UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN GÉNIE ÉLECTRIQUE

PAR
OMID MOHAGHEGH DOUST

CONVERSION DE PUISSANCE DIRECTE AVEC FAIBLE DISTORSION DE COURANT D'ENTRÉE POUR UN SYSTÈME MULTIMOTEUR

MAI 2008

Abstract

Implementation of an effective solution in Adjustable Speed Drive (ASD), for reduction of Total Harmonic Distortion (THD) level in input line current, is currently the object of many efforts of scientists and engineers in power electronics.

In this thesis, a popular converter structure, two-stage Direct Power Electronic Converter (DPEC), and a popular control strategy, Space Vector Modulation (SVM), are presented as an effective solution for ASDs.

The solution has been simulated in the real-time simulation environments Xilinx System Generator™ (XSG™) and RT-LAB™. The results show an effective reduction of the THD level in input line current and output voltage, sinusoidal input/output, with respect to other current solutions.

Hardware Design Language (HDL) codes are generated according to the control algorithm applied for the PWM pattern generation. HDL codes are analyzed for the purpose of FPGA implementation, because of FPGAs' higher solution speed and capability in the mathematical complex equations than microcontrollers.

The result of codes synthesis, with Leonardo Spectrum software, is presented and an appropriate FPGA, with correspondent capacity and code volume, is selected.

As the result of this research, simulations of this solution in two real-time workshops and HDL codes generation with synthesis have been realised.

Preface

Paper [5] covers a big part of my interests in the field of control and power electronics and, at the time this research thesis was started, it presented some of the latest research in the field of motor control by power converters and presented one of the best current solutions, so that I was interested to do more research and to develop that.

I really thank Mr. Pierre Sicard, my thesis advisor for this research, who has done his best to direct me towards success in this research goal, so that I have practiced, realized and developed this thesis.

I also thank my wife who has accompanied me along with realization of this research.

Great thanks to "Groupe de Recherche en Électronique Industrielle" for the financial support to this research.

Table of content

List of tables

List of figures

Résumé du travail de recherche

1- INTRODUCTION

Présentement beaucoup d'efforts sont faits dans l'intégration de dispositifs d'électronique de puissance pour l'entraînement des moteurs électriques pour produire un convertisseur électromécanique efficace et flexible, qui permet une incorporation rapide dans les processus industriels et un temps de démarrage court. Beaucoup de recherches sur les topologies de convertisseurs et leurs stratégies de commande sont actuellement en cours pour atteindre les buts suivants: taille optimale et très compacte, économies en fabrication, taux de distorsion harmonique (TDH) moins élevé dans le courant d'entrée de la ligne, meilleure efficacité énergétique, possibilité de contrôler le moteur en mode quatre quadrants, facteur de puissance proche de l'unité, moins de sensibilité aux perturbations du réseau, contrôle précis des moteurs dans le cas des systèmes multimoteurs [1], [9]-[17].

Dans les systèmes d'entraînement multimoteurs à courant alternatif, avec l'alimentation triphasée et bus cc commun, un problème important de qualité d'alimentation est le contrôle des distorsions de courant de ligne qui engendrent de la pollution harmonique dans le réseau. La qualité du courant d'entrée est caractérisée par le niveau TDH qui est produit par l'interaction de l'entraînement à vitesse variable (EVV) avec le réseau [21]. Il y a ainsi beaucoup de recherche sur la mise au point de nouvelles topologies de convertisseurs (la modulation) comme une solution pour améliorer l'interaction entre le réseau et les EVV.

L'objectif de ce travail est d'évaluer une méthode de modulation qui permet de réduire les distorsions du courant d'entrée dans un système multimoteur en contrôlant les séquences des commutations des onduleurs, ainsi que son implémentation dans un circuit programmable (*Field-Programmable Gate Array,* FPGA). En utilisant un FPGA, on peut potentiellement obtenir un niveau de TDH moindre dans le courant d'entrée de la ligne qu'avec un microcontrôleur (*Peripheral Interface Controller,* PIC), car le FPGA est plus rapide pour générer des ondes porteuses et commander les interrupteurs (*Insulated Gate Bipolar Transistors,* IGBT) du convertisseur. Ce gain est cependant limité par la rapidité des circuits pilotes.

La haute capacité de programmation, la vitesse de calcul des équations complexes, des instructions de programmation plus simples, la possibilité de générer des codes facilement par le logiciel, la synthèse des codes et le choix optimal de circuit avant la fabrication sont les avantages d'implémentation des FPGA dans un circuit au lieu de microcontrôleurs.

Des travaux de simulation permettront de réaliser une validation de la loi de modulation proposée et une comparaison préliminaires avec des lois classiques. Les expériences de ce travail sont comme suit : la considération d'une solution efficace au niveau du contrôle des moteurs à induction (MI) triphasés avec la pratique et le développement de la méthodologie de simulation (TDH moindre dans le courant d'entrée que les résultats d'autres auteurs) dans les environnements Matlab®, Simulink™, RT-LAB™, Xilinx System Generator™ (XSG™) en temps réel; la conception d'une méthode efficace pour la génération aisée des codes (*Hardware Description Language,* HDL) qui pourront être implémentés dans un FPGA; la synthèse de cette loi afin de valider cette implémentation.

Les simulations sur un système avec 3 moteurs à induction (3HP, 220V, 60Hz, 1725rpm) sont effectuées. La topologie de convertisseur de puissance direct (*Direct Power Electronic Converter*, DPEC) à deux étages est considérée dans ce travail pour qu'il y ait moins de composant IGBT et de diodes que d'autres topologies. Le DPEC est constitué de deux étages : un étage commun constitué d'un redresseur et un étage comprenant des onduleurs (connectés par le bus cc commun au redresseur), utilisés pour alimenter les moteurs dans le système EVV.

En combinant la technique de modulation vectorielle (*Space Vector Modulation*, SVM) pour deux étages de la topologie de DPEC avec les bonnes séquences de commande des canaux de modulation de largeur d'impulsion (MLI) pour les commutations des interrupteurs IGBT, on arrive aux buts de la recherche de produire moins de distorsions et d'augmenter la qualité du courant d'entrée de la ligne et de la tension de sortie à la charge du convertisseur.

En plus, le DPEC résout les désavantages les plus importants du convertisseur matriciel et il possède d'autres avantages, comme un moindre coût de fabrication, de moindres pertes d'énergie, il est plus compact et il offre la possibilité de contrôle indépendant de quelques moteurs en même temps dans le système d'entraînement multimoteur.

La figure 1 montre le schéma bloc du convertisseur DPEC à deux étages et la méthodologie SVM en combinaison avec la méthode de contrôle Volt/Hz constant en boucle fermée [19]. Dans ce résumé, les différentes méthodes et algorithmes utilisés pour les simulations de DPEC à deux étages par Simulink, RT-LAB et XSG sont expliqués et les résultats (*Fast Fourier Transform,* FFT) pour le courant d'entrée au point de

connexion au réseau et la tension de sortie à la charge sont montrés et comparés en termes de niveau du contenu d'harmonique.

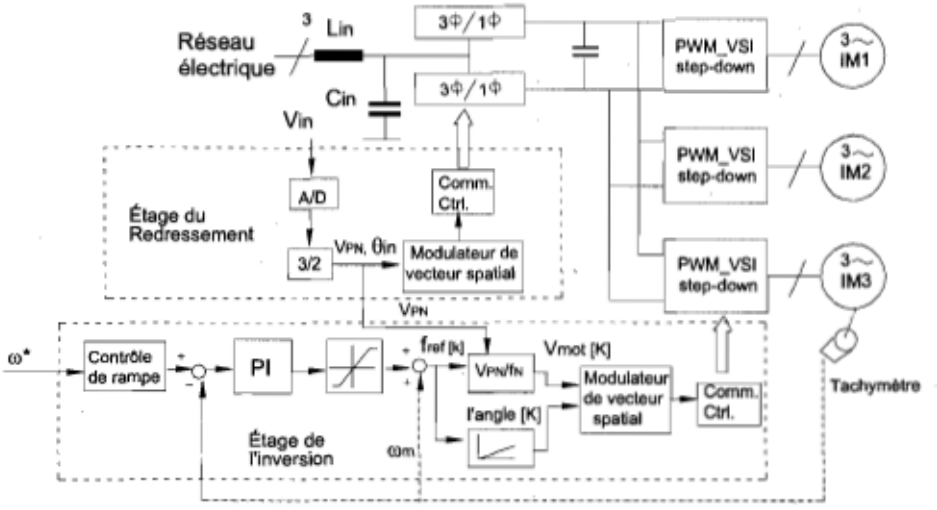


Figure 1 Méthode de contrôle Volts/Hz constant en boucle fermée sur DPEC à deux étages

Pour expliquer pourquoi la structure de redresseur est composée de deux modules 3ϕ/1ϕ, la figure 2 montre la structure contenant douze diodes et douze IGBT. La moitié en haut, comme un module 3ϕ/1ϕ, est constituée de six diodes et six IGBT. Elle doit fournir un courant positif au bus cc. Par contre, la moitié en bas, doit fournir un courant négatif [5].

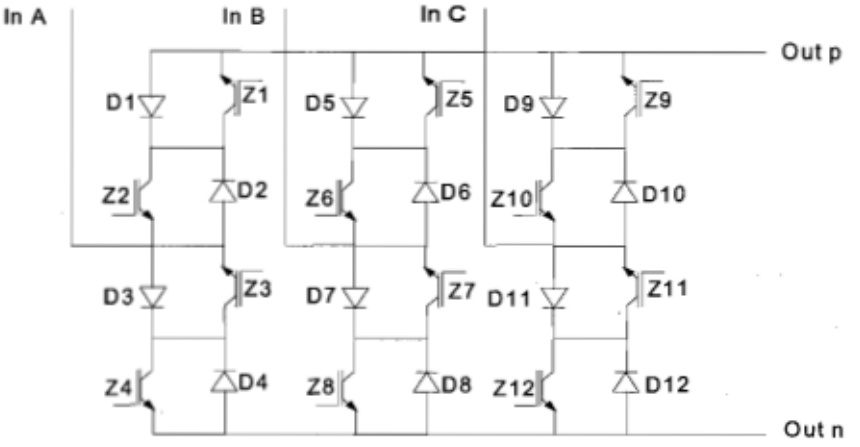
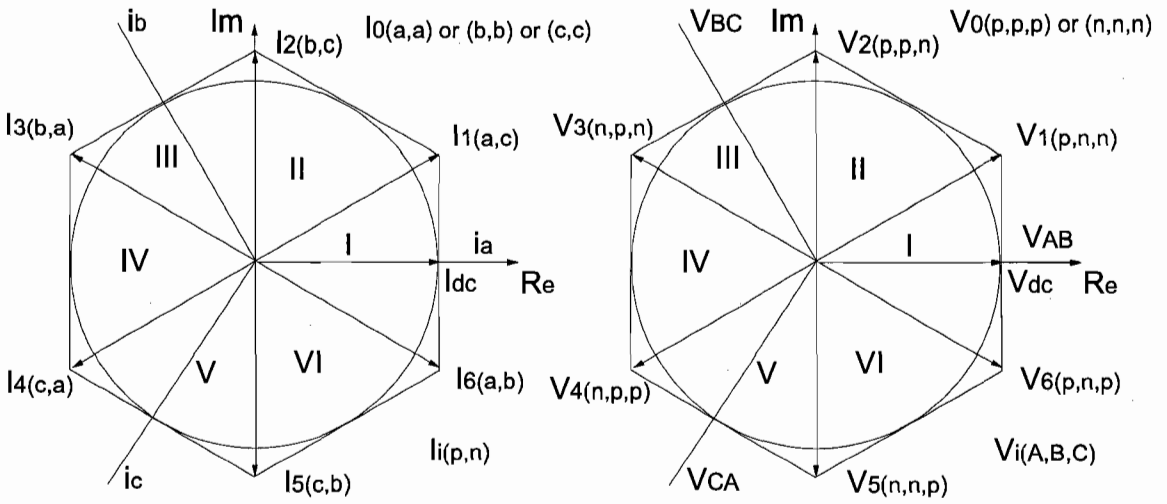


Figure 2 Étage de redressement, 3ϕ/2ϕ , constitué de 12-IGBT et 12-diodes



(a) Hexagone VSR-SVM pour l'étage de redressement

(b) Hexagone VSI-SVM pour l'étage d'inversion

Figure 3 Principe de la commande vectorielle

La méthodologie de SVM de l'étage redresseur est définie par le vecteur du courant d'entrée et celle de l'étage onduleur par le vecteur de la tension de sortie [7], Figure 3, tel que la combinaison de SVM des deux étages, Figure 4, définisse les rapports cycliques pertinents donnés par les équations (1)-(5), [22]. Afin d'arriver aux objectifs de la recherche, il faut suivre les séquences pertinentes des commutations montrées dans la figure 5 qui causent le moins d'interaction entre le convertisseur et le réseau électrique.

$$d_\mu^R = \frac{d_\mu}{d_\mu + d_\nu} \qquad d_\nu^R = \frac{d_\nu}{d_\mu + d_\nu} \tag{1}$$

$$d_\alpha^k = m_U^k.\sin(\frac{\pi}{3} - \theta_{out}^{*k}) \qquad d_\beta^k = m_U^k.\sin(\theta_{out}^{*k}) \tag{2}$$

$$\begin{aligned} d_\mu &= m_c.\sin(60° - \theta_{in}) \\ d_\nu &= m_c.\sin(\theta_{in}) \end{aligned} \qquad d_0^k = \frac{d_\mu.[1-(d_\mu + d_\nu).(d_\alpha^k + d_\beta^k)]}{d_\mu + d_\nu} \tag{3}$$

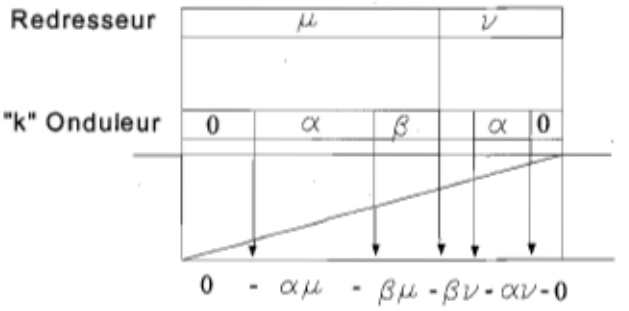


Figure 4 Synthèse de rapports cycliques pour l'état des interrupteurs dans l'étage onduleur "k"

$$d_1^k = d_\mu \cdot d_\alpha^k$$
$$d_2^k = (d_\mu + d_\nu) \cdot d_\beta^k \quad (4)$$
$$d_3^k = d_\nu \cdot d_\alpha^k$$

$$d_4^k = \frac{d_\nu \cdot [1 - (d_\mu + d_\nu) \cdot (d_\alpha^k + d_\beta^k)}{d_\mu + d_\nu} = 1 - d_0^k - d_1^k - d_2^k - d_3^k \quad (5)$$

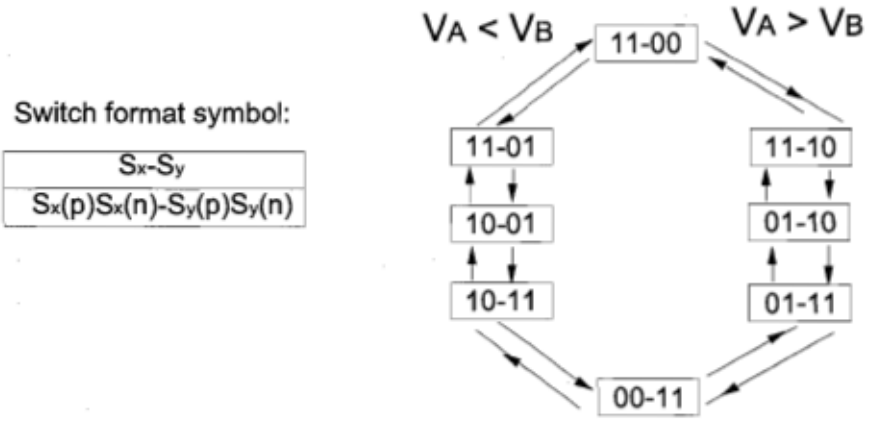


Figure 5 Quatre étages de commutation bidirectionnels contrôlés par le signe de la tension ligne à ligne d'entrée

2. Simulation de la solution avec Simulink™

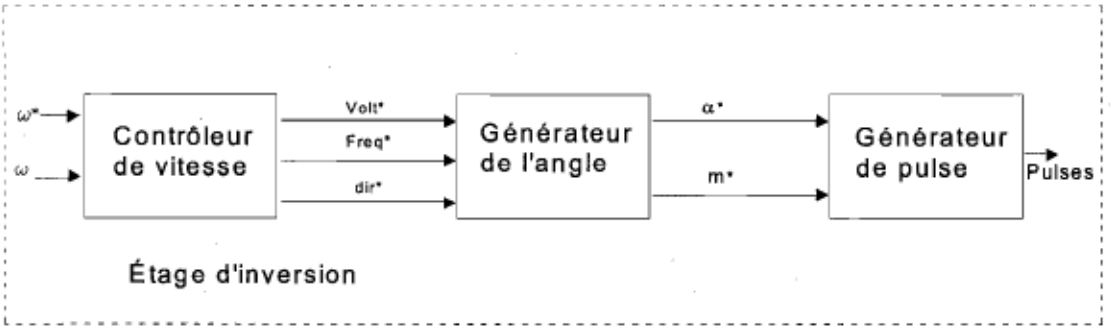
Les algorithmes de la boucle fermée de contrôle Volt/Hz constant avec le logiciel Simulink™ pour l'étage onduleur et l'étage redresseur sont montrés dans les figures 6, 7 et 8. L'état de IGBT, dans le cas ouvert et fermé, est défini par 1 ou 0 dans l'algorithme SateFlow qui produit des canaux de MLI pour attaquer les pilotes des IGBT.

Tous les IGBT de l'étage doivent rester dans le même état pendant la durée de temps définie par le rapport cyclique. Le secteur de l'hexagone, où le vecteur de la tension est placé, est très important pour définir les états des IGBT dans un cas de commutation. StateFlow est employé comme un outil dans Simulink pour définir ces états; StateFlow est facile à utiliser grâce à son interface graphique, figure 7.

La figure 8 montre l'algorithme de génération des canaux de MLI du redresseur dans l'environnement Simulink. Les tensions d'entrée au convertisseur de DPEC à deux étages sont analysées par un circuit à verrouillage de phase (*Phase Locked Loop*, PLL) pour déterminer l'angle du vecteur de la tension et la tension du bus cc.

Ensuite, on peut utiliser l'information de l'angle instantané, de concert avec les équations de SVM, afin de déterminer les rapports cycliques des canaux de MLI. En déterminant les temps de commutation et en comparant avec une onde triangulaire, on produit les impulsions pour attaquer les IGBT.

La même méthodologie est employée pour les onduleurs (étage d'inversion), sauf que les signaux de consigne de fréquence et de la tension appliquée au moteur sont variables, car le signal de consigne de vitesse est variable ou par l'effet du contrôleur de vitesse devient variable.



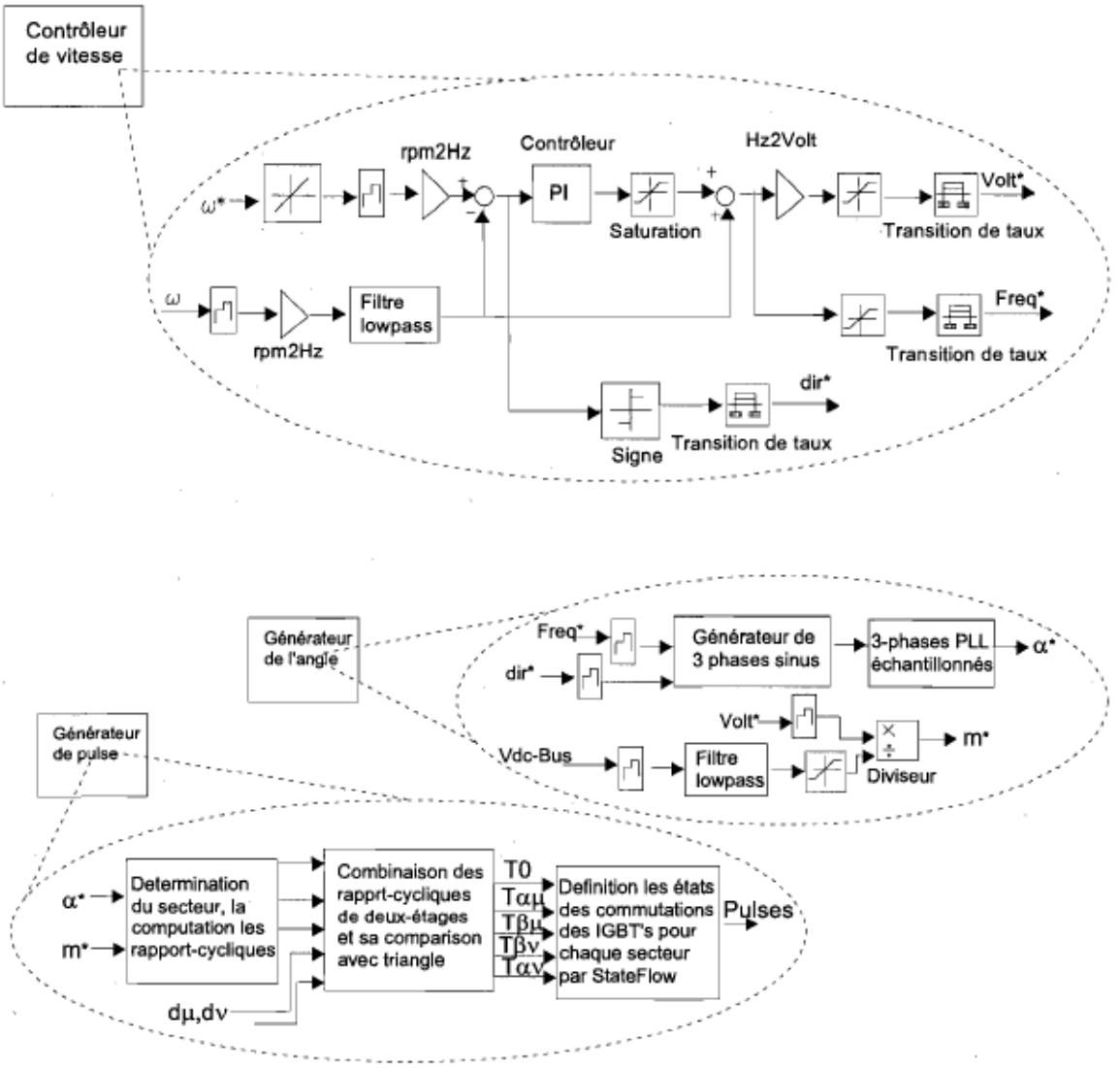


Figure 6 Modèle Simulink de la simulation pour l'algorithme de contrôle de l'étage d'inversion

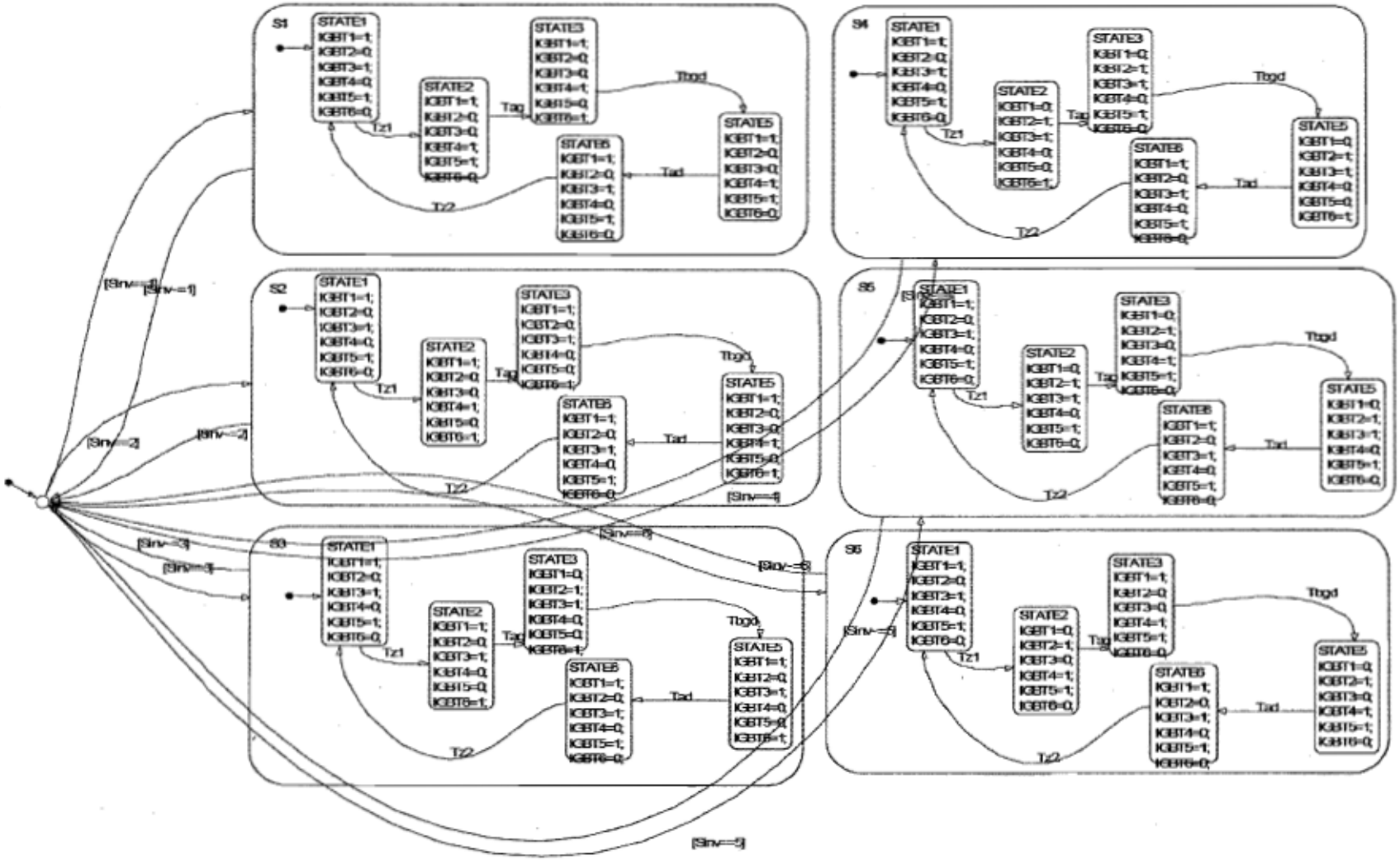


Figure 7 États des IGBT; le temps de transfert, les états et le secteur du vecteur de tension d'entrée sont définis pour l'étage d'inversion par StateFlow

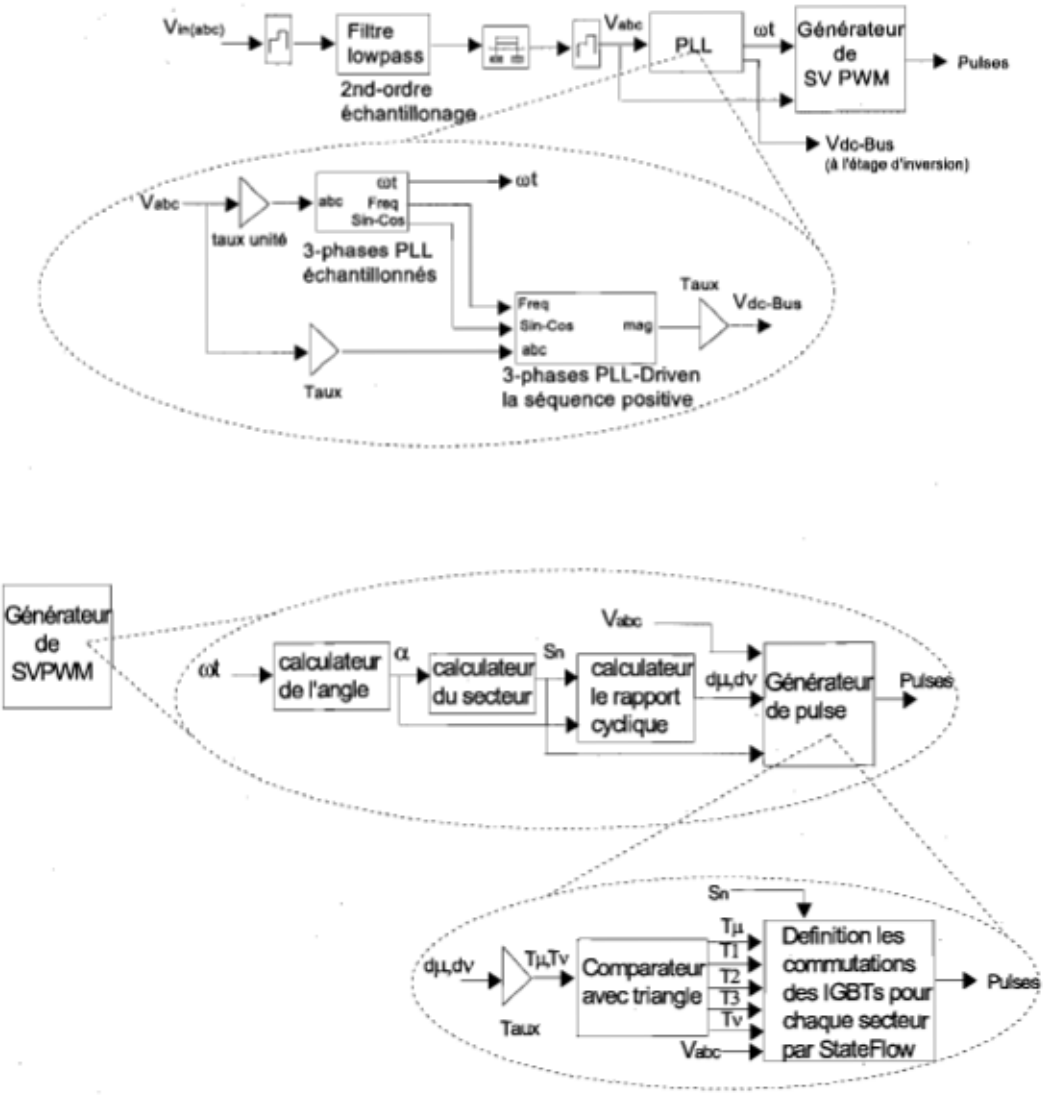


Figure 8 Modèle Simulink de la simulation pour l'algorithme de contrôle de l'étage de redressement

3. Simulation de la solution avec RT-LAB™

La simulation avec RT-LAB est très similaire à la simulation avec Simulink, avec les mêmes algorithmes pour les deux étages. La seule différence est le remplacement des blocs pertinents de ARTEMIS™ (*Advanced Real-Time Electro Mechanical Simulator*) et de RT-Events (*Real Time compensation of switching Events*) avec des blocs de SimPowerSystem (SPS).

Il faut construire 3 sous-systèmes contenant l'algorithme du contrôle dans le sous-système esclave (*Slave Subsystem*, SS), le système du puissance incluant le générateur, la ligne, le transformateur, le convertisseur et les moteurs dans le sous-système maître (*Master Subsystem*, SM) et le module de la visualisation des paramètres et d'ordre aux moteurs par des consignes des vitesses dans le sous-système de la console (*Console Subsystem*, SC), figure 9.

La simulation du module, dans l'environnement RT-LAB, est faite par une simulation à pas fixe, avec l'algorithme d'intégration ode4 (Runge-Kutta), avec 2 CPU Intel Xeon 501MHz, Opal-RT I/O Hardware OP5110-1, SW FirmWare: S17-0016-PCI-20-4. La configuration des 3 moteurs à induction (3HP, 220V, 60Hz, 1725rpm) avec couples statiques constants a été simulée. Le bloc compatible avec SPS 3-*leg time-stamped bridge* est utilisé comme onduleur et le bloc du redresseur est constitué par 3-*level IGBT time-stamped bridge*, tous deux provenant de la bibliothèque de RT-LAB.

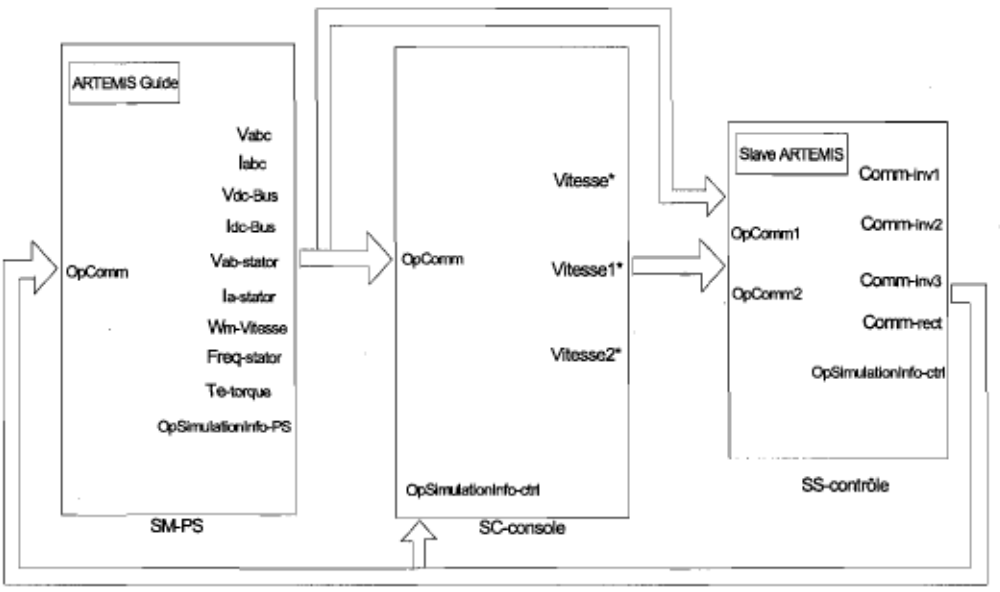


Figure 9 Modèle RT-LAB™ de la simulation pour le convertisseur avec l'algorithme de contrôle

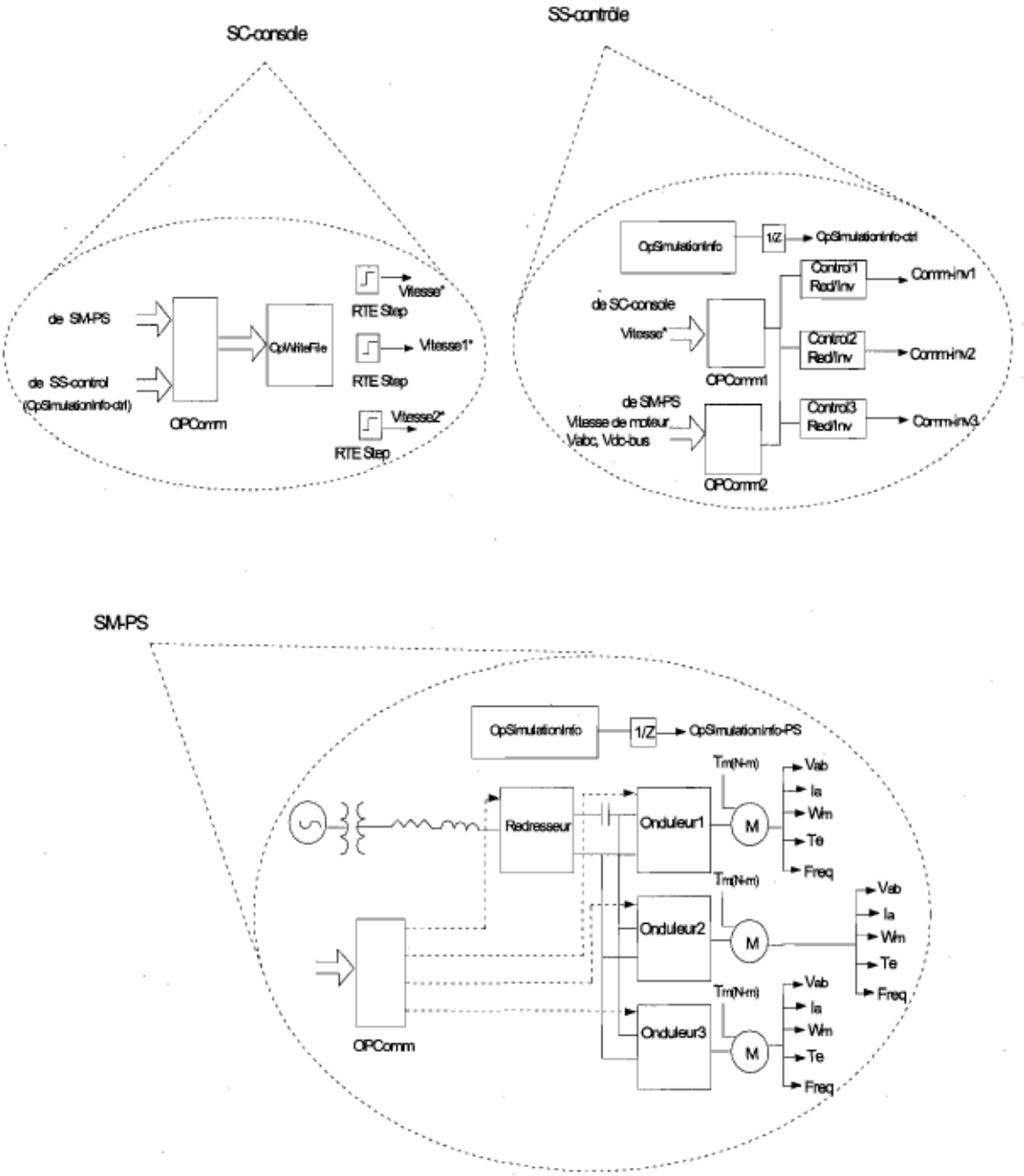


Figure 9 (suite)

Les figures 10 et 11 montrent le résultat de la simulation par RT-LAB pour le courant de ligne et la tension de sortie à la charge. Il faut noter que la simulation est faite sans le

filtre d'entrée avant le convertisseur, ce qui permet de considérer les harmoniques contenues dans le courant de ligne mieux qu'avec le filtre d'entrée.

Dans cette simulation, la fréquence de la porteuse (Fc) (onde triangulaire) et la période d'échantillonnage (Ts) sont supposées 3.6 kHz et 5e-6 s, respectivement. Pour plus d'information sur les autres résultats de simulation comme la vitesse du moteur, le couple, la fréquence et le courant du bus cc, veuillez consulter l'annexe A. Le fichier de compilation RT-LAB concernant la simulation en temps réel est consigné à l'annexe B.

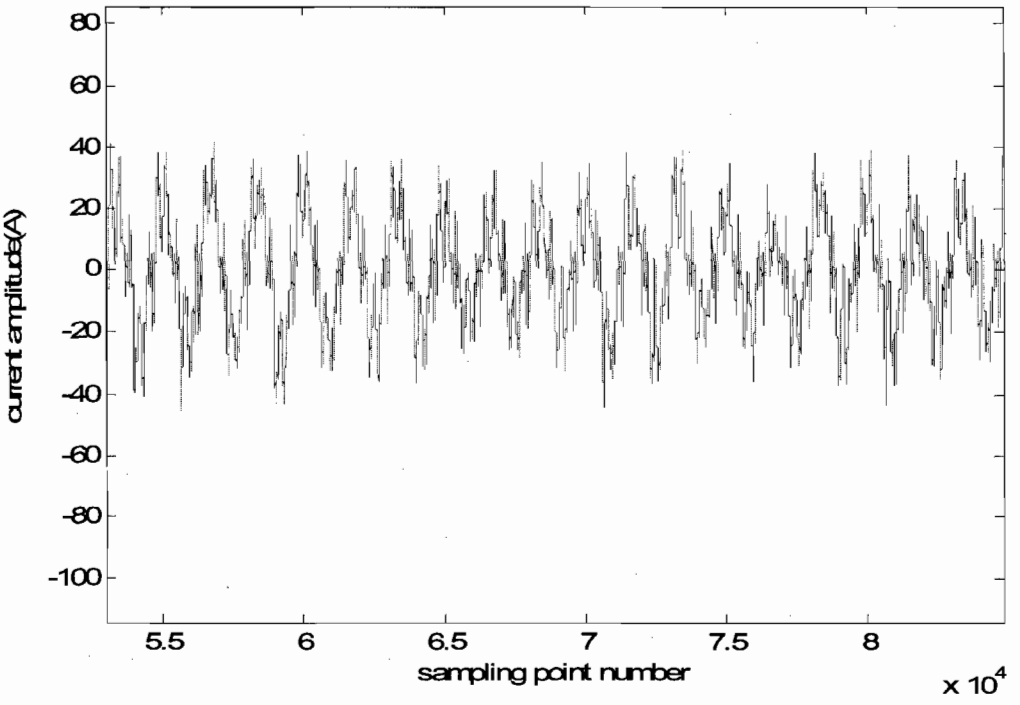


Figure 10 Courant de la ligne de DPEC simulé par RT-LAB sans le filtre d'entrée, Fc=3.6 kHz, Ts=5e-6 s

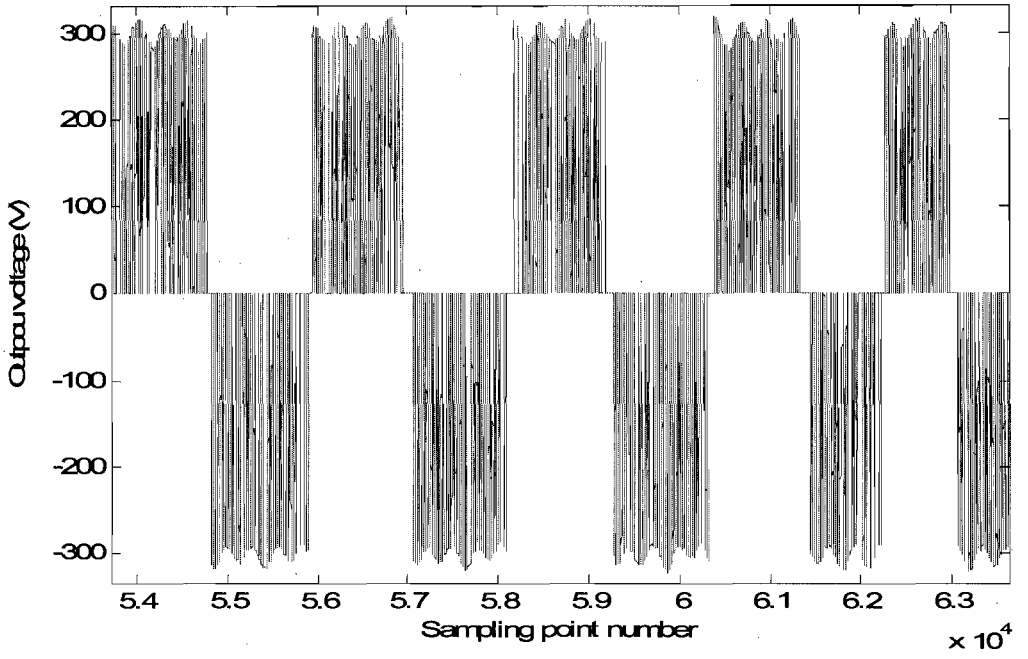


Figure 11 Tension de sortie du DPEC simulée par RT-LAB, Fc=3.6 kHz, Ts=5e-6 s

4. Simulation de la solution avec Xilinx System Generator (XSG™) et Simulink™

Dans cette simulation, la partie puissance est réalisée dans l'environnement Simulink et la partie de contrôle est construite par des blocs de XSG. On utilise StateCAD, l'équivalent de SateFlow, dans la bibliothèque de XSG pour définir les états des IGBT sous la forme d'une machine d'état. Alors, il permet de générer des codes HDL et de valider le travail de machine d'état, signal par signal, correspondant à chaque temps de transfert. Il faut importer les codes HDL générés dans l'environnement XSG par le bloc boite noire (*Black Box*). Pour arriver à ce but, on écrit un programme d'interface pour la modulation de XSG et de Simulink sous la forme d'un fichier script (fichier-m) dans MATLAB (Annexe C). Donc, la machine d'état peut être exécutée dans XSG précisément. Dans le reste de l'algorithme, des blocs de la bibliothèque de XSG sont

utilisés. Notons qu'il faut faire beaucoup de tests, étape par étape, dans XSG car la sortie de chaque bloc est en type binaire (point fixe).

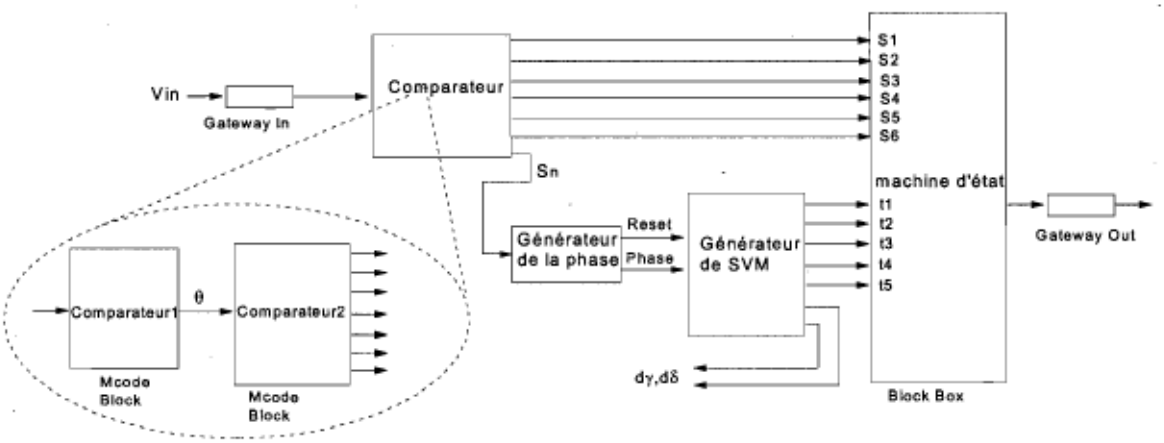


Figure 12 Modèle XSG de la simulation pour l'algorithme de l'étage de redressement

Les figures 12 et 13 montrent le modèle XSG de la simulation utilisé pour l'algorithme de contrôle du DPEC à deux étages. Il peut générer des canaux MLI pour attaquer aux IGBT tel qu'ils sont définis dans la machine d'état.

