ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Υλοποίηση των Τεχνικών PWM, Sinusoidal PWM και Παραμετρικής Πολυφασικής και Πολυεπίπεδης Space Vector PWM σε FPGA

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Τσακανίκας Θεοφάνης

Επιβλέπων: Γεώργιος Οικονομάκος

Επίκουρος Καθηγητής Ε.Μ.Π

ΑΘΗΝΑ ΙΟΥΛΙΟΣ 2013

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Υλοποίηση των Τεχνικών PWM, Sinusoidal PWM και Παραμετρικής Πολυφασικής και Πολυεπίπεδης Space Vector PWM σε FPGA

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Τσακανίκας Θεοφάνης

Επιβλέπων: Γεώργιος Οικονομάκος

Επίκουρος Καθηγητής Ε.Μ.Π

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 3η Ιουλίου 2013.

.....	.....	.....
Γεώργιος Οικονομάκος	Δημήτριος Σούντρης	Κιαμάλ Πεκμεστζή
Επ. Καθηγητής Ε.Μ.Π	Επ. Καθηγητής Ε.Μ.Π	Καθηγητής Ε.Μ.Π

ΑΘΗΝΑ, ΙΟΥΛΙΟΣ 2013

…….

Τσακανίκας Θεοφάνης

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

ΠΕΡΙΛΗΨΗ

Σκοπός της παρούσας διπλωματικής εργασίας είναι η παρουσίαση τριών τεχνικών διαμόρφωσης εύρους παλμών(Pulse Width Modulation Techniques), οι οποίες χρησιμοποιούνται κυρίως σε εφαρμογές που έχουν να κάνουν με τα ηλεκτρονικά ισχύος, και η υλοποίησή τους με την γλώσσα περιγραφής υλικού VHDL. Η πρώτη τεχνική είναι μία απλή διαμόρφωση εύρους παλμών(Pulse Width Modulation), η οποία αποτελεί αναπόσπαστο κομμάτι των περισσοτέρων ενσωματωμένων συστημάτων. Η δεύτερη τεχνική αποτελεί μία εναλλακτική υλοποίηση της τεχνικής ημιτονοειδούς διαμόρφωσης εύρους παλμών(Sinusoidal Pulse Width Modulation) για τον έλεγχο της τάσης εξόδου ενός αναστροφέα, η οποία υλοποιείται με δύο τρόπους, ενώ η τρίτη τεχνική αποτελεί μία παραμετρική πολυφασική αλλά και πολυεπίπεδη διαμόρφωση εύρους παλμών με χωρικά διανύσματα(Space Vector Pulse Width Modulation). Και οι τρεις αυτές τεχνικές υλοποιούνται με την γλώσσα περιγραφής υλικού VHDL και ελέγχονται με την χρήση ενός Field Programmable Gate Array(FPGA). Πιο συγκεκριμένα παρουσιάζεται μία μέθοδος για την υλοποίηση της PWM τεχνικής η οποία χρησιμοποιεί ένα μετρητή για την επίτευξη του επιθυμητού duty cycle. Επίσης παρουσιάζονται 2 τεχνικές και για την υλοποίηση της SPWM όπου στην μία έχουμε δυναμική παραγωγή ημιτονοειδών δειγμάτων, όποτε αυτό χρειάζεται, ενώ στην άλλη έχουμε αποθηκευμένα 1000 δείγματα ημιτόνου σε μία ROM τα οποία χρησιμοποιούμε για να πετύχουμε τις επιθυμητές συγκρίσεις με ένα τριγωνικό παλμό. Όσον αφορά στην SVPWM παρουσιάζεται ένα παράδειγμα προσομοίωσης που αφορά στο χειρισμό των διακοπτών ενός πενταφασικού, πέντε επιπέδων αναστροφέα και ένα παράδειγμα προσομοίωσης αλλά και πραγματικής υλοποίησης σε FPGA και παρατήρησης στον παλμογράφο, για το χειρισμό των διακοπτών ενός μονοφασικού αναστροφέα τριών επιπέδων. Τα αποτελέσματα των τριών αυτών τεχνικών προσδιορίζονται θεωρητικά, αποδεικνύονται μέσω προσομοιώσεων και εφαρμόζονται σε πραγματικά FPGA μέσω των οποίων παρατηρούνται σε παλμογράφο.

Οι διάφορες προσομοιώσεις γίνονται με την χρήση των εργαλείων προσομοίωσης Modelsim Altera 10.0c(Quartus II 11.1) Starter Edition, PSim και Matlab-Simulink ενώ η ψηφιακή υλοποίηση πάνω στο υλικό γίνεται με την χρήση του ISE Design Suite 14.4 της Xilinx. Το FPGA το οποίο χρησιμοποιήσαμε για τις τεχνικές SPWM και SVPWM ήταν το SPARTAN6 XC6SLX16(Package CSG324), ενώ για την PWM το SPARTAN3E.

Η διπλωματική αυτή χωρίζεται σε πέντε βασικές ενότητες:

1) Μία εισαγωγή στις βασικές έννοιες που διέπουν τα FPGA.
2) Υλοποίηση της PWM τεχνικής.
3) Υλοποίηση των SPWM τεχνικών.
4) Υλοποίηση της παραμετρικής πολυεπίπεδης και πολυφασικής SVPWM.
5) Συμπεράσματα.

Λέξεις Κλειδιά: Ημιτονοειδής διαμόρφωση εύρους παλμών, υπολογισμός δειγμάτων ημιτόνου, διαμόρφωση εύρους παλμών με χωρικά διανύσματα τάσης, παραμετρική σχεδίαση, πολυεπίπεδοι και πολυφασικοί μετατροπείς, γλώσσα περιγραφής υλικού VHDL, FPGA.

ABSTRACT

The scope of this diploma thesis is the presentation of three Pulse Width Modulation(PWM) techniques, which are mainly used in power electronic applications, and their implementation using the hardware descriptive language VHDL. The first technique is a simple Pulse Width Modualtion(PWM) technique, that is an integral part of most embedded systems. The second technique is an alternative implementation of the Sinusoidal Pulse Width Modulation(SPWM) technique used to control the output voltage of an inverter, which is implemented in two ways, while the third one is a parametric, multiphase and multilevel Space Vector Pulse Width Modulation(SVPWM) technique. These three techniques are implemented with the hardware descriptive language VHDL and are being controlled with the use of a Field Programmable Gate Array(FPGA). More specifically, one method is illustrated to implement the PWM technique which uses a counter in order to achieve the desired duty cycle. Also there are presented two ways in order to implement the SPWM technique where in the first one we produce dynamic sinusoidal samples when needed, while on the other one we have stored 1000 samples of a sinusoidal wave in a ROM memory which we use in order to achieve the desired comparisons with a triangular wave. Regarding SVPWM an example of handling the switches of a five – phase, five – level inverter is presented as well as another example that has to do with the handling of a single phase inverter circuit of three levels. The results of these three techniques, are identified theoretically, are proved through simulations and are applied to real FPGAs through which are being observed with the use of an oscilloscope.

The various simulations are done using the simulation tools Modelsim Altera 10.0c (Quartus II 11.1) Starter Edition, PSim and Matlab-Simulink and the digital implementation on the material is achieved with the use of the ISE Design Suite 14.4 Xilinx. The FPGA which was used for the techniques SPWM and SVPWM was SPARTAN6 XC6SLX16 (Package CSG324), while for the PWM technique we used SPARTAN3E.

This diploma thesis is divided into five main sections:

1) An introduction to the basic concepts of FPGAs.
2) Implementation of the PWM technique.
3) Implementation of the SPWM technique.
4) Implementation of the parametric, multilevel and multiphase SVPWM.
5) Conclusions.

Key Words: Sinusoidal pulse width modulation, computation of sinusoidal samples, pulse width modulation with space vectors, parametric design, multilevel and multiphase converters, hardware description language VHDL, FPGA.

ΠΡΟΛΟΓΟΣ

Η παρούσα διπλωματική εργασία εκπονείται στα πλαίσια της πενταετούς υποχρεωτικής φοίτησης στη σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών του Ε.Μ.Π. Σκοπός της είναι η υλοποίηση τριών τεχνικών διαμόρφωσης εύρους παλμών, που αφορούν σε χειρισμό διακοπτών μετατροπέων οι οποίοι βρίσκουν εφαρμογή στον τομέα των ηλεκτρονικών ισχύος, με τη γλώσσα περιγραφής υλικού VHDL και τη χρήση FPGA. Η υλοποίηση των τεχνικών αυτών όπως και η γενικότερη μελέτη πάνω σε θέματα που αφορούν στην εφαρμογή των FPGA στα ηλεκτρονικά ισχύος παρουσιάζει ιδιαίτερο ενδιαφέρον για τους ηλεκτρολόγους μηχανικούς, καθώς συνδυάζει άμεσα δύο επιστημονικά πεδία: εκείνο της ενέργειας & των ηλεκτρονικών ισχύος με αυτό των μικροϋπολογιστών και των ψηφιακών συστημάτων. Ιδιαίτερα αυξημένο είναι το ενδιαφέρον τον τελευταίο καιρό λόγω της αυξανόμενης από τη μία εισαγωγής των FPGA σε ενσωματωμένα συστήματα αλλά και των δυνατοτήτων από την άλλη που προσφέρει ο τομέας της ενέργειας κυρίως σε θέματα που αφορούν την εξοικονόμηση και τα οποία μπορούν να φέρουν πολλαπλά κέρδη όχι μόνο οικονομικά αλλά και για το ίδιο το περιβάλλον. Οι τεχνικές οι οποίες εξετάστηκαν στην παρούσα εργασία, δοκιμάστηκαν τόσο σε επίπεδο προσομοίωσης όσο και πειραματικής επιβεβαίωσης στο εργαστήριο Μικροϋπολογιστών και Ψηφιακών Συστημάτων του Ε.Μ.Π.

Στο πρώτο κεφάλαιο αρχικά αναφέρονται κάποια γενικά στοιχεία που αφορούν στην αρχιτεκτονική, στα πλεονεκτήματα και μειονεκτήματα αλλά και στη διαδικασία σχεδίασης των FPGA. Στη συνέχεια γίνεται αναφορά στη χρήση των ψηφιακών επεξεργαστών σήματος αλλά και των FPGA στα ηλεκτρονικά ισχύος και παρουσιάζεται ένα παράδειγμα που αναφέρεται στη σύγκριση των αποδόσεών τους. Τέλος γίνεται αναφορά σε βασικά στοιχεία που αφορούν στη γλώσσα περιγραφής υλικού VHDL.

Στο δεύτερο κεφάλαιο, παρουσιάζεται η πρώτη τεχνική διαμόρφωσης εύρους παλμών. Αυτή είναι μία απλή PWM(Pulse Width Modulation) τεχνική, η οποία παράγει, ανάλογα με την είσοδο που λαμβάνει, τους επιθυμητούς παλμούς στην έξοδο.

Στο τρίτο κεφάλαιο μελετάται η δεύτερη τεχνική διαμόρφωσης εύρους παλμών. Αυτή είναι η SPWM(Sinusoidal Pulse Width Modulation) τεχνική, η οποία παράγει τους παλμούς χειρισμού των τεσσάρων διακοπτών ενός μονοφασικού, στην προκειμένη περίπτωση, αναστροφέα βασιζόμενη στη σύγκριση ενός ημιτονοειδούς με ένα τριγωνικό σήμα. Η τεχνική αυτή πραγματοποιείται με δύο τρόπους οι οποίοι και συγκρίνονται με βάση την ακρίβεια των αποτελεσμάτων αλλά και τους απαιτούμενους πόρους που δεσμεύουν από το FPGA για την υλοποίησή τους.

Στο τέταρτο κεφάλαιο παρουσιάζεται η τρίτη και αποδοτικότερη τεχνική διαμόρφωσης εύρους παλμών. Αυτή είναι η SVPWM(Space Vector Pulse Width Modulation) τεχνική και η οποία αφορά στο χειρισμό των διακοπτών πολυφασικών αλλά και πολυεπίπεδων μετατροπέων με τη χρήση χωρικών διανυσμάτων. Η υλοποίηση αυτή είναι επίσης και παραμετρική καθώς δίνεται η δυνατότητα προσαρμογής του συστήματος σε οποιοδήποτε αριθμό φάσεων και επιπέδων. Στο κεφάλαιο αυτό παρουσιάζεται αναλυτικά ένας αλγόριθμος για την τεχνική SVPWM, μέσω του οποίου επιτυγχάνεται

πέρα από την ακριβή προσέγγιση του σήματος αναφοράς, μέσω της ρυθμιζόμενης αγωγής των ημιαγωγικών διακοπτών, και η ελαχιστοποίηση των διακοπτικών απωλειών.

Στο πέμπτο κεφάλαιο γίνεται μία σύνοψη των κυριότερων συμπερασμάτων που αφορούν στις τρεις προαναφερθείσες τεχνικές, και παρατίθενται κάποιες προτάσεις για μελλοντική έρευνα και εργασία.

ΕΥΧΑΡΙΣΤΙΕΣ

Στο πλαίσιο της παρούσας διπλωματικής εργασίας, θέλω να εκφράσω της θερμές μου ευχαριστίες στον επιβλέποντα καθηγητή κ. Οικονομάκο Γεώργιο για την ευκαιρία που μου έδωσε να ασχοληθώ με ένα τόσο ενδιαφέρον και πρωτότυπο θέμα. Τον ευχαριστώ για την έμπρακτη εμπιστοσύνη που έδειξε στο πρόσωπό μου, την πάντα πρόθυμη και έγκαιρη βοήθειά του αλλά και την άριστη συνεργασία μας. Επίσης θέλω να ευχαριστήσω ιδιαιτέρως την οικογένειά μου και το φιλικό μου περιβάλλον, καθώς στάθηκαν δίπλα μου, με στήριξαν, και συνέβαλαν τα μέγιστα τόσο στην ολοκλήρωση της εργασίας όσο και καθ' όλη τη διάρκεια των σπουδών μου.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1ο:

ΒΑΣΙΚΕΣ ΓΝΩΣΕΙΣ ΚΑΙ ΔΙΑΔΙΚΑΣΙΑ ΣΧΕΔΙΑΣΗΣ ΣΕ FPGA

1.1 Εισαγωγή……23

1.2 Περιγραφή των FPGA……23

1.3 Πλεονεκτήματα και μειονεκτήματα των FPGA……26

1.3.1 Πλεονεκτήματα των FPGA……26

1.3.2 Μειονεκτήματα των FPGA……27

1.4 Ροή Σχεδιασμού FPGA……28

1.4.1 Είσοδος Σχεδίασης……29

1.4.2 Προσομοίωση Συμπεριφοράς……29

1.4.3 Σύνθεση Σχεδιάσης……29

1.4.4 Υλοποίηση Σχεδίασης……29

1.4.5 Προγραμματισμός της Xilinx συσκευής(FPGA)……30

1.4.6 Ρύθμιση συσκευής στόχου……30

1.4.7 Αντιστοίχιση ακροδεκτών FPGA με σήματα εισόδου εξόδου……32

1.5 Τα FPGA στα ηλεκτρονικά ισχύος……32

1.5.1 Τα DSP στα ηλεκτρονικά ισχύος……32

1.5.2 FPGA έναντι DSP στα ηλεκτρονικά ισχύος……33

1.6 Η γλώσσα περιγραφής υλικού VHDL……35

1.6.1 Πλεονεκτήματα της VHDL……36

1.6.2 Μειονεκτήματα της VHDL……36

ΚΕΦΑΛΑΙΟ 2ο:

ΥΛΟΠΟΙΗΣΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΤΗΣ PWM ΤΕΧΝΙΚΗΣ

2.1 Εισαγωγή……37

2.2 Ψηφιακές τεχνικές PWM παραγωγής……37

2.3 Υψηλής συχνότητας PWM γεννήτρια βασισμένη σε μετρητή……38

2.3.1 Αρχιτεκτονική σχεδίασης……39

2.3.2 VHDL κώδικας αρχιτεκτονικής……39

2.3.3 Προσομοίωση στο Modelsim……41

2.3.4 RTL Schematic στο Xilinx ISE ……42

2.3.5 Αναφορά χρησιμοποίησης συσκευής……43

2.4 Πειραματική διαδικασία……44

ΚΕΦΑΛΑΙΟ 3ο:

ΥΛΟΠΟΙΗΣΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΤΗΣ SPWM ΤΕΧΝΙΚΗΣ

3.1 Εισαγωγή……47

3.2 Προσομοίωση της SPWM τεχνικής με το PSim……48

3.3 Ψηφιακή υλοποίηση της SPWM τεχνικής……50

3.4 SPWM τεχνική με χρήση μνήμης ROM……61

3.4.1 Παραγωγή Δειγμάτων Ημιτόνου……61

3.4.2 Παραγωγή Παλμών Διακοπτών……67

3.4.3 Αποτελέσματα προσομοίωσης σχεδίασης……70

3.4.4 Αποτελέσματα πειραματικής διαδικασίας……74

3.5 SPWM τεχνική με δυναμικό υπολογισμό δειγμάτων……82

3.5.1 Σύστημα Xilinx Core Generator……83

3.5.2 Ανάλυση VHDL κώδικα σχεδίασης……83

3.5.3 Αποτελέσματα προσομοίωσης σχεδίασης……88

3.5.4 Αποτελέσματα πειραματικής διαδικασίας……93

3.6 Σύγκριση των δύο SPWM τεχνικών……96

ΚΕΦΑΛΑΙΟ 4ο:

ΥΛΟΠΟΙΗΣΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΤΗΣ ΠΑΡΑΜΕΤΡΙΚΗΣ, ΠΟΛΥΦΑΣΙΚΗΣ ΚΑΙ ΠΟΛΥΕΠΙΠΕΔΗΣ SVPWM ΤΕΧΝΙΚΗΣ

4.1 Εισαγωγή……99

4.2 Η τεχνική SVPWM……100

4.3 Υλοποίηση αλγορίθμου……102

4.3.1 Παραμετρική σχεδίαση, ανεξάρτητη τεχνολογίας……105

4.3.2 Αποσύνθεση κανονικοποιημένου σήματος αναφοράς……109

4.3.3 Ταξινόμηση κλασματικού μέρους αναφοράς……112

4.3.4 Υπολογισμός πίνακα D……116

4.3.5 Εξαγωγή μετατοπισμένων διανυσμάτων μεταγωγής……122

4.3.6 Υπολογισμός χρόνων μεταγωγής……125

4.3.7 Υπολογισμός διανυσμάτων μεταγωγής……127

4.4 Προσομοίωση και πειραματική διαδικασία της διφασικής SVPWM τεχνικής……133

4.4.1 Αποτελέσματα προσομοίωσης σχεδίασης……133

4.4.2 Αναφορά χρησιμοποίησης συσκευής……..136

4.4.3 Αποτελέσματα πειραματικής διαδικασίας……138

4.5 Υλοποίηση της SVPWM με το Matlab Simulink……..140

ΚΕΦΑΛΑΙΟ 5^{ο}:

ΣΥΝΟΨΗ, ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΗ ΕΡΓΑΣΙΑ

5.1 Σύνοψη και συμπερασματα……..142

5.2 Μελλοντική εργασία……..144

Παράρτημα……145

Βιβλιογραφία……167

ΠΕΡΙΕΧΟΜΕΝΑ ΣΧΗΜΑΤΩΝ

Σχήμα 1.1: Αρχιτεκτονική FPGA……23

Σχήμα 1.2: Εσωτερική Αρχιτεκτονική FPGA……24

Σχήμα 1.3: Διάγραμμα Ευελιξίας – Απόδοσης……25

Σχήμα 1.4: Ροή Σχεδίασης FPGA……28

Σχήμα 1.5: Πλακέτα SPARTAN-3E Starter Kit (FPGA)……31

Σχήμα 1.6: Πλακέτα SPARTAN6 XC6SLX16 FPGA……31

Σχήμα 1.7: Μπλοκ Διάγραμμα ενός τυπικού DSP……33

Σχήμα 1.8: Σύγκριση αποδόσεων του Spartan 6 FPGA με ένα διπύρηνο επεξεργαστή……34

Σχήμα 2.1: Γενικό Μπλοκ Διάγραμμα Ψηφιακού Ελέγχου Μετατροπέα……38

Σχήμα 2.2: Αρχιτεκτονική της PWM Γεννήτριας……39

Σχήμα 2.3: Σήματα PWM Σχεδίασης……41

Σχήμα 2.4: Για DUTY CYCLE = 001000000 => DUTY CYCLE = 25%……41

Σχήμα 2.5: Για DUTY CYCLE = 010000000 => DUTY CYCLE = 50%……42

Σχήμα 2.6: Για DUTY CYCLE = 011000000 => DUTY CYCLE = 75%……42

Σχήμα 2.7: RTL σχηματική αναπαράσταση αρχιτεκτονικής……43

Σχήμα 2.8: Υλοποίηση αρχιτεκτονικής στο FPGA Spartan 3E και απεικόνιση στον παλμογράφο……44

Σχήμα 2.9: Αποτελέσματα πειραματικής διαδικασίας για duty cycle 50%……45

Σχήμα 2.10: Αποτελέσματα πειραματικής διαδικασίας για duty cycle 75%……46

Σχήμα 3.1: Μονοφασικός αναστροφέας πλήρους γέφυρας με πηγή τάσης υλοποιημένος με ημιαγωγικούς διακόπτες τύπου IGBTs……47

Σχήμα 3.2: Κυκλωματικό ισοδύναμο της τεχνικής SPWM με το εργαλείο προσομοίωσης PSim……..48

Σχήμα 3.3: Αποτελέσματα προσομοίωσης του κυκλώματος της SPWM τεχνικής με το εργαλείο προσομοίωσης Psim……49

Σχήμα 3.4: Ψηφιακή υλοποίηση του τριγωνικού παλμού……55

Σχήμα 3.5: Δημιουργία βραχυκυκλώματος εξ' αιτίας της μη ιδανικής συμπεριφοράς των ημιαγωγικών διακοπτών όταν τίθενται σε αγωγή ή αποκοπή……..57

Σχήμα 3.6: Βραχυκύκλωμα λόγω εισαγωγής του ''νεκρού'' χρονικού διαστήματος στον διακόπτη που μεταβαίνει από αγωγή σε αποκοπή……..57

Σχήμα 3.7: Μονοφασικός αναστροφέας πλήρους γέφυρας με πηγή τάσης υλοποιημένος με ημιαγωγικούς διακόπτες τύπου IGBTs……58

Σχήμα 3.8: Τιμές Αρχικών Δειγμάτων……65

Σχήμα 3.9: Τιμή Ημιτόνου $500^{ου}$ Δείγματος……66

Σχήμα 3.10: Τιμές Τελευταίων Δειγμάτων……66

Σχήμα 3.11: Παλμοί διακοπτών στην αρχή της ημιτονοειδούς περιόδου με το εργαλείο προσομοίωσης Modelsim……70

Σχήμα 3.12: Παλμοί των διακοπτών στην αρχή ημιτονοειδούς περιόδου με το εργαλείο προσομοίωσης PSim……71

Σχήμα 3.13: Παλμοί διακοπτών για το T/4 της ημιτονοειδούς περιόδου με το εργαλείο προσομοίωσης Modelsim……71

Σχήμα 3.14: Παλμοί διακοπτών για το T/2 της ημιτονοειδούς περιόδου με το εργαλείο προσομοίωσης Modelsim……72

Σχήμα 3.15: Παλμοί διακοπτών για το 3T/4 της ημιτονοειδούς περιόδου με το εργαλείο προσομοίωσης Modelsim……73

Σχήμα 3.16: Παλμοί διακοπτών για το τέλος της ημιτονοειδούς περιόδου με το εργαλείο προσομοίωσης Modelsim……73

Σχήμα 3.17: ''Νεκρό'' Χρονικό Διάστημα……74

Σχήμα 3.18: Υλοποίηση αρχιτεκτονικής στο FPGA Spartan 6 XC6LX16-CS324 και απεικόνιση στον παλμογράφο ……75

Σχήμα 3.19: Απεικόνιση στον παλμογράφο των αποτελεσμάτων για τους παλμούς οδήγησης των διακοπτών για ενεργοποιημένο το reset……77

Σχήμα 3.20: Απεικόνιση στον παλμογράφο των αποτελεσμάτων για τους παλμούς οδήγησης των διακοπτών για απενεργοποιημένο το reset……78

Σχήμα 3.21: Απεικόνιση στον παλμογράφο του συμπληρωματικού ζεύγους παλμών S1,S2 της SPWM τεχνικής με χρήση ROM για εύρος παλμών περίπου 50%.....79

Σχήμα 3.22: Απεικόνιση στον παλμογράφο του συμπληρωματικού ζεύγους παλμών S1,S2 της SPWM τεχνικής με χρήση ROM για εύρος παλμών S1 περίπου 75% και S2 25%.....80

Σχήμα 3.23: Απεικόνιση στον παλμογράφου του συμπληρωματικού ζεύγους παλμών S3-S4 της SPWM τεχνικής με χρήση ROM……81

Σχήμα 3.24: Απεικόνιση στον παλμογράφο του συμπληρωματικού ζεύγους παλμών S1,S2 της SPWM τεχνικής με χρήση ROM για εύρος παλμών S3 περίπου 25% και S2 75%.....82

Σχήμα 3.25: $1^{η}$ Περίοδος Λειτουργίας……89

Σχήμα 3.26: $2^{η}$ Περίοδος λειτουργίας……89

Σχήμα 3.27: Παλμοί διακοπτών στην αρχή της ημιτονοειδούς περιόδου με το εργαλείο προσομοίωσης Modelsim……90

Σχήμα 3.28: Παλμοί διακοπτών για το T/4 της ημιτονοειδούς περιόδου με το εργαλείο προσομοίωσης Modelsim……91

Σχήμα 3.29: Παλμοί διακοπτών για το T/2 της ημιτονοειδούς περιόδου με το εργαλείο προσομοίωσης Modelsim……91

Σχήμα 3.30: Παλμοί διακοπτών για το 3T/4 της ημιτονοειδούς περιόδου με το εργαλείο προσομοίωσης Modelsim……92

Σχήμα 3.31: Παλμοί διακοπτών για το τέλος της ημιτονοειδούς περιόδου με το εργαλείο προσομοίωσης Modelsim……92

Σχήμα 3.32: ‘’Νεκρό’’ Χρονικό Διάστημα……93

Σχήμα 3.33: Απεικόνιση στον παλμογράφο του συμπληρωματικού ζεύγους παλμών S1-S2 της SPWM τεχνικής με δυναμικό υπολογισμό δειγμάτων……95

Σχήμα 3.34: Απεικόνιση στον παλμογράφο του συμπληρωματικού ζεύγους παλμών S3-S4 της SPWM τεχνικής με δυναμικό υπολογισμό δειγμάτων……96

Σχήμα 4.1: Τριφασικός αναστροφέας με πηγή τάσης και IGBTs ημιγωγικούς διακόπτες……..101

Σχήμα 4.2: Μετασχηματισμός από το abc στο dq σύστημα συντεταγμένων……..102

Σχήμα 4.3: Χωρικά διανύσματα φασικών τάσεων φορτίου του αναστροφέα δύο επιπέδων στο επίπεδο d-q……..102

Σχήμα 4.4: Μπλοκ διάγραμμα της πολυφασικής πολυεπίπεδης SVPWM. **(a)** Πολυφασική πολυεπίπεδη SVPWM βασισμένη σε μία δύο επιπέδων SVPWM. **(b)** Μπλοκ διάγραμμα της δύο επιπέδων πολυφασικής SVPWM.....105

Σχήμα 4.5: Διάγραμμα ροής της πολυφασικής πολυεπίπεδης SVPWM μονάδας……108

Σχήμα 4.6: Παράδειγμα Αποσύνθεσης στον Δισδιάστατο Χώρο……..109

Σχήμα 4.7: Αποτελέσματα προσομοίωσης κυκλώματος Vr_dec με το Modelsim……112

Σχήμα 4.8: Αποτελέσματα προσομοίωσης κυκλώματος Vf_sort με το Modelsim……115

Σχήμα 4.9: Αποτελέσματα προσομοίωσης κυκλώματος D_calc με το Modelsim……..121

Σχήμα 4.10: Αποτελέσματα προσομοίωσης κυκλώματος Vd_extr με το Modelsim……124

Σχήμα 4.11: Αποτελέσματα προσομοίωσης κυκλώματος t_calc με το Modelsim……127

Σχήμα 4.12: Αποτελέσματα προσομοίωσης κυκλώματος Vs_calc με το Modelsim……129

Σχήμα 4.13: Ένα σκέλος μίας φάσης ενός αναστροφέα με διόδους περιορισμού m επιπέδων τάσης(DCMI) ……..131

Σχήμα 4.14: Τριφασικός αναστροφέας FCMI πέντε επιπέδων……..132

Σχήμα 4.15: Κύκλωμα ισχύος μίας φάσης SDCSMI αναστροφέα που έχει υλοποιηθεί από μονάδες μονοφασικού αναστροφέα πλήρους γέφυρας……..133

Σχήμα 4.16: Παλμοί διακοπτών S1,S3 του μονοφασικού αναστροφέα με την τεχνική SVPWM μέσω του εργαλείου προσομοίωσης Modelsim……135

Σχήμα 4.17: Παλμοί διακοπτών S1,S2,S3,S4 της τεχνικής SVPWM για μονοφασικό αναστροφέα μέσω του εργαλείου προσομοίωσης……136

Σχήμα 4.18: Απεικόνιση στον παλμογράφο του συμπληρωματικού ζεύγους παλμών S1-S2 της SVPWM τεχνικής……..139

Σχήμα 4.19: Απεικόνιση στον παλμογράφο του συμπληρωματικού ζεύγους παλμών S3-S4 της SVPWM τεχνικής……..140

Σχήμα 4.20: Μπλοκ Διάγραμμα της Πολυφασικής Πολυεπίπεδης SVPWM……..141

Σχήμα 4.21: Simulink μοντέλο της υλοποίησης στο υλικό του SVPWM αλγορίθμου……..141

ΠΕΡΙΕΧΟΜΕΝΑ ΠΙΝΑΚΩΝ

Πίνακας 1.1: Βαθμοί απόδοσης Spatan6 FPGA / Dual Core DSP……..34

Πίνακας 1.2: Διαφορές ανάμεσα στα FPGA και στα DSP……..34

Πίνακας 2.1: Χρησιμοποιούμενοι πόροι της συσκευής για την υλοποίηση της υψηλής συχνότητας PWM τεχνικής……..43

Πίνακας 3.1: Χρησιμοποιούμενοι πόροι της συσκευής για την υλοποίηση της SPWM τεχνικής με χρήση μνήμης ROM……..97

Πίνακας 3.2: Χρησιμοποιούμενοι πόροι της συσκευής για την υλοποίηση της SPWM τεχνικής με δυναμικό υπολογισμό δειγμάτων……..98

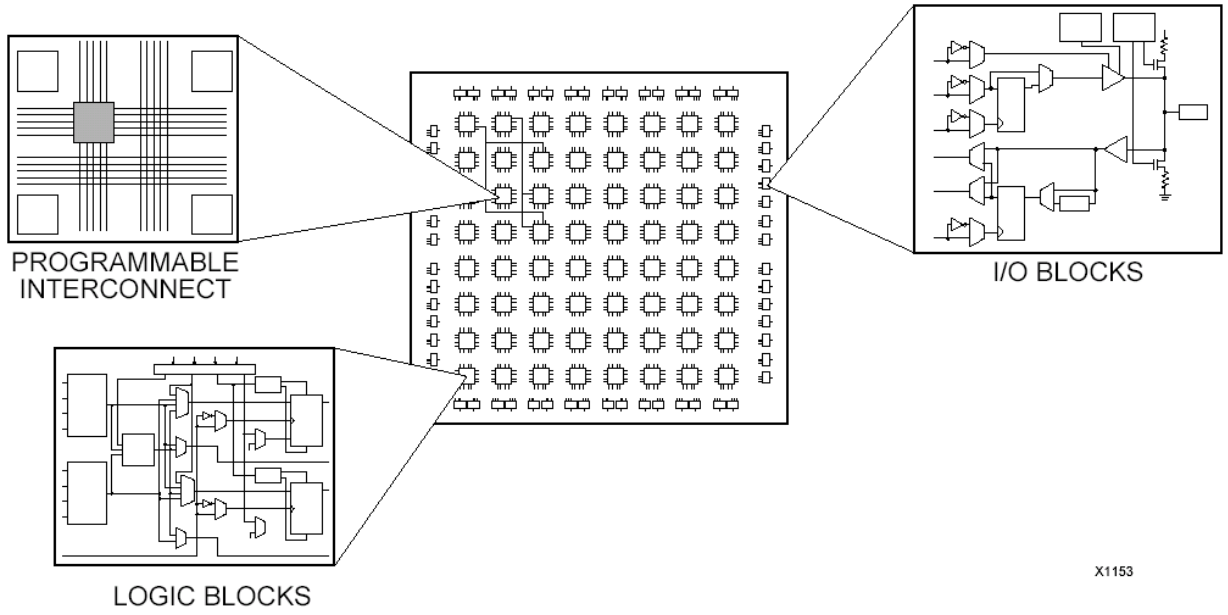
Πίνακας 4.1: Χρησιμοποιούμενοι πόροι της συσκευής για την υλοποίηση της διφασικής SVPWM τεχνικής……..137

ΚΕΦΑΛΑΙΟ 1^{ο}

ΒΑΣΙΚΕΣ ΓΝΩΣΕΙΣ ΚΑΙ ΔΙΑΔΙΚΑΣΙΑ ΣΧΕΔΙΑΣΗΣ ΣΕ FPGA

1.1 Εισαγωγή

Το FPGA (Field Programmable Gate Array) , όπως δηλώνει και το όνομά του, είναι μία σειρά από λογικά κύτταρα και διασυνδέσεις, τα οποία μπορούν να επαναπρογραμματιστούν ανάλογα με τις απαιτήσεις του χρήστη. Μπορούμε να το σχεδιάσουμε και να κάνουμε αλλαγές σε αυτό όποτε κάτι τέτοιο απαιτείται. Παρέχει άμεση μεταστροφή κατασκευής και αμελητέο κόστος πρωτοτύπου κάτι το οποίο το καθιστά κατάλληλο για τον σχεδιασμό ενσωματωμένων συστημάτων. Σχηματικά η δομή ενός FPGA παρουσιάζεται παρακάτω:



Σχήμα 1.1: Αρχιτεκτονική FPGA

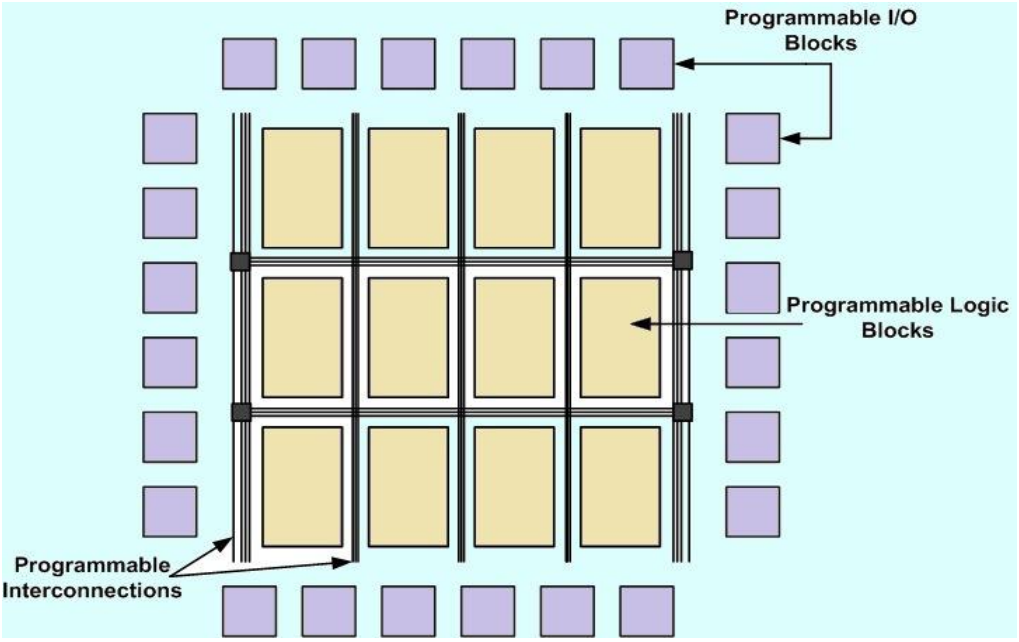
Στο κεφάλαιο αυτό, ασχολούμαστε με την περιγραφή των βασικών συστατικών ενός FPGA, με τα πλεονεκτήματα και μειονεκτήματα τα οποία παρουσιάζει όπως και με την χρησιμοποίηση των FPGA στα ηλεκτρονικά ισχύος το οποίο και αποτελεί το κύριο θέμα της παρούσας εργασίας.

1.2 Περιγραφή των FPGA

Η βασική αρχιτεκτονική των FPGA αποτελείται από τρία κύρια στοιχεία:

1) Προγραμματιζόμενα λογικά μπλοκ, τα οποία υλοποιούν τις λογικές συναρτήσεις.
2) Προγραμματιζόμενες διασυνδέσεις για την υλοποίηση αυτών των συναρτήσεων.
3) Μπλοκ εισόδου/εξόδου για την πραγματοποίηση συνδέσεων εκτός του chip.

Μία παρουσίαση της τυπικής αρχιτεκτονικής ενός FPGA παρουσιάζεται παρακάτω:



Σχήμα 1.2: Εσωτερική Αρχιτεκτονική FPGA

Παρακάτω αναλύονται τα τρία κύρια στοιχεία που αφορούν στην αρχιτεκτονική των FPGA και τα οποία αναφέραμε παραπάνω:

1) Προγραμματιζόμενα Λογικά Μπλοκ:

Ο σκοπός των προγραμματιζόμενων αυτών μπλοκ σε ένα FPGA είναι να παρέχουν τα βασικά υπολογιστικά και αποθηκευτικά στοιχεία τα οποία είναι απαραίτητα στα ψηφιακά συστήματα. Τα βασικά λογικά στοιχεία περιλαμβάνουν κάποιου είδους προγραμματιζόμενης συνδυαστικής λογικής, flip-flops ή μανδαλωτές και κάποια γρήγορη λογική κρατουμένου για την μείωση του κόστους της επιφάνειας αλλά και της καθυστέρησης. Επίσης, εκτός από τα βασικά λογικά μπλοκ, πολλά σύγχρονα FPGA περιέχουν ένα ετερογενές μείγμα από διαφορετικά μπλοκ, μερικά από τα οποία μπορούν να χρησιμοποιηθούν μόνο για συγκεκριμένες συναρτήσεις, όπως μπλοκ αποκλειστικά για μνήμη αλλά και πολλαπλασιαστές ή πολυπλέκτες. Βεβαίως, προσαρμοζόμενη μνήμη χρησιμοποιείται και μέσα στα λογικά μπλοκ για τον έλεγχο της συγκεκριμένης συνάρτησης του κάθε στοιχείου μέσα στο μπλοκ.

2) Προγραμματιζόμενες Διασυνδέσεις:

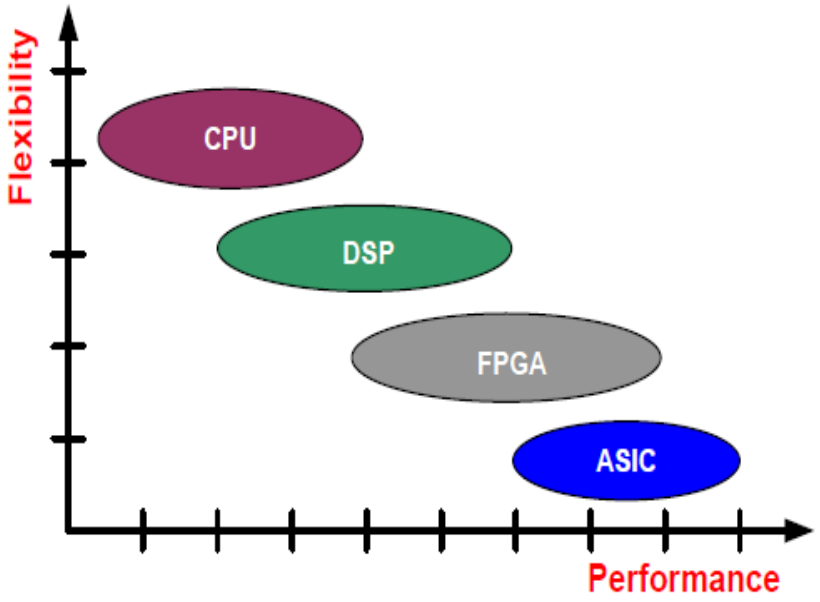
Οι προγραμματιζόμενες αυτές συνδέσεις παρέχουν σε ένα FPGA την διασύνδεση ανάμεσα στα λογικά αλλά και στα εισόδου/εξόδου μπλοκ με σκοπό την ολοκλήρωση της ορισμένης από το χρήστη σχεδίασης. Αποτελούνται από πολυπλέκτες, τρανζίστορ διέλευσης και buffers τριών καταστάσεων τα οποία υλοποιούν τις επιθυμητές συνδέσεις. Γενικά, τα τρανζίστορ διέλευσης αλλά και οι πολυπλέκτες

χρησιμοποιούνται κατά μήκος ενός λογικού συμπλέγματος για να συνδέσουν τα λογικά στοιχεία μαζί ενώ και τα τρία προαναφερθέντα στοιχεία χρησιμοποιούνται για τη σύνδεση πιο γενικών δομών.

3) Προγραμματιζόμενες Είσοδοι/Εξοδοι:

Το κομμάτι αυτό αναφέρεται στα απαραίτητα μέσα για την διεπαφή ανάμεσα στα λογικά μπλοκ και τις τεχνικές διασύνδεσης και στα εξωτερικά στοιχεία του FPGA. Αυτά τα στοιχεία είναι σημαντικά για το FPGA και καταλαμβάνουν ένα σημαντικό μέρος(περίπου το 40%) της επιφάνειάς του. Ο σχεδιασμός των προγραμματιζόμενων μπλοκ εισόδου/εξόδου αποτελεί πρόκληση καθώς υπάρχει μεγάλη ποικιλία όσον αφορά στα πρότυπα της τάσης τροφοδοσίας αλλά και της τάσης αναφοράς. Μία από τις σημαντικότερες αποφάσεις στη σχεδίαση της αρχιτεκτονικής εισόδου/εξόδου, είναι η επιλογή των πρωτοτύπων που θα υποστηριχτούν. Αυτό περιλαμβάνει προσεχτικά επιλεγμένα trade-offs. Η υποστήριξη πολλών πρωτοτύπων μπορεί να αυξήσει την επιφάνεια του πυριτίου που απαιτείται για τα κύτταρα εισόδου/εξόδου σημαντικά.

Εκτός αυτών αξίζει να σημειώσουμε πως τα FPGA, λόγω της δυνατότητας προγραμματισμού των συνδέσεων αλλά και των λογικών μπλοκ που προσφέρουν, παρουσιάζουν μεγάλη ευελιξία σε σχέση με τα ολοκληρωμένα κυκλώματα ειδικού σκοπού, καθώς επιτυγχάνουν μεγαλύτερες συχνότητες λειτουργίας από τις CPU ή DSP συσκευές. Αυτό φαίνεται και στο παρακάτω γράφημα:



Σχήμα 1.3: Διάγραμμα Ευελιξίας - Απόδοσης

1.3 Πλεονεκτήματα και μειονεκτήματα των FPGA

Τα FPGA παρέχουν στις σχεδιάσεις και εφαρμογές μας πληθώρα πλεονεκτημάτων αλλά παρουσιάζουν και σημαντικά μειονεκτήματα συγκρινόμενα με τους επεξεργαστές ψηφιακού σήματος. Τα κυριότερα από τα πλεονεκτήματα και τα μειονεκτήματα αυτά παρουσιάζονται παρακάτω.

1.3.1 Πλεονεκτήματα των FPGA

Τα FPGA (Field Programmable Gate Arrays) είναι μία ειδική κατηγορία ολοκληρωμένων κυκλωμάτων. Τα κύρια πλεονεκτήματα που προσφέρουν τα FPGA είναι:

1) **Απόδοση:** Λόγω του hardware παραλληλισμού, τα FPGA υπερβαίνουν την υπολογιστική δύναμη των DSP, σπάζοντας το πρότυπο της ακολουθιακής εκτέλεσης και επιτυγχάνοντας περισσότερους κύκλους ανά παλμό του ρολογιού. Επίσης ελέγχοντας τις εισόδους και τις εξόδους στο hardware επίπεδο πετυχαίνουμε γρηγορότερους χρόνους απόκρισης και εξειδικευμένη λειτουργία για να ταιριάζει σχεδόν ακριβώς στις απαιτήσεις του συστήματος.

2) **Time to Market:** Η FPGA τεχνολογία προσφέρει ευελιξία και ταχεία προτυποποίηση για να ανταπεξέλθει στις προσδοκίες της αγοράς. Μπορούμε να τεστάρουμε μία ιδέα και να την επαληθεύσουμε στο hardware χωρίς να μπούμε στην μακρά διαδικασία της ASIC σχεδίασης. Μπορούμε μετά να υλοποιήσουμε σταδιακές αλλαγές και να επανελέγξουμε την FPGA σχεδίαση μέσα σε ώρες αντί για εβδομάδες. Είναι επίσης διαθέσιμο Commersial Off The Self (COTS) hardware με διαφορετικού τύπου εισόδους – εξόδους οι οποίες είναι ήδη ενωμένες στο FPGA chip. Τέλος η αυξανόμενη διαθεσιμότητα software εργαλείων υψηλού επιπέδου μειώνει την απαραίτητη προγραμματιστική γνώση με αρκετά επίπεδα αφαίρεσης και συχνά προσφέρει IP Cores (προκατασκευασμένες συναρτήσεις) για προηγμένο έλεγχο και ανάλυση σήματος.

3) Κόστος: Το NRE (Non Recurring Engineering) κόστος της ASIC σχεδίασης ξεπερνά κατά πολύ αυτό των βασισμένων σε FPGA hardware λύσεων. Το μεγάλο αρχικό κόστος επένδυσης στα ASICs αποσβένεται για εφαρμογές που πουλούν χιλιάδες chip το χρόνο, αλλά κάποιοι χρήστες χρειάζονται συγκεκριμένη hardware λειτουργία για δεκάδες ή εκατοντάδες συστήματα. Επίσης η φύση αυτή του προγραμματιζόμενου πυριτίου του FPGA σημαίνει μηδενικό κόστος κατασκευής όπως και συναρμολόγησης. Επειδή οι απαιτήσεις του συστήματος συχνά αλλάζουν με το χρόνο, το κόστος για μικρές αλλαγές του hardware στο FPGA είναι αμελητέο συγκρινόμενο με αυτό της ανακατασκευής του ASIC.

4) **Αξιοπιστία:** Τα Processor – based συστήματα συχνά περιλαμβάνουν αρκετά στρώματα αφαίρεσης για να βοηθήσουν την χρονοδρομολόγηση των εργασιών και τον διαμοιρασμό των πόρων ανάμεσα στις πολλαπλές διαδικασίες. Για κάθε δοσμένο πυρήνα επεξεργαστή, μία μόνο εντολή μπορεί να εκτελεστεί ανά φορά. Στα Processor – based συστήματα είναι συνεχώς σε κίνδυνο οι time – critical εργασίες να επηρεάζουν η μία την άλλη. Τα FPGA από την άλλη ελαχιστοποιούν τους κινδύνους

αξιοπιστίας με την πραγματική παράλληλη εκτέλεση και το ντετερμινιστικό hardware κομμάτι το οποίο είναι αποκλειστικά αφιερωμένο για την κάθε εργασία.

5) Μακροπρόθεσμη Διατήρηση: Τα FPGA chip είναι Field – Upgradable και δεν χρειάζονται τον χρόνο και τα έξοδα που σχετίζονται με την ASIC επανασχεδίαση. Τα ψηφιακά πρωτόκολλα επικοινωνίας, για παράδειγμα, θέτουν στα συστήματα διάφορες προδιαγραφές οι οποίες μπορεί να αλλάζουν με τον καιρό. Άρα οι αναπροσαρμοστικότητα των FPGA τους επιτρέπει να ακολουθούν τις μελλοντικές, απαραίτητες αλλαγές. Τέλος καθώς το προϊόν ή το σύστημα ωριμάζει, μπορούμε να κάνουμε λειτουργικές βελτιώσεις χωρίς να σπαταλούμε χρόνο για την επανασχεδίαση του hardware ή την τροποποίηση του ήδη υπάρχοντος όπως θα απαιτούνταν στα ASIC.

1.3.2 Μειονεκτήματα των FPGA

1) Μεγαλύτερη κατανάλωση ισχύος.

2) Λιγότερο αποδοτική χρήση του πυριτίου.

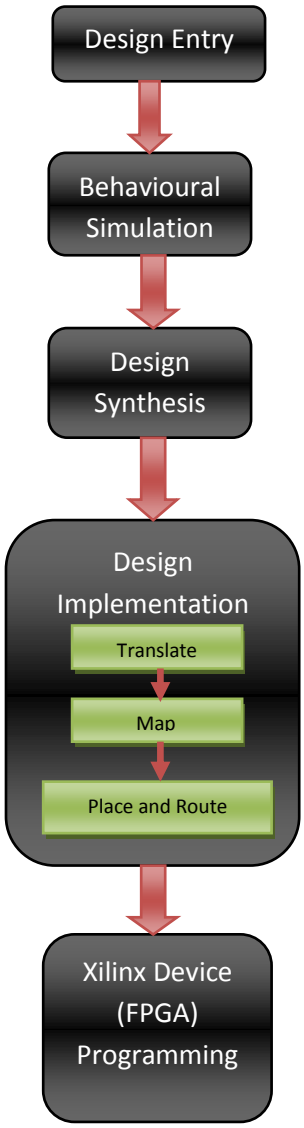
3) Είναι τάξεις μεγέθους πιο δύσκολο να προγραμματιστούν.

4) Το 70% του κόστους ανάπτυξης λογισμικού προέρχεται από την ανάπτυξη της Εισόδου/Εξόδου διεπαφής (αντί από αλγόριθμους).

5) Δεν είναι η καταλληλότερη λύση για εφαρμογές που εμπεριέχουν υπολογισμούς αριθμών κινητής υποδιαστολής. Ακόμα και οι υπολογισμοί με σταθερής υποδιαστολής αριθμούς υψηλής ακρίβειας χρειάζονται μεγάλη ποσότητα λογικών κυττάρων για να υλοποιηθούν.

1.4 ΡΟΗ ΣΧΕΔΙΑΣΜΟΥ FPGA

Στο σημείο αυτό θα ασχοληθούμε συνοπτικά με την διαδικασία που απαιτείται για την πραγματική υλοποίηση της εκάστοτε σχεδίασης σε ένα FPGA. Το παρακάτω σχήμα απεικονίζει την ακολουθία των βημάτων που ακολουθείται για την εφαρμογή μίας οποιασδήποτε σχεδίασης σε ένα FPGA.



Σχήμα 1.4: Ροή Σχεδίασης FPGA

Τα βήματα του παραπάνω σχήματος είναι απαραίτητα για την υλοποίηση της οποιασδήποτε σχεδίασης σε FPGA. Τα βήματα αυτά παρουσιάζονται αναλυτικά παρακάτω.

1.4.1 ΕΙΣΟΔΟΣ ΣΧΕΔΙΑΣΗΣ

Αυτό είναι το πρώτο βήμα για την εφαρμογή μίας σχεδίασης σε ένα FPGA. Σε αυτό το βήμα ο VHDL(Very High Speed Integrated Chip Hardware Description Language) κώδικας της αρχιτεκτονικής της σχεδίασης που θέλουμε να υλοποιήσουμε, γράφεται με την χρήση λογισμικού(Xilinx ISE 14.4 στην προκείμενη περίπτωση). Αφού γραφεί ο κώδικας γίνεται συντακτικός έλεγχος για πιθανά συντακτικά λάθη.

1.4.2 ΠΡΟΣΟΜΟΙΩΣΗ ΣΥΜΠΕΡΙΦΟΡΑΣ

Το επόμενο βήμα είναι η προσομοίωση συμπεριφοράς. Αυτό το βήμα ελέγχει αν η εισαγόμενη σχεδίαση είναι λειτουργικά σωστή ή όχι. Αυτή η προσομοίωση ονομάζεται RTL(Register Transfer Level) προσομοίωση. Γι αυτή την προσομοίωση γράφεται ένα Testbench σε VHDL για την αρχιτεκτονική που έχουμε χρησιμοποιήσει και η προσομοίωση φαίνεται στο Xilinx ISE Simulator. Αφού ελεγχθεί πως είναι λειτουργικά σωστή περνάμε στο επόμενο βήμα.

1.4.3 ΣΥΝΘΕΣΗ ΣΧΕΔΙΑΣΗΣ

Ο VHDL κώδικας της επιθυμητής σχεδίασης συντίθεται με την χρήση του Xilinx XST το οποίο είναι μέρος του Xilinx ISE λογισμικού. Η διαδικασία της σύνθεσης χρησιμοποιείται για την βελτιστοποίηση της αρχιτεκτονικής σχεδιασμού η οποία έχει επιλεχθεί. Μετά την σύνθεση της σχεδίασης, δημιουργείται η αναφορά της σύνθεσης η οποία μας δίνει πληροφορίες για το πόσα λογικά μπλοκ χρησιμοποιούνται και ποια είναι η χρήση της συσκευής της αρχιτεκτονικής σχεδιασμού που συντέθηκε. Η σύνθεση ουσιαστικά χαρτογραφεί την σχεδίαση συμπεριφοράς σε σχεδίαση στο επίπεδο πύλης(τρανζίστορ).

1.4.4 ΥΛΟΠΟΙΗΣΗ ΣΧΕΔΙΑΣΗΣ

Μετά την σύνθεση της σχεδίασης ακολουθεί η υλοποίησή της η οποία αποτελείται από τα παρακάτω βήματα:

1) Μετάφραση

2) Χαρτογράφηση

3) Τοποθέτηση και δημιουργία μονοπατιών / διασυνδέσεων

Πριν την μετάφραση της σχεδίασης, γράφεται ένα User Constrained File (UCF) για να αντιστοιχίσει τους ακροδέκτες του FPGA με τις Εισόδους / Εξόδους της σχεδίασης. Αμέσως μετά η μετάφραση συγχωνεύει αυτό το αρχείο UCF και την netlist που προκύπτει από την σύνθεση και η χαρτογράφηση αντιστοιχίζει τη σχεδίαση στους διαθέσιμους πόρους του FPGA. Αυτό είναι ένα πολύ σημαντικό βήμα του σχεδιασμού. Το τελευταίο βήμα της σχεδιαστικής υλοποίησης είναι η τοποθέτηση των λογικών

μπλοκ της σχεδίασης στο FPGA και η συνένωσή τους ώστε να καταλαμβάνουν τον λιγότερο δυνατό χώρο και να πληρούν τις χρονικές απαιτήσεις. Αυτή η ενέργεια παράγει ένα NCD αρχείο εξόδου.

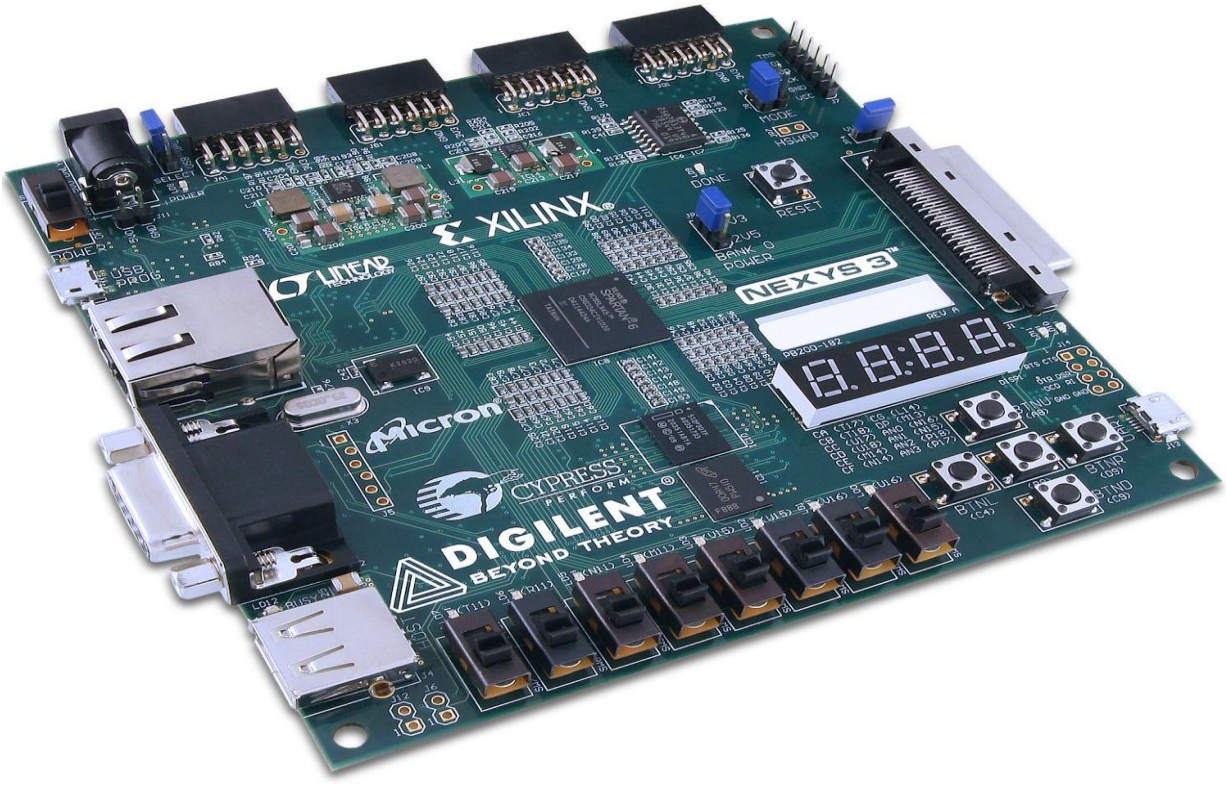
1.4.5 ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΤΗΣ XILINX ΣΥΣΚΕΥΗΣ (FPGA)

Υπάρχει η δυνατότητα παραγωγής αρχείου προγραμματισμού μέσω του Xilinx ISE κατά την οποία μετατρέπεται το NCD αρχείο σε αρχείο BIT. Παράγεται έτσι ένα bitstream (ακολουθία από λογικά 0 και 1) για την διαμόρφωση του FPGA. Αυτό το bit αρχείο χρησιμοποιείται για να προγραμματίσει το FPGA.

1.4.6 ΡΥΘΜΙΣΗ ΣΥΣΚΕΥΗΣ ΣΤΟΧΟΥ

Υπάρχει η επιλογή της παραγωγής στόχου PROM / ACE στην καρτέλα διαδικασίας του Xilinx ISE η οποία μετατρέπει το bit αρχείο στο PROM ή ACE αρχείο. Αυτό το αρχείο μπορεί να ‘’κατέβει’’ απευθείας στα κύτταρα μνήμης του FPGA. Πρέπει να βεβαιωθούμε ότι το FPGA είναι συνδεδεμένο με τον υπολογιστή όπου αναπτύσσουμε την σχεδίαση. Αφού ‘’κατεβάσουμε’’ το αρχείο PROM ή ACE στο FPGA, είναι πλέον έτοιμο να χρησιμοποιηθεί και να υλοποιήσει την σχεδίασή μας. Μπορούμε να δώσουμε διαφορετικούς συνδυασμούς εισόδων για να δούμε πως η έξοδος του FPGA αλλάζει. Αυτές οι είσοδοι μπορούν να δοθούν από διακόπτες, ανάλογα με το εκάστοτε FPGA, των οποίων οι αριθμοί των ακροδεκτών έχουν αντιστοιχιστεί μέσω του UCF αρχείου με τις εισόδους της εκάστοτε σχεδίασης. Οι έξοδος μπορεί να απεικονιστεί σε LED, σε LCD οθόνη ή σε κάποιον ακροδέκτη του FPGA, ανάλογα με το τι επιθυμούμε και το τι παρέχεται από το συγκεκριμένο FPGA που χρησιμοποιούμε, των οποίων οι αριθμοί των ακροδεκτών έχουν αντιστοιχιστεί μέσω του UCF αρχείου με τις εξόδους της σχεδίασης. Παρακάτω παρουσιάζονται το SPARTAN 3E και το SPARTAN6 XC6SLX16 FPGA τα οποία χρησιμοποιήσαμε για την δημιουργία της PWM και των SPWM και SVPWM αντίστοιχα.

Σχήμα 1.5: Πλακέτα SPARTAN-3E Starter Kit (FPGA)



Σχήμα 1.6: Πλακέτα SPARTAN6 XC6SLX16 FPGA

1.4.7 ΑΝΤΙΣΤΟΙΧΙΣΗ ΑΚΡΟΔΕΚΤΩΝ FPGA ΜΕ ΣΗΜΑΤΑ ΕΙΣΟΔΟΥ-ΕΞΟΔΟΥ

Η αντιστοίχιση αυτή γίνεται με το αρχείο περιορισμού(UCF). Ανάλογα με το FPGA που χρησιμοποιούμε αντιστοιχούμε τα σήματα της σχεδίασής μας με τους διάφορους ακροδέκτες, διακόπτες, τα LED, την οθόνη και μέσω αυτών μπορούμε κατά τη διάρκεια που ‘’τρέχει’’ το πρόγραμμά μας στο FPGA να αλλάζουμε τις εισόδους και να παρατηρούμε τις εξόδους.

