Ανάπτυξη σε FPGA ηλεκτρονικού κυκλώματος για την αφαίρεση υποβάθρου και θορύβου ψηφιοποιημένου σήματος που προέρχεται από μικρολωριδιακούς αισθητήρες πυριτίου.

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Οδυσσέας Γ. Μητρόπουλος
Φυσικός

Επίβλεψη: Επίκουρος Καθηγητής Ν. Μάνθος
Εργαστήριο Φυσικής Υψηλών Ενεργειών

ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΣΤΙΣ ΣΥΓΧΡΟΝΕΣ ΗΛΕΚΤΡΟΝΙΚΕΣ ΤΕΧΝΟΛΟΓΙΕΣ

ΤΜΗΜΑ ΦΥΣΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

Ιωάννινα Σεπτέμβριος 2001

αφιερώνεται στους γονείς μου Γιώργο και Τρίκκη,
στον αδελφό μου Κωνσταντίνο και στη Βάγια

Περίληψη

Το αντικείμενο της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη σε ένα FPGA ηλεκτρονικού κυκλώματος, που σκοπό έχει την αφαίρεση υποβάθρου (PEDESTAL) και θορύβου (COMMON NOISE) ψηφιοποιημένων σημάτων, που προέρχονται από μικρολωριδιακούς αισθητήρες πυριτίου. Τα σήματα από τους αισθητήρες πυριτίου ενισχύονται–μορφοποιούνται με κατάλληλα αναλογικά ηλεκτρονικά FE (Front End) και οδηγούνται σε μετατροπείς αναλογικών σε ψηφιακών σημάτων (ADCs). Τα ψηφιακά δεδομένα από τους ADCs οδηγούνται σε μονάδες επεξεργασίας ψηφιακού σήματος (FPGAs) προκειμένου να γίνει η αφαίρεση του υποβάθρου του σήματος και του κοινού θορύβου. Στην παρούσα εργασία θεωρήθηκε ότι κάθε υποσύστημα αισθητήρα-FE έχει 32 κανάλια. Αρχικά μελετήθηκαν αλγόριθμοι με τεχνικές Monte Carlo για τον υπολογισμό και αφαίρεση του υποβάθρου και του θορύβου. Στη μελέτη θεωρήθηκε ότι το υπόβαθρο είναι διαφορετικό για κάθε κανάλι και ότι ο θόρυβος είναι κοινός για κάθε ομάδα 8 καναλιών. Ο βέλτιστος αλγόριθμος υλοποιήθηκε σε ηλεκτρονικό κύκλωμα ενός FPGA. Ο αλγόριθμος λειτουργεί μέχρι τα 80 MHz, και χρειάζεται 30 κύκλους ρολογιού για να ολοκληρώσει τους υπολογισμούς και να δώσει το αποτέλεσμα στην έξοδο. Το FPGA αναπτύχθηκε για να χρησιμοποιηθεί σε ένα γενικότερο ηλεκτρονικό σύστημα συλλογής δεδομένων από ανιχνευτή αισθητήρων πυριτίου σε πείραμα Φυσικής Υψηλών Ενεργειών.

Abstract

The subject of the present MSc thesis is the development of an electronic circuit in an FPGA in order to remove the pedestals and the common noise of digitized signals originated from Silicon microstrip sensors. The signals from the silicon sensors are amplified – shaped in an analog FE (Front End) chip and driven into the analog to digital converters (ADCs). The digital data from the ADCs are driven into digital signal processing units (FPGAs) in order to remove the background and the common noise. In this thesis it has been considered that each subsystem sensor-FE includes 32 channels. Algorithms have been studied using Monte Carlo techniques in order to calculate and remove the pedestals and the common noise. It was assumed that the pedestal is different for each channel and that the noise is the same for groups of 8 neighbor channels. The best algorithm has been implemented in an electronic circuit in an FPGA. The algorithm operates up to 80 MHz and takes 30 clock cycles to complete the calculations and output the result. This FPGA has been developed in order to be used into a DAQ system of a Silicon sensors based detector in a High Energy Physics experiment.

Ευχαριστίες

Αισθάνομαι την υποχρέωση να ευχαριστήσω τους ανθρώπους που κατά το χρονικό διάστημα εκπόνησης της διπλωματικής μου εργασίας η συμβολή τους υπήρξε σημαντική.

Τον επιβλέποντα Επίκουρο Καθηγητή Ν. Μάνθο, μέλος του Εργαστηρίου Φυσικής Υψηλών Ενεργειών του Πανεπιστημίου Ιωαννίνων, για την καθοδήγησή του και την επιστημονική και ηθική συμπαράσταση καθ' όλη τη διάρκεια του προγράμματος σπουδών μου.

Τον καθηγητή Φ. Τριάντη, διευθυντή του Εργαστηρίου Φυσικής Υψηλών Ενεργειών του Πανεπιστημίου Ιωαννίνων για το συντονισμό και τη μακρόχρονη συμπαράσταση στη διάρκεια της εργασίας και τους Επίκουρους Καθηγητές του ΕΦΥΕ , κ. Ι. Ευαγγέλου και Π. Κόκκα , για τις συμβουλές και την βοήθειά τους κατά τη διάρκεια της εκπόνησης της εργασίας.

Τους συναδέλφους μου κ.κ. Α. Αναστασίου, Α. Ασημίδη, Ν. Γριμάνη, Λ. Κοντομάρκο, Φ. Παπαστεφάνου, Κ. Προύσκα, Ν. Τζούλη, Ν. Τσαγκούρια για την πολύτιμη βοήθειά τους και συνεργασία τους.

Περιεχόμενα

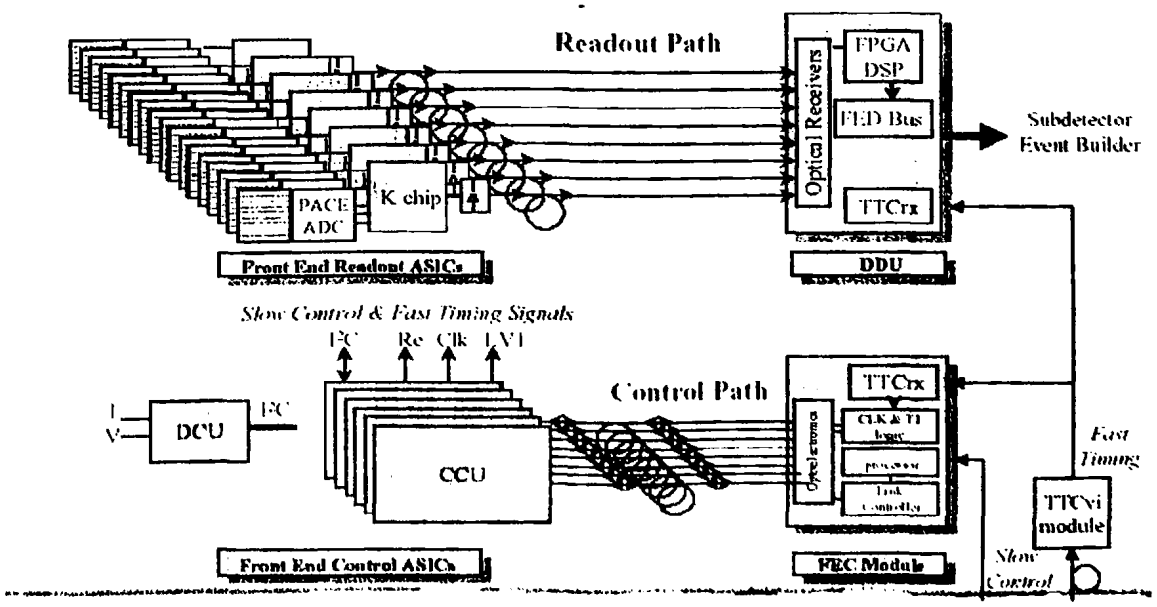
ΕΙΣΑΓΩΓΗ

Οι μικρολωριδιακοί αισθητήρες πυριτίου χρησιμοποιούνται ευρέως στα πειράματα φυσικής υψηλών ενεργειών διότι παρέχουν ακριβείς μετρήσεις της θέσης των φορτισμένων σωματιδίων ή των φωτονίων λόγω της πολύ καλής χωρικής διακριτικής ικανότητας που διαθέτουν. Έτσι στα πειράματα φυσικής υψηλών ενεργειών χρησιμοποιούνται σε περιβάλλον Υψηλής Ακτινοβολίας κοντά στο σημείο αλληλεπίδρασης.

Συγκεκριμένα στο υπό κατασκευή πείραμα CMS [1] που θα λειτουργήσει στο LHC ο κεντρικός ανιχνευτής θα αποτελείται από μικρολωριδιακούς αισθητήρες πυριτίου και σε συνδυασμό με αισθητήρες κουκίδας (pixel) με σκοπό την ανακατασκευή των τροχιών για όλα τα φορτισμένα σωμάτια. Για τον διαχωρισμό των φωτονίων που προέρχονται από τα σωμάτια π^0 από τα φωτόνια που παράγονται κοντά στο σημείο αλληλεπίδρασης pp θα χρησιμοποιηθεί ένα ανιχνευτικό σύστημα (preshower) [2] που βασίζεται σε δυο στρώματα από υλικό με μεγάλο Z (μόλυβδος) κατάλληλο για την δημιουργία καταιγισμών (showers) e^+ e^- που δημιουργούνται από την αλληλεπίδραση των γ και δυο στρώματα μικρολωριδιακών ανιχνευτών πυριτίου μιας όψης [3] που είναι τοποθετημένα

πίσω από τα στρώματα του μολύβδου. Ο αριθμός των αισθητήρων πυριτίου στο Preshower είναι 4288 και καθένας από αυτούς έχει 32 κανάλια οπότε ο συνολικός αριθμός των καναλιών φτάνει τα 137.216. Το μήκος των λωρίδων είναι 61 mm και το πλάτος τους περίπου 1.9 mm. Κάθε κανάλι του Preshower διαβάζεται από έναν ενισχυτή φορτίου που ακολουθείται από έναν μορφοποιητή (PACE) [4]. Η έξοδος του μορφοποιητή δειγματοληπτείται με τη συχνότητα των 40 MHz και τα δεδομένα αποθηκεύονται σε κελιά αναλογικής μνήμης. Στη συνέχεια τα αναλογικά δεδομένα πολυπλέκονται και οδηγούνται στους ADCs έναν για κάθε 32 κανάλια όπου μετατρέπονται σε αριθμό και κατόπιν στα Κ ολοκληρωμένα στα οποία μετά από πολυπλεξία μεταδίδονται μέσω μιας οπτικής σύνδεσης υψηλής ταχύτητας (Readout Path) στη μονάδα FED (Front End Driver). Εκεί ένα ηλεκτρονικό κύκλωμα που είναι αναπτυγμένο σε ένα ολοκληρωμένο (FPGA) αναλαμβάνει να πραγματοποιήσει την αφαίρεση του υποβάθρου και του κοινού θορύβου λόγω των ηλεκτρονικών. [5]

Στο διάγραμμα 1 φαίνεται ένα γενικό διάγραμμα του εν λόγω συστήματος.



Διάγραμμα 1: Preshower Readout & Control Architecture

Το υπόβαθρο του κάθε καναλιού (Pedestal) ορίζεται ως η μέση τιμή του ψηφιοποιημένου σήματος όταν δεν υπάρχει κάποιο πραγματικό σήμα στον ανιχνευτή. Επειδή χρησιμοποιούνται αναλογικά ηλεκτρονικά αυτό έχει ως αποτέλεσμα την πρόσθεση ενός επιπέδου τάσης για το κάθε κανάλι (DC shift). Αυτό προκαλεί την μετατόπιση του μηδενικού σημείου σε κάποια τιμή τάσης η οποία ονομάζεται υπόβαθρο. Συνήθως οφείλεται στην μη ύπαρξη τέλειας κοινής γείωσης καθώς επίσης εξαρτάται και από τη θερμοκρασία. Επίσης υπεισέρχεται μια επιπλέον μετατόπιση από τον ίδιο τον μετατροπέα αναλογικού σε ψηφιακό σήμα (ADC) η οποία όμως είναι κοινή και για τα 32 κανάλια. Επίσης προσδίδεται μια μετατόπιση και από τον κοινό θόρυβο (Common noise). Αυτή οφείλεται κυρίως στη τεχνολογία που χρησιμοποιείται για τη κατασκευή του wafer του ολοκληρωμένου FE. Δηλαδή ακόμα και στο ίδιο wafer υπάρχει ανομοιογένεια με αποτέλεσμα ομάδες γειτονικών καναλιών να συμπεριφέρονται διαφορετικά από ομάδες καναλιών που βρίσκονται σε απόσταση μεταξύ τους. Για τον παραπάνω λόγο, επιλέχθηκαν ομάδες των 8 καναλιών για τα οποία θεωρήθηκε ότι έχουν τον ίδιο κοινό θόρυβο. [6]

Στην παρούσα διπλωματική εργασία μελετώνται αλγόριθμοι για τον προσδιορισμό του κοινού θορύβου που περιέχεται μαζί με το υπόβαθρο στα κανάλια εισόδου. Στον κάθε μετατροπέα αναλογικού σε ψηφιακό ADC οδηγούνται 32 κανάλια και έχουμε 4 μετατροπείς συνολικά. Επίσης τα 32 κανάλια είναι ομαδοποιημένα ανά οκτάδες και με την υλοποίηση του βέλτιστου αλγόριθμου που έγινε λαμβάνεται υπόψη και η μη γραμμικότητα της αναλογικής μνήμης. Πιο αναλυτικά στο 1° κεφάλαιο περιγράφονται οι αλγόριθμοι που χρησιμοποιήθηκαν και στο 2° κεφάλαιο η προσομοίωση του αλγόριθμου που επιλέχθηκε στο λογισμικό πακέτο Matlab με τα αποτελέσματα της. Ακολούθως στο 3° κεφάλαιο γίνεται εκτενής αναφορά στο προγραμματισμό του ολοκληρωμένου μέσω του πακέτου λογισμικού XILINX και στη προσομοίωση του. Το 4° κεφάλαιο αναφέρεται στην αναπτυξιακή

κάρτα πάνω στην οποία προγραμματίστηκε και έγιναν οι έλεγχοι του ολοκληρωμένου και τέλος στο $5^{ο}$ κεφάλαιο γίνεται η αποτίμηση των αποτελεσμάτων και η σύγκρισή τους με τα αποτελέσματα του Matlab.

ΚΕΦΑΛΑΙΟ 1
Αλγόριθμοι.

Όπως αναφέρθηκε στο προηγούμενο κεφάλαιο, εκτός από το πραγματικό σήμα που προέρχεται από τους αισθητήρες πυριτίου έχουμε και κάποια επίπεδα θορύβου τα οποία είναι αποτέλεσμα του υπόβαθρου και του κοινού θορύβου γειτονικών καναλιών. Τα επίπεδα θορύβου πρέπει να αφαιρεθούν ώστε να μετρηθεί με ακρίβεια το σήμα των καναλιών που αντιστοιχούν στις μικρολωρίδες του αισθητήρα από τις οποίες πέρασε σωμάτιο ή φωτόνιο. Στην παρούσα ανάπτυξη μελετήθηκαν δυο αλγόριθμοι για την αφαίρεση του θορύβου εκ των οποίων τελικά επιλέχτηκε ο δεύτερος. Αξίζει να σημειωθεί ότι ο πρώτος αλγόριθμος είναι υποσύνολο του δεύτερου αλγόριθμου. Επίσης θεωρήθηκε ότι ο κοινός θόρυβος (common noise) είναι ο ίδιος για έναν αριθμό γειτονικών καναλιών και έγινε ομαδοποίηση των καναλιών ανά 8άδες.

1.1 1^{ος} αλγόριθμος.

Πιο αναλυτικά στον 1° αλγόριθμο θεωρήθηκε ότι οι τιμές τόσο του κοινού θορύβου (Common noise) όσο και του υποβάθρου (Pedestals) έχουν υπολογιστεί εκ των προτέρων (με offline αλγόριθμους). Το πλήθος των τιμών του κοινού θορύβου είναι 16 (κοινός θόρυβος ανά οκτάδα καναλιών) και το πλήθος των υποβάθρων είναι 128, όσα και ο αριθμός των καναλιών. Οι τιμές αυτές είναι αποθηκευμένες σε μνήμες 16 x 6 bits και 128 x 8 bit αντίστοιχα. Με την άφιξη στην είσοδο του FPGA των σημάτων από τους ADCs , για κάθε οκτάδα τιμών αφαιρείται ο κοινός θόρυβος και το αντίστοιχο υπόβαθρο και το καθαρό σήμα (ψηφιοποιημένο) βγαίνει στην έξοδο του FPGA.

1.2 2ος αλγόριθμος.

Σε ότι αφορά τον 2° αλγόριθμο ο οποίος και υλοποιήθηκε τελικά και αναφέρεται ως αλγόριθμος από εδώ και στο εξής μόνο οι τιμές του υποβάθρου (Pedestals) έχουν υπολογιστεί εκ των προτέρων και είναι αποθηκευμένες σε μνήμη 128 x 8 bit. Ας σημειωθεί ότι σε αυτή τη περίπτωση λαμβάνεται υπόψη και η μη γραμμικότητα της αναλογικής μνήμης κατά την αφαίρεση του υποβάθρου. Αυτή οφείλεται στους πυκνωτές από τους οποίους αποτελείται το κάθε κανάλι. Ο υπολογισμός είτε έχει γίνει από πριν χωρίς την παρουσία σημάτων και για επαρκή αριθμό επαναλήψεων οπότε και παραμένουν σταθερές οι τιμές είτε υπολογίζονται κατά τον ίδιο τρόπο αλλά ανά δεδομένο χρονικό διάστημα (π.χ. ανά μια ώρα) οπότε και αλλάζουν.

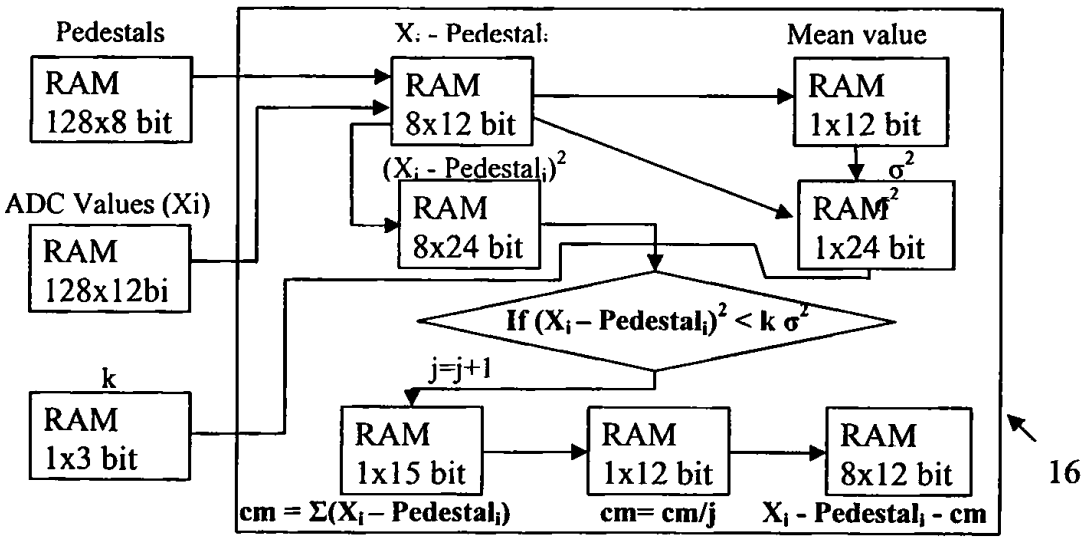
Οι τιμές του κοινού θορύβου (Common Noise) υπολογίζονται σε πραγματικό χρόνο σύμφωνα με τον αλγόριθμο παρακάτω. Με την άφιξη των τιμών των καναλιών στο FPGA γίνεται αφαίρεση του υποβάθρου (που έχει υπολογιστεί από πριν για κάθε κανάλι) και στη συνέχεια ακολουθεί ο υπολογισμός του κοινού θορύβου. [7] Για κάθε οκτάδα τιμών-καναλιών υπολογίζεται η μέση τιμή της οκτάδας (ADC_i – $pedestal_i$) , όπου i το κάθε

κανάλι, και στη συνέχεια και η τυπική απόκλιση από τη μέση τιμή σ, όπου $\sigma=\{\Sigma(ADC_i - \overline{ADC})^2]/N(N-1)\}^{1/2}$, με N τον αριθμό των καναλιών με κοινό θόρυβο. Στην περίπτωσή μας το N=8. Ακολούθως συγκρίνεται το (ADC_i – $pedestal_i$) με το kσ , όπου το k είναι ένας συντελεστής ο οποίος εξαρτάται από τα σχετικά μεγέθη του κοινού θορύβου, υπόβαθρου και τιμής πραγματικού σήματος. Το k ύστερα από δοκιμές κατά τη προσομοίωση τέθηκε ίσο με τη μονάδα. Εάν ισχύει (ADC_i – $pedestal_i$) < kσ τότε τα αντίστοιχα (ADC_i – $pedestal_i$) των συγκεκριμένων καναλιών χρησιμοποιούνται στον υπολογισμό του κοινού θορύβου της συγκεκριμένης οκτάδας ως εξής:

Common $noise_j = \Sigma (x_{i,j} - pedestal_{i,j})/m$, όπου το άθροισμα πραγματοποιείται μόνο στα κανάλια m για τα οποία ισχύει $x_{i,j} - pedestal_{i,j} < k\sigma_j$.

1.2.1 Γενικό διάγραμμα

Στο διάγραμμα 2 φαίνονται αναλυτικά τα βήματα τα οποία ακολουθεί ο αλγόριθμος προκειμένου να υπολογιστεί ο κοινός θόρυβος. Το συγκεκριμένο διάγραμμα αναφέρεται στον υπολογισμό του κοινού θορύβου από μια οκτάδα τιμών-καναλιών.



Διάγραμμα 2 : Γενικό διάγραμμα του αλγόριθμου.

Οι τιμές των υποβάθρων επιλέχθηκαν 8 bits καθότι θεωρήθηκε ότι η μέση τιμή τους θα κυμαίνεται από 0 – 255. Αντίστοιχα για τις τιμές ADC επιλέχθηκαν 12 bits διότι ο ADC που θεωρήθηκε είναι ο AD9042 της Analog Devices. Με βάση αυτές τις αρχικές συνθήκες το μέγεθος των υπολοίπων μεταβλητών ορίζεται από τις επιμέρους πράξεις.

ΚΕΦΑΛΑΙΟ 2
Προσομοίωση με το πακέτο λογισμικού Matlab.

Πριν από την υλοποίηση του αλγόριθμου στο FPGA κρίθηκε ως αναγκαίο να προηγηθεί μια off-line προσομοίωση ώστε να διαπιστωθούν τυχόν προβλήματα (αδυναμίες) στον αλγόριθμο και να διορθωθούν όπως επίσης και να προσδιοριστούν παράμετροι όπως το κ. Το λογισμικό πακέτο που επιλέχτηκε είναι το MATLAB 6 Release12 της Mathworks. [8]

2.1 Πρόγραμμα προσομοίωσης

Το πρόγραμμα αποτελείται από δυο μέρη. Στο πρώτο μέρος προσομοιώνονται τα σήματα (ψηφιακά) στην είσοδο του FPGA όσο το δυνατόν πιο ρεαλιστικά ενώ στο δεύτερο μέρος προσομοιώνεται η διαδικασία υπολογισμού του κοινού θορύβου (common noise) και συγκρίνεται με τον αρχικό κοινό θόρυβο ο οποίος είχε εισαχθεί στο πρόγραμμα. Η προσομοίωση έγινε για 8 κανάλια και για 1000 επαναλήψεις. Τα κανάλια του ανιχνευτή υπενθυμίζεται ότι είναι 32, άρα θα έχουμε 4 οκτάδες. Οι μέσες τιμές που

υπολογίστηκαν παρακάτω όμως αφορούν την περίπτωση της μιας οκτάδας. Αναλυτικότερα στο πρώτο μέρος:

- Δημιουργείται ένας πίνακας **comon** (1000,1) στον οποίο τοποθετούνται 1000 τυχαίοι ακέραιοι αριθμοί προερχόμενοι από κανονική κατανομή (Gauss) με σ=30 ADC counts και κέντρο το 0 (Διάγραμμα 3). Επίσης δημιουργείται ένας πίνακας **commmonxy** (1000,8) στον οποίο τοποθετούνται 8000 τυχαίοι ακέραιοι αριθμοί προερχόμενοι από κανονική κατανομή (Gauss) με σ=3 ADC counts και μέσο το 0 (Διάγραμμα 4). Επειδή ανά οκτάδα ο κοινός θόρυβος είναι περίπου ο ίδιος, ο τελευταίος πίνακας χρησιμοποιείται για να προσδίδει στο κοινό θόρυβο μια μικρή διαφορά μεταξύ των γειτονικών καναλιών. Επομένως ο κοινός θόρυβος (6 bits) που εισάγεται είναι ο **common2(i,j) = comon(i) + commmonxy(i,j)** (Διάγραμμα 5).
- Για κάθε τιμή ADC έχουμε και διαφορετικό υπόβαθρο. Επομένως δημιουργείται ένας πίνακας **pedestals** (1000,8) 8000 τυχαίων ακέραιων αριθμών προερχόμενων από κανονική κατανομή (Gauss) με σ=70 ADC counts και μέσο το 128. Οι αριθμοί αυτοί (8 bits) αναπαριστούν τα υπόβαθρα των καναλιών (Διάγραμμα 6).
- Για την επιλογή του πλήθους των καναλιών των πραγματικών σημάτων που παρήχθησαν στον ανιχνευτή χρησιμοποιείται μια κατανομή poisson με μέση τιμή 1 (Διάγραμμα 7). Με αυτό τον τρόπο η μεταβλητή **signal** περιέχει τον αριθμό των καναλιών που έχουν πραγματικό σήμα. Αναφορικά με τις τιμές των ADCs δημιουργείται ένας πίνακας **channel** (1,8) ο οποίος γεμίζει με τυχαίους αριθμούς από 0 – 1 που προέρχονται από μια ομογενή κατανομή, ακολούθως δημιουργείται ένας πίνακας **adcs(i,j)** ο οποίος προκύπτει από τον πολλαπλασιασμό των στοιχείων του πίνακα **channel(i,j)** με τον αριθμό 4095 (θεωρείται ότι οι τιμές των ADCs είναι 12 bits).
- Η αρχική τιμή επομένως που έρχεται στην είσοδο του FPGA είναι: **value(i,j) = pedestals(i,j) + common2(i,j) + adcs(i,j)**.