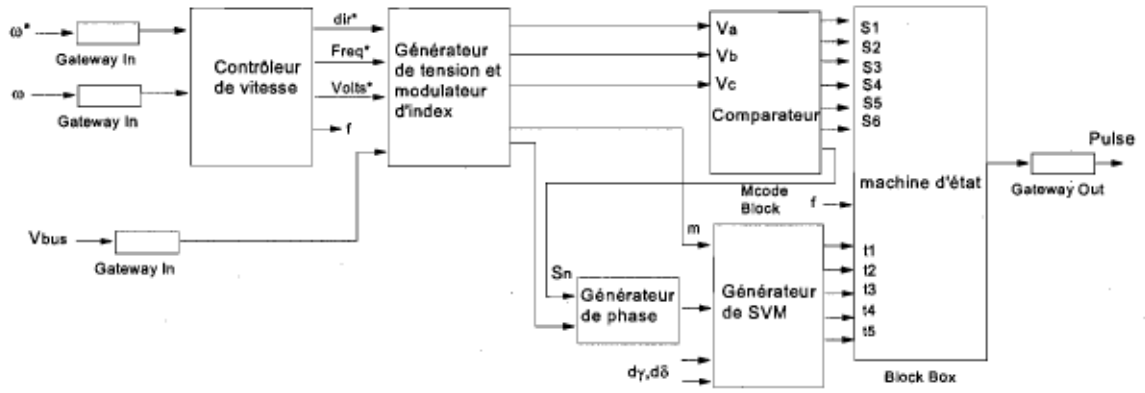


Figure 13 Modèle XSG de la simulation pour l'algorithme de l'étage d'inversion

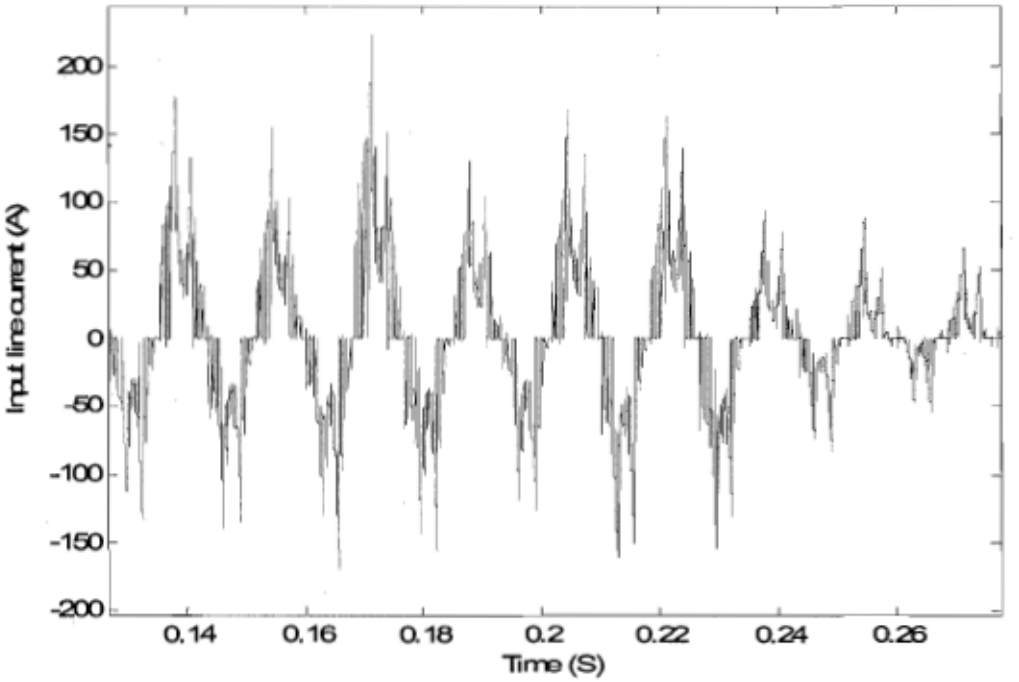


Figure 14 Courant de la ligne du DPEC simulé par XSG sans le filtre d'entrée, Fc=3.6 kHz,Ts=5e-6 s

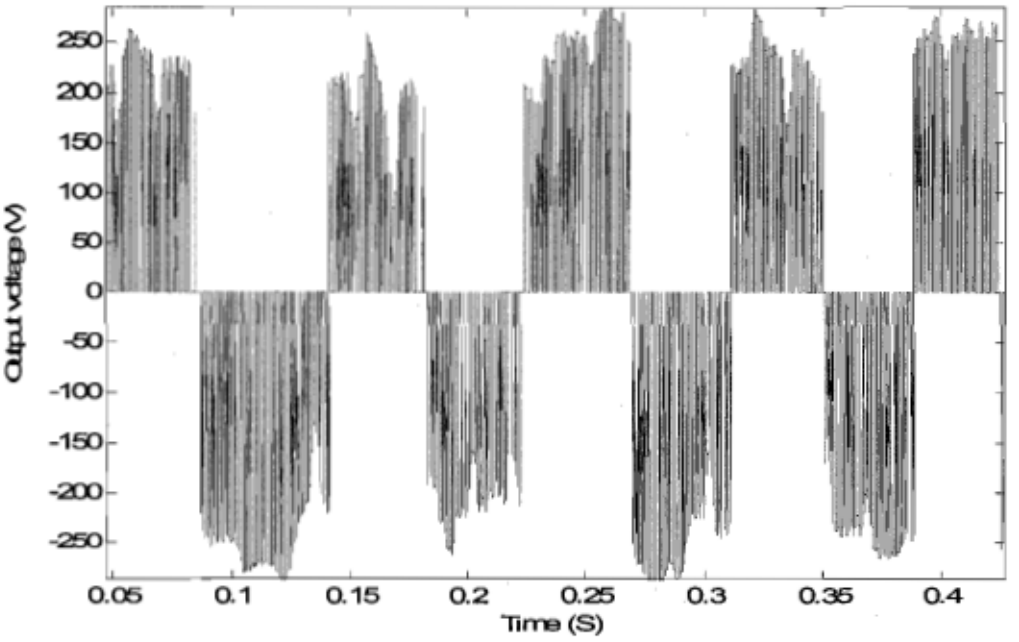


Figure 15 Tension de sortie du DPEC simulée par XSG, Fc=3.6 kHz, Ts=5e-6 s

5. Résultats des contenus harmoniques du courant d'entrée du DPEC simulé par XSG et RT-LAB

Les figures 16 et 17 indiquent que les harmoniques contenus dans le courant d'entrée, ont des amplitudes basses et qu'en plus du fondamental à 60 Hz, seulement les 5^e et 7^e harmoniques ont des amplitudes notables. Donc, ils confirment que les courants d'entrée obtenus par les deux simulations sont semi-sinusoïdaux. Les différences dans les amplitudes des harmoniques sur ces deux figures sont dues à la simulation à des points d'opération différents : les consignes de vitesse sont différentes. Notons qu'il n'y a pas de raison spéciale pour cette différence; les simulations ont simplement été réalisées avec des consignes différentes. Pour la figure FFT de la tension de sortie, veuillez vous référer au texte principal du mémoire (version anglaise).

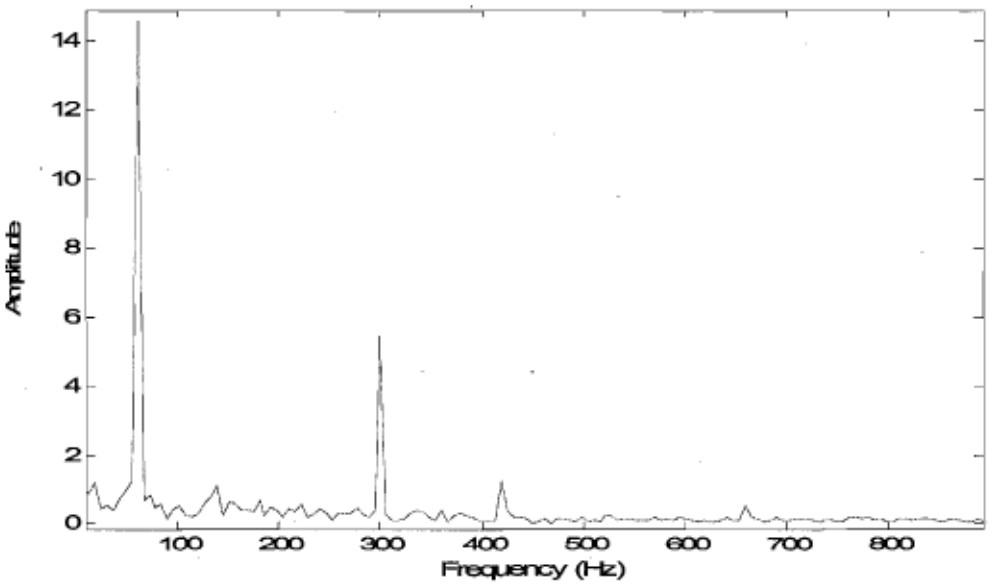


Figure 16 FFT du courant d'entrée du DPEC simulé par RT-LAB, Fc=3.6 kHz, Ts=5e-6 s

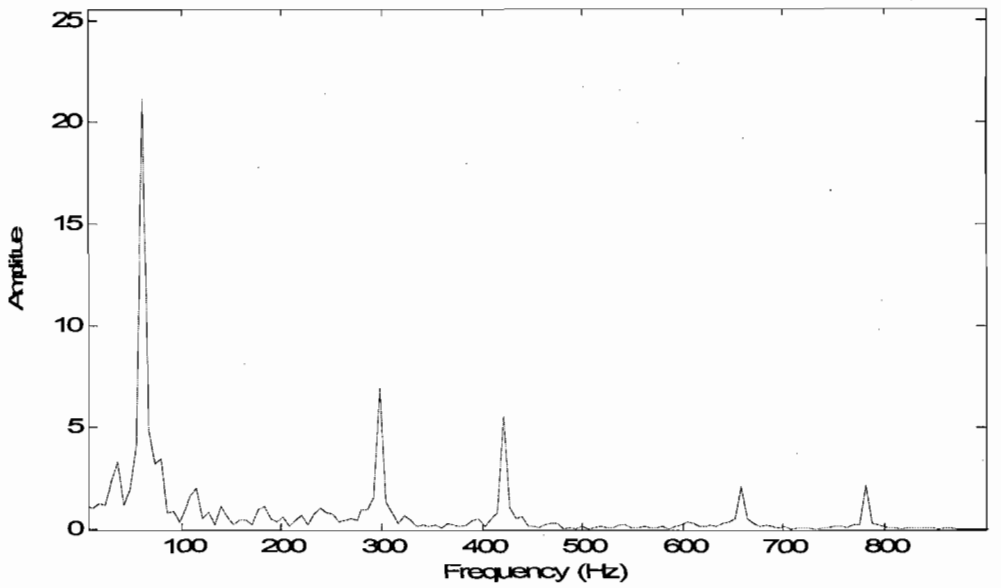


Figure 17 FFT du courant d'entrée du DPEC simulé par RT-LAB, Fc=3.6 kHz, Ts=5e-6 s

6. Évaluation des codes HDL pour un FPGA choisi par le logiciel Leonardo Spectrum L3

Le logiciel Leonardo Spectrum offre la possibilité de faire un choix optimal de FPGA pour les codes HDL. Les codes HDL de l'étage d'inversion ont été analysés, ce qui a mené au dispositif 2V250fg256 FPGA qui a été choisi par logiciel comme un matériel optimal pour ces codes en vertu de l'utilisation de la capacité maximum des ressources. Par exemple, le tableau I montre qu'il emploie 93.13% des générateurs de fonction. Le tableau II est le résultat de synthèse pour le dispositif 2v80fg256 FPGA. Pour plus d'informations, veuillez vous référer aux annexes D et E.

Il faut mentionner que les fréquences des dispositifs (18.1 MHz, 69.6 MHz) sont plus grandes que les fréquences d'échantillonnage et de la porteuse (1 MHZ, 3.6 KHz), ce qui valide le résultat de synthèse par Leonardo Spectrum L3.

Tableau I Synthèse des codes HDL de l'étage d'inversion pour le dispositif 2v250fg256 par Leonardo

```
*********************************************
      Device Utilization for 2V250fg256
*********************************************
Resource                Used    Avail    Utilization
---------------------------------------------
IOs                     70      172      40.70%
Global Buffers          1       16        6.25%
Function Generators     2861    3072     93.13%
CLB Slices              1431    1536     93.16%
Dffs or Latches         1519    3588     42.34%
Block RAMs              0       24        0.00%
Block Multipliers       9       24       37.50%
Block Multiplier Dffs   252     864      29.17%
---------------------------------------------

          Clock Frequency Report
  Clock                : Frequency
-----------------------------------
  clk_1                : 18.1 MHz
```

Tableau II Synthèse des codes HDL de l'étage de redressement pour le dispositif 2v80fg256 par Leonardo Spectrum L3

```
*********************************************
      Device Utilization for 2V80fg256
*********************************************
Resource                Used    Avail    Utilization
---------------------------------------------
IOs                     104     120      86.67%
Global Buffers          1       16        6.25%
Function Generators     657     1024     64.16%
CLB Slices              329     512      64.26%
Dffs or Latches         110     1384      7.95%
Block RAMs              0       8         0.00%
Block Multipliers       0       8         0.00%
Block Multiplier Dffs   0       288       0.00%
---------------------------------------------

          Clock Frequency Report
  Clock                : Frequency
-----------------------------------
  clk_1                : 69.6 MHz
```

7. Conclusion

Nous avons confirmé l'intérêt d'une méthode de modulation qui permet de réduire les distorsions du courant d'entrée dans un système EVV avec 3 moteurs à courant alternatif avec alimentation triphasée et bus cc commun. Le convertisseur DPEC avec modulation

vectorielle SVM pour générer des canaux MLI est une solution efficace pour ce travail. Cette loi de modulation est validée avec des travaux de simulation par Simulink, RT-LAB et XSG. La génération des codes HDL est réalisée par XSG et le potentiel d'implémentation des codes dans un FPGA est validé par la synthèse. C'est un travail très intéressant qui a permis de pratiquer la simulation d'un système multimoteur et multiconvertisseur, incluant la génération des signaux de modulation, le contrôle des moteurs avec différentes consignes de vitesse, de réaliser la commande de plusieurs variables des moteurs comme la vitesse, le courant, la tension, le couple, etc. Nous avons aussi mesuré la FFT de courant d'entrée et comparé le résultat FFT de XSG avec le résultat FFT de [5] (dans le corps principal du mémoire). Ceci a permis de mettre en évidence le potentiel de l'algorithme de commande par le résultat avec XSG (éventuellement le FPGA), la génération des codes VHDL et de valider l'implémentation de la loi de modulation (des codes) dans un FPGA par la synthèse Leonardo Spectrum L3.

Chapter 1- Introduction

The evolution of Adjustable Speed Drives (ASDs) has started from heavy and large size electromechanical assemblies to more compact and efficient power electronics equipments, so that presently the trend is to integrate the power electronics equipment inside the electrical motor in order to produce an efficient and flexible single-unit electromechanical converter. This allows for fast industry process design and short commissioning time.

The necessity to reduce energy consumption in industrial processes, allowing for important energy savings, reducing harmonic pollution in the grid and a shorter payback time of the drive reaction are the matters which engineers are involved in to find an effective solution.

The demands of an industrial drive are analyzed with respect to the mechanical characteristics, reliability, ride-through capability, inrush problems and other standards [1]. Other standards have to be fulfilled by the drive in a final industrial product for the immunity and Electro Magnetic Compatibility (EMC): low and high frequency on the line supply (voltage harmonic distortion, commutation notches, voltage interruption, voltage unbalance, frequency variations, surge, burst, Radio Frequency Interference (RFI) and harmonic current).

A short-term ride-through strategy is tested in [2] and can be implemented on standard drives. Its main control tasks are to decelerate the drive and to recover the necessary energy from the load inertia, during loss of the power grid, in order to feed the control electronics and when the grid is re-established, to accelerate the drive to the reference speed from non-zero shaft speed [3]-[4].

The power-up problems are expected to appear in the drive, due to the L-C series topology of the input filter. Therefore, a power-up circuit can be implemented on the converter prototype to damp the oscillations and to limit the over voltage level during power-up.

One of the most important problems in the case of using the ASDs is the harmonic distortion in the current of the grid caused by interaction of the ASD and the grid. This interaction may produce some disturbances in the electrical grid such as noises, voltage unbalance and frequency variation which cause distortion of the electrical insulation in the devices used by costumer.

High harmonic frequency contents in the input line currents of each ASD accumulate in the grid as harmonic pollution. The objectives of this research are to find an effective solution to reduce the grid current Total Harmonic Distortion (THD) level, as the input line current to the ASDs, in a multi-drive system with induction motors and to implement this methodology in the Field-Programmable Gate Array (FPGA) as a processor to send Pulse Width Modulation (PWM) patterns to ASD gates.

Because of high speed in processing and calculation of complex mathematical equations conducted by FPGA, in comparison with other technologies such as Peripheral Interface Controller (PIC), utilisation of a FPGA in this solution is the major tentative part of this research. So we should find a proper methodology for synchronising PWM pulses including the specific commutation and sequence strategies to command the ASDs with utilisation of a proper topology to produce a semi-sinusoidal input line current wave. The major benefit of this research goal, in addition to controlling the motors properly in a multi-drive system, is to increase the quality of the grid currents towards the ASD and

consequently to reduce more the harmonic pollution in the grid with respect to other available technologies by grace of using FPGAs.

The two-stage Direct Power Electronic Conversion (DPEC) topology [5] is considered to independently control each motor in a multi-drive system: it is a converter with a reduced number of components with a structure consisting of a common rectification stage with a common DC bus connected to several inverter stages to drive the motors. In fact this type of converter is an extended type of Matrix Converter (MC) [6] which has additional structural advantages such as fewer components of Insulated Gate Bipolar Transistor (IGBT) and diodes.

To control the DPEC topology, Space Vector Modulation (SVM) is chosen as the basic methodology with a combination strategy [7] of SVM in each inversion stage with common rectification stage, for the goal of approaching semi-sinusoidal input current and output voltage waves of the converter, chapter 2.

Utilisation of these topology and methodology, chapter 3, have more advantages than other previous solutions such as voltage transfer ratio from input to output near to unity, they are more compact, they are cheap to manufacture and result in less loss of energy, less converter sensibility to the grid variations and instability. In section 3.6, an analysis of SVM hexagon based on three phase input voltages is presented and the formulas of this chapter are expanded. Chapter 2 and 3 are a part of my research study and my contributions include some formulas' development, description and some analysis (i.e. section 3.6).

Simulation of the modulation in real time workshop with implementation of the combined SVM methodology for the two-stage DPEC in the FPGA are considered to

respond the research objective, so that a new method of Hardware Description Language (HDL) code generation with implementation eligibility into the FPGA specially used for command of the power converter is presented as the development part of this research.

Xilinx System Generator™ (XSG™) and RT-LAB™ softwares with the real time workshop used for simulation are described in chapter 4 and FFT of the input line current and output voltage waves are calculated and compared with the result of FFT from other article.

As the XSG™ for Digital Signal Processor (DSP) simulates and implements high-performance DSP designs on the Xilinx's FPGAs, FPGA code generation with XSG™ and codes synthesis by use of Leonardo Spectrum Level-3 software to verify its liability for implementation in an optimal FPGA are described in chapter 5.

Chapter 6 presents conclusion part of this research and all programming in the HDL, Matlab® and generated HDL codes and its synthesis are demonstrated in the appendices.

Chapter 2- The concept of two-stage DPEC (Direct Power Electronic Conversion) and Space Vector Modulation (SVM) strategy

In multi-drive applications, in order to decrease the specific cost, a new concept has been developed and is already industrially implemented [8]. A single rectifier is used to feed several inverters. If a PWM rectifier that is able to provide a better interaction with the grid is used, the cost will be less than equipping each converter unit with its own PWM rectifier. As the inverters may be placed in different cabinets, decoupling of the dc-bus is necessary as well as taking extra precaution against dc-link short circuits, which may cause massive destruction due to the high amount of stored energy. Among the solutions, a dc/ac PWM step-down VSI in the inversion stage seems to offer the best performance on the motor side. The Space Vector Modulation (SVM) strategy is based on space vector representation of the converter ac side voltage and has become very popular because of its relative simplicity. It is a better solution in comparison with Alesina-Venturini strategy to control the Matrix Converter (MC) [9]. A brief description of IM equations is also provided since it will be part of the simulation module.

2.1 Introduction to two-stage DPEC topology and to Matrix Converters (MC)

Regarding of the rectification stage and dc-link, two options exist: a PWM voltage source rectifier with a large dc-capacitor [10] or a direct link of a PWM current source rectification stage to an inversion stage, shown in Fig.2.1. This consists of a standard B6

PWM-VSI stage and a rectification stage, which may be a 3/2-phase matrix converter presented in Fig.2.2. It is possible to obtain a cost-effective topology, requiring only 9 IGBTs and 18 Fast Recovery Diodes (FRD). The topologies of two 3/1-phase power modules for the rectification stage are based on the anti-paralleled connection of two unidirectional switches, Common Collector (CC) and Common Emitter (CE) [11].

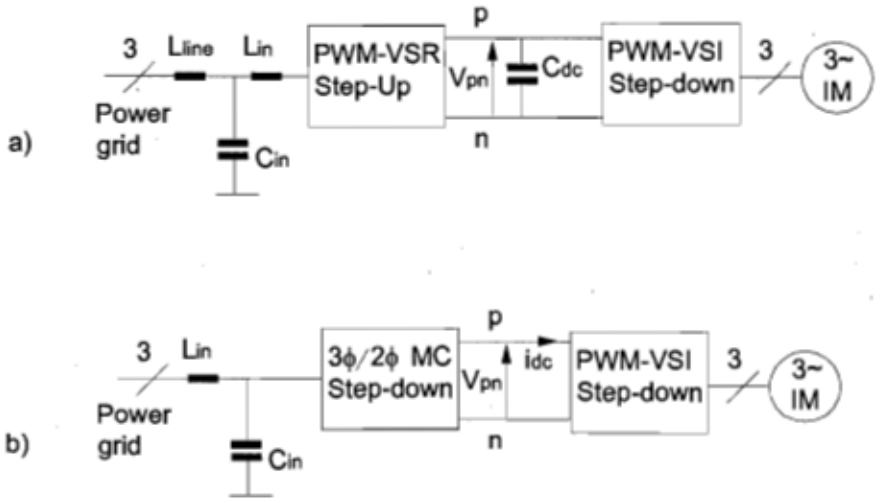


Figure 2.1 Adjustable Speed Drive (ASD) topologies a) Classical front-end with a stiff dc-link voltage using a large dc capacitor b) Based on two-stage DPEC concept

The task of the rectification stage is to switch between two line-to-line voltages in order to provide sinusoidal input currents and to keep the connection of the dc-link to a given line-to-line voltage at any instant. Zero-vector is produced inherently by the inversion stage because a zero output voltage vector is equivalent to connecting all outputs to the same dc-bus polarity and therefore disconnecting the link between the rectification stage and the load. Only one line-to-line voltage is selected at a time and applied with the right polarity to the inversion stage. For DPEC prototype equipped with a PWM-VSR in rectification stage and a small dc-link capacitance, the mathematical model of capacitance in dc-link is investigated in [12].

The matrix converter, as an all-silicon topology, has advantages like lack of the bulky dc-link capacitors, the bidirectional power flow, unity power factor and the sinusoidal input current/output voltage waveforms with only high order harmonics [13]. Because of the high integration capability and the higher reliability of the semiconductor, the matrix converter topology is recommended for extreme temperatures and critical volume/weight applications.

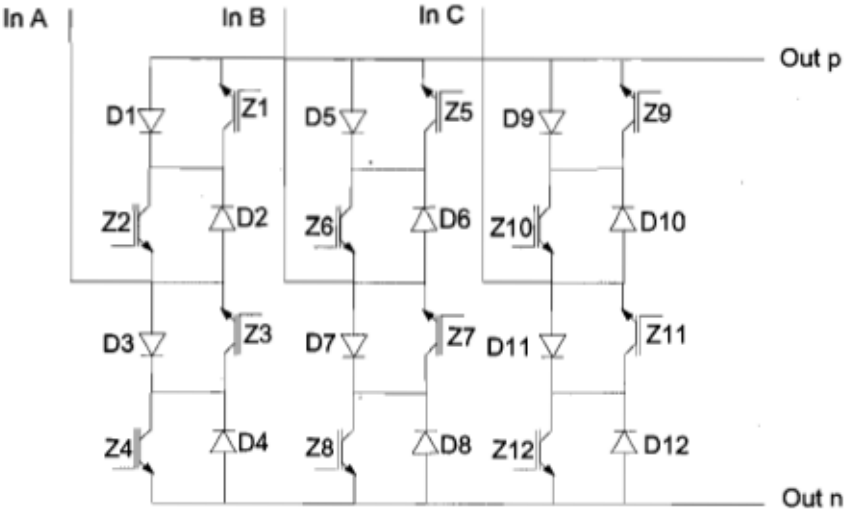


Figure 2.2 12-IGBTs scheme of the rectification stage (3/2 matrix converter) for two-stage DPEC

2.2 Using Reverse Blocking-IGBT (RB-IGBT) in two-stage DPEC

Reverse Blocking-IGBTs, a bidirectional switch topology, is one of the important advances achieved in this technology so that, because of its soft commutation advantage [14], it is proposed to be used in the DPEC rectification stage. It has been shown that higher efficiency may be reached at higher switching frequencies, more than 10 KHz. The reverse blocking IGBT (RB-IGBT) decreases the number of semiconductor devices per load phase path. The advantage offered by this solution is to increase the conduction

efficiency of the DPEC because the conduction losses will be produced only by a single RB-IGBT per phase [15].

2.3 Input filter implementation aspect and calculation

Input filter has to reduce the input current ripple with minimum installed energy on the reactive elements. The most used topology is an LC series circuit. The design of the input filter [1] has to accomplish the following criteria:

1. Input filter with a cut-off frequency lower than the switching frequency

$$L_{in} \cdot C_{in} = 1/\omega_0^2 \tag{2.1}$$

where L_{in}, C_{in} are the inductance and the capacitor of the input filter and ω_0 is the resonance pulsation of input filter.

2. Maximize the displacement angle ($\varphi_{\min-in}$) for a given minimum output power

$$\frac{C_{in}}{P_n} = k_{\min} \cdot \tan \varphi_{\min-in} \frac{1}{3.\omega_n \cdot U_n^2} \tag{2.2}$$

where $P_n \cong 3.U_n \cdot I_n$ is the input active power (considering a close to unity power factor at full load); U_n, I_n are the rated input phase voltage and current of the converter; $P_{\min} = k_{\min} \cdot P_n$ is the minimum power level where the displacement angle $\varphi_{\min-in}$ reaches its limit ; $k_{\min}$ is the ratio of minimum power level and nominal power level; $\omega_n = 2.\pi.f_n$ is the pulsation of the power grid.

3. Minimize the input filter volume or weight for a given reactive power, by taking into account the energy densities which are different for film capacitors and for iron chokes:

$$\frac{S_L}{S_C} = \frac{1}{\left[3.\omega_0 \cdot U_n^2\right]^2} \cdot \left(\frac{p_n}{C_{in}}\right)^2 \tag{2.3}$$

where $S_L = \omega_n \cdot L_{in} \cdot I_n^2$ (choke), $S_C = \omega_n \cdot C_{in} \cdot U_n^2$ (capacitor) are the installed VA in reactive components and $\omega_0 = 2.\pi . f_0 = \left(L_{in} \cdot C_{in} \right)^{-1/2}$.

4. Minimize the voltage drop on the filter inductance at the rated current in order to provide the highest voltage transfer ratio:

$$\frac{\Delta U_n}{U} = 1 - \sqrt{1 - (\omega_n \cdot L_{in})^2 \cdot \left(\frac{I_n}{U_n} \right)^2} = 1 - \sqrt{1 - l_{in}^2} \tag{2.4}$$

where ΔU_n is the drop in voltage magnitude due to the influence of the input filter and l_{in} is the filter inductance in p.u.

2.4 Comparison of two-stage DPEC with MC

The matrix converter has not become commercial so far because the voltage transfer ratio is limited to 0.86, the immunity to power grid disturbance is low, it needs a higher number of power semiconductor devices, the control is more complicated and extra circuits such as an LC input filter and a clamp circuit are needed to provide EMC compliance and safe operation.

Therefore, the overall cost remains high. The two-stage DPEC has an input port for a three-phase power supply and several output ports to connect three-phase loads, which are independently controlled. The voltage transfer ratio is close to unity, it is less sensitive to power grid disturbance, it has fewer power semiconductor and diode devices; henceforth it has less power loss, it is more compact and it is cheaper than the matrix converter. Different types of DPEC structures such as Ultera Spark Matrix Converter (USMC) and Very Spark Matrix Converter (VSMC) are implemented with fewer components [16].

2.5 Basics of SVM technique

In SVM technique [17], a three-phase, two-level converter provides eight possible switching states, made up of six active and two zero switching states. Active vectors divide the plane for six sectors, where a reference vector U^* is obtained by switching (for the proper time) on two adjacent vectors (Fig. 2.3). Reference vector U^* is sampled with fixed clock frequency $2f_s = 1/T_s$ and the next $U^*(T_s)$ is related to times t_1, t_2, t_0 and t_7 where M and α are the modulation index and vector angle. The residual sampling time is reserved for zero vectors U_0 and U_7 with the condition that $t_1 + t_2 \le T_s$.

$$t_1 = \frac{2\sqrt{3}}{\pi} \mathrm{M}\, T_s \sin(\pi/3 - \alpha) \tag{2.5}$$

$$t_2 = \frac{2\sqrt{3}}{\pi} \mathrm{M} T_s \sin\alpha \tag{2.6}$$

$$t_{0,7} = T_s - t_1 - t_2 = t_0 + t_7 \tag{2.7}$$

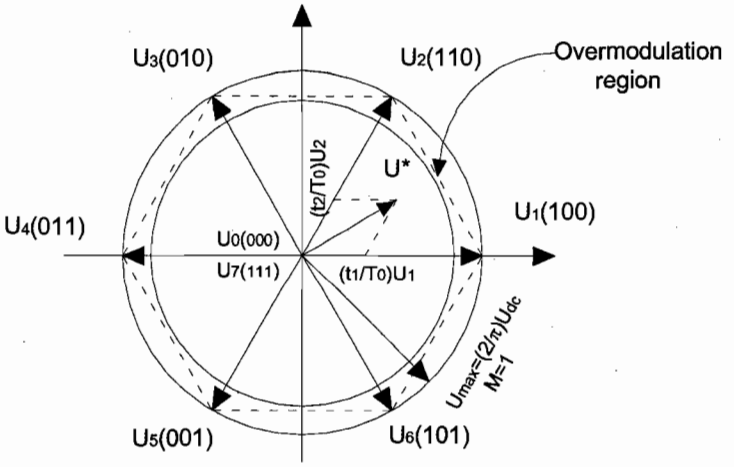


Figure 2.3 Space vector representation of three-phase converter

2.5.1 Three-phase SVM with symmetrical placement of zero vectors

The most popular SVM method is modulation with symmetrical zero states (SVPWM):

$$t_0 = t_7 = (T_s - t_1 - t_2)/2 \tag{2.8}$$

Figure 2.4a shows the information of gate pulses for (SVPWM) and the correlation between duty time T_{on}, T_{off} and the duration of vectors t_1, t_2, t_0, and t_7 [9]. For the first sector, commutation delay can be computed ($T_{0t} = t_0 + t_7, t_0 = t_7$):

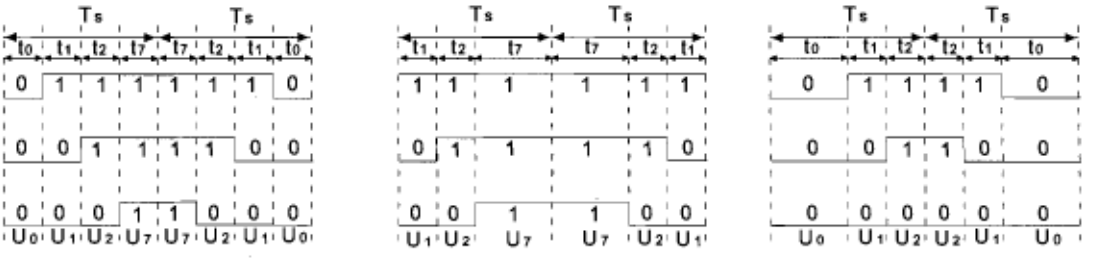
$$T_{aon} = \frac{T_{0t}}{2} \qquad T_{aoff} = \frac{T_{0t}}{2} + t_1 + t_2 \tag{2.9}$$

$$T_{bon} = \frac{T_{0t}}{2} + t_1 \qquad T_{boff} = \frac{T_{0t}}{2} + t_2 \tag{2.10}$$

$$T_{con} = \frac{T_{0t}}{2} + t_1 + t_2 \qquad T_{coff} = \frac{T_{0t}}{2} \tag{2.11}$$

2.5.2 Two-phase SVM

This type of modulation is a Discontinuous Pulse Width Modulation (DPWM) as CB technique with an additional Zero Sequence Signal (ZSS). The idea is based on the assumption that only two phases are switched (one phase is clamped to the lower or upper dc bus). It gives only one zero state per sampling time, Figure 2.4b.



(a) Conventional SVPWM **(b) DPWM**

Figure 2.4 Vector placement in sampling time.

2.6 SVM over modulation

In SVM-PWM, the allowable length of reference vector U^* which provides linear modulation is equal to $U^*_{max} = U_{dc}/\sqrt{3}$ (circle inscribed in hexagon M= 0.906). To obtain higher values of output voltage (over modulation) up to the maximal modulation index M= 1 (six step mode), an additional nonlinear over modulation algorithm has to be applied, Figure 2.5. This is because the minimal pulse width becomes shorter than critical pulse width (mainly dependent on power switch characteristics and usually in the range of a few microseconds) or even negative. Zero vectors are never used in over modulation.

2.7 Performance criteria

Several performance criteria are considered for selection of suitable modulation method. The power quality criteria and distortion factor under unbalanced voltage supply is investigated in [18]. In this subsection, further important criteria such as range of linear operation, distortion factor and switching losses are discussed [19].

2.7.1 Range of linear operation

The range of the linear part of the control characteristic for sinusoidal Carrier Based PWM (CB-PWM) ends at $M=\pi/4=0.785$ of the modulation index, when reference and carrier amplitudes become equal. The SVM or CB-PWM with ZSS injection provides extension of the linear range up to $M_{max}=\pi/2\sqrt{3}=0.906$ and the region above that is the nonlinear over modulation range.

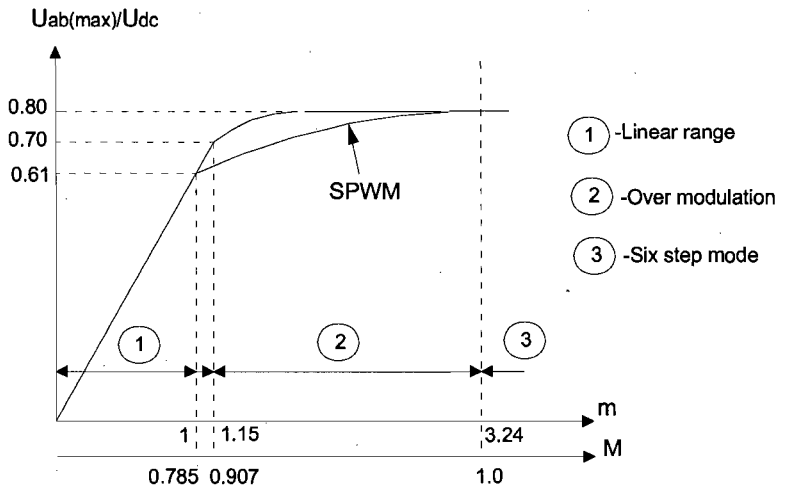


Figure 2.5 Control characteristics of PWM converter

2.7.2 Switching losses

Power losses of the PWM converter can be generally divided into *conduction* and *switching losses*. Conduction losses are practically the same for different PWM techniques and they are much lower than switching losses. For the switching loss calculation, linear dependency of the switching energy loss on the switched current is assumed. This also was proved by the measurement results. Therefore, for high switching frequency, the total average value of the transistor switching power losses, $P_{sl(c)}$, for continuous PWM, can be expressed as:

$$P_{sl}(c) = \frac{1}{2\pi}\int_{(-\pi/2)+\varphi}^{(\pi/2)+\varphi} k_{TD} \cdot i \cdot f_s \; d\alpha = \frac{k_{TD} I f_s}{\pi} \tag{2.12}$$

Where $k_{TD} = k_T + k_D$ is the proportional relation of the switching energy loss per pulse period to the switched current for the transistor and the diode; f_s is the switching frequency and i is the current which passes through the transistor.

In the case of discontinuous PWM, the following properties hold from the symmetry of the pole voltage:

$$P_{sl}(-\varphi)=P_{sl}(\varphi)\ ,\ P_{sl}(\varphi)=P_{sl}(\pi-\varphi) \qquad 0<\varphi<\pi \tag{2.13}$$

where φ is voltage vector angle. Therefore, it is sufficient to consider the range from 0 to $\frac{\pi}{2}$ for the DPWM, such that

$$P_{sl}(\varphi)=P_{sl}(c)\cdot k \qquad 0<k<1 \tag{2.14}$$

where k is a coefficient for continuous PWM switching losses, which are reduced by DPWM technique, on average about 33%. In favourable conditions, when modulation is clamped in phase conducting maximum current, switching losses decrease up to 50%.

2.7.3 Distortion and harmonic copper loss factor

The effective harmonic current depends on the type of PWM and also on the value of the ac side impedance. The *distortion factor* is defined as:

$$d=I_{h(rms)}/I_{h(six-step)(rms)} \tag{2.15}$$

It is unity for the six-step mode (continuous modulation). To reduce the influence of ac side impedance parameters on the distortion factor, current harmonics shall be reduced. It should be noted that harmonic copper losses in the ac-side load are proportional to d^2 *(loss factor).* Values of distortion factor can be computed for different modulation methods. It depends on the switching frequency, the modulation index M and the shape of ZSS. For continuous modulation with ZSS:

SPWM $$d=\frac{f_{spwm}(M)}{k\,f_{SB}},\ \mathrm{M}\in\left[0,\frac{\pi}{4}\right] \tag{2.16}$$

$$\text{SVPWM} \quad d=\frac{f_{svpwm}(\mathrm{M})}{k\, f_{SB}},\ \mathrm{M}\in\left[0,\frac{\pi}{2\sqrt{3}}\right] \tag{2.17}$$

This SPWM is Sinusoidal PWM: it as a modulation based on comparison of a common triangular carrier signal with three sinusoidal signals, which define the switching instants of the power transistor. For discontinuous modulation (DPWM):

$$\text{DPWM(n)} \quad d=\frac{f_{DPWM(n)}(\mathrm{M})}{k\, f_{SB}},\ \mathrm{M}\in\left[0,\frac{\pi}{2\sqrt{3}}\right],n=\{1,2,3\} \tag{2.18}$$

where k is defined as the ratio of carrier frequency (f_{SB}) and $f(\mathrm{M})$ is function of the modulation index with regards to the modulation method. The carrier frequency can be increased by a factor of 3/2 for a 66% copper loss reduction or doubled for a 75% copper loss reduction while the current distortion of DPWM is lower than continuous PWM at the same carrier frequency.

2.8 Steady-state equivalent circuit of induction motor

An equivalent circuit based on rotor flux linkage is presented in Fig. 2.6. If we adopt, the voltage equations for a cage motor take the form:

$$U_s = r_s i_s + j\,\omega_s \sigma x_s i_s + j\,\omega_s \left(\frac{x_M}{x_r}\right)^2 x_r i_{Mr} \tag{2.19}$$

$$0 = j\,\omega_s \left(\frac{x_M}{x_r}\right)^2 x_r i_{Mr} + \frac{\omega_s}{\omega_r} r_r \left(\frac{x_M}{x_r}\right)^2 i_r \frac{x_r}{x_M} \tag{2.20}$$

where $x_s - (x_M/x_r)x_M = \sigma x_s$ and x_M are the total leakage reactance and the magnetizing reactance. The stator and rotor winding reactance and resistance are x_s, x_r, r_s, r_r .The magnetizing current is expressed as:

$$i_{Mr} = \frac{\psi_r}{x_M} = i_s + \frac{x_r}{x_M} i_r \tag{2.21}$$

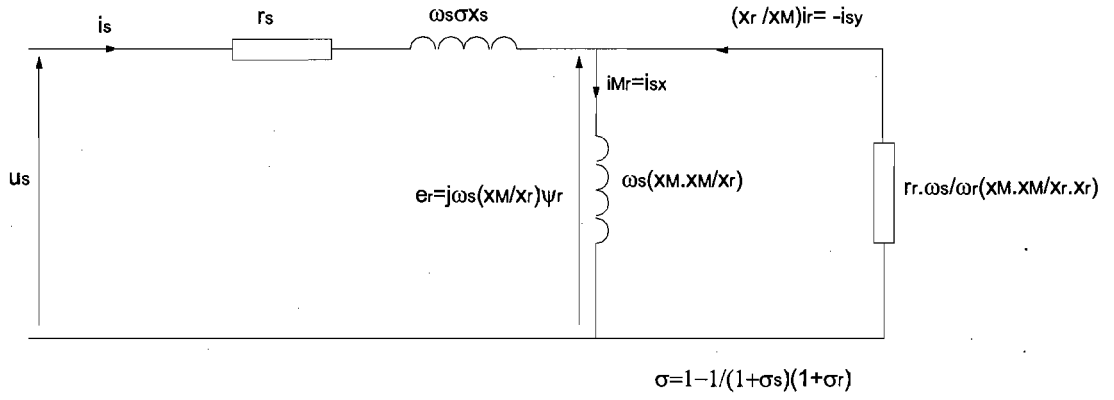


Figure 2.6 Steady-state equivalent circuit of induction motor based on the rotor flux linkage $a=x_M/x_r$

The total leakage reactance, σx_s , appearing in the circuit, is the sum of stator and rotor leakage reactance, which occurs in the case of the equivalent circuit:

$$\sigma x_s = \sigma_r x_M + \sigma_s x_M \tag{2.27}$$

where σx_s is often referred to as the transient reactance. IM block diagrams such as voltage controlled IM in stator based on the fixed coordinate system (α, β, 0) and current controlled IM in synchronous coordinates (x, y, 0), and interesting mathematical equations of the IM based on space vectors in a coordinate system are presented in [19].

2.9 Conclusion

In addition that the two-stage DPEC covers all the matrix converter functionalities, it provides more advantages than the MC such as the immunity due to the grid disturbance, voltage transfer ratio about unity and fewer components. This chapter has proposed input

filter calculation and using the RB-IGBT in rectification stage for bidirectional power transfer purpose, which is more compact, cheaper and better functionality than the normal CC or CE IGBT (Fig. 2.2). The switching losses are increased by the magnitude of the phase current (approximately linearly), then a suitable modulation strategy can significantly improve performance of the converter. For the same carrier frequency, discontinuous PWM possesses higher current harmonic content than continuous methods. The harmonic content is similar for both methods at high PWM index only. However, we should remember that discontinuous modulation possesses lower switching losses. Switching losses depend on the type of discontinuous modulation and power factor angle. Because of importance of minimum harmonic content in input line current, we would rather select a continuous modulation strategy like SVPWM than DPWM. Then, more investigation and research on two-stage DPEC have an intrinsic value for the development of the converter technology.

Chapter 3- Implementation of SVM technique into two-stage Direct Power Electronic Converter (DPEC) and commutation strategy

An overview on Indirect Modulation Technique for Matrix Converter (MC) can be helpful to explain Space Vector Modulation (SVM) technique and to achieve the equations of switching duty-cycles to control the IGBTs in both the rectification stage and the inversion stage. The purpose of implementation of the SVM technique is to obtain sinusoidal input and output waves by combination of duty-cycles of the two stages in Direct Power Electronic Converter (DPEC). The specific commutation strategy for bidirectional switches are explained for the purpose of harmonic minimization. A detailed description and analysis of SVM hexagons both in rectification and inversion stages are presented in this chapter for further consideration in programming the control strategy as explained in the next chapters.

3.1 Introduction to indirect modulation

The main idea of the indirect modulation technique [6] is to consider MC as a two-stage transformation converter: a rectification stage to provide a constant virtual dc-link voltage U_{pn} during the switching period by mixing the line-to-line voltages in order to produce sinusoidal distribution of the input currents, and an inverter stage to produce the three output voltages.

The indirect modulation model uses only active vectors. Fig. 3.1 shows the converter model when the indirect modulation technique is used.

