National Technical University of Athens
School of Electrical & Computer Engineering
Division of Computer Science

Efficient Acceleration of Complex DSP Applications on Reconfigurable Devices: Platform and Architecture Optimizations

Ph.D. Thesis

Ioannis G. Stratakos

Athens, December 2023

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργατήριο Μικροϋπολογιστών και Ψηφιακών Συστημάτων VLSI

Αποτελεσματική Επιτάχυνση Πολύπλοκων Εφαρμογών Ψηφιακής Επεξεργασίας Σημάτων σε Επαναδιαμορφώσιμες Συσκευές: Βελτιστοποιήσεις Πλατφόρμας και Αρχιτεκτονικής

Διδακτορική Διατριβή
του
Ιωάννη Στρατάκου

Συμβουλευτική Επιτροπή: Δημήτριος Σούντρης (Επιβλέπων)
Κιαμάλ Πεκμεστζή
Διονύσιος Ρεΐσης

Εγκρίθηκε από την επταμελή εξεταστική επιτροπή την 15η Δεκεμβρίου 2023.

.....	.....	.....
Δημήτριος Σούντρης	Κιαμάλ Πεκμεστζή	Διονύσιος Ρεΐσης
Καθηγητής Ε.Μ.Π.	Ομότιμος Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Κ.Π.Α.
.....	.....	
Ηρακλής Αβραμόπουλος	Αθανάσιος Παναγόπουλος	
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	
.....	.....	
Σωτήριος Ξύδης	Νίκολαος Πλέρος	
Επίκουρος Καθηγητής Ε.Μ.Π.	Καθηγητής Α.Π.Θ.	

Αθήνα, Δεκέμβριος 2023

.....
Ιωάννης Γ. Στρατάκος
Πτυχιούχος Φυσικών Επιστημών Ε.Κ.Π.Α.
Διπλωματούχος Ηλεκτρονικής & Αυτοματισμού Ε.Κ.Π.Α.
Διδάκτωρ Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Η διδακκοτιρή διατριβή συγχρηματοδοτήθηκε από την Ευρωπαϊκή Επιτροπή μέσω των έργων H2020 5G-PHOS και H2020 INT5GENT.

Abstract

In recent years, there has been a remarkable surge in the field of embedded systems, especially in the consumer electronics sector. The growing demand for high-performance and energy-efficient systems has motivated researchers to come up with innovative design techniques to meet these challenging requirements. Among various aspects, streaming Digital Signal Processing (DSP) has gained particular attention, involving complex mathematical computations such as matrix inversions, filtering, and basic arithmetic operations. As DSP continues to gain importance in various devices, certain factors like time-to-market and flexibility for late design changes have become critical considerations. Software-based solutions provide flexibility and the ability to make adjustments even in the later stages, but they often lag behind hardware in terms of performance due to the limitations of serial processing. On the other hand, developing custom Application-Specific Integrated Circuits (ASICs) can be a time-consuming process and lacks reconfigurability once the fabrication is completed.

This dissertation revolves around accelerating demanding and complex digital signal processing applications on re-configurable devices, with a particular emphasis on SoC-FPGAs. The applications covered in this research span two domains: image/video processing and telecommunications. In the image/video processing domain, the main focus is on accelerating 1) Medical Imaging and 2) Vision-based navigation applications for space deployments. In the telecom domain, the research centers on low-level physical layer processing, specifically targeting the next-generation 5G/B5G mobile networks. These applications share common characteristics, requiring strict adherence to latency requirements while keeping power consumption at a minimum. To achieve these objectives, extensive Design Space Exploration (DSE) is carried out, and advanced design techniques are employed. The utilization of the unique features of the underlying SoC-FPGA device further enhances the effectiveness of the proposed solutions.

Keywords: Embedded Systems, SoC-FPGA, Voltage Scaling, Image/Video Digital Signal Processing, Telecommunications Digital Signal Processing, 5G/B5G Networks, Hardware/Software Co-design, Hardware Accelerators

Περίληψη

Τα τελευταία χρόνια έχει σημειωθεί αξιοσημείωτη εξέλιξη στον τομέα των ενσωματωμένων συστημάτων. Η αυξανόμενη ζήτηση για συστήματα υψηλών επιδόσεων και ενεργειακά αποδοτικά, έχει παρακινήσει τους ερευνητές να επινοήσουν καινοτόμες τεχνικές σχεδιασμού για να ανταποκριθούν σε αυτές τις δύσκολες απαιτήσεις. Καθώς η ψηφιακή επεξεργασία σήματος, η οποία περιλαμβάνει πολύπλοκους μαθηματικούς, συνεχίζει να αποκτά σημασία σε διάφορες συσκευές, ορισμένοι παράγοντες, όπως ο χρόνος διάθεσης στην αγορά και η ευελιξία για καθυστερημένες αλλαγές στη σχεδίαση, έχουν γίνει κρίσιμοι παράγοντες. Οι λύσεις που βασίζονται σε λογισμικό παρέχουν ευελιξία και δυνατότητα προσαρμογών ακόμη και σε μεταγενέστερα στάδια, αλλά συχνά υστερούν σε απόδοση έναντι του υλικού λόγω των περιορισμών της σειριακής επεξεργασίας. Από την άλλη πλευρά, η ανάπτυξη ολοκληρωμένων κυκλωμάτων ειδικών εφαρμογών (ASIC) μπορεί να είναι χρονοβόρα διαδικασία και στερείται δυνατότητας αναδιαμόρφωσης μόλις ολοκληρωθεί η κατασκευή.

Η παρούσα διατριβή επικεντρώνεται στην επιτάχυνση απαιτητικών και πολύπλοκων εφαρμογών ψηφιακής επεξεργασίας σήματος σε επαναδιαμορφώσιμες συσκευές, με ιδιαίτερη έμφαση στα SoC-FPGAs. Οι εφαρμογές που μελετώνται καλύπτουν δύο τομείς: επεξεργασία εικόνας/βίντεο και τηλεπικοινωνίες. Στον τομέα της επεξεργασίας εικόνας/βίντεο, η έμφαση δίνεται 1) στην επιτάχυνση εφαρμογών ιατρικής απεικόνισης και 2) αυτόνομης πλοήγησης με βάση την όραση για διαστημικές εφαρμογές. Στον τομέα των τηλεπικοινωνιών, η έρευνα επικεντρώνεται στην επεξεργασία φυσικού επιπέδου, με ιδιαίτερη έμφαση τα κινητά δίκτυα επόμενης γενιάς 5G/B5G. Αυτές οι εφαρμογές έχουν κοινά χαρακτηριστικά, απαιτώντας αυστηρή τήρηση των απαιτήσεων καθυστέρησης, διατηρώντας παράλληλα την κατανάλωση ισχύος στο ελάχιστο. Για την επίτευξη αυτών των στόχων, πραγματοποιείται εκτεταμένη εξερεύνηση του χώρου σχεδίασης και χρησιμοποιούνται προηγμένες τεχνικές σχεδίασης. Η αξιοποίηση των μοναδικών χαρακτηριστικών της υποκείμενης συσκευής SoC-FPGA ενισχύει περαιτέρω την αποτελεσματικότητα των προτεινόμενων λύσεων.

