People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research University of KASDI MERBAH OUARGLA

Faculty of New Technologies of Information and Communication

Department of Electronic and Telecommunications

Academic master thesis in electronic

Specialty: Electronic of Embedded Systems

Title

Advanced FPGA Implementation and Comparative Investigation of PWM Control Strategies for Multilevel Converters

Presented by:

Dlili Mohammed Taha

Dehikel Ayoub

Presented on the 1^{st} June 2025 Before jury:

President	Mehaouchi Azeddine	MAA (Ouargla university)
Examiner	Melheguegue Nacer	MAA (Ouargla university)
Supervisor	Bouzidi Mansour	MCA (Ouargla university)

Academic Year: 2024-2025

Acknowledgement

We would like to express our deepest gratitude and sincere appreciation to our esteemed supervisor, Dr. BOUZIDI Mansour, for his unwavering support, insightful guidance, valuable advice, and constant encouragement throughout every stage of this project. His dedication and expertise were instrumental in the successful completion of this work.

We also extend our heartfelt thanks to all those who contributed, in any way, to the realization of this thesis. Whether through academic input, moral support, or practical assistance, your contributions have been truly meaningful and are deeply appreciated.

Dedication

With heartfelt gratitude and deep appreciation,

we dedicate this humble work to:

Our beloved mothers and fathers,

whose unwavering support and sacrifices have shaped our journey.

Our cherished sisters, brothers, uncles, aunts, and cousins,

who have surrounded us with love and encouragement.

Our dear friends,

who stood by us through every challenge and joy.

Our inspiring teachers,

from the earliest days of school to the final year of university,

whose wisdom and guidance lit our path.

And to all those with whom we have shared unforgettable moments

your presence has made this journey meaningful.

Contents

List of figures

List of tables

List of symbols

v	Output voltage vector
n	Number of converter level
v_{dc}	DC source voltage
α	Real axis
β	Imaginary axis
v_{ab}, v_{bc}, v_{ca}	Line-to-line voltages
v_{a0}, v_{b0}, v_{c0}	Outputs inverter phase voltages
v_a^*, v_b^*, v_c^*	Reference voltage vector
F_{x0}, F_{x1}, F_{x2}	Switching functions
S_{ix}	Pulses generation
m	Modulation index
v_{ci}	carrier signals
v_m	Amplitude
U_a^*, U_b^*, U_c^*	new reference voltage vector components
U_g^*, U_h^*	new reference voltage vector components in g-h coordinate system
G, H	The integer parts of U_g^* and U_h^*
Δ_1, Δ_2	Upper and lower triangles
i_a, i_b, i_c	The input currents
v_1, v_2, v_3	Switching sequences
T_s	Switching period
t_i	Duration time

List of Abbreviations

PWM	Pulse Width Modulation
DC	Direct Current
IGBT	Insulated Gate Bipolar Transistor
AC	Alternating Current
NPC	Naturel Point Clamped
THD	Total Harmonic Distortion
SVM	Space Vector Modulation
SSVM	Simplified Space Vector Modulation
DAC	Digital-to-Analog Converter
DSP	Digital Signal Processor
FPGA	Field Programmable Gate Array
Pmod	Peripheral Module
USB	Universal Serial Bus
RAM	Random Access Memory
AXI	Advanced eXtensible Interface
GPIO	General Purpose Input/Output
CPU	Central Processing Unit
DMA	Direct Memory Access
ARM	Advanced RISC Machine
ASIC	Application-Specific Integrated Circuits
VHDL	Very-High-speed integrated Hardware Description Language

LUT	Look Up Table
BUFG	Global Clock Buffer
FF	Flip Flop
IO	Input/Output
BRAM	Block Random Access Memory
LUTRAM	Look Up Table Random Access Memory
SoC	System on Chip
HDL	Hardware Description Language
PS	Processing System
PL	Programmable Logic
APOD	Alternative Phase Opposition Disposition
POD	Phase Opposition Disposition
PD	Phase Disposition

General introduction

Multilevel diode-clamped converters (DCCs) have garnered considerable attention in recent years due to their ability to produce high-quality output voltages with significantly reduced harmonic distortion. These attributes make them well-suited for medium- and high-power applications, including renewable energy systems, motor drives, and grid-connected inverters [1-3]. The effectiveness of DCCs, however, is highly influenced by the modulation techniques employed, as these directly impact output voltage quality, switching losses, and overall system efficiency.

Among the widely adopted modulation strategies are Phase Disposition (PD), Phase Opposition Disposition (POD), and Alternative Phase Opposition Disposition (APOD). These methods are favored for their simplicity and implementation ease. Nonetheless, they often struggle to deliver optimal harmonic performance—especially as the number of inverter levels increases [4-6].

To overcome these limitations, Space Vector Modulation (SVM) has emerged as a promising alternative, known for its superior harmonic performance and improved utilization of the DC-link voltage [7]. Despite its advantages, conventional SVM techniques involve high computational complexity, which poses challenges for real-time applications. To mitigate this, several Simplified SVM (SSVM) algorithms have been introduced, aiming to reduce computational load and generalize the modulation scheme for converters of varying levels [8], [9].

Recent advancements in Field-Programmable Gate Array (FPGA) technology—particularly using platforms such as the Xilinx Zynq-7000—have opened up new

opportunities for efficiently implementing complex modulation schemes. These devices combine Programmable Logic (PL) with a Processing System (PS), enabling parallel processing and efficient execution of computationally intensive tasks. This architecture supports the integration of advanced control algorithms with improved flexibility and performance [10-12].

In this work, we present a novel FPGA-based implementation of an SSVM technique tailored for multilevel DCCs. The proposed approach exploits the capabilities of both the PL and PS components of the FPGA: the PL section, programmed in VHDL, is responsible for generating the three-phase reference voltages, while the PS section, programmed in C, performs the SSVM calculations. This hybrid design streamlines the development process and significantly boosts computational efficiency, making it highly suitable for real-time applications.

The proposed SSVM method has been implemented and evaluated for 3-level, 5-level, 7-level, and 9-level DCCs. Its performance is benchmarked against traditional modulation schemes such as PD, POD, and APOD. Experimental results confirm that the SSVM approach delivers notably improved harmonic performance, especially in higher-level configurations, underscoring its potential to enhance output voltage quality in modern multilevel converter systems. The contributions of this work can be summarized as follows:

In order to achieve the thesis objective, this thesis is divided into three chapters, which are summarized as follows:

The first chapter is dedicated to presenting the operating principles and mathematical modeling of n-level DCC.

The second chapter presents a comprehensive study on the simulation and performance evaluation of various modulation techniques for DCC, including PD, POD, APOD, and the SSVM technique.

The third chapter is reserved to present the experimental results for the FPGA implementation of the presented PWM Techniques PD, POD, APOD and Simplified Space Vector Modulation.

Finally, a conclusion of this work is provided to summarize the main findings and highlight the significance of the presented approaches.

General introduction

Multilevel diode-clamped converters (DCCs) have garnered considerable attention in recent years due to their ability to produce high-quality output voltages with significantly reduced harmonic distortion. These attributes make them well-suited for medium- and high-power applications, including renewable energy systems, motor drives, and grid-connected inverters [1-3]. The effectiveness of DCCs, however, is highly influenced by the modulation techniques employed, as these directly impact output voltage quality, switching losses, and overall system efficiency.

Among the widely adopted modulation strategies are Phase Disposition (PD), Phase Opposition Disposition (POD), and Alternative Phase Opposition Disposition (APOD). These methods are favored for their simplicity and implementation ease. Nonetheless, they often struggle to deliver optimal harmonic performance—especially as the number of inverter levels increases [4-6].

To overcome these limitations, Space Vector Modulation (SVM) has emerged as a promising alternative, known for its superior harmonic performance and improved utilization of the DC-link voltage [7]. Despite its advantages, conventional SVM techniques involve high computational complexity, which poses challenges for real-time applications. To mitigate this, several Simplified SVM (SSVM) algorithms have been introduced, aiming to reduce computational load and generalize the modulation scheme for converters of varying levels [8], [9].

Recent advancements in Field-Programmable Gate Array (FPGA) technology—particularly using platforms such as the Xilinx Zynq-7000—have opened up new

opportunities for efficiently implementing complex modulation schemes. These devices combine Programmable Logic (PL) with a Processing System (PS), enabling parallel processing and efficient execution of computationally intensive tasks. This architecture supports the integration of advanced control algorithms with improved flexibility and performance [10-12].

In this work, we present a novel FPGA-based implementation of an SSVM technique tailored for multilevel DCCs. The proposed approach exploits the capabilities of both the PL and PS components of the FPGA: the PL section, programmed in VHDL, is responsible for generating the three-phase reference voltages, while the PS section, programmed in C, performs the SSVM calculations. This hybrid design streamlines the development process and significantly boosts computational efficiency, making it highly suitable for real-time applications.

The proposed SSVM method has been implemented and evaluated for 3-level, 5-level, 7-level, and 9-level DCCs. Its performance is benchmarked against traditional modulation schemes such as PD, POD, and APOD. Experimental results confirm that the SSVM approach delivers notably improved harmonic performance, especially in higher-level configurations, underscoring its potential to enhance output voltage quality in modern multilevel converter systems. The contributions of this work can be summarized as follows:

In order to achieve the thesis objective, this thesis is divided into three chapters, which are summarized as follows:

The first chapter is dedicated to presenting the operating principles and mathematical modeling of n-level DCC.

The second chapter presents a comprehensive study on the simulation and performance evaluation of various modulation techniques for DCC, including PD, POD, APOD, and the SSVM technique.

The third chapter is reserved to present the experimental results for the FPGA implementation of the presented PWM Techniques PD, POD, APOD and Simplified Space Vector Modulation.

Finally, a conclusion of this work is provided to summarize the main findings and highlight the significance of the presented approaches.

Chapter I:

Modeling of the three-phase multilevel diode clamped converter

I.1. Introduction

Recent technological advances in power electronics have expanded the use of static converters in demanding applications such as active filtering, grid decontamination, and electric traction, which require high dynamic performance. Despite improvements in switch technology, their blocking voltage remains below 10 kV, insufficient for high-voltage applications [1]-[3]. Two main solutions address this limitation:

- Using series-connected switches (macro-switches),
- Adopting multilevel inverters.

Multilevel inverters offer several advantages, including reduced voltage stress on switches, improved output voltage quality, and smaller passive filters. However, these benefits come with increased complexity, control challenges, and reduced reliability due to the higher number of components [1]-[3].

Among the three main multilevel inverter topologies—diode-clamped (also known as Neutral Point Clamped, NPC), capacitor-clamped (flying capacitor), and cascaded H-bridge—the diode-clamped inverter (DCC) is often preferred due to its relatively simple and robust structure [1]-[3]. Unlike the capacitor-clamped topology, which requires multiple floating capacitors for each phase leg and careful voltage balancing to avoid parasitic resonances, the DCC achieves voltage level generation using a common DC bus and clamping diodes, reducing the complexity of energy storage components [1]-[3]. Compared to the cascaded H-bridge topology, which demands multiple isolated DC sources and has a modular but bulky architecture, the DCC only needs a single DC source with fewer redundancy requirements [1]-[3]. Additionally, the diode-clamped topology benefits from better fault tolerance, improved efficiency at high voltage levels, and more straightforward implementation of pulse width modulation (PWM) strategies, making it particularly suitable for medium and high-power applications such as motor drives, renewable energy systems, and high-voltage grid integration [1]-[3].

This chapter focuses on the detailed modeling of the diode-clamped multilevel inverter, presenting and thoroughly analyzing its switching states, corresponding output voltage levels, and the associated space vector diagram.

I.2. *n*-Level Diode-Clamped Inverter

I.2.1. Operating Principles

The structure of the DCC was introduced by A.Nabae and H. Akagi in 1981 [1]. Figure (I.1) presents the *n*-level three-phase multilevel diode clamped inverter, where a series of (n-1) capacitors is used to create (n-2) midpoint capacitive nodes. Each capacitor must be sized for a voltage of $v_{dc}/(n\text{-}1)$, and each inverter leg consists of $2n$-2 fully controllable switches. These switches must not be opened or closed simultaneously to prevent a short circuit of the DC input source or the disconnection of the inductive load circuit. For this reason, the switch pairs of the lower half-bridge are complementary to those of the upper half-bridge. The voltage distribution v_{dc} across the various series-connected switches is ensured by floating diodes ($2n$-4 per leg), which are connected to the capacitive midpoint nodes. Intermediate voltage levels on the output voltage of the leg can thus be created by connecting each of these nodes to the output, as illustrated in the functional diagram of the *n*-level inverter in Fig I.2. This allows for the generation of n distinct voltage levels, summarized in Table I.1, considering that the voltage of node 0 is used as the reference. All other states are undefined and prohibited [1]-[3].

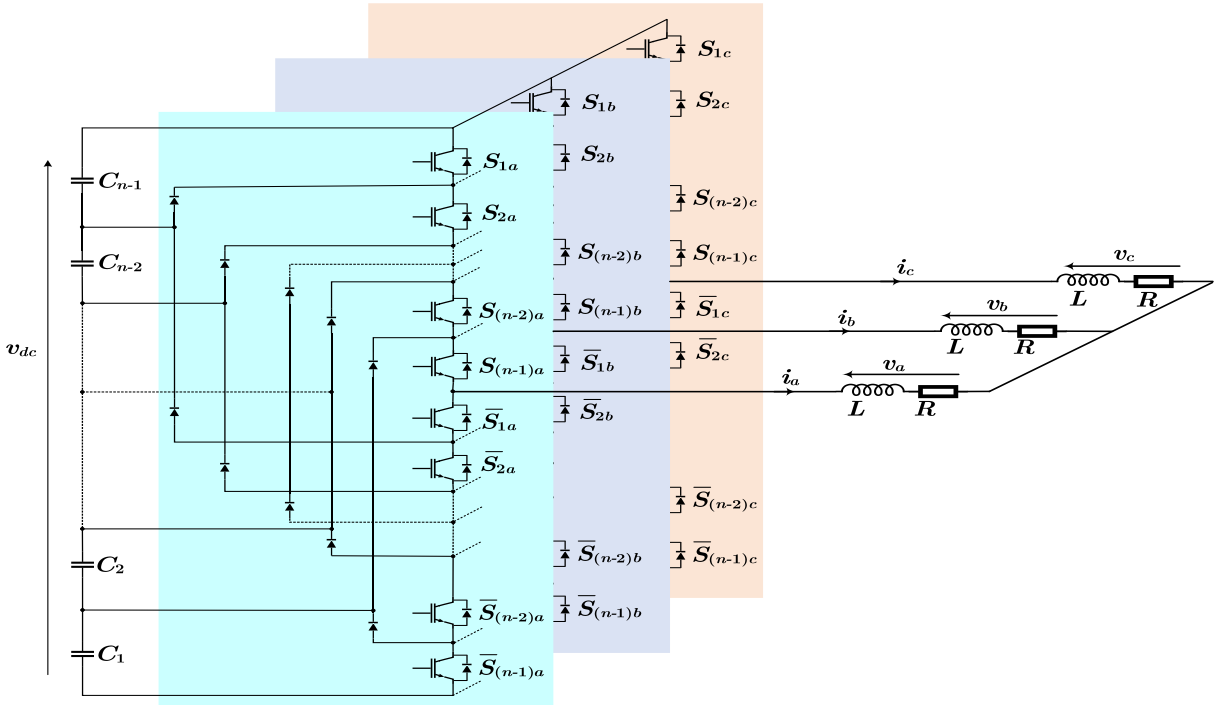


Fig I.1: Main circuit of the three-phase ***n***-level DCC

Table I.1: States of one leg of an ***n***-level DCC ($x = a, b, or\ c$)

State of the leg	*Switching States of the Leg*										*Output voltage*
	S_{x1}	S_{x2}	....	$S_{x(n-2)}$	$S_{x(n-1)}$	$\overline{S}_{x1}$	$\overline{S}_{x2}$	....	$\overline{S}_{x(n-2)}$	$\overline{S}_{x(n-1)}$	v_{x0}
n-1	1	1	....	1	1	0	0	....	0	0	v_{dc}
n-2	0	1	....	1	1	1	0	....	0	0	$\frac{(n\text{-}2)\ v_{dc}}{n\text{-}1}$
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
1	0	0	....	0	1	1	1	....	1	0	$\frac{v_{dc}}{n\text{-}1}$
0	0	0	....	0	0	1	1	....	1	1	0

I.2.2. Modeling of the *n*-level DCC

I.2.2.1. Connection Functions

For each leg of the inverter, $\boldsymbol{n}$ connection functions are defined (see Fig I.2), each associated with one of the $\boldsymbol{n}$ states of the leg, as follows:

$$
\begin{aligned}
F_{x(n-1)} &= S_{x(n-1)}\ S_{x(n-2)}\ S_{x(n-3)} \cdots S_{x2}\ S_{x1}\ S_{x0} \\
F_{x(n-2)} &= S_{x(n-1)}\ S_{x(n-2)}\ S_{x(n-3)} \cdots S_{x2}\ S_{x1}\ \overline{S}_{x0} \\
F_{x(n-3)} &= S_{x(n-1)}\ S_{x(n-2)}\ S_{x(n-3)} \cdots S_{x2}\ \overline{S}_{x1}\ \overline{S}_{x0} \\
&\vdots \qquad\qquad \vdots \qquad\qquad \vdots \qquad\qquad x = a, b, \text{ or } c \\
F_{x2} &= S_{x(n-1)}\ S_{x(n-2)}\ \overline{S}_{x(n-3)} \cdots \overline{S}_{x2}\ \overline{S}_{x1}\ \overline{S}_{x0} \\
F_{x1} &= S_{x(n-1)}\ \overline{S}_{x(n-2)}\ \overline{S}_{x(n-3)} \cdots \overline{S}_{x2}\ \overline{S}_{x1}\ \overline{S}_{x0} \\
F_{x0} &= \overline{S}_{x(n-1)}\ \overline{S}_{x(n-2)}\ \overline{S}_{x(n-3)} \cdots \overline{S}_{x2}\ \overline{S}_{x1}\ \overline{S}_{x0}
\end{aligned}
\qquad \text{(I.1)}
$$

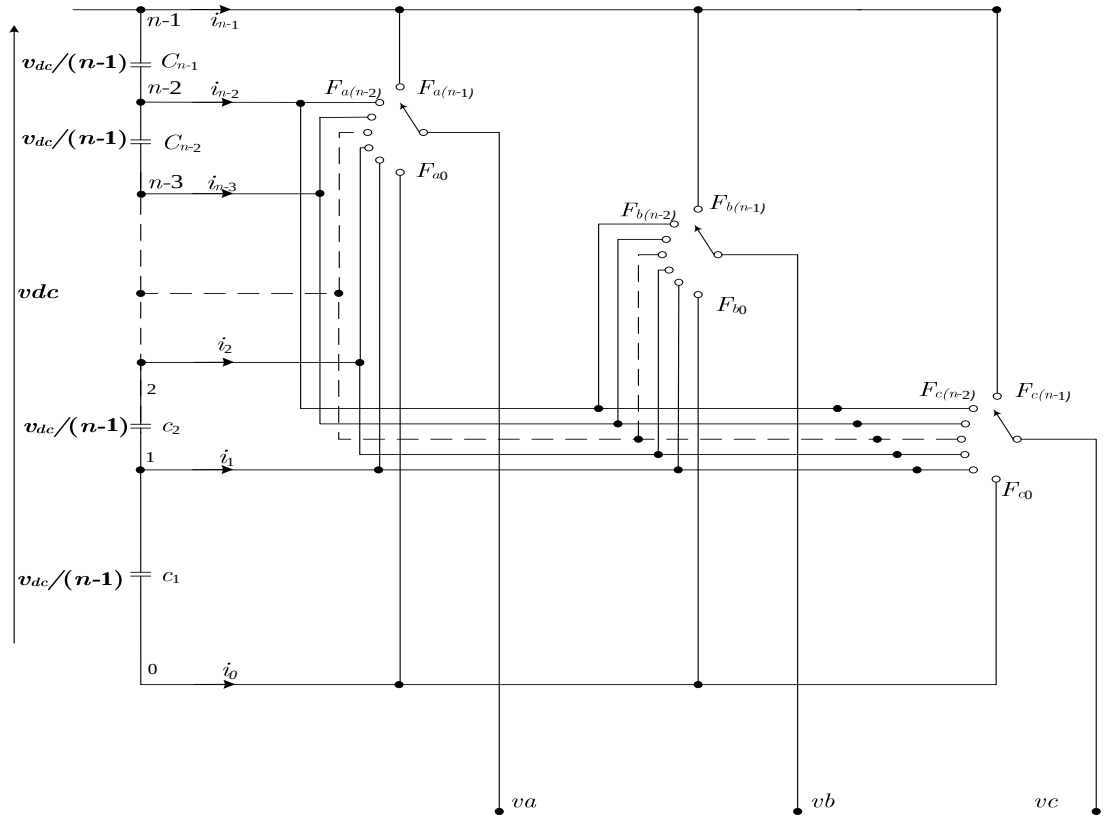


Fig I.2: Functional diagram of the *n*-level DCC

I.2.2.2. Output voltage

The three-phase voltages of the ***n***-level DCC with respect to point 0 (see Fig I.1) are expressed as:

$$v_{x0} = \sum_{j=0}^{n-1}\left(F_{xj} \sum_{i=0}^{j} v_{Ci} \right), \qquad x = a,\ b,\ c \tag{I.2}$$

Where:

$v_{Ci}\,(i = 0, 1, \ldots, n\text{-}1)$: DC-link capacitor voltages.

Ideally, when the capacitor voltages are balanced (equal), equation (I.2) simplifies to:

$$v_{x0} = \frac{v_{dc}}{n-1} \sum_{j=0}^{n-1} jF_{xj}\ , \qquad x = a,\ b,\ c \tag{I.3}$$

Equation (I.3) can be written in a matrix form as follows:

$$\begin{bmatrix} v_{a0} \\ v_{b0} \\ v_{c0} \end{bmatrix} = \begin{bmatrix} F_{a0} & F_{a1} & .... & F_{a(n-2)} & F_{a(n-1)} \\ F_{b0} & F_{b1} & .... & F_{b(n-2)} & F_{b(n-1)} \\ F_{c0} & F_{c1} & .... & F_{c(n-2)} & F_{c(n-1)} \end{bmatrix} \begin{bmatrix} 0 \\ \dfrac{v_{dc}}{n-1} \\ \vdots \\ \dfrac{(n-2)v_{dc}}{n-1} \\ v_{dc} \end{bmatrix} \tag{I.4}$$

Each of these voltages can have ***n*** voltage levels: $0,\ v_{dc}/(n-1),\ \ldots\ ,\ v_{dc}$, Which is the origin of the term: ***n***-level inverter.

The line-to-line voltages between the inverter legs are:

$$\begin{bmatrix} v_{ab} \\ v_{bc} \\ v_{ca} \end{bmatrix} = \begin{bmatrix} v_{a0} - v_{b0} \\ v_{b0} - v_{c0} \\ v_{c0} - v_{a0} \end{bmatrix} \tag{I.5}$$

The output phase voltages in the case of a balanced load, are expressed by:

$$\begin{bmatrix} v_a \\ v_b \\ v_c \end{bmatrix} = \frac{1}{3}\begin{bmatrix} v_{ab} - v_{ca} \\ v_{bc} - v_{ab} \\ v_{ca} - v_{bc} \end{bmatrix} \tag{I.6}$$

The input currents of the inverter at the DC side are expressed as a function of the load currents i_a, i_b, and i_c by:

$$i_k = F_{ak} i_a + F_{bk} i_b + F_{ck} i_c, \qquad k = 0,1,...n-1 \tag{I.7}$$

I.2.2.3. Space vector diagram of the *n*-level DCC

Since each phase of the inverter has n distinct switching states, the n-level DCC can achieve n^3 possible states, determined by the combination of the states of its three legs (i,j,k) where $i,j,k \in [0,1,...,n-1]$ .These states define the appropriate configuration of the n-pole switches for the three phases (Fig I.2) [5]. For instance, the state $(n-1, n-2, 0)$ indicates that the selector of the first leg is at the position $(n-1)$, the second in position $(n-2)$, and the third in the position 0, meaning that the corresponding switches are activated as follows: $F_{a(n-1)} = 1$, $F_{b(n-2)} = 1$ et $F_{c0} = 1$.

