphi 1_k + P14_{k-1} \varphi 2_k + P15_{k-1} \varphi 3_k + P16_{k-1} \varphi 4_k$$

$$L4_k = \frac{numL4_k}{denL_k} \tag{59}$$

A Equação (37) também deve ser resolvida para a obtenção de uma equação para cada parâmetro estimado [21].

$$\begin{bmatrix} \theta 1_k \\ \theta 2_k \\ \theta 3_k \\ \theta 4_k \end{bmatrix} = \begin{bmatrix} \theta 1_{k-1} \\ \theta 2_{k-1} \\ \theta 3_{k-1} \\ \theta 4_{k-1} \end{bmatrix} + \begin{bmatrix} L1_k \\ L2_k \\ L3_k \\ L4_k \end{bmatrix} \left(y(k) - [\varphi 1_k \ \varphi 2_k \ \varphi 3_k \ \varphi 4_k] \begin{bmatrix} \theta 1_{k-1} \\ \theta 2_{k-1} \\ \theta 3_{k-1} \\ \theta 4_{k-1} \end{bmatrix} \right)$$

$$\begin{aligned}
\theta 1_k = \theta 1_{k-1} + L1_k (y(k) - \varphi 1_k \theta 1_{k-1} - \varphi 2_k \theta 2_{k-1} \\
- \varphi 3_k \theta 3_{k-1} - \varphi 4_k \theta 4_{k-1})
\end{aligned} \tag{60}$$

$$\begin{aligned}
\theta 2_k = \theta 2_{k-1} + L2_k (y(k) - \varphi 1_k \theta 1_{k-1} - \varphi 2_k \theta 2_{k-1} \\
- \varphi 3_k \theta 3_{k-1} - \varphi 4_k \theta 4_{k-1})
\end{aligned} \tag{61}$$

$$\begin{aligned}
\theta 3_k = \theta 3_{k-1} + L3_k (y(k) - \varphi 1_k \theta 1_{k-1} - \varphi 2_k \theta 2_{k-1} \\
- \varphi 3_k \theta 3_{k-1} - \varphi 4_k \theta 4_{k-1})
\end{aligned} \tag{62}$$

$$\theta 4_k = \theta 4_{k-1} + L 4_k(y(k) - \varphi 1_k \theta 1_{k-1} - \varphi 2_k \theta 2_{k-1} - \varphi 3_k \theta 3_{k-1} - \varphi 4_k \theta 4_{k-1}) \quad (63)$$

Abaixo segue a solução da Equação (39) para a obtenção de uma equação para cada parâmetro de covariância [21].

$$\begin{bmatrix} P1_k & P2_k & P3_k & P4_k \\ P5_k & P6_k & P7_k & P8_k \\ P9_k & P10_k & P11_k & P12_k \\ P13_k & P14_k & P15_k & P16_k \end{bmatrix} = \begin{pmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} L1_k \\ L2_k \\ L3_k \\ L4_k \end{bmatrix} [\varphi 1_k \ \varphi 2_k \ \varphi 3_k \ \varphi 4_k] \end{pmatrix} \begin{bmatrix} P1_{k-1} & P2_{k-1} & P3_{k-1} & P4_{k-1} \\ P5_{k-1} & P6_{k-1} & P7_{k-1} & P8_{k-1} \\ P9_{k-1} & P10_{k-1} & P11_{k-1} & P12_{k-1} \\ P13_{k-1} & P14_{k-1} & P15_{k-1} & P16_{k-1} \end{bmatrix}$$

$$P1_k = P1_{k-1} - L1_k(P1_{k-1}\varphi 1_k + P5_{k-1}\varphi 2_k + P9_{k-1}\varphi 3_k + P13_{k-1}\varphi 4_k) \quad (64)$$

$$P2_k = P2_{k-1} - L1_k(P2_{k-1}\varphi 1_k + P6_{k-1}\varphi 2_k + P10_{k-1}\varphi 3_k + P14_{k-1}\varphi 4_k) \quad (65)$$

$$P3_k = P3_{k-1} - L1_k(P3_{k-1}\varphi 1_k + P7_{k-1}\varphi 2_k + P11_{k-1}\varphi 3_k + P15_{k-1}\varphi 4_k) \quad (66)$$

$$P4_k = P4_{k-1} - L1_k(P4_{k-1}\varphi 1_k + P8_{k-1}\varphi 2_k + P12_{k-1}\varphi 3_k + P16_{k-1}\varphi 4_k) \quad (67)$$

$$P5_k = P5_{k-1} - L2_k(P5_{k-1}\varphi 2_k + P1_{k-1}\varphi 1_k + P9_{k-1}\varphi 3_k + P13_{k-1}\varphi 4_k) \quad (68)$$

$$P6_k = P6_{k-1} - L2_k(P6_{k-1}\varphi 2_k + P2_{k-1}\varphi 1_k + P10_{k-1}\varphi 3_k + P14_{k-1}\varphi 4_k) \quad (69)$$

$$P7_k = P7_{k-1} - L2_k(P7_{k-1}\varphi 2_k + P3_{k-1}\varphi 1_k + P11_{k-1}\varphi 3_k + P15_{k-1}\varphi 4_k) \quad (70)$$

$$P8_k = P8_{k-1} - L2_k(P8_{k-1}\varphi 2_k + P4_{k-1}\varphi 1_k + P12_{k-1}\varphi 3_k + P16_{k-1}\varphi 4_k) \quad (71)$$

$$P9_k = P9_{k-1} - L3_k(P9_{k-1}\varphi3_k + P1_{k-1}\varphi1_k + P5_{k-1}\varphi2_k + P13_{k-1}\varphi4_k) \quad (72)$$

$$P10_k = P10_{k-1} - L3_k(P10_{k-1}\varphi3_k + P2_{k-1}\varphi1_k + P6_{k-1}\varphi2_k + P14_{k-1}\varphi4_k) \quad (73)$$

$$P11_k = P11_{k-1} - L3_k(P11_{k-1}\varphi3_k + P3_{k-1}\varphi1_k + P7_{k-1}\varphi2_k + P15_{k-1}\varphi4_k) \quad (74)$$

$$P12_k = P12_{k-1} - L3_k(P12_{k-1}\varphi3_k + P4_{k-1}\varphi1_k + P8_{k-1}\varphi2_k + P6_{k-1}\varphi4_k) \quad (75)$$

$$P13_k = P13_{k-1} - L4_k(P13_{k-1}\varphi4_k + P1_{k-1}\varphi1_k + P5_{k-1}\varphi2_k + P9_{k-1}\varphi3_k) \quad (76)$$

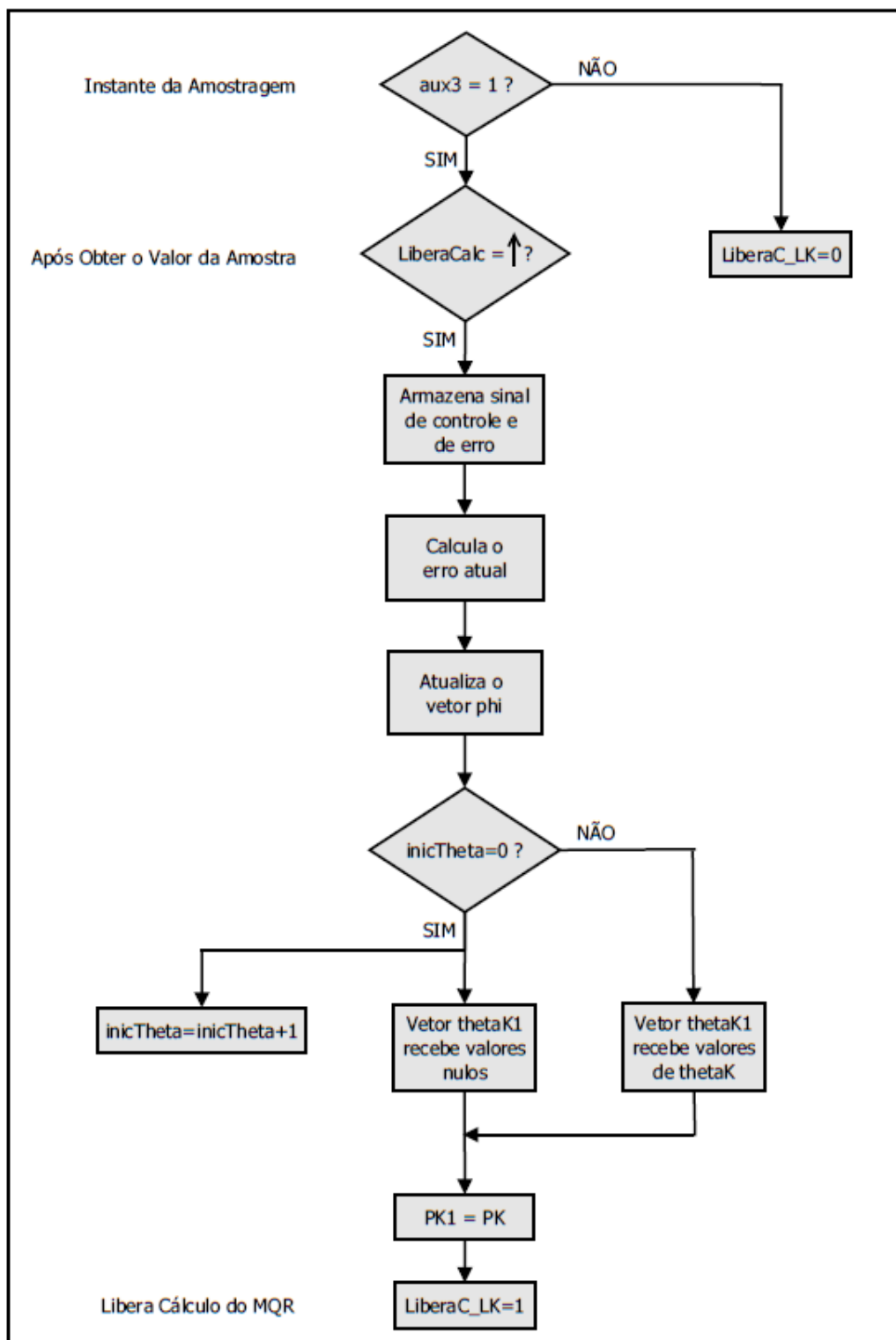
$$P14_k = P14_{k-1} - L4_k(P14_{k-1}\varphi4_k + P2_{k-1}\varphi1_k + P6_{k-1}\varphi2_k + P10_{k-1}\varphi3_k) \quad (77)$$

$$P15_k = P15_{k-1} - L4_k(P15_{k-1}\varphi4_k + P3_{k-1}\varphi1_k + P7_{k-1}\varphi2_k + P11_{k-1}\varphi3_k) \quad (78)$$

$$P16_k = P16_{k-1} - L4_k(P16_{k-1}\varphi4_k + P4_{k-1}\varphi1_k + P8_{k-1}\varphi2_k + P12_{k-1}\varphi3_k) \quad (79)$$

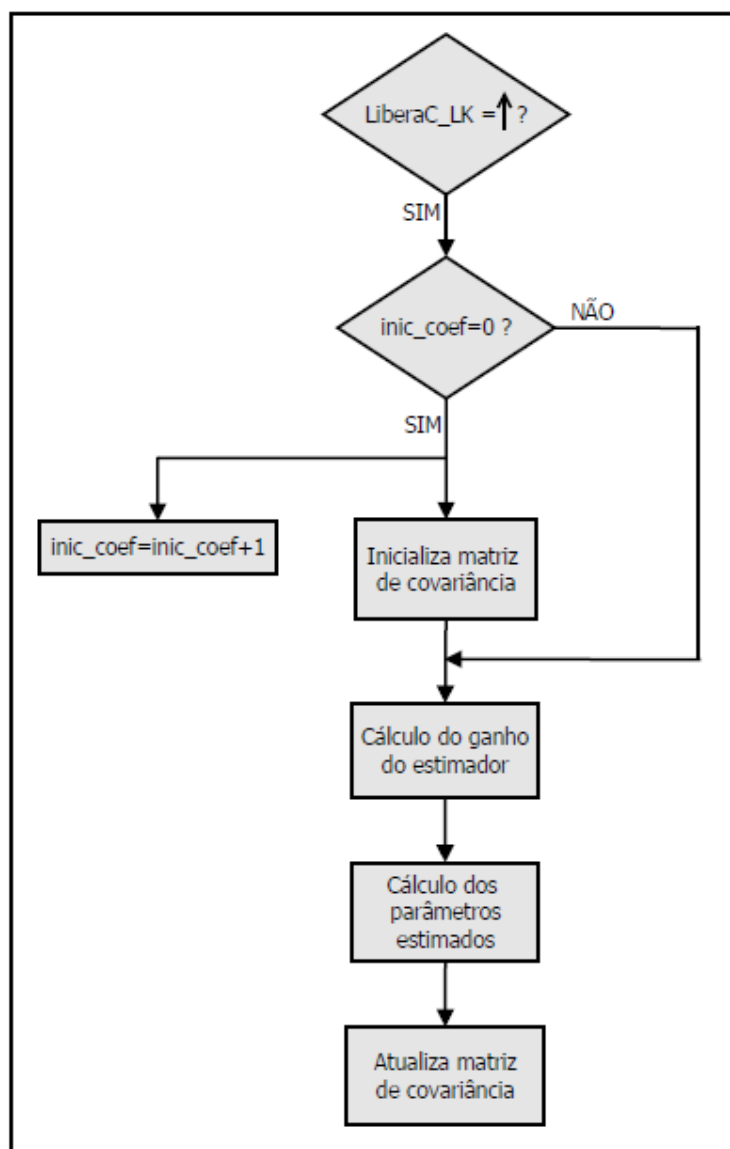
As Equações de (56) à (79) foram escritas em VHDL para implementar o MQR em um FPGA. O código completo é apresentado no Apêndice C e o MQR está compreendido entre as linhas 599 a 904. As Figuras 84 e 85 apresentam dois fluxogramas, referentes ao algoritmo do MQR, pertencente ao código que implementa um sistema de controle *Indirect Self Tuning*, para controle de tensão em um conversor NPCm de três níveis monofásico.

Figura 84 – Fluxograma do algoritmo habilita o cálculo do MQR



Fonte: produção do próprio autor

Figura 85 – Fluxograma do MQR

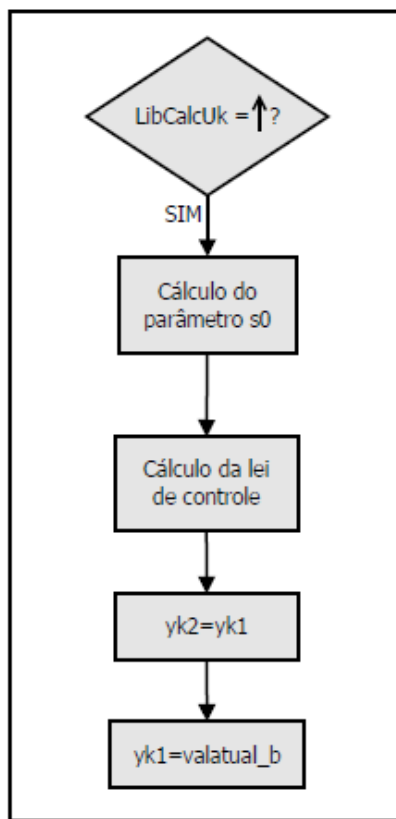


Fonte: produção do próprio autor

6.5.2 Implementação da lei de controle do PID digital recursivo

As Equações (42), (45) e (46) foram escritas em VHDL de forma a obter um código sintetizável em *hardware* e que promova um sistema de controle do tipo *Indirect Self Tuning*. Entre as linhas 906 a 922 do Apêndice C, pode-se observar a rotina que efetua o cálculo do sinal de controle. Abaixo, segue o fluxograma do algoritmo referente ao cálculo da lei de controle do PID Digital recursivo.

Figura 86 – Fluxograma do cálculo da lei de controle



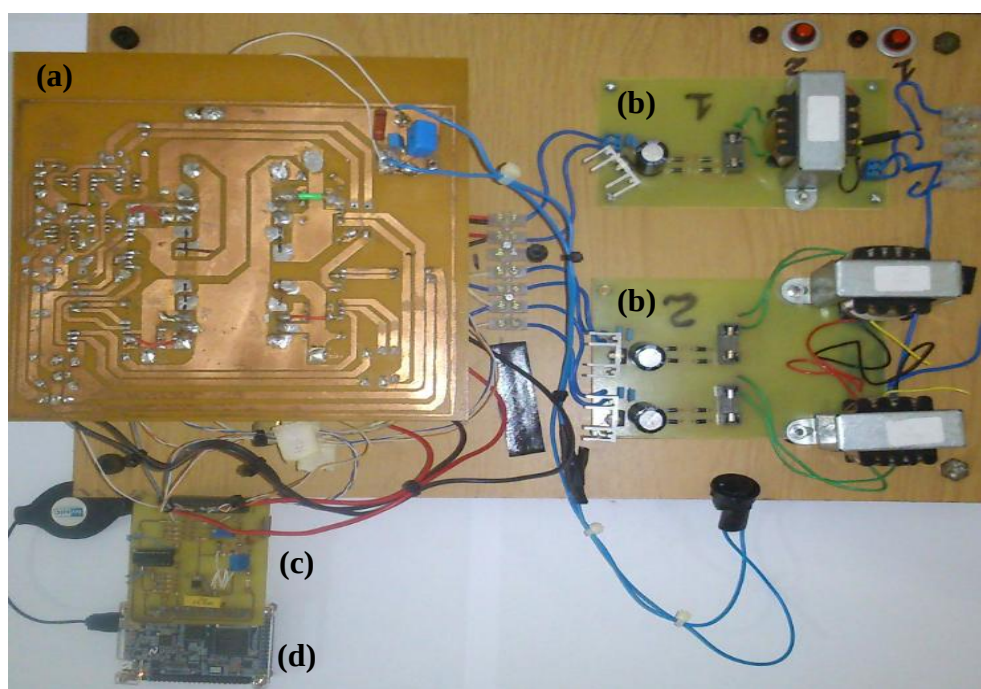
Fonte: produção do próprio autor

O código completo em VHDL implementando um sistema de controle *Indirect Self-Tuning* para controlar a tensão de um conversor NPCm de três níveis monofásico cujo modelo é dado pela Equação (19) é apresentado no Apêndice C.

7 RESULTADOS EXPERIMENTAIS

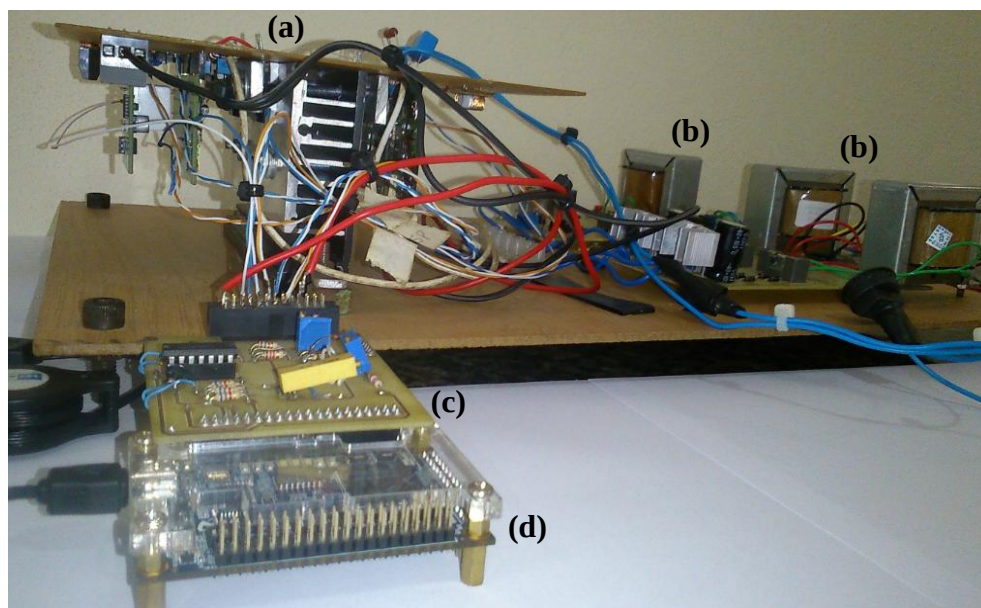
O sistema montado para efetuar os testes práticos deste trabalho é apresentado nas Figuras 87 e 88. O sistema é composto por 3 fontes de 15Vcc, por um conversor NPCm de três níveis monofásico, por uma placa que realiza o condicionamento do sinal de tensão medido e também faz a interface, para os canais PWM, entre o kit DE0 – Nano e os drivers do conversor NPCm. O conversor NPCm é um projeto do Eng^o José Adriano Damacena Diesel, realizado durante o estágio obrigatório e apresentado em [23]. Algumas adaptações foram realizadas, a fim de substituir o sistema de controle analógico original pelo sistema de controle digital. A placa de condicionamento e interface, conta com um amplificador operacional MAX4238, para proporcionar ao sinal medido o ganho e o *offset* necessários. A placa também possui um circuito integrado SN54LS07, com 6 buffers, para realizar a interface mencionada acima. O projeto desta placa não é apresentado neste trabalho, pois foi baseado nos circuitos apresentados em [12]. Esta placa, as 3 fontes de 15Vcc e a plataforma com os botões que acionam as fontes, foram confeccionadas pelo autor deste trabalho.

Figura 87 – Vista superior do sistema: (a) Conversor NPCm 1 ϕ ; (b) Fontes auxiliares; (c) Placa de condicionamento e interface; (d) Kit DE0-Nano.



Fonte: produção do próprio autor

Figura 88 – Vista lateral do sistema: (a) Conversor NPCm 1 ϕ ; (b) Fontes auxiliares; (c) Placa de condicionamento e interface; (d) Kit DE0-Nano.



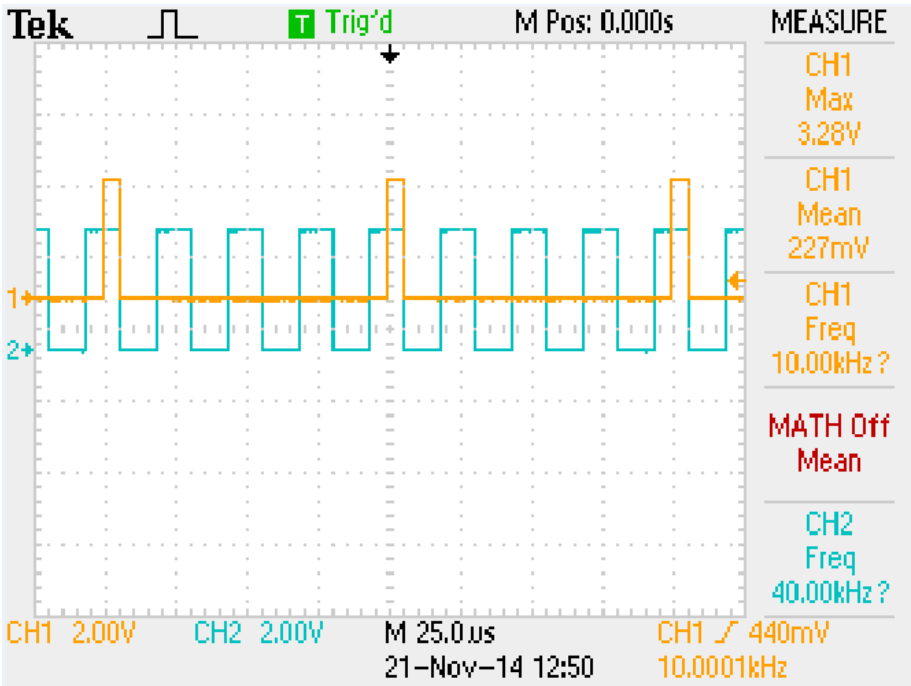
Fonte: produção do próprio autor

Os itens 7.1 e 7.4 estão relacionado ao código exibido no Apêndice A, que implementa um controlador PID digital para o conversor NPCm de três níveis monofásico. Os tópicos 7.2 e 7.3 referem-se ao código apresentado no Apêndice B, que implementa um controlador PID digital para o conversor NPCm de três níveis trifásico.

7.1 ATRASOS

Um aspecto importante na avaliação de desempenho do *hardware* inferido no FPGA é o tempo em que ele executa determinadas tarefas após a solicitação das mesmas. Na Figura 89, a onda quadrada em azul foi gerada a partir dos valores mínimo e máximo da portadora positiva. A borda de subida ocorre no momento em que a portadora alcança seu valor mínimo (zero) e a borda de descida ocorre quando a portadora alcança o seu valor máximo (1996). Este sinal, resultante do código criado em VHDL, foi atribuído à um pino externo do kit DE0 – Nano. A borda de subida do sinal em laranja, indica o momento em que o valor de uma amostra é obtido, após a solicitação da amostragem que ocorre na borda de subida do sinal em azul. A frequência da portadora é de 40kHz e a frequência de amostragem é de 10kHz.