Λέξεις Κλειδιά: Ενσωματωμένα Συστήματα, SoC-FPGA, Κλιμάκωση Τάσης, Ψηφιακή Επεξεργασία Σήματος Εικόνας/Βίντεο, Ψηφιακή Επεξεργασία Σήματος Τηλεπικοινωνιών, Δίκτυα 5G/B5G, Συν-σχεδιασμός Υλικού/Λογισμικού, Επιταχυντές Υλικού

Acknowledgments

Firstly, I want to extend my sincere thanks to my supervisor, Prof. Dimitrios Soudris. In 2015, Prof. Soudris welcomed me to Microlab for my Master's thesis, despite my coming from a different university. At first, I felt somewhat daunted, yet his welcoming demeanor and keen interest quickly put me at ease and prepared me to embark on this academic endeavor. I did my best to repay his trust and kindness, and I like to think that I achieved this when he offered me the chance to stay on at Microlab to pursue a PhD degree. His encouragement and guidance, both personally and professionally, has been constant over the years, extending to the moment I am writing this, and I hope this bond will continue for many years to come.

Next, I must also extend my thanks to Prof. Dionysios Reisis from NKUA. Unplanned as it was, Prof. Reisis has coincidentally been involved in all phases of my academic life, from my Bachelor's degree, through my Master's, and into my PhD. He provided favorable feedback to Prof. Soudris about me, influencing my acceptance to Microlab for the Master thesis.

It goes without saying that my thanks also extend to the individuals directly involved in my day-to-day activities during my doctoral studies. From day one at Microlab, two individuals were instrumental in my journey. Assistant Prof. George Lentaris, initially a Post-Doctoral researcher, offered continual guidance during my PhD. Konstantinos Maragos, a fellow PhD student at the start, collaborated closely with me during our concurrent studies. I'm deeply thankful for their guidance and teamwork, which enriched my PhD experience and broadened my knowledge. My best wishes to both.

The journey I've had throughout these years would have been significantly different without the collective presence of Microlab's past and current members. They have made this experience memorable and ensured a positive working environment. With that in mind, I would first like to express my gratitude to the individuals with whom I embarked on this academic journey: Dimosthenis, Konstantina, Vasilis, and Harris. Certainly, I cannot overlook the individuals with whom I shared many hours, both inside and/or outside Microlab. These are Manolis, Ioannis, Achilleas, Christos, Aggelos and George. Additionally, it's important to recognize Zefi, our lab's secretary, who was al-

ways there to talk and support us. Lastly, a heartfelt thank you to George and Kostas, former members of Microlab, who are always ready to catch up whenever the chance arises.

Throughout all these years, enduring without the presence of good friends would have been quite unbearable. With that being said, I want to express my gratitude to Filippos, Antonis, Maria, Areti and Thodoris. Their presence has been and continues to be a refreshing break from life's daily challenges, and I sincerely wish them all the best in their lives.

Last but absolutely not least, I need to express my deep appreciation and love for my family. It's their ongoing encouragement and support through both my personal and academic challenges that have provided me with the resilience to keep striving towards my goals. Therefore, a heartfelt THANK YOU goes out to my parents, George and Georgia, my sisters, Stavroula and Anastasia, and my brother-in-law, Methodios. You all hold a special place in my heart, and I deeply respect each of you.

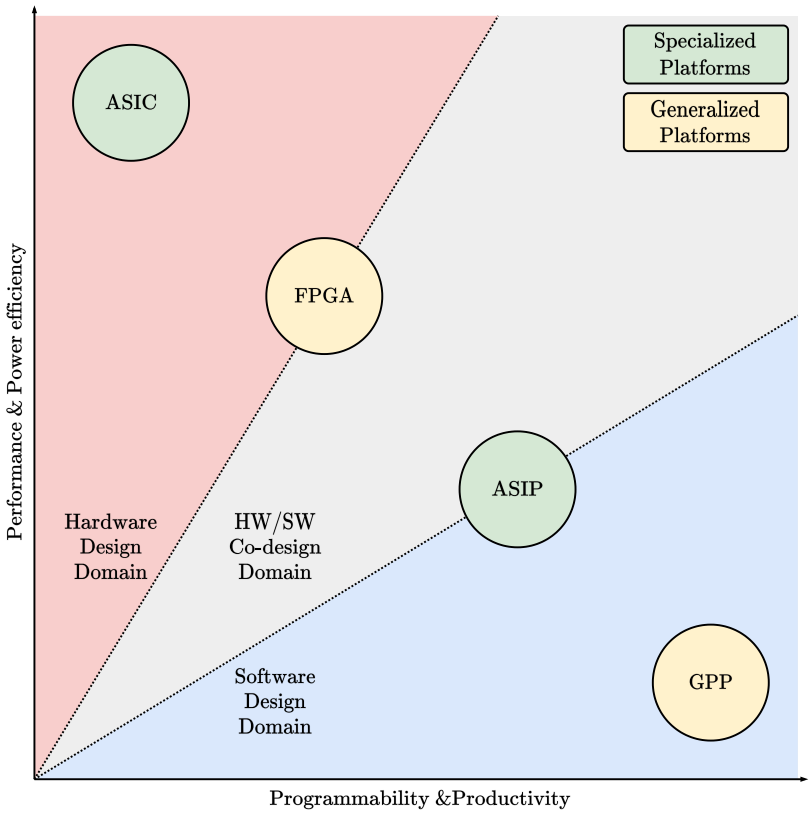
Εκτεταμένη Ελληνική Περίληψη

1 Εισαγωγή

Ενσωματωμένα Υπολογιστικά Συστήματα

Ένα ενσωματωμένο σύστημα είναι ένα ειδικό υπολογιστικό σύστημα που ενσωματώνεται σε ένα μεγαλύτερο προϊόν για την εκτέλεση συγκεκριμένων καθηκόντων σε έναν επιλεγμένο τομέα εφαρμογής. Η αποδοτικότητα όσον αφορά το μέγεθος, το κόστος, η κατανάλωση ενέργειας και ο χρόνος απόκρισης αποτελούν κρίσιμες απαιτήσεις για τα ενσωματωμένα συστήματα. Επιπλέον, πρέπει να παρουσιάζουν υψηλό επίπεδο αξιοπιστίας, που σημαίνει ότι οποιαδήποτε δυσλειτουργία δεν είναι ανεκτή. Με την πάροδο του χρόνου, καθώς τα ενσωματωμένα συστήματα επιτυγχάνουν υψηλότερες επιδόσεις και αποδοτικότητα, ενσωματώνουν πιο εξελιγμένους αλγορίθμους, με αποτέλεσμα να εξελίσσονται σε πολύπλοκα συστήματα και επεκτείνονται σε διάφορους τομείς εφαρμογών, επιτρέποντας την εφαρμογή καινοτόμων τεχνολογιών. Ως αποτέλεσμα αυτών των παραγόντων, έχει γίνει όλο και πιο δύσκολο να διαφοροποιηθούν τα ενσωματωμένα υπολογιστικά συστήματα από τα συστήματα γενικής χρήσης.