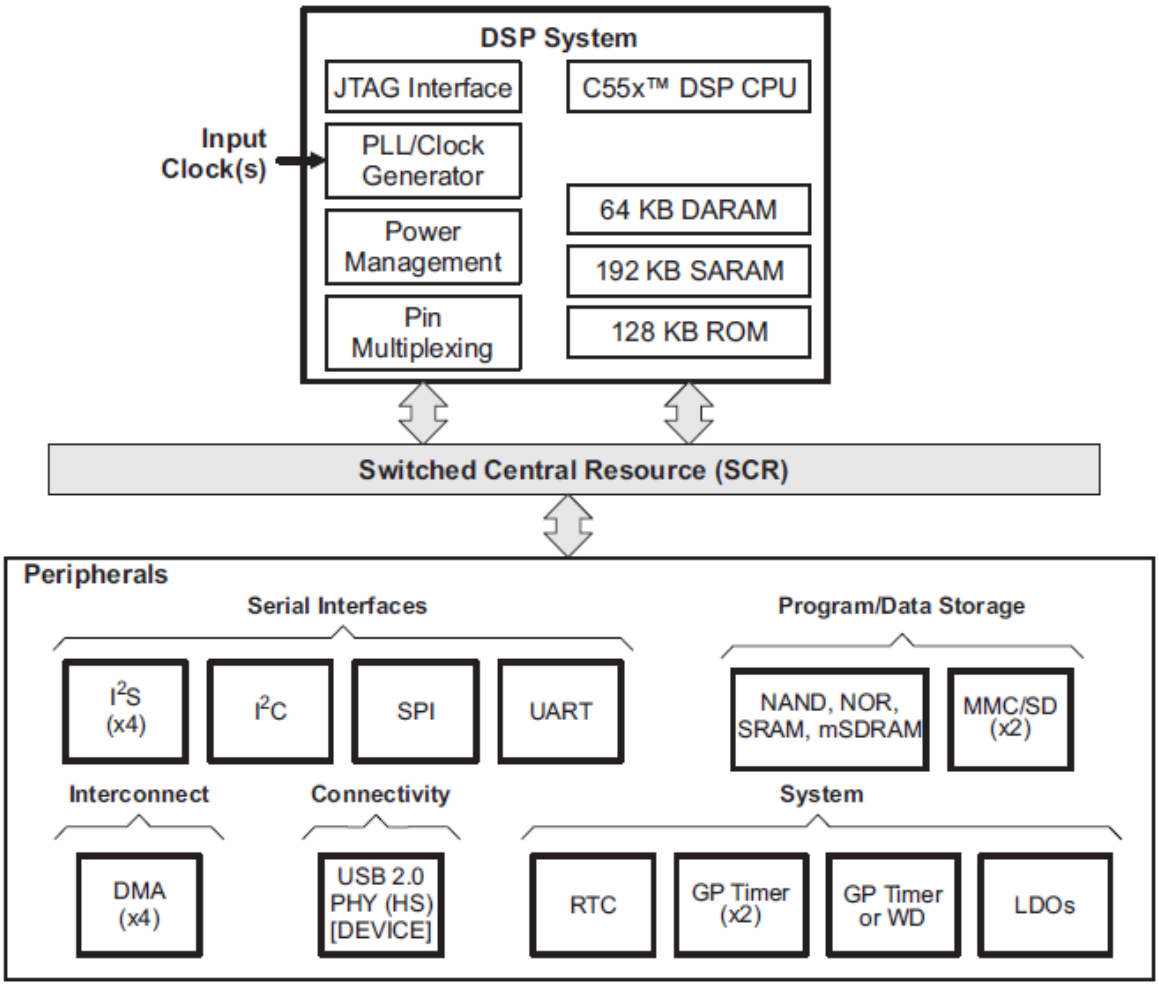
1.5 Τα FPGA στα ηλεκτρονικά ισχύος

Η παρούσα εργασία αφορά σε θέματα που ασχολούνται με τα ηλεκτρονικά ισχύος και ως εκ’ τούτου θα κάνουμε μία εισαγωγική αναφορά στη σχέση που έχουν τα FPGA με τον τομέα αυτό όπως και τις αλλαγές τις οποίες μπορούν να επιφέρουν. Επειδή σήμερα για τις περισσότερες εφαρμογές που αφορούν σε ενεργειακά θέματα χρησιμοποιούνται DSP θα κάνουμε μία συνοπτική αναφορά και σε αυτά και θα εξετάσουμε τις διαφοροποιήσεις που προσφέρουν τα FPGA.

1.5.1 Τα DSP στα ηλεκτρονικά ισχύος

Ο ψηφιακός έλεγχος των διακοπτικών πηγών ενέργειας χρησιμοποιείται πλέον δραστικά στην βιομηχανία, λόγω του χαμηλού κόστους, το οποίο συνοδεύεται και από υψηλές αποδόσεις. Επίσης οι ψηφιακοί ελεγκτές επηρεάζονται λιγότερο από τον περιβαλλοντικό θόρυβο, την μεταβολή της θερμοκρασίας και δεν επηρεάζονται από τις μεταβολές των στοιχείων και τις απώλειες μεταγωγής. Πλέον οι επεξεργαστές ψηφιακού σήματος αποτελούν αναπόσπαστο κομμάτι όλων των μετατροπέων ηλεκτρονικών ισχύος και χρησιμοποιούνται κατά κύριο λόγο για τον κατάλληλο προγραμματισμό αγωγής των ημιαγωγικών διακοπτών που χρησιμοποιούνται.

Για λόγους πληρότητας παραθέτουμε παρακάτω και το μπλοκ διάγραμμα ενός τυπικού επεξεργαστή ψηφιακού σήματος. Αυτό το κάνουμε καθώς συχνά τίθεται το ερώτημα για το αν τελικά, λαμβάνοντας υπ’ όψιν όλες τις παραμέτρους, είναι συμφέρουσα η χρησιμοποίηση των FPGA έναντι των DSP για την υλοποίηση της εκάστοτε σχεδίασης. Το παρακάτω διάγραμμα αφορά σε έναν ειδικό επεξεργαστή βελτιστοποιημένο για γρήγορη λειτουργία.

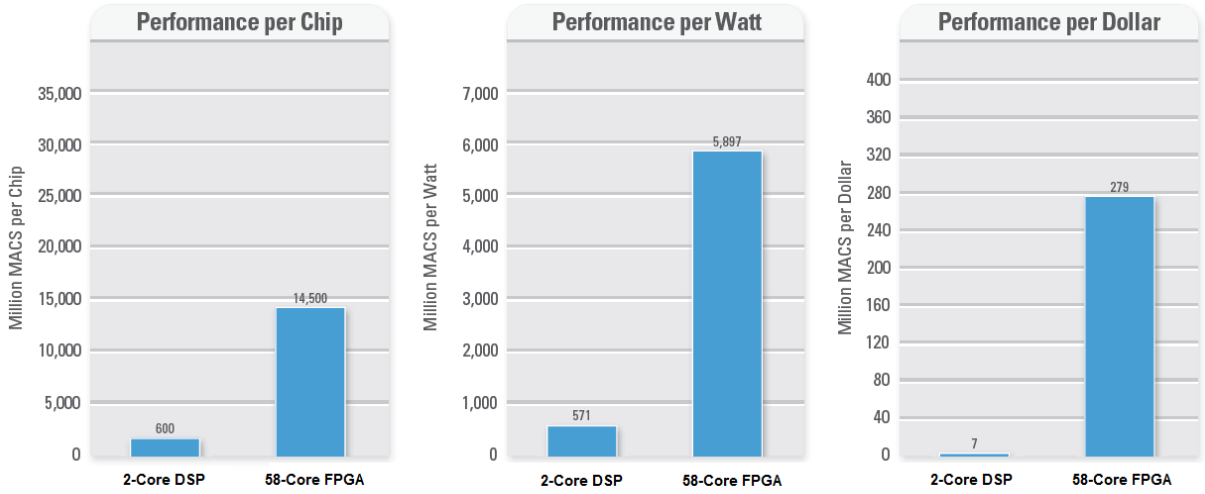


Σχήμα 1.7: Μπλοκ Διάγραμμα ενός τυπικού DSP

Στο ερώτημα το οποίο αναφέρθηκε και παραπάνω και αφορά στο αν θα προτιμήσουμε για την υλοποίηση μίας σχεδίασης τη χρήση ενός ψηφιακού επεξεργαστή σήματος(DSP) ή ενός FPGA η απάντηση ποικίλει από εφαρμογή σε εφαρμογή. Ανάλογα με τα στοιχεία και τις απαιτήσεις που έχει η κάθε εφαρμογή αλλά και την αναμενόμενη παραγωγή του εκάστοτε προϊόντος μπορεί να προτιμήσουμε την χρησιμοποίηση ενός DSP ή ενός FPGA.

1.5.2FPGA έναντι DSP στα Ηλεκτρονικά Ισχύος

Με βάση τα πλεονεκτήματα που αναλύσαμε παραπάνω, γίνεται κατανοητό πως η απόδοση με τη χρήση των FPGA αυξάνεται κατά πολύ, ανάλογα πάντα και με την εφαρμογή που υλοποιούμε, όπως φαίνεται και στο παρακάτω παράδειγμα όπου συγκρίνονται οι αποδόσεις ενός διπύρηνου DSP επεξεργαστή και του Spartan-6 FPGA. Τα παρακάτω αποτελέσματα αντλήθηκαν από σχετικό άρθρο ερευνητών της Xilinx και της National Instruments.



Σχήμα 1.8: Σύγκριση αποδόσεων του Spartan 6 FPGA με ένα διπύρηνο επεξεργαστή

	Dual-Core DSP	Spartan-6 LX45 FPGA	Βαθμός Απόδοσης (FPGA/DSP)
Million MACS per Chip	600	14.500	24
Million MACS per Watt	571	5.897	10
Million MACS per Dollar	7	279	40

Πίνακας 1.1: Βαθμοί απόδοσης Spatan6 FPGA / Dual Core DSP

Όπου MACS = Multiply-accumulate operations per second, και αφορά σε μέτρο απόδοσης ενός DSP.

Ο παρακάτω πίνακας είναι διαφωτιστικός όσον αφορά στις διαφορές ανάμεσα σε FPGA και DSP επεξεργαστές. Με κόκκινο χρώμα παρουσιάζεται το στοιχείο το οποίο υπερτερεί σε κάθε κατηγορία:

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ	DSPs	FPGA
Απόδοση	600 MMACS	**14,500 MMACS**
Περιφερειακά	Σταθερά	**Ευέλικτα**
Πυρήνες Επεξεργαστή	Μονός ή Διπλός	**Πολλαπλοί**
Σχήμα Διαμόρφωσης	Σταθερή PWM	**Ευέλικτο**
Χρόνος Απόκρισης Βρόγχου	~50 μs	**~2.5μs**
Silicon Gate Level(SGL) Reconfigurability	-	**√**
Περιβάλλον Σχεδίασης	**C (Υψηλότερο Επίπεδο Αφαίρεσης)**	VHDL & Verilog (RTL)

Πίνακας 1.2: Διαφορές ανάμεσα στα FPGA και στα DSP

1.6 Η γλώσσα περιγραφής υλικού VHDL

Η VHDL(VHSIC hardware description language ή γλώσσα περιγραφής υλικού VHSIC) είναι μία γλώσσα περιγραφής υλικού που χρησιμοποιείται στον αυτοματισμό ηλεκτρονικών σχεδιάσεων(electronic design automation) για την περιγραφή ψηφιακών και μεικτών(mixed-signal) συστημάτων, όπως τα FPGA και τα ολοκληρωμένα κυκλώματα. Η VHDL συνήθως χρησιμοποιείται για τη συγγραφή μοντέλων σε κείμενο που περιγράφουν ένα λογικό κύκλωμα. Ένα πρόγραμμα σύνθεσης μπορεί να επεξεργαστεί ένα τέτοιο μοντέλο μόνο αν είναι μέρος της λογικής σχεδίασης, επομένως χρησιμοποιείται ένα πρόγραμμα προσομοίωσης για να δοκιμαστεί η λογική σχεδίαση χρησιμοποιώντας μοντέλα προσομοίωσης που αναπαριστούν τα λογικά κυκλώματα που αντιστοιχούν στη σχεδίαση. Η συλλογή αυτή από μοντέλα προσομοίωσης συνήθως αποκαλείται testbench.

Η VHDL έχει δομές που χειρίζονται τον παραλληλισμό που υπάρχει στις σχεδιάσεις υλικού. Επίσης, η VHDL έχει ισχυρούς τύπους (strongly typed) και δεν κάνει διάκριση μεταξύ κεφαλαίων και μικρών γραμμάτων. Για την απευθείας αναπαράσταση συχνών λειτουργιών του υλικού, υπάρχουν πολλά χαρακτηριστικά της VHDL, όπως ένα μεγάλο σύνολο από λογικές πράξεις όπως η nand και η nor. Η VHDL επιτρέπει επίσης τη δεικτοδότηση πινάκων σε σειρά από το μικρότερο προς το μεγαλύτερο δείκτη ή αντίστροφα - και οι δύο συμβάσεις χρησιμοποιούνται στο υλικό, ενώ στις περισσότερες άλλες γλώσσες προγραμματισμού χρησιμοποιείται μόνο ο πρώτος τρόπος δεικτοδότησης.

Η VHDL έχει δυνατότητες εισόδου και εξόδου σε αρχεία και μπορεί να χρησιμοποιηθεί σαν γλώσσα γενικών καθηκόντων για επεξεργασία κειμένου, αλλά τα αρχεία χρησιμοποιούνται συνήθως από ένα testbench προσομοίωσης για τον ορισμό διεγέρσεων (stimuli), αλληλεπίδρασης με τον χρήστη και για τη σύγκριση των ληφθέντων δεδομένων με τα επιθυμητά δεδομένα. Παρόλα αυτά, οι περισσότεροι σχεδιαστές αφήνουν αυτήν την εργασία στον προσομοιωτή.

Για έναν προγραμματιστή χωρίς εμπειρία είναι σχετικά εύκολο να παράγει κώδικα που προσομοιώνεται με επιτυχία αλλά δε μπορεί να παραχθεί σαν πραγματική υλοποίηση, ή είναι πολύ μεγάλος για να χρησιμοποιηθεί στην πράξη.

Η σχεδίαση του υλικού μπορεί να γίνει σε ένα ολοκληρωμένο περιβάλλον ανάπτυξης για VHDL (για υλοποίηση FPGA τέτοια είναι το Xilinx ISE, το Altera Quartus, το Synopsys Synplify και το Mentor Graphics HDL Designer), ώστε να παραχθεί το σχηματικό διάγραμμα RTL του επιθυμητού κυκλώματος. Μετά από αυτό, το παραγόμενο σχηματικό διάγραμμα μπορεί να επαληθευτεί με χρήση λογισμικού προσομοίωσης που δείχνει τις κυματομορφές των εισόδων και των εξόδων του κυκλώματος μετά την δημιουργία του κατάλληλου testbench. Η δημιουργία του σωστού testbench για ένα κύκλωμα ή έναν κώδικα σε VHDL απαιτεί τον σωστό ορισμό των εισόδων.

Όταν ένα μοντέλο σε VHDL μεταφράζεται σε "πύλες και γραμμές" που αντιστοιχίζονται σε μια προγραμματιζόμενη λογική συσκευή όπως ένα CPLD ή ένα FPGA, τότε το πραγματικό υλικό είναι αυτό που ρυθμίζεται και δεν "εκτελείται" ο κώδικας VHDL σε κάποιου τύπου επεξεργαστή.

1.6.1 Πλεονεκτήματα της γλώσσας VHDL

Η γλώσσα περιγραφής υλικού VHDL παρουσιάζει διάφορα πλεονεκτήματα που καθιστούν τη χρησιμοποίησή της συμφέρουσα για μία πληθώρα εφαρμογών. Τα σημαντικότερα από τα πλεονεκτήματα αυτά παρουσιάζονται παρακάτω:

1) Το βασικό πλεονέκτημα της VHDL, όταν αυτή χρησιμοποιείται για σχεδίαση συστημάτων, είναι ότι επιτρέπει την περιγραφή (μοντελοποίηση) και την επαλήθευση (προσομοίωση) του επιθυμητού συστήματος, πριν τα εργαλεία σύνθεσης μεταφράσουν τη σχεδίαση σε πραγματικό υλικό (πύλες και γραμμές).

2) Ένα άλλο όφελος της VHDL είναι ότι επιτρέπει τον ορισμό ταυτόχρονων συστημάτων (concurrent systems). Η VHDL είναι γλώσσα ροής δεδομένων, σε αντίθεση με τις διαδικαστικές γλώσσες προγραμματισμού όπως η BASIC, η C και η συμβολική γλώσσα, οι οποίες εκτελούνται ακολουθιακά, με κάθε εντολή να ακολουθεί την προηγούμενη.

3) Ένα έργο σε VHDL έχει πολλές εφαρμογές. Ένα μπλοκ υπολογισμού (calculation block) δημιουργείται μια φορά αλλά μπορεί να χρησιμοποιηθεί σε άλλα έργα. Μπορούν επίσης να ρυθμιστούν διάφορες παράμετροι διαμόρφωσης και λειτουργίας του μπλοκ (παράμετροι χωρητικότητας, το μέγεθος της μνήμης, η βάση των στοιχείων (element base), η σύνθεση μπλοκ και η δομή διασύνδεσης).

4) Ένα έργο σε VHDL είναι επίσης μεταφέρσιμο. Αν έχει δημιουργηθεί για μια βάση στοιχείων, μπορεί να μεταφερθεί σε μια άλλη βάση, για παράδειγμα σε ένα VLSI με διάφορες τεχνολογίες.

1.6.2 Μειονεκτήματα της γλώσσας VHDL

Εκτός βεβαίως από τα παραπάνω πλεονεκτήματα που χαρακτηρίζουν την γλώσσα περιγραφής υλικού VHDL, παρουσιάζει και μερικά μειονεκτήματα. Τα κυριότερα από αυτά παρουσιάζονται παρακάτω:

1) Η VHDL θα μπορούσε να χαρακτηριστεί ως φλύαρη, καθώς για την περιγραφή ορισμένων λειτουργιών απαιτούνται αρκετές γραμμές κώδικα, και συνεπώς πολλές φορές περίπλοκη και δυσνόητη.

2) Επίσης, υπάρχουν δομές που εξυπηρετούν παρόμοιο σκοπό αλλά παρουσιάζουν πολύ διαφορετική σύνταξη. Χαρακτηριστικό παράδειγμα αποτελούν οι δομές case και select.

3) Αντίστοιχο μειονέκτημα αποτελεί το γεγονός πως δομές που έχουν παρόμοια σύνταξη, όπως οι variables και τα signals, παρουσιάζουν διαφορετική σημασιολογία.

4) Τέλος, το υλικό το οποίο συντίθεται δεν είναι πάντα προφανές, όπως για παράδειγμα η δημιουργία μανδαλωτών αντί για flip-flop λόγω της μη δήλωσης όλων των δυνατών περιπτώσεων απόδοσης τιμής ενός σήματος.

ΚΕΦΑΛΑΙΟ 2^{ο}

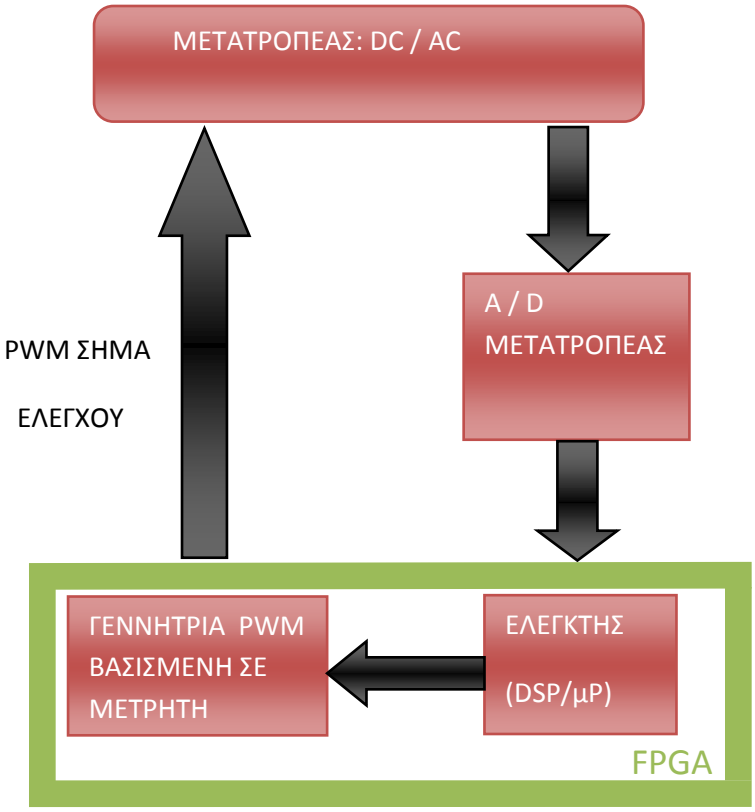
ΥΛΟΠΟΙΗΣΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΤΗΣ PWM ΤΕΧΝΙΚΗΣ

2.1 Εισαγωγή

Στο κεφάλαιο αυτό θα ασχοληθούμε με την PWM(Pulse Width Modulation)τεχνική. Είναι μία τεχνική διαμόρφωσης η οποία παράγει παλμούς μεταβλητού πλάτους για την αναπαράσταση του πλάτους ενός αναλογικού σήματος εισόδου. Η PWM είναι στις μέρες μας ένα αναπόσπαστο κομμάτι όλων των ενσωματωμένων συστημάτων. Συνιστά μία ευρέως αποδεκτή τεχνική ελέγχου στις περισσότερες ηλεκτρονικές εφαρμογές. Η PWM είναι μία ισχυρή τεχνική για τον έλεγχο αναλογικών συστημάτων με τις ψηφιακές εξόδους ενός επεξεργαστή. Η PWM χρησιμοποιείται σε πλήθος εφαρμογών , που κυμαίνονται από μετρήσεις και επικοινωνίες μέχρι τον έλεγχο της ισχύος και τις εφαρμογές μετατροπής. Υπάρχουν πολλές μέθοδοι ανάλογα με την αρχιτεκτονική και τις απαιτήσεις του συστήματος. Η υλοποίηση της σχεδίασης τους εξαρτάται από τον τύπο της εφαρμογής, την κατανάλωση ισχύος, τους διαθέσιμους ημιαγωγούς και τα κριτήρια κόστους και απόδοσης. Οι ψηφιακές μέθοδοι είναι οι πιο κατάλληλες για τον σχεδιασμό PWM γεννητριών καθώς είναι πολύ ευέλικτες και λιγότερο ευαίσθητες στον περιβαλλοντικό θόρυβο.

2.2 ΨΗΦΙΑΚΕΣ ΤΕΧΝΙΚΕΣ PWM ΠΑΡΑΓΩΓΗΣ

Πολλές ψηφιακές τεχνικές βασίζονται στην χρήση μετρητών και συγκριτών για την σχεδίαση. Αυτές οι ψηφιακές τεχνικές είναι ευκολότερο να εφαρμοστούν από ότι οι αναλογικές. Επίσης είναι απρόσβλητες από τον περιβαλλοντικό θόρυβο, τη μεταβολή της θερμοκρασίας και δεν επηρεάζονται από τις μεταβολές των στοιχείων και τις απώλειες μεταγωγής. Για εξελιγμένα συστήματα ελέγχου, είναι επιθυμητή η χρήση των συστημάτων ψηφιακής διαμόρφωσης. Παρακάτω παρουσιάζουμε το γενικό μπλοκ διάγραμμα του ψηφιακού ελέγχου ενός μετατροπέα:



Σχήμα 2.1: Γενικό Μπλοκ Διάγραμμα Ψηφιακού Ελέγχου Μετατροπέα

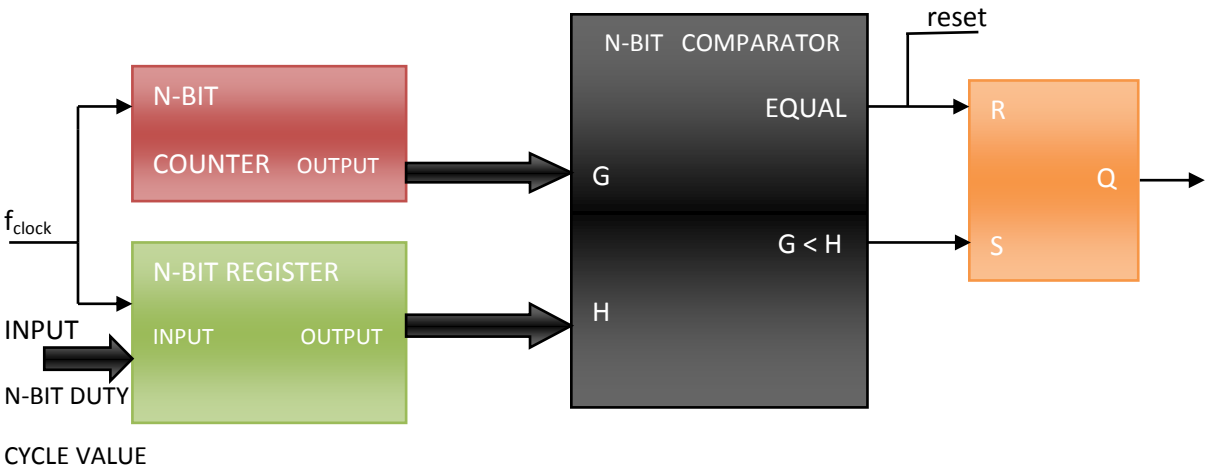
Υπάρχουν πολλές ψηφιακές τεχνικές διαθέσιμες ανάλογα με την διάταξη που χρησιμοποιείται και τον τύπο του μετρητή αλλά σε αυτό το κεφάλαιο θα ασχοληθούμε με μία βασική τοπολογία γεννήτριας PWM. Αυτή είναι η παρακάτω:

- Υψηλής Συχνότητας PWM Γεννήτρια βασισμένη σε Μετρητή.

2.3 ΥΨΗΛΗΣ ΣΥΧΝΟΤΗΤΑΣ PWM ΓΕΝΝΗΤΡΙΑ ΒΑΣΙΣΜΕΝΗ ΣΕ ΜΕΤΡΗΤΗ

Σύμφωνα με αυτή την αρχιτεκτονική υπάρχει ένας υψηλής ταχύτητας μετρητής των N-bit του οποίου η έξοδος συγκρίνετε με το περιεχόμενο ενός καταχωρητή, ο οποίος αποθηκεύει το επιθυμητό duty cycle της εισόδου(N-bit τιμές), με την βοήθεια ενός συγκριτή. Η έξοδος του συγκριτή είναι ‘’1’’ για όσο χρόνο ο μετρητής λαμβάνει μικρότερη τιμή από το duty cycle. Αντίθετα, μηδενίζεται όταν ο μετρητής έχει τιμή ίση ή μεγαλύτερη του duty cycle. Αυτή η έξοδος του συγκριτή χρησιμοποιείται για να ενεργοποιήσει έναν RS μανδαλωτή. Η έξοδος του μανδαλωτή δίνει την επιθυμητή έξοδο PWM. Το πλεονέκτημα αυτής της μεθόδου είναι ότι χρησιμοποιείται για να δημιουργήσει έξοδο PWM υψηλής συχνότητας κάτι το οποίο δεν είναι εφικτό με την κανονική προσέγγιση με μετρητή. Το παρακάτω σχήμα απεικονίζει το αντίστοιχο διάγραμμα αυτής της αρχιτεκτονικής:

2.3.1 ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΧΕΔΙΑΣΗΣ



Σχήμα 2.2: Αρχιτεκτονική της PWM Γεννήτριας

2.3.2 VHDL ΚΩΔΙΚΑΣ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ

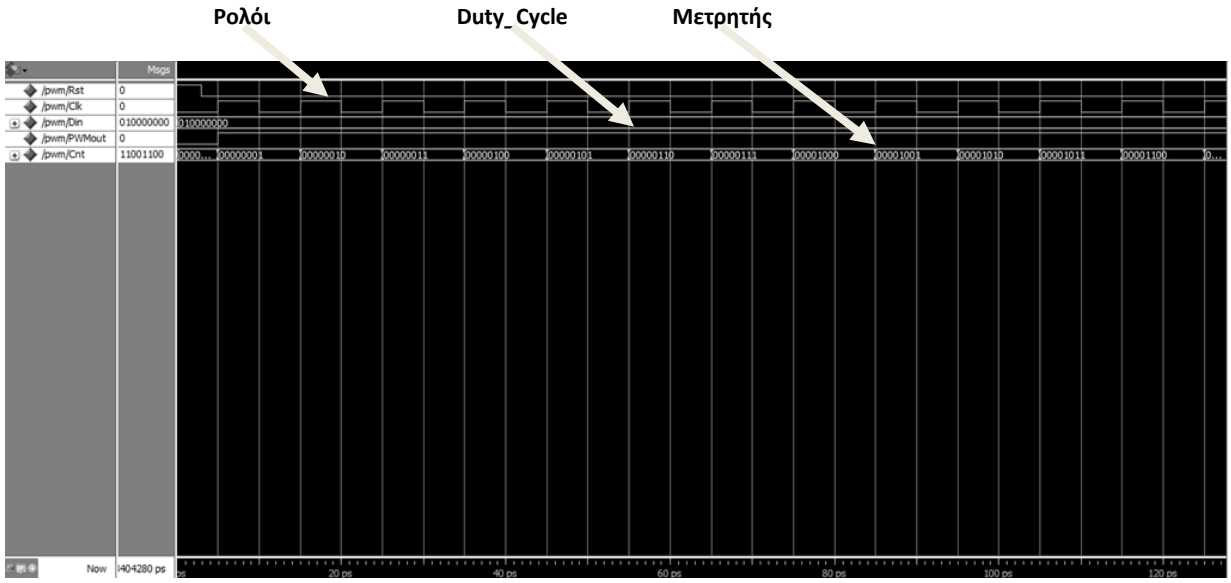
Ο κώδικας VHDL για την υλοποίηση αυτής της αρχιτεκτονικής παρουσιάζεται παρακάτω:

```
library IEEE;
---------- ΔΗΛΩΣΗ ΤΩΝ ΑΠΑΡΑΙΤΗΤΩΝ ΠΑΚΕΤΩΝ ΤΩΝ ΒΙΒΛΙΩΘΗΚΩΝ----------
 use IEEE.std_logic_1164.all;
 use IEEE.numeric_std.all;

----------

---------- ΔΗΛΩΣΗ ΤΗΣ ΕΝΟΤΗΤΑΣ ΠΟΥ ΘΑ ΧΡΗΣΙΜΟΠΟΙΗΣΟΥΜΕ ----------


Entity PWM is
Port (   Rst : In  std_logic;
         Clk : In  std_logic;
         Din : In  std_logic_vector (8 downto 0);
```

```
        PWMout : Out std_logic );
end PWM;
-------------------------------------------------------------------------------------------------------------------------------
---------- ΔΗΛΩΣΗ ΚΑΙ ΥΛΟΠΟΙΗΣΗ ΤΗΣ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΠΟΥ ΘΑ ΧΡΗΣΙΜΟΠΟΙΗΣΟΥΜΕ ----------
Architecture RTL_Comp of PWM is
    signal Cnt : unsigned (7 downto 0);
begin
process (Clk,Rst)
begin
 if Rst='1' then
   Cnt <= (others=>'0');
   PWMout <= '0';
 elsif rising_edge(Clk) then
   Cnt <= Cnt + 1;
   if unsigned(Din(7 downto 0)) >= Cnt  then
     PWMout <= '1';
   else
     PWMout <= '0';
   end if;
 end if;
end process;

end RTL_Comp;
-------------------------------------------------------------------------------------------------------------------------------
```

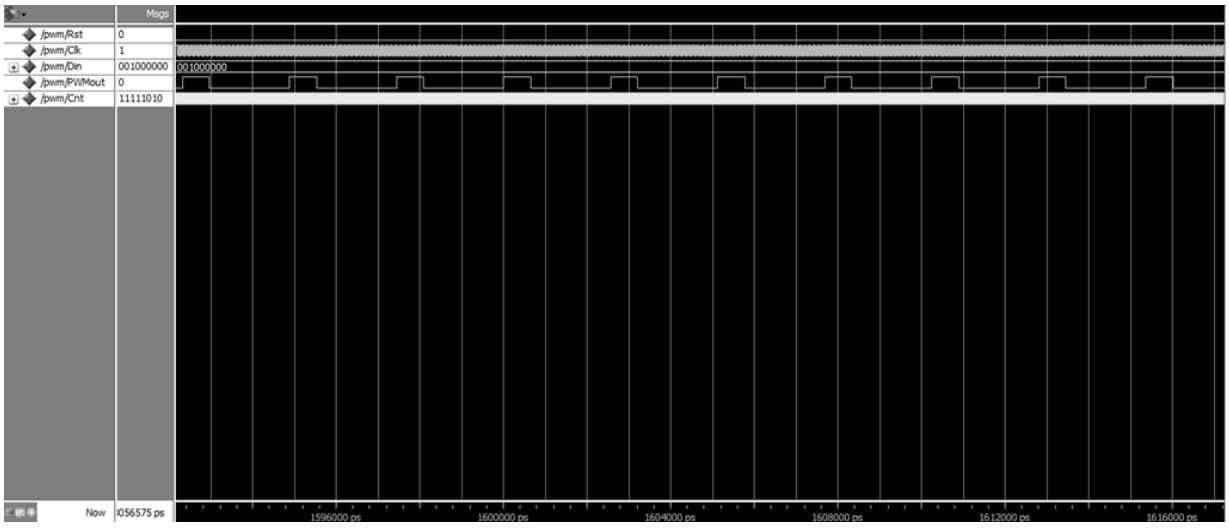
2.3.3 ΠΡΟΣΟΜΟΙΩΣΗ ΣΤΟ MODELSIM

Για την επαλήθευση της λειτουργίας της παραπάνω σχεδίασης χρησιμοποιήσαμε το εργαλείο προσομοίωσης Modelsim Altera 10.0c(Quartus II 11.1) Starter Edition. Με αυτό μεταγλωττίσαμε τον κώδικα και κάναμε προσομοιώσεις για διαφορετικά DUTY CYCLE. Τα αποτελέσματα των προσομοιώσεων αυτών παρουσιάζονται παρακάτω:

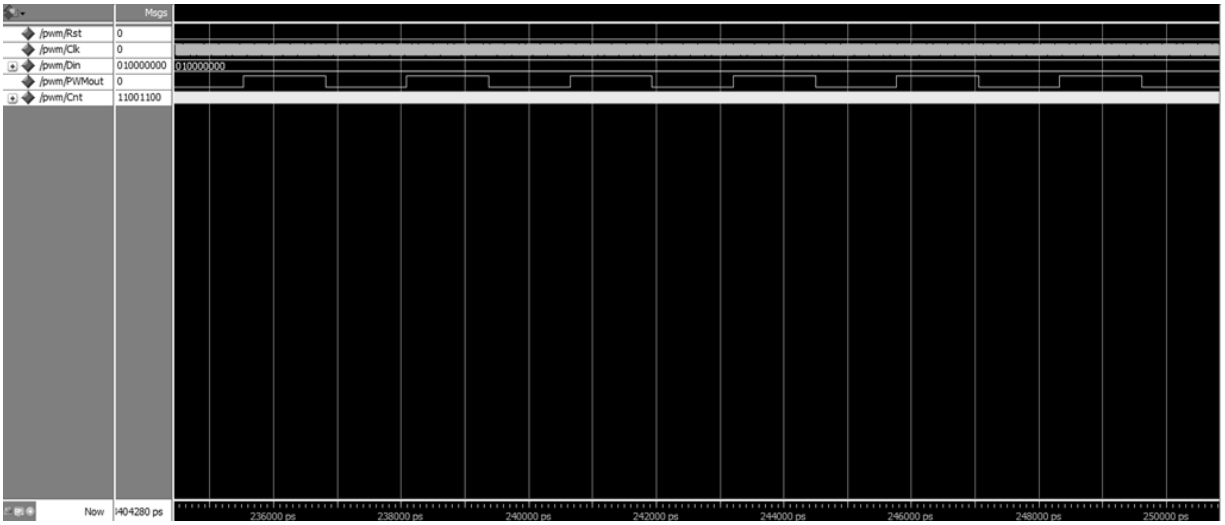


Σχήμα 2.3: Σήματα PWM Σχεδίασης

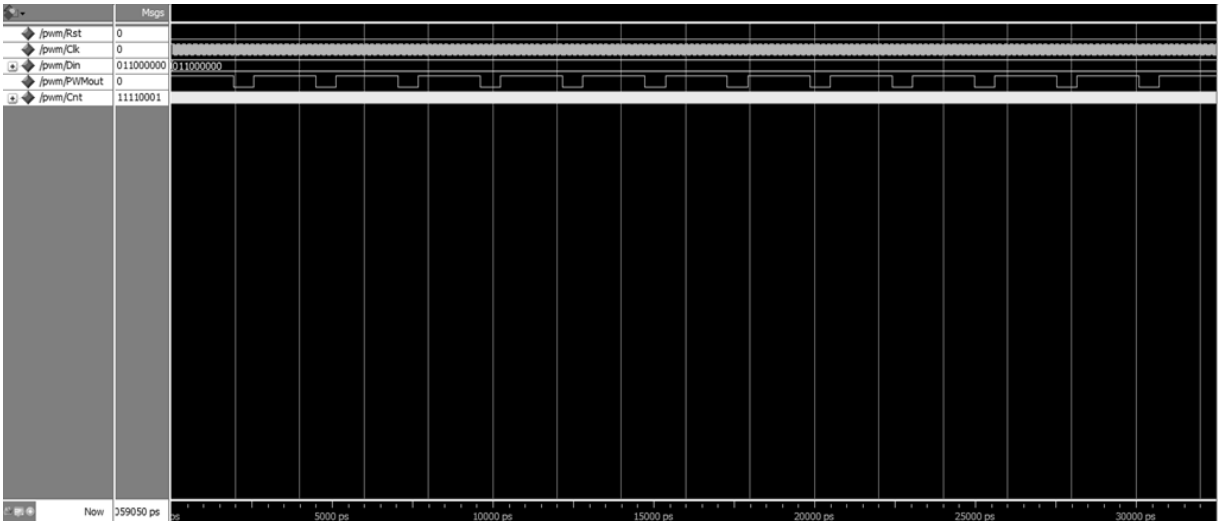
Τα αποτελέσματα για διαφορετικές τιμές Duty Cycle παρουσιάζονται παρακάτω:



Σχήμα 2.4: Για DUTY CYCLE = 001000000 => DUTY CYCLE = 25%



Σχήμα2.5: Για DUTY CYCLE = 010000000 => DUTY CYCLE = 50%

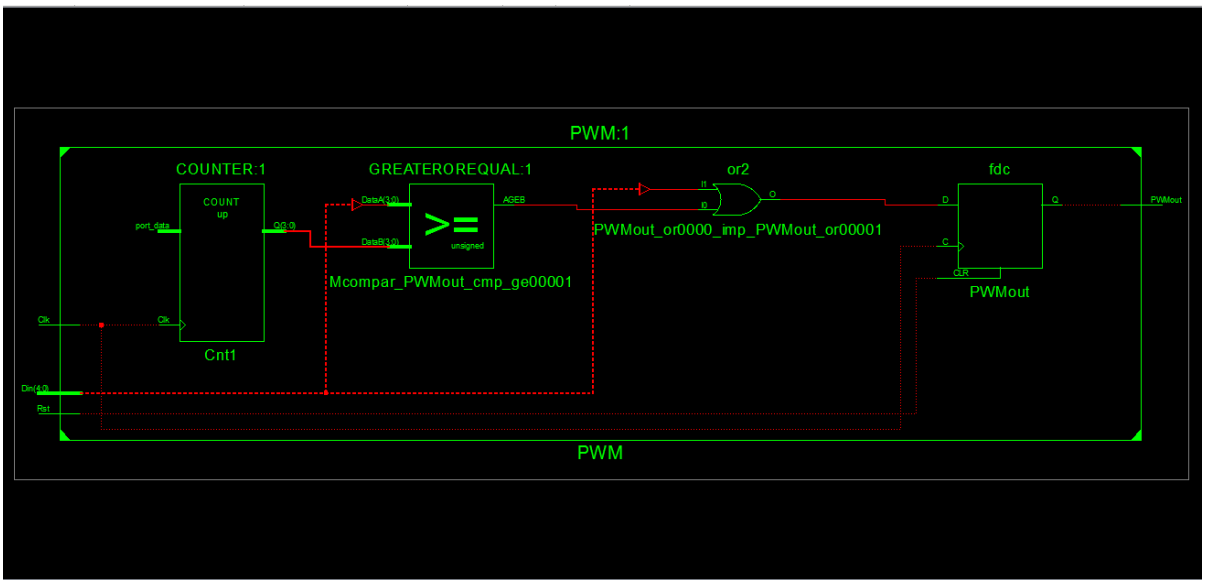


Σχήμα 2.6: Για DUTY CYCLE = 011000000 => DUTY CYCLE = 75%

2.3.4 RTL SCHEMATIC ΣΤΟ XILINX ISE

Παρακάτω παρουσιάζεται το RTL Schematic της αρχιτεκτονικής όπου:

RTL(Register Transfer Level) = Σχεδιαστική αφαίρεση που μοντελοποιεί ένα σύγχρονο ψηφιακό σύστημα με την ροή ψηφιακών σημάτων(δεδομένων) ανάμεσα σε καταχωρητές και λογικές λειτουργίες που εκτελούνται σε αυτά τα σήματα.



Σχήμα 2.7: RTL σχηματική αναπαράσταση αρχιτεκτονικής

Παρατηρούμε πως όντως ο μετρητής και το σήμα εισόδου εισέρχονται σε ένα συγκριτή και ανάλογα με το αποτέλεσμα προκύπτει και η έξοδος.

2.3.5 ΑΝΑΦΟΡΑ ΧΡΗΣΙΜΟΠΟΙΗΣΗΣ ΣΥΣΚΕΥΗΣ

Μετά τη σύνθεση της σχεδίασής μας την οποία κάναμε μέσα από το εργαλείο ISE Design Suite της Xilinx προέκυψε μία αναφορά με τα λογικά μπλοκ τα οποία θα χρησιμοποιηθούν για την υλοποίηση στο FPGA. Τα στοιχεία της αναφορά αυτής παρουσιάζονται παρακάτω:

Slice Logic Utilization	Used	Available	Utilization
Number of Slices	9	960	0%
Number of Slice Flip Flops	9	1920	0%
Number of 4 input LUTs	16	1920	0%
Number of bonded IOBs	11	66	16%
Number of GCLKs	1	24	4%

Πίνακας 2.1: Χρησιμοποιούμενοι πόροι της συσκευής για την υλοποίηση της υψηλής συχνότητας PWM τεχνικής

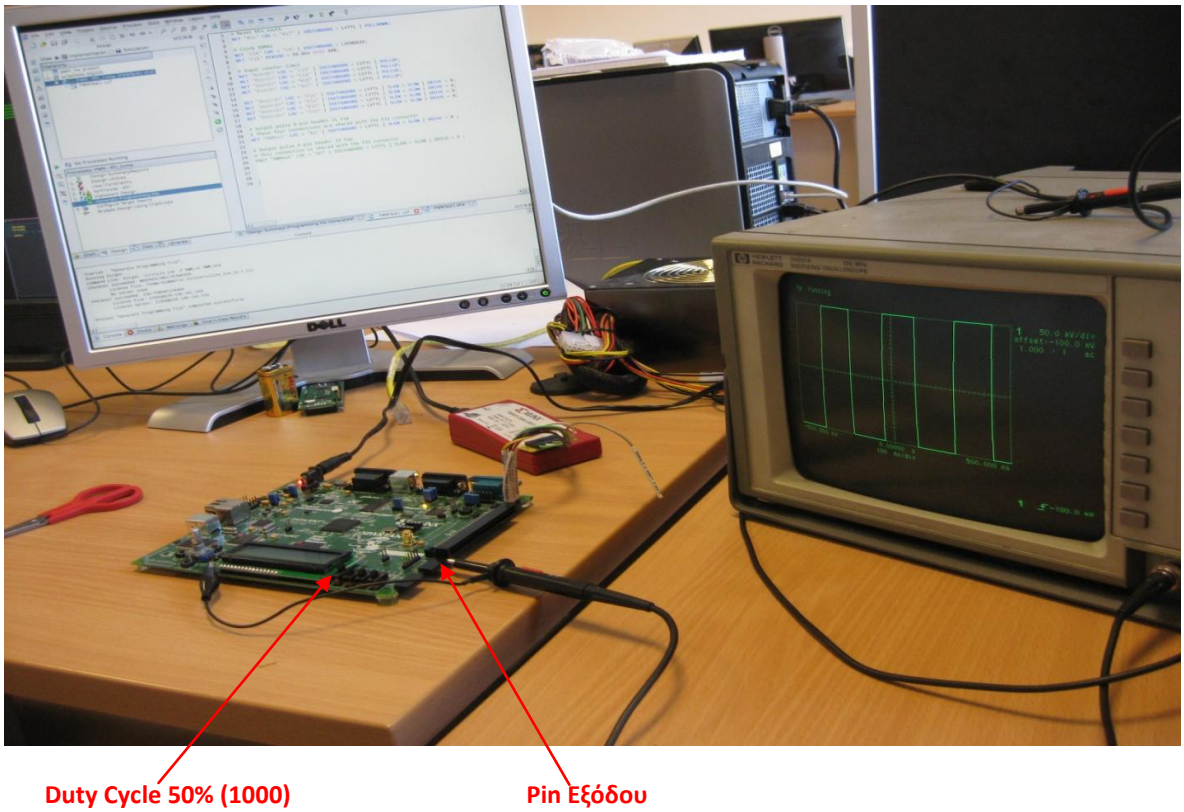
Παρατηρούμε πως ο χώρος ο οποίος δεσμεύεται είναι πολύ μικρός σε σχέση με τον διαθέσιμο από το FPGA.

2.4 ΠΕΙΡΑΜΑΤΙΚΗ ΔΙΑΔΙΚΑΣΙΑ

Για να επαληθεύσουμε τα θεωρητικά αλλά και τα αποτελέσματα των προσομοιώσεών μας πραγματοποιήσαμε στο εργαστήριο την παραγωγή PWM παλμών με duty cycle 50% και 75% με την χρήση του FPGA SPARTAN-3E Starter Kit το οποίο παρουσιάστηκε παραπάνω. Ακολουθήσαμε τα παρακάτω βήματα:

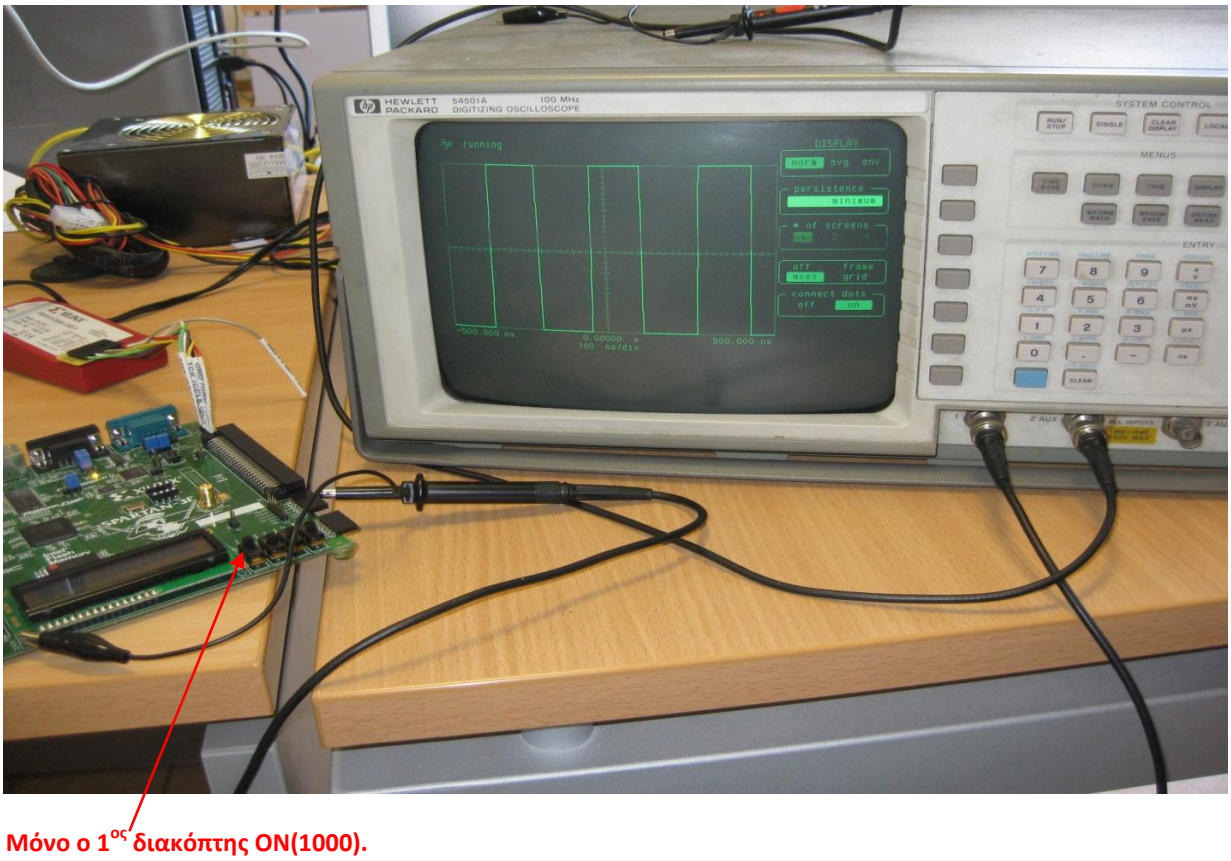
1) Τεστάραμε τον VHDL κώδικα για την παραγωγή των PWM παλμών με την χρήση του προγράμματος Xilinx ISE Design Suite.
2) Με την χρήση του ίδιου προγράμματος(Xilinx ISE Design Suite) ‘’φορτώσαμε’’ τον VHDL κώδικα στο FPGA το οποίο προηγουμένως είχαμε συνδέσει με τον υπολογιστή.
3) Τέλος συνδέσαμε το probe του παλμογράφου στο pin του FPGA στο οποίο είχαμε τοποθετήσει, με το κατάλληλο αρχείο περιορισμών, την έξοδο του κώδικά μας. Την είσοδο, μέσω της οποίας δίνουμε το επιθυμητό duty cycle την συνδέσαμε μέσω του αρχείου περιορισμών με τους τέσσερις διακόπτες του FPGA όπως φαίνεται και στα παρακάτω σχήματα.

Το αποτέλεσμα της διαδικασίας αυτής παρουσιάζεται παρακάτω:



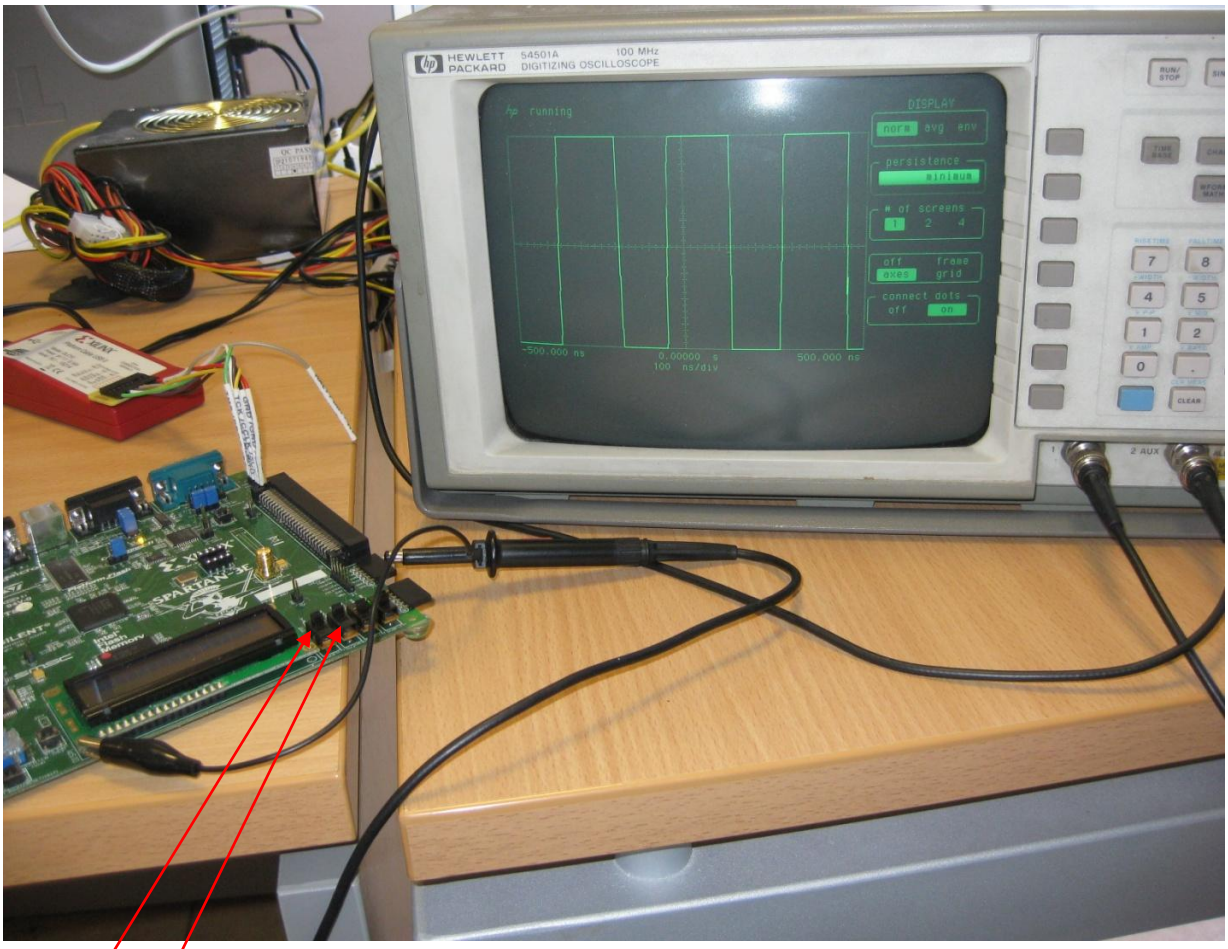
Σχήμα 2.8: Υλοποίηση αρχιτεκτονικής στο FPGA Spartan 3E και απεικόνιση στον παλμογράφο

Όπως βλέπουμε και παραπάνω αντιστοιχίσαμε την είσοδο του VHDL κώδικα, δηλαδή το επιθυμητό duty cycle, με τους 4 διακόπτες και για να δώσουμε duty cycle 50% (D_{IN} = 1000) σηκώσαμε μόνο τον πρώτο διακόπτη. Αυτό φαίνεται καλύτερα και στην παρακάτω εικόνα:



Σχήμα 2.9: Αποτελέσματα πειραματικής διαδικασίας για duty cycle 50%

Τέλος για μεγαλύτερη πληρότητα παραθέτουμε το πειραματικό αποτέλεσμα και για Duty Cycle 75%. Τώρα είναι οι δύο πρώτοι διακόπτες ON όπως φαίνεται και παρακάτω:



1ος και 2ος Διακόπτης ON(1100 => Duty Cycle 75%).

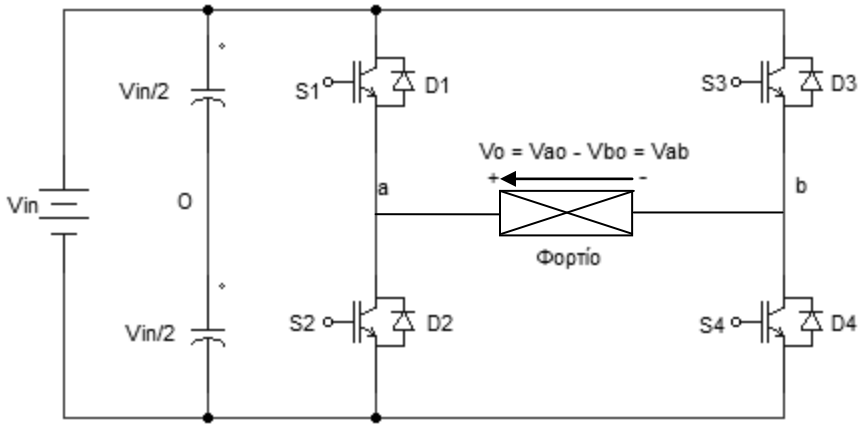
Σχήμα 2.10: Αποτελέσματα πειραματικής διαδικασίας για duty cycle 75%

ΚΕΦΑΛΑΙΟ 3^{ο}

ΥΛΟΠΟΙΗΣΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΤΗΣ SPWM ΤΕΧΝΙΚΗΣ

3.1 Εισαγωγή

Πολλές είναι οι τεχνικές που υπάρχουν για τον έλεγχο της τάσης εξόδου ενός αναστροφέα. Μία από αυτές είναι και η ημιτονοειδής διαμόρφωση εύρους παλμών(SPWM). Στην τεχνική αυτή σε επίπεδο κυκλώματος ελέγχου δημιουργούνται δύο ημιτονοειδή σήματα αναφοράς, ένα για κάθε σκέλος του παρακάτω αναστροφέα:

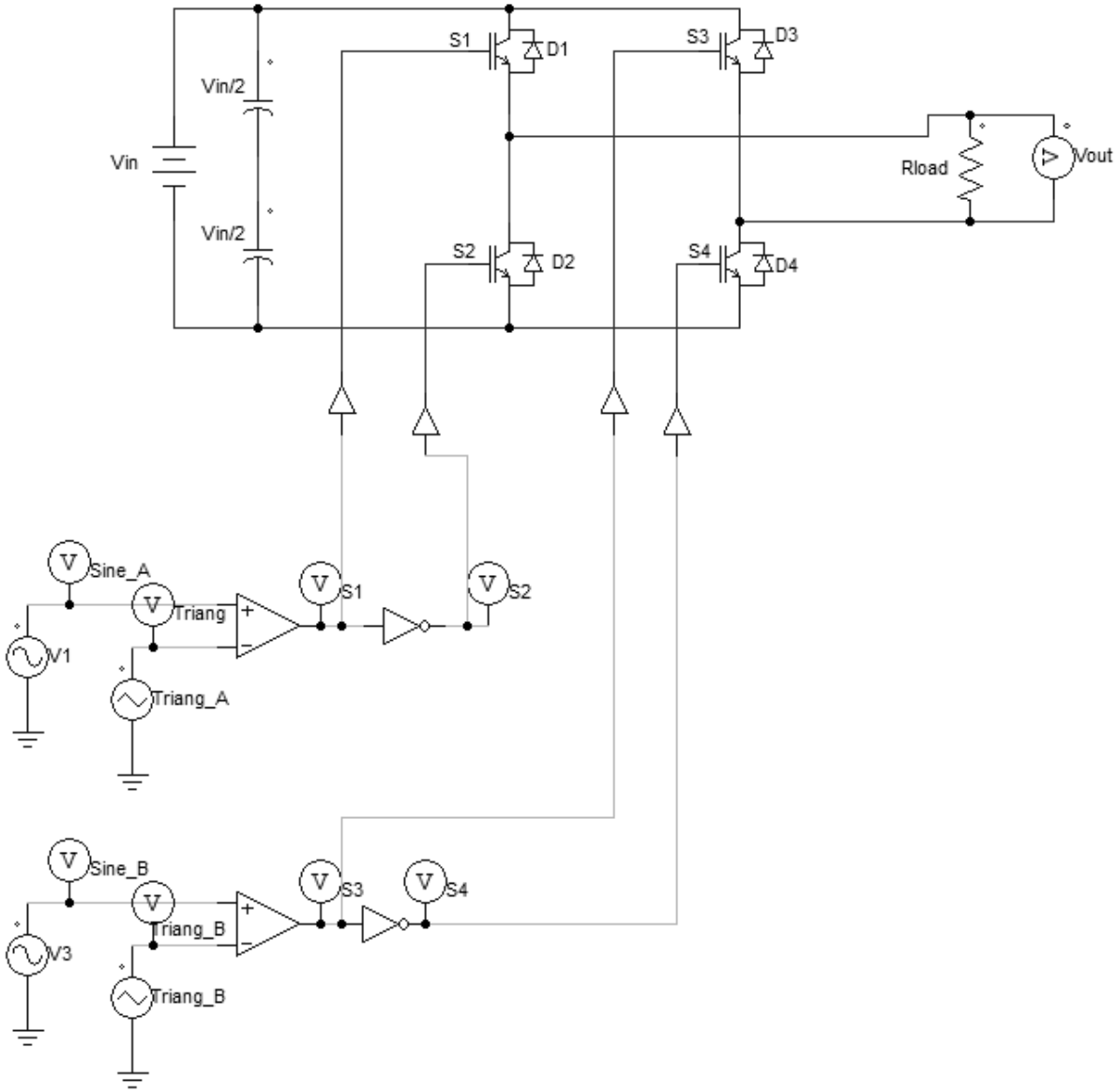


Σχήμα 3.1: Μονοφασικός αναστροφέας πλήρους γέφυρας με πηγή τάσης υλοποιημένος με ημιαγωγικούς διακόπτες τύπου IGBTs.