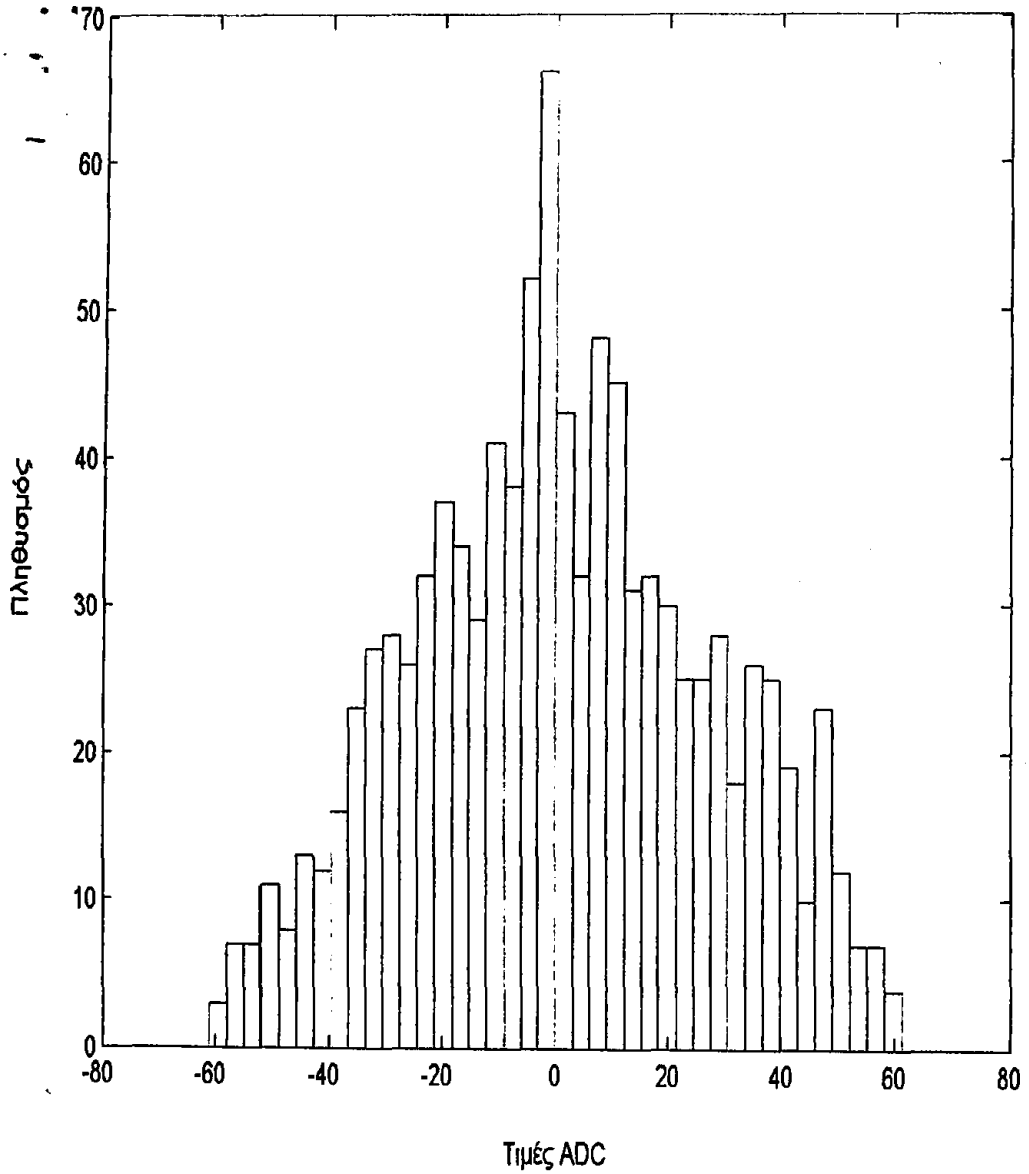
Στο δεύτερο και σημαντικότερο μέρος του προγράμματος προσομοίωσης αναλυτικά:

- Με την ανανέωση του ψηφιοποιημένου σήματος **value(i,j)**, αφαιρείται από αυτό το αντίστοιχο pedestal(i,j) και το αποτέλεσμα αποθηκεύεται στον πίνακα **clean(i,j)**. Στη συνέχεια πρέπει να εντοπισθούν ποιο ή ποια από τα οκτώ γειτονικά κανάλια έχει πραγματικό σήμα και ποιο έχει μόνο θόρυβο. Για το λόγο αυτό υπολογίζεται η μέση τιμή **mes(i)** και η εκάστοτε διαφορά της τιμής **clean(i,j)** από τη μέση τιμή **mes(i)** η οποία και αποθηκεύεται στον πίνακα **diafora** (1000,8). Ακολούθως αθροίζονται τα στοιχεία κάθε γραμμής του πίνακα **diafora(i,j)** και υπολογίζεται η τυπική απόκλιση από τη μέση τιμή **finalsigma(i)**.
- Στη συνέχεια ελέγχεται εάν η εκάστοτε τιμή του σήματος του καναλιού clean(i,j) είναι μικρότερη ή όχι από το b, όπου b=κσ και στην προκειμένη περίπτωση έχει επιλεχθεί να είναι ίσο με το finalsigma. Στην περίπτωση που ισχύει η ανωτέρω συνθήκη τότε το σήμα του καναλιού είναι απλά θόρυβος. Έτσι η τιμή του συγκεκριμένου clean αποθηκεύεται στον πίνακα **exw** (1000,8) ενώ παράλληλα καταμετράται στον πίνακα **ne** (1000,1) ο αριθμός από τα οκτώ κανάλια για τα οποία ισχύει η παραπάνω συνθήκη. Επίσης προστίθενται τα στοιχεία της κάθε γραμμής του clean(i,j) και η τιμή τους αποθηκεύεται στον πίνακα **commononly(i)**. Με το πέρας της σύγκρισης κάθε οκτάδας, τα στοιχεία της εκάστοτε γραμμής του πίνακα exw(i,j) προστίθενται και αποθηκεύονται στον πίνακα **commonnoise(i)**.
- Στην περίπτωση κατά την οποία το ne έχει την τιμή 0, επειδή αποκλείεται η πιθανότητα να έχουμε 8 σήματα, συμπεραίνεται ότι πρόκειται για θόρυβο οπότε γίνεται αντιστοίχιση στην περίπτωση όπου ne=8. Ο κοινός θόρυβος επομένως υπολογίζεται ως **cmm = commonnoise/ne** ενώ για **ne=0** ο κοινός θόρυβος ισούται **cmm= commononly/8**.

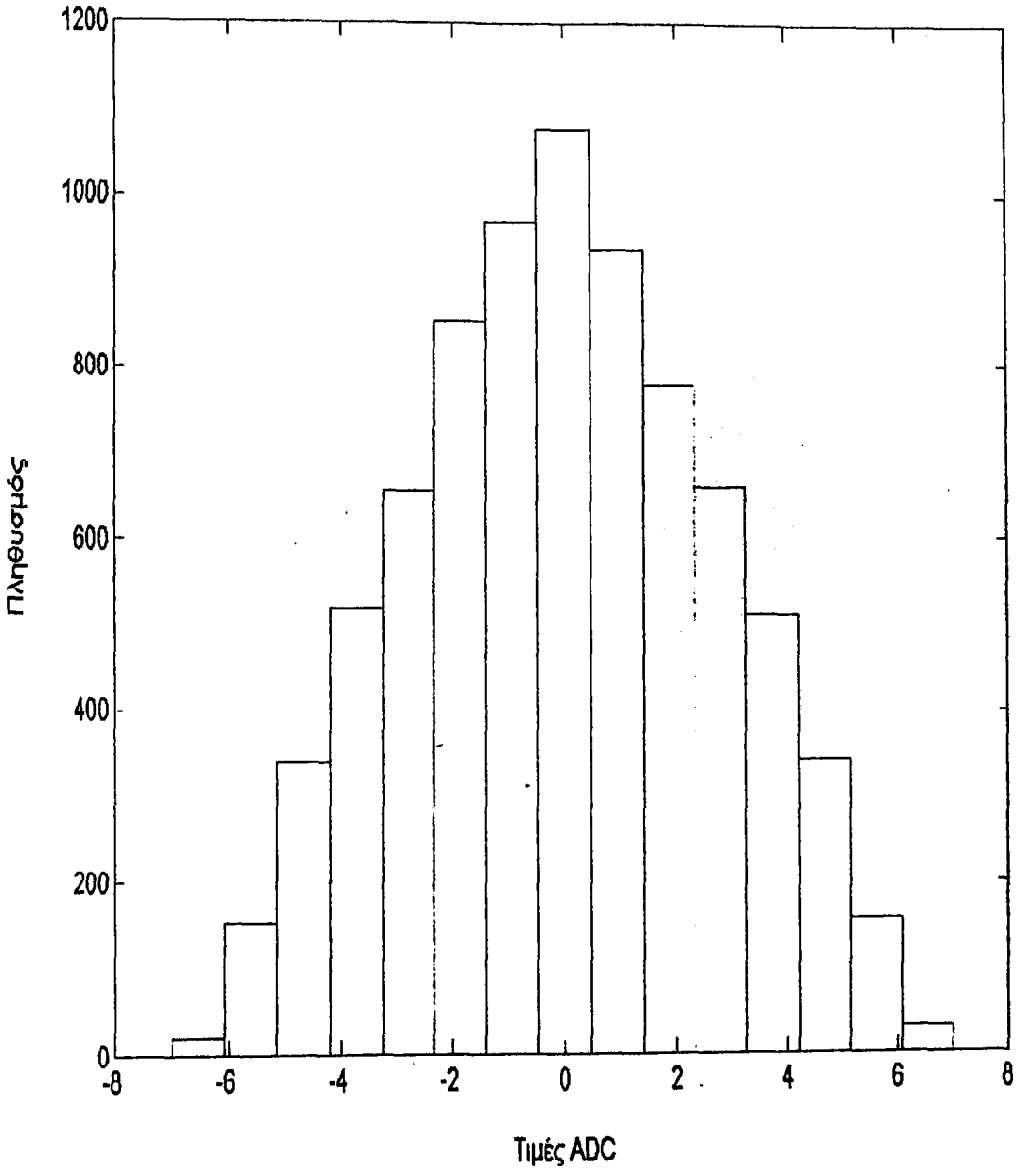
Αφού υπολογιστεί ο κοινός θόρυβος **cmm(i,j)**, βρίσκεται η διαφορά μεταξύ του **cmm(i,j)** και του κοινού θορύβου **common2(i,j)** που είναι είσοδος στο πρόγραμμα προσομοίωσης: **difference(i,j) = common2(i,j) – cmm(i,j).**

2.1.1 Διαγράμματα σημάτων εισόδου

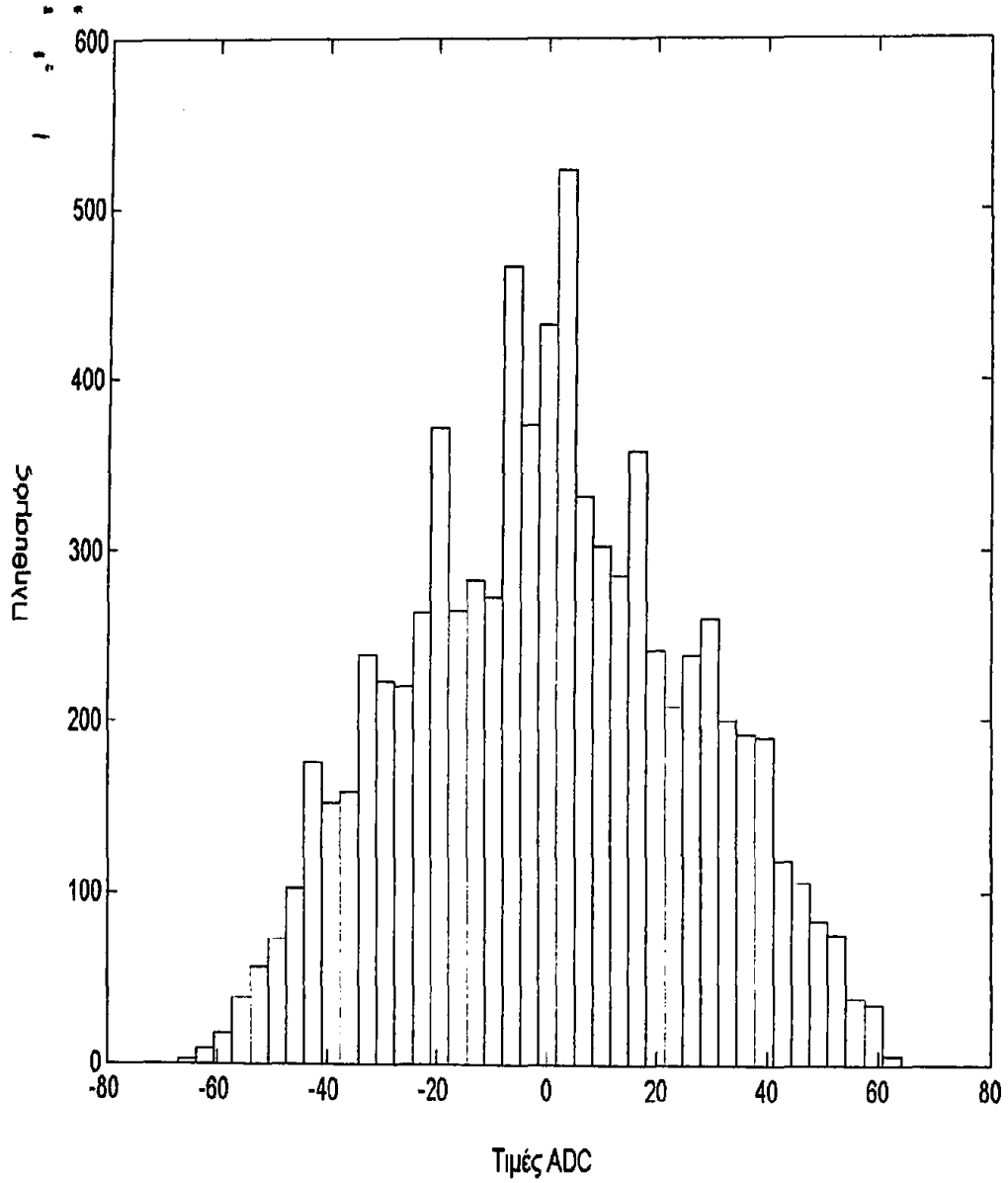
Στη συνέχεια ακολουθούν τα διαγράμματα των σημάτων που δημιουργήθηκαν στο MATLAB και χρησιμοποιήθηκαν ως είσοδος ώστε να δοκιμαστεί η σωστή λειτουργία του αλγόριθμου.



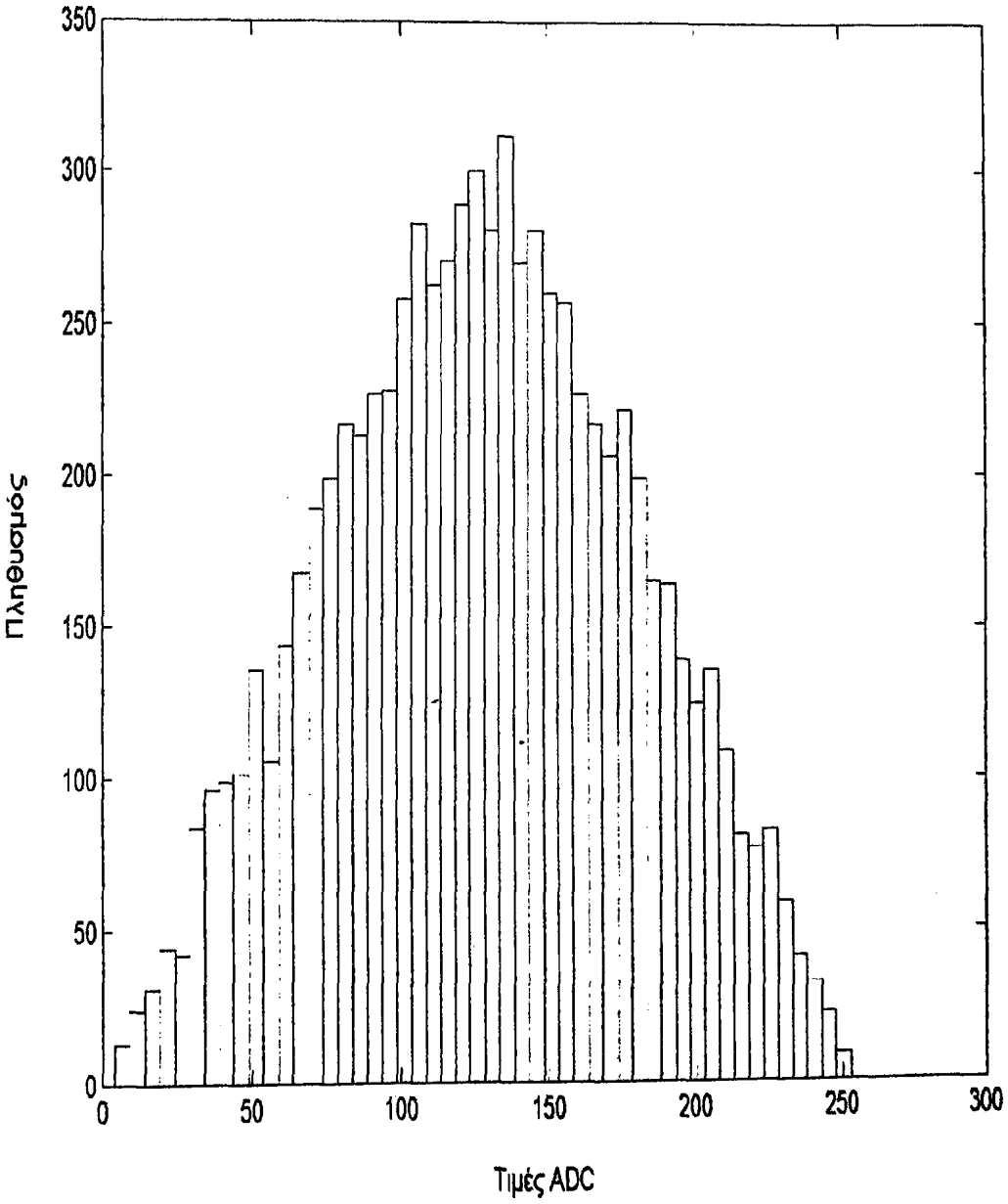
Διάγραμμα 3: Η κανονική κατανομή του κοινού θορύβου **comon** που εισάγεται στο πρόγραμμα προσομοίωσης του αλγόριθμου.



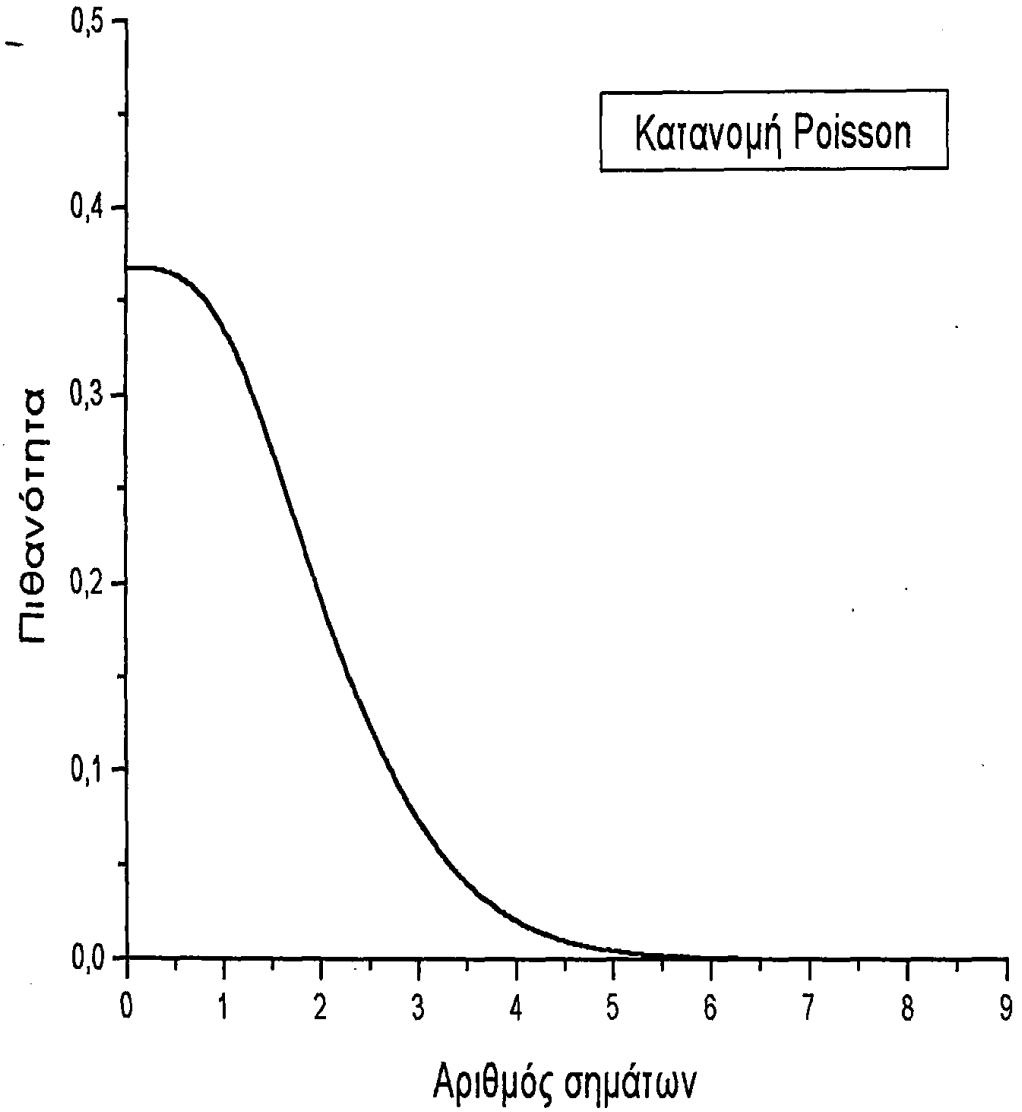
Διάγραμμα 4 : Η κανονική κατανομή του κοινού θορύβου **commmonxy** μεταξύ των καναλιών που εισάγεται στο πρόγραμμα προσομοίωσης του αλγόριθμου.



Διάγραμμα 5 : Η κανονική κατανομή του θορύβου **commmon2** που εισάγεται στο πρόγραμμα προσομοίωσης του αλγόριθμου.



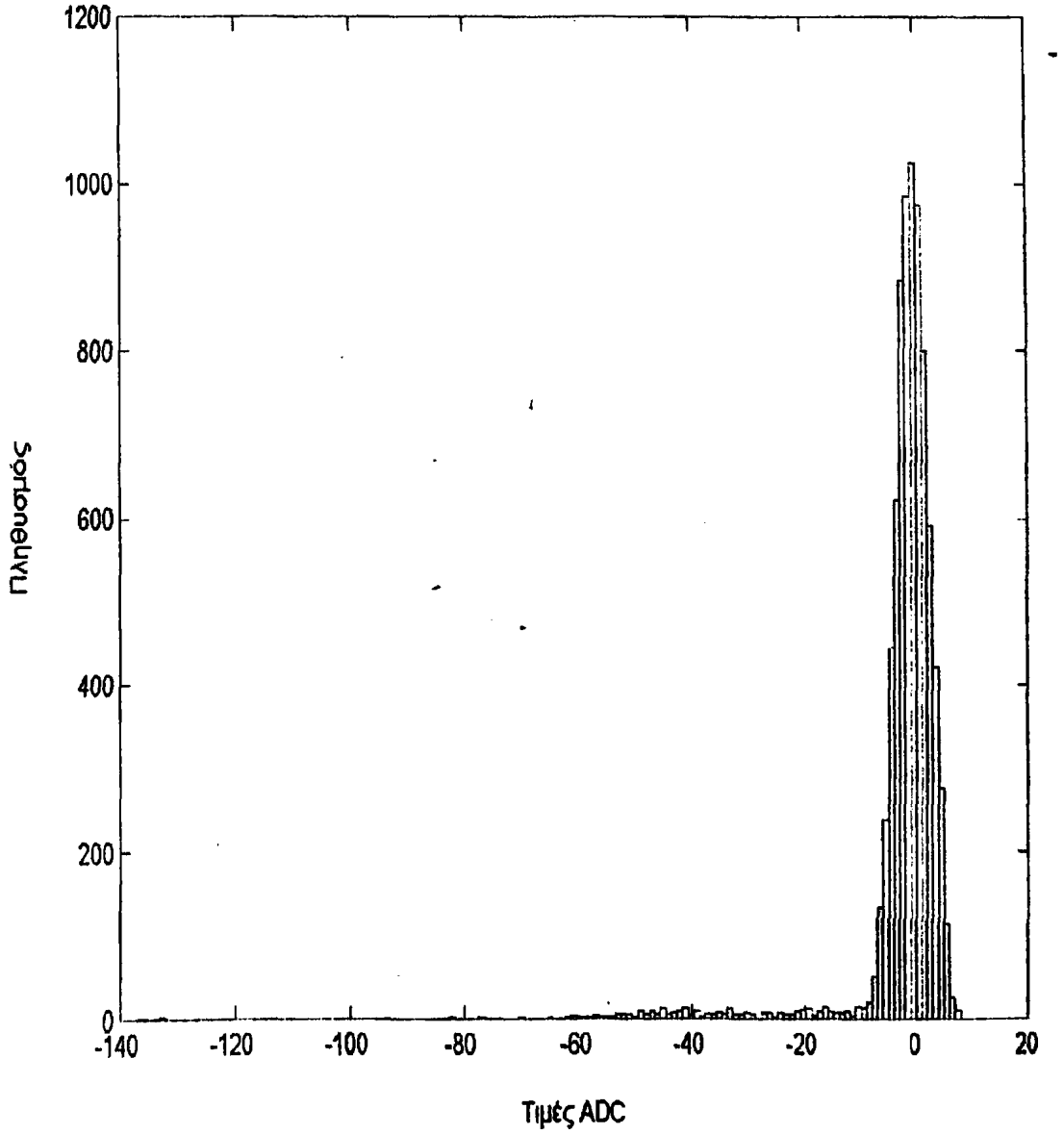
Διάγραμμα 6 : Η κανονική κατανομή των υποβάθρων **Pedestals** που εισάγονται στο πρόγραμμα προσομοίωσης του αλγόριθμου.



Διάγραμμα 7 : Η poisson κατανομή του αριθμού των καναλιών που έχουν σήμα όπως εισάγονται στο πρόγραμμα προσομοίωσης του αλγόριθμου [9].

2.2 *Αποτελέσματα προσομοίωσης*

Η διαφορά μεταξύ του υπολογισμένου από τον αλγόριθμο κοινού θορύβου **cmm** από τον κοινό θόρυβο που εισάγεται στο πρόγραμμα προσομοίωσης φαίνεται στο ιστόγραμμα του διαγράμματος 8.



Διάγραμμα 8 : Η κανονική κατανομή της διαφοράς **Difference** από το αποτέλεσμα της προσομοίωσης του αλγόριθμου.

Το ιστόγραμμα ακολουθεί κανονική κατανομή όπως αναμενόταν με μέση τιμή το μηδέν και με σ= 3 ADC counts (Διάγραμμα 8) που αντιστοιχεί σε 0,3 MIP (1 MIP αντιστοιχεί σε φορτίο 24.000 e^-). Το 96,3% των περιπτώσεων της εν λόγω διαφοράς διαφέρει λιγότερο από 10 ADC counts. Αξίζει να σχολιαστεί ότι η κανονική κατανομή του διαγράμματος 8 έχει μια ενίσχυση προς τις αρνητικές τιμές για ένα μικρό ποσοστό των γεγονότων. Δεν βρέθηκε κάποια συγκεκριμένη κατηγορία γεγονότων που να προκαλεί αυτή την ενίσχυση και έτσι δεν έγινε κάποια διόρθωση στον αλγόριθμο.

ΚΕΦΑΛΑΙΟ 3
Προγραμματισμός στο πακέτο λογισμικού XILINX Foundation.

3.1 Ανάπτυξη κυκλώματος.

Το κύκλωμα του FPGA έγινε για το VIRTEX-E XCV100E PQ240 speed –8 της XILINX το οποίο περιέχει 128.236 πύλες συστήματος. Στην ανάπτυξη του κυκλώματος χρησιμοποιήθηκε ευρέως ο Core Generator 3.1i, εργαλείο που είναι ενσωματωμένο στο πακέτο λογισμικού XILINX Foundation [10].

Το κύκλωμα που αναπτύχθηκε φαίνεται στο διάγραμμα 9 και σε αναγνώσιμη μεγέθυνση στο διάγραμμα 10. Τα στοιχεία του κυκλώματος δημιουργήθηκαν με το Core Generator του XILINX Foundation 3.1i (Παράρτημα Α). Όπως προαναφέρθηκε το κύκλωμα του FPGA που αναπτύχθηκε λειτουργεί για 8 από τα συνολικά 128 κανάλια. Το κύκλωμα με μια απλή επανάληψή του κατά 16 φορές μπορεί να λειτουργήσει για τα 128 κανάλια. Το κύκλωμα που σχεδιάστηκε είναι σύγχρονο δηλαδή όλες οι υπομονάδες και τα στοιχεία του έχουν κοινό ρολόι και δουλεύουν σε

συγχρονισμό με αυτό. Το κύκλωμα έχει 2 ακροδέκτες εισόδου και 20 ακροδέκτες εξόδου. Από τους τελευταίους οι 8 χρησιμοποιούνται για τον έλεγχο του συστήματος και οι υπόλοιποι 12 είναι αυτοί που μας δίνουν το αποτέλεσμα (12 bits) του υπολογισμένου κοινού θορύβου. Οι τιμές των ADCs και του υποβάθρου είναι αποθηκευμένες εσωτερικά στο FPGA και είναι δυνατόν να αλλάζουν μέσω των εκτελέσιμων αρχείων ADCS_VALUES.coe και PEDES_VALUES.coe ακολουθώντας μια συγκεκριμένη διαδικασία (Παράρτημα ΣΤ). Ένας από τους ακροδέκτες εισόδου είναι το ρολόι του κυκλώματος το οποίο διανέμεται σε όλες τις υπομονάδες και ο άλλος ακροδέκτης εισόδου είναι το CLR (CLEAR) το οποίο μηδενίζει ή θέτει σε κάποια αρχική κατάσταση όλες τις μονάδες του κυκλώματος. Στις ακόλουθες παραγράφους παρατίθεται η διαδικασία λειτουργίας του κυκλώματος και λεπτομέρειες για τα χαρακτηριστικά των στοιχείων που τον αποτελούν.

