ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΦΟΡΟΦΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Parallel Architecture Design and Trade-off Analysis for Hyperspectral Image Processing on FPGA

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Σίμων Μ. Βέλλα

Επιβλέπων: Δημήτριος Ι. Σούντρης
Αναπληρωτής Καθηγητής

Αθήνα, Ιούλιος 2016

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ
ΚΑΙ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Parallel Architecture Design and Trade-off Analysis for Hyperspectral Image Processing on FPGA

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Σίμων Μ. Βέλλα

Επιβλέπων: Δημήτριος Ι. Σούντρης
Αναπληρωτής Καθηγητής

Εγκρίθηκε από την τριμελή επιτροπή την 21η Ιουλίου 2016.

…...........................…..	…...........................…..	…...........................…..
Δημήτριος Ι. Σούντρης	Κιαμάλ Ζ. Πεσκμεστζή	Κωνσταντίνος Καράντζαλος
Αναπληρωτής Καθηγητής	Καθηγητής	Λέκτορας

Αθήνα, Ιούλιος 2016

...
ΣΙΜΩΝ Μ. ΒΕΛΛΑΣ
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών ΕΜΠ

Περίληψη

Τα τελευταία χρόνια, η υπερφασματική απεικόνιση συναντάται σε ένα σύνολο διαφορετικών εφαρμογών, από συστήματα καταγραφής γεωλογικών δεδομένων μέχρι συστήματα υψηλής ταχύτητας πραγματικού χρόνου όπως ο ποιοτικός έλεγχος τροφίμων στην παραγωγή. Οι αυξημένες δυνατότητες που μας προσφέρει αυτή η τεχνολογία, συνδυάζονται με αντίστοιχα αυξημένες απαιτήσεις για υπολογιστική ισχύ οι οποίες δεν μπορούν να ικανοποιηθούν εύκολα από κοινές επεξεργαστικές μονάδες. Ως εκ τούτου, επιταχυντές υλικού όπως το FPGA λαμβάνουν μια όλο και πιο κεντρική θέση σε συστήματα επεξεργασίας υπερφασματικών δεδομένων. Το ενδιαφέρον γύρω από αυτές τις συσκευές υπάρχει λόγω των αυξημένων δυνατοτήτων παράλληλης επεξεργασίας δεδομένων καθώς και της ικανότητας επαναδιαμόρφωσης των μονάδων. Ο προγραμματισμός των FPGA γίνεται αποδοτικά με τη χρήση γλωσσών περιγραφής υλικού, όπως η VHDL και η Verilog.

Στην παρούσα εργασία αναπτύξαμε σε παραμετρική VHDL τον πυρήνα ενός συστήματος υπερφασμαστικής απεικόνισης και υλοποιήσαμε πολλαπλές διατάξεις στο Xilinx Zynq-7000 FPGA. Από λειτουργική άποψη, το σύστημα που αναπτύξαμε, αντιστοιχίζει εισερχόμενα Pixel από μια υπερφασματική κάμερα με ένα σύνολο, ήδη γνωστών, φασματικών υπογραφών. Διερευνήσαμε παράλληλες αρχιτεκτονικές σε τρία επίπεδα αφαίρεσης ενώ σχεδιάσαμε το σύστημα για εύκολη εναλλαγή των κεντρικών υπολογιστικών μονάδων και τη προσαρμογή του στην εκάστοτε μετρική/αλγόριθμο. Αξιολογήσαμε την επιτάχυνση του συστήματός μας σε σύγκριση με τις αντίστοιχες software υλοποιήσεις σε επεξεργαστές, π.χ. Intel i3 ή έναν ενσωματωμένο ARM. Επιπλέον, σε MATLAB, πραγματοποιήσαμε αξιολόγηση των επιδόσεων του συστήματος ταιριάσματος φασματικών υπογραφών ως προς την ακρίβειά του. Δουλέψαμε με υπερφασματικές εικόνες με έως 285 κανάλια (από διάφορες κάμερες), τρεις αλγορίθμους/μετρικές σύγκρισης και εξετάσαμε την απόδοση του συστήματος μέσω τριών μετρικών ποιότητας. Με βάση τη μελέτη μας, προβήκαμε σε μια tradeoff ανάλυση μεταξύ της υλοποίησης σε FPGA, της ταχύτητας και της ακρίβειας. Εν τέλει, επιτύχαμε εντυπωσιακά αποτελέσματα ακόμα και για την πιο ρεαλιστική από τις διατάξεις μας: Η επιτάχυνση υλικού κυμαίνεται από 70x-321x (έναντι του i3-3110M) και 626x-9694x (έναντι του ARM Cortex A9) ενώ η πληρότητα/ορθότητα/ποιότητα του συστήματος ανίχνευσης κυμαίνεται από 70% έως 98% για διάφορα αντικείμενα.

Λέξεις-Κλειδιά: Υπερφασματική Ανάλυση, Σύγκριση Φασματικών Υπογραφών, Επιταχυντές Υλικού, FPGA, Xilinx Zynq SoC, VHDL, Παράλληλες Αρχιτεκτονικές,

This page is intentionally left blank.

Abstract

In recent years, hyperspectral imaging has found its way in a series of applications ranging from environmental monitoring to high speed sensing and food processing with real time performance. Enhancing the detection abilities of this technology relies on increasing the processing capabilities of the computational units supporting the hyperspectral imaging sensors. In this direction, FPGA-based hardware accelerators have gained the interest of the scientific community due to the very fast processing capabilities as well as the reconfiguration ability offered by this platform. Until today, the efficient programming of these devices is usually done with hardware description languages like VHDL or Verilog.

In the current thesis, we developed in parametric VHDL the kernel of a hyperspectral imaging system and we implemented multiple configurations on Zynq-7000 SoC FPGA. From a functional point of view, the developed system matches incoming pixels from a hyperspectral camera with a set of spectral signatures known a priori. We combined various parallel architectures on three abstraction levels and we followed a modular design approach allowing for easy adaptation to algorithm/matching. We evaluated the acceleration of our system and compared it against software implementations on CPUs, e.g. Intel i3 and embedded ARM. Additionally, on MATLAB, we conducted a performance evaluation of the signature matching system with respect to its accuracy; we considered hyperspectral images of up to 285 spectral channels (from various cameras), three matching algorithms/metrics, and we examined the performance by using three quality measures. Based on our study, we provided a trade-off analysis among hardware implementation, system speed and accuracy. In the end, impressive results were achieved even for the most realistic of the considered scenarios: the hardware acceleration ranges from 70x–321x (vs. i3-3110M) to 626x–9694x (vs. ARM Cortex A9), whereas the completeness/correctness/quality of the detection system ranges from 70% to 98% for various objects.

Keywords: Hyperspectral Imaging, Spectral matching, FPGA, Hardware Accelerators, Xilinx Zynq SoC, VHDL, Parallel Architectures,

This page is intentionally left blank.

Ευχαριστίες

Η παρούσα διπλωματική εκπονήθηκε στο εργαστήριο Μικροϋπολογιστών και Ψηφιακών Συστημάτων της σχολής ΗΜΜΥ του ΕΜΠ. Θα ήθελα να ευχαριστήσω ειλικρινά τον καθηγητή μου κύριο Δημήτριο Σούντρη για την εμπιστοσύνη του στις δυνάμεις μου και τη συνεχή στήριξη κατά τη διάρκεια της παρουσίας μου στο εργαστήριο.

Επίσης, ευχαριστώ τον καθηγητή Κωνσταντίνο Καράντζαλο της σχολής τοπογράφων μηχανικών του ΕΜΠ για το μεγάλο ενδιαφέρον που έδειξε εξαρχής για τη συγκεκριμένη συνεργασία μας. Οι εποικοδομητικές και σίγουρα δημιουργικές συναντήσεις μας ενθάρρυναν σημαντικά την προσπάθειά μου.

Η ολοκλήρωση της εργασίας αυτής δε θα ήταν δυνατή χωρίς τη πολύ στενή και συνεχή συνεργασία με τον Γιώργο Λεντάρη, μεταδιδακτορικό ερευνητή και τον Κωνσταντίνο Μαραγκό, διδακτορικό φοιτητή στο εργαστήριο μικροϋπολογιστών. Δε θα μπορούσα να τους ευχαριστήσω αρκετά μέσα από αυτό το σημείωμα όμως χαίρομαι ειλικρινά που δουλέψαμε μαζί όλο αυτό το χρόνο στο εργαστήριο. Για το μεγάλο ενδιαφέρον που έδειξαν εξαρχής και την αμέριστη συμβολή τους στην παρούσα εργασία σε πολλαπλά επίπεδα τους είμαι ειλικρινά ευγνώμων.

Ιδιαίτερα εποικοδομητική ήταν η συνεργασία μου και με τον Ζαχαρία Κανδυλάκη, διδακτορικό φοιτητή στο εργαστήριο τηλεπισκόπησης της σχολής τοπογράφων μηχανικών του ΕΜΠ. Δουλέψαμε πολλές φορές μαζί και μου παρείχε ιδιαίτερα σημαντική βοήθεια σε διάφορα θέματα στα οποία είχα περιορισμένη γνώση. Για τη συμβολή του αυτή και τη μόνιμα ευχάριστη και υποστηρικτική του διάθεση τον ευχαριστώ θερμά.

Για το ευχάριστο και δημιουργικό κλίμα στο εργαστήριο μικροϋπολογιστών πρέπει να ευχαριστήσω όλα τα παιδιά του εργαστήριου, προπτυχιακούς και διδακτορικούς φοιτητές και ερευνητές με τους οποίους κατά καιρούς συνεργαστήκαμε, συζητήσαμε και περάσαμε πολλές στιγμές μαζί, ιδιαίτερα σημαντικές για εμένα.

Τέλος, κρατάω ένα ξεχωριστό και το μεγαλύτερο ευχαριστώ για την οικογένειά μου, για την εντυπωσιακά μεγάλη υπομονή τους όλα αυτά τα χρόνια και τη συνεχή στήριξή τους στις αποφάσεις μου καθώς και για τους κοντινότερους ανθρώπους στη ζωή μου για όσα μου έχουν δώσει και όσα έχουνε κάνει για εμένα.

This page is intentionally left blank.

Acknowledgements

For the current diploma thesis, completed in the Microprocessors and Digital Systems lab of the Electrical and Computer Engineering Department of the NTUA, I would like to sincerely thank my professor, Mr Dimitrios Soudris for his confidence in me and my work, and his constant support throughout my presence in the lab.

I also thank Professor Constantinos Karaνtzalos from the School of Rural and Surveying Engineering of the NTUA, for the great interest shown in our common project from the beginning of our work together. Our constructive and definitely creative meetings greatly encouraged me during my work.

The completion of this thesis however would be impossible without the enormous support and cooperation with George Lentaris, postdoctoral researcher and Konstantinos Maragkos, PhD student in the MicroLab. I could not thank them enough in this note but for their interest and their great contribution throughout my work I am sincerely grateful.

A special thanks also goes to Zacharias Kandylakis, PhD student in the Remote Sensing Laboratory of the School of Rural and Srurveying Engineering at the NTUA. For his great help and his much appreciated constantly supportive and good mood, I thank him warmly.

For the pleasant and creative atmosphere in the laboratory where I spend almost every day during this past year, I would like to thank all the people working in the lab, post graduate and PhD students and researchers. We supported each other when needed, had great discussions and spent many hours together. These moments will surely be missed.

Finally and last but not least, a huge thank you to my family for their constant support and for being incredibly patient throughout my study years as well as to the closest people in my life for everything they have done for me and for everything we’ve been through together.

This page is intentionally left blank.

Contents

Εκτεταμένη Περίληψη

Σκοπός της Διπλωματικής Εργασίας

Σκοπός της παρούσας διπλωματικής εργασίας είναι η μελέτη και αξιολόγηση της υλοποίησης ενός συστήματος υπερφασματικής απεικόνισης σε κύκλωμα FPGA. Συγκεκριμένα, υλοποιήσαμε ένα σύστημα για το ταίριασμα των στοιχείων μιας εικόνας υπερφασματικής ανάλυσης με έναν συγκεκριμένο αριθμό φασματικών υπογραφών, αποθηκευμένες στη μνήμη του συστήματός μας. Με αυτή τη διαδικασία, μπορούμε να βρούμε μέσω συγκεκριμένων μετρικών ταιριάσματος, τη φασματική υπογραφή που αντιστοιχεί σε κάθε στοιχείο, επιτρέποντάς μας την αναγνώριση δεδομένων στην εικόνα.
Η επιλογή της πλατφόρμας του FPGA για την υλοποίηση του συστήματός μας έγινε καθώς το FPGA έχει ορισμένα πλεονεκτήματα τα οποία αποδεικνύονται ιδανικά για την αντιμετώπιση των ιδιαίτερα αυξημένων απαιτήσεων σε επεξεργαστική ισχύ της εφαρμογή που θέλουμε να υλοποιήσουμε.

Εξετάσαμε συγκεκριμένα ορισμένες υλοποιήσεις του προαναφερθέντος συστήματος για τρεις διαφορετικές μετρικές σύγκρισης. Τα αποτελέσματα καταγράφηκαν εκτενώς ως προς τις απαιτήσεις σε υλικό όσο και τα αποτελέσματα σε επιτάχυνση σε σχέση με αντίστοιχες υλοποιήσεις σε C κώδικα τις οποίες τρέξαμε σε έναν τυπικό επεξεργαστή ενός προσωπικού υπολογιστή καθώς και σε έναν επεξεργαστή ενσωματωμένων συστημάτων.

Για την εξαγωγή ασφαλών συμπερασμάτων, προβήκαμε στην εκτενή ανάλυση των ποιοτικών χαρακτηριστικών των τριών διαφορετικών μετρικών σύγκρισης που υλοποιήσαμε. Η παραπάνω μελέτη έγινε σε MATLAB και για τη διενέργεια των πειραμάτων χρησιμοποιήθηκαν εικόνες υπερφασματικής ανάλυσης με εκατοντάδες φασματικά κανάλια.

Κλείνοντας τη διπλωματική, προβήκαμε στη μελέτη των συμβιβασμών που πρέπει να γίνουν μεταξύ της μέγιστης επιτεύξιμης επίδοσης του κυκλώματός μας και των ποιοτικών αποτελεσμάτων για την εκάστοτε εφαρμογή. Η παραπάνω ανάλυση μας οδήγησε στην εξαγωγή ιδιαίτερα σημαντικών συμπερασμάτων για τις δυνατότητες του συστήματός μας καθώς και την ανάγκη προσαρμογής στις απαιτήσεις της εκάστοτε εφαρμογής.

Υπερφασματική Απεικόνιση

Η υπερφασματική απεικόνιση είναι μια τεχνική απεικόνισης κατά την οποία συλλέγονται δεδομένα από ένα ευρύ φάσμα του ηλεκτρομαγνητικού φάσματος. Σε αντίθεση με τις κοινές κάμερες τριών καναλιών οι οποίες περιορίζονται στη συλλογή και αποτύπωση δεδομένων εντός του ορατού φάσματος, η υπερφασματική κάμερα έχει τη δυνατότητα

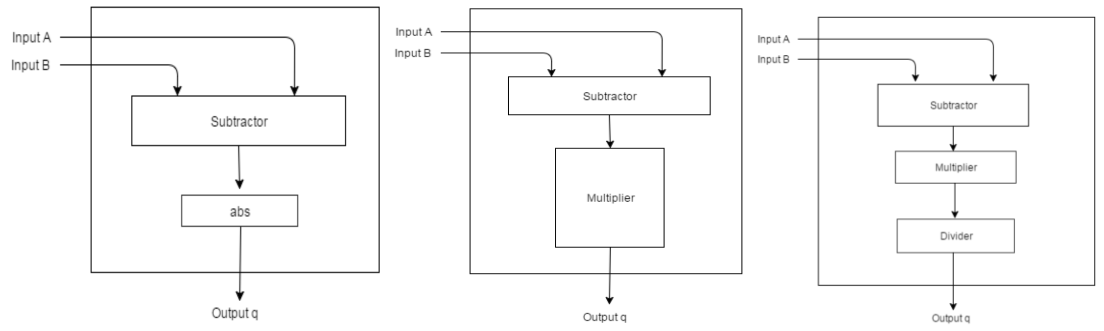
συλλογής δεδομένων σε συγκεκριμένα μήκη κύματος και την αποτύπωσή τους σε εκατοντάδες φασματικά κανάλια παρέχοντας τη δυνατότητα καταγραφής και αναγνώρισης ενός πολύ μεγαλύτερου όγκου δεδομένων απ’ ότι είναι δυνατό με συμβατικές κάμερες.
Η μορφή της καμπύλης στο ηλεκτρομαγνητικό φάσμα για κάθε εικονοστοιχείο μπορεί να χρησιμοποιηθεί στη συνέχεια για την αναγνώριση και ταξινόμησή του.

Κάθε στοιχείο που μας ενδιαφέρει σε μια υπερφασματική εικόνα έχει μια μοναδική απεικόνιση στο ηλεκτρομαγνητικό φάσμα η οποία καλείται φασματική υπογραφή. Γνωρίζοντας αυτήν την υπογραφή ενός αντικειμένου, ενός υλικού ή οποιουδήποτε στοιχείου που μας ενδιαφέρει, μπορούμε να αναγνωρίσουμε και να ταξινομήσουμε αντικείμενα, υλικά ή ολόκληρες περιοχές με κοινά χαρακτηριστικά σε μια εικόνα.

Μετρικές ταιριάσματος φασματικών υπογραφών

Προκειμένου να αναγνωρίσουμε χαρακτηριστικά που μας ενδιαφέρουν σε μια εικόνα υπερφασματικής ανάλυσης, προσπαθούμε να ταιριάξουμε κάθε φάσμα σε μια εικόνα με ένα ήδη γνωστό φασματικό «αποτύπωμα» από μια βιβλιοθήκη υπογραφών. Ένας τρόπος για το ταίριασμα αυτό είναι ο υπολογισμός της διανυσματικής απόστασης του φασματικού διανύσματος ενός εικονοστοιχείου με το αντίστοιχο διάνυσμα κάθε υπογραφής. Στη συνέχεια, με κάποιο κριτήριο βέλτιστου αποτελέσματος, ταξινομούμε τα αποτελέσματα διαλέγοντας από αυτά την καλύτερη υπογραφή. Έτσι, κατασκευάζουμε έναν χάρτη από την αρχική εικόνα ο οποίος στην συνέχεια μπορεί να χρησιμοποιηθεί μαζί με άλλα κριτήρια για την αναγνώριση υλικών, αντικειμένων ή χαρακτηριστικών στην εικόνα.

Στην παρούσα εργασία εξετάζουμε τρεις κοινές μετρικές για την διεξαγωγή του παραπάνω ταιριάσματος με βάση την μέτρηση διανυσματικών αποστάσεων. Αυτές οι τρεις μετρικές μελετήθηκαν εκτενώς σε ένα σύνολο εικόνων διαφορετικών χαρακτηριστικών προκειμένου να βγάλουμε αξιόπιστα συμπεράσματα για τη συμπεριφορά τους σε διάφορες συνθήκες.
Οι τρεις μετρικές αυτές είναι το άθροισμα απολύτων διαφορών, η ευκλείδεια απόσταση και η κανονικοποιημένη τετραγωνική απόσταση.



Σχήμα 1: Οι τρεις μετρικές ταιριάσματος που υλοποιήθηκαν στη διπλωματική εργασία.

Στη συνέχεια ακολουθούν και οι μαθηματικοί τύποι των τριών μετρικών που υλοποιήσαμε.

Άθροισμα απολύτων διαφορών:

$$SAD = \sum_{i=1}^{N} \left| x_i - y_i \right|$$

Ευκλείδεια απόσταση σημείων:

$$\sqrt{\sum_{i=1}^{N} \left(x_i - y_i \right)^2}$$

Κανονικοποιημένη τετραγωνική απόσταση σημείων:

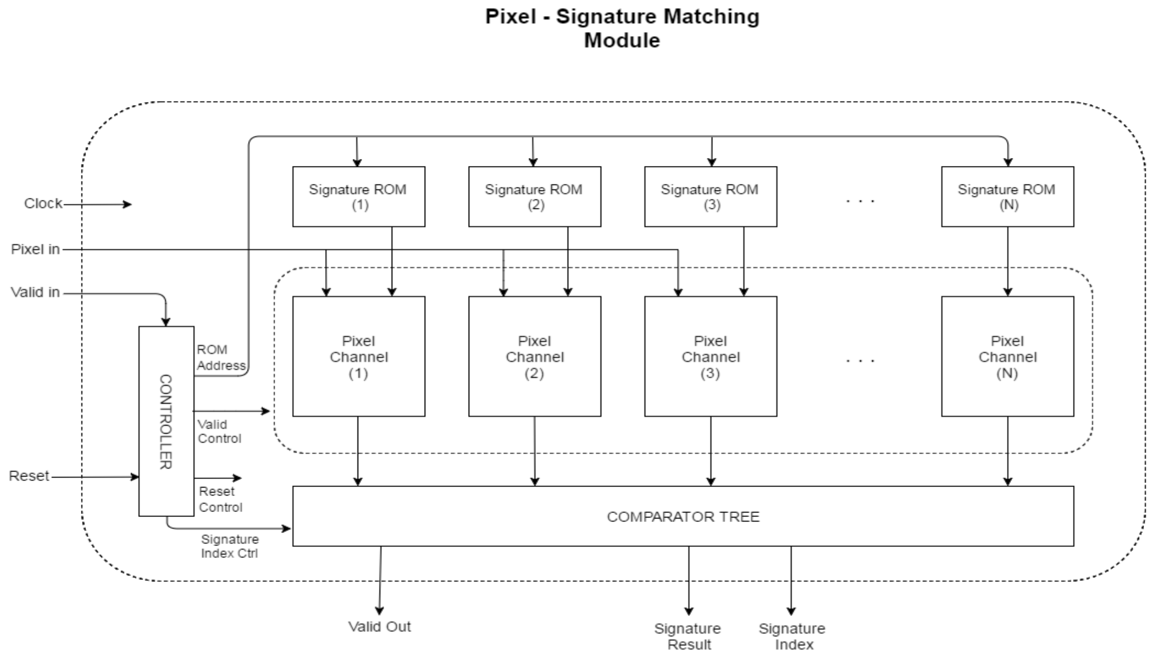
$$\sum_{i=1}^{N} \frac{\left(x_i - y_i \right)^2}{(x_i + y_i)}$$

Επιτάχυνση με FPGA

Η τεχνολογική πρόοδος των τελευταίων ετών σε πολλούς τομείς έχει επιφέρει μια ανάγκη για αυξημένη επεξεργαστική ισχύ ιδιαίτερα σε ενσωματωμένα συστήματα και συστήματα χαμηλής ενέργειας. Επιπλέον, οι επαναδιαμορφούμενες αρχιτεκτονικές καλύπτουν ένα όλο και μεγαλύτερο κομμάτι από αυτές τις εφαρμογές.
Στην παρούσα εργασία επικεντρωνόμαστε στη χρήση των FPGA οι οποίες είναι επεξεργαστικές μονάδες που βασίζονται σε παραμετροποιήσιμα λογικά μπλοκ για την εκτέλεση σχεδόν οποιασδήποτε εφαρμογής με χρήση ειδικών γλωσσών περιγραφής υλικού όπως η VHDL. Το πλεονέκτημα αυτών των μονάδων είναι η δυνατότητα υλοποίησης συστημάτων πολύ υψηλής παραλληλίας επιτυγχάνοντας με αυτόν τον τρόπο επιδόσεις πολύ υψηλότερες από κοινές επεξεργαστικές μονάδες.
Εμείς χρησιμοποιήσαμε την αναπτυξιακή πλακέτα Zynq-7000 της Xilinx η οποία είναι ένα σύστημα υλοποιημένο σε ολοκληρωμένο κύκλωμα και συνδυάζει έναν επεξεργαστή ενσωματωμένων συστημάτων με FPGA. Για τον προγραμματισμό της πλακέτας χρησιμοποιήθηκε VHDL για την περιγραφή του κυκλώματος και η σουίτα της Xilinx για την απεικόνιση του συστήματος στο FPGA.

Υλοποίηση Συστήματος Υπερφασματικής ανάλυσης σε FPGA

Το σύστημα που δημιουργήσαμε με VHDL αποτελείται έχει ορισμένα χαρακτηριστικά τα οποία μας επιτρέπουν μια μεγάλη παραμετροποίηση και επεξεργασία του κυκλώματός μας.
Συγκεκριμένα, δημιουργήσαμε ένα σύστημα τεσσάρων επιπέδων παραλληλίας όπου μπορούμε εύκολα να επιλέξουμε τον κατάλληλο αριθμό παράλληλων επεξεργαστικών μονάδων όσο αναφορά την παραλληλία καναλιών, την παράλληλη επεξεργασία φασματικών υπογραφών καθώς και την παράλληλη επεξεργασία εικονοστοιχείων. Επίσης, οι επεξεργαστικές μονάδες για τον υπολογισμό της διανυσματικής απόστασης είναι εναλλάξιμες και μας επιτρέπουν την εύκολη τροποποίηση του κυκλώματος ανάλογα την εφαρμογή. Στο επόμενο σχήμα δίνεται μια γενική εικόνα του κυκλώματος από το ανώτατο επίπεδο αφαίρεσης.



Σχήμα 2: Διάγραμμα συστήματος ταιριάσματος φασματικών υπογραφών εικόνας υπερφασματικής ανάλυσης.

Η λειτουργία του παραπάνω κυκλώματος είναι η εξής: Εισερχόμενα εικονοστοιχεία μιας εικόνας προς επεξεργασία προωθούνται σε κάθε παράλληλη επεξεργαστική μονάδα μαζί με μια γνωστή φασματική υπογραφή που βρίσκεται αποθηκευμένη στη μνήμη του συστήματος. Ο μέγιστος αριθμός υπογραφών προφανώς ορίζει και τη θεωρητική μέγιστη δυνατή παραλληλία υπογραφών. Κάθε μονάδα στη συνέχεια προωθεί ορισμένα από τα φασματικά κανάλια σε ανεξάρτητες και παράλληλες επεξεργαστικές μονάδες οι οποίες με τη σειρά τους υπολογίζουν την απόσταση των σημείων του διανύσματος. Το αποτέλεσμα της διαφοράς κάθε καναλιού αθροίζεται με τη χρήση ενός δέντρου αθροιστών και στη συνέχεια προωθείται εκ νέου στην προηγούμενη μονάδα η οποία για όλες τις υπογραφές τροφοδοτεί ένα δέντρο συγκριτών με τα αποτελέσματα κάθε σύγκρισης ζεύγους εικονοστοιχείου / υπογραφής μέσα από τον οποίο επιλέγεται το «καλύτερο» αποτέλεσμα και προωθείται στην έξοδο μαζί με την αντίστοιχη υπογραφή.

