

UNIVERSIDADE DO ESTADO DE SANTA CATARINA - UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS - CCT
DEPARTAMENTO DE ENGENHARIA ELÉTRICA - DEE

HALLAN WILLIAM VEIGA

**PROGRAMAÇÃO VHDL DE DISPOSITIVO FPGA PARA APLICAÇÃO EM
MODULAÇÃO PWM SENOIDAL DE CONVERSORES MULTINÍVEIS**

Joinville – SC

2013

HALLAN WILLIAM VEIGA

**PROGRAMAÇÃO VHDL DE DISPOSITIVO FPGA PARA APLICAÇÃO EM
MODULAÇÃO PWM SENOIDAL DE CONVERSORES MULTINÍVEIS**

Trabalho de conclusão de curso referente à disciplina TCC II sobre o Estudo e programação de Técnicas de Modulação PWM utilizando FPGA para Implementação em Conversores Multiníveis.

Orientador: Prof. MSc. Joselito Anastácio Heerdt

Joinville – SC

2013

HALLAN WILLIAM VEIGA

TRABALHO DE CONCLUSÃO DE CURSO – TCC II
PROGRAMAÇÃO VHDL DE DISPOSITIVO FPGA PARA APLICAÇÃO EM
MODULAÇÃO PWM SENOIDAL DE CONVERSORES MULTINÍVEIS

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

Banca Examinadora

Orientador: _____

Prof. MSc. Joselito Anastácio Heerdt
Universidade do Estado de Santa Catarina-UDESC

Membro: _____

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

Membro: _____

Prof. MSc. Fabrício Noveletto
Universidade do Estado de Santa Catarina-UDESC

Joinville, julho de 2013

AGRADECIMENTOS

Agradeço primeiramente aos meus pais Geraldo Veiga e Jucélia Veiga que durante a realização desse trabalho me ofereceram suporte, carinho e apoio.

Ao meu orientador Prof. MSc. Joselito Anastácio Heerdt pelos ensinamentos e pela dedicação ao me orientar.

Ao Prof. Dr. Alessandro Luiz Batschauer que leciona à disciplina de Eletrônica de Potência e me motivou a realizar o trabalho nesta área.

Aos demais professores, bolsistas e mestrandos do nPEE da UDESC que além da amizade e companheirismo ajudaram respondendo a dúvidas que surgiam no desenvolvimento do trabalho.

RESUMO

A evolução de circuitos lógicos programáveis e a necessidade da otimização dos circuitos de potência favorecem a aplicação dos FPGA's (*Field Programmable Gate Array*) em conversores multiníveis estudados na área de eletrônica de potência.

Este trabalho apresenta estratégias de modulação PWM (*Pulse Width Modulation*) que podem ser descritas por uma linguagem de programação. O FPGA formará um conjunto lógico digital através dessa linguagem descrita, que no caso é utilizada a linguagem VHDL (*Very High Speed Integrated Circuit Hardware Description Language*). Por sua vez, esse conjunto realizará a comutação das chaves de conversores multiníveis. Outro objetivo deste trabalho é ser uma fonte de pesquisa e auxílio para futuros trabalhos que utilizem circuitos lógicos programáveis em conversores multiníveis.

O funcionamento de conversores multiníveis, o entendimento das diversas técnicas de modulação por largura de pulsos, a apresentação em passos da programação de um FPGA e a sua posterior implementação em conversores multiníveis são tratados no trabalho.

Palavras-chave: *FPGA, PWM, conversor multinível, VHDL, modulação multinível*

ABSTRACT

The evolution of programmable logic circuits and the need for optimization of power circuits allows the application of FPGA's (Field Programmable Gate Array) in multilevel converters studied on power electronics.

This work presents strategies for PWM (Pulse Width Modulation) which can be described by a programming language. The FPGA will form a logical set through this digital language described, in which case the language used is VHDL (Very High Speed Integrated Circuit Hardware Description Language). In turn, this set will perform the switching keys of multilevel converters. Another objective is to be a source of research and support for future studies that utilize programmable logic circuits on multilevel converters.

The operation of multilevel converters, the understanding of different modulation techniques of pulse width, the presentation of the steps in programming an FPGA and its subsequent implementation in multilevel converters are treated at work.

Keywords: *FPGA, PWM, multilevel converter, VHDL, multilevel modulation.*

LISTA DE FIGURAS

Figura 1 - Princípio do modulador PWM senoidal.....	14
Figura 2 - Variação dos sinais ao longo do tempo.....	14
Figura 3 - Gráfico de semicondutores encontrados no mercado.....	17
Figura 4 - Topologia do conversor NPC de 3 níveis.....	18
Figura 5 - Quadrantes de funcionamento conversor NPC.....	19
Figura 6 - Etapas de operação do conversor NPC nos diferentes quadrantes.....	20
Figura 7 - Forma de onda da tensão de saída do conversor NPC à 3 níveis.....	23
Figura 8 - Conversor monofásico NPC à 5 níveis.....	24
Figura 9 - Formas de onda da tensão de saída do conversor NPC 5 níveis.....	25
Figura 10 - Conversor NPC 5 níveis trifásico.....	26
Figura 11 - Formas de onda das tensões de saída do conversor NPC trifásico 5 níveis.....	26
Figura 12 - Conversor <i>Flying Capacitor</i> de cinco níveis.....	28
Figura 13 - Conversor em cascata H-Bridge.....	29
Figura 14 - Conversor monofásico 5 níveis tipo cascata.....	30
Figura 15 - Conversor de m níveis de conjuntos meia ponte em cascata.....	31
Figura 16 - Modulação IPD para 5 níveis.....	32
Figura 17 - Modulação POD para 5 níveis.....	33
Figura 18 - Modulação APOD para 5 níveis.....	34
Figura 19 - Modulação PSD para 5 níveis.....	35
Figura 20 - Estrutura Interna de um FPGA.....	37
Figura 21 - Sequência de etapas do processo de um projeto com FPGA.....	38
Figura 22 - Kit DE0-nano Altera.....	42
Figura 23 - Instalação do software de driver.....	43
Figura 24 - Criação de um novo projeto.....	44
Figura 25 - Seleção do modelo do kit a ser trabalhado.....	44
Figura 26 - Configuração das ferramentas de simulação e análise temporal.....	45
Figura 27 - Seleção de um novo projeto em VHDL.....	46
Figura 28 - Projeto de uma porta lógica AND.....	46
Figura 29 - Mapa com todas as entradas e saídas da placa do FPGA.....	47
Figura 30 - Programação do código e comunicação com o hardware.....	48
Figura 31 - Seleção de um novo projeto usando circuitos lógicos.....	49
Figura 32 - Seleção de uma porta lógica AND.....	49
Figura 33 - Seleção das entradas e saídas.....	50
Figura 34 - Seleção do arquivo do projeto para simulação.....	51
Figura 35 - Biblioteca contendo os projetos de simulação compilados.....	51
Figura 36 - Tela de simulação com as entradas e saídas do projeto.....	52
Figura 37 - Tela de simulação com as entradas e saídas do projeto.....	53
Figura 38 - Fluxograma de modulação de uma saída PWM.....	56
Figura 39 - Fluxograma de modulação PWM para conversor NPC três níveis monofásico ..	59
Figura 40 - Conversor monofásico ponte completa.....	65
Figura 41 - Comando das chaves do conversor monofásico ponte completa.....	66
Figura 42 - Formas de onda de tensão na saída do FPGA para modulação 3 níveis.....	67

Figura 43 - Tempo-morto na modulação três níveis	67
Figura 44 - Formas de onda experimentais da tensão na saída do conversor ponte completa	68
Figura 45 - Conversor NPCm monofásico à 5 níveis	69
Figura 46 - Chaveamento do conversor à cinco níveis	70
Figura 47 - Circuito lógico responsável pela modulação do conversor NPCm à 5 níveis	70
Figura 48 - Formas de onda de tensão obtidas nos pinos de saída do FPGA para o Conversor NPCm 5 níveis.....	71

LISTA DE ABREVIATURAS

PWM	Pulse Width Modulation
MOSFET	Metal Oxide Semiconductor Field-Effect Transistor
IGBT	Insulate Gate Bipolar Transistor
GTO	Gate Turn Off
SCR	Silicon Controlled Rectifier
NPC	Neutral Point Clamped
IPD	In Phase Disposition
POD	Phase Opposition Disposition
APOD	Alternative Phase Opposition Disposition
PSD	Phase Shifted Disposition
FPGA	Field Programmable Gate Array
DSP	Digital Signal Processor
VHDL	Very High Speed Integrated Circuit Hardware Description Language
NPC	Neutral Point Clamped
SPWM	Sinusoidal <i>Pulse Width Modulation</i>

SIMBOLOGIA

m	Número de níveis
m_a	Índice de modulação de amplitude
m_f	Índice de modulação de frequência
$n^\circ \text{ cap}$	Número de capacitores do conversor de capacitores flutuantes
V_y	Tensão em cada fonte do conversor NPC
V_{ref}	Tensão de pico da referência senoidal
V_{port}	Tensão de pico da portadora
f_{port}	Frequência da portadora
f_{mod}	Frequência da moduladora
n	Número de fontes
f_{clock}	Frequência do clock
φ	Diferença de fase das portadoras

SUMÁRIO

1	INTRODUÇÃO	13
2	CONVERSORES MULTINÍVEIS	16
2.1	CONVERSORES COM GRAMPEAMENTO A DIODO	17
2.2	CONVERSOR COM GRAMPEAMENTO A CAPACITOR FLUTUANTE – <i>FLYING CAPACITOR</i>	27
2.3	CONVERSORES EM CASCATA	29
3	ESTRATÉGIAS DE MODULAÇÃO	32
3.1	DISPOSIÇÃO EM FASE (<i>IPD-IN PHASE DISPOSITION</i>)	32
3.2	DISPOSIÇÃO EM OPOSIÇÃO DE FASE (<i>POD - PHASE OPPOSITION DISPOSITION</i>)	33
3.3	DISPOSIÇÃO EM OPOSIÇÃO DE FASES ALTERNADAS (<i>APOD-ALTERNATIVE PHASE OPPOSITION DISPOSITION</i>)	33
3.4	DISPOSIÇÃO DE FASES DESLOCADAS (<i>PSD-PHASE SHIFTED DISPOSITION</i>)	34
4	FPGA - FIELD PROGRAMMABLE GATE ARRAY	36
4.1	METODOLOGIA DE IMPLEMENTAÇÃO	36
4.1.1	Aspectos gerais do FPGA	37
4.1.2	Etapas de um projeto utilizando FPGA	38
4.1.3	Especificação e entrada de projetos	38
4.1.4	Síntese lógica	39
4.1.5	Posicionamento e roteamento	39
4.1.6	Verificação e teste	39
4.1.7	Programação do FPGA	40
5	DESENVOLVIMENTO DO PROJETO	41
5.1	KIT DE DESENVOLVIMENTO DE0-NANO ALTERA	41
5.2	SEQUÊNCIA DE PASSOS DE PROGRAMAÇÃO, GRAVAÇÃO E CONFIGURAÇÃO DO KIT DE0-NANO ALTERA	42
5.3	PASSOS INICIAIS PARA SIMULAÇÃO DE UM PROJETO UTILIZANDO PORTAS LÓGICAS	48
5.4	ALGORITMO DE PROGRAMAÇÃO	53
5.4.1	Algoritmo de uma modulação simples PWM	53
5.4.2	Algoritmo da modulação do conversor NPC três níveis	57
5.5	PROGRAMA EM LINGUAGEM VHDL DE MODULAÇÃO PWM PARA CONVERSOR NPC TRÊS NÍVEIS MONOFÁSICO COM TEMPO-MORTO	60

5.6	IMPLEMENTAÇÃO EXPERIMENTAL EM CONVERSORES CHAVEADOS	64
6	CONCLUSÕES FINAIS	72
	REFERÊNCIAS BIBLIOGRÁFICAS.....	73

1 INTRODUÇÃO

O crescimento da demanda energética, a busca pela obtenção de novas fontes de energia e o desenvolvimento de novas tecnologias exigem pesquisas e estudos tanto em geração como em qualidade energética. A exigência por uma energia confiável e de qualidade é evidente em cargas que necessitam de um fornecimento livre de interrupções e perturbações como exemplo equipamentos hospitalares e sistemas de comunicação.

No ramo da eletrônica de potência há estudos com conversores multiníveis. A topologia de tais circuitos permite que haja uma baixa distorção de harmônicos de tensão na saída desses conversores, isso possibilita um maior aproveitamento de energia que é fornecida à carga. O desenvolvimento dessa área da eletrônica nas últimas décadas favoreceu que houvesse uma composição com a área de eletrônica digital de maneira que o controle dos circuitos analógicos ofereceu espaço aos microcontroladores e circuitos com lógica programável. Existe uma tendência para o uso de tais dispositivos no controle de fontes chaveadas. Algumas vantagens de tais dispositivos são: flexibilidade para o uso em vários tipos de conversores permitindo alterações de operação por *software* sem a necessidade de alteração de *hardware*, confiabilidade, compactação dos circuitos conversores e redução de custos do projeto.

Nesse contexto surgem os FPGAs (*Field Programmable Gate Array*), circuitos com lógica programável que através da linguagem de programação montam um conjunto lógico digital que é capaz de atender as aplicações desejadas. Tais circuitos podem ser utilizados em conversores multiníveis que necessitam de um complexo sistema de controle das chaves. Com o FPGA tal sistema se tornaria mais compacto e confiável e flexível, permitindo total liberdade de sequência e intertravamento dos interruptores.

Uma maneira muito utilizada na comutação de chaves de conversores é a chamada Modulação PWM (*Pulse Width Modulation*). Essa é caracterizada por ser uma modulação em largura de pulso, ou seja, dados períodos fixos no domínio do tempo e alterando a razão cíclica obtém-se um sinal de saída [1].

Para a obtenção de uma modulação PWM senoidal, utilizada em conversores CC-CA, pode ser utilizado um comparador que relaciona um sinal de entrada como,

por exemplo, uma onda triangular com um sinal de referência senoidal [1] [2], conhecido por SPWM (Sinusoidal *Pulse Width Modulation*). Na saída do comparador é observado um trem de pulsos que irá variar sua largura de acordo com a variação do sinal de referência como mostram a Figura 1 e a Figura 2.

Figura 1 - Princípio do modulador PWM senoidal

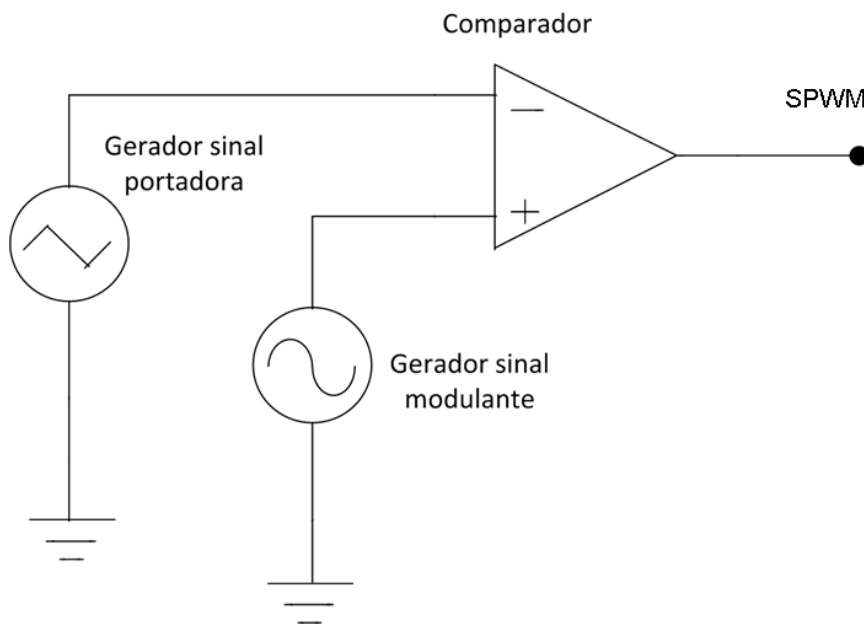
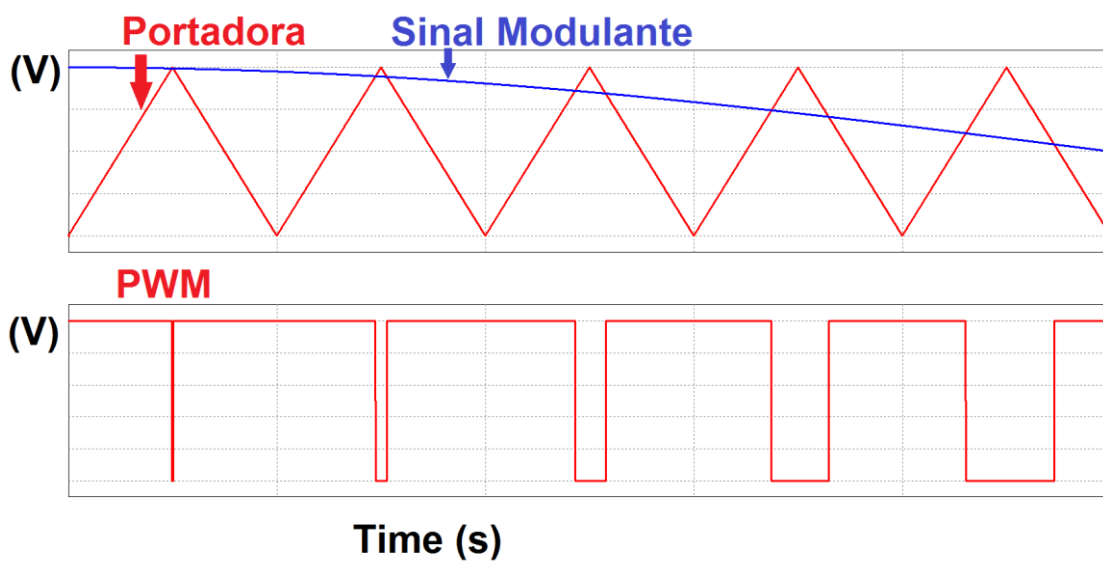


Figura 2 - Variação dos sinais ao longo do tempo



Através deste método apresentado podem ser implementados diversos tipos de modulação PWM: IPD (*In Phase Disposition*), POD (*Phase Opposition Disposition*), APOD (*Alternative Phase Opposition Disposition*), etc. Com o auxílio de um FPGA essas modulações podem ser aplicadas em circuitos comutados [1][2][3].

Cada conversor necessita de um diferente algoritmo de comando para as chaves semicondutoras. Nesse algoritmo são determinados a sequência e o tempo de comutação dos seus interruptores.

Este trabalho tem os seguintes objetivos: a) apresentar uma análise teórica de técnicas utilizadas para a modulação SPWM, b) programação e simulação de duas estratégias de modulação por largura de pulso, uma estratégia baseada na disposição das portadoras (POD) e uma outra baseada no deslocamento de fase das portadoras (PSD), c) aplicação das estratégias mencionadas no item 'b' em diferentes tipos de conversores multiníveis, em específico o conversor NPC, d) estudo dos conversores multiníveis bem como os circuitos responsáveis pela modulação dos interruptores, e) entendimento completo do FPGA desde a sua estrutura interna e o estudo da linguagem de programação VHDL.

2 CONVERSORES MULTINÍVEIS

Os conversores multiníveis fazem parte de uma classe recente de conversores que são utilizados no acionamento de motores e aplicações de alta potência [4]. Esses são responsáveis pelo chaveamento de fontes de tensão contínua produzindo na saída um sinal alternado com uma frequência determinada pelo período de comutação das chaves.

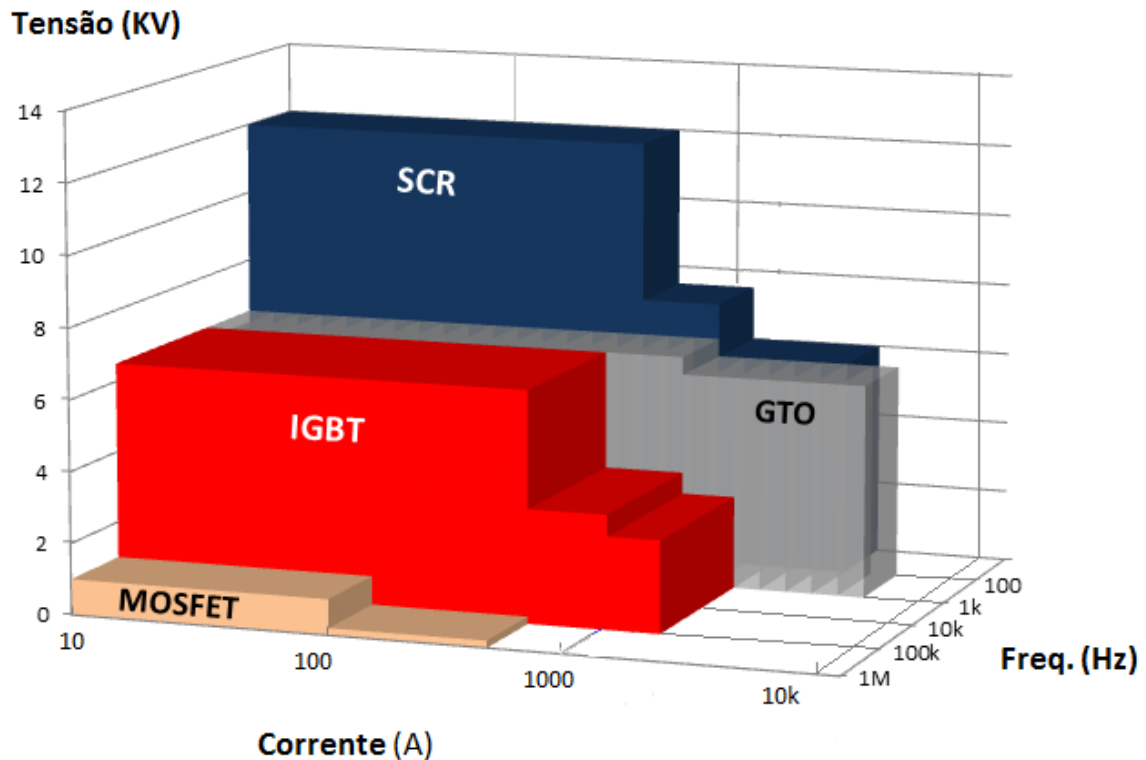
A sequência de chaveamento e o tempo de comutação são definidos pela estratégia de modulação empregada. Essas estratégias objetivam fundamentalmente produzir um sinal com vários níveis de tensão na saída.

Pelo fato dos conversores serem aplicados em motores de corrente alternada há uma necessidade de que o sinal de saída possua uma baixa taxa de harmônicas que causam perdas e vibrações no motor. O emprego da modulação PWM junto aos conversores multiníveis reduzem essa quantidade e aumentam a eficiência do sistema.

A estrutura das chaves pode ser formada por transistores com diodos em antiparalelo. Os tipos de transistores são variados e escolhidos de acordo com a necessidade da aplicação que é desejada. Os principais tipos são os MOSFET's e os IGBT's (*Insulate Gate Bipolar Transistor*). O transistor MOSFET possui um tempo de comutação reduzido por isso ele é utilizado para aplicações de alta frequência de comutação (por volta de 100 kHz a 1 MHz). Já o transistor IGBT é comumente usado para aplicações com maior potência e com frequência de comutação mais baixa justamente por possuir um tempo de comutação maior que o MOSFET [1][2].

Existem outros dispositivos semicondutores como os tiristores SCR (*Silicon Controlled Rectifier*) que são utilizados para aplicações com altíssima potência e são em grande parte aplicados em subestações retificadoras de energia. Os GTO's (*Gate Turn Off*) que são utilizados para alta potência possuem um comando extra que é capaz de bloquear a condução. A Figura 3 relaciona todos esses dispositivos citados encontrados no mercado em um gráfico de tensão e corrente por frequência de comutação.

Figura 3 - Gráfico de semicondutores encontrados no mercado



Dados baseados em [2]

Existem três principais topologias de conversores multiníveis que são frequentemente encontradas na literatura:

- Conversor com diodos de grampeamento do neutro (Neutral Point Clamped-NPC)
- Conversor de Capacitores Flutuantes (*Flying-Capacitor*)
- Conversor em cascata

2.1 CONVERSORES COM GRAMPEAMENTO A DIODO.

Um conversor muito conhecido que possui a estrutura de diodos grampeados é o conversor NPC (*Neutral Point Clamped*). Esse é uma evolução do conversor de dois níveis. Suas principais características são a diminuição da derivada de tensão e redução de conteúdo harmônico na tensão de saída possibilitando a redução ou eliminação de filtros passivos [1][2][5].

O cálculo da tensão (V_y) de cada fonte que compõe o barramento CC, considerando que o conversor possui m níveis, pode ser realizado a partir da equação (1).

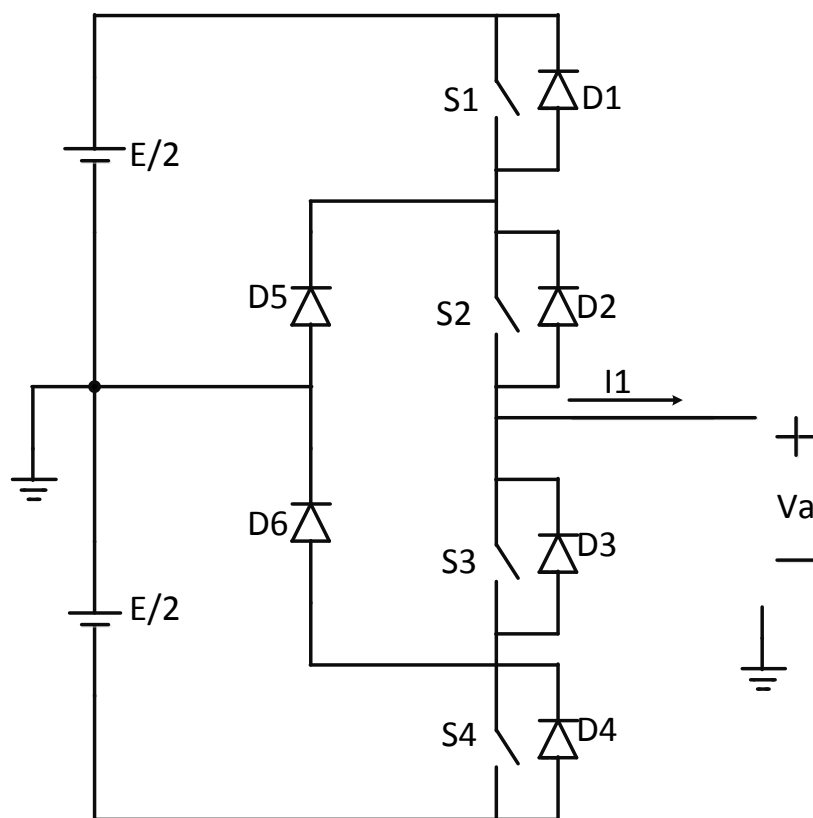
$$V_y = \frac{V_{cc}}{m - 1} \quad (1)$$

Onde: V_{cc} - tensão total do barramento

m - número de níveis

A estrutura básica do NPC à três níveis apresenta quatro transistores por braço (S1,S2,S3,S4) funcionando como chave, dois diodos por braço (D5 e D6) além dos que ficam em anti-paralelo com os transistores, como ilustra a Figura 4.

Figura 4 - Topologia do conversor NPC de 3 níveis



Na Figura 4 foi apresentado um conversor monofásico CC-CA à três níveis. Vale salientar que em um conversor trifásico desse modelo a tensão de linha possuiria cinco níveis.

