NTNU – Trondheim
Norwegian University of
Science and Technology

Synchronous Optimal Modulation for Medium voltage Multi-phase Machines

Implementation of three-level modulators in
Field Programmable Gate Arrays

Mamta Maharjan

Master of Science in Electric Power Engineering
Submission date: June 2013
Supervisor: Tore Marvin Undeland, ELKRAFT

Norwegian University of Science and Technology
Department of Electric Power Engineering

Problem Description

Medium voltage ac drives based on voltage source inverters are in increasing demand for numerous industrial applications. The advancement in power electronics devices like IGBTs has allowed to rise the low voltage ac drive to the medium voltage drive system. Medium Voltage IGBTs must have low switching frequency. Operation at reduced switching frequency requires synchronous modulation techniques because the conventional asynchronous modulation will generate too much harmonics.

Programmed modulation is synchronous modulation which pre-calculates optimal switching angles within a period of the fundamental period. If those switching angles are calculated by minimizing total harmonic contents then it is called Synchronous Optimal Modulation. These switching angles are not formed by carrier wave like in conventional way. These optimized PWM strategies are extremely difficult to realize with conventional analog circuitry. Therefore, the firmware implementation of synchronous optimal modulation in digital circuitry will be created by an extremely fast computation microchip called Field-Programmable Gate Array (FPGA).

In this Master thesis a modulator for a 3-level converter should be developed. The modulator should be possible to be extended to 9-phase machines .The modulator should give the interrupt to the processor routine where the calculation of relative switching instance is executed. From the software point of view the input to the modulator is memory mapped as an array with switching instance stored increased order due to time. The tasks to be performed are:

- Work out a specification together with the master student making the control software for the modulator.
- Decide which method to be used for dc-bus balancing for the 3-level converter
- Implement the modulator in a FPGA
- Test the modulator in the Lab by help of a 3-level 3-phase inverter operating an Induction Machine

Supervisor: Prof. Tore M. Undeland

Co-supervisor: Roy Nilsen, Wärtsilä Norway AS

ABSTRACT

The advancement in power electronics devices like Insulated Gate Bipolar Transistors (IGBT) has increased the low voltage ac drive to the medium voltage drive system. These IGBTs have higher voltage capability. For eliminating the harmonic content three level converters are chosen rather than two level converters. But these inverter switches using IGBTs should be operated in low switching frequency so that the huge loss associated with switching loss is reduced to large extent. When switching frequency is low then conventional asynchronous modulation technique is not appropriate because of the formation of sub-harmonic components. Therefore, it is wise to apply synchronous modulation technique. Programmed Modulation is one of best synchronous modulation technique which pre-calculates the optimum switching angles. If those switching angles are calculated by minimizing total harmonic contents then it is called Synchronous Optimal Modulation.

These optimized PWM strategies are extremely difficult to realize with conventional analog circuitry, but they can be effectively implemented in field-programmable gate array (FPGA) which has extremely fast computation capability and allows a few microseconds real-time computation of complex control algorithms. FPGA is a microchip which consists of matrix of configurable logic blocks (CLB) made up of flip flops and lookup tables. VHDL is Hardware Descriptive Language which is utilized to synthesize hardware designs in FPGA. VHDL creates digital circuitry, which performs operations in parallel so FPGA is extremely fast and performs various time critical tasks in the system.

The FPGA which is used in this project is Vertex-5 ppc440 FX30T. The digital electronics for implementing Synchronous Optimal Modulation are written in Xilinx Embedded System. The extremely fast computation power of FPGA finds its application in multiphase machine. Although this thesis deals with three phase Induction motor, it can easily be extended up to nine phase Machine.

This master thesis implements synchronous optimal modulation in FPGA for Neutral Point Clamped inverter feeding Induction machine. The result shows that synchronous optimal modulation can be applied for low switching frequency with lower total harmonics distortion.

Preface

Electric Drives, Power electronics and digital control system have always been my favourite subjects. I am extremely overwhelmed to do this project titled **Synchronous Optimal Modulation for Medium voltage Multi-phase Machines**. This master thesis requires the ideas of all these subjects. Doing this master thesis, my practical knowledge in these fields has immensely flourished.

I want to thank my supervisor Prof. Tore Undeland for his lecture in Design of power electronic converter which has augmented my knowledge in power electronics to great extent. I want to show my deep gratitude and respect towards my co-supervisor Roy Nilsen, Wartsila, Norway. I would like to express my co-supervisor how much immensely lucky I feel to get all those supports, guidance from the initiation of the project work. I would not have been able to fulfil this project work without the knowledge, encouragement and the positive spirit from my co-supervisor consistently.

My unlimited gratitude goes to Kjell Ljøkelsøy from SINTEF without whom timely completion of master thesis would have been impossible for me. He has helped me a lot during my learning of FPGA and embedded development kit (EDK) . The threshold of learning FPGA using Xilinx tools is very extremely high. The statement found on the Internet tells how FPGAs are seen by most software developers: "FPGA is a dark and scary corner of the universe teeming with HDLs, synthesis, place-and-route, and other unseen evils". He made this journey easier for me by being there in all my confusions. He was always so much supportive, encouraging and helpful.

I am also thankful to my classmate Biruk Bekele for supporting me through technical advice, profitable discussion. I also want to thank Sverre Skalleberg Gjerde for helping me learn VHDL language. My sincere thanks go to all the technical staffs of department of Electrical Engineering, NTNU for being so much supportive in dealing with the lab equipments. I am very grateful towards Prof. Indraman Tamrakar, Institute of Engineering, Nepal for always helping me out with my confusion in Induction Machine. Last but not the least I would like to thank Kabin Tamrakar for giving me ideas on programming skills and also for the care, moral support and encouragement throughout the thesis.

Mamta Maharjan,
Trondhiem, Norway

[illegible]

Contents

List of Figure

List of Table

Acronym

Acronym	**Description**
IGBT	Insulated Gate Bipolar Transistor
THD	Total Harmonic Distortion
WHTD	Weighted Total Harmonic Distortion
WHTD	Normalised Weighted Total Harmonic Distortion
FPGA	Field-programmable Gate Array
SOM	Synchronous Optimal Modulation
SPIM	Six Phase Induction Machine
CLB	Configurable Logic Block
HDL	Hardware Descriptive Language
VHDL	Very High Speed Integrated Circuits Hardware Descriptive Language
IP	Intellectual Property
LVDS	Low Voltage Differential Signal
ADC	Analog to Digital Converter
DSP	Digital Signal Processor
PWM	Pulse Width Modulator
NPC	Neutral Point Clamped
BJT	Bipolar Junction Transistor
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
USB	Universal Serial Bus
DRAM	Dynamic Random Access Memory
EROM	Erasable Read Only Memory
MAC	Media Access Controller MSPS
CMOS	Complementary Metal oxide Semiconductor
TTL	Transistor Transistor Logic
ISE	Integrated Software Environment
XPS	Xilinx Platform Studio
SDK	Software Development Kit
EDK	Embedded Development Kit
NPSM	Nine-phase Synchronous Machine
UCF	User Constraint File
MPD	Microprocessor Peripheral Description
MHS	Microprocessor Hardware Specification
PLB	Processor Local Bus
SPWM	Sinusoidal PWM
SVPWM	Space Vector PWM
BSP	Board Support Package

Chapter 1

1 Introduction

In this chapter, the background of this master thesis is presented. In addition to that the motivation to this master thesis is discussed. The literatures read during this master thesis are reviewed and presented here. Because of the time constraints, this master thesis has scope and limitation. These are also given in this chapter.

1.1 Background

The power electronic converters decouple three phase mains from the variable speed ac drives so the number of phases of ac machine do not have to be limited to three anymore [1]. This led to the development of multiphase machine. Even though the concept of multiphase machine was emerged some 40 years ago, its interest took new surge in recent years through the advancement of power electronics switching devices [2]. The pace of research in the field of multiphase machine was even more accelerated from 1990s along with the development of electric ship propulsion. Even though there are many other benefits of multiphase machine, they find their major application in area of electric ship propulsion [1].

Electric ship propulsion is beneficial as it provides precise control of the shaft speed, increased manoeuvrability, increased fuel efficiency, reduced environmental impact, and quiet operation [3]. Multiphase machine drives are preferred to the conventional three phase machine drives in electric ship propulsion. It is because the requirements of high power rating, high efficiency, reliability and fault tolerant operation of electric drives for ship propulsion can be met by multiphase machine drives [4].

Medium Voltage ac drives produce greater power than lower voltage drive. They have ability to control high rating electric motors that industrial load require. The higher voltage of MV drive also indicates the lower loss and use of smaller cables that outputs higher efficiency and lower system cost [5].

Wärtsilä, a global leader in ship power technology, has been conducting research on the application of multiphase machine and medium voltage drives in electric ship propulsion. This Master thesis is also done as the part of the research.

1.2 Motivation

The research of multiphase variable speed drives started from 1960s when inverter-fed ac drives were in the initial development stage. In six steps mode of three phase inverter operation, there was one particular problem with low frequency torque ripple. *"Since the lowest frequency torque ripple harmonic in an n-phase machine is caused by the time harmonics of the supply of the order 2n ± 1 (its frequency is 2n times higher than the supply frequency), an increase in the number of phases of the machine appeared as the best solution to the problem* "[1].

Previously, Voltage source pulse width modulated inverters use square wave or sinusoidal PWM strategies for low voltage ac drives [6]. As industrial load require higher rating ac drives, medium voltage ac drives based voltage source inverter are of higher demand [7]. Advancement in the development of semiconductor devices with higher voltage capabilities like insulated gate bipolar transistors (IGBTs), has allowed increasing the voltage level to medium voltage and power rating of the system up to several MVAs. These IGBTs must be operated in lower switching frequency to keep the switching loss to tolerable level. Once the switching frequency is reduced, appropriate modulation technique is required. The appropriate modulation technique is called Synchronous Optimal Modulation [8].

In order to use IGBTs for Medium Voltage drive, sophisticated technology like Synchronous Optimal Modulation (SOM) is the better choice as it minimizes total harmonic contents. This optimized PWM technique is extremely difficult to be implied by conventional analog circuitry hence they need modern microprocessor or digital hardware like DSP. These sophisticated techniques have access to the effective look-up tables on which this modulation technique can be implemented effectively[9].

Industrial electrical control system has progressed due to technology revolution. The revolution has outcome very powerful components to solve extremely complex control algorithm. The most recent advancement of extremely fast computing device is called field programmable gate array

(FPGA). FPGA allows real time computation of complex control algorithm in a few microseconds [10].

Therefore, it is of interest to implement this sophisticated modulation strategy like SOM in medium voltage drive by using field-programmable gate array (FPGA) components.

1.3 Literature review

For the industrial demand of high power, medium voltage drive is used. Medium voltage ac machines must be operated at low switching frequency to reduce the switching losses of the power semiconductor devices. Once the switching frequency is reduced, appropriate modulation technique like program modulation is required. In 1973 Patel and Hoft [11] generalised the method for eliminating fixed number of harmonic. They provided the solution for eliminating up to five harmonics. While the unlimited higher order harmonics can be attenuated by using filter circuits. Buja and Indri [12] in 1977 started to develop optimal PWM for AC motor. They developed the analytical procedure to calculate proper choice of commutation angles which minimise the rms value of the current harmonics. In 1992, Programmed PWM technique was used by Enjeti and Jakkli [13] to eliminate lower harmonics at the output of Neutral point clamped inverter topology. In 1994, Holtz [14] employed the method called Synchronous optimal pulsewidth modulation(SOM) for medium voltage drives at low switching frequency. SOM is type of Program Modulation pattern.

Murphy, Howard and Hoft[9] came with the idea in 1979 that these optimized PWM strategies are extremely difficult to realize with conventional analog circuitry, but they can be effectively implemented with using a look-up table accessed by modern microprocessor-based control techniques or digital hardware. Recently digital hardware like FPGA is developed which has exceptionally fast computation capability hence complex control algorithm can be computed in few microseconds. In 2007 Naouar and Monmasson [10] presented the interest of implementing digital controllers in AC machine using field-programmable gate array (FPGA) components. Numerous experimental results are given in order to illustrate the efficiency of FPGA-based solutions to achieve high-performance control of electrical systems.

1.4 Thesis Scope and Limitation

For generating synchronous optimal modulation technique, the optimal angles must be calculated. This calculation itself accepts lot of work. Hence such calculation is done in other collaborating master thesis. This can be referred from [15]. In this thesis, it is assumed that the optimal angles are already provided by the collaborating thesis. The other collaborating thesis is termed as software and processor routine in this master thesis.

Even though this project is meant to produce synchronous optimal modulation technique for multiphase machine up to 9 phases, this project generates modulation pulse only for three phases. Hence if this project is successful in implementing modulation pulse for three phases, only the slight modification can be done to make it appropriate for nine phases.

1.5 Organization of report

In Chapter 1, background, motivation, literature review, scope and limitation of the project are discussed. The theory behind the generation of Synchronous Optimal Modulation is provided in Chapter 2. The type of inverter used is three level Neutral Point Clamped Inverter. Since this project is for medium voltage, Insulated Gate Bipolar Transistor is used as switch. The implementation of three level modulator is done on extremely fast digital device called Field Programmable Gate Array (FPGA). The important portion of the project commence from Chapter 3 which introduces FPGA architecture, FPGA control card developed by SINTEF along with the description of tool called Xilinx which is used to program FPGA. The methodology applied in this project for implementing SOM by using FPGA is presented in Chapter 4 which contains the description of Intelligent Properties (IP) and the connection between various IPs used in this project. The output of the simulation is also shown in this chapter. This thesis also consists of laboratory works for verification of the theories. Therefore Chapter 5 discuss about the experimental setup of the project. Finally results, discussion are presented in Chapter 6 to verify the validity of digital circuitry created in FPGA to fulfil the objective of the project. At last conclusion and further works are presented in the Chapter 7.

.

Chapter 2

2 Theory

In this chapter, multiphase machines are described. In addition inverters and modulators needed for feeding the machine are discussed. The converter which is used is Neutral Clamped Three Level converter while new modulation technique called Synchronous Optimal Modulation is discussed along with its implication in this project.

2.1 Electric Propulsion

The shipping industry has advanced from its conventional era. The cost of propulsion has reduced significantly without increasing marine pollution [16]. Electrically powered ship requires energy source for all the ship's functions, including propulsion. They are quieter, are less susceptible to vibration and are comfortable for everyone on board [17].

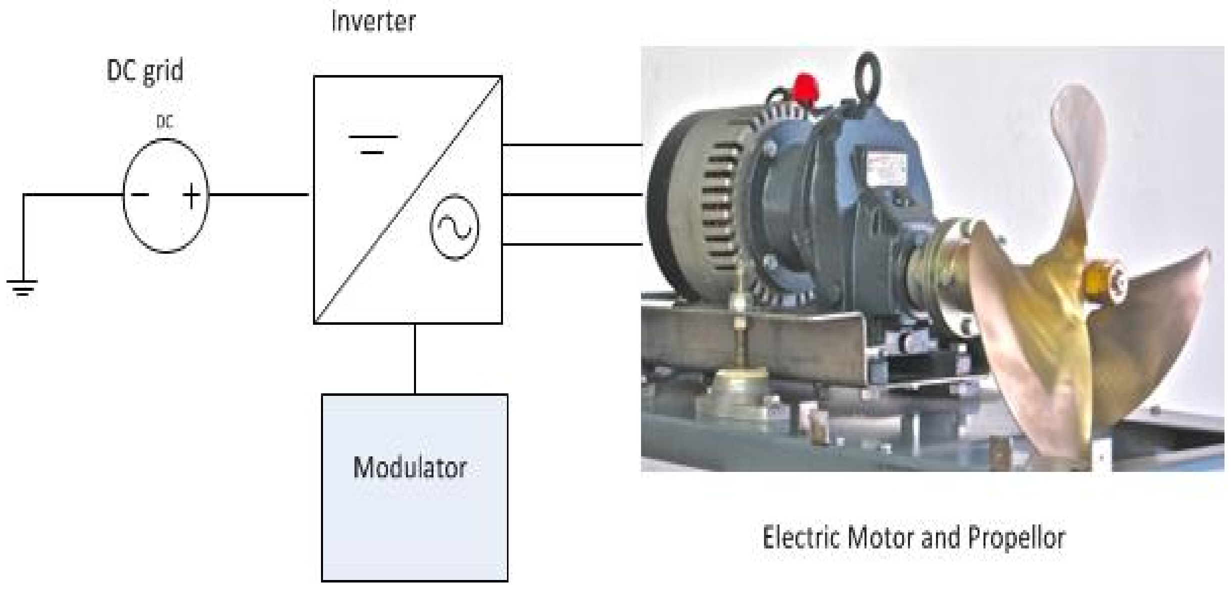


Figure 2-1 Overview of electric propulsion.

The schematic overview of the main electrical and automation components for electric propulsion is shown in Figure 2-1. The propeller shaft of the ship is connected to large motors, which is A.C driven and are known as propulsion motors. These propulsion motors are fed by inverter. Inverters are controlled by Modulator to convert the DC from the DC grid in ships to required AC needed by motor. Figure shows three phase propulsion motor. But the intense research is going on to replace the three phase motor by multiphase machine.

2.2 Multiphase Machine

The power electronic converters decouple three phase mains from the variable speed ac drives so the number of phases of ac machine do not have to be limited to three anymore [1]. This led to the development of multiphase machine. Even though the concept of multiphase machine was emerged some 40 years ago, its interest took new surge in recent years through the advancement of power electronics switching devices [2]. The advantage of multiphase machine include high power ratings, efficiency, reliability and fault tolerant operation. Its advantages over three phase induction machine are listed below.

• The stator copper loss for multiphase machine are less than three phase machine assuming the same torque and same speed [18].
• The harmonics of stator current are of higher order and more attenuated than three phase machine which will reduce the torque pulsation[18].
• As the number of inverter switches increases, ratings of inverter switches decrease which helps to reduce the cost[3].

Even though this thesis deals with three phase Induction machine, the main motivation of this project is to make suitable modulation technique which can be extended up to nine phase machine like Nine-phase Synchronous Machine (NPSM). There are not many NPSM in industries. But it has many advantages over three phase machine that its research is intensively going on. The nine-phase Synchronous Machine can be presented by windings as shown in Figure 2-2.

Different winding layouts can be chosen for nine-phase machines. For the physical modelling, nine phase synchronous machine with 3 sets of three phase with 20^0 phase shift is used [19]. The modelling of multiphase machine is out of scope of this project. This project deals with modulation technique developed for the multiphase machine.

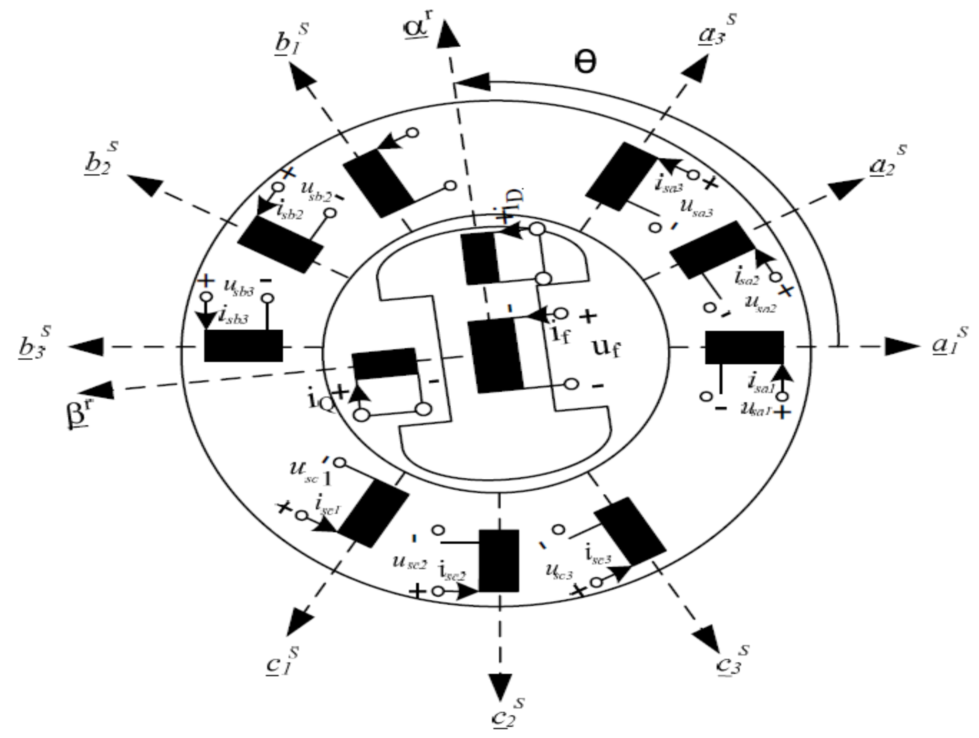


Figure 2-2 Nine Phase Synchronous Machine [19].

Another kind of multiphase machine can be Six Phase Induction machine (SPIM). It consists of 2 sets of three phase windings. They have gathered lot of interest of drive engineers for the industrial applications. The SPIM can be of two types, symmetric and asymmetric depending upon the winding arrangements as shown in the Figure 2-3, if the axes of the individual phase windings are spaced equiangular as shown in (a) then the machine is called symmetrical SPIM. In asymmetrical SPIM the winding axes are spaced as shown in the Figure 2-3(b).

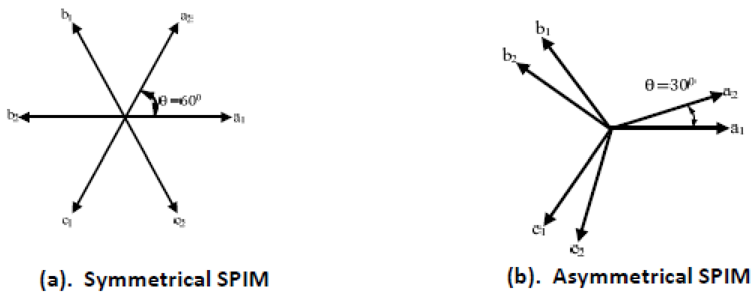


Figure 2-3 Stator winding types [3].

Asymmetrical SPIM is used for the research purpose than symmetrical as it has less pulsating torque and less harmonic currents as compared to symmetric SPIM [20].

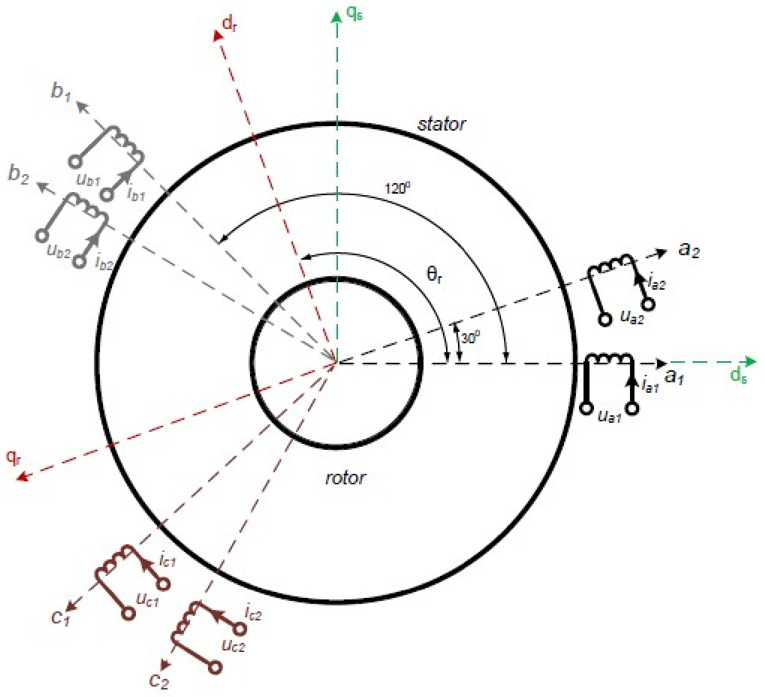


Figure 2-4 Six Phase Induction Machine(SPIM) [3].

2.2.1 Harmonic Losses in Induction Machine

Power electronic converter lies between three phase mains and the induction machine. The voltage generated by power electronics device to be fed to the induction machine is not pure sinusoidal. Fourier Analysis can be done to such non sinusoidal signal which helps to split them into harmonic components. The typical definition for a harmonic is "*a sinusoidal component of a periodic wave or quantity having a frequency that is an integral multiple of the fundamental frequency*." Except the fundamental frequency, all other higher harmonic components participate in losses of induction machine. Motors are susceptible to harmonic pollution. In a balanced system, the fundamental, 4th, 7th and other such types of voltage harmonics are positive sequence and they rotate the motor forward. The 2nd, 5th, 8th voltage harmonics are negative sequence, which try to rotate motor backward while the 3rd, 6th, 9th voltage harmonics are zero sequence, which just heat up the motor[21] .

Table 2-1 Harmonics and sequence [21].

HARMONIC	FUND	2ND	3RD	4TH	5TH	6TH	7TH	ETC
SEQUENCE	+	-	0	+	-	0	+	.....

However the losses are not divided evenly over the harmonic spectra. The lower harmonic components contribute more to losses than the higher harmonic components [22]. The reason is lower harmonics indicates lower impedance increasing harmonic current. The higher order harmonics produce lower harmonic current and they can be eliminated easily by using filter circuits in the output stage of the inverter [11] . The filters to attenuate higher order of harmonic components are smaller in size, weight and cost efficient [23].

The uneven distribution of loss in the harmonic spectra require appropriate performance indicator that weights the lower frequency spectra of harmonic components more dominant than the high spectra of the harmonic components [22]. Such indicator is commonly known as Weighted Total Harmonic Distortion (WTHD), which can be calculated by normalizing the total current harmonic distortion by the maximum inrush current. This theory after simplification becomes as equation below[22].

$$WTHD = \frac{\sqrt{\sum_{n=2}^{\infty}(\frac{Un}{n})^2}}{U1}$$

Where, n is nth number of harmonic component. From equation it is clear that the lower order harmonics participate more due to their high $1/n^2$ factor compared to that of higher order harmonics.

Harmonic currents also introduce unwanted torque pulsations in induction machines due to the interaction of the harmonic currents and the magnetic field of fundamental frequency. This would create mechanical oscillations and eventual wear of the machine.

2.3 Three Level Converter

The fundamental advantages of the Multilevel Converter topologies are low distorted output waveforms and limited voltage stress on the switching devices [24]. In this Project, three level converters called Neutral Point Clamped (NPC) is implemented as shown in Figure 2-5. Further, the NPC inverter is particularly suitable in high-voltage applications since it guarantees equal voltage sharing of series-connected power devices in each phase [13].The figure is shown for three phase induction motor but if the numbers of bridge leg are added then it can be extended to any multiphase machine. The working principle of three level converters can be understood by analysing single branch as illustrated in Figure 2-6.

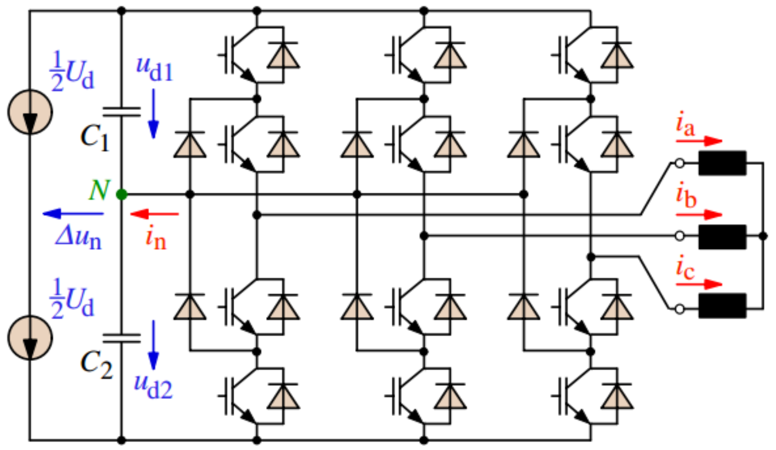


Figure 2-5 Three level inverter NPC Technology [24].

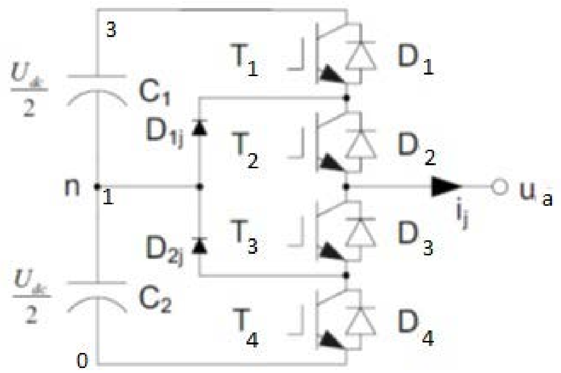


Figure 2-6 Bridge leg of three level converter [24].

Figure 2-6 shows one phase-leg of a three-level, three phase converter. This topology can produce three levels at the output with respect to the neutral point n. Table 2-2 indicates the possible output voltages for each switching state of device. There are three switching levels or states which are +1, 0, -1.

Table 2-2 Switching states in three level converter[24].

Switching of each IGBT				$\frac{2Uoj}{Udc}$ **(analog switching state)**	**Digital switching state(taking T1 and T2)**	**Ua0**
T_1	T_2	T_3	T_4			
1	1	0	0	+1	11 (3)	U_{dc}
0	1	1	0	0	01 (1)	$\frac{U_{dc}}{2}$
0	0	1	1	-1	00 (0)	0

If the signals to upper two IGBTs are considered for three level converter, then +1 can be obtained by giving signal 11 to T_1 and $T_{2,}$ while 0 can be obtained by giving signal 01. Similarly -1 can be obtained by giving signal 00 to T_1 and $T_{2.}$ Hence 11, 01, 00 are the states that can define the signal +1, 0, -1 or in decimal, the states are 3,1,0. Here 10 (2) is the forbidden state. Signal to T3 and T4 are complementary to T1 and T2 respectively.

Conventionally, the three level modulating signals for the IGBTs are generated by comparing two carrier waves with the sinusoidal wave as shown in Figure 2-7. If sinusoidal wave is greater than carrier signal, switch is turned on otherwise turned off. The upper carrier wave decides the switching state of T_1 while the lower carrier decides the switching state of T_2. The two switches below, T_3 and T_4 are complementary to two switches above T_1 and T_2 respectively.

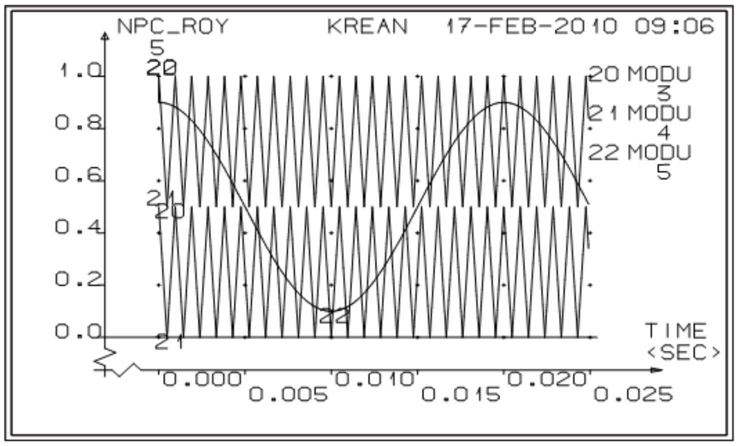


Figure 2-7 Naturally sampled PWM for 3 level converter [25].

By combining the different states, it is possible to get waveform close to sinusoidal averaged bridge leg output as shown in figure Figure 2-8. There are three states in bridge leg voltage. Line to Line voltage is also calculated. The voltage between two phases, the line-to-line voltage can achieve five different voltage levels which are +2,+1,0,-1,-2.

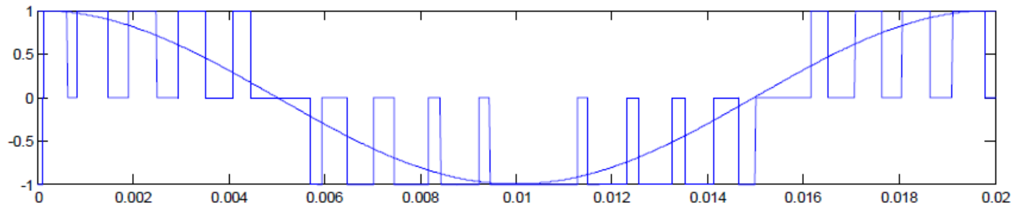


Figure 2-8 Bridge leg voltage [26].

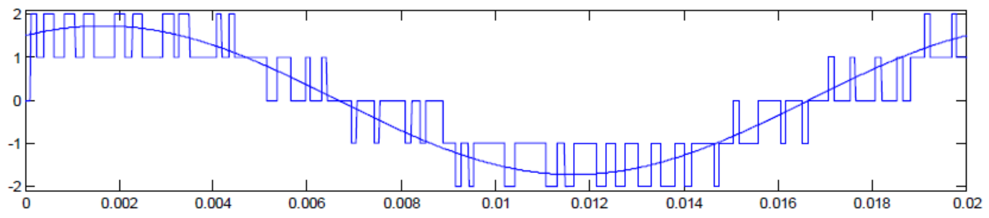


Figure 2-9 Line to Line voltage[26].

2.3.1 Switching loss in voltage source Inverter

The switching power loss P_{sv} of a Voltage source Inverter with a sinusoidal ac line current is estimated using the following relation [27]:

$$P_{sv} = \frac{6}{\pi} . f_s . \left(E_{on,1} + E_{off,1} + E_{off,D}\right) . \frac{V_{dc}}{V_{ref}} . \frac{I_L}{I_{ref}}$$

Where f_s is the switching frequency, $E_{on,1}$ and $E_{off,1}$ are the turn-on and the turn-off energies of the IGBT respectively, $E_{off,D}$ is the turn-off energy in the power modules' diode due to reverse recovery charge current, V_{dc} is the dc link voltage, I_L is the peak value of the ac line current assumed to be sinusoidal and V_{ref} and I_{ref} are the reference voltage and current where the switching energies provided by data sheets are given.

It is seen that switching loss increases with increasing switching frequency and increase in Dc link voltage. If Medium voltage is used in order to increase the power rating in drive system then switching loss is also increased along with it. Therefore as the compensation, switching frequency can be reduced. For medium voltage and low switching frequency, the perfect choice of power conductor device is IGBT. The construction of IGBT is made is such a way that it has higher reverse voltage blocking capability being suitable for medium voltage drive and lower operating switching frequency which shows its suitability for lower operating frequency. However if the switching frequency is low then there will be problem in modulation which will be discussed in the remaining sections.

2.3.2 Harmonic content of voltage source Inverter

The output of the voltage source converter is not pure sinusoidal wave. Hence it possesses certain amount of harmonics at their output due to finite switching frequency. The output voltage of the converter can be analyzed by Fourier series.

$$V(t) = V_o + \sum_{h=1}^{\infty} V_h \ (t)$$

The three level voltage source converter is shown in Figure 2-5.

If the load is balanced three phase load then some harmonics in line to line voltage will be cancelled out even though they exist in bridge leg voltage. Let U_{a0}, U_{b0} and U_{c0} are the voltages at the phase outputs (or bridge leg) of the inverter with respect to the dc-link then line-to-line voltage will be [25].

$$U_{ab}(t) = U_{a0}(t) - U_{b0}(t)$$
$$U_{bc}(t) = U_{b0}(t) - U_{c0}(t)$$
$$U_{ca}(t) = U_{c0}(t) - U_{a0}(t)$$

The neutral voltage $U_0 \ (t)$ is common in all the bridge leg hence the above equation can be further expressed as.

$$\mathrm{U_{a0}(t) - U_{b0}(t) = U_a(t) - U_b(t)}$$
$$\mathrm{U_{b0}(t) - U_{c0}(t) = U_b(t) - U_c(t)}$$
$$\mathrm{U_{c0}(t) - U_{a0}(t) = U_c(t) - U_a(t)}$$

In balanced three phase load, phases are 120^0 out of phase with each other, so at any instant the sum of all the phase voltage sum upto zero.

$$U_{a0} + U_{b0} + U_{c0} = 0$$

$$U_a - U_0 + U_b - U_0 + U_c - U_0 = 0$$

The neutral voltage U_0 can be expressed as function of the phase voltages of the load as follows:

$$U_0(t) = \frac{U_a(t) + U_b(t) + U_c(t)}{3}$$

Using this value and inserting in above equations, the following equations can be obtained.

$$U_a(t) = \frac{1}{3}\cdot\left(2\cdot U_{a0}(t) - U_{b0}(t) - U_{c0}(t)\right) + U_0$$
$$U_b(t) = \frac{1}{3}\cdot\left(2\cdot U_{b0}(t) - U_{c0}(t) - U_{a0}(t)\right) + U_0$$
$$U_c(t) = \frac{1}{3}\cdot\left(2\cdot U_{c0}(t) - U_{a0}(t) - U_{b0}(t)\right) + U_0$$

The classical dq0-transformation one obtains:

$$\begin{bmatrix} U_{sd}^s \\ U_{sq}^s \\ U_{s0} \end{bmatrix} = \frac{1}{3}\cdot\begin{bmatrix} 2 & -1 & -1 \\ 0 & \sqrt{3} & -\sqrt{3} \\ 1 & 1 & 1 \end{bmatrix}\cdot\begin{bmatrix} U_a \\ U_b \\ U_c \end{bmatrix} = \frac{1}{3}\cdot\begin{bmatrix} 2\cdot U_{a0}(t) - U_{b0}(t) - U_{c0}(t) \\ \sqrt{3}\cdot\left(U_{b0}(t) - U_{c0}(t)\right) \\ 3\cdot U_0(t) \end{bmatrix}$$

The d-axis voltage component is equal to the difference between two line voltages. The q-axis voltage component is proportional to the line voltage:

$$U_{sd}^s(t) = \frac{1}{3}\cdot\left(2\cdot U_{a0}(t) - U_{b0}(t) - U_{c0}(t)\right) = \frac{1}{3}\cdot\left(U_{ab}(t) - U_{ca}(t)\right)$$
$$U_{sq}^s(t) = \frac{1}{\sqrt{3}}\cdot\left(U_b(t) - U_c(t)\right) = \frac{1}{\sqrt{3}}\cdot\left(U_{b0}(t) - U_{c0}(t)\right) = \frac{U_{bc}(t)}{\sqrt{3}}$$

The d-component will have the same harmonics as the difference between two line voltages, while the q-component has the same harmonics as one line voltage. As common components in all bridge leg voltages will not be present in the load. This indicates that motor as a load will not see the harmonics presented in the bridge leg voltage but it only sees the harmonics presented in the line-to-line voltage. Some harmonics in line to line will be cancelled out even though they exist in bridge leg voltage. If the PWM patterns of the three bridge legs are 120 degree phase shifted, i.e. the pattern is three phase symmetrical so no harmonics of multiple of three, exists in the d- and q-axis system even if they exist in the bridge leg voltages. Two important conclusions are:

- Even signals in the bridge leg voltages is cancelled in the line voltages
- Harmonics multiple with 3 do not exist in the d- and q- voltages.

2.3.3 DC Bus Balancing

2.3.3.1 Introduction

In NPC converter as shown in Figure 2-5, there are two capacitors. Depending upon the type of neutral point current, the charging of two capacitors may differ creating voltage difference between them. This phenomenon can increase the voltage stress in the equipment connected across them. In the worst case, all the DC link voltage could be across only one of the capacitors. Hence DC Bus balancing is necessary to keep the voltage difference between the two capacitors small. The output voltage also does not follow the reference value and increase the harmonic distortion in presence of unbalance in capacitor voltages [26]. Space Vector method can be used in order to control the voltage across the capacitor. In space vector, Phase R,Y,B are located 120^0 out of phase to each other in vector space. A three level converter has 27 vectors to create the desired voltage with 19 different states as shown in Figure 2-10.

NPC converter has three states namely 0,1,3 where 2 is the forbidden state. In Figure 2-10 MSB consists the state of R phase while LSB consist of state of B phase. The currents flowing in different states for a bridge leg are given in Figure 2-11. In level 3, the current flows from the top, in level 1, there is flow of current from neutral point while in level 0, there is flow of current from the bottom.

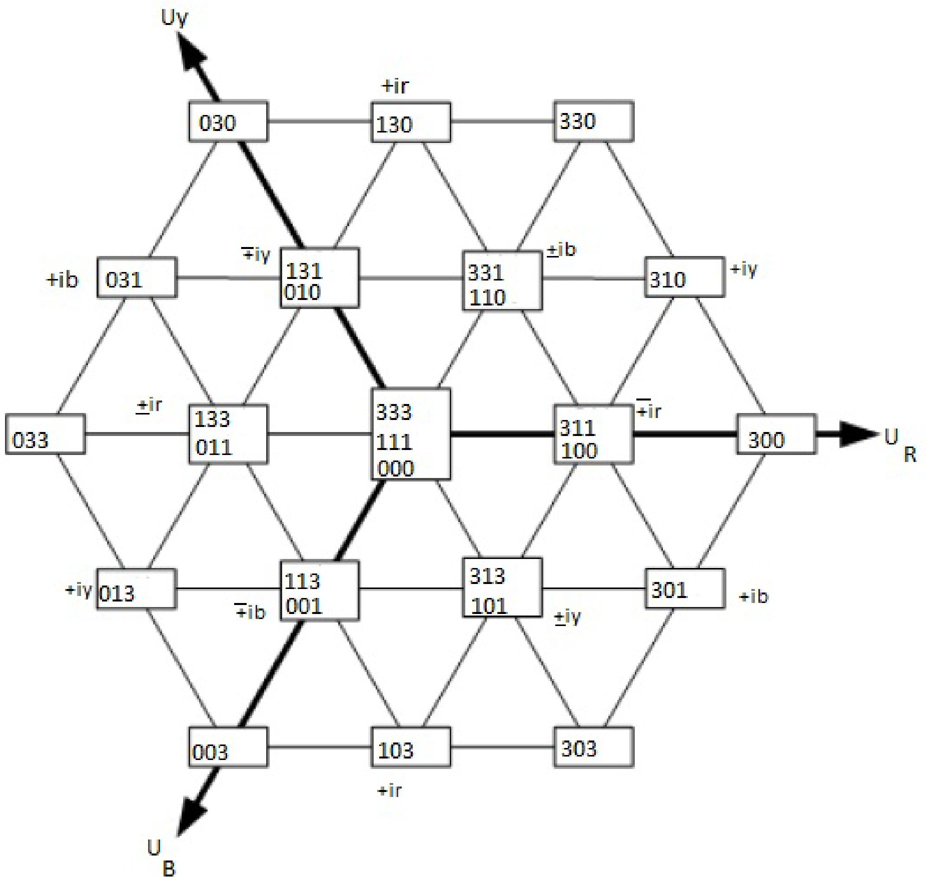


Figure 2-10 Space vector diagram for three level converter.

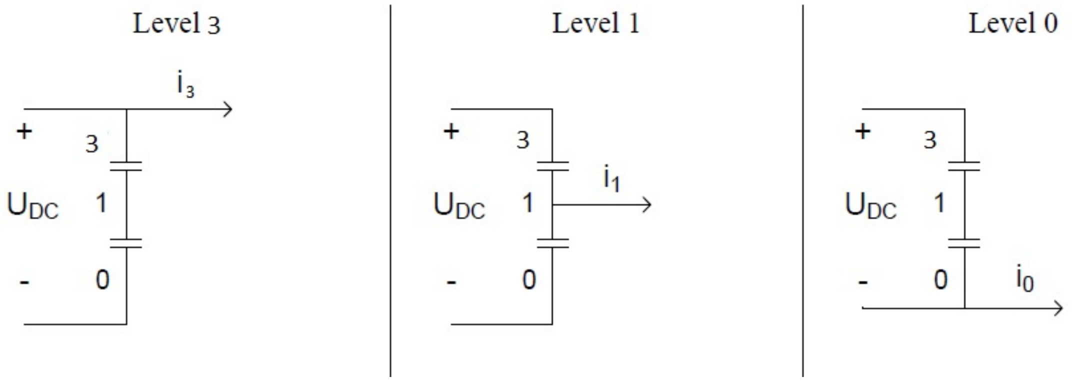


Figure 2-11 Position of current at different level [26].

Vectors can be divided into four different groups according to magnitude of the vectors. The overview is given in Table 2-3.

Table 2-3 Overview of space vectors[26].

Zero Vectors	Small Vectors	Medium Vectors	Large Vectors
(000)	(100)(311)(110)(331)	(310)(130)	(300)(330)
(111)	(131)(010)(011)(133)	(031)(013)	(030)(033)
(333)	(001)(113)(101)(313)	(103)(301)	(003)(303)

The vectors are $U_s = \frac{1}{2} U_{DC}. e^{j\theta}$

$$U_m = \frac{1}{\sqrt{3}} U_{DC}. e^{j\theta}$$

$$U_l = \frac{2}{3} U_{DC}. e^{j\theta}$$

Where, U_{DC} is DC link voltage. U_s, U_m, U_l means small, medium, large vectors respectively.

The neural current determines the charging and discharging of capacitor. A zero vector is when all the bridge legs are connected to the same point and all of the line-to-line voltages are zero. Hence zero vector does not create voltage difference. The six large vectors also do not affect unbalance in voltage in capacitor. It is because the large vector do not have neutral current as it does not involve level 1. Therefore only six medium vectors and six pairs (twelve) of small vectors influence voltage balance. The overview is given in Table 2-4. The sum of current in three phase equal to zero. The same table shows the neutral current flowing at different vector combination. For instance 100, Ir flows through neutral point in R bridge leg while Iy and Ib flows through the bottom of the Y and B bridge leg as shown in Figure 2-11.

Table 2-4 Overview of relation between space vectors and neutral current [26].

Positive small vectors	I_{Np}	Negative small Vectors	I_{Np}	Medium vectors	I_{Np}
100	Ir	311	-Ir	310	Iy
331	Ib	110	-Ib	130	Ir
010	Iy	131	-Iy	031	Ib
133	Ir	011	-Ir	013	Iy
001	Ib	113	-Ib	103	Ir
313	Iy	101	-Iy	301	Ib

Similarly for 311, Ir flows through top of R bridge leg while Iy and Ib flows through the neutral point in Y and B bridge leg. But we know, Ir+Iy+Ib=0 and Iy+Ib=-Ir. So it can be said that the total neutral current flowing for the state 311 is –Ir as shown in Table 2-4.

The positive neutral current will charge the upper capacitor while discharge the lower capacitor while negative neutral current charges oppositely. It can be seen in Figure 2-12, where the direction of current is taken positive. The figure clearly shows how this positive current is charging the upper capacitor and discharging the lower capacitor. For the negative current, the phenomenon is just opposite. In the Figure 2-10 the belonging phase currents are shown with a plus and minus sign. The plus sign is representing a positive vector, while a negative sign is representing a negative vector.

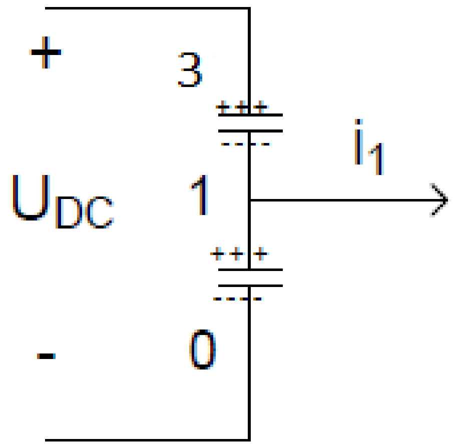


Figure 2-12 Charging of upper capacitor and discharging of lower capacitor.

2.3.3.2 DC balancing

In order to remove the unbalance in voltage between two capacitors, six small vector pairs are controlled. When a small vector pair is involved a total duty cycle of d1 is calculated, where d1 is a duty cycle of T_{tri}. This is given in detail in [26].This duty cycle is divided into two for DC bus balancing.

$$d_1 = d_{1,p} + d_{1,n}$$

Where $d_{1,n}$= duty cycle of negative vector

$d_{1,p}$ =duty cycle of positive vector

These two duty cycle will be equal in the case of balanced voltage. For unbalanced voltage control parameter f_1 is introduced to define positive and negative vector.

$d_{1,p} = f_1 . d_1$ and $d_{1,n} = (1 - f_1). d1$

The value of f_1 must be calculated which depends on phase current and capacitor voltage. For the situation when voltage in upper capacitor is greater than the lower capacitor or $U_{dcu} > U_{dcl}$, such vector which discharge the upper and charge the lower must be selected. That means that negative vector must be selected. It is already defined in above section that negative vector will give negative neutral current which discharges the upper capacitor and charge the lower capacitor. For the opposite case, positive vector must be selected. The value of f_1 should be 0.5 at balanced situations and it should be regulated depending on the voltage difference and direction of the current. The equation of f_1 is as shown below.

$$f_1 = 0.5 - k.\frac{abs(U_{dcu} - U_{dcl})}{(U_{dcu} + U_{dcl})}$$

K is in this case a constant which has to be given a proper value. As it can be seen from the formula, the value of f_1 is depending on the difference in capacitor voltage.

2.4 Modulation Strategy

At present, voltage source converters are mostly used in electrical drives. These converters utilize capacitors in the DC-link to store temporarily electrical energy. Switching the power electronic devices allows the DC voltage to be modulated which can result in a variable voltage and frequency waveform. The purpose of the modulator is to generate the required switching signals for these switching devices.

There are various ways of modulation. The overview of a modulation method is listed in Figure 2-13. Multilevel Modulation strategies are mainly divided into Synchronous and Asynchronous Modulation[25]. The division is based on the value of index called frequency modulation index.

$m_f = \frac{f_c}{f_s}$. Here m_f is called frequency modulation index, f_c is called carrier frequency and f_s is called reference fundamental frequency or stator frequency.

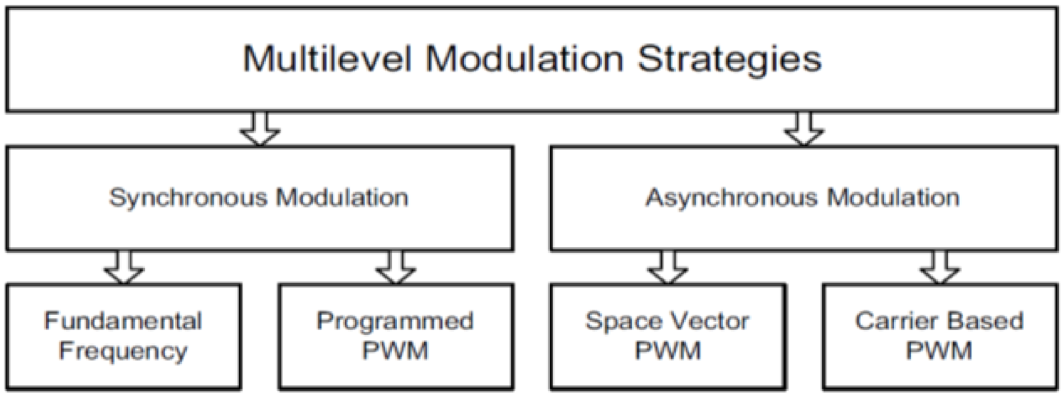


Figure 2-13 Modulation strategies[24].

The modulation strategies are as shown in Figure 2-13. Modulation strategies are divided into two major areas which are as follows:

a) **Asynchronous Modulation**:

When stator frequency is very low, it impacts frequency modulation to be greater than 20. There will be high pulse number. In such scenario, zero crossing between carrier wave and reference signal is not synchronized even if it results in unequal number of pulses in positive and negative half cycle. For such high pulse number, difference of few pulse numbers between positive and negative half cycle does not make huge impact hence the effect of sub-harmonic component is minimal. Such kind of modulation in which zero crossings between the carrier wave and reference signal are not synchronized is called asynchronous modulation.

There are two types of asynchronous modulation.

i) Space Vector PWM

ii) Carrier Based PWM

Among these two type Carrier Based is discussed here.

- **Carrier Based PWM :**

In Carrier Based PWM, generally there is reference wave which is compared with carrier wave. The switching pulses are determined by the result of the comparison. If reference wave is greater in magnitude than the carrier wave then pulse is ON otherwise OFF. Carrier Based PWM is further divided according to the sampling of the reference wave. They are

a) Naturally sampled PWM

b) Regularly Sampled PWM

- **Naturally sampled Pulse Width Modulation***:*

This is the classical method where analog circuitry like amplifier is used for comparison of carrier wave and reference wave. The reference voltage then change continuously within the triangular period (carrier wave) as in Figure 2-14. If continuously changing reference wave is greater than triangular wave, the switching pulse is ON otherwise OFF.

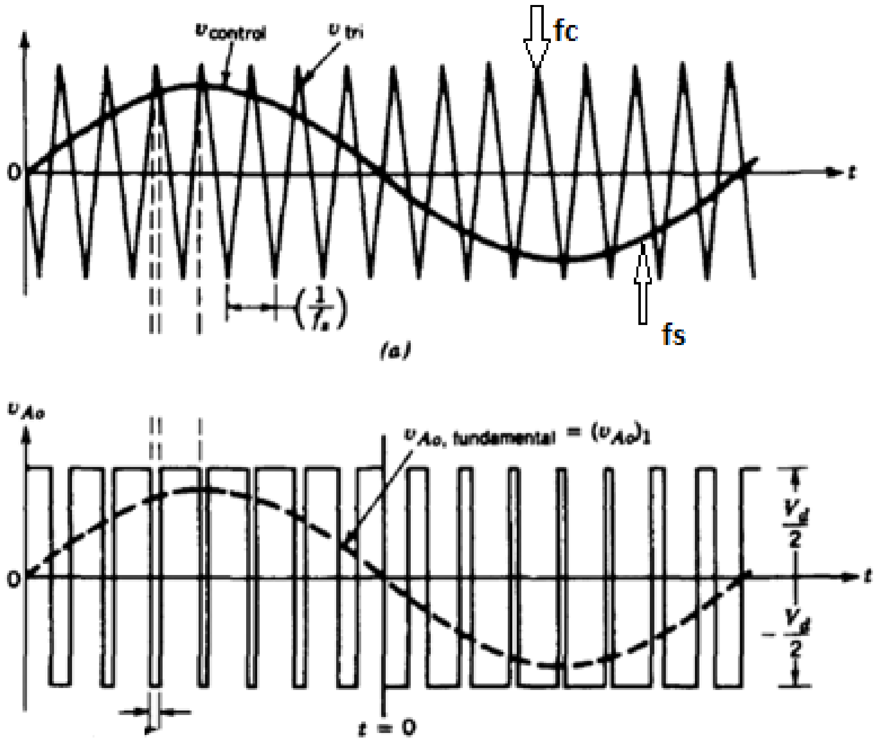


Figure 2-14 Naturally sampled PWM[23].

