ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικόν και Καποδιστριακόν Πανεπιστήμιον Αθηνών
ΙΔΡΥΘΕΝ ΤΟ 1837

Δ.Π.Μ.Σ. :Ηλεκτρονικός Αυτοματισμός

Τμημάτων Φυσικής/Πληροφορικής

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη εφαρμογών για επεξεργασία εικόνας σε FPGA.

Ονοματεπώνυμο:Σαντοριναίος Αλέξανδρος

Α.Μ: 7110132100215

Επιβλέπων: Διονύσιος Ι. Ρεΐσης ,Καθηγητής

Τριμελής επιτροπή:

Διονύσιος Ι. Ρεΐσης ,Καθηγητής

Νυσταζάκης Έκτορας, Καθηγητής

Τζανακάκη Άννα, Αναπληρώτρια Καθηγήτρια

ΑΘΗΝΑ

[06/2023]

Περίληψη

Η παρακάτω πτυχιακή εργασία εξετάζει την ανίχνευσης αιχμών (egde detection), που αποτελεί στοιχειώδη τεχνική επεξεργασίας εικόνας(image processing) του τομέα του computer vision.

Συγκεκριμένα, θα εξετασθεί η λειτουργία του αλγορίθμου Sobel και η υλοποίησή του στο λογισμικό matlab και στην πλατφόρμα ανάπτυξης λογισμικού για FPGA, Vivado.

Η διαδικασία της υλοποίησής του θα χωριστεί σε δύο στάδια.

Στο πρώτο στάδιο θα κατασκευαστεί ένα bit-accurate model ή bit-true implementation στο λογισμικό matlab. Το μοντέλο αυτό θα αποτελέσει μια απλή υλοποίηση του αλγορίθμου , καθώς και μέθοδος επιβεβαίωσης των αποτελεσμάτων, που θα προκύψουν από την επεξεργασία δεδομένων στο FPGA. Επιπλέον στο matlab θα πραγματοποιηθούν και διαδικασίες προ-επεξεργσίας δεδομένων, όπως μετατροπή εικόνας σε gray-scale και μετατροπή δεδομένων εισόδου, σε μορφή κατάλληλη για χρήση στο FPGA.

Στη συνέχεια, ο ίδιος αλγόριθμος θα υλοποιηθεί στη γλώσσα περιγραφής υλικού(HDL) VHDL. Για τη σύνδεση των δομικών μονάδων (entities) θα χρησιμοποιηθεί το πρωτόκολλο επικοινωνίας AXI4-Stream.

Τέλος τα δεδομένα που θα προκύψουν από την επεξεργασία στο FPGA, θα συγκριθούν με αυτά του matlab, με σκοπό την επιβεβαίωση της ορθότητας της υλοποίησης.

Λέξεις Κλειδία :

FPGA,SOBEL,MATLAB,AXI4-STREAM,VHDL

Abstract

The following thesis examines edge detection, which is a fundamental technique in image processing within the field of computer vision.

To be more precise, this thesis will examine the operation of the Sobel algorithm and it's implementation in Matlab and on the FPGA software development platform, Vivado.

The implementation process will be divided into two stages.

In the first stage, a bit-accurate model or bit-true implementation will be constructed in MATLAB software. This model will serve as a simple implementation of the algorithm, as well as a method for verifying the results obtained from data processing on the FPGA. Additionally, preprocessing procedures will be performed in MATLAB, such as converting the image to grayscale and transforming input data into a suitable format for use on the FPGA.

Next, the same algorithm will be implemented in the hardware description language (HDL) VHDL. The AXI4-Stream communication protocol will be used to connect the structural units (entities). Finally, the FPGA processed data will be compared with the MATLAB results to verify the correctness of the implementation.

Key Words :

FPGA,SOBEL,MATLAB,AXI4-STREAM,VHDL

Περιεχόμενα

Κεφάλαιο 1: Στοιχεία Θεωρίας

1.1 FPGA

Το FPGA ή field programmable gate array, είναι ένας τύπος ολοκληρωμένου κυκλώματος(integrated circuit) , το οποίο διαθέτει ένα σύνολο από λογικές πύλες και λογικές μονάδες. Ο χρήστης μπορεί να προγραμματίσει το FPGA ανάλογα με τις ανάγκες του. Για τον προγραμματισμό των FPGAs χρησιμοποιείται μια γλώσσα περιγραφής υλικού, όπως η VHDL ή η Verilog. Τα FPGAs κατέχουν σημαντικό ρόλο στον τομέα των ενσωματωμένων συστημάτων, καθώς μπορούν να χρησιμοποιηθούν για την προσομοίωση και τον έλεγχο της αρχιτεκτονικης διαφόρων συστημάτων. Για παράγειδμα, τα FPGAs, χρησιμοποιούνται για την προσομοίωση της λειτουργίας πρωτοτύπων ASICs(application specific integrated circuit). Ο χρήστης σχεδιάζει την αρχιτεκτονική που επιθυμεί να προσομείωσει στο FPGA, ελέγχει τη λειτουργία του project του και μπορεί να διορθώνει άμεσα τυχόν σφάλματα που θα ανιχνεύσει. Όταν η αρχιτεκντονική συμπεριφέρεται όπως ακριβώς θα ήθελε, τότε μπορεί να προβεί στο στάδιο της χάραξης. Μέσω της διαδικασίας της προσομοίωσης, ο χρήστης θα έχει εξοικονομήσει χρήματα και χρόνο, μιας και η κατασκευή ASICs είναι χρονοβόρα και δαπανηρή διαδικασία.

1.2 Γλώσσα περιγραφής υλικού (HDL)

Οι γλώσσες περιγραφής υλικού (hardware description language ή hdl για συντομία) είναι γλώσσες προγραμματισμού,που χρησιμοποιούνται για να περιγράψουν τη συμπεριφορά και τη λειτουργία ψηφιακών συστημάτων.Οι 2 πιο γνωστές είναι οι Verilog και η VHDL.

1.3 Γλώσσα VHDL

Η γλώσσα VHDL(Very High-Speed Integrated Circuit Hardware Description Language) είναι μια γλώσσα περιγραφής υλικού, η οποία χρησιμοποιείται για την περιγραφή της δομής και της συμπεριφοράς ψηφιακών συστημάτων, σε επίπεδο καταχωρητή(Register transfer level/rtl). Η διαφορά της με την Verilog είναι ότι αυτή έχει περισσότερες απαιτήσεις για το πόσο σαφείς πρέπει να είναι οι λεπτομέριες του πρόγραμματός μας, πράγμα που δίνει και τη δυνατότητα μεγαλύτερου ελέγχου.

1.4 Πρωτόκολλο AXI4-Stream

1.4.1 Εισαγωγικές έννοιες του AXI4-Stream

Το πρωτόκολλο επικοινωνίας AXI4-Stream χρησιμοποιείται για τη διασύνδεση δομικών στοιχείων σε ένα σύστημα, τα οποία επιθυμούν να ανταλλάξουν μεταξύ τους μεγάλο όγκου δεδομένων. Αναπτύχθηκε από την ARM Holdings και χρησιμοποιείται σε εφαρμογές επεξεργασίας σημάτων (digital signal processing). Αποτελεί πιο απλή παραλλαγή του πρωτοκόλλου AXI4, χωρίς τα επιπλέον σήματα ελέγχου. Το AXI4-Stream υποστηρίζει τη διασύνδεση δύο ή και περισσότερων στοιχείων.

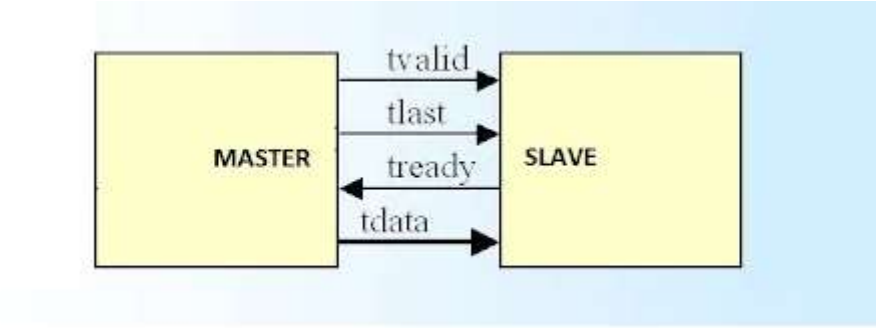
1.4.2 Σήματα του AXI4-Stream

Οι διασυνδέσεις(interfaces) του AXI4-Stream διακρίνονται σε 2 κατηγορίες, ανάλογα με το αν λαμβάνουν οι μεταδίδουν δεδομένα. Συγκεκριμένα

- Master Interface: Το interface που μεταδίδει τα δεδομένα.
- Slave Interface: Το Interface που λαμβάνει τα δεδομένα.

Τα σήματα από τα οποία αποτελείται μία διεπαφή είναι τα ακόλουθα :

ΟΝΟΜΑ ΣΗΜΑΤΟΣ	ΠΗΓΗ ΣΗΜΑΤΟΣ	ΛΕΙΤΟΥΡΓΙΑ ΣΗΜΑΤΟΣ
ACLK	Πηγή του ρολογιού	Ρολόι συστήματος
ARESETN	Εξωτερικοί διακόπτες	Ρυθμίζει την εκκίνηση του συστύματος
TVALID	Master Interface	Υποδηλώνει αν τα δεδομένα που μεταδίδει το master interface ενος στοιχείου είναι έγκυρα.
TREADY	Slave Interface	Υποδηλώνει αν το στοιχείο που διαθέτει το slave interface, είναι έτοιμο να λάβει δεδομένα από το master.
TDATA	Master Interface	Τα δεδομένα που μεταδίδονται από το master interface στο slave interface.
TLAST	Master Interface	Υποδηλώνει τη λήξη της μετάδωσης ενός stream ή ενός πακέτου δεδομένων.



Εικόνα 1: Τυπική σύνδεση ενός master με ένα slave

1.4.3 Περιγραφή του μηχανισμού λειτουργίας του AXI4-Stream

Η λειτουργία του AXI4-Stream βασίζεται σε ένα μηχανισμό με τη μορφή χειραψίας (handshake) μεταξύ master και slave, μέσω των σημάτων TVALID και TREADY. Αυτός ο μηχανισμός επιτρέπει στα δύο interfaces, να ελέγχουν το ρυθμό και την εγκυρότητα της μετάδοσης των δεδομένων.

Για να ξεκινήσει η μεταφορά δεδομένων, πρέπει τα σήματα TVALID και TREADY να έχουν την τιμή '1' **στον ίδιο κύκλο του ρολογιου ACLK.** Οποιοδήποτε από τα δύο μπορεί να πάρει πρώτο την τιμή '1' , αρκεί να συμβεί στον ίδιο κύκλο ρολογιού.

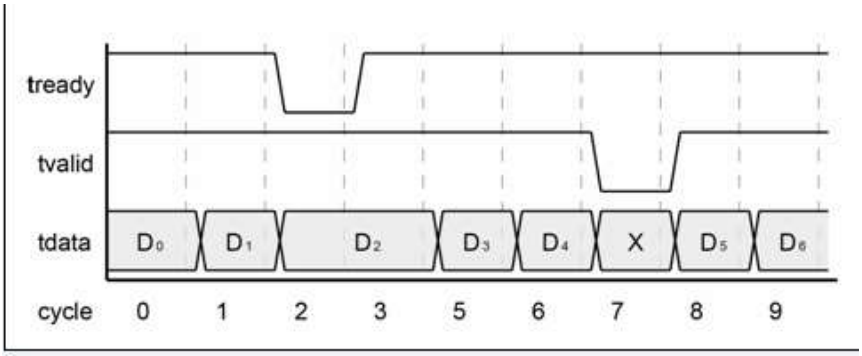
Σημαντικές λεπτομέριες είναι η εξής. Ο Master δεν επιτρέπεται να περιμένει μέχρι το TREADY να γίνει '1' , ώστε να δώσει στο σήμα TVALID την τιμή '1'. Όταν το TVALID πάρει την τιμή '1', πρέπει να τη διατηρήσει μέχρι την επόμενη χειραψία.