Τα σήματα αυτά έχουν μεταξύ τους μία διαφορά φάσης 180°. Επίσης δημιουργείται και μία τριγωνική κυματομορφή μεγάλης συχνότητας(σε σχέση με τα σήματα αναφοράς) η οποία ονομάζεται κυματομορφή φέροντος σήματος. Και τα τρία αυτά σήματα είναι συγχρονισμένα μεταξύ τους και μπορούν να δημιουργηθούν με αναλογικά ή ψηφιακά κυκλώματα. Η λογική της μεθόδου αυτής είναι η παραγωγή παλμών που εφαρμόζονται στις πύλες των τεσσάρων διακοπτών(S1,S2,S3,S4) ανάλογα με το αποτέλεσμα της σύγκρισης των σημάτων αναφοράς με το φέρον. Να τονίσουμε πως για την αποφυγή βραχυκυκλώματος, δύο διακόπτες που βρίσκονται στο ίδιο σκέλος(S1,S2 ή S3,S4) δεν θα πρέπει σε καμία περίπτωση να άγουν μαζί. Αυτός είναι και ο λόγος που σε δύο διακόπτες του ίδιου σκέλους εφαρμόζονται συμπληρωματικοί παλμοί. Οι αναφερόμενες κυματομορφές παρουσιάζονται παρακάτω με τη χρήση του εργαλείου προσομοίωσης PSim.

3.2 Προσομοίωση της SPWM τεχνικής με το PSim

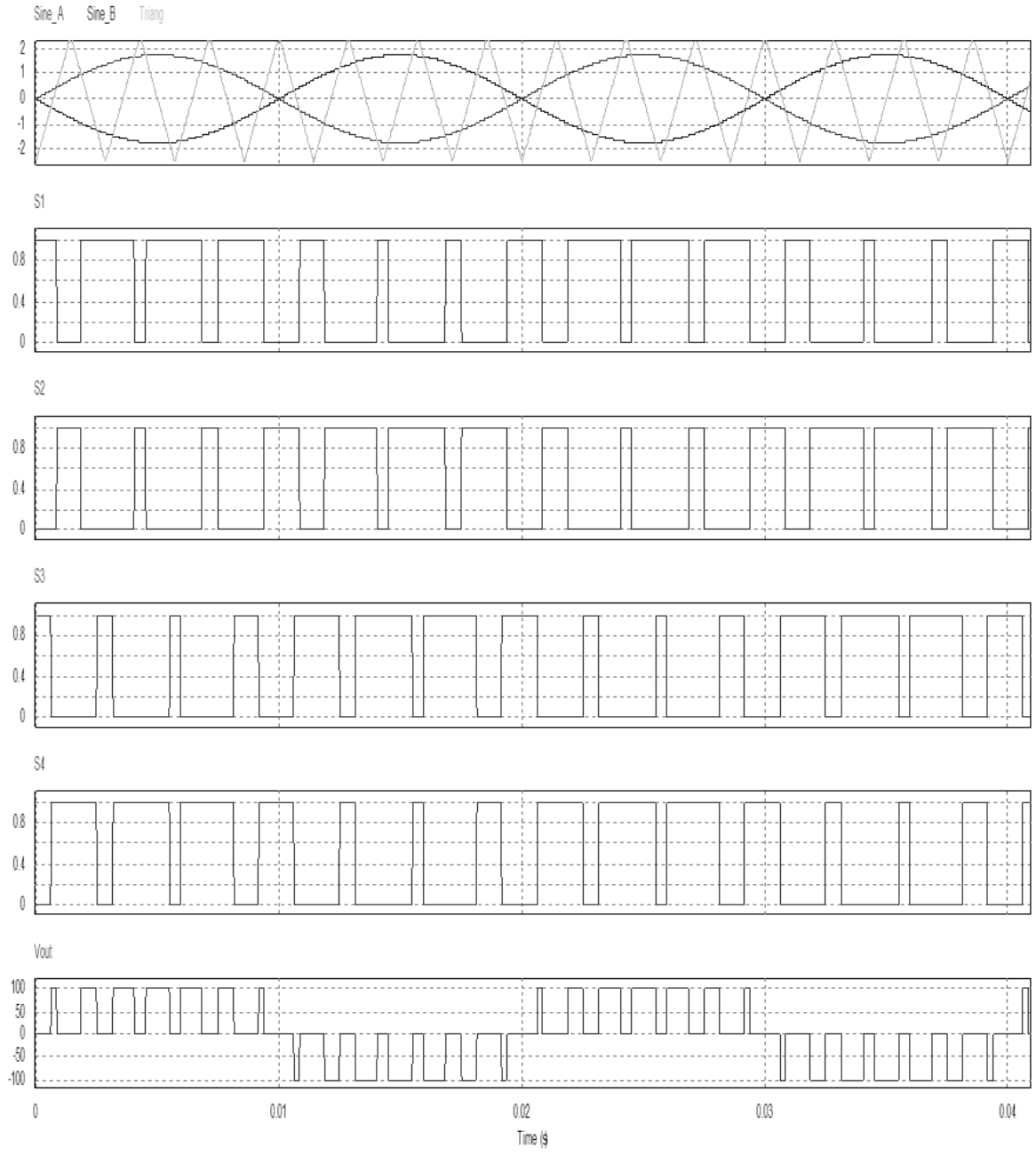
Για να γίνει περισσότερο κατανοητή η παραπάνω περιγραφή υλοποιήσαμε την SPWM τεχνική με την χρήση του εργαλείου προσομοίωσης PSim. Το κυκλωματικό ισοδύναμο για την υλοποίηση αυτή παρουσιάζεται παρακάτω:



Σχήμα 3.2: Κυκλωματικό ισοδύναμο της τεχνικής SPWM με το εργαλείο προσομοίωσης PSim

Παρατηρούμε πως οι παλμοί για τους διακόπτες S1 και S3 παράγονται από τη σύγκριση ενός ημιτονοειδούς σήματος και του τριγωνικού παλμού με τη μόνη διαφορά πως το ημιτονοειδές σήμα που χρησιμοποιείται για την παραγωγή των παλμών για τον S3 διακόπτη έχει διαφορά φάσης 180° με αυτό που χρησιμοποιείται για τον S1 διακόπτη. Επίσης παρατηρούμε πως οι παλμοί για τους διακόπτες S2 και S4 παράγονται αντιστρέφοντας αυτούς των S1 και S3 αντίστοιχα. Οι κυματομορφές

που προκύπτουν από την προσομοίωση του παραπάνω κυκλώματος παρουσιάζονται στα παρακάτω γραφήματα:



Σχήμα 3.3: Αποτελέσματα προσομοίωσης του κυκλώματος της SPWM τεχνικής με το εργαλείο προσομοίωσης Psim

Σημαντικό ρόλο στην SPWM τεχνική διαδραματίζει ο συντελεστής διαμόρφωσης πλάτους m_a όπου ισούται με τον λόγο του πλάτους του ημιτονοειδούς σήματος αναφοράς(A_r) προς το πλάτος του τριγωνικού σήματος(Ac). Δηλαδή ισχύει:

- $m_a = \frac{A_r}{A_c}$ όπου $0 < m_a < 1$.

Επίσης χαρακτηριστικό μέγεθος της τεχνικής αυτής αποτελεί και ο συντελεστής διαμόρφωσης συχνότητας που ορίζεται ως ο λόγος της συχνότητας του φέροντος σήματος(f_c) προς τη συχνότητα του σήματος αναφοράς(f_r). Δηλαδή ισχύει:

- $m_f = \frac{f_c}{f_r}$

Όσον αφορά στις συχνότητες αυτές αξίζει να σημειωθούν τα παρακάτω:

1) Η συχνότητα του σήματος αναφοράς f_r προσδιορίζει την επιθυμητή συχνότητα της τάσης εξόδου του αναστροφέα f_o.
2) Η συχνότητα του σήματος φέροντος f_c προσδιορίζει την επιθυμητή διακοπτική συχνότητα των ημιαγωγικών διακοπτών του αναστροφέα καθώς και την συχνότητα των ανωτέρων αρμονικών συνιστωσών.

3.3 Ψηφιακή υλοποίηση της SPWM τεχνικής

Στην παρούσα διπλωματική εργασία παρουσιάζονται δύο τρόποι ψηφιακής υλοποίησης της SPWM τεχνικής. Και οι δύο υλοποιούνται με την γλώσσα περιγραφής υλικού VHDL. Η κύρια διαφορά συνίσταται στο γεγονός πως στη μία περίπτωση ο υπολογισμός της τιμής του ημιτόνου γίνεται σε κάθε χτύπο ρολογιού για την πρώτη μισή περίοδο του ημιτονοειδούς σήματος ενώ στην άλλη 1000 δείγματα ημιτόνου έχουν αποθηκευτή σε μία μνήμη ROM και διαβάζονται ανά 500 χτύπους ρολογιού όπως θα παρουσιαστεί αναλυτικά παρακάτω.

Αρχικά θα αναφερθούμε στα στοιχεία που είναι κοινά και στις δύο υλοποιήσεις.

- **Συχνότητα Ημιτονοειδούς Σήματος:**

  Η συχνότητα του ημιτονοειδούς σήματος που χρησιμοποιούμε και στις δύο υλοποιήσεις είναι ίση με 50 Hz. Για να υλοποιήσουμε την συχνότητα αυτή χρησιμοποιούμε ένα μετρητή ο οποίος αυξάνει κατά ένα σε κάθε θετική ακμή του ρολογιού της σχεδίασης. Δηλαδή αυξάνεται κατά ένα σε κάθε περίοδο του ρολογιού. Επειδή λοιπόν το ρολόι των FPGA που χρησιμοποιούμε τίθεται ίσο με 20ns ο μετρητής ο οποίος θα χρησιμοποιήσουμε, για την υλοποίηση των 50 Hz, θα πρέπει να μετρά μέχρι την τιμή που προκύπτει από την παρακάτω πράξη:

  $\frac{1}{50Hz} = 20ns * T_{counter}$ => $T_{counter}$ = 1.000.000

Αυτό πρακτικά σημαίνει πως για να ολοκληρωθεί μία περίοδος του ημιτονοειδούς σήματος θα πρέπει ο μετρητής να κάνει 1.000.000 βήματα. Στην σχεδίασή μας ο μετρητής, για να ολοκληρωθεί μία περίοδος, πρέπει ξεκινώντας από το -250.000 να φτάσει στο 250.000 και στη συνέχεια να γυρίσει πάλι στο -250.000. Αυτό υλοποιείται με το παρακάτω τμήμα κώδικα:

```
---------------------------------- Sinusoidal Frequency = 50Hz ----------------------------------------------------

 PROCESS(CLK,reset)

  variable count2: INTEGER RANGE -250000 TO 250000 :=0;

  begin

  if(reset = '1') then

   if(CLK'event and CLK='1') then  --- Σε κάθε θετική ακμή του ρολογιού ο counter αυξάνεται ή μειώνεται

    if(up1 = '1') then                  ---  ανάλογα με την τιμή του σήματος up1.

      count2:= count2 + 1;

      if(count2 = 250000) then

        up1<='0';

      end if;

    else

      count2:= count2 - 1;

      if(count2 = -250000) then

        up1<='1';

      end if;

    end if;

   end if;

  else

   up1<= '1';

  end if;

end process;

-------------------------------------------------------------------------------------------------------------------
```

- **Συχνότητα Τριγωνικού Σήματος:**

Η συχνότητα του τριγωνικού σήματος που χρησιμοποιούμε και στις δύο υλοποιήσεις είναι ίση με 5κHz. Αυτή υλοποιείται όπως και η αντίστοιχη συχνότητα του ημιτονοειδούς σήματος, και βασίζεται στην παρακάτω πράξη:

$\frac{1}{5\kappa Hz} = 20\text{ns} * T_{counter} \Rightarrow T_{counter} = 10.000$

Αυτό πρακτικά σημαίνει πως για να ολοκληρωθεί μία περίοδος του τριγωνικού σήματος πρέπει ο μετρητής να κάνει 10.000 βήματα. Στην σχεδίασή μας ο μετρητής, για να ολοκληρωθεί μία περίοδος, πρέπει ξεκινώντας από το -2.500 να φτάσει στο 2.500 και στη συνέχεια να γυρίσει πάλι στο -2.500. Ο λόγος που βάζουμε το μετρητή να κινείται μεταξύ του -2.500 και του 2.500 και όχι μεταξύ του 0 και του 10.000 θα γίνει κατανοητός όταν θα εξετάσουμε τον τρόπο υλοποίησης του τριγωνικού σήματος. Η περίοδος του τριγωνικού σήματος υλοποιείται με το παρακάτω τμήμα κώδικα:

```
---------------------------------- Triangular Frequency = 5kHz ----------------------------------------------

PROCESS(CLK,reset)

 variable count1: INTEGER RANGE -2500 TO 2500;

 variable up: STD_LOGIC;

 begin

  if(CLK'event and CLK='1') then    -- Σε κάθε θετική ακμή του ρολογιού ο μετρητής αυξάνεται ή μειώνεται

   if(reset = '1') then                    -- ανάλογα με την τιμή του σήματος up.

    if (up = '1') then

     count1:= count1+1;

     if (count1 = 2500) then

      up:='0';

      Triangular<='0';

     end if;

    else

     count1:= count1-1;

     if (count1 = -2500) then

      up:='1';
```

```
        Triangular<='1';      --Τέλος της περιόδου του τριγωνικού σήματος.
      end if;
    end if;
      else
            up:= '1';
            count1:= 0;
            Triangular<= '1';
      end if;
    end if;
end process;
```

- **Πλάτος Ημιτονοειδούς Σήματος:**

Το πλάτος του ημιτονοειδούς σήματος A_r υπολογίζεται με βάση την τιμή του συντελεστή διαμόρφωσης m_a καθώς όπως έχουμε αναφέρει και ανωτέρω ισχύει η σχέση:

$m_a = \frac{A_r}{A_c}$

Άρα ανάλογα με τον εκάστοτε συντελεστή διαμόρφωσης που θα χρησιμοποιείται, το πλάτος του ημιτονοειδούς σήματος θα δίνεται από την παρακάτω σχέση:

$A_r = m_a * A_c$

Όπου A_r: Πλάτος του ημιτονοειδούς σήματος.

A_c: Πλάτος του τριγωνικού σήματος.

m_a: Συντελεστής διαμόρφωσης.

- **Υπολογισμός Γωνιών των Ημιτονοειδών Δειγμάτων:**

Για να υπολογίσουμε τα ημιτονοειδή δείγματα που χρειαζόμαστε για τις σχεδιάσεις μας, είναι απαραίτητο να γνωρίζουμε και τις γωνίες στις οποίες αυτά αντιστοιχούν. Οι συναρτήσεις που χρησιμοποιούμε για τον υπολογισμό του ημιτόνου, τις οποίες θα δούμε παρακάτω, λαμβάνουν ως εισόδους γωνίες σε rad. Θέλουμε λοιπόν να βρούμε πόσα rad αντιστοιχούν σε κάθε περίοδο του ρολογιού της σχεδίασης. Αυτό μπορούμε εύκολα να το υπολογίσουμε αν θυμηθούμε πως ο μετρητής τον οποίο χρησιμοποιούμε για την προσομοίωση της συχνότητας του ημιτονοειδούς

σήματος χρειάζεται 500.000 βήματα για να ολοκληρωθεί η μισή περίοδος του σήματος αυτού. Άρα με απλή συλλογιστική μπορούμε να πούμε:

Βήματα Μετρητή	**rad**
500.000	3.14
1	x

Από το παραπάνω προφανώς προκύπτει πως τα rad ανά βήμα του μετρητή, ή ανά περίοδο ρολογιού, θα είναι:

$x = \frac{3.14}{500.000} \approx 6.28 * 10^{-6}$.

Αυτή λοιπόν την τιμή τη θέτουμε σε μία μεταβλητή που την ονομάζουμε rad_offset και την οποία την προσθέτουμε σε κάθε περίοδο ρολογιού στην προηγούμενη τιμή rad ώστε να λάβουμε την επόμενη. Ο υπολογισμός αυτός του rad_offset υλοποιείται στις δύο διαφορετικές σχεδιάσεις μας με τις παρακάτω δύο εντολές(Η διαφορά τους έχει να κάνει με τον τύπο των μεταβλητών που χρησιμοποιούμε):

rad_offset<= MATH_PI/(500000.0); -- Χρησιμοποιείται σε πρόγραμμα που δεν συντίθεται αλλά μας

-- παρέχει τα προς αποθήκευση σε ROM δείγματα.

rad_offset<= pi/(val_500000); -- Οι αριθμοί είναι σε μορφή σταθερής υποδιαστολής.

Τέλος ο υπολογισμός των γωνιών που χρειαζόμαστε για την εύρεση των τιμών των ημιτόνων γίνεται με την παρακάτω εντολή σε κάθε περίοδο ρολογιού:

rad<= rad + rad_offset;

- **Υλοποίηση Τριγωνικού Παλμού:**

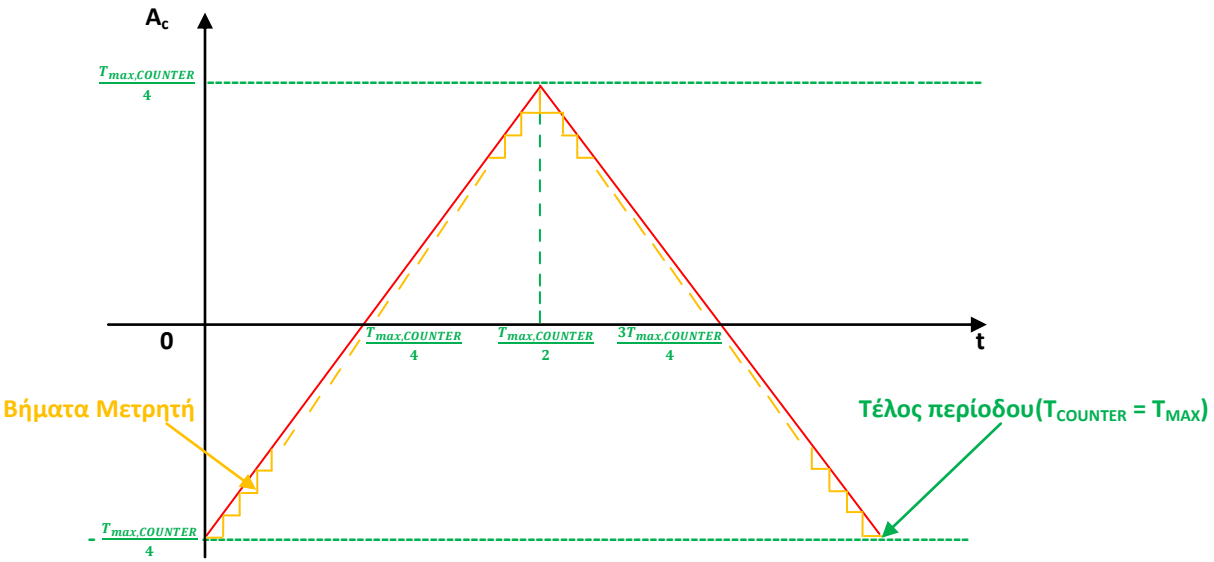
Για να αναπαρασταθεί ο τριγωνικός παλμός ψηφιακά είναι απαραίτητη η χρησιμοποίηση ενός μετρητή που θα αυξάνει έως μία μέγιστη τιμή και θα μειώνεται έως μία ελάχιστη. Οι δύο αυτές τιμές θα είναι το μέγιστο και το ελάχιστο πλάτος του τριγωνικού παλμού. Για να επιλέξουμε όμως κατάλληλα αυτές τις τιμές έτσι ώστε ο μετρητής αυτός ξεκινώντας από την ελάχιστη τιμή του να φτάνει στη μέγιστη και να επιστρέφει και πάλι στην ελάχιστη στο διάστημα μίας περιόδου θα πρέπει να λάβουμε υπ' όψιν σε πόσους κτύπους ρολογιού ολοκληρώνεται μία περίοδος του τριγωνικού σήματος. Παρατηρούμε λοιπόν πως για να ολοκληρωθεί μία περίοδος του τριγωνικού σήματος χρειάζονται 10.000 βήματα του χρησιμοποιούμενου μετρητή όπως αναλύσαμε παραπάνω. Άρα για να μπορεί ο μετρητής που προσομοιώνει το τρίγωνο να αυξάνεται από μία ελάχιστη τιμή μέχρι μία μέγιστη και να επιστρέφει και πάλι στην ελάχιστη στο διάστημα μίας περιόδου, θα πρέπει η ελάχιστη και η μέγιστη τιμή του να παίρνουν την τιμή που ορίζεται από την παρακάτω σχέση:

$A_c = \pm \frac{T_{max,counter}}{4} = \pm \frac{10.000}{4} = \pm 2.500$

Όπου: A_c = Το πλάτος του τριγωνικού σήματος.

$T_{max,counter}$ = Τα βήματα που πρέπει να κάνει ο μετρητής για την ολοκλήρωση μίας περιόδου.

Αυτό γίνεται πιο κατανοητό από το παρακάτω σχήμα:



Σχήμα 3.4: Ψηφιακή υλοποίηση του τριγωνικού παλμού

Συνεπώς, με βάση τα παραπάνω, για την υλοποίηση του τριγωνικού σήματος χρησιμοποιούμε ένα μετρητή(count_Ac) ο οποίος αυξάνεται και μειώνεται μέχρι τις τιμές 2.500 και -2.500 αντίστοιχα. Αυτόν τον μετρητή συγκρίνουμε κάθε φορά με τις εκάστοτε τιμές του ημιτονοειδούς σήματος. Η υλοποίηση αυτή γίνεται με το παρακάτω τμήμα κώδικα:

```
if(up_Ac = '1') then
    count_Ac := count_Ac + 1;
    if (count_Ac = 2500) then
     up_Ac := '0';
    end if;
else
    count_Ac:= count_Ac - 1;
    if (count_Ac = -2500) then
     up_Ac:= '1';
```

```
    end if;

 end if;
```

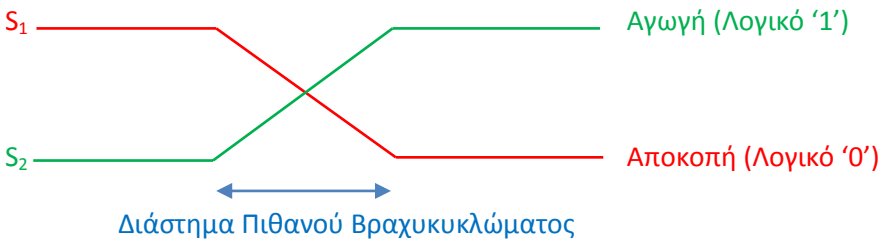
Για να θέσω τιμή στους τέσσερις διακόπτες συγκρίνω αυτό το μετρητή με τις τιμές των ημιτονοειδών σημάτων οι οποίες βρίσκονται κάθε φορά στις μεταβλητές CMPA και CMPB. Ενδεικτικά η σύγκριση αυτή γίνεται όπως παρακάτω. Να τονίσουμε πως στον τελικό κώδικα που υλοποιούμε το παρακάτω κομμάτι δεν υπάρχει αμιγώς όπως παρουσιάζεται παρακάτω αλλά έχει διάφορες προσμίξεις. Το παρουσιάζουμε όμως έτσι για να γίνει κατανοητή η διαδικασία απόδοσης τιμών στους διακόπτες:

```
if (count_Ac < CMPA ) then  -- Αν η τιμή του πρώτου ημιτόνου είναι μεγαλύτερη του τριγώνου τότε ο

    S1:= '1';                    -- διακόπτης S1 παίρνει τιμή '1' αλλιώς '0'.

else

    S1:= '0';

end if;

if(count_Ac < CMPB) then  -- Αν η τιμή του δευτέρου ημιτόνου είναι μεγαλύτερη του τριγώνου τότε ο

    S3:= '1';                    -- ο διακόπτης S2 παίρνει τιμή '1' αλλιώς '0'.

else

    S3:= '0';

end if;
```

- **Υλοποίηση ''Νεκρού'' Χρονικού Διαστήματος:**

Η υλοποίηση αυτή αφορά στην εισαγωγή ενός ''νεκρού'' χρονικού διαστήματος ανάμεσα στη χρονική στιγμή που ένας διακόπτης λαμβάνει λογικό '0' στην πύλη του, και ως εκ' τούτου τίθεται σε αποκοπή, και τη χρονική στιγμή που ο διακόπτης του ίδιου σκέλους του αναστροφέα, ο οποίος λαμβάνει πάντα συμπληρωματικές τιμές, θα λάβει λογικό '1' στην πύλη του, και ως εκ' τούτου θα τεθεί σε αγωγή.

Ο λόγος που καθιστά απαραίτητη την εισαγωγή ενός τέτοιου διαστήματος είναι πως όταν ένας διακόπτης λαμβάνει λογικό '1' ή '0' στην πύλη του, δεν οδηγείται ακαριαία σε αγωγή ή αποκοπή αντίστοιχα, αλλά μεσολαβεί ένα χρονικό διάστημα. Αν λοιπόν δύο διακόπτες του ίδιου σκέλους ενός αναστροφέα οδηγηθούν ταυτόχρονα ο ένας σε αγωγή και ο άλλος σε αποκοπή τότε στο διάστημα που θα μεσολαβήσει μέχρι να γίνει η μετάβαση αυτή θα υπάρξουν χρονικές στιγμές κατά τις οποίες και οι δύο διακόπτες θα βρίσκονται ταυτόχρονα σε αγωγή με συνέπεια την δημιουργία βραχυκυκλώματος. Η περίπτωση αυτή παρουσιάζεται και σχηματικά παρακάτω:



Σχήμα 3.5: Δημιουργία βραχυκυκλώματος εξ' αιτίας της μη ιδανικής συμπεριφοράς των ημιαγωγικών διακοπτών όταν τίθενται σε αγωγή ή αποκοπή

Στο παραπάνω σχήμα παρατηρούμε πως όταν οι δύο συμπληρωματικοί διακόπτες S_1 και S_2 οδηγούνται ταυτόχρονα από αγωγή σε αποκοπή και από αποκοπή σε αγωγή αντίστοιχα προκύπτει ένα χρονικό διάστημα στο οποίο είναι πιθανή η ταυτόχρονη αγωγή τους και η δημιουργία βραχυκυκλώματος. Για το λόγο αυτό είναι απαραίτητη η εισαγωγή του ''νεκρού'' χρονικού διαστήματος που αναφέραμε παραπάνω. Στο σημείο αυτό θα πρέπει να προσέξουμε πως για την υλοποίηση αυτή θα πρέπει να τηρούμε τον παρακάτω κανόνα:

Κανόνας: Ο διακόπτης που καθυστερεί να λάβει τιμή, λόγω της εισαγωγής της καθυστέρησης, είναι πάντα αυτός που μεταβαίνει από αποκοπή σε αγωγή.

Ο λόγος για τον οποίο πρέπει να τηρείται πάντα αυτός ο κανόνας γίνεται κατανοητός με το παρακάτω σχήμα στο οποίο εισάγουμε την καθυστέρηση αντίθετα από αυτό που ορίζει ο κανόνας. Την εισάγουμε δηλαδή στο διακόπτη που μεταβαίνει από αγωγή σε αποκοπή. Στο παρακάτω σχήμα θεωρούμε τους διακόπτες ιδανικούς, θεωρούμε δηλαδή πως μεταβαίνουν ακαριαία σε αγωγή και αποκοπή όταν λάβουν το κατάλληλο σήμα στην πύλη τους. Η θεώρηση αυτή δεν επηρεάζει το συμπέρασμα στο οποίο θέλουμε να καταλήξουμε και γίνετε για λόγους ευκολότερης κατανόησης του παραδείγματος. Η παρακάτω εικόνα προέκυψε από το εργαλείο προσομοίωσης Modelsim αφού γράψαμε έναν μικρό και απλό κώδικα για να μας δώσει το επιθυμητό για το παράδειγμα αποτέλεσμα:

Σχήμα 3.6: Βραχυκύκλωμα λόγω εισαγωγής του ''νεκρού'' χρονικού διαστήματος στον διακόπτη που μεταβαίνει από αγωγή σε αποκοπή

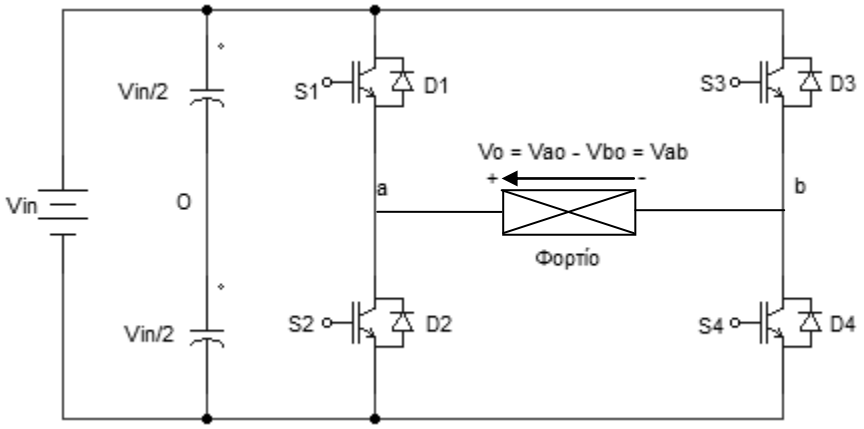
Γίνεται λοιπόν φανερό από τα παραπάνω πως για να υλοποιήσουμε το ''νεκρό'' διάστημα καθυστέρησης και να μην έχουμε τη δημιουργία βραχυκυκλώματος είναι απαραίτητη η τήρηση του κανόνος που αναφέραμε παραπάνω.

Τέλος πρέπει να μελετήσουμε και την τιμή που θα επιλέξουμε για το χρονικό αυτό διάστημα αλλά και τον τρόπο υπολογισμού αυτού. Το χρονικό διάστημα καθυστέρησης θα πρέπει να τεθεί σε μία τιμή λίγο μεγαλύτερη από τον συνολικό χρόνο που χρειάζονται οι ημιαγωγικοί διακόπτες για να πάνε σε αποκοπή. Για το λόγο αυτό η καθυστέρηση που εισάγουμε θα είναι της τάξης του 1μs. Για να την υλοποιήσουμε θα χρησιμοποιήσουμε ένα μετρητή ο οποίος θα αυξάνει κατά ένα σε κάθε περίοδο του ρολογιού μέχρι μία μέγιστη τιμή και στη συνέχεια θα μηδενίζεται για να είναι έτοιμος για την υλοποίηση της επόμενης καθυστέρησης. Η τιμή αυτή υπολογίζεται από την παρακάτω σχέση, δεδομένου ότι το ρολόι της σχεδίασής μας τίθεται ίσο με 20ns:

$1\mu s = 20ns * T_{COUNTER} \Rightarrow T_{COUNTER} = 50.$

Λόγω της πολυπλοκότητας που παρουσιάζει το συγκεκριμένο κομμάτι κώδικα αναλύουμε παρακάτω τη χρησιμότητα και λειτουργία κάποιων βασικών σημάτων που χρησιμοποιούμε:

1) **set_S1/3 :** Τα σήματα αυτά μας πληροφορούν για την τιμή που πρέπει να έχουν οι διακόπτες S1 και S3. Όπως έχουμε πει και παραπάνω οι διακόπτες S2 και S4 θα πρέπει να πάρουν τις συμπληρωματικές τιμές των S1 και S3 αντίστοιχα για την αποφυγή βραχυκυκλωμάτων. Οι διακόπτες αυτοί έχουν την διάταξη του παρακάτω σχήματος:



Σχήμα 3.7: Μονοφασικός αναστροφέας πλήρους γέφυρας με πηγή τάσης υλοποιημένος με ημιαγωγικούς διακόπτες τύπου IGBTs.

Το κρίσιμο σημείο στο κομμάτι αυτό είναι πως θέλουμε να εισάγεται η ανάλογη καθυστέρηση κάθε φορά που τα σήματα set_S1/3 αλλάζουν τιμή και όχι για όσο χρονικό διάστημα διατηρούνται σταθερά. Για το λόγο αυτό χρησιμοποιούμε τα παρακάτω σήματα.

2) **Delay_eval_S1/3 :** Τα σήματα αυτά λαμβάνουν την τιμή ‘1’ κάθε φορά που τα set_S1/3 αλλάζουν τιμή. Την τιμή αυτή την διατηρούν έως ότου ο μετρητής μέσω του οποίου υλοποιείται η καθυστέρηση λάβει την τιμή 50 και ουσιαστικά ολοκληρωθεί το επιθυμητό ‘’νεκρό’’ διάστημα(20ns * 50 = 1μs). Μόλις γίνει αυτό τίθενται στην τιμή ‘0’ και οι τιμές των διακοπτών πλέον παραμένουν σταθερές έως ότου να μεταβληθούν τα set_S1/3 και

κατά συνέπεια γίνουν ‘1’ τα Delay_eval_S1/3. Τέλος μόλις τα set_S1/3 αλλάζουν τιμή τότε θέτουμε τον διακόπτη που πάει σε αποκοπή ίσο με ‘0’ ενώ αυτόν που πρέπει να πάει σε αγωγή τον θέτουμε ίσο με ‘1’ μόλις ολοκληρωθεί το διάστημα καθυστέρησης(count_S1/3 = ‘0’). Στο παρακάτω τμήμα κώδικα, το οποίο χρησιμοποιούμε για να γίνουν περισσότερο κατανοητά τα παραπάνω, ελέγχουμε την περίπτωση που το ο διακόπτης S1 πρέπει να πάει ή βρίσκεται ήδη σε αγωγή(set_S1 = ‘1’) :

```
if(set_S1 = '1') then              -- Θέλω η καθυστέρηση να εισάγεται μόνο όταν το S1 αλλάζει τιμή
  if(Delay_eval_S1 = '1') then -- και όχι για όσο χρονικό διάστημα παραμένει στο λογικό ‘1’ ή ‘0’.
    S2<= '0';                  -- Κανόνας: Πάντα καθυστερεί ο διακόπτης που πάει σε αγωγή!!
     if(count_S1 = 50) then   -- Υλοποίηση της καθυστέρησης με χρήση μετρητή => 50*20ns = 1us!
      S1<= '1';
      Delay_eval_S1:= '0';    -- Αφού γίνει η καθυστέρηση δεν θέλω να ξαναγίνει για όσο χρονικό
      count_S1:= 0;           --  διάστημα  διατηρείται η τιμή του S1 σταθερή.
```

Λαμβάνοντας υπ’ όψιν τα όσα είπαμε παρουσιάζουμε παρακάτω το συνολικό τμήμα του κώδικα που υλοποιεί την εισαγωγή του επιθυμητού ‘’νεκρού’’ χρονικού διαστήματος:

```
--------------------------------  Deadband Modulation ------------------------------------------------------

  if(set_S1 = '1') then              -- Θέλω η καθυστέρηση να εισάγεται μόνο όταν το S1 αλλάζει τιμή
    if(Delay_eval_S1 = '1') then -- και όχι για όσο χρονικό διάστημα παραμένει στο λογικό ‘1’ ή ‘0’.
      S2<= '0';                  -- Αυτό συμβαίνει όταν το Delay_eval_S1 = ‘1’!!
      if(count_S1 = 50) then     -- Υλοποίηση της καθυστέρησης με χρήση μετρητή => 50*20ns = 1us!
        S1<= '1';
        Delay_eval_S1:= '0';     -- Αφού γίνει η καθυστέρηση δεν θέλω να ξαναγίνει για όσο χρονικό
        count_S1:= 0;            -- διάστημα διατηρείται η τιμή του S1 σταθερή.
      else
        count_S1:= count_S1 + 1;
      end if;
    end if;
  elsif(set_S1 = '0') then       -- Το S1 είναι ή πρόκειται να γίνει ‘0’.
    if(Delay_eval_S1 = '1') then
```

```
   S1<= '0';
   if(count_S1 = 50) then
    S2<= '1';
    Delay_eval_S1:= '0';
    count_S1:= 0;
   else
    count_S1:= count_S1 + 1;
   end if;
  end if;
end if;
if(set_S3 = '1') then              -- Το S3 είναι ή πρόκειται να γίνει '1'.
  if(Delay_eval_S3 = '1') then
   S4<= '0';
   if(count_S3 = 50) then
     S3<= '1';
     Delay_eval_S3:= '0';
     count_S3:= 0;
   else
     count_S3:= count_S3 + 1;
   end if;
  end if;
elsif(set_S3 = '0') then           -- Το S3 είναι ή πρόκειται να γίνει '0'.
  if(Delay_eval_S3 = '1') then
     S3<= '0';
     if(count_S3 = 50) then
       S4<= '1';
       Delay_eval_S3:= '0';
```

```
        count_S3:= 0;
      else
        count_S3:= count_S3 + 1;
      end if;
    end if;
  end if;
-------------------------------------------------------------------------------------------------------------
```

3.4 SPWM τεχνική με χρήση μνήμης ROM

Η υλοποίηση της SPWM με χρήση μνήμης ROM για την αποθήκευση ημιτονοειδών δειγμάτων αποτελεί την πρώτη από τις δύο υλοποιήσεις που έχουμε δημιουργήσει. Το κύριο χαρακτηριστικό αυτής της μεθόδου είναι ότι δεν υπολογίζει δυναμικά τα δείγματα του ημιτόνου για κάθε αύξηση του μετρητή που προσομοιώνει τον τριγωνικό παλμό αλλά έχει αποθηκευμένα σε μία μνήμη ROM 1000 δείγματα του ημιτονοειδούς σήματος.

Τα δείγματα αυτά αφορούν στο μισό της περιόδου του ημιτόνου καθώς για την άλλη μισή περίοδο διαβάζω τα ίδια στοιχεία αλλά με αρνητικό πρόσημο!

Για την υλοποίηση της τεχνικής αυτής εκτελούνται δύο βασικά βήματα, το κάθε ένα από διαφορετικό VHDL κώδικα:

1) Παραγωγή Δειγμάτων Ημιτόνου.

2) Παραγωγή Παλμών Διακοπτών.

Τα δύο βήματα αυτά εξετάζονται παρακάτω:

3.4.1 Παραγωγή Δειγμάτων Ημιτόνου

Το πρώτο βήμα το οποίο πρέπει να κάνουμε είναι να υπολογίσουμε τα στοιχεία τα οποία θα αποθηκεύσουμε στη μνήμη ROM. Αυτά δεν επιλέγονται τυχαία αλλά κατανέμονται σε ίσα διαστήματα στη μισή περίοδο του ημιτονοειδούς σήματος. Για να γίνει αυτό πρέπει να θυμηθούμε το πως υλοποιήσαμε την συχνότητα – περίοδο του ημιτονοειδούς σήματος. Αυτό το κάναμε με την χρησιμοποίηση ενός μετρητή ο οποίος έπρεπε να κάνει συνολικά 1.000.000 βήματα μέχρι να ολοκληρωθεί η περίοδος του ημιτόνου(κάνει ένα βήμα ανά περίοδο ρολογιού(20ns)). Συνεπώς για την μισή περίοδο θα πρέπει ο μετρητής να κάνει 500.000 βήματα. Τα 1.000 λοιπόν δείγματα ημιτόνου θα

πρέπει να τα κατανείμουμε στα 500.000 αυτά βήματα. Αυτό γίνεται προφανώς με την παρακάτω πράξη:

Βήματα / Δείγμα = $\frac{500.000}{1.000}$ =500.

Συνεπώς ανά 500 βήματα του μετρητή που θα χρησιμοποιώ θα υπολογίζω μία τιμή ημιτόνου. Τα σημαντικότερα στοιχεία του κώδικα που χρησιμοποιείται για την παραγωγή των δειγμάτων αυτών παρατίθενται παρακάτω:

1) Ανά 500 βήματα του μετρητή που χρησιμοποιώ υπολογίζω μία τιμή ημιτόνου. Τις τιμές αυτές τις αποθηκεύω σε ένα πίνακα sine_values ο οποίος αποτελείται από προσημασμένους αριθμούς σταθερής υποδιαστολής και δηλώνεται όπως παρακάτω:

```
TYPE vector_array is ARRAY (0 TO 999) of sfixed(1 downto -18);    -- Πίνακας 1000 στοιχείων.

signal sine_values: vector_array;
```

Ο υπολογισμός των τιμών ανά 500 βήματα γίνεται με το παρακάτω τμήμα κώδικα:

```
if(counter > 499) then

    counter:= 0;      -- Μηδενισμός του μετρητή για μέτρηση των επομένων 500 βημάτων.

    sine_values(index)<= to_sfixed(SIN(rad),1,-18);      -- Αποθήκευση της τιμής σε πίνακα.

    index:= index + 1;     -- Επόμενη θέση του πίνακα.

end if;
```

2) Όταν το index γίνει ίσο με 1000, έχω αποθηκεύσει 1000 δείγματα και έχει τελειώσει η περίοδος του ημιτονοειδούς σήματος αφού αυξάνω το index κατά 1 κάθε 500 βήματα(500 * 1000 =500.000). Αυτό υλοποιείται με το παρακάτω τμήμα κώδικα:

```
if(index = 1000) then

     stop<= '1';      -- Αν το index γίνει 1000 η διαδικασία έχει ολοκληρωθεί.

end if;
```

3) Τα δείγματα που προκύπτουν τα αποθηκεύω σε ένα αρχείο εξόδου απ’ όπου μπορώ να τα πάρω και να τα εισάγω στον κώδικά που χρησιμοποιώ για προσομοίωση και σύνθεση. Το όνομα του αρχείου αυτού είναι sine_1000. Η δήλωσή του γίνεται με το παρακάτω τμήμα κώδικα:

```
file     outfile  : text is out "sine_1000.txt";    --declare output file

variable  outline  : line;    --line number declaration
```

Η αποθήκευση στο αρχείο αυτό γίνεται με το παρακάτω τμήμα κώδικα:

```
temp:= to_slv(to_sfixed(SIN(rad),1,-18));

write(outline,chr(0)&str(temp)&chr(0)&","); -- Φέρνω τα στοιχεία στην επιθυμητή μορφή.

 writeline(outfile,outline);
```

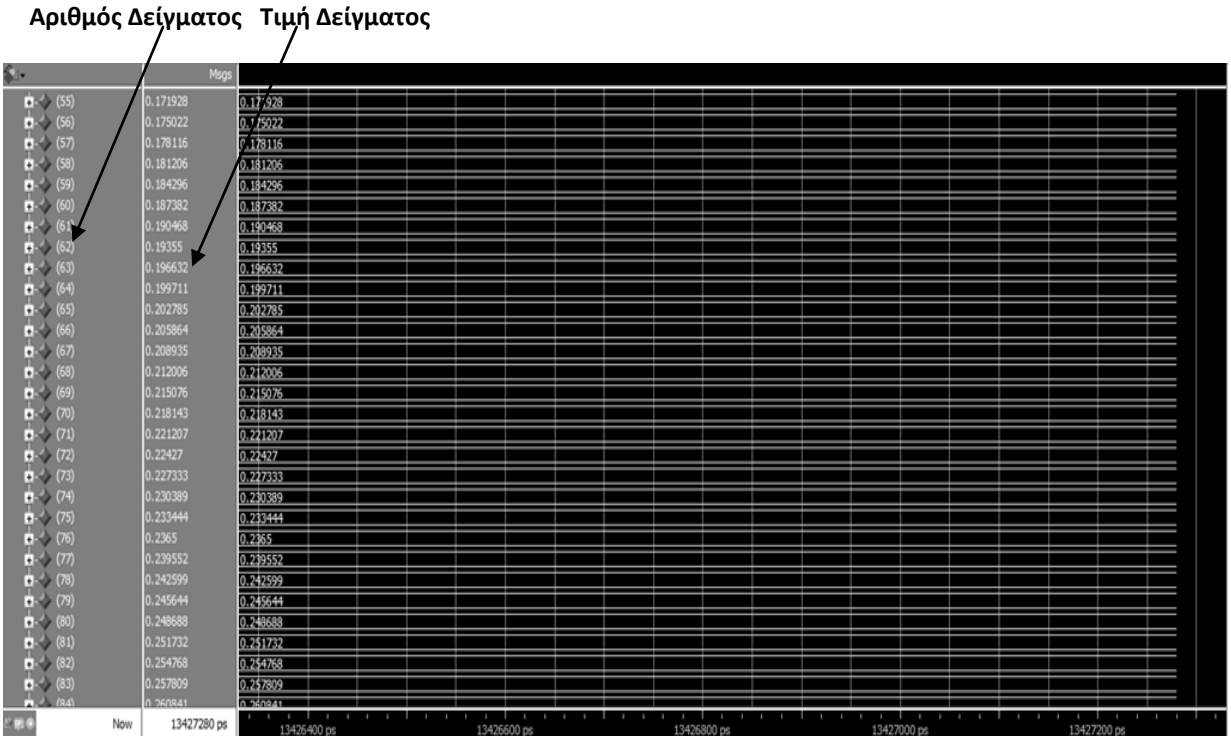
Ολόκληρος ο κώδικας που μας δίνει τα επιθυμητά δείγματα παρουσιάζεται παρακάτω. Να σημειωθεί πως ο κώδικας αυτός χρησιμοποιείται μόνο για την παραγωγή των επιθυμητών δειγμάτων και δεν είναι συνθέσιμος.

```
---------- Δήλωση Απαραίτητων Βιβλιοθηκών και Πακέτων----------

use STD.TEXTIO.all;

library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;

use work.my_data_types.all;

use ieee.math_real.all;

use ieee.fixed_float_types.all;

use ieee.fixed_pkg.all;

library std;

use std.standard.all;

use work.txt_util.all;

----------

---------- Δήλωση της Ενότητας  που θα χρησιμοποιήσουμε ----------

entity Sinusoidal_ROM_1000 is

  GENERIC (fr: REAL:= 50.0; fc: REAL:= 5000.0);

  port(clk: in STD_LOGIC);

end Sinusoidal_ROM_1000;-

----------

----------Δήλωση της Αρχιτεκτονικής που θα χρησιμοποιήσουμε ----------
```

```
Architecture behavioural of Sinusoidal_ROM_1000 is
signal samples,rad_offset: REAL;
signal rad: REAL:= 0.0;
signal stop: STD_LOGIC:= '0';
TYPE vector_array is ARRAY (0 TO 999) of sfixed(1 downto -18);
signal sine_values: vector_array;
begin
 PROCESS(clk)
   variable index: INTEGER RANGE 0 TO 1000:= 0;
   variable counter: INTEGER:= 499;
   variable temp: STD_LOGIC_VECTOR(19 downto 0);
   file     outfile  : text is out "sine_1000.txt";  --declare output file
   variable  outline  : line;   --line number declaration
   begin
   if(stop = '0') then
    if(clk'event and clk = '1') then
      rad<= rad + rad_offset;
      counter:= counter + 1;
      if(counter > 499) then
        counter:= 0;
        sine_values(index)<= to_sfixed(SIN(rad),1,-18);
        temp:= to_slv(to_sfixed(SIN(rad),1,-18));
        write(outline,chr(0)&str(temp)&chr(0)&",");
        writeline(outfile,outline);
        index:= index + 1;
        if(index = 1000) then
          stop<= '1';
```

```
        end if;
      end if;
     end if;
   end if;
 end PROCESS;

samples<= 2.0*(fc/(2.0*fr));
rad_offset<= MATH_PI/(500000.0);
end behavioural;
```

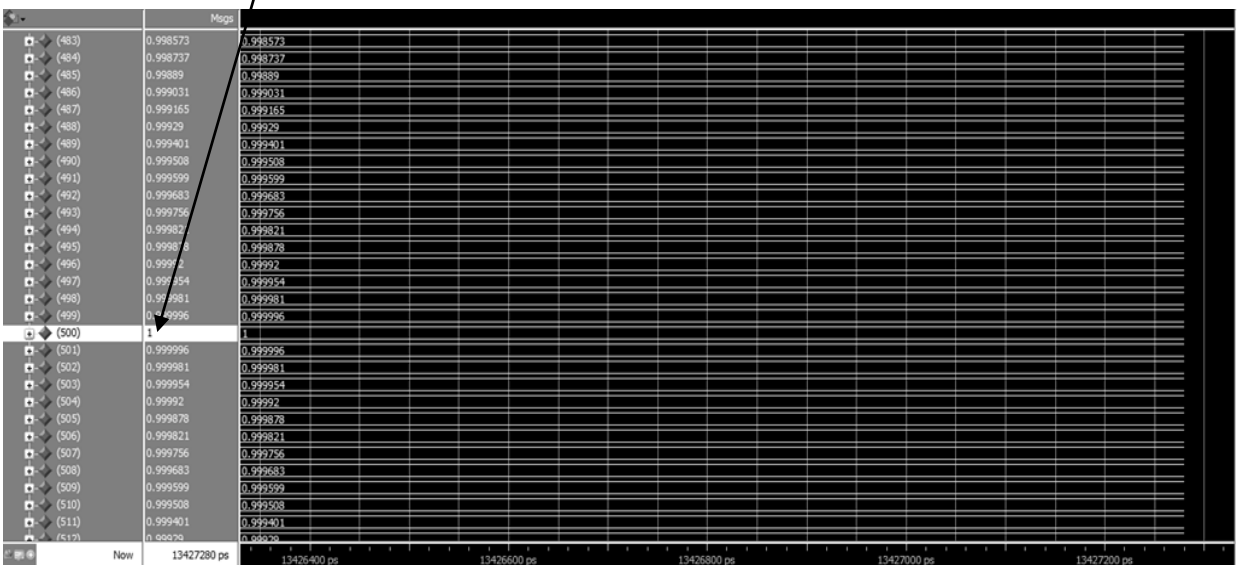
Παρακάτω παρουσιάζουμε ενδεικτικά κάποια από τα αποτελέσματα των δειγμάτων που προέκυψαν. Η προσομοίωση έγινε με το εργαλείο προσομοίωσης Modelsim Altera 10.0c(Quartus II 11.1) Starter Edition.



Σχήμα 3.8: Τιμές Αρχικών Δειγμάτων

Όπως αναμέναμε οι τιμές των αρχικών δειγμάτων έχουν τιμές μεγαλύτερες του 0 αλλά και μικρότερες του 0.5.

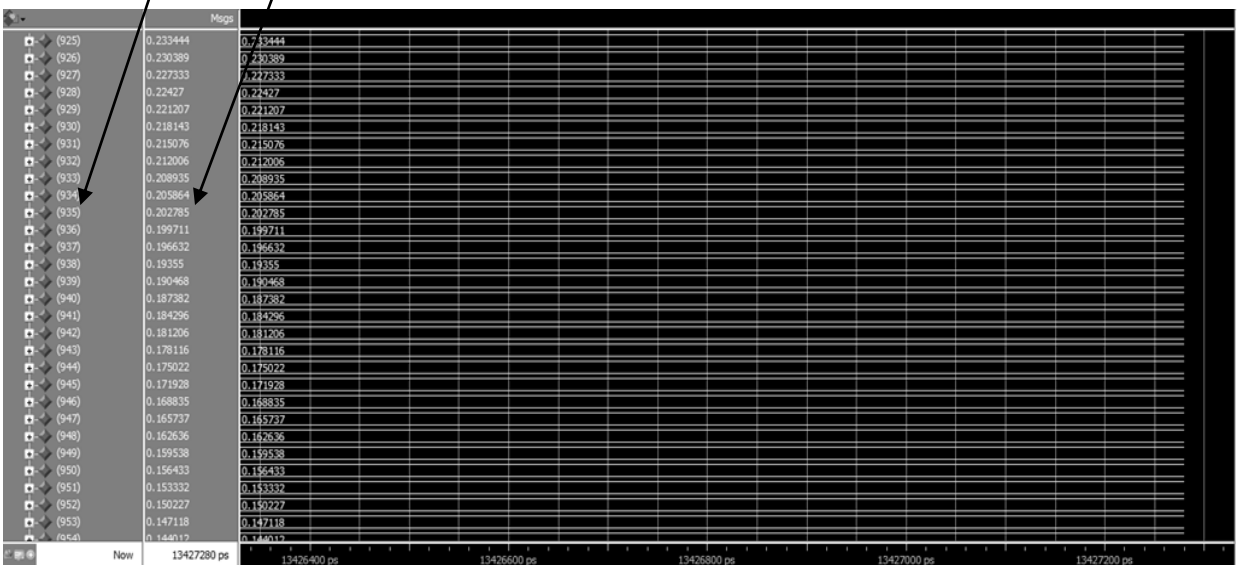
500^{ο} Δείγμα = 1(Τιμή Ημιτόνου για π/2)



Σχήμα 3.9: Τιμή Ημιτόνου 500^{ου} Δείγματος

Παρατηρούμε πως στο μεσαίο δείγμα(500) εμφανίζεται όπως αναμέναμε η τιμή 1 καθώς τότε υπολογίζεται το ημίτονο για τιμή ίση με π/2.

Αριθμός Δείγματος Τιμή Δείγματος



Σχήμα 3.10: Τιμές Τελευταίων Δειγμάτων

Παρατηρούμε πως όπως αναμέναμε οι τιμές του ημιτονοειδούς σήματος τείνουν στο μηδέν όσο πλησιάζουμε προς τα μέσα της περιόδου του.

3.4.2 Παραγωγή Παλμών Διακοπτών

Στόχος του κώδικα που θα παρουσιάσουμε στο τμήμα αυτό είναι η παραγωγή των κατάλληλων παλμών για τους ημιαγώγιμους διακόπτες S1,S2,S3,S4. Αυτό γίνεται ακολουθώντας τα παρακάτω βήματα:

1) Αποθηκεύουμε τα δείγματα που προέκυψαν από την προηγούμενη διαδικασία σε ένα πίνακα, με όνομα sine_values, που αποτελείται από 1000 προσημασμένους αριθμούς σταθερής υποδιαστολής. Αυτό γίνεται με την παρακάτω δήλωση:

```
TYPE vector_array is ARRAY (0 TO 999) of sfixed(1 downto -18);

CONSTANT sine_values: vector_array;
```

2) Όπως έχουμε αναλύσει και παραπάνω η υλοποίηση του τριγωνικού σήματος γίνεται με την χρήση ενός μετρητή(count_Ac) όπως παρακάτω:

```
if(up_Ac = '1') then
 count_Ac := count_Ac + 1;
 if (count_Ac = 2500 ) then
  up_Ac := '0';
 end if;
else
 count_Ac:= count_Ac - 1;
 if (count_Ac = -2500) then
  up_Ac:= '1';
 end if;
end if;
```

3) Επειδή τα δείγματα που έχουμε αποθηκεύσει τα έχουμε υπολογίσει ανά 500 βήματα του μετρητή τότε κάθε 500 βήματα διαβάζουμε και από μία τιμή του πίνακα με τα δείγματα. Αφού τη διαβάσουμε δημιουργούμε τα αντίστοιχα σημεία των δύο ημιτονοειδών σημάτων αναφοράς, τα οποία συγκρίνονται με το τριγωνικό σήμα. Τα δύο αυτά σημεία αποθηκεύονται στις μεταβλητές CMPA και CMPB με τα ανάλογα πρόσημα που εξαρτώνται από το αν βρισκόμαστε στο 1^{o} ή το 2^{o} μισό της περιόδου των ημιτονοειδών σημάτων. Διακρίνουμε λοιπόν τις δύο αυτές περιπτώσεις και δίνουμε τιμές στα CMPA και CMPB όπως παρακάτω:

- **1^{ο} Μισό της Περιόδου του Ημιτόνου(0 - π):**

Σε αυτή την περίπτωση ισχύει ότι το σήμα pos_index = ‘1’. Αν λοιπόν ισχύει αυτό έχουμε τις παρακάτω αποδόσεις τιμών:

```
---------------------------------- 1ο Μισό της Περιόδου του Ημιτόνου(0 - π) ----------------------------------------

    if(pos_index = '1') then

     CMPA:= "00"&Ar*sine_values(index);   -- Στο πρώτο μισό της περιόδου έχουμε θετικές τιμές στο 1ο

     CMPB:= minus1*Ar*sine_values(index); -- ημίτονο(CMPA) και αρνητικές στο 2ο(CMPB).

     index:= index + 1;

     if(index = 1000) then

      pos_index:= '0';

      index:= 0;

     end if;
```

- **2^{ο} Μισό της Περιόδου του Ημιτόνου(π – 2π):**

Σε αυτή την περίπτωση ισχύει ότι το σήμα pos_index = ‘0’. Αν λοιπόν ισχύει αυτό έχουμε τις παρακάτω αποδόσεις τιμών:

```
 else

------------------------ 2ο Μισό της Περιόδου του Ημιτόνου(π – 2π) ------------------------------------------------

     CMPA:= minus1*Ar*sine_values(index); -- Στο δευτερο μισό της περιόδου έχουμε αρνητικές τιμές

     CMPB:= "00"&Ar*sine_values(index);     -- στο 1ο ημίτονο(CMPA) και θετικές στο 2ο(CMPB).

     index:= index + 1;

     if(index = 1000) then

      pos_index:= '1';

      index:= 0;

     end if;
```

---

4) Αφού τα σήματα CMPA και CMPB λάβουν τις τιμές των αντίστοιχων ημιτόνων, συγκρίνονται με τον μετρητή που προσομοιώνει το τριγωνικό σήμα και ανάλογα με το αποτέλεσμα της σύγκρισης αποδίδουν τιμές στους τέσσερις ημιαγωγικούς διακόπτες S1,S2,S3,S4. Κατά τη σύγκριση διακρίνουμε δύο περιπτώσεις για το κάθε ζεύγος διακοπτών(S1,S2 και S3,S4). Ο κώδικας που αποδίδει τιμές στους διακόπτες ανάλογα με τα δύο πιθανά αποτελέσματα των συγκρίσεων αυτών παρουσιάζεται παρακάτω:

- **Ημιαγωγικοί Διακόπτες S1-S2:**

```
if (count_Ac < CMPA ) then
   set_S1:= '1';              -- Ο διακόπτης S1 πρέπει να τεθεί στο '1' και ο S2 στο '0'.
   Delay_eval_S1:= '1';   -- Αφορά στη δημιουργία ''νεκρού'' διαστήματος.
else                          -- count_Ac > CMPA.
   set_S1:= '0';              -- Ο διακόπτης S1 πρέπει να τεθεί στο '0' και ο S2 στο '1'.
   Delay_eval_S1:= '1';   -- Αφορά στη δημιουργία ''νεκρού'' διαστήματος
end if;
```

- **Ημιαγωγικοί Διακόπτες S3-S4:**

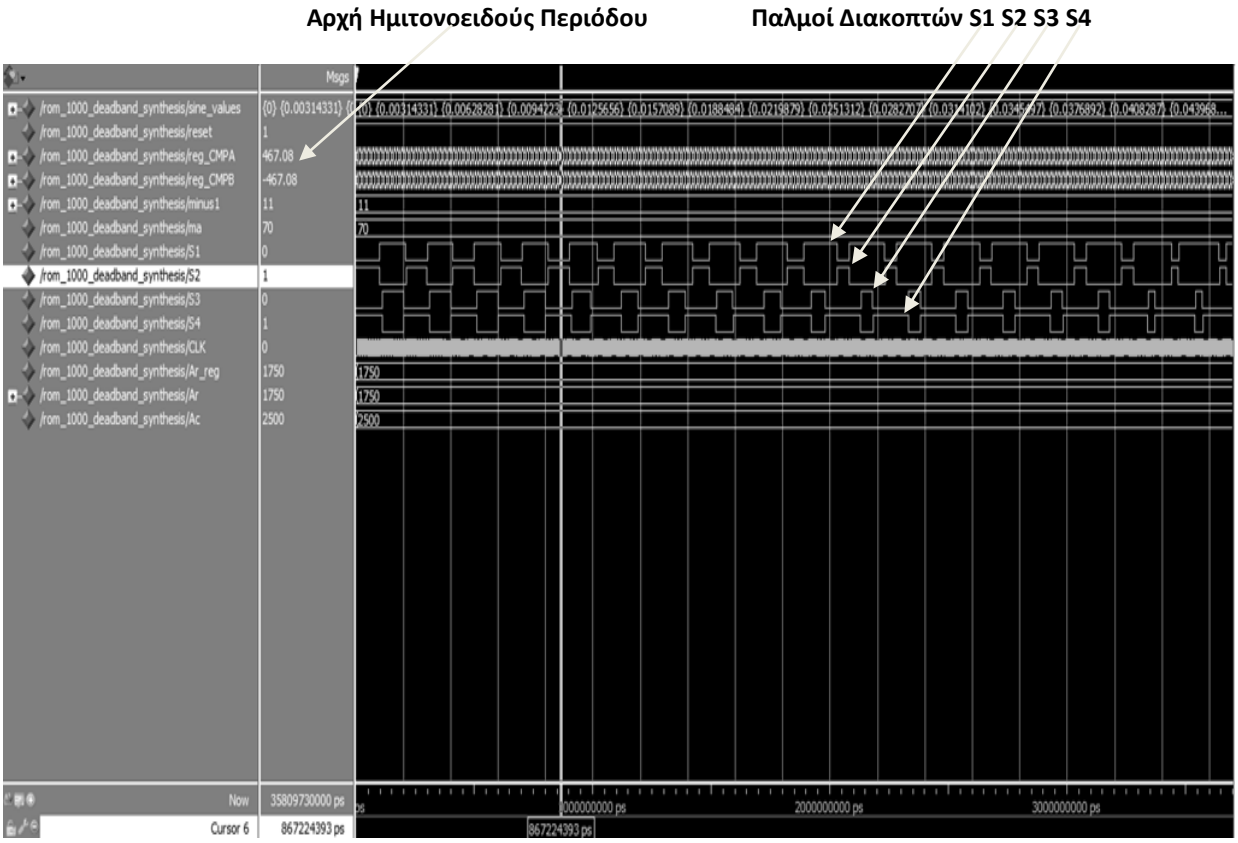
```
if(count_Ac < CMPB) then
   set_S3:= '1';            -- Ο διακόπτης S3 πρέπει να τεθεί στο '1' και ο S4 στο '0'.
   Delay_eval_S3:= '1';  -- Αφορά στη δημιουργία ''νεκρού'' διαστήματος.
else                          --  Εδώ ισχύει count_Ac > CMPB.
   set_S3:= '0';              -- Ο διακόπτης S3 πρέπει να τεθεί στο '0' και ο S4 στο '1'.
   Delay_eval_S3:= '1';   -- Αφορά στη δημιουργία ''νεκρού'' διαστήματος.
end if;
```

Έχοντας λοιπόν υπ' όψιν τα τέσσερα παραπάνω βήματα αλλά και όλα όσα έχουμε πει για τη δημιουργία των ''νεκρών'' χρονικών διαστημάτων, παρουσιάζεται στο παράρτημα ο συνολικός κώδικας ο οποίος υλοποιεί την τεχνική SPWM με τη χρήση μίας μνήμης ROM των 1000 στοιχείων.