3.1 Υπολογισμός μέσης τιμής.

Μόλις το CLEAR γίνει 0, τότε το σήμα ENABLE1C γίνεται 1 και ο απαριθμητής **U136** (counter10) ενεργοποιείται. Αυτός ο απαριθμητής ξεκινάει να μετράει από το 0 και σταματάει στο 7. Αποστολή του είναι να διευθυνσιοδοτήσει με το bus MONDAY τις μνήμες Single Port ADCS_VALUES και PEDES_VALUES. Οι μνήμες αυτές είναι των 8 x 13 bits και 8 x 9 bits αντίστοιχα. Περιέχουν 8 αριθμούς η καθεμία και επειδή οι αριθμοί που περιέχονται είναι προσημασμένοι για αυτό και είναι κατά 1 bit μεγαλύτεροι από αυτούς που χρησιμοποιήθηκαν στην προσομοίωση με το MATLAB. Την απενεργοποίησή του **U136** αναλαμβάνει το σήμα TELEIO το οποίο παράλληλα επηρεάζει την εκκίνηση του **U108** (ADRESSCHANGE2). Επίσης έχει γίνει χρήση κάποιων D Flip-Flop FDS (στην άνοδο του clear το flip-flop γίνεται set, το Q παίρνει δηλαδή τη τιμή 1) και FDR (στην άνοδο του clear το flip-flop γίνεται reset, το Q παίρνει δηλαδή τη τιμή 0) ώστε με τη

βοήθεια των σημάτων ENABLE1B, ENABLE1A, ENABLE1 να επιτευχθεί ο συγχρονισμός.

Καθώς με κάθε άνοδο του ρολογιού αδειάζουν οι μνήμες με τις τιμές των ADCs και των υποβάθρων, οι τιμές οδηγούνται στο **U137** (NEWSUB1) στο οποίο η τιμή του υποβάθρου αφαιρείται από την αντίστοιχη τιμή του ADC. Το αποτέλεσμα της αφαίρεσης B1A[12:0] οδηγείται σε τέσσερις υπομονάδες. Στο **U139** (NEW_ATHROISMA_3), στο **U97** (CLEAN_FIFO2BB), στο **U140** (MULTI2) και στο **U95** (CLEAN_FIFO2BB). Το **U139** είναι ένας αθροιστής (accumulator) ο οποίος υπολογίζει το άθροισμα των οκτώ τιμών Q[15:0]. Η μέση τιμή που μένει να υπολογιστεί δεν είναι τίποτα άλλο από το Q[15:3] που υπολογίζεται απλά αγνοώντας τα 3 λιγότερο σημαντικά ψηφία. Τα **U97** και **U95** είναι μνήμες Dual Port οι οποίες χρησιμοποιούνται ώστε τη δεδομένη χρονική στιγμή να μας παρέχουν τις τιμές B1A[12:0]. Το **U140** είναι ένας πολλαπλασιαστής 13 bits ο οποίος μας δίνει στην έξοδό του X-PED_TETRAG[25:0] το τετράγωνο του εκάστοτε B1A[12:0].

Στη συνέχεια το bus Q[15:3] οδηγείται μαζί με το bus DDD[12:0] στο **U98** (SUBSTRACTOR3) ο οποίος ενεργοποιείται από ένα Flip-flop FDR (CE_98_OUT). Το **U98** αφαιρεί την εκάστοτε τιμή του B1A[12:0] από τη μέση τιμή <B1A> και το αποτέλεσμα εξάγεται στο δίαυλο SUBOUT3A[12:0]. Το τελευταίο οδηγείται στο U110 (MULTI2) ο οποίος το υψώνει στο τετράγωνο και το σήμα MULTIOUT[25:0] καταλήγει στον αθροιστή **U121** (ACUM24). Ο αθροιστής **U121** όσο είναι ενεργοποιημένος αθροίζει τις τιμές του MULTIOUT[25:0] και το αποτέλεσμα IDEA[28:0] οδηγείται στο **U142** (CHANGER_FORM). Το **U121** γίνεται ανενεργό ύστερα από τον παλμό (Q_FDS_MULTI). Ο παλμός Q_FDS_MULTI προέρχεται από ένα Flip-flop FDS το οποίο αλλάζει κατάσταση όταν το σήμα (Q_THRESH0) γίνει 1. Το σήμα (Q_THRESH0) προέρχεται από τον μετρητή **U101** (COUNT_TO_14). Ο μετρητής δίνει 2 παλμούς, έναν όταν η καταμέτρηση φτάσει στο 12

(Q_THRESH0) και άλλον έναν μόλις φτάσει στο 14 (Q_THRESH1). Στη συνέχεια το **U142** απλά μετασχηματίζει τον αριθμό IDEA[28:0] στον S_READY_FOR_COMPARATOR[31:0].

Παράλληλα με την παραπάνω διαδικασία εκτελείται και μια άλλη σειρά από πράξεις. Το σήμα X-PED_TETRAG[25:0] οδηγείται στον πολλαπλασιαστή **U141** (MLT56) ο οποίος πολλαπλασιάζει τον εκάστοτε αριθμό στην είσοδό του με το 56 και μας δίνει το αποτέλεσμα στο bus OUT_56_MULT[31:0]. Ο αριθμός 56 προκύπτει από την σχέση της τυπικής απόκλισης $\sigma=\sqrt{\frac{\Sigma(xi-\bar{x})2}{N(N-1)}}$. Στην περίπτωση του συγκεκριμένου κυκλώματος N=8, και το υπόριζο είναι 7*8=56 . Η σχέση έχει υψωθεί στο τετράγωνο για να απαλειφθεί το ριζικό. Ο δίαυλος OUT_56_MULT[31:0] οδηγείται στο **U104** (DUALL5BB) που είναι μια μνήμη Dual Port με έξοδο τον δίαυλο IS_IT_WRITE[31:0]. Τη διευθυνσιοδότηση για την εγγραφή της αναλαμβάνει το **U102** (COUNT_TO_8BB), ένας μετρητής 3 bits ο οποίος ξεκινά να μετράει από το 1 μόλις το CLR γίνει 0. Σε ότι αφορά την ανάγνωσή της, αυτό το αναλαμβάνει ένας άλλος μετρητής ο **U125** (COUNTER8) ο οποίος είναι και αυτός 3 bits και ξεκινάει το μέτρημα από το 1 όταν το σήμα POINT1 γίνει 1. Το POINT1 προέρχεται από τον μετρητή **U106** (COUNT_TO_11) ο οποίος ξεκινάει να μετράει από το 0 και δίνει δυο παλμούς, έναν όταν φτάσει στο 13 (POINT0) και τον άλλο στο 14 (POINT1) οπότε και σταματάει κρατώντας το τελευταίο σήμα στο 1. Έτσι λοιπόν οι δίαυλοι IS_READY_FOR_COMPARATOR[31:0] και IS_IT_WRITE[31:0] φτάνουν στον συγκριτή **U115** (COMPARATOR_SXTETR) και ενεργοποιείται από το σήμα QTHRESH1 του **U101**. Το **U115** ελέγχει τη συνθήκη, αν το IS_IT_WRITE[31:0] είναι μικρότερο από το IS_READY_FOR_COMPARATOR[31:0] και στην περίπτωση που ισχύει το σήμα LAST_CHECK_A γίνεται 1.

Το σήμα LAST_CHECK_A οδηγείται στους μετρητές **U124** (KAPA) και **U143** (ATHROISMA4B). Ο μετρητής **U124** μετράει πόσες φορές ισχύει η συνθήκη και το αποτέλεσμα KAPA_OUT[2:0] καταλήγει στο **U126** (CHOOSER). Την ενεργοποίηση του **U124** αναλαμβάνει το σήμα CE_U60 και την απενεργοποίηση το QTHRESH0 του **U96** (METRITIS_A) ο οποίος είναι ένας μετρητής που ξεκινάει να μετράει από το 0 και δίνει έναν 1 μόλις φτάσει στον αριθμό 9 (QTHRESH0). Όπως και το **U124** έτσι και το **U96** ενεργοποιείται από το CE_U60 το οποίο εξαρτάται από το σήμα QTHRESH1 του **U101**.

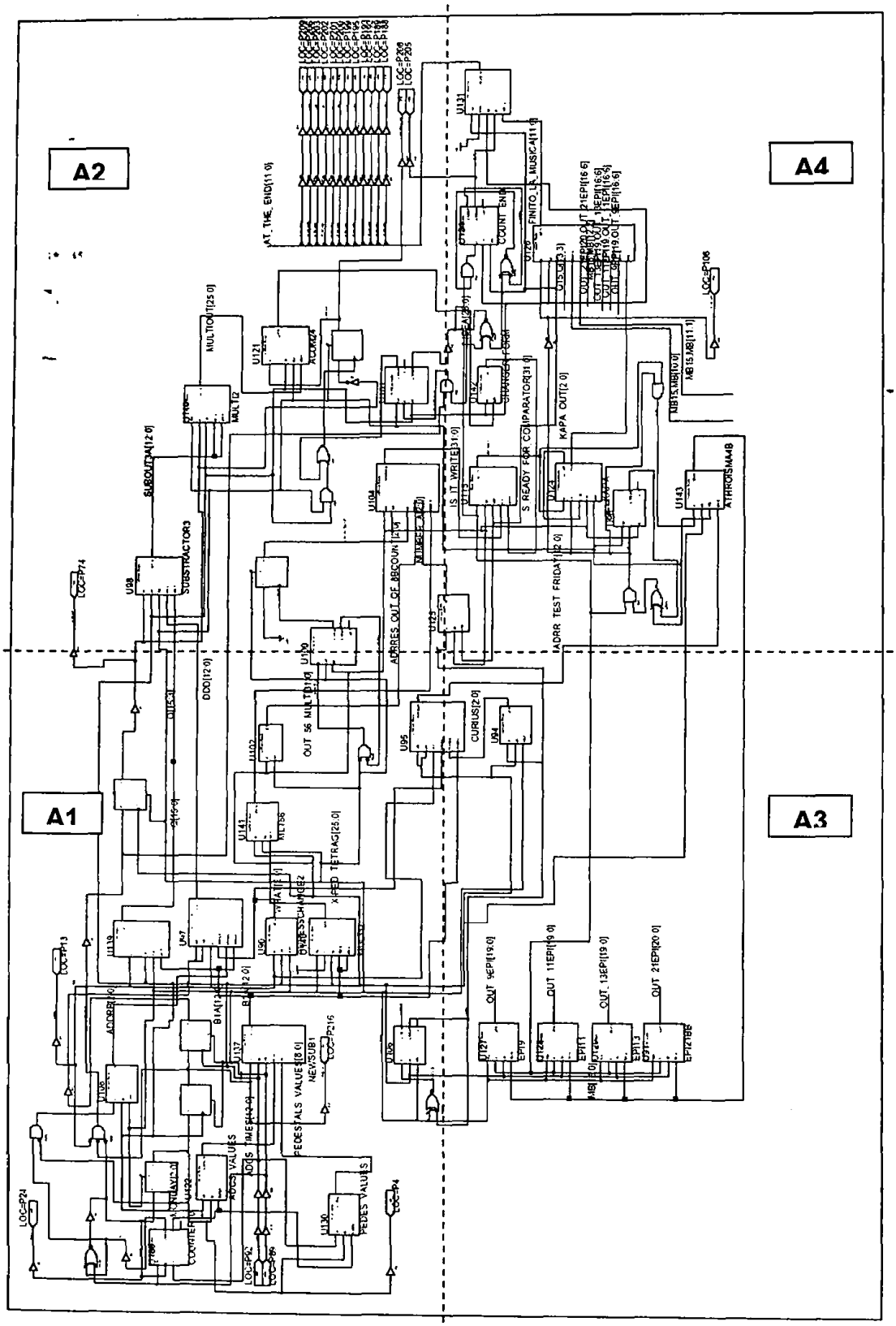
Ο μετρητής **U143** ενεργοποιείται από το σήμα CE_U66 και αθροίζει τα σήματα B1A για τα οποία ισχύει η προαναφερόμενη συνθήκη. Τα σήματα ADRR_TEST_FRIDAY[12:0] έρχονται την κατάλληλη χρονική στιγμή από την Dual Port μνήμη **U95** η οποία διευθυνσιοδοτείται από τον μετρητή **U94** (counter6) μέσω του διαύλου CURIUS[2:0]. Ο συγκεκριμένος μετρητής ξεκινάει την καταμέτρηση από το 0 και ενεργοποιείται από το σήμα POINT1 του **U106**.

Το αποτέλεσμα από το **U143** MB[15:0] οδηγείται ταυτόχρονα σε τέσσερις πολλαπλασιαστές, τους **U127** (EPI9), **U128** (EPI11), **U129** (EPI13) και **U91** (EPI21BB). Ο καθένας από αυτούς πολλαπλασιάζει το MB[15:0] με το 9, 11, 13 και 21 αντίστοιχα ενώ στην έξοδό τους παίρνουμε το εκάστοτε σήμα διαιρεμένο με το 64. Έτσι έχουμε τα σήματα OUT_9EPI[19:0], OUT_11EPI[19:0], OUT_13EPI[19:0], OUT_21EPI[20:0]. Η παραπάνω διαδικασία έγινε με σκοπό να διαιρέσουμε το σήμα MB[15:0] με το 7, 6, 5 και 3 αντίστοιχα. Έτσι για παράδειγμα το MB[15:0]/7 μπορεί να υπολογιστεί με προσέγγιση ως 9*MB[15:0]/64 ώστε μετά τον πολλαπλασιασμό με το 9 αγνοούνται τα τελευταία 6 λιγότερο σημαντικά ψηφία γεγονός που αντιστοιχεί σε διαίρεση δια 64. Οι υπόλοιπες διαιρέσεις MB[15:0]/5, MB[15:0]/6 και

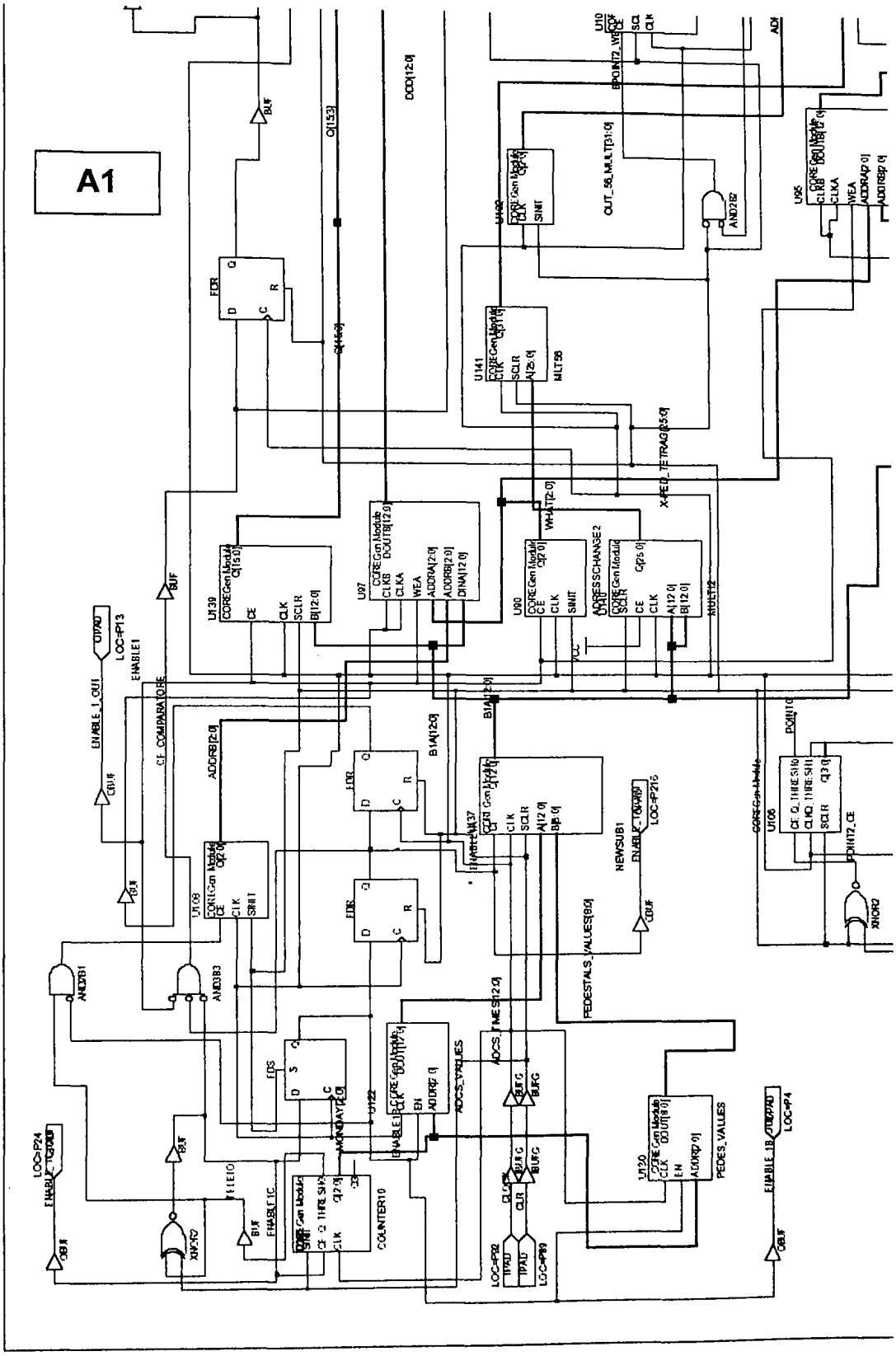
MB[15:0]/3 αντίστοιχα προσεγγίζονται με 13*MB[15:0]/64, 11*MB[15:0]/64 και 21*MB[15:0]/64.

Τέλος στο πολυπλέκτη **U126** καταλήγουν τα σήματα (Q15, Q[13:3]), (MB15, MB[10:0]), (MB15, MB[11:1]), (OUT_21EPI20, OUT_21EPI[16:6]), (MB15, MB[12:2]), (OUT_13EPI19, OUT_13EPI[16:6]), (OUT_11EPI19, OUT_11EPI[16:6]), (OUT_9EPI19, OUT_9EPI[16:6]). Η επιλογή κάθε φορά για το ποιο από τα σήματα θα οδηγείται στην έξοδο FINITO_LA_MUSICA[11:0] γίνεται από το σήμα KAPA_OUT[2:0]. Η ενεργοποίηση του **U126** γίνεται από το σήμα CE_U67. Η τιμή του bus FINITO_LA_MUSICA[11:0] αποθηκεύεται στη μνήμη Single Port **U131** (END) η οποία ενεργοποιείται για εγγραφή για ένα μόνο παλμό ρολογιού από τον μετρητή **U134** (COUNT_END) με το σήμα WE_U131_OUT. Ο μετρητής **U134** ξεκινάει να μετρά από το 0 έχοντας ενεργοποιηθεί από το QTHRESH1 του **U101,** δίνει ένα παλμό όταν φτάσει στο 13 και σταματάει το μέτρημα όταν μετρήσει 14. Στον επόμενο παλμό ρολογιού η μνήμη **U131** δίνει στην έξοδό της τον κοινό θόρυβο που τελικά υπολογίστηκε με τον δίαυλο AT_THE_END[11:0].

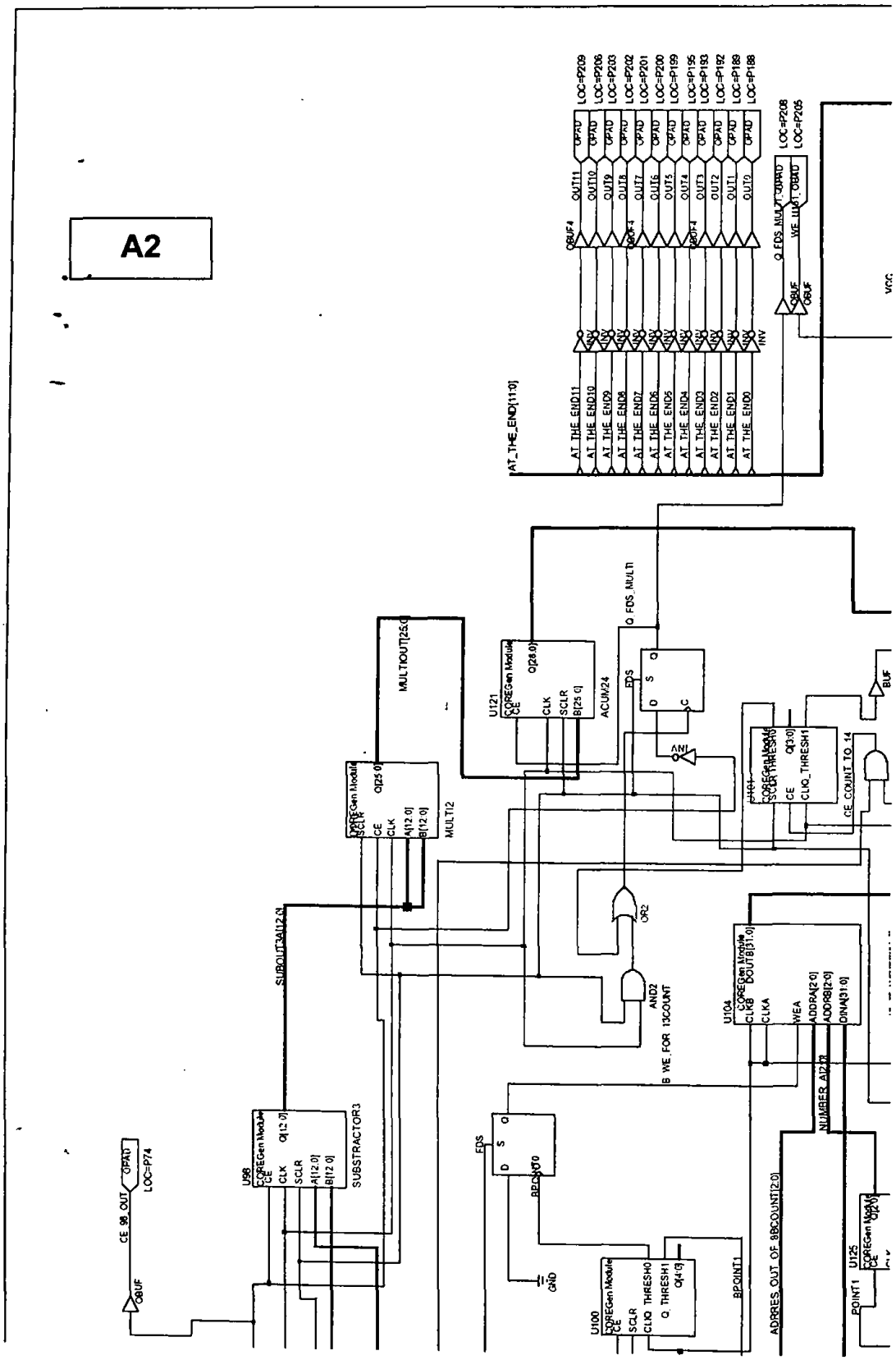
Ας σημειωθεί ότι στην περίπτωση που στην έξοδο υπάρχει 1 στο πιο σημαντικό ψηφίο τότε, επειδή όπως προαναφέρθηκε ο αριθμός είναι προσημασμένος, ο κοινός θόρυβος είναι αρνητικός αριθμός οπότε είναι το συμπλήρωμα του αριθμού AT_THE_END ως προς το 2 (2s compliment) [11].



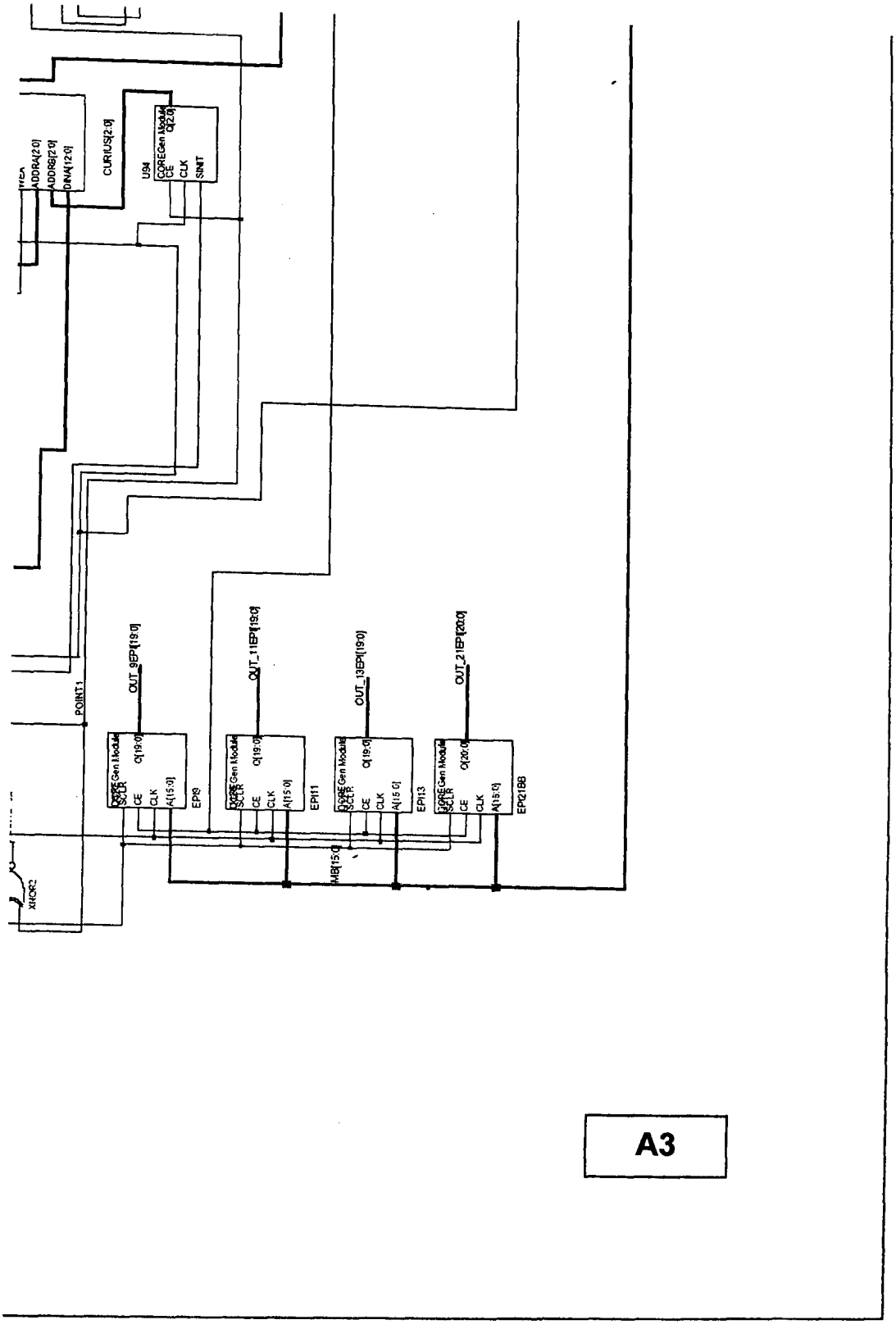
Διάγραμμα 9: Σχηματικό του κυκλώματος που υλοποιήθηκε στο FPGA



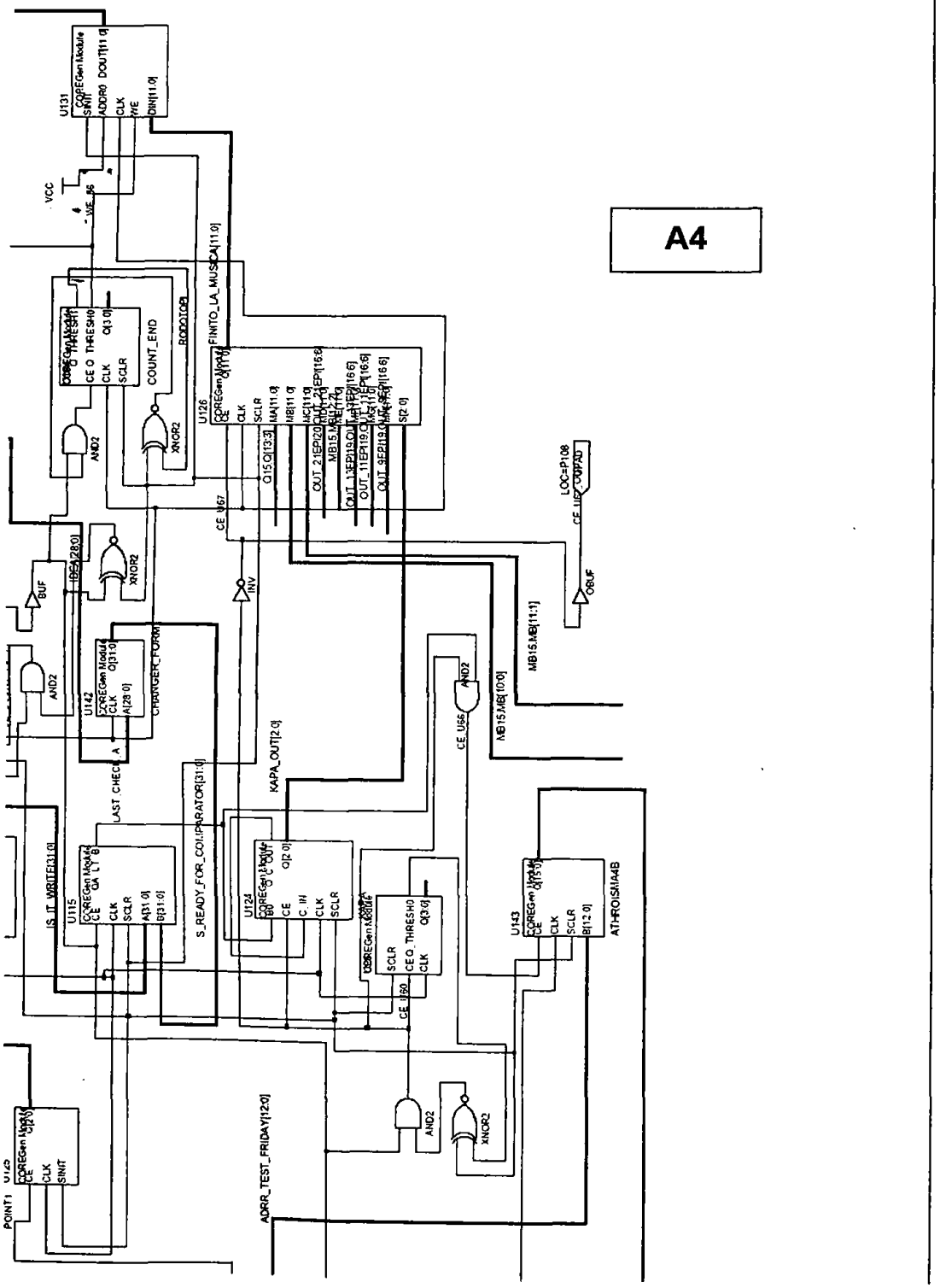
Διάγραμμα 10: Μεγέθυνση του τομέα A1 του σχηματικού διαγράμματος



Διάγραμμα 10: Μεγέθυνση του τομέα A2 του σχηματικού διαγράμματος



Διάγραμμα 10: Μεγέθυνση του τομέα Α3 του σχηματικού διαγράμματος



Διάγραμμα 10: Μεγέθυνση του τομέα Α4 του σχηματικού διαγράμματος

3.2 Οι ακροδέκτες του FPGA.