The output voltage vector v is defined by:

$$\vec{v} = v_a e^{j0} + v_b e^{-j2\pi/3} + v_c e^{-j4\pi/3} \tag{I.8}$$

The transformation from the three-phase plane abc to the stationary α-β plane is achieved using the following matrix transformation:

$$\begin{bmatrix} v_\alpha \\ v_\beta \end{bmatrix} = \sqrt{\frac{2}{3}}\begin{bmatrix} 1 & -1/2 & -1/2 \\ 0 & \sqrt{3}/2 & -\sqrt{3}/2 \end{bmatrix}\begin{bmatrix} v_a \\ v_b \\ v_c \end{bmatrix} \tag{I.9}$$

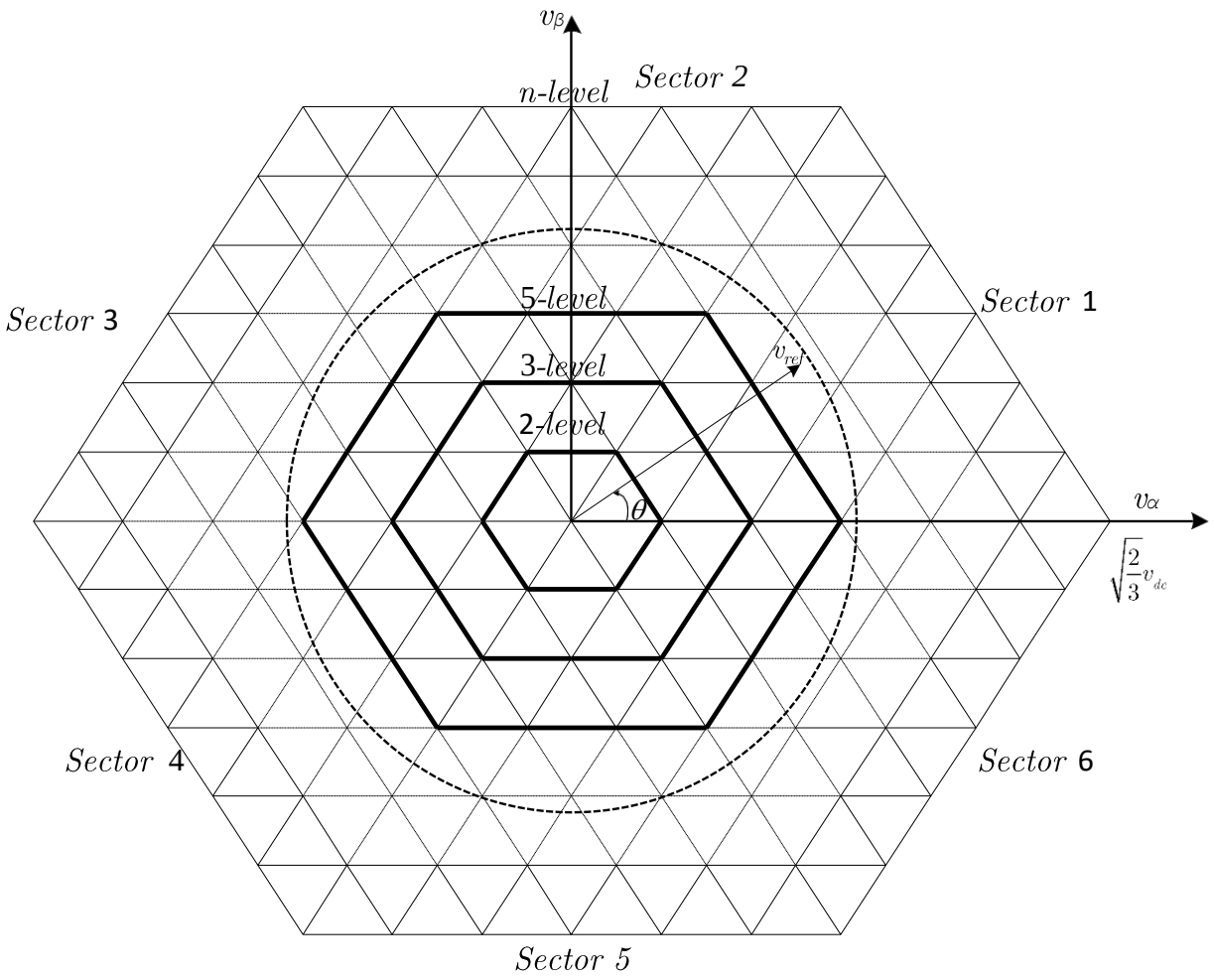


Fig I.3: Space vector diagram of the n-level DCC

In the stationary $\alpha-\beta$ reference frame, the output voltage vector $\vec{v}$ is expressed as:

$$\vec{v} = v_\alpha + jv_\beta \tag{I.10}$$

Based on the inverter states, the output voltage vector can assume multiple positions within the α-β reference frame, distributed across $(n-1)$ hexagons centered at the origin, as illustrated in Fig I.3. Each position in the diagram (excluding the outermost hexagon) can be generated by multiple distinct states, known as redundant states (see Figs 4, 5, 6, 7 and 8), while the corresponding vectors are referred to as redundant vectors. The discrete voltage vector positions divide the space vector diagram into six triangular sectors, with their vertices converging at the origin. Each sector contains $(n-1)^2$ triangular regions, leading to a total of 6 $(n-1)^2$ triangle regions in the complete space vector diagram [15].

As the number of voltage levels in a multilevel converter increases, the space vector diagram in the αβ plane becomes progressively denser and more refined, significantly improving the quality of the synthesized output voltage.

- As shown in Fig. I.4, the space vector diagram of the 2-level converter is composed of six active vectors forming a regular hexagon, with two zero vectors at the center. The output voltage transitions are coarse, which results in higher harmonic distortion.
- With a 3-level inverter, the diagram becomes more populated (see Fig. I.5). Additional intermediate vectors appear between the primary directions of the 2-level inverter, forming multiple smaller hexagons nested within the original. This increased resolution allows better approximation of a sinusoidal waveform and reduces harmonics.
- As we move to 5-level, 7-level, and 9-level inverters (see Figs I.6, I.7 and I.8), the number of possible switching states increases exponentially, leading to a highly granular and nearly circular vector space within the boundaries of the original hexagon. Each additional level introduces more intermediate voltage vectors, resulting in:
    - Smoother voltage transitions .
    - Lower total harmonic distortion (THD).
    - Improve spectral quality.
    - Reduced need for bulky output filters.

In the space vector diagrams for the 5-level, 7-level, and 9-level inverters, we have not included all the redundant states, as doing so would clutter the diagrams and reduce clarity. Instead, only the first two redundant states for each vector position are shown. The remaining redundant states can be deduced by successively subtracting 1 from the displayed switching states until at least one of the phase indices reaches zero.

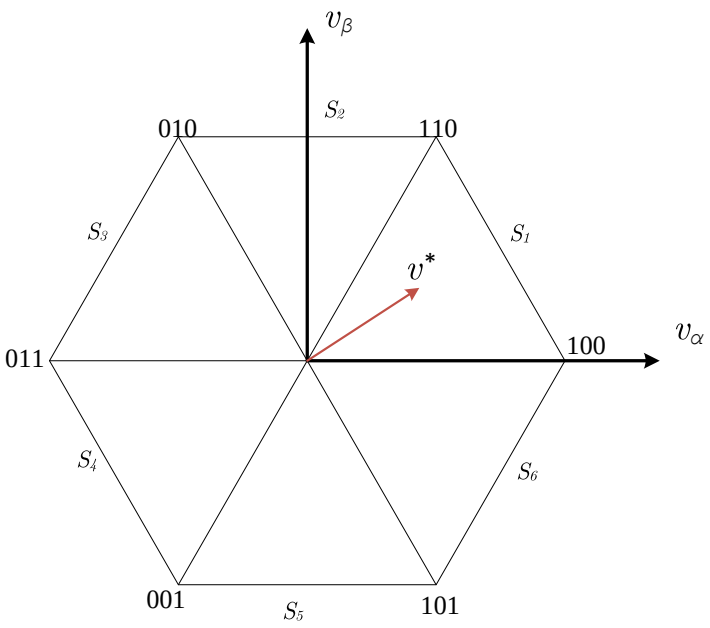


Fig I.4: Space vector diagram for 2-level converter

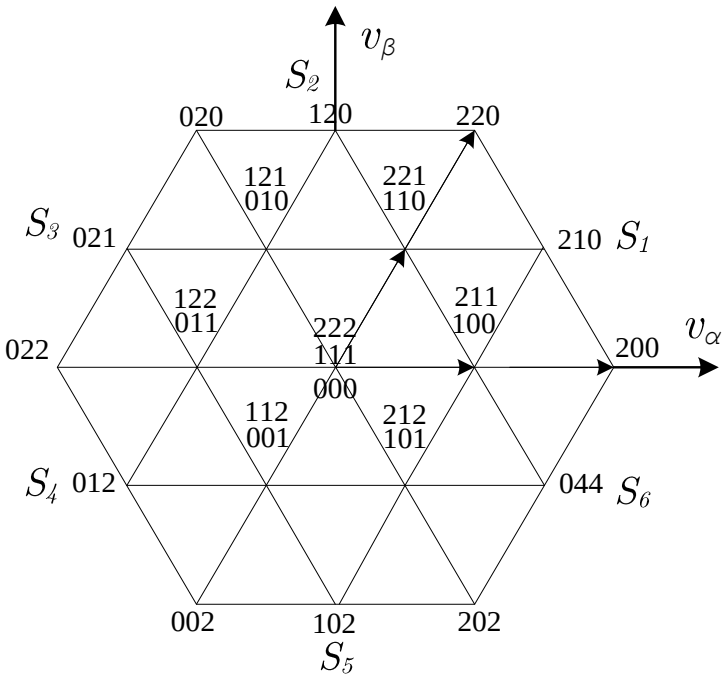


Fig I.5: Space vector diagram for 3-level converter

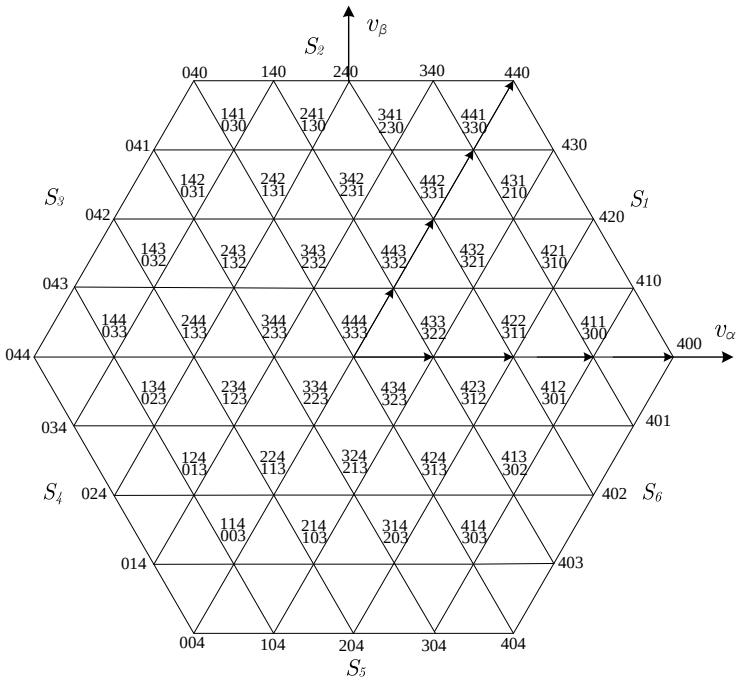


Fig I.6: Space vector diagram for 5-level converter

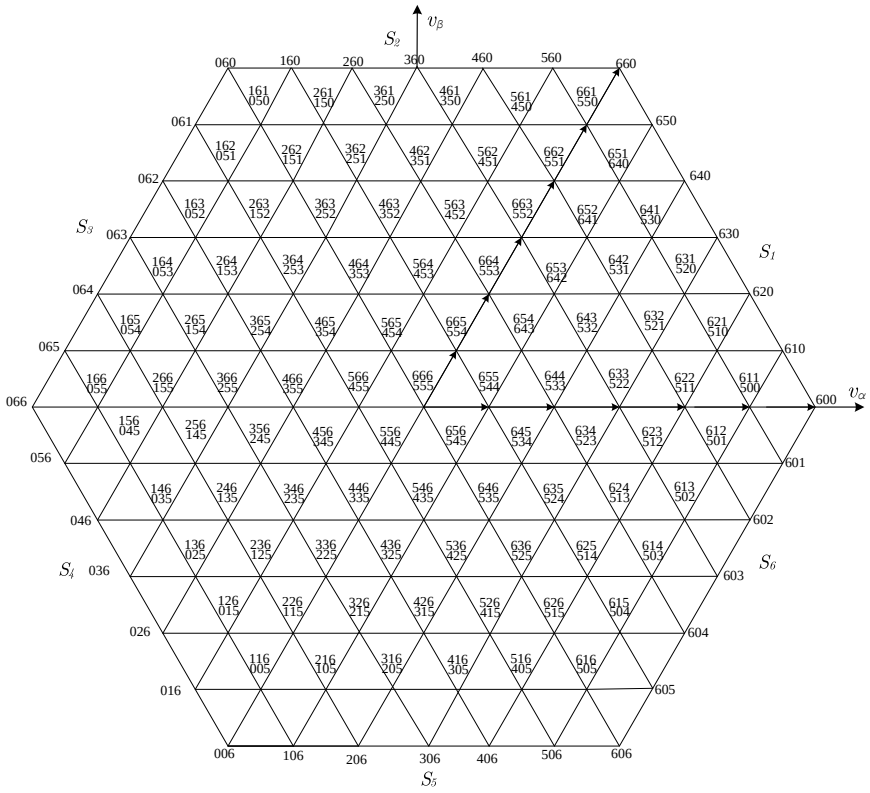


Fig I.7: Space vector diagram for 7-level converter

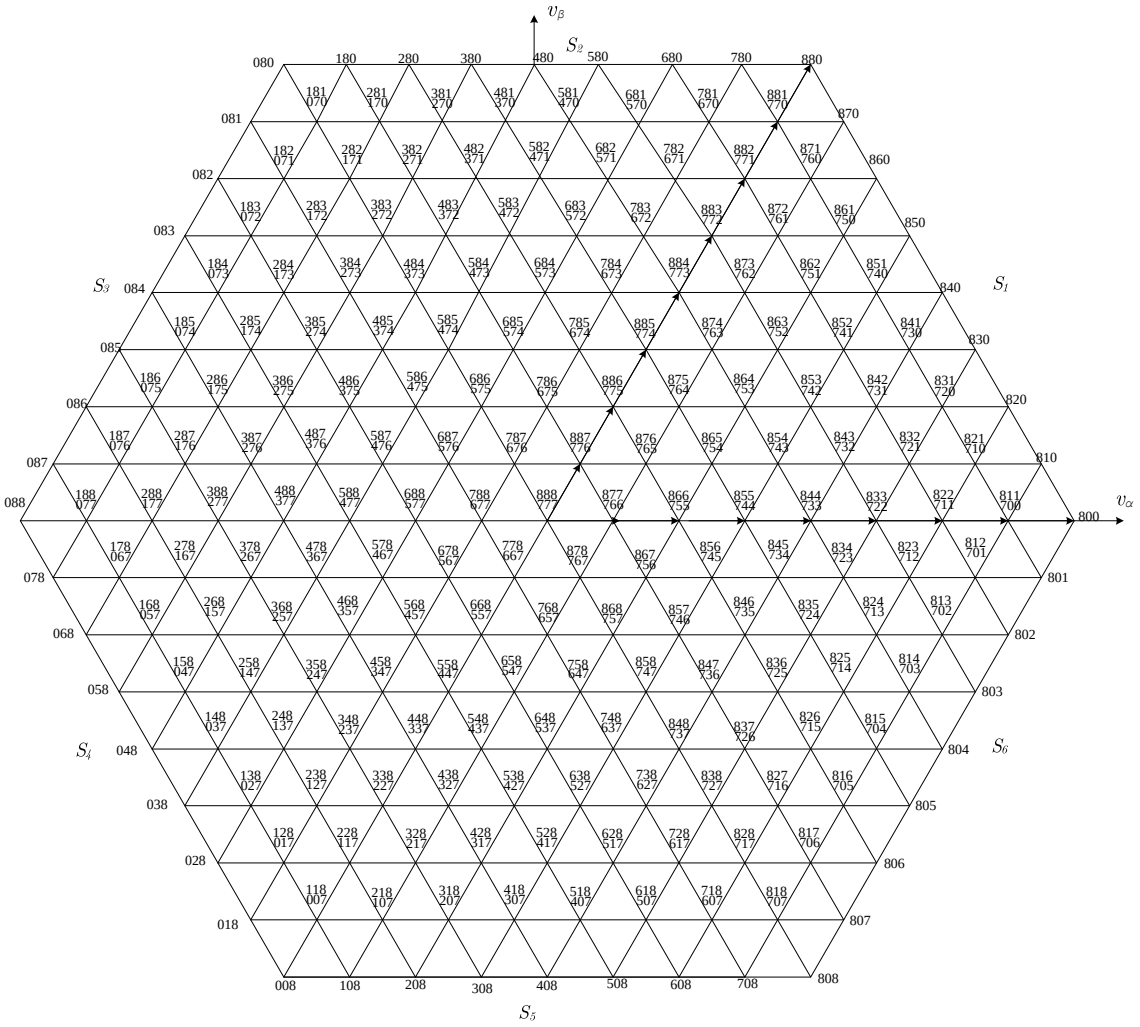


Fig I.8: Space vector diagram for 9-level converter

I.3. Conclusion

In this chapter, we developed a comprehensive model of the multilevel Diode-Clamped Converter (DCC). A general formulation for an *n*-level DCC was presented, detailing the possible switching states, associated output voltages, and the corresponding space vector diagram in the $\alpha\beta$ plane. Additionally, specific examples for 2-level, 3-level, 5-level, 7-level, and 9-level inverters were provided to illustrate the increasing complexity and vector resolution as the number of levels rises.

The next chapter will be dedicated to the synthesis and simulation of various Pulse Width Modulation (PWM) techniques specifically designed for multilevel Diode-Clamped Converters (DCC). It will explore and compare different modulation strategies, highlighting their operating principles, implementation aspects, and suitability for different converter levels.

Chapter II:

PWM techniques for multilevel converters

II.1. Introduction

Multilevel diode-clamped converters (DCCs) have gained significant attention in recent years due to their ability to generate high-quality output voltages with reduced harmonic distortion, making them ideal for medium- and high-power applications such as renewable energy systems, motor drives, and grid-connected inverters [4], [5]. However, the performance of these converters heavily depends on the modulation techniques employed, which directly influence the output voltage quality, switching losses, and overall system efficiency. Among the various modulation strategies, Phase Disposition (PD), Phase Opposition Disposition (POD), and Alternative Phase Opposition Disposition (APOD)

are widely used due to their simplicity and effectiveness. However, these techniques often face challenges in achieving optimal harmonic performance, particularly as the number of converter levels increases [6].
To address these challenges, Space Vector Modulation (SVM) has emerged as a promising alternative, offering superior harmonic performance and better utilization of the DC-link voltage [7]. However, the high computational complexity of traditional SVM techniques often restricts their practical implementation, particularly in real-time systems. To address this limitation, several simplified SVM (SSVM) algorithms have been proposed in the literature, aiming to reduce complexity and generalize the SSVM approach for converters of any level [8], [9].

In this chapter, carrier-based PWM techniques—including Phase Disposition (PD), Phase Opposition Disposition (POD), and Alternative Phase Opposition Disposition (APOD)—as well as the simplified Space Vector Modulation (SVM) method presented in [9], have been synthesized and simulated using the MATLAB/Simulink environment for multilevel converters.

II.2. Carrier based PWM techniques for multilevel DCC

II.2.1. PD modulation technique

In carrier-based PWM techniques, the *n*-level multilevel DCC requires *n*−1 carrier signals to properly synthesize the output voltage waveform. Fig II.1 illustrates the three-phase voltage references (v_a^*, v_b^*, v_c^*) and carrier waveforms for a 5-level DCC operating in PD technique. In the PD modulation technique, all carrier waveforms have identical frequency and amplitude, ensuring uniform voltage step transitions. Furthermore, the *n*−1 carriers ($v_{c1}, v_{c2}, ..., v_{c(n-1)}$) are aligned in phase with each other, meaning they share the same zero-crossing points and maintain a fixed phase relationship throughout the modulation process (see Fig II.1). This configuration helps maintain symmetry in the generated output waveform and reduces harmonic distortion by ensuring effective voltage level transitions [4]-[6].

The pulses for the *n*-level converter shown in Fig. II.2 are generated using the PD modulation technique, based on the comparison defined in Equation (II.1) as follows:

$$S_{ix} = 1, \text{ when: } v_x^* > v_{ci}, \quad \text{else } S_{ix} = 0 \tag{II.1}$$

where: $x = a,\ b,$ or c, j=1, 2, …, n-1.

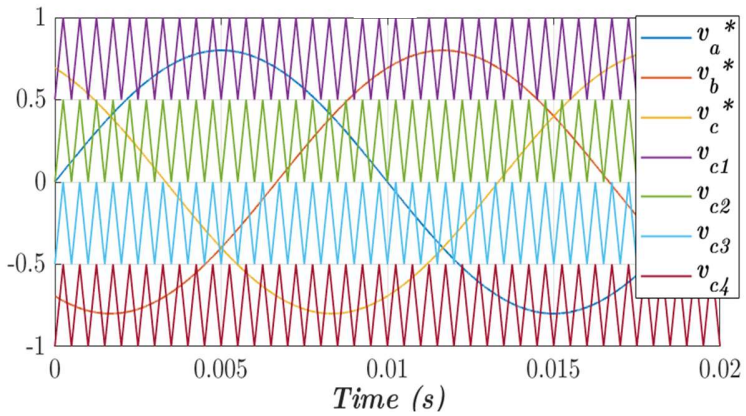


Fig II.1: Reference voltages and carrier waveforms of the 5-level DCC for the PD modulation technique

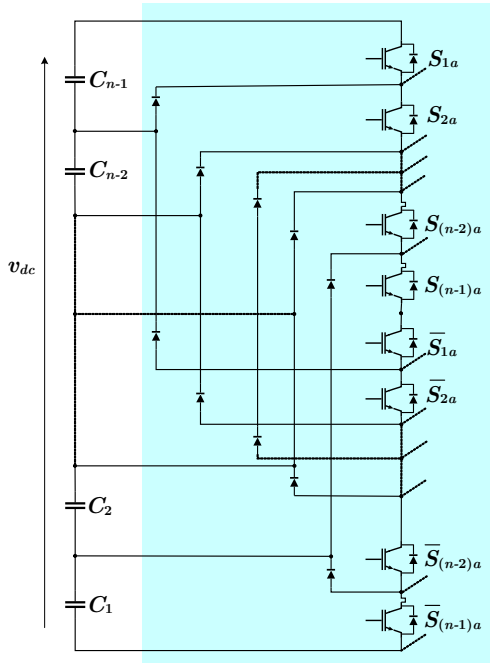


Fig II.2: Leg-a of the multilevel DCC

For better clarity, the 5-level DCC is taken as an example. The phase-a pulses are generated using the PD modulation technique, based on the condition given in equation (II.2):

$$\begin{aligned} S_{1a} &= 1, \text{ when: } v_a^* > v_{c1}, \quad \text{else } S_{1a} = 0 \\ S_{2a} &= 1, \text{ when: } v_a^* > v_{c2}, \quad \text{else } S_{2a} = 0 \\ S_{3a} &= 1, \text{ when: } v_a^* > v_{c3}, \quad \text{else } S_{3a} = 0 \\ S_{4a} &= 1, \text{ when: } v_a^* > v_{c4}, \quad \text{else } S_{4a} = 0 \end{aligned} \tag{II.2}$$

I.2.2. POD modulation technique

As the PD technique, for an (n-1)-level inverter, there are (n-1) carrier signals that are compared with three-phase reference voltages. As shown in Fig II.3 for 5-level DCC, the POD technique arranges these carrier signals so that [4]-[6]:

- Carriers above the zero-reference axis are in phase with each other.
- Carriers below the zero-reference axis are in phase with each other but 180° out of phase with the carriers above the zero axis.

This means that all carriers are shifted in opposite polarity across the zero-axis, ensuring phase opposition between the upper and lower halves of the carriers [4]-[6].

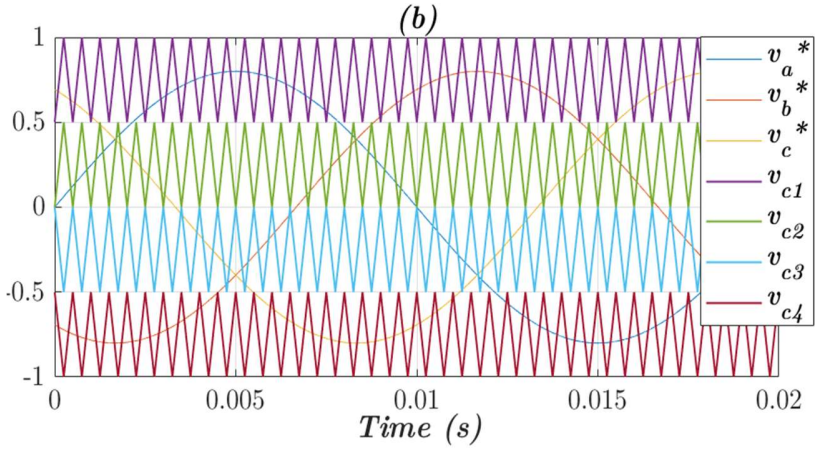


Fig II.3: Reference voltages and carrier waveforms of the 5-level DCC for the POD modulation technique

The phase-a pulses for the 5-level DCC are generated using the POD modulation technique, based on the same condition as given in (II.2).

I.2.2. APOD modulation technique

For an n-level DCC, there are (n-1) carrier signals, in APOD (see Fig. II.4 for 5-level DCC):

- Each adjacent carrier is phase-shifted by 180° relative to its neighbor.
- This means that no two consecutive carriers have the same phase, leading to an alternating phase arrangement [4]-[6].

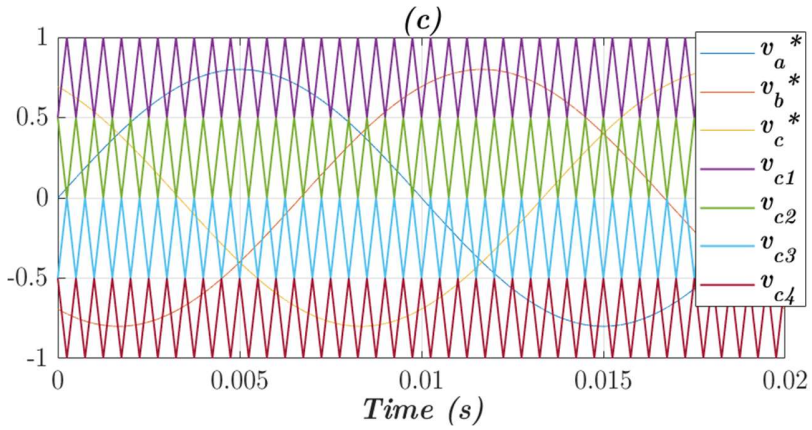


Fig II.4: Reference voltages and carrier waveforms of the 5-level DCC for the APOD modulation technique

The phase-a pulses for the 5-level DCC are generated using the POD modulation technique, based on the same condition as given in (II.2).