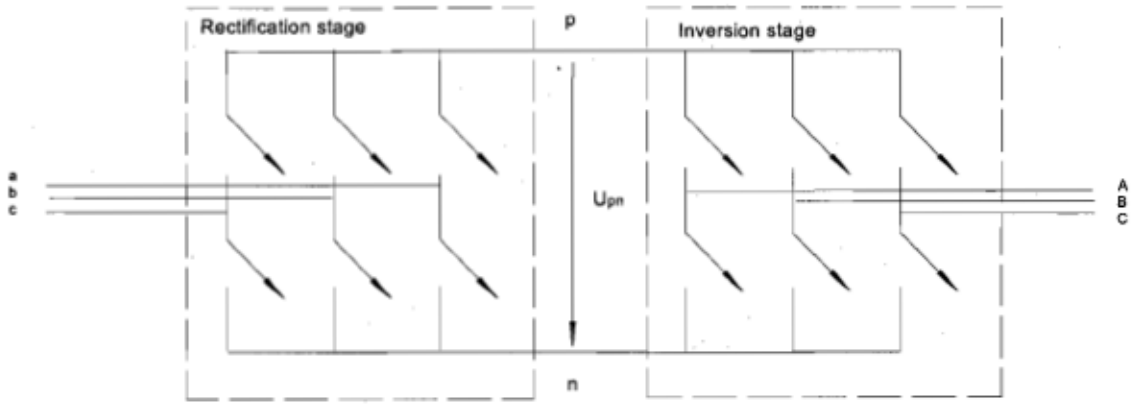


Figure 3.1 Converter model used for indirect modulation technique

3.2 Indirect modulation for MC

The $3\Phi-3\Phi$ Matrix Converter (MC) topology [20] with an input filter and an ac motor load is shown in Fig. 3.2. Because the MC is supplied by an ac voltage source, the input should never be shorted and due to the inductive nature of the load, the output phase must not be left open. A switching function [7] is defined as:

$$s_{jk}=\begin{Bmatrix}1, s_{jk}\,closed\\ 0, s_{jk}\,open\end{Bmatrix}, j\in\{A,B,C\}, k\in\{a,b,c\} \tag{3.1}$$

so that these constraints can be expressed as $s_{ja}+s_{jb}+s_{jc}=1$, $j\in\{A,B,C\}$. Then, the $3\Phi-3\Phi$ MC switches can be assumed only to take 27 allowed combinations. The following expressions for the output line voltages and input phase currents are directly obtained:

$$V_{oL}=\begin{bmatrix}v_{AB}\\ v_{BC}\\ v_{CA}\end{bmatrix}=\begin{bmatrix}s_{Aa}-s_{Ba} & s_{Ab}-s_{Bb} & s_{Ac}-s_{Bc}\\ s_{Ba}-s_{Ca} & s_{Bb}-s_{Cb} & s_{Bc}-s_{Cc}\\ s_{Ca}-s_{Aa} & s_{Cb}-s_{Ab} & s_{Cc}-s_{Ac}\end{bmatrix}\begin{bmatrix}v_{a0}\\ v_{b0}\\ v_{c0}\end{bmatrix}=T_{PhL}.V_{iPh} \tag{3.2}$$

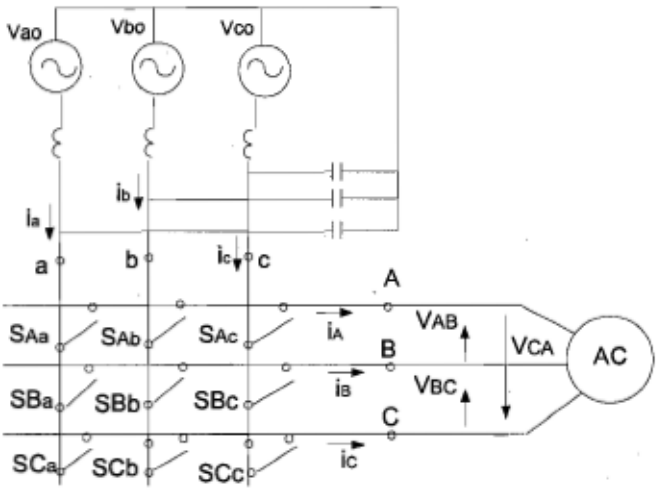


Figure 3.2 Simplified MC topology

$$i_{iPh} = \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} = T_{PhL}^{T} \cdot \begin{bmatrix} i_{AB} \\ i_{BC} \\ i_{CA} \end{bmatrix} = T_{PhL}^{T} \cdot i_{oL} \tag{3.3}$$

where T_{PhL} is the converter transfer matrix. V_{oL} is the output line voltage, V_{iPh} is the input phase voltage, i_{iPh} is the input phase current, i_{oL} is the output line current, V_{im} and V_{om} are the maximum input and output voltages. Fig. 3.3 shows indirect model of matrix converter including two parts, VSR and VSI.

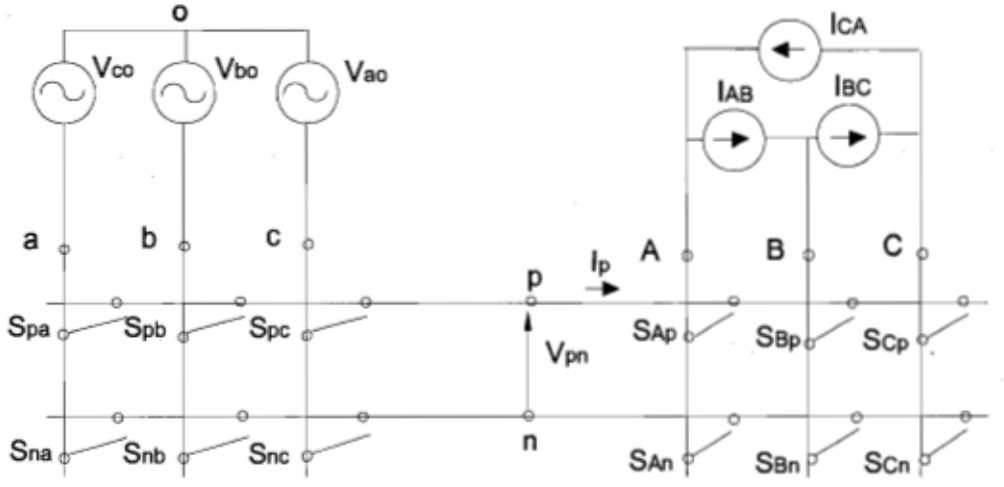


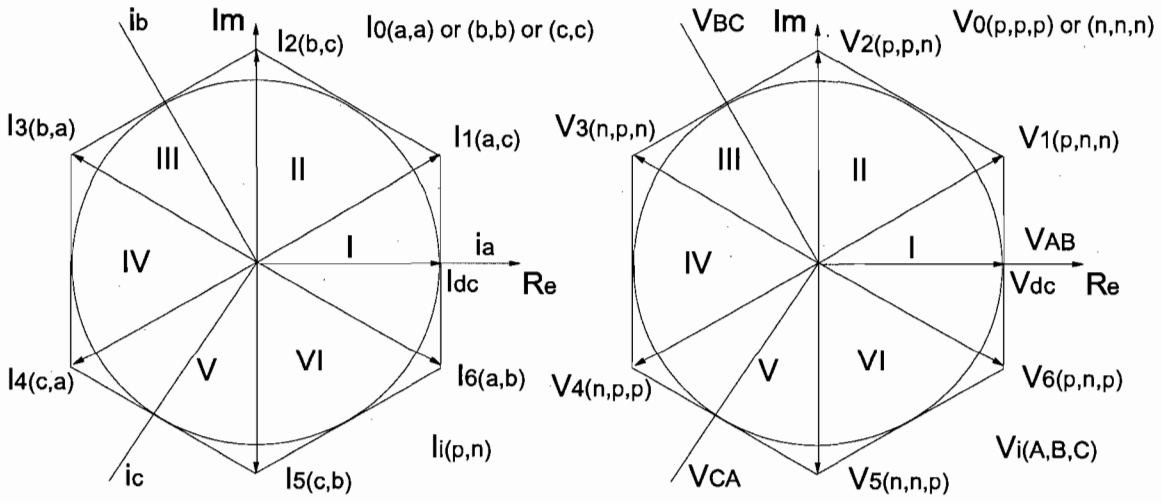
Figure 3.3 VSR-VSI conversion in indirect matrix converter model

3.3 SVM technique for Indirect Matrix Converter (IMC)

In this section based on the ITF approach, the space vector modulation is simultaneously employed in both VSR and VSI parts of $3\Phi-3\Phi$ Indirect Matrix Converter (IMC). However, there is another control strategy for IMC with simple commutations [5], [13].

3.3.1 VSI-SVM output voltage

We consider the VSI part of the circuit, as a stand-alone VSI supplied by a dc voltage source, $v_{pn}=V_{dc}$. The VSI switches can assume only six allowed combinations that yield non-zero output voltages and two combinations with zero output voltages. We can assume only seven discrete values, V_0–V_6, called voltage switching state vectors (SSVs). The VSI hexagon with six 60° segments within a period of the desired 3Φ output line voltage is shown in Fig. 3.4b and the correspondent duty cycles of the SSVs can be calculated.



(a) VSR-SVM hexagon for rectification stage **(b) VSI-SVM hexagon for inversion stage**

Figure 3.4 VSR-SVM and VSI-SVM hexagons

3.3.2 VSR-SVM input current

Consider the VSR part of the circuit as a stand-alone VSR loaded by a dc current generator $i_p=I_{dc}$. Similar to VSI –SVM vectors, the corresponding VSR duty cycles can be calculated for the desired 3Φ input line current. The VSR hexagon is shown in Fig. 3.4a.

3.4 Combined SVM technique for single drive two-stage DPEC

This section presents the combination of output-voltage and input-current SVM for $3\Phi-3\Phi$ IMC topology, as a two-stage DPEC. By utilization of $3\Phi/2\Phi$ matrix converter power module, in the rectification stage of two-stage DPEC, rectifier acts as a Current Source Rectifier (CSR). The local-averaged output voltage of CSR-SVM and local-averaged input current of the VSI-SVM are constant and it results in $M=m_v \cdot m_c$. For simplicity, it is convenient to choose $m_c=1$ and $M=m_v$. Since both the VSI and CSR hexagon contain six sectors, there are 6*6=36 combinations or operating modes in one period and the combined duty cycles can be calculated [5].

3.5 Combined SVM technique for multi-drive two-stage DPEC

It is possible to connect several dc/ac conversion stages to the same dc-link [21]. Semi-sinusoidal line current and output voltage can be obtained, if the switching patterns of the inversion stages are synchronized with the rectification stage. The adjusted duty-cycles of the rectification stage are given by:

$$d_\mu^R=\frac{d_\mu}{d_\mu+d_v} \qquad d_v^R=\frac{d_v}{d_\mu+d_v} \tag{3.4}$$

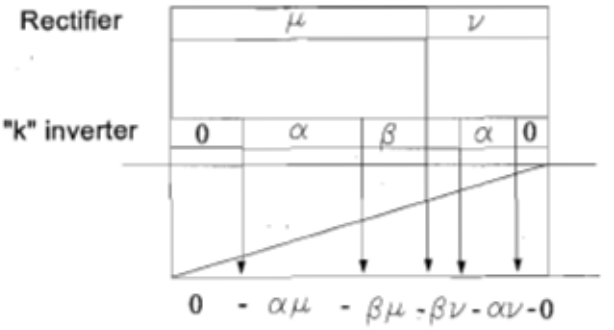


Figure 3.5 Synthesis of duty-cycles and switching state for the "k" inversion stage

where the modulation index of the rectification stage is unity and in fact these are weight of duty-cycles which directly drive the rectification stage. Since the average voltage in the dc-link is not constant anymore due to the cancellation of the zero-vector in the rectification stage, then

$$V_{pn} = d_{\mu}.V_{line-\mu} + d_{\nu}.V_{line-\nu} \quad \text{and} \quad m_U^k = \sqrt{2}.V_{out}^k / V_{PN} \tag{3.5}$$

The inverter stage uses a double sided switching sequence as follows: $0-\alpha-\beta-\beta-\alpha-0$, which is asymmetrical (Fig. 3.5), because each side should apply on the rectification sequence duty-cycles, where:

$$d_{\alpha}^k = m_U^k.\sin(\frac{\pi}{3} - \theta_{out}^{*k}) \quad , \quad d_{\beta}^k = m_U^k.\sin(\theta_{out}^{*k}) \tag{3.6}$$

Therefore, the duty-cycles [22] for the switching sequences of any of the "k" inversion stages are found:

$$d_1^k = d_{\mu}.d_{\alpha}^k, \;\; d_2^k = d_{\mu}.d_{\beta}^k + d_{\nu}.d_{\beta}^k, \;\; d_3^k = d_{\nu}.d_{\alpha}^k \tag{3.7}$$

$$d_0^k = d_{01}^k + d_{02}^k = 1 - (d_1^k + d_2^k + d_3^k) = 1 - (d_{\mu} + d_{\nu}).(d_{\alpha}^k + d_{\beta}^k) \tag{3.8}$$

$$d_{02}^{k} = \frac{d_v}{d_\mu + d_v} \cdot d_0^k = \frac{d_v \cdot [1 - (d_\mu + d_v) \cdot (d_\alpha^k + d_\beta^k)]}{d_\mu + d_v} \tag{3.9}$$

$$d_0^k = \frac{d_\mu}{d_\mu + d_v} \cdot d_0^k = \frac{d_\mu \cdot [1 - (d_\mu + d_v) \cdot (d_\alpha^k + d_\beta^k)]}{d_\mu + d_v} \tag{3.10}$$

The total zero-vector duty-cycle consists of two parts. Then, for calculation of each part of zero vector duty-cycle, which directly depends on duty-cycle in rectification stage, it should be multiplied to the weights, which are $\frac{d_\mu}{d_\mu + d_v}, \frac{d_v}{d_\mu + d_v}$.

3.6 Analysis for VSI-SVM and CSR-SVM hexagons

The VSI and CSR hexagons of SVM modulation are used for DPEC converter. The creation of these hexagons and division into 6 equal 60° segments are analyzed. The material presented in this section is original and based on the three phase input voltage analysis by the author.

3.6.1 VSI-SVM output voltage hexagon for DPEC

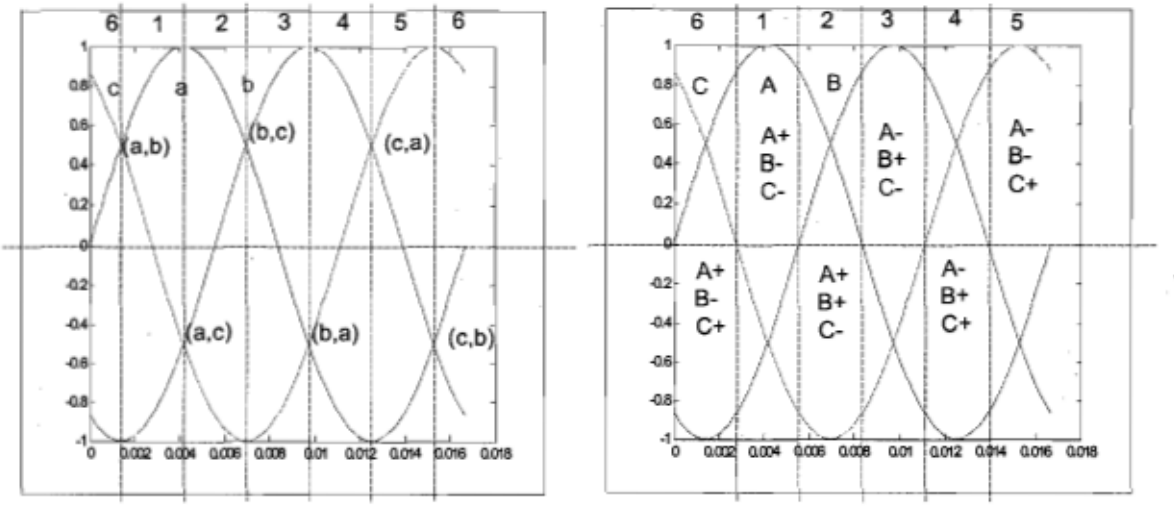
The inversion stage task in two-stage DPEC is to connect the inductive load directly to the dc-link. To produce a sinusoidal output voltage by the inversion stage, we should define a switching sequence for the 6 switches (IGBT) in the inversion stage that corresponds to positive sequence for the 3 voltage phases.

As shown in Fig. 3.3, each output phase is connected to two IGBT's so that by switching the upper IGBT in each phase, the inductive load will be connected to the positive voltage potential of the dc-link and in reverse, switching of the lower IGBT provides connection of the inductive load to the negative potential of dc-link.

Therefore by consideration of 3 phase output voltages in Fig. 3.6b, we note that the load is connected through each phase to a defined positive or negative voltage potential of the dc-link until the sign changes in one of the 3 phases. This will show 6 vectors as 6 different switching states during one period of 3 sinusoidal output voltage phases. Then, it results 6 switching states so that the output voltage vector simultaneously is located between two different switching states as a sector of 60° segment that corresponds on VSI-SVM hexagon description.

The 6 different switching states regarding of positive sequence of 3 output voltage phases are (A+,B-,C+), (A+,B-,C-), (A+,B+,C-), (A-,B+,C-), (A-,B+,C+), (A-,B-,C+) so that we assume the positive potential of the dc-link as p and the negative voltage as n.

By definition $V_1(p,n,n), V_2(p,p,n), V_3(n,p,n), V_4(n,p,p), V_5(n,n,p), V_6(p,n,p)$ where $V_i(A,B,C)$ is an active vector and $V_0(p,p,p), or\, V_0(n,n,n)$ is a zero vector that can be obtained by connection of all three phases to one voltage potential side of the dc-link.



(a) CSR for input current **(b)VSI for output voltage**

Figure 3.6 Description of SVM hexagons

3.6.2 CSR-SVM input current hexagon for DPEC

The DPEC is a convenient solution to achieve integration and the development towards industrial application may be fast. The $3\Phi/2\Phi$ MC power module, in the rectification stage, is constituted of two upside and downside $3\Phi/1\Phi$ MC power modules [23] and it provides a dc current to the dc-link. The upside $3\Phi/1\Phi$ module supplies a positive dc current to the positive potential of the dc-link and in contrast, a negative dc current to the negative potential of the dc-link by the $3\Phi/1\Phi$ downside module.

Therefore two of the 3 input phases simultaneously should be connected to the dc-link through $3\Phi/1\Phi$ power modules. By conduction of IGBTs in the upside module, the positive potential of one of the input voltage phase at that instant can be connected to the positive side of the dc-link and in contrast, the same for the negative link. Then, by proper combination of switching states corresponding to the "k" inversion stages and unique rectification stage, we can achieve sinusoidal input current and output voltage vectors in two-stage DPEC.

By consideration of 3 input phases in the rectification stage shown in Fig. 3.6a, we can define some proper switching states that adapt to the description of input current CSR-SVM hexagon.

Between intersections of two input vectors during one period, it is seen that one input vector has positive value and other one is negative. Then 6 different intersection instants of input vectors can be referenced to define 6 different switching states and consequently 6 sectors of 60° segment. Therefore the 6 intersection as (a,c), (b,c), (b,a), (c,a), (c,b) and (a,b) during an input voltage period can determine 6 switching states, which indicate that at each switching state, there are two input phases in positive and negative value

connected to the dc-link: $I_1(a,c), I_2(b,c), I_3(b,a), I_4(c,a), I_5(c,b), I_6(a,b)$ where $I_i(p,n)$ is an active vector.

3.7 Bidirectional switches commutation strategies

The greatest advantage of using an overlap commutation strategy, Fig. 3.7c, for bidirectional switches, is to decrease harmonic levels caused by switching. A dead-time commutation, Fig. 3.7b, is necessary to eliminate the risk of shoot-through (short-circuit of the dc-link capacitors through an inverter leg) caused by non ideal commutation characteristics [19]. Figure 3.7a shows the basic circuit and the ideal command signals when the output line output is switched from one input line x to the other input line y, by operating two bidirectional switches Sx and Sy. The commutation technique depends on the type of bidirectional switches employed in the matrix converter hardware. In the case of ideal true four-quadrant switches, there are two types of commutation:

1. Current sign four-step commutation strategy
2. Voltage sign four-step commutation strategy

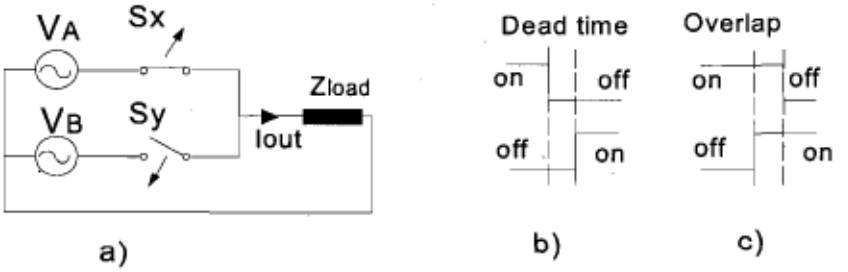


Figure 3.7 Commutation strategy from input phase x to input phase y : (a) principle diagram; (b) dead-time commutation; (c) overlap commutation

3.7.1 Voltage sign four-step commutation strategy

Because of advantage of the voltage sign four-step commutation strategy over the current sign commutation strategy, this commutation strategy is selected for simulation in this research. This specific bi-directional switch commutation technique provides safe and snubber-less operation of the converter. Detail of the four-step bi-directional switch commutations used for the rectification stage in this research is shown in Fig. 3.8.

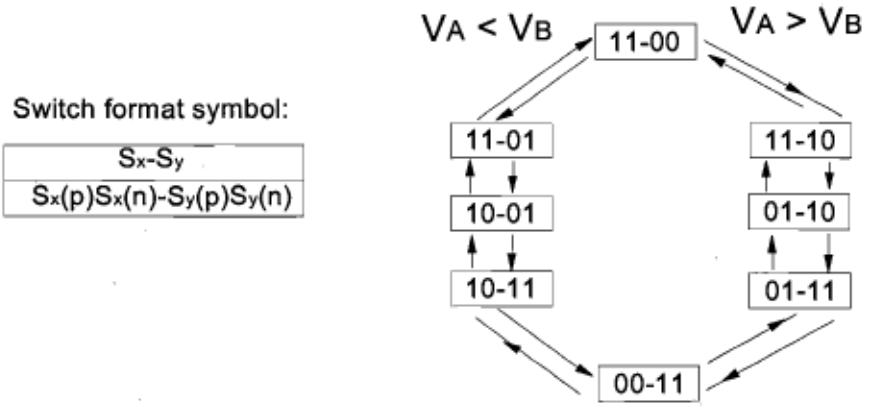


Figure 3.8 Four-step bi-directional switch commutations controlled by the sign of the input line-to-line voltage

In this process, ON status is defined as 1 and OFF status as 0 for simulation so that there is always a correspondent current path by concept of commutation strategy.

3.8 Conduction losses in two-stage DPEC

For a matrix converter, the conduction path for each of the three load currents consists always of an IGBT and a FRD connected in series (anti-paralleled IGBTs). Therefore, for a constant load current, the conduction losses remain constant, which makes the modeling of the conduction losses simple. The losses will vary both with load power factor and

with modulation index. This is because during a zero-vector, all load terminals are connected to the same polarity of the dc-link and the conduction path for the load current consists only of a single power device, an IGBT or a diode, depending on the inversion stage switching state and on the sign of load current. Compared to the current path during an active vector, this causes lower conduction losses. The duration of the zero-vector is given mainly by the modulation index, which is influenced by the ratio between the output frequency and the rated load frequency and the type of voltage profile: linear, quadratic or other V/Hz dependence. The displacement angle of the load dictates the percentage of load current that flows through the diode or IGBT.

3.9 Conclusion

It is shown in this chapter a methodology that can be used to control two-stage DPEC for the purpose of harmonic reduction in input currents. The main idea of this methodology is derived from SVM control in matrix converters by expanding MC topology to two-stage DPEC. Therefore by combination of duty-cycles of these two-stage SVM, CSR-SVM and VSI-SVM, we can achieve sin-input and sin-output waves. Also it has been noticed that for efficiency in harmonic reduction and producing semi-sinusoidal waves, specific control procedures should be followed for bi-directional IGBT switches commutations.

So we can profit by use of this methodology and topology to start simulation and programming in order to realize the general goal of this thesis.

Chapter 4- Simulation of multi-drive two-stage DPEC using Matlab®, Simulink™, RT-Lab™ and Xilinx System Generator™ Softwares

Implementation of the methodology into real time requires to simulate and test the methodology and to optimize the control algorithms so that implementation on hardware can be realized by some proper programming language and platforms. For this purpose, introduction to some platforms used in this research for simulation of the control algorithm implemented to control a multi-drive two-stage DPEC follows. Using proper commands for the control algorithm to achieve the goal of this research is of prime importance. Simulation is done with Matlab® , Simulink™, RT-LAB™ , Xilinx System Generator™ and comparison of frequency spectrums for input line current wave achieved by simulation platforms are shown.

4.1 Introduction to Matlab® and Simulink™

Matlab is high-level language and interactive environment that enables you to perform computationally intensive tasks for algorithm development, data visualization, data analysis and numeric computation. Matlab is a foundation for Simulink and all other The Mathworks Inc. products extended with add-on toolbox in Matlab environment. Simulink is a platform for multi-domain simulation and model-based design for dynamic systems with an interactive graphical environment and a set of block libraries. Simulink simulates the model with variable time step and fixed time step with different solvers. Fixed-point targets processors and Real-Time Workshop. SateFlow as an event-based modeling and a part of Simulink product family is used for simulation of the command algorithm and to

produce PWM pulses implemented on IGBTs of the two-stage DPEC: it is accurate and simple for implementation.

4.2 Introduction to RT-LAB™

RT-LAB™ is a distributed real-time platform [24] that facilitates the design process for engineering systems from Simulink dynamic models to real-time with hardware-in-the-loop, in a very short time, at a low cost. Its scalability allows the developer to add computing power where and when needed. It is flexible enough to be applied to the most complex simulation and control problems, whether it is for real-time hardware-in-the-loop applications or for speeding up model execution, control and test.

RT-LAB is fully integrated with Matlab /Simulink and supports models from StateFlow and API environments such as LabVIEW. It provides a specialized blockset and tools for easy separation of the system model into subsystem models that can be executed on parallel target processors, standard PCs running either the QNX Real-Time operating system or RedHawk Linux. In this way, if you need to run a model in real-time that can not be run on a single processor, RT-LAB provides a means of sharing the load over several processors and integrates with Opal-RT's OP5000 (RT-LAB XSG) hardware interface devices for nanosecond precision timing and real-time performance.

The XHP (eXtra High Performance) mode of RT-LAB allows very fast computation of the real-time model on the target system such as simulations for complex systems over distributed processors, with analog and digital I/O, at cycle times below 10 microseconds. You can dynamically select signals to trace, modify any model signal or parameter in real-time with RT-LAB's visualization and control panel.

4.3 Introduction to Xilinx System Generator™ (XSG™)

The Xilinx System Generator™ (XSG™) for Digital Signal Processor (DSP) is a plug-in to Simulink that enables designers to develop high-performance DSP systems for Xilinx FPGAs. Designers can design and simulate a system using MATLAB, Simulink and Xilinx library of bit/cycle-true models.

Then, the tool automatically generates synthesizable Hardware Description Language (HDL) code mapped to Xilinx pre-optimized algorithms. This HDL design then can be synthesized for implementation in Virtex-II Pro Platform FPGAs and Spartan-IIE FPGAs. It provides automatic generation of a HDL testbench, which enables design verification upon implementation.

4.4 Control algorithm for two-stage DPEC based on constant Volts/Hz

This method is based on the assumption that the flux amplitude is constant in steady-state operation. The stator voltage vector for $\omega_K = \omega_s$ is

$$u_s = r_s i_s + j\,\omega_s \psi_s \tag{4.1}$$

from which the normalized stator vector magnitude can be calculated [19].

So that for constant stator flux linkages of $\psi_s = 1/2\pi$, the applied voltage u_S versus per unit stator frequency is shown in Fig. 4.1. Please note that for $r_s = 0$, the relationship between stator voltage magnitude and frequency is linear and (4.1) takes the form $\frac{u_s}{f_s} = 1$, i.e. constant Volts/Hz.

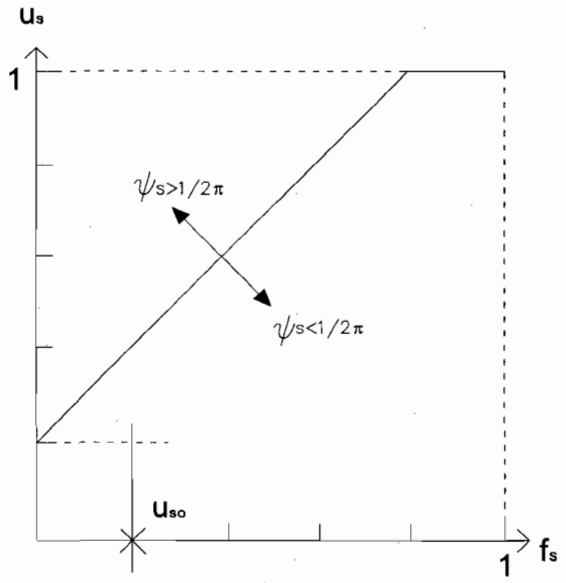


Figure 4.1 Characteristics of stator voltage versus stator frequency for constant stator flux

The closed loop constant Volts/Hz control algorithm [5] is modified as shown in Fig. 4.2. By measuring the input line voltages and motor speeds simultaneously, the gate pulses can be produced in each control stages. More details about some control blocks are considered in this chapter according to each simulation environment.

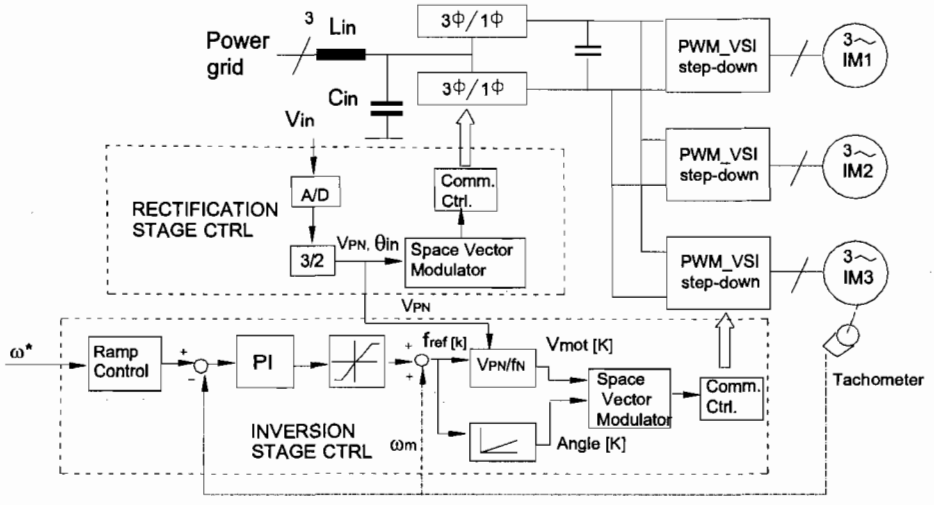


Figure 4.2 Closed-loop constant Volts/Hz control scheme implemented on two-stage DPEC

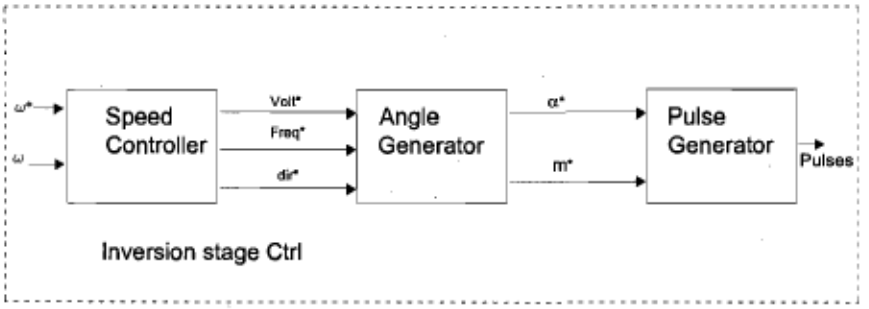
4.5 Simulation of multi-drive two-stage DPEC using Simulink™

In this section, a summary overview on simulation of multi-drive DPEC with 3 induction motor loads (3HP, 220V, 60Hz, 1725rpm) and control algorithms using Simulink™ will be considered. The descriptions of simulation methods for two-stage DPEC are discussed.

4.5.1 Simulation of the DPEC inversion stage

As shown in Fig. 4.3, for inversion stage control simulation the measured speed of the load by tachometer, ω, and speed command signal, ω^*, enter into the speed controller block so that the outputs will be the stator voltage command signal, Volt*, stator frequency command signal, Freq*, and motor direction command signal, dir*, for control of the induction load. When a speed command signal is applied to the system, this change will gradually be implemented by using a ramp control block in the simulation and consequently the speed of the load will vary gradually towards the speed command amplitude. By using a proper PI controller, the difference between the speed command signal and measured speed signal will become approximately zero and also the overshoot of response can be controlled and minimized.

In this control algorithm, the load speed is controlled by both the stator voltage and the stator frequency. Since the speed change occur in the load, both the Volt* and Freq* vary and are not stable. Then it should be defined a method to measure the expected simultaneous angle, α^*, of stator voltage vector.



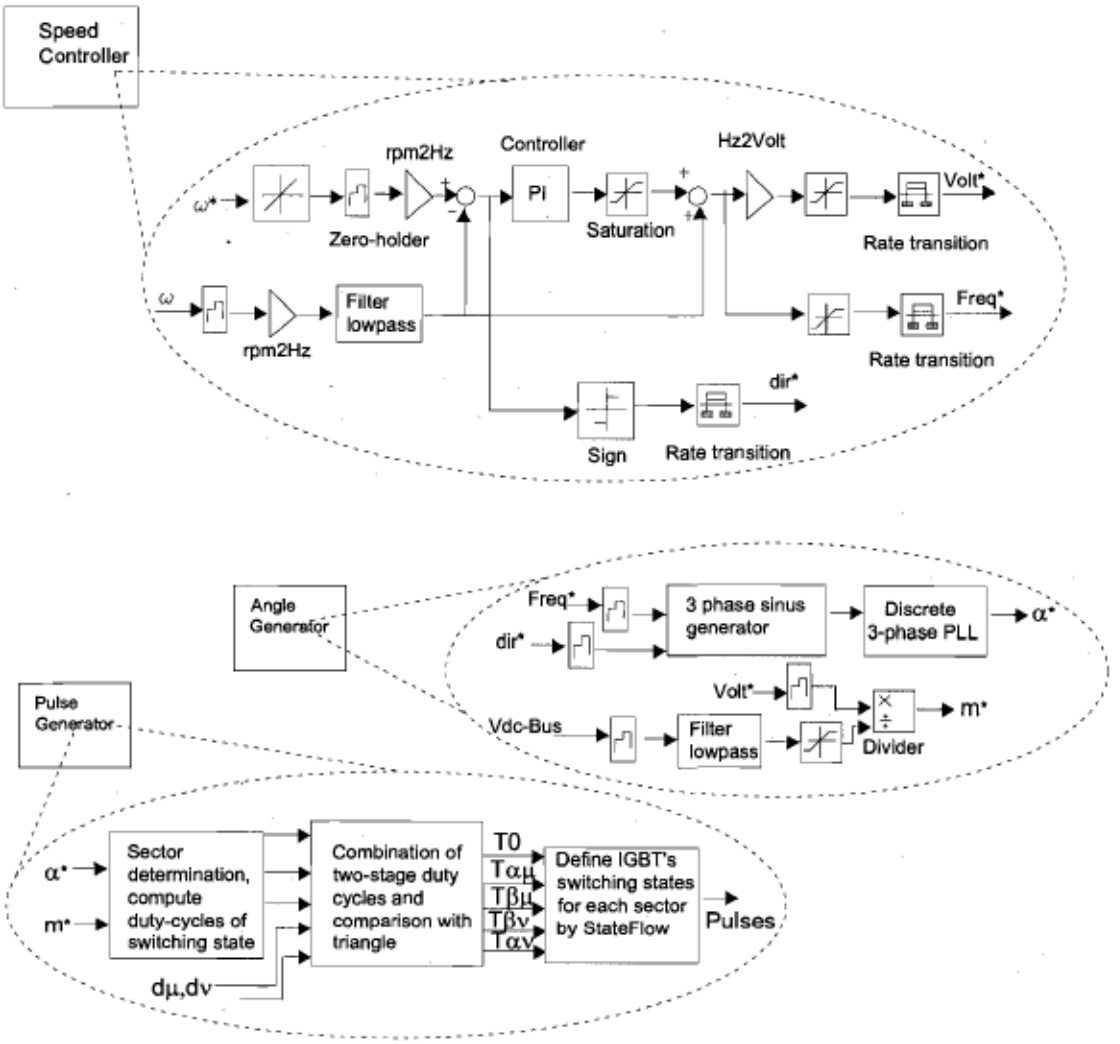


Figure 4.3 Simulation details of inversion stage used in Simulink™

By producing the 3-phase sinusoidal vectors in per unit correspondence on variable frequency as an expected simultaneous stator voltage vector with the same frequency, it will be possible to measure α^* by a discrete 3-phase PLL block. Finally the modulation index, m*, for inversion stage can be calculated by measuring dc-link voltage simultaneously or by calculation from measuring input voltage in the rectification stage.

$$V_{PN} = d_{\mu} \cdot V_{line-\mu} + d_{v} \cdot V_{line-v} = 0.86\sqrt{2}\, U_{line} \tag{4.2}$$

The dir* is used for acceleration and deceleration of the load. In fact determining the sector of the vector is essential so that it allows us to define the correct switching status, ON or OFF, for the correspondence vector for each IGBT in the inversion stage. Combinations of two-stage duty-cycles just determine ON time of each switching state. An example is given by Table 4.1 when the voltage vector and current vector are occurring in sector I. The detail of combination sequence and the time required for each switching state are mentioned for further note.

Table 4.1 Switching sequence example for the voltage and the current vectors occurring in the first sector

SSV Pair	Switching combination					ON Time	Output Line Voltage			Input phase current			Switching states								
	p	n	A	B	C		V_{AB}	V_{BC}	V_{CA}	i_a	i_b	i_c	S_{Aa}	S_{Ab}	S_{Ac}	S_{Ba}	S_{Bb}	S_{Bc}	S_{Ca}	S_{Cb}	S_{Cc}
I_8 - V_6	a	b	a	b	a	$d_{\alpha\mu}.T_s$	V_{ab}	$-V_{ab}$	0	$-i_B$	i_B	0	1	0	0	0	1	0	1	0	0
I_6 - V_1	a	b	a	b	b	$d_{\beta\mu}.T_s$	V_{ab}	0	$-V_{ab}$	i_A	$-i_A$	0	1	0	0	0	1	0	0	1	0
I_1 - V_6	a	c	a	c	c	$d_{\beta v}.T_s$	V_{ac}	0	$-V_{ac}$	i_A	0	$-i_A$	1	0	0	0	0	1	0	0	1
I_1 - V_1	a	c	a	c	a	$d_{\alpha v}.T_s$	V_{ac}	$-V_{ac}$	0	$-i_B$	0	i_B	1	0	0	0	0	1	1	0	0
I_0 - V_0	a	a	a	a	a	$d_0.T_s$	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0

A triangle wave is used to be compared with the switching times so that the carrier wave frequency defines the number of samplings of vector status per sector. Then the switching sample period is equal to the total switching time per sample.

StateFlow provides facilities and strong tools for design, especially those described with sequences and time, so it is used in this thesis to increase the accuracy of switching timing and control of the procedure anytime and anywhere during the simulation. Simulation of switching states of rectification stage, using StateFlow, is shown in Fig. 4.4 so that it consists of six sections S1 to S6, as representative of six sectors, and six states, STATE 1 to 6, which indicate the sequences and status of six IGBTs of rectification stage.

The trigger keys between two states are based on calculation time, as ON Time switching (Table 4.1), when each switching state needs to be remained. So the switching state change occurs only when the concerned trigger time enters to the StateFlow package inside Simulink environment during the simulation. The switching states, in each sector, have been defined according to switching sequences for minimizing the switching interactions with each other and the grid, Fig. 4.9.

In Fig. 4.6, two different duty-cycles for zero state are shown to provide symmetrical sequence of PWM, so zero state used in StateFlow is implemented by two triggers, Tz1 and Tz2. Each sector has its own switching states and when sector conditional signal enters to the SateFlow package, the sector change occurs by comparing the value of that signal with the sector number. The sector conditional signal comes from status of three phases supply voltage to the converter.

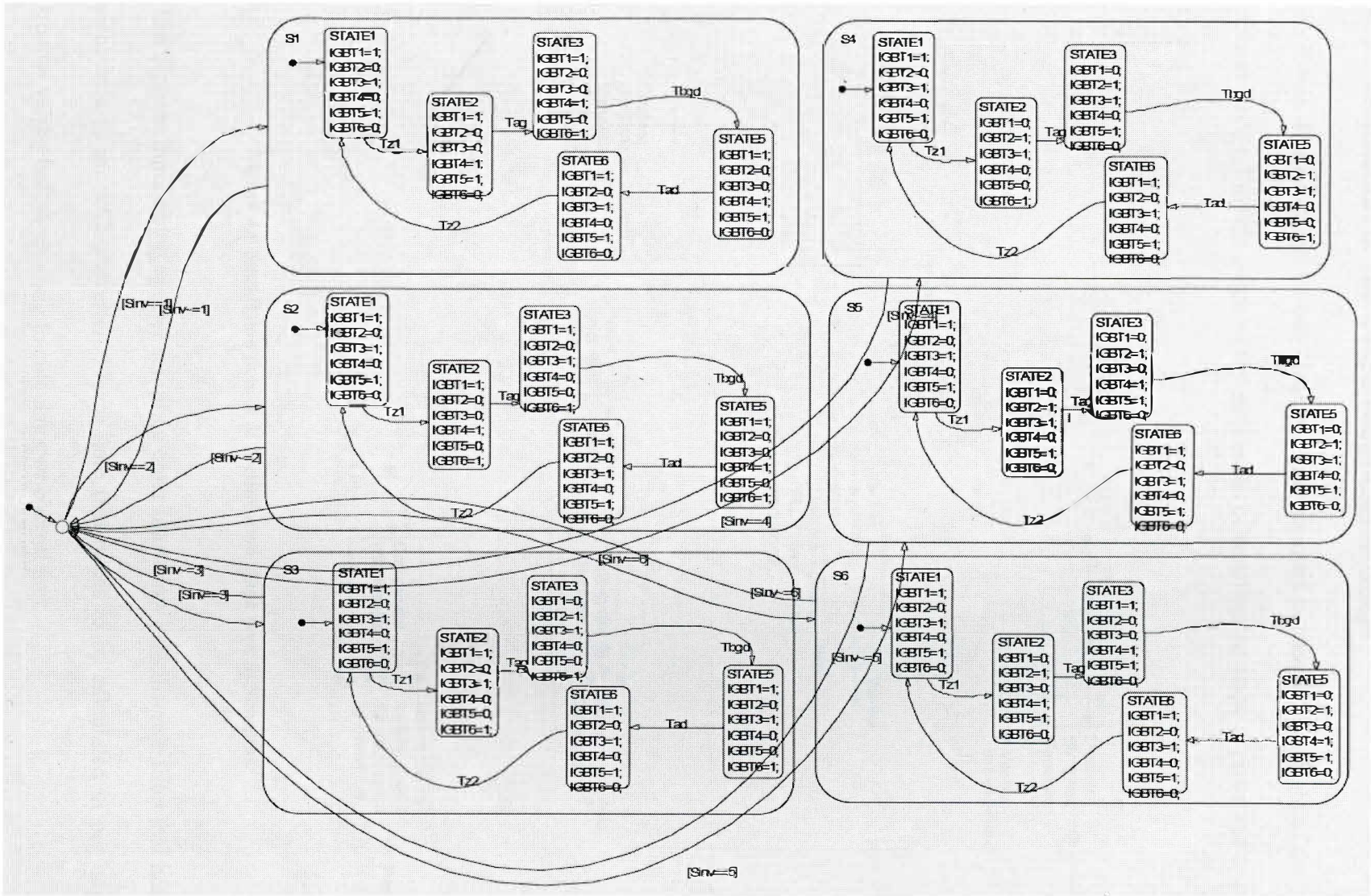


Figure 4.4 Switching states, timing and status of each IGBT in the inversion stage using StateFlow

4.5.2 Simulation of the DPEC rectification stage

As it was discussed in previous chapters, first we need to identify the angle of the input voltage vector. For this purpose, the simulation of rectification stage is presented to detail in Fig. 4.5.

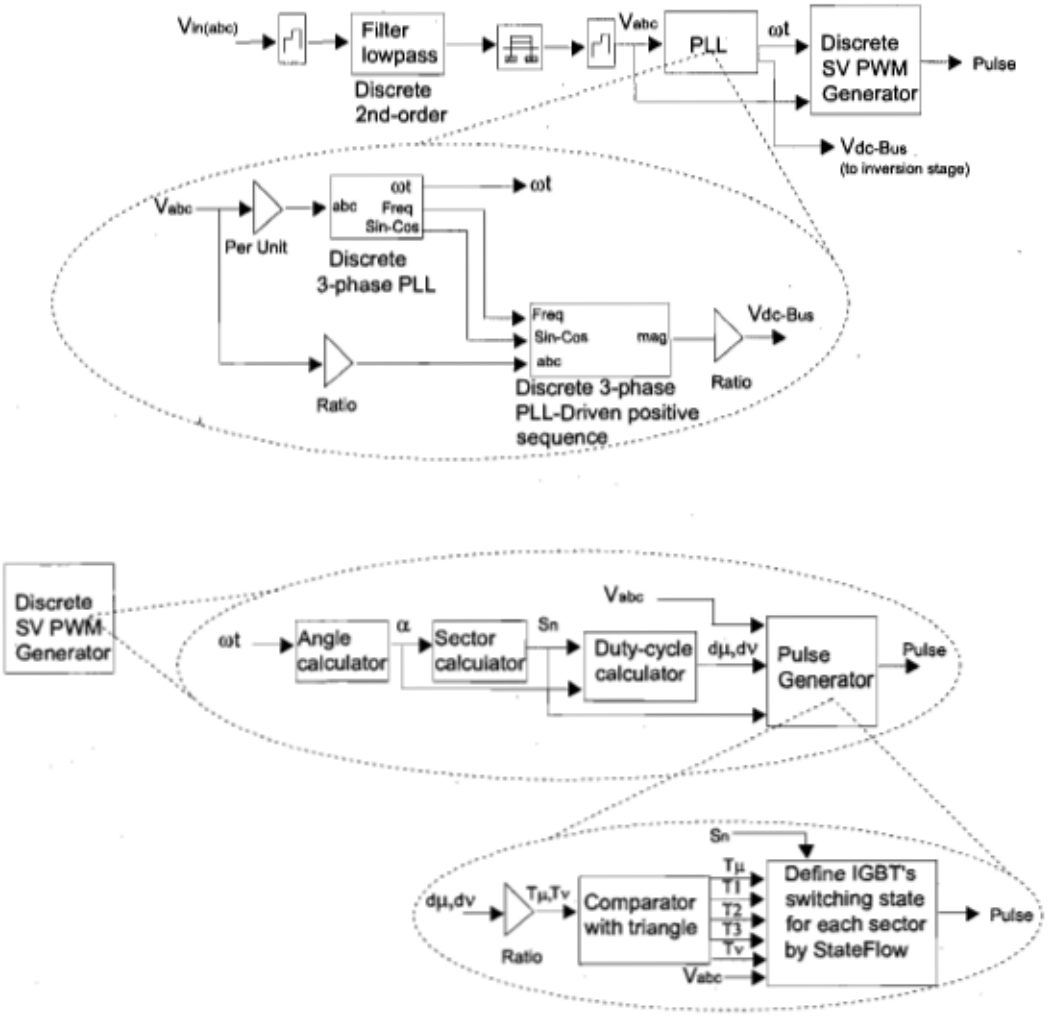


Figure 4.5 Simulation details of rectification stage used in Simulink™

The 3-phase input voltages after sampling enter into a low pass discrete filter that has the same role as an A/D converter. Please note that needs to be calculated in per unit or smaller values before entering to calculator or control unit.

