CHAPTER 3

MULTILEVEL SVPWM AND FPGA HABITAT FOR HARDWARE IMPLEMENTATION

3.1 INTRODUCTION

Based on the exhaustive literature reviews on the SVPWM applicable to MLI, aiming at arriving OVM mode, balancing capacitor voltage, reducing the CMV and hysteresis vector control, the emphasis of these areas is understood. Also it is clearly found that the implementation of SVPWM for MLIs is complex and computationally intensive due to difficulty in determining the location of reference vector, calculation of on-times and determination of switching states. The performance compliance of the SVPWM can be valid only if accurate implementation is guaranteed. Earlier attempts in implementation are either complex or inaccurate. An innovative, conceited theory to enhance above mentioned problem may only result in marginal improvement. Also it is obvious that this marginal improvement can only be highlighted with accurate implementations. This chapter proposes a simple implementation algorithm for SVPWM applied to NPC-MLI.

There are few drawbacks in SVPWM implementation as described in section 2.2.1.2, which are alleviated in the proposed scheme. It presents a significantly different approach from prior attempts and provides a general solution. This simple algorithm is based on standard 2-level SVPWM for 3-level NPC-MLI and can be implemented for any level using a counter. The proposed simplified SVPWM is developed through MATLAB/Simulink system generator model and validation is done using a Xilinx family SPARTAN-III-3A XC3SD1800A-FG676 DSP-FPGA processor board for its robustness to perform over a wide range of operating conditions through a 2kW NPC-MLI laboratory prototype. Theoretical
analysis, design and experimental results are presented in this chapter to demonstrate the feasibility of the developed SVPWM based on systematic design procedure. With developed environment, the other modified SVPWM methods (in consecutive chapters) can be implemented with minor algorithmic changes.

3.2 PROPOSED ON-TIME CALCULATION BASED ON 2-LEVEL SVPWM

The basic idea of SVPWM is to compensate the targeted/required volt-seconds using discrete switching states and their on-times. Traditionally, in order to determine the on-times of a triangle of an n-level inverter, three simultaneous equations are solved. However, classical 2-level space vector geometry can be used for on-time calculation for a multilevel SVPWM [173]. Let us first understand the on-time calculation in 2-level SVPWM.

3.2.1 On-time Calculation for 2-Level SVPWM

Figure 3.1 shows the Space Vector Diagram (SVD) of a 2-level inverter [2]. Every sector is an equilateral triangle of unity side and \( h = \sqrt{3}/2 \) is the height of a sector. Here \( V_1 \) to \( V_6 \) vectors are called as active vectors and \( V_0, V_7 \) are called as zero vectors. On-time calculation for any of the six sectors (\( S_i \)) (where \( i = 1, 2, 3, 4, 5, 6 \)) is same therefore, the operation in sector-1 may be considered to understand the SVD.

![Figure 3.1. SVD for 2-level inverter](image-url)
The reference voltage $V^*$ is the rotating SVD form of three-phase voltage. The projection of $V^*$ in the $\alpha$-$\beta$ plane at any period will lie in the area of any one of the sector. Two vectors edge each sector and each of them relates to a fixed period of time. The time integral value of $V^*$ can be approximated by the sum of the products of the two of vectors and their time widths. For example, in Figure 3.1 reference $V^*$ lie in sector-1, which is edged by vector $V_1$ and $V_2$. Starting from time $t_0$, $V^*$ moves to $t_1$ and the relation for the time integral can be written as

$$\int_{t_0}^{t_1} V^* = T_a V_1 + T_b V_2$$

(3.1)

Where $T_a$ and $T_b$ represent the time widths for vectors $V_1$ and $V_2$ respectively, and the integration time interval $t_0$ to $t_1$ is defined as the sampling period (much less than period corresponding to one sector). By adjusting $T_a$ and $T_b$ right-hand side of this expression can be made as close as possible to left-hand side value (but still they can never be completely equivalent to each other). Thus, by keeping the inverter-switching states to constitute $V_1$ for a time period $T_a$ and $V_2$ for a time period $T_b$, the pulse pattern during the period $t_0$ to $t_1$ is obtained. The same approximation (of $V_1$ and $V_2$) is repeated for all samplings of this sector. In the same sector while the reference $V^*$ moves such that the angle $\theta$ increases then the corresponding time periods $T_a$ and $T_b$ will be calculated again instant by instant. It is worthwhile to note that when $V^*$ approaches $V_2$, the period $T_b$ will be more than $T_a$. Similarly when $V^*$ enters into next sector (sector edged by $V_2$ and $V_3$) the time integral value of $V^*$ can be approximated by the sum of the products of $V_2$ and $V_3$ states and their time widths. The same procedure will have to be repeated for all other sectors.

(3.2)

Consider the sector-1 of the 2-level SVD hexagon. If one has to realize a time averaged space vector of amplitude $V^*$ at an angle $\theta$ from the $\alpha$-$\beta$ axis, then the inverter switching must be bounded inside the sector-1 with the help of their active and zero vectors ($V_7$ or $V_0$) [2]. Let $T_a$ be the duration of time in state $V_1$ (001), $T_b$ be
the duration of time in the state $V_2 (110)$ and $T_0$ be the duration of time in the zero
space vector state (000 or 111). To derive the time durations, the time averaged
space vector can be projected on to the $\alpha$-axis and an axis orthogonal to the $\alpha$-axis
called the $\beta$-axis [2]. Projecting reference vector $V^*$ along $\alpha$-axis and $\beta$-axis results in

$$V^* T = h T$$

(3.3)

$$V^* T = h T$$

(3.4)

From the above two Equations (3.3) and (3.4), the time durations $T_a$ and $T_b$ can be estimated. Thus, from Equation (3.4)

$$V^* T = h T$$

(3.5)

Where $T_S (= 1/f_S)$ is the sampling period.

Thus, the new $T_a$ equation is redefined as

$$V^* T = h T$$

(3.6)

The sampling period is given by

$$V^* T = h T$$

(3.7)

Therefore the time spent by the zero vector state is

$$V^* T = h T$$

(3.8)

3.3 SVPWM FOR MLI

The SVPWM treats the target output sinusoidal voltage as a constant
amplitude vector rotating with the constant frequency. The constant reference
target voltage vector ($V_{ref}$) defined by $V^* = |V^*|e^{j\omega t}$, rotates about the center of the space
vector field at an angular frequency $\omega=2\pi f_{sys}$ (say $f_{sys}=50$ Hz). The basic idea of
SVM is to compensate the required volt-seconds using discrete switching states and
their on-times. In order to determine the on-times of the MLI SVD, traditionally the $T_a$, $T_b$ and $T_c$ values has to be solved by Equation (3.5-3.7) [2], [174].

A SVPWM applied for more than 2-level inverter is called MLI SVPWM. Any of the three-phase n-level SVD can be divided into 6 sectors ($S_i$), where $i = 1, 2, 3, 4, 5, 6$. Each sector is further divided into $(n-1)^2$ sub-triangles ($\Delta_{i,j}$) where $j = 0, 1, 2, 3$. The first subscript ‘i’ in the $\Delta_{i,j}$ represents the sector number (ranges from 1 to 6) and the second subscript ‘j’ represents the sub-triangle number (ranges from 1 to 4). The n-level SVPWM consist of $n^3$ switching states. Consider 3-level SVD, which consist of 27 switching states ($3^3 = 27$) and 24 sub-triangles (No. of sub-triangles in each sector is $(3-1)^2 = 4$; therefore $4 \times 6 = 24$). Figure 3.2 shows the three-phase 3-level SVD which consists of 6 sectors, 19 voltage vectors, 27 switching states and each sector contains 4 sub-triangles [98].

\[\text{Figure 3.2. SVD for 3-level MLI}\]

The 19 voltage vectors can be classified into 4-types: large vector (LV), medium vector (MV), short vector (SV) and zero vector (ZV). There are six LVs viz., (1-1-1), (11-1), (-11-1), (-1-11), (1-11), (-1-1); six MVs viz., (10-1), (01-1), (-10), (-101), (0-11), (1-10); six SVs and each SV is having two possible switching states viz., \{(100),(0-1-1)\},\{(110),(00-1)\},\{(010),(101)\},\{(011),(100)\},\{(001)
(110), (101), (0-10) and one ZV which is having three possible switching states as \{(000), (111), (-1-1-1)\}. Presence of more than one possible switching state for a particular voltage vector is called as redundancy in switching state. Hence 3-level SVD consists of six SVs and one ZV with redundancy switching state. Here switching states of vectors are represented as \(V_x\) where V implies voltage vector, x implies Small/Large/Zero/Medium voltage vector, \(i=1,2,3,4,5,6\) (example: \(V_{S1}\) is called as first small voltage vector switching stage). The redundancy switching state is shown as \(V_{xi}^k\). Here, \(k\) means redundancy switching state number (example: \(V_{S1}^1\) is called as small voltage vector 1st redundancy switching state).

The rotating reference vector \(V^*\) can lie in any of the sectors and particularly in one of the 4 sub-triangles in the sector. If the \(V^*\) lies in the 3rd sub-triangle of 1st sector \((\Delta_{1,3})\) then \(V_{S1}, V_{S2}\) and \(V_{M1}\) vectors can be used to synthesize the sample reference voltage vector \(V^*\). Let \(\delta_{VS1}, \delta_{VS2}\) and \(\delta_{VM1}\) are calculated by duty cycles of the switching vectors.

