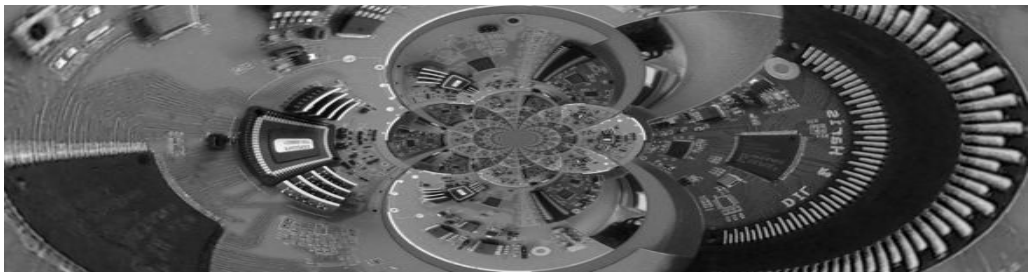
ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

Σχεδίαση ηλεκτρονικών για τον έλεγχο λειτουργίας VLSI κυκλώματος σε διάταξη κυψελίδων (pixel) για ανίχνευση ακτίνων χ

Διπλωματική Εργασία

Αθανάσιος Η. Παπαδημητρίου



Εκπονήθηκε στο Εργαστήριο Οργανολογίας Ανιχνευτών του ΕΚΕΦΕ Δημόκριτος

Επιβλέποντες:

Τσουκαλάς Δημήτριος, Καθηγητής ΣΕΜΦΕ ΕΜΠ
Λουκάς Δημήτριος, Ερευνητής ΕΚΕΦΕ Δημόκριτος

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

Σχεδίαση ηλεκτρονικών για τον έλεγχο λειτουργίας VLSI κυκλώματος σε διάταξη κυψελίδων (pixel) για ανίχνευση ακτίνων χ

Διπλωματική Εργασία

Αθανάσιος Η. Παπαδημητρίου

Επιβλέποντες:

Τσουκαλάς Δημήτριος, Καθηγητής ΣΕΜΦΕ ΕΜΠ
Λουκάς Δημήτριος, Ερευνητής ΕΚΕΦΕ Δημόκριτος

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή

.....	.....	.....
Τσουκαλάς Δημήτριος Καθηγητής ΕΜΠ	Αλεξόπουλος Θεόδωρος Καθηγητής ΕΜΠ	Μαλτέζος Σταύρος Επίκουρος Καθηγητής ΕΜΠ

Αφιερώνεται στην γιαγιά μου
Κατερίνα και στους γονείς μου Ηλία και Ρόη

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον κ. Δημήτριο Τσουκαλά, Καθηγητή ΕΜΠ γιατί με έκανε να αγαπήσω τον κλάδο της Μικροηλεκτρονικής, μέσω της διδασκαλίας και της μεταδοτικότητας του στα σχετικά μαθήματα της ΣΕΜΦΕ. Επιπλέον έδειξε μεγάλο ενδιαφέρον για τα πρώτα μου βήματα στον τομέα αυτό και μου έδωσε την δυνατότητα να ασχοληθώ με το συγκεκριμένο αντικείμενο και να αποκομίσω πολύτιμη εμπειρία.

Στην συνέχεια θα ήθελα να αναφερθώ στον επιβλέποντα μου Δημήτρη Λουκά, Ερευνητή ΕΚΕΦΕ Δημόκριτος και να τον ευχαριστήσω, για την αμέριστη υποστήριξη που μου προσέφερε κατά την εκπόνηση της παρούσας διπλωματικής εργασίας, όχι μόνο γιατί με βοήθησε μέσω της εμπειρίας του και μέσω των γνώσεων του, αλλά και μέσω της παροχής κατάλληλων πόρων και εξαιρετικού κλίματος εργασίας.

Ευχαριστώ τον Χαράλαμπο Λαμπρόπουλο, Αναπληρωτή Καθηγητή ΤΕΙ Χαλκίδας για την βοήθεια του και την εμπιστοσύνη που μου έδειξε κατά την παραμονή μου στο εργαστήριο. Επίσης ευχαριστώ θερμά τους Γιάννη Παπαδάκη και Δημήτρη Χατζηστρατή για την συνεργασία, την βοήθεια αλλά και για την καλή παρέα και τις ωραίες στιγμές που περάσαμε.

Το μεγαλύτερο ευχαριστώ αξίζει η οικογένεια μου, οι γονείς μου Ηλίας και Ρόη και η γιαγιά μου Κατερίνα. Μου είναι πολύ δύσκολο να περιγράψω με λόγια το πόσο με έχουν στηρίξει από την μέρα που γεννήθηκα μέχρι και σήμερα. Ότι έχω καταφέρει μέχρι σήμερα το οφείλω σε αυτούς.

Τέλος θέλω να ευχαριστήσω την σύντροφο μου Ηγησώ για την στήριξη που μου προσέφερε και συνεχίζει να μου προσφέρει, την ιδιαίτερα πιεσμένη αυτή περίοδο και για τα όσα έχουμε μαζί καταφέρει σε τόσο σύντομο χρονικό διάστημα.

Αθήνα Οκτώβριος, 2012

Diploma Thesis Title

Electronics Design to Test the Functionality of a VLSI Circuit for Pixel Level Detection of X-Rays

Abstract

This thesis is part of the effort for the advance in the development of gamma ray imagers based on Cadmium Telluride (CdTe) solid state radiation detectors. The main focus of the thesis is the design of electronics in order to control, test and readout an Application Specific Integrated Circuit (ASIC). The ASIC under test constitutes a charge integrating, pixel level, Analog to Digital Converter (ADC), for use with two dimensional pixilated CdTe radiation sensors. After the integration of the charge originating in the detector, the ASIC performs A/D conversion to the acquired signal by utilizing the Wilkinson technique. The ASIC's control logic, after the conversion, stores the 8bit output word, for each of the 16 pixels, inside a Dynamic RAM and proceeds with data readout using one 8bit bus for each of the two 8-pixel columns.

The testing platform which was implemented includes analog blocks for providing the ASIC with the necessary inputs, in order to emulate the behavior of the detector. Furthermore it feeds the circuit with proper power regulation and includes special circuit blocks for the optimal measurement of the ASIC's circuits designed for testing purposes. In order to control the integrated circuit digital electronics and forward its output to a computer the testing platform utilizes two FPGA devices and an USBv2.0 microcontroller. After the successful implementation of the board level electronics and their own testing, the embedded devices were programmed with the aid of VHDL in the case of the FPGAs and C for the USB microcontroller. As a last step to finalize the testing system, a Graphical User Interface (GUI) was designed to allow the testing with the use of a computer and to store the received data.

By means of this system the testing procedure led to the discovery of both functional and non-functional blocks within the ASIC under test and the characterization of the functional blocks. With minimal redesign in the Hardware and/or Software the system can provide a testing platform for future designs of pixel level analog to digital converters for high energy radiation imaging.

Περιεχόμενα

1 Εισαγωγή

Η παρούσα διπλωματική ασχολείται με την μελέτη και τον έλεγχο συστήματος λήψης δεδομένων από ανιχνευτές ακτίνων χ και γάμμα. Εκπονήθηκε στο εργαστήριο οργανολογίας ανιχνευτών του ινστιτούτου πυρηνικής φυσικής του ΕΚΕΦΕ Δημόκριτος και σκοπός της ήταν η σχεδίαση ηλεκτρονικών για τον έλεγχο λειτουργίας ενός ολοκληρωμένου κυκλώματος υψηλής ολοκλήρωσης και η λήψη δεδομένων από αυτό. Η βασική λειτουργία του ολοκληρωμένου κυκλώματος είναι η λήψη σημάτων από ημιαγώγιμους ανιχνευτές τελουριούχου καδμίου σε τοπολογία κυψελίδας και η μετατροπή των σημάτων από αναλογικά σε ψηφιακά σε επίπεδο pixel. Τα ηλεκτρονικά που υλοποιήθηκαν απαρτίζονται από τρία διαφορετικά τυπωμένα κυκλώματα και συνδέουν το ολοκληρωμένο προς έλεγχο με ηλεκτρονικό υπολογιστή μέσω του Universal Serial Bus. Η υλοποίηση περιλαμβάνει αναλογικά και ψηφιακά ηλεκτρονικά. Τα αναλογικά που χρειάστηκαν αφορούν την παροχή κατάλληλων σημάτων στο ολοκληρωμένο, απαραίτητων για την διαδικασία της μετατροπής του αναλογικού σήματος σε ψηφιακό. Για τα ψηφιακά ηλεκτρονικά χρησιμοποιήθηκαν ενσωματωμένα προγραμματιζόμενα συστήματα, όπως FPGA και USB μικροελεγκτής. Η σχεδίαση στα δύο FPGA της διάταξης πραγματοποιήθηκε με την χρήση της γλώσσας περιγραφής υλικού VHDL και ο προγραμματισμός του USB μικροελεγκτή με χρήση της γλώσσας C. Για τον έλεγχο της διάταξης από ηλεκτρονικό υπολογιστή σχεδιάστηκε γραφικό περιβάλλον με την γλώσσα προγραμματισμού C# (C sharp). Ιδιαίτερη προσοχή δόθηκε στην επιλογή των ολοκληρωμένων του συστήματος ελέγχου, με σκοπό την ελαχιστοποίηση του θορύβου, την υλοποίηση του σε μορφή επεκτάσιμης πλατφόρμας μετρήσεων και λήψης δεδομένων και την όσο το δυνατόν ελαχιστοποίηση του κόστους.

Στο δεύτερο κεφάλαιο περιγράφεται η φυσική ανιχνευτικών διατάξεων ημιαγωγών με σκοπό να περιγραφεί η διαδικασία μέσω της οποίας δημιουργείται το σήμα που θέλουμε να ενισχύσουμε, να μετρήσουμε και να μετατρέψουμε σε ψηφιακό.

Το τρίτο κεφάλαιο περιέχει την περιγραφή του ολοκληρωμένου κυκλώματος που διενεργεί την ενίσχυση του σήματος και την μετατροπή του σε ψηφιακό.

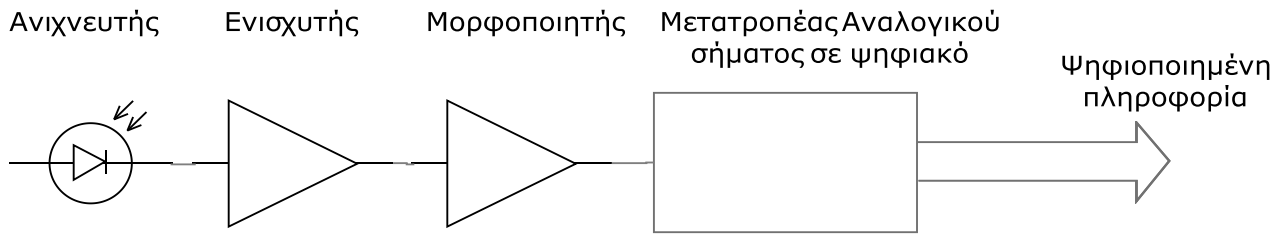
Στο τέταρτο κεφάλαιο ακολουθεί η περιγραφή του συστήματος και της διαδικασίας σχεδίασης του συστήματος ελέγχου και λήψη δεδομένων.

Τα κεφάλαια 5 και 6 περιέχουν συνοπτική περιγραφή των ενσωματωμένων κυκλωμάτων που χρησιμοποιήθηκαν, FPGA και USB αντίστοιχα καθώς και το γραφικό περιβάλλον που σχεδιάστηκε. Στο κεφάλαιο 7 σχολιάζεται ο κώδικας με τον οποίο προγραμματίστηκαν τα

ενσωματωμένα συστήματα. Περιγράφεται η διαδικασία των μετρήσεων και γίνεται σχολιασμός των αποτελεσμάτων.

2 Ημιαγωγοί Αισθητήρες ακτίνων χ

Όλα τα συστήματα ημιαγώγιμων ανιχνευτών περιλαμβάνουν τις ίδιες βασικές λειτουργίες, καθώς τα φωτόνια προσκρούουν στον ανιχνευτή γεννούν φορτίο το οποίο αποτελεί το σήμα που θέλουμε να επεξεργαστούμε. Οι συγκεκριμένοι ανιχνευτές είναι συνήθως πολυκάναλοι και είτε έχουν τοπολογία λωρίδας είτε κυψελίδας, όπου κάθε λωρίδα ή κυψελίδα αποτελεί ένα κανάλι. Το σήμα από κάθε κανάλι του ανιχνευτή πρέπει αρχικά να ενισχυθεί, να γίνει οποία ενδεχόμενη αναλογική επεξεργασία του και μετά να ψηφιοποιηθεί και να αποθηκευθεί για περεταίρω, ψηφιακή πλέον επεξεργασία, και ανάλυση. Τέτοιοι μετατροπείς αναλογικού σήματος σε ψηφιακό χρησιμοποιούνται για υβριδικά συστήματα και σχεδιάζονται και αυτοί σε διάταξη κυψελίδας, ώστε να ενωθούν με την διαδικασία του bump bonding με τον ανιχνευτή. Έτσι έχουμε την σύνδεση κάθε κυψελίδας του ανιχνευτή με έναν ξεχωριστό μετατροπέα του ολοκληρωμένου κυκλώματος, επιτυγχάνοντας ψηφιοποίηση ανά κυψελίδα, μείωση του θορύβου και καλύτερη απεικόνιση.



Εικόνα 2-1 Η ακτινοβολία απορροφάται από τον ανιχνευτή και μετατρέπεται σε ηλεκτρικό σήμα, το οποίο ολοκληρώνεται, μορφοποιείται και τελικά ψηφιοποιείται.

2.1 Σχηματισμός και λήψη του σήματος

Οι ημιαγώγιμοι ανιχνευτές αποτελούν θαλάμους ιονισμού στους οποίους γεννώνται οι φορείς οι οποίοι πρέπει στην συνέχεια να ενισχυθούν ώστε να αναγνωσθεί η χρήσιμη πληροφορία. Έχουμε λοιπόν δημιουργία του σήματος μέσω ιονισμού καθώς τα προσπίπτοντα φωτόνια προσδίδουν ενέργεια στα ατομικά ηλεκτρόνια για τον σχηματισμό ζευγών ηλεκτρονίων - οπών. Η ενέργεια ιονισμού είναι ανάλογη του ενεργειακού χάσματος και έτσι το ενεργειακό χάσμα είναι αυτό που καθορίζει το ελάχιστο όριο ανίχνευσης. Όταν η ενέργεια των

φωτονίων είναι μικρότερη από το ενεργειακό χάσμα, δεν έχουμε μεταφορά ενέργειας και η απορρόφηση είναι πολύ μικρή, ενώ ο συντελεστής απορρόφησης αυξάνεται απότομα μόνο όταν οι ενέργειες πλησιάζουν αυτή του ενεργειακού χάσματος. Το ισοδύναμο κύκλωμα που αντιπροσωπεύει τον ανιχνευτή είναι μια πηγή ρεύματος παράλληλα με έναν πυκνωτή, ο οποίος προκύπτει από την χωρητικότητα των ηλεκτροδίων του ανιχνευτή κυψελίδας. Η ολική χωρητικότητα του ανιχνευτή παίζει σημαντικό ρόλο στην σχεδίαση των ηλεκτρονικών ανάγνωσης όπως θα δούμε.

2.2 Δημιουργία του μετρούμενου σήματος

Οι ημιαγώγιμοι ανιχνευτές αποτελούν θαλάμους ιονισμού στους οποίους τα σωματίδια πρός ανίχνευση δημιουργούν ζεύγη ηλεκτρονίων – οπών. Εφαρμόζοντας υψηλό ηλεκτρικό πεδίο στον ανιχνευτή σε διάταξη pn ή διόδου Schottky με χαμηλό ρεύμα διαρροής, επιτυγχάνεται καλύτερη συλλογή φορτίου και οι φορείς που γεννώνται, κατά την κίνηση τους επάγουν φορτίο στα ηλεκτρόδια.

2.2.1 Ανιχνευτές σε διάταξη Διόδου

Σε μια δίοδο η χαρακτηριστική ρεύματος – τάσης δίνεται από την σχέση:

$$I = I_0(e^{\frac{eV}{kT}} - 1) \tag{1.1}$$

όπου για ισχυρή αρνητική πόλωση ο εκθετικός όρος είναι αμελητέος και το ρεύμα είναι ίσο με

$$I = -I_0 \tag{1.2}$$

Το ρεύμα αυτό επηρεάζεται από προσμίξεις και ατέλειες του ημιαγωγού και αυξάνεται καθώς μεγαλώνει το πλάτος της περιοχής απογύμνωσης. Έχουμε λοιπόν σχηματισμό ενός πυκνωτή, το διηλεκτρικό του οποίου είναι η περιοχή απογύμνωσης, και το εφαρμοζόμενο ηλεκτρικό πεδίο θα οδηγεί τους φορείς που θα σχηματίζονται λόγω ιονισμού στα ηλεκτρόδια.

2.2.2 Δίοδος PN

Θεωρούμε μια απότομη επαφή $p^+ - n$ όπου $N_A >> N_D$, όταν είναι σε θερμική ισορροπία

$$J_n = \mu_n n \frac{dE_F}{dx} = 0 \Rightarrow \frac{dE_F}{dx} = 0 \tag{1.3}$$

και

$$J_p = \mu_p n \frac{dE_F}{dx} = 0 \Rightarrow \frac{dE_F}{dx} = 0\,, \tag{1.4}$$

επομένως το επίπεδο Fermi πρέπει να είναι σταθερό σε ολόκληρη την διάταξη. Το εσωτερικό δυναμικό είναι:

$$\begin{aligned} qV_{bi} = E_g - (q\varphi_n + q\varphi_p) = qV_n + qV_p \Rightarrow V_{bi} = \frac{kT}{q}\left(\ln\left(\frac{n_{n0}}{n_i}\right) + \ln\left(\frac{p_{p0}}{p_i}\right)\right) \Rightarrow \\ \Rightarrow V_{bi} \simeq \frac{kT}{q}\ln\left(\frac{N_D N_A}{n_i}\right) \end{aligned} \tag{1.5}$$

Αντικαθιστώντας και την $n_{n0}p_{n0} = n_{p0}p_{p0} = n_i^2$, έχουμε:

$$V_{bi} = \frac{kT}{q}\ln\left(\frac{p_{p0}}{p_{n0}}\right) = \frac{kT}{q}\ln\left(\frac{n_{n0}}{n_{p0}}\right) \tag{1.6}$$

Επειδή το ηλεκτρικό πεδίο μακριά από την επαφή πρέπει να είναι μηδέν έπεται ότι το ολικό αρνητικό φορτίο στην p πλευρά, πρέπει να είναι ακριβώς ίσο με το θετικό φορτίο στην πλευρά n.

$$N_A W_{Depl}^p = N_D W_{Depl}^n \tag{1.7}$$

Αυτό που μας μένει πλέον είναι να λύσουμε την εξίσωση Poisson, έχουμε:

$$-\frac{d^2V_i}{dx^2} = \frac{dE}{dx} = \frac{\rho(x)}{\varepsilon_s} = \frac{q}{\varepsilon_s}\left[N_D^+ - n(x) - N_A^-(x) + p(x)\right] \tag{1.8}$$

Στην περιοχή απογύμνωσης ισχύει, $n(x) \simeq p(x) \simeq 0$

$$\frac{d^2V_i}{dx^2} \simeq \frac{qN_A}{\varepsilon_s} \qquad \text{για } -W_D^p \le x \le 0\,, \tag{1.9}$$

$$-\frac{d^2V_i}{dx^2} \simeq \frac{qN_D}{\varepsilon_s} \qquad \text{για } 0 \le x \le W_D^n\,, \tag{1.10}$$

Ολοκληρώνοντας τις παραπάνω εξισώσεις παίρνουμε το ηλεκτρικό πεδίο:

$$E(x) = -\frac{qN_A(x+W_D^p)}{\varepsilon_s} \qquad \text{για } -W_D^p \le x \le 0 \tag{1.11}$$

$$E(x) = -E_{\max} + \frac{qN_D x}{\varepsilon_s} = -\frac{qN_D}{\varepsilon_s}\left(W_D^n - x\right) \qquad \text{για } 0 \le x \le W_D^n \tag{1.12}$$

Όπου το $E_{\max}$ είναι το μέγιστο πεδίο, εμφανίζεται στην θέση $x = 0$ και δίνεται από την σχέση:

$$|E_{\max}| = \frac{qN_D W_D^n}{\varepsilon_s} = \frac{qN_A W_D^p}{\varepsilon_s} \tag{1.13}$$

Σε κάθε μια από τις δύο περιοχές έχουμε τα ακόλουθα δυναμικά:

$$V_p = \frac{qN_A (W_D^p)^2}{2\varepsilon_s}, \tag{1.14}$$

$$|V_n| = \frac{qN_D (W_D^n)^2}{2\varepsilon_s} \tag{1.15}$$

Τελικά

$$V_{bi} = V_p + |V_p| = V_i W_D^n = \frac{|E_{\max}|}{2}(W_D^p + W_D^n) \tag{1.16}$$

Τα πλάτη της περιοχής απογύμνωσης καθώς και το συνολικό είναι:

$$W_D^p = \sqrt{\frac{2\varepsilon_s V_{bi}}{q}\frac{N_D}{N_A(N_A+N_D)}} \tag{1.17}$$

$$W_D^n = \sqrt{\frac{2\varepsilon_s V_{bi}}{q}\frac{N_A}{N_D(N_A+N_D)}} \tag{1.18}$$

$$W_D^p + W_D^n = \sqrt{\frac{2\varepsilon_s}{q}\frac{(N_A+N_D)}{N_A N_D}V_{bi}} \tag{1.19}$$

Όταν θεωρήσουμε απότομη επαφή $p^+ - n$:

$$W_D = \sqrt{\frac{2\varepsilon_s V_{bi}}{qN}} \tag{1.20}$$

Όπου $N = N_D$ όταν $N_A >> N_D$, και

$$V_i(x) = |E_{\max}|\left(x - \frac{x^2}{2W_D}\right) \tag{1.21}$$

Στην περίπτωση που εφαρμόσουμε στην δίοδο τάση V αρκεί να αντικαταστήσουμε την V_{bi} με $V_{bi} - V_{applied}$, δηλαδή:

$$W_D = \sqrt{\frac{2\varepsilon_s}{qN}\left[(V_{bi} - V_{applied}) - \frac{2kT}{q}\right]} \tag{1.22}$$

2.2.3 Επαφή μετάλλου – ημιαγωγού, ιδανική δίοδος Schottky

Καθώς το μέταλλο και ο ημιαγωγός έρχονται σε επαφή, τα επίπεδα Fermi στα δύο υλικά ευθυγραμμίζονται, αυτό επιτυγχάνεται με την μείωση του επιπέδου του ημιαγωγού κατά ποσό ίσο με την διαφορά των έργων εξόδου των δύο υλικών. Το έργο εξόδου είναι η διαφορά ανάμεσα στο επίπεδο του κενού και στο επίπεδο Fermi για το κάθε υλικό, και για το μέταλλο είναι $q\varphi_m$, ενώ για τον ημιαγωγό $q(\chi + \varphi_n)$, όπου $q\chi$ η συνάφεια των ηλεκτρονίων, μετρούμενη από την ζώνη αγωγιμότητας E_C μέχρι το επίπεδο κενού και $q\varphi_n$ η διαφορά ενέργειας ανάμεσα στην E_C και το επίπεδο Fermi. Το φράγμα που σχηματίζεται είναι υπεύθυνο για την δυνατότητα ελέγχου της αγωγιμότητας, αλλά και για την χωρητική συμπεριφορά μιας διόδου Schottky.

Δυναμικό επαφής της διόδου είναι η διαφορά δυναμικού $\varphi_m - (\chi - \varphi_n)$, στην κατάσταση όπου το μέταλλο είναι ηλεκτρικά συνδεδεμένο με τον ημιαγωγό, αλλά είναι σε απόσταση d μεταξύ τους. Καθώς το d μειώνεται υπάρχει αύξηση του ηλεκτρικού πεδίου στο διάκενο και συσσώρευση αρνητικού φορτίου στην επιφάνεια του μετάλλου. Όταν το d μηδενιστεί και ενωθούν μέταλλο και ημιαγωγός, έχει σχηματιστεί φράγμα ύψους:

$$q\varphi_{Bn0} = q(\varphi_m - \chi) \tag{1.23}$$

Το ύψος δηλαδή είναι ίσο με την διαφορά στο έργο εξόδου του μετάλλου και στην συνάφεια των ηλεκτρονίων του ημιαγωγού. Για ημιαγωγό τύπου p το φράγμα είναι:

$$q\varphi_{Bp0} = E_g - q(\varphi_m - \chi) \tag{1.24}$$

Η περιοχή απογύμνωσης για δίοδο Schottky είναι παρόμοια με αυτή της απότομης επαφής p-n και δίνεται από την σχέση:

$$W_D = \sqrt{\frac{2\varepsilon_s}{qN_D}\left[(V_{bi} - V_{applied}) - \frac{kT}{q}\right]} \tag{1.25}$$

Για το ηλεκτρικό πεδίο έχουμε,

$$|E(x)| = \frac{qN_D}{\varepsilon_s}(W_D - x) = E_{\max} - \frac{qN_D}{\varepsilon_s}x \tag{1.26}$$

Το φορτίο χώρου και η χωρητικότητα της περιοχής απογύμνωσης είναι:

$$Q_{sc} = qN_D W_D = \sqrt{2q\varepsilon_s N_D (V_{bi} - V_{applied} - \frac{kT}{q})} \tag{1.27}$$

$$C_D = \frac{\varepsilon_s}{W_D} = \sqrt{\frac{q\varepsilon_s N_D}{2\left[V_{bi} - V - (\frac{kT}{q})\right]}} \tag{1.28}$$

Βλέπουμε λοιπόν ότι το πλάτος της περιοχής απογύμνωσης είναι ανάλογο της τετραγωνικής ρίζας της εφαρμοζόμενης ανάστροφης πόλωσης. Παρατηρούμε ότι αυξάνοντας την ανάστροφη πόλωση αυξάνεται ο όγκος του ημιαγωγού που είναι ευαίσθητος στην διέλευση ακτινοβολίας και επιπλέον μειώνεται η χωρητικότητα απογύμνωσης κάτι το οποίο αυξάνει το φορτίο που σχηματίζεται κατά την διέλευση ενός φορτισμένου σωματιδίου, άρα και το σήμα που θέλουμε να μετρήσουμε, ενώ επίσης μειώνεται και ο θόρυβος. Υπάρχει βέβαια και κάποιο όριο στην μέγιστη δυνατή εφαρμοζόμενη τάση πόλωσης, καθώς μετά από κάποια τιμή του πεδίου μέσα στον ημιαγωγό έχουμε φαινόμενα χιονοστιβάδας και επέρχεται κατάρρευση. Ένας άλλος παράγοντας που οδηγεί στην διεύρυνση της περιοχής απογύμνωσης είναι η μείωση της πυκνότητας προσμίξεων. Αξίζει να σημειωθεί ότι σε ημιαγωγούς μικρής πυκνότητας προσμίξεων πρέπει να ληφθεί υπόψη η διαφορά των δύο τύπων προσμίξεων $N_A - N_D$.

2.2.3.1 Μελέτη διόδων Schottky CdTe

Το CdTe τις τελευταίες δεκαετίες αποτελεί ένα πολύ σημαντικό ημιαγώγιμο υλικό για χρήση σε ανιχνευτές ακτίνων χ και γάμμα, με εφαρμογές σε πολλούς τομείς όπως την ιατρική και την διαστημική. Εξαιτίας των μεγάλων ατομικών αριθμών του Cd(48) και του Te(52) είναι δυνατό να ανιχνεύσουν φορτισμένα σωματίδια με ενέργειες μέχρι 1MeV, μεγαλύτερες δηλαδή από αυτές που μπορεί να ανιχνεύσει το πυρίτιο. Επιπλέον καθώς έχει μεγαλύτερο ενεργειακό χάσμα $E_g^{CdTe} = 1.46eV@300K$ και επιτρέπει την λειτουργία του ως ανιχνευτή σε θερμοκρασία δωματίου. Για την επίτευξη υψηλής συλλογής φορτίου με μικρό ρεύμα διαρροής το CdTe χρησιμοποιείται σε διάταξη Schottky, έχουμε δηλαδή την δημιουργία ενός φράγματος στην μια πλευρά του κρυστάλλου, ενώ ωμική επαφή στην άλλη. Αρχικά η χρήση του ήταν σε διάταξη με ωμικές επαφές και στις δύο πλευρές αλλά καθώς οι συγκεκριμένοι ανιχνευτές εξελίσσονταν κατασκευάστηκαν δίοδοι Schottky με p-τύπου αλλά

και με n-τύπου ημιαγωγό με χαρακτηριστική διαφορά ότι οι n-τύπου διατάξεις έχουν μικρότερη αντίσταση και επιτυγχάνουν πλήρη συλλογή φορτίου με μικρότερη τάση πόλωσης.

Τυπικές τιμές αντίστασης για n-τύπου διόδους Schottky – CdTe είναι της τάξης των $10^2 - 10^3 \Omega cm$, ενώ η συγκέντρωση ηλεκτρονίων είναι $10^{13} - 10^{14} cm^{-3}$ και η ευκινησία τους $(1-1.5)10^3 cm^2 \cdot (Vs)^{-1}$. Η επαφή Schottky μπορεί να σχηματιστεί με χρήση Ni ενώ η ωμική επαφή με In. Η χωρητικότητα της διόδου δίνεται από την σχέση:

$$C(V) = A\sqrt{\frac{\varepsilon_{CdTe}\varepsilon_0 q^2 (N_D - N_A)}{2(\varphi_0 - qV)}} \tag{1.29}$$

Α η επιφάνεια της διόδου, ε_{CdTe} η σχετική διηλεκτρική σταθερά του υλικού και ε_0 αυτή του κενού. $N_D - N_A$ η συγκέντρωση των μη αντισταθμισμένων δοτών και $\varphi_0 = qV_{bi}$ το ύψος του φράγματος.

Ημιαγωγός	Πυκνότητα (g/cm^3)	Ενεργειακό Χάσμα (eV)	Ενδογενής Πυκνότητα (cm^{-3})	Μέσο Z	E_{e-h}^{pair} (eV)	Ευκινησία Φορέων $(cm^2/(Vs))$ $\underbrace{\quad e \quad}$	$\underbrace{\quad h \quad}$	Χρόνος Ζωής
Si	2.3	1.12	1.45E10	14	3.61	1,415	480	250μs
Ge	5.3	0.66	2.4E13	32	2.96	3,900	1,800	250μs
GaAs	5.4	1.42	1.8E6	32	4.35	8,800	320	1-10ns
CdTe	6.1	1.44	1E7	50	4.43	1,050	100	0.1-2μs
CdZnTe	5.8	~1.6	1E7	49.1	4.6	~1,000	50-80	~μs

Η παραπάνω εξίσωση είναι πολύ χρήσιμη γιατί μετρώντας την χωρητικότητα της διόδου μπορεί να υπολογιστούν το ύψος του φράγματος και η πυκνότητα των μη αντισταθμισμένων δοτών $N_D - N_A$. Τιμές που αναφέρονται στην βιβλιογραφία είναι $\varphi_0 = 1.05eV$ και $N_D - N_A = 1\times10^{14} cm^{-3}$. Το πλάτος της περιοχής απογύμνωσης ως συνάρτηση της τάσης δίνεται από την σχέση:

$$W(V) = \sqrt{\frac{2\varepsilon\varepsilon_0(\varphi_0 - qV)}{q^2(N_D - N_A)}} \tag{1.30}$$

2.3 Συλλογή Φορτίου

Η μεταφορά του φορτίου μπορεί να γίνει με δύο τρόπους, είτε μέσω διάχυσης, είτε μέσω ολίσθησης. Η θερμική ενέργεια κάνει τους φορείς να κινούνται σε τυχαίες διευθύνσεις και παρουσία μιας βαθμίδας συγκέντρωσης, αυτοί συγκρούονται με μεγαλύτερη συχνότητα στην κατεύθυνση της μεγαλύτερης συγκέντρωσης και έτσι η συνισταμένη κίνηση γίνεται στην αντίθετη κατεύθυνση.

Παρουσία ηλεκτρικού πεδίου οι φορείς, κινούνται παράλληλα στο πεδίο και αλληλεπιδρούν με το κρυσταλλικό πλέγμα, προκαλώντας ταλαντώσεις του πλέγματος (φωνόνια). Οι χαρακτηριστικοί χρόνοι για την διέγερση φωνονίων είναι κατά πολύ μικρότεροι από τους χρόνους μεταφοράς, επομένως οι φορείς αποκτούν ταχύτητα που είναι συνάρτηση αποκλειστικά του ηλεκτρικού πεδίου.

$$\vec{v} = \mu \vec{E} \tag{1.31}$$

Όπου μ είναι η ευκινησία των φορέων και συνδέεται με την σταθερά διάχυσης μέσω της σχέσης Einstein,

$$\mu = \frac{q_e}{kT} D \tag{1.32}$$

Καθώς η ακτινοβολία απορροφάται στην ενεργό περιοχή του ανιχνευτή, σχηματίζονται εκεί ελεύθερα ηλεκτρόνια και οπές, τα οποία κινούνται υπό την επίδραση του ηλεκτρικού πεδίου. Παρά το γεγονός ότι κινούνται σε αντίθετες διευθύνσεις, καθώς έχουν αντίθετο φορτίο η συνεισφορά τους στο ρεύμα έχει την ίδια φορά. Το χρονικό διάστημα που απαιτείται για έναν φορέα να διασχίσει την ενεργό περιοχή του ανιχνευτή ονομάζεται χρόνος συλλογής του φορτίου. Για το ηλεκτρικό πεδίο έχουμε για δίοδο Schottky:

$$E = \frac{qN_a}{\varepsilon_s}\left(w_D - x\right) = E_{\max} - \frac{qN_d}{\varepsilon_s} x \tag{1.33}$$

Όπου

$$w_D = \sqrt{\frac{2\varepsilon_s}{qN_D}}\left[\left(V_{bi} - V_{applied}\right) - \frac{kT}{q}\right] \tag{1.34}$$

και η περιοχή απογύμνωσης εκτείνεται σε ολόκληρο τον όγκο του ανιχνευτή για $W_D = d$
Το οποίο συμβαίνει για εφαρμοζόμενη εξωτερική τάση:

$$d = \sqrt{\frac{2\varepsilon_s}{qN_D}\left[\left(V_{bi} - V_{applied}\right) - \frac{kT}{q}\right]} \tag{1.35}$$

$$d^2 = \frac{2\varepsilon_s}{qN_D}V_{bi} - \frac{2\varepsilon_s}{q\mathrm{N}_D}V_{applied} - \frac{2\varepsilon_s}{qN_D}\frac{kT}{q} \tag{1.36}$$

$$V_{applied} = \left(V_{bi} - \frac{KT}{q}\right) - \frac{qN_D}{2\varepsilon_s}d^2 \tag{1.37}$$

Όπου το $V_{applied}$ είναι θετικό για ορθή πόλωση και αρνητικό για αναστροφή.
Για το ηλεκτρικό πεδίο έχουμε:

$$E(x) = -\frac{qN_D}{e_s}\left(W_D - x\right) \overset{W_d = d}{=} -\frac{qN_D}{\varepsilon_s}\left(d - x\right) \tag{1.38}$$

Όπου το ηλεκτρικό πεδίο μειώνεται γραμμικά από μια μέγιστη τιμή $\frac{qN_d}{\varepsilon_s}d$ στο $x = 0$ μέχρι την τιμή μηδέν στην άλλη μεριά του ανιχνευτή όπου X = d.
Ας υποθέσουμε ότι ο ανιχνευτής είναι μερικώς απογυμνωμένος όπου $V_{applied} < V_{depletion}^{full}$

$$\mathrm{E}(\mathrm{X}) = \frac{qN_D}{\varepsilon_s}\left(W - x\right) = E_0\left(W - x\right) \tag{1.39}$$

και η ταχύτητα του φορέα είναι:

$$U(x) = \mu E(x) = \mu E_0\left(W - x\right) \tag{1.40}$$

Για το Si στους 300K η ευκινησία για ασθενή ηλεκτρικά πεδία είναι για τα ηλεκτρόνια $1350cm^2 \backslash Vs$ και $480cm^2/Vs$ για τις οπές. Ενώ για το CdTe είναι $1050cm^2/Vs$ για τα ηλεκτρόνια και $100cm^2/Vs$ για τις οπές. Για υψηλά ηλεκτρικά πεδία η ευκινησία γίνεται αντιστρόφως ανάλογη του ηλεκτρικού πεδίου και οι φορείς λαμβάνουν μια σταθερή τιμή ταχύτητας, αυξανομένου του ηλεκτρικού πεδίου.
Ο χρόνος που χρειάζεται για ένα φορτίο το οποίο βρίσκεται στην θέση Xo για να φθάσει στο σημείο x είναι

$$t(x) = \int_{x_0}^{x}\frac{1}{U_{(x)}}dx = \frac{1}{\mu E_0}\int_{x_0}^{x}\frac{1}{W - x}dx = -\frac{1}{\mu E_0}\left[\log\left(W - x\right)\right]_{x_0}^{x} \tag{1.41}$$

$$t(x) = \frac{-1}{\mu E_0}\log\frac{W - x}{W - x_0} = -\frac{1}{\mu\frac{qN_D}{\varepsilon_s}}\log\frac{W - x}{W - x_0} = \frac{\varepsilon_s}{\mu qN_D}\log\frac{W - x}{W - x_0} \tag{1.42}$$

Για μία οπή η οποία ολισθαίνει και συλλέγεται στο $x = 0$ με ευκινησία μ_p, η παραπάνω εξίσωση μας δίνει:

$$t(x_0) = \frac{\varepsilon_s}{\mu_p q N_D} \log \frac{W - x}{W - x_0} = \frac{\varepsilon_s}{\mu_p q N_D} \log \frac{W}{W - x_0} \tag{1.43}$$

και ορίζοντας την ποσότητα $\frac{\varepsilon_s}{\mu_p q N_D} = \tau_p$ ως τον χαρακτηριστικό χρόνο συλλογής, έχουμε:

$$t(x_0) = \tau_p \log \frac{W}{W - x_0} \tag{1.44}$$

Για ηλεκτρόνια η ίδια εξίσωση δεν δίνει λύση για συλλογή τους στο x=W, όμως με αναδιάταξη της, έχουμε:

$$x(t) = W - (W - x_0) e^{-\frac{t}{\tau_n}} \tag{1.45}$$

όπου το τ_n ορίζεται ανάλογα με το τ_p.

Για ένα φορτίο που βρίσκεται στην θέση $x_0 = 0$ και ολισθαίνει προς την , έχουμε:

$$x(t) = W(1 - e^{-\frac{t}{\tau_n}} \tag{1.46}$$

Ο χρόνος συλλογής μπορεί να μειωθεί αν η τάση πόλωσης του ανιχνευτή είναι μεγαλύτερη της τάσης πλήρους απογύμνωσης. Τότε στο ηλεκτρικό πεδίο που έχουμε υπολογίσει: $E(x) = E_0(d - x)$ προστίθεται ένας σταθερός όρος E_{OB}.

Έχουμε:

$$E(x) = E_0(d - x) + E_{OB} \tag{1.47}$$

και τοτε

$$U(x) = \mu . \mathrm{E}(x) = \mu \left[\mathrm{E}_0 (d - x) + \mathrm{E}_{\mathrm{OB}} \right] \tag{1.48}$$

$$t(x) = \int_{x_0}^{x} \frac{1}{U(x)} dx = \frac{1}{\mu} \int_{x_0}^{x} \frac{1}{E_0 (d - x) + E_{OB}} dx \overset{E_0 = \frac{qN_D}{\varepsilon_s}}{=} -\frac{d}{\mu \mathrm{E}_0} \left[\log \left(E_0 + E_1 - E_0 \frac{x}{d} \right) \right]_{x_0}^{x} \tag{1.49}$$

$$t_{(x)} = \frac{d}{\mu \mathrm{E}_0} \log \frac{E_0 + E_1 - E_0 \frac{x}{d}}{E_0 + E_1 - E_0 \frac{x_0}{d}} \tag{1.50}$$

Στην περίπτωση των οπών που ξεκινούν στο $x_0 = w = d$ και ολισθαίνουν προς το x=0

$$t_{collection.h} = \frac{d}{\mu_\rho E_0} \log\left(1 + \frac{E_0}{E_1}\right) \tag{1.51}$$

ενώ για ηλεκτρόνια που βρίσκονται στο $x_0 = 0$ και ολισθαίνουν προς το x=w=d

$$t_{collection,e} = \frac{d}{\mu_n E_0} \log \frac{E_1}{E_0 + E_1} \tag{1.52}$$

Για $E_1 >> E_0$, $\log\left(1 + \frac{E_0}{E_1}\right) \approx \frac{E_0}{E_1}$

Όπου:

$$\mathrm{E}_0 = -\frac{qN_d}{\varepsilon_s} = \left[\left(V_{b_i} - \frac{kT}{q}\right) - V_{applied}\right]\frac{2}{d^2} \tag{1.53}$$

$$E_1 = \frac{V_{applied} - V_{depletion} - V_{bi}}{d} \tag{1.54}$$

2.4 Λήψη και ενίσχυση του σήματος

Το σήμα που προέρχεται από τον ανιχνευτή αναπαρίσταται με μία πηγή ρεύματος η οποία εξ' ορισμού έχει άπειρη αντίσταση . Η πραγματική πεπερασμένη αντίσταση μιας πηγής παριστάνεται με την βοήθεια της αντίστασης παράλληλα με την πηγή ρεύματος. Το ρεύμα του υπο-μέτρηση σήματος χωρίζεται στο ρεύμα i_R που διαρρέει την αντίσταση και i_i που είναι το ρεύμα που εισέρχεται στον ενισχυτή, έχουμε $i_i = \frac{R_s}{R_s + R_i} i_s$.

Για να ισχύει $i_i \approx i_s$ πρέπει $R_i \ll R_s$. και για να λειτουργήσει ως ενισχυτής ρεύματος πρέπει η εμπέδηση εισόδου του ενισχυτή να είναι πολύ μικρότερη της εμπέδησης της πηγής. Μία πηγή ρεύματος μπορεί να μοντελοποιηθεί και από μία πηγή τάσης σε σειρά με μία αντίσταση. Για να μην επηρεάζεται το σήμα από την αντίσταση εισόδου του ανιχνευτή, θα πρέπει η αντίσταση εισόδου να είναι μικρή συγκριτικά με την αντίσταση της πηγής. Στην έξοδο πρέπει η αντίσταση εξόδου να είναι μεγάλη συγκριτικά με την αντίσταση εισόδου ενός επόμενου σταδίου . Έτσι για βέλτιστη μεταφορά του σήματος πρέπει η αντίσταση της πηγής

να είναι μεγάλη σε σύγκριση με αυτήν του φορτίου. Οι ενισχυτές μπορούν να, είτε να δέχονται στην είσοδο τους ρεύμα και να το μετατρέπουν σε τάση, είτε να δέχονται τάση και στην είσοδο και να την μετατρέπουν σε ρεύμα στην έξοδο. Το κέρδος εκφράζεται ως $V/A\rightarrow$ διεμπέδηση είτε ως $A/V\rightarrow$διαγωγιμότητα. Εφόσον ο τρόπος λειτουργίας εξαρτάται από τον λόγο της αντίστασης της πηγής προς την αντίσταση εισόδου, ένας ενισχυτής τάσης μπορεί να λειτουργήσει είτε σε λειτουργία τάσης είτε ρεύματος ανάλογα με την αντίσταση της πηγής. Σε λειτουργία ρεύματος η τάση στην είσοδο του ενισχυτή είναι $U_i = i_i R_i$ και η τάση εξόδου: $U_0 = A_U U_i$ όπου A_U είναι το κέρδος τάσης.

2.4.1 Λειτουργία τάσης και ρεύματος με χωρητικές πηγές

Το σήμα που προέρχεται από τον ανιχνευτή είναι ένας παλμός ρεύματος μεγέθους i_s και διάρκειας $t_{c,}$, έτσι το φορτίο θα είναι : $Q_s = \int i_s(t)dt = i_s t_c$

Το κέρδος τάσης του ενισχυτή είναι A_u και έτσι η τάση στην έξοδο θα είναι: $U_0 = A_u U_s$
Ο τρόπος λειτουργίας του ενισχυτή εξαρτάται από τον χρόνο συλλογής φορτίου t_c του ανιχνευτή και την σταθερά χρόνου στην είσοδο $R_i C_d$

1. Αν $R_i C_d \ll t_c$ η χωρητικότητα του ανιχνευτή αποφορτίζεται γρήγορα και η τάση εξόδου είναι ανάλογη του στιγμιαίου ρεύματος $U_o = f\left(i_s(t)\right)$ και το σύστημα αισθητήρα – ενισχυτή είναι σε λειτουργία ρεύματος.

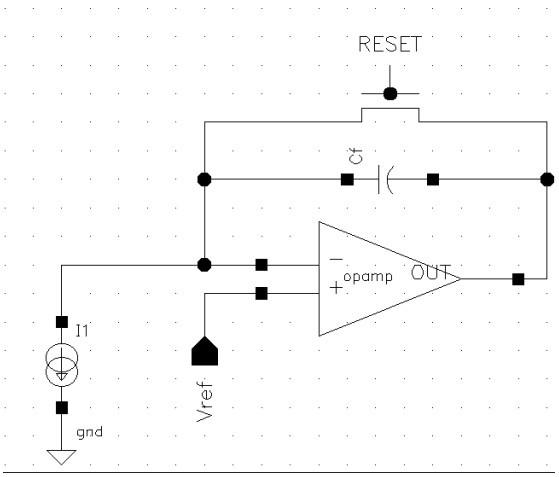
2. Αν $R_i C_d \gg t_c$, η χωρητικότητα του ανιχνευτή αποφορτίζεται αργά και έτσι το επαγόμενο ρεύμα αρχικά ολοκληρώνεται στην χωρητικότητα του αισθητήρα πριν αποφορτισθεί μέσω της αντίστασης εισόδου. Η τάση εισόδου είναι:

$$U_0 = V_0 \exp\left[-t / R_i C_d\right] \qquad (1.55)$$

Όπου $V_0 = \frac{Q_s}{C_d} = f\left(\int i_s(t)\right)dt$ Εδώ το σήμα εμφανίζεται ως τάση και το σύστημα βρίσκεται σε λειτουργία τάσης. Και στις δύο περιπτώσεις ο ενισχυτής παρέχει κέρδος τάσης και το σήμα στην έξοδο του είναι ανάλογο της τάσης στην είσοδο.

2.4.2 Ενισχυτές με ανατροφοδότηση – Ο ενισχυτής φορτίου

Η ανατροφοδότηση μπορεί να χρησιμοποιηθεί στους ενισχυτές για τον έλεγχο του κέρδους και της αντίστασης εισόδου τους. Μια πολύ χρήσιμη διάταξη για ανάγνωση από ανιχνευτές είναι ο ενισχυτής φορτίου που φαίνεται στην παρακάτω εικόνα.



Εικόνα 2-2

Το βασικό δομικό στοιχείο είναι ένας αναστρέφων ενισχυτής τάσης με μεγάλη αντίσταση εισόδου. Αρχικά θα υποθέσουμε ότι διαθέτει άπειρη αντίσταση εισόδου έτσι ώστε να μην έχουμε ροή ρεύματος εντός του ανιχνευτή. Εφόσον ο ενισχυτής αναστρέφει, το κέρδος τάσης του είναι:

$$\frac{dU_0}{dU_i} = -A \tag{1.56}$$

Δηλαδή, $U_0 = -AU_i$

Επίσης συνδέεται ένας πυκνωτής ανάδρασης C_f ανάμεσα στην είσοδο και την έξοδο. Αν ένα σήμα εισόδου παράγει μια τάση U_i στην είσοδο του ενισχυτή , τότε η τάση στην έξοδο του ενισχυτή είναι: $-AU_i$. Επομένως η πτώση τάσης στα άκρα του πυκνωτή ανάδρασης είναι:

$$U_f = (A+1)U_i \tag{1.57}$$

Και το φορτίο στον πυκνωτή C_f είναι :

$$Q_f = C_f U_f = C_f (A+1)U_i \tag{1.58}$$

Εφόσον το ρεύμα δεν μπορεί να εισέλθει στον ενισχυτή, το σήμα δεν μπορεί παρά να φορτίσει τον πυκνωτή ανάδρασης ώστε: $Q_f = Q_i$

Η είσοδος του ενισχυτή αποτελεί μια δυναμική χωρητικότητα εισόδου

$$C_1 = \frac{Q_i}{U_i} = C_f\left(A+1\right) \tag{1.59}$$

Η τάση εξόδου ανά μονάδα φορτίου στην είσοδο είναι:

$$A_Q = \frac{U_0}{Q_i} = \frac{AU_i}{C_i U_i} = \frac{A}{C_i} = \frac{A}{A+1}\frac{1}{C_f} \approx \frac{1}{C_f} \tag{1.60}$$

Όπου το κέρδος προσδιορίζεται από τον πυκνωτή ανατροφοδότησης.

Το φορτίο Q_s διαμοιράζεται ανάμεσα στην χωρητικότητα του αισθητήρα C_d και στην δυναμική χωρητικότητα εισόδου C_i.

Ο λόγος του μετρούμενου φορτίου προς το φορτίο του σήματος είναι:

$$\frac{Q_i}{Q_s} = \frac{Q_i}{Q_d + Q_s^{amplifiet}} = \frac{C_i}{C_d + C_i} = \frac{1}{1+\frac{C_d}{C_i}}$$

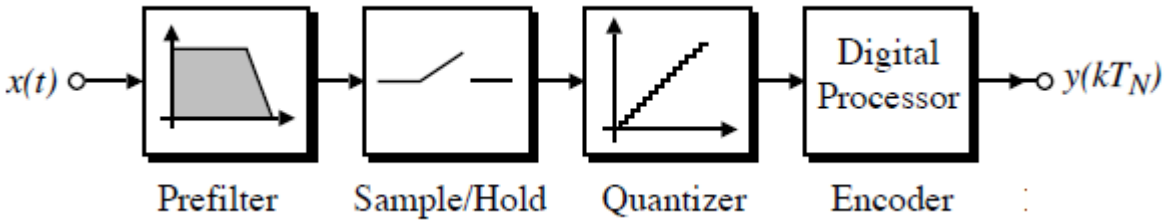
Ο οποίος τείνει στην μονάδα για μεγάλες τιμές της χωρητικότητας εισόδου σε σχέση με την χωρητικότητα του αισθητήρα. Οι ενισχυτές ολοκλήρωσης φορτίου είναι επίσης χρήσιμοι καθώς είναι εύκολη η βαθμονόμηση τους. Με χρήση ενός πυκνωτή και εφαρμόζοντας έναν παλμό ρεύματος συγκεκριμένης διάρκειας μπορούμε να εισάγουμε στον ενισχυτή ένα ρεύμα συγκεκριμένης τιμής. Εάν η δυναμική χωρητικότητα εισόδου C_1 είναι πολύ μεγαλύτερη από την χωρητικότητα Ctest τότε ο παλμός ΔV στην είσοδο του Ct θα εφαρμοστεί εξ ολοκλήρου στην Ct και θα εισάγει ένα φορτίο Ct ΔV.

Το φορτίο αυτό είναι:

$$Q_t = \frac{C_t}{1+\frac{C_t}{C_i + C_d}}.\Delta V \approx C_t\left(1-\frac{C_t}{C_i + C_d}\right)\Delta V \tag{1.61}$$

Έτσι λοιπόν για μεγαλύτερη ακρίβεια το σύστημα πρέπει να βαθμονομηθεί με τον ανιχνευτή συνδεδεμένο.

3 Περιγραφή Ολοκληρωμένου



Εικόνα 3-1

Οι αισθητήρες ακτινοβολίας βρίσκουν εφαρμογές σε πεδία όπως στην ασφάλεια, στην βιομηχανία, στην ιατρική και σε διάφορους άλλους επιστημονικούς κλάδους, εξαιτίας της ικανότητας τους να μετατρέπουν μια οπτική εικόνα σε ηλεκτρικό σήμα. Η ανίχνευση των ακτίνων χ, είναι ιδιαίτερα χρήσιμη λόγω της διεισδυτικής της ικανότητας.

3.1 Θόρυβος Αισθητήρων

Οι πηγές θορύβου των ανιχνευτών ακτινοβολίας περιορίζουν την απόδοση τους. Ο θόρυβος των ηλεκτρονικών ανάγνωσης μετριέται στην έξοδο του συστήματος ανάγνωσης και θα πρέπει να είναι μικρότερος από ένα συγκεκριμένο και δεδομένο όριο. Το όριο αυτό προκύπτει ανάλογα με την εφαρμογή ώστε να μπορεί να διαχωριστεί με ακρίβεια το χρήσιμο σήμα από το σήμα θορύβου. Αν το επίπεδο θορύβου είναι αρκετά υψηλό υπάρχει αυξημένη πιθανότητα κάποια διέγερση που θα παράγει ασθενές σήμα να μην καταγραφεί. Σε πολλές περιπτώσεις όπου τα σήματα που προέρχονται από τον ανιχνευτή είναι πολύ ασθενή όπως συμβαίνει στις περισσότερες ιατρικές εφαρμογές η προδιαγραφή θορύβου θεωρείται η πλέον αυστηρή και κρίσιμη και θα πρέπει να ικανοποιηθεί οπωσδήποτε. Ο θόρυβος μετριέται με ένα μέγεθος που ονομάζεται Ισοδύναμο Φορτίο Θορύβου (Equivalent Noise Charge - ENC) με μονάδα το ισοδύναμο αριθμό ηλεκτρονίων (e^-). Έτσι για ιατρικές εφαρμογές το επίπεδο θορύβου πρέπει να είναι μερικές δεκάδες ηλεκτρόνια ενώ αντίθετα στα πειράματα Φυσικής Υψηλών Ενεργειών μπορεί να μέχρι ορισμένες χιλιάδες (50-5000 e^-).

3.1.1 Χρονικός Θόρυβος

Ο χρονικός θόρυβος είναι χρονικά τυχαίος και όχι σταθερός από frame σε frame. Περιγράφεται με στατιστικές κατανομές και μπορεί να μειωθεί υπολογίζοντας τον μέσο όρο

διαδοχικών frame. Υπάρχουν διάφορες πηγές χρονικού θορύβου στους αισθητήρες ακτινοβολίας. Πολύ σημαντικός είναι ο θόρυβος βολής (shot noise) και ο θερμικός θόρυβος.

3.1.2 Θόρυβος Βολής

Ο θόρυβος βολής περιγράφει την στατιστική αβεβαιότητα στην γένεση των φορέων κατά την πρόσπτωση των φωτονίων στον ημιαγωγό. Το μέτρο του είναι ίσο με την τετραγωνική ρίζα του μέσου αριθμού ηλεκτρονίων που γεννούνται στον ανιχνευτή, σε μονάδες αριθμού ηλεκτρονίων).

$$N_{SN} = \sqrt{N_{pelectrons}}$$

Όπου Npelectrons είναι ο αριθμός των ηλεκτρονίων που γεννούνται σον ημιαγωγό.

3.1.3 Θερμικός Θόρυβος kTC (Reset Noise)

Μετά την εκκαθάριση έχουμε φόρτιση του πυκνωτή ολοκλήρωσης στην ανάδραση του ενισχυτή. Αυτή η διαδικασία της εκκαθάρισης γεννά θόρυβο δειγματοληψίας ίσο με:

$$V_{kTC} = \sqrt{\frac{kT}{C}}$$

Το μέτρο του θορύβου αυτού είναι μεγάλο σε σύγκριση με άλλες χρονικές πηγές θορύβου, όπως για παράδειγμα ο 1/f θόρυβος.

3.1.4 Fixed Pattern Noise

Είναι ένα είδος χωρικού θορύβου που δείχνει την διακύμανση της εξόδου από pixel σε pixel στον αισθητήρα και τα ηλεκτρονικά ανάγνωσης. Δεν μεταβάλλεται από πλαίσιο σε πλαίσιο και δημιουργείται από το κακό ταίριασμα ανάμεσα στον αισθητήρα και στα ηλεκτρονικά ανάγνωσης.

3.2 Προδιαγραφές Συστημάτων Απεικόνισης

3.2.1 Χωρητικότητα ανιχνευτή

Ένα πολύ σημαντικό στοιχείο για τον σχεδιασμό των ηλεκτρονικών ανάγνωσης και ιδιαίτερα της πρώτης αναλογικής βαθμίδας είναι η χωρητικότητα του ανιχνευτή. Πιο συγκεκριμένα για βελτιστοποίηση της συμπεριφοράς των κυκλωμάτων ανάγνωσης θα πρέπει να ταιριαστεί η χωρητικότητα του στοιχείου εισόδου με την χωρητικότητα του ανιχνευτή. Αυτή η συνθήκη έχει σαν αποτέλεσμα την ελαχιστοποίηση του συνολικού θορύβου.

3.2.2 Δυναμική Περιοχή

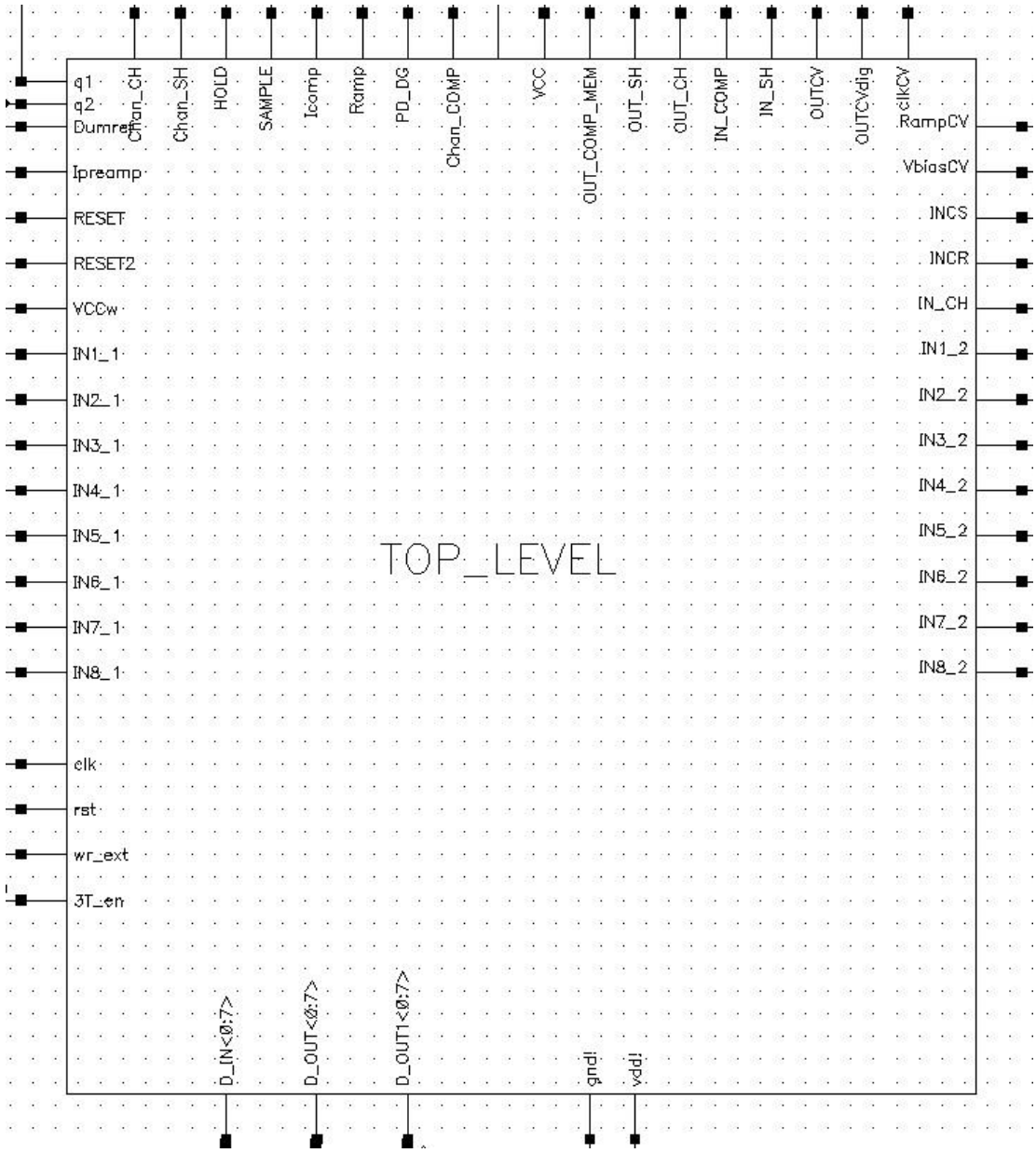
Η δυναμική περιοχή των ηλεκτρονικών ανάγνωσης εκφράζει το εύρος τιμών του φορτίου εισόδου που αναμένεται να παράγει ο ανιχνευτής στερεάς κατάστασης, το οποίο είναι ανάλογο με την ενέργεια της προσπίπτουσας ακτινοβολίας. Σε ορισμένες εφαρμογές η τιμή του φορτίου είναι δεδομένη και σταθερή οπότε ουσιαστικά η δυναμική περιοχή είναι μηδενική, ενώ αντίθετα σε άλλες περιπτώσεις είναι ιδιαίτερα εκτεταμένη. Το φορτίο εισόδου επηρεάζει με άμεσο τρόπο και την προδιαγραφή θορύβου των ηλεκτρονικών ανάγνωσης. Έτσι για εφαρμογές που το φορτίο εισόδου είναι ασθενές θα πρέπει αντίστοιχα χαμηλά να είναι και το επίπεδο θορύβου. Ορίζεται και ως ο λόγος του μεγαλύτερου προς το μικρότερο σήμα που μπορεί να μετρηθεί.