Πίνακας 1

PIN	signal	description	PIN	signal	description
1	**Gnd**		43	**Vccint**	
2	**TMS**		44	**Vcco**	
3			45	**Gnd**	
4	**ENABLE_1B_OUT**	Control signal	46		
5			47		
6			48		
7			49		
8	**Gnd**		50		
9			51	**Gnd**	
10			52		
11			53		
12			54		
13	**ENABLE_1_OUT**	Control signal	55	**Vcco**	
14	**Gnd**		56		
15	**Vcco**		57		
16	**Vccint**		58	**M1**	
17			59	**Gnd**	
18			60	**M0**	
19			61	**Vcco**	
20			62	**M2**	
21			63		
22	**Gnd**		64		
23			65		
24	**ENABLE_1C_OUT**	Control signal	66		
25	**Vcco**		67		
26			68		
27			69	**Gnd**	
28			70		
29	**Gnd**		71		
30	**Vcco**		72		
31			73		
32	**Vccint**		74	**CE_98_OUT**	Control signal
33			75	**Gnd**	
34			76	**Vcco**	
35			77	**Vccint**	
36			78		
37	**Gnd**		79		
38			80		
39			81		
40			82		
41			83	**Gnd**	
42			84		

PIN	signal	description	PIN	signal	description
85	Vcco		127		
86			128		
87			129	Gnd	
88	Vccint		130		
89	CLR	Clear	131		
90	Vcco		132		
91	Gnd		133		
92	CLK	Clock	134		
93			135	Gnd	
94			136	Vcco	
95			137	Vccint	
96			138		
97			139		
98	Gnd		140		
99			141		
100			142		
101			143	Gnd	
102			144		
103			145		
104	Vccint		146	Vcco	
105	Vcco		147		
106	Gnd		148	Vccint	
107			149		
108	CE_U67_OUT	Control signal	150	Vcco	
109			151	Gnd	
110			152		
111			153		
112	Gnd		154		
113			155		
114			156		
115			157		
116	Vcco		158	Gnd	
117			159		
118			160		
119	Gnd		161		
120	DONE		162		
121	Vcco		163		
122	PROGRAM		164	Vccint	
123			165	Vcco	
124			166	Gnd	
125			167		
126			168		

PIN	signal	description	PIN	signal	Description
169			205	WE_U131_OUT	Control signal
170			206	OUT10	Result
171			207	Vcco	
172	Gnd		208	Q_FDS_MULTI_OUT	Control signal
173			209	OUT11	Result
174			210		
175			211	Gnd	
176	Vcco		212	Vcco	
177			213		
178			214	Vccint	
179	CCLK		215		
180	Vcco		216	ENABLE_1A_OUT	Control signal
181	TDO		217		
182	Gnd		218		
183	TDI		219	Gnd	
184			220		
185			221		
186			222		
187			223		
188	OUT0	Result	224		
189	OUT1	Result	225	Vccint	
190	Gnd		226	Vcco	
191			227	Gnd	
192	OUT2	Result	228		
193	OUT3	Result	229		
194	CE_U67_OUT	Control signal	230		
195	OUT4	Result	231		
196	Gnd		232	Vcco	
197	Vcco		233	Gnd	
198	Vccint		234		
199	OUT5	Result	235		
200	OUT6	Result	236		
201	OUT7	Result	237		
202	OUT8	Result	238		
203	OUT9	Result	239	TCK	
204	Gnd		240	Vcco	

Σήματα ελέγχου
Σήματα προγραμματισμού FPGA

Στον Πίνακα 1 φαίνονται οι ακροδέκτες του FPGA και η αντίστοιχη περιγραφή τους. Όπου δεν υπάρχει περιγραφή είτε οι ακροδέκτες δεν χρησιμοποιούνται είτε έχουν προκαθορισμένα από την XILINX σήματα.

3.3 Χρονική προσομοίωση.

Το κύκλωμα προσομοιώθηκε χρονικά (timing simulation) με τον προσομοιωτή που διαθέτει το πακέτο λογισμικού της XILINX, Foundation 3.3.08i, και διαπιστώθηκε η καλή συναρτησιακή και χρονική λειτουργία του. Κατά την διαδικασία του προγραμματισμού του XCV100E PQ240, έγινε πετυχημένη υλοποίηση (implementation) με τις προκαθορισμένες (default) παραμέτρους του λογισμικού της υλοποίησης που έχει το πακέτο λογισμικού XILINX foundation 3.3.08i, εκτός των:

Optimize and Map Options:

- Pack CLB Registers for: Maximum Area

Παρακάτω παρατίθεται η έξοδος του λογισμικού της υλοποίησης.

Design Information

Command Line : map -p xcv100e-8-pq240 -o map.ncd withpads.ngd withpads.pcf
Target Device : xv100e
Target Package : pq240
Target Speed : -8
Mapper Version : virtexe -- D.27
Mapped Date : Tue Aug 21 23:40:13 2001

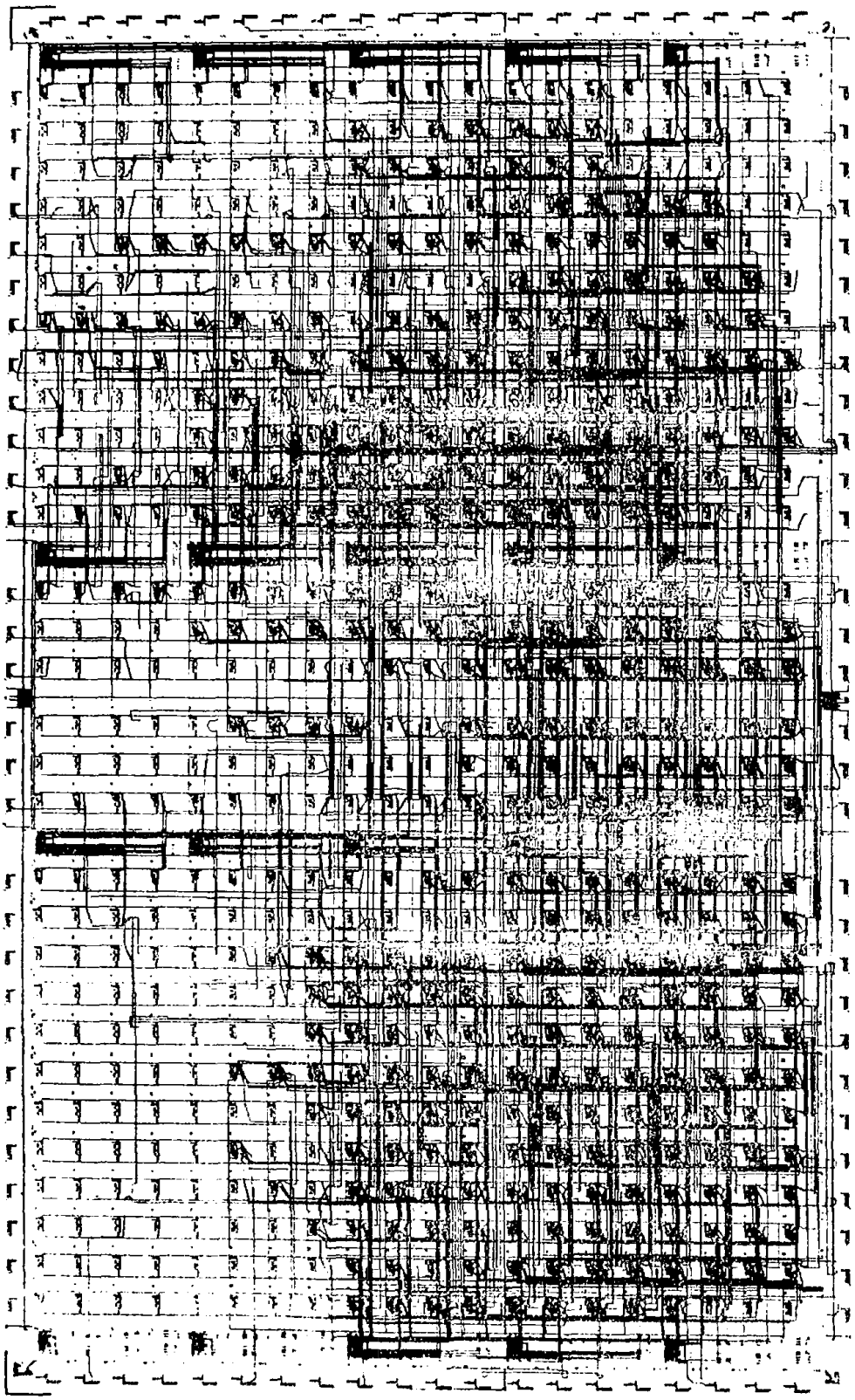
Design Summary

Number of Slices:	588	out of 1,200	49%
Number of Slices containing unrelated logic:	0	out of 588	0%
Number of Slice Flip Flops:	610	out of 2,400	25%
Number of 4 input LUTs:	934	out of 2,400	38%
Number of bonded IOBs:	20	out of 158	12%
Number of Block RAMs:	18	out of 20	90%
Number of GCLKs:	2	out of 4	50%
Number of GCLKIOBs:	2	out of 4	50%
Number of RPM macros:	27		

Total equivalent gate count for design: 309,894

Additional JTAG gate count for IOBs: 1,056

Από την έξοδο του λογισμικού της υλοποίησης φαίνεται ότι το κύκλωμα χρειάζεται περίπου 300 χιλιάδες λογικές πύλες για να υλοποιηθεί. Το συγκεκριμένο FPGA δηλαδή δεν θα επαρκούσε, όμως επειδή χρησιμοποιήθηκε ο Core Generator οι πύλες δημιουργήθηκαν μέσα από Look Up Tables (LUTs) και έγινε η βέλτιστη υλοποίηση [19]. Πρέπει να σημειωθεί ότι το πακέτο λογισμικού XILINX Foundation προτείνει ότι το κύκλωμα λειτουργεί κανονικά με ρολόι μέχρι τα 80 MHz. Μια εποπτική εικόνα της συνολικής επιφάνειας κάλυψης των πόρων του FPGA μας δίνει ο FPGA editor, εργαλείο που περιέχεται στο XILINX Foundation. Στο διάγραμμα 11 φαίνεται το τμήμα που έχει δρομολογηθεί με διασυνδέσεις (routed) είναι σκούρο λόγω της πυκνότητας των γραμμών, ενώ τα τμήματα που έμειναν αχρησιμοποίητα είναι τα λιγότερο έντονα.

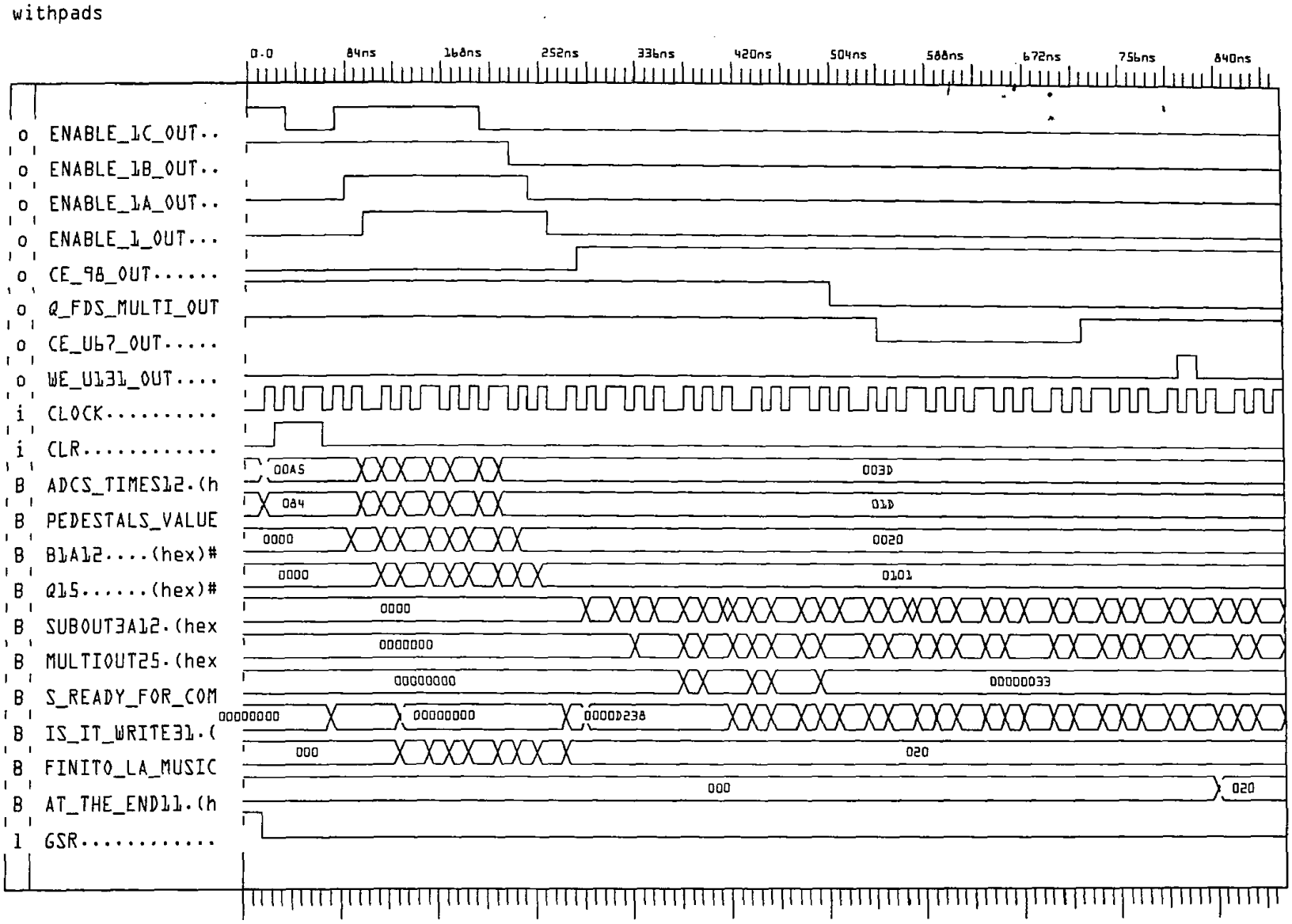


Διάγραμμα 11: Η κάλυψη των πόρων του ολοκληρωμένου

Στο διάγραμμα 12 φαίνεται η χρονική αλληλουχία των κυριοτέρων σημάτων του κυκλώματος. Αρχικά φαίνονται τα σήματα ελέγχου και στη συνέχεια τα σήματα των διαύλων του κυκλώματος και ο τρόπος με τον οποίο αλλάζουν τα δεδομένα τους.

Μόλις το σήμα CLR γίνει 0, στον επόμενο παλμό τα σήματα ENABLE_1C_OUT (ενεργοποιεί τον απαριθμητή για τη διευθυνσιοδότηση των μνημών) και ENABLE_1A_OUT (ενεργοποιεί τον αφαιρέτη των υποβάθρων) γίνονται 1 και τα σήματα ENABLE_1C_OUT και ENABLE_1_OUT (ενεργοποιεί τις υπόλοιπες υπομονάδες του κυκλώματος) ακολουθούν διαδοχικά. Το ENABLE_1C_OUT αφού μείνει για 6 παλμούς ρολογιού στο 1 ξαναγίνεται 0, στον επόμενο παλμό το ακολουθεί και το ENABLE_1B_OUT (το οποίο ήταν 1 από το CLR και αφορά τις μνήμες που περιέχονται τα υπόβαθρα και οι τιμές των ADCs) και διαδοχικά ακολουθούν και τα υπόλοιπα σήματα ENABLE_1A_OUT και ENABLE_1_OUT. Στην επόμενη άνοδο του ρολογιού το σήμα CE_98_OUT (ενεργοποιείται η αφαίρεση της κάθε τιμής από το μέσο όρο) γίνεται 1. Το σήμα Q_FDS_MULTI_OUT (ενεργοποιεί τον αθροιστή των τετραγώνων της διαφοράς από τη μέση τιμή) από 1 γίνεται 0 ύστερα από 13 παλμούς ρολογιού αφότου το ENABLE_1A_OUT είχε γίνει 0. Το σήμα CE_U67_OUT (ενεργοποιεί τον αποπολυπλέκτη) αντίστοιχα μετά από 15 παλμούς ρολογιού (αφότου μηδενίστηκε το ENABLE_1A_OUT) γίνεται 0 και διατηρείται σε αυτή τη κατάσταση για 9 παλμούς ρολογιού, κατόπιν ξαναγίνεται 1. Τέλος το σήμα WE_U131_OUT δίνει 1 παλμό μετά από 28 κύκλους ρολογιού, αφότου μηδενίστηκε το ENABLE_1A_OUT, και με αυτό το σήμα γίνεται και η εγγραφή στη τελική μνήμη.

Από ότι φαίνεται και στο διάγραμμα το αποτέλεσμα δηλ. ο υπολογισμός κοινού θορύβου 8 καναλιών, απαιτεί 30 κύκλους ρολογιού ενώ το όλο κύκλωμα λειτουργεί σε συχνότητα μέχρι 80 MHz.



Διάγραμμα 12: Χρονική προσομοίωση των σημάτων του FPGA.

ΚΕΦΑΛΑΙΟ 4
Αναπτυξιακή κάρτα της XILINX.

Η συνηθισμένη προσέγγιση να πειραματίζεται κανείς με καινούργιες συσκευές ενώνοντας τα διάφορα ολοκληρωμένα με καλώδια πάνω σε μια δοκιμαστική πλακέτα γρήγορα γίνεται μη πρακτική και αναποτελεσματική κυρίως λόγω των υψηλών συχνοτήτων που χρησιμοποιούνται στις σημερινές εφαρμογές. Ακριβώς για αυτό το λόγο οι σχεδιαστές που χρησιμοποιούν νέες υψηλής πυκνότητας και ταχύτητας συσκευές, χρειάζονται προσαρμοσμένες πλακέτες ολοκληρωμένων ανάλογα με τις ανάγκες τους. Οι αναπτυξιακές κάρτες των κατασκευαστών ολοκληρωμένων κυκλωμάτων καλύπτουν την ανάγκη για πειραματισμό πάνω στα ολοκληρωμένα και επιπρόσθετα, διευκολύνουν την κατανόηση και την αξιοποίηση των πλεονεκτημάτων που παρέχουν τα ολοκληρωμένα κυκλώματα.

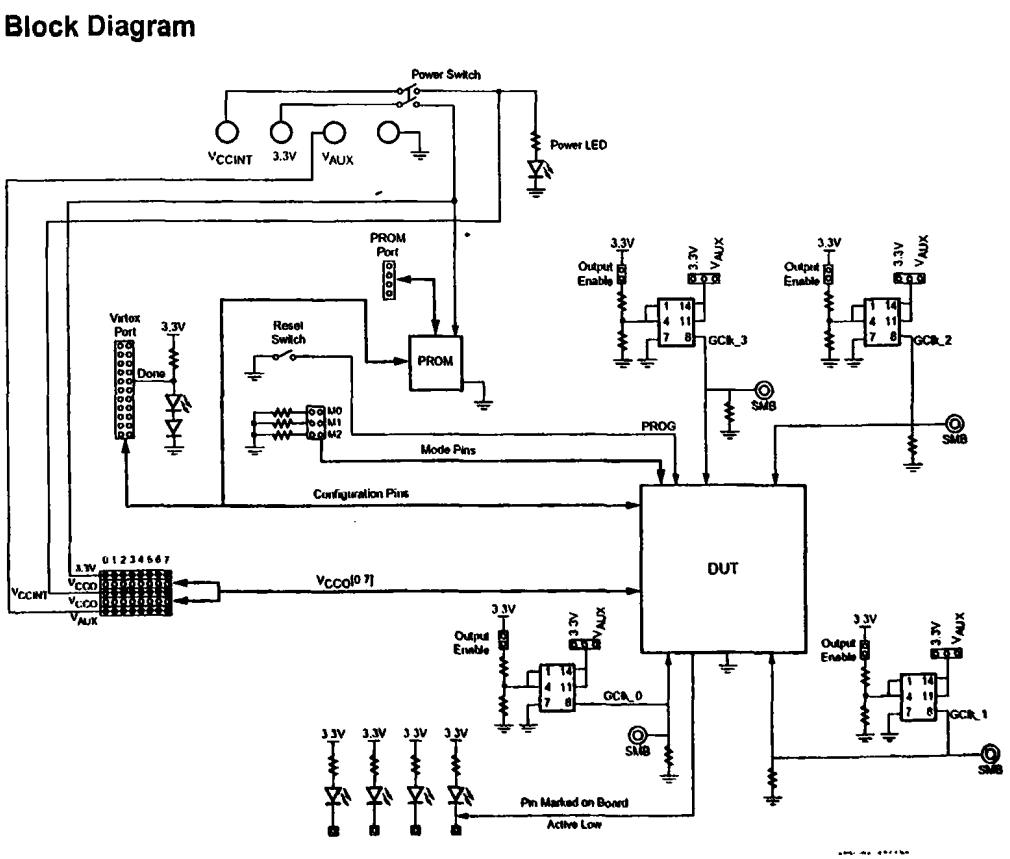
Η αναπτυξιακή κάρτα της XILINX που χρησιμοποιήθηκε για τον έλεγχο του κυκλώματος του FPGA που αναπτύχθηκε επιτρέπει τον πειραματισμό με FPGA της σειράς Virtex της XILINX τα οποία χρησιμοποιούνται για συγκεκριμένες εφαρμογές. Επιτρέπει στους σχεδιαστές

να δοκιμάζουν χαρακτηριστικά των Virtex όπως block RAM, DLLs κ.α. Επίσης διευκολύνει στη κατανόηση ολόκληρου του σχεδίου από το σχηματικό μέρος μέχρι το προγραμματισμό και την επαλήθευσή του.

Η αναπτυξιακή κάρτα της XILINX που χρησιμοποιήθηκε είναι η HW-AFX-PQ240-100 που βοηθάει στον έλεγχο FPGAs της σειράς E με κέλυφος PQ240.

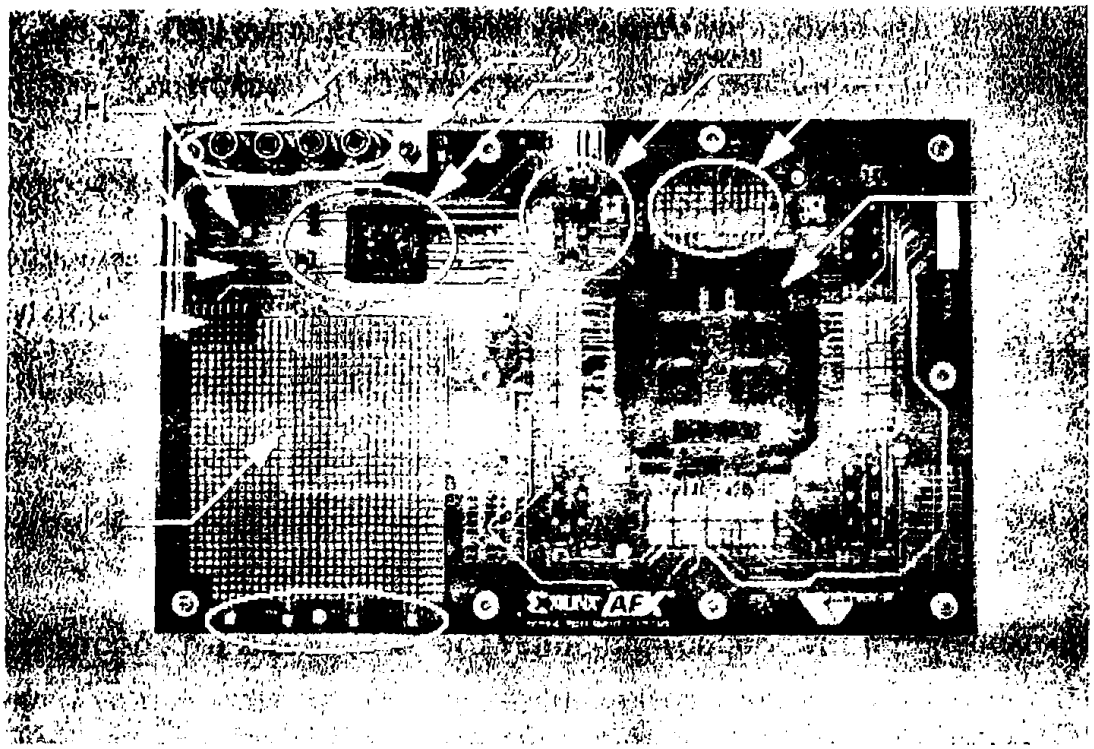
4.1 Σχηματικό διάγραμμα – Περιγραφή της κάρτας.