Η τάση για αυξανόμενη πολυπλοκότητα συνοδευόμενη από τις συνεχείς εξελίξεις στην τεχνολογία ημιαγωγών, καθιστά τους ενσωματωμένους υπολογιστές ευρέως διαθέσιμους σε σχετικά χαμηλό κόστος. Έτσι, τα ενσωματωμένα υπολογιστικά συστήματα έχουν γίνει ευρέως διαδεδομένα, με στόχο την αυτοματοποίηση και τη βελτιστοποίηση διαφόρων εργασιών. Η εμφάνιση εννοιών όπως το Διαδίκτυο των Πραγμάτων (Internet of Things - IoT) και το Edge Computing έχουν αυξήσει ακόμη περισσότερο τις υπολογιστικές και επικοινωνιακές απαιτήσεις στα ενσωματωμένα συστήματα [1]. Ταυτόχρονα, η σημαντική αύξηση των απαιτητικών εφαρμογών και των προηγμένων αλγορίθμων σε τομείς όπως η Ψηφιακή Επεξεργασία Σήματος (Digital Signal Processing - DSP) υποδηλώνει μια νέα εποχή για τα ενσωματωμένα υπολογιστικά συστήματα. Οι συμβατικοί επεξεργαστές γενικής χρήσης, όπως οι κεντρικές μονάδες επεξεργασίας (Central Processing Unit - CPU), δεν έχουν την υπολογιστική ικανότητα να χειριστούν αυτά τα απαιτητικές εργασίες, με αποτέλεσμα να υπάρχουν περιορισμοί στις επιδόσεις [2]. Λαμβάνοντας υπόψη τους περιορισμούς της χαμηλής κατανάλωσης



Σχήμα 1: Σύγκριση μεταξύ υπολογιστικών πλατφορμών.

ενέργειας, προκύπτει ένα παράδοξο: στοχεύουμε σε υψηλές υπολογιστικές επιδόσεις και ταυτόχρονα χρησιμοποιούμε μια συσκευή χαμηλής κατανάλωσης ισχύος. Ταυτόχρονα, ο τεράστιος όγκος δεδομένων που δημιουργείται κατακλύζει τα συμβατικά υπολογιστικά συστήματα, με αποτέλεσμα την ανάγκη ενσωμάτωσης ειδικών επιταχυντών υπολογισμού [3]. Τέτοιες συσκευές είναι οι συστοιχίες προγραμματιζόμενων πυλών (Field-Programmable Gate Arrays - FPGAs).

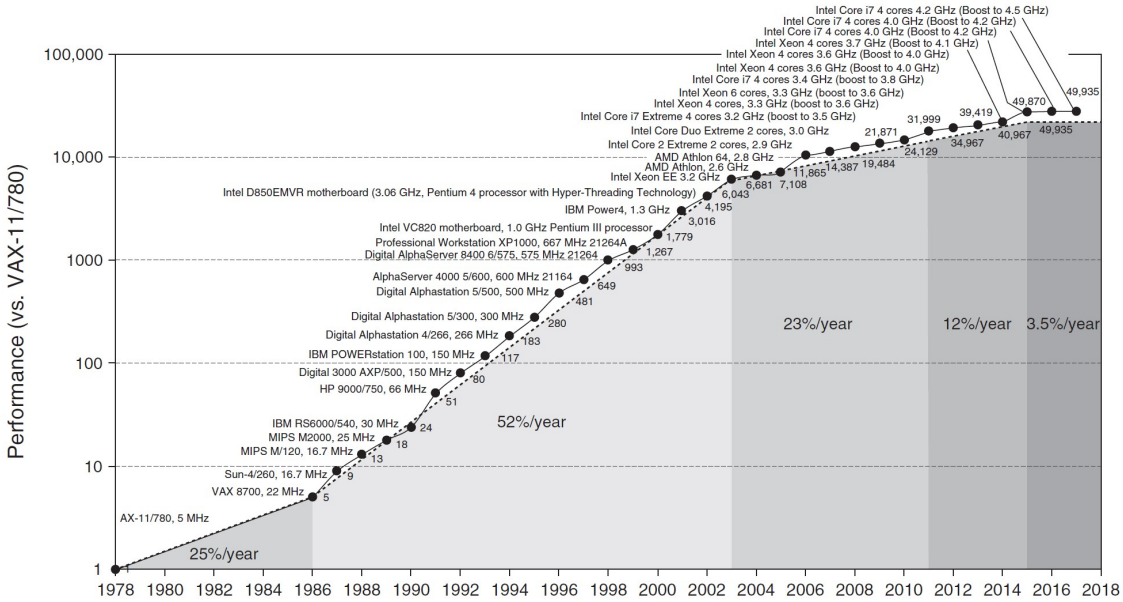
Σε σύγκριση με άλλες υπολογιστικές πλατφόρμες, το Σχήμα 1 καταδεικνύει ότι οι συσκευές FPGA καταλαμβάνουν μια ενδιάμεση θέση μεταξύ των εξειδικευμένων συσκευών υλικού (ASIC) και των πιο φιλικών προς την ανάπτυξη εφαρμογών, αλλά συνήθως πιο αργών, συσκευών υλικού, όπως οι CPUs και οι επεξεργαστές ειδικών εντολών (Application-Specific Instruction Set Processor - ASIP). Τα ASICs προσφέρουν μια βελτιστοποιημένη λύση υλικού προσαρμοσμένη είτε σε έναν τομέα εφαρμογής είτε σε μια μεμονωμένη εφαρμογή. Ως αποτέλεσμα, παρέχουν καλύτερες επιδόσεις και χαμηλότερη κατανάλψησ ισχύος σε σύγκριση με τα FPGA ή τις υλοποιήσεις που βασίζονται σε λογισμικό. Εναλλακτικά, οι λύσεις

λογισμικού για υπολογιστικές πλατφόρμες γενικού σκοπού σχεδιάζονται για να εκτελούνται σε επεξεργαστές γενικής χρήσης και έτσι είναι πολύ πιο απλές και γρήγορες στην ανάπτυξη. Το λογισμικό μπορεί εύκολα να ενημερωθεί ή να βελτιωθεί σε περίπτωση που προκύψουν σφάλματα στη σχεδίαση ή αναπτυχθούν νέες βελτιωμένες εφαρμογές ή υλοποιήσεις. Είτε βλέπουμε τα FPGA ως μια πιο ευέλικτη, αναδιαμορφώσιμη έκδοση ενός ASIC, είτε ως μια πλατφόρμα υψηλότερων επιδόσεων και χαμηλότερης κατανάλωσης ενέργειας για συστήματα βασισμένα σε λογισμικό, τα συστήματα βασισμένα σε FPGA έχουν τα δικά τους πλεονεκτήματα και προκλήσεις.

