

UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM
ENGENHARIA ELÉTRICA

Lucas Salomon Carneiro

**Um circuito modulador de largura de pulso
(DPWM) utilizando FPGA**

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica como parte dos requisitos para obtenção do Título de Mestre em Ciências em Engenharia Elétrica.

Área de Concentração: Microeletrônica

Orientador: Prof. Dr. Robson Luiz Moreno

Co-orientador: Prof. Dr. Tales Cleber Pimenta

Dezembro de 2015
Itajubá – MG

Resumo

Este trabalho descreve o projeto de um modulador de largura de pulso (DPWM) digital de alta resolução implementado em FPGA. Foi desenvolvida uma arquitetura híbrida de um PWM digital de 17 bits. O sistema foi descrito utilizando a linguagem de descrição de *hardware* VHDL. No desenvolvimento do projeto, algumas descrições específicas foram realizadas para o acesso a determinados blocos da FPGA utilizada. Uma arquitetura alternativa de PWM foi desenvolvida com objetivo de atingir maior resolução. Porém, ela não atingiu as expectativas e foi documentada a fim de dar mais informações para trabalhos futuros. Os resultados incluem simulações e resultados experimentais, nos quais, foram medidas as menores variações de cada incremento do PWM.

Palavras-chave: DPWM, alta resolução, FPGA, modulação de largura de pulso, Cyclone IV®.

Abstract

This work describes a design of a high resolution digital pulse width modulator (DPWM) in FPGA. It was implemented in FPGA hybrid architecture of a 17-bit digital PWM. The system has been described in hardware description language VHDL. Some specific descriptions were developed to access to certain blocks of the used FPGA. An alternative PWM architecture was developed in order to achieve higher resolution, however, it did not met expectations and an annotation is provided for future work. It is included simulations and experimental results, which show the measurements of minor variation increments.

Keywords: DPWM, high-resolution, FPGA, pulse width modulation, Cyclone IV®.

Agradecimentos

Agradeço à minha família pelo suporte e apoio no desenvolvimento deste trabalho.

Ao meu orientador Robson Luiz Moreno que me apoiou e me auxiliou neste percurso.

Agradeço também ao grupo de microeletrônica por toda ajuda e suporte para a conclusão do projeto.

Por fim, agradeço à CAPES, ao CNPq e FAPEMIG que, através do suporte financeiro, viabilizaram a realização deste trabalho.

Sumário

1	Introdução.....	1
1.1	Motivação	1
1.2	Objetivos.....	2
1.3	Organização do trabalho	2
2	Fundamentação Teórica.....	4
2.1	Modulação por Largura de Pulso.....	4
2.2	Dispositivos Lógicos Programáveis.....	6
2.3	Linguagens – VHDL e Verilog.....	10
3	Arquiteturas DPWM.....	12
3.1	Contadores	12
3.2	Linhas de Atraso	14
3.3	Linhas de Atraso Segmentadas	15
3.4	Arquitetura Híbrida.....	16
3.5	Bloco de controle de Clock.....	18
3.5.1	Somadores	22
3.5.2	Carry Chain	22
4	DPWM desenvolvido	27
4.1	Arquitetura DPWM.....	27
4.1.1	Contador (CNT).....	28
4.1.2	Comparadores.....	29
4.1.3	Flip-Flops	30
4.1.4	Memória	30
4.1.5	Cadeia de Portas Lógicas Interconectadas – CPLI.....	32
4.1.6	Flip-flop de saída (FFOut).....	40
5	Resultados e Discussões	41
5.1	Simulação da arquitetura principal de 17 bits.....	41
5.2	Resultados experimentais	48
6	Trabalhos Futuros e conclusão	65
6.1	Arquitetura DPWM alternativa.....	65
6.2	Simulação da Arquitetura de 12 bits.....	68
6.3	Conclusões finais	70
7	Referências Bibliográficas.....	73

Lista de Figuras

Figura 1 - Ciclos ativos de um sinal PWM.....	5
Figura 2 - Diagrama básico de um PWM analógico [3].....	5
Figura 3 - Diagrama básico de um PWM digital [3].	6
Figura 4 - Estrutura básica de uma FPGA.....	7
Figura 5 - Circuito interno de um bloco lógico da Xilinx® (CLB).....	8
Figura 6 - Blocos da FPGA Cyclone IV® da Altera® mostrados no <i>floorplan</i> do software Quartus II.	8
Figura 7 - Localização dos LEs na FPGA Cyclone IV® mostradas no <i>floorplan</i> do software Quartus II da Altera®.	9
Figura 8 - Arquitetura clássica de um PWM digital baseado em contador [15].	13
Figura 9 - Formas de onda da arquitetura baseada em contador.	13
Figura 10: PWM baseado em linhas de atraso [17].....	14
Figura 11 - DPWM baseado em linhas de atraso segmentadas de 6 bits.	15
Figura 12 - Formas de onda da arquitetura híbrida.	17
Figura 13 - Diagrama do bloco DCM da Xilinx®.....	19
Figura 14 - Arquitetura DPWM proposta em [19].	20
Figura 15 - Arquitetura DPWM proposta em [20].	21
Figura 16 - Elemento Lógico do dispositivo Cyclone IV®.....	23
Figura 17 - Conexão entre LEs conectadas por <i>carry chains</i> [10].	23
Figura 18 - Arquitetura DPWM proposta por Lu-Sheng [21].	24
Figura 19 - Diagrama da estrutura Adder Delay Line [21].	25
Figura 20 - Roteamento utilizado entre a memória e os somadores em [22].	26
Figura 21 - Circuito DPWM.....	27
Figura 22 – Descrição do código do contador de nove bits.....	28
Figura 23 - Implementação das funções lógicas do comparador de nove bits.	29
Figura 24 - Código de descrição do comparador de nove bits.	30
Figura 25 - Descrição do código da memória para o DPWM de 17 bits.....	31
Figura 26 - Funcionamento da Cadeia de Portas Lógicas Interconectadas.	32
Figura 27 – (a) celula1 (b) celula2.....	33
Figura 28 - Descrição do componente celula1	34
Figura 29 - Acesso ao valor de uma <i>Lut_Mask</i> de um LE.	34
Figura 30 - Descrição do componente celula2.	36
Figura 31 - Conexão dos componentes celula1 e celula2 no <i>floorplan</i> do Quartus II®.	36
Figura 32 - Código da descrição do bloco CPLI	37
Figura 33 - Descrição do <i>clock</i> para o arquivo SDC.	38
Figura 34 - Criação do arquivo SDC.....	38
Figura 35 - Criação e unificação de partições.	39
Figura 36 - Flip-flop SR criado a partir de um flip-flop D.....	40
Figura 37 - Código da descrição do flip-flop de saída	40
Figura 38 – Simulação do sinal PWM para $DC_D = 0$	42
Figura 39 - Simulação do sinal PWM para $DC_D = 1$ e $DC_D = 2$	42
Figura 40 - Simulação do sinal PWM para $DC_D = 26366$	43
Figura 41 - Simulação do sinal PWM para $DC_D = 26367$ e $DC_D = 26368$	43
Figura 42 - Simulação do sinal PWM para $DC_D = 78846$	44
Figura 43 - Simulação do sinal PWM para $DC_D = 78847$ e $DC_D = 78848$	44

Figura 44 - Simulação do sinal PWM para $DC_D = 104958$	45
Figura 45 - Simulação do sinal PWM para $DC_D = 104959$ e $DC_D = 104960$	45
Figura 46 - Simulação do sinal PWM para $DC_D = 130778$	46
Figura 47 - Simulação do sinal PWM para $DC_D = 130779$ e $DC_D = 130780$	46
Figura 48 - 64 sinais PWM com variações consecutivas do ajuste principal.....	49
Figura 49 - Intervalos Δc de $DC_D = 256$ a $DC_D = 16384$	49
Figura 50 - 64 medidas do sinal PWM. Variações de $DC_D = 16384$ e $DC_D = 32768$..	50
Figura 51 - Intervalos Δc de $DC_D = 16384$ a $DC_D = 32768$	50
Figura 52 - 64 medidas do sinal PWM. Variações de $DC_D = 32768$ e $DC_D = 49152$..	51
Figura 53 - Intervalos Δc de $DC_D = 32768$ a $DC_D = 49152$	51
Figura 54 - 64 medidas do sinal PWM. Variações de $DC_D = 49152$ e $DC_D = 65536$..	52
Figura 55 - Intervalos Δc de $DC_D = 49152$ a $DC_D = 65536$	52
Figura 56 - 64 medidas do sinal PWM. Variações de $DC_D = 65536$ e $DC_D = 81920$..	53
Figura 57 - Intervalos Δc de $DC_D = 65536$ a $DC_D = 81920$	53
Figura 58 - 64 medidas do sinal PWM. Variações de $DC_D = 81920$ e $DC_D = 98304$..	54
Figura 59 - Intervalos Δc de $DC_D = 81920$ a $DC_D = 98304$	54
Figura 60 - 64 medidas do sinal PWM. Variações de $DC_D = 98304$ e $DC_D = 114688$..	55
Figura 61 - Intervalos Δc de $DC_D = 98304$ a $DC_D = 114688$	55
Figura 62 - 64 medidas do sinal PWM. Variações de $DC_D = 114688$ e $DC_D = 130560$	56
Figura 63 - Intervalos Δc de $DC_D = 114688$ e $DC_D = 130560$	56
Figura 64 - Primeiras 33 medidas de largura de pulso consecutivas - de $DC = 512$ a $DC = 544$	57
Figura 65 - Intervalos Δe de $DC_D = 512$ a $DC_D = 544$	57
Figura 66 - Intervalos Δe de $DC_D = 544$ a $DC_D = 576$	58
Figura 67 - Intervalos Δe de $DC_D = 576$ a $DC_D = 608$	58
Figura 68 - Intervalos Δe de $DC_D = 608$ a $DC_D = 640$	59
Figura 69 - Intervalos Δe de $DC_D = 640$ a $DC_D = 672$	59
Figura 70 - Intervalos Δe de $DC_D = 672$ a $DC_D = 704$	60
Figura 71 - Intervalos Δe de $DC_D = 704$ a $DC_D = 736$	60
Figura 72 - Intervalos Δe de $DC_D = 736$ a $DC_D = 768$	61
Figura 73 - Gráfico da regressão linear da largura de pulso pelo ciclo de trabalho.	61
Figura 74 - Ampliação do gráfico da regressão linear para os primeiros 74 elementos de atraso.....	62
Figura 75 - Ampliação do gráfico da regressão linear para 119 elementos de atraso. ...	62
Figura 76 - Ampliação do gráfico da regressão linear para os últimos elementos de atraso.....	63
Figura 77- Arquitetura DPWM.	66
Figura 78 - Funcionamento da CPLI da arquitetura 2.	67
Figura 79 - Simulação do sinal PWM para $DC_D = 2048$ e $DC_D = 2049$	68
Figura 80 - Simulação do sinal PWM para $DC_D = 2050$ e $DC_D = 2051$	68
Figura 81 - Simulação do sinal PWM para $DC_D = 2052$ e $DC_D = 2053$	69
Figura 82 - Simulação do sinal PWM para $DC_D = 2054$ e $DC_D = 2055$	69
Figura 83 - Síntese após a compilação no Quartus II®.	71

Simbologia

Símbolo	Significado	Unidade
t_H	<i>Tempo em que o sinal PWM permanece em nível lógico alto</i>	<i>s</i>
T	<i>Período</i>	<i>s</i>
f_S	<i>Frequência do oscilador</i>	<i>Hz</i>
T_{REF}	<i>Tensão de referência</i>	<i>V</i>
f_P	<i>Frequência da portadora</i>	<i>Hz</i>
$d[n]$	<i>Ciclo de trabalho de um PWM</i>	<i>-</i>
f_{CLK}	<i>Frequência do sinal de clock</i>	<i>Hz</i>
f_{SW}	<i>Frequência do sinal PWM</i>	<i>Hz</i>
ΔD	<i>Menor incremento de um sinal PWM</i>	<i>-</i>
ΔV_0	<i>Varição da tensão média do PWM</i>	<i>V</i>
V_{IN}	<i>Tensão de alimentação</i>	<i>V</i>
Δt	<i>Atraso controlado que determina a resolução das linhas de atraso</i>	<i>s</i>
N	<i>Número total de bits do ciclo de trabalho do PWM</i>	<i>-</i>
n	<i>Número de bits do ciclo de trabalho responsável pela resolução fina do PWM</i>	<i>-</i>
DC_D	<i>Ciclo de trabalho do PWM com valores decimais</i>	<i>-</i>
DC_B	<i>Ciclo de trabalho do PWM com valores binários</i>	<i>-</i>
Δe	<i>Menor incremento de resolução fina do sinal PWM</i>	<i>s</i>
Δc	<i>Menor incremento de resolução principal do sinal PWM</i>	<i>s</i>

Acrônimos e Abreviaturas

Símbolo	Significado
ASIC	<i>Application Specific Integrated Circuit</i>
BL	<i>Bloco Lógico</i>
CH	<i>Matriz de chaves</i>
CPLD	<i>Complex Programmable Logic Devices</i>
CPLI	<i>Cadeias de Portas Lógicas Interconectadas</i>
DC	<i>Duty Cycle</i>
DCM	<i>Digital Clock Manager</i>
DFS	<i>Digital Frequency Synthesizer</i>
DLL	<i>Delay-Locked Loop</i>
DoD	<i>Departament of Defense</i>
DPWM	<i>Digital Pulse Width Modulator</i>
EDA	<i>Eletronic Design Automation</i>
EFSM	<i>Extended Finite-State Machine</i>
E/S	<i>Entrada/Saída</i>
FPGA	<i>Field Programmable Gate Array</i>
IEEE	<i>Institute of Electrical and Electronics Engineers</i>
IP	<i>Intellectual Properties</i>
LAB	<i>Logic Array Block</i>
LE	<i>Logic Element</i>
LNLS	<i>Laboratório Nacional de Luz Síncrotron</i>
LUT	<i>Look-Up Tables</i>
MUX	<i>Multiplexador</i>
PLD	<i>Programmable logic device</i>
PLL	<i>Phase-Locked Loop</i>
PLI	<i>Programming Language Interface</i>
OS	<i>Phase Shift</i>
PWM	<i>Pulse Width Modulator</i>
RAM	<i>Random Access Memory</i>
RTL	<i>Register Transfer Level</i>
VHDL	<i>Very High Density Logic</i>
VHSIC	<i>Very High Speed Integrated Circuits</i>
WYSIWYG	<i>What You See Is What You Get</i>

1 Introdução

1.1 *Motivação*

O Grupo de Microeletrônica e o Grupo de Eletrônica de Pesquisa em Potência e Aplicações – GPEPA, ambos pertencentes a Universidade Federal de Itajubá, são parceiros no desenvolvimento de pesquisas na área de circuitos de controle para aplicações em eletrônica de potência.

Algumas atividades de interesse comum são o desenvolvimento de fontes de tensão e corrente, inversores de potência e circuitos de controles para aplicações em painéis solares. Essas podem se beneficiar da utilização de circuitos de modulação de largura de pulso digitais (DPWM) para permitir a construção de equipamentos que possibilitem o seu controle através de softwares embarcados em microcontroladores ou controle localizados em ambientes separados do local de utilização.

Uma aplicação de interesse da equipe é a utilização de circuitos DPWM no controle de fontes de corrente de potência para a utilização em aceleradores de partículas. O Laboratório Nacional de Luz Síncrotron (LNLS), Campinas – SP, está desenvolvendo uma nova fonte de luz síncrotron no Brasil com o projeto Sirius, que deve ser construído até meados de 2018. As fontes de luz síncrotron servem para a análise de matéria, importante para diversas aplicações como o desenvolvimento de novos remédios, processos menos poluentes, novas fontes de energia, descobertas de alimentos com propriedades mais nutritivas, entre outros.

Essas fontes de luz possuem um acelerador de partículas, que gera um tipo de radiação de intensa luminosidade chamada de luz síncrotron. Essa luz tem a capacidade de atravessar os materiais e revelar suas propriedades químicas e estruturais. A geração desse fenômeno ocorre quando os elétrons percorrem uma trajetória circular em um tubo fechado no vácuo e então são conduzidos a um anel de aceleração (*booster*). Esse acelerador faz com que os elétrons ganhem mais energia e são injetados no acelerador principal. No acelerador principal, os elétrons circulam quase na velocidade da luz e passam por ímãs que guiam sua trajetória. Existem algumas estruturas magnéticas responsáveis por mudar a direção dos elétrons. O ponto onde ocorre essa mudança faz com que os elétrons emitam a luz síncrotron. Essa luz é direcionada para as amostras em estudo, revelando informações sobre a estrutura do material.

Diversos equipamentos como bobinas corretoras de órbita, dipolos, quadrupolos e sextupolos, responsáveis pelo direcionamento dos elétrons, são alimentados por fontes de corrente. O controle das fontes será realizado em ambiente separado e utilizará softwares de acompanhamento das condições do acelerador, o que exige um controle digital das fontes de corrente. A opção escolhida envolve a utilização de circuitos de modulação da largura de pulso com implementação digital (DPWM) com resolução maior que 16 bits.

Essa foi a motivação para o estudo e desenvolvimento de um trabalho que possibilitasse o conhecimento de técnicas de projeto de circuitos DPWM.

O baixo volume de produção para a aplicação desejada não justifica o uso de circuitos integrados dedicados, o que motivou o desenvolvimento de um circuito PWM de alta resolução em FPGAs, uma vez que essa tecnologia permite uma facilidade de prototipação e construção de circuitos em pequena escala.

1.2 Objetivos

Este trabalho visa o conhecimento de técnicas de construção de circuitos digitais para controle da largura de pulso de sinais que controlam fontes de tensão ou correntes.

O conhecimento do uso de estruturas em dispositivos lógico programáveis (FPGAs) que permitam pequenos atrasos de propagação para sua utilização em circuitos mais complexos, também é um dos objetivos desse trabalho.

1.3 Organização do trabalho

Este trabalho está organizado em seis capítulos, sendo um capítulo de introdução, um de conclusão e os demais de desenvolvimento.

O Capítulo 2 descreve os aspectos teóricos básicos sobre linguagens de programação, dispositivos lógicos programáveis e técnicas de modulação de largura de pulso.

O Capítulo 3 apresenta uma revisão bibliográfica sobre circuitos DPWM.

O Capítulo 4 trata dos conceitos e considerações adotados no projeto do circuito proposto.

Os resultados obtidos após a implementação da arquitetura escolhida e sua programação em placa de desenvolvimento da Altera® denominada KIT DE2-115 são

abordados no Capítulo 5.

O Capítulo 6 apresenta as conclusões e as sugestões para trabalhos futuros que podem ser implementados a partir deste trabalho.

2 Fundamentação

Teórica

2.1 Modulação por Largura de Pulso

A Modulação por Largura de Pulso (PWM – *Pulse Width Modulation*) é uma técnica amplamente utilizada em sistemas de controle, robótica, acionamento de motores, controles de movimento, fontes chaveadas, inversores de frequência e outras aplicações [1]. A técnica consiste em variar a largura de pulso de um sinal (PWM) por meio da variação de um parâmetro. Esse parâmetro pode ser uma tensão de referência, no caso de um PWM analógico, ou um ciclo de trabalho (DC – *Duty Cycle*), no caso de um PWM digital (DPWM – *Digital Pulse Width Modulation*) [2]. O tempo em que o sinal PWM permanece em nível lógico alto (t_H) pode ser expresso em porcentagem. O ciclo ativo, tempo em que o sinal fica em nível alto, do PWM representa essa porcentagem e é calculado por:

$$\text{Ciclo ativo} = \frac{t_H}{T} \times 100\% \quad (1)$$

Em que T é o período do sinal PWM.

A Figura 1 apresenta diferentes ciclos ativos para um sinal PWM. Para um ciclo ativo de 50%, o tempo em que o sinal permanece em nível lógico alto é o mesmo em que o sinal permanece em nível lógico baixo. Para um ciclo ativo de 10%, o tempo em que o sinal permanece em nível lógico alto é de 10%. Para um ciclo ativo de 90%, o sinal permanece em nível lógico alto em 90% do período total.

O diagrama básico de um PWM analógico é mostrado na Figura 2 [3]. O sinal de saída do oscilador (f_S) ativa o sinal *set* (S) do *latch* SR [4], iniciando o pulso DPWM. A duração do pulso termina quando o sinal *reset* (R) é ativado. O sinal r é gerado após uma comparação do sinal p , geralmente triangular ou “dente de serra”, com uma tensão de referência (T_{REF}). A frequência da portadora (f_P) é igual a f_S . Enquanto T_{REF} for maior

que p , o sinal na saída do comparador permanece em nível baixo. Quando p ultrapassa o valor de T_{REF} , o comparador muda seu estado para nível alto.

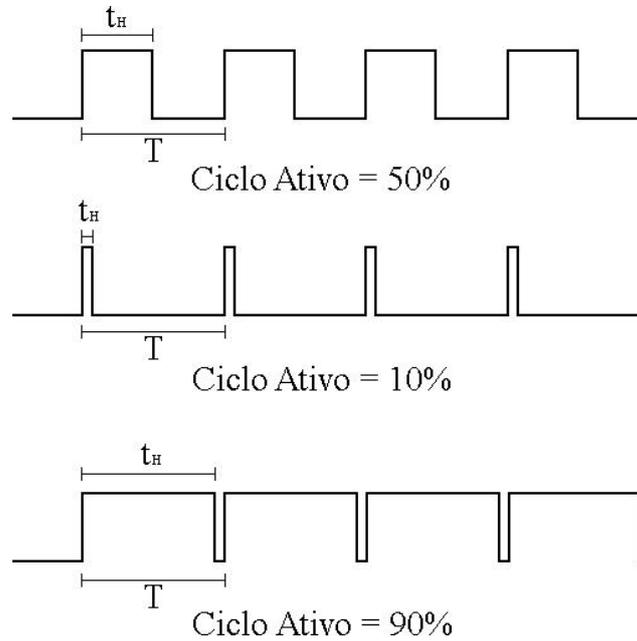


Figura 1 - Ciclos ativos de um sinal PWM.

A estrutura básica de um PWM digital é apresentada na Figura 3 [3]. A frequência de entrada f_s é discretizada pelo bloco Quantizador (contador digital), com 2^n valores, onde n é o número de células básicas (*bits*) utilizadas na implementação do Quantizador. O bloco “Comparador Digital” seleciona um desses valores, de acordo com o ciclo de trabalho $d[n]$ escolhido.

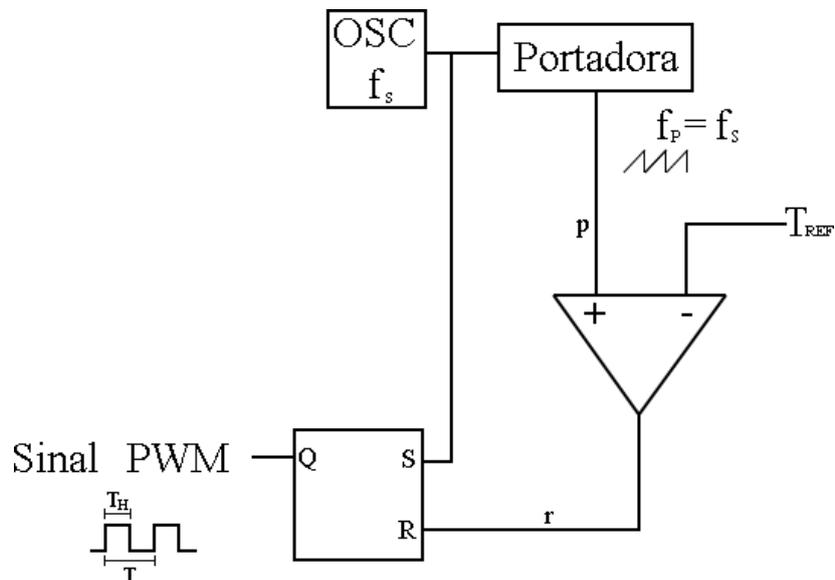


Figura 2 - Diagrama básico de um PWM analógico [3].

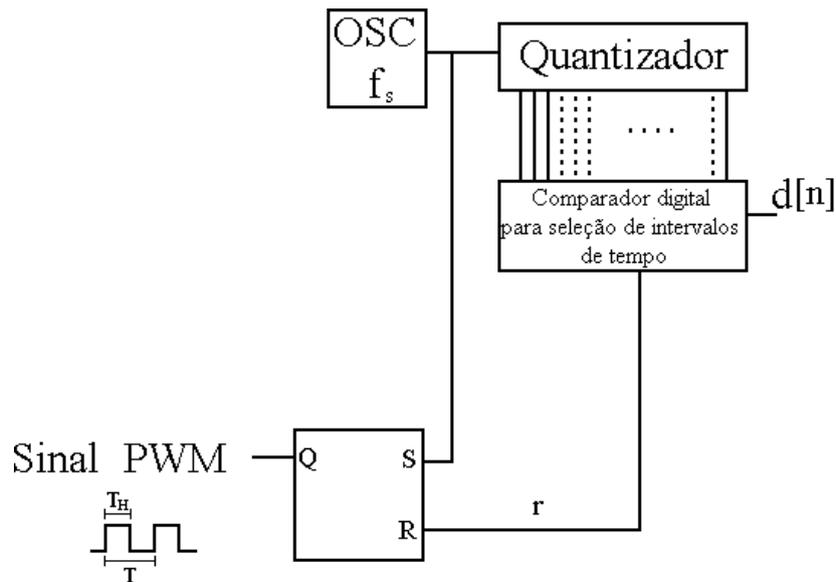


Figura 3 - Diagrama básico de um PWM digital [3].

2.2 Dispositivos Lógicos Programáveis

Os dispositivos lógicos programáveis (PLDs – *Programmable logic device*) são arquiteturas que combinam estruturas altamente programáveis, como o processador, com o alto desempenho e baixo consumo de circuitos integrados de aplicação específica (ASICs – *Application Specific Integrated Circuits*), além de serem reprogramáveis.

Das diversas arquiteturas PLDs existentes, destaca-se a FPGA (*Field Programmable Gate Array*), um dispositivo que possui em sua estrutura combinações lógicas LUTs (*Look-Up Tables*), flip-flops, blocos de memórias (RAMs), estruturas específicas para processamento de sinal (DSP), PLLs (*Phase-Locked Loop*), núcleo de processador e vários núcleos de comunicação [5].

A primeira FPGA surgiu em 1984, produzida pela empresa Xilinx® [6]. O dispositivo possuía 64 blocos lógicos e 58 blocos de entrada/saída [6]. Os dispositivos atuais possuem centenas de milhares de blocos lógicos e centenas de blocos de entrada/saída. O dispositivo Virtex-6®, da empresa Xilinx®, por exemplo, possui 760000 blocos lógicos e 1200 pinos de entrada/saída [7].

Enquanto um projeto em ASIC demora meses para ser fabricado e possui um alto custo, o qual pode chegar a ordem de milhão de dólares, FPGAs podem ser configuradas e reconfiguradas em segundos e possuem custo que varia de poucos dólares a alguns milhares de dólares [8]. Porém, a utilização de FPGAs tem algumas desvantagens

em relação aos ASICs. O dispositivo programável ocupa uma área maior na pastilha de silício, de 20 a 35 vezes, sua velocidade é de 3 a 4 vezes menor e a potência dinâmica dissipada é cerca de 10 vezes maior, se considerar a mesma implementação em ASIC. Apesar dessas desvantagens, as FPGAs são uma excelente alternativa para um desenvolvimento rápido de sistemas personalizados de alta velocidade com produção em baixa escala [8].

A estrutura básica de uma FPGA é apresentada na Figura 4. Essa estrutura é formada por blocos lógicos (BL), uma matriz de chaves (CH) de interconexão e blocos de entrada e saída (E/S). Os blocos lógicos são formados, basicamente, por flip-flops, estruturas para implementação de combinações lógicas e multiplexadores para configuração do bloco - Figura 5. Alguns blocos lógicos possuem funções mais específicas, como PLLs, memórias, DSPs, etc [4].

A Figura 6 ilustra os blocos lógicos específicos do dispositivo Cyclone IV® da Altera®, no qual, há quatro PLLs, localizadas nas quatro extremidades da FPGA, 529 pinos de entrada/saída, 114480 elementos lógicos (LE – *Logic Element*), 3981312 bits de memória e 532 elementos multiplicadores de 9 bits. Através de uma programação, as chaves de interconexão são configuradas e os blocos lógicos são conectados entre si, formando um sistema digital proposto pelo desenvolvedor [9].