Nesse circuito a tensão de saída é medida em V_a . Ela apresenta três diferentes valores de tensão: $E/2$, 0 , $-E/2$. Para obter cada valor citado de tensão as chaves devem apresentar um determinado estado de comutação como descrito na Tabela 1.

Tabela 1 - Estado comutação conversor NPC

TENSÃO FASE-NEUTRO SAÍDA	$E/2$ (V)	0 (V)	$-E/2$ (V)
CHAVES FECHADAS	S1	S2	S3
	S2	S3	S4
CHAVES ABERTAS	S3	S1	S1
	S4	S4	S2

Quando uma chave está fechada não significa que a corrente do conversor irá necessariamente percorrer por ela. Para facilitar o entendimento do funcionamento do circuito são ilustradas as etapas de condução em cada quadrante. Na Figura 5 é apresentada a divisão dos quadrantes e na Figura 6 são apresentadas as diferentes etapas de condução nos determinados quadrantes [6][7].

Figura 5 - Quadrantes de funcionamento conversor NPC

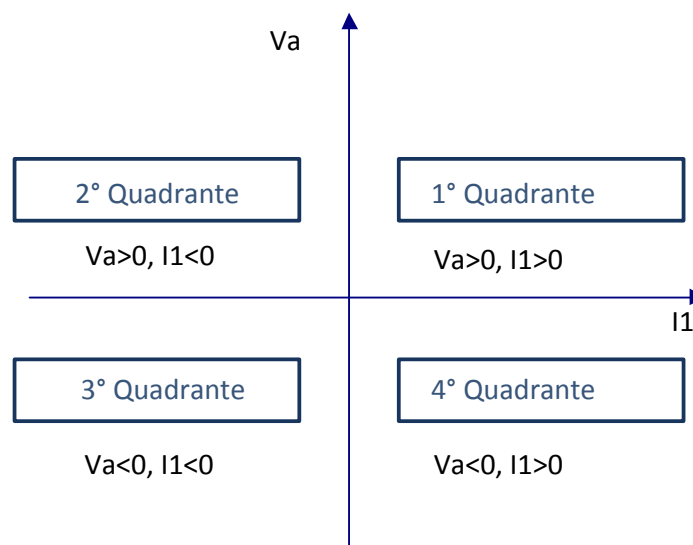
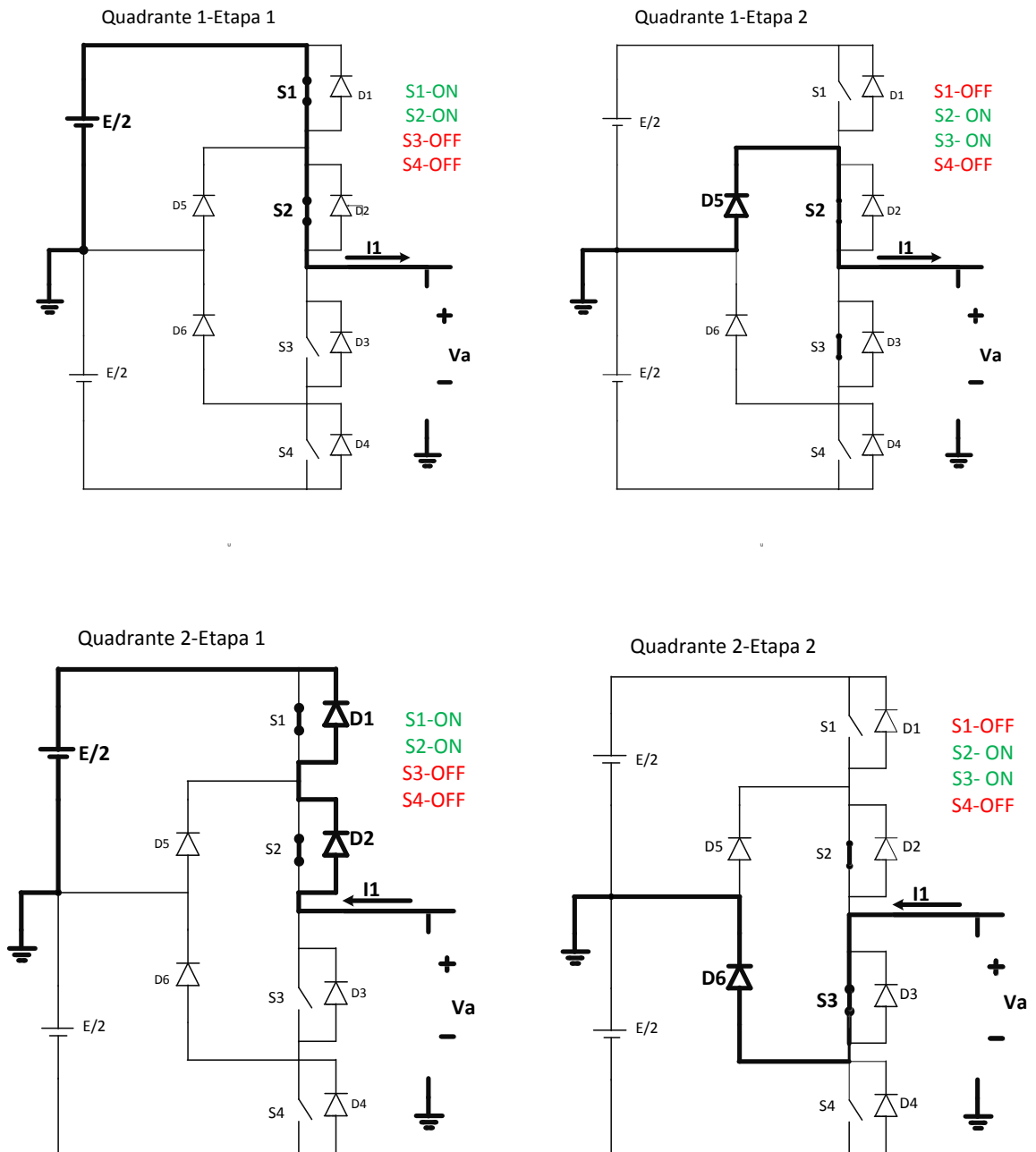
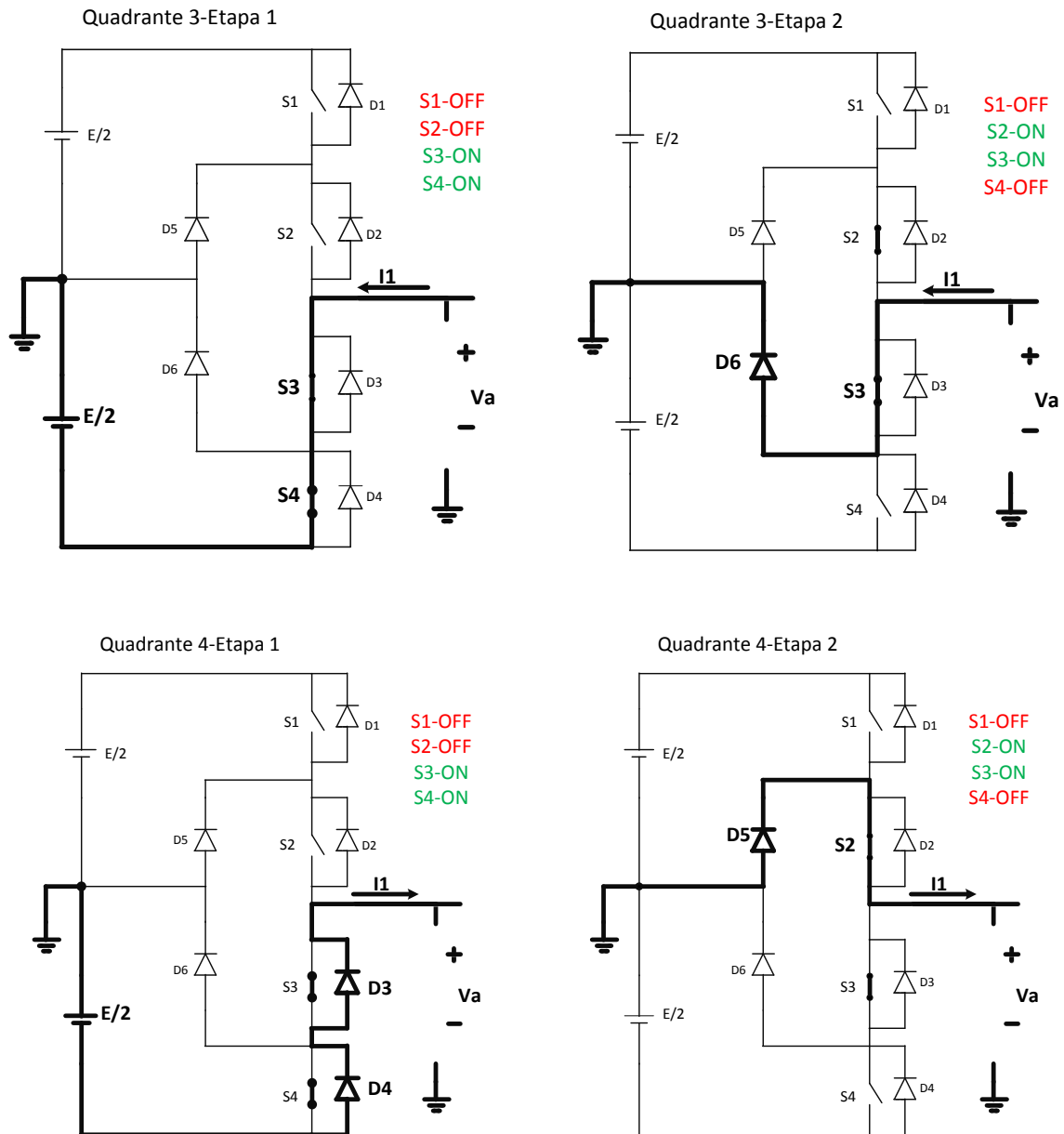


Figura 6 - Etapas de operação do conversor NPC nos diferentes quadrantes





Como ilustrado em parte da Figura 6, que diz respeito as etapas do conversor no primeiro quadrante, a abertura da chave S1 estabelece um novo caminho para a passagem de corrente. Por ser uma carga indutiva sabe-se que existe uma inércia de corrente no indutor, assim o diodo D5 passa a conduzir que por sua vez chaveia uma tensão igual a de referência na saída.

Na eletrônica de potência, especificamente no estudo de conversores que utilizam modulação PWM, são determinados dois índices que são levados em consideração na análise do funcionamento dos conversores. Existem o índice de modulação de amplitude e o índice de modulação de frequência [8].

O índice de modulação de amplitude é calculado a partir da equação (2) que relaciona o valor de pico da tensão do sinal de referência da senóide com o valor de pico de tensão da portadora responsável pela comutação das chaves.

$$m_a = \frac{V_{ref}}{(m-1)V_{port}} \quad (2)$$

Pode não parecer claro em um primeiro momento, pois ainda não foi abordado a fundo o assunto da obtenção de um sinal PWM. Mas basicamente é feita uma comparação entre uma senóide e um sinal triangular (portadora). As características destes dois sinais são determinantes para o cálculo tanto do índice de modulação de amplitude como o índice de modulação de frequência.

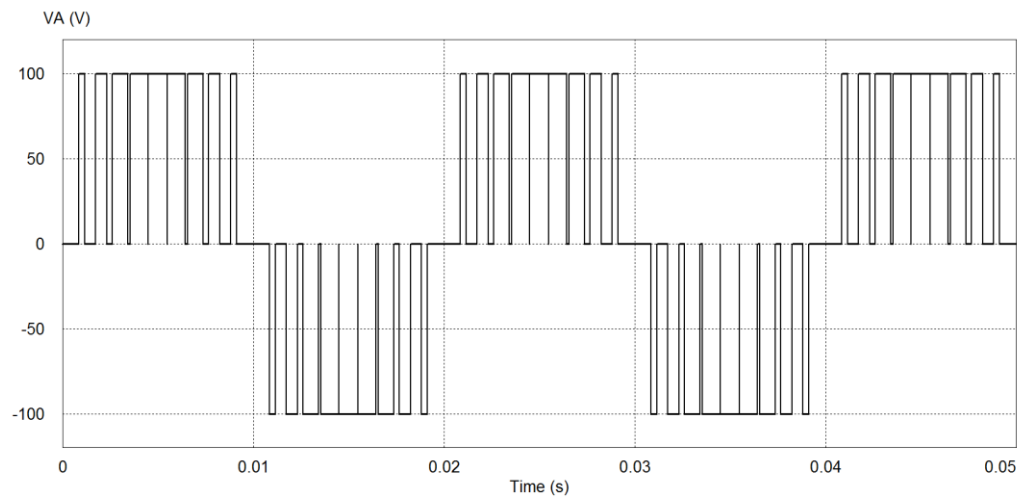
O índice de modulação de frequência, calculado pela equação (3), relaciona a frequência de comutação das chaves ou da frequência da portadora com a frequência da moduladora do conversor.

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

A taxa de distorção harmônica sofre influência quando há uma alteração dos índices de modulação de amplitude e frequência. Quanto maior o valor do índice de modulação de amplitude menor é essa taxa. O aumento do número de níveis também causa uma diminuição na taxa de distorção harmônica.

A Figura 7 ilustra o gráfico do sinal da saída V_a do conversor NPC para uma frequência de modulação $f_{port}=1000$ Hz, índice de modulação em frequência $m_f=20$ e $m_a=0,9$. Pode ser observado claramente a utilização da modulação PWM.

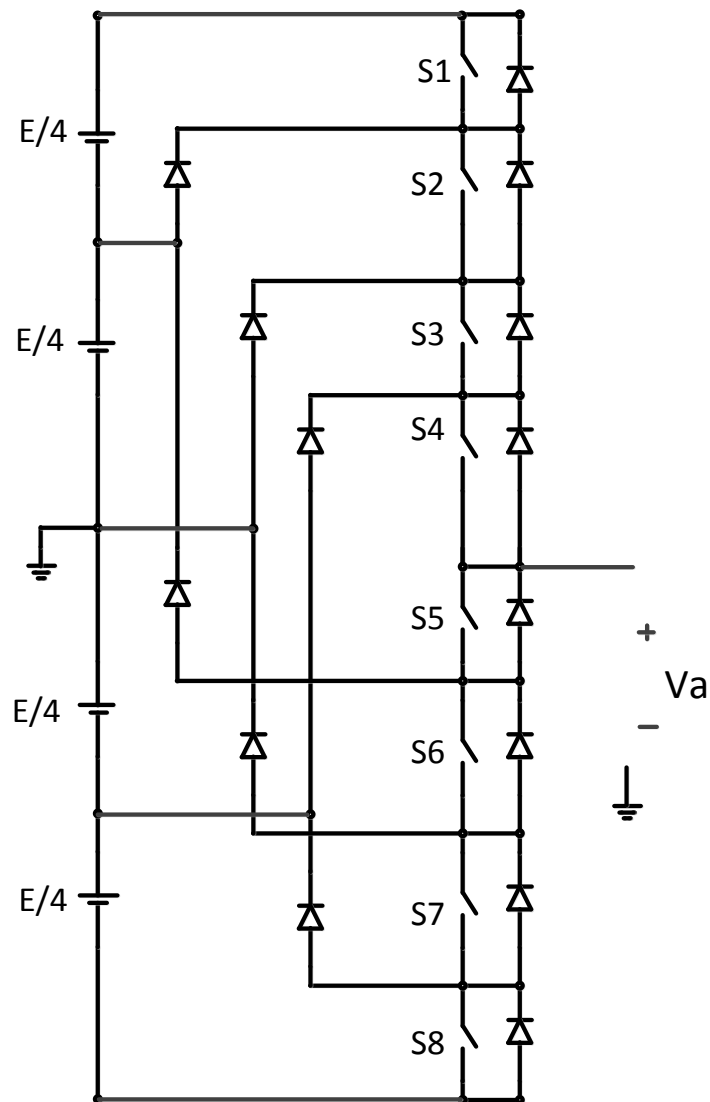
Figura 7 - Forma de onda da tensão de saída do conversor NPC à 3 níveis



Se for calculada uma média em instantes de tempo do módulo da tensão obtém-se uma senóide. Na Figura 7 o período da moduladora é de 0,02 s. correspondente a frequência de 50 Hz.

Fazendo analogia à estrutura de três níveis, é apresentado na Figura 8 um conversor à cinco níveis monofásico. Pode ser notado que o número de chaves é duplicado e o número de diodos de grampeamento do barramento é triplicado. O aumento do número de níveis influi diretamente na qualidade do sinal pois há uma redução na taxa de distorção harmônica [9].

Figura 8 - Conversor monofásico NPC à 5 níveis

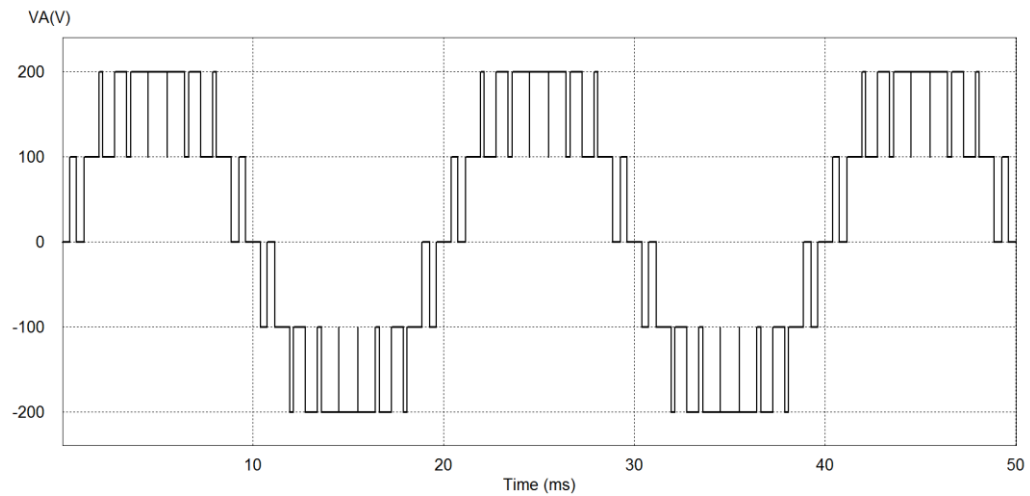


Para obter cinco valores diferentes de tensão na saída V_a são apresentadas na Tabela 2 as chaves que deverão estar fechadas para o nível de tensão desejado. A Figura 9 apresenta a forma de onda da tensão de saída do conversor a cinco níveis.

Tabela 2- Estados de comutação do Conversor NPC 5 níveis

TENSÃO FASE-NEUTRO SAÍDA	$E/2$ (V)	$E/4$ (V)	0 (V)	$-E/4$ (V)	$-E/2$ (V)
CHAVES FECHADAS	S1	S2	S3	S4	S5
	S2	S3	S4	S5	S6
	S3	S4	S5	S6	S7
	S4	S5	S6	S7	S8

Figura 9 - Formas de onda da tensão de saída do conversor NPC 5 níveis



No mesmo caso do conversor à três níveis, os valores médios em instantes de tempo da onda apresentada formam uma senóide. O detalhe desta onda está no fato de que quando a onda está no nível mais elevado de tensão, por exemplo como mostra a figura entre 2 e 8 ms, o conversor não grampeia entre o sinal de referência (0V) e o último nível (200V), mas sim entre o nível intermediário (100V) e o de pico (200V).

A Figura 10 e a Figura 11 apresentam um conversor trifásico de cinco níveis e o gráfico da tensão de saída em cada fase, respectivamente.

Figura 10 - Conversor NPC 5 níveis trifásico

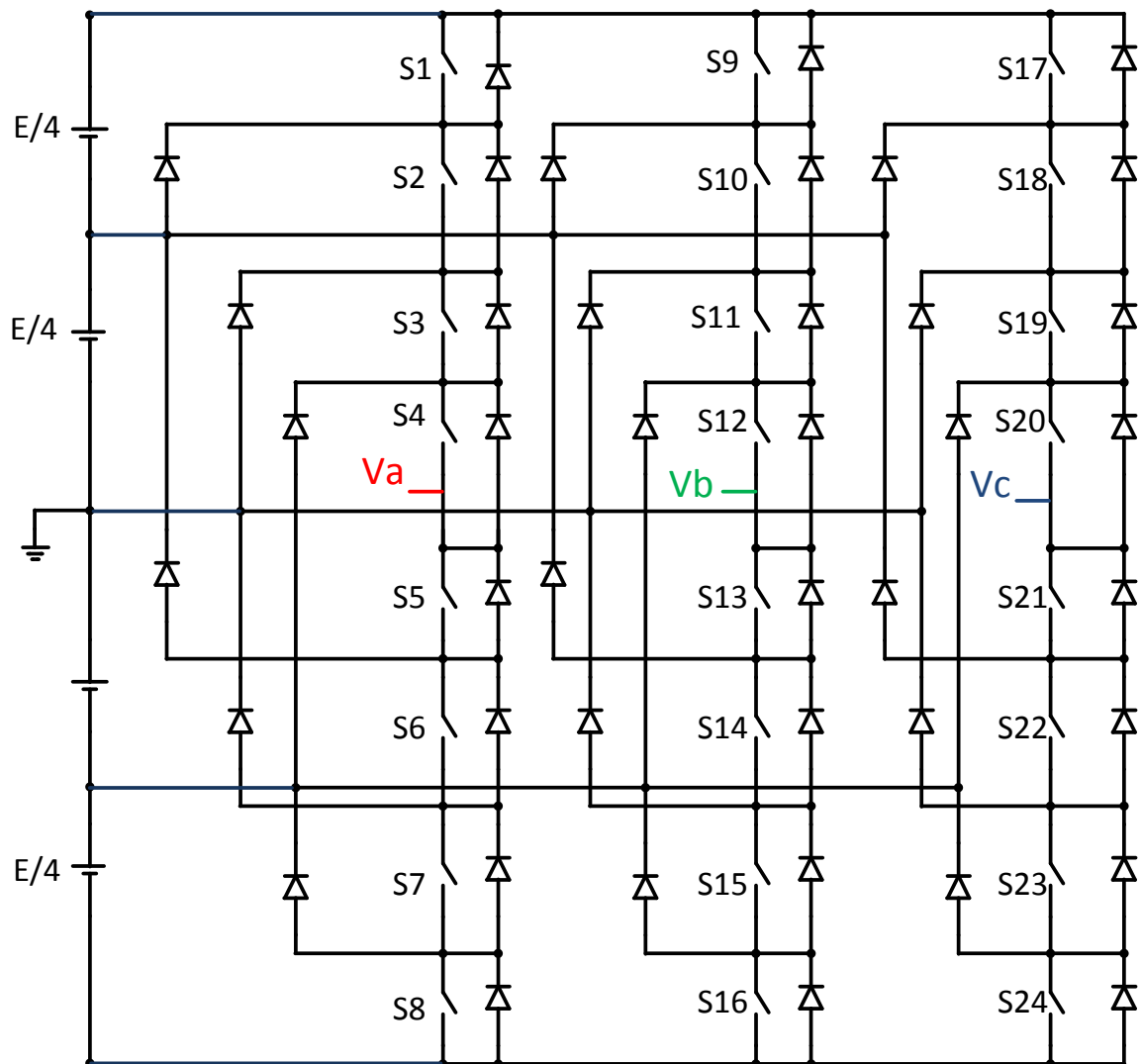
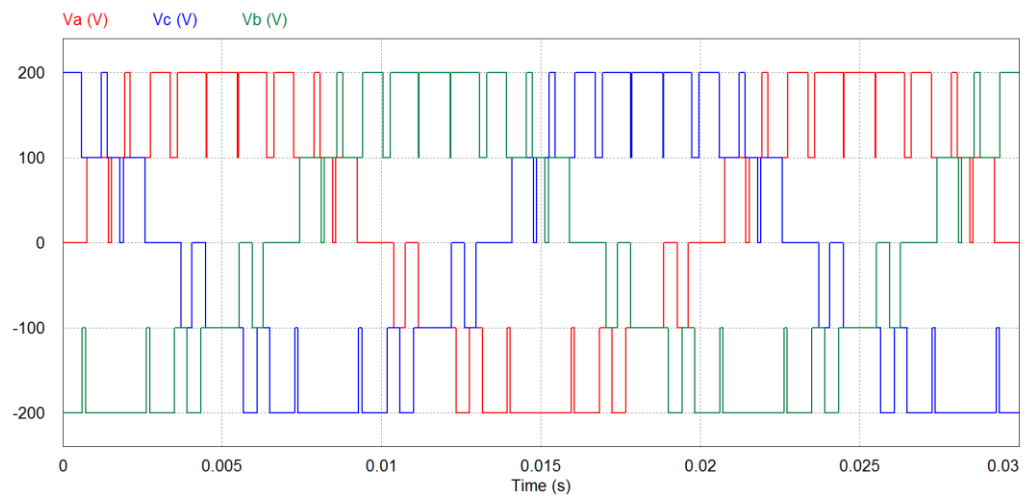


Figura 11 - Formas de onda das tensões de saída do conversor NPC trifásico 5 níveis



2.2 CONVERSOR COM GRAMPEAMENTO A CAPACITOR FLUTUANTE – *FLYING CAPACITOR*

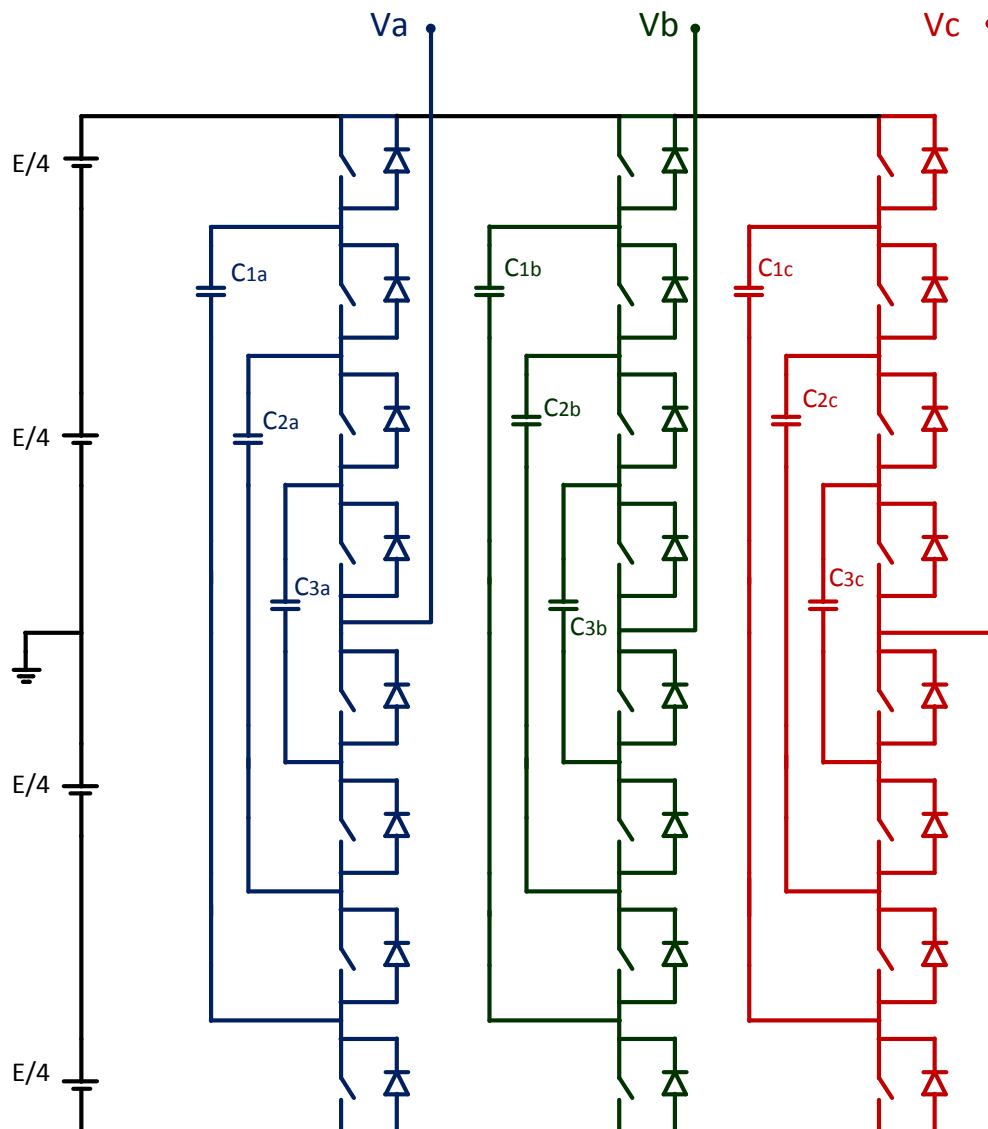
Essa topologia surgiu após a estrutura com diodo de grampeamento. Ela apresenta algumas vantagens. A principal delas é a redundância nos estados de comutação das chaves. Além disso os capacitores são responsáveis pela redução nos esforços nas chaves [1][2].