Υλοποιήσεις και Αποτελέσματα

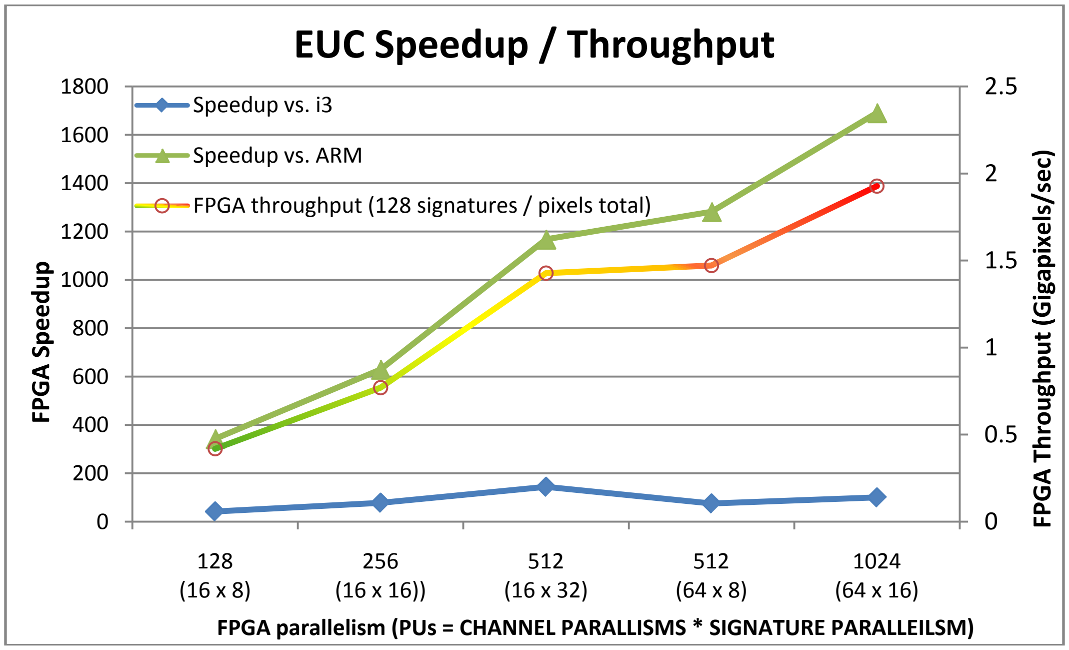
Το σύστημα που κατασκευάστηκε σε VHDL υλοποιήθηκε και μετρήθηκε στο FPGA με διαφορετικές τιμές παραλληλίας. Κάθε υλοποίηση αξιολογήθηκε ως προς το χρόνο εκτέλεσης και την κατανάλωση πόρων της συσκευής. Τα αποτελέσματα από κάθε υλοποίηση συγκρίθηκαν στη συνέχεια με τα αποτελέσματα των αντίστοιχων υλοποιήσεων σε C στον επεξεργαστή της Intel και τον ενσωματωμένο επεξεργαστή της ARM. Στη συνέχεια υπολογίστηκε η επιτάχυνση για κάθε περίπτωση καθώς και το throughput δεδομένων εισόδου. Στο Σχήμα 3 δίνονται τα διαγράμματα επιτάχυνσης και throughput για έξι υλοποιήσεις διαφορετικής παραλληλίας. Η επιτάχυνση που επιτύχαμε είναι ξεκάθαρη καθώς επίσης και ο περιορισμός που εισάγεται από τη δυνατότητα τροφοδοσίας του συστήματος με νέα δεδομένα.
Στους Πίνακες 1 και 2, παρουσιάζονται συνοπτικά αντιπροσωπευτικά αποτελέσματα επιτάχυνσης και κατανάλωσης πόρων για ορισμένες υλοποιήσεις.

16 CHANNELS	**UTILIZATION (ZYNQ 7045)**		
16 SIG. PARALLELISM	**SAD**	**EUC**	**CHI-SQUARED**
REGISTERS	23.8K (5%)	24.7K (5%)	281K (64%)
LUTs	16.4K (7%)	17.6K (8%)	81.7K (37%)
DSPs	0	256 (28%)	256 (28%)

Πίνακας 1: Κατανάλωση πόρων τριών μετρικών για την ίδια υλοποίηση

FPGA Configuration		**Data**	**SAD Execution time**			**Speedup**	
Signature Parallelism	**Channel Parallelism**	**Channels**	**Software (C)**		**VHDL**	**VHDL vs.**	
			i3	**ARM**	**FPGA**	**i3**	**ARM**
8	16	16	0.37 sec	3.5 sec	9.35 ms	40	374
	64	64	0.52 sec	26.72 sec	12.38 ms	46	2377
		256	3.23 sec	106.88 sec	49.52 ms	72	2377
16	16	16	0.75 sec	6.73 sec	10.75 ms	70	626
	64	64	1.03 sec	40.36 sec	13.94 ms	74	2895
		256	6.99 sec	161.44 sec	55.76 ms	125	2895
32	16	16	1.45 sec	12.87 sec	11.33 ms	128	1136
	64	64	2.02 sec	83.07 sec	24.85 ms	81	3343
		256	13.17 sec	332.28 sec	99.4 ms	133	3612

Πίνακας 2: Επιτάχυνση VHDL vs. i3 & ARM για επιλεγμένες αντιπροσωπευτικές υλοποιήσεις



Σχήμα 6: Πειραματικά αποτελέσματα επιτάχυνσης / throughput για έξι υλοποιήσεις

Είναι εμφανές από τα παραπάνω αποτελέσματα πως οι μετρικές ταιριάσματος έχουν σημαντικές διαφορές μεταξύ τους ως προς την κατανάλωση πόρων του συστήματος. Η υλοποίηση της διαίρεσης είναι, ως γνωστόν, πολύ «ακριβή» μειώνοντας σημαντικά τη μέγιστη επιτεύξιμη παραλληλία. Η παρατήρηση αυτή είναι ιδιαίτερα σημαντική για την επιλογή της βέλτιστης σχεδίασης. Επιπλέον, είναι σαφές πως ο κυριότερος περιοριστικός παράγοντας για την επίτευξη πολύ υψηλών επιδόσεων είναι η τροφοδοσία δεδομένων εισόδου στο FPGA. Το throughput που απαιτείται από το σύστημά μας αποτελεί μια μεγάλη πρόκληση για το σχεδιαστή και προφανώς θέτει τα όρια του συστήματός μας.

Ποιοτική Αξιολόγηση

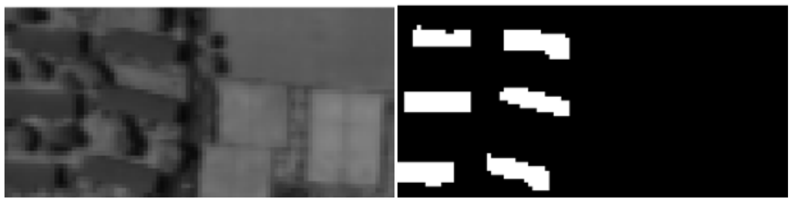
Μετά τη μελέτη και αξιολόγηση του συστήματός μας και των τριών διαφορετικών μετρικών σε επίπεδο υλικού, προχωρούμε στην ποιοτική μελέτη του συστήματος ταιριάσματος φασματικών υπογραφών.
Η μελέτη αυτή, κρίνεται απαραίτητη για την εξαγωγή συμπερασμάτων ποιοτικής φύσης για τις τρεις μετρικές ταιριάσματος. Τα αποτελέσματα αυτά θα μας επιτρέψουν στο τέλος να εξάγουμε σαφή συμπεράσματα για την επιλογή της καλύτερης διάταξης για κάθε εφαρμογή.

Υπερφασματικές Εικόνες που Χρησιμοποιήθηκαν

Για την διεξαγωγή των πειραμάτων χρησιμοποιήσαμε δυο υπερφασματικές εικόνες διαφορετικών χαρακτηριστικών. Η κύρια εικόνα που χρησιμοποιήσαμε προέρχεται από το πρόγραμμα APEX της ESA και αποτελείται από 285 υπερφασματικά κανάλια τα οποία καλύπτουν το ηλεκτρομαγνητικό φάσμα από τα 400nm - 2500nm περίπου. Για τη συγκεκριμένη εικόνα εξετάσαμε πέντε διαφορετικές περιπτώσεις διαφορετικών χαρακτηριστικών. Επιπλέον, επαναλάβαμε τα ίδια πειράματα για διαφορετικές διατάξεις υποθετικών καμερών και συνθηκών.
Τέλος, το δεύτερο σετ εικόνων που χρησιμοποιήθηκε ελήφθη με κάμερες υπερφασματικής ανάλυσης 16 καναλιών του εργαστηρίου τηλεπισκόπισης της σχολής των τοπογράφων μηχανικών του ΕΜΠ. Τα συγκεκριμένα δεδομένα χρησιμοποιήθηκαν για την εξαγωγή σημαντικών ποιοτικών συμπερασμάτων και την ανάλυση διαφόρων παραγόντων και όχι για τη μελέτη συγκεκριμένων ποσοτικών πειραμάτων.

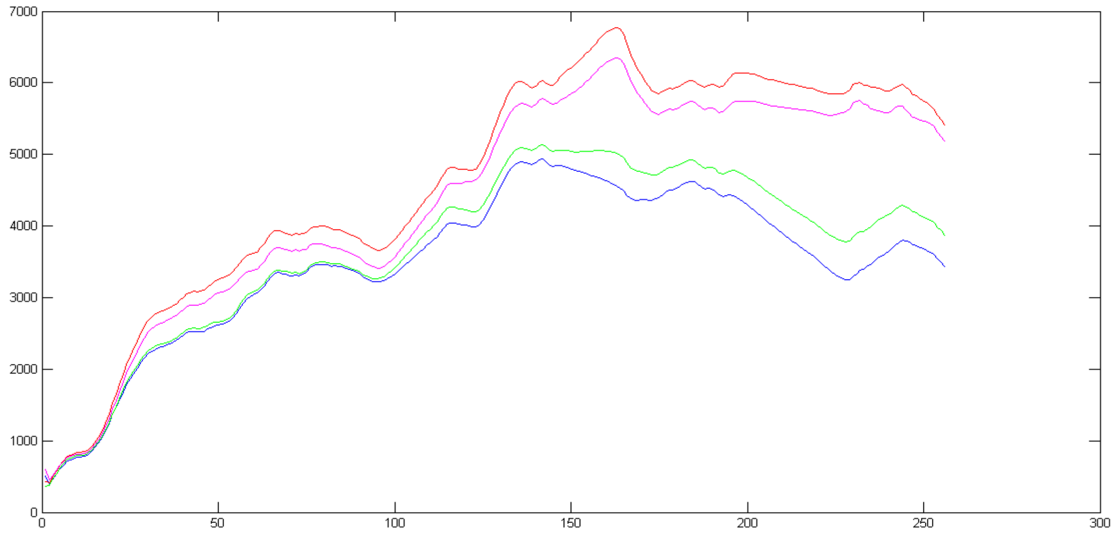
Δημιουργία Δεδομένων Ελέγχου και Φασματικών Υπογραφών

Για τη διεξαγωγή των πειραμάτων, χρησιμοποιήσαμε ως κύρια εικόνα, μια υπερφασματική εικόνα 285 καναλιών μεγέθους 1000×1000 εικονοστοιχεία. Από τη συγκεκριμένη εικόνα δημιουργήσαμε έξι διαφορετικά δεδομένα ελέγχου τα οποία χρησιμοποιήθηκαν στη συνέχεια για την ποιοτική ανάλυση. Κατασκευάσαμε περιοχές ελέγχου για: νερό, γήπεδα τένις, σκεπές, δρόμους, γήπεδο με γρασίδι και βλάστηση (δέντρα).
Ένα παράδειγμα δεδομένων ελέγχου φαίνεται στο Σχήμα 7. Με χρήση του MATLAB δημιουργήσαμε περιοχές που μας ενδιέφεραν από την αρχική εικόνα σε μορφή δυαδικού χάρτη.



Σχήμα 7: Περιοχή ελέγχου για σκεπές

Από τις περιοχές ελέγχου, αντλήσαμε στη συνέχεια ορισμένα εικονοστοιχεία τα οποία χρησιμοποιήσαμε ως φασματικές υπογραφές για τη συγκεκριμένη περιοχή. Στη γενική περίπτωση χρησιμοποιήσαμε περίπου 5-6 υπογραφές για κάθε περιοχή καθώς μας ενδιέφερε να εξετάσουμε τη διαφορά στα ποιοτικά αποτελέσματα για διαφορετικό αριθμό υπογραφών. Ένα παράδειγμα τέτοιων φασματικών υπογραφών δίνεται στο Σχήμα 8.



Σχήμα 8: Τέσσερις φασματικές υπογραφές από την περιοχή των γηπέδων τένις.

Δείκτες Ποιότητας και Πειραματικά Αποτελέσματα

Για την ποσοτικοποίηση της έννοιας της ποιότητας, χρησιμοποιήσαμε τρεις βασικούς δείκτες ποιότητας που χρησιμοποιούνται συχνά σε ποιοτικές μελέτες εικόνων.
Οι δείκτες αυτοί είναι:

Δείκτης Πληρότητας (Completeness) = $\frac{TP}{TP+FN}$

Δείκτης Ορθότητας (Correctness) = $\frac{TP}{TP+FP}$

Δείκτης Ποιότητας (Quality) = $\frac{TP}{TP+FN+FP}$

Οι παραπάνω τρεις δείκτες μας δίνουν αποτελέσματα σε επί τοις εκατό τα οποία μπορούμε εμείς να ερμηνεύσουμε για την εξαγωγή απαραίτητων συμπερασμάτων.

Αποτελέσματα Ανάλυσης

Για την κάλυψη όλων των πιθανών περιπτώσεων μελέτης, υλοποιήσαμε σε MATLAB ένα μοντέλο με το οποίο μετρήσαμε για κάθε μία από τις περιοχές ελέγχου μας τους παραπάνω δείκτες. Οι μετρήσεις έγιναν αρχικά για μια φασματική υπογραφή κάθε περιοχής για όλες τις διαφορετικές υποθετικές διατάξεις της κάμεράς μας. Η παραπάνω διαδικασία επαναλήφθηκε για περισσότερες των μια υπογραφών και για διαφορετικές συνθήκες φωτισμού (υποθετική μείωση φωτεινότητας). Στον Πίνακα 3 δίνονται τα αποτελέσματα για μια αντιπροσωπευτική περίπτωση μελέτης. Αντίστοιχα, στον Πίνακα 4 δίνονται αποτελέσματα για υλοποιήσεις περισσότερων υπογραφών και / ή μειωμένης φωτεινότητας.

	Quality									
	DATASET	**WATER**			**ROOF**			**COURT**		
	METRIC	SAD	EUC	X-2	SAD	EUC	X-2	SAD	EUC	X-2
CAMERAS	**Full Range** 256 Ch.	97.30%	96.87%	97.43%	78.37%	77.88%	78.03%	55.88%	54.44%	67.48%
	Near Infrared ~40 Ch.	95.92%	96.02%	96.00%	75.95%	75.84%	76.67%	73.06%	75.83%	78.05%
	Short-wave Infrared ~ 50 Ch.	97.61%	97.37%	97.08%	78.36%	77.87%	77.98%	32.43%	34.63%	36.75%
	Compact ~80 Ch.	97.27%	96.92%	97.45%	78.29%	77.88%	78.06%	55.64%	54.24%	67.28%
	Multi Spectral ~25 Ch. 10 Ch. Av.	97.34%	96.94%	97.40%	78.30%	77.88%	78.06%	55.05%	53.75%	66.13%

Πίνακας 3: Αποτελέσματα δείκτη ποιότητας για 5 υποθετικές κάμερες και 3 περιοχές ελέγχου.

COURT								
1 Signature 100% Luminosity			3 Signatures 100% Luminosity			3 Signatures 60% Luminosity		
SAD	EUC	X-2	SAD	EUC	X-2	SAD	EUC	X-2
55.88%	54.44%	67.48%	66.35%	67.82%	75.92%	51.16%	52.30%	62.40%

Πίνακας 4: Ενδεικτικά αποτελέσματα για περισσότερες υπογραφές και μειωμένη φωτεινότητα.

Ιδιαίτερα σημαντική είναι η παρατήρηση ότι σχεδόν σε όλες τις παραπάνω περιπτώσεις η μετρική Ευκλείδειας απόστασης παρέχει χειρότερα αποτελέσματα από την απόλυτη διαφορά. Η σημασία της συγκεκριμένης παρατήρησης έγκειται στο γεγονός ότι η υλοποίηση του Ευκλείδη στο FPGA είναι πιο ακριβή από πλευράς υλικού. Συνεπώς, δεν έχει νόημα η υλοποίηση της ευκλείδειας απόστασης καθώς δε δικαιολογείται από πλευράς ποιότητας αποτελεσμάτων.
Για την κανονικοποιημένη απόσταση μπορούμε να εξάγουμε το συμπέρασμα ότι σχεδόν για κάθε υλοποίηση, η μετρική μας δίνει καλύτερα αποτελέσματα σε σχέση με τις άλλες δυο εναλλακτικές. Η βελτίωση ωστόσο στις περισσότερες περιπτώσεις υπό κανονικές συνθήκες δε δικαιολογεί τις ιδιαίτερα αυξημένες απαιτήσεις σε υλικό λόγω της διαίρεσης. Ως εκ τούτου, είναι σημαντικό να γίνει καλή ανάλυση του προβλήματος και να εξεταστούν εναλλακτικές υλοποιήσεις πριν εφαρμοστεί η διαίρεση ως λύση.
Το παραπάνω γίνεται εύκολα αντιληπτό αν παρατηρήσουμε πως στις περισσότερες περιπτώσεις και οι τρεις μετρικές αναγνωρίζουν το αντικείμενο ή την περιοχή στην εικόνα μας. Έχοντας αυτή την πληροφορία, μπορούμε με τη βοήθεια άλλων τεχνικών να εξάγουμε τα επιθυμητά χαρακτηριστικά από την εικόνα μας. Η επάρκεια ή όχι της κάθε μεθόδου-μετρικής ορίζεται προφανώς από τις απαιτήσεις της εφαρμογής μας.

Σημαντικό είναι επίσης το γεγονός ότι τα αποτελέσματα βελτιώθηκαν σχεδόν σε κάθε υλοποίηση στην οποία εξετάσαμε ταυτόχρονα περισσότερες υπογραφές από την ίδια περιοχή. Το συμπέρασμα αυτό μας αρέσει καθώς η αρχιτεκτονική του FPGA είναι τέτοια που επωφελείται από αυξημένα δεδομένα προς επεξεργασία.

Τέλος, βλέπουμε από τους παραπάνω πίνακες αποτελεσμάτων πως οι κάμερες με περισσότερα κανάλια δεν δίνουν απαραίτητα και καλύτερα ποιοτικά αποτελέσματα. Στην περίπτωση σύγκρισης φασματικών υπογραφών παρατηρούμε πως ανάλογα το αντικείμενο ή το υλικό που ψάχνουμε, υπάρχουν περιοχές στο φάσμα οι οποίες παρουσιάζουν τη μεγαλύτερη διαφορά. Για παράδειγμα η βλάστηση έχει μεγάλη παρουσία στο ορατό φάσμα. Αυτό το χαρακτηριστικό των φασματικών υπογραφών μας δείχνει πως δε χρειαζόμαστε τα υπόλοιπα κανάλια.
Η σημασία αυτής της παρατήρησης είναι μεγάλη για την υλοποίηση του συστήματος στο FPGA όπου μειωμένος αριθμός καναλιών μεταφράζεται σε μεγαλύτερη παραλληλία και υψηλότερη επιτάχυνση.

Συμπεράσματα

Είδαμε στην παρούσα εργασία τα πλεονεκτήματα της τεχνολογίας του FPGA στην επίτευξη υψηλών επιδόσεων. Επιτύχαμε μια επιτάχυνση τουλάχιστον μίας τάξης μεγέθους σε σχέση με μια υλοποίηση σε έναν κοινό επεξεργαστή και έως τριών τάξεων μεγέθους σε σχέση με έναν επεξεργαστή ενσωματωμένου συστήματος. Η επίτευξη αυτών των αποτελεσμάτων δεν είναι εύκολη διαδικασία και απαιτεί μια αυξημένη προσπάθεια από μεριά του σχεδιαστή.
Είδαμε πως κάθε υλοποίηση επιφέρει ορισμένους περιορισμούς που πρέπει να αντιμετωπιστούν και να αναλυθούν ενώ η εξέταση των ποιοτικών χαρακτηριστικών του συστήματος ταιριάσματος μας έδωσε σημαντικά αποτελέσματα, ιδιαίτερα χρήσιμα για την επίτευξη ενός συμβιβασμού μεταξύ υλικού και λογισμικού και την βέλτιστη υλοποίηση της εκάστοτε εφαρμογής.

This page is intentionally left blank.

List of Figures

This page is intentionally left blank.

List of Tables

This page is intentionally left blank.

CHAPTER 1

INTRODUCTION

1.1 Objective summary

The objective of this diploma thesis is the evaluation of the performance and tradeoffs of the implementation of a hyperspectral imaging system on reconfigurable hardware, specifically a field programmable gate array circuit (FPGA). Hyperspectral imaging has been an area in recent years of great research and development with the technology entering more and more the mainstream of remote sensing. The main gain of this technology, as will be extensively analyzed in the following chapters, is the increased detail and accuracy in image analysis it delivers. This advantage is of great importance for a wide range of applications in areas like geology and agriculture, surveillance, physics and astronomy. In order to use this technology at its full potential, the understanding of the limitations and challenges it carries with it is required. Great detail in image analysis includes the necessity of greater processing power and data manipulation abilities. This limitation can be efficiently handled with parallel programming techniques and hardware acceleration technologies. The aforementioned reconfigurable hardware is an ideal way to tackle these problems.

In this sense, our goal in this thesis is to show how the effective use of an FPGA platform and its processing capabilities greatly increase the ability to process large amounts of data simultaneously and with increased speed. In order to show the capabilities of the examined platform, we implemented several data processing configurations covering a wide range of available cases. We did pay particular attention in taking advantage of the parallel processing capabilities of our platform implementing several levels of parallelism as well as numerous configurations for a large amount of data. Furthermore, we examined different algorithms for our system showing the impact of different implementations on hardware resources and performance.

In order to cover the impact of the different algorithms on the final results, we conducted a quality analysis using MATLAB for the different matching techniques implemented on our device. We examined the impact of three different matching algorithms in terms of qualitative results for various image data and camera configurations performed on two hyperspectral image data cubes. The first one was obtained in collaboration with the Remote Sensing Laboratory of the National Technical University of Athens (NTUA) using two hyperspectral cameras and a number of different objects. The second data cube we examined was an open science dataset available online from APEX (Airborne Prism Experiment), an imaging spectrometer developed on behalf of ESA [1]. We concluded our work combining the results for the above analysis with the hardware implementation results to perform a tradeoff analysis taking into account the benefits and limitations of each case.

1.2 Thesis structure

We will describe at this point with a few words the structure of the thesis and the chapters we are going to follow. This work consists of 7 Chapters which are organized as follows:

- Chapter 2 talks about Hyperspectral imaging starting with a short introduction to the fundamentals of this technology, moving on to a more in depth description of data and signature acquisition as well as spectral characteristics. We will describe methods used for spectral matching and will conclude the chapter presenting the datasets used in the thesis.

- In chapter 3 we get into the theory of hardware design focusing on reconfigurable architectures and hardware accelerators mentioning their advantages and disadvantages as well as their importance in modern applications. We introduce the FPGA circuit and the hardware description languages (HDL) mainly used to program such devices. We will also talk about the different programming tools used for programming an FPGA and will close the chapter with a brief description of the specific board used in the thesis, its main features and advantages as well as the challenges faced.

- Chapter 4 is the chapter where we analyze the hardware implementation of our system. We describe the architecture of our design and examine in depth its structure and the chosen strategies. A detailed recording of all the results follows, covering performance improvements over conventional implementations, resource utilization and examination of the limitations imposed by the system.

- In this chapter, having already concluded the hardware implementation of our system and having documented the results in the previous chapter, we move on to developing a system on MATLAB in order to measure the qualitative behavior of different matching metrics and camera configurations. Furthermore, we will describe the method used to create ground truth data, acquire signatures for each ground truth and how we conducted the experiments. An extended documentation of the results will follow including numerous measures for different quality metrics, camera configuration impact on the results. We close this chapter explaining the obtained results and the effects of each factor on the final data.

- Chapter 6 is devoted to the correlation of the results of the two analyses and the effect they have on each other. A tradeoff analysis is conducted explaining in detail the gains and losses of each design decisions on both hardware and software and the compromises a designer has to made in order to satisfy design specifications.

- Finally, we conclude our work summarizing the results of our work and making proposals for future research to be made.

CHAPTER 2

HYPERSPECTRAL IMAGING

2.1 Introduction to Hyperspectral Imaging

Hyperspectral imaging, or imaging spectroscopy, is a spectral imaging technique that, like all spectral imaging techniques, collects image information from across the electromagnetic spectrum and subsequently processes it. The main characteristic of hyperspectral imaging is that for each pixel in an image, the camera acquires a large number of spectral bands from a much wider area of the light spectrum than a normal 3-band camera. Each spectral band consists of a specific range of wavelengths and the bands extending usually beyond the visible spectrum. Whereas a normal camera and the human eye collect and process light in mostly three bands (red, green, blue) in the visible spectrum, a hyperspectral imaging system can obtain imagery over hundreds of narrow, continuous spectral bands with typical bandwidths of 10 nanometers or less over a spectral range from 400 to 2500 nanometers (visible through middle infrared wavelength ranges). [2][3]

Each pixel in a hyperspectral image contains both spatial and spectral information from materials within a scene. In order to understand this, think of each image captured by the camera as a representation of a narrow spectral band. These images are combined to form a three-dimensional (x, y, λ) hyperspectral data cube where x and y represent the two spatial dimensions of the scene and λ represents the spectral dimension. The third dimension can be considered as a collection of images, each one acquired at a different wavelength as can be seen in Figure 2.1. [4] Besides spectral resolution, it is important to also take into account spatial resolution. In order to identify features in the scene, for example a distinct material, it has to be large enough to cover a whole pixel or, respectively, the pixel would have to be small enough to capture only one material. If the camera resolution is too low, then each pixel would cover multiple objects making the image features difficult to identify.

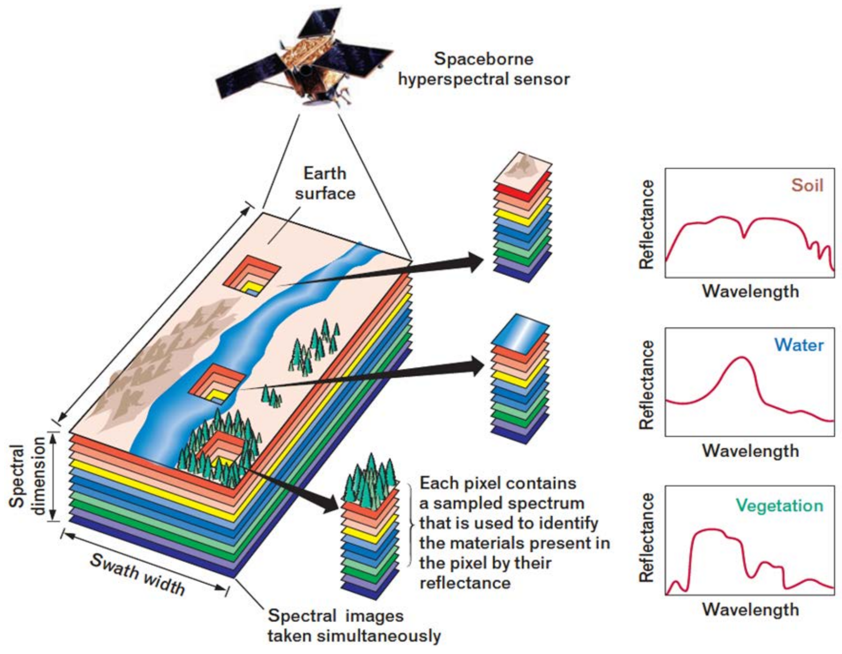

Spaceborne hyperspectral sensor
Earth surface
Spectral dimension
Swath width
Spectral images taken simultaneously
Each pixel contains a sampled spectrum that is used to identify the materials present in the pixel by their reflectance
Soil
Reflectance
Wavelength
Water
Reflectance
Wavelength
Vegetation
Reflectance
Wavelength

Figure 2.1 Illustration of the hyperspectral imaging concept

Hyperspectral images provide much more information about the captured scene than a normal camera leading to an enormous improvement in the ability to classify objects based on their spectral properties [5]. In Figure 2.2, you can see the electromagnetic spectrum ranging from 400nm to 14000nm with a brief reference of the main properties of each region as well as applications where these properties are of benefit.