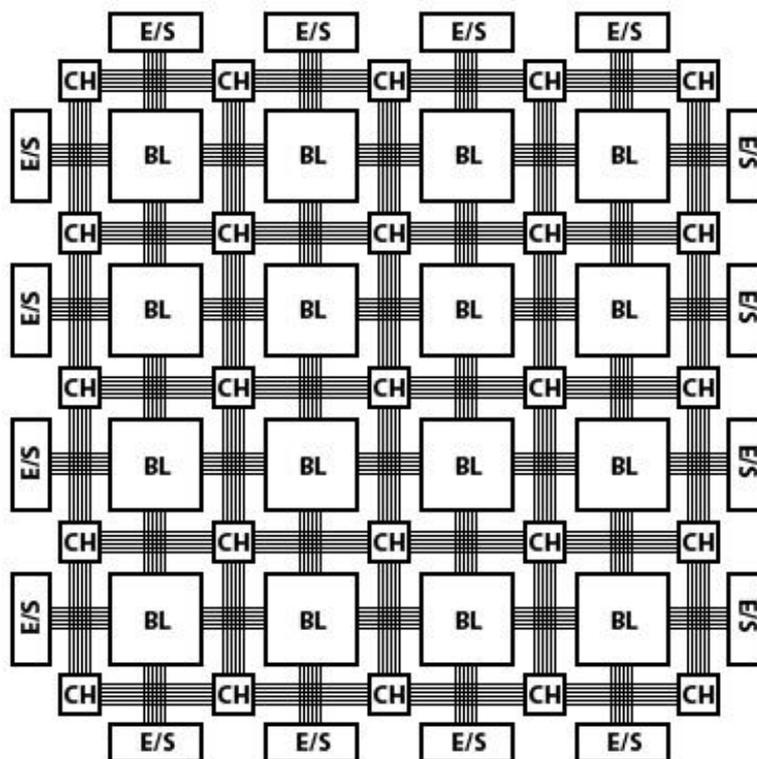


Figura 4 - Estrutura básica de uma FPGA.

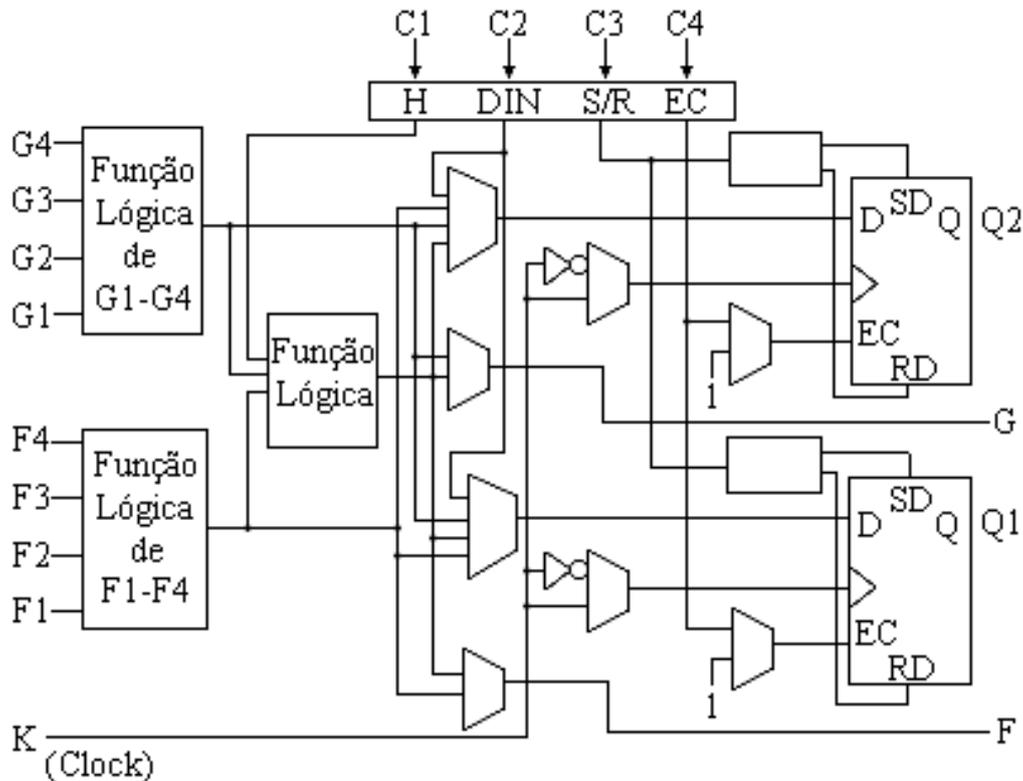


Figura 5 - Circuito interno de um bloco lógico da Xilinx® (CLB).



Figura 6 - Blocos da FPGA Cyclone IV® da Altera® mostrados no *floorplan* do software Quartus II.

Uma matriz de Blocos Lógicos (LAB – *Logic Array Block*) da FPGA Cyclone IV® contém 16 elementos lógicos. A Figura 7 mostra a localização dos elementos lógicos e seus respectivos registradores. Uma unidade da estrutura LAB é composta por uma pequena memória (LUT – *Look Up Table*) com a função de implementar lógica combinacional e, um registrador (flip-flop), como destacado na Figura 7 [10].

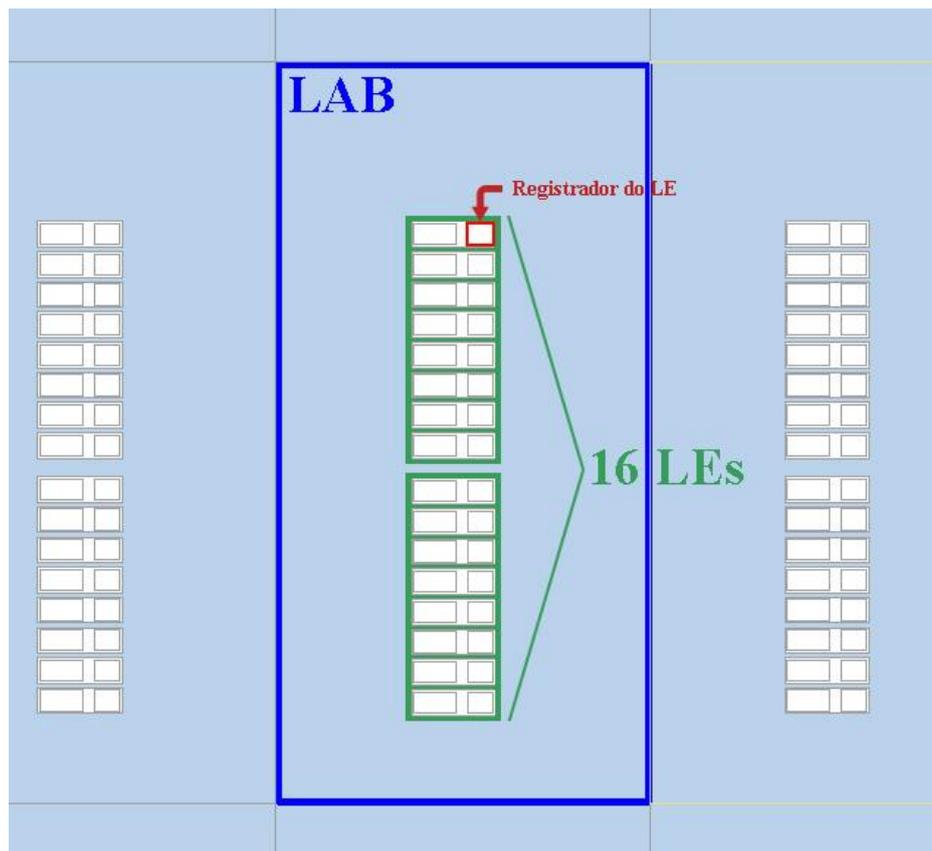


Figura 7 - Localização dos LEs na FPGA Cyclone IV® mostradas no *floorplan* do software Quartus II da Altera®.

Os dispositivos lógicos podem conter uma grande quantidade de transistores. Para que o projetista possa gerenciar esses transistores foram criados alguns níveis de abstração. Esses níveis tem o propósito de reduzir a quantidade de dados, permitindo a possibilidade de análise por parte do projetista. Nível alto de abstração significa que são apresentados apenas os dados mais relevantes do circuito. O nível baixo de abstração contém mais detalhes, apresenta uma estrutura complexa e mais próxima da implementação real do circuito.

Há quatro níveis de abstração [11]:

1. Nível Transistor – É o nível mais baixo. Os blocos básicos são os transistores, resistores, capacitores, etc. O circuito digital neste nível é considerado como um sistema analógico. A representação do sinal é dada pela tensão e o tempo é uma função contínua. A descrição comportamental é, geralmente, um conjunto de equações diferenciais e a descrição física é representada pelo transistor.
2. Nível Gate – Os blocos são constituídos por lógicas simples, como portas AND, OR, XOR, elementos básicos de memória, como *latch* e flip-flops

e multiplexadores simples. A representação do sinal é dada pelos níveis lógicos zero e um. O tempo é representado pelo atraso de propagação. A descrição comportamental é dada pelas equações booleanas e a descrição física é representada por células.

3. Nível RTL (*Register Transfer Level*) – Os blocos típicos deste nível são somadores, multiplexadores, registradores e comparadores. Os sinais são agrupados e interpretados de outro modo, como por exemplo, um sinal inteiro não sinalizado ou um estado. O tempo é baseado em pulsos de *clock*. A descrição comportamental utiliza Máquinas de Estado Estendida Finita (EFSM – *Extended Finite-State Machine*) e utiliza expressões gerais para especificar operações funcionais e roteamento de dados. A descrição física é conhecida como *floorplan* (Figura 6).
4. Nível de Processador – É o nível mais alto. Os blocos são conhecidos como IPs (*Intellectual Properties*), formados por processadores, módulos de memória, interfaces de barramento e outros. Os sinais são agrupados e interpretados de maneira mais ampla. O tempo é baseado em eventos sequenciais. A descrição comportamental é parecida com os códigos de linguagem de programação e a descrição física também é conhecida como *floorplan*, porém, os componentes utilizados são maiores que os de nível RTL.

2.3 Linguagens – VHDL e Verilog

VHDL (*Very High Density Logic*) ou VHSIC (*Very High Speed Integrated Circuits*) é uma linguagem de descrição de hardware que descreve a estrutura e o comportamento de um sistema eletrônico digital. Foi definida pelo Departamento de Defesa dos Estados Unidos (DoD) em 1980. Sua primeira versão, VHDL 87, foi a primeira linguagem padronizada pelo IEEE (*Institute of Electrical and Electronics Engineers*), através do padrão 1076. Em 1993 adotou-se um novo padrão, o 1164, no qual foi adicionado um sistema de multi-valores lógicos [12].

As principais aplicações da linguagem VHDL são em CPLD (*Complex Programmable Logic Devices*), ASICs e FPGA. Uma vez escrito o código, é possível implementar em dispositivos programáveis (Altera, Xilinx, Atmel, etc.) ou até mesmo

fabricar o chip em ASIC. As declarações em VHDL são chamadas de código, em vez de programa, pois a execução do código ocorre de forma paralela, ao contrário de outras linguagens que ocorrem de forma sequencial, uma vez que a linguagem implementa circuitos físicos. Somente funções declaradas dentro das estruturas PROCESS, FUNCTION ou PROCEDURE são executadas de forma sequencial [12].

A estrutura básica do código VHDL apresenta três unidades principais:

- LIBRARY: É composta por uma lista das bibliotecas utilizadas no código.
- ENTITY: Especifica os pinos de entrada e saída do projeto.
- ARCHITECTURE: Contém o código que descreve o comportamento do circuito.

Geralmente o circuito descrito em VHDL é dividido em instâncias, chamadas de componentes, que se conectam entre si em uma hierarquia principal. A importância dos componentes está na reutilização, particionamento e compartilhamento do código [12].

Outra linguagem bastante utilizada para a descrição de hardware é a linguagem Verilog. Tanto VHDL quanto Verilog podem ser modeladas com a mesma eficiência quando se implementa uma estrutura de hardware. Para implementações de hardware com nível de abstração mais elevado a capacidade da linguagem Verilog é um pouco menos eficiente, podendo ser igualada ao VHDL, em alguns casos, utilizando o mecanismo de interface PLI (*Programming Language Interface*). A escolha da linguagem não se baseia apenas na capacidade técnica. Outros fatores devem ser considerados, como preferência pessoal, disponibilidade da ferramenta EDA e questões comerciais, empresariais e de marketing [13]. Para este trabalho, adotou-se a linguagem VHDL por ser a mais utilizada no grupo de pesquisa da UNIFEI.

3 Arquiteturas DPWM

Diversas arquiteturas foram propostas em busca de um circuito PWM com baixa interferência a ruídos, alta resolução, baixo consumo de energia, redução de área, etc. Cada técnica apresenta vantagens e desvantagens de acordo com a aplicação desejada.

Visando uma prototipação rápida e a utilização em equipamentos com produção pequena, este estudo tem como objetivo escolher técnicas que possam ser implementadas em dispositivos lógicos programáveis (FPGA).

As principais técnicas de DPWM são baseadas em [14]:

- Contadores.
- Linhas de atraso.
- Linhas de atraso segmentadas.
- DLL (*Delay-Locked Loop*) e PLL (*Phase-Locked Loop*).
- Arquitetura híbrida.

Neste capítulo, são apresentadas as principais implementações dessas técnicas, com o objetivo de facilitar a compreensão da técnica escolhida para este trabalho.

3.1 Contadores

Utiliza uma arquitetura clássica [15], a qual compara o ciclo de trabalho do DPWM com um contador. Um *latch* SR inicia e finaliza o pulso PWM através dos sinais de *set* e *reset* - Figura 8.

O oscilador OSC, geralmente um gerador de sinal de *clock*, aciona o sinal *set* do *latch* SR, iniciando o pulso PWM. O contador digital, de frequência $f_{CLK} = 2^N \times f_S$, inicia a contagem e, ao se igualar ao valor do ciclo de trabalho $d(n)$, faz o Comparador ativar o sinal *reset* (SetR) do *latch*. As formas de onda para o exemplo da Figura 8, para $n = 2$, são mostradas na Figura 9. Para $d(2) = 10_b$, tem-se que o término do pulso no momento que o contador atinge o valor 2. Ao mudar o ciclo de trabalho para $d(2) = 11_b$, a duração do pulso PWM tem um acréscimo de um período de *clock*, terminando seu ciclo ativo quando o contador atinge o valor 3.

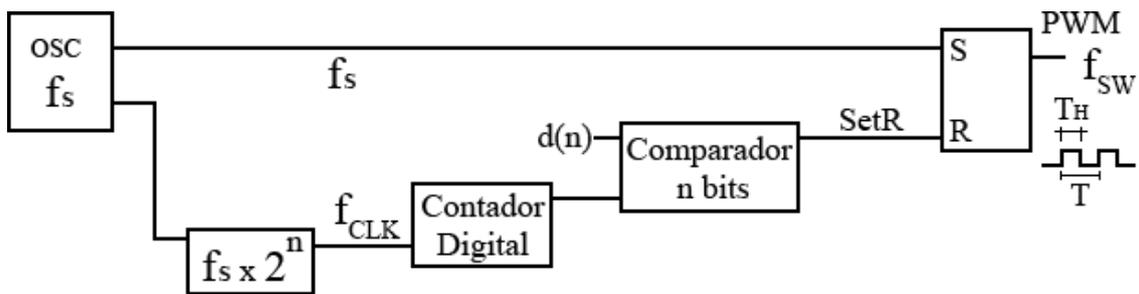


Figura 8 - Arquitetura clássica de um PWM digital baseado em contador [15].

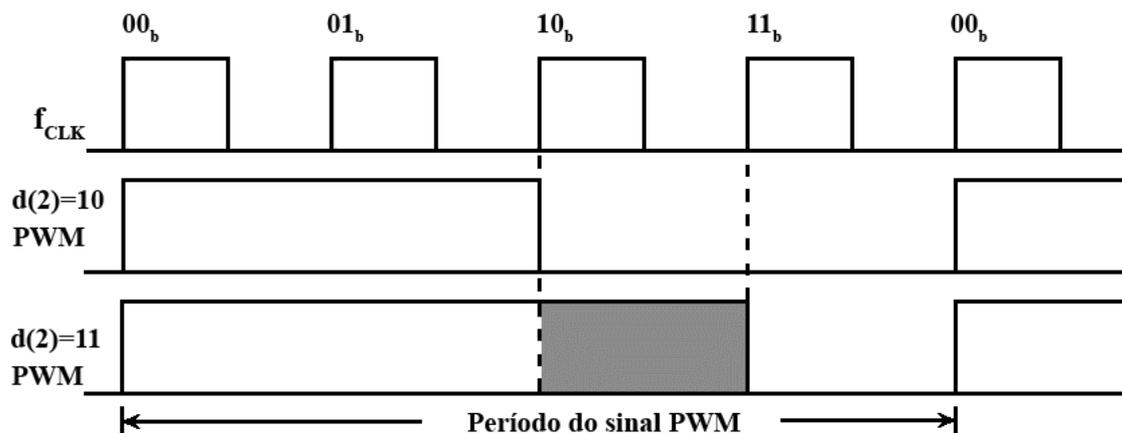


Figura 9 - Formas de onda da arquitetura baseada em contador.

A resolução do ciclo de trabalho ΔD , que determina o passo de um incremento no ciclo de trabalho, depende da frequência de *clock* f_{CLK} e da frequência de chaveamento (*Switching*) f_{SW} , sendo calculada pela expressão [16]:

$$\Delta D = \frac{f_{SW}}{f_{CLK}} = \frac{1}{2^n} \quad (1)$$

Na qual, n é a resolução expressa em bits. Essa resolução expressa o menor incremento dado no sinal PWM. Como exemplo, considerando uma frequência de um sinal PWM (f_{SW}) igual a 100KHz e uma resolução de 17 bits ($2^{17} = 131072$), seria necessária uma frequência de clock de 13,1 GHz, a qual é de difícil obtenção através das tecnologias normalmente disponíveis para implementação de circuitos digitais, o que praticamente inviabiliza a sua utilização.

Uma forma alternativa de expressar a resolução é utilizar denominação que relaciona o valor do menor incremento do ciclo de trabalho (ΔD) com o número de bits

equivalentes. Remanejando a equação (1) para expressar a resolução em número de bits, pode-se utilizar a seguinte relação:

$$n = \log_2 \left(\frac{1}{\Delta D} \right) \quad (2)$$

Obtendo a média do sinal PWM, para um valor de tensão, tem-se que a variação na saída é dada por Jian Li [16]:

$$\Delta V_O = V_{IN} \times \Delta D \quad (3)$$

Na qual, V_{IN} é a tensão de alimentação e ΔV_O é a variação da tensão média do sinal PWM.

Essa arquitetura apresenta uma excelente linearidade, porém, para alcançar resoluções altas é necessária uma frequência de entrada muito elevada, o que dificulta a sua implementação.

3.2 Linhas de Atraso

Esta arquitetura utiliza linhas de atraso combinadas com um multiplexador. Um exemplo de estrutura está representado na Figura 10. O sinal de *clock* ativa o *Set* do *latch* SR, inicializando o pulso PWM. O sinal que reinicializa o *latch* é o próprio sinal de *clock*, porém, com um determinado atraso (Δt). O valor desse atraso é determinado pelo ciclo de trabalho. Por exemplo, o primeiro incremento no ciclo de trabalho corresponde a um atraso Δt . O segundo incremento corresponde a dois Δt e assim por diante. A frequência do PWM é a mesma do sinal de temporização (*clock*) [17]. O bloco Rede de Atraso está presente no circuito para compensar os atrasos do Multiplexador.

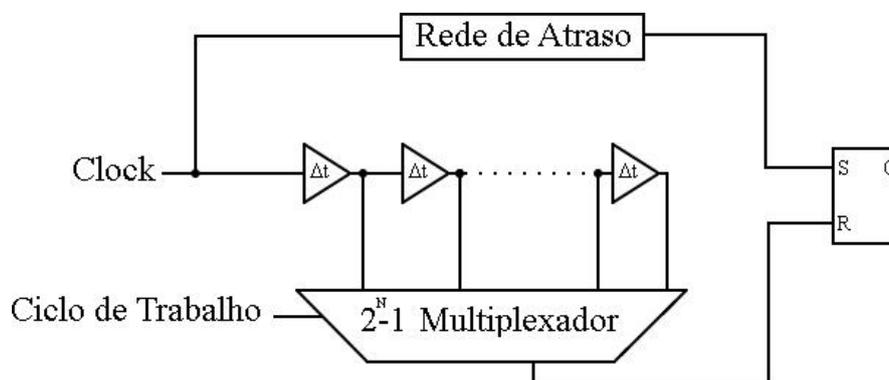


Figura 10: PWM baseado em linhas de atraso [17].

Essa arquitetura elimina a necessidade de um sinal de *clock* elevado, reduzindo o consumo de energia. Por outro lado, a área ocupada pelos elementos de atraso é maior, necessitando de 2^N elementos, no qual, N é o número de bits do PWM. Além disso, devido às 2^N linhas de entrada no multiplexador os atrasos no circuito não serão uniformes, afetando a linearidade do circuito.

O controle dos atrasos, nessa arquitetura, é de difícil execução em FPGAs e assim considerando a principal aplicação, não é indicada.

3.3 Linhas de Atraso Segmentadas

A técnica de Linhas de Atraso Segmentadas foi desenvolvida para diminuir a quantidade de elementos de atraso da arquitetura do item 3.2. Em vez de utilizar um único multiplexador, utilizam-se vários multiplexadores em cascata e cada um deles possui atrasos diferentes. Por exemplo, para um PWM de resolução de seis bits (Figura 11) há três multiplexadores de 4X1, em vez de um multiplexador 64X1. Cada multiplexador possui um peso diferente, o MUX 3 apresenta um atraso Δt_0 equivalente a uma célula, enquanto o MUX 2 e o MUX 1 apresentam atrasos Δt_1 e Δt_2 equivalentes a 4 e 16 células, respectivamente. A fórmula genérica para os atrasos em cada multiplexador é dada por Trescases [18]:

$$\Delta t_i = 4\Delta t_{i-1} = 2^{2^i} \Delta t_0 \quad (4)$$

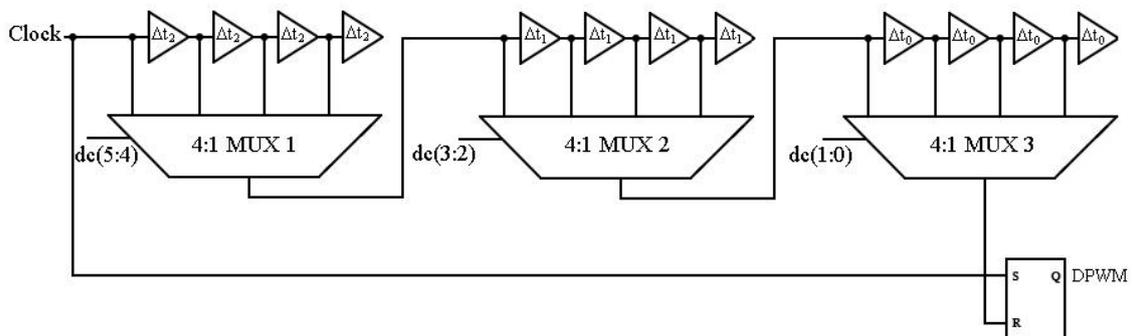


Figura 11 - DPWM baseado em linhas de atraso segmentadas de 6 bits.

O ciclo de trabalho (*dc – duty cycle*) é representado por $dc(5:0)$, em que $dc(5)$ é o bit mais significativo e $dc(0)$ o bit menos significativo. A Tabela 1 mostra os valores de atraso para os respectivos valores de dc .

Ciclo de Trabalho – $dc(5:0)$	Atrasos (Δt_i)
000001	Δt_0
000010	$2 \times \Delta t_0$
000011	$3 \times \Delta t_0$
000100	Δt_1
⋮	⋮
010001	$\Delta t_2 + \Delta t_0$
⋮	⋮

Tabela 1 - Atrasos de alguns ciclos de trabalho.

Essa técnica envolve atrasos com pesos diferentes e o roteamento deve ser cuidadosamente projetado para evitar atrasos não planejados. Devido a esses fatos, a implementação em FPGAs não é adequada.

3.4 Arquitetura Híbrida

A arquitetura híbrida é a junção de duas arquiteturas com resoluções baixas que dão origem a um circuito de alta resolução. Por apresentar uma boa linearidade, a arquitetura do item 3.1.1, baseada em contadores, é utilizada como resolução base. A outra arquitetura complementar é semelhante a do item 3.1.2, que utiliza uma rede de atrasos. Essa rede possui vários elementos de atraso que, no caso de implementações em FPGA, podem ser obtidos por atrasos específicos de algum bloco.

Esses elementos, descritos mais detalhadamente no decorrer do texto, podem ser obtidos com a inserção de um ou mais atrasos obtidos em estruturas normalmente disponíveis em FPGAs (tecnologia que permite flexibilização das implementações), tais como linhas de atraso em somadores.

A técnica também pode ser implementada com o uso de PLLs digitais, disponíveis em algumas FPGAs, visando o ajuste do sincronismo de contadores com objetivo de obter ajustes na largura do pulso de um sinal PWM.

A resolução de circuitos que implementam essa técnica pode ser dividida em duas componentes:

- **Resolução Principal:** É a resolução de um contador responsável pela definição das principais subdivisões do período do sinal PWM e seu menor incremento é dado pelo período do sinal de *clock*.
- **Resolução Fina:** É a resolução da estrutura responsável por ajustes realizados com o uso de atrasos de blocos pré-definidos em um dispositivo programável. Cada atraso pode definir o instante de término do pulso do sinal PWM em períodos menores do que o período do sinal de *clock*, sem, contudo, necessitar da implementação de osciladores de alta frequência, o que dificultaria a implementação. O menor incremento é obtido pelo menor elemento de atraso que pode ser acessado pelo software de gerenciamento da programação de uma FPGA.

As formas de onda das duas resoluções podem ser vistas na Figura 12. O Ajuste Principal é o incremento do contador, enquanto o Ajuste Fino (resolução de 2 bits neste exemplo – 4 possibilidades de atraso), é formado pelos elementos de atraso.

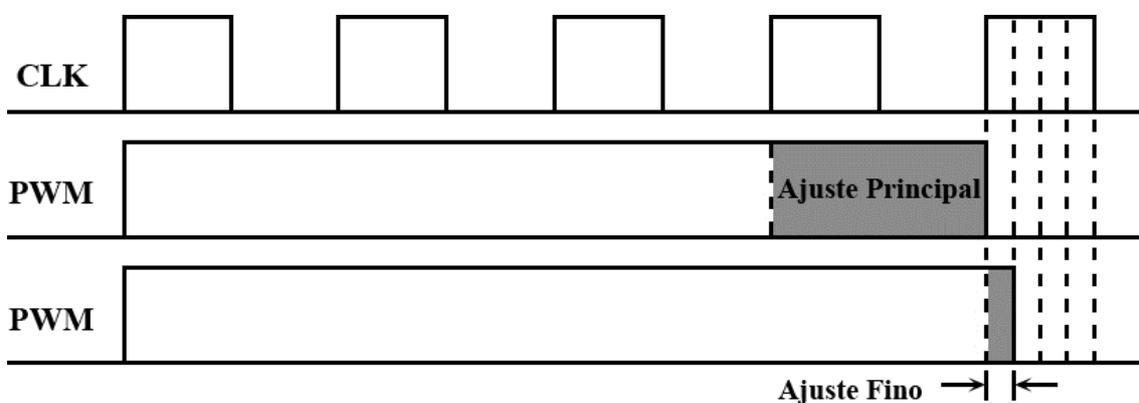


Figura 12 - Formas de onda da arquitetura híbrida.

Circuitos projetados em ASICs permitem que a estrutura híbrida seja implementada de forma precisa, pois há o controle total do roteamento do circuito. Mas a necessidade de adequação a variações de uso do circuito gerador de PWM durante a vida de um equipamento, seja para acrescentar funções extras ou mudar a característica do circuito, e o alto custo de produção dificulta o uso dessa tecnologia a um custo acessível a grande parte de usuários de circuitos PWM. O uso de FPGAs para esta

arquitetura torna dinâmico esse processo, permitindo ao projetista alterar o projeto conforme a necessidade. No entanto, a dificuldade de se implementar circuitos de atraso nas FPGAs, que possuem pouca flexibilidade de uso de algumas estruturas internas como linhas de atraso em somadores internos aos blocos lógicos, torna-se o desafio.

Diversos trabalhos de DPWMs [19], [20], [21] e [22] em FPGAs foram realizados com base na arquitetura híbrida e serão descritos a seguir. Foram utilizados como elementos de atraso nessas implementações somadores ou blocos de controle – circuitos que geram deslocamento de fase.