3.2.3 Χρόνος κορύφωσης – χρόνος επαναφοράς

Ο χρόνος κορύφωσης ορίζεται το χρονικό εκείνο σημείο μετά την διέγερση του ανιχνευτή στο οποίο το αναλογικό σήμα εξόδου των ηλεκτρονικών ανάγνωσης φτάνει στην μέγιστη τιμή του. Επιπλέον ο χρόνος επαναφοράς είναι το χρονικό εκείνο διάστημα που χρειάζεται το σήμα ώστε να επανέλθει μετά από μια διέγερση στην στάθμη ηρεμίας. Αφού επανέρθει το σήμα το κανάλι είναι έτοιμο να δεχτεί το επόμενο γεγονός. Τόσο ο χρόνος κορύφωσης όσο και ο χρόνος επαναφοράς καθορίζουν μαζί πόσο γρήγορα αποκρίνεται το σύστημα ανάγνωσης. Όσο πιο αργό είναι το σύστημα ανάγνωσης τόσο καλύτερη είναι η συμπεριφορά θορύβου του οπότε πρέπει να γίνει ένας συμβιβασμός ανάλογα και με την εφαρμογή.

3.2.4 Ρυθμός άφιξης γεγονότων

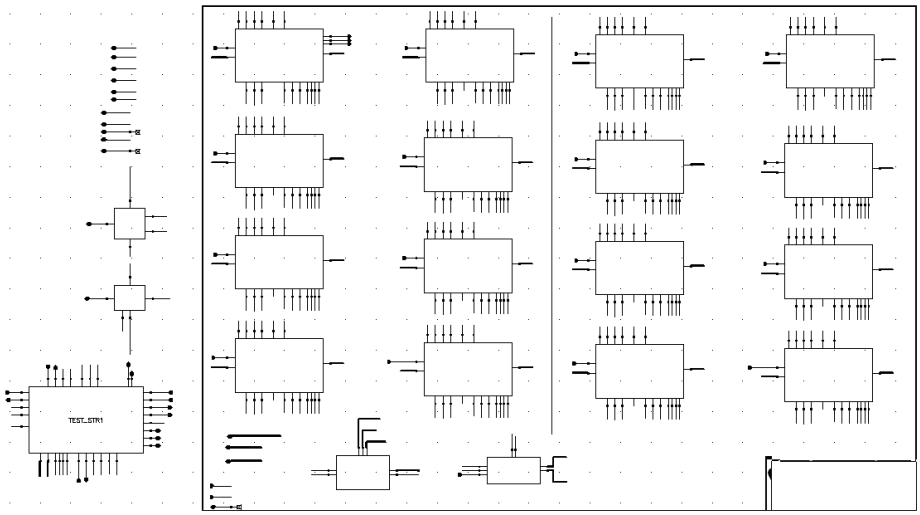
Ο ρυθμός άφιξης γεγονότων καθορίζει το ελάχιστο χρονικό διάστημα που αναμένεται ανάμεσα σε δύο διαδοχικές διεγέρσεις. Ο ρυθμός άφιξης γεγονότων είναι αντιστρόφως ανάλογος του χρόνου επαναφοράς. Αν το αναλογικό σήμα εξόδου δεν έχει επανέρθει στην στάθμη ηρεμίας και το κανάλι δεχτεί νέα διέγερση τότε θα υπάρξει μετατόπιση στάθμης και σταδιακά το σύστημα ανάγνωσης θα τεθεί εκτός λειτουργίας.



Εικόνα 3-2 Κορυφή της ιεραρχίας.

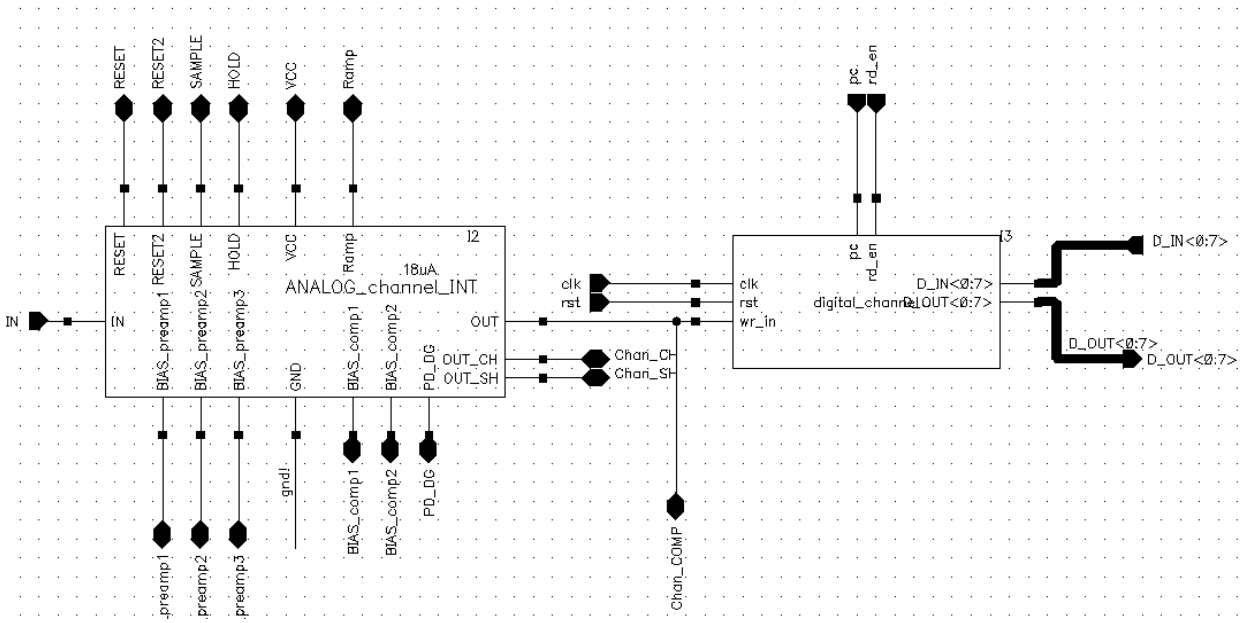
Στην Εικόνα 3-2 Κορυφή της ιεραρχίας.φαίνεται το σχηματικό διάγραμμα του κυκλώματος υπό – έλεγχο, το οποίο αποτελεί έναν πολύ-κάναλο ενισχυτή και μετατροπέα αναλογικού σήματος σε ψηφιακό. Το συγκεκριμένο ολοκληρωμένο κύκλωμα είναι σχεδιασμένο στην υπο-μικρονική τεχνολογία 0.35 της AMS και πραγματοποιεί ενίσχυση του σήματος, το οποίο λαμβάνει από ανιχνευτή ακτινοβολίας CdTe σε διάταξη pixel, και επιπλέον μετατρέπει το σήμα αυτό σε ψηφιακό. Σε κάθε pixel λοιπόν του ανιχνευτή αντιστοιχεί και ένα pixel ηλεκτρονικών ανάγνωσης με στόχο την ελαχιστοποίηση του θορύβου και την δημιουργία μιας κάμερας ακτίνων χ και γάμμα. Στην παραπάνω εικόνα φαίνονται οι εξωτερικές

συνδέσεις του κυκλώματος και αποτελεί την κορυφή της ιεραρχίας του σχηματικού διαγράμματος. Στο εσωτερικό του εξαρτήματος φαίνονται και οι ονομασίες των εξωτερικών συνδέσεων, των οποίων θα ακολουθήσει περιγραφή.



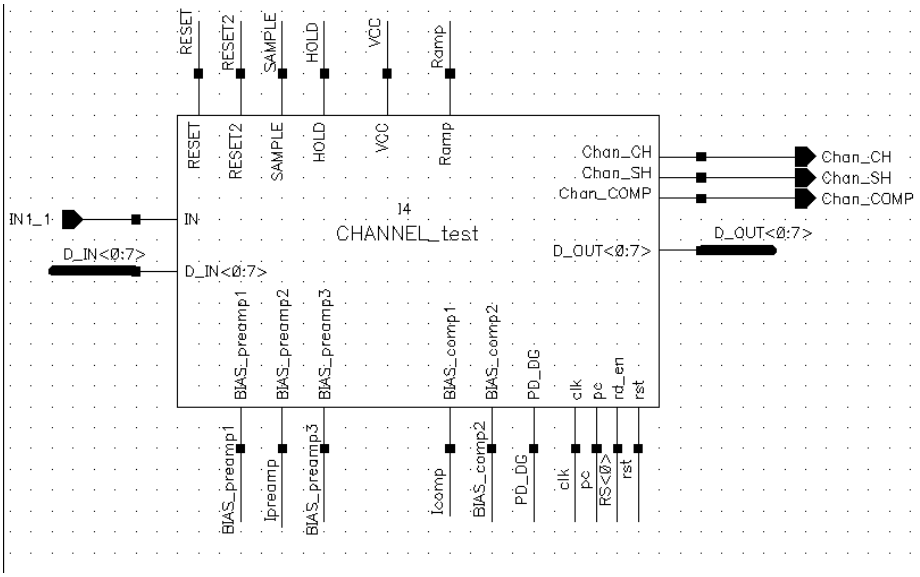
Εικόνα 3-3 Υπο-κυκλώματα του ASIC.

Στην Εικόνα 3-3 Υπο-κυκλώματα του ASIC. βλέπουμε τις δομικές μονάδες του ολοκληρωμένου κυκλώματος , στο δεύτερο επίπεδο ιεραρχίας. Το κύκλωμα που μελετήθηκε αποτελείται από δεκαέξι κανάλια και σχεδιάστηκε για την μελέτη ηλεκτρονικών ανάγνωσης σε επίπεδο pixel και δεν ήταν εξαρχής προορισμένο για να συνδεθεί με ανιχνευτή. Εντός του πλαισίου βρίσκονται τα δεκαέξι υπο-κυκλώματα, τα οποία στο εξής θα καλούνται pixel, κάθε ένα από τα οποία αντιστοιχεί σε ένα pixel του ανιχνευτή ακτινοβολίας και είναι πανομοιότυπα. Το αναλογικό σήμα που συλλέγεται σε κάθε pixel μετατρέπεται σε ψηφιακή πληροφορία των οκτώ bit και μπορεί να αναγνωσθεί από έναν ηλεκτρονικό υπολογιστή την βοήθεια των κατάλληλων ηλεκτρονικών επιπέδου πλακέτας η σχεδίαση των οποίων θα περιγραφεί αναλυτικά παρακάτω. Επίσης στην Εικόνα 3-3 Υπο-κυκλώματα του ASIC.διακρίνονται και τα μπλόκ τροφοδοσίας τα ψηφιακά υπο-κυκλώματα ελέγχου, αλλά και βασικές μονάδες που σχεδιάστηκαν για λόγους ελέγχου.



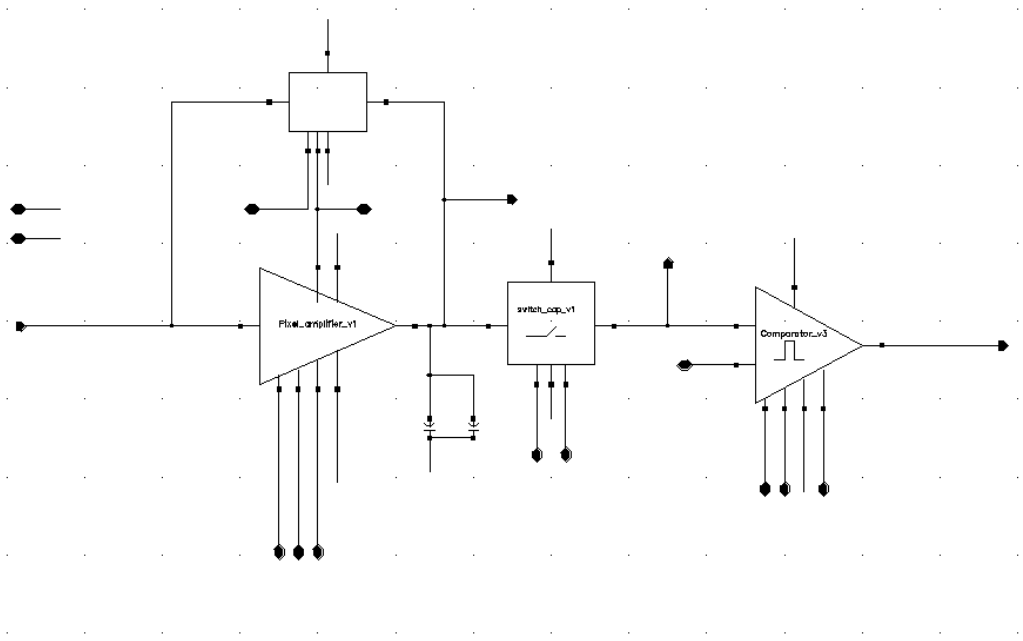
Εικόνα 3-4 Σχηματικό διάγραμμα ενός pixel

Τα περιεχόμενα του κάθε ένα από τα δεκαέξι Pixel ηλεκτρονικών ανάγνωσης φαίνονται στην Εικόνα 3-4 Σχηματικό διάγραμμα ενός pixel. Έχουμε λοιπόν βασικά δομικά στοιχεία, το αναλογικό τμήμα (αριστερά) και το ψηφιακό (δεξιά). Στο αναλογικό τμήμα της αλυσίδας συντελείται η μετατροπή του ρεύματος που προέρχεται από τον ανιχνευτή σε τάση με χρήση ενός ενισχυτή διεμπέδησης. Ο ενισχυτής είναι σε διάταξη ολοκληρωτή ο οποίος ολοκληρώνει το σήμα εισόδου για κάποιο καθοριζόμενο από τον χρήστη χρονικό διάστημα. Μετά το τέλος του καθορισμένου χρόνου ολοκλήρωσης προκύπτουν στην έξοδο του ενισχυτή παλμοί τάσης ανάλογοι του προσπίπτοντος ρεύματος, οι οποίοι πρέπει να ψηφιοποιηθούν.

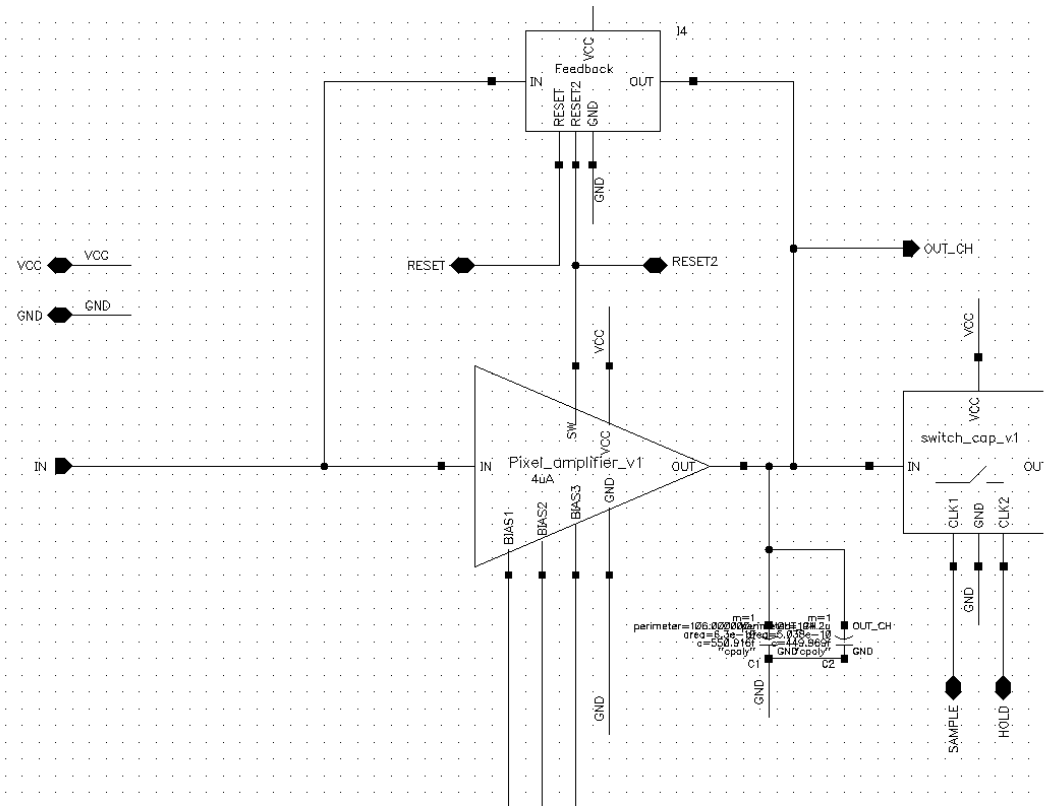


Εικόνα 3-5 Pixel 1_1 Test Channel

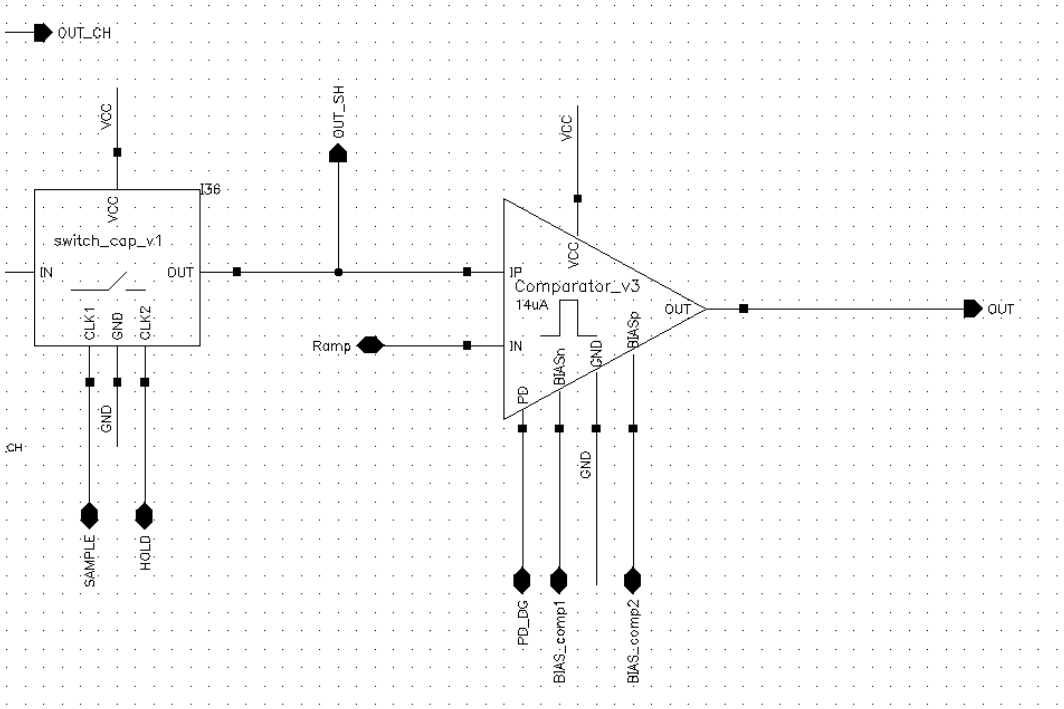
Στην συνέχεια το σήμα αποθηκεύεται σε έναν πυκνωτή στην επόμενη βαθμίδα, που είναι ένα κύκλωμα δειγματοληψίας και αποθήκευσης (Sample and Hold – S&H) και τελικά μετατρέπεται σε ψηφιακή πληροφορία ενός ψηφίου, 1 bit στον συγκριτή (Comparator).



Εικόνα 3-6 Αναλογικές βαθμίδες



Εικόνα 3-7 Ενισχυτής φορτίου και η ανατροφοδότηση του

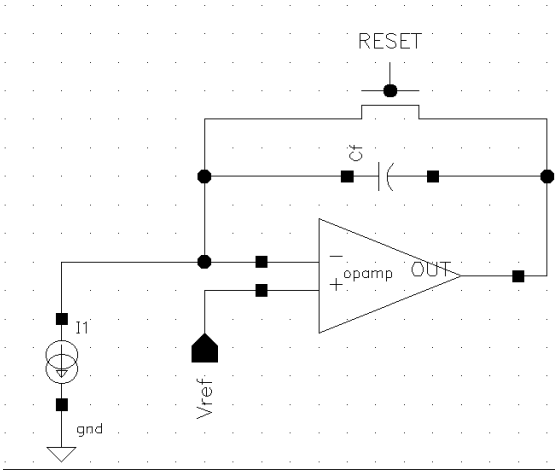


Εικόνα 3-8 Κύκλωμα δειγματοληψίας και αποθήκευσης και συγκριτής

3.3 Ενισχυτής Διεμπέδησης

Το σχηματικό διάγραμμα ενός τυπικού ενισχυτή διεμπέδησης φαίνεται στην Εικόνα 3-9 Ενισχυτής διεμπέδησης. Το ρεύμα που προέρχεται από τον ανιχνευτή ολοκληρώνεται στην χωρητικότητα ανάδρασης και η τάση στην έξοδο, δίνεται από την σχέση:

$$V_{out} = V_{ref} - \left(\frac{I}{C_f}\right)t \qquad (1.1)$$

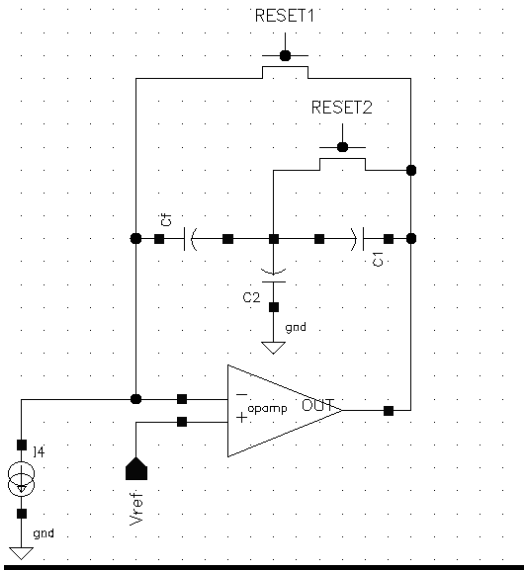


Εικόνα 3-9 Ενισχυτής διεμπέδησης

Όταν ο διακόπτης κλείσει, τότε η τάση εξόδου γίνεται ίση με την τάση αναφοράς $V_{out} = V_{ref}$. Στην έξοδο του ενισχυτή διεμπέδησης η τάση είναι πριονωτή με μέγιστη τιμή την τάση αναφοράς V_{ref} και ρυθμό πτώσης και ελάχιστη τιμή που εξαρτώνται από τον χρόνο που γίνεται εκκαθάριση (reset). Σε περίπτωση που δεν γίνει εκκαθάριση ο ενισχυτής διεμπέδησης θα οδηγηθεί σε κόρο με αποτέλεσμα να σταματήσει να λειτουργεί σωστά οπότε και δεν θα ολοκληρώνει το ρεύμα του ανιχνευτή. Από την προηγούμενη σχέση προκύπτει επίσης ότι για να έχουμε μεγαλύτερο κέρδος μετατροπής του ρεύματος σε τάση είναι επιθυμητό η χωρητικότητα ανάδρασης να είναι όσο το δυνατόν μικρότερη, γεγονός όμως το οποίο αυξάνει τον θόρυβο FPN (Fixed Pattern Noise). Ένας τρόπος για να έχει ο ενισχυτής διεμπέδησης ταυτόχρονα υψηλό κέρδος μετατροπής και χαμηλό θόρυβο είναι η υλοποίηση του δικτυώματος ανάδρασης που φαίνεται στο παρακάτω σχήμα. Το δικτύωμα αυτό που αποτελεί έναν χωρητικό διαιρέτη τάσης έχει σαν αποτέλεσμα η τάση εξόδου να δίνεται πλέον από την σχέση:

$$V_{out} = J_1 - J_2\left(\frac{I}{C_f}\right)t \tag{1.2}$$

Όπου J_1 σταθερά και $J_2 = 1 + \frac{C_2}{C_f} + \frac{C_f}{C_1}$



Εικόνα 3-10 Ενισχυτής διεμπέδησης με την τοπολογία ανάδρασης που επιλέχθηκε

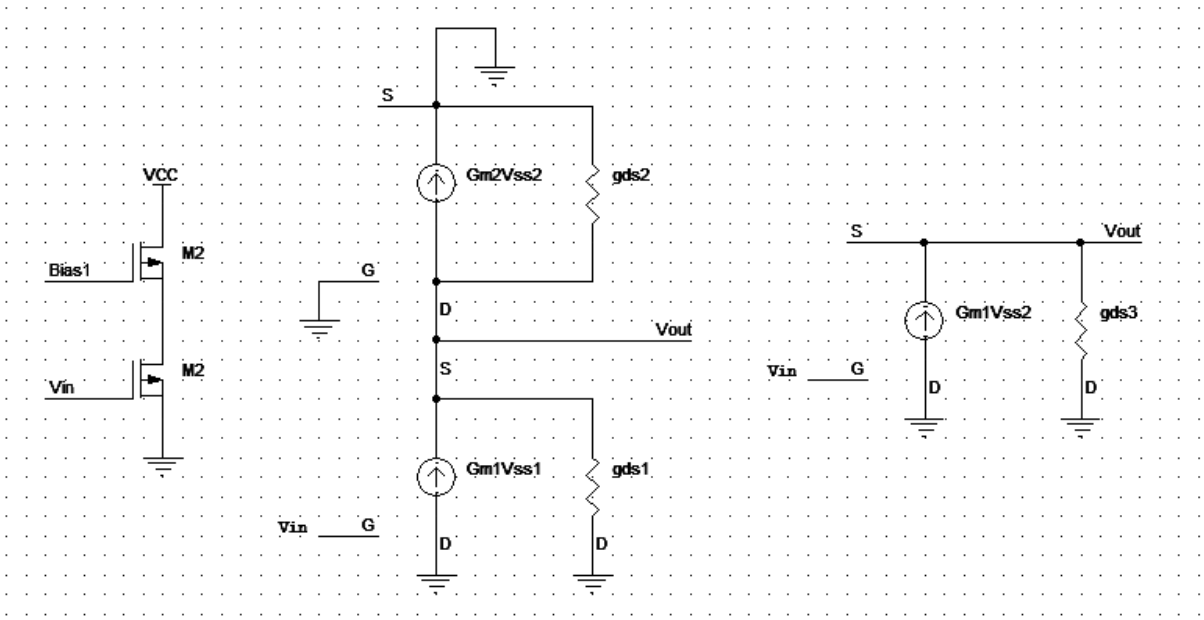
Όπως προκύπτει από την σχέση για την τάση στην έξοδο, αφού για την σταθερά J_2 ισχύει $J_2 > 1$ είναι δυνατόν να επιλέξουμε αισθητά μεγαλύτερη τιμή για την χωρητικότητα C_f και ταυτόχρονα να έχουμε υψηλό κέρδος μετατροπής. Χρησιμοποιώντας μεγαλύτερη τιμή C_f τόσο οι σχετικές αποκλίσεις είναι μικρότερες οπότε και ο αντίστοιχος θόρυβος (FPN). Ο διακόπτης RESET1 υλοποιείται με ένα απλό NMOS ενώ ο διακόπτης RESET2 από μια πύλη μετάδοσης(transmission gate). Τα δύο αυτά σήματα έχουν ως σκοπό να επαναφέρουν τις τάσεις στους κόμβους και να μην επιτρέψουν να φτάσουν στον κόρο. Οι δύο διακόπτες κλείνουν ταυτόχρονα για να γίνει η αρχικοποίηση του ενισχυτή διεμπέδησης ενώ στη συνέχεα ανοίγουν οπότε και αρχίζει η ολοκλήρωση του ρεύματος. Όσο το δυνατόν αργότερος είναι ο ρυθμός που ανοίγει ο διακόπτης RESET τόσο μικρότερος είναι και ο αντίστοιχος θόρυβος.

Ο ενισχυτής αποτελείται από δύο στάδια το πρώτο εκ των οποίων είναι ένας απλός απομονωτής με PMOS τρανζίστορ εισόδου για να ελαχιστοποιείται ο θόρυβος flicker. Το δεύτερο στάδιο είναι ένας κασκωδικός ενισχυτής ενώ τα δύο στάδια είναι AC συνδεδεμένα.

Μέσω ενός επιπλέον διακόπτη που ελέγχεται με το ίδιο σήμα του διακόπτη RESET2 ο ενδιάμεσος κόμβος αρχικοποιείται και το δεύτερο στάδιο πολώνεται.

3.3.1 Πρώτο στάδιο – Απομονωτής

Ως πρώτο στάδιο ενίσχυσης έχει επιλεγεί ένας απομονωτής με PMOS τρανζίστορ στην είσοδο. Εάν ένα τέτοιο στάδιο πρέπει να οδηγήσει μια μικρή εμπέδηση φόρτου, τότε ένας απομονωτής πρέπει να χρησιμοποιηθεί για να οδηγήσει το φορτίο με ελάχιστη μείωση του επιπέδου της τάσης. Ο ακόλουθος πηγής μπορεί να χρησιμοποιηθεί ως ένας τέτοιος απομονωτής. Οι ακόλουθοι πηγής παρουσιάζουν υψηλή αντίσταση εισόδου και μεσαία αντίσταση εξόδου. Η μη γραμμικότητα που παρουσιάζουν εξαιτίας της μη γραμμικής εξάρτησης της V_{TH} με το δυναμικό της πηγής και του φαινομένου σώματος, μπορεί να εξαλειφτεί με χρήση PMOS τρανζίστορ. Στην Εικόνα 3-11 Ισοδύναμο σχηματικό διάγραμμα μικρού σήματος του απομονωτήβλέπουμε τον ακόλουθο πηγής που χρησιμοποιήθηκε και τα ισοδύναμα σχηματικά διαγράμματα μικρού σήματος.



Εικόνα 3-11 Ισοδύναμο σχηματικό διάγραμμα μικρού σήματος του απομονωτή

Σύμφωνα με το απλοποιημένο σχηματικό διάγραμμα μικρού σήματος, έχουμε,

$$gm_1\left(U_{IN}-U_{OUT}\right)=U_{OUT}\left(gds_1+gds_2\right) \quad (1.3)$$

$$\frac{U_{OUT}}{U_{IN}} = \frac{gm_1}{gm_1 + gds_1 + gds_2} \qquad (1.4)$$

και επειδή $gm_1 >> gds_1 + gds_2 \Rightarrow \frac{U_{OUT}}{U_{IN}} \approx 1$

3.3.2 Δεύτερο στάδιο – Κασκωδικός Ενισχυτής

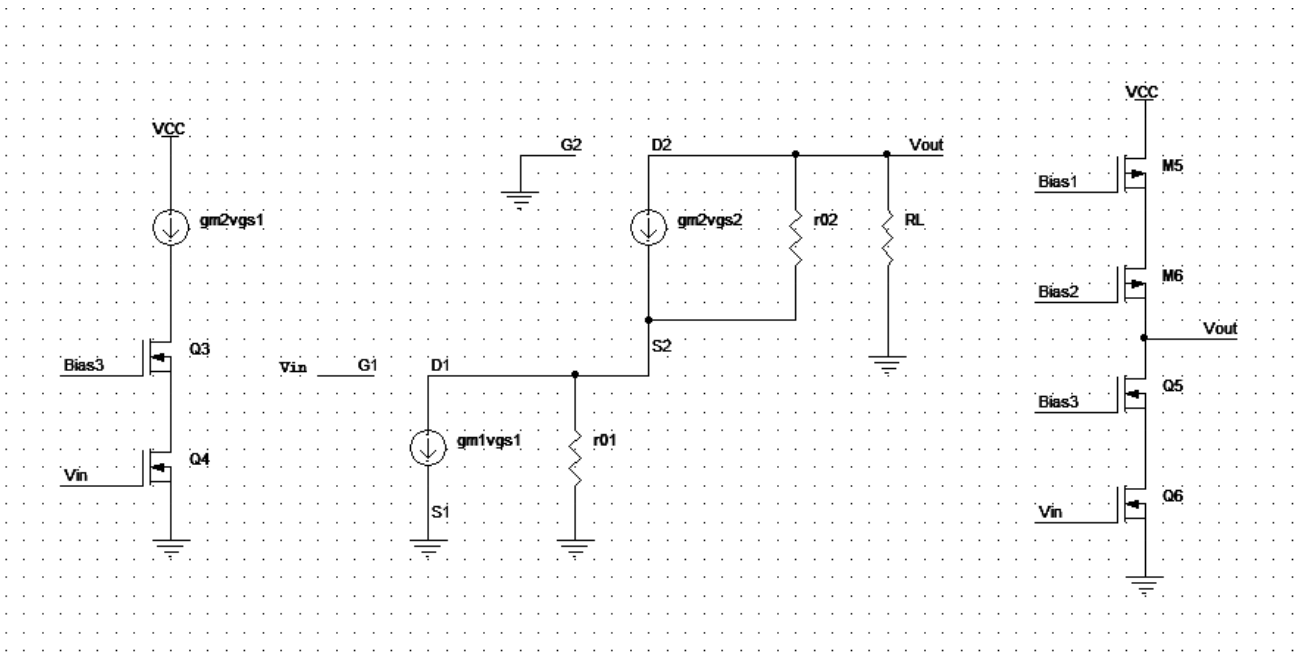
Η έξοδος του πρώτου σταδίου είναι συνδεδεμένη μέσω ενός πυκνωτή με την είσοδο του επόμενου σταδίου, που αποτελείται από έναν κασκωδικό ενισχυτή τεσσάρων τρανζίστορ. Αρχικά για την ανάλυση του εν λόγω ενισχυτή θεωρούμε ότι ο κασκωδικός καθρέπτης ρεύματος (τα δύο PMOS) είναι ισοδύναμα με μία πηγή ρεύματος όπως φαίνεται στο παρακάτω σχήμα. Έχουμε:

$$\frac{v_{out} - v_{S1}}{r_{02}} - g_{m2}v_{S1} = 0 \Rightarrow v_{out} = (1 + g_{m2}r_{02})v_1 \qquad (1.5)$$

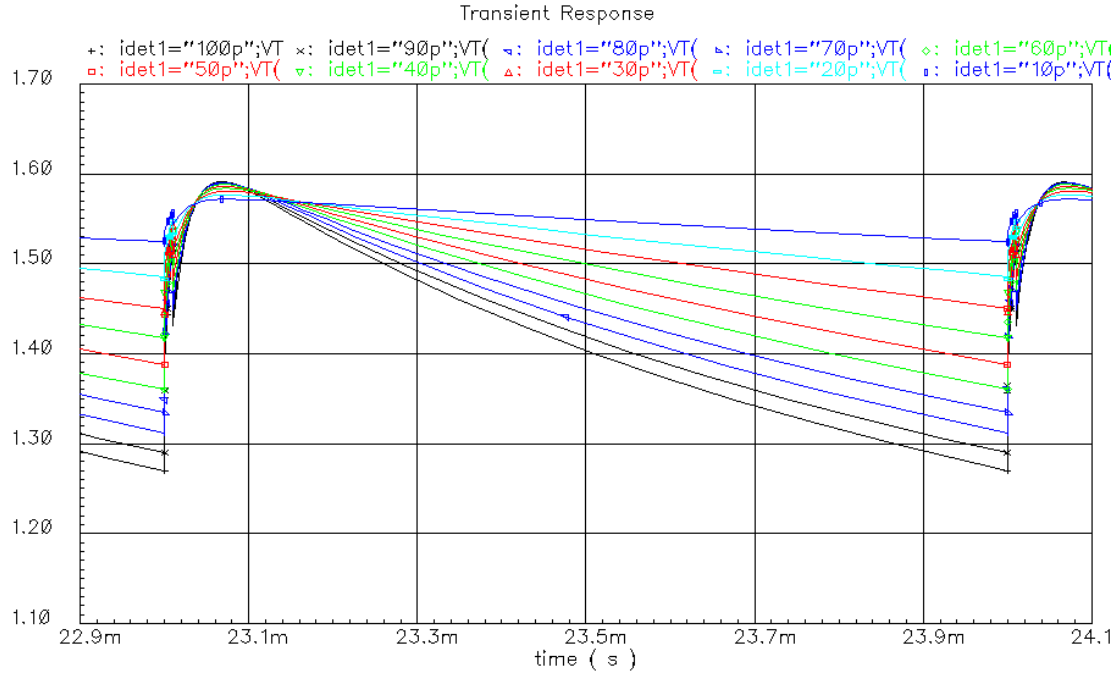
$$\frac{v_{S1}}{r_{02}} + g_{m1}v_{in} = 0 \Rightarrow v_{S1} = -g_{m1}r_{01}v_{in} \qquad (1.6)$$

Επομένως για το κέρδος του ενισχυτή έχουμε:

$$A = \frac{v_{out}}{v_{in}} = -g_{m1}r_{01}(1 + g_{m2}r_{02}) \approx -g_{m1}r_{01}g_{m2}r_{02} \qquad (1.7)$$

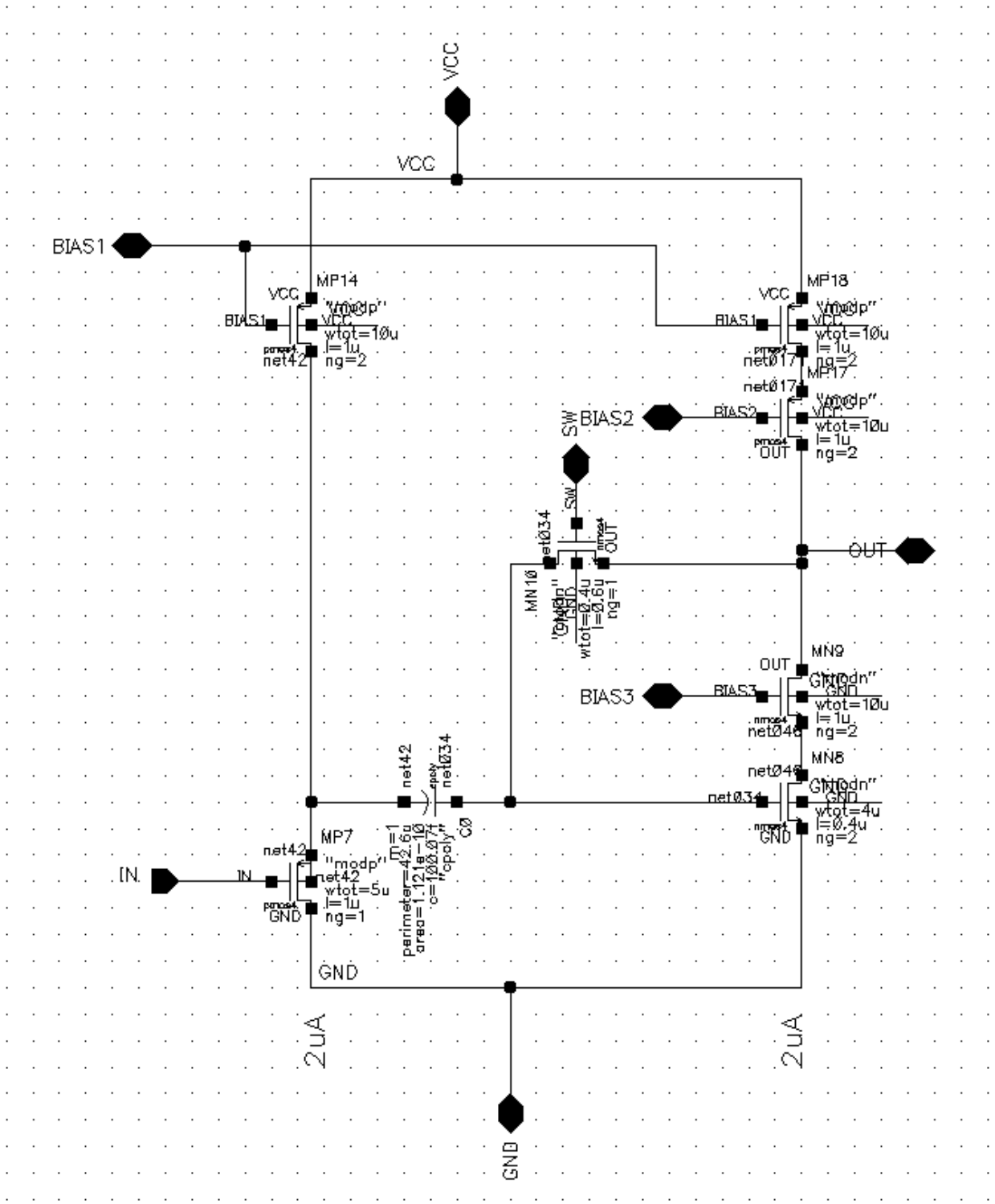


Εικόνα 3-12 Ισοδύναμο σχηματικό διάγραμμα μικρού σήματος του κασκωδικού ενισχυτή



Εικόνα 3-13 Έξοδος Ενισχυτή Διεμπέδησης

Το τελικό σχηματικό του ενισχυτή φαίνεται στην Εικόνα 3-14 Τοπολογία ενισχυτή φορτίου, στο οποίο φαίνεται και το τρανζίστορ M10 το οποίο συνδέει την έξοδο του πρώτου σταδίου με την έξοδο του δευτέρου.

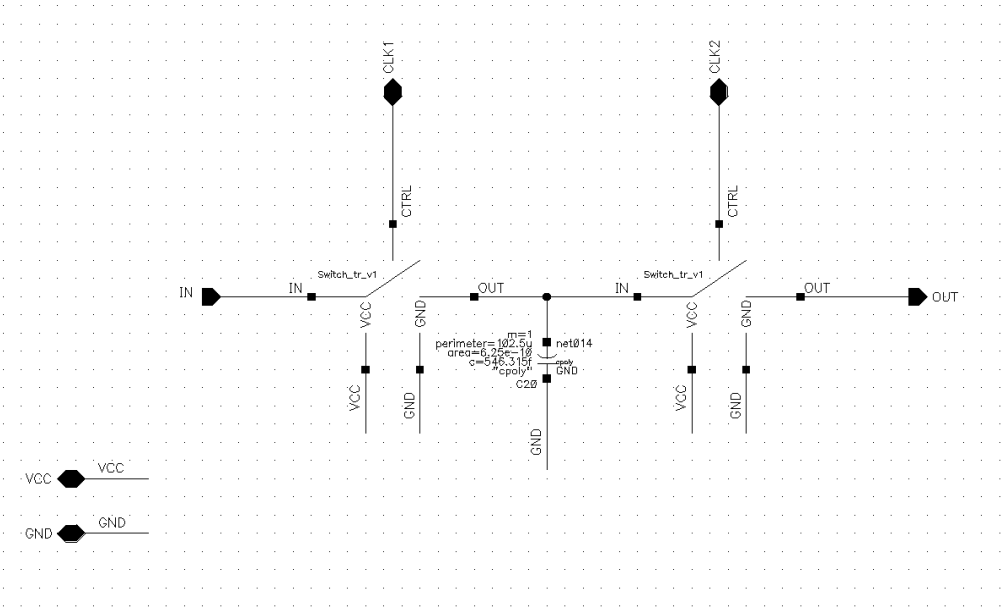


Εικόνα 3-14 Τοπολογία ενισχυτή φορτίου

3.4 Κύκλωμα Δειγματοληψίας και Αποθήκευσης

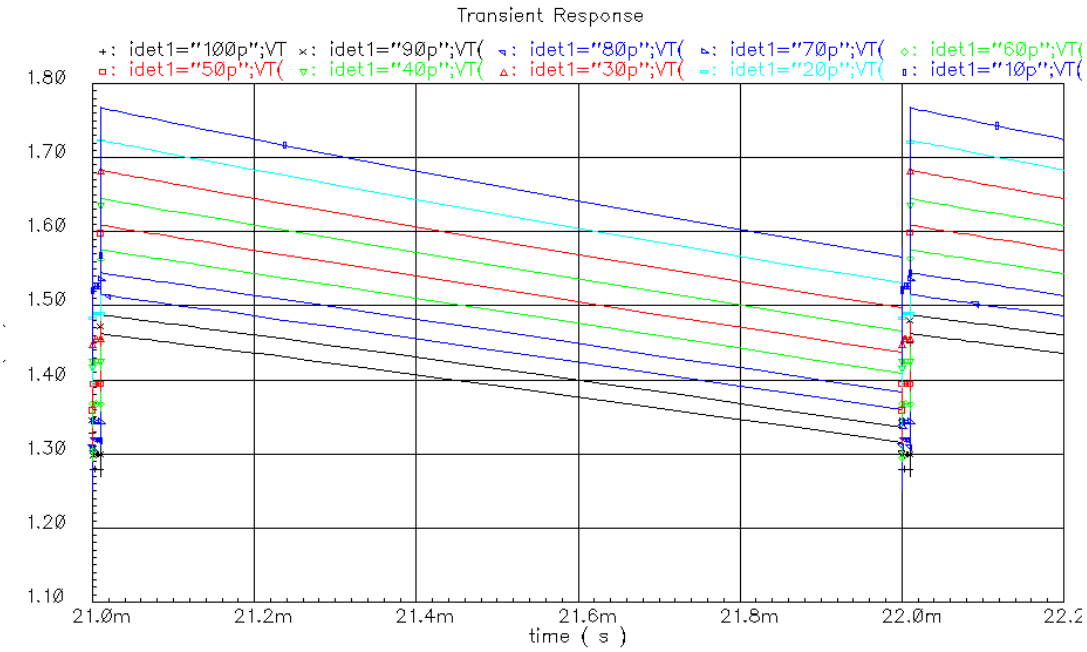
Το σήμα στην έξοδο του ενισχυτή διεμπέδησης είναι παλμοί τάσης που μεταβάλλονται μεταξύ κάποιων ορίων. Για να αποθηκεύτεί η αναλογική τάση κατά την περίοδο που ο

ενισχυτής διεμπέδησης αρχικοποιείται χρησιμοποιείται ένα κύκλωμα δειγματοληψίας και αποθήκευσης. Το πλήρες σχηματικό του φαίνεται στο παρακάτω σχήμα όπου και οι δύο διακόπτες έχουν υλοποιηθεί με πύλες μετάδοσης με έμφαση στο να έχουν μικρή αντίσταση αγωγής. Το συγκεκριμένο κύκλωμα ταυτόχρονα αποτελεί και φίλτρο για τα μεταβατικά φαινόμενα εξαιτίας των διακοπτών καθώς έχει περιορισμένο εύρος συχνοτήτων. Ο πρώτος διακόπτης (SAMPLE) επιτρέπει κατά την διάρκεια ολοκλήρωσης η τάση να αποθηκεύεται στον πυκνωτή ενώ κατά την διάρκεια της αρχικοποίησης του ενισχυτή διεμπέδησης απομονώνει το κύκλωμα δειγματοληψίας. Ταυτόχρονα ο δεύτερος διακόπτης (HOLD) κλείνει και επιτρέπει στην αποθηκευμένη τιμή στον πυκνωτή να εφαρμοστεί στην είσοδο του συγκριτή και να αρχίσει η διαδικασία μετατροπής.

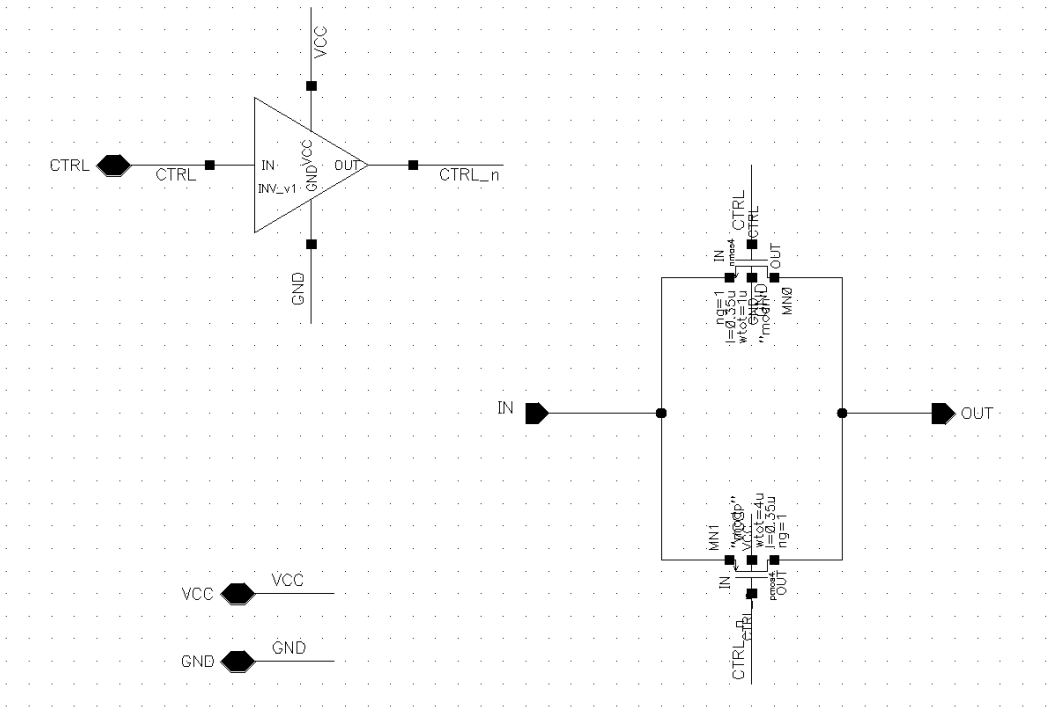


Εικόνα 3-15 Κύκλωμα Δειγματοληψίας και Αποθήκευσης

Η μεταβατική απόκριση του κυκλώματος για διάφορες τιμές του ρεύματος εισόδου (10pA-100pA) φαίνεται στην Εικόνα 3-16 Έξοδος του κυκλώματος δειγματοληψίας και αποθήκευσης. Είναι φανερό ότι στην έξοδο του κυκλώματος υπάρχει ένα σήμα τάσης χωρίς έντονα μεταβατικά φαινόμενα η τιμή του οποίου εξαρτάται από το ρεύμα εισόδου.



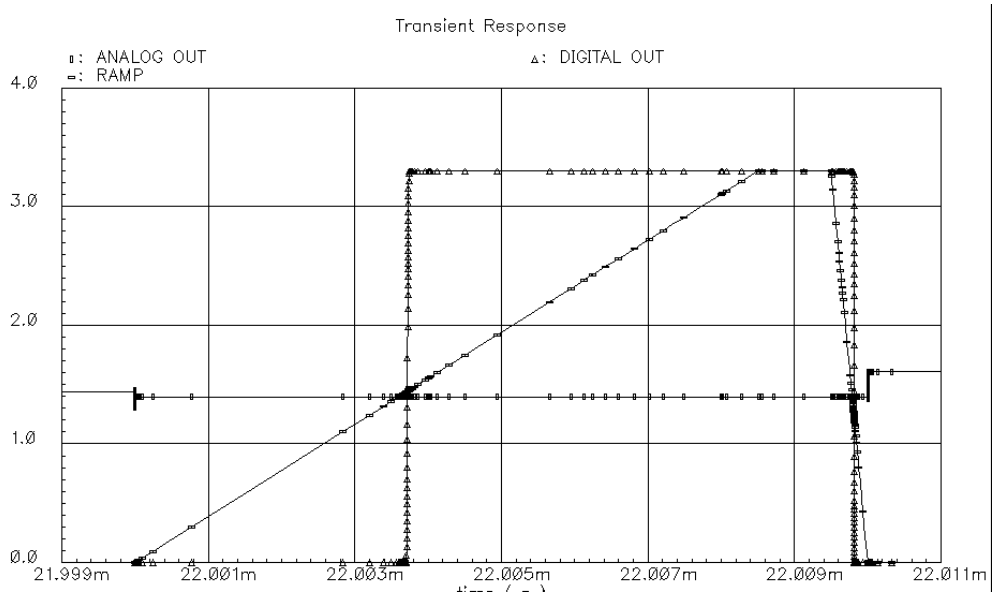
Εικόνα 3-16 Έξοδος του κυκλώματος δειγματοληψίας και αποθήκευσης



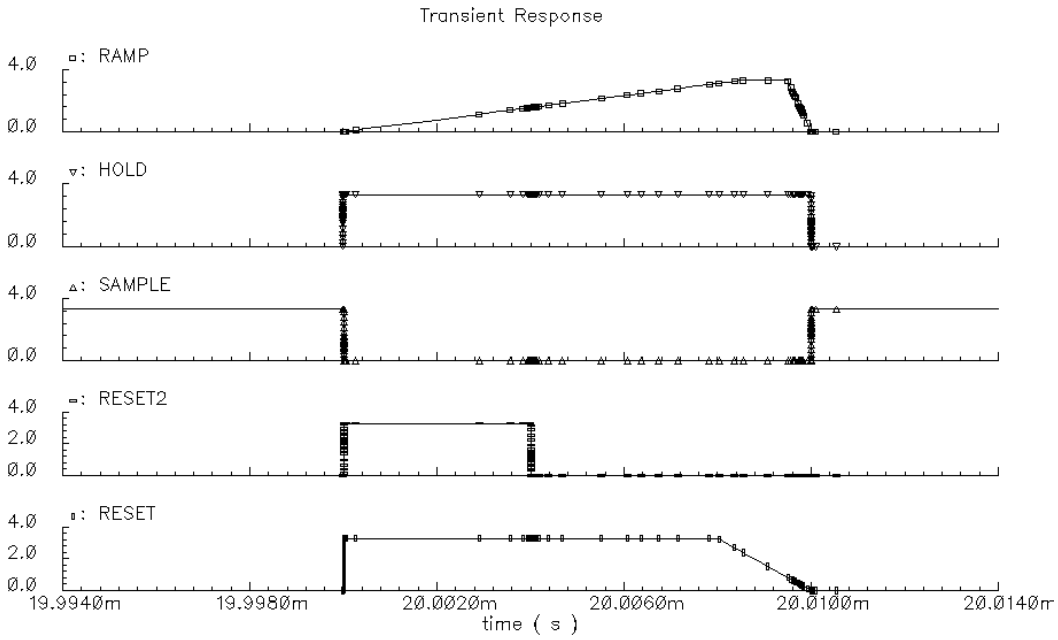
Εικόνα 3-17 Υλοποίηση του κυκλώματος δειγματοληψίας και αποθήκευσης

3.5 Μετατροπέας αναλογικού σήματος σε ψηφιακό.

Αφού η τάση στην έξοδο του κυκλώματος S&H είναι σταθερή μπορεί πλέον να μετατραπεί το αναλογικό σήμα σε ψηφιακό το οποίο και στην συνέχεια θα αποθηκευτεί στην μνήμη του συστήματος. Ο μετατροπέας ουσιαστικά αποτελείται από ένα απλό συγκριτή που στην έξοδο του παράγει σήματα 1 bit. Ο συγκριτής αποτελείται από τρεις βαθμίδες, έναν απομονωτή, έναν δυσταδιακό ενισχυτή και έναν αντιστροφέα. Ο απομονωτής έχει σαν αποτέλεσμα να μετατοπίζει την στάθμη στα επιθυμητά επίπεδα και υλοποιείται από έναν ακόλουθο εκπομπού. Ο δυσταδιακός ενισχυτής έχει το ιδιαίτερο χαρακτηριστικό ότι συνδέοντας δύο επιπλέον τρανζίστορ 'χιαστί' στο φορτίο του πρώτου σταδίου δημιουργούμε μια ελεγχόμενη υστέρηση στον συγκριτή. Η δημιουργία υστέρησης είναι σημαντική καθώς με αυτό τον τρόπο αποφεύγονται τα λανθασμένα σήματα στην έξοδο του μετατροπέα. Το τελευταίο στάδιο αποτελείται από έναν αντιστροφέα ο οποίος βελτιώνει την μορφή των παραγόμενων παλμών και ταυτόχρονα έχει την δυνατότητα να φορτίσει την επόμενη βαθμίδα του ψηφιακού τμήματος της αλυσίδας. Το αναλογικό σήμα που έχει αποθηκευτεί στο τέλος της φάσης ολοκλήρωσης στην έξοδο του κυκλώματος S&H συγκρίνεται με ένα εξωτερικό σήμα που έχει την μορφή ράμπας και το οποίο έχει συγχρονιστεί με το ψηφιακό τμήμα της αλυσίδας. Η φάση μετατροπής ξεκινάει με την έναρξη του σήματος ράμπας και την σύγκριση του με το σήμα που έρχεται από το αναλογικό τμήμα της αλυσίδας. Τα σήματα αυτά φαίνονται και στην Εικόνα 3-18 Γράφημα απόκρισης του συγκριτή όταν ενεργοποιηθείόπου διακρίνονται τόσο τα σήματα εισόδου του συγκριτή όσο και η ψηφιακή του έξοδος. Είναι φανερό πως όταν το σήμα ράμπας γίνει μεγαλύτερο από την στάθμη που έχει αποθηκευτεί στο κύκλωμα S&H τότε στην έξοδο του μετατροπέα παράγεται ένας ψηφιακός παλμός.