II.3. Space vector modulation

II.3.1. Conventional SVM Algorithm for *n*-level DCC

As we stated in chapter II, the *n*-level DCC offers n^3 possible switching combinations, which can be arranged into *n*-1 centered hexagons within the $\alpha\beta$ plane, as illustrated in Fig II.5.

The space vector diagram is divided into six sectors, with each sector further subdivided into $(n\text{-}1)^2$ triangular regions. The three-phase reference voltages in $\alpha\beta$ plane can be represented by one reference voltage vector $\vec{v}^*$ which rotates in the space vector diagram of Fig II.5 and cross all sectors [9].

To build the reference voltage vector using the conventional SVM The reference voltage vector v^* is approximated by the time-averaged effect of the nearest switching voltage vectors within each switching period. To achieve this, the first step is to determine the sector and triangle in which the reference voltage vector is located [9].

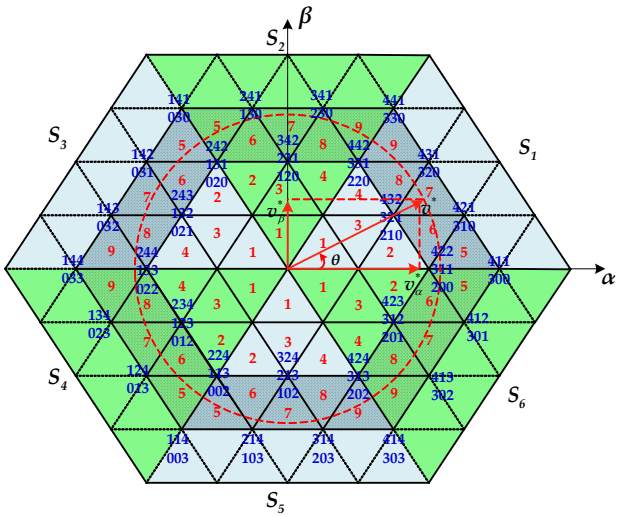


Fig II.5: Space vector diagram of *n*-level DCC including the reference voltage vector

Once the corresponding triangle is identified, the time intervals for the adjacent switching vectors are computed using the following equation:

$$\begin{aligned} v_1 t_1 + v_2 t_2 + v_3 t_3 &= T_s v^* \\ t_1 + t_2 + t_3 &= T_s \end{aligned} \tag{II.3}$$

Here, T_s represents the switching period, while v_1, v_2 and v_3 are the switching vectors closest to the reference voltage vector v^*. Their corresponding time intervals are denoted as t_1, t_2 and t_3 respectively [9].

The conventional SVM algorithm for an *n*-level DCC follows three key steps:

1. Reference voltage vector location: identify the specific triangle containing the reference voltage vector among the $6(n-1)^2$ possible triangles across all sectors.
2. Time intervals calculation of for adjacent switching sectors: solve $6(n-1)^2$ equations similar to equation (II.3) to determine the duration of each switching vector.
3. Design of switching sequences: develop appropriate switching sequences for all $6(n-1)^2$ triangles to ensure smooth operation.

Clearly, the computational complexity of the conventional SVM algorithm grows exponentially with the increase in the number of levels in the DCC [9].

II.3.2. Simplified SVM algorithm for *n*-level DCC

The core concept of the simplified SVM (SSVM) algorithm is to execute the conventional SVM steps within a single sector, significantly reducing computational complexity and arithmetic load.

As illustrated in Fig II.5, the reference voltage vector v^* rotates counterclockwise, traversing all sectors of the space vector diagram. To simplify calculations, the SSVM algorithm introduces a new reference voltage vector U^* which remains confined to the first sector while encapsulating all necessary information from v^* as it moves across other sectors [9].

II.3.2.1. *New Reference Voltage Vector*

The new vector U^* is established based on the old vector v^*. As given in [9], each triangle of the first sector has an equivalent triangle in the other sectors. This equivalence resides in the possibility of changing the phase orders of the switching states in the first sector to deduce all the switching states in the other sectors. As shown in Fig II.5, each triangle in the other sectors takes the same number of its equivalent triangle in the first sector [9].

Consider the case where the reference voltage vector v^* traverses through triangles 5, 6, 7, 8, and 9 as it moves from sector 1 to sector 6 (see Fig II.5). If v^* originates from triangle 5 in the first sector, it follows the sequence $5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9$ within the first sector. This same sequence is maintained when v^* moves through the odd-numbered sectors (3 and 5). However, in the even-numbered sectors (2, 4, and 6), v^* follows a reverse order, passing through triangles $9 \rightarrow 8 \rightarrow 7 \rightarrow 6 \rightarrow 5$ [9].

Although the new reference voltage vector (U^*) is constrained to rotate solely within the first sector, its rotational direction must still correspond to the position of the original v^*. To ensure that U^* traverses the triangles in the same order as v^* across all sectors, its direction of rotation should be adjusted accordingly (see Fig II.6) [9].

- Counterclockwise rotation when v^* is located in odd-numbered sectors (1, 3, and5).
- Clockwise rotation when v^* is in even-numbered sectors (2, 4, and 6).

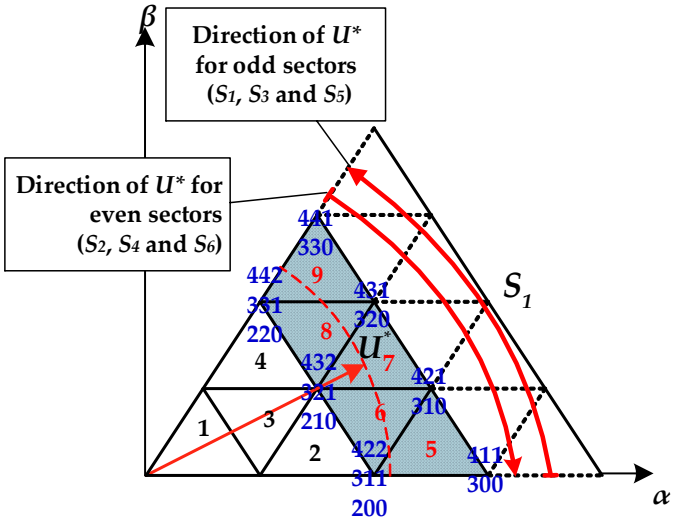


Fig II.6: Direction of the new vector U^* in the first sector [9]

The new vector U^* can be efficiently constructed in *abc* coordinates using straightforward instructions. Table II.1 provides a concise summary of the selection process for its *abc* components, which are determined based on the components of the original vector v^* and the sector number where its tip is located [9].

Table II.1. Selection of the new vector components in *abc* coordinates system [9]

	S_1	S_2	S_3	S_4	S_5	S_6
U_a^*	v_a^*	v_b^*	v_b^*	v_c^*	v_c^*	v_a^*
U_b^*	v_b^*	v_a^*	v_c^*	v_b^*	v_a^*	v_c^*
U_c^*	v_c^*	v_c^*	v_a^*	v_a^*	v_b^*	v_b^*

Fig II.7 illustrates the components of both the original reference vector and new vector within the abc coordinate system.

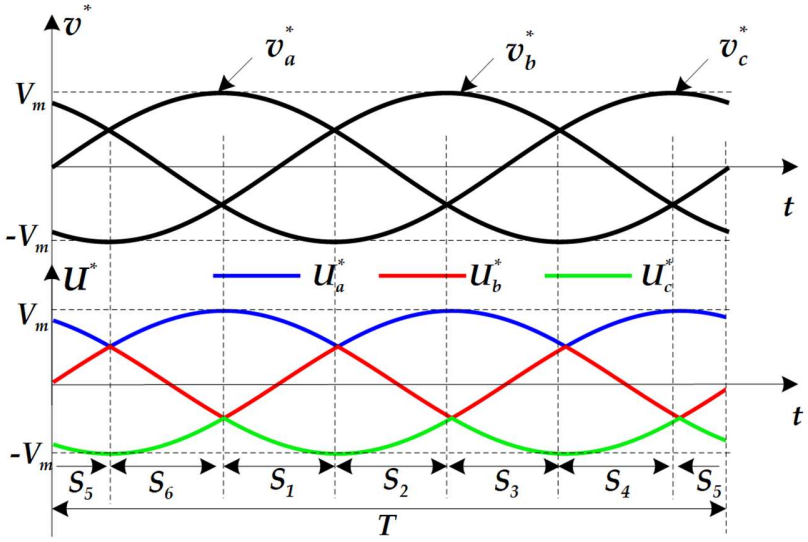


Fig II.7: Components of original reference vector and new one in *abc* coordinates [9]

II.3.2.2. *Sector Number Identification*

To construct the new reference voltage vector, it is essential to determine the sector number containing v^*. The sector number is determined through a straightforward comparison of the components of the reference voltage vector v^* in abc coordinates, as illustrated in Table II.2.

Table II.2: Sector identification [9]

Condition	***Sector number*** (S_i)
$v_a^* \geq v_b^* \geq v_c^*$	1
$v_a^* \geq v_c^* \geq v_b^*$	6
$v_b^* \geq v_a^* \geq v_c^*$	2
$v_b^* \geq v_c^* \geq v_a^*$	3
$v_c^* \geq v_b^* \geq v_a^*$	4
$v_c^* \geq v_a^* \geq v_b^*$	5

II.3.2.3. *Triangle Number Identification*

In the first sector, the new reference voltage vector (U^*) can reside within one of $(n-1)^2$ possible triangles. These triangles are classified into two distinct types: upper triangles (Δ_1) characterized by a vertex pointing upward, and lower triangles (Δ_2), with a vertex

directed downward, as illustrated in Fig II.8. This structured categorization facilitates efficient identification and computation within the sector [9].

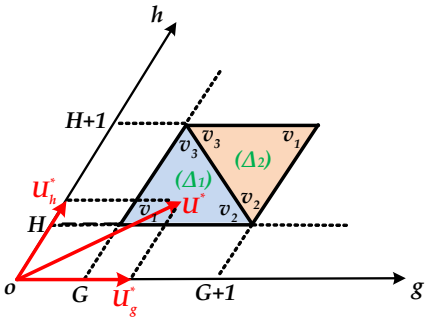


Fig II.8: Upper and lower triangles Δ_1 and Δ_2 [9]

The simplified algorithm requires only the identification of whether U^* is within an upper or lower triangle, making it unnecessary to determine all possible triangles where U^* may be located. This simplification enhances computational efficiency without compromising accuracy.

To simplify the calculation of the new reference vector's location and duration, the g–h coordinate system is employed [9].

The components of the new reference vector U^* in g-h coordinate system are given as:

$$\begin{aligned} U_g^* &= (n-1)\left(U_a^* - U_b^*\right)/v_{dc} \\ U_h^* &= (n-1)\left(U_b^* - U_c^*\right)/v_{dc} \end{aligned} \tag{II.4}$$

Where: n is the number of the DCC level.

Based on the new components of the reference voltage vector $\left(U_g^*, U_h^*\right)$, the triangle type can be easily identified as follows:

$$\begin{aligned} &\text{if } U_g^* + U_h^* < G + H + 1 \quad \rightarrow U^* \text{ is in } (\Delta_1) \\ &\text{if } U_g^* + U_h^* \geq G + H + 1 \quad \rightarrow U^* \text{ is in } (\Delta_2) \end{aligned} \tag{II.5}$$

Where: G and H correspond to the integer parts of U_g^* and U_h^*, respectively, and are defined as follows:

$$\begin{aligned} G &= \text{integer}(U_g^*) \\ H &= \text{integer}(U_h^*) \end{aligned} \tag{II.6}$$

II.3.2.4. *Duration Times Calculation*

By substituting the new reference voltage vector in place of the old one in expression (II.3) and decomposing it in the *g-h* coordinate system, equation (II.3) transforms into the following form:

$$\begin{aligned} v_{1g}t_1 + v_{2g}t_2 + v_{3g}t_3 &= T_s U_g^* \\ v_{1h}t_1 + v_{2h}t_2 + v_{3h}t_3 &= T_s U_h^* \\ t_1 + t_2 + t_3 &= T_s \end{aligned} \tag{II.7}$$

The coordinates of the adjacent switching vectors can be derived as follows:

$$\begin{aligned} (v_{2g}, v_{2h}) &= (G+1, H), \quad \text{for } \Delta_1 \text{ or } \Delta_2 \\ (v_{3g}, v_{3h}) &= (G, H+1), \quad \text{for } \Delta_1 \text{ or } \Delta_2 \\ (v_{1g}, v_{1h}) &= (G+1, H+1), \quad \text{for } \Delta_2 \end{aligned} \tag{II.8}$$

Note that v_2 and v_3 in Δ_1 are identical to v_2 and v_3 in Δ_2 (see Fig II.8), sharing the same coordinates.

Using (II.6), (II.7), and (II.8), the time intervals t_1, t_2 and t_3 corresponding to the adjacent switching vectors v_1, v_2 and v_3, respectively can be determined as follows:

For Δ_1 :

$$\begin{aligned} t_2 &= T_s \left(U_g^* - G \right) \\ t_3 &= T_s \left(U_h^* - H \right) \\ t_1 &= T_s - t_2 - t_3 \end{aligned} \tag{II.9}$$

For Δ_2 :

$$\begin{aligned} t_2 &= T_s \left(H + 1 - U_h^* \right) \\ t_3 &= T_s \left(G + 1 - U_g^* \right) \\ t_1 &= T_s - t_2 - t_3 \end{aligned} \tag{II.10}$$

It is evident that the proposed expressions for the time intervals are straightforward and can be generalized to apply to any triangle within the first sector and any DCC level.

II.3.2.5. Switching Sequences Definition

Since capacitor voltage balancing is not incorporated into the SSVM algorithm in this work, only the first redundant state of each switching vector is utilized. The switching states are organized to generate symmetrical PWM signals, following a specific sequence (illustrated using Triangle 8 from Fig. II.6 as an example for the 5-level DCC):

- Ascending orders for the first-half period: 431→432→442.
- Descending orders for the second-half period: 442→432→431

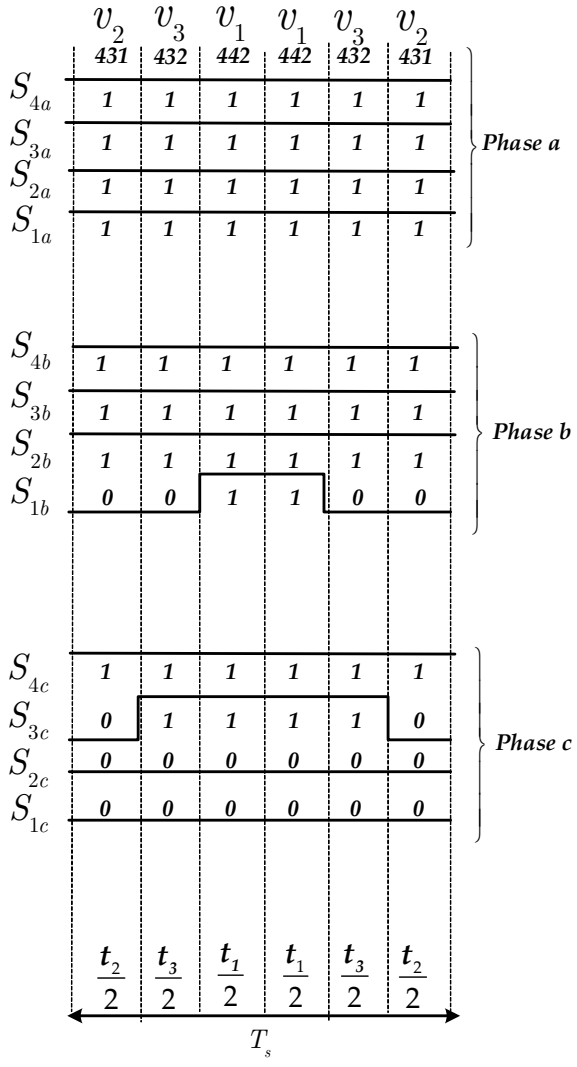


Fig II.9: Symmetrical PWM gate signals in Triangle 8 for 5-level DCC

When the PWM pulses of phases a, b and c are generated in the first sector, the adopted interchanging between phases proposed in [9] is adopted to generate the PWM pulses in other sectors (2, 3, 4, 5 and 6) as clarified in Table II.3.

Table II.3: Switching states interchanging in all sectors [9]

S_1	S_2	S_3	S_4	S_5	S_6
a	$a \rightarrow b$	$a \rightarrow b$	$a \rightarrow c$	$a \rightarrow c$	$a \rightarrow a$
b	$b \rightarrow a$	$b \rightarrow c$	$b \rightarrow b$	$b \rightarrow a$	$b \rightarrow c$
c	$c \rightarrow c$	$c \rightarrow a$	$c \rightarrow a$	$c \rightarrow b$	$c \rightarrow b$

Finaly, the detailed flowchart of the presented SSVM algorithm is illustrated in Fig II.10, where the steps of the SSVM are: i) Sector identification, ii) new reference voltages, iii) Triangle type identification, iv) Duration time calculation and v) Pulses generation.

Fig II.10: Block diagram of the SSVM for n-level inverter

II.4. Simulation results

This section presents detailed simulation results for the proposed carrier-based PWM techniques, as well as the simplified space vector modulation (SSVM) method, applied to 3-level, 5-level, 7-level, and 9-level DCC. The simulations were conducted using MATLAB/Simulink to evaluate and compare the performance of the modulation strategies under consistent operating conditions. In all cases, the carrier signals were configured with a switching frequency of 5 kHz, while the DC-link voltage was maintained at 200 V. The amplitude of the reference signal was set to 80 V to ensure appropriate modulation index levels for accurate waveform generation and system analysis.

Figs II.11, II.13, II.15, and II.17 present the output voltages of the 3-level, 5-level, 7-level, and 9-level DCC, respectively, using PD, POD, and APOD modulation techniques. Correspondingly, Figs II.12, II.14, II.16, and II.18 display the harmonic spectra of these output voltages.

The output voltage waveforms in Figs II.11, II.13, II.15, and II.17 clearly demonstrate how increasing the number of levels in the DCC leads to a more refined staircase approximation of the sinusoidal reference. This improved waveform quality directly

contributes to reduced total harmonic distortion (THD), as evidenced in the harmonic spectra shown in Figures II.12, II.14, II.16, and II.18.

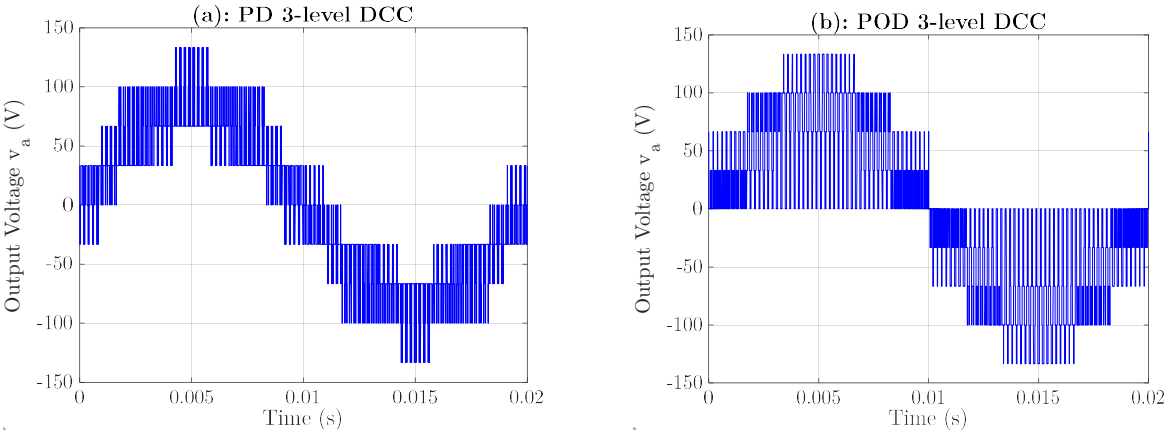


Fig II.11: Output voltage for 3-level DCC, (a): using PD modulation technique, (b): using POD modulation technique

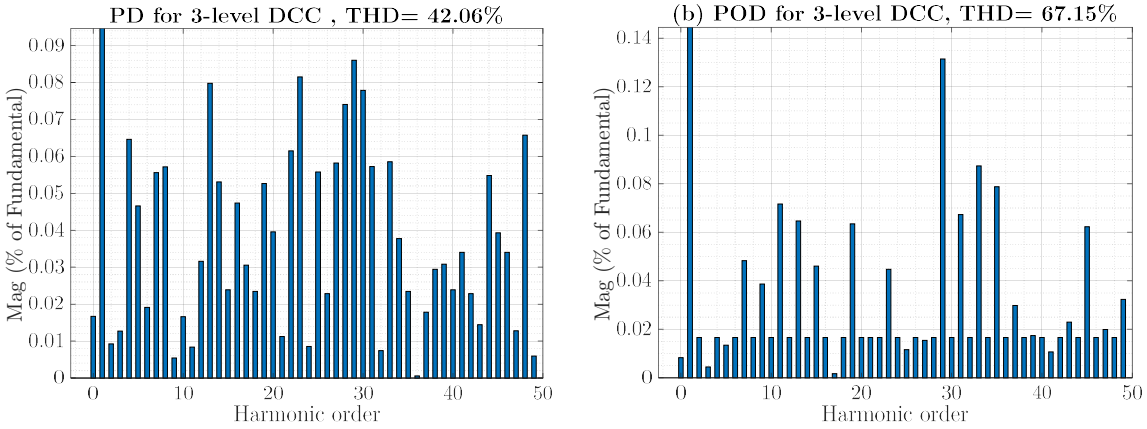
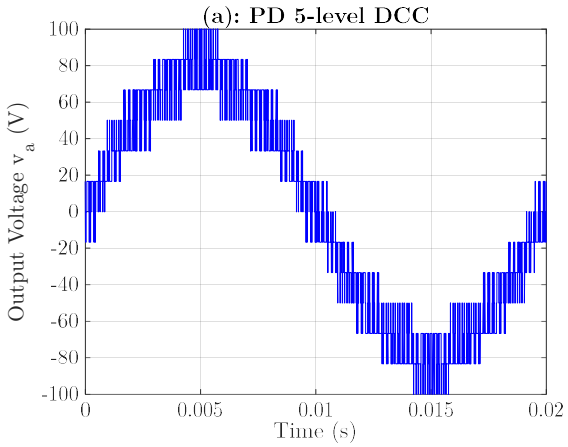


Fig II.12: Harmonic spectrum of the output voltage for 3-level DCC, (a): using PD modulation technique, (b): using POD modulation technique



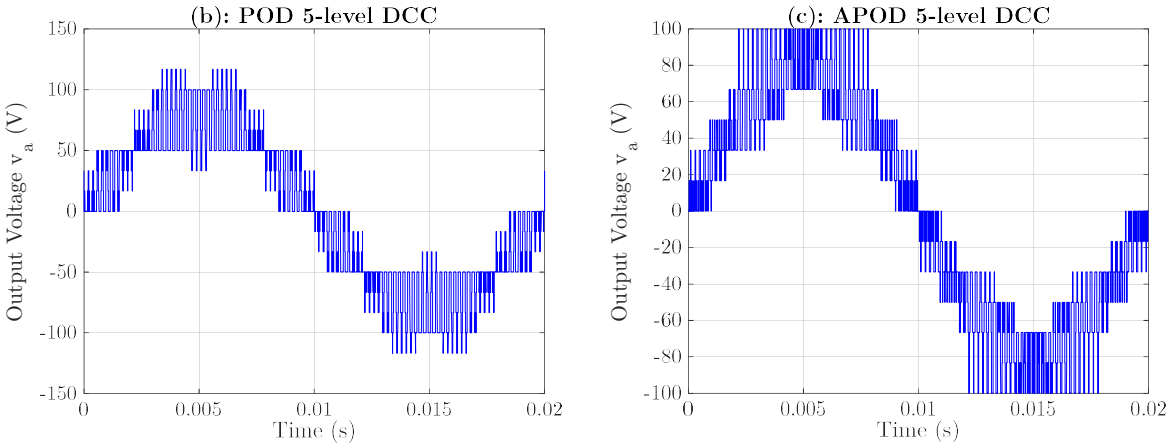


Fig II.13: Output voltage for 5-level DCC, (a): using PD modulation technique, (b): using POD modulation technique, (c): using APOD modulation technique