Επαναδιαμορφώσιμες Συσκευές - FPGAs

Τα βασικά χαρακτηριστικά των FPGAs περιλαμβάνουν την εγγενή παράλληλη αρχιτεκτονική τους, που διευκολύνει την ανάπτυξη εξαιρετικά παράλληλων κυκλωμάτων με εντυπωσιακούς ρυθμούς επεξεργασίας. Επιπλέον, η χαμηλή συχνότητα λειτουργίας τους συμβάλλει στη μειωμένη κατανάλωση ενέργειας, ενώ η εκ νέου επαναδιαμόρφωση/επαναπρογραμματισμός παρέχει αυξημένη ευελιξία και δυνατότητα επαναχρησιμοποίησης σε ένα ευρύ φάσμα εφαρμογών και λειτουργιών. Αυτά τα χαρακτηριστικά καθιστούν τα FPGAs μία πολλά υποσχόμενη λύση για την αντιμετώπιση των αυξανόμενων απαιτήσεων για επιδόσεις στα κέντρα δεδομένων και στις ενσωματωμένες εφαρμογές [2,4]. Παρά το γεγονός ότι απαιτούν περισσότερη ενέργεια σε σύγκριση με τα ASICs [5], τα FPGA θεωρούνται οι βέλτιστοι εξειδικευμένοι επιταχυντές σε πολλές περιπτώσεις, ξεπερνώντας τις CPU και τις GPU [2] σε επιδόσεις. Σε συνεχώς αναπτυσσόμενους τομείς εφαρμογών, όπως η Τεχνητή Νοημοσύνη (Artificial Intelligence - AI)/η Μηχανική Μάθηση (Machine Learing - ML) και τα κινητά δίκτυα τηλεπικοινωνιών (5G/B5G), τα FPGA ξεχωρίζουν λόγω της δυνατότητας επαναδιαμόρφωσής τους. Αυτό τους επιτρέπει την ευελιξία να διατηρούνται ενημερωμένα με βάση την εξελισσόμενη αρχιτεκτονική και την ενσωμάτωση χαρακτηριστικών σε αυτούς τους τομείς, να προσαρμόζονται γρήγορα στις μεταβαλλόμενες απαιτήσεις επεξεργασίας, να προσφέρουν λύσεις σε επίπεδο συστήματος με βελτιωμένη απόδοση και να εξασφαλίζουν γρήγορο χρόνο εμφάνισης προϊόντων στην αγορά.

Καθώς αυξήθηκε η ζήτηση για μεγαλύτερη απόδοση επεξεργασίας, εν μέρει λόγω της επιβράδυνσης της αύξησης των επιδόσεων των επεξεργαστών (Σχήμα 2), το 2011 εμφανίστηκε μια νέα τάση στην αγορά FPGA, με την εισαγωγή των FPGAs System-on-Chip (SoC). Τα SoC-FPGAs είναι ετερογενείς πλατφόρμες που συνδυάζουν την παραδοσιακή συσκευή FPGA με διάφορες μονάδες επεξεργασίας, όπως CPUs, GPUs, κλπ. σε ένα μόνο ςηιπ. Εκτός από τις προαναφερθείσες μονάδες επεξεργασίας, τα τσιπ FPGA μπορούν να ενσωματώσουν διάφορα άλλα στοιχεία υψηλής απόδοσης, όπως υψηλής ταχύτητας σειριακούς πομποδέκτες (που υπερβαίνουν τα 10 Gbps), προσαρμόσιμες διεπαφές εξωτερικής μνήμης και ενσωματωμένες διεπαφές όπως SPI, USB, Ethernet και PCI-express. Αυτή η ενσωμάτωση

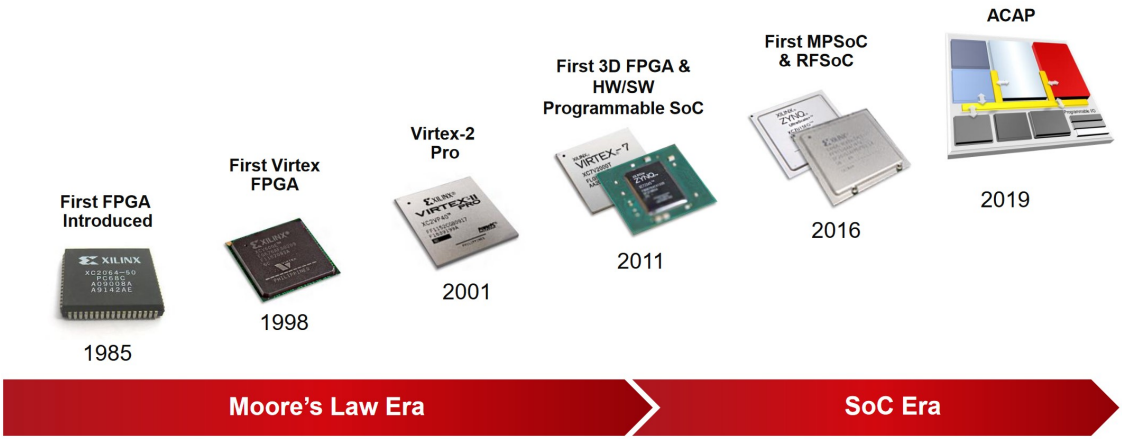


Σχήμα 2: Αύξηση των επιδόσεων των επεξεργαστών σε βάθος 40 ετών [6].

προσφέρει πολλά πλεονεκτήματα, όπως μειωμένη κατανάλωση ενέργειας, βελτιωμένη αξιοπιστία, ακόμη ταχύτερο χρόνο διάθεσης στην αγορά λόγω της ενοποιημένης ανάπτυξης και αποσφαλμάτωσης, μικρότερο μέγεθος και βάρος, καθώς πολλαπλά στοιχεία ενσωματώνονται σε ένα ενιαίο τσιπ, και το σημαντικότερο, βελτιωμένη απόδοση και προσαρμοστικότητα μέσω της αυξημένης ετερογένειας και του αποτελεσματικού συσχεδιασμού Υλικού/Λογισμικού (Hardware/Software Co-design - HW/SW Co-design) [7]. Τα SoC-FPGA έχουν γίνει ολοένα και πιο δημοφιλή στον κόσμο των ενσωματωμένων συστημάτων λόγω της ικανότητάς τους να βελτιώνουν σημαντικά την απόδοση και ενεργειακή αποδοτικότητα των παραδοσιακών συστημάτων που βασίζονται μόνο σε CPU [8]. Αυτή η τάση ανάπτυξης ακόμη πιο ετερογενών πλατφορμών με βάση τα FPGA συνεχίζεται ακόμη και τώρα, με την Xilinx (που τώρα ανήκει στην AMD) να είναι ο ηγέτης (Σχήμα 3).