Εικόνα 3-18 Γράφημα απόκρισης του συγκριτή όταν ενεργοποιηθεί

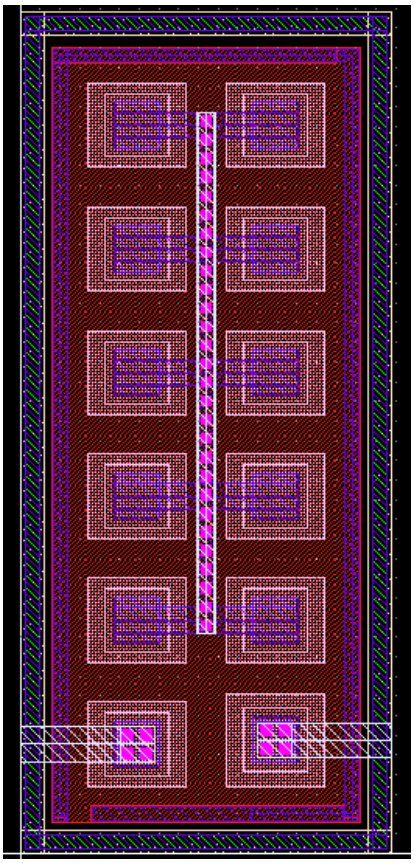


Εικόνα 3-19 Σήματα RAMP, SAMPLE, HOLD, RESET και RESET2

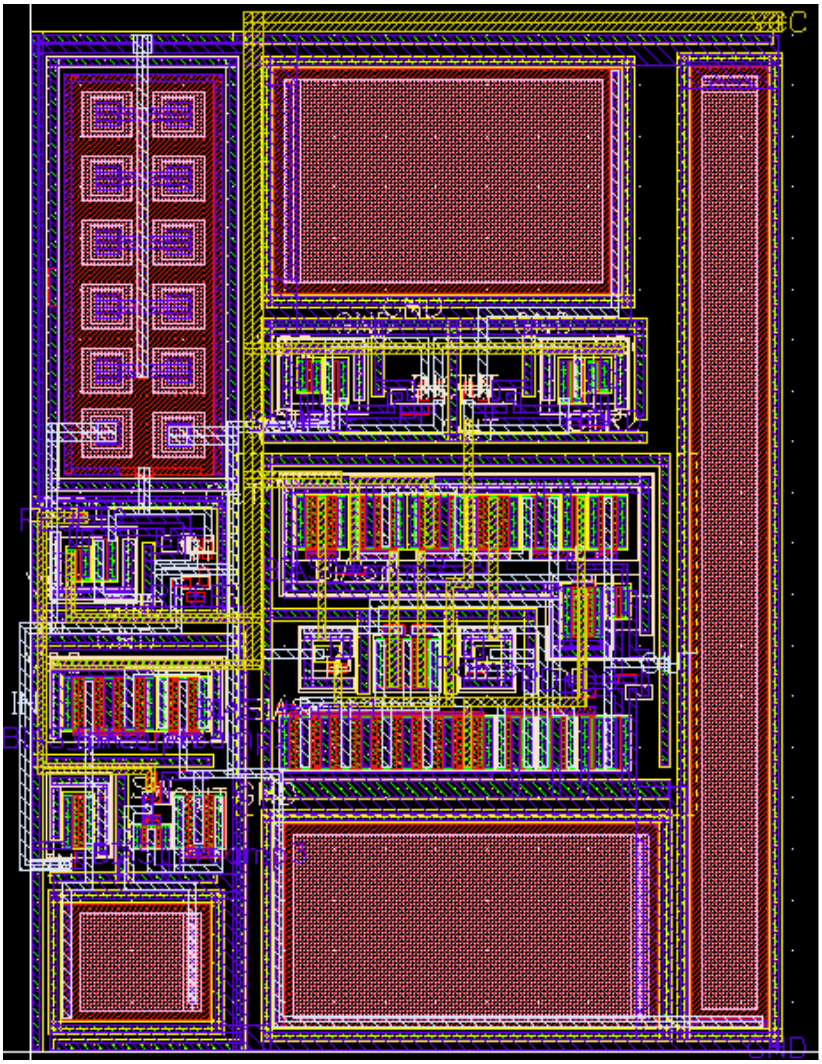
Στο παραπάνω σχήμα φαίνονται τα σήματα ελέγχου της αναλογικής αλυσίδας. Τα δύο σήματα που αρχικοποιούν τον ενισχυτή διεμπέδησης (RESET1 και RESET2) και επαναφέρουν τους κόμβους στις αρχικές τους τάσεις, τα σήματα SAMPLE και HOLD που διαχειρίζονται τα σήματα του κυκλώματος δειγματοληψίας και αποθήκευσης και το σήμα ράμπας του συγκριτή.

3.5.1 Φυσικός σχεδιασμός.

Το ολοκληρωμένο κύκλωμα σχεδιάστηκε στην υπομικρονική τεχνολογία 0.35um CMOS της AMS που παρέχει δύο επίπεδα πολυκρυσταλλικού πυριτίου και τέσσερα επίπεδα μετάλλου. Αυτό δίνει την δυνατότητα να υλοποιηθούν οι πυκνωτές όχι με την χρήση μετάλλων που θα είχε σαν αποτέλεσμα την αυξημένη περιοχή αλλά εκμεταλλευόμενοι την χωρητικότητα που δημιουργείται μεταξύ των δύο επιπέδων πολυκρυσταλλικού πυριτίου. Στην πραγματικότητα οι τρεις πυκνωτές που αποτελούν το δικτύωμα ανάδρασης υλοποιήθηκαν χρησιμοποιώντας την ίδια δομική μονάδα και είναι τοποθετημένοι στο ίδιο πηγάδι. Η τελική υλοποίηση φαίνεται στην Εικόνα 3-20 Φυσικός σχεδιασμός δικτυώματος ανάδρασης. όπου φαίνεται τόσο το πηγάδι όσο και η επαναλαμβανόμενη δομική μονάδα.



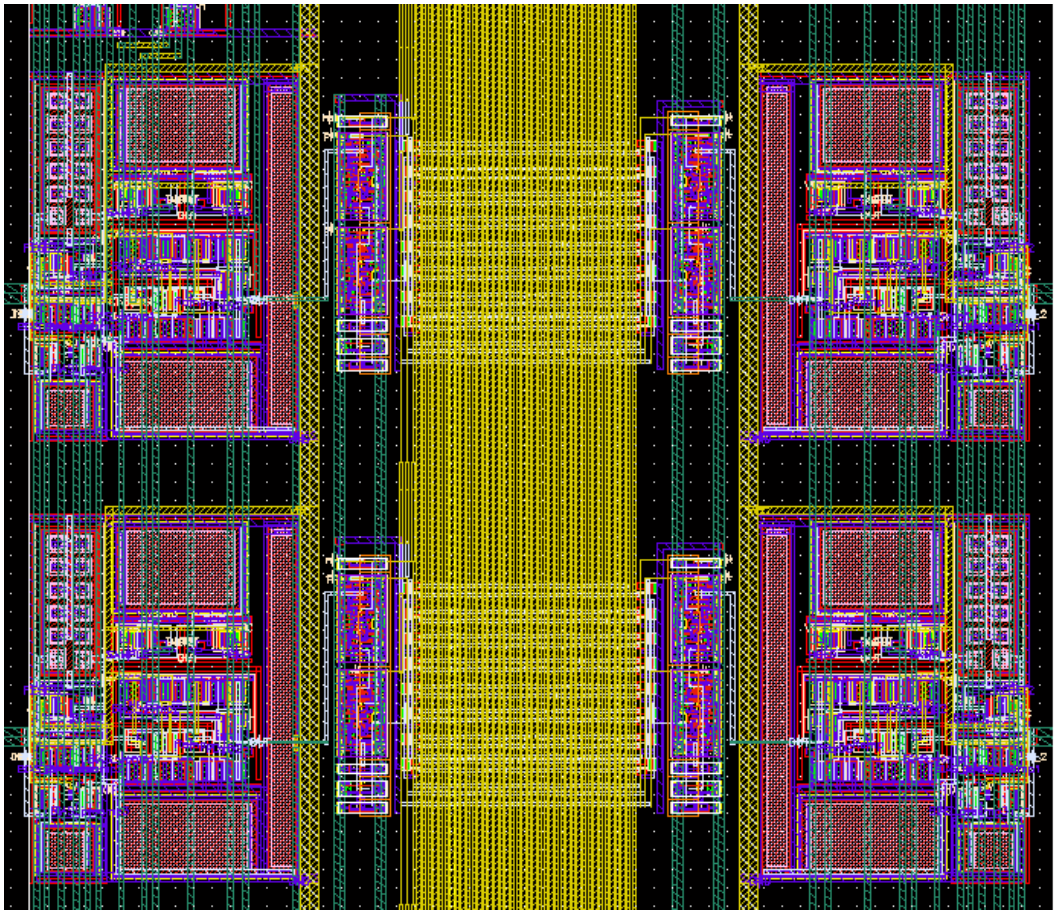
Εικόνα 3-20 Φυσικός σχεδιασμός δικτυώματος ανάδρασης.



Εικόνα 3-21 Φυσικός σχεδιασμός αναλογικής αλυσίδα επεξεργασίας.

Ο συνολικός φυσικός σχεδιασμός του αναλογικού τμήματος της αλυσίδας φαίνεται στην Εικόνα 3-21 Φυσικός σχεδιασμός αναλογικής αλυσίδα επεξεργασίας. και η διάσταση του είναι 100um x 75um. Στο κάτω αριστερά μέρος διακρίνεται ο ενισχυτής διεμπέδησης ενώ ακριβώς από πάνω του το δικτύωμα ανάδρασης. Στο πάνω δεξιά τμήμα είναι το κύκλωμα δειγματοληψίας και αποθήκευσης ενώ ακριβώς απο κάτω ο μετατροπέας αναλογικού σήματος σε ψηφιακό. Στο ολοκληρωμένο κύκλωμα τοποθετήθηκαν συνολικά 16 αλυσίδες επεξεργασίας διαχωρισμένες σε 8 γραμμές και δύο στήλες. Παρακάτω φαίνονται 4 αλυσίδες επεξεργασίας τοποθετημένες σε δύο γραμμές και δύο στήλες όπου διακρίνονται τόσο το

αναλογικό τμήμα της αλυσίδας όσο και το ψηφιακό. Η συνολική αλυσίδα επεξεργασίας, αναλογικό και ψηφιακό τμήμα, έχει διάσταση 100um x 100um.



Εικόνα 3-22 Φυσικός σχεδιασμός αλυσίδα επεξεργασίας(2x2).

3.6 Ψηφιακό Τμήμα του Ολοκληρωμένου Κυκλώματος

Το ψηφιακό τμήμα έχει ως κύριο σκοπό την μετατροπή της αναλογικής τάσης σε ψηφιακό σήμα, την αποθήκευση του ψηφιακού σήματος σε ένα κελί δυναμικής μνήμης 8 ψηφίων/bits και την περιοδική ανάκτηση/ανάγνωση των δεδομένων που αποθηκεύτηκαν για περεταίρω επεξεργασία. Το αναλογικό τμήμα της αλυσίδας έχει σαν κύρια λειτουργία να μετατρέπει το ρεύμα που έρχεται από τον ανιχνευτή σε τάση, να συγκρίνει την τιμή της τάσης με μια τάση

εισόδου τύπου ράμπας και να στέλνει ένα σήμα εξόδου προς το ψηφιακό κύκλωμα όταν οι δυο παραπάνω τάσεις γίνουν ίσες.

Σε μια γενική περιγραφή του όλου συστήματος, η λειτουργία του χωρίζεται σε τέσσερα στάδια:

1. μετατροπή ρεύματος σε τάση
2. μετατροπή αναλογικής τάσης σε ψηφιακό σήμα
3. εγγραφή του ψηφιακού σήματος στη μνήμη
4. ανάγνωση των δεδομένων της μνήμης.

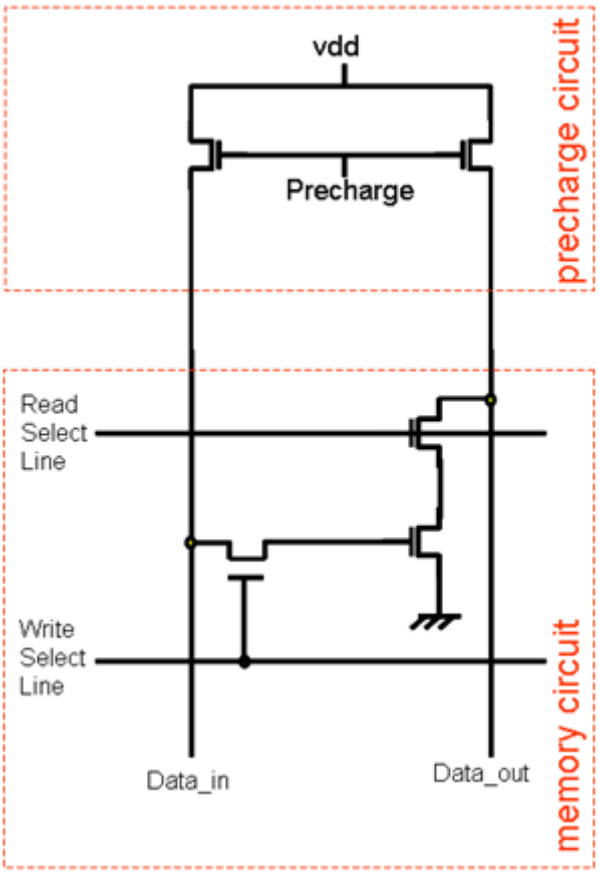
Στο $1^{ο}$ στάδιο είναι ενεργά μόνο τα αναλογικά κυκλώματα. Στο $2^{ο}$ στάδιο είναι ενεργά τόσο τα αναλογικά όσο και τα ψηφιακά κυκλώματα. Η τεχνική στην οποία βασίζεται η μετατροπή A/D είναι η ταυτόχρονη ενεργοποίηση μιας αναλογικής πηγής τάσης τύπου ράμπας με ένα ψηφιακό απαριθμητή. Ο ρυθμός με τον οποίο αυξάνεται η ράμπα είναι αντίστοιχος με εκείνον του απαριθμητή. Δηλαδή όταν ξεκινά η ράμπα ο απαριθμητής έχει την τιμή αρχικοποίησης και όταν η ράμπα πάρει την μέγιστη τιμή της, τότε και ο απαριθμητής έχει φτάσει την τελική του τιμή.

Στο $3^{ο}$ στάδιο είναι ενεργά και τα δύο τμήματα (αναλογικό και ψηφιακό). Όταν το σήμα εισόδου και η τιμή της ράμπας εξισωθούν, ο αναλογικός συγκριτής στέλνει ένα ασύγχρονο σήμα προς το ψηφιακό κύκλωμα εγγραφής μνήμης. Το τελευταίο παράγει ένα σύγχρονο ψηφιακό σήμα συγκεκριμένης διάρκειας και χρονισμού ώστε να γράψει/αποθηκεύσει την τιμή του απαριθμητή στο αντίστοιχο κελί μνήμης. Στο $4^{ο}$ στάδιο μόνο το ψηφιακό τμήμα είναι ενεργό. Σε αυτό το στάδιο παράγεται μια σειρά παλμών που ελέγχουν την ανάγνωση της μνήμης. Το ψηφιακό κύκλωμα ανάγνωσης είναι ένα και αφορά το σύνολο των κελιών της μνήμης σε αντίθεση με το αντίστοιχο κύκλωμα εγγραφής που είναι ενσωματωμένο σε κάθε κελί.

3.6.1 Κύκλωμα Δυναμικής Μνήμης (DRAM)

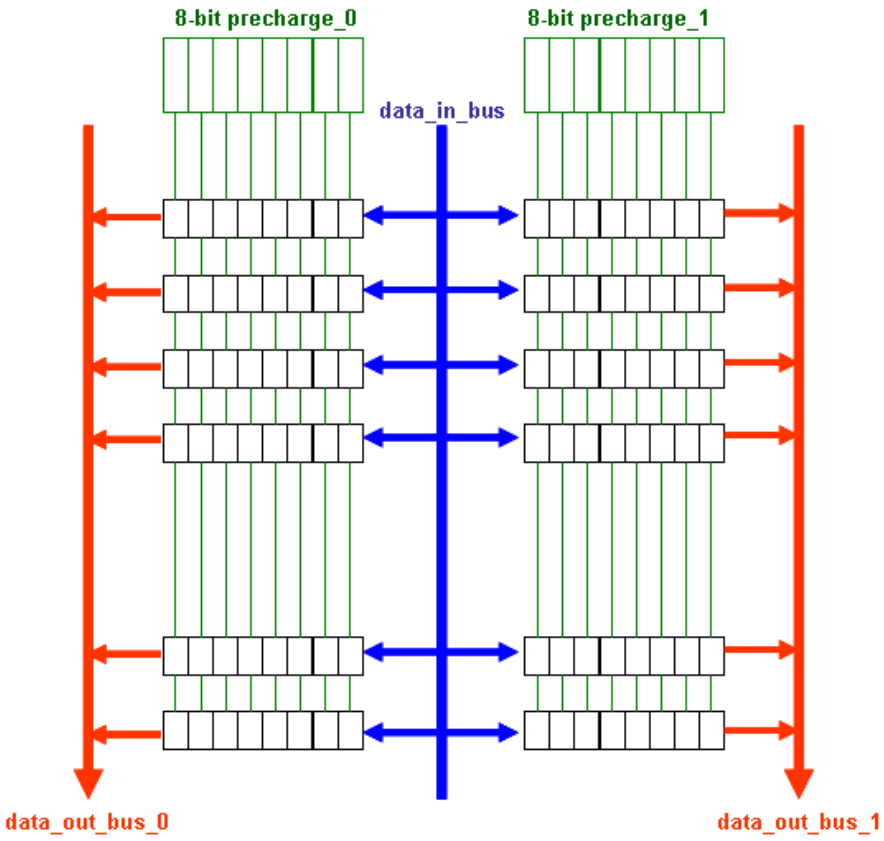
3.6.1.1 Αρχιτεκτονική

Η αρχιτεκτονική της δυναμικής μνήμης αποτελείται από πέντε transistor. Τα τρία αριστερά transistors υλοποιούν το κύκλωμα της μνήμης και τα δύο δεξιά το κύκλωμα precharge, που είναι ένα κύκλωμα φόρτισης της αρτηρίας πριν από την πρόσβαση στην μνήμη. Το κύκλωμα precharge είναι κοινό για όλα τα bit μνήμης που ανήκουν στην ίδια στήλη. Δηλαδή για μια μνήμη 16 γραμμών x 8 στηλών (16 bytes) απαιτούνται 8 κυκλώματα precharge. Η οργάνωση της μνήμης είναι 32 λέξεις των 8-bits (1byte), χωρισμένες σε δυο δεκαεξάδες.



Εικόνα 3-23 Αρχιτεκτονική δυναμικής μνήμης

Η αρχιτεκτονική χρησιμοποιεί μια κοινή αρτηρία 8-bit (data_in bus) για εγγραφή δεδομένων σε όλες τις λέξεις της μνήμης και δύο αρτηρίες των 8-bits (data_out buses) για ανάγνωση δεδομένων. Οι δύο αρτηρίες επιτρέπουν την ταυτόχρονη ανάγνωση δύο 8-

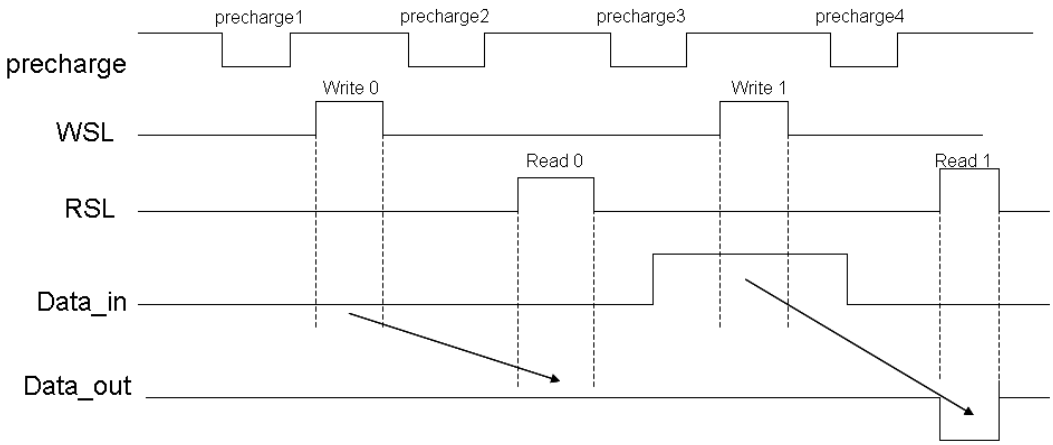


bit λέξεων που βρίσκονται στην ίδια γραμμή. Για την οργάνωση της απαιτούνται δύο ξεχωριστά κυκλώματα precharge για κάθε αρτηρία ανάγνωσης.

Να σημειωθεί ότι πριν γίνει οποιαδήποτε πρόσβαση, η μνήμη πρέπει να γίνει precharged. Τα σήματα που ελέγχουν το κύκλωμα μνήμης είναι: το precharge, το Write Select Line (WSL) και το Read Select Line (RSL). Τα δύο τελευταία ελέγχουν την εγγραφή και την ανάγνωση της μνήμης, ενώ το πρώτο την επαναφόρτιση των γραμμών εισόδου και εξόδου (data_in , data_out). Τα διαγράμματα χρονισμού για την πρόσβαση της μνήμης περιγράφονται την παρακάτω ενότητα.

3.6.1.2 Διάγραμμα Χρονισμού Πρόσβασης

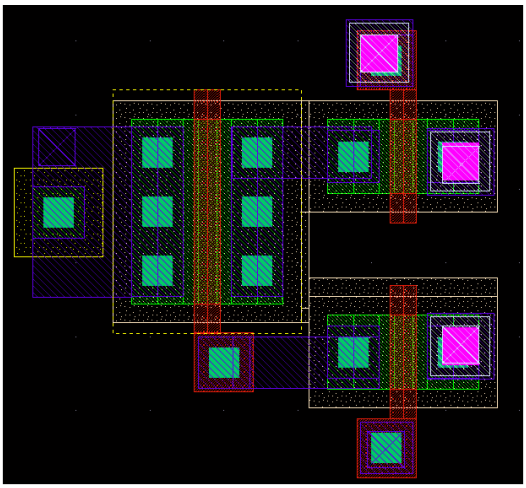
Στην επόμενη εικόνα παρουσιάζεται ο χρονισμός των σημάτων που ελέγχουν το κύκλωμα δυναμικής μνήμης.. Να σημειωθεί ότι όταν το σήμα precharge είναι ενεργό δεν μπορούμε να κάνουμε εγγραφή ή ανάγνωση από τη μνήμη. Επίσης η ανάγνωση είναι αντίστροφης λογικής από την εγγραφή. Δηλαδή, όταν στην μνήμη εγγράφουμε λογικό 1, αυτό που θα διαβάσουμε είναι το λογικό 0 και αντίστροφα.



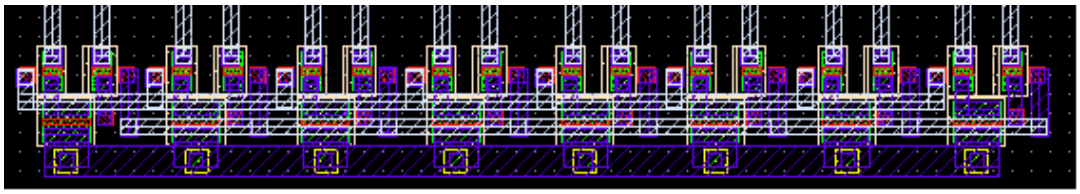
Διάγραμμα χρονισμού πρόσβασης στη δυναμική μνήμη

3.6.1.3 Φυσικός σχεδιασμός

Στις παρακάτω εικόνες φαίνεται ο φυσικός σχεδιασμός της δυναμικής μνήμης.



Εικόνα 3-24 Layout κυκλώματος δυναμικής μνήμης 1 bit



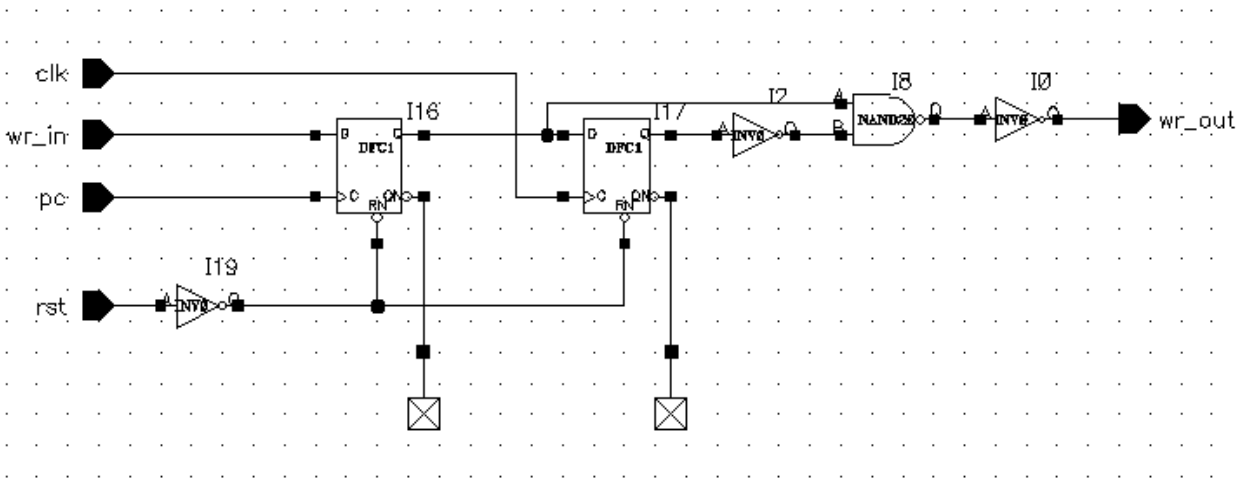
Εικόνα 3-25 Layout κυκλώματος δυναμικής μνήμης 8-bits

3.6.2 Κύκλωμα Εγγραφής στην DRAM

3.6.2.1 Αρχιτεκτονική

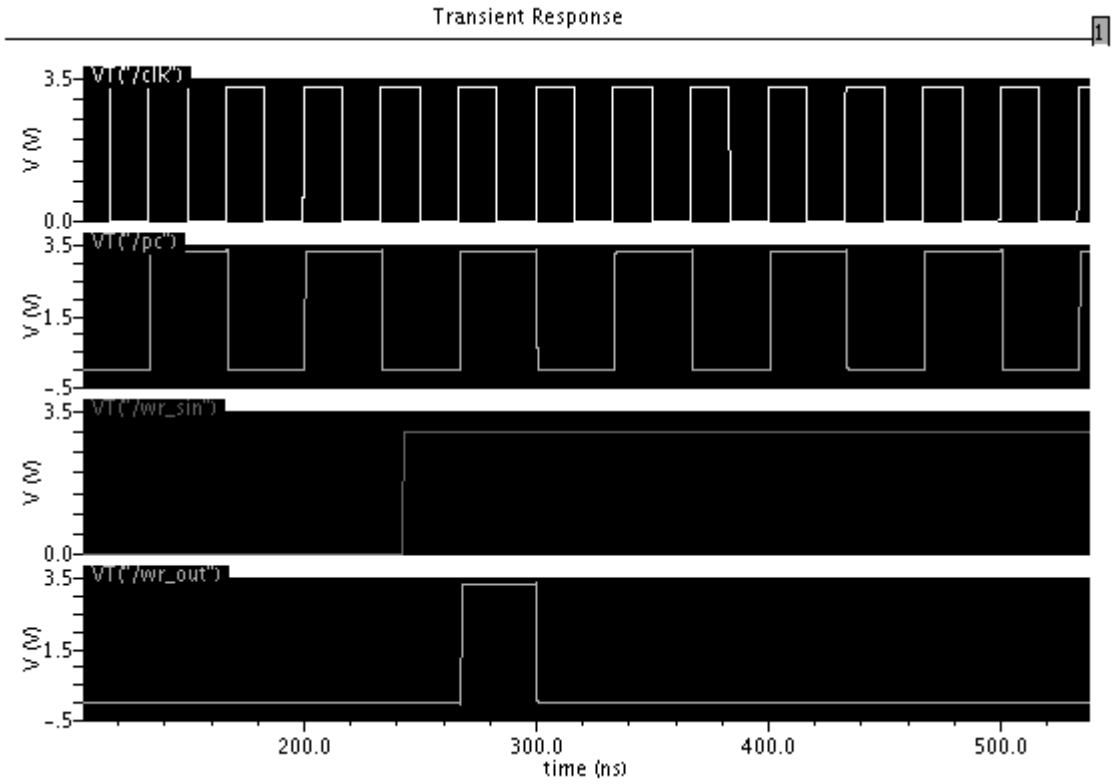
Η κύρια λειτουργία του κυκλώματος εγγραφής είναι η λήψη ενός ασύγχρονου σήματος, που στέλνει ο συγκριτής και ενημερώνει ότι η στάθμη/τιμή της ράμπας είναι ίδια με τη στάθμη/τιμή του αναλογικού σήματος εισόδου, η παραγωγή ενός παλμού διάρκειας ενός κύκλου εγγραφής στη μνήμη και ο συγχρονισμός αυτού του παλμού ώστε να έπεται από μια περίοδο precharge της μνήμης. Ο παλμός που παράγει το κύκλωμα αυτό οδηγεί τη γραμμή ελέγχου εγγραφής μνήμης της αντίστοιχης λέξης (Write Select Line – WSL). Αυτό σημαίνει ότι το κύκλωμα εγγραφής μνήμης είναι ενσωματωμένο σε κάθε κελί μνήμης των 8-bits. Το

κύκλωμα αποτελείται από δύο D-Flip Flops και συνδυαστική λογική. Το σήμα wr_in είναι το ασύγχρονο σήμα εισόδου, το pc είναι το σήμα precharge της μνήμης, το clk είναι το ρολόι του κυκλώματος (σύγχρονο κύκλωμα) και το wr_out είναι το συγχρονισμένο σήμα εξόδου που οδηγεί το WSL.



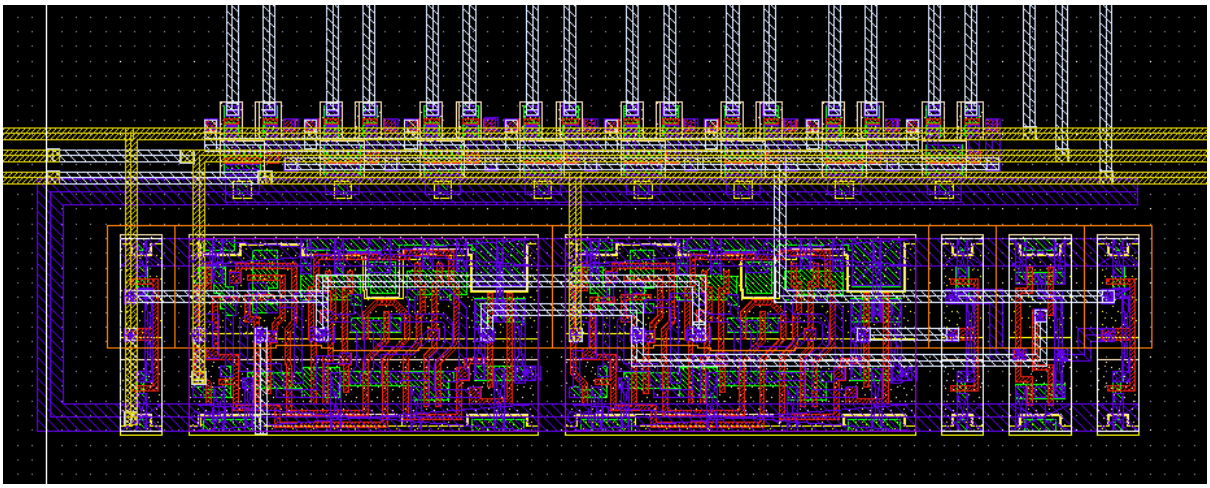
Εικόνα 3-26 Αρχιτεκτονική του σύγχρονου κυκλώματος εγγραφής στη μνήμη.

Διάγραμμα χρονισμού



Εικόνα 3-27 Διάγραμμα χρονισμού του ανωτέρω κυκλώματος

3.6.2.2 Φυσικός σχεδιασμός



Εικόνα 3-28 Layout ψηφιακού κυκλώματος εγγραφής με το αντίστοιχο κελί μνήμης των 8-bits

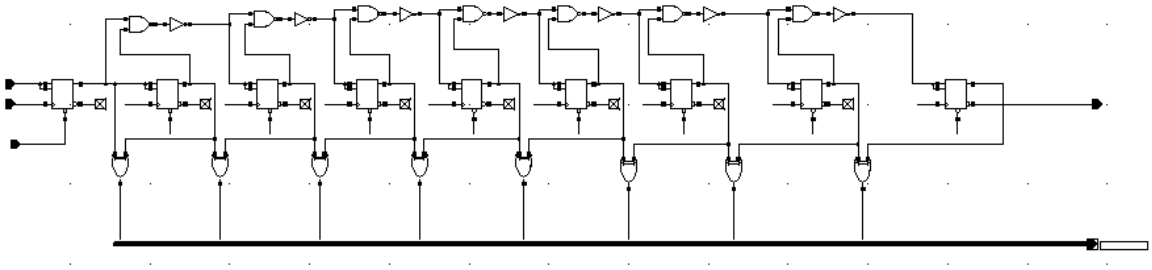
3.6.3 Κύκλωμα Μετατροπής της αναλογικής τάσης σε ψηφιακή

Η μετατροπή του αναλογικού σήματος σε ψηφιακό κάνει χρήση ενός ψηφιακού απαριθμητή και συγκεκριμένα, ενός Gray Code Counter (GCC). Το κύριο χαρακτηριστικό του GCC

απαριθμητή είναι ότι οι διαδοχικές του τιμές διαφέρουν μόνο σε ένα bit. Περισσότερα για την αρχιτεκτονική υλοποίησης, τον χρονισμό του GCC παρουσιάζονται παρακάτω.

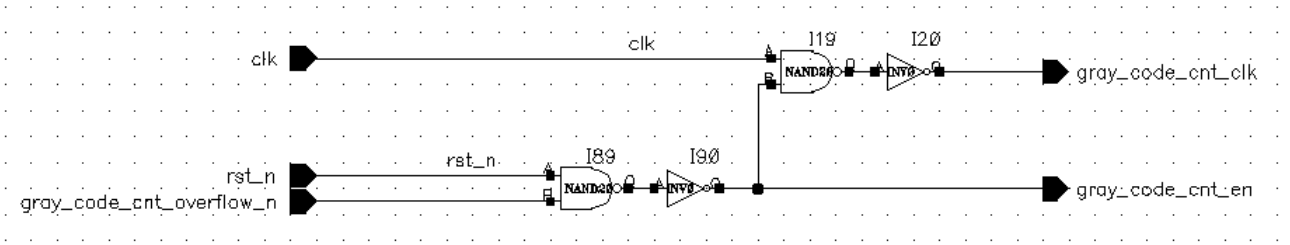
3.6.3.1 Gray Code Counter (GCC)

Η αρχιτεκτονική του GCC παρουσιάζεται στην Εικόνα 3-29 Αρχιτεκτονική του Gray Code Counter. Αποτελείται από 9 JK Flip-Flops και συνδυαστική λογική.



Εικόνα 3-29 Αρχιτεκτονική του Gray Code Counter

Το κύκλωμα ελέγχου του GCC αποτελείται από συνδυαστική λογική, εχει τρεις εισόδους: το ρολόι, το ασύγχρονο σήμα reset και το σήμα υπερχείλισης του GCC. Έχει δύο εξόδους: το σήμα ρολογιού που οδηγεί τον απαριθμητή και το σήμα gray_code_cnt_en που υποδεικνύει το χρονικό διάστημα που ο GCC είναι ενεργός. Το τελευταίο σήμα σηματοδοτεί το τέλος της περιόδου μετατροπής A/D και εγγραφής στη μνήμη και ενεργοποίησης του κυκλώματος ανάγνωσης μνήμης.

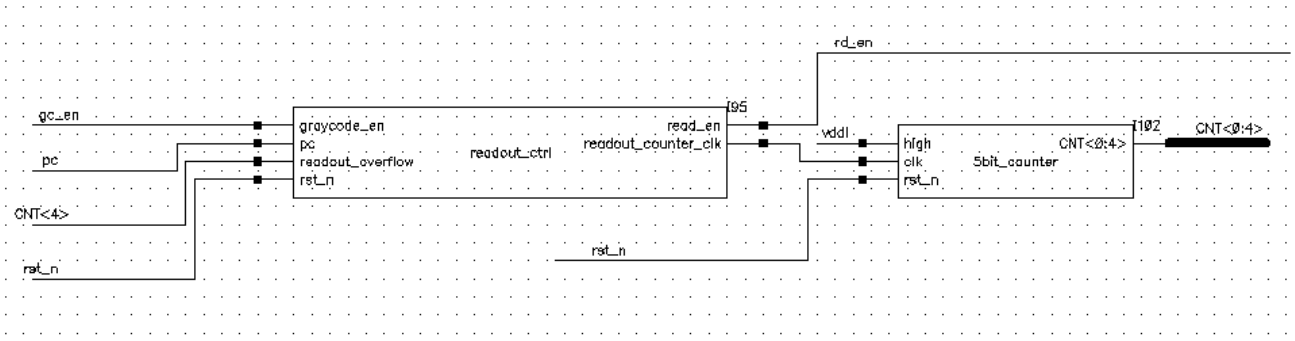


Εικόνα 3-30

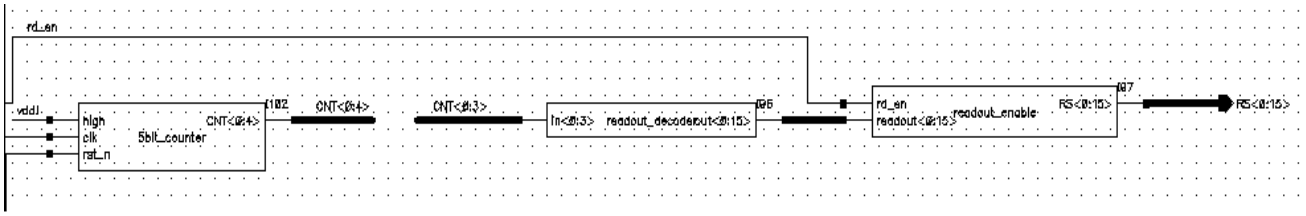
3.6.4 Κύκλωμα ανάγνωσης από τη μνήμη DRAM (readout)

3.6.4.1 Αρχιτεκτονική

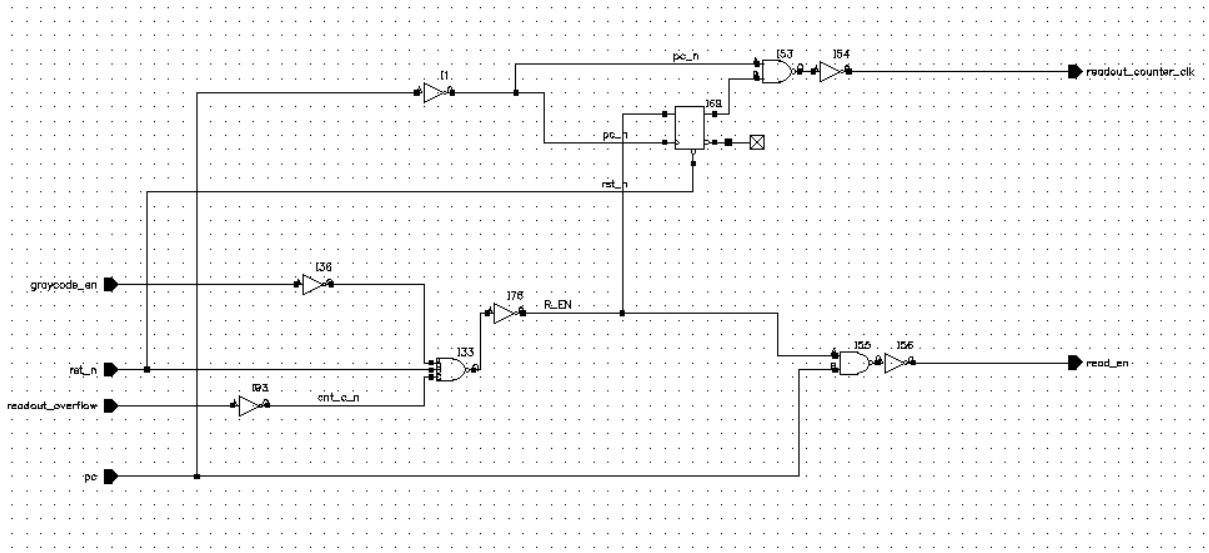
Η αρχιτεκτονική του κυκλώματος ανάγνωσης δεδομένων από τη μνήμη αποτελείται από τέσσερις βαθμίδες: τη βαθμίδα ελέγχου, έναν απαριθμητή 4-bit, έναν αποκωδικοποιητή 4x16, και ένα κύκλωμα συγχρονισμού των σημάτων εξόδου. Η διαδικασία του «readout» ξεκινά μόλις τελειώσει η μετατροπή A/D και η φάση εγγραφής στη μνήμη. Η πρώτη βαθμίδα υλοποιεί όλη τη λογική ελέγχου για το readout. Ο απαριθμητής παράγει 16 διαδοχικές ψηφιακές λέξεις των 4-bits από «0000» έως «1111». Η τρίτη βαθμίδα μετατρέπει τις διαδοχικές 4-bit λέξεις σε 16 διαφορετικά σήματα, που σε κάθε χρονική στιγμή μόνο ένα σήμα από τα 16 παίρνει λογική τιμή 1. Τα 16 αυτά σήματα είναι τα σήματα που οδηγούν το RSL των 16 διαφορετικών λέξεων της μνήμης, όπου μόνο μια λέξη πρέπει να διαβάζεται κάθε φορά. Επειδή όμως τα σήματα RSL πρέπει να έπονται από περιόδους precharge της μνήμης, η τελευταία βαθμίδα συγχρονίζει τα 16 σήματα με το σήμα precharge της μνήμης. Τα σχηματικά διαγράμματα του κυκλώματος ανάγνωσης μνήμης παρουσιάζονται στις παρακάτω εικόνες.



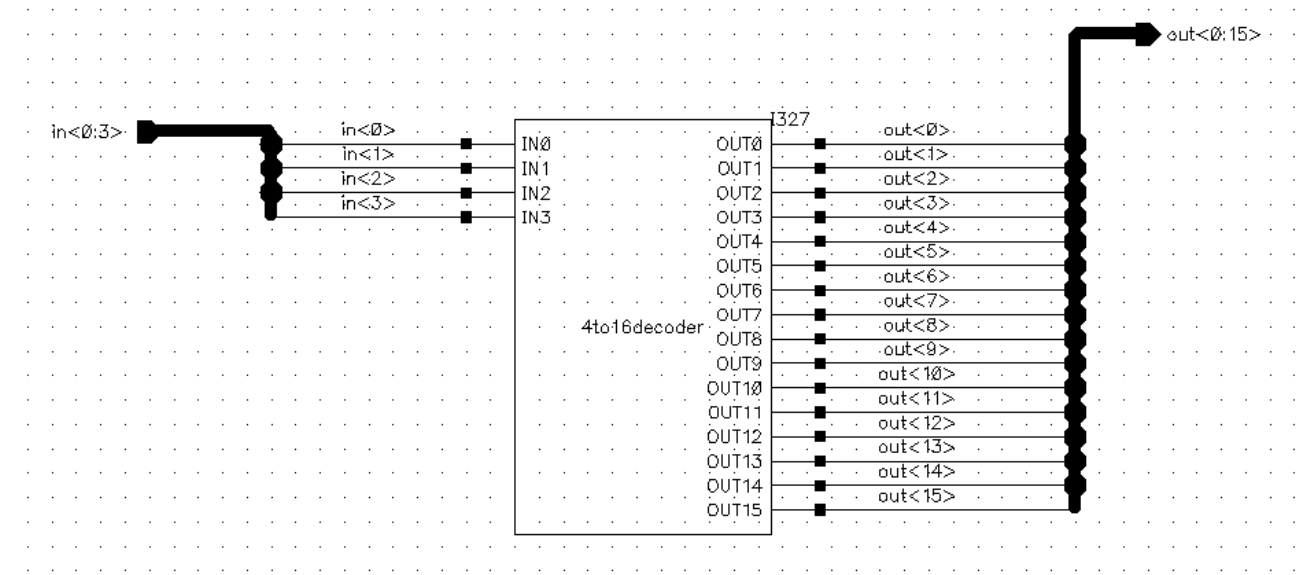
Εικόνα 3-31 Αρχιτεκτονική readout (1η και 2η βαθμίδα)



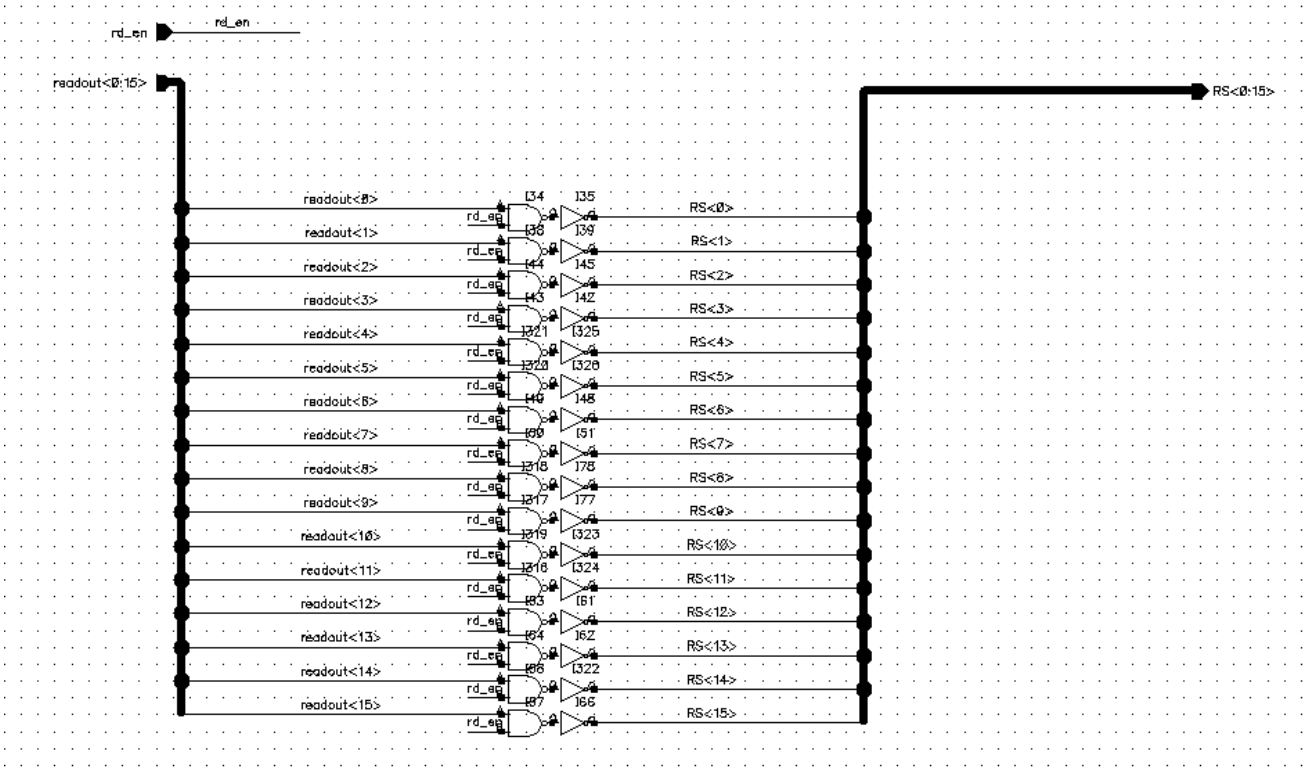
Εικόνα 3-32 Αρχιτεκτονική readout (2η, 3η και 4η βαθμίδα)



Εικόνα 3-33 Περιγραφή 1ης βαθμίδας (ελέγχου)



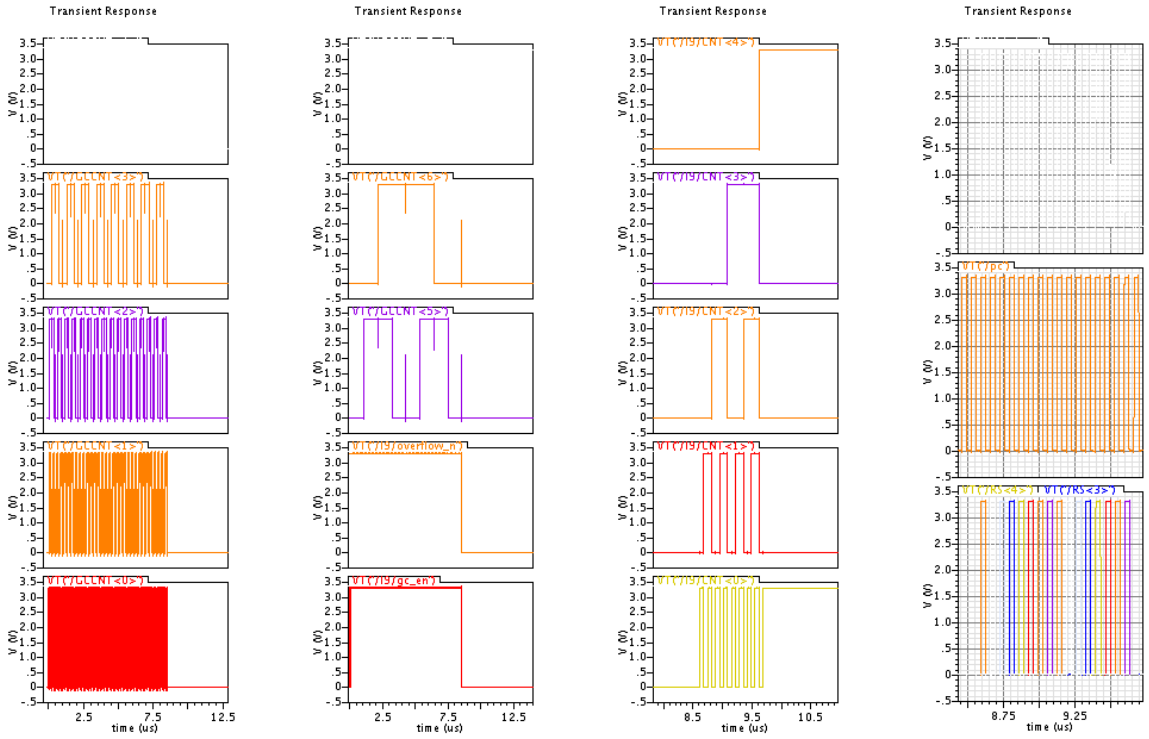
Εικόνα 3-34 Περιγραφή 3ης βαθμίδας (αποκωδικοποιητή)



Εικόνα 3-35 Περιγραφή 4ης βαθμίδας (κυκλώματος συγχρονισμού)

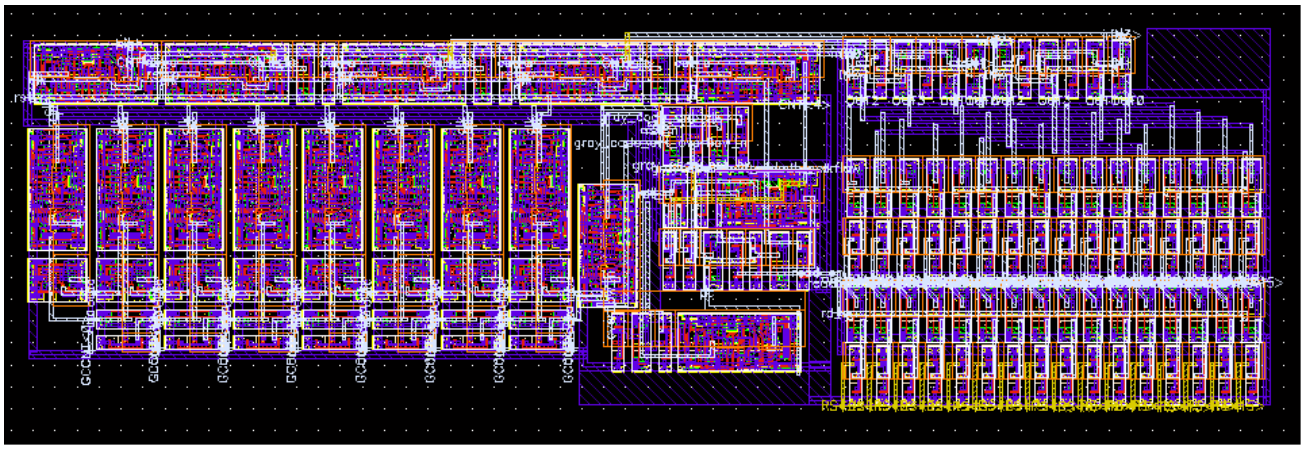
3.6.5 Διάγραμμα χρονισμού GCC και Readout

Το διάγραμμα χρονισμού που παρουσιάζεται στην Εικόνα 3-36 Διάγραμμα χρονισμού GCC & readout περιγράφει τη χρονική ακολουθία των σημάτων του απαριθμητή GCC και των σημάτων που ελέγχουν την ανάγνωσης της μνήμης.



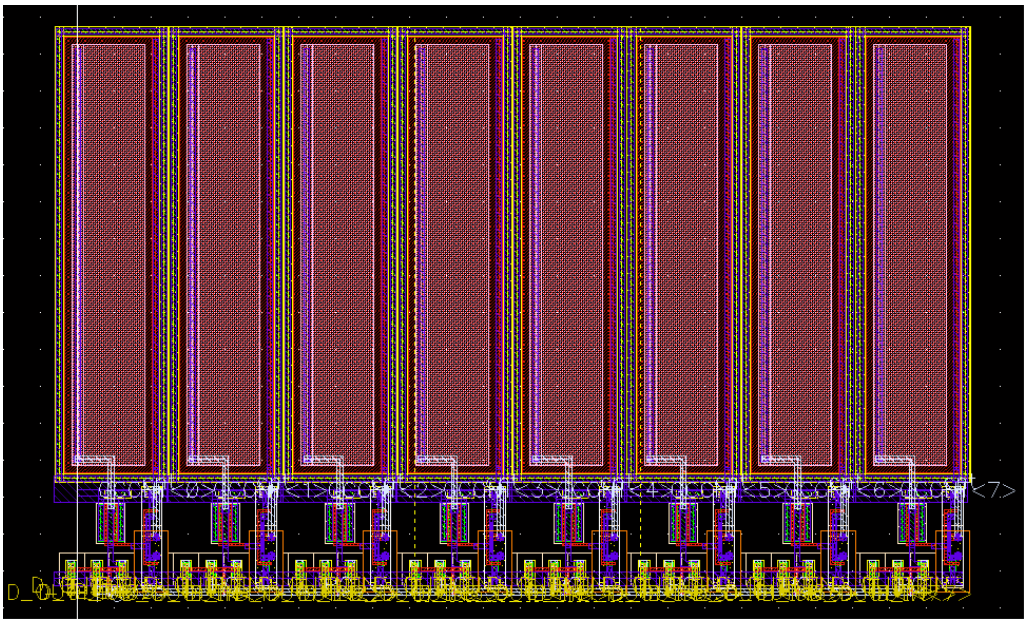
Εικόνα 3-36 Διάγραμμα χρονισμού GCC & readout

3.6.6 Φυσικός σχεδιασμός

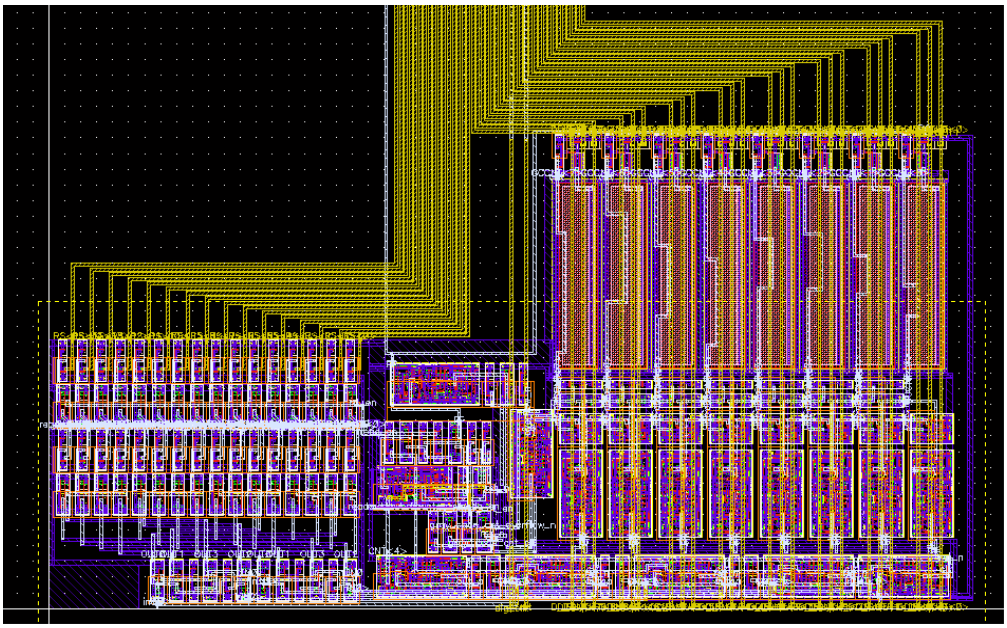


Εικόνα 3-37 Φυσικός σχεδιασμός των κυκλωμάτων GCC και readout

3.7 Φυσικός σχεδιασμός του συνολικού ψηφιακού τμήματος



Εικόνα 3-38 Precharge layout



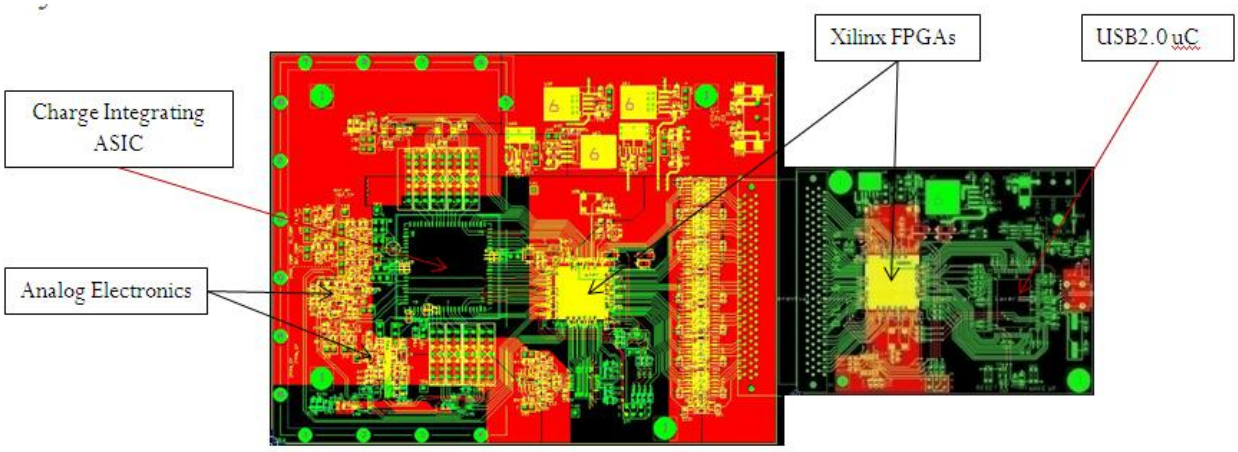
Εικόνα 3-39 Layout Συνολικού Ψηφιακού κυκλώματος

4 Σχεδίαση Ηλεκτρονικών Ελέγχου

Η σχεδίαση του συστήματος ελέγχου και λήψης δεδομένων, που είναι το κύριο μέρος της παρούσας εργασίας, χωρίζεται σε τρία διαφορετικά κυκλώματα σε επίπεδο πλακέτας (PCBs).

1. Πηγές Ρεύματος (Π1)
2. Πλακέτα μεικτών ηλεκτρονικών (αναλογικά και ψηφιακά ηλεκτρονικά) (Π2)
3. Πλακέτα ψηφιακών ηλεκτρονικών για την λήψη των δεδομένων σε ηλεκτρονικό υπολογιστή (Π3)

Στην Εικόνα 4-1 Σύστημα λήψης δεδομένων και ελέγχου του ολοκληρωμένουφαίνεται ο συνολικός φυσικός σχεδιασμός του συστήματος ελέγχου, όπου αριστερά βρίσκεται η πλακέτα Π2 και δεξιά η πλακέτα Π3, ενώ οι πηγές ρεύματος Π1 τοποθετούνται κάθετα στο επίπεδο των άλλων δύο με κατάλληλους ακροδέκτες, στο πάνω και κάτω μέρος του ASIC.