A estrutura é formada por um conjunto de capacitores por barramento que são quantificados de acordo com a equação (4).

$$n^{\circ}cap = \frac{(m - 2) \times (m - 1)}{2} \quad (4)$$

Onde m – número de níveis

Essa relação evidencia que com o aumento do número de níveis do conversor há um aumento no número de capacitores recaindo sobre o custo do projeto. O alto número de capacitores e o complexo sistema de controle de tensão do mesmo são vistos como as principais desvantagens deste conversor. A figura 12 apresenta a topologia do conversor *Flying Capacitor*.

Figura 12 - Conversor *Flying Capacitor* de cinco níveis

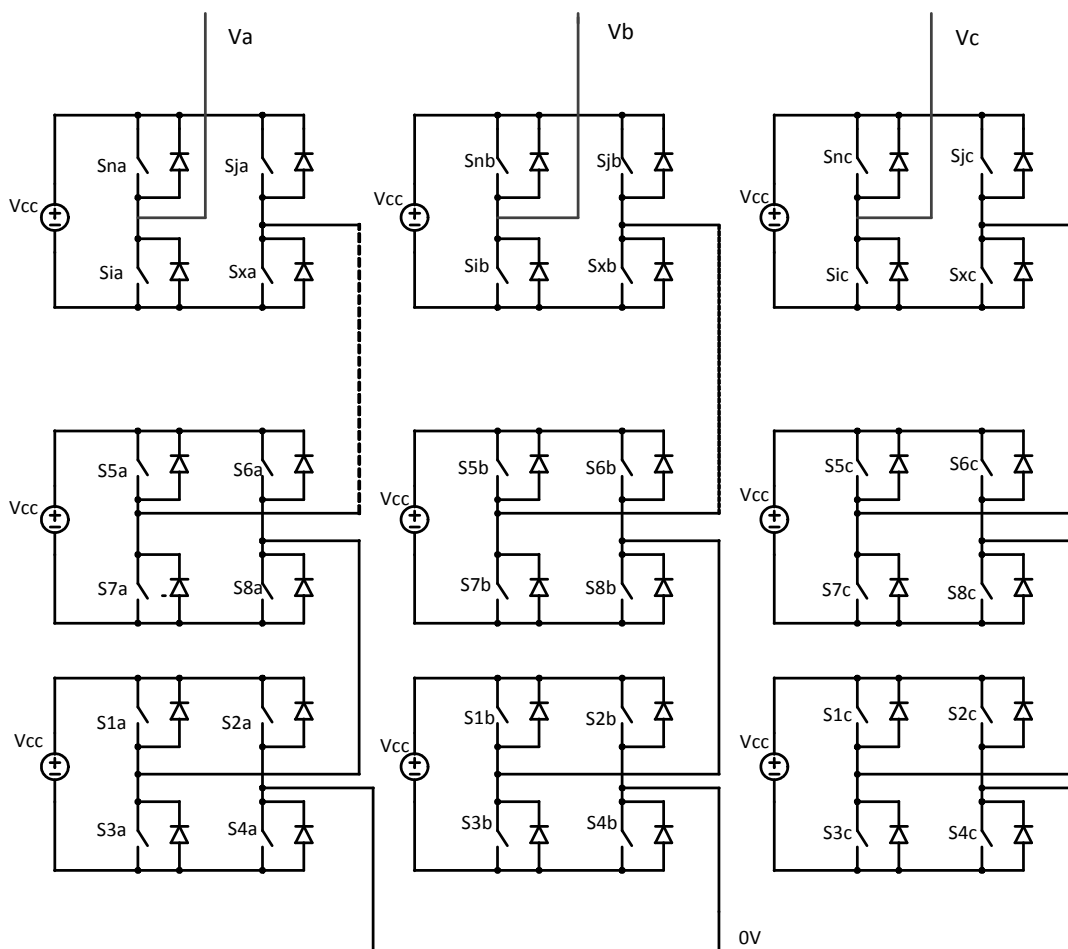
Em [10] é apresentada uma operação que possibilita o aumento do número de níveis na saída sem que haja aumento no número de capacitores. Por exemplo, um conversor estruturado para quatro níveis pode ser comutado de modo a produzir oito níveis na saída. Essa modificação é feita alterando a relação de tensão dos capacitores. Com isso o conversor passa a não ter mais a vantagem de redundância dos estados de comutação e perde o balanceamento da tensão em cada capacitor.

2.3 CONVERSORES EM CASCATA

São conversores formados pela associação de conversores meia ponte ou ponte completa que são conectados em série. O primeiro conversor em série a ser estudado é conversor formado por células *H-Bridge* (ponte completa) por volta da década de 70. A sua principal vantagem está na independência da utilização de diodos e capacitores de grampeamento. Devido a isso não possui um desbalanceamento da tensão dos capacitores observado no conversor com grampeamento a diodo e no *Flying Capacitor*.

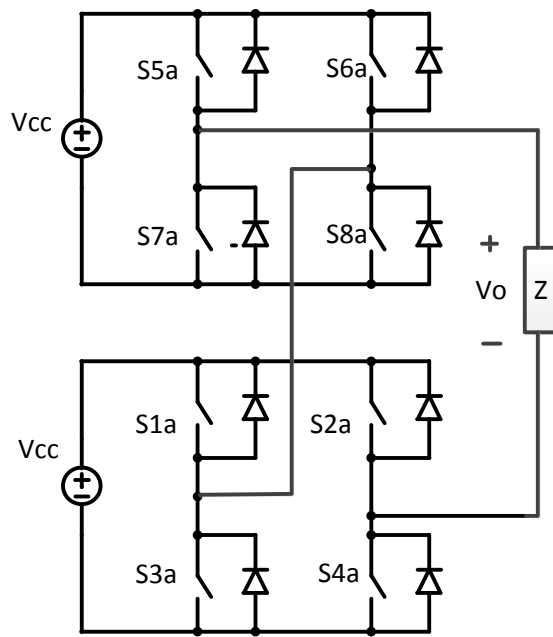
É largamente utilizado em aplicações de alta potência e compensação de reativos. Outra vantagem está na redundância dos estados de comutação, como citado em tópicos anteriores, que possibilita uma liberdade na opção da escolha de que chaves serão comandadas. A Figura13 mostra um conversor em cascata trifásico de m-níveis.

Figura 13 - Conversor em cascata H-Bridge



Na Figura 14 pode ser observado um conversor em cascata à cinco níveis apresentado em [4]. Ao lado a Tabela 3 com diferentes estados de comutação facilitando a compreensão de que existem estados redundantes para uma mesma tensão de saída. O estado 1 representa o acionamento da chave.

Figura 14 - Conversor monofásico 5 níveis tipo cascata



Fonte[4]

Tabela 3- Possíveis Estados de comutação das chaves

S1a	S2a	S3a	S4a	S5a	S6a	S7a	S8a	Vo
1	0	0	1	1	0	0	1	+2Vcc
1	1	0	0	1	0	0	1	+Vcc
0	0	1	1	1	0	0	1	+Vcc
1	0	0	1	1	1	0	0	+Vcc
1	0	0	1	0	0	1	1	+Vcc
0	1	1	0	1	0	0	1	0
1	1	0	0	1	1	0	0	0
0	0	1	1	1	1	0	0	0
1	1	0	0	0	0	1	1	0
0	0	1	1	0	0	1	1	0
1	0	0	1	0	1	1	0	0
1	1	0	0	0	1	1	0	-Vcc
0	0	1	1	0	1	1	0	-Vcc
0	1	1	0	1	1	0	0	-Vcc
0	1	1	0	0	0	1	1	-Vcc
0	1	1	0	0	1	1	0	-2Vcc

Fonte[4]

Fica claro perceber que com duas fontes é possível obter um conversor de cinco níveis na saída. Logo pode-se afirmar que o conversor em cascata com células em ponte completa obedece a equação 5.

$$m = 2n + 1 \quad (5)$$

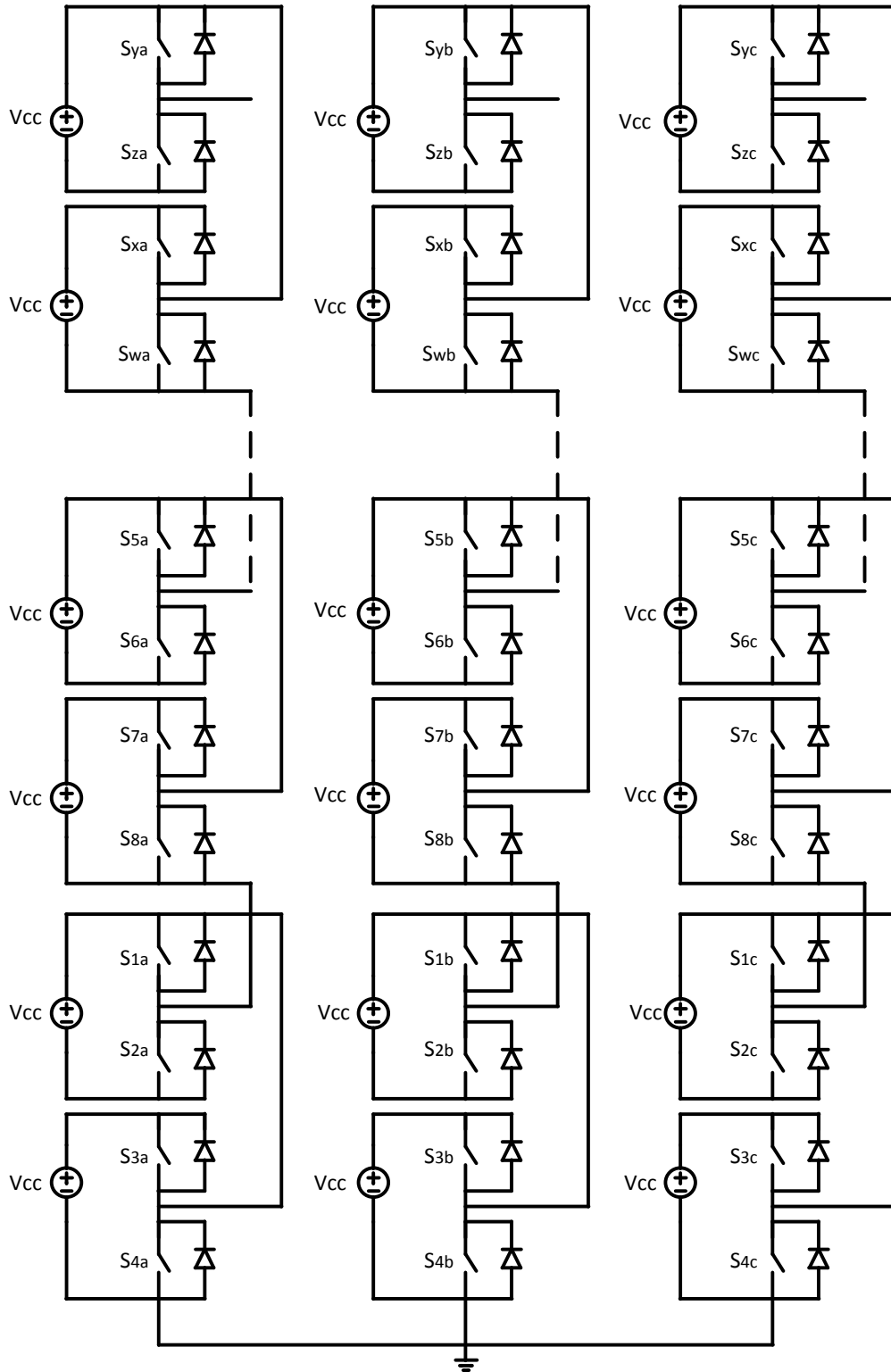
Onde: m - número de níveis

n - número de fontes

Outro conversor tratado em [1][2][11] utiliza células meia ponte em cascata. Ele é apresentado na Figura 15. É exigido o dobro do número de fontes de tensão para que ele atinja o mesmo número de níveis do conversor em cascata com células em ponte completa. Logo esse atende a equação (6).

$$m = 2 \times (2n + 1) \quad (6)$$

Figura 15 - Conversor de m níveis de conjuntos meia ponte em cascata



3 ESTRATÉGIAS DE MODULAÇÃO

As estratégias que utilizam largura de pulso são as mais utilizadas em conversores multiníveis. São caracterizadas pela comparação de uma onda senoidal de referência e portadoras triangulares dispostas de diversas maneiras, como serão apresentadas abaixo.

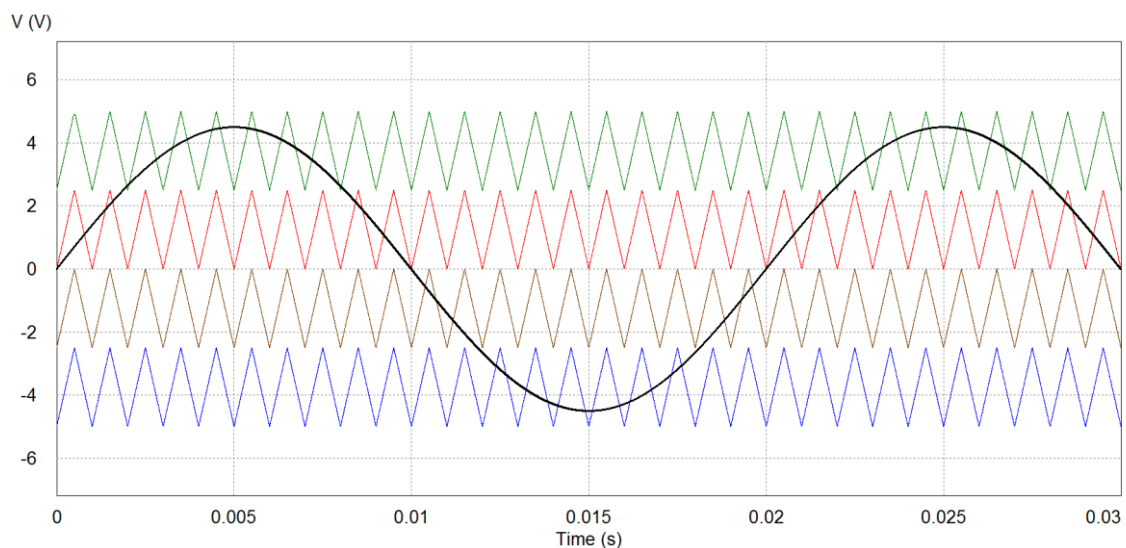
3.1 DISPOSIÇÃO EM FASE (IPD-*IN PHASE DISPOSITION*)

Nesta estratégia de modulação as ondas portadoras são dispostas com a mesma fase. Cada uma possui também a mesma amplitude e a mesma frequência. Existe apenas uma diferença de nível CC entre elas.

O número de portadoras é definido como sendo o número de níveis desejados menos uma unidade, ou seja, se a saída de determinado conversor apresenta cinco níveis, a modulação possuirá quatro portadoras. Geralmente estas portadoras são ondas triangulares de alta frequência que são comparadas com um sinal senoidal.

A Figura 16 ilustra a estratégia IPD com uma moduladora de frequência fundamental de 50 Hz (frequência da senóide) e frequência da portadora com 1 kHz. Logo o índice de modulação em frequência, comentado anteriormente, equivale a $m_f=20$.

Figura 16 - Modulação IPD para 5 níveis

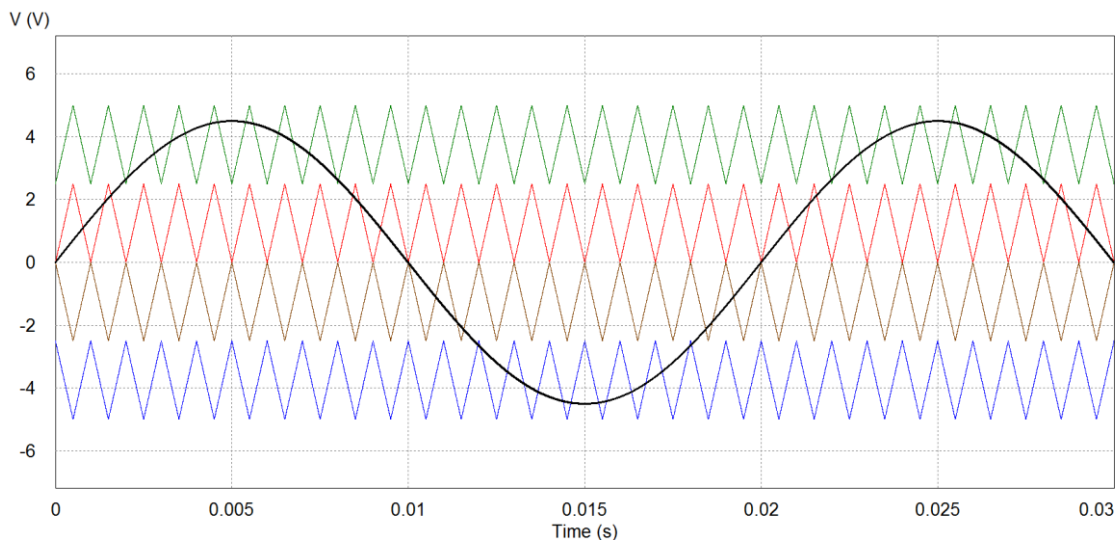


A amplitude de tensão da senóide apresentada é de 4,5 V e a tensão pico a pico da portadora é de 2,5. Utilizando a equação 3 conclui-se que o índice de modulação em amplitude é $m_a=0,9$

3.2 DISPOSIÇÃO EM OPOSIÇÃO DE FASE (POD - *PHASE OPPOSITION DISPOSITION*)

No mesmo caso que o anterior, a modulação POD possui suas portadoras com a mesma frequência e mesma amplitude, porém agora elas possuem uma defasagem entre si. As portadoras negativas ficam defasadas em 180° das portadoras positivas [3][12]. O eixo das abscissas funciona como se fosse um espelho para as portadoras. Utilizando os mesmos índices de modulação em amplitude e frequência, são mostradas na Figura 17 as formas de onda de tensão de referência senoidal e das portadoras.

Figura 17 - Modulação POD para 5 níveis

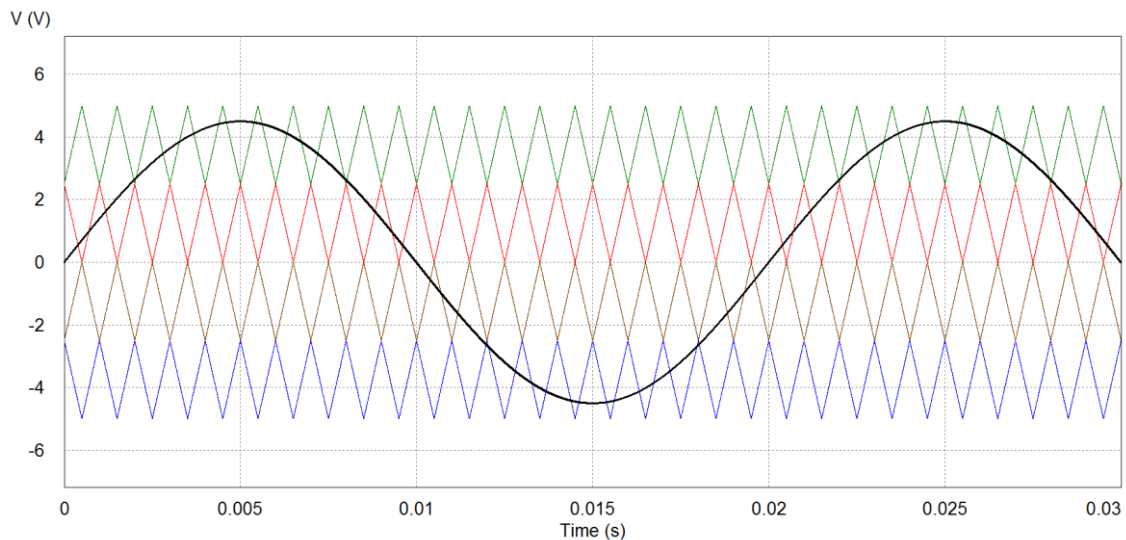


3.3 DISPOSIÇÃO EM OPOSIÇÃO DE FASES ALTERNADAS (APOD- *ALTERNATIVE PHASE OPPOSITION DISPOSITION*)

Já neste tipo de estratégia de modulação as portadoras são defasadas em 180° uma a uma. Ainda possuem a mesma amplitude, a mesma frequência e uma

diferença DC entre elas. No caso da Figura 18, que apresenta a estratégia do tipo APOD, o nível DC entre as portadoras corresponde a $\pm 2,5$ V pelo fato de existirem quatro portadoras dispostas na faixa de -5V e +5V.

Figura 18 - Modulação APOD para 5 níveis



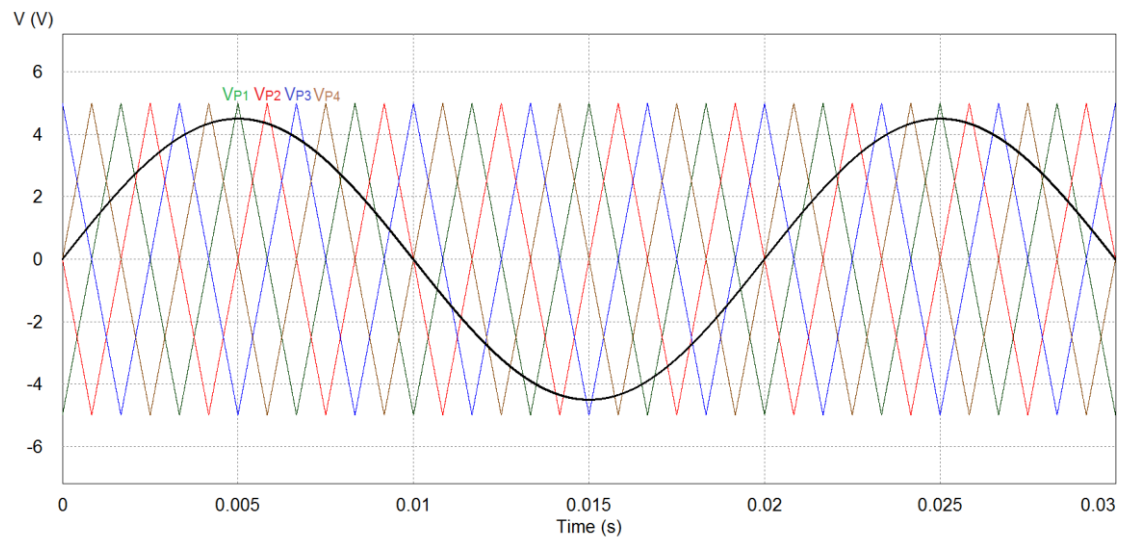
3.4 DISPOSIÇÃO DE FASES DESLOCADAS (PSD-PHASE SHIFTED DISPOSITION)

Diferentemente das três outras estratégias anteriormente citadas, a modulação *Phase Shifted Disposition* apresenta portadoras sem uma diferença de nível CC entre elas. As portadoras possuem a mesma frequência e amplitude e são deslocadas em fase entre si de acordo com o número de níveis. O cálculo do deslocamento de fase é dado pela equação (7).

$$\varphi = \frac{2\pi}{(m-1)} [rad] \quad (7)$$

A Figura 19 apresenta as formas de onda da estratégia discutida onde a frequência da portadora é de 300 Hz e é empregada em um conversor com cinco níveis. Logo a modulação possui quatro portadoras deslocadas em 90°.

Figura 19 - Modulação PSD para 5 níveis



4 FPGA - FIELD PROGRAMMABLE GATE ARRAY

Nas últimas décadas o avanço da eletrônica digital possibilitou a substituição dos controladores analógicos pelos controladores digitais. A flexibilidade de reconfiguração desses novos dispositivos facilita o trabalho dos projetistas. Para realizar o chaveamento dos conversores multiníveis podem ser utilizados no projeto o DSP (*Digital Signal Processor*), o FPGA (*Field Programmable Gate Array*) ou microcontroladores. Cabe ao projetista definir qual deles atenderá melhor as suas necessidades.

O primeiro processador de sinais digitais (DSP) foi lançado pela empresa Bell Labs em 1979. Porém só ficou conhecido em 1983 quando a Texas Instruments lançou o modelo TMS32010. Eles são utilizados para o processamento de operações como adição, multiplicação e transferência de dados. Possuem uma arquitetura otimizada para computação intensiva. São versáteis e detêm seu próprio código de instruções. Cada empresa cria seu próprio ambiente de desenvolvimento para cada tipo de dispositivo.[12][13]

O FPGA é formado por uma matriz de blocos lógicos contidos em um único circuito integrado. O primeiro dispositivo desenvolvido foi em 1983 pela empresa Xilinx Inc. Atualmente quem domina o mercado é a empresa Altera.

Se o sistema trabalha com uma banda de baixa frequência (unidades de kHz) e com implementação de um canal simples torna-se vantajoso o uso do DSP. Já para aplicações em que a banda de frequência é alta ou requer mais de um canal simples a melhor escolha passa a ser o FPGA.

O funcionamento do FPGA é baseado em ciclos de clock, onde cada operação matemática é realizada a cada ciclo [14]. Também possuem um processamento em paralelo evitando situações de loop infinito. O DSP efetua um processamento sequencial e alguns tipos necessitam de várias interações para realizar uma operação matemática

4.1 METODOLOGIA DE IMPLEMENTAÇÃO

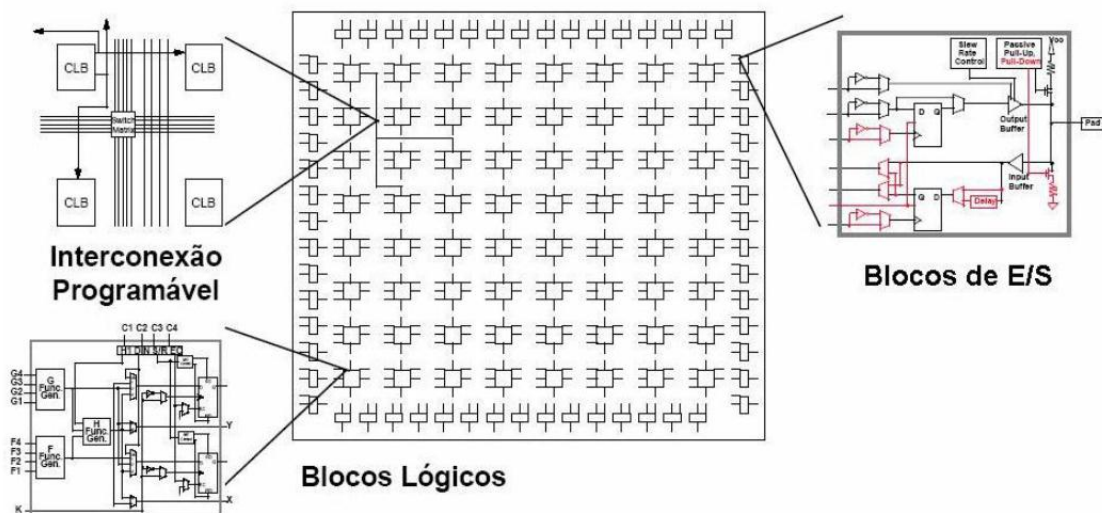
Como o objetivo do trabalho é a implementação da modulação PWM em conversores com múltiplos níveis é preferida a utilização de um dispositivo FPGA

para ser responsável pela modulação do comando das chaves. Esse dispositivo atende melhor as necessidades por ser mais rápido e por trabalhar com uma banda mais alta de frequência, além de permitir maior flexibilidade na geração de sinais com intertravamento e modulação mais complexa.