3.5 Bloco de controle de Clock

Diversas FPGAs possuem PLLs ou DLLs digitais responsáveis por gerar múltiplos sinais de *clock* a partir de um sinal de entrada, além da possibilidade de multiplicar, dividir ou deslocar a frequência de *clock* de entrada. Esse sistema está integrado em um bloco lógico, o qual dependendo do fabricante e modelo de FPGA pode estar disponibilizado em maior número.

As FPGAs da Xilinx® possuem um bloco desse sistema, o DCM (*Digital Clock Manager*), composto por um DLL (*Delay-Locked Loop*), um DFS (*Digital Frequency Synthesizer*) e um PS (*Phase Shift*). O diagrama do bloco DCM é apresentado na Figura 13. A quantidade de blocos varia de 4 a 12, dependendo do modelo da FPGA [23].

As FPGAs da Actel utilizam PLLs (*Phase-Locked Loops*) e possuem apenas 2 para os dispositivos APA075, APA150, APA300, APA450, APA600, APA750 e APA1000 [24].

Dispositivos da Altera® também utilizam PLLs. A família Cyclone® varia de 2 a 16 PLLs, dependendo do modelo.

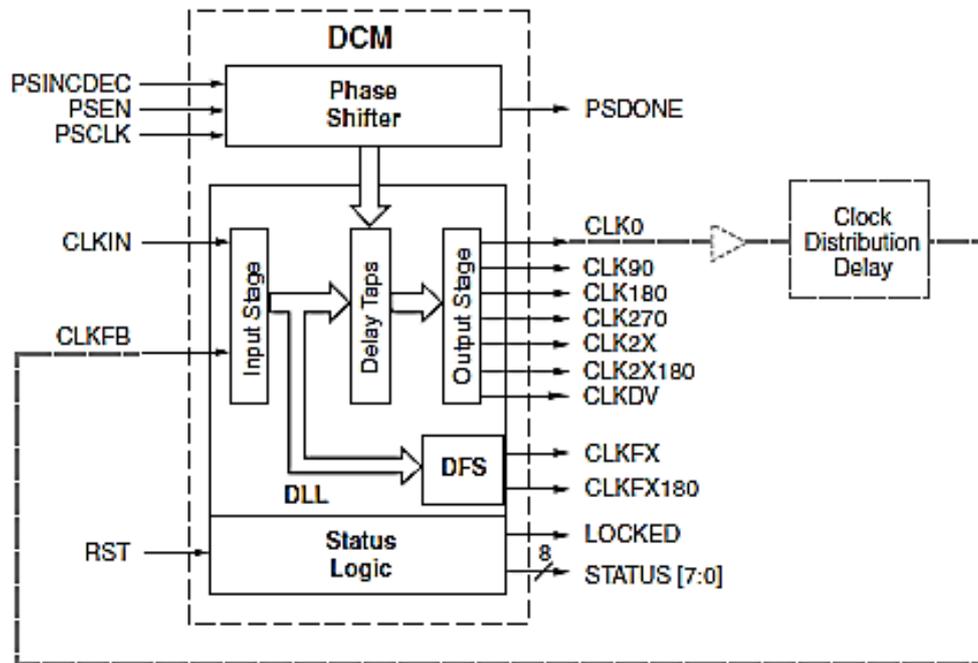


Figura 13 - Diagrama do bloco DCM da Xilinx®.

Esses blocos de controle deslocam a fase do sinal que alimenta o *reset* do *latch* de saída do DPWM, utilizando o mesmo princípio do diagrama da Figura 3.

Uma forma de implementar o bloco DLL é apresentado por Scharrer [19]. A Figura 14 apresenta a arquitetura proposta que utiliza, como linhas de atraso, um bloco DLL (bloco DCM da Xilinx®).

Considerando que o ciclo de trabalho possui um número total de bits (N_{Total} bits), os comparadores comparam os $(N_{Total} - 2)$ bits mais significativos do ciclo de trabalho com os bits do contador. Quando o contador é zerado, a estrutura *Coarse Pulse Reg* ativa o *Set* do *latch* SR, iniciando o pulso DPWM. No momento em que o contador é igual aos $(N_{Total} - 2)$ bits mais significativos do ciclo de trabalho, a estrutura *Coarse Pulse Reg* ativa o *Reset* do *latch* SR. Estes $(N_{Total} - 2)$ bits do ciclo de trabalho determinam a resolução principal. Os dois bits menos significativos do ciclo determinam a resolução fina.

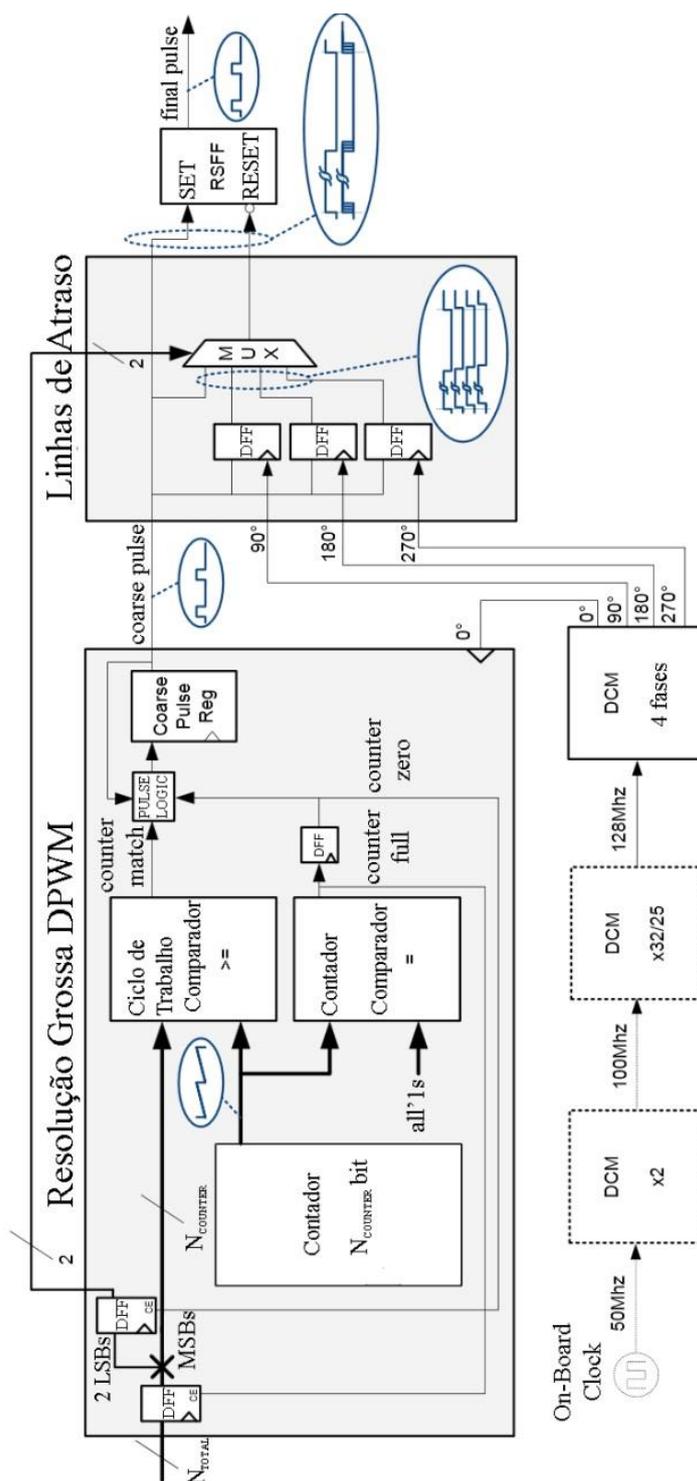


Figura 14 - Arquitetura DPWM proposta em [19].

O trabalho de Navarro [20] utiliza o mesmo princípio, mas utiliza um circuito síncrono multifásico. A Figura 15 ilustra a implementação de um DPWM de 11 bits, no qual 8 bits representam a resolução principal (contador) e 3 bits a resolução fina. Há três blocos DCM:

- DCMx4: Multiplica o sinal de *clock* de entrada por quatro e alimenta todas as estruturas que utilizam o *clock*.
- DCM0 e DCM1: Estão em paralelo e juntos formam uma estrutura que fornece 8 sinais de *clock* defasados.

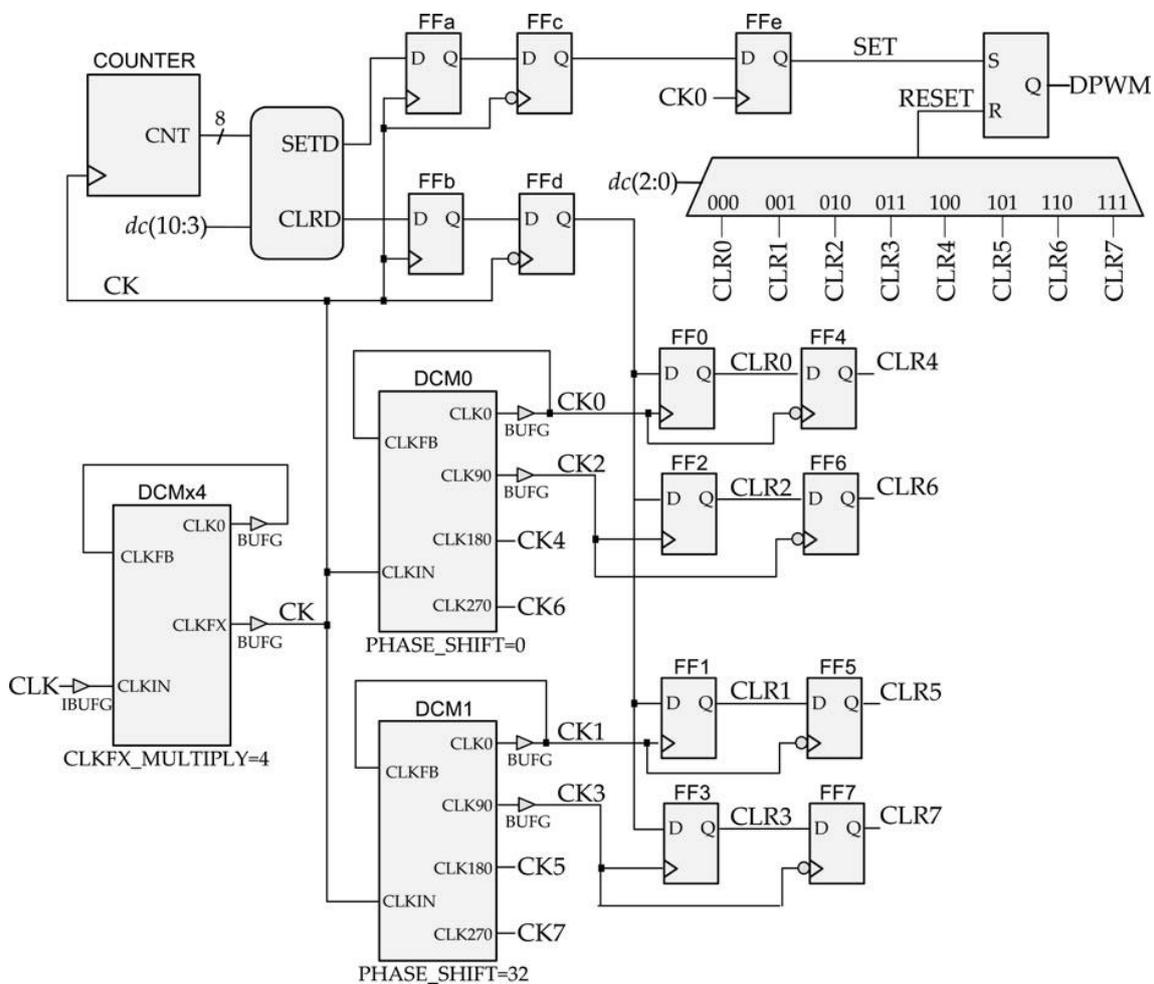


Figura 15 - Arquitetura DPWM proposta em [20].

Destaca-se a implementação de um circuito síncrono multifásico através dos flip-flops FF0 – FF7. A vantagem da utilização desses registradores é que a geração do sinal que ativa o *reset* do *latch* SR é síncrona e evita formação de *glitching*. FFa e FFb também evitam a presença de *glitches* formados pela saída do comparador.

A resolução alcançada foi, em média, de 625 ps, com 11 bits de resolução.

Diversas arquiteturas foram criadas com blocos de controle de *clock* presentes nas FPGAs. Elas possuem os mesmos conceitos das arquiteturas citadas, variando entre DCM, DLL e PLL. Os principais problemas desse tipo de arquitetura para atingir alta resolução são:

- A baixa quantidade de blocos de controle presentes nas FPGAs.
- A quantidade de defasagens que um bloco produz é limitada.
- Algumas FPGAs não permitem a interação de PLL/DLLs em conjunto visando aumentar a quantidade de deslocamentos do sinal.
- A distâncias entre os blocos de controle de *clock* podem apresentar atrasos significativos, dificultando ainda mais a implementação de resoluções altas.

3.5.1 Somadores

Alguns trabalhos utilizaram somadores como linhas de atraso para a criação de DPWMs de alta resolução. Somadores geralmente desfrutam de roteamentos extremamente curtos nas FPGAs, denominados *carry chains*, nas conexões entre os *carrys*.

3.5.2 Carry Chain

Nas FPGAs modernas estruturas que utilizam o modo aritmético aumentam o desempenho e a densidade lógica de determinados circuitos, tais como blocos multiplicadores DSP, somadores, contadores, acumuladores e comparadores. Esse modo permite que as conexões de *carry chains* estejam disponíveis para a construção de circuitos. Os *carry chains* conectam os elementos lógicos adjacentes sem a utilização de redes de conexão, reduzindo significativamente a propagação de atraso dessas ligações. Nas FPGAs da Xilinx®, os *carry chains* são utilizados nas portas *xor* e em multiplexadores programáveis. Porém, nas FPGAs da Altera® os *carry chains* são usados em somadores [25].

Para exemplificar essa conexão, a estrutura de um elemento lógico (LE) da Cyclone IV® está representada na Figura 16. As conexões LE *Carry-In* e LE *Carry-Out* fazem as interconexões entre outros LEs.

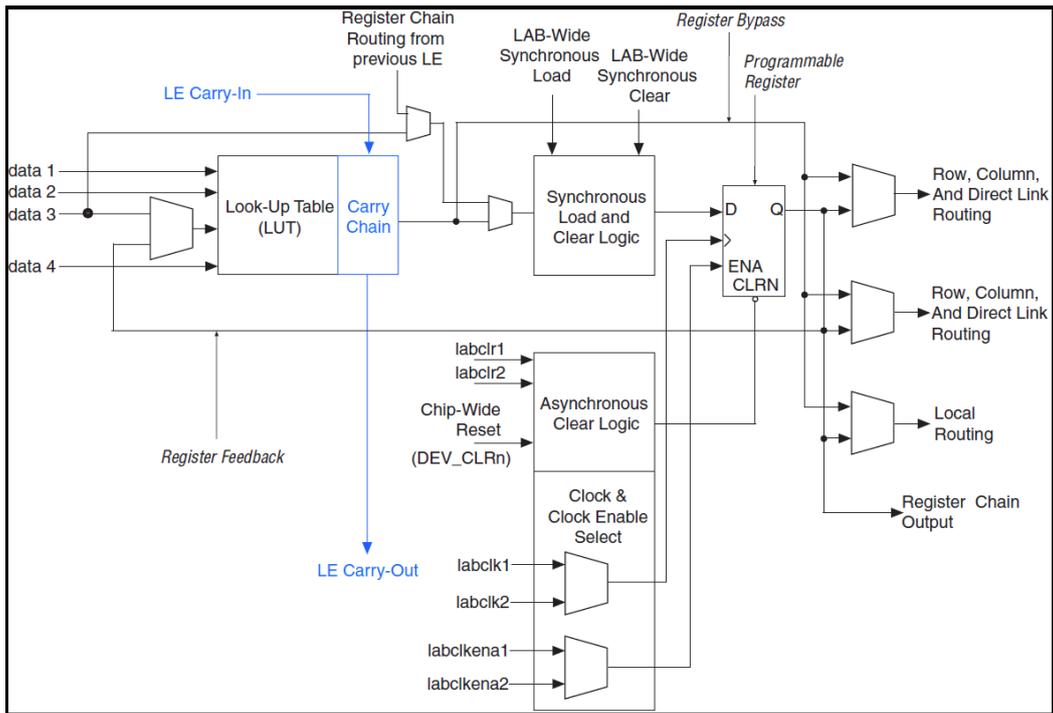


Figura 16 - Elemento Lógico do dispositivo Cyclone IV®.

A Figura 17 ilustra a conexão entre os LEs no *floorplan*. As setas em azul indicam o caminho percorrido pelos *carry chains* entre os elementos.

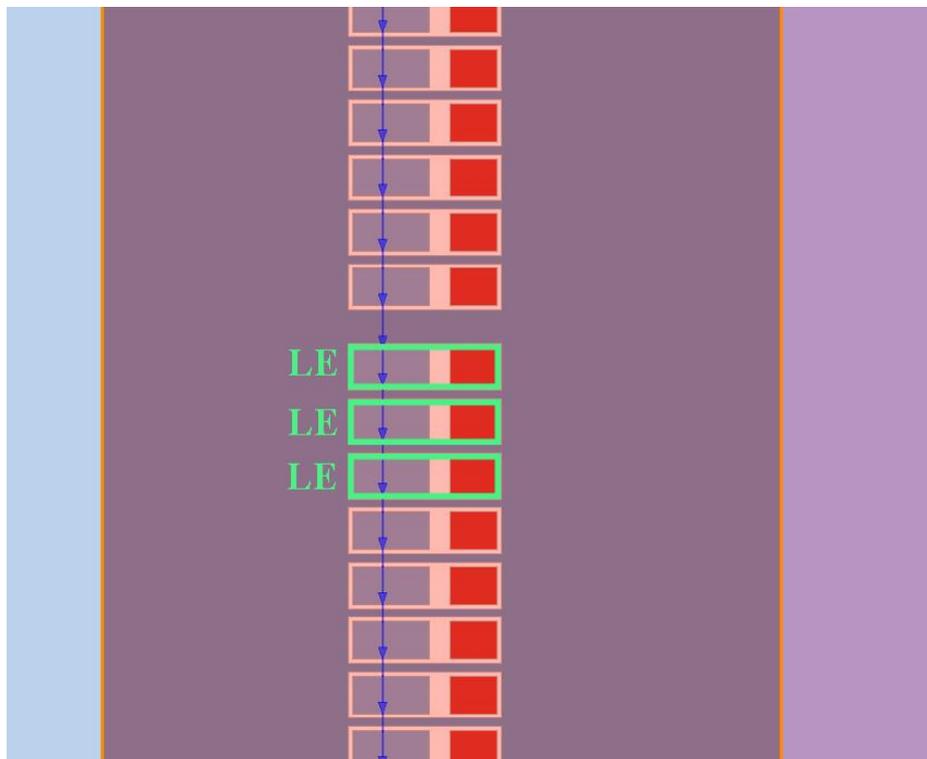


Figura 17 - Conexão entre LEs conectadas por *carry chains* [10].

3.5.2.1 Arquiteturas que utilizaram os carry chains

A arquitetura proposta por Lu-Sheng [21] está representada na Figura 18. O DPWM possui 15 bits de resolução. Os seis bits mais significativos são comparados com os contadores e formam a resolução principal. Os nove bits menos significativos estão conectados na estrutura *Adder Delay Line* e formam a resolução fina.

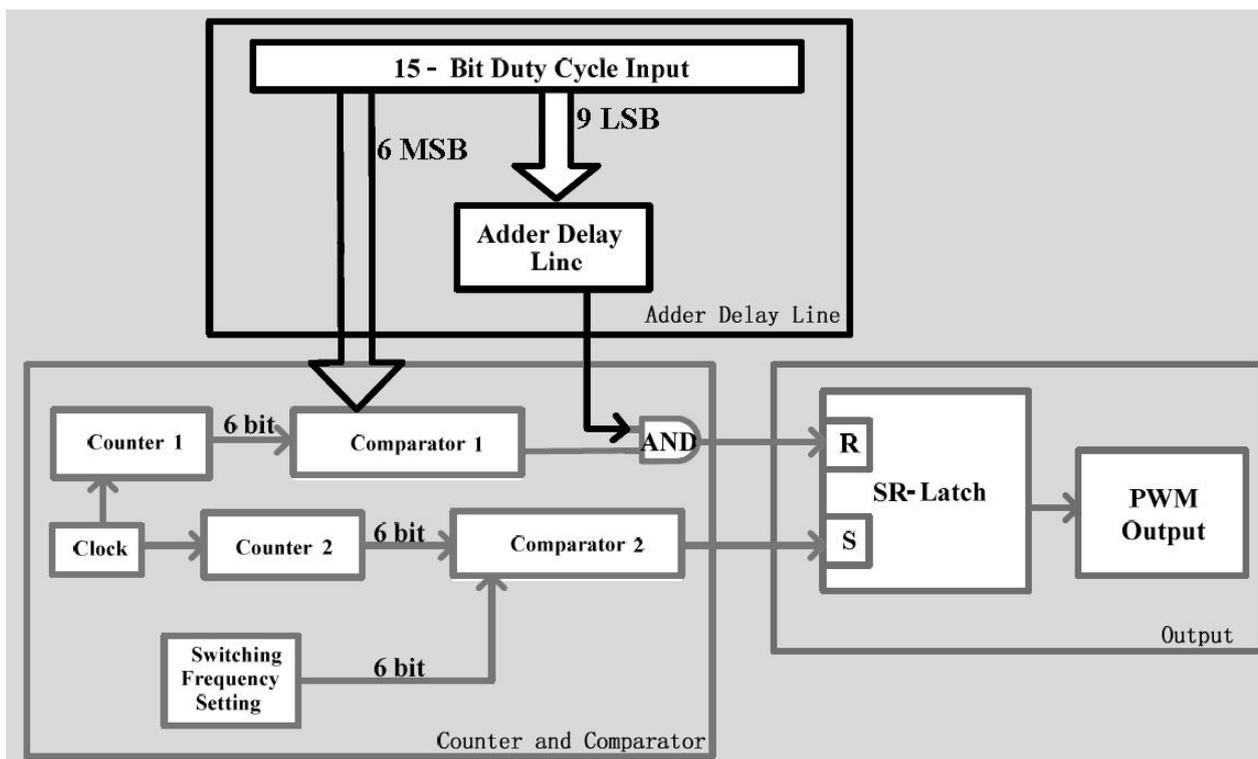


Figura 18 - Arquitetura DPWM proposta por Lu-Sheng [21].

A estrutura *Adder Delay Line* (Figura 19) possui uma cadeia de somadores de dois bits. Uma das entradas dos somadores está ligada a uma porta AND, a outra entrada é conectada em nível alto. O sinal *Start Signal* ativa ou desativa a estrutura e está ligado em uma das entradas da porta AND. A outra entrada é ligada ao *Shift Register*, uma estrutura que transmite uma sequência de bits para os somadores. Estes bits são selecionados pelos nove bits menos significativos do ciclo de trabalho e definem a dimensão do atraso. O atraso de cada somador é representado por τ . Na Figura 19 há um exemplo do funcionamento de uma estrutura *Adder Delay Line* de 3 bits com um atraso de três τ . Quando o sinal *Start Signal* inicia um pulso, o somador *Adder 4* realiza a soma e transmite o sinal para os somadores *Adder 5*, *Adder 6* e *Adder 7*. Este último gera o sinal que reinicia o flip-flop SR [21].

Alguns atrasos extras, devido ao roteamento automático da ferramenta de síntese, comprometem a linearidade do circuito. Por isso, o roteamento manual é imprescindível nesse tipo de arquitetura [21].

A resolução alcançada foi de 70 ~ 80 ps, com uma frequência DPWM de 1 MHz e ciclo de trabalho de 15 bits.

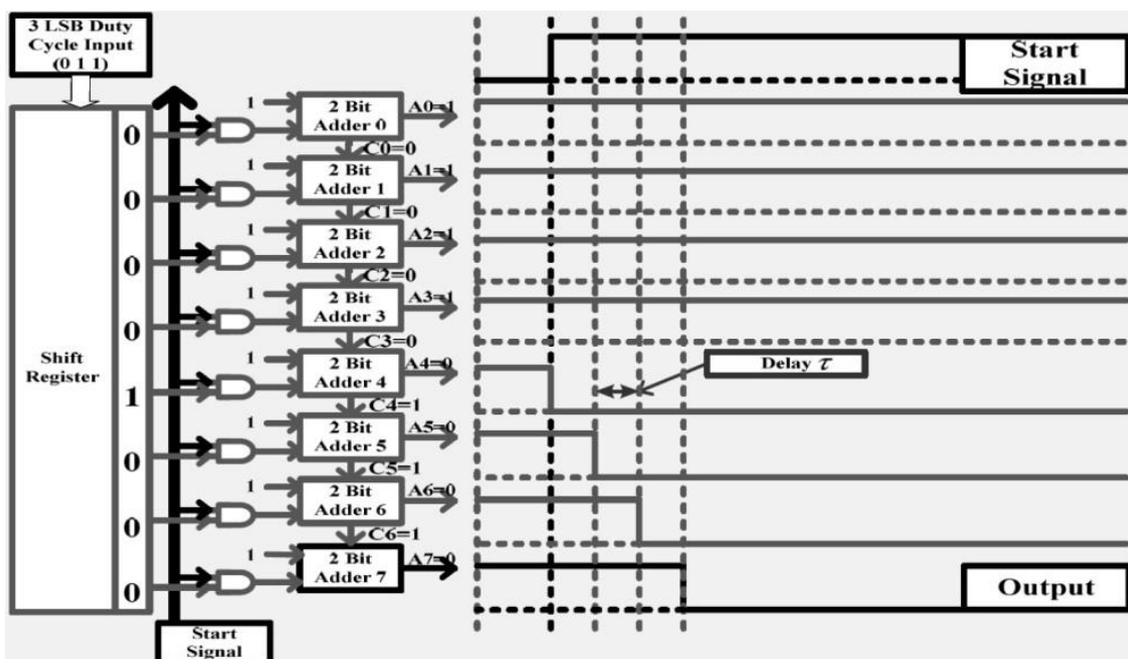


Figura 19 - Diagrama da estrutura Adder Delay Line [21].

A proposta do Costinett [22] tem como principal solução evitar a necessidade do roteamento manual no trabalho de Lu-Sheng [21]. Na arquitetura das linhas de atraso, os somadores são conectados diretamente aos blocos de memória e utilizam um roteamento específico para garantir a linearidade do circuito, eliminando a utilização de multiplexadores. A Figura 20 ilustra o roteamento específico entre um bloco de memória e 16 somadores. Para que esse roteamento seja automático é necessário alterar a opção *set_location_assignment* na ferramenta Quartus II® da Altera® [22].

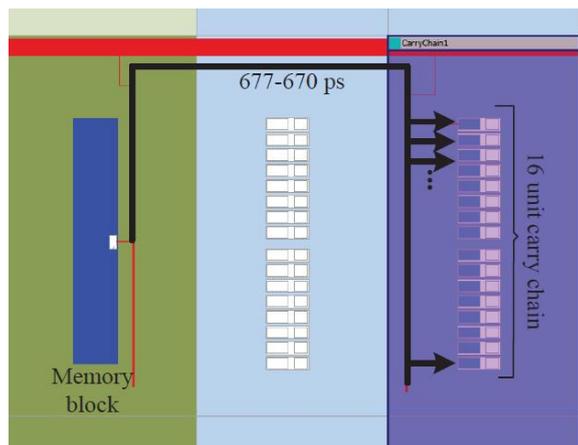


Figura 20 - Roteamento utilizado entre a memória e os somadores em [22].

Foram mostradas algumas arquiteturas implementadas em FPGA. As baseadas somente em contadores não são mais utilizadas pois exigem um frequência muito alta. Porém, desempenham um papel importante nas arquiteturas híbridas. As arquiteturas de linhas de atraso exigem uma grande quantidade de elementos, tornando a área ocupada muito extensa. Além disso, grandes quantidades de elementos geram atrasos nos sinais. As linhas de atraso segmentadas diminuem a quantidade de elementos, porém, exigem um controle mais rígido no roteamento, uma vez que é composto por atrasos diferenciados e alguns multiplexadores, o que inviabiliza a implementação em FPGA. Na literatura, a arquitetura híbrida foi bastante explorada e obteve destaque em relação as outras pois se beneficia da vantagem da arquitetura do contador com a vantagem das linhas de atraso ou DLLs.

A implementação da arquitetura híbrida em FPGAs, basicamente, pode ser dividida em duas categorias: elementos de atraso utilizando uma estrutura síncrona [26], [20], e [27] e elementos de atraso que utilizam estruturas assíncronas [21] [22]. Para a estrutura síncrona é utilizado blocos de controle que defasam o sinal de *clock*, funcionando como elementos de atraso. Por outro lado, a estrutura assíncrona utiliza qualquer elemento que gere um atraso.