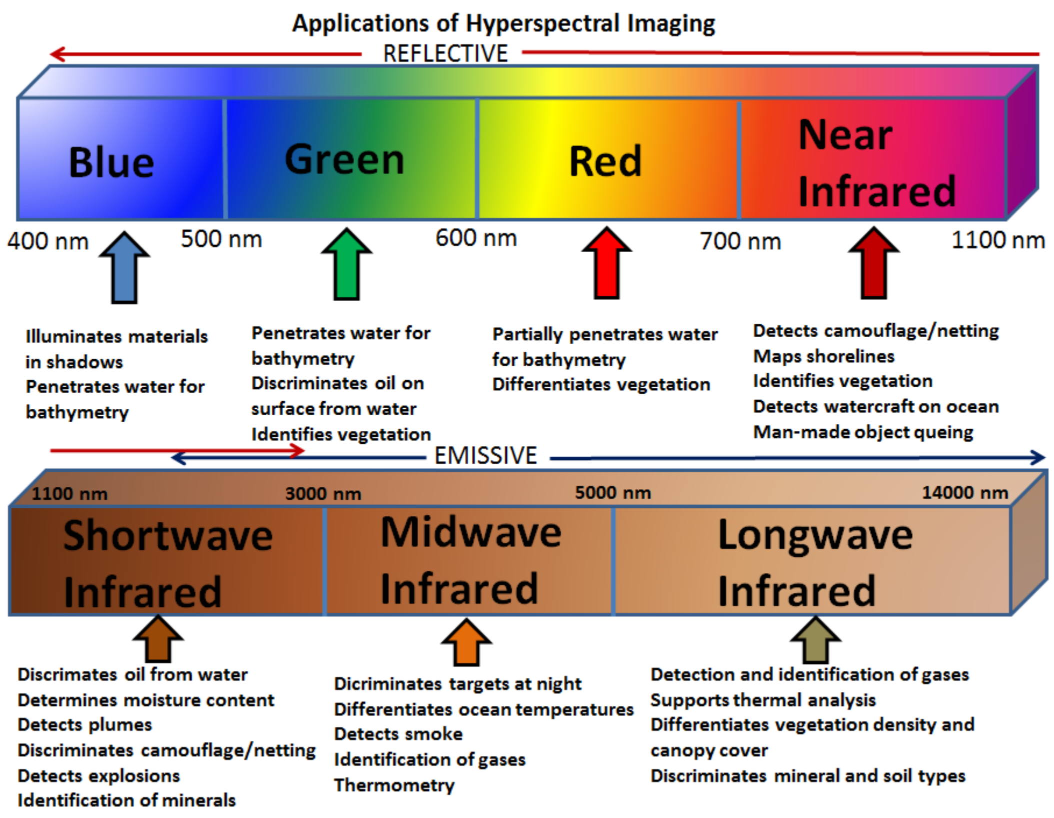

Applications of Hyperspectral Imaging
REFLECTIVE
Blue
Green
Red
Near Infrared
400 nm
500 nm
600 nm
700 nm
1100 nm
Illuminates materials in shadows
Penetrates water for bathymetry
Penetrates water for bathymetry
Discriminates oil on surface from water
Identifies vegetation
Partially penetrates water for bathymetry
Differentiates vegetation
Detects camouflage/netting
Maps shorelines
Identifies vegetation
Detects watercraft on ocean
Man-made object queing
EMISSIVE
1100 nm
3000 nm
5000 nm
14000 nm
Shortwave Infrared
Midwave Infrared
Longwave Infrared
Discrimates oil from water
Determines moisture content
Detects plumes
Discriminates camouflage/netting
Detects explosions
Identification of minerals
Dicriminates targets at night
Differentiates ocean temperatures
Detects smoke
Identification of gases
Thermometry
Detection and identification of gases
Supports thermal analysis
Differentiates vegetation density and canopy cover
Discriminates mineral and soil types

Figure 2.2 Electromagnetic spectrum and applications of hyperspectral imaging

Closing this brief introduction we should mention the technology's primary disadvantages which are the increased cost of the imaging equipment and the increased complexity of the system as well as the necessary data processing. An increased detail in the acquired images, which often exceed hundreds of megabytes in size, is tied together with a corresponding increase in needed data storage capacity and processing capabilities. Providing solutions to these challenges will allow researchers to further realize the full potential of hyperspectral imaging in the next years.

2.2 Spectral Signatures

The spectrum of each pixel is the plot of the brightness values (radiance or reflectance) versus the wavelength. The spectrum can then be used to identify and characterize a particular feature within the scene based on unique spectral characteristics of each element. Each material in an image has thus a unique spectral "fingerprint" or signature which can be used to identify similar materials based on this signature.

The fundamental property we obtain in a spectral signature is the *spectral reflectance*. Reflectance varies with wavelength for most materials because energy at certain wavelengths is scattered or absorbed to different degrees. These unique characteristics of the reflectance curve can be then used to identify and discriminate different materials.

Figure 2.3 depicts a number of different signatures for various earth surface materials obtained from the APEX dataset. The differences in the curves for each material are clearly visible.

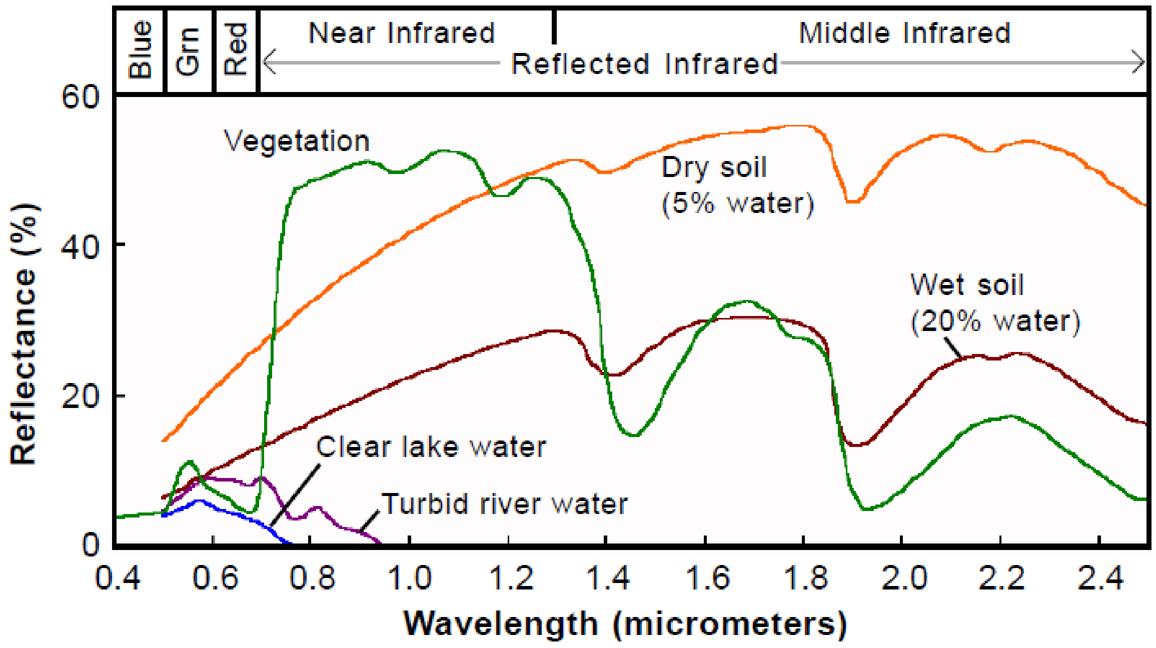


Figure 2.3 Spectral reflectance curves for different materials

2.3 Factors that affect quality results

The image quality of a scene captured by a hyperspectral imaging sensor is affected by a number of factors related to sensor properties, atmospheric effects and illumination differences.
We already mentioned in the introduction of the current chapter that pixel size is a factor that affects the spectral results in the sense that for a large cell, more than one material might contribute to the measured spectrum. Variations between detectors also sometimes alter raw measurements that have to be scaled in order to compensate for these differentiations.
From what we have talked about so far, surface reflectance of the surface materials is only one of the factors affecting measured values. Hyperspectral remote sensing systems use airborne or satellite mounted equipment that captures images from an altitude that reaches up to hundreds of kilometers. Thus, spectral radiance measured by a remote sensor depends on the spectrum of the solar energy and its interaction during its passages through the atmosphere, illumination geometry of certain areas, shadowing and

atmospheric absorption. The aforementioned factors have to be taken into account in order to correct measured values. [3]
It is clear that identification of spectral signatures for each material is not easy since we have to take into account the factors we discussed in this section. Several methods are available to match image spectrums as we will see in the next section.

2.4 Difference-based Spectrum Matching and Metrics

In order to identify features in a hyperspectral image scene we try to match each image spectrum individually to one of the known spectral "fingerprints" in a signature library. A captured spectrum typically shows different reflectance values which match in different extents to more than one reference spectra. The matching reference spectra must then be ranked with some kind of goodness fit in order to decide for the "best" one. We match each pixel of an image with the above mentioned method assigning every one of them to one of the reference signatures. The resulting map can then be used along with other criteria to identify material, objects and other features in an image.

We will discuss two common methods used to match image spectra. These methods rely on two main characteristics of reflectance spectra, position a general shape of a spectrum is one and steep slopes and absorption features is the other. In our thesis we will focus on matching the continuum characteristics of the spectra.

In order to achieve this we will examine three matching metrics: The *sum of absolute differences (SAD)*, the *Euclidean* metric and the *chi-squared (x^2)* metric. In reality because we are only interested in the relative distance of two vectors and not their actual distance, we will not calculate the square root of the Euclidean distance, a decision that has an evident impact on hardware resources as we will see in Chapter 5.

2.4.1 Sum of Absolute Differences

The sum of absolute differences is the simplest distance metric we are going to examine. It is also one of the most commonly used metric for matching applications. The SAD adds up the absolute differences between corresponding elements in the candidate and reference vector.

$$SAD = \sum_{i=1}^{N} |x_i - y_i|$$

where $x_{i,j}$ are the elements of the reference vector and $y_{i,j}$ the elements of the candidate vector. Thus, the computation of the SAD is divided in three steps: Computation of differences between corresponding elements, determination of the absolute value of each differences and addition of these absolute values.

Comparing this method with both the Euclidean Distance Metric and the Euclidean squared distance metric described before, the higher computational complexity of the former two is quite obvious as they involve numerous multiplication operations.

2.4.2 Euclidean Distance

This metric is an immediate consequence of the Pythagorean Theorem which if applied to distances in two-dimensional space vectors proves that the squared distance between two vectors $x = [x_1\, x_2]$ and $y = [y_1\, y_2]$ is the sum of squared differences in their coordinates. If we denote this distance as d_{xy} , the distance itself is the square root of this result.

$$\sqrt{\sum_{i=1}^{N} \left(x_i - y_i\right)^2}$$

2.4.3 Chi-Squared Distance

The chi-squared distance is a metric, similar to the Euclidean squared distance with the difference that it normalizes the data before accumulating the result. This has a major impact in the quality of the overall result since it normalizes the differences in spectral curves that have an offset due to external factors as mentioned above. It also introduces a high cost in hardware resources for implementing the division operation as we will discuss in Chapter 5. The mathematical expression for this metric is the following. An alternative to this expression is shown in equation a.2 where we divide by $\frac{x_i + y_i}{2}$.

$$\sum_{i=1}^{N} \frac{\left(x_i - y_i\right)^2}{(x_i + y_i)} \quad (a.1)$$

$$\sum_{i=1}^{N} \frac{\left(x_i - y_i\right)^2}{(x_i + y_i)/2} \quad (a.2)$$

2.5 Computation Requirements and Applications

The ability to capture a large amount of information from a single scene has established hyperspectral imaging of great importance for numerous applications ranging from satellite based/airborne remote sensing and military target detection to industrial quality control, quality assessment and food inspection as well as lab applications in medicine and biophysics.

It is expected in future years that hyperspectral sensors will increase in resolution resulting in an equally big increase of the corresponding data. Spectral channels will increase into the thousands offering a wealth of information that opens a many more opportunities for hyperspectral imaging in several fields such as environmental modeling and real time atmospheric studies, hazard prevention and wildfire tracking, biological and environmental threat detection or security and defense purposes. [6]-[8]

As we can see from the above analysis, one big drawback of HSI systems and a reason why they haven't been widely used for many years is the enormous data volume that leads to the requirements of increased processing power, which often cannot be handled by common processing units. Also speed requirements in industrial production, quality inspection and real time monitoring are greatly limited by slow computational performance and/or very high implementation cost.

In order to speed-up computations, in particular in real-time computation scenarios where fast data processing is essential, hardware accelerators are required which can provide very high performance and constitute an excellent solution for the fraction of the cost and size of alternatives like or cluster- or network-based parallel processing units [9]. We will discuss in more depth hardware accelerators and their advantages in the following chapter.

2.6 Examined Hyperspectral Datasets and Hypothetical Camera Configurations

For the purposes of the current thesis, we examined two hyperspectral datasets. One main dataset obtained from an online source and taken with a camera with hundreds of channels, and one small dataset obtained in the NTUA with small hyperspectral cameras.
Furthermore, in order to simulate different conditions and configurations, we examined a number of camera configurations which allowed us to draw results for a number of different factors that affect the final quality results and are important for optimal system design

2.6.1 Hyperspectral Datasets

Our main dataset was acquired online and is available for free from APEX (Airborne Prism Experiment), an imaging spectrometer developed on behalf of ESA [1]. The acquired image has been taken on a clear day, from a research aircraft in the vicinity of Baden, Switzerland. A true color version of the Data Set can be seen in Figure 2.4 and technical key facts are listed in Table 2.1. The spectral resolution of the camera is 285 spectral channels covering a range of 413nm – 2421nm.

Date	2011-06-26
Altitude	4600 m asl
Dimensions	1500 x 1000 pixels
Spectral coverage	413nm – 2421 nm
Spectral channels	285
Pixel size	1.8 m
Land covers	Forest, Urban, Freshwater, Agriculture
Units	Reflectance (HCRF)
File size	815 MB

Table 2.1 APEX Open Science Data Set key facts

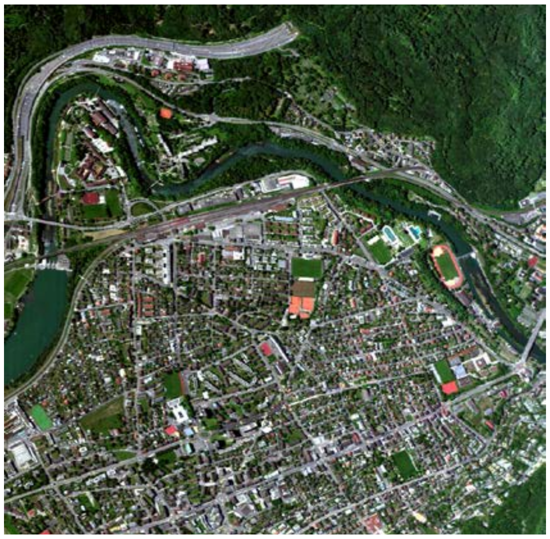


Figure 2.4 Open Science Data Set – True Color

The second data set we used in our analysis was obtained with the collaboration of the remote sensing laboratory of the NTUA using two hyperspectral cameras with 16 and 25 channels respectively. Key technical details about the cameras are listed on Table 2.2. The scene we captured shows a table with diverse objects of different material shape and color. A true color image of the scene can be seen in Figure 2.5

Sensors	Hyperspectral Mosaic Snapshot Imager
	Camera 1 / Camera 2
Dimensions	272 x 512 pixels / 220 x 400 pixels
Spectral coverage	470nm – 620nm / 650nm – 1000nm
Spectral channels	16 / 25
Bit depth	10 bit
Land covers	Vegetation, Metal, Wool, Plastic, Ceramic, Glass

Table 2.2 NTUA Remote sensing lab Data Set key facts

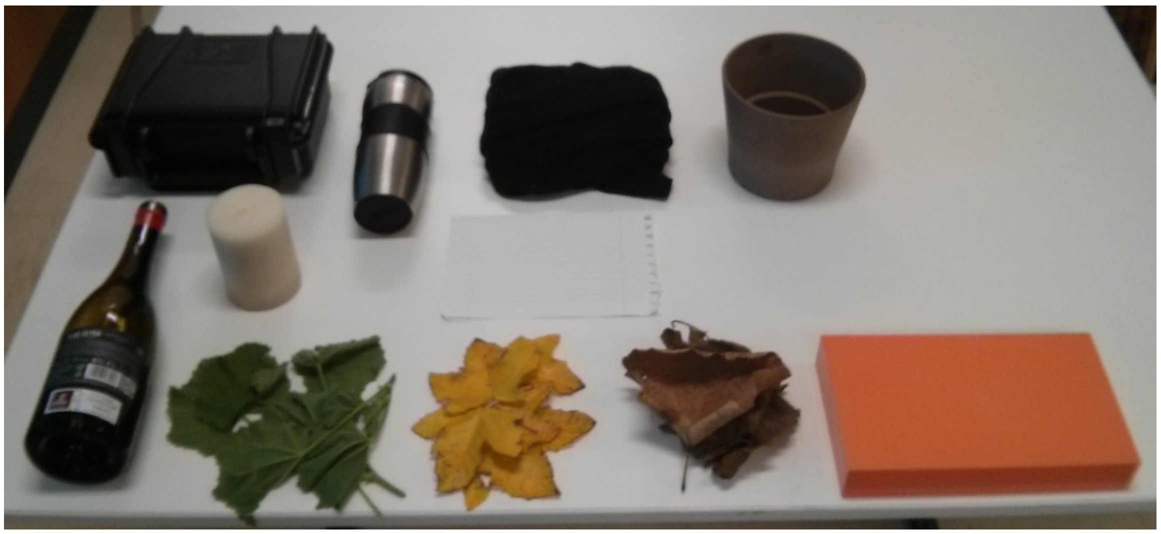


Figure 2.5 NTUA Remote sensing lab Data Set

2.6.2 Hypothetical Cameras

In order to examine the importance and the effects of different spectral channels on the quality results of the matching process, we looked into four commonly used camera configurations with distinctive characteristics. For all configurations we approximated the different camera configurations utilizing only part of the 285 spectral channels available in the APEX image. The four different camera configurations tested are:

	HYPERSPECTRAL CAMERA CONFIGURATIONS				
	FULL RANGE	NEAR INFRARED	SHORT-WAVE INFRARED	COMPACT	MULTI SPECTRAL
CHANNEL NUMBER	256	~ 40	~ 50	~ 80	~ 25
CHANNEL RANGE	1 - 256	1 - 110	110 - 256	1 - 256	1 - 256
STEP	1	3	3	3	10 ch. Average
SPECTRUM (nm)	400 - 2400	400 – 1100	1100 - 2400	400 - 2400	400 - 2400

Table 2.3 Camera Configurations

For all cameras except the full range camera and the multi spectral camera, we acquired the reduced channel number by simple sub-sampling of all channels. The step value in the above table stands for the channel sub-sampling step. In the case of the multi spectral camera we simulate the wide spectra of each channel by calculating 10 channel average values for all channels resulting in approximately 25 wide-spectrum channels.

CHAPTER 3

ACCELERATION WITH FIELD PROGRAMMABLE GATE ARRAYS (FPGA)

3.1 Introduction

Modern systems often rely on embedded systems and micro-processors in a wide range of applications varying from small common systems to large highly complicated systems with high processing requirements. The processing burden in these systems is taken by processors that range from general purpose processors to highly specialized and application specific processing units. In recent years, the requirements for certain applications demand the use of units with high processing power, low power and small size. Furthermore, the need to easily alter a device's functionality has led to the development of reconfigurable devices that can quickly be programmed and reprogrammed when needed even after they have been installed in a device for the first time.

In the current thesis, we focus on the use of a specific reprogrammable device, the "Field Programmable Gate Array" (FPGA) circuit. This reconfigurable circuit, which will be discussed in detail in the following sections, is a device that is becoming more and more popular in a wide range of applications.

3.2 Field Programmable Gate Arrays

Field Programmable Gate Arrays (FPGAs) are semiconductor devices that are based around a matrix of configurable logic blocks which are connected via programmable interconnects. In contrast to common processors that can be found in a PC, programming an FPGA rewires the chip itself to implement a desired functionality rather than to run a software application. FPGAs can be reprogrammed after manufacturing. This feature distinguishes FPGAs from Application Specific Integrates Circuits (ASICs) which are custom manufactured for specific design tasks [21] [22]. This programmability gives the user access to complex integrated designs without the high engineering costs associated with application specific integrated circuits.

In recent years, as process geometries have shrunk into the deep-submicron region, to logic capacity of FPGAs has greatly increased making these devices a viable and preferred in many cases alternative for large designs [23]. With unprecedented logic density increased and a host of other features, such as embedded processors, DSP blocks, clocking and high-speed serial at ever lower price points, modern FPGA are an ideal fit for many different markets [21].

3.3 FPGA Architecture

FPGAs are pre-fabricated circuits that consist of three fundamental components. These three components are logic blocks, I/O blocks and programmable routing channels. The logic resources are used to implement the digital logic of the circuit while routing channels are used to connect the logic blocks to form a larger circuit. Most FPGAs also include a number of embedded hard blocks that perform certain tasks. An abstract design of a generic FPGA can be seen in Figure 2.1

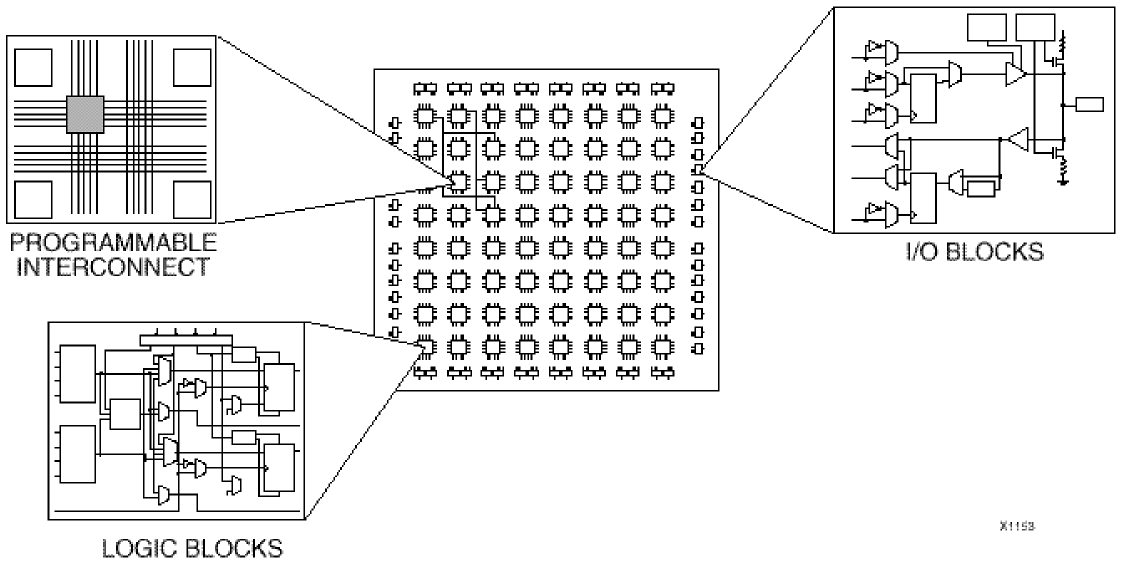


Figure 3.1 Typical FPGA architecture

Logic Blocks

Configurable logic blocks (CLBs) are identical and organized in the FPGA in a two dimensional array connected through the aforementioned programmable routing channels. In general, a typical CLB consists of a number of logical cells, organized in slices. In each logical cell, a Look-Up Table (LUT), a full adder and a d-type flip-flop can be found.
An N-input look-up table (LUT) is a functional unit designed to compute any function of N-inputs. Its operational principle is similar to that of truth table of a logical function. Thus, the truth table of a function can be used to program a look-up table accordingly. This means that according to the pattern of the N inputs, the table chooses the correct row and generates the output value. N-input LUTs can be used to perform a number of functions and can be combined to implement more complex functions. For example, besides basic functions, LUTs can implement an N-bit shift register or can be used as a distributed memory module of N-bits.

Hard Blocks

Modern FPGAs, apart from the above components, include additional blocks fixed into the silicon providing additional functionality. A number of common functions are embedded into the silicon reducing the required area for implementation and increasing the speed of these functions. Examples of such functions include multipliers, generic DSP blocks as well as embedded processors and embedded memory modules. A DPS block is a combination of basic arithmetic modules, that is, adders, subtractors and multipliers, put together to compose an arithmetic logic unit (ALU).

Interconnection

As we already mentioned, logic blocks in a typical FPGA are identical and organized in a two dimensional array island-like architecture. In between the logic blocks, a number of interconnections are configured appropriately in order to route the signals among the logic blocks. The connection of the blocks through the routing resources is performed with the use of a connection block. A connection block is located on every channel interconnection and allows, through programmable switches, logic block I/Os to be assigned to arbitrary horizontal and vertical tracks.

3.4 FPGA Programming and CAD Tools

Specific Computer-Aided Design tools are software tools necessary for the efficient programming of a specific FPGA. Implementing a circuit on a modern FPGA requires a very high number of logic blocks, embedded blocks and switches to be correctly configured. It is evident that this task is impossible to be performed by a circuit designer who would have to manually reprogram each element individually. This task can instead be completed with the description of the circuit at a higher level of abstraction, typically using a hardware description language (HDL) such as Verilog and VHDL. A more detailed discussion about HDL languages follows in section 2.4.1. An alternative method of describing a circuit at higher level is through high-level synthesis. HLS provides an even greater abstraction level of design reducing the complexity of programming a circuit. In the current work, all circuits have been implemented using the VHDL language and therefore, a greater analysis of alternative programming methods is not presented here.

Having fully described a system in an appropriate description language, the aforementioned CAD tools can then convert these high level description files in a programming bit file specifying the state of each programmable element on the FPGA board. The complete procedure of converting a circuit to a bit-stream file is broken into a series of sequential steps, which are described in Section 2.4.2.

Computer-aided design software tools also provide the necessary features in order to investigate the quality of different architectures. Once a circuit has been implemented in a specific architecture, it is essential to generate accurate area, delay and power models to evaluate the quality of the circuit implementation under test [1].