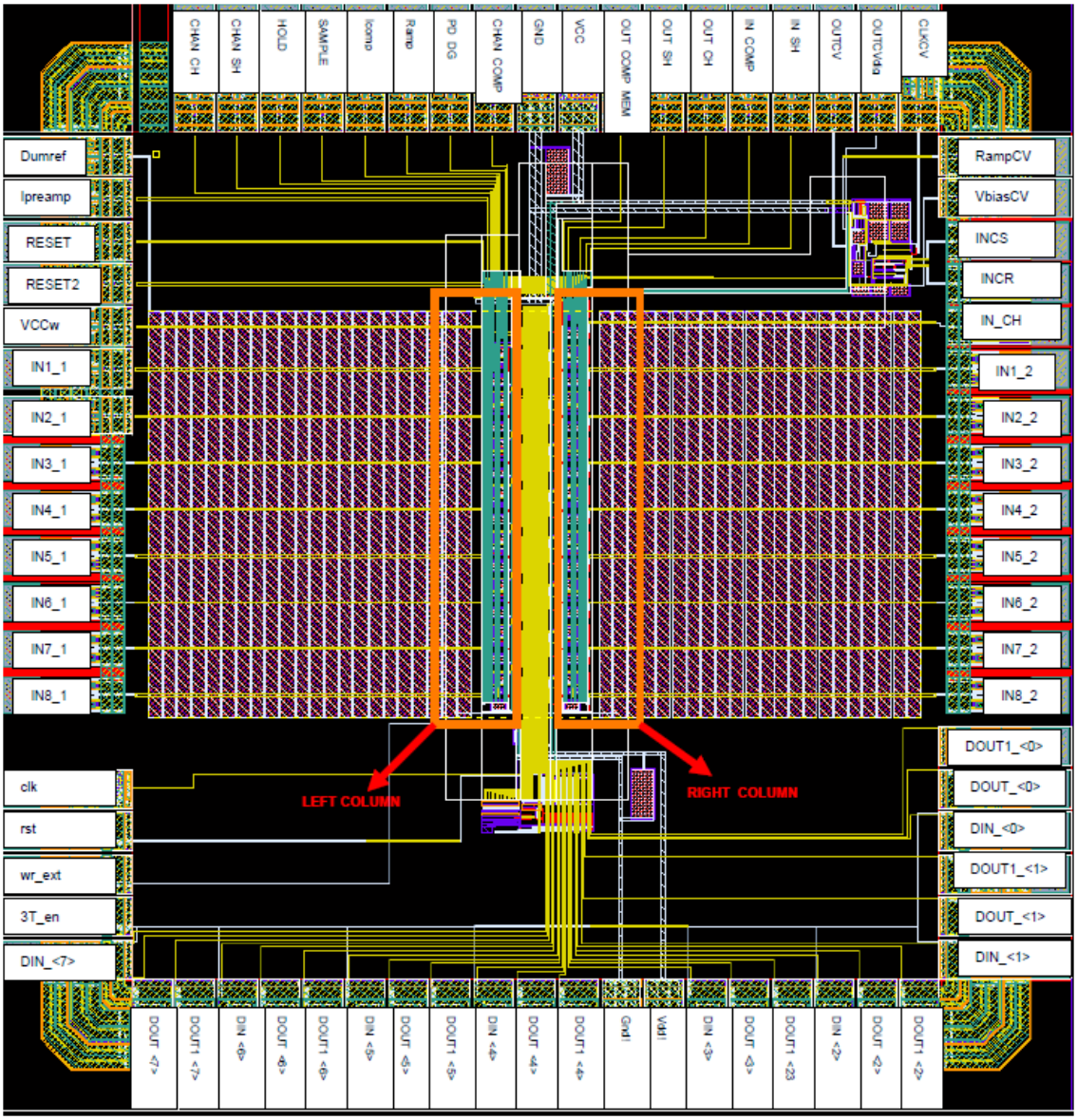


Εικόνα 4-1 Σύστημα λήψης δεδομένων και ελέγχου του ολοκληρωμένου

Τα κύρια συστατικά μέρη της πλακέτας Π2 είναι το ολοκληρωμένο προς έλεγχο, τα περιφερειακά ηλεκτρονικά (στα αριστερά του ASIC), ένα FPGA [Field Programmable Gate Array] (το οποίο θα σχολιαστεί ξεχωριστά σε επόμενο κεφάλαιο), μετατροπείς ψηφιακού σήματος σε αναλογικό, τροφοδοσίες (επάνω δεξιά) και απομονωτές (isolators).

Η πλακέτα Π3 αποτελείται από ένα FPGA και έναν USB 2.0 μικροελεγκτή (Universal Serial Bus), τροφοδοσίες και τα απαραίτητα παθητικά στοιχεία.

Η επιλογή των συγκεκριμένων ενσωματωμένων κυκλωμάτων δίνει την δυνατότητα για έλεγχο του ολοκληρωμένου αλλά και μεταφορά των δεδομένων προς τον ηλεκτρονικό υπολογιστή με υψηλή ταχύτητα



Εικόνα 4-2 Συνολικός φυσικός σχεδιασμός του ολοκληρωμένου και οι εξωτερικές του διασυνδέσεις

APIC ASIC packaged JLCC84

75	Not Connected	12	Not Connected
76	Dumref con to dummy metal	13	D_OUT<7> left col out bit 7
77	Ipreamp pixels CTIAs' bias current	14	D_OUT1<7> right col out bit 7
78	RESET CTIA Reset	15	D_IN<6> 3state out gcnt bit6
79	RESET2 CTIA Reset	16	D_OUT<6> left col out bit 6
80	VCCw 3.3V low noise for wells	17	D_OUT1<6> right col out bit 6
81	IN1_1 test channel input	18	D_IN<5> 3state out gcnt bit5
82	IN2_1 left column analog IN2**	19	D_OUT<5> left col out bit 5
83	IN3_1 left column analog IN3	20	D_OUT1<5> right col out bit 5
84	IN4_1 left column analog IN4	21	D_IN<4> 3state out gcnt bit4
1	IN5_1 left column analog IN5	22	D_OUT<4> left col out bit 4
2	IN6_1 left column analog IN6	23	D_OUT1<4> right col out bit 4
3	IN7_1 left column analog IN7	24	gnd! Digital ground
4	IN8_1 left column analog IN8	25	vdd! 3.3V digital
5	clk digital clock	26	D_IN<3> 3state out gcnt bit3
6	rst digital reset active HIGH	27	D_OUT<3> left col out bit 3
7	wr_ext DRAM wr_enable at RS9	28	D_OUT1<3> right col out bit 3
8	3T_en enable 3stat out D_IN<0:7>	29	D_IN<2> 3state out gcnt bit2
9	D_IN<7> 3state out gcnt bit7	30	D_OUT<2> left col out bit 2
10	Not Connected	31	D_OUT1<2> right col out bit 2
11	Not Connected	32	Not Connected

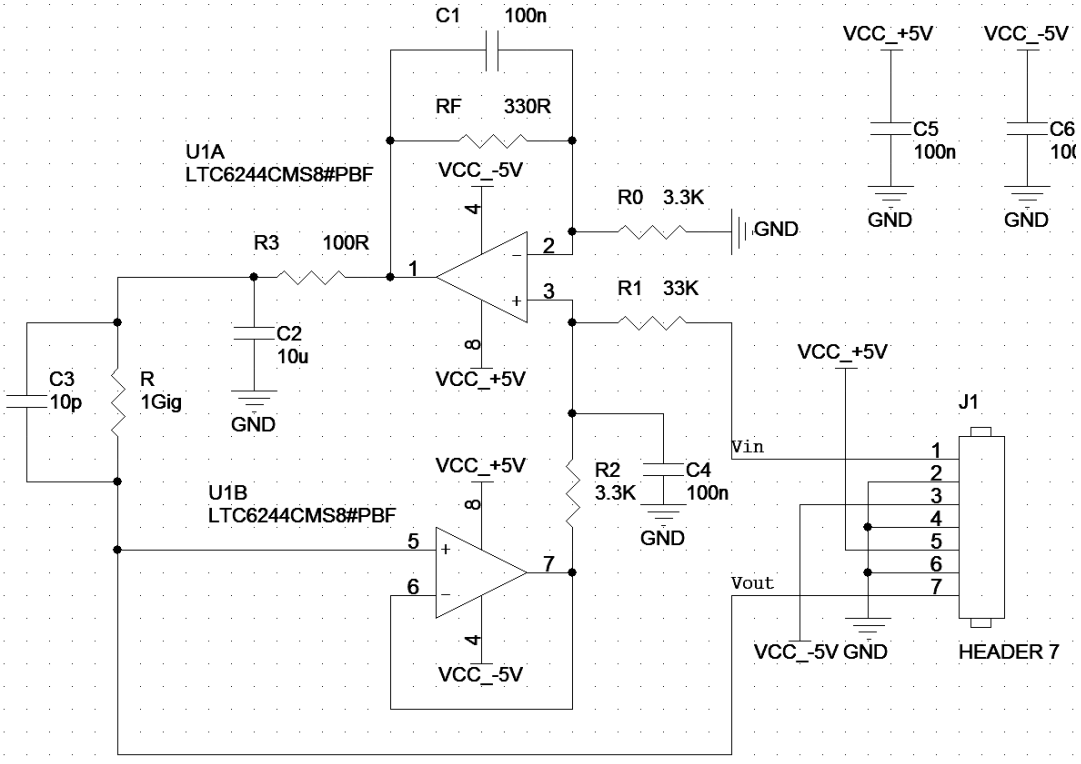
33	Not Connected	54	Not Connected
34	D_IN<1> 3state out gcnt bit1	55	Not Connected
35	D_OUT<1> left col out bit 1	56	clkCV
36	D_OUT1<1> right col out bit 1	57	OUTCVdig
37	D_IN<0> 3state out gcnt bit0	58	OUTCV
38	D_OUT<0> left col out bit 0	59	IN_SH input Sample and Hold
39	D_OUT1<0> right col out bit 0	60	IN_COMP input Comparator
40	IN8_2 right column analog IN8**	61	OUT_CH output CTIA
41	IN7_2 right column analog IN7	62	OUT_SH output Sample and Hold
42	IN6_2 right column analog IN6	63	OUT_COMP_MEM Output Comparator
43	IN5_2 right column analog IN5	64	VCC 3.3V Analog
44	IN4_2 right column analog IN4	65	GND Analog ground
45	IN3_2 right column analog IN3	66	Chan_COMP test channel Comparator out
46	IN2_2 right column analog IN2	67	PD_DG Comparators Pwr On Active High
47	IN1_2 right column analog IN1	68	Ramp Analog input RAMP Voltage
48	IN_CH input CTIA	69	Icomp Pixels Comparators bias current
49	INCR	70	SAMPLE Active High during integration
50	INCS	71	HOLD Active High during A/D Conversion
51	VbiasCV	72	Chan_SH test channel Sample&Hold out
52	RampCV	73	Chan_CH test channel CTIA out
53	Not Connected	74	GND_lid

Ipreamp	Output from the ASIC 8uA	INX_Y	Input to ASIC (0-200pA)		
Icomp	Input to the ASIC 4uA	VbiasCV			

Εικόνα 4-3

4.1 Πηγές Ρεύματος

Όπως αναφέρθηκε στην περιγραφή του ολοκληρωμένου κυκλώματος προς έλεγχο, το συγκεκριμένο ολοκληρωμένο σχεδιάστηκε με σκοπό την μελέτη pixel ηλεκτρονικών ανάγνωσης για ανιχνευτές ακτινοβολίας δύο διαστάσεων. Το σήμα εισόδου λοιπόν δεν θα προέρχεται από τον ανιχνευτή, εξαιτίας της πρόσπτωσης φωτονίων στον όγκο του, αλλά από πηγές ρεύματος οι οποίες θα είναι σε θέση να προσομοιώσουν (Emulate) το εύρος ρευμάτων που θα μας παρείχε ο ανιχνευτής με στόχο τον έλεγχο του ολοκληρωμένου. Επομένως επειδή, όπως φαίνεται και στην εικόνα που περιέχει τις εξωτερικές διασυνδέσεις του ASIC (Application Specific Integrated Circuit) το συγκεκριμένο κύκλωμα σχεδιάστηκε ως ξεχωριστή πλακέτα καθώς έπρεπε να υλοποιηθεί δεκαέξι φορές, μια για κάθε κανάλι εισόδου. Στο παρακάτω σχήμα παρουσιάζεται η τοπολογία που επιλέχθηκε για την υλοποίηση των εν λόγω πηγών ρεύματος.



Εικόνα 4-4 Κύκλωμα πηγής ρεύματος

Η V_{out} είναι η τάση ελέγχου της πηγής ρεύματος και I είναι το ρεύμα στο φορτίο. Το ρεύμα που παρέχεται στο φορτίο από την πηγή είναι ανεξάρτητο της αντίστασης του φορτίου καθώς όσο ρεύμα διέρχεται από την αντίσταση R, διέρχεται και από αυτό. Θεωρώντας ιδανικούς τελεστικούς ενισχυτές η V_{out} εξαρτάται από τις V_{in} και την V_{out}^{U1B} σύμφωνα με την σχέση:

$$V_{out} = \frac{R_0 + R_F}{R_0} \frac{R_1 R_2}{R_1 + R_2} \left(\frac{V_{in}}{R_1} + \frac{V_{out}^{U1B}}{R_2} \right) \quad \textbf{(1.8)}$$

Θέτοντας $R_1 = 10R_2$, $R_0 = R_1 // R_2 = \frac{10}{11} R_2$ και $R_F = \frac{R_2}{11}$, έχουμε:

$$V_{out} = \frac{V_{in}}{10} + V_1 \quad \textbf{(1.9)}$$

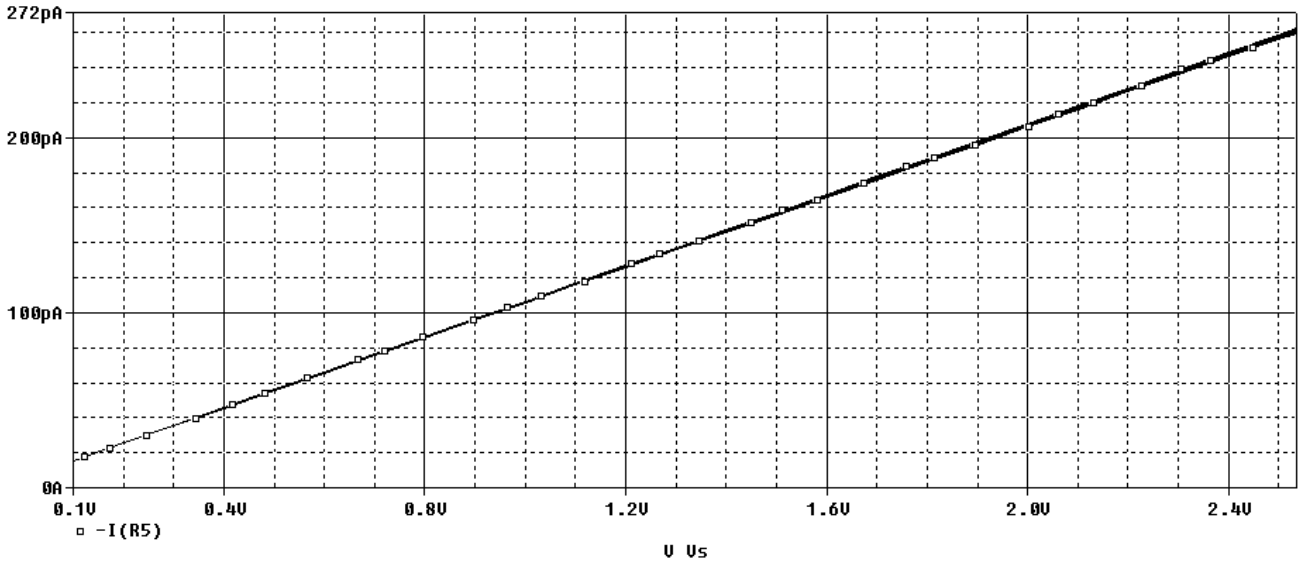
Η τάση ελέγχου V_{in} συνεισφέρει στην V_{out} με $V_{in}/10$ και έτσι είναι δυνατόν να επιτευχθούν μικρά ρεύματα της τάξης των 100 pA με 1V τάση ελέγχου. Επομένως επιτυγχάνεται εύκολα το επιδιωκόμενο εύρος ρευμάτων, που είναι $1 \sim 200\,pA$ με χρήση ενός μετατροπέα ψηφιακού σήματος σε αναλογικό.

Σε συνθήκες σταθερής κατάστασης το ρεύμα που διαρρέει την αντίσταση , διαρρέει και το φορτίο. Έχουμε:

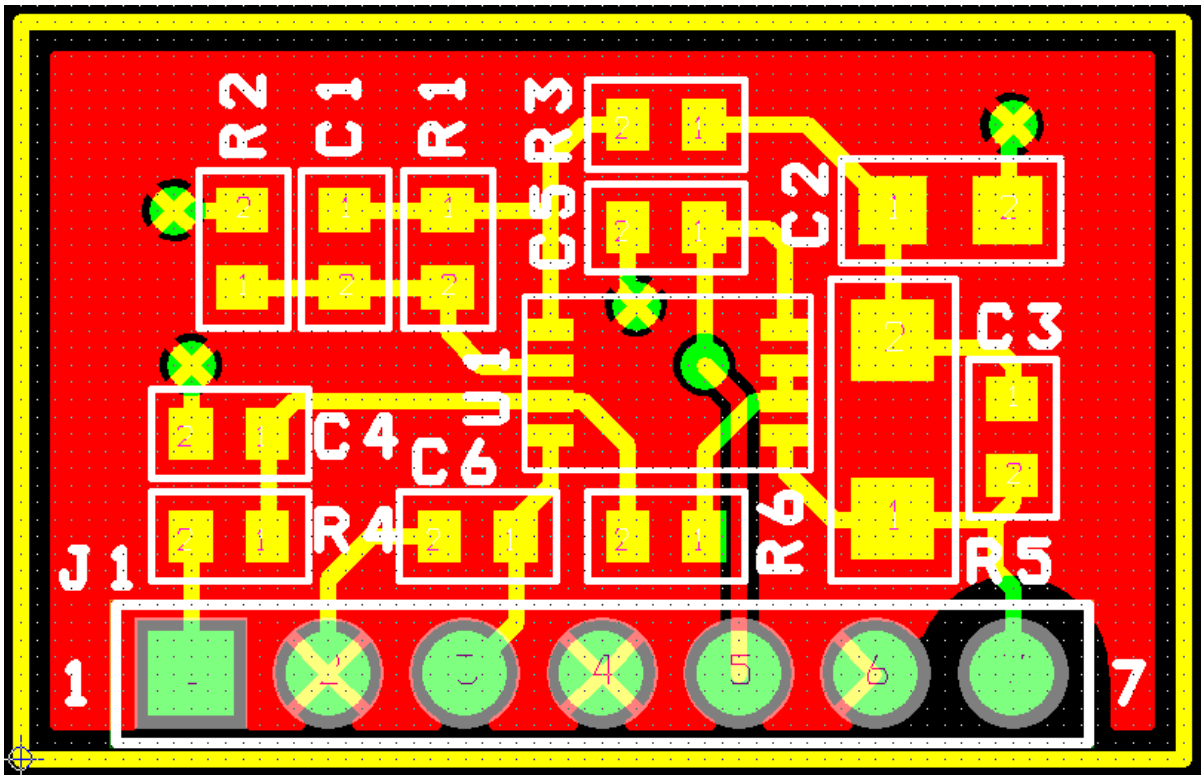
$$\frac{V_{load} - V_{out}}{R} = -\frac{V}{R_{load}} \quad \textbf{(1.10)}$$

Είναι $V_{out}^{UB1} = \frac{R_{load}}{R} \frac{V_{in}}{10} \Rightarrow I = \frac{V_{in}/10}{R}$ και επομένως το ρεύμα που παρέχεται από την πηγή είναι ανεξάρτητο από την αντίσταση του φορτίου.

Για την υλοποίηση του κυκλώματος χρησιμοποιήθηκε ο τελεστικός ενισχυτής LTC6244 της εταιρίας Linear Technology που περιέχει δύο τελεστικούς ενισχυτές σε κάθε εξάρτημα εξαιτίας κυρίως του χαμηλού ρεύματος διαρροής στην εισόδου του (input bias current ~ 1pA) έτσι ώστε να είναι χαμηλότερο από το εύρος των ρευμάτων που θέλαμε να παράγει η ελεγχόμενη πηγή ρεύματος. Για το κύκλωμα αυτό έγινε προσομοίωση με το λογισμικό SPICE με την χρήση του μοντέλου που παρείχε ο κατασκευαστής. Στο παρακάτω διάγραμμα φαίνεται το ρεύμα που παρέχει η πηγή για τάση ελέγχου από μηδέν έως 2.5 $Volt$ και ταυτόχρονα παραμετρική ανάλυση για διαφορετικές τιμές της R_{load} από μηδέν έως 1 $G\Omega$. Κανονικά θα έπρεπε να δούμε 20 διαφορετικές καμπύλες I/V αλλά επειδή έχουμε πηγή ρεύματος είναι όλες πολύ κοντά μεταξύ τους. Παρατηρούμε ότι διακρίνονται οι καμπύλες όσο μεγαλώνει η τάση έλεγχου και γι' αυτό η χαρακτηριστική δεν έχει το ίδιο πάχος όσο μεταβάλλεται η τάση.



Εικόνα 4-5 Χαρακτηριστική I/V της πηγής ρεύματος



Εικόνα 4-6 Φυσικός σχεδιασμός της πηγής ρεύματος

4.2 Πλακέτα Ελέγχου – Μεικτού σήματος Π2

4.2.1 Περιφερειακά αναλογικά ηλεκτρονικά

Για να μπορέσουν να «διαβασθούν» με την βοήθεια ενός παλμογράφου οι διάφορες αναλογικές έξοδοι του ολοκληρωμένου, είτε για να δώσουμε κάποια είσοδο στις εισόδους του ολοκληρωμένου, οι οποίες αναφέρονται στις δομές TEST_STR_1 και PIXEL_TEST υπάρχει η ανάγκη να τοποθετηθεί ένας ενισχυτής μοναδιαίου κέρδους στην κάρτα ελέγχου ώστε να μπορεί να οδηγηθεί η γραμμή μεταφοράς στην είσοδο του παλμογράφου είτε για να οδηγηθούν οι είσοδοι. Ένας επιπλέον λόγος για τον οποίο πρέπει να γίνει αυτό είναι ότι το ολοκληρωμένο δεν διαθέτει εσωτερικά μοναδιαίους ενισχυτές στην έξοδο ή είσοδο των σημάτων. Ο ενισχυτής που επιλέχθηκε είναι ο LMH6559 της Texas Instruments.

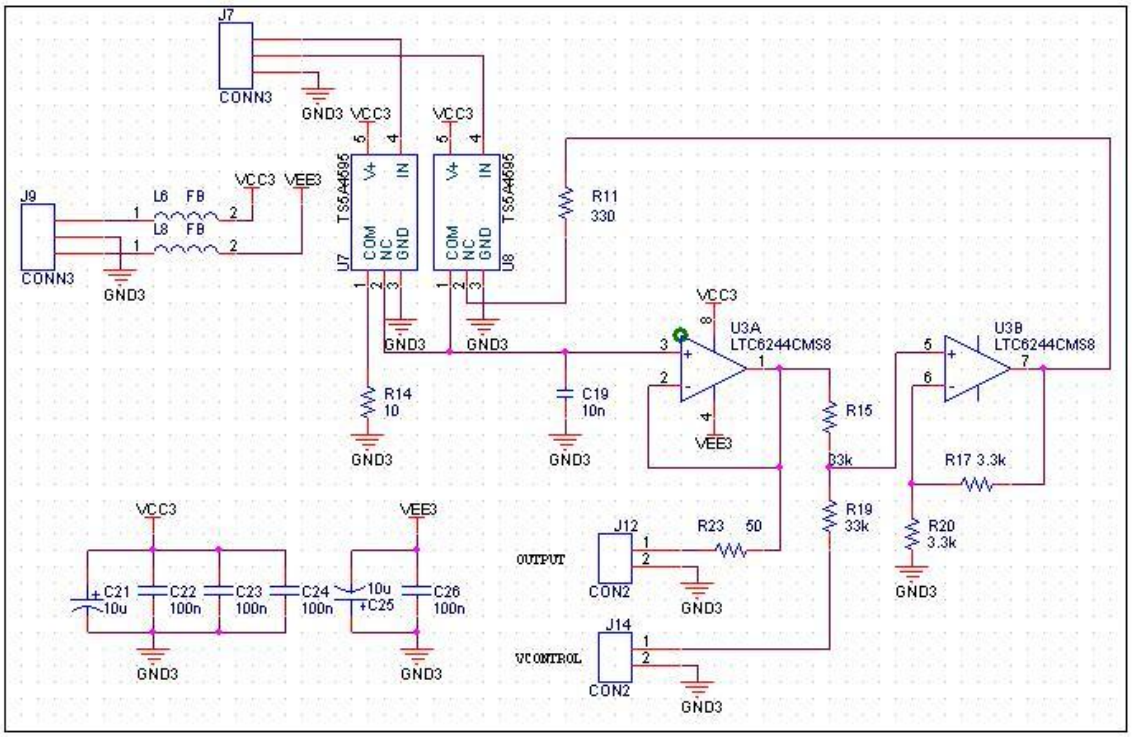
Ο ενισχυτής είναι ρυθμισμένος εσωτερικά ώστε να έχει μοναδιαίο κέρδος, επιπλέον έχει αντίσταση εισόδου 200kΩ και αντίσταση εξόδου 1.2Ω. Στην πραγματικότητα χρησιμοποιείται κυρίως για μοναδιαία ενίσχυση (buffering) σημάτων εικόνας και την ασφαλή χωρίς απώλειες μεταφορά τους. Έτσι το εύρος ζώνης μικρού σήματος του, που είναι 1750MHz είναι πολύ παραπάνω από τις απαιτήσεις των σημάτων που θέλουμε να ενισχύσουμε αλλά εξαιτίας του μικρού του κόστους και επειδή θέλουμε να μετρήσουμε ένα πρωτότυπο ολοκληρωμένο καθίσταται ιδανική επιλογή για την συγκεκριμένη εφαρμογή. Επιπλέον η εσωτερική του ρύθμιση για μοναδιαίο ακριβές κέρδος επιτρέπει την μείωση περαιτέρω του κόστους καθώς δεν απαιτεί επιπλέον παθητικά στοιχεία για την μοναδιαία ρύθμιση του, όπως θα απαιτούσε ένας τελεστικός ενισχυτής και επιτρέπει την μείωση της πολυπλοκότητας ενός ήδη πολύπλοκου συστήματος λήψης δεδομένων. Στον παρακάτω πίνακα φαίνονται τα σήματα για τα οποία χρησιμοποιείται ο εν λόγω ενισχυτής.

IC PIN	IC Block	Description
IN_SH	Test structure 1 – S/H	Sample and Hold Input
OUT_CH	Test structure 1 – Charge Amplifier	Charge Amplifier Output
IN_COMP	Test structure 1 – Comparator	Comparator Input
OUT_COMP_MEM	Test structure 1 – Comparator	Comparator Output
OUT_SH	Test structure 1 – S/H	Sample and Hold Output
CHAN_COMP	Test channel	Pixel 1 Comparator Output

CHAN_SH	Test channel	Pixel 1 Sample and Hold Output
CHAN_CH	Test channel	Pixel 1 Charge Amplifier Output

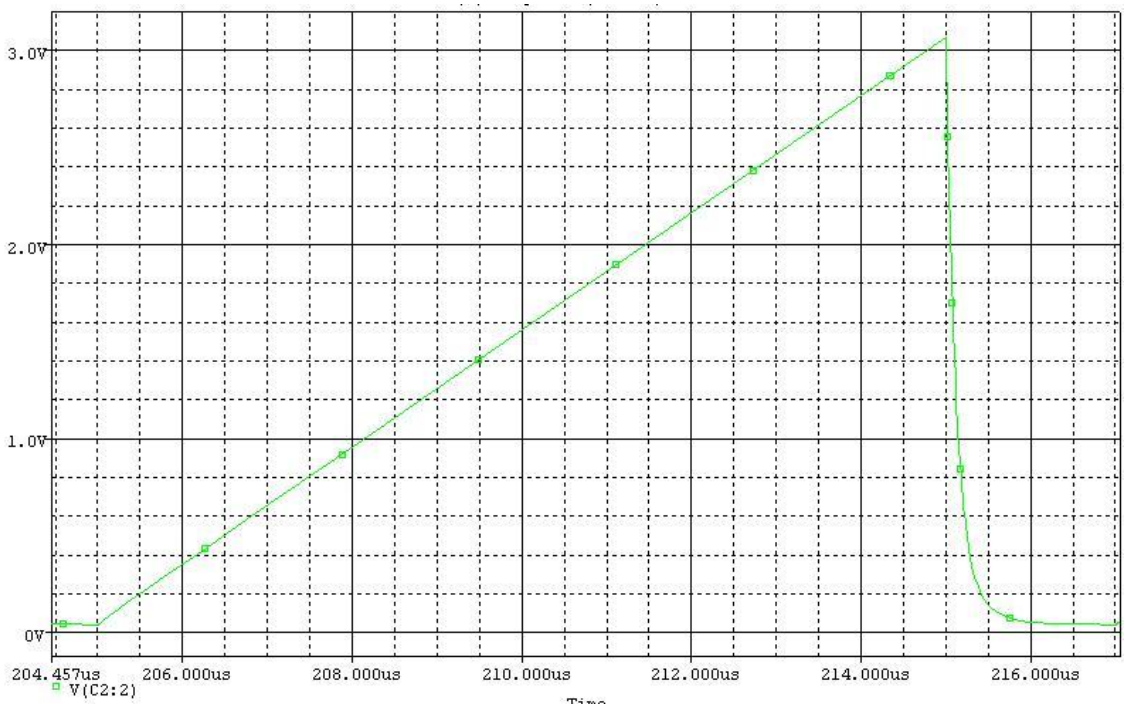
4.2.2 Κύκλωμα Ράμπας

Για την μετατροπή του αναλογικού σήματος σε ψηφιακό το ολοκληρωμένο χρησιμοποιεί την μέθοδο μονής ανοδικής ράμπας τάσης. Η λογική είναι ότι η στάθμη τάσης στην έξοδο του ενισχυτή φορτίου αποθηκεύεται στο υπο-κύκλωμα δειγματοληψίας και αποθήκευσης και είναι πλέον έτοιμο για μετατροπή σε ψηφιακό σήμα. Η έξοδος του κυκλώματος δειγματοληψίας και αποθήκευσης εφαρμόζεται στην μια από τις δύο εισόδους του συγκριτή, ο οποίος βρίσκεται σε κάθε pixel. Η δεύτερη είσοδος κάθε συγκριτή σε κάθε pixel είναι συνδεδεμένη με, το ίδιο κύκλωμα ράμπας, το οποίο είναι υλοποιημένο σε επίπεδο πλακέτας. Η ράμπα τάσης ξεκινά από τάση μηδέν Volt και ανεβαίνει γραμμικά μέχρι την τάση τροφοδοσίας, που στην περίπτωση μας είναι τα 3.3Volt. Την στιγμή που η ράμπα ξεκινά, αρχίζει να μετρά ένας Grey Code Counter, που είναι υλοποιημένος στο ψηφιακό τμήμα του ολοκληρωμένου. Όταν η ράμπα τάσης γίνει ίση με την στάθμη που θέλουμε να μετατρέψουμε, ενεργοποιείται ο συγκριτής και στην έξοδο του εμφανίζει έναν ασύγχρονο παλμό τάσης συγκεκριμένης διάρκειας. Ο παλμός αυτός ειδοποιεί το ψηφιακό τμήμα του ολοκληρωμένου και την στιγμή που γίνεται HIGH εγγράφεται στην δυναμική μνήμη, στο αντίστοιχο pixel η τρέχουσα τιμή του μετρητή Grey Code Counter.Στο παρακάτω σχηματικό διάγραμμα φαίνεται το κύκλωμα που σχεδιάστηκε για την υλοποίηση του κυκλώματος ράμπας.



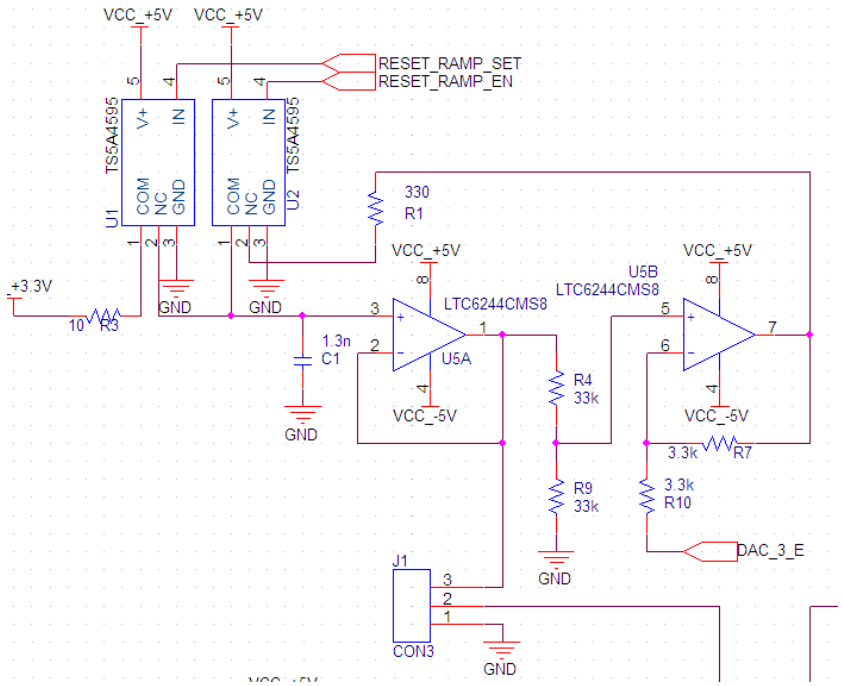
Εικόνα 4-7 Κύκλωμα ράμπας τάσης

Χρησιμοποιήθηκαν δύο τελεστικοί ενισχυτές (LTC6244), οι ίδιοι που χρησιμοποιήθηκαν και για το κύκλωμα πηγής ρεύματος και δύο αναλογικοί διακόπτες. Ο ένας από τους δύο τελεστικούς χρησιμεύει ως απομονωτής της τάσης που είναι αποθηκευμένη στον πυκνωτή φόρτισης. Ενώ ο δεύτερος προσφέρει μια κατάλληλη τάση στην αντίσταση R11 και το ρεύμα που δημιουργείται φορτίζει γραμμικά τον πυκνωτή C19. Οι δύο αναλογικοί διακόπτες ελέγχονται από το FPGA και έτσι ο χρήστης μπορεί να διακόψει την παροχή ρεύματος στον πυκνωτή ανοίγοντας τον ένα και έτσι να παγώσει την τρέχουσα τάση για όσο χρόνο επιθυμεί και επίσης μπορεί να μηδενίσει απότομα την τάση ράμπας κλείνοντας τον δεύτερο διακόπτη, γειώνοντας έτσι τον πυκνωτή. Στην Εικόνα 4-8 Έξοδος του κυκλώματος ράμπας φαίνεται η έξοδος του κυκλώματος ράμπας ρυθμισμένη για την εφαρμογή που θέλουμε.



Εικόνα 4-8 Έξοδος του κυκλώματος ράμπας

Επιπλέον όπως αναφέρεται στο τμήμα που περιγράφει την λειτουργία του ολοκληρωμένου, είναι απαραίτητο ένα σήμα (RESET) το οποίο πρέπει να ανεβαίνει απότομα στα 3.3Volt και αφού παραμείνει για κάποιο καθοριζόμενο από τον χρήστη χρονικό διάστημα στην τάση αυτή να αρχίσει να μειώνεται γραμμικά μέχρι τα μηδέν Volt. Για την δημιουργία αυτού του σήματος χρησιμοποιήθηκε μια παραλλαγή του κυκλώματος ράμπας έτσι ώστε να έχουμε απότομη φόρτιση του πυκνωτή και γραμμική αποφόρτιση του. Να σημειωθεί ότι και το ρεύμα που παρέχεται στον πυκνωτή είτε για την φόρτιση, είτε για την αποφόρτιση του είναι ελεγχόμενο από τον χρήστη μέσω της τάσης ελέγχου η οποία προέρχεται από έναν μετατροπέα ψηφιακού σήματος σε αναλογικό. Το σχηματικό φαίνεται στην Εικόνα 4-9.



Εικόνα 4-9

4.2.3 Πηγές Ρεύματος για την πόλωση του ολοκληρωμένου

Για την πόλωση του ολοκληρωμένου και συγκεκριμένα για την τροφοδοσία των εισόδων Icomp και Ipreamp χρησιμοποιήθηκε μια παραλλαγή του κυκλωματος των πηγών ρεύματος που αναφέρθηκε παραπάνω με μια παραλλαγή ώστε οι πηγές αυτές να προσφέρουν ρεύμα 4uA και 8uA αντίστοιχα.

4.2.4 Κυκλώματα τροφοδοσίας

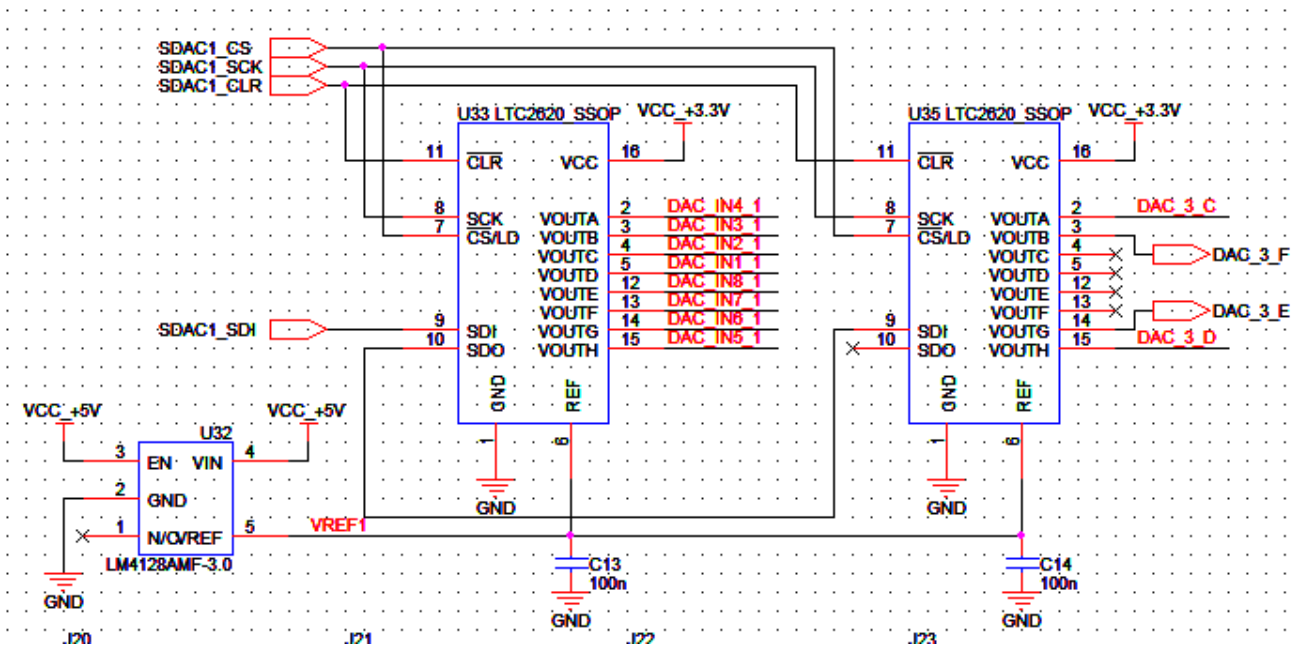
Για την τροφοδοσίας της Πλακέτας Π2 χρησιμοποιήθηκαν κανονικοποιητές τάσης (voltage regulators) για να παράγουν τάσεις 5.0V, -5.0V, 3.3V και 1.2V. Τα εξαρτήματα που χρησιμοποιήθηκαν είναι τα εξής:

Εξάρτημα	**Τάση εξόδου**
REG104FA	+5V
REG104GA	+3.3V
REG104FA	+3.3V

REG104GA	+3.3V
LM2991S	-5.0V
TPS76912	+1.2V

4.2.5 Κυκλώματα μετατροπής ψηφιακού σήματος σε αναλογικό

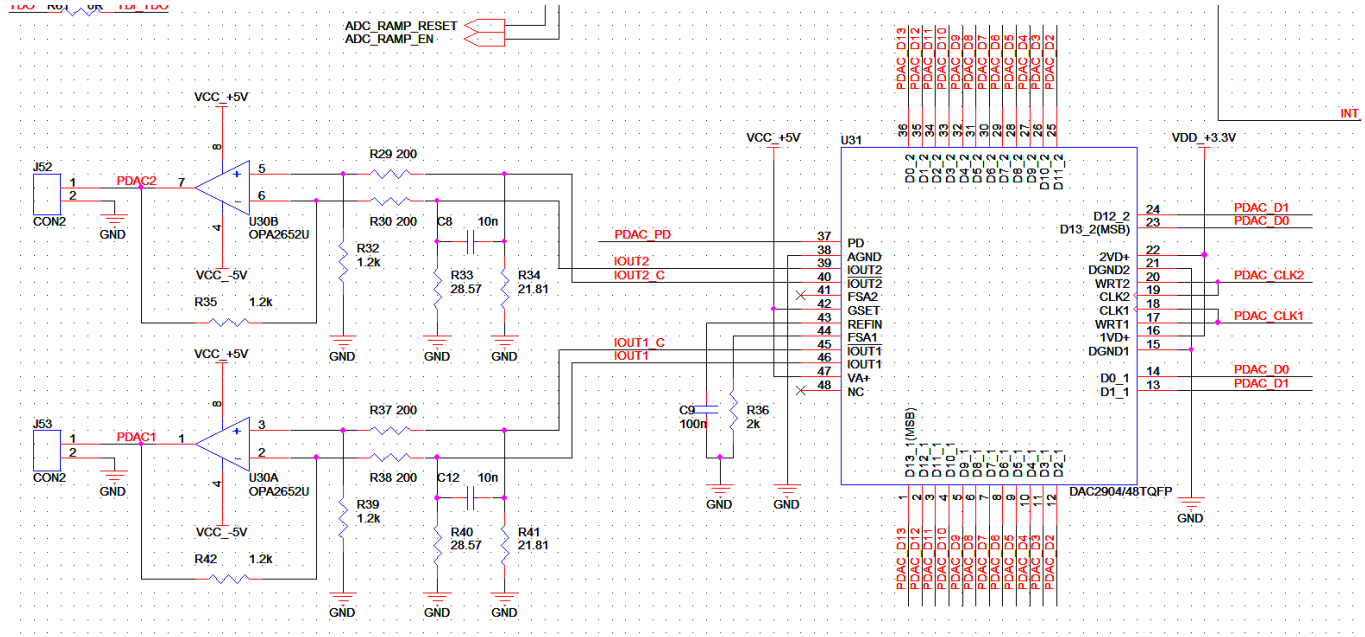
Για την λειτουργία και τον έλεγχο των αναλογικών υπο-κυκλωμάτων, αναφέρθηκε στις αντίστοιχες παραγράφους η ανάγκη μετατροπέων ψηφιακού σήματος σε αναλογικό. Έτσι για τον έλεγχο των πηγών ρεύματος (16) για την είσοδο των pixel, των πηγών ρεύματος πόλωσης στο ολοκληρωμένο (2), για τις δύο παραλλαγές του κυκλώματος ράμπας (2) και επιπλέον δύο τάσεις για την παροχή κατάλληλης εισόδου στα κυκλώματα IN_COMP και IN_SH (2) χρησιμοποιήθηκαν τέσσερις μετατροπείς με οκτώ κανάλια ο καθένας. Τα διαθέσιμα κανάλια είναι 8x4=32 και επειδή η συγκεκριμένες εφαρμογές δεν είχαν ιδιαίτερες απαιτήσεις σε ταχύτητα μετατροπής του σήματος επιλέχθηκε ο οκτα-κάναλος σειριακός μετατροπέας LTC2620 με διεπαφή SPI. Ο προγραμματισμός των μετατροπέων έγινε μέσω του FPGA και του USB μικροελεγκτή και θα αναφερθούν περισσότερα στο αντίστοιχο κεφάλαιο.



Εικόνα 4-10

4.2.6 Κύκλωμα ράμπας με χρήση DAC υψηλής ταχύτητας

Επειδή η μετατροπή του αναλογικού σήματος από τα pixel των ανιχνευτών σε ψηφιακό βασίζεται στην υλοποίηση του κυκλώματος ράμπας, σχεδιάστηκε και ένα δεύτερο κύκλωμα παραγωγής αυτού του σήματος. Η ιδέα είναι ότι με την βοήθεια ενός DAC υψηλής ταχύτητας μπορούμε να αναπαράγουμε ένα σήμα ράμπας. Ο συγκεκριμένος μετατροπέας έχει παράλληλη ψηφιακή είσοδο και ανάλογα με αυτήν δίνει στην έξοδο του ρεύμα αντί για τάση και μάλιστα διαφορικό. Για την μετατροπή του ρεύματος αυτού σε τάση αλλά και την μετατροπή του διαφορικού σήματος σε μονό (single ended), χρησιμοποιήθηκε ο τελεστικός ενισχυτής OPA2652 στην διάταξη που φαίνεται στο επόμενο σχηματικό διάγραμμα.

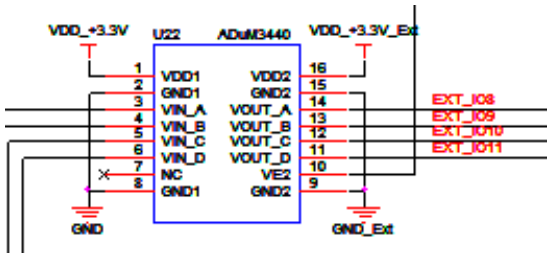


Εικόνα 4-11

4.2.7 Κύκλωμα απομόνωσης ανάμεσα στις πλακέτες Π2 και Π3

Η πλακέτα Π2 συνδέεται μέσω ενός connector με την πλακέτα Π3 η οποία αναλαμβάνει την μεταφορά των δεδομένων από και προς τον ηλεκτρονικό υπολογιστή. Κατά τον σχεδιασμό του συστήματος μια σημαντική παράμετρος είναι ο θόρυβος και επειδή η πλακέτα Π2 η οποία περιέχει το ολοκληρωμένο προς έλεγχο είναι μεικτού σήματος, έπρεπε με κάποιο τρόπο να διασφαλιστεί ότι κατά την διάρκεια της ολοκλήρωσης του ρεύματος σε κάθε pixel από τον ενισχυτή διεμπέδησης, ο θόρυβος να είναι όσο δυνατόν μικρότερος. Ένας τρόπος για να επιτευχθεί αυτό είναι να αφαιρεθούν από την συγκεκριμένη πλακέτα όλα

τα σήματα χρονισμού (ρολόγια) για όσο διαρκεί η ολοκλήρωση. Ο διαχωρισμός του συστήματος σε δύο πλακέτες εξυπηρετεί στο να είναι εφικτό να αφαιρεθούν όλα τα σήματα χρονισμού από την πλακέτα Π2. Κάτι τέτοιο δεν θα ήταν εφικτό εάν είχαμε το σύστημα υλοποιημένο σε μια πλακέτα καθώς ο USB μικροελεγκτής για να λειτουργήσει απαιτεί σήμα χρονισμού από ένα κρύσταλλο (ταλαντωτή). Με την χρήση απομονωτών ανάμεσα στις δύο πλακέτες επιτυγχάνεται ο πλήρης διαχωρισμός των σημάτων και ελαχιστοποιείται ο θόρυβος.



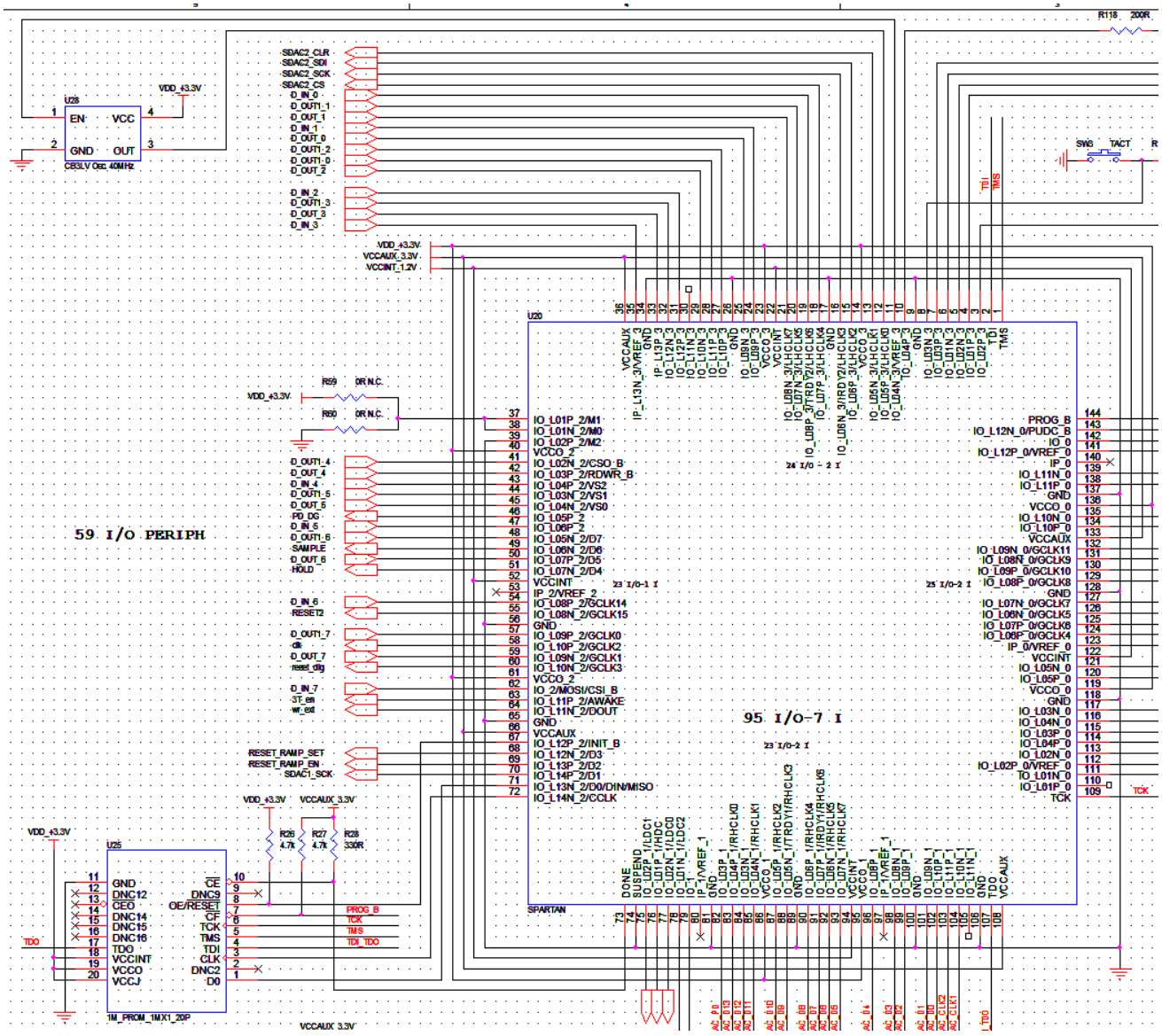
Εικόνα 4-12 ADUM3440

Τα εξαρτήματα που επιλέχθηκαν υποστηρίζουν ταχύτητες μεταφοράς ψηφιακών δεδομένων με ρυθμό έως και 150Mbps και διαθέτουν χαρακτηριστικά καλύτερα από τις φωτοδιόδους (optocouplers) οι οποίες μπορούν να χρησιμοποιηθούν για τον ίδιο σκοπό. Κάθε απομονωτής έχει τέσσερα κανάλια.

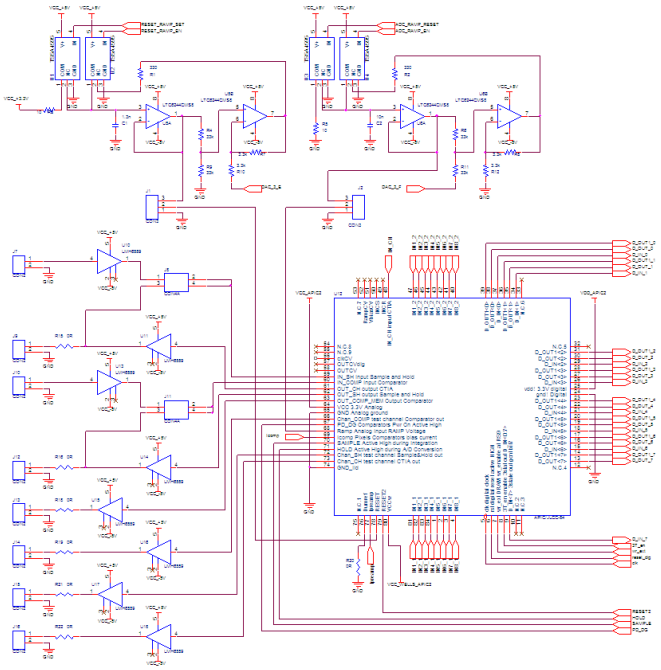
4.2.8 FPGA

Για την συγκεκριμένη εφαρμογή χρησιμοποιήθηκε το FPGA Xilinx Spartan 3AN 144TQFP, το οποίο αποτελεί ενα platform FPGA με 144 εξωτερικές διασυνδέσεις και ενσωματωμένη Flash memory έτσι ώστε να κρατάει αποθηκευμένη την ρύθμιση του και όταν αφαιρεθεί η τροφοδοσία, χωρίς την ανάγκη ύπαρξης εξωτερικής μνήμης. Η διαδικασία της τοποθέτησης και ρύθμισης ενός τέτοιου FPGA σε μια πλακέτα, είναι μια ιδιαίτερα εξειδικευμένη και τεχνική εργασία όπου απαιτείται ο σχεδιαστής να γνωρίζει ακριβώς την εφαρμογή και να γνωρίζει τις δυνατότητες του FPGA, ώστε να το ρυθμίσει σωστά. Επιπλέον απαιτείται γνώση όλων των παραμέτρων του συγκεκριμένου FPGA και μελέτη του εγχειριδίου του. Βασικές ρυθμίσεις είναι το ποιες διασυνδέσεις θα συνδεθούν με τα εξωτερικά ψηφιακά σήματα είτε ως είσοδοι είτε ως έξοδοι αλλά και οι τάσεις τροφοδοσίας καθεμιάς από της τέσσερις ομάδες διασυνδέσεων που διαθέτει το συγκεκριμένο FPGA. Επίσης πρέπει να ελεγχθεί προσεκτικά το κύκλωμα, οι διασυνδέσεις προγραμματισμού του και οι τροφοδοσίες του, καθώς κάποιο σφάλμα σχεδίασης στο συγκεκριμένο μέρος του κυκλώματος μπορεί να καταστήσει άχρηστή την πλακέτα μετά την κατασκευή της. Στην

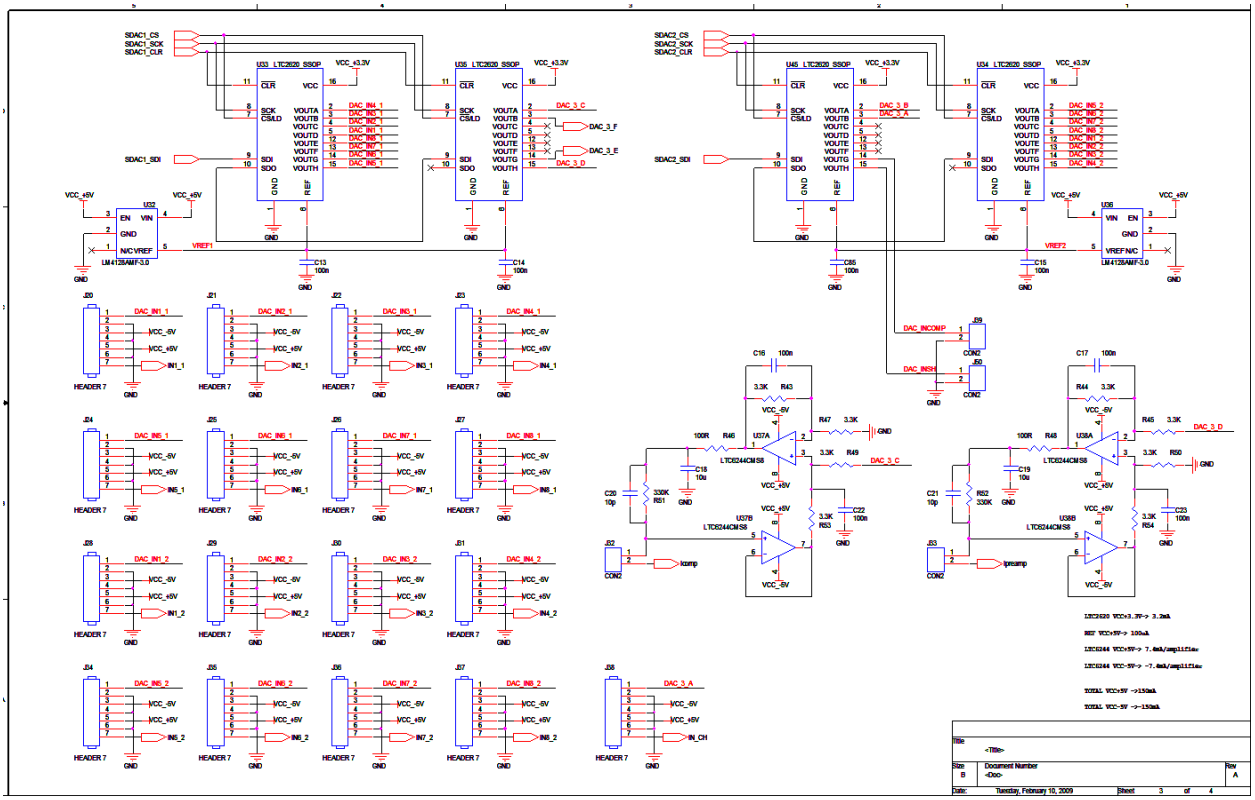
εφαρμογή που μελετάμε ή τάση τροφοδοσίας είναι 3.3V και ο προγραμματισμός γίνεται μέσω του JTAG καλωδίου. Στην Εικόνα 4-13 Ρύθμιση (configuration) του Spartan 3AN φαίνεται η ρύθμιση (configuration) του Spartan 3AN FPGA της Xilinx για την πλακέτα Π2.