Για την επαλήθευση της λειτουργίας του χρησιμοποιήσαμε το εργαλείο προσομοίωσης Modelsim Altera 10.0c(Quartus II 11.1) Starter Edition. Το αποτέλεσμα της προσομοίωσης της παραπάνω αρχιτεκτονικής παρουσιάζεται παρακάτω:

3.4.3 Αποτελέσματα προσομοίωσης σχεδίασης

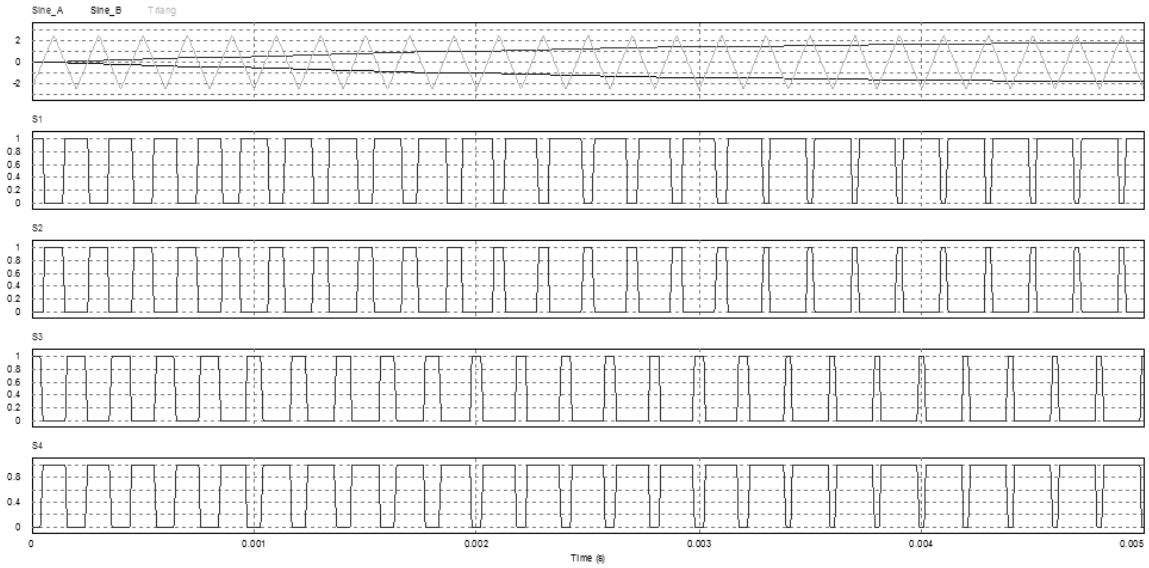
- **Αρχή Ημιτονοειδούς Περιόδου:**



Σχήμα 3.11: Παλμοί διακοπτών στην αρχή της ημιτονοειδούς περιόδου με το εργαλείο προσομοίωσης Modelsim

Από το παραπάνω σχήμα διαπιστώνουμε πως βρισκόμαστε στην έναρξη της ημιτονοειδούς περιόδου καθώς η τιμή που παίρνουν τα CMPA και CMPB στο σημείο που έχουμε τονίσει είναι ±467.08 αντίστοιχα, το οποίο είναι αρκετά μικρότερο συγκρινόμενο με το πλάτος του ημιτόνου Ar = 1750.

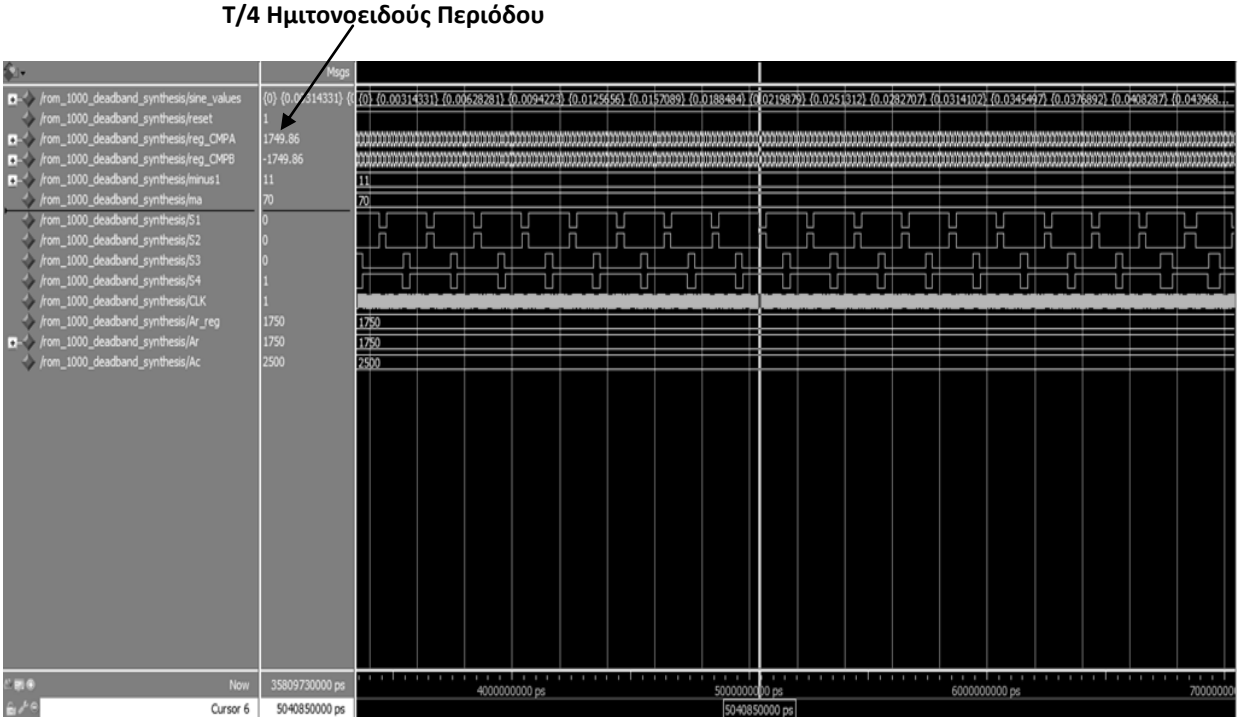
Επίσης όπως αναμένουμε, στην αρχή της περιόδου του ημιτόνου και καθώς ο χρόνος περνά μέχρι τα μέσα της περιόδου ο θετικός(μηδενικός) παλμός του διακόπτη S1(S2) αυξάνει σε διάρκεια από τον αντίστοιχο θετικό(μηδενικό) παλμό του διακόπτη S3(S4). Αυτό γίνεται ευκολότερα κατανοητό και από την αντίστοιχη προσομοίωση μέσω του εργαλείου προσομοίωσης PSim για το διάστημα 0 – T/4 της περιόδου του ημιτονοειδούς σήματος:



Σχήμα 3.12: Παλμοί των διακοπτών στην αρχή ημιτονοειδούς περιόδου με το εργαλείο προσομοίωσης PSim

Παρατηρούμε αντίστοιχα την αύξηση του εύρους του παλμού του διακόπτη S1 και την μείωση του εύρους του παλμού του διακόπτη S3.

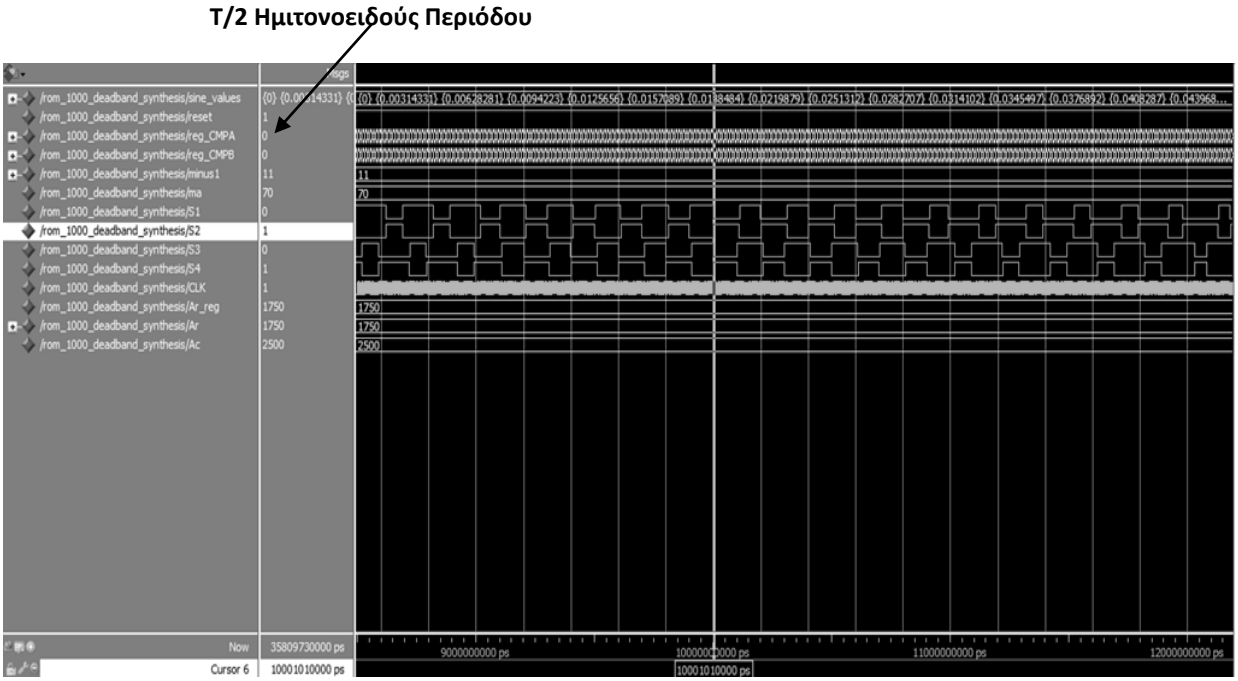
- **T/4 Ημιτονοειδούς Περιόδου:**



Σχήμα 3.13: Παλμοί διακοπτών για το T/4 της ημιτονοειδούς περιόδου με το εργαλείο προσομοίωσης Modelsim

Από το παραπάνω σχήμα διαπιστώνουμε πως βρισκόμαστε στο 1/4 της ημιτονοειδούς περιόδου καθώς η τιμή που παίρνουν τα CMPA και CMPB στο σημείο που έχουμε τονίσει είναι ±1749.86 το οποίο είναι σχεδόν ίσο με το πλάτος του ημιτόνου Ar = 1750. Σε αυτό μάλιστα το διάστημα αναμένουμε, όπως φαίνεται και παραπάνω, να έχουμε τον μεγαλύτερο σε διάρκεια θετικό παλμό για το διακόπτη S1 και το μικρότερο για το διακόπτη S3.

- **T/2 Ημιτονοειδούς Περιόδου:**



Σχήμα 3.14: Παλμοί διακοπτών για το T/2 της ημιτονοειδούς περιόδου με το εργαλείο προσομοίωσης Modelsim

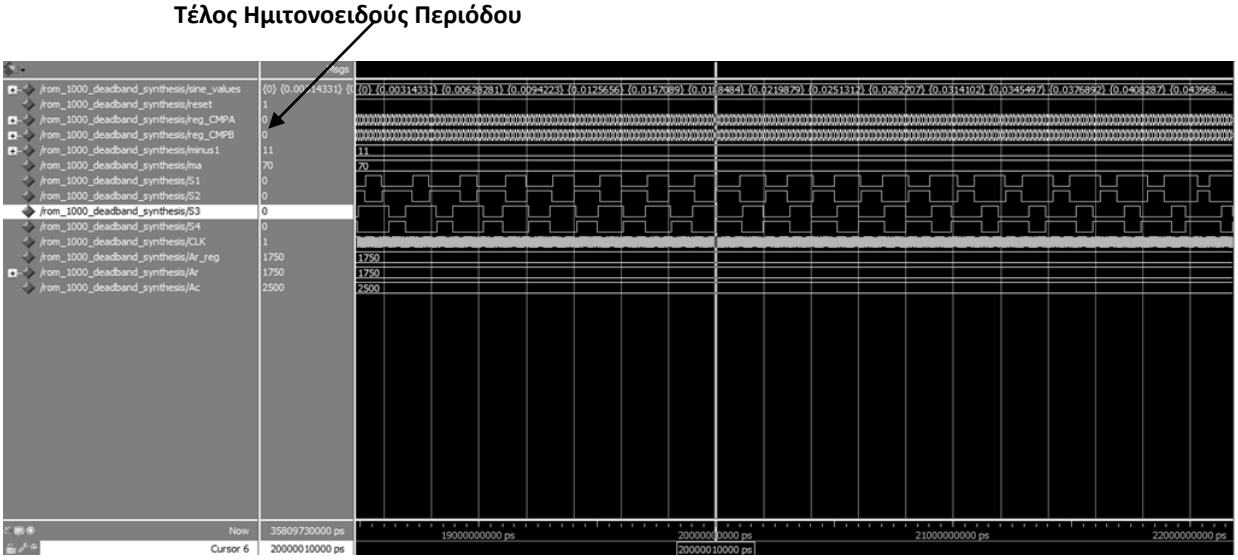
Από το παραπάνω σχήμα διαπιστώνουμε πως βρισκόμαστε στο 1/2 της ημιτονοειδούς περιόδου καθώς η τιμή που παίρνουν τα CMPA και CMPB στο σημείο που έχουμε τονίσει είναι 0 και οι τιμές του CMPA(CMPB) δεξιά του σημείου αυτού είναι θετικές(αρνητικές). Σε αυτό μάλιστα το διάστημα αναμένουμε, όπως φαίνεται και παραπάνω, η διάρκεια των θετικών παλμών των διακοπτών S1 και S3 να περίπου η ίδια.

- **3T/4 Ημιτονοειδούς Περιόδου:**

Σχήμα 3.15: Παλμοί διακοπτών για το 3T/4 της ημιτονοειδούς περιόδου με το εργαλείο προσομοίωσης Modelsim

Από το παραπάνω σχήμα διαπιστώνουμε πως βρισκόμαστε στα 3/4 της ημιτονοειδούς περιόδου καθώς στο σημείο που έχουμε τονίσει το CMPA = -1749.86 και το CMPB = 1749.86, τιμές οι οποίες είναι σχεδόν ίσες με το πλάτος του ημιτόνου Ar = 1750. Σε αυτό μάλιστα το διάστημα αναμένουμε, όπως φαίνεται και παραπάνω, να έχουμε τον μικρότερο σε διάρκεια θετικό παλμό για το διακόπτη S1 και το μεγαλύτερο για το διακόπτη S3.

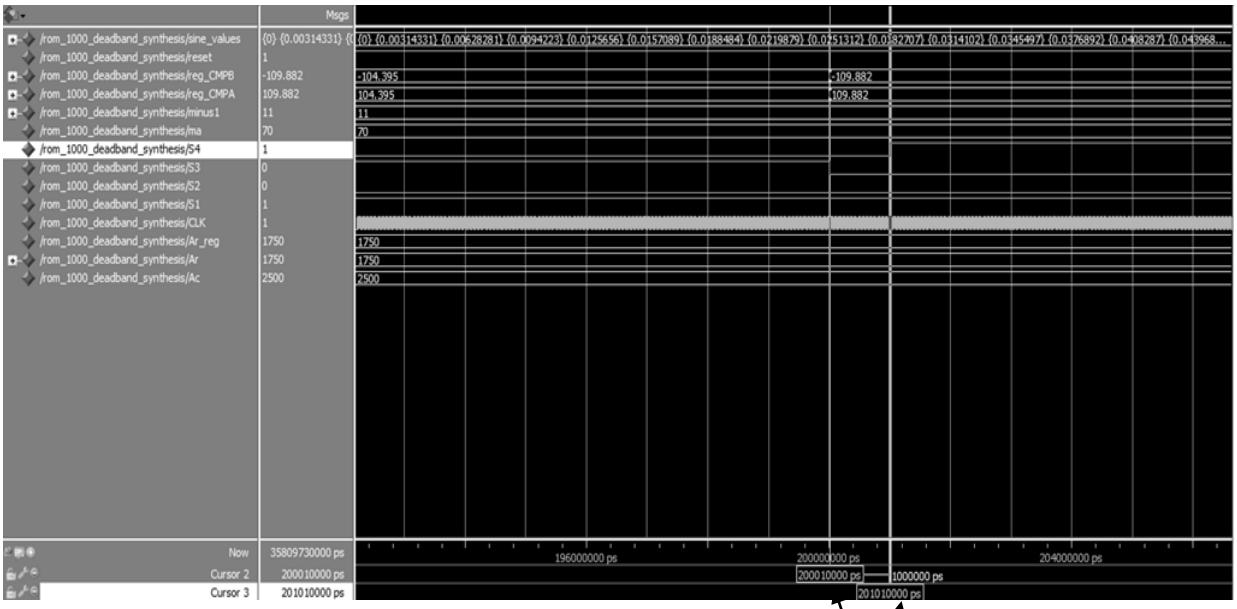
- **Τέλος Ημιτονοειδούς Περιόδου:**



Σχήμα 3.16: Παλμοί διακοπτών για το τέλος της ημιτονοειδούς περιόδου με το εργαλείο προσομοίωσης Modelsim

Από το παραπάνω σχήμα διαπιστώνουμε πως βρισκόμαστε στο τέλος της ημιτονοειδούς περιόδου καθώς η τιμή που παίρνουν τα CMPA και CMPB στο σημείο που έχουμε τονίσει είναι 0 και επίσης οι τιμές του CMPA(CMPB) δεξιά του σημεία αυτού είναι αρνητικές(θετικές). Σε αυτό μάλιστα το διάστημα αναμένουμε, όπως φαίνεται και παραπάνω, η διάρκεια των θετικών παλμών των διακοπτών S1 και S3 να περίπου η ίδια.

- **''Νεκρό'' Χρονικό Διάστημα:**



Σχήμα 3.17: ''Νεκρό'' Χρονικό Διάστημα

''Νεκρό'' Χρονικό Διάστημα

Από το παραπάνω σχήμα παρατηρούμε πως από την αποκοπή του διακόπτη S3 έως την αγωγή του διακόπτη S4(συμπληρωματικός του S3) μεσολαβεί το παρακάτω χρονικό διάστημα:

t_{DEAD} = 201010000ps – 200010000ps = 1000000ps = 1μs.

Άρα όντως το παραπάνω διάστημα είναι ίσο με το επιθυμητό(1μs).

3.4.4 Αποτελέσματα πειραματικής διαδικασίας

Στο σημείο αυτό, αφού προσομοιώσαμε τον κώδικα που υλοποιεί την συγκεκριμένη SPWM τεχνική είμαστε σε θέση να ''φορτώσουμε'' τον κώδικα αυτό στο FPGA και να παρατηρήσουμε στον παλμογράφο αν όντως τα αποτελέσματα που είδαμε στην προσομοίωση, ισχύουν και στην πράξη. Για το σκοπό αυτό χρησιμοποιήσαμε το FPGA Spartan 6 XC6LX16-CS324 το οποίο μας δίνει την δυνατότητα να χρησιμοποιήσουμε και αριθμούς κινητής υποδιαστολής στη σχεδίασή μας. Για να υλοποιήσουμε τη σχεδίαση στο FPGA ακολουθήσαμε όλα τα βήματα που περιγράψαμε στο 1^{ο} κεφάλαιο και συνδέσαμε μέσω του αρχείου περιορισμού τα σήματα που υπολογίζουν τους παλμούς

των τεσσάρων διακοπτών(S1,S2,S3,S4) με τέσσερεις ακροδέκτες του FPGA. Αυτό το πραγματοποιήσαμε με την παρακάτω δήλωση στο αρχείο περιορισμού:

NET "S1" LOC = "T12" |IOSTANDARD = "LVCMOS33"; -- Σύνδεση Σήματος S1

NET "S2" LOC = "V12" |IOSTANDARD = "LVCMOS33"; -- Σύνδεση Σήματος S2

NET "S3" LOC = "N10" |IOSTANDARD = "LVCMOS33"; -- Σύνδεση Σήματος S3

NET "S4" LOC = "P11" |IOSTANDARD = "LVCMOS33"; -- Σύνδεση Σήματος S4

Επίσης έχουμε αναφέρει και παραπάνω πως το ρολόι της σχεδίασής μας τίθεται ίσο με 20ns. Η περίοδος αυτή του ρολογιού, καθορίζεται από το αρχείο περιορισμού μέσω της αποδιδόμενης σε αυτό συχνότητας. Για να πετύχουμε λοιπόν περίοδο ίση με 20ns θα πρέπει η συχνότητα που θα θέσουμε να είναι ίση με:

$f_{clk} = \frac{1}{20\ ns}$ = 50.000 kHz.

Αυτό πραγματοποιείται με την παρακάτω δήλωση:

TIMESPEC TS_sys_clk_pin = PERIOD "CLK" 50000 kHz HIGH 50%; -- Ρολόι Περιόδου 20 ns.

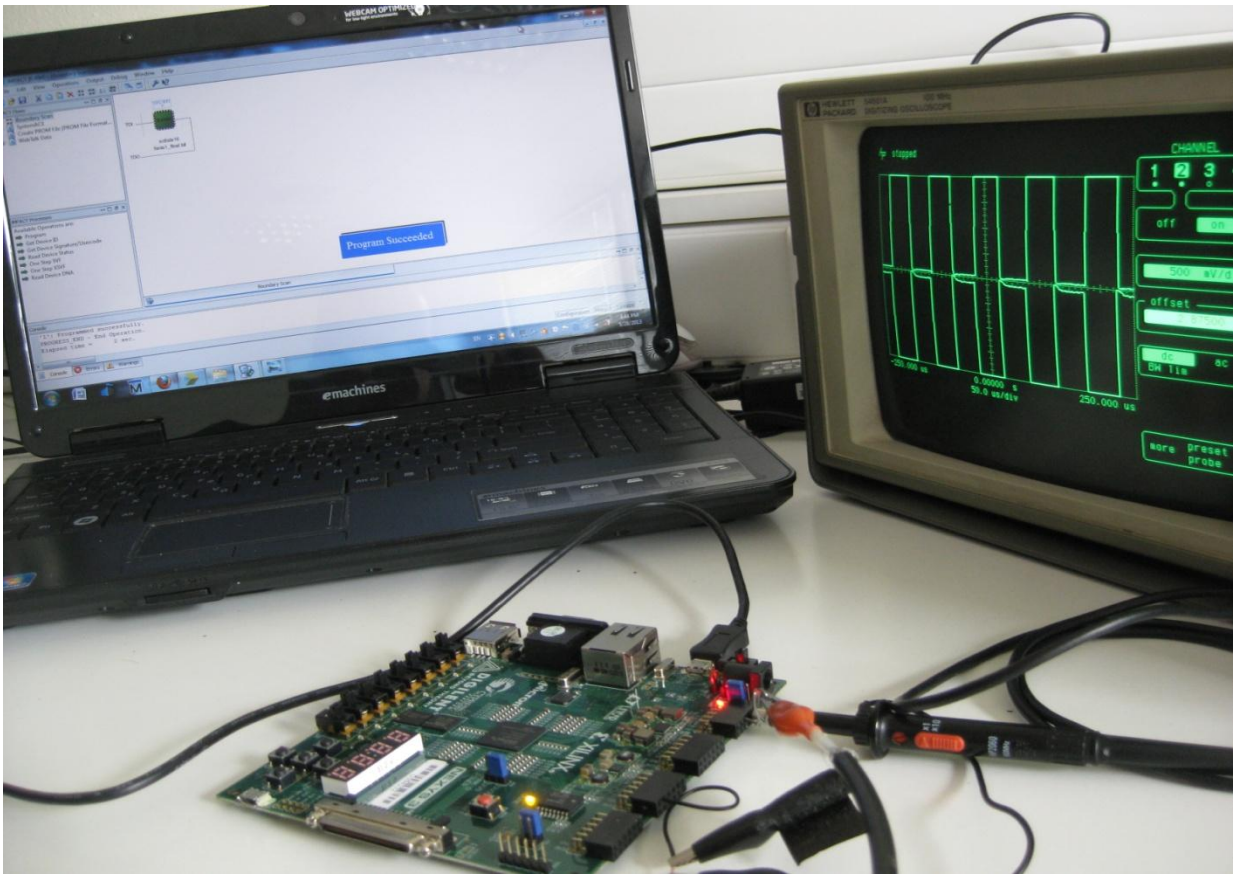
Μετά την παραγωγή του αρχείου προγραμματισμού της συσκευής ‘’τρέξαμε’’ την εφαρμογή μας και προέκυψαν στον παλμογράφο τα αποτελέσματα , τα οποία αφορούν στους παλμούς οδήγησης των ημιαγωγικών διακοπτών. Αρχικά παραθέτουμε παρακάτω τα συστατικά στοιχεία της όλης εφαρμογής τα οποία είναι:

1) Ο υπολογιστής(laptop) μέσω του οποίου ‘’φορτώνεται’’ η σχεδίασή μας στο FPGA.

2) Το FPGA το οποίο χρησιμοποιούμε(Spartan 6 XC6LX16-CS324).

3) Ο παλμογράφος στον οποίο αποτυπώνουμε τα αποτελέσματα.

Τα τρία αυτά στοιχεία παρουσιάζονται στο παρακάτω σχήμα:



Σχήμα 3.18: Υλοποίηση αρχιτεκτονικής στο FPGA Spartan 6 XC6LX16-CS324 και απεικόνιση στον παλμογράφο

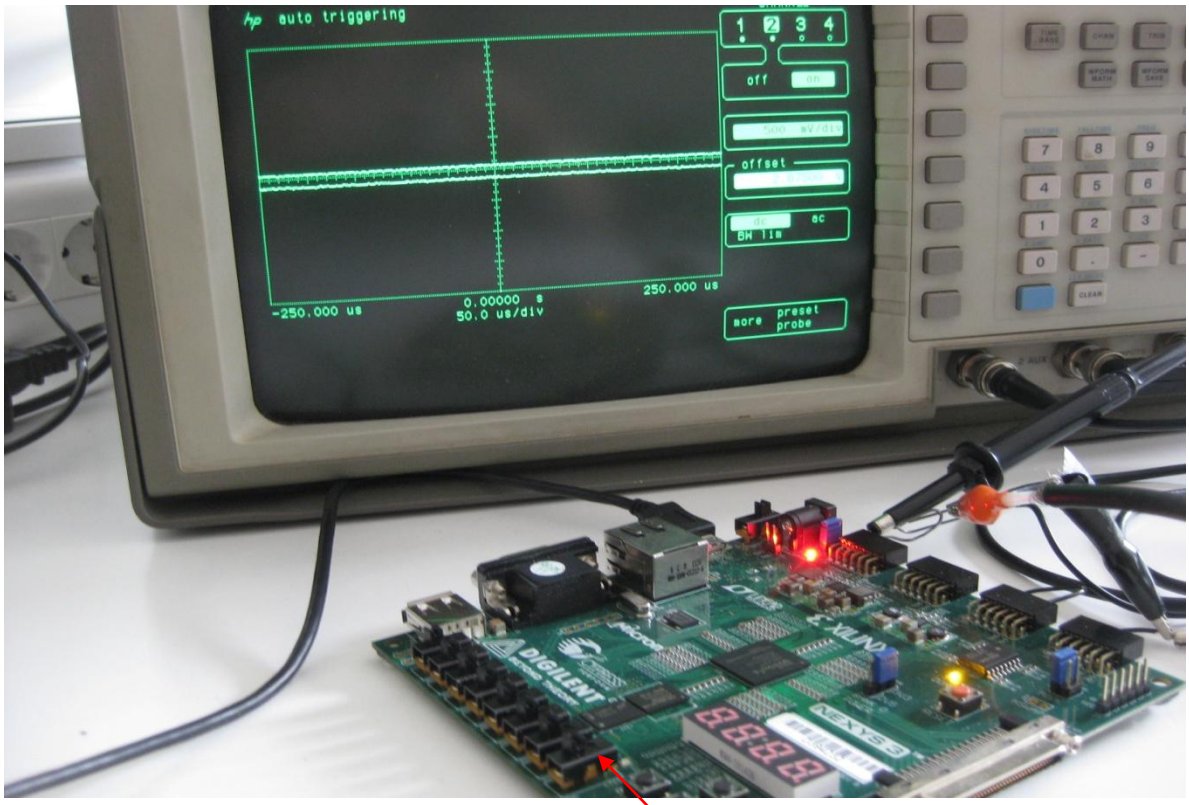
Στη συνέχεια περάσαμε και τις δύο διαφορετικές υλοποιήσεις μας στο FPGA και παρατηρήσαμε τα αποτελέσματα και πάλι στον παλμογράφο. Τα αποτελέσματα της SPWM τεχνικής με χρήση μνήμης ROM παρουσιάζονται παρακάτω.

Σε κάθε ένα από τα στιγμιότυπα που παρουσιάζουμε τοποθετούμε τα δύο συμπληρωματικά κάθε φορά σήματα, δηλαδή τους παλμούς που προορίζονται για την οδήγηση ή των διακοπτών S1 και S2 ή των διακοπτών S3 και S4. Παρακάτω παραθέτουμε τα αποτελέσματα για τα ζεύγη S1,S2 και S3,S4.

Αρχικά παρουσιάζουμε την γενική εικόνα που παρατηρούμε στον παλμογράφο όταν έχουμε και όταν δεν έχουμε ενεργοποιημένο το reset και στη συνέχεια παραθέτουμε τα στιγμιότυπα από τους παλμούς που προκύπτουν για τα ζεύγη διακοπτών S1-S2 και S3-S4.

- **Λειτουργία Reset:**
  Το σήμα reset για να είναι εμφανής η λειτουργία του το έχουμε συνδέσει με τον $1^{ο}$ διακόπτη(sw<0>) του FPGA μέσω του αρχείου περιορισμού με την παρακάτω δήλωση:
  NET "reset" LOC = "T10" | IOSTANDARD = "LVCMOS33"; -- Δήλωση Σήματος reset

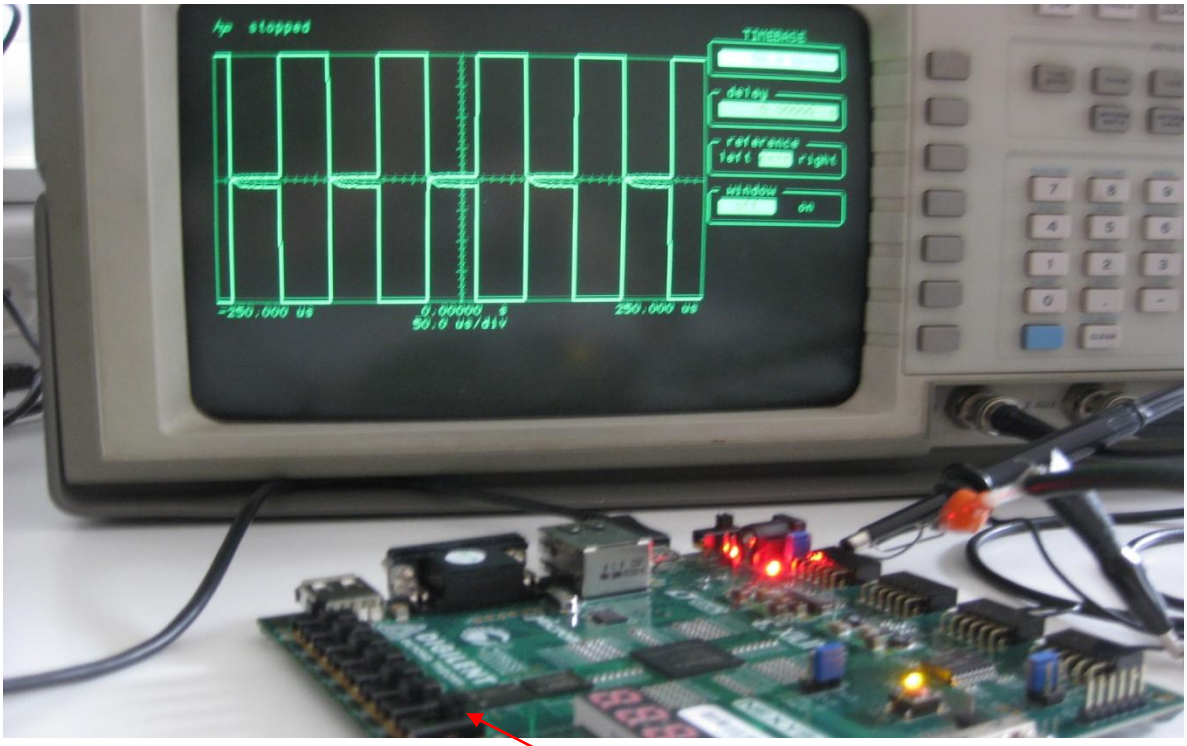
  Αρχικά παραθέτουμε τη λειτουργία με το reset ενεργοποιημένο(reset = ‘0’). Αυτή παρουσιάζεται στο παρακάτω σχήμα:



Σχήμα 3.19: Απεικόνιση στον παλμογράφο των αποτελεσμάτων για τους παλμούς οδήγησης των διακοπτών για ενεργοποιημένο το reset

Παρατηρούμε από το παραπάνω σχήμα πως όπως αναμέναμε, όταν ενεργοποιούμε το σήμα reset τότε ο ένας παλμός τίθεται ίσος με το '0', στην προκείμενη περίπτωση ο S1, ενώ ο άλλος τίθεται ίσος με το '1'(S2) .

Αν απενεργοποιήσουμε τώρα τη λειτουργία του σήματος reset(reset = '1') τότε προκύπτουν όπως αναμέναμε συμπληρωματικοί παλμοί μεταβαλλόμενου εύρους όπως φαίνεται και στο παρακάτω σχήμα:

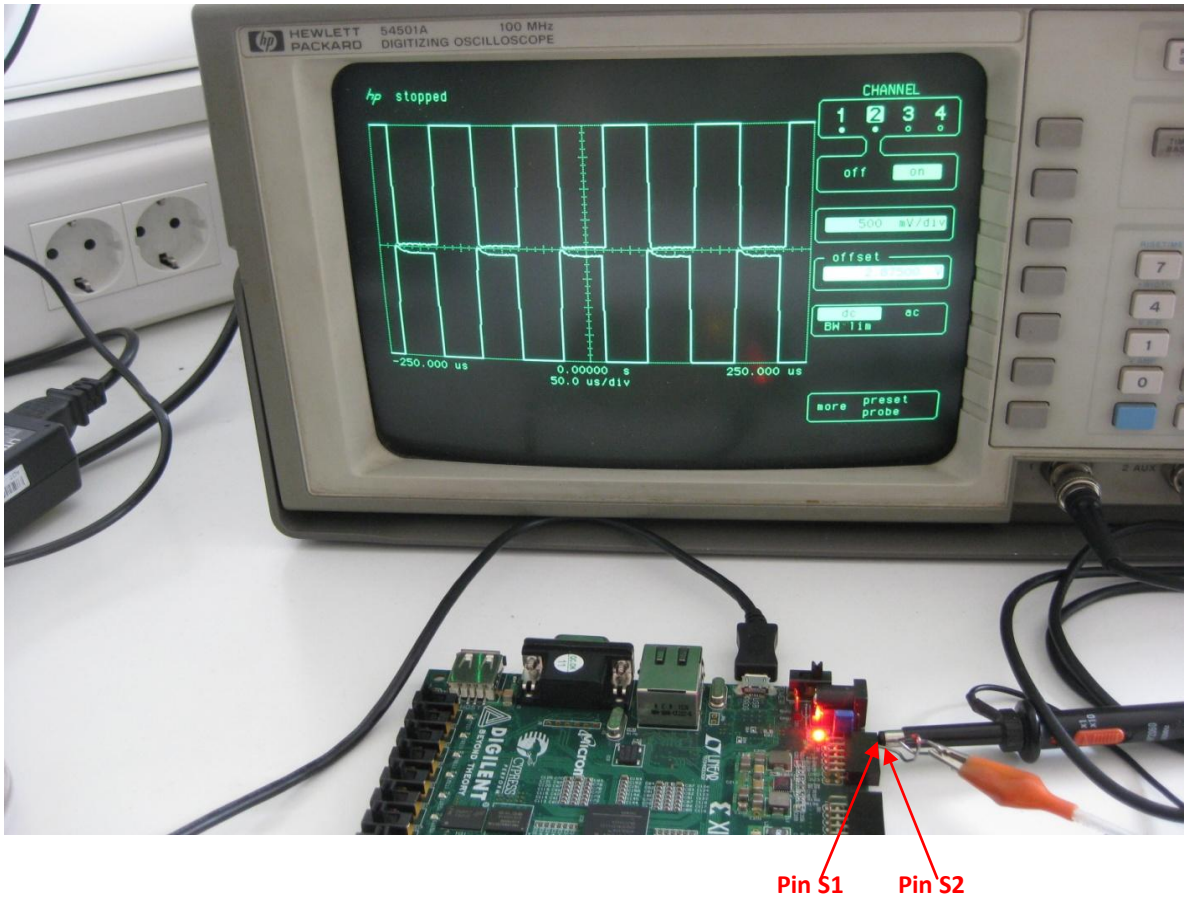


Σχήμα 3.20: Απεικόνιση στον παλμογράφο των αποτελεσμάτων για τους παλμούς οδήγησης των διακοπτών για απενεργοποιημένο το reset

Αφού παρατηρήσαμε τη λειτουργία με ενεργοποιημένο και απενεργοποιημένο το reset, το απενεργοποιούμε και παρατηρούμε τα αποτελέσματα που προκύπτουν για τους παλμούς οδήγησης των ζευγών διακοπτών S1-S2 και S3-S4.

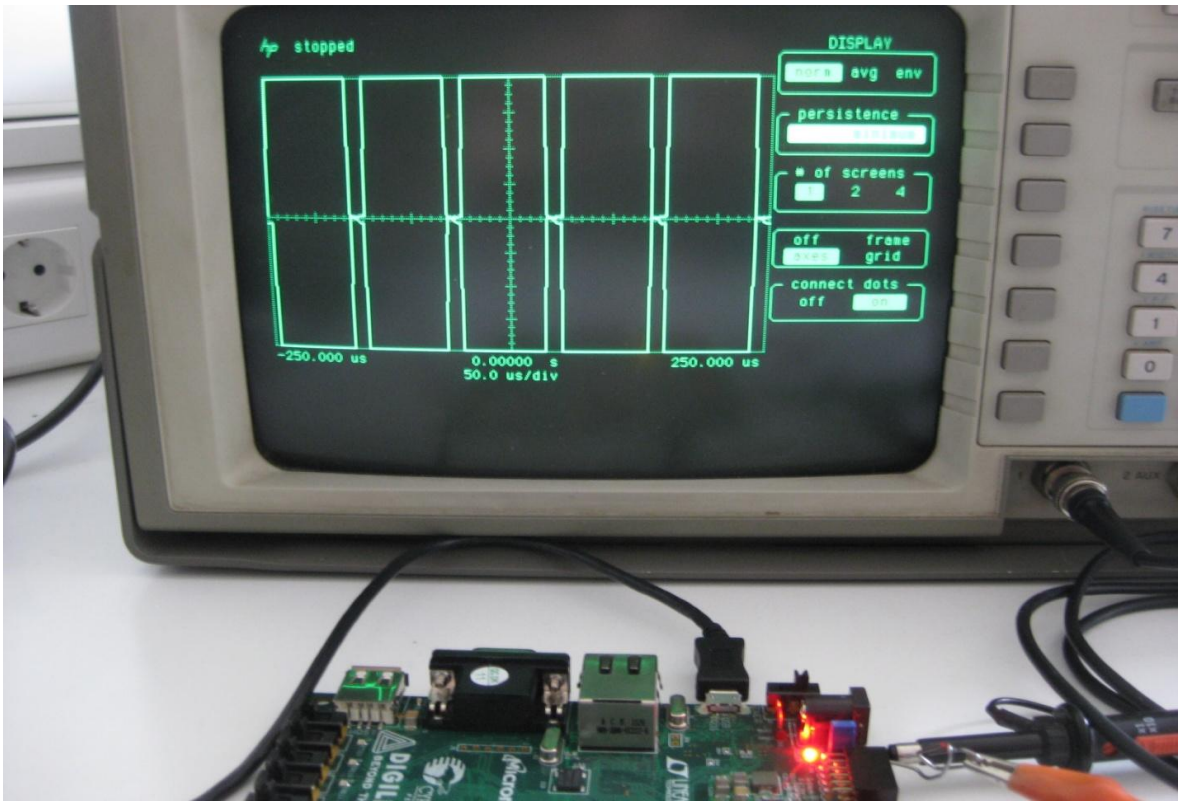
- **Ζεύγος S1 – S2:**

Η απεικόνιση των παλμών αυτών φαίνεται στο παρακάτω σχήμα:



Σχήμα 3.21: Απεικόνιση στον παλμογράφο του συμπληρωματικού ζεύγους παλμών S1,S2 της SPWM τεχνικής με χρήση ROM για εύρος παλμών περίπου 50%

Για μεγαλύτερη πληρότητα και κατανόηση των αποτελεσμάτων παραθέτουμε και το παρακάτω σχήμα στο οποίο παρατηρούμε πως το εύρος των παλμών για τον διακόπτη S1 είναι περίπου ίσο με 75% ενώ για τον διακόπτη S2 ίσο με 25%:

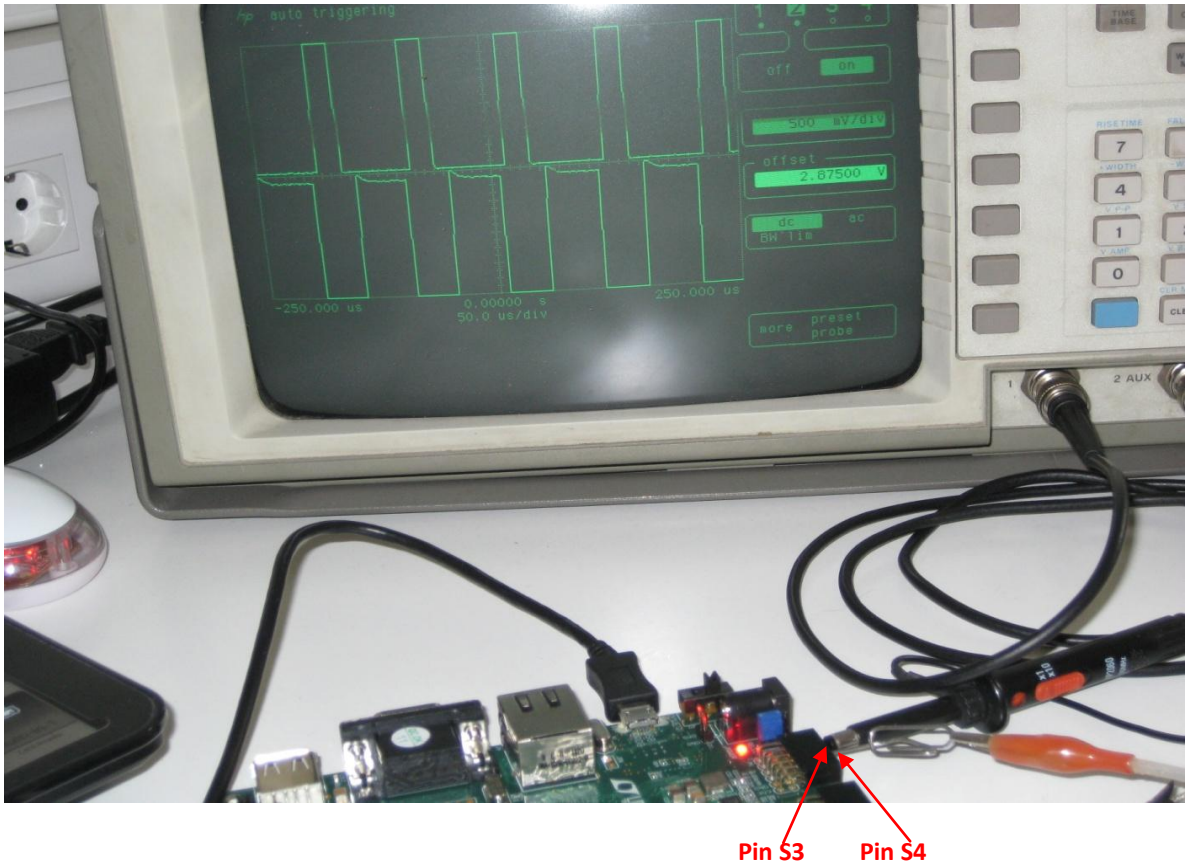


Σχήμα 3.22: Απεικόνιση στον παλμογράφο του συμπληρωματικού ζεύγους παλμών S1,S2 της SPWM τεχνικής με χρήση ROM για εύρος παλμών S1 περίπου 75% και S2 25%

Αυτό που αξίζει να παρατηρήσουμε και από τις δύο παραπάνω απεικονίσεις είναι η συμπληρωματικότητα των παλμών, η οποία είναι καθοριστική για τη σωστή λειτουργία της σχεδίασής μας.

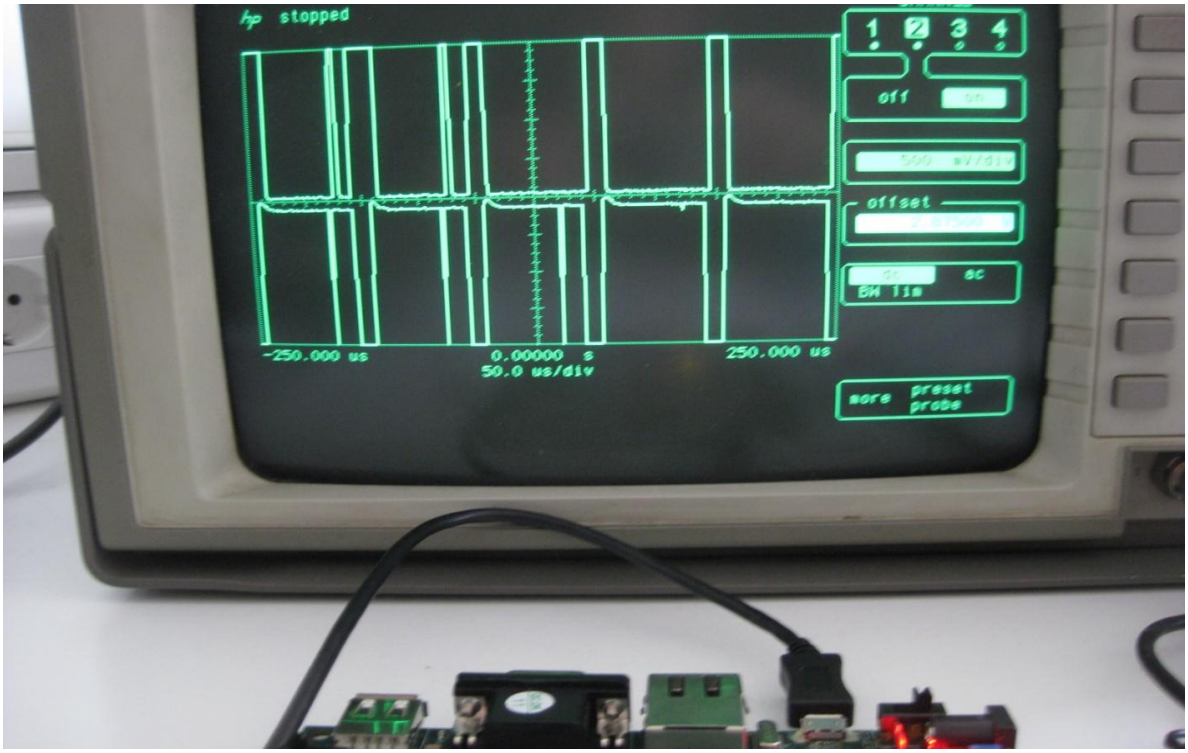
- **Ζεύγος S3 – S4:**

Η απεικόνιση των παλμών αυτών φαίνεται στα παρακάτω σχήματα:



Σχήμα 3.23: Απεικόνιση στον παλμογράφου του συμπληρωματικού ζεύγους παλμών S3-S4 της SPWM τεχνικής με χρήση ROM

Για μεγαλύτερη πληρότητα και κατανόηση των αποτελεσμάτων παραθέτουμε και το παρακάτω σχήμα στο οποίο παρατηρούμε πως το εύρος των παλμών για τον διακόπτη S3 είναι περίπου ίσο με 25% ενώ για τον διακόπτη S2 ίσο με 75%:



Σχήμα 3.24: Απεικόνιση στον παλμογράφο του συμπληρωματικού ζεύγους παλμών S1,S2 της SPWM τεχνικής με χρήση ROM για εύρος παλμών S3 περίπου 25% και S2 75%

Και πάλι αυτό που παρατηρούμε και αξίζει να τονιστεί από τις παραπάνω απεικονίσεις είναι η συμπληρωματικότητα που παρουσιάζουν οι δύο παλμοί, πράγμα απαραίτητο για την αποφυγή βραχυκυκλωμάτων.

3.5 SPWM τεχνική με δυναμικό υπολογισμό δειγμάτων

Η δεύτερη αυτή τεχνική υλοποίησης της SPWM τεχνικής παρουσιάζει μία κύρια διαφορά από την προηγούμενη υλοποίηση. Οι τιμές των δειγμάτων του ημιτόνου υπολογίζονται με την χρήση ενός πυρήνα του FPGA για κάθε περίοδο του ρολογιού μέχρι να φτάσουμε στη μέση της περιόδου του ημιτονοειδούς σήματος. Επίσης αποθηκεύουμε κατά το διάστημα αυτό σε δύο πίνακες τις τιμές των ημιτόνων οι οποίες είναι ίσες(σχεδόν) με τον μετρητή που προσομοιώνει το τριγωνικό σήμα. Τις τιμές αυτές χρησιμοποιούμε στο υπόλοιπο της διαμόρφωσης. Συνεπώς, γίνεται κατανοητό πως στην υλοποίηση αυτή κατά το πρώτο μισό της περιόδου του ημιτονοειδούς σήματος, δεν συγκρίνουμε το τριγωνικό σήμα με τα ημιτονοειδή δείγματα ανά 500 περιόδους ρολογιού αλλά σε κάθε περίοδο. Αυτό από τη μία μας οδηγεί σαφώς σε πιο ακριβή αποτελέσματα αλλά από την άλλη αυξάνει την πολυπλοκότητα και τους πόρους που πρέπει να διατεθούν για την υλοποίηση της σχεδίασης αυτής. Παρακάτω παραθέτουμε τα κύρια συστατικά της υλοποίησής μας, τον VHDL κώδικα της σχεδίασης όπως και τα αποτελέσματα της προσομοίωσης τα οποία αποδεικνύουν τα θεωρητικώς αναμενόμενα αποτελέσματα.

3.5.1 Σύστημα Xilinx Core Generator

Το σύστημα αυτό, το οποίο προσφέρεται από την Xilinx, επιταχύνει τον χρόνο σχεδίασης παρέχοντας πρόσβαση σε υψηλά παραμετρικοποιημένες πνευματικές ιδιοκτησίες για τα FPGA της Xilinx και περιλαμβάνεται στο εργαλείο ISE Design Suite. Προσφέρει μία πληθώρα από προσαρμόσιμες από το χρήστη συναρτήσεις όπως για παράδειγμα διάφορες μαθηματικές συναρτήσεις. Με τη χρήση αυτού του συστήματος επιτυγχάνουμε τον υπολογισμό των τιμών του ημιτόνου που επιθυμούμε. Συγκεκριμένα χρησιμοποιούμε τον πυρήνα CORDIC του οποίου η λειτουργία περιγράφεται περιληπτικά παρακάτω:

- **LogiCORE IP CORDIC v4.0**

  Ο πυρήνας CORDIC υλοποιεί ουσιαστικά έναν αλγόριθμο που επιτρέπει τον υπολογισμό τριγωνομετρικών συναρτήσεων με την χρήση απλών ολισθήσεων και προσθέσεων. Αυτό αποτελεί ένα μεγάλο πλεονέκτημα για υλοποιήσεις στο υλικό καθώς οι πολλαπλασιαστές απαιτούν χρήση πολλών πόρων. Η λειτουργική αυτή μονάδα υπολογίζει τιμές ημιτόνων και συνημίτονων σε μορφή σταθερής υποδιαστολής αφού της δώσουμε ως είσοδο μία γωνία σε rad. Ο λόγος που χρειάζεται να χρησιμοποιήσουμε τον πυρήνα αυτό για την παραγωγή του ημιτόνου είναι ότι στην γλώσσα περιγραφής υλικού VHDL η συνάρτηση που δίνει το ημίτονο δεν είναι συνθέσιμη και κατά συνέπεια μπορεί να χρησιμοποιηθεί μόνο για λόγους προσομοίωσης και όχι σε πραγματική υλοποίηση σε FPGA.

3.5.2 Ανάλυση VHDL κώδικα σχεδίασης

Κάποια τμήματα του κώδικα αυτού είναι όμοια με αντίστοιχα της υλοποίησης με την χρήση ROM και δεν θα περιγραφούν ξανά καθώς αυτό έχει γίνει παραπάνω. Στόχος του κώδικα που θα παρουσιάσουμε στο τμήμα αυτό είναι η παραγωγή των κατάλληλων παλμών για τους ημιαγώγιμους διακόπτες S1,S2,S3,S4. Αυτό γίνεται ακολουθώντας τα παρακάτω βήματα:

1) Το πρώτο βήμα που κάνουμε είναι να κατασκευάσουμε τον πυρήνα CORDIC για την παραγωγή των επιθυμητών τιμών των ημιτόνων. Αυτό γίνεται με την χρήση του εργαλείου ISE Design Suite και μετά από κάποια συγκεκριμένη διαδικασία προκύπτει το παρακάτω λειτουργικό μοντέλο σε VHDL:

```
LIBRARY ieee;

USE ieee.std_logic_1164.ALL;

-- synthesis translate_off

LIBRARY XilinxCoreLib;

-- synthesis translate_on
```

```
ENTITY sin_function IS
  PORT (
    phase_in : IN STD_LOGIC_VECTOR(30 DOWNTO 0);
    y_out : OUT STD_LOGIC_VECTOR(15 DOWNTO 0);
    clk : IN STD_LOGIC
  );
END sin_function_float;

ARCHITECTURE sin_function_a OF sin_function IS
-- synthesis translate_off
COMPONENT wrapped_sin_function
  PORT (
    phase_in : IN STD_LOGIC_VECTOR(30 DOWNTO 0);
    y_out : OUT STD_LOGIC_VECTOR(15 DOWNTO 0);
    clk : IN STD_LOGIC
  );
END COMPONENT;

-- Configuration specification
  FOR ALL : wrapped_sin_function USE ENTITY XilinxCoreLib.cordic_v4_0(behavioral)
    GENERIC MAP (
      c_architecture => 2,
      c_coarse_rotate => 1,
      c_cordic_function => 2,
      c_data_format => 0,
      c_family => "virtex7",
      c_has_ce => 0,
```

```
    c_has_clk => 1,
    c_has_nd => 0,
    c_has_phase_in => 1,
    c_has_phase_out => 0,
    c_has_rdy => 0,
    c_has_rfd => 0,
    c_has_sclr => 0,
    c_has_x_in => 0,
    c_has_x_out => 0,
    c_has_y_in => 0,
    c_has_y_out => 1,
    c_input_width => 31,
    c_iterations => 0,
    c_output_width => 16,
    c_phase_format => 0,
    c_pipeline_mode => -2,
    c_precision => 0,
    c_reg_inputs => 1,
    c_reg_outputs => 1,
    c_round_mode => 0,
    c_scale_comp => 0,
    c_xdevicefamily => "virtex7"
  );
-- synthesis translate_on
BEGIN
-- synthesis translate_off
U0 : wrapped_sin_function
```

```
 PORT MAP (
   phase_in => phase_in,
   y_out => y_out,
   clk => clk
 );
-- synthesis translate_on
END sin_function_float;
```

Το μόνο λοιπόν που έχουμε να κάνουμε για να το χρησιμοποιήσουμε στον κώδικά μας είναι να δηλώσουμε το στοιχείο αυτό στην αρχιτεκτονική που χρησιμοποιούμε και να του δώσουμε κατάλληλα σήματα εξόδου και εισόδου.

2) Διακρίνουμε στο πρόγραμμά μας δύο βασικές περιόδους λειτουργίας οι οποίες διακρίνονται μεταξύ τους με βάση την τιμή του σήματος SAMPLES_READY και παρουσιάζονται παρακάτω:

- **1η Περίοδος Λειτουργίας(SAMPLES_READY = '0'):**

  Η περίοδος αυτή εκτείνεται από την χρονική στιγμή που ξεκινά να ''τρέχει'' το πρόγραμμά μας μέχρι να ολοκληρωθεί η πρώτη μισή περίοδος των ημιτονοειδών σημάτων. Κατά τη φάση αυτή, σε κάθε περίοδο ρολογιού υπολογίζεται και από μία τιμή ημιτόνου η οποία πολλαπλασιαζόμενη με το πλάτος των ημιτονοειδών σημάτων(Ar) και με το κατάλληλο πρόσημο καταχωρείται στις μεταβλητές reg_CMPB και reg_CMPB οι οποίες με την σειρά τους συγκρίνονται με τον μετρητή που προσομοιώνει το τριγωνικό σήμα. Η απόδοση των τιμών υλοποιείται με το παρακάτω τμήμα κώδικα:

  ```
  reg_CMPA<= Ar*to_sfixed(sin_val,1,-14);
  reg_CMPB<= minus1*Ar*to_sfixed(sin_val,1,-14);
  ```

  Μόλις παρατηρούμε από την σύγκριση των ημιτονοειδών δειγμάτων με τον μετρητή του τριγωνικού σήματος πως η ανισότητα μεταξύ τους αλλάζει κατεύθυνση καταλαβαίνουμε πως βρισκόμαστε σε σημείο επαφής και κατά συνέπεια δίνουμε τις τιμές '0' ή '1' στα σήματα sample_A και sample_B τα οποία οδηγούν στην αποθήκευση των σημείων αυτών στους δύο πίνακες sine_values_A και sine_values_B. Γίνεται λοιπόν κατανοητό πως κατά την περίοδο αυτή αποθηκεύουμε ουσιαστικά τα σημεία επαφής των δύο ημιτονοειδών σημάτων με το τριγωνικό σήμα σε δύο πίνακες για να τα χρησιμοποιήσουμε κατά το υπόλοιπο διάστημα της εφαρμογής. Για καλύτερη κατανόηση παρουσιάζουμε ενδεικτικά το παρακάτω τμήμα κώδικα που αφορά σε μέρος μόνο της προαναφερθείσας υλοποίησης:

  ```
  if(count_Ac >= reg_CMPA) then
    sample_A<= '1';    -- Το sample_A είναι '0' για όσο ισχύει count_Ac < reg_CMPA.
  ```

```
   set_S1:= '0';
  else
   set_S1:= '1';    -- Για όσο δεν αλλάζει φορά η ανισότητα.
  end if;
  if(count_Ac >= reg_CMPB) then    -- Το sample_B είναι '0' για όσο ισχύει count_Ac < reg_CMPB.
   sample_B<= '1';
   set_S3:= '0';
  else
  set_S3:= '1';    -- Για όσο δεν αλλάζει φορά η ανισότητα.
  end if;
```

- **2η Περίοδος Λειτουργίας(SAMPLES_READY = '1'):**

  Η περίοδος αυτή ξεκινά μόλις ολοκληρωθεί το μισό της 1ης περιόδου των ημιτονοειδών σημάτων και εκτείνεται μέχρι την ολοκλήρωση της εφαρμογής. Στη περίοδο αυτή η λογική υλοποίησης είναι απλούστερη. Κάθε φορά που ο μετρητής που προσομοιώνει το τριγωνικό σήμα φτάνει στη μέγιστη ή στη ελάχιστη τιμή του ενεργοποιείται το σήμα Triangular το οποίο με τη σειρά του επιτρέπει στις μεταβλητές CMPA και CMPB να διαβάσουν από τους ανάλογους πίνακες το επόμενο προς σύγκριση στοιχείο. Να τονίσουμε πως όπως είδαμε και προηγουμένως στους πίνακες έχουμε αποθηκευμένες τις τιμές των ημιτόνων πολλαπλασιασμένες με το πλάτος των ημιτονοειδών σημάτων και με το κατάλληλο πρόσημο. Συγκεκριμένα ο πίνακας sine_values_A περιέχει θετικές ενώ ο sine_values_B αρνητικές τιμές. Για την απόδοση των τιμών αυτών ελέγχουμε την τιμή του σήματος pos_index, το οποίο μας πληροφορεί για το αν βρισκόμαστε στο 1ο ή στο 2ο μισό της περιόδου των ημιτονοειδών σημάτων, και ανάλογα με αυτή δίνουμε στα CMPA και CMPB τιμές από τους πίνακες sine_values_A ή sine_values_B. Η υλοποίηση αυτή παρουσιάζεται στο παρακάτω τμήμα κώδικα:

```
  if(SAMPLES_READY = '1') then
---------- 0 – T/2 της Περιόδου του Ημιτόνου ----------
    if(pos_index = '1') then
     CMPA<= sine_values_A(index);   -- Λαμβάνει στο διάστημα αυτό θετικές τιμές.
     CMPB<= sine_values_B(index);   -- Λαμβάνει στο διάστημα αυτό αρνητικές τιμές.
     index:= index + 1;
```

```
      if(index = 99) then
        pos_index:= '0';
        index:= 0;
      end if;
-------------------------------------------------------------------------------------------------------------------------------
    else
---------- T/2 - T της Περιόδου του Ημιτόνου ----------------------------------------------------------------------
      CMPA<= sine_values_B(index);   -- Λαμβάνει στο διάστημα αυτό αρνητικές τιμές.
      CMPB<= sine_values_A(index);   -- Λαμβάνει στο διάστημα αυτό θετικές τιμές.
      index:= index + 1;
      if(index = 99) then
        pos_index:= '1';
        index:= 0;
      end if;
-------------------------------------------------------------------------------------------------------------------------------
   end if;
  end if;
```

3) Τέλος υλοποιούμε, όπως και στην παραπάνω σχεδίαση, τα ‘’νεκρά’’ χρονικά διαστήματα τα οποία είναι απαραίτητα για την αποφυγή βραχυκυκλωμάτων. Αυτό γίνεται και πάλι με την χρήση των σημάτων set_S1,set_S3,Delay_eval_S1,Delay_eval_S3. Δεν θα αναλύσουμε καθόλου την υλοποίηση της λειτουργίας αυτής καθώς έχει γίνε αναλυτικά παραπάνω.