Then, after calculating angle of input voltage vector, the vector sector and duty-cycles can be produced easily. The dc link voltage can be calculated in this section for the purpose of application in inversion stage control unit.

StateFlow platform is used to make easy and accurate definition of switching status. Please note that in this stage, all IGBTs' switching status and sequences are defined with respect to voltage values and signs to minimize harmonics caused by switching interactions as discussed in chapter 3.

4.6 Simulation of multi-drive two-stage DPEC using RT-LAB™ and simulation results for input line current and output voltage

Simulation with Simulink™ platform is performed with variable step type and it is not possible to simulate it with fixed-step type because there are non-linear elements of SimPowerSystems (SPS) in the simulation.

RT-LAB uses Simulink to define models that will be executed by the real time multiprocessing system and it defines its own simulation parameters through Simulink. ARTEMIS™ and RT-Events™ are compatible with the Real Time Workshop (RTW) and RT-LAB™ for real time applications.

4.6.1 Introduction to ARTEMIS™

ARTEMIS™, the Advanced Real-Time Electro Mechanical Simulator, is a modular simulation toolset [25], [27] that includes a performance-enhancing add-on product for SPS. Real-time computation, higher precision for linear circuits, better accuracy with non-linear elements and freedom from numerical oscillations are its features provided for SPS.

4.6.2 Introduction to RT-Events™

RT-Events, the Real Time compensation of switching Events, is a Simulink toolbox for the fixed-time-step simulation of hybrid systems [26] involving dynamics and discrete events asynchronous with respect to simulation clock.

The RTE algorithm is very good at handling difficult fixed-step simulation cases like multiple single time-step discontinuities encountered in the simulation of electric drives and power converters.

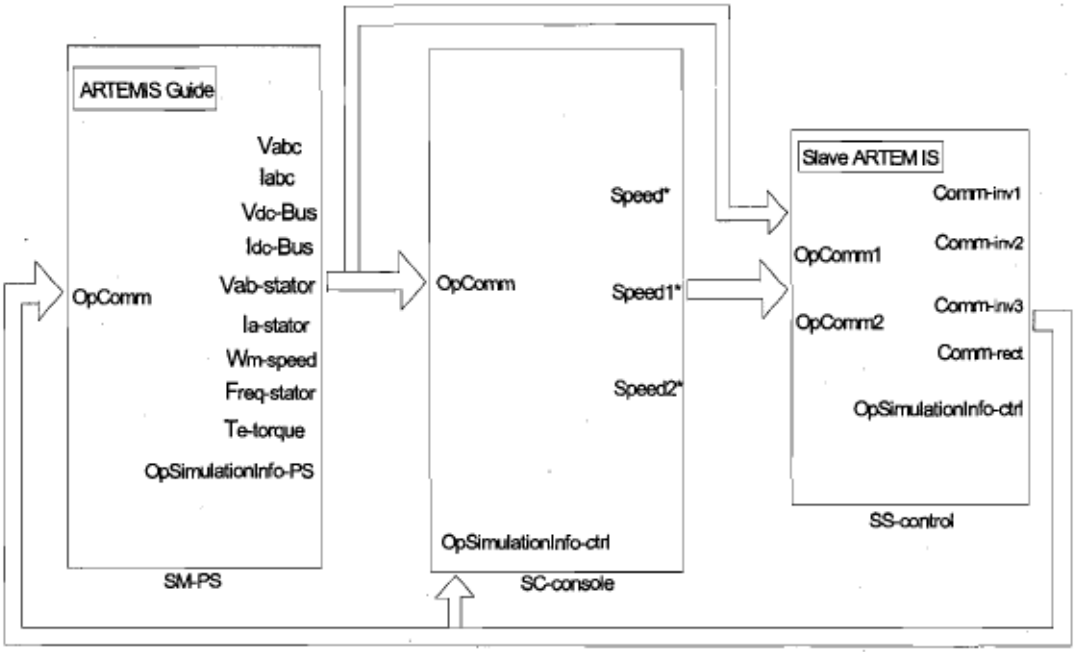
4.6.3 Simulation of the DPEC using RT-LAB™, RT-Events™ and ARTEMIS™

The control algorithm used for this simulation is the same control algorithm defined before. Some Artemis and RT-Events blocks have been used in replacement of some Simulink blocks.

To simulate a model with RT-LAB™ and devote the model to some microprocessors (nodes) for increased calculation and simulation speed with real time workshop, it should build a distributed model including some subsystems.

Each model should have 3 subsystems at least including Console Subsystem (SC), Master Subsystem (SM) and Slave Subsystem (SS) which are connected by OpComm blocks, as shown in Fig. 4.6.

ARTEMIS Guide block in its library permits to implement strictly fixed time step simulations with SPS schematics using highly stable and accurate discrete methods. Slave ARTEMIS block acts the same as ARTEMIS Guide block but can be used for some specific Simulink subsystems.



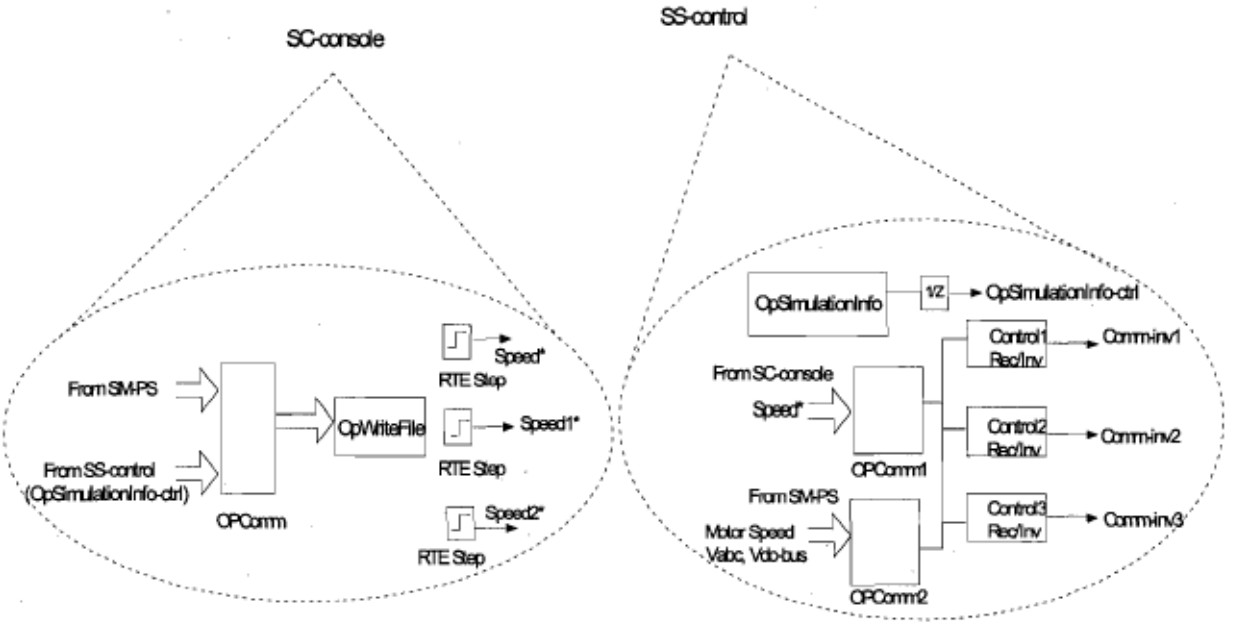


Figure 4.6 Simulation model of two-stage DPEC used in RT-LAB™

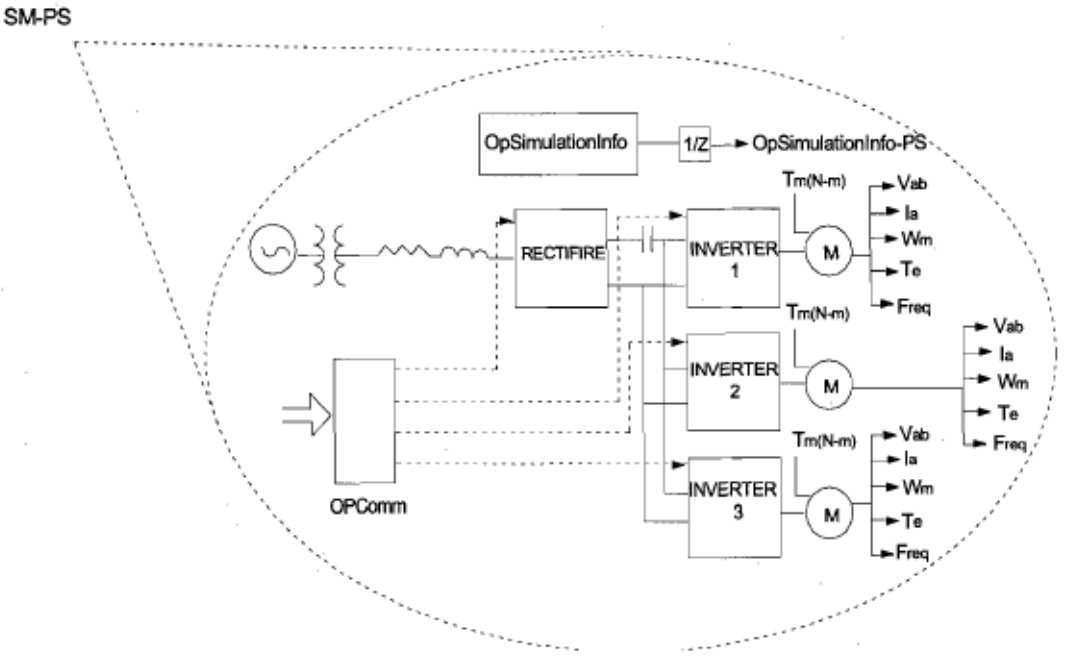


Figure 4.6 (Cont.)

OpWriteFile block is used to save the simulation results in Matlab workspace and OpSimulationInfo block gives various information about the model mode and performance (timing, monitoring).

Simulation has been done with fixed time step and ode4 (Runge-Kutta) solver with 2 CPU Intel Xeon 501MHz, Opal-RT I/O Hardware OP5110-1, SW Firm Ware: S17-0016-PCI-20-4 for 3 induction motor loads (3HP, 220V, 60Hz, 1725rpm) with constant static torques. SimPowerSystems (SPS) compatible 3-leg time stamped bridge block is used as inverter for this simulation and the rectifier has been built by using 3-level IGBT time stamped bridge.

In this simulation, the carrier frequency (Fc) of the triangle wave was assumed 3.6 kHz and the sample time (Ts) was assumed 5e-6s. The information from one subsystem should enter to a separate OpComm block, then the command speeds (speed*) form

console subsystem (SC-console), the operator command entry and viewing console, enters to OpComm1 and the other necessary information for calculation in the control algorithm enters to OpComm2 from Master Subsystem (SM-PS) assigned to power system, such as each of the three motors actual instantaneous speeds, input three phase voltages and dc-link voltage.

It is possible to calculate dc-link voltage by measuring the input three phase voltages and SPS toolset provides the blocks both to measure and to calculate it. For more accuracy of the simulation results, dc-link voltage has been measured directly in this thesis.

Fig. 4.7 and 4.8 show input line current and output voltage of DPEC simulated by RT-LAB without input filter. The correspondent time is achieved by multiplying Ts and sampling number for each point. For example, for the sampling point 6e+4, the time is 6e-2 s.

This simulation is done without input harmonic filter in order to consider only the effect of this method implementation on grid current harmonics. Please refer to Appendix A, for more figures about motor speed, torque and stator frequency.

RT-LAB compiles the simulation model for building the defined subsystems for the microprocessor and provides the possibility for user to devote the built subsystem to an optional microprocessor or node.

An RT-LAB compilation file and nodes transferring and execution files are mentioned in Appendix B, for further details. Obviously by increasing Fc, the sampling frequency of the input voltage per period, the harmonic content decreases. Please refer to Appendix A, Fig. A.8 and A.9.

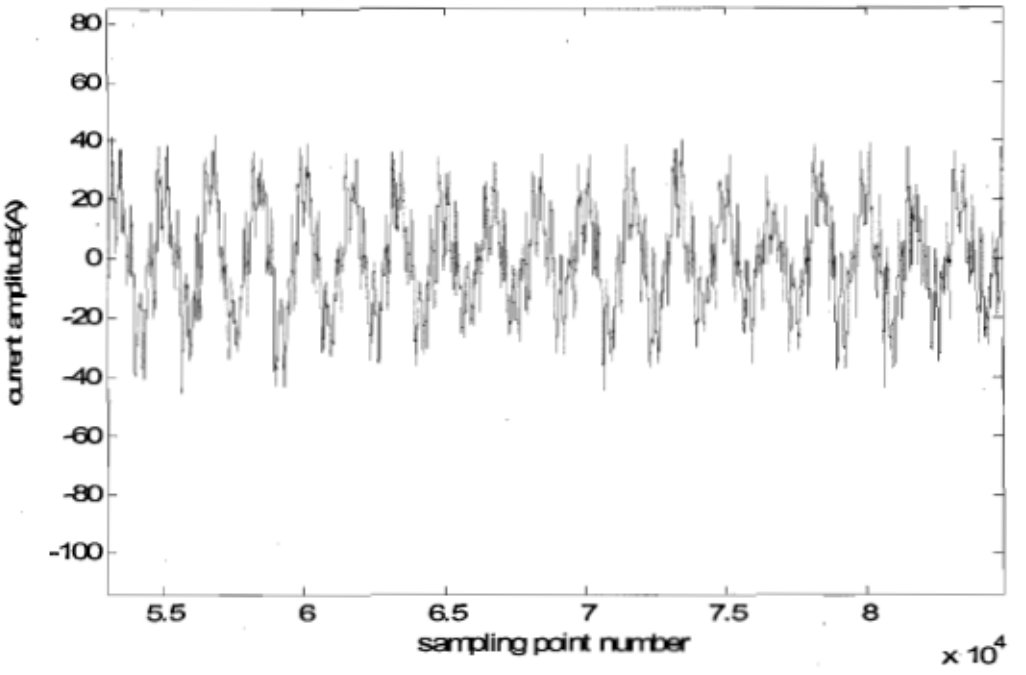


Figure 4.7 Input line current of DPEC simulated by RT-LAB without input filter, Fc=3.6 kHz, Ts=5e-6 s

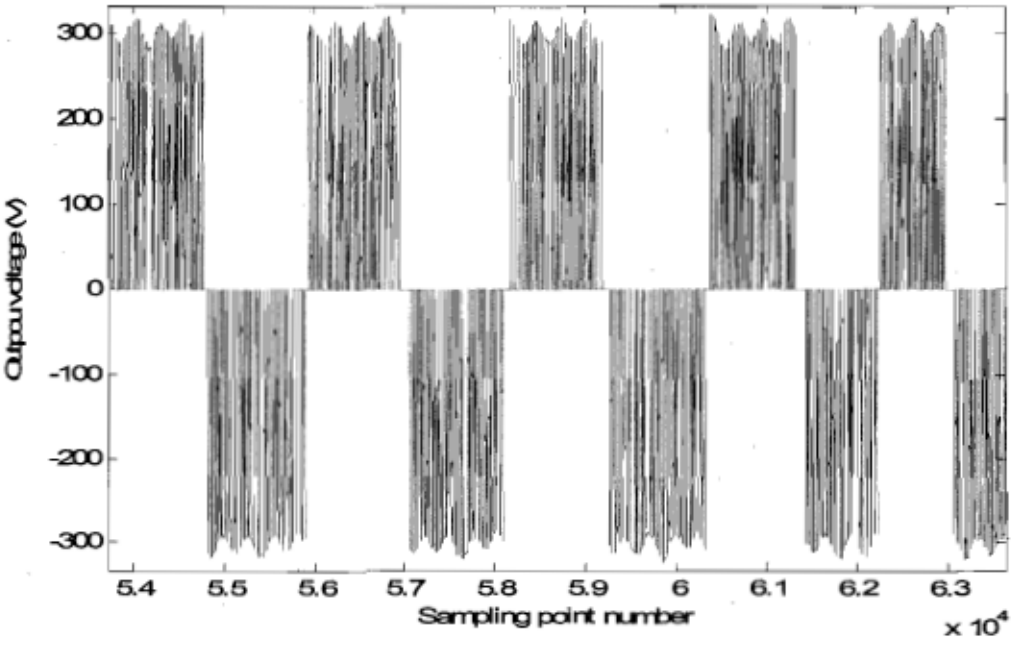


Figure 4.8 Output voltage of DPEC simulated by RT-LAB, Fc=3.6 kHz, Ts=5e-6 s

4.7 Simulation of multi-drive two-stage DPEC using XSG™ and simulation results for input line current and output voltage

HDL codes can be generated by using the XSG™. In fact this part of my research is an initiative to further develop the previous researches to find a solution to implement this control algorithm in FPGAs. Building a simulation model, especially a SPS model, with XSG required a lot of simulation tests in part-to-part of the model to acknowledge the results. Then, using the proper blocks of XSG, programming with Matlab M-file, HDL programming with ModelSim® or using StateCAD can help to achieve the goal.

4.7.1 Introduction to StateCAD

StateCAD, as an accessory file of Xilinx ISE 7.1i, is a graphical entry tool that allows engineers and hardware designers to express their ideas without the use of handwritten documentation such as a state diagram. As a tool for digital design, it includes StateBench, test bench generation and behavioural verification, optimization. It generates synthesizable HDL code in VHDL or Verilog directly from a diagram.

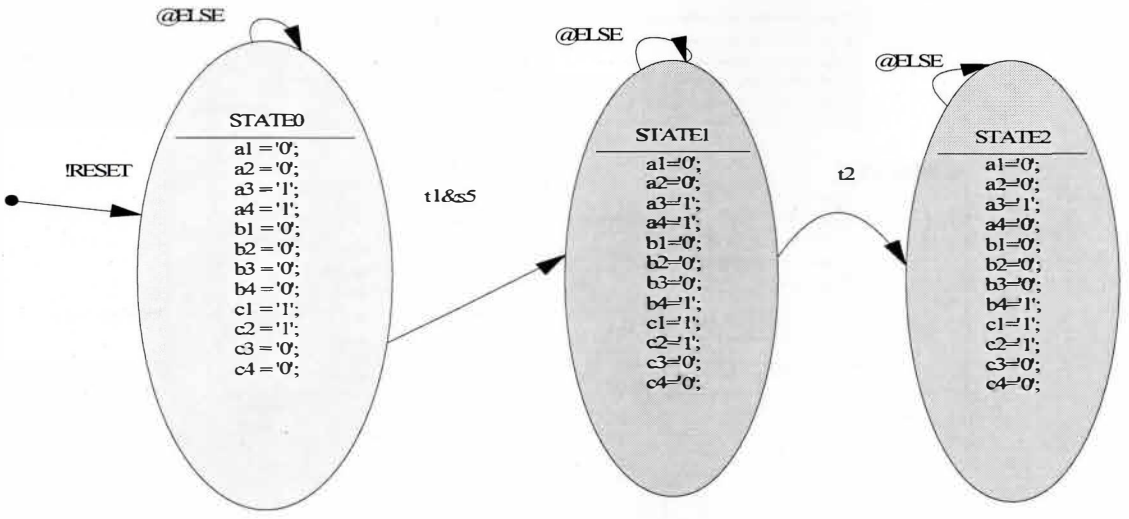


Figure 4.9 State machine defined in StateCAD environment used for HDL code generation

A defined state machine used in StateCAD environment is shown in Fig. 4.9 with appropriate conditions on each transition vectors from one state to another state. This state machine functionality is similar to the StateFlow state machine.

4.7.2 Simulation of DPEC with XSG™ and Simulink™

The whole simulation model of the control algorithm should be re-built with XSG library. Power system model and console model would be like the Simulink models because the goal is HDL code generation for the control algorithm. XSG provides the tools called Gateway In and Gateway Out for connecting the two environments. Black box in XSG library is used for importing VHDL program into XSG Simulink environment with interface of some M-files. Mcode block of XSG is also a tool for importing M-file into XSG Simulink environment. Fig. 4.10 shows XSG Simulink simulation model for rectification stage so that input voltage, after acquisition and passing through the Analog-Digital Converter (CAD), enters the comparator block for determination of vector sector. Vector angle is calculated by phase generator block. Two M-files for the comparator

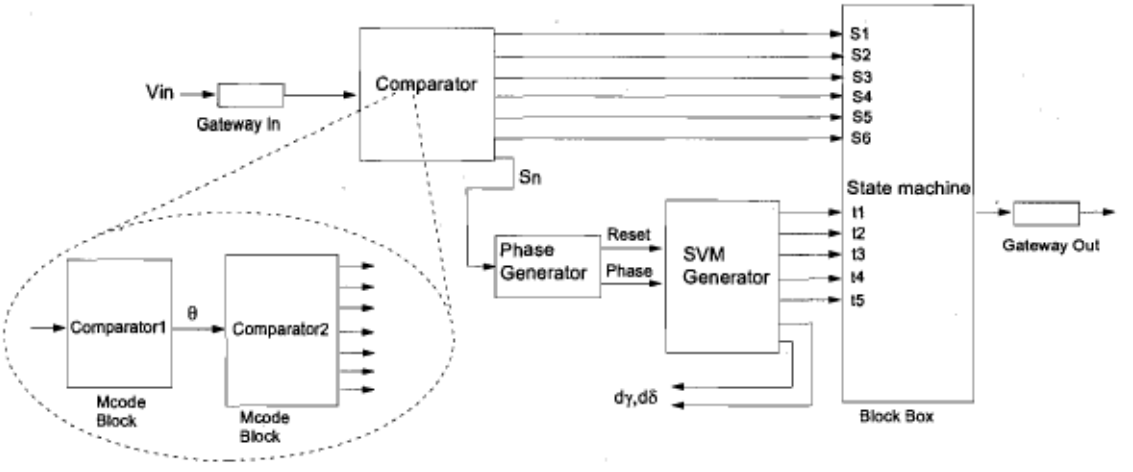


Figure 4.10 XSG Simulink simulation model for control algorithm of rectification stage

block, a VHDL file generated by StateCAD and an interface M-file into the black box are used. Fig. 4.11 shows XSG Simulink simulation for inversion stage. The inputs and outputs are separated form Simulink environment by XSG gateway blocks. Two M-files and a VHDL file are available for this purpose in Appendix C.

VHDL programming can be done in ModelSim environment and also it offers the possibility to test the program. In this research, StatCAD has been used for programming purposes to generate HDL codes. In fact StateCAD is similar to StateFlow so that it is used for hardware.

In Fig. 4.12 and 4.13, the simulation results of input line current and output voltage of the load for Fc=3.6 kHz and Ts=5e-6 s done by XSG and Simulink are shown. The simulation has been done without input harmonic filter for consideration of the method effect on harmonics content in input current and consequently on harmonics content in output voltage.

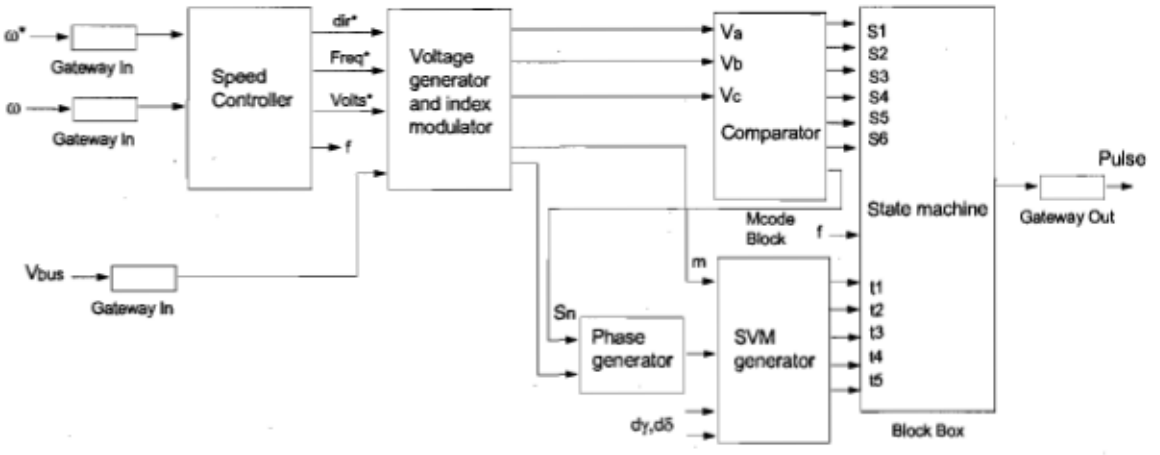


Figure 4.11 XSG Simulink simulation model for control algorithm of inversion stage

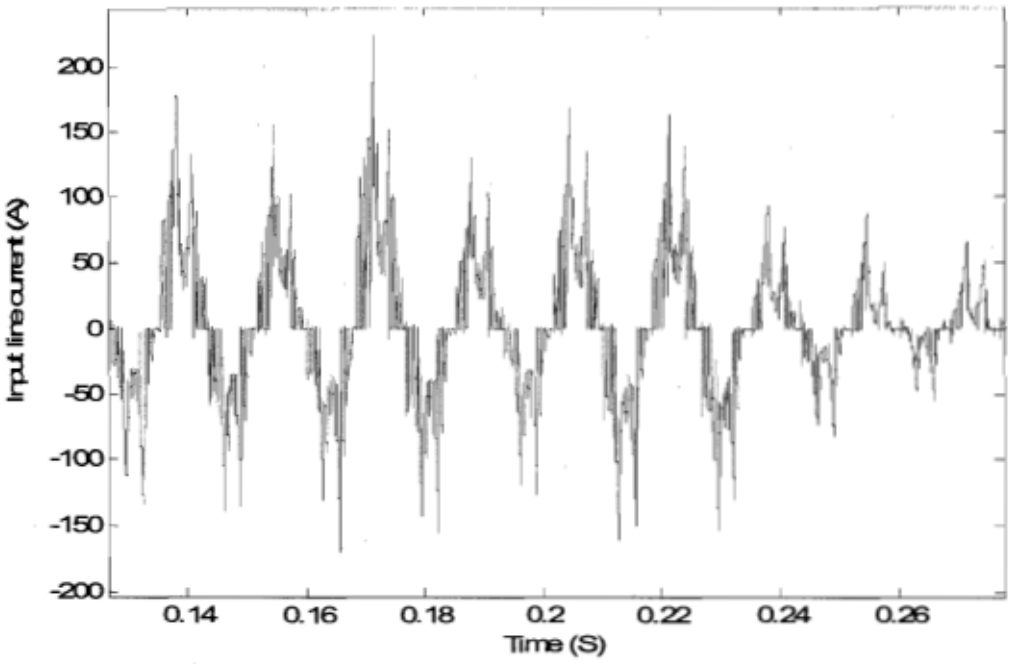


Figure 4.12 Input line current of DPEC simulated by XSG without input filter Fc=3.6 kHz,Ts=5e-6 s

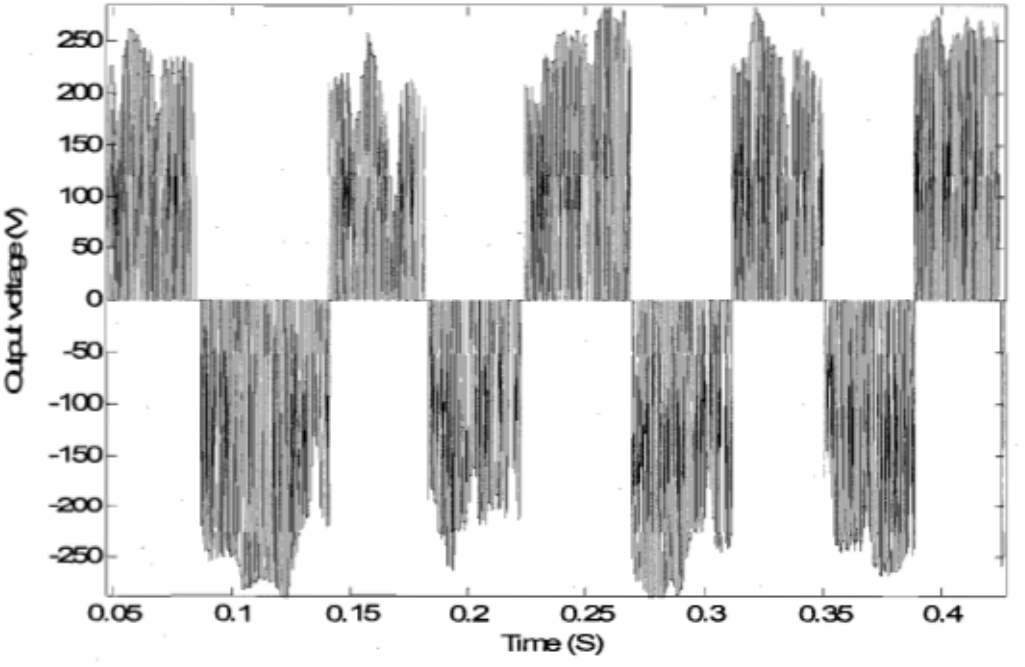


Figure 4.13 Output voltage of DPEC simulated by XSG, Fc=3.6 kHz, Ts=5e-6 s

4.8 Comparison of input line current FFT resulting from RT-LAB™ and XSG™ simulations

Fig. 4.14 shows the Fast Fourier Transform (FFT) of the input line current wave (Fig. 4.7), into the two-stage DPEC simulated by RT-LAB with carrier frequency, Fc=3.6 kHz, and sampling time, Ts=5e-6s. The amplitude of the fundamental harmonic at 60 Hz is more than 14 A. The maximum fundamental amplitude of the input line current is about 21 A in steady-state regime (Fig. 4.7), so its RMS at fundamental is about 21 A/1.4=14.28 A, which verifies the current amplitude at fundamental in Fig. 4.14. The existence of the other noticeable harmonic, 5^{th} harmonic at 300 Hz, results into a semi-sinusoidal wave. The harmonic content of the input line current simulated by XSG with Fc=3.6 kHz and Ts=5e-6 s, but at lower operation motor speed, is shown in Fig. 4.15. The maximum amplitude of input line current at fundamental (60 Hz) is about 32 A in steady-state (Fig. 4.12), which verifies the RMS of fundamental harmonic (about 21.5 A). Then, it results into a semi-sinusoidal wave because the 5^{th} and 7^{th} harmonics have noticeable values in that FFT figure, about 6 A.

The harmonic content of the output voltages, for both RT-LAB and XSG simulations with Fc=3.6 kHz and Ts=5e-6 s, are presented in Fig. 4.16 and 4.17. The fundamental harmonic frequencies of the output voltages are about 75 Hz with RT-LAB simulation and 25 Hz with XSG simulation. For further information, please refer to Fig. A.4 (Appendix A). The different harmonics amplitudes and output voltage frequencies are not due to the simulation tools but to the use of different operation points: different motor speed reference signals. There is no special reason for this choice. Speed reference signal refers to the desired speed of the motor during steady-state as an option of operation.

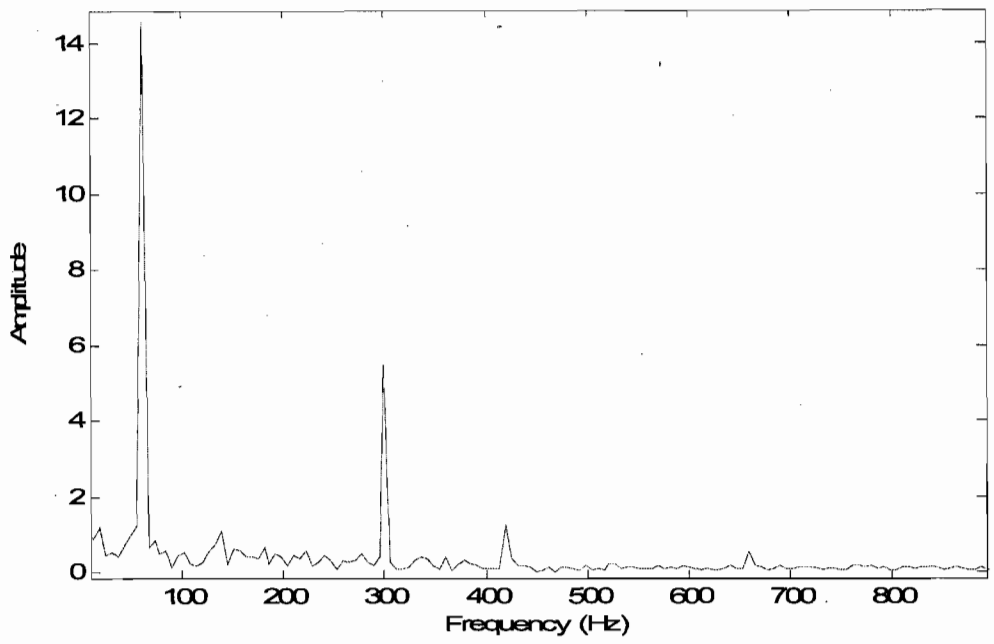


Figure 4.14 FFT of input line current of DPEC simulated by RT-LAB, Fc=3.6 kHz, Ts=5e-6 s.

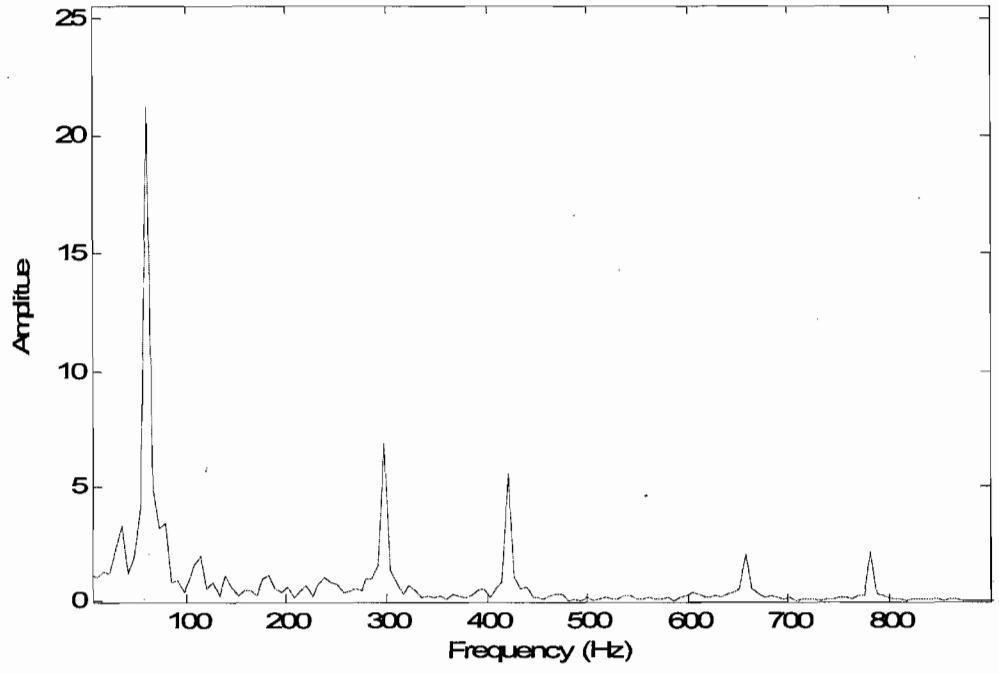


Figure 4.15 FFT of input line current of DPEC simulated by XSG, Fc=3.6 kHz, Ts=5e-6 s.

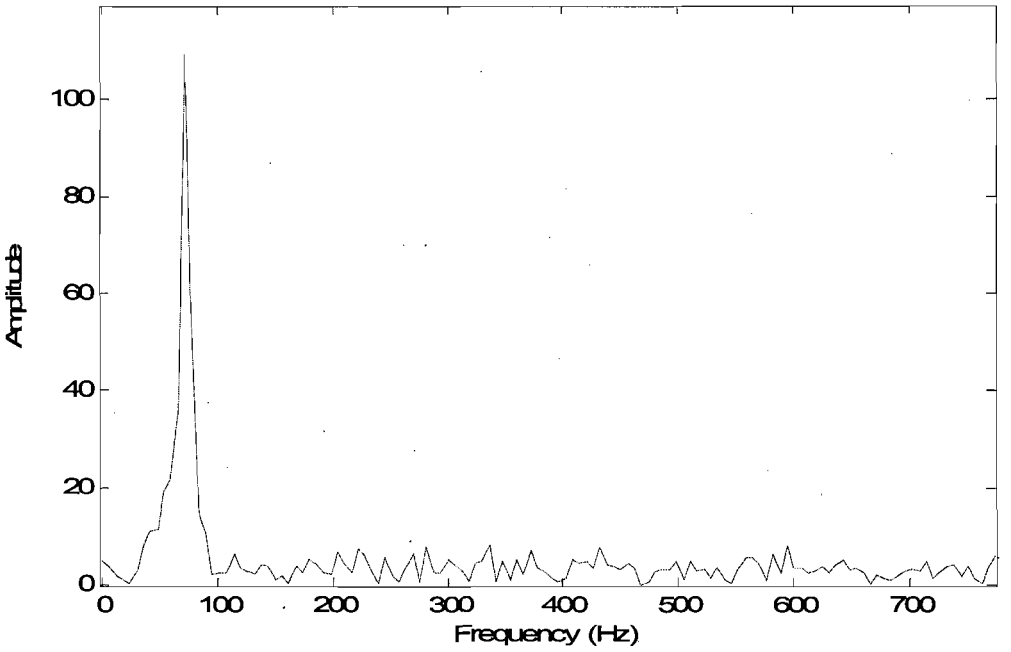


Figure 4.16 FFT of output voltage of DPEC simulated by RT-LAB, Fc=3.6 kHz, Ts=5e-6 s.

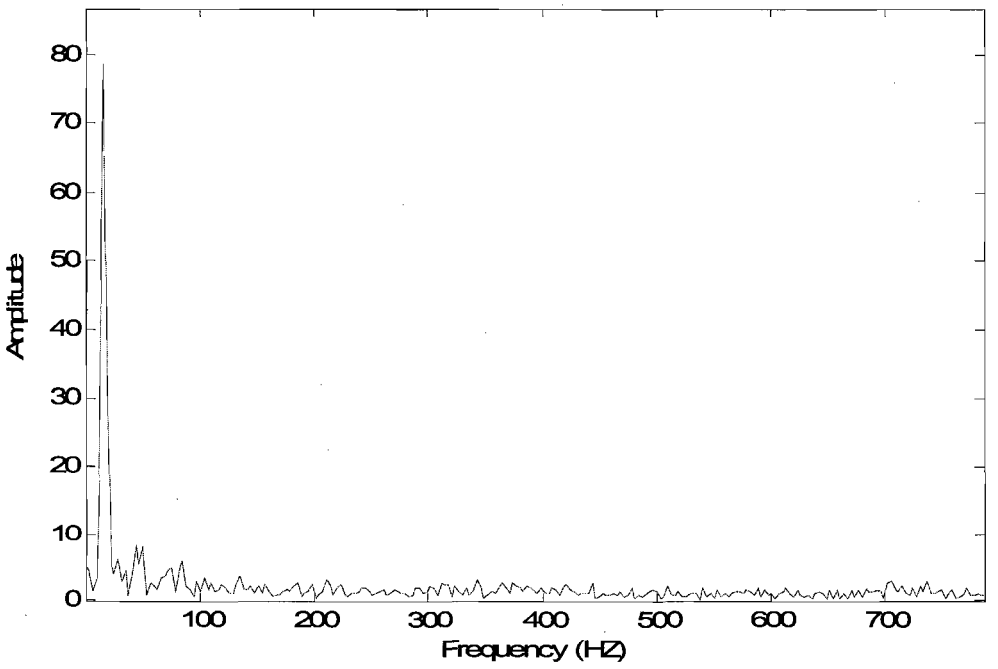


Figure 4.17 FFT of output voltage of DPEC simulated by XSG, Fc=3.6 kHz, Ts=5e-6 s.

By comparison of above FFT results simulated by RT-LAB and XSG, it appears that harmonic frequency contents are approximately similar.

A loose comparison of FFT of input current obtained by XSG and RT-LAB simulations with FFT of input current in article [5] was performed. In [5], the algorithms were implemented on a dual microprocessor (32 bit DSP (2106), 16 bit microcontroller (80C167)) and two programmable logic devices (XC95108). For quite similar operating conditions, there are more harmonic frequencies up to 900 Hz in [5] than in the simulation result of XSG shown in figure 4.15. It appears that, due to bit stream channels of PWM by XSG simulation, the FFT result of input line current is improved.

4.9 Conclusion

This chapter describes an important part of the research, namely the simulation of the hardware by Simulink, RT-LAB and XSG software. For the purpose of understanding the result of control algorithm implementation in hardware, it was simulated by RT-LAB in real time workshop so that all parts of the simulation model, including power system, control part of rectification stage and inversion stage, have been built separately by C code in Opal-RT multiprocessors as hardware implementation.

Simulation with XSG has been experimented because of its essential importance for HDL code generation. XSG blocks definitions are in binary, so it is required to test part-to-part of the XSG control algorithm for making sure every part of the algorithm responds correctly. Implementation of module (control algorithm) in FPGA device, results in a good harmonic reduction of input line current.

Chapter 5- Generation of VHDL codes for the purpose of implementation in FPGAs and code synthesis

In this chapter we describe how to generate VHDL codes, how to synthesize the codes, the code optimization method and the optimized selection of FPGA. A brief introduction to FPGAs and VHDL are also presented.

5.1 Introduction to FPGAs

FPGA (Field Programmable Gate Array) is a general purpose integrated circuit [28]-[29] that is "programmed" by the designer rather than the device manufacturer. A FPGA is programmed by downloading a configuration program called a *bitstream* into static on-chip random access memory. Much like the object code for a microprocessor, this bitstream is the product of compilation tools that translate the high level abstractions produced by a designer into something equivalent. XSG (Xilinx System Generator) pioneered the idea of compiling an FPGA program from a high level Simulink model. A FPGA provides the user with a two-dimensional array of configurable resources (memories, registers, multipliers, lookup tables, buffers, multiplexers, digital clock managers) that can implement a wide range of arithmetic and logic functions. Xilinx FPGA contains sophisticated I/O mechanisms that can handle a wide range of bandwidth and voltage requirements.

5.2 Introduction to VHDL language

VHDL means VHSIC (Very High Speed Integrate Circuit) Hardware Description Language. VHDL defines a hardware description language which can be used for

simulation of digital hardware. VHDL allows us to simulate hardware before realisation. VHDL can be used for simulation by black box. In fact black box provides a way to import VHDL into System Generator designs for assembling a design, simulating and generating hardware. HDL Co-Simulation simulates black box by automatically launching an HDL simulator, generating additional HDL as needed, compiling HDL, scheduling simulation events and handling the exchange of the data between the Simulink and the HDL simulator. VHDL program can be tested and verified by ModelSim.

5.3 VHDL code generation method

The Xilinx basic elements library includes the standard building blocks for digital design. The System Generator, Figure 5.1, is a special Xilinx block that invokes the tool's code generation software. By placing System Generator on your Simulink project sheet, you can generate VHDL and Xilinx LogiCOREs for all the Xilinx blocks on that sheet. The System Generator block parameter allows you to tailor your Simulink simulation and code generation. Henceforth HDL Co-Simulation has been used for black box model simulation.