- **Regular sampled Pulse Width Modulation*:***

A method which is easier to implement in digital form is the regular sampled PWM. In this case the reference voltage is sampled at top and bottom of the triangular wave and then kept constant until next sample. This sampled reference wave is compared with the magnitude of triangular wave and switching occurs. Similar to Naturally Sampled PWM, the pulse is ON if sampled reference wave is greater than triangular wave and OFF for the opposite case. This method is further divided into:

- ***Symmetrical Regular sampled PWM***
- ***Asymmetrical Regular sampled PWM***

- ***Symmetrical Regular sampled PWM:*** The reference voltage is only sampled at the top of the triangular wave and kept constant within the complete triangular period.
- ***Asymmetrical Regular sampled PWM:*** The reference voltage is sampled both at the top and bottom of the triangular wave and kept constant within half the triangular period.

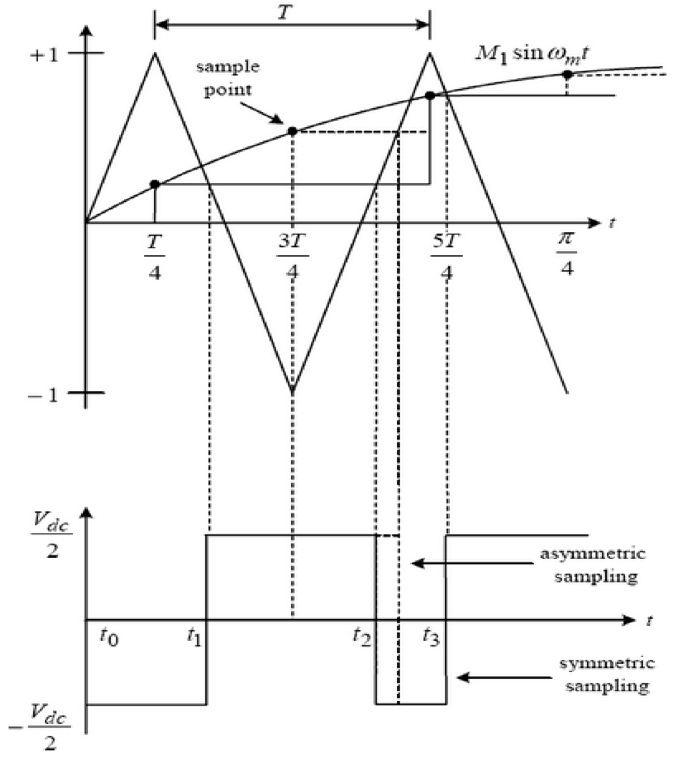


Figure 2-15 Symmetric and asymmetric regular sampled PWM.

The reference wave is stator frequency, which keep on changing. There will be change in the carrier wave as well if frequency modulation is kept constant. As we know the formula for the frequency modulation,

$$m_f = \frac{f_c}{f_s}$$

where m_f is called frequency modulation index, f_c is called carrier frequency (or switching frequency) and f_s is called reference fundamental frequency.

$$f_c = m_f * f_s$$

For, the application in motor, for the variable speed drive, the stator frequency (or fundamental frequency f_s) may vary. For particular m_f, f_c varies proportional to f_s. However to keep the carrier frequency constant, close to its maximum value, frequency modulation index must decrease when stator frequency is increased or vice versa as seen in Figure 2-16. In order to keep carrier frequency constant, there is change in the ratio (m_f) in the different ranges of f_c.

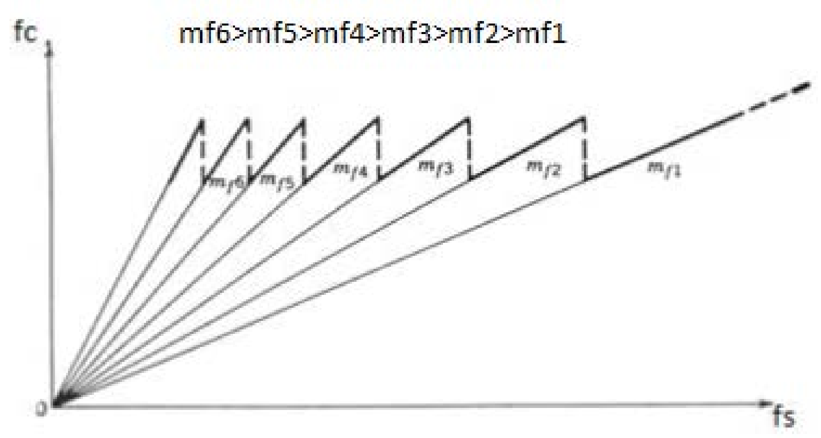


Figure 2-16 Carrier based PWM [23].

b) **Synchronous Modulation**:

For small values of frequency modulation (20 or less), the number of pulse are lower hence even the difference in pulse number in positive and negative half cycle by one unit will generate sub harmonics. To prevent from such scenario, the carrier waveform signal and the control signal should be synchronized to each other. That means the zero crossing of carrier wave (triangular) and reference wave (sinusoidal) should coincide in order to prevent from sub harmonic. This type of modulation is called synchronous modulation. As a result of this, number of pulses in positive half cycle is equal to that in negative half cycle. The average switching frequency is f_c /2. This type of modulation is important for low pulse number whose violation can result in sub-harmonics in motor voltage. Synchronous Modulation is further divided into two parts called fundamental frequency synchronous PWM and Program PWM.

i) ***Fundamental frequency Synchronous PWM***:

ii) ***Program Modulation PWM***:

Among these two, this thesis deals with the second type.

- **Program Modulation PWM**

This method pre-calculates optimal switching angles within a period of the fundamental period $1/f_s$ to minimize the harmonic distortion. But these switching angles are not formed by carrier wave like in conventional way. This pattern or angles are programmed, or stored, as look-up tables in hardware. This type of modulation is termed as Program Modulation. Since this is also synchronous modulation the number of pulses in positive and negative half cycle are equal. This is explained in detail in section below.

2.5 Program Modulation Technique

2.5.1 Background

BJT and MOSFET have characteristic that complement each other. BJT have lower conduction losses in the on state, especially in devices with larger blocking voltages, but have low operating frequency. MOSFETs can be turned on and off much faster but their on state conduction losses are larger, especially in devices rated for higher blocking voltages. Hence device is made whose performance is midway between BJT and MOSFET called Insulated Gate Bipolar Transistor (IGBT). Its switching frequency is higher than a comparable BJT but lower than MOSFET whereas the on state losses are smaller than MOSFET and are comparable with those of BJT for larger blocking voltage [23].

It is shown in section 2.3.1, switching loss increases with increasing switching frequency and increase in Dc link voltage. If Medium voltage is used in order to increase the power rating in drive system then switching loss is also increased along with it. Therefore as the compensation, switching frequency can be reduced. For medium voltage and low switching frequency, the perfect choice of power conductor device is IGBT. The construction of IGBT is made is such a way that it has higher reverse voltage blocking capability and lower operating switching frequency.

For the medium voltage drive, the inverter will be designed for a 6.6 kV output voltage, which means a dc-link voltage of approximately 10 kV. The most realistic choice of IGBTs are those with a blocking voltage of 4.5 kV [25]. Hence series connection of devices is required in a three Level inverter. In these devices the switching loss is a quite large part of the total losses. Hence the switching frequency of IGBT must be lower, in the range of 200-1000Hz.

In the equation, $m_f = \frac{f_c}{f_s}$, f_s=stator frequency and f_c=carrier frequency or switching frequency

m_f= frequency modulation index. For lower frequency, the value of frequency modulation is low (20 or less). Lower carrier frequency results in lower pulse number hence even the difference of pulse in positive and negative cycle by one unit will make huge difference. However such kind of equalization is not necessary in higher carrier frequency.

Hence, the lower carrier/switching frequency demand synchronized modulation. Among two types of synchronous modulation, Programmed Modulation is introduced here. Program Modulation pre-calculates optimal switching angles within a period of the fundamental period 1/fs. These patterns or angles are programmed, or stored, as look-up tables in hardware [25].

2.5.2 Program Modulation

Programmed Modulation is a synchronous modulation technique. In Programmed Modulation switching events can take place freely over the fundamental period as indicated in Figure 2-17. However, these switching instances are not given by a carrier signal. These switching instances are pre-calculated by some software program. The Figure 2-17 is the output for the program modulation for three level converter for one phase. This method is described initially in [13] and revised in [22] and [25]. For two level converter this method is initially studied in [11]. This type of modulation signal is generated by software program.

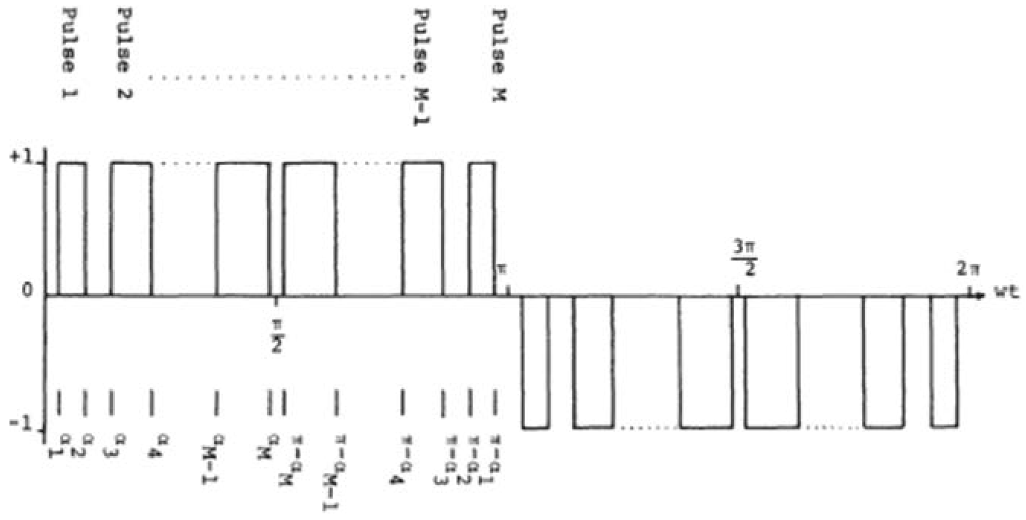


Figure 2-17 Programmed PWM pattern (M=N)[22].

In Figure 2-17, Instead of M, N will be used in this thesis. There are some parameters which characterize this signal which are number of pulses in a half period (N), fundamental frequency or stator frequency (f_s) and switching frequency of IGBT (f_{sw}).

N is number of pulses per half period. In addition, the number of turn-on of upper switch is N and number of turn-off is N as well in one half period. The total number of commutations is thus 2*N per half period and 4*N per period. In a complete three-phase inverter this becomes 12*N switching. The average switching frequency over one period for the upper switch is thus:

$$f_{sw1,avg} = N \cdot f_s \qquad \text{where } f_s \text{ is stator frequency.}$$

If this is compared with a carrier based PWM, with the same number of pulses N per half period the equivalent ratio m_f becomes:

$$N = \begin{cases} 0.5 \cdot (m_f + 1) \\ 0.5 \cdot (m_f - 1) \end{cases} \qquad m_f = \frac{f_c}{f_s} = 3,9,15,21,27,......$$

Program Modulation technique is synchronous modulation. Hence at first it is needed to check if synchronous or asynchronous modulation has to be applied. At low speed and thus frequency f_s, asynchronous modulation can be used with a fixed switching frequency of 200 Hz. For 200 Hz max switching frequency this gives a max motor frequency f_s, with N=10 (even):

$$f_{s1} = \frac{f_c}{m_f} = \frac{200}{21} = 9.52 Hz$$

Above this stator frequency, synchronous modulation is chosen. This also means below N=10 (below$m_f = 21$), synchronous modulation is chosen.

For synchronous modulation in motor drives using Program Modulation, the number N has to be selected to give a proper curve similar to Figure 2-16, but with N as parameter instead of m_f and f_{sw} instead of f_c, where f_{sw} is called switching frequency while f_c is called carrier frequency. The Figure 2-16 is modified for program modulation and it looks like Figure 2-18

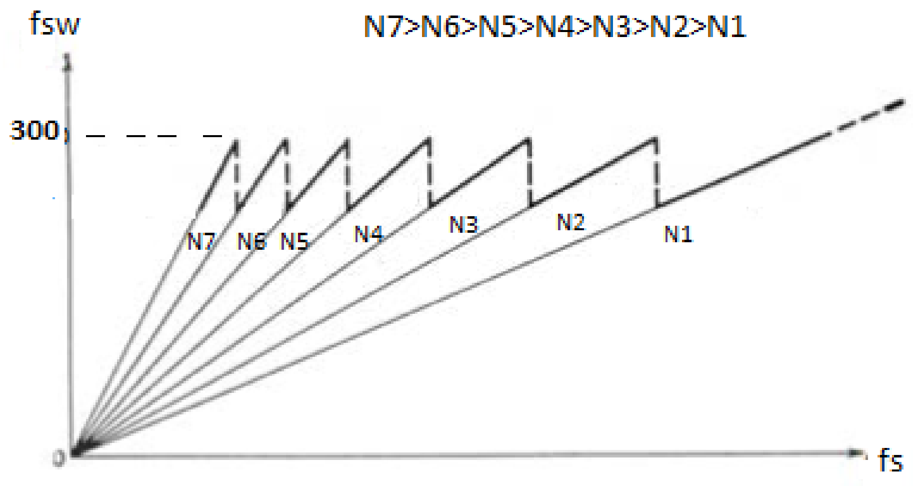


Figure 2-18 Programmed PWM.

Switching frequency increases with increase in stator frequency for constant N. However the maximum operating frequency for IGBT is 300 Hz. In this process, if switching frequency becomes more than 300Hz then N is decreased as stator frequency increases.

2.6 Synchronous Optimal Modulation

In Program Modulation, these switching instances are pre-calculated by some software program, in order to achieve a purpose by applying some kind of constraint in Fourier series of the fundamental frequency. Usually these methods are divided into **Harmonic Elimination PWM** and **Minimum-loss PWM** methods. Harmonic Elimination PWM is used for the active rectifier which is out of scope of this thesis. For motor drive application Minimum-loss Program Modulation method can be used. This kind of modulation technique is called Synchronous Optimal Modulation (SOM). In SOM pattern, switching events are determined by software in a way that reduces the harmonic content in the current, also reducing losses due to harmonic distortion in the controlled induction machine [28].

Losses in an induction machine due to presence of harmonic components of higher order than the fundamental component, is not divided evenly over the harmonic spectra. The lower harmonic components contribute more to losses than the higher harmonic components. The low harmonic components produce lower impedance resulting in higher harmonic current. The uneven distribution of loss in the harmonic spectra require appropriate performance indicator that weights the lower frequency spectra of harmonic components more heavily than the high spectra of the harmonic components [22]. Such indicator is commonly known as Weighted Total Harmonic Distortion (WTHD).

In drives applications the motor leakage inductance is limiting the harmonic currents. The weighted total harmonic distortion factor for current is then [22]:

$$THD_i = \frac{1}{\omega_1 L} \cdot \sqrt{\sum_{h=2}^{\infty} \left(\frac{\hat{U}_{a,h}}{h} \right)^2}$$

$$WTHD = \frac{THD_i}{\hat{I}_{a,1}} = \frac{\omega_1 L \cdot THD_i}{\hat{U}_{a0,1}} = \frac{1}{\hat{U}_{a0,1}} \cdot \sqrt{\sum_{h=2}^{\infty} \left(\frac{\hat{U}_{a0,h}}{h} \right)^2}$$

The SOM pulse patterns for a Three-Level NPC converter is as shown in Figure 2-17. In the figure, Instead of M , N will be used in this paper. N stands for number of pulses in a half period or number of transitions in quarter wave. Here +1 actually means Udc1 and -1 is equal –Udc2 and zero potential is the Neutral Point (NP) in the three level inverter. The total dc-link voltage Udc is equal:

$$U_{dc} = U_{dc1} + U_{dc2} \qquad U_{dc1} = U_{dc2} \Rightarrow U_{dc1} = U_{dc2} = \frac{U_{dc}}{2}$$

The Fourier series coefficients do only have sine-term and odd number of harmonics due to Half Wave Symmetry (HWS) and the Quarter Wave Symmetry (QWS). The peak value of the hth harmonic becomes [22]:

$$\hat{U}_{a0,h} = \frac{4 \cdot U_{dc1}}{h \cdot \pi} \cdot \sum_{k=1}^{N} (-1)^{k+1} \cdot \cos(h \cdot \alpha_k) \qquad 0 < \alpha_1 < \alpha_2 .... < \alpha_N < \frac{\pi}{2} \qquad (1)$$
$$h = 6 \cdot \mathrm{i} \pm 1, \quad \mathrm{i} = 1,2,3,4....$$

At this point, another modulation index called amplitude modulation index (u_{st} or m) is introduced. Amplitude Modulation index is proportional to fundamental voltage component [29]

amplitude modulation, $m\ or\ u_{st} = \frac{u}{u_{dc}}$, u is rms voltage of the modulated signal, u_{dc} is dc link voltage. The modulation index is

$$u_{st} = u_{a0,1} = \frac{4}{\pi} \cdot \sum_{k=1}^{N} (-1)^{k+1} \cdot \cos(\alpha_k) \qquad 0 < \alpha_1 < \alpha_2 \ldots\ldots\ldots < \alpha_N < \frac{\pi}{2} \tag{2}$$

Hence the objective function is to minimize the following WHTD expression

$$WTHD = \frac{1}{\hat{U}_{a0,1}} \cdot \sqrt{\sum_{h=2}^{\infty} \left(\frac{\hat{U}_{a0,h}}{h} \right)^2} = \frac{1}{\hat{U}_{a0,1}} \cdot \sqrt{\sum_{i=1}^{\infty} \left(\left(\frac{\hat{U}_{a0,6i-1}}{6i-1} \right)^2 + \left(\frac{\hat{U}_{a0,6i+1}}{6i+1} \right)^2 \right)} \tag{3}$$

The optimization task is then, for each modulation index, to find the set of N angles from α1 , …, αN of equation (2), to minimize the function of WTHD of equation (3).

Algorithm is developed by Roy Nilsen at Wartsila Norway [25] to find the required switching angles for given amplitude modulation index to minimize the total harmonic distortion. The example is shown in Figure 2-19 for N (no. of switching transitions in quarter wave)=4. Whenever the value of modulation index varies, the switching angles are changed in order to minimize the harmonic loss. The angles are only shown for quarter wave but the angles for rest of the period can easily be predicted as will be mentioned in section 2.7. Table 2-5 has tabulated the angles for five amplitude modulation index. As the modulation index change, the angles also change. Similarly there are other SOM patterns for different values of N. The example is shown for N=8 in the Appendix A. It means that for one particular value of (m,N) there is one particular type of switching signals. The harmonics as the function of modulation index is given in Figure 2-20.

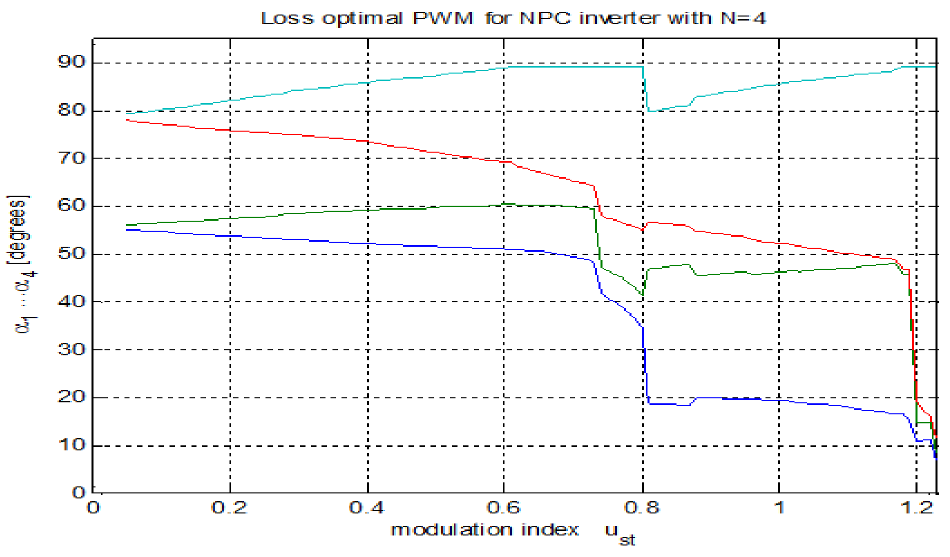


Figure 2-19 SOM pattern for N=4 [25].

Table 2-5 Switching angles for different modulation index

u_{st}	α1	α2	α3	α4
0.2	53	58	76	82
0.4	52	59	73	86
0.6	51	60	69	89
0.8	18	42	55	80
1	19	46	52	86

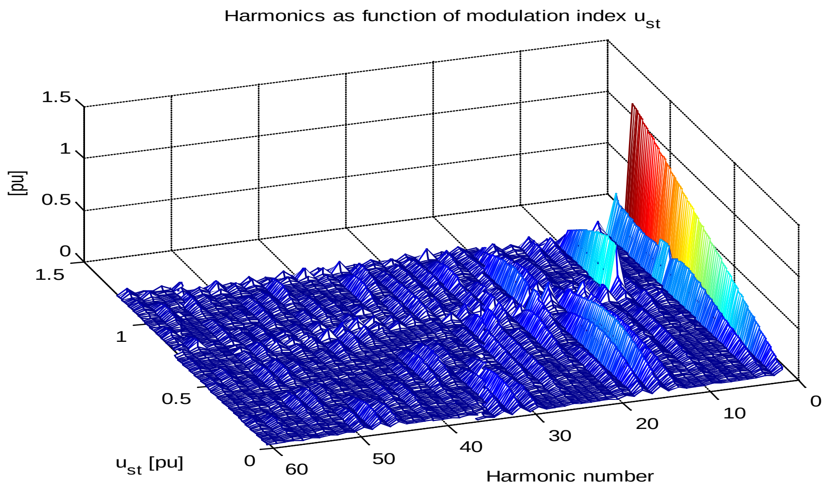


Figure 2-20 Harmonics for N=4 [25].

Using Figure 2-19, the different switching pulse can be generated for different modulation index. The switching pulse for modulation index ust(m)=0.4 for quarter wave is shown in Figure 2-21.

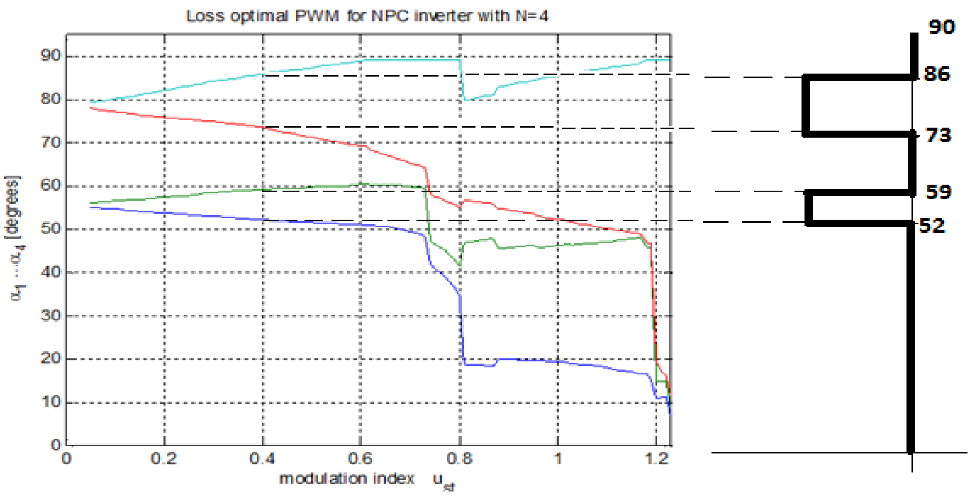


Figure 2-21 Pulse generated from SOM pattern.

2.7 Open Loop Method

When there is change in magnetic flux then voltage is induced. So there is relation

$$u = f_s\ \psi$$

Where u is the applied stator voltage, fs is stator frequency, ψ is stator flux.

$\psi = \frac{u}{f_s} = \frac{u_{dc}\ .m}{f_s}$, m is amplitude modulation, $m = \frac{u}{u_{dc}}$

After rearranging,

$f_s = \frac{u_{dc}}{\psi}\ m = k.m$

k is constant if flux ψ is kept constant. Hence if m(proportional to output voltage) change then stator frequency should also change in order to make k constant(or flux constant). Such kind of control is called u/f control. In many industries, the induction motors are controlled by variable frequency drives with the Volts/Hertz (U/f) control; this strategy intends to keep a constant flux, imposing a constant volts/hertz ratio. Whenever m has to be changed then fs is also changed for keeping flux constant.

In other to apply u/f control in Synchronous Optimal Modulation technique, there is a relation,

$f_{sw,avg} = N.f_s$=N.k.m

Here, N is number of switching angles or transitions in one quarter wave. In this, whenever m changes, fs must change to keep flux constant. However if fs change $f_{sw,avg}$ changes proportionaly but if $f_{sw,avg}$ is greater than 300Hz, then N is decreased as shown in Figure 2-18.

Synchronous Optimal Modulation pre calculates switching angles for particular value of (m,N) from algorithm as said in section 2.6 and store in a memory table pattern as P(m, N). Hence for the set of one m and N, there is set of switching angles stored in memory table which generate the pulse. For instance, for (m=0.4, N=4), the switching patterns for quarter wave is shown in Figure 2-21. The switching angles calculated in this way, is only for quarter wave. However, it is easy to predict the angle for rest of the period. Prediction is based on the fact that the signal has quarter wave symmetry and half wave symmetry. It uses the following rule:

In interval I, $0 \leq \alpha \leq \pi/2$: uss(α) = f {P(m,N)}

The remaining portions of a full fundamental cycle are determined using the conditions for quarter-wave symmetry and half wave symmetry.

In interval II, π≤ α ≤ π : uss(α) = uss(π − α)

According to half-wave symmetry

In interval III, π≤ α ≤ 2π : uss (α) = uss (2π − α) [29]

After using this prediction to Figure 2-21 , the switching pulse for one fundamental period is shown in Figure 2-22.

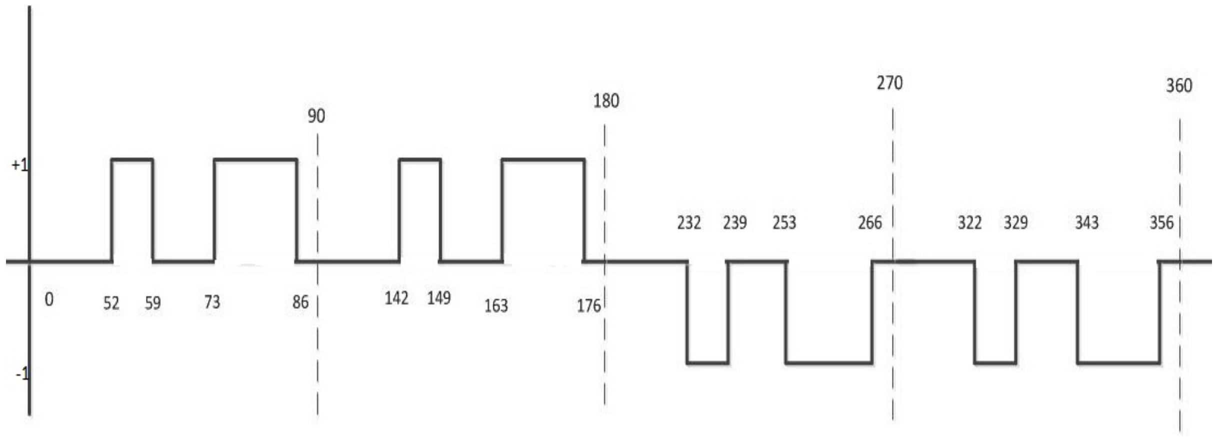


Figure 2-22 Switching Pulse in one fundamental frequency.

2.8 Synchronous Optimal Modulation signal flow graph

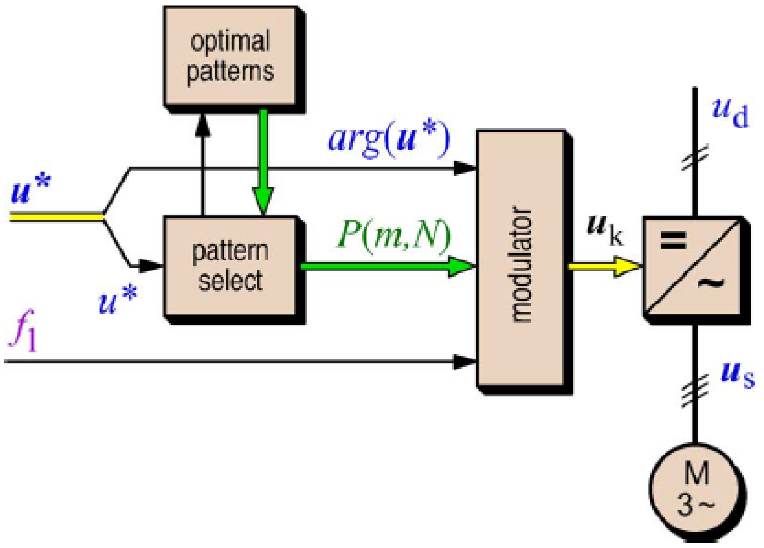


Figure 2-23 Synchronous Optimal Modulation signal flow graph [29].

The Figure 2-23 shows the signal flow graph of the project for open loop system. The input to the system is the fundamental frequency and reference voltage vector u*. The modulation index m is proportional to the magnitude of the reference voltage vector u*. There is relation

$$f_{sw,avg} = N.f_s$$

The value of N can be calculated using the graph Figure 2-18 for keeping the maximum switching frequency of IGBT as 300Hz. And m depends on u* and N depends on fundamental frequency. Once m and N are calculated, for patterns as P(m,N), the optimal switching angles can be calculated as described in section 2.6. The patterns are functions of the modulation index m and the pulse number N. A parameter pair (m, N) selects the corresponding pulse pattern from pattern selector. This pattern is fed to the modulator. The modulator converts the switching angles αi defined by P(m, N) to switching times. The fundamental frequency signal f1 translates the resulting switching angles αi to switching time ti = αi /(2πf1). The modulator thus creates the sequence of switching state at various switching time needed for each IGBTs of inverter to obtain desired synchronous optimal modulation pattern as prescribed by the optimal pattern P(m, N). The modulator is designed in FPGA. The FPGA control card is programmed accordingly for firmware implementation of synchronous optimal modulation. In the chapter 3 below, there is detail explanation of application of FPGA in this project.

Chapter 3

3 Introduction to FPGA and Xilinx EDK

The implementation of three level modulator is done on extremely fast digital device called Field Programmable Gate Array (FPGA). The important portion of the project commence from chapter 3 which introduces FPGA architecture, FPGA control card developed by SINTEF along with the description of tool called Xilinx which is used to program FPGA.

3.1 Background

The reliability and performance of digital technologies have been improving consistently. With this growth, digital control has replaced analog counterpart because they are flexible to modify the control schemes, they can adapt to different systems and operating conditions. Other benefits of digital control system are immunity to noise and insensitivity to component variation. These digital control systems are implied on microcontrollers or digital signal processors (DSPs) due to their software flexibility and low cost. These components have arithmetic logic, analog to digital converters, timers for solving real time operation. Nevertheless, some of the benefits of analog control like accuracy, absence of feedback loop delays etc are very difficult to be replaced by these microcontrollers and DSPs. However high performance DSP can provide such benefits but they are limited to complex algorithm. The cost of such DSPs can exceed the benefits they bring [30].

After the development of field-programmable gate array (FPGA), the gap between the analog and digital world is reduced. FPGA has exceptionally fast computation capability hence complex control algorithm can be computed in few microseconds and they are cost efficient as well. They are adapted in electrical drive application. They are successfully applied in control of pulse width-modulation (PWM) converters, machine drives and multiphase machine control system [30].

3.2 General Architecture of FPGA

"A Field Programmable Gate Array (FPGA) is a microchip. FPGA consists of matrix of configurable logic blocks (CLB) made up of flip flops and lookup tables, which can be used along with the configurable input output blocks and interconnection programmable network to make a programmable hardware circuit known as Intellectual Property (IP)"[31]. IP are meant to perform certain time critical task. One of the important specification of FPGA is matrix size, the bigger it is, more IPs it can have. The structures of CLB include two, four, or more logic cells, also called logic elements, which are the basic grains of the FPGA [31]. It is illustrated in Figure 3-3.

Even by changing the configuration of the switch matrix, the functionality of FPGA can be modified hence it is a flexible chip. Such flexibility helps the user to create their own hardware design according to their specification. This obviously contrasts the conventional micro controller where pin configuration are fixed during manufacture [32].

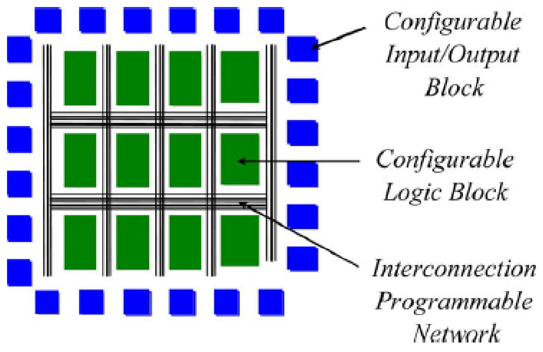

Configurable Input/Output Block
Configurable Logic Block
Interconnection Programmable Network

Figure 3-1 General architecture of FPGA [31].

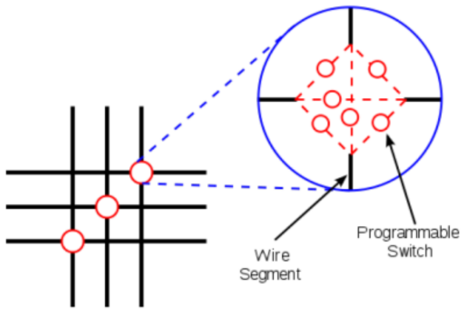


Figure 3-2 Interconnection programmable network.

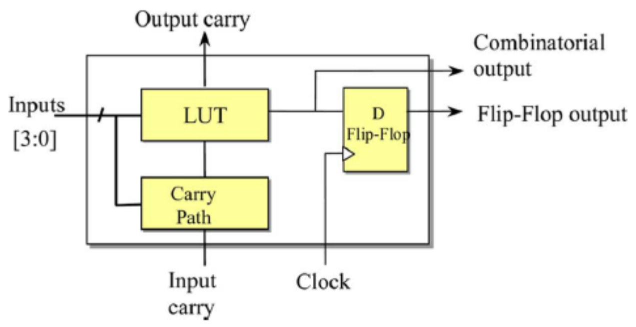


Figure 3-3 Inside logic block [31].

Even though FPGA has such immense benefits, the design process is extremely complex. All the hardware components have to be designed by users themselves, so the amount of programming is huge. For designing such chips, special hardware description language called VHDL or Verilog has to be applied. In this thesis VHDL is used. This HDL is different from other programming language. It is because the statements in HDL create digital circuitry which performs operation in parallel. Other programming languages like C create processor instructions which are executed sequential. FPGA really is a digital circuit. The syntax is also different from C [32]. VHDL stands for VHSIC hardware Description Language. VHSIC is itself an abbreviation for Very High Speed Integrated Circuits. These HDL are executed parallel on contrary to other computer program which are executed sequentially [33].

3.3 Embedded System Development

FPGA is operated in parallel so it is dramatically fast and can be used for time critical application. The whole control system for a motor drive converter could be placed in an FPGA in order to achieve high speed operations. However this would demand a very large FPGA with complicated design structure. In addition, every function of the whole system are not time critical. Electric Drive control system generally has slow outer loop and fast inner loop. Moreover the speed/power/frequency control of a converter is slower process than modulation process. Hence there is no point of implementing such relatively slower process on FPGA. The cost and complexity of using large FPGA would be extremely high if whole control system is implemented [32].

Advanced FPGA are manufactured with processor inside it. This kind of system is called embedded system. This is the good solution to deal problems described in above paragraph.

Embedded FPGA helps to split the design procedure in two: hardware part (for FPGA) and software part (processor). This makes it possible to take advantage of both sector but still use one single chip. The advantage of FPGA is exceptional speed gain and that of processor is the implementation of simple and versatile software programming [32]. The processes which are relatively slow and which is seeking complex design are programmed in software. And faster processes are programmed in hardware.

The concept of System on Chip (SOC) is based on integration of a variety of features combining digital and analog part, hardware and software, equipped with a communication infrastructure. As system on-chip architectures continue to receive more and more attention from the embedded systems community, FPGA manufacturers such as Xilinx are responding with a new generation of FPGA architectures that contain a variety of embedded resources. One of new generation FPGA is Vertex-5 ppc440 FX30T which has Embedded PowerPC440 Processor Core. This type of Soc based on platform FPGA is device which integrates field programmable logic cells with predetermined collection of resources such as embedded CPUs, SRAM, versatile general purpose IO ports, high speed, serial links, various standard peripherals and others [34].

3.4 Xilinx Board

Xilinx Board used for this project is Xilinx Virtex FX30T which is built by SINTEF to suit the various requirements for different power electronic control applications. In this Xilinx board, the Embedded Development Kit (EDK) tools can be used to design a full featured embedded system consisting of hardware and software. The control card outline with the chip and other important peripherals is as shown in the Figure 3-4.The important components of the control cards apart from FPGA are mentioned below [35], [3].

3.4.1 Communication Port

The card is having RS232 serial port connected with a male 9 pol D-sub connector. The serial port is not isolated.

Ethernet port for 10MB/sec and 100 MB/sec connection is available on board. The port is based on the MAC block which is embedded in the Virtex 5 chip. The standard physical interface is located on the separate chip (National Semiconductor DP83848) outside FPGA on the board. The board is also having the high speed V.2 USB port with the separate controller NXP ISP1582 available on board outside FPGA.

CAN controller IC, Microchip MCP2515, is available on card for the with the other control cards

3.4.2 memory

Card is having ample amount of memory as summarized below.

• DDR2 DRAM -Micron MT74H64M16HR-3E in 1GB, 16x64Mbit 333MHz clock frequency.

• EROM-Renesas HN58V257A. 32k x 8 bits.

• FLASH-Spansion S29GL512P is 512 Mbit flash chip, 32Mx16 64Mx8.

3.4.3 clock

The FPGA is supplied with the 40 MHz clock on the card. FPGA generates different clock frequencies inside which are used by various components are as listed below.

• AD-converter is operated with a differential 40 MHz clock.

• AD converter generates a 240 MHz clock. This drives the transfer of data from the AD

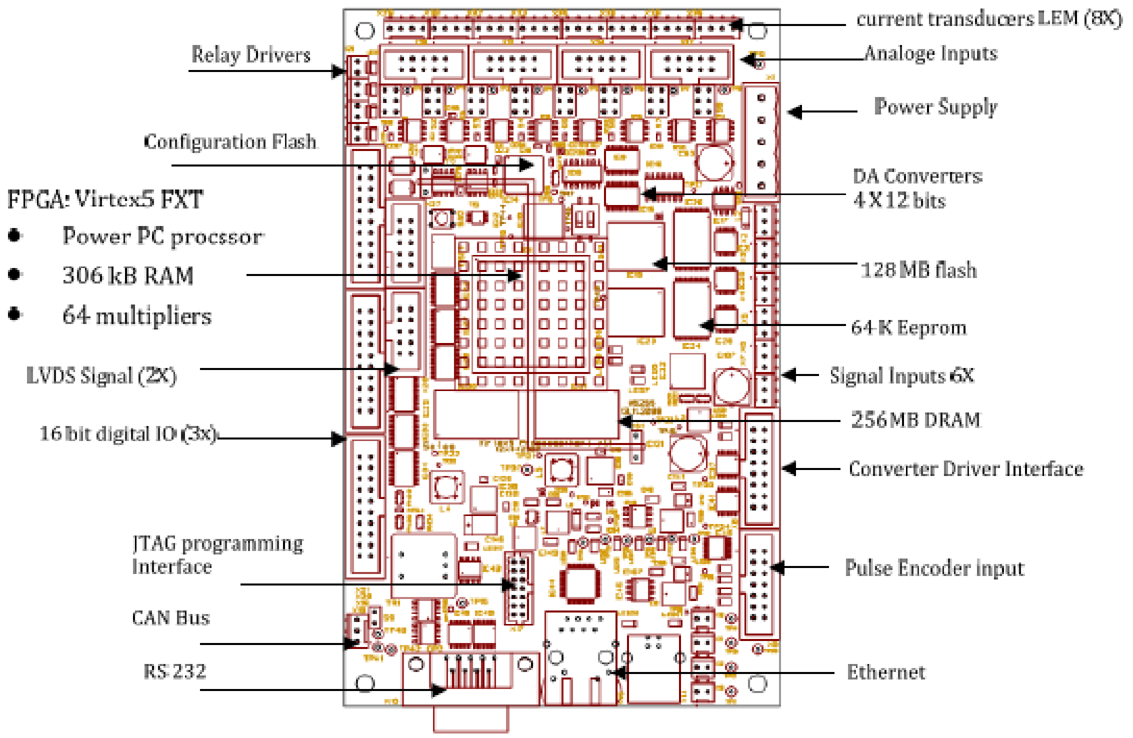


Figure 3-4 FPGA control card[35].

3.4.4 Converter

• USB controller is powered with a 12 MHz clock.

• CAN controller is driven by a clock of 20 MHZ.

• Ethernet PHY circuit is run a 25 MHZ clock.

• The RISC processor is powered by a 300 MHz (PowerPC) or 100 MHz (Micro Blaze) clock.

• Processor bus powered by a 100 MHz clock.

• DDR2 DRAM chips driven by a differential clock of 200 MHz.

• Transport correction (IDELAYCTRL) for DDR2 DRAM is run by a 200MHz clock.

3.4.5 Pulse Encoder:

Card has input for connecting a two-phase encoder with a reference signal. The signals are adapted to a Heidenhain ROD 420 pulse encoder, which has balanced signals with RS422 signal levels. Pin numbering is the same pin numbers on the Round 12 pole connector ROD420. Receiver circuit MAX3097 detects signal on lines. Three green LEDs at the connector can be used to indicate the signal levels from the pulse sensor, while a red LED can be used to indicate errors.

3.4.6 Relay drivers.

Card is equipped with four relay driver. The output voltage is 5V. This is sufficient to run many types of relays with coil voltages 6V. Small cooling fans can be operated with this. An LED lluminates when a relay driver is turned on.

3.4.7 LVDS:

The card is having contacts for high speed serial communication with LVDS signals having 600 Mbit/sec. transfer rate. LVDS transmission line has 100 Ω differential line impedance. The line pair is terminated on FPGA with 100 Ω resistance between lines.

3.4.8 Signal Inputs:

The card is having six general purpose signal inputs either analog or digital .The analog signals are read through the voltage divider network and with low pass filter with 30 µ sec time constant signal level for the analog signal is 0-5 V. Digital signals are inputted through the low pass filter with 1 µ sec time constant, the signal level being 5 V CMOS.

3.4.9 AD Converter:

The card is equipped with an Analog Devices AD9222, 8-channel 12-bit AD converter. Each channel has its own AD converter that works with 40 MSPS. The channels thus sampled synchronously. Data is transferred to the FPGA in a serial form, the baud rate is 12BIT x 40 MSPS = 480 Mbit/ sec rates.

3.4.10 Digital Input Output port:

There are three channels on card with 16 bit digital IO ports. The signal level is the 0-3V. This can drive 5V TTL inputs; however, as the FPGA block is fed with the 3 volts so voltage is clipped before entering into FPGA.

3.5 Xilinx Embedded Development Kit

3.5.1 Introduction

Xilinx FPGAs provide customizable silicon on top of which different kinds of hardware can be created. The Xilinx Embedded Development Kit (EDK) provides tools to create custom embedded hardware on Xilinx FPGAs. EDK is a series of software products developed to extend the Xilinx Integration Software Environment (ISE) into the realm of system level design. The Embedded Development Kit, EDK, produced by Xilinx, provides an entry point for both embedded software and hardware designers who want to design with the PowerPC processors embedded into the Vertex-5 ppc440 FX30T[36].

The programming environment developed by Xilinx is called Xilinx Integrated Software Environment (Xilinx ISE). It consists of many different programs, but three of them are more important for embedded system.

i) ISE Project Navigator
ii) Xilinx Platform Studio(XPS)
iii) Software Development Kit. (SDK).

The design environment employed for this thesis is shown in Figure 3-5

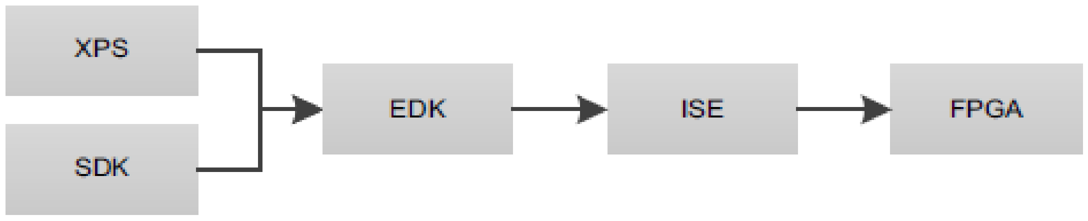


Figure 3-5 FPGA design flow adopted in this work.

XPS and SDK together are known as Xilinx Embedded Development Kit. When working with the design of an embedded system built around an FPGA-chip, four different abstraction levels to work on can be assumed. They are

i) Physical description level
ii) Behavior Level
iii) Module Interaction level
iv) Software Level

3.5.2 Different Abstraction Level

Each program (ISE Project Navigator, XPS, SDK) is working on one specific level of the abstraction layers. Below is given brief introduction of the abstraction level with Xilinx program used in each level. This section is taken from ref [32].

i) Physical description level

The lowest level is normally the result of an automatic process performed in the programming suite, i.e. the synthesis. The synthesis tool takes the description created during the programming phase (written in a hardware description language), and translates it to a list with the signal routing and placement of the different ports and logic circuits.

ii) Behaviour Level

In behaviour level digital circuits are described by writing codes in hardware descriptive language, VHDL. ISE Project Navigator is utilized when working on the behavioral level. It is typically utilized when developing new sub modules, but also entire FPGA-configurations can be created here. The designer needs to specify the input and output ports for the circuit, the signal routing and the tasks, the circuit is supposed to perform. Since the circuit is a physical circuit, all actions which are described by the designer are performed in parallel. A module is a digital circuit designed to perform a specific task within a larger system.

iii) Module Interaction level

In behavior level, new programmable hardware circuit known as Intellectual Property (IP) are written using VHDL. Drive control applications seek complex algorithm structure. In order to make the design of control algorithms more manageable and less intuitive, reusability of already made IP can be done. These kinds of interaction of different modules are done in this level for complex system [31]. The Xilinx Platform Studio (XPS) program is especially dedicated for this level.

This program is utilized when synthesizing an FPGA-system from existing IP-modules (either from a local library, or from the included Xilinx blocks) and connecting signals between these. The platform is built around the PowerPC-processor and its processor bus. In addition to the information on the processor, XPS also needs a User Constraint File (UCF), which contains the description of all the input and output ports of the FPGA. The different IP-blocks are added to the design from the IP-catalogue. Connections to the I/O-ports, the processor bus or other FPGA-blocks are also created here. The communication between the processor and the hardware modules is performed by a processor bus. The output of XPS is a hardware platform which can be exported to the Software level to be discussed below. Such a platform contains information about the FPGA configuration, necessary driver files and address specifications.

iv) Software Level

From the earlier section, it is clear that FPGA programming is done in two part: hardware part and Software part. Hardware part is considered in behaviour level and module interaction level while software part is considered in this level of abstraction. Xilinx Software Development Kit (SDK) program is used here. The output of XPS is a hardware platform which can be exported to the Software Development Kit. The interaction with the hardware is through a processor bus, which transfers input data to the processor, and takes the output back to the FPGA. At this level, programming the FPGA is done in either C or C++.

3.5.3 Xilinx ISE overview

Xilinx ISE system is an integrated design environment that consists of a set of programs to create (capture), simulate and implement digital designs in a FPGA target device [37]. It is used in first two abstraction level of design in Physical Description Level and Behaviour Level. The design flow is shown in Figure 3-6. These steps are involved in the realization of a digital system using Xilinx FPGA.

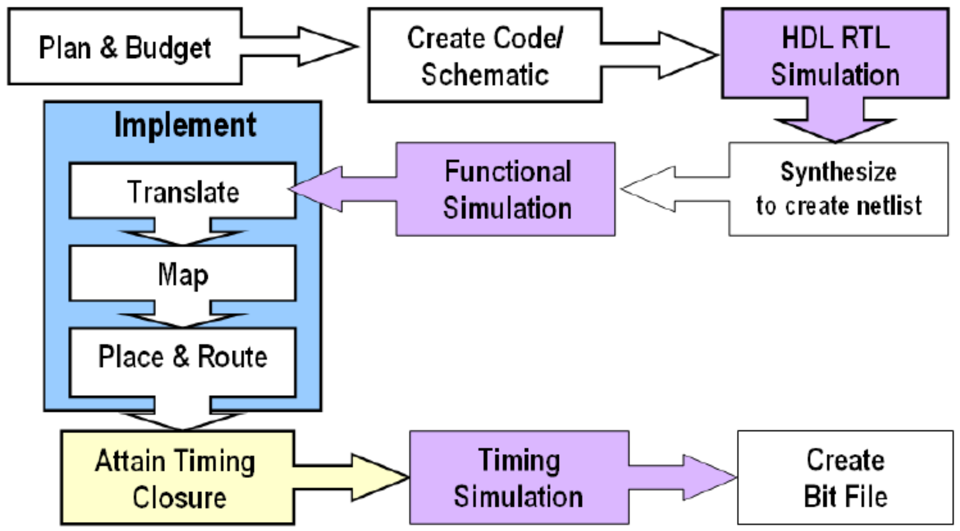


Figure 3-6 Overview of various steps in design flow of digital system.

Design Entry

The first step is to enter the design. This can be done by creating "Source" files. Source files can be created in Hardware Description Language (HDL) such as VHDL, Verilog.

Synthesis

• Breaks down the VHDL design into logic elements.

• Recognizes common elements as counters, multiplexers.

• Generates a logic circuit of the design.

Implementation

• Builds FPGA circuitry representing the logic elements

• Places the logic elements onto the FPGA structure

• Routes connections between the elements

• Calculates signal delay through connections and logic.

• Rearranges the layout in order to improve timing.

• Result: A BIT file, containing binary configuration data for the FPGA

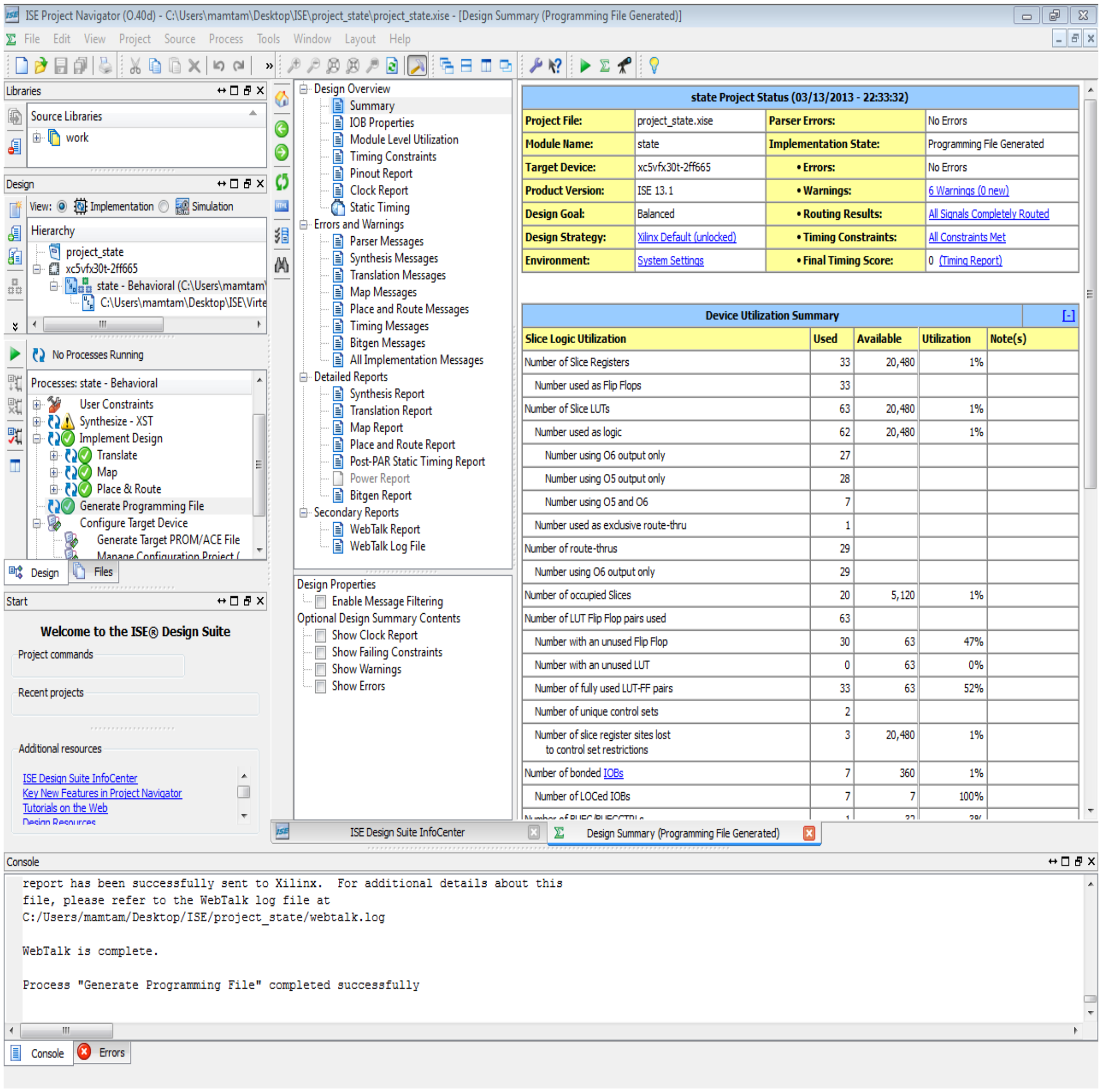
Downloading

• Straight to the FPGA, for test and debug.