4.1.1 Aspectos gerais do FPGA

A estrutura interna do FPGA é basicamente formada por blocos lógicos, blocos de entrada e saída e chaves de interconexão. A Figura 20 ilustra com detalhes essa estrutura.

Figura 20 - Estrutura Interna de um FPGA



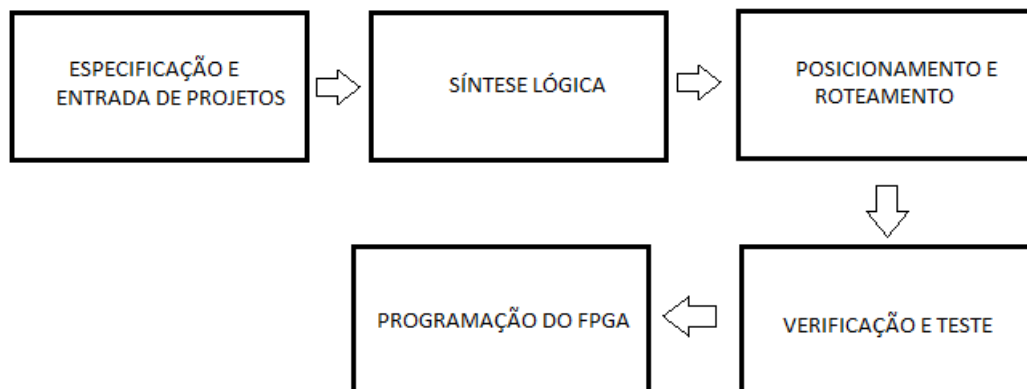
Fonte [10]

As funções do programa são implementadas nos blocos lógicos. Esses possuem células que são capazes de armazenar valores lógicos zero (0) ou um (1) e possuem recursos sequenciais como, por exemplo, registradores. Existe um processo chamado de roteamento onde ocorre a interconexão desses blocos. O projeto com FPGA é caracterizado por ser modular, ou seja, cada bloco do programa é responsável por uma função [15].

4.1.2 Etapas de um projeto utilizando FPGA

O processo de desenvolvimento de um projeto utilizando FPGA envolve uma série de etapas. A sequência dessas etapas são representadas na Figura 21.

Figura 21 - Sequência de etapas do processo de um projeto com FPGA



Fonte [12]

4.1.3 Especificação e entrada de projetos

Essa é a primeira etapa no desenvolvimento do projeto do FPGA onde o projetista cria o código com as instruções na ferramenta de programação. Esta pode ser realizada de duas formas: a primeira delas a partir de um editor gráfico onde é possível inserir portas lógicas formando um circuito capaz de realizar a aplicação desejada e a segunda forma através de um editor de texto é utilizada a linguagem de descrição de *hardware* HDL (*Hardware Description Language*).

A forma de descrever o circuito pelo editor gráfico é mais utilizada em situações em que o modelo do projeto é mais simples. A tela de edição permite uma visualização 2D do circuito. Todo *software* possui bibliotecas com diferentes famílias de multiplexadores, portas lógicas, buffers, etc.

Já em modelos mais complexos é utilizada a linguagem de descrição de hardware. As principais linguagens existentes desse tipo são: ABEL (*Advanced Boolean Equation Language*), VHDL (*Very High Speed Integrated Circuit Hardware Description Language*) e Verilog. Todas elas são capazes de descrever o

comportamento de um conjunto digital formado por portas lógicas, flip-flops e decodificadores.

A linguagem VHDL surgiu na década de 80 e foi padronizada pela IEEE em 1986 após uma série de revisões. Ela foi criada com o objetivo de padronizar o intercâmbio de informações referentes ao comportamento dos circuitos digitais entre os grupos de pesquisa. Uma das principais vantagens é a independência da tecnologia. Mesmo com a evolução dos dispositivos de diferentes fornecedores a linguagem não sofre alterações. A desvantagem está no fato dos projetistas necessitarem um bom conhecimento na área de circuitos digitais, diferentemente do DSP em que basicamente é necessário o conhecimento da linguagem C.

4.1.4 Síntese lógica

A síntese lógica é um processo automatizado onde cabe ao *software* simplificar o código desenvolvido anteriormente. Essa otimização é extremamente necessária para reduzir a área ocupada no circuito integrado. Assim há uma diminuição do atraso dos sinais envolvidos.

4.1.5 Posicionamento e roteamento

O posicionamento é caracterizado pela atribuição dos componentes lógicos do projeto com os componentes do circuito integrado. Já o roteamento é a atribuição das trilhas de comunicação entre cada componente. Depois disso ainda há uma checagem se 100% das interconexões foram realizadas.

4.1.6 Verificação e teste

Nessa etapa são realizados os testes de funcionalidade do circuito através de uma simulação onde é exibida uma tela com os estados lógicos de todas as portas de entradas e saídas (determinadas pelo projetista) ao longo do tempo. É possível gerar sinais de estímulo (mudanças de estado) nas entradas para simular, por exemplo, a situação em que o FPGA recebe um comando externo.

4.1.7 Programação do FPGA

Considerada a última etapa do processo, a programação consiste na geração de um arquivo de configuração, definido pelo código que foi desenvolvido anteriormente, e sua transferência para o hardware. Essa transferência pode ser realizada por uma porta USB ou por uma porta de comunicação paralela.

5 DESENVOLVIMENTO DO PROJETO

Na primeira etapa de desenvolvimento do projeto foi realizado um estudo das características do FPGA utilizado. Logo depois foi necessário o aprendizado dos passos para a criação de um projeto com o *software* de programação do dispositivo. Inicialmente alguns circuitos básicos foram criados juntamente com a simulação por *software* objetivando a prática de utilização.

Ao evoluir no aprendizado da programação dos circuitos básicos partiu-se para a construção de algoritmos de programação das modulações dos conversores escolhidos. Com o algoritmo em mãos foi realizada a criação em VHDL dos códigos. Assim ficaram restando apenas a análise das formas de onda na saída do FPGA e sua implementação. Esses passos são discutidos nesse capítulo.

5.1 KIT DE DESENVOLVIMENTO DE0-NANO ALTERA

O kit de desenvolvimento usado no trabalho é o DE0-Nano da Altera devido a disponibilidade no laboratório e a capacidade de atender as necessidades do projeto. Se comparado aos outros kits do mesmo fornecedor esse é o que possui menos funções e o menor custo.

O kit DE0-Nano possui um FPGA modelo EP4CE22F17C6N da família Cyclone IV e apresenta as seguintes características:

- 153 portas I/O
- Conversor A/D de 8 canais e 12 bits
- 8 leds para testes ou simulações
- 4 micro chaves
- 2 botões
- Clock de 50 MHz
- 2 barramentos de 40 pinos cada para conexão às I/O do circuito integrado
- Conexão USB

- Alimentação em 5V DC (pela porta USB, barramento de expansão ou por dois pinos externos)
- 32 MB SDRAM
- 22320 elementos lógicos
- 1 acelerômetro

A tensão de saída dos pinos é de 3,3V DC em relação à referência. A Figura 22 apresenta uma foto ilustrativa do dispositivo em questão.

Figura 22 - Kit DE0-nano Altera



Fonte [16]

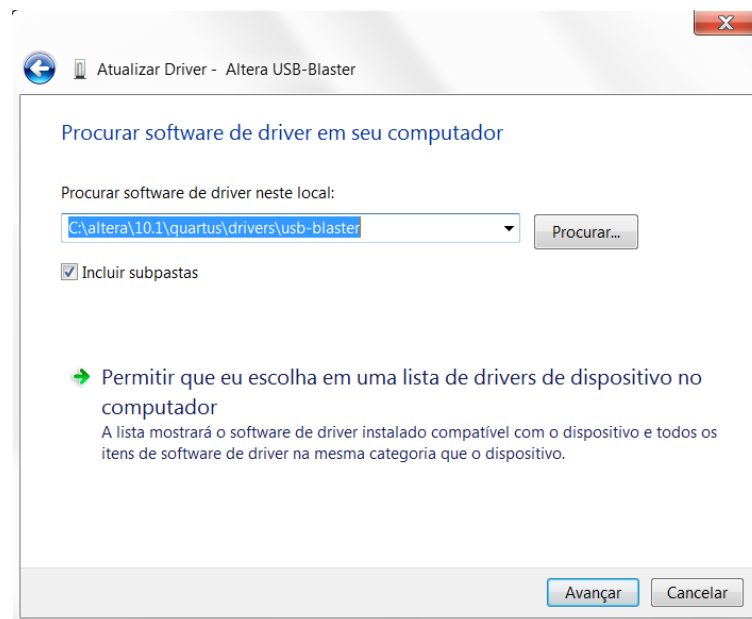
Mais detalhes sobre o endereço dos pinos dos LED's, barramentos, entrada de clock e chaves são apresentados no manual do kit em [16].

5.2 SEQUÊNCIA DE PASSOS DE PROGRAMAÇÃO, GRAVAÇÃO E CONFIGURAÇÃO DO KIT DE0-NANO ALTERA.

Inicialmente é necessária a instalação do software *Quartus II 10.1 Web Edition*. Esse é disponibilizado no site do fabricante Altera ou no DVD-R que vem junto com o kit.

Antes de iniciar a programação também é necessária a instalação do driver do dispositivo. Para instalar é preciso conectar o FPGA na porta USB do computador.

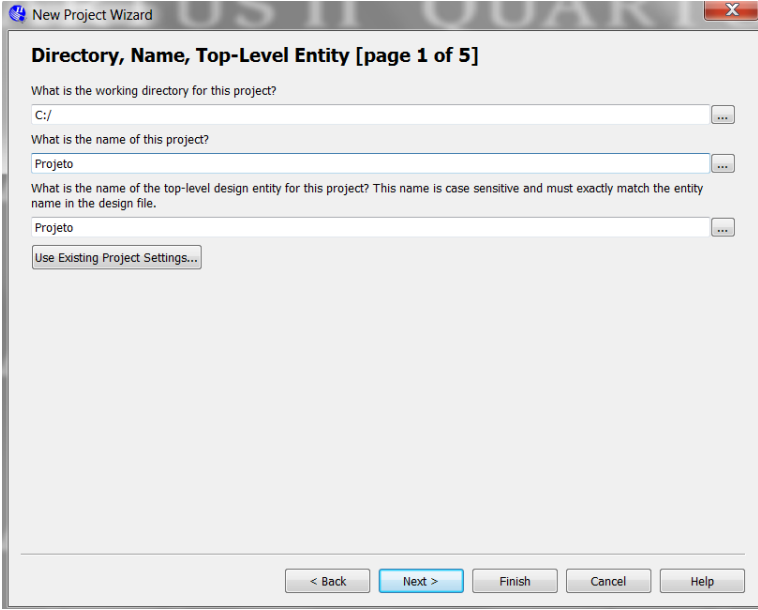
Figura 23 - Instalação do software de driver



O computador irá reconhecer o dispositivo e solicitará o *software* de driver, como mostra a Figura 23. O caminho do mesmo se encontra na pasta onde o *Quartus II 10.1 Web Edition* foi instalado que por padrão possui o seguinte endereço: C:\altera\10.1\quartus\drivers\usb-blaster.

Após a instalação do driver pode ser iniciada a criação de um novo projeto. Para isso deve-se iniciar o programa Quartus. Na tela inicial é sugerida a opção de criar um novo projeto (*Create a New Project*). Selecionada essa opção é apresentada uma tela de introdução. Em seguida é solicitado o diretório onde o projeto será salvo e o nome que o mesmo possuirá como mostra a Figura 24.

Figura 24 - Criação de um novo projeto



Directory, Name, Top-Level Entity [page 1 of 5]

What is the working directory for this project?
C:/

What is the name of this project?
Projeto

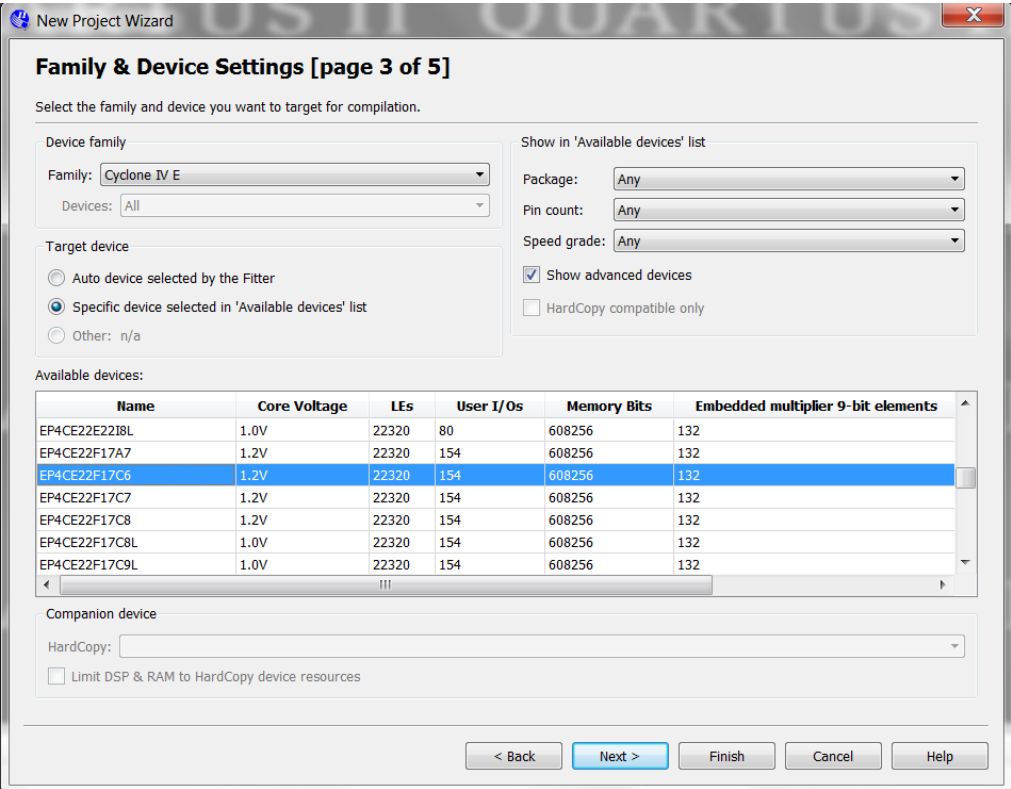
What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.
Projeto

Use Existing Project Settings...

< Back Next > Finish Cancel Help

O próximo passo será a escolha da família do dispositivo a ser trabalhado. O kit DE0-Nano corresponde a família Cyclone IV E e possui o código comercial EP4CE22F17C6. A Figura 25 mostra a tela de configuração desta etapa.

Figura 25 - Seleção do modelo do kit a ser trabalhado



Family & Device Settings [page 3 of 5]

Select the family and device you want to target for compilation.

Device family
Family: Cyclone IV E
Devices: All

Target device
☐ Auto device selected by the Fitter
☒ Specific device selected in 'Available devices' list
☐ Other: n/a

Show in 'Available devices' list
 Package: Any
 Pin count: Any
 Speed grade: Any
☒ Show advanced devices
☐ HardCopy compatible only

Available devices:

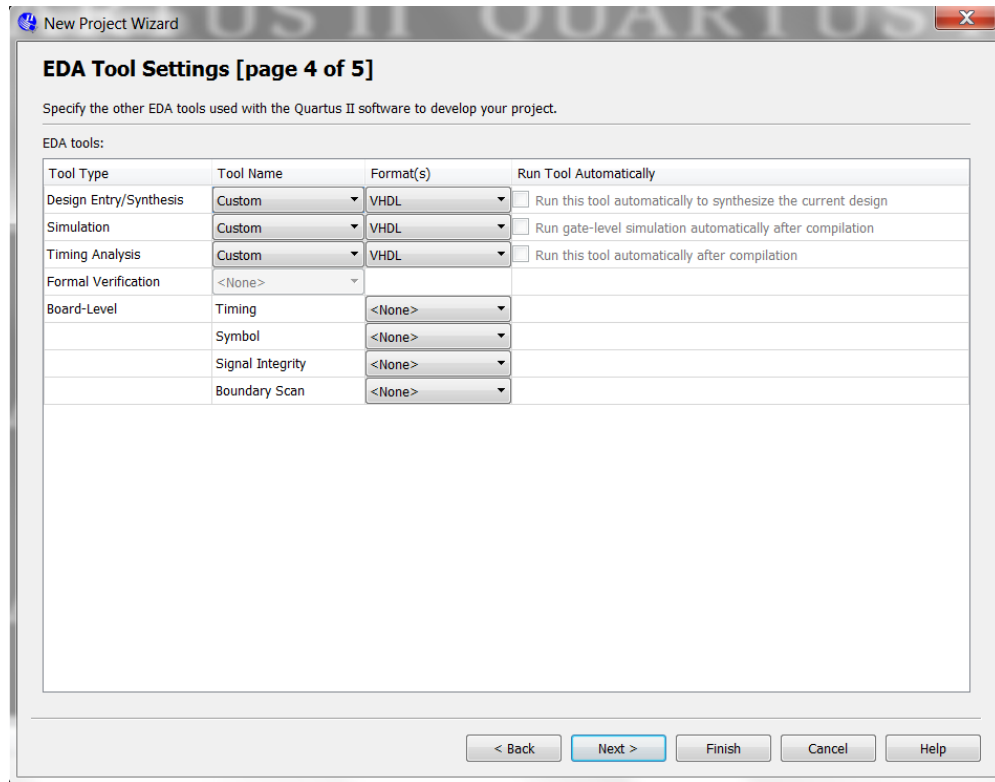
Name	Core Voltage	LEs	User I/Os	Memory Bits	Embedded multiplier 9-bit elements
EP4CE22E22I8L	1.0V	22320	80	608256	132
EP4CE22F17A7	1.2V	22320	154	608256	132
EP4CE22F17C6	1.2V	22320	154	608256	132
EP4CE22F17C7	1.2V	22320	154	608256	132
EP4CE22F17C8	1.2V	22320	154	608256	132
EP4CE22F17C8L	1.0V	22320	154	608256	132
EP4CE22F17C9L	1.0V	22320	154	608256	132

Companion device
 HardCopy:
☐ Limit DSP & RAM to HardCopy device resources

< Back Next > Finish Cancel Help

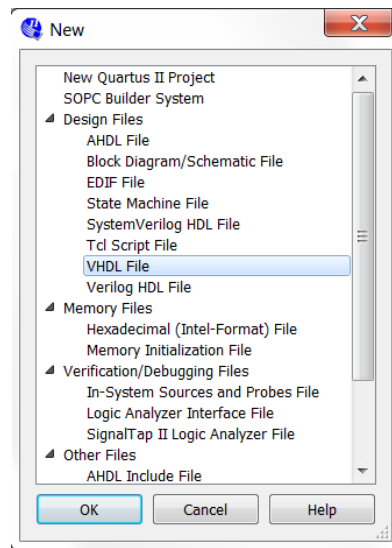
O próximo passo consiste na última etapa de configuração. As configurações recomendadas são ilustradas na Figura 26.

Figura 26 - Configuração das ferramentas de simulação e análise temporal



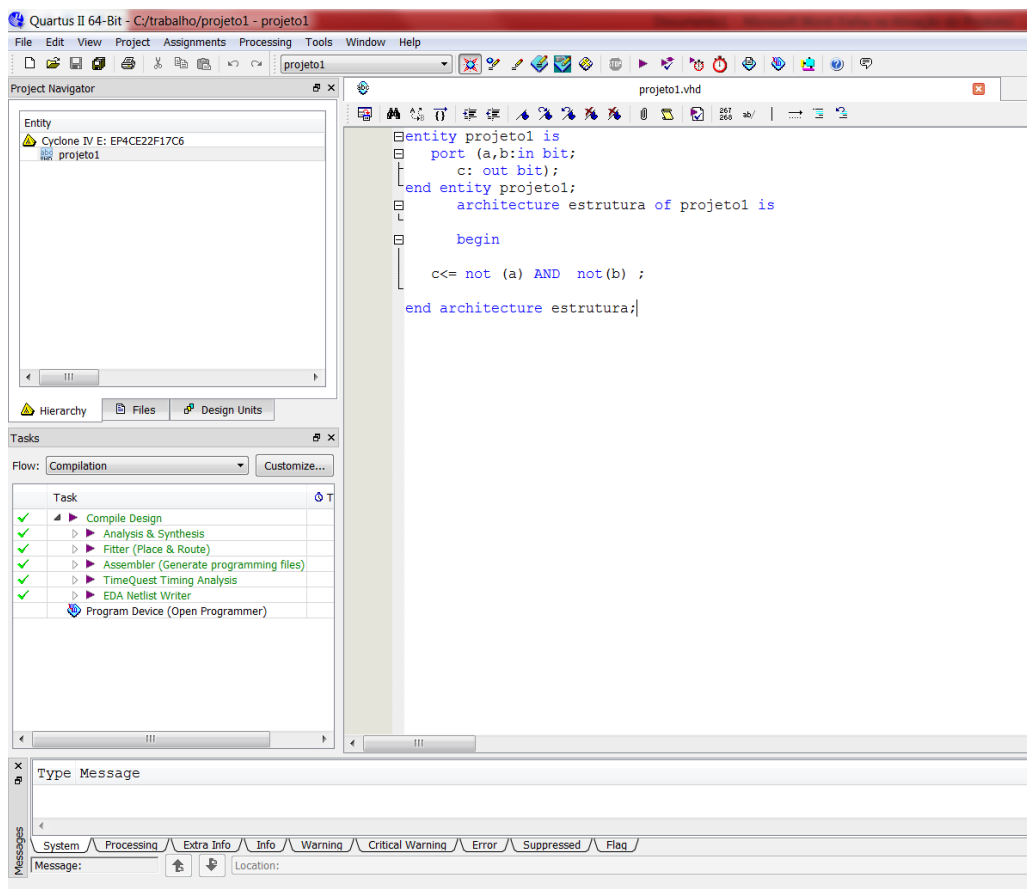
Prosseguir a configuração e finalizar o assistente. O próximo passo consiste na escolha de qual tipo de linguagem de programação será empregada, por exemplo, pode ser utilizada a linguagem VHDL, diagrama de blocos ou máquina de estados. Nesse exemplo é apresentado o projeto em VHDL. Para a criação do ambiente de programação é necessário ir na guia 'File' e em seguida em 'New'. Selecionar a opção *VHDL File* como ilustra a Figura 27.

Figura 27 - Seleção de um novo projeto em VHDL



Será aberta uma janela onde deve ser descrito o código do projeto. Na Figura 28 é apresentado um projeto simples de uma porta lógica AND.

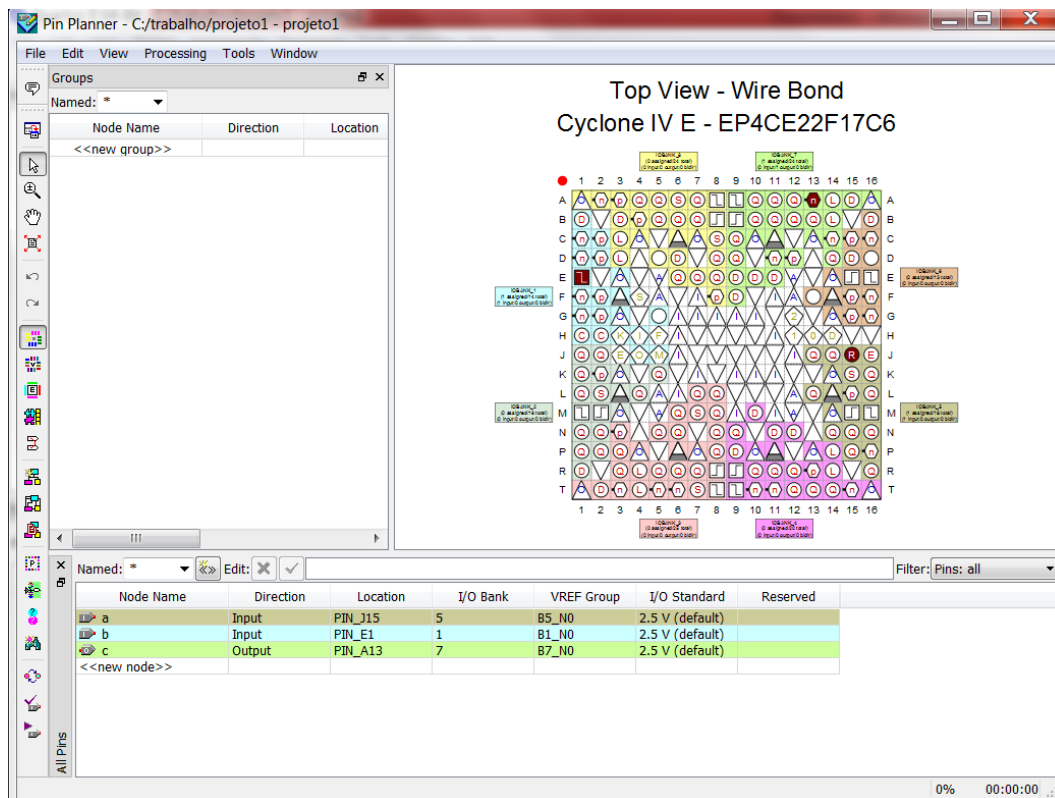
Figura 28 - Projeto de uma porta lógica AND



Posteriormente é necessária a compilação do projeto. No ícone 'Start Compilation' essa tarefa pode ser executada. Uma nova janela abrirá relatando os erros encontrados na elaboração do código ou configuração do projeto. Se nenhum erro for acusado pode-se prosseguir ao próximo passo que consiste em configurar os pinos de entrada e saída do kit em estudo. Assim é necessário ir no item 'Assignments' e logo depois em 'Pin Planer'.