Το γενικό διάγραμμα της αναπτυξιακής κάρτας φαίνεται στο διάγραμμα 13 [12].



Διάγραμμα 13: Το γενικό διάγραμμα της αναπτυξιακής κάρτας της XILINX.

Τα κύρια μέρη της φαίνονται στο επόμενο διάγραμμα-φωτογραφία 14 και περιγράφονται στις επόμενες παραγράφους.



Διάγραμμα 14: Φωτογραφία της αναπτυξιακής κάρτας XILINX που χρησιμοποιήθηκε.

1. Τροφοδοσία

Η πλακέτα έχει 4 εισόδους τροφοδοσίας Vccint, 3.3V, Vaux και GND. Η Vccint τροφοδοτεί το πυρήνα του FPGA Virtex. Στο εγχειρίδιο [12] της XILINX μπορούμε να βρούμε ποια είναι η μέγιστη τάση Vccint ανάλογα με ποιο τύπο Virtex χρησιμοποιούμε. Στην περίπτωση μας ειδικότερα, για το VIRTEX –E XCV100E PQ240 –8 η προτεινόμενη τάση λειτουργίας είναι 1.8V με 5% απόκλιση. Η τάση 3.3V τροφοδοτεί την μνήμη PROM που βρίσκεται ενσωματωμένη πάνω στη πλακέτα, τους 4 ταλαντωτές και όλους τους ακροδέκτες που είναι μαρκαρισμένοι με 3.3V συμπεριλαμβανόμενου της Virtex port, της περιοχής ανάπτυξης πρωτοτύπου και των βραχυκυκλωτήρων Vcco.

2. Διακόπτης τροφοδοσίας

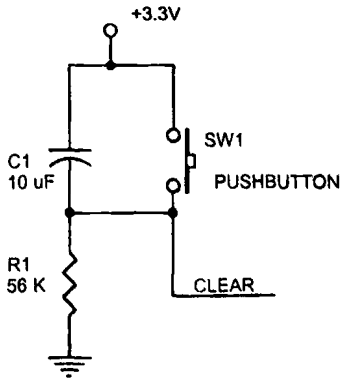
Ένας διακόπτης τροφοδοσίας συνδέει τη Vccint και την τάση 3.3V στη πλακέτα. Δεν συνδέει την Vaux ή την GND. Ένα πράσινο LED μας δείχνει ότι έχουμε τάση στο πυρήνα όταν το Vccint είναι 2.0V η μεγαλύτερο.

3. Βάση υποδοχής DUT

Είναι η βάση υποδοχής του FPGA. Πρέπει να χρησιμοποιείται πάντα το ειδικό εργαλείο κενού για εισαγωγή – εξαγωγή του FPGA ώστε να αποφεύγεται πιθανή ζημιά σε κάποιο από τους ακροδέκτες της συσκευής VIRTEX.

4. Ακροδέκτες

Η περιοχή με τους ακροδέκτες χρησιμοποιείτε για τη παρακολούθηση η την εφαρμογή σημάτων στο FPGA. Κάθε ακροδέκτης έχει δίπλα του μια γείωση για ευκολία στη χρήση. Είναι η περιοχή στην οποία τοποθετήθηκαν τα LEDs για την παρακολούθηση των αποτελεσμάτων καθώς και ο διακόπτης CLR (CLEAR). Τα LEDs έχουν συνδεθεί στη τάση ώστε να ανάβουν όταν ο ακροδέκτης του FPGA που είναι συνδεδεμένα γίνεται 0. Επειδή η τάση είναι τύπου LVTTL , συνδέθηκε μια αντίσταση R των 120 Ω σε σειρά με κάθε LED. Ο διακόπτης CLEAR συνδέθηκε παράλληλα με ένα πυκνωτή 10μF και σε σειρά με μια αντίσταση 56 K όπως φαίνεται στο διάγραμμα 15.



Διάγραμμα 15: Ο διακόπτης CLR του κυκλώματος.

5. PROM και διακόπτης RESET

- Η υποδοχή SPROM μπορεί να χρησιμοποιηθεί για το προγραμματισμό της συσκευής Virtex σε master serial mode. Η υποδοχή δέχεται τις σειρές XC1700 & XC1800 της XILINX σε PC44 packages. Η PROM port μπορεί να χρησιμοποιηθεί για τον επαναπρογραμματισμό της XC1800 μέσω του JTAG. Πατώντας το διακόπτη reset, ο ακροδέκτης PROGRAM του FPGA γειώνεται και το FPGA επαναπρογραμματίζεται.

6. Virtex Port

Η Virtex Port χρησιμοποιείται για να συνδέσουμε ένα καλώδιο φόρτωσης του προγράμματος στο FPGA. Η Virtex port υποστηρίζει όλους τους τρόπους προγραμματισμού των συσκευών Virtex.

7. Γέφυρες Βραχυκύκλωσης ακροδεκτών τρόπου προγραμματισμού.

Οι γέφυρες βραχυκύκλωσης M0, M1 και M2 συνδέουν τους ακροδέκτες που αφορούν τον τρόπο προγραμματισμού του FPGA με τη γείωση, επιλέγοντας έτσι τον επιθυμητό τρόπο. Το FPGA παρέχει ένα ασθενές 1 στον κάθε ακροδέκτη το οποίο το οδηγεί στο 1 όταν αφαιρεθεί η γέφυρα βραχυκύκλωσης.

8. Γέφυρες Βραχυκύκλωσης τροφοδοσίας Vcco.

Τα FPGAs Virtex έχουν 8 ομάδες εισόδου-εξόδου, καθεμία με μια Vcco τάση (μόνο μια για το PQ package). Οι γέφυρες βραχυκύκλωσης Vcco συνδέουν κάθε ομάδα με μια από τις 3 τροφοδοσίες: Vccint, 3.3V ή Vaux. Αυτές οι γέφυρες βραχυκύκλωσης πρέπει να υπάρχουν οπωσδήποτε για να λειτουργεί η συσκευή Virtex. Κάθε ομάδα (0-7) μπορεί να συνδεθεί με 3.3V, Vccint ή Vaux.

9. Ταλαντωτές GCLK και γέφυρες βραχυκύκλωσης.

Η πλακέτα έχει 4 υποδοχές για ταλαντωτές, μια για κάθε GCLK. Υπάρχουν γέφυρες βραχυκυκλωτήρων δίπλα στο κάθε ταλαντωτή που ελέγχουν την έξοδο του. Όταν η γέφυρα βραχυκύκλωσης είναι τοποθετημένη το enable pin είναι high, αλλιώς είναι low. Επίσης υπάρχουν γέφυρες βραχυκύκλωσης με τις οποίες επιλέγεται αν ο ταλαντωτής θα συνδεθεί στα 3.3V η στη Vaux. Ακόμα σε κάθε ακροδέκτη GCLK υπάρχει και ένας συνδέτης τύπου SMB.

10. LEDs Χρήστη.

Υπάρχουν 4 LED που καθορίζονται από το χρήστη. Το LED που βρίσκεται στο κάτω δεξιά μέρος της περιοχής ανάπτυξης πρωτοτύπου είναι συνδεδεμένο με τον αριθμό του pin που αναγράφεται δίπλα του. Υπάρχει ένα σημείο ελέγχου για κάθε LED, όταν αυτό είναι Low το LED ανάβει.

11. DONE LED.

Το LED που βρίσκεται δίπλα στη Virtex port δείχνει την κατάσταση του ακροδέκτη DONE. Το LED ανάβει όταν το DONE είναι High και σβήνει όταν το DONE είναι Low. Αν επίσης εφαρμόσουμε τάση στη πλακέτα χωρίς FPGA στην υποδοχή DUT το LED ανάβει.

12. Περιοχή ανάπτυξης πρωτοτύπου.

Η περιοχή για προτυποποίηση έχει οπές με απόσταση μεταξύ των 0.10". Επίσης οι τάσεις τροφοδοσίας Vccint, 3.3V & Vaux είναι διαθέσιμες σε αυτή την περιοχή.

4.2 Τρόποι λειτουργίας

Ο πίνακας 1 που ακολουθεί δείχνει τις συνδέσεις μεταξύ του MultiLinx καλωδίου (xchecker) και της αναπτυξιακής κάρτας της XILINX για

τον αντίστοιχο τρόπο λειτουργίας. Επίσης φαίνονται καθαρά και οι θέσεις των βραχυκυκλωτήρων προγραμματισμού με τη σειρά M2, M1 και M0. Ο αριθμός 1 δηλώνει ότι ο βραχυκυκλωτήρας έχει αφαιρεθεί, ενώ το 0 ότι είναι εγκατεστημένος.

Table 1: Slave Serial Mode; Mode Pins: 111 or 011

MultiLinx Cable	Prototype Board
PWR	3.3V
GND	GND
CCLK	CCLK
DONE	DONE
DIN	D0
PROG	PROG
INIT	INIT

Πίνακας 1: Οι θέσεις των βραχυκυκλωτήρων προγραμματισμού και οι συνδέσεις του xchecker με την αναπτυξιακή πλακέτα XILINX.

Στην συγκεκριμένη πειραματική διάταξη χρησιμοποιήθηκε ο τρόπος λειτουργίας Slave Serial Master και κάθε φορά το ολοκληρωμένο προγραμματίζεται με τη βοήθεια του σειριακού καλωδίου.

4.3 Φόρτωση λογισμικού στο ολοκληρωμένο.

Με την έναρξη λειτουργίας (power up) της αναπτυξιακής κάρτας XILINX στην οποία τοποθετείται το FPGA, προγραμματίζεται το FPGA μέσω του καλωδίου xchecker της XILINX. Ο εξωτερικός ταλαντωτής που

χρησιμοποιήθηκε είναι 50 MHz. Η τάση Vccint = 1.8 Volt (τάση πυρήνα) και το κύκλωμα τροφοδοτείται με 3.3 Volt. Τίθεται σε λειτουργία το ενσωματωμένο πρόγραμμα Hardware Debugger και αφού ανιχνεύσει το καλώδιο xchecker και τον ρυθμό μετάδοσης που είναι 9600 bits αρχίζει τον προγραμματισμό των τμημάτων (blocks) του προγράμματος στο ολοκληρωμένο. Μόλις τελειώσει ο προγραμματισμός το LED DONE της αναπτυξιακής κάρτας θα ανάψει καθώς και το 1° αποτέλεσμα θα φανεί μέσω των LEDs που έχουν τοποθετηθεί. Στην αναπτυξιακή κάρτα το πράσινο LED αντιστοιχεί στο λιγότερο σημαντικό ψηφίο.

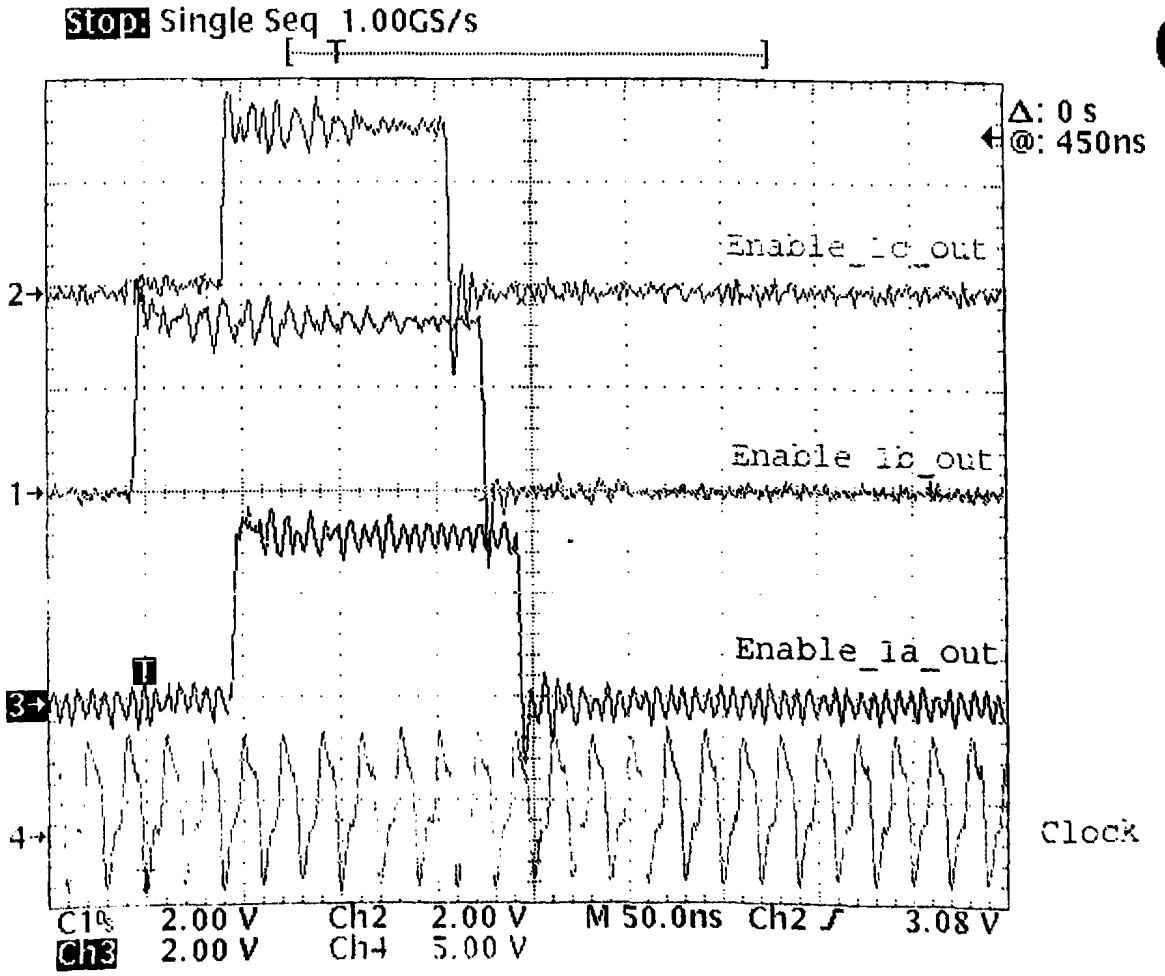
Με πληκτρολόγηση του πλήκτρου αρχικοποίησης (reset) στην αναπτυξιακή κάρτα το κύκλωμα αρχικοποιείται και στον επόμενο παλμό ρολογιού φορτώνονται στο κύκλωμα οι τιμές των pedestals και των ADCs από τις μνήμες (το reset έχει ως αποτέλεσμα να δίνονται στις μνήμες προκαθορισμένες τιμές από δυο αρχεία τιμών, παράρτημα ΣΤ) και ταυτόχρονα αρχίζει να λειτουργεί ο αλγόριθμος.

ΚΕΦΑΛΑΙΟ 5
Αποτίμηση του κυκλώματος αφαίρεσης κοινού θορύβου και υποβάθρου.

5.1 Αποτελέσματα ελέγχου.

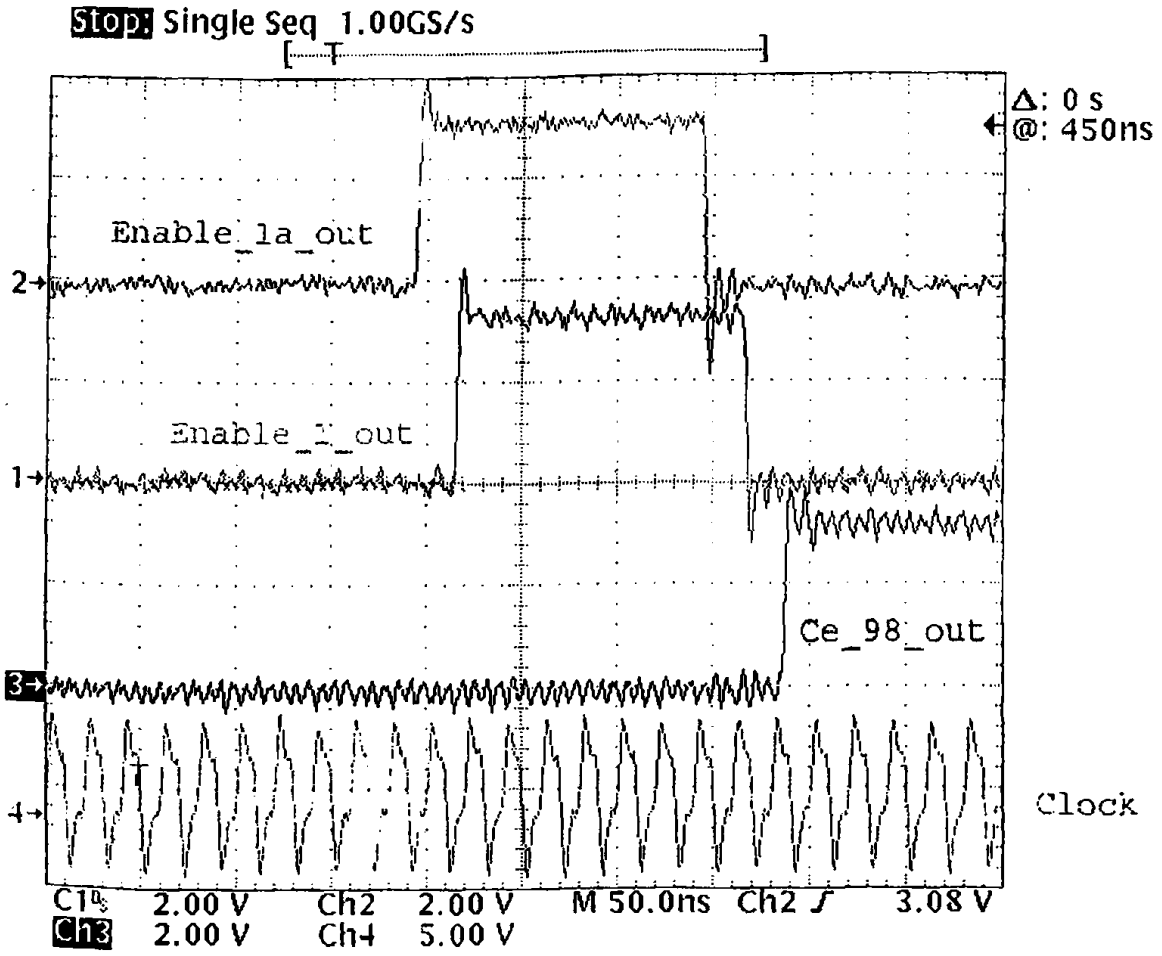
Ο έλεγχος λειτουργίας του κυκλώματος του FPGA έγινε σε δυο στάδια. Πρώτα ελέγχθηκε μέσω του verification του πακέτου λογισμικού XILINX FOUNDATION (timing simulation) όπως προαναφέρθηκε στο Κεφάλαιο 3. Κατά τον κύριο έλεγχο μετρήθηκαν τα κύρια σήματα ελέγχου του κυκλώματος με τον παλμογράφο αφού φορτώθηκε το κύκλωμα στο FPGA μέσω του καλωδίου xchecker της XILINX. Με τη βοήθεια του παλμογράφου Tektronix TDS 684B απεικονίσθηκε η χρονική αλληλουχία των αντίστοιχων σημάτων ελέγχου μετρήθηκαν τα αντίστοιχα σήματα ελέγχου και επιβεβαιώθηκε η ορθή λειτουργία του κυκλώματος. Παρακάτω ακολουθούν εικόνες από τον παλμογράφο με τα σήματα που μετρήθηκαν. Τα εν λόγω σήματα αντιστοιχούν στα σήματα του διαγράμματος προσομοίωσης (Διάγραμμα 12).

Η χρονική αλληλουχία όλων των σημάτων είναι η αναμενόμενη και ταυτίζεται πλήρως με τα αποτελέσματα της χρονικής προσομοίωσης του κυκλώματος. Στο διάγραμμα 16 φαίνονται τα σήματα enable_1c_out, enable_1b_out και enable_1a_out σε σχέση με το ρολόι (clock). Παρατηρούμε ότι μόλις το enable_1c_out γίνει 0, στον επόμενο παλμό ρολογιού γίνεται και το enable_1b_out 0 και στον επόμενο παλμό ρολογιού το ακολουθεί και το enable_1a_out, ακριβώς όπως και στη χρονική προσομοίωση.



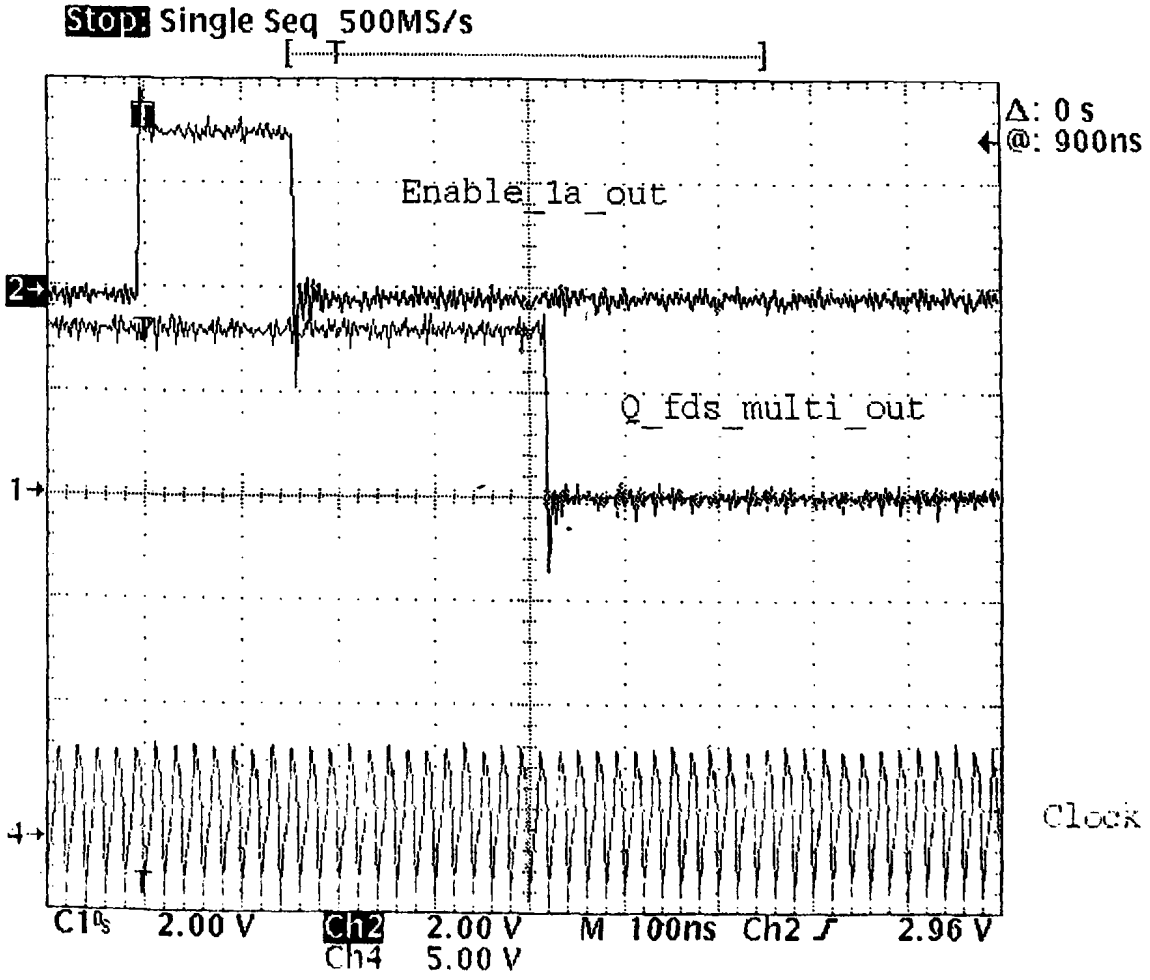
Διάγραμμα 16: Τα σήματα ελέγχου enable_1c_out, enable_1b_out και enable_1a_out σε σχέση με το ρολόι (clock) όπως απεικονίζονται στον παλμογράφο.

Στο Διάγραμμα 17 φαίνονται τα σήματα enable_1a_out, enable_1_out, ce_98_out σε σχέση με το ρολόι (clock). Παρατηρούμε ότι μόλις το enable_1α_out γίνει 0, στον επόμενο παλμό ρολογιού γίνεται και το enable_1_out 0 και στον επόμενο παλμό ρολογιού το ce_98_out γίνεται από 0 → 1, όπως και στο διάγραμμα 12 της χρονικής προσομοίωσης.



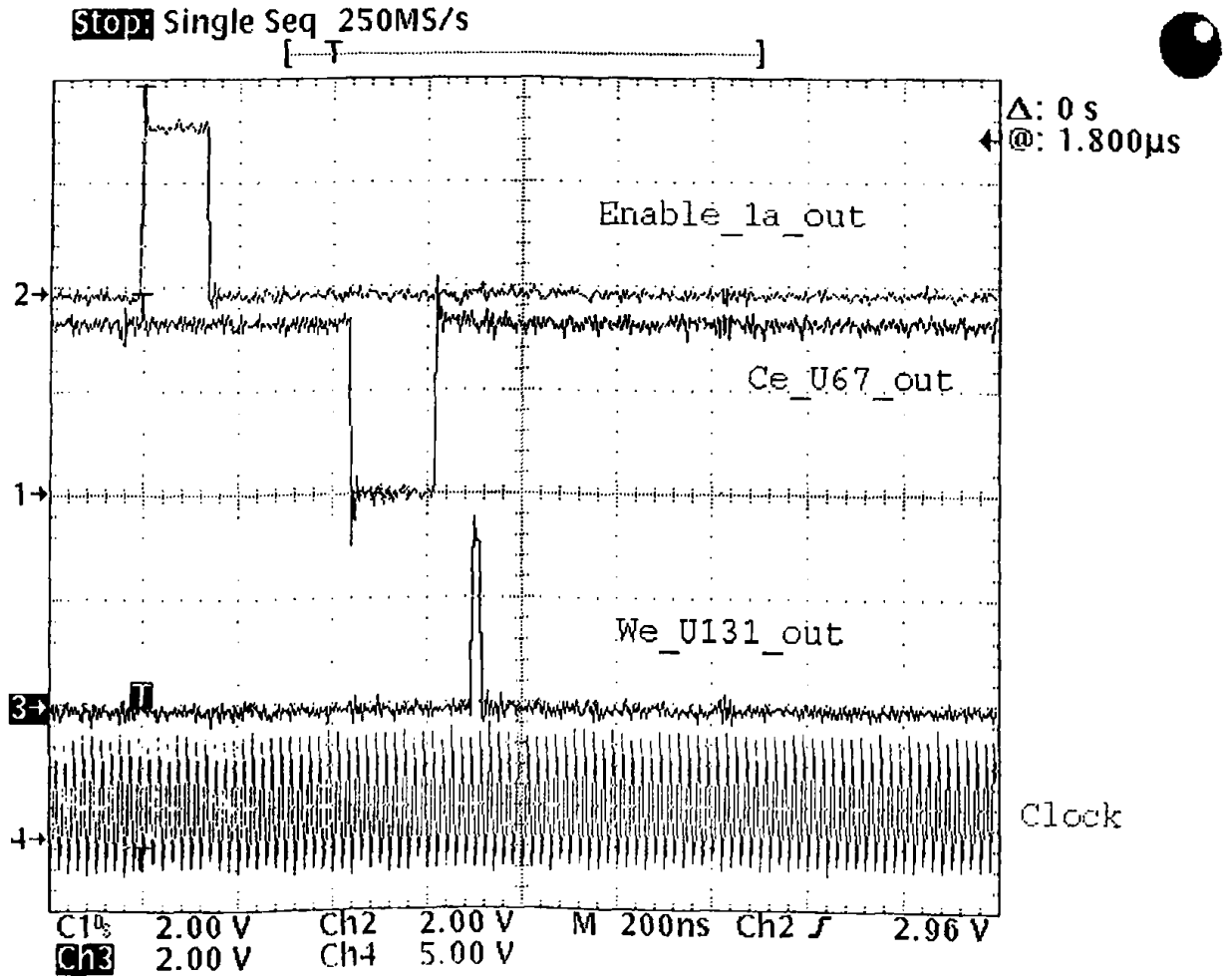
Διάγραμμα 17: Τα σήματα ελέγχου enable_1a_out, enable_1_out και Ce_98_out σε σχέση με το ρολόι (clock) όπως απεικονίζονται στον παλμογράφο.