Πεδία Εφαρμογής & Συμβολή της Διατριβής

Σε αυτή τη διδακτορική διατριβή, η εστίασή μας έγκειται στην επιτάχυνση απαιτητικών και πολύπλοκων εφαρμογών DSP σε επαναδιαμορφώσιμες συσκευές, ιδιαίτερα σε SoC-FPGA. Οι εφαρμογές που εξετάζονται σε αυτή τη διατριβή προέρχονται από τον τομέα της επεξεργασίας εικόνας/βίντεο και τον τομέα των τηλεπικοινωνιών. Στον τομέα της επεξεργασίας εικόνας/βίντεο, εστιάζουμε ιδιαίτερα στην επιτάχυνση μιας εφαρμογής ιατρικής απεικόνισης



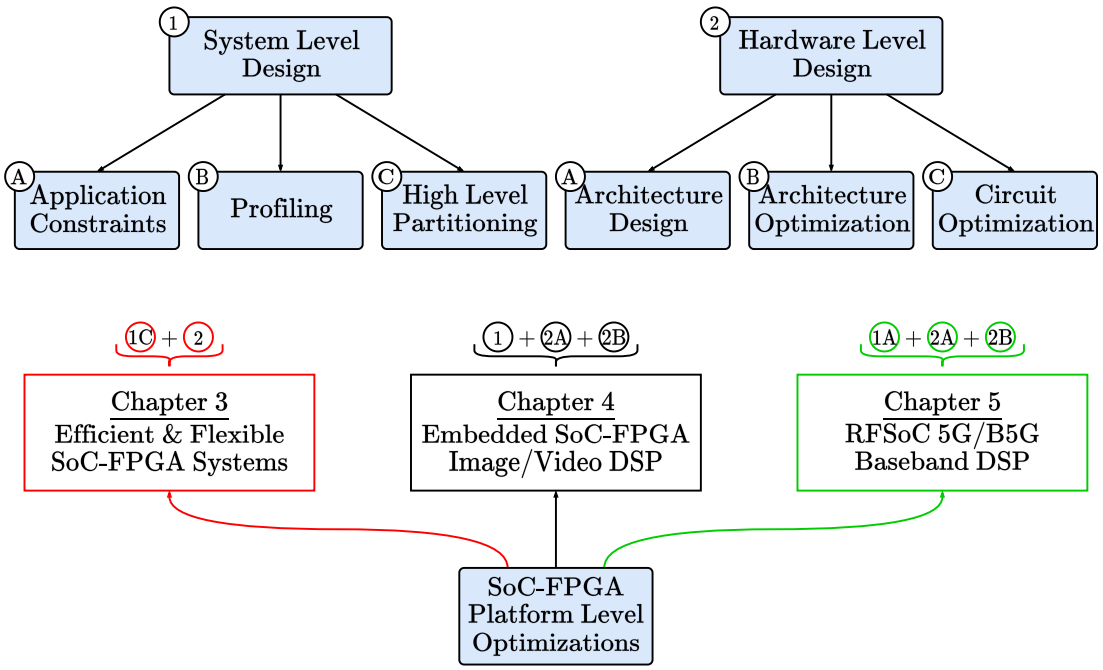
Σχήμα 3: Εξέλιξη των FPGA: από FPGA σε SoC σε ACAP [9]

και μιας εφαρμογής πλοήγησης με βάση την όραση για διαστημικές εφαρμογές. Από την άλλη πλευρά, στον τομέα των τηλεπικοινωνιών, επικεντρωνόμαστε στην επεξεργασία φυσικού επιπέδου (Physical Layer - PHY), στοχεύοντας ιδιαίτερα στα κινητά δίκτυα επόμενης γενιάς 5G/B5G. Όλες αυτές οι εφαρμογές έχουν αυστηρές προδιαγραφές καθυστέρησης και απαιτούν ελάχιστη κατανάλωση ενέργειας.

Για να επιτύχουμε τους στόχους μας, διεξάγουμε διεξοδική εξερεύνηση του χώρου σχεδίασης (Design Space Exploration - DSE) και εφαρμόζουμε προηγμένες τεχνικές σχεδίασης. Επιπλέον, λαμβάνουμε υπόψη τα μοναδικά χαρακτηριστικά της υποκείμενης συσκευής SoC-FPGA για να βελτιώσουμε περαιτέρω τις λύσεις μας. Το Σχήμα 4 παρέχει μια επισκόπηση της έρευνας που διεξήχθη στην παρούσα διατριβή.

Οι κύριες συνεισφορές της παρούσας διατριβής μπορούν να συνοψιστούν ως εξής:

> **Ενότητα 2**: Στην ενότητα αυτή παρουσιάζουμε μια σχεδιαστική προσέγγιση για ενσωματωμένες εφαρμογές DSP που απευθύνονται σε SoC-FPGAs. Αρχικά επικεντρωνόμαστε στη βελτιστοποίηση της διαδικασίας σχεδίασης σε διάφορα επίπεδα, ξεκινώντας από το αλγοριθμικό επίπεδο και επεκτείνεται μέχρι το κυκλωματικό επίπεδο. Στο επόμενο μέρος, παρουσιάζουμε μια συστηματική διαδικασία για τον προσδιορισμό των ιδανικών τάσεων τροφοδοσίας, εξασφαλίζοντας τη συμμόρφωση με τις ονομαστικές συνθήκες λειτουργίας που καθορίζονται από τα εργαλεία του κατασκευαστή. Τέλος, προτείνουμε μια λογική διαίρεση μιας ενιαίας συσκευής SoC-FPGA, επιτρέποντας την ανάπτυξη και εκτέλεση πολλαπλών διαφορετικών εφαρμογών στον ίδιο ευρύτερο τομέα (π.χ. computing@Edge). Η προτεινόμενη μεθοδολογία σχεδιασμού μας επιτρέπει την ανεξάρτητη εκτέλεση κάθε εφαρμογής ή τη συμπληρωματική χρήση



Σχήμα 4: Επισκόπηση της διδακτορικής διατριβής.

τους. Απώτερος στόχος είναι η ενίσχυση της αποδοτικότητας των εφαρμογών που αναπτύσσονται και η διευκόλυνση της ευέλικτης εκτέλεσης πολλαπλών εφαρμογών σε μια ενιαία συσκευή SoC-FPGA.