There are four main requirements to implement SVPWM for multilevel inverters [98], [105].

- Sector \(S_i\) and sub-triangles \((\Delta_{i,j})\) calculation,
- On-times of the three nearest vectors (NTVs) of the particular \(\Delta_{i,j}\).
- Switching sequence for the given switching cycle and
- The switching states in the sequence are applied for the calculated on-times to generate switching signals.

Fixing the sub-triangle holding the tip of the reference vector is difficult. Few algorithms have been developed for finding \(\Delta_{i,j}\) and its pulse pattern are reviewed in section 2.2.1.1 of chapter 2 which are complex, computationally intensive and difficult to implement. On the other hand, several programmable logic devices are available to implement SVPWM algorithm with varying degree capabilities such as microcontrollers, DSPs, FPGAs, ASICs and many others [175],[176]. A high speed, pre-programmable FPGA platform is more suitable for
modern VSD incorporating computational intensive SVPWM for large numbered switching devices (requires pulse segregation).

In this thesis, a simple algorithm to perform SVPWM for a NPC-MLI is proposed. The algorithm is based on standard 2-level SVPWM and can be implemented for any level using one counter. Since the considered SVPWM for MLI uses the basic 2-level SVM to calculate the on-times computation process for n-level inverter, it is easily understandable and simpler. The main advantage of the considered SVPWM is that it uses a simple mapping process to achieve the multilevel SVPWM.

3.3.1 Determination of Sector

The procedural application of SVPWM requires, determination of the angle \( \gamma \) and sector, \( S_i \) of \( V_{\text{ref}} \) as follows

\[
\theta = \text{rem} \left( \frac{\gamma}{60} \right)
\]

\[
S_i = \text{int} \left( \frac{\theta}{60} + 1 \right)
\]

Where ‘\( \gamma \)’ is the angle of the reference vector with respect to \( \alpha \)-axis and ‘int’ and ‘rem’ represent integer and remainder of standard functions respectively [2].

3.3.2 Determination of Sub-triangle Number

After the sector \( (S_i) \) identification, the sub-triangle determination with respect to the magnitude and rotating angle '0' of reference vector \( V^* \) is the most important one. Each sector in the 3-level inverter can be divided into four sub-triangles \( (\Delta_{ij}) \). The first subscript ‘i’ in the \( \Delta_{ij} \) represents the sector number (ranges from 1 to 6) and the second subscript ‘j’ represents the sub-triangle number (ranges from 1 to 4). The \( V^* \) can be located in any of these four sub-triangles. The objective is to identify the \( \Delta_{ij} \) in which the \( V^* \) is located. Once the \( \Delta_{ij} \) is identified, that particular \( \Delta_{ij} \) can be considered as a sector of a general 2-level SVPWM and the on-time values \( (t_i) \) are calculated.
In the determination process the objective is to identify where the $V^*$ reference is located. On-time calculation for any of the six sectors is same. Therefore, consider sector-1 for the sub-triangle identification as shown in Figure 3.3. Here the realization of $V^*$location on $\Delta_{1,3}$ and $\Delta_{1,2}$ is challenging (marked in shaded area). Hence, the search process of the triangle of the $V^*$ can be narrowed down by using two integers $K_1$ and $K_2$. They are defined by the coordinates ($V_{\alpha 0}$, $V_{\beta 0}$) of a point inscribed by the tip of $V^*$.

$\Delta_{ij}$ is identified by using two integers $X_1$ and $X_2$ and given by

$$X_1 = \text{int} \left( V_\alpha + \frac{V_\beta}{\sqrt{3}} \right)$$  \hspace{1cm} (3.11)$$

$$X_2 = \text{int} \left( \frac{V_\beta}{h} \right)$$  \hspace{1cm} (3.12)$$

In Equation (3.11), $X_1$ represents the part of the sector between the two lines joining the vertices separated by distance 'h' and inclined at 120° with respect to $\alpha$-axis. In Figure 3.3 $X_1=0$ signifies that the point $V^*$ is below the line $R_1R_2$. $X_1=1$ signifies that point $V^*$ is between the lines $R_1R_2$ and $R_3R_5$. $X_2$ represents the part of the sector between the two lines joining the vertices separated by distance 'h' and parallel to $\alpha$-axis. $X_2=0$ signifies that the point P is between the lines $R_0R_3$ and $R_2R_4$. $X_2=1$ signifies that the point $V^*$ is above the line $R_2R_4$. Geometrically, the
values of \(X_1\) and \(X_2\) are obtained at an intersection of two rectangular regions which is either a triangle or rhombus. In other words, \(V^*\) point lies in a triangle \((a)\) \(\Delta_{1,1}\) if \(X_1=0\) and \(X_2=0\), \((b)\) rhombus \(R_1\ R_3\ R_2\ R_4\) (shaded) if \(X_1=1\) and \(X_2=0\) and \((c)\) triangle \(\Delta_{1,4}\) if \(X_1=1\) and \(X_2=1\). The same analogy can be used for any level. In Figure 3.3, the reference vector is located in rhombus \(R_1\ R_3\ R_2\ R_4\). This rhombus is made up of two triangles \(\Delta_{1,2}\) and \(\Delta_{1,3}\).

The point \(V^*\) can be located in any of the two triangles. Let \((V_{\alpha}, V_\beta)\) be the co-ordinates of the point \(V^*\) with respect to the point \(R_1\) which is obtained as,

\[
V_{\alpha} = V_a - X_1 + 0.5X_2 \tag{3.13}
\]

\[
V_\beta = V_\beta - X_2 h \tag{3.14}
\]

The slope of \(R_1\ V^*\) is \(\frac{V_{\alpha}}{V_\beta}\) and the slope of diagonal \(R_1R_4\) is \(\sqrt{3}\). The triangle where point \(V^*\) is located can be determined by comparing the slope of \(R_1\ V^*\) with the slope of \(R_1R_4\). Slope comparison is done by evaluating the inequality \(V_\beta \leq \sqrt{3}V_{\alpha}\).

For \(X_1=1\) and \(X_2=0\), the common intersection is rhombus which is the combination of triangle \(\Delta_{1,2}\) and \(\Delta_{1,3}\). These two triangles can be determined by the slope comparison condition \((V_\beta \leq \sqrt{3}V_{\alpha})\). If \(V_\beta \leq \sqrt{3}V_{\alpha}\) is true, then the \(V^*\) is within the triangle \(\Delta_{1,3}\) otherwise, it is in triangle \(\Delta_{1,2}\).

For the up triangle, the triangle number \(\Delta_j\) is obtained by

\[
(3.15)
\]

and their co-ordinates are calculated by given equations

\[
(3.16)
\]

\[
(3.17)
\]

For the down triangle, the triangle number \(\Delta_j\) is obtained by
and their co-ordinates are calculated by given equations

\[ (3.18) \]

\[ (3.19) \]

\[ (3.20) \]

Therefore, it can be concluded that the triangle in a sector is found by an integer \( \Delta_{i,j} \). It is obtained by a simple logical expression. The triangle number \( \Delta_{i,j} \) is formulated to provide a simple way of arranging the triangles, leading to ease of identification and extension to any level.

### 3.3.3 Determination of On-times

Figure 3.4 illustrates the on-time calculation for a 3-level inverter SVPWM. Here each sector of a 3-level inverter can be split into four triangles (these triangles are called as sub-triangles).

![Figure 3.4. Virtual 2-level from 3-level SVPWM; a. Type-1 sub-triangle, b. Type-2 sub-triangle](image-url)
Figure 3.5. Algorithm of the simplified 3-level SVPWM
To simplify on-time calculation, these sub-triangles can be categorized into two types: type-1 and type-2. The triangle of type-1 has its base side at the bottom, as shown in Figure 3.4.a. Triangles $\Delta_{1,1}$, $\Delta_{1,3}$ and $\Delta_{1,4}$ are of type-1. The triangle of type-2 has its base side at the top as shown in Figure 3.4.b. Triangle $\Delta_{1,2}$ is of type-2.