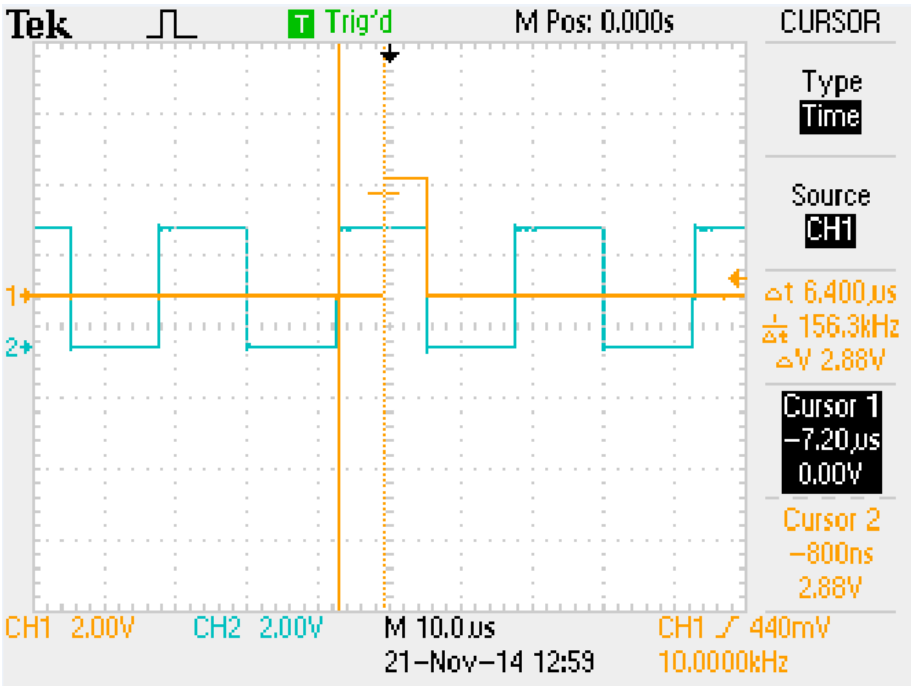
Figura 89 – Instante da obtenção do valor da amostra



Fonte: produção do próprio autor

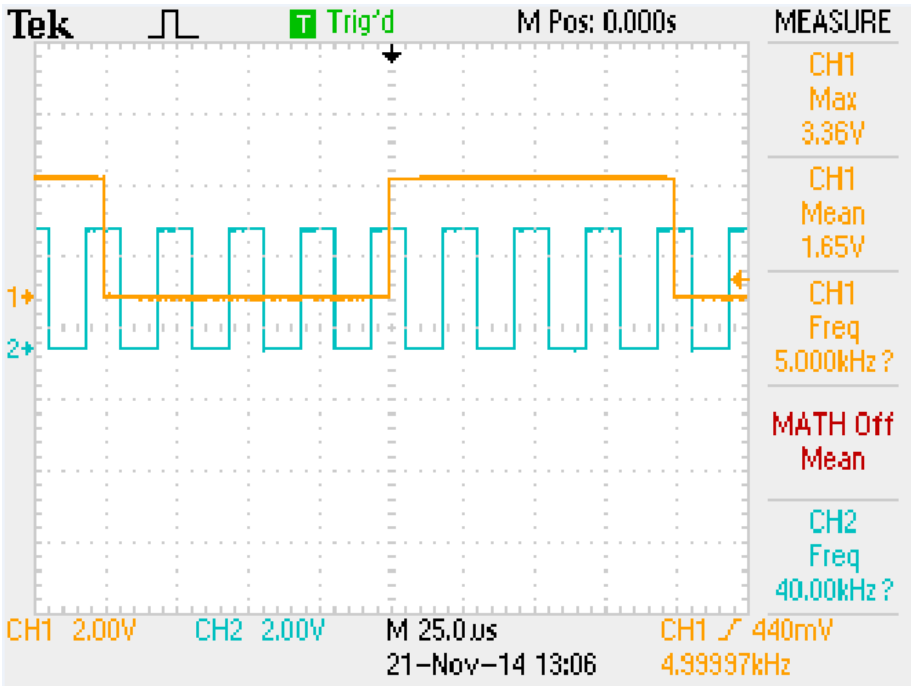
Nota-se na Figura 90 um atraso entre a solicitação de amostragem e a obtenção do valor da amostra de 6,4µs. Este atraso, como mencionado na seção 6.1 é decorrente da comunicação serial entre o FPGA e o conversor A/D do kit DE0 – Nano.

Figura 90 - Atraso entre a solicitação de amostragem e a obtenção do valor da amostra



Fonte: produção do próprio autor

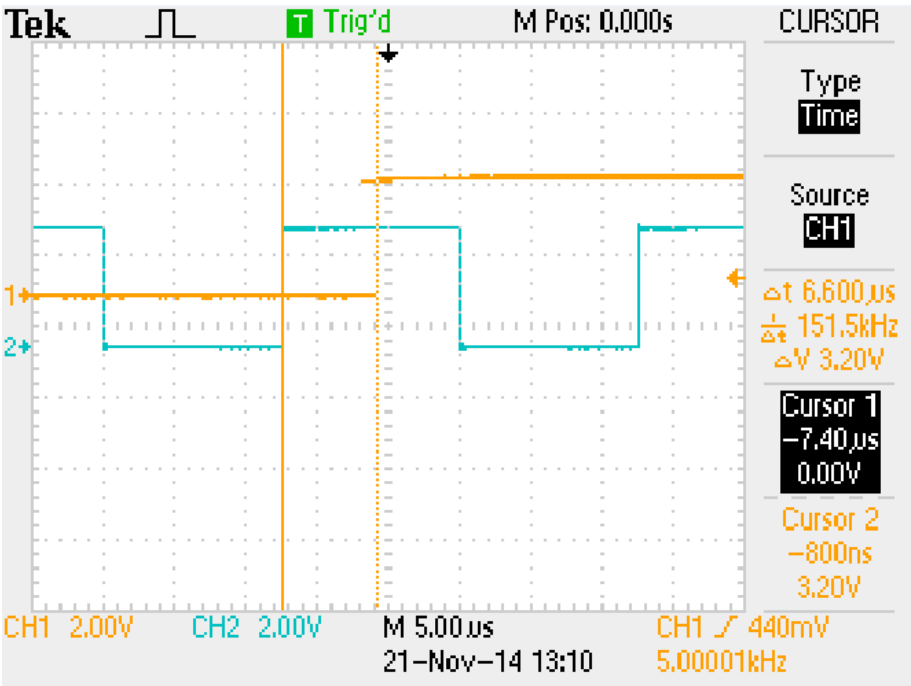
Figura 91 - Instante do fim do cálculo da lei de controle



Fonte: produção do próprio autor

Outra análise efetuada foi a verificação do atraso que ocorre entre a solicitação de amostragem e a obtenção do sinal de controle. As duas transições, bordas de subida e descida do sinal em laranja da Figura 91 indicam que o sinal de controle foi atualizado.

Figura 92 - Atraso entre o instante da amostragem e o fim do cálculo da lei de controle



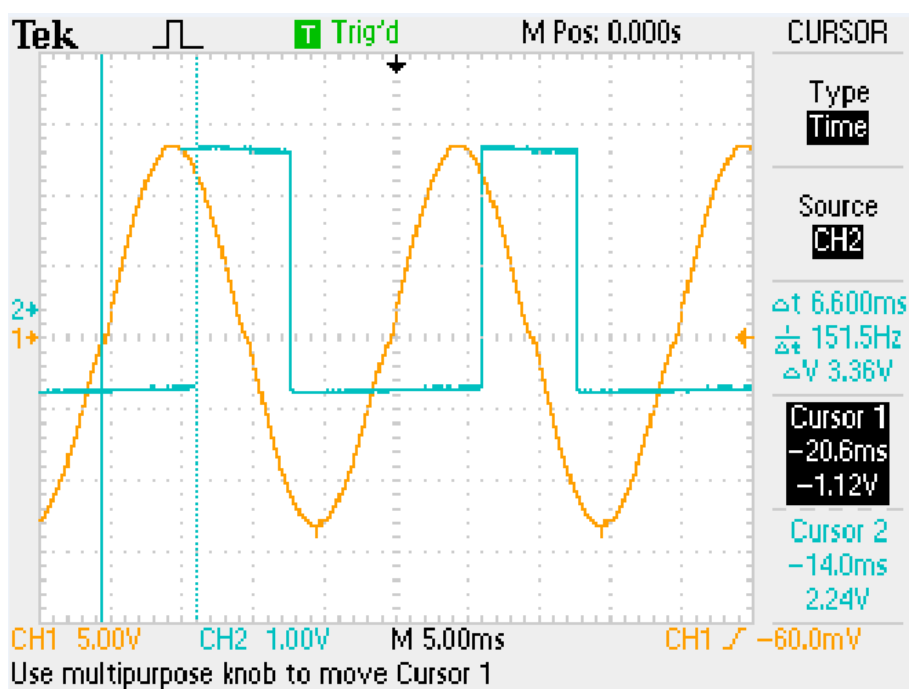
Fonte: produção do próprio autor

De acordo com a Figura 92 o sinal de controle é atualizado $6,6\mu\text{s}$ após a solicitação de amostragem. Como o valor da amostra é obtido com um atraso de $6,4\mu\text{s}$, pode-se dizer que o restante do processo para a obtenção do sinal de controle demora $0,2\mu\text{s}$.

7.2 DEFASAGEM ENTRE AS FASES

Como mencionado na seção 6.4 o *hardware* inferido no FPGA para o controle de um conversor NPCm trifásico (Figura 77) gera internamente três modulantes que também atuam como sinais de referência. Estes sinais devem estar defasados entre eles 120° . Na Figura 93, o sinal em laranja é o sinal de saída do conversor NPCm, referente a fase R. As bordas de subida e descida do sinal em azul indicam os instantes em que são liberadas as gerações das modulantes referentes as fases S e T. O tempo de $6,6\text{ms}$ entre o momento em que a fase R passa por zero e a borda de subida da onda quadrada indica que existe uma defasagem de aproximadamente 119° entre a fase R e a fase S.

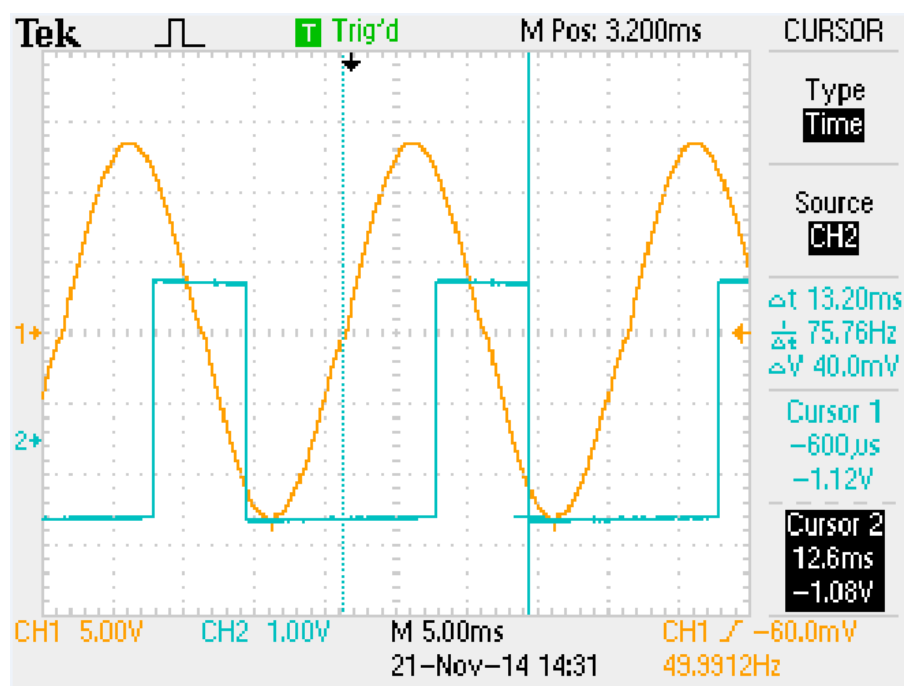
Figura 93 – Defasagem entre a fase R e a fase S



Fonte: produção do próprio autor

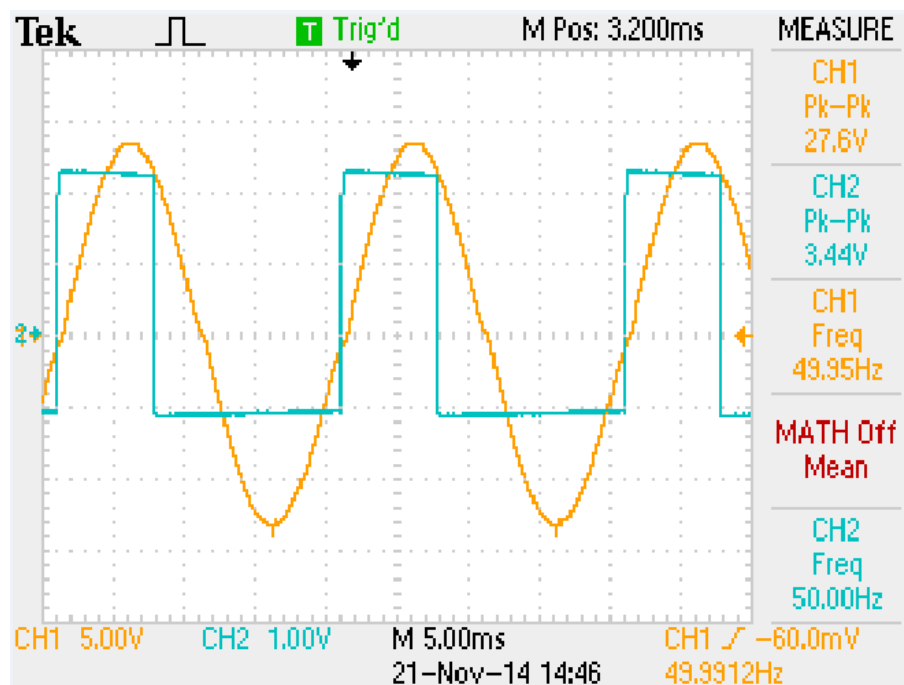
Na Figura 94, o tempo de $13,2\text{ms}$ entre o momento em que a fase R passa por zero e a borda de descida da onda quadrada indica que existe uma defasagem de aproximadamente 238° entre a fase R e a fase T.

Figura 94 – Defasagem entre a fase R e a fase T



Fonte: produção do próprio autor

Figura 95 – Instante da liberação da fase S e fase S



Fonte: produção do próprio autor

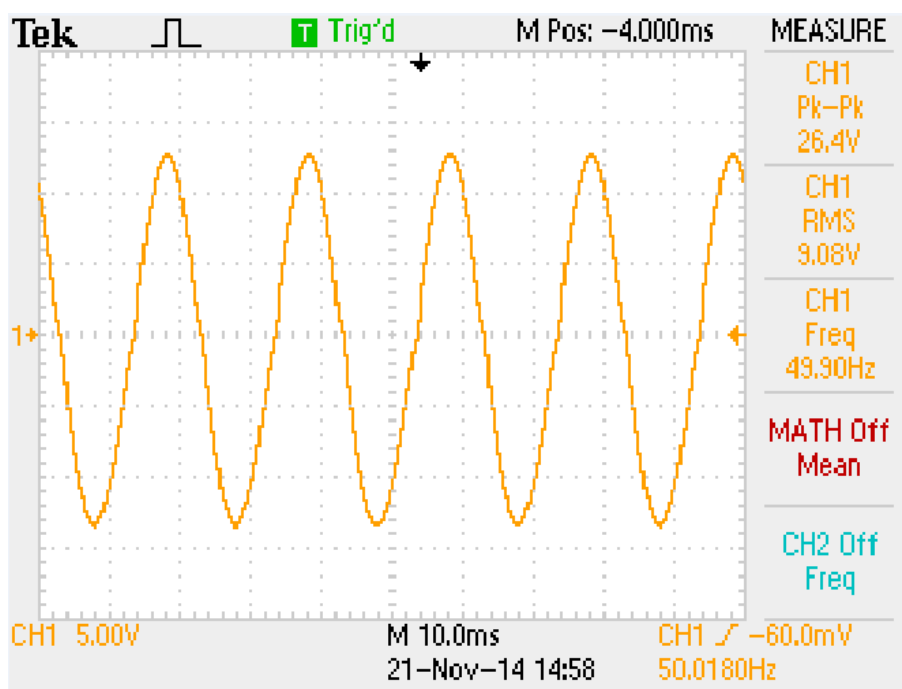
O sinal em laranja apresentado na Figura 95 é o sinal de tensão de saída do conversor NPCm, referente a fase S. Pode-se notar que a fase S, passa por zero no momento em que ocorre uma borda de subida da onda quadrada (aproximadamente). Como mencionado

anteriormente, esta transição é a indicação de que a geração do sinal modulante referente a fase S, foi liberada. Para realização deste teste, foram colocados nas saídas físicas do kit DE0 – Nano que conectam o kit ao conversor NPCM, os sinais PWM referente a fase S.

7.3 TENSÃO DE SAÍDA EM MALHA FECHADA

A forma de onda apresentada na Figura 96, trata-se do sinal de tensão de saída do conversor NPCM operando em malha fechada, onde o sinal de controle é proveniente do cálculo da Equação (48). Para obter esta forma de onda, foi utilizado o *hardware* inferido a partir do código escrito em VHDL para o sistema trifásico. A onda de tensão é referente a fase S, o que implica na utilização de um segundo canal analógico do conversor A/D.

Figura 96 – Fase S em malha fechada



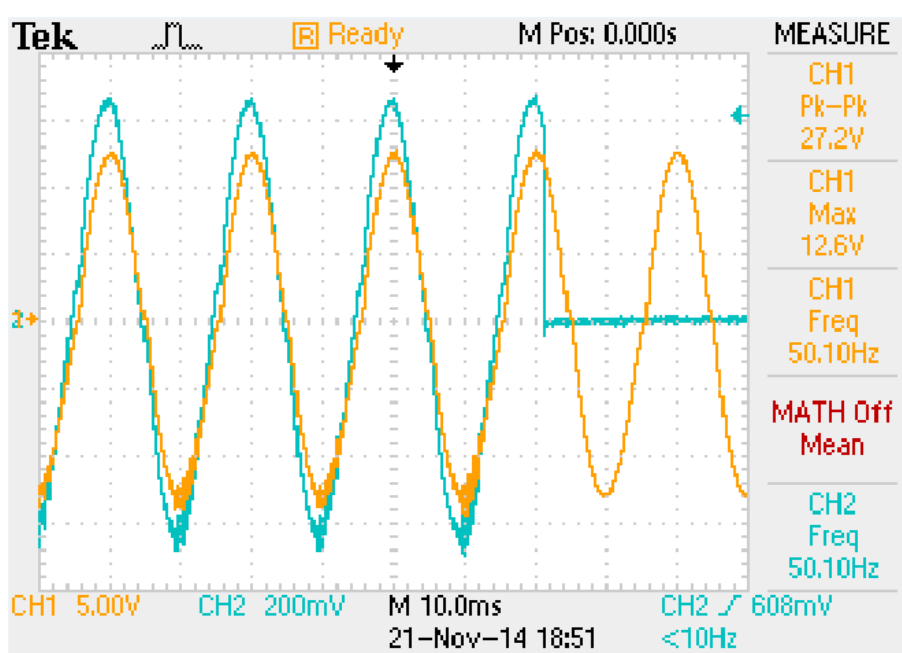
Fonte: produção do próprio autor

O sinal possui um valor de pico a pico de 26,4V e tem um valor de *offset* praticamente nulo. A frequência é de 49,9Hz.

7.4 CONSTATAÇÃO DA AÇÃO DE CONTROLE

Neste trabalho, o interesse é pelo sinal de tensão em cima da carga (resistor) que está acoplada ao filtro LC da saída do conversor NPCm monofásico. Tanto na simulação quanto no experimento prático é efetuada uma adição e uma redução de carga, através de uma inserção de um resistor em paralelo com um outro fixo, ou retirada do mesmo. Na Figura 97 é apresentado o instante da retirada de um dos resistores, diminuindo a carga.

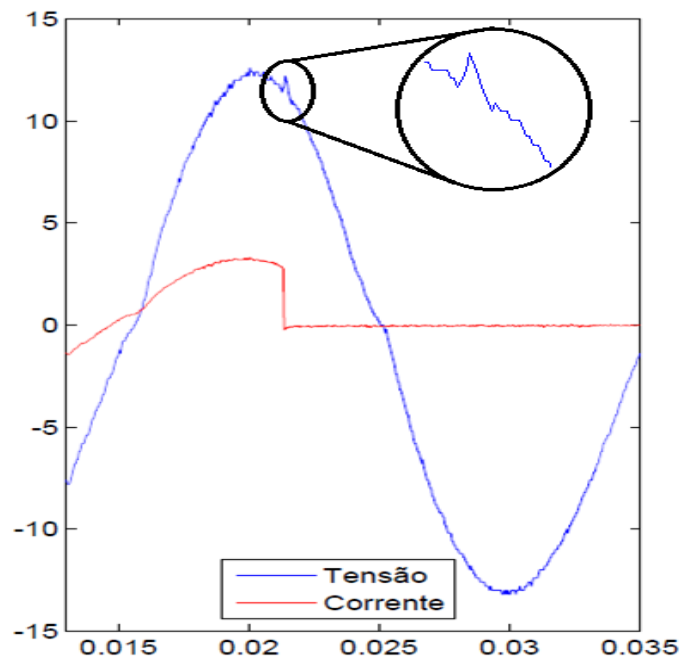
Figura 97 – Redução da carga



Fonte: produção do próprio autor

Utilizando o *software* MATLAB[®] é possível efetuar uma análise mais minuciosa dos dados obtidos com um osciloscópio digital. Os mesmos dados que compõem as ondas de corrente e tensão da Figura 97 foram utilizados para compor os sinais da Figura 98. Na Figura 98 fica evidente a elevação da tensão no momento da retirada da carga. O controlador então atua para acomodar este efeito.

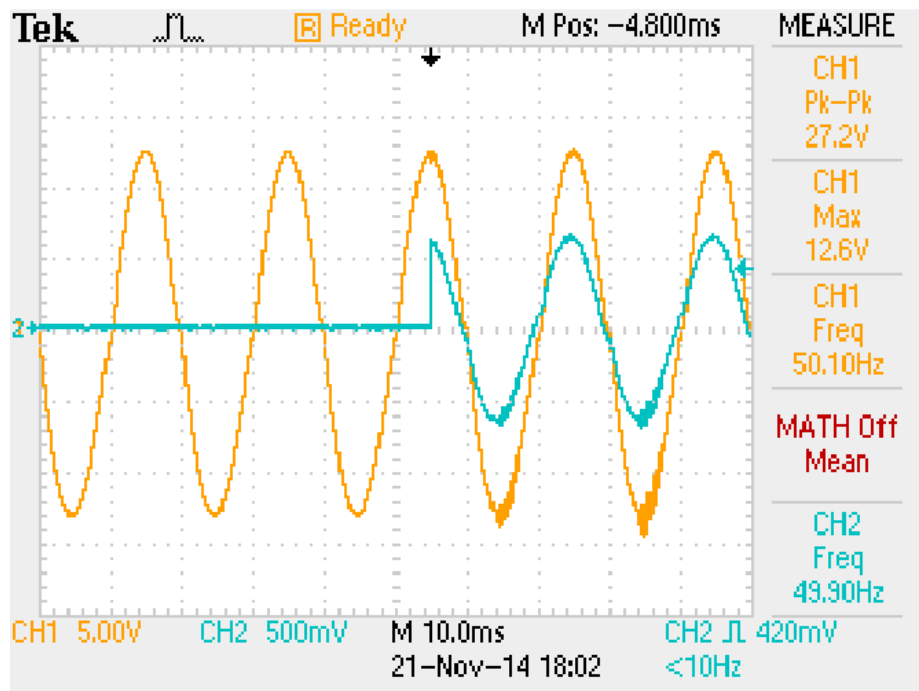
Figura 98 – Elevação da tensão na redução da carga



Fonte: produção do próprio autor

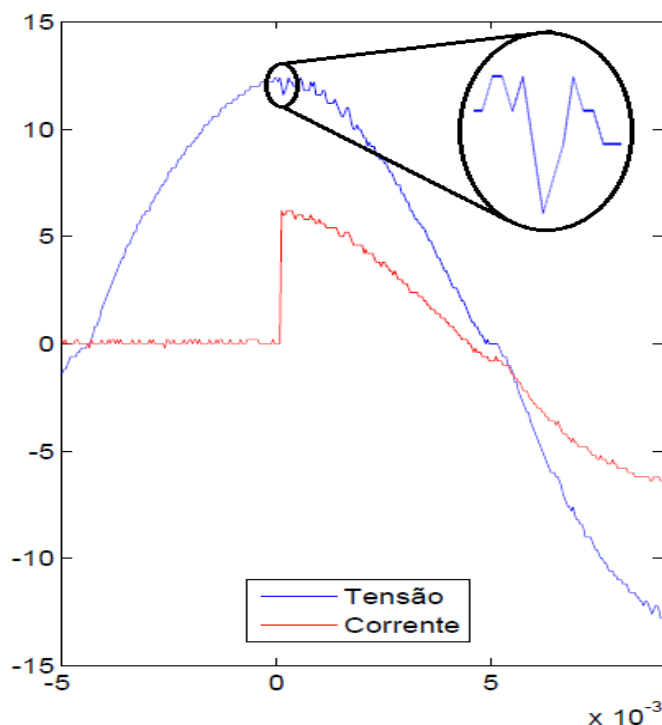
O contrário pode ser observado na Figura 99, onde ocorre um acréscimo de carga.