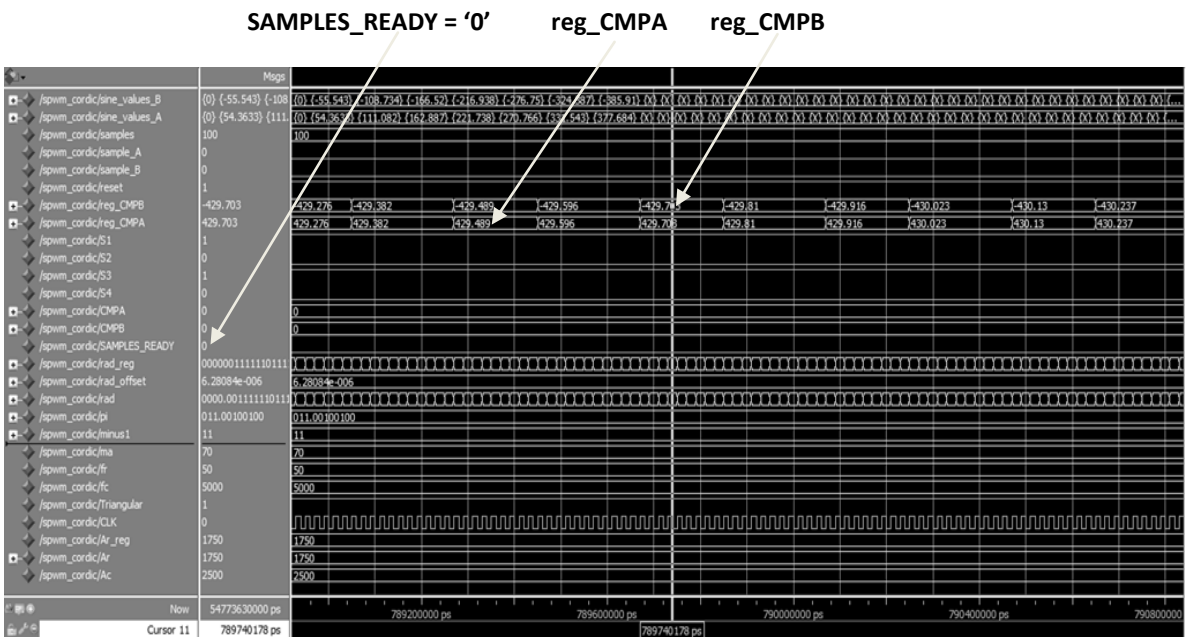
Με βάση λοιπόν τα παραπάνω παρουσιάζεται στο παράρτημα το σύνολο του κώδικα που χρησιμοποιούμε για την υλοποίηση αυτή.

3.5.3 Αποτελέσματα προσομοίωσης σχεδίασης

Στην ενότητα αυτή παρουσιάζουμε ενδεικτικά κάποια στιγμιότυπα από την προσομοίωση της παραπάνω αρχιτεκτονικής, ώστε να επαληθεύσουμε τα θεωρητικώς αναμενόμενα αποτελέσματα. Η προσομοίωση έγινε με το εργαλείο προσομοίωσης Modelsim Altera 10.0c(Quartus II 11.1) Starter

Edition. Τα αποτελέσματα της προσομοίωσης της παραπάνω αρχιτεκτονικής παρουσιάζονται παρακάτω:

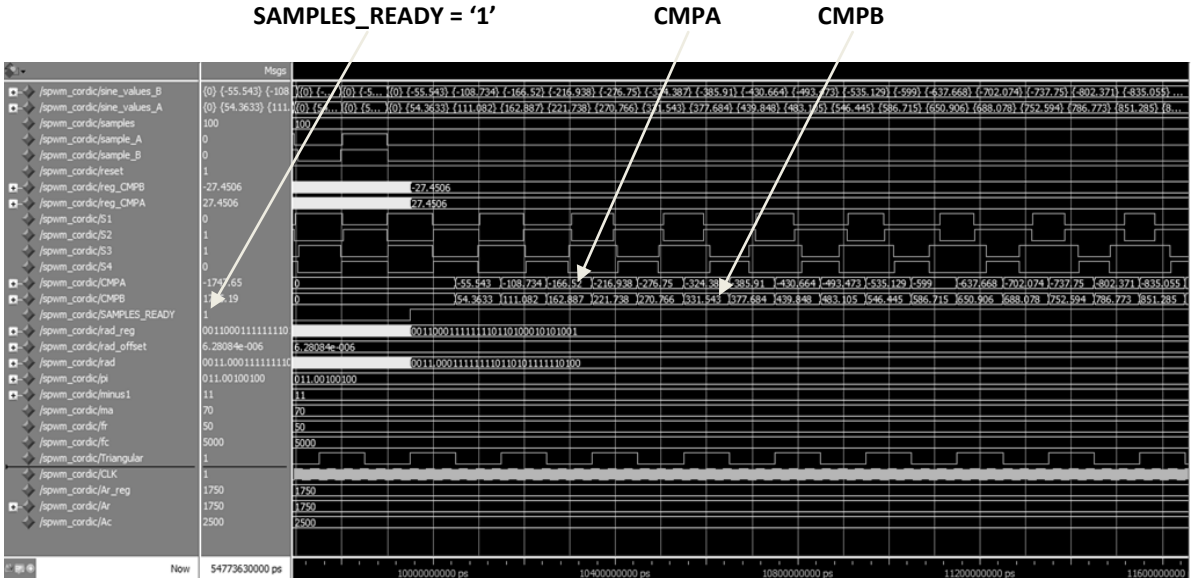
- **1η Περίοδος Λειτουργίας:**



Σχήμα 3.25: 1η Περίοδος Λειτουργίας

Παρατηρούμε πως όπως αναμέναμε σε αυτή την περίοδο λειτουργίας το σήμα SAMPLES_READY = '0' ενώ τα reg_CMPA και reg_CMPB λαμβάνουν διαδοχικές τιμές για να συγκριθούν με το μετρητή.

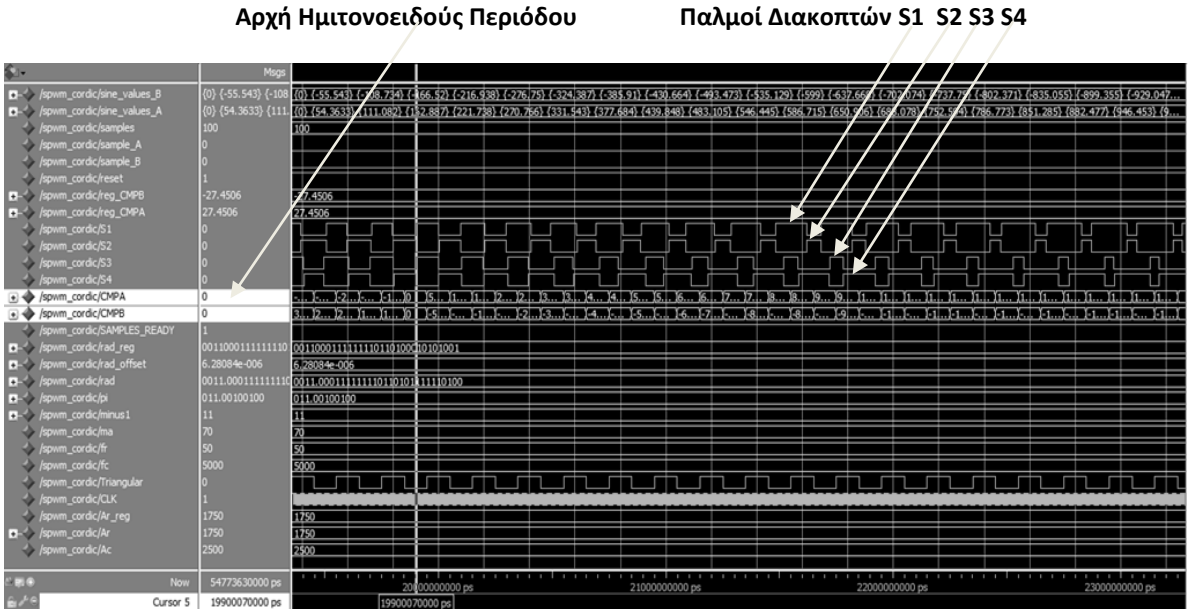
- **2η Περίοδος Λειτουργίας:**



Σχήμα 3.26: 2η Περίοδος λειτουργίας

Παρατηρούμε πως όπως αναμέναμε, σε αυτή την περίοδο λειτουργίας το σήμα SAMPLES_READY γίνεται ‘1’. Επίσης τα σήματα reg_CMPA και reg_CMPB σταματούν να λαμβάνουν τιμές ενώ αντίθετα ξεκινούν να αποκτούν τιμές τα CMPA και CMPB.

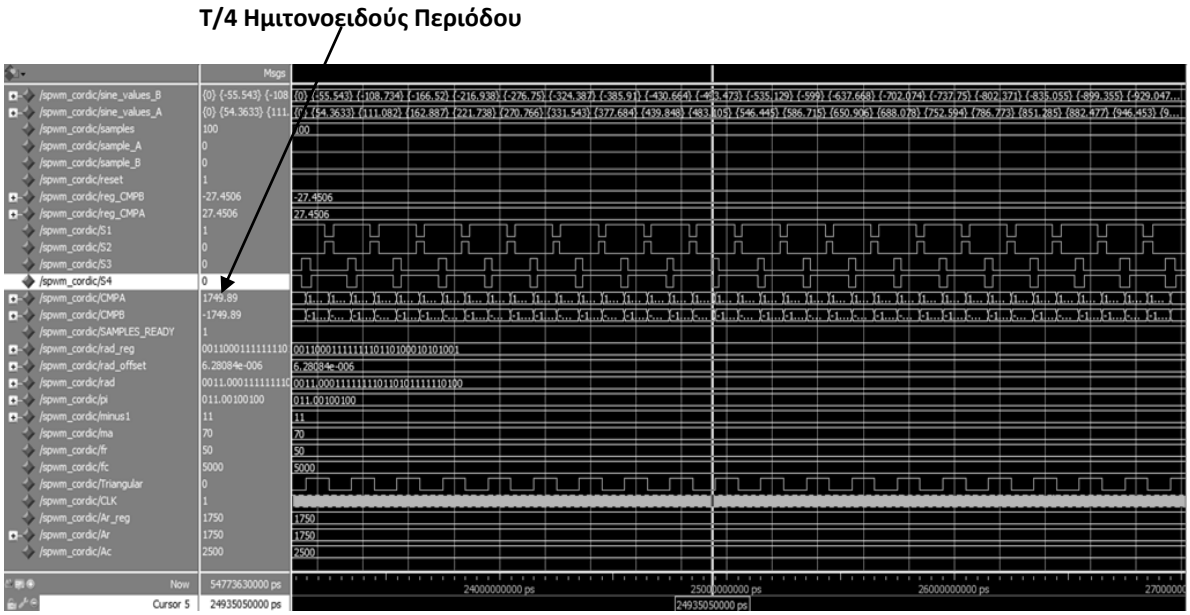
- **Αρχή Ημιτονοειδούς Περιόδου:**



Σχήμα 3.27: Παλμοί διακοπτών στην αρχή της ημιτονοειδούς περιόδου με το εργαλείο προσομοίωσης Modelsim

Από το παραπάνω σχήμα διαπιστώνουμε πως βρισκόμαστε στην έναρξη της ημιτονοειδούς περιόδου καθώς η τιμή που παίρνουν τα CMPA και CMPB στο σημείο που έχουμε τονίσει είναι 0. Επίσης όπως αναμένουμε, στην αρχή της περιόδου του ημιτόνου και καθώς ο χρόνος περνά μέχρι τα μέσα της περιόδου ο θετικός(μηδενικός) παλμός του διακόπτη S1(S2) αυξάνουν σε διάρκεια από τον αντίστοιχο θετικό(μηδενικό) παλμό του διακόπτη S3(S4).

- **T/4 Ημιτονοειδούς Περιόδου:**



Σχήμα 3.28: Παλμοί διακοπτών για το T/4 της ημιτονοειδούς περιόδου με το εργαλείο προσομοίωσης Modelsim

Από το παραπάνω σχήμα διαπιστώνουμε πως βρισκόμαστε στο 1/4 της ημιτονοειδούς περιόδου καθώς η τιμή που παίρνουν τα CMPA και CMPB στο σημείο που έχουμε τονίσει είναι ±1749.89 το οποίο είναι σχεδόν ίσο με το πλάτος του ημιτόνου Ar = 1750. Σε αυτό μάλιστα το διάστημα αναμένουμε, όπως φαίνεται και παραπάνω, να έχουμε τον μεγαλύτερο σε διάρκεια θετικό παλμό για το διακόπτη S1 και το μικρότερο για το διακόπτη S3.

- **T/2 Ημιτονοειδούς Περιόδου:**

Σχήμα 3.29: Παλμοί διακοπτών για το T/2 της ημιτονοειδούς περιόδου με το εργαλείο προσομοίωσης Modelsim

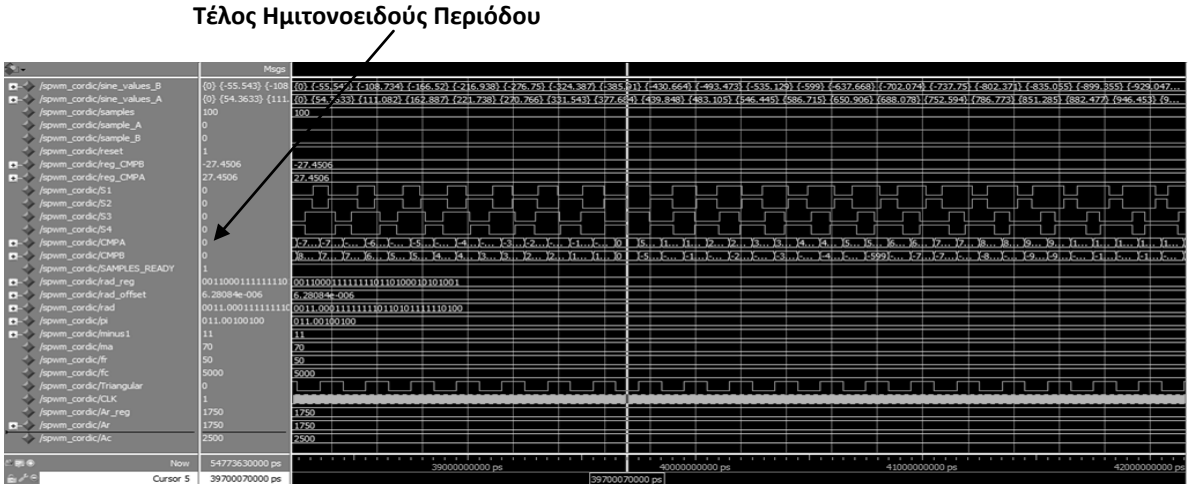
Από το παραπάνω σχήμα διαπιστώνουμε πως βρισκόμαστε στο 1/2 της ημιτονοειδούς περιόδου καθώς η τιμή που παίρνουν τα CMPA και CMPB στο σημείο που έχουμε τονίσει είναι 0 και οι τιμές του CMPA(CMPB) δεξιά του σημείου αυτού είναι θετικές(αρνητικές). Σε αυτό μάλιστα το διάστημα αναμένουμε, όπως φαίνεται και παραπάνω, η διάρκεια των θετικών παλμών των διακοπτών S1 και S3 να περίπου η ίδια.

- **3T/4 Ημιτονοειδούς Περιόδου:**

Σχήμα 3.30: Παλμοί διακοπτών για το 3T/4 της ημιτονοειδούς περιόδου με το εργαλείο προσομοίωσης Modelsim

Από το παραπάνω σχήμα διαπιστώνουμε πως βρισκόμαστε στα 3/4 της ημιτονοειδούς περιόδου καθώς στο σημείο που έχουμε τονίσει το CMPA = -1749.89 και το CMPB = 1749.89, τιμές σχεδόν ίσες με το πλάτος του ημιτόνου Ar = 1750. Σε αυτό το διάστημα αναμένουμε, να έχουμε τον μικρότερο σε διάρκεια θετικό παλμό για το διακόπτη S1 και το μεγαλύτερο για το διακόπτη S3.

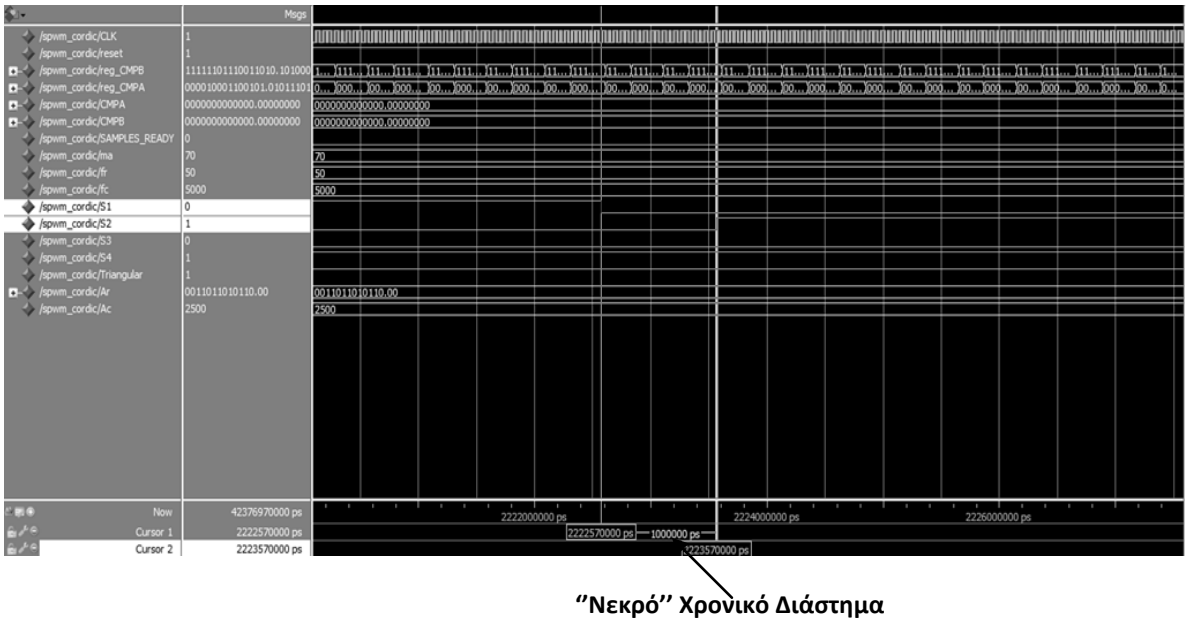
- **Τέλος Ημιτονοειδούς Περιόδου:**



Σχήμα 3.31: Παλμοί διακοπτών για το τέλος της ημιτονοειδούς περιόδου με το εργαλείο προσομοίωσης Modelsim

Από το παραπάνω σχήμα διαπιστώνουμε πως βρισκόμαστε στο τέλος της ημιτονοειδούς περιόδου καθώς η τιμή που παίρνουν τα CMPA και CMPB στο σημείο που έχουμε τονίσει είναι 0 και επίσης οι τιμές του CMPA(CMPB) δεξιά του σημεία αυτού είναι αρνητικές(θετικές). Σε αυτό μάλιστα το διάστημα αναμένουμε, όπως φαίνεται και παραπάνω, η διάρκεια των θετικών παλμών των διακοπτών S1 και S3 να περίπου η ίδια.

- **''Νεκρό'' Χρονικό Διάστημα:**



Σχήμα 3.32: ''Νεκρό'' Χρονικό Διάστημα

Από το παραπάνω σχήμα παρατηρούμε πως από την αποκοπή του διακόπτη S1 έως την αγωγή του διακόπτη S2(συμπληρωματικός του S1) μεσολαβεί το παρακάτω χρονικό διάστημα το οποίο τονίζεται και από το ίδιο το εργαλείο προσομοίωσης:

t_{DEAD} = 1000000ps = 1μs.

Άρα όντως το παραπάνω διάστημα είναι ίσο με το επιθυμητό(1μs).

3.5.4 Αποτελέσματα πειραματικής διαδικασίας

Στο σημείο αυτό, αφού προσομοιώσαμε τον κώδικα που υλοποιεί την συγκεκριμένη SPWM τεχνική είμαστε σε θέση να ''φορτώσουμε'' τον κώδικα αυτό στο FPGA και να παρατηρήσουμε στον παλμογράφο αν όντως τα αποτελέσματα που είδαμε στην προσομοίωση, ισχύουν και στην πράξη. Για το σκοπό αυτό χρησιμοποιήσαμε το FPGA Spartan 6 XC6LX16-CS324 το οποίο μας δίνει την δυνατότητα να χρησιμοποιήσουμε και αριθμούς κινητής υποδιαστολής στη σχεδίασή μας. Για να υλοποιήσουμε τη σχεδίαση στο FPGA ακολουθήσαμε όλα τα βήματα που περιγράψαμε στο 1^{ο} κεφάλαιο και συνδέσαμε μέσω του αρχείου περιορισμού τα σήματα που υπολογίζουν τους παλμούς

των τεσσάρων διακοπτών(S1,S2,S3,S4) με τέσσερεις ακροδέκτες του FPGA. Αυτό το πραγματοποιήσαμε με την παρακάτω δήλωση στο αρχείο περιορισμού:

NET "S1" LOC = "T12" |IOSTANDARD = "LVCMOS33"; -- Σύνδεση σήματος S1

NET "S2" LOC = "V12" |IOSTANDARD = "LVCMOS33"; -- Σύνδεση σήματος S2

NET "S3" LOC = "N10" |IOSTANDARD = "LVCMOS33"; -- Σύνδεση σήματος S3

NET "S4" LOC = "P11" |IOSTANDARD = "LVCMOS33"; -- Σύνδεση σήματος S4

Επίσης έχουμε αναφέρει και παραπάνω πως το ρολόι της σχεδίασής μας τίθεται ίσο με 20ns. Η περίοδος αυτή του ρολογιού, καθορίζεται από το αρχείο περιορισμού μέσω της αποδιδόμενης σε αυτό συχνότητας. Για να πετύχουμε λοιπόν περίοδο ίση με 20ns θα πρέπει η συχνότητα που θα θέσουμε να είναι ίση με:

$f_{clk} = \frac{1}{20\ ns}$ = 50.000 kHz.

Αυτό πραγματοποιείται με την παρακάτω δήλωση:

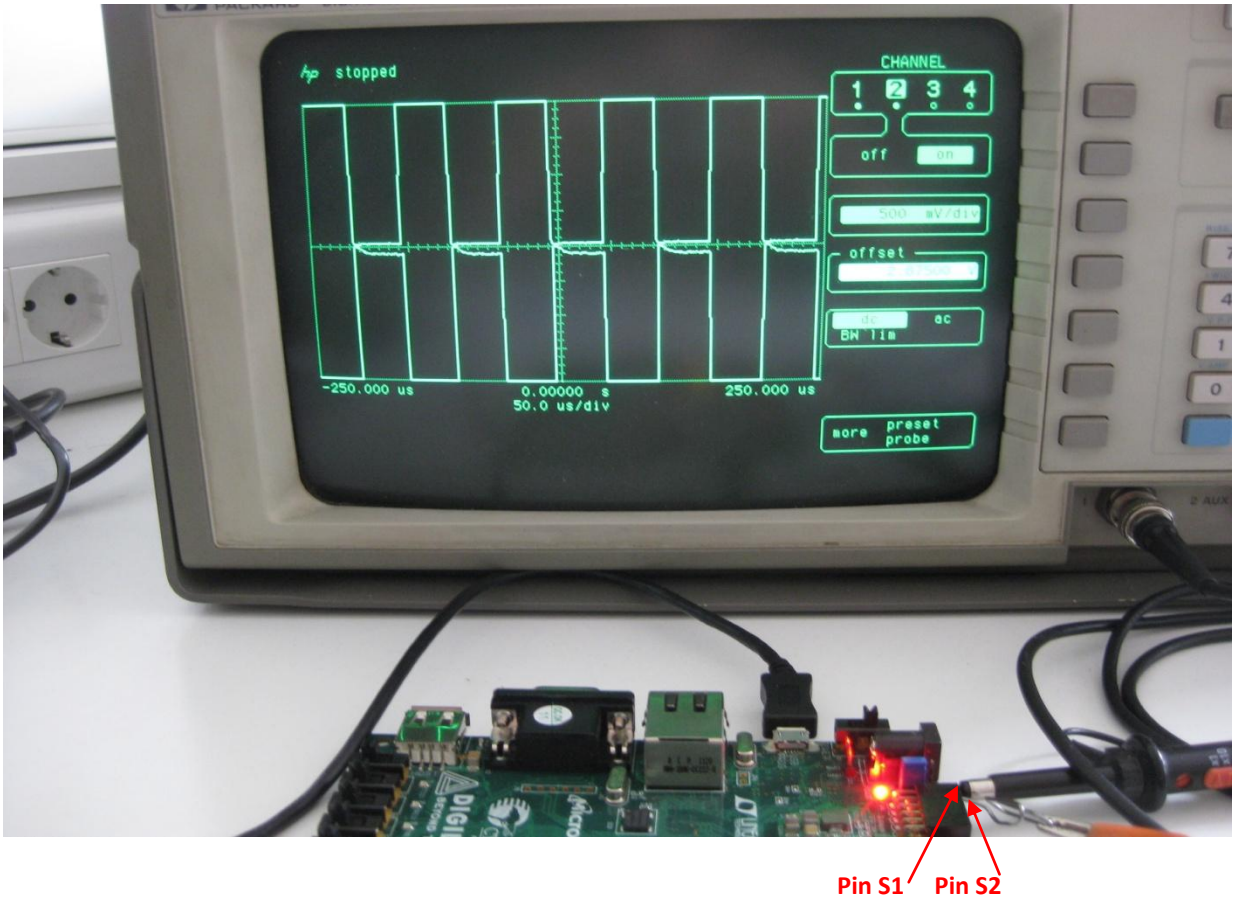
TIMESPEC TS_sys_clk_pin = PERIOD "CLK" 50000 kHz HIGH 50%; -- Ρολόι Περιόδου 20 ns.

Μετά την παραγωγή του αρχείου προγραμματισμού της συσκευής ‘’τρέξαμε’’ την εφαρμογή μας και προέκυψαν στον παλμογράφο τα αποτελέσματα, τα οποία αφορούν στους παλμούς οδήγησης των ημιαγωγικών διακοπτών.

Όπως και στην τεχνική με τη χρήση της μνήμης ROM έτσι και εδώ παρουσιάζουμε δύο στιγμιότυπα τα οποία αναφέρονται στους παλμούς οδήγησης των ζευγών διακοπτών S1-S2 και S3-S4. Τα αποτελέσματα των παλμών αυτών τα οποία παρατηρήθηκαν στον παλμογράφο παρουσιάζονται παρακάτω:

- **Ζεύγος S1-S2:**

Τα αποτελέσματα που προέκυψαν για το ζεύγος αυτό και τα οποία παρατηρήθηκαν με τη βοήθεια του παλμογράφου παρουσιάζονται στο παρακάτω σχήμα:

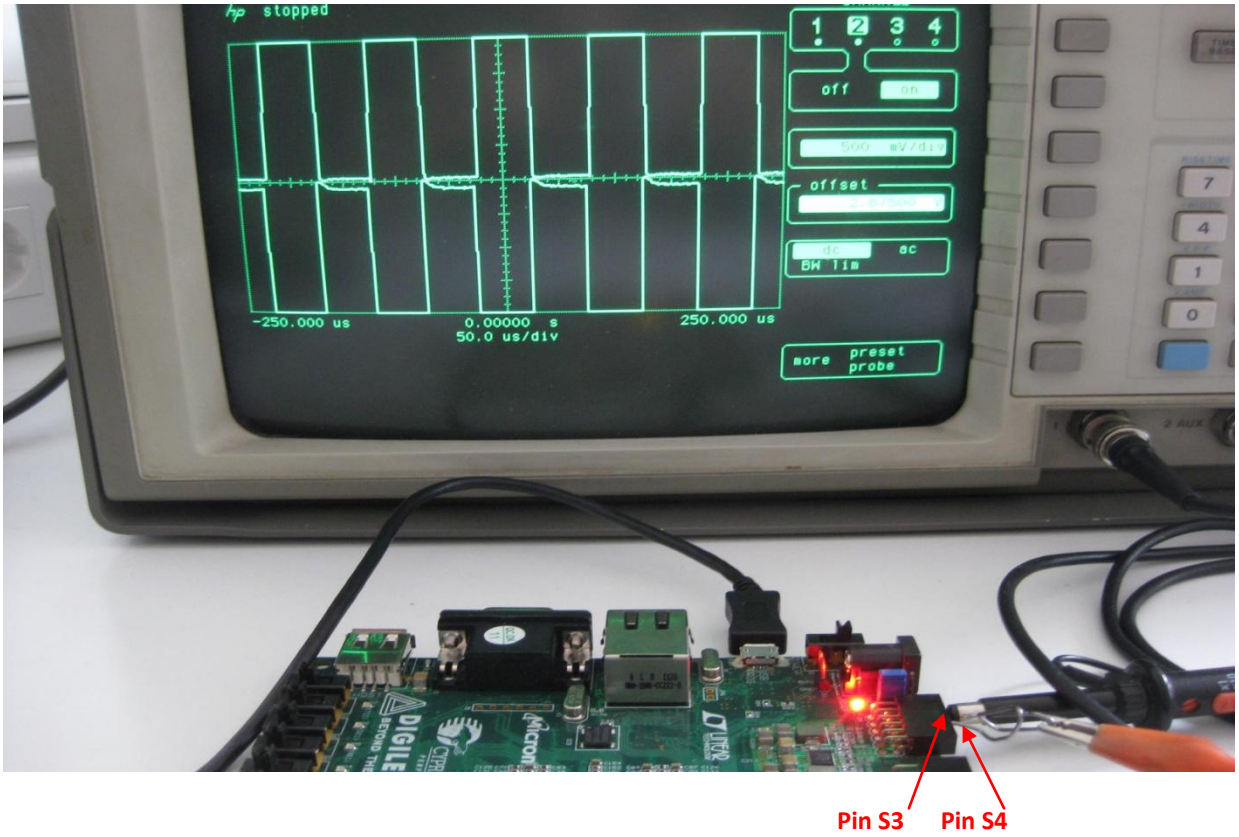


Σχήμα 3.33: Απεικόνιση στον παλμογράφο του συμπληρωματικού ζεύγους παλμών S1-S2 της SPWM τεχνικής με δυναμικό υπολογισμό δειγμάτων

Είναι φανερό με βάση το παραπάνω σχήμα πως και με την χρησιμοποίηση του CORDIC για την υλοποίηση του ημιτονοειδούς σήματος οι παλμοί των διακοπτών του ίδιου σκέλους του αναστροφέα προκύπτουν συμπληρωματικοί.

- **Ζεύγος S3-S4:**

Τέλος, για να ολοκληρώσουμε την παρουσίαση των αποτελεσμάτων, παραθέτουμε παρακάτω και τα αποτελέσματα που παρατηρήσαμε στον παλμογράφο από τις εξόδους του FPGA στις οποίες είχαμε συνδέσει τους παλμούς για την οδήγηση των διακοπτών S3 και S4.



Σχήμα 3.34: Απεικόνιση στον παλμογράφο του συμπληρωματικού ζεύγους παλμών S3-S4 της SPWM τεχνικής με δυναμικό υπολογισμό δειγμάτων

Όπως σε όλες τις παραπάνω απεικονίσεις έτσι και εδώ παρατηρούμε τη συμπληρωματικότητα που παρουσιάζουν οι παλμοί οδήγησης των δύο αυτών διακοπτών το οποίο είναι και το αναμενόμενο.

3.6 Σύγκριση των δύο SWPM τεχνικών

Στο σημείο αυτό, αφού επαληθεύσαμε με τις παραπάνω προσομοιώσεις πως οι δύο SPWM τεχνικές μας δίνουν τα αναμενόμενα αποτελέσματα, θα τις συγκρίνουμε με βάση τους πόρους τους οποίους χρειάζονται για να υλοποιηθούν. Η πληροφορία αυτή παρέχεται μέσω του εργαλείου ISE Design Suite της Xilinx. Παρακάτω λοιπόν παρατίθενται δύο πίνακες που παρουσιάζουν τους χρησιμοποιούμενους από τις δύο σχεδιάσεις πόρους:

1) SPWM Τεχνική με χρήση Μνήμης ROM:

Device Utilization Summary			
Slice Logic Utilization	Used	Available	Utilization
Number of Slice Registers	98	54,576	1%
Number of Slice LUTs	1,997	27,288	7%
Number used as logic	1,971	27,288	7%
Number used as Memory	0	6,408	0%

Number of occupied Slices	641	6,822	9%
Number of MUXCYs used	1,264	13,644	9%
Number of LUT Flip Flop pairs used	1,999		
Number of bonded IOBs	6	218	2%
Number of RAMB16BWERs	0	116	0%
Number of RAMB8BWERs	0	232	0%
Number of BUFIO2/BUFIO2_2CLKs	0	32	0%
Number of BUFIO2FB/BUFIO2FB_2CLKs	0	32	0%
Number of BUFG/BUFGMUXs	1	16	6%
Number of DCM/DCM_CLKGENs	0	8	0%
Number of ILOGIC2/ISERDES2s	0	376	0%
Number of IODELAY2/IODRP2/IODRP2_MCBs	0	376	0%
Number of OLOGIC2/OSERDES2s	0	376	0%
Number of BSCANs	0	4	0%
Number of BUFHs	0	256	0%
Number of BUFPLLs	0	8	0%
Number of BUFPLL_MCBs	0	4	0%
Number of DSP48A1s	3	58	5%
Number of ICAPs	0	1	0%
Number of MCBs	0	2	0%
Number of PCILOGICSEs	0	2	0%
Number of PLL_ADVs	0	4	0%
Number of PMVs	0	1	0%
Number of STARTUPs	0	1	0%
Number of SUSPEND_SYNCs	0	1	0%
Average Fanout of Non-Clock Nets	3.88		

Πίνακας 3.1: Χρησιμοποιούμενοι πόροι της συσκευής για την υλοποίηση της SPWM τεχνικής με χρήση μνήμης ROM

2) SPWM Τεχνική με Δυναμικό Υπολογισμό Δειγμάτων:

Device Utilization Summary			
Slice Logic Utilization	Used	Available	Utilization
Number of Slice Registers	1,163	54,576	2%
Number of Slice LUTs	3,262	27,288	11%
Number used as logic	3,218	27,288	11%
Number used as Memory	5	6,408	1%
Number of occupied Slices	955	6,822	13%
Number of MUXCYs used	2,296	13,644	16%
Number of LUT Flip Flop pairs used	3,311		
Number of bonded IOBs	6	218	2%
Number of RAMB16BWERs	0	116	0%
Number of RAMB8BWERs	0	232	0%
Number of BUFIO2/BUFIO2_2CLKs	0	32	0%
Number of BUFIO2FB/BUFIO2FB_2CLKs	0	32	0%
Number of BUFG/BUFGMUXs	1	16	6%
Number of DCM/DCM_CLKGENs	0	8	0%
Number of ILOGIC2/ISERDES2s	0	376	0%
Number of IODELAY2/IODRP2/IODRP2_MCBs	0	376	0%
Number of OLOGIC2/OSERDES2s	0	376	0%
Number of BSCANs	0	4	0%
Number of BUFHs	0	256	0%
Number of BUFPLLs	0	8	0%
Number of BUFPLL_MCBs	0	4	0%
Number of DSP48A1s	3	58	5%

Number of ICAPs	0	1	0%
Number of MCBs	0	2	0%
Number of PCILOGICSEs	0	2	0%
Number of PLL_ADVs	0	4	0%
Number of PMVs	0	1	0%
Number of STARTUPs	0	1	0%
Number of SUSPEND_SYNCs	0	1	0%
Average Fanout of Non-Clock Nets	3.29		

Πίνακας 3.2: Χρησιμοποιούμενοι πόροι της συσκευής για την υλοποίηση της SPWM τεχνικής με δυναμικό υπολογισμό δειγμάτων

Από τους παραπάνω πίνακες, γίνεται εύκολα αντιληπτό πως όπως αναμέναμε η δεύτερη υλοποίηση της SPWM τεχνικής η οποία αφορά στο δυναμικό υπολογισμό δειγμάτων απαιτεί τη χρησιμοποίηση μεγαλύτερου αριθμού πόρων του FPGA από αυτή που χρησιμοποιεί τη μνήμη ROM. Αυτό είναι αναμενόμενο καθώς στο πρόγραμμα με το δυναμικό υπολογισμό των δειγμάτων υπολογίζουμε με τη χρήση του αλγορίθμου CORDIC τις νέες τιμές που χρειάζονται ενώ σε αυτό με τη χρήση της ROM χρησιμοποιούμε ήδη υπολογισμένες τιμές. Βεβαίως παρότι χρειαζόμαστε περισσότερους πόρους, λαμβάνουμε πιο ακριβή αποτελέσματα. Καλούμαστε λοιπόν να επιλέξουμε μία εκ' των δύο τεχνικών ανάλογα με το αν θα θέσουμε ως πρώτη προτεραιότητα την ακρίβεια των αποτελεσμάτων ή τους χρησιμοποιούμενους πόρους.

ΚΕΦΑΛΑΙΟ 4^{ο}

ΥΛΟΠΟΙΗΣΗ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ ΤΗΣ ΠΑΡΑΜΕΤΡΙΚΗΣ, ΠΟΛΥΦΑΣΙΚΗΣ ΚΑΙ ΠΟΛΥΕΠΙΠΕΔΗΣ SVPWM ΤΕΧΝΙΚΗΣ

4.1 Εισαγωγή

Τα τελευταία χρόνια, το ενδιαφέρον για την τεχνολογία των πολυφασικών μετατροπέων έχει αυξηθεί λόγω των πλεονεκτημάτων που προσφέρει η χρήση περισσοτέρων των τριών φάσεων σε εφαρμογές κίνησης. Εκτός αυτού, η τεχνολογία πολυεπίπεδων μετατροπέων επιτρέπει την επίτευξη υψηλών τιμών ισχύος με την χρησιμοποίηση συσκευών περιορισμένης τάσης. Η πολυφασική, πολυεπίπεδη τεχνολογία συνδυάζει τα πλεονεκτήματα και των δύο τεχνολογιών.

Οι περισσότεροι από τους ηλεκτρικούς κινητήρες μεταβλητής ταχύτητας χρησιμοποιούν τριφασικές μηχανές. Παρ' όλα αυτά, δεδομένου ότι οι μεταβλητής ταχύτητας εναλλασσόμενοι κινητήρες περιλαμβάνουν έναν ηλεκτρονικό μετατροπέα ισχύος, ο αριθμός των φάσεων της μηχανής μπορεί να είναι μεγαλύτερος των τριών. Τα κύρια πλεονεκτήματα της χρήσης μία πολυφασικής μηχανής έναντι μίας τριφασικής είναι:

- Αυξημένη αξιοπιστία και αντοχή σε σφάλματα.
- Μεγαλύτερη αποδοτικότητα.
- Υψηλότερη πυκνότητα και μειωμένες ταλαντώσεις ροπής.
- Χαμηλότερες ανά φάση απαιτήσεις χειρισμού ισχύος.
- Βελτιωμένη προσαρμοστικότητα.
- Βελτιωμένα χαρακτηριστικά θορύβου.

Ορισμένες πρόσφατες εφαρμογές πολυφασικών συστημάτων περιλαμβάνουν χαμηλής ταχύτητας μηχανές, υψηλής ροπής και χωρίς ψήκτρες οι οποίες εφαρμόζονται σε ηλεκτρικά οχήματα πρόωσης, κινητήρες μονίμων μαγνητών για πρόωση πλοίων, κινητήρες μονίμων μαγνητών με χαμηλή ταλάντωση ροπής και στην σειριακή ένωση δύο κινητήρων με την υποστήριξη ενός μόνο αναστροφέα.

Η τεχνολογία πολυεπίπεδων μετατροπέων βασίζεται στην σύνθεση μίας κυματομορφής τάσης αποτελούμενη από διάφορα επίπεδα DC τάσης. Όσο αυξάνει ο αριθμός των επιπέδων, η συνθέσιμη τάση εξόδου αποκτά όλο και περισσότερα ''σκαλοπάτια'' και παράγεται κατ' αυτόν τον τρόπο μία κυματομορφή η οποία προσεγγίζει πληρέστερα την ημιτονοειδή μορφή που είναι και η επιθυμητή. Τα κύρια πλεονεκτήματα της χρήσης πολυεπίπεδων αναστροφέων είναι:

- Δυνατότητα παραγωγής υψηλής τάσεως από συσκευές περιορισμένης τάσης.
- Χαμηλή αρμονική παραμόρφωση.
- Μειωμένες απώλειες μεταγωγής.
- Αυξημένη αποδοτικότητα.
- Καλή ηλεκτρομαγνητική συμβατότητα.

Οι πολυεπίπεδοι μετατροπείς έχουν μελετηθεί ευρέως σε μεγάλη ποικιλία εφαρμογών. Πρόσφατες βιομηχανικές εφαρμογές πολυεπίπεδων αναστροφέων περιλαμβάνουν κινητήρες μηχανών επαγωγής, ενεργές ανορθώσεις, διασύνδεση ανανεώσιμων πηγών ενέργειας στο ηλεκτρικό δίκτυο και στατικούς, σύγχρονούς αντισταθμιστές. Πρόσφατα, έλαβε χώρα μία πρώτη προσπάθεια ενσωμάτωσης ενός πολυεπίπεδου αναστροφέα σε μία πολυφασική μηχανή κάτι που κατέδειξε τα πλεονεκτήματα του συνδυασμού των δύο τεχνολογιών.

Η SVPWM τεχνική προσφέρει σημαντικά οφέλη όσον αφορά στην απόδοση και έχει αποδειχθεί ιδιαίτερα δημοφιλής στα τριφασικά συστήματα. Στην παρούσα εργασία, παρουσιάζεται ένας γενικός αλγόριθμος για την υλοποίηση της SVPWM τεχνικής. Αυτός ο αλγόριθμος είναι το αποτέλεσμα δύο κύριων παρατηρήσεων. Την παρατήρηση πως ένας πολυεπίπεδος, πολυφασικός διαμορφωτής μπορεί να υλοποιηθεί από ένα πολυφασικό διαμορφωτή δύο επιπέδων και από την ανάπτυξη ενός νέου πολυφασικού SVPWM αλγορίθμου δύο επιπέδων. Η τεχνική που παρουσιάζεται για την υλοποίηση ενός πολυεπίπεδου διαμορφωτή με τη χρήση ενός διαμορφωτή δύο επιπέδων μπορεί να εφαρμοστεί σε οποιονδήποτε αριθμό φάσεων και επιπέδων. Επίσης παρουσιάζει πολύ χαμηλό υπολογιστικό κόστος, δεν χρησιμοποιεί τριγωνομετρικές συναρτήσεις, το οποίο είναι ανεξάρτητο από τον αριθμό των επιπέδων και είναι ταιριαστό για πραγματικού χρόνου υλοποιήσεις στο υλικό. Τέλος η νέα αυτή τεχνική διατυπώνεται στο μη μετασχηματισμένο πολυδιάστατο χώρο για ένα γενικό αριθμό φάσεων.

4.2 Η τεχνική SVPWM

Τα τελευταία χρόνια, η τεχνική διαμόρφωσης εύρους παλμών με χωρικά διανύσματα τάσης(SVPWM) βρίσκει μεγάλη εφαρμογή στους ηλεκτρονικούς μετατροπείς ισχύος. Αποτελεί μία προηγμένη τεχνική, η οποία απαιτεί μεγάλο υπολογιστικό φόρτο, υλοποιείται όμως σχετικά εύκολα με ψηφιακές διατάξεις ελέγχου, όπως είναι οι ψηφιακοί επεξεργαστές σήματος(Digital Signal Processor, DSP) ή οι μικροϋπολογιστές(microprocessors,μPs).

Η τεχνική SPWM, με την οποία ασχοληθήκαμε παραπάνω, πραγματοποιείται για κάθε φάση χωριστά. Αυτή λειτουργεί κανονικά όσο οι τρεις φάσεις του αναστροφέα λειτουργούν ανεξάρτητα μεταξύ τους. Όταν όμως αυτό δεν συμβαίνει, όπως για παράδειγμα σε ένα τριφασικό κινητήρα, και οι τρεις φάσεις δεν είναι πλέον ανεξάρτητες μεταξύ τους αλλά το ρεύμα της κάθε μιας εξαρτάται από αυτό των άλλων δύο τότε η τεχνική SPWM δεν δύναται να λάβει υπόψη την εξάρτηση αυτή. Αντιθέτως η τεχνική SVPWM λαμβάνει υπόψη αυτή την εξάρτηση, αφού η λειτουργία της βασίζεται στις πολικές και όχι στις φασικές τάσεις του φορτίου. Η μέθοδος SVPWM χρησιμοποιεί τα διανύσματα κατάστασης του αναστροφέα καθώς και το διάνυσμα αναφοράς της τάσης ώστε να διαμορφώσει το εύρος των παλμών που θα οδηγήσουν τους ημιαγωγικούς διακόπτες του αναστροφέα.

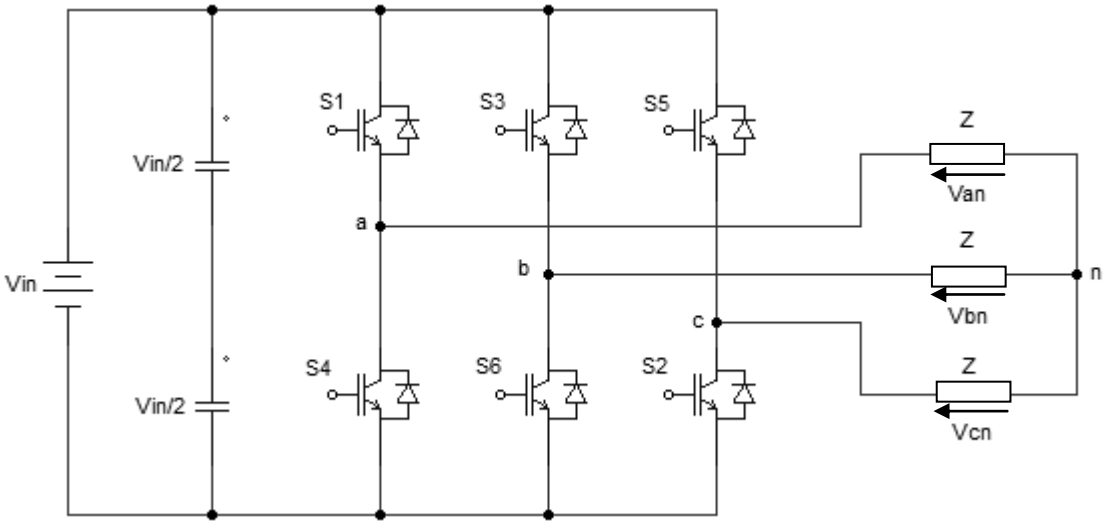
Η αρχή στην οποία στηρίζεται η ανάπτυξη της τεχνικής SVPWM είναι η δυνατότητα που υπάρχει να εκφραστούν οι τάσεις εξόδου ενός τριφασικού αναστροφέα ως σταθερά χωρικά διανύσματα στο επίπεδο d-q. Εναλλακτικά, τα διανύσματα τάσης ονομάζονται και χωρικά διανύσματα τάσης(voltage space vectors), όμως δεν πρόκειται για διανύσματα στο χώρο αλλά στο μιγαδικό επίπεδο, αφού τα τριφασικά συστήματα που εξετάζουμε είναι συμμετρικά και δεν εμφανίζουν συνιστώσα μηδενικής ακολουθίας(zero sequence component). Η μέθοδος SVPWM χρησιμοποιείται σε εφαρμογές οδήγησης

και ελέγχου επαγωγικών κινητήρων και κινητήρων μονίμων μαγνητών. Τα πλεονεκτήματα της τεχνικής SVPWM είναι τα ακόλουθα:

i. Μικρότερη αρμονική παραμόρφωση τάσης και ρεύματος εξόδου.
ii. Αποδοτικότερη χρήση της τάσης τροφοδοσίας εισόδου σε σχέση με την τεχνική SPWM.

Περιληπτικά οι αρχές στις οποίες στηρίζεται η ανάπτυξη της τεχνικής SVPWM είναι οι ακόλουθες:

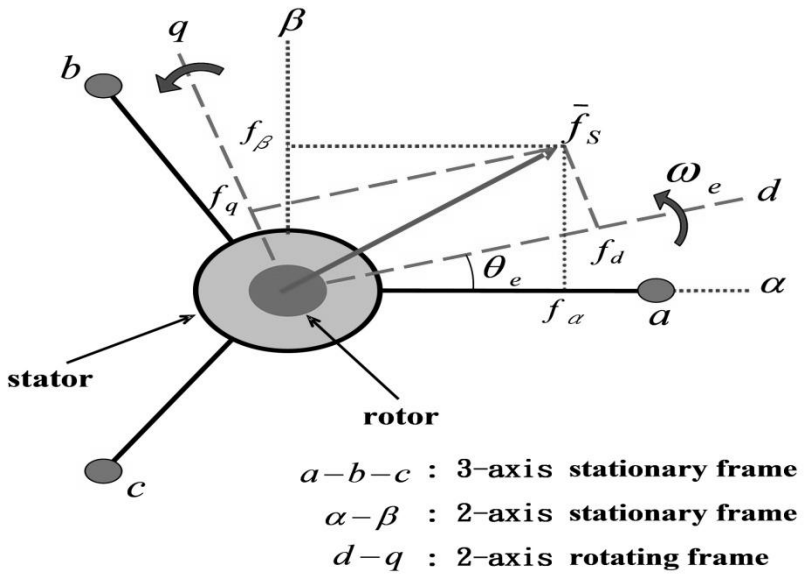
- Χειρίζεται την ημιτονοειδή τάση ως ένα στρεφόμενο χωρικό διάνυσμα σταθερού πλάτους και σταθερής συχνότητας.
- Η τεχνική SVPWM υπολογίζει προσεγγιστικά την τάση αναφοράς V_{ref} μέσω συνδυασμών των οκτώ διανυσμάτων που προκύπτουν από τα οκτώ διακοπτικά διανύσματα ($V_0 – V_7$) τα οποία αφορούν σε όλους τους δυνατούς συνδυασμούς αγωγής και αποκοπής των διακοπτών του παρακάτω τριφασικού αναστροφέα:



Σχήμα 4.1: Τριφασικός αναστροφέας με πηγή τάσης και IGBTs ημιγωγικούς διακόπτες

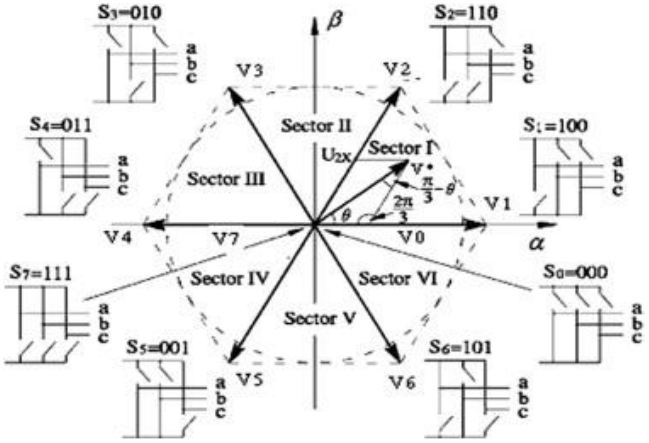
Να σημειωθεί πως οι δυνατοί διακοπτικοί συνδυασμοί είναι οχτώ καθώς μελετάμε μόνο τις δυνατές καταστάσεις των διακοπτών του πάνω τμήματος του αναστροφέα. Αυτό καθώς οι αντίστοιχοι διακόπτες του κάτω τμήματος είναι πάντα συμπληρωματικοί αυτών.

- Πραγματοποιείται μετασχηματισμός συντεταγμένων από το τριφασικό στρεφόμενο πεδίο αναφοράς abc στο διφασικό στατό d-q πεδίο. Ένα τριφασικό διάνυσμα τάσης μετασχηματίζεται σε ένα στατό διάνυσμα d-q συντεταγμένων το οποίο αντιπροσωπεύει το χωρικό διάνυσμα του αθροίσματος των τριών φασικών τάσεων. Ο μετασχηματισμός αυτός γίνεται πιο κατανοητός με το παρακάτω σχήμα που αφορά σε τριφασική μηχανή:



Σχήμα 4.2: Μετασχηματισμός από το abc στο dq σύστημα συντεταγμένων.

- Τα διανύσματα V_1 – V_6 διαιρούν το πεδίο σε έξι τομείς(κάθε τομέας καλύπτει 60°). Οι τομείς αυτοί παρουσιάζονται και στο παρακάτω σχήμα:



Σχήμα 4.3: Χωρικά διανύσματα φασικών τάσεων φορτίου του αναστροφέα δύο επιπέδων στο επίπεδο d-q.

- Η τάση αναφοράς V_{ref} παράγεται από δύο γειτονικά του ενεργά διανύσματα και από τα δύο μηδενικά διανύσματα(V_0 = [000] και V_7 = [111]).

4.3 Υλοποίηση αλγορίθμου

Στους πολυφασικούς μετατροπείς, η SVPWM τεχνική αποτελεί ένα πολυδιάστατο πρόβλημα στο οποίο η επιλογή του διανύσματος μπορεί απευθείας να γίνει σε ένα πολυδιάστατο χώρο. Για παράδειγμα το πρόβλημα διαμόρφωσης ενός P φάσεων μετατροπέα διαμορφώνεται σε έναν P διαστάσεων χώρο, και επιλύεται για πολυεπίπεδες τοπολογίες στις οποίες το επίπεδο εξόδου της κάθε φάσης είναι ένα

ακέραιο πολλαπλάσιο ενός σταθερού βήματος τάσης V_{DC}. Τέτοιες τοπολογίες μπορεί να είναι οι πολυεπίπεδοι αναστροφείς αιωρούμενων πυκνωτών(flying capacitors) και διόδων περιορισμού(diode clamped), αλλά και οι αναστροφείς πολλαπλών επιπέδων που αποτελούνται από μονοφασικούς αναστροφείς συνδεδεμένους σε σειρά με ανεξάρτητες πηγές τάσης όπως επίσης και οι υβριδικοί μετατροπείς. Αφού οι καταστάσεις των διακοπτών οποιασδήποτε τοπολογίας μετατροπέα ισχύος μένουν σε διακριτές καταστάσεις, η SVPWM τεχνική χρησιμοποιείται για την σύνθεση ενός διανύσματος τάσης αναφοράς V_r μέσω μίας ακολουθίας χωρικών διανυσμάτων, κατά τη διάρκεια κάθε κύκλου διαμόρφωσης. Κάθε χωρικό διάνυσμα V_{sj} πρέπει να εφαρμοστεί κατά τη διάρκεια ενός χρονικού διαστήματος t_j με βάση τον παρακάτω νόμο διαμόρφωσης:

$$V_r = \frac{1}{T} \sum_{j=1}^{P+1} V_{sj} * T_j$$

όπου το άθροισμα των χρονικών διαστημάτων T_j θα πρέπει να είναι ίσο με την περίοδο διαμόρφωσης T όπως φαίνεται και παρακάτω:

$$\sum_{j=1}^{P+1} T_j = T$$

Το διάνυσμα αναφοράς συνοψίζει την τάση αναφοράς για την κάθε φάση του συστήματος, ενώ κάθε διάνυσμα μεταγωγής συνοψίζει την διακοπτική κατάσταση της κάθε φάσης του μετατροπέα.

$$V_r = [V_r^1, V_r^2, \dots, V_r^P]^T \in R^P$$

$$V_{sj} = [V_{sj}^1, V_{sj}^2, \dots, V_{sj}^P]^T \in R^P$$

Συνεπώς τα διανύσματα αναφοράς και μεταγωγής ανήκουν στον πολυδιάστατο χώρο R^P όπου P είναι ο αριθμός των φάσεων του μετατροπέα.

Στις περισσότερες πολυεπίπεδες τοπολογίες το επίπεδο της τάσης εξόδου της κάθε φάσης V_s είναι ένα ακέραιο πολλαπλάσιο ενός σταθερού ‘’βήματος’’ τάσης V_{dc}.

$$V_s = n * V_{dc}$$

Συνεπώς, τα διανύσματα και οι χρόνοι μεταγωγής μπορούν να κανονικοποιηθούν με την χρήση του σταθερού βήματος τάσης V_{dc}, και της περιόδου μεταγωγής T αντίστοιχα όπως παρουσιάζεται παρακάτω:

$$v_r = \frac{V_r}{V_{dc}} \in R^P$$

$$v_{sj} = \frac{V_{sj}}{V_{dc}} \in Z^P$$

$$t_j = \frac{T_j}{T}$$

Με βάση λοιπόν τα παραπάνω ο νόμος διαμόρφωσης αλλά και τα νέα διανύσματα αναφοράς και μεταγωγής μπορούν να γραφούν όπως παρακάτω:

$v_r = \sum_{j=1}^{P+1} v_{sj} * t_j$ όπου $\sum_{j=1}^{P+1} t_j = 1$

$v_r = [u_r^1, u_r^2, \dots, u_r^P]^T$

$v_{sj} = [u_{sj}^1, u_{sj}^2, \dots, u_{sj}^P]^T$

Αν αυτά τα φέρουμε σε μορφή πινάκων τότε το σύστημα που καλούμαστε να επιλύσουμε είναι το παρακάτω:

$$\begin{bmatrix} 1 \\ u_r^1 \\ u_r^2 \\ . \\ . \\ . \\ u_r^P \end{bmatrix} = \begin{bmatrix} 1 & 1 & . & . & . & 1 \\ u_{s1}^1 & u_{s2}^1 & . & . & . & u_{sP+1}^1 \\ u_{s1}^2 & u_{s2}^2 & . & . & . & u_{sP+1}^2 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ u_{s1}^P & u_{s2}^P & . & . & . & u_{sP+1}^P \end{bmatrix} * \begin{bmatrix} t_1 \\ t_2 \\ . \\ . \\ . \\ t_{P+1} \end{bmatrix}.$$

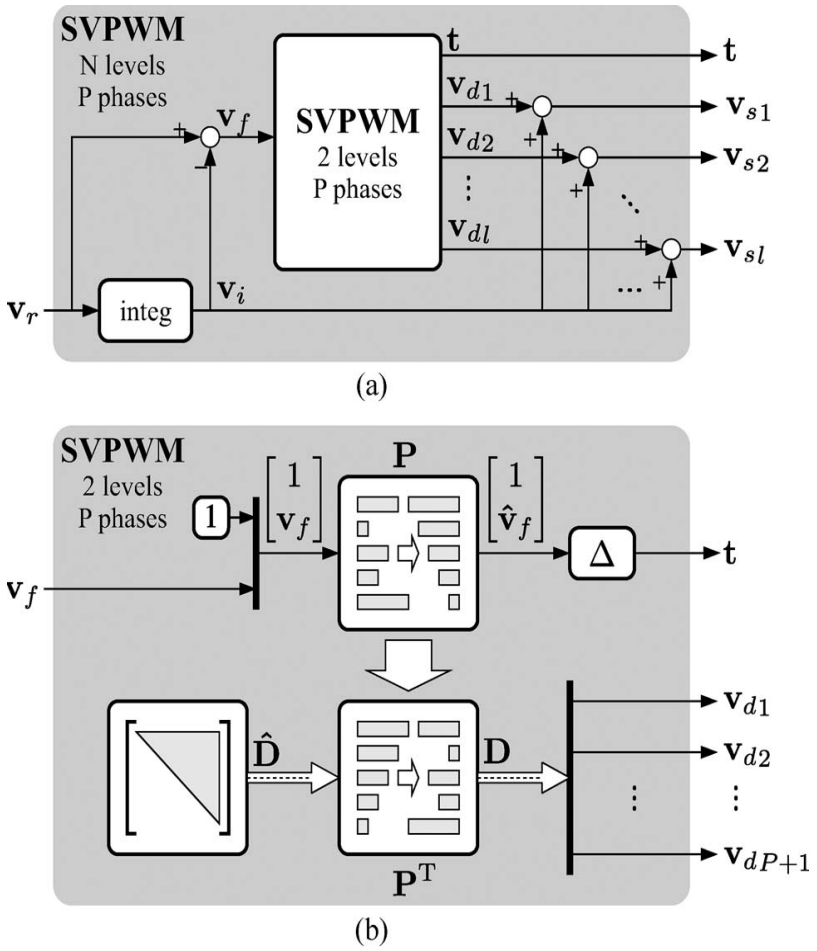
Το παραπάνω σύστημα γραμμικών εξισώσεων συνιστά το νόμο διαμόρφωσης και πρέπει να επιλυθεί από τον πολυεπίπεδο, πολυφασικό SVPWM αλγόριθμο. Η λύση του προβλήματος περιλαμβάνει τρία βασικά βήματα:

1) Αναζήτηση μίας σειράς ακεραίων συντελεστών για την παραπάνω μήτρα η οποία θα επιλύει το γραμμικό σύστημα.