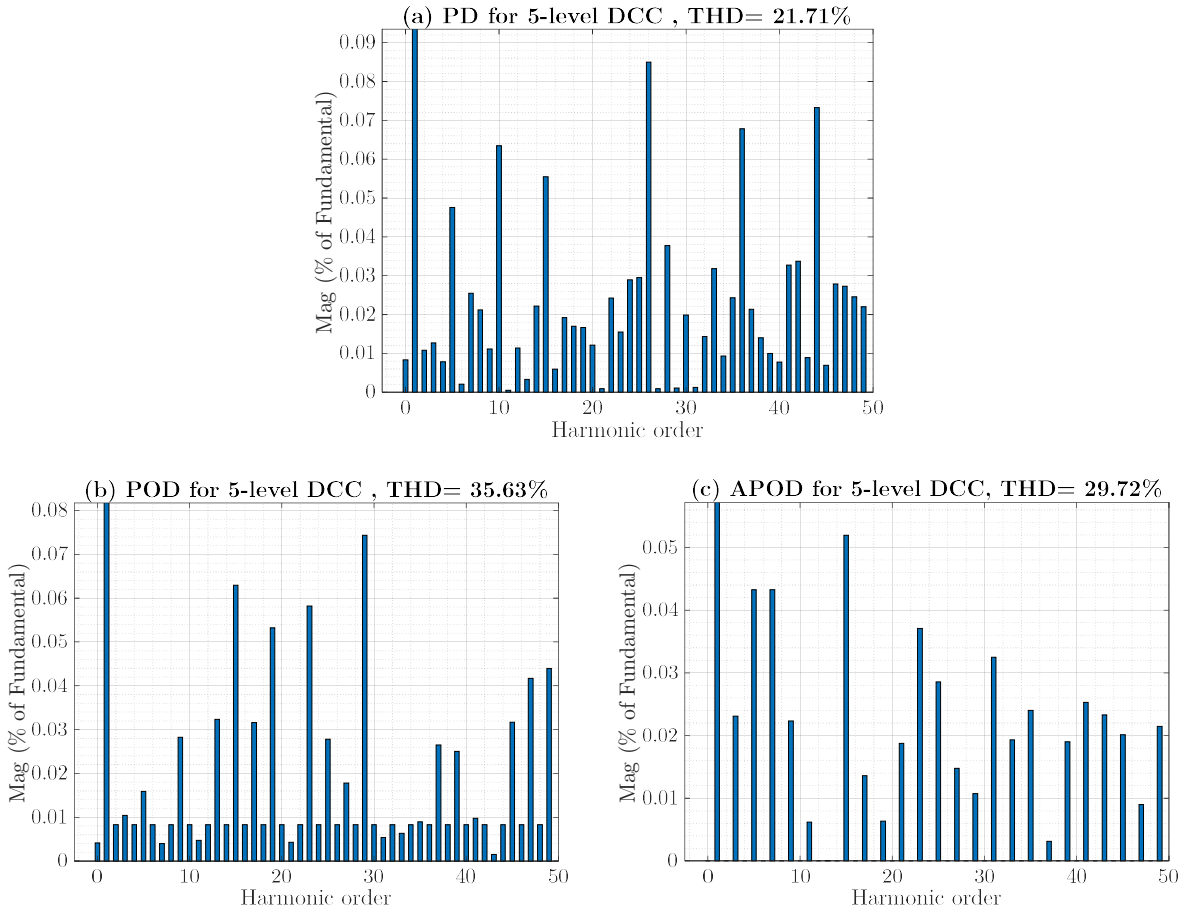


Fig II.14: Harmonic spectrum of the output voltage for 5-level DCC, (a): using PD modulation technique, (b): using POD modulation technique, (c): using APOD modulation technique

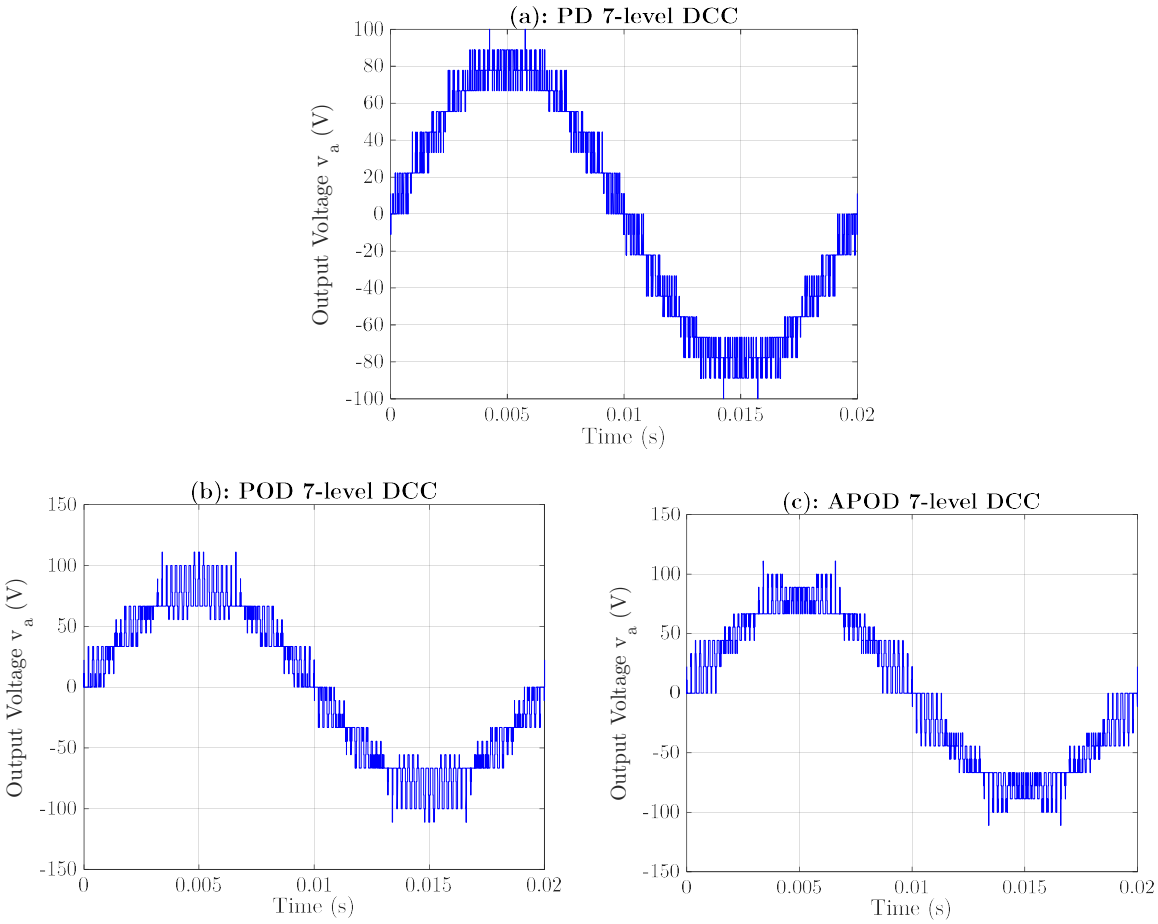
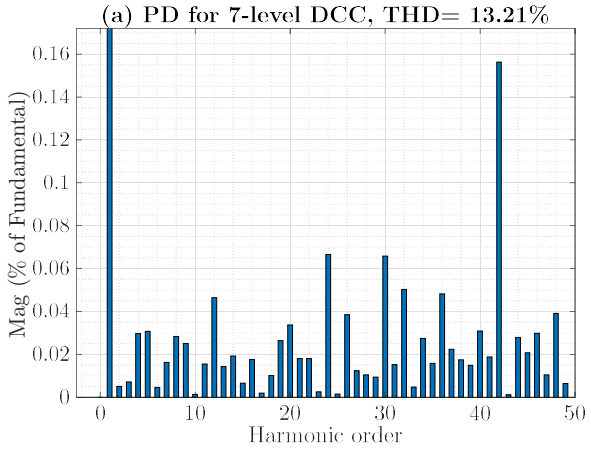


Fig II.15: Output voltage for 7-level DCC, (a): using PD modulation technique, (b): using POD modulation technique, (c): using APOD modulation technique



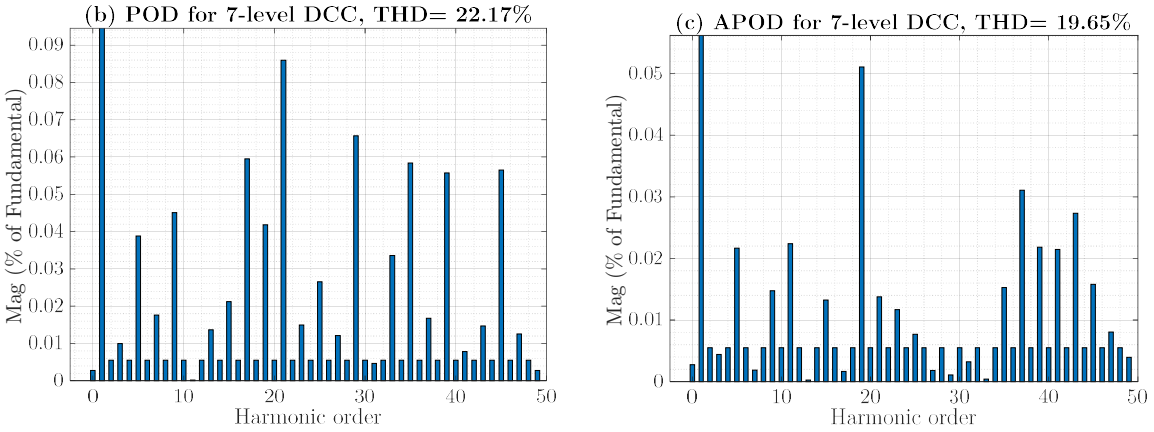
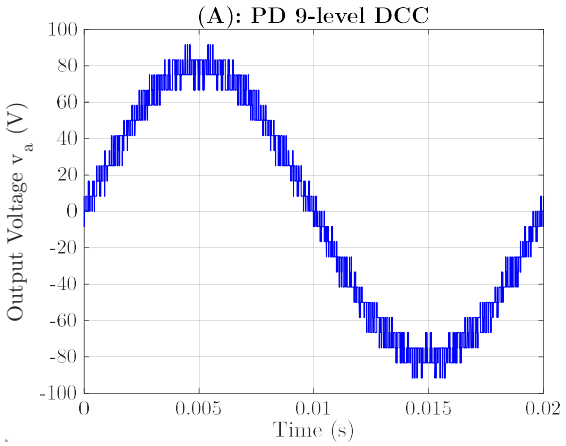


Fig II.16: Harmonic spectrum of the output voltage for 7-level DCC, (a): using PD modulation technique, (b): using POD modulation technique, (c): using APOD modulation technique



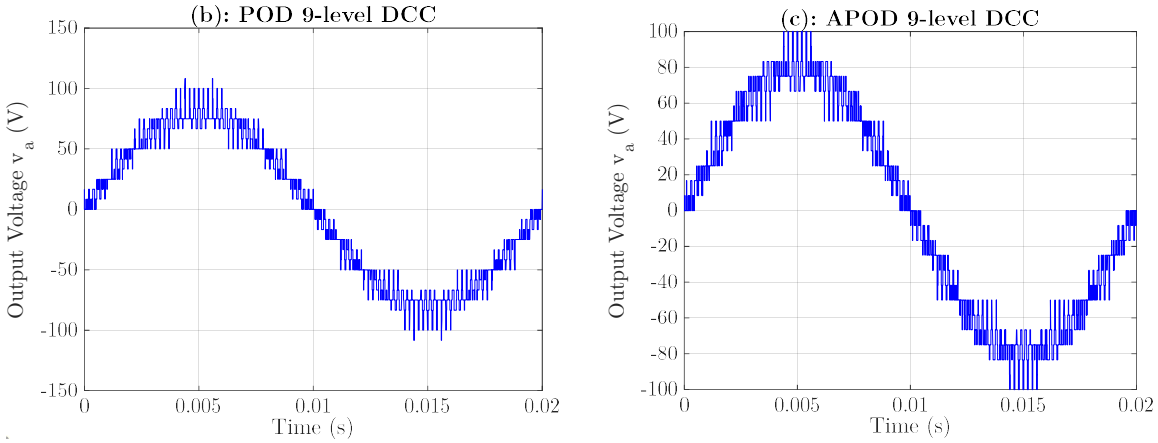


Fig II.17: Output voltage for 9-level DCC, (a): using PD modulation technique, (b): using POD modulation technique, (c): using APOD modulation technique

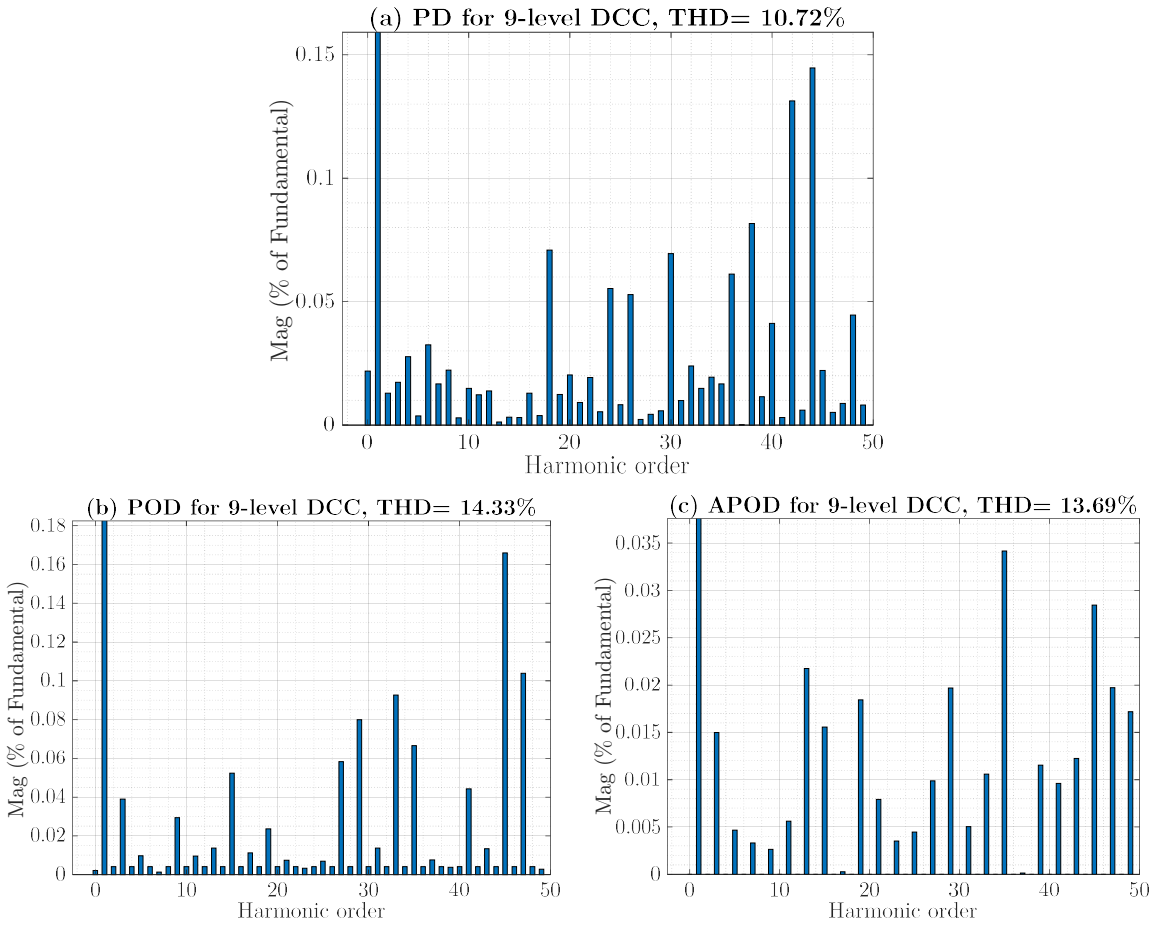


Fig II.18: Harmonic spectrum of the output voltage for 9-level DCC, (a): using PD modulation technique, (b): using POD modulation technique, (c): using APOD modulation technique

Fig II.19 presents the output phase voltage waveforms of the 3-level, 5-level, 7-level, and 9-level DCC using the SSVM technique, with a switching frequency of 5 kHz and a DC-link voltage of 200 V. Correspondingly, Fig II.20 illustrates the harmonic spectra of these output voltages.

Fig II.19 clearly demonstrates the impact of increasing the number of voltage levels on the output waveform quality when using the SSVM technique. For the 3-level converter, the waveform exhibits distinct steps and moderate deviation from the ideal sinusoidal reference. As the number of levels increases to 5, 7, and 9, the output waveforms become progressively smoother and more sinusoidal. This enhanced waveform fidelity is a key benefit of multilevel converters, as it reduces harmonic distortion and improves power quality.

Fig II.20 supports this observation by showing the corresponding harmonic spectra. The spectral analysis reveals a significant reduction in lower-order harmonics as the number of levels increases.

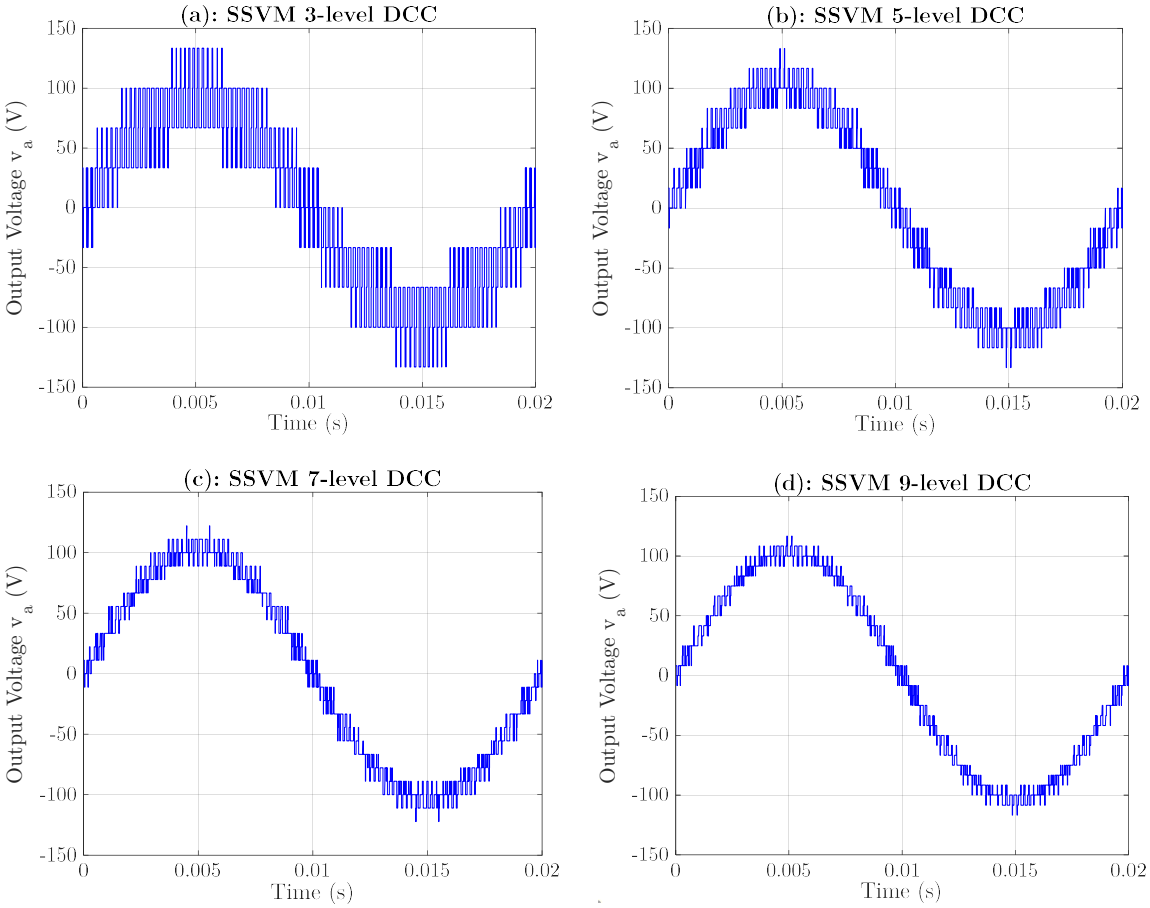
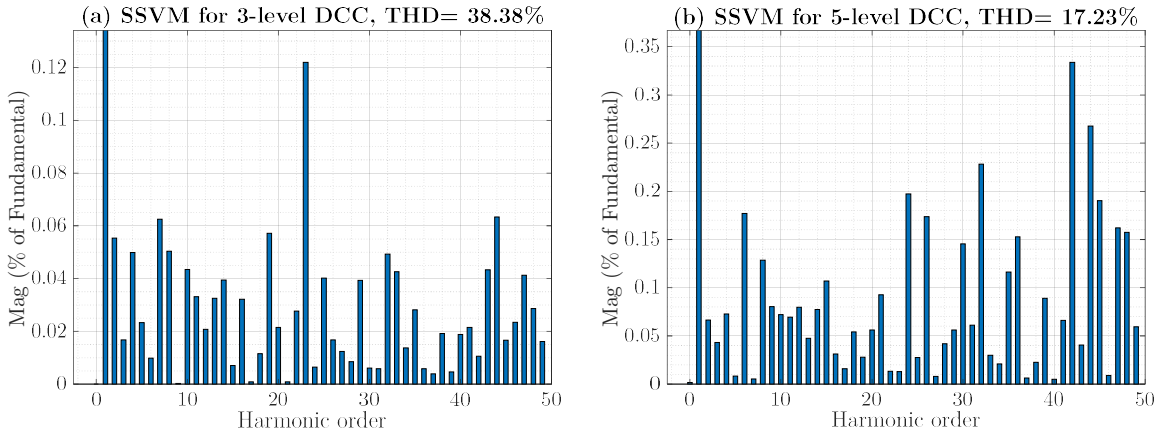


Fig II.19: Output voltage using the SSVM technique for, (a): 3-level DCC, (b): 5-level DCC, (c): 7-level DCC, and (d): 9-level DCC



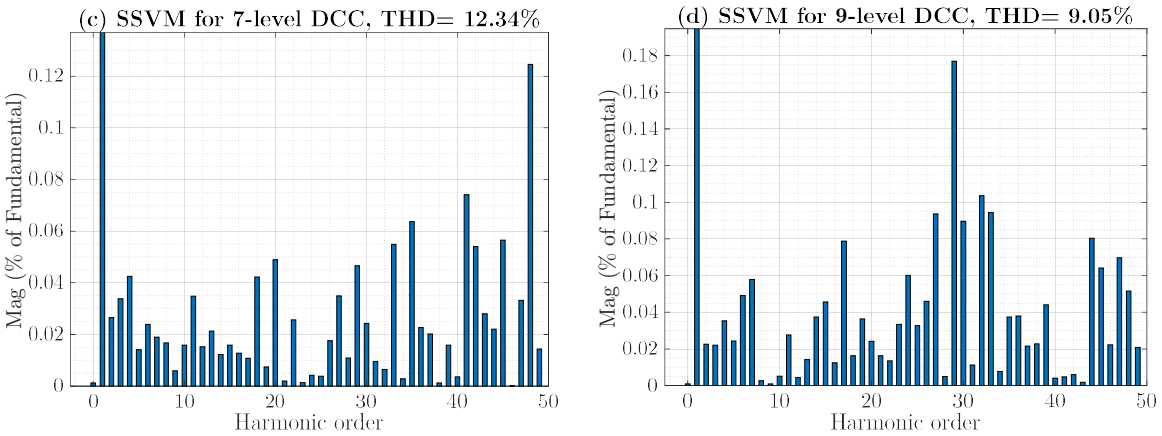


Fig II.20: Harmonic spectrum of the output voltage using SSVM technique for, (a): 3-level DCC, (b): 5-level DCC, (c): 7-level DCC, and (d): 9-level DCC

II. 5. Comparative study

This section presents a comparative analysis based on the THD of the output voltage. Initially, all the proposed PWM techniques are compared in terms of THD performance for 5-level, 7-level, and 9-level diode-clamped converters, using a fixed modulation index of 0.9 (Fig II.21). Subsequently, the variation of THD is examined as a function of both the modulation index and the switching frequency (Fig II.22).

Fig II.21 illustrates the THD of the output voltage for 5-level, 7-level, and 9-level DCC under different modulation techniques PD, POD, APOD, and SSVM. The analysis is conducted at a modulation index of 0.9. The results demonstrate that the THD decreases as the number of converter levels increases, with the 9-level DCC exhibiting the lowest THD across all modulation techniques. Among the modulation strategies, SSVM consistently achieves the lowest THD for all converter levels, highlighting its superior harmonic performance. In contrast, APOD tends to produce higher THD values compared to PD and POD, particularly for lower-level converters.

Fig II.22.a presents the THD of the output voltage for a 9-level DCC controlled using PD, POD, APOD, SSVM. The analysis is conducted over a modulation index range of 0.3 to 0.9, with a fixed switching frequency of 5 kHz. The results reveal that THD decreases significantly as the modulation index increases, with SSVM consistently achieving the lowest THD across the entire range. In contrast, APOD exhibits the highest THD, particularly at lower modulation indices. This highlights the superior harmonic

performance of SSVM in reducing output voltage distortion, especially at higher modulation indices.

Fig II.22.b examines the output voltage THD for the same 9-level DCC under PD, POD, APOD, and SSVM control, with a fixed modulation index of 0.9 and a switching frequency varying from 1 to 10 kHz. The results demonstrate that THD decreases as the switching frequency increases, with SVM again outperforming the other modulation techniques. Notably, APOD shows the highest sensitivity to switching frequency variations, with a more pronounced reduction in THD as the frequency increases. This suggests that SVM is not only effective in minimizing THD but also more robust across a wide range of operating conditions.