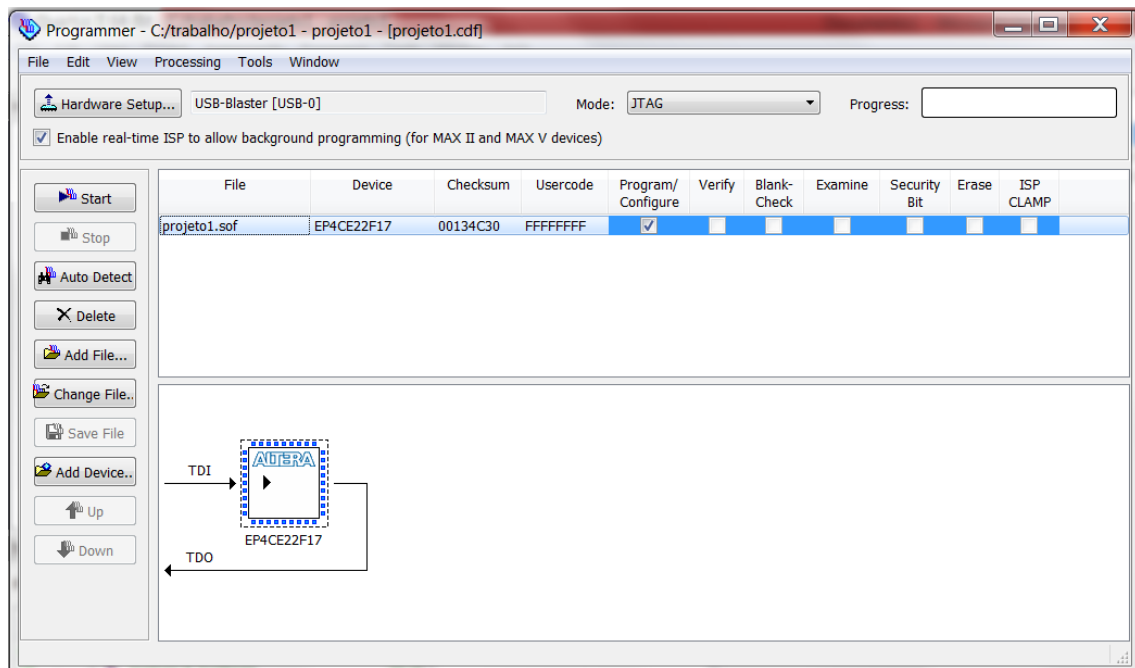
No programa apresentado da figura anterior foi mencionada duas portas de entrada 'a', 'b' e uma saída 'c'. Para efeitos didáticos serão atribuídas a essas entradas os *push buttons* do FPGA e para a saída 'c' atribuído um LED do dispositivo. Para realizar essa atribuição basta arrastar as variáveis I/O da lista no lado inferior para o mapa do FPGA (ver Figura 29). Esse mapa consiste de linhas e colunas formando uma matriz. Cada elemento dessa matriz corresponde a uma função do FPGA. Nessa parte faz necessário o uso do manual do dispositivo para análise da estrutura do FPGA. Por exemplo, os dois *push buttons* correspondem aos pinos J15 e E1; e um dos LED's corresponde ao pino A13.

Figura 29 - Mapa com todas as entradas e saídas da placa do FPGA



Uma vez atribuído os parâmetros do programa com os pinos do FPGA é necessário uma nova compilação do projeto. Se nenhum erro for apresentado já pode ser realizado a gravação no FPGA no item '*Tools*' e em seguida '*Programmer*'. Uma janela abrirá como mostra a Figura 30.

Figura 30 - Programação do código e comunicação com o hardware



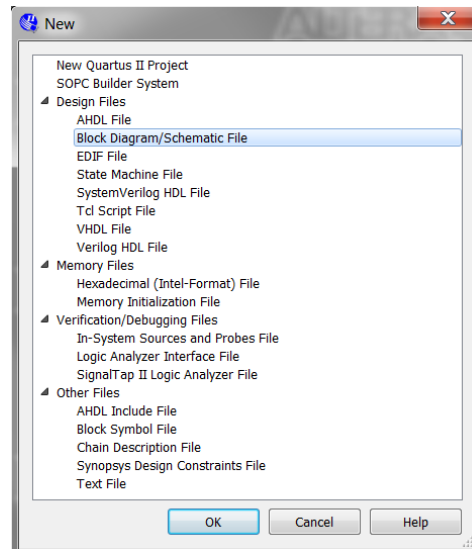
O dispositivo deve estar conectado na porta USB do computador. Em '*Hardware Setup*' deve ser selecionado '*USB-Blaster [USB-0]*'. A gravação já pode ser iniciada no item '*Start*' e a mesma estará completa quando a indicação do progresso estiver em 100%.

5.3 PASSOS INICIAIS PARA SIMULAÇÃO DE UM PROJETO UTILIZANDO PORTAS LÓGICAS

Neste exemplo é apresentado o projeto de circuitos lógicos. Para isso deve ser realizado os mesmos passos para a criação de um novo projeto descrito no tópico anterior. Após finalizar a configuração e o assistente escolher qual tipo de linguagem de programação será empregada que no caso é a de diagrama de blocos. Para a criação do ambiente de programação é necessário ir na guia '*File*' e

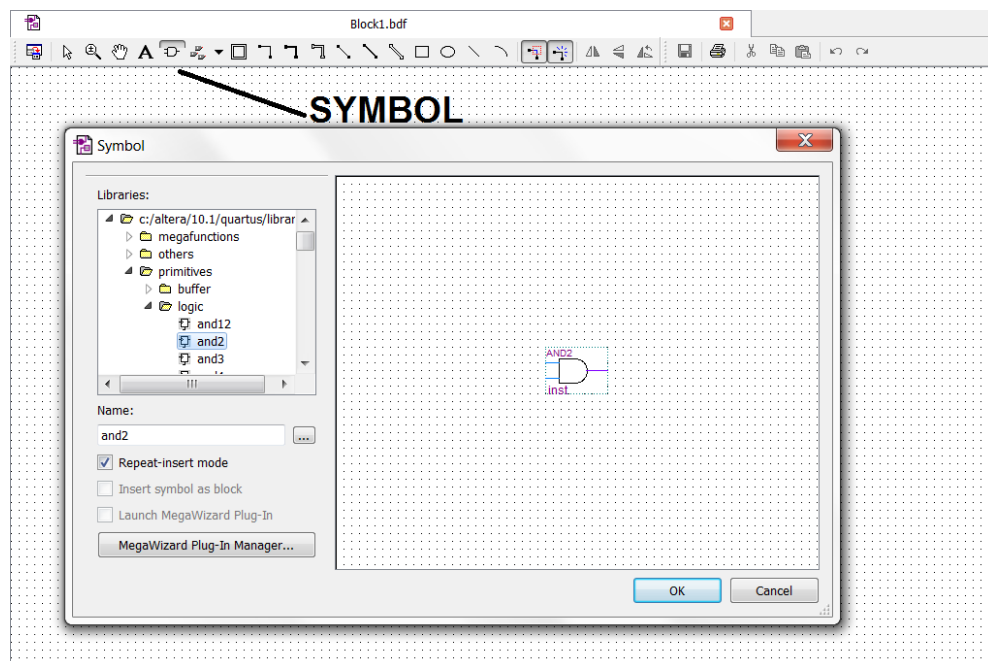
em seguida em 'New'. Selecionar a opção *Block Diagram/Schematic File* como ilustra a Figura 31.

Figura 31 - Seleção de um novo projeto usando circuitos lógicos



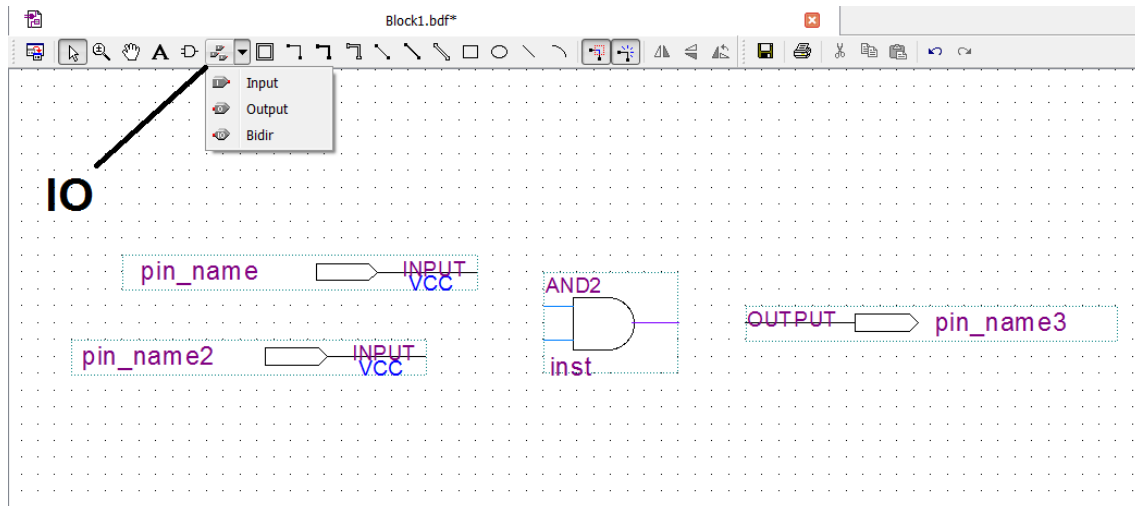
Para iniciar a elaboração de um circuito é necessária a seleção das portas lógicas desejadas e logo depois as entradas e saídas. No item *Symbol* pode ser selecionada a função lógica desejada através da biblioteca instalada junto ao *software*. No exemplo da Figura 32 é ilustrada a seleção de uma porta do tipo AND.

Figura 32 - Seleção de uma porta lógica AND



Para adicionar as entradas e as saídas selecionar o item apresentado na Figura 33. Logo depois realizar a ligação do circuito e nomear os pinos.

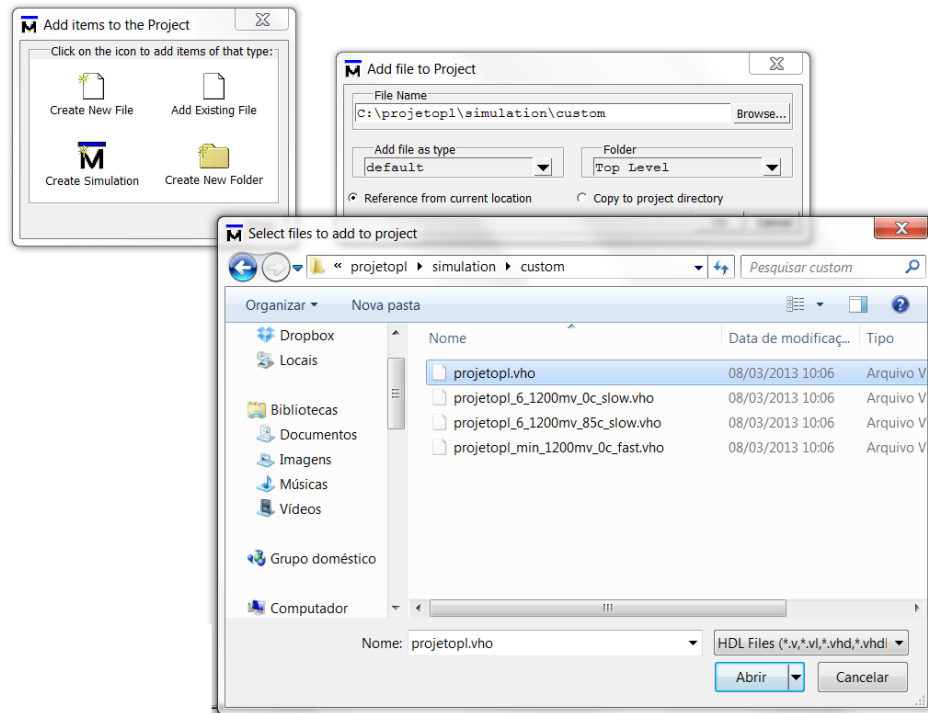
Figura 33 - Seleção das entradas e saídas



Após a criação do circuito desejado salvar o projeto e realizar a compilação do mesmo em '*Start Compilation*'. O próximo passo consiste na execução do programa **ModelSim-Altera 6.6c (Quartus II 10.1) Starter Edition**.

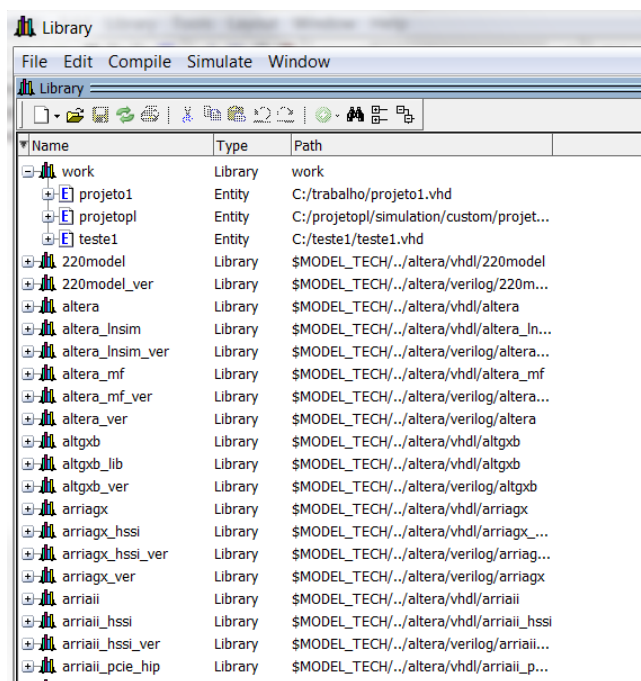
Logo na tela de apresentação selecionar o item '*File*' < '*New*' < '*Project*'. Uma janela é aberta aonde deve ser escolhido o nome do projeto de simulação e o diretório onde o mesmo deve ser salvo. Após a realização deste passo selecionar a opção '*Add Existing File*' para carregar o arquivo que foi compilado pelo Quartus. No exemplo da Figura 34 esse arquivo se encontra no diretório: C:\projktop\simulation\custom. A extensão do arquivo é ".vho".

Figura 34 - Seleção do arquivo do projeto para simulação



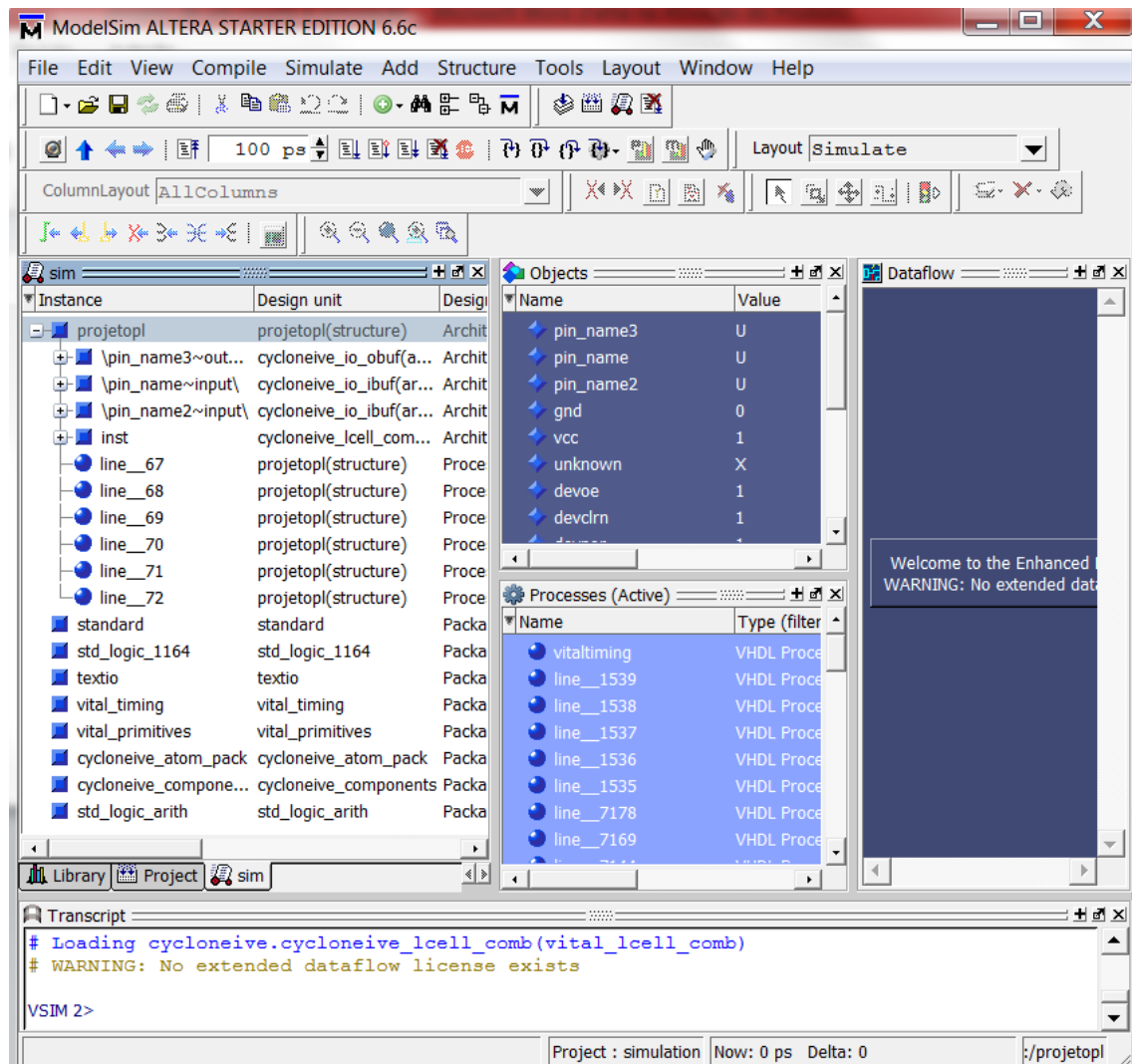
Uma janela será aberta listando o projeto selecionado. Será exibido um ícone de interrogação até que seja compilado esse arquivo. Para realizar a compilação selecionar o item 'Compile' < 'Compile All'. Na janela *Library* selecionar o item *Work* (Figura 35).

Figura 35 - Biblioteca contendo os projetos de simulação compilados



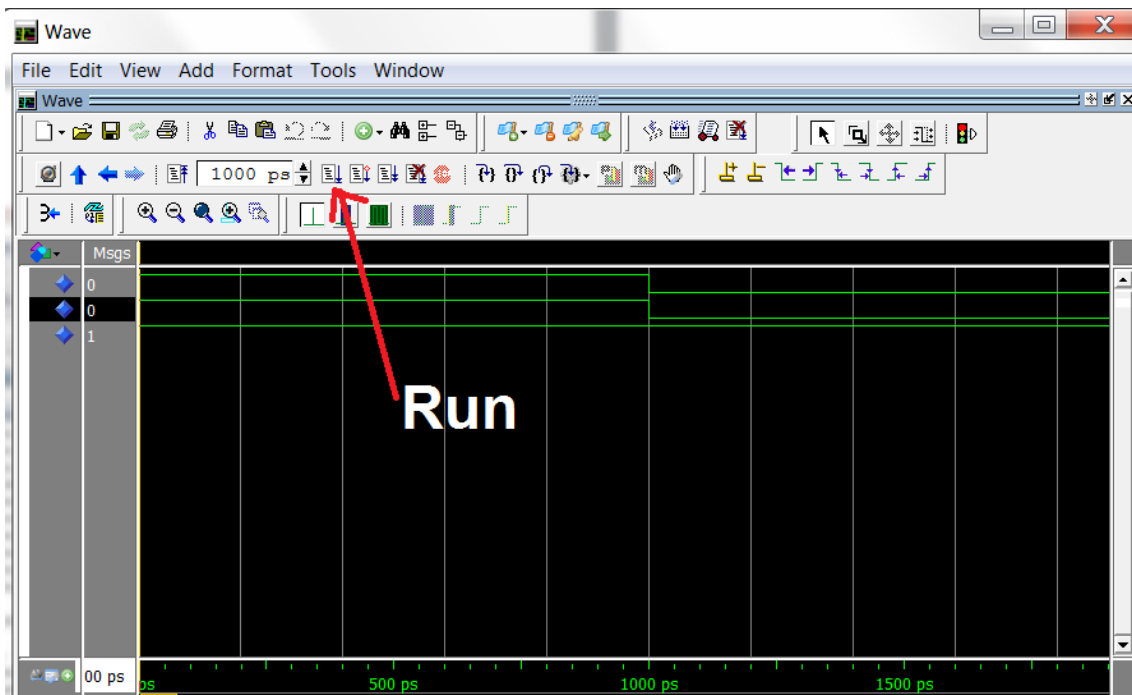
Em seguida abrir o arquivo correspondente ao projeto. Uma nova tela será apresentada como ilustra a Figura 36.

Figura 36 - Tela de simulação com as entradas e saídas do projeto



Para a visualização das formas de onda no tempo devem ser selecionadas as portas desejadas e em seguida o item *Add<To wave< Selected Signals*. Uma tela no domínio do tempo será apresentada. Para simular o sinal lógico '1' em uma das entradas selecionar a entrada desejada e em seguida o item *Edit<Force*. Atribuir o valor '1' na célula 'Value'. Após a atribuição dos valores selecionar o item 'Run' como ilustra a Figura 37. A simulação será realizada de acordo com o tempo que é atribuído, o *Default* do programa está em 100 pico-segundos.

Figura 37 - Tela de simulação com as entradas e saídas do projeto



5.4 ALGORITMO DE PROGRAMAÇÃO

Pelo fato do trabalho envolver programação faz-se necessária a construção de algoritmos que descrevem as instruções que devem ser seguidas para a execução das tarefas desejadas. Nos subtópicos seguintes são apresentados os algoritmos fundamentais que são utilizados no trabalho.

5.4.1 Algoritmo de uma modulação simples PWM

Inicialmente é proposto um algoritmo para descrever o processo que produz uma saída PWM. Nesse processo é utilizado o oscilador interno de 50 MHz. A estratégia básica do algoritmo utiliza a lógica de contadores. De acordo com a frequência que é desejada da portadora e a frequência do sinal modulante são feitos cálculos que determinam o número necessário de transições positivas de clock.

A determinação do valor da contagem de transições positivas de clock é realizada respeitando a seguinte fórmula:

$$N_x = \frac{f_{clock}}{f \times n} \quad (8)$$

O termo f_{clock} corresponde a frequência do oscilador do kit DE0-nano. Nesse kit ela corresponde a 50 MHz. O termo n corresponde ao número de amostras em um período do sinal. O termo f corresponde a frequência desejada do sinal na saída tanto da senoide como da portadora triangular. Portanto existirão dois valores de N sendo que o mesmo não pode ser menor que uma unidade e pertencer ao conjunto dos números inteiros. Logo a frequência desejada multiplicada pelo número de valores do vetor não pode ultrapassar a frequência do clock (50 MHz).

Ao atingir o número calculado de transações de subidas de clock, uma variável é carregada com o primeiro valor de um vetor com n valores amostrados da portadora ou do sinal modulante.

Para a determinação do vetor de valores de n amostras da onda senoidal a seguinte fórmula deve ser utilizada.

$$A_{senoide}[x] = A_s \cdot \sin \frac{2\pi x}{n} \quad (9)$$

para x variando de $[1, n]$

Onde: A_s - amplitude do sinal.

Para determinar o vetor de valores da portadora, inicialmente deve-se conhecer o número de níveis do conversor que será trabalhado. Como citado nos capítulos anteriores o número de portadoras é uma unidade a menos que o número de níveis.

No caso de um conversor que necessite apenas de uma portadora as seguintes fórmulas podem ser utilizadas para a determinação dos valores das n amostras de um vetor triangular:

$$A_{tri1}[x] = \frac{4 \cdot A_t \cdot x}{n} \quad \text{para } x \text{ variando de } \left[1, \frac{n}{4} \right] \quad (10)$$

$$A_{tri2}[x] = \left(2 - \frac{4 \cdot x}{n}\right) \cdot A_t \quad \text{para } x \text{ variando de } \left[\frac{n}{4} + 1, \frac{3n}{4}\right] \quad (11)$$

$$A_{tri3}[x] = \left(\frac{4 \cdot x}{n} - 4\right) \cdot A_t \quad \text{para } x \text{ variando de } \left[\frac{3n}{4} + 1, n\right] \quad (12)$$

$$A_{tri} = (A_{tri1}, A_{tri2}, A_{tri3}) \quad (13)$$

A_{tri} corresponde a amplitude do sinal da portadora triangular

A_{tri1} corresponde a primeira rampa de subida da onda triangular

A_{tri2} corresponde ao período de descida do sinal da portadora

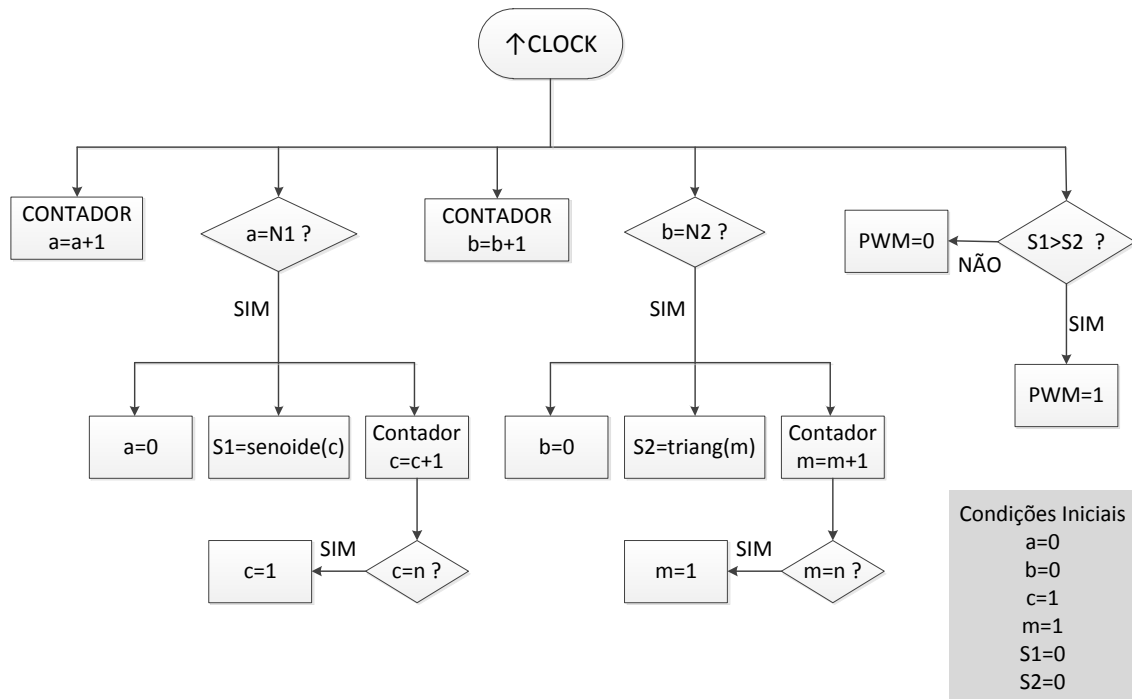
A_{tri3} corresponde a segunda rampa de subida

O índice de modulação de amplitude do algoritmo é determinado pela equação (14):

$$m_a = \frac{A_s}{A_t} \quad (14)$$

O fluxograma da Figura 38 ilustra o algoritmo completo que é utilizado para a criação de um único sinal PWM que basicamente é a comparação de uma onda triangular (portadora em alta frequência) com uma onda senoidal (baixa frequência).

Figura 38 - Fluxograma de modulação de uma saída PWM



Vale ressaltar que a análise do fluxograma é paralela devido ao processamento do dispositivo. Ou seja, logo após a subida de clock são analisados em paralelo os três eventos de decisões mencionados ($a=N1$; $b=N2$; $S1>S2$) e os dois eventos de contagem ($a=a+1$; $b=b+1$).

$N1$ e $N2$ correspondem aos valores, determinados pela equação (8), de contagens de subida de clock respectivos para a senoide e para a portadora.

No caso do primeiro evento de decisão ser verdadeiro ($a=N1$) é zerada a variável de contagem a , carregado em uma variável $S1$ a amostra c do vetor senoide e incrementado uma unidade na variável c , que corresponde ao número da amostra do vetor. O segundo evento de decisão ocorre de forma análoga

O terceiro evento de decisão corresponde a comparação entre a variável $S1$ e a variável $S2$. Na condição em que $S1$ é maior que $S2$ a saída digital possuirá nível lógico 1. Na condição contrária a saída possuirá nível lógico 0.