3.4.1 VHDL Hardware Description Language

VHDL (VHSIC Hardware Description Language) is a language primarily used to describe hardware and should not be considered as just another computer language. The main and most important characteristic of VHDL is that unlike higher-level computer languages that are sequential in nature, VHDL is not.
It was invented to describe hardware and in fact VHDL is a concurrent language. This means that, normally, VHDL instructions are all executed at the same time (concurrently) regardless of the size of the implementation. This inherent difference requires an alternative way of programming mentality [25].
VHDL has many features appropriate for describing (to an excruciating level of detail) the behavior of electronic components ranging from simple logic gates to complete microprocessors and custom chips. Features of VHDL allow electrical aspects of circuit behavior (such as rise and fall times of signals, delays through gates, and functional operation) to be precisely described. The resulting VHDL simulation models can then be used as building blocks in larger circuits (using schematics, block diagrams or system-level VHDL descriptions) for the purpose of simulation [26].
One of the most important (and under-utilized) aspects of VHDL is its ability to capture the performance specification for a circuit, in a form commonly referred to as a test bench. Test benches are VHDL descriptions of circuit stimulus and corresponding expected outputs that verify the behavior of a circuit over time. Test benches should be an integral part of any VHDL project, and should be created in parallel with other descriptions of the circuit [26].

3.4.2 Design Flow and Tools

In this section we briefly discuss the steps that are followed in order to map a design on an FPGA. These steps are Logic Synthesis, Technology mapping, Placement & Routing and finally, bit stream generation as shown in Figure 2.5

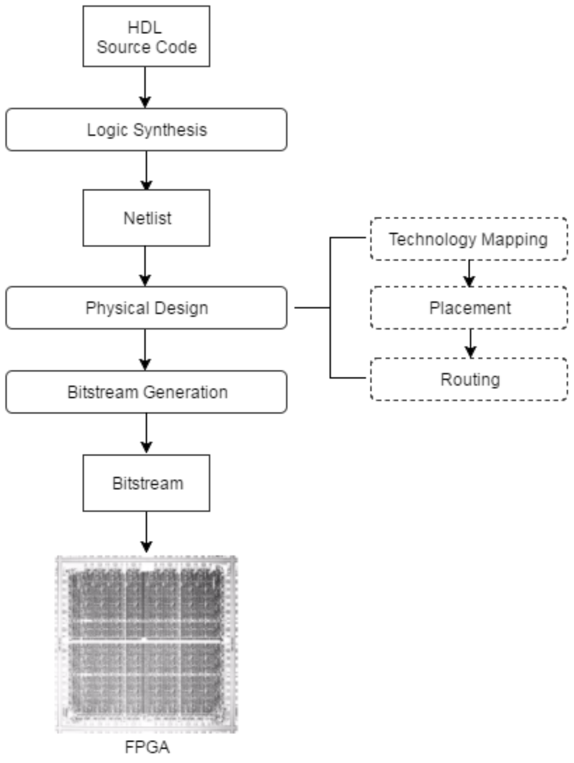


Figure 3.2 A typical FPGA design flow

The first stage of synthesis converts the circuit description from an HDL file into a netlist of basic gates. This netlist is converted then into a netlist of FPGA logic blocks according to the desired synthesis properties (speed, area or power specifications). During this stage, several optimizations take place removing redundant logic and simplifying the design.

After the synthesis stage, the implementation stage follows where the netlist is translated into a placed and routed FPGA design. During the physical design stage, in the technology mapping stage, several LUTs and registers are packed into one logic block respecting limitations imposed by the FPGA platform. In this stage, a number of optimizations are available depending on the goals the designer has chosen. Important optimizations are LUT combining in order to minimize resource utilization and minimize number of signals to be routed between logic blocks.

Once the circuit has been mapped on a specific device, the placement stage begins where heuristic placement algorithms determine which logic block within the FPGA should implement each of the logic blocks required by the circuit. The optimization goals are to place connected logic blocks close together to minimize the required wiring (wirelength-driven placement), and sometimes to place blocks to balance the wiring density across the FPGA (routability-driven placement) or to maximize circuit speed (timing-driven placement).

Now that the location for all the logic blocks in the circuit has been chosen, it is necessary to program the switches on the device to be used in order to connect all logic block input and output pins required by the circuit. In this stage, the routing architecture of the device is represented as a directed graph. Thus, routing a connection corresponds to finding a path in this routing-resource graph. Since most of the delay in FPGA designs is routing delay, a timing-driven optimization in the routing stage is crucial to minimize overall circuit delay [23].

Finally, after the design has been successfully placed and routed (PAR) on the chosen FPGA, the design tool creates a bitstream of the final design after PAR which is then downloaded to the FPGA and configures the device accordingly.

3.5 The Xilinx Zynq-7000 SoC and Xilinx Design Tools

For the purposes of this thesis, the device that was chosen for implementing our design is the Xilinx Zynq-7000 all programmable system on chip which integrates the software programmability of an ARM-based processor with the hardware programmability of an FPGA. The specific device used from the Zynq family is the Zynq Z-7045. A device overview and the specifications are presented in Figures 2.5 and 2.6. All measurements where performed on this device which utilizes a Kintex-7 FPGA while all measurements of the C code software were taken on the A9 Cortex ARM processor of the same device.

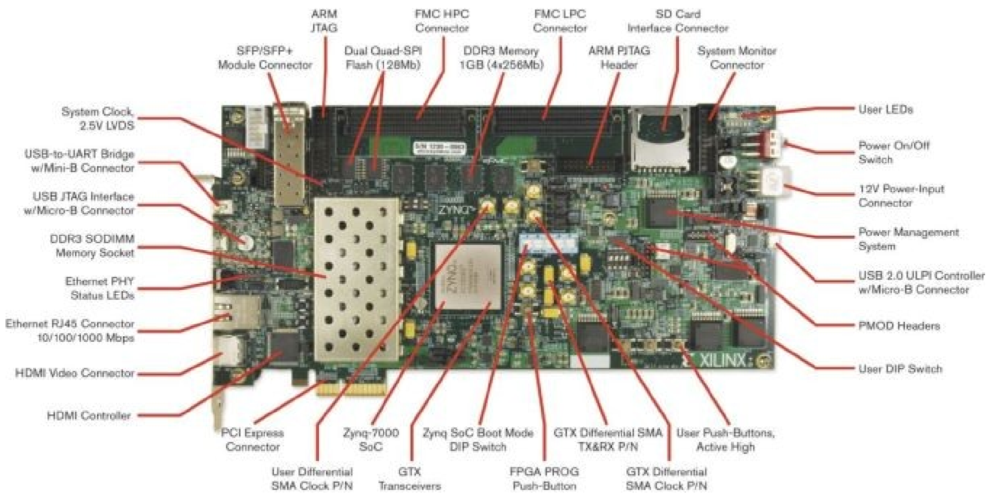
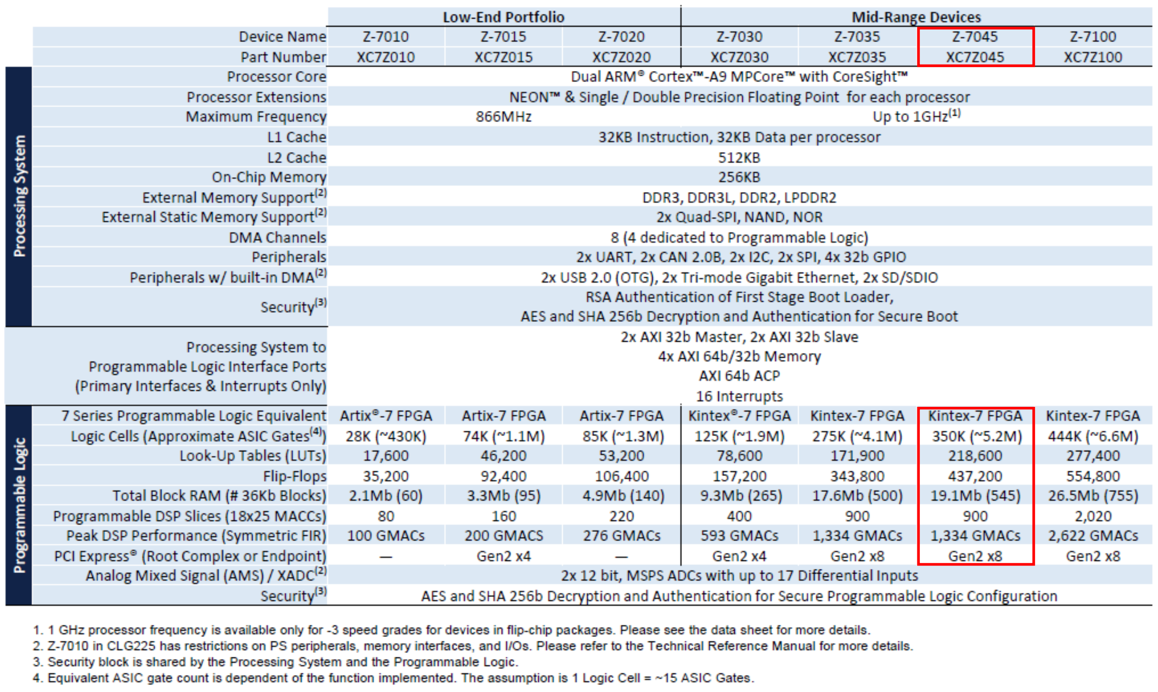

ARM JTAG
FMC HPC Connector
FMC LPC Connector
SD Card Interface Connector
SFP/SFP+ Module Connector
Dual Quad-SPI Flash (128Mb)
DDR3 Memory 1GB (4x256Mb)
ARM PJTAG Header
System Monitor Connector
System Clock, 2.5V LVDS
USB-to-UART Bridge w/Mini-B Connector
USB JTAG Interface w/Micro-B Connector
DDR3 SODIMM Memory Socket
Ethernet PHY Status LEDs
Ethernet RJ45 Connector 10/100/1000 Mbps
HDMI Video Connector
HDMI Controller
User LEDs
Power On/Off Switch
12V Power-Input Connector
Power Management System
USB 2.0 ULPI Controller w/Micro-B Connector
PMOD Headers
User DIP Switch
PCI Express Connector
Zynq-7000 SoC
Zynq SoC Boot Mode DIP Switch
GTX Differential SMA TX&RX P/N
User Push-Buttons, Active High
User Differential SMA Clock P/N
GTX Transceivers
FPGA PROG Push-Button
GTX Differential SMA Clock P/N

Figure 3.3 Xilinx Zynq-700 SoC [27]



		Low-End Portfolio			Mid-Range Devices			
	Device Name	Z-7010	Z-7015	Z-7020	Z-7030	Z-7035	Z-7045	Z-7100
	Part Number	XC7Z010	XC7Z015	XC7Z020	XC7Z030	XC7Z035	XC7Z045	XC7Z100
Processing System	Processor Core	Dual ARM® Cortex™-A9 MPCore™ with CoreSight™						
Processing System	Processor Extensions	NEON™ & Single / Double Precision Floating Point for each processor						
Processing System	Maximum Frequency	866MHz			Up to 1GHz[1]			
Processing System	L1 Cache	32KB Instruction, 32KB Data per processor						
Processing System	L2 Cache	512KB						
Processing System	On-Chip Memory	256KB						
Processing System	External Memory Support[2]	DDR3, DDR3L, DDR2, LPDDR2						
Processing System	External Static Memory Support[2]	2x Quad-SPI, NAND, NOR						
Processing System	DMA Channels	8 (4 dedicated to Programmable Logic)						
Processing System	Peripherals	2x UART, 2x CAN 2.0B, 2x I2C, 2x SPI, 4x 32b GPIO						
Processing System	Peripherals w/ built-in DMA[2]	2x USB 2.0 (OTG), 2x Tri-mode Gigabit Ethernet, 2x SD/SDIO						
Processing System	Security[3]	RSA Authentication of First Stage Boot Loader, AES and SHA 256b Decryption and Authentication for Secure Boot						
	Processing System to Programmable Logic Interface Ports (Primary Interfaces & Interrupts Only)	2x AXI 32b Master, 2x AXI 32b Slave 4x AXI 64b/32b Memory AXI 64b ACP 16 Interrupts						
Programmable Logic	7 Series Programmable Logic Equivalent	Artix®-7 FPGA	Artix-7 FPGA	Artix-7 FPGA	Kintex®-7 FPGA	Kintex-7 FPGA	Kintex-7 FPGA	Kintex-7 FPGA
Programmable Logic	Logic Cells (Approximate ASIC Gates[4])	28K (~430K)	74K (~1.1M)	85K (~1.3M)	125K (~1.9M)	275K (~4.1M)	350K (~5.2M)	444K (~6.6M)
Programmable Logic	Look-Up Tables (LUTs)	17,600	46,200	53,200	78,600	171,900	218,600	277,400
Programmable Logic	Flip-Flops	35,200	92,400	106,400	157,200	343,800	437,200	554,800
Programmable Logic	Total Block RAM (# 36Kb Blocks)	2.1Mb (60)	3.3Mb (95)	4.9Mb (140)	9.3Mb (265)	17.6Mb (500)	19.1Mb (545)	26.5Mb (755)
Programmable Logic	Programmable DSP Slices (18x25 MACCs)	80	160	220	400	900	900	2,020
Programmable Logic	Peak DSP Performance (Symmetric FIR)	100 GMACs	200 GMACS	276 GMACs	593 GMACs	1,334 GMACs	1,334 GMACs	2,622 GMACs
Programmable Logic	PCI Express® (Root Complex or Endpoint)	—	Gen2 x4	—	Gen2 x4	Gen2 x8	Gen2 x8	Gen2 x8
Programmable Logic	Analog Mixed Signal (AMS) / XADC[2]	2x 12 bit, MSPS ADCs with up to 17 Differential Inputs						
Programmable Logic	Security[3]	AES and SHA 256b Decryption and Authentication for Secure Programmable Logic Configuration						

1. 1 GHz processor frequency is available only for -3 speed grades for devices in flip-chip packages. Please see the data sheet for more details.
2. Z-7010 in CLG225 has restrictions on PS peripherals, memory interfaces, and I/Os. Please refer to the Technical Reference Manual for more details.
3. Security block is shared by the Processing System and the Programmable Logic.
4. Equivalent ASIC gate count is dependent of the function implemented. The assumption is 1 Logic Cell = ~15 ASIC Gates.

Figure 3.4 Zynq-7000 devices specifications [11]

Design Software used for mapping our design on the FPGA and thoroughly testing and certifying it, was the Xilinx ISE and the Xilinx Vivado design suites. These complete software suites, provided by Xilinx for their own FPGA boards, allow the implementation, mapping and testing of our design, according to specified requirements and goals. We performed timing analysis and resource optimizations for all components in order to achieve optimal speed and resource utilization.

3.6 Software Optimization and Measurement Theoretical Concepts

In this section we will discuss in depth about the designed system implemented on the FPGA as well as architectural decisions and characteristics. Before getting into design details, we will briefly introduce some hardware design concepts, Pipelining and Parallel Computing as well as some software definitions and code optimizations necessary to understand our implementation results.

3.6.1 Pipelining

Pipelining is a feature in most processors which is much like an assembly line in a factory. The basic idea behind pipelining is executing more than one instruction each clock cycle if these are independent allowing more instructions to be executed in a shorter period of time. Even though there are some problems associated with data dependencies and branch instructions, solutions to both of these problems are available making pipelining a highly efficient feature in digital design [13].

A pipelined approach has been adopted for our system greatly reducing critical paths in modules with many levels of logic and increasing system throughput. Pipelining of our design was achieved through the addition of multiple registers between each logic level with a long critical path. A full pipelined implementation of our design allowed us to achieve a two or three times faster clock frequency greatly improving the overall speedup.

The Xilinx synthesis and implementation tools, available in the Xilinx design suites, used many of the inserted registers effectively moving them across the whole hierarchy, thus breaking critical paths not only inside each component but also between component interconnections [14].

3.6.2 Parallel Computing

Parallel computing is the simultaneous computation of many tasks or the parallel computation of several smaller and often similar, subtasks of one broken down main task. Parallel processing can be realized in different types of parallelism on software level or hardware level [15].
As already mentioned in the previous Chapter, the FPGA is a device that can be programmed to implement very high levels of parallelism. In our design specifically, we implemented three levels of parallelism as we will see in the next section for various configurations fully utilizing the device's parallel computing potential.

3.6.3 Latency and Throughput

Latency refers to the time delay needed for an input in a system to reach its output. Latency is measured in units of time; hours, minutes, seconds, nanoseconds or clock periods. In a system with many components, latency is not the same for every component and a big delay in one part of the system can often affect the total system performance. In cases where latency is big the designer has to identify the length of the longest series of operations in a component that dominates time delay (*critical path*) and try and break down this path with the using pipelining as mentioned before [16].

Throughput on the other hand is the measure of how many units of information a system can process in a given amount of time. Throughput is an important measure when processing large amount of data as we will see in the results section of this Chapter since available bandwidth (maximum available throughput) is a limiting performance factor.
It is worth mentioning that pipelining increases throughput at the cost of latency which has to be taken into account when designing a system since increased latency could dramatically lower the systems performance in some applications.

3.6.4 Compiler Optimizations

As we will see in the results tables of our system, we achieved significant performance improvements optimizing our design using some important compiler optimizations we will briefly mention in this section.
When compiling a program, an optimization compiler gives us the option to minimize, if possible, some attributes of the executable program. Most commonly it is minimizing execution time and occupied memory. Turning on the optimization flags allows the compiler to perform optimizations based on the knowledge it has of the program. Several levels of optimization effort are available which can be enabled [17].

The second optimization we performed on our program is *loop unrolling*. Loop unrolling is an optimization that increases instruction-level parallelism by transforming a loop with N iterations into a loop with N/M iterations where every iteration in the transformed loop doesn't execute one instruction but M. This optimization improves execution time in parallel when loop instructions are independent [18].

This page is intentionally left blank.

CHAPTER 4

FPGA IMPLEMENTATION & EVALUATION

4.1 Introduction

We discuss in this chapter the designed and implemented system which matches each pixel from a given image with one of the reference spectral signatures we have stored in memory giving at its output the “winner”, namely the reference signature it matched each pixel with and the result of the distance metric between the pixel-signature pair. The metric result is useful in case we want to set a threshold for each pixel that determined when a pixel is matched with a signature or when it should be left unsigned.

In order to measure the achieved acceleration with the FPGA design, we need reference execution time measurements from other implementations. Thus, we first implemented the above system in C language and measured Execution times for different configurations on a common CPU and repeated the same measurements on an embedded processor. Then, for the same configurations we measured total execution time and speedup for our FPGA design.
The processors used to measure execution times for the C program are an *Intel® Core™ i3-3110M CPU @ 2.40GHz* [10] installed on a laptop and an embedded *Dual-core ARM® Cortex™ -A9 MPCore™ @ 666MHz*, part of the *Xilinx Zynq®-7020 All Programmable SoC* [11] and installed on the *ZedBoard*™ development kit [12].

4.2 System Description and Implementation

This Section is dedicated to the detailed description of the VHDL design implemented on the FPGA board. Each component will be discussed in depth, explaining its functionality, design choices and limitations. We begin the section with the presentation of an important concept in our design which is the implementation in VHDL of a system with adjustable parameters in order to achieve optimal performance and resource utilization of the available FPGA.

4.2.1 Parametric VHDL Design

The nature of the FPGA, as already seen in the previous Chapter, is such, that allows the implementation of highly parallelizable designs. This characteristic gives the designer the ability to adjust the system to the available hardware. In order to simplify this process and increase the portability and adjustability of a design, the VHDL code can be written in such a way that each component can be instantiated multiple times according to the designers needs. In the case of a multiple level design, a parametric design of each component allows maximum utilization of the available FPGA board.

In the case of our matching system, each component allows the designer to adjust the parallelism of the component depending on the needs. As we will see in the following subsection we have a three level design which allows us to adjust parallelism on each level individually. Furthermore, the bit length of each input in our system can be readjusted very easily in order to allow the use of the system in different applications. All computations and performance measurements in the following chapters have been conducted with a fixed word length of 10bit which was decided as a reasonable bit length for a common hyperspectral image channel.

4.2.2 System Design Description

The HSI matching system consists of several modules in a four level design with three levels of parallelism and a customizable configuration regarding parallelisms and data width. The systems functionality is the following:
Input to the system is a pixel we want to match and the output is the result of the matching metric as well as the index of the signature that was best matched.
Before describing the system architecture, let us talk about the pixel and signature format. We assume that each camera channel is a 10 bit wide positive integer. The channel width actually depends from the camera manufacturer and therefore can be easily adjusted in our design. The camera channel values are provided as positive integers and are converted into fixed point 10 bit wide integers. We examined two channel configurations, a 16 channel camera and a camera with 64 channels. All channels are converted in binary format and concatenated forming a large binary number that is then broken again into 10 bit wide numbers for processing. We will talk about the signature format later on in this section. Let us first start with the system description before moving on to that.

The top module, which can be seen in Figure 4.1, receives a pixel as an input and a signature from the FPGA Memory and feeds both of them into the “Pixel Channel” module which we will later on describe. The number of these modules is customizable and depends on the desirable signature parallelism. In this thesis, we have measured speed and utilization for 3 parallelism configurations, (8, 16 and 32 signatures). This number is of course a generic that can be set according to our needs. For a balanced comparator tree structure though, selected parallelism has to be a power of two.

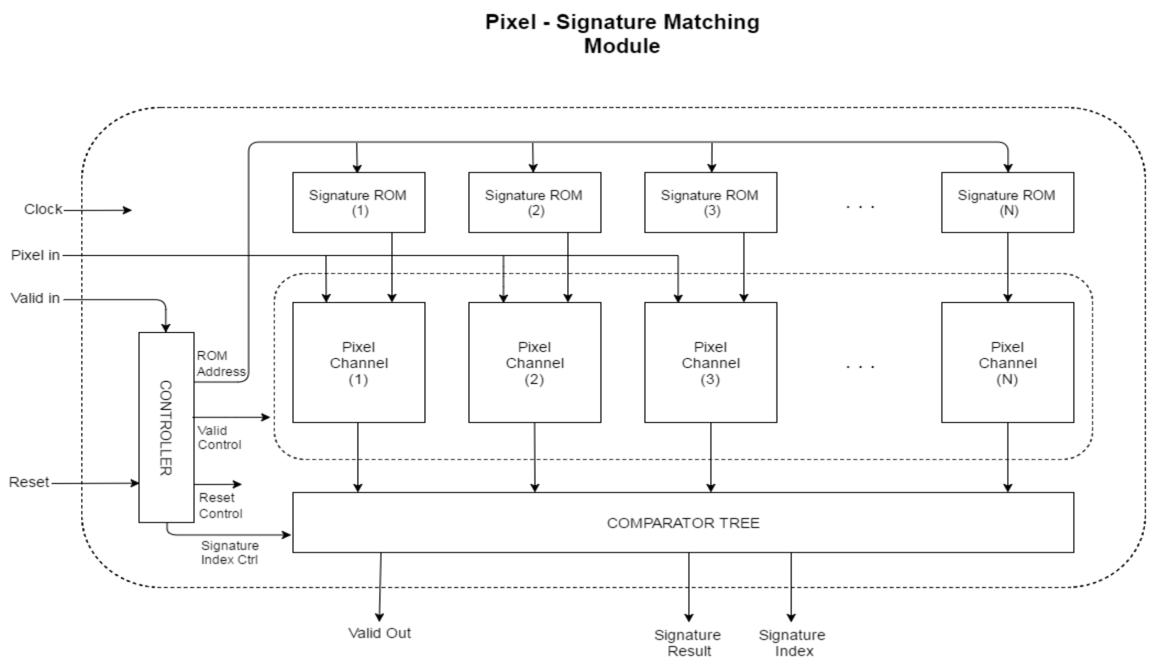


Figure 4.1 Top module for HSI pixel matching system

Each of the aforementioned matching modules does simultaneously compute the result of the matching of the input pixel with each signature it receives from the ROM and feeds its output into a comparator tree that compares all input results, giving the smallest of them at its output along with the index that indicates which signature was matched with each pixel.
It is important to mention at this point that for certain configurations the input pixels is not driven in the input in one piece but broken into smaller parts and then fed into the matching module over several clock cycles. Doing this allows us to reduce input throughput requirements by a great amount without sacrificing execution time. This is achieved by just exploiting the clock cycles needed for the computation of the final results. The following example illustrates this function.
Let's assume that we want to match a pixel with 32 signatures in a system configuration of full channel parallelism (i.e. 64 parallel channel computations for a 64 channel hyperspectral image sensor) and signature parallelism of 8. In order to match each pixel with all signatures we would need 32/8 = 4 clock cycles assuming a full pipelined architecture. As a result, the need of 4 cycles to match each pixel with all signatures means that the system receives a new input pixel every 4 cycles allowing us to "build up" the incoming pixel over these cycles. For a 640 bit pixel we can break down the I/O throughput requirements by a factor of 4. The input pixel fragments are then consecutively stored in a single register and fed at the end of the 4th cycle to the pixel matching module.
Having already described the functionality of the whole system, we will now describe each individual unit starting from the "Pixel Channel" component. The block diagram of this component is the following

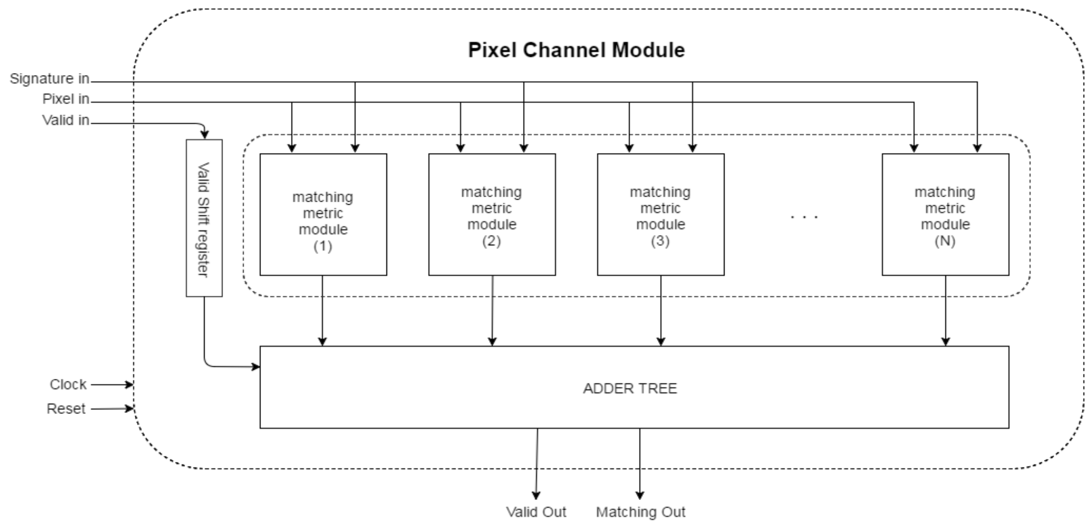


Figure 4.2 Pixel-Signature matching module

This component takes one pixel and one signature at its input and feeds them to the components that do the arithmetic calculation depending on the chosen channel parallelism. The arithmetic units are interchangeable in our design and the valid-bit register chain is adjusted accordingly. The number of arithmetic units for parallel computation is chosen by the designer depending on his needs and available hardware. Each unit does act on one channel each time. This means that maximum parallelism can theoretically be limited only by the total number of spectral channels. In practice though, we have implemented our design for up to 64 channel parallelism. A design with 256 channels is also available in which we compute 64 channels at a time and accumulate the results over 4 cycles at the end of the adder tree. A higher channel parallelism has not been implemented due to high device utilization. Under certain circumstances we could implement 128 simultaneous channel calculations but device routing delay would significantly reduce performance. Hence, it was considered redundant.
The adder tree now, takes all channel outputs and adds them giving at the end the total sum for each pixel-signature pair. If the chosen channel parallelism is smaller than the total number of channels, an accumulator at the exit of the adder tree accumulates the intermediate results. We have to note that the adder tree has to be balanced, for design simplicity, which in turn means that the chosen parallelism has to be a power of 2.
Now, we present the arithmetic units we implemented for our design. We have three different components for our three chosen distance metrics we discussed in section 2.4. The block diagram for each of the aforementioned components can be seen in Figure 4.3.