O presente trabalho utilizou uma arquitetura que tem como base estruturas assíncronas que fazem o uso de *carry chains* como linhas de atraso. A justificativa para essa escolha são as limitações dos blocos DLL/PLL. Esses blocos possuem alguns problemas que dificultam atingir resoluções mais altas, como a quantidade limitada de blocos de controle e a limitação de conexões entre esses blocos.

o valor zero do contador. Para a definição do ciclo ativo, o comparador CMP2 compara o valor do ciclo de trabalho com o contador. Essa comparação ativa a rede de atrasos CPLI (Cadeia de Portas Lógicas Interconectadas), reiniciando o flip-flop FFOut.

O funcionamento de cada bloco do circuito é descrito com mais detalhes a seguir, para arquitetura de 17 bits.

4.1.1 Contador (CNT)

É utilizado um contador síncrono de 9 bits. A contagem é contínua e é iniciada após a reinicialização (*reset*) do circuito ou quando atinge seu valor máximo.

O software Quartus II v. 15.0.0® possui uma ferramenta (*IP Catalog*) que gera blocos com determinadas funções. Esta ferramenta substituiu a ferramenta *MegaWizard Plug-In Manager* em versões anteriores do software. O contador foi desenvolvido utilizando o *IP catalog* para garantir a utilização de *carry chains* em sua estrutura, assegurando que os atrasos sejam os menores possíveis na contagem. Porém, uma descrição normal em VHDL é suficiente para o funcionamento do circuito. O contador de nove bits e uma parte do código está representada na Figura 22. O componente utilizado pertence à biblioteca *lpm*.

```

COMPONENT lpm_counter
  GENERIC (
    lpm_direction      : STRING;
    lpm_port_updown    : STRING;
    lpm_type           : STRING;
    lpm_width          : NATURAL);

  PORT (
    clock : IN STD_LOGIC;
    sclr  : IN STD_LOGIC;
    q     : OUT STD_LOGIC_VECTOR (8 DOWNTO 0));
END COMPONENT;

BEGIN
  C    <= sub_wire0(8 DOWNTO 0);

  LPM_COUNTER_component : LPM_COUNTER
  GENERIC MAP (
    lpm_direction      => "UP",
    lpm_port_updown    => "PORT_UNUSED",
    lpm_type           => "LPM_COUNTER",
    lpm_width          => 9)
  PORT MAP (
    clock => clk,
    sclr  => rst,
    q     => sub_wire0);

```

Figura 22 – Descrição do código do contador de nove bits.

4.1.2 Comparadores

Dois comparadores são utilizados no circuito (Figura 21). O comparador CMP1 compara os bits do contador com o valor zero (início da contagem). Ao atingir esse valor, CMP1 libera um pulso em sua saída. O comparador CMP2 compara os 9 bits mais significativos do ciclo de trabalho (DC – *Duty Cycle*) com os bits do contador e libera um pulso em sua saída quando os dois sinais são iguais.

Os pulsos gerados pelos comparadores possuem a função de reiniciar (CMP2) e iniciar (CMP1) o pulso do DPWM.

Os comparadores desenvolvidos possuem nove bits e foram implementados utilizando as funções lógicas XNOR e AND, como pode ser visto na Figura 23.

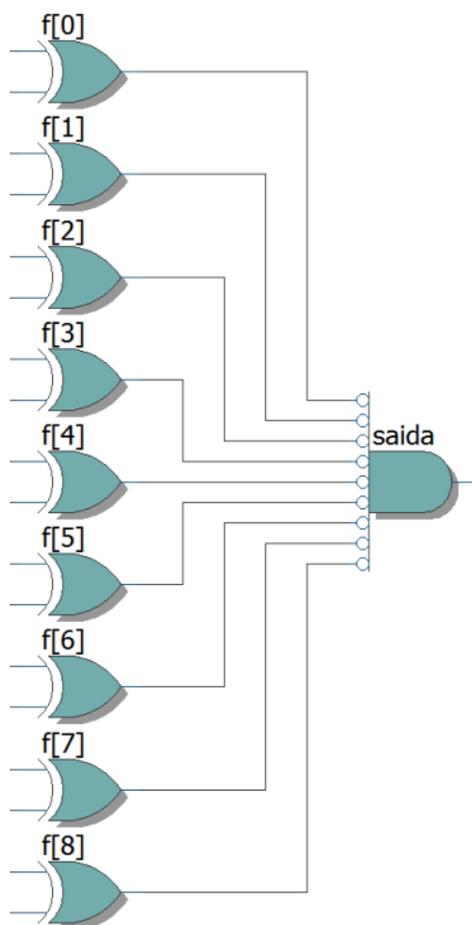


Figura 23 - Implementação das funções lógicas do comparador de nove bits.

Uma parte do código descrito do bloco comparador é apresentada na Figura 24.

```

f(8) <= A(8) XNOR B(8);
f(7) <= A(7) XNOR B(7);

f(6) <= A(6) XNOR B(6);
f(5) <= A(5) XNOR B(5);

f(4) <= A(4) XNOR B(4);
f(3) <= A(3) XNOR B(3);
f(2) <= A(2) XNOR B(2);
f(1) <= A(1) XNOR B(1);

f(0) <= A(0) XNOR B(0);

saida <= f(8) AND f(7) AND f(6) AND f(5) AND f(4) AND f(3) AND f(2) AND f(1) AND f(0);

```

Figura 24 - Código de descrição do comparador de nove bits.

4.1.3 Flip-Flops

Os flip-flops FF1, FF2, FF3 e FF4 (Figura 21), presentes na saída dos comparadores, possuem a função de evitar *glitches* dos comparadores. Os flip-flops $F1 - F2^n$ foram incluídos para minimizar a diferença no tempo de disponibilização dos sinais de saída da memória e as funções lógicas da estrutura CPLI (Cadeia de Portas Lógicas Interconectadas). Para compensar o atraso desses flip-flops no sinal de reinicialização (*reset* do FFOut) do DPWM, o flip-flop $FF5$ foi adicionado ao sinal de inicialização (*set* do FFOut).

4.1.4 Memória

A memória (MEM) implementada no circuito fornece uma sequência de sinais para a estrutura CPLI de acordo com o ciclo de trabalho escolhido. A sequência de sinais serve para determinar a precisão do sinal PWM.

Os endereços da memória são selecionados pelos 8 bits menos significativos de DC ($DC(7:0)$). O conteúdo da memória possui 2^8 sinais, que são conectados aos 2^8 elementos da estrutura CPLI, responsáveis pela resolução fina do DPWM.

A Tabela 2 mostra as sequências utilizadas neste trabalho para a implementação do sinal PWM (Figura 21) cujo ciclo de trabalho tem um ajuste de 17 bits. Pode-se observar que cada linha possui um bit em nível lógico “0” e todos os demais bits da linha são colocados em nível lógico “1”. O bit em nível lógico “0” tem a função de ativar as linhas de atraso visando gerar o sinal de reinicialização (“*reset*”) do Flip-Flop “FFOut” (Figura 21).

DC _D	DC[7:0]	Conteúdo da memória – 256 bits
0	00000000	01111 ... 1111
1	00000001	10111 ... 1111
2	00000010	11011 ... 1111
3	00000011	11101 ... 1111
4	00000100	11110 ... 1111
⋮	⋮	⋮
252	11111100	11111 ... 0111
253	11111101	11111 ... 1011
254	11111110	11111 ... 1101
255	11111111	11111 ... 1110

Tabela 2 - Sequência da memória enviada à estrutura CPLI de acordo com o ciclo de trabalho – DC.

A descrição do código para a implementação da memória do DPWM de 17 bits é mostrada na Figura 25.

```

mux: PROCESS(sel, clk)
BEGIN
  IF rising_edge(clk) THEN
    CASE sel IS
      WHEN "00000000" => saida <= (255 => '0', OTHERS => '1');
      WHEN "00000001" => saida <= (254 => '0', OTHERS => '1');
      WHEN "00000010" => saida <= (253 => '0', OTHERS => '1');
      WHEN "00000011" => saida <= (252 => '0', OTHERS => '1');
      WHEN "00000100" => saida <= (251 => '0', OTHERS => '1');
      .
      .
      .
      .
      .
      WHEN "11111010" => saida <= (5 => '0', OTHERS => '1');
      WHEN "11111011" => saida <= (4 => '0', OTHERS => '1');
      WHEN "11111100" => saida <= (3 => '0', OTHERS => '1');
      WHEN "11111101" => saida <= (2 => '0', OTHERS => '1');
      WHEN "11111110" => saida <= (1 => '0', OTHERS => '1');
      WHEN "11111111" => saida <= (0 => '0', OTHERS => '1');
      WHEN OTHERS => saida <= NULL;
    END CASE;
  END IF;
END PROCESS;

```

Figura 25 - Descrição do código da memória para o DPWM de 17 bits.

Embora a descrição seja semelhante a de um decodificador, o software de síntese faz a implementação utilizando blocos de memória. Para uma resolução fina, com menos bits, é provável que a síntese seja a implementação de um decodificador. Independente da implementação, os atrasos são minimizados pelos flip-flops $F1 - F2^n$.

4.1.5 Cadeia de Portas Lógicas Interconectadas – CPLI

A estrutura CPLI contém os elementos de atraso responsáveis pela resolução fina do DPWM. Os atrasos são controlados pelos n bits menos significativos do ciclo de trabalho. Os elementos são formados por implementações de portas lógicas que se conectam entre si, formando uma cadeia de 2^{n+1} elementos. Os n bits menos significativos de DC selecionam os endereços da memória. O bloco MEM transmite para a CPLI o seu conteúdo, que apresenta uma sequência de sinais.

Para melhor entendimento, considere dois bits de resolução fina:

$$2^{n+1} = 2^{2+1} = 8 \text{ elementos} \quad (5)$$

As implementações lógicas G1, G2, G3 e G4 (Figura 26) são interconectadas através de inversores e possuem como entrada a sequência de sinais. A justificativa para essas implementações será abordada no item 6.1. O menor atraso da CPLI é formado por dois atrasos de *carry chains*, como pode ser visto na Figura 26 (b).

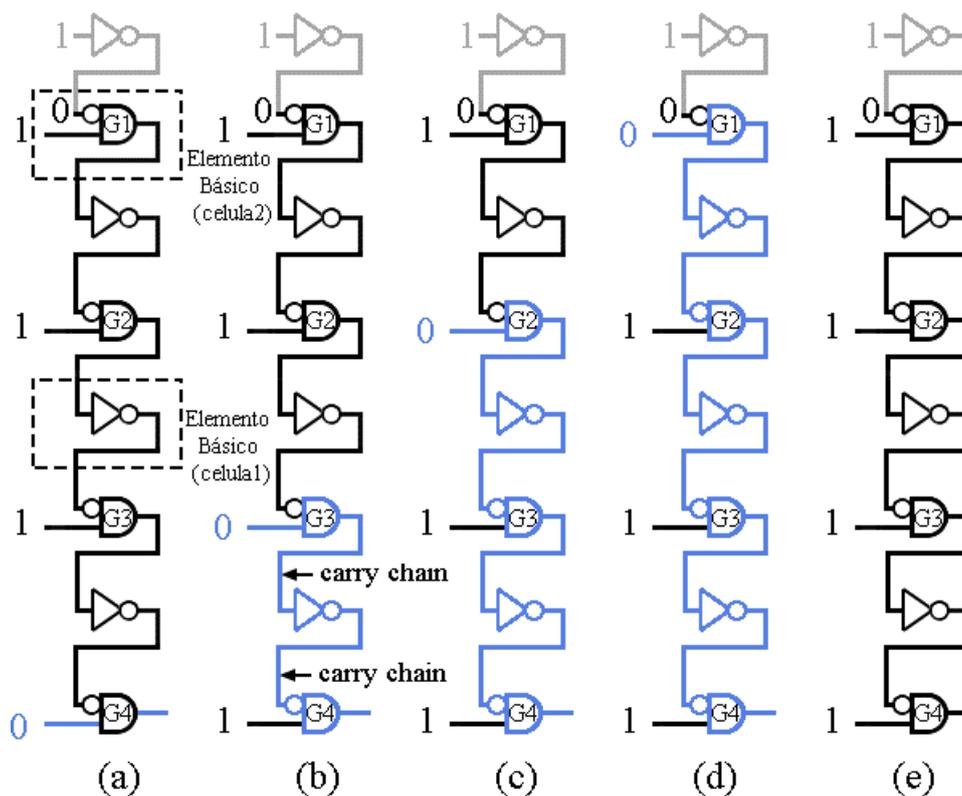


Figura 26 - Funcionamento da Cadeia de Portas Lógicas Interconectadas.

Como o *reset* do FFOut (Figura 21) é acionado em nível baixo, quando a saída do bloco CPLI está em nível alto tem-se o desligamento da cadeia – Figura 26 (e). Esse estado sempre acontece quando o sinal *srtsig* (Figura 21) está em nível baixo (inicialização do pulso DPWM). A Figura 26a apresenta o estado quando a configuração de $DC(1:0)$ é 00_b . A sequência enviada pelo MEM é “0111” e não há atrasos de *carry chains*. O caminho percorrido pelo sinal está representado na cor azul. Na Figura 26b há dois atrasos de *carry chains*, $DC(1:0) = 01_b$ e a sequência enviada é “1011”. Para $DC(1:0) = 10_b$, a sequência é “1101” e há o atraso de quatro *carry chains*. E por último, quando $DC(1:0) = 11_b$, a sequência é “1110” com seis atrasos de *carry chains*.

Para descrever cada elemento básico do bloco CPLI (Figura 26a), foram criados dois componentes: *celula1* e *celula2* (Figura 27), visando viabilizar atrasos iguais entre os elementos básicos. O elemento básico é implementado na FPGA por um elemento lógico (LE). Dependendo da posição na estrutura, o elemento básico utiliza o código da **celula1** ou da **celula2**.

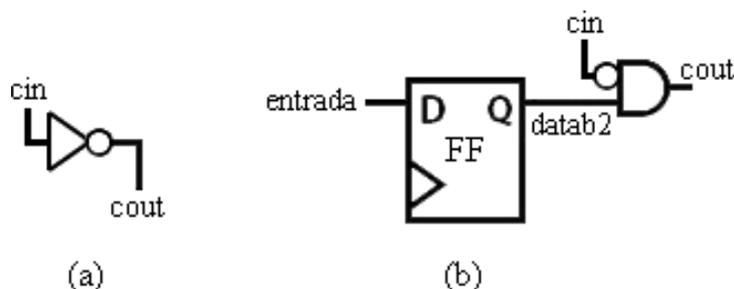


Figura 27 – (a) celula1 (b) celula2.

A descrição do componente **celula1** está representada na Figura 28 e equivale a um inversor (Figura 27a). Na linha 19, a *lut_mask* “0F0F” representa a implementação lógica de um inversor. Esta descrição utiliza um LE e possui uma entrada (*cin*) e uma saída (*cout*).

A *lut_mask* é um parâmetro que configura a LUT para desempenhar uma função lógica. O valor da *lut_mask* pode ser obtido, de uma maneira mais simples, utilizando a ferramenta *Chip Planner* do software Quartus II®. A Figura 29 mostra os passos para a obtenção desse valor.

```

1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  LIBRARY WYSIWYG;
4  USE WYSIWYG.CYCLONEIVE_COMPONENTS.ALL;
5
6  ENTITY celula1 IS
7  PORT ( cin      : IN   STD_LOGIC;
8        cout     : OUT  STD_LOGIC);
9  END celula1;
10
11 ARCHITECTURE archsoma OF celula1 IS
12 SIGNAL datab1 : STD_LOGIC;
13 BEGIN
14     cell: cycloneive_lcell_comb
15     GENERIC MAP(
16         dont_touch => "off",
17         lpm_hint   => "used",
18         lpm_type   => "cycloneive_lcell_comb",
19         lut_mask   => X"0FOF",
20         sum_lutc_input => "cin")
21     PORT MAP(cin => cin,
22             cout => cout);
23
24 END archsoma;
25

```

Figura 28 - Descrição do componente **celula1**.

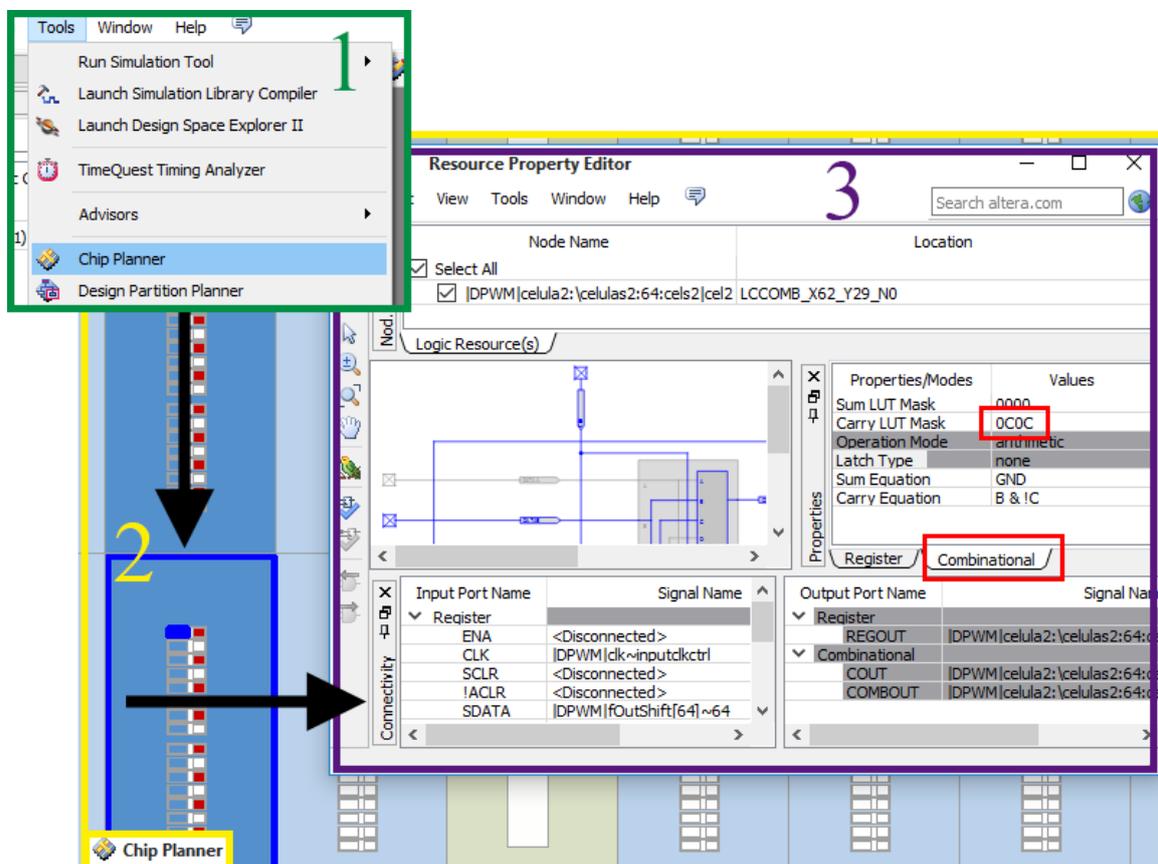


Figura 29 - Acesso ao valor de uma *Lut_Mask* de um LE.

O primeiro passo é o acesso da ferramenta, como mostrado na Figura 29 e delineado como 1. O segundo passo, representando *Chip Planner* (visualização do *floorplan*), é clicar duas vezes em cima de um LE implementado. Uma janela chamada *Resource Property Editor* é aberta (passo 3), mostrando a conexão das LUTs e do registrador. Na janela *Propriedades*, na aba *Combinacional*, são mostrados os valores da *lut_mask* juntamente com a equação da função lógica. Por exemplo, ao escrever a equação de uma função AND com um inversor em uma das entradas ($B \& !C$) no campo *Carry Equation* o valor da *lut_mask* é modificado automaticamente. A Tabela 3 mostra o valor da *lut_mask* de algumas funções lógicas da equação de *carry*. Não existe apenas um valor para uma função pois seu valor depende de quais entradas são utilizadas.

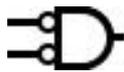
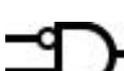
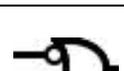
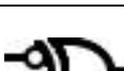
Valor da <i>lut_mask</i>	Função Lógica	Valor da <i>lut_mask</i>	Função Lógica
0303		FCFC	
3F3F		F3F3/F5F5	
0C0C		3C3C/5A5A	
CFCF		C3C3	

Tabela 3 - Valores das *lut_mask* para algumas funções lógicas.

A descrição do componente **celula2** inclui um registrador (implementação para um dos registradores $F1 - F2^n$ - Figura 21) e equivale a um dos elementos G1, G2, G3 e G4 da Figura 26. O componente possui duas entradas (*cin* e *entrada*) e uma saída (*cout*) – Figura 27b. A entrada do componente tem início no registrador, que é conectado ao elemento através do sinal *datab2* (linhas 22 e 32 da Figura 30).

Para melhor compreensão, a conexão dos componentes **celula1** e **celula2** podem ser vistas na Figura 31. Como exemplo, são mostrados três LEs conectados entre si. Dois deles implementados com o componente **celula1** e um implementado com o

celula2. Os dois primeiros LEs representam a conexão entre os dois componentes, que se alternam entre si no restante da cadeia.

```

1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  LIBRARY WYSIWYG;
4  USE WYSIWYG.CYCLONEIVE_COMPONENTS.ALL;
5
6  ENTITY celula2 IS
7  PORT( cin,clk,entrada: IN  STD_LOGIC;
8        cout           : OUT STD_LOGIC);
9  END celula2;
10
11 ARCHITECTURE archsoma OF celula2 IS
12 SIGNAL datab2 : STD_LOGIC;
13 BEGIN
14     cel2: cycloneive_lcell_comb
15     GENERIC MAP(
16     dont_touch => "off",
17     lpm_hint => "used",
18     lpm_type => "cycloneive_lcell_comb",
19     lut_mask => X"OCOC",
20     sum_lutc_input => "cin")
21     PORT MAP(datab => datab2,
22             cin => cin,
23             cout => cout);
24
25     REG2 : cycloneive_ff
26     GENERIC MAP(
27     power_up => "low",
28     x_on_violation => "on")
29     PORT MAP(clk => clk,
30             d => entrada,
31             q => datab2);
32
33 END archsoma;

```

Figura 30 - Descrição do componente **celula2**.

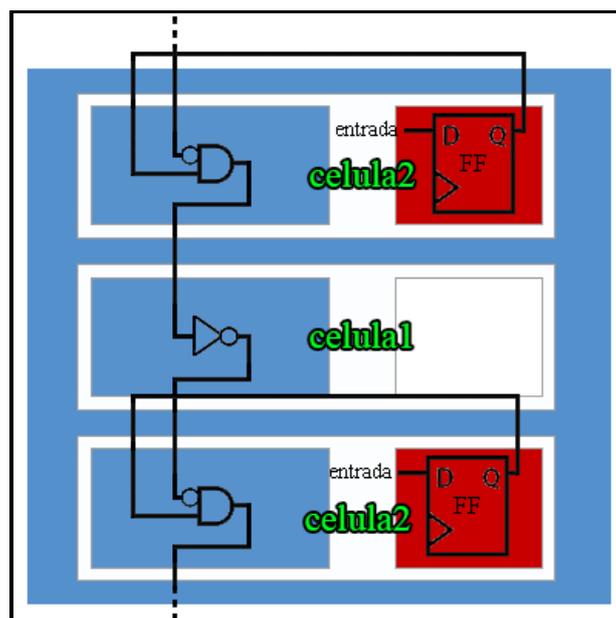


Figura 31 - Conexão dos componentes **celula1** e **celula2** no *floorplan* do Quartus II®.

As descrições dos dois componentes utilizaram a biblioteca WYSIWYG (*What You See Is What You Get*), cujo significado é: o que foi descrito é o que realmente foi implementado. Essa biblioteca permite a configuração de elementos físicos da FPGA CycloneIV®. É possível, através de uma descrição padrão, modificar os atributos desses elementos, garantindo que o software utilize aquele bloco sem que haja a criação de funções lógicas adicionais, evitando o comprometimento de um caminho considerado crítico.

Para ter acesso à descrição padrão dos blocos especiais é preciso abrir um arquivo que contenha esses dados. Esse arquivo é chamado de `cycloneive_components.vhd` e pode ser encontrado no seguinte caminho: `C:\altera\15.0\quartus\libraries\vhd\wysiwyg`.

Outros arquivos de outras FPGAs da Altera® também estão nessa pasta e podem ser acessados do mesmo modo. Os códigos na biblioteca funcionam como componentes e, quando são declarados, seguem as mesmas regras de implementação de um componente, exceto que não é preciso declarar a criação de uma componente, uma vez que estão declarados no arquivo da biblioteca. Com isso, é preciso incluir a biblioteca WYSIWYG à descrição do projeto.

Para a conexão entre os componentes foi utilizado o comando `GENERATE`, como mostra a Figura 32.

```

95  celulas1: FOR n IN ((cadeia/2)-1) DOWNT0 0
96  GENERATE
97      cels1: celula1
98  PORT MAP(cin => fcrry1(n), clk => clk,
99          cout => fcrry2(n));
100 END GENERATE celulas1;
101 fcrry1(0) <= '1';
102
103 celulas2: FOR n IN ((cadeia/2)-1) DOWNT0 0
104 GENERATE
105     cels2: celula2
106 PORT MAP(cin => fcrry2(n), entrada => fOutShift(n),
107         clk => clk, cout => fcrry1(n+1));
108 END GENERATE celulas2;

```

Figura 32 - Código da descrição do bloco CPLI.

O sinal `fcrry1(0)`, na linha 101, representa a conexão do inversor de cor cinza, na Figura 26. As entradas dos elementos formados pelo componente `celula2` são

conectadas pelas funções lógicas destacadas em azul na Figura 21, que são responsáveis por ligar e desligar a estrutura CPLI.

Para o DPWM de 17 bits, são 512 elementos (8 bits de resolução fina – 2^{8+1}), sendo 256 controlados.

Para que o software implemente a aproximação dos registradores com as funções lógicas (Figura 31), foi preciso configurar duas ferramentas do Quartus II®: *TimeQuest Timing Analyzer* e *Design Partition*.

A ferramenta *TimeQuest Analyzer* permite ao projetista restringir, analisar e relatar resultados para todos os caminhos de temporização (*timing paths*) do projeto. É criado um arquivo SDC contendo declarações que indicam ao software Quartus II® como posicionar as células do circuito de acordo com os requisitos de tempo (*timing requirements*) impostos pelo projetista. A primeira declaração em um arquivo SDC geralmente é a do *clock* base do circuito. O comando para a definição do *clock* é o *create_clock* [28]. Um exemplo da sintaxe básica do comando é mostrado na Figura 33.

```
create_clock -name sys_clk -period 8.0 [get_ports fpga_clk]
```

Figura 33 - Descrição do *clock* para o arquivo SDC.

O exemplo descreve um sinal de temporização cujo nome é *sys_clk* com um período de 8 ns aplicado na porta de nome *fpga_clk*. A Figura 34 mostra o acesso ao *Time Quest*, na janela principal do Quartus II®, a criação de um arquivo SDC e o exemplo descrito no editor em três passos.

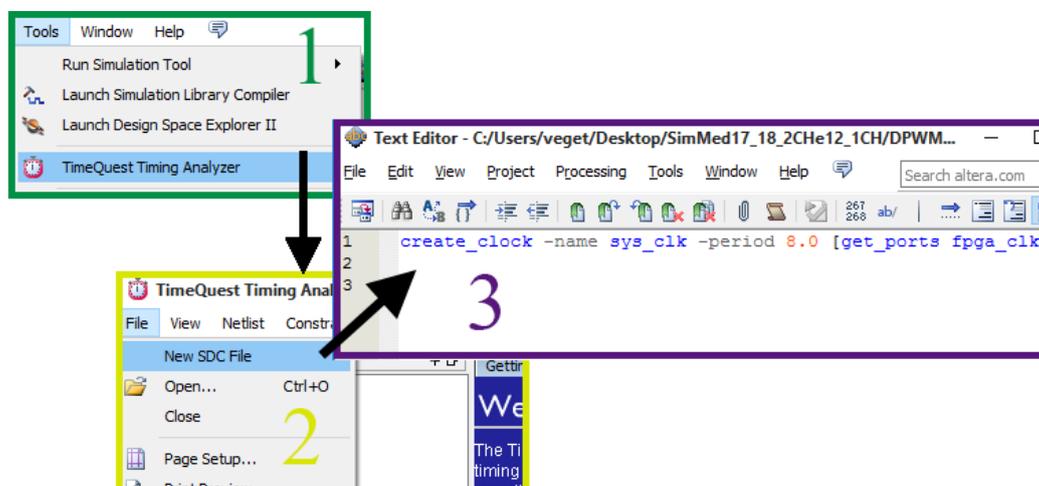


Figura 34 - Criação do arquivo SDC.