2) Επίλυση του συστήματος των γραμμικών εξισώσεων για τον υπολογισμό των χρόνων μεταγωγής.

3) Εξαγωγή της ακολουθίας των διανυσμάτων μεταγωγής από τον πίνακα.

Το προαναφερθέν πρόβλημα διαμόρφωσης επιλύεται με τη βοήθεια ενός μαθηματικού αλγορίθμου του οποίου το μπλοκ διάγραμμα παρουσιάζεται παρακάτω:



Σχήμα 4.4: Μπλοκ διάγραμμα της πολυφασικής πολυεπίπεδης SVPWM. (a) Πολυφασική πολυεπίπεδη SVPWM βασισμένη σε μία δύο επιπέδων SVPWM. (b) Μπλοκ διάγραμμα της δύο επιπέδων πολυφασικής SVPWM.

4.3.1 Παραμετρική σχεδίαση, ανεξάρτητη τεχνολογίας

Προκειμένου να δημιουργήσουμε ένα γενικό παραμετρικό κύκλωμα το οποίο θα είναι και ανεξάρτητο της τεχνολογίας υλοποίησης ακολουθούμε τα δύο παρακάτω βήματα:

1) Όλα τα συστατικά στοιχεία από τα οποία αποτελείται η σχεδίασή μας γίνονται παραμετρικά με την χρήση ενός VHDL πακέτου αλλά και της εντολής GENERIC(γενική παράμετρος) η οποία μας επιτρέπει τον ορισμό γενικών παραμέτρων οι οποίες εύκολα τροποποιούνται και χρησιμοποιούνται για διαφορετικές εφαρμογές. Ο χρήστης το μόνο που πρέπει να κάνει είναι να τροποποιήσει κατάλληλα, όποτε επιθυμεί, το πακέτο αυτό εισάγοντας τις επιθυμητές τιμές και να ξανακάνει σύνθεση την σχεδίασή του. Με αυτό τον τρόπο λαμβάνει ένα νέο παραμετρικό κύκλωμα.

Όλα τα συστατικά στοιχεία που χρησιμοποιούμε στην σχεδίαση είναι παραμετρικά λόγω της χρήσης του παρακάτω πακέτου:

```
---------- Δήλωση απαραίτητων Βιβλιοθηκών και Πακέτων ----------

library ieee;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;


library ieee_proposed;

use ieee_proposed.fixed_float_types.all;

use ieee_proposed.fixed_pkg.all;

use ieee_proposed.float_pkg.all;


library std;

use std.standard.all;

------------------------------------------------------------------------------------------------------------------------

---------- Δηλώσεις Τύπων Δεδομένων Πακέτου ----------

Package svpwm_constants is

   CONSTANT N: INTEGER RANGE 0 TO 127:= 5;   -- Δήλωση Αριθμού Επιπέδων του Μετατροπέα.

   CONSTANT P: INTEGER RANGE 0 TO 127:= 5;   -- Δήλωση Αριθμού Φάσεων του Μετατροπέα.

   CONSTANT Q: INTEGER RANGE 0 TO 127:= 9;   -- Bit Κλασματικού Μέρους Αναφοράς.

   CONSTANT I: INTEGER RANGE 0 TO 127:= 3;     -- Bit Ακεραίου Μέρους Αναφοράς.

     ---------- Δηλώσεις Πινάκων που χρησιμοποιούνται από τα Συστατικά Στοιχεία ----------

    TYPE vector_Vref is ARRAY(1 TO P) of sfixed(I-1 downto -Q);

    TYPE vector_Vf is ARRAY(1 TO P) of sfixed(I downto -Q);

    TYPE array_P is ARRAY(0 TO P) of INTEGER RANGE 0 TO P;

   TYPE array_Vi is ARRAY(1 TO P) of INTEGER RANGE -2 TO 2;
```

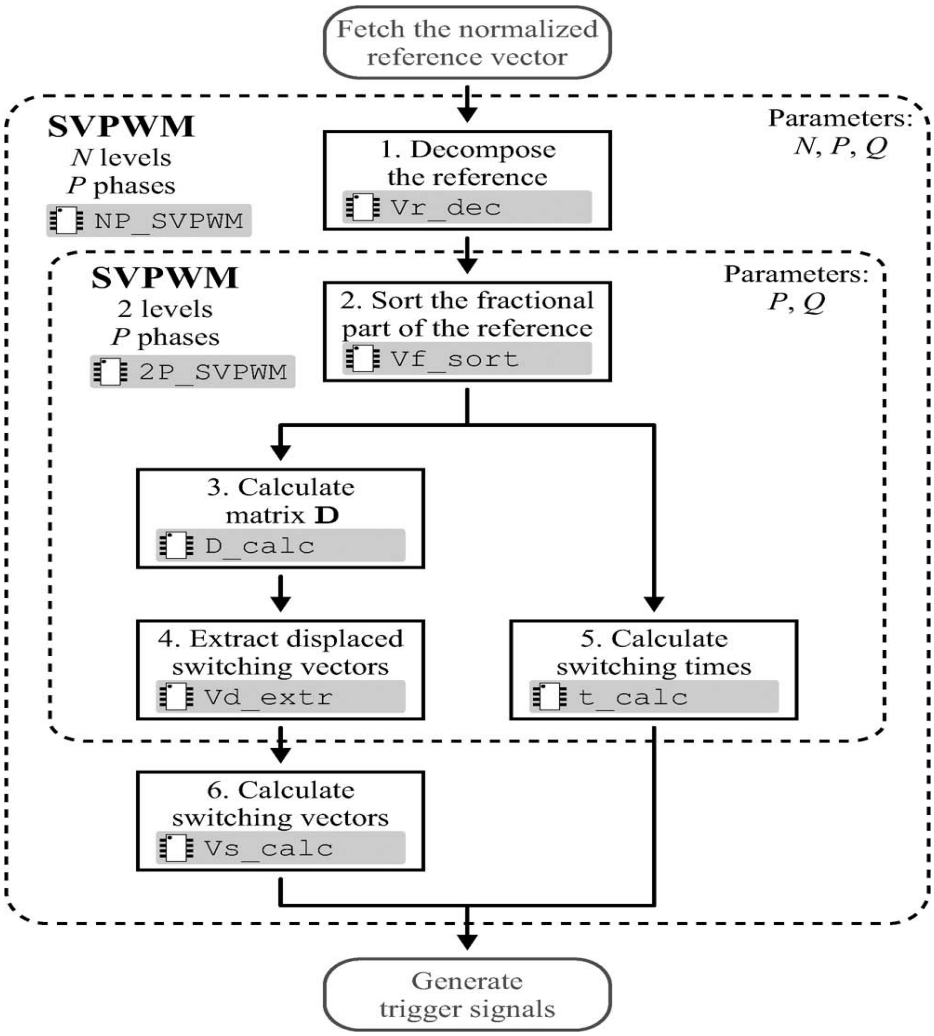
```
    TYPE array_D is ARRAY(0 TO P, 0 TO P) of INTEGER RANGE 0 TO 1;

    TYPE array_DP is ARRAY(1 TO P, 0 TO P) of INTEGER RANGE 0 TO 1;

    TYPE array_V is ARRAY(1 TO P) of INTEGER RANGE 0 TO 1;

    TYPE array_Vd is ARRAY(0 TO P) of array_V;

    TYPE array_for_Vs is ARRAY(1 TO P) of INTEGER RANGE -2 TO 2;

    TYPE array_Vs is ARRAY(0 TO P) of array_for_Vs;

    TYPE array_tj is ARRAY(1 TO P+1) of sfixed(I+1 downto -Q);

-------------------------------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------------------------------------------

end svpwm_constants;
```

Από το παραπάνω πακέτο βλέπουμε πως από τη μία ορίζονται ο αριθμός των φάσεων αλλά και των επιπέδων του μετατροπέα, τους οποίους μπορούμε να μεταβάλλουμε ανά πάσα στιγμή, και από την άλλη ορίζονται διάφοροι τύποι πινάκων οι οποίοι χρησιμοποιούνται από τα διάφορα συστατικά στοιχεία της σχεδίασης.

Τέλος να τονίσουμε πως το πακέτο αυτό όπως παρατηρούμε δεν έχει σώμα(Package Body) καθώς περιλαμβάνει μόνο δηλώσεις δεδομένων.

2) Η σχεδίασή μας είναι ανεξάρτητη της χρησιμοποιούμενης τεχνολογίας καθώς όλα τα κυκλώματα περιγράφονται εξ' ολοκλήρου με την γλώσσα περιγραφής υλικού VHDL. Μία τέτοια περιγραφή, επιτρέπει την υλοποίηση του κυκλώματος διαμόρφωσης σε οποιοδήποτε FPGA, ειδικού σκοπού ολοκληρωμένο κύκλωμα ή ισοδύναμο ψηφιακό ολοκληρωμένο κύκλωμα.

Μετά από αυτές τις δηλώσεις είμαστε έτοιμοι να αναλύσουμε βήμα-βήμα την υλοποίηση του αλγορίθμου, ο οποίος βασίζεται σε μία μετατόπιση και έναν δύο επιπέδων πολυφασικό SVPWM αλγόριθμο. Τα βήματα αυτά, έξι τον αριθμό, παρουσιάζονται από το παρακάτω διάγραμμα ροής και αναλύονται εκτενώς:



Σχήμα 4.5: Διάγραμμα ροής της πολυφασικής πολυεπίπεδης SVPWM μονάδας.

Για την κατανόηση και την επαλήθευση των αποτελεσμάτων της προσομοίωσης με τα θεωρητικώς αναμενόμενα εισάγαμε ως κανονικοποιημένο σήμα αναφοράς το παρακάτω:

V_{ref} = [1.43,1.13,-0.73,-1.58,-0.25].

Επειδή οι πραγματικοί(real) τύποι δεδομένων δεν είναι συνθέσιμοι στην γλώσσα VHDL για την υλοποίηση του παραπάνω διανύσματος αναφοράς χρησιμοποιούμε αριθμούς σταθερής υποδιαστολής όπως φαίνεται παρακάτω:

Vref: vector_Vref:=("001011011101", "001001000011", "111010001010", "110011010111", "111110000000");

Όπου ο πίνακας vector_Vref ορίζεται στο πακέτο svpwm_constants που έχουμε ορίσει παραπάνω. Με βάση αυτό παρουσιάζονται τα έξι βήματα υλοποίησης του αλγορίθμου:

4.3.2 Αποσύνθεση κανονικοποιημένου σήματος αναφοράς

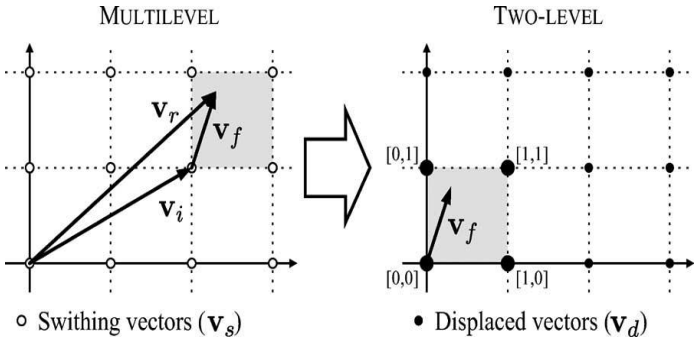
Το διάνυσμα αναφοράς μπορεί να αποσυντεθεί στο άθροισμα του ακεραίου μέρους $V_i = [u_i^1, u_i^2, \dots, u_i^P]^T$ και του κλασματικού του μέρους $V_f = [u_f^1, u_f^2, \dots, u_f^P]^T$. Παρατηρούμε πως το V_i περιέχει ακέραια στοιχεία και κατά συνέπεια μπορεί να συντεθεί κατ' ευθείαν από ένα διάνυσμα μεταγωγής. Το V_f όμως το οποίο είναι πραγματικός χρειάζεται μία ακολουθία διανυσμάτων μεταγωγής, τα οποία θα εφαρμόζονται για συγκεκριμένα χρονικά διαστήματα, για να μπορέσει να συντεθεί. Όπως είπαμε και παραπάνω θέλουμε να επιλύσουμε το παρακάτω σύστημα εξισώσεων:

$$\begin{bmatrix} 1 \\ u_r^1 \\ u_r^2 \\ . \\ . \\ . \\ u_r^P \end{bmatrix} = \begin{bmatrix} 1 & 1 & . & . & . & 1 \\ u_{s1}^1 & u_{s2}^1 & . & . & . & u_{sP+1}^1 \\ u_{s1}^2 & u_{s2}^2 & . & . & . & u_{sP+1}^2 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ u_{s1}^P & u_{s2}^P & . & . & . & u_{sP+1}^P \end{bmatrix} * \begin{bmatrix} t_1 \\ t_2 \\ . \\ . \\ . \\ t_{P+1} \end{bmatrix}$$

Για να το πετύχουμε αυτό δημιουργούμε μία ακολουθία μετατοπισμένων διανυσμάτων, μετατοπίζοντας το V_{sj} κατά το V_i διάνυσμα όπως φαίνεται στην παρακάτω σχέση:

- $V_{dj} = V_{sj} - V_i$

Αυτό γίνεται πιο κατανοητό από το παρακάτω σχήμα:



Σχήμα 4.6: Παράδειγμα Αποσύνθεσης στον Δισδιάστατο Χώρο

Πλέον το διάνυσμα αναφοράς V_r, μέσω της $V_{sj} = V_i + V_{dj}$, μπορεί να γραφεί όπως παρακάτω:

$$\begin{bmatrix} 1 \\ u_r^1 \\ u_r^2 \\ . \\ . \\ . \\ u_r^P \end{bmatrix} = \begin{bmatrix} 0 \\ u_i^1 \\ u_i^2 \\ . \\ . \\ . \\ u_i^P \end{bmatrix} + \begin{bmatrix} 1 & 1 & . & . & . & 1 \\ u_{d1}^1 & u_{d2}^1 & . & . & . & u_{dP+1}^1 \\ u_{d1}^2 & u_{d2}^2 & . & . & . & u_{dP+1}^2 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ u_{d1}^P & u_{d2}^P & . & . & . & u_{dP+1}^P \end{bmatrix} * \begin{bmatrix} t_1 \\ t_2 \\ . \\ . \\ . \\ t_{P+1} \end{bmatrix}$$

Όμως έχουμε αναφέρει και παραπάνω πως το διάνυσμα αναφοράς γράφεται και με τη σχέση:

$$\begin{bmatrix} 1 \\ u_r^1 \\ u_r^2 \\ . \\ . \\ . \\ u_r^P \end{bmatrix} = \begin{bmatrix} 0 \\ u_i^1 \\ u_i^2 \\ . \\ . \\ . \\ u_i^P \end{bmatrix} + \begin{bmatrix} 1 \\ u_f^1 \\ u_f^2 \\ . \\ . \\ . \\ u_f^P \end{bmatrix}$$

Συνεπώς από τις δύο παραπάνω εκφράσεις του διανύσματος αναφοράς V_r προκύπτει η έκφραση που μας δίνει το κλασματικό μέρος της αναφοράς V_f:

$$\begin{bmatrix} 1 \\ u_f^1 \\ u_f^2 \\ . \\ . \\ . \\ u_f^P \end{bmatrix} = \begin{bmatrix} 1 & 1 & . & . & . & 1 \\ u_{d1}^1 & u_{d2}^1 & . & . & . & u_{dP+1}^1 \\ u_{d1}^2 & u_{d2}^2 & . & . & . & u_{dP+1}^2 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ u_{d1}^P & u_{d2}^P & . & . & . & u_{dP+1}^P \end{bmatrix} * \begin{bmatrix} t_1 \\ t_2 \\ . \\ . \\ . \\ t_{P+1} \end{bmatrix}$$

Αυτό το νέο σύστημα γραμμικών εξισώσεων παριστάνει την ίδια μορφή όπως και ο αρχικός νόμος διαμόρφωσης. Όμως, σε αυτή την περίπτωση, τα στοιχεία του διανύσματος V_f είναι περιορισμένα στο διάστημα [0,1). Συνεπώς, αρκεί και μόνο το υποσύνολο των μετατοπισμένων διανυσμάτων με στοιχεία ίσα με μηδέν και ένα για την προσέγγιση του διανύσματος αναφοράς. Γίνεται λοιπόν κατανοητό πως αυτή η νέα εξίσωση αναπαριστά ένα διαμορφωτή δύο επιπέδων στον οποίο το σήμα αναφοράς είναι το κλασματικό μέρος V_f και η ακολουθία των διανυσμάτων μεταγωγής η ακολουθία V_{dj}. Οι χρόνοι μεταγωγής είναι οι ίδιοι και στις δύο περιπτώσεις. Το σχήμα 4.6 παρουσιάζει ένα δισδιάστατο παράδειγμα αποσύνθεσης, στο οποίο το διάνυσμα V_i συμπίπτει με ένα διάνυσμα μεταγωγής και το υποσύνολο {[0,0],[1,0],[0,1],[1,1]} των μετατοπισμένων διανυσμάτων είναι αρκετό για να συνθέσει το κλασματικό μέρος της αναφοράς V_f.

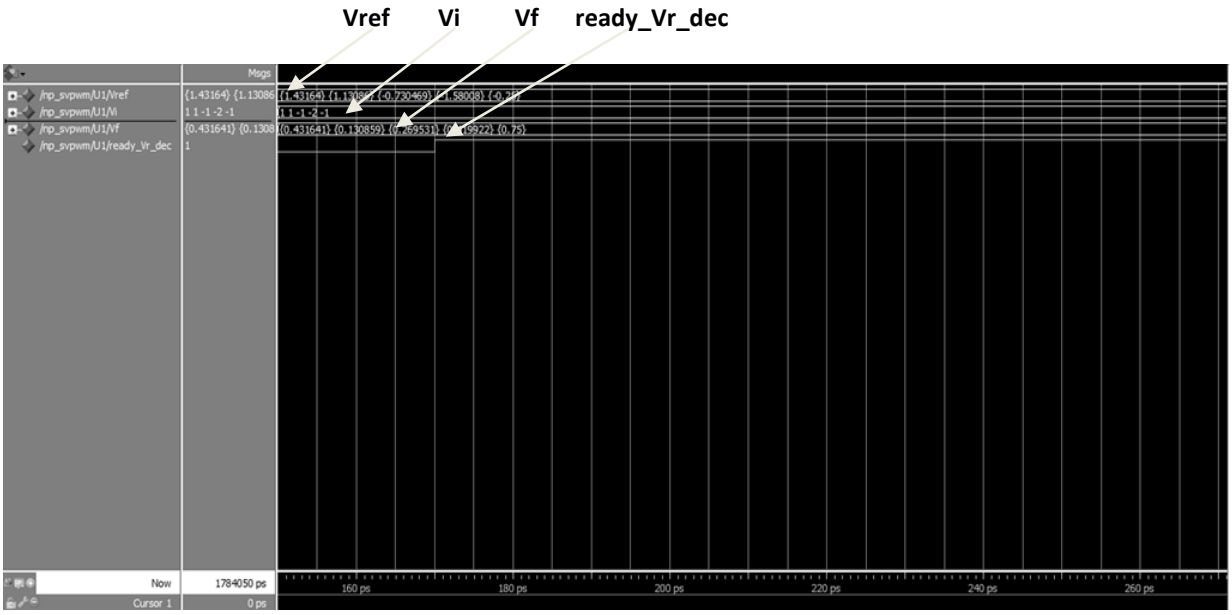
Η αποσύνθεση αυτή είναι δυνατόν να επιτευχθεί με το παρακάτω τμήμα κώδικα. Να τονίσουμε πως στα τμήματα κώδικα που θα παραθέτουμε για το εκάστοτε τμήμα της σχεδίασής μας θα παρουσιάζουμε μόνο την χρησιμοποιούμενη αρχιτεκτονική:

---------- Δήλωση και Υλοποίηση της Αρχιτεκτονικής που θα χρησιμοποιήσουμε ----------

```
Architecture behavioural of Vr_dec is

 begin

 PROCESS(clk,reset)

  variable counter: INTEGER;

        variable Vf_reg: vector_Vf:= (OTHERS => "1111111111111");

        variable Vi_reg: array_Vi:= (OTHERS => -2);
```

```
  begin
  if(clk'event and clk = '1') then
   if(reset = '1') then
     if(ready_Vr_dec = '0') then
      if (counter < P + 1) then
        Vi_reg(counter):= to_integer(Vref(counter)(I-1 downto 0));
        Vf_reg(counter):= Vref(counter) - to_sfixed(to_integer(Vref(counter)(I-1 downto 0)),I-1,-Q);
        counter:= counter + 1;
      else
        ready_Vr_dec<= '1';
      end if;
     end if;
    else
     counter:= 1;
     ready_Vr_dec<= '0';
     Vf_reg:= (OTHERS => "0000000000000");
     Vi_reg:= (OTHERS => 0001);
    end if;
  end if;
  Vi<= Vi_reg;
  Vf<= Vf_reg;
 end PROCESS;
end behavioural;
```

Προσομοιώνοντας την αρχιτεκτονική του όλου συστήματος και με βάση το διάνυσμα αναφοράς που περιγράψαμε παραπάνω προκύπτουν, με χρήση του εργαλείου προσομοίωσης Modelsim Altera 10.0c(Quartus II 11.1) Starter Edition, τα παρακάτω αποτελέσματα:



Σχήμα 4.7: Αποτελέσματα προσομοίωσης κυκλώματος Vr_dec με το Modelsim

Παρατηρούμε πως προκύπτουν τα παρακάτω, στρογγυλοποιημένα, αποτελέσματα τα οποία είναι όντως τα αναμενόμενα:

V_{ref} = [1.43,1.13,-0.73,-1.58,-0.25].

V_i = [1,1,-1,-2,-1].

V_f = [0.43,0.13,0.27,0.42,0.75].

Τέλος το σήμα ready_Vr_dec ενεργοποιείται μόλις το παρών κύκλωμα ολοκληρώσει τη λειτουργία του. Μόλις γίνει αυτό μπορεί πλέον να ξεκινήσει τη λειτουργία του το επόμενο κύκλωμα.

4.3.3 Ταξινόμηση κλασματικού μέρους αναφοράς

Στο βήμα αυτό ουσιαστικά υπολογίζουμε τον πίνακα μετάθεσης P, ο οποίος ταξινομεί το κλασματικό μέρος(V_f) του διανύσματος αναφοράς σε φθίνουσα σειρά σύμφωνα με την παρακάτω σχέση:

$$P * \begin{bmatrix} 1 \\ Vf \end{bmatrix} = \begin{bmatrix} 1 \\ Vf\_sort \end{bmatrix}$$

Όπου: $V_{f\_sort} = [V_{f\_sort}^{1}, V_{f\_sort}^{2}, \dots, V_{f\_sort}^{P}]^T$ το ταξινομημένο διάνυσμα του κλασματικού μέρους της αναφοράς για το οποίο ισχύει ότι $1 \geq V_{f\_sort}^{1} \geq V_{f\_sort}^{2} \geq \dots \geq V_{f\_sort}^{\kappa-1} \geq V_{f\_sort}^{\kappa} \geq V_{f\_sort}^{P} \geq 0$.

Αυτή η ταξινόμηση επιτυγχάνεται με τη χρήση του bubblesort αλγορίθμου κατά την εκτέλεση του οποίου γίνονται συγκρίσεις των διαδοχικών στοιχείων και αντίστοιχες αλλαγές θέσεων, ανάλογα με το αποτέλεσμα της σύγκρισης, μέχρις ότου να υπάρξει κάποιος κύκλος συγκρίσεων κατά τον οποίο δεν θα εκτελεστεί καμία εναλλαγή. Όταν συμβεί αυτό η ταξινόμηση έχει ολοκληρωθεί.

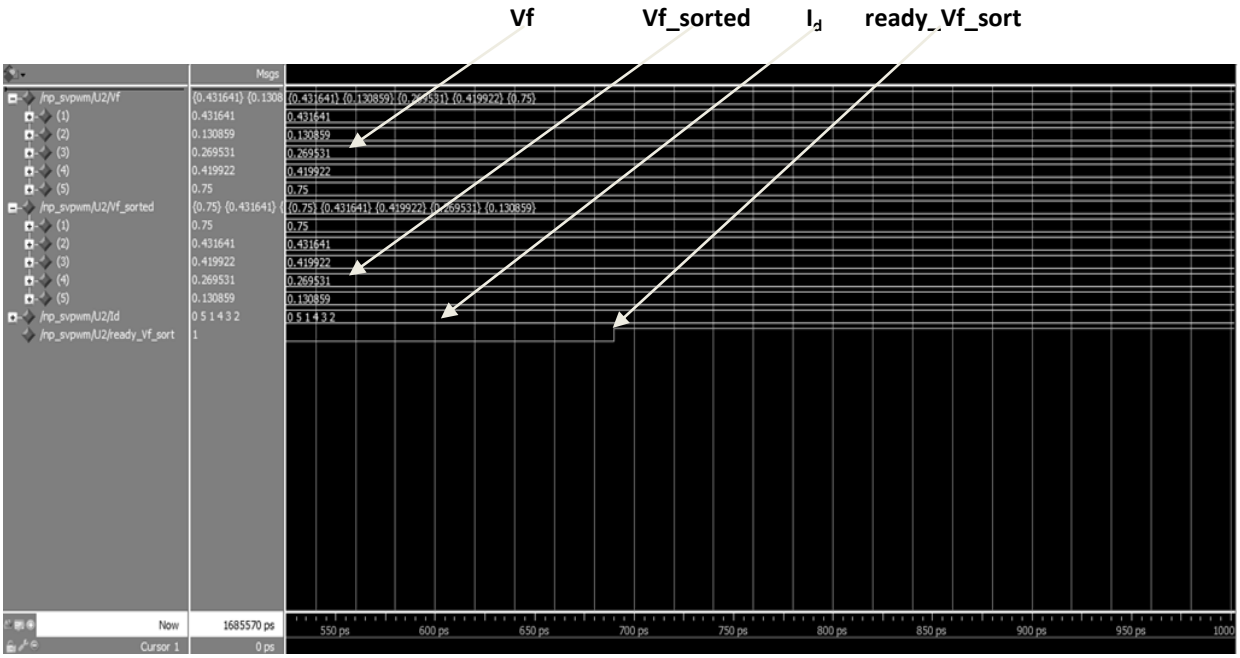
Η ταξινόμηση αυτή υλοποιείται με το παρακάτω τμήμα κώδικα:

```
---------- Δήλωση και Υλοποίηση της Αρχιτεκτονικής που θα χρησιμοποιήσουμε ----------
Architecture behavioural of Vf_sort is
 signal reg_Id: array_P;
 begin
  PROCESS(clk)
   variable counter1,counter2: INTEGER;
   variable temp,Vf_sorted_reg: vector_Vf:= (OTHERS => "1111111111111");
   variable NO_SWAPS: STD_LOGIC;
   begin
    if(clk'event and clk = '1') then
     if(reset = '1') then
      if(ready_Vr_dec = '1' and ready_Vf_sort = '0') then
       if(counter2 < P) then
        if(counter1 < P ) then
---------- Bubblesort Αλγόριθμος ----------
           if(counter1 = 0) then
            Vf_sorted_reg:= Vf;
            Id<= reg_Id;
           else
            if(Vf_sorted_reg(counter1) < Vf_sorted_reg(counter1 + 1))then
               temp(counter1):= Vf_sorted(counter1 + 1);
               Vf_sorted_reg(counter1 + 1):= Vf_sorted_reg(counter1);
               Vf_sorted_reg(counter1):= temp(counter1);
               NO_SWAPS:= '0';
               Id(counter1)<= Id(counter1 + 1);
               Id(counter1 + 1)<= Id(counter1);
```

```
            end if;
            temp:= Vf_sorted;
          end if;
          counter1:= counter1 + 1;
        else
          counter2:= counter2 + 1;
          counter1:= 1;
          if(NO_SWAPS = '1') then
            ready_Vf_sort<= '1';
          else
            NO_SWAPS:= '1';
          end if;
        end if;
      else
       ready_Vf_sort<= '1';
      end if;
    end if;
-------------------------------------------------------------------------------------------------------------------------------
   else
     counter1:= 0;
     counter2:= 0;
     NO_SWAPS:= '1';
     ready_Vf_sort<= '0';
     Vf_sorted_reg:= (OTHERS => "0000000000000");
   end if;
  end if;
  Vf_sorted<= Vf_sorted_reg;
```

```
  end PROCESS;


  G1: for a in 0 to P GENERATE

    reg_Id(a)<= a;

  end GENERATE;

end behavioural;
```

Προσομοιώνοντας το όλο σύστημα και με βάση το δεκαδικό μέρος της αναφοράς που προέκυψε από το προηγούμενο τμήμα προκύπτει το παρακάτω αποτέλεσμα:



Σχήμα 4.8: Αποτελέσματα προσομοίωσης κυκλώματος Vf_sort με το Modelsim

Παρατηρούμε πως προκύπτουν τα παρακάτω, στρογγυλοποιημένα, αποτελέσματα τα οποία είναι όντως τα αναμενόμενα:

V_f = [0.43,0.13,0.27,0.42,0.75].

V_{f_sorted} = [0.75,0.43,0.42,0.27,0.13].

I_d = [0,5,1,4,3,2].

Παρατηρούμε πως όντως ο πίνακας V_{f_sorted} αποτελείται από τα ταξινομημένα στοιχεία του πίνακα V_f κατά φθίνουσα σειρά.

Επίσης ο πίνακας I_d περιέχει την θέση στην οποία βρίσκονταν τα ταξινομημένα πλέον στοιχεία πριν γίνει η ταξινόμηση. Παρατηρούμε για παράδειγμα πως το στοιχείο 0.75 αρχικά, στον πίνακα V_f, βρισκόταν στην $5^η$ θέση ενώ στον ταξινομημένο V_{f_sorted} βρίσκεται στην $1^η$ και αυτός είναι και ο λόγος που ο αριθμός 5 βρίσκεται στο $1^ο$ στοιχείο του πίνακα I_d:

V_f = [0.43,0.13,0.27,0.42,0.75].

V_{f_sorted} = [0.75,0.43,0.42,0.27,0.13].

I_d = [0,5,1,4,3,2].

Μέσω του πίνακα αυτού μπορούμε στο κύκλωμα D_calc να υλοποιήσουμε τον πίνακα Per.

Τέλος το σήμα ready_Vf_sort ενεργοποιείται μόλις το παρών κύκλωμα ολοκληρώσει τη λειτουργία του. Μόλις γίνει αυτό μπορεί πλέον να ξεκινήσει τη λειτουργία του το επόμενο κύκλωμα.

4.3.4 Υπολογισμός πίνακα D

Ο πίνακας D ορίζεται με βάση τη σχέση:

$$\begin{bmatrix} 1 \\ u_f^1 \\ u_f^2 \\ . \\ . \\ . \\ u_f^P \end{bmatrix} = \begin{bmatrix} 1 & 1 & . & . & . & 1 \\ u_{d1}^1 & u_{d2}^1 & . & . & . & u_{dP+1}^1 \\ u_{d1}^2 & u_{d2}^2 & . & . & . & u_{dP+1}^2 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ u_{d1}^P & u_{d2}^P & . & . & . & u_{dP+1}^P \end{bmatrix} * \begin{bmatrix} t_1 \\ t_2 \\ . \\ . \\ . \\ t_{P+1} \end{bmatrix} \Rightarrow \begin{bmatrix} 1 \\ V_f \end{bmatrix} = D * t$$

Για την επίλυση του συστήματος αυτού υπάρχουν πολλοί τρόποι. Η απόδοση όμως του όλου συστήματος βασίζεται στη μέθοδο που θα χρησιμοποιήσουμε για τον υπολογισμό του πίνακα αυτού. Για να ελαχιστοποιηθούν οι απώλειες μεταγωγής πρέπει οι συντελεστές να επιλεγούν με τέτοιο τρόπο ώστε τα διαδοχικά διανύσματα μεταγωγής της ακολουθίας των διανυσμάτων να είναι γειτονικά. Με άλλα λόγια, πρέπει μόνο ένας συντελεστής να διαφέρει ανάμεσα σε δύο διαδοχικές στήλες του πίνακα που αναζητούμε. Μία μέθοδος που οδηγεί στο αποτέλεσμα αυτό περιγράφεται παρακάτω:

Όπως είπαμε ισχύει:

$$\begin{bmatrix} 1 \\ V_f \end{bmatrix} = D * t$$

Στο παραπάνω βήμα βρήκαμε τον πίνακα P ο οποίος ταξινομεί το κλασματικό μέρος της αναφοράς V_f σε φθίνουσα σειρά με βάση την παρακάτω σχέση:

$$P * \begin{bmatrix} 1 \\ V_f \end{bmatrix} = \begin{bmatrix} 1 \\ V_{f_{sorted}} \end{bmatrix}$$

Από τις δύο παραπάνω σχέσεις προκύπτει:

$\begin{bmatrix} 1 \\ V_{f_{sorted}} \end{bmatrix}$ = D_trig * t

όπου D_trig = P * D.

Ένας πίνακας D_trig με διαδοχικές γειτονικές στήλες, διαδοχικές δηλαδή στήλες οι οποίες διαφέρουν κατά ένα μόνο συντελεστή, ο οποίος καθιστά το νέο αυτό σύστημα γραμμικών εξισώσεων απολύτως ορισμένο είναι ο ακόλουθος άνω τριγωνικός πίνακας:

$$D\_trig = \begin{bmatrix} 1 & 1 & 1 & . & . & . & 1 \\ 0 & 1 & 1 & . & . & . & 1 \\ 0 & 0 & 1 & . & . & . & 1 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 0 & 0 & 0 & . & . & . & 1 \end{bmatrix}$$

Όπως θα φανεί και παρακάτω οι χρόνοι μεταγωγής που προκύπτουν με τη χρήση του πίνακα αυτού είναι πάντα θετικοί.

Ο πίνακας P που έχουμε υπολογίσει παραπάνω είναι ορθογώνιος. Συνεπώς είναι αντιστρέψιμος και μάλιστα ισχύει και η παρακάτω σχέση:

$P^{-1} = P^T$

Άρα λοιπόν ο πίνακας D τον οποίο αναζητούμε στο βήμα αυτό μπορεί εύκολα να προκύψει από την εξίσωση:

$D = P^T * D\_trig$

Αξίζει στο σημείο αυτό να τονίσουμε πως ο πίνακας P εφαρμόζει ένα σετ από αλλαγές γραμμών στα στοιχεία του πίνακα V_f. Αντίστοιχα, το αντίστροφο σετ από αλλαγές εφαρμόζεται στο πίνακα D_trig από τον πίνακα P^T και κατά συνέπεια ο αριθμός των ένα και των μηδενικών που υπάρχουν σε κάθε στήλη παραμένει αμετάβλητος. Άρα, ο αριθμός των αλλαγών καταστάσεων των διακοπτών ελαχιστοποιείται καθώς τα διαδοχικά διανύσματα της ακολουθίας παραμένουν γειτονικά και μετά το μετασχηματισμό.

Περνώντας στον υλοποίηση του συγκεκριμένου βήματος με την γλώσσα VHDL, αρχικά κατασκευάζουμε τον άνω τριγωνικό πίνακα του οποίο ο αριθμός των γραμμών και των στηλών εξαρτάται από τις φάσεις που χρησιμοποιούμε στην εκάστοτε σχεδίαση.

Στη συνέχεια με βάση την πληροφορία που έχουμε αποκομίσει από το προηγούμενο κύκλωμα, και την έχουμε αποθηκευμένη στον πίνακα Id, για την θέση στην οποία βρισκόταν το κάθε στοιχείο του κλασματικού μέρους της αναφοράς πριν αυτό μετατοπιστεί, αν έχει μετατοπιστεί, δημιουργούμε τον πίνακα Per. Ταυτόχρονα δημιουργούμε και τον αντίστροφό του Per_T ο οποίος πολλαπλασιαζόμενος με τον άνω τριγωνικό πίνακα D_trig αναδιατάσσει κατάλληλα τις γραμμές του, σε σχέση πάντα με την

διαδικασία ταξινόμησης που ακολουθήθηκε στο δεύτερο βήμα. Συνεπώς ο επιθυμητός πίνακας D παράγεται από τον παρακάτω πολλαπλασιασμό:

D = Per_T * D_trig

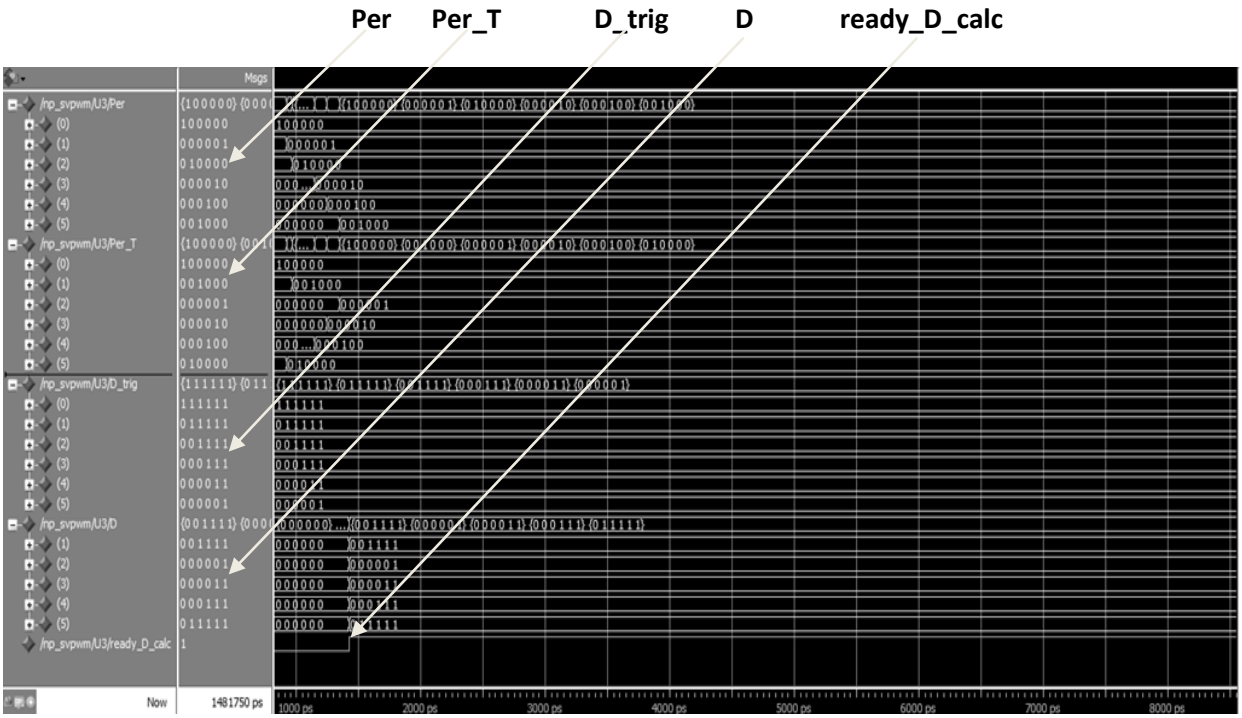
Ο υπολογισμός του πίνακα D υλοποιείται με το παρακάτω τμήμα κώδικα:

```
---------- Δήλωση και Υλοποίηση της Αρχιτεκτονικής που θα χρησιμοποιήσουμε ----------
Architecture behavioural of D_calc is
 signal D_trig: array_D;
 begin
 PROCESS(clk)
  variable counter1,counter2: INTEGER;
  variable Per_ready: STD_LOGIC;
  variable D_reg,Per,Per_T: array_D;
  begin
  if(clk'event and clk = '1') then
   if(reset = '1') then
    if(ready_Vf_sort = '1') then
     if(counter2 < P + 1) then
      if(counter1 = Id_P(counter2)) then
       Per(counter2,counter1):= 1;
       Per_T(counter1,counter2):= 1;
      else
       Per(counter2,counter1):= 0;
       Per_T(counter1,counter2):= 0;
      end if;
      counter1:= counter1 + 1;
      if(counter1 = P + 1) then
       counter2:= counter2 + 1;
```

```
      counter1:= 0;
    end if;
   else
    for a in 0 to P LOOP
     for b in 0 to P LOOP
       D_reg(a,b):= 0;
     end LOOP;
    end LOOP;
    Per_ready:= '1';
   end if;
  end if;
  if(Per_ready = '1') then
   for a in 0 to P LOOP
    for b in 0 to P LOOP
     for c in 0 to P LOOP
      D_reg(a,b):= D_reg(a,b) + Per_T(a,c)*D_trig(c,b);
     end LOOP;
    end LOOP;
   end LOOP;
   for a in 1 to P LOOP
    for b in 0 to P LOOP
     D(a,b)<= D_reg(a,b);
    end LOOP;
   end LOOP;
   ready_D_calc<= '1';
  end if;
 else
```

```
      counter1:= 0;
      counter2:= 0;
      ready_D_calc<= '0';
      Per_ready:= '0';
    end if;
  end if;
 end PROCESS;
  ------------------------------------ Υλοποίηση Τριγωνικού Πίνακα ----------------------------------------------------------
  G1: for j IN 0 TO P GENERATE
    G2: for k IN j to P GENERATE
      D_trig(j,k)<= 1;
    end GENERATE;
  end GENERATE;

  G3: for l IN 1 TO P GENERATE
    G4: for m IN 0 TO l-1 GENERATE
      D_trig(l,m)<= 0;
    end GENERATE;
  end GENERATE;
 end behavioural;
-------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------------
```

Προσομοιώνοντας το όλο σύστημα και με βάση τον πίνακα I_d που προέκυψε από το παραπάνω κύκλωμα προκύπτουν τα παρακάτω αποτελέσματα:



Σχήμα 4.9: Αποτελέσματα προσομοίωσης κυκλώματος D_calc με το Modelsim

Παρατηρούμε από την παραπάνω προσομοίωση πως προκύπτουν τα παρακάτω αποτελέσματα τα οποία είναι όντως τα επιθυμητά:

I_d = [0,5,1,4,3,2].

$$\text{Per} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

Παρατηρούμε πως στον πίνακα Per τα ‘1’ τοποθετούνται στην κάθε σειρά στην θέση που υποδεικνύεται από τα εκάστοτε στοιχεία του πίνακα I_d.

$$\text{Per\_T} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Ο πίνακας Per_T είναι ο ανάστροφος του Per.

$$D\_trig = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

$$D = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Παρατηρούμε πως ο πίνακας D έχει μία λιγότερη γραμμή από αυτές που θα αναμέναμε και αυτό συμβαίνει καθώς την πρώτη του γραμμή δεν την χρειαζόμαστε στο παρακάτω βήμα και γι αυτό την παραλείπουμε.

Τέλος το σήμα ready_D_calc ενεργοποιείται μόλις το παρών κύκλωμα ολοκληρώσει τη λειτουργία του. Μόλις γίνει αυτό μπορεί πλέον να ξεκινήσει τη λειτουργία του το επόμενο κύκλωμα.

4.3.5 Εξαγωγή μετατοπισμένων διανυσμάτων μεταγωγής

Στο βήμα αυτό εξάγουμε την μετατοπισμένη ακολουθία των διανυσμάτων μεταγωγής V_{dj} χρησιμοποιώντας τον πίνακα D τον οποίο υπολογίσαμε στο προηγούμενο βήμα. Αυτό γίνεται με βάση την παρακάτω θεώρηση του πίνακα D:

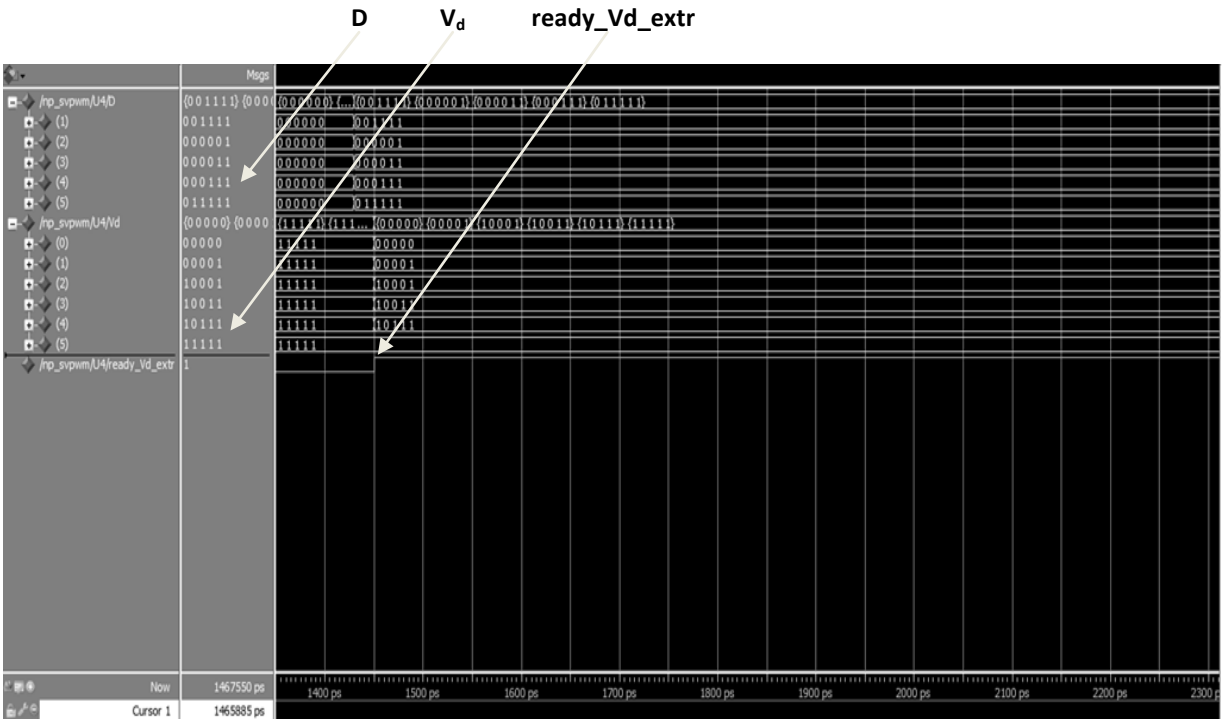
$$D = \begin{bmatrix} 1 & 1 & . & . & . & 1 \\ u_{d1}^{1} & u_{d2}^{1} & . & . & . & u_{d_{P+1}}^{1} \\ u_{d1}^{2} & u_{d2}^{2} & . & . & . & u_{d_{P+1}}^{2} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ u_{d1}^{P} & u_{d2}^{P} & . & . & . & u_{d_{P+1}}^{P} \end{bmatrix}$$

Ο υπολογισμός της ακολουθίας των διανυσμάτων αυτών γίνεται με την χρησιμοποίηση του παρακάτω τμήματος κώδικα:

```
---------- Δήλωση και Υλοποίηση της Αρχιτεκτονικής που θα χρησιμοποιήσουμε ----------

Architecture behavioural of Vd_extr is

  begin

    PROCESS(clk)

     variable Vd_reg: array_Vd:= (OTHERS => (OTHERS => 1));

     begin

       if(clk'event and clk = '1') then
```

```
        if(reset = '1') then
          if(ready_D_calc = '1') then
            for a in 0 to P LOOP
              for b in 1 to P LOOP
                Vd_reg(a)(b):= D(b,a);
              end LOOP;
            end LOOP;
            ready_Vd_extr<= '1';
          end if;
        else
          ready_Vd_extr<= '0';
        end if;
      end if;
      Vd<= Vd_reg;
    end PROCESS;
  end behavioural;
```

Προσομοιώνοντας το όλο σύστημα και με βάση τον πίνακα D που προέκυψε από το παραπάνω κύκλωμα προκύπτουν τα παρακάτω αποτελέσματα:



Σχήμα 4.10: Αποτελέσματα προσομοίωσης κυκλώματος Vd_extr με το Modelsim

Παρατηρούμε από την παραπάνω προσομοίωση πως προκύπτουν τα παρακάτω αποτελέσματα τα οποία είναι όντως τα επιθυμητά:

$$D = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

$$Vd = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Προκύπτουν δηλαδή ουσιαστικά τα παρακάτω διανύσματα:

$V_d(1) = [0,0,0,0,0]^T$

$V_d(2) = [0,0,0,0,1]^T$

$V_d(3) = [1,0,0,0,1]^T$

$V_d(4) = [1,0,0,1,1]^T$

$V_d(5) = [1,0,1,1,1]^T$

$V_d(6) = [1,1,1,1,1]^T$

Τέλος το σήμα ready_Vd_extr ενεργοποιείται μόλις το παρών κύκλωμα ολοκληρώσει τη λειτουργία του. Μόλις γίνει αυτό μπορεί πλέον να ξεκινήσει τη λειτουργία του το επόμενο κύκλωμα.

4.3.6 Υπολογισμός χρόνων μεταγωγής

Στο βήμα αυτό υπολογίζουμε τα χρονικά διαστήματα για τα οποία θα εφαρμόζονται οι εκάστοτε διακοπτικοί συνδυασμοί. Αυτό το καταφέρνουμε χρησιμοποιώντας το ταξινομημένο διάνυσμα το οποίο υπολογίσαμε στο δεύτερο βήμα της σχεδίασής μας με βάση την παρακάτω σχέση:

$$\begin{bmatrix} 1 \\ V_{f_{sorted}} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & . & . & . & 1 \\ 0 & 1 & 1 & . & . & . & 1 \\ 0 & 0 & 1 & . & . & . & 1 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 0 & 0 & 0 & . & . & . & 1 \end{bmatrix} * \begin{bmatrix} t_1 \\ t_2 \\ . \\ . \\ . \\ t_{P+1} \end{bmatrix}$$

Αφαιρώντας τις διαδοχικές γραμμές που προκύπτουν από το παραπάνω σύστημα προκύπτουν οι παρακάτω χρόνοι μεταγωγής:

$$t_j = \begin{cases} 1 - uf_{sorted}^{1} \\ uf_{sorted}^{j-1} - uf_{sorted}^{j} \\ uf_{sorted}^{P} \end{cases}$$

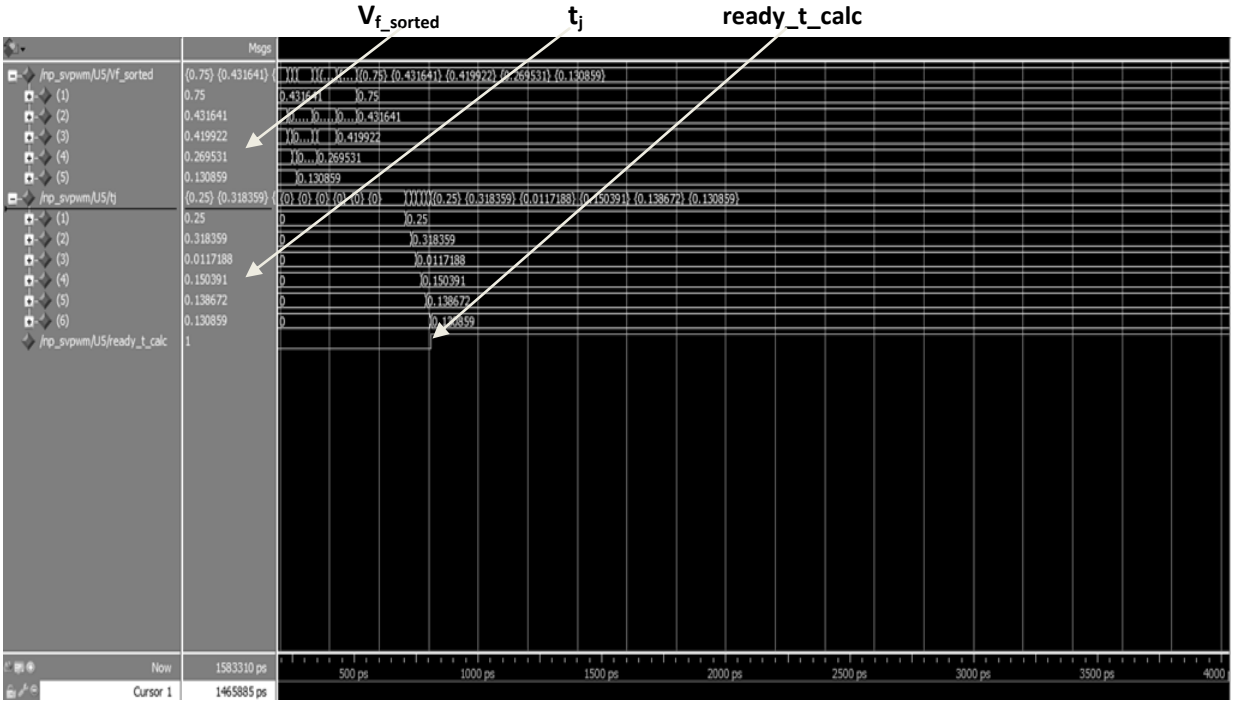
Παρατηρούμε πως για τον υπολογισμό των χρόνων μεταγωγής, είναι απαραίτητο μόνο το ταξινομημένο διάνυσμα V_{f_sorted}. Συνεπώς, ξεκινά ο υπολογισμός τους αμέσως μετά την ολοκλήρωση της ταξινόμησης και την δημιουργία του V_{f_sorted}.

Τα χρονικά αυτά διαστήματα υπολογίζονται μέσω του παρακάτω κώδικα:

```
---------- Δήλωση και Υλοποίηση της Αρχιτεκτονικής που θα χρησιμοποιήσουμε ----------

Architecture behavioural of t_calc is

 begin

  PROCESS(clk)

   variable counter: INTEGER;

   variable tj_reg: array_tj:= (OTHERS => "11111111111111");

   begin

   if(clk'event and clk = '1') then
```

```
      if(reset = '1') then
        if(ready_Vf_sort = '1') then
          if(ready_t_calc = '0') then
            if(counter = 1) then
              tj_reg(counter):= to_sfixed(1,I,-Q) - Vf_sorted(counter);
              counter:= counter + 1;
            elsif(counter < P + 1) then
              tj_reg(counter):= Vf_sorted(counter - 1) - Vf_sorted(counter);
              counter:= counter + 1;
            elsif(counter = P + 1) then
              tj_reg(counter):= to_sfixed(0,I,-Q) + Vf_sorted(counter - 1);
              counter:= 1;
              ready_t_calc<= '1';
            end if;
          end if;
        end if;
      else
        counter:= 1;
        ready_t_calc<= '0';
        tj_reg:= (OTHERS => "00000000000000");
      end if;
    end if;
    tj<= tj_reg;
  end PROCESS;
end behavioural;
```

Προσομοιώνοντας το όλο σύστημα και με βάση το ταξινομημένο διάνυσμα του κλασματικού μέρους της αναφοράς V_{f_sorted}, το οποίο έχει υπολογιστεί από το κύκλωμα Vf_sort προκύπτουν τα παρακάτω αποτελέσματα:



Σχήμα 4.11: Αποτελέσματα προσομοίωσης κυκλώματος t_calc με το Modelsim

Παρατηρούμε από την παραπάνω προσομοίωση πως προκύπτουν τα παρακάτω, στρογγυλοποιημένα, αποτελέσματα τα οποία είναι όντως τα επιθυμητά:

V_{f_sorted} = [0.75,0.43,0.42,0.27,0.13].

t_j = [0.25,0.32,0.01,0.15,0.14,0.13].

Οι χρόνοι αυτοί αντιστοιχούν στο ποσοστό εφαρμογής του εκάστοτε διακοπτικού συνδυασμού αναφορικά με τη συνολική περίοδο διαμόρφωσης.

Παρατηρούμε πως όντως είναι όλοι μεγαλύτεροι του μηδενός αλλά και το γεγονός πως το άθροισμα όλων αυτών των χρονικών διαστημάτων ισούται με την μονάδα.

4.3.7 Υπολογισμός διανυσμάτων μεταγωγής

Στο βήμα αυτό υπολογίζουμε την τελική ακολουθία διανυσμάτων μεταγωγής με βάση την παρακάτω σχέση:

$V_{sj} = V_i + V_{dj}$

Όπου

V_{sj} : Τελική ακολουθία διανυσμάτων μεταγωγής.

V_i : Ακέραιο μέρος του διανύσματος αναφοράς V_r.

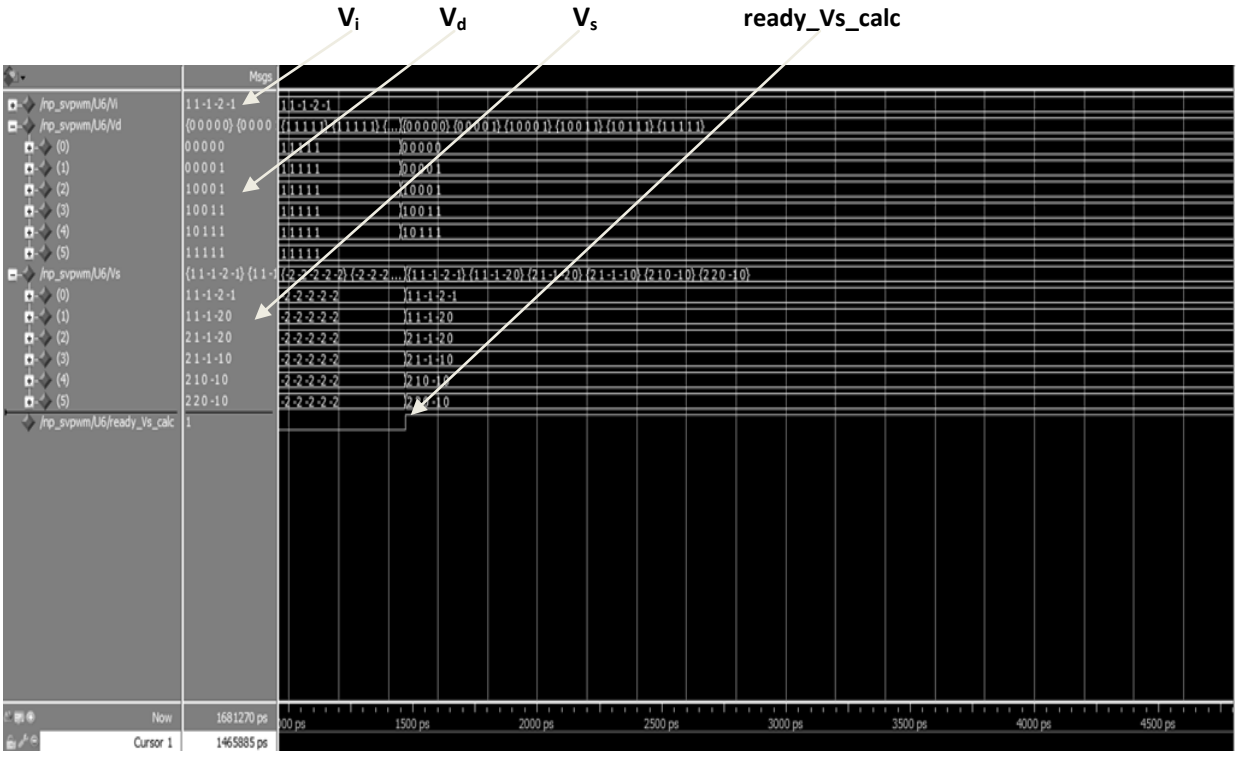
V_{dj} : Μετατοπισμένα διανύσματα μεταγωγής.