Ενότητα 3: Στην ενότητα αυτή παρουσιάζονται επιταχυντές υλικού (Hardware Accelerators) υψηλής απόδοσης σχεδιασμένοι ειδικά για εφαρμογές εικόνας και βίντεο. Η έμφαση δίνεται σε δύο διακριτούς τομείς: 1) Ιατρική απεικόνιση και 2) πλοήγηση με βάση την όραση στο διάστημα. Αυτοί οι τομείς επιβάλλουν αυστηρούς περιορισμούς υλοποίησης, απαιτώντας λειτουργία σε πραγματικό χρόνο και χαμηλή κατανάλωση ισχύος/ενέργειας. Επιπλέον, η εκτεταμένη χρήση βαθιά ενσωματωμένων συσκευών σε αυτούς τους τομείς εισάγει αξιοσημείωτες προκλήσεις υλοποίησης. Στην περίπτωση της εφαρμογής ιατρικής απεικόνισης, διερευνούμε μία εφαρμογή επεξεργασίας Image Registration (IR), ενώ για την πλοήγηση με βάση την όραση, η εφαρμογή είναι ο εντοπισμός της πόζας (6D Pose Tracking) μη συνεργάσιμων αντικειμένων στο διάστημα. Και για τις δύο περιπτώσεις, διεξάγουμε μια εις βάθος ανάλυση των εφαρμογών και περιγράφουμε αποδοτικά συστήματα HW/SW για συσκευές SoC-FPGA. Επιπλέον, παρουσιάζουμε βελτιστοποιήσεις σε επίπεδο υλικού για την περαιτέρω βελτίωση των επιδόσεων. Τέλος, παρέχονται ολοκληρωμένα αποτελέσματα επιδόσεων, συνοδευόμενα από συγκρίσεις με ενσωματωμένες CPU γενικού σκοπού, επιτρέποντας την ενδελεχή

αξιολόγηση των προτεινόμενων προσεγγίσεων.

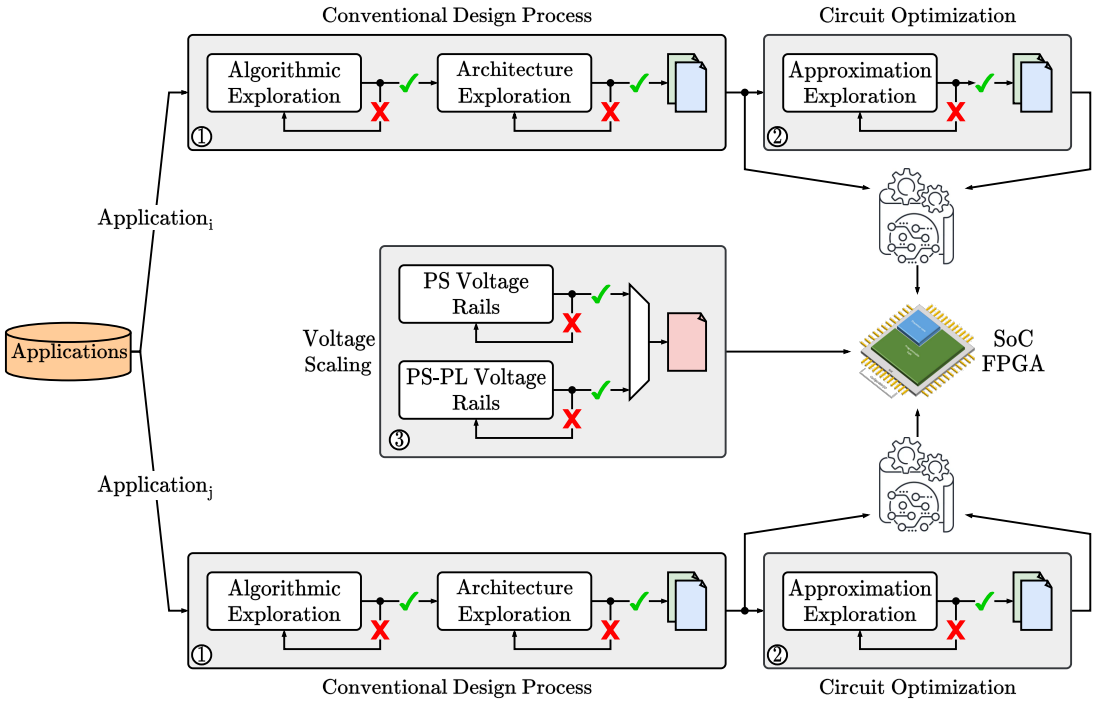
Ενότητα 4: Στην ενότητα αυτή παρουσιάζουμε ένα proof-of-concept επεξεργαστή υλικού βασικής ζώνης με Ορθογώνια Πολυπλεξία Διαίρεσης Συχνότητας (Orthogonal Frequency-Division Multiplexing - OFDM) που στοχεύει σε κινητά δίκτυα 5G/B5G και βασίζεται στις έννοιες Centralized-Radio Access Network (C-RAN) και Analog Radio-over-Fiber (A-RoF). Η έμφαση δίνεται στην ανάπτυξη αποδοτικών αλυσίδων επεξεργασίας δεδομένων μετάδοσης (Tx) και λήψης (Rx) στο υλικού που χρησιμοποιούν μετάδοση OFDM, σε μία μόνο συσκευή SoC-FPGA, την Xilinx RFSoC-FPGA. Συζητάμε την αρχιτεκτονική υλικού των βασικών υποσυστημάτων τόσο στην αλυσίδων επεξεργασίας δεδομένων Tx όσο και στην Rx, τονίζοντας την αποτελεσματική χρήση εξειδικευμένων μπλοκ επεξεργασίας ενσωματωμένων στα RFSoC-FPGA. Επιπλέον, διερευνούμε τις δυνατότητες παράλληλης ανάπτυξης πολλαπλών ζευγών αλυσίδων επεξεργασίας Tx/Rx για να καταστεί δυνατή η μετάδοση πολλαπλών σημάτων OFDM ταυτόχρονα, βελτιώνοντας έτσι το εύρος ζώνης ανά χρήστη ή εξυπηρετώντας ταυτόχρονα πολλούς χρήστες. Τέλος, παρέχονται μετρικές επιδόσεων και χρήσης πόρων, καθώς και δύο ενδεικτικές περιπτώσεις δοκιμών σε πειράματα με πραγματική κίνηση που παρουσιάζουν την επιτυχή ανάπτυξη και επίτευξη των στόχων με τις αναπτυχθείσες αλυσίδες επεξεργασίας δεδομένων Tx/Rx-OFDM στην πλατφόρμα RFSoC.

Τέλος, η Ενότητα 5 ολοκληρώνει την παρούσα διατριβή συνοψίζοντας τα αποτελέσματα που παρουσιάστηκαν και συζητά τις μελλοντικές επεκτάσεις της παρούσας εργασίας.