'Ενας slave μπορεί να περιμένει να πάρει το σήμα TVALID την τιμή '1', μέχρι να δώσει και αυτός την τιμή '1' στο σήμα TREADY.

Αν ένας slave δώσει στο σ'ημα TREADY την τιμή '1', μπορεί να την αλλάξει πριν το σήμα TVALID πάρει την τιμή '1'.

1.4.4 Παραδείγματα του μηχανισμού χειραψίας του AXI4-Stream

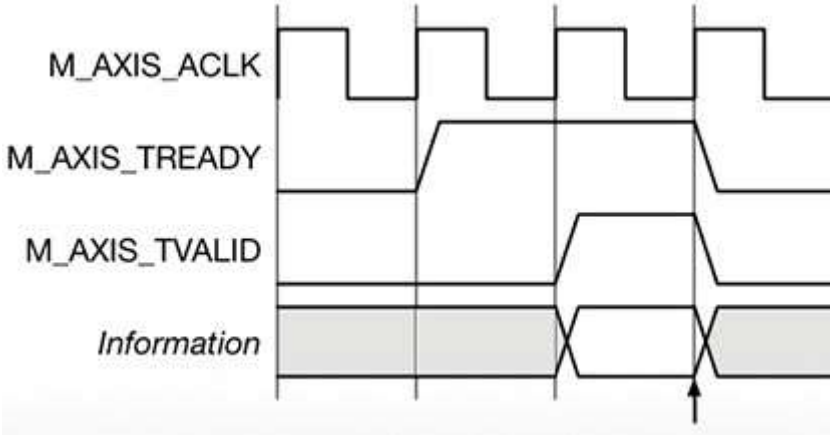
Ακολουθούν παραδείγματα για την καλύτερη κατανόηση του μηχανισμού handshake.



Εικόνα 2:Πρώτο παράδειγμα handshake.

Στην εικόνα 2, οι τιμές του TREADY και TVALID είναι ‘1’ στην αρχή των κύκλων 0 και 1, που σημαίνει ότι μπορεί να πραγματοποιηθεί μεταφορά των δεδομένων D0,D1.

Αντίθετα, στην αρχή του κύκλου 2, το σήμα TREADY παίρνει την τιμή ‘0’ και δεν μπορεί να πραγματοποιηθεί μεταφορά.Γι΄ αυτό η μεταφορά των δεδομένων D2 πραγματοποιείται στον κύκλο 3, αφού στην αρχή αυτού και τα δύο σήματα παίρνουν την τιμή ‘1’.



Εικόνα 3:Δεύτερο παράδειγμα handshake

Στο συγκεκριμένο παράδειγμα, τα δύο σήματα παίρνουν την τιμή ‘1’ στον τρίτο κύκλο. Αυτό σημαίνει πως η μεταφορά δεδομένων θα συμβεί μόνο σε αυτόν τον κύκλο.

1.5 Computer Vision

Computer vision ονομάζεται ο κλάδος της τεχνητής νοημοσύνης (artificial intelligence), που επιτρέπει στους υπολογιστές και γενικά τα συστήματα, να λαμβάνουν σημαντικές πληροφορίες από ψηφιακές εικόνες, βίντεο ή άλλα οπτικά ερεθίσματα. Με βάση αυτές τις πληροφορίες, οι υπολογιστές/συστήματατα μπορούν να λάβουν αποφάσεις ή να κάνουν προτάσεις στο χρήστη/χειριστή για να προβεί στην κατάλληλη ενέργεια .

Το computer vision λειτουργεί με τρόπο παρόμοιο με αυτό της ανθρώπινης όρασης. Η βασική διαφορά είναι ότι οι άνθρωποι έχουν ήδη συλλέξει εμπειρίες στη διάρκεια της ζωής τους και μπορούν να ξεχωρίζουν εύκολα τα αντικείμενα μεταξύ τους, αν κινούνται και πόσο γρήγορα κινούνται και εάν υπάρχει κάποιο λάθος σε μία εικόνα.

Οι υπολογιστές ‘’εκπαιδεύονται’’, ώστε να μπορούν να πραγματοποιήσουν αυτές τις διαδικασίες, σε πολύ λιγότερο χρόνο απ΄ ότι ένας άνθρωπος. Για παράδειγμα, ένα εκπαιδευμένο σύστημα θα πρέπει να είναι σε θέση να ανιχνεύσει, να παρακολουθήσει ή να επιβλέψει χιλιάδες αντικείμενα ή προϊόντα σε πολύ μικρό χρόνο, ώστε να αναφέρει τυχον βλάβες ή ατέλειες.

Γι΄ αυτό το λόγο, επειδή οι δυνατότες ενός υπολογιστή ξεπερνούν κατά πολύ αυτές του ανθρώπινου ματιού, το computer vision είναι διαδεδομένο σε διάφορους τομείς, από τη βιομηχανία αυτοκινήτων, έως και τη βιομηχανία ενέργειας.

1.6 Ανίχνευση Αιχμών (Edge Detection)

Η ανίχνευση αιχμών είναι μία τεχνική, που χρησιμοποιείται στο computer vision. Η τεχνική αυτή έχει ώς σκοπό την ανίχνευση ασυνεχειών σε μία ψηφιακή εικόνα. Ο όρος ασυνέχεια, αναφέρεται σε έντονες αλλαγές της φωτεινότητας σε μία φωτογραφία. Τα σημεία στο οποία η διαφορά αυτή είναι πιο έντονη, ονομάζονται αιχμές (edges).

Η ανίχνευση αυτών των σημείων γίνεται συνήθως με τη διαδικασία της συνέλιξης, δηλαδή με την εφαρμογή κάποιου φίλτρου/μάσκας/ kernel, πάνω στην αρχική εικόνα.

Η ανίχνευση των αιχμών μιας εικόνας, έχει ως σκοπό την αναγνώριση σημαντικών γεγονότων, αντικειμένων και γενικά αλλαγών στον κόσμο, για παράδειγμα τον προσδιορισμό της κατεύθησης ενός κινούμενου αντικειμένου, αλλαγές στις ιδιότητες της επιφάνειας ενός υλικού, αλλαγές στο background μιας φωτογραφίας.

Το ιδανικό αποτέλεσμα της εφαρμογής της αναγνώρισης αιχμών είναι ένα σύνολο από ενωμένες καμπύλες, που υποδυκνείουν τα όρια των αντικειμένων ή επιφανειών. Αυτό συνεπάγεται και ένα πολύ μικρότερο όγκο δεδομένων προς επεξεργασία. Άρα η εφαρμογή αυτής της τεχνικής, μπορεί να φιλτράρει τις άχρηστες λεπτομέρεις μιας φωτογραφίας, ενώ διατηρεί τις μόνο αυτές, που απαιτούνται για την εξαγωγή πληροφοριών και συμπερασμάτων.

Από τα παραπάνω συνεπάγεται ότι η ανίχνευση αιχμών μπορεί να είναι απλή σας διαδικασία, αποτελεί όμως δομικό λίθο σε πληθώρες πολύπλοκες εφαρμογές επεξεργασίας εικόνας.

1.7 Αλγόριθμος Sobel

1.7.1 Περιγραφή Αλγορίθμου

Ο αλγόριθμος Sobel ή αλλιώς Sobel-Feldman είναι ένας απ΄τους αλγόριθμους που χρησιμοποιούνται πιο συχνά στις εφαρμογές του computer vision και της επεξεργασίας εικόνας, για την ανίχνευση αιχμών.

Ο αλγόριθμος εφαρμόζεται κυρίως στις gray-scale εικόνες. Ο αλγόριθμος μπορεί να εφαρμοστεί και σε έχγρωμες είκόνες, όμως υπάρχουν πιο αποδωτικοί αλγόριθμοι για αυτές. Gray-scale εικόνες ονομάζονται αυτές, στις οποίες το χρώμα του κάθε Pixel μπορεί να είναι άσπρο, μαύρο ή και όλες οι ενδιάμεσες αποχρώσεις αυτών των δύο χρωμάτων. Η ένταση της φωτεινότητας του κάθε pixel αναπαρίσταται με μια τιμή μεταξύ ενός εύρους, συνήθως μεταξύ 0 έως 1 ή 0 έως 255. Το 0 συμβολίζει το μαύρο, ενώ το άνω όριο το άσπρο. Η παρούσα πτυχιακή εξετάζει τον αλγόριθμο για εύρος τιμών 0-255, δηλαδή η τιμή της έντασης του κάθε pixel, αναπαρίσταται με 8 bits.

Η βασική ιδέα πίσω απτη λειτουργία του αλγορίθμου είναι η συνέλιξη της gray-scale φωτογραφίας με 2 ισοτροπικούς 3χ3 πίνακες/φίλτρα Gx και Gy. Καθ΄ένας από αυτούς τους δύο πίνακες είναι υπεύθυνος για την εύρεση αιχμών και σε μια διαφορερική συσιστώσα τις εικόνας.

- Ο Gx εφαρμόζεται στον οριζόντιο άξονα (x) της εικόνας και είναι υπεύθυνος για την αναγνώριση των κάθετων αιχμών.
- Ο Gy εφαρμόζεται στον κάθετο άξονα της (y) εικόνας και είναι υπεύθυνος για την αναγνώριση των οριζόντιων αιχμών.

1.7.2 Ανάλυση της Διαδικασίας Εκτέλεσης του Αλγορίθμου

Όπως αναφέρθηκε και πριν, ο αλγόριθμος λειτουργεί χρησιμοποιώντας δύο 3χ3 ισοτροπικούς πίνακες. Αυτοί είναι οι ακόλουθοι:

X – Direction Kernel

-1	0	1
-2	0	2
-1	0	1

Y – Direction Kernel

-1	-2	-1
0	0	0
1	2	1

Figure 1 Πίνακες/φίλτρα αλγορίθμου Sobel

Τα στοιχεία (2,1), (2,3) του Gx και (1,2), (3,2) του Gy μπορούν να έχουν όποια τιμή θεωρεί ο χρήστης αναγκαία, ώστε να διακρίνει καλύτερα τυχών αλλαγές στη ένταση της φωτεινότητας των pixel, δηλαδή την κλίση. Ο συγκεκριμένος αλγόριθμος χρησιμοποιεί την τιμή 2. Επιπλέον το μεσαίο στοιχείο κάθε πίνακα ονομάζεται anchor και εκεί θα αποθηκεύεται κάθε φορά το αποτέλεσμα της συνέλιξης.

Ο αλγόριθμος εφαρμόζει ξεχωριστά τους 2 πίνακες στην εικόνα, ώστε να βρει τις κάθετες και οριζόντιες συνιστώσες και αφού υπολογίσει την απόλυτη τιμή τους, τις προσθέτει για να υπολογίσει το τελικό αποτέλεσμα. Αν το τελικό αποτέλεσμα υπερβεί το κατώφλι του εύρους των τιμών της έντασης των pixel, τότε η τιμή του αποτελέσματος μειλωνεται στο κατώφλι.