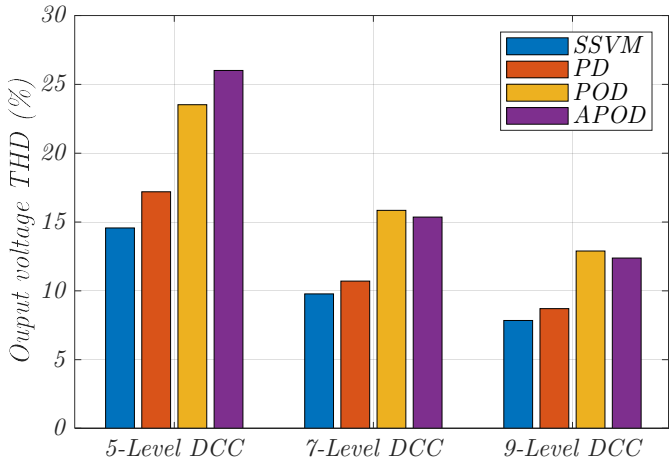
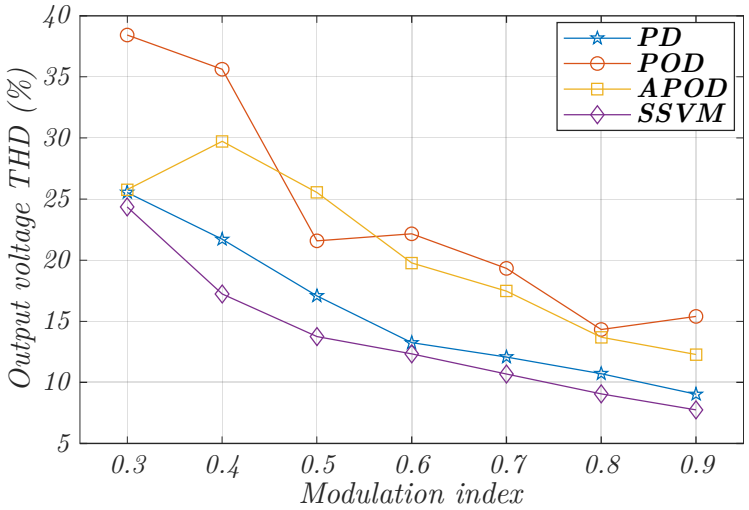


Fig II. 21: Comparative study between the PWM techniques for 5-level, 7-level and 9-level DCC based on output voltage THD

(a)



(b)

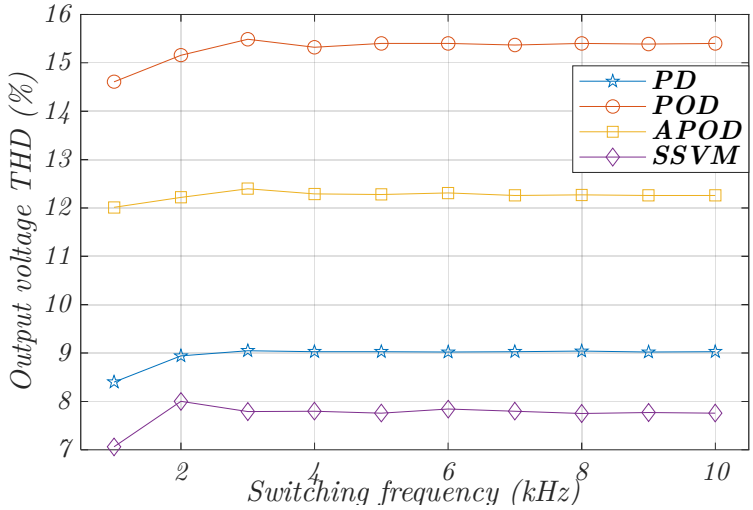


Fig II.22: Comparative study between the PWM techniques for 9-level DCC, (a): THD versus modulation index, (b): THD versus switching frequency

II.6. Conclusion

This chapter has presented a comprehensive study on the simulation and performance evaluation of various modulation techniques for diode-clamped converters (DCCs), including PD, POD, APOD, and the simplified space vector modulation (SSVM) technique. The SSVM approach was examined for its potential to reduce computational complexity while maintaining high output voltage quality.

The presented techniques were simulated and analyzed for 3-level, 5-level, 7-level, and 9-level DCCs, with performance evaluated primarily based on the total harmonic distortion (THD) of the output voltage. The simulation results demonstrated that the SSVM technique consistently outperformed the carrier-based methods (PD, POD, and APOD) in terms of harmonic performance, especially for converters with a higher number of levels. Furthermore, the comparative analysis confirmed the robustness of SSVM under variations in both modulation index and switching frequency, highlighting its effectiveness for applications requiring high-quality voltage output and reliable control performance.

The next chapter will focus on the FPGA-based implementation of the presented PWM techniques for the multilevel DCC.

Chapter III:

FPGA-Based implementation of the PWM techniques

III.1. Introduction

A Field-Programmable Gate Array (FPGA) is a type of integrated circuit that can be configured by the user after manufacturing, allowing for the creation of custom hardware logic. Unlike traditional processors that follow a fixed architecture and execute software instructions, FPGAs are made up of reprogrammable logic blocks, interconnects, and input/output components [10]. These building blocks can be configured to perform specific tasks in parallel, making FPGAs ideal for applications that demand high-speed, low-latency processing such as digital signal processing, real-time image and video processing,

artificial intelligence, telecommunications, and even aerospace systems [11]. The programming of an FPGA is typically done using hardware description languages (HDLs) like VHDL or Verilog, which describe the behavior of electronic circuits. One of the biggest advantages of FPGAs is their flexibility—they can be reprogrammed to adapt to changing requirements or standards. Compared to CPUs, FPGAs offer better performance for specific tasks due to their parallelism, though they may require more development time and specialized knowledge. They also differ from ASICs (Application-Specific Integrated Circuits), which are custom-built for one function and cannot be altered after fabrication. Overall, FPGAs strike a balance between flexibility and performance, making them valuable in both prototyping and final product development across many industries [12].

In this chapter, the PWM techniques for the multilevel DCC converter, previously discussed in Chapter II, are implemented on an FPGA. The implementation utilizes the Xilinx Zedboard Zynq-7000 platform, and the development is carried out using the Vivado Design Suite 2019.3.

III.1.1. Zedboard Zynq-7000 Overview

The Zynq FPGA ZedBoard shown in Fig III.1, is a powerful development platform that combines a dual-core ARM Cortex-A9 Processing System (PS) with Xilinx 7-series Programmable Logic (PL), enabling seamless integration of software and hardware design. The PS handles general-purpose computing tasks, running operating systems like Linux or bare-metal applications, while the PL provides customizable FPGA fabric for hardware acceleration, real-time processing, and peripheral interfacing [12].

Fig III.2 shows basic architecture of the Zynq-7000 showing PL and PS parts. The PS includes essential components like the ARM CPUs, memory controllers (DDR3, Flash), and standard peripherals (USB, UART, Ethernet). It operates independently but can offload computationally intensive tasks to the PL, which consists of configurable logic blocks, DSP slices, and block RAM [12].

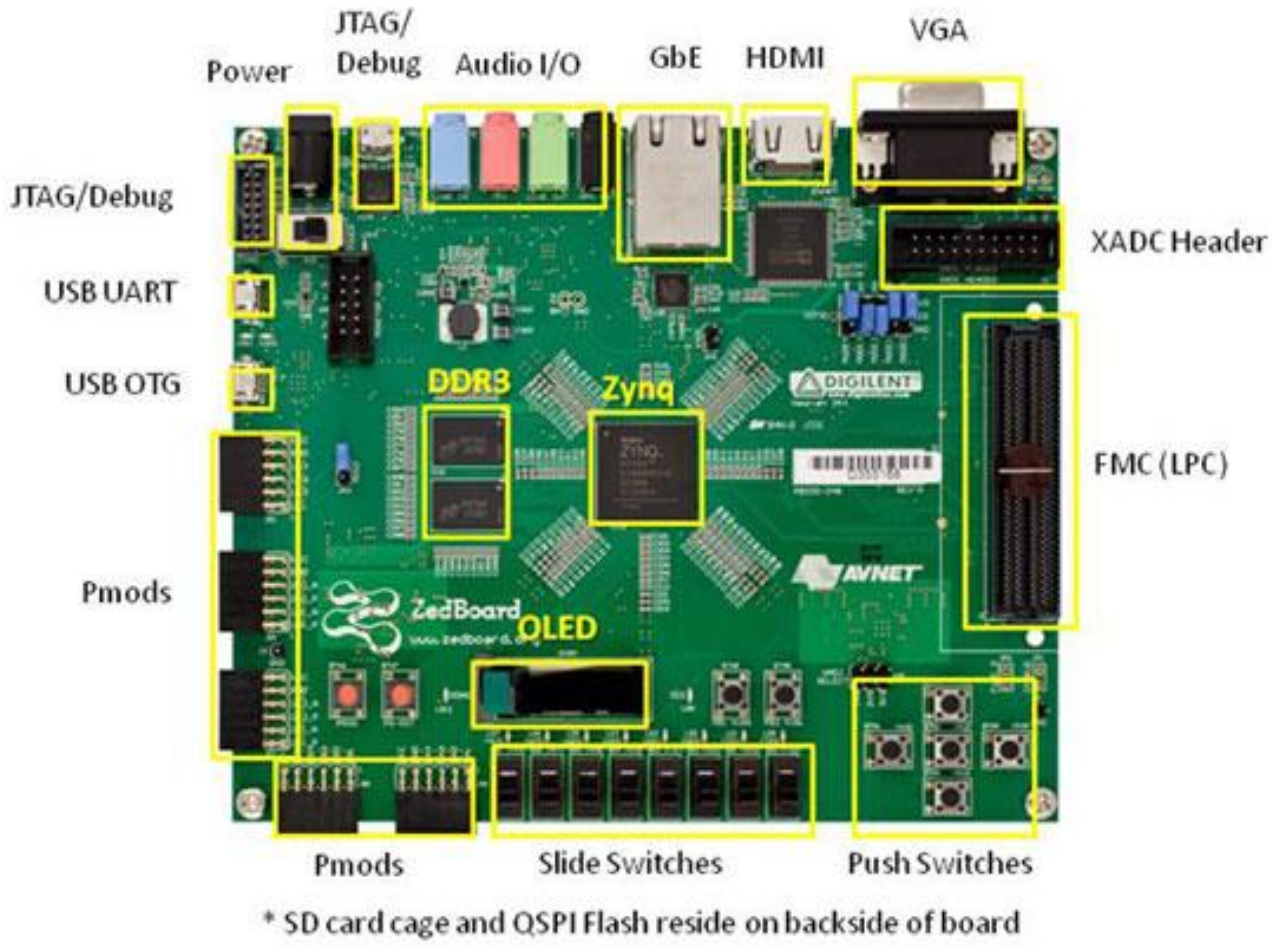


Fig III.1: ZedBoard Zynq-7000 Development Board

III.1.2. Communication between PL and PS

As shown in Fig III.2, the interaction between the PS and PL is facilitated through AXI (Advanced eXtensible Interface) protocols, which include AXI4-Lite for low-bandwidth control, AXI4-Full for high-throughput memory-mapped transfers, and AXI4-Stream for continuous data streaming. Additionally, GPIOs provide simple signal control, while interrupts allow event-driven communication, and DMA enables efficient bulk data transfers without CPU intervention [12].

To set up PS-PL collaboration, designers typically use Xilinx Vivado for hardware configuration and Vitis for software development. In Vivado, the Zynq PS is configured with necessary peripherals and AXI interfaces, while custom IP cores or FPGA logic are integrated into the PL. The AXI interconnects bridge the PS and PL, allowing the ARM cores to access FPGA registers or stream data efficiently [12]. After generating the

bitstream, the hardware design is exported to Vitis, where developers write C/C++ applications to control the PL, utilizing Xilinx drivers for GPIO, DMA, and interrupt handling [12].

By leveraging the ZedBoard's hybrid architecture, developers can optimize systems where the PS manages high-level tasks and the PL handles real-time processing, creating efficient, high-performance embedded solutions. The key lies in properly configuring the AXI infrastructure, ensuring low-latency communication, and balancing workloads between software and hardware for optimal performance [12].

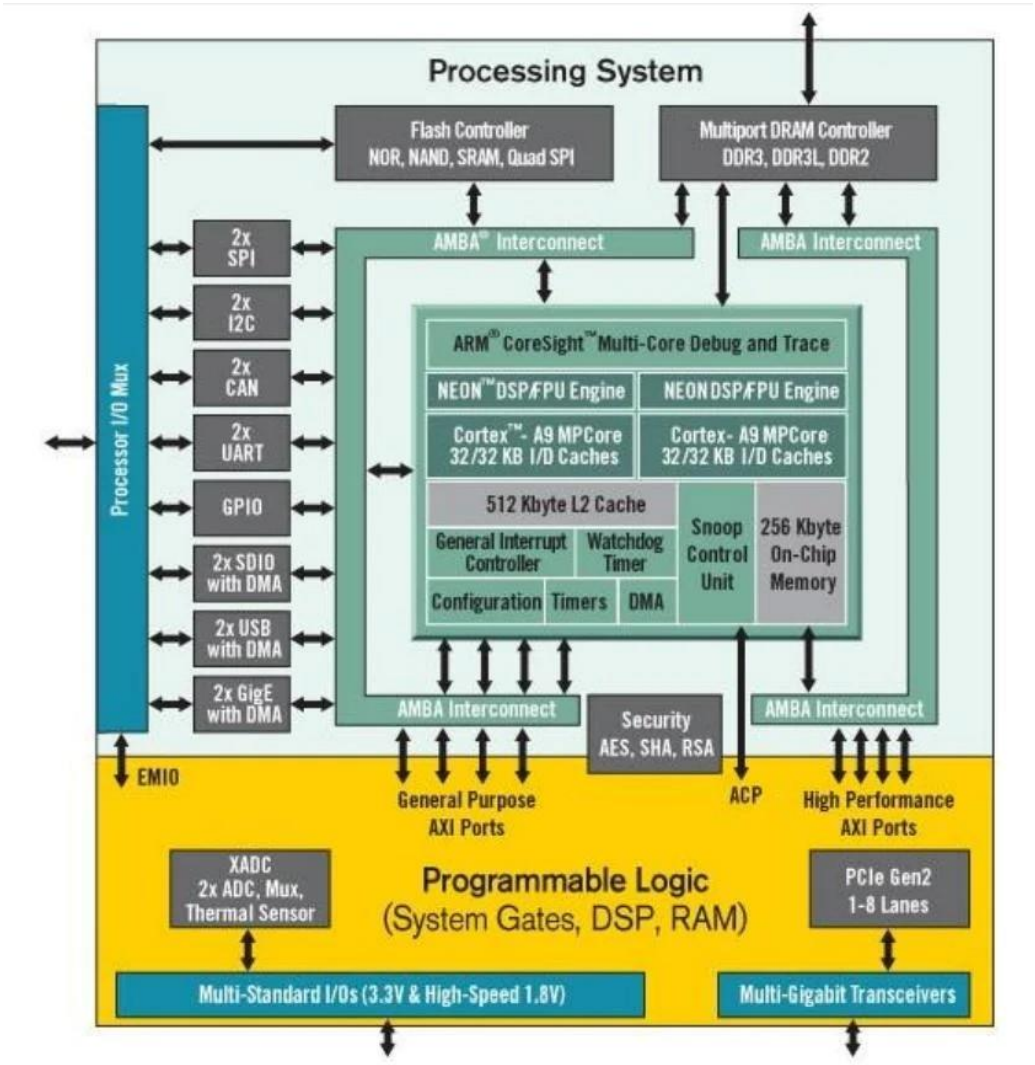


Fig. III.2: Basic Architecture of the Zynq-7000 showing PL and PS components

III.2. FPGA-Based implementation for PWM Techniques

Two different FPGA-based implementation strategies are adopted for the PWM techniques discussed in Chapter II. For the carrier-based PWM methods—PD, POD, and APOD—only the PL part of the FPGA is utilized. This is because these techniques are primarily based on simple, repetitive comparisons between reference signals and carrier waveforms, which are highly parallelizable and efficiently implemented in hardware using VHDL within the Vivado Design Suite 2019.2 [13-14].

In contrast, the simplified Space Vector Modulation (SSVM) technique involves more complex computations, including coordinate transformations, sector determination, and timing calculations. These operations are better suited to software execution due to their sequential and algorithmic nature. Therefore, a combined PL and PS implementation is employed: the PL handles the generation of reference voltages, while the PS (ARM Cortex-A9 processor) performs the SVM algorithm computations. This hybrid approach leverages the strengths of both PL for fast signal generation and PS for flexible numerical processing [13-14].

III.2.1. Implementation of the carrier based PWM techniques

PWM techniques—Phase Disposition (PD), Phase Opposition Disposition (POD), and Alternative Phase Opposition Disposition (APOD)—are implemented using the Vivado Design Suite 2019.2. The processing is carried out entirely in the Programmable Logic (PL) section of the Zynq ZedBoard FPGA. All functional blocks are described in VHDL and integrated through the block design illustrated in Fig. 5 [13-14].

In this subsection, the implementation of the carrier-based PWM techniques is presented and thoroughly explained. Since all carrier-based PWM methods—such as PD, POD, and APOD—share the same fundamental principle, which involves comparing modulating reference signals with carrier waveforms, we focus our detailed explanation on the PD technique applied to a 5-level DCC. The architecture of the implementation, along with the functional operation of each block within the system, is described in detail to illustrate the complete design flow and logic [13-14].

Fig III.3 presents the VHDL-based block diagram for the PD technique applied to a 5-level DCC. It consists of the following primary modules:

- **Voltage Generation Block**: Responsible for generating three-phase sinusoidal reference voltages (v_a, v_b, and v_c) corresponding to the desired output waveform.

- **Up-Down Counter Block**: Generates a set of triangular carrier signals using up-down counting logic. These carriers are used as modulation signals for level comparison.
- **Comparison Block**: Compares each phase's reference voltage with the corresponding carrier signals to generate gate signals that control the switching states of the converter's power devices.
- **Output Voltage Generation Block (v_abc_out)**: This block synthesizes the output voltages of the multilevel converter based on the gate signals and the supplied DC voltage, following the switching strategy described in Chapter 2.

All blocks in Fig III.3 are implemented in VHDL, with interconnections defined through Vivado's IP Integrator. Detailed descriptions of the internal structure and behavior of each block are provided in the following sub-sections.

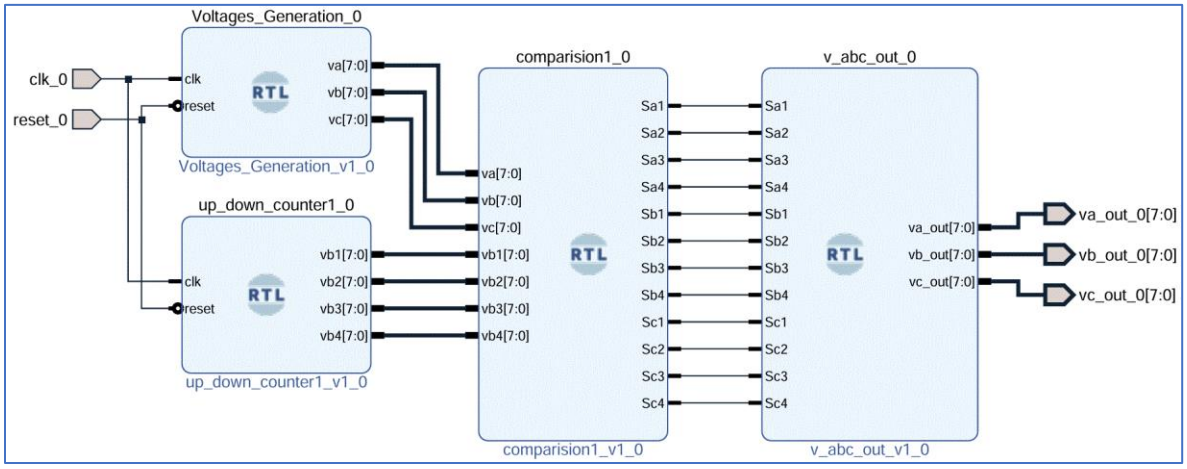


Fig III.3: Block design of the PD Technique for **5-**level inverter [13-14]

III.2.1.1. Voltages generation block

Fig III.4 shows this block separately, this VHDL code in this block implements a three-phase sine wave generator that produces 50 Hz reference voltages in 8-bit resolution, intended for use with PD modulation technique. The generated signals (v_a, v_b, and v_c) are 120° phase-shifted sine waves, which serve as the reference voltages for the PD modulation technique.

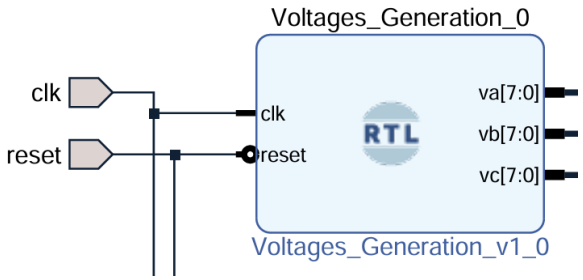


Fig III.4: The voltage generation of the PD technique for 5-level converter

III.2.1.1.1. VHDL code for voltage generation block

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.all;
entity Voltages_Generation is
                port (clk: in std_logic;
                        reset: in std_logic;
                        va: out std_logic_vector(7 downto 0);
                        vb: out std_logic_vector(7 downto 0);
                        vc: out std_logic_vector(7 downto 0));
end Voltages_Generation;
architecture Behavioral of Voltages_Generation is
signal i:integer:= 0;
signal j:integer:= 199;
signal k:integer:= 99;
signal clk_div: std_logic:='0';
signal a : integer:=0;
signal xa: unsigned (7 downto 0):=(others=>'0');--"00000000"
signal xb: unsigned (7 downto 0):=(others=>'0');--"00000000"
signal xc: unsigned (7 downto 0):=(others=>'0');--"00000000"
type memory_type is array (0 to 299) of integer range 0 to 200;
constant sin_a: memory_type :=(80, 82, 83, 85, 87, 88, 90, 92, 93, 95, 97, 98, 100, 102, 103, 105, 106,
108, 109, 111, 113, 114, 116, 117, 119, 120, 121, 123, 124, 126, 127, 128, 130, 131, 132, 134, 135, 136,
137, 138, 139, 141, 142, 143, 144, 145, 146, 147, 148, 148, 149, 150, 151, 152, 152, 153, 154, 154, 155,
156, 156, 157, 157, 157, 158, 158, 159, 159, 159, 159, 160, 160, 160, 160, 160, 160, 160, 160, 160, 160,
160, 159, 159, 159, 159, 158, 158, 157, 157, 157, 156, 156, 155, 154, 154, 153, 152, 152, 151, 150,149,
148, 148, 147, 146, 145, 144, 143, 142, 141, 139, 138, 137, 136, 135, 134, 132, 131, 130, 128, 127, 126,
124, 123, 121, 120, 119, 117, 116, 114, 113, 111, 109, 108, 106, 105, 103, 102, 100, 98, 97, 95, 93, 92,
90, 88, 87, 85, 83, 82, 80, 78, 77, 75, 73, 72, 70, 68, 67, 65, 63, 62, 60, 58, 57, 55, 54, 52, 51, 49, 47, 46,
44, 43, 41, 40, 39, 37, 36, 34, 33, 32, 30, 29, 28, 26, 25, 24, 23, 22, 21, 19, 18, 17, 16, 15, 14, 13, 12, 12,11,
10, 9, 8, 8, 7, 6, 6, 5, 4, 4, 3, 3, 3, 2, 2, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 3, 3, 3, 4, 4, 5,
6, 6, 7, 8, 8, 9, 10, 11, 12, 12, 13, 14, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28, 29, 30, 32, 33, 34, 36,
37, 39, 40, 41, 43, 44, 46, 47, 49, 51, 52, 54, 55, 57, 58, 60, 62, 63, 65, 67, 68, 70, 72, 73, 75, 77, 78);
begin
process (clk)
begin
    if (rising_edge(clk)) then
    if a=3332 then -- a=(Ts/T_clock)/2-1
```

```
            a<=0;
            clk_div<=not clk_div;
          else
            a<=a+1;
        end if;
        end if;
 end process;
 process (clk_div,reset)
begin
      if reset='1' then
xa<=(others=>'0');
xb<=(others=>'0');
xc<=(others=>'0');
i<=0;
j<=199;
k<=99;
  else
     if (rising_edge(clk_div)) then
               xa<=to_unsigned (sin_a(i)+20,8); --xa=to_unsigned(sin_a(i),8)
               xb<=to_unsigned (sin_a(j)+20,8);
               xc<=to_unsigned (sin_a(k)+20,8);
if (i<299) then
    i<=i+1;
               else
                  i<=0;
               end if;
if (j<299) then
 j<=j+1;
               else
                  j<=0;
               end if;
if (k<299) then
 k<=k+1;
               else
                  k<=0;
               end if;
 end if;
end if;
end process;
va<=std_logic_vector(xa);
vb<=std_logic_vector(xb);
vc<=std_logic_vector(xc);
end Behavioral;
```

III.2.1.1.2. Code explication for voltage generation block

1. *Clock and Reset Inputs:*
    - The design uses the FPGA's clock (clk) and a synchronous reset (reset) to control the generation of the sine waves.

2. *Frequency Generation:*
    - The input clock is divided to generate a lower-frequency clock (clk_div) using a counter (a). This division ensures that the sine wave samples are updated at the correct rate to produce a 50 Hz output.
3. *Three-Phase Sine Wave Generation*:
    - Three signals (v_a, v_b, v_c) are generated with 120° phase shifted between them.
    - A precomputed sine wave lookup table (sin_a) stores 300 samples of a single sine wave period (quantized to 8 bits).
    - The three phases are produced by accessing the lookup table at different offsets (i, j, and k), where:
        - $\boldsymbol{i}$ increments sequentially (0 to 299) for phase v_a.
        - $\boldsymbol{j}$ starts at 199 (≈240° offset, simulating -120° shift) for phase v_b.
        - $\boldsymbol{k}$ starts at 99 (≈120° offset) for phase v_c.
4. *Offset Adjustment:*
    - A DC offset of 20 is added to each sine wave sample (to ensure all values are positive and suitable for modulation).
    - The output voltages are 8-bit unsigned vectors, ranging approximately between 20 and 180 (since the sine table has a peak value of 160).

III.2.1.2. Up down counter block

This VHDL code generates four triangular carrier signals (v_{b1} to v_{b4}) in 8-bit resolution, designed specifically for PD technique used in multilevel inverters. The carriers are phase-aligned, uniformly spaced, and synchronized to create properly staggered PWM signals when compared with reference waveforms.

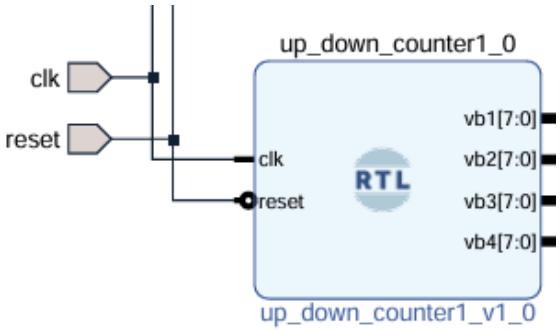


Fig III.5: up down counter block of the PD technique for 5-level converter

III.2.1.2.1. VHDL code for up down counter block