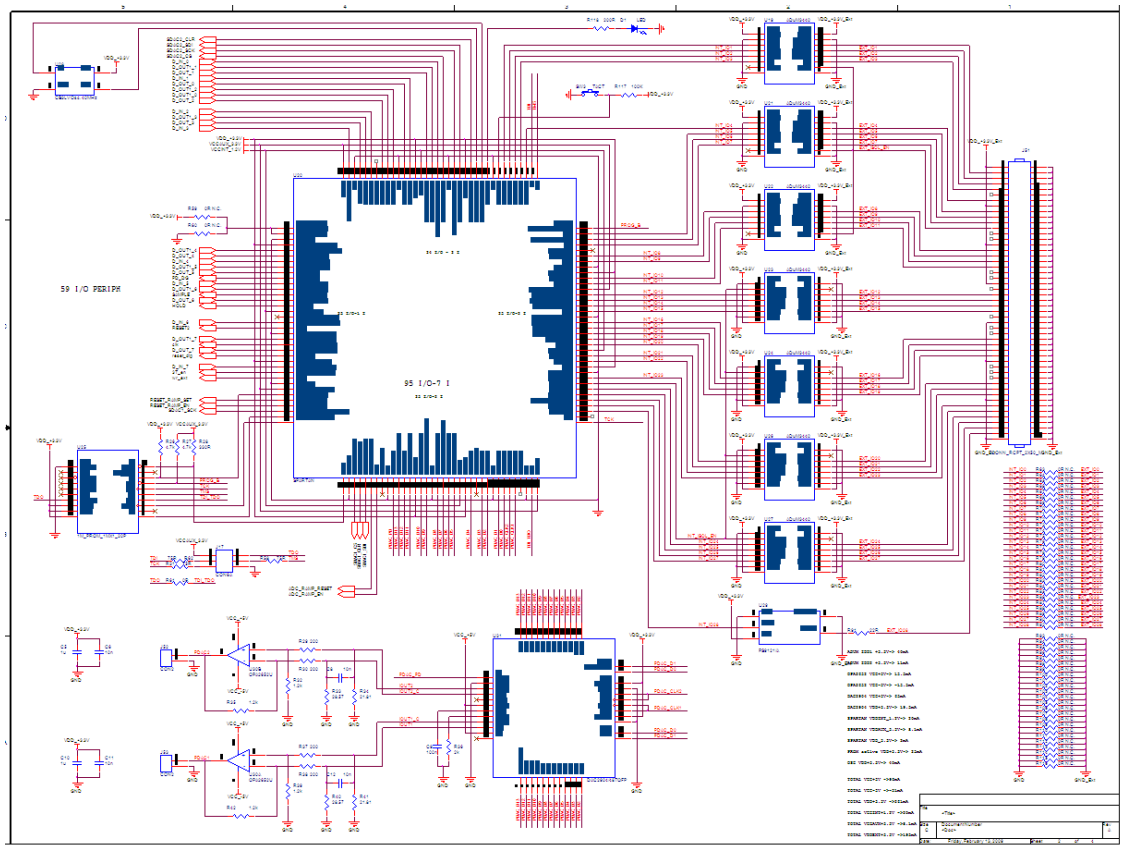
Εικόνα 4-13 Ρύθμιση (configuration) του Spartan 3AN



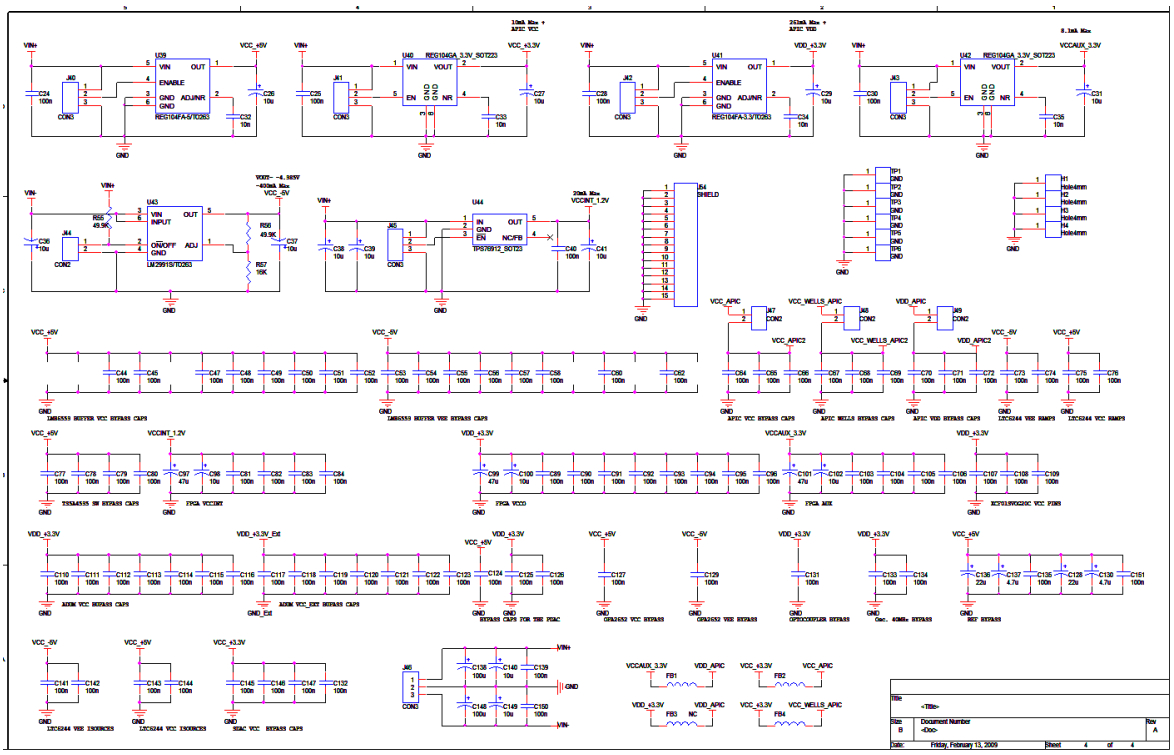
Εικόνα 4-14 Σχηματικό διάγραμμα ολοκληρωμένου, ράμπας τάσης και ενισχυτών των αναλογικών I/O



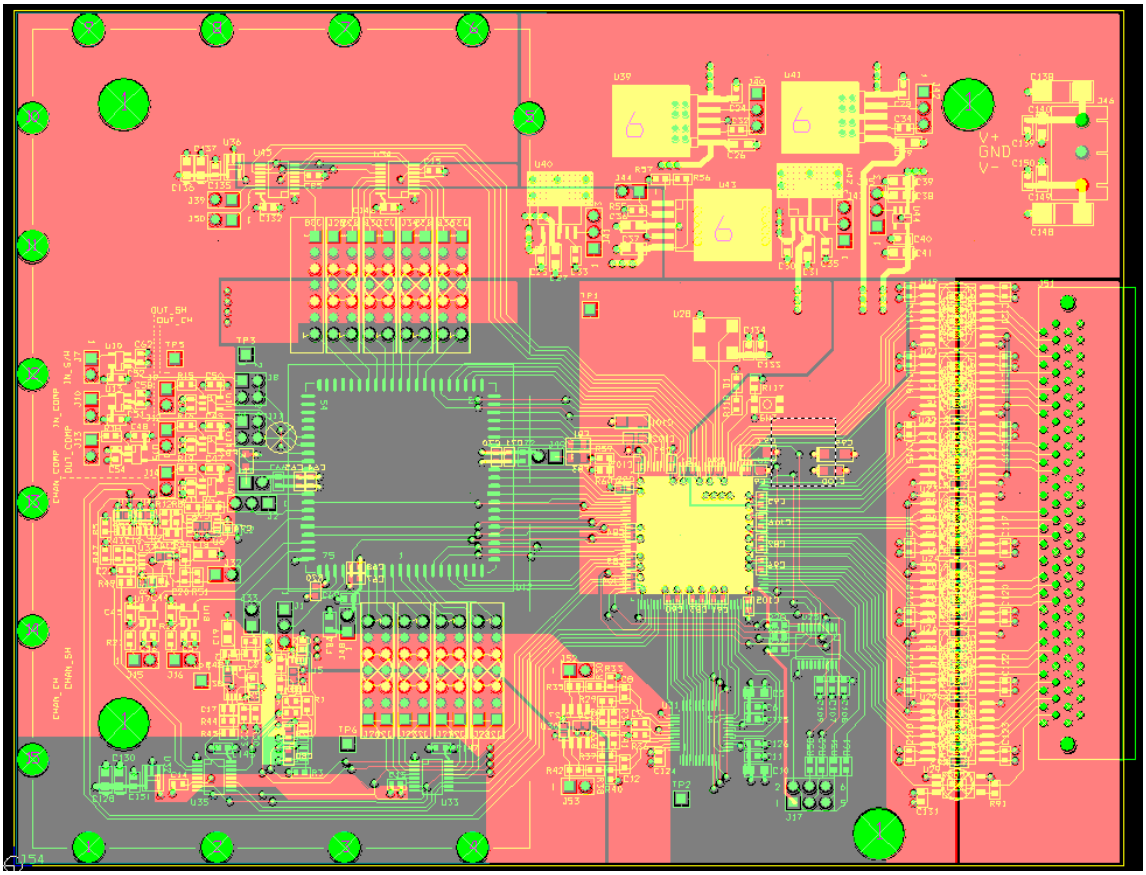
Εικόνα 4-15 Σχηματικό διάγραμμα DAC και πηγών ρεύματος



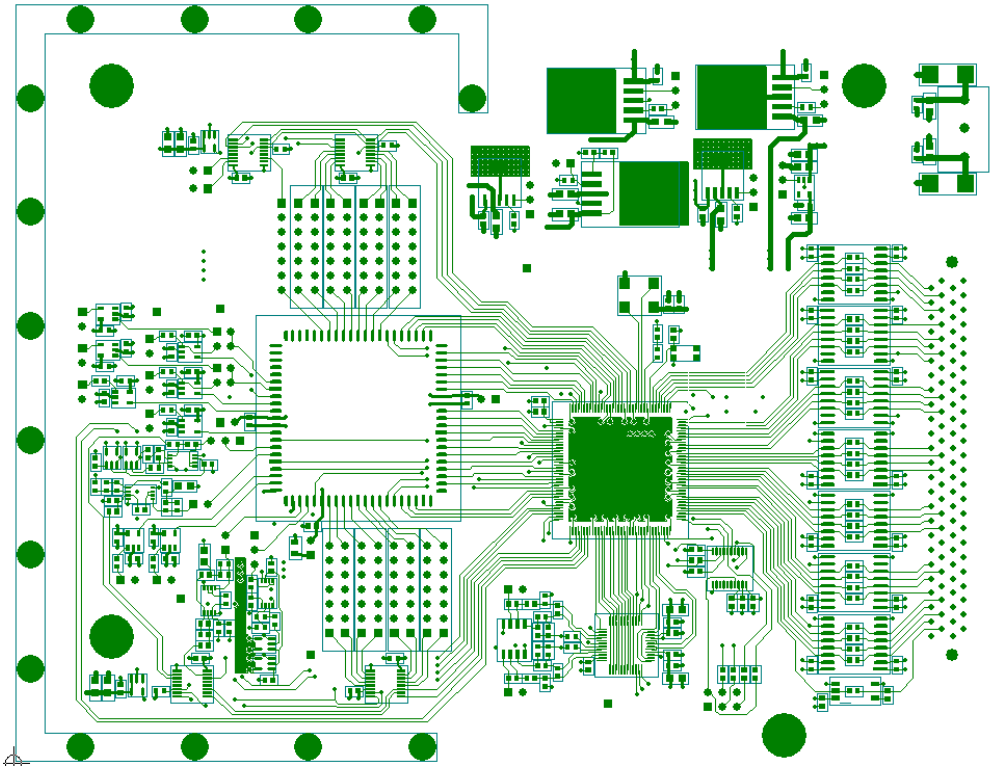
Εικόνα 4-16 Σχηματικό διάγραμμα FPGA, DAC και Isolators



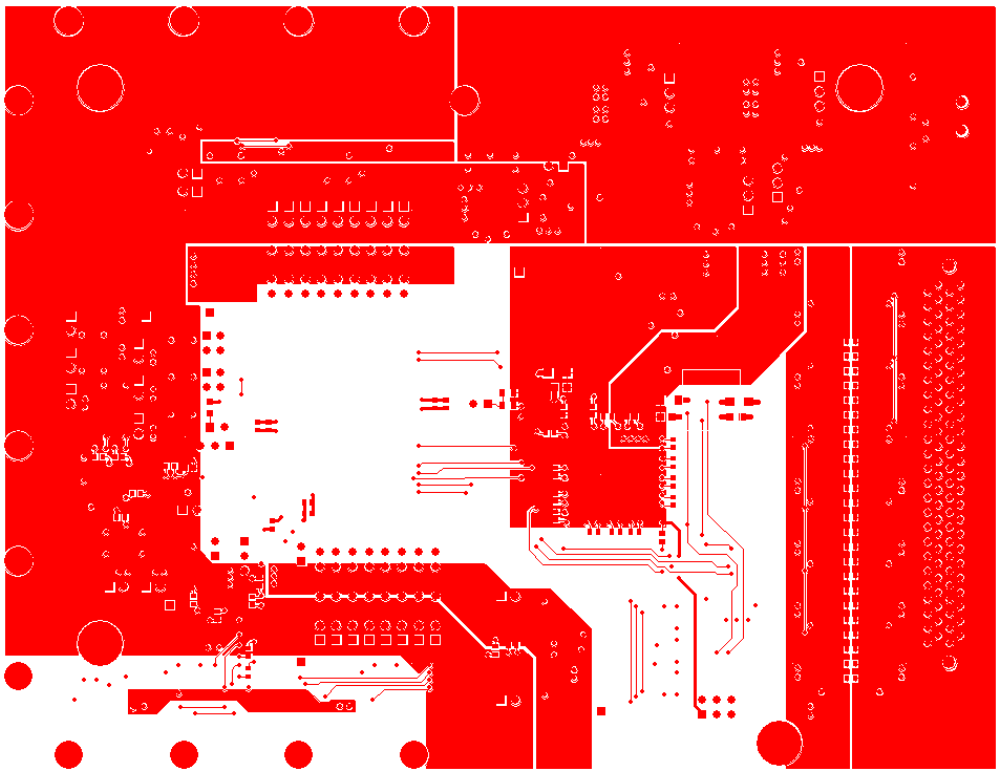
Εικόνα 4-17 Σχηματικό διάγραμμα τροφοδοσίας



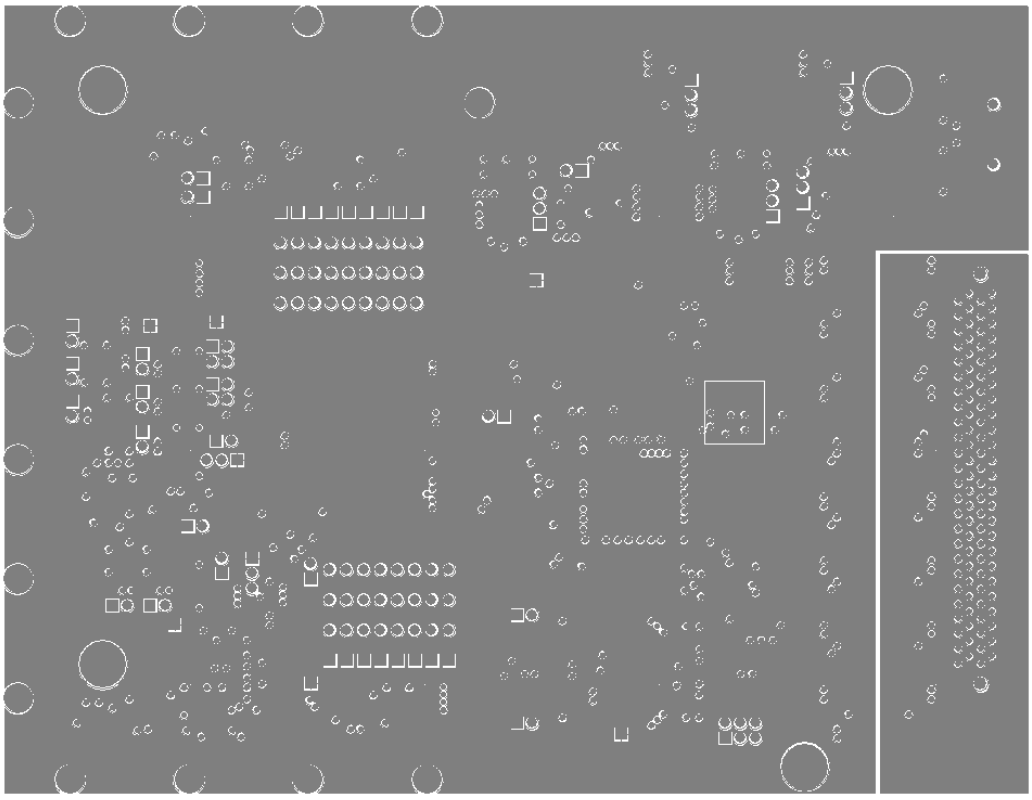
Εικόνα 4-18 Φυσικός σχεδιασμός πλακέτας Π2



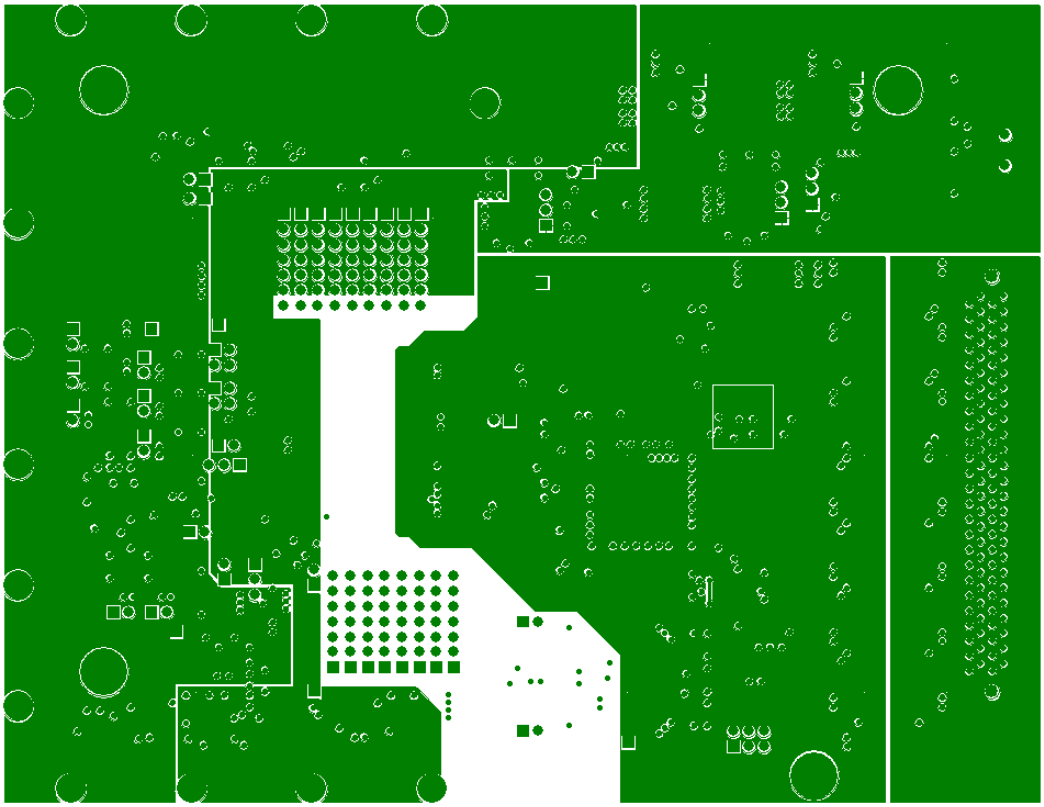
Εικόνα 4-19 Επάνω στρώμα του τυπωμένου κυκλώματος



Εικόνα 4-20 Κάτω στρώμα του τυπωμένου κυκλώματος



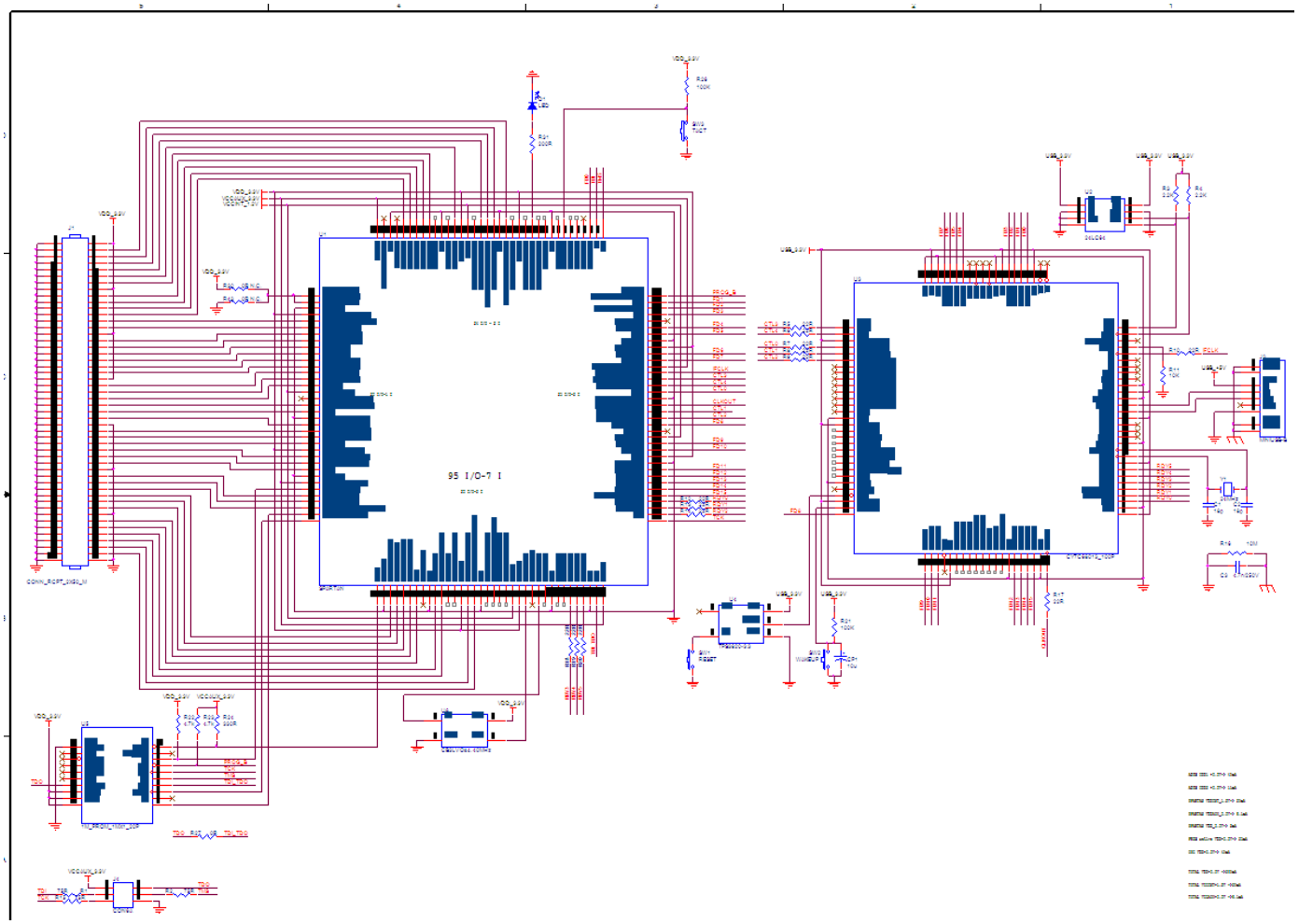
Εικόνα 4-21 Στρώμα γείωσης (πρώτο εσωτερικό στρώμα)



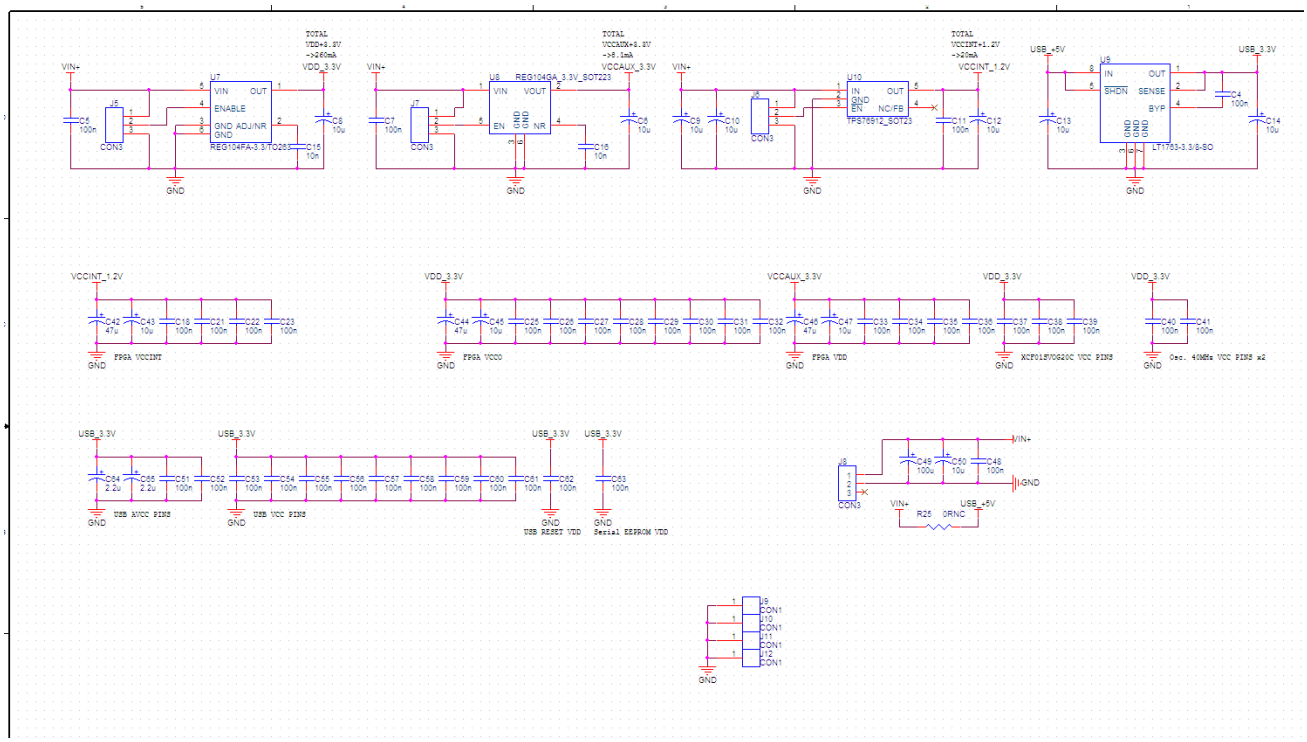
Εικόνα 4-22 Στρώμα τροφοδοσίας (δεύτερο εσωτερικό στρώμα)

4.3 Πλακέτα Π3

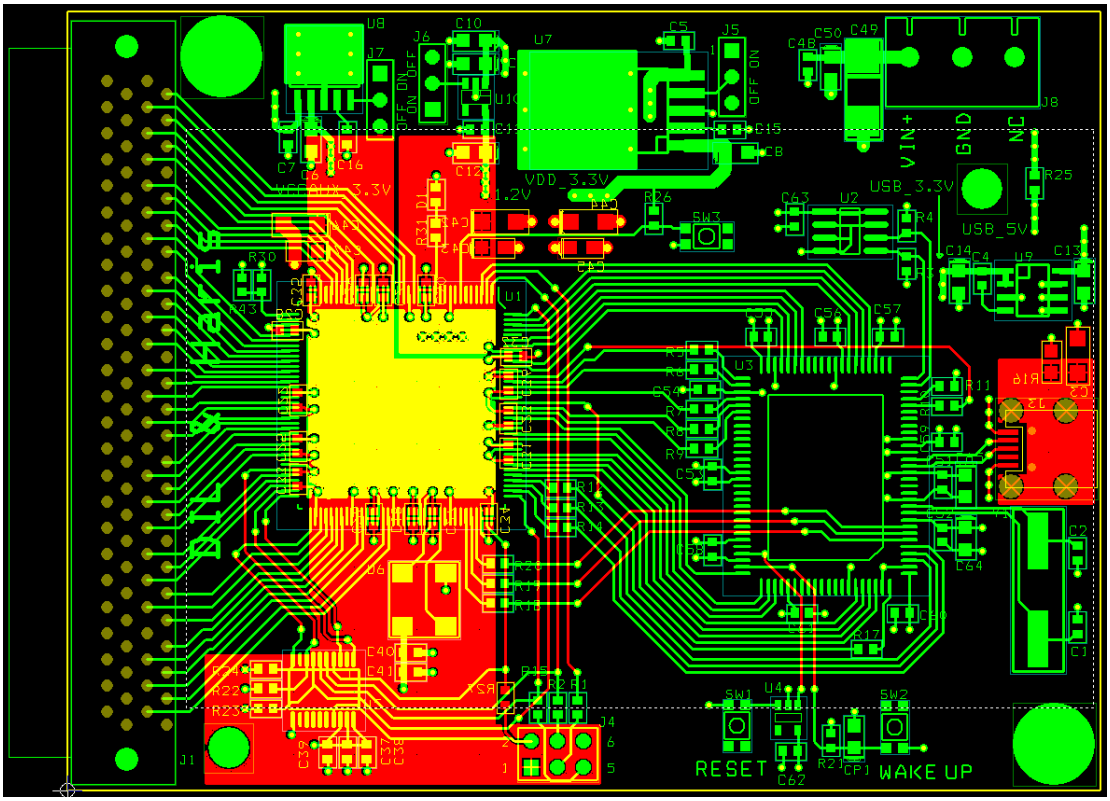
Η δεύτερη πλακέτα του συστήματος ελέγχου και λήψης δεδομένων αποτελείται από ένα FPGA Spartan 3AN και έναν USB 2.0 μικροελεγκτή, καθώς και από τα απαραίτητα παθητικά στοιχεία και τις τροφοδοσίες. Η σχεδίαση αυτού του κυκλώματος χρησιμεύει στην απομόνωση της πλακέτας Π2 από τα ρολόγια του συστήματος, προσφέρει περισσότερη χωρητικότητα λογικής, εξαιτίας του δεύτερου FPGA. Επιπλέον προσφέρει μια διεπαφή FPGA – USB, η οποία μπορεί να επαναχρησιμοποιηθεί και για άλλες εφαρμογές. Στα αριστερά της Εικόνα 4-23 Σχηματικό διάγραμμα διεπαφής FPGA – USBφαίνεται το FPGA και δεξιά το USB.



Εικόνα 4-23 Σχηματικό διάγραμμα διεπαφής FPGA – USB



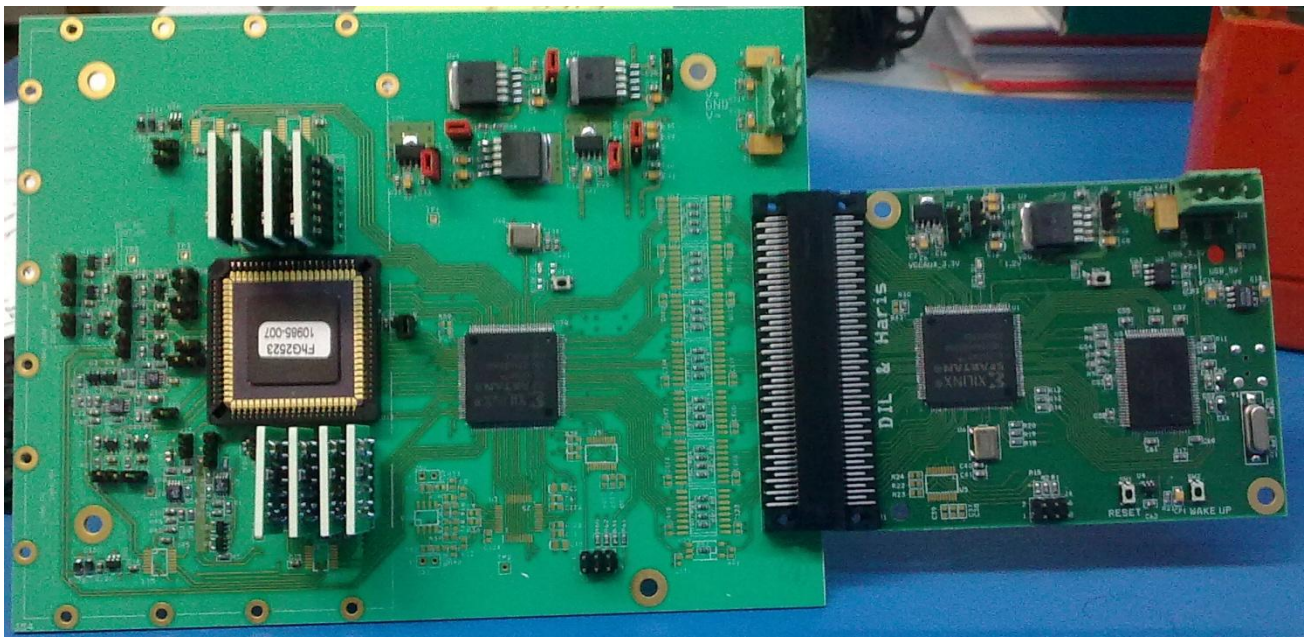
Εικόνα 4-24 Σχηματικό διάγραμμα τροφοδοσίας της πλακέτας Π3



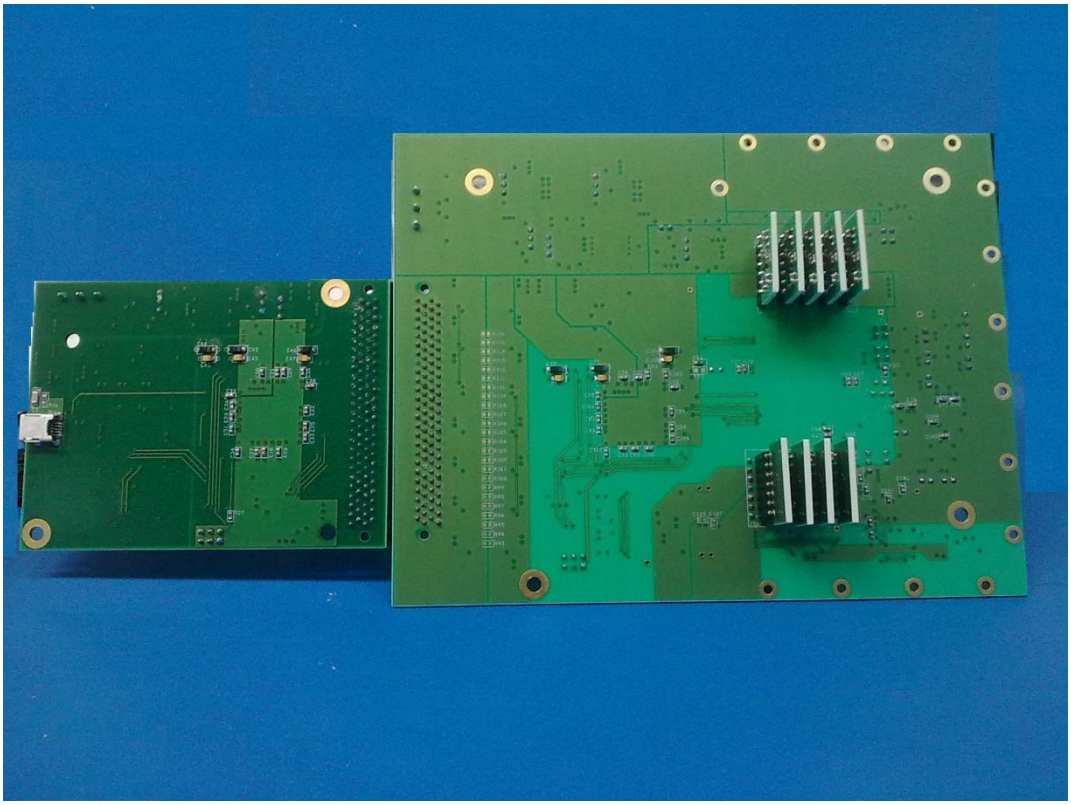
Εικόνα 4-25 Φυσικός σχεδιασμός πλακέτας Π3 (διακρίνονται το επάνω και το κάτω στρώμα)

4.4 Συναρμολόγηση και έλεγχος λειτουργίας των ηλεκτρονικών ελέγχου.

Η συναρμολόγηση των τριών πλακετών με τα εξαρτήματα επιφανειακής συγκόλλησης έγινε χειροκίνητα για τον καλύτερο έλεγχο της λειτουργίας τους. Το συγκεκριμένο τμήμα της υλοποίησης των ηλεκτρονικών ανάγνωσης είναι ιδιαίτερα χρονοβόρο και απαιτεί σταδιακή συναρμολόγηση και έλεγχο των κυκλωμάτων, ιδιαίτερα όταν και οι δύο πλακέτες προς συναρμολόγηση μαζί ξεπερνούσαν τα 600 εξαρτήματα. Αρχικά συναρμολογήθηκαν τα κυκλώματα τροφοδοσίας και με την χρήση πολύμετρου και παλμογράφου διαπιστώθηκε η ορθή τροφοδοσία σε όλες τις διασυνδέσεις όλων των ολοκληρωμένων. Συναρμολογήθηκαν πρώτα τα αναλογικά και μετά τα ψηφιακά. Τελευταία συγκολλήθηκαν τα ενσωματωμένα κυκλώματα και το ολοκληρωμένο προς έλεγχο. Ο λειτουργικός έλεγχος του FPGA και του USB ολοκληρώθηκε με τον προγραμματισμό τους με κάποιες στοιχειώδεις υλοποιήσεις (πχ ανάβοντας μια Led). Με το τέλος των λειτουργικών δοκιμών έγινε προγραμματισμός των FPGA και USB για να ελεγχθεί η συνδεσιμότητα όλων των αρτηριών που συνδέουν τα δύο FPGA και το USB. Ο στόχος ήταν να ελεγχθεί και να διαπιστωθεί η καλή και εντός προδιαγραφών λειτουργία των ηλεκτρονικών σε επίπεδο πλακετας, ώστε να μπορεί να γίνει μετά ο έλεγχος του ολοκληρωμένου όσο το δυνατόν πιο αξιόπιστα. Η μόνη αποσφαλμάτωση που χρειάστηκε ήταν η προσθήκη μιας επιπλέον αντίστασης ώστε να διατηρεί το FPGA της κάρτας Π2 την ρύθμιση του και μετά το σβήσιμο της τροφοδοσίας στην εσωτερική του μνήμη. Η συνολική κατανάλωση των ηλεκτρονικών σε πλήρη λειτουργία μετρήθηκε 350mA με συνδεδεμένο το ολοκληρωμένο και ενεργοποιημένες όλες τις αναλογικές και ψηφιακές δομές.



Εικόνα 4-26 Επάνω όψη συστήματος ελέγχου



Εικόνα 4-27 Κάτω όψη συστήματος ελέγχου

Εικόνα 4-28

5 Field Programmable Gate Array

Οι διατάξεις πυλών προγραμματιζόμενου πεδίου χρησιμοποιούνται τα τελευταία χρόνια όλο και περισσότερο στο σχεδιασμό ψηφιακών συστημάτων. Τα FPGA δεν είναι κυκλώματα σχεδιασμένα για την υλοποίηση μιας συγκεκριμένης λειτουργίας οπότε δεν φτάνουν την απόδοση ενός asic. Είναι γενικά πιο αργά και καταναλώνουν περισσότερη ενέργεια. Τα μειονεκτήματα τους αντισταθμίζονται επειδή είναι άμεσα προγραμματιζόμενα και αναπρογραμματίζονται πολλές φορές. Επίσης είναι ιδανικά για την σχεδίαση πρωτότυπων και από την στιγμή που έχει υλοποιηθεί το πρωτότυπο είναι αρκετά ευκολότερο και έχει μικρότερο ρίσκο καθώς το σχέδιο που πρέπει να μεταφερθεί σε πυρίτιο είναι δοκιμασμένο ήδη.

Τα FPGAs παρέχουν προγραμματιζόμενη λογική χρησιμοποιώντας λογική πολλαπλών επιπέδων, έτσι κάνουν χρήση προγραμματιζόμενων στοιχείων λογικής και προγραμματιζόμενες διασυνδέσεις ανάμεσα σε αυτά. Γενικά ένα fpga αποτελείται από τα εξής βασικά μέρη, την συνδυαστική λογική, την δομή διασύνδεσης και τα σημεία εισόδου - εξόδου. Η συνδυαστική λογική διαιρείται σε σχετικά μικρές μονάδες οι οποίες καλούνται στοιχεία λογικής (logic elements LEs) ή μπλοκ συνδυαστικής λογικής (Combinational logic blocks - CLBs). Το LE ή το CLB μπορεί συνήθως να διαμορφωθεί έτσι ώστε να υλοποιεί μια συνάρτηση ή οποία θα απαιτούσε πολλές λογικές πύλες. Η διασύνδεση των στοιχείων λογικής μεταξύ τους είναι προγραμματιζόμενη. Αυτή μπορεί να οργανωθεί σε λογικά κανάλια ή άλλες μονάδες. Τα FPGAs διαθέτουν διάφορους τύπους διασυνδέσεων ανάλογα με την απόσταση μετάξι τον στοιχείων λογικής που πρόκειται να συνδεθούν. Τα σήματα ρολογιού παρέχονται επίσης μαζί με το δίκτυο διασύνδεσης τους. Τα σημεία I/O καλούνται επίσης μπλοκ I/O - IOBs . Αυτά μπορούν να προγραμματιστούν έτσι ώστε να είναι είσοδοι ή έξοδοι και να παρέχουν συχνά αλλά χαρακτηριστικά όπως χαμηλή κατανάλωση ισχύος ή γρήγορες συνδέσεις.

Το σύστημα διασύνδεσης ενός fpga είναι μια από της πιο σύνθετες πτυχές του, επειδή ή καλωδίωση είναι μια καθολική ιδιότητα ενός στοιχείου λογικής. Οι συνδέσεις μετάξι των στοιχείων λογικής μπορούν να ακολουθούν σύνθετες πορείες δεδομένου ότι τα LEs τακτοποιούνται σε κάποιο είδος δισδιάστατης δομής. Επομένως οι συνδέσεις δεν γίνονται μόνο μετάξι των LEs και των καλωδίων αλλά και μετάξι των καλωδίων των ίδιων. Τα καλώδια οργανώνονται συνήθως σε κανάλια καλωδίων ή κανάλια δρομολόγησης τα οποία διατρέχουν το τσιπ οριζόντια και κάθετα. Επίσης προκειμένου να γίνουν όλες οι απαραίτητες συνδέσεις μετάξι των στοιχείων λογικής, τα κανάλια περιέχουν καλώδια ποικίλων μηκών.

Για να χρησιμοποιηθεί ένα FPGA πρέπει πρώτα να προγραμματιστεί. Και τα τρία βασικά στοιχεία του θα πρέπει να διαμορφωθούν. Υπάρχουν τρεις σημαντικές τεχνολογίες κυκλωμάτων για την διαμόρφωση ενός FPGA: οι στατικές μνήμες RAM, η τεχνική Antifuse και μνήμες FLASH. Οι δύο τελευταίες αναφέρονται σε μόνιμο προγραμματισμό, δηλαδή δεν υπάρχει δυνατότητα αναπρογραμματισθούν, ενώ οι πρώτες δίνουν την δυνατότητα αυτή.

5.1 FPGAs προγραμματιζόμενα από static RAMs

Η στατική μνήμη είναι ή ευρύτερα διαδεδομένη μέθοδος διαμόρφωσης των FPGAs. Η έξοδος του κάθε κυττάρου μνήμης συνδέεται άμεσα με ένα άλλο κύκλωμα και έτσι ή κατάσταση του ελέγχει συνεχώς το κύκλωμα το οποίο διαμορφώνεται. Τα πλεονεκτήματα είναι τα παρακάτω:

1. Το FPGA μπορεί να αναπρογραμματιστεί εύκολα. Επειδή τα FPGA μπορούν να επαναχρησιμοποιηθούν και να αναπρογραμματιστούν γενικά χωρίς να αφαιρεθούν από το κύκλωμα είναι η πιο κατάλληλη επιλογή για τη δημιουργία πρωτότυπων.
2. Μπορεί να αναπρογραμματιστεί κατά τη διάρκεια της λειτουργίας του υλοποιώντας δυναμικά συστήματα.
3. Τα κυκλώματα που χρησιμοποιούνται στο FPGA μπορούν να κατασκευαστούν με διαδικασίες που χρησιμοποιούνται σε κυκλώματα πολύ μεγάλης κλίμακας ολοκλήρωσης.

Μειονεκτήματα:

1. Η μνήμη διαμόρφωσης SRAM καταναλώνει αρκετή ενέργεια ακόμα και όταν το πρόγραμμα παραμένει αμετάβλητο.
2. Τα ψηφία διαμόρφωσης μπορούν εύκολα να υποκλαπούν.
3.

5.2 Στοιχεία λογικής

Η βασική μέθοδος που χρησιμοποιείται για να υλοποιηθεί ένα στοιχείο λογικής διαμορφώσιμο από SRAM είναι με τη βοήθεια ενός πίνακα ο οποίος ονομάζεται πίνακας αναζήτησης (LUT) ο LUT είναι μια μνήμη SRAM που χρησιμοποιείται για να υλοποιήσει ένα πίνακα αλήθειας. Κάθε διεύθυνση της SRAM αντιπροσωπεύει ένα συνδυασμό εισόδων στο στοιχείο λογικής. Η τιμή που αποθηκεύεται σε εκείνη την διεύθυνση αντιπροσωπεύει την

τιμή της λογικής συνάρτησης που υλοποιείται για αυτόν τον συνδυασμό εισόδων. Μια συνάρτηση ν εισόδων απαιτεί μια μνήμη SRAM με 2 εις την ν διευθύνσεις. Επειδή μια SRAM διαμόρφωσης δεν έχει είσοδο ρολογιού, ο LUT λειτουργεί σαν μια λογική πύλη: Όταν οι είσοδοι της αλλάζουν, αλλάζουν και οι έξοδοι της με κάποια στοιχειώδη χρονική καθυστέρηση.

Αντίθετα όμως από μια λογική πύλη, ή συνάρτηση που υλοποιεί το στοιχείο λογικής, LE μπορεί να αλλάζει με την αλλαγή των τιμών των ψηφίων που αποθηκεύονται στην SRAM. Σαν αποτέλεσμα, ένα LE n εισόδων μπορεί να αντιπροσωπεύει 2^{2^n} συναρτήσεις. Ένα τυπικό στοιχείο λογικής έχει τέσσερις εισόδους. Σημειώνεται ότι ή καθυστέρηση μέσω του LUT είναι ανεξάρτητη από τα ψηφία που αποθηκεύονται στην SRAM. Έτσι, ή καθυστέρηση μέσω ενός στοιχείου λογικής είναι ίδια για όλες της συναρτήσεις που αυτό υλοποιεί.

Τα στοιχεία λογικής περιέχουν γενικά καταχωρητές, φλίπ - φλόπ και μανδαλωτές, καθώς επίσης και συνδυαστική λογική. Ένα φλίπ - φλόπ ή ένας μανδαλωτής έχουν μικρό μέγεθος σε σύγκριση με το συνδυαστικό στοιχείο λογικής, οπότε δεν δημιουργείται πρόβλημα αν προστεθούν στο LE. Η χρήση ενός ξεχωριστού κυττάρου για το στοιχείο μνήμης θα απορροφούσε απλά τους πόρους δρομολόγησης (routing resources - θα δέσμευε περισσότερα καλώδια). Το στοιχείο μνήμης είναι συνδεδεμένο με την έξοδο του LE. Το λογικό στοιχείο μπορεί επίσης να έχει ποιο περίπλοκη λογική. Για παράδειγμα, πολλά LE περιέχουν ειδικά κυκλώματα για υλοποίηση της πρόσθεσης. Το κρίσιμο συστατικό ενός αθροιστή είναι ή αλυσίδα του κρατουμένου, ή οποία μπορεί να υλοποιηθεί αποτελεσματικότερα με εξειδικευμένη λογική παρά χρησιμοποιώντας LUTs.

5.3 Διασύνδεση στοιχείων λογικής.

Τα στοιχεία λογικής πρέπει να μπορούν να διασυνδεθούν για να υλοποιηθούν σύνθετες συναρτήσεις. Το FPGA χρησιμοποιεί την μνήμη SRAM για να φυλάξει την πληροφορία που χρησιμοποιείται για να προγραμματιστούν οι διασυνδέσεις. Κατά συνέπεια οι διασυνδέσεις μπορούν να διαμορφωθούν ακριβώς όπως τα στοιχεία λογικής.

Ένα σημείο διασύνδεσης καλείται συχνά κουτί σύνδεσης. Μια προγραμματιζόμενη σύνδεση μετάξι δύο καλωδίων μπορεί να γίνει με χρήση ενός τρανζίστορ CMOS που λειτουργεί ως διακόπτης. Η πύλη του τρανζίστορ ελέγχεται από ένα ψηφίο στατικής μνήμης. Όταν ή πύλη του τρανζίστορ είναι HIGH αυτό άγει και συνδέει τα δύο καλώδια ενώ όταν ή πύλη είναι LOW το τρανζίστορ δεν άγει και τα δύο καλώδια δεν συνδέονται. Μια εναλλακτική

προσέγγιση στην υλοποίηση προγραμματιζόμενων σημείων διασύνδεσης είναι ή χρήση πυλών τριών καταστάσεων ως μονάδες προσωρινής αποθήκευσης (tri-state buffers).

Η καλωδίωση ενός FPGA με προγραμματιζόμενες διασυνδέσεις είναι πιο αργή από την τυπική καλωδίωση για δύο λόγους: για το τρανζίστορ και το μήκος των καλωδίων. Το τρανζίστορ δεν είναι τέλειος διακόπτης και έτσι ένα προγραμματιζόμενο σημείο διασύνδεσης είναι κάπως πιο αργό από ένα ζευγάρι καλωδίων που συνδέονται μόνιμα. Επιπλέον τα καλώδια στο FPGA είναι γενικά μακρύτερα από ότι σε ένα ASIC και σχεδιάζονται έτσι για να μπορούν να συνδέουν ποικίλα στοιχεία λογικής και άλλους πόρους πάνω στο τσιπ.

Ένα FPGA απαιτεί έναν μεγάλο αριθμό προγραμματιζόμενων καλωδίων προκειμένου να εκμεταλλευτεί πλήρως τα LEs. Ο διαχωρισμός των τύπων των καλωδίων γίνεται συχνά με βάση τη δομή και τη χρήση για την οποία προορίζονται:

- Τα κοντά καλώδια συνδέουν μόνο γειτονικά LEs, δεν καταλαμβάνουν πολύ χώρο και εισάγουν την μικρότερη δυνατή καθυστέρηση.
- Τα καθολικά καλώδια σχεδιάζονται ειδικά για επικοινωνία σημείων που απέχουν μεγάλη απόσταση.
- Τα ειδικά καλώδια αφιερώνονται σε ειδικές λειτουργίες όπως το να διανείμουν το σήμα ρολογιού ή αλλά σήματα ελέγχου.
-

5.4 Διαμόρφωση συστήματος

Τα FPGAs με SRAMs αναδιαμορφώνονται με την αλλαγή του περιεχομένου των μνημών διαμόρφωσης που ενσωματώνουν. Μερικοί πόροι στο τσιπ χρησιμοποιούνται για αυτό το σκοπό και είναι δυνατόν να χρησιμοποιηθούν για την διαμόρφωση και έπειτα να απελευθερωθούν και να μπορούν να χρησιμοποιηθούν σαν είσοδοι - έξοδοι I/O. Κατά την διάρκεια της διαμόρφωσης πρωτότυπων και της αποσφαλμάτωσης ενός σχεδίου, συνήθως ή διαμόρφωση αλλάζει συχνά. Μπορεί να χρησιμοποιηθεί ένα καλώδιο ώστε να γίνει ή φόρτωση από έναν ηλεκτρονικό υπολογιστή. Μόλις το FPGA τροφοδοτηθεί με ρεύμα εκτελεί την διαδικασία διαμόρφωσης. Πολλά σύγχρονα FPGAs ενσωματώνουν αλυσίδες σάρωσης αναδιαμόρφωσης στα κυκλώματα ελέγχου τους. Τα κυκλώματα ελέγχου χρησιμοποιούνται για να εξασφαλίσουν ότι το τσιπ κατασκευάστηκε σωστά. Το πρωτόκολλο JTAG δημιουργήθηκε για τον προγραμματισμό τέτοιων συσκευών. Το JTAG εντοπίζεται στα I/O και στις διασυνδέσεις προγραμματισμού του FPGA . Κατά την διάρκεια του ελέγχου, αυτές οι διασυνδέσεις (pins) μπορούν να αποσυνδεθούν από της κανονικές λειτουργίες τους και να χρησιμοποιηθούν ως καταχωρητές μετατόπισης (shift registers) . Η διαδικασία ελέγχεται από τον ελεγκτή της θύρας ελέγχου πρόσβασης (TAP - controller). Ο

ελεγκτής συνδέεται σε τέσσερα σημεία διασύνδεσης, την είσοδο του καταχωρητή μετατόπισης (TDI), την έξοδο του (TDO), το ρολόι ελέγχου (TCK), και τον τρόπο ελέγχου (TMΣ), και προαιρετικά μια είσοδο μηδενισμού (TRST). Ο TAP περιλαμβάνει έναν καταχωρητή εντολών (IR) που καθορίζει ποιες ενέργειες θα εκτελεστούν κατά τον έλεγχο.

5.4.1 Platform FPGAs

Είναι μια σχετικά πρόσφατη κατηγορία τσιπ που συνδυάζει διαφορετικούς τύπους προγραμματιζόμενων συστατικών. Ένα FPGA αυτού του τύπου έχει όλα τα απαραίτητα συστατικά για να υλοποιηθεί άμεσα ένα πλήρες σύστημα και απαιτεί ελάχιστα επιπλέον εξαρτήματα. Το περιεχόμενο καθενός τέτοιου FPGA εξαρτάται από τον εκάστοτε κατασκευαστή και από το κάθε μοντέλο του ίδιου κατασκευαστή.

5.5 Προγραμματισμός FPGAs

Στο αντίστοιχο παράρτημα, βρίσκεται ο κώδικας στην γλώσσα VHDL που υλοποιεί την λογική που είναι απαραίτητη για, την λειτουργία του ASIC, την λήψη των δεδομένων από αυτό και για την αμφίδρομη επικοινωνία με την πλακέτα Π3, όπου βρίσκεται το δεύτερο FPGA.

6 Μικροελεγκτής 8051 με ενσωματωμένο πρωτόκολλο USB

Για την διασύνδεση του συστήματος ελέγχου και μέτρησης δεδομένων επιλέχθηκε το πρωτόκολλο USBv2.0 υλοποιημένο στο ολοκληρωμένο CY7C68013A της Cypress Semiconductors. Το συγκεκριμένο ολοκληρωμένο είναι ένας αναβαθμισμένος μικροελεγκτής 8051 και διαθέτει ενσωματωμένο το πρωτόκολλο USBv2.0. Περιέχει επίσης διάφορα πρωτόκολλα επικοινωνίας για την διασύνδεση με τα περιφερειακά, που στην περίπτωση μας είναι το FPGA της πλακέτας Π3 και διαμέσου αυτού επικοινωνεί με το FPGA της πλακέτας Π2, αλλά και με τα λοιπά περιφερειακά, όπως τους μετατροπείς ψηφιακού σήματος σε αναλογικό. Ο μέγιστος ρυθμός μεταφοράς δεδομένων από και προς τον ηλεκτρονικό υπολογιστή είναι 53Mbytes ανά δευτερόλεπτο, το μέγιστο δηλαδή που υποστηρίζει το συγκεκριμένο πρωτόκολλο. Αυτό επιτυγχάνεται εξαιτίας της υλοποίησης του usb σε υλικό, χωρίς δηλαδή να διαμεσολαβεί ο μικροελεγκτής στην μεταφορά των δεδομένων. Ο μικροελεγκτής χρησιμεύει για τον καθορισμό των παραμέτρων της επικοινωνίας, αλλά και για την διεπαφή με τα περιφερειακά κυκλώματα.

6.1 Μικροελεγκτής 8051

Ο μικροελεγκτής διαθέτει 256 bytes μνήμης καταχωρητών, εκτεταμένο σύστημα διακοπής (interrupts), τρεις χρονιστές/μετρητές και δύο USARTs. Ο χρονισμός του κυκλώματος γίνεται με έναν κρύσταλλο 24MHz και εσωτερικά αναλαμβάνει τον πολλαπλασιασμό του ρολογιού αυτού μέχρι τα 480MHz. Ο 8051 για την εφαρμογή που μελετάται χρονίστηκε στα 48MHz με ρύθμιση του κατάλληλου καταχωρητή.

Το USB είναι ένας ενσύρματος δίαυλος που υποστηρίζει ανταλλαγή δεδομένων μεταξύ του υπολογιστή (Host) και των περιφερειακών συσκευών. Τα συνδεδεμένα περιφερειακά διαμοιράζονται το εύρος ζώνης μέσω ενός πρωτοκόλλου βασιζόμενου σε κουπόνια. Ο host είναι υπεύθυνος για την δρομολόγηση των δεδομένων στον δίαυλο. Το πρωτόκολλο επιτρέπει την ρύθμιση, την αποσύνδεση και σύνδεση περιφερειακών ενώ ο host και αλλά περιφερειακά βρίσκονται σε λειτουργία.

Το σύστημα αποτελείται από τρία μέρη, την διασύνδεση, της συσκευές και τον host. Διασύνδεση USB είναι ο τρόπος με τον οποίο τα περιφερειακά συνδέονται και επικοινωνούν με τον host. Η τοπολογία της διασύνδεσης έχει την μορφή βαθμίδων αστέρα , όπου στο

κέντρο κάθε αστέρα βρίσκεται ένα hub. Κάθε περιφερειακό συνδέεται με τον host είτε άμεσα, είτε μέσω ενός ή περισσότερων hub. Στην μεριά του υπολογιστή υπάρχει μόνο ένας host ο οποίος αποτελείται από υλικό και λογισμικό. Οι περιφερειακές συσκευές είναι είτε διακλαδωτές που παρέχουν επιπρόσθετα σημεία σύνδεσης στον δίαυλο, είτε συσκευές που πραγματοποιούν κάποια εργασία και πρέπει να διασυνδεθούν με τον ηλεκτρονικό υπολογιστή.

Οι συσκευές πρέπει να είναι συμβατές με το πρωτόκολλο και να το υλοποιούν. Απαραίτητες λειτουργίες είναι ή ρύθμιση και επανεκκίνηση, και ή παροχή στον host πληροφοριών για το τη είδους συσκευές είναι και για τον τρόπο επικοινωνίας.

Η κατεύθυνση μιας μεταφοράς δεδομένων είναι είτε από το περιφερειακό προς τον υπολογιστή, είτε από τον υπολογιστή προς το περιφερειακό. Οι μεταφορές ξεκινούν με την αποστολή ενός κουπονιού από τον υπολογιστή. Το κουπόνι περιγράφει το είδος και την κατεύθυνση της μεταφοράς και επίσης την διεύθυνση της συσκευής και τον αριθμό του τερματικού σημείου (EndPoint) της συσκευής που ανταλλάσει δεδομένα με τον υπολογιστή. Κάθε συσκευή υποστηρίζει πολλαπλά τερματικά σημεία, το οποία αποτελούν της θύρες επικοινωνίας στης συσκευής με τον υπολογιστή μέσω του διαύλου. Οι συσκευές λαμβάνουν το κουπόνι και διαβάζουν το πεδίο διεύθυνσης. Η συσκευή που θα αναγνωρίσει την διεύθυνση της ετοιμάζεται για άμεση επικοινωνία με τον υπολογιστή. Οι διευθύνσεις του πομπού και του δεκτή των προς μεταφορά δεδομένων περιλαμβάνεται στο κουπόνι. Η όλη αυτή διαδικασία χρησιμοποιεί ένα αμφίδρομο τερματικό σημείο το οποίο ονομάζεται, τερματικό σημείο ελέγχου και είναι το μοναδικό αμφίδρομο τερματικό σημείο. Ο πομπός λοιπών στέλνει τα δεδομένα εάν πρέπει να τα μεταφέρει, ο δέκτης τα λαμβάνει και στέλνει πίσω στον αποστολέα ένα πακέτο χειραψίας για να τον ενημερώσει εάν ή μεταφορά ήταν επιτυχής ή όχι. Η παραπάνω διαδικασία ονομάζεται συναλλαγή και περιλαμβάνει τα πακέτα κουπόνι - δεδομένα - χειραψία.

6.2 Ρύθμιση του συστήματος

Το USB υποστηρίζει την σύνδεση και αποσύνδεση των συσκευών οποιαδήποτε στιγμή, έτσι ο υπολογιστής διαβάζει την κατάσταση των θυρών διασύνδεσης και εάν ανιχνεύσει μια σύνδεση νέας συσκευής ενεργοποιεί την θύρα και καθορίζει μια διεύθυνση για την συσκευή και έναν δίαυλο επικοινωνίας με την συγκεκριμένη συσκευή. Ο δίαυλος ελέγχου χρησιμοποιεί την

διεύθυνση της συσκευής και το τερματικό σημείο ελέγχου (ή αλλιώς το μηδενικό τερματικό σημείο) και μέσω αυτού ρυθμίζει την συσκευή.

6.3 Αποσύνδεση συσκευής

Όταν μια συσκευή αποσυνδεθεί το hub απενεργοποιεί την θύρα και ενημερώνει τον host, ώστε να διαχειριστεί την αποσύνδεση της συσκευής από τον δίαυλο. Η διαδικασία αυτή μαζί με την απόδοση διεύθυνσης ονομάζεται απαρίθμηση του διαύλου (Enumeration).

6.4 Τύποι μεταφοράς δεδομένων

Η μεταφορά δεδομένων γίνεται ανάμεσα στο λογισμικό του host και ενός συγκεκριμένου τερματικού σημείου της συσκευής.

ο usb διαθέτει τέσσερα είδη μεταφοράς δεδομένων:

- Μεταφορές ελέγχου: Χρησιμοποιούνται για την ρύθμιση μιας συσκευής κατά την σύνδεση της.
- Ογκώδεις μεταφορές: Μεταφέρουν μεγάλες ποσότητες δεδομένων που δεν είναι απαραίτητο να μεταφερθούν γρήγορα.
- Μεταφορές Διακοπής: Μεταφέρουν μικρές ποσότητες δεδομένων τα οποία πρέπει να μεταφερθούν με συγκεκριμένους χρονικούς περιορισμούς (για παράδειγμα πληκτρολόγια κλπ).
- Ισόχρονες μεταφορές: Χρησιμοποιούνται για την μεταφορά δεδομένων πραγματικού χρόνου και παρέχουν σταθερό ρυθμό μεταφοράς δεδομένων μέσω της δέσμευσης σταθερού εύρους ζώνης.

Όλοι οι παραπάνω τύποι απαιτούν ανίχνευση και διόρθωση σφαλμάτων εκτός από τις ισόχρονες μεταφορές καθώς πρέπει να υποστηρίζουν μεταφορά σε πραγματικό χρόνο.