• To configuration flash memory. Loaded into FPGA at power up.

Project Navigator Window

The above steps are managed through a central ISE Project Navigator window, shown below.



state Project Status (03/13/2013 - 22:33:32)			
Project File:	project_state.xise	**Parser Errors:**	No Errors
Module Name:	state	**Implementation State:**	Programming File Generated
Target Device:	xc5vfx30t-2ff665	**• Errors:**	No Errors
Product Version:	ISE 13.1	**• Warnings:**	6 Warnings (0 new)
Design Goal:	Balanced	**• Routing Results:**	All Signals Completely Routed
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	All Constraints Met
Environment:	System Settings	**• Final Timing Score:**	0 (Timing Report)

Device Utilization Summary [-]

Slice Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Registers	33	20,480	1%	
Number used as Flip Flops	33			
Number of Slice LUTs	63	20,480	1%	
Number used as logic	62	20,480	1%	
Number using O6 output only	27			
Number using O5 output only	28			
Number using O5 and O6	7			
Number used as exclusive route-thru	1			
Number of route-thrus	29			
Number using O6 output only	29			
Number of occupied Slices	20	5,120	1%	
Number of LUT Flip Flop pairs used	63			
Number with an unused Flip Flop	30	63	47%	
Number with an unused LUT	0	63	0%	
Number of fully used LUT-FF pairs	33	63	52%	
Number of unique control sets	2			
Number of slice register sites lost to control set restrictions	3	20,480	1%	
Number of bonded IOBs	7	360	1%	
Number of LOCed IOBs	7	7	100%	

```
report has been successfully sent to Xilinx.  For additional details about this
file, please refer to the WebTalk log file at
C:/Users/mamtam/Desktop/ISE/project_state/webtalk.log

WebTalk is complete.

Process "Generate Programming File" completed successfully
```

Figure 3-7 ISE windows.

Hierarchy Window

This window contains the design source files for a project. These are the source files that is created or added to the project. It also consist of the UCF (User Constraint File) which contains the description of all the input and output ports of the FPGA.

Processes Window

The processes windows list the available processes (corresponding to the process selected in the processes window). Typically a particular process to perform on the selected source file is available here. This can include a simulation, implementation, etc. To run a process one need to double click on the process. When a process has been successfully executed a green tick-off icon appears. When a high-level process is clicked, the Project Navigator will automatically run all the associated lower-level processes.

3.5.4 Xilinx platform studio overview

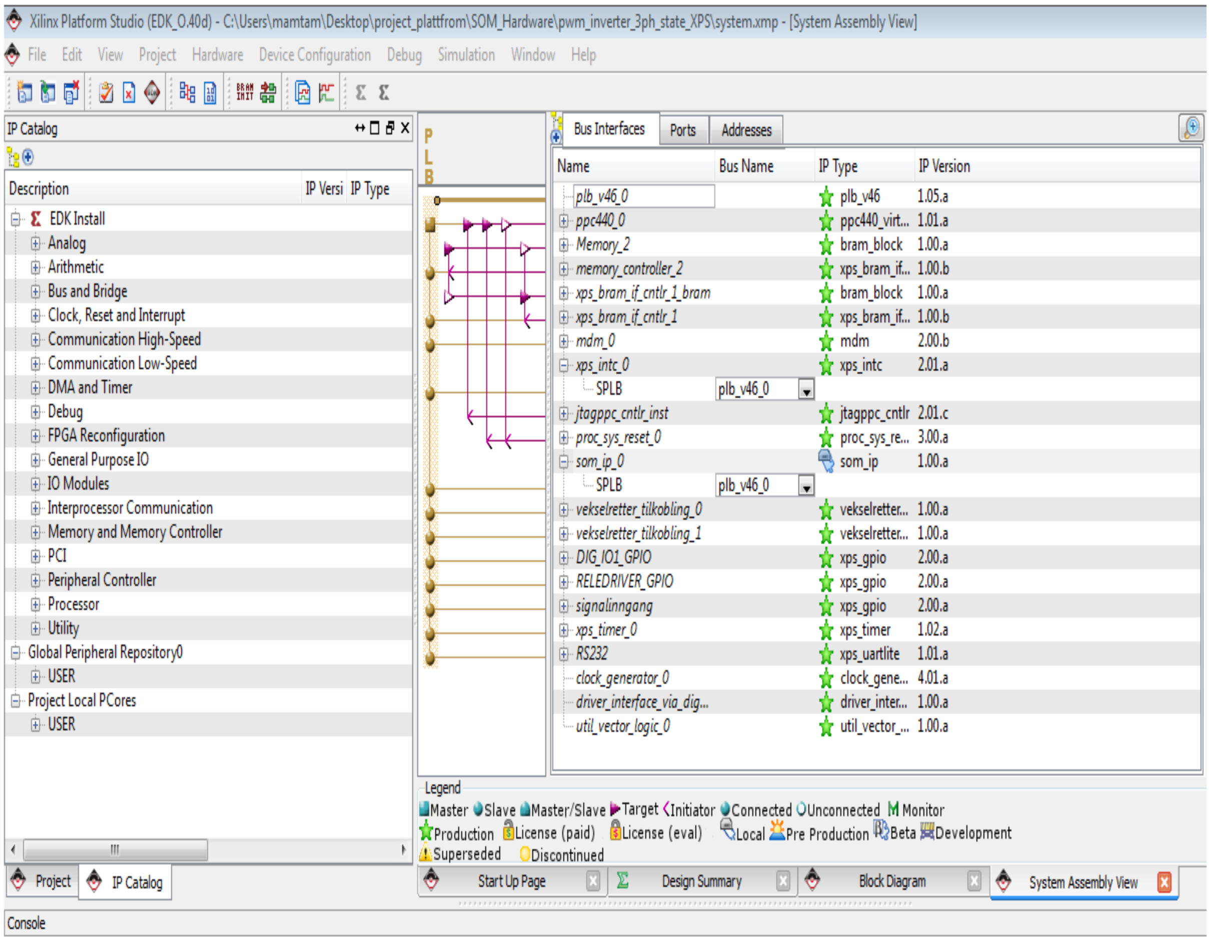
XPS is used in module interaction level. Intellectual Properties (IPs) are made by users and can be reused by other users in order to make the design of control algorithms more manageable and less intuitive. Interaction of various modules is done in this platform. Some of the basic modules like Processor, timer, memory block are already provided by Xilinx. Other modules like Digital to Analog conversion, Inverter, Driver Interface which are used in this project are created by Kjell Ljøkelsøy from SINTEF [35]. User IPs which are convenient to this project are also created and added to the processor bus along with all other modules. The window of XPS is shown in Figure 3-8.

The IP catalog window consists of library of available IPs. Some IPs are directly available from Xilinx, some IP modules are taken from Kjell Ljøkelsøy from SINTEF. While some of the IP are user IPs.

The Bus Interface window consists of all the IP modules necessary for this project. Along that window there is graphical connection showing the connections of the modules to the PLB (Processor Local Bus).

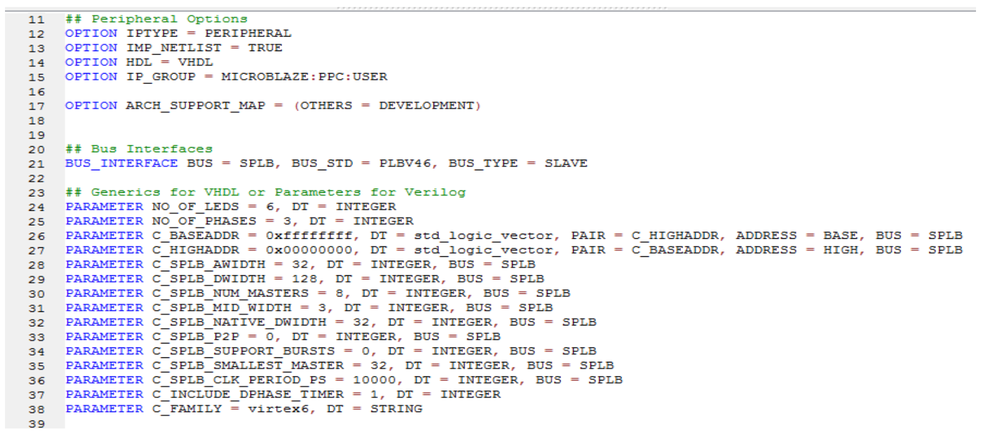
There is also Port window where different IPs are connected through the common signal name. There is address assigned to every address. The address is available in the Address window.

The VHDL description of an IP is not only enough to interface with the PLB hence an intermediate representation layer, the Microprocessor Peripheral Description (MPD) file are also needed which looks as shown in Figure 3-9. The MPD file contains basic information of underlying IP VHDL/Verilog implementation (generics, ports), adding flow dependent attributes, used for configuration.


Xilinx Platform Studio (EDK_O.40d) - C:\Users\mamtam\Desktop\project_plattfrom\SOM_Hardware\pwm_inverter_3ph_state_XPS\system.xmp - [System Assembly View]
File Edit View Project Hardware Device Configuration Debug Simulation Window Help
IP Catalog
Description
IP Versi
IP Type
EDK Install
Analog
Arithmetic
Bus and Bridge
Clock, Reset and Interrupt
Communication High-Speed
Communication Low-Speed
DMA and Timer
Debug
FPGA Reconfiguration
General Purpose IO
IO Modules
Interprocessor Communication
Memory and Memory Controller
PCI
Peripheral Controller
Processor
Utility
Global Peripheral Repository0
USER
Project Local PCores
USER
PLB
Bus Interfaces
Ports
Addresses
Legend
Master Slave Master/Slave Target Initiator Connected Unconnected Monitor
Production License (paid) License (eval) Local Pre Production Beta Development
Superseded Discontinued
Project
IP Catalog
Start Up Page
Design Summary
Block Diagram
System Assembly View
Console

Name	Bus Name	IP Type	IP Version
plb_v46_0		plb_v46	1.05.a
ppc440_0		ppc440_virt...	1.01.a
Memory_2		bram_block	1.00.a
memory_controller_2		xps_bram_if...	1.00.b
xps_bram_if_cntlr_1_bram		bram_block	1.00.a
xps_bram_if_cntlr_1		xps_bram_if...	1.00.b
mdm_0		mdm	2.00.b
xps_intc_0		xps_intc	2.01.a
SPLB	plb_v46_0		
jtagppc_cntlr_inst		jtagppc_cntlr	2.01.c
proc_sys_reset_0		proc_sys_re...	3.00.a
som_ip_0		som_ip	1.00.a
SPLB	plb_v46_0		
vekselretter_tilkobling_0		vekselretter...	1.00.a
vekselretter_tilkobling_1		vekselretter...	1.00.a
DIG_IO1_GPIO		xps_gpio	2.00.a
RELEDRIVER_GPIO		xps_gpio	2.00.a
signalinngang		xps_gpio	2.00.a
xps_timer_0		xps_timer	1.02.a
RS232		xps_uartlite	1.01.a
clock_generator_0		clock_gene...	4.01.a
driver_interface_via_dig...		driver_inter...	1.00.a
util_vector_logic_0		util_vector_...	1.00.a

Figure 3-8 XPS windows.



```
## Peripheral Options
OPTION IPTYPE = PERIPHERAL
OPTION IMP_NETLIST = TRUE
OPTION HDL = VHDL
OPTION IP_GROUP = MICROBLAZE:PPC:USER

OPTION ARCH_SUPPORT_MAP = (OTHERS = DEVELOPMENT)


## Bus Interfaces
BUS_INTERFACE BUS = SPLB, BUS_STD = PLBV46, BUS_TYPE = SLAVE

## Generics for VHDL or Parameters for Verilog
PARAMETER NO_OF_LEDS = 6, DT = INTEGER
PARAMETER NO_OF_PHASES = 3, DT = INTEGER
PARAMETER C_BASEADDR = 0xffffffff, DT = std_logic_vector, PAIR = C_HIGHADDR, ADDRESS = BASE, BUS = SPLB
PARAMETER C_HIGHADDR = 0x00000000, DT = std_logic_vector, PAIR = C_BASEADDR, ADDRESS = HIGH, BUS = SPLB
PARAMETER C_SPLB_AWIDTH = 32, DT = INTEGER, BUS = SPLB
PARAMETER C_SPLB_DWIDTH = 128, DT = INTEGER, BUS = SPLB
PARAMETER C_SPLB_NUM_MASTERS = 8, DT = INTEGER, BUS = SPLB
PARAMETER C_SPLB_MID_WIDTH = 3, DT = INTEGER, BUS = SPLB
PARAMETER C_SPLB_NATIVE_DWIDTH = 32, DT = INTEGER, BUS = SPLB
PARAMETER C_SPLB_P2P = 0, DT = INTEGER, BUS = SPLB
PARAMETER C_SPLB_SUPPORT_BURSTS = 0, DT = INTEGER, BUS = SPLB
PARAMETER C_SPLB_SMALLEST_MASTER = 32, DT = INTEGER, BUS = SPLB
PARAMETER C_SPLB_CLK_PERIOD_PS = 10000, DT = INTEGER, BUS = SPLB
PARAMETER C_INCLUDE_DPHASE_TIMER = 1, DT = INTEGER
PARAMETER C_FAMILY = virtex6, DT = STRING

```

Figure 3-9 Microprocessor Peripheral Description (MPD) file.

The IP implementations abstracted by the MPD files need to be parameterized at a higher level; this is done through the components instantiation in the Microprocessor Hardware Specification (MHS) file. As shown in Figure 3-10, Platgen (a Xilinx tool) reads a MHS as its primary design input. The tool also reads various hardware Microprocessor Peripheral Description (MPD) files from the EDK library. Platgen produces the top-level HDL design file for the embedded system that stitches together all the instances of parameterized IP blocks contained in the system. In the process, it resolves all the high-level bus connections in the MHS into the actual signals required to interconnect the processors, peripherals and on-chip memories. The EDK intermediate description, based in the MHS and MPD file (among others), represents an improvement over a purely VHDL description[38].

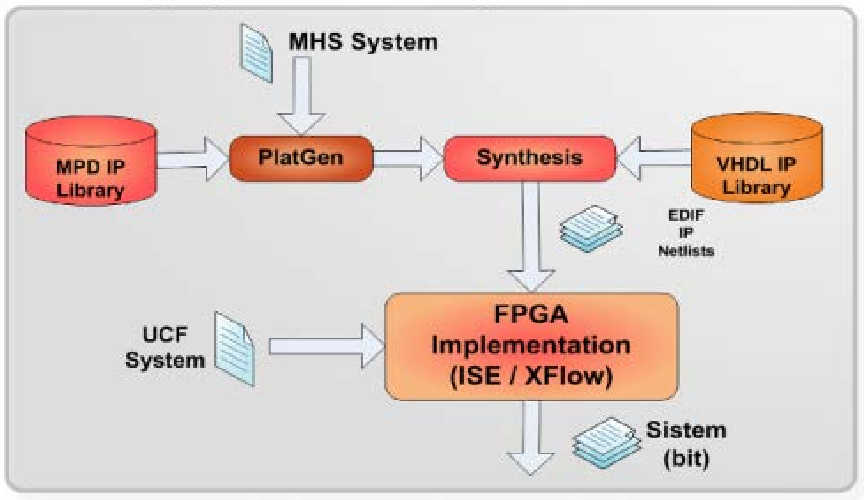


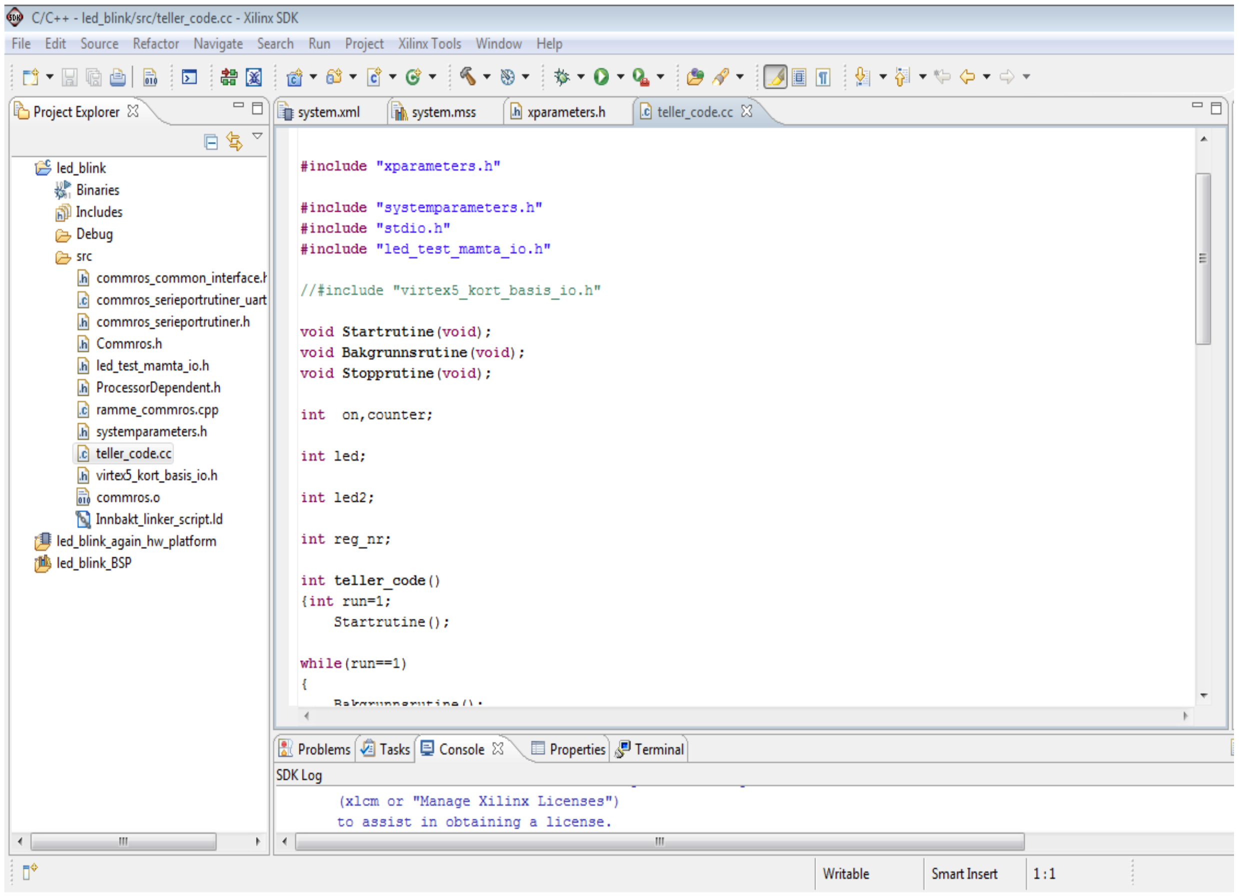
Figure 3-10 Xilinx EDK flow for processor based design.

3.5.5 Xilinx Software Development Kit overview

The Xilinx Software Development Kit (SDK) provides an environment for creating software platforms and applications targeted for Xilinx embedded processors. SDK works with hardware designs created with the Xilinx Platform Studio (XPS) embedded development tools[39].

The hardware specification captures all necessary information and files from a Xilinx Embedded Development Kit (EDK) hardware design that are required for a software developer to develop, debug, and deploy software applications for that hardware [39]. Typically, a hardware designer who develops hardware using Xilinx Platform Studio (XPS) exports this specification file to a directory. The software developer then imports this file using the Xilinx Software Development

Kit (SDK). SDK is based on the Eclipse open source standard. SDK features include feature-rich C/C++ code editor and compilation environment. The SDK windows is shown in Figure 3-11



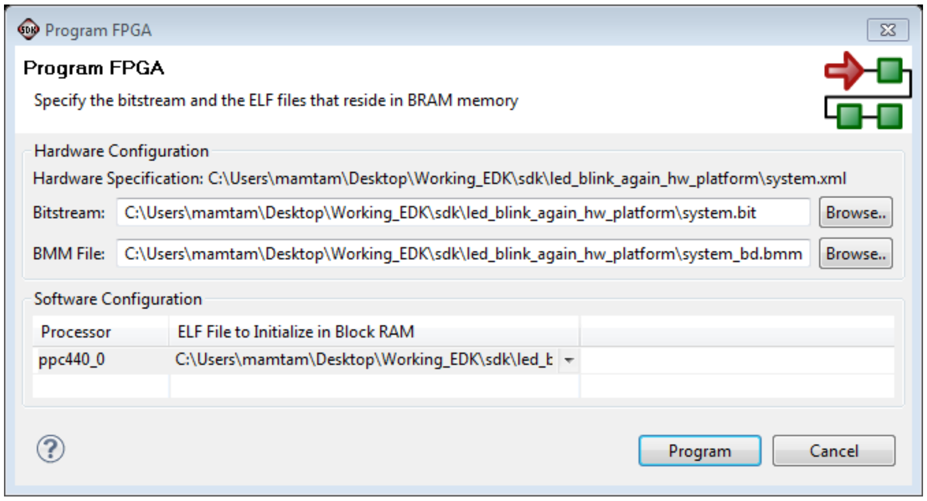
```
#include "xparameters.h"

#include "systemparameters.h"
#include "stdio.h"
#include "led_test_mamta_io.h"

//#include "virtex5_kort_basis_io.h"

void Startrutine(void);
void Bakgrunnsrutine(void);
void Stopprutine(void);

int  on,counter;

int led;

int led2;

int reg_nr;

int teller_code()
{int run=1;
    Startrutine();

while(run==1)
{
    Bakgrunnsrutine();
```

```
       (xlcm or "Manage Xilinx Licenses")
       to assist in obtaining a license.
```

Figure 3-11 SDK windows.

The Project Explorer window consists of hardware platform (filename_hw_platform) which is exported from Xilinx Platform studio. This window also consists of Board Support Package file (filename_BSP). In embedded system, BSP is implementation specific support software for the specific hardware. In this case the specific hardware is the one which is developed in XPS and exported to the SDK. A board support package is software that implements and supports an operating system on hardware like a development board. Usually built with a bootloader, a BSP contains the minimal device support to load the OS and device drivers for all the devices on the board[40]. for example it consists of address of all the modules in the hardware.

On this hardware the software application is developed. The codes are written in C++ format. Using the features of C++, complex software needed for the project is developed here. The output of the software is placed into the Processor local Bus which is common between SDK and XPS. Data placed in PLB is retrieved by XPS and implement it into FPGA. The Figure 3-12 shows the way, how FPGA is programmed from SDK.


Program FPGA

Program FPGA

Specify the bitstream and the ELF files that reside in BRAM memory

Hardware Configuration

Hardware Specification: C:\Users\mamtam\Desktop\Working_EDK\sdk\led_blink_again_hw_platform\system.xml

Bitstream: C:\Users\mamtam\Desktop\Working_EDK\sdk\led_blink_again_hw_platform\system.bit

BMM File: C:\Users\mamtam\Desktop\Working_EDK\sdk\led_blink_again_hw_platform\system_bd.bmm

Software Configuration

Processor	ELF File to Initialize in Block RAM
ppc440_0	C:\Users\mamtam\Desktop\Working_EDK\sdk\led_b

Figure 3-12 Program FPGA.

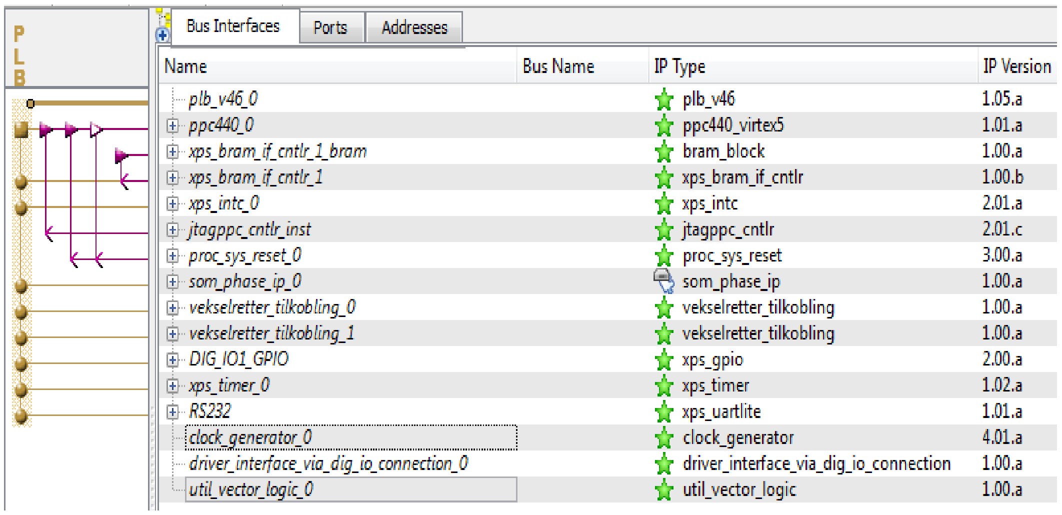
Once *Program FPGA* command is given in SDK, the software level put the data in the PLB and the command is given to the hardware on XPS to operate. Then synthesis, implementation occurs and bitfile is downloaded to the real FPGA and hence FPGA functions as the commanded by the software. In this way embedded system works.

Chapter 4

The methodology applied in this project for implementing SOM by using FPGA is presented in chapter 4 which contains the description of Intelligent Properties (IP) and the connection between various IPs used in this project.

4 Firmware Implementation of SOM in FPGA

Intellectual Properties (IPs) are key building blocks of Xilinx Targeted Design Platforms [41]. IP performs one particular task. IPs are programmable hardware circuits. New Intellectual Properties(IPs) are written using VHDL. Intellectual Properties (IPs) are made by users and can be reused by other users in order to make the design of control algorithms more manageable and less intuitive. The interaction of different IPs is done in Module Interaction level in one of the software package from Xilinx EDK called Xilinx Platform Studio. The IPs which are used in this thesis are named in Figure 4-1. Some of the IPs are som_phase_ip, DIG_IO1_GPIO, vekselretter_tikobling etc . MHS file of the whole project is presented in Appendix B



Name	Bus Name	IP Type	IP Version
plb_v46_0		plb_v46	1.05.a
ppc440_0		ppc440_virtex5	1.01.a
xps_bram_if_cntlr_1_bram		bram_block	1.00.a
xps_bram_if_cntlr_1		xps_bram_if_cntlr	1.00.b
xps_intc_0		xps_intc	2.01.a
jtagppc_cntlr_inst		jtagppc_cntlr	2.01.c
proc_sys_reset_0		proc_sys_reset	3.00.a
som_phase_ip_0		som_phase_ip	1.00.a
vekselretter_tilkobling_0		vekselretter_tilkobling	1.00.a
vekselretter_tilkobling_1		vekselretter_tilkobling	1.00.a
DIG_IO1_GPIO		xps_gpio	2.00.a
xps_timer_0		xps_timer	1.02.a
RS232		xps_uartlite	1.01.a
clock_generator_0		clock_generator	4.01.a
driver_interface_via_dig_io_connection_0		driver_interface_via_dig_io_connection	1.00.a
util_vector_logic_0		util_vector_logic	1.00.a

Figure 4-1 List of IPs used in the project.

All these IPs are connected to the Processor Local Bus (PLB). Plb_v46 is the type of Processor local bus used in this hardware. The communication between the processor PowerPc (ppc440) and other hardware (IPs) is done by PLB. Through PLB, parameters and variables are transferred to and from the program running in the processor. When IPs are connected to the PLB, base address is assigned to them which makes processor easy to access any register by pointers to the specific address. Figure 4-2 shows the block diagram of hardware built in FPGA. It has got processor at the top and all the Intelligent Properties (IPs) that are developed by user at the bottom. Processor communicates to IPs by Processor Local Bus (Plb_v46).

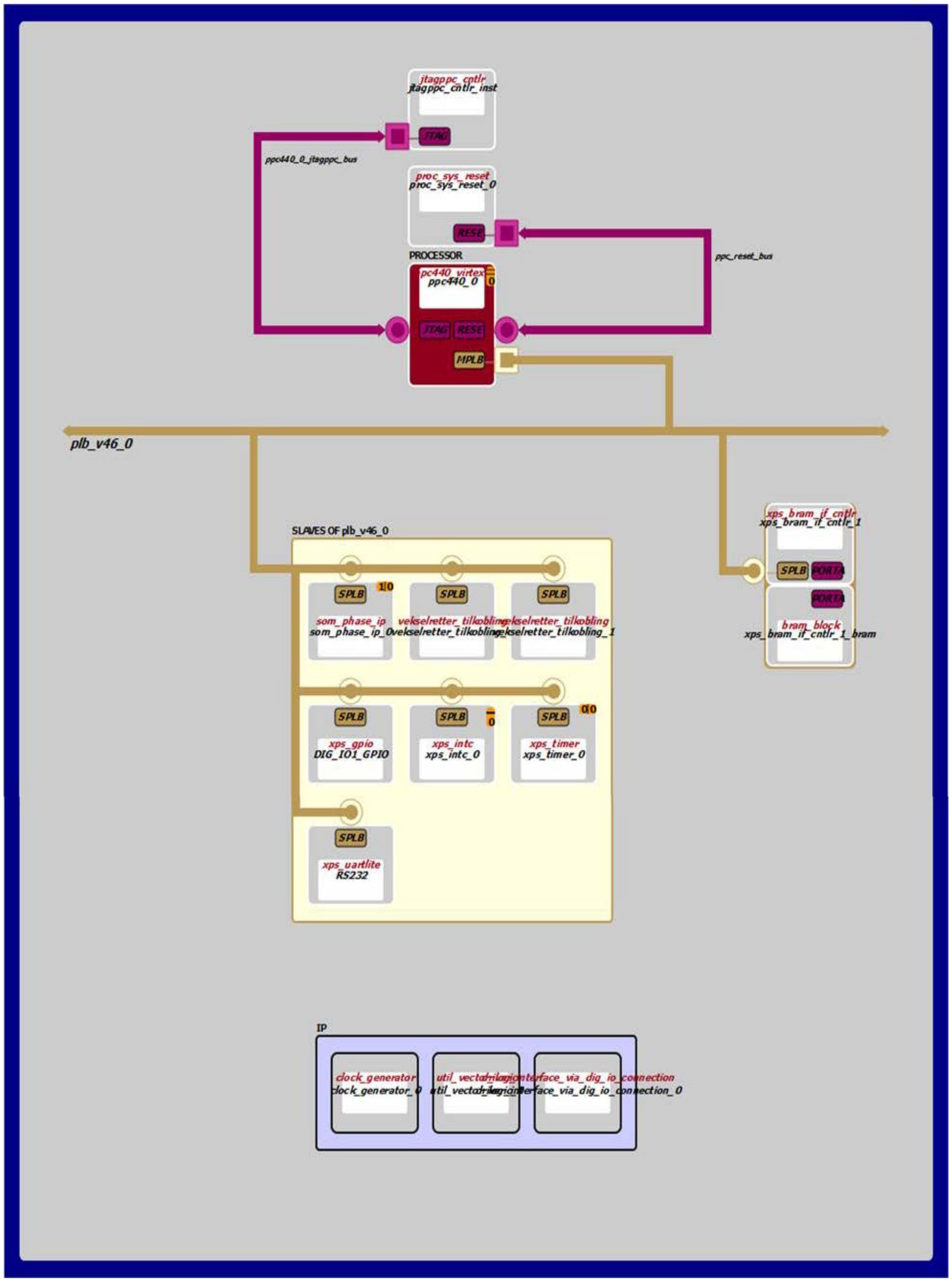


Figure 4-2 Block diagram.

There are three kinds of IPs used here. These are categorized as

1) EDK install
2) Project Local Pcores (user)
3) Global Peripheral Repository (user)

4.1 EDK install

Some of the IPs are provided by EDK itself. These are the commonly used IP blocks. They are described briefly [42].

a) ppc440: This IP is included to use Power PC 440 virtex 5 is the processor that is embedded in FPGA. Its features are

- PowerPC 440x5 dual-issue, superscalar 32-bit embedded processor developed by IBM
- 32 KB instruction cache, 32 KB data cache
- 128-bit Processor Local Bus (PLB) version 4.6 interfaces

b) XPS bram:: This IP is named IP Processor Block RAM. The BRAM Block structural HDL is generated by the EDK design tools based on the configuration of the BRAM interface controller IP.BRAM block is configurable memory module. All BRAM Block parameters are automatically calculated and assigned by the Platgen and Simgen EDK tools.

c) XPS_intc: This is interrupt controller IP available in XPS. It contains multiple interrupt inputs from peripheral device to single interrupt output to the system processor. It connects as a 32 bit slave on processor local bus. To provide additional interrupt they can easily be cascaded and prioritized. The interrupt can be edge triggered or level triggered.

d) Jtagppc_cntrl_inst: JTAGPPC Controller IP helps to connect the JTAG chain of FPGA with the PowerPC processor

e) Proc_sys_reset: This Xilinx Processor System Reset Module design allows user to set certain parameters to enable/disable features.

f) DIG_IO_GPIO: It stands for General Purpose input output port. It also connects as 32 bit slave on PLB. It is configured as single or dual GPIO channels. The number of GPIO bits can be configured from 1 to 32 bit. The width of each of the channel can be individually configured. The ports can be configured dynamically for input or output by enabling or disabling the 3-state buffer.

g) Xps_timer: The XPS Timer/Counter is a 32 bit timer module that is attached as 32 bit slave on PLB. It has configurable counter width.

4.2 Project Local Pcores (user)

FPGA can be used in numerous applications. Every application has its own specific hardware requirement. The intellectual properties developed for application specific need are categorized as project local Pcores. These user developed IP cores can even be transferred to other user. For the implementation of Synchronous Optimal Modulation technique, the IP called SOM_PHASE_IP and SOM_spacevector_IP are developed in this master thesis. These are two ways by which synchronous optimal modulation can be generated.

For generating synchronous optimal modulation, the optimal angles must be calculated. This calculation itself accepts lot of work. Hence such calculation is done in other collaborating master thesis. This can be referred from [15]. The collaborating project is termed as processor routine or software while this project is termed as hardware in this report. The processor routine can perform the task by phase to phase implementation or by space vector implementation. In phase to phase implementation the processor routine deals with different phases independently while in space vector implementation, all the phases are considered together.

If the processor routine performs phase to phase implementation then SOM_PHASE_IP is used in the hardware while SOM_spacevector_IP is used for space vector implementation.

4.2.1 Phase to Phase Vs Space vector Implementation

In Figure 4-3, the numbers of switching instances are shown for three different phases for one sampling period. There are total of seven switching instances in all the phases among which 2 lies in R phase, 3 lies in Y phase and remaining three lies in B phase. In phase by phase implementation, the switching time and state for different phases are calculated separately in software and kept separately in register as shown Figure 4-4. Even though the numbers of switching instances in R and B phases are two, three registers are used. Registers are predefined assuming that total number of switching instances that may occur in one sampling period is three in this case. Hence for the remaining portion, the last value must be repeated. While in this thesis for phase to phase implementation, five registers are dedicated to each phase. However in space vector, if switching occurs even in any one phase, Time register consists the switching time and the state register consists of the states of all the phases at that time as shown in Figure 4-5 .

For software, it can be seen that if phase to phase implementation is done, the calculation time for the optimal angle is lesser than the space vector implementation. It is because there is no unnecessary calculation of states of other phases where switching does not occur. This will help to choose lesser sampling time or higher sampling frequency. Higher the sampling frequencies help to get better output signals according to Nyquist Sampling theory. This is the benefit of using phase to phase implementation over Spacevector implementation.

Even at the cost of longer calculation time, from section 2.3.3.2 it is known that DC bus balancing can be applied easily using space vector implementation. This is benefit of using space vector over phase to phase implementation.

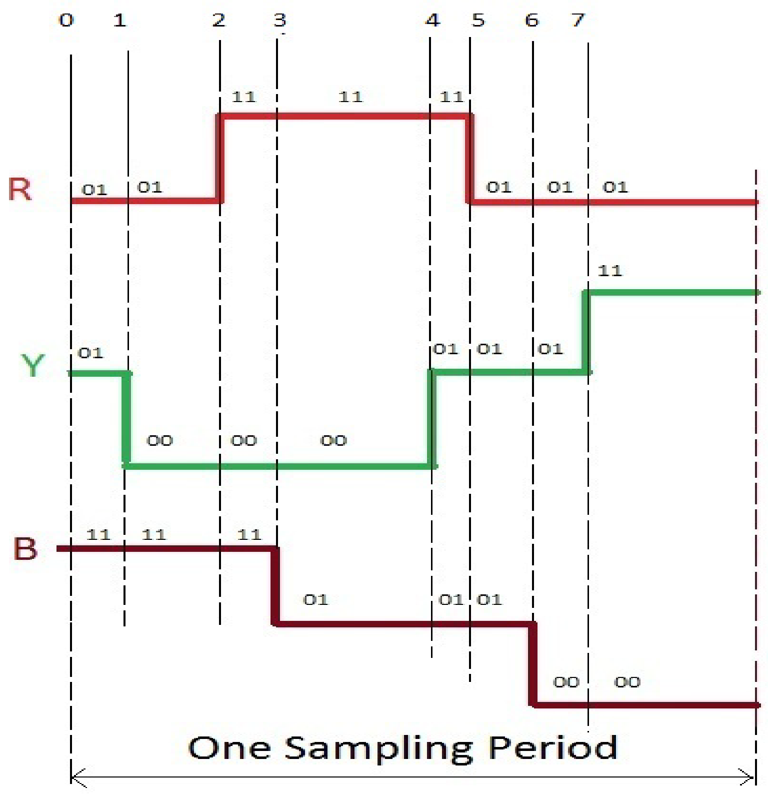
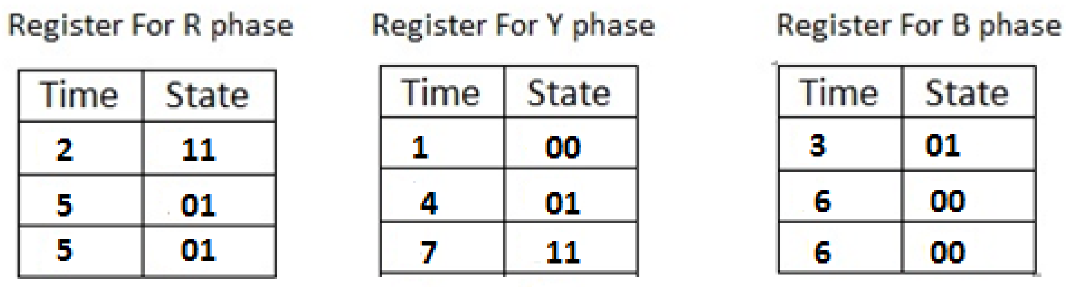


Figure 4-3 Switchings in three phase for one sampling period.



Register For R phase

Time	State
2	11
5	01
5	01

Register For Y phase

Time	State
1	00
4	01
7	11

Register For B phase

Time	State
3	01
6	00
6	00

Figure 4-4 Register used for phase to phase implementation.

Time	(StateR,StateY,StateB)
1	010011
2	110011
3	110001
4	110101
5	010101
6	010100
7	011100

Figure 4-5 Register used for spacevector implementation.

In the section below the two IPs namely SOM_PHASE_IP and SOM_spacevector_IP are discussed in detail for the generation of synchronous optimal signal.

4.2.2 SOM_PHASE IP

The IP module for generating synchronous optimal modulation using phase to phase implementation is called SOM_PHASE_IP. Its input and output ports are shown in Figure 4-6.

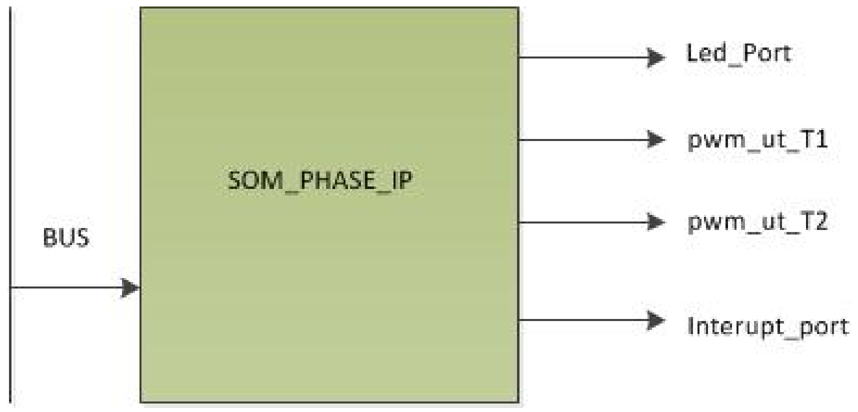


Figure 4-6 Input and output for SOM_PHASE_IP.

4.2.2.1 BACKGROUND

The Synchronous Optimal Modulation pattern for three level converter is illustrated in Figure 4-7 for one phase. There are various numbers of switching instances in one fundamental period. Since this SOM pattern is for three level converter, there are three levels (states) +1, 0, -1. At each switching time, the signal changes its state. In the same figure, the triangle indicates one sampling period. As clearly seen, Sample 1 has three switching instances. Similarly, Sample 2, Sample 3, Sample 4 consists of one, one, two switching instances respectively. Among three switching instances in Sample 1, at first switching instance, the state is changed from 0 to +1, in second switching instance, state is changed from +1 to 0 while in third switching instance, state is change from 0 to +1.

There is assumption of maximum number of switching instances that can occur in one sample. For example, it is assumed that maximum number of switching events that can occur in one sample is three. Then three pairs of registers are allocated individually for all the phases as shown in Figure 4-8. Even though the number of switching events are less than three, the remaining portion are repeated by the last switching time and switching state as shown in Figure 4-4.

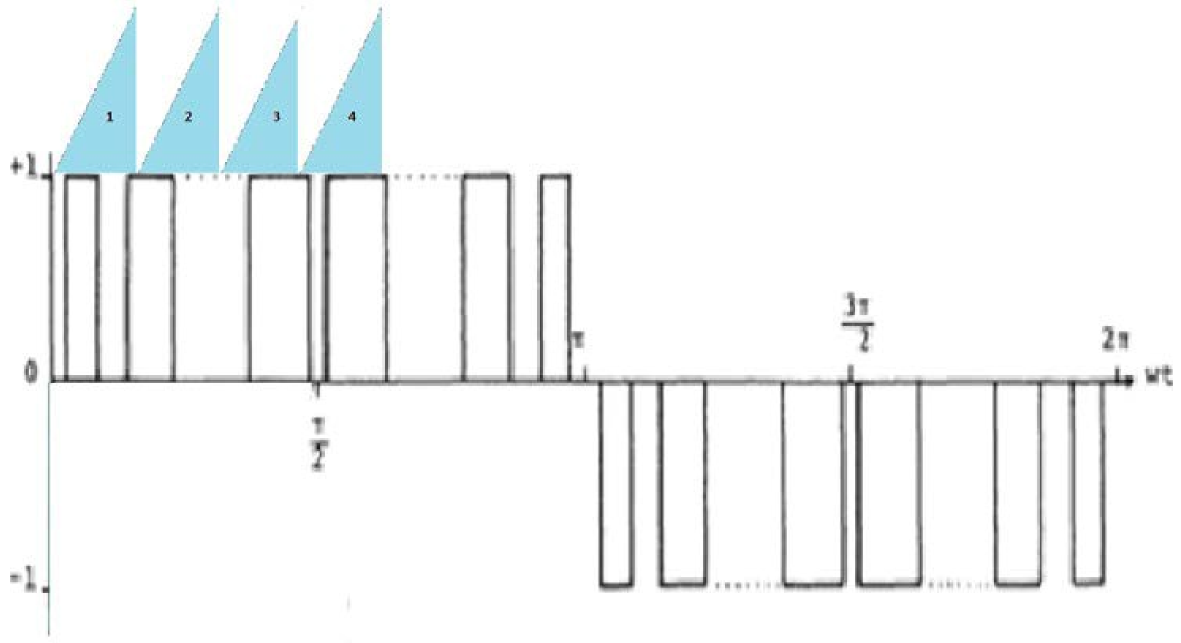
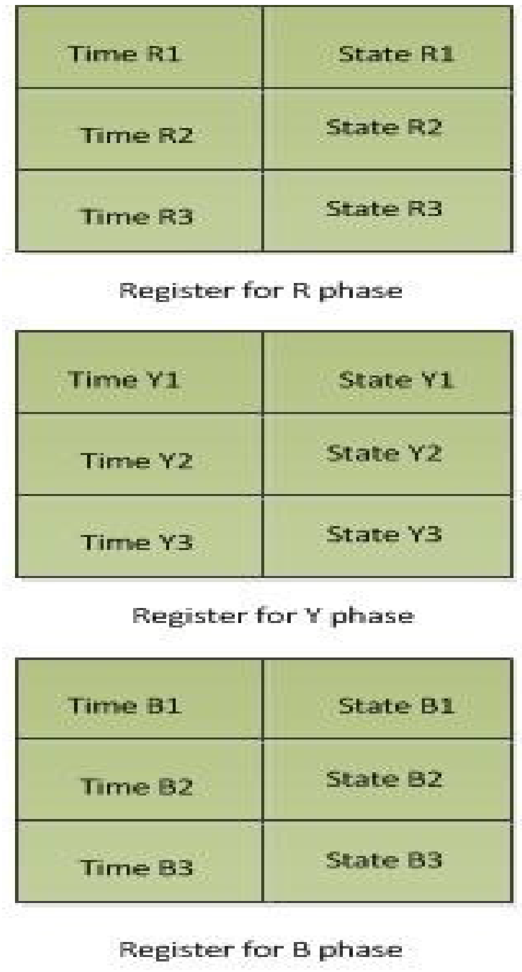


Figure 4-7 Program Modulation with sampling period.

The generation of synchronous optimal modulation pattern occurs in PowerPC processor. For every sampling period, the switching time and the state of all the phases are calculated independently and are stored in register independently as shown in Figure 4-8 This is the responsibility of other collaborating project [15]. While this project only pays attention to the processor bus or register where switching state for particular switching time for one sample are stored in register as shown in Figure 4-8. These are the input to the IP as shown in Figure 4-6



Time R1	State R1
Time R2	State R2
Time R3	State R3

Register for R phase

Time Y1	State Y1
Time Y2	State Y2
Time Y3	State Y3

Register for Y phase

Time B1	State B1
Time B2	State B2
Time B3	State B3

Register for B phase

Figure 4-8 Register for phase to phase implementation.

4.2.2.2 Methodology

This section defines how three phase Synchronous Optimal Modulation circuit is implanted as digital circuit in FPGA. From Power PC processor, time and state for one sampling period for all the phases are kept in Processor Local Bus in ascending order of switching time. It is seen from Figure 4-9, there are separate circuit for every phase which are operating in parallel. This kind of parallel processing is possible in hardware part of FPGA. The codes written in VHDL actually generate the digital circuits. These parallel circuits are synchronised by one single clock. The clock generates the up counter signal which counts from zero to uppermost value and settles down to zero. The counting signal is represented by the right angle triangle. Up counter continues to generate this signal. The time period of the up counting signal is equal to the length of one sampling period. Since the digital circuits are identical in all the phases, only one phase is discussed.

At the beginning of each sampling time or when counter starts from zero, registers are stored with all switching times and states. The switching times and switching states of all the phases stored in the registers are transferred to the local arrays. Switching time which is stored in array is compared with counter. The pointer called I is pointing to the first switching event. Whenever counter equals the switching time pointed by I then output of the comparator is high. As the rising edge of the pulse is seen, state for the switching time pointed by I is latched. This means that inverter is switched to the given state for the given switching time. The rising edge of pulse will also trigger the edge trigger block to increment the value of I and points to the next switching time and next switching state. The new switching time is not equal to counter value so the comparator will again give low pulse. The comparator will wait till counter equals to next switching time for the output to be high. Then the process continues by incrementing counter and comparing to the second switching time. By the end of one sampling period, all the switching states in one sample are latched. Counter reaches the maximum value and resets to zero. This is beginning of another sample and all the process repeats.

The method is same for all the phases. The R phase digital circuit latches the switching states of R phase and similarly other phases latch switching states of their own states. The latched value is taken to the inverter for switching.

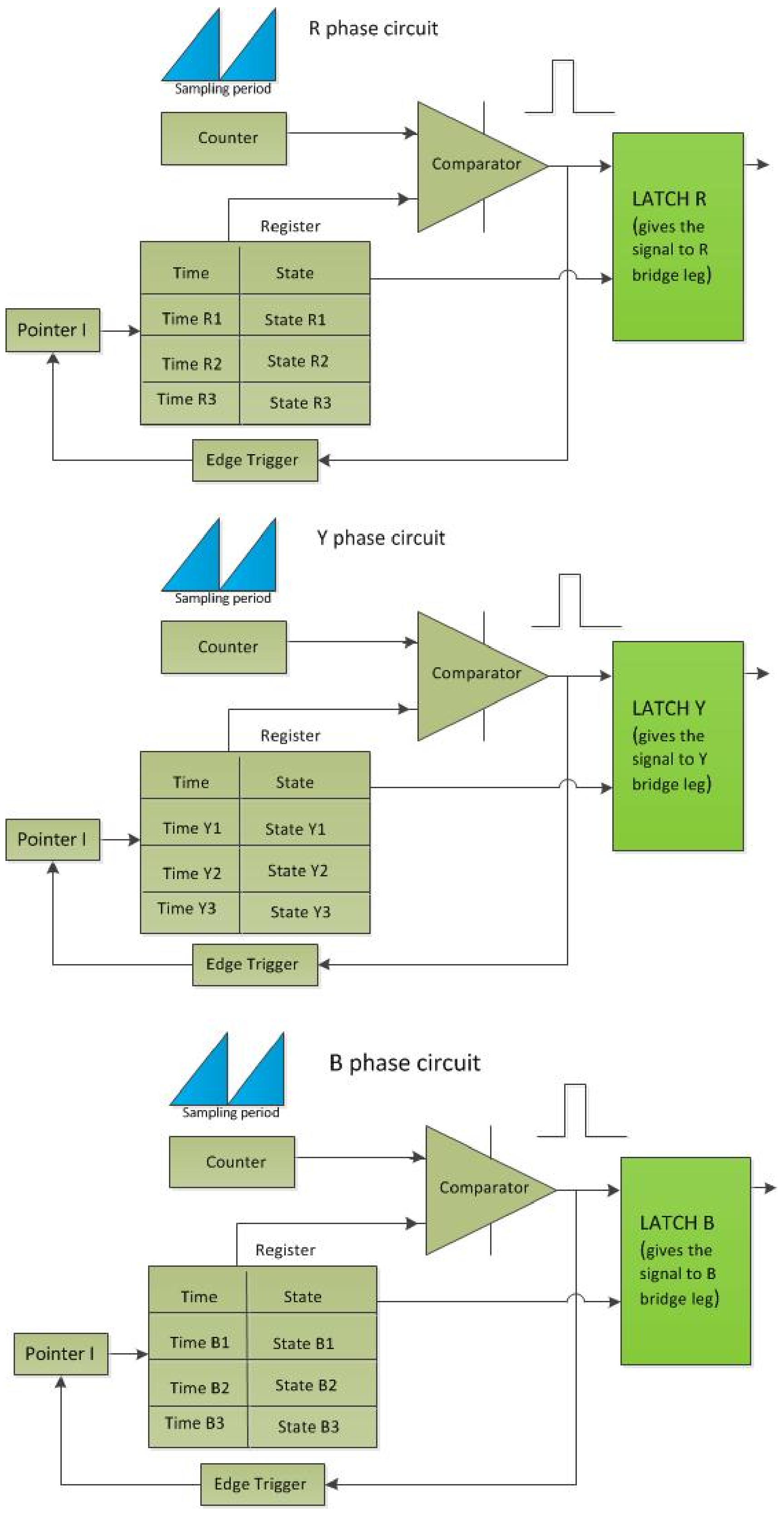


Figure 4-9 Function of SOM_PHASE_IP [43].

Since the parallel processing of different phases are possible at the same time, this indicates the possibility of extension of this firmware to the multiphase machine. By incrementing number of phases also, the executing time for the modulation process is also extremely fast because all the phases are processed parallel. However this kind of parallel processing is not available in conventional programming language like C, C++ etc.

The method is also understood from the flowchart given in Figure 4-10. The flowchart is given for R phase only. However the method is all same for all the phase.