Η ακολουθία αυτή των διανυσμάτων υπολογίζεται μέσω του παρακάτω τμήματος κώδικα:

```
---------- Δήλωση και Υλοποίηση της Αρχιτεκτονικής που θα χρησιμοποιήσουμε ----------
Architecture behavioural of Vs_calc is
 begin
  PROCESS(clk)
   variable Vs_reg: array_Vs:= (OTHERS => (OTHERS => 2));
    begin
     if(clk'event and clk = '1') then
       if(reset = '1') then
        if(ready_Vd_extr = '1') then
          for a in 0 to P LOOP
            for b in 1 to P LOOP
              Vs_reg(a)(b):= Vi(b) + Vd(a)(b);
            end LOOP;
          end LOOP;
          --Vs<= Vs_reg;
          ready_Vs_calc<= '1';
        end if;
       else
        Vs_reg:=(OTHERS => (OTHERS => -2));
        ready_Vs_calc<= '0';
       end if;
     end if;
```

```
    Vs<= Vs_reg;

  end PROCESS;

end behavioural;
```

Προσομοιώνοντας το όλο σύστημα και με βάση το ακέραιο μέρος της αναφοράς που υπολογίσαμε στο κύκλωμα Vr_dec αλλά και την ακολουθία των μετατοπισμένων διανυσμάτων V_d που υπολογίσαμε στο κύκλωμα Vd_extr προκύπτουν τα παρακάτω αποτελέσματα:



Σχήμα 4.12: Αποτελέσματα προσομοίωσης κυκλώματος Vs_calc με το Modelsim

Παρατηρούμε από την παραπάνω προσομοίωση πως προκύπτουν τα παρακάτω αποτελέσματα τα οποία είναι όντως τα επιθυμητά:

V_i = [1,1,-1,-2,-1].

$$Vd = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

$$Vs = \begin{bmatrix} 1 & 1 & -1 & -2 & -1 \\ 1 & 1 & -1 & -2 & 0 \\ 2 & 1 & -1 & -2 & 0 \\ 2 & 1 & -1 & -1 & 0 \\ 2 & 1 & 0 & -1 & 0 \\ 2 & 2 & 0 & -1 & 0 \end{bmatrix}.$$

Προκύπτουν δηλαδή ουσιαστικά τα παρακάτω διανύσματα:

$V_S(1) = [1,1,-1,-2,-1]^T$

$V_S(2) = [1,1,-1,-2,0]^T$

$V_S(3) = [2,1,-1,-2,0]^T$

$V_S(4) = [2,1,-1,-1,0]^T$

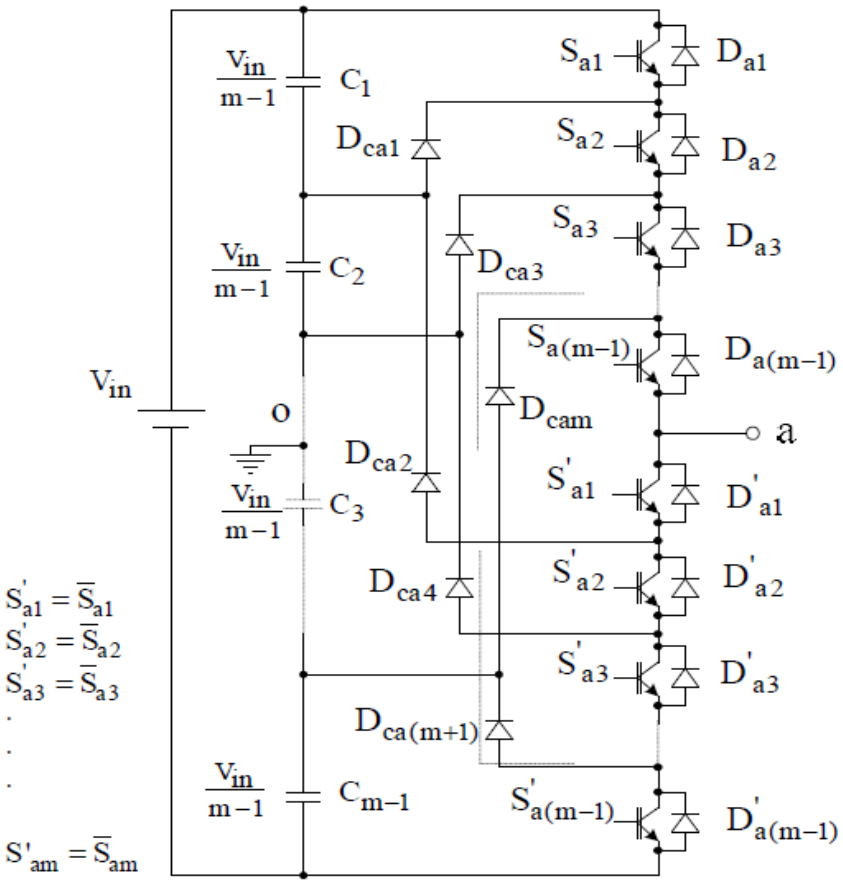
$V_S(5) = [2,1,0,-1,0]^T$

$V_S(6) = [2,2,0,-1,0]^T$

Παρατηρούμε πως όπως αναμέναμε και είχαμε τονίσει παραπάνω, από διάνυσμα σε διάνυσμα έχουμε μόνο μία αλλαγή τιμής κάτι που οδηγεί σε μεγάλη μείωση των απωλειών του συστήματος.

Επίσης όπως έχουμε αναφέρει και παραπάνω, τα διανύσματα αυτά μας δίνουν την κανονικοποιημένη τιμή της τάσης εξόδου στην οποία πρέπει να βρίσκεται η κάθε φάση. Ο προγραμματισμός των διακοπτών για την επίτευξη αυτής της τάσης, διαφέρει από τοπολογία σε τοπολογία αναστροφέα. Οι πιο συχνά παρατηρούμενες τοπολογίες πολυεπίπεδων αναστροφέων παρουσιάζονται παρακάτω:

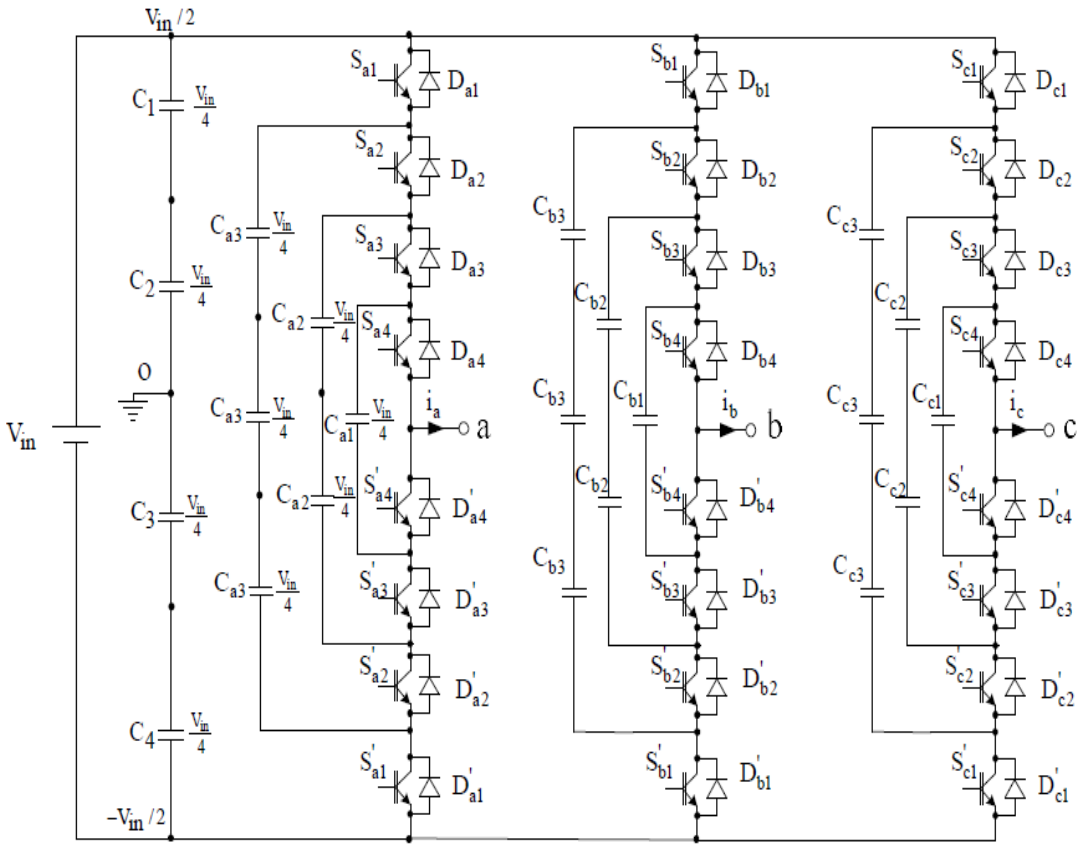
- **Αναστροφείς Πολλαπλών Επιπέδων με Διόδους Περιορισμού(Diode-Clamped Multilevel Inverters,DCMI):**
  Σε αυτή την τοπολογία χρησιμοποιούνται οι λεγόμενες δίοδοι περιορισμού οι οποίες εξασφαλίζουν σταθερή τάση αποκοπής στα άκρα κάθε ημιαγώγιμου διακόπτη. Χαρακτηριστικό της τοπολογίας αυτής είναι το γεγονός πως για την επίτευξη του κάθε επιπέδου τάσης είναι απαραίτητη η αγωγή συγκεκριμένων διακοπτικών στοιχείων. Το σκέλος μίας φάσης ενός αναστροφέα με διόδους περιορισμού m επιπέδων τάσης παρουσιάζεται στο παρακάτω σχήμα:



Σχήμα 4.13: Ένα σκέλος μίας φάσης ενός αναστροφέα με διόδους περιορισμού m επιπέδων τάσης(DCMI)

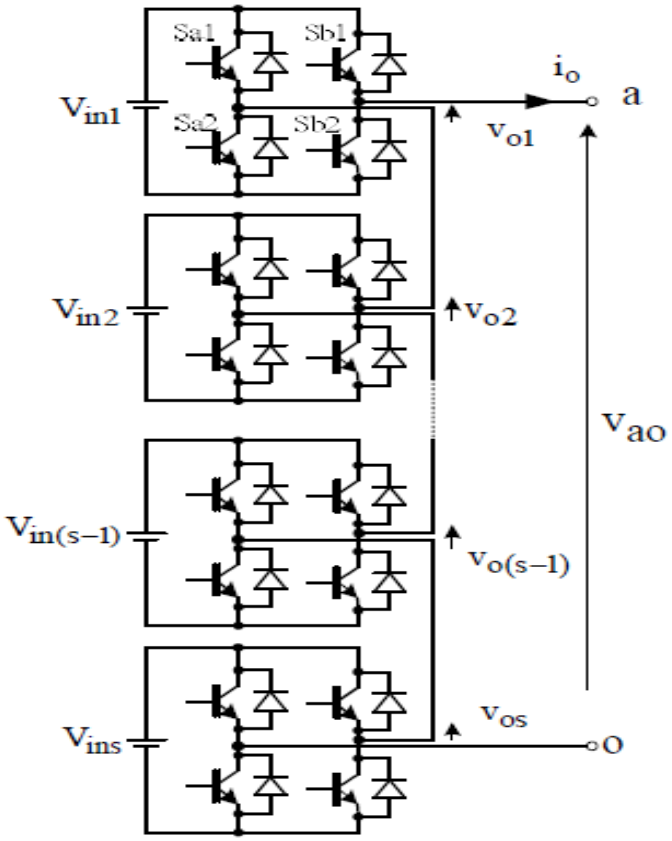
- **Αναστροφείς πολλαπλών επιπέδων με πυκνωτές περιορισμού(Flying-Capacitors Multilevel Inverters, FCMI):**

  Ο αναστροφέας πολλαπλών επιπέδων με πυκνωτές περιορισμού, ή αλλιώς με πλωτούς πυκνωτές, είναι μία παραλλαγή του αναστροφέα DCMI. Αντί των διόδων περιορισμού οι αναστροφείς FCMI διαθέτουν πυκνωτές. Χαρακτηριστικό της τοπολογίας αυτής είναι πως σε αντίθεση με τον DCMI αναστροφέα το κάθε επίπεδο τάσης μπορεί να δημιουργηθεί από διάφορους διακοπτικούς συνδυασμούς και όχι μόνο από κάποιο συγκεκριμένο. Το κύκλωμα του τριφασικού αναστροφέα FCMI πέντε επιπέδων παρουσιάζεται παρακάτω:



Σχήμα 4.14: Τριφασικός αναστροφέας FCMI πέντε επιπέδων

- **Αναστροφείς πολλαπλών επιπέδων αποτελούμενοι από επιμέρους μονοφασικούς αναστροφείς συνδεδεμένους σε σειρά με ανεξάρτητες πηγές τάσης τροφοδοσίας(Multilevel Inverters using Cascaded-Inverters with Separated DC Sources, SDCSMI):**
  Οι SDCSMI αναστροφείς συνθέτουν την επιθυμητή τάση εξόδου τροφοδοτούμενοι από ένα σύνολο ανεξαρτήτων πηγών DC τάσης, πηγές οι οποίες μπορεί να σχετίζονται με συσσωρευτές, με κυψέλες υδρογόνου ή με φωτοβολταϊκές κυψέλες. Οι αναστροφείς αυτοί αποτελούνται από μονάδες αναστροφέων που οι έξοδοί τους είναι συνδεδεμένες σε σειρά. Το κύκλωμα ισχύος μίας φάσης ενός SDCSMI αναστροφέα που έχει υλοποιηθεί από μονάδες μονοφασικού αναστροφέα πλήρους γέφυρας παρουσιάζεται παρακάτω:



Σχήμα 4.15: Κύκλωμα ισχύος μίας φάσης SDCSMI αναστροφέα που έχει υλοποιηθεί από μονάδες μονοφασικού αναστροφέα πλήρους γέφυρας

4.4 Προσομοίωση και πειραματική διαδικασία της διφασικής SVPWM τεχνικής

Στο σημείο αυτό για να γίνουν πιο εμφανή τα παραπάνω θα παρουσιάσουμε την περίπτωση που αναλύσαμε εκτενώς και στο τρίτο κεφάλαιο, το χειρισμό δηλαδή των διακοπτών του μονοφασικού αναστροφέα S1,S2,S3,S4. Για να το πετύχουμε αυτό κάνουμε κάποιες μικρές αλλαγές στην ήδη υπάρχουσα σχεδίασή μας οι οποίες παρουσιάζονται παρακάτω. Επίσης τα αποτελέσματα που λαμβάνουμε από την υλοποίηση αυτή τα παρατηρούμε με την χρήση του εργαλείου προσομοίωσης Modelsim Altera 10.0c(Quartus II 11.1) Starter Edition, αλλά και μέσω του παλμογράφου αφού πρώτα έχουμε ‘’φορτώσει’’ τη σχεδίασή μας στο FPGA Spartan 6 XC6LX16-CS324.

4.4.1 Αποτελέσματα προσομοίωσης σχεδίασης

Όπως αναφέραμε και παραπάνω, θα μελετήσουμε την περίπτωση παραγωγής των παλμών που αφορούν στο χειρισμό του μονοφασικού αναστροφέα του σχήματος 3.1. Για να το πετύχουμε αυτό θα πρέπει να θέσουμε τον αριθμό των φάσεων(a,b) που χρησιμοποιούντα από τη σχεδίασή μας σε δύο

αλλά και των αριθμό των επιπέδων σε τρία(1,0,-1). Αυτό γίνεται μέσω του πακέτου svpwm_constants με τις παρακάτω δηλώσεις:

CONSTANT N: INTEGER RANGE 0 TO 127:= 3; -- Number of levels of the converter.

CONSTANT P: INTEGER RANGE 0 TO 127:= 2; -- Number of phases of the converter.

Εκτός αυτού, εισάγαμε στο στοιχείο NP_SVPWM, στο οποίο καλούνται όλα τα στοιχεία της σχεδίασης, ως εξόδους τέσσερα σήματα στα οποία αναμένουμε να λάβουμε τους επιθυμητούς παλμούς χειρισμού των διακοπτών.

Τέλος προσθέσαμε ένα μικρό τμήμα κώδικα το οποίο επιτρέπει την αγωγή ή την αποκοπή των εκάστοτε διακοπτών για το ανάλογο χρονικό διάστημα το οποίο προέκυψε στο διάνυσμα t_j. Αυτό που ουσιαστικά πραγματοποιούμε είναι να εφαρμόζουμε τις λογικές τιμές που προκύπτουν από το διάνυσμα V_d για τα κατάλληλα χρονικά διαστήματα. Επίσης επειδή θέλουμε σε κάθε αλλαγή να μεταβάλλεται η κατάσταση ενός μόνο διακόπτη, ώστε να μειώνονται οι απώλειες μεταγωγής, αλλά και επειδή γνωρίζουμε ο πρώτος και ο τρίτος συνδυασμός που προκύπτουν είναι οι μηδενικές ακολουθίες (V_{d1} = [0,0], V_{d3} = [1,1]) υλοποιούμε την παρακάτω ακολουθία αγωγής:

1) V_{d1} = [0,0] για χρόνο $t_0/2$.

2) V_{d2} το οποίο προκύπτει ανάλογα με το σήμα αναφοράς [0,1] ή [1,0].

3) V_{d3} = [1,1] για χρόνο $t_1/2$.

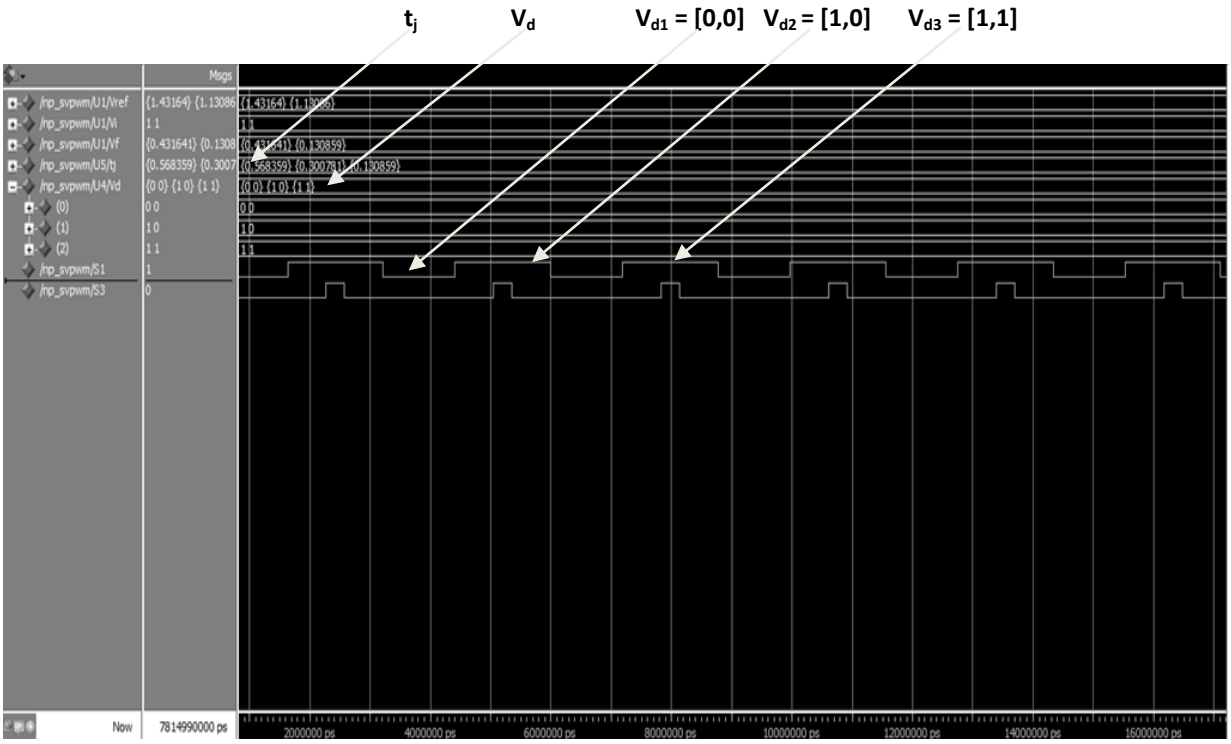
4) V_{d3} = [1,1] για χρόνο $t_1/2$.

5) V_{d2} το οποίο προκύπτει ανάλογα με το σήμα αναφοράς [0,1] ή [1,0].

6) V_{d1} = [0,0] για χρόνο $t_0/2$.

Από τα παραπάνω γίνεται λοιπόν προφανές πως κάθε φορά συντελείται μία μόνο αλλαγή κατάστασης διακόπτη. Επίσης να τονίσουμε πως παρατηρούμε δύο τιμές να εμφανίζονται για των χειρισμό των τεσσάρων διακοπτών και όχι τέσσερεις. Αυτό συμβαίνει καθώς με τον αλγόριθμο που υλοποιούμε υπολογίζουμε μόνο τους παλμούς των διακοπτών του άνω σκέλους του αναστροφέα του σχήματος 3.1(S1,S3) καθώς οι αντίστοιχοι του κάτω σκέλους(S2,S4) είναι συμπληρωματικοί αυτών.

Όταν προσομοιώσαμε τον κώδικα με τις παραπάνω αλλαγές με το εργαλείο προσομοίωσης Modelsim Altera 10.0c(Quartus II 11.1) Starter Edition πήραμε τα παρακάτω αποτελέσματα:



Σχήμα 4.16: Παλμοί διακοπτών S1,S3 του μονοφασικού αναστροφέα με την τεχνική SVPWM μέσω του εργαλείου προσομοίωσης Modelsim

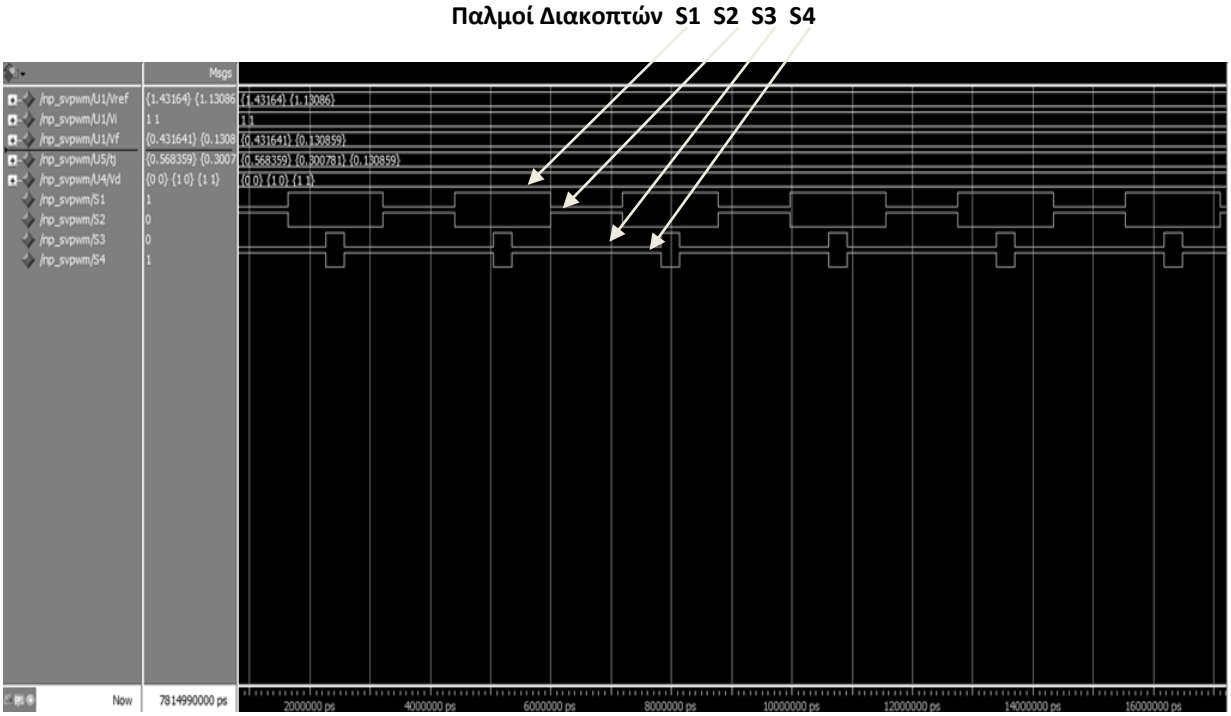
Τα αποτελέσματα που προκύπτουν από την παραπάνω προσομοίωση παρουσιάζονται παρακάτω:

t_j = [0.568359, 0.300781, 0.130859]

V_d = {0,0} {1,0} {1,1}

Παρατηρώντας το παραπάνω σχήμα βλέπουμε πως όντως ο διακοπτικός συνδυασμός {0,0} διαρκεί περισσότερο από τους άλλους δύο, ο συνδυασμός {1,0} έρχεται δεύτερος σε διάρκεια, ενώ ο {1,1} τρίτος. Αυτό είναι σε απόλυτη αντιστοιχία με τους χρόνους που προκύπτουν από το διάνυσμα t_j. Επίσης παρατηρούμε πως όπως αναμέναμε κάθε φορά που αλλάζει ο διακοπτικός συνδυασμός υπάρχει μόνο μία αλλαγή κατάστασης διακόπτη.

Τέλος για μεγαλύτερη πληρότητα παρουσιάζουμε και ένα γενικό στιγμιότυπο της παραπάνω προσομοίωσης στο οποίο έχουμε προσθέσει το σύνολο και των 4 παλμών των διακοπτών(S1,S2,S3,S4).



Σχήμα 4.17: Παλμοί διακοπτών S1,S2,S3,S4 της τεχνικής SVPWM για μονοφασικό αναστροφέα μέσω του εργαλείου προσομοίωσης

Από το παραπάνω σχήμα γίνεται εμφανής η συμπληρωματικότητα που παρουσιάζουν οι παλμοί S2 και S4 προς τους νωρίτερα υπολογισμένους S1 και S3 αντίστοιχα.

4.4.2 Αναφορά χρησιμοποίησης συσκευής

Μετά τη σύνθεση της σχεδίασής μας την οποία κάναμε μέσα από το εργαλείο ISE Design Suite της Xilinx προέκυψε μία αναφορά με τα λογικά μπλοκ τα οποία θα χρησιμοποιηθούν για την υλοποίηση στο FPGA. Τα στοιχεία της αναφορά αυτής παρουσιάζονται παρακάτω:

Slice Logic Utilization	Used	Available	Utilization
Number of Slice Registers	288	18,224	1%
Number used as Flip Flops	288		
Number used as Latches	0		
Number used as Latch-thrus	0		
Number used as AND/OR logics	0		
Number of Slice LUTs	509	9,112	5%
Number used as logic	491	9,112	5%
Number using O6 output only	166		
Number using O5 output only	219		
Number using O5 and O6	106		
Number used as ROM	0		
Number used as Memory	0	2,176	0%
Number used exclusively as route-thrus	18		

Number with same-slice register load	5		
Number with same-slice carry load	13		
Number with other load	0		
Number of occupied Slices	182	2,278	7%
Number of MUXCYs used	368	4,556	8%
Number of LUT Flip Flop pairs used	531		
Number with an unused Flip Flop	255	531	48%
Number with an unused LUT	22	531	4%
Number of fully used LUT-FF pairs	254	531	47%
Number of unique control sets	15		
Number of slice register sites lost	40	18,224	1%
to control set restrictions			
Number of bonded IOBs	72	232	31%
Number of LOCed IOBs	6	72	8%
Number of RAMB16BWERs	0	32	0%
Number of RAMB8BWERs	0	64	0%
Number of BUFIO2/BUFIO2_2CLKs	0	32	0%
Number of BUFIO2FB/BUFIO2FB_2CLKs	0	32	0%
Number of BUFG/BUFGMUXs	1	16	6%
Number used as BUFGs	1		
Number used as BUFGMUX	0		
Number of DCM/DCM_CLKGENs	0	4	0%
Number of ILOGIC2/ISERDES2s	0	248	0%
Number of IODELAY2/IODRP2/IODRP2_MCBs	0	248	0%
Number of OLOGIC2/OSERDES2s	0	248	0%
Number of BSCANs	0	4	0%
Number of BUFHs	0	128	0%
Number of BUFPLLs	0	8	0%
Number of BUFPLL_MCBs	0	4	0%
Number of DSP48A1s	4	32	12%
Number of ICAPs	0	1	0%
Number of MCBs	0	2	0%
Number of PCILOGICSEs	0	2	0%
Number of PLL_ADVs	0	2	0%
Number of PMVs	0	1	0%
Number of STARTUPs	0	1	0%
Number of SUSPEND_SYNCs	0	1	0%
Average Fanout of Non-Clock Nets	3.2		

Πίνακας 4.1: Χρησιμοποιούμενοι πόροι της συσκευής για την υλοποίηση της διφασικής SVPWM τεχνικής

4.4.3 Αποτελέσματα πειραματικής διαδικασίας

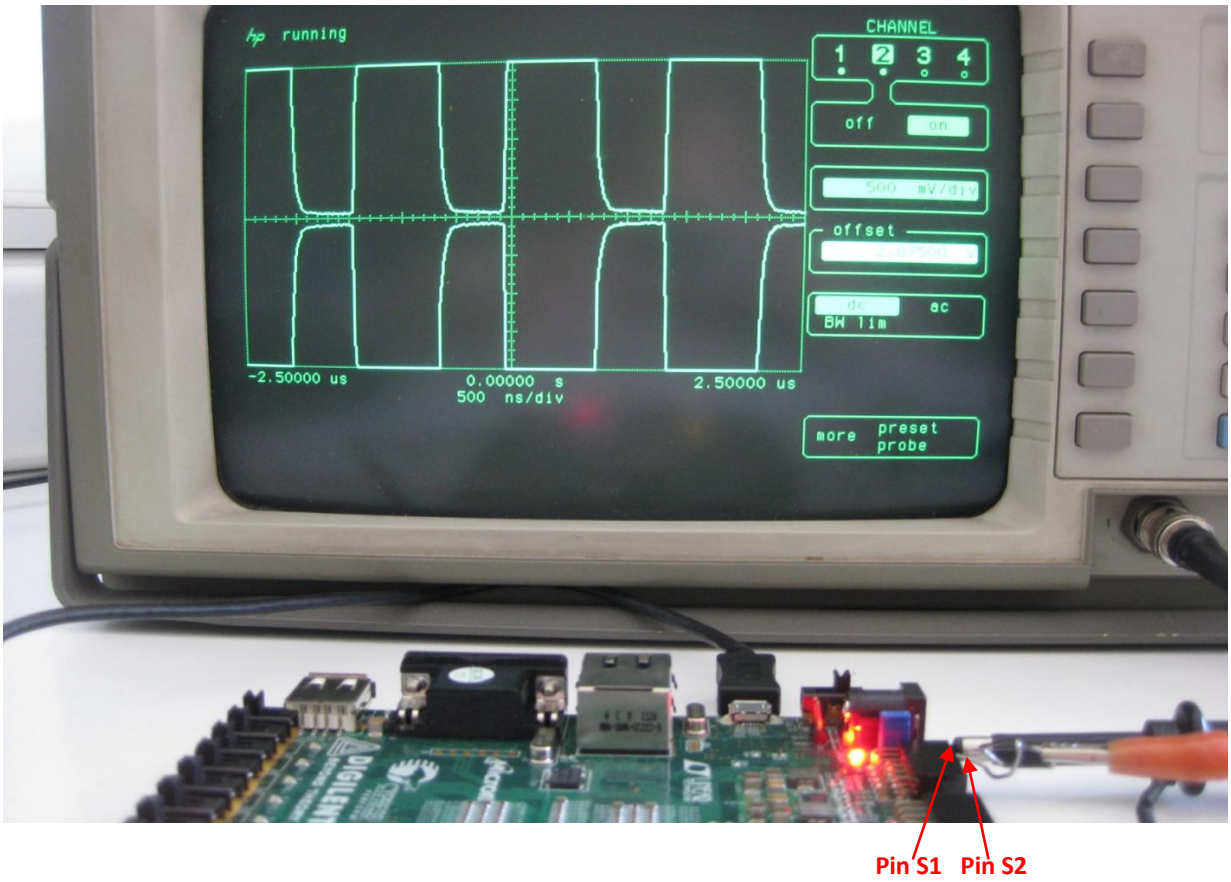
Στο σημείο αυτό, αφού προσομοιώσαμε τον κώδικα που υλοποιεί την SVPWM τεχνική είμαστε σε θέση να ‘’φορτώσουμε’’ τον κώδικα αυτό στο FPGA και να παρατηρήσουμε στον παλμογράφο αν όντως τα αποτελέσματα που είδαμε στην προσομοίωση, ισχύουν και στην πράξη. Για το σκοπό αυτό χρησιμοποιήσαμε το FPGA Spartan 6 XC6LX16-CS324 το οποίο μας δίνει την δυνατότητα να χρησιμοποιήσουμε και αριθμούς κινητής υποδιαστολής στη σχεδίασή μας. Για να υλοποιήσουμε τη σχεδίαση στο FPGA ακολουθήσαμε όλα τα βήματα που περιγράψαμε στο $1^{ο}$ κεφάλαιο και συνδέσαμε μέσω του αρχείου περιορισμού τα σήματα που υπολογίζουν τους παλμούς των τεσσάρων διακοπτών(S1,S2,S3,S4) με τέσσερεις ακροδέκτες του FPGA όπως και στην περίπτωση της SPWM τεχνικής.

Μετά την παραγωγή του αρχείου προγραμματισμού της συσκευής ‘’τρέξαμε’’ την εφαρμογή μας και προέκυψαν στον παλμογράφο τα αποτελέσματα , τα οποία αφορούν στους παλμούς οδήγησης των ημιαγωγικών διακοπτών.

Όπως και στην τεχνική SPWM παρουσιάζουμε δύο στιγμιότυπα τα οποία αναφέρονται στους παλμούς οδήγησης των ζευγών διακοπτών S1-S2 και S3-S4. Τα αποτελέσματα των παλμών αυτών τα οποία παρατηρήθηκαν στον παλμογράφο παρουσιάζονται παρακάτω:

- **Ζεύγος S1-S2:**

Τα αποτελέσματα που προέκυψαν για το ζεύγος αυτό και τα οποία παρατηρήθηκαν με τη βοήθεια του παλμογράφου παρουσιάζονται στο παρακάτω σχήμα:

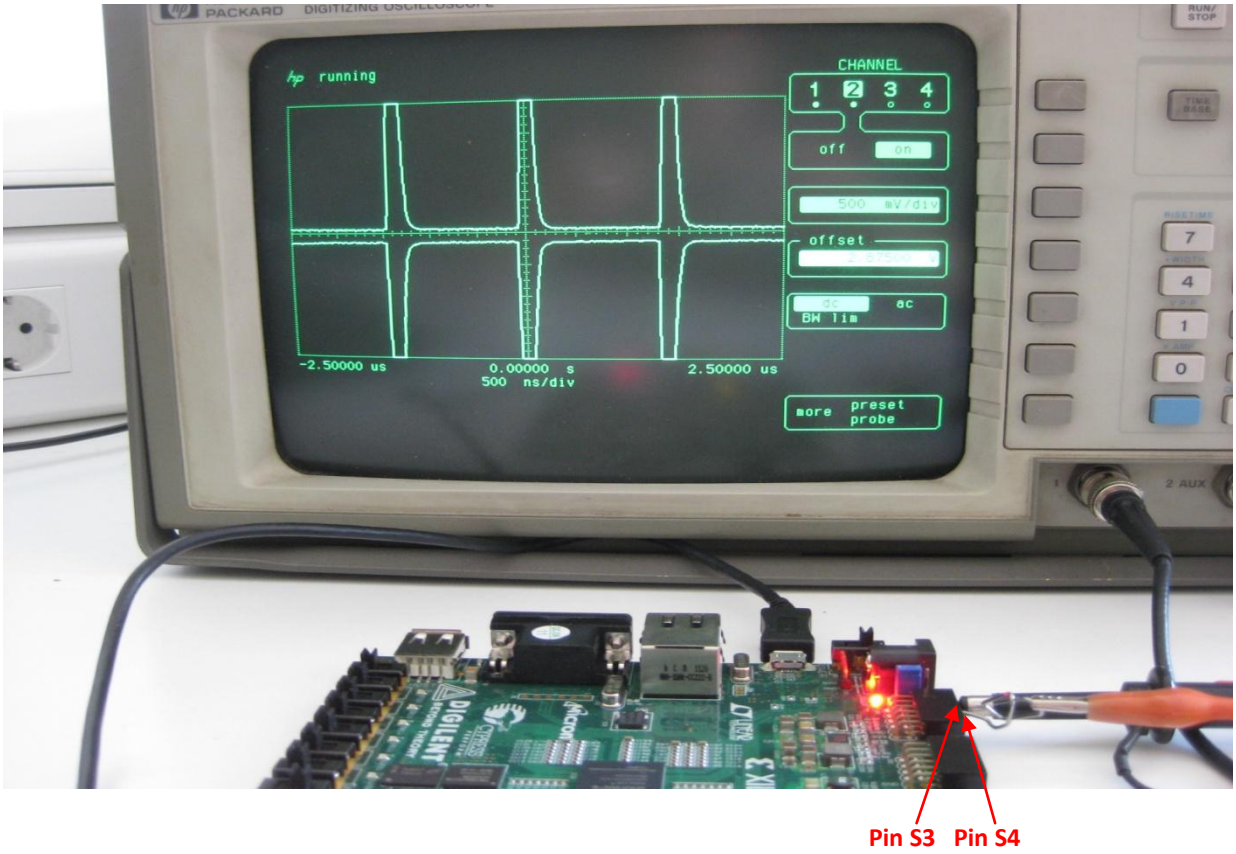


Σχήμα 4.18: Απεικόνιση στον παλμογράφο του συμπληρωματικού ζεύγους παλμών S1-S2 της SVPWM τεχνικής

Παρατηρούμε πως υπάρχει απόλυτη συμφωνία με τα αναμενόμενα αποτελέσματα, τα οποία προέκυψαν στην προσομοίωση, καθώς από τη μία το διάστημα αγωγής του παλμού γα το χειρισμό του διακόπτη S1 είναι λίγο μεγαλύτερο χρονικά από το αντίστοιχο για το διακόπτη S2 και από την άλλη παρατηρούμε τη συμπληρωματικότητα που παρουσιάζουν οι δύο αυτοί παλμοί, πράγμα απαραίτητο για την αποφυγή βραχυκυκλωμάτων.

- **Ζεύγος S3-S4:**

Τα αποτελέσματα που προέκυψαν για το ζεύγος αυτό και τα οποία παρατηρήθηκαν με τη βοήθεια του παλμογράφου παρουσιάζονται στο παρακάτω σχήμα:

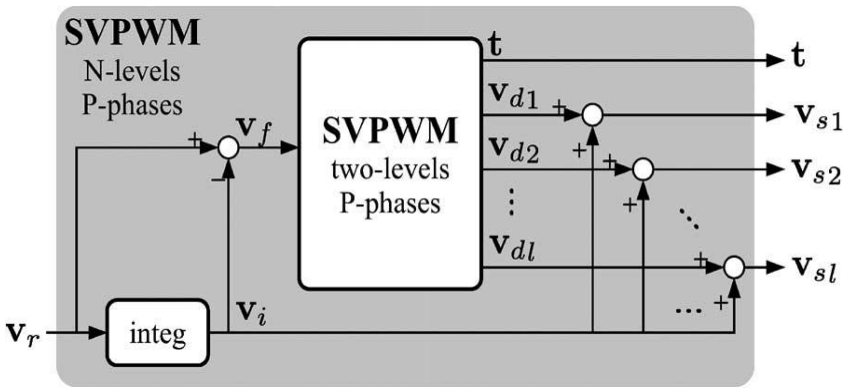


Σχήμα 4.19: Απεικόνιση στον παλμογράφο του συμπληρωματικού ζεύγους παλμών S3-S4 της SVPWM τεχνικής

Και εδώ, όπως και στο παραπάνω σχήμα, παρατηρούμε πως οι παλμοί που προκύπτουν για το χειρισμό του ζεύγους S3-S4 είναι οι αναμενόμενοι. Βλέπουμε πως το διάστημα αγωγής για το χειρισμό του διακόπτη S3 είναι αισθητά μικρότερο χρονικά από το αντίστοιχο για το χειρισμό του S4 αλλά επίσης παρατηρούμε και τη συμπληρωματικότητα που παρουσιάζουν οι δύο αυτοί παλμοί.

4.5 Υλοποίηση της SVPWM με το Matlab Simulink

Για να ολοκληρώσουμε την αναφορά μας στην τεχνική SVPWM την οποία μελετήσαμε στο παρόν κεφάλαιο θα την υλοποιήσουμε και με εργαλείο Simulink του Matlab. Στη λογική της υλοποίησης αυτής δεν αλλάζει τίποτα από όσα έχουμε ήδη αναφέρει και αφορούν στον αλγόριθμο της σχεδίασης. Το ολικό μοντέλο αποτελείται από δύο τμήματα όπου το ένα εσωκλείεται στο άλλο κάτι το οποίο γίνεται ευκολότερα κατανοητό από το παρακάτω σχήμα:



Σχήμα 4.20: Μπλοκ Διάγραμμα της Πολυφασικής Πολυεπίπεδης SVPWM

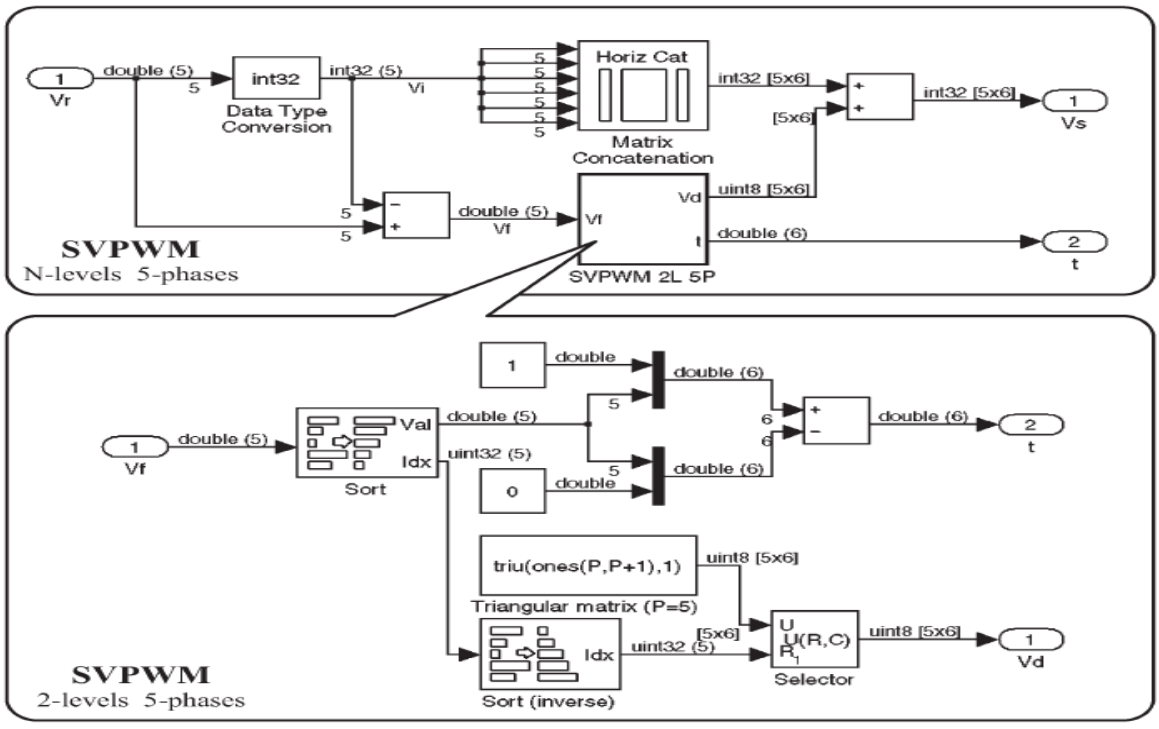
Με βάση το παραπάνω σχήμα καταλαβαίνουμε πως τα δύο τμήματα από τα οποία αποτελείται το μοντέλο το οποίο υλοποιούμε είναι τα εξής:

1) SVPWM Τεχνική N Επιπέδων και P Φάσεων.

2) SVPWM Τεχνική 2 Επιπέδων και P Φάσεων.

Παρατηρούμε πως η SVPWM των 2 επιπέδων και P φάσεων αποτελεί συστατικό στοιχείο της SVPWM των N επιπέδων και P φάσεων.

Το Simulink μοντέλο για την υλοποίηση της SVPWM τεχνικής παρουσιάζεται παρακάτω:



Σχήμα 4.21: Simulink μοντέλο της υλοποίησης στο υλικό του SVPWM αλγορίθμου

ΚΕΦΑΛΑΙΟ 5^{ο}

ΣΥΝΟΨΗ , ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΗ ΕΡΓΑΣΙΑ

5.1 Σύνοψη και συμπεράσματα

Στην παρούσα διπλωματική εργασία ασχοληθήκαμε με την υλοποίηση, μέσω της γλώσσας περιγραφής υλικού VHDL αλλά και την χρησιμοποίηση FPGA, τριών τεχνικών που εφαρμόζονται ευρέως στα ηλεκτρονικά ισχύος. Αυτές κατά σειρά είναι οι παρακάτω:

1) PWM(Pulse Width Modulation) Τεχνική.

2) SPWM(Sinusoidal Pulse Width Modulation) Τεχνική.

3) SVPWM(Space Vector Pulse Width Modulation) Τεχνική.

Η PWM τεχνική ουσιαστικά ελέγχει την αγωγή και την αποκοπή ενός διακόπτη ανάλογα με το duty cycle το οποίο δίνεται από τον χρήστη. Η υλοποίηση αυτή βασίζεται στη χρήση ενός ή δύο μετρητών οι οποίοι συγκρινόμενοι με το επιθυμητό duty cycle δίνουν το κατάλληλο αποτέλεσμα στην έξοδο. Στην παρούσα εργασία η υλοποίηση αυτή έγινε με ένα τρόπο. Τα θεωρητικώς αναμενόμενα αποτελέσματα τα επαληθεύσαμε μέσω προσομοίωσης αλλά και μέσω απεικόνισης στον παλμογράφο αφού πρώτα είχαμε υλοποιήσει τη σχεδίαση μας στο FPGA Spartan 3E.

Η δεύτερη τεχνική την οποία υλοποιήσαμε είναι η ημιτονοειδής διαμόρφωση εύρους παλμών(SPWM). Στην τεχνική αυτή οι επιθυμητοί παλμοί στην έξοδο προκύπτουν από την σύγκριση ενός ημιτονοειδούς με ένα τριγωνικό σήμα. Το τριγωνικό σήμα υλοποιείται εύκολα με τη χρήση ενός μετρητή και λαμβάνοντας υπ' όψιν το ρολόι της σχεδίασης. Το κρίσιμο σημείο της όλης σχεδίασης έχει να κάνει με την υλοποίηση του ημιτονοειδούς σήματος.

Στην παρούσα εργασία παρουσιάσαμε δύο τρόπους για την υλοποίηση αυτή. Στον πρώτο τρόπο χρησιμοποιήσαμε μία μνήμη ROM με 1000 δείγματα ημιτόνου τα οποία είχαν υπολογιστεί προηγουμένως μέσω άλλου προγράμματος. Το πλεονέκτημα της μεθόδου αυτής έγκειται στο γεγονός πως δεν χρειάζεται να υπολογίζουμε τιμές ημιτόνου κατά την υλοποίηση του κώδικά μας καθώς αυτές είναι ήδη υπολογισμένες. Κατά συνέπεια μειώνονται οι απαιτούμενοι πόροι για την πραγματοποίηση της σχεδίασης αυτής. Το μειονέκτημα όμως είναι πως δεν υπολογίζουμε με τόση ακρίβεια τα σημεία τομής του ημιτονοειδούς σήματος με το τριγωνικό καθώς τα σημεία του ημιτονοειδούς σήματος που χρησιμοποιούμε είναι σταθερά και υπολογισμένα. Στον δεύτερο τρόπο υλοποίησης του ημιτονοειδούς σήματος δεν χρησιμοποιήσαμε μνήμη ROM για τα δείγματα αλλά τα υπολογίζαμε δυναμικά όποτε αυτό ήταν απαραίτητο. Αυτό έγινε με χρήση του αλγορίθμου CORDIC μέσω του Core Generator που προσφέρει το εργαλείο ISE Design Suite της Xilinx. Για το πρώτο μισό της πρώτης περιόδου του ημιτόνου, κάθε φορά που παρατηρούσαμε τομή μεταξύ του ημιτονοειδούς και του τριγωνικού σήματος αποθηκεύαμε την τιμή αυτή στον ανάλογο πίνακα(ανάλογα με τον αν η τιμή αφορούσε στο 1^{ο} ή το 2^{ο} ημίτονο). Μετά το πέρας του πρώτου μισού της πρώτης περιόδου του ημιτόνου

σταματούσαμε τον υπολογισμό των τιμών μέσω του αλγορίθμου CORDIC και χρησιμοποιούσαμε πλέον τις αποθηκευμένες στους δύο πίνακες τιμές.

Για την επαλήθευση των θεωρητικώς αναμενομένων αποτελεσμάτων πραγματοποιήσαμε δύο προσομοιώσεις. Η πρώτη έγινε με το εργαλείο προσομοίωσης PSim και συμβάλει κυρίως στην ανάδειξη των αναμενόμενων αποτελεσμάτων. Αξίζει να σημειώσουμε πως μέσω του εργαλείου αυτού λαμβάνουμε μία πιο ακριβή εικόνα για τα αναμενόμενα αποτελέσματα, καθώς τα στοιχεία τα οποία χρησιμοποιούμε είναι μη ιδανικά. Η δεύτερη πραγματοποιήθηκε με το εργαλείο προσομοίωσης Modelsim Altera 10.0c(Quartus II 11.1) Starter Edition και τα ενδεικτικά στιγμιότυπα των δύο μεθόδων που παρουσιάστηκαν, επαλήθευσαν τα θεωρητικώς αναμενόμενα αποτελέσματα αλλά και τα αποτελέσματα που προέκυψαν μέσω του PSim. Εκτός αυτών υλοποιήσαμε τις σχεδιάσεις μας και στο FPGA Spartan 6 XC6LX16-CS324 μέσω του οποίου μπορέσαμε να τα απεικονίσουμε στον παλμογράφο και να παρατηρήσουμε και εκεί την επιθυμητή συμπεριφορά. Να τονίσουμε και εδώ πως δεν χρησιμοποιήσαμε το Spartan 3E αλλά το Spartan 6 καθώς το πρώτο δεν υποστηρίζει την βιβλιοθήκη ieee_proposed μέσω της οποίας μας δίνεται η δυνατότητα να χρησιμοποιήσουμε αριθμούς κινητής αλλά και σταθερής υποδιαστολής όπως στις προκείμενες σχεδιάσεις.

Η τρίτη και τελευταία τεχνική την οποία υλοποιήσαμε είναι η διαμόρφωση εύρους παλμών με χωρικά διανύσματα(SVPWM). Ιδιαίτερο χαρακτηριστικό της τεχνικής αυτής είναι πως είναι παραμετρική, πολυφασική και πολυεπίπεδη. Δίνεται δηλαδή η δυνατότητα στο χρήστη μεταβάλλοντας τις τιμές δύο μεταβλητών να αλλάξει τον αριθμό των επιπέδων αλλά και των φάσεων της σχεδίασης κάτι που δίνει ιδιαίτερη ευελιξία και προσαρμοστικότητα στον κώδικά μας. Στόχος της τεχνικής αυτής είναι η όσο το δυνατόν καλύτερη προσέγγιση του σήματος αναφοράς το οποίο έρχεται ως είσοδος της σχεδίασής μας. Για να το πετύχουμε αυτό ακολουθούμε έξι βήματα τα οποία υλοποιούνται από έξι διαφορετικά κυκλώματα το καθένα από τα οποία υλοποιεί μία συγκεκριμένη λειτουργία. Το κάθε ένα από αυτά τα κυκλώματα χρειάζεται συνήθως για την πραγμάτωση της λειτουργίας του, αποτελέσματα που προκύπτουν από κάποιο άλλο κύκλωμα. Γι αυτόν το λόγο χρησιμοποιούμε σήματα που ενημερώνουν ανάλογα το κάθε κύκλωμα για το αν τα απαραίτητα για τη λειτουργία του στοιχεία είναι έτοιμα ή όχι.

Ουσιαστικά ο σκοπός της τεχνικής είναι να πετύχουμε τον κατάλληλο συνδυασμό αγωγής, και για το κατάλληλο χρονικό διάστημα, των ημιαγωγικών διακοπτών του αναστροφέα που θα χρησιμοποιείται ανάλογα πάντα με την εφαρμογή ώστε το σήμα που θα λαμβάνουμε στην έξοδο να προσεγγίζει όσον το δυνατόν καλύτερα το σήμα της αναφοράς που δεχόμαστε στην είσοδο. Συνοπτικά αυτό επιτυγχάνεται με την ακόλουθη διαδικασία: Αρχικά αποσυνθέτουμε το σήμα της αναφοράς σε ακέραιο και κλασματικό μέρος. Το κλασματικό μέρος το διατάσουμε με τη χρήση του αλγορίθμου της φυσαλίδας(bubblesort) σε φθίνουσα σειρά και μέσω του ταξινομημένου πλέον διανύσματος εξάγουμε τους χρόνους αγωγής του εκάστοτε διακοπτικού συνδυασμού. Από τις μετατοπίσεις που έλαβαν χώρα κατά τη δημιουργία του κατά φθίνουσα σειρά ταξινομημένου διανύσματος δημιουργούμε έναν πίνακα από τον οποίο λαμβάνουμε τα μετατοπισμένα πλέον διανύσματα μεταγωγής. Τέλος, με τη χρήση των μετατοπισμένων αυτών διανυσμάτων μεταγωγής αλλά και του ακεραίου μέρους του σήματος αναφοράς παράγουμε τα τελικά διακοπτικά διανύσματα που προσδιορίζουν την τιμή του ακεραίου βήματος της κανονικοποιημένης τάσης εξόδου της εκάστοτε φάσης. Στο σημείο αυτό αξίζει να σημειώσουμε πως η τεχνική αυτή είναι υλοποιημένη με τέτοιο τρόπο ώστε να επιτυγχάνεται το

επιθυμητό αποτέλεσμα με τις λιγότερες δυνατές μεταβολές των διακοπτών, επιτυγχάνοντας με αυτό τον τρόπο την ελαχιστοποίηση των διακοπτικών απωλειών.

Μετά την υλοποίηση των τεχνικών SPWM και SVPWM παρατηρήσαμε τα εξής:

- Η τεχνική SPWM που υλοποιείται με τη χρήση της μνήμης ROM χρησιμοποιεί λιγότερους πόρους του FPGA αλλά υστερεί σε ακρίβεια σε σχέση με αυτή που χρησιμοποιεί δυναμικό υπολογισμό δειγμάτων. Η επιλογή της μίας ή της άλλης τεχνικής έχει να κάνει με το τι θεωρεί σημαντικότερο για την εφαρμογή του ο χρήστης.
- Η τεχνική SVPWM παρότι δεσμεύει περισσότερους πόρους του FPGA σε σχέση με τις άλλες δύο τεχνικές είναι αισθητά αποδοτικότερη και μας παρέχει εκτός των άλλων τη δυνατότητα να μεταβάλλουμε τον αριθμό των φάσεων αλλά και των επιπέδων της σχεδίασής μας ανάλογα με τις απαιτήσεις της εκάστοτε εφαρμογής.

5.2 Μελλοντική εργασία

Στην παρούσα διπλωματική εργασία ασχοληθήκαμε με την σχέση και την εφαρμογή που μπορούν να έχουν τα FPGA στα ηλεκτρονικά ισχύος. Υλοποιήσαμε τρεις τεχνικές διαμόρφωσης εύρους παλμών οι οποίες εφαρμόζονται ευρέως στα ηλεκτρονικά ισχύος. Τις τεχνικές αυτές τις υλοποιήσαμε με την χρήση της γλώσσας περιγραφής υλικού VHDL και τα αναμενόμενα θεωρητικά αποτελέσματα τα επαληθεύσαμε με τη χρήση του εργαλείου προσομοίωσης Modelsim Altera 10.0c(Quartus II 11.1) Starter Edition αλλά και με την πραγματική τους υλοποίηση στα FPGA Spartan 3E και Spartan 6 XC6LX16-CS324 και απεικόνισή τους στον παλμογράφο.

Μελλοντικό λοιπόν στόχο αποτελεί η εφαρμογή των τεχνικών αυτών με την χρήση FPGA σε μονοφασικούς, τριφασικούς αλλά και πολυφασικούς και πολυεπίπεδους αναστροφείς ώστε να επαληθεύσουμε και στην πράξη τα όσα υλοποιήσαμε στην εργασία αυτή. Επίσης θα θέλαμε να συγκρίνουμε τα αποτελέσματα που θα λάβουμε με την χρήση του FPGA με τα αντίστοιχα που θα προκύψουν από την χρησιμοποίηση ενός επεξεργαστή ψηφιακού σήματος(DSP) και να αναδείξουμε τα τυχόν πλεονεκτήματα που θα προκύψουν. Να αναδείξουμε δηλαδή λόγους για τους οποίους θα ήταν επικερδής και ωφέλιμη η αντικατάσταση σε ορισμένες εφαρμογές των χρησιμοποιούμενων DSP από FPGA. Τέλος επέκταση της εργασίας αυτής θα ήταν και η υλοποίηση και άλλων τεχνικών που αφορούν στα ηλεκτρονικά ισχύος και έχουν να κάνουν κυρίως με τον έλεγχο κινητήρων και άλλων μηχανών.