5.4.2 Algoritmo da modulação do conversor NPC três níveis

Para o caso de um conversor com três níveis o algoritmo deve possuir duas variáveis que carreguem dois vetores diferentes de valores amostrados de uma onda triangular. A determinação de um dos vetores pode ser obtida através das seguintes equações:

$$A_{tri1}[x] = \frac{2 \cdot A \cdot x}{n} \quad \text{para } x \text{ variando de } \left[1, \frac{n}{2} \right] \quad (15)$$

$$A_{tri2}[x] = \left(2 - \frac{2 \cdot x}{n} \right) \cdot A \quad \text{para } x \text{ variando de } \left[\frac{n}{2} + 1, n \right] \quad (16)$$

$$A_{tri} = (A_{tri1}, A_{tri2}) \quad (17)$$

A_{tri} corresponde a amplitude do sinal da portadora triangular

A_{tri1} corresponde a subida da rampa da onda triangular

A_{tri2} corresponde ao período de descida do sinal da portadora

Através das equações apresentadas obtêm-se o vetor da portadora positiva, por exemplo, se a amplitude fosse escolhida como 1000, A_{tri1} iria variar entre 0 até 1000 e A_{tri2} de 1000 até 0. Objetivando a simplificação do código, esse mesmo vetor pode ser aproveitado para a determinação da portadora negativa, ou seja, para o mesmo vetor do exemplo anterior bastaria multiplicar todos os valores do vetor por -1. Desse modo seriam obtidas duas portadoras. No desenvolvimento do programa basta realizar o carregamento do valor negativo das amostras em uma outra variável. As equações 18 e 19 mostram as atribuições do vetor às variáveis.

$$S_2 = A_{tri}(x) \quad (18)$$

$$S_3 = -A_{tri}(x) \quad (19)$$

Analisando as equações apresentadas a variável S3 possuirá o valor negativo das amostras, dessa maneira obtém-se uma modulação do tipo POD ou APOD, pelo fato das portadoras ficarem defasadas entre si em 180°.

A determinação dos valores do vetor senoidal ainda respeita a equação (1), assim como os valores de contagem $N1$ e $N2$.

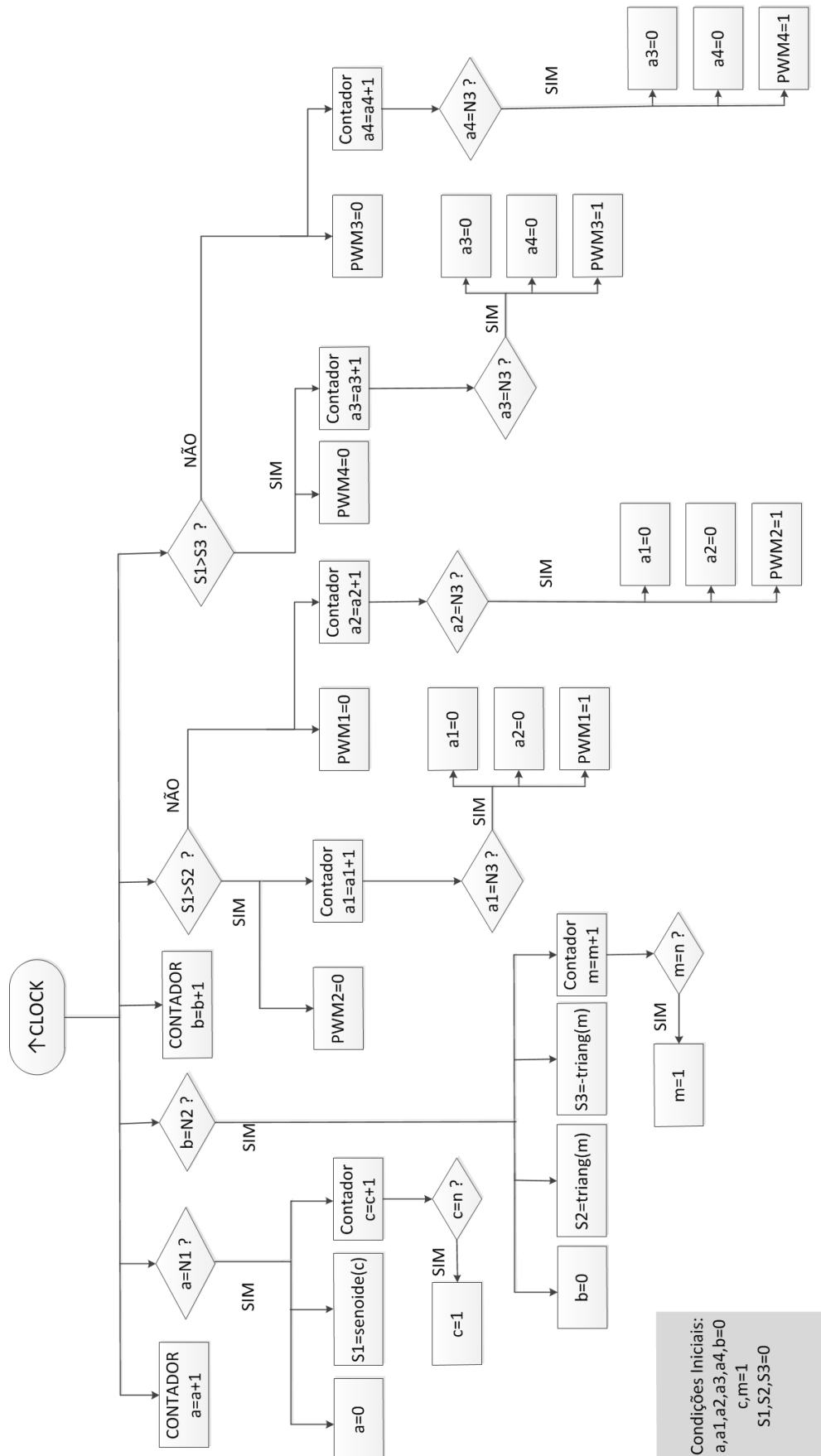
Nesse caso é adicionada a configuração com tempo morto. Para a execução de mais essa tarefa pode ser utilizado o uso de contadores objetivando causar um atraso na excursão de sinal (0 para 1) na saída digital. Assim mostra-se necessário o uso da equação (20).

$$N_x = \frac{f_{clock}}{100 \times f} \times P_{tempomorto} \quad (20)$$

O termo f_{clock} corresponde a frequência do oscilador. O termo n corresponde ao número de amostras em um período do sinal. O termo f corresponde a frequência desejada do sinal na portadora.

O algoritmo completo é apresentado na Figura 39.

Figura 39 - Fluxograma de modulação PWM para conversor NPC três níveis monofásico



Logo após a subida de clock são processados em paralelo os quatro eventos de comparação mencionados ($a=N1$; $b=N2$; $S1>S2$; $S1>S3$) e os dois eventos de contagem ($a=a+1$; $b=b+1$).

No caso do primeiro evento de decisão, que corresponde ao carregamento dos valores da senoide, ser verdadeiro ($a=N1$) é zerada a variável de contagem a , carregado em uma variável $S1$ a amostra c do vetor senoide e incrementado uma unidade na variável c , que corresponde ao número da amostra do vetor.

No caso do segundo evento de decisão ser verdadeiro ($b=N2$) é zerada a variável de contagem b , ocorre o carregamento das variáveis $S2$ e $S3$, e é dado um incremento na variável m correspondente ao número do vetor.

O terceiro evento de decisão corresponde a comparação entre a variável $S1$ e a variável $S2$. Na condição em que $S1$ é maior que $S2$ uma das saídas é zerada e a outra só comuta para o nível lógico 1 após um atraso. Esse atraso trata-se da função tempo morto.

5.5 PROGRAMA EM LINGUAGEM VHDL DE MODULAÇÃO PWM PARA CONVERSOR NPC TRÊS NÍVEIS MONOFÁSICO COM TEMPO-MORTO

Após feita a análise do fluxograma da Figura 39 é apresentado o código em VHDL para a utilização em um conversor NPC três níveis com quatro chaves. O usuário tem a opção de optar pela frequência da portadora, frequência da moduladora senoidal e a porcentagem de tempo-morto.

Utilizando a equação (8), sabendo que o limite mínimo de N contagens de subida de clock deve ser uma unidade e estabelecendo uma portadora de 80 kHz define-se um vetor com 625 valores. O usuário é livre para modificar a frequência da portadora desde que essa não ultrapasse 80 kHz.

O código é apresentado abaixo:

```
library ieee;
use ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.ALL;

entity modulacao is      --declaracao da entidade (I/O de/para operações externas)
port (clk: in std_logic;      --declaração da entrada clock
      pwm1,pwm2,pwm3,pwm4: out bit); --declaração das variáveis de saída
```

end entity modulacao; --fim da entidade

architecture estrutura of modulacao is --início da arquitetura

shared variable a : integer :=0; --declaração de variável p/contagem subidas do clock
 shared variable b : integer :=0; --declaração de variável p/contagem subidas do clock
 shared variable a1 : integer :=0; --declaração de variável p/contagem subidas do clock
 shared variable a2 : integer :=0; --declaração de variável p/contagem subidas do clock
 shared variable a3 : integer :=0; --declaração de variável p/contagem subidas do clock
 shared variable a4 : integer :=0; --declaração de variável p/contagem subidas do clock
 shared variable count1 : integer :=1; --declaração variável que aponta n do vetor
 shared variable count2 : integer :=1; --declaração variável que aponta n do vetor
 type triangular is array (integer range <>) of integer; -- declaração do vetor triangular
 de 625 amostras entre 0 e 1000 --

shared variable triangular1:triangular (1 to 625) :=

(3,6,10,13,16,19,22,26,29,32,35,38,42,45,48,51,54,58,61,64,67,70,74,77,80,83,86,90,93,96,99,102,106,109,112,115,118,122,125,128,131,134,138,141,144,147,150,154,157,160,163,166,170,173,176,179,182,186,189,192,195,198,202,205,208,211,214,218,221,224,227,230,234,237,240,243,246,250,253,256,259,262,266,269,272,275,278,282,285,288,291,294,298,301,304,307,310,314,317,320,323,326,330,333,336,339,342,346,349,352,355,358,362,365,368,371,374,378,381,384,387,390,394,397,400,403,406,410,413,416,419,422,426,429,432,435,438,442,445,448,451,454,458,461,464,467,470,474,477,480,483,486,490,493,496,499,502,506,509,512,515,518,522,525,528,531,534,538,541,544,547,550,554,557,560,563,566,570,573,576,579,582,586,589,592,595,598,602,605,608,611,614,618,621,624,627,630,634,637,640,643,646,650,653,656,659,662,666,669,672,675,678,682,685,688,691,694,698,701,704,707,710,714,717,720,723,726,730,733,736,739,742,746,749,752,755,758,762,765,768,771,774,778,781,784,787,790,794,797,800,803,806,810,813,816,819,822,826,829,832,835,838,842,845,848,851,854,858,861,864,867,870,874,877,880,883,886,890,893,896,899,902,906,909,912,915,918,922,925,928,931,934,938,941,944,947,950,954,957,960,963,966,970,973,976,979,982,986,989,992,995,998,999,995,992,989,986,982,979,976,973,970,966,963,960,957,954,950,947,944,941,938,934,931,928,925,922,918,915,912,909,906,902,899,896,893,890,886,883,880,877,874,870,867,864,861,858,854,851,848,845,842,838,835,832,829,826,822,819,816,813,810,806,803,800,797,794,790,787,784,781,778,774,771,768,765,762,758,755,752,749,746,742,739,736,733,730,726,723,720,717,714,710,707,704,701,698,694,691,688,685,682,678,675,672,669,666,662,659,656,653,650,646,643,640,637,634,630,627,624,621,618,614,611,608,605,602,598,595,592,589,586,582,579,576,573,570,566,563,560,557,554,550,547,544,541,538,534,531,528,525,522,518,515,512,509,506,502,499,496,493,490,486,483,480,477,474,470,467,464,461,458,454,451,448,445,442,438,435,432,429,426,422,419,416,413,410,406,403,400,397,394,390,387,384,381,378,374,371,368,365,362,358,355,352,349,346,342,339,336,333,330,326,323,320,317,314,310,307,304,301,298,294,291,288,285,282,278,275,272,269,266,262,259,256,253,250,246,243,240,237,234,230,227,224,221,218,214,211,208,205,202,198,195,192,189,186,182,179,176,173,170,166,163,160,157,154,150,147,144,141,138,134,131,128,125,122,118,115,112,109,106,102,99,96,93,90,86,83,80,77,74,70,67,64,61,58,54,51,48,45,42,38,35,32,29,26,22,19,16,13,10,6,3,0);

type senoide is array (integer range <>) of integer; --declaração do vetor senoide 625 amostras entre -950 e 950. Indice de modulação de amplitude igual a 0,95

shared variable senoide1:senoide (1 to 625):=

(10,19,29,38,48,57,67,76,86,95,105,114,124,133,143,152,162,171,180,190,199,208,218,227,236,245,
 ,255,264,273,282,291,300,309,318,327,336,345,354,363,372,381,389,398,407,415,424,432,441,449,
 458,466,474,483,491,499,507,515,523,531,539,547,555,562,570,578,585,593,600,607,615,622,629,6
 36,643,650,657,664,671,678,684,691,697,704,710,717,723,729,735,741,747,753,759,764,770,776,78
 1,786,792,797,802,807,812,817,822,827,831,836,840,845,849,853,858,862,866,869,873,877,881,884
 ,888,891,894,897,901,904,906,909,912,915,917,920,922,924,926,928,930,932,934,936,937,939,940,
 942,943,944,945,946,947,947,948,949,949,949,950,950,950,950,950,950,949,949,948,948,947,946,9
 45,944,943,942,941,940,938,937,935,933,931,929,927,925,923,921,918,916,913,911,908,905,902,89
 9,896,893,889,886,882,879,875,871,868,864,860,855,851,847,843,838,834,829,824,820,815,810,805
 ,800,794,789,784,778,773,767,761,756,750,744,738,732,726,720,713,707,701,694,688,681,674,668,
 661,654,647,640,633,626,618,611,604,596,589,581,574,566,558,551,543,535,527,519,511,503,495,4
 87,478,470,462,453,445,437,428,420,411,402,394,385,376,367,359,350,341,332,323,314,305,296,28
 7,278,268,259,250,241,232,222,213,204,194,185,176,166,157,147,138,129,119,110,100,91,81,72,62,
 53,43,33,24,14,5,-5,-14,-24,-33,-43,-52,-62,-72,-81,-91,-100,-110,-119,-129,-138,-147,-157,-166,-
 176,-185,-194,-204,-213,-222,-232,-241,-250,-259,-268,-278,-287,-296,-305,-314,-323,-332,-341,-
 350,-359,-367,-376,-385,-394,-402,-411,-420,-428,-437,-445,-453,-462,-470,-478,-487,-495,-503,-
 511,-519,-527,-535,-543,-551,-558,-566,-574,-581,-589,-596,-604,-611,-618,-626,-633,-640,-647,-
 654,-661,-668,-674,-681,-688,-694,-701,-707,-713,-720,-726,-732,-738,-744,-750,-756,-761,-767,-
 773,-778,-784,-789,-794,-800,-805,-810,-815,-820,-824,-829,-834,-838,-843,-847,-851,-855,-860,-
 864,-868,-871,-875,-879,-882,-886,-889,-893,-896,-899,-902,-905,-908,-911,-913,-916,-918,-921,-
 923,-925,-927,-929,-931,-933,-935,-937,-938,-940,-941,-942,-943,-944,-945,-946,-947,-948,-948,-
 949,-949,-950,-950,-950,-950,-950,-950,-949,-949,-949,-948,-947,-947,-946,-945,-944,-943,-942,-
 940,-939,-937,-936,-934,-932,-930,-928,-926,-924,-922,-920,-917,-915,-912,-909,-906,-904,-901,-
 897,-894,-891,-888,-884,-881,-877,-873,-869,-866,-862,-858,-853,-849,-845,-840,-836,-831,-827,-
 822,-817,-812,-807,-802,-797,-792,-786,-781,-776,-770,-764,-759,-753,-747,-741,-735,-729,-723,-
 717,-710,-704,-697,-691,-684,-678,-671,-664,-657,-650,-643,-636,-629,-622,-615,-607,-600,-593,-
 585,-578,-570,-562,-555,-547,-539,-531,-523,-515,-507,-499,-491,-483,-474,-466,-458,-449,-441,-
 432,-424,-415,-407,-398,-389,-381,-372,-363,-354,-345,-336,-327,-318,-309,-300,-291,-282,-273,-
 264,-255,-245,-236,-227,-218,-208,-199,-190,-180,-171,-162,-152,-143,-133,-124,-114,-105,-95,-86,-
 76,-67,-57,-48,-38,-29,-19,-10,0);

shared variable saida : integer :=0; --variável p/ carregamento de um vetor senoidal

shared variable saidat1 : integer:=0; --variável p/ carregamento de um vetor triangular

shared variable saidat2 : integer:=0; --variável p/ carregamento de um vetor triangular

shared variable ptempomorto: integer :=1; --porcentagem de tempo-morto (1%)

shared variable freqsen : integer :=50; --frequência desejada do sinal modulante

shared variable freqtri : integer :=80000; --frequência desejada das portadoras

shared variable N1: integer :=1; --variável dependente da freq. do sinal modulante

shared variable N2: integer :=1; --variável dependente da freq. desejada da portadora

shared variable N3: integer :=1; --variável dependente da porcent. de tempo morto

begin --início

process(clk) --início do processamento de clock

begin --início

```

if clk='1' then    --executa em toda subida de clock
a:=1+a;    --incrementa 1 na variável a
b:=1+b;    --incrementa 1 na variável b
N1:=50000000/(freqsen*625);    --cálculo de N1 (contagens de clock)
N2:=50000000/(freqtri*625);    --cálculo de N2 (contagens de clock)
N3:=(50000000*ptempomorto)/(freqtri*100);    --cálculo de N3 (contagens de clock)

if a=N1 then    --executa quando a for igual a N1 subidas de clock
saida := senoide1(count1);    --carregamento vetor senoide na variável saída
count1:=1+count1; --incrementa 1 na variável count1
    if count1=625 then --função para limitar count1 em 625
count1:=1; --quando count 1 atinge 625 assume valor 1
    end if;
a:=0; --zera variavel a
end if;

if b=N2 then    --executa quando a for igual a N2 subidas de clock
saidat1 := triangular1(count2);    -- vetor triangular na variável saidat1 (0 à +1000)
saidat2 :=-triangular1(count2);    -- vetor triangular negado na variável saidat2
count2:=1+count2; --incrementa 1 na variável count2
b:=0; --zera variável b
    if count2=625 then --função para limitar count2 em 625
count2:=1; -- quando count 2 atinge 625 assume valor 1
    end if;

end if;

    if saida > saidat1 then    -- comparação entre os valores carregados da
                                onda senoidal e da portadora positiva

pwm2<='0';    --reset pwm2
a1:=1+a1; --incrementa 1 na variável a1
    if a1=N3 then --tempo-morto, executa quando a1 for igual a N3 subidas de clock
    (quando pwm2 comutar para '0' a saída pwm1 terá um atraso ate comutar para '1').
pwm1<='1'; --set pwm1
a1:=0; --zera variavel a1
a2:=0; --zera variavel a2
    end if;

else
pwm1<='0';    --reset pwm1
a2:=1+a2; --incrementa 1 na variável a2
    if a2=N3 then --tempo-morto, executa quando a2 for igual a N3 subidas de clock (quando
pwm1 comutar para '0' a saída pwm2 terá um atraso até comutar para '1')
pwm2<='1'; --set pwm2
a2:=0; --zera variavel a2
a1:=0; --zera variavel a1
    end if;

end if;

```

```

        if saida > saidat2 then                                --comparação entre os valores carregados da onda
                                                                senoidal e da portadora negativa
pwm4<='0'; --reset pwm4
a3:=1+a3;
    if a3=N3 then --tempo-morto, executa quando a3 for igual a N3 subidas de clock
        (quando pwm4 comutar para '0' a saída pwm3 terá um atraso até comutar para '1')
        pwm3<='1'; --set pwm3
        a3:=0; --zera variavel a3
        a4:=0; --zera variavel a4
        end if;
    else
pwm3<='0'; --reset pwm3
a4:=1+a4;
    if a4=N3 then --tempo-morto, executa quando a1 for igual a N3 subidas de clock (quando
        pwm3 comutar para '0' a saída pwm4 terá um atraso até comutar para '1')
        pwm4<='1'; --set pwm4
        a3:=0; --zera variavel a3
        a4:=0; --zera variavel a4
        end if;
    end if;
end if;
end process; --fim do processo de leitura do clock
end architecture estrutura; --fim da estrutura

```

As chaves correspondem as saídas pwm1,pwm2,pwm3 e pwm4.

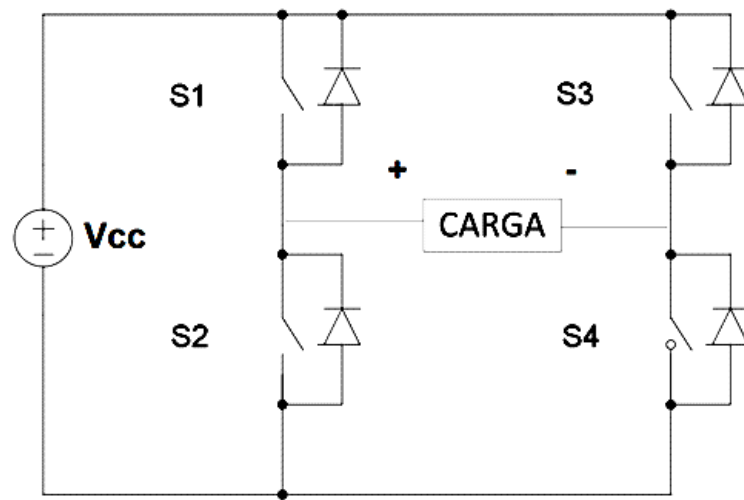
No Apêndice A é apresentado o código para um conversor trifásico utilizando três níveis. O mesmo possui doze saídas que comutam as doze chaves do conversor.

5.6 IMPLEMENTAÇÃO EXPERIMENTAL EM CONVERSORES CHAVEADOS

No mesmo laboratório do desenvolvimento desse trabalho foi realizada uma aplicação prática do programa desenvolvido e apresentado no tópico anterior, efetuando a modulação três níveis em um conversor monofásico ponte completa.

O acadêmico Enrique Martinez Gavilán realizou a montagem de um conversor ponte completa operando a três níveis e disponibilizou o conversor para teste com o programa desenvolvido. O esquemático do conversor ponte completa é apresentado na Figura 40.

Figura 40 - Conversor monofásico ponte completa



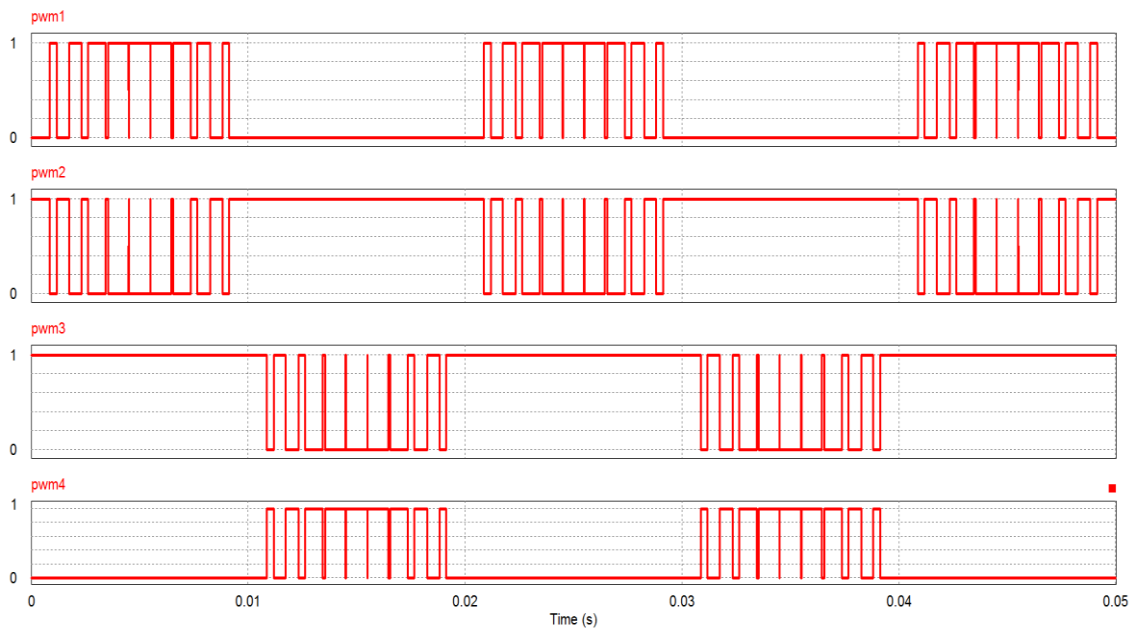
A sequência de chaveamento para operação em três níveis desse conversor deve seguir a Tabela 4 apresentada abaixo, onde V_o é a tensão na carga.

Tabela 4 - Sequência de chaveamento do conversor ponte completa

S1	S2	S3	S4	V_o
1	0	0	1	V_{cc}
0	1	0	1	0
1	0	1	0	0
0	1	1	0	$(-V_{cc})$

Como experimentalmente esse conversor já possui um driver para a realização do tempo-morto, essa função não foi utilizada. Nesse caso as formas de onda das chaves para a operação sem tempo-morto são mostradas na Figura 41.

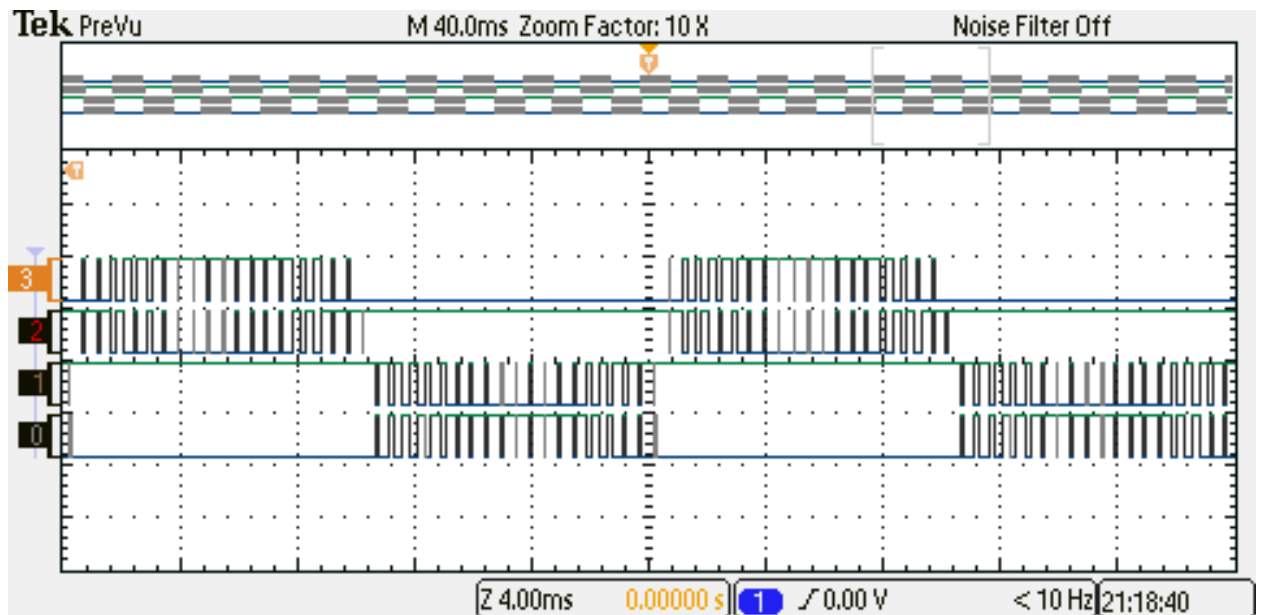
Figura 41 - Comando das chaves do conversor monofásico ponte completa



As saídas *pwm1* e *pwm2* fazem a modulação de um braço do conversor (chaves S1 e S2, respectivamente) enquanto as saídas *pwm 3* e *pwm 4* fazem a modulação do outro braço (chaves S4 e S3, respectivamente).