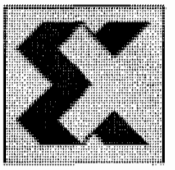


Figure 5.1 System generator block

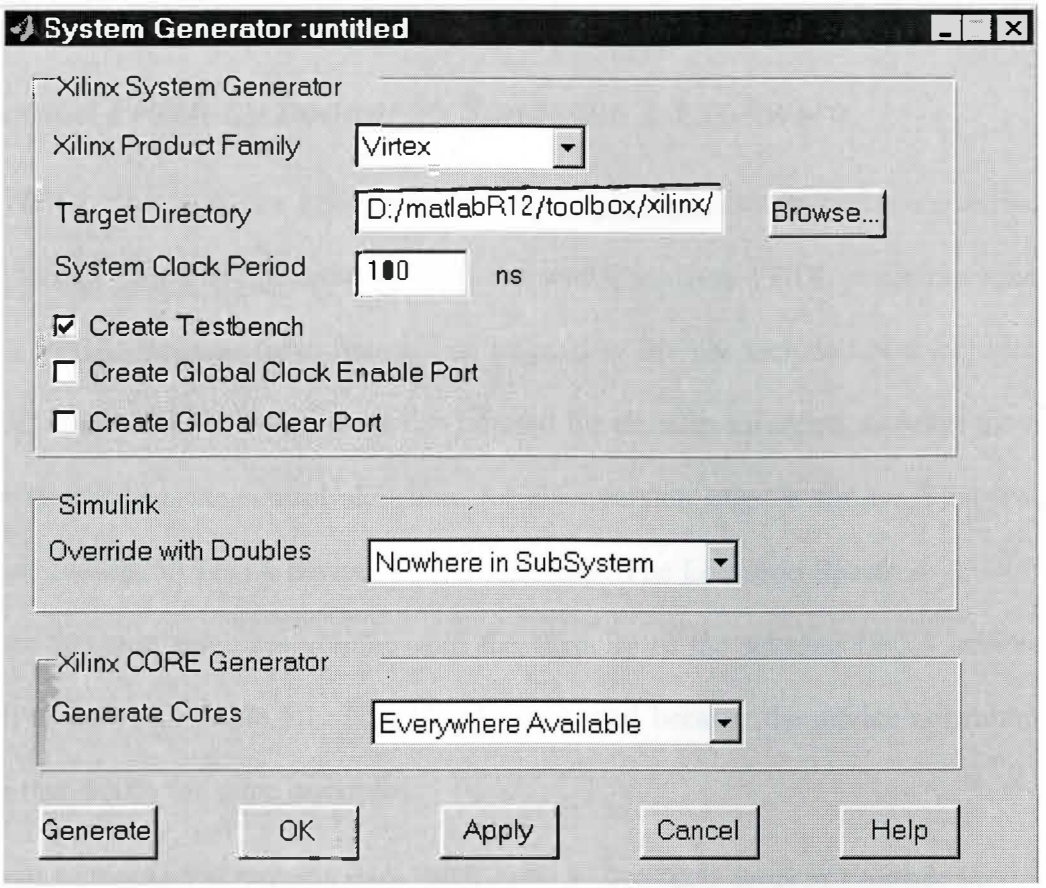


Figure 5.2 System generator block parameters dialog box

Xilinx product families currently include Virtex, Virtex2, Spartan2 and VirtexE. The System Clock Period, as your desired clock period in nanosecond, will be passed to Xilinx software tool and will be used as global PERIOD constraint.

When the Create Testbench box is checked, a VHDL testbench wrapper file and data vector are created for your design. The wrapper file is named to match the top level VHDL file generated for your project. By clicking Generate button in the dialog box, Figure 5.2, VHDL file will be produced.

5.4 Synthesis of generated VHDL codes for an appropriately selected FPGA by Leonardo Spectrum L3 software

The VHDL code files are generated for both rectification and inversion algorithms and analyzed by Leonardo Spectrum L3 software, which analyzes VHDL program application into a FPGA. Because these files are so large, they are not included in this thesis. The VHDL codes of an inversion stage can be used for all other inversion stages in the multi-drive system because control algorithm for all inversion stage is similar. First we have chosen 2v40fg256 FPGA device for this synthesis. The Leonardo Spectrum L3 software checks the code resources volume with the capacity of the selected FPGA device. The result is shown in Table 5.1. This FPGA is rejected because the device utilization was more than 100% for some resources.

Table 5.1 Synthesis of inversion stage VHDL codes for 2v40fg256 device by Leonardo Spectrum

```
*********************************************
       Device Utilization for 2V40fg256
*********************************************
Resource                Used    Avail    Utilization
---------------------------------------------
IOs                     70      88        79.55%
Global Buffers          1       16         6.25%
Function Generators     2861    512      558.79%
CLB Slices              1431    256      558.98%
Dffs or Latches         1519    776      195.75%
Block RAMs              0       4          0.00%
Block Multipliers       9       4        225.00%
Block Multiplier Dffs   252     144      175.00%
---------------------------------------------
This design does not fit in the device specified!
```

Leonardo Spectrum software has the ability to make an optimized FPGA selection for the codes during the synthesis. For the reason that synthesis result for 2v40fg256 FPGA device has not been approved, the VHDL codes of inversion stage has been re-

synthesized for 2V250fg256 FPGA device as an optimal selection of the software. The synthesis result was accepted as shown in Table 5.2.

Table 5.2 Synthesis of inversion stage VHDL codes for 2v250fg256 device by Leonardo Spectrum

```
*********************************************
       Device Utilization for 2V250fg256
*********************************************
Resource                Used    Avail    Utilization
---------------------------------------------
IOs                     70      172      40.70%
Global Buffers          1       16        6.25%
Function Generators     2861    3072     93.13%
CLB Slices              1431    1536     93.16%
Dffs or Latches         1519    3588     42.34%
Block RAMs              0       24        0.00%
Block Multipliers       9       24       37.50%
Block Multiplier Dffs   252     864      29.17%

---------------------------------------------
            Clock Frequency Report
       Clock                : Frequency
     -----------------------------------

       clk_1                : 18.1 MHz
```

For synthesis of the rectification stage VHDL codes, we select 2v80fg256 FPGA. The approved result of the synthesis for device utilization is shown in Table 5.3. As we actually have chosen 10 ns as System Clock Period in System Generator block parameter dialog box, Tables 5.2 and 5.3 show that the clock frequencies of synthesis (18.1, 69.6 MHz) are less than the clock frequency of the generated codes, 100 MHz and also larger than sampling and carrier frequency (1 MHz, 3.6 KHz), which validates the synthesis result. The synthesis detail for VHDL codes of two-stage are mentioned in Appendix D and E.

Table 5.3 Synthesis of rectification stage VHDL codes for 2v80fg256 device by Leonardo Spectrum

```
**********************************************
Device Utilization for 2V80fg256
**********************************************
```

Resource	Used	Avail	Utilization
IOs	104	120	86.67%
Global Buffers	1	16	6.25%
Function Generators	657	1024	64.16%
CLB Slices	329	512	64.26%
Dffs or Latches	110	1384	7.95%
Block RAMs	0	8	0.00%
Block Multipliers	0	8	0.00%
Block Multiplier Dffs	0	288	0.00%

```
---------------------------------------------
Clock Frequency Report
Clock                : Frequency
-------------------------------------
clk_1                : 69.6 MHz
```

5.5 Conclusion

In this chapter, we described how to generate the VHDL codes from an existing XSG model and how to synthesize these codes by special software, Leonardo Spectrum L3. It is shown also how to conclude from a synthesis that the selected FPGA is appropriate for that code implementation or not.

The synthesis of VHDL generated codes presented for both two stages, the rectification and the inversion stages, and optimal FPGAs have been chosen by software indicated in the Tables 5.2 and 5.3. Then, liability of the codes (modulation) for implementation in FPGAs is validated.

For more development in future code utilization in FPGA in order to gain more optimal codes and more optimal FPGAs, as the purpose of engineering design, I have described a procedure in a section as my proposition.

Chapter 6- Conclusion

This research evaluates a method of modulation, which reduces harmonics of input current (semi sinusoidal) in a multi-motor system with 3 AC induction motors and a 3-phase supply voltage. Implementation of the combined SVM methodology on the optimized topology of converter, two-stage DPEC, has provided an effective solution for this research and earned some advantages as following:

1. Minimum low frequency harmonic content in the input current of the converter from the grid because of semi-sinusoidal wave of input current, resulting in the reduction of harmonic pollution in the grid;
2. Semi-sinusoidal wave of output voltage implemented on the induction motor stator, which means minimum harmonic content in output voltage and henceforth the increase of motor insulation life time and accuracy of its operation;
3. Eligibility to control each motor independently and accurately in a multi-drive system;
4. Because of high speed of FPGAs in acquisition, calculation of complex mathematical equations and generating the control command pulses towards the IGBT gates of the converter, it reduces harmonic frequency contents in input current and output voltage due to asynchronous or delayed gate triggering with some other means for implementation;
5. Unity voltage transfer ratio from input to output, less sensibility to grid voltage instability and variation, less power loss in the two-stage DPEC in comparison with matrix converter;
6. Cheaper manufacturing, lighter and more compact than matrix converter.

Simulation of this modulation was realized by Simulink, RT-LAB and XSG and the FFT of input current resulting from RT-LAB and XSG simulations were compared with the input current FFT result of [5], which allowed concluding a good improvement by XSG simulation and eventually by FPGAs implementation. The experiences of this research include evaluation of a method of modulation, creation of a PWM channel, control of some motor parameters such as torque, speed, current, voltage, stator/rotor frequency, measurement of FFT of input current and output voltage.

My important conclusion of this research is the validation of implementation of a control algorithm (used for power converter module) in the FPGAs through XSG, which provides the tools for modeling, simulation and code generation, especially for those which have complex part of computation and can produce the binary PWM channel such as control applications in power electronics.

My conclusion and proposition for future development is optimization of the codes before implementation in FPGAs. Optimization is one of the most important goals in engineering design it allows to achieve cost-effective and compact design size. By knowledge of VHDL programming, we can optimize VHDL generated codes because if we look at the generated codes, each part of the VHDL code is devoted to a specific block of XSG model and all codes are related hierarchically so that we can eliminate most common parts of the codes and make an optimized and unique code.

More optimization of the code also is performed automatically by Leonardo during the synthesis procedure. It should be noted that an optimization needs to be performed carefully and verification is necessary after optimization. I propose to import an optimal VHDL code through an interface M-file into the System Generator environment and

make the command by this new block to the power system, then comparing two simulation results before and after optimization. If they are similar, optimal codes are acceptable.

The hardware optimal selection can be done by synthesis of codes and maximum capacity utilization of different FPGA devices. Then, the most appropriate FPGA that fulfills the requirements such as clock frequency and maximum device capacity utilization (IOs, Global Buffers, RAM, CLB Slices, Function Generator, Multipliers, Latches), less than and closer to 100%, can be chosen as an optimal FPGA.

Bibliography

[1] C. Klumpner, P. Nielsen, I. Boldea, F. Blaabjerg "A New Matrix Converter Motor (MCM) for Industry Applications," IEEE Trans. on Industrial Electronics, Vol.49, No.2, pp.325-335, April 2002.

[2] C. Klumpner, I. Boldea, an F. Blaabjerg, " Short term ride through capabilities for direct frequency converters," in Proc. IEEE PESC'00, vol.1, pp.235-241, 2000.

[3] A. Von Jouanne, P. Enjeti, and B. Banerjee, " Assessment of ride-through alternatives for adjustable speed drives," IEEE Trans. on Ind. Applicat., vol.35, pp.908-916, Jul/Aug. 1999.

[4] J. Holtz and W. Lotzkat, " Controlled AC drives with ride-through capability at power interruption," IEEE Trans. on Ind. Applicat., vol.30, pp.1275-1283, Sep./Oct. 1994.

[5] C. Klumpner, F. Blaabjerg "A New Cost-Effective Multi-Drive Solution based on a Two-Stage Direct Power Electronic Conversion Topology," Proc. of IAS'02, vol.1, pp.444-452, 2002.

[6] C. Klumpner, F. Blaabjerg, I. Boldea, P. Nielson "A New Modulation Method for Matrix Converters," IEEE Conf. on Ind. Applicat., vol.4, pp.2143-2150, Sep./Oct. 2001.

[7] L. Huber, D. Borojevic "Space Vector Modulation Three-Phase to Three-Phase Matrix Converter with Input Power Factor Correction," IEEE Trans. on Ind. Applicat., vol.31, pp.1234-1246, Nov./Dec. 1995.

[8] M.A. Valenzuela, J. Palma, R. Sanchez, " DSP based intersectional control in multi drive systems," Proc. of IAS'97, vol.1, pp.471-477, 1997.

[9] D. Casadei, G. Serra, and A. Tani "Matrix Converter Modulation Strategies: A New General Approach Based on Space-Vector Representation of the Switch State," IEEE Transaction on Industrial Electronics, Vol.49, No.2, pp.370-381, 2002.

[10] T. Rashid, D. Holliday, D.A. Grant "Effect of DC Link Capacitance on the Transient Response of A PWM AC-DC Converter," IEE 'Power Electronic and Variable Speed Drives' Conference Publication No.429 1996.

[11] C. Klumpner, P. Nielsen, I. Boldea, F. Blaabjerg "New Solutions for Low-Cost Power Electronic Building Block for Matrix Converters," IEEE Trans. on Industrial Electronics, Vol.49, No.2, pp.336-343, April 2002.

[12] C. Klumpner, F. Blaabjerg and M. Liserre "Improved Control of an Active-Front-End Adjustable Speed Drive with a Small dc-link Capacitance Under Real Grid Conditions," 35th Annual IEEE Conf. Power Electronics Specialists, pp.1156-1162, Germany 2004.

[13] L. Wei, T.A. Lipo "A Novel Matrix Converter Topology With Simple Commutation," Proc. of IAS'01, vol.3, pp.1749-1754, 2001.

[14] S. Jia, K.J. Tseng, X. Wang "Study on Reverse Recovery Characteristics of Reverse-Blocking IGBT Applied in Matrix Converter," IEEE Conf. on Applied Power Electronics, vol.3, pp.1917-1921, March 2005.

[15] C. Klumpner, F. Blaabjerg "Using Reverse Blocking IGBTs in Power Converters for Adjustable Speed Drives," IEEE Conf. on Ind. Applicat., vol.3, pp.1516-1523, 2003.

[16] J.W. Kolar, M. Baumann, F. Schafmeister, H. Ertl " Novel Three-Phase AC-DC-AC Sparse Matrix Converter," Proc. of APEC'01, vol.2, pp.777-791, 2002.

[17] D. Casadei, G. Serra, and A. Tani "Space Vector Control of Matrix Converters with Unity Input Power Factor and Sinusoidal Input/Output Waveforms," in Proc. EPE Conf., Vol 7, Brighton, U.K, Sept. 13-16, pp. 170-175, 1993.

[18] F. Blaabjerg, D. Casadei, C. Klumpner, and M. Matteini " Comparison of Two Current Modulation Strategies for Matrix Converter under Unbalanced Input Voltage Conditions," IEEE Trans. on Ind. Electronics, vol.49, no.2, pp.289-296, April 2002.

[19] M.P. Kazmierkowski, R. Krishnan, F. Blaabjerg "Control in Power Electronics Selected Problems", A Volume in the Academic Press Series in Engineering Edited by J.D.Irwin, Auburn University, 2002.

[20] C.L. Neft and C.D Shauder, "Theory and Design of a 30-hp Matrix Converter," IEEE Trans. Ind. Applicat., vol.28, pp.546-551, May/June 1992

[21] C. Klumpner and F. Blaabjerg "A New Generalized Two-Stage Direct Power Conversion Topology to Independently Supply Multiple AC loads from Multiple Power Grids with Adjustable Power Loading," 35th Annual IEEE Conf. Power Electronics Specialists, pp.2855-2861, Aachen, Germany 2004.

[22] C. Klumpner, F. Blaabjerg "Modulation Method for a Multiple Drive System Based on a Two-Stage Direct Power Conversion Technology with Reduced Input Current Ripple," IEEE Trans. on Power Electronics ,vol.20, no.4, pp.922-929, 2005.

[23] C. Klumpner, F. Blaabjerg, P. Nielsen "Speeding up the Maturation Process of the Matrix Converters Technology," Proc. of PESC'01, vol.2, pp. 1083-1088, 2001.

[24] S. Abourida, C. Dufour, J. Belanger, G. Murene, N. Lechevin, Y. Biao ''Real-time PC-based simulator of electronic systems and drives'' Proc. of APEC, IEEE 17th annual conference, vol.1, pp.433-438, 2002.

[25] C. Dufour, J. Belanger ''Discrete time compensation of switching events for accurate real-time simulation of power systems'' Ind. Electronics society, IEEE 27th annual conference, vol.2, pp.1533-1538, 2001.

[26] C.A. Rabbath, M. Abdoune, J. Belanger ''Effective real-time simulations of event-based systems'' Simulation conference proceedings, vol.1, pp.232-238, 2000 winter.

[27] C. Dufour, J. Belanger, S. Abourida "Accurate Simulation of 6-pulse Inverter with Real Time Event Compensation in ARTEMIS," Electrimacs Conf., Aug. 2002, Montreal.

[28] J. Cong, Y. Ding "FlowMap: an optimal technology mapping algorithm for delayoptimization in lookup-table based FPGA designs" IEEE transaction on Computer-Aided Design of ICS, vol.13, issue:1, pp.1-12, Jan 1994.

[29] R.D. Wittig, P. Chow "OneChip: an FPGA processor with reconfigurable logic" IEEE symposium, FPGA for Custom Computing Machines, PP.126-135, April 1996.

Appendix A – Simulation results with RT-LAB workshop

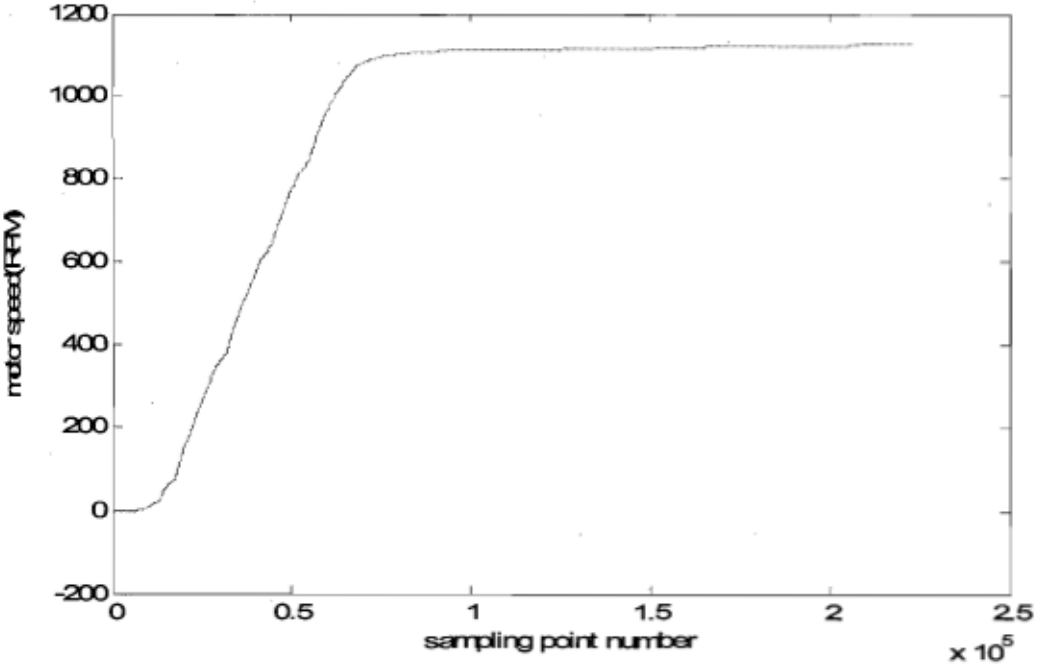


Figure A.1 Motor speed, no-load status and 1100 rpm speed reference signal, Fc=3.6 kHz, Ts=5e-6 s

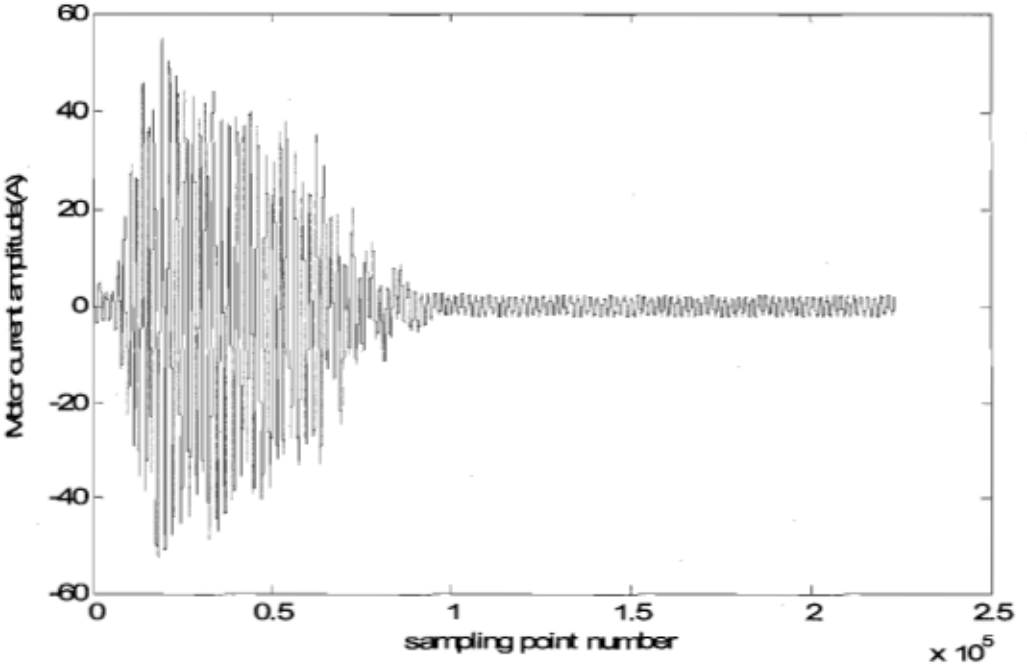


Figure A.2 Motor current of one phase with no-load, Fc=3.6 kHz, Ts=5e-6 s

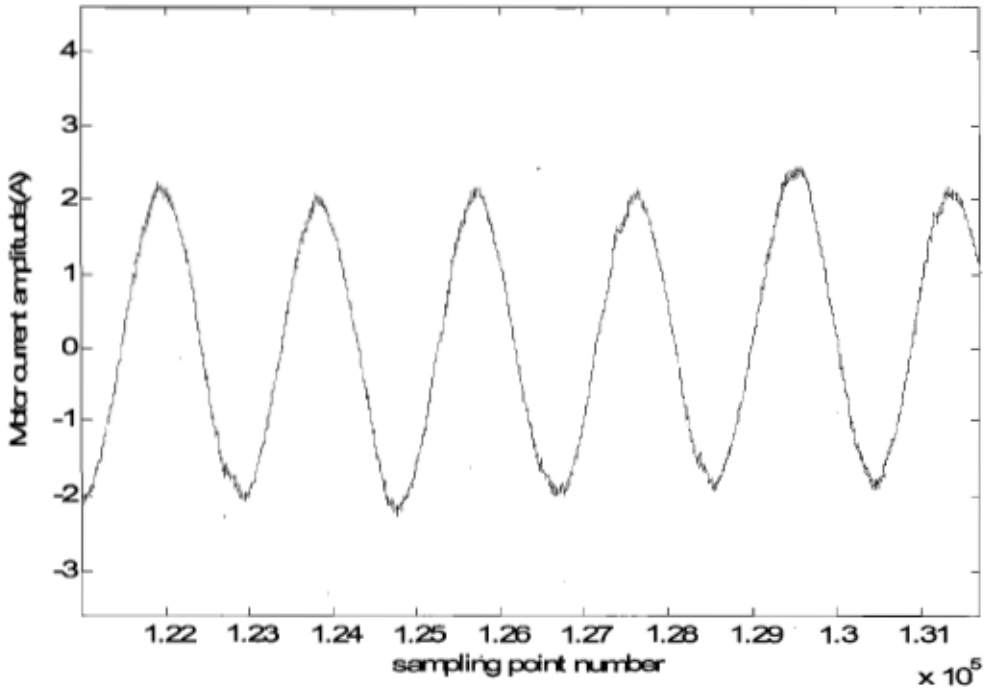


Figure A.3 Motor current of one phase in steady-state regime with no-load , Fc=3.6 kHz, Ts=5e-6 s

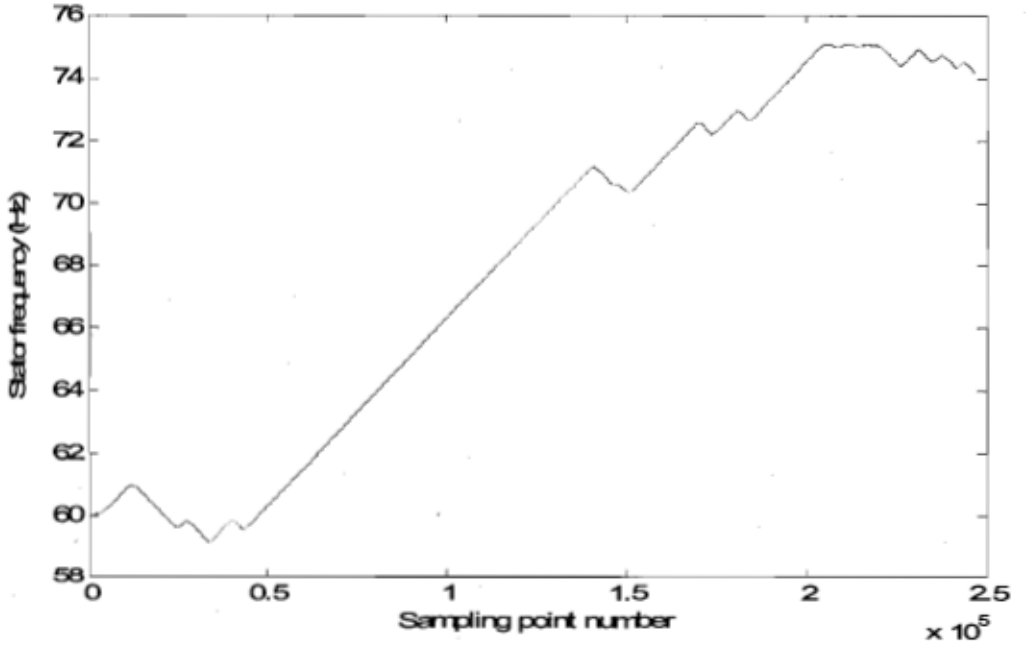


Figure A.4 Motor stator frequency with a non-zero pre-assumed frequency value (60 Hz) at the start time of the simulation, Fc=3.6 kHz, Ts=5e-6 s

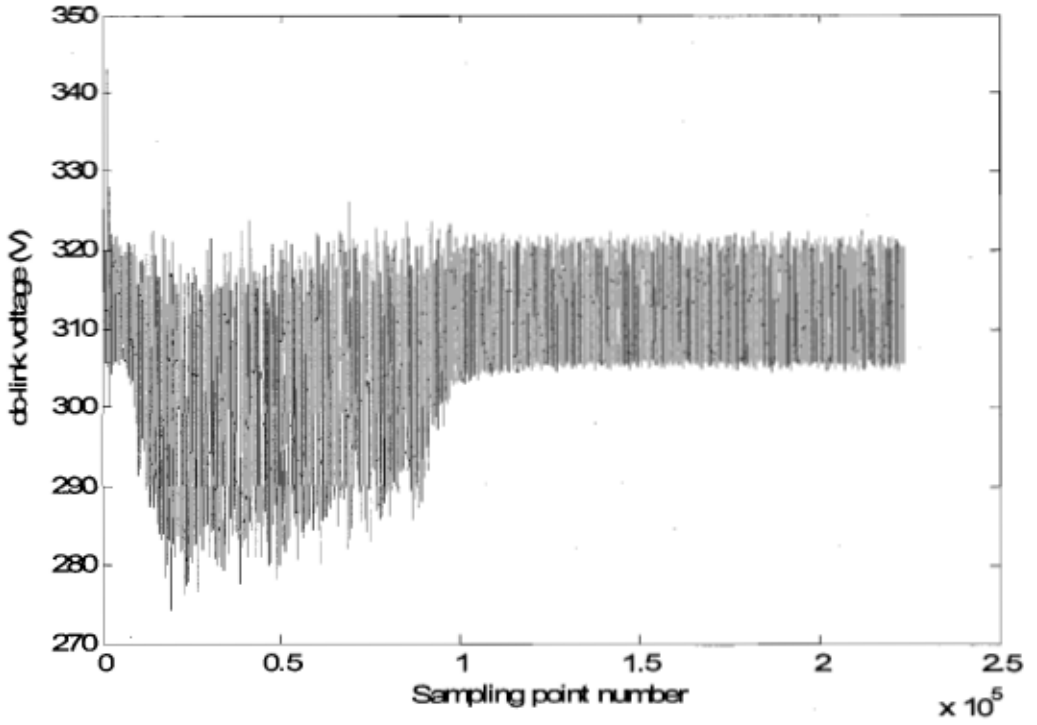


Figure A.5 Voltage of dc-link in simulation with the load condition, Fc=3.6 kHz, Ts=5e-6 s

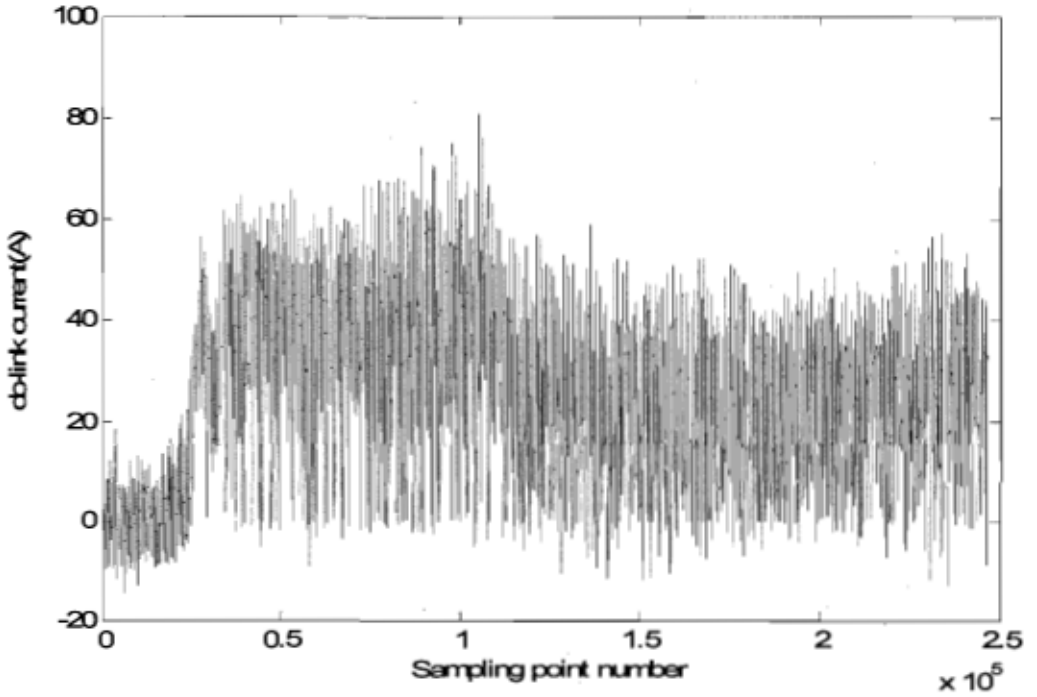


Figure A.6 Current of dc-link in simulation with the load condition, Fc=3.6 kHz, Ts=5e-6 s

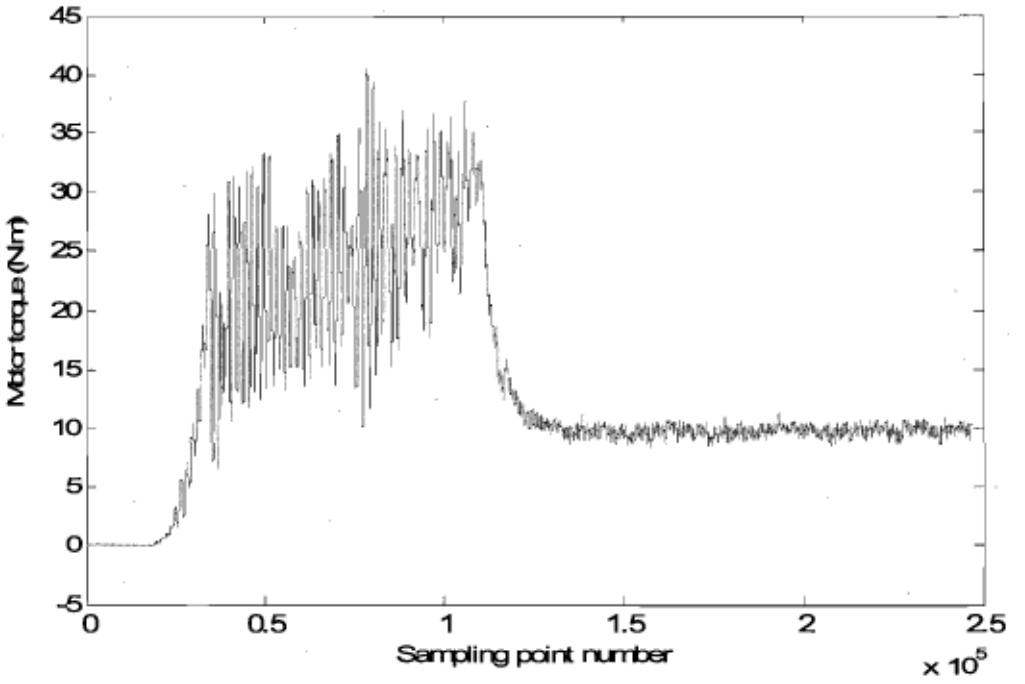


Figure A.7 Motor torque with load condition (10 N·m), Fc=3.6 kHz, Ts=5e-6 s

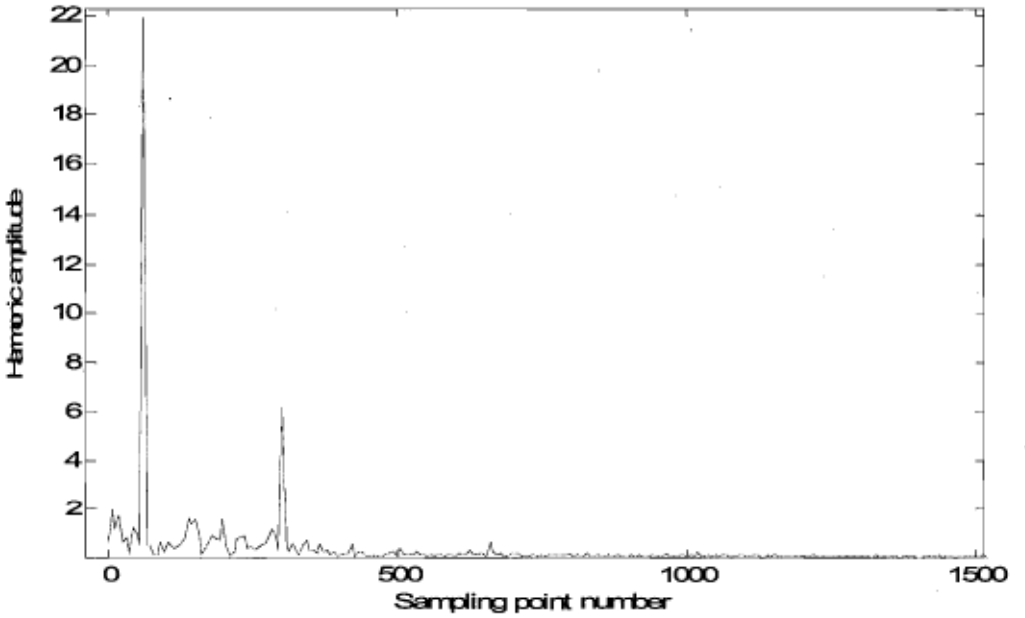


Figure A.8 FFT of input line current in the transient regime with Fc=7.2 kHz, Ts=5e-6 s

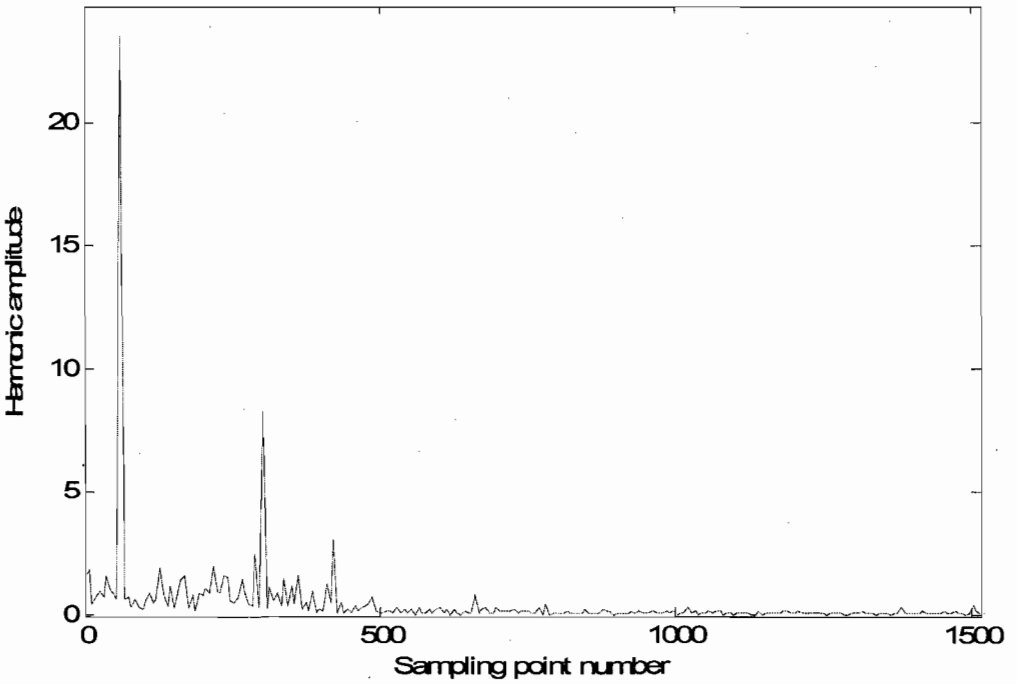


Figure A.9 FFT of input line current in the transient regime with Fc=1.8 kHz, Ts=5e-6 s

Appendix B – RT-LAB Compilation

```
------------------- Starting compilation -------------------
Start at : Saturday, November 11, 2006, 23:46:11

The current RT-LAB version is:  v8.0.3
The current model is:          C:\Travail\OMID2006\recherches21_modified1.mdl
The current host platform is:   NT/2000/XP
The current target platform is: QNX 6.x
WARNING: Python interpreter is not found on the command station.
WARNING: Diagnostic, report generation and user script calls will be disabled.


Preparing original model for code separation and generation...

The current Matlab version is:  v7.0.1


Diagnostics:

Reported by: RT-LAB
Source     : recherches21_modified1

SimPowerSystems processing network #1 of recherches21_modified1 ...
Computing state-space representation of linear electrical circuit ... (10 states ; 26 inputs ; 50
outputs)
Computing discrete-time domain model of linear part of network (Ts=5e-006) ...
Computing steady-state values of currents and voltages ...
Building the Simulink model inside "SM_PS/Redresseur/Voltage Measurement" block ...
Ready.
ARTEMIS executing for sensor named recherches21_modified1/SM_PS/Redresseur/Voltage
Measurement
ARTEMIS discretisation made with 5 us sampling time
Registering OpWriteFile group 26: static filename. The block writes.
Registering OpWriteFile group 26: static filename. The block writes.
Registering OpWriteFile group 26: static filename. The block writes.
Registering OpWriteFile group 26: static filename. The block writes.
Registering OpWriteFile group 26: static filename. The block writes.
Registering OpWriteFile group 26: static filename. The block writes.
Registering OpWriteFile group 26: static filename. The block writes.
Registering OpWriteFile group 26: static filename. The block writes.
Registering OpWriteFile group 26: static filename. The block writes.
Registering OpWriteFile group 26: static filename. The block writes.
Registering OpWriteFile group 26: static filename. The block writes.
Registering OpWriteFile group 26: static filename. The block writes.
Registering OpWriteFile group 26: static filename. The block writes.
Registering OpWriteFile group 26: static filename. The block writes.
Registering OpWriteFile group 26: static filename. The block writes.
Registering OpWriteFile group 26: static filename. The block writes.
Registering OpWriteFile group 26: static filename. The block writes.
Registering OpWriteFile group 26: static filename. The block writes.
Registering OpWriteFile group 26: static filename. The block writes.
Registering OpWriteFile group 26: static filename. The block writes.
```

```
Registering OpWriteFile group 26: static filename. The block writes.
Registering OpWriteFile group 26: static filename. The block writes.
Registering OpWriteFile group 26: static filename. The block writes.
Registering OpWriteFile group 26: static filename. The block writes.
Registering OpWriteFile group 26: static filename. The block writes.
Registering OpWriteFile group 26: static filename. The block writes.
Separating RT-LAB subsystem 'SC_console'...
Separating RT-LAB subsystem 'SM_PS'...
Separating RT-LAB subsystem 'SS_control'...

Model preparation and separation duration : 00h:03m:07s
---------- Completed successfully ----------

---------- Generating C code ----------

Using System Target File (TLC file) : rtlab.tlc...

Using Template Makefile (TMF file) : rtlab.tmf...

---------- Generating recherches21_modifie_1_sm_ps C code ----------
Calling RTW Make Command make_rtw...

### Starting Real-Time Workshop build procedure for model: recherches21_modifie_1_sm_ps
Note: An obsolete rtw_info_hook file "rtlab_rtw_info_hook" is found.  Information in this file is not
used in this code generation.
### Generating code into build directory:
C:\Travail\OMID2006\recherches21_modifie_1_sm_ps_rtlab

SimPowerSystems processing network #1 of recherches21_modifie_1_sm_ps ...
Computing state-space representation of linear electrical circuit ... (10 states ; 26 inputs ; 50
outputs)
Computing discrete-time domain model of linear part of network (Ts=5e-006) ...
Computing steady-state values of currents and voltages ...
Building the Simulink model inside "SM_PS/Onduleur1/Voltage Measurement" block ...
Ready.
ARTEMIS executing for sensor named
recherches21_modifie_1_sm_ps/SM_PS/Onduleur1/Voltage Measurement
ARTEMIS discretisation made with 5 us sampling time
### Invoking Target Language Compiler on recherches21_modifie_1_sm_ps.rtw
        tlc
        -r
        C:\Travail\OMID2006\recherches21_modifie_1_sm_ps_rtlab\recherches21_modifie_1_s
m_ps.rtw
        C:\OPAL-RT\RT-LAB8.0.3\Simulink\rtw\c\common\rtlab.tlc
        -OC:\Travail\OMID2006\recherches21_modifie_1_sm_ps_rtlab
        -aReleaseVersion=14.1
        -IC:\OPAL-RT\RT-LAB8.0.3\Simulink\rtw\c\common
        -Ic:\matlab701\toolbox\physmod\powersys\powersys_artemis_4.0a1\tlc_c
        -IC:\Travail\OMID2006\recherches21_modifie_1_sm_ps_rtlab\tlc
        -Ic:\matlab701\rtw\c\tlc\mw
        -Ic:\matlab701\rtw\c\tlc\lib
        -Ic:\matlab701\rtw\c\tlc\blocks
        -Ic:\matlab701\rtw\c\tlc\fixpt
        -Ic:\matlab701\stateflow\c\tlc
        -aEnforceIntegerDowncast=1
        -aFoldNonRolledExpr=1
```

```
-aInlineInvariantSignals=0
-aInlineParameters=0
-aLocalBlockOutputs=0
-aRollThreshold=5
-aZeroInternalMemoryAtStartup=1
-aZeroExternalMemoryAtStartup=1
-aInitFltsAndDblsToZero=1
-aGenerateReport=0
-aGenCodeOnly=1
-aRTWVerbose=1
-aIncludeHyperlinkInReport=0
-aLaunchReport=0
-aForceParamTrailComments=0
-aGenerateComments=1
-aIgnoreCustomStorageClasses=1
-aIncHierarchyInIds=0
-aMaxRTWIdLen=31
-aShowEliminatedStatements=0
-aPrefixModelToSubsysFcnNames=1
-aIncDataTypeInIds=0
-aInsertBlockDesc=0
-aSimulinkBlockComments=1
-aInlinedPrmAccess="Literals"
-aTargetFcnLib="ansi_tfl_tmw.mat"
-aIsPILTarget=0
-aLogVarNameModifier="rt_"
-aGenerateFullHeader=1
-aInitFltsAndDblsToZero=1
-p10000

### Loading TLC function libraries

.....
### Initial pass through model to cache user defined code
.....
### Caching model source code
.....
.....
.....
.....
.....
### Writing header file recherches21_modifie_1_sm_ps_types.h
### Writing header file recherches21_modifie_1_sm_ps.h
### Writing source file recherches21_modifie_1_sm_ps.c
### Writing header file recherches21_modifie_1_sm_ps_private.h
.
### Writing source file recherches21_modifie_1_sm_ps_data.c
### Writing header file rt_sfcn_helper.h
### Writing source file rt_sfcn_helper.c
### Writing header file rt_nonfinite.h
.
### Writing source file rt_nonfinite.c
### TLC code generation complete.
### Creating model mapping file recherches21_modifie_1_sm_ps.map using map_r14.tlc
.....
```

```
.....
.....### Creating project marker file: rtw_proj.tmw
### Creating recherches21_modifie_1_sm_ps.mk from C:\OPAL-RT\RT-
LAB8.0.3\Simulink\rtw\c\common\rtlab.tmf
### Wrapping unrecognized make command (angle brackets added)
###    <make>
### in default batch file

### Successful completion of Real-Time Workshop build procedure for model:
recherches21_modifie_1_sm_ps

recherches21_modifie_1_sm_ps : Generating C code duration : 00h:01m:59s
---------- Completed successfully ----------

---------- Generating recherches21_modifie_3_ss_control C code ----------
Calling RTW Make Command make_rtw...

### Starting Real-Time Workshop build procedure for model:
recherches21_modifie_3_ss_control
Note: An obsolete rtw_info_hook file "rtlab_rtw_info_hook" is found.  Information in this file is not
used in this code generation.
### Generating code into build directory:
C:\Travail\OMID2006\recherches21_modifie_3_ss_control_rtlab
Stateflow parsing for model "recherches21_modifie_3_ss_control"...Done
Stateflow code generation for model "recherches21_modifie_3_ss_control".....Done
### Invoking Target Language Compiler on recherches21_modifie_3_ss_control.rtw
        tlc
        -r
        C:\Travail\OMID2006\recherches21_modifie_3_ss_control_rtlab\recherches21_modifie_3
_ss_control.rtw
        C:\OPAL-RT\RT-LAB8.0.3\Simulink\rtw\c\common\rtlab.tlc
        -OC:\Travail\OMID2006\recherches21_modifie_3_ss_control_rtlab
        -aReleaseVersion=14.1
        -IC:\OPAL-RT\RT-LAB8.0.3\Simulink\rtw\c\common
        -Ic:\matlab701\toolbox\simulink\blocks\tlc_c
        -Ic:\matlab701\toolbox\physmod\powersys\powersys_artemis_4.0a1\tlc_c
        -IC:\Travail\OMID2006\recherches21_modifie_3_ss_control_rtlab\tlc
        -Ic:\matlab701\rtw\c\tlc\mw
        -Ic:\matlab701\rtw\c\tlc\lib
        -Ic:\matlab701\rtw\c\tlc\blocks
        -Ic:\matlab701\rtw\c\tlc\fixpt
        -Ic:\matlab701\stateflow\c\tlc
        -aEnforceIntegerDowncast=1
        -aFoldNonRolledExpr=1
        -aInlineInvariantSignals=0
        -aInlineParameters=0
        -aLocalBlockOutputs=0
        -aRollThreshold=5
        -aZeroInternalMemoryAtStartup=1
        -aZeroExternalMemoryAtStartup=1
        -aInitFltsAndDblsToZero=1
        -aGenerateReport=0
        -aGenCodeOnly=1
        -aRTWVerbose=1
        -aIncludeHyperlinkInReport=0
        -aLaunchReport=0
```

```
-aForceParamTrailComments=0
-aGenerateComments=1
-aIgnoreCustomStorageClasses=1
-aIncHierarchyInIds=0
-aMaxRTWIdLen=31
-aShowEliminatedStatements=0
-aPrefixModelToSubsysFcnNames=1
-aIncDataTypeInIds=0
-aInsertBlockDesc=0
-aSimulinkBlockComments=1
-aInlinedPrmAccess="Literals"
-aTargetFcnLib="ansi_tfl_tmw.mat"
-aIsPILTarget=0
-aLogVarNameModifier="rt_"
-aGenerateFullHeader=1
-aInitFltsAndDblsToZero=1
-p10000

### Loading TLC function libraries

.....
### Initial pass through model to cache user defined code
.....
### Caching model source code
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
### Writing header file recherches21_modifie_3_ss_control_types.h
### Writing header file recherches21_modifie_3_ss_control.h
.
### Writing header file recherches21_modifie_3_ss_control_private.h
### Writing source file recherches21_modifie_3_ss_control.c
### Writing source file recherches21_modifie_3_ss_control_data.c
### Writing header file rt_sfcn_helper.h
.
### Writing source file rt_sfcn_helper.c
### Writing header file rt_nonfinite.h
### Writing source file rt_nonfinite.c
### TLC code generation complete.
### Creating model mapping file recherches21_modifie_3_ss_control.map using map_r14.tlc
.....
.....### Creating project marker file: rtw_proj.tmw
### Creating recherches21_modifie_3_ss_control.mk from C:\OPAL-RT\RT-
LAB8.0.3\Simulink\rtw\c\common\rtlab.tmf
### Wrapping unrecognized make command (angle brackets added)
###    <make>
### in default batch file
```

Successful completion of Real-Time Workshop build procedure for model: recherches21_modifie_3_ss_control

recherches21_modifie_3_ss_control : Generating C code duration : 00h:03m:53s
---------- Completed successfully ----------

---------- Creating the parameter database ----------

Parameter(s) with more than 20 values will be disabled.
Use PARAM_VECTOR_SIZE_LIMIT environment variable to modify this limit.
---------- Parameter database created successfully ----------

---------- Creating the signals database ----------

Signal(s) with more than 20 values will be disabled.
Use SIGNALS_VECTOR_SIZE_LIMIT environment variable to modify this limit.
---------- Signal database created successfully ----------

---------- Transferring the generated C code ----------

Connecting to 192.168.0.8 ... OK.