Σημαντική λεπτομέρεια είναι ότι για να πραγματοποιηθεί η συνέλιξη, πρέπει το κεντρικό pixel anchor, να διαθέτει ένα πλήρες σύνολο απο γειτονικά pixel. Με απλά λόγια ο αλγόριθμος δεν μπορεί να εφαρμοστεί στα pixel που ανοίκουν στο περίγραμα (frame) της εικόνας. Αυτό το πρόβλημα αντιμετοπίζεται με 2 τρόπους.

- Η επεξεργασμένη εικόνα θα έχει το ίδιο μέγεθος με την αρχική. Στα pixel του περιγράμματος δε θα συμβεί επεξεργασία και θα τεθεί εξαρχής η τιμή 0.

- Η επεξεργασμένη εικόνα θα είναι κατά 2 γραμμές και στήλες μικρότερη απ΄την αρχική, δηλαδή δε θα περιλαμβανει τα pixel του περιγράμματος.

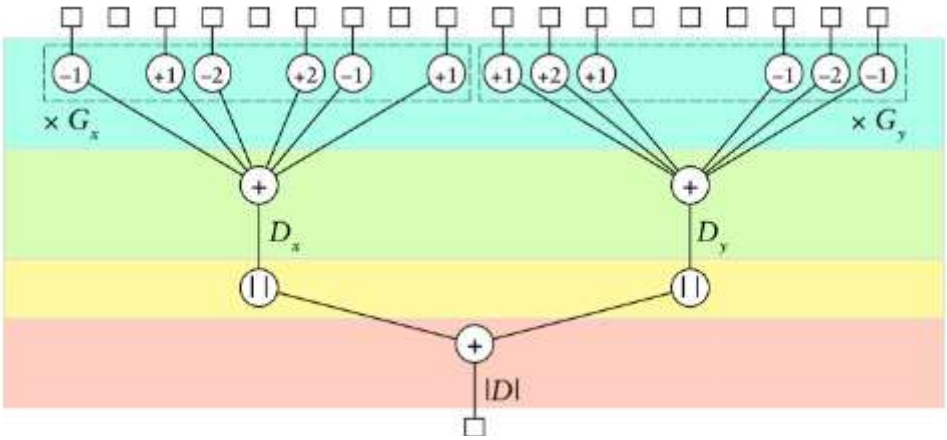


Figure 2 Γράφος εξάρτησης δεδομένων, για τον υπολογισμό ενός pixel της επεξεργασμένης εικόνας

Ακολουθεί παράδειγμα με την εφαρμογή του πίνακα Gy σε μία 6χ6 εικόνα.

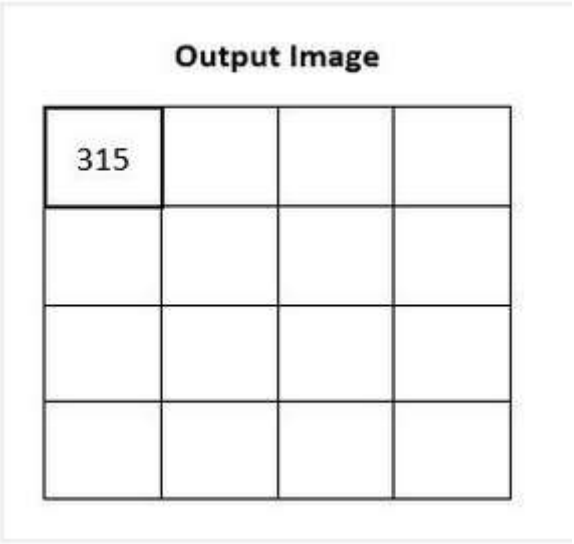
- Έστω η ότι η αρχική εικόνα είναι η ακόλουθη.

150	150	150	255	255	255
150	150	255	255	255	255
150	255	255	255	255	255
255	255	255	255	255	255
255	255	255	255	255	255
255	255	255	255	255	255

- Γίνεται εφαρμογή του Gy στο πάνω αριστερά κομμάτι του πίνακα.

150 * -1	150 * -2	150 * -1	255	255	255
150 * 0	150 * 0	255 * 0	255	255	255
150 * 1	255 * 2	255 * 1	255	255	255
255	255	255	255	255	255
255	255	255	255	255	255
255	255	255	255	255	255

Μετά την ολοκλήρωση των πράξεων, το pixel anchor θα πάρει την τιμή -150 + -300 + -150 + 0 + 0 + 0 + 150 + 510 + 255 = **315,** όπως φαίνεται και στην παρακάτω εικόνα:



Φαίνεται πως η επεξεργασμένη εικόνα έχει λιγότερα pixels από την αρχική. Αυτό σημαίνει ότι χρησιμοποιήθηκε η μέθοδος απομάκρυνσης του περιγράμματος.

Στη συνέχεια, ο Gy εφαρμόζεται στα pixel της επόμενης γραμμής και η διαδικασία επαναλαμβάνεται μέχρι να τελειώσουν τα pixel της γραμμής. Έπειτα ο Gy μετατοπίζεται κατά μία στήλη δεξιά και η διαδικασία επαναλαμβάνεται μέχρι ο Gy να έχει φτάσει στο κάτω δεξιά μέρος της εικόνας.

Η συνέλιξη με τον Gx γίνεται με παρόμοιο τρόπο, με τη διαφορά ότι μετατοπίζεται πρώτα στις στήλες και μετά στις γραμμές. Τέλος προστίθονται τα δύο επιμέρους αποτελέσματα. Έχει τεθεί τιμή κατωφλιού το 255, οπότε όποια τιμή pixel το ξεπεράσει, θα μειωθεί στο κατώφλι.

Η ακόλουθη εικόνα αποτελεί παράδειγμα για την καλύτερη κατανόηση, της επιμέρους εφαρμογής των δύο φίλτρων. Συγκεκριμένα:

- Η πάνω αριστερά εικόνα αποτελεί την αρχική εικόνα που θα τεθεί υπό επεξεργασία.
- Η κάτω αριστερά εικόνα αποτελεί το αποτέλεσμα της εφαρμογής του φίλτρου Gx. Το φίλτρο σαρώνει τον οριζόντιο άξονα και ανιχνεύει τις κάθετες αιχμές.
- Η κάτω δεξιά εικόνα αποτελεί το αποτέλεσμα της εφαρμογής του φίλτρου Gy. Το φίλτρο σαρώνει τον κάθετο άξονα και ανιχνεύει τις οριζόντιες αιχμές.
- Η πάνω δεξιά εικόνα αποτελεί το τελικό αποτέλεσμα, δηλαδή την επεξεργασμένη εικόνα, τη συνισταμένη των δύο παραπάνω συνιστωσών.

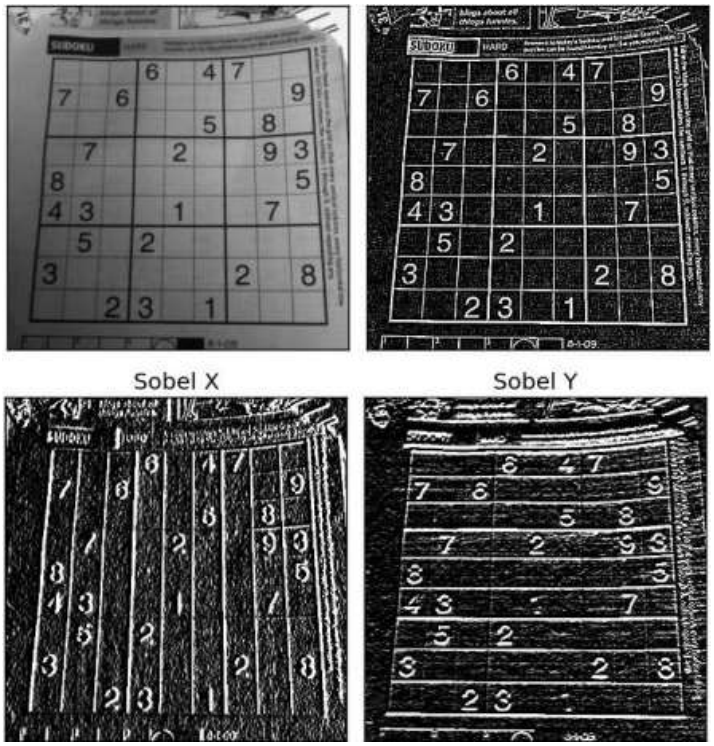


Figure 3 Παράδειγμα για κατανόηση των συνιστοσών του αλγορίθμου

Κεφάλαιο 2: Υλοποίηση Αλγορίθμου

Το λογισμικό matlab αποτελεί ένα πολύ χρήσιμο εργαλείο για τις εφαρμογές της επεξεργασίας εικόνας, μιας και διαθέτει έτοιμες εντολές, κάποιες από τις οποίες θα χρησιμοποιηθούν για την προεπεξεργασία, τόσο του bit-accurate model, όσο και των δεδομένων εισόδου για το fpga.

2.1 Προ-Επεξεργασία εικόνων στο matlab

Η διαδικασία προ-επεξεργασίας δεδομένων για το bit-accurate model είναι η ακόλουθη :

- Εύρεση εικόνων με διαφορετική ανάλυση (αριθμό συνολικών pixel)

- Μετατροπή των εικόνων σε grayscale, μέσω της έτοιμης συνάρτησης rbg2gray() του matlab. Η συνάρτηση αυτή δέχεται ως είσοδο μία έγχρωμη εικόνα, την οποία επεξεργάζεται ως τανιστή και δίνει ως έξοδο μια gray-scale εικόνα, την οποία αποθηκεύει ως πίνακα (matrix) δεκαδικών στοιχείων. Τα στοιχεία του πίνακα έχουν εύρος τιμών που είναι ορισμένο από το matlab. Πρέπει ο χρήστης να ορίσει νέο εύρος τιμών. Στη συγκεκριμένη περίπτωση ορίζεται ως εύρος το 0-255. Οι πίνακες αποθηκεύονται σε αρχεία .txt. Κάθε txt θα φτορτωθεί σε άλλο script για επεξεργασία.

Για τη χρήση των παραπάνω δεδομένων στο fpga θα ακολουθηθεί μια σχεδόν παρόμοια διαδικασία.

- Εύρεση εικόνων και περικοπή, ώστε όλες να έχουν σταθερό μέγεθος 100x100.

- Μετατροπή των παραπάνω εικόνων σε gray-scale μέσω του προηγούμενου script.

- Μετατροπή του πίνακα της grayscale εικόνας, από το δεκαδικό σύστημα στο δυαδικό.Το εύρος τιμών είναι 0-255, που σημαίνει ότι το κάθε pixel της εικόνας θα αναπαρίσταται από ένα vector των 8 bits.

- Κατασκευή των παραθύρων εισόδου. Με τον όρο παράθυρο, δηλώνεται το σύνολο των pixel που θα εισαχθεί κάθε φορά στον κομμάτι του αλγορίθμου, που πραγματοποιεί τη διαδικασία της συνέλιξης με τα φίλτρα, για τον υπολογισμό ενός από τα pixel της επεξεργασμένης εικόνας. Εδώ το κάθε φίλτρο είναι 3x3 και η ένταση του κάθε pixel απεικονίζεται με 8 bits, άρα το μέγεθος του κάθε παραθύρου θα είναι 72 bits ή 9 bytes. Κάθε byte θα θεωρηθεί ως string, οπότε θα συμβεί σύνδεση 9 string. Επιλέγεται η χρήση παραθύρων και όχι pixel, γιατί θα κάνουν πιο εύκολη την υλοποίηση του αλγορίθμου στο FPGA. Τα 9604 παραθυρα που προκύπτουν, εξάγονται σε αρχείο .coe, που είναι συμβατό για χρήση σε FPGA.