Στο Διάγραμμα 18 φαίνονται τα σήματα enable_1a_out και q_fds_multi_out σε σχέση με το ρολόι (clock). Παρατηρούμε ότι μόλις το enable_1α_out γίνει 0, το q_fds_multi_out γίνεται από 1 → 0 μετά από 13 παλμούς ρολογιού όπως και στη χρονική προσομοίωση.



Διάγραμμα 18: Τα σήματα ελέγχου enable_1a_out και q_fds_multi_out σε σχέση με το ρολόι (clock) όπως απεικονίζονται στον παλμογράφο.

Στο Διάγραμμα 19 φαίνονται τα σήματα enable_1a_out, ce_u67_out και we_u131_out σε σχέση με το ρολόι (clock). Παρατηρούμε ότι μόλις το enable_1α_out γίνει 0, μετά από 15 παλμούς ρολογιού το ce_u67_out γίνεται 0 και μένει σε αυτή τη κατάσταση για 9 παλμούς ρολογιού και στη συνέχεια ξαναγίνεται 1. Το σήμα we_u131_out είναι 0 και μας δίνει 1 παλμό μετά από 28 κύκλους ρολογιού από τη στιγμή που το enable_1a_out έγινε 0 όπως και στη χρονική προσομοίωση.



Διάγραμμα 19: Τα σήματα ελέγχου enable_1a_out , ce_u67_out και we_u131_out σε σχέση με το ρολόι (clock) όπως απεικονίζονται στον παλμογράφο.

5.2 Σύγκριση με αποτελέσματα του Matlab

Στη συνέχεια ακολουθεί μια σειρά μετρήσεων που πραγματοποιήθηκαν θέτοντας προσομοιωμένες τιμές ADC στα 8 κανάλια και προσομοιωμένες τιμές κοινού θορύβου. Στον πίνακα 2 φαίνονται οι τιμές ADC (σε ADC counts) του ψηφιοποιημένου σήματος που περιέχουν και θόρυβο και υπόβαθρο και χρησιμοποιήθηκαν σαν τιμές εισόδου στο κύκλωμα. Το υπόβαθρο και ο κοινός θόρυβος που περιέχουν φαίνονται στους πίνακες 3 και 4 αντίστοιχα. Οι εν λόγω τιμές ελήφθησαν από το πρόγραμμα προσομοίωσης Monte Carlo στο Matlab (κεφάλαιο 2).

Γεγονός #	ADC counts								Ch# με πραγματικό σήμα
	Ch#1	Ch#2	Ch#3	Ch#4	Ch#5	Ch#6	Ch#7	Ch#8	
A1	165	182	96	209	208	155	138	61	-
A2	64	122	128	245	130	256	90	221	-
A3	2653	75	116	70	132	99	182	34	1
A4	4066	3621	96	43	62	59	137	194	1,2
A5	1379	557	1476	31	56	77	140	133	1,2,3
A6	3691	4243	108	178	179	214	167	180	1,2
A7	4062	131	111	210	154	118	146	214	1
A8	2628	180	128	199	170	164	89	140	1
A9	2968	60	76	71	126	39	32	51	1
A10	2321	2476	105	114	82	60	86	171	1,2
A11	18	187	137	191	206	143	71	97	-
A12	2033	132	209	215	172	132	89	155	1
A13	3921	3165	106	92	157	114	144	77	1,2
A14	258	1727	86	162	203	129	104	243	1,2
A15	2843	98	27	113	98	117	107	90	1
A16	721	1499	62	154	83	60	154	144	1,2
A17	2446	147	158	159	176	136	109	186	1
A18	160	179	254	234	151	257	92	157	-
A19	1518	51	206	191	112	85	58	110	1
A20	960	1503	63	43	65	137	82	137	1,2

Πίνακας 2: Οι τιμές των ADCs.

Για χάρη απλούστευσης εάν υπάρχει σήμα από σωμάτιο (πραγματικό γεγονός) αυτό εμφανίζεται στο 1ο κανάλι, εάν υπάρχουν σήματα από 2 σωμάτια εμφανίζονται στο 1ο και 2ο κανάλι και ούτω καθ΄εξής.

Γεγονός #	ADC counts							
	Ch#1	Ch#2	Ch#3	Ch#4	Ch#5	Ch#6	Ch#7	Ch#8
B1	132	151	64	180	178	123	100	29
B2	32	90	90	213	94	220	58	188
B3	17	78	117	73	137	101	187	39
B4	121	78	98	41	64	62	144	195
B5	142	177	127	79	112	130	195	187
B6	172	145	102	173	177	213	161	175
B7	185	111	89	185	134	98	122	200
B8	86	153	104	175	143	145	66	117
B9	179	112	134	131	184	95	86	108
B10	104	160	129	136	105	84	112	198
B11	30	196	153	204	220	157	78	109
B12	145	130	201	211	168	122	86	150
B13	108	116	108	98	160	111	146	78
B14	201	186	90	163	210	130	111	246
B15	148	106	36	116	110	122	111	92
B16	90	161	93	182	119	87	184	180
B17	206	108	123	118	140	102	78	146
B18	112	130	208	185	103	209	43	108
B19	125	59	223	204	122	96	71	115
B20	168	56	90	67	89	160	103	160

Πίνακας 3: Οι τιμές των υποβάθρων.

Γεγονός #	ADC counts							
	Ch#1	Ch#2	Ch#3	Ch#4	Ch#5	Ch#6	Ch#7	Ch#8
Γ1	33	31	32	29	30	32	38	32
Γ2	32	32	38	32	36	36	32	33
Γ3	-6	-3	-1	-3	-5	-2	-5	-5
Γ4	0	-8	-2	2	-2	-3	-7	-1
Γ5	-47	-53	-54	-48	-56	-53	-55	-54
Γ6	11	6	6	5	2	1	6	5
Γ7	20	20	22	25	20	20	24	14
Γ8	18	27	24	24	27	19	23	23
Γ9	-61	-52	-58	-60	-58	-56	-54	-57
Γ10	-22	-22	-24	-22	-23	-24	-26	-27
Γ11	-12	-9	-16	-13	-14	-14	-7	-12
Γ12	4	2	8	4	4	10	3	5
Γ13	-2	5	-2	-6	-3	3	-2	-1
Γ14	-2	5	-2	-6	-3	3	-2	-1
Γ15	-7	-5	-4	-1	-7	-1	-7	-3
Γ16	-27	-25	-31	-28	-36	-27	-30	-36
Γ17	35	39	35	41	36	34	31	40
Γ18	48	49	46	49	48	48	49	49
Γ19	-9	-8	-17	-13	-10	-11	-13	-5
Γ20	-25	-25	-27	-24	-24	-23	-21	-23

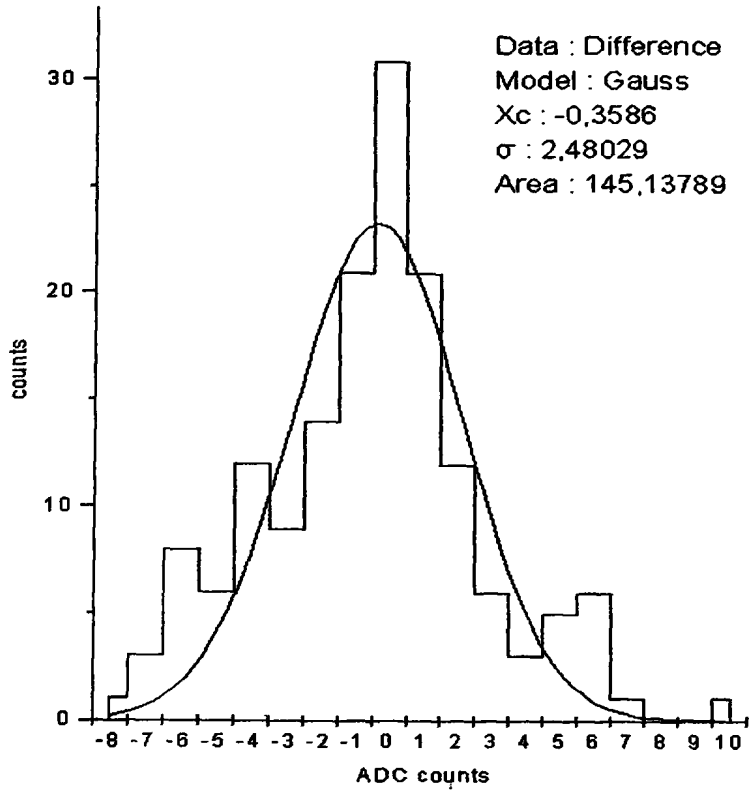
Πίνακας 4: Οι τιμές του κοινού θορύβου.

Οι τιμές που τελικά υπολογίστηκαν από το FPGA για τον κοινό θόρυβο (Common noise) φαίνονται στον πίνακα 5.

Γεγονός #	ADC counts
Δ1	32
Δ2	33
Δ3	-4
Δ4	-2
Δ5	-54
Δ6	4
Δ7	20
Δ8	23
Δ9	-56
Δ10	-26
Δ11	-13
Δ12	5
Δ13	-2
Δ14	4
Δ15	-7
Δ16	-33
Δ17	36
Δ18	48
Δ19	-11
Δ20	-25

Πίνακας 5: Οι τιμές του κοινού θορύβου όπως υπολογίστηκαν από το FPGA.

Στο διάγραμμα 20 φαίνεται το ιστόγραμμα της απόκλισης του κοινού θορύβου που υπολογίσθηκε από τον αρχικό κοινό θόρυβο. Παρατηρούμε ότι είναι μια κανονική κατανομή με μέση τιμή στο μηδέν και με σ ~ 3 ADC counts, γεγονός που μας δείχνει ότι το κύκλωμα υπολογίζει ικανοποιητικά τον κοινό θόρυβο [13].



Διάγραμμα 20: Το ιστόγραμμα της διαφοράς του κοινού θορύβου που υπολογίστηκε από το FPGA με τον κοινό θόρυβο που υπήρχε στην είσοδο.

Το κύκλωμα μπορεί να λειτουργήσει με ρόλοι μέχρι 80 MHz. Στον παραπάνω έλεγχο το ρόλοι που χρησιμοποιήθηκε είχε συχνότητα 50 MHz. Το κύκλωμα χρειάζεται 30 κύκλους ρολογιού για να μας δώσει τον κοινό θόρυβο 8 καναλιών στην έξοδο. Εάν το ρολόι είναι 80 MHz για τον υπολογισμό του κοινού θορύβου απαιτούνται 30 x 12.5ns = 375 ns. Εάν τα δεδομένα από τον ADC έρχονται με συχνότητα 20 MHz τότε 8 x 50ns =400 ns είναι ο χρόνος που απαιτείται για να έχουμε νέα πλήρη εγγραφή στη πρώτη θέση της μνήμης FIFO του κυκλώματος. Επομένως το FPGA προλαβαίνει να υπολογίζει τον κοινό θόρυβο και δεν εισάγει καθυστέρηση στο συνολικό κύκλωμα της επεξεργασίας πληροφορίας DAQ.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] CMS, The Compact Muon Solenoide, Letter of Intent CERN/ LHCC 92-3, LHCC/ I1, 1992.

[2] CMS ECAL, Technical Design Report, CERN / CHCC 97-33, 1997.

[3] Instrumentation in High Energy Physics – Silicon microstrip detectors, Anna Peisert, Instituto Nazionale di Fisica Nucleare, Sezione di Padova, 1992.

[4] CMS Preshower, Front-End Readout & Control system, Draft Kloukinas, v 0.2, 2001

[5] K. Kloukinas, Προφορική συζήτηση.

[6] M.D.M. de Fez-Laso, C. Bonaccorso, K. Gill, G. Hall, G. Iles, B. MacEvoy, M. Millmore, A. Potts, M. Raymond (Imperial Coll., London), M. French, L. Jones, P. Murray (Rutherford), J. Matheson, Beam test performance of the APV5 chip, Nucl.Instrum.Meth.A382:533-544,1996.

[7] A. Go, Προφορική συζήτηση.

[8] The student edition of MATLAB, Prentice Hall, Englewood Cliffs, NJ 07632.

[9] Probability and statistics in particle physics, Frodesen-Skjeggestad-Tofte, Universitetsforlaget, 1978

[10] XILINX, Libraries Guide 3.3.06i. (2000).

[11] Digital Electronics, Logic and systems, John D. Kersaw 3rd edition, Delmar Publishers Inc., 1988

[12] XILINX, XILINX prototype Platforms user Guide for VIRTEX and VIRTEX-E FPGAs, 1999

[13] Feature List and Tutorial Manual, Version 6.1, OriginLab Corporation, 2000

[14] VHDL for Designers, Stefan Sjoholm, Lennart Lindh, Prentice Hall, 1997

[15] The Programmable Logic Data Book, XILINX, 1994

[16] XILINX, Virtex-E 1.8 V Field Programmable Gate Arrays (2001)

[17] http://www.xilinx.com/partinfo/databook.htm

[18] http://www.mathworks.com/support/general/doc.shtml

[19] XILINX, Xilinx European Technical Support, Chris Mead, Case No:385364

ΠΑΡΑΡΤΗΜΑΤΑ

ΠΑΡΑΡΤΗΜΑ Α
Παράμετροι των υπομονάδων στο Core Generator του XILINX Foundation.

Στο παρόν παράρτημα παρατίθενται οι παράμετροι του κάθε ηλεκτρονικού εξαρτήματος του κυκλώματος που υλοποιήθηκε στο FPGA. Οι παράμετροι αυτοί συνθέτουν τα αρχεία xxxx.xco.

U121 (ACUM24)

```
# Xilinx CORE Generator 3.1i_ip_update4
# Username = odi
# COREGenPath = C:\Xilinx\coregen
# FoundationPath = C:\Xilinx\
# ProjectPath = C:\withpads
# ExpandedProjectPath = C:\withpads
SET BusFormat = BusFormatAngleBracket
SET SimulationOutputProducts = VHDL Verilog
SET ViewlogicLibraryAlias = ""
SET XilinxFamily = VIRTEX
SET DesignFlow = Schematic
SET FlowVendor = Foundation
SELECT Accumulator VIRTEX Xilinx,_Inc. 4.0
CSET synchronous_settings = clear
CSET port_b_width = 26
CSET component_name = acum24
CSET carry_borrow_input = false
```

CSET output_options = registered
CSET ce_overrides = sync_controls_override_ce
CSET port_b_sign = signed
CSET saturate = false
CSET sync_init_value = 0
CSET carry_borrow_output = false
CSET ce_override_for_bypass = false
CSET port_b_constant_value = 0
CSET port_a_feedback_scaling = 0
CSET async_init_value = 0
CSET operation = add
CSET bypass_sense = active_high
CSET set_clear_priority = clear_overrides_set
CSET output_width = 29
CSET clock_enable = true
CSET bypass = false
CSET asynchronous_settings = none
CSET overflow_output = false
CSET create_rpm = true
CSET port_b_constant = false
GENERATE

U122 (ADC_VALUES)

Xilinx CORE Generator 3.1i_ip_update4
Username = odi
COREGenPath = C:\Xilinx\coregen
FoundationPath = C:\Xilinx\
ProjectPath = C:\withpads

```
# ExpandedProjectPath = C:\withpads
SET BusFormat = BusFormatAngleBracket
SET SimulationOutputProducts = VHDL Verilog
SET ViewlogicLibraryAlias = ""
SET XilinxFamily = VIRTEX
SET DesignFlow = Schematic
SET FlowVendor = Foundation
SELECT Single_Port_Block_Memory VIRTEX Xilinx,_Inc. 3.2
CSET limit_data_pitch = 8
CSET coefficient_file = C:\withpads\ADCS_VALUES.coe
CSET width = 13
CSET register_inputs = false
CSET additional_output_pipe_stages = 0
CSET handshaking_pins = false
CSET component_name = adcs_values
CSET port_configuration = Read_Only
CSET write_mode = Read_After_Write
CSET depth = 8
CSET has_limit_data_pitch = false
CSET init_value = 0
CSET global_init_value = 0
CSET enable_pin = true
CSET init_pin = false
CSET load_init_file = true
GENERATE
```

U108, U90 (ADRESSCHANGE2)

```
# Xilinx CORE Generator 3.1i_ip_update4
```

```
# Username = odi
# COREGenPath = C:\Xilinx\coregen
# FoundationPath = C:\Xilinx\
# ProjectPath = C:\Temp\testodi
# ExpandedProjectPath = C:\Temp\testodi
SET BusFormat = BusFormatAngleBracket
SET SimulationOutputProducts = VHDL Verilog
SET ViewlogicLibraryAlias = ""
SET XilinxFamily = VIRTEX
SET DesignFlow = Schematic
SET FlowVendor = Foundation
SELECT Binary_Counter VIRTEX Xilinx,_Inc. 4.0
CSET threshold_options = registered
CSET synchronous_settings = init
CSET component_name = adresschange2
CSET count_to_value = MAX
CSET threshold_0_value = MAX
CSET count_by_value = 1
CSET ce_overrides = sync_controls_override_ce
CSET count_style = count_by_constant
CSET load = false
CSET threshold_early = true
CSET threshold_1 = false
CSET sync_init_value = 0
CSET threshold_0 = false
CSET ce_override_for_load = false
CSET async_init_value = 0
CSET operation = up
CSET set_clear_priority = clear_overrides_set
CSET output_width = 3
```

CSET clock_enable = true
CSET asynchronous_settings = none
CSET threshold_1_value = MAX
CSET load_sense = active_high
CSET restrict_count = false
CSET create_rpm = true
GENERATE

U143 (ATHROISMA4B)

Xilinx CORE Generator 3.1i_ip_update4
Username = odi
COREGenPath = C:\Xilinx\coregen
FoundationPath = C:\Xilinx\
ProjectPath = C:\withpads
ExpandedProjectPath = C:\withpads
SET BusFormat = BusFormatAngleBracket
SET SimulationOutputProducts = VHDL Verilog
SET ViewlogicLibraryAlias = ""
SET XilinxFamily = VIRTEX
SET DesignFlow = Schematic
SET FlowVendor = Foundation
SELECT Accumulator VIRTEX Xilinx,_Inc. 4.0
CSET synchronous_settings = clear
CSET port_b_width = 13
CSET component_name = athroisma4b
CSET carry_borrow_input = false
CSET output_options = registered
CSET ce_overrides = sync_controls_override_ce

```
CSET port_b_sign = signed
CSET saturate = false
CSET sync_init_value = 0
CSET carry_borrow_output = false
CSET ce_override_for_bypass = false
CSET port_b_constant_value = 0
CSET port_a_feedback_scaling = 0
CSET async_init_value = 0
CSET operation = add
CSET bypass_sense = active_high
CSET set_clear_priority = clear_overrides_set
CSET output_width = 16
CSET clock_enable = true
CSET bypass = false
CSET asynchronous_settings = none
CSET overflow_output = false
CSET create_rpm = true
CSET port_b_constant = false
GENERATE
```

U142 (CHANGER_FORM)

```
# Xilinx CORE Generator 3.1i_ip_update4
# Username = odi
# COREGenPath = C:\Xilinx\coregen
# FoundationPath = C:\Xilinx\
# ProjectPath = C:\withpads
# ExpandedProjectPath = C:\withpads
SET BusFormat = BusFormatAngleBracket
```


```
SET SimulationOutputProducts = VHDL Verilog
SET ViewlogicLibraryAlias = ""
SET XilinxFamily = VIRTEX
SET DesignFlow = Schematic
SET FlowVendor = Foundation
SELECT Adder_Subtracter VIRTEX Xilinx,_Inc. 4.0
CSET synchronous_settings = none
CSET port_a_sign = signed
CSET port_b_width = 32
CSET latency = 1
CSET component_name = changer_form
CSET carry_borrow_input = false
CSET output_options = registered
CSET ce_overrides = sync_controls_override_ce
CSET port_b_sign = signed
CSET sync_init_value = 0
CSET carry_borrow_output = false
CSET ce_override_for_bypass = true
CSET port_b_constant_value = 0
CSET async_init_value = 0
CSET operation = add
CSET bypass_sense = active_high
CSET port_a_width = 29
CSET set_clear_priority = clear_overrides_set
CSET output_width = 32
CSET clock_enable = false
CSET bypass = false
CSET asynchronous_settings = none
CSET overflow_output = false
CSET create_rpm = true
```

CSET port_b_constant = true
GENERATE

U126 (CHOSSER)

Xilinx CORE Generator 3.1i_ip_update4
Username = odi
COREGenPath = C:\Xilinx\coregen
FoundationPath = C:\Xilinx\
ProjectPath = C:\withpads
ExpandedProjectPath = C:\withpads
SET BusFormat = BusFormatAngleBracket
SET SimulationOutputProducts = VHDL Verilog
SET ViewlogicLibraryAlias = ""
SET XilinxFamily = VIRTEX
SET DesignFlow = Schematic
SET FlowVendor = Foundation
SELECT Bus_Multiplexer VIRTEX Xilinx,_Inc. 4.0
CSET synchronous_settings = clear
CSET latency = 1
CSET component_name = chooser
CSET output_options = registered
CSET ce_overrides = sync_controls_override_ce
CSET multiplexer_construction = lut_based
CSET number_of_input_buses = 8
CSET sync_init_value = 0
CSET async_init_value = 0
CSET set_clear_priority = clear_overrides_set
CSET output_enable = false

CSET clock_enable = true
CSET input_bus_width = 12
CSET asynchronous_settings = none
CSET create_rpm = true
GENERATE

U95, U97 (CLEAN_FIFO2BB)

Xilinx CORE Generator 3.1i_ip_update4
Username = odi
COREGenPath = C:\Xilinx\coregen
FoundationPath = C:\Xilinx\
ProjectPath = C:\withpads
ExpandedProjectPath = C:\withpads
SET BusFormat = BusFormatAngleBracket
SET SimulationOutputProducts = VHDL Verilog
SET ViewlogicLibraryAlias = ""
SET XilinxFamily = VIRTEX
SET DesignFlow = Schematic
SET FlowVendor = Foundation
SELECT Dual_Port_Block_Memory VIRTEX Xilinx,_Inc. 3.2
CSET port_b_register_inputs = true
CSET port_a_register_inputs = true
CSET depth_b = 8
CSET depth_a = 8
CSET configuration_port_b = Read_Only
CSET configuration_port_a = Write_Only
CSET component_name = clean_fifo2bb
CSET port_b_init_value = 0000

CSET port_a_init_value = 0000
CSET port_b_enable_pin = false
CSET port_a_enable_pin = false
CSET port_b_additional_output_pipe_stages = 1
77CSET port_a_additional_output_pipe_stages = 1
CSET port_a_init_pin = false
CSET port_b_handshaking_pins = false
CSET port_a_handshaking_pins = false
CSET width_b = 13
CSET width_a = 13
CSET port_b_init_pin = false
CSET global_init_value = 0000
CSET write_mode_port_b = Read_After_Write
CSET write_mode_port_a = Read_After_Write
CSET load_init_file = false
GENERATE

U115 (COMPARATOR_SXTETR)

Xilinx CORE Generator 3.1i_ip_update4
Username = odi
COREGenPath = C:\Xilinx\coregen
FoundationPath = C:\Xilinx\
ProjectPath = C:\Temp\testodi
ExpandedProjectPath = C:\Temp\testodi
SET BusFormat = BusFormatAngleBracket
SET SimulationOutputProducts = VHDL Verilog
SET ViewlogicLibraryAlias = ""
SET XilinxFamily = VIRTEX

SET DesignFlow = Schematic
SET FlowVendor = Foundation
SELECT Comparator VIRTEX Xilinx,_Inc. 4.0
CSET synchronous_settings = clear
CSET component_name = comparator_sxtetr
CSET output_options = registered
CSET ce_overrides = sync_controls_override_ce
CSET power_on_reset_value = 0
CSET port_b_constant_value = 0
CSET operation = a_lt_b
CSET set_clear_priority = clear_overrides_set
CSET input_sign = unsigned
CSET input_width = 32
CSET clock_enable = true
CSET asynchronous_settings = none
CSET create_rpm = true
CSET port_b_constant = false
GENERATE

U134 (COUNT_END)

Xilinx CORE Generator 3.1i_ip_update4
Username = odi
COREGenPath = C:\Xilinx\coregen
FoundationPath = C:\Xilinx\
ProjectPath = C:\withpads
ExpandedProjectPath = C:\withpads
SET BusFormat = BusFormatAngleBracket
SET SimulationOutputProducts = VHDL Verilog

```
SET ViewlogicLibraryAlias = ""
SET XilinxFamily = VIRTEX
SET DesignFlow = Schematic
SET FlowVendor = Foundation
SELECT Binary_Counter VIRTEX Xilinx,_Inc. 4.0
CSET threshold_options = registered
CSET synchronous_settings = clear
CSET component_name = count_end
CSET count_to_value = MAX
CSET threshold_0_value = d
CSET count_by_value = 1
CSET ce_overrides = sync_controls_override_ce
CSET count_style = count_by_constant
CSET load = false
CSET threshold_early = true
CSET threshold_1 = true
CSET sync_init_value = 0
CSET threshold_0 = true
CSET ce_override_for_load = false
CSET async_init_value = 0
CSET operation = up
CSET set_clear_priority = clear_overrides_set
CSET output_width = 4
CSET clock_enable = true
CSET asynchronous_settings = none
CSET threshold_1_value = e
CSET load_sense = active_high
CSET restrict_count = false
CSET create_rpm = true
GENERATE
```

U106 (COUNT_TO_11)

```
# Xilinx CORE Generator 3.1i_ip_update4
# Username = odi
# COREGenPath = C:\Xilinx\coregen
# FoundationPath = C:\Xilinx\
# ProjectPath = C:\withpads
# ExpandedProjectPath = C:\withpads
SET BusFormat = BusFormatAngleBracket
SET SimulationOutputProducts = VHDL Verilog
SET ViewlogicLibraryAlias = ""
SET XilinxFamily = VIRTEX
SET DesignFlow = Schematic
SET FlowVendor = Foundation
SELECT Binary_Counter VIRTEX Xilinx,_Inc. 4.0
CSET threshold_options = registered
CSET synchronous_settings = clear
CSET component_name = count_to_11
CSET count_to_value = MAX
CSET threshold_0_value = d
CSET count_by_value = 1
CSET ce_overrides = sync_controls_override_ce
CSET count_style = count_by_constant
CSET load = false
CSET threshold_early = true
CSET threshold_1 = true
CSET sync_init_value = 0
CSET threshold_0 = true
```

CSET ce_override_for_load = false
CSET async_init_value = 0
CSET operation = up
CSET set_clear_priority = clear_overrides_set
CSET output_width = 4
CSET clock_enable = true
CSET asynchronous_settings = none
CSET threshold_1_value = e
CSET load_sense = active_high
CSET restrict_count = false
CSET create_rpm = true
GENERATE

U100 (COUNT_TO_13)

Xilinx CORE Generator 3.1i_ip_update4
Username = odi
COREGenPath = C:\Xilinx\coregen
FoundationPath = C:\Xilinx\
ProjectPath = C:\testodi
ExpandedProjectPath = C:\testodi
SET BusFormat = BusFormatAngleBracket
SET SimulationOutputProducts = VHDL Verilog
SET ViewlogicLibraryAlias = ""
SET XilinxFamily = VIRTEX
SET DesignFlow = Schematic
SET FlowVendor = Foundation
SELECT Binary_Counter VIRTEX Xilinx,_Inc. 4.0
CSET threshold_options = registered

CSET synchronous_settings = clear
CSET component_name = count_to_13
CSET count_to_value = MAX
CSET threshold_0_value = f
CSET count_by_value = 1
CSET ce_overrides = sync_controls_override_ce
CSET count_style = count_by_constant
CSET load = false
CSET threshold_early = true
CSET threshold_1 = true
CSET sync_init_value = 0
CSET threshold_0 = true
CSET ce_override_for_load = false
CSET async_init_value = 0
CSET operation = up
CSET set_clear_priority = clear_overrides_set
CSET output_width = 5
CSET clock_enable = true
CSET asynchronous_settings = none
CSET threshold_1_value = 11
CSET load_sense = active_high
CSET restrict_count = false
CSET create_rpm = true
GENERATE