To achieve the volt-seconds for any reference vector in a sector of a 3-level inverter, the triangle needs to be identified in which the required reference vector is located, $(V^s_{\alpha_0}, V^s_{\beta_0})$ are found later. The on-time calculations can be performed using the geometry shown in Figure 3.4.a and 3.4.b, which would result in the same on-time equations as those for a classical 2-level SVPWM using Equations (3.5)-(3.7) [87], [174]. A triangle of type-1 is similar to a sector-1 of a virtual 2-level inverter. For example, in Figure 3.4.a triangle $\Delta_{1,4}$ can be assumed to be similar to sector-1 of a 2-level inverter if $R_2$ is taken as zero vector of the virtual 2-level sector as shown in Figure 3.4.b. Vector $R_2P$ defines the small vector $Vs$ $(V^s_{\alpha_0}, V^s_{\beta_0})$. On-times $T_a (t_{R3})$, $T_b (t_{R4})$ and $T_0 (t_{R2})$ are calculated by using Equations (3.5)-(3.7), where the multiplication operations are required only for Equations (3.5) and (3.6). A triangle of type-2 is similar to a sector-1 of a virtual 2-level inverter. For example, in Figure 3.4.a, triangle $\Delta_{1,2}$ can be considered similar to sector-1 of a 2-level inverter if $R_4$ is assumed to be zero vector (see Figure 3.4.b). In this example $R_4P$ represents small vector $Vs$ $(V^s_{\alpha_0}, V^s_{\beta_0})$. On-times $T_a (t_{R2})$, $T_b (t_{R1})$ and $T_0 (t_{R4})$ are calculated by using Equations (3.5)-(3.7). Figure 3.5 summarizes the simplified 3-level SVPWM algorithm including determination of sector, sub-triangle and on-time calculations.
3.3.4 Switching Pattern of the Considered SVPWM

This section explains the switching pattern of the 3-level MLI for the full SVD. The considered SVPWM gives symmetrical type of PWM pattern as shown in Figure 3.6. The nearest three vectors (NTV) have been selected for each sub-triangle to position the reference vector. There are redundant switching states at the vertices of each sub-triangle. Due to redundant states, there could be several switching sequences for a sub-triangle.

Figure 3.6. 3-level SVD sub-triangles in sector-1

Figure 3.6 shows the NTV switching pattern of the sector-1. Here sub-triangle $\Delta_{1,1}$ contains 7 switching states, $\Delta_{1,3}$ contains 5 switching states and, $\Delta_{1,4}$ and $\Delta_{1,2}$ both contain 4 switching states. For example, the reference vector is situated at $\Delta_{1,1}$. For this triangle the following vector sequence can be formed by using available redundant states of NTV. The sequence is: [-1-1-1] - [0-1-1] - [00-1] - [000] - [100] - [110] - [111]. Here on-time of the switches has been given equally for all the vectors. The Figures 3.7 and 3.8 show the switching sequence and their on-times in SVD.
Figure 3.7. Switching pulse generation for 3-level NPC-MLI; a. Switching pulse generation- $\Delta_{1,1}$, b. Switching pulse generation- $\Delta_{1,4}$
Figure 3.8. Switching pulse generation for 3-level NPC-MLI; a. Switching pulse generation- $\Delta t_3$, b. Switching pulse generation- $\Delta t_2$
3.4 IMPLEMENTATION OF 3-LEVEL SVPWM

The performance of the considered MLI SVPWM algorithm has been investigated and simulated by using MATLAB 11.b for 3-level 12-switch NPC-MLI drive with 300V DC-link, two 100μF capacitors and 5kHz switching frequency.

3.4.1 Simulation Implementation of 3-Level SVPWM

Figure 3.9. Flow chart of 3-level SVPWM MATLAB implementation
Figure 3.9 shows the MATLAB/Simulink simulation model of 3-level MLI-SVPWM. It has five main subsystems to produce twelve pulses for the inverter.

1. The first block in Figure 3.9 is “Park’s transformation” block [2]. In this block three-phase voltage vectors \( (V_a, V_b, V_c) \) in synchronous rotating frame are converted into two-phase voltage vectors \( (V_d, V_q) \) by using the “abc-dq transformation block/Park’s Transformation” available in MATLAB/Simulink, then the two-phase voltage vectors in synchronous reference frame are shifted to stationary reference frame.

2. The \(^2\)nd block named ‘Sector & \( \gamma \) identifier’ block, which holds four sub systems namely reference vector, theta, sector and \( \gamma \). Here the voltage vector in stationary reference frame \( (V_\alpha, V_\beta) \) are converted into reference voltage vector in polar form as \( V^* \angle \alpha \) where \( V^* \) is the voltage magnitude and \( \alpha \) is the angle of the voltage vector[2]. This \( V^* \) and it’s \( \theta \) are utilized to determine the sector of the SVD from 1 to 6.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{sector_identification.png}
\caption{Sector identification- MATLAB implementation}
\end{figure}
Figure 3.11. Sector number representation

Figure 3.10 shows the sector identification in MATLAB implementation. In this the ‘0’ value is compared with the sector ending angles by using the “comparator block” in MATLAB, the output of this comparator is used to determine the sector number ‘Si’. Next from the ‘α’ value, the angle of the each sector ‘γ’ is calculated by using γ Identifier subsystem as shown in Figure 3.11. The angle of voltage vector in particular sector is denoted by ‘γ’. γ is found by subtracting the ’0’ value from the sector starting angle using function and multiplexer blocks as shown in Figure 3.12. and Figure 3.13. These Si, θ and γ values are the inputs to the third block.

Figure 3.12. Theta calculation- MATLAB implementation
3. The next block is called as local vector generator block of the main MLI-SVM simulation model, which includes four subsystems to identify the sub-triangle number and its type in each sector of the SVD. The block also calculates \( V^{\alpha}_a, V^{\beta}_a \) to calculate timing of the switches by using 2-level SVM as shown in Figure 3.9.

4. In the fourth block, the switching time calculator is used to calculate the timing of the reference voltage vector. The inputs of this block are types of the sub-triangles \( V^{\alpha}_a, V^{\beta}_a \) and the sampling time period of switching frequency \( (f_s=1/T_s) \). In the same block, sample & hold blocks are also used after \( T_a \) and \( T_b \) calculator block. The purpose of these blocks is to hold the values of \( T_a \) and \( T_b \) fixed during each sampling time period. The times \( T_0, T_a \) and \( T_b \) are obtained from the calculations, then these timings are given to the different sub-triangles based on the sub-triangle block as shown in Figure 3.14. Here the block is designed by including the control degree of freedom of redundancy switching state. Therefore, inverter should use all the switching states. For example if \( \Delta_{1,2} \) is considered to have two zero states and two on-states, the inverter first operates in zero state, next in first on-state, then in second on-state and finally it operates in zero state to achieve redundancy operation. ‘Adder’, ‘Comparator’ and ‘Multiplexer’ blocks in MATLAB are used to achieve this operation. First \( T_0/2 \) is compared with \( T_s \) and the output of the comparator is given to the multiplexer. Then \( T_a \) is added to \( T_0/2 \), the sum is compared with \( T_s \) and the output is given to

---

**Figure 3.13. Theta value (γ) in sector-1**
multiplexer, similarly the same procedure is continued for \( T_0/2 + T_a + T_b \) and \( T_a/2 + T_a + T_b + T_b/2 \). Finally the switching sequence of the selected triangle is obtained as shown in Figure 3.15 and 3.16.

**Figure 3.14. Timing computation- MATLAB implementation**

**Figure 3.15. Time division waveform with respect to switching state**
5. Finally, in the fifth block calculating twelve switching pulse of the inverter is done. Here the inputs are sector number, sub-triangle number, $T_a$, $T_b$ and $T_0$. This block holds the switching pattern of the four different sub-triangles which are mapped with the switching sequence as shown in Table 3.1 and 3.2. Based on the sector number and sub-triangle number, the calculated on times are drawn to appropriate switches in order to develop the correct PWM as shown in Figure 3.17.
Figure 3.17. Switching state selection MATLAB implementation; a. Switching state selection- $\Delta_{1,2}$, b. Switching state selection- $\Delta_{1,4}$, c. Switching state selection- $\Delta_{1,1}$, d. Switching state selection- $\Delta_{1,3}$
Table 3.1. Switching sequence of the sub-triangles for sector 1 to 3