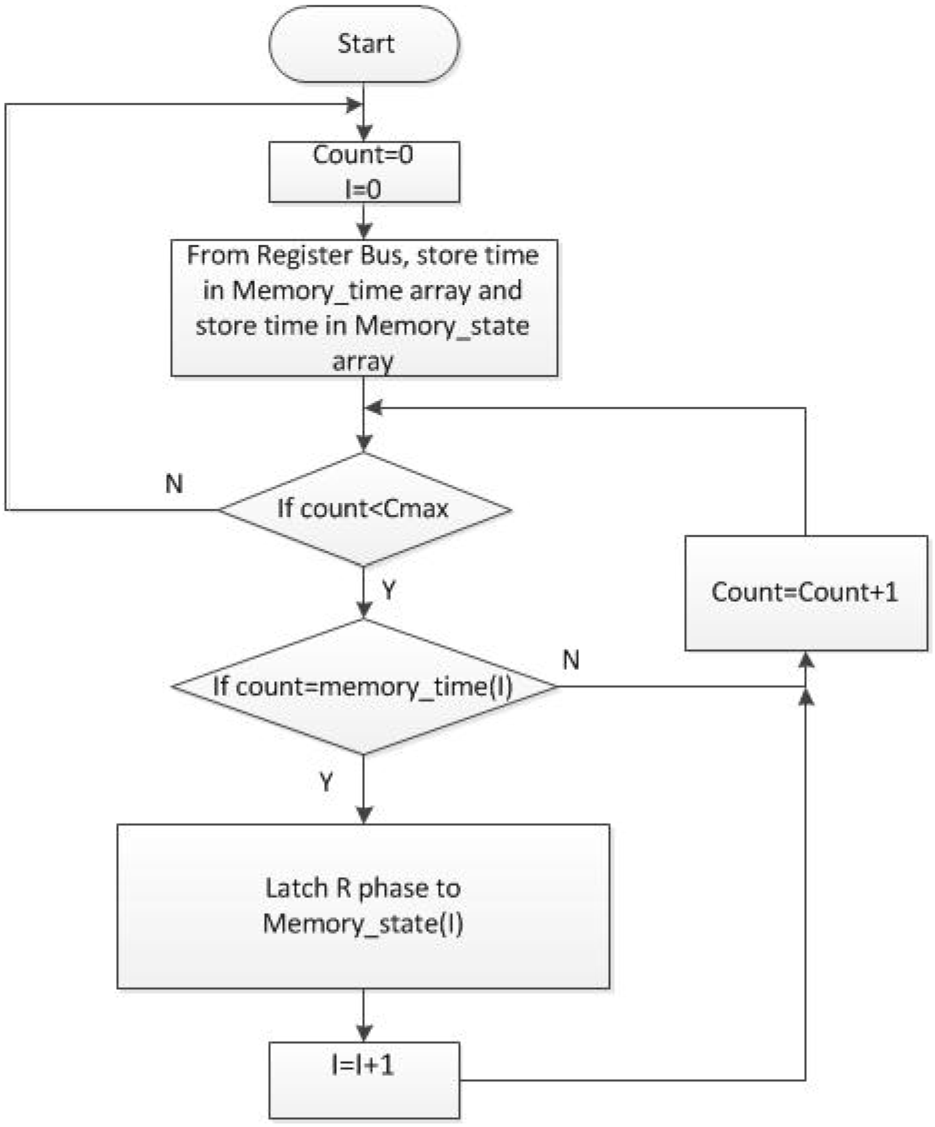

Start
Count=0
I=0
From Register Bus, store time in Memory_time array and store time in Memory_state array
N
If count<Cmax
Y
Count=Count+1
N
If count=memory_time(I)
Y
Latch R phase to Memory_state(I)
I=I+1

Figure 4-10 Flowchart for SOM_PHASE_IP.

At the start of every sampling, up counter, variable I starts from zero. The switching times and switching states of all the phases stored in the registers are transferred to the local arrays. Switching times are stored in local array called Memory_time while switching states are stored in local array called Memory_state. The maximum value of counter is called Cmax. The constraint for this IP is that software must put the switching times and switching states in registers in the ascending order of switching time.

When counter increments by one value, then it is compared with the first value of Memory_time(I), counter keeps on incrementing its value until the counter equals to the first switching time. Once it equals the first switching time, R phase would be latched to the first switching state stored in the array Memory_state(I). After latching, value of I is incremented.

Then the process continues by incrementing counter and comparing to the second switching time. By the end of one sampling period, all the switching states in one sample are latched. Counter reaches the maximum value and resets to zero which is the beginning of new sample and all the process repeats. VHDL codes for SOM_PHASE_IP is shown in Appendix C.

4.2.3 SOM_spacevector_IP

This IP also generates the synchronous optimal modulation for three phases. However the methodology of generation is slightly different from that defined in SOM_PHASE_IP. In the SOM_PHASE_IP, the states in each phase are treated separately while in this section all the states of all the phases are taken into consideration at once. But input and output signals are same as that of SOM_PHASE_IP seen from Figure 4-11.

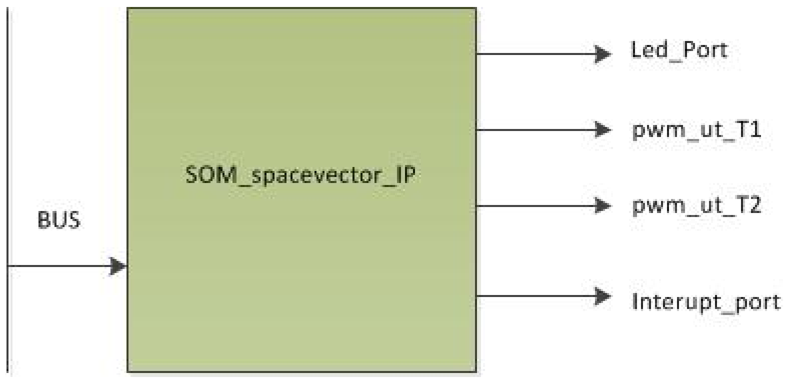
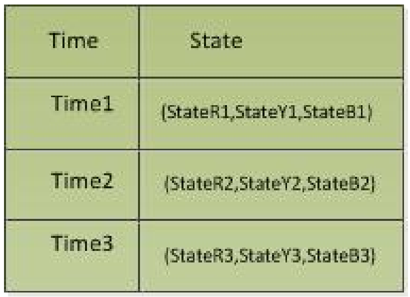


Figure 4-11 Input and output of SOM_spacevector_IP.

4.2.3.1 *Background*

The switching instances in all the phases (RYB) of one sampling period are given in the Figure 4-3. The total numbers of switching states in all the phases are seven. In SOM_spacevector_IP the states of all the phases are accumulated and placed in one register.



Time	State
Time1	{StateR1,StateY1,StateB1}
Time2	{StateR2,StateY2,StateB2}
Time3	{StateR3,StateY3,StateB3}

Figure 4-12 Register used for SOM_spacevector_IP.

Time registers consist the switching time of any phase and the state registers consist of the states of all the phases at that time. Even though the state of only one phase changes, state registers also consist of states of other two remaining phases. The state registers consist of vector of states of R phase, Y phase and B phase respectively. In this way, the numbers of registers are also decreased. This kind of space vector implementation also helps in DC bus balancing in NPC inverter.

4.2.3.2 *Methodology*

The registers store the switching times and switching states in ascending order of switching time. The state indicated in this section is the vector consisting states of all the three phases. According to Figure 4-13, counter generates up counter signal which counts from 0 to maximum value and resets again to 0. The counting signal is represented by the right angle triangle. Up counter continues to generate this signal. The time period of the up counting signal is equal to the length of one sampling period.

At the beginning of each sampling time, registers are stored with all switching times and states. The switching times and switching states of all the phases stored in the registers are transferred to the local arrays. Switching time which is stored in array is compared with counter. The pointer called I is pointing to the first switching time. Whenever Counter equals the switching time pointed by I then output of the comparator is high. As the rising edge of the pulse is seen, state vector for the switching event pointed by I is latched. This means that inverter is switched to the given state vector for the given switching time. The rising edge of pulse will also trigger the edge trigger block to increment the value of I and points to the next switching time and next switching state vector. The new switching time is not equal to Counter value so the comparator will again give low pulse. The comparator will wait till Counter equals to next switching time for the output to be high. Then the process continues by incrementing counter and comparing to the second switching time. By the end of one sampling period, all the switching states in one sample are latched. Counter reaches the maximum value and resets to zero which is the beginning of new sample and all the process repeats.

It is seen the methodology is almost same like that described for SOM_PHASE_IP. The only difference is the placement of state vectors in the state register. This method can also be implemented for multiphase machine. The state vector length would be increased and number of switching per sampling would be increased when implemented on multiphase system. Hence this IP can easily be extended for multiphase machine too.

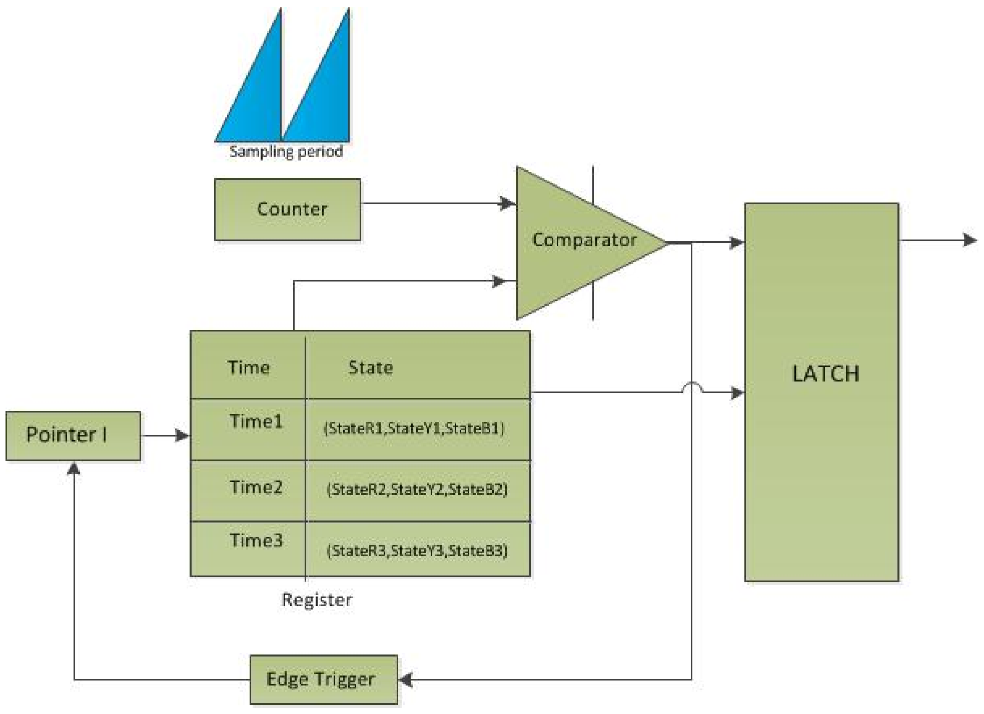


Figure 4-13 Function of SOM_spacevector_IP[43].

The flowchart for SOM_spacevector_IP is shown in Figure 4-14. At the start of every sampling period, Up counter, variable I starts from zero. The switching times and switching states of all the phases stored in the registered is transferred to the local arrays. Switching times are stored in local array called Memory_time while switching states are stored in local array called Memory_state. The maximum value of counter is called Cmax. The constraint for this IP is that software must put the switching times and switching states in registers in the ascending order of switching time.

When counter increment by one value, then it is compared with the first value of Memory_time(I), counter keeps on incrementing its value until the counter equals to the first switching time. Once it equals the switching time, all the vectors would be latched to the switching vector stored in the array Memory_state(I). After latching, value of I is incremented.

Then the process continues by incrementing counter and comparing to the second switching time. By the end of one sampling period, all the switching states in one sample are latched. Counter reaches the maximum value and resets to zero which is the beginning of new sample and the process repeats. VHDL codes for SOM_spacevector_IP is shown in Appendix D.

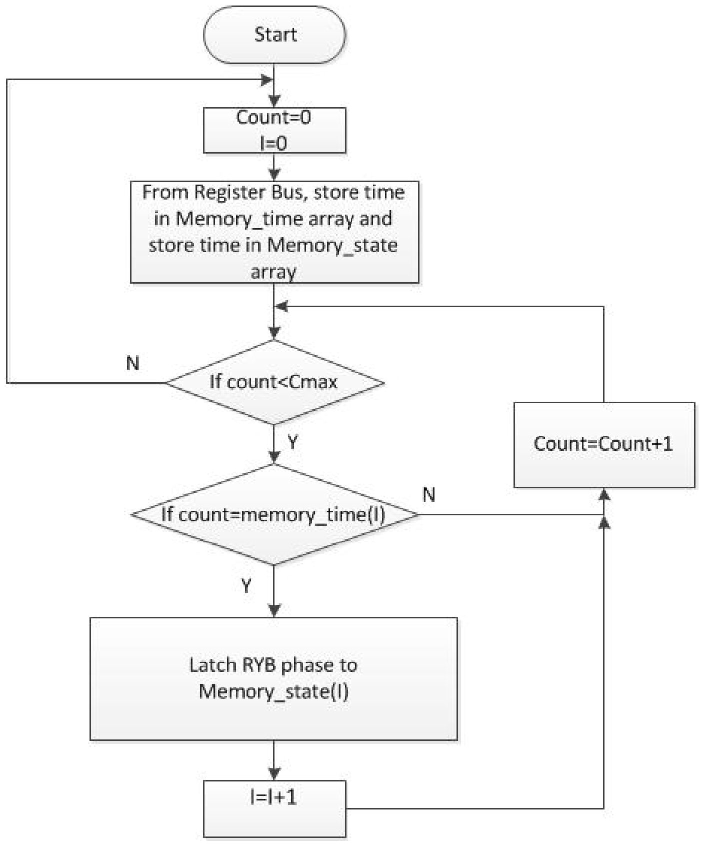

Start
Count=0
I=0
From Register Bus, store time in Memory_time array and store time in Memory_state array
N
If count<Cmax
Y
Count=Count+1
N
If count=memory_time(I)
Y
Latch RYB phase to Memory_state(I)
I=I+1

Figure 4-14 Flowchart for SOM_spacevector_IP.

4.3 Global Peripheral Repository (user)

Drive control applications seek complex algorithm structure. In order to make the design of control algorithms more manageable and less intuitive, reusability of already made IP can be done. Some of the IPs are already created by Kjell Ljøkelsøy of SINTEF. The IPs which are taken from his library source are described briefly below.

4.3.1.1 Vekselretter tilkobling (Inverter Connection):

Vekselretter tilkobling is the Norwegian word for Inverter Connection. The input and output signal for this IP is shown in Figure 4-15 .

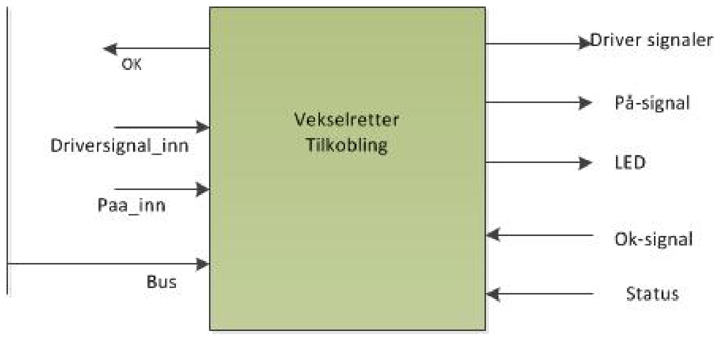


Figure 4-15 Input and output Vekslretter Tilkobling.

This IP is only applied for two level, three phase converter as shown in Figure 4-16. The upper switches are T_{A+} , T_{B+} , T_{C+} while the lower switches are T_{A-} , T_{B-} , T_{C-} . The signal to the lower switch is complementary to the signal to the upper switch. Only the signal for the upper switch is given as input (Driversignal_inn) to Vekselretter tilkobling. It does some operation to produce the signal for upper as well as lower switch. The output Driver Signaler consist the signal for all the switch of the two level inverter.

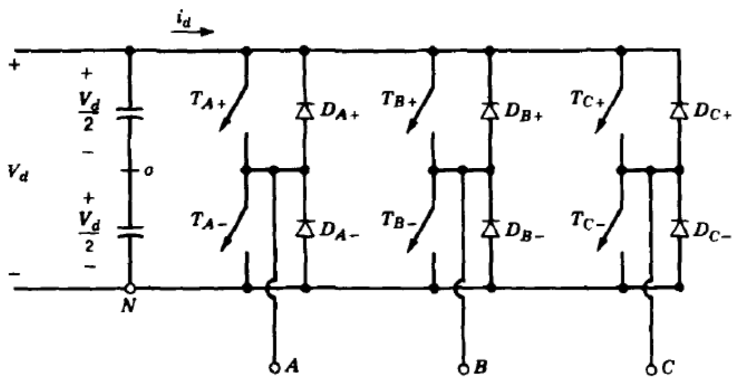


Figure 4-16 Two Level three phase converter.

The Figure 4-17 shows the function of IP. Driversignal_inn gives the signal to the upper switch while driversignal_inn_L gives the signal to the lower switch. These two signals can be separately given. Another choice is giving only the upper signal so that lower signal can be dependent on the upper signal. Such choice is handled by NEDRE_DRIVER_SEPARAT_KILDE. This is the Norwegian form of lower_driver_separate_source. There are four choices for getting the output, driver_ut_H and driver_ut_L controlled by signal called funksjon_H and funksjon_L respectively. The output can be permanently off (0), invertered driversignal_inn , same signal as driversignal_inn or permanently on(1).

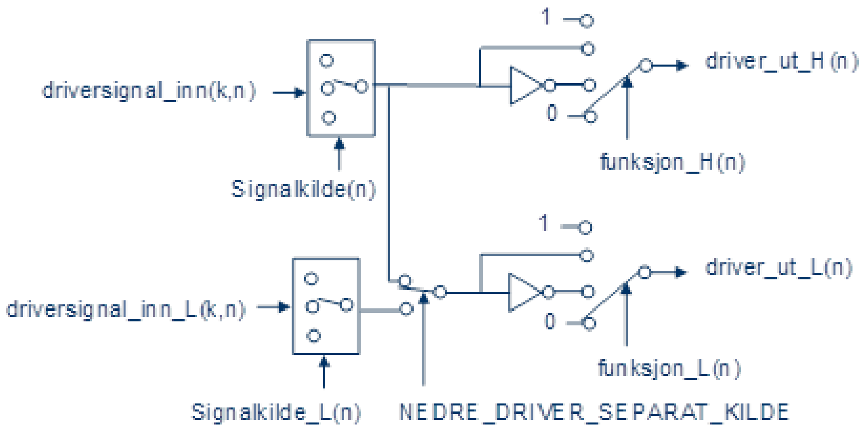


Figure 4-17 Function of Vekslretter Tilkobling[42].

In this project work, NEDRE_DRIVER_SEPARAT_KILDE signal is made zero meaning there are no separate lower driver signal. Only upper driver signal is used and lower signal depends on upper signal. The output of the driver_ut_H is the signal driversignal_inn itself while the output of the driver_ut_L is inverted driversignal_inn. It is because signals to the lower switches are complementary to the upper switches drive signal. driver_ut_H, driver_ut_L is collectively called Driver Signaler which is the output of the Vekselretter tilkobling IP.

4.3.1.2 *Inverter for three level converter*

Neutral Point clamped (NPC) converter is used for three level converter as shown in Figure 4-18. NPC converter is combination of two, 2 level converter. T_1 and T_3 are one set of two level converter where T_3 is the complement of T_1. Similarly T_2 and T_4 are another set of two level converter. Hence for the application of three level converter, two Vekselretter tilkobling IPs must be used as shown in Figure 4-19. Vekselretter tilkobling 1 must feed to the one set of two level converter (T1,T3) by Driver signaler1 while Vekselretter tilkobling2 must feed to another set of two level converter (T2, T4) by Driver signaler 2 as shown in Figure 4-18 and Figure 4-19.

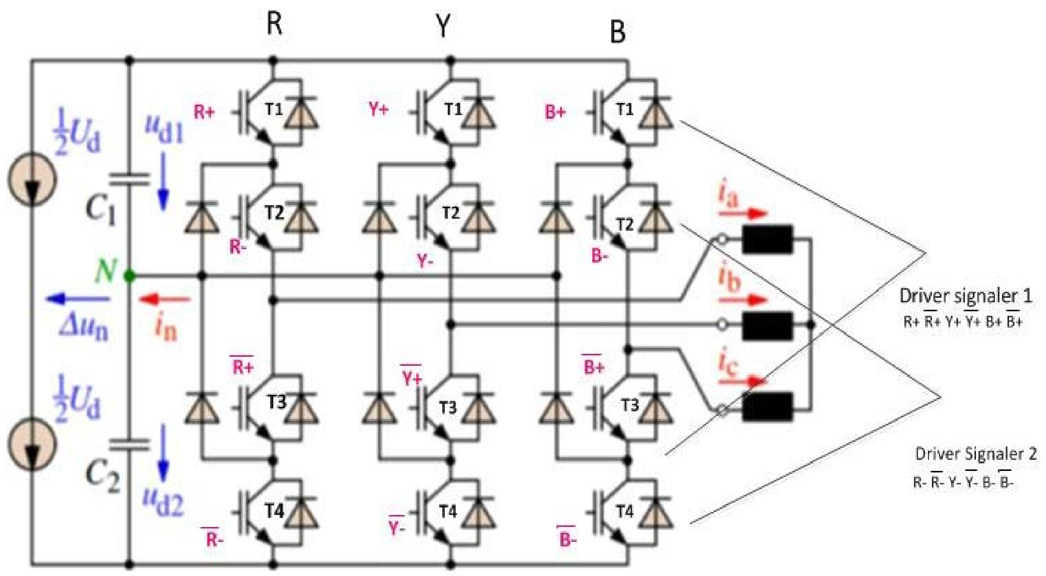


Figure 4-18 Neutral Point Clamped Converter.

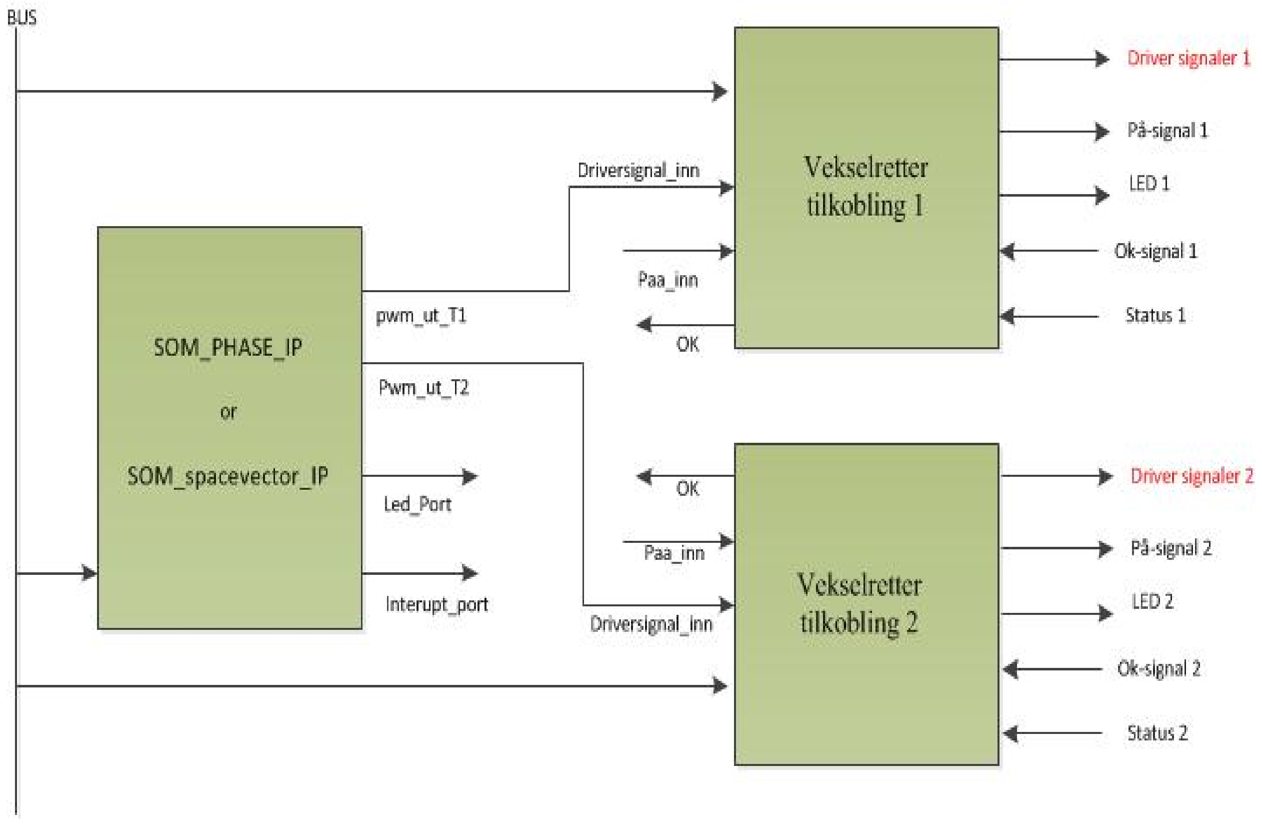
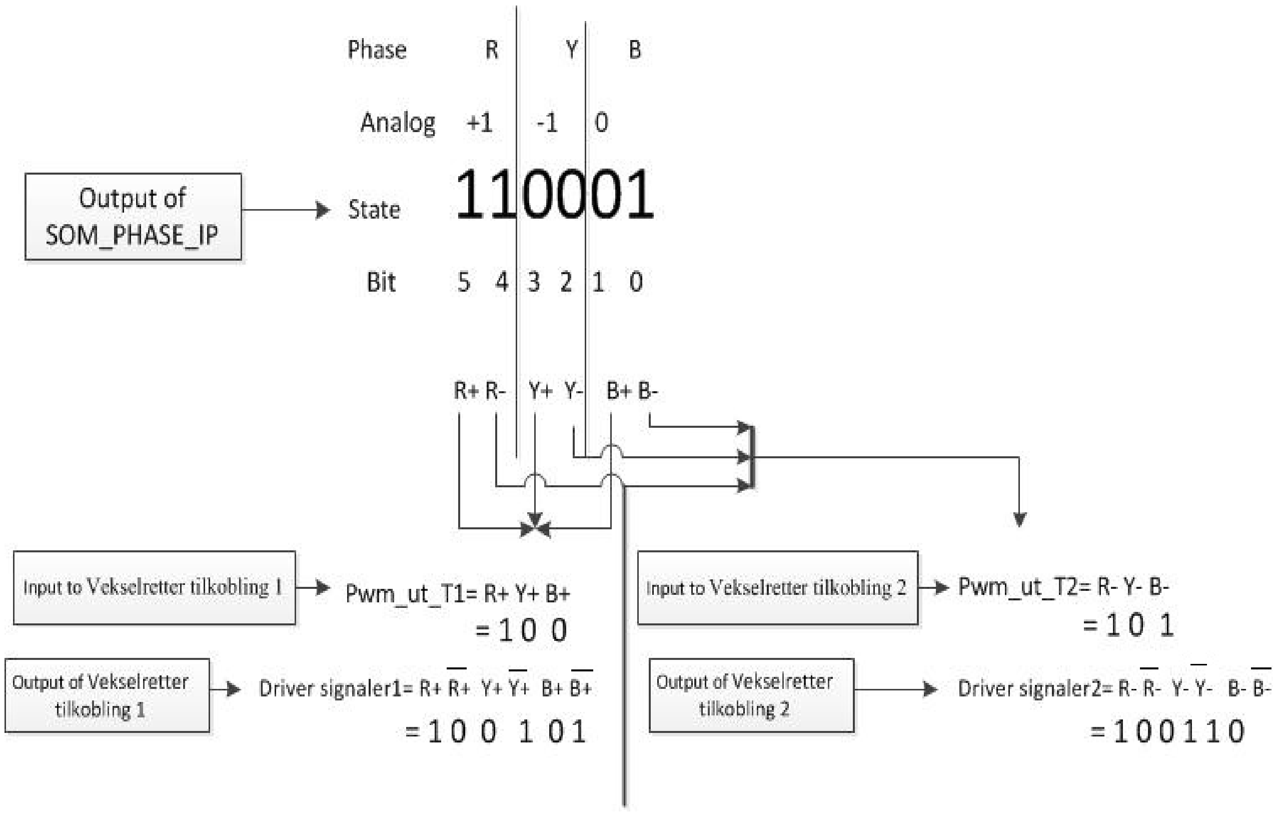


Figure 4-19 Use of two Vekselretter tilkobling for three level converter.

There are three levels in three level converter which are +1 , 0 , -1. In the section 2.3 it is defined that if only signals to upper two IGBT are considered then +1 can be obtained by giving signal 11 to T_1 and $T_{2,}$ while 0 can be obtained by giving signal 01 to T_1 and T_2. Similarly -1 can be obtained by giving signal 00 to T_1 and $T_{2.}$ The signals to the lower drives are the complementary to the upper drive signals. Hence 11, 01 , 00 are the states that can define the level +1, 0, -1.

SOM_PHASE_IP or SOM_spacevector_IP gives the state of three phases as the output. The output is six bitstream binary value. Suppose the output of the SOM_PHASE_IP is 110001 as shown in example shown in Figure 4-20. That means R phase, Y phase, B phase should be switched to +1,-1,0 analog switching states respectively because Bit5 and bit4 indicate the state for R phase. Bit3 and bit2 is the state for Y phase and remaining bit1 and bit0 is the state for B phase. These two bit per phase must be the input to the upper two IGBTs i.e T1 and T2. T1 with T3 makes one 2 level converter while T2 with T4 makes another 2 level converter. Hence the first bit must be given to Inverter1 (T1, T3) while second bit must be given to Inverter2 (T2, T4).



Phase R Y B
Analog +1 -1 0
Output of SOM_PHASE_IP
State 110001
Bit 5 4 3 2 1 0
R+ R- Y+ Y- B+ B-
Input to Vekselretter tilkobling 1
Pwm_ut_T1= R+ Y+ B+
= 1 0 0
Input to Vekselretter tilkobling 2
Pwm_ut_T2= R- Y- B-
= 1 0 1
Output of Vekselretter tilkobling 1
Driver signaler1= R+ $\overline{R+}$ Y+ $\overline{Y+}$ B+ $\overline{B+}$
= 1 0 0 1 0 1
Output of Vekselretter tilkobling 2
Driver signaler2= R- $\overline{R-}$ Y- $\overline{Y-}$ B- $\overline{B-}$
= 1 0 0 1 1 0

Figure 4-20 Example.

These states are two bit for each phase. The collection of first bit of states of all the phase should be given to T_1 and collection of second bit of state of all the phase should be given to T_2 in order to generate the required analog signal from the digital states. The output states of SOM_PHASE_IP or SOM_spacevector_IP should be split and given separately to two Vekselretter tilkobling IPs.

Figure 4-19 illustrates how output of SOM_IP is splitted into two. One of the set called pwm_ut_T1 goes to Vekselretter tilkobling1 which produce the signal called Driversignaler1 for T_1 and T_3 for all the three phases RYB. Similarly, another set (pwm_ut_T2) goes to Vekselretter tilkobling2 to produce the signal called Driversignaler2 for T_2 and T_4 for all the three phases RYB. The input to the Vekselretter tilkobling is just the signal for the upper switch of the two level converter while the output of Vekselretter tilkobling is Driversignal which produce the signal for the upper and lower switch. It has been mentioned lot of time that the lower signal is the complementary of upper signal.

4.3.2 Driver Interface via dig_io_connection

It can be seen from Figure 3-4 that FPGA card used in this thesis has only one converter driver interface port from which driver signals are given to the converter. The output of Vekselretter tilkobling is connected to such driver interface port. But this project deals with three level converter so there is one extra Vekselretter tilkobling which also seek another converter driver interface port. Therefore this IP called Driver Interface via dig_io_connection is created. This would direct the output of Vekselretter tilkobling2 to General Purpose Input Output (GPIO) port through DIG_IO signal as shown in Figure 4-21. One additional card called buffer card is connected to that GPIO. The input terminal of Buffer card is connected to GPIO and at the output, there is converter driver interface port which is required by Vekselretter tilkobling2 to send the driver signals to converter. DIG_IO is the name of signal to GPIO. Hence it is clear from the name of IP itself that driver interface port is connected to Vekselretter tilkobling via dig_io. This is illustrated in Figure 4-21 and Figure 4-22 .

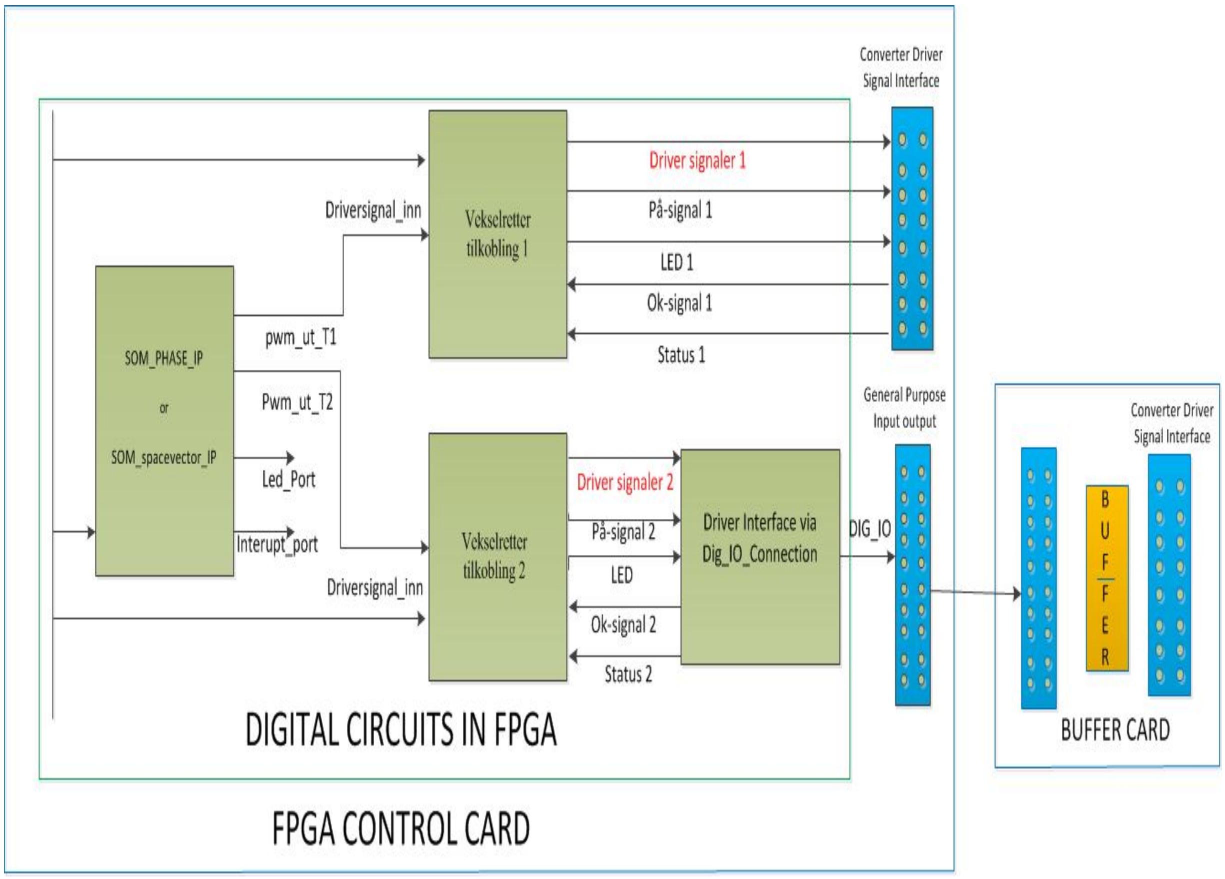


Figure 4-21 Use of Driver Interface via Dig_IO_Connection.

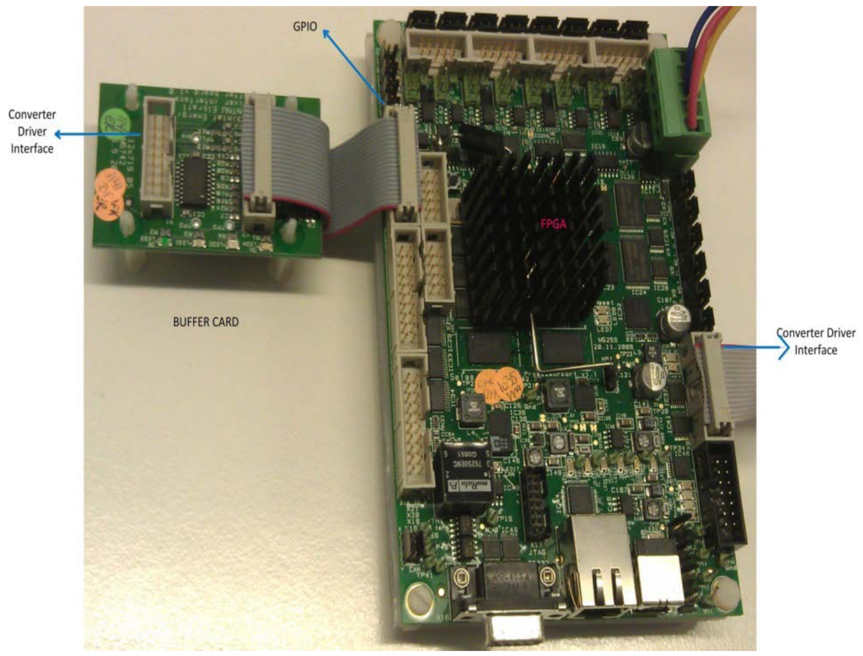


Figure 4-22 FPGA card with buffer card.

4.4 Driver circuit

From above section it is found that the output of Vekselretter tilkobling is the driver signal circuit for two level, three phase converter. Hence two Vekselretter tilkobling IP is used in this project because this project deals with three level, three phase converter. But keeping aside this fact, only one Vekselretter tilkobling is considered in this section to find out how the signals generated are given to IGBT switch through the driver circuit.

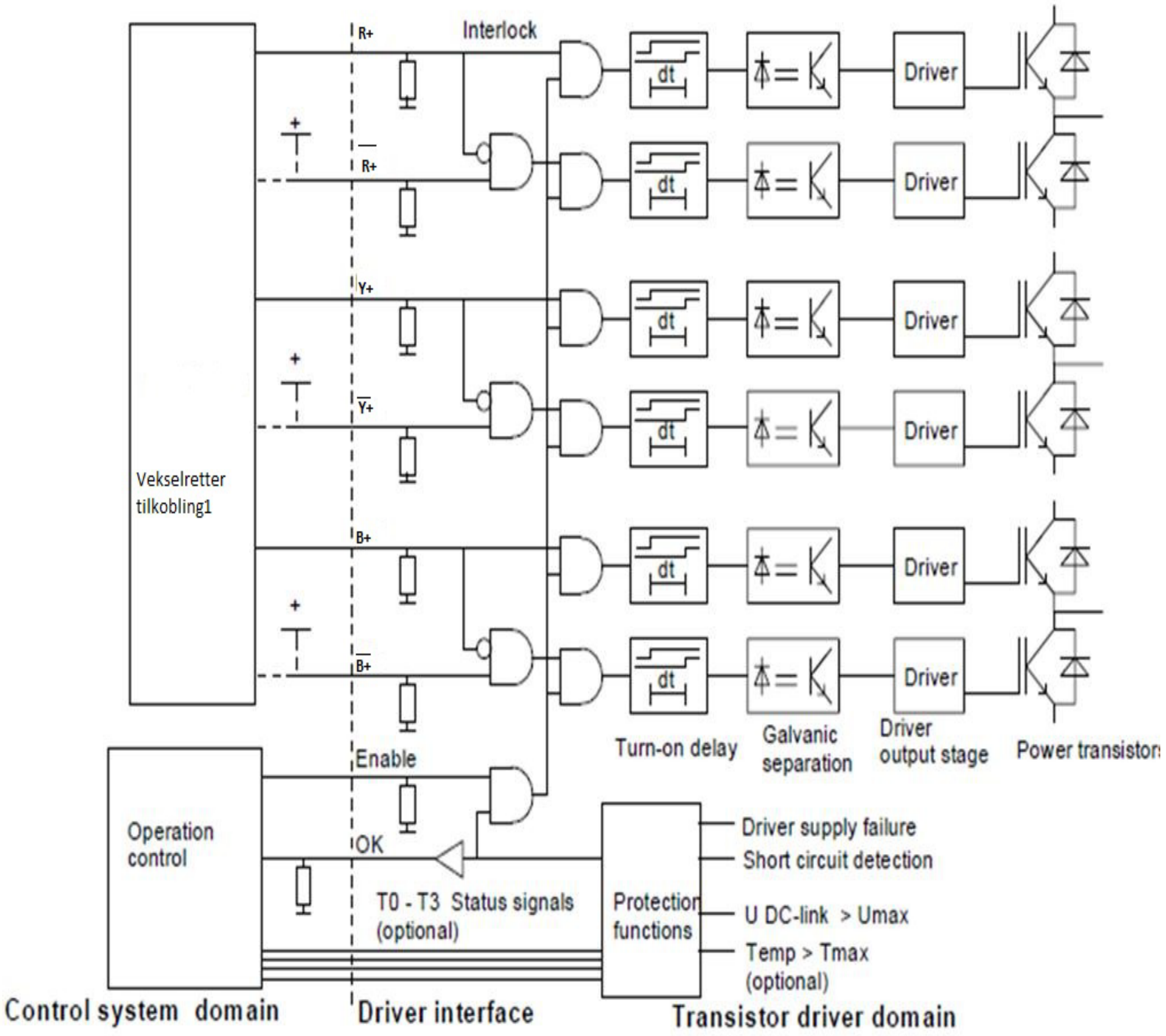


Figure 4-23 Driver Circuit[42].

The Figure 4-23 shows how the signals from modulator are connected to the IGBTs. The output signals of Vekselretter tilkobling are Driver signaler which contains the signals for upper and lower switch of two level, three phase converter. The upper and lower switch of one bridge leg should not be ON at the same time which will short the dc bus. There must be some delay between turning ON of two switches in one bridge leg. This delay time is included in this circuit. Whenever any fault occurs in power circuit or power transition then they must not affect the FPGA control card which is operating in very low voltage than the power circuits. Hence galvanic separation is done to isolate the power circuit from logic circuit or FPGA control card.

4.5 Communication with other collaborating project

In chapter 2, it is already mentioned that in SOM pattern, switching events are determined in a way that reduces the harmonic content in the current, also reducing losses due to harmonic distortion in the controlled induction machine. Calculation of such switching events is done in the other collaboration project using Software Development Kit. In the following text, the terms like software, processor routine and SDK refer to the other collaborating project while the terms like XPS, hardware refer to this project.

Digital control signals are carried out by dividing signals into sampling intervals. The hardware consists of up counter which starts from zero to maximum value and settle down to zero again. Sampling period is equal to the length of counter. The sampling intervals are shown in Figure 4-24.

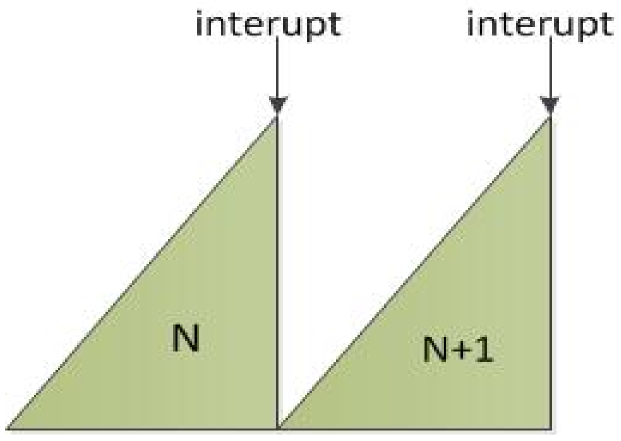


Figure 4-24 Samples N and N+1.

In sampling interval N, the switching times and states of SOM for next sampling interval (N+1) are calculated in software and place in register which is accessible by hardware. The time of calculation and placing in register is less than the sampling period. Software waits until the current sampling period is finished. As the up counter settles from maximum value to zero, the hardware sends interrupt to the software. Once Software gets interrupt, it again starts to calculate the switching time and states for next sample (N+2).

After giving interrupt to software, the up counter in hardware starts from zero in N+1 Sample. The switching times and states needed for N+1 Sample is already stored in register by Software. Hardware reads the switching times and states from the registers only once and copies into its own arrays in order to free the registers so that software can use them to calculate values for N+2 Sample. During N+1 Sampling period, hardware switch IGBTs to given state at the given time.

During Phase to Phase implementation, software places switching time and switching state of each phase independently. And in Space Vector implementation, software places the switching times and states of all the phases together as explained in section 4.2.1.

4.6 Challenges faced

The processor routine calculates the optimal angles for generating synchronous optimal modulation. Such calculation should be less than sampling time as described in section 4.5. Initially, it was assumed that the sampling time is 1ms. As the project was carried out, it was realized that the time for calculation of optimal angles exceed 1ms, to almost 1.5ms. In order to calculate optimum angle software needs to handle floating point unit. Power PC takes very long time to handle floating point unit. It takes longer time for the calculation which can be longer than sampling period. But it is not accepted.

In order to solve this problem, the floating point processor is included along with Power PC processor. This is added as separate auxiliary processor which is used only to handle the floating point unit. This will decrease the calculation time, lesser than the sampling time. After adding floating point unit, Figure 4-2 would have addition of apu_fpu_virtex5 as shown in Figure 4-25.

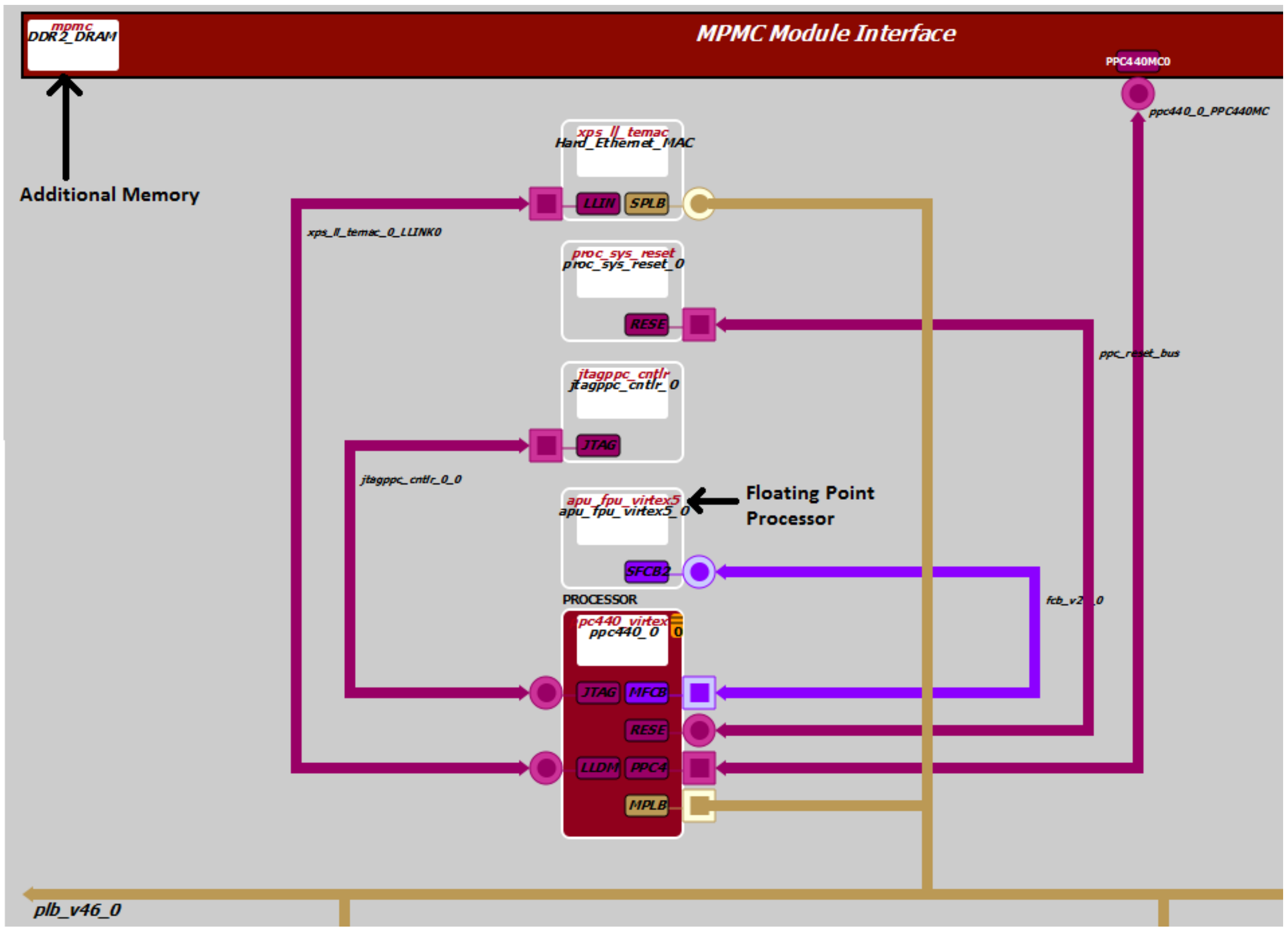


Figure 4-25 Floating Point Unit processor and additional memory.

Program Modulation technique is based on storing the optimum angles. Since this project deals with three phase modulation circuit, more memory is required, hence additional memory is added as seen in the same figure. After these modification the overall block diagram of hardware is shown in Appendix E.

4.7 Simulation

The project is divided into two parts. In this part implementation of three level converter in FPGA is done. Once the implementation is finished, this project must be exported to the other project. The combination of both of the project will create the synchronous optimal modulation for medium voltage multiphase machine. Before exporting to the other project, some of the test has to be performed to see if this project successfully achieved its objective. Two method of generating PWM signal are discussed in above sections. Therefore both of them are tested.

4.7.1 SOM_PHASE_IP

Suppose the generated signals for three phases from the processor routine (software) are as shown in Figure 4-26. It is just the assumption for the test signals.

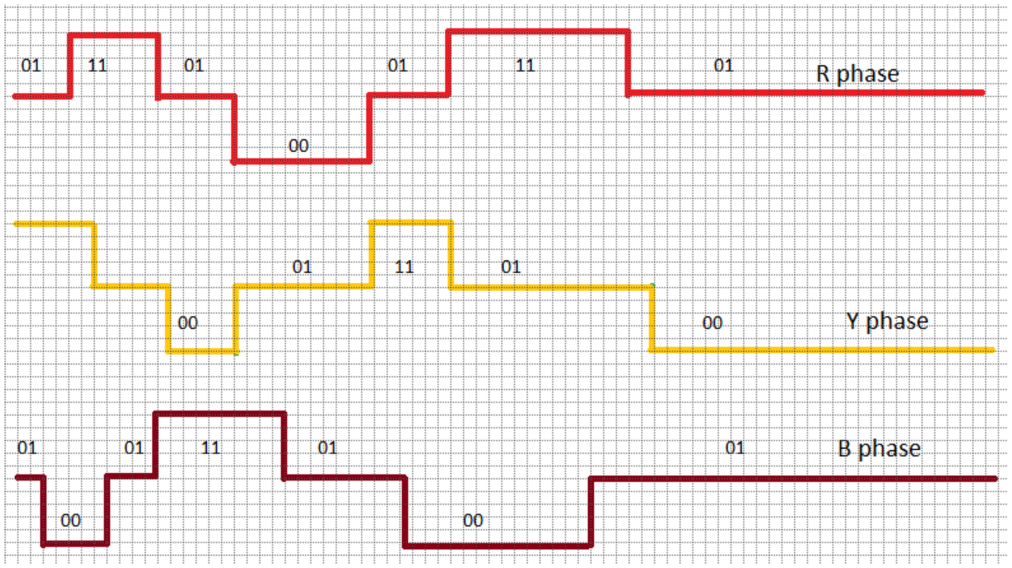


Figure 4-26 Test signals.

Digital control signals are sampled. The sampling period is taken to be 500us (sampling frequency= 2Khz) for testing while for the project sampling period is 1ms (sampling frequency= 1Khz). The sampled value for R, Y, B phase are shown in Figure 4-27,Figure 4-28 and Figure 4-29. It is seen that Sample 1 has three switching instances in all the phases. Similarly, Sample 2, Sample 3, Sample 4 has two, one and zero switching instances in all the phases. The meaning of zero switching instances in Sample 4 means the states of previous sample is continued in this sample.

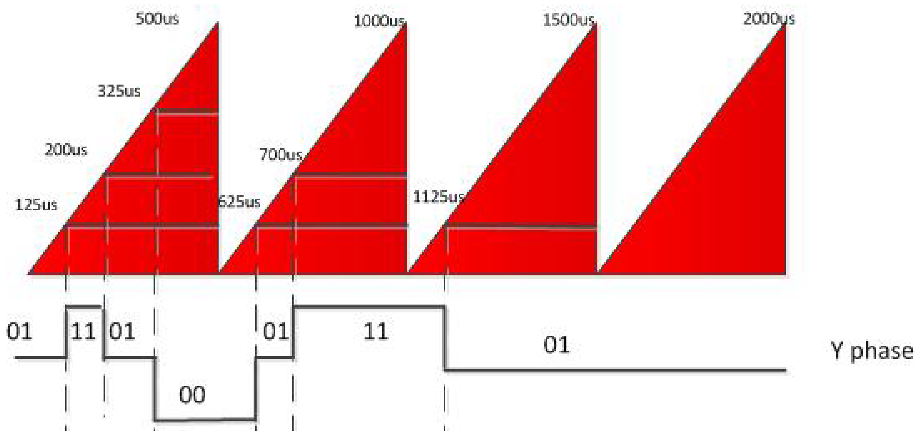


Figure 4-27 R phase.