ΠΑΡΑΡΤΗΜΑ

VHDL Κώδικας της SPWM τεχνικής με χρήση ROM

```
---------------------------------------- Δήλωση των απαραίτητων Βιβλιοθηκών και Πακέτων ----------------------------------------------------------

library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;

use IEEE.NUMERIC_STD.all;


library ieee_proposed;

use ieee_proposed.fixed_float_types.all;

use ieee_proposed.fixed_pkg.all;


library std;

use std.standard.all;

-------------------------------------------------------------------------------------------------------------------------------------------------------------

---------------------------------------- Δήλωση της Ενότητας που θα χρησιμοποιήσουμε ----------------------------------------------------------

entity ROM_1000_Deadband_Synthesis is

GENERIC (ma: INTEGER := 70); -- ma%

Port (CLK: in STD_LOGIC;

    reset: in STD_LOGIC;

    S1,S2,S3,S4: inout STD_LOGIC);

end ROM_1000_Deadband_Synthesis;

-------------------------------------------------------------------------------------------------------------------------------------------------------------

---------------------------------------- Δήλωση της Αρχιτεκτονικής που θα χρησιμοποιήσουμε ----------------------------------------------------------

Architecture behavioural of ROM_1000_Deadband_Synthesis is

signal Ar: sfixed(12 downto -2);

signal Ac: INTEGER;

signal Ar_reg: INTEGER;

TYPE vector_array is ARRAY (0 TO 999) of sfixed(1 downto -18);
```

---------- Πίνακας με 1000 Δείγματα Ημιτόνου ----------

CONSTANT sine_values :vector_array:=
("0000000000000000000","0000000000011001110000","0000000000110011011111","0000000001001101101110","0000000011001110111110","0
000000100000000010110","0000000010011010011010","0000000101101000010000","0000000111001101111100","0000000111100111100011","0000
0010000000101010","0000001000110110000 1","0000001001101001100 0","0000001010011100111 1","0000001011010000011 0","0000001
1000000111101","0000001100110111001 1","0000001101101010101 0","0000001110011100000 0","0000001111010001011 0","0000010000
0001001100","0000010000111000001 0","0000010001101011100 0","0000010010011101101 1","0000010011010010001 0","0000010100000
1011000","0000010100111000110 0","0000010101101100000 1","0000010110011111011 0","0000010111010010101 0","0000011000000101
1110","0000011000111001001 0","0000011001101100010 1","0000011010011111100 0","0000011011010010101 1","0000011100000010111
0","0000011100111001000 0","0000011101101100001 1","0000011110011111010 0","0000011111010010011 1","0000100000001010111",
"0000100000111000110 0","0000100001101011100 1","0000100010011101001 1","0000100011000001001000","0000100100111011110000","00001001001101011110","0000100101101010011 0","0000100111001110101 1","0000101000000110000","0000101001101100111 0","0000101011001100011 0","0000101100000001110","0000101100111001100111","0000101101100110001110","0000101110010111100 1","0000101111111111100001","0000110000100001010","0000110001100011 0010","0000110010010101101 0","0000110011000100000 1","0000110011110100111 1","0000110100100111001110","0000110101010111110011","0000110110100100100 0","0000110111010001100 1","0000111000100101011 0","0000111010000110111 1","0000111011111111100 0","0000111100101000010","0000111101010011001 0","0000111110000111010","0000111111010101000 0","0001000000111000110","0001000001001111111111","0001000010110001101 0","0001000011100011010 1","0001000100101000110 0","0001000101110000000 1","0001000111010100110 0","0001001000011111011100","0001001001101110000 1","0001001010100000010 1","0001001011010011001 1","0001001100000010110 0","0001001101100111001110","0001001110010100111 1","0001001111100100101 1","0001001111110111111 0","0001010000100001100","0001010001011001100 1","0001010010010100101 1","0001010010111010000 1","0001010011101011110 0","0001010100011100111 0","0001010101001110000 0","0001010110011111111 0","0001010111010111110111","0001011000000111101010","0001100001001101100 1","0001100001111101111 0","0001100010101101011 0","0001100011101001000 0","0001100100100001100 0","0001100101110100000 1","0001100111001001001 0","0001101000000110010100","0001101000111101111110","0001101001100111000101","0001101010000101010010","0001101010110011111 1","0001101011100010101011","0001101100010001011 0","0001101101110110110 0","0001101110111100000 0","0001101111100101100 0","0001110000111010100","0001110001111111111111","0001110010000110101 0","0001110011010111110111","0001110011111011101111","0001110100000111101010","0001110001001110011 0","0001110010111100101 1","0001110011100110100001","0001110011100100101 0","0001100111111100000101","0001110100100110111010","0001110101010000101 0","0001110101101011011110","0001110111011100100 0","0001110111101110100 1","0001110111111111110000","0001111001011111000","0001111010111110111","0001111011100111011 0","0001111100000111101010","0001111000100111001 0","0001111001110111101 0","0001111010011101010 1","0001111011100110100 0","0001111100100100000 1","0001111101010011100 1","0001111110000111010 1","0001111111111111111 1","0010000000011101110","0010000000111011011 0","0010000001110100 1","0010000010000000110","0010000011101100 1001","0010000100011000 1011","0010000101110000101 0","0010000110000110100 1","0010000111100111000 1","0010000111111111000 0","0010001000010111110","0010001001011100 1","0010001010000011 1000","0010001011100001010","0010001100111000 0111","0010001101000000 1","0010001110000111111000","0010010000100101000 0","0010010001110100 1","0010010010100001100","0010010011101100110","0010010101000011100","0010010110011010000","0010010111100111110","0010010000
111111000","0010010010010110000","0010010011101100110","0010010101000011100","0010010110011010000","0010010111100111110","0010010000
000010","0010011001000110011","0010011010011100011","0010011011110010001","0010011101000111110","0010011110011101101
001","0010011111110010011","0010100000100011 1011","0010100010011100010","0010100011110001000","0010100101000101100
","0010100110011001110","0010100111101101111","0010101001000001111","0010101010010101101","0010101011101001001","0010101100111100100","0010101110000111110","0010110000110101011","0010110101111101011","0010111000100010001","0010111001101100110","0010111010100101100","0010111011010100","0011000000111101010","0011000001001111111","0011000011111011110","0011000101011010110","0011000110111001110","0011001000011000100","0011001001110111010","0011001011010101110","0011001100110100001","0011001110010010100","0011001111110000101","0011010001001110101","0011010010101100100","0011010100000101 0010","0011010101100111111","0011010111000101011","0011011000100010110","0011011001111111111","0011011010110111010000","0011011100111010000","0011011110010110110","0011011111110011011","0011100000100111111 1","0011100001010110010 0","0011100010100000101 0","0011100011100100001","0011100100111010101 1","0011100101100111010","0011100110000111101 0","0011100111010111110 0","0011101000000111101010","0011101000100111001 0","0011101001110111101 0","0011101010011101010 1","0011101011100110100 0","0011101100100100000 1","0011101101010011100 1","0011101110000111010 1","0011101111111111111 1","0011110000100001100","0011110001011001100 1","0011110010010100101 1","0011110010111010000 1","0011110011101011110 0","0011110100011100111 0","0011110101001110000 0","0011110110011111111 0","0011110111010111110111","0011111001011111000","0011111010111110111","0011111011100111011 0","0011111100000111101010","0011111000100111001 0","0011111001110111101 0","0011111010011101010 1","0011111011100110100 0","0011111100100100000 1","0011111101010011100 1","0011111110000111010 1","0011111111111111111 1","0100000000011101110","0100000000111011011 0","0100000001110100 1","0100000010000000110","0100000011101100 1001","0100000100011000 1011","0100000101110000101 0","0100000110000110100 1","0100000111100111000 1","0100000111111111000 0","0100001000010111110","0100001001011100 1","0100001010000011 1000","0100001011100001010","0100001100111000 0111","0100001101000000 1","0100001110000111111000","0100010000100101000 0","0100010001110100 1","0100010010100001100","0100010011101100110","0100010101000011100","0100010110011010000","0100010111100111110","0100010000
111111000","0100010010010110000","0100010011101100110","0100010101000011100","0100010110011010000","0100010111100111110","0100010000
000010","0100011001000110011","0100011010011100011","0100011011110010001","0100011101000111110","0100011110011101101
001","0100011111110010011","0100100000100011 1011","0100100010011100010","0100100011110001000","0100100101000101100
","0100100110011001110","0100100111101101111","0100101001000001111","0100101010010101101","0100101011101001001","0100101100111100100","0100101110000111110","0100110000110101011","0100110101111101011","0100111000100010001","0100111001101100110","0100111010100101100","0100111011010100","0101000000111101010","0101000001001111111","0101000011111011110","0101000101011010110","0101000110111001110","0101001000011000100","0101001001110111010","0101001011010101110","0101001100110100001","0101001110010010100","0101100111111100000101","0101110100100110111010","0101110101010000101 0","0101110101101011011110","0101110111011100100 0","0101110111101110100 1","0101110111111111110000","0101111001011111000","0101111010111110111","0101111011100111011 0","0101111100000111101010","0101111000100111001 0","0101111001110111101 0","0101111010011101010 1","0101111011100110100 0","0101111100100100000 1","0101111101010011100 1","0101111110000111010 1","0101111111111111111 1","0110000000011101110","0110000000111011011 0","0110000001110100 1","0110000010000000110","0110000011101100 1001","0110000100011000 1011","0110000101110000101 0","0110000110000110100 1","0110000111100111000 1","0110000111111111000 0","0110001000010111110","0110001001011100 1","0110001010000011 1000","0110001011100001010","0110001100111000 0111","0110001101000000 1","0110001110000111111000","0110010000100101000 0","0110010001110100 1","0110010010100001100","0110010011101100110","0110010101000011100","0110010110011010000","0110010111100111110","0110010000
111111000","0110010010010110000","0110010011101100110","0110010101000011100","0110010110011010000","0110010111100111110","0110010000
000010","0110011001000110011","0110011010011100011","0110011011110010001","0110011101000111110","0110011110011101101
001","0110011111110010011","0110100000100011 1011","0110100010011100010","0110100011110001000","0110100101000101100
","0110100110011001110","0110100111101101111","0110101001000001111","0110101010010101101","0110101011101001001","0110101100111100100","0110101110000111110","0110110000110101011","0110110101111101011","0110111000100010001","0110111001101100110","0110111010100101100","0110111011010100","0111000000111101010","0111000001001111111","0111000011111011110","0111000101011010110","0111000110111001110","0111001000011000100","0111001001110111010","0111001011010101110","0111001100110100001","0111001110010010100","0111100111111100000101","0111110100100110111010","0111110101010000101 0","0111110101101011011110","0111110111011100100 0","0111110111101110100 1","0111110111111111110000","0111111001011111000","0111111010111110111","0111111011100111011 0","0111111100000111101010","0111111000100111001 0","0111111001110111101 0","0111111010011101010 1","0111111011100110100 0","0111111100100100000 1","0111111101010011100 1","0111111110000111010 1","0111111111111111111 1","1000000000011101110","1000000000111011011 0","1000000001110100 1","1000000010000000110","1000000011101100 1001","1000000100011000 1011","1000000101110000101 0","1000000110000110100 1","1000000111100111000 1","1000000111111111000 0","1000001000010111110","1000001001011100 1","1000001010000011 1000","1000001011100001010","1000001100111000 0111","1000001101000000 1","1000001110000111111000","1000010000100101000 0","1000010001110100 1","1000010010100001100","1000010011101100110","1000010101000011100","1000010110011010000","1000010111100111110","1000010000
111111000","1000010010010110000","1000010011101100110","1000010101000011100","1000010110011010000","1000010111100111110","1000010000
000010","1000011001000110011","1000011010011100011","1000011011110010001","1000011101000111110","1000011110011101101
001","1000011111110010011","1000100000100011 1011","1000100010011100010","1000100011110001000","1000100101000101100
","1000100110011001110","1000100111101101111","1000101001000001111","1000101010010101101","1000101011101001001","1000101100111100100","1000101110000111110","1000110000110101011","1000110101111101011","1000111000100010001","1000111001101100110","1000111010100101100","1000111011010100","1001000000111101010","1001000001001111111","1001000011111011110","1001000101011010110","1001000110111001110","1001001000011000100","1001001001110111010","1001001011010101110","1001001100110100001","1001001110010010100","1001100111111100000101","1001110100100110111010","1001110101010000101 0","1001110101101011011110","1001110111011100100 0","1001110111101110100 1","1001110111111111110000","1001111001011111000","1001111010111110111","1001111011100111011 0","1001111100000111101010","1001111000100111001 0","1001111001110111101 0","1001111010011101010 1","1001111011100110100 0","1001111100100100000 1","1001111101010011100 1","1001111110000111010 1","1001111111111111111 1","1010000000011101110","1010000000111011011 0","1010000001110100 1","1010000010000000110","1010000011101100 1001","1010000100011000 1011","1010000101110000101 0","1010000110000110100 1","1010000111100111000 1","1010000111111111000 0","1010001000010111110","1010001001011100 1","1010001010000011 1000","1010001011100001010","1010001100111000 0111","1010001101000000 1","1010001110000111111000","1010010000100101000 0","1010010001110100 1","1010010010100001100","1010010011101100110","1010010101000011100","1010010110011010000","1010010111100111110","1010010000
111111000","1010010010010110000","1010010011101100110","1010010101000011100","1010010110011010000","1010010111100111110","1010010000
000010","1010011001000110011","1010011010011100011","1010011011110010001","1010011101000111110","1010011110011101101
001","1010011111110010011","1010100000100011 1011","1010100010011100010","1010100011110001000","1010100101000101100
","1010100110011001110","1010100111101101111","1010101001000001111","1010101010010101101","1010101011101001001","1010101100111100100","1010101110000111110","1010110000110101011","1010110101111101011","1010111000100010001","1010111001101100110","1010111010100101100","1010111011010100","1011000000111101010","1011000001001111111","1011000011111011110","1011000101011010110","1011000110111001110","1011001000011000100","1011001001110111010","1011001011010101110","1011001100110100001","1011001110010010100","1011100111111100000101","1011110100100110111010","1011110101010000101 0","1011110101101011011110","1011110111011100100 0","1011110111101110100 1","1011110111111111110000","1011111001011111000","1011111010111110111","1011111011100111011 0","1011111100000111101010","1011111000100111001 0","1011111001110111101 0","1011111010011101010 1","1011111011100110100 0","1011111100100100000 1","1011111101010011100 1","1011111110000111010 1","1011111111111111111 1","1100000000011101110","1100000000111011011 0","1100000001110100 1","1100000010000000110","1100000011101100 1001","1100000100011000 1011","1100000101110000101 0","1100000110000110100 1","1100000111100111000 1","1100000111111111000 0","1100001000010111110","1100001001011100 1","1100001010000011 1000","1100001011100001010","1100001100111000 0111","1100001101000000 1","1100001110000111111000","1100010000100101000 0","1100010001110100 1","1100010010100001100","1100010011101100110","1100010101000011100","1100010110011010000","1100010111100111110","1100010000
111111000","1100010010010110000","1100010011101100110","1100010101000011100","1100010110011010000","1100010111100111110","1100010000
000010","1100011001000110011","1100011010011100011","1100011011110010001","1100011101000111110","1100011110011101101
001","1100011111110010011","1100100000100011 1011","1100100010011100010","1100100011110001000","1100100101000101100
","1100100110011001110","1100100111101101111","1100101001000001111","1100101010010101101","1100101011101001001","1100101100111100100","1100101110000111110","1100110000110101011","1100110101111101011","1100111000100010001","1100111001101100110","1100111010100101100","1100111011010100","1101000000111101010","1101000001001111111","1101000011111011110","1101000101011010110","1101000110111001110","1101001000011000100","1101001001110111010","1101001011010101110","1101001100110100001","1101001110010010100","1101100111111100000101","1101110100100110111010","1101110101010000101 0","1101110101101011011110","1101110111011100100 0","1101110111101110100 1","1101110111111111110000","1101111001011111000","1101111010111110111","1101111011100111011 0","1101111100000111101010","1101111000100111001 0","1101111001110111101 0","1101111010011101010 1","1101111011100110100 0","1101111100100100000 1","1101111101010011100 1","1101111110000111010 1","1101111111111111111 1","1110000000011101110","1110000000111011011 0","1110000001110100 1","1110000010000000110","1110000011101100 1001","1110000100011000 1011","1110000101110000101 0","1110000110000110100 1","1110000111100111000 1","1110000111111111000 0","1110001000010111110","1110001001011100 1","1110001010000011 1000","1110001011100001010","1110001100111000 0111","1110001101000000 1","1110001110000111111000","1110010000100101000 0","1110010001110100 1","1110010010100001100","1110010011101100110","1110010101000011100","1110010110011010000","1110010111100111110","1110010000
111111000","1110010010010110000","1110010011101100110","1110010101000011100","1110010110011010000","1110010111100111110","1110010000
000010","1110011001000110011","1110011010011100011","1110011011110010001","1110011101000111110","1110011110011101101
001","1110011111110010011","1110100000100011 1011","1110100010011100010","1110100011110001000","1110100101000101100
","1110100110011001110","1110100111101101111","1110101001000001111","1110101010010101101","1110101011101001001","1110101100111100100","1110101110000111110","1110110000110101011","1110110101111101011","1110111000100010001","1110111001101100110","1110111010100101100","1110111011010100","1111000000111101010","1111000001001111111","1111000011111011110","1111000101011010110","1111000110111001110","1111001000011000100","1111001001110111010","1111001011010101110","1111001100110100001","1111001110010010100","1111100111111100000101","1111110100100110111010","1111110101010000101 0","1111110101101011011110","1111110111011100100 0","1111110111101110100 1","1111110111111111110000","1111111001011111000","1111111010111110111","1111111011100111011 0","1111111100000111101010","1111111000100111001 0","1111111001110111101 0","1111111010011101010 1","1111111011100110100 0","1111111100100100000 1","1111111101010011100 1","1111111110000111010 1","1111111111111111111 1");

"0011000011101101001 0","0011000100001110010 0","0011000100010111001 1","0011000101010000000 1","0011000101110000110 1","0011000111001000010 1","0011000111010010010 1","0011001000010010101 1","0011001000100010100 0","0011001001000010101 1","0011001001100101010 0","0011001001110010100 1","0011001010000101110 0","0011001010010100010 1","0011001010100000010 1","0011001010110000010 1","0011001011000010010 1","0011001011010000010 1","0011001011100000010 1","0011001011110000010 1","0011001100000010010 1","0011001100010010010 1","0011001100100010010 1","0011001100110000010 1","0011001100110010010 1","0011001100110100010 1","0011001100111000010 1","0011001101000010010 1","0011001101010000010 1","0011001101010010010 1","0011001101010100010 1","0011001101010110010 1","0011001101011000010 1","0011001101011010010 1","0011001101011100010 1","0011001101011110010 1","0011001101100000010 1","0011001101100010010 1","0011001101100100010 1","0011001101100110010 1","0011001101101000010 1","0011001101101010010 1","0011001101101100010 1","0011001101101110010 1","0011001101110000010 1","0011001101110010010 1","0011001101110100010 1","0011001101110110010 1","0011001101111000010 1","0011001101111010010 1","0011001101111100010 1","0011001101111110010 1","0011001110000000010 1","0011001110000010010 1","0011001110000100010 1","0011001110000110010 1","0011001110001000010 1","0011001110001010010 1","0011001110001100010 1","0011001110001110010 1","0011001110010000010 1","0011001110010010010 1","0011001110010100010 1","0011001110010110010 1","0011001110011000010 1","0011001110011010010 1","0011001110011100010 1","0011001110011110010 1","0011001110100000010 1","0011001110100010010 1","0011001110100100010 1","0011001110100110010 1","0011001110101000010 1","0011001110101010010 1","0011001110101100010 1","0011001110101110010 1","0011001110110000010 1","0011001110110010010 1","0011001110110100010 1","0011001110110110010 1","0011001110111000010 1","0011001110111010010 1","0011001110111100010 1","0011001110111110010 1","0011001111000000010 1","0011001111000010010 1","0011001111000100010 1","0011001111000110010 1","0011001111001000010 1","0011001111001010010 1","0011001111001100010 1","0011001111001110010 1","0011001111010000010 1","0011001111010010010 1","0011001111010100010 1","0011001111010110010 1","0011001111011000010 1","0011001111011010010 1","0011001111011100010 1","0011001111011110010 1","0011001111100000010 1","0011001111100010010 1","0011001111100100010 1","0011001111100110010 1","0011001111101000010 1","0011001111101010010 1","0011001111101100010 1","0011001111101110010 1","0011001111110000010 1","0011001111110010010 1","0011001111110100010 1","0011001111110110010 1","0011001111111000010 1","0011001111111010010 1","0011001111111100010 1","0011001111111110010 1","0011010000000000010 1","0011010000010000010 1","0011010000100000010 1","0011010000110000010 1","0011010001000000010 1","0011010001010000010 1","0011010001100000010 1","0011010001110000010 1","0011010010000000010 1","0011010010010000010 1","0011010010100000010 1","0011010010110000010 1","0011010011000000010 1","0011010011010000010 1","0011010011100000010 1","0011010011110000010 1","0011010100000000010 1","0011010100010000010 1","0011010100100000010 1","0011010100110000010 1","0011010101000000010 1","0011010101010000010 1","0011010101100000010 1","0011010101110000010 1","0011010110000000010 1","0011010110010000010 1","0011010110100000010 1","0011010110110000010 1","0011010111000000010 1","0011010111010000010 1","0011010111100000010 1","0011010111110000010 1","0011011000000000010 1","0011011000010000010 1","0011011000100000010 1","0011011000110000010 1","0011011001000000010 1","0011011001010000010 1","0011011001100000010 1","0011011001110000010 1","0011011010000000010 1","0011011010010000010 1","0011011010100000010 1","0011011010110000010 1","0011011011000000010 1","0011011011010000010 1","0011011011100000010 1","0011011011110000010 1","0011011100000000010 1","0011011100010000010 1","0011011100100000010 1","0011011100110000010 1","0011011101000000010 1","0011011101010000010 1","0011011101100000010 1","0011011101110000010 1","0011011110000000010 1","0011011110010000010 1","0011011110100000010 1","0011011110110000010 1","0011011111000000010 1","0011011111010000010 1","0011011111100000010 1","0011011111110000010 1","0011100000000000010 1","0011100000010000010 1","0011100000100000010 1","0011100000110000010 1","0011100001000000010 1","0011100001010000010 1","0011100001100000010 1","0011100001110000010 1","0011100010000000010 1","0011100010010000010 1","0011100010100000010 1","0011100010110000010 1","0011100011000000010 1","0011100011010000010 1","0011100011100000010 1","0011100011110000010 1","0011100100000000010 1","0011100100010000010 1","0011100100100000010 1","0011100100110000010 1","0011100101000000010 1","0011100101010000010 1","0011100101100000010 1","0011100101110000010 1","0011100110000000010 1","0011100110010000010 1","0011100110100000010 1","0011100110110000010 1","0011100111000000010 1","0011100111010000010 1","0011100111100000010 1","0011100111110000010 1","0011101000000000010 1","0011101000010000010 1","0011101000100000010 1","0011101000110000010 1","0011101001000000010 1","0011101001010000010 1","0011101001100000010 1","0011101001110000010 1","0011101010000000010 1","0011101010010000010 1","0011101010100000010 1","0011101010110000010 1","0011101011000000010 1","0011101011010000010 1","0011101011100000010 1","0011101011110000010 1","0011101100000000010 1","0011101100010000010 1","0011101100100000010 1","0011101100110000010 1","0011101101000000010 1","0011101101010000010 1","0011101101100000010 1","0011101101110000010 1","0011101110000000010 1","0011101110010000010 1","0011101110100000010 1","0011101110110000010 1","0011101111000000010 1","0011101111010000010 1","0011101111100000010 1","0011101111110000010 1","0011110000000000010 1","0011110000010000010 1","0011110000100000010 1","0011110000110000010 1","0011110001000000010 1","0011110001010000010 1","0011110001100000010 1","0011110001110000010 1","0011110010000000010 1","0011110010010000010 1","0011110010100000010 1","0011110010110000010 1","0011110011000000010 1","0011110011010000010 1","0011110011100000010 1","0011110011110000010 1","0011110100000000010 1","0011110100010000010 1","0011110100100000010 1","0011110100110000010 1","0011110101000000010 1","0011110101010000010 1","0011110101100000010 1","0011110101110000010 1","0011110110000000010 1","0011110110010000010 1","0011110110100000010 1","0011110110110000010 1","0011110111000000010 1","0011110111010000010 1","0011110111100000010 1","0011110111110000010 1","0011111000000000010 1","0011111000010000010 1","0011111000100000010 1","0011111000110000010 1","0011111001000000010 1","0011111001010000010 1","0011111001100000010 1","0011111001110000010 1","0011111010000000010 1","0011111010010000010 1","0011111010100000010 1","0011111010110000010 1","0011111011000000010 1","0011111011010000010 1","0011111011100000010 1","0011111011110000010 1","0011111100000000010 1","0011111100010000010 1","0011111100100000010 1","0011111100110000010 1","0011111101000000010 1","0011111101010000010 1","0011111101100000010 1","0011111101110000010 1","0011111110000000010 1","0011111110010000010 1","0011111110100000010 1","0011111110110000010 1","0011111111000000010 1","0011111111010000010 1","0011111111100000010 1","0011111111110000010 1","0100000000000000000 0","0011111111111111111 1","0011111111111111110 0","0011111111111111101 1","0011111111111111100 0","0011111111111111010 1","0011111111111111001 1","0011111111111111000 0","0011111111111110101 1","0011111111111110010 1","0011111111111110001 1","0011111111111110000 0","0011111111111101111 1","0011111111111101100 0","0011111111111101010 1","0011111111111101001 1","0011111111111101000 0","0011111111111100111 1","0011111111111100100 1","0011111111111100000 0","0011111111111011111 1","0011111111111011100 0","0011111111111011010 1","0011111111111011001 1","0011111111111011000 0","0011111111111010111 1","0011111111111010100 1","0011111111111010010 1","0011111111111010001 1","0011111111111010000 0","0011111111111001111 1","0011111111111001100 0","0011111111111001010 1","0011111111111001001 1","0011111111111001000 0","0011111111111000111 1","0011111111111000100 1","0011111111111000010 1","0011111111111000001 1","0011111111111000000 0","0011111111110111111 1","0011111111110111100 0","0011111111110111010 1","0011111111110111001 1","0011111111110111000 0","0011111111110110111 1","0011111111110110100 1","0011111111110110010 1","0011111111110110001 1","0011111111110110000 0","0011111111110101111 1","0011111111110101100 0","0011111111110101010 1","0011111111110101001 1","0011111111110101000 0","0011111111110100111 1","0011111111110100100 1","0011111111110100010 1","0011111111110100001 1","0011111111110100000 0","0011111111110011111 1","0011111111110011100 0","0011111111110011010 1","0011111111110011001 1","0011111111110011000 0","0011111111110010111 1","0011111111110010100 1","0011111111110010010 1","0011111111110010001 1","0011111111110010000 0","0011111111110001111 1","0011111111110001100 0","0011111111110001010 1","0011111111110001001 1","0011111111110001000 0","0011111111110000111 1","0011111111110000100 1","0011111111110000010 1","0011111111110000001 1","0011111111110000000 0","0011111111101111111 1","0011111111101111100 0","0011111111101111010 1","0011111111101111001 1","0011111111101111000 0","0011111111101110111 1","0011111111101110100 1","0011111111101110010 1","0011111111101110001 1","0011111111101110000 0","0011111111101101111 1","0011111111101101100 0","0011111111101101010 1","0011111111101101001 1","0011111111101101000 0","0011111111101100111 1","0011111111101100100 1","0011111111101100010 1","0011111111101100001 1","0011111111101100000 0","0011111111101011111 1","0011111111101011100 0","0011111111101011010 1","0011111111101011001 1","0011111111101011000 0","0011111111101010111 1","0011111111101010100 1","0011111111101010010 1","0011111111101010001 1","0011111111101010000 0","0011111111101001111 1","0011111111101001100 0","0011111111101001010 1","0011111111101001001 1","0011111111101001000 0","0011111111101000111 1","0011111111101000100 1","0011111111101000010 1","0011111111101000001 1","0011111111101000000 0","0011111111100111111 1","0011111111100111100 0","0011111111100111010 1","0011111111100111001 1","0011111111100111000 0","0011111111100110111 1","0011111111100110100 1","0011111111100110010 1","0011111111100110001 1","0011111111100110000 0","0011111111100101111 1","0011111111100101100 0","0011111111100101010 1","0011111111100101001 1","0011111111100101000 0","0011111111100100111 1","0011111111100100100 1","0011111111100100010 1","0011111111100100001 1","0011111111100100000 0","0011111111100011111 1","0011111111100011100 0","0011111111100011010 1","0011111111100011001 1","0011111111100011000 0","0011111111100010111 1","0011111111100010100 1","0011111111100010010 1","0011111111100010001 1","0011111111100010000 0","0011111111100001111 1","0011111111100001100 0","0011111111100001010 1","0011111111100001001 1","0011111111100001000 0","0011111111100000111 1","0011111111100000100 1","0011111111100000010 1","0011111111100000001 1","0011111111100000000 0","0011111111011111111 1","0011111111011111100 0","0011111111011111010 1","0011111111011111001 1","0011111111011111000 0","0011111111011110111 1","0011111111011110100 1","0011111111011110010 1","0011111111011110001 1","0011111111011110000 0","0011111111011101111 1","0011111111011101100 0","0011111111011101010 1","0011111111011101001 1","0011111111011101000 0","0011111111011100111 1","0011111111011100100 1","0011111111011100010 1","0011111111011100001 1","0011111111011100000 0","0011111111011011111 1","0011111111011011100 0","0011111111011011010 1","0011111111011011001 1","0011111111011011000 0","0011111111011010111 1","0011111111011010100 1","0011111111011010010 1","0011111111011010001 1","0011111111011010000 0","0011111111011001111 1","0011111111011001100 0","0011111111011001010 1","0011111111011001001 1","0011111111011001000 0","0011111111011000111 1","0011111111011000100 1","0011111111011000010 1","0011111111011000001 1","0011111111011000000 0","0011111111010111111 1","0011111111010111100 0","0011111111010111010 1","0011111111010111001 1","0011111111010111000 0","0011111111010110111 1","0011111111010110100 1","0011111111010110010 1","0011111111010110001 1","0011111111010110000 0","0011111111010101111 1","0011111111010101100 0","0011111111010101010 1","0011111111010101001 1","0011111111010101000 0","0011111111010100111 1","0011111111010100100 1","0011111111010100010 1","0011111111010100001 1","0011111111010100000 0","0011111111010011111 1","0011111111010011100 0","0011111111010011010 1","0011111111010011001 1","0011111111010011000 0","0011111111010010111 1","0011111111010010100 1","0011111111010010010 1","0011111111010010001 1","0011111111010010000 0","0011111111010001111 1","0011111111010001100 0","0011111111010001010 1","0011111111010001001 1","0011111111010001000 0","0011111111010000111 1","0011111111010000100 1","0011111111010000010 1","0011111111010000001 1","0011111111010000000 0","0011111111001111111 1","0011111111001111100 0","0011111111001111010 1","0011111111001111001 1","0011111111001111000 0","0011111111001110111 1","0011111111001110100 1","0011111111001110010 1","0011111111001110001 1","0011111111001110000 0","0011111111001101111 1","0011111111001101100 0","0011111111001101010 1","0011111111001101001 1","0011111111001101000 0","0011111111001100111 1","0011111111001100100 1","0011111111001100010 1","0011111111001100001 1","0011111111001100000 0","0011111111001011111 1","0011111111001011100 0","0011111111001011010 1","0011111111001011001 1","0011111111001011000 0","0011111111001010111 1","0011111111001010100 1","0011111111001010010 1","0011111111001010001 1","0011111111001010000 0","0011111111001001111 1","0011111111001001100 0","0011111111001001010 1","0011111111001001001 1","0011111111001001000 0","0011111111001000111 1","0011111111001000100 1","0011111111001000010 1","0011111111001000001 1","0011111111001000000 0","0011111111000111111 1","0011111111000111100 0","0011111111000111010 1","0011111111000111001 1","0011111111000111000 0","0011111111000110111 1","0011111111000110100 1","0011111111000110010 1","0011111111000110001 1","0011111111000110000 0","0011111111000101111 1","0011111111000101100 0","0011111111000101010 1","0011111111000101001 1","0011111111000101000 0","0011111111000100111 1","0011111111000100100 1","0011111111000100010 1","0011111111000100001 1","0011111111000100000 0","0011111111000011111 1","0011111111000011100 0","0011111111000011010 1","0011111111000011001 1","0011111111000011000 0","0011111111000010111 1","0011111111000010100 1","0011111111000010010 1","0011111111000010001 1","0011111111000010000 0","0011111111000001111 1","0011111111000001100 0","0011111111000001010 1","0011111111000001001 1","0011111111000001000 0","0011111111000000111 1","0011111111000000100 1","0011111111000000010 1","0011111111000000001 1","0011111111000000000 1"

00000001001","00111110011101010111","00111110011010100010","00111110010111101010","00111110010100110000","001111100100
01110100","00111110001110110101","00111110001011110011","00111110001000101111","00111110000101101001","001111100000101
00000","00111101111111010100","00111101111100000110","00111101111000110110","00111101110101100010","001111011100100011
01","00111101101110110101","00111101101011011010","00111101100111111101","00111101100100011110","00111101100000111100"
,"00111101011101010111","00111101011001110000","00111101010110000111","00111101010010011011","00111101001110101100","0
0111101001010111011","00111101000111001000","00111101000011010010","00111100111111011010","00111100111011011111","0011
1100110111100010","00111100110011100010","00111100101111100000","00111100101011011011","00111100100111010100","0011110
0100011001010","00111100011110111110","00111100011010110000","00111100010110011111","00111100010010001100","0011110000
1101110110","00111100001001011110","00111100000101000011","00111100000000100110","00111011111100000111","0011101111011
1100101","00111011110011000001","00111011101110011010","00111011101001110001","00111011100101000101","001110111000000
1
0111","00111011011011100111","00111011010110110100","00111011010001111111","00111011001101000111","001110110010000011
0
1","00111011000011010001","00111010111110010010","00111010111001010001","00111010110100001110","00111010101111001000",
"00111010101010000000","00111010100100110101","00111010011111101000","00111010011010011000","00111010010101000111","00
111010001111110011","00111010001010011100","00111010000101000100","00111001111111101000","00111001111010001011","00111
001110100101011","00111001101111001001","00111001101001100100","00111001100011111110","00111001011110010100","00111001
011000101001","00111001010010111011","00111001001101001011","00111001000111011001","00111001000001100100","00111000111
011101101","00111000110101110100","00111000101111111000","00111000101001111010","00111000100011111010","00111000011101
110111","00111000010111110010","00111000010001101011","00111000001011100010","00111000000101010110","00110111111111001
001","00110111111000111000","00110111110010100110","00110111101100010001","00110111100101111010","00110111011111100001
","00110111011001000110","00110111010010101000","00110111001100001000","00110111000101100110","00110110111111000010","
00110110111000011011","00110110110001110011","00110110101011001000","00110110100100011010","00110110011101101011","001
10110010110111001","00110110010000000110","00110110001001010000","00110110000010010111","00110101111011011101","001101
01110100100000","00110101101101100010","00110101100110100001","00110101011111011110","00110101011000011001","001101010
10001010001","00110101001010001000","00110101000010111100","00110100111011101110","00110100110100011110","001101001011
01001100","00110100100101111000","00110100011110100010","00110100010111001001","00110100001111101110","001101000010000
10010","00110100000000110011","00110011111001010010","00110011110001101111","00110011101010001010","00110011100010100
0
11","00110011011010111001","00110011010011001110","00110011001011100000","00110011000011110001","00110010111011111111"
,"00110010110100001100","00110010101100010110","00110010100100011110","00110010011100100100","00110010010100101001","0
0110010001100101011","00110010000100101011","00110001111100101001","00110001110100100101","00110001101100011111","0011
0001100100010111","00110001011100001101","00110001010100000001","00110001001011110011","00110001000011100100","0011000
0111011010010","00110000110010111110","00110000101010101000","00110000100010010000","00110000011001110110","0011000001
0001011010","00110000001000111101","00110000000000011101","00101111110111111100","00101111101111011000","0010111110011
0110010","00101111011110001011","00101111010101100010","00101111001100110111","00101111000100001010","0010111011101101
1010","00101110110010101010","00101110101001110111","00101110100001000010","00101110011000001011","001011100011110100
1
1","00101110000110011001","00101101111101011100","00101101110100011110","00101101101011011110","00101101100010011101",
"00101101011001011001","00101101010000010100","00101101000111001100","00101100111110000011","00101100110100111000","00
101100101011101100","00101100100010011101","00101100011001001101","00101100001111111011","00101100000110100111","00101
011111101010001","00101011110011111010","00101011101010100001","00101011100001000110","00101011010111101001","00101011
001110001010","00101011000100101010","00101010111011001000","00101010110001100100","00101010100111111111","00101010011
110011000","00101010010100101111","00101010001011000100","00101010000001011000","00101001110111101010","00101001101101
111010","00101001100100001001","00101001011010010110","00101001010000100001","00101001000110101011","00101000111100110
010","00101000110010111001","00101000101000111101","00101000011111000000","00101000010101000010","00101000001011000010
","00101000000001000000","00100111110110111100","00100111101100110111","00100111100010110000","00100111011000101000","
00100111001110011110","00100111000100010010","00100110111010000101","00100110101111110110","00100110100101100110","001
00110011011010100","00100110010001000001","00100110000110101100","00100101111100010110","00100101110001111110","001001
01100111100100","00100101011101001001","00100101010010101101","00100101001000001111","00100100111101101111","001001001
10011001110","00100100101000101100","00100100011110001000","00100100010011100010","00100100001000111011","001000111111
10010011","00100011110011101001","00100011101000111110","00100011011110010001","00100011010011100011","001000110010001
10011","00100010111110000010","00100010110011010000","00100010101000011100","00100010011101100110","001000100100101100
00","00100010000111111000","00100001111100111110","00100001110010000100","00100001100111000111","00100001011100001010"
,"00100001010001001011","00100001000110001011","00100000111011001001","00100000110000000110","00100000100101000010","0
0100000011001111101","00100000001110110110","00100000000011101110","00011111111000100100","00011111101101011010","0001
1111100010001110","00011111010111000000","00011111001011110010","00011111000000100010","00011110110101010001","0001111
0101001111110","00011110011110101011","00011110010011010110","00011110001000000000","00011101111100101001","0001110111
0001010000","00011101100101110111","00011101011010011100","00011101001111000000","00011101000011100011","0001110011100
0000100","00011100101100100101","00011100100001000100","00011100010101100010","00011100001001111111","0001101111111001
1011","00011011110010110110","00011011100111010000","00011011011011101000","00011011001111111111","00011011000100001011

```
0","0001101011100010101 1","0001101010110011111 1","0001101010000101001 0","0001101001010110010 0","0001101000100111010 1",
"0001100111111100001 01","0001100111100100101 00","0001100111001100001","0001100101101010111 0","0001100100111011101 0","00
011001000001100010 0","0001100001101110011 10","0001100010101101011 0","0001100001111101111 0","0001100001001110010 1","00011
000000011101010","0001011111101110111 1","0001011110111111001 1","0001011100011110110","0001011101011111011 1","00010111
001011111000","0001011011111111000 0","0001011011001111011 1","0001011010011111010 1","0001011001111101011 0","00010110001
111101111","0001011000001110101 0","0001010111011110010 0","0001010110101101111 0","0001010101111101011 1","00010101010011
001111","0001010100001110001 10","0001010011101011110 0","0001010010111011000 1","0001010010000101001 01","0001010001011001100
1","0001010000101000110 0","0001001111110111111 0","0001001111000110111 1","0001001110101111111 1","0001001110010011001111
","0001001100111011 10","0001001100000001011 00","0001001011010001100 1","0001001010000001011 0","0001001011001100111 1","
0001001000011101110 0","0001001000001100011 0","0001000111010000001 1","0001000110011101010 1","0001000101101010011 0","0000
1001001011101111000","0000100100000100100 0","0000100011010001100 1","0000100010011110100 1","0000100001101011100 0","0000
0001110001000","0000100000000101011 1","0000011111010010011 0","0000011110011111010 0","0000011100001110010000","0000011000000
1011110","0000011010010100101011","0000011010011111000","0000011001101100010 1","0000011000011100010010","0000011000000101111 0
","0000010111010010101 0","0000010011010010011 0","0000010010011110111 0","0000010001101100001","0000010000111000001 0
","0000010000000010011 00","0000001111010001011 0","0000001110011110000 0","0000001101101010100 0","0000001100110111001 1",
"0000001100000011101","0000001011010000011 0","0000001010011100111 1","0000001001101110011 1","0000001000111011100 0","0000000
100000010110","0000000011001101111 0","0000000010011010011 0","0000000001100111000"

);

----------

CONSTANT minus1: sfixed(1 downto 0):= "11";

begin

---------- Απόδοση Τιμών στους Διακόπτες S1,S2,S3,S4 ----------

PROCESS(clk,reset)

 variable count_Ac: INTEGER RANGE -2500 TO 2500;

 variable index: INTEGER RANGE 0 TO 1000;

 variable count_index: INTEGER;

 variable CMPA,CMPB: sfixed(16 downto -20);

 variable count_S1,count_S3: INTEGER;

 variable Delay_eval_S1,Delay_eval_S3,set_S1,set_S3: STD_LOGIC;

 variable up_Ac,pos_index: STD_LOGIC;

 begin

  --if(reset = '1') then

   if(CLK'event and CLK='1') then
```

```
            if(reset = '1') then
    ------------------------------------------------------------- Υλοποίηση Τριγωνικού Σήματος ----------------------------------------------------------
      if(up_Ac = '1') then
        count_Ac := count_Ac + 1;
        if (count_Ac = 2500 ) then
          up_Ac := '0';
        end if;
      else
        count_Ac:= count_Ac - 1;
        if (count_Ac = -2500) then
          up_Ac:= '1';
        end if;
      end if;
-------------------------------------------------------------------------------------------------------------------------------------------------------------
    count_index:= count_index + 1;
    if(count_index > 499) then
     count_index:= 0;
------------------------------------------- 0 – T/2 της Περιόδου του Ημιτονοειδούς Σήματος ----------------------------------------------------
      if(pos_index = '1') then
        CMPA:= "00"&Ar*sine_values(index);   -- So as to have 16 bit Integer Part.
        CMPB:= minus1*Ar*sine_values(index);
        index:= index + 1;
        if(index = 1000) then
          pos_index:= '0';
          index:= 0;
        end if;
-------------------------------------------------------------------------------------------------------------------------------------------------------------
      else
------------------------------------------- T/2 – T της Περιόδου του Ημιτονοειδούς Σήματος ----------------------------------------------------
        CMPA:= minus1*Ar*sine_values(index);
        CMPB:= "00"&Ar*sine_values(index);
```

```
        index:= index + 1;
        if(index = 1000) then
          pos_index:= '1';
          index:= 0;
        end if;
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
      end if;
---------------------------------------------------- Χειρισμός Ημιαγωγικών Διακοπτών S1-S2 ----------------------------------------------------------------------------------------------
      if (count_Ac < CMPA ) then
         set_S1:= '1';       -- S1 should be set to '1'
         Delay_eval_S1:= '1';  -- Is '1' while S1 is delayed!!
      else
         set_S1:= '0';
         Delay_eval_S1:= '1';
      end if;
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------------- Χειρισμός Ημιαγωγικών Διακοπτών S3 – S4 ----------------------------------------------------------------------------------------
      if(count_Ac < CMPB) then
         set_S3:= '1';
         Delay_eval_S3:= '1';
      else
         set_S3:= '0';
         Delay_eval_S3:= '1';
      end if;
    end if;
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------- Υλοποίηση ‘’νεκρού’’ Χρονικού Διαστηματος για S1-S2 ----------------------------------------------------------------------------
    if(set_S1 = '1') then
      if(Delay_eval_S1 = '1') then
        S2<= '0';
        if(count_S1 = 50) then
```

```
          S1<= '1';
          Delay_eval_S1:= '0'; -- I dont want to be delayed any more.
          count_S1:= 0;
         else
          count_S1:= count_S1 + 1;
         end if;
       end if;
     elsif(set_S1 = '0') then
       if(Delay_eval_S1 = '1') then
          S1<= '0';
          if(count_S1 = 50) then
            S2<= '1';
            Delay_eval_S1:= '0';
            count_S1:= 0;
          else
            count_S1:= count_S1 + 1;
          end if;
        end if;
      end if;
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------ Υλοποίηση ‘‘νεκρού’’ Χρονικού Διαστήματος για S3-S4 ------------------------------------------------------------
      if(set_S3 = '1') then
        if(Delay_eval_S3 = '1') then
          S4<= '0';
          if(count_S3 = 50) then
            S3<= '1';
            Delay_eval_S3:= '0';
            count_S3:= 0;
          else
            count_S3:= count_S3 + 1;
          end if;
```

```
      end if;
    elsif(set_S3 = '0') then
     if(Delay_eval_S3 = '1') then
       S3<= '0';
       if(count_S3 = 50) then
        S4<= '1';
        Delay_eval_S3:= '0';
        count_S3:= 0;
       else
        count_S3:= count_S3 + 1;
       end if;
     end if;
    end if;
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

---------------------------------------------------- Αρχικοποίηση Σημάτων για reset = ‘0’ ----------------------------------------------------------------------------------------

  else
   pos_index:= '1';
   up_Ac:= '1';
   S1<= '0';
   S2<= '1';
   S3<= '0';
   S4<= '1';
              count_Ac:= 0;
  count_index:= 499;
              index:= 0;
  count_S1:= 0;
  count_S3:= 0;
  Delay_eval_S1:= '0';
  Delay_eval_S3:= '0';
  set_S1:= '0';
   set_S3:= '0';
```

```
  end if;
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 end if;
 end PROCESS;
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
Ac<= 2500;
Ar_reg<= ma*Ac/100;
Ar<= to_sfixed(Ar_reg,12,-2);
end behavioural;
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
```

VHDL Κώδικας της SPWM τεχνικής με Δυναμικό Υπολογισμό Δειγμάτων

```
---------------------------------------------- Δήλωση Απαραίτητων Βιβλιοθηκών και Πακέτων ----------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_SIGNED.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.all;
library ieee_proposed;
use IEEE_PROPOSED.FIXED_FLOAT_TYPES.ALL;
use IEEE_PROPOSED.FIXED_PKG.ALL;
library std;
use std.standard.all;
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------- Δήλωση της Ενότητας που θα χρησιμοποιήσουμε ----------------------------------------------------------------------
entity SPWM_CORDIC is
GENERIC (ma: INTEGER := 70; fr: INTEGER:= 50; fc: INTEGER:= 5000); -- ma is %
Port (CLK: in STD_LOGIC;
    reset: in STD_LOGIC;
```

```
    S1,S2,S3,S4: inout STD_LOGIC);
end SPWM_CORDIC;
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------- Δήλωση της Αρχιτεκτονικής που θα χρησιμοποιήσουμε ----------------------------------------------------------
Architecture behavioural of SPWM_CORDIC is
------------------------------------------- Δήλωση Συστατικού Στοιχείου για Υλοποίηση του Ημιτόνου ---------------------------------------------------
COMPONENT sin_function
 PORT (
   phase_in : IN STD_LOGIC_VECTOR(30 DOWNTO 0);
   y_out : OUT STD_LOGIC_VECTOR(15 DOWNTO 0);
   clk : IN STD_LOGIC
 );-
END COMPONENT;
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
signal Ar: sfixed(12 downto -2);
signal Ar_reg: INTEGER;
signal Ac: INTEGER;
signal samples: INTEGER;
signal sin_val: std_logic_vector(15 downto 0);
signal CMPA,CMPB: sfixed(12 downto -8);
signal reg_CMPA: sfixed(14 downto -16);
signal reg_CMPB: sfixed(16 downto -16);
signal sample_A,sample_B: STD_LOGIC;        -- Ενεργοποιείται μόλις βρεθεί σημείο τομής.
signal SAMPLES_READY: STD_LOGIC;
signal Triangular: STD_LOGIC;
signal rad: sfixed(3 downto -27);
CONSTANT pi: sfixed(2 downto -8):= "01100100100"; -- Αναπαράσταση σε sfixed αριθμό του π =3.14.
signal rad_reg: std_logic_vector(30 downto 0);
signal rad_offset: sfixed(3 downto  -27);
--CONSTANT val_10000: sfixed(13 downto 0):= "10011100010000";
CONSTANT val_500000: sfixed(19 downto 0):= "01111010000100100000";
```

```
CONSTANT minus1: sfixed(1 downto 0):= "11";


TYPE vector_array is ARRAY (0 TO 99) of sfixed(12 downto -8);
signal sine_values_A,sine_values_B: vector_array;
begin
---------------------------------------------- Υλοποίηση Συχνότητας Τριγωνικού Σήματος 5 κHz ----------------------------------------------------------
 PROCESS(CLK,reset)
  variable count1: INTEGER RANGE -2500 TO 2500;
  variable up: STD_LOGIC;
  begin
   if(CLK'event and CLK='1') then
     if(reset = '1') then
      if (up = '1') then
        count1:= count1+1;
        if (count1 = 2500) then
          up:='0';
          Triangular<='0';  -- Μέσο περιόδου τριγωνικού σήματος.
       end if;
     else
      count1:= count1-1;
      if (count1 = -2500) then
        up:='1';
        Triangular<='1'; -- Τέλος περιόδου τριγωνικού σήματος.
       end if;
     end if;
   else  -- Αρχικοποίηση σημάτων για reset = '1'.
      up:= '1';
     count1:= 0;
     Triangular<= '1';
   end if;
 end if;
```

```
end process;

-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

---------------------------------------------- Αποδόσεις Τιμών στα CMPA και CMPB ----------------------------------------------------------------------------------

PROCESS(Triangular,reset)

 variable index: INTEGER RANGE 0 TO 99;

 variable pos_index: STD_LOGIC;

 begin

 if(reset = '1') then

  if(SAMPLES_READY = '1') then


------------------------------------------------------ 0 – T/2 της Περιόδου του Ημιτόνου ------------------------------------------------------------------------------

    if(pos_index = '1') then

      CMPA<= sine_values_A(index);

      CMPB<= sine_values_B(index);

      index:= index + 1;

      if(index = 99) then

        pos_index:= '0';

        index:= 0;

      end if;

-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

    else

------------------------------------------------------ T/2 - T της Περιόδου του Ημιτόνου ------------------------------------------------------------------------------

      CMPA<= sine_values_B(index);

      CMPB<= sine_values_A(index);

      index:= index + 1;

      if(index = 99) then

        pos_index:= '1';

        index:= 0;

      end if;

-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

   end if;
```

```
  end if;
 else   -- Αρχικοποίηση σημάτων για reset = '1'.
   pos_index:= '0';
   index:= 0;
   CMPA<= "00000000000000000000";
   CMPB<= "00000000000000000000";
 end if;
end PROCESS;
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

------------------------------------------------ Παραγωγή Παλμών των Διακοπτών S1,S2,S3,S4 ------------------------------------------------------------------------

PROCESS(CLK,reset)
 variable count_Ac: INTEGER RANGE -2500 TO 2500;
 variable i,count_S1,count_S3: INTEGER;
 variable temp_rad: sfixed(4 downto -27);
 variable up_Ac,up_Ac0,set_S1,set_S3: STD_LOGIC;
 variable Delay_eval_S1,Delay_eval_S3: STD_LOGIC;
 begin
  if(CLK'event and CLK='1') then
     if(reset = '1') then
-------------------------------- 1ο Διάστημα Λειτουργίας: Αποθήκευση Δειγμάτων ------------------------------------------------------------------------------------

        if(SAMPLES_READY = '0') then
---------------------------------- Αποθήκευση Δειγμάτων κατά την Άνοδο του Τριγωνικού Σήματος-----------------------------------------------------------------
          if (up_Ac0 = '1') then
            if(count_Ac >= reg_CMPA) then
               if(set_S1 = '1') then
                  Delay_eval_S1:= '1';
               end if;
               sample_A<= '1';  -- Δείγμα για το CMPA κατά το ''ανέβασμα'' του τριγωνικού παλμού.
               set_S1:= '0';
          else
             if(set_S1 = '0') then
```

```
            Delay_eval_S1:= '1';
        end if;
        set_S1:= '1';
      end if;
      if(count_Ac >= reg_CMPB) then
        if(set_S3 = '1') then
            Delay_eval_S3:= '1';
        end if;
        sample_B<= '1';  -- Δείγμα για το CMPB κατά το ''ανέβασμα'' του τριγωνικού παλμού.
        set_S3:= '0';
      else
       if(set_S3 = '0') then
         Delay_eval_S3:= '1';
       end if;
       set_S3:= '1';
      end if;
       count_Ac:= count_Ac + 1;
       if(count_Ac = 2500) then
         up_Ac0:= '0';
       end if;
       reg_CMPA<= Ar*to_sfixed(sin_val,1,-14);
       reg_CMPB<= minus1*Ar*to_sfixed(sin_val,1,-14);
       temp_rad:= rad + rad_offset;
       rad<= temp_rad(3 downto -27);
       rad_reg<= To_slv(rad); -- Κατάλληλη μορφή για εισαγωγή στο CORDIC.
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------


------------------------------ Αποθήκευση Δειγμάτων κατά την Κάθοδο του Τριγωνικού Σήματος---------------------------------------------------------

    else
      count_Ac:= count_Ac - 1;
      if (count_Ac = -2500) then
```

```
 up_Ac0:= '1';
 i:= i + 1;                -- Ολοκλήρωση της περιόδου του τριγωνικού σήματος.
 if (i = samples/2) then
  SAMPLES_READY<= '1';
 end if;
end if;
if(count_Ac <= reg_CMPA ) then
 if(set_S1 = '0') then
  Delay_eval_S1:= '1';
 end if;
 sample_A<= '0';  -- Δείγμα για το CMPA κατά το ‘’κατέβασμα’’ του τριγωνικού παλμού
 set_S1:= '1';
else
 if(set_S1 = '1') then
  Delay_eval_S1:= '1';
 end if;
 set_S1:= '0';
end if;
if(count_Ac <= reg_CMPB) then
 if(set_S3 = '0') then
  Delay_eval_S3:= '1';
 end if;
 sample_B<= '0';  -- Δείγμα για το CMPB κατά το ‘’κατέβασμα’’ του τριγωνικού παλμού
 set_S3:= '1';
else
 if(set_S3 = '1') then
  Delay_eval_S3:= '1';
 end if;
 set_S3:= '0';
end if;
temp_rad:= rad + rad_offset;
```

```
      rad<= temp_rad(3 downto -27);
      rad_reg<= to_slv(rad); -- Κατάλληλη μορφή για εισαγωγή στο CORDIC.
      reg_CMPA<= Ar*to_sfixed(sin_val,1,-14);
      reg_CMPB<= minus1*Ar*to_sfixed(sin_val,1,-14);
    end if;
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------
---------------------------------------------- 2ο Διάστημα Λειτουργίας: Σύγκριση μέσω CMPA – CMPB ----------------------------------------------
  else
    if(up_Ac = '1') then
      count_Ac := count_Ac + 1;
      if (count_Ac = 2500) then
        up_Ac := '0';
      end if;
    else
      count_Ac:= count_Ac - 1;
      if (count_Ac = -2500) then
        up_Ac:= '1';
      end if;
    end if;
    if (count_Ac < CMPA ) then
      if(set_S1 = '0') then   -- Σε περίπτωση που το CMPA έγινε μόλις > count_Ac.
        Delay_eval_S1:= '1';
      end if;
      set_S1:= '1';
    else
      if(set_S1 = '1') then  -- Σε περίπτωση που το CMPA έγινε μόλις < count_Ac.
        Delay_eval_S1:= '1';
      end if;
      set_S1:= '0';
    end if;
    if(count_Ac < CMPB) then
```

```
    if(set_S3 = '0') then  -- Σε περίπτωση που το CMPB έγινε μόλις > count_Ac.
      Delay_eval_S3:= '1';
    end if;
    set_S3:= '1';
   else
    if(set_S3 = '1') then  -- Σε περίπτωση που το CMPB έγινε μόλις < count_Ac.
      Delay_eval_S3:= '1';
    end if;
    set_S3:= '0';
   end if;
  end if;
  if(set_S1 = '1') then
   if(Delay_eval_S1 = '1') then
    S2<= '0';
    if(count_S1 = 50) then
      S1<= '1';
      Delay_eval_S1:= '0';
      count_S1:= 0;
    else
      count_S1:= count_S1 + 1;
    end if;
   end if;
  elsif(set_S1 = '0') then
   if(Delay_eval_S1 = '1') then
    S1<= '0';
    if(count_S1 = 50) then
      S2<= '1';
      Delay_eval_S1:= '0';
      count_S1:= 0;
    else
      count_S1:= count_S1 + 1;
```

```
      end if;
    end if;
  end if;
  if(set_S3 = '1') then
    if(Delay_eval_S3 = '1') then
      S4<= '0';
      if(count_S3 = 50) then
        S3<= '1';
        Delay_eval_S3:= '0';
        count_S3:= 0;
      else
        count_S3:= count_S3 + 1;
      end if;
    end if;
  elsif(set_S3 = '0') then
    if(Delay_eval_S3 = '1') then
      S3<= '0';
      if(count_S3 = 50) then
        S4<= '1';
        Delay_eval_S3:= '0';
        count_S3:= 0;
      else
        count_S3:= count_S3 + 1;
      end if;
    end if;
  end if;
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 else  -- Αρχικοποίηση σημάτων για reset = '1'.
    SAMPLES_READY<= '0';
    sample_A<= '0';
    sample_B<= '0';
```

```
    up_Ac:= '1';
    up_Ac0:= '1';
    rad<= "00000000000000000000000000000000";
    set_S1:= '0';
    set_S3:= '0';
    reg_CMPA<= "00000000000000000000000000000000";
    reg_CMPB<= "00000000000000000000000000000000";
    count_Ac:= 0;
      i:= 0;
      Delay_eval_S1:= '0';
      Delay_eval_S3:= '0';
      count_S1:= 0;
      count_S3:= 0;
      S1<= '0';
      S2<= '1';
      S3<= '0';
      S4<= '1';
    end if;
  end if;
end PROCESS;
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 PROCESS(sample_A,reset)
 variable index: INTEGER RANGE 0 TO 100;
  begin
   if(reset'event and reset = '1') then
   elsif(reset = '1') then
    if(sample_A = '1' or sample_A = '0') then
     sine_values_A(index)<= reg_CMPA(12 downto -8);
     index:= index + 1;
    end if;
   elsif(reset = '0') then
```

```
      index:= 0;
    end if;
  end PROCESS;

PROCESS(sample_B,reset)
 variable index: INTEGER RANGE 0 TO 100;
  begin
    if(reset'event and reset = '1') then
    elsif(reset = '1') then
     if(sample_B = '1' or sample_B = '0') then
      if(index = 0) then
        sine_values_B(index)<= "00000000000000000000";
        index:= index + 1;
      else
        sine_values_B(index)<= '1'&reg_CMPB(11 downto -8); -- That's in order to achieve (-)!!
        index:= index + 1;
      end if;
     end if;
    elsif(reset = '0') then
      index:= 0;
    end if;
  end PROCESS;

U1: sin_function_float port map (rad_reg,sin_val,CLK);
samples<= 2*(fc/(2*fr));
rad_offset<= pi/(val_500000);--pi/(10000*2*(samples/2)) because we have 2 samples per triangular wave.
Ac<= 2500;
Ar_reg<= (ma*Ac)/100;
Ar<= to_sfixed(Ar_reg,12,-2);
end behavioural;
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] **Στέφανος Ν. Μανιάς Καθηγητής Ε.Μ.Π,** ‘’Ηλεκτρονικά Ισχύος’’, Εκδόσεις Συμεών Αθήνα 2007

[2] **Στέφανος Ν. Μανιάς Καθηγητής Ε.Μ.Π,** ‘’Ηλεκτρονικά Ισχύος’’, Εκδόσεις Συμεών Αθήνα 2012

[3] **Volnei A. Pedroni,** ‘’Σχεδιασμός κυκλωμάτων με τη VHDL’’, Επιστημονική επιμέλεια ελληνικής έκδοσης: **Γεώργιος Θεοδωρίδης, Λέκτορας Α.Π.Θ.,** Εκδόσεις Κλειδάριθμος, Αθήνα 2007

[4] **Neil H. E. Weste, David M. Harris,** ‘’Σχεδίαση Ολοκληρωμένων Κυκλωμάτων, CMOS VLSI Design’’, Εκδόσεις Παπασωτηρίου, Αθήνα 2011

[5] **Wayne Wolf,** ‘’Οι Υπολογιστές ως Συστατικά Στοιχεία’’, Επιμέλεια Ελληνικής Έκδοσης: **Νικόλαος Σπ. Βώρος, Δημήτρης Σ. Κριθαρίδης, Κων/νος Γ. Μασσέλος,** Εκδόσεις Νέων Τεχνολογιών

[6] **Stephen J. Chapman,** ‘’Ηλεκτρικές Μηχανές’’, Επιμέλεια Μετάφρασης: **Θεόδωρος Π. Θεοδουλίδης Αναπληρωτής Καθηγητής Τμήμα Μηχανολόγων Μηχανικών Πανεπιστήμιο Δυτικής Μακεδονίας,** Εκδόσεις Τζιόλα

[7] **Ν. Καδιανάκης Αν. Καθηγητής Ε.Μ.Π, Σ. Καρανάσιος Αν. Καθηγητής Ε.Μ.Π,** ‘’Γραμμική Άλγεβρα Αναλυτική Γεωμετρία και Εφαρμογές’’, Αθήνα 2008

[8] **Κ.Ζ. Πεκμεστζή Καθηγητής Ε.Μ.Π,** ‘’Ψηφιακά Συστήματα VLSI’’ Αθήνα 2003

[9] **L. Hassaine, E. Olias, M. Haddadi and A. Malek,** ‘’ Asymmetric SPWM used in inverter grid connected’’, Revue des Energies Renouvelables Vol. 10 N°3 (2007) pp 421 – 429, September 2007

[10] **S. N. Singh, A. K. Singh,** ‘’FPGA Based Sinusoidal Pulse Width Modulated Waveform Generation for Solar (PV) Rural Home Power Inverter’’, Journal of Telecommunications, Volume 1, Issue 1, February 2010

[11] **C. Senthilsingh and Dr. M. Manikandan,** ‘’FPGA based ON – Line UPS using PWM Technique’’, International Conference on Computing and Control Engineering (ICCCE 2012), 12 & 13 April, 2012

[12] **Majd Ghazi Batarseh, Wisam Al-Hoor, Lilly Huang, Chris Iannello, Issa Batarseh,** ‘’Window-Masked Segmented Digital Clock Manager-FPGA-Based Digital Pulsewidth Modulation Technique’’, IEEE Transactions on Power Electronics, Vol. 24, No. 11, November 2009

[13] **Jakirhusen I. Tamboli, Prof. Satyawan R. Jagtap, Amol R. Sutar,** ‘’Pulse Width Modulation Implementation using FPGA and CPLD ICs’’, International Journal of Scientific & Engineering Research Volume 3, Issue 8, August 2012

[14] **Matina Lakka, Eftichios Koutroulis, Apostolos Dollas,** ‘’An FPGA-based SPWM Generator for High-Frequency DC/AC Inverters’’, Department of Electronic & Computer Engineering, Technical University of Crete

[15] **Eftichios Koutroulis, Apostolos Dollas, Kostas Kalaitzakis,** ‘’High – frequency Pulse Width Modulation Implementation using FPGA and CPLD ICs’’, Journal of Systems Architecture 52(2006) pp 332 – 344, 2006

[16] **Suryakant Behera,** ‘’FPGA based PWM techniques for controlling Inverter’’, Bachelor of Technology in Electronics and Instrumentation Engineering, 2010

[17] **Sadhana Kumari,** ''Development of a Controller for Fuel Cell using FPGA'', Master of Technology in VLSI Design & Embedded System, 2011

[18] **Guan – Lin Wu,** ''Introduction to FPGA''

[19] **David Bishop,** ''Fixed Point Package User's Guide''

[20] **Xilinx,** ''LogiCORE IP CORDIC v4.0'', DS249 March 1, 2011

[21] **Lin MacCleery,** ''Reconfigurable Grid? FPGAs Versus DSPs for Power Electronics'', Final ETS 2012

[22] **Brian MacCleery National Instruments Principal Product Manager for Clean Energy Technology,** ''FGPA-based Real-Time Simulation of Power Electronics: Challenges & Solutions''

[23] **Bindeshwar Singh, Nupur Mittal, Dr. K.S. Verma, Dr. Deependra Singh, S.P.Singh, Rahul, Dixit, Manvendra Singh, Aanchal Baranwal,** ''Multi-Level Inverter: A Literature Survey on Topologies and Control Strategies'', International Journal of Reviews in Computing, Vol.10 31th July 2012

[24] **Oscar Lopez, Jacobo Alvarez, Jesus Doval – Candoy, Francisco D. Freijedo, Andres Nogueiras, Alfonso Lago, Carlos M. Penalver,** ''Comparison of the FPGA Implementation of Two Multilevel Space Vector PWM Algorithms'', IEEE Transactions on Industrial Electronics, Vol. 55, No. 4, April 2008

[25] **C. Govindaraju, K. Baskaran,** ''Performance Improvement of Multiphase Multilevel Inverter Using Hybrid Carrier Based Space Vector Modulation'', International Journal on Electrical Engineering and Informatics – Volume 2, Number 2, 2010

[26] **Jose I. Leon, Samir Kouro, Sergio Vazquez, Ramon Portillo, Leopoldo G. Franquelo, Juan M. Carrasco, Jose Rodriguez,** ''Multidimensional Modulation Technique for Cascaded Multilevel Converters'', IE Tech News, March 2013

[27] **R.Rajendran, Dr.N.Deverajan,** ''Analysis and FPGA Realization of a Pulse Width Modulator based on Voltage Space Vectors'', International Journal of Computer Applications, Volume 2 – No. 6, pp 46-51, June 2010

[28] **Bahram Rashidi, Mehran Sabahi,** ''High Performance FPGA Based Digital Space Vector PWM Three Phase Voltage Source Inverter'', I.J.Modern Education and Computer Science, 2013, 1, pp 62-71

[29] **Oscar Lopez, Jacobo Alvarez, Jesus Doval – Candoy, Francisco D. Freijedo,** ''Multilevel Multiphase Space Vector PWM Algorithm'' IEEE Transactions on Industrial Electronics , Vol. 55, No.5, May 2008

[30] **Oscar Lopez, Jacobo Alvarez, Jesus Doval – Candoy, Francisco D. Freijedo,** ''Multilevel Multiphase Space Vector PWM Algorithm with Switching State Redundancy'', IEEE Transactions on Industrial Electronics, Vol. 56, No. 3, March 2009

[31] **Oscar Lopez, Jacobo Alvarez, Jesus Doval – Candoy, Francisco D. Freijedo,** ''Digital Parameterizable VHDL Module for Multilevel Multiphase Space Vector PWM'', IEEE Transaction on Industrial Electronics, Vol. 58, No.9, September 2011