- Το usb διαθέτει τέσσερα είδη μεταφοράς δεδομένων:
  Μεταφορές ελέγχου: Χρησιμοποιούνται για την ρύθμιση μιας συσκευής κατά την σύνδεση της.
- Ογκώδεις μεταφορές: Μεταφέρουν μεγάλες ποσότητες δεδομένων που δεν είναι απαραίτητο να μεταφερθούν γρήγορα.

- Μεταφορές Διακοπής: Μεταφέρουν μικρές ποσότητες δεδομένων τα οποία πρέπει να μεταφερθούν με συγκεκριμένους χρονικούς περιορισμούς (για παράδειγμα πληκτρολόγια κλπ).
- Ισόχρονες μεταφορές: Χρησιμοποιούνται για την μεταφορά δεδομένων πραγματικού χρόνου και παρέχουν σταθερό ρυθμό μεταφοράς δεδομένων μέσω της δέσμευσης σταθερού εύρους ζώνης.

Όλοι οι παραπάνω τύποι απαιτούν ανίχνευση και διόρθωση σφαλμάτων εκτός από τις ισόχρονες μεταφορές καθώς πρέπει να υποστηρίζουν μεταφορά σε πραγματικό χρόνο.

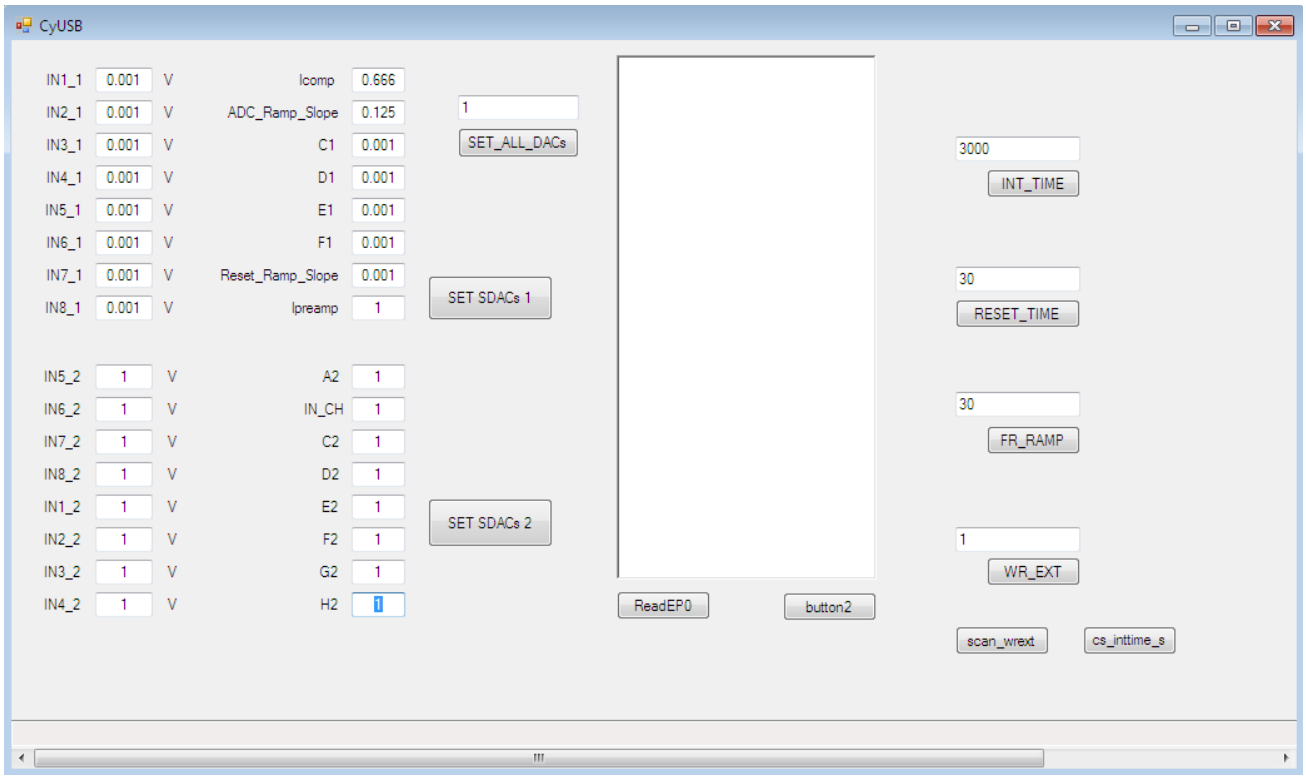
Για την συγκεκριμένη εφαρμογή επιλέχτηκε μεταφορά δεδομένων τύπου Bulk (ογκώδης μεταφορά) παρά το γεγονός ότι η ποσότητα της πληροφορίας που πρέπει να μεταφερθεί δεν είναι καθόλου ογκώδης. Ουσιαστικά η μέγιστη ποσότητα δεδομένων είναι 16pixel x 8bit/pixel + 4x8bit για τα δεδομένα και την διεύθυνση αντίστοιχα. Συνολικά έχουμε λοιπόν 160 bit ωφέλιμης πληροφορίας ανά πλαίσιο μεταφοράς (δηλαδή ανά μετατροπή αναλογικού σήματος σε ψηφιακό. Αυτό σημαίνει ότι επειδή ένα παράθυρο ολοκλήρωσης του προς μέτρηση σήματος έχει διάρκεια της τάξης μερικών δεκάδων us (~100usec) προκύπτει ποσότητα δεδομένων 10.000frames x 160bit/frame = 1.600.000bit/sec που είναι ίσο με 200kByte/sec. Η ποσότητα αυτή ακόμα και αν τις προσθέσεις ορισμένα bit για σκοπούς ελέγχου παραμένει ελάχιστη σε σχέση με τα 56MByte/sec που είναι η μέγιστη ταχύτητα μεταφοράς του usb διαύλου.

Ο λόγος για τον οποίο χρησιμοποιήθηκε η τεχνολογία αυτή, ενώ δεν έχουμε τόσο μεγάλη ποσότητα πληροφορίας είναι καθώς σε περίπτωση που το ολοκληρωμένο δεν ήταν σχεδιασμένο απλά για ερευνητικούς σκοπούς, αλλά προοριζόταν για κάποια εφαρμογή, θα περιείχε μερικές χιλιάδες pixel, αντί για 16 και τότε η ποσότητα της πληροφορίας θα ήταν παρά πολύ μεγάλη ακόμα και για τον δίαυλο USB.

7 Σχολιασμός Κώδικα και Μετρήσεις

7.1 Γραφικό Περιβάλλον (GUI)

Το γραφικό περιβάλλον, για την σχεδίαση του οποίου επιλέχθηκε η γλώσσα προγραμματισμού C# (C sharp), παρουσιάζεται στην Εικόνα 7-1 Γραφικό περιβάλλον ελέγχου και λήψης μετρήσεων.



Εικόνα 7-1 Γραφικό περιβάλλον ελέγχου και λήψης μετρήσεων

Η συγκεκριμένη, υψηλού επιπέδου γλώσσα ήταν η καλύτερη επιλογή για την εφαρμογή που μελετάμε καθώς δίνει την δυνατότητα για σχεδιασμό γραφικού περιβάλλοντος και επίσης η κατασκευάστρια εταιρία του μικροελεγκτή USB παρέχει συναρτήσεις για την επικοινωνία του με την γλώσσα αυτή.

Για την καλύτερη κατανόηση του κώδικα συνολικά είναι χρησιμότερο ο σχολιασμός να αρχίσει από το γραφικό περιβάλλον. Ο χρήστης εισάγει από το γραφικό περιβάλλον στα αντίστοιχα κελιά της τιμές τάσης που επιθυμεί. Υπάρχουν 32 κελιά, ένα για κάθε κανάλι ένα για κάθε κανάλι των τεσσάρων οκτα-κάναλων DAC της πλακέτας Π2. Τα δεκαέξι από αυτά

αφορούν την ρύθμιση των ρευμάτων που προσφέρουν οι πηγές ρεύματος στα δεκαέξι κανάλια (εισόδους) του ολοκληρωμένου. Τα κελιά Icomp και Ipreamp είναι οι τάσεις που ελέγξουν της πηγές ρεύματος για την πόλωση του ολοκληρωμένου. Οι τάσεις που εισάγονται μετατρέπονται στην αντίστοιχη 12bit λέξη και σε αυτήν προστίθενται τα υπόλοιπα bit ελέγχου για τον προγραμματισμό των DAC, που είναι το είδος της λειτουργίας και η διεύθυνση του καναλιού του DAC που θα λάβει την συγκεκριμένη τιμή. Η πληροφορία αυτή χωρίζεται σε τρείς λέξεις των 8bit και μεταφέρεται στον μικροελεγκτή USB.

Το κελί ADC_RAMP_SLOPE ρυθμίζει την κλίση της ράμπας τάσης στην είσοδο του συγκριτή. Τα υπόλοιπα κανάλια δεν χρησιμοποιούνται. Όταν ο χρήστης εισάγει της επιθυμητές τιμές ενεργοποιεί το αντίστοιχο κουμπί: SET_SDACs1 και SET_SDACs2 για να στείλει τα δεδομένα στον μικροελεγκτή USB, διαμέσου του διαύλου και έπειτα αυτά σειριοποιούνται με την βοήθεια κώδικα που είναι γραμμένος στον μικροελεγκτή. Η σειριακή αυτή πληροφορία φθάνει τελικά στους μετατροπείς διαμέσου των δύο FPGAs.

Επίσης ο χρήστης μπορεί να ρυθμίσει την διάρκεια των ακόλουθων χρονικών περιόδων:

1. Integration Time που αφορά την διάρκεια της ολοκλήρωσης του εισερχόμενου ρεύματος στους ενισχυτές φορτίου του ASIC.
2. RESET TIME που είναι η διάρκεια των παλμών εκκαθάρισης των πυκνωτών που ολοκληρώνουν το ρεύμα , στην ανάδραση των ενισχυτών διεμπέδησης.
3. Freeze Ramp, είναι η διάρκεια του παγώματος της ράμπας στην μέγιστη της τιμή.
4. WR_EXT είναι η διάρκεια που μένει ενεργοποιημένος (HIGH) ο παλμός wr_ext.

Ο παλμός wr_ext αποτελεί την είσοδο σε ένα κύκλωμα δοκιμής που αποτελείται από το ψηφιακό μόνο τμήμα του pixel. Η είσοδος αυτή οδηγείται από την έξοδο του συγκριτή κατά την κανονική λειτουργία. Όλοι οι παραπάνω χρόνοι είναι ακέραιοι αριθμοί και συμβολίζουν την διάρκεια σε κύκλους ρολογιού του εκάστοτε παλμού. Εφόσον έχουν ρυθμιστεί όλες οι παράμετροι αρκεί να πατηθεί το κουμπί ReadEP0 για να ληφθούν τα τρέχοντα δεδομένα από το ASIC, να απεικονιστούν στο γραφικό περιβάλλον και να αποθηκευθούν σε ένα αρχείο κειμένου.

Μια επιπλέον αυτόματη λειτουργία λήψης δεδομένων γίνεται με την ενεργοποίηση του κουμπιού wr_ext η οποία θα περιγράφει στην παράγραφο που περιλαμβάνει τις μετρήσεις.

7.2 Περιγραφή Κώδικα USB

Γραμμή 340: Υπορουτίνα SET INTEGRATION TIME:

Λαμβάνει τα δεδομένα από το κελί του γραφικού περιβάλλοντος και αφού πρώτα ειδοποιήσει το FPGA της πλακέτας Π2, κατεβάζει τα δεδομένα αυτά στην αρτηρία μεταφοράς δεδομένων FD_BUS. Μετά την ολοκλήρωση της μεταφοράς, αδρανοποιεί και πάλι την αρτηρία.

Με πανομοιότυπο τρόπο γίνεται η μεταφορά των δεδομένων των κελιών του GUI:

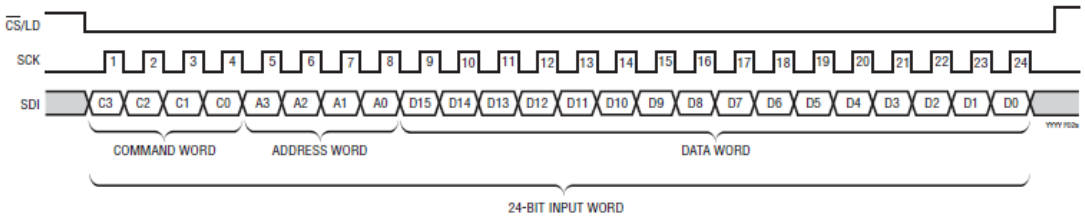
1. SET RESET TIME
2. SET FREEZE RAMP TIME και
3. WR_EXT TIME της πλακέτας Π2.

7.2.1 Ανάγνωση δεδομένων

Με την ενεργοποίηση του κουμπιού Read_EP0 στέλνεται μια εντολή για την λήψη των δεδομένων στο FPGA της πλακέτας Π2. Τα δεδομένα διέρχονται από τα δύο FPGA και από τον μικροελεγκτή και εγγράφονται σε ένα αρχείο κειμένου με κατάλληλη μορφοποίηση.

7.2.2 Σειριακός προγραμματισμός LTC2620 DAC

Γραμμή 246: Η υπορουτίνα VX_B5 στον μικροελεγκτή λαμβάνει τα δεδομένα από το γραφικό περιβάλλον και τα αποθηκεύει στο τερματικό σημείο μηδέν (EndPoint0). Έπειτα τα σειριοποιεί και τα στέλνει στο DAC διαμέσου των δύο FPGA, σύμφωνα με τις απαιτήσεις του LTC2620.



Εικόνα 7-2 .Διάγραμμα χρονισμού LTC2620

Για να εκκινήσει την διαδικασία γίνεται LOW το σήμα CSLD (chip select) και τοποθετούνται τα bit προς μεταφορά ένα - ένα στην αρτηρία. Μόλις τοποθετηθεί το κάθε bit στην αρτηρία, γίνεται HIGH το SCK και στην αιχμή ανόδου του κάθε σήματος εγγράφεται η τρέχουσα τιμή του SDI στον καταχωρητή του DAC. Ο καταχωρητής του DAC μετατοπίζει τα δεδομένα που δέχεται, μέχρι να πάρει ολόκληρη την πληροφορία. Μόλις ολοκληρωθεί η μεταφορά ο μικροελεγκτής κάνει HIGH το σήμα CSLD και τα δεδομένα που βρίσκονται στον καταχωρητή μετατόπισης του DAC φορτώνονται και ξεκινάει η μετατροπή της λέξης σε αναλογική τάση. Η παραπάνω ρουτίνα επαναλαμβάνεται και για το δεύτερο ζευγάρι DAC που βρίσκεται στην πλακέτα Π2.

Ο υπόλοιπος κώδικας του USB μικροελεγκτή χρησιμοποιείται για την ρύθμιση του ρολογιού λειτουργίας του μικροελεγκτή, την ρύθμιση των τερματικών σημείων και για την αρχικοποίηση των υπολοίπων καταχωρητών του USB.

7.3 Προγραμματισμός FPGA

7.3.1 FPGA1 (Π2)

Στο FPGA1 η υλοποίηση έχει γίνει με την γλώσσα περιγραφής υλικού VHDL. Στην σελίδα 1 περιέχονται οι ορισμοί των βιβλιοθηκών που χρησιμοποιούνται και οι ορισμοί των θυρών του ψηφιακού κυκλώματος που υλοποιείται. Ορίζονται τα σήματα που συνδέονται με το ολοκληρωμένο προς μέτρηση, τα σήματα που συνδέονται με τα σειριακά DAC και τέλος τα σήματα επικοινωνίας με το FPGA2 της πλακέτας Π3.

Στην δεύτερη σελίδα έχουμε τον ορισμό και την αρχικοποίηση των εσωτερικών σημάτων του κυκλώματος που υλοποιείται.

Στην τρίτη σελίδα έχουμε την διεργασία (process), η οποία αποτελείται από ακολουθιακή λογική και υλοποιεί το κύκλωμα που επικοινωνεί με το FPGA2. Λαμβάνει αφενός τα δεδομένα από αυτό για την διάρκεια του χρόνου ολοκλήρωσης και των χρόνων εκκαθάρισης και στέλνει τα δεδομένα όταν ζητηθεί από τον USB μικροελεγκτή. Επιπλέον μετατρέπει τα δεδομένα που είναι κωδικοποιημένα με κώδικα Grey σε δυαδική μορφή, πριν τα αποστείλει στον USB.

Στην σελίδα 4 έχουμε ένα κύκλωμα υποδιαίρεσης του κεντρικού ρολογιού π, που παρέχει ο USB, από 48MHz σε 3MHz. Έτσι η μεταφορά των δεδομένων από και προς τον υπολογιστή, γίνεται με ταχύτητα 48MHz, ενώ το ολοκληρωμένο λειτουργεί στα 3MHz.

Στην σελίδα 5 έχουμε άλλη μια διεργασία ακολουθιακής λογικής που υλοποιεί το κύκλωμα το οποίο υλοποιεί τον κύκλο λειτουργίας του ολοκληρωμένου. Πρώτα αρχικοποιεί τους πυκνωτές ολοκλήρωσης ενεργοποιώντας τα σήματα RESET και RESET2 καθώς και τα ψηφιακά. Μετά από το πέρας χρόνου INTEGRATION TIME κύκλων ρολογιού, ξεκινάει η φάση της μετατροπής του αναλογικού σήματος σε ψηφιακό με την εκκίνηση ανόδου της αναλογικής ράμπας τάσης. Ακολουθεί η μετατροπή και η εγγραφή στην μνήμη, της ψηφιακής λέξης για το κάθε pixel. Έπειτα τα δεδομένα είναι διαθέσιμα στις αντίστοιχες αρτηρίες για ανάγνωση από το FPGA1.

Σελίδα 6: Η διεργασία αυτή διαβάζει τα δεδομένα μόλις αυτά είναι διαθέσιμα στις δύο αρτηρίες. Τα δεδομένα αποθηκεύονται σε έναν πίνακα της γλώσσας VHDL, που δεν είναι τίποτε άλλο παρά ένα σύνολο από καταχωρητές. Η παραπάνω μεταφορά και αποθήκευση, καθώς και η μεταφορά αυτών των δεδομένων στο FPGA2 γίνεται σε κάθε περίοδο ολοκλήρωσης, μόνο εάν ζητηθεί από το γραφικό περιβάλλον. Αυτό γίνεται για να μην είναι δυνατό να επιχειρηθεί ανάγνωση του πίνακα όταν εγγράφονται σε αυτών νέα δεδομένα. Με αυτόν τον τρόπο η ανάγνωση και η εγγραφή του πίνακα διαφέρουν χρονικά.

7.3.2 VHDL κώδικας για το FPGA2 (Π3)

Στις γραμμές 1-25 βρίσκονται οι ορισμοί των βιβλιοθηκών που χρησιμοποιούνται και των θυρών του κυκλώματος, και ακολουθεί ο ορισμός των εσωτερικών σημάτων.

Γραμμές 61-108: Υλοποιείται ένα πεπερασμένο αυτόματο τριών καταστάσεων. Η κατάσταση S0 είναι η αδρανής κατάσταση (IDLE). Στην κατάσταση αυτή το κύκλωμα είναι έτοιμο να στείλει δεδομένα αρχικοποίησης στην πλακέτα Π2 για τον προγραμματισμό των DAC και για τον καθορισμό παραμέτρων του FPGA1 (integration time κτλ). Όταν λάβει εντολή μέσω του σήματος RDY_1 (HIGH) μεταβαίνει στην κατάσταση S1 και παραμένει εκεί για

τους επόμενους 19 κύκλους ρολογιού, ώστε να λάβει τα δεδομένα των pixel και της αντίστοιχες διευθύνσεις. Με το πέρας των 19 κύκλων τα δεδομένα έχουν μεταφερθεί σε καταχωρητές στο FPGA2, οι οποίοι υλοποιούνται και πάλι με μια δομή πίνακα VHDL.

Έπειτα έχουμε μετάβαση στην κατάσταση S2, όπου ξεκινά η μεταφορά από το FPGA2 στον μικροελεγκτή. Όσο το σήμα starttransfer είναι HIGH, η κατάσταση S2 τοποθετεί στην αρτηρία που συνδέει το FPGA2 με τον μικροελεγκτή, τα δεδομένα τύπου πίνακα. Συγκεκριμένα χρησιμοποιεί τον δείκτη wordscounter για να βγάλει κάθε γραμμή του πίνακα καταχωρητή σε διαδοχικές χρονικές περιόδους. Η διεργασία, η οποία ξεκινά από την γραμμή 110 και τελειώνει στην γραμμή 126, αλλάζει την τιμή του δείκτη (μετρητή) wordscounter ώστε το αυτόματο να φορτώσει στην αρτηρία τα κατάλληλα δεδομένα.

Έτσι ο μικροελεγκτής πρέπει να στείλει 19 ανοδικές αιχμές του σήματος ADRESSBUS_IN(6) (rising edges) για να λάβει όλα τα δεδομένα. Μόλις ο μικροελεγκτής στείλει την πρώτη αιχμή υπάρχουν στην αρτηρία τα πρώτα 8bit. Ο μικροελεγκτής τα διαβάζει και τα αποθηκεύει στην δικιά του μνήμη, στην συνέχεια στέλνει μια νέα αιχμή, ώστε να τοποθετηθούν στην αρτηρία τα νέα δεδομένα και συνεχίζει μέχρι να λάβει 8x19 bit.

Αφού τελειώσει αυτή η διαδικασία, το αυτόματο μεταβαίνει και πάλι στην κατάσταση S0 (IDLE). Τέλος ο μικροελεγκτής μόλις λάβει όλα τα δεδομένα τα στέλνει στον υπολογιστή και εκεί εγγράφονται σε ένα αρχείο κειμένου με την κατάλληλη μορφοποίηση.

7.4 Μετρήσεις

Τα πρώτα κυκλώματα του ολοκληρωμένου που ελέγχθηκαν ήταν τα αναλογικά μέσω των ειδικών δομών για το σκοπό αυτό. Μετά από πολλούς επανελέγχους διαπιστώθηκε ότι η έξοδος του ενισχυτή διεμπέδησης δεν ήταν η αναμενόμενη. Έγιναν εποπτικές μετρήσεις για διάφορες τιμές ρευμάτων στην είσοδο του ενισχυτή και έδειξαν ότι δε δούλευε όπως αναμενόταν. Όλες οι πηγές ρεύματος είχαν μετρηθεί ξεχωριστά, με αμπερόμετρο Keithley για χαμηλά ρεύματα και είχαν σωστή συμπεριφορά (γραμμική για ρεύματα από μηδέν έως 250pA). Επίσης συνδέθηκαν πυκνωτές διαφορετικών τιμών στις εισόδους και στάλθηκαν παλμοί τάσης με σκοπό της φόρτιση των πυκνωτών ανάδρασης με συγκεκριμένο φορτίο. Οι ενισχυτές και με αυτόν τον τρόπο δεν άλλαξαν συμπεριφορά. Αποκλείστηκε με αυτό τον τρόπο η περίπτωση να φταίει η συγκεκριμένη είσοδος για τη συμπεριφορά των ενισχυτών.

Επιπλέον ο θόρυβος στις συγκεκριμένες μετρήσεις δεν αποτελεί μεγάλο πρόβλημα καθώς ολοκληρώνεται στους πυκνωτές ανάδρασης μαζί με το ρεύμα εισόδου. Παρόλο λοιπόν που

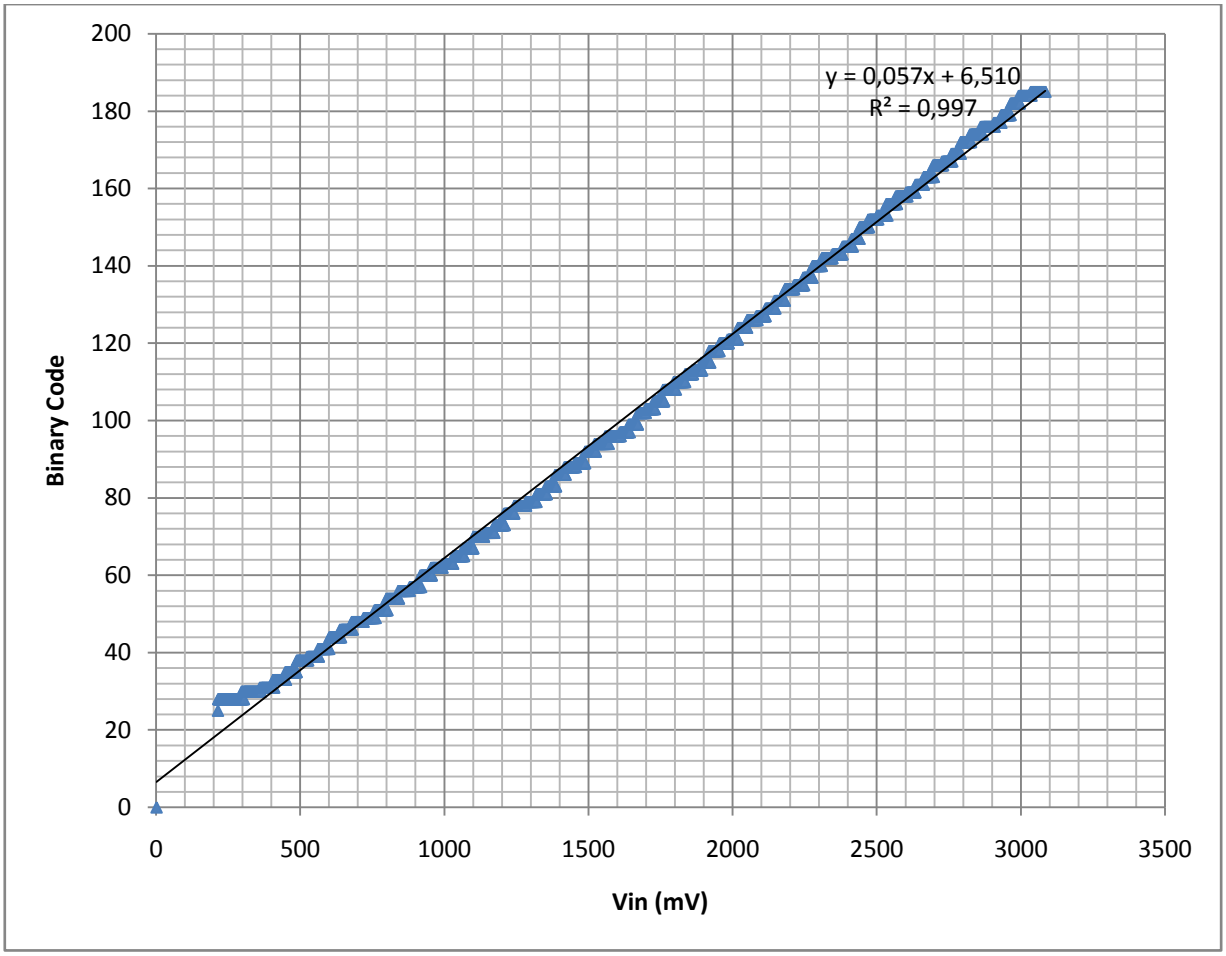
θα είχε επίδραση ο θόρυβος στις μετρούμενες τιμές δεν είναι λογικό να κάνει τους ενισχυτές να έχουν ποιοτικά διαφορετική συμπεριφορά.

Πιθανά αιτία για αυτή την συμπεριφορά των ενισχυτών θα μπορούσε να είναι η έλλειψη απομονωτών (μοναδιαίων ενισχυτών) εντός του ολοκληρωμένου ώστε να οδηγηθούν τα σήματα στο παλμογράφο. Αφού έγινε ο προγραμματισμός των ενσωματωμένων και υπήρξε η δυνατότητα λήψης των ψηφιακών δεδομένων στον ηλεκτρονικό υπολογιστή φάνηκε ότι η ψηφιοποιημένες τιμές των εξόδων των ενισχυτών επίσης δεν ήταν οι αναμενόμενες. Με αυτή τη μέθοδο δεν είναι εύκολη η επιβεβαίωση της λειτουργίας και του συγκεκριμένου σφάλματος καθώς για κάθε κύκλο ολοκλήρωσης λαμβάνουμε μια μόνο τιμή. Ενδεικτικό του ότι δεν είχαμε τις αναμενόμενες εξόδους στους ενισχυτές διεμπέδησης ήταν ότι δεν υπήρξε γραμμικότητα καθώς αυξανόταν η τιμή του ρεύματος στην είσοδο των ενισχυτών. Αυτή η διαδικασία επιβεβαίωσε την εσφαλμένη συμπεριφορά που παρατηρήθηκε με τη βοήθεια του παλμογράφου για τις δομές ελέγχου των αναλογικών κυκλωμάτων. Επομένως, η έλλειψη ισχύος του σήματος δεν μπορεί να είναι η αιτία καθώς το πρόβλημα εξακολουθεί και στην έξοδο των ενισχυτών εσωτερικά του ολοκληρωμένου.

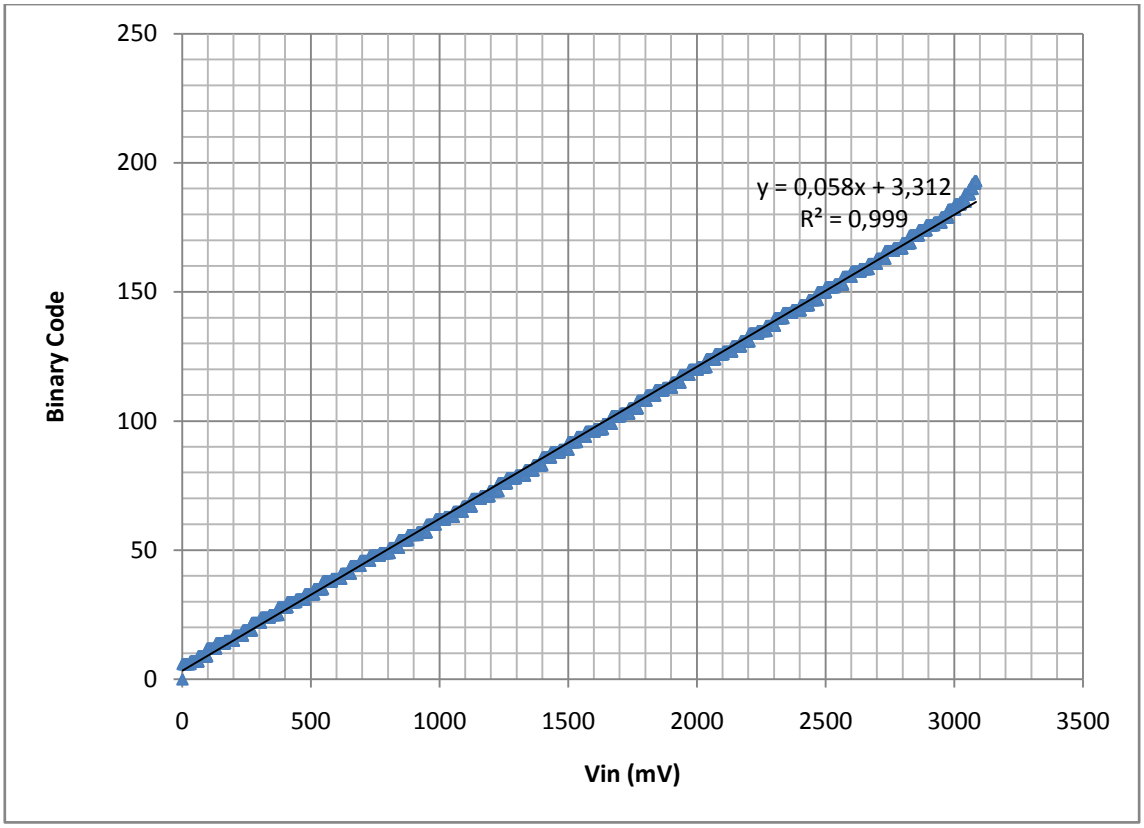
Ένα άλλος παράγοντας λάθους στα ολοκληρωμένα κυκλώματα είναι η ελαττωματική επεξεργασία του δισκίου πυριτίου (Wafer) κατά τη κατασκευή. Για το συγκεκριμένο σχέδιο κατασκευάστηκαν 14 ολοκληρωμένα. Επειδή μετά τον έλεγχο τους παρατηρήθηκε ότι είχαν την ίδια συμπεριφορά αποκλείστηκε και αυτή η αιτία σφάλματος. Η πιθανότητα να έχει γίνει σφάλμα και για τα 14 αντίγραφα δεν μπορεί να ληφθεί υπόψη καθώς είναι εξαιρετικά μικρή. Επίσης δεδομένης τις υπόλοιπης λειτουργικότητας των ψηφιακών κυκλωμάτων δεν δικαιολογείται αυτός ο παράγοντας σφάλματος.

7.4.1 Έλεγχος ψηφιακών κυκλωμάτων

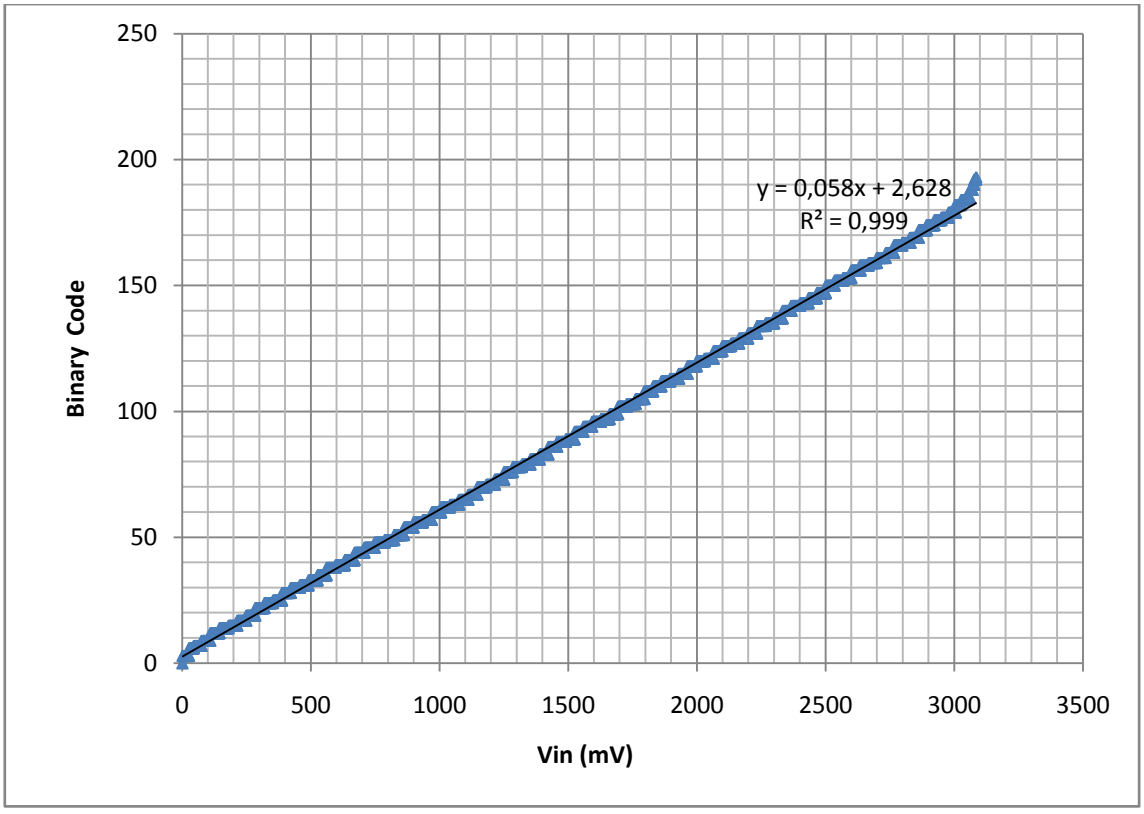
Εφόσον δεν ήταν δυνατή η λήψη μετρήσεων από τους ενισχυτές διεμπέδησης καθώς δεν εμφάνιζαν την αναμενόμενη συμπεριφορά, δόθηκε ως είσοδος στους ενισχυτές μια τάση που προερχόταν από ένα κανάλι των DAC κοινή για όλα τα pixel. Η τάση αυτή εφαρμοζόταν κατευθείαν στην έξοδο των ενισχυτών. Η σάρωση τις τάσης αυτής από τα 0 έως τα 3.3V μας έδωσε διαδοχικές ψηφιοποίησεις που βοήθησαν στη μελέτη του συστήματος μετατροπής του αναλογικού σήματος σε ψηφιακό.


y = 0,057x + 6,510
R² = 0,997
Binary Code
Vin (mV)
0
20
40
60
80
100
120
140
160
180
200
0
500
1000
1500
2000
2500
3000
3500