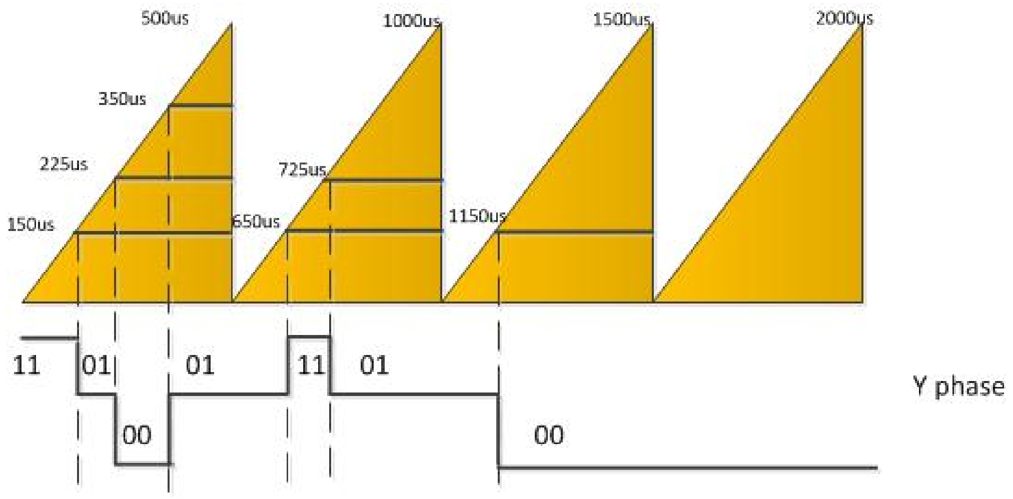


Figure 4-28 Y phase.

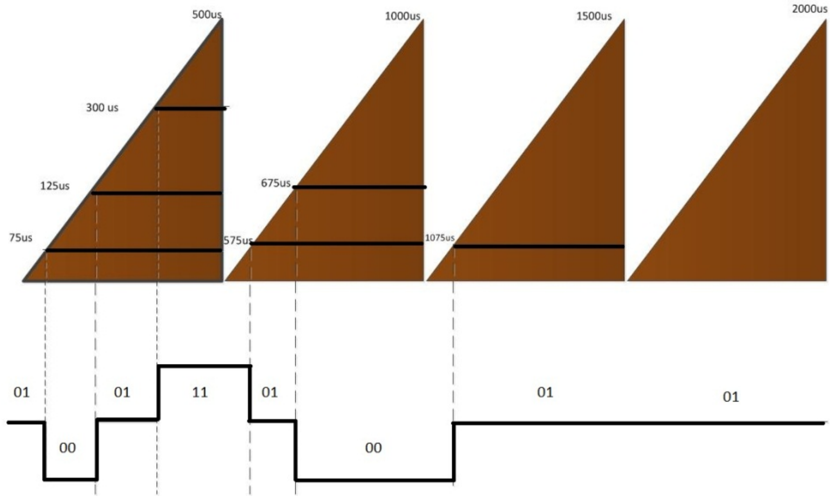


Figure 4-29 B phase.

This is SOM_PHASE_IP. Hence from the section 4.2.1 it is known that switching times and switching states of all the phases are calculated independently and put in the register separately. The values of R phase, Y phase and B phase are kept respectively in the register as shown in Table 4-1.

The switching time is converted into relative numbers.
Frequency of clock in FPGA =40 Mhz
Time period = 25 ns
Sampling rate frequency=2 Khz
Number of counter in one sampling period=40Mhz/2Khz =20000

Hence 20000 counting number=500μs

The sampling period is taken to be 500μs or relatively presented by number 20,000. The time must be converted to relative number and placed in register along with its switching state as shown in Table 4-1.

Table 4-1 Placement of Switching Time and State by Software

Number of sample	Phase where switching occurs	Switching Time μs	Switching Time numbers	Switching State
Sample1	R	125	5000	11
	R	200	8000	01
	R	325	13000	00
	Y	150	6000	01
	Y	225	9000	00
	Y	350	14000	01
	B	75	3000	00
	B	175	7000	01
	B	300	12000	11
Sample2	R	125	5000	01
	R	200	8000	11
	Y	150	6000	11
	Y	225	9000	01
	B	75	3000	01
	B	175	7000	00
Sample3	R	125	5000	01
	Y	150	6000	00
	B	75	3000	01
Sample4	R			01
	Y			00
	B			01

There is one simulation tool called ISIM in Xilinx tool which displays the result before it is downloaded into the FPGA card. Register stored as in Table 4-1 is the input to the SOM_PHASE_IP. This IP takes the input as the value stored in register and does its function and the result is shown as six bit stream as shown in Figure 4-30. The bit5 and bit 4 shows the state of R phase, bit 3 and bit 2 shows the state of Y phase while bit 1 and bit 0 shows the state of

B phase. These states are taken to the inverter for switching to the desired state at the desired time.

The simulation is done in Xilinx project Navigator by ISIM tool.

The simulation frequency is 500 Ghz. (i.e T=2 ps).

The upcounter counts from 0 to 20,000.

For FPGA, frequency of clock=40 Mhz i.e time period=25 ns

Therefore, 20000=20000*25 ns=500 us is the Sampling time when implementing on FPGA

But, For simulation, frequency of clock=500 Ghz i.e time period= 2 ps

Therefore, 20000= 20000*2 ps=40ns is the sampling time when simulation in ISIM.

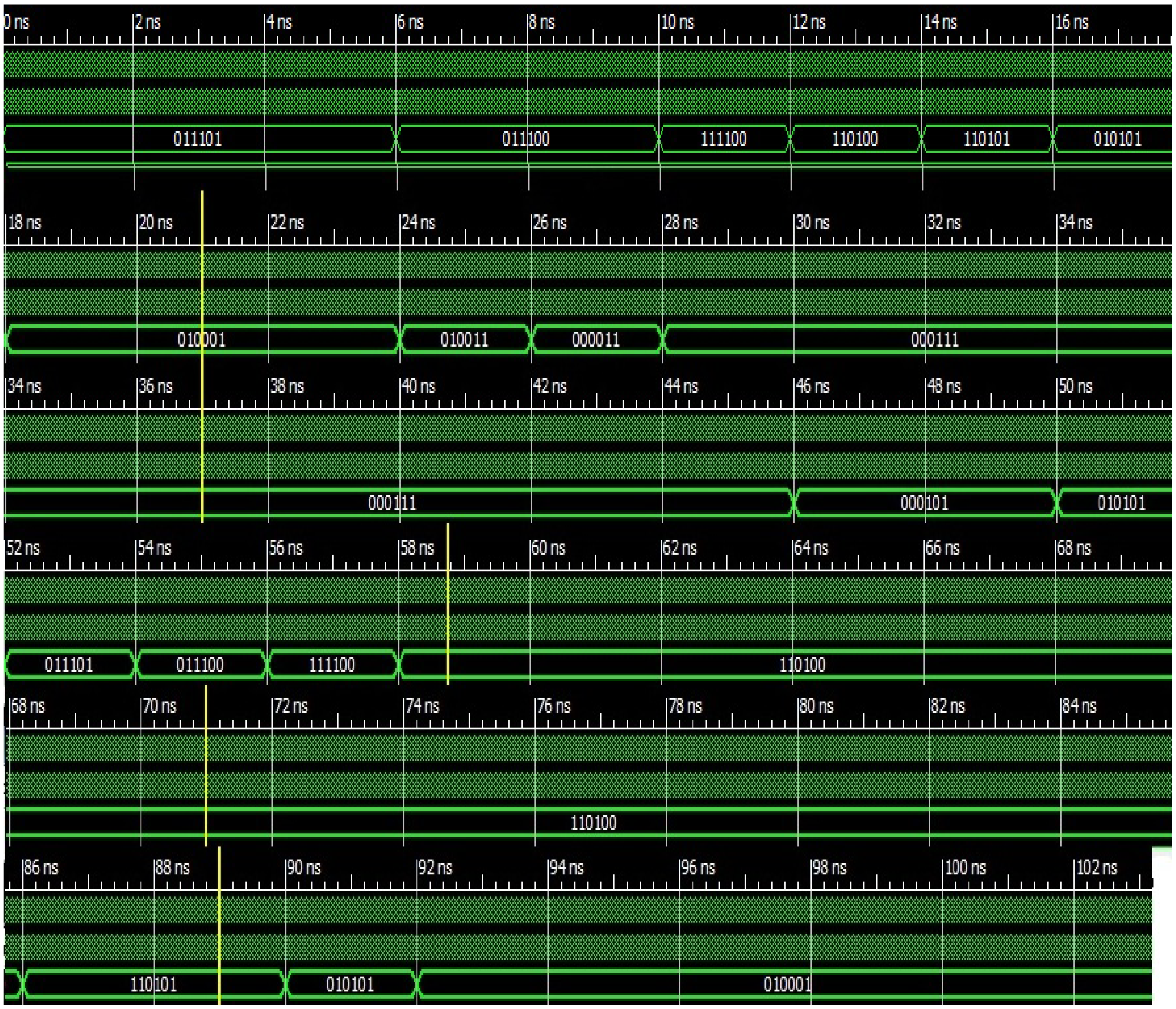


Figure 4-30 Output of SOM_PHASE_IP in ISIM.

The initial values of the states are 011101. From the section 4.2.2 it is known that the switching is done according to the ascending order of switching time in SOM_PHASE_IP. Hence from the register in Table 4-1, the desired switching time for desired phase to the desired state are given in Table 4-2. In the same table the result of the simulation is also tabulated from the Figure 4-30. And the obtained switching time match with the desired switching time. The required phase is switched to required switching state which is illustrated Figure 4-30 and Table 4-2. The phase in which switching occurs is shown in bold letter in Table 4-2 . The remaining two phases continue its previous state.

Table 4-2 result of the simulation.

Number of sample	Desired Switching time in FPGA (with 500µs sampling time) µs	Sampling period Number	Phase where switching occurs	Desired Switching time in ISIM (with 40ns as sampling time) ns	Desired Switching state for the phase	Simulation output in ISIM Switching time ns	Obtained State in ISIM for all the phases		
							R	Y	B
Initial	0	0		0		0	01	11	01
Sample 1	75	3000	B	6	00	6	01	11	**00**
	125	5000	R	10	11	10	**11**	11	00
	150	6000	Y	12	01	12	11	**01**	00
	175	7000	B	14	01	14	11	01	**01**
	200	8000	R	16	01	16	**01**	01	01
	225	9000	Y	18	00	18	01	**00**	01
	300	12000	B	24	11	24	01	00	**11**
	325	13000	R	26	00	26	**00**	00	11
	350	14000	Y	28	01	28	00	**01**	11
Sample 2	75+500=575	3000	B	46	01	46	00	01	**01**
	125+500=625	5000	R	50	01	50	**01**	01	01
	150+500=650	6000	Y	52	11	52	01	**11**	01
	175+500=675	7000	B	54	00	54	01	11	**00**
	200+500=700	8000	R	56	11	56	**11**	11	00
	225+500=725	9000	Y	58	01	58	11	**01**	00
Sample 3	75+1000=1075	3000	B	86	01	86	11	01	**01**
	125+1000=1125	5000	R	90	01	90	**01**	01	01
	150+1000=1150	6000	Y	92	00	92	01	**00**	01
Sample 4	0+1500=1500	0		120		120	01	00	01

After the successful simulation, experiment is carried out in actual FPGA control card as shown Figure 4-31 . The output is displayed in the oscilloscope. The switching states of three phases are displayed in six leds. Using digital probe, the datas in the leds are fed to the digital oscilloscope. The output of the digital oscilloscope is shown in Figure 4-32. There are 6 signals named as R1, R0, Y1, Y0, B1, B0 pointing the output in led5 to led0 respectively. In this experiment, the switching time between two states are observed from graphical way. For example as shown in Figure 4-32, the switching time between state 010101 and 011101 is 25μs as presented in upper right corner in the graph.

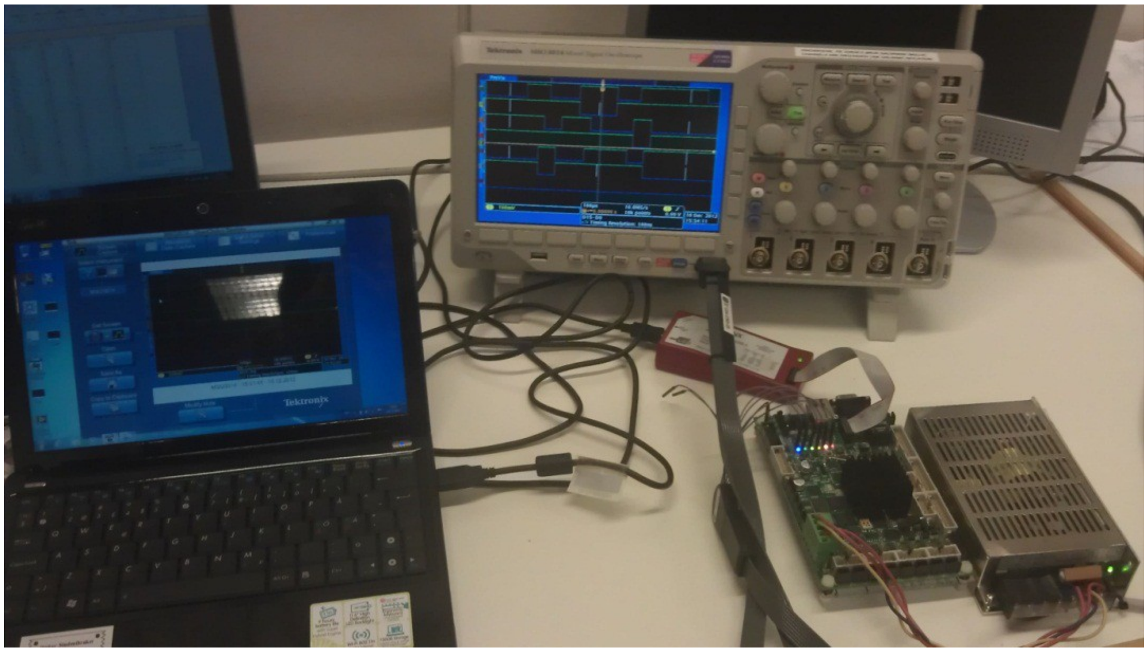


Figure 4-31 Testing in FPGA card.

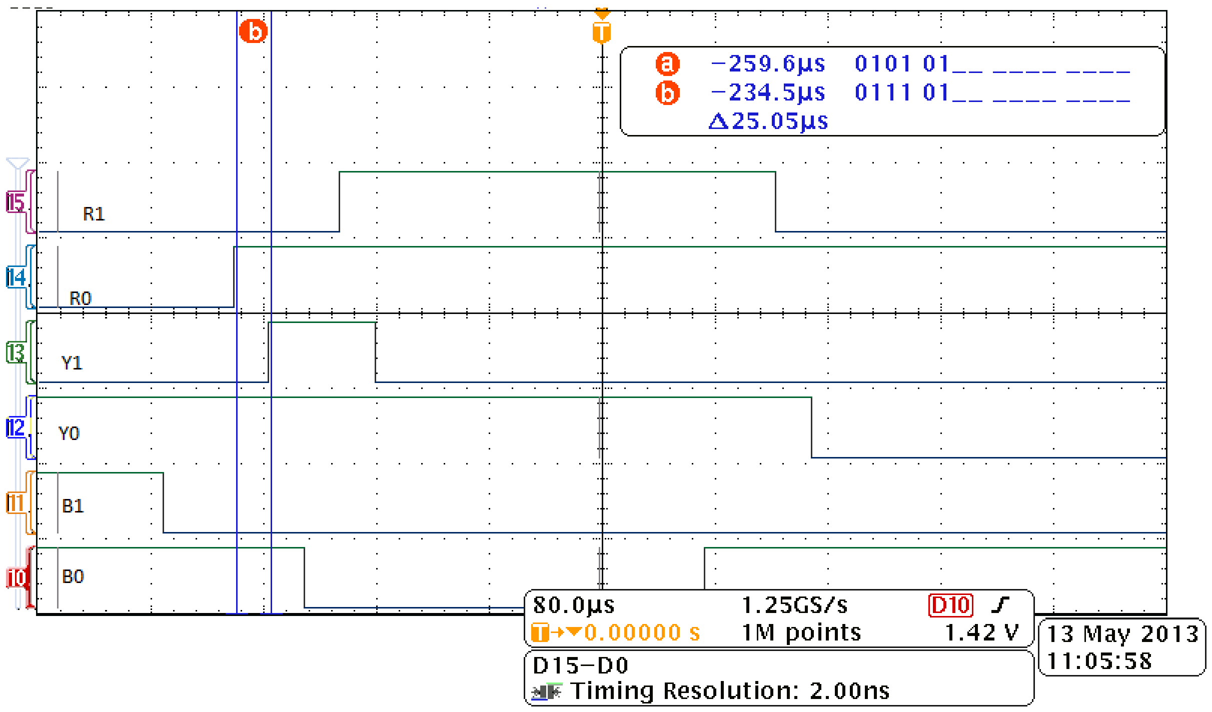


Figure 4-32 Graphical output of states in oscilloscope.

In First state 01**01**01 and second state 01**11**01, there is only change in state of Y phase. The time required to obtained 010101 state is 625us while the time required to obtained 011101 is 650us from Table 4-2. This means that the desired switching time between these two states is 25us. And the obtained value is also same.

The experiment is carried out to see the switching time between all states. The result of the experiment is presented in tabular form below:

Table 4-3 Test result.

State 1			Switching time 1 (µs)	State 2			Switching time 2 (µs)	Desired Time Difference (µs)	Observed time difference (µs)
R	Y	B		R	Y	B			
01	11	01	0	01	11	**00**	75	75	74
01	11	**00**	75	**11**	11	00	125	50	49.8
11	11	00	125	11	**01**	00	150	25	24
11	**01**	00	150	11	01	**01**	175	25	24.6
11	01	**01**	175	**01**	01	01	200	25	24.8
01	01	01	200	01	**00**	01	225	25	24
01	**00**	01	225	01	00	**11**	300	75	76
01	00	**11**	300	**00**	00	11	325	25	24
00	00	11	325	00	**01**	11	350	25	24
00	**01**	11	350	00	01	**01**	575	225	223
00	01	**01**	575	**01**	01	01	625	50	51
01	01	01	625	01	**11**	01	650	25	25
01	**11**	01	650	01	11	**00**	675	25	24
01	11	**00**	675	**11**	11	00	700	25	24.8
11	11	00	700	11	**01**	00	725	25	24.8
11	**01**	00	725	11	01	**01**	1075	350	349
11	01	**01**	1075	**01**	01	01	1125	50	51
01	01	01	1125	01	**00**	01	1150	25	24.8
01	**00**	01	1150	01	00	01	1500	350	351

It is seen from the table that observed time difference match with the desired time difference except for very small errors. This concludes that it is possible to switch to the desired state at the desired switching time for the all three phases. This verifies that codes written are valid for implementing synchronous optimal modulation through FPGA.

4.7.2 SOM_spacevector_IP

It is assumed that the input signal for SOM_spacevector_IP is also same as for SOM_PHASE_IP as shown as in Figure 4-26. They are also sampled as Figure 4-27, Figure 4-28 and Figure 4-29 According to SOM_spacevector_IP, all the phases are analysed together. Whenever there is switching in one of the phase, then states of all other phases must be placed in register. In this IP, software has to place switching time and states in the ascending order of time. Assuming Sampling period and all other condition same like in section 4.7.1, the input register to SOM_spacevector_IP is shown Table 4-4.

Table 4-4 Placement of state vector in SOM_spacevector_IP.

Number of sample	Switching_time us	Phase where switching occurs	State
Initial	0		0111 01
Sample 1	75	B	0111**00**
	125	R	**11**1100
	150	Y	11**01**00
	175	B	1101**01**
	200	R	**01**0101
	225	Y	01**00**01
	300	B	0100**11**
	325	R	**00**0011
	350	Y	00**01**11
Sample 2	75	B	0001**01**
	125	R	**01**0101
	150	Y	01**11**01
	175	B	0111**00**
	200	R	**11**1100
	225	Y	11**01**00
Sample 3	75	B	1101**01**
	125	R	**01**0101
	150	Y	01**00**01
Sample 4	0		010001

The simulation output in ISIM is exactly same as Figure 4-30 and Table 4-2. In addition the test is done on FPGA whose output is also same like Figure 4-32 and Table 4-3.This is expected because the input signals are same. Only, the way of giving input and methodology of functioning are different in SOM_PHASE_IP and SOM_spacevector_IP. If the input to both of them are same, the output is expected to be same.

From these test it is proved FPGA is able to switch to desired state in desired time. It can be exported to the other collaborating project for generating synchronous optimal modulation pattern. Due to the time constraint for the collaborating project only SOM_PHASE_IP is used by it.

Chapter 5

5 Experiment Setup

This thesis also consists of laboratory works for verification of theories. This chapter discuss about the experimental setup of the project. The high level schematic laboratory setup is shown along with the software development and real time interaction tool.

5.1 Hardware Setup

The experiment setup consists of hardware and software components. The hardware component is described in this section.

5.1.1 Setup Overview

This thesis is meant for the multiphase machine. Hence Six Phase Induction Motor(SPIM) is used. As shown in section 2.2, SPIM consists of two set of three phase winding separated by some angle. However in this thesis only one set of three phase winding is used as shown in Figure 5-1. The load for the motor is DC machine connected to load resistors. SPIM is supplied by three level, three phase inverter which is modulated by modulating circuit embedded in FPGA board. FPGA is controlled by PC.

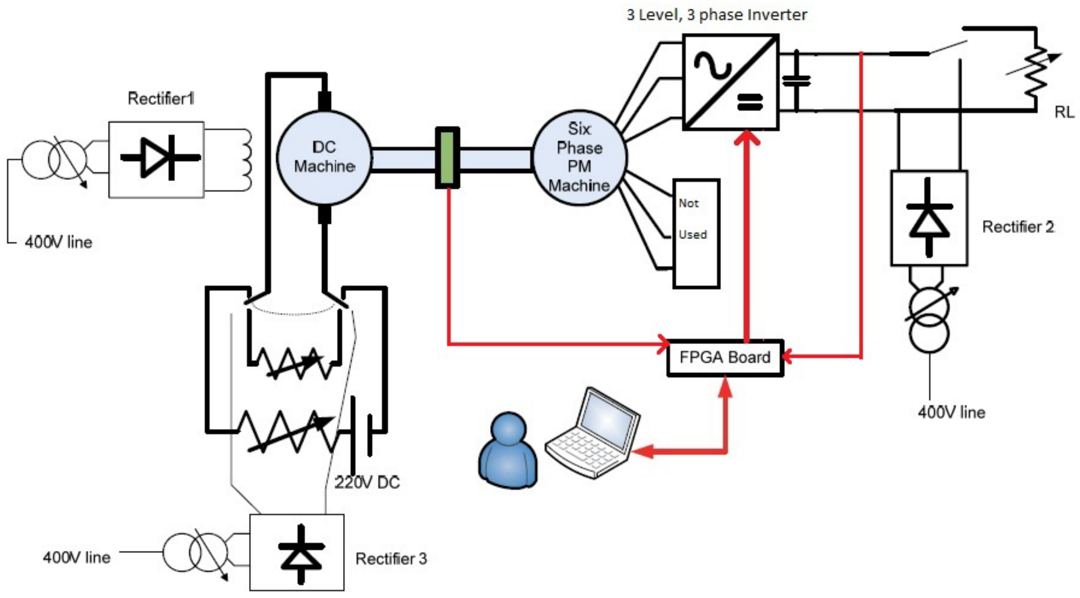


Figure 5-1 Overview of experimental setup.

The DC link voltage is obtained by rectifier. The AC input to the rectifiers can be varied using the three phase auto transformer which helps to vary DC link voltage as required. Real time monitoring is also done using software in PC. The assembled drive is shown in Figure 5-2.

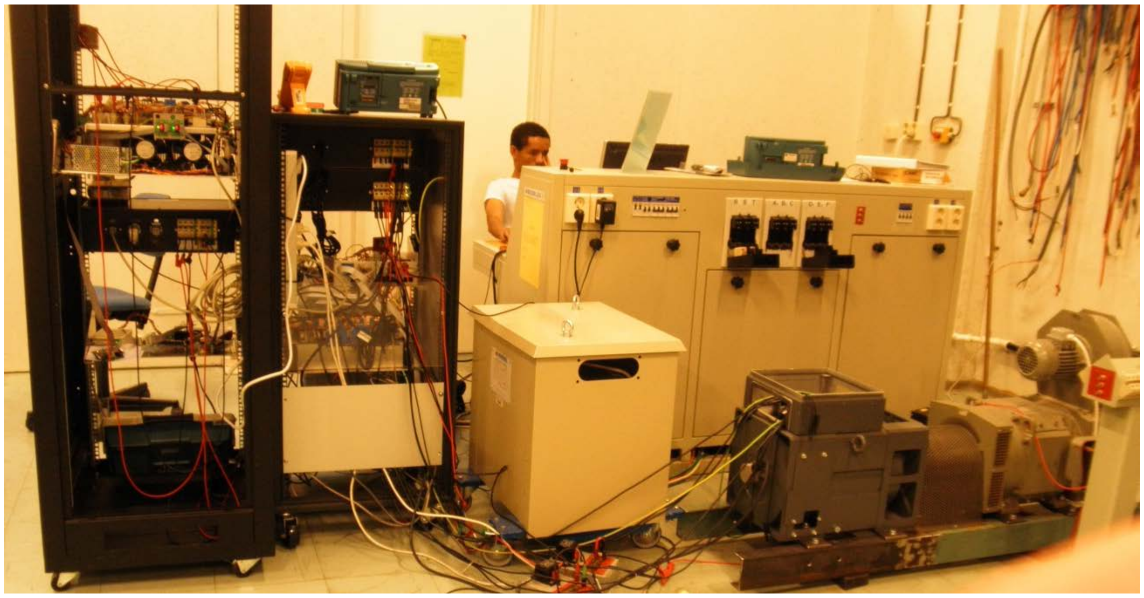


Figure 5-2 Assembly of Six phase machine drives.

5.1.2 Six-Phase Induction Machine

The six-phase induction machine used in the experiment has two 3-phase stator winding groups, separated by 30 electrical degrees in space. The machine has a squirrel-cage rotor. Figure 5-3 and Figure 5-4 shows the external view and terminals of the machine.

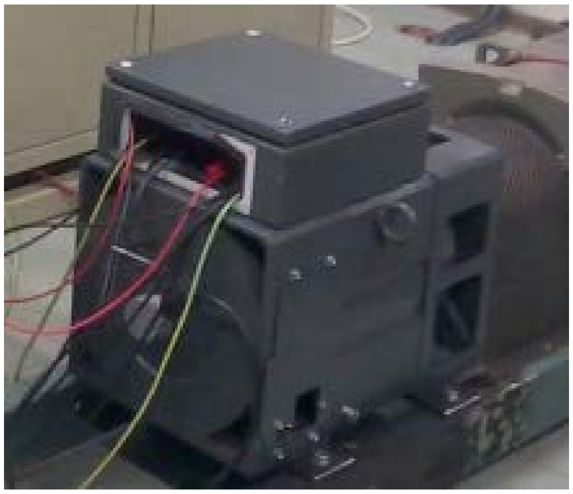


Figure 5-3 Six Phase Induction machine (external overview).

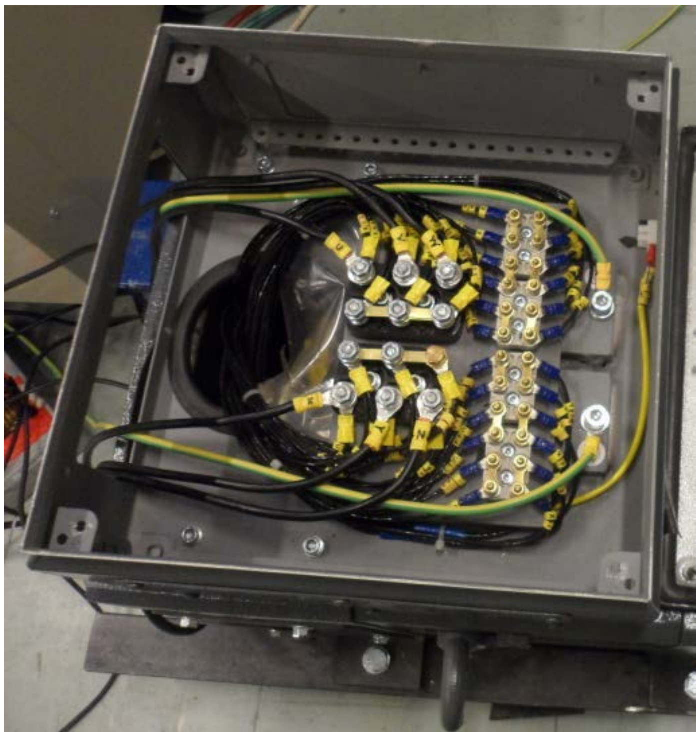


Figure 5-4 Six Phase Induction Machine, stator terminal.

The rated power output of the machine is 11.7 kW. The nameplate data and parameters of the machine are given in Appendix A.

5.1.3 DC Machine

DC machine is separately excited. It is mounted on the same shaft as six phase induction machine as load. The armature of the DC machine is connected to load resistors. The load can be varied either by changing the value of the load resistors or by varying the field voltage (excitation) of the DC machine. The six-phase induction machine is operated as a motor, and acts as prime mover as viewed from the DC machine.

5.1.4 Converter

Three-phase diode rectifiers are used to convert the AC line voltage into DC. The rectifiers can take voltage input of 0 – 400 V rms. The current rating is 63 A. The DC voltage output range is 0 – 540 V. The front view of one rectifier is shown in Figure 5-5.

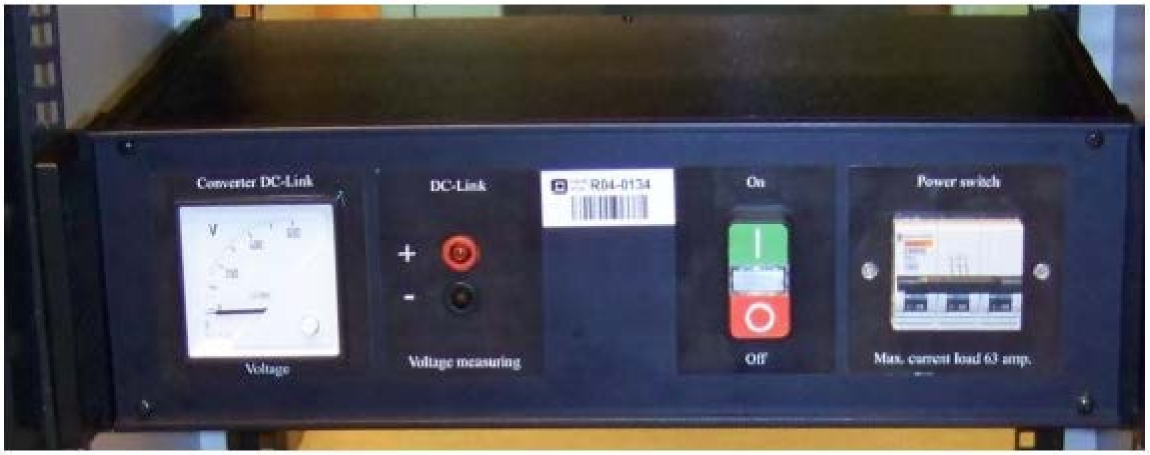


Figure 5-5 Rectifier used in the lab.

Three phase three level Neutral Point Clamped Inverter is used as inverter to supply Motor. The switch used in the inverter is IGBTs. The inverter is shown in Figure 5-6. This NPC is rated to be 40KW with dc link voltage of 2x315V. IGBTs are 1200V, 200A rated.

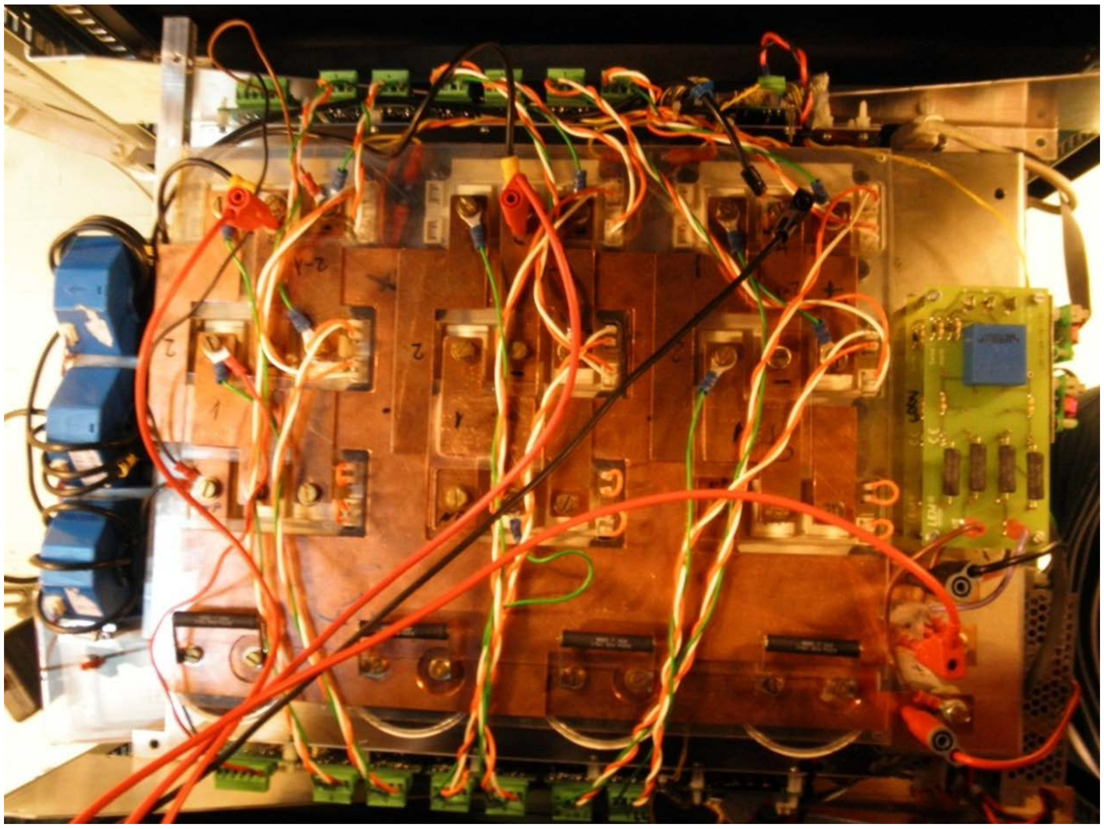


Figure 5-6 Three Level Inverter.

5.1.5 FPGA control card

FPGA used for this project is Xilinx Virtex FX30T. The control card outline with the chip and other important peripherals are as shown in Figure 3-4. The modulator signal is designed in FPGA control card. The use of FPGA in this thesis is shown in Figure 5-7.

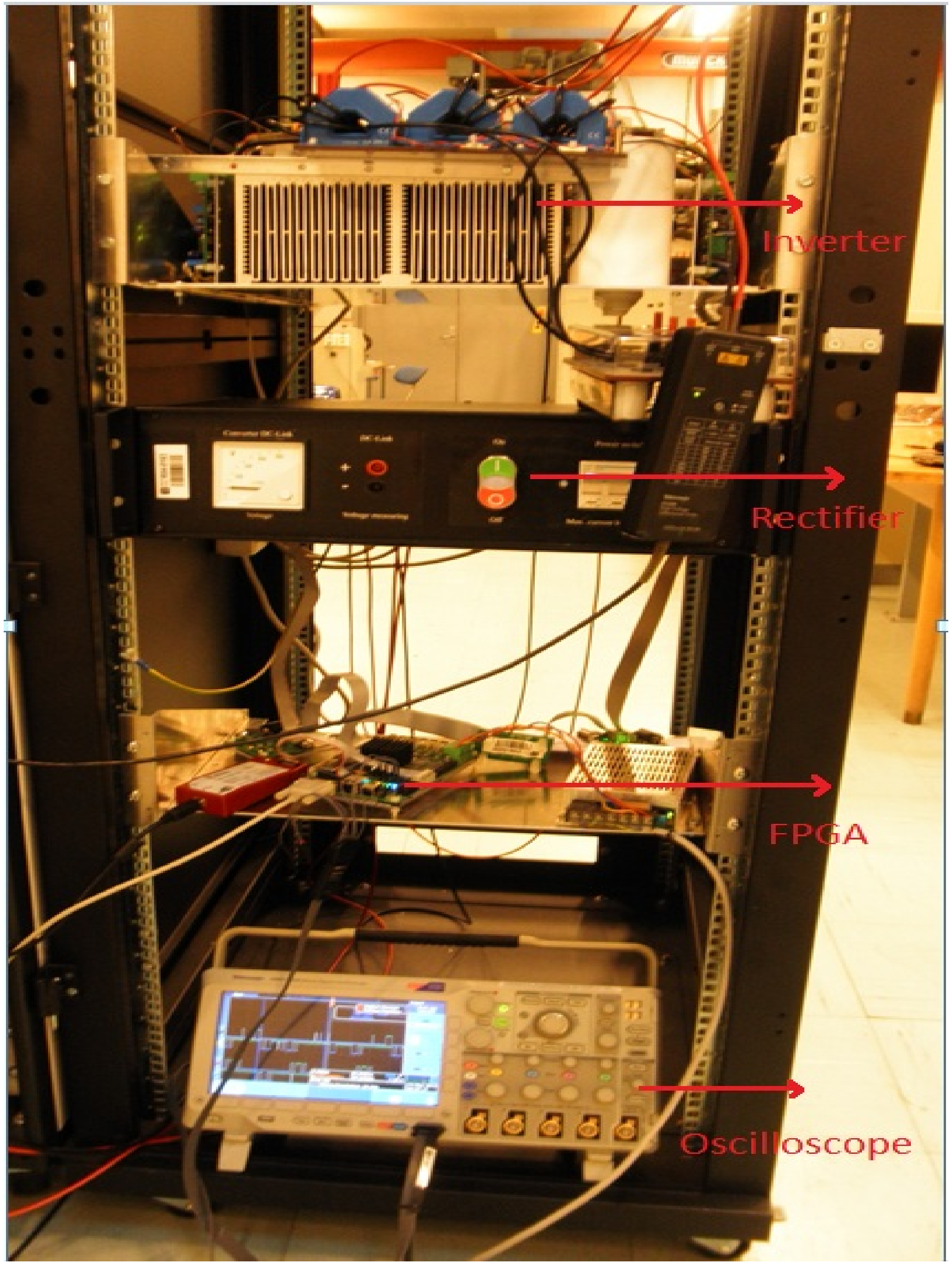


Figure 5-7 Use of FPGA in experiment.

5.1.6 Current Measurements

LEM Current Transducer LA 205-S. is used for stator current measurement. It has a primary nominal rms current of 200 A and current transformation ratio is 1:2000. The secondary nominal current is 100mA.

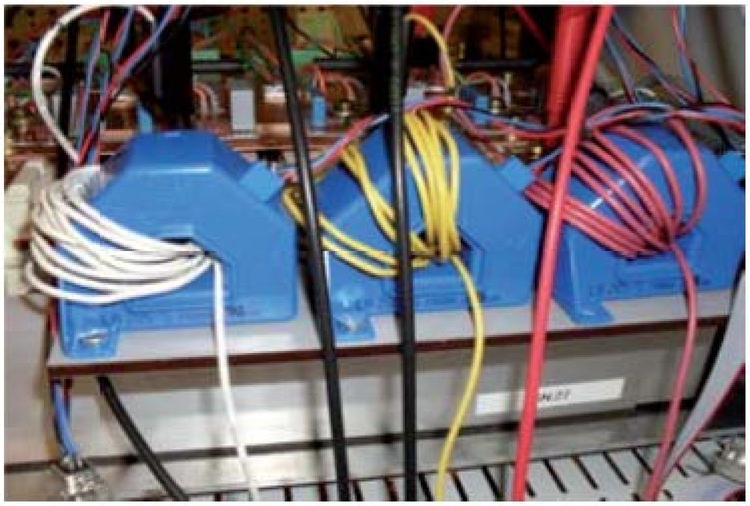


Figure 5-8 Current sensor.

5.2 Software Implementation

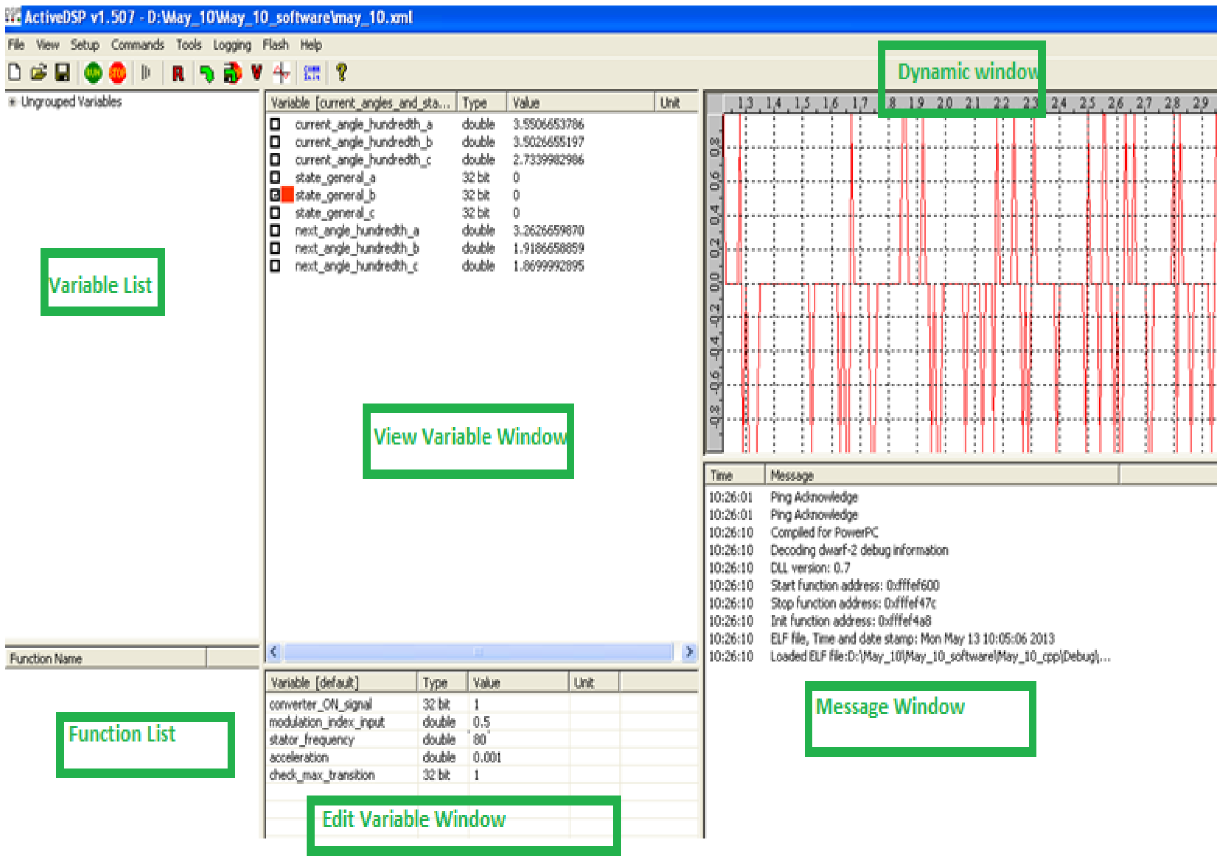
In this section, softwares used to control the motor drives are described.

5.2.1 Software Environment

The Xilinx Software Development Kit (SDK) provides an environment for creating software platforms and applications targeted for Xilinx embedded processors. SDK features include feature-rich C/C++ code editor and compilation environment. The SDK windows looks as shown in Figure 3-11.

The processes which are relatively slow and which are seeking complex design are programmed in software. And faster processes are programmed in hardware using IP module which are explained in section 3.3. The control code together with the library of IP modules are programmed from the PC into the FPGA using a USB connection. The modulating signal for the inverter is generated by FPGA.

Real time monitoring of the drive system can be done using ActiveDSP software on the PC as shown in Figure 5-9. The communication is done through the RS232 serial cable connected to the computer. Two way communications are possible using Active DSP. It can give input to and output from FPGA. The software enables the real-time logging of data. This data can be saved and processed using other software tools such as Matlab.



Variable [current_angles_and_sta...	Type	Value	Unit
current_angle_hundredth_a	double	3.5506653786	
current_angle_hundredth_b	double	3.5026655197	
current_angle_hundredth_c	double	2.7339982986	
state_general_a	32 bit	0	
state_general_b	32 bit	0	
state_general_c	32 bit	0	
next_angle_hundredth_a	double	3.2626659870	
next_angle_hundredth_b	double	1.9186658859	
next_angle_hundredth_c	double	1.8699992895	

Variable [default]	Type	Value	Unit
converter_ON_signal	32 bit	1	
modulation_index_input	double	0.5	
stator_frequency	double	80	
acceleration	double	0.001	
check_max_transition	32 bit	1	

```
Time      Message
10:26:01  Ping Acknowledge
10:26:01  Ping Acknowledge
10:26:10  Compiled for PowerPC
10:26:10  Decoding dwarf-2 debug information
10:26:10  DLL version: 0.7
10:26:10  Start function address: 0xfffef600
10:26:10  Stop function address: 0xfffef47c
10:26:10  Init function address: 0xfffef4a8
10:26:10  ELF file, Time and date stamp: Mon May 13 10:05:06 2013
10:26:10  Loaded ELF file:D:\May_10\May_10_software\May_10_cpp\Debug\...
```

Figure 5-9 Active DSP for real time monitoring.

5.2.2 Program Structure

The program flow and interrupt routines of the drive control system are shown in Figure 5-10. The start routine initializes the system and sets up the interrupt mechanisms. The service routine consists of the main function of program.

From section 4.5 it is known that the calculation of switching times and states for N th sample is done in N-1 th sample. As the interrupt comes from the hardware for N th sample, the first task of the software would be to place those switching time and states of N th Sample to the register so that hardware can performs its task. Then after placing them in the register, software calculates the switching times and states for N+1 th sample. Then program returns to Background routine and checks if the program is running or not. If it is running the process continues. If not then stop routine is executed which will stop PWM interrupt, timer interrupts etc.

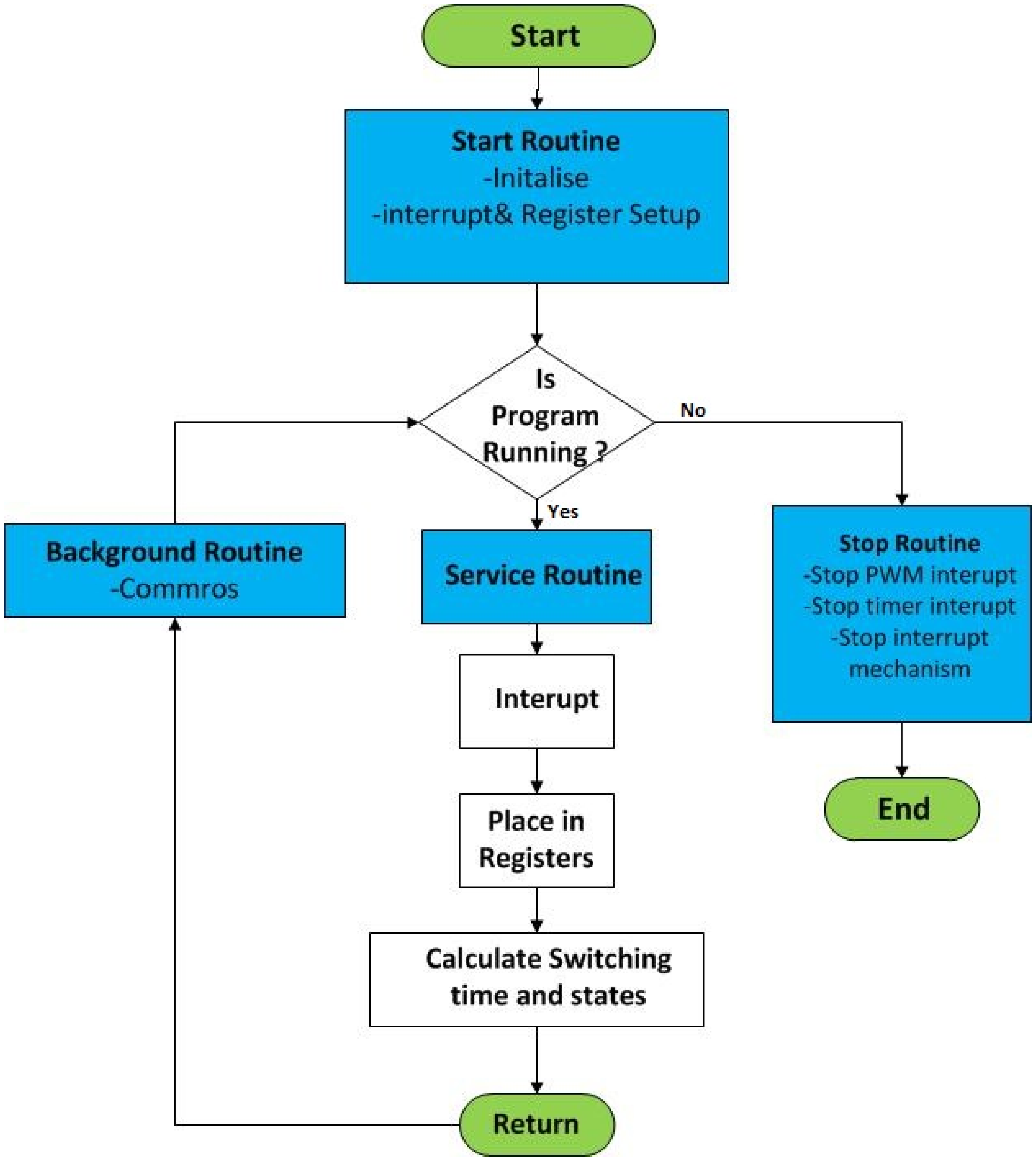


Figure 5-10 Program flow and interrupt.

Chapter 6

6 Experimental Results and Discussion

Processor routine generates the SOM pattern as indicated by chapter 2. It generates optimal angles and states acquired at that time and place in register. On the other hand, FPGA switch the IGBT to the given switching states at the given optimal angles.

6.1 Driver, Voltage and Current signal

6.1.1 Driver signal

From section 2.6, it is known that for particular value of amplitude modulation index (m) and number of pulse per half cycle (N), there is algorithm which gives the optimal angles to minimize the total harmonic distortion. These optimal angles are generated in processor routine which place these optimal angles and states in register for each sampling period. These register are read by FPGA and SOM patterns are generated as shown in figure 6-1. In the figure, 15th and 14th digital signals are the switching signals for R bridge leg, 13th and 12th gives the signals of Y bridge leg while 11th and 10th gives the switching signals for B bridge leg.

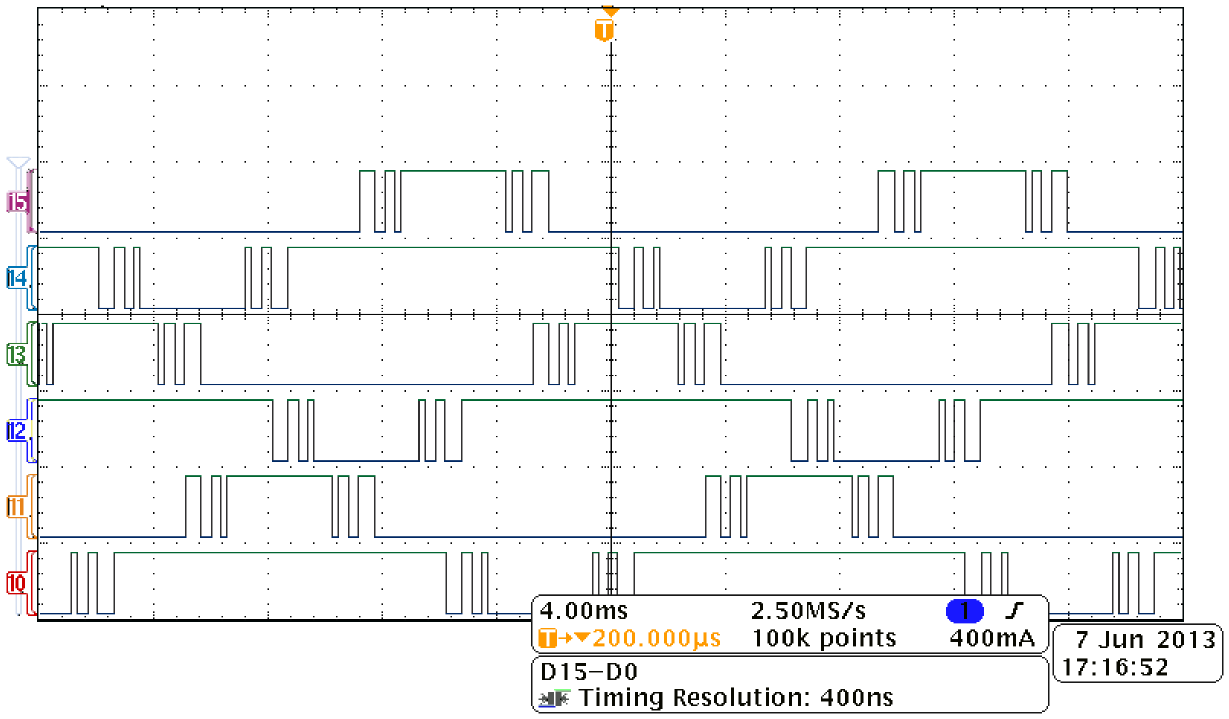


Figure 6-1 Driver Signals For Three phase, Three level Converter (for m=1, N=5).