Para que o software reconheça o arquivo SDC, é preciso adicioná-lo ao projeto através da configuração *Assignments > Settings* na aba *TimeQuest Timing Analyzer*. Para o projeto, foi realizada uma declaração simples do *clock* no arquivo SDC.

A segunda ferramenta, chamada de *Design Partition*, pode ser utilizada para garantir que uma lógica esteja agrupada e para evitar que o compilador realize otimizações em todos os limites da partição [29]. Uma partição criada no projeto inclui os componentes **celula1** e **celula2**. A Figura 35 apresenta a criação da partição CPLI em cinco passos.

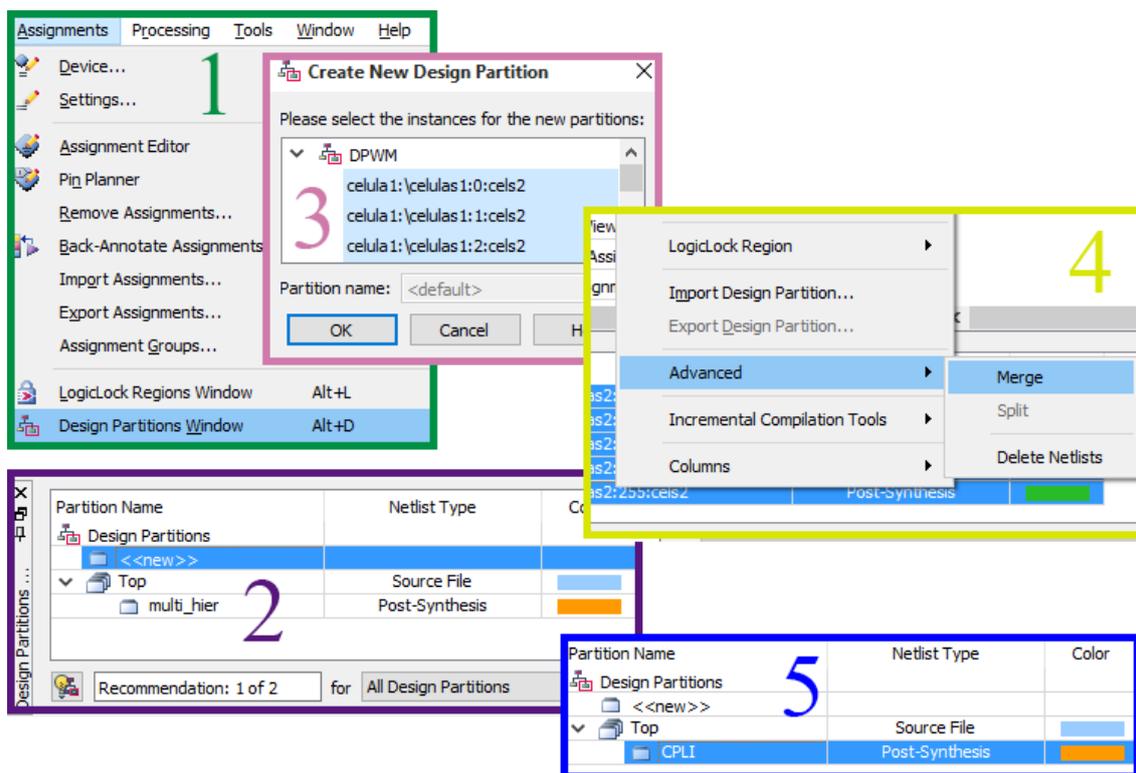


Figura 35 - Criação e unificação de partições.

Em *Assignments > Design Partition Window*, no Quartus II®, uma pequena janela é aberta (passo 1). Clicando duas vezes em '<new>' (passo 2), foram adicionados todos os componentes de *celula1* e *celula2* (passo 3), totalizando 512 partições. Para transformar todas essas partições em uma, essas foram selecionadas e, com o botão direito do mouse, a opção *Advanced > Merge* foi acionada (passo 4). No último passo, a fusão de todas as partições criadas é transformada em uma partição com o nome de CPLI.

A ausência de um desses procedimentos realizados faz com que os registradores sejam separados dos LEs que contêm as funções lógicas. O atraso gerado compromete a monotonicidade do sinal.

4.1.6 Flip-flop de saída (FFOut)

O registrador básico da Cyclone IV® é um flip-flop D. Por este motivo, optou-se por criar um *latch* SR a partir deste registrador. Considerando a Figura 36, nota-se que a entrada D é conectada em nível alto. O sinal que inicializa o pulso DPWM entra no sinal de temporização S, passando o sinal da entrada D para a saída. O pulso DPWM é definido quando o sinal *RstSR*, oriundo do bloco CPLI, adquire nível lógico baixo.

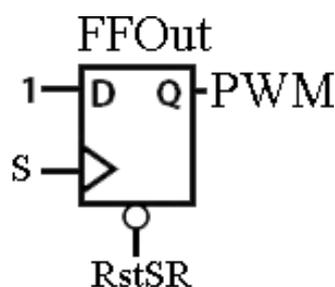


Figura 36 - Flip-flop SR criado a partir de um flip-flop D.

A descrição do registrador é apresentada na Figura 37. A entrada D é conectada em nível lógico 1, conforme descrito na linha 19. O *reset* do registrador é a entrada *clrn* e está conectado ao bloco CPLI através do sinal *RstSR*, na linha 20. Na linha 21, o sinal de *clock* é ligado ao sinal do comparador CPM1 através de S.

```

1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  LIBRARY WYSIWYG;
4  USE WYSIWYG.CYCLONEIVE_COMPONENTS.ALL;
5
6  ENTITY ff_SR IS
7  PORT(
8      S,RstSR          : IN  std_logic;
9      Q                : OUT std_logic);
10
11 ARCHITECTURE archffsr OF ff_SR IS
12 BEGIN
13
14     SR : cycloneive_ff
15     GENERIC MAP(
16         power_up => "low",
17         x_on_violation => "on")
18     PORT MAP(
19         d      => '1',
20         clrn => RstSR,
21         clk  => S,
22         q    => Q);
23
24 END archffsr;

```

Figura 37 - Código da descrição do flip-flop de saída

5 Resultados e Discussões

Os testes realizados incluem a simulação e os resultados de medições da arquitetura principal de 17 bits. A FPGA utilizada é a Cyclone IV® EP4CE115F29C7, da Altera®.

5.1 Simulação da arquitetura principal de 17 bits

As simulações foram realizadas no software Quartus II Versão 15.0.0 Build 145®. A frequência de entrada utilizada foi de 57,6 Mhz e a frequência do DPWM foi de 112,52 kHz.

Para $DC_D = 0$, não é possível obter um sinal PWM válido, pois o ciclo de trabalho determina a largura do pulso e, nesse caso, não há pulso. Teoricamente, o *set* e o *reset* do *latch* SR são acionados ao mesmo tempo. No entanto, é possível observar um pulso de 2,243 ns para o valor mencionado (Figura 38) na simulação da implementação na FPGA. Essa constatação é devido ao roteamento, entre os flip-flops F_{256} e FFOut, apresentar maior atraso em relação ao roteamento entre os flip-flops $FF5$ e FFOut (Figura 21). Os dois roteamentos deveriam apresentar o mesmo atraso, teoricamente.

A Figura 38 mostra o valor da largura de pulso para $DC_D = 0$. O destaque em rosa representa o sinal de *clock*. O destaque cinza representa o sinal PWM. O destaque em vermelho mostra a diferença dos cursores, que representa a largura de pulso do PWM, cujo valor é de 2,243 ns. Uma ampliação do sinal PWM e do sinal de *clock* é mostrada na imagem circular. Os cursores em azul mostram a largura do pulso PWM.

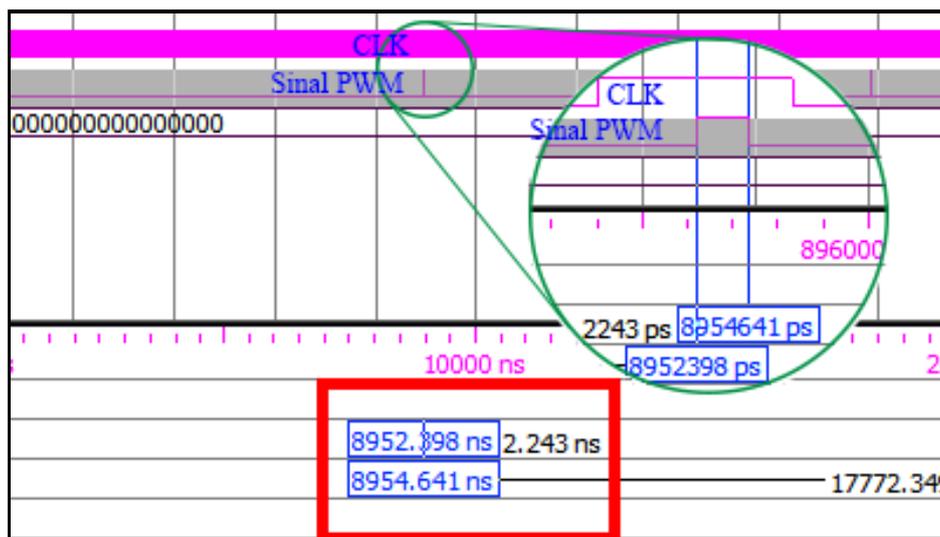


Figura 38 – Simulação do sinal PWM para $DC_D = 0$.

Na Figura 39, é apresentado a simulação para os valores $DC_D = 1$ e $DC_D = 2$. Pode-se observar que ao incrementar um valor no ciclo de trabalho tem-se uma variação de 68 ps, com valor inicial de 2,243 ns.

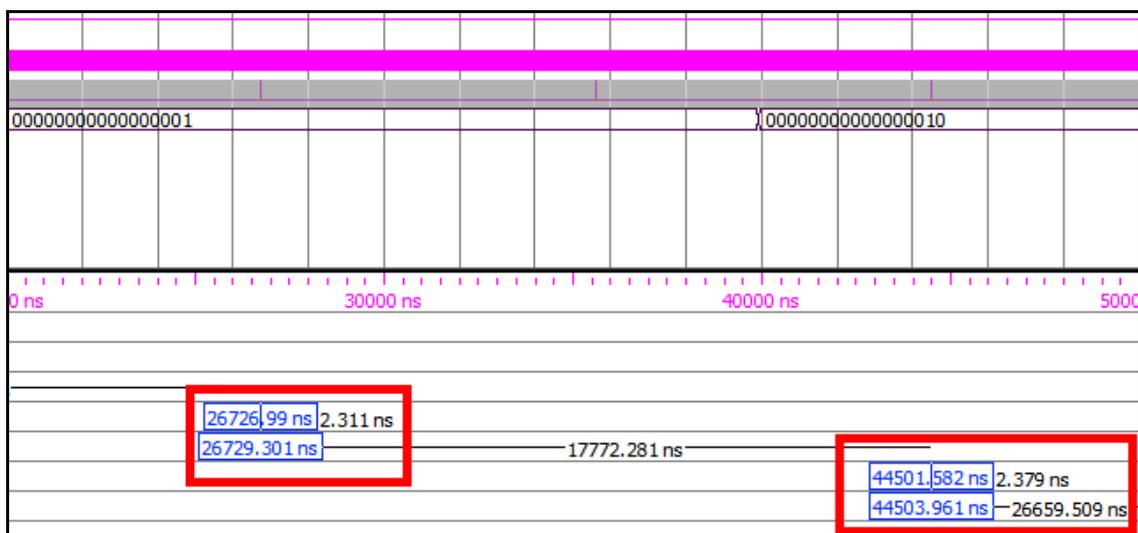


Figura 39 - Simulação do sinal PWM para $DC_D = 1$ e $DC_D = 2$.

Para $DC_D = 26366$ tem-se a simulação na Figura 40. A largura de pulso foi de 1789,982 ns.

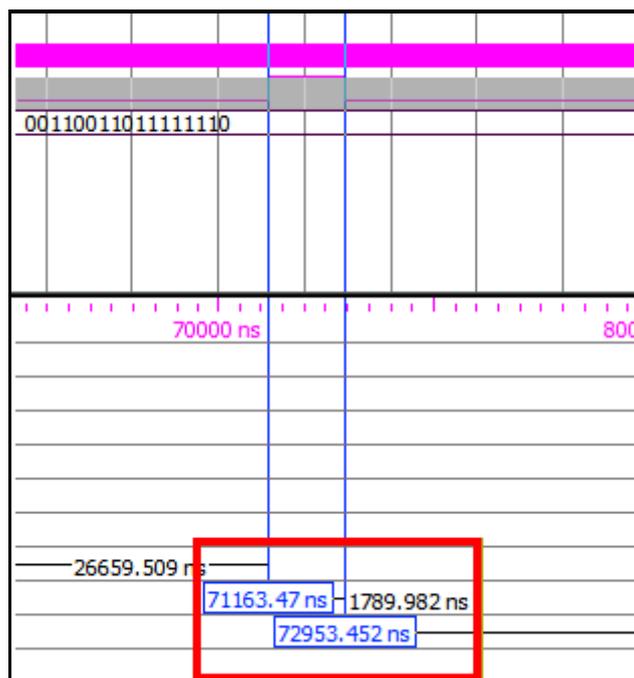


Figura 40 - Simulação do sinal PWM para $DC_D = 26366$.

A Figura 41 mostra a largura de pulso para os seguintes valores de DC: $DC_D = 26367$ e 26368 . As larguras de pulso medidas foram de $1790,05$ ns e $1790,117$ ns. Os intervalos entre esses incrementos em DC foram de 68 ps e 67 ps, respectivamente.

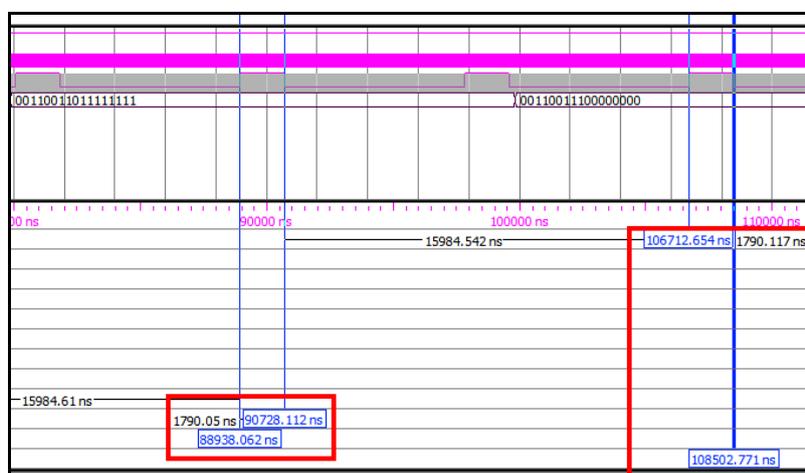


Figura 41 - Simulação do sinal PWM para $DC_D = 26367$ e $DC_D = 26368$.

A simulação para o ciclo $DC_D = 78846$ está representada na Figura 42. A largura de pulso medida para esse ciclo foi de $5348,372$ ns.

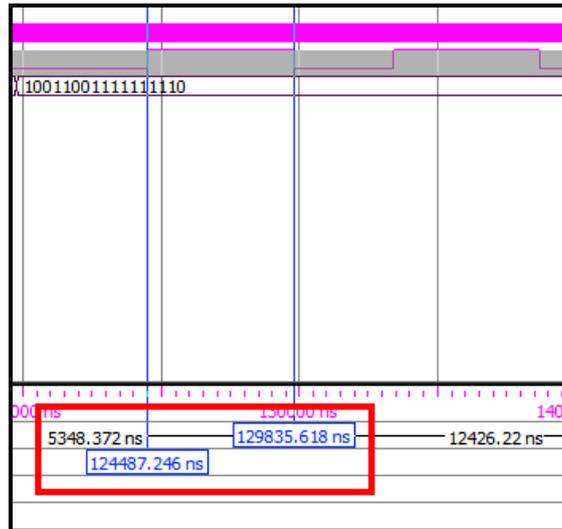


Figura 42 - Simulação do sinal PWM para $DC_D = 78846$.

Para os ciclos $DC_D = 78847$ e $DC_D = 78848$, a Figura 43 mostra as medidas das larguras de pulso. Os valores são de 5348,44 ns e 5348,507 ns. Desses 3 ciclos de trabalho consecutivos, a diferença entre as larguras foram de 68 ps e 67 ps.

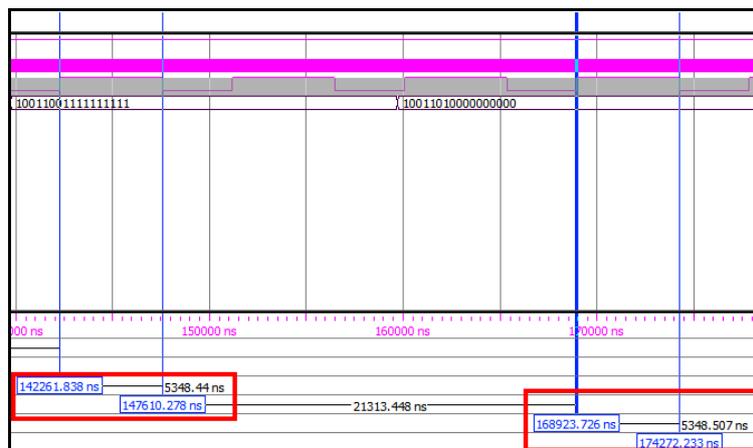


Figura 43 - Simulação do sinal PWM para $DC_D = 78847$ e $DC_D = 78848$.

É apresentada, na Figura 44, a simulação para o ciclo $DC_D = 104958$, cuja largura de pulso é de 7118,888 ns.

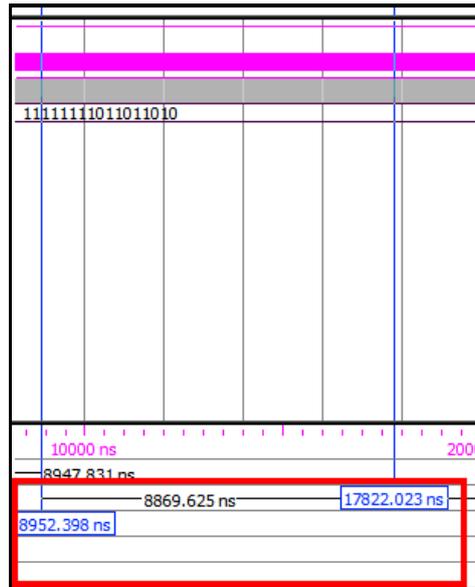


Figura 46 - Simulação do sinal PWM para $DC_D = 130778$.

A Figura 47 ilustra a simulação com os valores $DC_D = 130779$ e $DC_D = 130780$. A largura de pulso para o primeiro foi de 8869,694 ns. A diferença entre os dois pulsos (130778 e 130779) é de 69 ps.

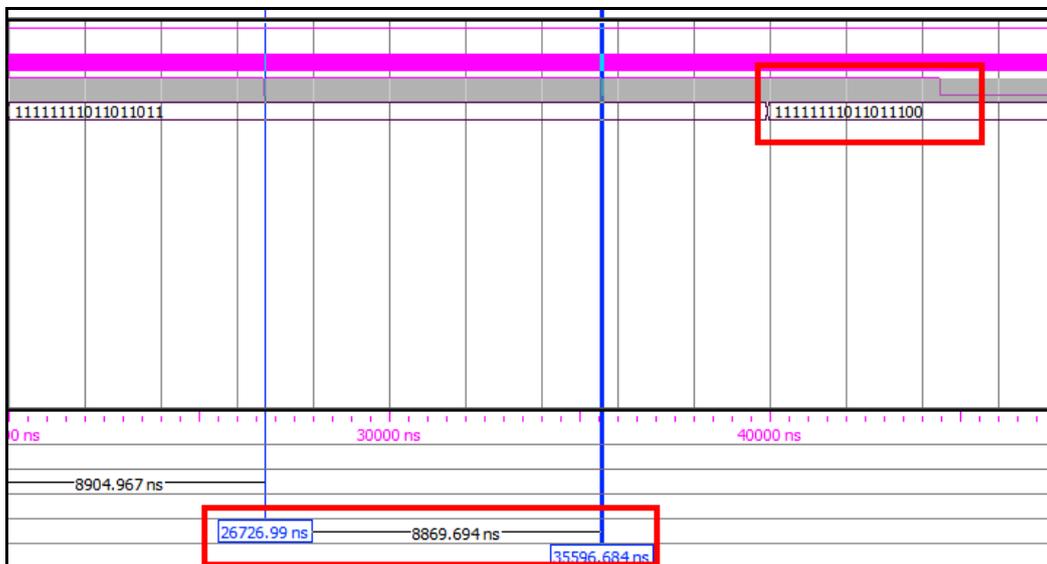


Figura 47 - Simulação do sinal PWM para $DC_D = 130779$ e $DC_D = 130780$.

O sinal PWM para $DC_D = 130780$ apresentou erro devido aos atrasos na chegada dos sinais do *set* e *reset* do FFOut e também do tempo em que o sinal de ativação do *reset* do FFOut permanece em nível lógico zero. Enquanto esse sinal permanece em nível baixo o sinal de *set* aciona a temporização do flip-flop FFOut, anulando o início do

pulso DPWM. Portanto, o ciclo de trabalho varia de 0 a 130779, com 130780 variações, em vez de 131072.

Tendo em vista os resultados de simulação, o circuito possui uma limitação quanto à faixa de valores de DC devido aos roteamentos automáticos implementado pelo software. Porém, esses ciclos de trabalho perdidos correspondem a menos de 0,3% de todas as possibilidades de variações.

A Tabela 4 mostra os valores das larguras de pulso dos respectivos ciclos de trabalho e a diferença de larguras de pulso consecutivas do ciclo Δe – menor incremento. Os valores de Δe nas simulações variaram de 67 ps a 69 ps.

DC_D	DC_B	Largura de pulso - ns	Δe - ps
0	000000000000000000	2,243	
1	0000000000000000001	2,311	68
2	0000000000000000010	2,379	68
⋮	⋮	⋮	⋮
26366	001100110111111110	1789,982	...
26367	001100110111111111	1790,050	68
26368	001100111000000000	1790,117	67
⋮	⋮	⋮	⋮
78846	100110011111111110	5348,372	...
78847	100110011111111111	5348,440	68
78848	100110100000000000	5348,507	67
⋮	⋮	⋮	⋮
104958	110011001111111110	7118,888	...
104959	110011001111111111	7118,956	68
104960	110011010000000000	7119,023	67
⋮	⋮	⋮	⋮
130778	11111111011011010	8869,625	...
130779	11111111011011011	8869,694	69
130780	11111111011011100	-	-

Tabela 4 - Largura de pulso de acordo com as variações dos ciclos de trabalho.

5.2 Resultados experimentais

Os resultados experimentais foram realizados na placa DE2-115®. Utilizou-se o osciloscópio Tektronix MSO4104B 1 GHz para a realização de 768 medidas. A frequência de entrada foi ajustada para 46,45 MHz e a frequência do DPWM foi de 90,72 kHz. Existe uma tolerância de 0,25 MHz para o ajuste da frequência de entrada. Acima desse valor há perdas dos bits de resolução fina. Abaixo desse valor há um atraso maior no último incremento de resolução fina. A cada incremento na resolução principal o último incremento de resolução fina sofre variação de acordo com essa tolerância.

Os nove bits mais significativos do ciclo de trabalho foram testados. A Figura 48 mostra 64 sinais PWM sobrepostos com variações consecutivas do ajuste principal. Os ciclos de trabalho foram variados de $DC_D = 256$ a $DC_D = 16384$. Para o valor $DC_D = 0$, não foi possível realizar a medição devido à dimensão dos tempos de subida e descida, que são maiores que o valor de alguns incrementos. Cada incremento da resolução principal representa 0,195% do ciclo ativo. Portanto, a variação, em porcentagem, é de 0,195% a 12,5%, com passos de 0,195%.

O intervalo entre as larguras de pulso Δc foram medidas e podem ser mais bem observadas com o aumento da região destacada – Figura 49. Os pulsos têm início em, aproximadamente, -1,1 ns e o fim do 64º pulso em, aproximadamente, 1,4 μ s.

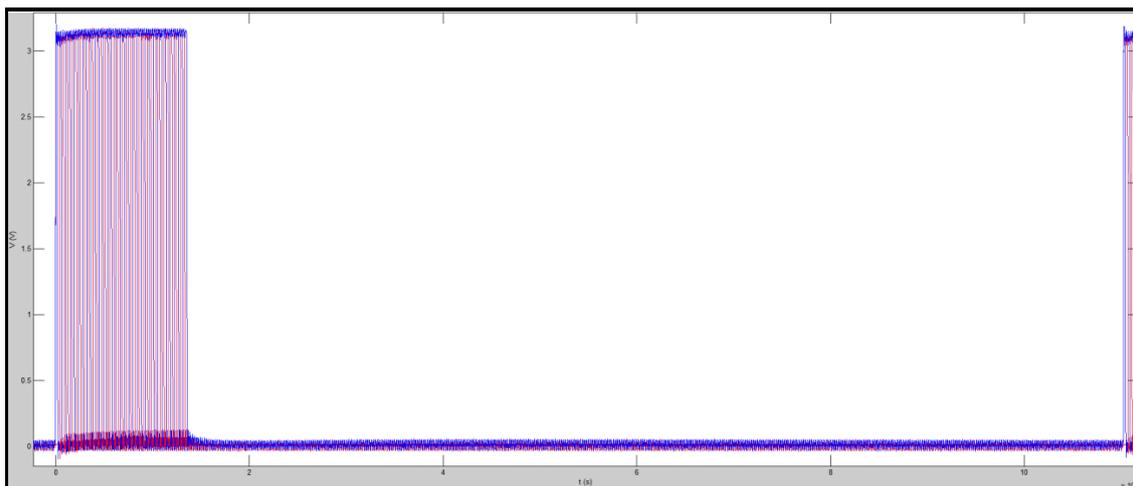


Figura 48 - 64 sinais PWM com variações consecutivas do ajuste principal.

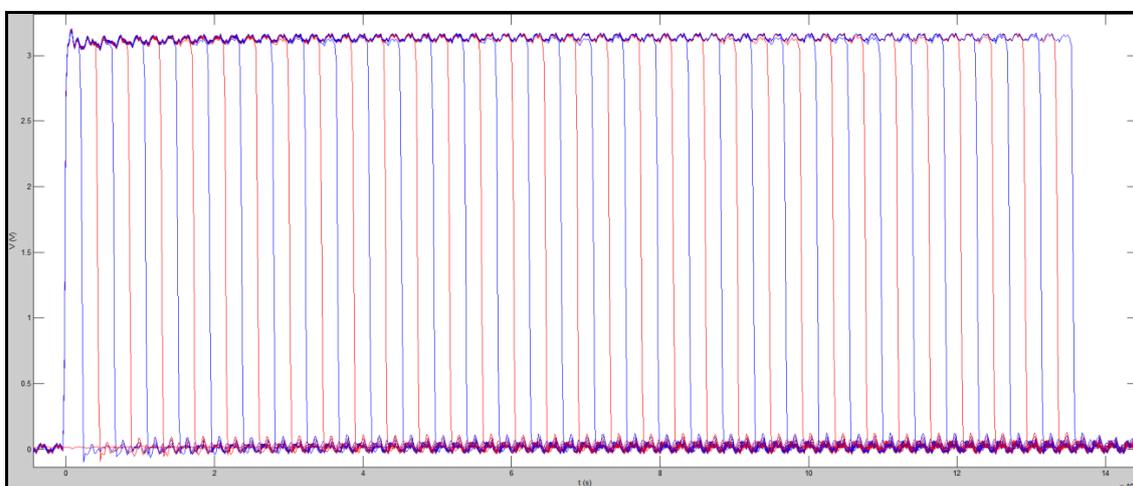


Figura 49 - Intervalos Δc de $DC_D = 256$ a $DC_D = 16384$.

Mais 64 medidas do ajuste principal estão representadas na Figura 50. Os valores do ciclo foram variados entre $DC_D = 16384$ e $DC_D = 32768$. Em porcentagem, essa variação foi de 12,5% a 25%.