```
library IEEE;use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.ALL;
entity up_down_counter1 is
  Port ( clk : in STD_LOGIC;
         reset : in STD_LOGIC;
         vb1,vb2,vb3,vb4 : out STD_LOGIC_VECTOR (7 downto 0));
end up_down_counter1;
architecture Behavioral of up_down_counter1 is
signal clk_div,dir: std_logic:='0';
signal a,d1,d2,d3,d4,b: integer:=0;
begin
process (clk)begin
      if (rising_edge(clk)) then
            if a=150 then -- a=(Ts/T_clock)/2-1
              a<=0;
              clk_div<=not clk_div;
            else
              a<=a+1;
            end if;
     end if;
 end process;
    process (clk_div,reset) begin
            if reset='1' then
              b<=0;
              d1<=0;
            else
             if rising_edge (clk_div) then

              if   b=49 then
                   dir<=not dir;
                   b<=0;
              else
                   b<=b+1;
              end if;

                if dir='0'  then
                   d1<=d1+1;
                else
```

```
                d1<=d1-1;
              end if;
              end if;
        end if;
  end process;
  d2<=d1+50;
  d3<=d1+100;
  d4<=d1+150;
 vb4<=std_logic_vector(to_unsigned (d1,8));
 vb3<=std_logic_vector(to_unsigned (d2,8));
 vb2<=std_logic_vector(to_unsigned (d3,8));
 vb1<=std_logic_vector(to_unsigned (d4,8));
end Behavioral;
```

III.2.1.2.2. Code explication for up down counter:

1. *Inputs*

- clk: FPGA system clock (high-frequency input)
- reset: synchronous reset (initializes all counters to zero)

2. *Clock Divider for Carrier Frequency*

- The input clock is divided using a counter (a) to generate a slower clock (clk_div).
- **Calculation**:
  - a_max = 150 (division factor)
  - clk_div toggles every 151 clock cycles (a=150 then reset).
  - This controls the switching frequency of the triangular carriers.

3. *Triangular Wave Generation Using an Up-Down Counter*

- A bidirectional counter (d1) generates the base triangular waveform:
  - Up-counting (dir = '0'): d1 <= d1 + 1
  - Down-counting (dir = '1'): d1 <= d1 - 1
- The counter changes direction when b reaches 49 (b=49), ensuring a symmetric triangular wave with a peak value of 49.

4. *Four Phase-Disposed Carriers*

- Base carrier (d1): Triangular wave ranging from 0 to 49.
- Three additional carriers (d2, d3, d4) are generated by adding fixed offsets:

- d2 = d1 + 50
- d3 = d1 + 100
- d4 = d1 + 150

III.2.1.3. The comparison block:

This VHDL code implements the comparator stage of a Phase Disposition (PD) Modulation system for a five-level inverter. It takes three-phase reference voltages (v_a, v_b, v_c) and four triangular carrier signals (v_{b1}-v_{b4}) as inputs, and generates 12 switching signals (S_{a1}-S_{a4}, S_{b1}-S_{b4}, S_{c1}-S_{c4}) that control the inverter's power devices.

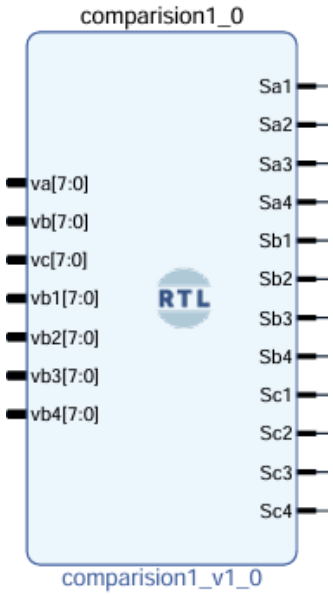


Fig III.6: Comparison block of the PD technique for 5-level converter

III.2.1.3.1. VHDL code of the comparison block:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.all;
entity comparision1 is
    Port ( va,vb,vc : in STD_LOGIC_VECTOR (7 downto 0);
           vb1,vb2,vb3,vb4 : in STD_LOGIC_VECTOR (7 downto 0);
          Sa1,Sa2,Sa3,Sa4,Sb1,Sb2,Sb3,Sb4,Sc1,Sc2,Sc3,Sc4 : out STD_LOGIC);
end comparision1;
architecture Behavioral of comparision1 is
begin
```

```
Sa1<='1'when (unsigned(va)>unsigned(vb1)) else   '0';
Sa2<='1'when (unsigned(va)>unsigned(vb2)) else   '0';
Sa3<='1'when (unsigned(va)>unsigned(vb3)) else   '0';
Sa4<='1'when (unsigned(va)>unsigned(vb4)) else   '0';

Sb1<='1'when (unsigned(vb)>unsigned(vb1)) else   '0';
Sb2<='1'when (unsigned(vb)>unsigned(vb2)) else   '0';
Sb3<='1'when (unsigned(vb)>unsigned(vb3)) else   '0';
Sb4<='1'when (unsigned(vb)>unsigned(vb4)) else   '0';

Sc1<='1'when (unsigned(vc)>unsigned(vb1)) else   '0';
Sc2<='1'when (unsigned(vc)>unsigned(vb2)) else   '0';
Sc3<='1'when (unsigned(vc)>unsigned(vb3)) else   '0';
Sc4<='1'when (unsigned(vc)>unsigned(vb4)) else   '0';

end Behavioral;
```

III.2.1.3.2. code explication for the comparison block

1. *Inputs*

- Three-phase reference voltages ($\boldsymbol{v_a}$, $\boldsymbol{v_b}$, $\boldsymbol{v_c}$):
    - 8-bit sine wave signals (typically from a sine wave generator).
    - Represent the desired output voltage waveforms (120° phase-shifted).
- Four triangular carriers ($\boldsymbol{v_{b1}}$, $\boldsymbol{v_{b2}}$, $\boldsymbol{v_{b3}}$, $\boldsymbol{v_{b4}}$):
    - 8-bit triangular waves (from a carrier generator).
    - Vertically aligned (in-phase) with uniform spacing (PD modulation).

2. *Outputs*

- 12 switching signals (4 per phase):
    - $\boldsymbol{S_{a1}}$-$\boldsymbol{S_{a4}}$ (Phase A switches)
    - $\boldsymbol{S_{b1}}$-$\boldsymbol{S_{b4}}$ (Phase B switches)
    - $\boldsymbol{S_{c1}}$-$\boldsymbol{S_{c4}}$ (Phase C switches)
    - Each signal is '1' (ON) when the reference voltage > carrier voltage, else '0' (OFF).

3. *Operation Principle*

- The code performs real-time comparison between each reference phase and all four carriers:
    - For Phase A:

- S_{a1} = '1' if $v_a > v_{b1}$
- S_{a2} = '1' if $v_a > v_{b2}$
- S_{a3} = '1' if $v_a > v_{b3}$
- S_{a4} = '1' if $v_a > v_{b4}$

- The same logic applies for Phases B (v_b) and C (v_C).

III.2.1.4. The output voltage block:

This VHDL code reconstructs the three-phase output voltages (va_out, vb_out, vc_out) of a four-level inverter using the switching states (S_{a1}-S_{a4}, S_{b1}-S_{b4}, S_{c1}-S_{c4}) generated by a Phase Disposition (PD) PWM modulator. It implements the final stage of a multilevel inverter control system, converting switching signals into analog-equivalent output voltages.

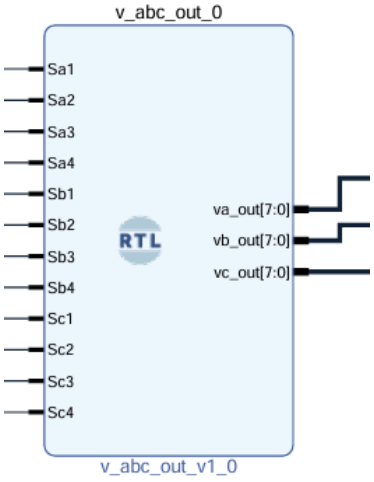


Fig III.7: The output voltage block

III.2.1.4.1. VHDL code for the output voltage block

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.ALL;
entity v_abc_out is
    Port (Sa1,Sa2,Sa3,Sa4,Sb1,Sb2,Sb3,Sb4,Sc1,Sc2,Sc3,Sc4 : in STD_LOGIC;
          va_out,vb_out,vc_out   : out STD_LOGIC_VECTOR (7 downto 0));
end v_abc_out;
architecture Behavioral of v_abc_out is
```

```
constant Vdc : integer:=96;--Vdc
signal  Saa1,Saa2,Saa3,Saa4,Sbb1,Sbb2,Sbb3,Sbb4,Scc1,Scc2,Scc3,Scc4 : integer:=0;
signal Fa1,Fa2,Fa3,Fa4,Fb1,Fb2,Fb3,Fb4,Fc1,Fc2,Fc3,Fc4 : integer:=0;
signal va_outt_0,vb_outt_0,vc_outt_0 : integer:=0;
signal va_outt,vb_outt,vc_outt : integer:=0;
begin
Saa1<=1 when (Sa1='1') else 0;
Saa2<=1 when (Sa2='1') else 0;
Saa3<=1 when (Sa3='1') else 0;
Saa4<=1 when (Sa4='1') else 0;

Sbb1<=1 when (Sb1='1') else 0;
Sbb2<=1 when (Sb2='1') else 0;
Sbb3<=1 when (Sb3='1') else 0;
Sbb4<=1 when (Sb4='1') else 0;

Scc1<=1 when (Sc1='1') else 0;
Scc2<=1 when (Sc2='1') else 0;
Scc3<=1 when (Sc3='1') else 0;
Scc4<=1 when (Sc4='1') else 0;

   Fa4<=Saa1*Saa2*Saa3*Saa4;
   Fb4<=Sbb1*Sbb2*Sbb3*Sbb4;
   Fc4<=Scc1*Scc2*Scc3*Scc4;
   Fa3<=(1-Saa1)*Saa2*Saa3*Saa4;
   Fb3<=(1-Sbb1)*Sbb2*Sbb3*Sbb4;
   Fc3<=(1-Scc1)*Scc2*Scc3*Scc4;

   Fa2<=(1-Saa1)*(1-Saa2)*Saa3*Saa4;
   Fb2<=(1-Sbb1)*(1-Sbb2)*Sbb3*Sbb4;
   Fc2<=(1-Scc1)*(1-Scc2)*Scc3*Scc4;
   Fa1<=(1-Saa1)*(1-Saa2)*(1-Saa3)*Saa4;
   Fb1<=(1-Sbb1)*(1-Sbb2)*(1-Sbb3)*Sbb4;
   Fc1<=(1-Scc1)*(1-Scc2)*(1-Scc3)*Scc4;
va_outt_0<= Fa4*Vdc + Fa3*(3*vdc/4) + Fa2*(vdc/2) + Fa1*(vdc/4);
vb_outt_0<= Fb4*Vdc + Fb3*(3*vdc/4) + Fb2*(vdc/2) + Fb1*(vdc/4);
vc_outt_0<= Fc4*Vdc + Fc3*(3*vdc/4) + Fc2*(vdc/2) + Fc1*(vdc/4);
va_outt<= (2*va_outt_0 - vb_outt_0 - vc_outt_0)/3 ;
vb_outt<= (-va_outt_0 +2*vb_outt_0 - vc_outt_0)/3 ;
vc_outt<= (-va_outt_0 - vb_outt_0 + 2*vc_outt_0)/3;
va_out<=STD_LOGIC_VECTOR(to_signed(va_outt,8));
vb_out<=STD_LOGIC_VECTOR(to_signed(vb_outt,8));
vc_out<=STD_LOGIC_VECTOR(to_signed(vc_outt,8));
end Behavioral;
```

III.2.1.4.2. Code explication of the output voltage block

1. *Inputs*

- 12 switching signals (4 per phase):
  - S_{a1}-S_{a4} (Phase A switching states)
  - S_{b1}-S_{b4} (Phase B switching states)

- S_{c1}-S_{c4} (Phase C switching states)
- Each signal is '1' (ON) or '0' (OFF), generated by comparing reference sine waves with triangular carriers.

2. *Outputs*

- Three-phase output voltages (va_out, vb_out, vc_out):
  - 8-bit signed vectors representing the instantaneous voltage levels.
  - Scaled to the DC bus voltage ($v_{dc} = 96$ in this case).

3. *Core Operation*

Step 1: Switching State to Voltage Level Conversion

- Each phase's switching combination determines its output voltage level:
  - Phase A:
    - $F_{a4} = 1$ if all switches (S_{a1}-S_{a4}) are ON $\rightarrow v_{dc}$
    - $F_{a3} = 1$ if S_{a2}-S_{a4} are ON, S_{a1} OFF $\rightarrow 3\ v_{dc}/4$
    - $F_{a2} = 1$ if S_{a3}-S_{a4} are ON, S_{a1}-S_{a2} OFF $\rightarrow v_{dc}/2$
    - $F_{a1} = 1$ if only S_{a4} is ON $\rightarrow v_{dc}/4$
  - The same logic applies to Phases B and C.

Step 2: Plot voltages

- The raw phase voltages (va_outt_0, vb_outt_0, vc_outt_0) are calculated using:

$$va\_outt\_0 = Fa4 \cdot Vdc + Fa3 \cdot \frac{3Vdc}{4} + Fa2 \cdot \frac{Vdc}{2} + Fa1 \cdot \frac{Vdc}{4}$$

- (Similarly for vb_outt_0 and vc_outt_0.)

Step 3: Output voltages

- To ensure balanced output, the final voltages are computed using Clark transformation-like averaging:

$$va\_outt = \frac{2 \cdot va\_outt\_0 - vb\_outt\_0 - vc\_outt\_0}{3}$$

- (Similarly for vb_outt and vc_outt.)

III.2.1.5. Resources utilization:

To evaluate the hardware efficiency of the implemented carrier based PWM techniques, FPGA resource utilization metrics were extracted using the Vivado Design Suite 2019.3. After completing the synthesis and implementation processes for each design, resource usage was obtained from the Vivado Utilization Report. Key parameters analyzed include the number of Look-Up Tables (LUTs), Flip-Flops (FFs), and Digital Signal Processing (DSP) blocks. The summarized resource utilization for each PWM technique is presented in Table III.1. For enhanced clarity and visual comparison, these results are also illustrated in Figs III.8, III.9, and III.10.

Table III.1: FPGA Resource Utilization Summary for Implemented carrier based PWM Techniques

Ressources	Converter level and PWM technique		Utilization	Utilization (%)	Available
LUT	3-level	PD	904	1.70	53200
		POD	909	1.71	
	5-level	PD	1033	1.94	
		POD	1031	1.94	
		APOD	1028	1.93	
	7-level	PD	1067	2.01	
		POD	1075	2.02	
		APOD	1069	2.01	
	9-level	PD	1112	2.09	
		POD	1120	2.11	
		APOD	1112	2.09	
FF	3-level	PD	254	0.24	106400
		POD	262	0.25	
	5-level	PD	254	0.24	
		POD	262	0.25	
		APOD	262	0.25	
	7-level	PD	254	0.24	
		POD	262	2.02	
		APOD	262	0.25	
	9-level	PD	254	2.09	
		POD	262	0.25	
		APOD	262	0.25	
DSP	3-level	PD	0	0	0
		POD			
	5-level	PD	12	5.45	220
		POD			
		APOD			
	7-level	PD	9	4.09	
		POD			
		APOD			
	9-level	PD	24	10.91	
		POD			
		APOD			

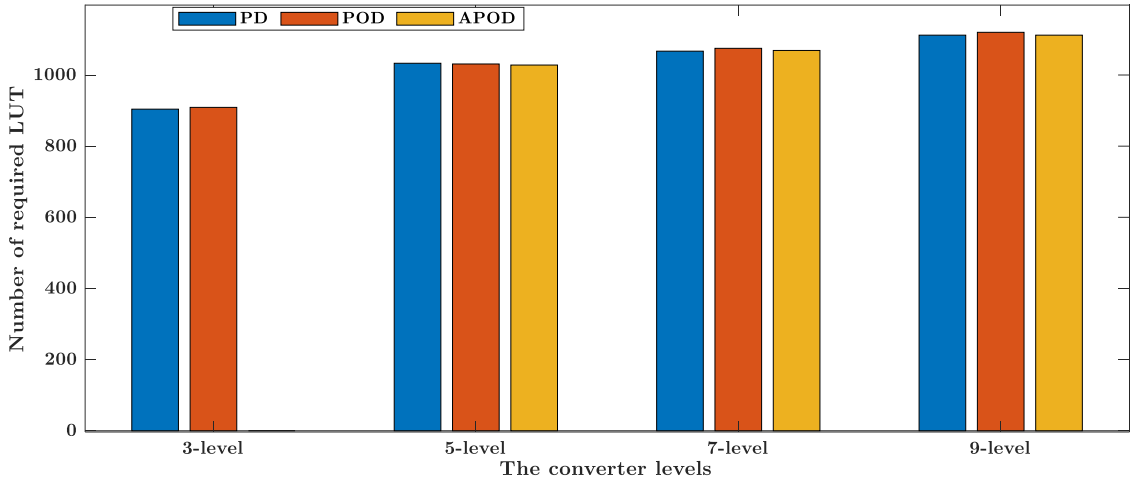


Fig III.8: Number of required LUT for each PWM technique versus converter level

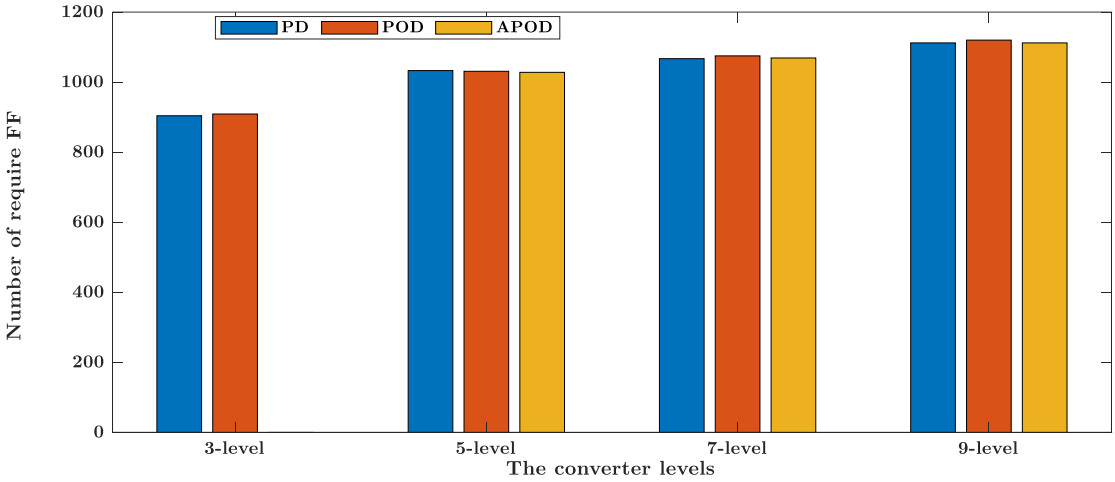


Fig III.9: Number of required FF for each PWM technique versus converter level

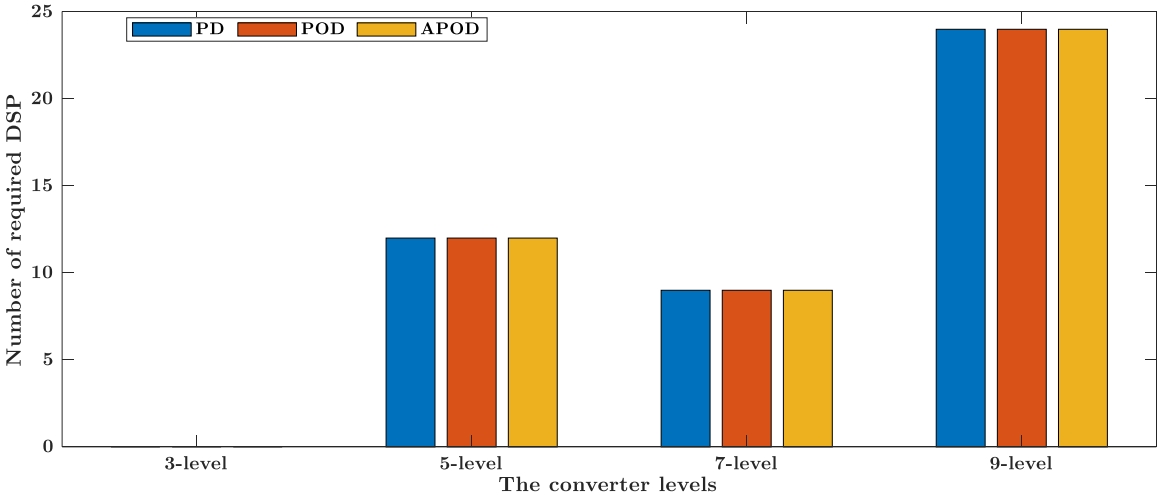


Fig III.10: Number of required DSP for each PWM technique versus converter level

Figs III.8, III.9, and III.10 illustrate the FPGA resource utilization LUT, FF, and DSP respectively for three different PWM techniques across various converter levels. Overall, the results show that resource consumption increases with the number of converter levels regardless of the PWM technique used. DSP usage starts at zero for the 3-level converters and rises significantly with higher levels, reaching approximately 24 for all techniques at the 9-level, indicating that DSP demand is more dependent on converter complexity than the modulation method. Similarly, FF and LUT utilizations show a gradual increase from the 3-level to the 9-level converters, but the differences between PD, POD, and APOD are minimal at each level. This suggests that the three PWM techniques have similar hardware implementation requirements and do not significantly affect the use of Flip-Flops or Look-Up Tables.

III.2.2. Proposed FPGA implementation of the SSVM

Fig III. 11 illustrates the proposed FPGA-based implementation of the SSVM for the *n*-level DCC, integrating both the PL and PS components. The block design is implemented using Vivado Design Suite 2019.2 with the FPGA Zedboard 7000 [13-14].

As shown in Fig III.11, the block design for implementing the SSVM algorithm on FPGA consists of the following components, showing the connection between the PL and PS parts:

A. Programmable Logic (PL)

- The *Voltages_Generation* block (written in VHDL) generates the reference voltages $\boldsymbol{v_a}^*$, $\boldsymbol{v_b}^*$, and $\boldsymbol{v_c}^*$.
- The *Switching_Period* block selects the switching period (switching frequency) through the *Select_Ts* input.

B. Processing System (PS)

- The *ZYNQ7 Processing System* represents the ARM processor of the FPGA.
- It is connected to the AXI Interconnect to enable data exchange between the PS and PL.
- The PS is responsible for performing the SSVM algorithm calculations using C code.

C. AXI GPIO Interfaces

- Several *AXI GPIO* blocks are used to transfer data between the PS and PL.
- The input voltages $\boldsymbol{v_a}^*$, $\boldsymbol{v_b}^*$, and $\boldsymbol{v_c}^*$ are sent from the PL to the PS using *AXI GPIO*.
- The calculated output voltages are sent from the PS to the PL using *axi_gpio_4*.

D. AXI Interconnect

- This module connects the AXI master from the PS to multiple AXI slaves in the PL.
- It facilitates data transfer between the PS and AXI GPIO blocks.

E. Clock and Reset System

- The Processor System Reset block ensures proper synchronization between the PS and PL using clk and reset signals.

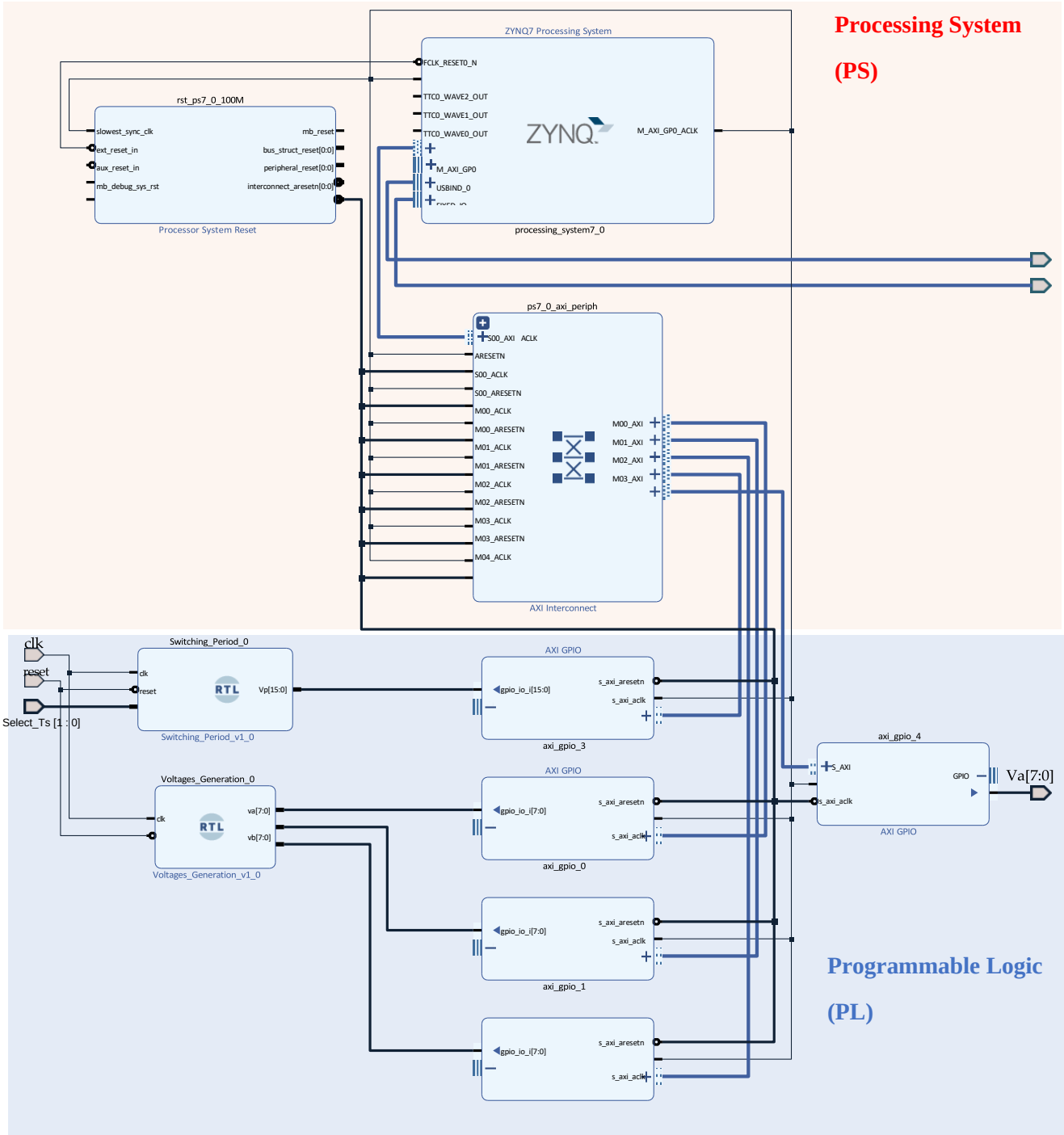


Fig III.11: Proposed block design implementation for the SSVM for *n*-level DCC using the combination of the PL and PS in the Zynq FPGA [13-14]

A detailed description of the proposed SSVM implementation block is provided below, offering an in-depth explanation of each individual component and its specific function within the overall system.

III.2.2.1. The switching period block

The Switching Period block determines the switching frequency used in the SSVM, which directly influences the quality and efficiency of the inverter's output. This block generates a 2-bit control signal that is sent to the PL section to configure the switching frequency via signals SW1 and SW2. The frequency settings are as follows: '00' corresponds to 2 kHz, '01' to 4 kHz, '10' to 5 kHz, and '11' to 10 kHz.

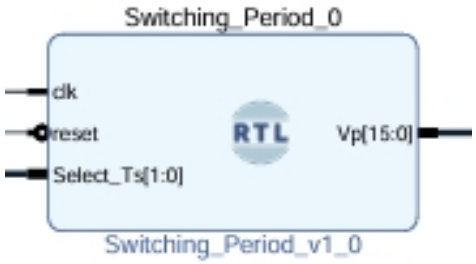


Fig III.12: Switching period block

The VHDL code of this block is given as:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.all;

entity Switching_Period is
                port (clk: in std_logic;
                        reset: in std_logic;
                       Select_Ts: in std_logic_vector(1 downto 0);
                       Vp: out std_logic_vector(15 downto 0));
end Switching_Period;
architecture Behavioral of Switching_Period is
signal v_p:integer:=0;
signal a,b:integer:=0;
signal clk_div:std_logic:='0';
begin
with Select_Ts select
   b <= 24 when "00",   -- for 2  kHz
         11 when "01",   -- for 4  kHz
          9  when "10",   -- for 5  kHz
          4  when others; -- for 10 kHz
process (clk)
begin
      if (rising_edge(clk))then
        if a=b then
```

```
            a<=0;
            clk_div<=not clk_div;
          else
            a<=a+1;
          end if;
      end if;
 end process;
-------------------------------------------------------------
process (clk_div,reset)
  begin
         if reset='1' then
        v_p<=0;
        else
        if rising_edge (clk_div) then
           if v_p=1000 then
              v_p<=0;
           else
              v_p<=v_p+1;
           end if;
          end if;
          end if;
end process;
Vp<=std_logic_vector(to_unsigned (v_p, 16));
end Behavioral;
```