Figura 99– Acréscimo de carga



Fonte: produção do próprio autor

Figura 100 – Redução da tensão no acréscimo de carga

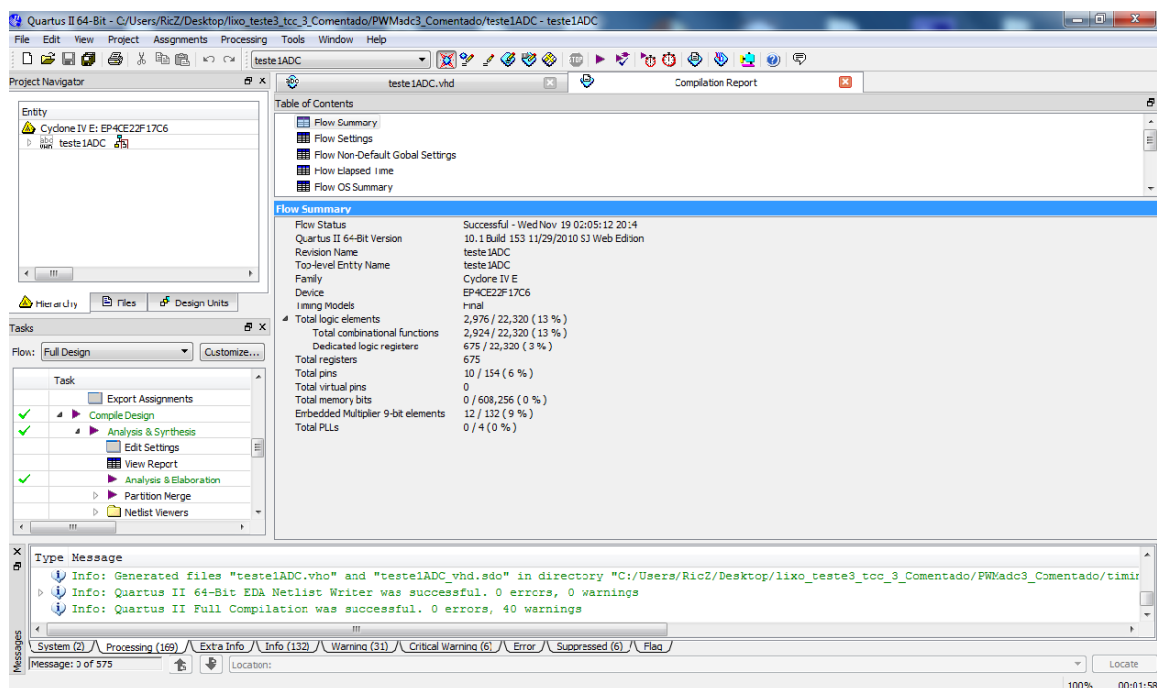


Fonte: produção do próprio autor

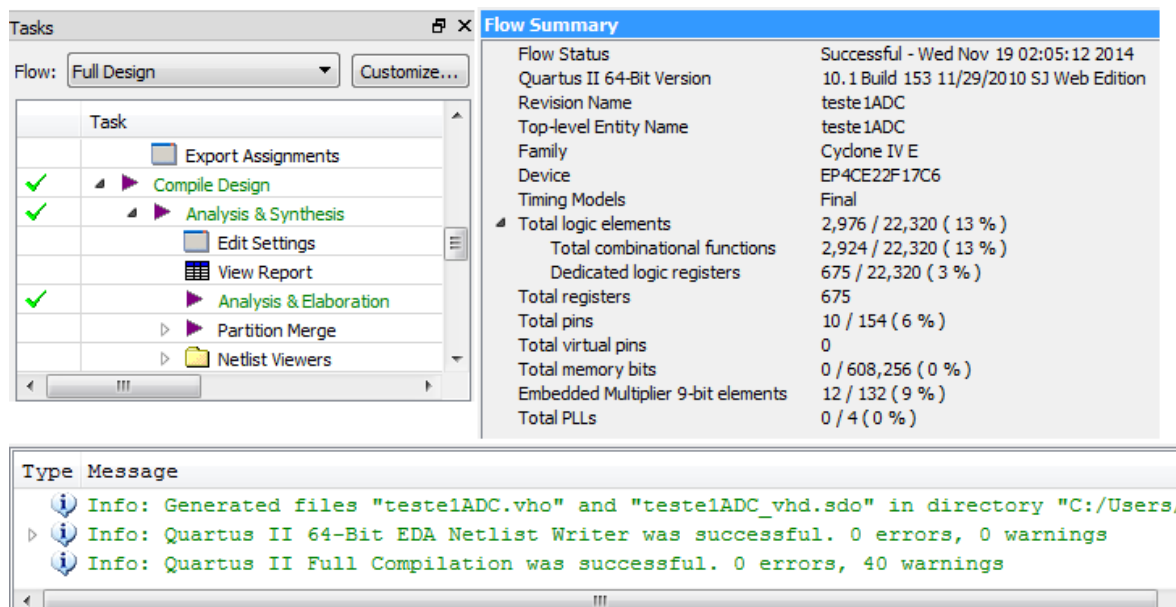
Verifica-se na Figura 100 que o acréscimo de carga provoca um afundamento na onda de tensão. Neste caso, o controlador também atua para acomodar este efeito.

7.5 RELATÓRIOS GERADOS PELO QUARTUS II®

As Figuras 101 e 102 exibem o relatório gerado pelo Quartus II®, após a compilação do código escrito em VHDL para inferir um *hardware* que implemente um controlador PID digital, a fim de controlar a tensão de saída de um conversor NPCm monofásico de três níveis. Verifica-se nestas figuras, que para inferir este *hardware* foi necessário um total de 2976 elementos lógicos, o que representa 13% de um total de 22320 LEs presentes no FPGA utilizado.

Figura 101 – Relatório referente ao *hardware* de controle projetado para o sistema monofásico

Fonte: produção do próprio autor

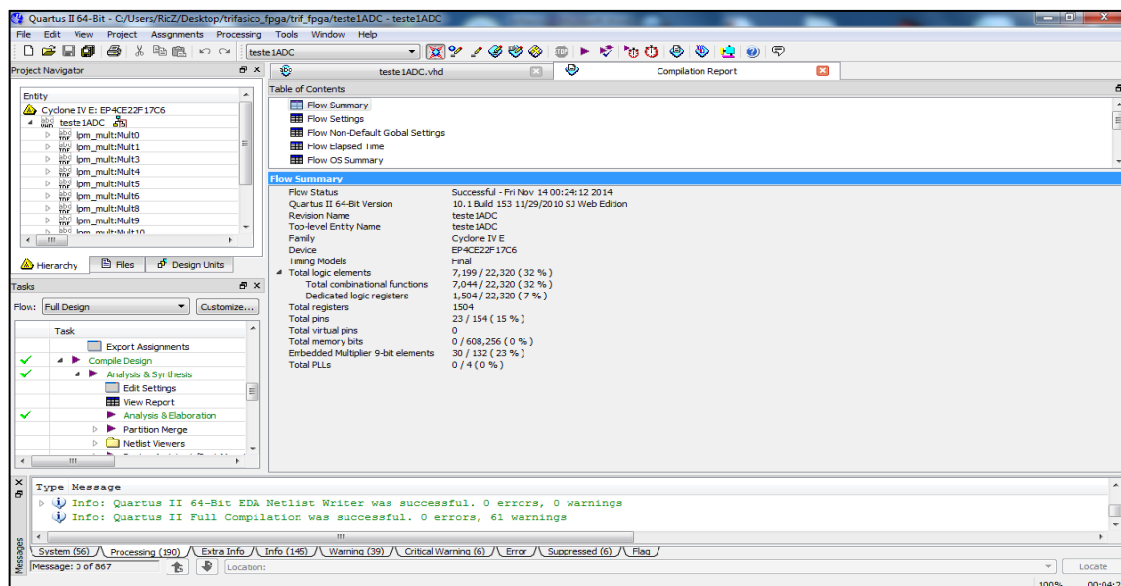
Figura 102 – Relatório referente ao *hardware* de controle projetado para o sistema monofásico (zoom)

Fonte: produção do próprio autor

O relatório gerado pelo Quartus II[®], após a compilação do código escrito em VHDL para inferir um *hardware* que implemente um controlador PID digital, a fim de controlar as tensões das saídas de um conversor NPCm trifásico de três níveis é apresentado nas Figuras

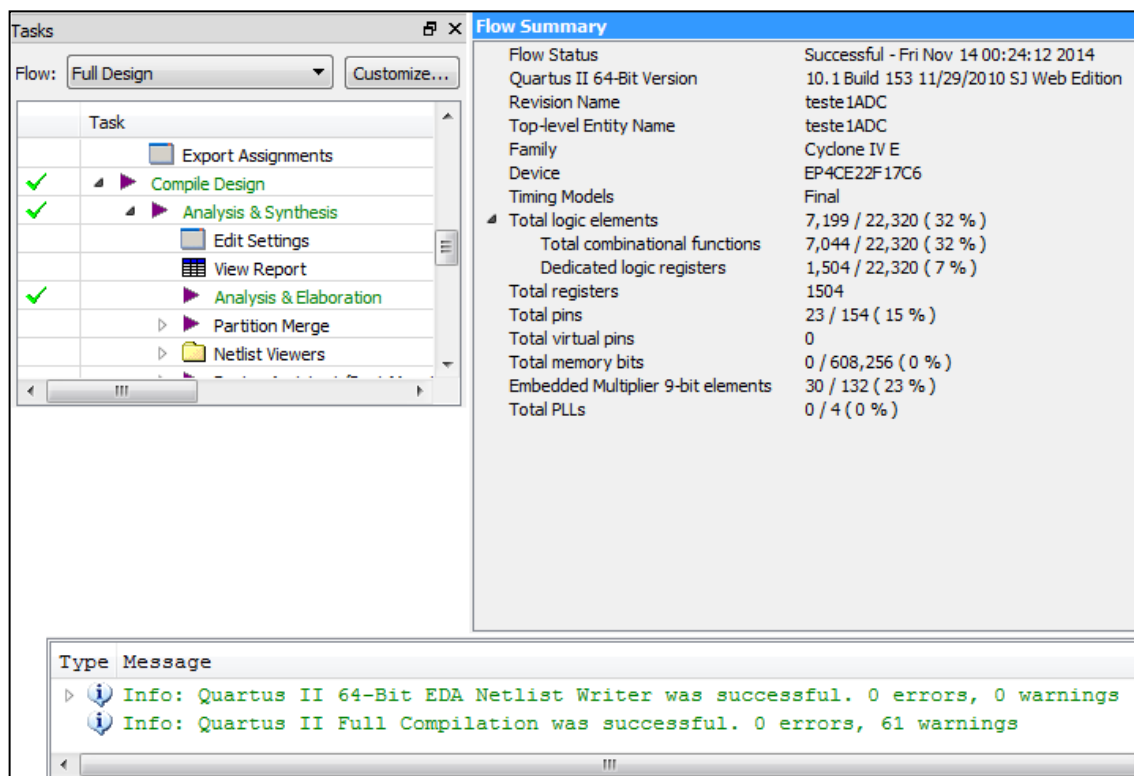
103 e 104. Nota-se que foi necessário 7199 elementos lógicos de um total de 22320 LEs para gerar este *hardware*. Isto representa 32% da capacidade do FPGA utilizado.

Figura 103 – Relatório referente ao *hardware* de controle projetado para o sistema trifásico



Fonte: produção do próprio autor

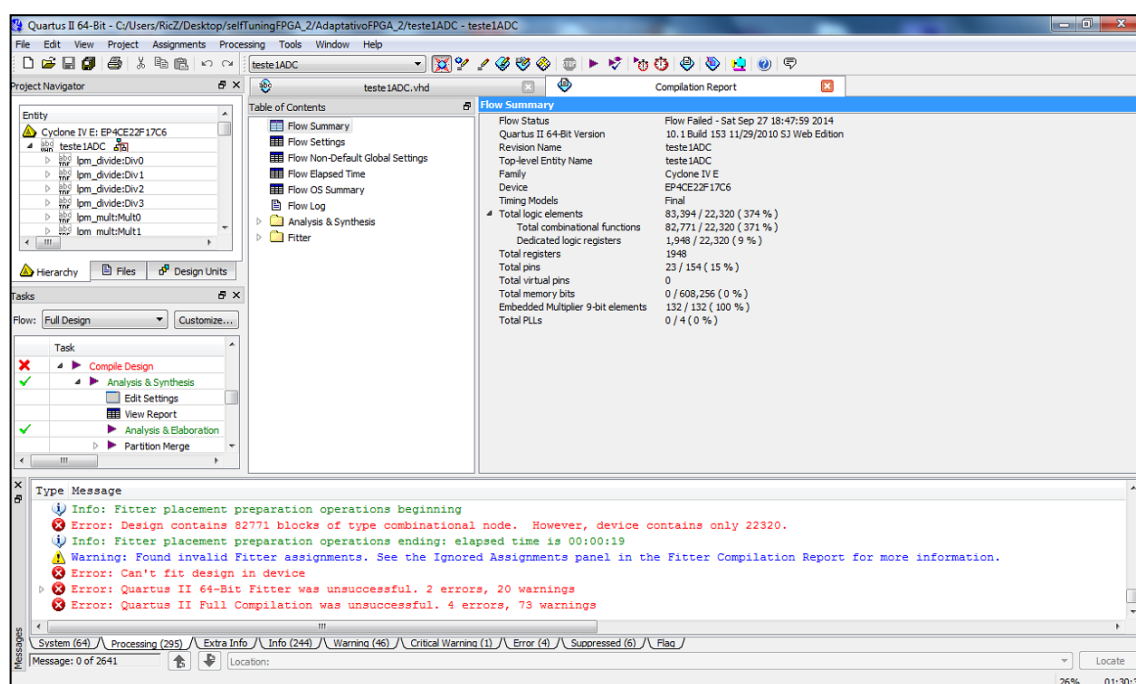
Figura 104 – Relatório referente ao *hardware* de controle projetado para o sistema trifásico (zoom)



Fonte: produção do próprio autor

O FPGA EP4CE22F17C6N da família Cyclone IV E, presente no KIT DE0 – Nano, não disponibiliza um número de elementos lógicos suficientes para a implementação de um sistema de controle *Indirect Self Tuning* para uma planta de segunda ordem, como o conversor NPCm com um filtro LC, objeto deste trabalho. As Figuras 105 e 106 apresentam o relatório do processo de compilação fornecido pelo Quartus II®.

Figura 105 – Relatório referente ao *hardware* de controle (Indirect Self Tuning) projetado para o sistema monofásico



Fonte: produção do próprio autor

Figura 106 – Relatório referente ao *hardware* de controle (Indirect Self Tuning) projetado para o sistema monofásico (zoom)

The screenshot displays the Quartus II software interface. On the left, a 'Task' pane shows the compilation process: 'Compile Design' (failed), 'Analysis & Synthesis' (successful), 'Edit Settings' (button), 'View Report' (button), 'Analysis & Elaboration' (successful), and 'Partition Merge' (button). The main area shows the 'Flow Summary' for a failed flow on Saturday, September 27, 2014, at 18:47:59. The summary lists various design metrics and their utilization percentages. Below the summary, a 'Type Message' pane shows several error and warning messages, including a critical error about exceeding the combinational logic limit of the device.

Flow Summary	
Flow Status	Flow Failed - Sat Sep 27 18:47:59 2014
Quartus II 64-Bit Version	10.1 Build 153 11/29/2010 SJ Web Edition
Revision Name	teste1ADC
Top-level Entity Name	teste1ADC
Family	Cyclone IV E
Device	EP4CE22F17C6
Timing Models	Final
Total logic elements	83,394 / 22,320 (374 %)
Total combinational functions	82,771 / 22,320 (371 %)
Dedicated logic registers	1,948 / 22,320 (9 %)
Total registers	1948
Total pins	23 / 154 (15 %)
Total virtual pins	0
Total memory bits	0 / 608,256 (0 %)
Embedded Multiplier 9-bit elements	132 / 132 (100 %)
Total PLLs	0 / 4 (0 %)

Type	Message
Info	Fitter placement preparation operations beginning
Error	Design contains 82771 blocks of type combinational node. However, device contains only 22320.
Info	Fitter placement preparation operations ending: elapsed time is 00:00:19
Warning	Found invalid Fitter assignments. See the Ignored Assignments panel in the Fitter Compilation
Error	Can't fit design in device
Error	Quartus II 64-Bit Fitter was unsuccessful. 2 errors, 20 warnings
Error	Quartus II Full Compilation was unsuccessful. 4 errors, 73 warnings

Fonte: produção do próprio autor

Nota-se que o número de elementos lógico necessários para sintetizar o *hardware* deveria ser 3,74 vezes maior do que o número existente. Observa-se que a análise e a síntese são realizadas com sucesso, demonstrando que não existem erros de escrita do código. Porém o projeto não é compilado devido as limitações do FPGA utilizado.

No intuito de verificar a demanda por elementos lógicos de uma rotina que calcula a lei de controle, foram realizados dois testes. No primeiro teste, foi escrito o código em VHDL que é apresentado na Figura 107, onde a equação a diferenças representa um controlador com dois polos e dois zeros.

Figura 107 – Rotina que calcula o sinal de controle (Controlador com 2 polos e 2 zeros)

```

67  rotinal: process (clk)
68
69      begin
70
71          if clk='1' then
72
73              senoComp := senoComp + 100;
74              ValorAtual := ValorAtual + 30;
75
76              k0_b <= to_sfixed(0.733,k0_b);
77              k1_b <= to_sfixed(0.733,k1_b);
78              k2_b <= to_sfixed(0.733,k2_b);
79              k3_b <= to_sfixed(0.733,k3_b);
80              k4_b <= to_sfixed(0.733,k4_b);
81              refComp_b <= to_sfixed(senoComp,refComp_b);
82              valatual_b <= to_sfixed(ValorAtual,valatual_b);
83
84              ek2_b <= ek1_b;
85              uk2_b <= uk1_b;
86              ek1_b <= ek_b;
87              uk1_b <= uk_b;
88
89              ek_b <= refComp_b - valatual_b;
90
91              uk_b <= resize(k0_b*uk1_b + k1_b*uk2_b + k2_b*ek_b + k3_b*ek1_b +
92 k4_b*ek2_b,uk_b);
93              --uk_b <= resize(k0_b*uk1_b + k2_b*ek_b + k3_b*ek1_b,uk_b);
94
95              if uk_b > 10 then
96                  pwm1 <= '1';
97                  pwm2 <= '1';
98                  pwm3 <= '0';
99                  pwm4 <= '0';
100             else
101                 pwm1 <= '0';
102                 pwm2 <= '0';
103                 pwm3 <= '1';
104                 pwm4 <= '1';
105             end if;
106         end if;
107     end process rotinal;
108 end process rotinal;
109
110 end ADConverter;

```

Fonte: produção do próprio autor

A demanda por elementos lógicos da rotina apresentada na Figura 107 é exibida na Figura 108.

Figura 108 – Relatório referente à rotina que calcula o sinal de controle (Controlador com 2 polos e 2 zeros)

The screenshot displays the Quartus II Flow Summary window. The 'Tasks' pane on the left shows the compilation flow: 'Full Design' (selected), 'Edit Logic Options (Open Assignm...', 'Export Assignments', 'Compile Design' (checked), 'Analysis & Synthesis' (checked), 'Edit Settings', 'View Report', 'Analysis & Elaboration' (checked), and 'Partition Merge'. The 'Flow Summary' pane on the right provides a detailed overview of the compilation process, including the flow status, version, revision name, top-level entity name, family, device, timing models, and resource usage. The 'Messages' pane at the bottom shows three informational messages: 'Generated files "teste1AIC.vho" and "teste1ADC_vhd.sdo" in directory "C:/Users/RicZ/Desktop"', 'Quartus II 64-Bit EDA Netlist Writer was successful. 0 errors, 0 warnings', and 'Quartus II Full Compilation was successful. 0 errors, 28 warnings'.

Flow Summary	
Flow Status	Successful - Mon Nov 24 15:43:01 2014
Quartus II 64-Bit Version	10.1 Build 153 11/29/2010 SJ Web Edition
Revision Name	teste1ADC
Top-level Entity Name	teste1ADC
Family	Cyclone IV E
Device	EP4CE22F17C6
Timing Models	Final
Total logic elements	942 / 22,320 (4 %)
Total combinational functions	904 / 22,320 (4 %)
Dedicated logic registers	201 / 22,320 (< 1 %)
Total registers	201
Total pins	10 / 154 (6 %)
Total virtual pins	0
Total memory bits	0 / 608,256 (0 %)
Embedded Multiplier 9-bit elements	12 / 132 (9 %)
Total PLLs	0 / 4 (0 %)

Type	Message
Info	Generated files "teste1AIC.vho" and "teste1ADC_vhd.sdo" in directory "C:/Users/RicZ/Desktop"
Info	Quartus II 64-Bit EDA Netlist Writer was successful. 0 errors, 0 warnings
Info	Quartus II Full Compilation was successful. 0 errors, 28 warnings

Fonte: produção do próprio autor

No segundo teste, foi escrito o código em VHDL que é apresentado na Figura 109, onde a equação a diferença representa um controlador com um polo e um zero. A demanda por elementos lógicos da rotina apresentada na Figura 109 é exibida na Figura 110. Entre as linhas 94 e 104 dos códigos apresentados nas Figuras 107 e 109, as saídas recebem valores simbólicos, apenas para que o *hardware* pudesse ser gerado corretamente, a fim de que a análise de demanda por LEs fosse possível.

Figura 109 – Rotina que calcula o sinal de controle (Controlador com 1 polo1 e 1 zero1)

```

67  rotinal: process (clk)
68
69      begin
70
71          if clk='1' then
72
73              senoComp := senoComp + 100;
74              ValorAtual := ValorAtual + 30;
75
76              k0_b <= to_sfixed(0.733,k0_b);
77              --k1_b <= to_sfixed(0.733,k1_b);
78              k2_b <= to_sfixed(0.733,k2_b);
79              k3_b <= to_sfixed(0.733,k3_b);
80              --k4_b <= to_sfixed(0.733,k4_b);
81              refComp_b <= to_sfixed(senoComp,refComp_b);
82              valatual_b <= to_sfixed(ValorAtual,valatual_b);
83
84              --ek2_b <= ek1_b;
85              --uk2_b <= uk1_b;
86              ek1_b <= ek_b;
87              uk1_b <= uk_b;
88
89              ek_b <= refComp_b - valatual_b;
90
91              --uk_b <= resize(k0_b*uk1_b + k1_b*uk2_b + k2_b*ek_b + k3_b*ek1_b +
92              k4_b*ek2_b,uk_b);
93              uk_b <= resize(k0_b*uk1_b + k2_b*ek_b + k3_b*ek1_b,uk_b);
94
95              if uk_b > 10 then
96                  pwm1 <= '1';
97                  pwm2 <= '1';
98                  pwm3 <= '0';
99                  pwm4 <= '0';
100             else
101                 pwm1 <= '0';
102                 pwm2 <= '0';
103                 pwm3 <= '1';
104                 pwm4 <= '1';
105             end if;
106         end if;
107     end process rotinal;
108 end process rotinal;
109
110 end ADConverter;

```

Fonte: produção do próprio autor

Figura 110 – Relatório referente à rotina que calcula o sinal de controle (Controlador com 1 polo e 1 zero)

The image shows two windows from the Quartus II software interface. The top window is the 'Flow Summary' window, which displays the status of the compilation process. The bottom window is the 'Type Message' window, which shows the output of the compilation process.

Flow Summary

Flow Status	Successful - Mon Nov 24 15:52:35 2014
Quartus II 64-Bit Version	10.1 Build 153 11/29/2010 SJ Web Edition
Revision Name	teste1ADC
Top-level Entity Name	teste1ADC
Family	Cyclone IV E
Device	EP4CE22F17C6
Timing Models	Final
Total logic elements	641 / 22,320 (3 %)
Total combinational functions	640 / 22,320 (3 %)
Dedicated logic registers	151 / 22,320 (< 1 %)
Total registers	151
Total pins	10 / 154 (6 %)
Total virtual pins	0
Total memory bits	0 / 608,256 (0 %)
Embedded Multiplier 9-bit elements	6 / 132 (5 %)
Total PLLs	0 / 4 (0 %)

Type Message

```

Info: Generated files "teste1ADC.vho" and "teste1ADC_vhd.sdo" in directory "C:/Us
Info: Quartus II 64-Bit EDA Netlist Writer was successful. 0 errors, 0 warnings
Info: Quartus II Full Compilation was successful. 0 errors, 28 warnings
  
```

Fonte: produção do próprio autor

7.6 CONSIDERAÇÕES SOBRE O CONTROLE

Embora a ação de controle possa ser observada nas Figuras 98 e 100, neste trabalho, não é apresentado ao leitor uma análise da resposta do sistema na situação em que é efetuada a adição ou diminuição do valor da carga. Isto deve-se ao fato, de não ser a proposta do trabalho, avaliar o desempenho de diferentes controladores e sim avaliar a capacidade que o FPGA tem, de receber uma arquitetura capaz de efetuar o controle de um conversor NPCm três níveis monofásico e trifásico, como já mencionado anteriormente. Neste contexto, não foram estipulados pré-requisitos precisos para o projeto do controlador, a fim de que se pudesse ter uma flexibilidade na implementação prática e se obtivesse êxito na geração de um sinal de tensão senoidal, sem grandes distorções e com uma atenuação aceitável. As simulações realizadas no Capítulo 5 serviram para trazer um embasamento teórico para a implementação prática. Entretanto, os resultados simulados e práticos não foram confrontados, pois existe a consciência de que há a necessidade de um levantamento de um modelo mais sofisticado para representar o sistema utilizado, considerando por exemplo, entre outros fatores, a quantização da portadora. Entende-se que para este projeto, um refinamento

do controlador também seria necessário, considerando a ideia de se utilizar mais de um polo na origem, por exemplo, já que o sinal de referência trata-se de uma senóide.

A forma de onda senoidal da tensão presente na carga, indica que o sinal de controle obtido a partir do cálculo da equação a diferenças escrita em VHDL, também tem forma senoidal e não apresenta grandes distorções. Este fato, comprova que o *hardware* inferido no FPGA é capaz de realizar os cálculo inerentes ao controlador projetado, que possui dois polos e dois zeros, ou seja, não se trata do modelo mais básico de controlador existente. Esta é uma conclusão importante do trabalho.