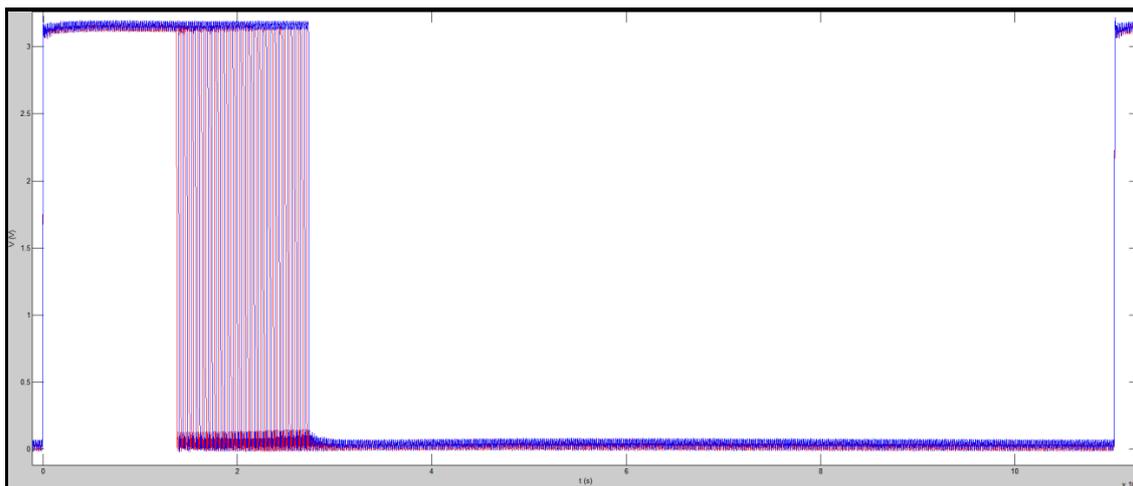


Figura 50 - 64 medidas do sinal PWM. Variações de $DC_D = 16384$ e $DC_D = 32768$.

A ampliação da área em destaque na Figura 50 está representado na Figura 51.
O 128º pulso termina em, aproximadamente, $2,7 \mu\text{s}$.

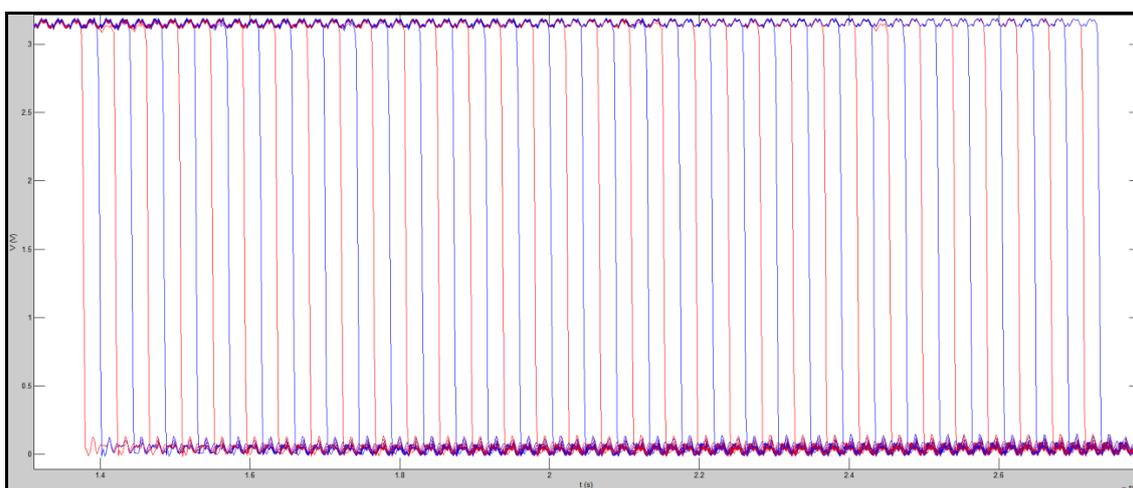


Figura 51 - Intervalos Δc de $DC_D = 16384$ a $DC_D = 32768$.

Na Figura 52 mostra sinais PWM para variações de $DC_D = 32768$ a $DC_D = 49152$. Em porcentagem, a variação do ciclo de trabalho foi de 25% a 37,5%, com mais 64 incrementos.

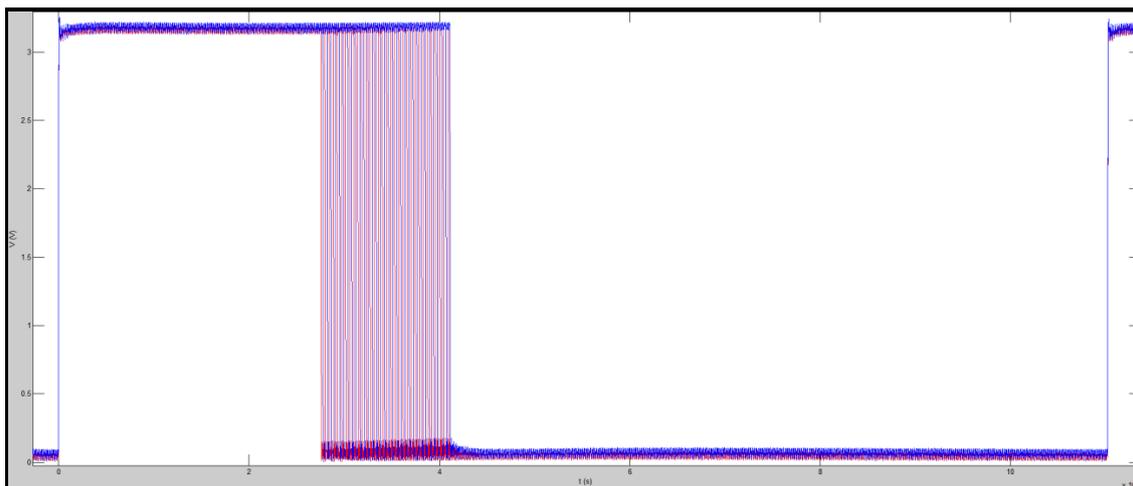


Figura 52 - 64 medidas do sinal PWM. Variações de $DC_D = 32768$ e $DC_D = 49152$.

A Figura 53 apresenta o aumento da região em destaque da Figura 52. O 192° pulso termina em torno de 4,1 μs .

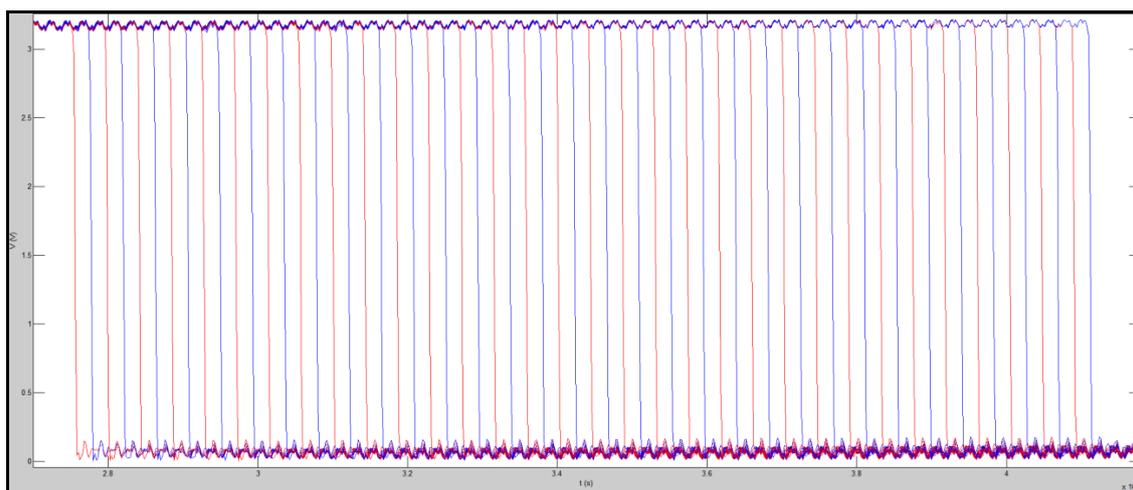


Figura 53 - Intervalos Δc de $DC_D = 32768$ a $DC_D = 49152$.

As variações do ciclo de trabalho de $DC_D = 49152$ a $DC_D = 65536$, do ajuste principal, estão representadas na Figura 54. A variação em porcentagem foi de 37,5% a 50%.

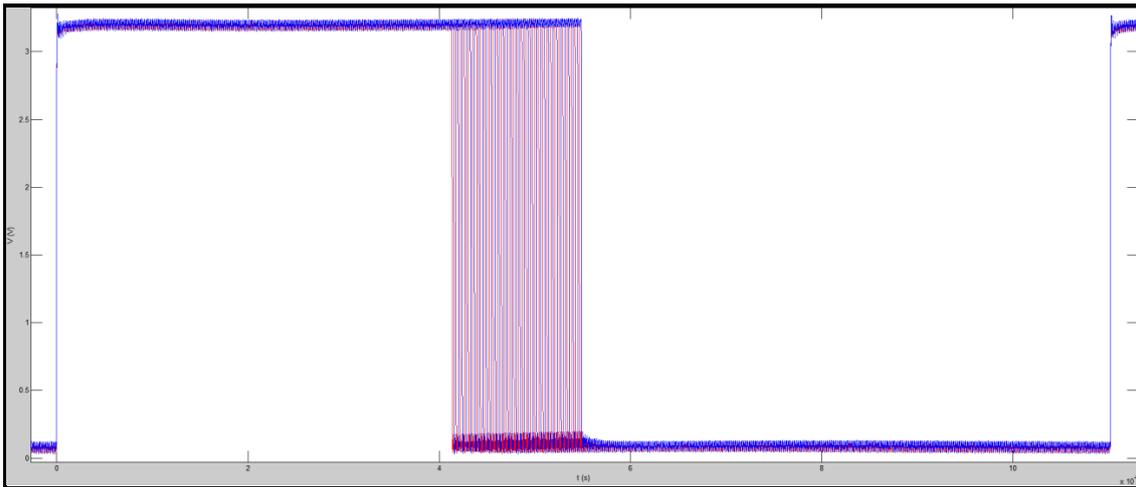


Figura 54 - 64 medidas do sinal PWM. Variações de $DC_D = 49152$ e $DC_D = 65536$.

O aumento da região destacada da Figura 54 é mostrado na Figura 55. O 256º pulso termina em, aproximadamente, $5,49 \mu s$.

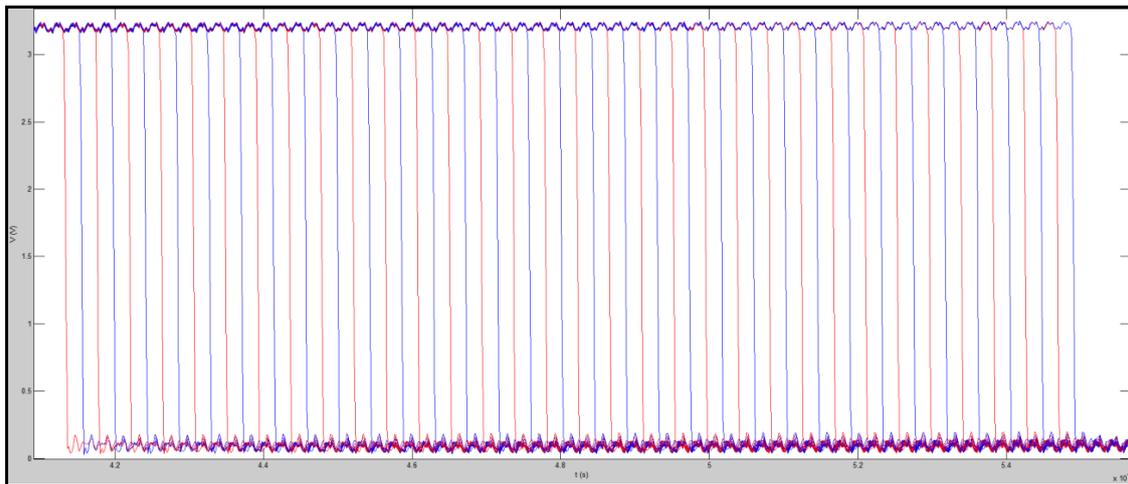


Figura 55 - Intervalos Δc de $DC_D = 49152$ a $DC_D = 65536$.

A Figura 56 apresenta os sinais PWM para os seguintes ciclos: $DC_D = 65536$ a $DC_D = 81920$. A variação em porcentagem foi de 50% a 62,5%.

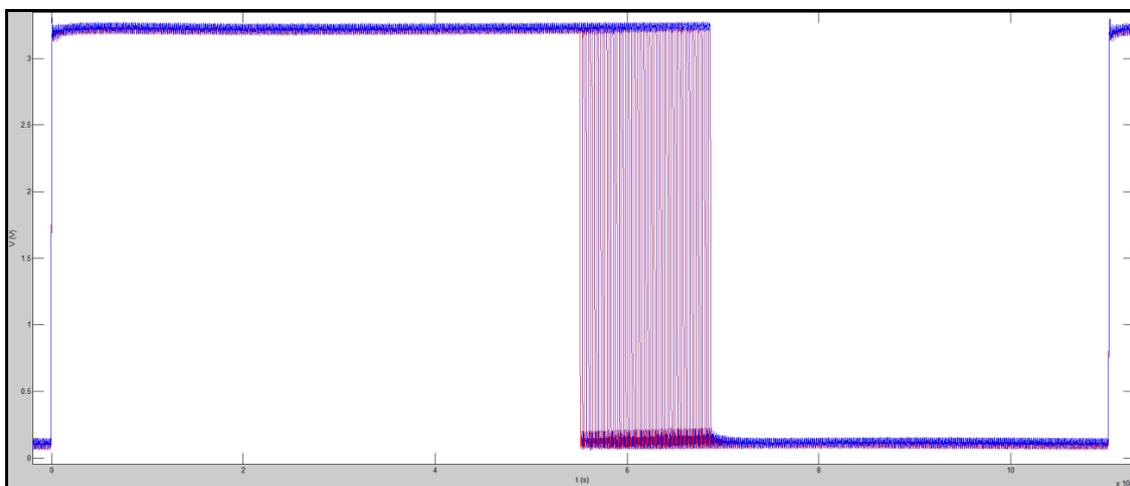


Figura 56 - 64 medidas do sinal PWM. Variações de $DC_D = 65536$ e $DC_D = 81920$.

O aumento da região em destaque da Figura 56 pode ser observado na Figura 57. O 320° pulso termina em torno de 6,9 μ s.

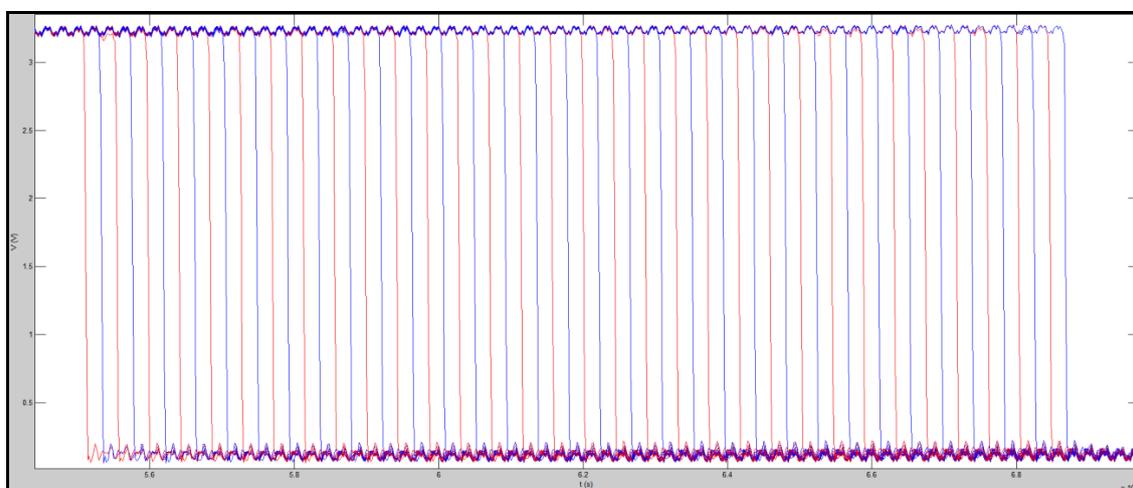


Figura 57 - Intervalos Δc de $DC_D = 65536$ a $DC_D = 81920$.

Para as variações do ciclo de trabalho de $DC_D = 81920$ a $DC_D = 98304$, a Figura 58 representa os sinais PWM para esta faixa. Em porcentagem, o ciclo de trabalho variou entre 62,5% a 75%.

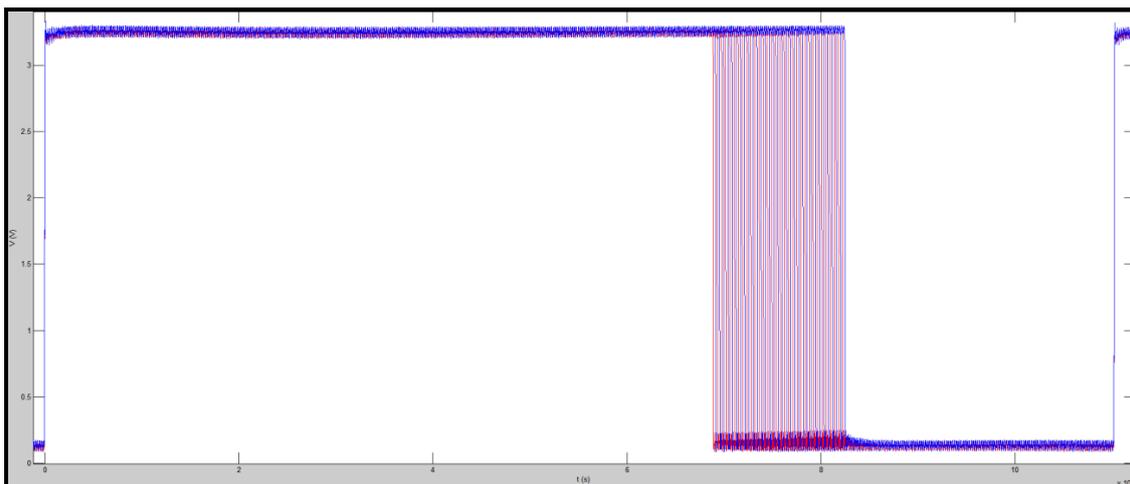


Figura 58 - 64 medidas do sinal PWM. Variações de $DC_D = 81920$ e $DC_D = 98304$.

A Figura 59 mostra o aumento da região em destaque da Figura 58. O 384° pulso termina por volta de $8,25 \mu s$.

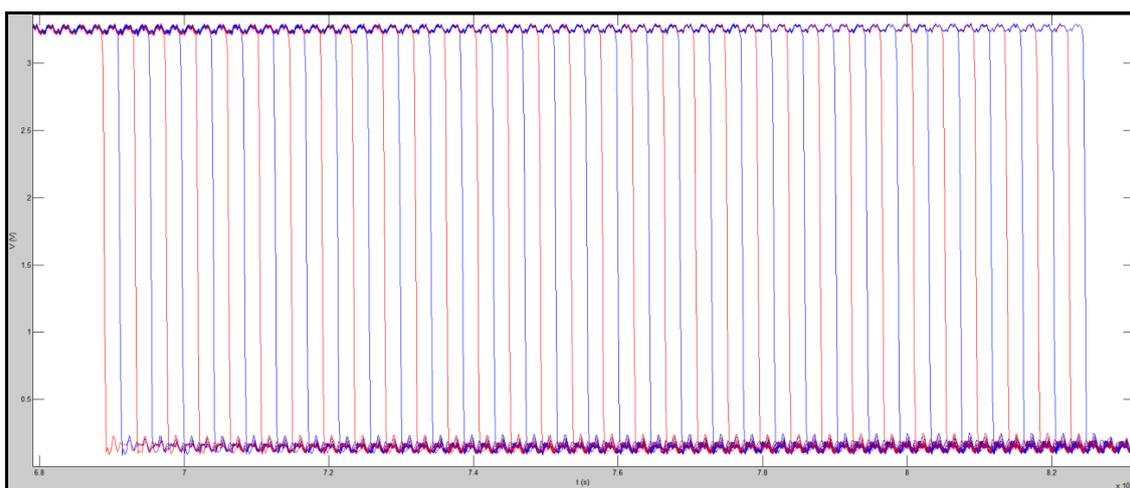


Figura 59 - Intervalos Δc de $DC_D = 81920$ a $DC_D = 98304$.

A Figura 60 apresenta o sinal PWM para as variações de $DC_D = 98304$ a $DC_D = 114688$. Em porcentagem, o ciclo de trabalho variou entre 75% a 87,5%.

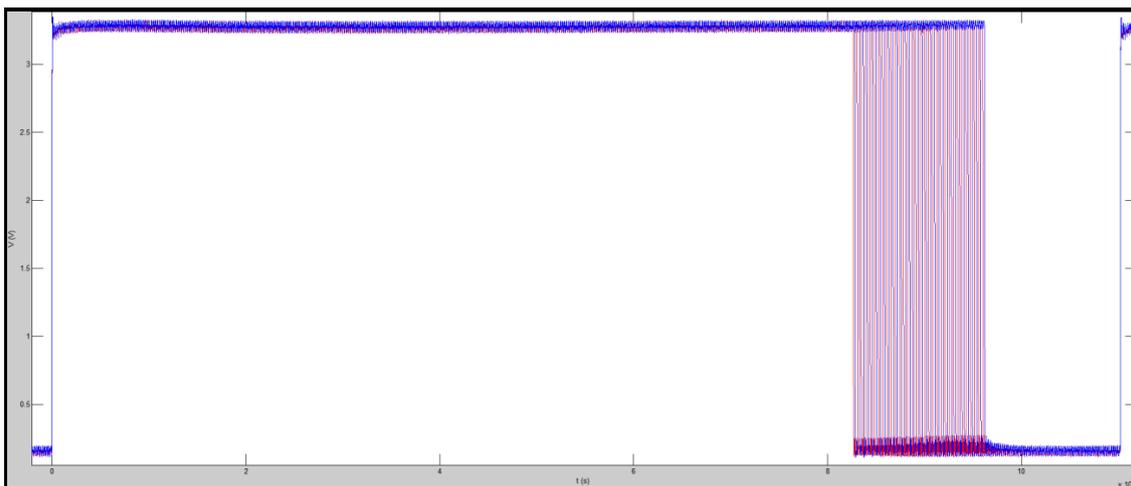


Figura 60 - 64 medidas do sinal PWM. Variações de $DC_D = 98304$ e $DC_D = 114688$.

O aumento da região destacada na Figura 60 está representado na Figura 61. O fim do pulso 448° é, aproximadamente, de 9,6 μ s.

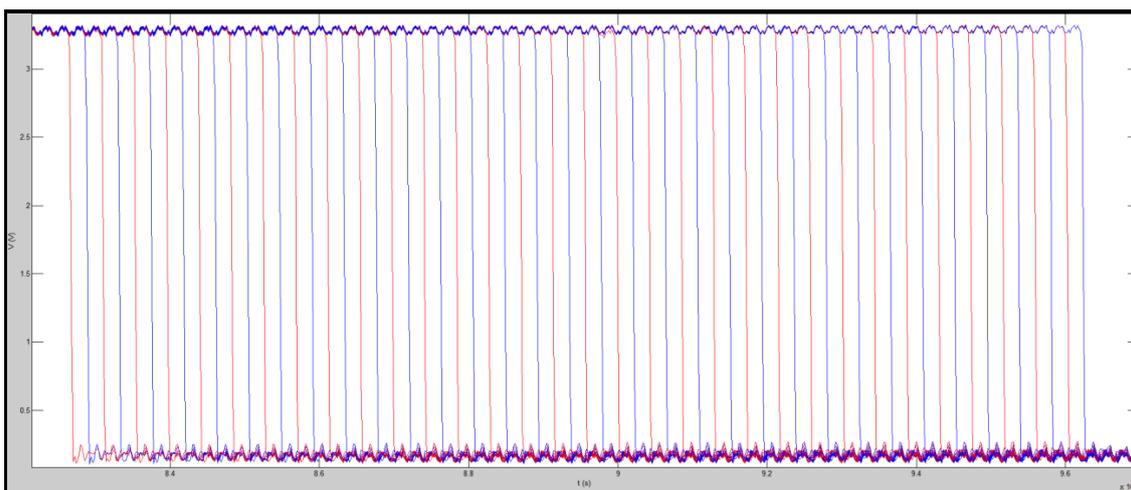


Figura 61 - Intervalos Δc de $DC_D = 98304$ a $DC_D = 114688$.

E por último, a Figura 62 mostra as últimas 63 medidas do sinal PWM. Os ciclos foram variados de $DC_D = 114688$ a $DC_D = 130560$. $DC_D = 130816$ ($DC_B(16:8) = 111111111$) é uma medida inválida, pois nesta configuração o sinal de *set* é acionado no momento em que o *reset* está ativado, anulando o início do pulso PWM. Portanto, esse valor não está inclusa no gráfico. A variação em porcentagem foi de 87,5% a 99,6%.

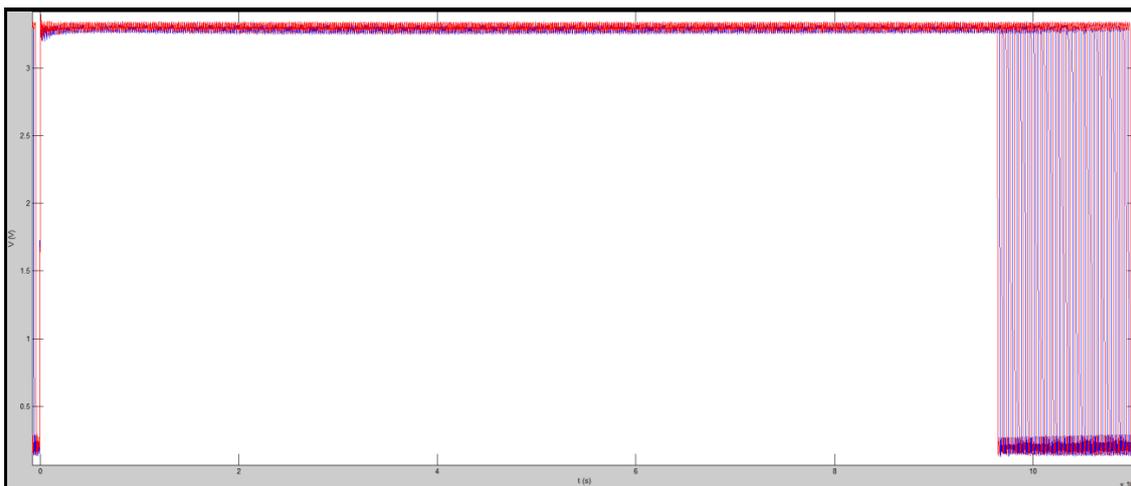


Figura 62 - 64 medidas do sinal PWM. Variações de $DC_D = 114688$ e $DC_D = 130560$.

O aumento da área em destaque, da Figura 62, pode ser observado na Figura 63. O último pulso 511° do ajuste principal termina em torno de $10,9 \mu\text{s}$, sendo o período do PWM de, aproximadamente, $11,023 \mu\text{s}$.

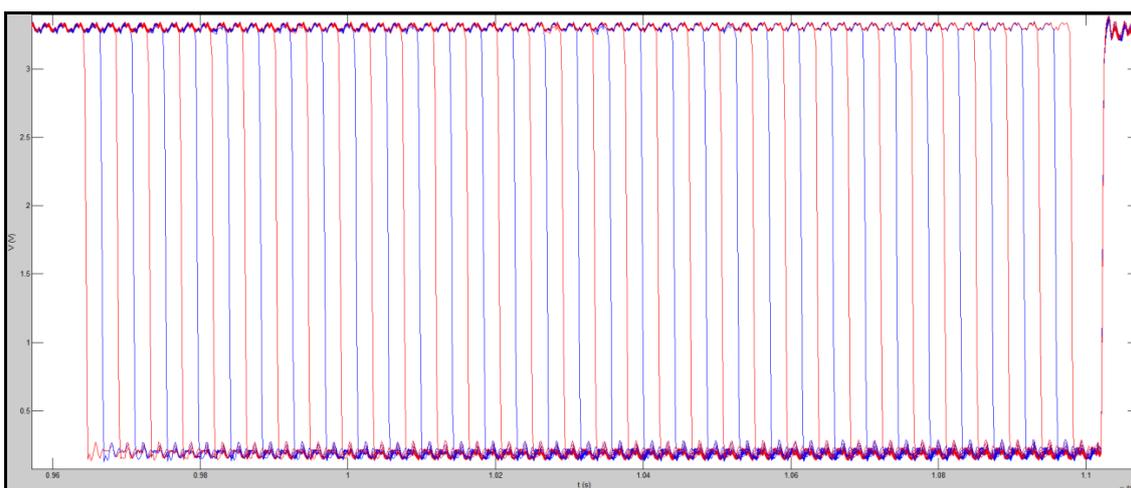


Figura 63 - Intervalos Δc de $DC_D = 114688$ e $DC_D = 130560$.