<table>
<thead>
<tr>
<th>Si</th>
<th>$\Delta_{ij}$</th>
<th>Switching sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\Delta_{1,1}$</td>
<td>$[-1-1-1]$ $[0-1-1]$ $[00-1]$ $[00]$ $[010]$ $[011]$ $[111]$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$T_0/3$ $T_1/2$ $T_2/2$ $T_0/3$ $T_1/2$ $T_2/2$ $T_0/3$</td>
</tr>
<tr>
<td>2</td>
<td>$\Delta_{1,2}$</td>
<td>$[110]$ $[100]$ $[10-1]$ $[00-1]$ $[0-11]$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$T_0/2$ $T_1/2$ $T_2$ $T_0/2$ $T_1/2$</td>
</tr>
<tr>
<td>3</td>
<td>$\Delta_{1,3}$</td>
<td>$[0-1-1]$ $[-1-1-1]$ $[10-1]$ $[100]$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$T_0/2$ $T_1$ $T_2$ $T_0/2$</td>
</tr>
<tr>
<td>4</td>
<td>$\Delta_{1,4}$</td>
<td>$[00-1]$ $[10-1]$ $[11-1]$ $[110]$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$T_0/2$ $T_1$ $T_2$ $T_0/2$</td>
</tr>
<tr>
<td>1</td>
<td>$\Delta_{2,1}$</td>
<td>$[111]$ $[110]$ $[010]$ $[000]$ $[00-1]$ $[-10-1]$ $[-1-1-1]$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$T_0/3$ $T_1/2$ $T_2/2$ $T_0/3$ $T_1/2$ $T_2/2$ $T_0/3$</td>
</tr>
<tr>
<td>2</td>
<td>$\Delta_{2,2}$</td>
<td>$[-10-1]$ $[00-1]$ $[01-1]$ $[010]$ $[110]$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$T_0/2$ $T_1/2$ $T_2$ $T_0/2$ $T_1/2$</td>
</tr>
<tr>
<td>3</td>
<td>$\Delta_{2,3}$</td>
<td>$[110]$ $[11-1]$ $[01-1]$ $[00-1]$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$T_0/2$ $T_1$ $T_2$ $T_0/2$</td>
</tr>
<tr>
<td>4</td>
<td>$\Delta_{2,4}$</td>
<td>$[010]$ $[01-1]$ $[11-1]$ $[-10-1]$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$T_0/2$ $T_1$ $T_2$ $T_0/2$</td>
</tr>
<tr>
<td>1</td>
<td>$\Delta_{3,1}$</td>
<td>$[-1-1-1]$ $[-10-1]$ $[-100]$ $[000]$ $[010]$ $[011]$ $[111]$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$T_0/3$ $T_1/2$ $T_2$ $T_0/3$ $T_1/2$ $T_2/2$ $T_0/3$</td>
</tr>
<tr>
<td>2</td>
<td>$\Delta_{3,2}$</td>
<td>$[011]$ $[010]$ $[110]$ $[100]$ $[1-10]$ $[-10-1]$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$T_0/2$ $T_1/2$ $T_2$ $T_0/2$ $T_1/2$</td>
</tr>
<tr>
<td>3</td>
<td>$\Delta_{3,3}$</td>
<td>$[-10-1]$ $[-1-1-1]$ $[-11-1]$ $[010]$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$T_0/2$ $T_1$ $T_2$ $T_0/2$</td>
</tr>
<tr>
<td>4</td>
<td>$\Delta_{3,4}$</td>
<td>$[-100]$ $[-110]$ $[-111]$ $[011]$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$T_0/2$ $T_1$ $T_2$ $T_0/2$</td>
</tr>
</tbody>
</table>
Table 3.2. Switching sequence of the sub-triangles for sector 4 to 6

<table>
<thead>
<tr>
<th>$S_i$</th>
<th>$\Delta_{ij}$</th>
<th>Switching sequence</th>
</tr>
</thead>
</table>
|       | $\Delta_{4,1}$ | $\begin{array}{cccccc}
T_0/3 & T_1/2 & T_2/2 & T_0/3 & T_1/2 & T_2/2 & T_0/3 \\
\end{array}$ |
|       | $\Delta_{4,2}$ | $\begin{array}{cccccc}
T_0/2 & T_1/2 & T_2 & T_0/2 & T_1/2 \\
\end{array}$ |
| 4     | $\Delta_{4,3}$ | $\begin{array}{cccccc}
T_0/2 & T_1 & T_2 & T_0/2 \\
\end{array}$ |
|       | $\Delta_{4,4}$ | $\begin{array}{cccccc}
T_0/2 & T_1 & T_2 & T_0/2 \\
\end{array}$ |
|       | $\Delta_{5,1}$ | $\begin{array}{cccccccc}
T_0/3 & T_1/2 & T_2/2 & T_0/3 & T_1/2 & T_2/2 & T_0/3 \\
\end{array}$ |
|       | $\Delta_{5,2}$ | $\begin{array}{cccccccc}
T_0/2 & T_1/2 & T_2 & T_0/2 & T_1/2 \\
\end{array}$ |
| 5     | $\Delta_{5,3}$ | $\begin{array}{cccccc}
T_0/2 & T_1 & T_2 & T_0/2 \\
\end{array}$ |
|       | $\Delta_{5,4}$ | $\begin{array}{cccccc}
T_0/2 & T_1 & T_2 & T_0/2 \\
\end{array}$ |
|       | $\Delta_{6,1}$ | $\begin{array}{ccccccccc}
T_0/3 & T_1/2 & T_2/2 & T_0/3 & T_1/2 & T_2/2 & T_0/3 \\
\end{array}$ |
|       | $\Delta_{6,2}$ | $\begin{array}{cccccc}
T_0/2 & T_1/2 & T_2 & T_0/2 & T_1/2 \\
\end{array}$ |
| 6     | $\Delta_{6,3}$ | $\begin{array}{cccccc}
T_0/2 & T_1 & T_2 & T_0/2 \\
\end{array}$ |
|       | $\Delta_{6,4}$ | $\begin{array}{cccccc}
T_0/2 & T_1 & T_2 & T_0/2 \\
\end{array}$ |

3.4.2 Simulation Results

The performance of the proposed scheme is studied by an extensive simulation study by using MATLAB/Simulink 11.b in entire LM modulation range $(0 < M_a \leq 0.907)$ with 3kHz switching frequency. Initially, the simulation studies are
conducted for $M_a=0.5$. Here the line voltage ($V_{line}$) is measured as a 2-level output because only the SVs and ZVs have participated in the switching sequence. Hence, the $V_{line}$ resulted in 2-level output as 147.8V with THD value of 13.06%. It is found that the fundamental voltage magnitude is less because of the less modulation index. Next the same simulation study is extended for the higher modulation ranges (more than 0.5) and resulting in increased voltage magnitude as shown in Figure 3.18.

**Figure 3.18.** Simulation results; a. $V_{line}$ [200V/div] [2ms/div] at $M_a = 0.5$

b. $V_{line}$ [200V/div] [2ms/div] at $M_a = 0.907$
When the inverter is operated at maximum LM range of 0.907, the $V_{\text{line}}$ resulted is 268.4V with THD values of 15.66% as shown in Figure 3.19. The aforementioned simulation results confirm the theoretical values. Based on the Figure 3.20, as expected fundamental voltage is increasing leniently by increasing $M_a$. Similarly, dependency of the voltage THD on $M_a$ is pictured in Figure 3.21 and obvious variation is confirmed.
Next, the study was carried out for the IGBT switching loss calculation based on [176]. Figure 3.22 shows the simulation results of IGBT collector current ($i_C$) and voltage across collector to emitter ($V_{CE}$) for the $S_{C1}$. In this, the switching action was performed for the time period 0 to 0.15sec at $M_a$=0.5 and during 0.15 to 0.3sec at $M_a$=0.907. Based on the $i_C$ and $V_{CE}$ values, power losses ($P_s$) of the $S_{C1}$ was calculated for $M_a$=0.5 as 6.1W. Similarly, when $M_a$=0.907 power losses calculated as 11.4W. Figure 3.23 represents power losses $S_{C1}$ for different $M_a$. Based on the chart, it could be understood that the $P_s$ increases when inverter $M_a$ is increased. Based on the results it could be comprehended that in lesser $M_a$ values the switching losses is lesser.

**Figure 3.21. Line voltage percentage THD versus $M_a$**

**Figure 3.22. Simulation results of switching losses; $i_C$ [0.5Amp/div] [2ms/div] $V_{CE}$ [200V/div] [2ms/div], $i_g$ [1V/div] [2ms/div]**
3.5 DEVELOPED EXPERIMENTAL SETUP

This section discusses about the experimental setup of considered SVPWM fed NPC-MLI and its FPGA implementation.

The block diagram in Figure 3.24 shows the experimental setup.

Figure 3.24. Experimental setup of 3-level NPC-MLI drive with FPGA
The experimental setup comprises of following units,

- Three-phase 3-level NPC-MLI hardware prototype
- Interface and gate drive boards for logic operations, buffering, isolations etc.
- A SPARTAN-III-3A XC3SD1800A-FG676 DSP-FPGA board for running control programs, generating control signals, sampling feedback signals and communicating with the computer
- A 866MHz speed Pentium-III Xeon processor PC for Integrated System Environment (ISE) - Xilinx FPGAs
- An 2kW three-phase squirrel cage induction motor (SCIM) load
- Corroborating experimental results are captured using six channels YOKOGAWA spectrum analyzer

3.5.1 Prototype of a 2kW Three-Phase 3-Level NPC-MLI

A 2kW three-phase 3-level NPC-MLI drive connected with 1.5HP, 50Hz, 4-pole, 1200rpm, three-phase squirrel cage induction motor (SCIM) is built for the experimental purpose.

3.5.1.1 Switching device selection

Even though there are many intelligent power modules (IPM) developed in literature, the device selection is not clear when the power switch peak inverse voltage is high. In this range, vendors advocate to use the IGBTs. Choosing between them is a very application specific task in which cost, size, speed and thermal requirements should be considered.

While considering all the above issues, a small analysis is done between Mitsubishi IPM (CM1000DUC-34NF) and Semikron IPM (SK 100 MLI 066 T). The Semikron IPM has been the preferred device because it consists of four IGBT’s with in-built gate driver and it has added features of low duty cycle, low frequency
(<20kHz) and small line or load variations [176]. When both device types are tested in hard-switching applications, measurements show that the Semikron IPM exhibit lower losses. When the temperature is raised to reflect operating conditions, the Mitsubishi IPM rise more quickly than the Semikron IPM leading to high power loss. The power losses at 70 to 80°C temperature for Mitsubishi IPM leads to 27 to 28W, while for Semikron IPM, it is 17 to 8.1W. Hence, at room temperature 27°C the power loss is almost equal for the both Mitsubishi IPM and Semikron IPM. At 2kW, the advantage goes to the Semikron IGBT (SK 100 MLI 066 T) [177].