Switching states are defined by two bits per phase. These two bits can be combined in four different ways among which only three of them are used. They are 11(+1), 01(0) and 00(-1) where 10 is the forbidden state. In this thesis three level modulator for three phase is designed in FPGA. These digital signals are sent to the 3 level, 3 phase converter(NPC inverter). As mentioned in section 2.7, switching waveform has quarter wave symmetry and half wave symmetry. This figure shows the switching pulse for three phases. These phases are 120^0 out of phase with each other. It is seen that number of Pulse per half wave (N) is 5.

6.1.2 Voltage waveform

The driver signals defined in section 6.1 are given to three phase, three level converter. The state represented by these driver signals are two bits. According to Figure 4-18 and Figure 4-20 the first bit is given to T1 while second bit is given to T2. The signals to T3 and T4 are complementary to T1 and T2 respectively. It has been mentioned many times that driver signal 11 gives +1(+Vd/2) analog state, 01 gives 0 (0 V) analog state while 00 gives -1(–Vd/2) analog state. Here Vd is dc link voltage which is given as 50V.

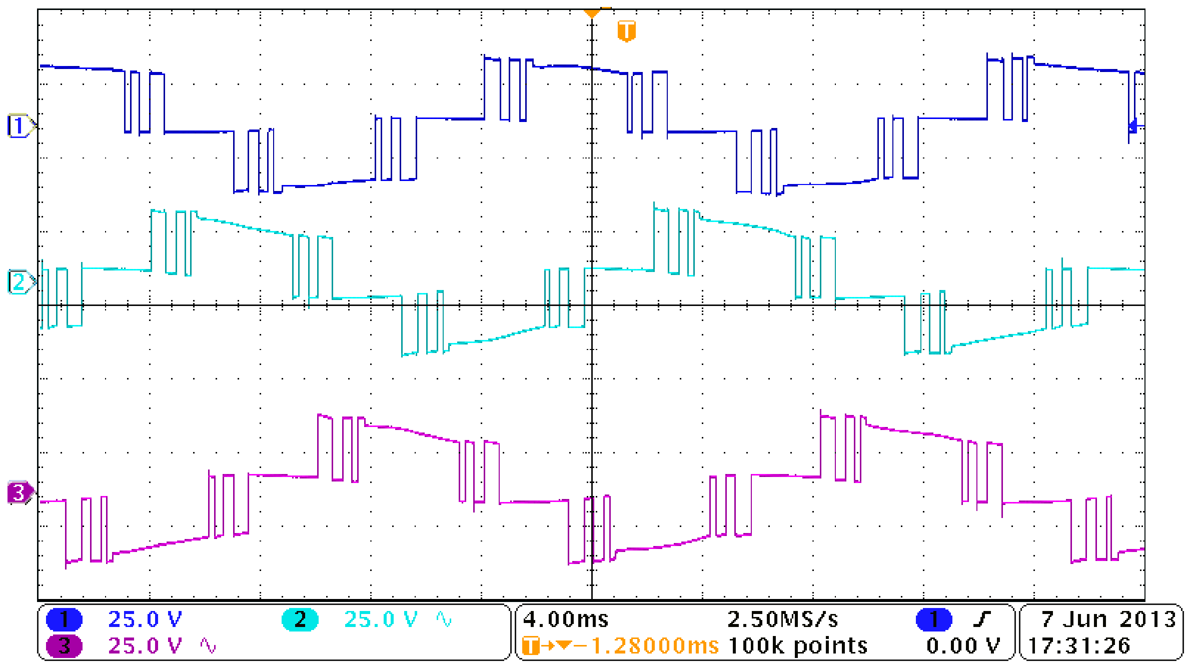


Figure 6-2 Voltage generated by Three phase, Three level Converter (for m=1, N=5).

Even though dc link voltage is given 50V in lab, this can be extended for medium voltage because IGBT with blocking voltage of 4.5KV is available. It is interesting to see if the driver signals generated by FPGA would be able to switch the IGBT in right manner. Digital signals generated by FPGA are given by Figure 6-1. This signal is given to three phase, three level converter. The output is shown in Figure 6-2. First waveform is for R phase and other two is for Y and B phase respectively. RYB phases are 120^0 out of phase with each other. It is seen that there are three voltage levels: +25V(+Vd/2), 0V(0V), -25V(-Vd/2). If these two figures are noticed carefully then it is seen than +Vd/2 is result of digital signal 11, 0V is result of digital signal 01 while –Vd/2 is formed by digital signal 00.

6.1.3 Current Waveform

The three phase, three level converter acts like voltage source converter and it is connected to induction machine whose rating is shown in Appendix A. The voltage generated by synchronous optimal modulation is shown in section 6.1.2. For the same modulation index and number of pulse per half cycle(m=1, N=5) the current waveform is inspected.

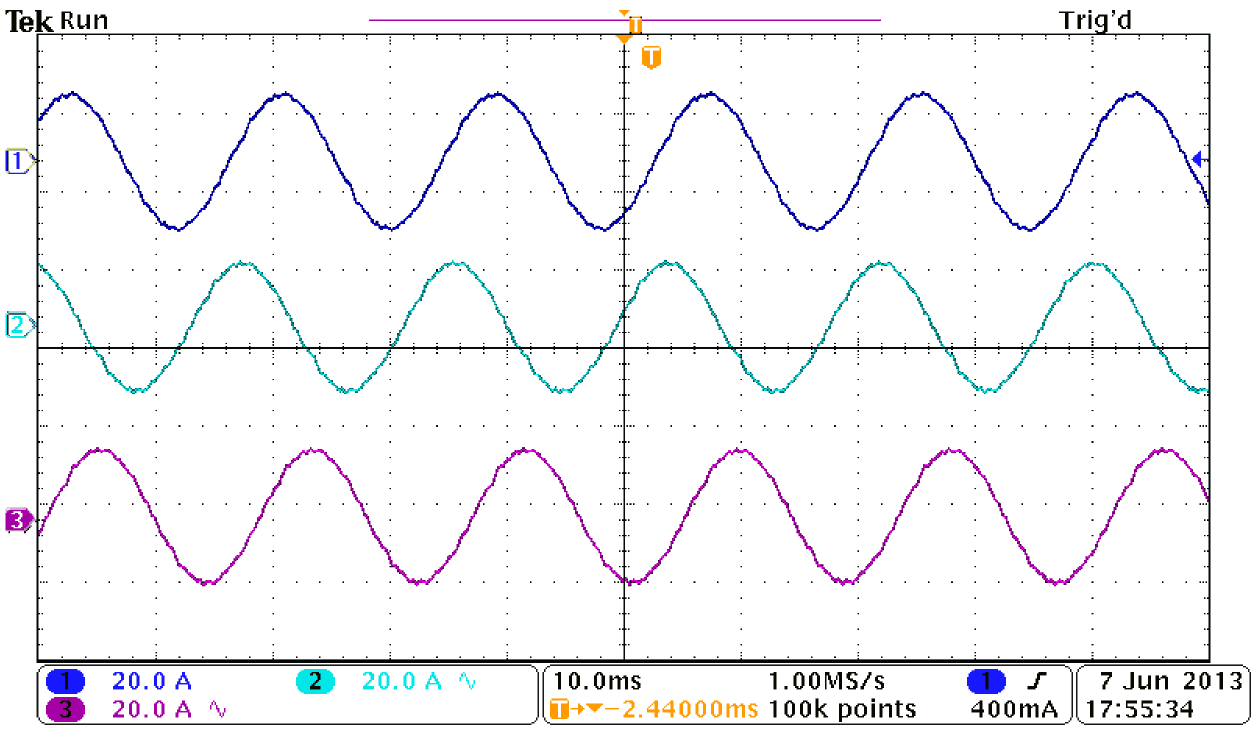


Figure 6-3 Current given to load (for m=1, N=5).

First waveform is for R phase and other two is for Y and B phase respectively. RYB phases are 120^0 out of phase with each other. The current waveform is nearly perfectly sinusoidal without ripples for this case. The driver signal, voltage signal and current signal for m=0.3,N=4 and m=0.87, N=3 are given in Appendix F and Appendix G respectively.

6.1.4 Digital, Voltage, Current signal

To summarize, per phase digital signal, voltage signal and current signal are shown in the same frame as shown in Figure 6-4 for m=1, N=5. The lower most waveform is digital signal which produce voltage signal in the middle portion. The voltage generated by inverter produce the current signal (upper most) to the induction machine. As expected, current is lagging voltage. The dc link voltage is given as 50V. It is seen very clearly here that 11 digital signal produce +Vd/2 (+25V), 01 produce 0V while 00 digital signal produce –Vd/2(-25V).

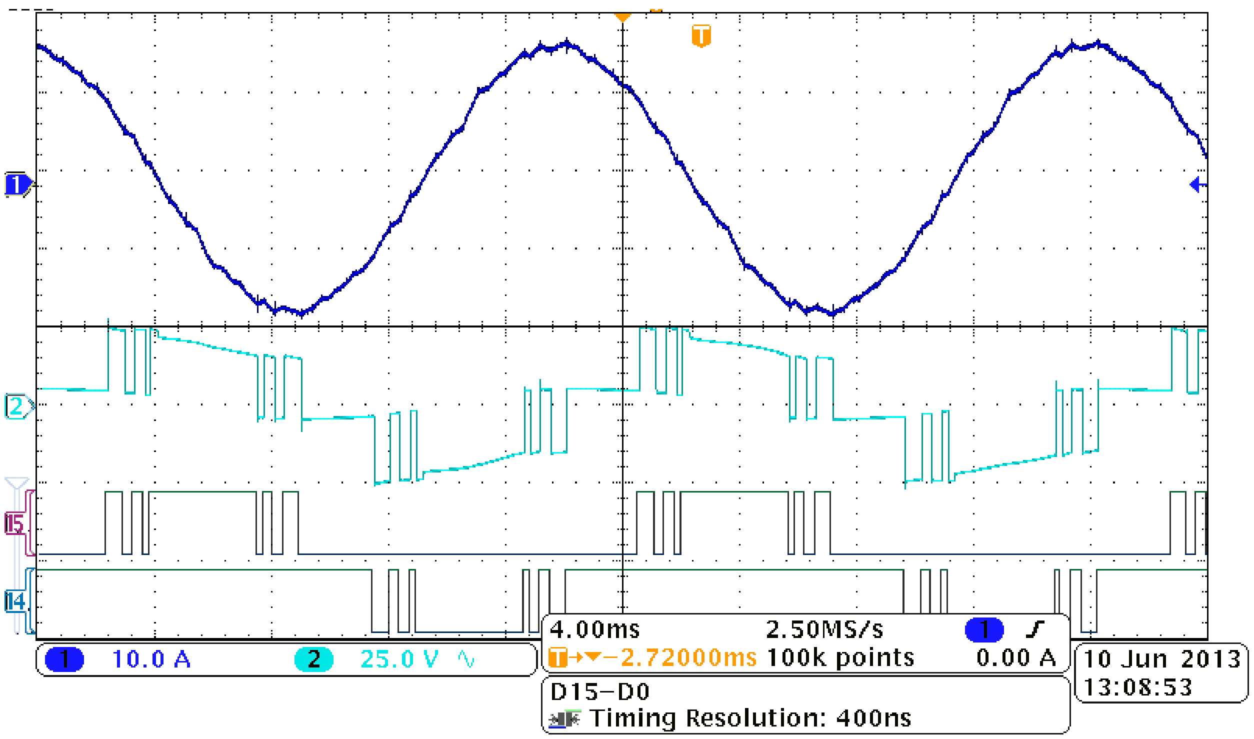


Figure 6-4 Digital, Voltage, Current signal per phase for m=1, N=5.

6.2 Variation of N with Stator frequency

Section2.5.2 also indicates that for constant N, switching frequency increase with increased stator frequency but when switching frequency is greater than 300Hz then N is decreased as stator frequency is increased so that the switching frequency of IGBTs does not exceed 300Hz. To verify this, experiment was done. From section 2.8 it is known that (m,N) are the input to the modulator, where N is decided by value of stator frequency of induction machine as shown in Figure 2-18. Induction machine is the load for voltage source converter. For one particular value of m, stator frequency is varied. Experiment is done to see if N is maintained with changed in stator frequency so that switching frequency does not exceed 300Hz.

$f_{sw1,avg} = N \cdot f_s$ where f_{sw} is switching frequency of IGBT and f_s is stator frequency.

The experiment was conducted for m=0.5 and frequency is varied. The following figures from Figure 6-5 to Figure 6-8 was obtained for frequency 45 Hz, 55Hz , 65Hz and 80 Hz respectively and result is tabulated in Table 6-1.

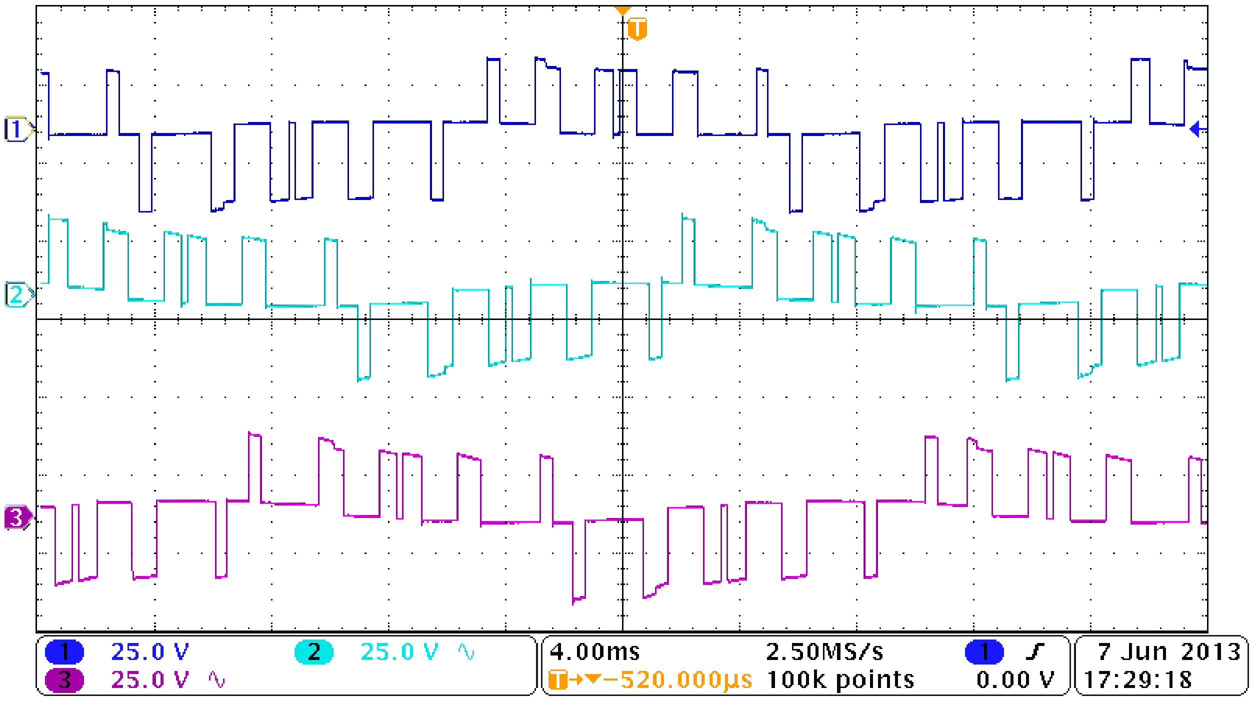


Figure 6-5 For m=0.5, fs=45Hz, N=6.

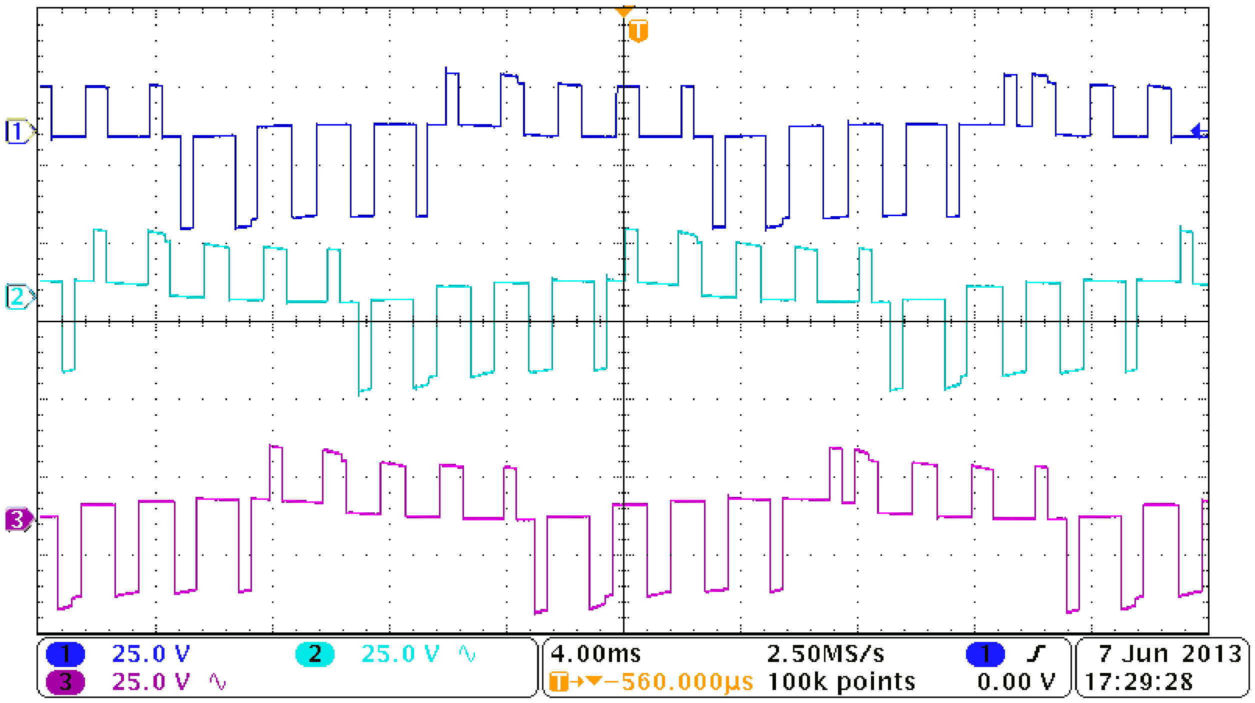


Figure 6-6 For m=0.5, fs=55Hz, N=5.

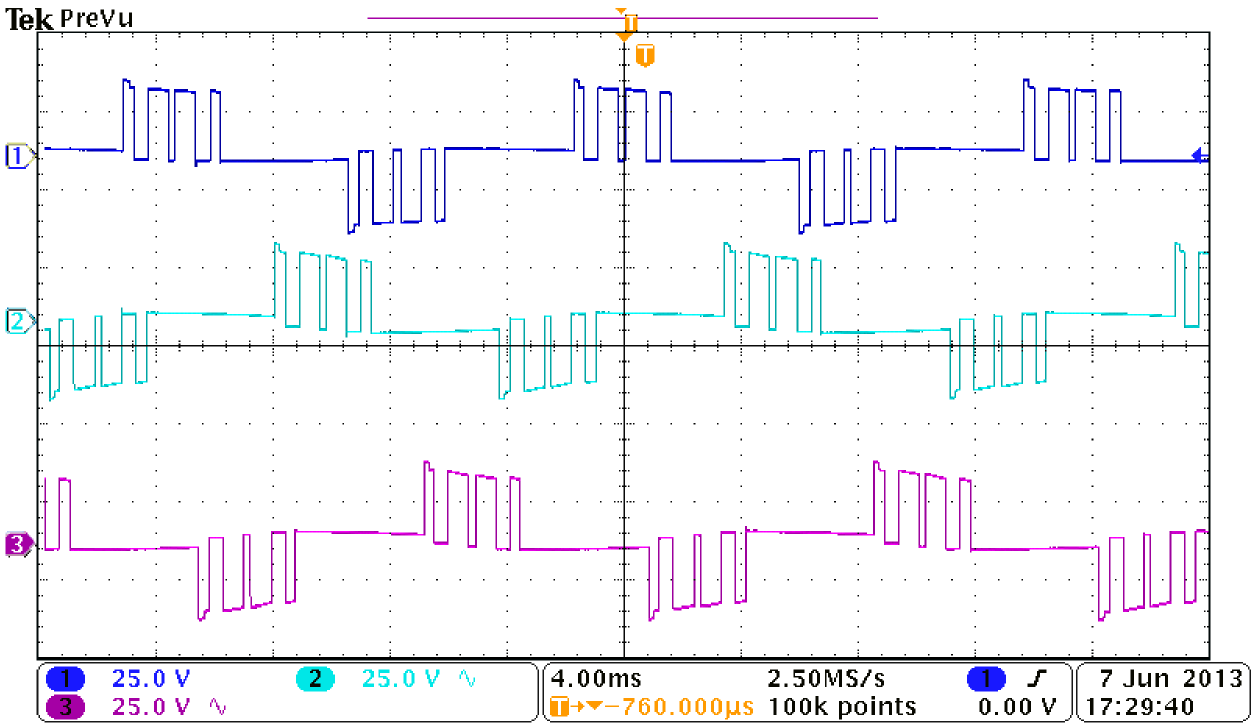


Figure 6-7 For m=0.5, fs=65Hz, N=4.

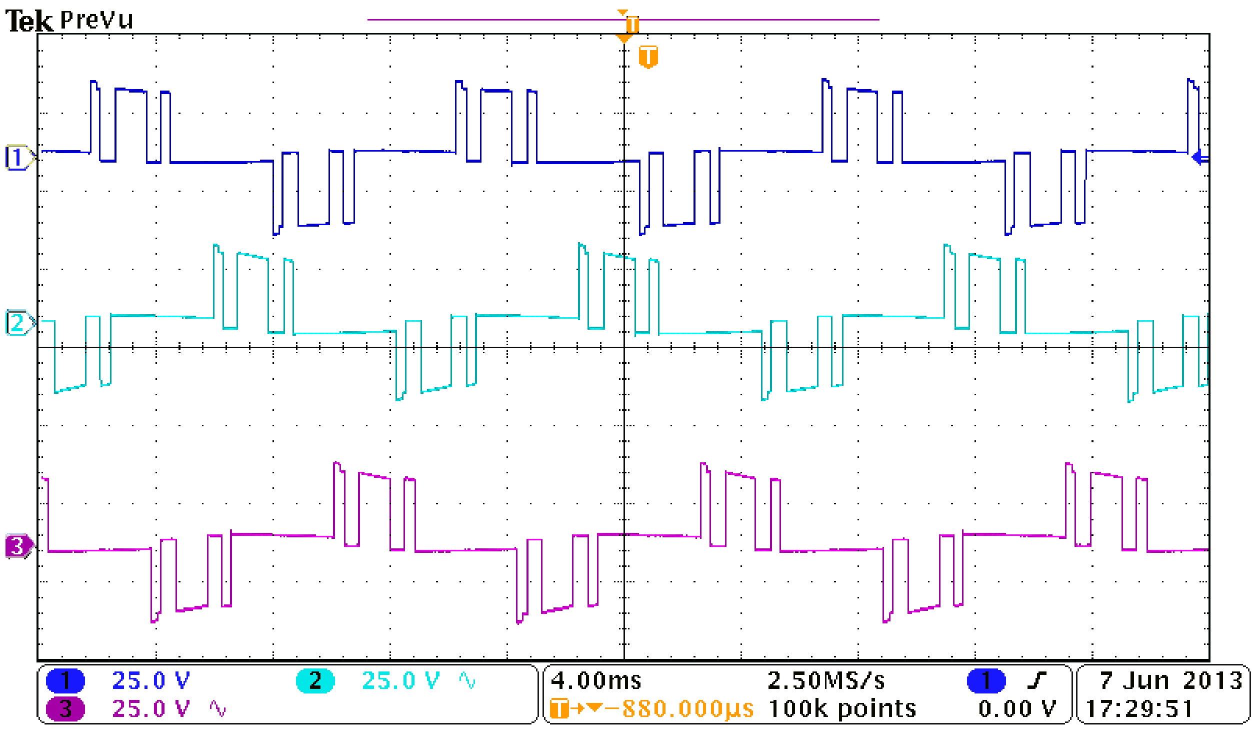


Figure 6-8 For m=0.5, fs=80Hz, N=3.

Table 6-1 variation of N for different fs (m=0.5).

fs (Hz)	N	fsw=N.fs (Hz)
45	6	270
55	5	275
65	4	260
80	3	240

The results verify that when stator frequency is increased, number of pulse per half period is decreased in order to maintain switching frequency of IGBT less than 300Hz.

It is noticed in Figure 6-6 that the positive pulse and negative pulse are not equal. This problem has occurred due to lack of DC bus balancing in this system. The three level converter consists of two capacitor which are not equally charged. It seems that lower capacitor is more charged than the upper capacitor because negative pulse is higher than positive. From section 2.3.3 the reason behind this imbalance is mentioned to be the generation of negative current from neutral point which is charging the lower capacitor and discharging the upper capacitor. Hence positive vector must be used which charge the upper capacitor and discharge the lower capacitor to balance voltage in two capacitor.

6.3 Total harmonic distortion

6.3.1 Line-Line Voltage

It is shown in Figure 2-9 that line to line voltage of three level converter has five different levels. They are +2(+Vd), +1(+Vd/2), 0,-1(-Vd/2),-2(-Vd). The dc link voltage (Vd) is given as 50V. The output of three level converter for this thesis for m=0.87, fs=45 is shown in Figure 6-9. It is also able to achieve five states as 50V, 25V, 0V, -25V, -50V. The three level converter feeds the induction motor.

From section 2.3.2, it is known that it is the harmonics of line-line voltage that would affect the machine. Even if the harmonics occurs in bridge leg voltage, they might not appear in line-line voltage and do not affect the machine. Since the line-line voltage has five levels, the harmonics are reduced significantly because the multilevel output voltage waveforms have less distorted output and are close to sinusoidal waveforms.

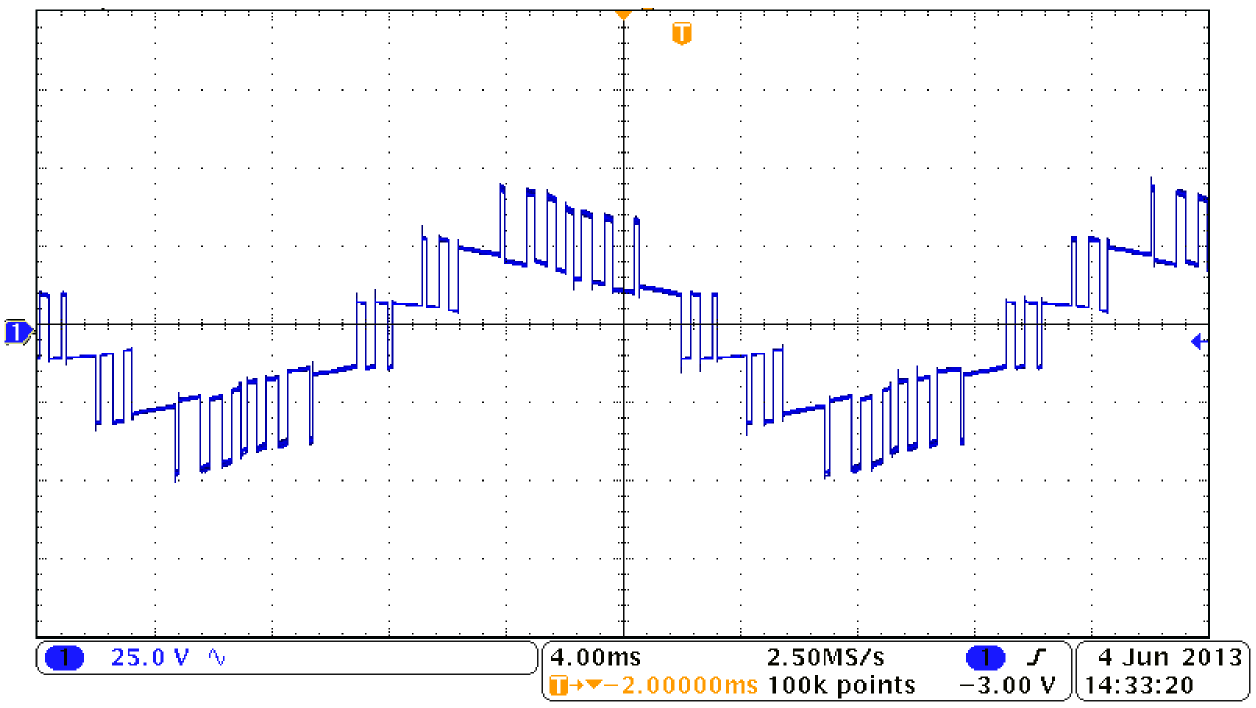


Figure 6-9 For m= 0.87, fs=45 Hz.

6.3.2 Measurement of THD

The total harmonic distortion for line-line voltage and line current are analyzed at 45 Hz, 50 Hz and 55 Hz by increasing the amplitude modulation index. The result of THD for line-line voltage is tabulated in Table 6-2. THD is taken as percentage of DC link voltage. The DC link voltage is 50V.

Table 6-2 THD (Line Voltage) for different modulation index (m).

m	THD(%) fs=45 Hz	THD(%) fs=50 Hz	THD(%) fs=55 Hz
0.3	42	40.95	44.25
0.5	38.3	44.2	43
0.8	29.84	26.96	26.8
1	27.7	29.96	29.96

THD for line-line voltage is analysed by using Power analyser The example is shown for m=1, f=45Hz in Figure 6-10. The result of THD for line current is tabulated in Table 6-3. THD is taken as percentage of fundamental current component at m=1 for particular frequency.

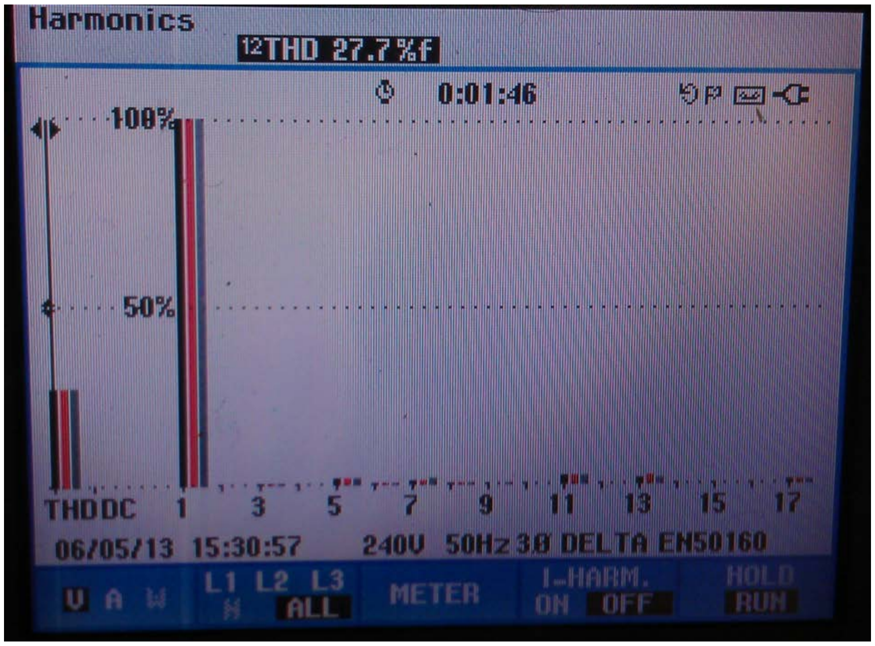


Figure 6-10 THD for line voltage for m=1, 45 Hz.

Table 6-3 THD(Line current) for different modulation index (m).

m	THD(%) fs=45 Hz	THD(%) fs=50 Hz	THD(%) fs=55 Hz
0.3	3.9	5.9	4.59
0.5	4.4	6.2	6.2
0.8	2.24	2	1.9
1	1.9	2.5	2.5

THD for line current is analysed by using Power analyser The example is shown for m=1, f=50Hz in Figure 6-11.

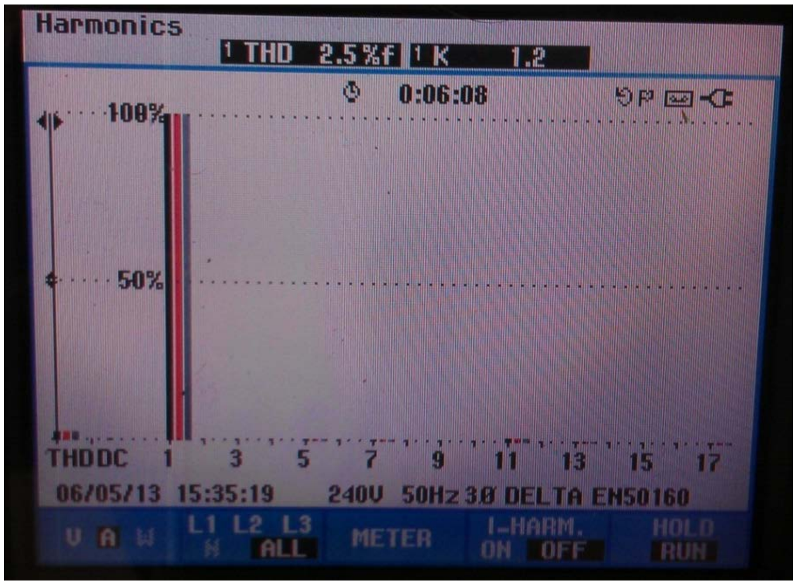


Figure 6-11 THD for line current for m=1, 50 Hz.

6.3.3 Analysis

It is seen from Table 6-2 and Table 6-3 that lower modulation index has higher total harmonic distortion. Normally machines do not perform smoothly due to large amount of %THD, which causes noise, vibration and heating in machines [44]. For analyzing, per phase voltage and current waveform is inspected at different modulation index for particular frequency. The following four figures from Figure 6-12 to Figure 6-15 shows per phase voltage and current for different modulation index for stator frequency 45Hz. In all these figures, upper figure indicates the voltage waveform and lower indicate the current waveform for one phase.

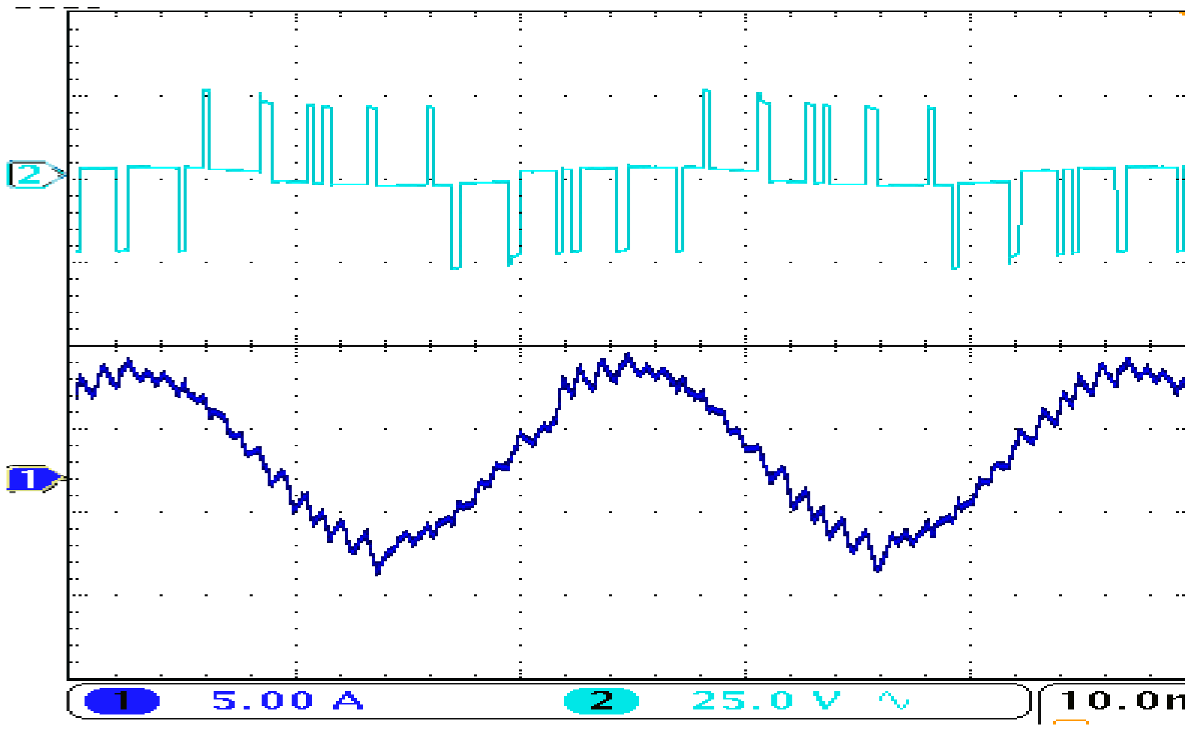


Figure 6-12 Current and Voltage for m=0.3, f=45Hz, N=6.

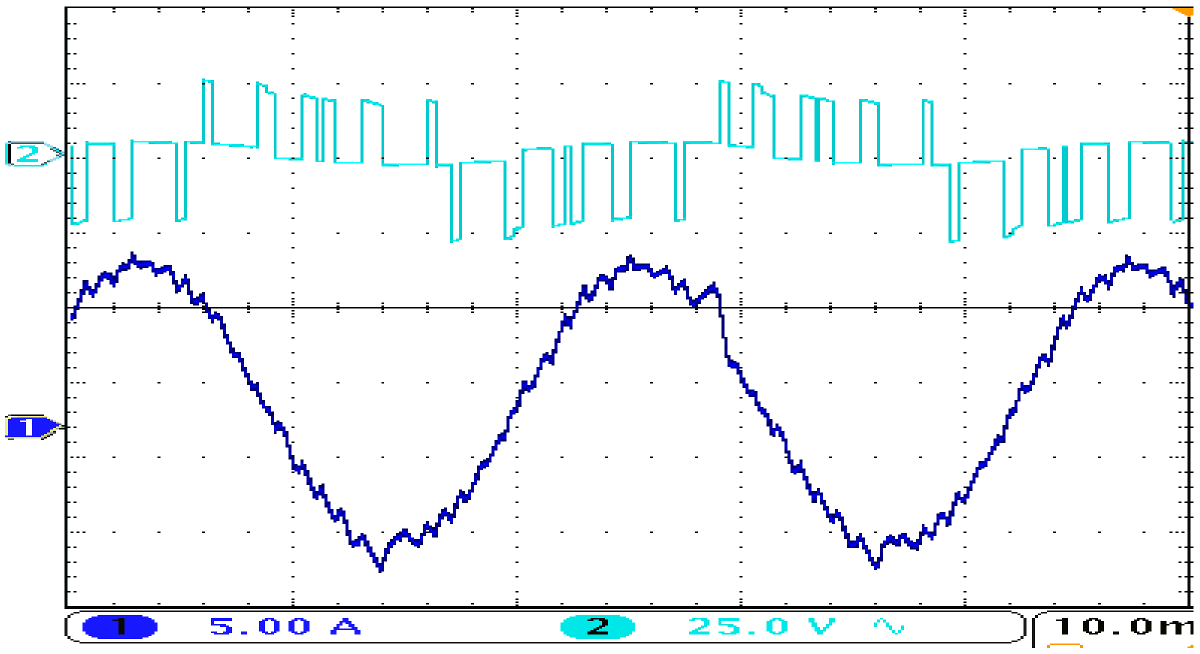


Figure 6-13 Current and Voltage for m=0.5, f=45Hz, N=6.

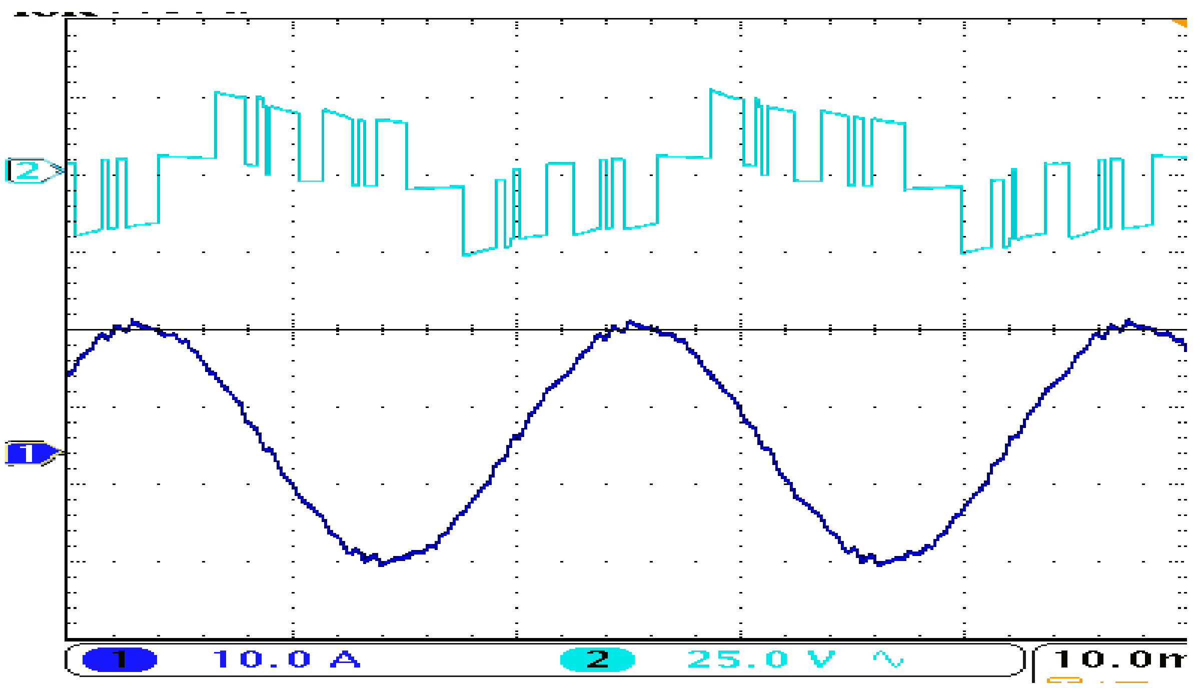


Figure 6-14 Current and Voltage for m=0.8, f=45Hz, N=6.

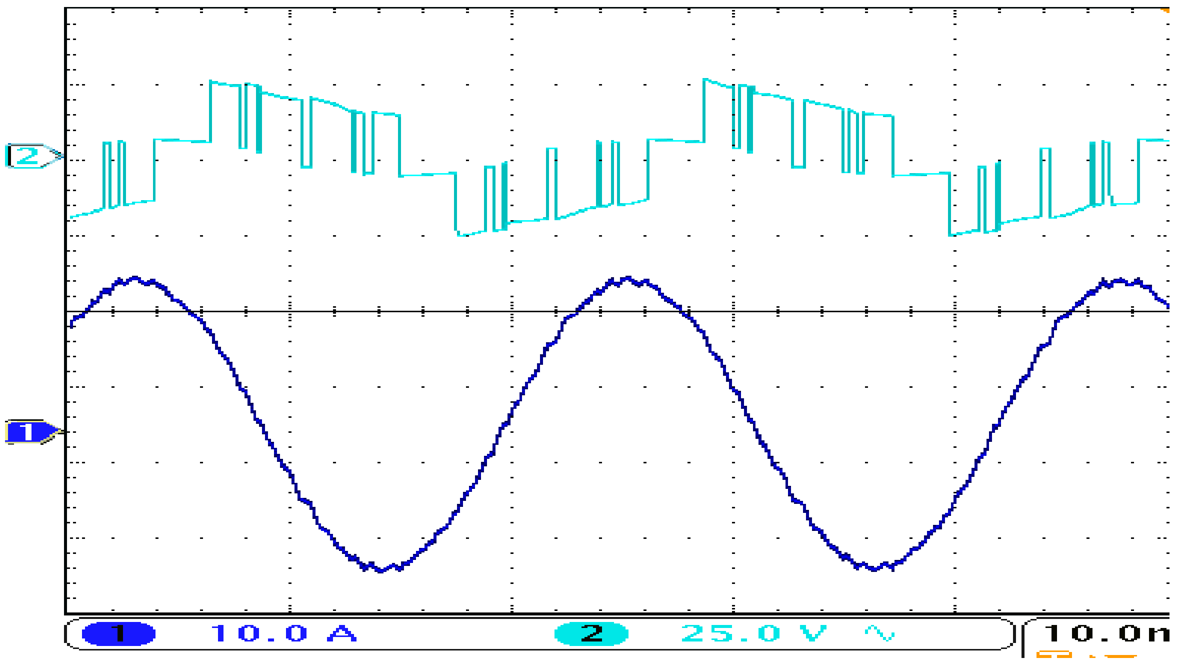


Figure 6-15 Current and Voltage for m=1, f=45Hz, N=6.

Similarly, the four figures from Figure 16-1 and Figure 16-4 in Appendix H shows per phase voltage and current for different modulation index for stator frequency 55Hz. In these figures, the upper figures are voltage signals. From these figures it is noticed that lower modulation has longer gaps between the voltage pulse or more zero voltage pulse than positive or negative voltage, so it is obvious that it is more deviated than the fundamental sinusoidal components and has higher THD. These voltage signals generated by lower modulation index has lower order harmonic component which could not be even filtered by inductance of induction motor so it is seen that current signal has more ripples and more THD than those generated by high modulation index. Since DC bus balancing algorithm is not included in this system, it can also increase the harmonics in the system.

In [28], there is graph between normalized weighted total harmonic distortion(WTHD0) versus modulation index for SOM pattern as shown in Figure 6-16. In this master thesis instead of WTHD0, THD is calculated. Hence it should not be compared with the graph completely but the trend can be seen and it seems to match with Table 6-3.

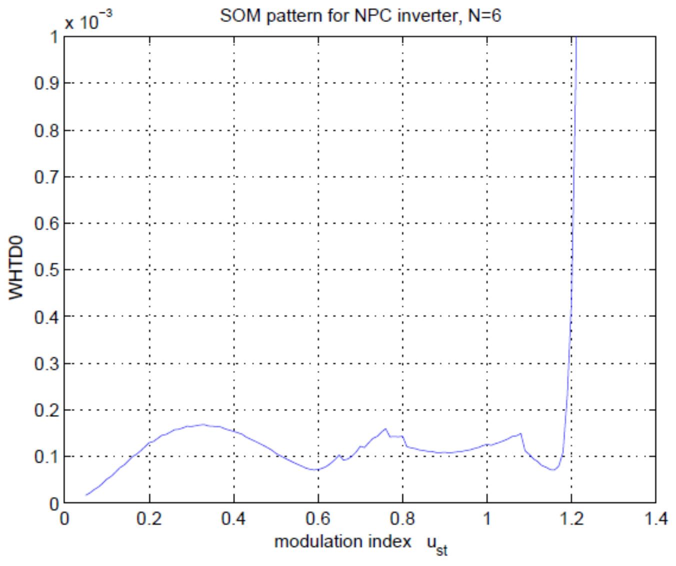


Figure 6-16 WHTD0 vs modulation index for N=6 (f=45Hz).

6.3.4 Comparison

Total harmonic distortion for various conventional PWM technique for line voltage is given in [44]. It is interesting to compare THD of the Synchronous Optimal Modulation with them at stator frequency of 50Hz. Till now the frequency of IGBT is taken as 300Hz. Hence number of pulse per half period (N) is 6 for 50Hz stator frequency. But for comparison, the switching frequency of IGBT is made 150Hz. Once switching frequency is made 150Hz in Figure 2-18 N would be 3 for 50 Hz stator frequency.

Since, $f_{sw1,avg} = N \cdot f_s$

THD for line-line voltage is compared for stator frequency = 50Hz, m=1 and switching frequency= 150 Hz.

Table 6-4 Comparison with conventional PWM technique.

Switching Frequency	150Hz
SPWM(%THD)	53.88
Trapezoidal(%THD)	52.70
SVPWM(%THD)	52.70
Synchronous Optimal Modulation (%THD)	24.4

THD of Program Modulation technique is extremely lower than the conventional PWM technique like sinusoidal PWM, Trapezoidal PWM and Space vector PWM. From section 2.6, it is explained that in SOM, the switching events are pre-calculated by some software program in order to minimize THD. Hence the result is positive that THD is highly minimised and the objective of generating SOM pattern to minimize THD for low frequency is fulfilled. Another reason for improvement is because in [44] two level inverter is used for feeding induction machine while in this thesis three level inverter is used. It is seen from section 6.3.1that line-line voltage for three level has 5 level and is more close to sine wave and harmonics are greatly reduced.

6.4 Some of Errors found in the system

There are some errors at some modulation index and stator frequency. One of the example is shown in Figure 6-17 where one switching instance is missed per fundamental period as marked by the red circle. The missing pulse in digital signal is reflected upon the voltage signal as well. The current signal is also highly distorted than the sinusoidal wave and has high ripples. For the same modulation index for frequency equals to 65 Hz, there is no any missing pulse as shown in Figure 6-18. Hence the current waveform is close to sinusoidal, however it has some ripples because the modulation index is very low. Another example is given in Appendix I for m=1. Figure 17-1 is when there are some missing pulse for m=1, 80Hz and current waveform is deviated from sine wave while Figure 17-2 is when there are no pulse missing for m=1, f=45Hz and current wave is close to sine wave.

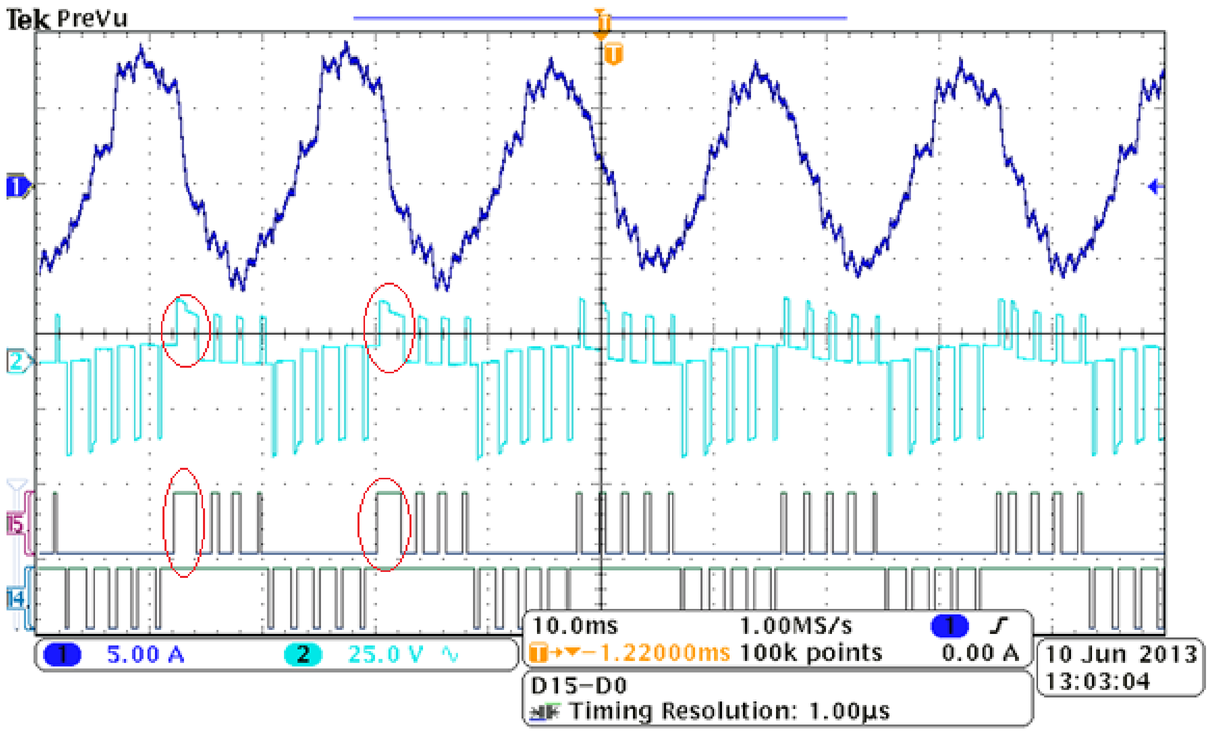


Figure 6-17 Missing Pulse for m=0.4, f=55 Hz

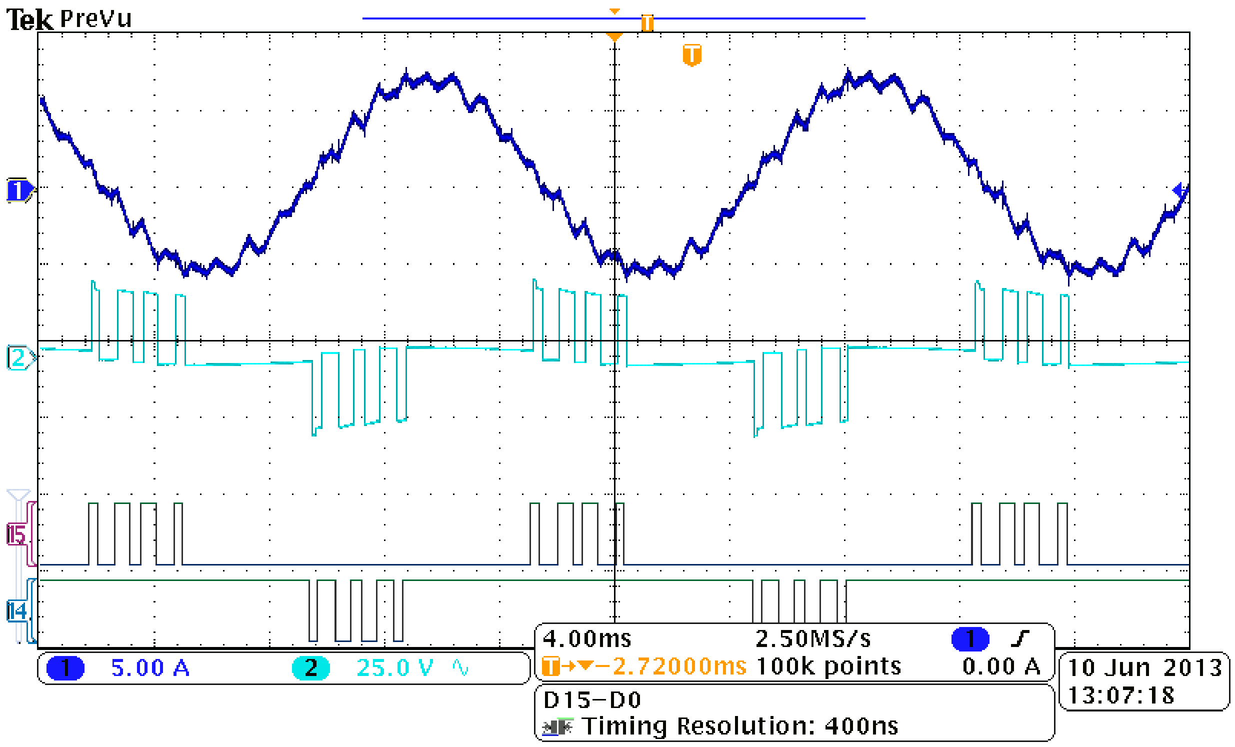


Figure 6-18 No missing Pulse for m=0.4, f=65 Hz

6.4.1 Analysis

Whenever SOM patterns are completely obtained, current signals have lower ripples and lower harmonics. The reason behind this missing pulse is difficult to predict since this master thesis works combined with collaborating master thesis. One reason seen from this master thesis is the resetting of the up counter after every sampling period. From section 4.2.2 it is known that the counter counts from zero to the maximum value and settles down to zero. Actually it does not settle down to zero naturally but it is done forcefully by programming in VHDL. From section 4.5, it is known that whenever counter resets to zero, interrupt is given to processor routine to place the value in the register for next Sample. Hence whenever switching event occur in this transition, it may be missed.