8 CONCLUSÕES GERAIS

Este trabalho tratou, no âmbito da eletrônica de potência, da capacidade de um FPGA em efetuar o controle de tensão, para aplicações em conversores NPCm de três níveis monofásico e trifásico. Avaliando a demanda por elementos lógicos de cada projeto, o tempo de resposta entre a solicitação de amostragem e a atualização do sinal de controle e de forma implícita, a capacidade de realização de cálculos inerentes a sistemas de controle.

Apresentou-se, ao início do trabalho, a arquitetura básica de um FPGA, com enfoque no modelo EP4CE22F17C6N da família Cyclone IV da fabricante Altera, a fim de ilustrar ao leitor alguns dos termos utilizados ao longo do trabalho. Em seguida, de maneira resumida, comentou-se sobre o método de programação e as etapas de projeto na utilização de um FPGA. Em sequência, foram expostas informações sobre o padrão mais recente da linguagem VHDL, o VHDL 2008. Este tópico é de fundamental importância para que o leitor compreenda os códigos disponibilizados nos Apêndices A, B e C e conheça as novas possibilidades que esta versão traz. Uma explanação sobre controladores PID foi apresentada para proporcionar fundamentação teórica ao controlador que seria projetado mais adiante. Com o objetivo de fundamentar o capítulo que tratou da modelagem, foi apresentado a estrutura básica e as etapas de operação de um conversor NPCm de três níveis monofásico. Apresentaram-se o modelo do conversor utilizado, acoplado a um filtro LC, e o projeto do controlador digital realizado por alocação de polos e zeros no plano Z. Foi mostrado a comprovação do modelo matemático, através da comparação dos resultados da simulação do modelo comutado no *software* PSIM[®] e da simulação feita no Simulink do modelo matemático. Uma contribuição importante deste trabalho é a demonstração do uso do conversor A/D, do kit de desenvolvimento DE0 – Nano da fabricante Altera, que foi apresentado na seção 6.1. Também foram apresentados os métodos de quantização utilizados nos projetos, necessários na abordagem por ponto fixo do VHDL 2008. Com a finalidade de verificar a capacidade do FPGA utilizado, em receber uma arquitetura de controle, foi implementado um controlador adaptativo do tipo *Indirect Self Tuning*, pois a aplicação deste método demanda uma quantidade relativamente grande de operações matemáticas e consequentemente uma quantidade grande de LEs.

Os resultados demonstraram que o FPGA EP4CE22F17C6N da família Cyclone IV da fabricante Altera, apesar de ser um modelo classificado pela fabricante como “*Low-Cost FPGAs*”, tem condições de receber uma arquitetura capaz de realizar um controle PID de um conversor NPCm de três níveis monofásico e trifásico. Sendo que o sistema trifásico utilizou

cerca de um terço do número de LEs disponíveis no FPGA. Verificou-se também que a principal limitação na velocidade de resposta entre o instante da amostragem e a atualização do sinal de controle está na comunicação serial entre o conversor A/D e o FPGA presentes no kit DE0 – Nano. No caso do controle adaptativo, o FPGA utilizado não foi capaz de atender a necessidade imposta pelo método *Indirect Self Tuning* para controlar o conversor NPCm de três níveis monofásico. Entretanto, o FPGA EP4CE115 também da família Cyclone IV da mesma fabricante e também classificada como “Low-Cost FPGAs”, teria condições de receber a arquitetura projetada. Os FPGAs da família Stratix IV E, classificados pela Altera como “High-End FPGAs” podem conter até 813050 LEs, ou seja, quase dez vezes mais do que a necessidade do *hardware* que implementa o método *Indirect Self Tuning*.

O presente trabalho deixa como contribuição os códigos apresentados nos Apêndices, referentes as implementações dos controladores PID digital e adaptativo para aplicação em conversores NPCm de três níveis monofásico e trifásico. Também foi disponibilizado o estudo e o código para o uso do conversor A/D do kit DE0-Nano, que pode ser útil na disciplina de DLP (Dispositivos Lógicos Programáveis). E por fim, exemplos da utilização do VHDL 2008.

Por fim destaca-se, que os projetos realizados neste trabalho foram de grande importância para uma compreensão mais profunda de diversos conteúdos apresentados ao longo da graduação e dos *softwares* MATLAB[®], PSIM[®] e Quartus II[®].

Como sugestões para trabalhos futuros têm-se:

- Otimizar os códigos em VHDL, estudando diferentes métodos de projeto;
- Inserir nos códigos em VHDL rotinas para proteção;
- Obter um modelo que represente com maior precisão o sistema utilizado, como por exemplo, considerar a quantização da portadora e os atrasos;
- Melhorar o projeto do controlador, levando em consideração que o sinal de referência trata-se de uma senóide;
- Reescrever os códigos com a abordagem em ponto flutuante;
- Melhorar o *layout* da placa de condicionamento do sinal de saída;
- Testar o código que implementa o método *Indirect Self Tuning* em um FPGA que atenda a necessidade do projeto;

REFERÊNCIAS

- [1] CÂMARA, R. A. **Análise Comparativa de Desempenho de Conversores CA-CC Monofásicos Utilizando FPGA para Aplicação em No-Breaks**. Exame de Qualificação para Doutorado. UFC. Fortaleza 2012.

- [2] PEDRONI, V. **Eletrônica Digital Moderna com VHDL**. 1. ed. Elsevier, 2010.

- [3] ALTERA, **Cyclone IV Device Handbook**, 2013. Disponível em: <<http://www.altera.com/literature/hb/cyclone-iv/cyclone4-handbook.pdf>> Acesso em: 20 jan. 2014.

- [4] ALTERA, **Quartus II Handbook**, 2014. Disponível em: <http://www.altera.com/literature/hb/qts/quartusii_handbook.pdf> Acesso em: 20 ago. 2014.

- [5] BROWN, S.; VRANESIC, Z. **Fundamentals of Digital Logic with VHDL Design**. 3. ed. New York: McGraw-Hill, 2009.

- [6] VEIGA, H. W. **Programação VHDL de Dispositivo FPGA para Aplicação em Modulação PWM Senoidal de Conversores Multiníveis**. Monografia de graduação em engenharia elétrica. UDESC. Joinville 2013.

- [7] TERASIC. **DE0-nano User manual**, 2014. Disponível em: <<http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=139&No=593&PartNo=4>> Acesso em: 10 ago. 2014.

- [8] OLIVEIRA, A. A. **Estudo e Implementação de Operações em Ponto Fixo em FPGA com VHDL 2008: Aplicação em Controle de Sistemas em Tempo Discreto**. Dissertação (Mestrado). UTFPR. Curitiba 2012.

- [9] BISHOP, D. **Fixed point package user's guide**. Disponível em: <http://www.vhdl.org/fphdl/Fixed_ug.pdf> Acesso em: 10 ago. 2014.

- [10] HEMERLY, E. M. **Controle por Computador de Sistemas Dinâmicos**. 2. ed. São Paulo: Edgard Blücher Ltda., 2000.
- [11] LAGES, W. F. **Controladores PID**. 2010. 22p. Notas de aula.
- [12] HEERDT, J. A. **Carga Eletrônica Ativa Trifásica**. Exame de Qualificação para Doutorado. UFSC. Florianópolis 2011.
- [13] BATSCHAUER, A. L. **Apostila da Disciplina de Conversores Estáticos**. UDESC. Joinville. 2012.
- [14] BARBI, Ivo. **Eletrônica de potência**. 3. ed. Florianópolis: UFSC, 2002.
- [15] ERICKSON, R. W.; Maksimovic, D. **Fundamentals of Power Electronics**. 1. ed. New York: Kluwer Academic, 2001
- [16] LAGES, W. F. **Projeto de Controladores Digitais no Plano Z Utilizando Lugar das Raízes**. 2011. 11p. Notas de aula.
- [17] OGATA, K.. **Engenharia de Controle Moderno**. 5. ed. São Paulo: Pearson, 2010.
- [18] BUSO, S.; MATTAVELLI, P. **Digital Control in Power Electronics**. 1. ed. 2006.
- [19] TEXAS INSTRUMENTS. **Data sheet: ADC128S022**. [S.I], 2013. Disponível em: <<http://www.ti.com/lit/ds/symlink/adc128s022.pdf>> Acesso em: 10 mai. 2014.
- [20] ASTROM, K. J.; WITTENMARK, B. **Adaptive Control**. 2 ed. New York: Addison-Wesley Publishing Company, 1995.
- [21] MATOS, S. D dos. **Controle da Dinâmica de um Elevador Didático**. Monografia de graduação em engenharia elétrica. UDESC. Joinville 2013.
- [22] LIM, C.M. **Experimental Evaluation of a Self-Tuning Controller**. IEEE Trans. on Industrial Electronics, vol. 37, no. 3, 1990.

[23] DIESEL, J. A. D. **Desenvolvimento de um Driver para Acionamento de Transistor JFET de Carboneto de Silício**. Estágio curricular desenvolvido na UDESC, com ênfase em eletrônica de potência. UDESC. Joinville 2012.

APÊNDICE A – CÓDIGO EM VHDL QUE IMPLEMENTA O CONTROLADOR PID PARA O NPC_m MONOFÁSICO

[illegible]

```

37  shared variable a5: integer :=1;--auxiliar p/ o clock do ADC.
38  shared variable K: integer :=0;--utilizado no contador do Divisor de
39      --Frequencia p/ obter o clock do ADC.
40  shared variable Ten: integer :=0;--auxiliar no "process" do ADC.
41  signal aux10: STD_LOGIC;--Atua simultaneamente ao clock do ADC. Habilita
42      --a rotina que obtem os bits da conversao.
43  signal LiberaCalculo: STD_LOGIC;--Libera o clculo da lei de controle.
44
45  signal k0_b : sfixed(0 downto -17);--recebe o coeficiente a1.
46  signal k1_b : sfixed(0 downto -17);--recebe o coeficiente a2.
47  signal k2_b : sfixed(0 downto -17);--recebe o coeficiente b0.
48  signal k3_b : sfixed(0 downto -17);--recebe o coeficiente b1.
49  signal k4_b : sfixed(0 downto -17);--recebe o coeficienteb2.
50
51  signal refComp_b : sfixed(11 downto 0);--recebe o vetor senoide(+/-2047).
52  signal valatual_b : sfixed(11 downto 0);--recebe saida convertida(+/-2047).
53  signal ek_b : sfixed(12 downto 0);--recebe o erro;malha fechada(+/-4095).
54  signal ek1_b : sfixed(12 downto 0);--armazena o erro e(k-1).
55  signal ek2_b : sfixed(12 downto 0);--armazena o erro e(k-2).
56  signal uk_b : sfixed(18 downto -18);--sinal de controle.
57  signal uk1_b : sfixed(18 downto -18);--sinal de controle no instante u(k-1).
58  signal uk2_b : sfixed(18 downto -18);--sinal de controle no instante u(k-2).
59
60  signal saida_b : sfixed(18 downto -18);--recebe o sinal de controle.
61
62  shared variable a : integer :=0;--variavel p/contagem subidas do clock.
63  shared variable b : integer :=0;--variavel p/contagem subidas do clock.
64  shared variable a1 : integer :=0;--variavel p/contagem subidas do clock.
65  shared variable a2 : integer :=0;--variavel p/contagem subidas do clock.
66  shared variable a3 : integer :=0;--variavel p/contagem subidas do clock.
67  shared variable a4 : integer :=0;--variavel p/contagem subidas do clock.
68  shared variable count1 : integer :=1;--variavel que aponta o indice do vetor.
69  shared variable count2 : integer :=1;--variavel que aponta o indice do vetor.
70  shared variable contADC: integer :=0;--auxiliar no "process" responsavel pelo ADC.
71  shared variable hab_ADC: integer :=1;--auxiliar no "process" responsavel pelo ADC.
72

```

```

73  type triangular is array (integer range <>) of integer;--declaracao do vetor da portadora.
74                                     --625 amostras entre 0 e 1996.
75
76  shared variable triangular1:triangular (1 to 625) :=
77  (6,12,20,26,32,38,44,52,58,64,70,76,84,90,96,102,108,116,122,128,134,140,148,154,160,166,
78  172,180,186,192,198,204,212,218,224,230,236,244,250,256,262,268,276,282,288,294,
79  300,308,314,320,326,332,340,346,352,358,364,372,378,384,390,396,404,410,416,
80  422,428,436,442,448,454,460,468,474,480,486,492,500,506,512,518,524,532,538,
81  544,550,556,564,570,576,582,588,596,602,608,614,620,628,634,640,646,652,660,
82  666,672,678,684,692,698,704,710,716,724,730,736,742,748,756,762,768,774,780,
83  788,794,800,806,812,820,826,832,838,844,852,858,864,870,876,884,890,896,902,
84  908,916,922,928,934,940,948,954,960,966,972,980,986,992,998,1004,1012,1018,1024,
85  1030,1036,1044,1050,1056,1062,1068,1076,1082,1088,1094,1100,1108,1114,1120,1126,1132,1140,1146,
86  1152,1158,1164,1172,1178,1184,1190,1196,1204,1210,1216,1222,1228,1236,1242,1248,1254,1260,1268,
87  1274,1280,1286,1292,1300,1306,1312,1318,1324,1332,1338,1344,1350,1356,1364,1370,1376,1382,1388,
88  1396,1402,1408,1414,1420,1428,1434,1440,1446,1452,1460,1466,1472,1478,1484,1492,1498,1504,1510,
89  1516,1524,1530,1536,1542,1548,1556,1562,1568,1574,1580,1588,1594,1600,1606,1612,1620,1626,1632,
90  1638,1644,1652,1658,1664,1670,1676,1684,1690,1696,1702,1708,1716,1722,1728,1734,1740,1748,1754,
91  1760,1766,1772,1780,1786,1792,1798,1804,1812,1818,1824,1830,1836,1844,1850,1856,1862,1868,1876,
92  1882,1888,1894,1900,1908,1914,1920,1926,1932,1940,1946,1952,1958,1964,1972,1978,1984,1990,1996,
93  1996,1990,1984,1978,1972,1964,1958,1952,1946,1940,1932,1926,1920,1914,1908,1900,1894,1888,1882,
94  1876,1868,1862,1856,1850,1844,1836,1830,1824,1818,1812,1804,1798,1792,1786,1780,1772,1766,1760,
95  1754,1748,1740,1734,1728,1722,1716,1708,1702,1696,1690,1684,1676,1670,1664,1658,1652,1644,1638,
96  1632,1626,1620,1612,1606,1600,1594,1588,1580,1574,1568,1562,1556,1548,1542,1536,1530,1524,1516,
97  1510,1504,1498,1492,1484,1478,1472,1466,1460,1452,1446,1440,1434,1428,1420,1414,1408,1402,1396,
98  1388,1382,1376,1370,1364,1356,1350,1344,1338,1332,1324,1318,1312,1306,1300,1292,1286,1280,1274,
99  1268,1260,1254,1248,1242,1236,1228,1222,1216,1210,1204,1196,1190,1184,1178,1172,1164,1158,1152,
100  1146,1140,1132,1126,1120,1114,1108,1100,1094,1088,1082,1076,1068,1062,1056,1050,1044,1036,1030,
101  1024,1018,1012,1004,998,992,986,980,972,966,960,954,948,940,934,928,922,916,908,
102  902,896,890,884,876,870,864,858,852,844,838,832,826,820,812,806,800,794,788,
103  780,774,768,762,756,748,742,736,730,724,716,710,704,698,692,684,678,672,666,
104  660,652,646,640,634,628,620,614,608,602,596,588,582,576,570,564,556,550,544,
105  538,532,524,518,512,506,500,492,486,480,474,468,460,454,448,442,436,428,422,
106  416,410,404,396,390,384,378,372,364,358,352,346,340,332,326,320,314,308,300,
107  294,288,282,276,268,262,256,250,244,236,230,224,218,212,204,198,192,186,180,172,166,
108  160,154,148,140,134,128,122,116,108,102,96,90,84,76,70,64,58,52,44,38,32,26,20,12,6,0);

```

```

109
110 type senoide is array (integer range <>) of integer;--declaracao do vetor do sinal modulante.
111                                     --625 amostras entre 0 e 1900. Indice de
112                                     --modulacao de amplitude igual a 0,95.
113
114 shared variable senoidel:senoide (1 to 625):=
115 (20,38,58,76,96,114,134,152,172,190,210,228,248,266,286,304,324,342,360,380,398,
116 416,436,454,472,490,510,528,546,564,582,600,618,636,654,672,690,708,726,744,
117 762,778,796,814,830,848,864,882,898,916,932,948,966,982,998,1014,1030,1046,1062,
118 1078,1094,1110,1124,1140,1156,1170,1186,1200,1214,1230,1244,1258,1272,1286,1300,1314,1328,1342,
119 1356,1368,1382,1394,1408,1420,1434,1446,1458,1470,1482,1494,1506,1518,1528,1540,1552,1562,1572,
120 1584,1594,1604,1614,1624,1634,1644,1654,1662,1672,1680,1690,1698,1706,1716,1724,1732,1738,1746,
121 1754,1762,1768,1776,1782,1788,1794,1802,1808,1812,1818,1824,1830,1834,1840,1844,1848,1852,1856,
122 1860,1864,1868,1872,1874,1878,1880,1884,1886,1888,1890,1892,1894,1894,1896,1898,1898,1898,1900,
123 1900,1900,1900,1900,1900,1898,1898,1896,1896,1894,1892,1890,1888,1886,1884,1882,1880,1876,1874,
124 1870,1866,1862,1858,1854,1850,1846,1842,1836,1832,1826,1822,1816,1810,1804,1798,1792,1786,1778,
125 1772,1764,1758,1750,1742,1736,1728,1720,1710,1702,1694,1686,1676,1668,1658,1648,1640,1630,1620,
126 1610,1600,1588,1578,1568,1556,1546,1534,1522,1512,1500,1488,1476,1464,1452,1440,1426,1414,1402,
127 1388,1376,1362,1348,1336,1322,1308,1294,1280,1266,1252,1236,1222,1208,1192,1178,1162,1148,1132,
128 1116,1102,1086,1070,1054,1038,1022,1006,990,974,956,940,924,906,890,874,856,840,822,
129 804,788,770,752,734,718,700,682,664,646,628,610,592,574,556,536,518,500,482,
130 464,444,426,408,388,370,352,332,314,294,276,258,238,220,200,182,162,144,124,106,86,
131 66,48,29,10,-10,-29,-48,-66,-86,-104,-124,-144,-162,-182,-200,-220,-238,-258,-276,-294,
132 -314,-332,-352,-370,-388,-408,-426,-444,-464,-482,-500,-518,-536,-556,-574,-592,
133 -610,-628,-646,-664,-682,-700,-718,-734,-752,-770,-788,-804,-822,-840,-856,-874,
134 -890,-906,-924,-940,-956,-974,-990,-1006,-1022,-1038,-1054,-1070,-1086,-1102,-1116,-1132,
135 -1148,-1162,-1178,-1192,-1208,-1222,-1236,-1252,-1266,-1280,-1294,-1308,-1322,-1336,-1348,-1362,
136 -1376,-1388,-1402,-1414,-1426,-1440,-1452,-1464,-1476,-1488,-1500,-1512,-1522,-1534,-1546,-1556,
137 -1568,-1578,-1588,-1600,-1610,-1620,-1630,-1640,-1648,-1658,-1668,-1676,-1686,-1694,-1702,-1710,
138 -1720,-1728,-1736,-1742,-1750,-1758,-1764,-1772,-1778,-1786,-1792,-1798,-1804,-1810,-1816,-1822,
139 -1826,-1832,-1836,-1842,-1846,-1850,-1854,-1858,-1862,-1866,-1870,-1874,-1876,-1880,-1882,-1884,
140 -1886,-1888,-1890,-1892,-1894,-1896,-1896,-1898,-1898,-1900,-1900,-1900,-1900,-1900,-1900,-1898,
141 -1898,-1898,-1896,-1894,-1894,-1892,-1890,-1888,-1886,-1884,-1880,-1878,-1874,-1872,-1868,-1864,
142 -1860,-1856,-1852,-1848,-1844,-1840,-1834,-1830,-1824,-1818,-1812,-1808,-1802,-1794,-1788,-1782,
143 -1776,-1768,-1762,-1754,-1746,-1738,-1732,-1724,-1716,-1706,-1698,-1690,-1680,-1672,-1662,-1654,
144 -1644,-1634,-1624,-1614,-1604,-1594,-1584,-1572,-1562,-1552,-1540,-1528,-1518,-1506,-1494,-1482,

```

```

145  -1470,-1458,-1446,-1434,-1420,-1408,-1394,-1382,-1368,-1356,-1342,-1328,-1314,-1300,-1286,-1272,
146  -1258,-1244,-1230,-1214,-1200,-1186,-1170,-1156,-1140,-1124,-1110,-1094,-1078,-1062,-1046,-1030,
147  -1014,-998,-982,-966,-948,-932,-916,-898,-882,-864,-848,-830,-814,-796,-778,-762,
148  -744,-726,-708,-690,-672,-654,-636,-618,-600,-582,-564,-546,-528,-510,-490,-472,
149  -454,-436,-416,-398,-380,-360,-342,-324,-304,-286,-266,-248,-228,-210,-190,-172,
150  -152,-134,-114,-96,-76,-58,-38,-20,0);
151
152  shared variable saidat1 : integer:=0;--recebe o vetor da portadora positiva (POD).
153  shared variable saidat2 : integer:=0;--recebe o vetor da portadora negativa (POD).
154  shared variable ptempomorto : integer :=5;--tempo morto em percentual(5%).
155  shared variable freqsen : integer :=50;--frequencia do sinal modulante (50Hz).
156  shared variable freqtri : integer :=40000;--frequencia das portadoras (40KHz).
157  shared variable N1: integer :=1;--variavel dependente da frequencia do sinal modulante.
158  shared variable N2: integer :=1;--variavel dependente da frequencia da portadora.
159  shared variable N3: integer :=1;--variavel dependente do percentual do tempo morto.
160  shared variable senoComp: integer :=0;--recebe o valor do sinal de referencia no
161      --instante da amostragem.
162  signal aux3: bit;--Sinal utilizado p/ habilitar o "process" responsavel
163      --pelo ADC.
164
165
166  --*****
167  --*****
168  begin
169
170  rotinal: process (clk)
171
172      begin
173
174          if clk='1' then
175
176              a:=1+a;--Incrementa 1 na variavel a.
177              b:=1+b;--Incrementa 1 na variavel b.
178              N1:=50000000/(freqsen*625);--Define a frequencia em que a
179                  --variavel "saida" sera atualizada.
180              N2:=50000000/(freqtri*625);--Define a frequencia em que as variaveis

```

```

181                                     --"saidat1 e saidat2" serao atualizadas.
182 N3:=(500000000 *ptempomorto)/(freqtri*100);
183
184 if a=N1 then
185
186     saida_b <= uk_b;--Atribui a variavel "saida_b" o valor do
187                     --sinal de controle "uk_b".
188     count1:=1+count1;--Incrementa 1 na variavel "count1".
189     if count1>625 then--Limita count1 em 625
190                     --(tamanho do vetor "senoidel")
191         count1:=1;--Atribui 1 a count1, pois o indice do vetor
192                     --inicia em 1.
193     end if;
194     a:=0;--zera a variavel "a" p/ reiniciar o processo.
195 end if;
196 if b=N2 then
197     saidat1 := triangular1(count2);--Gera internamente a portadora
198                                     --(triangular positiva).
199     saidat2 :=-triangular1(count2);--Gera internamente a portadora
200                                     --(triangular negativa).
201     count2:=1+count2;--Incrementa 1 na variavel "count2".
202     b:=0;--zera a variavel "b" p/ reiniciar o processo.
203
204     if saidat1 = 0 then -- alteracao para verificar o
205                         --momento da amostragem
206         contTS := contTS + 1;
207         if contTS = 4 then
208             aux3<='1';
209             contTS := 0;
210         end if;
211     end if;
212
213     if (saidat1 = 1996) then
214         aux3<='0';
215     end if;
216

```

```

217         if count2>625 then--Limita count2 em 625
218             --(tamanho do vetor "triangular1").
219             count2:=1;--Atribui 1 a count2, pois o indice do
220                 --vetor inicia em 1.
221         end if;
222     end if;
223
224     if saida_b > saidat1 then--Comparacao entre o sinal de
225         --controle e a portadora positiva
226         pwm2<='0';--Coloca em nivel logico baixo a saida "pwm2"
227         a1:=1+a1;--Incrementa 1 na variavel "a1" a cada pulso de
228             --clock, estabelecendo um contador.
229         if a1=N3 then--Logica utilizada p/ gerar o tempo morto
230             --entre as chaves.
231             pwm1<='1';--Coloca em nivel logico alto a saida "pwm1"
232             a1:=0;
233             a2:=0;
234         end if;
235     else
236         pwm1<='0';--Coloca em nivel logico baixo a saida "pwm1"
237         a2:=1+a2;--Incrementa 1 na variavel "a2" a cada pulso de
238             --clock, estabelecendo um contador.
239         if a2=N3 then--Logica utilizada p/ gerar o tempo morto
240             --entre as chaves.
241             pwm2<='1';--Coloca em nivel logico alto a saida "pwm2"
242             a2:=0;
243             a1:=0;
244         end if;
245     end if;
246
247
248     if saida_b > saidat2 then--Comparacao entre o sinal
249         --de controle e a portadora negativa
250         pwm4<='0';--Coloca em nivel logico baixo a saida "pwm4"
251         a3:=1+a3;--Incrementa 1 na variavel "a3" a cada pulso de
252             --clock, estabelecendo um contador.