The above mentioned features of the directly mounted driver optimum Semikron IGBT allow for a compact, linear switching characteristics, low inductance and reduction on gate noises inverter design.

The DC-link connections can be short and very low inductive, resulting in reduced voltage overshoots. With SK 100 MLI 066 T modules, the entire inverter design can be simplified. Furthermore, the assembly processes involved in inverter production for the units are less complex (e.g. no manual or additional wave soldering). As a result, quality is boosted, while the overall system costs decrease.

Hence, in this research SK 100 MLI 066 T has been used for 2kW IPM NPC-MLI, which has an in-built structure of four IGBTs and two diodes used in each phase leg and helps to establish the basic structure of NPC-MLI as shown in Figure 3.25. Every module can operate till $V_{dc}=1000V$ and $I=20A$.

![Figure 3.25. SK100 MLI 066 T NPC-MLI for phase-A](image-url)
3.5.1.2 IPM protection

Since every power module needs an isolated single-phase AC supply, a three-phase transformer with six isolated windings is used to cater this need. The experimental setup has also been assimilated with the protection circuits. The over current protection is incorporated using CD4081 IC with help of LTS25NP Hall Effect current sensor which has a current capability of 25A [178]. IC4506 and IC4584 are used as opto-isolators which provides the isolation between the FPGA processor and the IGBTs [179]. A snubber capacitance of 1nF and resistance of 470Ω is used to protect the IGBTs from high voltage transients [177]. The diagram in Figure 3.26 to 3.28 shows the experimental setup.

Figure 3.26. Intelligent power module for Phase-A
Figure 3.27. Hardware setup for 3-level NPC-MLI drive system

Figure 3.28. Input DC supply with capacitors for NPC-MLI
3.5.2 The Peripheral Interface Circuit

To transfer the digital signals from the FPGA board to the multilevel inverter, peripheral interface circuits are needed [176], [180]. There are two main boards which are needed for the gate pulse interface, (i) interface board and (ii) gate driver board. They perform the following functions,

- Generating mid symmetric PWM signals to control the power switches
- Incorporating ‘dead time (t_d)’ in the PWM signals
- Providing isolation between control and power circuits
- Setting over current/voltage protections and reset/stop functions

(i) Interfacing Board: Figure 3.29 shows the interfacing board for the 3-level inverter. The 3-level inverter requires twelve PWM signals. They can be divided into three groups of four PWM signals. Therefore essentially twelve signals are required from the FPGA SPARTAN-III-3A XC3SD1800A-FG676 DSP-FPGA controller board. Proper dead time is then incorporated in these twelve PWM signals to avoid ‘mutual-ON’ of the upper and lower switches on the same phase of the inverter [176].

![DC supply for gate driver](image-url)
(ii) **Gate Driver Board:** The gate driver circuit provides isolation between the FPGA control and power circuits. Every IGBT of the power circuit has its own gate driver circuit consisting of isolated DC supplies, opto-couplers and gate drivers. Figure 3.30 shows a typical gate driver circuit [180], [181] to drive an IGBT. The opto-couplers ‘Avango-A4506’ transfer the PWM signal and isolate the control circuit from the power circuit.

![Gate driver circuit with boost strap capacitor](image)

**Figure 3.30. Gate driver circuit with boost strap capacitor**

This driver IR2110 has many important features such as floating channel designed for bootstrap operation, tolerance to negative transient voltage dv/dt, immune gate drive supply range from 10 to 20V, power ground, ±5V offset, CMOS schmitt-triggered inputs with pull-down cycle by cycle edge-triggered shutdown logic and matched propagation delay for both channels outputs in phase with inputs.

### 3.5.3 FPGA Board

FPGA offers the most preferred way of designing PWM generator tool for power converter applications [116], [121], [176]. The SPARTAN-III generation of FPGAs include the extended SPARTAN-3A family (SPARTAN-3A, SPARTAN-3AN and SPARTAN-3A DSP platforms), along with the earlier SPARTAN-3 and
SPARTAN-3E families. These families of FPGAs are specifically designed to meet the needs of high volume, cost-sensitive electronic applications such as consumer products. The SPARTAN-3 generation includes 25 devices offering densities ranging from 50,000 to 5 million system gates. Xilinx SPARTAN-III-3A XC3SD1800A-FG676 DSP-FPGA board is chosen for the processor in SPARTAN-3AN family [182] as shown in Figure 3.31.

Figure 3.31. Schematic view of SPARTAN-III-3A XC3SD1800A - FG676 DSP-FPGA

It is specifically designed for the development of high speed multivariable digital controllers and real-time simulations in various fields. It is a complete real-time control system based on a 603 Power PC floating point processor running at 250MHz. With the aim of evaluating the recommended simplified SVPWM algorithm an experimental setup has been built and a validation test procedure is performed through SPARTAN-III-3A XC3SD1800A-FG676 DSP-FPGA based controller which includes the following components and features [120]:
- 8 numbers of digital input using DIP switches
- 16 numbers of digital outputs using discrete LEDs
- One Reset switch
- FPGA configuration through
  # Joint Test Action Group (JTAG) port
  # On board Flash Prom XCF16PV048
- On board programmable oscillator from 3MHz to 100MHZ
- 16 × 2 LCD interface
- ADC and DAC interface
- Add on card VSDA-03 for ADC and DAC with SDA bus
- RS232 Serial port
- PWM
- USB 2.0 Compliance interface (480MBit/sec)

One of the main contributions of this research work is to develop an easy VHDL code for 3-level NPC-MLI. Implementation is done using MATLAB system generator which allows minimizing the time spent by the designer for the description and simulation of the circuit. In addition, the design is flexible; it is possible to change the design parameters and check quickly the effect on the performance and the architecture of the system. The Xilinx ISE project navigator tool and the system generator has its own Simulink models which are in three tabs called the Xilinx block-set which helps to interface MATLAB Simulink file and Xilinx board through 36 pin Flat Ribbon Cable (FRC) [183].

3.5.4 Power Analyzer

The 6 channel WT1800-Yokogawa power analyser is used to measure the inverter parameters. It is having a vertical resolution in power measurement scope. The WT1800-Yokogawa power analyser is capable of 16-bit high resolution
and approximately 2MHz sampling to make it possible to measure faster signals with higher precision. The WT1800 is capable of performing harmonics measurement for the input and output signals simultaneously and also it performs concurrent data analysis for the input and output.

3.6 MATLAB SYSTEM GENERATOR BASED FPGA IMPLEMENTATION PROCEDURE

This section discusses about how to use MATLAB/Simulink and Xilinx System Generator as implementation tools for development of FPGA implementation through MATLAB system generator tool and Xilinx ISE project navigator. This allows the minimization of the time spent for the design and cost of implementation. In addition, the design is flexible; it is possible to change the design parameters and check quickly the effect on the performance and the architecture of the system. The implementation will start from the model designing in Simulink and cover all the steps required for VHDL implementation in FPGA platform.

3.6.1 VHDL Code Generation using MATLAB/Simulink Xilinx Environment

This section discuss about MATLAB/Simulink code conversion to VHDL code using MATLAB system generator.

3.6.1.1 MATLAB/Simulink

MATLAB software package provides a powerful high level modelling environment for people who are involved in system modelling and simulations [123]. The MATLAB/Simulink is a widely used simulating tool with a graphical overview that can be easily integrated with DSP, dSPACE and FPGAs. The graphical nature of the tool greatly reduces the time for designing and analysing of large dynamic systems. The physical system and observer models are first designed and tested in the Simulink environment using the default block set from the Simulink library.
3.6.1.2 Xilinx system generator

Xilinx offers 93 highly optimized library of blocks called the Xilinx block-set. This can be simulated within Simulink and then compiled for FPGA implementation [116], [120]. This tool is called Xilinx System Generator (XSG). It would be required to transfer the design into the Simulink environment and replace the Simulink simulation only blocks with System Generator blocks. This design flow offers higher performance than Hardware Descriptive Language (HDL) coder as each block is pre-optimized Intellectual Property (IP) for Xilinx FPGAs.

XSG developed for MATLAB/Simulink package is widely used for algorithm development and verification purposes in DSP and FPGAs. XSG allows an abstraction level algorithm development while keeping the traditional Simulink block sets, but at the same time automatically translating designs into hardware implementations that are faithful, synthesizable and efficient. Basically, the system generator allows minimizing the time spent by the designer for the description and simulation of the circuit. On the other hand, the design is flexible. It is possible to change the design parameters and check quickly the effect on the performance and the architecture of the system.

In a typical design flow supported by System Generator, the following steps are followed:

- Describe the algorithm in mathematical terms,
- Realize the algorithm in the design environment, initially using double precision,
- Trim double precision arithmetic down to fixed point and
- Translate the design into efficient hardware (DSP /FPGAs).
3.6.1.3 VHDL Code Generation through XSG

To implement the considered PWM algorithm, a SPARTAN-III-3A XC3SD1800A-FG676 DSP-FPGA processor is used as a PWM generator to build the appropriate gating signals to the inverter switches. One of the main contributions of this research when compared to previous reported work is to develop the VHDL code for the FPGA implementation using MATLAB-Xilinx System Generator. The system generator has its own Simulink models that are in three tabs called the Xilinx block-set. System Generator is a FPGA design tool from Xilinx that enables the use of the Math works model-based design environment Simulink for FPGA design. When installing MATLAB/Simulink, XSG is installed in PC/Laptop and then Xilinx block sets are included in Simulink library. Each block is configured after opening its dialog window which permits fast and flexible designs.