III.2.2.2. AXI_gpio for three phase reference voltages

The three blocks shown in Fig. III.13 represent AXI General Purpose Input/Output (GPIO) peripherals, each configured to interface with the AXI bus in an FPGA or SoC design. Each AXI GPIO block is set up with a single 8-bit input channel, as indicated by the left-pointing arrow on the GPIO port. These inputs correspond to reference voltages generated in the PL section, which are then transferred to the PS section for further processing.

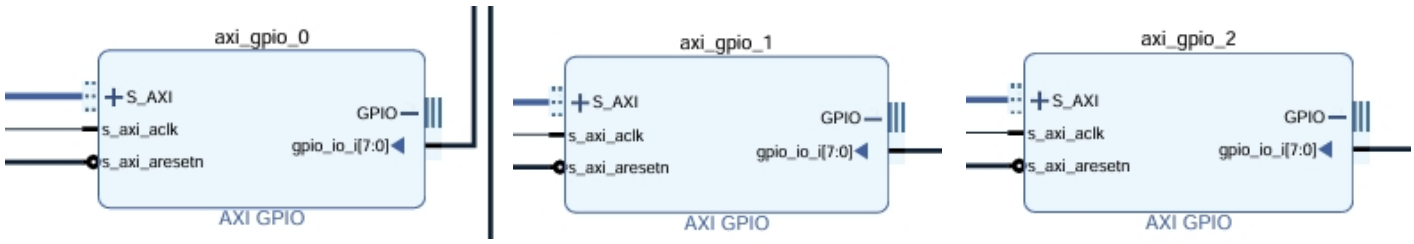


Fig III.13: AXI_gpio of the input for three phase reference voltages

III.2.2.3. AXI_gpio of the switching period

To interface the Switching Period block designed in the PL section, a 16-bit AXI GPIO input is required, as the output variable from this block is 16 bits wide. The AXI GPIO block labeled axi_gpio_3, shown in Fig. III.13, is used to transfer this variable from the PL to the PS section.

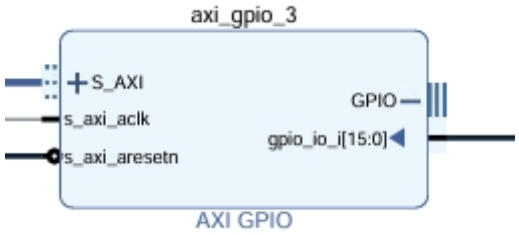


Fig III.14: AXI_gpio of the input for the switching frequency

III.2.2.4. AXI_gpio of the output voltage

The output voltage signal generated in the PS section must be transferred to the physical I/O pins through the PL section. This transfer is achieved using the AXI GPIO block labeled axi_gpio_4, as shown in Fig. III.15. This block is configured as an output GPIO, enabling the PS to send digital control signals representing the output voltage to the PL. From there, the signal can be routed to external interfaces, such as DACs or Pmod connectors, for further processing or visualization.

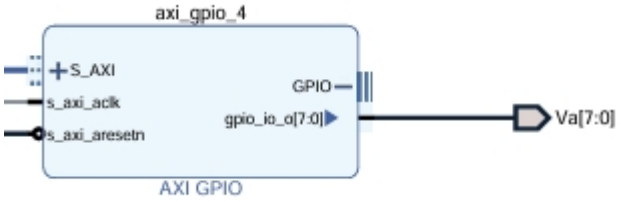


Fig III.15: AXI_gpio of the output voltage

III.2.2.5. ZYNQ7 Processing system

The ZYNQ7 Processing System integrates a dual-core ARM Cortex-A9 processor with FPGA fabric, enabling efficient hardware-software co-design. It provides interfaces for memory, USB, AXI communication, and clock/reset management for precise control. In this implementation, the Zynq processor is used to code the SSVM algorithm steps in C, simplifying calculations compared to using VHDL and improving overall design flexibility.

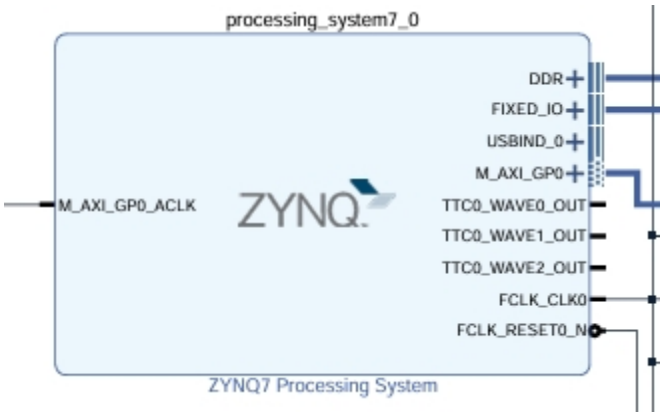


Fig III.16: ZYNQ7 Processing system

III.2.2.6. Processing system reset

The Processing System Reset block in a ZYNQ-based design manages the reset signals between the PS and the PL. When using SSVM, precise synchronization between control algorithms (typically running on the PS) and modulation logic (implemented in PL) is crucial. This block ensures that all modules start from a known state, aligning clock domains and preventing metastability. It generates reset outputs based on PS status and external reset inputs, maintaining system stability during initialization or fault recovery.

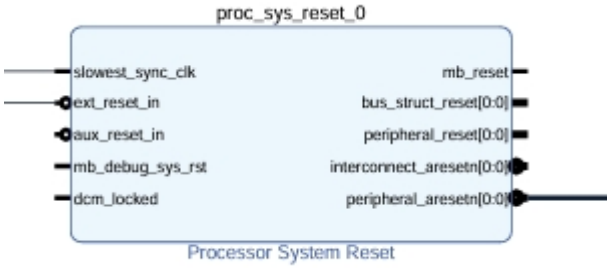


Fig III.17: Processing System Reset

III.2.2.7. AXI_Interconect

The AXI Interconnect block facilitates communication between the PS and multiple AXI-based peripherals, such as AXI_GPIO. It acts as a bridge, routing data between a single AXI master (e.g., PS) and multiple AXI slave interfaces. In the context of SSVM, this interconnect ensures fast, efficient data transfer from the PS to control peripherals like GPIO, timers, and PL part.

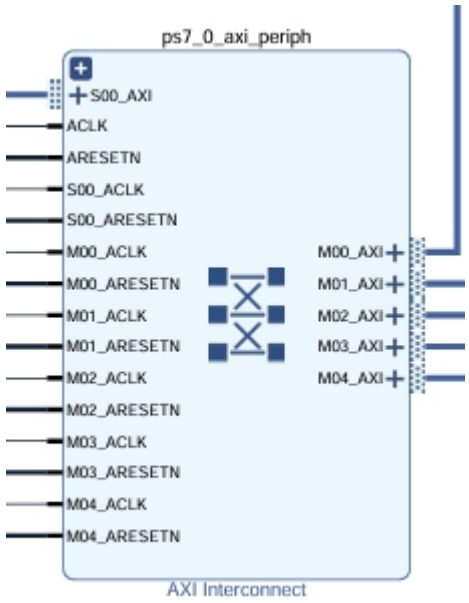


Fig III.18: AXI Interconnect

Below is an example of the C code implementation of the SSVM algorithm for a 3-level DCC.

```
#include <stdio.h>
#include "platform.h"
#include "xgpio.h"
#include "xparameters.h"
#include "xil_printf.h"
#include <math.h> // For mathematical operations if floating-point is needed
int main() {
    init_platform();

    XGpio va_in, vb_in, vc_in, Ts_in, V_out,m;
    int8_t Va, Vb, Vc;
    int Sector, Delta, SS1a, SS2a, SS1b, SS2b, SS1c, SS2c, S1a, S2a, S1b, S2b, S1c, S2c;
    float Ug, Uh, d1, d2, d3, Ts;
    float d_1a_off, d_2a_off, d_1b_off, d_2b_off, d_1c_off, d_2c_off;
    float va_out, va_out_0, vb_out_0, vc_out_0;
    const int vdc = 180;
    const int n = 3;
    int G, H, Tss,M;
    int V1_1a, V1_2a, V1_1b, V1_2b, V1_1c, V1_2c;
    int V2_1a, V2_2a, V2_1b, V2_2b, V2_1c, V2_2c;
    int V3_1a, V3_2a, V3_1b, V3_2b, V3_1c, V3_2c;
    int Fa1, Fa2, Fb1, Fb2, Fc1, Fc2;
    float va, vb, vc, Vaa,Vbb,Vcc;

    // Initialize GPIO for inputs va, vb, vc and output sector
    XGpio_Initialize(&va_in, XPAR_AXI_GPIO_0_DEVICE_ID);  // Input GPIO for va Reference voltage va
```

```
    XGpio_Initialize(&vb_in, XPAR_AXI_GPIO_1_DEVICE_ID);  // Input GPIO for vb Reference voltage vb
    XGpio_Initialize(&vc_in, XPAR_AXI_GPIO_2_DEVICE_ID);  // Input GPIO for vc Reference voltage vc
    XGpio_Initialize(&Ts_in, XPAR_AXI_GPIO_3_DEVICE_ID);  // Input GPIO for Ts Switching period
    XGpio_Initialize(&m, XPAR_AXI_GPIO_5_DEVICE_ID);  // Input GPIO for Ts Switching period

    // Initialize GPIO for output sector
    XGpio_Initialize(&V_out, XPAR_AXI_GPIO_4_DEVICE_ID); // Output GPIO for output converter
voltage
    // Set directions inputs
    XGpio_SetDataDirection(&va_in, 1, 0xFF);
    XGpio_SetDataDirection(&vb_in, 1, 0xFF);
    XGpio_SetDataDirection(&vc_in, 1, 0xFF);
    XGpio_SetDataDirection(&Ts_in, 1, 0xFFFF); // 16 bit
    XGpio_SetDataDirection(&m, 1, 0x01); // 16 bit
    // Set directions outputs
    XGpio_SetDataDirection(&V_out, 1, 0x00);

    while (1) {
       // Read input values for va, vb, vc (8-bit each)
       Va = XGpio_DiscreteRead(&va_in, 1);
       Vb = XGpio_DiscreteRead(&vb_in, 1);
       Vc = XGpio_DiscreteRead(&vc_in, 1);
       Tss = XGpio_DiscreteRead(&Ts_in, 1);
       M = XGpio_DiscreteRead(&m, 1);
       Ts = (float)Tss;
     if(M==0){
            Vaa=0.5*(float)Va;
            Vbb=0.5*(float)Vb;
            Vcc=0.5*(float)Vc;
     }else{
            Vaa=(float)Va;
            Vbb=(float)Vb;
            Vcc=(float)Vc;
     }
// Sector Identification ------------------------------------------------
       if ((Vaa >= Vbb) && (Vbb > Vcc)) { Sector = 1; }
       else if ((Vaa >= Vcc) && (Vcc > Vbb)) { Sector = 6; }
       else if ((Vbb >= Vaa) && (Vaa > Vcc)) { Sector = 2; }
       else if ((Vbb >= Vcc) && (Vcc > Vaa)) { Sector = 3; }
       else if ((Vcc >= Vbb) && (Vbb > Vaa)) { Sector = 4; }
       else { Sector = 5; }
       // New Reference Voltage Vector identification ---------------------------
       if ((Sector == 1) || (Sector == 6)) { va = Vaa; }
       else if ((Sector == 2) || (Sector == 3)) { va = Vbb; }
       else if ((Sector == 4) || (Sector == 5)) { va = Vcc; }
       if ((Sector == 2) || (Sector == 5)) { vb = Vaa; }
       else if ((Sector == 1) || (Sector == 4)) { vb = Vbb; }
       else if ((Sector == 3) || (Sector == 6)) { vb = Vcc; }

       if ((Sector == 3) || (Sector == 4)) { vc = Vaa; }
       else if ((Sector == 5) || (Sector == 6)) { vc = Vbb; }
       else if ((Sector == 1) || (Sector == 2)) { vc = Vcc; }
       //  G,H components
       Ug = (va - vb) * (n - 1) / vdc;
       Uh = (vb - vc) * (n - 1) / vdc;
       G = (int)Ug;
```

```
H = (int)Uh;
// select Triangles(1,2,3,4) of sector 1
if ((G == 0) && (H == 0)) {
   if (Ug + Uh < G + H + 1) {
      Delta = 1; //Triangle 1
   }
   else {
      Delta = 3;
   } //Triangle 3
}
else if ((G == 1) && (H == 0)) {
   Delta = 2; //Triangle 2
}
else {
   Delta = 4; //Triangle 4
}
// select Duration time calculation
if (Delta == 1) {
   d2 = (Ug - G) * 1000;
   d3 = (Uh - H) * 1000;
   d1 = 1000 - d2 - d3;

   V1_2a = 1;
   V1_1a = 1;

   V1_2b = 1;
   V1_1b = 1;

   V1_2c = 1;
   V1_1c = 1;

   V2_2a = 1; V2_1a = 1; V2_2b = 1; V2_1b = 0; V2_2c = 1; V2_1c = 0;
   V3_2a = 1; V3_1a = 1; V3_2b = 1; V3_1b = 1; V3_2c = 1; V3_1c = 0;
}
else if (Delta == 2) {
   d2 = (Ug - G) * 1000;
   d3 = (Uh - H) * 1000;
   d1 = 1000 - d2 - d3;
   V1_2a = 1; V1_1a = 1; V1_2b = 1; V1_1b = 0; V1_2c = 1; V1_1c = 0;
   V2_2a = 1; V2_1a = 1; V2_2b = 0; V2_1b = 0; V2_2c = 0; V2_1c = 0;
   V3_2a = 1; V3_1a = 1; V3_2b = 1; V3_1b = 0; V3_2c = 0; V3_1c = 0;
}
else if (Delta == 4) {
   d2 = (Ug - G) * 1000;
   d3 = (Uh - H) * 1000;
   d1 = 1000 - d2 - d3;
   V1_2a = 1; V1_1a = 1; V1_2b = 1; V1_1b = 1; V1_2c = 1; V1_1c = 0;
   V2_2a = 1; V2_1a = 1; V2_2b = 1; V2_1b = 0; V2_2c = 0; V2_1c = 0;
   V3_2a = 1; V3_1a = 1; V3_2b = 1; V3_1b = 1; V3_2c = 0; V3_1c = 0;
}
else {
   d2 = (H + 1 - Uh) * 1000;
   d3 = (G + 1 - Ug) * 1000;
   d1 = 1000 - d2 - d3;
   V1_2a = 1; V1_1a = 1; V1_2b = 1; V1_1b = 0; V1_2c = 0; V1_1c = 0;
   V2_2a = 1; V2_1a = 1; V2_2b = 1; V2_1b = 0; V2_2c = 1; V2_1c = 0;
```

```
    V3_2a = 1; V3_1a = 1; V3_2b = 1; V3_1b = 1; V3_2c = 1; V3_1c = 0;
}
// d_a_off=d1/2+d2+d3;
 //d_b_off=d3+d1/2;
 //d_c_off=d1/2;
d_2a_off = (V1_2a * d1 + V2_2a * d2 + V3_2a * d3);
d_1a_off = (V1_1a * d1 + V2_1a * d2 + V3_1a * d3);
d_2b_off = (V1_2b * d1 + V2_2b * d2 + V3_2b * d3);
d_1b_off = (V1_1b * d1 + V2_1b * d2 + V3_1b * d3);
d_2c_off = (V1_2c * d1 + V2_2c * d2 + V3_2c * d3);
d_1c_off = (V1_1c * d1 + V2_1c * d2 + V3_1c * d3);

if (Ts < d_2a_off) { S2a = 1; } else { S2a = 0; }
if (Ts < d_1a_off) { S1a = 1; } else { S1a = 0; }
if (Ts < d_2b_off) { S2b = 1; } else { S2b = 0; }
if (Ts < d_1b_off) { S1b = 1; } else { S1b = 0; }
if (Ts < d_2c_off) { S2c = 1; } else { S2c = 0; }
if (Ts < d_1c_off) { S1c = 1; } else { S1c = 0; }
//  if (Ts<d_b_off) {Sb=1;} else {Sb=0;}
//  if (Ts<d_c_off) {Sc=1;} else {Sc=0;}
if (Sector == 1) {
    SS1a = S1a;
    SS2a = S2a;
    SS1b = S1b;
    SS2b = S2b;

    SS1c = S1c;
    SS2c = S2c;
}
else if (Sector == 2)
{
    SS1a = S1b;
    SS2a = S2b;
    SS1b = S1a;
    SS2b = S2a;
    SS1c = S1c;
    SS2c = S2c;
}
else if (Sector == 3)
{
    SS1a = S1c;
    SS2a = S2c;
    SS1b = S1a;
    SS2b = S2a;
    SS1c = S1b;
    SS2c = S2b;
}
else if (Sector == 4)
{
    SS1a = S1c;
    SS2a = S2c;
    SS1b = S1b;
    SS2b = S2b;
    SS1c = S1a;
    SS2c = S2a;
}
```

```
    else if (Sector == 5)
    {   SS1a = S1b;
        SS2a = S2b;
        SS1b = S1c;
        SS2b = S2c;
        SS1c = S1a;
        SS2c = S2a; }
    else if (Sector == 6)
    {  SS1a = S1a;
        SS2a = S2a;
        SS1b = S1c;
        SS2b = S2c;
        SS1c = S1b;
        SS2c = S2b; }
    //---------------------------------------------- Output voltage calculation
    //va_out=vdc_3*(2*SSa-SSb-SSc)+100;
    Fa2 = SS1a * SS2a;
    Fb2 = SS1b * SS2b;
    Fc2 = SS1c * SS2c;
    Fa1 = (1 - SS1a) * SS2a;
    Fb1 = (1 - SS1b) * SS2b;
    Fc1 = (1 - SS1c) * SS2c;
    va_out_0 = vdc * Fa2 + Fa1 * vdc / 2;
    vb_out_0 = vdc * Fb2 + Fb1 * vdc / 2;
    vc_out_0 = vdc * Fc2 + Fc1 * vdc / 2;
    va_out = (2 * va_out_0 - vb_out_0 - vc_out_0) / 3 + 125;
    //   vb_out= (-va_out_0 +2*vb_out_0 - vc_out_0)/3 ;
    //   vc_out= (-va_out_0 - vb_out_0 + 2*vc_out_0)/3;
    XGpio_DiscreteWrite(&V_out, 1, va_out); }
  cleanup_platform();
  return 0;}
```

III.2.2.7. Resources utilization

To assess the hardware efficiency of the implemented SSVM, FPGA resource utilization metrics were obtained using the Vivado Design Suite 2019.3. After completing synthesis and implementation for each design, resource usage data was extracted from the Vivado Utilization Report. Key parameters analyzed include the number of Look-Up Tables (LUTs), Look Up Tables RAMs (LUTRAM), Flip-Flops (FFs), Block RAMs (BRAMs), Input/Output (IO) ports, and global clock buffers (BUFGs). A summary of the resource utilization for the SSVM technique is provided in Table III.2. Since the SSVM algorithm remains identical across all DCC levels in Vivado (despite differences in their C code implementations in Vitis), the resource usage remains consistent and matches the data presented in Table III.2.

Table III.2: FPGA Resource Utilization Summary for Implemented SSVM

Ressources	Utilization	Utilization (%)	Available
LUT	1131	2.13	53200
LUTRAM	62	0.36	17400
FF	1476	1.39	106400
BRAM	0.50	0.36	140

III.3. Experimental Result

This section presents the experimental results of carrier-based PWM techniques as well as the SSVM technique. The same parameters used in the simulation are applied here: the switching frequency is set to 5 kHz, the DC-link voltage to 200 V, and the modulation index of the reference voltages to 0.8.

After generating the bitstream and programming the FPGA, the output voltage of the *n*-level converter is modulated within the FPGA and represented as an 8-bit digital signal. This digital output is then converted to an analog signal using a Digital-to-Analog Converter (DAC), allowing the *n*-level output voltage to be visualized on an oscilloscope (see Fig. III.19).