6.4.2 Suggestion

To solve this problem, counter must be used without force resetting. The counter which is used in this system is 32 bit. Hence counter reads from 0 to 2^{32}-1 and reset to zero itself after overflowing or reaching the maximum value. Instead of giving counter like Figure 6-19 which resets itself, overflow counter like Figure 6-20 is suggested to be used.

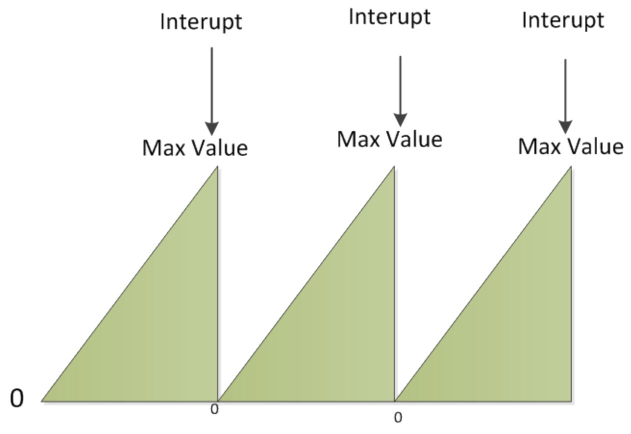


Figure 6-19 Reseting counter.

However in order to implement this type of overflow counter, processor routine should also increment its angular value suitable for every sampling interval. It is because every sampling interval starts with different value in this method.

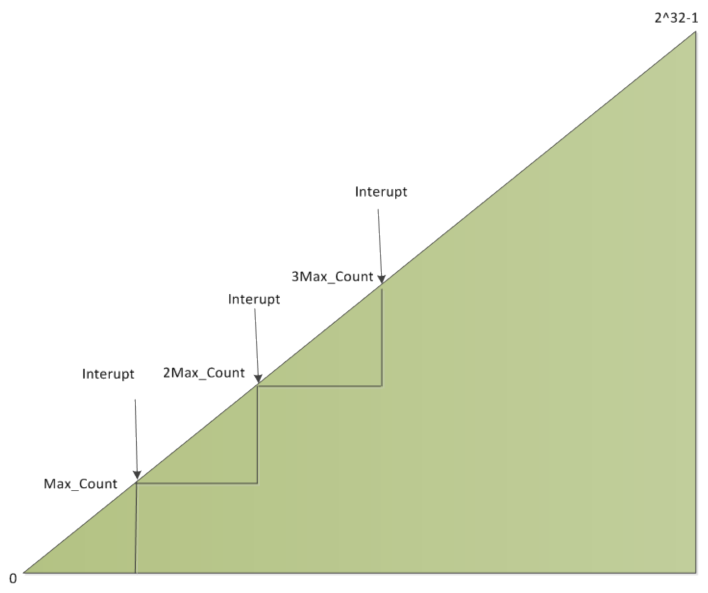


Figure 6-20 Overflow counter.

6.5 Discussion

This thesis proposes new modulation technique called synchronous optimal modulation for medium voltage multiphase machine. It has to be proposed because medium voltage switch must be operated in low switching frequency to reduce the switching loss. It was explained in theory why SOM is useful for low switching frequency. This SOM is implemented in FPGA using Xilinx tool. SOM patterns are generated by FPGA and given to 3 phase, 3 level converter feeding induction machine.

SOM patterns are sucessfully generated by FPGA for three phase. The lab was set up to see if these SOM patterns can be used to operate IGBT in low switching frequency with reduced switching loss. As per the expectation positive results were obtained and total harmonic distortion in voltage and current is reduced to great extent without even using the filters except for the inductance of induction machine. The reduction in harmonics is also enhanced by using three level converters because higher level converter would make the signal more close to the sinusoidal wave.

Since DC bus balancing is not included in this thesis, some voltage signals have unequal positive and negative pulse. The three level converter consists of two capacitor which are not equally charged. Method is suggested to improve DC bus balancing in 3-level converter by using space vector method. Harmonics can further be reduced if DC bus balancing is implemented.

In this thesis, IGBTs are used which are operated in low switching frequency 300 Hz. In order to maintain switching frequency of IGBTs less than 300Hz, number of pulse per half period (N) in SOM pattern decrease according to increase in stator frequency or vice versa. For operating in low stator frequency, required number of pulse is large. But this system works only up to N=6 and stator frequency of 45 Hz for switching frequency of 300 Hz. Further study can be done to increase this value up to 10. For lower stator frequency, N can be greater than 10. It is not good idea to implement SOM in this case. It is because SOM works by storing the optimum angle in memory. For large N, extremely large value of memory would be required which can be costly. It has been explained that for N greater than 10, conventional asynchronous modulation technique can be implemented. If asynchronous modulation technique is to be implemented in this system for lower stator frequency then software must be able to switch in between asynchronous and synchronous modulation.

Chapter 7

7 Conclusion and further work

7.1 Conclusion

For medium voltage drive, IGBTs are used. In order to reduce the harmonics, low switching frequency must be used. Low switching frequency needs special modulation technique in order to remove sub harmonics. Hence synchronous optimal modulation was introduced for operating IGBTs at low switching frequency. Synchronous Optimal Modulation is type of Program modulation which generates the optimal switching angles in one fundamental period in order to optimize weighted total harmonic distortion by some software program.

This master thesis is successfully collaborated with other master thesis in order to implement synchronous optimal modulation in FPGA for Neutral Point Clamped inverter feeding Induction machine. The optimal switching patterns are generated from processor routines which are taken by FPGA to create the digital signals for NPC inverter which acts like voltage source converter for induction machine.

The total harmonic distortion is measured for line to line voltage and line current in lab for switching frequency of 300Hz. The results show that THD is greatly reduced even for low switching frequency. It is also compared with the conventional modulation technique like sinusoidal PWM and Space Vector PWM for low switching frequency of 150Hz. The quality of output voltage is excessively improved by using synchronous optimal modulation. Hence it is proved that synchronous optimal modulation can be applied for low switching frequency with lower total harmonics distortion.

Even though dc link voltage is given 50V in lab, this can be extended for medium voltage because IGBT with blocking voltage of 4.5KV is available. In addition even if modulator for three phase is built for this master thesis, this can be easily extended for multiphase machine. It is because in FPGA, digital circuits per phase are operating in parallel and synchronised by one single clock.

7.2 Further Works

The results obtained in this thesis are interesting. Total harmonic distortion is excessively reduced than conventional modulation technique for low switching frequency. However some of the improvements can still be done and new features can be added in this system.

1. Missing pulse in this system can be improved.
2. The main motivation behind this thesis is to implement synchronous optimal modulation for multiphase machine. Even though modulation for three phase machine is implemented in FPGA, the system can be easily extended to multiphase machine.
3. DC bus balancing algorithm can be included in this system.
4. Test the modulator in the Lab by help of a 3-level 3-phase inverter operating an Induction Machine in open-loop control, i.e. V/f-control.
5. The important filtering functions for voltage and current should be implemented in the FPGA.
6. For operating in low stator frequency, required number of pulse is large. But this system works only up to N=6 for switching frequency =300 Hz. Further study can be done to increase this value up to 10.
7. For lower stator frequency, N can be greater than 10 for which conventional asynchronous modulation technique can be implemented.
8. If asynchronous modulation technique is to be implemented in this system for lower stator frequency then software must be able to switch in between asynchronous and synchronous modulation.
9. Close loop system can be implemented in order to control speed of motor.
10. Two different types of hardwares are developed in this master thesis to implement modulator by phase to phase method and space vector method. Due to time constraint only phase to phase method is used by processor routine. In future, processor routine can be developed to implement space vector method as well.

8 Bibliography

[1] E. Levi, "Multiphase Electric Machines for Variable-Speed Applications," *Industrial Electronics, IEEE Transactions on,* vol. 55, pp. 1893-1909, 2008.

[2] S. Lu and K. Corzine, "Multilevel multi-phase propulsion drives," in *Electric Ship Technologies Symposium, 2005 IEEE*, 2005, pp. 363-370.

[3] S. Thopate, "Modeling, simulation and implementation of Multi-phase Induction Motor Drives," Master in Electric Power Engineering, Department of Electric Power Engineering, Norwegian University of Science and Technology, Norway, 2011.

[4] G. K. Singh, "Multi-phase induction machine drive research—a survey," vol. 61, pp. 139-147, March 28, 2002 2002.

[5] "http://www.csemag.com/home/single-article/why-choose-medium-voltage-drives/300c6244667657657f313913d2a4a167.html."

[6] J. M. D. Murphy and M. G. Egan, "A Comparison of PWM Strategies for Inverter-Fed Induction Motors," *Industry Applications, IEEE Transactions on,* vol. IA-19, pp. 363-369, 1983.

[7] J. Holtz and N. Oikonomou, "Fast Dynamic Control of Medium Voltage Drives Operating at Very Low Switching Frequency&#x2014;An Overview," *Industrial Electronics, IEEE Transactions on,* vol. 55, pp. 1005-1013, 2008.

[8] N. Oikonomou and J. Holtz, "Closed-Loop Control of Medium-Voltage Drives Operated With Synchronous Optimal Pulsewidth Modulation," *Industry Applications, IEEE Transactions on,* vol. 44, pp. 115-123, 2008.

[9] J. M. D. Murphy, L. S. Howard, and R. G. Hoft, "Microprocessor control of a PWM inverter induction motor drive," *Institute of Electrical and Electronics Engineers, Inc,* pp. 344-348, 00/1979 1979.

[10] M. w. Naouar, E. Monmasson, A. A. Naassani, I. Slama-Belkhodja, and N. Patin, "FPGA-Based Current Controllers for AC Machine Drives&#x2014;A Review," *Industrial Electronics, IEEE Transactions on,* vol. 54, pp. 1907-1925, 2007.

[11] H. S. Patel and R. G. Hoft, "Generalized Techniques of Harmonic Elimination and Voltage Control in Thyristor Inverters: Part I--Harmonic Elimination," *Industry Applications, IEEE Transactions on,* vol. IA-9, pp. 310-317, 1973.

[12] G. S. Buja and G. B. Indri, "Optimal Pulsewidth Modulation for Feeding AC Motors," *Industry Applications, IEEE Transactions on,* vol. IA-13, pp. 38-44, 1977.

[13] P. N. Enjeti and R. Jakkli, "Optimal power control strategies for neutral point clamped (NPC) inverter topology," *Industry Applications, IEEE Transactions on,* vol. 28, pp. 558-566, 1992.

[14] J. Holtz, "Pulsewidth modulation for electronic power conversion," *Proceedings of the IEEE,* vol. 82, pp. 1194-1214, 1994.

[15] B. B. Yenore, "Low switching frequency modulation scheme for high power three level converters: FPGA based implementation " Norwegian University of Science and Technology2013.

[16] http://www.marineinsight.com/tech/marine-electrical/electrical-propulsion-system-in-ships/

[17] "http://www.imtech.eu/eCache/DEF/12/552.bGFuZz1FTg.html."

[18] S. Williamson and S. Smith, "Pulsating torque and losses in multiphase induction machines," *Industry Applications, IEEE Transactions on,* vol. 39, pp. 986-993, 2003.

[19] R. Nilsen, "Modelling of multi-phase Synchronous Machines," Wärtsilä Norway AS, Trondhiem, ConfidentialApril 6, 2009 2009.

[20] R. H. Nelson and P. C. Krause, "Induction Machine Analysis for Arbitrary Displacement Between Multiple Winding Sets," *Power Apparatus and Systems, IEEE Transactions on,* vol. PAS-93, pp. 841-848, 1974.

[21] R. P. Bingham, "HARMONICS - Understanding the Facts."

[22] D. G. Holmes and L. A. Thomas, *Pulse width modulation for power converters*. USA, 2003.

[23] N. Mohan, T. M. Undeland, and R. P. William, "The Power Electronics: Converter, Applications and Design," 2003.

[24] R. Lund, "Multilevel Power Electronic Converters for Electrical Motor Drives," Phd, Department of Electrical Power Engineering, Norwegian University of Science and Technology, Trondhiem, 2005.

[25] R. Nilsen, "Modulation Methods for 3 Level Inverter," Wärtsilä Norway AS, Stiklestadveien 1 N-7041 TRONDHEIM, Confidential2010-02-17 2010.

[26] S. Floten and T. S. Haug, "Modulation Methods for Neutral-Point-Clamped Three-Level Inverter," N. U. o. S. a. Technology, Ed., ed. Trondhiem, 2010.

[27] M. H. Bierhoff and F. W. Fuchs, "Semiconductor losses in voltage source and current source IGBT converters based on analytical derivation," in *Power Electronics Specialists Conference, 2004. PESC 04. 2004 IEEE 35th Annual*, 2004, pp. 2836-2842 Vol.4.

[28] R. Enes, "Modelling and Control of High Performance Medium Voltage Drives," Master of Science in Electric Power Engineering, Department of Electric Power Engineering, Norwegian University of Science and Technology, Trondhiem, 2012.

[29] J. Holtz and N. Oikonomou, "Synchronous optimal pulsewidth modulation and stator flux trajectory control for medium voltage drives," in *Industry Applications Conference, 2005. Fourtieth IAS Annual Meeting. Conference Record of the 2005*, 2005, pp. 1748-1791 Vol. 3.

[30] H. Berriri, W. Naouar, I. Bahri, I. Slama-Belkhodja, and E. Monmasson, "Field programmable gate array-based fault-tolerant hysteresis current control for AC machine drives," *Electric Power Applications, IET,* vol. 6, pp. 181-189, 2012.

[31] E. Monmasson and M. N. Cirstea, "FPGA Design Methodology for Industrial Control Systems&#x2014;A Review," *Industrial Electronics, IEEE Transactions on,* vol. 54, pp. 1824-1842, 2007.

[32] S. S. Gjerde, "Introduction to Xilinx FPGA for Digital Control of Power Electronics," Norwegian University of Science and Technology, TrondhiemOctober 31. 2011 2011.

[33] V. A. Pedroni, *Circuit Design with VHDL*: MIT Press, 2004.

[34] M. Aymen, A. Abdelaziz, S. Halim, and H. Maaref, "Hardware Implementation on a Xilinx Virtex4," Laboratory of Micro-Optoélectronique and Nanostructure, Faculty of sciences Monastir, Tunisia.

[35] K. Ljøkelsøy, "FPGA based processor board for control of power electroncs converters," SINTEF, Norway, Technical reportDecember 12, 2008 2008.

[36] "http://www.plagscan.com/highlight?doc=2084618&source=0."

[37] " http://www.seas.upenn.edu/~ese171/ise/ISEIntroduction.pdf."

[38] G. Ochoa-Ruiz, O. Labbani-Narsis, and E. Bourennane, "FACILITATING IP DEPLOYMENT IN A MARTE-BASED MDE METHODOLOGY USING IP-XACT: A XILINX EDK CASE STUDY," LE2I Laboratory, Université de Bourgogne, France.

[39] http://www.xilinx.com/support/documentation/sw_manuals/xilinx12_1/SDK_Doc/concepts/sdk_c_hwspec.htm

[40] *http://www.adeneo-embedded.com/Services/BSP-and-Driver-Development*. Available: http://www.adeneo-embedded.com/Services/BSP-and-Driver-Development

[41] "http://www.origin.xilinx.com/products/intellectual-property/."
[42] "PDF attached to the IPs in XPS."
[43] R. Nilsen, "Ideas ".
[44] R. R. Kumar, S. Kumar, and A. Yadav, "Comparison of PWM Techniques and Inverter Performance," *IOSR Journal of Electrical and Electronics Engineering (IOSR-JEEE),* vol. 4, pp. 18-22, 2013.

9 Appendix A

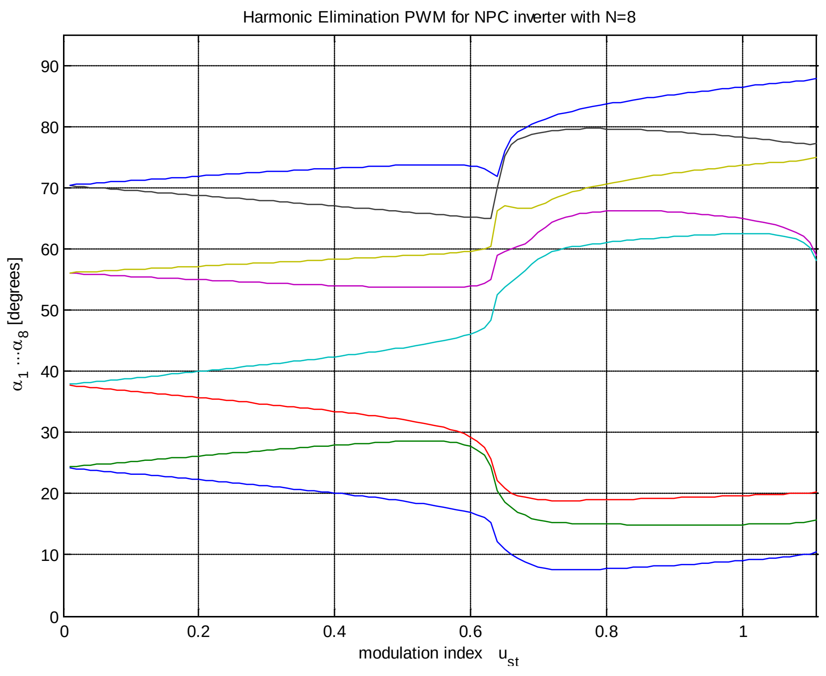


Figure 9-1 SOM pattern for N=8.

Table 9-1Rating of induction machine.

Parameter	Explanation	Value
U_N	Nominal line to line voltage [V_{rms}]	400 V
I_N	Nominal line current [A_{rms}]	11.8 A
f_N	Nominal frequency [Hz]	75 Hz
p	Number of pole pairs	2
n_N	Nominal speed [mechanical rpm]	2235 rpm
M_N	Nominal output torque [Nm]	50 Nm
P_N	Nominal power output [kW]	11.7 kW
$cos\varphi_N$	Nominal power factor	0.77
n_{MAX}	Maximum speed [mechanical rpm]	5000

10 Appendix B

```
# Virtex5_EDK_basisutgave
# #
# Kjell Ljøkelsøy, Sintef Energi AS
# #
# Device:    virtex5 xc5vfx30t
# Package:   ff665
# Speed Grade:  -1
# Processor: ppc440_0
# Processor clock frequency: 400.00 MHz
# Bus clock frequency: 100.00 MHz
# On Chip Memory : 128 KB

PARAMETER VERSION = 2.1.0

 PORT KLOKKEOSC = KLOKKEOSC_s, DIR = I, SIGIS = CLK, CLK_FREQ = 40000000
 PORT LED_RESET_GRONN_ROD = LED_RESET_GRONN_ROD_s, DIR = O, SIGIS = RST, RST_POLARITY = 0
 PORT RS232_INN = RS232_INN, DIR = I
 PORT RS232_UT = RS232_UT, DIR = O
 PORT SYSMON_AD_N = SYSMON_AD_N_s, DIR = I, VEC = [10:0]
 PORT SYSMON_AD_P = SYSMON_AD_P_s, DIR = I, VEC = [10:0]
 PORT SIG_D = SIG_D_s, DIR = I, VEC = [5:0]
 PORT DIG_IO1_D = DIG_IO1_D, DIR = IO, VEC = [15:0]
 PORT RELE = RELE_s, DIR = O, VEC = [3:0]
 PORT LED_SYSMON_ROD = LED_SYSMON_ROD_s, DIR = O
 PORT LED_SYSMON_GUL = LED_SYSMON_GUL_s, DIR = O
 PORT LED_USB_GRONN = LED_USB_GRONN_s, DIR = O
 PORT DDR2_ODT = DDR2_ODT, DIR = O
 PORT DDR2_A = DDR2_A, DIR = O, VEC = [12:0]
 PORT DDR2_BA = DDR2_BA, DIR = O, VEC = [2:0]
 PORT DDR2_CAS = DDR2_CAS, DIR = O
 PORT DDR2_CKE = DDR2_CKE, DIR = O
 PORT DDR2_CS = DDR2_CS, DIR = O
 PORT DDR2_RAS = DDR2_RAS, DIR = O
 PORT DDR2_WE = DDR2_WE, DIR = O
 PORT DDR2_DM = DDR2_DM, DIR = O, VEC = [3:0]
 PORT DDR2_DQS = DDR2_DQS, DIR = IO, VEC = [3:0]
 PORT DDR2_DQSN = DDR2_DQSN, DIR = IO, VEC = [3:0]
 PORT DDR2_D = DDR2_D, DIR = IO, VEC = [31:0]
 PORT DDR2_CK = DDR2_CK, DIR = O
 PORT DDR2_CKN = DDR2_CKN, DIR = O
 PORT F_WR_LL = F_WR_LL, DIR = O
 PORT F_WR_HL = F_WR_HL, DIR = O
 PORT F_WR_HH = F_WR_HH, DIR = O
 PORT F_RD = F_RD, DIR = O
 PORT F_A = F_A, DIR = O, VEC = [26:1]
 PORT F_D = F_D, DIR = IO, VEC = [31:0]
 PORT F_CS = F_CS, DIR = O
 PORT F_RESET = F_RESET_s, DIR = O, SIGIS = RST
 PORT EE_CS = EE_CS, DIR = O
 PORT USB_CS = USB_CS, DIR = O
 PORT USB_INT = USB_INT_s, DIR = I, SIGIS = INTERRUPT, SENSITIVITY =
EDGE_RISING
 PORT CAN_INT = CAN_INT_s, DIR = I, SIGIS = INTERRUPT, SENSITIVITY =
EDGE_RISING
PORT ETH_CRS_CRS_DV = 'Z', DIR = I
 PORT ETH_COL = 'Z', DIR = I
 PORT ETH_PD_INT = 'Z', DIR = I
 PORT ETH_RX_CLK = ETH_RX_CLK, DIR = I
 PORT ETH_RX_DV = ETH_RX_DV, DIR = I
 PORT ETH_RX_D = ETH_RX_D, DIR = I, VEC = [3:0]
 PORT ETH_TX_CLK = ETH_TX_CLK, DIR = I
 PORT ETH_TX_D = ETH_TX_D, DIR = O, VEC = [3:0]
 PORT ETH_TX_EN = ETH_TX_EN, DIR = O
 PORT ETH_RX_ERR = ETH_RX_ERR, DIR = I
 PORT ETH_RESET_N = ETH_RESET_N, DIR = O
 PORT ETH_KLOKKE = clk_25_0000MHz, DIR = O, SIGIS = CLK, CLK_FREQ = 25000000
 PORT ETH_MDIO = ETH_MDIO, DIR = IO
 PORT ETH_MDC = ETH_MDC, DIR = O
# dummypinne
 PORT EKSTRAPINNE2 = ETH_TX_ERR, DIR = O
 PORT CAN_KLOKKE = klokke_20_MHz, DIR = O, SIGIS = CLK, CLK_FREQ = 20000000
 PORT USB_KLOKKE = klokke_12_MHz, DIR = O, SIGIS = CLK, CLK_FREQ = 12000000
 PORT VE_PAA = VE_PAA, DIR = O
 PORT VE_DRIVER = VE_DRIVER_UT, DIR = O, VEC = [5:0]
 PORT VE_OK = VE_OK, DIR = I
 PORT VE_T = VE_T, DIR = I, VEC = [3:0]
 PORT LED_VE = LED_VE, DIR = O, VEC = [1:0]
 PORT AD_REF_KLOKKE_N = AD_REF_KLOKKE_N, DIR = O
 PORT AD_REF_KLOKKE_P = AD_REF_KLOKKE_P, DIR = O
 PORT AD_DCON = AD_DCON, DIR = I, SIGIS = CLK, CLK_FREQ = 240000000
 PORT AD_DCOP = AD_DCOP, DIR = I, SIGIS = CLK, CLK_FREQ = 240000000
 PORT AD_FCON = AD_FCON, DIR = I
 PORT AD_FCOP = AD_FCOP, DIR = I
 PORT AD_D_N = AD_D_N, DIR = I, VEC = [7:0]
 PORT AD_D_P = AD_D_P, DIR = I, VEC = [7:0]
 PORT AD_CSB = AD_CSB, DIR = O
 PORT AD_SDIO = AD_SDIO, DIR = IO
 PORT AD_SCLK = AD_SCLK, DIR = O
 PORT CAN_SCK = CAN_SCK, DIR = O
 PORT CAN_SDO = CAN_SDO, DIR = I
 PORT CAN_SDI = CAN_SDI, DIR = O
 PORT CAN_CS = CAN_CS, DIR = O
 PORT DA_D = DA_D, DIR = O, VEC = [11:0]
 PORT DA_AB_CS = DA_AB_CS, DIR = O
 PORT DA_CD_CS = DA_CD_CS, DIR = O
 PORT DA_RW = DA_RW, DIR = O
 PORT DA_A_B = DA_A_B, DIR = O
 PORT LED_TEST = LED_TEST_s, DIR = O, VEC = [5:0]
 PORT som_phase_ip_0_interupt_port_pin = som_phase_ip_0_interupt_port, DIR = O, SIGIS =
INTERRUPT, SENSITIVITY = EDGE_RISING
 PORT DIG_IO3_D = DIG_IO3_D, DIR = IO, VEC = [15:0]


BEGIN ppc440_virtex5
 PARAMETER INSTANCE = ppc440_0
 PARAMETER HW_VER = 1.01.a
 PARAMETER C_PPC440MC_CONTROL = 0x8060008F
 PARAMETER C_APU_CONTROL = 0b00000010000000001
 PARAMETER C_IDCR_BASEADDR = 0b0000000000
 PARAMETER C_IDCR_HIGHADDR = 0b0011111111
 PARAMETER C_SPLB0_NUM_MPLB_ADDR_RNG = 0
 PARAMETER C_SPLB1_NUM_MPLB_ADDR_RNG = 0
 PARAMETER C_NUM_DMA = 1
 BUS_INTERFACE JTAGPPC = jtagppc_cntlr_0_0
 BUS_INTERFACE RESETPPC = ppc_reset_bus
 BUS_INTERFACE MPLB = plb_v46_0
 BUS_INTERFACE PPC440MC = ppc440_0_PPC440MC
 BUS_INTERFACE LLDMA0 = xps_ll_temac_0_LLINK0
 BUS_INTERFACE MFCB = fcb_v20_0
 PORT CPMC440CLK = proc_clk_s
 PORT CPMPPCMPLBCLK = sys_clk_s
 PORT CPMPPCS0PLBCLK = sys_clk_s
 PORT CPMDMA0LLCLK = sys_clk_s
 PORT CPMINTERCONNECTCLKNTO1 = net_vcc
 PORT CPMMCCLK = clk_200_0000MHzPLL0_ADJUST
 PORT CPMINTERCONNECTCLK = clk_200_0000MHzPLL0
 PORT EICC440EXTIRQ = ppc440_0_EICC440EXTIRQ
 PORT PPCEICINTERCONNECTIRQ = ppc440_0_PPCEICINTERCONNECTIRQ
END

BEGIN jtagppc_cntlr
 PARAMETER INSTANCE = jtagppc_cntlr_0
 PARAMETER HW_VER = 2.01.c
 BUS_INTERFACE JTAGPPC0 = jtagppc_cntlr_0_0
END
```

```
BEGIN apu_fpu_virtex5
 PARAMETER INSTANCE = apu_fpu_virtex5_0
 PARAMETER HW_VER = 1.01.a
 BUS_INTERFACE SFCB2 = fcb_v20_0
END

BEGIN fcb_v20
 PARAMETER INSTANCE = fcb_v20_0
 PARAMETER HW_VER = 1.00.a
 PORT SYS_RST = sys_bus_reset
 PORT FCB_CLK = klokke_133_333MHz
END

BEGIN plb_v46
 PARAMETER INSTANCE = plb_v46_0
 PARAMETER C_DCR_INTFCE = 0
 PARAMETER HW_VER = 1.05.a
 PORT PLB_Clk = sys_clk_s
 PORT SYS_Rst = sys_bus_reset
 PORT Bus_Error_Det = plb_v46_0_Bus_Error_Det
END

BEGIN clock_generator
 PARAMETER INSTANCE = clock_generator_0
 PARAMETER HW_VER = 4.01.a
 PARAMETER C_CLKIN_FREQ = 40000000
 PARAMETER C_CLKOUT0_FREQ = 100000000
 PARAMETER C_CLKOUT0_PHASE = 0
 PARAMETER C_CLKOUT0_GROUP = PLL0_ADJUST
 PARAMETER C_CLKOUT0_BUF = TRUE
 PARAMETER C_CLKOUT1_FREQ = 133333333
 PARAMETER C_CLKOUT1_PHASE = 0
 PARAMETER C_CLKOUT1_GROUP = PLL0
 PARAMETER C_CLKOUT1_BUF = TRUE
 PARAMETER C_CLKOUT2_FREQ = 200000000
 PARAMETER C_CLKOUT2_PHASE = 90
 PARAMETER C_CLKOUT2_GROUP = PLL0_ADJUST
 PARAMETER C_CLKOUT2_BUF = TRUE
 PARAMETER C_CLKOUT3_FREQ = 200000000
 PARAMETER C_CLKOUT3_PHASE = 0
 PARAMETER C_CLKOUT3_GROUP = PLL0
 PARAMETER C_CLKOUT3_BUF = TRUE
 PARAMETER C_CLKOUT4_FREQ = 200000000
 PARAMETER C_CLKOUT4_PHASE = 0
 PARAMETER C_CLKOUT4_GROUP = PLL0_ADJUST
 PARAMETER C_CLKOUT4_BUF = TRUE
 PARAMETER C_CLKOUT5_FREQ = 25000000
 PARAMETER C_CLKOUT5_PHASE = 0
 PARAMETER C_CLKOUT5_GROUP = NONE
 PARAMETER C_CLKOUT5_BUF = TRUE
 PARAMETER C_CLKOUT6_FREQ = 400000000
 PARAMETER C_CLKOUT6_PHASE = 0
 PARAMETER C_CLKOUT6_GROUP = PLL0
 PARAMETER C_CLKOUT6_BUF = TRUE
 PARAMETER C_EXT_RESET_HIGH = 1
 PARAMETER C_CLKOUT7_FREQ = 20000000
 PARAMETER C_CLKOUT8_FREQ = 12000000
 PORT CLKIN = KLOKKEOSC_s_buf
 PORT CLKOUT0 = sys_clk_s
 PORT CLKOUT1 = klokke_133_333MHz
 PORT CLKOUT2 = clk_200_0000MHz90PLL0_ADJUST
 PORT CLKOUT3 = clk_200_0000MHzPLL0
 PORT CLKOUT4 = clk_200_0000MHzPLL0_ADJUST
 PORT CLKOUT5 = clk_25_0000MHz
 PORT CLKOUT6 = proc_clk_s
 PORT LOCKED = Dcm_all_locked
 PORT RST = net_gnd
 PORT CLKOUT7 = klokke_20_MHz
 PORT CLKOUT8 = klokke_12_MHz
END

BEGIN bufg_modul
 PARAMETER INSTANCE = bufg_modul_0
 PARAMETER HW_VER = 1.00.a
 PORT Clk_ut = KLOKKEOSC_s_buf
 PORT Clk_in = KLOKKEOSC_s
END

BEGIN xps_bram_if_cntlr
 PARAMETER INSTANCE = FPGA_BLOKKRAM_0
 PARAMETER C_SPLB_NATIVE_DWIDTH = 64
 PARAMETER C_SPLB_P2P = 0
 PARAMETER C_SPLB_SUPPORT_BURSTS = 1
 PARAMETER HW_VER = 1.00.b
 PARAMETER C_BASEADDR = 0xfffe0000
 PARAMETER C_HIGHADDR = 0xffffffff
 BUS_INTERFACE SPLB = plb_v46_0
 BUS_INTERFACE PORTA = FPGA_BLOKKRAM_0_PORTA
END

BEGIN bram_block
 PARAMETER INSTANCE = bram_block_0
 PARAMETER HW_VER = 1.00.a
 BUS_INTERFACE PORTA = FPGA_BLOKKRAM_0_PORTA
END

BEGIN xps_bram_if_cntlr
 PARAMETER INSTANCE = FPGA_BLOKKRAM_1
 PARAMETER HW_VER = 1.00.b
 PARAMETER C_SPLB_P2P = 0
 PARAMETER C_SPLB_SUPPORT_BURSTS = 1
 PARAMETER C_SPLB_NATIVE_DWIDTH = 64
 PARAMETER C_BASEADDR = 0x85810000
 PARAMETER C_HIGHADDR = 0x8581ffff
 BUS_INTERFACE PORTA = FPGA_BLOKKRAM_1_PORTA
 BUS_INTERFACE SPLB = plb_v46_0
END

BEGIN bram_block
 PARAMETER INSTANCE = bram_block_1
 PARAMETER HW_VER = 1.00.a
 BUS_INTERFACE PORTA = FPGA_BLOKKRAM_1_PORTA
END

BEGIN proc_sys_reset
 PARAMETER INSTANCE = proc_sys_reset_0
 PARAMETER HW_VER = 3.00.a
 PARAMETER C_EXT_RESET_HIGH = 1
 PARAMETER C_NUM_PERP_RST = 2
 BUS_INTERFACE RESETPPC0 = ppc_reset_bus
 PORT Slowest_sync_clk = sys_clk_s
 PORT Dcm_locked = Dcm_all_locked
 PORT Bus_Struct_Reset = sys_bus_reset
 PORT Peripheral_Reset = sys_periph_reset & LED_RESET_GRONN_ROD_s
 PORT Peripheral_aresetn = F_RESET_s
END

BEGIN xps_intc
 PARAMETER INSTANCE = xps_intc_0
 PARAMETER HW_VER = 2.01.a
 PARAMETER C_BASEADDR = 0x81800000
 PARAMETER C_HIGHADDR = 0x8180ffff
 BUS_INTERFACE SPLB = plb_v46_0
 PORT Irq = ppc440_0_EICC440EXTIRQ
 PORT Intr = ppc440_0_PPCEICINTERCONNECTIRQ & plb_v46_0_Bus_Error_Det &
xps_ll_temac_0_TemacIntc0_Irpt & xps_timer_1_Interrupt & RS232_Interrupt & CAN_INT_s
& USB_INT_s & som_phase_ip_0_interupt_port
END
```

```
BEGIN mpmc
 PARAMETER INSTANCE = DDR2_DRAM
 PARAMETER HW_VER = 6.03.a
 PARAMETER C_NUM_PORTS = 1
 PARAMETER C_MEM_PARTNO = MT47H64M16-5E
 PARAMETER C_MEM_DATA_WIDTH = 32
 PARAMETER C_MEM_ODT_TYPE = 1
 PARAMETER C_MEM_REDUCED_DRV = 1
 PARAMETER C_PIM0_BASETYPE = 5
 PARAMETER C_MPMC_BASEADDR = 0x00000000
 PARAMETER C_MPMC_HIGHADDR = 0x0fffffff
 BUS_INTERFACE PPC440MC0 = ppc440_0_PPC440MC
 PORT MPMC_Clk0 = clk_200_0000MHzPLL0_ADJUST
 PORT MPMC_Clk90 = clk_200_0000MHz90PLL0_ADJUST
 PORT MPMC_Clk0_DIV2 = sys_clk_s
 PORT MPMC_Clk_200MHz = clk_200_0000MHzPLL0_ADJUST
 PORT MPMC_Rst = sys_periph_reset
 PORT DDR2_ODT = DDR2_ODT
 PORT DDR2_Addr = DDR2_A
 PORT DDR2_BankAddr = DDR2_BA
 PORT DDR2_CAS_n = DDR2_CAS
 PORT DDR2_CE = DDR2_CKE
 PORT DDR2_CS_n = DDR2_CS
 PORT DDR2_RAS_n = DDR2_RAS
 PORT DDR2_WE_n = DDR2_WE
 PORT DDR2_DM = DDR2_DM
 PORT DDR2_DQS = DDR2_DQS
 PORT DDR2_DQS_n = DDR2_DQSN
 PORT DDR2_DQ = DDR2_D
 PORT DDR2_Clk = DDR2_CK
 PORT DDR2_Clk_n = DDR2_CKN
END

BEGIN xps_epc
 PARAMETER INSTANCE = FLASH_EEPROM_USB
 PARAMETER HW_VER = 1.02.a
 PARAMETER C_PRH0_ADDR_TSU = 100000
 PARAMETER C_PRH0_ADDR_TH = 120000
 PARAMETER C_PRH0_ADS_WIDTH = 0
 PARAMETER C_PRH0_CSN_TSU = 10000
 PARAMETER C_PRH0_CSN_TH = 120000
 PARAMETER C_PRH0_WRN_WIDTH = 120000
 PARAMETER C_PRH0_WR_CYCLE = 300000
 PARAMETER C_PRH0_DATA_TSU = 100000
 PARAMETER C_PRH0_DATA_TH = 30000
 PARAMETER C_PRH0_RDN_WIDTH = 120000
 PARAMETER C_PRH0_RD_CYCLE = 150000
 PARAMETER C_PRH0_DATA_TOUT = 120000
 PARAMETER C_PRH0_DATA_TINV = 20000
 PARAMETER C_PRH0_RDY_TOUT = 1000
 PARAMETER C_PRH0_RDY_WIDTH = 100000
 PARAMETER C_PRH0_DWIDTH_MATCH = 1
 PARAMETER C_PRH0_SYNC = 0
 PARAMETER C_NUM_PERIPHERALS = 3
 PARAMETER C_PRH_MAX_AWIDTH = 27
 PARAMETER C_PRH0_AWIDTH = 27
 PARAMETER C_PRH1_AWIDTH = 16
 PARAMETER C_PRH1_ADDR_TSU = 100000
 PARAMETER C_PRH1_ADDR_TH = 10000
 PARAMETER C_PRH1_ADS_WIDTH = 1000
 PARAMETER C_PRH1_CSN_TSU = 100000
 PARAMETER C_PRH1_CSN_TH = 150000
 PARAMETER C_PRH1_WRN_WIDTH = 200000
 PARAMETER C_PRH1_WR_CYCLE = 300000
 PARAMETER C_PRH1_DATA_TSU = 150000
 PARAMETER C_PRH1_DATA_TH = 1000
 PARAMETER C_PRH1_RDN_WIDTH = 150000
 PARAMETER C_PRH1_RD_CYCLE = 250000
 PARAMETER C_PRH1_DATA_TOUT = 150000
 PARAMETER C_PRH1_DATA_TINV = 10500
 PARAMETER C_PRH1_RDY_TOUT = 100
 PARAMETER C_PRH1_RDY_WIDTH = 1000
 PARAMETER C_PRH2_AWIDTH = 9
 PARAMETER C_PRH2_ADDR_TSU = 20000
 PARAMETER C_PRH2_ADDR_TH = 201000
 PARAMETER C_PRH2_ADS_WIDTH = 20000
 PARAMETER C_PRH2_CSN_TSU = 20000
 PARAMETER C_PRH2_CSN_TH = 40000
 PARAMETER C_PRH2_WRN_WIDTH = 40000
 PARAMETER C_PRH2_WR_CYCLE = 50000
 PARAMETER C_PRH2_DATA_TSU = 30000
 PARAMETER C_PRH2_DATA_TH = 30000
 PARAMETER C_PRH2_RDN_WIDTH = 20000
 PARAMETER C_PRH2_RD_CYCLE = 40000
 PARAMETER C_PRH2_DATA_TOUT = 40000
 PARAMETER C_PRH2_DATA_TINV = 30000
 PARAMETER C_PRH2_RDY_TOUT = 100
 PARAMETER C_PRH2_RDY_WIDTH = 10000
 PARAMETER C_PRH1_SYNC = 0
 PARAMETER C_PRH2_SYNC = 0
 PARAMETER C_PRH_CLK_PERIOD_PS = 10000
```

```
 PARAMETER C_PRH1_DWIDTH_MATCH = 1
 PARAMETER C_PRH2_DWIDTH_MATCH = 1
 PARAMETER C_PRH1_DWIDTH = 16
 PARAMETER C_PRH2_DWIDTH = 16
 PARAMETER C_PRH1_FIFO_ACCESS = 0
 PARAMETER C_PRH1_BUS_MULTIPLEX = 0
 PARAMETER C_PRH2_FIFO_ACCESS = 0
 PARAMETER C_PRH2_BUS_MULTIPLEX = 0
 PARAMETER C_PRH0_BASEADDR = 0x88000000
 PARAMETER C_PRH0_HIGHADDR = 0x8fffffff
 PARAMETER C_PRH1_BASEADDR = 0x80a00000
 PARAMETER C_PRH1_HIGHADDR = 0x80a0ffff
 PARAMETER C_PRH2_BASEADDR = 0x80c00000
 PARAMETER C_PRH2_HIGHADDR = 0x80c0ffff
 BUS_INTERFACE SPLB = plb_v46_0
 PORT PRH_Rdy = net_vcc
 PORT PRH_Wr_n = F_WR_n
 PORT PRH_Rd_n = F_RD
 PORT PRH_BE = F_BE_h
 PORT PRH_Addr = F_A & 'Z'
 PORT PRH_Data = F_D
 PORT PRH_CS_n = F_CS & EE_CS & USB_CS
 PORT PRH_Rst = sys_bus_reset
 PORT PRH_Clk = sys_clk_s
END

BEGIN util_vector_logic
 PARAMETER INSTANCE = f_wr_logikk
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_OPERATION = or
 PARAMETER C_SIZE = 4
 PORT Res = F_WR_HH & F_WR_HL & 'Z' & F_WR_LL
 PORT Op1 = F_WR_n & F_WR_n & F_WR_n & F_WR_n
 PORT Op2 = F_BE_inv
END

BEGIN util_vector_logic
 PARAMETER INSTANCE = f_byte_enable_inverter
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_OPERATION = not
 PARAMETER C_SIZE = 4
 PORT Res = F_BE_inv
 PORT Op1 = F_BE_h
END

BEGIN xps_timer
```

```
 PARAMETER INSTANCE = xps_timer_0
 PARAMETER HW_VER = 1.02.a
 PARAMETER C_COUNT_WIDTH = 32
 PARAMETER C_ONE_TIMER_ONLY = 0
 PARAMETER C_BASEADDR = 0x83c00000
 PARAMETER C_HIGHADDR = 0x83c0ffff
 BUS_INTERFACE SPLB = plb_v46_0
 PORT Interrupt = xps_timer_1_Interrupt
END

BEGIN xps_uartlite
 PARAMETER INSTANCE = RS232
 PARAMETER HW_VER = 1.01.a
 PARAMETER C_BAUDRATE = 115200
 PARAMETER C_DATA_BITS = 8
 PARAMETER C_ODD_PARITY = 1
 PARAMETER C_USE_PARITY = 0
 PARAMETER C_SPLB_CLK_FREQ_HZ = 100000000
 PARAMETER C_BASEADDR = 0x84000000
 PARAMETER C_HIGHADDR = 0x8400ffff
 BUS_INTERFACE SPLB = plb_v46_0
 PORT RX = RS232_INN
 PORT TX = RS232_UT
 PORT Interrupt = RS232_Interrupt
END

BEGIN spi_io_logikk
 PARAMETER INSTANCE = spi_io_logikk_0
 PARAMETER HW_VER = 1.00.a
 PORT CAN_CS = CAN_CS
 PORT CAN_SDI = CAN_SDI
 PORT CAN_SDO = CAN_SDO
 PORT CAN_SCK = CAN_SCK
 PORT AD_CSB = AD_CSB
 PORT AD_SCLK = AD_SCLK
 PORT AD_SDIO = AD_SDIO
END

BEGIN xps_ll_temac
 PARAMETER INSTANCE = Hard_Ethernet_MAC
 PARAMETER C_FAMILY = virtex5
 PARAMETER C_PHY_TYPE = 0
 PARAMETER C_TEMAC1_ENABLED = 0
 PARAMETER C_BUS2CORE_CLK_RATIO = 1
 PARAMETER C_TEMAC_TYPE = 0
 PARAMETER C_TEMAC0_PHYADDR = 0b00001
 PARAMETER HW_VER = 2.03.a
 PARAMETER C_BASEADDR = 0x87000000
 PARAMETER C_HIGHADDR = 0x8707ffff
 BUS_INTERFACE SPLB = plb_v46_0
 BUS_INTERFACE LLINK0 = xps_ll_temac_0_LLINK0
 PORT TemacPhy_RST_n = ETH_RESET_N
 PORT TemacIntc0_Irpt = xps_ll_temac_0_TemacIntc0_Irpt
 PORT MII_TXD_0 = ETH_TX_D
 PORT MII_TX_EN_0 = ETH_TX_EN
 PORT MII_TX_CLK_0 = ETH_TX_CLK
 PORT MII_RX_CLK_0 = ETH_RX_CLK
 PORT MII_RX_ER_0 = ETH_RX_ERR
 PORT MII_TX_ER_0 = ETH_TX_ERR
PORT MII_RX_DV_0 = ETH_RX_DV
 PORT MII_RXD_0 = ETH_RX_D
 PORT MDIO_0 = ETH_MDIO
 PORT MDC_0 = ETH_MDC
 PORT LlinkTemac0_CLK = sys_clk_s
END

BEGIN xps_gpio
 PARAMETER INSTANCE = DIG_IO1_GPIO
 PARAMETER HW_VER = 2.00.a
 PARAMETER C_ALL_INPUTS = 0
 PARAMETER C_GPIO_WIDTH = 16
 PARAMETER C_IS_DUAL = 0
 PARAMETER C_BASEADDR = 0x81440000
 PARAMETER C_HIGHADDR = 0x8144ffff
 BUS_INTERFACE SPLB = plb_v46_0
 PORT GPIO_IO = DIG_IO1_D
END

BEGIN util_vector_logic
 PARAMETER INSTANCE = signal_inn_inverter
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_OPERATION = not
 PARAMETER C_SIZE = 6
 PORT Op1 = SIG_D_s
END

BEGIN ad_omformer_seriemottaker
 PARAMETER INSTANCE = ad_omformer_seriemottaker_0
 PARAMETER HW_VER = 1.00.a
 PARAMETER LEGG_INN_INTERN_DELAYCTRL = 0
 PARAMETER C_BASEADDR = 0xc0400000
 PARAMETER C_HIGHADDR = 0xc040ffff
 BUS_INTERFACE SPLB = plb_v46_0
 PORT AD_REF_KLOKKE_INN = KLOKKEOSC_s
 PORT AD_REF_KLOKKE_N = AD_REF_KLOKKE_N
 PORT AD_REF_KLOKKE_P = AD_REF_KLOKKE_P
 PORT AD_DCON = AD_DCON
 PORT AD_DCOP = AD_DCOP
 PORT AD_FCON = AD_FCON
 PORT AD_FCOP = AD_FCOP
 PORT AD_D_N = AD_D_N
 PORT AD_D_P = AD_D_P
 PORT AD_200MHZ_DELAYREFKLOKKE = clk_200_0000MHzPLL0
 PORT AD_SIGNAL_A = AD_strom_a
END

BEGIN da_omformer_utgang
 PARAMETER INSTANCE = DA_omformer_utgang_0
 PARAMETER HW_VER = 1.00.a
PARAMETER DEFAULTVERDI_OPPSETTREGISTER = 0x0000000F
 PARAMETER BREDDE_INN = 16
 PARAMETER ANTALL_SIGNALKILDER = 2
 PARAMETER C_BASEADDR = 0xcd600000
 PARAMETER C_HIGHADDR = 0xcd60ffff
BUS_INTERFACE SPLB = plb_v46_0
 PORT DA_RW = DA_RW
 PORT DA_D = DA_D
 PORT DA_A_B = DA_A_B
 PORT DA_AB_CS = DA_AB_CS
 PORT DA_CD_CS = DA_CD_CS
 PORT DA_SIGNAL_NY = net_vcc
END

BEGIN vekselretter_tilkobling
 PARAMETER INSTANCE = vekselretter_tilkobling_0
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_BASEADDR = 0xc5620000
 PARAMETER C_HIGHADDR = 0xc562ffff
 BUS_INTERFACE SPLB = plb_v46_0
 PORT ve_paa = VE_PAA
 PORT ve_driver = VE_DRIVER_UT
 PORT ve_ok = VE_OK
 PORT ve_t = VE_T
 PORT led_ve = LED_VE
PORT paa_inn = net_vcc
 PORT driversignal_inn = sv_pwm_ut_T1
END
```

```
BEGIN vekselretter_tilkobling
 PARAMETER INSTANCE = vekselretter_tilkobling_1
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_BASEADDR = 0xc5600000
 PARAMETER C_HIGHADDR = 0xc560ffff
 BUS_INTERFACE SPLB = plb_v46_0
 PORT driversignal_inn = sv_pwm_ut_T2
 PORT ve_t = ve_t1
 PORT ve_ok = ve_ok1
 PORT ve_paa = ve_paa1
 PORT ve_driver = ve_driver1
 PORT led_ve = led_ve1
 PORT paa_inn = net_vcc
END

BEGIN som_phase_ip
 PARAMETER INSTANCE = som_phase_ip_0
 PARAMETER HW_VER = 1.00.a
 PARAMETER C_BASEADDR = 0xcea00000
 PARAMETER C_HIGHADDR = 0xcea0ffff
 BUS_INTERFACE SPLB = plb_v46_0
 PORT Led_port = LED_TEST_s
 PORT pwm_ut_T1 = sv_pwm_ut_T1
 PORT pwm_ut_T2 = sv_pwm_ut_T2
 PORT interupt_port = som_phase_ip_0_interupt_port
END

BEGIN driver_interface_via_dig_io_connection
 PARAMETER INSTANCE = driver_interface_via_dig_io_connection_0
 PARAMETER HW_VER = 1.00.a
 PORT ve_ok = ve_ok1
 PORT ve_paa = ve_paa1
 PORT ve_driver = ve_driver1
 PORT ve_t = ve_t1
 PORT led_ve = led_ve1
 PORT dig_io = DIG_IO3_D
END
```