```

```

253         if a3=N3 then--Logica utilizada p/ gerar o tempo morto
254             --entre as chaves.
255             pwm3<='1';--Coloca em nivel logico alto a saida "pwm3"
256             a3:=0;
257             a4:=0;
258         end if;
259     else
260         pwm3<='0';--Coloca em nivel logico baixo a saida "pwm3"
261         a4:=1+a4;--Incrementa 1 na variavel "a4" a cada pulso de
262             --clock, estabelecendo um contador.
263         if a4=N3 then--Logica utilizada p/ gerar o tempo morto
264             --entre as chaves.
265             pwm4<='1';--Coloca em nivel logico alto a saida "pwm4"
266             a3:=0;
267             a4:=0;
268         end if;
269     end if;
270
271     end if;
272
273 end process rotinal1;
274
275
276 rotina2: process (DOUT, clk)
277
278 begin
279
280     if aux3 = '1' then--aux3 vai p/ nivel alto no momento da amostragem.
281
282         confAD <= '0';--Utilizado na configuracao do canal 0 do ADC.
283
284         if (clk'event and clk = '1')then
285
286             if Ten = 1 then--Condicao p/ carregamento das
287                 --condicoes iniciais p/ uso do ADC.
288                 CS <= '1';--Chip Select (Desabilita o Conversor A/D).

```

```

289         aux10 <= '1';--Atua simultaneamente ao clock do ADC.
290             --Condicao inicial p/ uso do ADC.
291         SCLK <= '1';--Clock do Conversor A/D (Coloca nivel alto
292             --na saida conectada ao pino SCLK do ADC).
293         Ten:=0;--Desabilita a logica que promove o carregamento
294             --das condicoes iniciais p/ uso do ADC.
295     end if;
296
297     K:=K+1;--Contador utilizado no Divisor de Frequencia p/
298         --obter o clock do ADC.
299
300     if K=N then--Divisor de frequencia (50e6/10 = 5e6)
301
302         if hab_ADC = 1 then--Condicao p/ que o ADC seja habilitado.
303             CS <= '0';--Chip Select (Habilita o Conversor A/D).
304         end if;
305
306         if a5 /= 1 then-- Gera a onda quadrada p/ enviar ao ADC.
307
308             a5:= 1;
309             SCLK <= '1';--Clock do Conversor A/D (Coloca nivel alto na
310                 --saida conectada ao pino SCLK do ADC).
311             aux10 <= '1';--Atua simultaneamente ao clock do ADC. Habilita
312                 --a rotina que obtem os bits da conversao.
313
314         else a5:= 0;
315
316             SCLK <= '0';--Clock do Conversor A/D (Coloca nivel baixo na
317                 --saida conectada ao pino SCLK do ADC).
318             aux10 <= '0';--Atua simultaneamente ao clock do ADC. Habilita
319                 --a rotina que obtem os bits da conversao.
320
321             contADC := contADC + 1;
322             if contADC = 17 then --Permite que o ADC seja desabilitado
323                 --apos a obtencao do frame.
324                 CS <= '1';--Chip Select (Desabilita o Conversor A/D).

```

```

325             hab_ADC := 0;--Bloqueia a logica que permite que o ADC
326                 --seja habilitado.
327             end if;
328         end if;
329         K:=0;--Estabelece condicao inicial p/ a proxima amostragem
330     end if;
331
332
333     end if;
334
335     else
336
337         CS <= '1';--Chip Select (Desabilita o Conversor A/D).
338         Ten:=1;--Habilita a logica que promove o carregamento
339             --das condicoes iniciais p/ uso do ADC.
340         K:=0;--Estabelece condicao inicial p/ a proxima amostragem
341         a5:= 1;--Estabelece condicao inicial p/ a proxima amostragem
342         contADC := 0;--Estabelece condicao inicial p/ a proxima amostragem
343         hab_ADC := 1;--Estabelece condicao inicial p/ a proxima amostragem
344
345     end if;
346 end process rotina2;
347
348 rotina3: process (DOUT,aux10)
349
350 begin
351
352     if aux3 = '1' then--sinal eh levado a nivel alto no momento da amostragem.
353
354         if rising_edge (aux10) then--Executado na borda se subida de aux10.
355
356             if (cont10 >= 5)and(cont10 < 17) then--Permite a obtencao dos bits da
357                 --conversao que sao apresentados no pino
358                 --DOUT apos a quinta borda de descida do
359                 --SCLK. O carregamento eh realizado na
360                 --quinta borda de subida de SCLK.

```

```

361
362     if cont10 = 5 then--Permite salvar o valor do sinal de referencia
363         --no instante da amostragem p/ comparacao posterior.
364         senoComp := senoidel(count1);--Carrega o valor da referencia na
365         -- variavel "senoComp".
366     end if;
367
368     if DOUT = '1' then--Permite que "ValorConv" seja incrementada
369         --sempre que um nivel alto eh apresentado em DOUT.
370         ValorConv:= ValorConv + peso;--Atribui a "ValorConv" o valor passado
371         --de "ValorConv" somado ao valor de peso.
372         peso:= peso/2;--Divide a variavel "peso" a cada borda de subida de aux10.
373     else
374         peso:= peso/2;--Divide a variavel "peso" a cada borda de subida de aux10.
375     end if;
376 end if;
377
378     if cont10 = 2 then--Condicao p/ inicializar variaveis.
379         ValorConv:= 0;--Estabelece condicao inicial p/ a proxima amostragem.
380         peso:= 2048;--Estabelece condicao inicial p/ a proxima amostragem.
381
382     end if;
383
384 end if;
385
386 if falling_edge (aux10) then--Executado na borda de descida de aux10.
387
388     cont10:= cont10 + 1;--contador que controla a comunicacao serial entre
389     --ADC e FPGA.
390
391     if cont10 = 17 then--Permite o carregamento da variavel
392         --"ValorAtual" apos a obtencao do frame.
393         ValorAtual:=(ValorConv - 1900);--Elimina o offset adquirido no
394         --circuito de condicionamento.
395         LiberaCalculo <= '1';--Estabelece um nivel alto no sinal "LiberaCalculo".
396

```

```

397         else
398             ValorAtual:= ValorAtual;--Enquanto a convesao nao eh finalizada,
399                 --ValorAtual eh carregado c/ o proprio valor.
400         end if;
401
402
403     end if;
404
405 else
406     LiberaCalculo <= '0';--Estabelece um nivel baixo no sinal "LiberaCalculo".
407     cont10:=0; -- Para uma atualizacao colocar zero.
408
409 end if;
410 end process rotina3;
411
412 rotina4: process (LiberaCalculo)
413
414 begin
415
416     k0_b <= to_sfixed(0.733,k0_b);--Coeficiente a1. Atribui a k0_b
417         --um valor ja convertido p/ um tipo
418         --pto fixo c/ sinal.
419     k1_b <= to_sfixed(0.267,k1_b);--Coeficiente a2. Atribui a k1_b um
420         --valor ja convertido p/ um tipo pto
421         --fixo c/ sinal.
422     k2_b <= to_sfixed(0.40,k2_b);--Coeficiente b0. Atribui a k2_b um valor
423         --ja convertido p/ um tipo pto fixo c/ sinal.
424     k3_b <= to_sfixed(0.48,k3_b);--Coeficiente b1. Atribui a k3_b um valor
425         --ja convertido p/ um tipo pto fixo c/ sinal.
426     k4_b <= to_sfixed(0.3172,k4_b);--Coeficiente b2. Atribui a k4_b um valor
427         --ja convertido p/ um tipo pto fixo c/ sinal.
428     refComp_b <= to_sfixed(senoComp,refComp_b);--Atribui a refComp_b o valor da
429         --referencia convertido p/ um
430         --tipo pto fixo c/ sinal.
431     valatual_b <= to_sfixed(ValorAtual,valatual_b);--Atribui a valatual_b o
432         --valor da amostragem convertido

```

```

433                                     --p/ um tipo pto fixo c/ sinal.
434
435
436     if rising_edge (LiberaCalculo) then
437
438         ek2_b <= ek1_b;--Atribui ek1_b a ek2_b p/ que na proxima
439                     --amostragem o calculo do sinal de controle
440                     --conte com o valor do erro de duas amostragem passada.
441         uk2_b <= uk1_b;--Atribui uk1_b a uk2_b p/ que na proxima
442                     --amostragem o calculo do sinal de controle
443                     --conte com o valor do sinal de controle obtido
444                     --a duas amostragem passada.
445         ek1_b <= ek_b;--Atribui ek_b a ek1_b p/ que na proxima amostragem
446                     --o calculo do sinal de controle conte com o valor
447                     --anterior do erro.
448         uk1_b <= uk_b;--Atribui uk_b a uk1_b p/ que na proxima amostragem
449                     --o calculo do sinal de controle conte com o valor
450                     --anterior do sinal de controle.
451
452         ek_b <= refComp_b - valatual_b;--Atribui a "ek_b" a diferenca entre
453                                     --o valor de referencia e o valor
454                                     --amostrado.
455
456
457     --*****
458     --Calculo do sinal de controle.
459
460         uk_b <= resize(k0_b*uk1_b + k1_b*uk2_b + k2_b*ek_b - k3_b*ek1_b + k4_b*ek2_b,uk_b);
461
462     --*****
463
464     end if;
465
466     end process rotina4;
467
468 end ADConverter;

```



```

37  shared variable peso: integer :=2048;--utilizado p/ obtermos um
38      --valor decimal na conversao A/D.
39  shared variable cont10: integer :=0;--usado no contador que controla a
40      --comunicacao serial entre ADC e FPGA.
41  shared variable contTS: integer :=0;--contador que auxilia na definicao
42      --da frequencia de amostragem(fase R).
43  shared variable contTS_S: integer :=0;--contador que auxilia na definicao
44      --da frequencia de amostragem(fase S).
45  shared variable contTS_T: integer :=0;--contador que auxilia na definicao
46      --da frequencia de amostragem(fase T).
47  shared variable hab_amostrT : integer :=0;
48  shared variable a5: integer :=1;--auxiliar p/ o clock do ADC.
49  shared variable habilita : integer :=1;
50  shared variable K: integer :=0;--utilizado no contador do Divisor de
51      --Frequencia p/ obter o clock do ADC.
52  shared variable Ten: integer :=1;--auxiliar no "process" do ADC.
53  signal aux10: STD_LOGIC;--Atua simultaneamente ao clock do ADC. Habilita
54      --a rotina que obtem os bits da conversao.
55  signal LiberaCalculo_R: STD_LOGIC;--Libera o clculo da lei de controle(fase R).
56  signal LiberaCalculo_S: STD_LOGIC;--Libera o clculo da lei de controle(fase S).
57  signal LiberaCalculo_T: STD_LOGIC;--Libera o clculo da lei de controle(fase T).
58
59  --*****
60
61  signal k0_b : sfixed(0 downto -17);--recebe o coeficiente a1.
62  signal k1_b : sfixed(0 downto -17);--recebe o coeficiente a2.
63  signal k2_b : sfixed(0 downto -17);--recebe o coeficiente b0.
64  signal k3_b : sfixed(0 downto -17);--recebe o coeficiente b1.
65  signal k4_b : sfixed(0 downto -17);--recebe o coeficiente b2.
66
67  signal refComp_b_R : sfixed(11 downto 0);--recebe o vetor senoide(+/-2047) (fase R).
68  signal valatual_b_R : sfixed(11 downto 0);--recebe saida convertida(+/-2047) (fase R).
69  signal ek_b_R : sfixed(12 downto 0);--recebe o erro;malha fechada(+/-4095) (fase R).
70  signal ek1_b_R : sfixed(12 downto 0);--armazena o erro e(k-1) (fase R).
71  signal ek2_b_R : sfixed(12 downto 0);--armazena o erro e(k-2) (fase R).
72  signal uk_b_R : sfixed(18 downto -18);--sinal de controle(fase R).

```

```

73     signal uk_bn : sfixed(18 downto -18);
74     signal uk1_b_R : sfixed(18 downto -18);--sinal de controle no instante u(k-1) (fase R).
75     signal uk2_b_R : sfixed(18 downto -18);--sinal de controle no instante u(k-2) (fase R).
76
77     signal refComp_b_S : sfixed(11 downto 0);--recebe o vetor senoide(+/-2047) (fase S).
78     signal valatual_b_S : sfixed(11 downto 0);--recebe saida convertida(+/-2047) (fase S).
79     signal ek_b_S : sfixed(12 downto 0);--recebe o erro;malha fechada(+/-4095) (fase S).
80     signal ek1_b_S : sfixed(12 downto 0);--armazena o erro e(k-1) (fase S).
81     signal ek2_b_S : sfixed(12 downto 0);--armazena o erro e(k-2) (fase S).
82     signal uk_b_S : sfixed(18 downto -18);--sinal de controle(fase S).
83     signal uk1_b_S : sfixed(18 downto -18);--sinal de controle no instante u(k-1) (fase S).
84     signal uk2_b_S : sfixed(18 downto -18);--sinal de controle no instante u(k-2) (fase S).
85
86     signal refComp_b_T : sfixed(11 downto 0);--recebe o vetor senoide(+/-2047) (fase T).
87     signal valatual_b_T : sfixed(11 downto 0);--recebe saida convertida(+/-2047) (fase T).
88     signal ek_b_T : sfixed(12 downto 0);--recebe o erro;malha fechada(+/-4095) (fase T).
89     signal ek1_b_T : sfixed(12 downto 0);--armazena o erro e(k-1) (fase T).
90     signal ek2_b_T : sfixed(12 downto 0);--armazena o erro e(k-2) (fase T).
91     signal uk_b_T : sfixed(18 downto -18);--sinal de controle(fase T).
92     signal uk1_b_T : sfixed(18 downto -18);--sinal de controle no instante u(k-1) (fase T).
93     signal uk2_b_T : sfixed(18 downto -18);--sinal de controle no instante u(k-2) (fase T).
94
95     signal saida_R : sfixed(18 downto -18);--recebe o sinal de controle(fase R).
96     signal saida_S : sfixed(18 downto -18);--recebe o sinal de controle(fase S).
97     signal saida_T : sfixed(18 downto -18);--recebe o sinal de controle(fase T).
98
99     --*****
100
101     shared variable a : integer :=0;--variavel p/contagem subidas do clock.
102     shared variable b : integer :=0;--variavel p/contagem subidas do clock.
103     shared variable bS : integer :=0;--variavel p/contagem subidas do clock.
104     shared variable aS : integer :=0;--variavel p/contagem subidas do clock.
105     shared variable aT : integer :=0;--variavel p/contagem subidas do clock.
106     shared variable bT : integer :=0;--variavel p/contagem subidas do clock.
107     shared variable a1R : integer :=0;--variavel p/contagem subidas do clock.
108     shared variable a1S : integer :=0;--variavel p/contagem subidas do clock.

```

```

109     shared variable a1T : integer :=0;--variavel p/contagem subidas do clock.
110     shared variable a2R : integer :=0;--variavel p/contagem subidas do clock.
111     shared variable a2S : integer :=0;--variavel p/contagem subidas do clock.
112     shared variable a2T : integer :=0;--variavel p/contagem subidas do clock.
113     shared variable a3R : integer :=0;--variavel p/contagem subidas do clock.
114     shared variable a3S : integer :=0;--variavel p/contagem subidas do clock.
115     shared variable a3T : integer :=0;--variavel p/contagem subidas do clock.
116     shared variable a4R : integer :=0;--variavel p/contagem subidas do clock.
117     shared variable a4S : integer :=0;--variavel p/contagem subidas do clock.
118     shared variable a4T : integer :=0;--variavel p/contagem subidas do clock.
119     shared variable ref_S : integer :=0;--habilita o process que gera a
120         --modulacao PWM p/ a fase S.
121
122     shared variable ref_T : integer :=0;--habilita o process que gera a
123         --modulacao PWM p/ a fase T.
124     shared variable count_R : integer :=1;--variavel que aponta o indice do vetor(fase R).
125     shared variable count2 : integer :=1;--variavel que aponta o indice do vetor
126     shared variable count2_S : integer :=1;
127     shared variable count_S : integer :=1;--variavel que aponta o indice do vetor(fase S).
128     shared variable count_T : integer :=1;--variavel que aponta o indice do vetor(fase T).
129     shared variable cont_AD_T : integer :=0;--Define qual nivel logico sera
130         --colocado no pino DIN na subida
131         --do SCLK.
132     shared variable contADC: integer :=0;--auxiliar no "process" responsavel pelo ADC.
133     shared variable hab_ADC: integer :=1;--auxiliar no "process" responsavel pelo ADC.
134
135
136
137     type triangular is array (integer range <>) of integer;
138
139     shared variable triangular1:triangular (1 to 625) :=
140     (0,6,12,20,26,32,38,44,52,58,64,70,76,84,90,96,102,108,116,122,128,134,140,148,154,160,166,
141     172,180,186,192,198,204,212,218,224,230,236,244,250,256,262,268,276,282,288,294,
142     300,308,314,320,326,332,340,346,352,358,364,372,378,384,390,396,404,410,416,
143     422,428,436,442,448,454,460,468,474,480,486,492,500,506,512,518,524,532,538,
144     544,550,556,564,570,576,582,588,596,602,608,614,620,628,634,640,646,652,660,

```

```

145 666,672,678,684,692,698,704,710,716,724,730,736,742,748,756,762,768,774,780,
146 788,794,800,806,812,820,826,832,838,844,852,858,864,870,876,884,890,896,902,
147 908,916,922,928,934,940,948,954,960,966,972,980,986,992,998,1004,1012,1018,1024,
148 1030,1036,1044,1050,1056,1062,1068,1076,1082,1088,1094,1100,1108,1114,1120,1126,1132,1140,1146,
149 1152,1158,1164,1172,1178,1184,1190,1196,1204,1210,1216,1222,1228,1236,1242,1248,1254,1260,1268,
150 1274,1280,1286,1292,1300,1306,1312,1318,1324,1332,1338,1344,1350,1356,1364,1370,1376,1382,1388,
151 1396,1402,1408,1414,1420,1428,1434,1440,1446,1452,1460,1466,1472,1478,1484,1492,1498,1504,1510,
152 1516,1524,1530,1536,1542,1548,1556,1562,1568,1574,1580,1588,1594,1600,1606,1612,1620,1626,1632,
153 1638,1644,1652,1658,1664,1670,1676,1684,1690,1696,1702,1708,1716,1722,1728,1734,1740,1748,1754,
154 1760,1766,1772,1780,1786,1792,1798,1804,1812,1818,1824,1830,1836,1844,1850,1856,1862,1868,1876,
155 1882,1888,1894,1900,1908,1914,1920,1926,1932,1940,1946,1952,1958,1964,1972,1978,1984,1990,1996,
156 1997,1990,1984,1978,1972,1964,1958,1952,1946,1940,1932,1926,1920,1914,1908,1900,1894,1888,1882,
157 1876,1868,1862,1856,1850,1844,1836,1830,1824,1818,1812,1804,1798,1792,1786,1780,1772,1766,1760,
158 1754,1748,1740,1734,1728,1722,1716,1708,1702,1696,1690,1684,1676,1670,1664,1658,1652,1644,1638,
159 1632,1626,1620,1612,1606,1600,1594,1588,1580,1574,1568,1562,1556,1548,1542,1536,1530,1524,1516,
160 1510,1504,1498,1492,1484,1478,1472,1466,1460,1452,1446,1440,1434,1428,1420,1414,1408,1402,1396,
161 1388,1382,1376,1370,1364,1356,1350,1344,1338,1332,1324,1318,1312,1306,1300,1292,1286,1280,1274,
162 1268,1260,1254,1248,1242,1236,1228,1222,1216,1210,1204,1196,1190,1184,1178,1172,1164,1158,1152,
163 1146,1140,1132,1126,1120,1114,1108,1100,1094,1088,1082,1076,1068,1062,1056,1050,1044,1036,1030,
164 1024,1018,1012,1004,998,992,986,980,972,966,960,954,948,940,934,928,922,916,908,
165 902,896,890,884,876,870,864,858,852,844,838,832,826,820,812,806,800,794,788,
166 780,774,768,762,756,748,742,736,730,724,716,710,704,698,692,684,678,672,666,
167 660,652,646,640,634,628,620,614,608,602,596,588,582,576,570,564,556,550,544,
168 538,532,524,518,512,506,500,492,486,480,474,468,460,454,448,442,436,428,422,
169 416,410,404,396,390,384,378,372,364,358,352,346,340,332,326,320,314,308,300,
170 294,288,282,276,268,262,256,250,244,236,230,224,218,212,204,198,192,186,180,172,166,
171 160,154,148,140,134,128,122,116,108,102,96,90,84,76,70,64,58,52,44,38,32,26,20,12,6);
172
173 type senoide is array (integer range <>) of integer;
174
175
176 shared variable senoidel:senoide (1 to 625):=
177 (0,20,38,58,76,96,114,134,152,172,190,210,228,248,266,286,304,324,342,360,380,398,
178 416,436,454,472,490,510,528,546,564,582,600,618,636,654,672,690,708,726,744,
179 762,778,796,814,830,848,864,882,898,916,932,948,966,982,998,1014,1030,1046,1062,
180 1078,1094,1110,1124,1140,1156,1170,1186,1200,1214,1230,1244,1258,1272,1286,1300,1314,1328,1342,

```

```

181 1356,1368,1382,1394,1408,1420,1434,1446,1458,1470,1482,1494,1506,1518,1528,1540,1552,1562,1572,
182 1584,1594,1604,1614,1624,1634,1644,1654,1662,1672,1680,1690,1698,1706,1716,1724,1732,1738,1746,
183 1754,1762,1768,1776,1782,1788,1794,1802,1808,1812,1818,1824,1830,1834,1840,1844,1848,1852,1856,
184 1860,1864,1868,1872,1874,1878,1880,1884,1886,1888,1890,1892,1894,1894,1896,1898,1898,1898,1900,
185 1900,1900,1900,1900,1900,1898,1898,1896,1896,1894,1892,1890,1888,1886,1884,1882,1880,1876,1874,
186 1870,1866,1862,1858,1854,1850,1846,1842,1836,1832,1826,1822,1816,1810,1804,1798,1792,1786,1778,
187 1772,1764,1758,1750,1742,1736,1728,1720,1710,1702,1694,1686,1676,1668,1658,1648,1640,1630,1620,
188 1610,1600,1588,1578,1568,1556,1546,1534,1522,1512,1500,1488,1476,1464,1452,1440,1426,1414,1402,
189 1388,1376,1362,1348,1336,1322,1308,1294,1280,1266,1252,1236,1222,1208,1192,1178,1162,1148,1132,
190 1116,1102,1086,1070,1054,1038,1022,1006,990,974,956,940,924,906,890,874,856,840,822,
191 804,788,770,752,734,718,700,682,664,646,628,610,592,574,556,536,518,500,482,
192 464,444,426,408,388,370,352,332,314,294,276,258,238,220,200,182,162,144,124,106,86,
193 66,48,29,10,-10,-29,-48,-66,-86,-104,-124,-144,-162,-182,-200,-220,-238,-258,-276,-294,
194 -314,-332,-352,-370,-388,-408,-426,-444,-464,-482,-500,-518,-536,-556,-574,-592,
195 -610,-628,-646,-664,-682,-700,-718,-734,-752,-770,-788,-804,-822,-840,-856,-874,
196 -890,-906,-924,-940,-956,-974,-990,-1006,-1022,-1038,-1054,-1070,-1086,-1102,-1116,-1132,
197 -1148,-1162,-1178,-1192,-1208,-1222,-1236,-1252,-1266,-1280,-1294,-1308,-1322,-1336,-1348,-1362,
198 -1376,-1388,-1402,-1414,-1426,-1440,-1452,-1464,-1476,-1488,-1500,-1512,-1522,-1534,-1546,-1556,
199 -1568,-1578,-1588,-1600,-1610,-1620,-1630,-1640,-1648,-1658,-1668,-1676,-1686,-1694,-1702,-1710,
200 -1720,-1728,-1736,-1742,-1750,-1758,-1764,-1772,-1778,-1786,-1792,-1798,-1804,-1810,-1816,-1822,
201 -1826,-1832,-1836,-1842,-1846,-1850,-1854,-1858,-1862,-1866,-1870,-1874,-1876,-1880,-1882,-1884,
202 -1886,-1888,-1890,-1892,-1894,-1896,-1896,-1898,-1898,-1900,-1900,-1900,-1900,-1900,-1900,-1898,
203 -1898,-1898,-1896,-1894,-1894,-1892,-1890,-1888,-1886,-1884,-1880,-1878,-1874,-1872,-1868,-1864,
204 -1860,-1856,-1852,-1848,-1844,-1840,-1834,-1830,-1824,-1818,-1812,-1808,-1802,-1794,-1788,-1782,
205 -1776,-1768,-1762,-1754,-1746,-1738,-1732,-1724,-1716,-1706,-1698,-1690,-1680,-1672,-1662,-1654,
206 -1644,-1634,-1624,-1614,-1604,-1594,-1584,-1572,-1562,-1552,-1540,-1528,-1518,-1506,-1494,-1482,
207 -1470,-1458,-1446,-1434,-1420,-1408,-1394,-1382,-1368,-1356,-1342,-1328,-1314,-1300,-1286,-1272,
208 -1258,-1244,-1230,-1214,-1200,-1186,-1170,-1156,-1140,-1124,-1110,-1094,-1078,-1062,-1046,-1030,
209 -1014,-998,-982,-966,-948,-932,-916,-898,-882,-864,-848,-830,-814,-796,-778,-762,
210 -744,-726,-708,-690,-672,-654,-636,-618,-600,-582,-564,-546,-528,-510,-490,-472,
211 -454,-436,-416,-398,-380,-360,-342,-324,-304,-286,-266,-248,-228,-210,-190,-172,
212 -152,-134,-114,-96,-76,-58,-38,-20);
213
214 shared variable saidat1 : integer:=0;--recebe o vetor da portadora positiva (POD).
215 shared variable saidat2 : integer:=0;--recebe o vetor da portadora negativa (POD).
216 shared variable ptempomorto : integer :=5;--tempo morto em percentual(5%).