A média dos intervalos Δc foi de $21,53 \text{ ns}$.

Para o ajuste fino, foram realizadas 256 medidas dos elementos de atraso. O gráfico da Figura 64 mostra 33 larguras de pulso de $DC_D = 512$ a $DC_D = 544$. Cada incremento da resolução fina corresponde a $0,000763\%$. A variação em porcentagem das 33 larguras de pulso foi de $0,003906\%$ a $0,004150\%$.

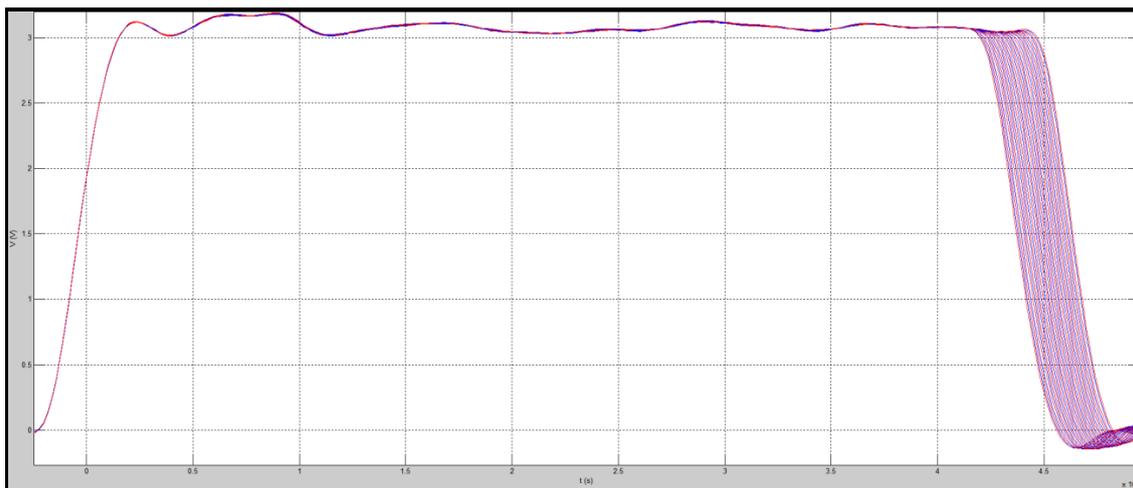


Figura 64 - Primeiras 33 medidas de largura de pulso consecutivas - de $DC = 512$ a $DC = 544$.

A Figura 65 apresenta um aumento da região das variações da largura de pulso da Figura 64. Desta maneira, é possível observar a menor variação Δe do ajuste fino. Nesta faixa de variações, de $DC_D = 512$ a $DC_D = 544$, é possível observar que algumas medidas estão mais bem espaçadas do que outras. A variação média nesta faixa foi de 82,7 ps e um desvio padrão de 10,7 ps.

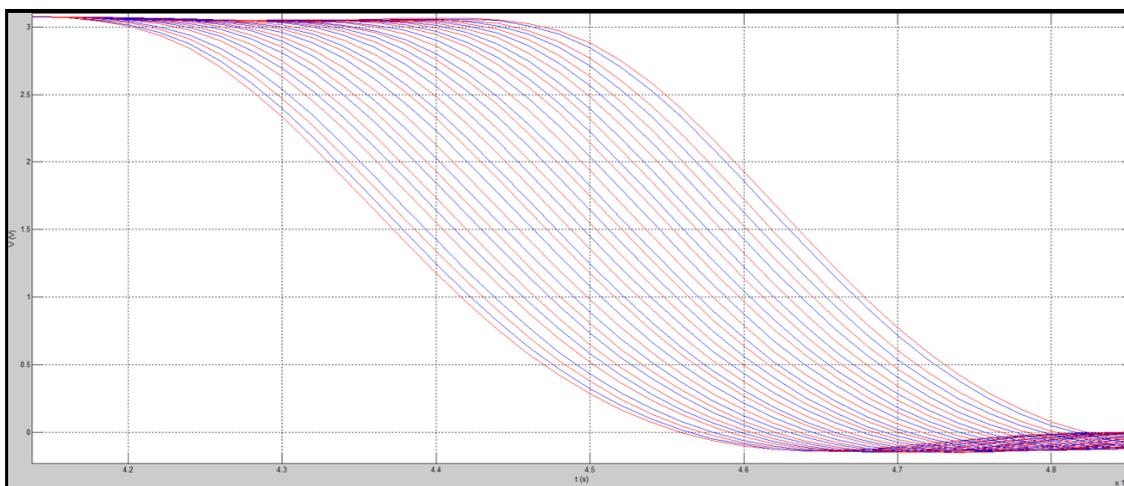


Figura 65 – Intervalos Δe de $DC_D = 512$ a $DC_D = 544$.

O gráfico da Figura 66 apresenta 32 intervalos Δe que variam de $DC = 544$ a $DC = 576$. Nesta faixa de intervalos, a média de variação Δe foi de 83,6 ps e o desvio padrão foi de 14,3 ps. Este intervalo foi o que apresentou maior variação Δe , como pode ser visto na imagem. A variação em porcentagem foi de 0,004150% a 0,004395%.

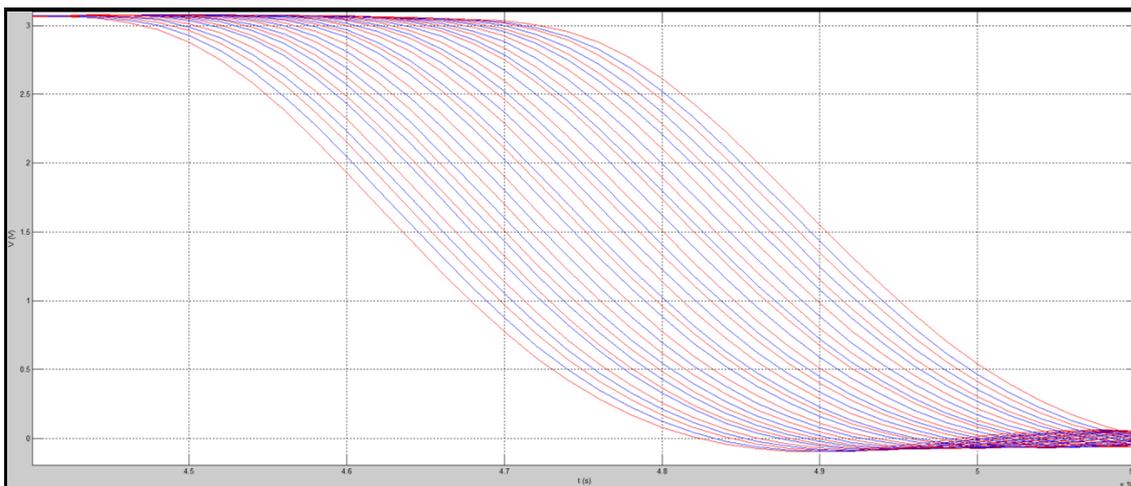


Figura 66 - Intervalos Δe de $DC_D = 544$ a $DC_D = 576$.

A Figura 67 representa 32 intervalos Δe para variações de $DC_D = 576$ a $DC_D = 608$. A média dos valores de Δe , para esta faixa, foi de 84,2 ps e desvio padrão de 9,1 ps. Em porcentagem, a variação do ciclo de trabalho foi de 0,004395% a 0,004639%.

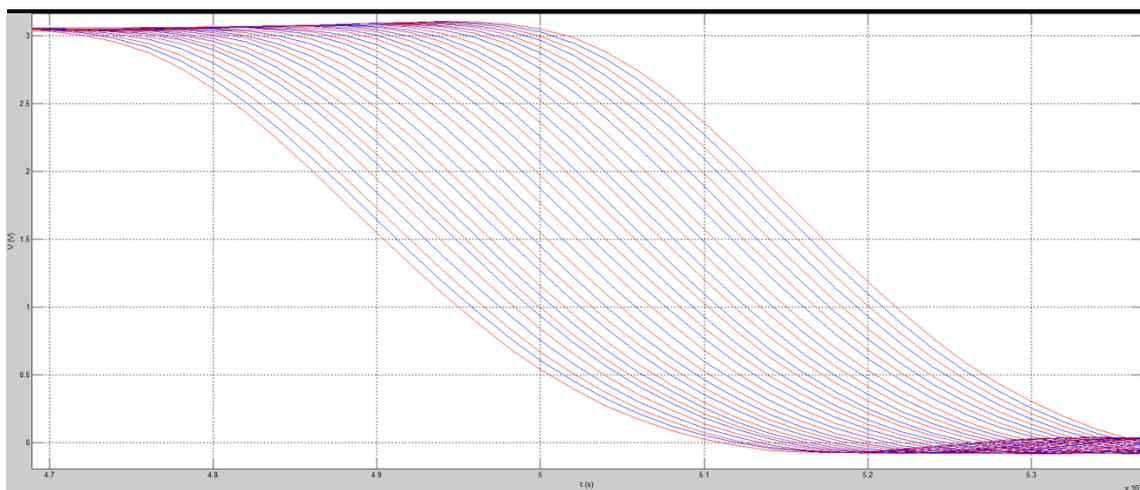


Figura 67 - Intervalos Δe de $DC_D = 576$ a $DC_D = 608$.

Para o gráfico da Figura 68, os 32 valores Δe correspondem às variações de $DC = 608$ a $DC = 640$. Nesta faixa, o valor médio de Δe foi de 83,1 ps, com desvio padrão de 9,2 ps. O ciclo de trabalho variou, em porcentagem, de 0,004639% a 0,004883%.

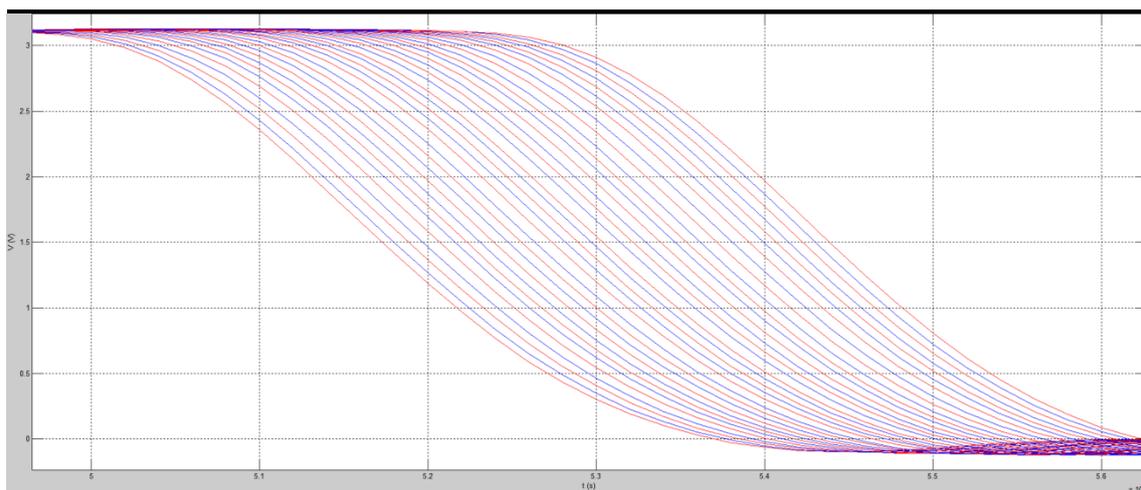


Figura 68 - Intervalos Δe de $DC_D = 608$ a $DC_D = 640$.

Na Figura 69, os intervalos Δe das 33 medidas foram obtidos na faixa de $DC = 640$ a $DC = 672$. A média do valor Δe e o valor do desvio padrão foram de 85 ps e 7 ps, respectivamente. A variação do ciclo, em porcentagem, foi de 0,004883% a 0,005127%.

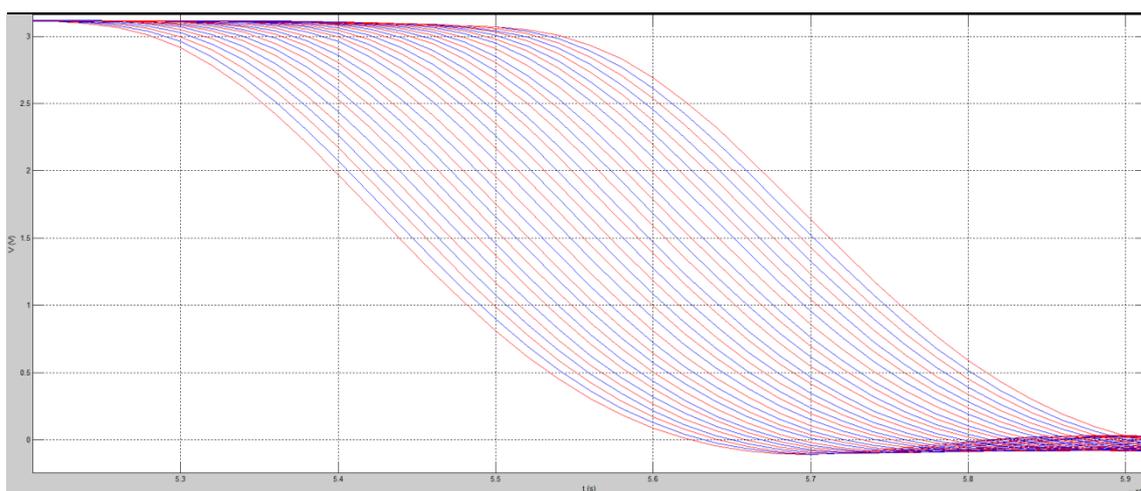


Figura 69 - Intervalos Δe de $DC_D = 640$ a $DC_D = 672$.

O gráfico da Figura 70 mostra os intervalos Δe para as variações de $DC = 672$ a $DC = 704$. A média do valor de Δe foi de 85,7 ps, com desvio padrão de 9,4 ps. A faixa de variação em porcentagem foi de 0,005127% a 0,005371%.

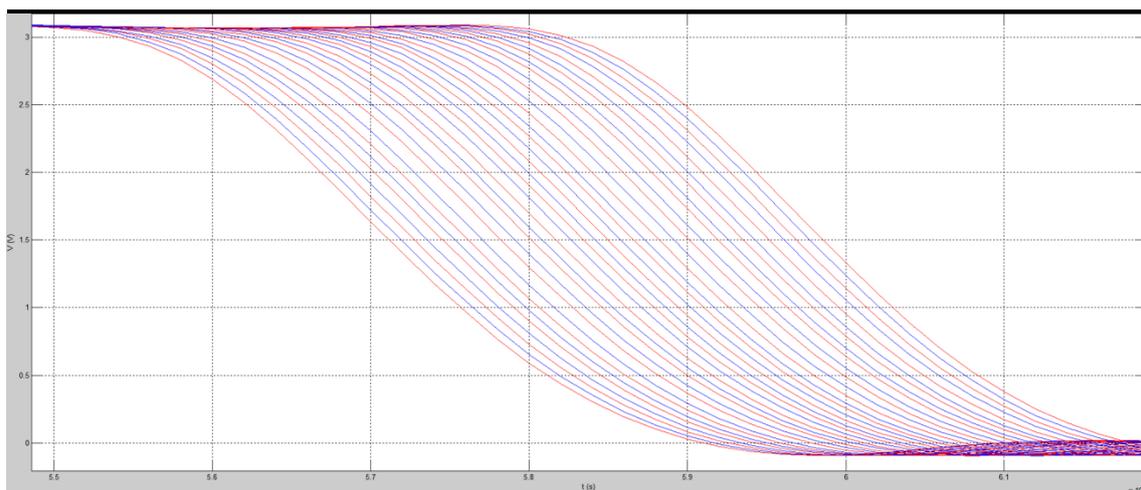


Figura 70 - Intervalos Δe de $DC_D = 672$ a $DC_D = 704$.

As 33 medidas mostradas na Figura 71 foram variadas na faixa de $DC = 704$ a $DC = 736$. A média de Δe foi de 85 ps e o desvio padrão 8,3 ps. A variação em porcentagem do ciclo foi de 0,005371% a 0,005615%.

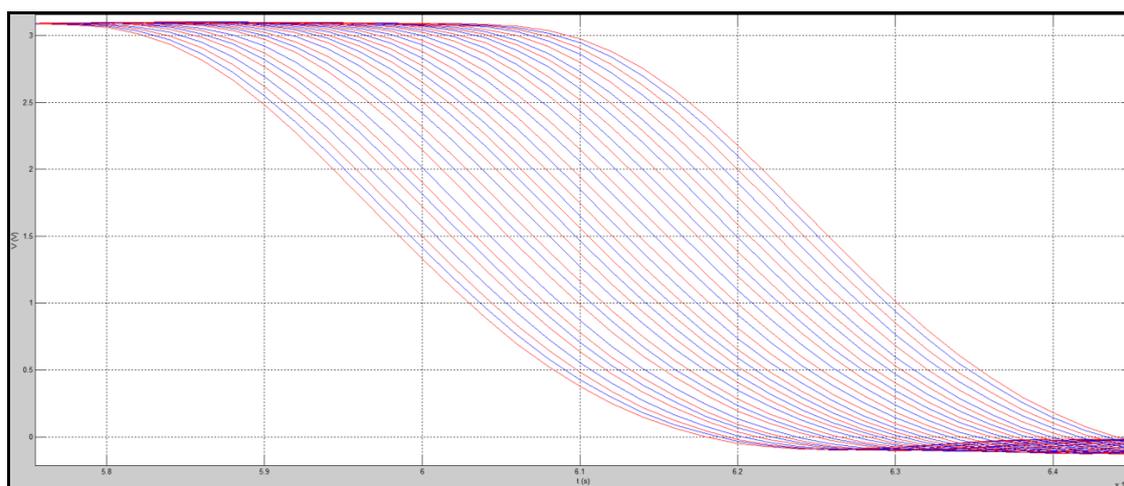


Figura 71 - Intervalos Δe de $DC_D = 704$ a $DC_D = 736$.

O gráfico da Figura 72 apresenta as últimas 33 medidas para o teste do ajuste fino. A faixa de variação foi de $DC = 736$ a $DC = 768$. O valor médio de Δe foi de 84 ps. O desvio padrão foi de 6,5 ps, correspondendo a menor variação nesta faixa. Em porcentagem, o ciclo de trabalho apresentou variação de 0,005371% a 0,005859%.

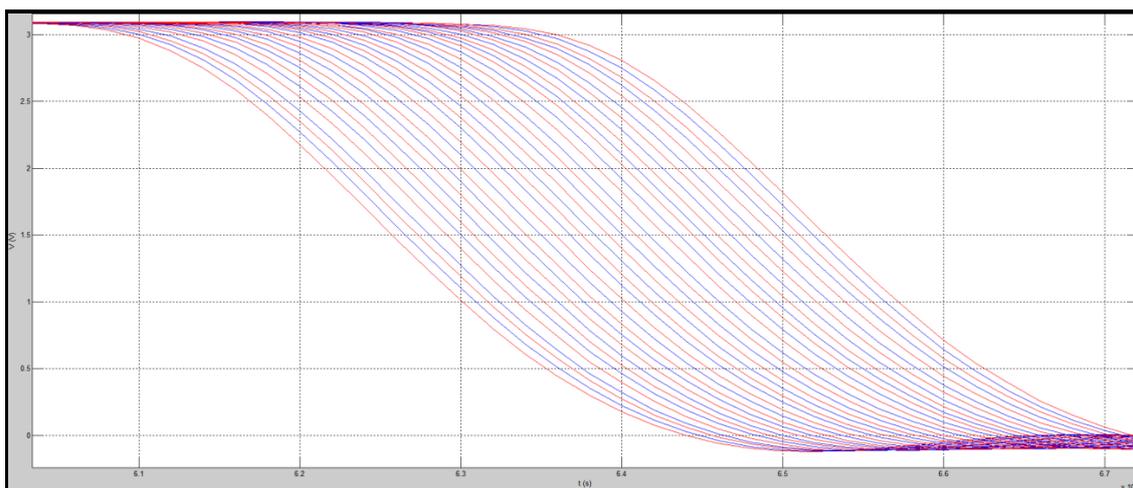


Figura 72 - Intervalos Δe de $DC_D = 736$ a $DC_D = 768$.

A média total dos 256 intervalos Δe foi de 84,22 ps, com desvio padrão de 9,5 ps. O valor máximo foi de 109,63 ps, enquanto o valor mínimo foi de 51,49 ps.

A Figura 73 apresenta a regressão linear dos valores de largura de pulso de acordo com o ciclo de trabalho.

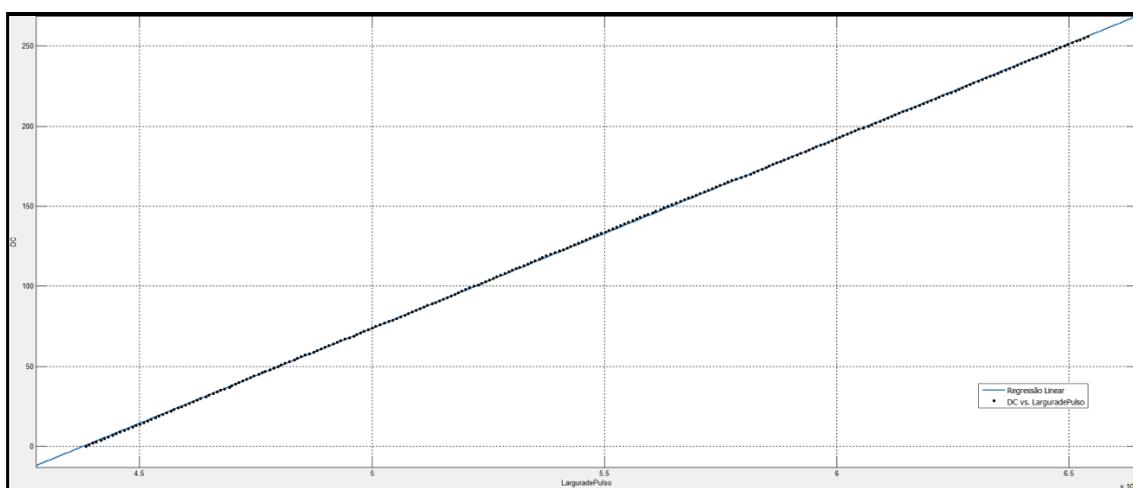


Figura 73 - Gráfico da regressão linear da largura de pulso pelo ciclo de trabalho.

Para melhor visualização, a Figura 74 apresenta a regressão linear dos 74 primeiros elementos de atraso.

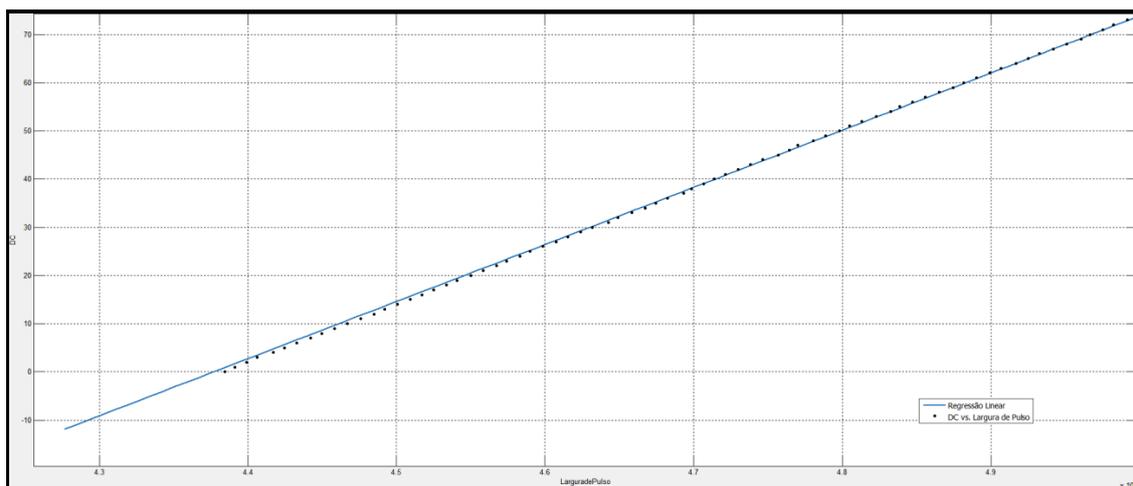


Figura 74 - Ampliação do gráfico da regressão linear para os primeiros 74 elementos de atraso.

A Figura 75 ilustra a regressão linear de 119 medidas (74 ao 192).

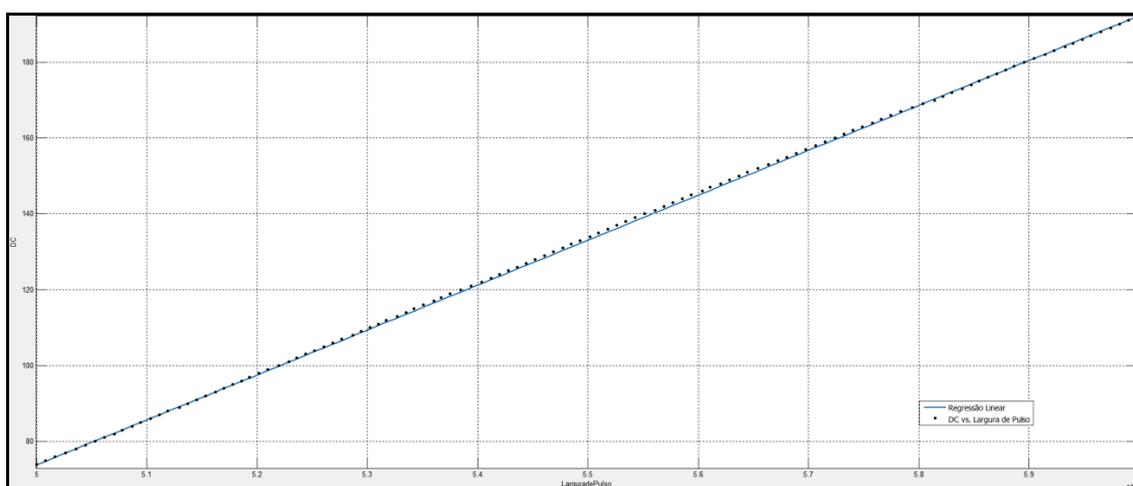


Figura 75 - Ampliação do gráfico da regressão linear para 119 elementos de atraso.

A regressão linear dos últimos elementos de atraso (do 189 ao 256) está representado na Figura 76.

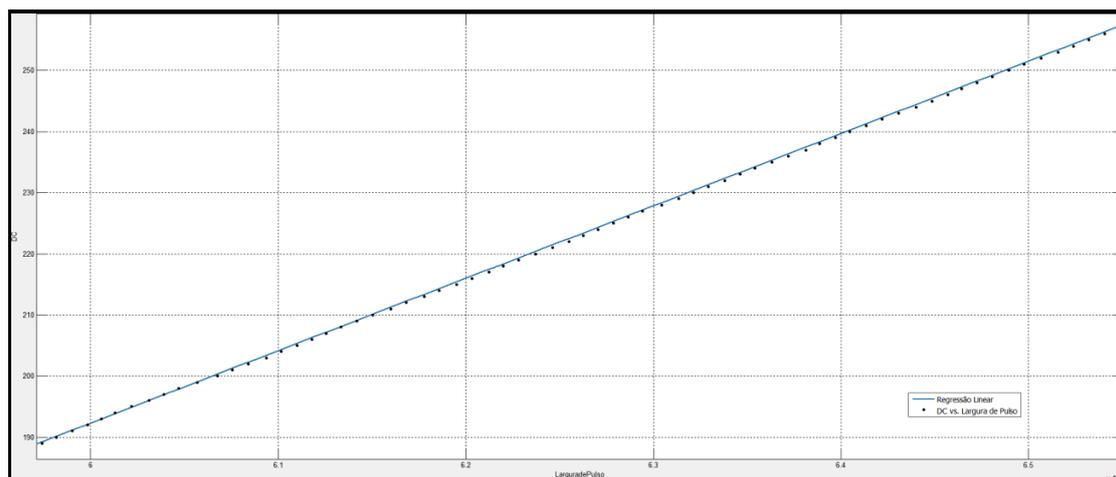


Figura 76 - Ampliação do gráfico da regressão linear para os últimos elementos de atraso.

Todas as variações medidas dos 256 elementos de atraso apresentaram monotonicidade, como pode ser observado nos gráficos de regressão.