2.2 Κατασκευή Bit-Accurate Model στο matlab

Όπως αναφέρθηκε και στην εισαγωγή, το λογισμικό matlab θα χρησιμοποιηθεί με σκοπό την κατασκευή του bit-accurate model ή bit-true implementation. Το μοντέλο αυτό αφενός θα εφαρμόζει τον αλγόριθμο και θα παράγει επεξεργασμένες εικόνες, αφετέρου θα χρησιμοποιηθεί για τον έλεγχο της ορθότητας των αποτελεσμάτων από την επεξεργασία της εικόνας στο fpga.

Η υλοποίση του προγράμματος είναι αρκετά απλή. Το πρόγραμμα θα δέχεται ως είσοδο ένα αρχείο .txt, θα δημιουργεί τον αντίστοιχο n x m πίνακα τιμών των pixel και με εφαρμογή των φίλτων Gx και Gy θα δίνει ως έξοδο ένα πίνακα n x m, τον οποίο θα εμφανίζει και με τη μορφή εικόνας.

Το βασικό κομμάτι του προγράμματος, που θα πραγματοποιεί την συνέλιξη, βασίζεται πάνω στο γράφο εξάρτησης που υπάρχει στη θεορία. Αποτελείται από μία εμφολευμένη for-loop, η οποία θα δέχεται κάθε φορά ως είσοδο 9 στοιχεία-Pixel, θα υπολογίζει τα δύο μερικά αποτελέσματα των συνελίξεων με τους πίνακες Gx και Gy . Στο τέλος θα προσθέτει την απόλυτη τιμή τους, ώστε να υπολογίσει το ολικό αποτέλεσμα. Αν κάποιο από αυτά έχει τιμή μεγαλύτερη του 255, τίθεται στο 255. Σημαντική λεπτομέρεια είναι ότι ο πίνακα εξόδου έχει το ίδιο μέγεθος με τον πίνακα εισόδου. Αυτό σημαίνει ότι ο αλγόριθμος θετει την τιμή των Pixel του περιγράμματος ‘0’.

2.3 Αποτελέσματα Εφαρμογής της Υλοποίησης σε matlab

Ακολουθούν παραδείγματα της εφαρμογής του αλγορίθμου στο matlab

Figure 4:Παράδειγμα 1

Figure 5:Παράδειγμα 2

Figure 6: Παράδειγμα 3

Figure 7: Παράδειγμα 4

Figure 8: Παράδειγμα 5

2.4 Θετικά και Αρνητικά Αλγορίθμου Sobels

2.4.1 Θετικά Αλγορίθμου Sobel

Τα θετικά του αλγορίθμου Sobel συνοψίζονται στην ευκολία της εφαρμογής της μεθόδου και την προσαρμοστικότητά του. Συγκεκριμένα:

- Προσαρμοστικότητα : Όπως αναφέρθηκε και στη θεορία, τα στοιχεία (2,1), (2,3) του Gx και (1,2), (3,2) του Gy, μπορούν να έχουν όποια τιμή θεωρεί ο χρήστης αναγκαία. Πολλές φορές οι αιχμές δε θα είναι ευδιάκριτες στο αλγόριθμο, οπότε με αύξηση της τιμής αυτών των στοιχείων, ο αλγόριθμος θα μπορεί να τις ανιχνεύσει με μεγαλύτερη ευκολία.

- Ευκολία εφαρογής : Ο αλγόριθμος είναι απλός στην εφαρμογή και αρκετά αποδωτικός από άποψη χρόνου, αφου αποτελείται κυρίως από απλές μαθηματικές πράξεις.

2.4.2 Αρνητικά Αλγορίθμου Sobel

- Λόγος σήματος/θορύβου : Ο αλγόριθμος μπορεί να ανιχνεύσει αιχμές , οι οποίες δεν είναι πραγματικά αντικείμενα και η τελική εικόνα να περιλαμβάνει πολύ θόρυβο. Στο παράδειγμα 4 της φιγούρας 7, ο αλγόριθμος ανίχνευσε “νιφάδες” στο background.

- Αδυναμία ανίχνευσης αιχμών με μικρή κλίση : Για να ανιχνεύσει μία αιχμή, ο αλγόριθμος βασίζεται στο ότι θα υπάρχουν σημεία με μεγάλη διαφορά στην τιμή της έντασης της φωτεινώτητας των pixel. Όταν όμως το αντικείμενο έχει το ίδιο χρώμα με το background, είναι πολύ εύκολο για τον αλγόριθμο να χάσει σημαντικές λεπτομέρειες, όπως στο παράδειγμα 5 της φιγούρας 8.

2.5 Υλοποίηση αλγορίθμου στο FPGA

Ο σκοπός της παρακάτω ενότητας είναι η δημιουργία ip-core, το οποίο θα δέχεται ένα αρχείο .coe με τα δεδομένα των παραθύρων μιας εικόνας και θα εξάγει ένα txt με τις τιμές των pixel της επεξεργασμένης εικόνας. Ο ip-core θα αποτελείται από 4 components :

- Ένα wrapper(πρόγραμμα ελέγχου) για μνήμη Rom, η οποία θα περιλαμβάνει το αρχείο .coe με τα παράθυρα.
- Component που θα πραγματοποιεί τις συνελίξεις.
- Ένα FIFO .
- Μία μνήμη Rom, που θα αποτελείται από ένα αρχείο .txt με τις τιμές ένασης των pixel της επεξεργασμένης εικόνας. Θα υλοποιηθεί στη διαδικασία του simulation.

Ακολουθεί λεπτομερής περιγραφή καθενός από τα components.

2.5.1 Περιγραφή του ROM-Wrapper

Top entity, η οποία είναι υπεύθυνη για τη λειτουργία μιας rom ip-core του vivado. Αποτελείται από τη rom ip-core που περιέχει το αρχείο .coe, στο οποίο είναι αποθηκευμένα τα δεδομένα με τα παράθυρα και ένα finite state machine (fsm), που θα λειτουργεί ως λογική ελέγχου. Ο Wrapper θα διαθέτει ένα AXI4-Stream master interface.

Ο ip-core της μνήμης rom θα δέχεται σε κάθε κύκλο μία διεύθυνση και στον επόμενο κύκλο θα επιστρέφει το περιεχόμενο της. Δέχεται τα σήματα addra, clk και έχει ως έξοδο το σήμα data_out. Συγκεκριμένα:

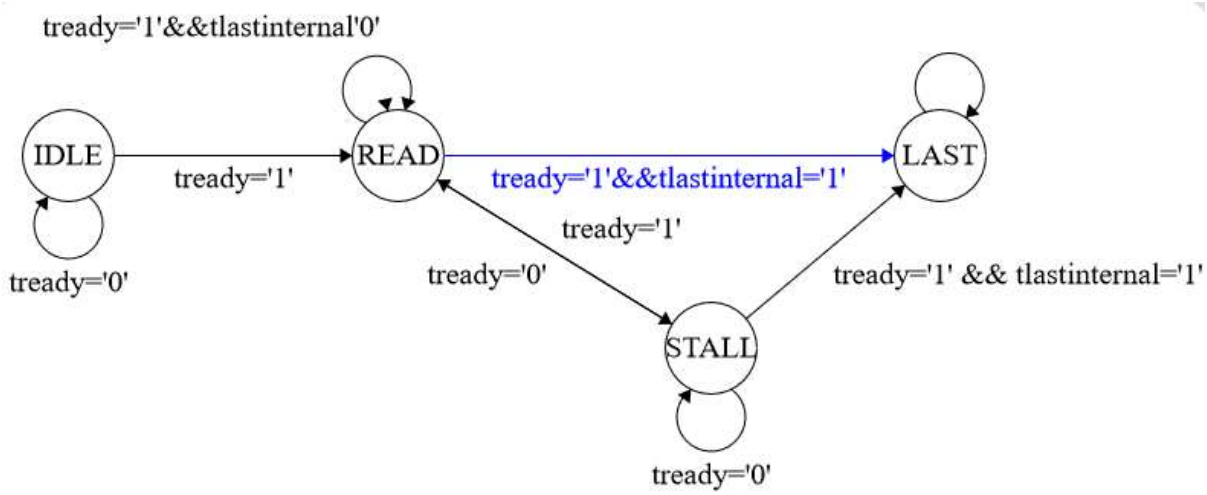
- addra θα παίρνει την τιμή του από το σήμα addr_in, που θα του δίνει ο f.s.m. και θα είναι η τιμή της διεύθυνσης, το περιεχόμενο της οποίας θα πρέπει να επιστρέψει στον επόμενο κύκλο.

- clk είναι συνδεδεμένο με το καθολικό σήμα ρολογιού του συστήματος.
- Data_out, θα είναι συνδεδεμένο στην έξοδο m_axis_data του wrapper και θα αποτελεί την ολική έξοδο του top entity. Αποτελεί τα δεδομένα ενός παραθύρου.

Ο f.s.m. θα ελέγχει το master interface και θα ρυθμίζει όλα τα εωτερικά σήματα της διάταξης. Αυτά θα είναι :

- addr_in, η διεύθυνση μνήμης, της οποίας το περιεχόμενο πρέπει να επιστρέψει η Rom στον επόμενο κύκλο.
- data_counter, σήμα το οποίο μετράει πόσα παράθυρα έχουν προοθηθεί από τη rom. Χρησιμοποιείται σε ελέγχους για τη μετάβαση στην κατάσταση LAST.
- m_axis_tvalid_internal, σήμα το οποίο είναι αντίγραφο του m_axis_tvalid και χρησιμοποιείται για εσωτερικούς ελέγχους.
- m_axis_tlast_internal, σήμα το οποίο είναι αντίγραφο του m_axis_tlast και χρησιμοποιείται για εσωτερικούς ελέγχους.

Ο f.s.m. θα έχει την ακόλουθη μορφή



Εικόνα 4: Μορφή fsm του rom wrapper

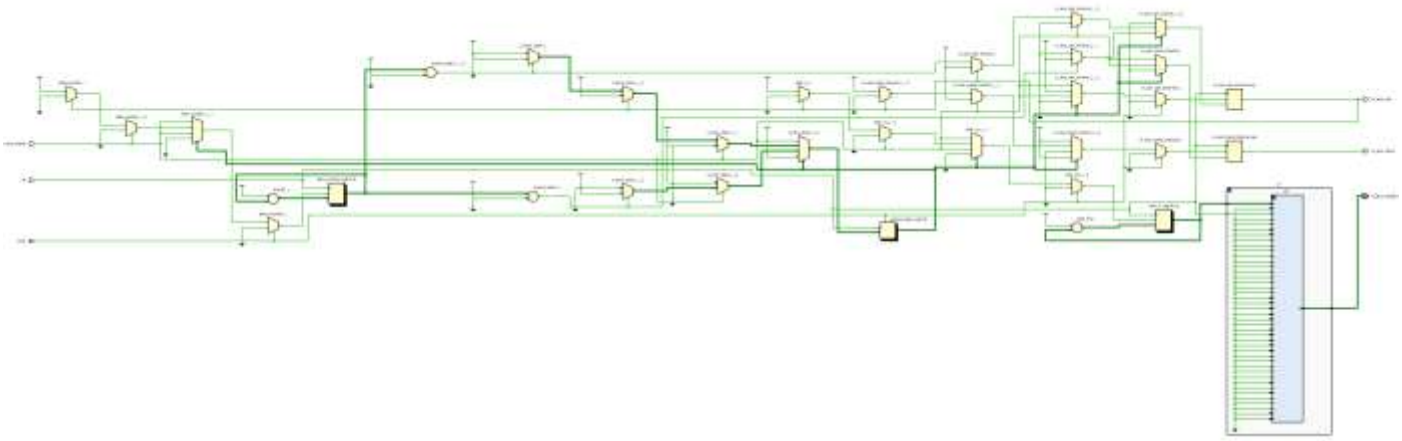
Αποτελείται από 4 καταστάσεις : IDLE, READ, STALL και LAST. Η αλλαγή από μία κατάστασταση σε μία άλλη, γίνεται με βάση τις τιμές των σημάτων tready και tlast_internal.