2 Συνεργατική Προσέγγιση Σχεδίασης για Πλατφόρμες SoC-FPGA

2.1 Εισαγωγή

Στην ενότητα αυτή παρουσιάζεται μια συνεργατική προσέγγιση σχεδιασμού για την αύξηση της αποδοτικότητας και της ευελιξίας σε πλατφόρμες SoC-FPGA. Η πρωταρχική εστίαση είναι στην ενίσχυση των εργασιών ψηφιακής επεξεργασίας σήματος εντός της αρχιτεκτονικής του SoC, αντιμετωπίζοντας τις προκλήσεις που σχετίζονται με την κατανάλωση ισχύος/ενέργειας, τη βελτιστοποίηση των επιδόσεων και την ενσωμάτωση ετερογενών στοιχείων. Αξιοποιώντας μεθόδους προσεγγιστικών αριθμητικών κυκλωμάτων, επιτυγχάνονται αξιοσημείωτες βελτιώσεις στην απόδοση και την πολυπλοκότητα του κυκλώματος. Η ενότητα αυτή, διερευνά περαιτέρω την προσαρμογή των παραμέτρων λειτουργίας στα SoC-FPGA, επιτρέποντας τη λεπτομερή ρύθμιση των τάσεων τροφοδοσίας. Αυτή η στρατηγική προσαρ-



Σχήμα 5: Προτεινόμενη προσέγγιση σχεδιασμού για αποδοτικές και ευέλικτες πλατφόρμες SoC-FPGA.

μογής βελτιστοποιεί την κατανάλωση ισχύος, διατηρώντας παράλληλα τα επιθυμητά επίπεδα απόδοσης. Επιπλέον, διερευνάται η ενσωμάτωση προηγμένων SoC-FPGA, όπως οι διατάξεις RFSoC [10] ή Versal ACAP [11], σε διάφορες εφαρμογές ψηφιακών συστημάτων. Αυτές οι εξαιρετικά εξελιγμένες συσκευές χρησιμεύουν ως επιταχυντές για ένα ευρύ φάσμα εφαρμογών DSP, παρέχοντας δυνατότητες υψηλών επιδόσεων, δυνατότητα επαναπρογραμματισμού και χαμηλή κατανάλωση ενέργειας. Συζητούνται βασικά ζητήματα όπως οι διεπαφές υψηλής απόδοσης, οι δυνατότητες συνεπεξεργασίας HW/SW και η στρατηγική ανάπτυξη των επιταχυντών, προσφέροντας μία αρχική αξιολόγηση των επιδόσεων και τη χρήση των πόρων.

Έχοντας αυτούς τους στόχους κατά νου, προτείνουμε μια πολυεπίπεδη προσέγγιση σχεδιασμού για εφαρμογές DSP, εστιάζοντας κυρίως σε πλατφόρμες SoC-FPGA, για τη μεγιστοποίηση της αποδοτικότητάς τους. Το Σχήμα 5 απεικονίζει την προτεινόμενη σχεδιαστική μας προσέγγιση, η οποία περιλαμβάνει τρία κύρια βήματα που αλληλοσυμπληρώνονται και περιγράφονται συνοπτικά παρακάτω:

- Το πρώτο βήμα (① στο Σχήμα 5), που αναφέρεται ως *Conventional Design Pro-*

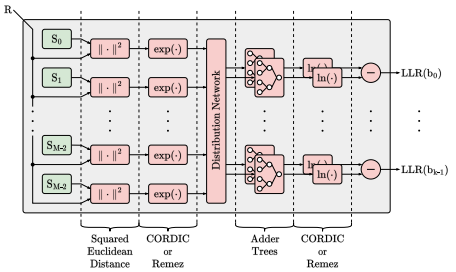
cess, περιλαμβάνει την ανάπτυξη των αλγορίθμων για την εφαρμογή, την εξερεύνηση συνδυασμών λειτουργικών μπλοκ, καθώς και αριθμητικές βελτιστοποιήσεις (π.χ. αριθμητικές πράξεις κινητής υποδιαστολής έναντι σταθερής υποδιαστολής). Το αποτέλεσμα αυτού του βήματος είναι η αρχιτεκτονική χαμηλού επιπέδου του υλικού ή/και του λογισμικού που θα αναπτυχθεί στη συσκευή SoC-FPGA.

- Το επόμενο βήμα (② στο Σχήμα 5), που αναφέρεται ως *Circuit Optimization*, επικεντρώνεται στην αρχιτεκτονική του υλικού από το προηγούμενο βήμα και επιδιώκει να επιτύχει περαιτέρω βελτιστοποιήσεις στα αριθμητικά κυκλώματα χρησιμοποιώντας τεχνικές προσεγγιστικών υπολογισμών. Αυτό οδηγεί σε μια ακόμη πιο βελτιστοποιημένη αρχιτεκτονική υλικού, η οποία αποσκοπεί στη μείωση της κατανάλωσης ισχύος/ενέργειας και της χρήσης των πόρων, καθώς και στην αύξηση της απόδοσης.
- Το τελικό βήμα (③ στο Σχήμα 5), που αναφέρεται ως *Voltage Scaling*, είναι μια διαδικασία, που εφαρμόζεται μετά την ανάπτυξη της αρχιτεκτονικής HW/SW στη και εκτέλεσής της στην συσκευή SoC-FPGA. Στόχος είναι η ρύθμιση των τάσεων τροφοδοσίας της συσκευής στο ελάχιστο επιτρεπτό επίπεδο, διασφαλίζοντας παράλληλα την κανονική λειτουργία του συστήματος.

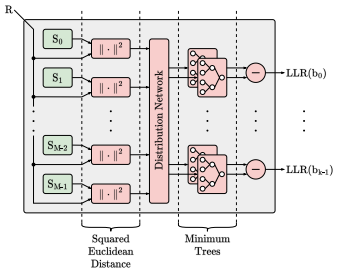
Τέλος, διερευνούμε την απρόσκοπτη ενσωμάτωση διαφορετικών ετερογενών εφαρμογών ταυτόχρονα στην ίδια πλατφόρμα SoC-FGPA. Αυτή η προτεινόμενη συνεργατική προσέγγιση είναι πολλά υποσχόμενη για την σχεδίαση με βάση τα SoC-FPGAs, όπου η σύγκλιση υλικού και λογισμικού, είναι το κλειδί για να αξιοποιήσουμε πλήρως τις δυνατότητές τους. Στην συνέχεια θα παρουσιάσουμε τρεις ενδεικτικές περιπτώσεις εφαρμογής αυτής της προτεινόμενης συνεργατικής σχεδιαστικής προσέγγισης.