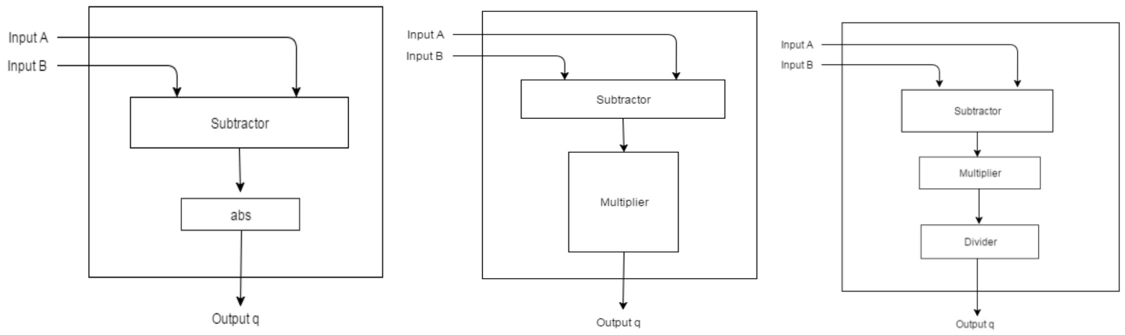


Figure 4.3 Distance metrics: **a**. Sum of Absolute Differences, **b**. Euclidean, **c**. Chi-squared

We will shortly describe at this point how each of the above modules is implemented on the FPGA and what resources are needed for optimal mapping. This will help us later on in the discussion of the total resources utilization as well as in the tradeoff analysis in Chapter 6.

- The SAD component performs two actions on the incoming data. One subtraction and an absolute value calculation. Even though the subtractor could be implemented in a dedicated DSP block, the simplicity of this operation is such that the Xilinx design tool synthesizes the above component using only lookup tables for faster performance. The very low resource requirements of the sum of absolute difference metric is its main and very important advantage as we will see later on in this chapter and also in Chapter 6.

- The difference in the Euclidean distance metric is the fact that after subtracting each value, a multiplication takes place on the subtracted values, as we already discussed in Chapter 2. The subtraction is again implemented using lookup tables but the multiplication now is implemented in the DSP blocks. This fact makes the Euclidean distance metric significantly pricier (in terms of resources), especially since the DSP blocks are limited and could be needed for other operations in a larger design. Thus, we should take into account the aforementioned resource requirements when conducting the tradeoff analysis for results quality and utilization.

- The last metric we implemented is the Chi-squared distance metric. As you can see on Figure 4.3, this one is also the most expensive regarding device resources. Again as before, this metric is implemented using lookup tables for the subtraction and DSP blocks for the multiplication.
  The challenge here is to implement the division operation. Division in hardware is a very hard to implement and costly operation [ref needed]. In our case we created the divider we needed using the Xilinx Core Generator for specific inputs and outputs. The generated core was then integrated in the rest of our design to form

the chi-squared distance metric component. The generated divider was synthesized and implemented using lookup tables and registers. Later in this chapter you can see the significant increase in resource utilization from the divider. The high resources requirements are a major limitation in the implementation of large designs which is also discussed in detail later on in this chapter. The advantage though of this metric can be significant in certain cases as we will see when we conduct our qualitative analysis in Chapter 5.

Having described the matching modules of our system, we continue with the description of the adder tree which is needed for the summation of all channels. The diagram of the adder tree is shown in Figure 4.4. Each adder in this structure is a separate adder module which can be seen in Figure 4.5. The adder tree in our structure is balanced (power of 2) for design simplicity. First level adder parallelism depends on channel parallelism. In our design we have two configurations; 16 and 64 channels, each fully parallel which means we have two adder tree configurations, one for 16 inputs and one 64 inputs.

If channel parallelism is smaller than the total channel number, the intermediate results at the end of the adder tree have to me accumulated. For this reason, an accumulator can be seen at the end of the adder tree structure. This component has been implemented in a test module but cannot be seen in the two designs that were implemented for my thesis due to full channel parallelism configurations.

Furthermore, on each level of the tree, intermediate results are 1 bit wider that their respective inputs due to potential overflow consideration. This means that for a 64 input adder tree, we have a 6 level adder tree structure which in turn means that the total output will be bigger than the inputs by 6 bits. This is the worst case scenario and is redundant in most cases. For optimal resource utilization a quality analysis has to be performed depending on the available data and the application specifications. For this reason, in our design, each adder module has a variable data size that can be adapted very easily to the designer’s requirements.

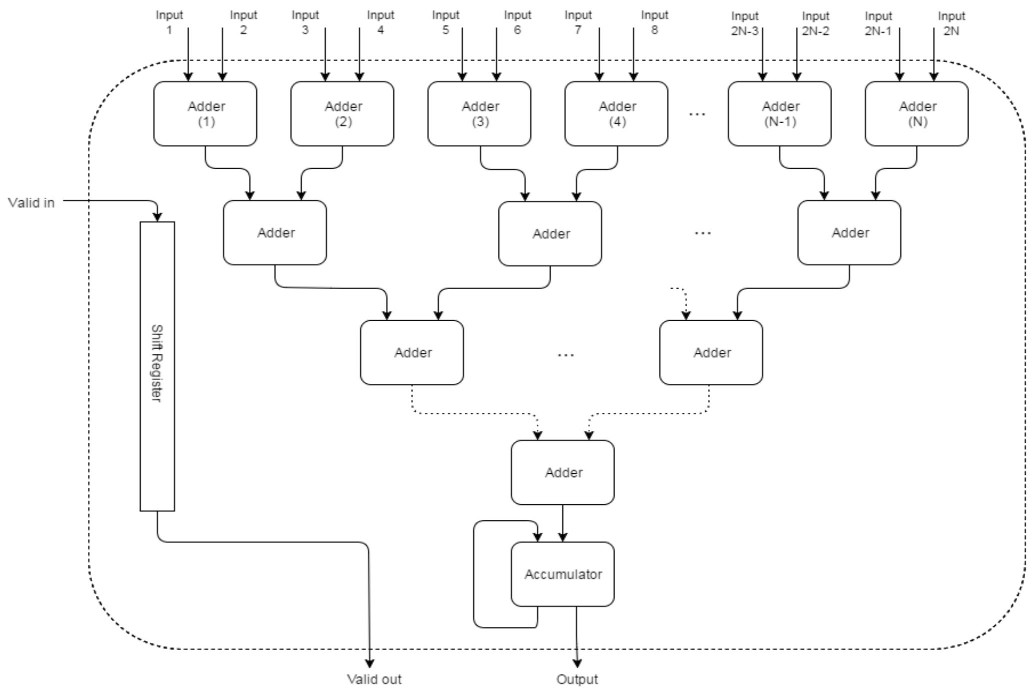


Figure 4.4 Balanced adder tree and accumulator

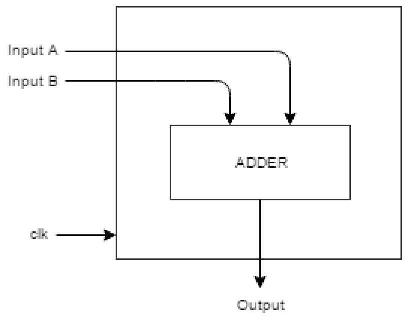


Figure 4.5 Adder module

At this point we have completely described the metric computation component for each pixel/channel pair.

Now we move on to describing the comparator structure needed to compare the results for all signatures in order to receive the final result. As we have already talked about, our final result is the result of the best signature match as well as the index of the signature.

The needed comparator component has been designed and implemented in a balanced tree structure. The full comparator tree structure can be seen in Figure 4.6.

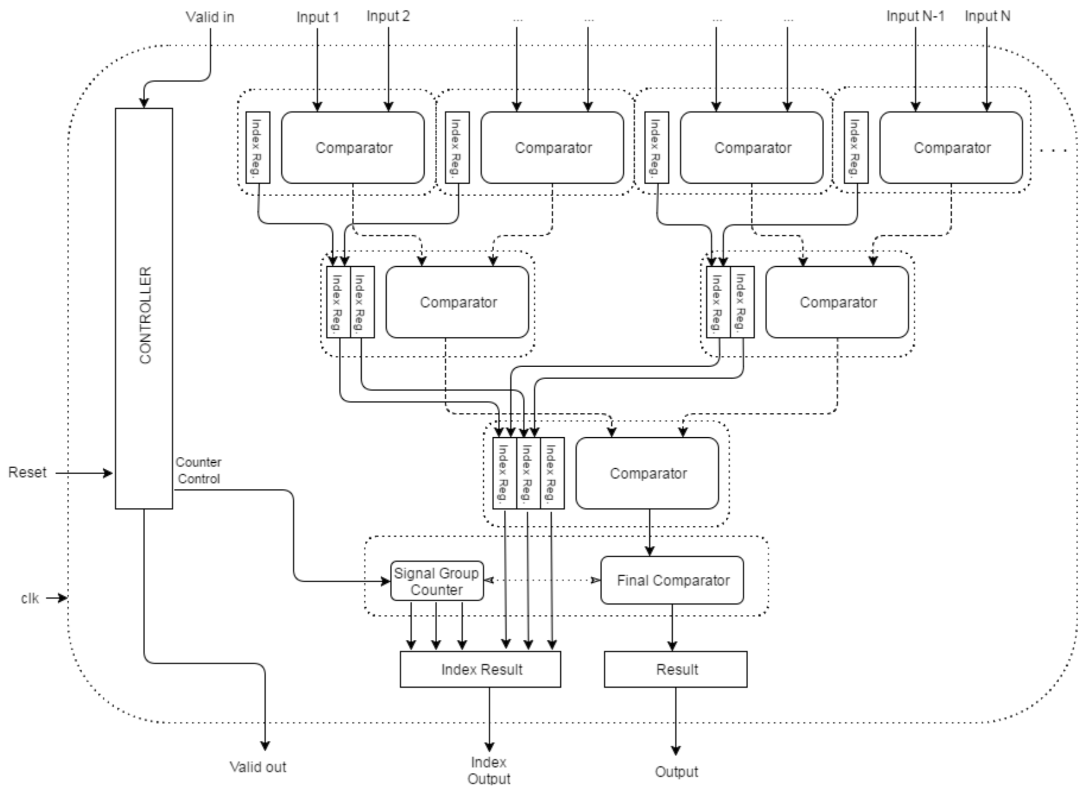


Figure 4.6 Comparator tree structure with result index output

The comparator tree is very similar in its main structure to the adder tree. Again each comparator is implemented as a separate component that is generated on each level. The bit width of the input and output results are in this case the same though since no arithmetical operation is being performed on the inputs and an overflow bit consideration is not needed.

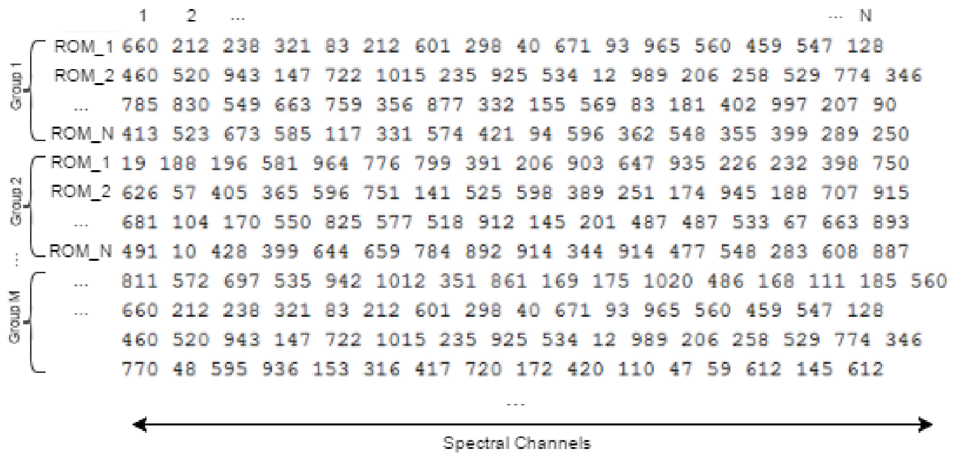
It is of big importance to mention two major points here that make this component more complicated than the previous tree structure.
The first one derives from the total number of signatures to be processed and the chosen signature parallelism. For example, in our design we examined three different signature-number / signature-parallelism configurations. One for 32 signatures and 8 signature parallelism, one for 64 signatures and 16 signature parallelism and finally, a 128 signatures - 32 signature parallelism configuration. In all three cases the number of signatures is bigger than the chosen parallelism. In our case the signature number/parallelism ratio is constant even though in the general case the design does not limit us in this. The reason that drove us to that design decision is the fact that the

signature parallelism is a limiting factor due to high resource utilization of each instance. The total signature number could be arbitrarily large, depending on the application.

As a result, at the end of the comparator tree we have an extra comparator level which keeps the result of each group. A counter controls the number of comparisons needed depending on the total signature number. In the case of our three configurations, we need four comparisons for all signatures. This architecture decision has an impact on the next point that follows now.

The second important point of our comparator structure is the ability to save the index of the best matching signature. This is essential in order to be able to identify and categorize each pixel. Let’s assume we have 32 signatures and a signature parallelism equal to 8. These 32 signatures are stored in the FPGAs memory in dedicated memory blocks or lookup tables [14].

It is important at this point to describe the order in which we store the signatures in the device memory. The signatures are loaded into the FPGA memory from an external file in which all signatures are listed in column order. A section of this file layout can be seen in Figure 4.7



```
        1   2   ...                                              ... N
ROM_1 660 212 238 321 83 212 601 298 40 671 93 965 560 459 547 128
ROM_2 460 520 943 147 722 1015 235 925 534 12 989 206 258 529 774 346
 ...  785 830 549 663 759 356 877 332 155 569 83 181 402 997 207 90
ROM_N 413 523 673 585 117 331 574 421 94 596 362 548 355 399 289 250
ROM_1 19 188 196 581 964 776 799 391 206 903 647 935 226 232 398 750
ROM_2 626 57 405 365 596 751 141 525 598 389 251 174 945 188 707 915
 ...  681 104 170 550 825 577 518 912 145 201 487 487 533 67 663 893
ROM_N 491 10 428 399 644 659 784 892 914 344 914 477 548 283 608 887
 ...  811 572 697 535 942 1012 351 861 169 175 1020 486 168 111 185 560
 ...  660 212 238 321 83 212 601 298 40 671 93 965 560 459 547 128
      460 520 943 147 722 1015 235 925 534 12 989 206 258 529 774 346
      770 48 595 936 153 316 417 720 172 420 110 47 59 612 145 612
                              ...
```

Figure 4.7 Signature file layout and memory storage order

The above signature grouping order into the memory is necessary for signature tracking. As we already mentioned, since signature parallelism is smaller than the total signature number, several iterations are needed for all signature processing. One group is processed in each iteration. Each group consists of that many signatures, as the chosen parallelism. The total number of groups equals to the ratio of total signatures to signature parallelism. The final index result consists of two parts, one for the group in which the best result is located and one for the best signature in each group.

The group index part is a simple counter that saves its value for every new best result. The spatial indexing process is a more complicated and its logical architecture can be seen in Figure 4.8

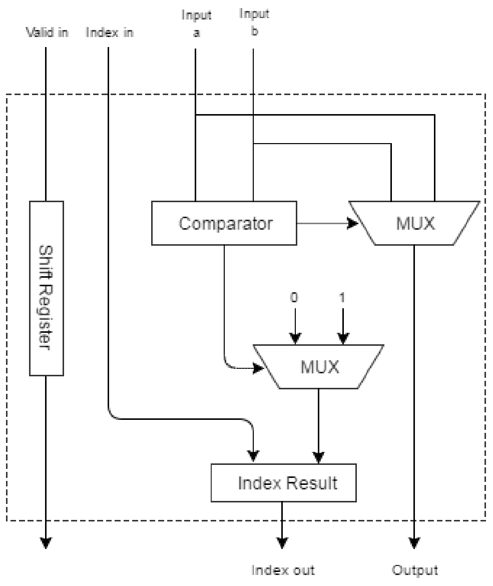


Figure 4.8 Comparator module

A comparator's output is used to control two multiplexers. One to chose the best input (lower or higher) and one to chose the right index. '1' if input A is lower or '0' if not. The input index is the index from the previous level of the tree structure. At the end of the tree structure, the final index register contains the path to the input where the best signature was located.

4.3 Execution and Utilization

In this section, we will present and discuss the complete results of all aforementioned implementations. We will show execution times, device utilization and throughput requirements for sever configurations. We will also present the results of the software implementations of the C program implementing the same configurations as on hardware. For each case we calculated hardware acceleration against both the conventional processor and the embedded processor.
Specifically, we implemented and measured the results for 6 configurations which are listed in Table 4.1. We also obtained results for a 256 channel configuration and 64 channel parallelism through extrapolation. This case implementation was deemed unnecessary due to the fact that it was the same as configurations 4 – 6.

Configurations	Signatures	Signature Parallelism	Channels (full parallelism)	Implemented metrics
1	32	8	16	3
2			64	3
3	64	16	16	3
4			64	2
5	128	32	16	2
6			64	1

Table 4.1 Implemented Configurations

4.3.1 Implementation Results

In the following tables aforementioned implementation results for the C program as well as the VHDL design, for each of the three matching metrics. We will comment the on the results after presenting the results.
Execution time in the following cases was always measured for matching 1 million pixels (1000×1000 image size) with known signature number

SAD		DEVICE UTILIZATION (ZYNQ 7045)			
		Total Signatures / Sig. parallelism	**32 / 8**	**64 / 16**	**128 / 32**
DEVICE RESOURCES	**CHANNELS 16**	**REGISTERS**	9.5K (2%)	23.8K (5%)	48.4K (11%)
		LUTs	7.6K (3%)	16.4K (7%)	33.3K (15%)
		DSPs	0	0	0
	CHANNELS 64	**REGISTERS**	50.9K (11%)	109.9K (25%)	195K (44%)
		LUTs	39.5K (18%)	84.1K (38%)	128.5K (58%)
		DSPs	0	0	0

Table 4.2 Sum of absolute differences device utilization

EUCLIDEAN		DEVICE UTILIZATION (ZYNQ 7045)			
		Total Signatures / Sig. parallelism	32 / 8	64 / 16	128 / 32
DEVICE RESOURCES	CHANNELS 16	REGISTERS	12.3K (2%)	24.7K (5%)	50.3K (11%)
		LUTs	7.9K (3%)	17.6K (8%)	29.3K (13%)
		DSPs	128 (14%)	256 (28%)	512 (56%)
	CHANNELS 64	REGISTERS	43.6K (9%)	149.6K (34%)	exceeds device resources
		LUTs	30.7K (14%)	150.6K (68%)	
		DSPs	512 (56%)	496 (55%)	

Table 4.3 Euclidean device utilization

CHI-SQUARED		DEVICE UTILIZATION (ZYNQ 7045)			
		Total Signatures / Sig. parallelism	32 / 8	64 / 16	128 / 32
DEVICE RESOURCES	CHANNELS 16	REGISTERS	141.2K (32%)	281K (64%)	exceeds device resources
		LUTs	40.9K (19%)	81.7K (37%)	
		DSPs	128 (14%)	256 (28%)	
	CHANNELS 64	REGISTERS	281K (64%)	exceeds device resources	exceeds device resources
		LUTs	81.7K (37%)		
		DSPs	256 (28%)		

Table 4.4 Chi-squared device utilization

Sum of Absolute Differences					
Channels/ parallelism	**Execution time**				
	Software (C) - (Intel Core i3)		Software (C) - (ARM Cortex - A9)		VHDL
	gcc unoptimized	gcc -O3 optimization	gcc unoptimized	gcc –O3 optimization	
(32 signatures and 8 signature parallelism in VHDL)					
256 / 64	38.74 sec	3.23 sec	-	-	44.96 ms
64 / 64	10.24 sec	0.52 sec	92.62 sec	26.72 sec	11.24 ms (356 MHz)
16 / 16	2.74 sec	0.37 sec	24.44 sec	3.5 sec	9.35 ms (428 MHz)
(**64 signatures and 16 signature parallelism in VHDL**)					
256 / 64	78.05 sec	6.99 sec	-	-	55.76 ms
64 / 64	20.95 sec	1.03 sec	176.11 sec	40.36 sec	13.94 ms (287 MHz)
16 / 16	5.30 sec	0.75 sec	50.49 sec	6.73 sec	10.75 ms (372 MHz)
(128 signatures and 32 signature parallelism in VHDL)					
256 / 64	155.15 sec	13.17 sec	-	-	99.4 ms
64 / 64	39.89 sec	2.02 sec	349.22 sec	83.07 sec	24.85 ms (161 MHz)
16 / 16	10.45 sec	1.45 sec	100.08 sec	12.87 sec	11.33 ms (353 MHz)

Table 4.5 Sum of absolute differences timing results

Euclidean									
Channels/ parallelism									
	Software (C) - (Intel Core i3)				Software (C) - (ARM Cortex - A9)				VHDL
	gcc unoptimized		gcc - O3 optimization		gcc unoptimized		gcc - O3 optimization		
	no unrolling	x16 loop unrolling	no unrolling	x16 loop unrolling	no unrolling	x16 loop unrolling	no unrolling	x16 loop unrolling	
(32 Signatures and 8 signature parallelism in VHDL)									
256 / 64	43.80 sec	35.29 sec	4.17 sec	3.45 sec	-		-		43.48 ms
64 / 64	11.47 sec	9.01 sec	0.85 sec	0.92 sec	132.85 sec	96.48 sec	20.59 sec	15.32 sec	10.87 ms (368MHz)
16 / 16	3.27 sec	2.52 sec	0.42 sec	0.43 sec	36.08 sec	26.16 sec	3.89 sec	3.44 sec	9.55 ms (419MHz)
(**64 signatures and 16 signature parallelism in VHDL**)									
256 / 64	86.48 sec	67.96 sec	9.32 sec	6.89 sec	-		-		65.32 ms
64 / 64	22.82sec	17.30 sec	1.64 sec	1.82 sec	263.10 sec	189.92 sec	31.98 sec	27.63 sec	16.33 ms (241MHz)
16 / 16	6.25 sec	5.01 sec	0.81 sec	0.86 sec	71.07 sec	51.26 sec	7.43 sec	6.62 sec	10.39 ms (385 MHz)

(128 signatures and 32 signature parallelism in VHDL)									
256 / 64	171.3 sec	134.31 sec	15.00 sec	13.84 sec	-		-		-
64 / 64	45.03 sec	35.11 sec	3.23 sec	3.69 sec	530.97 sec	375.84 sec	64.23 sec	53.68 sec	-
16 / 16	11.87 sec	9.67 sec	1.60 sec	1.72 sec	141.43 sec	101.58 sec	14.56 sec	12.96 sec	11.21 ms (357 MHz)

Table 4.6 Euclidean distance timing results

Chi-squared								
Channels/ parallelism	Software (C) - (Intel Core i3)				Software (C) - (ARM Cortex - A9)			VHDL
	gcc unoptimized		gcc - O3 optimization		gcc unoptimized	gcc - O3 optimization		
	No unrolling	X16 loop unrolling	no unrolling	x16 loop unrolling	no unrolling	no unrolling	x16 loop unrolling	
(32 signatures and 8 signature parallelism in VHDL)								
256 / 32	87.04 sec	76.66 sec	34.74 sec	31.64 sec	-	-		89.6 ms
64 / 32	19.51 sec	22.52 sec	8.70 sec	7.95 sec	362.73 sec	215.68 sec	222.15 sec	22.4 ms 357 MHz
16 / 16	5.12 sec	5.89 sec	1.79 sec	1.88 sec	92.55 sec	54.17 sec	56.42 sec	10.44 ms 383 MHz vivado
(64 signatures and 16 signature parallelism in VHDL)								
256 / 16	159.54	157.2 sec	71 sec	62.06 sec	-	-		179.2 ms
64 / 16	40 sec	44.64 sec	17.68 sec	15.85 sec	713.73 sec	425.6 sec		44.8 ms
16 / 16	10.31 sec	11.64 sec	3.59 sec	3.74 sec	184.47 sec	108.57 sec		11.2 ms 357 MHz vivado

Table 4.7 Chi-squared distance timing results

4.3.2 Results Commentary

In this section, we discuss the above tables and add some necessary clarifications on certain cases for better understanding of these results.

a. As already mentioned in the introduction, the cases for 256 spectral channels have not been implemented on the FPGA since the system configuration is identical to the 64 channel case with the addition of an accumulator at the end of the adder tree structure. Hence, for the Sum of absolute differences and the Euclidean distance metrics, the results for these cases were extracted through extrapolation from the 64 channel configurations.

   In the case of 8 signatures parallelism for the chi-squared metrics, due to increased device resource utilization of the division operation, implementation of the 64 and 256 channels designs was not achievable. A 32 channel parallelism is achievable and again with the same clock as the 16 signatures parallelism configuration. Execution times for these implementations are presented with a 32 channel parallelism and full signatures parallelism.

For the case of 16 signatures parallelism though, 64 channels and 256 channels configurations exceeded device resources, even for a 32 channels configuration, and could not be implemented. Thus, execution time results for these configurations were obtained through extrapolation from the 16 channel case.

b. In the case of the 64 channels and 32 signatures parallelism, only the sum of absolute differences metric configuration fitted in the device. In order to achieve this though, we had to enable area optimization options in Xilinx ISE (LUT combining, resource sharing) which have an impact on maximum achievable frequency. Therefore, clock frequency is significantly lower than the other implementations. Since data processing parallelism is very high, overall speedup is increased as we will see in the next section.

c. In the case of the 64 channels and 16 signatures parallelism, the total needed DSP blocks exceeded the available blocks. (1024 needed – 900 available). We implemented the above configuration mapping only some of the multipliers in DSP blocks and implementing the rest in LUTs. This was achieved using the "AutoMAX DSP" option in the Xilinx ISE Synthesis tool. Hence, the resource utilization does not follow a linear increase like the other implementations.

d. As we can see from the device utilization tables, resource requirements increase in a linear way to the increase of parallelism either in channel parallelism or signature parallelism. An exception to this rule is the case for the Euclidean distance metric where DPS requirements exceed available resources. In this case, linearity is lost due to an increased Register and LUT utilization.

e. The 16 channels – 16 signatures parallelism configuration also fits in the smaller Xilinx Zynq®-7020 SoC [11]. Almost all designs that are larger than that don't fit in the smaller board.

f. It is very interesting to discuss the difference in utilization for each metric. The Sum of absolute differences is the implementation with the least resource requirements. Even though Register and LUT resource requirements differences are not very large, the main difference is the DSP utilization which is very important in designs where DSP slices are valuable. Thus, the maximum achievable parallelism for the sum of absolute differences system is higher than the other metrics.
Furthermore, it is quite obvious that the chi-squared metric has very high resource requirements due to the difficulty in implementing the division on hardware.

4.4 Speedup and Throughput

In this section we present the outcomes of the acceleration analysis we performed on our VHDL design. We provide speedup results for the FPGA design versus the two C implementations.
Furthermore, as we already discussed, we conducted a throughput analysis in order to determine the data input requirements. This analysis has a great significance as we will see in this section since maximum achievable input throughput is a major limiting factor in the total maximum acceleration we can attain.
Finally, all results are illustrated in the form of diagrams for better understanding of the results and the correlation of the different factors that affect performance.