Setting remote directory to /home/cpeepc14/c/travail/omid2006/recherches21_modified1_sm_ps/... OK.
Transferring C:\Travail\OMID2006\recherches21_modified1_SM_PS\OpNTOtarget\recherches21_modifie_1_sm_ps.c in ascii mode... OK.
Transferring C:\Travail\OMID2006\recherches21_modified1_SM_PS\OpNTOtarget\recherches21_modifie_1_sm_ps_data.c in ascii mode... OK.
Transferring C:\Travail\OMID2006\recherches21_modified1_SM_PS\OpNTOtarget\recherches21_modifie_1_sm_ps.h in ascii mode... OK.
Transferring C:\Travail\OMID2006\recherches21_modified1_SM_PS\OpNTOtarget\recherches21_modifie_1_sm_ps_private.h in ascii mode... OK.
Transferring C:\Travail\OMID2006\recherches21_modified1_SM_PS\OpNTOtarget\recherches21_modifie_1_sm_ps_types.h in ascii mode... OK.
Transferring C:\Travail\OMID2006\recherches21_modified1_SM_PS\OpNTOtarget\recherches21_modifie_1_sm_ps.mk in ascii mode... OK.
Transferring C:\Travail\OMID2006\recherches21_modified1_SM_PS\OpNTOtarget\rt_nonfinite.c in ascii mode... OK.
Transferring C:\Travail\OMID2006\recherches21_modified1_SM_PS\OpNTOtarget\rt_nonfinite.h in ascii mode... OK.
Transferring C:\Travail\OMID2006\recherches21_modified1_SM_PS\OpNTOtarget\rtwtypes.h in ascii mode... OK.
Transferring C:\Travail\OMID2006\recherches21_modified1_SM_PS\OpNTOtarget\rt_sfcn_helper.c in ascii mode... OK.

```
Transferring
C:\Travail\OMID2006\recherches21_modified1_SM_PS\OpNTOtarget\rt_sfcn_helper.h in ascii
mode... OK.
Transferring C:\OPAL-RT\RT-LAB8.0.3\Simulink\rtw\c\common\qnxnto.opt in ascii mode... OK.
Transferring C:\OPAL-RT\RT-LAB8.0.3\Simulink\rtw\c\common\posix.rules in ascii mode... OK.

Setting remote directory to
/home/cpeepc14/c/travail/omid2006/recherches21_modified1_ss_control/... OK.
Transferring
C:\Travail\OMID2006\recherches21_modified1_SS_control\OpNTOtarget\recherches21_modifie_
3_ss_control.c in ascii mode... OK.
Transferring
C:\Travail\OMID2006\recherches21_modified1_SS_control\OpNTOtarget\recherches21_modifie_
3_ss_control_data.c in ascii mode... OK.
Transferring
C:\Travail\OMID2006\recherches21_modified1_SS_control\OpNTOtarget\recherches21_modifie_
3_ss_control.h in ascii mode... OK.
Transferring
C:\Travail\OMID2006\recherches21_modified1_SS_control\OpNTOtarget\recherches21_modifie_
3_ss_control_private.h in ascii mode... OK.
Transferring
C:\Travail\OMID2006\recherches21_modified1_SS_control\OpNTOtarget\recherches21_modifie_
3_ss_control_types.h in ascii mode... OK.
Transferring
C:\Travail\OMID2006\recherches21_modified1_SS_control\OpNTOtarget\recherches21_modifie_
3_ss_control.mk in ascii mode... OK.
Transferring
C:\Travail\OMID2006\recherches21_modified1_SS_control\OpNTOtarget\rt_nonfinite.c in ascii
mode... OK.
Transferring
C:\Travail\OMID2006\recherches21_modified1_SS_control\OpNTOtarget\rt_nonfinite.h in ascii
mode... OK.
Transferring C:\Travail\OMID2006\recherches21_modified1_SS_control\OpNTOtarget\rtwtypes.h
in ascii mode... OK.
Transferring
C:\Travail\OMID2006\recherches21_modified1_SS_control\OpNTOtarget\rt_sfcn_helper.c in ascii
mode... OK.
Transferring
C:\Travail\OMID2006\recherches21_modified1_SS_control\OpNTOtarget\rt_sfcn_helper.h in ascii
mode... OK.
Transferring C:\OPAL-RT\RT-LAB8.0.3\Simulink\rtw\c\common\qnxnto.opt in ascii mode... OK.
Transferring C:\OPAL-RT\RT-LAB8.0.3\Simulink\rtw\c\common\posix.rules in ascii mode... OK.

File transfer duration : 00h:00m:06s
---------- Completed successfully ----------

---------- Building the generated C code ----------

---------- Building recherches21_modifie_1_sm_ps ----------
rm -f recherches21_modifie_1_sm_ps
gcc -c  -O2 -ffast-math -mpentium -malign-loops=2 -malign-jumps=2 -malign-functions=2    -
DMODEL=recherches21_modifie_1_sm_ps -DRT -DNUMST=2 -DTID01EQ=1 -DNCSTATES=6 -
DMULTITASKING=0 -D_SIMULINK -DRTLAB -DOP_MATLABR14         -DUNIX -I. -
I/usr/matlab/v7.0.1/simulink/include -I/usr/matlab/v7.0.1/extern/include -
I/usr/matlab/v7.0.1/rtw/c/src -I/usr/matlab/v7.0.1/rtw/c/libsrc -I/usr/opalrt/v8.0.3/common/include -
I/usr/opalrt/v8.0.3/common/include_target -I/usr/opalrt/v8.0.3/RT-LAB/include -I/usr/opalrt/rt-
```

```
events/v2.4/rte_r14/include/dummy -I/usr/opalrt/rt-
events/v2.4/rte_r14/include/rte_tsb2level_sfcn_rtw -I/usr/opalrt/rt-events/v2.4/rte_r14/include -
I/usr/opalrt/rt-events/v2.4/rte_r14/include/rte_tsb3level_sfcn_rtw
recherches21_modifie_1_sm_ps.c
recherches21_modifie_1_sm_ps.c: In function `recherches21_modifie_1__f':
recherches21_modifie_1_sm_ps.c:1164: warning: type mismatch in implicit declaration for built-in
function `sin'
recherches21_modifie_1_sm_ps.c:1170: warning: type mismatch in implicit declaration for built-in
function `cos'
recherches21_modifie_1_sm_ps.c: In function `MdlOutputs':
recherches21_modifie_1_sm_ps.c:3136: warning: type mismatch in implicit declaration for built-in
function `sqrt'
gcc -c  -O2 -ffast-math -mpentium -malign-loops=2 -malign-jumps=2 -malign-functions=2    -
DMODEL=recherches21_modifie_1_sm_ps -DRT -DNUMST=2 -DTID01EQ=1 -DNCSTATES=6 -
DMULTITASKING=0 -D_SIMULINK -DRTLAB -DOP_MATLABR14        -DUNIX -I. -
I/usr/matlab/v7.0.1/simulink/include -I/usr/matlab/v7.0.1/extern/include -
I/usr/matlab/v7.0.1/rtw/c/src -I/usr/matlab/v7.0.1/rtw/c/libsrc -I/usr/opalrt/v8.0.3/common/include -
I/usr/opalrt/v8.0.3/common/include_target -I/usr/opalrt/v8.0.3/RT-LAB/include -I/usr/opalrt/rt-
events/v2.4/rte_r14/include/dummy -I/usr/opalrt/rt-
events/v2.4/rte_r14/include/rte_tsb2level_sfcn_rtw -I/usr/opalrt/rt-events/v2.4/rte_r14/include -
I/usr/opalrt/rt-events/v2.4/rte_r14/include/rte_tsb3level_sfcn_rtw
recherches21_modifie_1_sm_ps_data.c
gcc -c  -O2 -ffast-math -mpentium -malign-loops=2 -malign-jumps=2 -malign-functions=2    -
DMODEL=recherches21_modifie_1_sm_ps -DRT -DNUMST=2 -DTID01EQ=1 -DNCSTATES=6 -
DMULTITASKING=0 -D_SIMULINK -DRTLAB -DOP_MATLABR14        -DUNIX -I. -
I/usr/matlab/v7.0.1/simulink/include -I/usr/matlab/v7.0.1/extern/include -
I/usr/matlab/v7.0.1/rtw/c/src -I/usr/matlab/v7.0.1/rtw/c/libsrc -I/usr/opalrt/v8.0.3/common/include -
I/usr/opalrt/v8.0.3/common/include_target -I/usr/opalrt/v8.0.3/RT-LAB/include -I/usr/opalrt/rt-
events/v2.4/rte_r14/include/dummy -I/usr/opalrt/rt-
events/v2.4/rte_r14/include/rte_tsb2level_sfcn_rtw -I/usr/opalrt/rt-events/v2.4/rte_r14/include -
I/usr/opalrt/rt-events/v2.4/rte_r14/include/rte_tsb3level_sfcn_rtw  rt_sfcn_helper.c

gcc -c  -O2 -ffast-math -mpentium -malign-loops=2 -malign-jumps=2 -malign-functions=2    -
DMODEL=recherches21_modifie_1_sm_ps -DRT -DNUMST=2 -DTID01EQ=1 -DNCSTATES=6 -
DMULTITASKING=0 -D_SIMULINK -DRTLAB -DOP_MATLABR14        -DUNIX -I. -
I/usr/matlab/v7.0.1/simulink/include -I/usr/matlab/v7.0.1/extern/include -
I/usr/matlab/v7.0.1/rtw/c/src -I/usr/matlab/v7.0.1/rtw/c/libsrc -I/usr/opalrt/v8.0.3/common/include -
I/usr/opalrt/v8.0.3/common/include_target -I/usr/opalrt/v8.0.3/RT-LAB/include -I/usr/opalrt/rt-
events/v2.4/rte_r14/include/dummy -I/usr/opalrt/rt-
events/v2.4/rte_r14/include/rte_tsb2level_sfcn_rtw -I/usr/opalrt/rt-events/v2.4/rte_r14/include -
I/usr/opalrt/rt-events/v2.4/rte_r14/include/rte_tsb3level_sfcn_rtw  /usr/opalrt/v8.0.3/RT-
LAB/lib/model_main.c
gcc -c  -O2 -ffast-math -mpentium -malign-loops=2 -malign-jumps=2 -malign-functions=2    -
DMODEL=recherches21_modifie_1_sm_ps -DRT -DNUMST=2 -DTID01EQ=1 -DNCSTATES=6 -
DMULTITASKING=0 -D_SIMULINK -DRTLAB -DOP_MATLABR14        -DUNIX -I. -
I/usr/matlab/v7.0.1/simulink/include -I/usr/matlab/v7.0.1/extern/include -
I/usr/matlab/v7.0.1/rtw/c/src -I/usr/matlab/v7.0.1/rtw/c/libsrc -I/usr/opalrt/v8.0.3/common/include -
I/usr/opalrt/v8.0.3/common/include_target -I/usr/opalrt/v8.0.3/RT-LAB/include -I/usr/opalrt/rt-
events/v2.4/rte_r14/include/dummy -I/usr/opalrt/rt-
events/v2.4/rte_r14/include/rte_tsb2level_sfcn_rtw -I/usr/opalrt/rt-events/v2.4/rte_r14/include -
I/usr/opalrt/rt-events/v2.4/rte_r14/include/rte_tsb3level_sfcn_rtw
/usr/matlab/v7.0.1/rtw/c/src/rt_sim.c
gcc -c  -O2 -ffast-math -mpentium -malign-loops=2 -malign-jumps=2 -malign-functions=2    -
DMODEL=recherches21_modifie_1_sm_ps -DRT -DNUMST=2 -DTID01EQ=1 -DNCSTATES=6 -
DMULTITASKING=0 -D_SIMULINK -DRTLAB -DOP_MATLABR14        -DUNIX -I. -
I/usr/matlab/v7.0.1/simulink/include -I/usr/matlab/v7.0.1/extern/include -
```

```
I/usr/matlab/v7.0.1/rtw/c/src -I/usr/matlab/v7.0.1/rtw/c/libsrc -I/usr/opalrt/v8.0.3/common/include -
I/usr/opalrt/v8.0.3/common/include_target -I/usr/opalrt/v8.0.3/RT-LAB/include -I/usr/opalrt/rt-
events/v2.4/rte_r14/include/dummy -I/usr/opalrt/rt-
events/v2.4/rte_r14/include/rte_tsb2level_sfcn_rtw -I/usr/opalrt/rt-events/v2.4/rte_r14/include -
I/usr/opalrt/rt-events/v2.4/rte_r14/include/rte_tsb3level_sfcn_rtw
/usr/matlab/v7.0.1/rtw/c/src/rtwlog.c
gcc -c  -O2 -ffast-math -mpentium -malign-loops=2 -malign-jumps=2 -malign-functions=2    -
DMODEL=recherches21_modifie_1_sm_ps -DRT -DNUMST=2 -DTID01EQ=1 -DNCSTATES=6 -
DMULTITASKING=0 -D_SIMULINK -DRTLAB -DOP_MATLABR14         -DUNIX -I. -
I/usr/matlab/v7.0.1/simulink/include -I/usr/matlab/v7.0.1/extern/include -
I/usr/matlab/v7.0.1/rtw/c/src -I/usr/matlab/v7.0.1/rtw/c/libsrc -I/usr/opalrt/v8.0.3/common/include -
I/usr/opalrt/v8.0.3/common/include_target -I/usr/opalrt/v8.0.3/RT-LAB/include -I/usr/opalrt/rt-
events/v2.4/rte_r14/include/dummy -I/usr/opalrt/rt-
events/v2.4/rte_r14/include/rte_tsb2level_sfcn_rtw -I/usr/opalrt/rt-events/v2.4/rte_r14/include -
I/usr/opalrt/rt-events/v2.4/rte_r14/include/rte_tsb3level_sfcn_rtw  rt_nonfinite.c
gcc -c  -O2 -ffast-math -mpentium -malign-loops=2 -malign-jumps=2 -malign-functions=2    -
DMODEL=recherches21_modifie_1_sm_ps -DRT -DNUMST=2 -DTID01EQ=1 -DNCSTATES=6 -
DMULTITASKING=0 -D_SIMULINK -DRTLAB -DOP_MATLABR14         -DUNIX -I. -
I/usr/matlab/v7.0.1/simulink/include -I/usr/matlab/v7.0.1/extern/include -
I/usr/matlab/v7.0.1/rtw/c/src -I/usr/matlab/v7.0.1/rtw/c/libsrc -I/usr/opalrt/v8.0.3/common/include -
I/usr/opalrt/v8.0.3/common/include_target -I/usr/opalrt/v8.0.3/RT-LAB/include -I/usr/opalrt/rt-
events/v2.4/rte_r14/include/dummy -I/usr/opalrt/rt-
events/v2.4/rte_r14/include/rte_tsb2level_sfcn_rtw -I/usr/opalrt/rt-events/v2.4/rte_r14/include -
I/usr/opalrt/rt-events/v2.4/rte_r14/include/rte_tsb3level_sfcn_rtw
/usr/matlab/v7.0.1/rtw/c/src/ode4.c
g++   -L. -L/usr/opalrt/v8.0.3/RT-LAB/lib -L/usr/opalrt/v8.0.3/common/lib -L/x86/usr/lib  -
L/usr/opalrt/rt-events/v2.4/rte_r14/lib  -o recherches21_modifie_1_sm_ps
recherches21_modifie_1_sm_ps.o recherches21_modifie_1_sm_ps_data.o rt_sfcn_helper.o
model_main.o rt_sim.o rtwlog.o rt_nonfinite.o ode4.o
/usr/opalrt/artemis/v4.0a1/art_r14/lib/artemis40a1r14n.a -lRtEvents24R14n -lOpalRTER14  -
lOpalTestR14 -lOpalOhci -lOpalAsyncApiR14 -lOpalSfunR14 -lOpalR14 -lOpalOhci -
lOpalOpSigWire -lOpalR14 -lOpalOPHSDIO64R14 -lOpalSfunR14 -lOpalOpSigWire -lpython2.2 -
lsocket -lm -lcpp -lpccard    /usr/matlab/v7.0.1/rtw/c/libsrc/rtwlibr_qnx6.a
chmod a+x recherches21_modifie_1_sm_ps
### Created executable: recherches21_modifie_1_sm_ps

recherches21_modifie_1_sm_ps : Building subsystem duration : 00h:00m:20s
------------------- Completed successfully --------------------

------------------- Building recherches21_modifie_3_ss_control --------------------
rm -f recherches21_modifie_3_ss_control
gcc -c  -O2 -ffast-math -mpentium -malign-loops=2 -malign-jumps=2 -malign-functions=2    -
DMODEL=recherches21_modifie_3_ss_control -DRT -DNUMST=4 -DTID01EQ=1 -
DNCSTATES=0 -DMULTITASKING=1 -D_SIMULINK -DRTLAB -DOP_MATLABR14       -DUNIX
-I. -I/usr/matlab/v7.0.1/simulink/include -I/usr/matlab/v7.0.1/extern/include -
I/usr/matlab/v7.0.1/rtw/c/src -I/usr/matlab/v7.0.1/rtw/c/libsrc -I/usr/opalrt/v8.0.3/common/include -
I/usr/opalrt/v8.0.3/common/include_target -I/usr/opalrt/v8.0.3/RT-LAB/include -I/usr/opalrt/rt-
events/v2.4/rte_r14/include/dummy -I/usr/opalrt/rt-
events/v2.4/rte_r14/include/rte_reloperator_sfcn_rtw -I/usr/opalrt/rt-events/v2.4/rte_r14/include
recherches21_modifie_3_ss_control.c
recherches21_modifie_3_ss_control.c: In function `MdlOutputs':
recherches21_modifie_3_ss_control.c:6727
recherches21_modifie_3_ss_control.c:6730
gcc -c  -O2 -ffast-math -mpentium -malign-loops=2 -malign-jumps=2 -malign-functions=2    -
DMODEL=recherches21_modifie_3_ss_control -DRT -DNUMST=4 -DTID01EQ=1 -
DNCSTATES=0 -DMULTITASKING=1 -D_SIMULINK -DRTLAB -DOP_MATLABR14      -DUNIX
```

```
-I. -I/usr/matlab/v7.0.1/simulink/include -I/usr/matlab/v7.0.1/extern/include -
I/usr/matlab/v7.0.1/rtw/c/src -I/usr/matlab/v7.0.1/rtw/c/libsrc -I/usr/opalrt/v8.0.3/common/include -
I/usr/opalrt/v8.0.3/common/include_target -I/usr/opalrt/v8.0.3/RT-LAB/include -I/usr/opalrt/rt-
events/v2.4/rte_r14/include/dummy -I/usr/opalrt/rt-
events/v2.4/rte_r14/include/rte_reloperator_sfcn_rtw -I/usr/opalrt/rt-events/v2.4/rte_r14/include
recherches21_modifie_3_ss_control_data.c
gcc -c -O2 -ffast-math -mpentium -malign-loops=2 -malign-jumps=2 -malign-functions=2 -
DMODEL=recherches21_modifie_3_ss_control -DRT -DNUMST=4 -DTID01EQ=1 -
DNCSTATES=0 -DMULTITASKING=1 -D_SIMULINK -DRTLAB -DOP_MATLABR14 -DUNIX
-I. -I/usr/matlab/v7.0.1/simulink/include -I/usr/matlab/v7.0.1/extern/include -
I/usr/matlab/v7.0.1/rtw/c/src -I/usr/matlab/v7.0.1/rtw/c/libsrc -I/usr/opalrt/v8.0.3/common/include -
I/usr/opalrt/v8.0.3/common/include_target -I/usr/opalrt/v8.0.3/RT-LAB/include -I/usr/opalrt/rt-
events/v2.4/rte_r14/include/dummy -I/usr/opalrt/rt-
events/v2.4/rte_r14/include/rte_reloperator_sfcn_rtw -I/usr/opalrt/rt-events/v2.4/rte_r14/include
rt_sfcn_helper.c

gcc -c -O2 -ffast-math -mpentium -malign-loops=2 -malign-jumps=2 -malign-functions=2 -
DMODEL=recherches21_modifie_3_ss_control -DRT -DNUMST=4 -DTID01EQ=1 -
DNCSTATES=0 -DMULTITASKING=1 -D_SIMULINK -DRTLAB -DOP_MATLABR14 -DUNIX
-I. -I/usr/matlab/v7.0.1/simulink/include -I/usr/matlab/v7.0.1/extern/include -
I/usr/matlab/v7.0.1/rtw/c/src -I/usr/matlab/v7.0.1/rtw/c/libsrc -I/usr/opalrt/v8.0.3/common/include -
I/usr/opalrt/v8.0.3/common/include_target -I/usr/opalrt/v8.0.3/RT-LAB/include -I/usr/opalrt/rt-
events/v2.4/rte_r14/include/dummy -I/usr/opalrt/rt-
events/v2.4/rte_r14/include/rte_reloperator_sfcn_rtw -I/usr/opalrt/rt-events/v2.4/rte_r14/include
/usr/opalrt/v8.0.3/RT-LAB/lib/model_main.c
gcc -c -O2 -ffast-math -mpentium -malign-loops=2 -malign-jumps=2 -malign-functions=2 -
DMODEL=recherches21_modifie_3_ss_control -DRT -DNUMST=4 -DTID01EQ=1 -
DNCSTATES=0 -DMULTITASKING=1 -D_SIMULINK -DRTLAB -DOP_MATLABR14 -DUNIX
-I. -I/usr/matlab/v7.0.1/simulink/include -I/usr/matlab/v7.0.1/extern/include -
I/usr/matlab/v7.0.1/rtw/c/src -I/usr/matlab/v7.0.1/rtw/c/libsrc -I/usr/opalrt/v8.0.3/common/include -
I/usr/opalrt/v8.0.3/common/include_target -I/usr/opalrt/v8.0.3/RT-LAB/include -I/usr/opalrt/rt-
events/v2.4/rte_r14/include/dummy -I/usr/opalrt/rt-
events/v2.4/rte_r14/include/rte_reloperator_sfcn_rtw -I/usr/opalrt/rt-events/v2.4/rte_r14/include
/usr/matlab/v7.0.1/rtw/c/src/rt_sim.c
gcc -c -O2 -ffast-math -mpentium -malign-loops=2 -malign-jumps=2 -malign-functions=2 -
DMODEL=recherches21_modifie_3_ss_control -DRT -DNUMST=4 -DTID01EQ=1 -
DNCSTATES=0 -DMULTITASKING=1 -D_SIMULINK -DRTLAB -DOP_MATLABR14 -DUNIX
-I. -I/usr/matlab/v7.0.1/simulink/include -I/usr/matlab/v7.0.1/extern/include -
I/usr/matlab/v7.0.1/rtw/c/src -I/usr/matlab/v7.0.1/rtw/c/libsrc -I/usr/opalrt/v8.0.3/common/include -
I/usr/opalrt/v8.0.3/common/include_target -I/usr/opalrt/v8.0.3/RT-LAB/include -I/usr/opalrt/rt-
events/v2.4/rte_r14/include/dummy -I/usr/opalrt/rt-
events/v2.4/rte_r14/include/rte_reloperator_sfcn_rtw -I/usr/opalrt/rt-events/v2.4/rte_r14/include
/usr/matlab/v7.0.1/rtw/c/src/rtwlog.c
gcc -c -O2 -ffast-math -mpentium -malign-loops=2 -malign-jumps=2 -malign-functions=2 -
DMODEL=recherches21_modifie_3_ss_control -DRT -DNUMST=4 -DTID01EQ=1 -
DNCSTATES=0 -DMULTITASKING=1 -D_SIMULINK -DRTLAB -DOP_MATLABR14 -DUNIX
-I. -I/usr/matlab/v7.0.1/simulink/include -I/usr/matlab/v7.0.1/extern/include -
I/usr/matlab/v7.0.1/rtw/c/src -I/usr/matlab/v7.0.1/rtw/c/libsrc -I/usr/opalrt/v8.0.3/common/include -
I/usr/opalrt/v8.0.3/common/include_target -I/usr/opalrt/v8.0.3/RT-LAB/include -I/usr/opalrt/rt-
events/v2.4/rte_r14/include/dummy -I/usr/opalrt/rt-
events/v2.4/rte_r14/include/rte_reloperator_sfcn_rtw -I/usr/opalrt/rt-events/v2.4/rte_r14/include
rt_nonfinite.c
g++ -L. -L/usr/opalrt/v8.0.3/RT-LAB/lib -L/usr/opalrt/v8.0.3/common/lib -L/x86/usr/lib -
L/usr/opalrt/rt-events/v2.4/rte_r14/lib -o recherches21_modifie_3_ss_control
recherches21_modifie_3_ss_control.o recherches21_modifie_3_ss_control_data.o
rt_sfcn_helper.o model_main.o rt_sim.o rtwlog.o rt_nonfinite.o
```

```
/usr/opalrt/artemis/v4.0a1/art_r14/lib/artemis40a1r14n.a -lRtEvents24R14n -lOpalRTER14  -
lOpalTestR14 -lOpalOhci -lOpalAsyncApiR14 -lOpalSfunR14 -lOpalR14 -lOpalOhci -
lOpalOpSigWire -lOpalR14 -lOpalOPHSDIO64R14 -lOpalSfunR14 -lOpalOpSigWire -lpython2.2 -
lsocket -lm -lcpp -lpccard    /usr/matlab/v7.0.1/rtw/c/libsrc/rtwlibr_qnx6.a
chmod a+x recherches21_modifie_3_ss_control
### Created executable: recherches21_modifie_3_ss_control

recherches21_modifie_3_ss_control : Building subsystem duration : 00h:00m:27s
------------------- Completed successfully --------------------


-------------------- Transferring the built model ---------------------

Connecting to 192.168.0.8 ... OK.

Setting remote directory to
/home/cpeepc14/c/travail/omid2006/recherches21_modified1_sm_ps/... OK.
Transferring recherches21_modifie_1_sm_ps in binary mode... OK.

Setting remote directory to
/home/cpeepc14/c/travail/omid2006/recherches21_modified1_ss_control/... OK.
Transferring recherches21_modifie_3_ss_control in binary mode... OK.

File transfer duration : 00h:00m:02s
------------------- Completed successfully --------------------

End at : Saturday, November 11, 2006, 23:56:25

Compilation duration : 00h:10m:14s

Node Transfer and Execution
---------- Transferring files ... ----------
Setting remote directory to
/home/cpeepc14/c/travail/omid2006/recherches21_modified1_sm_ps/... OK.
Transferring
C:\Travail\OMID2006\recherches21_modified1_SM_PS\OpNTOtarget\recherches21_modifie_1_s
m_ps in binary mode... OK.

---------- Done transferring files ----------


        Model 'recherches21_modifie_1_sm_ps' compiled in RELEASE mode.
        QNX version: 601
        2 CPUs active on this Computer
        libOpalR14.a : v8.0.3 (build = 20060726155255)
        A Unit delay is applied on status exchange.
        Monitoring is enabled
        RECV: connection to host established
        SEND: connection to host established
        Display of standard output will be disabled
        Monitoring: start time = 0.000 ms, using CPU speed = 501 MHz
        SubSystem step size = 0.000005 sec. Status updated at every 1 local step.

        Real-time SingleTasking mode.
        Snapshot taken (oprecherches21_modifie_sm_ps_0.snap).
        [0]: PAUSE mode, IO set to pause value.
```

```
        Total of 0 Overrun detected.
        Sun Jun  8 09:14:11 1997

[1]: RUN mode, IO set to run value.
        Synchronized step size = 5 us.
        Sun Jun  8 09:15:51 1997

Main priority set to 63
[1105353]: PAUSE mode, IO set to pause value.
        Total of 1105342 Overruns detected.
        Sun Jun  8 09:20:58 1997

[1105353]: Reset
        Total of 1105342 Overruns detected.
        Sun Jun  8 09:21:01 1997
```

Appendix C – VHDL files and M-files used for two-stage DPEC in XSG simulation

VHDL file of inversion stage:

```
-- F:\ONDSTAT1.vhd
-- VHDL code created by Xilinx's StateCAD 8.2i
-- Fri Nov 24 17:41:26 2006

-- This VHDL code (for use with Synopsys) was generated using:
-- enumerated state assignment with structured code format.
-- Minimization is enabled, implied else is enabled,
-- and outputs are speed optimized.

LIBRARY ieee;
USE ieee.std_logic_1164.all;

LIBRARY synopsys;
USE synopsys.attributes.all;

ENTITY ONDSTAT1 IS
        PORT (clk,RESET,ce,s1,s2,s3,s4,s5,s6,t1,t2,t3,t4,t5: IN std_logic;
                a1,a2,b1,b2,c1,c2 : OUT std_logic);

END;

ARCHITECTURE BEHAVIOR OF ONDSTAT1 IS
        TYPE type_sreg IS
(STATE0,STATE1,STATE2,STATE3,STATE4,STATE5,STATE6,STATE7,
        STATE8,STATE9,STATE10,STATE11,STATE12,STATE13,STATE14,STATE15,STATE1
6,
STATE17,STATE18,STATE19,STATE20,STATE21,STATE22,STATE23,STATE24,STATE25,
STATE26,STATE27,STATE28,STATE29);
        SIGNAL sreg, next_sreg : type_sreg;
        SIGNAL next_a1,next_a2,next_b1,next_b2,next_c1,next_c2 : std_logic;
BEGIN
        PROCESS (clk,ce)
        BEGIN
                IF ce='1' AND clk='1' AND clk'event THEN
                        IF ( RESET='1') THEN
                                sreg <= STATE0;
                        ELSE
                                sreg <= next_sreg;
                        END IF;
                END IF;
        END PROCESS;

        PROCESS (clk,ce)
        BEGIN
```

```
        IF ce='1' AND clk='1' AND clk'event THEN
                IF ( RESET='1') THEN
                        a1 <= '1';
                        ELSE
                        a1 <= next_a1;
                END IF;
        END IF;
END PROCESS;

PROCESS (clk,ce)
BEGIN
        IF ce='1' AND clk='1' AND clk'event THEN
                IF ( RESET='1') THEN
                        a2 <= '0';
                        ELSE
                        a2 <= next_a2;
                END IF;
        END IF;
END PROCESS;

PROCESS (clk,ce)
BEGIN
        IF ce='1' AND clk='1' AND clk'event THEN
                IF ( RESET='1') THEN
                        b1 <= '1';
                        ELSE
                        b1 <= next_b1;
                END IF;
        END IF;
END PROCESS;

PROCESS (clk,ce)
BEGIN
        IF ce='1' AND clk='1' AND clk'event THEN
                IF ( RESET='1') THEN
                        b2 <= '0';
                        ELSE
                        b2 <= next_b2;
                END IF;
        END IF;
END PROCESS;

PROCESS (clk,ce)
BEGIN
        IF ce='1' AND clk='1' AND clk'event THEN
                IF ( RESET='1') THEN
                        c1 <= '1';
                        ELSE
                        c1 <= next_c1;
                END IF;
        END IF;
END PROCESS;

PROCESS (clk,ce)
BEGIN
        IF ce='1' AND clk='1' AND clk'event THEN
```

```
                IF ( RESET='1') THEN
                        c2 <= '0';
                        ELSE
                        c2 <= next_c2;
                END IF;
        END IF;
END PROCESS;

PROCESS (sreg,RESET,s1,s2,s3,s4,s5,s6,t1,t2,t3,t4,t5)
BEGIN
        next_a1 <= '0'; next_a2 <= '0'; next_b1 <= '0'; next_b2 <= '0'; next_c1 <=
                '0'; next_c2 <= '0';

        next_sreg<=STATE0;

        IF ( RESET='1') THEN
                next_sreg<=STATE0;
                next_c2<='0';
                next_b2<='0';
                next_a2<='0';
                next_a1<='1';
                next_b1<='1';
                next_c1<='1';
        ELSE
                CASE sreg IS
                        WHEN STATE0 =>
IF ( s1='0' AND s2='0' AND s3='0' AND s4='0' AND s5='0' AND s6='0' ) OR       ( t1='0' AND
s2='0' AND s3='0' AND s4='0' AND s5='0' AND s6='0' ) OR ( t1='0'AND s3='1' AND s2='1' ) OR (
t1='0' AND s4='1' AND s2='1' ) OR ( t1='0' AND s4='1' AND s3='1' ) OR ( t1='0' AND s5='1' AND
s2='1' ) OR ( s1='0' AND
s5='1' AND s2='1' ) OR ( t1='0' AND s5='1' AND s3='1' ) OR ( s1='0' AND s5='1' AND s3='1' ) OR
( t1='0' AND s5='1' AND s4='1' ) OR ( s1='0' AND s5='1' AND s4='1' ) OR ( t1='0' AND s1='1' ) OR
( s1='0' AND s3='1' AND s2='1' ) OR ( s1='0' AND s4='1' AND s2='1' ) OR ( t1='0' AND s6='1'
AND s2='1' ) OR ( s1='0' AND s6='1' AND s2='1' ) OR ( s1='0' AND s4='1' AND s3='1' ) OR (
t1='0' AND s6='1' AND s3='1' ) OR ( s1='0' AND s6='1' AND s3='1' ) OR ( t1='0' AND s6='1' AND
s4='1' ) OR ( s1='0' AND s6='1' AND
s4='1' ) OR ( t1='0' AND s6='1' AND s5='1' ) OR ( s1='0' AND s6='1' AND s5='1' ) THEN
                                        next_sreg<=STATE0;
                                        next_a1<='1';
                                        next_a2<='0';
                                        next_b1<='1';
                                        next_b2<='0';
                                        next_c1<='1';
                                        next_c2<='0';
                                END IF;
                                IF ( t1='1' AND s1='1' ) THEN
                                        next_sreg<=STATE1;
                                        next_a1<='1';
                                        next_a2<='0';
                                        next_b1<='0';
                                        next_b2<='1';
                                        next_c1<='1';
                                        next_c2<='0';
                                END IF;
                        IF ( s2='1' AND s1='0' AND s3='0' AND s4='0' AND s5='0' AND s6='0' )
                                        THEN
```

```
                    next_sreg<=STATE5;
                    next_a1<='1';
                    next_a2<='0';
                    next_b1<='1';
                    next_b2<='0';
                    next_c1<='1';
                    next_c2<='0';
                END IF;
IF ( s3='1' AND s2='0' AND s1='0' AND s4='0' AND s5='0' AND s6='0' )
                    THEN
                    next_sreg<=STATE10;
                    next_a1<='1';
                    next_a2<='0';
                    next_b1<='1';
                    next_b2<='0';
                    next_c1<='1';
                    next_c2<='0';
                END IF;
IF ( s4='1' AND s5='0' AND s6='0' AND s1='0' AND s2='0' AND s3='0' )
                    THEN
                    next_sreg<=STATE15;
                    next_a1<='1';
                    next_a2<='0';
                    next_b1<='1';
                    next_b2<='0';
                    next_c1<='1';
                    next_c2<='0';
                END IF;
IF ( s5='1' AND s6='0' AND s1='0' AND s2='0' AND s3='0' AND s4='0' )
                    THEN
                    next_sreg<=STATE20;
                    next_a1<='1';
                    next_a2<='0';
                    next_b1<='1';
                    next_b2<='0';
                    next_c1<='1';
                    next_c2<='0';
                END IF;
IF ( s6='1' AND s1='0' AND s2='0' AND s3='0' AND s4='0' AND s5='0' )
                    THEN
                    next_sreg<=STATE25;
                    next_a1<='1';
                    next_a2<='0';
                    next_b1<='1';
                    next_b2<='0';
                    next_c1<='1';
                    next_c2<='0';
                END IF;
            WHEN STATE1 =>
                IF ( t2='1' ) THEN
                    next_sreg<=STATE2;
                    next_a1<='1';
                    next_a2<='0';
                    next_b1<='0';
                    next_b2<='1';
                    next_c1<='0';
```

```
                next_c2<='1';
        ELSE
                next_sreg<=STATE1;
                next_a1<='1';
                next_a2<='0';
                next_b1<='0';
                next_b2<='1';
                next_c1<='1';
                next_c2<='0';
        END IF;
WHEN STATE2 =>
        IF ( t3='1' ) THEN
                next_sreg<=STATE3;
                next_a1<='1';
                next_a2<='0';
                next_b1<='0';
                next_b2<='1';
                next_c1<='1';
                next_c2<='0';
        ELSE
                next_sreg<=STATE2;
                next_a1<='1';
                next_a2<='0';
                next_b1<='0';
                next_b2<='1';
                next_c1<='0';
                next_c2<='1';
        END IF;
WHEN STATE3 =>
        IF ( t4='1' ) THEN
                next_sreg<=STATE4;
                next_a1<='1';
                next_a2<='0';
                next_b1<='1';
                next_b2<='0';
                next_c1<='1';
                next_c2<='0';
        ELSE
                next_sreg<=STATE3;
                next_a1<='1';
                next_a2<='0';
                next_b1<='0';
                next_b2<='1';
                next_c1<='1';
                next_c2<='0';
        END IF;
WHEN STATE4 =>
        IF ( t5='1' ) THEN
                next_sreg<=STATE0;
                next_a1<='1';
                next_a2<='0';
                next_b1<='1';
                next_b2<='0';
                next_c1<='1';
                next_c2<='0';
        ELSE
```

```
                                        next_sreg<=STATE4;
                                        next_a1<='1';
                                        next_a2<='0';
                                        next_b1<='1';
                                        next_b2<='0';
                                        next_c1<='1';
                                        next_c2<='0';
                                END IF;
                        WHEN STATE5 =>
IF ( s2='0' AND s3='0' ) OR ( t1='0' AND s3='0' ) OR ( t1='0' AND s2='1') OR ( s2='0' AND s1='1' )
OR ( t1='0' AND s1='1' ) OR ( s2='0' AND s4='1' )OR ( t1='0' AND s4='1' ) OR ( s2='0' AND s5='1'
) OR ( t1='0' AND s5='1' ) OR ( s2='0' AND s6='1' ) OR ( t1='0' AND s6='1' ) THEN
                                        next_sreg<=STATE5;
                                        next_a1<='1';
                                        next_a2<='0';
                                        next_b1<='1';
                                        next_b2<='0';
                                        next_c1<='1';
                                        next_c2<='0';
                                END IF;
                                IF ( t1='1' AND s2='1' ) THEN
                                        next_sreg<=STATE6;
                                        next_a1<='1';
                                        next_a2<='0';
                                        next_b1<='0';
                                        next_b2<='1';
                                        next_c1<='0';
                                        next_c2<='1';
                                END IF;
                        IF ( s3='1' AND s2='0' AND s1='0' AND s4='0' AND s5='0' AND s6='0' )
                                        THEN
                                        next_sreg<=STATE10;
                                        next_a1<='1';
                                        next_a2<='0';
                                        next_b1<='1';
                                        next_b2<='0';
                                        next_c1<='1';
                                        next_c2<='0';
                                END IF;
                        WHEN STATE6 =>
                                IF ( t2='1' ) THEN
                                        next_sreg<=STATE7;
                                        next_a1<='1';
                                        next_a2<='0';
                                        next_b1<='1';
                                        next_b2<='0';
                                        next_c1<='0';
                                        next_c2<='1';
                                 ELSE
                                        next_sreg<=STATE6;
                                        next_a1<='1';
                                        next_a2<='0';
                                        next_b1<='0';
                                        next_b2<='1';
                                        next_c1<='0';
                                        next_c2<='1';
```

```
        END IF;
WHEN STATE7 =>
        IF ( t3='1' ) THEN
                next_sreg<=STATE8;
                next_a1<='1';
                next_a2<='0';
                next_b1<='0';
                next_b2<='1';
                next_c1<='0';
                next_c2<='1';
         ELSE
                next_sreg<=STATE7;
                next_a1<='1';
                next_a2<='0';
                next_b1<='1';
                next_b2<='0';
                next_c1<='0';
                next_c2<='1';
        END IF;
WHEN STATE8 =>
        IF ( t4='1' ) THEN
                next_sreg<=STATE9;
                next_a1<='1';
                next_a2<='0';
                next_b1<='1';
                next_b2<='0';
                next_c1<='1';
                next_c2<='0';
         ELSE
                next_sreg<=STATE8;
                next_a1<='1';
                next_a2<='0';
                next_b1<='0';
                next_b2<='1';
                next_c1<='0';
                next_c2<='1';
        END IF;
WHEN STATE9 =>
        IF ( t5='1' ) THEN
                next_sreg<=STATE5;
                next_a1<='1';
                next_a2<='0';
                next_b1<='1';
                next_b2<='0';
                next_c1<='1';
                next_c2<='0';
         ELSE
                next_sreg<=STATE9;
                next_a1<='1';
                next_a2<='0';
                next_b1<='1';
                next_b2<='0';
                next_c1<='1';
                next_c2<='0';
        END IF;
WHEN STATE10 =>
```

```
IF ( s3='0' AND s4='0' ) OR ( t1='0' AND s4='0' ) OR ( s3='0' AND s5='1' ) OR ( t1='0' AND s5='1' )
OR ( s3='0' AND s6='1' ) OR ( t1='0' AND s6='1' ) OR ( s3='0' AND s1='1' ) OR ( t1='0' AND s1='1'
) OR ( s3='0' AND s2='1' ) OR ( t1='0' AND s2='1' ) OR ( t1='0' AND s3='1' ) THEN
                                        next_sreg<=STATE10;
                                        next_a1<='1';
                                        next_a2<='0';
                                        next_b1<='1';
                                        next_b2<='0';
                                        next_c1<='1';
                                        next_c2<='0';
                                END IF;
                                IF ( t1='1' AND s3='1' ) THEN
                                        next_sreg<=STATE11;
                                        next_a1<='1';
                                        next_a2<='0';
                                        next_b1<='1';
                                        next_b2<='0';
                                        next_c1<='0';
                                        next_c2<='1';
                                END IF;
                        IF ( s4='1' AND s5='0' AND s6='0' AND s1='0' AND s2='0' AND s3='0' )
                                        THEN
                                        next_sreg<=STATE15;
                                        next_a1<='1';
                                        next_a2<='0';
                                        next_b1<='1';
                                        next_b2<='0';
                                        next_c1<='1';
                                        next_c2<='0';
                                END IF;
                        WHEN STATE11 =>
                                IF ( t2='1' ) THEN
                                        next_sreg<=STATE12;
                                        next_a1<='0';
                                        next_a2<='1';
                                        next_b1<='1';
                                        next_b2<='0';
                                        next_c1<='0';
                                        next_c2<='1';
                                 ELSE
                                        next_sreg<=STATE11;
                                        next_a1<='1';
                                        next_a2<='0';
                                        next_b1<='1';
                                        next_b2<='0';
                                        next_c1<='0';
                                        next_c2<='1';
                                END IF;
                        WHEN STATE12 =>
                                IF ( t3='1' ) THEN
                                        next_sreg<=STATE13;
                                        next_a1<='1';
                                        next_a2<='0';
                                        next_b1<='1';
                                        next_b2<='0';
                                        next_c1<='0';
```

```
                next_c2<='1';
            ELSE
                next_sreg<=STATE12;
                next_a1<='0';
                next_a2<='1';
                next_b1<='1';
                next_b2<='0';
                next_c1<='0';
                next_c2<='1';
            END IF;
        WHEN STATE13 =>
            IF ( t4='1' ) THEN
                next_sreg<=STATE14;
                next_a1<='1';
                next_a2<='0';
                next_b1<='1';
                next_b2<='0';
                next_c1<='1';
                next_c2<='0';
            ELSE
                next_sreg<=STATE13;
                next_a1<='1';
                next_a2<='0';
                next_b1<='1';
                next_b2<='0';
                next_c1<='0';
                next_c2<='1';
            END IF;
        WHEN STATE14 =>
            IF ( t5='1' ) THEN
                next_sreg<=STATE10;
                next_a1<='1';
                next_a2<='0';
                next_b1<='1';
                next_b2<='0';
                next_c1<='1';
                next_c2<='0';
            ELSE
                next_sreg<=STATE14;
                next_a1<='1';
                next_a2<='0';
                next_b1<='1';
                next_b2<='0';
                next_c1<='1';
                next_c2<='0';
            END IF;
        WHEN STATE15 =>
IF ( s4='0' AND s5='0' ) OR ( t1='0' AND s5='0' ) OR ( s4='0' AND s6='1' ) OR ( t1='0' AND s6='1' )
OR ( s4='0' AND s1='1' ) OR ( t1='0' AND s1='1' ) OR ( s4='0' AND s2='1' ) OR ( t1='0' AND s2='1'
) OR ( s4='0' AND s3='1' ) OR ( t1='0' AND s3='1' ) OR ( t1='0' AND s4='1' ) THEN
                next_sreg<=STATE15;
                next_a1<='1';
                next_a2<='0';
                next_b1<='1';
                next_b2<='0';
                next_c1<='1';
```

```
                next_c2<='0';
            END IF;
            IF ( t1='1' AND s4='1' ) THEN
                next_sreg<=STATE16;
                next_a1<='0';
                next_a2<='1';
                next_b1<='1';
                next_b2<='0';
                next_c1<='0';
                next_c2<='1';
            END IF;
IF ( s5='1' AND s6='0' AND s1='0' AND s2='0' AND s3='0' AND s4='0' )
                THEN
                next_sreg<=STATE20;
                next_a1<='1';
                next_a2<='0';
                next_b1<='1';
                next_b2<='0';
                next_c1<='1';
                next_c2<='0';
            END IF;
        WHEN STATE16 =>
            IF ( t2='1' ) THEN
                next_sreg<=STATE17;
                next_a1<='0';
                next_a2<='1';
                next_b1<='1';
                next_b2<='0';
                next_c1<='1';
                next_c2<='0';
             ELSE
                next_sreg<=STATE16;
                next_a1<='0';
                next_a2<='1';
                next_b1<='1';
                next_b2<='0';
                next_c1<='0';
                next_c2<='1';
            END IF;
        WHEN STATE17 =>
            IF ( t3='1' ) THEN
                next_sreg<=STATE18;
                next_a1<='0';
                next_a2<='1';
                next_b1<='1';
                next_b2<='0';
                next_c1<='0';
                next_c2<='1';
             ELSE
                next_sreg<=STATE17;
                next_a1<='0';
                next_a2<='1';
                next_b1<='1';
                next_b2<='0';
                next_c1<='1';
                next_c2<='0';
```

```
                END IF;
            WHEN STATE18 =>
                IF ( t4='1' ) THEN
                        next_sreg<=STATE19;
                        next_a1<='1';
                        next_a2<='0';
                        next_b1<='1';
                        next_b2<='0';
                        next_c1<='1';
                        next_c2<='0';
                 ELSE
                        next_sreg<=STATE18;
                        next_a1<='0';
                        next_a2<='1';
                        next_b1<='1';
                        next_b2<='0';
                        next_c1<='0';
                        next_c2<='1';
                END IF;
            WHEN STATE19 =>
                IF ( t5='1' ) THEN
                        next_sreg<=STATE15;
                        next_a1<='1';
                        next_a2<='0';
                        next_b1<='1';
                        next_b2<='0';
                        next_c1<='1';
                        next_c2<='0';
                 ELSE
                        next_sreg<=STATE19;
                        next_a1<='1';
                        next_a2<='0';
                        next_b1<='1';
                        next_b2<='0';
                        next_c1<='1';
                        next_c2<='0';
                END IF;
            WHEN STATE20 =>
IF ( s5='0' AND s6='0' ) OR ( t1='0' AND s6='0' ) OR ( s5='0' AND s1='1' ) OR ( t1='0' AND s1='1' )
OR ( s5='0' AND s2='1' ) OR ( t1='0' AND s2='1' ) OR ( s5='0' AND s3='1' ) OR ( t1='0' AND s3='1'
) OR ( s5='0' AND s4='1' ) OR ( t1='0' AND s4='1' ) OR ( t1='0' AND s5='1' ) THEN
                        next_sreg<=STATE20;
                        next_a1<='1';
                        next_a2<='0';
                        next_b1<='1';
                        next_b2<='0';
                        next_c1<='1';
                        next_c2<='0';
                END IF;
                IF ( t1='1' AND s5='1' ) THEN
                        next_sreg<=STATE21;
                        next_a1<='0';
                        next_a2<='1';
                        next_b1<='1';
                        next_b2<='0';
                        next_c1<='1';
```

```
                    next_c2<='0';
                END IF;
IF ( s6='1' AND s1='0' AND s2='0' AND s3='0' AND s4='0' AND s5='0' )
                    THEN
                    next_sreg<=STATE25;
                    next_a1<='1';
                    next_a2<='0';
                    next_b1<='1';
                    next_b2<='0';
                    next_c1<='1';
                    next_c2<='0';
                END IF;
            WHEN STATE21 =>
                IF ( t2='1' ) THEN
                    next_sreg<=STATE22;
                    next_a1<='0';
                    next_a2<='1';
                    next_b1<='0';
                    next_b2<='1';
                    next_c1<='1';
                    next_c2<='0';
                 ELSE
                    next_sreg<=STATE21;
                    next_a1<='0';
                    next_a2<='1';
                    next_b1<='1';
                    next_b2<='0';
                    next_c1<='1';
                    next_c2<='0';
                END IF;
            WHEN STATE22 =>
                IF ( t3='1' ) THEN
                    next_sreg<=STATE23;
                    next_a1<='0';
                    next_a2<='1';
                    next_b1<='1';
                    next_b2<='0';
                    next_c1<='1';
                    next_c2<='0';
                 ELSE
                    next_sreg<=STATE22;
                    next_a1<='0';
                    next_a2<='1';
                    next_b1<='0';
                    next_b2<='1';
                    next_c1<='1';
                    next_c2<='0';
                END IF;
            WHEN STATE23 =>
                IF ( t4='1' ) THEN
                    next_sreg<=STATE24;
                    next_a1<='1';
                    next_a2<='0';
                    next_b1<='1';
                    next_b2<='0';
                    next_c1<='1';
```

```
                next_c2<='0';
        ELSE
                next_sreg<=STATE23;
                next_a1<='0';
                next_a2<='1';
                next_b1<='1';
                next_b2<='0';
                next_c1<='1';
                next_c2<='0';
        END IF;
WHEN STATE24 =>
        IF ( t5='1' ) THEN
                next_sreg<=STATE20;
                next_a1<='1';
                next_a2<='0';
                next_b1<='1';
                next_b2<='0';
                next_c1<='1';
                next_c2<='0';
        ELSE
                next_sreg<=STATE24;
                next_a1<='1';
                next_a2<='0';
                next_b1<='1';
                next_b2<='0';
                next_c1<='1';
                next_c2<='0';
        END IF;
WHEN STATE25 =>
        IF ( t1='1' AND s6='1' ) THEN
                next_sreg<=STATE26;
                next_a1<='0';
                next_a2<='1';
                next_b1<='0';
                next_b2<='1';
                next_c1<='1';
                next_c2<='0';
        ELSE
                next_sreg<=STATE25;
                next_a1<='1';
                next_a2<='0';
                next_b1<='1';
                next_b2<='0';
                next_c1<='1';
                next_c2<='0';
        END IF;
WHEN STATE26 =>
        IF ( t2='1' ) THEN
                next_sreg<=STATE27;
                next_a1<='1';
                next_a2<='0';
                next_b1<='0';
                next_b2<='1';
                next_c1<='1';
                next_c2<='0';
        ELSE
```

```
                next_sreg<=STATE26;
                next_a1<='0';
                next_a2<='1';
                next_b1<='0';
                next_b2<='1';
                next_c1<='1';
                next_c2<='0';
        END IF;
WHEN STATE27 =>
        IF ( t3='1' ) THEN
                next_sreg<=STATE28;
                next_a1<='0';
                next_a2<='1';
                next_b1<='0';
                next_b2<='1';
                next_c1<='1';
                next_c2<='0';
         ELSE
                next_sreg<=STATE27;
                next_a1<='1';
                next_a2<='0';
                next_b1<='0';
                next_b2<='1';
                next_c1<='1';
                next_c2<='0';
        END IF;
WHEN STATE28 =>
        IF ( t4='1' ) THEN
                next_sreg<=STATE29;
                next_a1<='1';
                next_a2<='0';
                next_b1<='1';
                next_b2<='0';
                next_c1<='1';
                next_c2<='0';
         ELSE
                next_sreg<=STATE28;
                next_a1<='0';
                next_a2<='1';
                next_b1<='0';
                next_b2<='1';
                next_c1<='1';
                next_c2<='0';
        END IF;
WHEN STATE29 =>
        IF ( t5='1' AND s6='1' ) THEN
                next_sreg<=STATE25;
                next_a1<='1';
                next_a2<='0';
                next_b1<='1';
                next_b2<='0';
                next_c1<='1';
                next_c2<='0';
        END IF;
```

```
IF ( s6='0' AND s1='0' ) OR ( t5='0' AND s1='0' ) OR ( s6='0' AND s2='1') OR ( t5='0' AND s2='1' )
OR ( s6='0' AND s3='1' ) OR ( t5='0' AND s3='1' )OR ( s6='0' AND s4='1' ) OR ( t5='0' AND s4='1'
) OR ( s6='0' AND s5='1' ) OR ( t5='0' AND s5='1' ) OR ( t5='0' AND s6='1' ) THEN
                                        next_sreg<=STATE29;
                                        next_a1<='1';
                                        next_a2<='0';
                                        next_b1<='1';
                                        next_b2<='0';
                                        next_c1<='1';
                                        next_c2<='0';
                                    END IF;
                                IF ( s1='1' AND s2='0' AND s3='0' AND s4='0' AND s5='0' AND s6='0' )
                                        THEN
                                        next_sreg<=STATE4;
                                        next_a1<='1';
                                        next_a2<='0';
                                        next_b1<='1';
                                        next_b2<='0';
                                        next_c1<='1';
                                        next_c2<='0';
                                    END IF;
                                WHEN OTHERS =>
                            END CASE;
                        END IF;
                END PROCESS;
END BEHAVIOR;
```