U101 (COUNT_TO_14)

Xilinx CORE Generator 3.1i_ip_update4
Username = odi

```
# COREGenPath = C:\Xilinx\coregen
# FoundationPath = C:\Xilinx\
# ProjectPath = C:\withpads
# ExpandedProjectPath = C:\withpads
SET BusFormat = BusFormatAngleBracket
SET SimulationOutputProducts = VHDL Verilog
SET ViewlogicLibraryAlias = ""
SET XilinxFamily = VIRTEX
SET DesignFlow = Schematic
SET FlowVendor = Foundation
SELECT Binary_Counter VIRTEX Xilinx,_Inc. 4.0
CSET threshold_options = registered
CSET synchronous_settings = clear
CSET component_name = count_to_14
CSET count_to_value = MAX
CSET threshold_0_value = c
CSET count_by_value = 1
CSET ce_overrides = sync_controls_override_ce
CSET count_style = count_by_constant
CSET load = false
CSET threshold_early = true
CSET threshold_1 = true
CSET sync_init_value = 0
CSET threshold_0 = true
CSET ce_override_for_load = false
CSET async_init_value = 0
CSET operation = up
CSET set_clear_priority = clear_overrides_set
CSET output_width = 4
CSET clock_enable = true
```

CSET asynchronous_settings = none
CSET threshold_1_value = e
CSET load_sense = active_high
CSET restrict_count = false
CSET create_rpm = true
GENERATE

U102 (COUNT_TO_8BB)

Xilinx CORE Generator 3.1i_ip_update4
Username = odi
COREGenPath = C:\Xilinx\coregen
FoundationPath = C:\Xilinx\
ProjectPath = C:\testodi
ExpandedProjectPath = C:\testodi
SET BusFormat = BusFormatAngleBracket
SET SimulationOutputProducts = VHDL Verilog
SET ViewlogicLibraryAlias = ""
SET XilinxFamily = VIRTEX
SET DesignFlow = Schematic
SET FlowVendor = Foundation
SELECT Binary_Counter VIRTEX Xilinx,_Inc. 4.0
CSET threshold_options = registered
CSET synchronous_settings = init
CSET component_name = count_to_8bb
CSET count_to_value = MAX
CSET threshold_0_value = MAX
CSET count_by_value = 1
CSET ce_overrides = sync_controls_override_ce

CSET count_style = count_by_constant
CSET load = false
CSET threshold_early = true
CSET threshold_1 = false
CSET sync_init_value = 1
CSET threshold_0 = false
CSET ce_override_for_load = false
CSET async_init_value = 1
CSET operation = up
CSET set_clear_priority = clear_overrides_set
CSET output_width = 3
CSET clock_enable = false
CSET asynchronous_settings = none
CSET threshold_1_value = MAX
CSET load_sense = active_high
CSET restrict_count = false
CSET create_rpm = true
GENERATE

U94 (COUNTER6)

Xilinx CORE Generator 3.1i_ip_update4
Username = odi
COREGenPath = C:\Xilinx\coregen
FoundationPath = C:\Xilinx\
ProjectPath = C:\testodi
ExpandedProjectPath = C:\testodi
SET BusFormat = BusFormatAngleBracket
SET SimulationOutputProducts = VHDL Verilog

```
SET ViewlogicLibraryAlias = ""
SET XilinxFamily = VIRTEX
SET DesignFlow = Schematic
SET FlowVendor = Foundation
SELECT Binary_Counter VIRTEX Xilinx,_Inc. 4.0
CSET threshold_options = registered
CSET synchronous_settings = init
CSET component_name = counter6
CSET count_to_value = MAX
CSET threshold_0_value = MAX
CSET count_by_value = 1
CSET ce_overrides = sync_controls_override_ce
CSET count_style = count_by_constant
CSET load = false
CSET threshold_early = true
CSET threshold_1 = false
CSET sync_init_value = 0
CSET threshold_0 = false
CSET ce_override_for_load = false
CSET async_init_value = 0
CSET operation = up
CSET set_clear_priority = clear_overrides_set
CSET output_width = 3
CSET clock_enable = true
CSET asynchronous_settings = none
CSET threshold_1_value = MAX
CSET load_sense = active_high
CSET restrict_count = false
CSET create_rpm = true
GENERATE
```

U125 (COUNTER8)

```
# Xilinx CORE Generator 3.1i_ip_update4
# Username = odi
# COREGenPath = C:\Xilinx\coregen
# FoundationPath = C:\Xilinx\
# ProjectPath = C:\testodi
# ExpandedProjectPath = C:\testodi
SET BusFormat = BusFormatAngleBracket
SET SimulationOutputProducts = VHDL Verilog
SET ViewlogicLibraryAlias = ""
SET XilinxFamily = VIRTEX
SET DesignFlow = Schematic
SET FlowVendor = Foundation
SELECT Binary_Counter VIRTEX Xilinx,_Inc. 4.0
CSET threshold_options = registered
CSET synchronous_settings = init
CSET component_name = counter8
CSET count_to_value = MAX
CSET threshold_0_value = MAX
CSET count_by_value = 1
CSET ce_overrides = sync_controls_override_ce
CSET count_style = count_by_constant
CSET load = false
CSET threshold_early = true
CSET threshold_1 = false
CSET sync_init_value = 1
CSET threshold_0 = false
CSET ce_override_for_load = false
```

CSET async_init_value = 1
CSET operation = up
CSET set_clear_priority = clear_overrides_set
CSET output_width = 3
CSET clock_enable = true
CSET asynchronous_settings = none
CSET threshold_1_value = MAX
CSET load_sense = active_high
CSET restrict_count = false
CSET create_rpm = true
GENERATE

U136 (COUNTER10)

Xilinx CORE Generator 3.1i_ip_update4
Username = odi
COREGenPath = C:\Xilinx\coregen
FoundationPath = C:\Xilinx\
ProjectPath = C:\withpads
ExpandedProjectPath = C:\withpads
SET BusFormat = BusFormatAngleBracket
SET SimulationOutputProducts = VHDL Verilog
SET ViewlogicLibraryAlias = ""
SET XilinxFamily = VIRTEX
SET DesignFlow = Schematic
SET FlowVendor = Foundation
SELECT Binary_Counter VIRTEX Xilinx,_Inc. 4.0
CSET threshold_options = registered
CSET synchronous_settings = init

```
CSET component_name = counter10
CSET count_to_value = MAX
CSET threshold_0_value = 7
CSET count_by_value = 1
CSET ce_overrides = sync_controls_override_ce
CSET count_style = count_by_constant
CSET load = false
CSET threshold_early = true
CSET threshold_1 = false
CSET sync_init_value = 0
CSET threshold_0 = true
CSET ce_override_for_load = false
CSET async_init_value = 0
CSET operation = up
CSET set_clear_priority = clear_overrides_set
CSET output_width = 4
CSET clock_enable = true
CSET asynchronous_settings = none
CSET threshold_1_value = MAX
CSET load_sense = active_high
CSET restrict_count = false
CSET create_rpm = true
GENERATE
```

U104 (DUALL5BB)

```
# Xilinx CORE Generator 3.1i_ip_update4
# Username = odi
# COREGenPath = C:\Xilinx\coregen
```

```
# FoundationPath = C:\Xilinx\
# ProjectPath = C:\testodi
# ExpandedProjectPath = C:\testodi
SET BusFormat = BusFormatAngleBracket
SET SimulationOutputProducts = VHDL Verilog
SET ViewlogicLibraryAlias = ""
SET XilinxFamily = VIRTEX
SET DesignFlow = Schematic
SET FlowVendor = Foundation
SELECT Dual_Port_Block_Memory VIRTEX Xilinx,_Inc. 3.2
CSET port_b_register_inputs = true
CSET port_a_register_inputs = true
CSET depth_b = 8
CSET depth_a = 8
CSET configuration_port_b = Read_Only
CSET configuration_port_a = Write_Only
CSET component_name = dual5bb
CSET port_b_init_value = 00000000
CSET port_a_init_value = 00000000
CSET port_b_enable_pin = false
CSET port_a_enable_pin = false
CSET port_b_additional_output_pipe_stages = 1
CSET port_a_additional_output_pipe_stages = 1
CSET port_a_init_pin = false
CSET port_b_handshaking_pins = false
CSET port_a_handshaking_pins = false
CSET width_b = 32
CSET width_a = 32
CSET port_b_init_pin = false
CSET global_init_value = 00000000
```

CSET write_mode_port_b = Read_After_Write
CSET write_mode_port_a = Read_After_Write
CSET load_init_file = false
GENERATE

U131 (END)

Xilinx CORE Generator 3.1i_ip_update4
Username = odi
COREGenPath = C:\Xilinx\coregen
FoundationPath = C:\Xilinx\
ProjectPath = C:\testodi
ExpandedProjectPath = C:\testodi
SET BusFormat = BusFormatAngleBracket
SET SimulationOutputProducts = VHDL Verilog
SET ViewlogicLibraryAlias = ""
SET XilinxFamily = VIRTEX
SET DesignFlow = Schematic
SET FlowVendor = Foundation
SELECT Single_Port_Block_Memory VIRTEX Xilinx,_Inc. 3.2
CSET limit_data_pitch = 8
CSET width = 12
CSET register_inputs = true
CSET additional_output_pipe_stages = 0
CSET handshaking_pins = false
CSET component_name = end
CSET port_configuration = Read_And_Write
CSET write_mode = Read_After_Write
CSET depth = 2

CSET has_limit_data_pitch = false
CSET init_value = 0
CSET global_init_value = 0
CSET enable_pin = false
CSET init_pin = true
CSET load_init_file = false
GENERATE

U127 (EPI9)

Xilinx CORE Generator 3.1i_ip_update4
Username = odi
COREGenPath = C:\Xilinx\coregen
FoundationPath = C:\Xilinx\
ProjectPath = C:\withpads
ExpandedProjectPath = C:\withpads
SET BusFormat = BusFormatAngleBracket
SET SimulationOutputProducts = VHDL Verilog
SET ViewlogicLibraryAlias = ""
SET XilinxFamily = VIRTEX
SET DesignFlow = Schematic
SET FlowVendor = Foundation
SELECT Multiplier VIRTEX Xilinx,_Inc. 3.1
CSET asynchronous_clear = false
CSET port_b_width = 4
CSET component_name = epi9
CSET output_options = Registered
CSET output_hold_register = false
CSET ce_overrides = CE_Overrides_SCLR

CSET port_b_constant_value = 9
CSET rfd = false
CSET register_input = true
CSET pipelined = Minimum
CSET reload_options = Stop_During_Reload
CSET memory_type = Dual_Port_Block_Memory
CSET port_a_input = Parallel
CSET port_a_width = 16
CSET synchronous_clear = true
CSET port_a_data = Signed
CSET output_width = 20
CSET reloadable = false
CSET clk_cycles_per_input = 1
CSET nd = false
CSET clock_enable = true
CSET port_b_data = Unsigned
CSET multiplier_construction = Use_LUTs
CSET create_rpm = true
CSET style = Optimal_Packing
CSET multiplier_type = Parallel
CSET rdy = false
CSET load_done_output = false
CSET port_b_constant = true
GENERATE

U128 (EPI11)

Xilinx CORE Generator 3.1i_ip_update4
Username = odi

```
# COREGenPath = C:\Xilinx\coregen
# FoundationPath = C:\Xilinx\
# ProjectPath = C:\withpads
# ExpandedProjectPath = C:\withpads
SET BusFormat = BusFormatAngleBracket
SET SimulationOutputProducts = VHDL Verilog
SET ViewlogicLibraryAlias = ""
SET XilinxFamily = VIRTEX
SET DesignFlow = Schematic
SET FlowVendor = Foundation
SELECT Multiplier VIRTEX Xilinx,_Inc. 3.1
CSET asynchronous_clear = false
CSET port_b_width = 4
CSET component_name = epi11
CSET output_options = Registered
CSET output_hold_register = false
CSET ce_overrides = CE_Overrides_SCLR
CSET port_b_constant_value = 11
CSET rfd = false
CSET register_input = true
CSET pipelined = Minimum
CSET reload_options = Stop_During_Reload
CSET memory_type = Dual_Port_Block_Memory
CSET port_a_input = Parallel
CSET port_a_width = 16
CSET synchronous_clear = true
CSET port_a_data = Signed
CSET output_width = 20
CSET reloadable = false
CSET clk_cycles_per_input = 1
```

CSET nd = false
CSET clock_enable = true
CSET port_b_data = Unsigned
CSET multiplier_construction = Use_LUTs
CSET create_rpm = true
CSET style = Optimal_Packing
CSET multiplier_type = Parallel
CSET rdy = false
CSET load_done_output = false
CSET port_b_constant = true
GENERATE

U129 (EPI13)

Xilinx CORE Generator 3.1i_ip_update4
Username = odi
COREGenPath = C:\Xilinx\coregen
FoundationPath = C:\Xilinx\
ProjectPath = C:\withpads
ExpandedProjectPath = C:\withpads
SET BusFormat = BusFormatAngleBracket
SET SimulationOutputProducts = VHDL Verilog
SET ViewlogicLibraryAlias = ""
SET XilinxFamily = VIRTEX
SET DesignFlow = Schematic
SET FlowVendor = Foundation
SELECT Multiplier VIRTEX Xilinx,_Inc. 3.1
CSET asynchronous_clear = false
CSET port_b_width = 4

```
CSET component_name = epi13
CSET output_options = Registered
CSET output_hold_register = false
CSET ce_overrides = CE_Overrides_SCLR
CSET port_b_constant_value = 13
CSET rfd = false
CSET register_input = true
CSET pipelined = Minimum
CSET reload_options = Stop_During_Reload
CSET memory_type = Dual_Port_Block_Memory
CSET port_a_input = Parallel
CSET port_a_width = 16
CSET synchronous_clear = true
CSET port_a_data = Signed
CSET output_width = 20
CSET reloadable = false
CSET clk_cycles_per_input = 1
CSET nd = false
CSET clock_enable = true
CSET port_b_data = Unsigned
CSET multiplier_construction = Use_LUTs
CSET create_rpm = true
CSET style = Optimal_Packing
CSET multiplier_type = Parallel
CSET rdy = false
CSET load_done_output = false
CSET port_b_constant = true
GENERATE
```

U91 (EPI21BB)

```
# Xilinx CORE Generator 3.1i_ip_update4
# Username = odi
# COREGenPath = C:\Xilinx\coregen
# FoundationPath = C:\Xilinx\
# ProjectPath = C:\withpads
# ExpandedProjectPath = C:\withpads
SET BusFormat = BusFormatAngleBracket
SET SimulationOutputProducts = VHDL Verilog
SET ViewlogicLibraryAlias = ""
SET XilinxFamily = VIRTEX
SET DesignFlow = Schematic
SET FlowVendor = Foundation
SELECT Multiplier VIRTEX Xilinx,_Inc. 3.1
CSET asynchronous_clear = false
CSET port_b_width = 5
CSET component_name = epi21bb
CSET output_options = Registered
CSET output_hold_register = false
CSET ce_overrides = CE_Overrides_SCLR
CSET port_b_constant_value = 21
CSET rfd = false
CSET register_input = true
CSET pipelined = Minimum
CSET reload_options = Stop_During_Reload
CSET memory_type = Dual_Port_Block_Memory
CSET port_a_input = Parallel
CSET port_a_width = 16
CSET synchronous_clear = true
```

CSET port_a_data = Signed
CSET output_width = 21
CSET reloadable = false
CSET clk_cycles_per_input = 1
CSET nd = false
CSET clock_enable = true
CSET port_b_data = Unsigned
CSET multiplier_construction = Use_LUTs
CSET create_rpm = true
CSET style = Optimal_Packing
CSET multiplier_type = Parallel
CSET rdy = false
CSET load_done_output = false
CSET port_b_constant = true
GENERATE

U124 (KAPA)

Xilinx CORE Generator 3.1i_ip_update4
Username = odi
COREGenPath = C:\Xilinx\coregen
FoundationPath = C:\Xilinx\
ProjectPath = C:\Temp\testodi
ExpandedProjectPath = C:\Temp\testodi
SET BusFormat = BusFormatAngleBracket
SET SimulationOutputProducts = VHDL Verilog
SET ViewlogicLibraryAlias = ""
SET XilinxFamily = VIRTEX
SET DesignFlow = Schematic

```
SET FlowVendor = Foundation
SELECT Accumulator VIRTEX Xilinx,_Inc. 4.0
CSET synchronous_settings = clear
CSET port_b_width = 1
CSET component_name = kapa
CSET carry_borrow_input = true
CSET output_options = registered
CSET ce_overrides = sync_controls_override_ce
CSET port_b_sign = unsigned
CSET saturate = false
CSET sync_init_value = 0
CSET carry_borrow_output = true
CSET ce_override_for_bypass = false
CSET port_b_constant_value = 0
CSET port_a_feedback_scaling = 0
CSET async_init_value = 0
CSET operation = add
CSET bypass_sense = active_high
CSET set_clear_priority = clear_overrides_set
CSET output_width = 3
CSET clock_enable = true
CSET bypass = false
CSET asynchronous_settings = none
CSET overflow_output = false
CSET create_rpm = true
CSET port_b_constant = false
GENERATE
```

U96 (METRITIS_A)

```
# Xilinx CORE Generator 3.1i_ip_update4
# Username = odi
# COREGenPath = C:\Xilinx\coregen
# FoundationPath = C:\Xilinx\
# ProjectPath = C:\withpads
# ExpandedProjectPath = C:\withpads
SET BusFormat = BusFormatAngleBracket
SET SimulationOutputProducts = VHDL Verilog
SET ViewlogicLibraryAlias = ""
SET XilinxFamily = VIRTEX
SET DesignFlow = Schematic
SET FlowVendor = Foundation
SELECT Binary_Counter VIRTEX Xilinx,_Inc. 4.0
CSET threshold_options = registered
CSET synchronous_settings = clear
CSET component_name = metritis_a
CSET count_to_value = MAX
CSET threshold_0_value = 9
CSET count_by_value = 1
CSET ce_overrides = sync_controls_override_ce
CSET count_style = count_by_constant
CSET load = false
CSET threshold_early = true
CSET threshold_1 = false
CSET sync_init_value = 0
CSET threshold_0 = true
CSET ce_override_for_load = false
CSET async_init_value = 0
```

CSET operation = up
CSET set_clear_priority = clear_overrides_set
CSET output_width = 4
CSET clock_enable = true
CSET asynchronous_settings = none
CSET threshold_1_value = MAX
CSET load_sense = active_high
CSET restrict_count = false
CSET create_rpm = true
GENERATE

U141 (MLT56)

Xilinx CORE Generator 3.1i_ip_update4
Username = odi
COREGenPath = C:\Xilinx\coregen
FoundationPath = C:\Xilinx\
ProjectPath = C:\withpads
ExpandedProjectPath = C:\withpads
SET BusFormat = BusFormatAngleBracket
SET SimulationOutputProducts = VHDL Verilog
SET ViewlogicLibraryAlias = ""
SET XilinxFamily = VIRTEX
SET DesignFlow = Schematic
SET FlowVendor = Foundation
SELECT Multiplier VIRTEX Xilinx,_Inc. 3.1
CSET asynchronous_clear = false
CSET port_b_width = 6
CSET component_name = mlt56

```
CSET output_options = Registered
CSET output_hold_register = false
CSET ce_overrides = CE_Overrides_SCLR
CSET port_b_constant_value = 56
CSET rfd = false
CSET register_input = true
CSET pipelined = Minimum
CSET reload_options = Stop_During_Reload
CSET memory_type = Dual_Port_Block_Memory
CSET port_a_input = Parallel
CSET port_a_width = 26
CSET synchronous_clear = true
CSET port_a_data = Signed
CSET output_width = 32
CSET reloadable = false
CSET clk_cycles_per_input = 1
CSET nd = false
CSET clock_enable = false
CSET port_b_data = Unsigned
CSET multiplier_construction = Use_LUTs
CSET create_rpm = true
CSET style = Optimal_Packing
CSET multiplier_type = Parallel
CSET rdy = false
CSET load_done_output = false
CSET port_b_constant = true
GENERATE
```

U110, U140 (MULTI2)

```
# Xilinx CORE Generator 3.1i_ip_update4
# Username = odi
# COREGenPath = C:\Xilinx\coregen
# FoundationPath = C:\Xilinx\
# ProjectPath = C:\withpads
# ExpandedProjectPath = C:\withpads
SET BusFormat = BusFormatAngleBracket
SET SimulationOutputProducts = VHDL Verilog
SET ViewlogicLibraryAlias = ""
SET XilinxFamily = VIRTEX
SET DesignFlow = Schematic
SET FlowVendor = Foundation
SELECT Multiplier VIRTEX Xilinx,_Inc. 3.1
CSET asynchronous_clear = false
CSET port_b_width = 13
CSET component_name = multi2
CSET output_options = Registered
CSET output_hold_register = false
CSET ce_overrides = CE_Overrides_SCLR
CSET port_b_constant_value = 1
CSET rfd = false
CSET register_input = true
CSET pipelined = Minimum
CSET reload_options = Stop_During_Reload
CSET memory_type = Distributed_Memory
CSET port_a_input = Parallel
CSET port_a_width = 13
CSET synchronous_clear = true
```

CSET port_a_data = Signed
CSET output_width = 26
CSET reloadable = false
CSET clk_cycles_per_input = 1
CSET nd = false
CSET clock_enable = true
CSET port_b_data = Signed
CSET multiplier_construction = Use_LUTs
CSET create_rpm = true
CSET style = Optimal_Packing
CSET multiplier_type = Parallel
CSET rdy = false
CSET load_done_output = false
CSET port_b_constant = false
GENERATE

U137 (NEWSUB1)

Xilinx CORE Generator 3.1i_ip_update4
Username = odi
COREGenPath = C:\Xilinx\coregen
FoundationPath = C:\Xilinx\
ProjectPath = C:\withpads
ExpandedProjectPath = C:\withpads
SET BusFormat = BusFormatAngleBracket
SET SimulationOutputProducts = VHDL Verilog
SET ViewlogicLibraryAlias = ""
SET XilinxFamily = VIRTEX
SET DesignFlow = Schematic

```
SET FlowVendor = Foundation
SELECT Adder_Subtracter VIRTEX Xilinx,_Inc. 4.0
CSET synchronous_settings = clear
CSET port_a_sign = signed
CSET port_b_width = 9
CSET latency = 1
CSET component_name = newsub1
CSET carry_borrow_input = false
CSET output_options = registered
CSET ce_overrides = sync_controls_override_ce
CSET port_b_sign = signed
CSET sync_init_value = 0
CSET carry_borrow_output = false
CSET ce_override_for_bypass = true
CSET port_b_constant_value = 0
CSET async_init_value = 0
CSET operation = subtract
CSET bypass_sense = active_high
CSET port_a_width = 13
CSET set_clear_priority = clear_overrides_set
CSET output_width = 13
CSET clock_enable = true
CSET bypass = false
CSET asynchronous_settings = none
CSET overflow_output = false
CSET create_rpm = true
CSET port_b_constant = false
GENERATE
```

U139 (NEW_ATHROISMA_3)

```
# Xilinx CORE Generator 3.1i_ip_update4
# Username = odi
# COREGenPath = C:\Xilinx\coregen
# FoundationPath = C:\Xilinx\
# ProjectPath = C:\withpads
# ExpandedProjectPath = C:\withpads
SET BusFormat = BusFormatAngleBracket
SET SimulationOutputProducts = VHDL Verilog
SET ViewlogicLibraryAlias = ""
SET XilinxFamily = VIRTEX
SET DesignFlow = Schematic
SET FlowVendor = Foundation
SELECT Accumulator VIRTEX Xilinx,_Inc. 4.0
CSET synchronous_settings = clear
CSET port_b_width = 13
CSET component_name = new_athroisma_3
CSET carry_borrow_input = false
CSET output_options = registered
CSET ce_overrides = sync_controls_override_ce
CSET port_b_sign = signed
CSET saturate = false
CSET sync_init_value = 0
CSET carry_borrow_output = false
CSET ce_override_for_bypass = false
CSET port_b_constant_value = 0
CSET port_a_feedback_scaling = 0
CSET async_init_value = 0
CSET operation = add
```

CSET bypass_sense = active_high
CSET set_clear_priority = clear_overrides_set
CSET output_width = 16
CSET clock_enable = true
CSET bypass = false
CSET asynchronous_settings = none
CSET overflow_output = false
CSET create_rpm = true
CSET port_b_constant = false
GENERATE

U130 (PEDES_VALUES)

Xilinx CORE Generator 3.1i_ip_update4
Username = odi
COREGenPath = C:\Xilinx\coregen
FoundationPath = C:\Xilinx\
ProjectPath = C:\withpads
ExpandedProjectPath = C:\withpads
SET BusFormat = BusFormatAngleBracket
SET SimulationOutputProducts = VHDL Verilog
SET ViewlogicLibraryAlias = ""
SET XilinxFamily = VIRTEX
SET DesignFlow = Schematic
SET FlowVendor = Foundation
SELECT Single_Port_Block_Memory VIRTEX Xilinx,_Inc. 3.2
CSET limit_data_pitch = 8
CSET coefficient_file = C:\withpads\pedes_values.coe
CSET width = 9

CSET register_inputs = false
CSET additional_output_pipe_stages = 0
CSET handshaking_pins = false
CSET component_name = pedes_values
CSET port_configuration = Read_Only
CSET write_mode = Read_After_Write
CSET depth = 8
CSET has_limit_data_pitch = false
CSET init_value = 0
CSET global_init_value = 0
CSET enable_pin = true
CSET init_pin = false
CSET load_init_file = true
GENERATE

U98 (SUBSTRACTOR3)

Xilinx CORE Generator 3.1i_ip_update4
Username = odi
COREGenPath = C:\Xilinx\coregen
FoundationPath = C:\Xilinx\
ProjectPath = C:\withpads
ExpandedProjectPath = C:\withpads
SET BusFormat = BusFormatAngleBracket
SET SimulationOutputProducts = VHDL Verilog
SET ViewlogicLibraryAlias = ""
SET XilinxFamily = VIRTEX
SET DesignFlow = Schematic
SET FlowVendor = Foundation

```
SELECT Adder_Subtracter VIRTEX Xilinx,_Inc. 4.0
CSET synchronous_settings = clear
CSET port_a_sign = signed
CSET port_b_width = 13
CSET latency = 1
CSET component_name = substractor3
CSET carry_borrow_input = false
CSET output_options = registered
CSET ce_overrides = sync_controls_override_ce
CSET port_b_sign = signed
CSET sync_init_value = 0
CSET carry_borrow_output = false
CSET ce_override_for_bypass = true
CSET port_b_constant_value = 0
CSET async_init_value = 0
CSET operation = subtract
CSET bypass_sense = active_high
CSET port_a_width = 13
CSET set_clear_priority = clear_overrides_set
CSET output_width = 13
CSET clock_enable = true
CSET bypass = false
CSET asynchronous_settings = none
CSET overflow_output = false
CSET create_rpm = true
CSET port_b_constant = false
GENERATE
```

ΠΑΡΑΡΤΗΜΑ Β
Κώδικας VHDL αριθμητικών πράξεων και πολλαπλασιασμού της συχνότητας. [14]

Στο αρχικό στάδιο της διπλωματικής εργασίας προκειμένου να υλοποιηθεί ο αλγόριθμος στο FPGA σε κώδικα VHDL αναπτύχθηκαν ορισμένες υπομονάδες του κυκλώματος σε VHDL. Ωστόσο λόγω της συγκεκριμένης έκδοσης του Virtex-E κρίθηκε αναγκαίο να χρησιμοποιηθεί τελικά ο Core Generator. Παρακάτω παρατίθενται οι υπομονάδες που γράφτηκαν σε κώδικα VHDL.

Αφαιρέτης που αφαιρεί πάντα τον μικρότερο από τον μεγαλύτερο αριθμό ώστε το αποτέλεσμα να είναι πάντα η απόλυτη τιμή.

Στην είσοδο της εν λόγω υπομονάδας είναι 2 αριθμοί των 8 και 12 bits και το αποτέλεσμα στην έξοδό της είναι και αυτό των 12 bits.