4.4.1 Results Tables and Diagrams

In the next tables the reader can see, in the execution time column, the best performance result for each of the aforementioned cases, taken from the tables in the previous section, along with the FPGA execution time results. In the speedup column, the achieved acceleration versus the two C code implementations is listed.

FPGA Configuration		**Data**	**SAD Execution time**			**Speedup**	
Signature Parallelism	**Channel Parallelism**	**Channels**	**Software (C)**		**VHDL**	**VHDL vs.**	
			i3	**ARM**	**FPGA**	**i3**	**ARM**
8	16	16	0.37 sec	3.5 sec	9.35 ms	40	374
	64	64	0.52 sec	26.72 sec	12.38 ms	46	2377
		256	3.23 sec	106.88 sec	49.52 ms	72	2377
16	16	16	0.75 sec	6.73 sec	10.75 ms	70	626
	64	64	1.03 sec	40.36 sec	13.94 ms	74	2895
		256	6.99 sec	161.44 sec	55.76 ms	125	2895
32	16	16	1.45 sec	12.87 sec	11.33 ms	128	1136
	64	64	2.02 sec	83.07 sec	24.85 ms	81	3343
		256	13.17 sec	332.28 sec	99.4 ms	133	3612

Table 4.8 Sum of absolute differences speedup results

FGPA Configuration		Data	EUCLIDEAN Execution time			Speedup	
Signature parallelism	Channel parallelism	Channels	Software (C)		VHDL	VHDL vs.	
			i3	ARM	FPGA	i3	ARM
8	16	16	0.42 sec	3.44 sec	9.55 ms	44	360
	64	64	0.85 sec	15.32 sec	10.87 ms	78	1409
		256	3.45 sec	61.28 sec	43.48 ms	79	1410
16	16	16	0.81 sec	6.62 sec	10.39 ms	78	637
	64	64	1.64 sec	27.63 sec	16.33 ms	100	1692
		256	6.89 sec	110.52 sec	65.32 ms	105	1692
32	16	16	1.60 sec	12.96 sec	11.21 ms	143	1156
	64	64	3.23 sec	53.68 sec	-	-	-
		256	13.84 sec	-	-	-	-

Table 4.9 Euclidean speedup results

FPGA Configuration		Data	CHI-SQUARED Execution time			Speedup	
Signature Parallelism	Channel Parallelism	Channels	Software (C)		VHDL	VHDL vs.	
			i3	ARM	FPGA	i3	ARM
8	16	16	1.79 sec	54.17 sec	10.44 ms	171	5189
	32	64	7.95 sec	215.68 sec	22.4 ms	355	9629
		256	31.64 sec	862.7 sec	89.6 ms	353	9628
16	16	16	3.59 sec	108.57 sec	11.2 ms	321	9694
		64	15.85 sec	425.6 sec	44.8 ms	354	9500
		256	62.06 sec	1702.4 sec	179.2 ms	346	9500

Table 4.10 Chi-squared speedup results

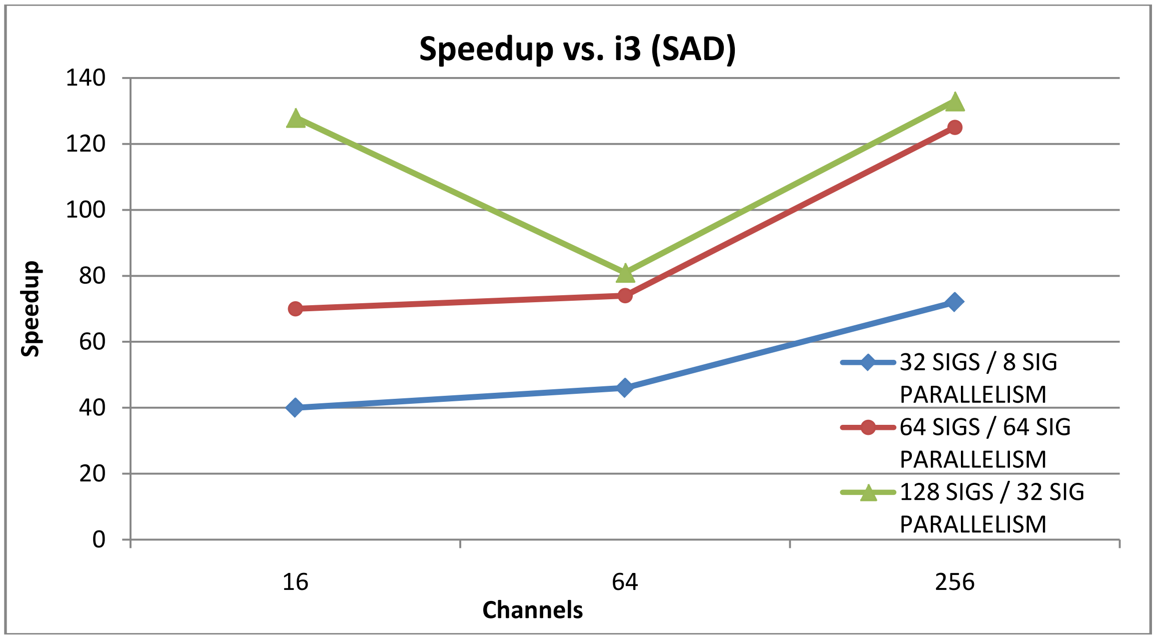


Figure 4.9 SAD speedup, FPGA vs. Intel i3

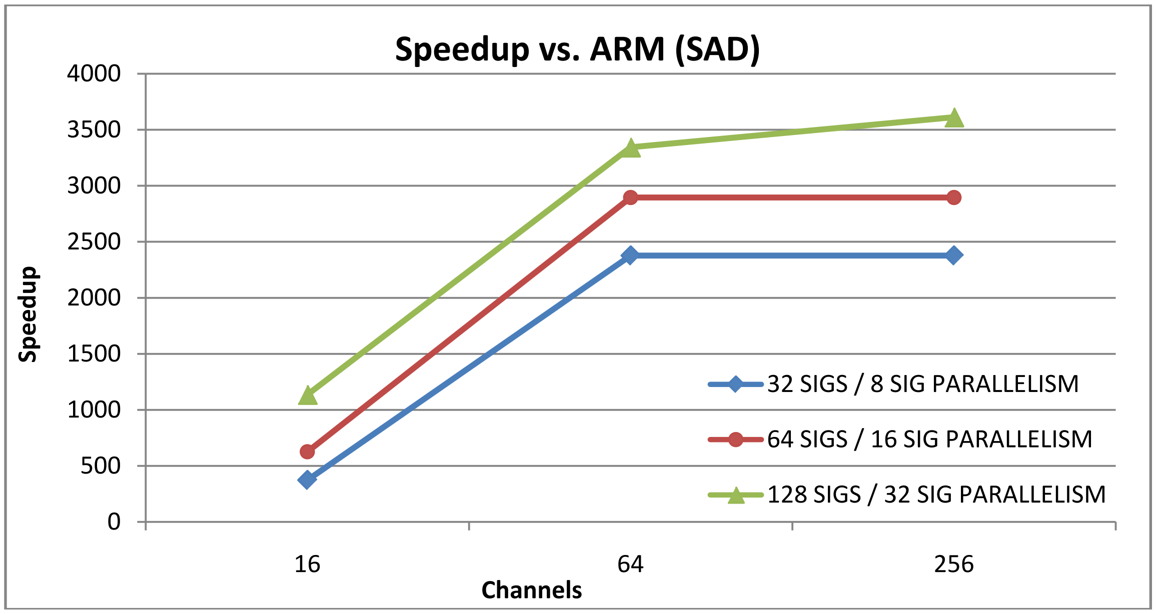


Figure 4.10 SAD Speedup, FPGA vs. ARM

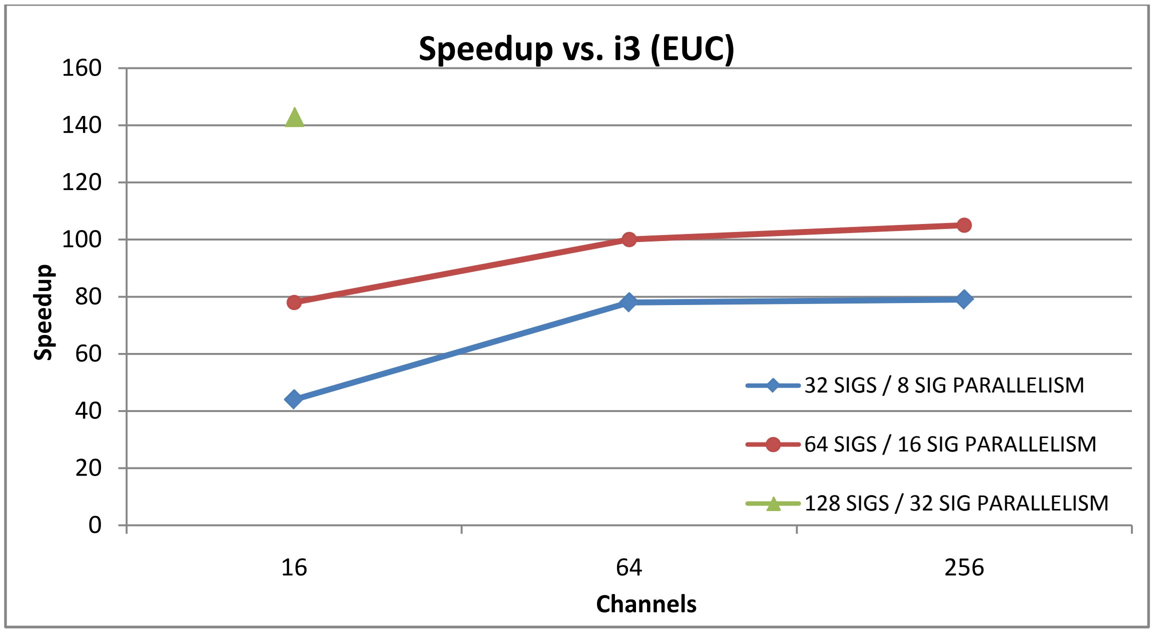


Figure 4.11 EUC speedup, FPGA vs. Intel i3

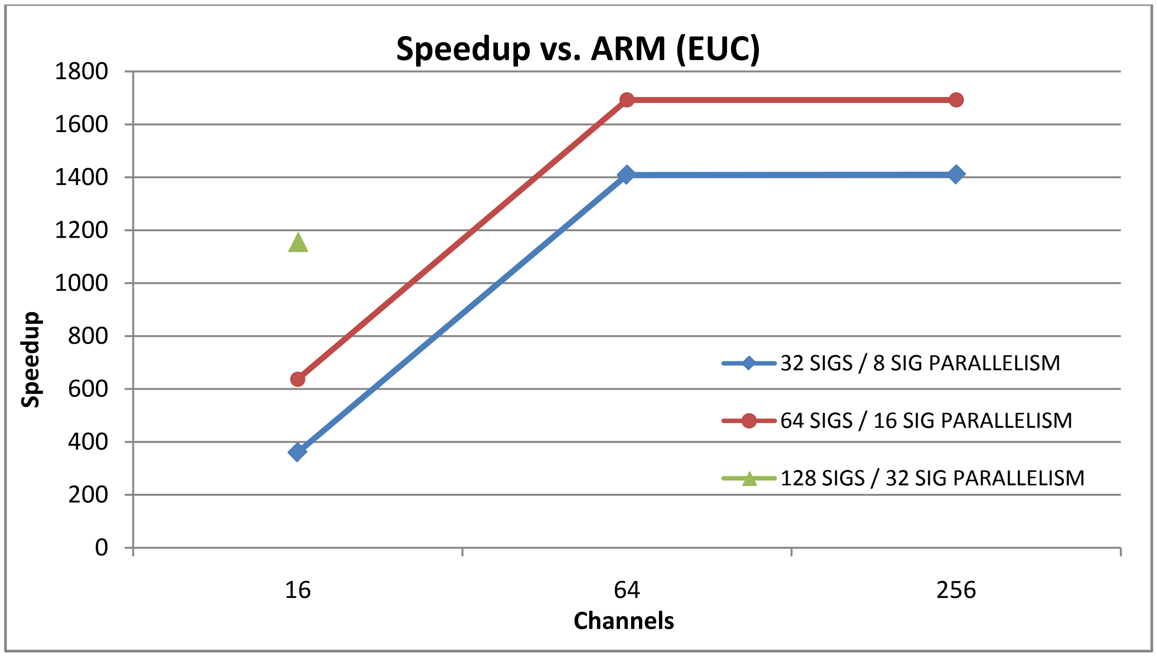


Figure 4.12 EUC speedup, FPGA vs. ARM

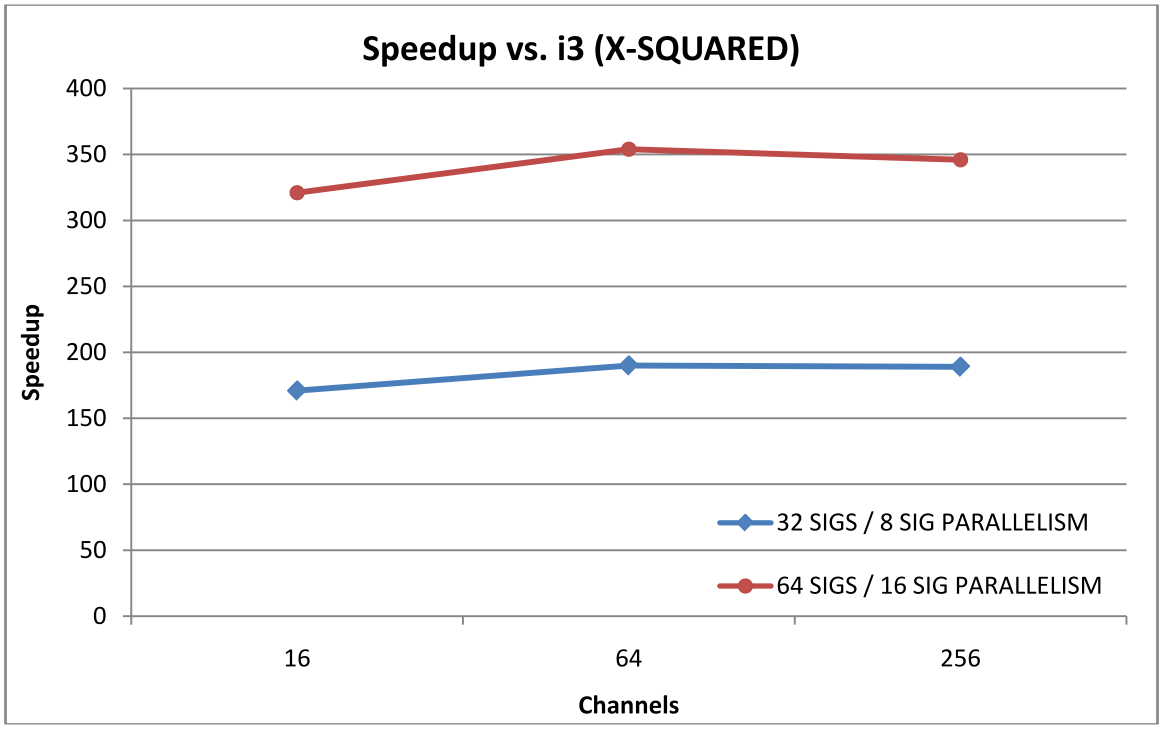


Figure 4.13 Chi-squared speedup, FPGA vs. Intel i3

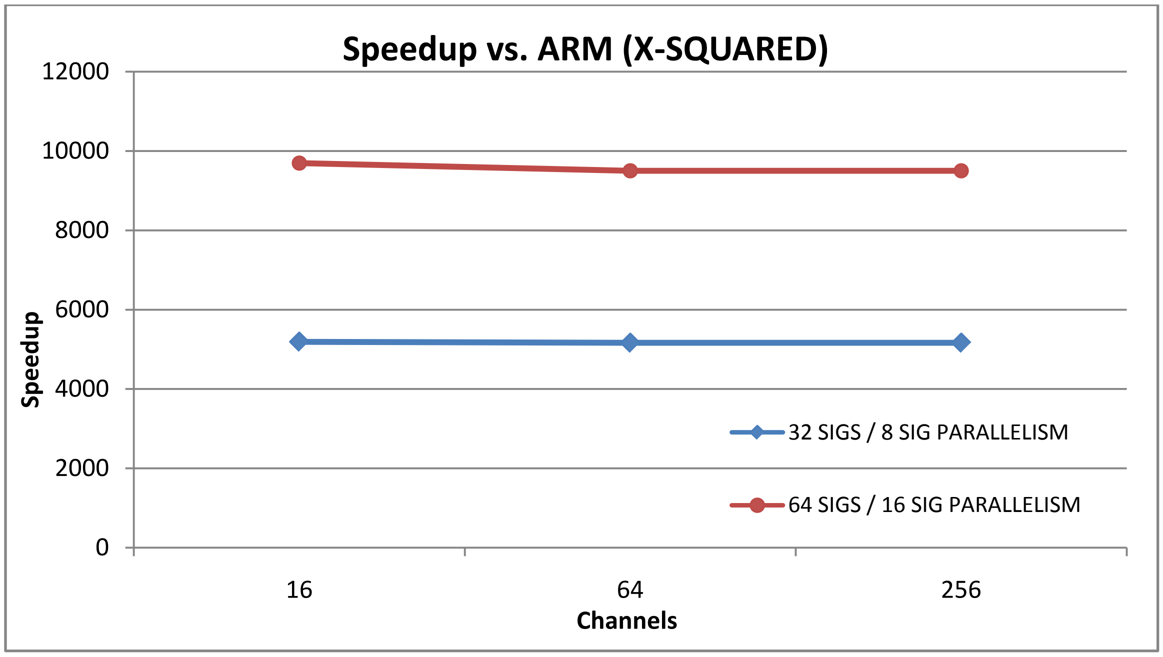


Figure 4.14 Chi-squared speedup, FPGA cs. ARM

We now present the throughput results obtained for the same configurations. Throughput was calculated using the following formula which takes into account clock speed of the FPGA, clock cycles needed until the final result has been calculated and data size.

$$Throughput = \frac{F}{C_p} \cdot w_d$$

where F is clock frequency, C_p is the needed clock cycles to match one pixel and w_d is the pixel / signature bit-width.

SAD		**PIXEL INPUT THROUGHPUT**					
		16 CHANNELS			**64 CHANNELS**		
Signature Parallelism		**8**	**16**	**32**	**8**	**16**	**32**
FPGA Clock		428 MHz	372 MHz	353 MHz	356 MHz	287 MHz	161 MHz
Speedup (Mean)	**i3**	48	83	156	282	465 MHz	521 MHz
	ARM	356	619	1174	1987	3173	3560
Number of Signatures	32	17.12 Gbps	29.76 Gbps	56.48 Gbps	56.96 Gbps	91.84 Gbps	103.04 Gbps
	64	8.56 Gbps	14.88 Gbps	28.24 Gbps	28.48 Gbps	45.92 Gbps	51.52 Gbps
	128	4.28 Gbps	7.44 Gbps	14.12 Gbps	14.14 Gbps	22.96 Gbps	25.76 Gbps
	256	2.14 Gbps	3.72 Gbps	7.06 Gbps	7.12 Gbps	11.48 Gbps	12.88 Gbps
	512	1.07 Gbps	1.86 Gbps	3.53 Gbps	3.56 Gbps	5.74 Gbps	6.44 Gbps

Table 4.11 Sum of absolute differences throughput results

EUC		PIXEL INPUT THROUGHPUT					
		16 CHANNELS			64 CHANNELS		
Signature Parallelism		8	16	32	8	16	32
FPGA Clock		419 MHz	385 MHz	357 MHz	368 MHz	241 MHz	-
Speedup (Mean)	i3	42	78	144	75	100	-
	ARM	343	630	1168	1282	1691	-
Number of Signatures	32	16.76 Gbps	30.8 Gbps	57.12 Gbps	58.89 Gbps	77.12 Gbps	-
	64	8.38 Gbps	15.4 Gbps	28.56 Gbps	29.45 Gbps	38.56 Gbps	-
	128	4.19 Gbps	7.7 Gbps	14.28 Gbps	14.72 Gbps	19.28 Gbps	-
	256	2.10 Gbps	3.85 Gbps	7.14 Gbps	7.36 Gbps	9.64 Gbps	-
	512	1.05 Gbps	1.93 Gbps	3.57 Gbps	3.68 Gbps	4.82 Gbps	-

Table 4.12 Euclidean throughput results

CHI-SQUARED		PIXEL INPUT THROUGHPUT		
		16 CHANNELS		
Signature Parallelism		8	16	32
FPGA Clock		383 MHz	357 MHz	-
Speedup (Mean)	i3	176	291	-
	ARM	4965	9475	-
Number of Signatures	32	15.32 Gbps	28.56 Gbps	-
	64	7.66 Gbps	14.28 Gbps	-
	128	3.83 Gbps	7.14 Gbps	-
	256	1.92 Gbps	3.57 Gbps	-
	512	0.96 Gbps	1.79 Gbps	-

Table 4.13 Chi-squared throughput results

In the following diagrams, we can see speedup for the FPGA versus the Intel i3 and the ARM processor for three signature parallelism as a function of spectral channels.

We also present a speedup / throughput diagram for 6 configurations for a fixed number of signatures and variable channel and signal parallelism. Specifically we chose to plot the results for 128 signatures which is a case that gives representative results for the whole system. In these diagrams, the gradient color of the FPGA throughput results line corresponds to the color coding of the throughput results in Tables 4.11 – 4.13. It has been depicted in such a way that illustrates the difficulty of each implementation in terms of input throughput.

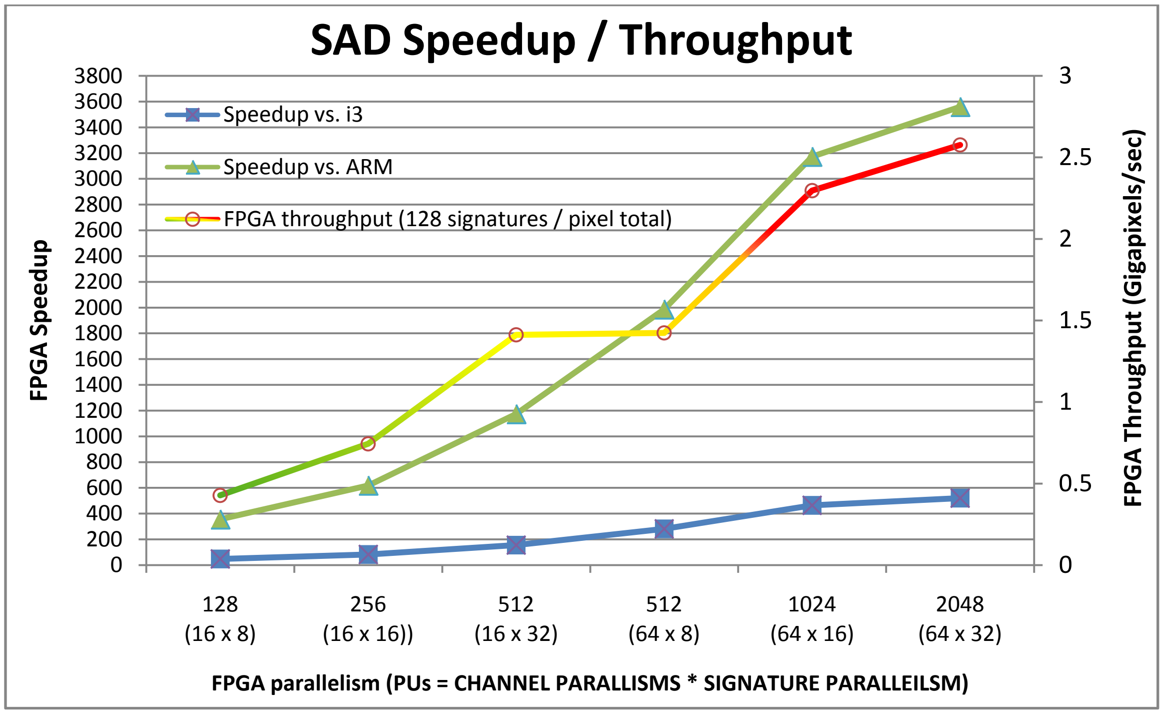


Figure 4.15 SAD Speedup and throughput for 128 signature and total pixel configuration

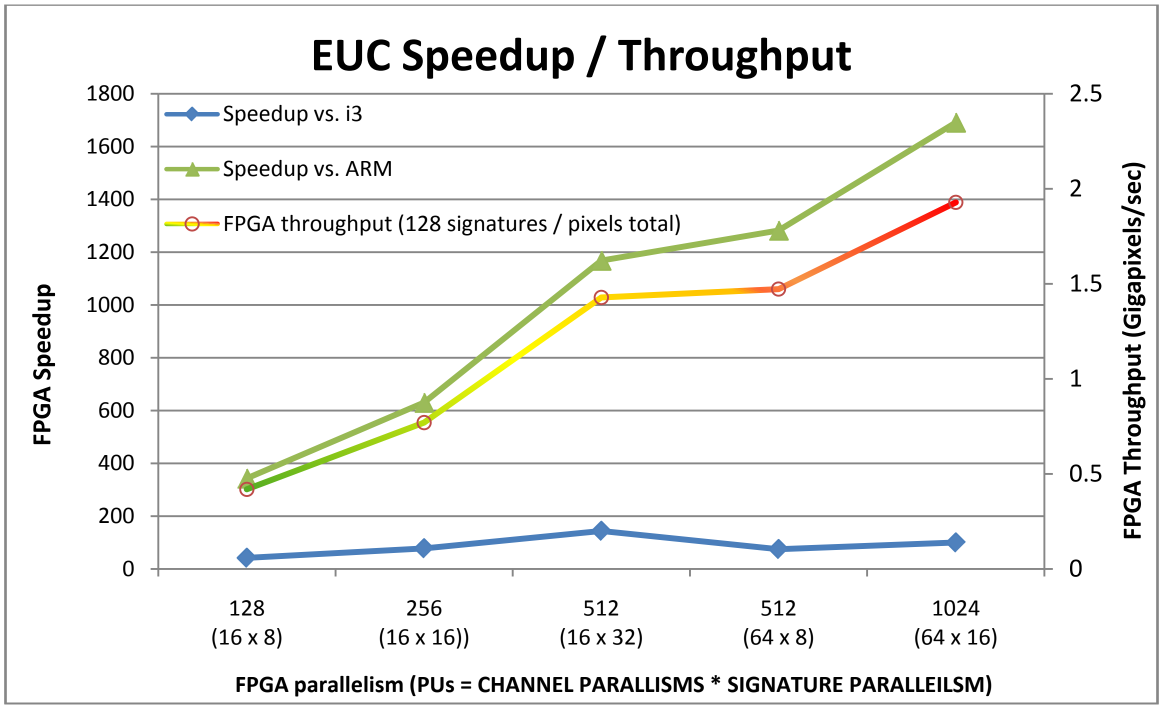


Figure 4.16 EUC Speedup and throughput for 128 signature and total pixel configuration

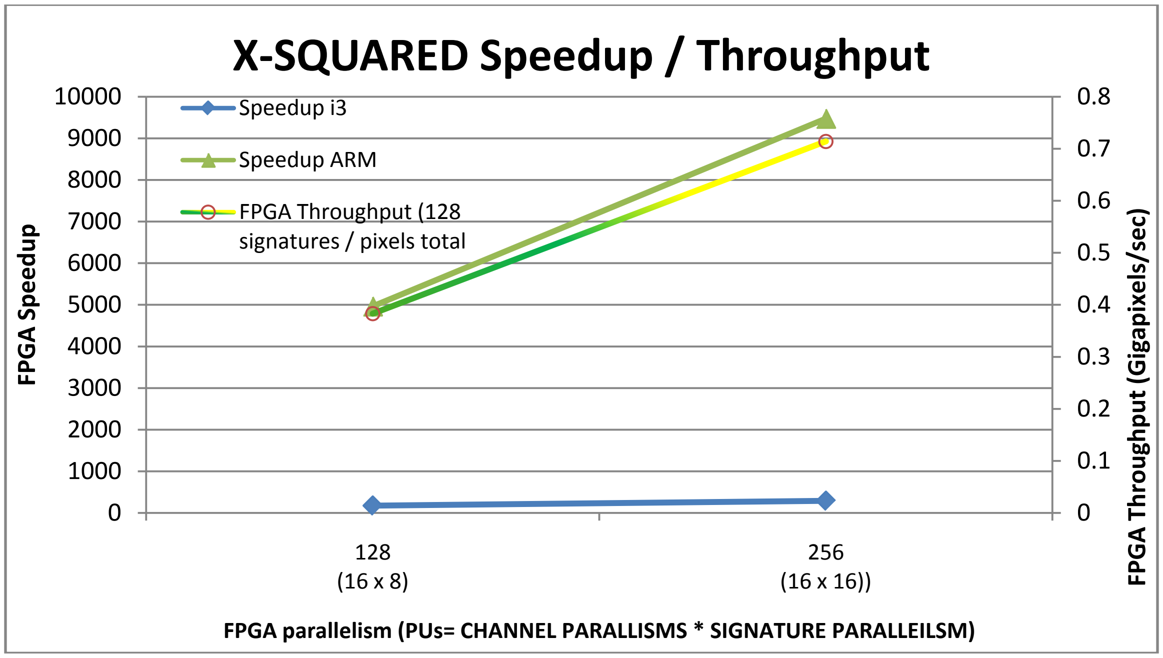


Figure 4.17 Chi-squared Speedup and throughput for 128 signature and total pixel configuration

4.4.2 Results Discussion

We discuss in this section about some important matters regarding the above diagrams.

a. In the result table 4.8 – 4.10, execution time results for the 256 channel C code implementation on the embedded ARM processor, were extracted through extrapolation from the 64 channel results assuming a linear relationship between execution time and workload by studying the rest of the results.
The reason behind the need of extrapolating the results is the limitation imposed by the board's internal memory. Our C program creates two matrices, one for 1 million pixels and one for given number of signatures before it performs the computations between them. In the case of the 256 channels, the available device memory was too small to fit said matrices.

b. As also mentioned before, the execution time numbers for the 256 channel cases in the VHDL case were also obtained through extrapolation from the 64 channel cases. Device configuration is almost identical due to same parallelisms in both cases.