Interface M-file for importing VHDL file of inversion stage:

```
function ONDSTAT1_config(this_block)

  this_block.setTopLevelLanguage('VHDL');

  this_block.setEntityName('ONDSTAT1');

  % System Generator has to assume that your entity  has a combinational feed through;
  %   if it  doesn't, then comment out the following line:
  % this_block.tagAsCombinational;
  this_block.tagAsCombinational;

  this_block.addSimulinkInport('s1');
  this_block.addSimulinkInport('s2');
  this_block.addSimulinkInport('s3');
  this_block.addSimulinkInport('s4');
  this_block.addSimulinkInport('s5');
  this_block.addSimulinkInport('s6');
  this_block.addSimulinkInport('t1');
  this_block.addSimulinkInport('t2');
  this_block.addSimulinkInport('t3');
  this_block.addSimulinkInport('t4');
  this_block.addSimulinkInport('t5');
  this_block.addSimulinkInport('RESET');
```

```
this_block.addSimulinkOutport('a1');
this_block.addSimulinkOutport('a2');
this_block.addSimulinkOutport('b1');
this_block.addSimulinkOutport('b2');
this_block.addSimulinkOutport('c1');
this_block.addSimulinkOutport('c2');


s1 = this_block.port('s1');
s1.setType('UFix_1_0');
s1.useHDLVector(false);
s2 = this_block.port('s2');
s2.setType('UFix_1_0');
s2.useHDLVector(false);
s3 = this_block.port('s3');
s3.setType('UFix_1_0');
s3.useHDLVector(false);
s4 = this_block.port('s4');
s4.setType('UFix_1_0');
s4.useHDLVector(false);
s5 = this_block.port('s5');
s5.setType('UFix_1_0');
s5.useHDLVector(false);
s6 = this_block.port('s6');
s6.setType('UFix_1_0');
s6.useHDLVector(false);
RESET = this_block.port('RESET');
RESET.setType('UFix_1_0');
RESET.useHDLVector(false);

t1 = this_block.port('t1');
t1.setType('Bool');
t1.useHDLVector(false);
t2 = this_block.port('t2');
t2.setType('Bool');
t2.useHDLVector(false);
t3 = this_block.port('t3');
t3.setType('Bool');
t3.useHDLVector(false);
t4 = this_block.port('t4');
t4.setType('Bool');
t4.useHDLVector(false);
t5 = this_block.port('t5');
t5.setType('Bool');
t5.useHDLVector(false);

a1 = this_block.port('a1');
a1.setType('Bool');
a1.useHDLVector(false);

a2 = this_block.port('a2');
a2.setType('Bool');
a2.useHDLVector(false);

b1 = this_block.port('b1');
```

```
b1.setType('Bool');
b1.useHDLVector(false);

b2 = this_block.port('b2');
b2.setType('Bool');
b2.useHDLVector(false);

c1 = this_block.port('c1');
c1.setType('Bool');
c1.useHDLVector(false);

c2 = this_block.port('c2');
c2.setType('Bool');
c2.useHDLVector(false);

 if (this_block.inputTypesKnown)

 if (this_block.port('s1').width ~= 1);
 this_block.setError('Input data type for port "s1" must have width=1.');
 end
 if (this_block.port('s2').width ~= 1);
 this_block.setError('Input data type for port "s2" must have width=1.');
 end
  if (this_block.port('s3').width ~= 1);
  this_block.setError('Input data type for port "s3" must have width=1.');
  end
  if (this_block.port('s4').width ~= 1);
  this_block.setError('Input data type for port "s4" must have width=1.');
  end
  if (this_block.port('s5').width ~= 1);
  this_block.setError('Input data type for port "s5" must have width=1.');
  end
  if (this_block.port('s6').width ~= 1);
  this_block.setError('Input data type for port "s6" must have width=1.');
  end
  if (this_block.port('t1').width ~= 1);
  this_block.setError('Input data type for port "t1" must have width=1.');
  end
  if (this_block.port('t2').width ~= 1);
  this_block.setError('Input data type for port "t2" must have width=1.');
  end
  if (this_block.port('t3').width ~= 1);
  this_block.setError('Input data type for port "t3" must have width=1.');
  end
  if (this_block.port('t4').width ~= 1);
  this_block.setError('Input data type for port "t4" must have width=1.');
  end
  if (this_block.port('t5').width ~= 1);
  this_block.setError('Input data type for port "t5" must have width=1.');
  end
  if (this_block.port('RESET').width ~= 1);
  this_block.setError('Input data type for port "reset" must have width=1.');
  end

  this_block.port('RESET').useHDLVector(false);
```

```
  end  % if(inputTypesKnown)

  % ----------------------------
  if (this_block.inputRatesKnown)

  setup_as_single_rate(this_block,'clk','ce')
  end  % if(inputRatesKnown)

 this_block.addFile('ONDSTAT1.vhd');

return;

 % ------------------------------------------------------------

function setup_as_single_rate(block,clkname,cename)
 inputRates = block.inputRates;
 uniqueInputRates = unique(inputRates);
 if (length(uniqueInputRates)==1 & uniqueInputRates(1)==Inf)
   block.setError('The inputs to this block cannot all be constant.');
   return;
 end
 if (uniqueInputRates(end) == Inf)
    hasConstantInput = true;
    uniqueInputRates = uniqueInputRates(1:end-1);
 end
 if (length(uniqueInputRates) ~= 1)
   block.setError('The inputs to this block must run at a single rate.');
   return;
 end
 theInputRate = uniqueInputRates(1);
 for i = 1:block.numSimulinkOutports
    block.outport(i).setRate(theInputRate);
 end
 block.addClkCEPair(clkname,cename,theInputRate);
 return;

% ------------------------------------------------------------
```

VHDL file of rectification stage:

```
-- F:\STCAD1MO.vhd
-- VHDL code created by Xilinx's StateCAD 8.2i
-- Fri Nov 17 22:04:59 2006

-- This VHDL code (for use with Xilinx XST) was generated using:
-- enumerated state assignment with structured code format.
-- Minimization is enabled,  implied else is enabled,
-- and outputs are speed optimized.

LIBRARY ieee;
USE ieee.std_logic_1164.all;
```

```
ENTITY STCAD1MO IS
        PORT (clk_1,ce_1,reset,s1,s2,s3,s4,s5,s6,t1,t2,t3,t4,t5: IN std_logic;
                a1,a2,a3,a4,b1,b2,b3,b4,c1,c2,c3,c4 : OUT std_logic);
END;

ARCHITECTURE BEHAVIOR OF STCAD1MO IS
        TYPE type_sreg IS
(STATE0,STATE1,STATE2,STATE3,STATE4,STATE5,STATE6,STATE7,
        STATE8,STATE9,STATE10,STATE11,STATE12,STATE13,STATE14,STATE15,STATE1
6,

STATE17,STATE18,STATE19,STATE20,STATE21,STATE22,STATE23,STATE24,STATE25,
         STATE26,STATE27,STATE28,STATE29,STATE30);
        SIGNAL sreg, next_sreg : type_sreg;
        SIGNAL next_a1,next_a2,next_a3,next_a4,next_b1,next_b2,next_b3,next_b4,
                next_c1,next_c2,next_c3,next_c4 : std_logic;
BEGIN
        PROCESS (clk_1,ce_1)
        BEGIN
                IF  clk_1='1' AND clk_1'event THEN
                        IF ( reset='1' AND ce_1='1' ) THEN
                                sreg <= STATE30;
                        ELSIF (ce_1='1') THEN
                                sreg <= next_sreg;
                        END IF;
                END IF;
        END PROCESS;

        PROCESS (clk_1,ce_1)
        BEGIN
                IF  clk_1='1' AND clk_1'event THEN
                        IF ( reset='1' AND ce_1='1' ) THEN
                                a1 <= '0';
                        ELSIF (ce_1='1') THEN
                                a1 <= next_a1;
                        END IF;
                END IF;
        END PROCESS;

        PROCESS (clk_1,ce_1)
        BEGIN
                IF  clk_1='1' AND clk_1'event THEN
                        IF ( reset='1' AND ce_1='1' ) THEN
                                a2 <= '0';
                        ELSIF (ce_1='1') THEN
                                a2 <= next_a2;
                        END IF;
                END IF;
        END PROCESS;

        PROCESS (clk_1,ce_1)
        BEGIN
                IF  clk_1='1' AND clk_1'event THEN
                        IF ( reset='1' AND ce_1='1' ) THEN
                                a3 <= '0';
                        ELSIF (ce_1='1') THEN
```

```
                        a3 <= next_a3;
                END IF;
        END IF;
END PROCESS;

PROCESS (clk_1,ce_1)
BEGIN
        IF  clk_1='1' AND clk_1'event THEN
                IF ( reset='1' AND ce_1='1' ) THEN
                        a4 <= '0';
                ELSIF (ce_1='1') THEN
                        a4 <= next_a4;
                END IF;
        END IF;
END PROCESS;

PROCESS (clk_1,ce_1)
BEGIN
        IF  clk_1='1' AND clk_1'event THEN
                IF ( reset='1' AND ce_1='1' ) THEN
                        b1 <= '0';
                ELSIF (ce_1='1') THEN
                        b1 <= next_b1;
                END IF;
        END IF;
END PROCESS;

PROCESS (clk_1,ce_1)
BEGIN
        IF  clk_1='1' AND clk_1'event THEN
                IF ( reset='1' AND ce_1='1' ) THEN
                        b2 <= '0';
                ELSIF (ce_1='1') THEN
                        b2 <= next_b2;
                END IF;
        END IF;
END PROCESS;

PROCESS (clk_1,ce_1)
BEGIN
        IF  clk_1='1' AND clk_1'event THEN
                IF ( reset='1' AND ce_1='1' ) THEN
                        b3 <= '0';
                ELSIF (ce_1='1') THEN
                        b3 <= next_b3;
                END IF;
        END IF;
END PROCESS;

PROCESS (clk_1,ce_1)
BEGIN
        IF  clk_1='1' AND clk_1'event THEN
                IF ( reset='1' AND ce_1='1' ) THEN
                        b4 <= '0';
                ELSIF (ce_1='1') THEN
                        b4 <= next_b4;
```

```
                END IF;
        END IF;
END PROCESS;

PROCESS (clk_1,ce_1)
BEGIN
        IF  clk_1='1' AND clk_1'event THEN
                IF ( reset='1' AND ce_1='1' ) THEN
                        c1 <= '0';
                ELSIF (ce_1='1') THEN
                        c1 <= next_c1;
                END IF;
        END IF;
END PROCESS;

PROCESS (clk_1,ce_1)
BEGIN
        IF  clk_1='1' AND clk_1'event THEN
                IF ( reset='1' AND ce_1='1' ) THEN
                        c2 <= '0';
                ELSIF (ce_1='1') THEN
                        c2 <= next_c2;
                END IF;
        END IF;
END PROCESS;

PROCESS (clk_1,ce_1)
BEGIN
        IF  clk_1='1' AND clk_1'event THEN
                IF ( reset='1' AND ce_1='1' ) THEN
                        c3 <= '0';
                ELSIF (ce_1='1') THEN
                        c3 <= next_c3;
                END IF;
        END IF;
END PROCESS;

PROCESS (clk_1,ce_1)
BEGIN
        IF  clk_1='1' AND clk_1'event THEN
                IF ( reset='1' AND ce_1='1' ) THEN
                        c4 <= '0';
                ELSIF (ce_1='1') THEN
                        c4 <= next_c4;
                END IF;
        END IF;
END PROCESS;

PROCESS (sreg,reset,ce_1,s1,s2,s3,s4,s5,s6,t1,t2,t3,t4,t5)
BEGIN
        next_a1 <= '0'; next_a2 <= '0'; next_a3 <= '0'; next_a4 <= '0'; next_b1 <= '0';
          next_b2 <= '0'; next_b3 <= '0'; next_b4 <= '0'; next_c1 <= '0'; next_c2  <= '0';
          next_c3 <= '0'; next_c4 <= '0';

        next_sreg<=STATE0;
```

```
IF ( reset='1' AND ce_1='1') THEN
        next_sreg<=STATE30;
        next_a1<='0';
        next_a2<='0';
        next_a3<='0';
        next_a4<='0';
        next_b1<='0';
        next_b2<='0';
        next_b3<='0';
        next_b4<='0';
        next_c1<='0';
        next_c2<='0';
        next_c3<='0';
        next_c4<='0';
ELSE
        CASE sreg IS
                WHEN STATE0 =>
        IF ( s6='1' AND s5='0' AND s4='0' AND s3='0' AND s2='0' AND s1='0' )
                                THEN
                                next_sreg<=STATE5;
                                next_a1<='0';
                                next_a2<='0';
                                next_a3<='0';
                                next_a4<='0';
                                next_b1<='0';
                                next_b2<='0';
                                next_b3<='1';
                                next_b4<='1';
                                next_c1<='1';
                                next_c2<='1';
                                next_c3<='0';
                                next_c4<='0';
                        END IF;
                        IF ( t1='1' AND s5='1' ) THEN
                                next_sreg<=STATE1;
                                next_a1<='0';
                                next_a2<='0';
                                next_a3<='1';
                                next_a4<='1';
                                next_b1<='0';
                                next_b2<='0';
                                next_b3<='0';
                                next_b4<='1';
                                next_c1<='1';
                                next_c2<='1';
                                next_c3<='0';
                                next_c4<='0';
                        END IF;
IF ( s1='1' AND t1='0' ) OR ( s2='1' AND t1='0' ) OR ( s3='1' AND t1='0') OR ( s4='1' AND t1='0' )
OR ( s5='1' AND t1='0' ) OR ( s6='0' AND t1='0' ) OR ( s1='1' AND s5='0' ) OR ( s2='1' AND s5='0'
) OR ( s3='1' AND s5='0' ) OR ( s4='1' AND s5='0' ) OR ( s6='0' AND s5='0' ) THEN
                                next_sreg<=STATE0;
                                next_a1<='0';
                                next_a2<='0';
                                next_a3<='1';
                                next_a4<='1';
```

```
                next_b1<='0';
                next_b2<='0';
                next_b3<='0';
                next_b4<='0';
                next_c1<='1';
                next_c2<='1';
                next_c3<='0';
                next_c4<='0';
        END IF;
WHEN STATE1 =>
        IF ( t2='1' ) THEN
                next_sreg<=STATE2;
                next_a1<='0';
                next_a2<='0';
                next_a3<='1';
                next_a4<='0';
                next_b1<='0';
                next_b2<='0';
                next_b3<='0';
                next_b4<='1';
                next_c1<='1';
                next_c2<='1';
                next_c3<='0';
                next_c4<='0';
        ELSE
                next_sreg<=STATE1;
                next_a1<='0';
                next_a2<='0';
                next_a3<='1';
                next_a4<='1';
                next_b1<='0';
                next_b2<='0';
                next_b3<='0';
                next_b4<='1';
                next_c1<='1';
                next_c2<='1';
                next_c3<='0';
                next_c4<='0';
        END IF;
WHEN STATE2 =>
        IF ( t3='1' ) THEN
                next_sreg<=STATE3;
                next_a1<='0';
                next_a2<='0';
                next_a3<='1';
                next_a4<='0';
                next_b1<='0';
                next_b2<='0';
                next_b3<='1';
                next_b4<='1';
                next_c1<='1';
                next_c2<='1';
                next_c3<='0';
                next_c4<='0';
        ELSE
                next_sreg<=STATE2;
```

```
                next_a1<='0';
                next_a2<='0';
                next_a3<='1';
                next_a4<='0';
                next_b1<='0';
                next_b2<='0';
                next_b3<='0';
                next_b4<='1';
                next_c1<='1';
                next_c2<='1';
                next_c3<='0';
                next_c4<='0';
        END IF;
WHEN STATE3 =>
        IF ( t4='1' ) THEN
                next_sreg<=STATE4;
                next_a1<='0';
                next_a2<='0';
                next_a3<='0';
                next_a4<='0';
                next_b1<='0';
                next_b2<='0';
                next_b3<='1';
                next_b4<='1';
                next_c1<='1';
                next_c2<='1';
                next_c3<='0';
                next_c4<='0';
         ELSE
                next_sreg<=STATE3;
                next_a1<='0';
                next_a2<='0';
                next_a3<='1';
                next_a4<='0';
                next_b1<='0';
                next_b2<='0';
                next_b3<='1';
                next_b4<='1';
                next_c1<='1';
                next_c2<='1';
                next_c3<='0';
                next_c4<='0';
        END IF;
WHEN STATE4 =>
        IF ( t5='1' ) THEN
                next_sreg<=STATE0;
                next_a1<='0';
                next_a2<='0';
                next_a3<='1';
                next_a4<='1';
                next_b1<='0';
                next_b2<='0';
                next_b3<='0';
                next_b4<='0';
                next_c1<='1';
                next_c2<='1';
```

```
                next_c3<='0';
                next_c4<='0';
            ELSE
                next_sreg<=STATE4;
                next_a1<='0';
                next_a2<='0';
                next_a3<='0';
                next_a4<='0';
                next_b1<='0';
                next_b2<='0';
                next_b3<='0';
                next_b4<='1';
                next_c1<='1';
                next_c2<='1';
                next_c3<='0';
                next_c4<='0';
            END IF;
        WHEN STATE5 =>
    IF ( s1='1' AND s2='0' AND s3='0' AND s4='0' AND s5='0' AND s6='0' )
                THEN
                next_sreg<=STATE10;
                next_a1<='1';
                next_a2<='1';
                next_a3<='0';
                next_a4<='0';
                next_b1<='0';
                next_b2<='0';
                next_b3<='1';
                next_b4<='1';
                next_c1<='0';
                next_c2<='0';
                next_c3<='0';
                next_c4<='0';
            END IF;
            IF ( t1='1' AND s6='1' ) THEN
                next_sreg<=STATE6;
                next_a1<='0';
                next_a2<='1';
                next_a3<='0';
                next_a4<='0';
                next_b1<='0';
                next_b2<='0';
                next_b3<='1';
                next_b4<='1';
                next_c1<='1';
                next_c2<='1';
                next_c3<='0';
                next_c4<='0';
            END IF;
IF ( s6='1' AND t1='0' ) OR ( s5='1' AND t1='0' ) OR ( s4='1' AND t1='0' ) OR ( s3='1' AND t1='0' )
OR ( s2='1' AND t1='0' ) OR ( s1='0' AND t1='0' ) OR ( s5='1' AND s6='0' ) OR ( s4='1' AND s6='0'
) OR ( s3='1' AND s6='0' ) OR ( s2='1' AND s6='0' ) OR ( s1='0' AND s6='0' ) THEN
                next_sreg<=STATE5;
                next_a1<='0';
                next_a2<='0';
                next_a3<='0';
```

```
                next_a4<='0';
                next_b1<='0';
                next_b2<='0';
                next_b3<='1';
                next_b4<='1';
                next_c1<='1';
                next_c2<='1';
                next_c3<='0';
                next_c4<='0';
        END IF;
WHEN STATE6 =>
        IF ( t2='1' ) THEN
                next_sreg<=STATE7;
                next_a1<='0';
                next_a2<='1';
                next_a3<='0';
                next_a4<='0';
                next_b1<='0';
                next_b2<='0';
                next_b3<='1';
                next_b4<='1';
                next_c1<='1';
                next_c2<='0';
                next_c3<='0';
                next_c4<='0';
         ELSE
                next_sreg<=STATE6;
                next_a1<='0';
                next_a2<='1';
                next_a3<='0';
                next_a4<='0';
                next_b1<='0';
                next_b2<='0';
                next_b3<='1';
                next_b4<='1';
                next_c1<='1';
                next_c2<='1';
                next_c3<='0';
                next_c4<='0';
        END IF;
WHEN STATE7 =>
        IF ( t3='1' ) THEN
                next_sreg<=STATE8;
                next_a1<='1';
                next_a2<='1';
                next_a3<='0';
                next_a4<='0';
                next_b1<='0';
                next_b2<='0';
                next_b3<='1';
                next_b4<='1';
                next_c1<='1';
                next_c2<='0';
                next_c3<='0';
                next_c4<='0';
         ELSE
```

```
                next_sreg<=STATE7;
                next_a1<='0';
                next_a2<='1';
                next_a3<='0';
                next_a4<='0';
                next_b1<='0';
                next_b2<='0';
                next_b3<='1';
                next_b4<='1';
                next_c1<='1';
                next_c2<='0';
                next_c3<='0';
                next_c4<='0';
        END IF;
WHEN STATE8 =>
        IF ( t4='1' ) THEN
                next_sreg<=STATE9;
                next_a1<='1';
                next_a2<='1';
                next_a3<='0';
                next_a4<='0';
                next_b1<='0';
                next_b2<='0';
                next_b3<='1';
                next_b4<='1';
                next_c1<='0';
                next_c2<='0';
                next_c3<='0';
                next_c4<='0';
         ELSE
                next_sreg<=STATE8;
                next_a1<='1';
                next_a2<='1';
                next_a3<='0';
                next_a4<='0';
                next_b1<='0';
                next_b2<='0';
                next_b3<='1';
                next_b4<='1';
                next_c1<='1';
                next_c2<='0';
                next_c3<='0';
                next_c4<='0';
        END IF;
WHEN STATE9 =>
        IF ( t5='1' ) THEN
                next_sreg<=STATE5;
                next_a1<='0';
                next_a2<='0';
                next_a3<='0';
                next_a4<='0';
                next_b1<='0';
                next_b2<='0';
                next_b3<='1';
                next_b4<='1';
                next_c1<='1';
```

```
                    next_c2<='1';
                    next_c3<='0';
                    next_c4<='0';
                ELSE
                    next_sreg<=STATE9;
                    next_a1<='1';
                    next_a2<='1';
                    next_a3<='0';
                    next_a4<='0';
                    next_b1<='0';
                    next_b2<='0';
                    next_b3<='1';
                    next_b4<='1';
                    next_c1<='0';
                    next_c2<='0';
                    next_c3<='0';
                    next_c4<='0';
                END IF;
            WHEN STATE10 =>
        IF ( s2='1' AND s1='0' AND s3='0' AND s4='0' AND s5='0' AND s6='0' )
                    THEN
                    next_sreg<=STATE15;
                    next_a1<='1';
                    next_a2<='1';
                    next_a3<='0';
                    next_a4<='0';
                    next_b1<='0';
                    next_b2<='0';
                    next_b3<='0';
                    next_b4<='0';
                    next_c1<='0';
                    next_c2<='0';
                    next_c3<='1';
                    next_c4<='1';
                END IF;
                IF ( t1='1' AND s1='1' ) THEN
                    next_sreg<=STATE11;
                    next_a1<='1';
                    next_a2<='1';
                    next_a3<='0';
                    next_a4<='0';
                    next_b1<='0';
                    next_b2<='0';
                    next_b3<='1';
                    next_b4<='1';
                    next_c1<='0';
                    next_c2<='0';
                    next_c3<='0';
                    next_c4<='1';
                END IF;
IF ( s6='1' AND t1='0' ) OR ( s5='1' AND t1='0' ) OR ( s4='1' AND t1='0') OR ( s3='1' AND t1='0' )
OR ( s1='1' AND t1='0' ) OR ( s2='0' AND t1='0' ) OR ( s6='1' AND s1='0' ) OR ( s5='1' AND s1='0'
) OR ( s4='1' AND s1='0' ) OR ( s3='1' AND s1='0' ) OR ( s2='0' AND s1='0' ) THEN
                    next_sreg<=STATE10;
                    next_a1<='1';
                    next_a2<='1';
```

```
                next_a3<='0';
                next_a4<='0';
                next_b1<='0';
                next_b2<='0';
                next_b3<='1';
                next_b4<='1';
                next_c1<='0';
                next_c2<='0';
                next_c3<='0';
                next_c4<='0';
        END IF;
WHEN STATE11 =>
        IF ( t2='1' ) THEN
                next_sreg<=STATE12;
                next_a1<='1';
                next_a2<='1';
                next_a3<='0';
                next_a4<='0';
                next_b1<='0';
                next_b2<='0';
                next_b3<='1';
                next_b4<='0';
                next_c1<='0';
                next_c2<='0';
                next_c3<='0';
                next_c4<='1';
         ELSE
                next_sreg<=STATE11;
                next_a1<='1';
                next_a2<='1';
                next_a3<='0';
                next_a4<='0';
                next_b1<='0';
                next_b2<='0';
                next_b3<='1';
                next_b4<='1';
                next_c1<='0';
                next_c2<='0';
                next_c3<='0';
                next_c4<='1';
        END IF;
WHEN STATE12 =>
        IF ( t3='1' ) THEN
                next_sreg<=STATE13;
                next_a1<='1';
                next_a2<='1';
                next_a3<='0';
                next_a4<='0';
                next_b1<='0';
                next_b2<='0';
                next_b3<='1';
                next_b4<='0';
                next_c1<='0';
                next_c2<='0';
                next_c3<='1';
                next_c4<='1';
```

```
        ELSE
                next_sreg<=STATE12;
                next_a1<='1';
                next_a2<='1';
                next_a3<='0';
                next_a4<='0';
                next_b1<='0';
                next_b2<='0';
                next_b3<='1';
                next_b4<='0';
                next_c1<='0';
                next_c2<='0';
                next_c3<='0';
                next_c4<='1';
        END IF;
WHEN STATE13 =>
        IF ( t4='1' ) THEN
                next_sreg<=STATE14;
                next_a1<='1';
                next_a2<='1';
                next_a3<='0';
                next_a4<='0';
                next_b1<='0';
                next_b2<='0';
                next_b3<='0';
                next_b4<='0';
                next_c1<='0';
                next_c2<='0';
                next_c3<='1';
                next_c4<='1';
        ELSE
                next_sreg<=STATE13;
                next_a1<='1';
                next_a2<='1';
                next_a3<='0';
                next_a4<='0';
                next_b1<='0';
                next_b2<='0';
                next_b3<='1';
                next_b4<='0';
                next_c1<='0';
                next_c2<='0';
                next_c3<='1';
                next_c4<='1';
        END IF;
WHEN STATE14 =>
        IF ( t5='1' ) THEN
                next_sreg<=STATE10;
                next_a1<='1';
                next_a2<='1';
                next_a3<='0';
                next_a4<='0';
                next_b1<='0';
                next_b2<='0';
                next_b3<='1';
                next_b4<='1';
```

```
                    next_c1<='0';
                    next_c2<='0';
                    next_c3<='0';
                    next_c4<='0';
                ELSE
                    next_sreg<=STATE14;
                    next_a1<='1';
                    next_a2<='1';
                    next_a3<='0';
                    next_a4<='0';
                    next_b1<='0';
                    next_b2<='0';
                    next_b3<='0';
                    next_b4<='0';
                    next_c1<='0';
                    next_c2<='0';
                    next_c3<='1';
                    next_c4<='1';
                END IF;
            WHEN STATE15 =>
        IF ( s3='1' AND s1='0' AND s2='0' AND s4='0' AND s5='0' AND s6='0' )
                    THEN
                    next_sreg<=STATE20;
                    next_a1<='0';
                    next_a2<='0';
                    next_a3<='0';
                    next_a4<='0';
                    next_b1<='1';
                    next_b2<='1';
                    next_b3<='0';
                    next_b4<='0';
                    next_c1<='0';
                    next_c2<='0';
                    next_c3<='1';
                    next_c4<='1';
                END IF;
                IF ( t1='1' AND s2='1' ) THEN
                    next_sreg<=STATE16;
                    next_a1<='1';
                    next_a2<='1';
                    next_a3<='0';
                    next_a4<='0';
                    next_b1<='0';
                    next_b2<='1';
                    next_b3<='0';
                    next_b4<='0';
                    next_c1<='0';
                    next_c2<='0';
                    next_c3<='1';
                    next_c4<='1';
                END IF;
IF ( s6='1' AND t1='0' ) OR ( s5='1' AND t1='0' ) OR ( s4='1' AND t1='0') OR ( s2='1' AND t1='0' )
OR ( s1='1' AND t1='0' ) OR ( s3='0' AND t1='0' ) OR ( s6='1' AND s2='0' ) OR ( s5='1' AND s2='0'
) OR ( s4='1' AND s2='0' ) OR ( s1='1' AND s2='0' ) OR ( s3='0' AND s2='0' ) THEN
                    next_sreg<=STATE15;
                    next_a1<='1';
```

```
                next_a2<='1';
                next_a3<='0';
                next_a4<='0';
                next_b1<='0';
                next_b2<='0';
                next_b3<='0';
                next_b4<='0';
                next_c1<='0';
                next_c2<='0';
                next_c3<='1';
                next_c4<='1';
        END IF;
WHEN STATE16 =>
        IF ( t2='1' ) THEN
                next_sreg<=STATE17;
                next_a1<='1';
                next_a2<='0';
                next_a3<='0';
                next_a4<='0';
                next_b1<='0';
                next_b2<='1';
                next_b3<='0';
                next_b4<='0';
                next_c1<='0';
                next_c2<='0';
                next_c3<='1';
                next_c4<='1';
         ELSE
                next_sreg<=STATE16;
                next_a1<='1';
                next_a2<='1';
                next_a3<='0';
                next_a4<='0';
                next_b1<='0';
                next_b2<='1';
                next_b3<='0';
                next_b4<='0';
                next_c1<='0';
                next_c2<='0';
                next_c3<='1';
                next_c4<='1';
        END IF;
WHEN STATE17 =>
        IF ( t3='1' ) THEN
                next_sreg<=STATE18;
                next_a1<='1';
                next_a2<='0';
                next_a3<='0';
                next_a4<='0';
                next_b1<='1';
                next_b2<='1';
                next_b3<='0';
                next_b4<='0';
                next_c1<='0';
                next_c2<='0';
                next_c3<='1';
```

```
                next_c4<='1';
        ELSE
                next_sreg<=STATE17;
                next_a1<='1';
                next_a2<='0';
                next_a3<='0';
                next_a4<='0';
                next_b1<='0';
                next_b2<='1';
                next_b3<='0';
                next_b4<='0';
                next_c1<='0';
                next_c2<='0';
                next_c3<='1';
                next_c4<='1';
        END IF;
WHEN STATE18 =>
        IF ( t4='1' ) THEN
                next_sreg<=STATE19;
                next_a1<='0';
                next_a2<='0';
                next_a3<='0';
                next_a4<='0';
                next_b1<='1';
                next_b2<='1';
                next_b3<='0';
                next_b4<='0';
                next_c1<='0';
                next_c2<='0';
                next_c3<='1';
                next_c4<='1';
        ELSE
                next_sreg<=STATE18;
                next_a1<='1';
                next_a2<='0';
                next_a3<='0';
                next_a4<='0';
                next_b1<='1';
                next_b2<='1';
                next_b3<='0';
                next_b4<='0';
                next_c1<='0';
                next_c2<='0';
                next_c3<='1';
                next_c4<='1';
        END IF;
WHEN STATE19 =>
        IF ( t5='1' ) THEN
                next_sreg<=STATE15;
                next_a1<='1';
                next_a2<='1';
                next_a3<='0';
                next_a4<='0';
                next_b1<='0';
                next_b2<='0';
                next_b3<='0';
```

```
                next_b4<='0';
                next_c1<='0';
                next_c2<='0';
                next_c3<='1';
                next_c4<='1';
        ELSE
                next_sreg<=STATE19;
                next_a1<='0';
                next_a2<='0';
                next_a3<='0';
                next_a4<='0';
                next_b1<='1';
                next_b2<='1';
                next_b3<='0';
                next_b4<='0';
                next_c1<='0';
                next_c2<='0';
                next_c3<='1';
                next_c4<='1';
        END IF;
    WHEN STATE20 =>
IF ( s4='1' AND s5='0' AND s6='0' AND s1='0' AND s2='0' AND s3='0' )
                THEN
                next_sreg<=STATE25;
                next_a1<='0';
                next_a2<='0';
                next_a3<='1';
                next_a4<='1';
                next_b1<='1';
                next_b2<='1';
                next_b3<='0';
                next_b4<='0';
                next_c1<='0';
                next_c2<='0';
                next_c3<='0';
                next_c4<='0';
        END IF;
        IF ( t1='1' AND s3='1' ) THEN
                next_sreg<=STATE21;
                next_a1<='0';
                next_a2<='0';
                next_a3<='0';
                next_a4<='1';
                next_b1<='1';
                next_b2<='1';
                next_b3<='0';
                next_b4<='0';
                next_c1<='0';
                next_c2<='0';
                next_c3<='1';
                next_c4<='1';
        END IF;
IF ( s3='1' AND t1='0' ) OR ( s2='1' AND t1='0' ) OR ( s1='1' AND t1='0') OR ( s6='1' AND t1='0' )
OR ( s5='1' AND t1='0' ) OR ( s4='0' AND t1='0' ) OR ( s2='1' AND s3='0' ) OR ( s1='1' AND s3='0'
) OR ( s6='1' AND s3='0' ) OR ( s5='1' AND s3='0' ) OR ( s4='0' AND s3='0' ) THEN
                next_sreg<=STATE20;
```

```
                next_a1<='0';
                next_a2<='0';
                next_a3<='0';
                next_a4<='0';
                next_b1<='1';
                next_b2<='1';
                next_b3<='0';
                next_b4<='0';
                next_c1<='0';
                next_c2<='0';
                next_c3<='1';
                next_c4<='1';
        END IF;
WHEN STATE21 =>
        IF ( t2='1' ) THEN
                next_sreg<=STATE22;
                next_a1<='0';
                next_a2<='0';
                next_a3<='0';
                next_a4<='1';
                next_b1<='1';
                next_b2<='1';
                next_b3<='0';
                next_b4<='0';
                next_c1<='0';
                next_c2<='0';
                next_c3<='1';
                next_c4<='0';
         ELSE
                next_sreg<=STATE21;
                next_a1<='0';
                next_a2<='0';
                next_a3<='0';
                next_a4<='1';
                next_b1<='1';
                next_b2<='1';
                next_b3<='0';
                next_b4<='0';
                next_c1<='0';
                next_c2<='0';
                next_c3<='1';
                next_c4<='1';
        END IF;
WHEN STATE22 =>
        IF ( t3='1' ) THEN
                next_sreg<=STATE23;
                next_a1<='0';
                next_a2<='0';
                next_a3<='1';
                next_a4<='1';
                next_b1<='1';
                next_b2<='1';
                next_b3<='0';
                next_b4<='0';
                next_c1<='0';
                next_c2<='0';
```

```
                next_c3<='1';
                next_c4<='0';
        ELSE
                next_sreg<=STATE22;
                next_a1<='0';
                next_a2<='0';
                next_a3<='0';
                next_a4<='1';
                next_b1<='1';
                next_b2<='1';
                next_b3<='0';
                next_b4<='0';
                next_c1<='0';
                next_c2<='0';
                next_c3<='1';
                next_c4<='0';
        END IF;
WHEN STATE23 =>
        IF ( t4='1' ) THEN
                next_sreg<=STATE24;
                next_a1<='0';
                next_a2<='0';
                next_a3<='1';
                next_a4<='1';
                next_b1<='1';
                next_b2<='1';
                next_b3<='0';
                next_b4<='0';
                next_c1<='0';
                next_c2<='0';
                next_c3<='0';
                next_c4<='0';
        ELSE
                next_sreg<=STATE23;
                next_a1<='0';
                next_a2<='0';
                next_a3<='1';
                next_a4<='1';
                next_b1<='1';
                next_b2<='1';
                next_b3<='0';
                next_b4<='0';
                next_c1<='0';
                next_c2<='0';
                next_c3<='1';
                next_c4<='0';
        END IF;
WHEN STATE24 =>
        IF ( t5='1' ) THEN
                next_sreg<=STATE20;
                next_a1<='0';
                next_a2<='0';
                next_a3<='0';
                next_a4<='0';
                next_b1<='1';
                next_b2<='1';
```

```
                next_b3<='0';
                next_b4<='0';
                next_c1<='0';
                next_c2<='0';
                next_c3<='1';
                next_c4<='1';
        ELSE
                next_sreg<=STATE24;
                next_a1<='0';
                next_a2<='0';
                next_a3<='1';
                next_a4<='1';
                next_b1<='1';
                next_b2<='1';
                next_b3<='0';
                next_b4<='0';
                next_c1<='0';
                next_c2<='0';
                next_c3<='0';
                next_c4<='0';
        END IF;
WHEN STATE25 =>
        IF ( t1='1' AND s4='1' ) THEN
                next_sreg<=STATE26;
                next_a1<='0';
                next_a2<='0';
                next_a3<='1';
                next_a4<='1';
                next_b1<='1';
                next_b2<='1';
                next_b3<='0';
                next_b4<='0';
                next_c1<='0';
                next_c2<='1';
                next_c3<='0';
                next_c4<='0';
        ELSE
                next_sreg<=STATE25;
                next_a1<='0';
                next_a2<='0';
                next_a3<='1';
                next_a4<='1';
                next_b1<='1';
                next_b2<='1';
                next_b3<='0';
                next_b4<='0';
                next_c1<='0';
                next_c2<='0';
                next_c3<='0';
                next_c4<='0';
        END IF;
WHEN STATE26 =>
        IF ( t2='1' ) THEN
                next_sreg<=STATE27;
                next_a1<='0';
                next_a2<='0';
```

```
                next_a3<='1';
                next_a4<='1';
                next_b1<='1';
                next_b2<='0';
                next_b3<='0';
                next_b4<='0';
                next_c1<='0';
                next_c2<='1';
                next_c3<='0';
                next_c4<='0';
        ELSE
                next_sreg<=STATE26;
                next_a1<='0';
                next_a2<='0';
                next_a3<='1';
                next_a4<='1';
                next_b1<='1';
                next_b2<='1';
                next_b3<='0';
                next_b4<='0';
                next_c1<='0';
                next_c2<='1';
                next_c3<='0';
                next_c4<='0';
        END IF;
WHEN STATE27 =>
        IF ( t3='1' ) THEN
                next_sreg<=STATE28;
                next_a1<='0';
                next_a2<='0';
                next_a3<='1';
                next_a4<='1';
                next_b1<='1';
                next_b2<='0';
                next_b3<='0';
                next_b4<='0';
                next_c1<='1';
                next_c2<='1';
                next_c3<='0';
                next_c4<='0';
        ELSE
                next_sreg<=STATE27;
                next_a1<='0';
                next_a2<='0';
                next_a3<='1';
                next_a4<='1';
                next_b1<='1';
                next_b2<='0';
                next_b3<='0';
                next_b4<='0';
                next_c1<='0';
                next_c2<='1';
                next_c3<='0';
                next_c4<='0';
        END IF;
WHEN STATE28 =>
```

```
                IF ( t4='1' ) THEN
                        next_sreg<=STATE29;
                        next_a1<='0';
                        next_a2<='0';
                        next_a3<='1';
                        next_a4<='1';
                        next_b1<='0';
                        next_b2<='0';
                        next_b3<='0';
                        next_b4<='0';
                        next_c1<='1';
                        next_c2<='1';
                        next_c3<='0';
                        next_c4<='0';
                 ELSE
                        next_sreg<=STATE28;
                        next_a1<='0';
                        next_a2<='0';
                        next_a3<='1';
                        next_a4<='1';
                        next_b1<='1';
                        next_b2<='0';
                        next_b3<='0';
                        next_b4<='0';
                        next_c1<='1';
                        next_c2<='1';
                        next_c3<='0';
                        next_c4<='0';
                END IF;
            WHEN STATE29 =>
IF ( s5='1' AND s6='0' AND s1='0' AND s2='0' AND s3='0' AND s4='0' )
                        THEN
                        next_sreg<=STATE4;
                        next_a1<='0';
                        next_a2<='0';
                        next_a3<='0';
                        next_a4<='0';
                        next_b1<='0';
                        next_b2<='0';
                        next_b3<='0';
                        next_b4<='1';
                        next_c1<='1';
                        next_c2<='1';
                        next_c3<='0';
                        next_c4<='0';
                END IF;
                IF ( t5='1' AND s4='1' ) THEN
                        next_sreg<=STATE25;
                        next_a1<='0';
                        next_a2<='0';
                        next_a3<='1';
                        next_a4<='1';
                        next_b1<='1';
                        next_b2<='1';
                        next_b3<='0';
                        next_b4<='0';
```

```
                                        next_c1<='0';
                                        next_c2<='0';
                                        next_c3<='0';
                                        next_c4<='0';
                                    END IF;
IF ( s4='1' AND t5='0' ) OR ( s3='1' AND t5='0' ) OR ( s2='1' AND t5='0' ) OR ( s1='1' AND t5='0' )
OR ( s6='1' AND t5='0' ) OR ( s5='0' AND t5='0' ) OR ( s3='1' AND s4='0' ) OR ( s2='1' AND s4='0'
) OR ( s1='1' AND s4='0' ) OR ( s6='1' AND s4='0' ) OR ( s5='0' AND s4='0' ) THEN
                                        next_sreg<=STATE29;
                                        next_a1<='0';
                                        next_a2<='0';
                                        next_a3<='1';
                                        next_a4<='1';
                                        next_b1<='0';
                                        next_b2<='0';
                                        next_b3<='0';
                                        next_b4<='0';
                                        next_c1<='1';
                                        next_c2<='1';
                                        next_c3<='0';
                                        next_c4<='0';
                                    END IF;
                                WHEN STATE30 =>
                            IF ( s5='1' AND s6='0' AND s4='0' AND s3='0' AND s2='0' AND s1='0' )
                                        THEN
                                        next_sreg<=STATE0;
                                        next_a1<='0';
                                        next_a2<='0';
                                        next_a3<='1';
                                        next_a4<='1';
                                        next_b1<='0';
                                        next_b2<='0';
                                        next_b3<='0';
                                        next_b4<='0';
                                        next_c1<='1';
                                        next_c2<='1';
                                        next_c3<='0';
                                        next_c4<='0';
                                    END IF;
                            IF ( s4='1' AND s1='0' AND s2='0' AND s3='0' AND s5='0' AND s6='0' )
                                        THEN
                                        next_sreg<=STATE25;
                                        next_a1<='0';
                                        next_a2<='0';
                                        next_a3<='1';
                                        next_a4<='1';
                                        next_b1<='1';
                                        next_b2<='1';
                                        next_b3<='0';
                                        next_b4<='0';
                                        next_c1<='0';
                                        next_c2<='0';
                                        next_c3<='0';
                                        next_c4<='0';
                                    END IF;
                            IF ( s3='1' AND s1='0' AND s2='0' AND s4='0' AND s5='0' AND s6='0' )
```

```
                                    THEN
                                    next_sreg<=STATE20;
                                    next_a1<='0';
                                    next_a2<='0';
                                    next_a3<='0';
                                    next_a4<='0';
                                    next_b1<='1';
                                    next_b2<='1';
                                    next_b3<='0';
                                    next_b4<='0';
                                    next_c1<='0';
                                    next_c2<='0';
                                    next_c3<='1';
                                    next_c4<='1';
                            END IF;
                IF ( s2='1' AND s1='0' AND s3='0' AND s4='0' AND s5='0' AND s6='0' )
                                    THEN
                                    next_sreg<=STATE15;
                                    next_a1<='1';
                                    next_a2<='1';
                                    next_a3<='0';
                                    next_a4<='0';
                                    next_b1<='0';
                                    next_b2<='0';
                                    next_b3<='0';
                                    next_b4<='0';
                                    next_c1<='0';
                                    next_c2<='0';
                                    next_c3<='1';
                                    next_c4<='1';
                            END IF;
                IF ( s1='1' AND s2='0' AND s3='0' AND s4='0' AND s5='0' AND s6='0' )
                                    THEN
                                    next_sreg<=STATE10;
                                    next_a1<='1';
                                    next_a2<='1';
                                    next_a3<='0';
                                    next_a4<='0';
                                    next_b1<='0';
                                    next_b2<='0';
                                    next_b3<='1';
                                    next_b4<='1';
                                    next_c1<='0';
                                    next_c2<='0';
                                    next_c3<='0';
                                    next_c4<='0';
                            END IF;
IF ( s5='0' AND s4='0' AND s3='0' AND s2='0' AND s1='0' ) OR ( s2='1' AND s3='1' ) OR ( s4='1'
AND s5='1' ) OR ( s2='1' AND s4='1' ) OR ( s2='1' AND s5='1' ) OR ( s1='1' AND s2='1' ) OR (
s3='1' AND s4='1' ) OR ( s3='1' AND s5='1' ) OR ( s1='1' AND s3='1' ) OR ( s1='1' AND s4='1' )
OR ( s1='1' AND s5='1' ) OR ( s6='1' ) THEN                 next_sreg<=STATE30;
                                    next_a1<='0';
                                    next_a2<='0';
                                    next_a3<='0';
                                    next_a4<='0';
                                    next_b1<='0';
```

```
                                        next_b2<='0';
                                        next_b3<='0';
                                        next_b4<='0';
                                        next_c1<='0';
                                        next_c2<='0';
                                        next_c3<='0';
                                        next_c4<='0';
                                    END IF;
                                WHEN OTHERS =>
                            END CASE;
                        END IF;
                END PROCESS;
END BEHAVIOR;
```