Utilizando o programa apresentado no tópico anterior e com o auxílio de um osciloscópio digital é analisada a saída da tensão dos quatro comandos para as chaves no FPGA como mostra a Figura 42. Sabe-se que cada saída possui uma tensão de 3,3V em relação a referência. Essas formas de onda de tensão foram obtidas utilizando o barramento GPIO-0 do kit do FPGA.

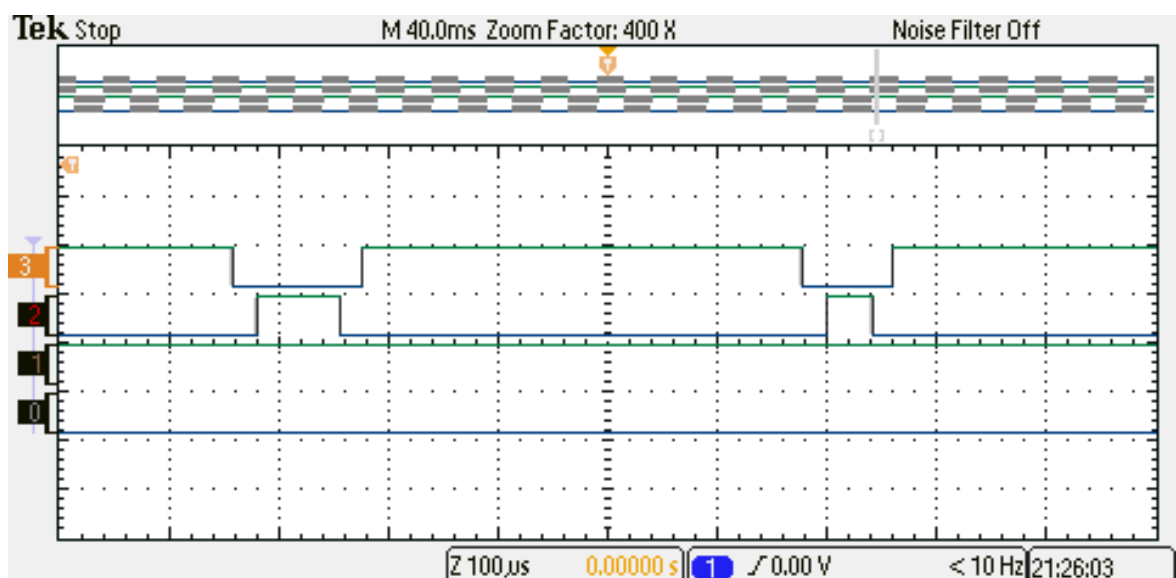
Figura 42 - Formas de onda de tensão na saída do FPGA para modulação 3 níveis



Para efeitos de visualização da comutação das chaves foi utilizada uma frequência de portadoras de 2 KHz e frequência de referência 50 Hz.

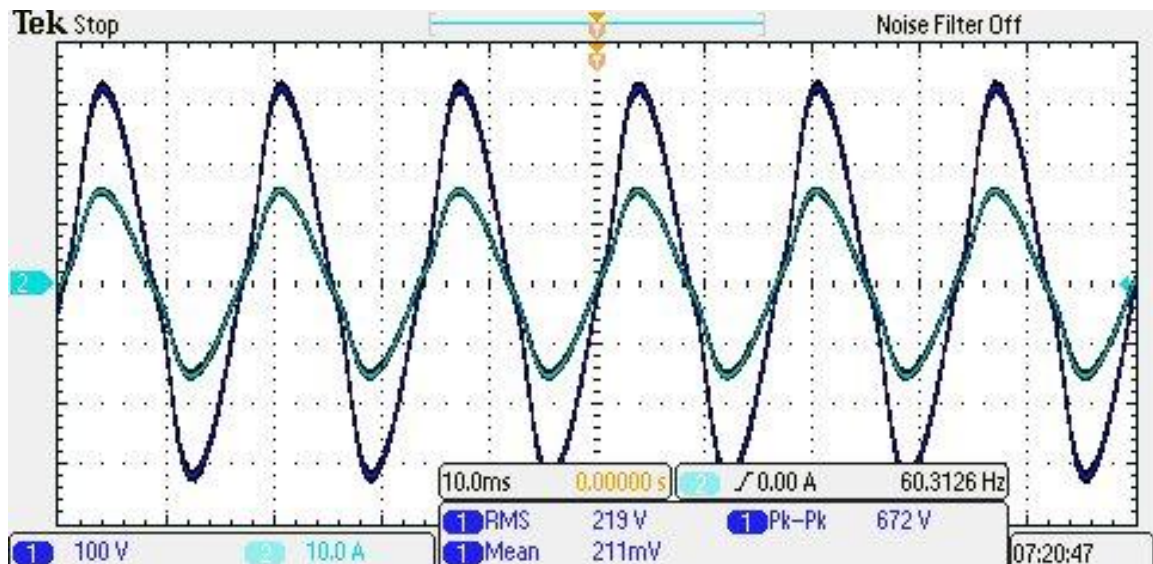
A Figura 43 apresenta uma análise das ondas de tensão de saída do FPGA em uma escala pequena de tempo para visualização do tempo-morto.

Figura 43 - Tempo-morto na modulação três níveis



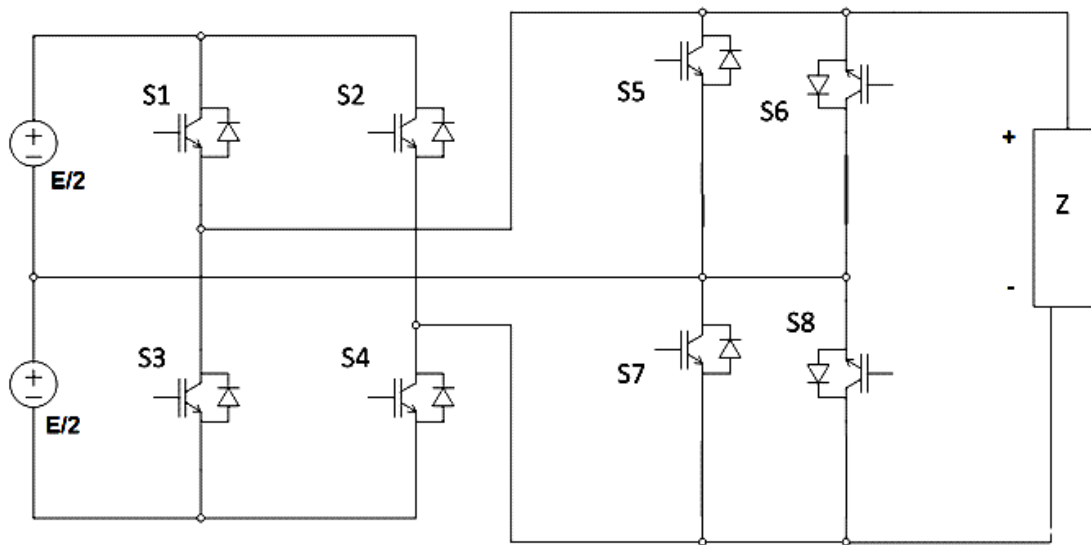
A Figura 44 mostra as formas de onda da tensão e da corrente retiradas na carga do conversor ponte completa. O teste é realizado com uma frequência modulante de 60 Hz e frequência de portadora de 50 kHz.

Figura 44 - Formas de onda experimentais da tensão na saída do conversor ponte completa



Outra aplicação prática foi realizada em um conversor que opera à cinco níveis em estudo pelo mestrando Tiago Lemes da Silva no laboratório do nPEE da UDESC. O conversor possui oito chaves de comutação como mostra a Figura 45. Não é intenção nesse trabalho explorar em detalhes o funcionamento do mesmo. Apenas apresentar uma solução para a modulação do mesmo, como realizado no conversor a três níveis citado anteriormente.

Figura 45 - Conversor NPCm monofásico à 5 níveis



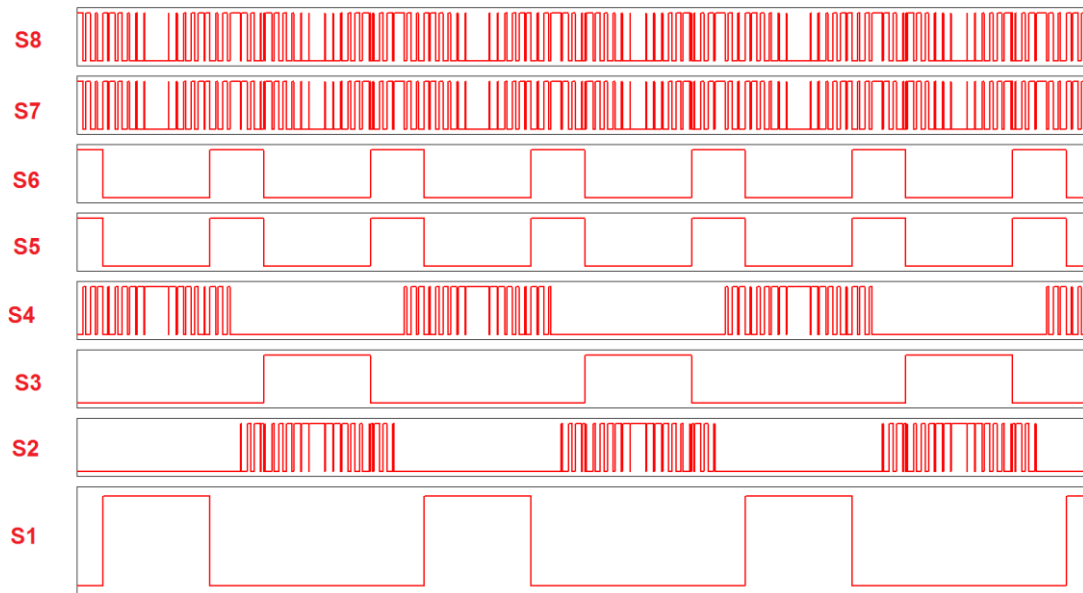
A sequência de chaveamento desse conversor é mostrada Tabela 5 mostrada a seguir, onde V_{cc} é a tensão somada das duas fontes do conversor e V_o a tensão na carga do mesmo.

Tabela 5- Estados possíveis das chaves do conversor monofásico NPCm 5 níveis

S1	S2	S3	S4	S5	S6	S7	S8	V0
0	0	0	0	1	1	1	1	0
0	0	0	1	1	1	0	0	$V_{CC}/2$
1	0	0	0	0	0	1	1	$V_{CC}/2$
1	0	0	1	0	0	0	0	V_{CC}
1	0	0	0	0	0	1	1	$V_{CC}/2$
0	0	0	1	1	1	0	0	$V_{CC}/2$
0	0	0	0	1	1	1	1	0
0	0	0	0	1	1	1	1	0
0	1	0	0	1	1	0	0	$(-V_{CC}/2)$
0	0	1	0	0	0	1	1	$(-V_{CC}/2)$
0	1	1	0	0	0	0	0	$(-V_{CC})$
0	0	1	0	0	0	1	1	$(-V_{CC}/2)$
0	1	0	0	1	1	0	0	$(-V_{CC}/2)$
0	0	0	0	1	1	1	1	0

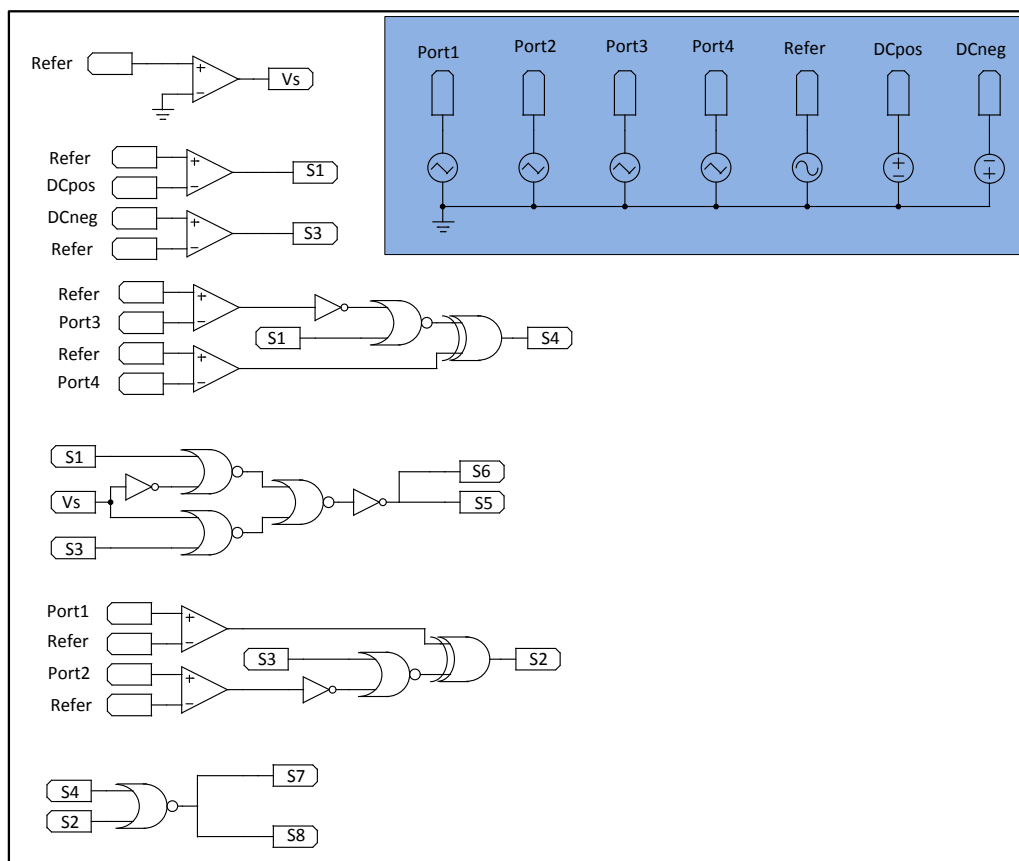
Com a tabela em mãos as formas de onda da tensão nas chaves são mostradas na Figura 46. Essa modulação possui quatro portadoras e foi testada uma estratégia de modulação do tipo IPD.

Figura 46 - Chaveamento do conversor à cinco níveis



Uma estratégia indicada pelo mestrando Tiago Lemes da Silva para realizar a sequência de chaveamento é descrita por um algoritmo que utiliza portas lógicas. Esse é simulado no *software* PSIM e é ilustrado na Figura 47.

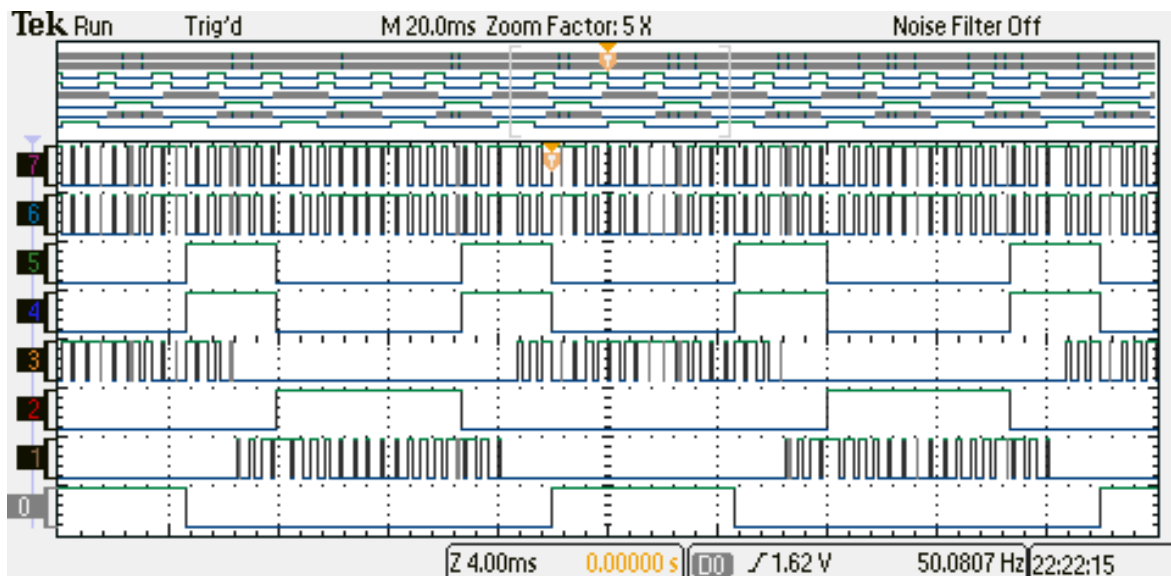
Figura 47 - Circuito lógico responsável pela modulação do conversor NPCm à 5 níveis



No apêndice B é mostrado o código de descrição em VHDL criado com base no algoritmo descrito por portas lógicas. Além disso o código é aprimorado e o usuário passa a ter a opção, através de uma chave do kit, de escolher se o dispositivo utiliza uma referência interna senoidal ou realiza a leitura de uma entrada digital de 16 bits. Assim em trabalhos posteriores poderá ser realizado o controle em malha fechada do conversor através do FPGA.

As formas de onda de tensão obtidas nas saídas do FPGA que comutarão as chaves do conversor à cinco níveis são mostradas na Figura 48. Para este exemplo foi escolhida uma frequência de referência senoidal de 50 Hz.

Figura 48 - Formas de onda de tensão obtidas nos pinos de saída do FPGA para o Conversor NPCm 5 níveis



No apêndice C é mostrado o código de descrição em VHDL para o conversor ponte completa com 3 níveis utilizando uma estratégia de modulação PSD.

6 CONCLUSÕES FINAIS

Nesse trabalho foi possível realizar e apresentar com sucesso algumas das técnicas de modulação de conversores multiníveis descritas por linguagem VHDL e a posterior implementação em um conversor ponte completa. Foi apresentada também toda a metodologia necessária para a criação dos códigos que envolvia a construção de algoritmos, estudo da linguagem VHDL, apresentação de topologias de diferentes conversores multiníveis e estratégias de modulação.

Além do *software* de programação Quartus 10.1, foram utilizados também os seguintes programas: PSIM, ModelSIM, Microsoft Visio, Microsoft Excel, Microsoft Word.

Um dos objetivos do trabalho é ser uma fonte de pesquisa e auxílio para futuros trabalhos e estudos que envolvam modulação PWM utilizando FPGA em conversores multiníveis. Por isso procurou-se apresentar o conteúdo de forma didática utilizando recursos como: apresentação de várias ilustrações tanto das topologias como das formas de onda, comentários nas linhas dos códigos, descrição de passos para criação de projetos e a realização de simulações, entre outros.

Fica como sugestão para trabalhos futuros: a) a otimização do código apresentado; b) implementação de técnicas de controle digital em conversores trabalhando em malha fechada; c) aquisição de valores analógicos com a utilização do conversor AD; d) programação por portas lógicas dos problemas apresentados nesse trabalho; e) utilização do microcontrolador interno NIOS-II para controle em malha fechada.

REFERÊNCIAS BIBLIOGRÁFICAS

- [1] RECH, Cassiano. Analise, Projeto e Desenvolvimento de Sistemas Multiníveis Híbridos. Tese (Doutorado em Engenharia Elétrica) – Universidade Federal de Santa Maria, Santa Maria-RS-2005.
- [2] BATSCHAUER, Alessandro Luiz. Inversor Multiníveis Híbrido Trifásico Baseado em Módulos Meia-Ponte. Tese Doutorado. UFSC. Florianópolis. 2011.
- [3] FAR, Zahra Daneshi; RADAN, Ahmad; FAR, Mehrdad Davari. Introduction and Evaluation of Novel Multi-level Carrier-Based PWM Strategies Using a Generalized Algorithm.
- [4] WATKINS, S.J., "Optimal Control of Multilevel Flying-Capacitor Converters", PhD Thesis, University of Leeds, UK, September 2005.
- [5] NABAE, Akira; TAKAHASHI, Isao; AKAGI, Hirofumi. A New Neutral-Point-Clamped PWM Inverter. 1981.
- [6] HEERDT, Joselito Anastácio. Carga Eletrônica Ativa Trifásica. Exame de Qualificação para Doutorado. UFSC. Florianópolis 2011.
- [7] BHAGWAT, Pradeep M.; STEFANOVIC V. R.. Generalized Structure of a Multilevel PWM Inverter. 1983.
- [8] BARBI, Ivo. Eletrônica de potência. 3ed. Florianópolis: Ed do Autor, 2002.
- [9] BARRETO, Eduardo Maldaun; MEZARROBA, Marcello. Inversor Multinível Npc Monofásico Com Comutação Suave E Grampeamento Ativo. 2011. Dissertação (mestrado)- Universidade do Estado de Santa Catarina, Programa de Pós-Graduação em Engenharia Elétrica, Joinville, 2011.

- [10] AVELINO, Wellington de Oliveira, Desenvolvimento de um Inversor Monofásico Utilizando Controlador Digital Baseado em FPGA. TCC (Graduação em Engenharia Elétrica). Universidade Federal do Ceara. Fortaleza. 2010.
- [11] BARBI, Ivo; MARTINS, Denizar Cruz. Introdução Ao Estudo De Conversores CC-CA. Florianópolis : ED dos Autores,2005.
- [12] COSTA, Cesar da. Projetando controladores digitais com FPGA. São Paulo: Novatec, 2006.
- [13] HUANG, Jing; CORZINE, Keith. Extended Operation of Flying Capacitor Multilevel Inverters. Department of Electrical Engineering, University of Missouri. USA.
- [14] KILTS, STEVE. Advanced FPGA design: architecture, implementation, and optimization. New Jersey: J. Wiley, 2007.
- [15] MUSSA, Samir. Synchronous State Machine Inner FPGA Controlling PFC Boost Converter. UFSC. Florianópolis-SC. 2010
- [16] WU, Bin. High-Power Converters and AC drives. IEEE, Wiley Interscience. New Jersey. 2006.
- [17] Terasic DE0-nano User manual - 2012

APÊNDICE A

A seguir é apresentado o código em VHDL para a modulação de um conversor NPC trifásico à três níveis. Foram escolhidos 200 valores de amostragem:

```
library ieee;
use ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.ALL;

entity teste1 is --declaração da entidade
port (clk: in std_logic; --declaração da entrada clock
d1,e1,f1,g1,d2,e2,f2,g2,d3,e3,f3,g3: out bit); --declaração das variáveis de saída

end entity teste1; --fim da entidade

architecture estrutura of teste1 is --início da arquitetura

    shared variable c1 : integer :=1; --valor inicial p/ c1 que define n do vetor
    shared variable c2 : integer :=66; --defasagem da senoide fase B
    shared variable c3 : integer :=132; --defasagem da senoide fase C
    shared variable a : integer :=0; -- declaração variavel a de contagem
    shared variable saida1 : integer:=0; -- declaração saída senoide Fase A
    shared variable saida2 : integer:=0; -- declaração saída senoide Fase B
    shared variable saida3 : integer:=0; -- declaração saída senoide Fase C
        type senoide is array (integer range <>) of integer; --declaração 200 valoroes
                                                amostrados para a função de referencia
        shared variable senoide1:senoide (1 to
200):=(31,63,94,125,156,187,218,249,279,309,339,368,397,426,454,482,509,536,562,588,6
13,637,661,685,707,729,750,771,790,809,827,844,861,876,891,905,918,930,941,951,960,9
69,976,982,988,992,996,998,1000,1000,1000,998,996,992,988,982,976,969,960,951,941,93
0,918,905,891,876,861,844,827,809,790,771,750,729,707,685,661,637,613,588,562,536,50
9,482,454,426,397,368,339,309,279,249,218,187,156,125,94,63,31,0,-31,-63,-94,-125,-156,-
187,-218,-249,-279,-309,-339,-368,-397,-426,-454,-482,-509,-536,-562,-588,-613,-637,-661,-
685,-707,-729,-750,-771,-790,-809,-827,-844,-861,-876,-891,-905,-918,-930,-941,-951,-960,-
969,-976,-982,-988,-992,-996,-998,-1000,-1000,-1000,-998,-996,-992,-988,-982,-976,-969,-
960,-951,-941,-930,-918,-905,-891,-876,-861,-844,-827,-809,-790,-771,-750,-729,-707,-685,-
661,-637,-613,-588,-562,-536,-509,-482,-454,-426,-397,-368,-339,-309,-279,-249,-218,-187,-
156,-125,-94,-63,-31,0);
        shared variable saidat1 : integer:=0; --declaração variavel da portadora 1
        shared variable saidat2 : integer:=0; --declaração variavel da portadora 2
        shared variable l : integer :=0; --declaração variável l de contagem
        shared variable m : integer :=1; --declaração variável m de contagem
        shared variable N1: integer :=1; --declaração variável N1
        shared variable N2: integer :=1; --declaração variável N2
```

```

shared variable freqsen : integer :=50; --freq moduladora desejada pelo usuario
shared variable freqtri : integer :=5000;-- freq portadora desejada pelo usuario
    type triangular is array (integer range <>) of integer; --declaração 200 valores
                                amostrados para a função da portadora

    shared variable triangular1:triangular (1 to
200):=(10,20,30,40,50,60,70,80,90,100,110,120,130,140,150,160,170,180,190,200,210,220,
230,240,250,260,270,280,290,300,310,320,330,340,350,360,370,380,390,400,410,420,430,
440,450,460,470,480,490,500,510,520,530,540,550,560,570,580,590,600,610,620,630,640,
650,660,670,680,690,700,710,720,730,740,750,760,770,780,790,800,810,820,830,840,850,
860,870,880,890,900,910,920,930,940,950,960,970,980,990,1000,990,980,970,960,950,940,
930,920,910,900,890,880,870,860,850,840,830,820,810,800,790,780,770,760,750,740,730,
720,710,700,690,680,670,660,650,640,630,620,610,600,590,580,570,560,550,540,530,520,
510,500,490,480,470,460,450,440,430,420,410,400,390,380,370,360,350,340,330,320,310,
300,290,280,270,260,250,240,230,220,210,200,190,180,170,160,150,140,130,120,110,100,
90,80,70,60,50,40,30,20,10,0);

begin

    process(clk) --inicio processamento do clock
    begin --inicio do programa

        if clk='1' then --executa quando há subida do clock
            a:=1+a; --incrementa em 'a' uma unidade

            N1:=500000000/(freqsen*200); --cálculo de N1 (subidas de clock)
            N2:=500000000/(freqtri*200);      --cálculo de N2 (subidas de clock)

            if a=N1 then
                saida1 := senoide1(c1);--carrega na variavel saida1 elemento c1 do vetor
                c1:=1+c1; --incrementa em 'c1' uma unidade
                saida2 := senoide1(c2); --senoide2 defasada em 2*pi*66/200
                c2:=1+c2; --incrementa em 'c2' uma unidade
                saida3 := senoide1(c3); --senoide3 defasada em 2*pi*132/200
                c3:=1+c3; --incrementa em 'c3' uma unidade
                a:=0; -- zera variável 'a'

                if c1=200 then --retorno para o 0 quando c1=200
                    c1:=1;
                end if;

                if c2=200 then --retorno para o 0 quando c2=200
                    c2:=1;
                end if;

                if c3=200 then --retorno para o 0 quando c1=200
                    c3:=1;
                end if;
            end if;
        end if;
    end process;
end begin