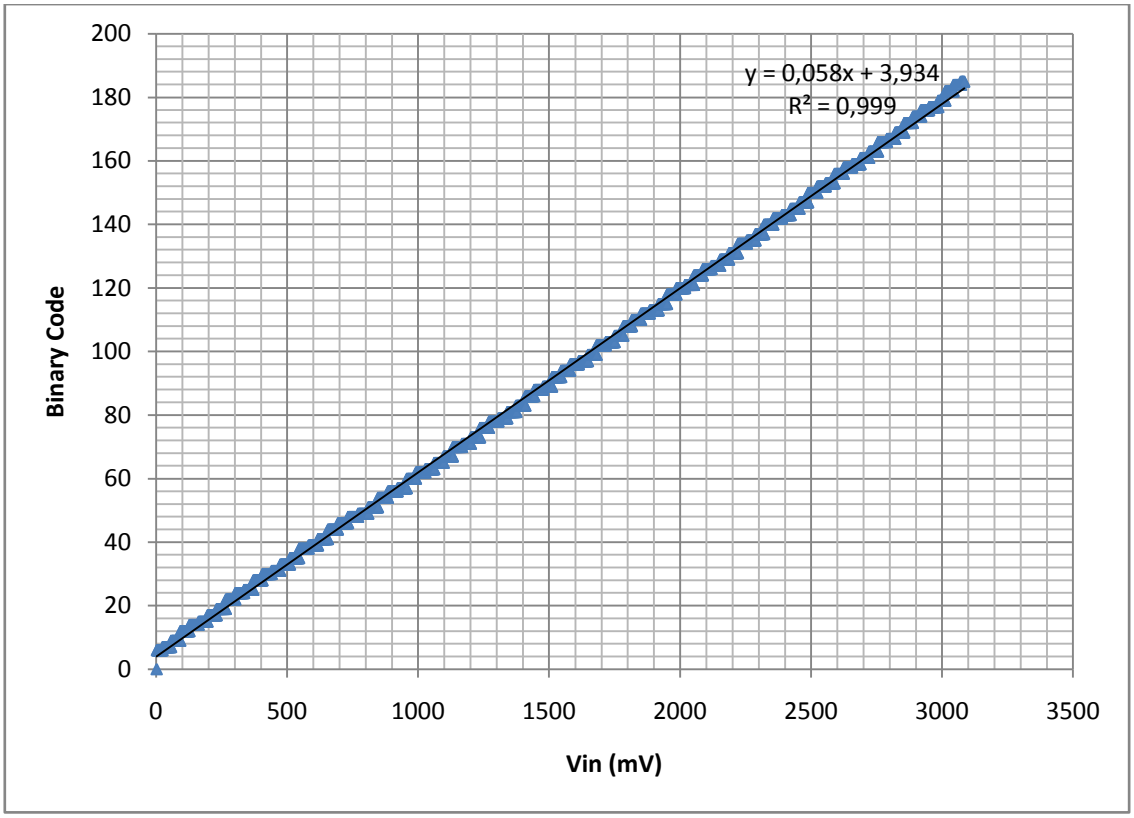
Pixel 1_1



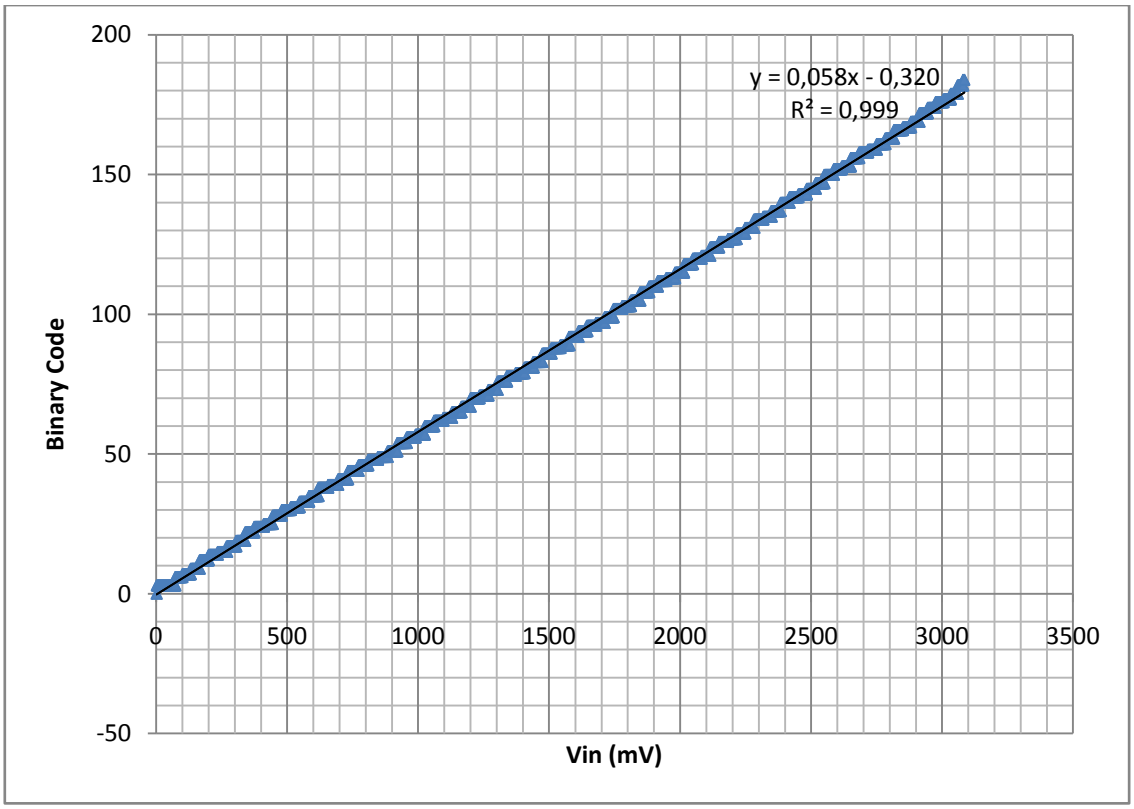
Pixel 1_2



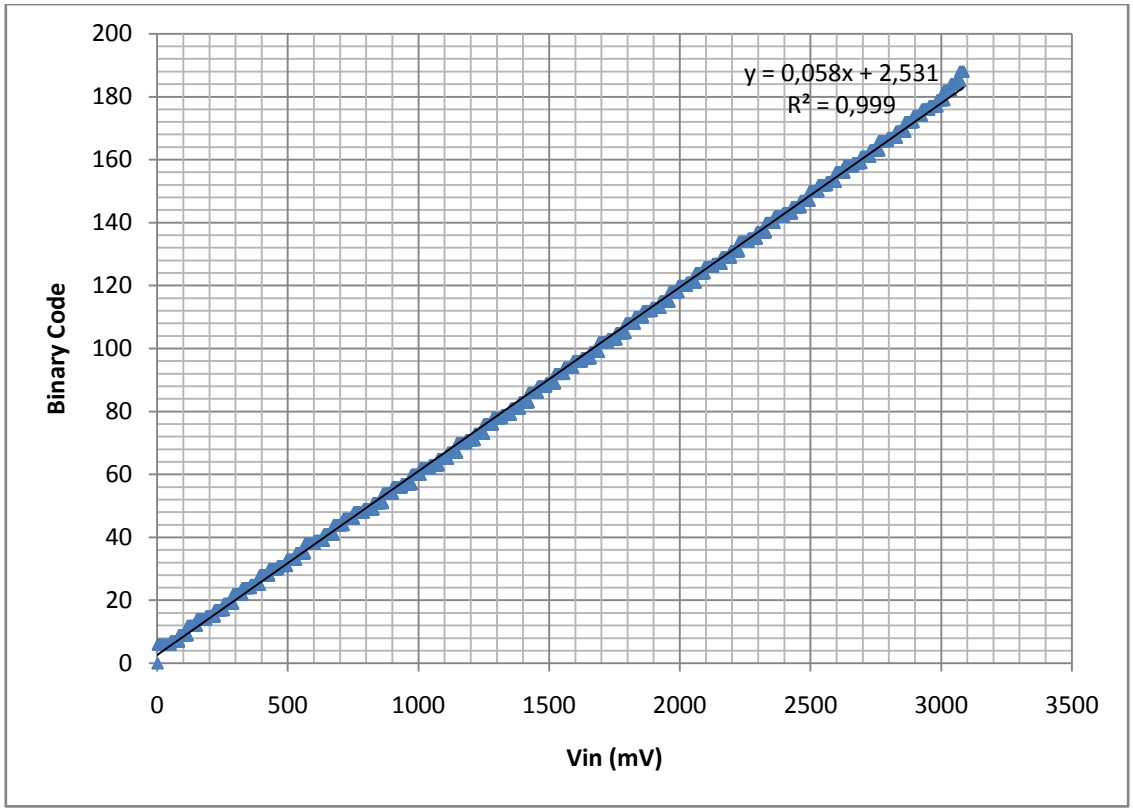
Pixel 1_3



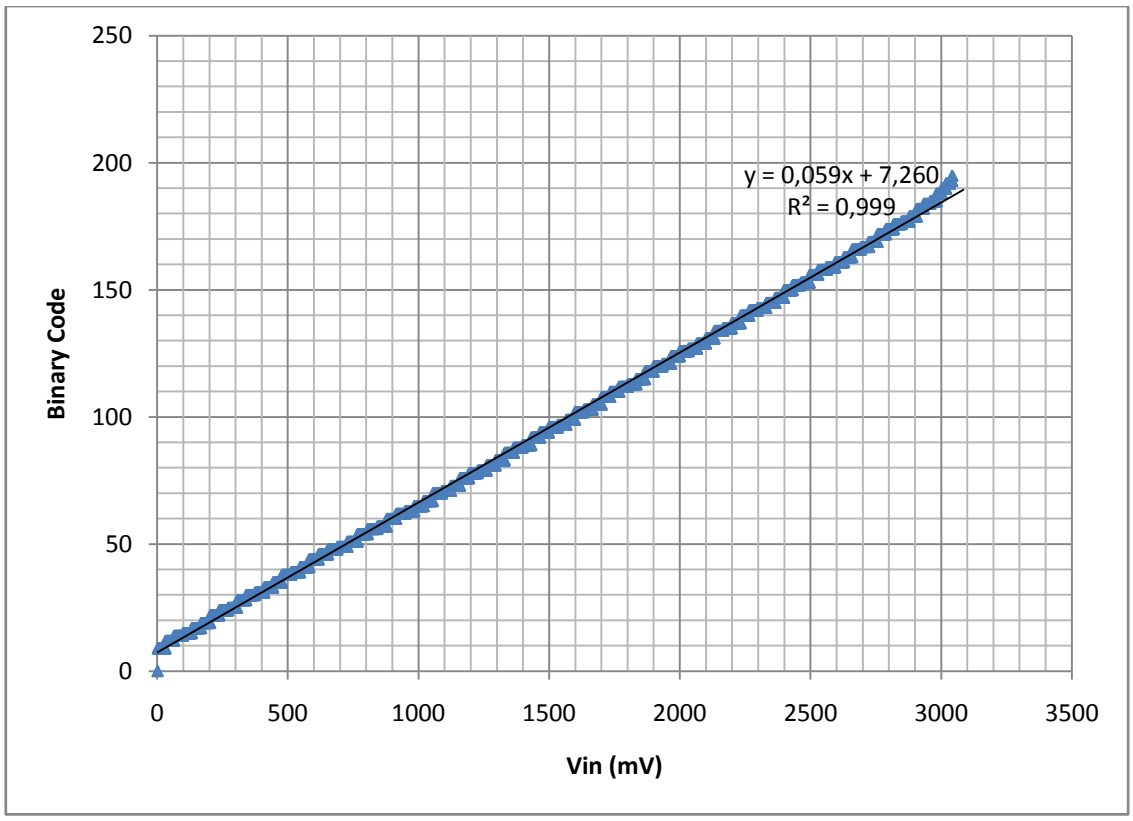
Pixel 1_4



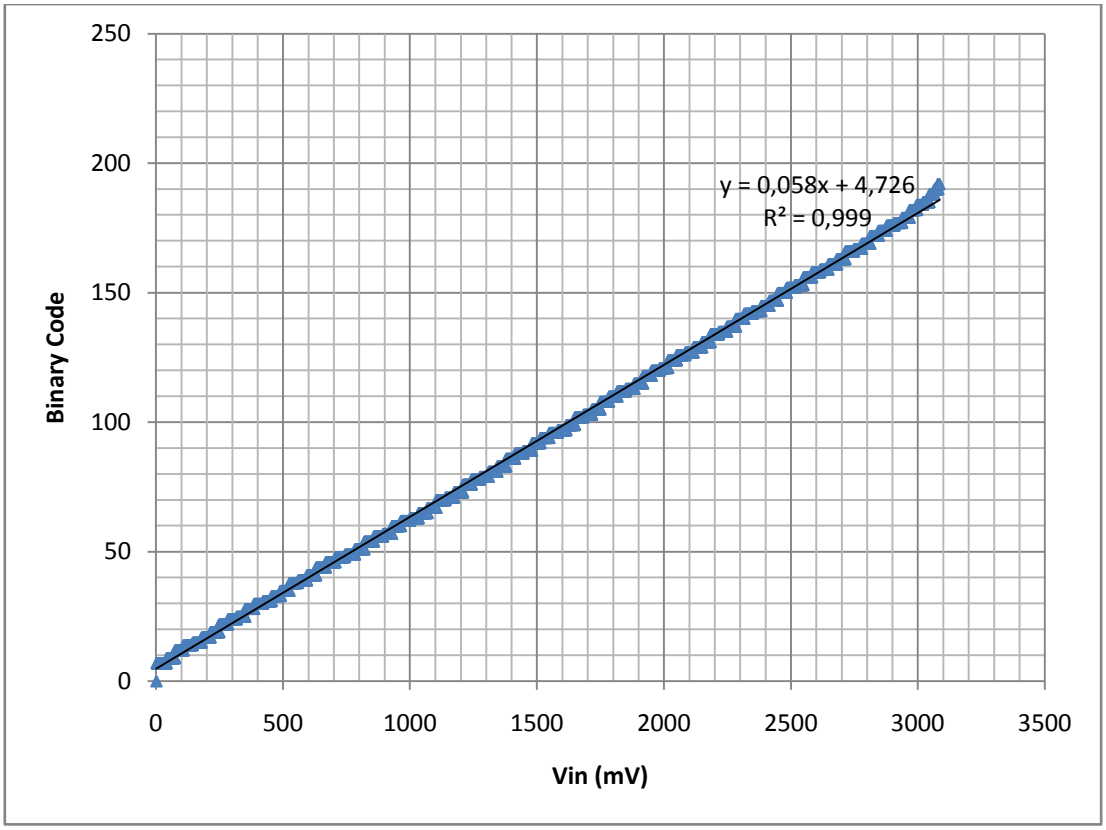
Pixel 1_5



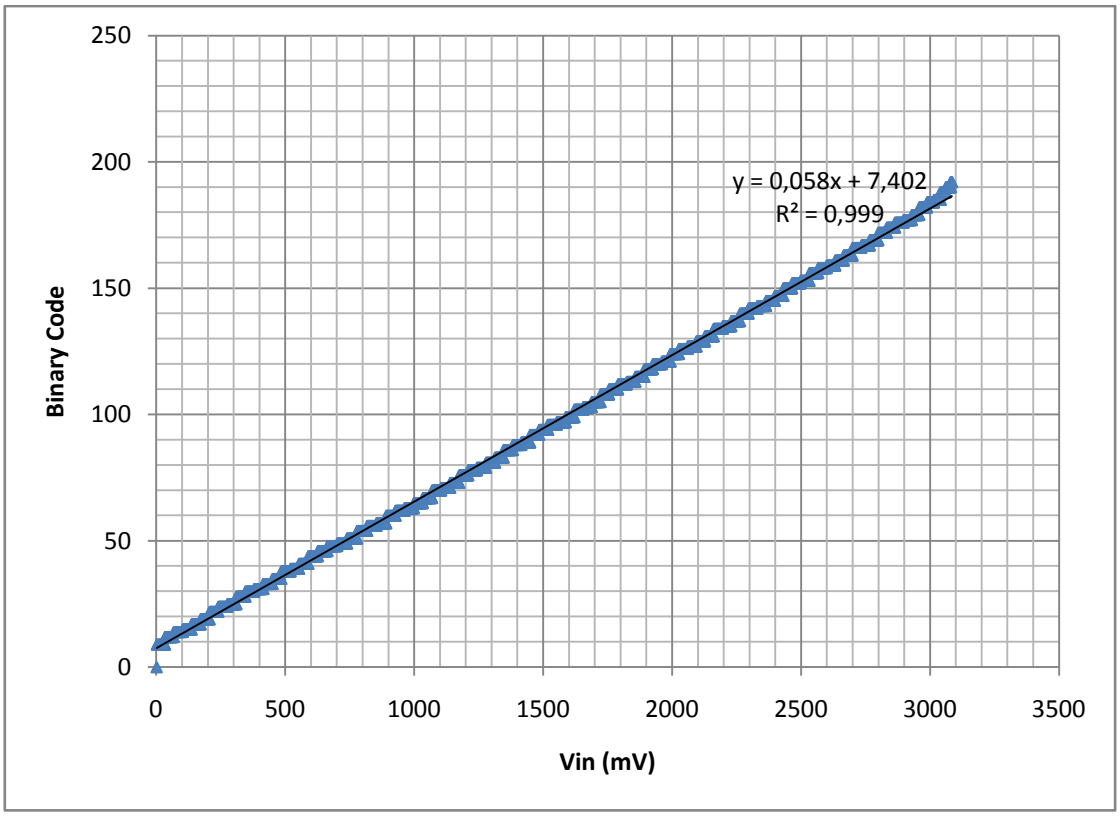
Pixel 1_6



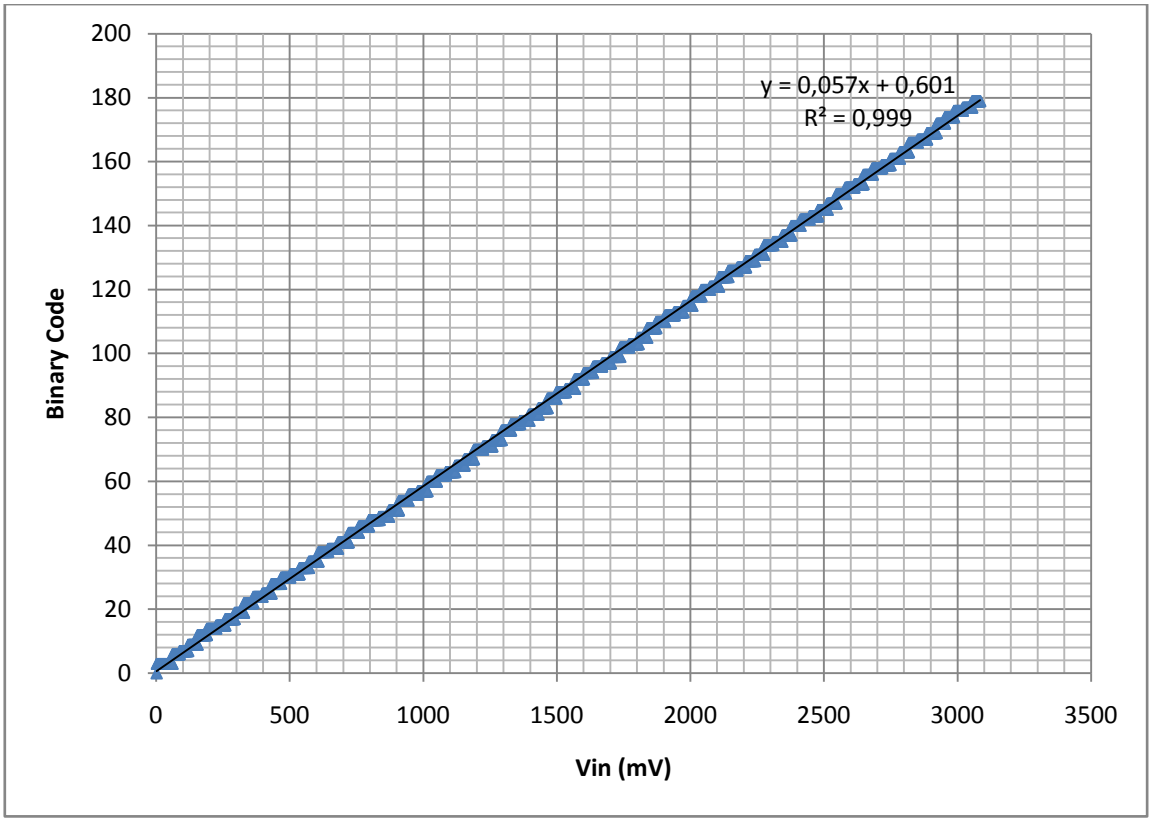
Pixel 1_7



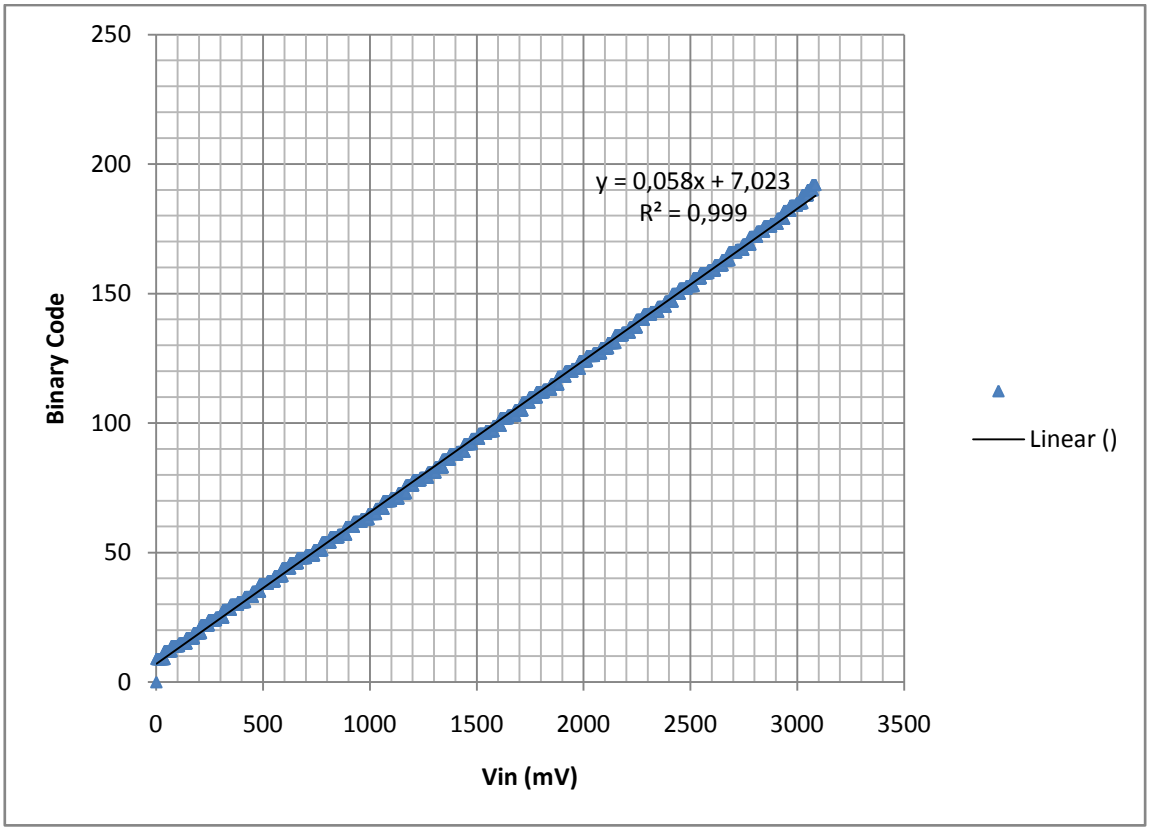
Pixel 1_8



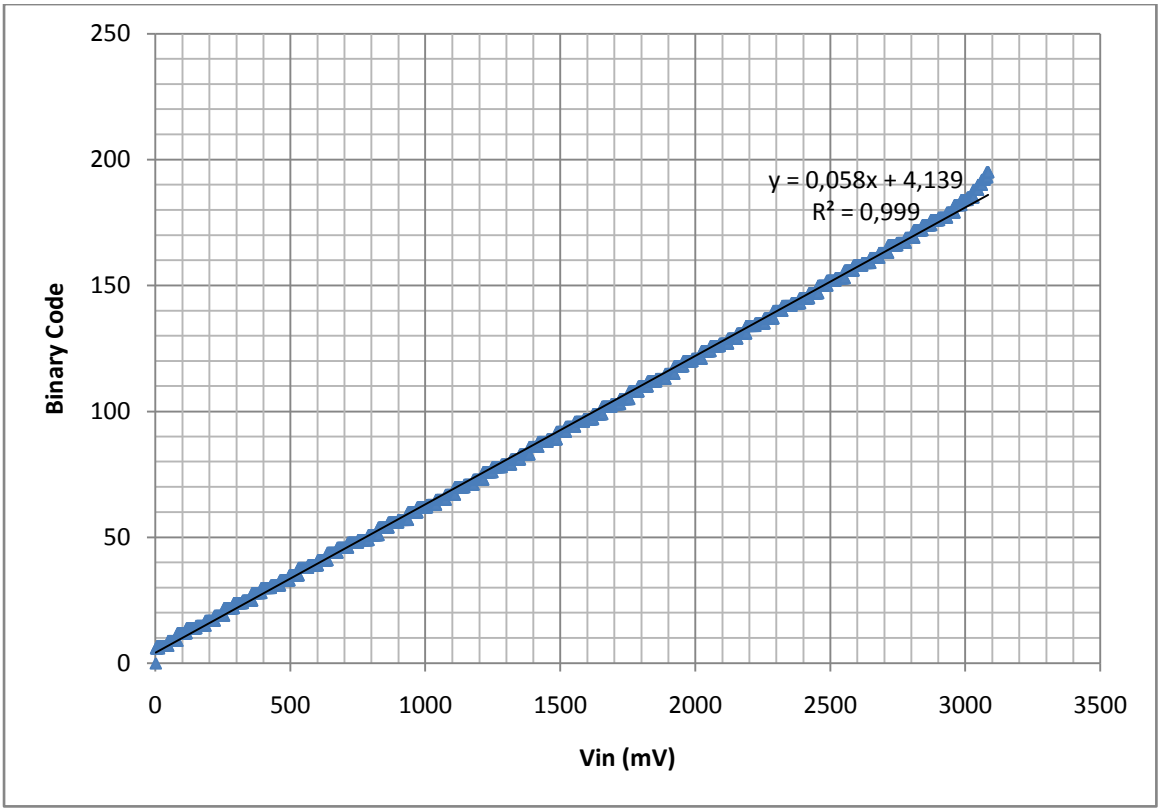
Pixel 2_1



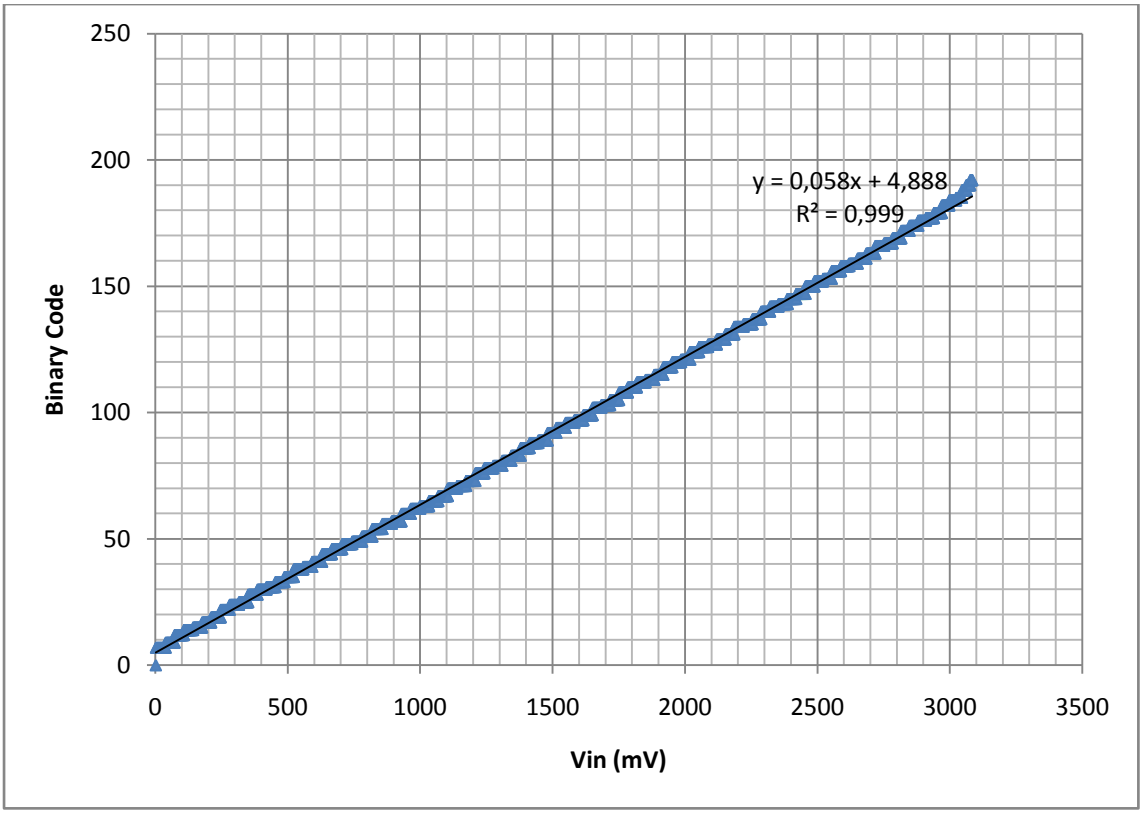
Pixel 2_2



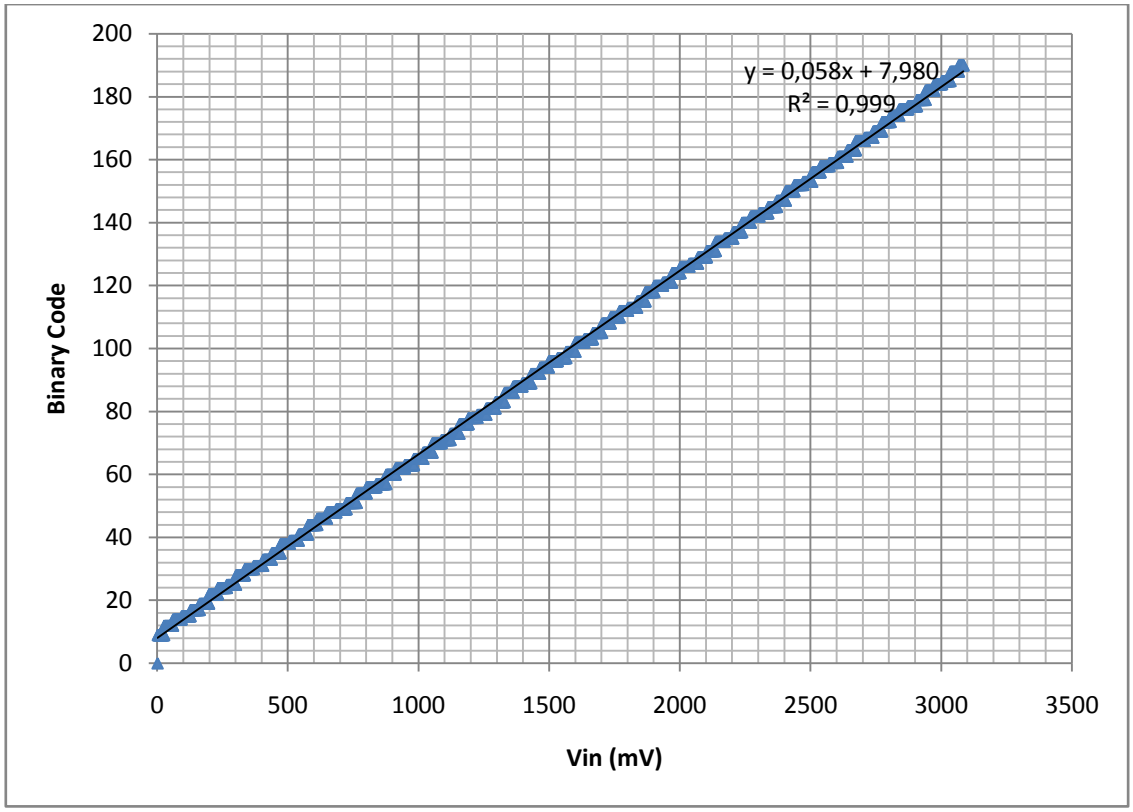
Pixel 2_3



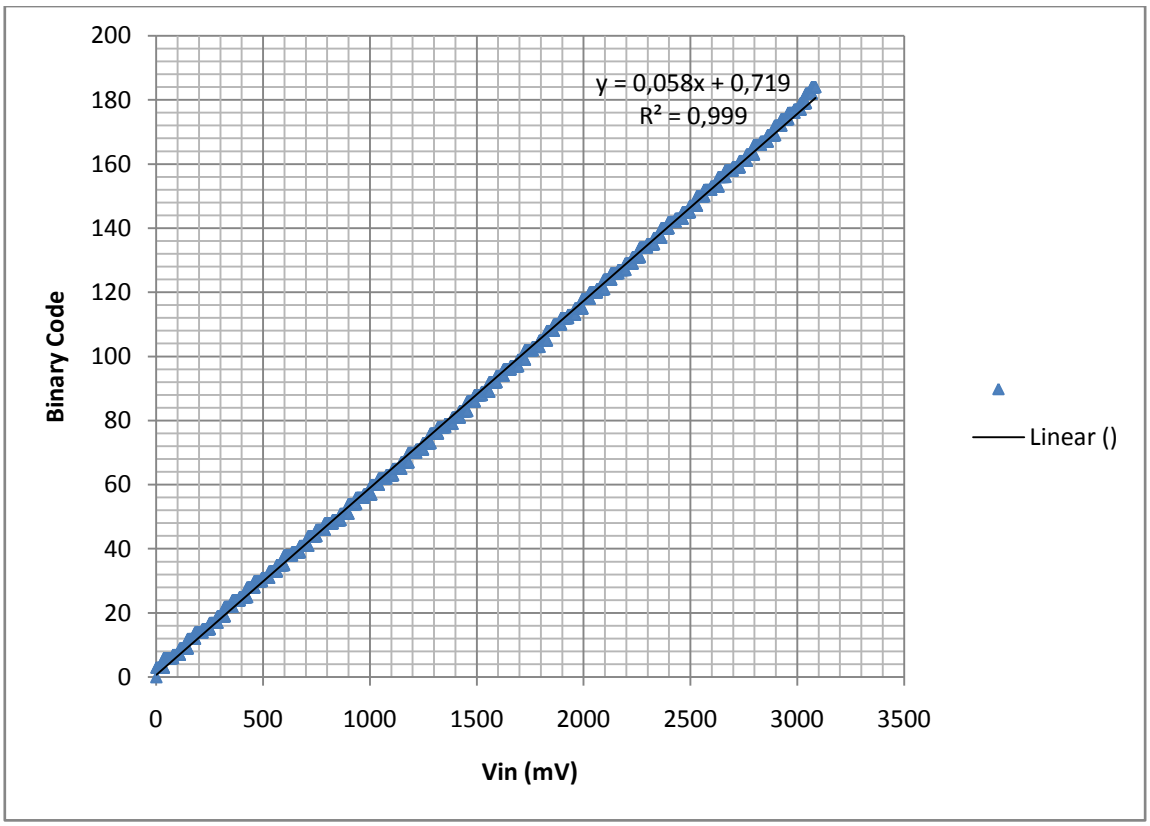
Pixel 2_4



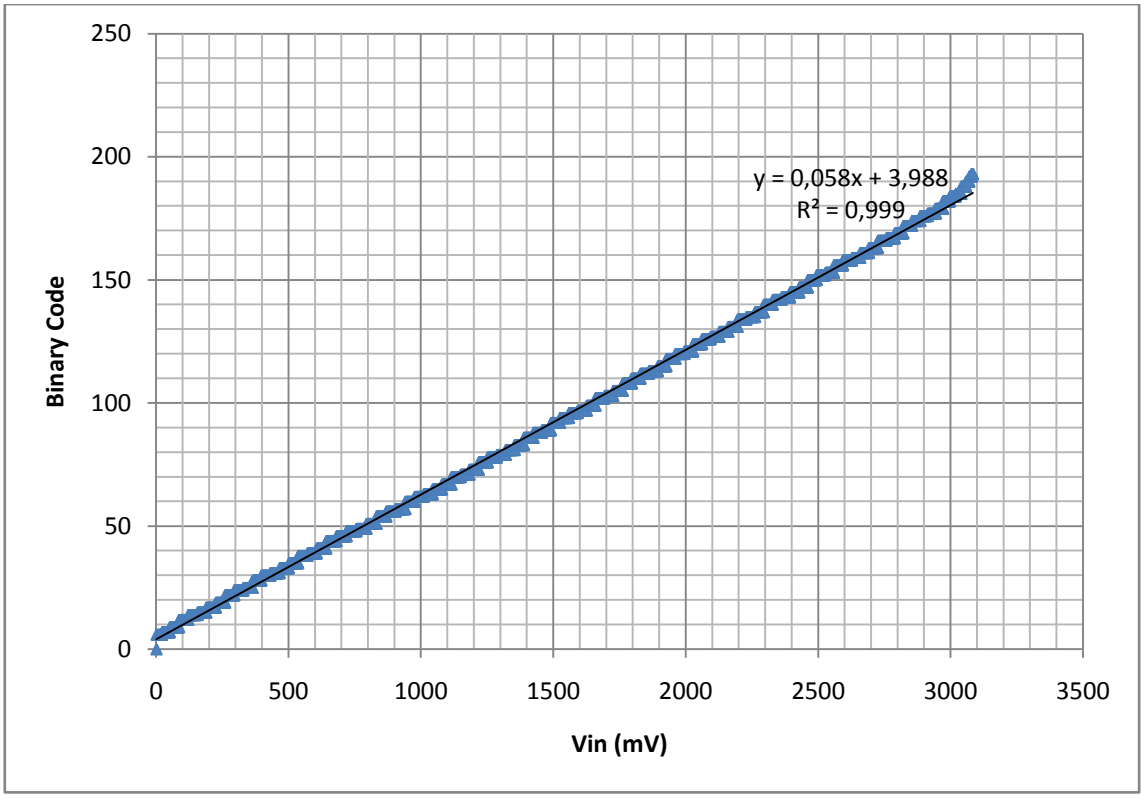
Pixel 2_5



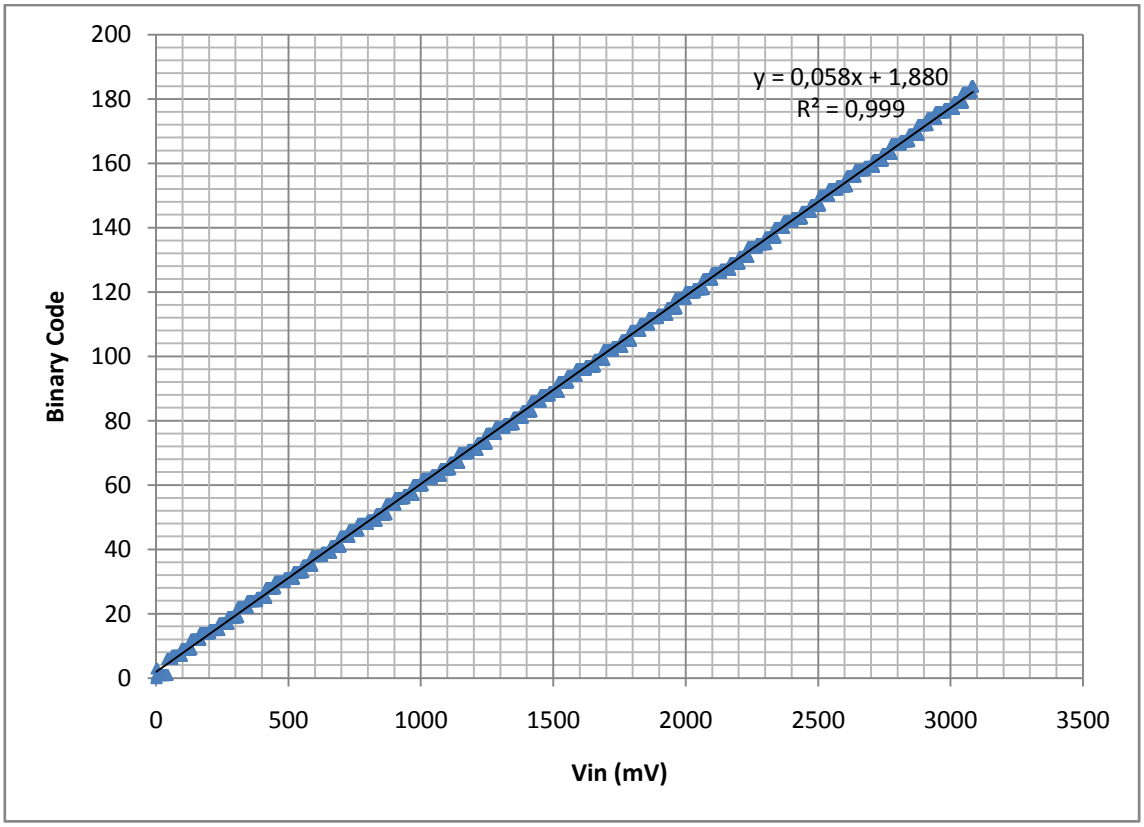
Pixel 2_6



Pixel 2_7



Pixel 2_8



Pixel Test_Str_1

Τα δεδομένα που λάβαμε ήταν σε πρώτη φάση ενθαρρυντικά καθώς όπως φαίνεται και από τα διαγράμματα υπάρχει γραμμικότητα τις εισόδου με την έξοδο. Προσεκτική διερεύνηση των τιμών που λήφθηκαν έδειξε ότι το σύστημα μετατροπής δε μπορούσε να καταγράψει όλες τις τιμές από 0 έως 255 όπως θα έπρεπε, καθώς πρόκειται για 8 bit μετατροπέα. Οι τιμές που έλειπαν ήταν σε όλο το εύρος τιμών και όχι ίδιες για όλα τα pixel.

Προσομοιώσεις έδειξαν ότι το σφάλμα αυτό οφειλόταν σε σχεδιαστικό λάθος. Ένα εσωτερικό σήμα του ψηφιακού τμήματος είχε διπλάσια περίοδο από αυτή του ρολογιού και έτσι δεν γινόταν σωστή φόρτιση των αρτηριών (precharge). Ενώ ο μετρητής Grey code μετρούσε όλο το εύρος τιμών από 0 έως 255 δεν ήταν δυνατή η εγγραφή των μισών τιμών στη μνήμη. Αυτό κάνει ουσιαστικά το μετατροπέα να έχει διακριτικότητα 7bit καθώς χάνονται οι μισές τιμές. Επειδή όμως οι τιμές που λείπουν είναι ομοιόμορφα κατανεμημένες σε όλο το εύρος από 0 έως 255, τελικά λαμβάνουμε ως έξοδο τις γραμμικές ευθείες που απεικονίζονται στα προηγούμενα σχήματα.

Με τη χρήση του σήματος wr_ext και τη ξεχωριστή μνήμη για σκοπούς ελέγχου στο ολοκληρωμένο ήταν δυνατός ο χαρακτηρισμός τις μνήμης dram. Το wr_ext σήμα οδηγήθηκε από το fpga και προσομοιώνει το σήμα που προέρχεται από το συγκριτή και δίνει την εντολή για εγγραφή στη μνήμη τις τρέχουσας τιμής του μετρητή Grey code. Το fpga τοποθετούσε μια λέξη στην αρτηρία D_IN και έστελνε high στο wr_ext. Έτσι τα δεδομένα τις αρτηρίας εγγράφονταν στην μνήμη DRAM . Η ορθή λειτουργία τις μνήμης επιβεβαιώθηκε με τη λήψη των μετρήσεων καθώς διαβάζονταν σωστά από τη μνήμη οι τιμές που εγγράφονταν σε αυτήν, εγγράφηκαν και διαβάστηκαν δηλαδή όλες οι λέξεις από 0 έως 255.

Για την μέτρηση αυτή χρειαζόταν να τοποθετείται μια τιμή στην αρτηρία D_IN και να διαβάζεται έπειτα από την αρτηρία D_OUT. Αυτή η σειρά μετρήσεων έδειξε ότι διαβάζαμε από την μνήμη τις τιμές που εγγράφαμε σε αυτή.

Τα παραπάνω σφάλματα απέτρεψαν την σε μεγαλύτερο βάθος μελέτη και λήψη μετρήσεων από το ολοκληρωμένο καθώς δεν ήταν δυνατόν να χαρακτηριστούν ούτε το αναλογικό μέρος αλλά ούτε και το ψηφιακό καθώς έδινε πληροφορία 7 bit, και μάλιστα χωρίς να λαμβάνει όλες τις διαδοχικές τιμές.

7.5 Συμπεράσματα

Η πλατφόρμα ελέγχου του ολοκληρωμένου που σχεδιάστηκε κατά την εκπόνηση της παρούσας διπλωματικής λειτούργησε ορθά και σχεδιάστηκε με τρόπο ώστε να μπορεί να επαναχρησιμοποιηθεί για επόμενες εκδόσεις ολοκληρωμένων. Η πλακέτα Π3 είναι χρήσιμη για την απομόνωση των σημάτων συγχρονισμού του USB μικροελεγκτή, ώστε να μειωθεί όσο το δυνατόν ο θόρυβος κατά την μέτρηση των αναλογικών σημάτων και επιπλέον προσφέρει περισσότερα λογικά στοιχεία. Για τον έλεγχο επόμενου ολοκληρωμένου κυκλώματος χρειάζονται μικρές αλλαγές στα σχέδια της πλακέτας Π2 και τα εξαρτήματα που επιλέχθηκαν μπορούν να επαναχρησιμοποιηθούν. Ο κώδικας που γράφηκε δίνει την δυνατότητα επικοινωνίας με την κάρτα ελέγχου και την ρύθμιση των παραμέτρων της από γραφικό περιβάλλον windows, καθώς και την αποθήκευση των δεδομένων σε μορφοποιημένα αρχεία κειμένου.

Κατά την αξιολόγηση του ολοκληρωμένου κυκλώματος επιβεβαιώθηκε η ορθή λειτουργία της δυναμικής του μνήμης D-RAM και η ορθή λειτουργία του ψηφιακού τμήματος, με εξαίρεση το κύκλωμα precharge, το οποίο είναι εύκολο να διορθωθεί σε επόμενη σχεδίαση αλλά απέτρεψε τον χαρακτηρισμό του μετατροπέα αναλογικού σήματος σε ψηφιακό. Το αναλογικό τμήμα δεν ήταν δυνατό να χαρακτηρισθεί καθώς δεν συμπεριφέρθηκε όπως αναμενόταν.

8 Παραρτήματα

8.1 Παράρτημα Α – Κώδικας Γραφικού Περιβάλλοντος

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Threading;
using CyUSB;
using NPlot;
using ZedGraph;
using System.Globalization;
namespace APIC
{
   public partial class Form1 : Form
   {
      USBDeviceList usbDevices;
      CyUSBDevice myDevice;
      const int base10 = 10;
      char[] cHexa = new char[] { 'A', 'B', 'C', 'D', 'E', 'F' };
      int[] iHexaNumeric = new int[] { 10, 11, 12, 13, 14, 15 };
      int[] iHexaIndices = new int[] { 0, 1, 2, 3, 4, 5 };
      const int asciiDiff = 48;
      public Form1()
      {
         InitializeComponent();
         usbDevices = new USBDeviceList(CyConst.DEVICES_CYUSB);
```

```
    usbDevices.DeviceAttached += new EventHandler(usbDevices_DeviceAttached);
    usbDevices.DeviceRemoved += new EventHandler(usbDevices_DeviceRemoved);
    myDevice = usbDevices[0] as CyUSBDevice;
    if (myDevice != null)
    {
        StatusLabel.Text = myDevice.FriendlyName + " connected.";
    }
}
//Device removal
void usbDevices_DeviceRemoved(object sender, EventArgs e)
{
    USBEventArgs usbEvent = e as USBEventArgs;
    StatusLabel.Text = usbEvent.FriendlyName + " removed.";
}
//Device attachment
void usbDevices_DeviceAttached(object sender, EventArgs e)
{
    USBEventArgs usbEvent = e as USBEventArgs;
    StatusLabel.Text = usbEvent.Device.FriendlyName + " connected.";
}
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    if (usbDevices != null)
        usbDevices.Dispose();
}
     private void SDACS1_Click(object sender, EventArgs e)
{
    byte[] receivedbackEP0 = new byte[64];
    byte[] bytes64toEP0 = new byte[64];
    UInt16[] kapa = new UInt16[16];
    byte[] kapafirst8 = new byte[16];
    byte[] kapalast8 = new byte[16];
```

```
byte dontcare8bits = 0xFF;

//byte[] commandWriteUpdateAddressN = new byte[8] { 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37 };//Write to and Update N DAC A-->H.

byte[] commandWriteUpdateAddressN = new byte[8] { 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37 };//Write to and Update N DAC A-->H.

byte[] buf = new byte[8];

byte[] buff = new byte[8];

byte[] bits64toEP0 = new byte[8];

for (int i = 0; i < 64; i = i + 4) //loop for all the bytes.

{

   bytes64toEP0[i] = dontcare8bits;

}

kapa[0] = Convert.ToUInt16(Math.Round((4096 * ((Convert.ToDouble(Icomp.Text)) / 3))));

kapa[0] <<= 4; //Left shift kapa 4 bits xxxxxxxxxxxx --> xxxxxxxxxxxx0000

kapafirst8[0] = Convert.ToByte((kapa[0] & 0xFF00) >> 8);//Place the first  8 bits of the 16 bit kapa to a separate byte (16bits due to the previous shift).

kapalast8[0]  = Convert.ToByte(kapa[0] & 0x00FF);//Place the last  8 bits of the 16 bit kapa to a separate byte (16bits due to the previous shift).

bytes64toEP0[1]  = commandWriteUpdateAddressN[0];

bytes64toEP0[2]  = kapafirst8[0];

bytes64toEP0[3]  = kapalast8[0];

kapa[1] = Convert.ToUInt16(Math.Round((4096 * ((Convert.ToDouble(IN4_1.Text)) / 3))));

kapa[1] <<= 4;

kapafirst8[1] = Convert.ToByte((kapa[1] & 0xFF00) >> 8);

kapalast8[1] = Convert.ToByte(kapa[1] & 0x00FF);

bytes64toEP0[5] = commandWriteUpdateAddressN[0];

bytes64toEP0[6] = kapafirst8[1];

bytes64toEP0[7] = kapalast8[1];

kapa[2] = Convert.ToUInt16(Math.Round((4096 * ((Convert.ToDouble(ADC_Ramp_Slope.Text)) / 3))));

kapa[2] <<= 4;

kapafirst8[2] = Convert.ToByte((kapa[2] & 0xFF00) >> 8);

kapalast8[2] = Convert.ToByte(kapa[2] & 0x00FF);

bytes64toEP0[9]  = commandWriteUpdateAddressN[1];

bytes64toEP0[10] = kapafirst8[2];
```

```
bytes64toEP0[11] = kapalast8[2];


kapa[3] = Convert.ToUInt16(Math.Round((4096 * ((Convert.ToDouble(IN3_1.Text)) / 3))));
kapa[3] <<= 4;
kapafirst8[3] = Convert.ToByte((kapa[3] & 0xFF00) >> 8);
kapalast8[3] = Convert.ToByte(kapa[3] & 0x00FF);
bytes64toEP0[13] = commandWriteUpdateAddressN[1];
bytes64toEP0[14] = kapafirst8[3];
bytes64toEP0[15] = kapalast8[3];
kapa[4] = Convert.ToUInt16(Math.Round((4096 * ((Convert.ToDouble(C1.Text)) / 3))));
kapa[4] <<= 4;
kapafirst8[4] = Convert.ToByte((kapa[4] & 0xFF00) >> 8);
kapalast8[4] = Convert.ToByte(kapa[4] & 0x00FF);
bytes64toEP0[17] = commandWriteUpdateAddressN[2];
bytes64toEP0[18] = kapafirst8[4];
bytes64toEP0[19] = kapalast8[4];
kapa[5] = Convert.ToUInt16(Math.Round((4096 * ((Convert.ToDouble(IN2_1.Text)) / 3))));
kapa[5] <<= 4;
kapafirst8[5] = Convert.ToByte((kapa[5] & 0xFF00) >> 8);
kapalast8[5] = Convert.ToByte(kapa[5] & 0x00FF);
bytes64toEP0[21] = commandWriteUpdateAddressN[2];
bytes64toEP0[22] = kapafirst8[5];
bytes64toEP0[23] = kapalast8[5];
kapa[6] = Convert.ToUInt16(Math.Round((4096 * ((Convert.ToDouble(D1.Text)) / 3))));
kapa[6] <<= 4;
kapafirst8[6] = Convert.ToByte((kapa[6] & 0xFF00) >>8);
kapalast8[6] = Convert.ToByte(kapa[6] & 0x00FF);
bytes64toEP0[25] = commandWriteUpdateAddressN[3];
bytes64toEP0[26] = kapafirst8[6];
bytes64toEP0[27] = kapalast8[6];
kapa[7] = Convert.ToUInt16(Math.Round((4096 * ((Convert.ToDouble(IN1_1.Text)) / 3))));
kapa[7] <<= 4;
```

```
kapafirst8[7] = Convert.ToByte((kapa[7] & 0xFF00) >> 8);
kapalast8[7] = Convert.ToByte(kapa[7] & 0x00FF);
bytes64toEP0[29] = commandWriteUpdateAddressN[3];
bytes64toEP0[30] = kapafirst8[7];
bytes64toEP0[31] = kapalast8[7];
kapa[8] = Convert.ToUInt16(Math.Round((4096 * ((Convert.ToDouble(E1.Text)) / 3))));
kapa[8] <<= 4;
kapafirst8[8] = Convert.ToByte((kapa[8] & 0xFF00) >> 8);
kapalast8[8] = Convert.ToByte(kapa[8] & 0x00FF);
bytes64toEP0[33] = commandWriteUpdateAddressN[4];
bytes64toEP0[34] = kapafirst8[8];
bytes64toEP0[35] = kapalast8[8];
kapa[9] = Convert.ToUInt16(Math.Round((4096 * ((Convert.ToDouble(IN8_1.Text)) / 3))));
kapa[9] <<= 4;
kapafirst8[9] = Convert.ToByte((kapa[9] & 0xFF00) >> 8);
kapalast8[9] = Convert.ToByte(kapa[9] & 0x00FF);
bytes64toEP0[37] = commandWriteUpdateAddressN[4];
bytes64toEP0[38] = kapafirst8[9];
bytes64toEP0[39] = kapalast8[9];
kapa[10] = Convert.ToUInt16(Math.Round((4096 * ((Convert.ToDouble(F1.Text)) / 3))));
kapa[10] <<= 4;
kapafirst8[10] = Convert.ToByte((kapa[10] & 0xFF00) >> 8);
kapalast8[10] = Convert.ToByte(kapa[10] & 0x00FF);
bytes64toEP0[41] = commandWriteUpdateAddressN[5];
bytes64toEP0[42] = kapafirst8[10];
bytes64toEP0[43] = kapalast8[10];
kapa[11] = Convert.ToUInt16(Math.Round((4096 * ((Convert.ToDouble(IN7_1.Text)) / 3))));
kapa[11] <<= 4;
kapafirst8[11] = Convert.ToByte((kapa[11] & 0xFF00) >> 8);
kapalast8[11] = Convert.ToByte(kapa[11] & 0x00FF);
bytes64toEP0[45] = commandWriteUpdateAddressN[5];
bytes64toEP0[46] = kapafirst8[11];
```

```
bytes64toEP0[47] = kapalast8[11];
kapa[12] = Convert.ToUInt16(Math.Round((4096 * ((Convert.ToDouble(Reset_Ramp_Slope.Text)) / 3))));
kapa[12] <<= 4;
kapafirst8[12] = Convert.ToByte((kapa[12] & 0xFF00) >> 8);
kapalast8[12] = Convert.ToByte(kapa[12] & 0x00FF);
bytes64toEP0[49] = commandWriteUpdateAddressN[6];
bytes64toEP0[50] = kapafirst8[12];
bytes64toEP0[51] = kapalast8[12];
kapa[13] = Convert.ToUInt16(Math.Round((4096 * ((Convert.ToDouble(IN6_1.Text)) / 3))));
kapa[13] <<= 4;
kapafirst8[13] = Convert.ToByte((kapa[13] & 0xFF00) >> 8);
kapalast8[13] = Convert.ToByte(kapa[13] & 0x00FF);
bytes64toEP0[53] = commandWriteUpdateAddressN[6];
bytes64toEP0[54] = kapafirst8[13];
bytes64toEP0[55] = kapalast8[13];
kapa[14] = Convert.ToUInt16(Math.Round((4096 * ((Convert.ToDouble(Ipreamp.Text)) / 3))));
kapa[14] <<= 4;
kapafirst8[14] = Convert.ToByte((kapa[14] & 0xFF00) >> 8);
kapalast8[14] = Convert.ToByte(kapa[14] & 0x00FF);
bytes64toEP0[57] = commandWriteUpdateAddressN[7];
bytes64toEP0[58] = kapafirst8[14];
bytes64toEP0[59] = kapalast8[14];
kapa[15] = Convert.ToUInt16(Math.Round((4096 * ((Convert.ToDouble(IN5_1.Text)) / 3))));
kapa[15] <<= 4;
kapafirst8[15] = Convert.ToByte((kapa[15] & 0xFF00) >> 8);
kapalast8[15] = Convert.ToByte(kapa[15] & 0x00FF);
bytes64toEP0[61] = commandWriteUpdateAddressN[7];
bytes64toEP0[62] = kapafirst8[15];
bytes64toEP0[63] = kapalast8[15];
int counter = 0;
for (int i = 0; i < 8; i = i + 1)
{
```

```
            for (int k = 0; k < 8; k = k + 1)
            {
                bits64toEP0[k] = bytes64toEP0[counter + k];
            }
            counter = counter + 8;
            CyControlEndPoint CtrlEndPt = null;
            USBDeviceList usbDevices = new USBDeviceList(CyConst.DEVICES_CYUSB);
            CyUSBDevice MyDevice = usbDevices[0] as CyUSBDevice;
            if (MyDevice != null)
                CtrlEndPt = MyDevice.ControlEndPt;
            if (CtrlEndPt != null)
            {
                CtrlEndPt.Target = CyConst.TGT_ENDPT;
                CtrlEndPt.ReqType = CyConst.REQ_VENDOR;
                CtrlEndPt.ReqCode = 0xB5;
                CtrlEndPt.Value = 0;
                CtrlEndPt.Index = 0;
                int len = 8;
                //byte[] buf = new byte[len];
                buf = bits64toEP0;

                CtrlEndPt.Write(ref buf, ref len);
                bool success = (len == 64);
                Console.WriteLine("Vendor Request Sent");
            }
        }
    }
    byte[] readoutEP0 = new byte[64];
    int[] readoutEP0_Int = new int[64];
    String[] convertedEP0_BinString= new String[64];
    int[][] convertedEP0_BinMatrix= new int[64][];
    int[][] ungrayCodedEP0_BinMatrix = new int[64][];
```

```
String[] Giannis = new String[64];
private void ReadBack_Click(object sender, EventArgs e)
{
   int countheprintedframes = 1;
   byte[] buff = new byte[8];
   byte[] bits64toEP0 = new byte[8];
   CyControlEndPoint CtrlEndPt = null;
   USBDeviceList usbDevices = new USBDeviceList(CyConst.DEVICES_CYUSB);
   CyUSBDevice MyDevice = usbDevices[0] as CyUSBDevice;
   if (MyDevice != null)
      CtrlEndPt = MyDevice.ControlEndPt;
   if (CtrlEndPt != null)
   {
      CtrlEndPt.Target = CyConst.TGT_DEVICE;
      CtrlEndPt.ReqType = CyConst.REQ_VENDOR;
      CtrlEndPt.ReqCode = 0xB2;
   }
   CtrlEndPt.Value = 0;
   CtrlEndPt.Index = 0;
   int len = 64;
   byte[] buf = new byte[len];
   CtrlEndPt.Read(ref buf, ref len);
   bool success = (len > 0);
   readoutEP0 = buf;
   for (int i = 0; i < 64; i++)
   {
      convertedEP0_BinString[i] = DecimalToBase(readoutEP0[i],2);
     // DecimalToBase(int iDec, int numbase);
   }
   //string path = @"c:\temp\MyTest.txt";
   //FileStream fileStream = new FileStream(@"c:\file#@@#.txt", FileMode.Append);
   FileStream APICDATA = new FileStream(@"c:\apic\APIC.txt", FileMode.Append);
```

```
        StreamWriter streamWriter = new StreamWriter(APICDATA);

        streamWriter.WriteLine(ALL_DACs.Text + "," + integration_time.Text + "," + readoutEP0[19] + "," +
readoutEP0[18] + "," + readoutEP0[17] + "," + readoutEP0[16] + "," + readoutEP0[15] + "," + readoutEP0[14] + "," +
readoutEP0[13] + "," + readoutEP0[12] + "," + readoutEP0[11] + "," + readoutEP0[10] + "," + readoutEP0[9] + "," +
readoutEP0[8] + "," + readoutEP0[7] + "," + readoutEP0[6] + "," + readoutEP0[5] + "," + readoutEP0[4] + "," +
readoutEP0[3] + "," + readoutEP0[2] + "," + readoutEP0[1] + "," + readoutEP0[0]);

        streamWriter.Close();

        countttheprintedframes = countttheprintedframes + 1;

        String line;

        StreamReader sr = new StreamReader(@"c:\apic\APIC.txt");

        output.Clear();

        while ((line = sr.ReadLine()) != null)

            {

                output.AppendText(line);

                output.AppendText("\n");

            }

        sr.Close();

    }

    private void SDACSB2_Click(object sender, EventArgs e)

    {

        byte[] receivedbackEP0 = new byte[64];

        byte[] bytes64toEP0 = new byte[64];

        UInt16[] kapa = new UInt16[16];

        byte[] kapafirst8 = new byte[16];

        byte[] kapalast8 = new byte[16];

        byte dontcare8bits = 0xFF;

        //byte[] commandWriteUpdateAddressN = new byte[8] { 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37 };//Write
to and Update N DAC A-->H.

        byte[] commandWriteUpdateAddressN = new byte[8] { 0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37 };//Write to
and Update N DAC A-->H.

        byte[] buf = new byte[8];

        byte[] buff = new byte[8];

        byte[] bits64toEP0 = new byte[8];

        for (int i = 0; i < 64; i = i + 4) //loop for all the bytes.

        {
```

```
            bytes64toEP0[i] = dontcare8bits;

        }

        kapa[0] = Convert.ToUInt16(Math.Round((4096 * ((Convert.ToDouble(IN5_2.Text)) / 3))));

        kapa[0] <<= 4; //Left shift kapa 4 bits xxxxxxxxxxxx --> xxxxxxxxxxxx0000

        kapafirst8[0] = Convert.ToByte((kapa[0] & 0xFF00) >> 8);//Place the first  8 bits of the 16 bit kapa to a separate byte (16bits due to the previous shift).

        kapalast8[0] = Convert.ToByte(kapa[0] & 0x00FF);//Place the last  8 bits of the 16 bit kapa to a separate byte (16bits due to the previous shift).

        bytes64toEP0[1] = commandWriteUpdateAddressN[0];

        bytes64toEP0[2] = kapafirst8[0];

        bytes64toEP0[3] = kapalast8[0];

        kapa[1] = Convert.ToUInt16(Math.Round((4096 * ((Convert.ToDouble(A2.Text)) / 3))));

        kapa[1] <<= 4;

        kapafirst8[1] = Convert.ToByte((kapa[1] & 0xFF00) >> 8);

        kapalast8[1] = Convert.ToByte(kapa[1] & 0x00FF);

        bytes64toEP0[5] = commandWriteUpdateAddressN[0];

        bytes64toEP0[6] = kapafirst8[1];

        bytes64toEP0[7] = kapalast8[1];

        kapa[2] = Convert.ToUInt16(Math.Round((4096 * ((Convert.ToDouble(IN6_2.Text)) / 3))));

        kapa[2] <<= 4;

        kapafirst8[2] = Convert.ToByte((kapa[2] & 0xFF00) >> 8);

        kapalast8[2] = Convert.ToByte(kapa[2] & 0x00FF);

        bytes64toEP0[9] = commandWriteUpdateAddressN[1];

        bytes64toEP0[10] = kapafirst8[2];

        bytes64toEP0[11] = kapalast8[2];

        kapa[3] = Convert.ToUInt16(Math.Round((4096 * ((Convert.ToDouble(IN_CH.Text)) / 3))));

        kapa[3] <<= 4;

        kapafirst8[3] = Convert.ToByte((kapa[3] & 0xFF00) >> 8);

        kapalast8[3] = Convert.ToByte(kapa[3] & 0x00FF);

        bytes64toEP0[13] = commandWriteUpdateAddressN[1];

        bytes64toEP0[14] = kapafirst8[3];

        bytes64toEP0[15] = kapalast8[3];
```

```
kapa[4] = Convert.ToUInt16(Math.Round((4096 * ((Convert.ToDouble(IN7_2.Text)) / 3))));
kapa[4] <<= 4;
kapafirst8[4] = Convert.ToByte((kapa[4] & 0xFF00) >> 8);
kapalast8[4] = Convert.ToByte(kapa[4] & 0x00FF);
bytes64toEP0[17] = commandWriteUpdateAddressN[2];
bytes64toEP0[18] = kapafirst8[4];
bytes64toEP0[19] = kapalast8[4];
kapa[5] = Convert.ToUInt16(Math.Round((4096 * ((Convert.ToDouble(C2.Text)) / 3))));
kapa[5] <<= 4;
kapafirst8[5] = Convert.ToByte((kapa[5] & 0xFF00) >> 8);
kapalast8[5] = Convert.ToByte(kapa[5] & 0x00FF);
bytes64toEP0[21] = commandWriteUpdateAddressN[2];
bytes64toEP0[22] = kapafirst8[5];
bytes64toEP0[23] = kapalast8[5];
kapa[6] = Convert.ToUInt16(Math.Round((4096 * ((Convert.ToDouble(IN8_2.Text)) / 3))));
kapa[6] <<= 4;
kapafirst8[6] = Convert.ToByte((kapa[6] & 0xFF00) >> 8);
kapalast8[6] = Convert.ToByte(kapa[6] & 0x00FF);
bytes64toEP0[25] = commandWriteUpdateAddressN[3];
bytes64toEP0[26] = kapafirst8[6];
bytes64toEP0[27] = kapalast8[6];
kapa[7] = Convert.ToUInt16(Math.Round((4096 * ((Convert.ToDouble(D2.Text)) / 3))));
kapa[7] <<= 4;
kapafirst8[7] = Convert.ToByte((kapa[7] & 0xFF00) >> 8);
kapalast8[7] = Convert.ToByte(kapa[7] & 0x00FF);
bytes64toEP0[29] = commandWriteUpdateAddressN[3];
bytes64toEP0[30] = kapafirst8[7];
bytes64toEP0[31] = kapalast8[7];
kapa[8] = Convert.ToUInt16(Math.Round((4096 * ((Convert.ToDouble(IN1_2.Text)) / 3))));
kapa[8] <<= 4;
kapafirst8[8] = Convert.ToByte((kapa[8] & 0xFF00) >> 8);
kapalast8[8] = Convert.ToByte(kapa[8] & 0x00FF);
```

```
bytes64toEP0[33] = commandWriteUpdateAddressN[4];
bytes64toEP0[34] = kapafirst8[8];
bytes64toEP0[35] = kapalast8[8];

kapa[9] = Convert.ToUInt16(Math.Round((4096 * ((Convert.ToDouble(E2.Text)) / 3))));
kapa[9] <<= 4;
kapafirst8[9] = Convert.ToByte((kapa[9] & 0xFF00) >> 8);
kapalast8[9] = Convert.ToByte(kapa[9] & 0x00FF);
bytes64toEP0[37] = commandWriteUpdateAddressN[4];
bytes64toEP0[38] = kapafirst8[9];
bytes64toEP0[39] = kapalast8[9];
kapa[10] = Convert.ToUInt16(Math.Round((4096 * ((Convert.ToDouble(IN2_2.Text)) / 3))));
kapa[10] <<= 4;
kapafirst8[10] = Convert.ToByte((kapa[10] & 0xFF00) >> 8);
kapalast8[10] = Convert.ToByte(kapa[10] & 0x00FF);
bytes64toEP0[41] = commandWriteUpdateAddressN[5];
bytes64toEP0[42] = kapafirst8[10];
bytes64toEP0[43] = kapalast8[10];
kapa[11] = Convert.ToUInt16(Math.Round((4096 * ((Convert.ToDouble(F2.Text)) / 3))));
kapa[11] <<= 4;
kapafirst8[11] = Convert.ToByte((kapa[11] & 0xFF00) >> 8);
kapalast8[11] = Convert.ToByte(kapa[11] & 0x00FF);
bytes64toEP0[45] = commandWriteUpdateAddressN[5];
bytes64toEP0[46] = kapafirst8[11];
bytes64toEP0[47] = kapalast8[11];
kapa[12] = Convert.ToUInt16(Math.Round((4096 * ((Convert.ToDouble(IN3_2.Text)) / 3))));
kapa[12] <<= 4;
kapafirst8[12] = Convert.ToByte((kapa[12] & 0xFF00) >> 8);
kapalast8[12] = Convert.ToByte(kapa[12] & 0x00FF);
bytes64toEP0[49] = commandWriteUpdateAddressN[6];
bytes64toEP0[50] = kapafirst8[12];
bytes64toEP0[51] = kapalast8[12];
```

```
kapa[13] = Convert.ToUInt16(Math.Round((4096 * ((Convert.ToDouble(G2.Text)) / 3))));
kapa[13] <<= 4;
kapafirst8[13] = Convert.ToByte((kapa[13] & 0xFF00) >> 8);
kapalast8[13] = Convert.ToByte(kapa[13] & 0x00FF);
bytes64toEP0[53] = commandWriteUpdateAddressN[6];
bytes64toEP0[54] = kapafirst8[13];
bytes64toEP0[55] = kapalast8[13];
kapa[14] = Convert.ToUInt16(Math.Round((4096 * ((Convert.ToDouble(IN4_2.Text)) / 3))));
kapa[14] <<= 4;
kapafirst8[14] = Convert.ToByte((kapa[14] & 0xFF00) >> 8);
kapalast8[14] = Convert.ToByte(kapa[14] & 0x00FF);
bytes64toEP0[57] = commandWriteUpdateAddressN[7];
bytes64toEP0[58] = kapafirst8[14];
bytes64toEP0[59] = kapalast8[14];
kapa[15] = Convert.ToUInt16(Math.Round((4096 * ((Convert.ToDouble(H2.Text)) / 3))));
kapa[15] <<= 4;
kapafirst8[15] = Convert.ToByte((kapa[15] & 0xFF00) >> 8);
kapalast8[15] = Convert.ToByte(kapa[15] & 0x00FF);
bytes64toEP0[61] = commandWriteUpdateAddressN[7];
bytes64toEP0[62] = kapafirst8[15];
bytes64toEP0[63] = kapalast8[15];
int counter = 0;
for (int i = 0; i < 8; i = i + 1)
{
   for (int k = 0; k < 8; k = k + 1)
   {
      bits64toEP0[k] = bytes64toEP0[counter + k];
   }
   counter = counter + 8;
   CyControlEndPoint CtrlEndPt = null;
   USBDeviceList usbDevices = new USBDeviceList(CyConst.DEVICES_CYUSB);
```

```
            CyUSBDevice MyDevice = usbDevices[0] as CyUSBDevice;
            if (MyDevice != null)
                CtrlEndPt = MyDevice.ControlEndPt;
            if (CtrlEndPt != null)
            {
                CtrlEndPt.Target = CyConst.TGT_ENDPT;
                CtrlEndPt.ReqType = CyConst.REQ_VENDOR;
                CtrlEndPt.ReqCode = 0xB6;
                CtrlEndPt.Value = 0;
                CtrlEndPt.Index = 0;
                int len = 8;
                //byte[] buf = new byte[len];
                buf = bits64toEP0;

                CtrlEndPt.Write(ref buf, ref len);
                bool success = (len == 64);
                Console.WriteLine("Vendor Request Sent");
            }
        }
        Console.WriteLine("DONE");
    }
    string DecimalToBase(int iDec, int numbase)
    {
        string strBin = "";
        int[] result = new int[32];
        int MaxBit = 32;
        for (; iDec > 0; iDec /= numbase)
        {
            int rem = iDec % numbase;
            result[--MaxBit] = rem;
        }
        for (int i = 0; i < result.Length; i++)
```

```
            if ((int)result.GetValue(i) >= base10)
                strBin += cHexa[(int)result.GetValue(i) % base10];
            else
                strBin += result.GetValue(i);
        strBin = strBin.TrimStart(new char[] { '0' });
        return strBin;
    }
    private void button1_Click(object sender, EventArgs e)
    {
        IN1_1.Text = ALL_DACs.Text;
        IN2_1.Text = ALL_DACs.Text;
        IN3_1.Text = ALL_DACs.Text;
        IN4_1.Text = ALL_DACs.Text;
        IN5_1.Text = ALL_DACs.Text;
        IN6_1.Text = ALL_DACs.Text;
        IN7_1.Text = ALL_DACs.Text;
        IN8_1.Text = ALL_DACs.Text;

        IN1_2.Text = ALL_DACs.Text;
        IN2_2.Text = ALL_DACs.Text;
        IN3_2.Text = ALL_DACs.Text;
        IN4_2.Text = ALL_DACs.Text;
        IN5_2.Text = ALL_DACs.Text;
        IN6_2.Text = ALL_DACs.Text;
        IN7_2.Text = ALL_DACs.Text;
        IN8_2.Text = ALL_DACs.Text;
    }
    private void button2_Click(object sender, EventArgs e)
    {
        CyControlEndPoint CtrlEndPt = null;
        USBDeviceList usbDevices = new USBDeviceList(CyConst.DEVICES_CYUSB);
        CyUSBDevice MyDevice = usbDevices[0] as CyUSBDevice;
```

```
byte[] bytes64toEP0Dac1 = new byte[8];
byte[] bytes64toEP0Dac2 = new byte[8];
UInt16 kapa;
byte kapafirst8;
byte kapalast8;
byte dontcare8bits = 0xFF;
byte commandWriteUpdateAddressALL = 0x2F;//Write to and Update ALL N DAC A-->H.
byte commandWriteUpdateAddressNoOperation = 0xFF;//No Operation ALL N DAC A-->H.
int suffix = 0;
   FileStream APICDATA = new FileStream(@"c:\apic\APIC"+(suffix)+".txt", FileMode.Append);
   StreamWriter streamWriter = new StreamWriter(APICDATA);

   streamWriter.WriteLine("Pixel: " + suffix);
   for (int dacValue = 0; dacValue < 4096; dacValue = dacValue + 1)
   {
      PauseForMilliSeconds(2);
      double dacValueVar = dacValue;
      double voltage = (dacValueVar * 3.0) / 4096.0;
      kapa = (UInt16)dacValue;
      kapa <<= 4;
      kapafirst8 = Convert.ToByte((kapa & 0xFF00) >> 8);
      kapalast8 = Convert.ToByte(kapa & 0x00FF);
      bytes64toEP0Dac1[0] = dontcare8bits;
      bytes64toEP0Dac1[1] = commandWriteUpdateAddressNoOperation;
      bytes64toEP0Dac1[2] = kapafirst8;
      bytes64toEP0Dac1[3] = kapalast8;
      bytes64toEP0Dac1[4] = dontcare8bits;
      bytes64toEP0Dac1[5] = commandWriteUpdateAddressALL;
      bytes64toEP0Dac1[6] = kapafirst8;
      bytes64toEP0Dac1[7] = kapalast8;

      bytes64toEP0Dac2[0] = dontcare8bits;
```

```
bytes64toEP0Dac2[1] = commandWriteUpdateAddressALL;
bytes64toEP0Dac2[2] = kapafirst8;
bytes64toEP0Dac2[3] = kapalast8;
bytes64toEP0Dac2[4] = dontcare8bits;
bytes64toEP0Dac2[5] = commandWriteUpdateAddressALL;
bytes64toEP0Dac2[6] = kapafirst8;
bytes64toEP0Dac2[7] = kapalast8;
if (MyDevice != null)
    CtrlEndPt = MyDevice.ControlEndPt;
if (CtrlEndPt != null)
{
    CtrlEndPt.Target = CyConst.TGT_ENDPT;
    CtrlEndPt.ReqType = CyConst.REQ_VENDOR;
    CtrlEndPt.ReqCode = 0xB5;
    CtrlEndPt.Value = 0;
    CtrlEndPt.Index = 0;
    int len = 8;
    byte[] buf = new byte[len];
    buf = bytes64toEP0Dac1;
    CtrlEndPt.Write(ref buf, ref len);
    bool success = (len == 8);
    //Console.WriteLine("Vendor Request Sent");
}
PauseForMilliSeconds(2);
if (MyDevice != null)
    CtrlEndPt = MyDevice.ControlEndPt;
if (CtrlEndPt != null)
{
    CtrlEndPt.Target = CyConst.TGT_ENDPT;
    CtrlEndPt.ReqType = CyConst.REQ_VENDOR;
    CtrlEndPt.ReqCode = 0xB6;
    CtrlEndPt.Value = 0;
```

```
            CtrlEndPt.Index = 0;
            int len = 8;
            byte[] buf = new byte[len];
            buf = bytes64toEP0Dac2;
            CtrlEndPt.Write(ref buf, ref len);
            bool success = (len == 8);
            //Console.WriteLine("Vendor Request Sent");
        }
        PauseForMilliSeconds(2);
        int countttheprintedframes = 1;
        for (int measurements = 0; measurements < 1; measurements = measurements + 1)
        {       if (MyDevice != null)
                CtrlEndPt = MyDevice.ControlEndPt;
            if (CtrlEndPt != null)
            {
                CtrlEndPt.Target = CyConst.TGT_DEVICE;
                CtrlEndPt.ReqType = CyConst.REQ_VENDOR;
                CtrlEndPt.ReqCode = 0xB2;
            }
            CtrlEndPt.Value = 0;
            CtrlEndPt.Index = 0;
            int lenn = 64;
            byte[] buff = new byte[lenn];
            CtrlEndPt.Read(ref buff, ref lenn);
            readoutEP0 = buff;
            for (int i = 0; i < 64; i++)
            {
                convertedEP0_BinString[i] = DecimalToBase(readoutEP0[i], 2);
            }
            String voltageString = voltage.ToString("0.0000");
            streamWriter.WriteLine(voltageString + "," + readoutEP0[19] + "," + readoutEP0[18] + "," +
readoutEP0[17] + "," + readoutEP0[16] + "," + readoutEP0[15] + "," + readoutEP0[14] + "," + readoutEP0[13] + "," +
readoutEP0[12] + "," + readoutEP0[11] + "," + readoutEP0[10] + "," + readoutEP0[9] + "," + readoutEP0[8] + "," +
```

```
readoutEP0[7] + "," + readoutEP0[6] + "," + readoutEP0[5] + "," + readoutEP0[4] + "," + readoutEP0[3] + "," +
readoutEP0[2] + "," + readoutEP0[1] + "," + readoutEP0[0]);

                countttheprintedframes = countttheprintedframes + 1;

            }

            }

            suffix = suffix + 1;

            streamWriter.Close();

          // }

        }

        public static DateTime PauseForMilliSeconds(int MilliSecondsToPauseFor)

        {

            System.DateTime ThisMoment = System.DateTime.Now;

            System.TimeSpan duration = new System.TimeSpan(0, 0, 0, 0, MilliSecondsToPauseFor);

            System.DateTime AfterWards = ThisMoment.Add(duration);

            while (AfterWards >= ThisMoment)

            {

                System.Windows.Forms.Application.DoEvents();

                ThisMoment = System.DateTime.Now;

            }

            return System.DateTime.Now;

        }

        private void WR_EXT_HIT_Click(object sender, EventArgs e)

        {

            UInt16 WR_EXT_userInput;

            byte WR_EXT_first8;

            byte WR_EXT_last8;

            WR_EXT_userInput = Convert.ToUInt16(wr_ext.Text);

                WR_EXT_first8 = Convert.ToByte((WR_EXT_userInput & 0xFF00) >> 8);

                WR_EXT_last8 = Convert.ToByte(WR_EXT_userInput & 0x00FF);

            byte[] bytes2toEP0_WR_EXT = new byte[2];

            bytes2toEP0_WR_EXT[1] = WR_EXT_first8;

            bytes2toEP0_WR_EXT[0] = WR_EXT_last8;

            CyControlEndPoint CtrlEndPt = null;
```

```
        USBDeviceList usbDevices = new USBDeviceList(CyConst.DEVICES_CYUSB);
        CyUSBDevice MyDevice = usbDevices[0] as CyUSBDevice;


        if (MyDevice != null)
            CtrlEndPt = MyDevice.ControlEndPt;
        if (CtrlEndPt != null)
        {
            CtrlEndPt.Target = CyConst.TGT_ENDPT;
            CtrlEndPt.ReqType = CyConst.REQ_VENDOR;
            CtrlEndPt.ReqCode = 0xBB;
            CtrlEndPt.Value = 0;
            CtrlEndPt.Index = 0;
            int len = 2;
            byte[] buf = new byte[len];
            buf = bytes2toEP0_WR_EXT;
            CtrlEndPt.Write(ref buf, ref len);
            bool success = (len == 2 );
            //Console.WriteLine("Vendor Request Sent");
        }
    }
    private void INT_TIME_Click(object sender, EventArgs e)
    {
        UInt16 INT_TIME_userInput;
        byte INT_TIME_first8;
        byte INT_TIME_last8;
        INT_TIME_userInput = Convert.ToUInt16(integration_time.Text);
        INT_TIME_first8 = Convert.ToByte((INT_TIME_userInput & 0xFF00) >> 8);
        INT_TIME_last8 = Convert.ToByte(INT_TIME_userInput & 0x00FF);
        byte[] bytes2toEP0_INT_TIME = new byte[2];
        bytes2toEP0_INT_TIME[1] = INT_TIME_first8;
        bytes2toEP0_INT_TIME[0] = INT_TIME_last8;
        CyControlEndPoint CtrlEndPt = null;
```

```
        USBDeviceList usbDevices = new USBDeviceList(CyConst.DEVICES_CYUSB);
        CyUSBDevice MyDevice = usbDevices[0] as CyUSBDevice;
        if (MyDevice != null)
            CtrlEndPt = MyDevice.ControlEndPt;
        if (CtrlEndPt != null)
        {
            CtrlEndPt.Target = CyConst.TGT_ENDPT;
            CtrlEndPt.ReqType = CyConst.REQ_VENDOR;
            CtrlEndPt.ReqCode = 0xB7;
            CtrlEndPt.Value = 0;
            CtrlEndPt.Index = 0;
            int len = 2;
            byte[] buf = new byte[len];
           buf = bytes2toEP0_INT_TIME;
            CtrlEndPt.Write(ref buf, ref len);
            bool success = (len == 2);
            //Console.WriteLine("Vendor Request Sent");
        }
    }
    private void RESET_TIME_Click(object sender, EventArgs e)
    {
        UInt16 RESET_TIME_userInput;
        byte RESET_TIME_first8;
        byte RESET_TIME_last8;
        RESET_TIME_userInput = Convert.ToUInt16(reset.Text);
        RESET_TIME_first8 = Convert.ToByte((RESET_TIME_userInput & 0xFF00) >> 8);
        RESET_TIME_last8 = Convert.ToByte(RESET_TIME_userInput & 0x00FF);
        byte[] bytes2toEP0_RESET_TIME = new byte[2];
        bytes2toEP0_RESET_TIME[1] = RESET_TIME_first8;
        bytes2toEP0_RESET_TIME[0] = RESET_TIME_last8;
        CyControlEndPoint CtrlEndPt = null;
        USBDeviceList usbDevices = new USBDeviceList(CyConst.DEVICES_CYUSB);
```