![Figure 3.32. Block Diagram of VHDL code generation](image)

The functional simulation is possible even before the compilation of the model designed. The compilation generates the files of the structural description of the system in a standard hardware description language (VHDL/Verilog) for the Integrated System Environment (ISE) suitable for Xilinx FPGAs.
Figure 3.33. Design flow of Integrated System Environment

The System Generator works within the Simulink model-based design methodology. Often an executable spec is created using the standard Simulink block-sets. This spec can be designed using floating point numerical precision and without hardware detail. Once the functionality and basic dataflow issues have been defined, System Generator can be used to specify the hardware implementation details for the Xilinx devices. System Generator uses the Xilinx block-set for Simulink and will automatically invoke Xilinx Core Generator to generate highly optimized net lists for the FPGA building blocks. System Generator can execute all the downstream implementation tools to produce a bit stream for programming the FPGA. An optional test bench can be created using test vectors extracted from the Simulink environment to use with ModelSim or the Xilinx ISE Simulator.
Flow chart (Figure 3.33) shows the complete VHDL design flow and FPGA implementation for the SVPWM algorithm using Xilinx ISE project navigator.

The XSG based implementation is split into 3 steps:

i. XSG off-line simulation

ii. Synthesis, Register Transfer Level (RTL) and VHDL generation

iii. Design implementation

The following steps give the clear idea about the implementation:

- First, the developed MATLAB/Simulink “.mdl” file is simulated and verified through MATLAB/Simulink.

- After installing Xilinx ISE project navigator, XSG is an add-on to the Simulink. It adds Xilinx block-set to the current Simulink library. After the models are analysed through Simulink, an equivalent Xilinx model is implemented using the Xilinx block-sets (Gateway In, Gateway Out system etc.)

- Next all the performance parameters are set including the System Generator Token containing the FPGA clock period and Simulink system period Xilinx model is then run once with all the necessary inputs connected to the appropriate Simulink models. This provides the System Generator with all the necessary parameters required for the implementation. For hardware implementation files, double click the System Generator Token, select VHDL net list as the compilation tool.

- Then, the system generated “.mdl” file is compiled in the Xilinx ISE project navigator platform and its generating VHDL net list, RTL bit stream for programming etc. This system generator simulations are bit and cycle accurate. The generated VHDL can be simulated using off line simulated tool called Model sim 5.8e.

- Then, the developed VHDL code is synthesized in Xilinx ISE project navigator simulator. It allows for the RTL synthesis, place and route to be performed using a tool such as project navigator.
After the above steps a net list folder is created in the parent directory. This folder contains a file with the same name as the Xilinx model in the Simulink .mdl file with an .ise extension. Processes contain Synthesis-XST, implement design and generate programming file. Right clicking Synthesis-XST and then by selecting the run option turns the circuit behaviour RTL into an implementable design in terms of logic. After successful synthesis, schematics can be viewed. After synthesising, it can generate device utilization and power utilization report. Then the developed model is entered into implementation process which consists of Translation, Mapping, Place & Routing, converting the implementable design into a file that can be downloaded into the in target FPGA processor.

- Generate programming file produces a bit stream which can be used to configure the Xilinx device. To configure the device, run configure device (iMPACT). Select configure device using Boundary Scan in the new window opened and click Finish. Select the generated bit file which will be used to configure the FPGA. After successful configuration of the device a message, “Program succeeded” will appear. Also on the board, the LED will change from INT to DONE.

- Pin assignment: Before translating the download process, User Constrained File (UCF) is written to assign pin configuration of the FPGA board to the PWM generator I/O’s. Once, the translate process merges the UCF and net list files, the mapping process is enabled. The mapping helps to fit the design into the available resource of the target FPGA processor and finally placing the code into the FPGA.

- Then, the ISE Project Navigator runs to analyze the design using Chip Scope. This will open a new window for the Chip Scope Pro. Finally through JTAG, chain device would be detected automatically.
3.7 IMPLEMENTATION OF THE SVPWM SCHEME IN FPGA

This section discusses about the SVPWM design and its FPGA implementation schemes through MATLAB system generator tool.

3.7.1 Overview of the SVPWM core

To design the proposed PWM core SPARTAN-III-3A XC3SD1800A-FG676 DSP-FPGA processor has been chosen. Block diagram in Figure 3.34 gives an overview of the proposed SVPWM design for the FPGA functional building block. The designed SVPWM core consists of two basic units namely processing unit and switching vector mapping unit. Processing unit consists of a base scheme or algorithm and a counter. The calculation flow is same for any reference vector that stays in any one of the 24 sub-triangles; however for the voltage vectors and the corresponding switching states, switching sequences are different in each sub-triangle. The switching vector mapping unit is holding the switching vector sequence for the inverter operation.

![Figure 3.34. Processing and switching mapping unit structure of SVPWM](image)

(a) Processing unit:

The processing unit is basically a part of the implementation and it does four main tasks:
\begin{itemize}
  \item determination of the sector number \( S_i \),
  \item determination of vector versus co-ordinates \( (V_{s0}, V_{s0}) \),
  \item identification of sub-triangle and
  \item On-time \( T_0 \), \( T_a \) and \( T_b \) calculation of the switches.
\end{itemize}

(b) \textbf{Switching vector mapping unit:}

The vector mapping unit uses memory. It fires the pre-stored switching sequence for the three-phase inverter based on sector \( S_i \), sub-triangle for the on-times obtained from the processing unit.

The functional block diagram of the recommended MLI-SVPWM is shown in Figure 3.35. The top-level design for the 3-level SVPWM IP core is constructed based on these functional blocks and then compiled, optimized and synthesis with the Xilinx platform.

In this implementation, a vertex of any triangle can have multiple redundancy switching states (two or more possible switching states). For a sub-triangle, a switching sequence is formed using a combination of the most suitable switching states from all possible switching states at the vertices. The resulting switching sequence is mapped with respect to the triangle and sector number. The switching sequence is then fired for the on-times obtained from the processing unit. Thus the suggested algorithm is able to make use of any redundancies for any vertex of the triangle. While designing the IP core, many other factors are considered such as computation accuracy, simplicity and flexibility. Based on the simplicity and flexibility the planned core involves several computational blocks such as sector detector, on-time calculator, switching state selector and so on. Hence the design gives computation efficiency and easy data manipulation. The core also considers some key design measures for improving computation accuracy and simplifying hardware design and fixed point arithmetic unit is adopted for implementing the calculations.
The architecture of proposed SVPWM for MLI is shown in the Figure 3.35, which consists of mainly decoder and the interfacing block, sector detecting module, triangle determination unit, modulation index, trajectory identifier, on-time calculator block, switching state selector unit, SVPWM generator building block and its registers. The IP core is designed to operate at 20MHz clock frequency and high switching frequency as well as the $t_d$ is adjustable.

![Figure 3.35. Proposed digital implementation structure of SVPWM](image)

The architecture and its functional blocks of the designed SVPWM core are presented in Figure 3.35. The architecture of proposed SVPWM sub-blocks is explained as follows:

- 3/2 axis converter block: It performs the abc to d-q conversion, which generates the $V_{ref}$ and angle ($\theta$).
• Modulation index block: It generates the required $M_a$ for the inverter operation. Depending upon the $V_{ref}$ requirement, the $M_a$ value of the inverter can be given through the $M_a$ block.

• Switching period block: It holds the switching frequency for the inverter switches.

• Sector identification block: This block finds the $V_{ref}$ location based on the angle ($\theta$) and magnitude of the $V_{ref}$.

• Triangle identifier block: The block computes $V_{ref}$, sub-triangle location based on the flow chart.

• Trajectory identifier block: This block measures the $V_{ref}$ location based on the $M_a$ values. The identifier module provides respective on-time to the switching state selector unit, which stores the switching states and switching sequences. The trajectory identifier module also provides the crossover angle and holding angle values which must be needed when the inverter operates in the over modulation.

• On-time calculation block: The on-time calculation block provides the respective on-times to the switching state selector unit based on 2-level SVM schemes. This unit uses the LUT's to store the switching states and the switching sequences. Lastly SVPWM generator unit generates the pulses to the 3-level NPC-MLI after inserting the $t_d$ whose content is part of $t_d$ register. Figure 3.36 shows the switching sequence for $\Delta_{1,1}$ pattern for the LUT.

<table>
<thead>
<tr>
<th>[-1-1-1]</th>
<th>[0-1-1]</th>
<th>[00-1]</th>
<th>[000]</th>
<th>[010]</th>
<th>[011]</th>
<th>[111]</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_0/3$</td>
<td>$T_a/2$</td>
<td>$T_b/2$</td>
<td>$T_0/3$</td>
<td>$T_a/2$</td>
<td>$T_b/2$</td>
<td>$T_0/3$</td>
</tr>
</tbody>
</table>

**Figure 3.36. Switching sequence for $\Delta_{1,1}$**
• Switching state unit: It holds the specified switching event including the redundant switching states.