Συγκεκριμένα:

- IDLE : Η αρχική κατάσταση του f.s.m. Βρίσκεται σε αυτή έως ότου το σήμα tready να πάρει την τιμή '1'. Τότε θα μεταβεί στην κατάσταση READ. Όσο ο f.s.m. βρίσκεται στην κατάταση IDLE, δε γίνεται ανάγνωση δεδομένων από τη μνήμη Rom.
- READ: Κατάσταση στην οπόια μεταβαίνει ο f.s.m. όταν το σήμα tready πάρει την τιμή '1'. Σε αυτήν την κατάσταση συμβαίνει ανάγνωση δεδομένων από τη μνήμη Rom.

Το εσωτερικό σήμα data_counter καταγράφει το πλήθος των τιμών που έχουν διαβαστεί. Σε περίπτωση που το σήμα tready πάρει την τιμή ‘0’, ο f.s.m. μεταβαίνει στην κατάσταση STALL. Σε περίπτωση που το σήμα data_counter πάρει την τιμή 9603, που σημαίνει ότι έχουν διαβαστεί όλες οι τιμές της Rom, ο f.s.m θα μεταβεί στην κατάσταση LAST.

- STALL: Κατάσταση στην οποία μεταβαίνει ο f.s.m οταν βρίσκεται στην κατάσταση READ και το σημα tready παίρνει την τιμή ‘0’. Στην κατάσταση αυτή δε συμβαίνει ανάγνωση δεδομένων από τη μνήμη. Ο f.s.m. περιμένει μέχρι το σήμα tready να πάρει πάλι την τιμή ‘1’. Όταν το σήμα tready πάρει πάλι την τιμή ‘1’ γίνεται και ένας δεύτερος έλεγχος για την τιμή tlast_internal. Αν έχει την τιμή ‘0’ μεταβαίνει στην κατάσταση READ, αλλιώς μεταβαίνει στην κατάσταση LAST.
- LAST: Κατάσταση στην οποία μεταβαίνει ο f.s.m. όταν έχει διαβαστεί όλο το περιεχόμενο την Rom. Στην κατάσταση αυτή το σήμα tlast παίρνει την τιμή ‘1’, για να σηματοδωτήσει τη λήξη της ανάγνωσης, ενώ το σήμα tvalid παίρνει την τιμή ‘0’, μιας και δεν υπάρχουν άλλα έγκυρα δεδομένα προς ανάγνωση.



Εικόνα 5:Σχηματική αναπαράσταση σε επίπεδο RTL του Rom wrapper

2.5.2 Περιγραφή της Μονάδας Συνέλιξης

Μονάδα, η οποία είναι υπεύθυνη για τη λήψη δεδομένων από το rom_wrapper και την πραγματοποίηση της διαδικασίας συνέλιξης. Διαθέτει 2 AXI4-Stream interfaces, ένα slave που δέχεται τα δεδομένα από το rom wrapper και ένα master, ο οποίος στέλνει τα επεξεργασμένα δεδομένα στο FIFO. Η μονάδα συνέλιξης δέχεται ως είσοδο τα δεδομένα ενός παραθύρου, τα χωρίζει σε 9 pixel και πραγματοποιεί τη διαδικασία της συνέλιξης. Τα σήματα που δέχεται είναι τα εξής:

- Reset: Σήμα που είναι υπεύθηνο για την ενεργοποίηση της μονάδας.
- Clk : Σήμα που αποτελεί το ρολόι του συστήματος

Το slave interface αποτελείται από τα εξής σήματα :

- s_axis_tdata : Έξοδος της rom_wrapper. Αποτελεί την είσοδο της μονάδας συνέλιξης και άρα τα δεδομένα ενός παραθύρου
- s_axis_tvalid: Σήμα που δηλώνει εάν τα δεδομένα που λαμβάνει η μονάδα είναι έγκυρα.
- s_axis_tready: Σήμα που δηλώνει εάν ο slave(μονάδα συνέλιξης) είναι έτοιμος να δεχθεί δεδομένα.
- S_axis_tlast: Σήμα που δηλώνει εάν ο rom_wrapper έχει στείλει και το τελευταίο παράθυρο.

Το master interface αποτελείται από τα εξής σήματα :

- m_axis_tdata : Έξοδος της μονάδας συνέλιξης. Αποτελεί το pixel της επεξεργασμένης εικόνας. Θα συνδεθεί με την είσοδο του FIFO.
- m_axis_tvalid: Σήμα που δηλώνει εάν τα δεδομένα που στέλνει η μονάδα συνέλιξης είναι έγκυρα.
- m_axis_tready: Σήμα που δηλώνει εάν ο slave(FIFO) είναι έτοιμος να δεχθεί δεδομένα.
- m_axis_tlast: Σήμα που δηλώνει εάν η μονάδα συνέλιξης έχει στείλει τα δεδομένα για το τελευταίο επεξεργασμένο pixel.

Η λειτουργία της μονάδας είναι απλή και βασίζεται σε μια σειρά απο ελέγχους “if”. Συγκεκριμένα:

- Η πρώτη If ελέγχει αν η μονάδα είναι στην κατάσταση reset. Αν είναι, τότε η έξοδος όλων των σημάτων είναι ‘0’.
- Όταν η μονάδα δεν είναι σε κατάσταση reset, ελέγχεται εάν τα εξωτερικά σήματα s_axis_tvalid και m_axis_treay είναι ‘1’. Αν ναι, τότε η μονάδα ξεκινάει την επεξεργασία δεδομένων. Οι δύο αυτοί έλεγχοι σημαίνουν ότι η μονάδα λαμβάνει έγκυρα δεδομένα από τη rom και το FIFO είναι έτοιμο να δεκτεί δεδομένα. Η διαδικασία επεξεργασίας των δεδομένων είναι αρκετά απλή. Το παράθυρο χωρίζεται σε 9 pixel, και κάθε pixel πολλαπλασιάζεται με τον αντίστοιχο συντελεστή, σύμφωνα με το γράφο εξάρτησης, για τον υπολογισμό των 2 μερικών αθροισμάτων. Στο τέλος τα 2 μερικά αθροίσματα προστίθενται μεταξύ τους ώστε να προκύψει το ολικό άθροισμα. Αν αυτό είναι μεγαλύτερο του 255(τιμή κατωφλιού), μειώνεται στο 255.
  Για την αποθήκευση των παραπάνω τιμών, επιλέκτηκε η χρήση μεταβλητών έναντι εσωτερικών σημάτων και δε χρησιμοποιήθηκε η μέθοδος του Pipeline. Οι λόγοι γιαυτό ήταν οι εξής:
  1) Η βελτίωση στο χρόνο που προέκυπτε από τη χρήση εσωτερικών σημάτων και pipeline ήταν μικρή.
  2) Με τη χρήση μεταβλητών, οι υπολογισμοί πραγματοποιούνται στον ίδιο κύκλο. Αυτό σημαίνει ότι η μονάδα θα έχει μικρότερο latency και ο ορισμός του σήματος m_axis_tvalid θα είναι πιο εύκολος.

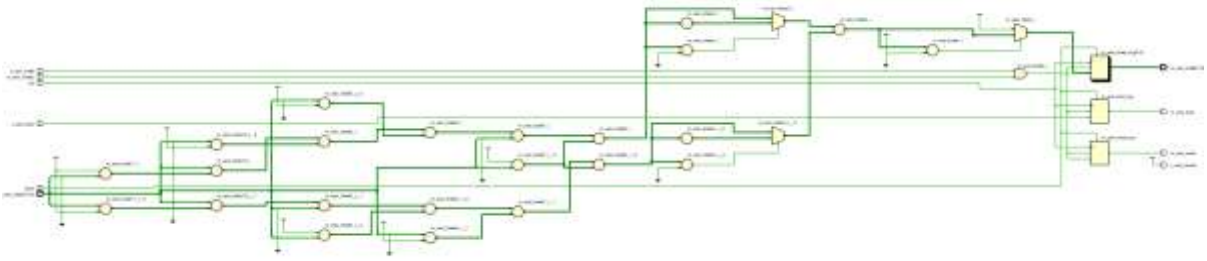


Figure 9:Σχηματική αναπαράσταση της μονάδας συνέλιξης

2.5.3 Περιγραφή της Μονάδας FIFO

Η μονάδα FIFO αποτελεί ένα από τα ip cores του περιβάλλοντος vivado και μπορεί να χρησιμοποιηθεί, με περιορισμούς, στην αρχιτεκτονική της εργασίας. Αυτοί οι περιορισμοί είναι ότι ο χρήστης δεν έχει πρόσβαση στο πρόγραμμά του, καθώς και στη σχηματική αναπαράστασή του.

Η μονάδα FIFO διαθέτει ένα slave και ένα master AXI4-Stream interface και τα ακόλουθα σήματα:

- Areset_n: Σήμα που είναι υπεύθηνο για την ενεργοποίηση της μονάδας.
- Clk : Σήμα που αποτελεί το ρολόι του συστήματος.

Το slave interface αποτελείται από τα εξής σήματα :

- s_axis_tdata : Έξοδος της μονάδας συνέλιξης. Αποτελεί την είσοδο του FIFO και άρα τα δεδομένα ενός pixel της επεξεργασμένης εικόνας.
- s_axis_tvalid: Σήμα που δηλώνει εάν τα δεδομένα που λαμβάνει η μονάδα είναι έγκυρα.
- s_axis_tready: Σήμα που δηλώνει εάν ο slave (FIFO) είναι έτοιμος να δεχθεί δεδομένα.
- S_axis_tlast: Σήμα που δηλώνει εάν ο η μονάδα συνέλιξης έχει στείλει και το τελευταίο παράθυρο.

Το master interface αποτελείται από τα εξής σήματα :

- m_axis_tdata : Έξοδος της μονάδας FIFO. Αποτελεί το pixel της επεξεργασμένης εικόνας. Θα συνδεθεί με την έξοδο της ολικής αρχιτεκτονικής.
- m_axis_tvalid: Σήμα που δηλώνει εάν τα δεδομένα που στέλνει η μονάδα FIFO είναι έγκυρα.
- m_axis_tready: Σήμα που δηλώνει εάν ο slave (rom) είναι έτοιμος να δεχθεί δεδομένα.

- m_axis_tlast: Σήμα που δηλώνει εάν η μονάδα FIFO έχει στείλει τα δεδομένα για το τελευταίο επεξεργασμένο pixel.

Η μονάδα FIFO χρειάζεται 3 επιπλέον κύκλους από τη στιγμή που θα απενεργοποιηθεί το σήμα areset_n, ώστε να είναι σε θέση να λάβει δεδομένα. Γι' αυτό το λόγο το σήμα areset_n συνδέεται με ένα δεύτερο σήμα reset_FIFO της ολικής αρχιτεκτονικής, το οποίο θα ενεργοποιεί το FIFO τουλάχιστον 3 κύκλους πριν τις υπόλοιπες μονάδες. Επιπλέον, η μονάδα FIFO έχει 2 κύκλους latency.

Ο λόγος που χρησιμοποιείται η μονάδα FIFO είναι να συλλέγει τα δεδομένα από τη μονάδα συνέλιξης και να τα στέλνει στη μνήμη rom με τέτοιο τρόπο, που μειώνει το latency και αυξάνει το throughput.