Na Tabela 5 são apresentados alguns trabalhos de DPWM desenvolvidos em FPGA. O trabalho de Castro [26] apresenta o menor incremento, com 19,5 ps. Porém, não foi possível obter resultados experimentais uma vez que a obtenção das medidas não são precisas para o valor dessa dimensão. A resolução alcançada foi de 13 bits e o método utilizado foi a utilização de DCM da FPGA Virtex-5®.

O menor incremento do sinal PWM obtida deste trabalho foi bastante satisfatório em comparação com os outros trabalhos, pois apresentou valores próximos aos trabalhos de Costinett [22], Lu-Sheng [21] e Navarro [30] (método que utiliza o bloco IODELAYE1 da FPGA Virtex-6® da empresa Xilinx®) e melhores que Leon [27] e Navarro [30] (bloco DCM).

Dos trabalhos desenvolvidos na literatura, este trabalho apresentou a maior resolução em bits desenvolvida em circuitos implementados em FPGA, com 17 bits, com perdas abaixo de 0,5% de toda a faixa de incrementos.

Trabalhos	$F_{Entrada}$	F_{DPWM}	Resolução em bits	Menor incremento (ps)	Método/FPGA
Castro2010 [26]	100 MHz	6,25 MHz	13	19,5	DCM/Virtex-5
Lu-sheng2010 [21]	50 MHz	1 MHz	15	70 – 80	Somadores/Cyclone II®
Costinett2013 [22]	50 MHz	50 MHz	8	60	Somadores/Cyclone II®
	178 MHz	1 MHz	14	90	Carry chain/ Virtex 4
Navarro2012 [30]	50 MHz	781,25 kHz	11	625	DCM/Spartan-3E
	50 MHz	-	-	78	IODELAYE1/Virtex-6
Leon2014 [27]	25 MHz	955 Hz	10	1022,6	PLL/ Cyclone II®
	200 MHz	777 Hz	10	1256,4	SERDES/ Cyclone II®
Este trabalho	46,45 MHz	90,72 kHz	17	84,22	Carry Chain/Cyclone IV®

Tabela 5 -Trabalhos de DPWM desenvolvidos em FPGA

6 Trabalhos Futuros e conclusão

Inicialmente, outra arquitetura foi desenvolvida com o objetivo de atingir 18 bits com uma frequência próxima a 100 kHz. Cada elemento de atraso possui um atraso de *carry chain* e apresenta o mesmo princípio da arquitetura anterior.

Embora essa arquitetura, na teoria, apresente uma precisão maior que a anterior, na prática os atrasos não apresentaram monotonicidade, como será observado nos resultados, sendo então descartada a sua utilização, uma vez que o tempo disponível não permitia um trabalho mais detalhado que possibilitasse as devidas correções. Essa arquitetura está sendo documentada a fim de dar mais informações e permitir a sua análise e melhoria de desempenho em trabalhos futuros.

6.1 Arquitetura DPWM alternativa

A Figura 77 apresenta o circuito da arquitetura alternativa visando trabalhos futuros. O bloco CPLI possui atrasos mínimos de um *carry chain* e a forma com que os sinais são fornecidos a este bloco foram alterados. O circuito tem 12 bits de resolução, com 3 bits na resolução fina. Para melhor entendimento do funcionamento dos atrasos, é importante destacar os tipos de implementações de portas lógicas criadas pela LUT de cada LE.

Foram verificados, por meio de simulações experimentais, que as LUTs da FPGA Cyclone IV® implementam oito funcionalidades lógicas básicas no modo aritmético. Essas implementações estão representadas na Tabela 6.

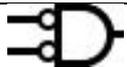
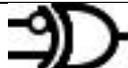
Implementação Lógica 1		Implementação Lógica 5	
Implementação Lógica 2		Implementação Lógica 6	
Implementação Lógica 3		Implementação Lógica 7	
Implementação Lógica 4		Implementação Lógica 8	

Tabela 6 - Implementações lógicas utilizadas nos elementos lógicos da Cyclone IV®.

Todas as combinações dessas funções lógicas foram testadas na estrutura CPLI. Entretanto, somente as que possuíam inversores entre conexões de duas implementações (1, 2, 3 e 4), com exceção da implementação 8, utilizaram as linhas de *carry chain* com menor atraso no processo, em torno de quatro vezes menor em relação as conexões sem inversores. Para esta arquitetura, Figura 77, foi escolhida a implementação lógica 3.

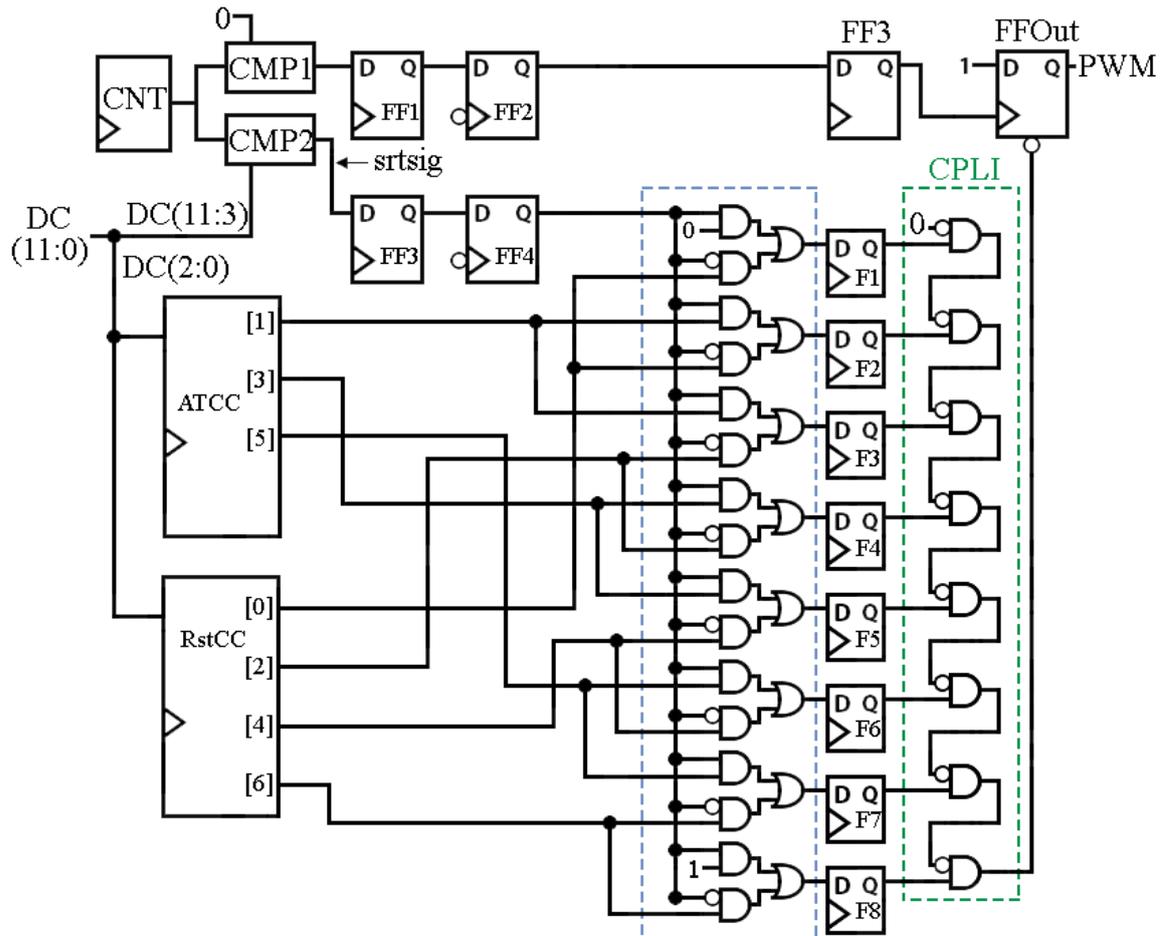


Figura 77- Arquitetura DPWM.

Há dois blocos que fornecem sinais para a estrutura CPLI. O bloco ATCC fornece os sinais que habilitam o funcionamento da CPLI. O bloco RstCC desativa a estrutura. A Figura 78a mostra a CPLI desativada para $DC(2:0) = "000"$. Os sinais de entrada ("00000000") provêm do bloco RstCC. A estrutura é ligada quando G8 recebe nível lógico 1 (Figura 78b). Desta maneira, a CPLI é ligada sem utilizar os atrasos de *carry chain*. Para $DC(0:2) = "001"$, a CPLI é desligada com a sequência "11000000" (Figura 78c). Quando ATCC envia uma mudança de sinal em G7, esta ativa a CPLI com um atraso de *carry chain*, como pode ser visto na Figura 78d. Para $DC(2:0) = "010"$,

RstCC envia a sequência "11000000", que desliga a CPLI, Figura 78e. O atraso de dois *carry chains* é efetivado quando ATCC muda o sinal em G6, conforme a Figura 78f.

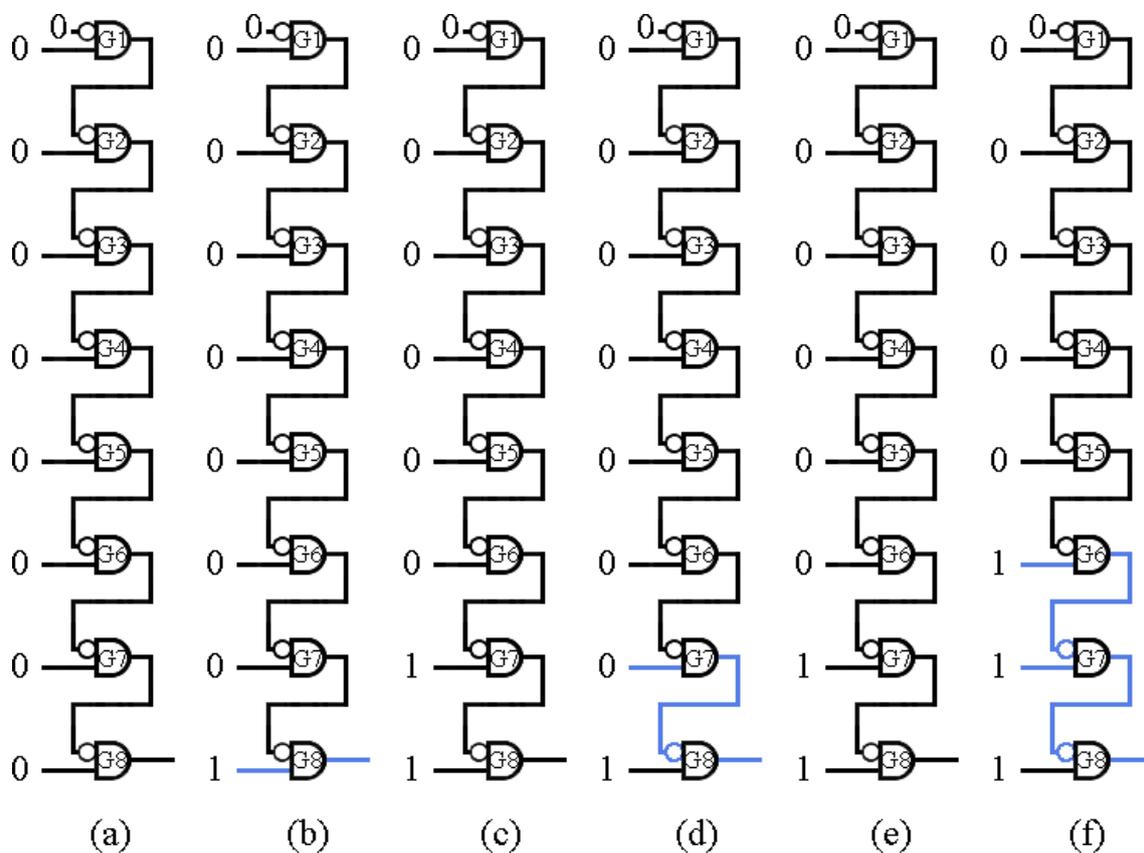


Figura 78 - Funcionamento da CPLI da arquitetura 2.

A sequência de ativação e desativação da estrutura CPLI é mostrada na Tabela

7.

DC[3:0]	RstCC	ATCC
000	00000000	10000000
001	11000000	10000000
010	11000000	11100000
011	11110000	11100000
100	11110000	11111000
101	11111100	11111000
110	11111100	11111110
111	11111111	11111110

Tabela 7 - Sequência de sinais de RstCC e ATCC enviadas a CPLI.

6.2 Simulação da Arquitetura de 12 bits

A simulação da arquitetura de 12 bits foi realizada utilizando-se o software Quartus II Versão 15.0.0 Build 145. A frequência de entrada utilizada foi de 59,21 Mhz. As simulações a seguir foram realizadas apenas a fim de comprovar a funcionalidade dos elementos de atraso (ajuste fino).

A Figura 79 apresenta os sinais de PWM para $DC_D = 2048$ e $DC_D = 2049$. As larguras de pulso medidas foram 4324,048 ns, para $DC_D = 2048$ e 4324,064 ns para $DC = 2049$.

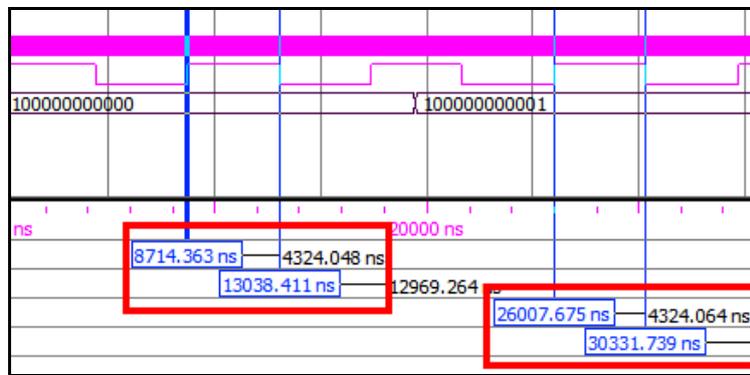


Figura 79 - Simulação do sinal PWM para $DC_D = 2048$ e $DC_D = 2049$.

Para os ciclos $DC_D = 2050$ e $DC_D = 2051$, a Figura 80 mostra os valores das larguras de pulso: 4324,155 ns e 4324,180 ns, respectivamente.

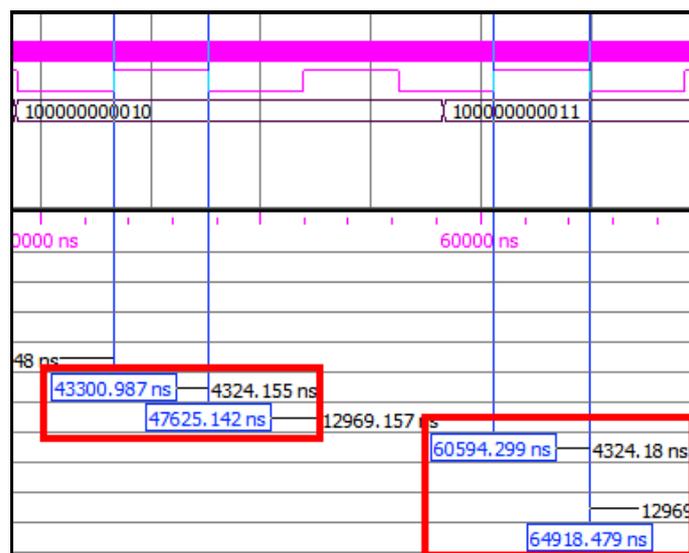


Figura 80 - Simulação do sinal PWM para $DC_D = 2050$ e $DC_D = 2051$.

A Figura 81 mostra as larguras de pulso para $DC_D = 2052$ e $DC_D = 2053$ e seus respectivos valores foram: 4324,279 ns e 4324,295 ns.

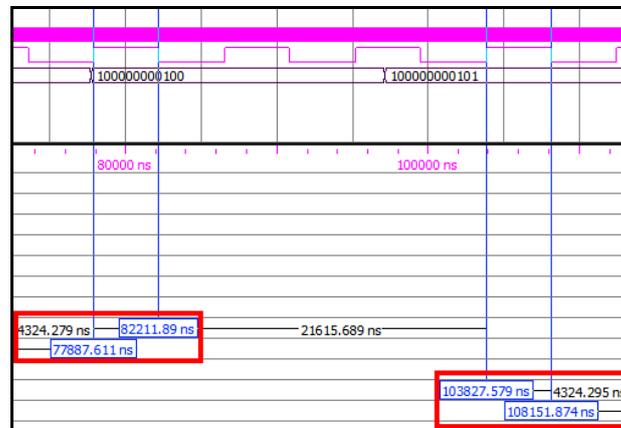


Figura 81 - Simulação do sinal PWM para $DC_D = 2052$ e $DC_D = 2053$.

As larguras de pulso para os ciclos $DC_D = 2054$ e $DC_D = 2055$ apresentaram os seguintes valores: 4324,394 ns e 4324,41 ns, respectivamente (Figura 82).

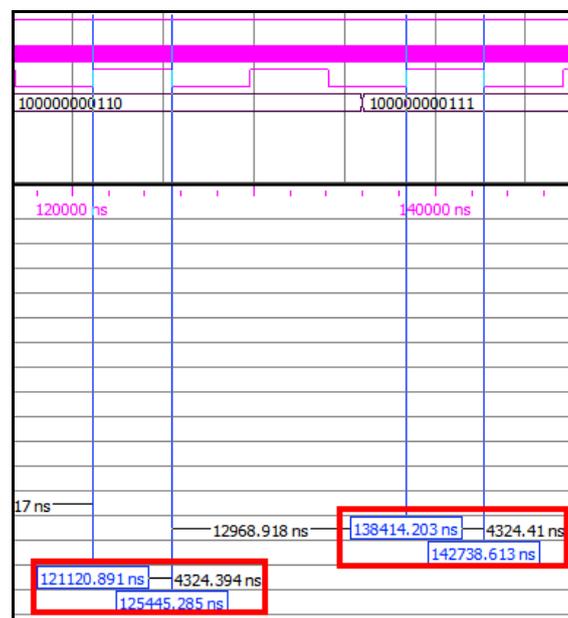


Figura 82 - Simulação do sinal PWM para $DC_D = 2054$ e $DC_D = 2055$.

A Tabela 8 apresenta os valores Δc para a simulação do sinal PWM da arquitetura 2.

DC _D	DC _B	Largura de pulso - ns	Δc - ps
2048	100000000000	4324,048	-
2049	100000000001	4324,064	16
2050	100000000010	4324,155	91
2051	100000000011	4324,180	25
2052	100000000100	4324,279	99
2053	100000000101	4324,295	16
2054	100000000110	4324,394	99
2055	100000000111	4324,41	16

Tabela 8 - Larguras de pulso dos sinais PWM para os respectivos ciclos de trabalho da arquitetura secundária.

Os valores Δc da Tabela 8 apresentaram resultados não uniformes e não confiáveis. A cada incremento no ciclo de trabalho os valores de Δc alternaram, aproximadamente, entre 16 ps e 99 ps. Descobriu-se que essa variação é devido ao roteamento entre os flip-flops $F_1 - F_{2^n}$ e as funções lógicas de elementos de atraso. A solução encontrada foi a implementação da arquitetura principal, no qual, tem-se uma alternância entre LE com utilização do flip-flop e LE sem a utilização do flip-flop (apenas inversor).

A arquitetura mostrada pode ser estudada a fim de obter maior precisão. Com a configuração das funções lógicas é possível obter atrasos de um *carry chain*. Mas a estrutura de flip-flops possui atrasos diferenciados, o que inviabilizou o circuito. O projeto serve para trabalhos futuros, com o objetivo de atingir maior precisão com uma frequência em torno de 100 kHz.

6.3 Conclusões finais

O estudo realizado proporcionou a criação de um projeto de um Modulador de Largura de Pulso Digital – DPWM – de 17 bits, arquitetura apresentada no capítulo 3. Com os resultados apresentados foi possível verificar que uma implementação com 18 bits na FPGA Cyclone IV® é viável. Para tal, a frequência do sinal PWM seria em torno de 50 kHz. Devido aos resultados iniciais não satisfatórios dessa arquitetura alternativa optou-se pela criação de outra arquitetura que atendesse o objetivo, que foi descrita como

a arquitetura principal deste trabalho. Porém, fica como sugestão para trabalhos futuros o estudo da arquitetura alternativa para atingir maior resolução.

Foi utilizada a linguagem VHDL para a descrição do hardware. Apesar da adaptação do projeto em uma FPGA específica, é possível utilizar a estrutura de *carry chains* de outros fabricantes de FPGAs, uma vez que a maioria das FPGAs possui estruturas que implementam elementos de atraso e cujo roteamento direto é permitido.

O projeto da arquitetura de 12 bits mostrou-se impraticável, para a Cyclone IV®, pois apresentou intervalos que poderiam comprometer a monotonicidade do sinal PWM. Porém, este circuito pode servir para trabalhos futuros a fim de se obter maior precisão.

A arquitetura principal apresentou resultados satisfatórios, pois atingiu alta resolução em bits com uma monotonicidade bem definida. O circuito pode sofrer influência com as características de processo das FPGAs, sendo necessário o ajuste da frequência de entrada de acordo com os atrasos dos elementos de atraso. A temperatura também pode influenciar os atrasos de cada elemento, porém, de acordo com algumas simulações, parece não interferir de modo significativo na monotonicidade do sinal. Contudo, é necessário um estudo mais aprofundado desse parâmetro. A síntese do projeto pode ser vista na Figura 83. No total, foram utilizados 792 elementos lógicos e 271 registradores.

Family	Cyclone IV E
Device	EP4CE115F29C7
Timing Models	Final
Total logic elements	792 / 114,480 (< 1 %)
Total combinational functions	790 / 114,480 (< 1 %)
Dedicated logic registers	271 / 114,480 (< 1 %)
Total registers	271
Total pins	20 / 529 (4 %)
Total virtual pins	0
Total memory bits	65,536 / 3,981,312 (2 %)
Embedded Multiplier 9-bit elements	0 / 532 (0 %)
Total PLLs	0 / 4 (0 %)

Figura 83 - Síntese após a compilação no Quartus II®.

O estudo dos blocos de uma FPGA específica proporcionou um aprendizado que permitiu o desenvolvimento de um circuito mais otimizado. A criação de um circuito

com menos blocos extras e a redução dos atrasos de roteamento permitiram obter um controle maior sobre o circuito de uma FPGA.

7 Referências Bibliográficas

- [1] Sabarinath, V.; Sivanandam, K., “Design and implementation of FPGA based high resolution digital pulse width modulator,” *Communications and Signal Processing (ICCSP), 2013 International Conference on*, pp. 410-414, 3-5 April 2013.
- [2] T. Arunachalam, *Digital Pulse Width Modulation Techniques for Power Converters, Trabalho de Mestrado*, Tuscaloosa, Alabama: University of Alabama, 2010, p. 68.
- [3] A. Syed, E. Ahmed, D. Maksimovic e E. Alarcón, “Digital Pulse Width Modulator Architectures,” *Power Electronics Specialists Conference, 2004. PESC 04. 2004 IEEE 35th Annual*, vol. 6, pp. 4689-4695, 2004.
- [4] T. L. Floyd, *Sistemas Digitais Fundamentos e Aplicações*, 9ª ed., Bookman Companhia Editora Ltda, 2007.
- [5] D. F. Bacon, R. Rabbah e S. Shukla, “FPGA Programming for the Masses,” *Acmqueue*, vol. 56, nº 4, pp. 56 - 63, 2013.
- [6] W. Carter, K. Duong, R. H. Freeman, H. Hsieh, J. Y. Ja, J. E. Mahoney, L. T. Ngo e S. L. Sze, “A user programmable reconfiguration gate array,” *PROceedings of the IEEE Custom Integrated Circuits Conference*, pp. 233-235, 1986.
- [7] “Xilinx - FPGA,” Xilinx®, [Online]. Available: <http://www.xilinx.com/fpga/>. [Acesso em 18 Junho 2015].
- [8] I. Kuon, R. Tessier e J. Rose, “FPGA Architecture: Survey and Challenges,” *Foundations and Trends® in Electronic Design Automation*, vol. 2, nº 2, pp. 135-253, 2008.
- [9] A. Corp., “Altera®,” [Online]. Available: <http://www.altera.com>. [Acesso em 2015].
- [10] “Altera® Measurable Advantage™,” [Online]. Available: <https://cloud.altera.com/ds/part/?family=cyclone-iv&device=ep4ce115>. [Acesso em 21 07 2015].
- [11] P. P. Chu, *RTL Hardware Design Using VHDL, Coding for Efficiency, Portability, and Scalability*, Hoboken, NJ: John Wiley & Sons, Inc., 2006.
- [12] V. A. Pedroni, *Circuit Design with VHDL*, Cambridge, MA: MIT Press, 2004.
- [13] D. J. Smith, *HDL Chip Design, A Practical Guide for Designing, Synthesizing and Simulating ASICs and FPGAs using VHDL & Verilog*, Madison, AL: Doone Publications, 1996.
- [14] M. G. Y. Batarseh, *Digital Pulse Width Modulator Techniques for DC-DC Converters*, Orlando, Florida: University of Central Florida, 2006, p. 156.
- [15] G. S. Shirnewar, S. Samanta e S. Gupta, *Digital Pulse Width Modulation Generation Using 8051 for DC DC Buck Converter*, Rourkela: National Institute of Technology, 2013.

- [16] J. Li, Y. Qiu, Y. Sun, B. Huang, M. Xu, D. S. Ha e F. C. Lee, “High Resolution Digital Duty Cycle Modulation Schemes for Voltage Regulators,” *Proc. IEEE 22nd Applied Power Electronics Conf.*, pp. 871-876, 2007.
- [17] A. P. Dancy e A. P. Chandrakasan, “Ultra Low Power Control Circuits for PWM Converters,” *Proc. IEEE PESC Conf.*, pp. 21-27, 1997.
- [18] O. Trescases, G. Wei e W. T. Ng, “A Segmented Digital Pulse Width Modulator with Self-Calibration for Low-Power SMPS,” *IEEE Electron Devices and Solid-State Circuits Conf.*, pp. 367-370, 2005.
- [19] M. Scharrer, M. Halton e T. Scanlan, “FPGA-Based Digital Pulse Width Modulator With Optimized Linearity,” *Proc. IEEE APEC*, pp. 1220-1225, 2009.
- [20] D. Navarro, Ó. Lucía, L. A. Barragán, J. I. Artigas, I. Urriza e Ó. Jiménez, “FPGA-based High Resolution Synchronous Digital Pulse Width Modulator,” *Industrial Electronics (ISIE)*, nº International Symposium on, pp. 2771-2776, 2010.
- [21] L. S. Ge, Z. X. Chen, Z. J. Chen e F. Y. Liu, “Design and Implementation of A High Resolution DPWM Based on A Low-Cost FPGA,” *Proc. IEEE Energ. Conv. Cong. Exp.*, pp. 2306-2311, 2010.
- [22] D. Costinett, M. Rodriguez e D. Maksimovic, “Simple Digital Pulse Width Modulator Under 100 ps Resolution Using General-Purpose FPGAs,” *IEEE Transactions on Power Electronics*, vol. 28, nº 10, pp. 4466-4472, 2013.
- [23] “Application Note: Spartan-3 and Spartan-3L FPGA Families,” em *Using Digital Clock Managers (DCMs) in Spartan-3 FPGAs*, 2006, p. 68.
- [24] “Actel® Power Matters,” em *ProASIC PLUS® Flash Family FPGAs*, 2009, p. 77.
- [25] H. Parandeh-Afshar, P. Brisk e P. Ienne, “Exploiting fast carry-chains of FPGAs for designing compressor trees,” *Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on*, pp. 242-249, 2009.
- [26] A. De Castro e E. Todorovich, “High Resolution FPGA DPWM Based on Variable Clock Phase Shifting,” *Power Electronics, IEEE Transactions on*, vol. 25, nº 5, pp. 1115-1119, 2010.
- [27] I. De León, G. Sotta, G. Eirea e J. P. Acle, “Analysis and Implementation of Low-cost FPGA-Based Digital Pulse-width Modulators,” *Instrumentation and Measurement Technology Conference (I2MTC) Proceedings, 2014 IEEE International*, pp. 1523-1528, 2014.
- [28] Altera®, “The Quartus® II Time Quest Timing Analyzer,” em *Quartus II Handbook Version 13.1*, vol. 3, 2013, p. 60.
- [29] Altera®, “Quartus II Incremental Compilation for Hierarchical and Team-Based Design,” em *Quartus II Handbook Version 13.1*, 2013, p. 64.
- [30] D. Navarro, Ó. Lucía, L. A. Barragán, J. I. Artigas, I. Urriza e Ó. Jiménez, “Synchronous FPGA-Based High-Resolution Implementations of Digital Pulse-Width Modulators,” *IEEE TRANSACTIONS ON POWER ELECTRONICS*, vol. 27, nº 5, pp. 2515-2525, 2012.