```
        CyUSBDevice MyDevice = usbDevices[0] as CyUSBDevice;
        if (MyDevice != null)
            CtrlEndPt = MyDevice.ControlEndPt;
        if (CtrlEndPt != null)
        {
            CtrlEndPt.Target = CyConst.TGT_ENDPT;
            CtrlEndPt.ReqType = CyConst.REQ_VENDOR;
            CtrlEndPt.ReqCode = 0xB8;
            CtrlEndPt.Value = 0;
            CtrlEndPt.Index = 0;
            int len = 2;
            byte[] buf = new byte[len];
            buf = bytes2toEP0_RESET_TIME;
            CtrlEndPt.Write(ref buf, ref len);
            bool success = (len == 2);
            //Console.WriteLine("Vendor Request Sent");
        }
    }
    private void FR_RAMP_Click(object sender, EventArgs e)
    {
        UInt16 FR_RAMP_TIME_userInput;
        byte FR_RAMP_TIME_first8;
        byte FR_RAMP_TIME_last8;
        FR_RAMP_TIME_userInput = Convert.ToUInt16(freeze_ramp_time.Text);
        FR_RAMP_TIME_first8 = Convert.ToByte((FR_RAMP_TIME_userInput & 0xFF00) >> 8);
        FR_RAMP_TIME_last8 = Convert.ToByte(FR_RAMP_TIME_userInput & 0x00FF);
        byte[] bytes2toEP0_FR_RAMP = new byte[2];
        bytes2toEP0_FR_RAMP[1] = FR_RAMP_TIME_first8;
        bytes2toEP0_FR_RAMP[0] = FR_RAMP_TIME_last8;
        CyControlEndPoint CtrlEndPt = null;
        USBDeviceList usbDevices = new USBDeviceList(CyConst.DEVICES_CYUSB);
        CyUSBDevice MyDevice = usbDevices[0] as CyUSBDevice;
```

```
        if (MyDevice != null)
            CtrlEndPt = MyDevice.ControlEndPt;
        if (CtrlEndPt != null)
        {
            CtrlEndPt.Target = CyConst.TGT_ENDPT;
            CtrlEndPt.ReqType = CyConst.REQ_VENDOR;
            CtrlEndPt.ReqCode = 0xB9;
            CtrlEndPt.Value = 0;
            CtrlEndPt.Index = 0;
            int len = 2;
            byte[] buf = new byte[len];
            buf = bytes2toEP0_FR_RAMP;
            CtrlEndPt.Write(ref buf, ref len);
            bool success = (len == 2);
            //Console.WriteLine("Vendor Request Sent");
        }
    }
    private void scan_wr_ext_Click(object sender, EventArgs e)
    {
        for (int i = 130; i < 256; i++)//101 255
        {
            wr_ext.Text = Convert.ToString(i);
            PauseForMilliSeconds(10);
            WR_EXT_HIT_Click(WR_EXT_HIT, EventArgs.Empty);
            PauseForMilliSeconds(10);
            ReadBack_Click(ReadBack, EventArgs.Empty);
            PauseForMilliSeconds(10);
        }
    }

    private void int_time_scan_Click(object sender, EventArgs e)
    {
```

```
        for (int i = 100; i < 65000; i=i+500)//101 255
        {
            integration_time.Text = Convert.ToString(i);
            PauseForMilliSeconds(10);
            INT_TIME_Click(WR_EXT_HIT, EventArgs.Empty);
            PauseForMilliSeconds(10);
            ReadBack_Click(ReadBack, EventArgs.Empty);
            PauseForMilliSeconds(10);
        }
    }
}
```

8.2 Παράρτημα Β – Κώδικας USB Μικροελεγκτή

```
#pragma NOIV               // Do not generate interrupt vectors
#include "fx2.h"
#include "fx2regs.h"
#include "fx2sdly.h"       // SYNCDELAY macro, see Section 15.14 of FX2 Tech.
#define bmEP0BSY      0x01
#define BIT0    0x01
#define BIT1    0x02
#define BIT2    0x04
#define BIT3    0x08
#define BIT4    0x10
#define BIT5    0x20
#define BIT6    0x40
#define BIT7    0x80
#define GPIFTRIGRD 4
#define GPIF_EP2 0
#define GPIF_EP4 1
#define GPIF_EP6 2
#define GPIF_EP8 3
#define ADDR_IDLE       0
#define ADDR_SPI        1
#define ADDR_CFG        2
#define ADDR_CFG_RST    3
#define ADDR_PULSE_LSB  4
#define ADDR_PULSE_MSB  5
#define SPI_MOSI        bmBIT0
#define SPI_SCK         bmBIT1
#define CS_CPLD         bmBIT2
#define CS_TEMPSENSOR   bmBIT3
#define CS_SDAC         bmBIT4
#define CLR_SDAC        bmBIT5
extern BOOL GotSUD;                // Received setup data flag
extern BOOL Sleep;
extern BOOL Rwuen;
extern BOOL Selfpwr;
BYTE Configuration;                // Current configuration
BYTE AlternateSetting;             // Alternate settings
BOOL auto_enable = FALSE;          // flag to enable auto IN transfers
BOOL enum_high_speed = FALSE;      // flag to let firmware know FX2 enumerated at high speed
extern const char xdata FlowStates[36];
BYTE i,k;
BYTE SPI_PINS;
WORD SPI_Data;
BYTE DACBUF[16];
WORD pulse_counter;
//=============================================================================
BYTE i,k,j,m;
BYTE BITMASK[8] = {BIT0,BIT1,BIT2,BIT3,BIT4,BIT5,BIT6,BIT7};
BYTE INTEGRATION_TIME;
BYTE RESET_TIME;
BYTE FREEZE_RAMP_TIME;
//-----------------------------------------------------------------------------
```

```
52  // Task Dispatcher hooks
53  // The following hooks are called by the task dispatcher.
54  //-----------------------------------------------------------------------------
55  void GpifInit ();
56  void SingleWrite ( BYTE trdata );
57  BYTE SingleRead ( void );
58
59  void TD_Init(void)      // Called once at startup
60  {
61      // set the CPU clock to 48MHz
62      CPUCS = ((CPUCS & ~bmCLKSPD) | bmCLKSPD1);
63      SYNCDELAY;
64
65      EP2CFG = 0xA0;      // EP2OUT, bulk, size 512, 4x buffered
66      SYNCDELAY;
67      EP4CFG = 0x00;      // EP4 not valid
68      SYNCDELAY;
69      EP6CFG = 0xE0;      // EP6IN, bulk, size 512, 4x buffered
70      SYNCDELAY;
71      EP8CFG = 0x00;      // EP8 not valid
72      SYNCDELAY;
73
74      FIFORESET = 0x80;  // set NAKALL bit to NAK all transfers from host
75      SYNCDELAY;
76      FIFORESET = 0x02;  // reset EP2 FIFO
77      SYNCDELAY;
78      FIFORESET = 0x06;  // reset EP6 FIFO
79      SYNCDELAY;
80      FIFORESET = 0x00;  // clear NAKALL bit to resume normal operation
81      SYNCDELAY;
82
83      EP2FIFOCFG = 0x00; // allow core to see zero to one transition of auto out bit
84      SYNCDELAY;
85      EP2FIFOCFG = 0x10; // auto out mode, disable PKTEND zero length send, byte ops
86      SYNCDELAY;
87      EP6FIFOCFG = 0x08; // auto in mode, disable PKTEND zero length send, byte ops
88      SYNCDELAY;
89      EP4FIFOCFG = 0x04; // default value except byte ops to free PortD for firmware use
90      SYNCDELAY;
91      EP8FIFOCFG = 0x04; // default value except byte ops to free PortD for firmware use
92      SYNCDELAY;
93
94      GpifInit (); // initialize GPIF registers
95
96      SYNCDELAY;
97      EP2GPIFFLGSEL = 0x01; // For EP2OUT, GPIF uses EF flag
98      SYNCDELAY;
99      EP6GPIFFLGSEL = 0x02; // For EP6IN, GPIF uses FF flag
100     SYNCDELAY;
101     // global flowstate register initializations
102     FLOWLOGIC = FlowStates[19];      // 0011 0110b - LFUNC[1:0] = 00 (A AND B), TERMA/B[2:0]=110 (FIFO Flag)
```

```
103     SYNCDELAY;
104     FLOWSTB = FlowStates[22];        // 0000 0100b - MSTB[2:0] = 100 (CTL4), not used as strobe
105     SYNCDELAY;
106     GPIFHOLDAMOUNT = FlowStates[26]; // hold data for one half clock (10ns) assuming 48MHz IFCLK
107     SYNCDELAY;
108     FLOWSTBEDGE = FlowStates[24];    // move data on both edges of clock
109     SYNCDELAY;
110     FLOWSTBHPERIOD = FlowStates[25]; // 20.83ns half period
111     SYNCDELAY;
112     OED = 0xFF;
113 //=========================================================================================
114     IFCONFIG = 0xE8;
115     OED=0xFF;
116     OEB=0xFF;
117 }
118 void TD_Poll(void)
119 {
120
121 }
122 BOOL TD_Suspend(void)          // Called before the device goes into suspend mode
123 {
124     return(TRUE);
125 }
126 BOOL TD_Resume(void)          // Called after the device resumes
127 {
128     return(TRUE);
129 }
130 //-----------------------------------------------------------------------------
131 // Device Request hooks
132 // The following hooks are called by the end point 0 device request parser.
133 //-----------------------------------------------------------------------------
134 BOOL DR_GetDescriptor(void)
135 {
136     return(TRUE);
137 }
138 BOOL DR_SetConfiguration(void)   // Called when a Set Configuration command is received
139 {
140     if( EZUSB_HIGHSPEED( ) )
141     { // FX2 enumerated at high speed
142         SYNCDELAY;                  //
143         EP6AUTOINLENH = 0x02;       // set AUTOIN commit length to 512 bytes
144         SYNCDELAY;                  //
145         EP6AUTOINLENL = 0x00;
146         SYNCDELAY;
147         enum_high_speed = TRUE;
148     }
149     else
150     { // FX2 enumerated at full speed
151         SYNCDELAY;
152         EP6AUTOINLENH = 0x00;       // set AUTOIN commit length to 64 bytes
153         SYNCDELAY;
```

```
switch (SETUPDAT[1])
{
  case VX_B2://READOUT!//it takes a total of 100us to complete this task.
  {
      OEB=0x00; // SET PORT B AS INPUT
      //EP0BCL = 0;       //EP0BUF LSB BYTE COUNT
      while(EP01STAT & bmEP0BSY);     // wait until EP0 is available to be accessed by CPU
      IOD = 0xA0;//10100000 //Start data transfer from FPGA_A to FPGA_D plus a signal at ADDR(5) to count the duration of this ven
      for (i=0;i<150;i=i+1) //Allow some time to get the data to FPGA_D.
        {

        }
      IOD = 0x20;    //00100000
      for (i=0;i<20;i=i+1) //Get the data.
        {
          EP0BUF[i]=IOB; //Store the currently availiable data to EP0BUF.

          IOD = 0x40;//01000000 //While in the state machine, send a rising edge of ADDRESSBUS_IN(6)
          IOD = 0x00;//00000000 //While in the state machine, send a falling edge of ADDRESSBUS_IN(6)

        }
     IOD = 0x40;//01000000 //While in the state machine, send a rising edge of ADDRESSBUS_IN(6)
     IOD = 0x00;//00000000 //While in the state machine, send a falling edge of ADDRESSBUS_IN(6)
     SYNCDELAY;
     EP0BCH = 0;
     EP0BCL = 20;                  // Arm endpoint with 64# unsigned chars to transfer
     EP0CS |= bmHSNAK;             // Acknowledge handshake phase of device request
   break;
  }
  case VX_B3: // enable IN transfers
  {
   break;
  }
  case VX_B4: // disable IN transfers
  {
     SYNCDELAY;
     EP0BCH = 0;
     EP0BCL = 64;                  // Arm endpoint with 64# unsigned chars to transfer
     EP0CS |= bmHSNAK;             // Acknowledge handshake phase of device request
   break;
  }
  case VX_B5: // read GPIFREADYSTAT register
  {
      OEB=0xFF; // SET PORT B AS OUTPUT
      IOD = 0x01; //Select SPI1.
      EP0BCL = 0;      //EP0BUF LSB BYTE COUNT
      while(EP01STAT & bmEP0BSY);     // wait until EP0 is available to be accessed by CPU
      m = 0;
      for (i=0;i<150;i=i+1)
      {
          IOB = IOB | 0x04;          //CSLDHI
```

```
256             }
257             //EZUSB_Delay(1);
258             IOB = IOB & ~(0x01);        //SDILOW
259                 for (i=0;i<8;i=i+1)             // Loop 64 times: from 0xE740 to 0xE77F memory bytes(the memory addresses of EP0BUF).
260                 {
261                     for(j=0;j<150;j++)
262                     {
263                         IOB = IOB & ~(0x04);        //CSLDLOW
264                     }
265                     IOB = IOB & ~(0x02);        //SDILOW
266
267                     for(k=0;k<8;k++)
268                     {
269                         IOB = IOB & ~(0x04);        //CSLDLOW
270                         if(EP0BUF[i] & BITMASK[7-k])        //BitMask EP0BUF Bytes (7-k to force MSB to LSB, as required by LTC2620).
271                         {
272                             IOB = IOB | 0x02;           //SDIHI
273                             IOB = IOB | 0x01;           //SCKHI
274                             IOB = IOB & ~(0x01);        //SCKLOW
275                         }
276                         else
277                         {
278                             IOB = IOB & ~(0x04);        //CSLDLOW
279                             IOB = IOB & ~(0x02);        //SDILOW
280                             IOB = IOB | 0x01;           //SCKHI
281                             IOB = IOB & ~(0x01);        //SCKLOW
282                         }
283                     }
284                 }
285                 for (i=0;i<150;i=i+1)
286                 {
287                     IOB = IOB | 0x04;           //CSLDHI
288                 }
289                 IOD = 0x00;//change back to 0x00;
290
291     break;
292 }
293 case VX_B6: // read GPIFTRIG register
294 {
295             OEB=0xFF;
296             IOD = 0x02; //Select SPI1.
297             EP0BCL = 0;       //EP0BUF LSB BYTE COUNT
298             while(EP01STAT & bmEP0BSY);     // wait until EP0 is available to be accessed by CPU
299             m = 0;
300             for (i=0;i<150;i=i+1)
301             {
302                 IOB = IOB | 0x04;           //CSLDHI
303             }
304             //EZUSB_Delay(1);
305             IOB = IOB & ~(0x01);        //SDILOW
306                 for (i=0;i<8;i=i+1)             // Loop 64 times: from 0xE740 to 0xE77F memory bytes(the memory addresses of EP0BUF).
```

```
307                 {
308                     for(j=0;j<150;j++)
309                     {
310                         IOB = IOB & ~(0x04);        //CSLDLOW
311                     }
312                     IOB = IOB & ~(0x02);        //SDILOW
313 
314                     for(k=0;k<8;k++)
315                     {
316                         IOB = IOB & ~(0x04);        //CSLDLOW
317                         if(EP0BUF[i] & BITMASK[7-k])        //BitMask EP0BUF Bytes (7-k to force MSB to LSB, as required by LTC262
318                         {
319                             IOB = IOB | 0x02;           //SDIHI
320                             IOB = IOB | 0x01;           //SCKHI
321                             IOB = IOB & ~(0x01);        //SCKLOW
322                         }
323                         else
324                         {
325                             IOB = IOB & ~(0x04);        //CSLDLOW
326                             IOB = IOB & ~(0x02);        //SDILOW
327                             IOB = IOB | 0x01;           //SCKHI
328                             IOB = IOB & ~(0x01);        //SCKLOW
329                         }
330                     }
331                 }
332                 for (i=0;i<150;i=i+1)
333                 {
334                     IOB = IOB | 0x04;           //CSLDHI
335                 }
336                 IOD = 0x00;
337         break;
338     }
339 //SET INTEGRATION_TIME
340     case VX_B7: // read GPIFTRIG register
341     {
342         OEB=0xFF;
343         IOD = 0x04;             //Select WR_EXT LSB
344         EP0BCL = 0;             //EP0BUF LSB BYTE COUNT
345         while(EP01STAT & bmEP0BSY);     // wait until EP0 is available to be accessed by CPU
346         for (i=0;i<10;i=i+1){}  //WAIT FOR SOME TIME.
347         IOB = EP0BUF[0];
348         for (i=0;i<10;i=i+1){}  //WAIT FOR SOME TIME.
349         IOD = 0x0C;             //Select WR_EXT MSB
350         for (i=0;i<10;i=i+1){}  //WAIT FOR SOME TIME.
351         IOB = EP0BUF[1];
352         for (i=0;i<10;i=i+1){}  //WAIT FOR SOME TIME.
353         IOD = 0x00;
354         for (i=0;i<10;i=i+1){}  //WAIT FOR SOME TIME.
355         IOB = 0x00;
356     break;
357     }
```

```
//SET RESET_TIME
    case VX_B8: // read GPIFTRIG register
    {
        OEB=0xFF;
        IOD = 0x05;             //Select WR_EXT LSB
        EP0BCL = 0;             //EP0BUF LSB BYTE COUNT
        while(EP01STAT & bmEP0BSY);     // wait until EP0 is available to be accessed by CPU
        for (i=0;i<10;i=i+1){}  //WAIT FOR SOME TIME.
        IOB = EP0BUF[0];
        for (i=0;i<10;i=i+1){}  //WAIT FOR SOME TIME.
        IOD = 0x0D;             //Select WR_EXT MSB
        for (i=0;i<10;i=i+1){}  //WAIT FOR SOME TIME.
        IOB = EP0BUF[1];
        for (i=0;i<10;i=i+1){}  //WAIT FOR SOME TIME.
        IOD = 0x00;
        for (i=0;i<10;i=i+1){}  //WAIT FOR SOME TIME.
        IOB = 0x00;

      break;
    }
//SET FREEZE_RAMP_TIME
    case VX_B9: // read GPIFTRIG register
    {

        OEB=0xFF;
        IOD = 0x06;             //Select WR_EXT LSB
        EP0BCL = 0;             //EP0BUF LSB BYTE COUNT
        while(EP01STAT & bmEP0BSY);     // wait until EP0 is available to be accessed by CPU
        for (i=0;i<10;i=i+1){}  //WAIT FOR SOME TIME.
        IOB = EP0BUF[0];
        for (i=0;i<10;i=i+1){}  //WAIT FOR SOME TIME.
        IOD = 0x0E;             //Select WR_EXT MSB
        for (i=0;i<10;i=i+1){}  //WAIT FOR SOME TIME.
        IOB = EP0BUF[1];
        for (i=0;i<10;i=i+1){}  //WAIT FOR SOME TIME.
        IOD = 0x00;
        for (i=0;i<10;i=i+1){}  //WAIT FOR SOME TIME.
        IOB = 0x00;

      break;
    }
//SET WR_EXT_HIT_TIME
    case VX_BB: // read GPIFTRIG register
    {
        OEB=0xFF;
        IOD = 0x08;             //Select WR_EXT LSB
        EP0BCL = 0;             //EP0BUF LSB BYTE COUNT
        while(EP01STAT & bmEP0BSY);     // wait until EP0 is available to be accessed by CPU
        for (i=0;i<10;i=i+1){}  //WAIT FOR SOME TIME.
        IOB = EP0BUF[0];
        for (i=0;i<10;i=i+1){}  //WAIT FOR SOME TIME.
```

```
409             IOD = 0x09;             //Select WR_EXT MSB
410             for (i=0;i<10;i=i+1){}  //WAIT FOR SOME TIME.
411             IOB = EP0BUF[1];
412             for (i=0;i<10;i=i+1){}  //WAIT FOR SOME TIME.
413             IOD = 0x00;
414             for (i=0;i<10;i=i+1){}  //WAIT FOR SOME TIME.
415             IOB = 0x00;
416         break;
417       }
418
419       default:
420          return(TRUE);
421     }
422     return(FALSE);
423 }
424 //-----------------------------------------------------------------------------
425 // USB Interrupt Handlers
426 //   The following functions are called by the USB interrupt jump table.
427 //-----------------------------------------------------------------------------
428
429 // Setup Data Available Interrupt Handler
430 void ISR_Sudav(void) interrupt 0
431 {
432     GotSUD = TRUE;            // Set flag
433     EZUSB_IRQ_CLEAR();
434     USBIRQ = bmSUDAV;         // Clear SUDAV IRQ
435 }
436 // Setup Token Interrupt Handler
437 void ISR_Sutok(void) interrupt 0
438 {
439     EZUSB_IRQ_CLEAR();
440     USBIRQ = bmSUTOK;         // Clear SUTOK IRQ
441 }
442 void ISR_Sof(void) interrupt 0
443 {
444     EZUSB_IRQ_CLEAR();
445     USBIRQ = bmSOF;            // Clear SOF IRQ
446 }
447 void ISR_Ures(void) interrupt 0
448 {
449     // whenever we get a USB reset, we should revert to full speed mode
450     pConfigDscr = pFullSpeedConfigDscr;
451     ((CONFIGDSCR xdata *) pConfigDscr)->type = CONFIG_DSCR;
452     pOtherConfigDscr = pHighSpeedConfigDscr;
453     ((CONFIGDSCR xdata *) pOtherConfigDscr)->type = OTHERSPEED_DSCR;
454
455     EZUSB_IRQ_CLEAR();
456     USBIRQ = bmURES;         // Clear URES IRQ
457 }
458 void ISR_Susp(void) interrupt 0
459 {
```

```
460         Sleep = TRUE;
461         EZUSB_IRQ_CLEAR();
462         USBIRQ = bmSUSP;
463     }
464     void ISR_Highspeed(void) interrupt 0
465     {
466         if (EZUSB_HIGHSPEED())
467         {
468             pConfigDscr = pHighSpeedConfigDscr;
469             ((CONFIGDSCR xdata *) pConfigDscr)->type = CONFIG_DSCR;
470             pOtherConfigDscr = pFullSpeedConfigDscr;
471             ((CONFIGDSCR xdata *) pOtherConfigDscr)->type = OTHERSPEED_DSCR;
472         }
473         EZUSB_IRQ_CLEAR();
474         USBIRQ = bmHSGRANT;
475     }
476     void ISR_Ep0ack(void) interrupt 0 {}
477     void ISR_Stub(void) interrupt 0 {}
478     void ISR_Ep0in(void) interrupt 0 {}
479     void ISR_Ep0out(void) interrupt 0 {}
480     void ISR_Ep1in(void) interrupt 0 {}
481     void ISR_Ep1out(void) interrupt 0 {}
482     void ISR_Ep2inout(void) interrupt 0 {}
483     void ISR_Ep4inout(void) interrupt 0 {}
484     void ISR_Ep6inout(void) interrupt 0 {}
485     void ISR_Ep8inout(void) interrupt 0 {}
486     void ISR_Ibn(void) interrupt 0 {}
487     void ISR_Ep0pingnak(void) interrupt 0 {}
488     void ISR_Ep1pingnak(void) interrupt 0 {}
489     void ISR_Ep2pingnak(void) interrupt 0 {}
490     void ISR_Ep4pingnak(void) interrupt 0 {}
491     void ISR_Ep6pingnak(void) interrupt 0 {}
492     void ISR_Ep8pingnak(void) interrupt 0 {}
493     void ISR_Errorlimit(void) interrupt 0 {}
494     void ISR_Ep2piderror(void) interrupt 0 {}
495     void ISR_Ep4piderror(void) interrupt 0 {}
496     void ISR_Ep6piderror(void) interrupt 0 {}
497     void ISR_Ep8piderror(void) interrupt 0 {}
498     void ISR_Ep2pflag(void) interrupt 0 {}
499     void ISR_Ep4pflag(void) interrupt 0 {}
500     void ISR_Ep6pflag(void) interrupt 0 {}
501     void ISR_Ep8pflag(void) interrupt 0 {}
502     void ISR_Ep2eflag(void) interrupt 0 {}
503     void ISR_Ep4eflag(void) interrupt 0 {}
504     void ISR_Ep6eflag(void) interrupt 0 {}
505     void ISR_Ep8eflag(void) interrupt 0 {}
506     void ISR_Ep2fflag(void) interrupt 0 {}
507     void ISR_Ep4fflag(void) interrupt 0 {}
508     void ISR_Ep6fflag(void) interrupt 0 {}
509     void ISR_Ep8fflag(void) interrupt 0 {}
510     void ISR_GpifComplete(void) interrupt 0 {}
```

```
154             EP6AUTOINLENL = 0x40;
155             SYNCDELAY;
156             enum_high_speed = FALSE;
157         }
158         Configuration = SETUPDAT[2];
159         return(TRUE);            // Handled by user code
160     }
161     BOOL DR_GetConfiguration(void)   // Called when a Get Configuration command is received
162     {
163         EP0BUF[0] = Configuration;
164         EP0BCH = 0;
165         EP0BCL = 1;
166         return(TRUE);            // Handled by user code
167     }
168     BOOL DR_SetInterface(void)       // Called when a Set Interface command is received
169     {
170         AlternateSetting = SETUPDAT[2];
171         return(TRUE);            // Handled by user code
172     }
173     BOOL DR_GetInterface(void)       // Called when a Set Interface command is received
174     {
175         EP0BUF[0] = AlternateSetting;
176         EP0BCH = 0;
177         EP0BCL = 1;
178         return(TRUE);            // Handled by user code
179     }
180     BOOL DR_GetStatus(void)
181     {
182         return(TRUE);
183     }
184     BOOL DR_ClearFeature(void)
185     {
186         return(TRUE);
187     }
188     BOOL DR_SetFeature(void)
189     {
190         return(TRUE);
191     }
192     //=========================================================================
193     #define VX_BB 0xBB // Set SPI Port
194     #define VX_B2 0xB2 // Set SPI Port
195     #define VX_B3 0xB3 // Set SPI Port
196     #define VX_B4 0xB4 // Set SPI Port
197     #define VX_B5 0xB5 // Set SPI Port
198     #define VX_B6 0xB6 // Set CPLD Outputs
199     #define VX_B7 0xB7 // Set SDAC Outputs
200     #define VX_B8 0xB8 // Read Temperature Sensor
201     #define VX_B9 0xB9 // Set Configuration Pixel Data
202     #define VX_BA 0xBA // Set pulse_counter @ Digital FPGA
203     BOOL DR_VendorCmnd(void)
204     {
```

8.3 Παράρτημα Γ – Κώδικας FPGA1

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
entity APIC_Analog is
    Port ( FDPORTB_D              : inout  STD_LOGIC_VECTOR (7 downto 0);
           ADDRESSBUS_IN          : in     STD_LOGIC_VECTOR (7 downto 0);
           CTRL_D                    : in    STD_LOGIC_VECTOR (1 downto 0);
           RDY_D                  : out    STD_LOGIC_VECTOR (5 downto 0);
           IFCLK_IN               : in     STD_LOGIC;
-------------------------SPI DAC SIGNALS-------------------------------------------------------
           SPI_SDI1               : out    STD_LOGIC;
           SPI_SCK1               : out    STD_LOGIC;
           SPI_CSLD1              : out    STD_LOGIC;
           SPI_CLR1               : out    STD_LOGIC;
           SPI_SDI2               : out    STD_LOGIC;
           SPI_SCK2               : out    STD_LOGIC;
           SPI_CSLD2              : out    STD_LOGIC;
           SPI_CLR2               : out    STD_LOGIC;
-------------------------ADC AND RESET RAMPS SIGNALS-------------------------------------------
           RESET_RAMP_SET      : out   STD_LOGIC;
           RESET_RAMPEN           : out   STD_LOGIC;
           ADC_RAMP_RESET      : out   STD_LOGIC;
           ADC_RAMP_EN            : out    STD_LOGIC;
-------------------------ASIC SIGNALS----------------------------------------------------------
           D_OUT                  : in  STD_LOGIC_VECTOR (7 downto 0);
           D_OUT1                 : in  STD_LOGIC_VECTOR (7 downto 0);
           D_IN                   : in  STD_LOGIC_VECTOR (7 downto 0);
           T3_EN                  : out    STD_LOGIC;
           WR_EXT                 : out    STD_LOGIC;
           RESET_DIG              : out    STD_LOGIC;
           APIC_CLK_OUT           : out    STD_LOGIC;
           LED                    : out    STD_LOGIC;
           BUTTON                : in     STD_LOGIC;
           RESET2                 : out    STD_LOGIC;
           SAMPLE              : out   STD_LOGIC;
           HOLD                   : out    STD_LOGIC;
           PD_DG               : out   STD_LOGIC);
-----------------------------------------------------------------------------------------------
end APIC_Analog;
```

```
architecture Behavioral of APIC_Analog is
signal ifclkcounter     : STD_LOGIC_VECTOR (3 downto 0);
signal clkcounter       : STD_LOGIC_VECTOR (15 downto 0);
signal apic_clk         : STD_LOGIC;
signal INTEGRATION_TIME : STD_LOGIC_VECTOR (15 downto 0);
signal RESET_TIME       : STD_LOGIC_VECTOR (15 downto 0);
signal FREEZE_RAMP_TIME : STD_LOGIC_VECTOR (15 downto 0);
signal WR_EXT_HIT_TIME  : STD_LOGIC_VECTOR (15 downto 0);
type DATA_OUT_ARRAY is array(0 to 19) of std_logic_vector(7 downto 0);
signal DATA_OUT         : DATA_OUT_ARRAY;
signal i : integer range -1 to 20;
signal k : integer range 0 to 20;
signal j : integer range 0 to 20;
signal startdatatransfer : STD_LOGIC;
signal framecount : integer range 0 to 200;
begin
    T3_EN                   <= '1'; --for T3_EN=1 DIN  bus is off.
    WR_EXT              <= '0';
    RESET_RAMP_SET      <= '0';
    PD_DG                   <= '0'; --for PD_DG=0 comparators are on.
    SPI_CLR1            <= '1';
    SPI_CLR2            <= '1';
    LED                 <= NOT BUTTON;
```

```
process (IFCLK_IN)
begin
    if rising_edge(IFCLK_IN) then
        if CTRL_D(0)='0' then
            j<=0;
            FDPORTB_D <=(others => 'Z');
            case ADDRESSBUS_IN is
                when "00000001" =>   --SET SPI1
                    SPI_SCK1        <= FDPORTB_D(0);
                    SPI_SDI1        <= FDPORTB_D(1);
                    SPI_CSLD1       <= FDPORTB_D(2);
                when "00000010" =>  --SET SPI2
                    SPI_SCK2        <= FDPORTB_D(0);
                    SPI_SDI2        <= FDPORTB_D(1);
                    SPI_CSLD2       <= FDPORTB_D(2);
                when "00000100" =>  --SET INTEGRATION_TIME
                    INTEGRATION_TIME(7 downto 0)  <= FDPORTB_D;
                when "00001100" =>  --SET INTEGRATION_TIME
                    INTEGRATION_TIME(15 downto 8) <= FDPORTB_D;
                when "00000101" =>  --SET   RESET_TIME
                    RESET_TIME(7 downto 0)  <= FDPORTB_D;
                when "00001101" =>  --SET   RESET_TIME
                    RESET_TIME(15 downto 8) <= FDPORTB_D;
                when "00000110" =>  --SET   FREEZE_RAMP_TIME
                    FREEZE_RAMP_TIME(7 downto 0)  <= FDPORTB_D;
                when "00001110" =>  --SET   FREEZE_RAMP_TIME
                    FREEZE_RAMP_TIME(15 downto 8) <= FDPORTB_D;
                when "00001000" =>--0x08
                    WR_EXT_HIT_TIME(7 downto 0) <= FDPORTB_D;
                when "00001001" =>--0x09
                    WR_EXT_HIT_TIME(15 downto 8) <= FDPORTB_D;
                when others =>
            end case;
        elsif CTRL_D(0)='1' then --to kanw asso oso thelw na pairnw data --So as not to double drive the FD
            if j<20 then
                FDPORTB_D(7) <= (NOT DATA_OUT(j)(7));
                FDPORTB_D(6) <= (NOT DATA_OUT(j)(6)) XOR (NOT DATA_OUT(j)(7));
                FDPORTB_D(5) <= (NOT DATA_OUT(j)(5)) XOR (NOT DATA_OUT(j)(6)) XOR (NOT DATA_OUT(j)(7));
                FDPORTB_D(4) <= (NOT DATA_OUT(j)(4)) XOR (NOT DATA_OUT(j)(5)) XOR (NOT DATA_OUT(j)(6)) XOR
                FDPORTB_D(3) <= (NOT DATA_OUT(j)(3)) XOR (NOT DATA_OUT(j)(4)) XOR (NOT DATA_OUT(j)(5)) XOR
                FDPORTB_D(2) <= (NOT DATA_OUT(j)(2)) XOR (NOT DATA_OUT(j)(3)) XOR (NOT DATA_OUT(j)(4)) XOR
                FDPORTB_D(1) <= (NOT DATA_OUT(j)(1)) XOR (NOT DATA_OUT(j)(2)) XOR (NOT DATA_OUT(j)(3)) XOR
                FDPORTB_D(0) <= (NOT DATA_OUT(j)(0)) XOR (NOT DATA_OUT(j)(1)) XOR (NOT DATA_OUT(j)(2)) XOR
                j<=j+1;
            end if;
        end if;
```

```
        ifclkcounter <= ifclkcounter+1;
        if ifclkcounter=x"0" then
            APIC_CLK_OUT<='1';
            apic_clk<='1';
        elsif ifclkcounter=x"8" then
            APIC_CLK_OUT<='0';
            apic_clk<='0';
        end if;
    end if;
end process;
```

```
    process (apic_clk)
    begin
        if rising_edge(apic_clk) then
            clkcounter <= clkcounter+1;
            if clkcounter=x"000" then --Start of Integration Phase
                RDY_D(0) <='1';
                SAMPLE <='1';
                HOLD <= '0';
                RESET2<= '1';   --0     --RESET2(Non linear reset)
                RESET_RAMPEN<= '1';--0  --RESET(linear reset)
                startdatatransfer<='1';
            elsif clkcounter = (INTEGRATION_TIME - x"3") then
                RESET_DIG<= '1';
            elsif clkcounter = INTEGRATION_TIME then --Start of ADC Phase.
                RDY_D(0) <='0';
                SAMPLE <='0';
                HOLD <='1';
                RESET_DIG<= '0';        --RESETS THE DIGITAL CHAIN [ACTIVE HI].
                ADC_RAMP_RESET<= '1';   --if =1 off, if=0 on. Start Ramp.
                ADC_RAMP_EN <='0';      --if =1 off, if=0 on.
                RESET2<= '1';--1        --RESET2
                RESET_RAMPEN<= '1';--1  --RESET
                startdatatransfer<='0';
            elsif clkcounter = (INTEGRATION_TIME + RESET_TIME) then --Cancel the RESET Signals
                RESET2<= '1';--0            --RESET2
                RESET_RAMPEN<= '1';--0      --RESET
            elsif clkcounter = (INTEGRATION_TIME + x"0FF") then
                ADC_RAMP_RESET<= '1';--if =1 off, if=0 on. TELEIWMA ANODOU RAMPAS.
                ADC_RAMP_EN <='1';
                RESET2<= '1';
                RESET_RAMPEN<= '1';
                --elsif clkcounter = (INTEGRATION_TIME + x"0FE"+x"11") then
                --RDY_D(4)<='0';
            elsif clkcounter = (INTEGRATION_TIME + x"0FF" + FREEZE_RAMP_TIME) then --ZERO RAMP AND COUNTER.
                ADC_RAMP_EN <='1';
                ADC_RAMP_RESET<= '0';
                RESET2<= '1';--0            --RESET2
                RESET_RAMPEN<= '1';--0  --RESET
                --PD_DG <='1';
            elsif clkcounter = (INTEGRATION_TIME + x"0FF" + FREEZE_RAMP_TIME + x"12") then
                clkcounter <= (others => '0');
            end if;
        end if;
    end process;

-------GET THE DATA AND STORE THEM INSIDE AN ARRAY-----------------
    process (apic_clk)
    begin
        if falling_edge(apic_clk) then
            if clkcounter=(INTEGRATION_TIME + x"0FF") then
            --      WR_EXT<='0';
                i <= -1;
                k <= 0;
            --elsif clkcounter = (INTEGRATION_TIME + WR_EXT_HIT_TIME) then
            --  DATA_OUT(0)  <= NOT D_IN;
            elsif (clkcounter>=(INTEGRATION_TIME + x"100")) AND (clkcounter<=(INTEGRATION_TIME + x"113")) then
                i <= i+1;
                if  i mod 2 = 0 then
                    RDY_D(5)<='0';
                elsif i mod 2 = 1 then
                    DATA_OUT(19-k) <= D_OUT;
                    RDY_D(5)<='1';
                    DATA_OUT(9-k)  <= D_OUT1;
                    k<=k+1;
                end if;
            end if;
        end if;
    end process;
end Behavioral;
```

8.4 Παράρτημα Δ – Κώδικας FPGA2

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity APIC_Digital is

        Port ( FDPORTB_D            : inout  STD_LOGIC_VECTOR (7 downto 0);
               FDPORTB_A            : inout  STD_LOGIC_VECTOR (7 downto 0);
               ADDRESSBUS_IN        : in     STD_LOGIC_VECTOR (7 downto 0);
               ADDRESSBUS_OUT       : out     STD_LOGIC_VECTOR (7 downto 0);
               IFCLK_IN             : in    STD_LOGIC;
               IFCLK_OUT            : out    STD_LOGIC;
               CTRL_D               : in    STD_LOGIC_VECTOR (4 downto 0);
               RDY_D                : out   STD_LOGIC_VECTOR (5 downto 0);
               CTRL_A               : out   STD_LOGIC_VECTOR (4 downto 0);
               RDY_A                : in    STD_LOGIC_VECTOR (5 downto 0);
               LED                  : out    STD_LOGIC);

end APIC_Digital;

architecture Behavioral of APIC_Digital is

type DATA_OUT_ARRAY is array(0 to 19) of std_logic_vector(7 downto 0);
signal DATA_OUT  : DATA_OUT_ARRAY;
signal j : integer range 0 to 20;
signal wordscounter : integer range 0 to 20;
signal starttransfer : STD_LOGIC :='1';

type STATE is (S0, S1, S2);
signal CURRENT_STATE, NEXT_STATE: STATE;
begin

ADDRESSBUS_OUT(5 downto 0)<=ADDRESSBUS_IN(5 downto 0);
IFCLK_OUT <= IFCLK_IN;
```

```
61      process(IFCLK_IN)
62      begin
63
64      if (falling_edge(IFCLK_IN)) then
65          CURRENT_STATE <= NEXT_STATE;
66          case CURRENT_STATE is
67              when S0 =>
68                  LED<='0';
69                  j<=0;
70                  FDPORTB_A <= FDPORTB_D;
71                  if (ADDRESSBUS_IN(7) = '1') then
72                      if RDY_A(0)='1' then
73                          CTRL_A(0)<='1';--READ ENABLE SENT TO FPGA_A
74                          FDPORTB_D<=(others=>'Z');
75                          FDPORTB_A<=(others=>'Z');
76                          NEXT_STATE <= S1;
77                      elsif RDY_A(0)='0' then
78                          NEXT_STATE <= S0;
79                      end if;
80                  elsif (ADDRESSBUS_IN(7) = '0') then
81                      CTRL_A(0)<='0';--READ ENABLE SENT TO FPGA_A
82                      NEXT_STATE <= S0;
83                  end if;
84              when S1 =>
85                  LED<='0';
86                  CTRL_A(0)<='1';--READ ENABLE SENT TO FPGA_A
87                  --DATA_OUT(j)<=FDPORTB_A;
88                  if j=19 then
89                      NEXT_STATE <= S2;
90                  else
91                      j<=j+1;
92                      NEXT_STATE <= S1;
93                  end if;
94              when S2 =>
95                      CTRL_A(0)<='0';--READ ENABLE SENT TO FPGA_A
96                      LED<='0';
97                      if starttransfer='1' then
98                          FDPORTB_D<=DATA_OUT(wordscounter);
99                          NEXT_STATE <= S2;
100                         LED<='1';
101                     elsif starttransfer='0' then
102                         NEXT_STATE <= S0;
103                         LED<='0';
104                         FDPORTB_D<=(others=>'Z');
105                     end if;
106             end case;
107         end if;
108     end process;
```

```
110     process (ADDRESSBUS_IN(6))
111 --  Se kathe rising edge tou ADDR(6) exoume diathesima dedomena sto bus
112 --  gia na ta diavasei to usb kai mexri to epomeno rising edge.
113     begin
114         if (ADDRESSBUS_IN(6)='1' and ADDRESSBUS_IN(6)'event) then
115             if wordscounter<19 then -- 0A1A2A3A4A5A6A7A8A9A10A11A12A13A14A15A16
116                 starttransfer<='1';
117                 wordscounter <= wordscounter + 1;
118             elsif wordscounter=19 then -- A17
119                 starttransfer<='0';
120                 wordscounter <= wordscounter + 1;
121             elsif wordscounter>19 then  -- A18
122                 starttransfer<='1';--A19
123                 wordscounter <= 0;--SEND 19 RISING EDGES OF ADDR6
124             end if;
125         end if;
126     end process;
127 end Behavioral;
```

8.5 Λίστα Εξαρτημάτων

	Value	Description	Package	Manufacturer	Manufacturer Part No	Quantity per pcb		
						Analog	Digital	Source
Capacitors	10p	CAP CER 10PF 50V 5% C0G 0603	0603	Murata	GRM1885C1H100JA01D	2	0	1
	18p	CAP CER 18PF 50V 5% C0G 0603	0603	Murata	GRM1885C1H180JA01D	0	2	0
	1.3n	CAP CER 1300PF 50V 5% C0G 0603	0603	Murata	GRM1885C1H132JA01D	1	0	0
	4.7n/250V	CAP CER 4700PF 250V C0G 5% 1206	1206	TDK	C3216C0G2E472J	0	1	0
	10n	CAP CER 10000PF 50V 10% X7R 0603	0603	Murata	GRM188R71H103KA01D	9	2	0
	100n	CAP CER .1UF 50V 10% X7R 0603	0603	Murata	GRM188R71H104KA93D	111	43	4
	2.2u	CAP TANT 2.2UF 10V 20% SMD	0805 (2012 metric)	Rohm	TCP1A225M8R	0	2	0
	4.7u	CAP TANTALUM 4.7UF 10V 20% SMD	2012-12 (EIA) 0805	Rohm	TCFGP1A475M8R	2	0	0
	6.8u	CAP TANT 6.8UF 6.3V 20% SMD	0805 (2012 metric)	Rohm	TCP0J685M8R	2	0	0
	10u	CAP CER 10UF 10V 10% X5R 0805	0805	Murata	GRM21BR61A106KE19L	2	0	1
	10u	CAP TANTALUM 10UF 10V 20% SMD	2012-15 (EIA) 0805	AVX	TAJP106M010R	14	12	0
	22u	CAP TANTALUM 22UF 4.0V 20% SMD	2012-12 (EIA) 0805	Rohm	TCFGP0G226M8R	2	0	0
	47u	CAP TANT 47UF 6.3V 20% SMD	1206 (3216 metric)	Rohm	TCA0J476M8R	3	3	0
	100u	CAP TANT 100UF 10V 10% LOESR SMD	7343-31 (EIA)	Kemet	B45197A2107K409	2	1	0
	TBD	IN_CSCR Defined by Zervakis	0603	-	-	2	0	0
Resistors	0R	RES 0.0 OHM 1/10W 5% 0603 SMD	0603	Yageo	RC0603JR-070RL	60	1	0
	10R	RES 10.0 OHM 1/10W 1% 0603 SMD	0603	Yageo	RC0603FR-0710RL	2	0	0
	21.81R	RES 22 OHM 1/16W .5% 0603 SMD	0603	Susumu	RR0816Q-220-D	2	0	0
	22R	RES 22.0 OHM 1/10W 1% 0603 SMD	0603	Yageo	RC0603FR-0722RL	0	13	0
	28.57R	RES 28.7 OHM 1/10W 1% 0603 SMD	0603	Yageo	RC0603FR-0728R7L	2	0	0
	75R	RES 75.0 OHM 1/10W 1% 0603 SMD	0603	Yageo	RC0603FR-0775RL	3	3	0
	100R	RES 100 OHM 1/10W 1% 0603 SMD	0603	Yageo	RC0603FR-07100RL	2	0	1
	200R	RES 200 OHM 1/10W .1% 0603 SMD	0603	Susumu	RG1608P-201-B-T5	4	0	0
	330R	RES 330 OHM 1/10W .1% 0603 SMD	0603	Panasonic	ERA-3AEB331V	0	0	1
	330	RES 330 OHM 1/10W 1% 0603 SMD	0603	Yageo	RC0603FR-07330RL	3	1	0
	1.2k	RES 1.2K OHM 1/16W .1% 0603 SMD	0603	Susumu	RR0816P-122-B-T5	4	0	0
	2k	RES 2.00K OHM 1/10W 1% 0603 SMD	0603	Yageo	RC0603FR-072KL	2	0	0
	2.2K	RES 2.20K OHM 1/10W 1% 0603 SMD	0603	Yageo	RC0603FR-072K2L	0	2	0
	3.3k	RES 3.3K OHM 1/10W .1% 0603 SMD	0603	Panasonic	ERA-3AEB332V	12	0	2
	4.7k	RES 4.70K OHM 1/10W 1% 0603 SMD	0603	Yageo	RC0603FR-074K7L	2	2	0
	16K	RES 16K OHM 1/10W .5% 0603 SMD	0603	Yageo	RT0603DRD0716KL	1	0	0
	10K	RES 10.0K OHM 1/10W 1% 0603 SMD	0603	Yageo	RC0603FR-0710KL	0	1	0
	33k	RES 33K OHM 1/10W .1% 0603 SMD	0603	Panasonic	ERA-3AEB333V	4	0	1
	50K	RES 49.9K OHM 1/10W 1% 0603 SMD	0603	Yageo	RC0603FR-0749K9L	1	0	0
	100K	RES 100K OHM 1/10W 1% 0603 SMD	0603	Yageo	RC0603FR-07100KL	0	1	0
	330k	RES 330K OHM 1/10W .1% 0603 SMD	0603	Panasonic	ERA-3AEB334V	2	0	0
	10M	RES 10.0M OHM 1/8W 1% 0805 SMD	0805	Yageo	RC0805FR-0710ML	0	1	0
	1Gig	RES 1.0G OHM 300MW 5% 1206	1206	Ohmite	HVF1206T1007JE	0	0	1
	49.9k	RES 49.9K OHM 1/10W .1% 0603 SMD	0603	Susumu	RG1608P-4992-B-T5	3	0	0
	16k	RES 16.0K OHM 1/10W .1% 0603 SMD	0603	Susumu	RG1608P-163-B-T5	2	0	0

ICs	TS5A4595	IC ANALOG SWITCH 5V SOT23-5	SOT23-5	TI	TS5A4595DBVR	4	0	0
	LTCHV6244CMS8	IC OP AMP DUAL R-R 8-MSOP	8-MSOP	Linear	LTC6244HVCMS8#PBF	4	0	1
	LMH6559	IC BUFFER HS CLOSED LOOP SOT23-5	SOT23-5	National	LMH6559MF/NOPB	11	0	0
	OPA2652U	IC OPAMP VOLT-FDBK DUAL 8-SOIC	8-SOIC	TI	OPA2652U	2	0	0
	ADuM3440	IC DIGITAL ISOLATOR 4INP 16SOIC	16-SOIC (7.5mm Width)	Analog	ADUM3440CRWZ	7	0	0
	DAC2904_48FP	IC DAC DUAL 14BIT 125MSPS 48TQFP	48-TQFP	TI	DAC2904Y/250	1	0	0
	LTC2620_SSOP	IC DAC OCTAL R-R 12BIT 16SSOP	16SSOP	Linear	LTC2620CGN#PBF	4	0	0
	24LC64	IC SERIAL EEPROM 64K 2.5V 8-SOIC	8-SOIC	Microchip	24LC64-I/SN	0	1	0
	XCF01SVOG20C	IC PROM SRL FOR 1M GATE 20-TSSOP	20-TSSOP	Xilinx	XCF01SVOG20C	1	1	0
	SPARTAN	IC SPARTAN-3AN FPGA 50K 144TQFP	144TQFP	Xilinx	XC3S50AN-4TQG144C	1	1	0
	CB3LV Osc. 40MHz	OSC 24.000 MHZ 3.3V SMD	SMD7.5X5.5MM	CTS	CB3LV-3C-24M0000	1	1	0
	REG104FA-5KTTT	IC LDO REG 1A 5.0V 5-DDPAK	DDPAK-5	TI	REG104FA-5KTTT	1	0	0
	REG104FA-3.3/500	IC LDO REG 1A 3.3V 5-DDPAK	DDPAK-5	TI	REG104FA-3.3/500	1	1	0
	TPS76912_SOT23	IC LDO REG ADJ 100MA SOT-23-5	SOT23-5	TI	TPS76912DBVT	1	1	0
	LM4128AMF-3.0	IC REF PREC MCRPWR 3.0V SOT23-5	SOT23-5	National	LM4128AMF-3.0/NOPB	2	0	0
	TPS3820-3.3	IC 2.93V SUPPLY MONITOR SOT23-5	SOT23-5	TI	TPS3820-33DBVT	0	1	0
	CY7C68013_100P	IC MCU USB PERIPH HI SPD 100TQFP	100-TQFP	Cypress	CY7C68013A-100AXC	0	1	0
	LM2991S/TO263	IC REG NEG LDO ADJ V 1A TO-263-5	DDPAK-5	National	LM2991S/NOPB	1	0	0
	REG104GA_3.3V_SOT223	IC LDO REG 3.3V 1A SOT223-6	SOT-223-6	TI	REG104GA-3.3	2	1	0
	PS9121/A	PHOTOCOUPLER HS DGTL OC OUT S	5-SOP	NEC	PS9121-A	1	0	0
	LT1763-3.3/8-SO	IC REG LDO 3.3V 500MA 8-SOIC	8-SOIC	Linear	LT1763CS8-3.3#PBF	0	1	0
	APIC/JLCC-84	CONN SOCKET PLCC 84POS TIN SMD	PLCC-84 SOCKET	Tyco	3-822516-6	1	0	0
Other	24MHz	CRYSTAL 24.00 MHZ 12PF SMD	11.4mmX4.8mm	ECS	ECS-240-12-5PX-TR	0	1	0
	8 HEADER-F/STR	CONN SOCKET STRIP 50PIN .100 STR	TH-Non STD	Mill-Max	801-43-050-10-001000	3	0	0
	8 HEADER-M/R/A	CONN HEADER 64POS .100 R/ANGLE	TH-Non STD	Mill-Max	800-10-064-20-001000	0,00	0,00	0,13
	CON3	CONN HEADER 50POS .100" SGL GOLD	TH-Non STD	Samtec	TSW-150-07-G-S	0,81	0,31	0,00
	CON6A	Do not Buy	-	-	-	1	0	0
	CON9	CONN HEADER 50POS .100" SGL GOLD	TH-Non STD	Samtec	TSW-150-07-G-S	0,20	0,20	0,00
	CONN_RCPT_2X50_R	CONN RECEPT R/A 100POS 1.27MM	TH-Non STD	Hirose	FX2-100S-1.27DS(71)	1	0	0
	CONN_RCPT_2X50_H	CONN HEADER R/A 100POS 1.27MM	TH-Non STD	Hirose	FX2-100P-1.27DS(71)	0	1	0
	LEDs BLUE	LED 1.6X0.8MM 470NM BLUE CLR SMD	0603	KingBright	APG1608QBC/D	1	2	0
	LEDS GREEN	LED 1.6X0.8MM 525NM GRN CLR SMD	0603	KingBright	APT1608ZGC	1	2	0
	LCD CONNECTOR	CONN FPC/FFC 30POS .5MM SMD GOLD		Hirose	FH19-30S-0.5SH(05)	0	0	0
	LCD Panel	LCD GRAPH MOD 128X64 COG WHT LED		Optex	F-51320GNB-LW-AEN	0	1	0
	INDUCTOR FB	TBD	0603	-	-	4	0	0
	MINI USB-B	CONN RECEPT MINI USB2.0 5POS		Hirose	UX60-MB-5S8	0	1	0
			pfl8t	Perancea				
	BUTTON	SWITCH TACT SPST-NO 120GF GW SMD		C&K	KMR211GLFS	1	3	0

8.6 Βιβλιογραφικές Αναφορές

Βιβλία

1. "Design of Analog CMOS Integrated Circuits", Behzad Razavi, εκδόσεις McGraw-Hill 2000.
2. "Fundamentals of Microelectronics" , Behzad Razavi, εκδόσεις McGraw-Hill 2008.
3. "Microelectronic Circuits", Adel S. Sedra, Kenneth C. Smith, Oxford University Press 2009.
4. "Foundations of Analog and Digital Electronic Circuits", Anant Agarwal, Morgan Kaufmann 2005.
5. "Operational Amplifier Circuits: Design and Applications", D.E. Johnson et al, Prentice Hall 1982.
6. "CMOS Circuit Design, Layout, and Simulation" ,R. Jacob Baker, Wiley-IEEE Press 2007.
7. "Αναλογικά κυκλώματα VLSI" προσχέδιο σημειώσεων, Γιάννης Τσιβίδης, ΕΜΠ 1992.
8. "Operation and Modeling of the MOS Transistor", Yannis Tsividis and Colin McAndrew, Oxford University Press 2010.
9. "Αρχές Ηλεκτρονικών Υλικών και Διατάξεων", S.O. Kasap, Παπασωτηρίου 2002.
10. "Principles of Semiconductor Devices", Sima Dimitrijev, Oxford University Press 2006.
11. "Φυσική των Ηλεκτρονικών Διατάξεων ", Τσουκαλάς Δημήτριος, Σημειώσεις ΕΜΠ 2007.
12. "Physics of Semiconductor Devices" ,Simon M. Sze, Wiley-Interscience 2006.
13. "CdTe and Related Compounds; Physics, Defects, Hetero- and Nano-structures, Crystal Growth, Surfaces and Applications", Robert Triboulet (Editor), Paul Siffert (Editor)

14. "Semiconductor Detector Systems", Helmuth Spieler, Oxford University Press 2005.
15. Synopsys Sentaurus Manuals

16. "Pixel Detectors: From Fundamentals to Applications", Leonardo Rossi et al, Springer 2010.
17. "Radiation Detection and Measurement", Glenn F. Knoll, Wiley 2010.
18. "Complete PCB Design Using OrCAD Capture and PCB Editor", Kraig Mitzner, Newnes 2009.
19. "Complete PCB Design Using OrCAD Capture and Layout", Kraig Mitzner, Newnes 2007.
20. "Ψηφιακά Συστήματα, Μοντελοποίηση και προσομοίωση με την Γλώσσα VHDL", Σταύρος Σουράβλας, Μάνος Ρουμελιώτης, Τζιόλας 2008.
21. "The Designer's Guide to VHDL", Peter J. Ashenden, Morgan Kaufmann 2008.
22. "Digital Design (VHDL): An Embedded Systems Approach Using VHDL" Peter J. Ashenden**,** Morgan Kaufmann 2007.

Δημοσιεύσεις

1. "Calculation of pixel detector capacitances through three dimensional numerical solution of the Laplace equation", Kavadias, S., Misiakos, K, Loukas, D. Nuclear Science, IEEE Transactions on, Volume: 41 , Issue: 2, Page(s): 397 – 401.
2. "Charge integrating ASIC with pixel level A/D conversion", Lambropoulos, C.P. , Zervakis, E.G. , Loukas, D. , Nuclear Science Symposium Conference Record, 2007. NSS '07. IEEE, Volume: 1, Page(s): 357 – 359.
3. "A CMOS area image sensor with pixel-level A/D conversion", Solid-State Circuits Conference, 1994. Digest of Technical Papers. 41st ISSCC., 1994 IEEE International, Fowler, B., El Gamal, A., Yang, D.X.D., Page(s): 226 – 227.

4. “A 128×128 pixel CMOS area image sensor with multiplexed pixel level A/D conversion”, Custom Integrated Circuits Conference, 1996., Proceedings of the IEEE 1996, Yang, D.X.D., Fowler, B., El Gamal, A., Page(s): 303 - 306
5. The COCAE Detector: An Instrument for Localization - Identification of Radioactive Sources, C.P. Lambropoulos, T. Aoki, J. Crocco, E. Dieguez, C. Disch, A. Fauler, M. Fiederle, D.S. Hatzistratis, V.A. Gnatyuk, K. Karafasoulis6, L.A. Kosyachenko, S.N. Levytskyi, D. Loukas, O.L. Maslyanchuk, A. Medvids, T. Orphanoudakis, I. Papadakis, A. Papadimitriou, C. Potiriadis, T. Schulman, V.M. Sklyarchuk, K. Spartiotis, G. Theodoratos, O.I. Vlasenko, K. Zachariadou, M. Zervakis, , IEEE Transactions on Nuclear Science, Vol. 58, No. 5, Oct. 2011, pp. 2363-2370.
6. “Pixel electronics for a hybrid X/γ- ray imager”, C. T. Lambropoulos, E.Zervakis, G. Theodoratos, A. Nikolgiannis, D. Xatzistratis, I. Papadakis, A. Papadimitriou, D. Loukas, The conference on Hard X-Ray, Gamma-Ray, and Neutron Detector Physics XII (OP324), Part of the SPIE International Symposium on SPIE Optics Engineering & Applications, SPIE Optics & Photonics, 2010, 1-5 August 2010, San Diego, CA, USA.
7. "Technology developments for a Cd(Zn)Te detector for the localization identification of radioactive sources", C. P. Lambropoulos, T. Aoki, J. Crocco, E. Dieguez, C. Disch, A. Fauler, M.Fiederle, D. S. Hatzistratis, V. A. Gnatyuk, K. Karafasoulis, L. A. Kosyachenko, S. N. Levytskyi, D. Loukas, O. L. Maslyanchuk, A. Medvids, T. Orphanoudakis, I. Papadakis, A. Papadimitriou, K. Papakonstantinou, C. Potiriadis, T. Schulman, V. M. Sklyarchuk, K. Spartiotis, G. Theodoratos, O.I. Vlasenko, K. Zachariadou, M. Zervakis, , E-MRS 2010 Fall Meeting, New materials and homeland security Symposium, 14-16 Sep. 2010, Warsaw, Poland.
8. “Electric Field Properties of CdTe Nuclear Detectors”, Nuclear Science Symposium Conference Record, 2006. IEEE, Cola, A. Farella, I., Mancini, A.M. Donati, A., Volume: 6, Page(s): 3772 - 3777

9. “Electric field distribution and charge transport properties in diode-like CdTe X-ray detectors”, Adriano Cola, Isabella Farella, A. Maria Mancini, Waldes Dusi, Eugenio Perillo, 10th European Symposium on Semiconductor Detectors.
10. “Performance limits of a 55 μm pixel CdTe detector”, Pellegrini, G., Chmeissani, M. ; Maiorino, M, Blanchot, G., Garcia, J., Lozano, M., Martinez, R., Puigdengoles, C., Ullan, M., Casado, P., Volume: 4, Page(s): 2104 - 2109 Vol. 4
11. “Development of a simplified simulation model for performance characterization of a pixellated CdZnTe multimodality imaging system”, P Guerra, A Santos and D G Darambara, Guerra et al 2008 Phys. Med. Biol. 53 1099.
12. “Techniques for small-signal analysis of semiconductor devices” Laux S.E., Electron Devices, IEEE Transactions on, Volume: 32 , Issue: 10 Page(s): 2028 – 2037.
13. “ Capacitance of silicon pixels”,Grant Gorfine, Martin Hoeferkamp, Geno Santistevan, Sally Seidel, Nuclear Instruments and Methods in Physics Research Section A, Volume 460, Issues 2–3, 21 March 2001, Pages 336–351.
14. "Concentration of uncompensated impurities as a key parameter of CdTe and CdZnTe crystals for Schottky diode x/γ-ray detectors", L A Kosyachenko, C P Lambropoulos, T Aoki, E Dieguez, M Fiederle, D Loukas, O V Sklyarchuk, O L Maslyanchuk, E V Grushko, V M Sklyarchuk, J Crocco and H Bensalah, , Semicond. Sci. Technol. 27 (2012) 015007 (11pp) doi:10.1088/0268-1242/27/1/015007.