2.5.4 Περιγραφή της μνήμης Rom

Για την κατασκευή της μνήμης αυτής, θα χρησιμοποιειθεί η βιβλιοθήκη textio και ένα αρχείο .txt, κατά τη διαδικασία του Testbench. Συγκεκριμένα, η βιβλιοθήκη textio επιτρέπει στο χρήστη να αποθηκεύει τις τιμές τις προσομοίωσης σε ένα αρχείο .txt.

Στο testbench, μια process θα αποθηκεύει σε κάθε κύκλο,την τιμή που θα παίρνει από το FIFO, εφόσον το σήμα tvalid έχει την τιμή '1', δηλαδή για το διάστημα που τα δεδομένα θα είναι έγκυρα.

2.5.5 Περιγραφή του top entity

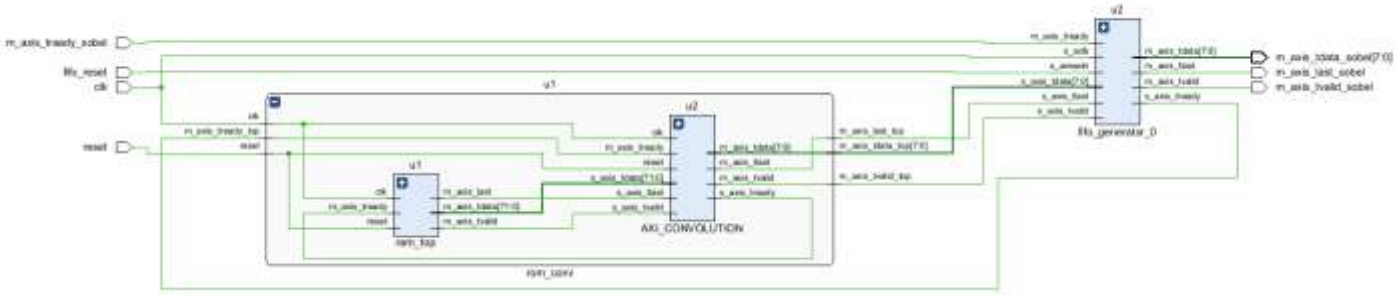
Το top entity αποτελεί την ολική αρχιτεκτονική της υλοποίησης. Αποτελείται από 3 μονάδες, το rom_wrapper, το AXI_Convolution(μονάδα συνέλιξης) και το FIFO. Διαθέτει ένα AXI4-Stream master interface. Τα σήματα που διαθέτει είναι τα εξής:

- fifo_reset: Σήμα που είναι υπεύθυνο για την ενεργοποίηση της μονάδας FIFO.
- reset: Σήμα που είναι υπεύθυνο για την ενεργοποίηση των υπόλοιπων μονάδων.
- clk : Σήμα που αποτελεί το ρολόι του συστήματος.

Το master interface αποτελείται από τα εξής σήματα :

- m_axis_tdata_sobel : Έξοδος της ολικής αρχιτεκτονικής. Αποτελεί το pixel της επεξεργασμένης εικόνας.

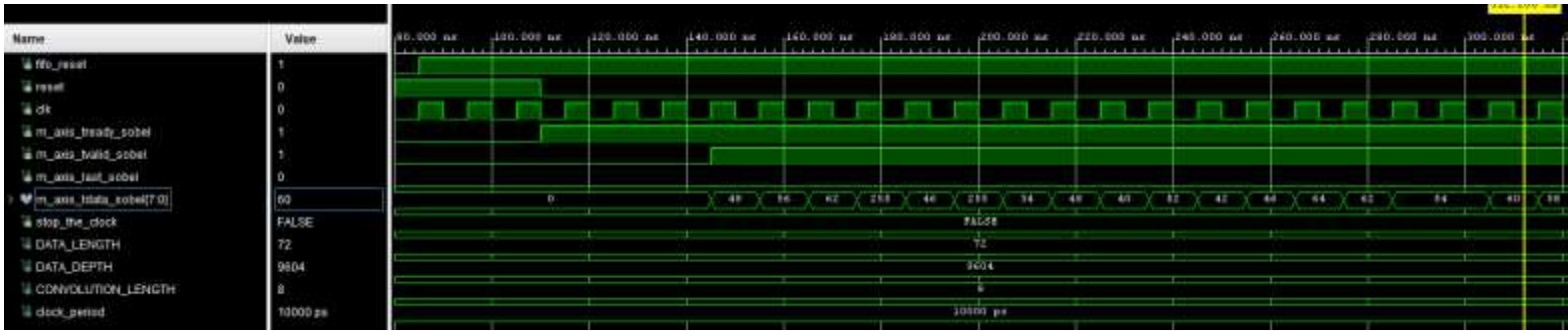
- m_axis_tvalid _sobel: Σήμα που δηλώνει εάν τα δεδομένα που στέλνει η αρχιτεκτονική είναι έγκυρα.
- m_axis_tready _sobel: Σήμα που δηλώνει εάν ο slave (rom) είναι έτοιμος να δεχθεί δεδομένα.
- m_axis_tlast _sobel: Σήμα που δηλώνει εάν η αρχιτεκτονική έχει στείλει τα δεδομένα για το τελευταίο επεξεργασμένο pixel.

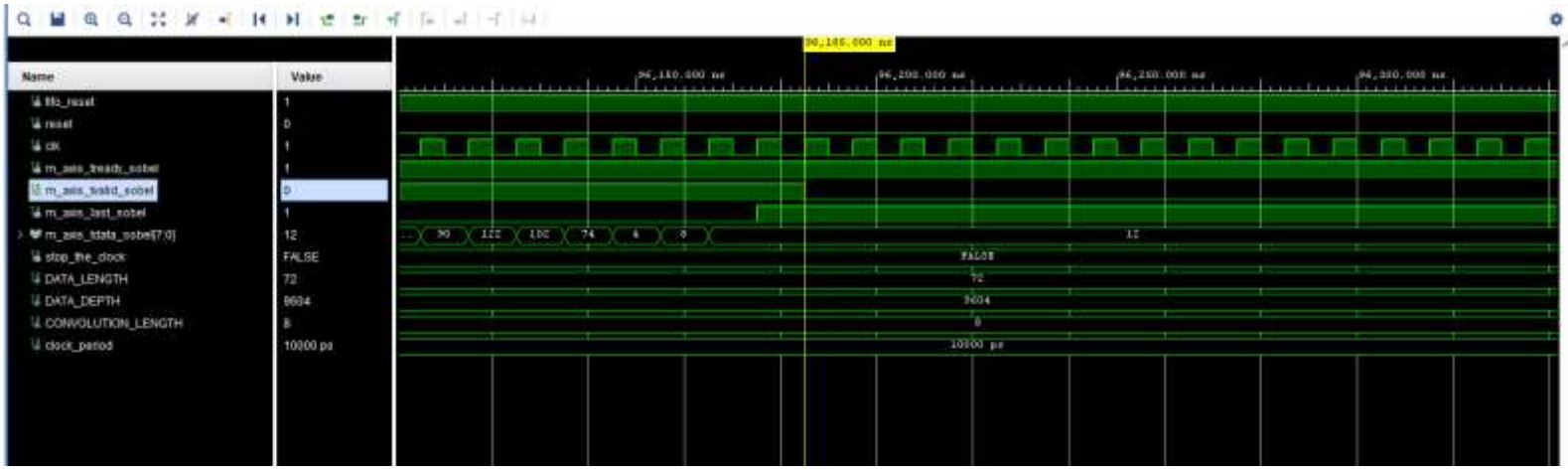


Εικόνα 6:Σχηματική αναπαράσταση του top entity

2.5.6 Ανάλυση Αποτελεσμάτων Simulation του top entity

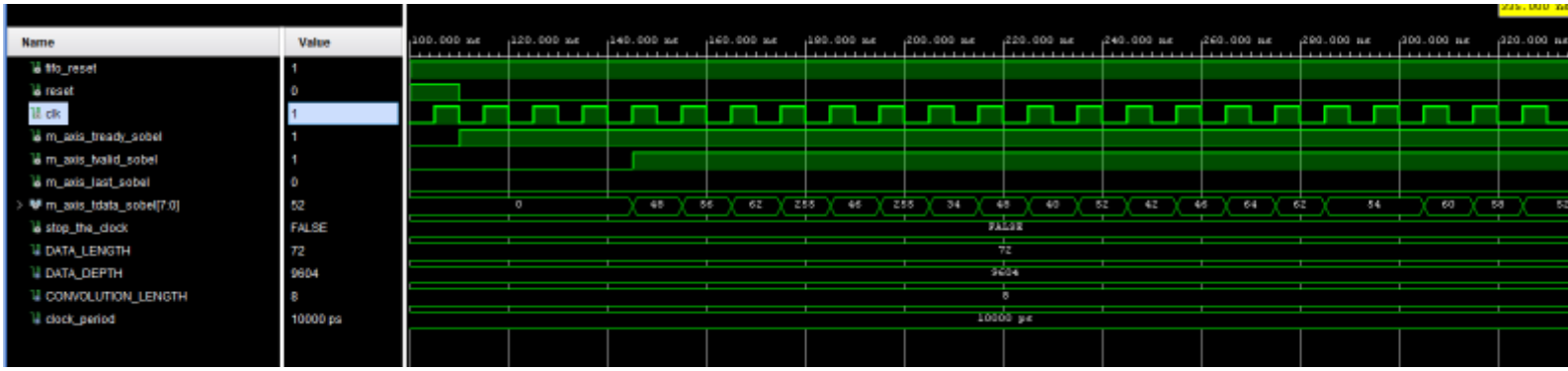
Ακολουθούν εικόνες από behavioural, και post implementation functional simulation αντιστοιχα.





Εικόνα 7:Κυματομορφές behavioural simulation

Το σημείο αυτό αποτελεί και τη λήξη του behavioral simulation. Όταν ξεκινάει η μεταφορά των δεδομένων του τελαυταίο Pixel, το σήμα t_last παίρνει την τιμή ‘1’. Όταν τελειώνει αντίστοιχα η μεταφορά των δεδομένων του, το σήμα t_valid παίρνει την τιμή ‘0’, αφού δεν υπάρχουν άλλα έγκυρα δεδομένα.



Εικόνα 8:Post- mplementation functinal simulation

2.5.7 Επιβεβαίωση των Αποτελεσμάτων του Top Entity

Τα αποτελέσματα της επεξεργασίας θα αποθηκευτούν σε ένα αρχείο .txt, που θα λειτουργήσει ώς μνήμη rom και θα φορτωθεί στο matlab. Τα στοιχεία θα αποθηκευτούν στη συνέχεια σε ένα πίνακα.

Στο matlab, μέσω ενός script, θα συφκριθούν τα αποτελέσματα του FPGA με αυτά του bit-accurate model. Συγκεκριμένα, μια εμφωλευμένη for-loop θα ελέγχει ένα-ένα τα στοιχεία των δύο πινάκων. Μια μεταβλητή κατάστασης (flag) θα ελέγχει αν οι τιμές αυτές διαφέρουν μεταξύ τους. Αν διαφέρουν, θα παίρνει την τιμή ‘1’ και θα εκτυπώνει μήνυμα σφάλματος.

2.5.8 Ανάλυση Αποτελεσμάτων Υλοποίησης του top entity

Για τη λειτουργία του top entity, απαιτείται η χρήση ενός αρχείου, που θα περιλαμβάνει τους χρονικούς περιορισμούς (timing constraints).

Στη συγκεκριμένη περίπτωση, έχει δωθεί χρονικός περιορισμός 12 nanosecond.