```

```

217  shared variable freqsen : integer :=50;--frequencia do sinal modulante (50Hz).
218  shared variable freqtri : integer :=40000;--frequencia das portadoras (40KHz).
219  shared variable N1: integer :=1;--variavel dependente da frequencia do sinal modulante.
220  shared variable N2: integer :=1;--variavel dependente da frequencia da portadora.
221  shared variable N3: integer :=1;--variavel dependente do percentual do tempo morto.
222  shared variable senoComp_R: integer :=0;--recebe o valor do sinal de referencia no
223                                     --instante da amostragem(fase R).
224  shared variable senoComp_S: integer :=0;--recebe o valor do sinal de referencia no
225                                     --instante da amostragem(fase S).
226  shared variable senoComp_T: integer :=0;--recebe o valor do sinal de referencia no
227                                     --instante da amostragem(fase T).
228  signal auxR3: bit;--auxiliar p/ habilitar o modulo ADC
229  signal auxS3: bit;--auxiliar p/ habilitar o modulo ADC
230  signal auxT3: bit;--auxiliar p/ habilitar o modulo ADC
231
232
233  --*****
234  --*****
235  begin
236
237  rotinal: process (clk)
238
239      begin
240
241
242          if clk='1' then
243
244              a:=1+a;--Incrementa 1 na variavel a.
245              b:=1+b;--Incrementa 1 na variavel b.
246              N1:=50000000/(freqsen*625); --Define a frequencia em que a variavel
247                                     --"saida" sera atualizada.
248              N2:=50000000/(freqtri*625); --Define a frequencia em que as variaveis
249                                     --"saidat1 e saidat2" serao atualizadas.
250              N3:=(50000000 *ptempomorto)/(freqtri*100);
251
252

```

```

253     if a=N1 then
254
255         saida_R <= uk_b_R;--Atribui a variavel "saida_R" o valor do
256             --sinal de controle "uk_b_R".
257
258         --saida_R :=senoide1(count_R);
259
260         count_R:=1+count_R;--Incrementa 1 na variavel "count1".
261
262         if count_R = 209 then--logica p/ promover a defasagem de 120
263             --graus da fase S.
264             ref_S := 1;--habilita o process que gera a modulacao PWM
265                 --p/ a fase S.
266             defasag <= '1';--p/ verificar a defasagem entre as fases
267         end if;
268         if count_R = 417 then--logica p/ promover a defasagem de 240
269             --graus da fase T.
270             ref_T := 1;--habilita o process que gera a modulacao PWM
271                 --p/ a fase T.
272             defasag <= '0';--p/ verificar a defasagem entre as fases
273         end if;
274         -----
275         if count_R>625 then--Limita count1 em 625 (tamanho do vetor "senoide1")
276             count_R:=1;
277         end if;
278         a:=0;--zera a variavel "a" p/ reiniciar o processo.
279     end if;
280
281     if b=N2 then--
282         saidat1 := triangular1(count2);--Gera internamente a portadora
283             --(triangular positiva).
284         saidat2 :=-triangular1(count2);--Gera internamente a portadora
285             --(triangular negativa).
286         count2:=1+count2;--Incrementa 1 na variavel "count2".
287         b:=0;--zera a variavel "b" p/ reiniciar o processo.
288

```

```

289      --*****
290
291      if hab_amostrT = 1 then--Condicao p/ habilitar a amostragem da fase T
292          --no instante predefinido.
293          if saidat1 = 0 then--Define o instante da Amostragem da fase T.
294              auxT3<='1';--Habilita o "process" responsavel pelo ADC (p/ faseT).
295              hab_amostrT := 0;--zera a variavel "hab_amostrT" p/
296                  --reiniciar o processo.
297          end if;
298      end if;
299
300      if (saidat1 = 1760)then--Condicao p/ interromper o uso do ADC na fase T.
301          auxT3<='0';--Interrompe o uso do ADC na fase T.
302      end if;
303
304      --*****
305
306
307      if saidat1 = 0 then --Condicao p/ habilitar a amostragem da fase R
308          --no instante predefinido.
309          contTS := contTS + 1;
310          if contTS = 4 then--Amostra a cada 2 ciclos da portadora.
311              auxR3<='1';--Habilita o "process" responsavel pelo ADC (p/ faseR).
312              hab_amostrT := 1;--Inicializa a variavel usada p/ possibilitar a
313                  --sequencia de amostragem.
314              contTS := 0;
315          end if;
316      end if;
317
318      if (saidat1 = 1760)then--Condicao p/ interromper o uso do ADC na fase R.
319          auxR3<='0';--Interrompe o uso do ADC na fase R.
320      end if;
321
322      --*****
323      if hab_amostrT = 1 then--Condicao p/ habilitar a amostragem da fase S no
324          --instante predefinido.

```

```

325         if saidat1 = 1997 then--Define o instante da Amostragem da fase S.
326             auxS3<='1';--Habilita o "process" responsavel pelo ADC (p/ fase S).
327         end if;
328     end if;
329
330     if (saidat1 = 300) then--Condicao p/ interromper o uso do ADC na fase S.
331         auxS3<='0';--Interrompe o uso do ADC na fase S.
332     end if;
333
334     if count2>625 then--Limita count2 em 625 (tamanho do vetor "triangular1").
335         count2:=1;--Atribui 1 a count2, pois o indice do vetor inicia em 1.
336     end if;
337 end if;
338 --*****
339
340     if saida_R > saidat1 then--Comparacao entre o sinal de controle e a
341         --portadora positiva (Modlacao da fase R).
342
343         pwmR2<='0';--Coloca em nivel logico baixo a saida "pwm2"
344         a1R:=1+a1R;--Incrementa 1 na variavel "a1" a cada pulso de clock,
345             --estabelecendo um contador.
346         if a1R=N3 then--Logica utilizada p/ gerar o tempo morto entre
347             --as chaves.
348             pwmR1<='1';--Coloca em nivel logico alto a saida "pwm1"
349             a1R:=0;
350             a2R:=0;
351         end if;
352     else
353         pwmR1<='0';--Coloca em nivel logico baixo a saida "pwm1"
354         a2R:=1+a2R;--Incrementa 1 na variavel "a2" a cada pulso de clock,
355             --estabelecendo um contador.
356         if a2R=N3 then--Logica utilizada p/ gerar o tempo morto
357             --entre as chaves.
358             pwmR2<='1';--Coloca em nivel logico alto a saida "pwm2"
359             a2R:=0;
360             a1R:=0;

```

```

361         end if;
362     end if;
363
364
365     if saida_R > saidat2 then--Comparacao entre o sinal de controle e a
366         --portadora negativa (Modlacao da fase R).
367
368         pwmR4<='0';--Coloca em nivel logico baixo a saida "pwm4"
369         a3R:=1+a3R;--Incrementa 1 na variavel "a3" a cada pulso de clock,
370         --estabelecendo um contador.
371         if a3R=N3 then--Logica utilizada p/ gerar o tempo morto
372             --entre as chaves.
373             pwmR3<='1';--Coloca em nivel logico alto a saida "pwm3"
374             a3R:=0;
375             a4R:=0;
376         end if;
377     else
378         pwmR3<='0';--Coloca em nivel logico baixo a saida "pwm3"
379         a4R:=1+a4R;--Incrementa 1 na variavel "a4" a cada pulso de clock,
380         --estabelecendo um contador.
381         if a4R=N3 then--Logica utilizada p/ gerar o tempo morto
382             --entre as chaves.
383             pwmR4<='1';--Coloca em nivel logico alto a saida "pwm4"
384             a3R:=0;
385             a4R:=0;
386         end if;
387     end if;
388
389     end if;
390
391 end process rotinal;
392
393 rotina7: process (clk)
394
395 begin
396

```

```

397
398     if clk='1' then
399
400         if ref_S = 1 then
401
402             aS:=1+aS;
403             bS:=1+bS;
404
405             if aS=N1 then
406
407                 saida_S <= uk_b_S;--Atribui a variavel "saida_S" o
408                     --valor do sinal de controle "uk_b_S".
409                 --saida_S := senoidel(count_S);
410                 count_S:=count_S+1;
411                 if count_S>625 then--Limita count_S em 625
412                     --(tamanho do vetor "senoidel")
413                     count_S:=1;
414                 end if;
415
416                 aS:=0;
417             end if;
418
419
420             if saida_S > saidat1 then--Comparacao entre o sinal de controle
421                 --ea portadora positiva(Modulacao da fase S).
422
423                 pwmS2<='0';--Coloca em nivel logico baixo a saida "pwmS2"
424                 a1S:=1+a1S;--Incrementa 1 na variavel "a1S" a cada pulso de clock,
425                     --estabelecendo um contador.
426                 if a1S=N3 then--Logica utilizada p/ gerar o tempo morto
427                     --entre as chaves.
428                     pwmS1<='1';--Coloca em nivel logico alto a saida "pwmS1"
429                     a1S:=0;
430                     a2S:=0;
431                 end if;
432             else

```

```

433     pwmS1<='0';--Coloca em nivel logico baixo a saida "pwmS1"
434     a2S:=1+a2S;--Incrementa 1 na variavel "a2S" a cada pulso de clock,
435         --estabelecendo um contador.
436     if a2S=N3 then--Logica utilizada p/ gerar o tempo morto
437         --entre as chaves.
438         pwmS2<='1';--Coloca em nivel logico alto a saida "pwmS2"
439         a2S:=0;
440         a1S:=0;
441     end if;
442 end if;
443
444 if saida_S > saidat2 then--Comparacao entre o sinal de controle ea
445     --portadora negativa (Modulacao da fase S).
446
447     pwmS4<='0';--Coloca em nivel logico baixo a saida "pwmS4"
448     a3S:=1+a3S;--Incrementa 1 na variavel "a3S" a cada pulso de clock,
449         --estabelecendo um contador.
450     if a3S=N3 then--Logica utilizada p/ gerar o tempo morto
451         --entre as chaves.
452         pwmS3<='1';--Coloca em nivel logico alto a saida "pwmS3"
453         a3S:=0;
454         a4S:=0;
455     end if;
456 else
457     pwmS3<='0';--Coloca em nivel logico baixo a saida "pwmS3"
458     a4S:=1+a4S;--Incrementa 1 na variavel "a4S" a cada pulso de clock,
459         --estabelecendo um contador.
460     if a4S=N3 then--Logica utilizada p/ gerar o tempo morto
461         --entre as chaves.
462         pwmS4<='1';--Coloca em nivel logico alto a saida "pwmS4".
463         a3S:=0;
464         a4S:=0;
465     end if;
466 end if;
467
468 end if;

```

```

469
470         end if;
471
472     end process rotina7;
473
474     rotina8: process (clk)
475     begin
476
477
478         if clk='1' then
479
480             if ref_T = 1 then
481
482                 aT:=1+aT;
483                 bT:=1+bT;
484
485                 if aT=N1 then
486
487
488                     saida_T <= uk_b_T;--Atribui a variavel "saida_T" o valor
489                                     --do sinal de controle "uk_b_T".
490                     count_T:=count_T+1;
491                     if count_T>625 then--Limita count_T em 625
492                                     --(tamanho do vetor "senoide1")
493                         count_T:=1;
494                     end if;
495
496
497                     aT:=0;
498                 end if;
499
500                 if saida_T > saidat1 then--Comparacao entre o sinal de controle e a
501                                     --portadora positiva (Modulacao da fase T).
502
503
504                     pwmT2<='0';--Coloca em nivel logico baixo a saida "pwmT2".

```

```

505         a1T:=1+a1T;--Incrementa 1 na variavel "a1T" a cada pulso de clock,
506             --estabelecendo um contador.
507         if a1T=N3 then--Logica utilizada p/ gerar o tempo morto
508             --entre as chaves.
509             pwmT1<='1';--Coloca em nivel logico alto a saida "pwmT1".
510             a1T:=0;
511             a2T:=0;
512         end if;
513     else
514         pwmT1<='0';--Coloca em nivel logico baixo a saida "pwmT1".
515         a2T:=1+a2T;--Incrementa 1 na variavel "a2T" a cada pulso de clock,
516             --estabelecendo um contador.
517         if a2T=N3 then--Logica utilizada p/ gerar o tempo morto
518             --entre as chaves.
519             pwmT2<='1';--Coloca em nivel logico alto a saida "pwmT2".
520             a2T:=0;
521             a1T:=0;
522         end if;
523     end if;
524
525     if saida_T > saidat2 then--Comparacao entre o sinal de controle e a
526         --portadora negativa (Modulacao da fase T).
527
528         pwmT4<='0';--Coloca em nivel logico baixo a saida "pwmT4".
529         a3T:=1+a3T;--Incrementa 1 na variavel "a3T" a cada pulso de clock,
530             --estabelecendo um contador.
531         if a3T=N3 then--Logica utilizada p/ gerar o tempo morto
532             --entre as chaves.
533             pwmT3<='1';--Coloca em nivel logico alto a saida "pwmT3".
534             a3T:=0;
535             a4T:=0;
536         end if;
537     else
538         pwmT3<='0';--Coloca em nivel logico baixo a saida "pwmT3".
539         a4T:=1+a4T;--Incrementa 1 na variavel "a4T" a cada pulso de clock,
540             --estabelecendo um contador.

```

```

541         if a4T=N3 then--Logica utilizada p/ gerar o tempo morto
542             --entre as chaves.
543             pwmT4<='1';--Coloca em nivel logico alto a saida "pwmT4".
544             a3T:=0;
545             a4T:=0;
546         end if;
547     end if;
548
549     end if;
550 end if;
551
552 end process rotina8;
553
554
555 rotina2: process (startCS, DOUT, clk)
556
557 begin
558
559     if (auxR3 = '1')or((auxS3 = '1')and(ref_S = 1))or((auxT3 = '1')and(ref_T = 1)) then--Condicao
560         --p/ que o ADC seja acionado.
561
562         if (clk'event and clk = '1')then
563
564             if Ten = 1 then--Condicao p/ carregamento das condicoes
565                 --iniciais p/ uso do ADC.
566                 CS <= '1';--Chip Select (Desabilita o Conversor A/D).
567                 Ten:=0;--Desabilita a logica que promove o carregamento
568                     --das condicoes iniciais p/ uso do ADC.
569                 aux10 <= '1';--Atua simultaneamente ao clock do ADC.
570                     --Condicao inicial p/ uso do ADC.
571                 SCLK <= '1';--Clock do Conversor A/D (Coloca nivel alto
572                     --na saida conectada ao pino SCLK do ADC).
573             end if;
574
575             K:=K+1;--Contador utilizado no Divisor de Frequencia utilizado

```

```

577         --p/ obter o clock do ADC.
578
579     if K=N then --Divisor de frequencia (50e6/10 = 5e6)
580
581         if hab_ADC = 1 then --Condicao p/ que o ADC seja habilitado.
582             CS <= '0'; --Chip Select (Habilita o Conversor A/D).
583         end if;
584
585         if a5 /= 1 then -- Gera a onda quadrada p/ enviar ao ADC.
586
587             a5:= 1;
588             SCLK <= '1'; --Clock do Conversor A/D (Coloca nivel alto na
589                 --saida conectada ao pino SCLK do ADC).
590             aux10 <= '1'; --Atua simultaneamente ao clock do ADC. Habilita
591                 --a rotina que obtem os bits da conversao.
592
593         else a5:= 0;
594
595
596         if (auxR3 = '1') then --Condicao p/ selecionar o canal
597             --IN0 do ADC.
598             confAD <= '0'; --Seleciona canal IN0. Saida conectada
599                 --ao pino DIN do ADC.
600         end if;
601
602         if (auxS3 = '1') then --Condicao p/ selecionar o
603             --canal IN7 do ADC.
604             confAD <= '1'; --Seleciona canal IN7
605         end if;
606
607         if (auxT3 = '1') then --Condicao p/ selecionar o
608             --canal IN6 do ADC.
609         cont_AD_T := cont_AD_T + 1;
610         if cont_AD_T <= 4 then --Define qual nivel logico sera
611             --colocado no pino DIN na subida
612             --do SCLK.

```

```

613         confAD <= '1';--Seleciona canal IN6 do ADC.
614     else
615         confAD <= '0';--Seleciona canal IN6 do ADC.
616     end if;
617 end if;
618
619
620 SCLK <= '0';--Clock do Conversor A/D (Coloca nivel baixo na
621      --saida conectada ao pino SCLK do ADC).
622 aux10 <= '0';--Atua simultaneamente ao clock do ADC. Habilita
623      --a rotina que obtem os bits da conversao.
624
625 contADC := contADC + 1;
626 if contADC = 17 then --Permite que o ADC seja desabilitado
627     --apos a obtencao do frame.
628     CS <= '1';--Chip Select (Desabilita o Conversor A/D).
629     hab_ADC := 0;--Bloqueia a logica que permite que o ADC
630     --seja habilitado.
631 end if;
632
633
634     end if;
635     K:=0;
636 end if;
637
638
639 end if;
640
641 else
642     CS <= '1';--Chip Select (Desabilita o Conversor A/D).
643     Ten:=1;--Habilita a logica que promove o carregamento das condicoes
644     --iniciais p/ uso do ADC.
645     K:=0;
646     a5:= 1;
647     cont_AD_T := 0;
648     contADC := 0;

```

[illegible]

```

685         ValorConv:= ValorConv + peso;--Atribui a "ValorConv" o valor passado de
686         --"ValorConv" somado ao valor de peso.
687         peso:= peso/2;--Divide a variavel "peso" a cada borda de subida de aux10.
688     else
689         peso:= peso/2;--Divide a variavel "peso" a cada borda de subida de aux10.
690     end if;
691 end if;
692
693 if cont10 = 2 then
694     ValorConv:= 0;
695     peso:= 2048;
696
697 end if;
698
699 end if;
700
701 if falling_edge (aux10) then
702
703     cont10:= cont10 + 1;
704
705     if cont10 = 17 then--Permite o carregamento da variavel "ValorAtual"
706         --apos a obtencao do frame.
707
708         ValorAtual:=(ValorConv - 1900);--Elimina o offset adquirido
709         --no circuito de condicionamento.
710
711         if auxR3 = '1' then --Condicao p/ liberar a rotina de calculo do
712             --sinal de controle (faseR).
713             LiberaCalculo_R <= '1'; --Estabelece um nivel alto no sinal
714             --"LiberaCalculo".
715
716         end if;
717         if auxS3 = '1' then --Condicao p/ liberar a rotina de calculo do
718             --sinal de controle (faseS).
719             ValorAtual_S := ValorAtual;
720             LiberaCalculo_S <= '1';--Estabelece um nivel alto no sinal

```

```

721                                     --"LiberaCalculo_S".
722
723     end if;
724     if auxT3 = '1' then --Condicao p/ liberar a rotina de calculo do
725                         --sinal de controle (faseT).
726         ValorAtual_T := ValorAtual;
727         LiberaCalculo_T <= '1';--Estabelece um nivel alto no sinal
728                             --"LiberaCalculo_T".
729
730     end if;
731
732     else
733         ValorAtual := ValorAtual;--Atualiza ValorAtual com o proprio valor a
734                                 --cada borda de descida de aux10, exceto no
735                                 --momento em que cont10 eh igual a 17 indicando
736                                 --o fim da conversao e a obtencao do novo
737                                 --valor amostrado.
738
739     end if;
740
741
742     end if;
743
744     else
745         LiberaCalculo_R <= '0';--Estabelece um nivel baixo no sinal "LiberaCalculo_R".
746         LiberaCalculo_S <= '0';--Estabelece um nivel baixo no sinal "LiberaCalculo_S".
747         LiberaCalculo_T <= '0';--Estabelece um nivel baixo no sinal "LiberaCalculo_T".
748
749     end if;
750     end process rotina3;
751
752
753     rotina4: process (LiberaCalculo_R)
754
755     begin
756

```

```

757  --*****
758  --Coeficientes a e b. Atribui aos k_b's um valor ja convertido p/ um tipo
759  --pto fixo c/ sinal.
760
761      k0_b <= to_sfixed(0.733,k0_b);
762      k1_b <= to_sfixed(0.267,k1_b);
763      k2_b <= to_sfixed(0.40,k2_b);
764      k3_b <= to_sfixed(0.48,k3_b);
765      k4_b <= to_sfixed(0.3172,k4_b);
766
767  --*****
768
769      refComp_b_R <= to_sfixed(senoComp_R,refComp_b_R);--Atribui a refComp_b_R o valor da
770                                           --referencia convertido p/ um tipo
771                                           --pto fixo c/ sinal.
772      valatual_b_R <= to_sfixed(ValorAtual,valatual_b_R);--Atribui a valatual_b_R o valor
773                                           --da amostragem convertido p/
774                                           --um tipo pto fixo c/ sinal.
775
776  --*****
777
778      if rising_edge (LiberaCalculo_R) then--Na borda de subida do sinal "LiberaCalculo_R"
779                                           --inicia-se o calculo do sinal de controle
780                                           --(fase R).
781  --*****
782
783  --Armazena valores atuais p/ que na proxima amostragem o calculo do sinal de controle
784  --conte com valores de amostras passadas.
785      ek2_b_R <= ek1_b_R;
786      uk2_b_R <= uk1_b_R;
787      ek1_b_R <= ek_b_R;
788      uk1_b_R <= uk_b_R;
789
790  --*****
791
792      ek_b_R <= refComp_b_R - valatual_b_R;--Atribui a "ek_b" a diferenca entre o

```

```

791                                     --valor de referencia e o valor amostrado.
792  --*****
793  --Calculo do sinal de controle p/ a fase R.
794      uk_b_R <= resize(k0_b*uk1_b_R + k1_b*uk2_b_R + k2_b*ek_b_R - k3_b*ek1_b_R + k4_b*ek2_b_R,uk_b_R);
795  --*****
796
797
798      end if;
799
800      end process rotina4;
801
802  rotina5: process (LiberaCalculo_S)
803
804      begin
805
806          refComp_b_S <= to_sfixed(senoComp_S,refComp_b_S);--Atribui a refComp_b_S o valor da referencia
807                                     --convertido p/ um tipo pto fixo c/ sinal.
808          valatual_b_S <= to_sfixed(ValorAtual_S,valatual_b_S);--Atribui a valatual_b_S o valor da
809                                     --amostragem convertido p/ um tipo pto fixo
810                                     --c/ sinal.
811
812          if rising_edge (LiberaCalculo_S) then--Na borda de subida do sinal "LiberaCalculo_S" inicia-se o
813          calculo do sinal de controle (fase S).
814  --*****
815
816          --Armazena valores atuais p/ que na proxima amostragem o calculo do sinal de controle
817          --conte com valores de amostras passadas.
818          ek2_b_S <= ek1_b_S;
819          uk2_b_S <= uk1_b_S;
820          ek1_b_S <= ek_b_S;
821          uk1_b_S <= uk_b_S;
822  --*****
823
824          ek_b_S <= refComp_b_S - valatual_b_S;--Atribui a "ek_b_S" a diferenca entre o valor de

```

```

824                                     --referencia e o valor amostrado (fase S).
825
826  --*****
827  --Calculo do sinal de controle p/ a fase S.
828      uk_b_S <= resize(k0_b*uk1_b_S + k1_b*uk2_b_S + k2_b*ek_b_S - k3_b*ek1_b_S + k4_b*ek2_b_S,uk_b_S);
829
830  --*****
831
832      end if;
833
834  end process rotina5;
835
836  rotina6: process (LiberaCalculo_T)
837
838  begin
839
840      refComp_b_T <= to_sfixed(senoComp_T,refComp_b_T);--Atribui a refComp_b_T o valor da referencia
841                                     --convertido p/ um tipo pto fixo c/ sinal.
842      valatual_b_T <= to_sfixed(ValorAtual_T,valatual_b_T);--Atribui a valatual_b_T o valor da
843                                     --amostragem convertido p/ um tipo pto fixo
844                                     --c/ sinal.
845
846      if rising_edge (LiberaCalculo_T) then--Na borda de subida do sinal "LiberaCalculo_T"
847                                     --inicia-se o calculo do sinal de controle (fase T).
848
849  --*****
850
851  --Armazena valores atuais p/ que na proxima amostragem o calculo do sinal de controle
852  --conte com valores de amostras passadas.
853      ek2_b_T <= ek1_b_T;
854      uk2_b_T <= uk1_b_T;
855      ek1_b_T <= ek_b_T;
856      uk1_b_T <= uk_b_T;
857  --*****

```

```

857
858         ek_b_T <= refComp_b_T - valatual_b_T;--Atribui a "ek_b_T" a diferenca entre o valor de
859                                         --referencia e o valor amostrado (fase T).
860
861     --*****
862
863     --Calculo do sinal de controle p/ a fase T.
864     uk_b_T <= resize(k0_b*uk1_b_T + k1_b*uk2_b_T + k2_b*ek_b_T - k3_b*ek1_b_T + k4_b*ek2_b_T,uk_b_T);
865
866     --*****
867
868     end if;
869
870     end process rotina6;
871
872 end ADConverter;

```

APÊNDICE C – CÓDIGO EM VHDL QUE IMPLEMENTA O MÉTODO DE CONTROLE *INDIRECT SELF TUNING*

```

1  library ieee;
2
3  use ieee.std_logic_1164.all;--inclusao de biblioteca.
4
5  use ieee.std_logic_unsigned.all;--inclusao de biblioteca.
6
7  library ieee_proposed;
8  use ieee_proposed.fixed_pkg.all;--inclusao de biblioteca.
9
10 entity testelADC is--declaracao da entidade (I/O).
11
12     port(
13
14         clk, startCS, DOUT  : in std_logic;
15
16         confAD, CS, SCLK : out std_logic;
17
18         pwm1,pwm2,pwm3,pwm4: out bit);
19
20
21 end testelADC;--fim da entidade.
22
23 architecture ADConverter of testelADC is--inicio da arquitetura.
24
25     --*****
26
27     signal yk1 : sfixed(11 downto 0);--sinal de saida no instante y(k-1).
28     signal yk2 : sfixed(11 downto 0);--sinal de saida no instante y(k-2).
29
30     signal phi1 : sfixed(12 downto 0);--Compoe o vetor de regressores.
31     signal phi2 : sfixed(12 downto 0);--Compoe o vetor de regressores.
32     signal phi3 : sfixed(18 downto -18);--Compoe o vetor de regressores.
33     signal phi4 : sfixed(18 downto -18);--Compoe o vetor de regressores.
34
35     --*****
36     --Compoe o vetor de ganho do estimador.