• Dead time register block: \( t_d \) is an important factor to be considered especially in the real-time implementation which will protect the phase leg of the inverter from short circuit with the DC-link and it is set as 5\( \mu \)s in the register block which is 2.72 times greater than the recommended \( t_d \) given in Semikron IGBT (SK100 MLI 066 T) datasheet [177], [184].

• SVPWM Generating Unit: This block produces the pulses to the NPC-MLI after inserting the \( t_d \).

To simplify the interface with the processor, commands to these registers are routed through a decoder and interface circuit. The clock is acting as a base time for PWM generator and operated at 100MHz. The overflow flag from PWM generator unit indicates the value of PWM counter has reached the maximum count, which can be used to trigger events for the inverter.

### 3.7.2 Overview of the Experimental Platform

To implement the proposed SVPWM algorithm, a SPARTAN-III-3A XC3SD1800A-FG676 DSP-FPGA processor is used as a PWM generator to build the appropriate gating signals to the inverter switches as shown in Figure 3.37. JTAG serial mode configuration interface card is used to download the developed code to the FPGA processor and configure PROM through Master Serial. P7 connector is used as a JTAG downloading connector. The 36 pin FRC (Flat Ribbon Cable) bus is used for connecting pulses between FPGA boards to inverter drives.

![Figure 3.37. Overview of the experimental platform](image)
3.7.3 Implementation of the Proposed SVPWM Scheme in SPARTAN–III-3A FPGA

The main purpose of this section is to design the 3-level SVPWM core design using MATLAB System generator Xilinx Environment.

The implementation is basically divided into five stages as follows:

i. MATLAB code generation through XSG,

ii. VHDL code generation and its simulation,

iii. RTL file and bit file generation,

iv. Synthesis and

v. Download the program into target FPGA.

3.7.3.1 MATLAB code generation through XSG

The MATLAB/Simulink simulation model is designed based on the SVPWM architecture as seen in section 3.7.1 and then simulated and verified. After that the main MATLAB/Simulink model is modified through Xilinx System Generator Environment Simulink tool box block-set. Here each and every module input and output are connected through XSG ‘in’ and ‘out’ port block. The design offered SVPWM XSG MATLAB “.mdl” code for MLI as shown in the Figure 3.38.
Figure 3.38. Overall 3-level SVPWM in Simulink-Xilinx system generator environment

Its design architecture is totally built based on the design description of section 3.7.1. Figure 3.39 shows the angle and triangle calculation structure. Figure 3.40 and 3.41 shows timing calculation of the $T_a$, $T_b$, $T_0$ and switching pulse generation structure for the proposed implementation. It could be seen that all the subsystems numeral finder blocks are replaced with the system generator tool box block for the target implementation.
Figure 3.39. Angle and triangle calculation structure of 3-level SVPWM

Figure 3.40. $T_a$, $T_b$, $T_0$ generation structure of considered 3-level SVPWM
Once the MATLAB file has converted system generator block-set as discussed in previous section 3.6.1.3, off line simulation is possible through MATLAB before going for the VHDL.

![Diagram](image1)

**Figure 3.41. Pulse generation structure of considered 3-level SVPWM**

**Figure 3.42. Switching pulses for $S_{1A}$-S$_{4A}$**

Figure 3.42 shows the off line simulation of the inverter pulses for the leg-A. It ensures the system level design operation before generating the VHDL code.
3.7.3.2 VHDL code generation and its simulation

Once system generated MATLAB code is developed, the system generated “.mdl” file is compiled using Xilinx ISE project navigator platform. Then it generates VHDL net list and RTL bit stream. The developed VHDL code is in mixed level coding structure. The developed VHDL code of considered PWM is simulated (off-line) using ModelSim 5.8e. Figure 3.43 shows the off-line simulation of the inverter pulses. It ensures that the pulse pattern, td values and the whole system before synthesis process. From Figure 3.44, it is understood that the given td of 5µsec is fixed on the final target pulse of the inverter, which ensures the protection from the short circuit.

![VHDL code generation and its simulation](image)

Figure 3.43. SVPWM pulses (S1A-S4c) with f_s = 20kHz, f_o = 50Hz (through ModelSim 5.8e)

![Zoomed view SVPWM pulses](image)

Figure 3.44. Zoomed view SVPWM pulses (S1A-S4c) with dead time, f_s = 20kHz, f_o = 50Hz (through ModelSim 5.8e)
3.7.3.3 **RTL file and bit file generation**

As a next step of the created process of the designed PWM architecture, the RTL view is generated. The Figure 3.45 and 3.46 shows the full architecture RTL view and theta and sector calculation block RTL view of the planned SVPWM IC. This process is used for optimizing the design flow for the targeted design. Based on this, it could be understood that the designed PWM IC structure depends on the section 3.7.1 architecture, which forecast the device and power utilization of the offered PWM IC.

![Figure 3.45. SVPWM RTL file view](image-url)
Next, the developed code is blend in Xilinx ISE project navigator. It allows the RTL synthesis, place and route to be performed using a tool such as Project Navigator. After the above steps, a net list folder is created in the parent directory. This folder contains a file with the same name as the Xilinx model in the Simulink .mdl file with an .ise extension. Processes contain synthesis-XST, implement design and generate programming file [182]. Right clicking synthesis-XST and selecting the run option turns the circuit behaviour RTL into an implementable design in terms of logic. After successful synthesis, the device utilization and power utilization report is generated. It provides how many logical blocks, LUTs and FFs are used in architecture. The RTL view of the offered SVPWM IC core is shown in Figure 3.50 which also includes many internal structure based on the described design. It is synthesis using Xilinx ISE tool which is the part of the Xilinx ISE software before downloading the program into the target FPGA kit through JTAG connector cable.

Then the developed model will enter into implementation process which consists of Translation, Mapping, Place and Route converting the implementable design into a file that can be downloaded into the in target FPGA processor.
3.7.3.5 Download the program into target FPGA

Once the implementation design has been done the resulting net list is saved to an NGC file. Next the JTAG, the serial mode ("IEEE Standard 1149.1") configuration interface card is used to download the developed code to the Target SPARTAN-III-3A XC3SD1800A-FG676 DSP-FPGA. The JTAG configuration is through the independent boundary scan selection. Then the regenerated bit file is generated. Finally, the developed RTL is converted to bit stream format and then the UCF is written for pin assignment for the mapping process. Mapping is done to fit the design into the available resource of the target FPGA processor. Finally placing the code in the target SPARTAN-III-3A XC3SD1800A-FG676 DSP-FPGA is done.

3.7.4 Implementation Results and their Reports

In this section, software and hardware implemented results and their device and power utilization report for the suggested MLI-SVPWM are detailed.

3.7.4.1 Hardware simulation

Figure 3.47 shows the VHDL simulation results of the generated PWM pulse of MLI using ModelSim 5.8e. From Figure 3.44 it is understood that the given dead time of 5 µsec is fixed on the final target pulse of the inverter which ensures the protection from the short circuit.

There are three types of floor views that have been generated for the SVPWM IC which are overall floor view of device utilization, input port assign view and output port assign view. Figure 3.48.a and Figure 3.48.b show the designed VLSI architecture for SVPWM controller. From the Figure 3.49, it is observed that the proposed code occupies very less resource/area. Figure 3.49 shows the input and output port of proposed implementation.
Figure 3.47. SVPWM pulses (S1A-S4C) with $f_s=20$ kHz, $f_o=50$ Hz (through Xilinx)

The proposed PWM design I/O’s are mapped properly using UCF on the basis of reduction of the power losses. Input’s and output’s floor plan and power utilization report are shown in Figure 3.56.

Figure 3.48 Design view of proposed SVPWM in target SPARTAN-III-3A XC3SD1800A-FG676 DSP-FPGA; a. Device/Area utilization RTL view of FPGA implementation, b. Zoomed RTL view of proposed SVPWM implementation.
Figure 3.49  Input and output port view of proposed design for SVPWM in target SPARTAN-III- 3A XC3SD1800A-FG676 DSP-FPGA; a. Input port view, b. Output port view

3.7.4.2  Power and device utilization report

Figure 3.50 shows the power utilization of the proposed PWM design. From the power utilization report it is understood that the device design consume 0.13W for one cyclic operation of pulse generation.

Figure 3.50. Power utilization report of considered SVPWM
The proposed SVPWM implementation work also considered some key design measures for improving computational accuracy, simplify the hardware design and implementation of the calculation (using fixed point arithmetic unit).