Ο πραγματική περίοδος λειτουργίας είναι ο χρόνος περιορισμού μείον το χρόνο worst negative slack, που θα προκύψει κατά το στάδιο του implementation. Ο χρόνος worst negative slack είναι ο χρόνος της μεγαλύτερης διαδρομής μεταξύ δύο καταχωρητών. Αν είναι θετικός, η μονάδα λειτουργεί σε μικρότερη συχνότητα από τη δοσμένη. Αν είναι αρνητικός, η περίοδος στα constraints πρέπει να αυξηθεί.

Design Timing Summary

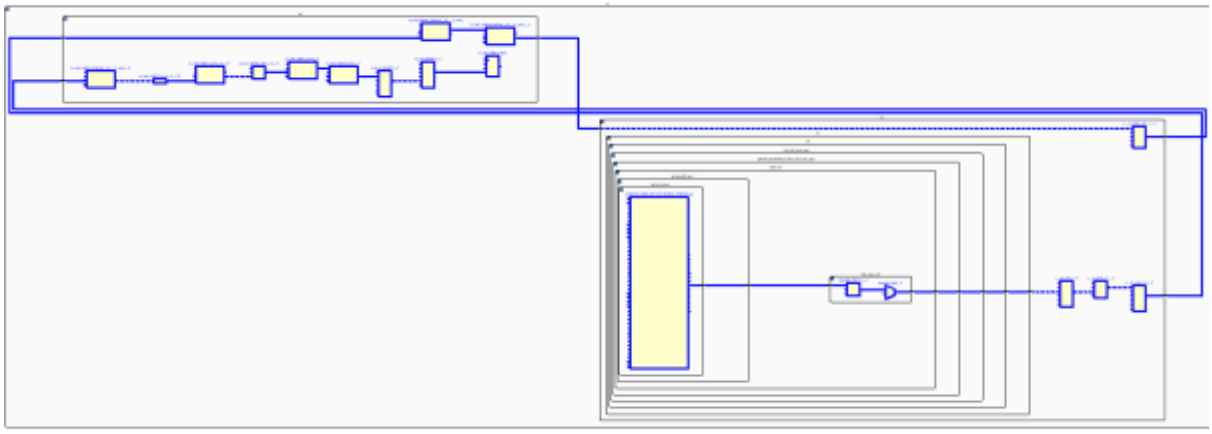
Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	4,640 ns	Worst Hold Slack (WHS):	0,055 ns	Worst Pulse Width Slack (WPWS):	5,458 ns
Total Negative Slack (TNS):	0,000 ns	Total Hold Slack (THS):	0,000 ns	Total Pulse Width Negative Slack (TPWS):	0,000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	461	Total Number of Endpoints:	461	Total Number of Endpoints:	148

All user specified timing constraints are met.

Εικόνα 9:Πίνακας με το χρόνο wns

Άρα η συγκεκριμένη μονάδα έχει περίοδο 12 – 4.64 = 7.36 nanosecond και συχνότητα λειτουργίας 135,87 MHz.

Η χειρότερη κρίσημη διαδρομή φαίνεται στην παρακάτω εικόνα. Πρόκειται για το path μεταξύ της rom και της μονάδας συνέλιξης.



Εικόνα 10:Worst critical path

Τέλος, ακολουθεί ο πίνακας στον οποίο αναγράφονται οι πόροι που χρησιμοποιήθηκαν.

Resource	Utilization	Available	Utilization %
LUT	368	230400	0.16
FF	107	460800	0.02
BRAM	19.50	312	6.25
IO	14	360	3.89
BUFG	1	544	0.18

Εικόνα 11: Utilization Board

Φαίνεται πως η μονάδα χρησιμοποιεί κυρίως LUT. Αυτό συμβαίνει διότι η μονάδα συνέλιξης χρησιμοποιεί μεταβλητές αντί για σήματα. Οι μεταβλητές υλοποιούνται μέσω LUT ενώ τα σήματα χρησιμοποιούν κυρίως flip flops. Ταυτόχρονα, οι υπολογισμοί της μονάδας συνέλιξης για κάθε pixel συνέβαιναν στον ίδιο κύκλο, που σημαίνει ότι το LUT της διαδικασίας ήταν αρκετά μεγάλο.

Ευχαριστίες

Το πρώτο άτομο που θα ήθελα να ευχαριστήσω είναι ο καθηγητής μου, Διονύσης Ρεΐσης, που μου έδωσε την ευκαιρία να συνεργαστούμε για άλλη μια φορά, αλλά και για την εμπιστοσύνη που μου έδειξε αυτά τα δύο χρόνια. Τον Αλέξη, που μου παρείχε συμβουλές πάνω στην εργασία όποτε τις χρειαζόμουν. Δεν θα μπορούσα όμως να ξεχάσω να αναφέρω και τους πολύ καλούς μου φίλους, ειδικά τον Κώστα και το Γιάννη, που ήταν δίπλα μου σε κάθε δύσκολη στιγμή και με στήριζαν, τόσο εντός, όσο και εκτός της σχολής.

Βιβλιογραφία

[1] Kanopoulos, N., Vasanthavada, N., & Baker, R. T. (1988). Design of an image edge detection filter using the Sobel operator

[2] Wikipedia. https://en.wikipedia.org

[3] Αντώνης Πασχάλης, Γιώργος Παπαδημητρίου.Digital system design laboratory guide

[4] Ιωάννης Α. Καλόμοιρος. Intro to VHDL

[5] XST User Guide, Xilinx

[6] AXI4-Stream FIFO v4.1, Xilinx

[7] AXI Reference Guide (AXI),Xilinx

Παράρτημα

Matlab script Δημιουργεία Παραθύρων

```
XLENGTH=100;
YLENGTH=100;
arrayIndex=1;

I= imread('crop.png');
I = rgb2gray(I); %input image in dec


sz1=size(I);
BinI=reshape(string(dec2bin(I)),sz1); % input image in binary



for x=3:XLENGTH      % window generator.concatenates 9 pixels into one array
element.Matlax works collumn-wise meaning
                      %each window is (1,4,7,2,5,8,3,6,9) instead
of(1,2,3,4,5,6,7,8,9)
     for y=3:YLENGTH

       windowArray(arrayIndex)=strcat(BinI(x-2,y-2),BinI(x-1,y-2),BinI(x,y-
2),BinI(x-2,y-1),BinI(x-1,y-1),BinI(x,y-1),BinI(x-2,y),BinI(x-1,y),BinI(x,y));
       arrayIndex=arrayIndex+1;
     end
end

fid = fopen('windows.coe','wt');

 myData = windowArray;
 if fid > 0
     fprintf(fid,'%s,\n',myData');
     fclose(fid);
 end
```

MATLAB Bit-Accurate Model

```
M=readmatrix('road.txt');

XLENGTH=100;
YLENGTH=100;
Sobel_in=reshape(M,XLENGTH,[]);
G_x=zeros(XLENGTH,YLENGTH);
G_y=zeros(XLENGTH,YLENGTH);
sobel_out=zeros(XLENGTH,YLENGTH);


for x=1:XLENGTH
     for y=1:YLENGTH

         if x == 1 || x == XLENGTH || y == 1 || y == YLENGTH
            G_x(x,y)=0;
            G_y(x,y)=0;

         else
             G_x(x,y) =Sobel_in(x+1,y - 1) + 2 * Sobel_in(x+1 ,y) + Sobel_in(x+1
,y+1 ) - Sobel_in(x - 1,y - 1) - 2 * Sobel_in(x - 1,y) - Sobel_in(x - 1,y+1 );
             G_y(x,y) =Sobel_in(x-1,y+1 ) + 2 * Sobel_in(x ,y+1)  + Sobel_in(x+1
,y+1 ) - Sobel_in(x - 1,y - 1) - 2 * Sobel_in(x,y-1) - Sobel_in(x +1,y -1);

         end
     end

end
sobel_out=abs(G_x)+abs(G_y);
for x=1:XLENGTH
     for y=1:YLENGTH
        if sobel_out(x,y)>255
             sobel_out(x,y)=255;
        end
     end
end
```

VHDL AXI-ROM WRAPPER

```
entity ram_top is
generic ( DATA_LENGTH : positive := 72;
        DATA_DEPTH : positive := 9604);

   port (
         reset : in std_logic;
         clk : in std_logic;

         m_axis_tready : in std_logic;
         m_axis_tvalid : out std_logic;
         m_axis_last : out std_logic;
         m_axis_tdata : out std_logic_vector(DATA_LENGTH-1 downto 0));
end ram_top;

architecture Behavioral of ram_top is

component  blk_mem_gen_0
    PORT (
   clka : IN STD_LOGIC;
   addra : IN STD_LOGIC_VECTOR(13 DOWNTO 0);
   douta : OUT STD_LOGIC_VECTOR(71 DOWNTO 0)
 );
 end component;

--axi4 master fsm declaration
type FSM_states is  (IDLE,READ,STALL,LAST);
signal current_state : FSM_states;
```

```
--internal signal declaration
signal addr_in : std_logic_vector(13 downto 0) := (others => '0');
signal data_counter : unsigned(DATA_LENGTH-1 downto 0) :=(others => '0');



signal m_axis_tvalid_internal :std_logic :='0';
signal m_axis_last_internal :  std_logic :='0';

begin

u1:blk_mem_gen_0 port map(clka=>clk,
                addra=>addr_in,
              douta=>m_axis_tdata);

m_axis_tvalid<=m_axis_tvalid_internal;
m_axis_last<=m_axis_last_internal;




 master_fsm: process(clk,reset)
  begin
    if reset = '1' then

      m_axis_last_internal<=m_axis_last_internal;
      m_axis_tvalid_internal <= m_axis_tvalid_internal;
      data_counter  <=  data_counter;
      addr_in<=addr_in;
      current_state   <=  IDLE;
```

```
elsif rising_edge(clk) then

  case(current_state) is
      when IDLE =>

       if(m_axis_tready='1')then
         m_axis_tvalid_internal <= '1';
         addr_in <= std_logic_vector(unsigned(addr_in) + to_unsigned(1,13));
         data_counter <= data_counter + 1;
         current_state <= READ;
         m_axis_last_internal<='0';
       else
          m_axis_tvalid_internal <= '0';
          addr_in <= addr_in;
          data_counter <= data_counter ;
          m_axis_last_internal<=m_axis_last_internal;
          current_state <= IDLE;
       end if;

      when READ  =>
       if(m_axis_tready='1') then
         if(m_axis_last_internal='0')then
              m_axis_tvalid_internal <= '1';
              addr_in <= std_logic_vector(unsigned(addr_in) + to_unsigned(1,13));
              data_counter    <= data_counter + 1;
              if (data_counter = DATA_DEPTH-1 ) then
                   m_axis_last_internal<='1';
                   current_state  <= LAST;
              else
                   m_axis_last_internal<='0';
                   current_state  <= READ;
```

```
            end if;
        else
            m_axis_tvalid_internal<=m_axis_tvalid_internal;
            m_axis_last_internal<='1';
            current_state  <= LAST;
            addr_in<=addr_in;
            data_counter<=data_counter;
        end if;
    else
    current_state  <= STALL;
    addr_in<=addr_in;
    data_counter<=data_counter;
    m_axis_tvalid_internal <= '0';
    m_axis_last_internal<='0';
    end if;

when STALL =>

    if(  m_axis_tready='0') then
        current_state  <= STALL;
        data_counter<=data_counter;
        m_axis_tvalid_internal <= m_axis_tvalid_internal;
        m_axis_last_internal<=m_axis_last_internal;
        addr_in <= addr_in;

    else

        m_axis_tvalid_internal <= m_axis_tvalid_internal;
        addr_in <= std_logic_vector(unsigned(addr_in) + to_unsigned(1,13));
        data_counter    <= data_counter + 1;
        if (data_counter < DATA_DEPTH-1  ) then
```

```
                    m_axis_last_internal<='0';
                    current_state  <= READ;


                else

                    m_axis_last_internal<='1';
                    current_state  <= LAST;


                end if;
               end if;
            when LAST =>

              m_axis_tvalid_internal<='0';
              addr_in<=addr_in;
              m_axis_last_internal<=m_axis_last_internal;
              current_state<=LAST;

            when others =>
              current_state <= IDLE;

              m_axis_tvalid_internal <= '0';
              m_axis_last_internal<='0';
        end case;
      end if;
 end process;
end Behavioral;
```