```

```

37
38     signal L1K : sfixed(18 downto -18);
39     signal L2K : sfixed(18 downto -18);
40     signal L3K : sfixed(18 downto -18);
41     signal L4K : sfixed(18 downto -18);
42
43     --*****
44     --Auxiliares p/ calculo do ganho do estimador.
45
46     signal L1Kaux1 : sfixed(18 downto -18);
47     signal L1Kaux2 : sfixed(18 downto -18);
48     signal L1Kaux3 : sfixed(18 downto -18);
49     signal L1Kaux4 : sfixed(18 downto -18);
50     signal L1Kaux5 : sfixed(18 downto -18);
51
52     signal L2Kaux1 : sfixed(18 downto -18);
53     signal L2Kaux2 : sfixed(18 downto -18);
54     signal L2Kaux3 : sfixed(18 downto -18);
55     signal L2Kaux4 : sfixed(18 downto -18);
56     signal L2Kaux5 : sfixed(18 downto -18);
57
58     signal L3Kaux1 : sfixed(18 downto -18);
59     signal L3Kaux2 : sfixed(18 downto -18);
60     signal L3Kaux3 : sfixed(18 downto -18);
61     signal L3Kaux4 : sfixed(18 downto -18);
62     signal L3Kaux5 : sfixed(18 downto -18);
63
64     signal L4Kaux1 : sfixed(18 downto -18);
65     signal L4Kaux2 : sfixed(18 downto -18);
66     signal L4Kaux3 : sfixed(18 downto -18);
67     signal L4Kaux4 : sfixed(18 downto -18);
68     signal L4Kaux5 : sfixed(18 downto -18);
69
70     --*****
71     --Compoe o vetor de parametros estimados.
72

```

```

73     signal theta1K : sfixed(4 downto -17);
74     signal theta2K : sfixed(4 downto -17);
75     signal theta3K : sfixed(4 downto -17);
76     signal theta4K : sfixed(4 downto -17);
77
78     --*****
79     --Compoe o vetor de parametros estimados no instante (k-1) .
80
81     signal theta1K1 : sfixed(4 downto -17);
82     signal theta2K1 : sfixed(4 downto -17);
83     signal theta3K1 : sfixed(4 downto -17);
84     signal theta4K1 : sfixed(4 downto -17);
85
86     --*****
87     signal aux1_theta : sfixed(18 downto -18);
88
89     --*****
90     --Compoe a matriz de covariancia.
91
92     signal P1K : sfixed(4 downto -17);
93     signal P2K : sfixed(4 downto -17);
94     signal P3K : sfixed(4 downto -17);
95     signal P4K : sfixed(4 downto -17);
96     signal P5K : sfixed(4 downto -17);
97     signal P6K : sfixed(4 downto -17);
98     signal P7K : sfixed(4 downto -17);
99     signal P8K : sfixed(4 downto -17);
100    signal P9K : sfixed(4 downto -17);
101    signal P10K : sfixed(4 downto -17);
102    signal P11K : sfixed(4 downto -17);
103    signal P12K : sfixed(4 downto -17);
104    signal P13K : sfixed(4 downto -17);
105    signal P14K : sfixed(4 downto -17);
106    signal P15K : sfixed(4 downto -17);
107    signal P16K : sfixed(4 downto -17);
108

```

```

109  --*****
110  --Compoee a matriz de covariancia no instante (k-1).
111
112  signal P1K1 : sfixed(4 downto -17);
113  signal P2K1 : sfixed(4 downto -17);
114  signal P3K1 : sfixed(4 downto -17);
115  signal P4K1 : sfixed(4 downto -17);
116  signal P5K1 : sfixed(4 downto -17);
117  signal P6K1 : sfixed(4 downto -17);
118  signal P7K1 : sfixed(4 downto -17);
119  signal P8K1 : sfixed(4 downto -17);
120  signal P9K1 : sfixed(4 downto -17);
121  signal P10K1 : sfixed(4 downto -17);
122  signal P11K1 : sfixed(4 downto -17);
123  signal P12K1 : sfixed(4 downto -17);
124  signal P13K1 : sfixed(4 downto -17);
125  signal P14K1 : sfixed(4 downto -17);
126  signal P15K1 : sfixed(4 downto -17);
127  signal P16K1 : sfixed(4 downto -17);
128
129  --*****
130
131  shared variable N: integer :=10;--Auxiliar do divisor de
132  --frequencia p/ clock do ADC.
133  shared variable ValorConv: integer :=0;--recebe o valor do ADC.
134
135  --*****
136  --Auxiliares p/ atribuir os valores iniciais do vetor
137  --de parametros estimados.
138
139  shared variable inic_coef1 : integer :=0;
140  shared variable inic_coef2 : integer :=0;
141  shared variable inic_coef3 : integer :=0;
142  shared variable inic_coef4: integer :=0;
143
144  --*****

```

```

145  shared variable ValorAtual : integer :=0;--recebe o valor do ADC
146                                     --s/ o offset.
147  shared variable iniciaTheta : integer :=0;--auxiliar p/ o carregamento das
148                                     --condicoes iniciais do vetor
149                                     --de regressor theta.
150  shared variable peso: integer :=2048;--utilizado p/ obtermos um
151                                     --valor decimal na conversao A/D.
152  shared variable cont10: integer :=0;--usado no contador que controla a
153                                     --comunicacao serial entre ADC e FPGA.
154  shared variable contTS: integer :=0;--contador que auxilia na definicao
155                                     --da frequencia de amostragem.
156  shared variable contTS_2: integer :=0;--contador que auxilia na definicao
157                                     --do instante do calculo do
158                                     --sinal de controle.
159  shared variable a5: integer :=1;--auxiliar p/ o clock do ADC.
160  shared variable K: integer :=0;--utilizado no contador do Divisor de
161                                     --Frequencia p/ obter o clock do ADC.
162  shared variable Ten: integer :=1;--auxiliar no "process" do ADC.
163  signal aux10: STD_LOGIC;--Atua simultaneamente ao clock do ADC. Habilita
164                                     --a rotina que obtem os bits da conversao.
165  signal LiberaCalculo: STD_LOGIC;--Libera codigo do MQR em um instante
166                                     --pre-definido.
167  signal LiberaC_LK: STD_LOGIC;--Libera calculo dos parametros estimados.
168
169  --*****
170  --Recebem os parametros estimados.
171
172  signal k0_b : sfixed(4 downto -17);
173  signal k2_b : sfixed(4 downto -17);
174  signal k3_b : sfixed(4 downto -17);
175  signal k4_b : sfixed(4 downto -17);
176
177  --*****
178
179  signal s0 : sfixed(12 downto -12);--utilizado p/ calcular a lei de
180                                     --de controle.

```

```

181
182     signal refComp_b : sfixed(11 downto 0);--recebe o vetor senoide(+/-2047).
183     signal valatual_b : sfixed(11 downto 0);--recebe saida convertida(+/-2047).
184     signal ek_b : sfixed(12 downto 0);--recebe o erro;malha fechada(+/-4095).
185     signal ek1_b : sfixed(12 downto 0);--armazena o erro e(k-1).
186     signal ek2_b : sfixed(12 downto 0);--armazena o erro e(k-2).
187     signal uk_b : sfixed(18 downto -18);--sinal de controle.
188     signal uk1_b : sfixed(18 downto -18);--sinal de controle no instante u(k-1).
189     signal uk2_b : sfixed(18 downto -18);--sinal de controle no instante u(k-2).
190
191     signal saida_b : sfixed(18 downto -18);--recebe o sinal de controle.
192
193     --*****
194
195     shared variable a : integer :=0;--variavel p/contagem subidas do clock
196     shared variable b : integer :=0;--variavel p/contagem subidas do clock
197     shared variable a1 : integer :=0;--variavel p/contagem subidas do clock
198     shared variable a2 : integer :=0;--variavel p/contagem subidas do clock
199     shared variable a3 : integer :=0;--variavel p/contagem subidas do clock
200     shared variable a4 : integer :=0;--variavel p/contagem subidas do clock
201     shared variable count1 : integer :=1;--variavel que aponta o indice do vetor.
202     shared variable count2 : integer :=1;--variavel que aponta o indice do vetor.
203
204     type triangular is array (integer range <>) of integer;--declaracao do vetor da portadora.
205                                     --625 amostras entre 0 e 1996.
206
207     shared variable triangular1:triangular (1 to 625) :=
208     (6,12,20,26,32,38,44,52,58,64,70,76,84,90,96,102,108,116,122,128,134,140,148,154,160,166,
209     172,180,186,192,198,204,212,218,224,230,236,244,250,256,262,268,276,282,288,294,
210     300,308,314,320,326,332,340,346,352,358,364,372,378,384,390,396,404,410,416,
211     422,428,436,442,448,454,460,468,474,480,486,492,500,506,512,518,524,532,538,
212     544,550,556,564,570,576,582,588,596,602,608,614,620,628,634,640,646,652,660,
213     666,672,678,684,692,698,704,710,716,724,730,736,742,748,756,762,768,774,780,
214     788,794,800,806,812,820,826,832,838,844,852,858,864,870,876,884,890,896,902,
215     908,916,922,928,934,940,948,954,960,966,972,980,986,992,998,1004,1012,1018,1024,
216     1030,1036,1044,1050,1056,1062,1068,1076,1082,1088,1094,1100,1108,1114,1120,1126,1132,1140,1146,

```

```

217 1152,1158,1164,1172,1178,1184,1190,1196,1204,1210,1216,1222,1228,1236,1242,1248,1254,1260,1268,
218 1274,1280,1286,1292,1300,1306,1312,1318,1324,1332,1338,1344,1350,1356,1364,1370,1376,1382,1388,
219 1396,1402,1408,1414,1420,1428,1434,1440,1446,1452,1460,1466,1472,1478,1484,1492,1498,1504,1510,
220 1516,1524,1530,1536,1542,1548,1556,1562,1568,1574,1580,1588,1594,1600,1606,1612,1620,1626,1632,
221 1638,1644,1652,1658,1664,1670,1676,1684,1690,1696,1702,1708,1716,1722,1728,1734,1740,1748,1754,
222 1760,1766,1772,1780,1786,1792,1798,1804,1812,1818,1824,1830,1836,1844,1850,1856,1862,1868,1876,
223 1882,1888,1894,1900,1908,1914,1920,1926,1932,1940,1946,1952,1958,1964,1972,1978,1984,1990,1996,
224 1996,1990,1984,1978,1972,1964,1958,1952,1946,1940,1932,1926,1920,1914,1908,1900,1894,1888,1882,
225 1876,1868,1862,1856,1850,1844,1836,1830,1824,1818,1812,1804,1798,1792,1786,1780,1772,1766,1760,
226 1754,1748,1740,1734,1728,1722,1716,1708,1702,1696,1690,1684,1676,1670,1664,1658,1652,1644,1638,
227 1632,1626,1620,1612,1606,1600,1594,1588,1580,1574,1568,1562,1556,1548,1542,1536,1530,1524,1516,
228 1510,1504,1498,1492,1484,1478,1472,1466,1460,1452,1446,1440,1434,1428,1420,1414,1408,1402,1396,
229 1388,1382,1376,1370,1364,1356,1350,1344,1338,1332,1324,1318,1312,1306,1300,1292,1286,1280,1274,
230 1268,1260,1254,1248,1242,1236,1228,1222,1216,1210,1204,1196,1190,1184,1178,1172,1164,1158,1152,
231 1146,1140,1132,1126,1120,1114,1108,1100,1094,1088,1082,1076,1068,1062,1056,1050,1044,1036,1030,
232 1024,1018,1012,1004,998,992,986,980,972,966,960,954,948,940,934,928,922,916,908,
233 902,896,890,884,876,870,864,858,852,844,838,832,826,820,812,806,800,794,788,
234 780,774,768,762,756,748,742,736,730,724,716,710,704,698,692,684,678,672,666,
235 660,652,646,640,634,628,620,614,608,602,596,588,582,576,570,564,556,550,544,
236 538,532,524,518,512,506,500,492,486,480,474,468,460,454,448,442,436,428,422,
237 416,410,404,396,390,384,378,372,364,358,352,346,340,332,326,320,314,308,300,
238 294,288,282,276,268,262,256,250,244,236,230,224,218,212,204,198,192,186,180,172,166,
239 160,154,148,140,134,128,122,116,108,102,96,90,84,76,70,64,58,52,44,38,32,26,20,12,6,0);
240
241 type senoide is array (integer range <>) of integer;--declaracao do vetor do sinal modulante.
242                                     --625 amostras entre 0 e 1900. Indice de
243                                     --modulacao de amplitude igual a 0,95.
244
245 shared variable senoidel:senoide (1 to 625):=
246 (20,38,58,76,96,114,134,152,172,190,210,228,248,266,286,304,324,342,360,380,398,
247 416,436,454,472,490,510,528,546,564,582,600,618,636,654,672,690,708,726,744,
248 762,778,796,814,830,848,864,882,898,916,932,948,966,982,998,1014,1030,1046,1062,
249 1078,1094,1110,1124,1140,1156,1170,1186,1200,1214,1230,1244,1258,1272,1286,1300,1314,1328,1342,
250 1356,1368,1382,1394,1408,1420,1434,1446,1458,1470,1482,1494,1506,1518,1528,1540,1552,1562,1572,
251 1584,1594,1604,1614,1624,1634,1644,1654,1662,1672,1680,1690,1698,1706,1716,1724,1732,1738,1746,
252 1754,1762,1768,1776,1782,1788,1794,1802,1808,1812,1818,1824,1830,1834,1840,1844,1848,1852,1856,

```

```

253 1860,1864,1868,1872,1874,1878,1880,1884,1886,1888,1890,1892,1894,1894,1896,1898,1898,1898,1900,
254 1900,1900,1900,1900,1900,1900,1898,1898,1896,1896,1894,1892,1890,1888,1886,1884,1882,1880,1876,1874,
255 1870,1866,1862,1858,1854,1850,1846,1842,1836,1832,1826,1822,1816,1810,1804,1798,1792,1786,1778,
256 1772,1764,1758,1750,1742,1736,1728,1720,1710,1702,1694,1686,1676,1668,1658,1648,1640,1630,1620,
257 1610,1600,1588,1578,1568,1556,1546,1534,1522,1512,1500,1488,1476,1464,1452,1440,1426,1414,1402,
258 1388,1376,1362,1348,1336,1322,1308,1294,1280,1266,1252,1236,1222,1208,1192,1178,1162,1148,1132,
259 1116,1102,1086,1070,1054,1038,1022,1006,990,974,956,940,924,906,890,874,856,840,822,
260 804,788,770,752,734,718,700,682,664,646,628,610,592,574,556,536,518,500,482,
261 464,444,426,408,388,370,352,332,314,294,276,258,238,220,200,182,162,144,124,106,86,
262 66,48,29,10,-10,-29,-48,-66,-86,-104,-124,-144,-162,-182,-200,-220,-238,-258,-276,-294,
263 -314,-332,-352,-370,-388,-408,-426,-444,-464,-482,-500,-518,-536,-556,-574,-592,
264 -610,-628,-646,-664,-682,-700,-718,-734,-752,-770,-788,-804,-822,-840,-856,-874,
265 -890,-906,-924,-940,-956,-974,-990,-1006,-1022,-1038,-1054,-1070,-1086,-1102,-1116,-1132,
266 -1148,-1162,-1178,-1192,-1208,-1222,-1236,-1252,-1266,-1280,-1294,-1308,-1322,-1336,-1348,-1362,
267 -1376,-1388,-1402,-1414,-1426,-1440,-1452,-1464,-1476,-1488,-1500,-1512,-1522,-1534,-1546,-1556,
268 -1568,-1578,-1588,-1600,-1610,-1620,-1630,-1640,-1648,-1658,-1668,-1676,-1686,-1694,-1702,-1710,
269 -1720,-1728,-1736,-1742,-1750,-1758,-1764,-1772,-1778,-1786,-1792,-1798,-1804,-1810,-1816,-1822,
270 -1826,-1832,-1836,-1842,-1846,-1850,-1854,-1858,-1862,-1866,-1870,-1874,-1876,-1880,-1882,-1884,
271 -1886,-1888,-1890,-1892,-1894,-1896,-1896,-1898,-1898,-1900,-1900,-1900,-1900,-1900,-1900,-1898,
272 -1898,-1898,-1896,-1894,-1894,-1892,-1890,-1888,-1886,-1884,-1880,-1878,-1874,-1872,-1868,-1864,
273 -1860,-1856,-1852,-1848,-1844,-1840,-1834,-1830,-1824,-1818,-1812,-1808,-1802,-1794,-1788,-1782,
274 -1776,-1768,-1762,-1754,-1746,-1738,-1732,-1724,-1716,-1706,-1698,-1690,-1680,-1672,-1662,-1654,
275 -1644,-1634,-1624,-1614,-1604,-1594,-1584,-1572,-1562,-1552,-1540,-1528,-1518,-1506,-1494,-1482,
276 -1470,-1458,-1446,-1434,-1420,-1408,-1394,-1382,-1368,-1356,-1342,-1328,-1314,-1300,-1286,-1272,
277 -1258,-1244,-1230,-1214,-1200,-1186,-1170,-1156,-1140,-1124,-1110,-1094,-1078,-1062,-1046,-1030,
278 -1014,-998,-982,-966,-948,-932,-916,-898,-882,-864,-848,-830,-814,-796,-778,-762,
279 -744,-726,-708,-690,-672,-654,-636,-618,-600,-582,-564,-546,-528,-510,-490,-472,
280 -454,-436,-416,-398,-380,-360,-342,-324,-304,-286,-266,-248,-228,-210,-190,-172,
281 -152,-134,-114,-96,-76,-58,-38,-20,0);
282
283 shared variable saidat1 : integer:=0;--recebe o vetor da portadora positiva (POD).
284 shared variable saidat2 : integer:=0;--recebe o vetor da portadora negativa (POD).
285 shared variable ptempomorto : integer :=5;--tempo morto em percentual(5%).
286 shared variable freqsen : integer :=50;--frequencia do sinal modulante (50Hz).
287 shared variable freqtri : integer :=40000;--frequencia das portadoras (40KHz).
288 shared variable N1 : integer :=1;--variavel dependente da frequencia do sinal modulante.

```

```

289  shared variable N2: integer :=1;--variavel dependente da frequencia da portadora.
290  shared variable N3: integer :=1;--variavel dependente do percentual do tempo morto.
291  shared variable aux5: integer :=0;--Utilizado para que o calculo do sinal de controle
292      --seja realizado apos a amostragem.
293  shared variable senoComp: integer :=0;--recebe o valor do sinal de referencia no
294      --instante da amostragem.
295  shared variable contADC: integer :=0;--auxiliar no "process" responsavel pelo ADC.
296  shared variable hab_ADC: integer :=1;--auxiliar no "process" responsavel pelo ADC.
297  signal aux3: bit;--Sinal utilizado p/ habilitar o "process" responsavel
298      --pelo ADC e o "process" responsavel pela estimacao
299      --dos parametros do controlador.
300  signal LibCalcUk: STD_LOGIC;--Libera o calculo da lei de controle no instante
301      --pre-definido.
302
303
304
305  --*****
306  --*****
307  begin
308
309  rotinal: process (clk)
310
311      begin
312
313
314          if clk='1' then
315
316              a:=1+a;--Incrementa 1 na variavel "a" a cada nivel logico alto do clock.
317              b:=1+b;--Incrementa 1 na variavel "b" a cada nivel logico alto do clock.
318              N1:=50000000/(freqsen*625); --Define a frequencia em que a variavel
319                  --"saida_b" sera atualizada.
320              N2:=50000000/(freqtri*625); --Define a frequencia em que as variaveis
321                  --"saidat1 e saidat2" serao atualizadas.
322              N3:=(50000000*ptempomorto)/(freqtri*100);--Usado p/ efetuar o tempo morto.
323
324

```

```

325     if a=N1 then
326
327         saida_b <= uk_b;--Atribui a variavel "saida_b" o valor do sinal de
328             --controle "uk_b".
329         count1:=1+count1;--Incrementa 1 na variavel "count1".
330         if count1>625 then--Limita count1 em 625 (tamanho do vetor "senoide1")
331             count1:=1;
332         end if;
333         a:=0;--zera a variavel "a" p/ reiniciar o processo.
334     end if;
335     if b=N2 then
336         saidat1 := triangular1(count2);--Gera internamente a portadora
337             --triangular positiva).
338         saidat2 :=-triangular1(count2);--Gera internamente a portadora
339             --(triangular negativa).
340         count2:=1+count2;--Incrementa 1 na variavel "count2".
341         b:=0;--zera a variavel "b" p/ reiniciar o processo.
342
343         if saidat1 = 0 then--Condicao p/ habilitar a amostragem no
344             --instante predefinido.
345             contTS := contTS + 1;
346             if contTS = 2 then--Amostra a cada 2 ciclos da portadora.
347                 aux3<='1';--Habilita o "process" responsavel pelo ADC e o "process"
348                     --responsavel pela estimacao dos parametros do controlador.
349                 LibCalcUk <= '0';--Desabilita o "process" responsavel pelo
350                     --calculo do sinal de controle.
351                 contTS := 0;--zera o contador p/ reiniciar o processo.
352                 aux5 := 1;--Garante que o calculo do sinal de controle
353                     --seja realizado apos a amostragem.
354             end if;
355         end if;
356
357         if aux5 = 1 then--Condicao p/ que o calculo do sinal de controle
358             --seja realizado apos a amostragem.
359
360             if saidat1 = 1996 then--Condicao p/ que o calculo do sinal de controle

```

```

361                                     --seja realizado no valor de pico da portadora.
362     contTS_2 := contTS_2 + 1;
363     if contTS_2 = 2 then--Condicao p/ que o calculo do sinal de controle
364                                     --seja realizado no segundo pico da portadora
365                                     --apos a amostragem.
366         aux3<='0';--Desabilita o "process" responsavel pelo ADC e o
367         --"process" responsavel pela estimacao dos parametros
368         --do controlador.
369         LibCalcUk <= '1';--Habilita o "process" responsavel pelo calculo
370         --do sinal de controle.
371         contTS_2 := 0;--zera o contador p/ reiniciar o processo.
372     end if;
373 end if;
374
375 end if;
376
377     if count2>625 then--Limita count2 em 625 (tamanho do vetor "triangular1").
378         count2:=1;--Atribui 1 a count2, pois o indice do vetor inicia em 1.
379     end if;
380 end if;
381
382 if saida_b > saidat1 then--Comparacao entre o sinal de controle
383     --e a portadora positiva.
384
385     pwm2<='0';--Coloca em nivel logico baixo a saida "pwm2"
386     a1:=1+a1;--Incrementa 1 na variavel "a1" a cada pulso de clock,
387     --estabelecendo um contador.
388     if a1=N3 then--Logica utilizada p/ gerar o tempo morto entre as chaves.
389         pwm1<='1';--Coloca em nivel logico alto a saida "pwm1"
390         a1:=0;
391         a2:=0;
392     end if;
393 else
394     pwm1<='0';--Coloca em nivel logico baixo a saida "pwm1"
395     a2:=1+a2;--Incrementa 1 na variavel "a2" a cada pulso de clock,
396     --estabelecendo um contador.