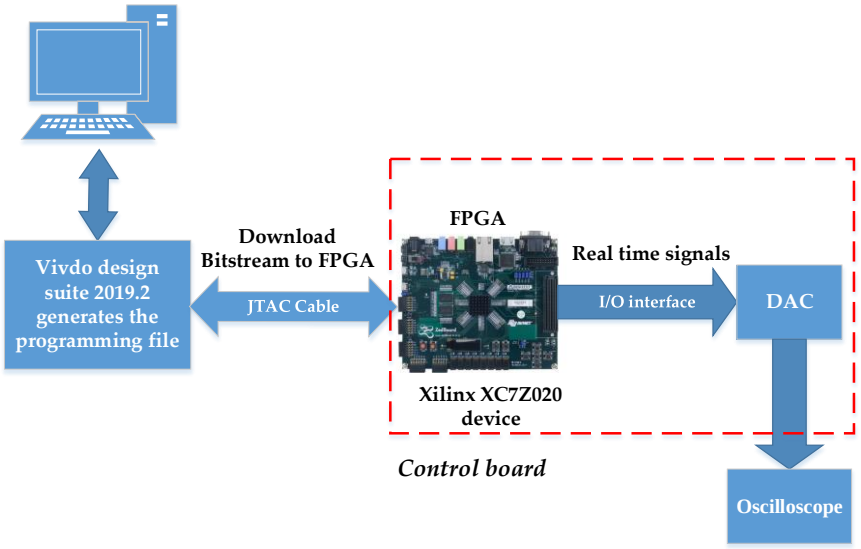


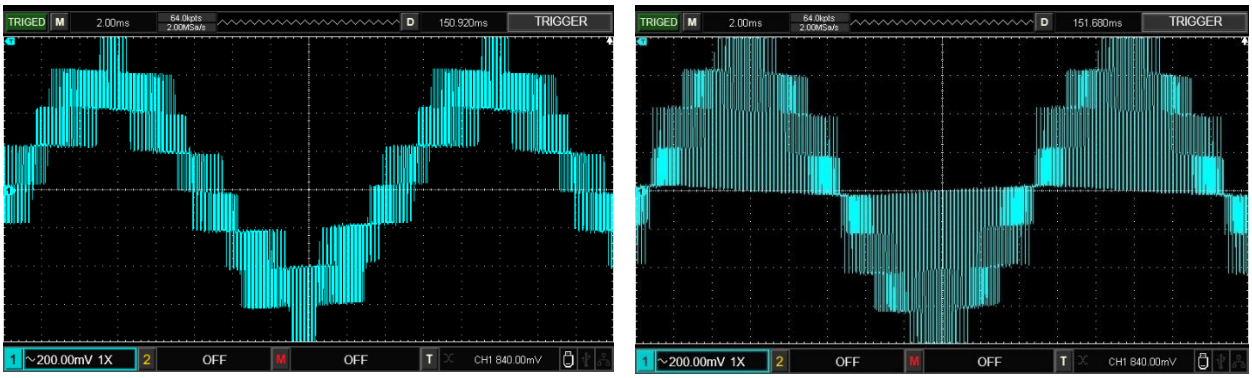
Fig III.19: Block diagram of the experimental setup

Figs. III.20, III.21, III.22, and III.23 show the experimental output voltage waveforms of the 3-level, 5-level, 7-level, and 9-level DCCs, respectively. For each converter level, the output waveforms generated by all the considered PWM techniques are presented together, allowing a clear comparison of their effects on the output voltage shape.

The experimental output voltage waveforms closely match the mathematical analysis and are consistent with the simulation results, thereby validating the accuracy and effectiveness of the implemented modulation techniques. Similar to the observations made during simulation, the output voltage generated using the SSVM technique shows significant improvement compared to those obtained using carrier-based PWM methods across all converter levels.

Furthermore, as the number of levels in the DCC increases, the quality of the output voltage waveform improves accordingly. This enhancement is evident in the reduced harmonic distortion and smoother waveform shape. Among all tested configurations, the 9-level DCC using the SSVM technique delivers the most refined and high-quality output voltage waveform, demonstrating the superior performance of the proposed method in generating near-sinusoidal output voltages.

(a) PD for 3-level DCC **(b) POD for 3-level DCC**



(c) SSVM for 3-level DCC

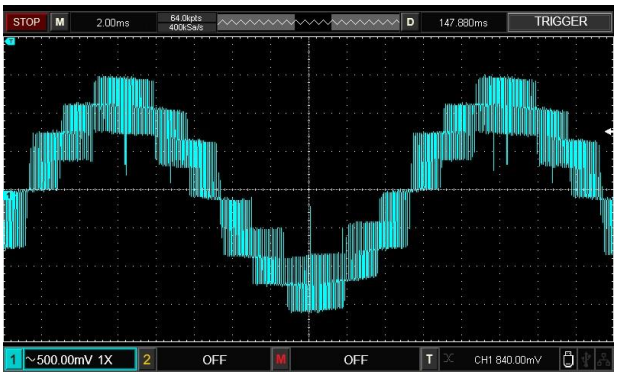
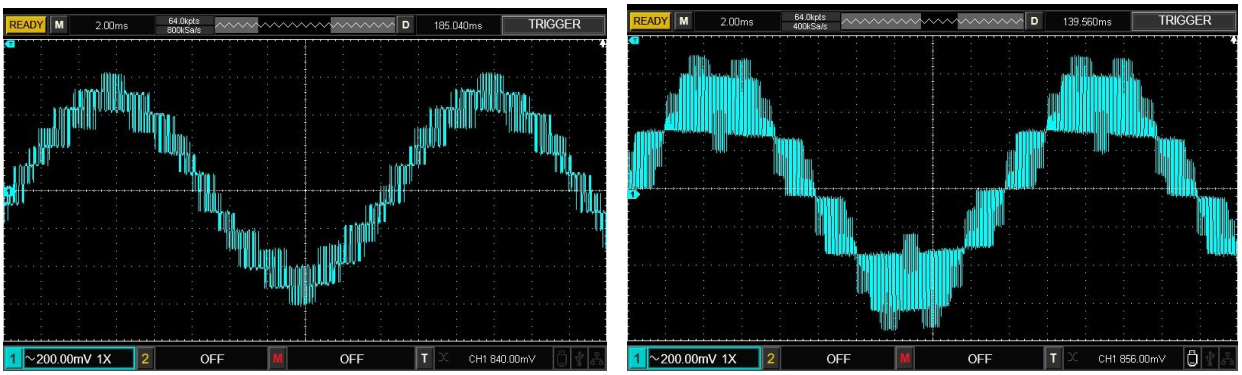


Fig III.20: Experimental results of the output voltage of the 3-level DCC using different PWM techniques, (a): PD, (b): POD, and (c): SSVM

(a) PD for 5-level DCC **(b) POD for 5-level DCC**



(c) APOD for 5-level DCC **(d) SSVM for 5-level DCC**

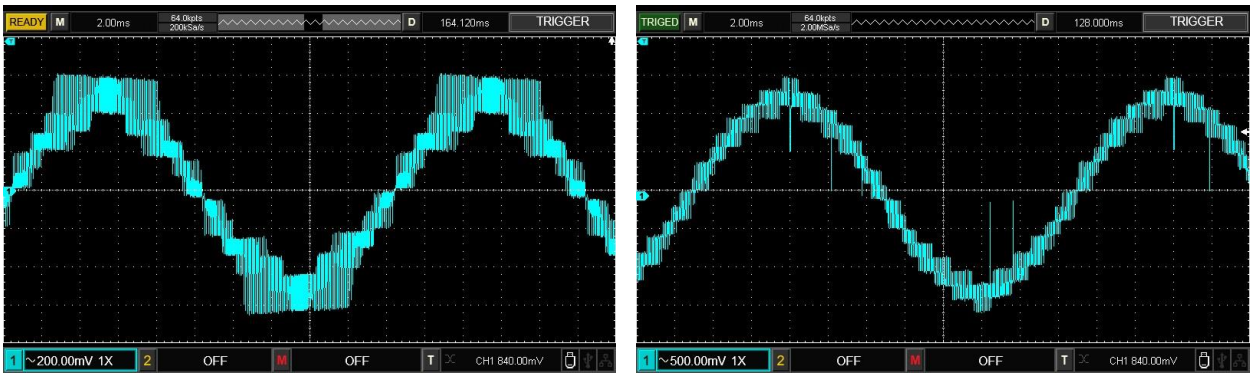
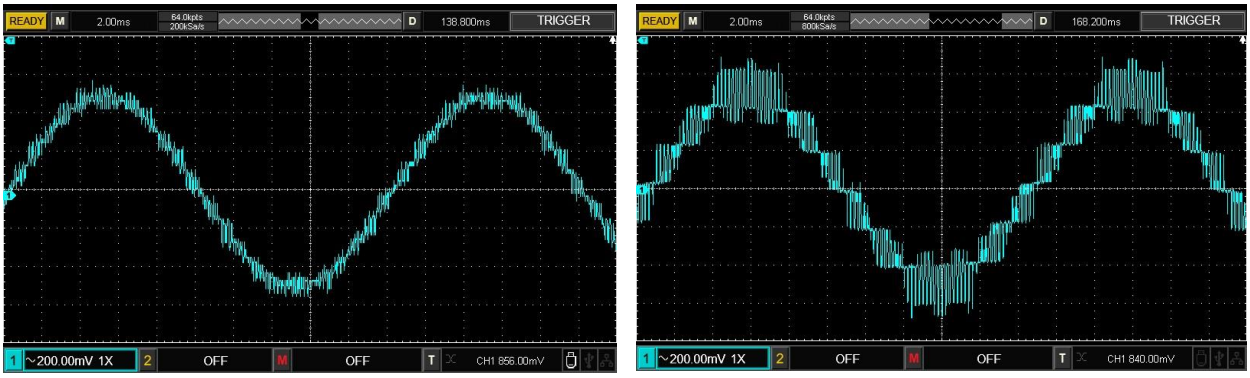


Fig III.21: Experimental results of the output voltage of the 5-level DCC using different PWM techniques, (a): PD, (b): POD, (c): APOD and (d): SSVM

(a) PD for 7-level DCC **(b) POD for 7-level DCC**



(c) APOD for 7-level DCC **(d) SSVM for 7-level DCC**

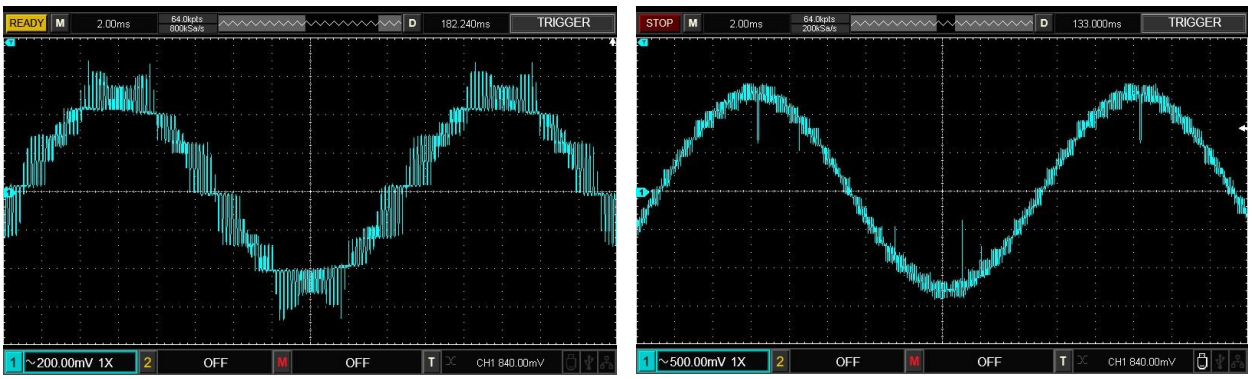


Fig III.22: Experimental results of the output voltage of the 7-level DCC using different PWM techniques, (a): PD, (b): POD, (c): APOD and (d): SSVM

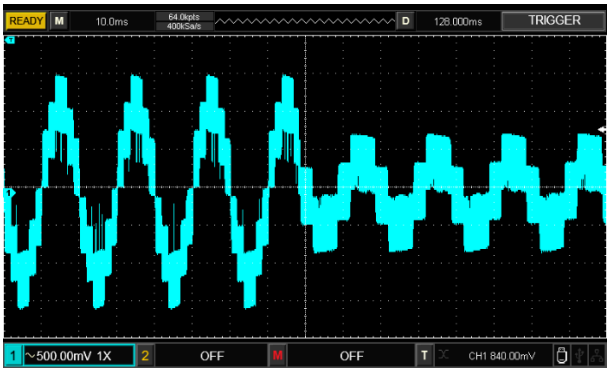
(a) PD for 9-level DCC **(b) POD for 9-level DCC**

(c) APOD for 9-level DCC **(d) SSVM for 9-level DCC**

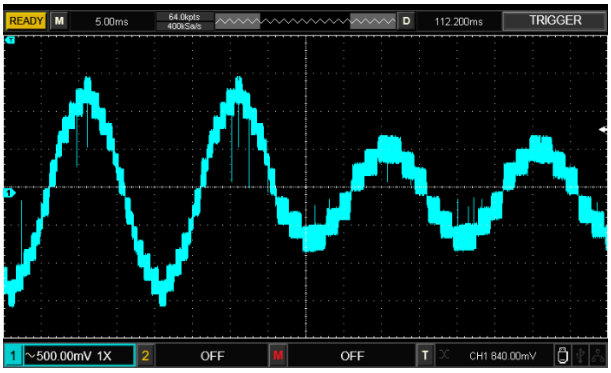
Fig III.23: Experimental results of the output voltage of the 9-level DCC using different PWM techniques, (a): PD, (b): POD, (c): APOD and (d): SSVM

Fig III.24 presents the output voltage waveforms of the multilevel DCC under a step change in the modulation index, varying from 0.8 to 0.4. The results demonstrate the system's dynamic response and its capability to generate the desired voltage levels with the SSVM technique.

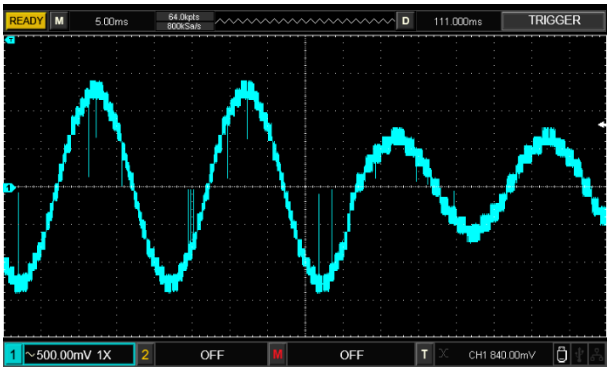
(a) SSVM for 3-level DCC



(b) SSVM for 5-level DCC



(c) SSVM for 7-level DCC



(d) SSVM for 9-level DCC

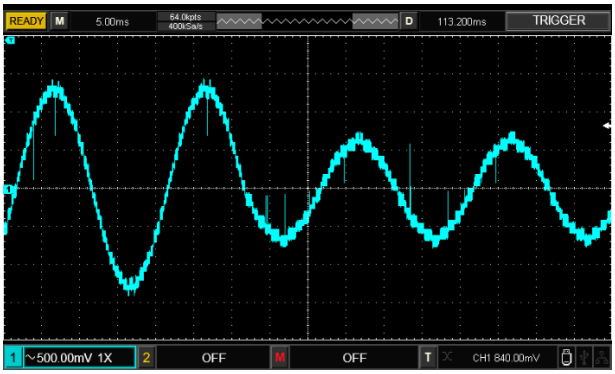


Fig III.24: Experimental results of the output voltage for the multilevel DCC using the SSVM technique under a step change in the modulation index.

III.6. Conclusion

This chapter has presented a comprehensive study on the implementation and performance evaluation of various modulation techniques for DCCs, including PD, POD, APOD, and SSVM. The carrier-based PWM techniques—PD, POD, and APOD—were implemented exclusively in the Programmable Logic (PL) section of the FPGA. Detailed explanations of each block design were provided, along with the corresponding VHDL code used for implementation.

For the SSVM technique, a novel FPGA-based architecture was proposed, utilizing both the PL and Processing System (PS) sections of the FPGA. This hybrid implementation was designed to simplify the development process and enhance computational efficiency. The three-phase reference voltages were generated in the PL section using VHDL, while the SSVM algorithm computations were carried out in the PS section using C code. This approach demonstrated the flexibility, modularity, and scalability of the proposed design.

The modulation techniques were implemented and experimentally validated on 3-level, 5-level, 7-level, and 9-level DCCs. The experimental results confirmed that the SSVM technique consistently outperformed the traditional carrier-based methods in terms of harmonic performance, with the performance advantage becoming more pronounced as the converter level increased. These findings highlight the effectiveness of the proposed SSVM-based implementation for improving output waveform quality in multilevel power converters.

General conclusion and future work

General conclusion

Multilevel inverters are well-suited for high-voltage applications due to their capability to generate output voltage waveforms with improved harmonic performance while achieving higher voltage levels using devices with limited voltage ratings.

This thesis has presented a comprehensive study on the implementation and performance evaluation of various modulation techniques for DCCs, including PD, POD, APOD, and SSVM. A novel FPGA-based design was proposed for the SSVM, leveraging the PL and PS parts of the FPGA to simplify the implementation process and enhance computational efficiency. The three-phase reference voltages were generated in the PL part using VHDL code, while the calculation steps of the SSVM were executed in the PS part using C code, demonstrating the flexibility and scalability of the proposed approach. The proposed techniques were implemented and tested for 3-level, 5-level, 7-level, and 9-level DCCs, and their performance was evaluated based on output voltage quality, with a focus on THD. The experimental results demonstrated that the SSVM technique consistently outperformed PD, POD, and APOD in terms of harmonic performance, particularly for higher level converters. The comparative analysis also highlighted the robustness of SSVM across varying modulation indices and switching frequencies, making it a suitable choice for real world applications requiring high-quality output voltages and efficient real-time control.

Future Work

The control schemes proposed in this study can be further developed and extended in various directions to enhance their performance and applicability. The following topics are suggested for future research:

- Develop advanced control strategies to regulate and balance the capacitor voltages in the DC-link of the DCC, which is crucial for stable and efficient operation.
- Investigate and implement other PWM strategies for the DCC to improve its performances.
- Design and test a real-time hardware prototype of the DCC to validate the proposed control schemes under practical operating conditions.
- Apply and adapt the presented control and PWM techniques to other multilevel converter structures likes modular multilevel converters.
- Investigate the performance of the proposed schemes when interfaced with renewable energy sources such as photovoltaic or wind systems.

References

[1] C. Dhanamjayulu, S. Padmanaban, V. K. Ramachandaramurthy, J. B. Holm-Nielsen, and F. Blaabjerg, "Design and Implementation of Multilevel Inverters for Electric Vehicles," *IEEE Access*, vol. 9, pp. 317–338, 2021, doi: 10.1109/ACCESS.2020.3046493.

[2] A. Poorfakhraei, M. Narimani, and A. Emadi, "A review of multilevel inverter topologies in electric vehicles: Current status and future trends," *IEEE Open J. Power Electron.*, vol. 2, pp. 155–170, 2021, doi: 10.1109/OJPEL.2021.3063550.

[3] H. Abu-Rub, M. Malinowski, and K. Al-Haddad, *Power Electronics for Renewable Energy Systems, Transportation and Industrial Applications*, vol. 9781118634. Wiley-IEEE Press, 2014.

[4] A. R. Ansari, A. Shahzad Siddiqui, A. Sarwar, M. Khursheed, and D. D. Sharma, "Comparison Analysis of Performance of different PWM techniques used in Multilevel Inverters," *2024 Second Int. Conf. Comput. Charact. Tech. Eng. & Sci.*, pp. 1–7, Nov. 2024, doi: 10.1109/IC3TES62412.2024.10877520.

[5] S. Chatterjee and A. Das, "A review on technological aspects of different PWM techniques and its comparison based on different performance parameters," *Int. J. Circuit Theory Appl.*, vol. 51, no. 5, pp. 2446–2498, May 2023, doi: 10.1002/CTA.3513;PAGE:STRING:ARTICLE/CHAPTER.

[6] A. Poorfakhraei, M. Narimani, and A. Emadi, "A Review of Modulation and Control Techniques for Multilevel Inverters in Traction Applications," *IEEE Access*, vol. 9, pp. 24187–24204, 2021, doi: 10.1109/ACCESS.2021.3056612.

[7] V. Jayakumar, B. Chokkalingam, and J. L. Munda, "A comprehensive review on space vector modulation techniques for neutral point clamped multi-level inverters," *IEEE Access*, vol. 9, pp. 112104–112144, 2021, doi: 10.1109/ACCESS.2021.3100346.

[8] M. Bouzidi, S. Barkat, and A. Krama, "New Simplified and generalized Three-Dimensional Space Vector Modulation Algorithm for Multilevel Four-Leg Diode Clamped Converter," *IEEE Trans. Ind. Electron.*, vol. 68, no. 10, pp. 9908–9918, 2021, doi: 10.1109/TIE.2020.3026298.

[9] M. Bouzidi, S. Barkat, and A. Krama, "Simplified hybrid space vector modulation for multilevel diode clamped converter," *IET Power Electron.*, vol. 13, no. 17, pp. 3861–3870, Dec. 2020, doi: 10.1049/iet-pel.2020.0529.

[10] R. Woods, J. McAllister, Y. Yi, and G. Lightbody, "Detailed FPGA Implementation Techniques," *FPGA-based Implement. Signal Process. Syst.*, pp. 116–139, Mar. 2017, doi: 10.1002/9781119079231.CH6.

[11] R. Mahajan, D. Sakhare, and R. Gadgil, "Review of Artificial Intelligence Applications and Architectures," *Artif. Intell. Appl. Reconfigurable Archit.*, pp. 25–34, Jan. 2023, doi: 10.1002/9781119857891.CH2.

[12] "Zynq 7000 SoC Technical Reference Manual • Zynq 7000 SoC Technical Reference Manual (UG585) • Reader • AMD Technical Information Portal." https://docs.amd.com/r/en-US/ug585-zynq-7000-SoC-TRM/Zynq-7000-SoC-Technical-Reference-Manual (accessed May 24, 2025).

[13] Mohammed Taha Dlili, Ayoub Dehikel, and Mansour Bouzidi "Advanced FPGA Implementation and Comparative Investigation of PWM Control Strategies for Multilevel Converters," in *The first International Conference on Engineering and Advanced Technologies ICEAT'2025 April 27-29, 2025, Mila, Algeria.*, pp. 1–13.

[14] Mohammed Taha Dlili, Ayoub Dehikel, Mansour Bouzidi, and Said Barkat "Advanced FPGA Implementation of the Simplified Space Vector Modulation for Multilevel Converters," in *The First National Conference on Renewable Energies and Advanced Electrical Engineering, NC-REAEE'25, Mai 06-07th, 2025, M'sila, Algeria.*, pp. 1–6.

Papers written during this research work

As part of this research, two conference papers were authored and presented at national and international conferences.

1- **Dlili Mohammed Taha, Dehikel Ayoub and Bouzidi Mansour**, "*Advanced FPGA Implementation and Comparative Investigation of PWM Control Strategies for Multilevel Converters*" The first International Conference on Engineering and Advanced Technologies, ICEAT'2025, April 27-29, 2025, Mila, Algeria

2- **Dehikel Ayoub, Dlili Mohammed Taha, Bouzidi Mansour, and Barkat Said** "*Advanced FPGA Implementation of the Simplified Space Vector Modulation for Multilevel Converters*" The First National Conference on Renewable Energies and Advanced Electrical Engineering NC-REAEE'25, Mai 06-07th, 2025, M'sila, Algeria.

Abstract

This thesis presents the synthesis and implementation of various modulation techniques for multilevel diode-clamped converters (DCC) using Field-Programmable Gate Arrays (FPGAs). The investigated techniques include Phase Disposition (PD), Phase Opposition Disposition (POD), Alternative Phase Opposition Disposition (APOD), and a Simplified Space Vector Modulation (SSVM). A novel FPGA-based design is proposed for the SSVM, leveraging the Programmable Logic (PL) and Processing System (PS) parts of the FPGA. The three-phase reference voltages are generated in the PL part using VHDL code, while the calculation steps of the SSVM are executed in the PS part using C code, simplifying the coding process and enhancing computational efficiency.

The presented techniques are implemented for 3-level, 5-level, 7-level, and 9-level DCCs, and their performance is evaluated through experimental results. A comparative analysis is conducted based on the output voltage quality, with a focus on THD.

Keywords: PWM Techniques, FPGA implementation, Space vector modulation, Multilevel inverter.

الملخص

تقدم هذه الرسالة تصميم وتنفيذ تقنيات تعديل مختلفة لمبدلات الجهد متعددة المستويات من نوع المشبك الثنائي (DCC) باستخدام المصفوفات المنطقية القابلة للبرمجة (FPGAs). تشمل التقنيات المدروسة كلاً من تقنية الترتيب المرحلي(PD) ، الترتيب المرحلي المعاكس(POD) ، الترتيب المرحلي المعاكس البديل(APOD) ، وتقنية التعديل الاتجاهي المبسطة (SSVM).تم اقتراح تصميم جديد يعتمد على FPGA لتقنية SSVM ، يتم فيه استغلال كل من الجزء المنطقي القابل للبرمجة (PL) والنظام المعالِج (PS) من شريحة الـ FPGA. يتم توليد إشارات الجهد المرجعي ثلاثية الطور في الجزء PLباستخدام كود VHDL، بينما يتم تنفيذ خطوات حساب SSVM في الجزء PS باستخدام كود بلغة C، مما يبسط عملية البرمجة ويعزز الكفاءة الحسابية.

تم تنفيذ التقنيات المقدمة على مبدلات DCC ذات 3، 5، 7، و9 مستويات، وتم تقييم أدائها من خلال نتائج تجريبية. كما أُجري تحليل مقارن بناءً على جودة الجهد الخارج، مع التركيز على التشوه التوافقي الكلي (THD).

الكلمات المفتاحية : تقنيات تعديل عرض النبضة (PWM) ، تنفيذ باستخدام FPGA ، التعديل الاتجاهي الفضائي، مبدل متعدد المستويات.