VHDL CONVOLUTION BLOCK

```
entity AXI_CONVOLUTION is
   generic ( windowlentgh : positive := 72;
             convolutionlength : positive :=8);

   port (  s_axis_tdata: in std_logic_vector(windowlentgh-1 downto 0);
           s_axis_tvalid : in std_logic;
           s_axis_tlast :in std_logic;
           s_axis_tready : out std_logic;

           m_axis_tdata: out std_logic_vector(convolutionlength-1 downto 0);
           m_axis_tvalid : out std_logic;
           m_axis_tlast :out std_logic;
           m_axis_tready : in std_logic;

           reset : in std_logic;
           clk : in std_logic );

end AXI_CONVOLUTION;

architecture Behavioral of AXI_CONVOLUTION is

type matrix is array (0 to 2, 0 to 2) of signed(8 downto 0);

constant coefX : matrix := ((to_signed( 1,9), to_signed( 0,9), to_signed( -1,9)),
                            (to_signed( 2,9), to_signed(0,9), to_signed( -2,9)),
                            (to_signed( 1,9), to_signed( 0,9), to_signed( -1,9)));


constant coefY : matrix := ((to_signed( 1,9), to_signed( 2,9), to_signed( 1,9)),
```

```
                (to_signed( 0,9), to_signed(0,9), to_signed( 0,9)),
                (to_signed( -1,9), to_signed( -2,9), to_signed( -1,9)));



begin

  s_axis_tready<='1';

   conv:process (CLK,reset)
     variable gx_sum, gy_sum : signed(17 downto 0) ;
     variable sum_3 : unsigned(17 downto 0);
     variable  pixel_1,pixel_2,pixel_3,pixel_4,pixel_5,pixel_6,pixel_7,pixel_8,pixel_9: signed(8 downto
0);
     begin

      if(reset='1') then

          m_axis_tdata <= (others => '0');
          m_axis_tvalid <='0';
          m_axis_tlast  <='0';


      elsif rising_edge(clk) then
          if (  s_axis_tvalid='1' and m_axis_tready='1') then

             pixel_1:=signed('0' & s_axis_tdata(71 downto 64));
             pixel_4:=signed('0' & s_axis_tdata(63 downto 56));
             pixel_7:=signed('0' & s_axis_tdata(55 downto 48));
             pixel_2:=signed('0' & s_axis_tdata(47 downto 40));
             pixel_5:=signed('0' & s_axis_tdata(39 downto 32));
```

```
            pixel_8:=signed('0' & s_axis_tdata(31 downto 24));
            pixel_3:=signed('0' & s_axis_tdata(23 downto 16));
            pixel_6:=signed('0' & s_axis_tdata(15 downto 8));
            pixel_9:=signed('0' & s_axis_tdata(7 downto 0));

            gx_sum := pixel_1*coefX(0,0)+  pixel_2*coefX(0,1)+ pixel_3*coefX(0,2)+
                  pixel_4*coefX(1,0) + pixel_5*coefX(1,1)+ pixel_6*coefX(1,2)+
                  pixel_7*coefX(2,0) + pixel_8*coefX(2,1)+ pixel_9*coefX(2,2);

            gy_sum := pixel_1*coefY(0,0)+  pixel_2*coefY(0,1)+ pixel_3*coefY(0,2)+
                  pixel_4*coefY(1,0) + pixel_5*coefY(1,1)+ pixel_6*coefY(1,2)+
                  pixel_7*coefY(2,0) + pixel_8*coefY(2,1)+ pixel_9*coefY(2,2);

            sum_3 := unsigned(abs(gx_sum)+abs( gy_sum));

            if sum_3>255 then
              m_axis_tdata<="11111111";
            else
              m_axis_tdata<=std_logic_vector(resize(sum_3,8));
            end if;
            m_axis_tlast<=s_axis_tlast;
            m_axis_tvalid<='1';

         else
            m_axis_tlast<=s_axis_tlast;
            m_axis_tvalid<='0';
         end if;
      end if;
   end process;
end Behavioral;
```

VHDL TOP ENTITY

```
entity sobel_top is
        generic ( DATA_LENGTH : positive := 72;
                DATA_DEPTH : positive := 9604;
                CONVOLUTION_LENGTH : positive := 8);
        Port ( fifo_reset : in std_logic;
            reset : in std_logic;
            clk : in std_logic;
            m_axis_tready_sobel : in std_logic;
            m_axis_tvalid_sobel: out std_logic;
            m_axis_last_sobel : out std_logic;
            m_axis_tdata_sobel : out std_logic_vector(CONVOLUTION_LENGTH-1 downto 0));



end sobel_top;

architecture Behavioral of sobel_top is
component rom_conv
generic ( DATA_LENGTH : positive := 72;
        DATA_DEPTH : positive := 9604;
       CONVOLUTION_LENGTH : positive := 8);

   Port ( reset : in std_logic;
          clk : in std_logic;
          m_axis_tready_top : in std_logic;
          m_axis_tvalid_top : out std_logic;
          m_axis_last_top : out std_logic;
          m_axis_tdata_top : out std_logic_vector(CONVOLUTION_LENGTH-1 downto 0));
end component;
```

```
component fifo_generator_0

   PORT (
        s_aclk : IN STD_LOGIC;
        s_aresetn : IN STD_LOGIC;

        s_axis_tvalid : IN STD_LOGIC;
        s_axis_tready : OUT STD_LOGIC;
        s_axis_tdata : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
        s_axis_tlast : IN STD_LOGIC;

        m_axis_tvalid : OUT STD_LOGIC;
        m_axis_tready : IN STD_LOGIC;
        m_axis_tdata : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
        m_axis_tlast : OUT STD_LOGIC );
end component;

signal m_axis_tvalid_top_internal :std_logic :='0';
signal m_axis_last_top_internal :  std_logic :='0';
signal m_axis_ready_top_internal :  std_logic :='0';

signal m_axis_tdata_top_internal :  std_logic_vector(CONVOLUTION_LENGTH -1 downto 0):=
(others=>'0');
signal m_axis_tdata_top_internal2 :  std_logic_vector(127 downto 0):= (others=>'0');

signal m_axis_tdata_sobel_internal :  std_logic_vector(CONVOLUTION_LENGTH -1 downto 0):=
(others=>'0');
signal reset_internal :  std_logic :='0';
```

```
begin
u1:rom_conv   generic  map ( DATA_LENGTH => 72,
                    DATA_DEPTH =>9604,
                    CONVOLUTION_LENGTH=>8)
          port map(clk=>clk,
                reset=>reset,
                m_axis_tready_top=>m_axis_ready_top_internal,
                m_axis_tvalid_top=>m_axis_tvalid_top_internal,
                m_axis_last_top=>m_axis_last_top_internal,
                m_axis_tdata_top=>m_axis_tdata_top_internal );

u2:fifo_generator_0
          port map(s_aclk=>clk,
                s_aresetn=>fifo_reset,

                s_axis_tvalid=>m_axis_tvalid_top_internal,
                s_axis_tready=>m_axis_ready_top_internal,
                s_axis_tlast=>m_axis_last_top_internal,
                s_axis_tdata=>m_axis_tdata_top_internal,

                m_axis_tvalid=>m_axis_tvalid_sobel,
                m_axis_tready=>m_axis_tready_sobel,
                m_axis_tlast=>m_axis_last_sobel,
                m_axis_tdata=> m_axis_tdata_sobel);



end Behavioral;
```

VHDL TOP ENTITY TESTBENCH

```
library IEEE;
use IEEE.Std_logic_1164.all;
use IEEE.Numeric_Std.all;
use IEEE.std_logic_textio.ALL;
library STD;
use std.textio.all;

entity sobel_top_tb is
end;

architecture bench of sobel_top_tb is

 component sobel_top
          -- generic ( DATA_LENGTH : positive := 72;
              --     DATA_DEPTH : positive := 9604;
               --    CONVOLUTION_LENGTH : positive := 8);
          Port ( fifo_reset : in std_logic;
              reset : in std_logic;
              clk : in std_logic;
              m_axis_tready_sobel : in std_logic;
              m_axis_tvalid_sobel: out std_logic;
              m_axis_last_sobel : out std_logic;
              m_axis_tdata_sobel : out std_logic_vector(8-1 downto 0));
 end component;
 constant DATA_LENGTH : positive := 72;
 constant DATA_DEPTH : positive := 9604;
 constant CONVOLUTION_LENGTH : positive := 8;

 signal fifo_reset: std_logic;
```

```
 signal reset: std_logic;
 signal clk: std_logic;
 signal m_axis_tready_sobel: std_logic;
 signal m_axis_tvalid_sobel: std_logic;
 signal m_axis_last_sobel: std_logic;
 signal m_axis_tdata_sobel: std_logic_vector(8-1 downto 0);

 constant clock_period: time := 10 ns;
 signal stop_the_clock: boolean;

begin

 -- Insert values for generic parameters !!
 uut: sobel_top
                port map ( fifo_reset         =>fifo_reset,
                        reset             => reset,
                        clk               => clk,
                        m_axis_tready_sobel => m_axis_tready_sobel,
                        m_axis_tvalid_sobel => m_axis_tvalid_sobel,
                        m_axis_last_sobel   => m_axis_last_sobel,
                        m_axis_tdata_sobel  => m_axis_tdata_sobel );

 stimulus: process
 begin
  fifo_reset<='0';
  reset <= '1';
   m_axis_tready_sobel<='0';

   wait for 8*clock_period + 5ns;
   fifo_reset<='1';
   wait for 2*clock_period + 5ns;
```

```
  reset <= '0';
  m_axis_tready_sobel<='1';

  wait ;


  stop_the_clock <= true;
  wait;

  wait;
end process;

clocking: process
begin
  while not stop_the_clock loop
    CLK <= '0', '1' after clock_period / 2;
    wait for clock_period;
  end loop;
  wait;
end process;




write_process: process(clk)
        file Fout : text open write_mode is "output.txt";
        variable row: line;
        begin
           if rising_edge(clk) then
             if(m_axis_tvalid_sobel='1' and m_axis_last_sobel='0') then
                 write(row, m_axis_tdata_sobel);
                 writeline(fout, row);
```

```
            end if;
          end if;
      end process;



end;
```

MATLAB SCRIPT Για Επιβεβαίωση Τιμών

```
fid = fopen('output.txt', 'rt');
datacell = textscan(fid, '%s');
fclose(fid);
binary_cell = datacell{1};
binary_array = zeros(size(binary_cell));
for i = 1:numel(binary_cell)
    binary_array(i) = bin2dec(binary_cell{i});
end

sobelTrim= sobel_out(2:end-1, 2:end-1);



Binary_matrix=transpose(reshape(binary_array, 98, []));

for x=1:XLENGTH-2
     for y=1:YLENGTH-2

      if  Binary_matrix(x,y)~=sobelTrim(x,y)
        disp("error");
      end
     end
end
```