11 Appendix C

```
--(VHDL codes for SOM_PHASE_IP)---
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
library proc_common_v3_00_a;
use proc_common_v3_00_a.proc_common_pkg.all;

entity user_logic is
  generic
  (
      NO_OF_LEDS          : integer             := 6;
    NO_OF_PHASES      : integer           := 3;
   NO_OF_SWITCHINGS    : integer    := 5;
     C_SLV_DWIDTH           : integer              := 32;
    C_NUM_REG                  : integer             := 100
     );
  port
  (
       Led_port                   : out  std_logic_vector(NO_OF_LEDS-1 downto 0);
      pwm_ut_T1                    : out std_logic_vector(NO_OF_PHASES-1 downto 0);
      pwm_ut_T2                   : out std_logic_vector(NO_OF_PHASES-1 downto 0);
     interupt_port                 : out std_logic;
    Bus2IP_Clk                    : in  std_logic;
    Bus2IP_Reset                   : in  std_logic;
    Bus2IP_Data                   : in  std_logic_vector(0 to C_SLV_DWIDTH-1);
    Bus2IP_BE                     : in  std_logic_vector(0 to C_SLV_DWIDTH/8-1);
    Bus2IP_RdCE                     : in  std_logic_vector(0 to C_NUM_REG-1);
    Bus2IP_WrCE                     : in  std_logic_vector(0 to C_NUM_REG-1);
    IP2Bus_Data                   : out std_logic_vector(0 to C_SLV_DWIDTH-1);
    IP2Bus_RdAck                    : out std_logic;
    IP2Bus_WrAck                    : out std_logic;
    IP2Bus_Error                  : out std_logic
     );
  attribute SIGIS : string;
  attribute SIGIS of Bus2IP_Clk    : signal is "CLK";
  attribute SIGIS of Bus2IP_Reset  : signal is "RST";

end entity user_logic;

------------------------------------------------------------------------------
-- Architecture section
------------------------------------------------------------------------------
architecture IMP of user_logic is
```

```
  --USER signal declarations added here, as needed for user logic
  ------------------------------------------
  -- Signals for user logic slave model s/w accessible register example
  ------------------------------------------
  signal slv_reg0                    : std_logic_vector(0 to C_SLV_DWIDTH-1);
  signal slv_reg1                    : std_logic_vector(0 to C_SLV_DWIDTH-1);
    .
     .
  signal slv_reg98                   : std_logic_vector(0 to C_SLV_DWIDTH-1);
  signal slv_reg99                   : std_logic_vector(0 to C_SLV_DWIDTH-1);
  signal slv_reg_write_sel           : std_logic_vector(0 to 99);
  signal slv_reg_read_sel            : std_logic_vector(0 to 99);
  signal slv_ip2bus_data             : std_logic_vector(0 to C_SLV_DWIDTH-1);
  signal slv_read_ack                : std_logic;
  signal slv_write_ack               : std_logic;
 signal Max_Counter                  : std_logic_vector( C_SLV_DWIDTH-1 downto 0);
type vector_array_time is array ( 0 to NO_OF_SWITCHINGS-1)of
std_logic_vector(C_SLV_DWIDTH-1 downto 0);
type vector_array_state is array ( 0 to NO_OF_SWITCHINGS-1) of std_logic_vector( 1 downto
0);
signal Rmemory_time                  :vector_array_time;
signal Rmemory_state                 :vector_array_state;
signal Ymemory_time                  :vector_array_time;
signal Ymemory_state                 :vector_array_state;
signal Bmemory_time                  :vector_array_time;
signal Bmemory_state                 :vector_array_state;
signal Rcount_down                   : integer;
signal Ycount_down                   : integer;
signal Bcount_down                   : integer;
type initial is array (0 to NO_OF_PHASES-1) of std_logic_vector(1 downto 0);
signal initial_value: initial:=("01", "11" ,"01");
signal Rn: integer:=0;
signal Yn: integer:=0;
signal Bn: integer:=0;
signal testsignal                  : std_logic_vector(5 downto 0);
signal pwm               : std_logic_vector(NO_OF_PHASES-1 downto 0);
signal pwm_com                     : std_logic_vector(NO_OF_PHASES-1 downto 0);
signal pwm_T1                      : std_logic_vector(NO_OF_PHASES-1 downto 0);
signal pwm_T2                      : std_logic_vector(NO_OF_PHASES-1 downto 0);
signal interupt_out     : std_logic:='0';
--- Here reading and writing in the 100 register take place which is removed in this thesis
because it is bulky
  --Main Part begins#######################################################
Max_Counter <= slv_reg60;
---R phase----------------
Rphase:process(Bus2IP_Clk)
```

```
begin
if(Bus2IP_Clk'event and Bus2IP_Clk='1') then
interupt_out<='0';
          if (Rcount_down<Max_Counter) then
                              if (Rcount_down=0) then
                              Rmemory_time(0) <=slv_reg0; -- time_R0
                              Rmemory_time(1) <=slv_reg1; -- time_R1
                              Rmemory_time(2) <=slv_reg2; -- time_R2
                              Rmemory_time(3) <=slv_reg3; -- time_R3
                              Rmemory_time(4) <=slv_reg4; -- time_R4

                              Rmemory_state(0) <=slv_reg30(30 to 31); -- state_R0
                              Rmemory_state(1) <=slv_reg31(30 to 31); -- state_R1
                              Rmemory_state(2) <=slv_reg32 (30 to 31);  -- state_R2
                              Rmemory_state(3) <=slv_reg33 (30 to 31);  -- state_R3
                              Rmemory_state(4) <=slv_reg34 (30 to 31);  -- state_R4

                              testsignal(5 downto 4) <= initial_value(0);
                              end if;

                              if (Rcount_down=Rmemory_time(Rn)) then
                                 testsignal(5 downto 4)<=Rmemory_state(Rn);
                                       if (Rn<NO_OF_SWITCHINGS-1) then
                                       Rn<=Rn+1;
                                       else
                                       Rn<=0;
                                       end if;
                              end if;
           Rcount_down<=Rcount_down+1;
          else
          Rn<=0;
          Rcount_down<=0;
          initial_value(0)<=testsignal(5 downto 4);
          interupt_out<='1';
end if;
end if;
end process;
------------------------Y Phase-----------------------------------------
Yphase:process(Bus2IP_Clk)
          begin
                    if(Bus2IP_Clk'event and Bus2IP_Clk='1') then
          if (Ycount_down<Max_Counter) then
                              if (Ycount_down=0) then
                               Ymemory_time(0) <=slv_reg10; -- time_Y0
                              Ymemory_time(1) <=slv_reg11; -- time_Y1
                              Ymemory_time(2) <=slv_reg12; -- time_y2
                              Ymemory_time(3) <=slv_reg13; -- time_Y3
                              Ymemory_time(4) <=slv_reg14; -- time_Y4

                              Ymemory_state(0) <=slv_reg40 (30 to 31);  -- state_Y0
                              Ymemory_state(1) <=slv_reg41 (30 to 31);  -- state_Y1
                              Ymemory_state(2) <=slv_reg42 (30 to 31);  -- state_Y2
                              Ymemory_state(3) <=slv_reg43 (30 to 31);  -- state_Y3
                              Ymemory_state(4) <=slv_reg44 (30 to 31);  -- state_Y4

                              testsignal(3 downto 2) <= initial_value(1);
                                 end if;

                              if (Ycount_down=Ymemory_time(Yn)) then
                                 testsignal(3 downto 2)<=Ymemory_state(Yn);
                                     if (Yn<NO_OF_SWITCHINGS-1) then
                                     Yn<=Yn+1;
                                     else
                                     Yn<=0;
                                     end if;
                              end if;

                   Ycount_down<=Ycount_down+1;
                  Else
                  Yn<=0;
                  Ycount_down<=0;
                  initial_value(1)<=testsignal(3 downto 2);
            end if;
            end if;
            end process;
            -------------------------------------------------------------------
            Bphase:process(Bus2IP_Clk)
                  begin
                        if(Bus2IP_Clk'event and Bus2IP_Clk='1') then
                  if (Bcount_down<Max_Counter) then
                              if (Bcount_down=0) then
                              Bmemory_time(0) <=slv_reg20; -- time_B0
                              Bmemory_time(1) <=slv_reg21; -- time_B1
                              Bmemory_time(2) <=slv_reg22; -- time_B2
                              Bmemory_time(3) <=slv_reg23; -- time_B3
                              Bmemory_time(4) <=slv_reg24; -- time_B4

                              Bmemory_state(0) <=slv_reg50 (30 to 31);  -- state_B0
                              Bmemory_state(1) <=slv_reg51 (30 to 31);  -- state_B1
                              Bmemory_state(2) <=slv_reg52 (30 to 31);  -- state_B2
                              Bmemory_state(3) <=slv_reg53 (30 to 31);  -- state_B3
                              Bmemory_state(4) <=slv_reg54 (30 to 31);  -- state_B4
```

```
                    testsignal(1 downto 0) <= initial_value(2);
                    end if;

            if (Bcount_down=Bmemory_time(Bn)) then
                    testsignal(1 downto 0)<=Bmemory_state(Bn);
                           if (Bn<NO_OF_SWITCHINGS-1) then
                        Bn<=Bn+1;
                         else
                        Bn<=0;
                         end if;
            end if;

    Bcount_down<=Bcount_down+1;
else
    Bcount_down<=0;
    initial_value(2)<=testsignal(1 downto 0);
end if;
end if;
end process;
----giving the switching pulse to three level converter--------------assuming R= bit 5-4, Y=bit 3-2,
B= bit 1-0
------signal for T1--------
pwm_T1(0) <=testsignal(5);  --R phase
pwm_T1(1) <=testsignal(3);  --Y phase
pwm_T1(2) <=testsignal(1); -- B phase
------signal for T2--------
pwm_T2(0) <=testsignal(4);  --R phase
pwm_T2(1) <=testsignal(2);  --Y phase
pwm_T2(2) <=testsignal(0); -- B phase

Led_port<=testsignal;
pwm_ut_T1<= pwm_T1;
pwm_ut_T2<= pwm_T2;
interupt_port<=interupt_out;
--
##################################################################################
##########
end IMP;
```

12 Appendix D

--VHDL codes for SOM_spacevector_IP

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
library proc_common_v3_00_a;
use proc_common_v3_00_a.proc_common_pkg.all;

entity user_logic is
  generic
  (
      NO_OF_LEDS          : integer            := 6;
    NO_OF_PHASES      : integer            := 3;
     C_SLV_DWIDTH            : integer             := 32;
    C_NUM_REG                  : integer             := 30
     );
  port
  (
        Led_port                   : out  std_logic_vector(NO_OF_LEDS-1 downto 0);
      pwm_ut_T1                     : out std_logic_vector(NO_OF_PHASES-1 downto 0);
      pwm_ut_T2                    : out std_logic_vector(NO_OF_PHASES-1 downto 0);
     interupt_port                 : out std_logic;
    Bus2IP_Clk                    : in  std_logic;
    Bus2IP_Reset                  : in  std_logic;
    Bus2IP_Data                   : in  std_logic_vector(0 to C_SLV_DWIDTH-1);
    Bus2IP_BE                     : in  std_logic_vector(0 to C_SLV_DWIDTH/8-1);
    Bus2IP_RdCE                     : in  std_logic_vector(0 to C_NUM_REG-1);
    Bus2IP_WrCE                     : in  std_logic_vector(0 to C_NUM_REG-1);
    IP2Bus_Data                   : out std_logic_vector(0 to C_SLV_DWIDTH-1);
    IP2Bus_RdAck                    : out std_logic;
    IP2Bus_WrAck                    : out std_logic;
    IP2Bus_Error                  : out std_logic
     );
  attribute SIGIS : string;
  attribute SIGIS of Bus2IP_Clk    : signal is "CLK";
  attribute SIGIS of Bus2IP_Reset  : signal is "RST";

end entity user_logic;

------------------------------------------------------------------------------
-- Architecture section
------------------------------------------------------------------------------
```

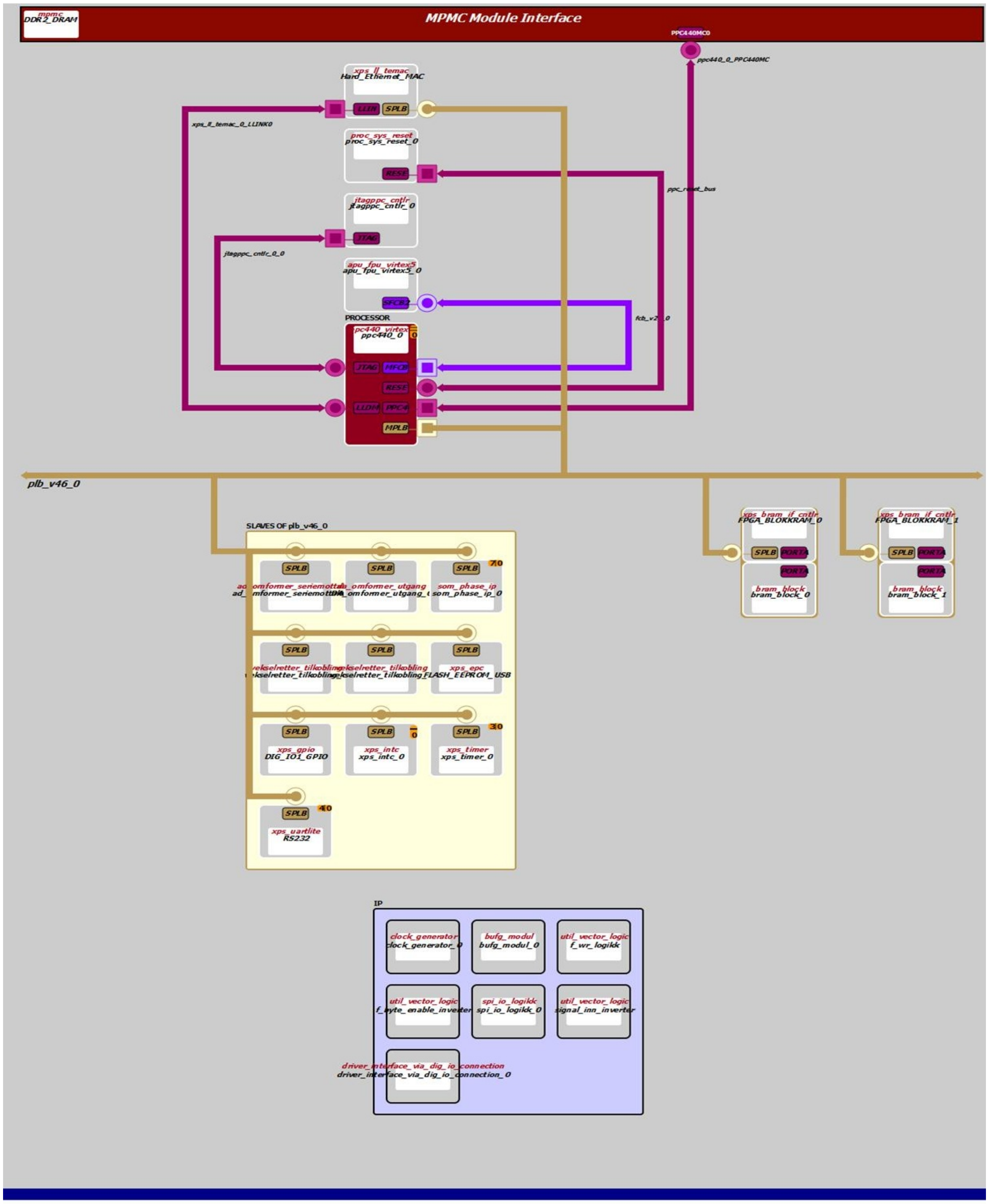
```
architecture IMP of user_logic is
  --USER signal declarations added here, as needed for user logic
  ------------------------------------------
  -- Signals for user logic slave model s/w accessible register example
  ------------------------------------------
  signal slv_reg0                    : std_logic_vector(0 to C_SLV_DWIDTH-1);
  signal slv_reg1                    : std_logic_vector(0 to C_SLV_DWIDTH-1);
    .
     .
  signal slv_reg28                   : std_logic_vector(0 to C_SLV_DWIDTH-1);
  signal slv_reg29                   : std_logic_vector(0 to C_SLV_DWIDTH-1);
  signal slv_reg_write_sel               : std_logic_vector(0 to 99);
  signal slv_reg_read_sel                : std_logic_vector(0 to 99);
  signal slv_ip2bus_data                 : std_logic_vector(0 to C_SLV_DWIDTH-1);
  signal slv_read_ack                  : std_logic;
  signal slv_write_ack                 : std_logic;

signal led_reg_Tm                      : std_logic_vector(C_SLV_DWIDTH-1 downto 0);

signal led_reg_T1                     : std_logic_vector(C_SLV_DWIDTH-1 downto 0);
signal led_reg_T2                     : std_logic_vector(C_SLV_DWIDTH-1 downto 0);
signal led_reg_T3                     : std_logic_vector(C_SLV_DWIDTH-1 downto 0);
signal led_reg_T4                     : std_logic_vector(C_SLV_DWIDTH-1 downto 0);
signal led_reg_T5                     : std_logic_vector(C_SLV_DWIDTH-1 downto 0);
signal led_reg_T6                     : std_logic_vector(C_SLV_DWIDTH-1 downto 0);
signal led_reg_T7                     : std_logic_vector(C_SLV_DWIDTH-1 downto 0);


signal led_reg_S1                     : std_logic_vector(C_SLV_DWIDTH-1 downto 0);
signal led_reg_S2                     : std_logic_vector(C_SLV_DWIDTH-1 downto 0);
signal led_reg_S3                     : std_logic_vector(C_SLV_DWIDTH-1 downto 0);
signal led_reg_S4                     : std_logic_vector(C_SLV_DWIDTH-1 downto 0);
signal led_reg_S5                     : std_logic_vector(C_SLV_DWIDTH-1 downto 0);
signal led_reg_S6                     : std_logic_vector(C_SLV_DWIDTH-1 downto 0);
signal led_reg_S7                     : std_logic_vector(C_SLV_DWIDTH-1 downto 0);


signal state_count                    : std_logic_vector(C_SLV_DWIDTH-1 downto 0);
signal clk                             : std_logic:='0';
signal testsignal                     : std_logic_vector(5 downto 0) := (others => '0');
type    states        is                (zero, one, two, three, four, five, six);
signal mem_state                       : states;
signal count_down                      : integer:=0;
signal pwm_T1                        : std_logic_vector(NO_OF_PHASES-1 downto 0);
signal pwm_T2                      : std_logic_vector(NO_OF_PHASES-1 downto 0);
 signal interupt_out                 : std_logic:='0';
```

```
begin

--- Here reading and writing in the 100 register take place which is removed in this thesis
because it is bulky
  --Main Part begins#####################################################
led_reg_Tm <= slv_reg1;

-----lower section:----
process(Bus2IP_Clk)
begin
if(Bus2IP_Clk'event and Bus2IP_Clk='1') then
interupt_out<='0';
            if (count_down<led_reg_Tm) then
                        if (count_down= 0) then
led_reg_T1 <= slv_reg2;
led_reg_T2 <= slv_reg3;
led_reg_T3 <= slv_reg4;
led_reg_T4 <= slv_reg5;
led_reg_T5 <= slv_reg6;
led_reg_T6 <= slv_reg7;
led_reg_T7 <= slv_reg8;

led_reg_S1 <= slv_reg9;
led_reg_S2 <= slv_reg10;
led_reg_S3 <= slv_reg11;
led_reg_S4 <= slv_reg12;
led_reg_S5 <= slv_reg13;
led_reg_S6 <= slv_reg14;
led_reg_S7 <= slv_reg15;

end if;

   count_down<=count_down+1;

if (count_down= led_reg_T1) then
mem_state<=zero;
end if;

if (count_down= led_reg_T2) then
mem_state<=one;
end if;

if (count_down= led_reg_T3) then
mem_state<=two;
  end if;
```

```
if (count_down= led_reg_T4) then
mem_state<=three;
end if;

if (count_down= led_reg_T5) then
mem_state<=four;
end if;

if (count_down= led_reg_T6) then
mem_state<=five;
end if;

if (count_down= led_reg_T7) then
mem_state<=six;
end if;

else
count_down<=0;
 interupt_out<='1';
end if;
end if;
end process;

----upper section:----
process(mem_state)
begin
case mem_state is

when zero=>
state_count<=led_reg_S1;


when one=>
state_count<=led_reg_S2;

when two=>
state_count<=led_reg_S3;

when three=>
state_count<=led_reg_S4;

when four=>
state_count<=led_reg_S5;

when five=>
```

```
state_count<=led_reg_S6;

when six=>
state_count<=led_reg_S7;
end case;
end process;
Led_port<=state_count(NO_OF_LEDS-1 downto 0);

----giving the switching pulse to three level converter--------------assuming R= bit 5-4, Y=bit 3-2,
B= bit 1-0
------signal for T1--------
pwm_T1(0) <=testsignal(5);  --R phase
pwm_T1(1) <=testsignal(3);  --Y phase
pwm_T1(2) <=testsignal(1); -- B phase
------signal for T2--------
pwm_T2(0) <=testsignal(4);  --R phase
pwm_T2(1) <=testsignal(2);  --Y phase
pwm_T2(2) <=testsignal(0); -- B phase

pwm_ut_T1<= pwm_T1;
pwm_ut_T2<= pwm_T2;
interupt_port<=interupt_out;
--
####################################################################################
##########
end IMP;
```

13 Appendix E



Figur 13-1 Block diagram of hardware.

14 Appendix F

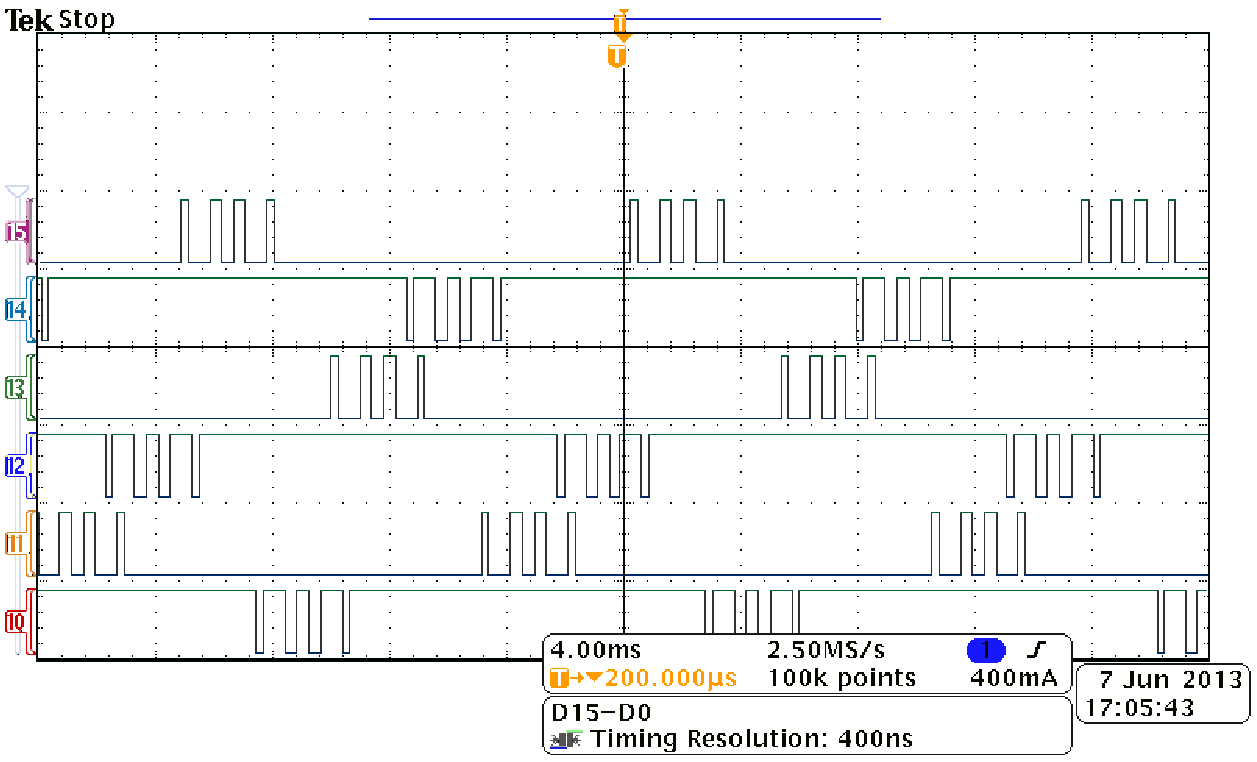


Figure 14-1 Digital signal for m=0.3, N=4.

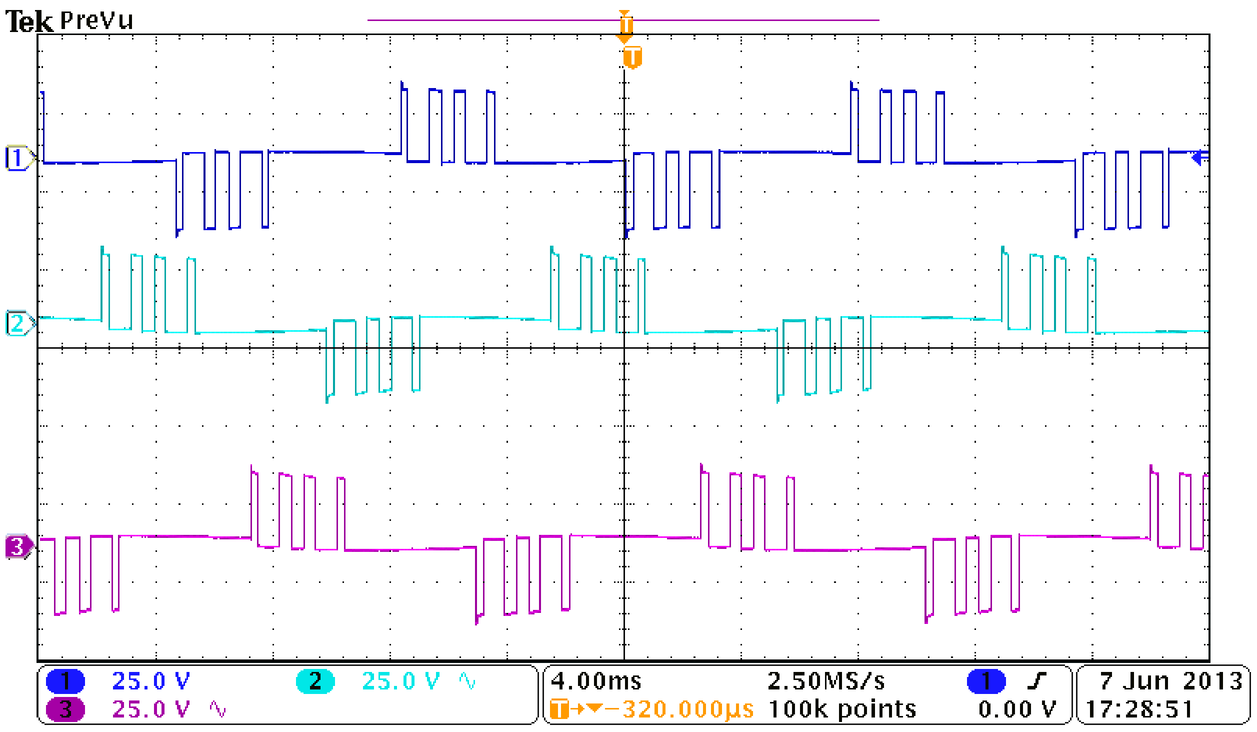


Figure 14-2 Voltage signal for m=0.3, N=4.

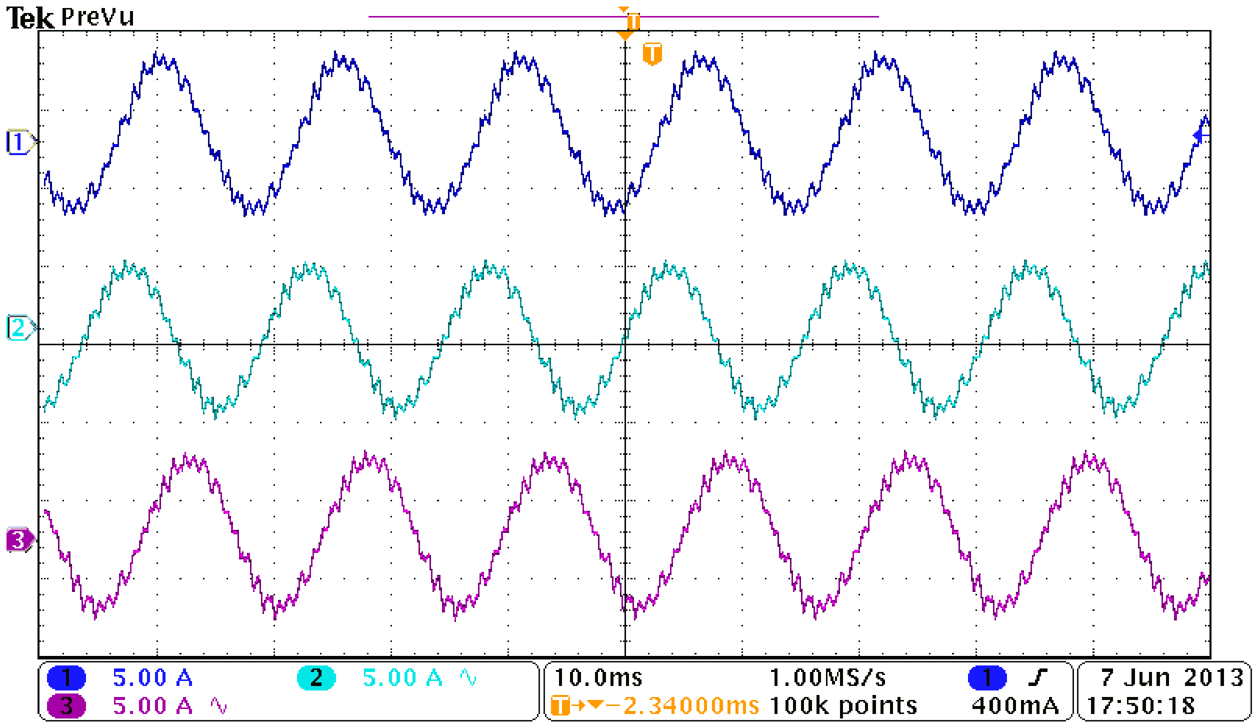


Figure 14-3 Current signal for m=0.3, N=4.

15 Appendix G

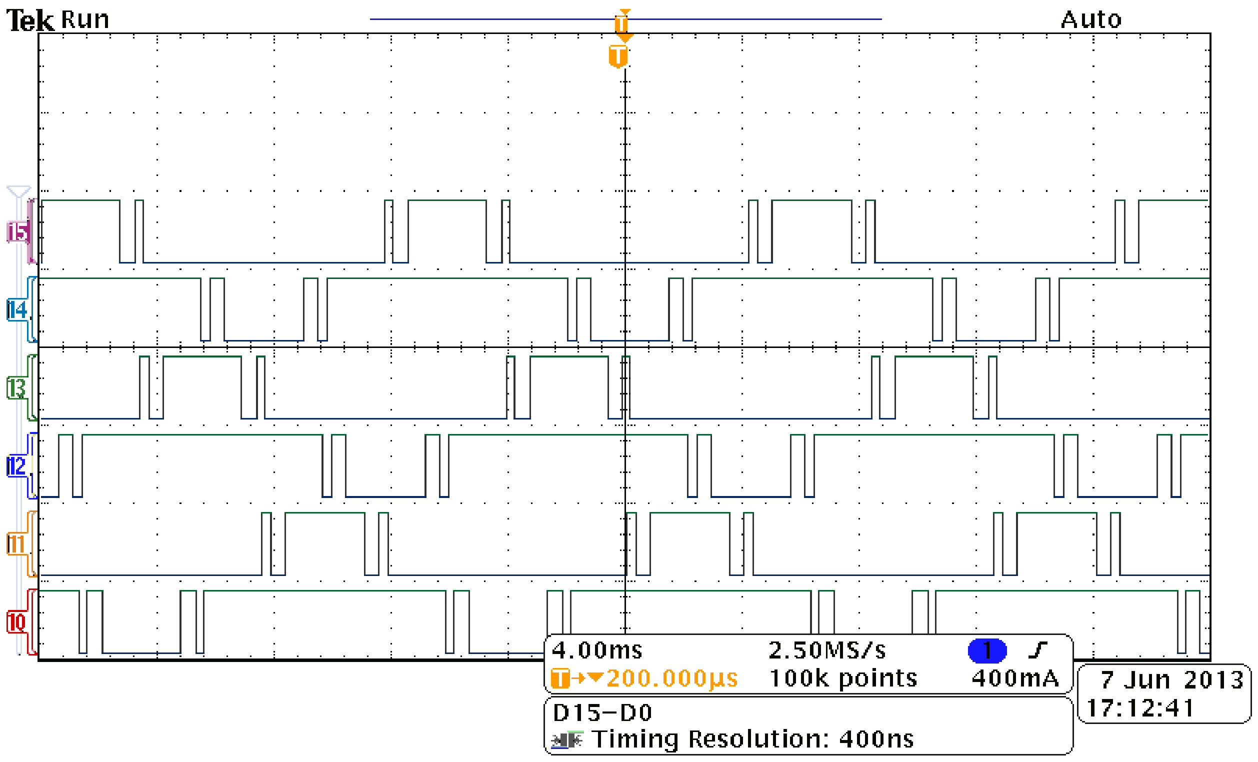


Figure 15-1 Digital Signal for m=0.87, N=3.

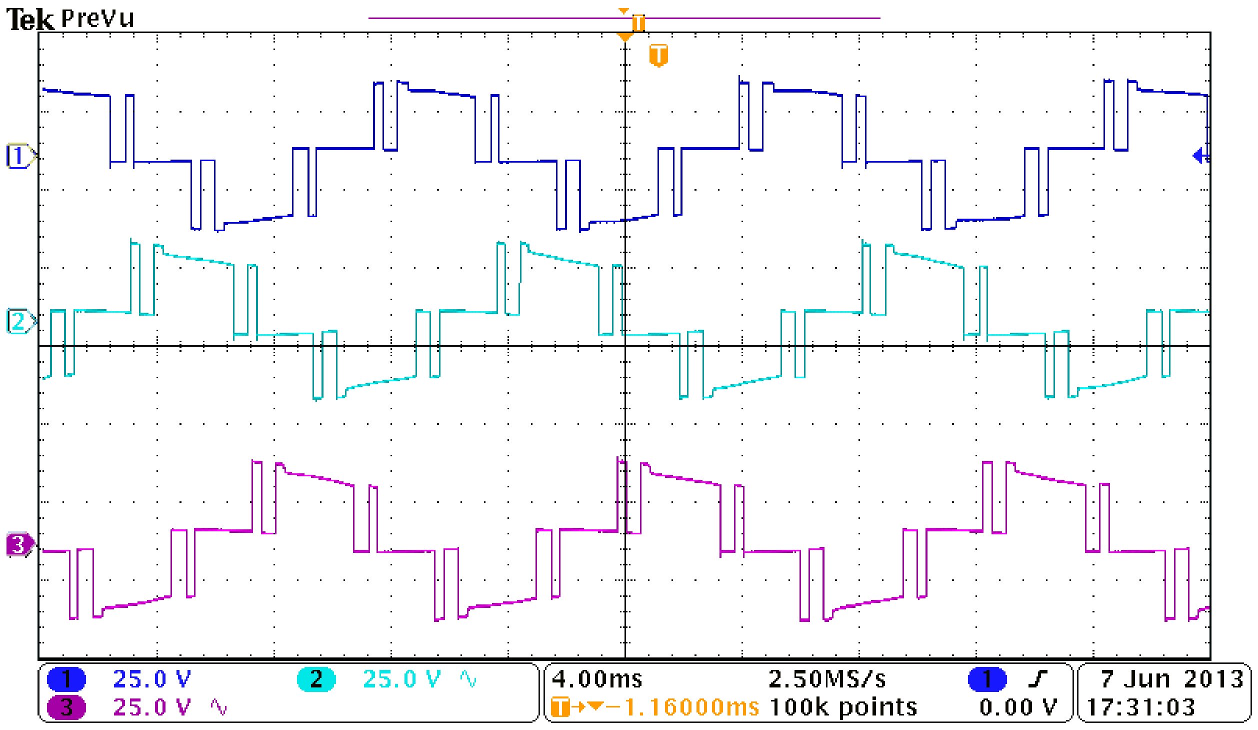


Figure 15-2 Voltage Signal for m=0.87, N=3.

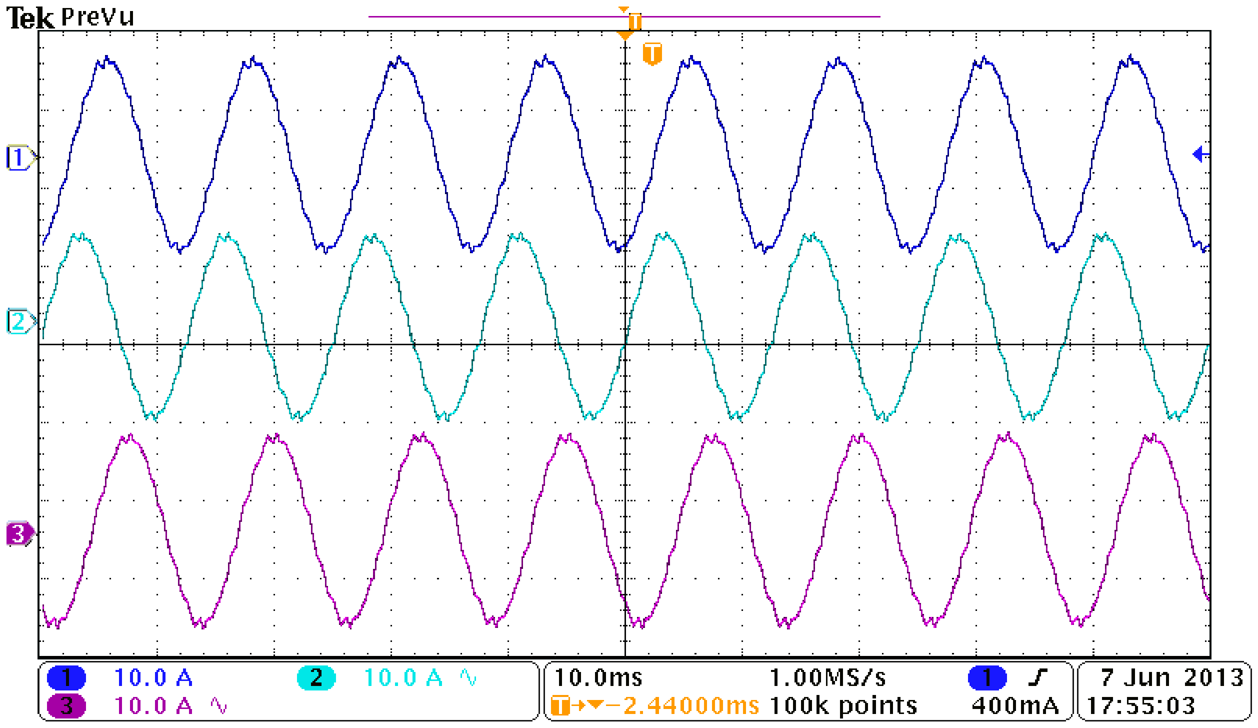


Figure 15-3 Current Signal for m=0.87, N=3.

16 Appendix H

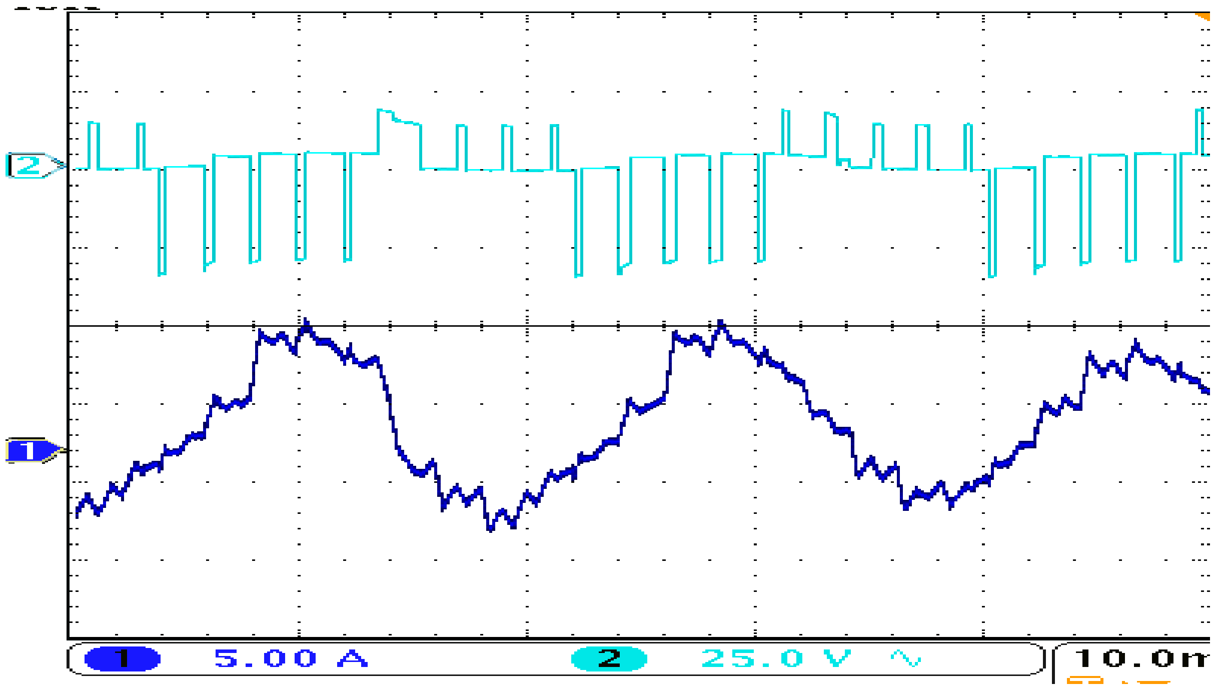


Figure 16-1 Current and Voltage for m=0.3, f=55Hz, N=5.

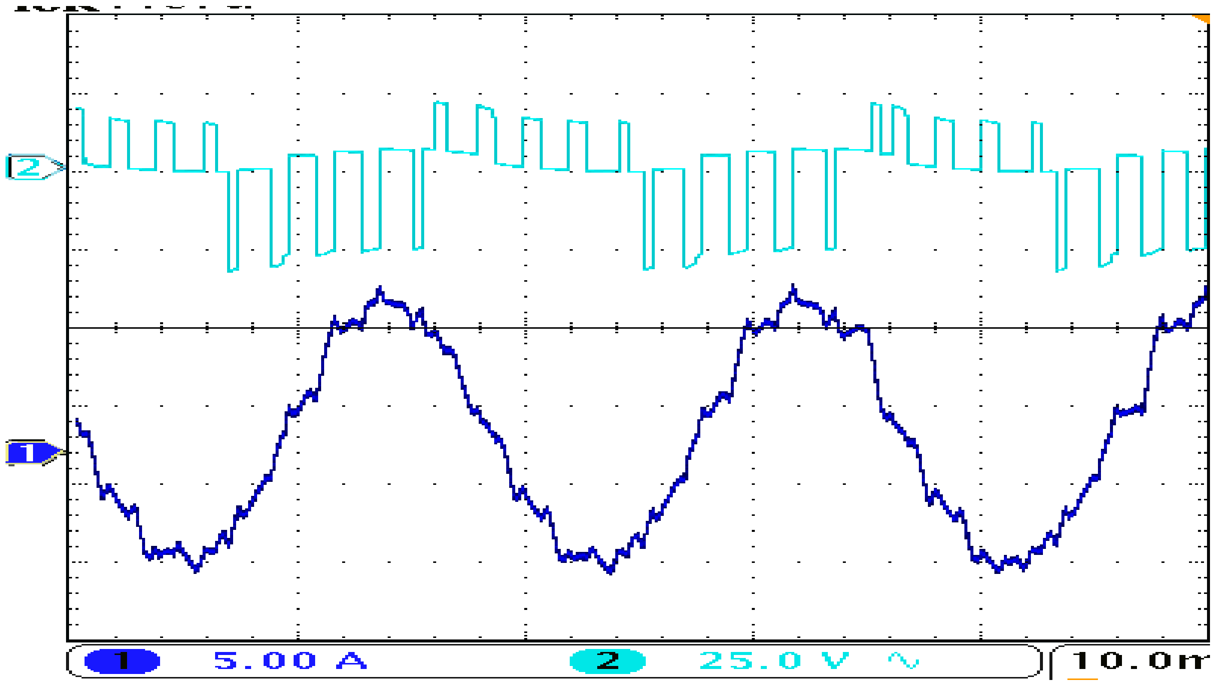


Figure 16-2 Current and Voltage for m=0.5, f=55Hz, N=5.

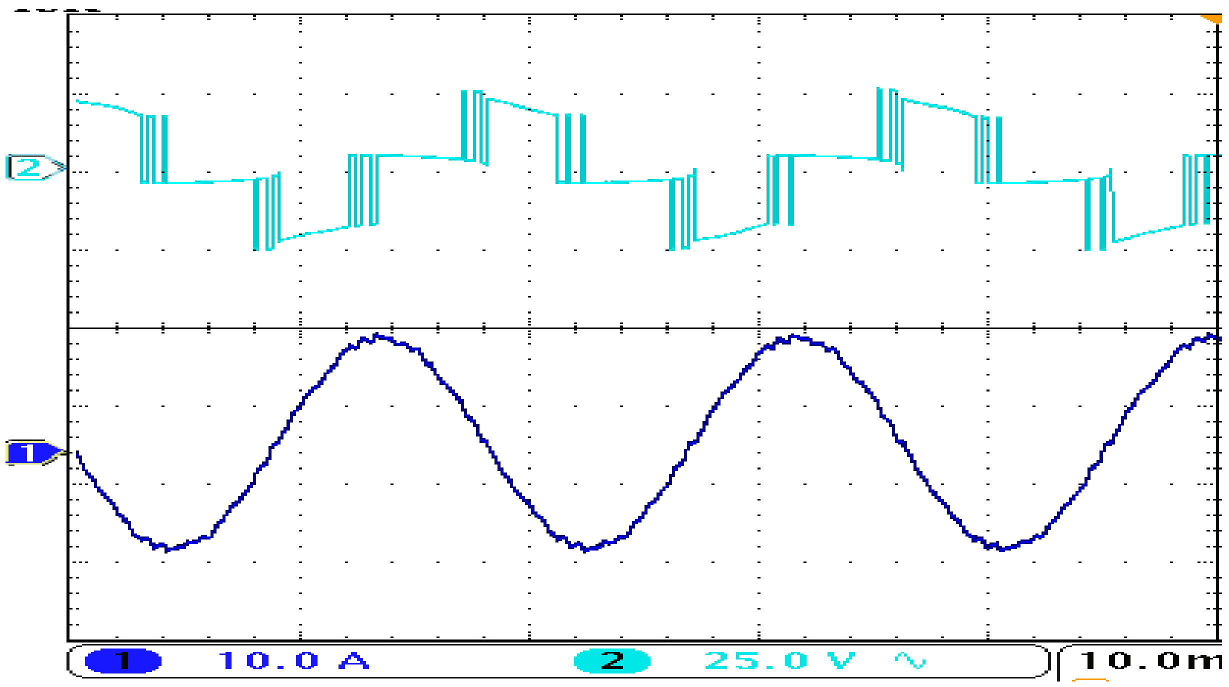


Figure 16-3 Current and Voltage for m=0.8, f=55Hz, N=5.

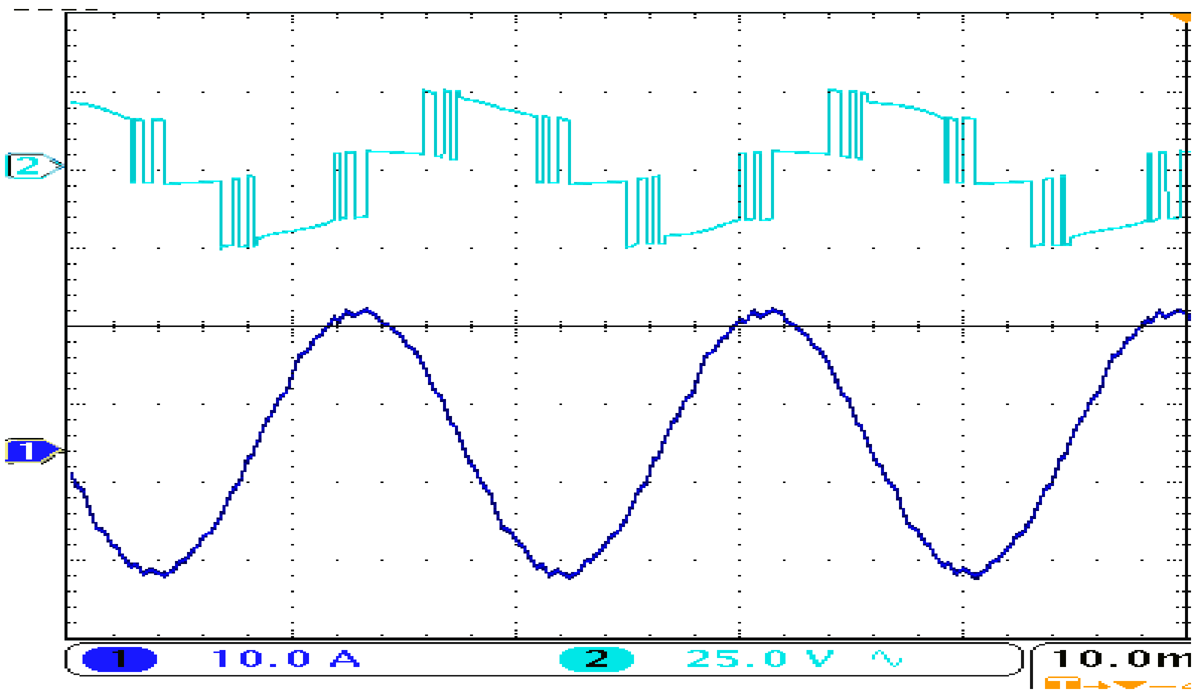


Figure 16-4 Current and Voltage for m=1, f=55Hz, N=5.

17 Appendix I

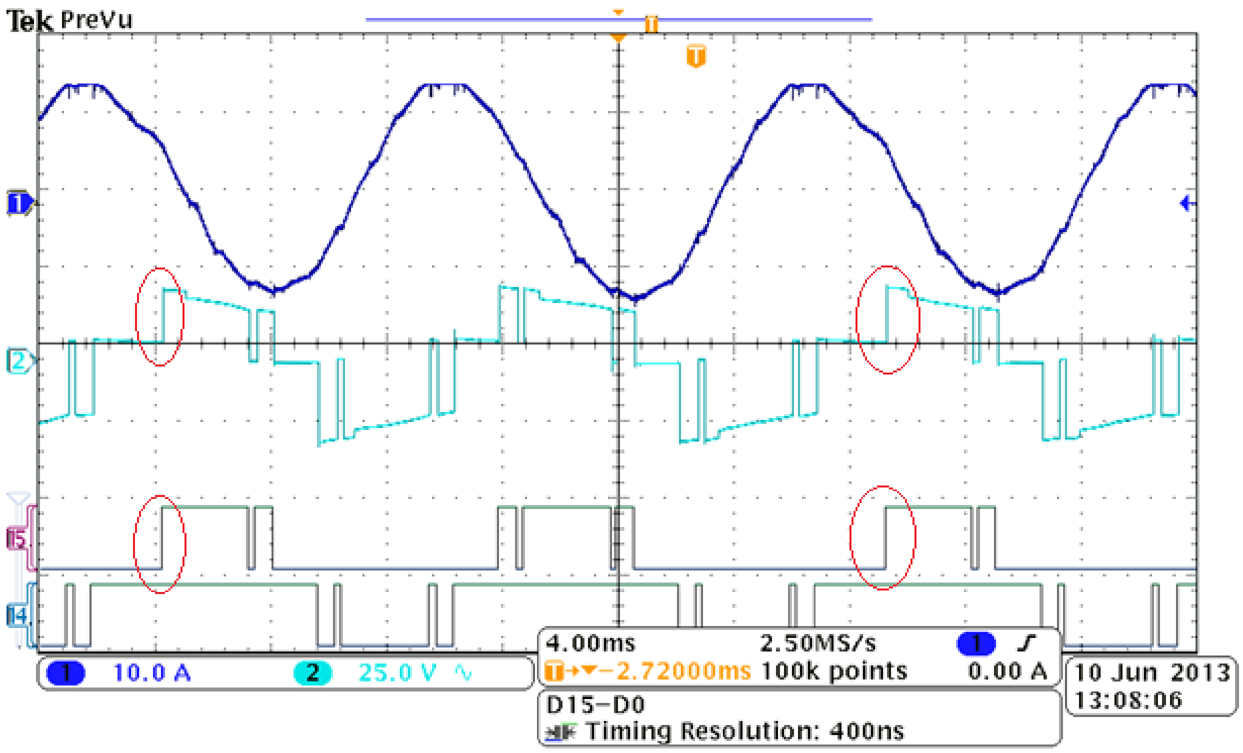


Figure 17-1 Missing Pulse for m=1, f=80 Hz.

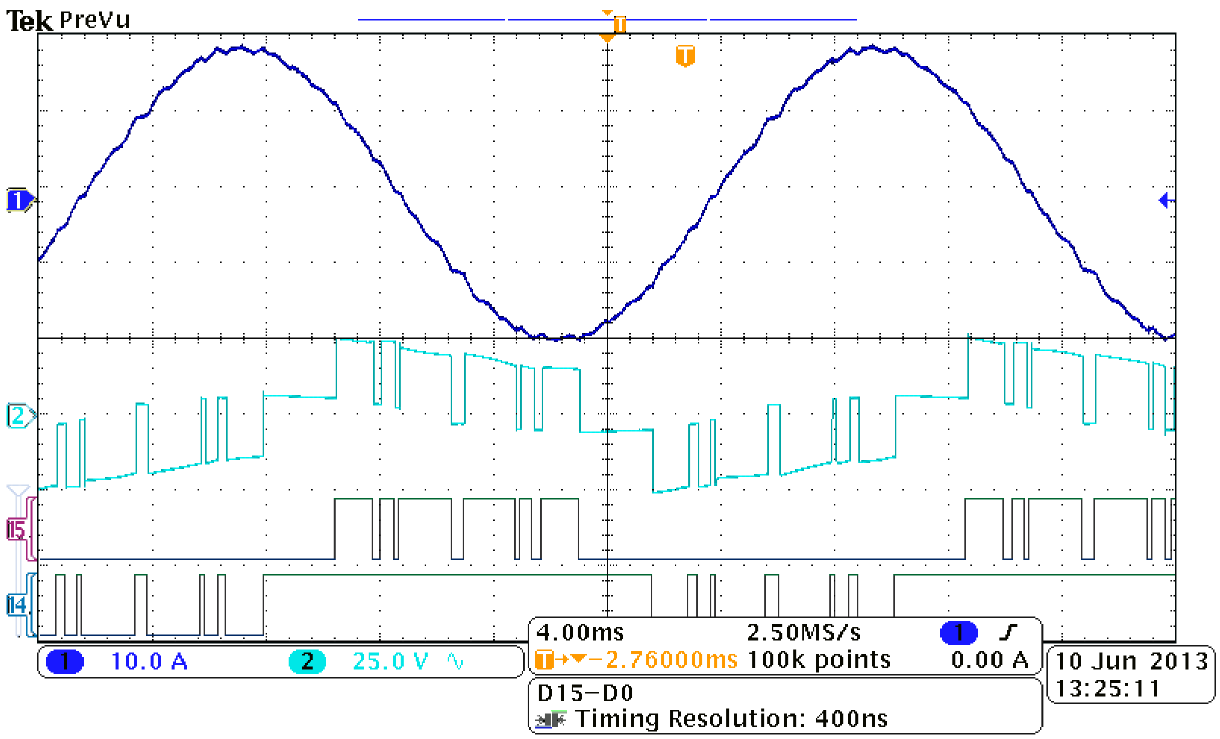


Figure 17-2 No Missing Pulse for m=1, f=45Hz.