```

```

end if;
l:=1+l; --incrementa uma unidade na variavel l

if l=N2 then
    saidat1 := triangular1(m);--carrega portadora positiva
    saidat2 :=-triangular1(m); --espelha portadora
    m:=1+m; incrementa uma unidade na variável m
    l:=0;

    if m=200 then --limita variável m em 200
        m:=1; --carrega valor inicial de m
    end if;

end if;

if saida1 > saidat1 then --comparação moduladora1 e portadora pos.
    d1<='1'; --chave d1 ativa
    e1<='0'; --chave e1 desativa
else
    d1<='0'; --chave d1 desativa
    e1<='1'; --chave e1 ativa
end if;

if saida1 > saidat2 then --comparação moduladora1 e portadora neg.
    f1<='1'; --chave f1 ativa
    g1<='0'; --chave g1 desativa
else
    f1<='0'; --chave f1 desativa
    g1<='1'; --chave g1 ativa
end if;

if saida2 > saidat1 then --comparação moduladora2 e portadora pos.
    d2<='1'; --chave d2 ativa
    e2<='0'; --chave e2 desativa
else
    d2<='0'; --chave d2 desativa
    e2<='1'; --chave e2 ativa
end if;

if saida2 > saidat2 then--comparação moduladora2 e portadora neg.
    f2<='1'; --chave f2 ativa
    g2<='0'; --chave g2 desativa
else

```

```

        f2<='0'; --chave f2 desativa
        g2<='1'; --chave g2 ativa
    end if;

    if saida3 > saidat1 then --comparaçao moduladora3 e portadora pos.
        d3<='1'; --chave d3 ativa
        e3<='0'; --chave e3 desativa
    else
        d3<='0'; --chave d3 desativa
        e3<='1'; --chave e3 ativa
    end if;

    if saida3 > saidat2 then --comparaçao moduladora3 e portadora neg.
        f3<='1'; --chave f3 ativa
        g3<='0'; --chave g3 desativa
    else
        f3<='0'; --chave f3 desativa
        g3<='1'; --chave g3 ativa
    end if;

end if;
end process; --fim processamento do clock

end architecture estrutura; --fim estrutura

```

A seguir é apresentado o código em VHDL para a modulação de um conversor que pode operar à cinco níveis e possui 8 chaves de comutação. Foram escolhidos 200 valores de amostragem:

```
library ieee;
use ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.ALL;

entity teste1 is --início da entidade
port (clk: in std_logic;      --declaração entrada de clock
      entr1,entr2,entr3,entr4,entr5,entr6,entr7,entr8,entr9,entr10,entr11,entr12,entr13,entr14,entr15,entr16: in bit; --declaração entradas p/ leitura de 16 bits
      switch: in bit; --declaração e entrada p/ escolha de aquisição da portadora
      pwm1,pwm2,pwm3,pwm4,pwm5,pwm6,pwm7,pwm8: out bit); --declaração saída das chaves

end entity teste1; --fim da entidade

architecture estrutura of teste1 is --início da estrutura

shared variable a : integer :=0;    --declaração da variável de contagem a
shared variable b : integer :=0;    --declaração da variável de contagem b
shared variable a1 : integer :=0;   --declaração da variável de contagem a1
shared variable a2 : integer :=0;   --declaração da variável de contagem a2
shared variable a3 : integer :=0;   --declaração da variável de contagem a3
shared variable a4 : integer :=0;   --declaração da variável de contagem a4
type x1 is array (integer range <>) of integer ; --declaração de vetor p/ entrada de bits
shared variable x : x1 (1 to 16) ; --declaração do vetor x p/ carregar valores que correspondem a cada bit. Ex.bit 3.- 2³. X(3)=8
shared variable var1,var2,var3,var4,var5,var6,var7,var8,var9: bit:='0';
shared variable count1 : integer :=1; --declaração da variável de contagem count1
shared variable count2 : integer :=1; --declaração da variável de contagem count2
type triangular is array (integer range <>) of integer; --declaração de 200 valores para vetor da portadora
shared variable triangular1:triangular (1 to 200):=(
175,350,525,700,875,1050,1225,1400,1575,1750,1925,2100,2275,2450,2625,2800,2975,3150,3325,3500,3675,3850,4025,4200,4375,4550,4725,4900,5075,5250,5425,5600,5775,5950,6125,6300,6475,6650,6825,7000,7175,7350,7525,7700,7875,8050,8225,8400,8575,8750,8925,9100,9275,9450,9625,9800,9975,10150,10325,10500,10675,10850,11025,11200,11375,11550,11725,11900,12075,12250,12425,12600,12775,12950,13125,13300,13475,13650,13825,14000,14175,14350,14525,14700,14875,15050,15225,15400,15575,15750,15925,16100,16275,16450,16625,16800,16975,17150,17325,17500,17675,17850,18025,18200,18375,18550,18725,18900,19075,19250,19425,19600,19775,19950,20125,20300,20475,20650,20825,21000,21175,21350,21525,21700,21875,22050,22225,22400,22575,22750,22925,23100,23275,23450,23625,23800,23975,24150,24325,24500,24675,24850,25025,25200,25375,25550,25725,25900,26075,26250,26425,26600,26775,26950,27125,27300,27475,27650,27825,28000,28175,28350,28525,28700,28875,29050,29225,29400,29575,29750,29925,30100,30275,30450,30625,30800,30975,31150,31325,31500,31675,31850,32025,32200,32375,32550,32725,32900,33075,33250,33425,33600,33775,33950,34125,34300,34475,34650,34825,35000,35175,35350,35525,35700,35875,36050,36225,36400,36575,36750,36925,37100,37275,37450,37625,37800,37975,38150,38325,38500,38675,38850,39025,39200,39375,39550,39725,39900,40075,40250,40425,40600,40775,40950,41125,41300,41475,41650,41825,42000,42175,42350,42525,42700,42875,43050,43225,43400,43575,43750,43925,44100,44275,44450,44625,44800,44975,45150,45325,45500,45675,45850,46025,46200,46375,46550,46725,46900,47075,47250,47425,47600,47775,47950,48125,48300,48475,48650,48825,49000,49175,49350,49525,49700,49875,50050,50225,50400,50575,50750,50925,51100,51275,51450,51625,51800,51975,52150,52325,52500,52675,52850,53025,53200,53375,53550,53725,53900,54075,54250,54425,54600,54775,54950,55125,55300,55475,55650,55825,56000,56175,56350,56525,56700,56875,57050,57225,57400,57575,57750,57925,58100,58275,58450,58625,58800,58975,59150,59325,59500,59675,59850,60025,60200,60375,60550,60725,60900,61075,61250,61425,61600,61775,61950,62125,62300,62475,62650,62825,63000,63175,63350,63525,63700,63875,64050,64225,64400,64575,64750,64925,65100,65275,65450,65625,65800,65975,66150,66325,66500,66675,66850,67025,67200,67375,67550,67725,67900,68075,68250,68425,68600,68775,68950,69125,69300,69475,69650,69825,70000,70175,70350,70525,70700,70875,71050,71225,71400,71575,71750,71925,72100,72275,72450,72625,72800,72975,73150,73325,73500,73675,73850,74025,74200,74375,74550,74725,74900,75075,75250,75425,75600,75775,75950,76125,76300,76475,76650,76825,77000,77175,77350,77525,77700,77875,78050,78225,78400,78575,78750,78925,79100,79275,79450,79625,79800,79975,80150,80325,80500,80675,80850,81025,81200,81375,81550,81725,81900,82075,82250,82425,82600,82775,82950,83125,83300,83475,83650,83825,84000,84175,84350,84525,84700,84875,85050,85225,85400,85575,85750,85925,86100,86275,86450,86625,86800,86975,87150,87325,87500,87675,87850,88025,88200,88375,88550,88725,88900,89075,89250,89425,89600,89775,89950,90125,90300,90475,90650,90825,91000,91175,91350,91525,91700,91875,92050,92225,92400,92575,92750,92925,93100,93275,93450,93625,93800,93975,94150,94325,94500,94675,94850,95025,95200,95375,95550,95725,95900,96075,96250,96425,96600,96775,96950,97125,97300,97475,97650,97825,98000,98175,98350,98525,98700,98875,99050,99225,99400,99575,99750,99925,100100,100275,100450,100625,100800,100975,101150,101325,101500,101675,101850,102025,102200,102375,102550,102725,102900,103075,103250,103425,103600,103775,103950,104125,104300,104475,104650,104825,105000,105175,105350,105525,105700,105875,106050,106225,106400,106575,106750,106925,107100,107275,107450,107625,107800,107975,108150,108325,108500,108675,108850,109025,109200,109375,109550,109725,109900,110075,110250,110425,110600,110775,110950,111125,111300,111475,111650,111825,112000,112175,112350,112525,112700,112875,113050,113225,113400,113575,113750,113925,114100,114275,114450,114625,114800,114975,115150,115325,115500,115675,115850,116025,116200,116375,116550,116725,116900,117075,117250,117425,117600,117775,117950,118125,118300,118475,118650,118825,119000,119175,119350,119525,119700,119875,120050,120225,120400,120575,120750,120925,121100,121275,121450,121625,121800,121975,122150,122325,122500,122675,122850,123025,123200,123375,123550,123725,123900,124075,124250,124425,124600,124775,124950,125125,125300,125475,125650,125825,126000,126175,126350,126525,126700,126875,127050,127225,127400,127575,127750,127925,128100,128275,128450,128625,128800,128975,129150,129325,129500,129675,129850,130025,130200,130375,130550,130725,130900,131075,131250,131425,131600,131775,131950,132125,132300,132475,132650,132825,133000,133175,133350,133525,133700,133875,134050,134225,134400,134575,134750,134925,135100,135275,135450,135625,135800,135975,136150,136325,136500,136675,136850,137025,137200,137375,137550,137725,137900,138075,138250,138425,138600,138775,138950,139125,139300,139475,139650,139825,140000,140175,140350,140525,140700,140875,141050,141225,141400,141575,141750,141925,142100,142275,142450,142625,142800,142975,143150,143325,143500,143675,143850,144025,144200,144375,144550,144725,144900,145075,145250,145425,145600,145775,145950,146125,146300,146475,146650,146825,147000,147175,147350,147525,147700,147875,148050,148225,148400,148575,148750,148925,149100,149275,149450,149625,149800,150000,150175,150350,150525,150700,150875,151050,151225,151400,151575,151750,151925,152100,152275,152450,152625,152800,152975,153150,153325,153500,153675,153850,154025,154200,154375,154550,154725,154900,155075,155250,155425,155600,155775,155950,156125,156300,156475,156650,156825,157000,157175,157350,157525,157700,157875,158050,158225,158400,158575,158750,158925,159100,159275,159450,159625,159800,160000,160175,160350,160525,160700,160875,161050,161225,161400,161575,161750,161925,162100,162275,162450,162625,162800,162975,163150,163325,163500,163675,163850,164025,164200,164375,164550,164725,164900,165075,165250,165425,165600,165775,165950,166125,166300,166475,166650,166825,167000,167175,167350,167525,167700,167875,168050,168225,168400,168575,168750,168925,169100,169275,169450,169625,169800,170000,170175,170350,170525,170700,170875,171050,171225,171400,171575,171750,171925,172100,172275,172450,172625,172800,172975,173150,173325,173500,173675,173850,174025,174200,174375,174550,174725,174900,175075,175250,175425,175600,175775,175950,176125,176300,176475,176650,176825,177000,177175,177350,177525,177700,177875,178050,178225,178400,178575,178750,178925,179100,179275,179450,179625,179800,180000,180175,180350,180525,180700,180875,181050,181225,181400,181575,181750,181925,182100,182275,182450,182625,182800,182975,183150,183325,183500,183675,183850,184025,184200,184375,184550,184725,184900,185075,185250,185425,185600,185775,185950,186125,186300,186475,186650,186825,187000,187175,187350,187525,187700,187875,188050,188225,188400,188575,188750,188925,189100,189275,189450,189625,189800,190000,190175,190350,190525,190700,190875,191050,191225,191400,191575,191750,191925,192100,192275,192450,192625,192800,192975,193150,193325,193500,193675,193850,194025,194200,194375,194550,194725,194900,195075,195250,195425,195600,195775,195950,196125,196300,196475,196650,196825,197000,197175,197350,197525,197700,197875,198050,198225,198400,198575,198750,198925,199100,199275,199450,199625,199800,200000);

end architecture estrutura;
```

```
12250,12075,11900,11725,11550,11375,11200,11025,10850,10675,10500,10325,10150,99
75,9800,9625,9450,9275,9100,8925,8750,8575,8400,8225,8050,7875,7700,7525,7350,7175
,7000,6825,6650,6475,6300,6125,5950,5775,5600,5425,5250,5075,4900,4725,4550,4375,4
200,4025,3850,3675,3500,3325,3150,2975,2800,2625,2450,2275,2100,1925,1750,1575,140
0,1225,1050,875,700,525,350,175,0,);
```

```
type senoide is array (integer range <>) of integer; --declaração de 200 valores para
vetor de referência
```

```
shared variable senoide1:senoide (1 to 200):=(
1029,2058,3084,4107,5126,6140,7148,8149,9142,10126,11100,12063,13014,13952,14876,
15786,16680,17558,18418,19261,20084,20887,21670,22431,23170,23887,24580,25248,25
892,26510,27102,27667,28205,28715,29196,29649,30073,30467,30831,31164,31467,3173
9,31979,32188,32365,32510,32623,32703,32752,32768,32752,32703,32623,32510,32365,3
2188,31979,31739,31467,31164,30831,30467,30073,29649,29197,28715,28205,27667,271
02,26510,25892,25248,24580,23887,23171,22431,21670,20887,20084,19261,18418,17558,
16680,15786,14876,13952,13014,12063,11100,10126,9142,8149,7148,6140,5126,4107,308
4,2058,1029,0,-1029,-2057,-3084,-4107,-5126,-6140,-7148,-8149,-9142,-10126,-11100,-
12063,-13014,-13952,-14876,-15786,-16680,-17558,-18418,-19260,-20084,-20887,-21670,-
22431,-23170,-23887,-24580,-25248,-25892,-26510,-27102,-27667,-28205,-28715,-29196,-
29649,-30073,-30467,-30831,-31164,-31467,-31739,-31979,-32188,-32365,-32510,-32623,-
32703,-32752,-32768,-32752,-32703,-32623,-32510,-32365,-32188,-31979,-31739,-31467,-
31164,-30831,-30467,-30073,-29649,-29197,-28715,-28205,-27667,-27102,-26510,-25892,-
25248,-24580,-23887,-23171,-22431,-21670,-20887,-20084,-19261,-18418,-17558,-16680,-
15786,-14877,-13952,-13014,-12063,-11100,-10126,-9142,-8149,-7148,-6140,-5126,-4107,-
3084,-2058,-1029,0);
```

```
shared variable saida : integer :=0; -- variável que carrega valores do vetor
```

```
shared variable saidat1 : integer:=0; --variável que carrega valores da portadora 1
```

```
shared variable saidat2 : integer:=0; --variável que carrega valores da portadora 2
```

```
shared variable saidat3 : integer:=0; --variável que carrega valores da portadora 3
```

```
shared variable saidat4 : integer:=0; --variável que carrega valores da portadora 4
```

```
shared variable freqsen : integer :=50; --freq. moduladora escolhida pelo usuario
```

```
shared variable freqtri : integer :=2000; --freq. portadora escolhida pelo usuario
```

```
shared variable N1: integer :=1; --declaração variavel N1 p/ contagem de subida do clk
```

```
shared variable N2: integer :=1; --declaração variável N2 p/ contagem de subida do clk
```

```
begin
```

```
    process(clk) --início do processo de clk
```

```
    begin
```

```
        if clk='1' then --
```

```
            a:=1+a;--incrementa uma unidade na variável a
```

```
            b:=1+b; --incrementa uma unidade na variável b
```

```
            N1:=50000000/(freqsen*200); --cálculo de N1
```

```
            N2:=50000000/(freqtri*200); --cálculo de N2
```

```
        if switch='1' then -- CHAVE OFF -LEITURA REFERENCIA INTERNA
```

```

        if a=N1 then --executa se a = n1 subidas de clk
saida := senoide1(count1); --carrega saída com vetor referencia
count1:=1+count1;--incrementa 1 na variável count 1
    if count1=200 then
        count1:=1;
    end if;
    a:=0;
end if;
else -- CHAVE ON -LEITURA REFERENCIA EXTERNA
    if entr1 ='1' then --LEITURA BIT1
        x(1):=1; --para entrada x(1), carrega '1' no primeiro elemento do vetor x
    else
        x(1):=0;
    end if;
    if entr2 ='1' then --LEITURA BIT2
x(2):=2; --para entrada x(2), carrega '2' no segundo elemento do vetor x
    else
        x(2):=0;
    end if;
    if entr3 ='1' then --LEITURA BIT3
x(3):=4; --para entrada x(3), carrega '4' no terceiro elemento do vetor x
    else
        x(3):=0;
    end if;
    if entr4 ='1' then --LEITURA BIT4
        x(4):=8;
    else
        x(4):=0;
    end if;
    if entr5 ='1' then --LEITURA BIT5
        x(5):=16;
    else
        x(5):=0;
    end if;
    if entr6 ='1' then --LEITURA BIT6
        x(6):=32;
    else
        x(6):=0;
    end if;
    if entr7 ='1' then --LEITURA BIT7
        x(7):=64;
    else
        x(7):=0;
    end if;
end if;

```

```

        end if;
    if entr8 ='1' then --LEITURA BIT8
        x(8):=128;
    else
        x(8):=0;
    end if;
    if entr9 ='1' then --LEITURA BIT9
        x(9):=256;
    else
        x(9):=0;
    end if;
    if entr10 ='1' then --LEITURA BIT10
        x(10):=512;
    else
        x(10):=0;
    end if;
    if entr11 ='1' then --LEITURA BIT11
        x(11):=1024;
    else
        x(11):=0;
    end if;
    if entr12 ='1' then --LEITURA BIT12
        x(12):=2048;
    else
        x(12):=0;
    end if;
    if entr13 ='1' then --LEITURA BIT13
        x(13):=4096;
    else
        x(13):=0;
    end if;
    if entr14 ='1' then --LEITURA BIT14
        x(14):=8192;
    else
        x(14):=0;
    end if;
    if entr15 ='1' then --LEITURA BIT15
        x(15):=16384;
    else
        x(15):=0;
    end if ;
    if entr16 ='1' then --LEITURA BIT16
        x(16):=32768;

```

```

        else
            x(16):=0;
        end if;

        saida := x(1)+x(2)+x(3)+x(4)+x(5)+x(6)+x(7)+x(8)+x(9)+x(10)+x(11)+
        x(12)+x(13)+x(14)+x(15)+x(16)+ (-32768); --soma de todos os elementos dos vetores de
        entrada .varia entre -32768 e +32768

    end if;

    if b=N2 then --carregamento das portadoras em N2 subidas de clock

        saidat1 := 17500 + triangular1(count2); --carrega e posiciona portadora1
        saidat2 := triangular1(count2); --carrega portadora2
        saidat3 := -17500 +triangular1(count2); --carrega e posiciona portadora3
        saidat4 := -35000 +triangular1(count2);-- carrega e posiciona portadora4
        count2:=1+count2;
        b:=0;

        if count2=200 then --limita count2 em 200
            count2:=1; --valor inicial
        end if;

    end if;

    if saida > 16384 then --executa se saida maior que ref. pos
        pwm1<='1'; --saída pwm1 seta
        var1:='1';
    else
        pwm1<='0';
        var1:='0';
    end if;

    if saida < -16384 then --executa se saida maior que ref. neg
        pwm3<='1';
        var2:='1';
    else
        pwm3<='0';
        var2:='0';
    end if;

    if saida > saidat2 then --executa se referência é maior que portadora 2
        var3:='1';
    else
        var3:='0';
    end if;

```

```

        if saida > saidat1 then
            var4:='1';
        else
            var4:='0';
        end if;

pwm4<= var4 XOR ( var1 NOR (NOT var3));
var9:= var4 XOR ( var1 NOR (NOT var3));
if saida > saidat4 then
    var5:='1';
else
    var5:='0';
end if;

if saida > saidat3 then
    var6:='1';
else
    var6:='0';
end if;

pwm2<= var5 XOR ( var2 NOR (NOT var6));
var8:= var5 XOR ( var2 NOR (NOT var6));

pwm7<= var8 nor var9;
pwm8<= var8 nor var9;

if saida > 0 then
    var7:='1';
else
    var7:='0';
end if;

    pwm5<=(var1 nor (not var7)) or (var2 nor var7);
    pwm6<=(var1 nor (not var7)) or (var2 nor var7);
end if;
end process; --fim do processo de clock
end architecture estrutura; --fim da estrutura

```

APÊNDICE C

A seguir é apresentado o código fonte para o caso de um conversor ponte completa monofásico operando com três níveis que usa a modulação PSD. São usados 200 valores amostrados para a portadora e a modulante.

```

library ieee;
use ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.ALL;

entity modulacao is          --declaracao da entidade (I/O de/para operações externas)
port (clk: in std_logic;          --declaração da entrada clock
      pwm1,pwm2,pwm3,pwm4: out bit); --declaração das variáveis de saída

end entity modulacao;          --fim da entidade

architecture estrutura of modulacao is --início da arquitetura

shared variable a : integer :=0; --declaração de variável p/contagem subidas do clock
shared variable b : integer :=0; --declaração de variável p/contagem subidas do clock

shared variable count1 : integer :=1; --declaração variável que aponta n do vetor
shared variable count2 : integer :=1; --declaração variável que aponta n do vetor
shared variable count2 : integer :=101; --declaração variável que aponta n do vetor
type triangular is array (integer range <>) of integer; -- declaração do vetor triangular
                                                    de 200 amostras entre -1000 e 1000

shared variable triangular1:triangular (1 to 200) :=
(20,40,60,80,100,120,140,160,180,200,220,240,260,280,300,320,340,360,380,400,420,440,4
60,480,500,520,540,560,580,600,620,640,660,680,700,720,740,760,780,800,820,840,860,88
0,900,920,940,960,980,1000,980,960,940,920,900,880,860,840,820,800,780,760,740,720,70
0,680,660,640,620,600,580,560,540,520,500,480,460,440,420,400,380,360,340,320,300,280
,260,240,220,200,180,160,140,120,100,80,60,40,20,0,-20,-40,-60,-80,-100,-120,-140,-160,-
180,-200,-220,-240,-260,-280,-300,-320,-340,-360,-380,-400,-420,-440,-460,-480,-500,-520,-
540,-560,-580,-600,-620,-640,-660,-680,-700,-720,-740,-760,-780,-800,-820,-840,-860,-880,-
900,-920,-940,-960,-980,-1000,-980,-960,-940,-920,-900,-880,-860,-840,-820,-800,-780,-
760,-740,-720,-700,-680,-660,-640,-620,-600,-580,-560,-540,-520,-500,-480,-460,-440,-420,-
400,-380,-360,-340,-320,-300,-280,-260,-240,-220,-200,-180,-160,-140,-120,-100,-80,-60,-
40,-20,0);

type senoide is array (integer range <>) of integer;          --declaração do vetor senoide 200
                                                    amostras entre -1000 e 1000. Índice de modulação de amplitude igual a 1
shared variable senoide1:senoide (1 to 200):=
(31,63,94,125,156,187,218,249,279,309,339,368,397,426,454,482,509,536,562,588,613,637
,661,685,707,729,750,771,790,809,827,844,861,876,891,905,918,930,941,951,960,969,976,
982,988,992,996,998,1000,1000,1000,998,996,992,988,982,976,969,960,951,941,930,918,9
05,891,876,861,844,827,809,790,771,750,729,707,685,661,637,613,588,562,536,509,482,4
54,426,397,368,339,309,279,249,218,187,156,125,94,63,31,0,-31,-63,-94,-125,-156,-187,-
218,-249,-279,-309,-339,-368,-397,-426,-454,-482,-509,-536,-562,-588,-613,-637,-661,-685,-
707,-729,-750,-771,-790,-809,-827,-844,-861,-876,-891,-905,-918,-930,-941,-951,-960,-969,-
976,-982,-988,-992,-996,-998,-1000,-1000,-1000,-998,-996,-992,-988,-982,-976,-969,-960,-

```

```
951,-941,-930,-918,-905,-891,-876,-861,-844,-827,-809,-790,-771,-750,-729,-707,-685,-661,-
637,-613,-588,-562,-536,-509,-482,-454,-426,-397,-368,-339,-309,-279,-249,-218,-187,-156,-
125,-94,-63,-31,0);
```

```
shared variable saida : integer :=0; --variável p/ carregamento de um vetor senoidal
shared variable saidat1 : integer:=0; --variável p/ carregamento de um vetor triangular
shared variable saidat2 : integer:=0; --variável p/ carregamento de um vetor triangular
shared variable freqsen : integer :=50; --frequência desejada do sinal modulante
shared variable freqtri : integer :=80000; --frequência desejada das portadoras
shared variable N1: integer :=1; --variável dependente da freq. do sinal modulante
shared variable N2: integer :=1; --variável dependente da freq. desejada da portadora
```

```
begin --início
```

```
process(clk) --início do processamento de clock
```

```
begin --início
```

```
if clk='1' then --executa em toda subida de clock
```

```
a:=1+a; --incrementa 1 na variável a
```

```
b:=1+b; --incrementa 1 na variável b
```

```
N1:=50000000/(freqsen*200); --cálculo de N1 (contagens de clock)
```

```
N2:=50000000/(freqtri*200); --cálculo de N2 (contagens de clock)
```

```
if a=N1 then --executa quando a for igual a N1 subidas de clock
```

```
saida := senoide1(count1); --carregamento vetor senoide na variável saída
```

```
count1:=1+count1; --incrementa 1 na variável count1
```

```
if count1=200 then --função para limitar count1 em 200
```

```
count1:=1; --quando count 1 atinge 200 assume valor 1
```

```
end if;
```

```
a:=0; --zera variavel a
```

```
end if;
```

```
if b=N2 then --executa quando a for igual a N2 subidas de clock
```

```
saidat1 := triangular1(count2); -- vetor triangular na variável saidat1
```

```
saidat2 := triangular1(count3); -- vetor triangular defasado na variável saidat2
```

```
count2:=1+count2; --incrementa 1 na variável count2
```

```
b:=0; --zera variável b
```

```
if count2<=100 then --executa se count2 for menor que 100
```

```
count3:=count2+100; --carrega count 3 com 100 valores à frente de count2
```

```
if count3=200 then --reinício de count3 quando atinge o limite
```

```
count3:=1; --valor inicial de count3
```

```
end if;
```

```
end if;
```

```
if (count2<200) AND (count2>100) then--executa quando count2 está entre 100 e 200
```

```
count3:=count3+1; --incrementa uma unidade em count3
```

```
end if;
```

```
if count2=200 then --reinício do número do vetor que será carregado
```

```

        count2:=1;      --início a partir do primeiro vetor
        count3:=101;    --count3 defasado em 100 valores
        end if;
    end if;
    if saida > saidat1 then      -- comparação entre os valores carregados da
                                onda senoidal e da primeira portadora

        pwm2<='0';      --reset pwm2
        pwm1<='1';      --set pwm1

    else

        pwm1<='0';      --reset pwm1
        pwm2<='1';      --set pwm2

    end if;

    if saida > saidat2 then      --comparação entre os valores carregados da onda senoidal
                                e da portadora defasada

        pwm4<='0';  --reset pwm4
        pwm3<='1';  --set pwm3

    else

        pwm3<='0';  --reset pwm3
        pwm4<='1';  --set pwm4

    end if;
end if;
end process; --fim do processo de leitura do clock
end architecture estrutura; --fim da estrutura

```