   The above two points are the explanation behind the fact that for the 64 channels and 256 channels, the resulting diagrams have a linear relationship between them. In the case of the FPGA vs. ARM, we see no improvement in speedup between the two cases while in the FPGA vs. i3 curves, the lines are still parallel to each other but a speedup gain is present due to nonlinear execution time relationship between the Intel processor results.

c. The Chi-squared distance metric implementation is a case that needs a bit more attention. The first thing we have to mention is the fact that the 32 signatures parallelism cases were not implementable due to device resource limitations. Thus, they were not examined at all and not included in the results tables.

   The only designs that could be mapped into the device we examined are the cases for the 16 channel cameras. The results for the 64 channels and 256 channels were obtained through extrapolation. This is the reason also why we only presented these two cases in the Speedup / Throughput graphs. The results for the other cases are identical as obviously expected and were not considered necessary to illustrate.

d. Throughput results in Tables 4.11 – 4.12 have different coloring depending on the device's capability to reach each throughput. Any I/O throughput up to ~4 Gbps is easily achieved with available boards. A theoretical throughput of up to 12 Gbps is achievable on the Zynq-7000 SoC [19].
Throughput values up to ~18 Gbps will be achievable in the next generation *Zynq Ultrascale+ MPSoC* according to the device's advertised specifications [20].
Anything higher than the aforementioned values is practically impossible to achieve (with currently available devices) and the speedups for these cases are

presented in order to complete the picture of the device’s computational capabilities.
It is evident that throughput is a factor that limits the achievable maximum speedup on a practical level and has therefore to be taken into account when designing a system.
Results for 256 and 512 signatures have been extracted assuming a same configuration as in the 128 signatures case. They are presented here in order to display the fact that increasing the total signatures number, provides us the advantage of lower I/O throughput and better quality results as we will see in Chapter 5.

CHAPTER 5

QUALITY EVALUATION

5.1 Introduction

This chapter is dedicated to the examination of the quality results of our design for a number of different configurations. We will discuss the methodology used to extract features from a hyperspectral image using MATLAB, impact of various factors on the resulting qualities and difference in the results for our three different matching metrics.

The main data set used for our experiment was the open science data set from APEX we already introduced in Chapter 1. We extracted specific regions that were of interest to us, as well as necessary spectral signatures, in order to obtain the desired results. We repeated our measurements for different camera configurations, studying the effect of different regions of the light spectrum and the number of channels on the qualities of the obtained results. We will also present certain results from the analysis of the smaller dataset obtained with the hyperspectral cameras provided by the remote sensing laboratory of the NTUA. Detailed analysis on this dataset was not conducted since the APEX dataset did cover a much broader spectrum of diverse cases.

Finally, we close the chapter with commentary on the results and with a brief presentation of other methods used by image analysts to obtain desired features from a hyperspectral image.

5.2 Dataset Manipulation and Quality Evaluation

After acquiring the needed datasets and all the different camera configurations mentioned in previous Chapters, we have to manipulate the data in order to test our matching metrics on various cases and evaluate the results obtained for each configuration. In order to do this, we generate a number of different ground truths for different cases. We also acquire numerous spectral signatures for each dataset that will then be used in the matching process. Finally, we close the section with the discussion of the quality measures used to evaluate the results in each case.

5.2.1 Ground Truth generation and spectral signatures acquisition

The APEX dataset consists of a hyperspectral image obtained from a remote sensor mounted on a research aircraft and taken over a small city in Switzerland. Since it depicts such a diverse area covering a wide range of different ground features and materials, we had to concentrate our analysis on specific areas, well suited to generate the required ground truth and the corresponding signatures. Before we discuss each region of interest individually, we present the full image for the aforementioned area. In Figure 5.1, one channel of the 285 spectral channels for the entire image can be seen in grayscale. This image was used to obtain the desired features from the image.

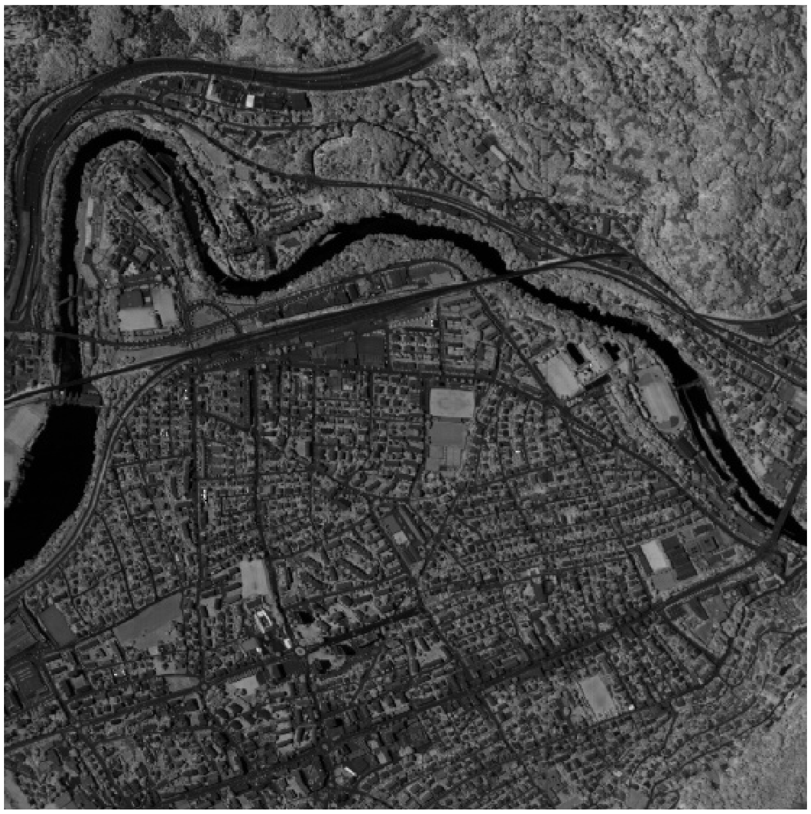


Figure 5.1 Channel 130 of the APEX open science data set

From the image depicted in Figure 5.1, we extracted certain areas for further analysis. Specifically, we concentrated on five subareas generating six ground truth regions with specific features. The examined regions are the following:

a. Water
b. Tennis Courts
c. Black Roof Tiles
d. Paved Road
e. Soccer field
f. Vegetation (trees)

For each one of the above regions of interest we generated a corresponding ground truth. These areas were created in MATLAB using the desired region from the original image and the ROI function from MATLAB. The resulting map is a binary image with a high value (‘1’) for every pixel inside the ground truth region and a low value (‘0’) anywhere else. Figures 5.2 to 5.4 depict each of the above regions along with the generated ground truth map.

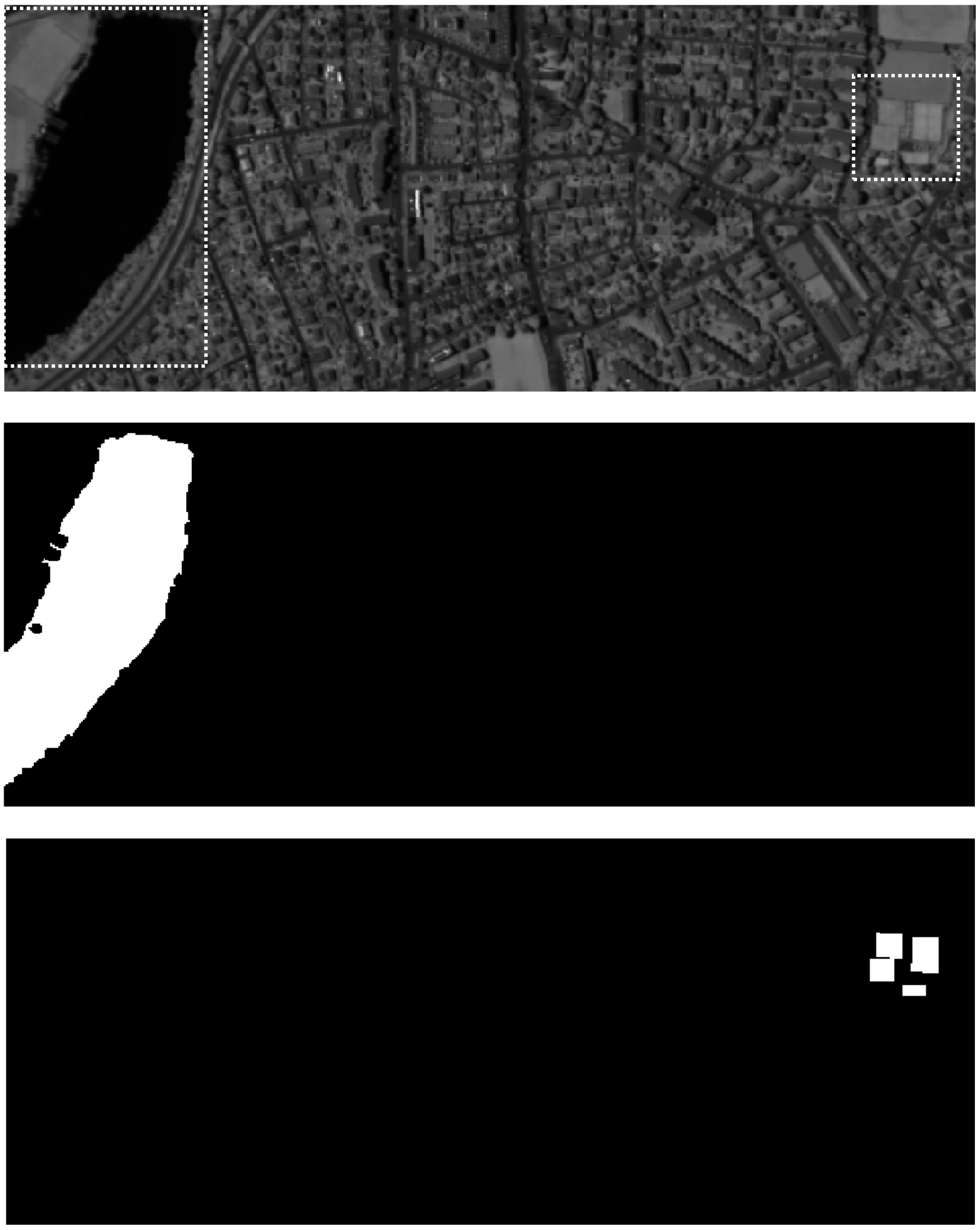


Figure 5.2 Regions of interest and ground truth map for **a**. Water **b**. Tennis courts

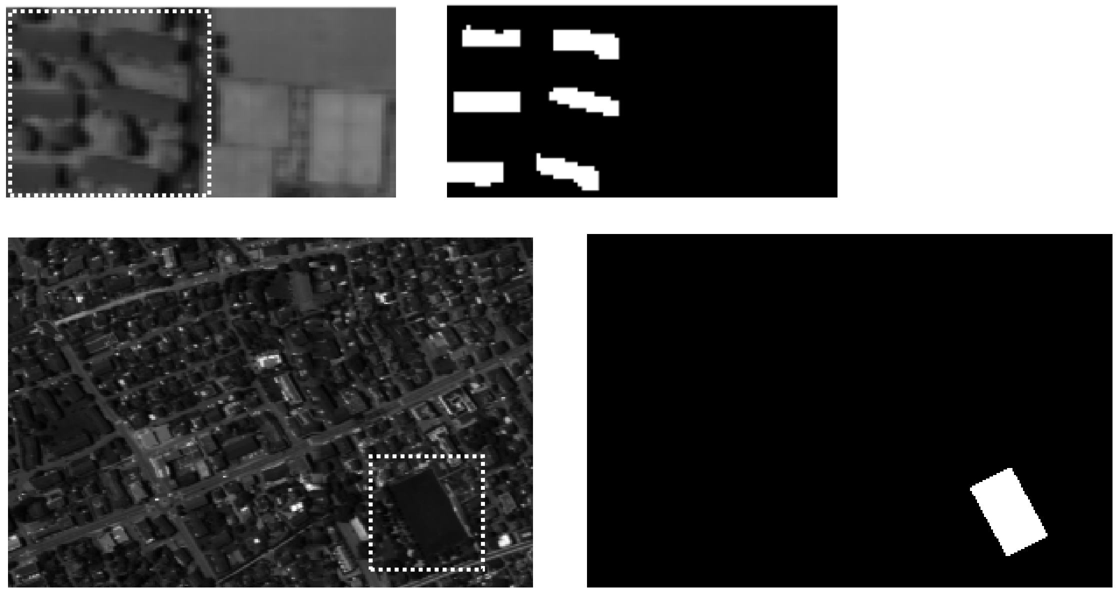


Figure 5.3 Regions of interest and ground truth map for **a**. Roof Tiles **b**. Soccer field

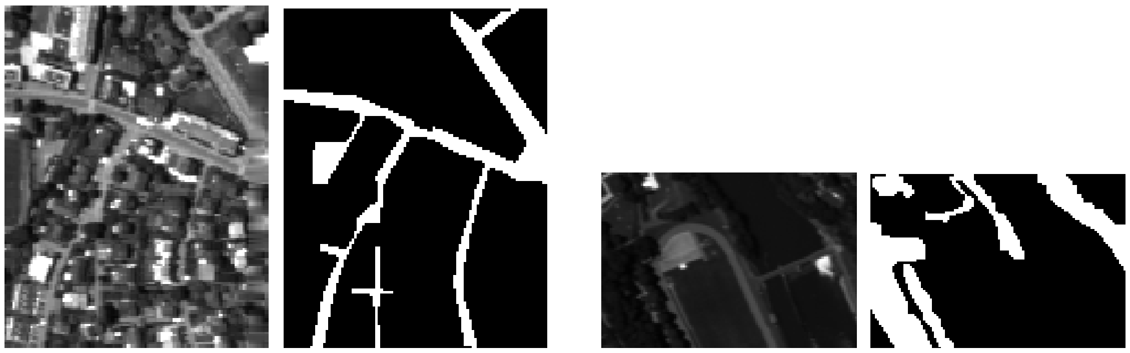


Figure 5.4 Regions of interest and ground truth map for **a**. Paved Road **b**. Vegetation

Now that we have defined and generated the ground truth for the needed calculations, we discuss how to extract spectral signatures from each region. The methodology we follow for signature acquisition is quite straight forward.
For each region we find the coordinated on MATLAB for specific pixels and save each one of them in a matrix variable of size 1x1x256 (we can subsample the needed channels if needed). We extract around four to five pixels from each region depending on the data diversity. We will see in the following sections why more than one signature is needed and the difference in results depending on the number of signatures used to match each region.
In general, for each region we can extract tens of signatures to match which is a common and acceptable practice in many applications. In our case we did limit our examination to four or five pixels per ground truth which were enough to prove the impact of increased signature numbers on the resulting qualities. It is possible to match each signature

independently and then combine the results or find a mean value from all signatures of the same ground truth region and then perform necessary calculations. Finally, we can also find the mean value of all signatures and add it as an "extra signature" in the first method. The differences in the results in each one of these cases will be evident in the results tables.

5.2.2 Quality Metrics

Trying to quantify the concept of quality in image matching, we use certain metrics that do that exactly. Before we describe these metrics let us first talk about some statistical measures of the performance of our matching functions. These measures are the *True Positive rate (TP)*, *False Positive rate (FP)* and *False Negative rate (FN)*. In our case of pixel matching, true positive pixels are correctly identified pixels in the image. False positive pixels are these pixels that were matched with a given signature but do actually not belong to the generated ground truth. Finally, false negative pixels are pixels that were not matched to a given signature but do actually belong to the generated ground truth.
The three aforementioned statistical measures are used to form three quality measures, commonly used in quality assessment systems. These measures are *Completeness*, *Correctness* and *Quality*. We also implemented an additional quality measure, similar to Completeness, which introduces different weights for the TP and FP pixels, giving greater importance to the TP pixels. The quality measures presented here are defined as follows:

$$Completeness = \frac{TP}{TP + FN} \tag{5.1}$$

$$Weighted = \frac{TP - \frac{1}{2}FP}{TP + FN} \tag{5.2}$$

$$Correctness = \frac{TP}{TP + FP} \tag{5.3}$$

$$Quality = \frac{TP}{TP + FP + FN} \tag{5.4}$$

Completeness measures correctly identified pixels as a percentage of the ground truth pixels, i.e how many pixels the system found from the ones it should have actually found.
As you can see from the above plot, completeness is a measure that decreases slightly while lowering the threshold until it collapses after a point. This is expected since lowering the threshold extremely, causes or system to reject "good" pixels.
The Weighted measure is similar to completeness but takes also into account the false positive pixels with a reduced weight.

Correctness on the other hand, measures how many of the matched pixels are actually correct. This measure increases as we lower the threshold approaching unity. A unity value would mean that our system has made no wrong matches. An increase in correctness is expected since lowering the threshold causes our system to reject “wrong” matches. It does not take into account lost “good” pixels which is done with the previous measure.

Finally, Quality describes a general measure that combines the completeness and correctness forming a more comprehensive picture of the concept of quality. The plot of this measure, as can be seen in the Figure 5.5 provides us with the necessary information to select the ideal threshold level.

5.3 Implementation on MATLAB

5.3.1 Introduction

Our goal, as we have described in previous sections, is to match each signature with its corresponding ground truth. In order to achieve this, we calculate the vector distance between a given signature and all image pixels using one of the previously mentioned distance metrics. For example, for a vector of 256 points (each point is one spectral channel), we calculate the distance of 256 points, accumulating the results to receive the final value. This value then is used to decide how good the match was.

In this chapter, we will now discuss the MATLAB system developed to extract the aforementioned matching results as well as certain quality metrics needed to quantify the concept of quality. We begin with a description of the system and its function and will move on to present the quality metrics and the results for different matching metric and camera configurations.

5.3.2 Matching System and Results Acquisition

The matching system we developed in MATLAB consists of two basic functions that perform the necessary calculations. Before we describe those functions in more detail let us describe the idea behind the matching process of our system.

Each aforementioned signature-pixel matching result has to be decided whether it is a “good” or a “bad result. The smaller the above mentioned result is, the closer the signature and the tested pixel vectors are. The decision of a “good match” is made using a threshold value for the above result so we can discriminate the pixels that we consider as “good matches” from the rest of the pixels in the image. The threshold value is chosen by the researcher based on several factors. Later on in this chapter we will discuss the method we used to choose the threshold value for each ground truth.

After choosing the threshold value, we create a logical map for each tested pixel. Every result below the given threshold is considered a match and is represented with a logical high value ('1') in the map. The rest is represented with a logical low ('0'). This map is then compared to the ground truth map we created in the beginning in order to calculate the quality metrics for each case.

The two functions created in MATLAB perform the above process. For every case we provide the functions with the image to be scanned, a high and low value for the threshold, the signatures we want to scan and the logical map for our ground truth.
The function does the following iteration in order to determine the optimal threshold for each ground truth and matching metric. Starting from the maximum distance value it repeats all calculations for the quality metrics recording the results lowering each time the threshold value by a given step until it reaches zero.
At the end of this process we receive a list of the quality results for all iterations. Every time we lower the threshold we see that the quality results improve since more pixels are left unmatched. However, for each signature / ground truth / matching metric combination, there is a threshold value after which quality starts declining again due to extremely strict limits. The value, for which the quality results are max, is chosen as the optimal threshold. A plot of the above mentioned results can be seen in Figure 5.5. In Section 5.3.2 we discuss these quality measures in order to better understand these results.

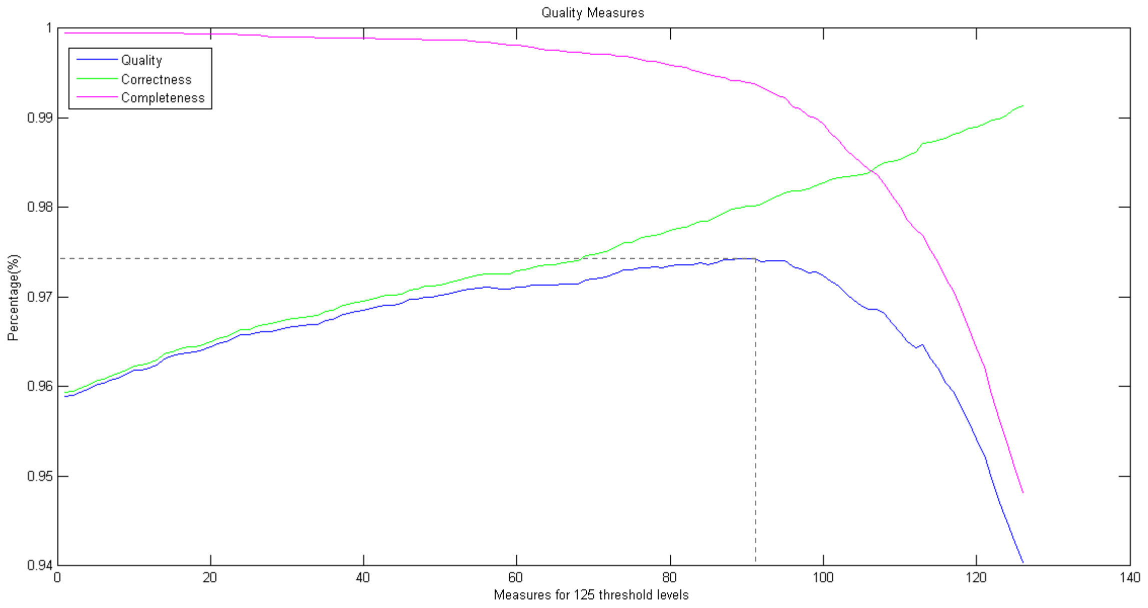


Figure 5.5 Quality results plot

5.4 Quality Results

In this Chapter we present the implementation results for the previously discussed MATLAB system and for the data described in Section 5.2. Before listing the final tables, we will briefly discuss the different cases examined in our experiment.
We performed each test for 6 different ground truths. For the Soccer field, Road and Vegetation ground truths, only the weighted completeness measure was recorded in the results tables since a complete examination of all possible configurations was not necessary. For the tested ground truths, we repeated our experiment for the three matching metrics also implemented on the FPGA. Finally, each of the aforementioned cases was tested on four camera configurations which will be analyzed in the following section.

5.4.1 Analytical Results for main Data Set

We now present the results of all cases we examined in the following tables. An in depth discussion about the results will follow.
Each table lists the results for the different ground truths, camera configurations, matching metrics and multiple signature combinations. We also present results for a limited number of cases for which we simulated reduced illumination conditions for our image. We approximated this effect dividing all channel values with a lowering factor.
Tables 5.2 to 5.4 list the results for the three commonly used quality measures. Tables 5.5 and 5.6 present the result for the weighted measure and have a slightly different layout.