Interface M-file for importing VHDL file of rectification stage:

```
function STCAD1MO_config(this_block)

 this_block.setTopLevelLanguage('VHDL');

 this_block.setEntityName('STCAD1MO');

 % System Generator has to assume that your entity  has a combinational feed through;
 %   if it  doesn't, then comment out the following line:
 % this_block.tagAsCombinational;
 this_block.tagAsCombinational;

 this_block.addSimulinkInport('s1');
 this_block.addSimulinkInport('s2');
 this_block.addSimulinkInport('s3');
 this_block.addSimulinkInport('s4');
 this_block.addSimulinkInport('s5');
 this_block.addSimulinkInport('s6');
 this_block.addSimulinkInport('t1');
 this_block.addSimulinkInport('t2');
 this_block.addSimulinkInport('t3');
 this_block.addSimulinkInport('t4');
 this_block.addSimulinkInport('t5');
 this_block.addSimulinkInport('reset');



 this_block.addSimulinkOutport('a1');
 this_block.addSimulinkOutport('a2');
 this_block.addSimulinkOutport('a3');
 this_block.addSimulinkOutport('a4');
 this_block.addSimulinkOutport('b1');
 this_block.addSimulinkOutport('b2');
 this_block.addSimulinkOutport('b3');
 this_block.addSimulinkOutport('b4');
 this_block.addSimulinkOutport('c1');
 this_block.addSimulinkOutport('c2');
 this_block.addSimulinkOutport('c3');
 this_block.addSimulinkOutport('c4');
```

```
s1 = this_block.port('s1');
s1.setType('UFix_1_0');
s1.useHDLVector(false);
s2 = this_block.port('s2');
s2.setType('UFix_1_0');
s2.useHDLVector(false);
s3 = this_block.port('s3');
s3.setType('UFix_1_0');
s3.useHDLVector(false);
s4 = this_block.port('s4');
s4.setType('UFix_1_0');
s4.useHDLVector(false);
s5 = this_block.port('s5');
s5.setType('UFix_1_0');
s5.useHDLVector(false);
s6 = this_block.port('s6');
s6.setType('UFix_1_0');
s6.useHDLVector(false);
reset = this_block.port('reset');
reset.setType('UFix_1_0');
reset.useHDLVector(false);

t1 = this_block.port('t1');
t1.setType('Bool');
t1.useHDLVector(false);
t2 = this_block.port('t2');
t2.setType('Bool');
t2.useHDLVector(false);
t3 = this_block.port('t3');
t3.setType('Bool');
t3.useHDLVector(false);
t4 = this_block.port('t4');
t4.setType('Bool');
t4.useHDLVector(false);
t5 = this_block.port('t5');
t5.setType('Bool');
t5.useHDLVector(false);

a1 = this_block.port('a1');
a1.setType('Bool');
a1.useHDLVector(false);

a2 = this_block.port('a2');
a2.setType('Bool');
a2.useHDLVector(false);

a3 = this_block.port('a3');
a3.setType('Bool');
a3.useHDLVector(false);


a4 = this_block.port('a4');
a4.setType('Bool');
a4.useHDLVector(false);

b1 = this_block.port('b1');
```

```
b1.setType('Bool');
b1.useHDLVector(false);

b2 = this_block.port('b2');
b2.setType('Bool');
b2.useHDLVector(false);

b3 = this_block.port('b3');
b3.setType('Bool');
b3.useHDLVector(false);

b4 = this_block.port('b4');
b4.setType('Bool');
b4.useHDLVector(false);


c1 = this_block.port('c1');
c1.setType('Bool');
c1.useHDLVector(false);

c2 = this_block.port('c2');
c2.setType('Bool');
c2.useHDLVector(false);

c3 = this_block.port('c3');
c3.setType('Bool');
c3.useHDLVector(false);


c4 = this_block.port('c4');
c4.setType('Bool');
c4.useHDLVector(false);



if (this_block.inputTypesKnown)

   if (this_block.port('s1').width ~= 1);
    this_block.setError('Input data type for port "s1" must have width=1.');
   end
  if (this_block.port('s2').width ~= 1);
    this_block.setError('Input data type for port "s2" must have width=1.');
  end
     if (this_block.port('s3').width ~= 1);
    this_block.setError('Input data type for port "s3" must have width=1.');
  end
  if (this_block.port('s4').width ~= 1);
    this_block.setError('Input data type for port "s4" must have width=1.');
  end
  if (this_block.port('s5').width ~= 1);
    this_block.setError('Input data type for port "s5" must have width=1.');
  end
  if (this_block.port('s6').width ~= 1);
    this_block.setError('Input data type for port "s6" must have width=1.');
  end
   if (this_block.port('t1').width ~= 1);
```

```
    this_block.setError('Input data type for port "t1" must have width=1.');
   end
    if (this_block.port('t2').width ~= 1);
    this_block.setError('Input data type for port "t2" must have width=1.');
   end
   if (this_block.port('t3').width ~= 1);
    this_block.setError('Input data type for port "t3" must have width=1.');
   end
   if (this_block.port('t4').width ~= 1);
    this_block.setError('Input data type for port "t4" must have width=1.');
   end


   if (this_block.port('t5').width ~= 1);
    this_block.setError('Input data type for port "t5" must have width=1.');
   end
    if (this_block.port('reset').width ~= 1);
    this_block.setError('Input data type for port "reset" must have width=1.');
   end
   this_block.port('reset').useHDLVector(false);
   end  % if(inputTypesKnown)
   % -----------------------------
   if (this_block.inputRatesKnown)

  setup_as_single_rate(this_block,'clk_1','ce_1')
  end  % if(inputRatesKnown)

  this_block.addFile('STCAD1MO.vhd');
  return;
  % ------------------------------------------------------------

function setup_as_single_rate(block,clkname,cename)
inputRates = block.inputRates;
uniqueInputRates = unique(inputRates);
if (length(uniqueInputRates)==1 && uniqueInputRates(1)==Inf)
 block.setError('The inputs to this block cannot all be constant.');
 return;
end
if (uniqueInputRates(end) == Inf)
  hasConstantInput = true;
  uniqueInputRates = uniqueInputRates(1:end-1);
end
if (length(uniqueInputRates) ~= 1)
  block.setError('The inputs to this block must run at a single rate.');
 return;
end
theInputRate = uniqueInputRates(1);
 for i = 1:block.numSimulinkOutports
  block.outport(i).setRate(theInputRate);
 end
block.addClkCEPair(clkname,cename,theInputRate);
  return;

% ------------------------------------------------------------
```

M-files of Mcode block for comparator section in Simulation:

```
function [s1, s2, s3, s4, s5, s6, Sn]=x1angle1(teta)

pi =3.14;
sp6 = xfix({xlUnsigned, 10, 6}, pi/6);
sp2 = xfix({xlUnsigned, 10, 6}, pi/2);
s5p6 = xfix({xlUnsigned, 10, 6}, 5*pi/6);
s7p6 = xfix({xlUnsigned, 10, 6}, 7*pi/6);
s11p6 = xfix({xlUnsigned, 10, 6}, 11*pi/6);
s9p6 = xfix({xlUnsigned, 10, 6}, 9*pi/6);
if (teta == s11p6)
   s2=0;s1=0;s3=0;s4=0;s5=1;s6=0;Sn=5;
elseif (teta == sp6)
   s1=0;s2=0;s3=0;s4=0;s5=0;s6=1;Sn=6;
  elseif (teta == sp2)
   s1=1;s2=0;s3=0;s4=0;s5=0;s6=0;Sn=1;
elseif (teta == s5p6)
   s1=0;s2=1;s3=0;s4=0;s5=0;s6=0;Sn=2;
elseif (teta == s7p6)
   s1=0;s2=0;s3=1;s4=0;s5=0;s6=0;Sn=3;
else
   s1=0;s2=0;s3=0;s4=1;s5=0;s6=0;Sn=4;
end

Sn = xfix({xlUnsigned, 8, 3}, Sn);
% ------------------------------------------------------------


function [s1, s2, s3, s4, s5, s6, Sn]=x1angle3(teta)


if (teta == 1)
   s2=0;s1=1;s3=0;s4=0;s5=0;s6=0;Sn=1;
elseif (teta == 2)
   s1=0;s2=1;s3=0;s4=0;s5=0;s6=0;Sn=2;
  elseif (teta == 3)
   s1=0;s2=0;s3=1;s4=0;s5=0;s6=0;Sn=3;
elseif (teta == 4)
   s1=0;s2=0;s3=0;s4=1;s5=0;s6=0;Sn=4;
elseif (teta == 5)
   s1=0;s2=0;s3=0;s4=0;s5=1;s6=0;Sn=5;
else
   s1=0;s2=0;s3=0;s4=0;s5=0;s6=1;Sn=6;
end

Sn = xfix({xlUnsigned, 8, 3}, Sn);
% ------------------------------------------------------------


function [phase]=xphase(Va,Vb,Vc)
```

```
pi=3.14;
if (Vc>0) && (Va<Vc) && (Vb<0) && (Vb<=Va)
phase = xfix({xlUnsigned, 10, 6}, 11*pi/6);

elseif (Va>0)&& (Vb<0)&& (Vb<Vc)&& (Va>=Vc)
 phase = xfix({xlUnsigned,10, 6}, pi/6);

elseif (Va>0)&& (Vc<0)&& (Vb>=Vc)&& (Va>Vb)
  phase = xfix({xlUnsigned, 10, 6}, pi/2);

elseif (Vb>0)&& (Vc<0)&& (Vb>=Va)&& (Va>Vc)
   phase = xfix({xlUnsigned, 10, 6}, 5*pi/6);

elseif (Vb>0)&& (Va<0)&& (Vb>Vc)&& (Vc>=Va)
   phase = xfix({xlUnsigned, 10, 6}, 7*pi/6);

else
  phase = xfix({xlUnsigned, 10, 6}, 9*pi/6);

end
% ------------------------------------------------------------
```

Appendix D – Synthesis of VHDL codes of rectification stage in two-stage DPEC

Rectification stage VHDL codes synthesis :

```
Using wire table: xcv2-80-6_wc

    Pass    Area   Delay     DFFs  PIs  POs  --CPU--
         (LUTs)   (ns)                 min:sec
    1      129      6        0   48   10     00:00
Info: setting opt_best_result to 817.595550
Info: setting opt_best_pass to 0
-- Start optimization for design .work.STCAD1MO.BEHAVIOR
Using wire table: xcv2-80-6_wc

    Pass    Area   Delay     DFFs  PIs  POs  --CPU--
         (LUTs)   (ns)                 min:sec
    1      166      5       43   14   12     00:02
Info: setting opt_best_result to 891.721290
Info: setting opt_best_pass to 0
-- Start optimization for design .work.accumulator2_21daa97b4d.behavior_unfold_3604
Using wire table: xcv2-80-6_wc

    Pass    Area   Delay     DFFs  PIs  POs  --CPU--
         (LUTs)   (ns)                 min:sec
    1       62      5       32    3   32     00:00
Info: setting opt_best_result to 312.042900
Info: setting opt_best_pass to 0
-- Start optimization for design .work.xilinx_rectifier.structural
Using wire table: xcv2-80-6_wc


    Pass    Area   Delay     DFFs  PIs  POs  --CPU--
         (LUTs)   (ns)                 min:sec
    1      270      7       35   51   54     00:01
Info: setting opt_best_result to 1773.260100
Info: setting opt_best_pass to 0
Info, Added global buffer BUFGP for port clk_1
Info: setting optimize_timing_cpu_limit to 36940
Using wire table: xcv2-80-6_wc
-- Start timing optimization for design .work.comparateur_b01776cc87.behavior_unfold_2761
Starting Timing Characterization...
Starting Timing Analysis...
Using wire table: xcv2-80-6_wc
Timing analysis done, time = 4 CPU secs.
Timing characterization done, time = 4 CPU secs.


Initial Timing Optimization Statistics:
---------------------------------------
```

```
        Most Critical Slack   :     -5.3
        Sum of Negative Slacks :    -283.0
        Area                  :     129.0


Final Timing Optimization Statistics:
-------------------------------------
        Most Critical Slack   :     -5.0
        Sum of Negative Slacks :    -245.3
        Area                  :     136.0

        Total time taken : 9 cpu secs
-- Start timing optimization for design .work.STCAD1MO.BEHAVIOR

Initial Timing Optimization Statistics:
---------------------------------------
        Most Critical Slack   :     -5.1
        Sum of Negative Slacks :    -104.2
        Area                  :     166.0

Final Timing Optimization Statistics:
-------------------------------------
        Most Critical Slack   :     -4.0
        Sum of Negative Slacks :     -90.5
        Area                  :     181.0

        Total time taken : 17 cpu secs
-- Start timing optimization for design .work.accumulator2_21daa97b4d.behavior_unfold_3604
No critical paths to optimize at this level
-- Start timing optimization for design .work.xilinx_rectifier.structural


Initial Timing Optimization Statistics:
---------------------------------------
        Clock                 : Frequency
    ---------------------------------
        clk_1                 : 72.3 MHz


        Most Critical Slack   :     -5.9
        Sum of Negative Slacks :    -601.5
        Area                  :     270.0


Final Timing Optimization Statistics:
-------------------------------------
        Clock                 : Frequency
    ---------------------------------
        clk_1                 : 72.3 MHz

        Most Critical Slack   :     -5.9
        Sum of Negative Slacks :    -601.5
        Area                  :     270.0


        Total time taken : 0 cpu secs
```

```
Info: setting optimize_timing_cpu_limit to 0
Info: setting modgen_select to auto

*******************************************************

*******************************************************
Cell: xilinx_rectifier   View: structural   Library: work

Cell: xilinx_rectifier   View: structural   Library: work
*******************************************************
 Number of ports :                  105
 Number of nets :                   770
 Number of instances :              670
 Number of references to this view :  0


Total accumulated area :

 Number of BUFGP :                    1
 Number of Dffs or Latches :        110
 Number of Function Generators :    657
 Number of IBUF :                    50
 Number of LUTs :                     1
 Number of MUX CARRYs :             379
 Number of MUXF5 :                   10
 Number of OBUF :                    54
 Number of accumulated instances :  1298
Number of global buffers used:        1
***********************************************
Device Utilization for 2V80fg256
***********************************************
Resource             Used   Avail   Utilization
-----------------------------------------------
IOs                  104    120       86.67%
Global Buffers       1      16         6.25%
Function Generators  657    1024      64.16%
CLB Slices           329    512       64.26%
Dffs or Latches      110    1384       7.95%
Block RAMs           0      8          0.00%
Block Multipliers    0      8          0.00%
Block Multiplier Dffs 0     288        0.00%
-----------------------------------------------
                Clock Frequency Report

        Clock            : Frequency
    ------------------------------------
        clk_1            : 69.6 MHz

                Critical Path Report
Critical path #1, (path slack = -4.4):
NAME                                    GATE        ARRIVAL          LOAD
-------------------------------------------------------------------------------------------------
gateway_in3(0)/                                   0.00  0.00 up          0.40
gateway_in3_ibuf(0)/O                      IBUF     1.06  1.06 up          0.30
comparateur/rel_4_37_gt_3_ix41/LO                  LUT2_L   0.33  1.39 up          0.20
comparateur/rel_4_37_gt_3_ix43/LO                  MUXCY_L  0.32  1.72 up          0.40
```

```
comparateur/rel_4_37_gt_3_ix47/LO                MUXCY_L   0.04  1.76 up    0.40
comparateur/rel_4_37_gt_3_ix51/LO                MUXCY_L   0.04  1.80 up    0.40
comparateur/rel_4_37_gt_3_ix55/LO                MUXCY_L   0.04  1.84 up    0.40
comparateur/rel_4_37_gt_3_ix59/LO                MUXCY_L   0.04  1.88 up    0.40
comparateur/rel_4_37_gt_3_ix63/LO                MUXCY_L   0.04  1.92 up    0.40
comparateur/rel_4_37_gt_3_ix67/LO                MUXCY_L   0.04  1.96 up    0.40
comparateur/rel_4_37_gt_3_ix71/LO                MUXCY_L   0.04  2.00 up    0.40
comparateur/rel_4_37_gt_3_ix75/LO                MUXCY_L   0.04  2.04 up    0.40
comparateur/rel_4_37_gt_3_ix79/LO                MUXCY_L   0.04  2.08 up    0.40
comparateur/rel_4_37_gt_3_ix83/LO                MUXCY_L   0.04  2.12 up    0.40
comparateur/rel_4_37_gt_3_ix87/LO                MUXCY_L   0.04  2.16 up    0.40
comparateur/rel_4_37_gt_3_ix91/LO                MUXCY_L   0.04  2.20 up    0.40
comparateur/rel_4_37_gt_3_ix95/LO                MUXCY_L   0.04  2.24 up    0.40
comparateur/rel_4_37_gt_3_ix99/LO                MUXCY_L   0.04  2.28 up    0.40
comparateur/rel_4_37_gt_3_ix103/O                MUXCY     1.23  3.51 up    0.40
comparateur/nx310/O                              LUT4      0.89  4.40 up    0.40
comparateur_phase_net(4)/O                       LUT4      0.75  5.15 up    0.30
comparateur/nx400/O                              LUT3      0.75  5.90 up    0.30
comparateur_phase_net(7)/O                       LUT3      0.75  6.65 up    0.30
nx168/O                                          LUT4      0.89  7.54 up    0.40
s5/O                                             LUT4      1.17  8.70 up    0.60
comparateur1_sn_net_x0(3)/O                      LUT3      0.75  9.45 up    0.30
s4/O                                             LUT3      0.89  10.34 up   0.40
state_machine_68fb0a1815_black_box/nx9610/O      LUT4      0.75  11.09 up   0.30
state_machine_68fb0a1815_black_box/NOT_modgen_select_60_nx0/O
                                                 LUT4      0.61  11.70 up   0.20
state_machine_68fb0a1815_black_box/nx9591/O      LUT4      0.61  12.32 up   0.20
state_machine_68fb0a1815_black_box/nx9625/O      LUT4      0.89  13.20 up   0.40
state_machine_68fb0a1815_black_box/nx250/O       LUT4      0.89  14.09 up   0.40
state_machine_68fb0a1815_black_box/reg_c2/D      FDE       0.00  14.09 up   0.00
data arrival time                                14.09

data required time  (default specified - setup time)              9.72
--------------------------------------------------------------------------------
data required time                                9.72
data arrival time                                14.09
                                                 ----------
slack                                            -4.37
--------------------------------------------------------------------------------
-- Design summary in file 'xilinx_rectifier_4.sum'

AutoWrite args are : xilinx_rectifier_4.edf
-- Applying renaming rule 'XILINX' to database
Warning, Renaming will cause your database to change
Info: setting edif_array_range_extraction_style to %s<%d:%d>
-- Calling set_xilinx_eqn to set up writing Equations
-- Writing file xilinx_rectifier_4.edf
Info, Writing NCF file 'xilinx_rectifier_4.ncf'
-- Writing file xilinx_rectifier_4.ncf
-- CPU time taken for this run was  102.79 sec
-- Run Successfully Ended On  Tue Nov 28 22:13:52 Est (heure d'été) 2006
0
Info: Finished Synthesis run
```

Appendix E – Synthesis of VHDL codes of inversion stage in two-stage DPEC

Inversion stage VHDL code synthesis :

```
Using wire table: xcv2-40-6_wc

    Pass    Area   Delay    DFFs  PIs  POs  --CPU--
         (LUTs)  (ns)                min:sec
    1      205     42      210   70   70    00:03
Info: setting opt_best_result to 8581.845300
Info: setting opt_best_pass to 0
-- Start optimization for design .work.ONDSTAT1.BEHAVIOR
Using wire table: xcv2-40-6_wc

    Pass    Area   Delay    DFFs  PIs  POs  --CPU--
         (LUTs)  (ns)                min:sec
    1      144      4       36   14    6    00:03
Info: setting opt_best_result to 581.675040
Info: setting opt_best_pass to 0
-- Start optimization for design .work.mult6_5143d79c87.behavior_unfold_4540
Using wire table: xcv2-40-6_wc

    Pass    Area   Delay    DFFs  PIs  POs  --CPU--
         (LUTs)  (ns)                min:sec
    1      685     47      284   90   88    00:07
Info: setting opt_best_result to 32515.032000
Info: setting opt_best_pass to 0
-- Start optimization for design .work.cordic_pe2_entity_d7cb85ab14.structural
Using wire table: xcv2-40-6_wc

    Pass    Area   Delay    DFFs  PIs  POs  --CPU--
         (LUTs)  (ns)                min:sec
    1       32      2       45   47   45    00:00
Info: setting opt_best_result to 78.675840
Info: setting opt_best_pass to 0
-- Start optimization for design .work.cordic_pe3_entity_c535525af9.structural
Using wire table: xcv2-40-6_wc

    Pass    Area   Delay    DFFs  PIs  POs  --CPU--
         (LUTs)  (ns)                min:sec
    1       32      2       45   47   45    00:00
Info: setting opt_best_result to 78.675840
Info: setting opt_best_pass to 0
-- Start optimization for design .work.cordic_pe4_entity_45f1a217df.structural
Using wire table: xcv2-40-6_wc

    Pass    Area   Delay    DFFs  PIs  POs  --CPU--
         (LUTs)  (ns)                min:sec
    1       32      2       45   47   45    00:00
Info: setting opt_best_result to 78.675840
```

```
Info: setting opt_best_pass to 0
-- Start optimization for design .work.cordic_pe5_entity_00f10ddc66.structural
Using wire table: xcv2-40-6_wc

   Pass   Area   Delay    DFFs  PIs  POs  --CPU--
        (LUTs)  (ns)              min:sec
   1      32      2      45   47   45    00:00
Info: setting opt_best_result to 78.675840
Info: setting opt_best_pass to 0
-- Start optimization for design .work.cordic_pe6_entity_49d3efd70e.structural
Using wire table: xcv2-40-6_wc

   Pass   Area   Delay    DFFs  PIs  POs  --CPU--
        (LUTs)  (ns)              min:sec
   1      32      2      45   47   45    00:00
Info: setting opt_best_result to 78.675840
Info: setting opt_best_pass to 0
-- Start optimization for design .work.cordic_pe7_entity_1a3256e0c7.structural_unfold_4294
Using wire table: xcv2-40-6_wc

   Pass   Area   Delay    DFFs  PIs  POs  --CPU--
        (LUTs)  (ns)              min:sec
   1      32      2      38   41   45    00:00
Info: setting opt_best_result to 78.675840
Info: setting opt_best_pass to 0
-- Start optimization for design .work.cordic_pe8_entity_4f7f5f4cbc.structural_unfold_4452
Using wire table: xcv2-40-6_wc

   Pass   Area   Delay    DFFs  PIs  POs  --CPU--
        (LUTs)  (ns)              min:sec
   1      32      2      37   40   45    00:00
Info: setting opt_best_result to 78.675840
Info: setting opt_best_pass to 0
-- Start optimization for design .work.cordic_pe9_entity_9968961590.structural_unfold_4181
Using wire table: xcv2-40-6_wc

   Pass   Area   Delay    DFFs  PIs  POs  --CPU--
        (LUTs)  (ns)              min:sec
   1      32      2      36   39   45    00:00
Info: setting opt_best_result to 78.675840
Info: setting opt_best_pass to 0
-- Start optimization for design .work.iterative_divider_entity_2ec7b01ff2.structural_unfold_4744
Using wire table: xcv2-40-6_wc

   Pass   Area   Delay    DFFs  PIs  POs  --CPU--
        (LUTs)  (ns)              min:sec
   1      62      3      60   32   13    00:00
Info: setting opt_best_result to 197.474340
Info: setting opt_best_pass to 0
-- Start optimization for design .work.mult_c53a60460b.behavior_unfold_3142
Using wire table: xcv2-40-6_wc

   Pass   Area   Delay    DFFs  PIs  POs  --CPU--
        (LUTs)  (ns)              min:sec
   1       0      6      64   34   32    00:00
Info: setting opt_best_result to 0.000000
```

```
Info: setting opt_best_pass to 0
-- Start optimization for design .work.map_entity_6421dc7ffe.structural_unfold_3449
Using wire table: xcv2-40-6_wc

    Pass    Area   Delay    DFFs  PIs  POs  --CPU--
         (LUTs)  (ns)              min:sec
    1      59     4      42   23   33    00:00
Info: setting opt_best_result to 230.831010
Info: setting opt_best_pass to 0
-- Start optimization for design .work.normalize_x_entity_33dc6271e9.structural_unfold_3970
Using wire table: xcv2-40-6_wc

    Pass    Area   Delay    DFFs  PIs  POs  --CPU--
         (LUTs)  (ns)              min:sec
    1      206     2      208   14   31    00:27
Info: setting opt_best_result to 386.219100
Info: setting opt_best_pass to 0
-- Start optimization for design .work.normalize_y_entity_44e36148d1.structural_unfold_3920
Using wire table: xcv2-40-6_wc

    Pass    Area   Delay    DFFs  PIs  POs  --CPU--
         (LUTs)  (ns)              min:sec
    1      203     2      206   13   31    00:26
Info: setting opt_best_result to 380.594550
Info: setting opt_best_pass to 0
-- Start optimization for design .work.accumulator2_7b528c7815.behavior_unfold_3608
Using wire table: xcv2-40-6_wc

    Pass    Area   Delay    DFFs  PIs  POs  --CPU--
         (LUTs)  (ns)              min:sec
    1      87     5      45    3   45    00:03
Info: setting opt_best_result to 431.205930
Info: setting opt_best_pass to 0
-- Start optimization for design .work.xilinx_onduleur.structural
Using wire table: xcv2-40-6_wc

    Pass    Area   Delay    DFFs  PIs  POs  --CPU--
         (LUTs)  (ns)              min:sec
    1      951     12      30   13    6    00:13
Info: setting opt_best_result to 11752.790850
Info: setting opt_best_pass to 0
Info, Added global buffer BUFGP for port clk_1

************************************************
Device Utilization for 2V40fg256
************************************************
Resource             Used   Avail   Utilization
------------------------------------------------
IOs                  70     88      79.55%
Global Buffers       1      16      6.25%
Function Generators  2861   512     558.79%
CLB Slices           1431   256     558.98%
Dffs or Latches      1519   776     195.75%
Block RAMs           0      4       0.00%
Block Multipliers    9      4       225.00%
Block Multiplier Dffs  252    144     175.00%
```

```
---------------------------------------------
This design does not fit in the device specified!
Trying an alternate device ...
Info: setting part to 2V250fg256
Info: Reset Device to 2V250fg256
Info: setting wire_table to xcv2-250-6_wc
Info: Reset wire_table to xcv2-250-6_wc
*************************************************
Info: setting optimize_timing_cpu_limit to 505
Using wire table: xcv2-250-6_wc
-- Start timing optimization for design .work.mult1_a66ba9a3a2.behavior_unfold_3233
Starting Timing Characterization...
Starting Timing Analysis...
Using wire table: xcv2-250-6_wc
Timing analysis done, time = 33 CPU secs.
Timing characterization done, time = 33 CPU secs.
Info: setting optimize_timing_cpu_limit to 0
Info: setting modgen_select to auto

*******************************************************

Cell: xilinx_onduleur    View: structural    Library: work

Cell: xilinx_onduleur    View: structural    Library: work

*******************************************************
 Number of ports :                 71
 Number of nets :                2599
 Number of instances :           2290
 Number of references to this view :    0

Total accumulated area :

 Number of BUFGP :                  1
 Number of Dffs or Latches :     1519
 Number of Function Generators :    2861
 Number of IBUF :                12
 Number of LUTs :                 1
 Number of MUX CARRYs :          1320
 Number of MUXF5 :                51
 Number of OBUF :                 6
 Number of accumulated instances :   6506
Number of global buffers used: 1
*************************************************
Device Utilization for 2V250fg256
*************************************************
Resource             Used   Avail   Utilization
-------------------------------------------------
IOs                  70     172     40.70%
Global Buffers       1      16      6.25%
Function Generators  2861   3072    93.13%
CLB Slices           1431   1536    93.16%
Dffs or Latches      1519   3588    42.34%
Block RAMs           0      24      0.00%
```

```
Block Multipliers      9      24      37.50%
Block Multiplier Dffs  252    864     29.17%

---------------------------------------------
                Clock Frequency Report
        Clock              : Frequency
    ----------------------------------
        clk_1              : 18.1 MHz

                Critical Path Report
Critical path #1, (unconstrained path)
NAME                                        GATE          ARRIVAL          LOAD
------------------------------------------------------------------------------
clock information not specified
delay thru clock network                                                0.00 (ideal)
svm_generator_and_switching_timing_87c01ab953_mult6/ix1001/Q
                    FDE       0.00  0.99 up          0.40
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_ix2074/LO
                    LUT2_L    0.33  1.32 up          0.30
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx2320/O
                    LUT4      0.75  2.07 up          0.30
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx2322/O
                    LUT3      0.61  2.68 up          0.20
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx2324/O
                    LUT3      0.61  3.29 up          0.20
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx278/O
                    LUT3      0.61  3.90 up          0.20
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx284/O
                    LUT3      0.61  4.51 up          0.20
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx290/O
                    LUT3      0.61  5.12 up          0.20
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx2329/O
                    LUT3      0.61  5.73 up          0.20
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx2331/O
                    LUT3      0.61  6.34 up          0.20
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx2333/O
                    LUT3      0.61  6.95 up          0.20
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx2335/O
                    LUT3      0.61  7.56 up          0.20
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx2337/O
                    LUT3      0.61  8.18 up          0.20
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx2339/O
                    LUT3      0.61  8.79 up          0.20
```

svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx2341/O
LUT3 0.61 9.40 up 0.20
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx2343/O
LUT3 0.61 10.01 up 0.20
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx2345/O
LUT3 0.61 10.62 up 0.20
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx350/O
LUT3 0.61 11.23 up 0.20
svm_generator_and_switching_timing_87c01ab953_mult6/ix3590/O
MUXF5 0.91 12.14 up 0.40
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx362/O
LUT4 0.75 12.89 up 0.30
svm_generator_and_switching_timing_87c01ab953_mult6/ix3589/O
MUXF5 0.91 13.81 up 0.40
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx2348/O
LUT4 0.75 14.56 up 0.30
svm_generator_and_switching_timing_87c01ab953_mult6/ix3588/O
MUXF5 0.91 15.47 up 0.40
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx2352/O
LUT4 0.75 16.22 up 0.30
svm_generator_and_switching_timing_87c01ab953_mult6/ix3587/O
MUXF5 0.91 17.14 up 0.40
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx2356/O
LUT4 0.75 17.89 up 0.30
svm_generator_and_switching_timing_87c01ab953_mult6/ix3586/O
MUXF5 0.91 18.80 up 0.40
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx2360/O
LUT4 0.75 19.55 up 0.30
svm_generator_and_switching_timing_87c01ab953_mult6/ix3585/O
MUXF5 0.91 20.47 up 0.40
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx422/O
LUT4 0.75 21.22 up 0.30
svm_generator_and_switching_timing_87c01ab953_mult6/ix3584/O
MUXF5 0.91 22.13 up 0.40
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx434/O
LUT4 0.75 22.88 up 0.30
svm_generator_and_switching_timing_87c01ab953_mult6/ix3583/O
MUXF5 0.91 23.80 up 0.40
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx446/O
LUT4 0.75 24.55 up 0.30
svm_generator_and_switching_timing_87c01ab953_mult6/ix3582/O
MUXF5 0.91 25.46 up 0.40
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx2364/O

```
                LUT4      0.75  26.21 up          0.30
svm_generator_and_switching_timing_87c01ab953_mult6/ix3581/O
                MUXF5     0.91  27.13 up          0.40
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx2368/O
                LUT4      0.75  27.88 up          0.30
svm_generator_and_switching_timing_87c01ab953_mult6/ix3580/O
                MUXF5     0.91  28.79 up          0.40
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx2372/O
                LUT4      0.75  29.54 up          0.30
svm_generator_and_switching_timing_87c01ab953_mult6/ix3579/O
                MUXF5     0.91  30.46 up          0.40
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx494/O
                LUT4      0.75  31.21 up          0.30
svm_generator_and_switching_timing_87c01ab953_mult6/ix3578/O
                MUXF5     0.91  32.12 up          0.40
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx2379/O
                LUT4      0.75  32.87 up          0.30
svm_generator_and_switching_timing_87c01ab953_mult6/ix3577/O
                MUXF5     0.91  33.79 up          0.40
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx2383/O
                LUT4      0.75  34.54 up          0.30
svm_generator_and_switching_timing_87c01ab953_mult6/ix3576/O
                MUXF5     0.91  35.45 up          0.40
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx2387/O
                LUT4      0.75  36.20 up          0.30
svm_generator_and_switching_timing_87c01ab953_mult6/ix3575/O
                MUXF5     0.91  37.12 up          0.40
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx2391/O
                LUT4      0.75  37.87 up          0.30
svm_generator_and_switching_timing_87c01ab953_mult6/ix3574/O
                MUXF5     0.91  38.78 up          0.40
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx2395/O
                LUT4      0.75  39.53 up          0.30
svm_generator_and_switching_timing_87c01ab953_mult6/ix3573/O
                MUXF5     0.91  40.44 up          0.40
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx2399/O
                LUT4      0.75  41.19 up          0.30
svm_generator_and_switching_timing_87c01ab953_mult6/ix3572/O
                MUXF5     0.91  42.11 up          0.40
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx578/O
                LUT4      0.75  42.86 up          0.30
svm_generator_and_switching_timing_87c01ab953_mult6/ix3571/O
                MUXF5     0.91  43.77 up          0.40
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx590/O
                LUT4      0.75  44.52 up          0.30
```

```
svm_generator_and_switching_timing_87c01ab953_mult6/ix3570/O
                MUXF5      0.91  45.44 up          0.40
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx602/O
                LUT4       0.75  46.19 up          0.30
svm_generator_and_switching_timing_87c01ab953_mult6/ix3569/O
                MUXF5      0.91  47.10 up          0.40
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx2402/O
                LUT4       0.75  47.85 up          0.30
svm_generator_and_switching_timing_87c01ab953_mult6/ix3568/O
                MUXF5      0.91  48.77 up          0.40
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx2406/O
                LUT4       0.75  49.52 up          0.30
svm_generator_and_switching_timing_87c01ab953_mult6/ix3567/O
                MUXF5      0.91  50.43 up          0.40
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx2410/O
                LUT4       0.75  51.18 up          0.30
svm_generator_and_switching_timing_87c01ab953_mult6/ix3566/O
                MUXF5      0.91  52.10 up          0.40
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx2414/O
                LUT4       0.61  52.71 up          0.20
svm_generator_and_switching_timing_87c01ab953_mult6/ix3565/O
                MUXF5      0.64  53.34 up          0.20
svm_generator_and_switching_timing_87c01ab953_mult6/mult_46_56_mult_1405_modgen_add
_2622_nx2418/O
                LUT4       0.61  53.96 up          0.20
svm_generator_and_switching_timing_87c01ab953_mult6/op_mem_65_20(1)(87)
                LUT4       0.89  54.84 up          0.40
svm_generator_and_switching_timing_87c01ab953_mult6/reg_op_mem_65_20(2)(87)/D
                FDE        0.00  54.84 up          0.00
data arrival time                                   54.84

data arrival time                                   54.84
----------

-- Design summary in file 'xilinx_onduleur_0.sum'
AutoWrite args are : xilinx_onduleur_0.edf
-- Applying renaming rule 'XILINX' to database
Info: setting edif_array_range_extraction_style to %s<%d:%d>
-- Calling set_xilinx_eqn to set up writing Equations
-- Writing file xilinx_onduleur_0.edf
Info, Writing NCF file 'xilinx_onduleur_0.ncf'
-- Writing file xilinx_onduleur_0.ncf
-- CPU time taken for this run was  349.15 sec
-- Run Successfully Ended On  Wed Nov 29 22:42:48 Est (heure d'été) 2006
0
Info: Finished Synthesis run
```