```

```

397         if a2=N3 then--Logica utilizada p/ gerar o tempo morto entre as chaves.
398             pwm2<='1';--Coloca em nivel logico alto a saida "pwm2"
399             a2:=0;
400             a1:=0;
401         end if;
402     end if;
403
404     if saida_b > saidat2 then--Comparacao entre o sinal de controle
405         --e a portadora negativa.
406
407         pwm4<='0';--Coloca em nivel logico baixo a saida "pwm4"
408         a3:=1+a3;--Incrementa 1 na variavel "a3" a cada pulso de clock,
409             --estabelecendo um contador.
410         if a3=N3 then--Logica utilizada p/ gerar o tempo morto entre as chaves.
411             pwm3<='1';--Coloca em nivel logico alto a saida "pwm3"
412             a3:=0;
413             a4:=0;
414         end if;
415     else
416         pwm3<='0';--Coloca em nivel logico baixo a saida "pwm3"
417         a4:=1+a4;--Incrementa 1 na variavel "a4" a cada pulso de clock,
418             --estabelecendo um contador.
419         if a4=N3 then--Logica utilizada p/ gerar o tempo morto entre as chaves.
420             pwm4<='1';--Coloca em nivel logico alto a saida "pwm4"
421             a3:=0;
422             a4:=0;
423         end if;
424     end if;
425
426 end if;
427
428 end process rotinal;
429
430 --*****
431 --*****
432

```

```

433     rotina2: process (DOUT, clk)
434
435     begin
436
437     if aux3 = '1' then--aux3 vai p/ nivel alto no momento da amostragem.
438
439         confAD <= '0';--Utilizado na configuracao do canal 0 do ADC.
440
441         if (clk'event and clk = '1')then
442
443             if Ten = 1 then--Condicao p/ carregamento das
444                 --condicoes iniciais p/ uso do ADC.
445                 CS <= '1';--Chip Select (Desabilita o Conversor A/D).
446                 aux10 <= '1';--Atua simultaneamente ao clock do ADC.
447                 --Condicao inicial p/ uso do ADC.
448                 SCLK <= '1';--Clock do Conversor A/D (Coloca nivel alto
449                 --na saida conectada ao pino SCLK do ADC).
450                 Ten:=0;--Desabilita a logica que promove o carregamento
451                 --das condicoes iniciais p/ uso do ADC.
452             end if;
453
454             K:=K+1;--Contador utilizado no Divisor de Frequencia p/
455                 --obter o clock do ADC.
456
457             if K=N then--Divisor de frequencia (50e6/10 = 5e6)
458
459                 if hab_ADC = 1 then--Condicao p/ que o ADC seja habilitado.
460                     CS <= '0';--Chip Select (Habilita o Conversor A/D).
461                 end if;
462
463                 if a5 /= 1 then-- Gera a onda quadrada p/ enviar ao ADC.
464
465                     a5:= 1;
466                     SCLK <= '1';--Clock do Conversor A/D (Coloca nivel alto na
467                     --saida conectada ao pino SCLK do ADC).
468                     aux10 <= '1';--Atua simultaneamente ao clock do ADC. Habilita

```

```

469             --a rotina que obtem os bits da conversao.
470
471         else a5:= 0;
472
473             SCLK <= '0';--Clock do Conversor A/D (Coloca nivel baixo na
474                 --saida conectada ao pino SCLK do ADC).
475             aux10 <= '0';--Atua simultaneamente ao clock do ADC. Habilita
476                 --a rotina que obtem os bits da conversao.
477
478             contADC := contADC + 1;
479             if contADC = 17 then --Permite que o ADC seja desabilitado
480                 --apos a obtencao do frame.
481                 CS <= '1';--Chip Select (Desabilita o Conversor A/D).
482                 hab_ADC := 0;--Bloqueia a logica que permite que o ADC
483                     --seja habilitado.
484             end if;
485         end if;
486         K:=0;--Estabelece condicao inicial p/ a proxima amostragem
487     end if;
488
489
490 end if;
491
492 else
493
494     CS <= '1';--Chip Select (Desabilita o Conversor A/D).
495     Ten:=1;--Habilita a logica que promove o carregamento
496         --das condicoes iniciais p/ uso do ADC.
497     K:=0;--Estabelece condicao inicial p/ a proxima amostragem
498     a5:= 1;--Estabelece condicao inicial p/ a proxima amostragem
499     contADC := 0;--Estabelece condicao inicial p/ a proxima amostragem
500     hab_ADC := 1;--Estabelece condicao inicial p/ a proxima amostragem
501
502 end if;
503 end process rotina2;
504

```

```

505     rotina3: process (DOUT,aux10)
506
507     begin
508
509     if aux3 = '1' then--sinal eh levado a nivel alto no momento da amostragem.
510
511         if rising_edge (aux10) then--Executado na borda se subida de aux10.
512
513             if (cont10 >= 5)and(cont10 < 17) then--Permite a obtencao dos bits da
514                 --conversao que sao apresentados no pino
515                 --DOUT apos a quinta borda de descida do
516                 --SCLK. O carregamento eh realizado na
517                 --quinta borda de subida de SCLK.
518
519                 if cont10 = 5 then--Permite salvar o valor do sinal de referencia
520                     --no instante da amostragem p/ comparacao posterior.
521                     senoComp := senoidel(count1);--Carrega o valor da referencia na
522                     -- variavel "senoComp".
523                 end if;
524
525                 if DOUT = '1' then--Permite que "ValorConv" seja incrementada
526                     --sempre que um nivel alto eh apresentado em DOUT.
527                     ValorConv:= ValorConv + peso;--Atribui a "ValorConv" o valor passado
528                     --de "ValorConv" somado ao valor de peso.
529                     peso:= peso/2;--Divide a variavel "peso" a cada borda de subida de aux10.
530                 else
531                     peso:= peso/2;--Divide a variavel "peso" a cada borda de subida de aux10.
532                 end if;
533             end if;
534
535             if cont10 = 2 then--Condicao p/ inicializar variaveis.
536                 ValorConv:= 0;--Estabelece condicao inicial p/ a proxima amostragem.
537                 peso:= 2048;--Estabelece condicao inicial p/ a proxima amostragem.
538
539             end if;
540

```

```

541     end if;
542
543     if falling_edge (aux10) then--Executado na borda de descida de aux10.
544
545         cont10:= cont10 + 1;--contador que controla a comunicacao serial entre
546             --ADC e FPGA.
547
548         if cont10 = 17 then--Permite o carregamento da variavel
549             --"ValorAtual" apos a obtencao do frame.
550             ValorAtual:=(ValorConv - 1900);--Elimina o offset adquirido no
551                 --circuito de condicionamento.
552             LiberaCalculo <= '1';--Estabelece um nivel alto no sinal "LiberaCalculo".
553
554         else
555             ValorAtual:= ValorAtual;--Enquanto a convesao nao eh finalizada,
556                 --ValorAtual eh carregado c/ o proprio valor.
557         end if;
558
559     end if;
560
561
562 else
563     LiberaCalculo <= '0';--Estabelece um nivel baixo no sinal "LiberaCalculo".
564     cont10:=0; -- Para uma atualizacao colocar zero.
565
566 end if;
567 end process rotina3;
568
569 --*****
570 --*****
571
572 rotina4: process (LiberaCalculo)
573
574 begin
575
576     if aux3 = '1' then--sinal eh levado a nivel alto no momento da amostragem.

```

```

577
578     refComp_b <= to_sfixed(senoComp,refComp_b);--Atribui a refComp_b o valor da
579                                           --referencia convertido p/ um tipo pto fixo
580                                           --c/ sinal e com um numero pre-definido de bits.
581     valatual_b <= to_sfixed(ValorAtual,valatual_b);--Atribui a valatual_b o valor da amostragem
582                                           --convertido p/ um tipo pto fixo c/ sinal e
583                                           --com um numero pre-definido de bits.
584
585     if rising_edge (LiberaCalculo) then
586
587         ek2_b <= ek1_b;--Atribui ek1_b a ek2_b p/ que na proxima amostragem o calculo do sinal
588                     --de controle conte com o valor do erro de duas amostragem passada.
589         uk2_b <= uk1_b;--Atribui uk1_b a uk2_b p/ ser usado por phi4.
590         ek1_b <= ek_b;--Atribui ek_b a ek1_b p/ que na proxima amostragem o calculo do sinal de
591                     --controle conte com o valor anterior do erro.
592         uk1_b <= uk_b;--Atribui uk_b a uk1_b p/ que na proxima amostragem o calculo do sinal de
593                     --controle conte com o valor anterior do sinal de controle e p/ ser
594                     --atribuido a phi3.
595
596         ek_b <= refComp_b - valatual_b;--Atribui a "ek_b" a diferenca entre o valor de referencia
597                     --e o valor amostrado.
598
599         phi1 <= -yk1;--fi1 compoe o vetor de regressor phi.
600         phi2 <= -yk2;--fi2 compoe o vetor de regressor phi.
601         phi3 <= uk1_b;--fi3 compoe o vetor de regressor phi.
602         phi4 <= uk2_b;--fi4 compoe o vetor de regressor phi.
603
604         if iniciaTheta = 0 then--Permite o carregamento das condicoes iniciais
605             --do vetor de regressor theta.
606             --*****
607             --Carrega nos parametros thetaXK1 um valor convertido p/ ponto fixo com sinal e c/
608             --um numero predefinido de bits.
609
610             theta1K1 <= to_sfixed(0,theta1K1);
611             theta2K1 <= to_sfixed(0,theta2K1);
612             theta3K1 <= to_sfixed(0,theta3K1);

```

```

613         theta4K1 <= to_sfixed(0,theta4K1);
614
615         --*****
616
617         iniciaTheta := iniciaTheta + 1;
618
619         else
620
621         --*****
622         --Possibilita a contrucao de um vetor de regressor theta com valores de uma amostra passada.
623             theta1K1 <= theta1K;
624             theta2K1 <= theta2K;
625             theta3K1 <= theta3K;
626             theta4K1 <= theta4K;
627
628         --*****
629         end if;
630
631         --*****
632         ****
633         --Sinal auxiliar que recebe o resultado de parte do calculo utilizado na obtencao de todos os parametros
634         theta.
635
636         aux1_theta <= resize(valatual_b - phi1*theta1K1 - phi2*theta2K1 - phi3*theta3K1 - phi4*theta4K1,
637         aux1_theta );
638
639         --*****
640         ****
641
642         --*****
643         --Possibilita a contrucao de uma matriz de covariancia com valores de uma amostra passada.
644
645         P1K1 <= P1K;
646         P2K1 <= P2K;
647         P3K1 <= P3K;
648         P4K1 <= P4K;

```

```

644         P5K1 <= P5K;
645         P6K1 <= P6K;
646         P7K1 <= P7K;
647         P8K1 <= P8K;
648         P9K1 <= P9K;
649         P10K1 <= P10K;
650         P11K1 <= P11K;
651         P12K1 <= P12K;
652         P13K1 <= P13K;
653         P14K1 <= P14K;
654         P15K1 <= P15K;
655         P16K1 <= P16K;
656
657         --*****
658
659         LiberaC_LK <= '1';--Habilita as rotinas de calculo dos parametros theta.
660
661         end if;
662     else
663
664         LiberaC_LK <= '0';--Desabilita as rotinas de calculo dos parametros theta.
665
666     end if;
667
668     end process rotina4;
669
670     rotina5: process (LiberaC_LK)
671     begin
672
673         if rising_edge (LiberaC_LK) then--Condicao p/ calculo dos parametros theta.
674
675
676
677         if inic_coef1 = 0 then--Condicao para o carregamento das condicoes iniciais da matriz de
678             --covariancia(Matriz diagonal inicialmente).
679

```

```

680  --*****
681  --Atribui a cada um dos parametros "P" um valor convertido p/ o tipo ponto fixo
682  --com sinal e com um numero de bits pre-definido.
683  --Inicializa a matriz de covariancia.
684
685      P1K <= to_sfixed(10,P1K);
686      P2K <= to_sfixed(0,P2K);
687      P3K <= to_sfixed(0,P3K);
688      P4K <= to_sfixed(0,P4K);
689
690  --*****
691
692      inic_coef1 := inic_coef1 + 1;
693
694      end if;
695
696  --*****
697  --Sinais auxiliares p/ calculo de L1K.
698
699      L1Kaux1 <= resize(P1K1*phi1 + P2K1*phi2 + P3K1*phi3 + P4K1*phi4,L1Kaux1);
700      L1Kaux2 <= resize(phi1*(P1K1*phi1 + P2K1*phi2 + P3K1*phi3 + P4K1*phi4),L1Kaux2);
701      L1Kaux3 <= resize(phi2*(P5K1*phi1 + P6K1*phi2 + P7K1*phi3 + P8K1*phi4),L1Kaux3);
702      L1Kaux4 <= resize(phi3*(P9K1*phi1 + P10K1*phi2 + P11K1*phi3 + P12K1*phi4),L1Kaux4);
703      L1Kaux5 <= resize(phi4*(P13K1*phi1 + P14K1*phi2 + P15K1*phi3 + P16K1*phi4),L1Kaux5);
704
705  --*****
706  --*****
707  --Calcula um dos parametros do vetor de ganho de Kalman.
708
709      L1K <= resize(L1Kaux1/(1 + L1Kaux2 + L1Kaux3 + L1Kaux4 + L1Kaux5),L1K);
710
711  --*****
712
713      theta1K <= resize(theta1K1 + L1K*aux1_theta,theta1K);--Calculo do parametro theta1K.
714
715      k0_b <= theta1K;

```

```

716
717  --*****
718  --Atualiza parametros da matriz de covariancia.
719
720      P1K <= resize(P1K1 - L1K*(P1K1*phi1 + P5K1*phi2 + P9K1*phi3 + P13K1*phi4),P1K);
721      P2K <= resize(P2K1 - L1K*(P2K1*phi1 + P6K1*phi2 + P10K1*phi3 + P14K1*phi4),P2K);
722      P3K <= resize(P3K1 - L1K*(P3K1*phi1 + P7K1*phi2 + P11K1*phi3 + P15K1*phi4),P3K);
723      P4K <= resize(P4K1 - L1K*(P4K1*phi1 + P8K1*phi2 + P12K1*phi3 + P16K1*phi4),P4K);
724
725  --*****
726
727      end if;
728
729      end process rotina5;
730
731  rotina6: process (LiberaC_LK)
732
733      begin
734
735          if rising_edge (LiberaC_LK) then--Condicao p/ calculo dos parametros theta.
736
737              if inic_coef2 = 0 then--Condicao para o carregamento das condicoes iniciais
738                  --da matriz de covariancia(Matriz diagonal inicialmente).
739
740              --*****
741              --Atribui a cada um dos parametros "P" um valor convertido p/ o tipo ponto fixo
742              --com sinal e com um numero de bits pre-definido.
743              --Inicializa a matriz de covariancia.
744
745                  P5K <= to_sfixed(0,P5K);
746                  P6K <= to_sfixed(10,P6K);
747                  P7K <= to_sfixed(0,P7K);
748                  P8K <= to_sfixed(0,P8K);
749
750              --*****
751

```

```

752         inic_coef2 := inic_coef2 + 1;
753
754     end if;
755
756     --*****
757     --Sinais auxiliares p/ calculo de L2K.
758
759     L2Kaux1 <= resize(P5K1*phi1 + P6K1*phi2 + P7K1*phi3 + P8K1*phi4,L2Kaux1);
760     L2Kaux2 <= resize(phi1*(P1K1*phi1 + P2K1*phi2 + P3K1*phi3 + P4K1*phi4),L2Kaux2);
761     L2Kaux3 <= resize(phi2*(P5K1*phi1 + P6K1*phi2 + P7K1*phi3 + P8K1*phi4),L2Kaux3);
762     L2Kaux4 <= resize(phi3*(P9K1*phi1 + P10K1*phi2 + P11K1*phi3 + P12K1*phi4),L2Kaux4);
763     L2Kaux5 <= resize(phi4*(P13K1*phi1 + P14K1*phi2 + P15K1*phi3 + P16K1*phi4),L2Kaux5);
764
765     --*****
766     --*****
767     --Calcula um dos parametros do vetor de ganho de Kalman.
768
769     L2K <= resize(L2Kaux1/(1 + L2Kaux2 + L2Kaux3 + L2Kaux4 + L2Kaux5),L2K);
770
771     --*****
772
773     theta2K <= resize(theta2K1 + L2K*aux1_theta,theta2K);--Calculo do parametro theta2K.
774
775     k2_b <= theta2K;
776
777     --*****
778     --Atualiza parametros da matriz de covariancia.
779
780     P5K <= resize(P5K1 - L2K*(P5K1*phi2 + P1K1*phi1 + P9K1*phi3 + P13K1*phi4),P5K);
781     P6K <= resize(P6K1 - L2K*(P6K1*phi2 + P2K1*phi1 + P10K1*phi3 + P14K1*phi4),P6K);
782     P7K <= resize(P7K1 - L2K*(P7K1*phi2 + P3K1*phi1 + P11K1*phi3 + P15K1*phi4),P7K);
783     P8K <= resize(P8K1 - L2K*(P8K1*phi2 + P4K1*phi1 + P12K1*phi3 + P16K1*phi4),P8K);
784
785     --*****
786     end if;
787

```

```

788     end process rotina6;
789
790   rotina7: process (LiberaC_LK)
791
792     begin
793
794       if rising_edge (LiberaC_LK) then--Condicao p/ calculo dos parametros theta.
795
796         if inic_coef3 = 0 then--Condicao para o carregamento das condicoes iniciais
797             --da matriz de covariancia(Matriz diagonal inicialmente).
798
799         --*****
800         --Atribui a cada um dos parametros "P" um valor convertido p/ o tipo ponto fixo
801         --com sinal e com um numero de bits pre-definido.
802         --Inicializa a matriz de covariancia.
803
804         P9K <= to_sfixed(0,P9K);
805         P10K <= to_sfixed(0,P10K);
806         P11K <= to_sfixed(10,P11K);
807         P12K <= to_sfixed(0,P12K);
808
809         --*****
810         inic_coef3 := inic_coef3 + 1;
811
812       end if;
813
814       --*****
815       --Sinais auxiliares p/ calculo de L3K.
816
817       L3Kaux1 <= resize(P9K1*phi1 + P10K1*phi2 + P11K1*phi3 + P12K1*phi4,L3Kaux1);
818       L3Kaux2 <= resize(phi1*(P1K1*phi1 + P2K1*phi2 + P3K1*phi3 + P4K1*phi4),L3Kaux2);
819       L3Kaux3 <= resize(phi2*(P5K1*phi1 + P6K1*phi2 + P7K1*phi3 + P8K1*phi4),L3Kaux3);
820       L3Kaux4 <= resize(phi3*(P9K1*phi1 + P10K1*phi2 + P11K1*phi3 + P12K1*phi4),L3Kaux4);
821       L3Kaux5 <= resize(phi4*(P13K1*phi1 + P14K1*phi2 + P15K1*phi3 + P16K1*phi4),L3Kaux5);
822
823       --*****

```

```

824  --*****
825  --Calcula um dos parametros do vetor de ganho de Kalman.
826      L3K <= resize(L3Kaux1/(1 + L3Kaux2 + L3Kaux3 + L3Kaux4 + L3Kaux5),L3K);
827
828  --*****
829
830      theta3K <= resize(theta3K1 + L3K*aux1_theta,theta3K);--Calculo do parametro theta3K.
831
832      k3_b <= theta3K;
833
834  --*****
835  --Atualiza parametros da matriz de covariancia.
836
837      P9K <= resize(P9K1 - L3K*(P9K1*phi3 + P1K1*phi1 + P5K1*phi2 + P13K1*phi4),P9K);
838      P10K <= resize(P10K1 - L3K*(P10K1*phi3 + P2K1*phi1 + P6K1*phi2 + P14K1*phi4),P10K);
839      P11K <= resize(P11K1 - L3K*(P11K1*phi3 + P3K1*phi1 + P7K1*phi2 + P15K1*phi4),P11K);
840      P12K <= resize(P12K1 - L3K*(P12K1*phi3 + P4K1*phi1 + P8K1*phi2 + P16K1*phi4),P12K);
841
842  --*****
843      end if;
844
845      end process rotina7;
846
847  rotina8: process (Liberac_LK)
848
849      begin
850
851          if rising_edge (Liberac_LK) then--Condicao p/ calculo dos parametros theta.
852
853              if inic_coef4 = 0 then--Condicao para o carregamento das condicoes iniciais
854                  --da matriz de covariancia(Matriz diagonal inicialmente).
855
856              --*****
857              --Atribui a cada um dos parametros "P" um valor convertido p/ o tipo ponto fixo
858              --com sinal e com um numero de bits pre-definido.
859              --Inicializa a matriz de covariancia.

```

```

860
861     P13K <= to_sfixed(0,P13K);
862     P14K <= to_sfixed(0,P14K);
863     P15K <= to_sfixed(0,P15K);
864     P16K <= to_sfixed(10,P16K);
865
866     --*****
867
868     inic_coef4 := inic_coef4 + 1;
869
870     end if;
871
872     --*****
873     --Sinais auxiliares p/ calculo de L4K.
874
875     L4Kaux1 <= resize(P13K1*phi1 + P14K1*phi2 + P15K1*phi3 + P16K1*phi4,L4Kaux1);
876     L4Kaux2 <= resize(phi1*(P1K1*phi1 + P2K1*phi2 + P3K1*phi3 + P4K1*phi4),L4Kaux2);
877     L4Kaux3 <= resize(phi2*(P5K1*phi1 + P6K1*phi2 + P7K1*phi3 + P8K1*phi4),L4Kaux3);
878     L4Kaux4 <= resize(phi3*(P9K1*phi1 + P10K1*phi2 + P11K1*phi3 + P12K1*phi4),L4Kaux4);
879     L4Kaux5 <= resize(phi4*(P13K1*phi1 + P14K1*phi2 + P15K1*phi3 + P16K1*phi4),L4Kaux5);
880
881     --*****
882     --*****
883     --Calcula um dos parametros do vetor de ganho de Kalman.
884     L4K <= resize(L4Kaux1/(1 + L4Kaux2 + L4Kaux3 + L4Kaux4 + L4Kaux5),L4K);
885
886     --*****
887
888     theta4K <= resize(theta4K1 + L4K*aux1_theta,theta4K);--Calculo do parametro theta4K.
889
890     k4_b <= theta4K;
891
892     --*****
893     --Atualiza parametros da matriz de covariancia.
894
895     P13K <= resize(P13K1 - L4K*(P13K1*phi4 + P1K1*phi1 + P5K1*phi2 + P9K1*phi3),P13K);

```

```

896         P14K <= resize(P14K1 - L4K*(P14K1*phi4 + P2K1*phi1 + P6K1*phi2 + P10K1*phi3),P14K);
897         P15K <= resize(P15K1 - L4K*(P15K1*phi4 + P3K1*phi1 + P7K1*phi2 + P11K1*phi3),P15K);
898         P16K <= resize(P16K1 - L4K*(P16K1*phi4 + P4K1*phi1 + P8K1*phi2 + P12K1*phi3),P16K);
899
900         --*****
901
902         end if;
903
904         end process rotina8;
905
906         rotina9: process (LibCalcUk)
907         begin
908
909             if rising_edge (LibCalcUk) then--Condicao p/ o calculo do sinal de controle.
910
911                 --s0 = (1-exp(-Ts/tau_mf))/(b0(k)+b1(k));
912                 s0 <= resize((0.3935/(k3_b + k4_b)),s0);--Calculo de s0 usado na sintonia do
913                                                         --controlador PID digital auto-ajustavel.
914                 uk_b <= resize((uk1_b + s0*(ek_b + k0_b*ek1_b + k2_b*ek2_b)),uk_b);--Calculo do
915                                                         --sinal de controle.
916                 yk2 <= yk1;--Atribui yk1 a yk2 p/ ser atribuido a phi2 na proxima amostragem.
917                 yk1 <= valatual_b;--Atribui valatual_b a yk1 p/ ser atribuido a phi1 na proxima amostragem.
918
919             end if;
920
921         end process rotina9;
922
923     end ADConverter;

```

APÊNDICE D – CÓDIGO PARA SIMULAÇÃO DO INDIRECT SELF TUNING NO SOFTWARE MATLAB®

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Indirect Self-Tuning
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all; close all; clc;

% Parâmetros do modelo contínuo (Conversor NPCm c/ filtro LC)
C = 4.3*10^(-6);
L = 3.3*10^(-3);
Fs = 20e3; %Hz
Ts = 1/Fs; %s
E = 30;
R = 50;

% F.T. contínua
Gps = tf([(E/2)/(L*C)], [1 1/(R*C) 1/(L*C)]);

% F.T. discreta
Gpz = c2d(Gps, Ts, 'zoh'); %  $G_p(z^{-1}) = B(z^{-1}) * z^{-1} / A(z^{-1})$ 
Bz = Gpz.num{1};
Az = Gpz.den{1};

% Resposta em M.A. figure(1); step(Gps,Gpz);
title('Resposta ao degrau em M.A.');
```

legend('Contínuo','Discreto');

```

% Simulacao do controlador em M.F.

% Gera referência senoidal
tfinal = 0.04;
t = 0:0.00005:tfinal; N = length(t);
yr = 15*sin(2*pi*50*t); figure(2); plot(t,yr);

```

```

% Perturbação de carga
d(1:N)=0.0;
d=1+1*(square(2*pi*(Fs/360)*t +1*pi));

% Simula planta com controlador
% Condições iniciais do Algoritmo dos MQR
theta = [Az(2) Az(3) Bz(2) Bz(3)]'; % [a1hat a2hat b0hat]
P = 10*eye(4); % Quanto maior a ponderacao, mais rápida a convergência
               % do estimador. Utilizar valores menores caso já se
               % conheça parâmetros iniciais adequados para theta.

for k=1:2,
    y(k)=0; u(k)=0; e(k)=0; % condições iniciais

    % Parâmetros do Algoritmo MQR
    alhat(k)=theta(1); a2hat(k)=theta(2);
    b0hat(k)=theta(3); blhat(k)=theta(4);
end

for k=3:N,
    y(k)=-Az(2)*y(k-1) -Az(3)*y(k-2) ...
    +Bz(2)*u(k-1) +Bz(3)*u(k-2) ...
    +d(k) +Az(2)*d(k-1) +Az(3)*d(k-2); %perturbacao de carga

    % Estimador MQR
    phi = [ -y(k-1) -y(k-2) u(k-1) u(k-2)]'; L= (P*phi)/(1+phi'*P*phi);
    theta=theta +L*( y(k) -phi'*theta ); % Novos parametros alhat(k)=theta(1); a2hat(k)=theta(2);
    b0hat(k)=theta(3); blhat(k)=theta(4);
    P=(eye(4,4)-L*phi')*P; % Atualiza a Matriz de Covariância
    % FIM do MQR

```

```
% Controlador PID digital auto-ajustável

% u(k) = u(k-1) +s0*e(k) +s1*e(k-1) +s2*e(k-2)
% s0 = (1 -exp(-Ts/tau_mf)/(b0hat+b1hat) ;
% b0hat+b1hat=ganho_estatico_estimado

% s1 = s0*a1hat ; a1hat=a1_estimado
% s2 = s0*a2hat ; a2hat=a2_estimado

e(k) = yr(k)-y(k);
tau_mf = 0.1e-3; %segundos
s0 = (1-exp(-Ts/tau_mf))/(b0hat(k)+b1hat(k));
u(k)=u(k-1) +s0*(e(k) +a1hat(k)*e(k-1) +a2hat(k)*e(k-2));

end

% Resultados em milisegundos t=1e3*t;
figure(3); subplot(211),
plot(t,yr,':k',t,y,'b',t,d,':g'); legend('y_r(t)','y(t)','Carga');title(['Resultados em milisegundos'])
ylabel('Saída');
subplot(212),plot(t,u,'r');ylabel('Controle');xlabel('Tempo
(ms)');
```