	Completeness									
	DATASET	**WATER**			**ROOF**			**COURT**		
	METRIC	SAD	EUC	X-2	SAD	EUC	X-2	SAD	EUC	X-2
CAMERAS	**Full Range** 256 Ch.	99.35%	99.60%	99.40%	90.80%	93.10%	94.71%	86.67%	87.46%	87 %
	Near Infrared ~40 Ch.	98.70%	98.70%	98.73%	92.18%	93.10%	92.18%	85.33%	86.79%	88.35%
	Short-wave Infrared ~ 50 Ch.	99.38%	99.53%	99.36%	92.41%	90.57%	90.34%	50.95%	53.30%	64.28%
	Compact ~80 Ch.	99.42%	99.61%	99.51%	94.48%	93.10%	90.80%	66.85%	65.17%	86.34%
	Multi Spectral ~25 Ch. 10 Ch. Av.	99.38%	99.62%	99.59%	95.40%	93.10%	90.80%	87.35%	64.95%	87.23%

Table 5.1 Completeness

	Correctness									
	DATASET	**WATER**			**ROOF**			**COURT**		
	METRIC	SAD	EUC	X-2	SAD	EUC	X-2	SAD	EUC	X-2
CAMERAS	**Full Range** 256 Ch.	98%	97%	98%	85.13%	82.65%	81.58%	61.14%	64.28%	75.83%
	Near Infrared ~40 Ch.	97.15%	97.26%	97.20%	81.17%	80.36%	82.00%	83.55%	85.73%	85.76%
	Short-wave Infrared ~ 50 Ch.	98.21%	97.82%	97.69%	83.75%	84.73%	85.06%	47.15%	51.80%	46.18%
	Compact ~80 Ch.	97.82%	97.28%	97.93%	82.04%	82.65%	84.76%	76.83%	76.38%	75.29%
	Multi Spectral ~25 Ch. 10 Ch. Av.	97.93%	97.30%	97.79%	81.37%	82.65%	84.76%	59.82%	75.72%	73.21%

Table 5.2 Correctness

	Quality									
	DATASET	**WATER**			**ROOF**			**COURT**		
	METRIC	SAD	EUC	X-2	SAD	EUC	X-2	SAD	EUC	X-2
CAMERAS	**Threshold**	15,000	2,250K	6,000	117,000	150,500K	29,500	174,000	86,000K	17,500
	Full Range 256 Ch.	97.30%	96.87%	97.43%	78.37%	77.88%	78.03%	55.88%	54.44%	67.48%
	Threshold	2,600	195,000	900	4,150	800,000	360	11,600	4,700K	1,000
	Near Infrared ~40 Ch.	95.92%	96.02%	96.00%	75.95%	75.84%	76.67%	73.06%	75.83%	78.05%
	Threshold	3,300	280,000	1,800	36,200	24,600K	3,100	26,000	17,500K	2,700
	Short-wave Infrared ~ 50 Ch.	97.61%	97.37%	97.08%	78.36%	77.87%	77.98%	32.43%	34.63%	36.75%

	Threshold	5,500	800,000	2,500	48,100	39,600K	4,000	42,500	30,000K	5,800
	Compact ~80 Ch.	97.27%	96.92%	97.45%	78.29%	77.88%	78.06%	55.64%	54.24%	67.28%
	Threshold	1,600	235,000	800	15,600	12,300K	1,220	18,000	9,100K	1,800
	Multi Spectral ~25 Ch. 10 Ch. Av.	97.34%	96.94%	97.40%	78.30%	77.88%	78.06%	55.05%	53.75%	66.13%

Table 5.3 Quality

	WEIGHTED (4 signatures)									
	DATASET:	**SOCCERFIELD**			**ROAD**			**TREES (minus grass)**		
	METRIC:	SAD	EUC	X-2	SAD	EUC	X-2	SAD	EUC	X-2
	Signatures:	3 signatures + mean			3 signatures + mean			3 signatures + mean		
CAMERAS	**Full Range** 256 Ch.	65.08%	61.74%	70.34%	33.14%	31.83%	34.92%	44.72%	41.87%	45.31%
	Hyper on Visible ~ 40 Ch. 0.4 - 0.7 μm	25.76%	25.55%	15.34%	25.76%	27.40%	29.08%	11.22%	1.82%	11.68%
	Visible and near infrared ~ 40 Ch. 0.4 - 1.1 μm	47.48%	46.91%	61.10%	25.98%	24.31%	28.84%	22.61%	21.56%	30.94%
	Shortwave Infrared ~ 50 Ch. 1.1 - 2.4 μm	60.21%	58.43%	62.03%	27.37%	26.53%	27.79%	42.85%	46.81%	44.85%

Table 5.4 Weighted measure for four signatures

	WEIGHTED (1 signature)									
	DATASET:	**SOCCERFIELD**			**ROAD**			**TREES (minus grass)**		
	METRIC:	SAD	EUC	X-2	SAD	EUC	X-2	SAD	EUC	X-2
	Signatures:	mean of 3 signatures			mean of 3 signatures			mean of 3 signatures		
CAMERAS	**Full Range** 256 Ch.	62.63%	58.18%	69.28%	30.89%	27.19%	33.06%	33.90%	28.39%	39.50%
	Hyper on Visible ~ 40 Ch. 0.4 - 0.7 μm	0.00%	4.28%	3.31%	21.89%	23.34%	24.88%	0.00%	0.00%	0.00%
	Visible and near infrared ~ 40 Ch. 0.4 - 1.1 μm	37.25%	34.19%	58.64%	19.64%	18.09%	23.31%	12.98%	11.22%	28.64%
	Shortwave Infrared ~ 50 Ch. 1.1 - 2.4 μm	57.50%	61.02%	59.07%	26.40%	24.49%	26.74%	36.98%	35.68%	38.37%

Table 5.5 Weighted measure for one signature

Finally, we present a number of comparison tables in order to better illustrate the impact of certain factors on quality results. The factors examined in these tables are the quality improvement achieved with the use of multiple signatures as well as the quality differentiation in the case of reduced image illumination. The values presented here will help us better understand the comments in the next section.

COURT								
1 Signature 100% Luminosity			**3 Signatures 100% Luminosity**			**3 Signatures 80% Luminosity**		
SAD	EUC	X-2	SAD	EUC	X-2	SAD	EUC	X-2
86.67%	87.46%	87%	85.67%	85.44%	94.62%	76.71%	84.66%	79.64 %

Table 5.6 Completeness results for multiple signatures and reduced luminosity

COURT								
1 Signature 100% Luminosity			**3 Signatures 100% Luminosity**			**3 Signatures 80% Luminosity**		
SAD	EUC	X-2	SAD	EUC	X-2	SAD	EUC	X-2
61.14%	64.28%	75.83%	74.63%	76.68%	79.34%	85.63%	84.56%	88.55%

Table 5.7 Correctness results for multiple signatures and reduced luminosity

COURT								
1 Signature 100% Luminosity			**3 Signatures 100% Luminosity**			**3 Signatures 60% Luminosity**		
SAD	EUC	X-2	SAD	EUC	X-2	SAD	EUC	X-2
55.88%	54.44%	67.48%	66.35%	67.82%	75.92%	51.16%	52.30%	62.40%

Table 5.8 Quality results for multiple signatures and reduced luminosity

	WEIGHTED (4 signatures - Reduced Illumination)						
	DATASET	**SOCCERFIELD**			**TREES (minus grass)**		
	METRIC	SAD	EUC	X-2	SAD	EUC	X-2
	signatures	3sigs + mean			3sigs + mean		
100% Illumination	**285 Channels** 0.4 - 2.5 nm	65.08%	61.74%	70.34%	45.72%	41.87%	44.31%
10% Illumination		34.04%	28.14%	40.85%	32.70%	30.08%	39.18%

Table 5.9 Weighted measure for reduced illumination

5.4.2 In-house Camera Results

Before closing the MATLAB results section, we will present limited results from the examination of the smaller, secondary data set obtained by us with the hyperspectral cameras provided by the remote sensing lab of the NTUA. We will not discuss quantitative results at this point but will illustrate the general picture of the examination we did on this data set. A true color image of our data cube was already presented in Section 2.5. In figure 5.6 an image of our hyperspectral camera for one channel can be seen.

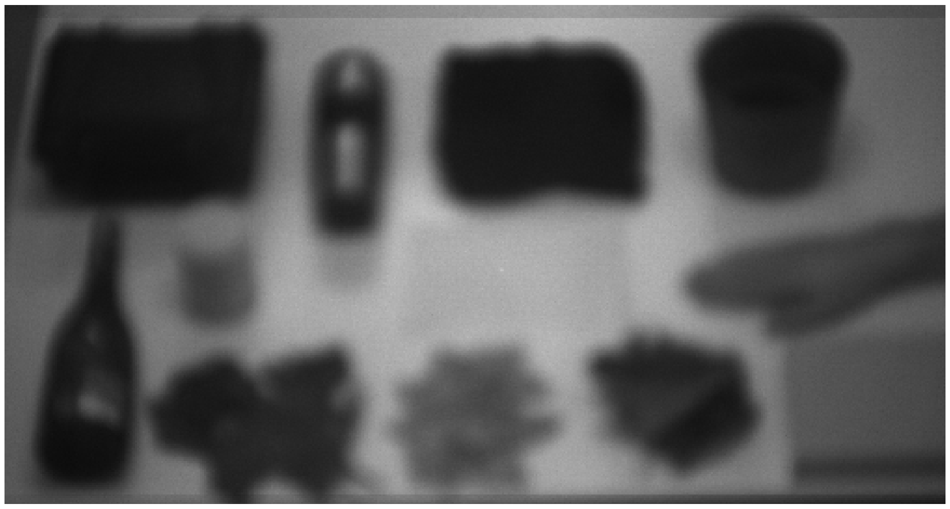
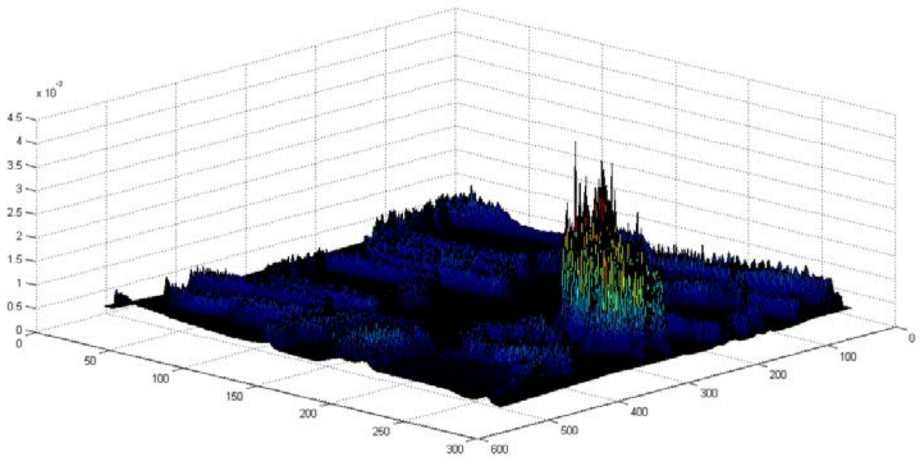


Figure 5.6 Channel 11 of Hyperspectral Image Data Set.

For the above image data we performed repeated experiments for the three different matching metrics trying to identify objects in the image. For this, we developed a set of functions in MATLAB that take a known signature (mean value of 9 signatures), our image and the desired matching metric and produce a map of the distance results for each pixel. The inverse distance map for each signature / metric combination can then be used in order to perform a visual evaluation of the quality of the results. An example of this procedure for the matching of yellow leaves can be seen in Figure 5.7. The differences between the two metrics are more obvious in this example but are illustrative of the fact we also observed for the large APEX data set.



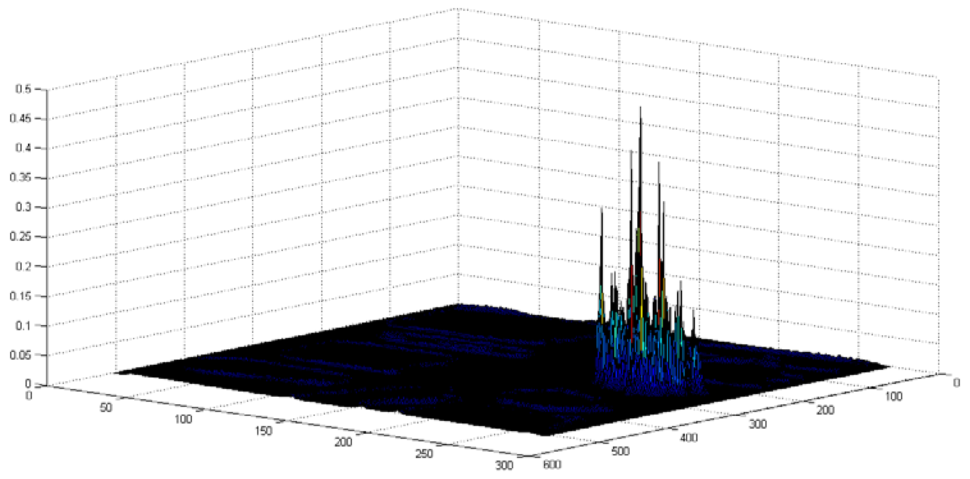


Figure 5.7 Matching yellow leaves for **a**. SAD, **b**. Chi-squared metrics

5.5 Results Analysis and Comments

From the previously mentioned tables we can draw a number of important conclusions about the different aspects of our image matching system.

We begin with the discussion of the best conclusion from the above analysis. As we can tell from most experiments, using more signatures form an object, improves the results compared to using only the mean signature. Also, including the mean value in the list of multiple signatures improves in certain cases the overall results.
The above observation is very important in our application since the FPGA is very well suited for increased data processing and parallelism configurations.

As a second most important observation we can say that for almost all cases, the Euclidean distance metric provides the same or even worse results than the sum of absolute differences. This fact is of great interest if we take into account the corresponding hardware cost of each implementation as we will see in greater detail in Chapter 6. In some cases though, the Euclidean distance does indeed provide better results which, depending on the application, would justify the use of it.

When we focus on the overall Quality in the above results, we can draw the conclusion that in almost every meaningful camera / ground truth setup, the chi-squared matching metric improves the results by a percentage of up to 14%. Especially in cases assuming different lightning conditions we see increased differences between the chi-squared and the other metrics. The normalization metric results in less false positive pixels that would influence the overall result.

A significant observation can also be made regarding percentage coverage of the features in the image we want to identify. In general, the three matching metrics seam in that field very similar. It is thus important to note that this quality measure alone does not justify the use of the more complex and costly chi-squared implementation if object coverage is our main goal. The chi-squared metric could still be preferred over the other ones for very demanding applications where small details are important. Still, it has to be decided after careful consideration in all cases.

As a general suggestion, the sum of absolute differences seems to be the preferable choice for most applications. Additional techniques for post processing / homogeneity analysis could be used to assist in the extraction of useful featured from a hyperspectral image. This observation is very important in certain cases where distinguishing of objects is almost impossible by signature alone, for example when trying to identify different types of vegetation. Another case where quality results are not the only criterion for choosing a matching metric is the case in which we want to identify a certain area or formation on the ground. In this case, other characteristics like general shape or outline geometry are of great help in extracting needed information from an image without the need of perfect object coverage.

Moving on to the different camera configurations, we can clearly see the fact that specific areas in the light spectrum are more important than others when trying to identify and match light spectra. Thus, certain cameras are useless for certain objects. For example vegetation can be identified around the 600 – 800 nm wavelengths. A camera that consists of spectral channels in the shortwave infrared region is not suitable for this use.
In some cases, using a camera with fewer channels but in a spectral range that includes the wavelengths that dominate a specific material provides better results than a camera with a lot of channels and an increased spectral coverage range. This can be seen for certain configurations in the results tables in the previous section. An application specific analysis of the requirements is thus highly important in order to minimize cost and optimize results.

Finally we should point out that in all results, exceptions and randomness are present. Even though in some cases the results do not follow the general rules analyzed in this section, we have drawn conclusions from examining the patterns and the common characteristics of each case.
Also, the above analysis was based on the assumption of a near-perfect thresholding technique. In practice, we search around 10^4 thresholds around known percentile coming from the ground truth file, a priori. Differences in the results could possibly occur in a more realistic scenario / training.

This page is intentionally left blank.

CHAPTER 6

TRADEOFF ANALYSIS

6.1 Introduction

In Chapters 4 we examined in depth the implementation of a hyperspectral matching system. We recorded limitations introduced by the FPGA, hardware requirements and performance for different implementations and discussed data parallelism achievable on the device.

On the Quality assessment side in Chapter 5, we performed a detailed examination of hyperspectral image matching techniques and recorded quality results for each implementation. We discussed the factors that affect the quality results, listed limitations and advantages of each case and examined the impact of different camera and signature configurations on the results.

In this Chapter, we combine all conclusions drawn previously and will perform a tradeoff analysis for each factor. As we already mentioned previously, the three different matching metrics provide a difference in the quality results depending on the configuration of the system. Each metric also has an impact on the hardware resources. A careful examination of both approaches has to be conducted in order to gain a wider picture of the problem.

6.2 HW/SW Implementation Tradeoffs

We begin our tradeoff analysis with the examination of the hardware resources requirements of the three different matching metrics. As we saw in Chapter 3, the matching metrics have major differences when implemented on the FPGA. The Sum of absolute differences can easily be mapped on the device using limited resources. The Euclidean distance requires the implementation of multipliers which are usually mapped into dedicated DSP blocks. The Chi-squared distance metric is the most “expensive” metric because of the increased hardware requirements for the division. The differences in hardware requirements are listed again in Table 5.11 for one illustrative case. The complete results Table can be found in Chapter 3.

16 CHANNELS	**UTILIZATION (ZYNQ 7045)**		
16 SIG. PARALLELISM	**SAD**	**EUC**	**CHI-SQUARED**
REGISTERS	23.8K (5%)	24.7K (5%)	281K (64%)
LUTs	16.4K (7%)	17.6K (8%)	81.7K (37%)
DSPs	0	256 (28%)	256 (28%)

Table 6.1 Hardware requirements of the same configuration for three matching metrics.

It is immediately understandable that we have to select carefully the metric that matches our application. The chi-squared metric limits the increased processing capabilities of the FPGA greatly and should be used only in applications where the difference in quality is crucial. If we consider the quality results from our experiments in Chapter 4, we can study the cases where the implementation of the normalization metric is useful. As we saw in Chapter 4, the chi-squared metric provides results with an up to 10% improvement in quality. In others cases the difference is much more limited and would not justify the use of such an expensive hardware implementation.

Another important factor is the number of signatures used in a matching system. As we already know, the FPGA provides the resources for increased parallelism implementations. This means that a system with many signatures can easily be implemented on the device. Specifically, in our design, we saw in Chapter 3 that a big limiting factor in our design is the input/output throughput for the pixels to be matched. In a system with a low number of signatures the pixel throughput greatly increases since less clock cycles are needed to match all signatures with the pixel. The increased processing capabilities of the circuit are thus useless if throughput can’t be achieved. In the case of many signatures though, the required throughput drops resulting in a design that can in fact be implemented.
The above observation can be seen in Table 5.12 that lists throughput requirements for the same design and different total signatures.

SAD		8 signature parallelism
Channels: 16		
FPGA Clock		428 MHz
Speedup (Mean)	i3	48
	ARM	356
Number of Signatures	32	17.12 Gbps
	64	8.56 Gbps
	128	4.28 Gbps
	256	2.14 Gbps
	512	1.07 Gbps

Table 6.2 Throughput requirements for different total signatures

It is evident that an increased number in signatures can easily be implemented on the FPGA. This fact drives us to the need of examining the increase in signatures on the application side of our design. As we discussed in the previous chapter, an increased number of signatures does provide improved results quality-wise. We repeat in Table 5.13 the results for the same configurations, for one and multiple signatures. This improvement is meaningful for many applications and when we take into account the above mentioned impact on the hardware side of the system, it is evident that the FPGA is ideal for such implementations.

COURT					
1 Signature			3 Signatures		
SAD	EUC	X-2	SAD	EUC	X-2
55.88%	54.44%	67.48%	66.35%	67.82%	75.92%

TREES (minus grass)					
1 Signature			3 Signatures + Mean		
SAD	EUC	X-2	SAD	EUC	X-2
33.9%	28.39%	39.5%	44.72%	41.87%	45.31%

Table 6.3 Quality results for one and three signatures

The ability to compute efficiently matching metrics for a high number of signatures is very useful also in a number of applications where the identification of features in an image in high speed is crucial. We mentioned in the introductory section applications like quality inspection or food sorting which would benefit greatly from the above resulting in higher production speed and improved quality.

CHAPTER 7

CONCLUSION

Now that the discussion of this thesis has been completed, we will present the conclusions drawn from the work done and the results obtained. Some final thoughts on the project will also be discussed which will hopefully help the reader to understand the general picture of the project. Finally, we will briefly discuss how the work done here and the corresponding results can be utilized in other projects.

7.1 Conclusions

Hyperspectral imaging is a field that has seen an intense research interest in recent years. Applications that utilize this technology range from satellite based/airborne remote sensing and military target detection to industrial quality control, quality assessment and food inspection as well as lab applications in medicine and biophysics. The major drawback of this technology is the increased data processing requirements that challenge most common processing units. In this thesis, we discuss the implementation of a matching system for hyperspectral images with up to hundreds of spectral channels.
The realization of this system was reached accomplished with the use of a Field Programmable Gate Array circuit. These chips offer high performance by taking advantage of hardware parallelism and a configurable architecture that can be programmed and re-programmed accordingly.
In this work, we created and tested two different approaches of the hyperspectral matching system. On one side we described the system using C code and ran the code on a common Intel CPU and an embedded ARM processor. The same system was then developed in VHDL and ran on an FPGA. The results obtained from this implementation do show the great parallel processing advantages of the FPGA. We achieved performance results that are two to three orders of magnitude faster than the common CPU implementation and three orders of magnitude faster than that of an embedded processor. Maximum achievable speedup and parallelism on the FPGA is only limited by I/O data throughput. These restrictions provide a challenge for the designer to make an optimal use of the available capabilities in order to achieve best results.

Following the hardware/performance analysis, we focused on the analysis of hyperspectral images and the different matching techniques. The purpose of this analysis is very important as was shown by the experimental results. In this thesis, we tested three different matching metrics for the implementation of the pixel/signature matching. These metrics have very different hardware requirements that do greatly influence the overall achievable performance. Thus, it is important to perform an extensive analysis regarding qualitative results. In order to achieve this, we developed a model in MATLAB that quantified and measured the quality of each implementation. These results allowed us to perform a trade-off analysis of our design. We saw that more “expensive” hardware

implementations cannot be justified for most applications and alternative approaches have to be examined. The designer has to choose carefully between all available configurations in order to obtain the best results for each application.
It is important to mention the fact that a good analysis and description of a problem is crucial for the hardware engineer in order to take the design decisions that will allow him to satisfy all the requirements in the best possible way.
It is obvious that the development of a VHDL system is a very intensive and time consuming work that imposes great challenges for the designer. The advantages of this approach though are a specific and well defined system that results in a high performance processing unit that provides a solution to a range of applications limited by common processing units.

7.2 Future Work and Final Thoughts

During the development of the system in this thesis, we had in mind the greater picture of our work. As already mentioned several times in this text, the VHDL hyperspectral matching system is the core of a number of applications based on hyperspectral image data processing. Hence, we developed a parametric VHDL code and architecture on multiple levels that allows for arbitrary parallelism regarding most application and equipment specific variables. That is a variable number of signatures to be matched, an arbitrary number of spectral channels and a variable number of parallel pixel processing units. Finally, the modular design of our processing core allows a very easy interchangeability of different arithmetic units depending on application specifications.
This means that the whole system developed during this thesis can very easily be extended or become a part of a greater system for hyperspectral image matching applications.
Furthermore, modern FPGA devices are part of a greater system on chip that includes also traditional embedded processors and peripheral devices that can be programmed to implement a complete system with SW/HW co-design.

Specific areas of further research and development include:

- Integration of the developed processing kernel in a real system with a camera connection as well as communication setup.
- Interconnection of the FPGA with the embedded processor for co-processing. Exploration of interconnection with the outside world.
- System design for data output processing. Numerous applications and projects could be developed around the already built processing core.
- Finally, more advanced hyperspectral imaging algorithms on FPGA could be examined, e.g. regression based spectral matching.

It is evident that many areas of further development and research exist around the work done in this thesis. This is particularly satisfying since it is important for scientific work to continue through new projects and studies.

Thus, we believe that the current thesis is important not only as a scientific study on its own but rather as a small contribution in a larger project providing the impetus for further research and development by young researchers and students, bringing together people, different areas of study and knowledge with the ultimate goal of completing a project that will be an achievement not of a single unit but the combined effort of a large number of various members.

Science should not only serve individual people's goals but instead contribute to the universal progress of humanity. It is not ours to keep and it is worthless if kept locked away. Having this idea in mind we urge anyone interested to use our work and we would be more than happy if our little contribution to science found its way in new and promising projects. For this contribution we are certainly proud.

REFERENCES

[1] M. E. Schaepman, M. Jehle, A. Hueni, K. Meuleman, The APEX Team, and K. I. Itten, "The 4th generation imaging spectrometer APEX and its application in Earth observation," IEEE Transactions on Geoscience and Remote Sensing, 2012 (in preparation)

[2] „*Imaging Spectroscopy*“, (2012). [Online]. Available: http://www.apex-esa.org /content/imaging-spectroscopy

[3] *Introduction to Hyperspectral Imaging with TNTmpis®*, Randall B. Smith, Ph.D., Microimages Inc., 2012

[4] „Nikon MicroscopyU – Confocal Microscopy – Spectral Imaging and Linear Unmixing” [Online]. Available: http://www.microscopyu.com/articles/confocal/spectralimaging.html

[5] “What is hyperspectral imaging”, HySpex, [Online]. Available: http://www.hyspex.no/hyperspectral_imaging/

[6] Carlos González, Sergio Sánchez, Abel Paz, Javier Resano , Daniel Mozos, Antonio Plaza, (2012) Use of FPGA or GPU-based architectures for remotely sensed hyperspectral image processing. *INTEGRATION, the VLSI journal*. **46** (2013), 89-103.

[7] A. Plaza, J.A. Benediktsson, J. Boardman, J. Brazile, L. Bruzzone, G. CampsValls, J. Chanussot, M. Fauvel, P. Gamba, J. Gualtieri, M. Marconcini, J.C. Tilton, G. Trianni, Recent advances in techniques for hyperspectral image processing, Remote Sensing of Environment 113 (2009) 110–122.

[8] A. Plaza, C.-I. Chang, High Performance Computing in Remote Sensing, Taylor & Francis, Boca Raton, FL, 2007

[9] A. Plaza, D. Valencia, J. Plaza, P. Martinez, Commodity cluster-based parallel processing of hyperspectral Imagery, Journal of Parallel and Distributed Computing 66 (2006) 345–358

[10] “Intel® Core™ i3-3110M Processor” [Online]. Available: http://ark.intel.com/products/65700/Intel-Core-i3-3110M-Processor-3M-Cache-2_40-GHz

[11] “Zynq-7000 All Programmable SoC” [Online]. Available: http://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html#productTable

[12] “ZedBoard development board” [Online]. Available: http://zedboard.org/sites/default/files/product_briefs/PB-AES-Z7EV-7Z020_G-v12.pdf

[13] Patterson, David A. and John L. Hennessy. *Computer Organization & Design.5th* ed. Morgan Kaufmann Publishers, 2013.

[14] *XST User guide*,v11.3, Xilinx, Sept.2009

[15] “Introduction to Parallel Computing”, Blaise Barney, Lawrence Livermore National Laboratory. [Online]. Available: https://computing.llnl.gov/tutorials/parallel_comp/

[16] Blelloch, Guy (1996). "Programming Parallel Algorithms" (PDF). *Communications of the ACM* 39 (3): 85–97. doi:10.1145/227234.227246

[17] “ GCC, the GNU Compiler Collection”, Free Software Foundation, Inc., 1988-2016

[18] David F. Bacon, Susan L. Graham, and Oliver J. Sharp. *Compiler transformations for high-performance computing.* Report No. UCB/CSD 93/781, Computer Science Division-EECS, University of California, Berkeley, Berkeley, California 94720, November 1993

[19] “Designing High-Performance Video Systems with the Zynq-7000 All Programmable SoC Using IP Integrator”, James Lucero, Bob Slous, XAPP1205 (v1.0), Xilinx, March 20014

[20] “UltraScale Architecture and Product Overview”,DS890 (v2.8), Xilinx, June 2016

[21]” Field Programmable Gate Array (FPGA)”. [Online]. Available: http://www.xilinx.com/training/fpga/fpga-field-programmable-gate-array.htm

[22] Clive Maxfield , *FPGAs: Instant Access*. Newnes; 1st edition (August 11, 2008)

[23] Χαράλαμπος Ν. Σιδηρόπουλος, “*Development of a Design Framework for Power/Energy consumption estimation in heterogeneous FPGA architectures”*, Diploma Thesis, National Technical University of Athens, July 2010
[24] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*, ISBN 0-7923-8460-1
[25] Bryan Mealy, Fabrizio Tappero, *Free Range VHDL. The no-frills guide to writing powerful code for your digital implementations,* Creative Commons Attribution-ShareAlike Unported License, 2015
[26] David Pellerin , *VHDL Made Easy!*, Prentice Hall PTR; 61526th edition (September 3, 1996)
[27] “Xilinx Zynq-7000 All Programmable SoC ZC706 Evaluation Kit”, [Online]. Available: http://www.xilinx.com/products/boards-and-kits/ek-z7-zc706-g.html#hardware