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity component_1 is
   port (
      a: in STD_LOGIC_VECTOR (7 downto 0);
      b: in STD_LOGIC_VECTOR (11 downto 0);
                              d: out bit;
      c: out STD_LOGIC_VECTOR (11 downto 0)
   );
```

```
end component_1;

architecture component_1 of component_1 is
begin
 process (a, b)
 variable temp :STD_LOGIC_VECTOR (11 downto 0);
 variable a1 :STD_LOGIC_VECTOR (11 downto 0);
  begin
       a1 :=("0000" & a);
                                  if(a1<=b) then d<='0';
                                  temp:= b - a1;
                                     else d<='1';
                                      temp:= a1 - b;
                                      end if;
                                  c <= temp(11 downto 0);
                                  end process;
end component_1;
```

Υπολογισμός του μέσου όρου από 8 αριθμούς.

Στην είσοδο της εν λόγω υπομονάδας είναι ένας 12 bits αριθμός a και το σήμα b1 που δείχνει το πρόσιμό του. Σε κάθε παλμό ρολογιού προστίθεται ή αφαιρείται ο a στο προηγούμενο άθροισμα. Μόλις περάσουν 8 κύκλοι ρολογιού εξάγεται η μέση τιμή και το πρόσημό της.

```
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.std_logic_unsigned.ALL;
```


```
Entity mesos_oros is
port(clk, resetn: in std_logic;
                                    a: in std_logic_vector(11 downto 0);
                                   b1: in bit;
                                    met:out std_logic_vector(3 downto 0);
     total_sum: out std_logic_vector(11 downto 0);
     b2: out bit);
end;

Architecture rtl of mesos_oros is
        signal prosimo :bit;
         signal count :std_logic_vector(14 downto 0);
       signal metritis :std_logic_vector(3 downto 0);
begin
       process(clk,resetn)
       variable a1 :std_logic_vector(12 downto 0);
       begin
       a1 :=("000" & a);
        if resetn='0' then
                              count <=(others=>'0');
                              prosimo<='0';
                              metritis<=(others=>'0');
      elsif clk'event and clk='1' then
                              if metritis=8 then
                               count<=(others=>'0');
                               prosimo<='0';
                               metritis<=(others=>'0');
                              else
                               metritis<=metritis+1;
```

```
                    if (b1='0' and prosimo='0') or (b1='1' and
prosimo='1') then
                        count <= count + a1;
                    elsif b1='0' and prosimo='1' then
                        if (count <= a1) then
                            prosimo<='0';
                            count<=a1 - count;
                        else
                            prosimo<='1';
                            count<=count - a1;
                        end if;
                    else
                        if (count <= a1) then
                            prosimo<='1';
                            count<=a1 - count;
                        else
                            prosimo<='0';
                            count<=count - a1;
                            end if;
                        end if;
                    end if;
                end if;

                end process;

                total_sum<=count(14 downto 3);
                b2<=prosimo;
                met<=metritis;
                end;
```

Μετρητής updown

Ένας μετρητής που ανάλογα με τη κατάσταση που βρίσκεται το σήμα up μετράει προς τα επάνω ή προς τα κάτω.

```
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.std_logic_unsigned.ALL;

Entity up_down is
port(clk, resetn: in std_logic;
  count_en, up : in std_logic;
        sum: out std_logic_vector(2 downto 0);
       cout: out std_logic);
end;

Architecture rtl of up_down is
signal count:std_logic_vector(2 downto 0);
begin
          process(clk,resetn)
          begin
           if resetn='0' then
           count<=(others=>'0');
           elsif clk'event and clk='1' then
           if count_en='1' then
           case up is
           when '1' =>count<=count+1;
           when others =>count<=count-1;
           end case;
```

```
end if;
end if;
end process;
sum<=count;
cout<='1' when count_en='1' and ((up='1' and count=7) or
(up='0' and count=0)) else '0';
end;
```

DLL 2X and 4X Παράδειγμα πολλαπλασιασμού της συχνότητας.

Πρόκειται για ένα πολλαπλασιαστή της συχνότητας ο οποίος χρειάζεται στην είσοδό του τουλάχιστον 25 MHz. Χρησιμοποιεί κάποιες υπομονάδες που υπάρχουν στο συγκεκριμένο Virtex-E και δίνει τη δυνατότητα επιλογής x2 ή x4 καθώς και διαφοράς φάσης κατά π/4, π/2, 3π/2. Επίσης παρέχει επιλογή διαίρεσης της συχνότητας σε βήματα των 1.5, 2, 2.5, 3, 4, 5, 8, 16.

```
library ieee;
use ieee.std_logic_1164.all;
-- pragma translate_off
library unisim;
use unisim.vcomponents.all;
-- pragma translate_on
entity dll_standard is
   port (CLKIN : in  std_logic;
         RESET : in  std_logic;
         CLK2X : out std_logic;
         CLK4X : out std_logic;
         LOCKED: out std_logic);
```

```
end dll_standard;

architecture structural of dll_standard is

component IBUFG
-- synopsys translate_off
  generic(
    TimingChecksOn: Boolean := DefaultTimingChecksOn;
    InstancePath: STRING := "*";
    Xon: Boolean := DefaultXon;
    MsgOn: Boolean := DefaultMsgOn;
    tpd_I_O          : VitalDelayType01 := (0.100 ns, 0.100 ns);
    tipd_I           : VitalDelayType01 := (0.000 ns, 0.000 ns));

-- synopsys translate_on
  port(
    O                : out STD_ULOGIC;
    I                : in  STD_ULOGIC);
end component;

component IBUF
-- synopsys translate_off
  generic(
    TimingChecksOn: Boolean := DefaultTimingChecksOn;
    InstancePath: STRING := "*";
    Xon: Boolean := DefaultXon;
    MsgOn: Boolean := DefaultMsgOn;
    tpd_I_O          : VitalDelayType01 := (0.100 ns, 0.100 ns);
    tipd_I           : VitalDelayType01 := (0.000 ns, 0.000 ns));
```

```
-- synopsys translate_on
  port(
    O                    :    out  STD_ULOGIC;
    I                    :    in   STD_ULOGIC);
end component;

component CLKDLL
-- synopsys translate_off
   generic ( TimingChecksOn : Boolean := DefaultTimingChecksOn;
         InstancePath : STRING := "*";
         Xon  : Boolean := DefaultXon;
        MsgOn : Boolean := DefaultMsgOn;

        tipd_CLKIN   : VitalDelayType01 := (0.000 ns, 0.000 ns);
        tipd_CLKFB   : VitalDelayType01 := (0.000 ns, 0.000 ns);
        tipd_RST     : VitalDelayType01 := (0.000 ns, 0.000 ns);

        tpd_CLKIN_LOCKED  : VitalDelayType01 := (0.100 ns, 0.100 ns);

        tperiod_CLKIN_POSEDGE    : VitalDelayType  := 0.010 ns;
        MAXPERCLKIN      : time := 100 ns;

        tpw_CLKIN_posedge : VitalDelayType  := 0.010 ns;
        tpw_CLKIN_negedge : VitalDelayType  := 0.010 ns;

        tpw_RST_posedge   : VitalDelayType  := 0.010 ns;

        DUTY_CYCLE_CORRECTION : Boolean := TRUE;
        CLKDV_DIVIDE : real := 2.0);
```

```
-- synopsys translate_on
   port ( CLKIN   : in  std_ulogic := '0';
          CLKFB   : in  std_ulogic := '0';
          RST     : in  std_ulogic := '0';
          CLK0    : out std_ulogic := '0';
          CLK90   : out std_ulogic := '0';
          CLK180  : out std_ulogic := '0';
          CLK270  : out std_ulogic := '0';
          CLK2X   : out std_ulogic := '0';
          CLKDV   : out std_ulogic := '0';
          LOCKED  : out std_ulogic := '0');
end component;
component BUFG
-- synopsys translate_off
  generic(
     TimingChecksOn: Boolean := DefaultTimingChecksOn;
     InstancePath: STRING := "*";
     Xon: Boolean := DefaultXon;
     MsgOn: Boolean := DefaultMsgOn;
     tpd_I_O                 : VitalDelayType01 := (0.100 ns, 0.100 ns);
     tipd_I                  :  VitalDelayType01 := (0.000 ns, 0.000 ns));

-- synopsys translate_on
  port(
     O                       :    out  STD_ULOGIC;
     I                       :    in   STD_ULOGIC);
end component;
component OBUF
-- synopsys translate_off
  generic(
```

```
    TimingChecksOn: Boolean := DefaultTimingChecksOn;
    InstancePath: STRING := "*";
    Xon: Boolean := DefaultXon;
    MsgOn: Boolean := DefaultMsgOn;
    tpd_I_O                 : VitalDelayType01 := (0.100 ns, 0.100 ns);
    tipd_I                  :   VitalDelayType01 := (0.000 ns, 0.000 ns));

-- synopsys translate_on
  port(
    O                   :   out  STD_ULOGIC;
    I                   :   in   STD_ULOGIC);
end component;

component SRL16
-- synopsys translate_off
 generic (
    TimingChecksOn: Boolean := DefaultTimingChecksOn;
    InstancePath: STRING := "*";
    Xon: Boolean := DefaultXon;
    MsgOn: Boolean := DefaultMsgOn;

    -- VITAL input wire delays

    tipd_A0  : VitalDelayType01 := (0.0 ns,  0.0 ns);
    tipd_A1  : VitalDelayType01 := (0.0 ns,  0.0 ns);
    tipd_A2  : VitalDelayType01 := (0.0 ns,  0.0 ns);
    tipd_A3  : VitalDelayType01 := (0.0 ns,  0.0 ns);

    tipd_D   : VitalDelayType01 := (0.0 ns,  0.0 ns);
    tipd_CLK : VitalDelayType01 := (0.0 ns,  0.0 ns);
```

```
    -- VITAL pin-to-pin propagation delays

       tpd_A0_Q  : VitalDelayType01 := (0.1 ns, 0.1 ns);
       tpd_A1_Q  : VitalDelayType01 := (0.1 ns, 0.1 ns);
       tpd_A2_Q  : VitalDelayType01 := (0.1 ns, 0.1 ns);
       tpd_A3_Q  : VitalDelayType01 := (0.1 ns, 0.1 ns);
       tpd_CLK_Q : VitalDelayType01 := (0.1 ns, 0.1 ns);
     -- VITAL setup and hold times

       tsetup_D_CLK_posedge_posedge  : VitalDelayType := 0.01 ns;
       tsetup_D_CLK_negedge_posedge  : VitalDelayType := 0.01 ns;

       thold_D_CLK_posedge_posedge   : VitalDelayType := 0.01 ns;
       thold_D_CLK_negedge_posedge   : VitalDelayType := 0.01 ns;
       -- VITAL minimum pulse width

       tpw_CLK_posedge : VitalDelayType := 0.01 ns;
       tpw_CLK_negedge : VitalDelayType := 0.01 ns;

       INIT : bit_vector := X"0000");

-- synopsys translate_on
  port (D   : in STD_ULOGIC;
        CLK : in STD_ULOGIC;
        A0  : in STD_ULOGIC;
        A1  : in STD_ULOGIC;
        A2  : in STD_ULOGIC;
        A3  : in STD_ULOGIC;
        Q   : out STD_ULOGIC);
```

```
end component;


signal CLKIN_w, RESET_w, CLK2X_dll, CLK2X_g, CLK4X_dll, CLK4X_g :
std_logic;

signal LOCKED2X, LOCKED2X_delay, RESET4X, LOCKED4X_dll :
std_logic;
signal logic1 : std_logic;


begin

logic1 <= '1';

clkpad : IBUFG  port map (I=>CLKIN, O=>CLKIN_w);
rstpad : IBUF   port map (I=>RESET, O=>RESET_w);

dll2x   : CLKDLL port map (CLKIN=>CLKIN_w,     CLKFB=>CLK2X_g,
RST=>RESET_w,
                CLK0=>open,CLK90=>open, LK180=>open, CLK270=>open,

CLK2X=>CLK2X_dll,CLKDV=>open,LOCKED=>LOCKED2X);

clk2xg : BUFG   port map (I=>CLK2X_dll,  O=>CLK2X_g);

rstsrl  :  SRL16      port map (D=>LOCKED2X, CLK=>CLK2X_g,
Q=>LOCKED2X_delay,
                A3=>logic1, A2=>logic1, A1=>logic1, A0=>logic1);
```

```
RESET4X <= not LOCKED2X_delay;

dll4x : CLKDLL port map (CLKIN=>CLK2X_g, CLKFB=>CLK4X_g,
RST=>RESET4X,
                CLK0=>open,CLK90=>open,CLK180=>open, LK270=>open,

CLK2X=>CLK4X_dll,CLKDV=>open,LOCKED=>LOCKED4X_dll);


clk4xg : BUFG  port map (I=>CLK4X_dll,  O=>CLK4X_g);
lckpad : OBUF  port map (I=>LOCKED4X_dll, O=>LOCKED);

CLK2X <= CLK2X_g;
CLK4X <= CLK4X_g;

end structural;
```

ΠΑΡΑΡΤΗΜΑ Γ
Το Λογισμικό Πακέτο Προγραμματισμού ολοκληρωμένων Xilinx. [15]

Ο ψηφιακός σχεδιασμός με χρησιμοποίηση προγραμματιζόμενων ολοκληρωμένων FPGAs (Field Programmable Gate Array) έχει γίνει ευρύτατα διαδεδομένος τα τελευταία χρόνια. Τα ολοκληρωμένα αυτά έχουν πολύ μεγάλη πυκνότητα πυλών και επιτρέπουν την επίτευξη υψηλών συχνοτήτων λειτουργίας. Έχουν πολλές εισόδους – εξόδους, μεγάλο αριθμό από flip-flops και υπάρχουν διάφορες οικογένειες που ανάλογα με τα τεχνικά χαρακτηριστικά τους και το μέγεθός τους ικανοποιούν και τον πιο απαιτητικό χρήστη.

Το λογισμικό πακέτο που χρησιμοποιήθηκε είναι της εταιρίας Xilinx και συγκεκριμένα το XILINX FOUNDATION SERIES 3.1i , Build 3.1.181 στο οποίο έχει γίνει αναβάθμιση στο 3.3i, service pack 8. Επίσης η έκδοση του Core Generator που χρησιμοποιήθηκε είναι η 3.1i_IP_update 4. Ο σχεδιασμός της λογικής του κυκλώματος μπορεί να γίνει με τρεις τρόπους, είτε με σχηματικό διάγραμμα, είτε με κώδικα σε γλώσσα VHDL , είτε με state machines. Φυσικά μπορεί να γίνει και οποιοσδήποτε συνδυασμός των ανωτέρω μεθόδων προγραμματισμού. Υπάρχουν διάφορα εργαλεία που βοηθούν στη σχεδίαση ενός κυκλώματος όπως ο Core Generator ο οποίος παρέχει τυποποιημένες ορισμένες υπομονάδες που είναι πολύ γρηγορότερες από ότι να σχεδιάζονταν από την αρχή και αυτό διότι αξιοποιεί τους πόρους του συγκεκριμένου ολοκληρωμένου κατά το βέλτιστο τρόπο. Αφού γίνει ο σχεδιασμός, ακολουθεί το στάδιο της λειτουργικής προσομοίωσης (functional simulation). Εδώ εντοπίζονται λάθη τα οποία έχουν να κάνουν με τη λογική του κυκλώματος. Στη συνέχεια γίνεται η υλοποίηση και η βελτιστοποίηση (implementation) όπου επιλέγονται προτεραιότητες σχετικά με την ελαχιστοποίηση των απαιτήσεων του κυκλώματος σε χώρο ή σε χρόνο. Με

βάση αυτές κρίνεται στη συνέχεια αν είναι δυνατή η υλοποίηση στο συγκεκριμένο ολοκληρωμένο (fitting) ή είναι απαραίτητο κάποιο FPGA με μεγαλύτερο αριθμό πυλών ενώ παράλληλα ελέγχεται το σχέδιο σε ότι αφορά κομμάτια που δεν χρησιμοποιούνται ή σήματα τα οποία είναι κοινά και απλοποιούνται. Μόλις τελειώσει το implementation συμπληρώνονται αυτόματα κάποια αρχεία .log που δίνουν πληροφορίες σχετικές με τη μέγιστη συχνότητα λειτουργίας της συγκεκριμένης υλοποίησης, το χρόνο καθυστέρησης στις συνδέσεις του σχεδίου, τη θέση και τη λειτουργία του κάθε ακροδέκτη από το ολοκληρωμένο, τυχόν προειδοποιητικά μηνύματα καθώς και στατιστικά στοιχεία για την κάλυψη πόρων του ολοκληρωμένου. Ακολούθως ελέγχεται χρονικά (timing verification) όπου μας δίνει αποτελέσματα για τις χειρότερες συνθήκες λειτουργίας που μπορεί να βρεθεί το ολοκληρωμένο. Τέλος το βήμα που απομένει είναι ο προγραμματισμός του ολοκληρωμένου, πράγμα που γίνεται χρησιμοποιώντας το ειδικό καλώδιο της XILINX και κάποια ειδική βάση ή την πλακέτα πάνω στη οποία πρόκειται να χρησιμοποιηθεί. Επιλέγεται το Hardware Debugger από το Device programming και αφού ανιχνεύει τον τύπο του καλωδίου που χρησιμοποιείται, αρχίζει ο προγραμματισμός.

ΠΑΡΑΡΤΗΜΑ Δ

Τεχνικά χαρακτηριστικά του ολοκληρωμένου VIRTEX-E [16].

Το ολοκληρωμένο που επιλέχτηκε για την υλοποίηση του αλγόριθμου είναι το FPGA XCV100E-PQ240 της σειράς VIRTEXE της XILINX [17]. Η οικογένεια VIRTEX-E αποτελείται από ολοκληρωμένα με αυξημένες δυνατότητες, νέας γενιάς και αρχιτεκτονικής. Βασίζονται στη τεχνολογία 0.18μm, αποτελούνται από 6 επίπεδα μετάλλου και περιέχουν αρκετά μεγάλο αριθμό πυλών.

Table 1: **Virtex-E Field-Programmable Gate Array Family Members**

Device	System Gates	Logic Gates	CLB Array	Logic Cells	Differential I/O Pairs	User I/O	BlockRAM Bits	Distributed RAM Bits
XCV50E	71,693	20,736	16 x 24	1,728	83	176	65,536	24,576
XCV100E	128,236	32,400	20 x 30	2,700	83	196	81,920	38,400
XCV200E	306,393	63,504	28 x 42	5,292	119	284	114,688	75,264
XCV300E	411,955	82,944	32 x 48	6,912	137	316	131,072	98,304
XCV400E	569,952	129,600	40 x 60	10,800	183	404	163,840	153,600
XCV600E	985,882	186,624	48 x 72	15,552	247	512	294,912	221,184
XCV1000E	1,569,178	331,776	64 x 96	27,648	281	660	393,216	393,216
XCV1600E	2,188,742	419,904	72 x 108	34,992	344	724	589,824	497,664
XCV2000E	2,541,952	518,400	80 x 120	43,200	344	804	655,360	614,400
XCV2600E	3,263,755	685,584	92 x 138	57,132	344	804	753,664	812,544
XCV3200E	4,074,387	876,096	104 x 156	73,008	344	804	851,968	1,038,336

Πίνακας 6: Στοιχεία σχετικά με τη χωρητικότητα του FPGA.

Table 3: **Virtex-E Family Maximum User I/O by Device/Package (Excluding Dedicated Clock Pins)**

	XCV50E	XCV100E	XCV200E	XCV300E	XCV400E	XCV600E	XCV1000E	XCV1600E	XCV2000E	XCV2600E	XCV3200E
CS144	94	94	94								
PQ240	158	158	158	158	158						
HQ240						158	158				
BG352		196	260	260							
BG432				316	316	316					
BG560					404	404	404	404	404		
FG256	176	176	176	176							
FG456			284	312							
FG676					404	444					
FG680						512	512	512	512		
FG860							660	660	660		
FG900						512	660	700			
FG1156							660	724	804	804	
CG1156											804

Πίνακας 7: Στοιχεία σχετικά με το package και τον μέγιστο αριθμό ακροδεκτών του FPGA.

Recommended Operating Conditions

Symbol	Description		Min	Max	Units
V_{CCINT}	Internal Supply voltage relative to GND, T_J = 0 °C to +85 °C	Commercial	1.8 – 5%	1.8 + 5%	V
	Internal Supply voltage relative to GND, T_J = –40 °C to +100 °C	Industrial	1.8 – 5%	1.8 + 5%	V
V_{CCO}	Supply voltage relative to GND, T_J = 0 °C to +85 °C	Commercial	1.2	3.6	V
	Supply voltage relative to GND, T_J = –40 °C to +100 °C	Industrial	1.2	3.6	V
T_{IN}	Input signal transition time			250	ns

Πίνακας 8: Στοιχεία σχετικά με τις τάσεις λειτουργίας του FPGA.

Επίσης είναι αξιοσημείωτο ότι τα FPGAs Virtex παρέχουν υψηλές επιδόσεις κάνοντας διαχείριση του ρολογιού τους μέσω της τεχνολογίας DLL (Digital Lock Loop). Με αυτή τη τεχνική είναι ικανές να φτάσουν επίδοση μέχρι 311 MHz.

ΠΑΡΑΡΤΗΜΑ Ε
Κώδικας Matlab.

Παρακάτω παρατίθεται ο κώδικας που δημιουργήθηκε στο πακέτο λογισμικού Matlab. [18]

```
clear all
% PART 1 ----------------------------------------------------
% Pedestals, Common Noise & Adc_values generation.
% Create a matrix (1000,1) of random numbers (gausian distribution)
% to be used as common noise.
commonx=rand(1000,1);
commony=rand(1000,1);
comon=round((commonx+commony-1)*63);
makis2=0;
makis3=0;
makis=0;
% create a matrix (1000,8) of random numbers (gausian distribution)
% with width (~ 7) to be used (including  the comon)
% as common noise for neighbor channels (octave)

commmonx=rand(1000,8);
commmony=rand(1000,8);
commmonxy=round((commmonx+commmony-1)*7);

% create a matrix (1000,8) of random numbers (gausian distribution)
% to be used as pedestals.

pedestalx=rand(1000,8);
```

```
pedestaly=rand(1000,8);
ped=128+((pedestalx+pedestaly-1)*127);
pedestals=round(ped);

% declaration of the matrixes common2 & value

common2=zeros(1000,8);
value=zeros(1000,8);

% begin of a loop for 1000 times

for i=1:1000;

   tyxaioi=rand;

   % with a random number i choose the numberof the real signals.
   % from a poisson distribution with mean equal to one.

   if tyxaioi<=0.368
      signal=0;
   elseif 0.736>=tyxaioi>0.368
      signal=1;
   elseif 0.920>=tyxaioi>0.736
      signal=2;
   elseif 0.981>=tyxaioi>0.92
      signal=3;
   elseif 0.996>=tyxaioi>0.981
      signal=4;
   elseif 0.9994>=tyxaioi>0.996
      signal=5;
```

```
else
   signal=6;
end

% give random values from a uniform distribution [0,1] to channels which
has signals.

s=rand(1,6);
channel(1:8)=0;
if signal==1
   channel(1:1)=s(1:1);
elseif signal==2
   channel(1:2)=s(1:2);
elseif signal==3
   channel(1:3)=s(1:3);
elseif signal==4
   channel(1:4)=s(1:4);
elseif signal==5
   channel(1:5)=s(1:5);
elseif signal==6
   channel(1:6)=s(1:6);
else
end

% random the values of the channels are multiplied by 4095
% The results are the adc values.

adc=round(channel*4095);
adcs=adc';
```

```
% Since i have the adc_values, i take an octave of the pedestals
% for each time (for (i)=1 until 1000).
% begin an internal loop of 8 times for each (i) to extract
% the input value we get each time.

for j=1:8
   common2(i,j)=comon(i)+commmonxy(i,j);
   provalue(i,j)=pedestals(i,j)+common2(i,j);
   value(i,j)=provalue(i,j)+adcs(j);
end

% PART 2----------------------------------------------------
% Now starts the off-line simulation of the algorithm.
% Substraction of the pedestals.

for k=1:8
   clean(i,k)=value(i,k)-pedestals(i,k);
end
i;
end

% calculation of the standard deviation (finalsigma).

mesitimi=(sum(clean'))/8;
mes=mesitimi';
for i1=1:1000
   for j1=1:8
      diafora(i1,j1)=clean(i1,j1)-mes(i1);
   end
   i1;
```

```
end
diafora2=diafora.^2;
sdiafora2=sum(diafora2');
sigma=round(sqrt((sdiafora2)/56));
finalsigma=sigma';

% check if clean is less than b. When it is true,  'HOT' channel doesn't exist

exw=zeros(1000,8);
for i2=1:1000
   n=0;
   b=(finalsigma);
   for j2=1:8
      asd(i2,j2)=b(i2)-(abs(clean(i2,j2)));
      if asd(i2,j2)>0
         exw(i2,j2)=clean(i2,j2);
         n=n+1;
      else
      end
   end
   ne(i2)=n;
   i2;
end
newcommon=sum(exw');
commonnoise=newcommon';
newcommononly=sum(clean');
commononly=newcommononly';
cmm=zeros(1000,1);

% calculation of the common noise mean value for each octave.
```

```
for i3=1:1000
   if ne(i3)==0;
      cmm(i3)=(commononly(i3)/8);
   else
      cmm(i3)=(commonnoise(i3)/ne(i3));
   end
end

% calculation of the difference between the input common noise
% and the one we measure.

for i5=1:1000
   for j5=1:8
   difference(i5,j5)=round(common2(i5,j5)-cmm(i5));
   if difference(i5,j5)>-20
      if difference(i5,j5)<20
         makis2=makis2+1;
      end
   end
   if difference(i5,j5)>-30
      if difference(i5,j5)<30
         makis3=makis3+1;
      end
   end
      if difference(i5,j5)>-10
         if difference(i5,j5)<10
         makis=makis+1;
      end
   end
```

```
end
end

% take a histogram of the difference
difference2=difference(:);
hist(difference2,150)
```

ΠΑΡΑΡΤΗΜΑ ΣΤ

Υπόδειγμα αρχείου .coe για τις αρχικές τιμές των ADCs και Pedestals.

– Τα αρχικά περιεχόμενα των μνημών καθορίζονται σε ένα ξεχωριστό αρχείο το οποίο καλείται αρχείο coe. Για την επιλογή και το φόρτωμα του αρχείου coe επιλέγεται η εντολή Load Init Values στο παράθυρο της Single Port μνήμης, μέσα στο Core Generator του λογισμικού πακέτου προγραμματισμού ολοκληρωμένων Xilinx και χρησιμοποιείται το επιθυμητό αρχείο από το πλαίσιο διαλόγου. Η σύνταξη ενός coe αρχείου, για παράδειγμα μιας 3 επί 16 RAM φαίνεται παρακάτω:

MEMORY_INITIALIZATION_RADIX=16;
MEMORY_INITIALIZATION_VECTOR=123, 456, aaaa;

Καθορίζοντας τις αρχικές τιμές μιας μνήμης σε ένα αρχείο xxxx.coe, μπορούν να χρησιμοποιηθούν οι λέξεις κλειδιά
MEMORY_INITIALIZATION_RADIX
MEMORY_INITIALIZATION_VECTOR
Το MEMORY_INITIALIZATION_VECTOR αποτελείται από την ακολουθία των τιμών χωρισμένων μεταξύ τους με κόμμα. Οποιοσδήποτε αριθμός κενών χαρακτήρων ή και κενή γραμμή μπορεί να χρησιμοποιηθεί για να γίνει πιο ευανάγνωστο το αρχείο. Η μορφή της κάθε τιμής καθορίζεται από την τιμή MEMORY_INITIALIZATION_RADIX η οποία μπορεί να είναι 2, 10 ή 16 (η εξ ορισμού είναι η 10 και αντιστοιχεί σε δεκαδική μορφή). Οι τιμές δεν μπορεί να είναι αρνητικές.

Αφού δημιουργηθεί το αρχείο και αλλαχθεί ώστε να φορτωθούν οι καινούργιες τιμές στο εκτελέσιμο αρχείο xxxx.xco, ενεργοποιείται ο Core Generator. Επιλέγεται η εντολή File -> Execute Command File και επιλέγεται

το συγκεκριμένο αρχείο xxxx.xco. Έτσι ενημερώνεται με τις νέες τιμές. Στη συνέχεια ακολουθεί η γνωστή διαδικασία του implementation και του download.