<table>
<thead>
<tr>
<th>Logic Utilization</th>
<th>Used</th>
<th>Available</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slice Flip Flops</td>
<td>424</td>
<td>33,280</td>
<td>1%</td>
</tr>
<tr>
<td>Number of 4 input LUTs</td>
<td>1,112</td>
<td>33,280</td>
<td>3%</td>
</tr>
<tr>
<td>Number of occupied Slices</td>
<td>895</td>
<td>15,640</td>
<td>4%</td>
</tr>
<tr>
<td>Number of Slices containing only related logic</td>
<td>895</td>
<td>995</td>
<td>100%</td>
</tr>
<tr>
<td>Number of Slices containing unrelated logic</td>
<td>0</td>
<td>995</td>
<td>0%</td>
</tr>
<tr>
<td>Total Number of 4 input LUTs</td>
<td>1,200</td>
<td>33,280</td>
<td>5%</td>
</tr>
<tr>
<td>Number used as logic</td>
<td>1,064</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number used as a route-thru</td>
<td>496</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number used as shift registers</td>
<td>24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of bonded I/Os</td>
<td>21</td>
<td>519</td>
<td>4%</td>
</tr>
<tr>
<td>I/O Flip Flops</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of BUFGMUXs</td>
<td>2</td>
<td>24</td>
<td>8%</td>
</tr>
<tr>
<td>Number of DSP48As</td>
<td>2</td>
<td>84</td>
<td>2%</td>
</tr>
<tr>
<td>Number of RAME16BWERs</td>
<td>2</td>
<td>84</td>
<td>2%</td>
</tr>
<tr>
<td>Average Fanout of Non-Clock Nets</td>
<td>2.36</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 3.51. Device utilization report of developed architecture SVPWM**

Figure 3.51 depicts the hardware resource utilisation of the proposed algorithm. From the report the resources taken by this module are very few. The overall device utilization of the PWM design is 6%. Hence it is very suitable to be considered as an IP core which can be integrated into a System On Chip (SOC) with other IP cores [171]. The SOC can greatly reduce the area of a PCB and improve the immunity to interferences for the power converters design.

3.7.5 Processing Time Calculation

The processing time calculation of the offered SVPWM scheme is explained as follows by using the following equations [185]

\[ \text{(3.21)} \]
Where $S_{\text{config}}$ in this case is determined by $S_{\text{config}} = N_{\text{frames}} \times L_{\text{length}}$. The chosen FPGA processor has 767 frames and each frame consists of 2208 bits, 16640 slices, in an array of 32 rows by 28 columns. On an average, 1 CLB column = 16640/28 = 594.285 slices. The 8 bit design takes 895 slices for SVPWM. Therefore, a minimum of (995/594.285 =1.5060) two columns are required. The required buffer space needed is calculated as follows: For partial reconfiguration, $S_{\text{config-partial}} = 1.6743*2208 = 3325.2732$ bits. For full reconfiguration $S_{\text{config-full}} = 28*2208 = 61824$ bits. The partial configuration time for one column is $(3325.2732 / 20 \text{ MHz}) = 0.1662\text{ms}$. Therefore, the partial configuration time for two columns is $T_{\text{part-config}} = 2*0.1662 = 0.3325\text{ms}$. The full reconfiguration time for one column is $(61824/ 20 \text{ MHz}) = 3.0912\text{ms}$ and the full reconfiguration time is $(28*61824)/20 \text{ MHz} = 86.553\text{ms}$. From the analysis, it is noted for a 20MHz clock, there is a significant decrease in the time taken for partial reconfiguration (0.3325ms i.e., 0.3841% of full reconfiguration time) when compared to full reconfiguration time (86.553ms).

Clock frequency = $(20\text{MHz} \times 0.3841)/100 = 76820\text{Hz}$

Processing time = $1/ \text{Clock frequency} = 1/76820 = 13.017\mu\text{sec}$, which is found to be less compared with the previous experimental implementations [126].

The overflow flag from PWM generator unit indicates that the value of PWM counter has reached the limit, which can be used to trigger events for the inverter. Unlike regular implementation, the proposed work has also considered some key design measures for improving computation accuracy, hardware design, and fixed point arithmetic unit for implementing the calculations.

From Table 3.3, the resources taken by this implementation are very few compared to the previous reported schemes [118], [186], [176]. Hence, it is very
suitable to be considered as an IP core which can be integrated into a SOC with other IP cores.

Table 3.3. Comparison of device utilization of existing and proposed schemes

<table>
<thead>
<tr>
<th>Device Utilization</th>
<th>Author name &amp; Year</th>
<th>FPGA Platform name</th>
<th>No. of slice FFs</th>
<th>No. of 4 input LUTs</th>
<th>No. of occupied slices</th>
<th>No. of bonded IOBs</th>
<th>Total device utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y. Guijie, (2008)</td>
<td>Xilinx SPARTAN-3 XCS400 PQ208</td>
<td>743/7,618</td>
<td>424/33,280</td>
<td>734/7168</td>
<td>37/141</td>
<td>10.23 %</td>
<td></td>
</tr>
<tr>
<td>Proposed implementation</td>
<td>Xilinx SPARTAN-3AN-XC3S400A</td>
<td>1365/7,168</td>
<td>1,112/33,280</td>
<td>9895/16,640</td>
<td>21/519</td>
<td>6.05 %</td>
<td></td>
</tr>
</tbody>
</table>

The considered algorithm does not require any additional calculations, which will reduce the memory space in the real-time implementation. In addition, the algorithm minimizes the processing time and also the hardware resource utilization.

3.8 EXPERIMENTAL RESULTS AND THEIR ANALYSIS

The performance of the considered SVPWM switching scheme has been modeled and an extensive simulation study is carried out by using MATLAB/Simulink with the system parameters shown in simulation study and the corroboration is done using a SPARTAN-III-3A XC3SD1800A-FG676 DSP-FPGA.
processor as explained in section 3.7. Figure 3.27 shows the experimental results for a laboratory prototype of a 2kW 3-level NPC-MLI feeding a 1.5HP induction motor at \( V_{\text{dc-link}}=300\text{V} \), fundamental frequency \( f_0=50\text{Hz} \) and switching frequency \( f_s=3\text{kHz} \). The hardware study is taken for different modulation indices from \( M_a=0.2 \) to 0.907. At \( M_a=0.5 \), the \( V_{\text{line}} \) is measured as a 2-level output because only the SVs and ZVs are participating in the switching sequence. The switching pulses are captured using four channel Agilent technologies DSO. Figure 3.52 demonstrates the experimental switching pulses for \( S_{1A}-S_{4A} \). Figure 3.53 and 3.54 show the line voltage inclusive of harmonic spectra, three-phase line voltage and phase current for \( M_a=0.907 \). It could be seen that \( V_{\text{line}} \) is resulted as 265.48V with THD value of 16.279%. From the above results, in LM zone the inverter is operated at maximum \( M_a \) of 0.907, which produced close to 90% of the fundamental voltage magnitude with respect to the input DC-link voltage.

![Agilent Technologies](image)

**Figure 3.52.** Experimental results: SVPWM switching pulses (\( S_{1A}-S_{4A} \)) for leg-A NPC-MLI
Figure 3.53. Experimental results; $V_{\text{line}}$ [150V/div] [1ms/div] & its THD spectrum at $M_a = 0.907$

Figure 3.54. Experimental results; Three-phase $V_{\text{line}}$ [150V/div] [1ms/div], $I_{\text{line}}$ [3A/div] [1ms/div] at $M_a = 0.907$
Table 3.4 shows the simulation and experimental results of $V_{\text{line}}$ and its $V_{\text{THD}}$ for different $M_a$. From the table it could be seen that there is a small variation between simulation and hardware results.

<table>
<thead>
<tr>
<th>$M_a$</th>
<th>Simulation results</th>
<th>Experimental results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$V_{\text{line}}$(volts)</td>
<td>THD in %</td>
</tr>
<tr>
<td>0.2</td>
<td>59.1</td>
<td>18.67</td>
</tr>
<tr>
<td>0.4</td>
<td>117.3</td>
<td>14.89</td>
</tr>
<tr>
<td>0.6</td>
<td>177.3</td>
<td>14.07</td>
</tr>
<tr>
<td>0.8</td>
<td>236.2</td>
<td>14.71</td>
</tr>
<tr>
<td>0.907</td>
<td>268.4</td>
<td>15.66</td>
</tr>
</tbody>
</table>

The comprehensive performance analysis of the simplified SVPWM algorithm is performed in LM zone as shape of the chapter is limited to LM whereas the output of higher values (0.907 to 1.0) belongs to OVM which is elaborated in next chapter.

### 3.9 SUMMARY

In this chapter the theoretical analysis, design habitat, FPGA implementation and experimental verification of the simplified 3-level MLI-SVPWM based on standard 2-level SVPWM has been successfully exhibited. The simulation and experimental results are presented in order to confirm the performance of the recommended algorithm at different modulation depths of the inverter.

The following are the salient features of the proposed scheme;

- The on-time calculation is simple due to the use of 2-level SVPWM logic. The on-time calculation equations do not change with the position of reference vector like the traditional approaches. Hence
there is no need for any additional LUTs unlike previous experimental implementations. [109], [117], [124].

- The considered SVPWM minimizes the device and power utilization for the FPGA implementation. It reduces the processing time to 13.017µsec, which is found to be less compared with the previous experimental implementation [126].

- In the SVD of an n-level inverter, the triangle where the reference vector is located is identified using a simple algebraic expression. This is leading to simplicity and flexibility of optimizing the switching sequence for the n-level implementation.

- The considered scheme can be used for any n-level (n≥3) inverter without any significant increase in computations.

- The proposed technique can be easily implemented using a commercially available DSP and FPGA processors, which normally supports only 2-level modulation.

- The scheme can be used for both neutral point clamped and cascaded H-bridge MLI topologies.