2.2 Εξερεύνηση του Χώρου Σχεδίασης Κυκλωμάτων DSP: Αλγόριθμοι και Τεχνικές Προσεγγιστικών Αριθμητικών Υπολογισμών

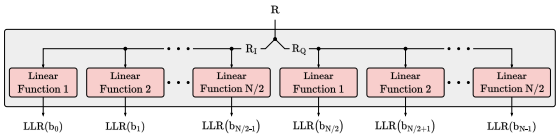
Στην πρώτη περίπτωση εφαρμόζουμε τα βήματα ① + ② σε μία τηλεπικοινωνιακή αλυσίδα επεξεργασίας και συγκεκριμένα επικεντρωνόμαστε στην QAM αποδιαμόρφωση στην πλευρά του δέκτη. Η διαμόρφωση QAM είναι ένας μετασχηματισμός μιας δυαδικής ακολουθίας στον μιγαδικό χώρο (I,Q). Στην αποδιαμόρφωση αντιστρέφουμε αυτόν τον μετασχηματισμό. Στην περίπτωση των προσεγγιστικών υπολογισμών εισάγουμε εκ προθέσεως σφάλματα όταν το προσεγγιστικό αποτέλεσμα είναι επαρκές για την εφαρμογή. Άρα γιατί να έχουμε προσεγγιστικούς υπολογισμούς στην QAM αποδιαμορφωση· Γιατί ορισμένοι αλγόριθμοι αποδιαμόρφωσης έχουν εγγενή ανοχή σε σφάλματα υπολογισμών.



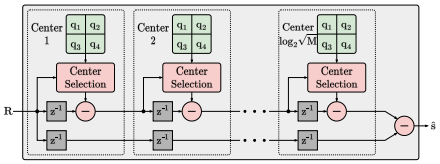
(α) Exact LLR.



(β) Approximate LLR.



(γ) Piecewise LLR.

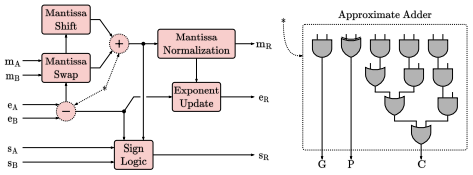


(δ) Approximate ML.

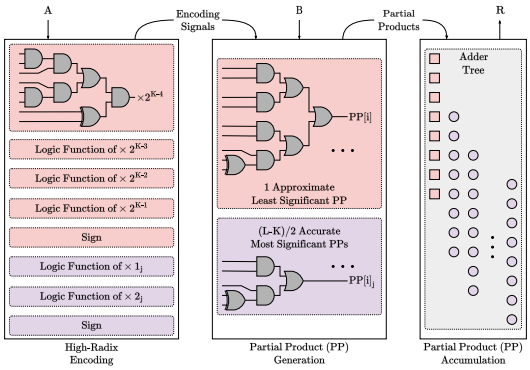
Σχήμα 6: Αρχιτεκτονικές αποδιαμορφωτών QAM.

Αλγόριθμοι Αποδιαμόρφωσης QAM

Για το αλγοριθμικό μέρος εξετάζουμε 4 αλγορίθμους. Οι τρεις πρώτοι ανήκουν στην κατηγορία των Soft-Decision αλγορίθμων, και ονομάζονται Log-Likelihood Ratio - LLR. Η LLR μετρική ορίζεται ως ο λογάριθμος του λόγου των πιθανοτήτων να έχει γίνει μετάδοση του bit 0 έναντι του να έχει μεταδοθεί το bit 1 για ένα λαμβανόμενο σύμβολο <I,Q>. Ο Exact LLR [12] είναι ο ακριβής αλγόριθμος για τον υπολογισμό της LLR μετρικής. Ο Approximate LLR [13] προσεγγίζει τον Exact LLR λαμβάνοντας υπόψη μόνο το πλησιέστερο σημείο του αστερισμού αναφοράς στο ληφθέν σύμβολο. Στον Piecewise LLR [14] προσεγγίζουμε τις καμπύλες πιθανότητας LLR με γραμμικό τρόπο σε τμήματα. Ο τέταρτος αλγόριθμος (Approximate ML [15]) ανήκει στους Hard-Decision αλγορίθμους και σταδιακά βρίσκει το πιο κοντινό σύμβολο που έστειλε ο πομπός. Σε όλες τις περιπτώσεις σχεδιάστηκαν ακριβή και προσεγγιστικά κυκλώματα για αριθμητική κινητής και σταθερής υποδιαστολής. Στο Σχήμα 6 βλέπουμε τις αρχιτεκτονικές των αλγορίθμων που σχεδιάστηκαν, καθώς και ορισμένες εναλλακτικές αλγοριθμικές επιλογές για τον υπολογισμό της εκθετικής και λογαριθμικής συνάρτησης στον Exact LLR.



(α) Αθροιστής κινητής υποδιαστολής.



(β) Πολλαπλασιαστής σταθερής υποδιαστολής.

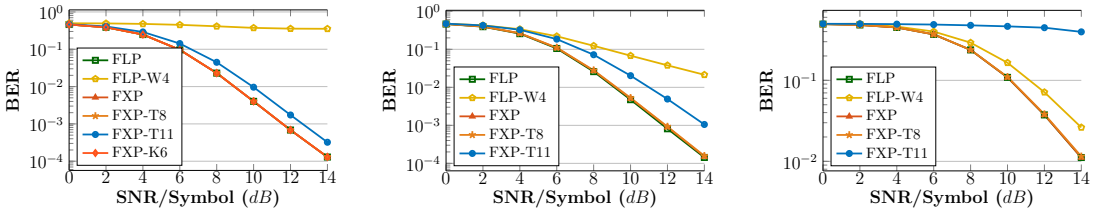
Σχήμα 7: Αρχιτεκτονικές προσεγγιστικών αριθμητικών κυκλωμάτων.

Προσεγγιστικά Αριθμητικά Κυκλώματα

Τα προσεγγιστικά κυκλώματα που χρησιμοποιήθηκαν ήταν ένας προσεγγιστικός αθροιστής κινητής υποδιαστολής, ένας προσεγγιστικός πολλαπλασιαστής σταθερής υποδιαστολής και περικοπή bit για υπολογισμούς σταθερής υποδιαστολής. Συγκεκριμένα για τον αθροιστή κινητής υποδιαστολής χρησιμοποιήθηκε ένας προσεγγιστικός αθροιστής Carry Look-Ahead στην πρόσθεση των mantissa και στην αφαίρεση των exponent. Η προσέγγιση εισάγεται στη δημιουργία του κρατουμένου εξόδου του i-οστού σταδίου. Για τον πολλαπλασιαστή σταθερής υποδιαστολής βασιζόμαστε στην προσεγγιστική δημιουργία των μερικών γινομένων κωδικοποιώντας τα L-K πιο σημαντικά bits σε έναν από τους 2 τελεστές με Radix-4 κωδικοποίηση και τα υπόλοιπα K λιγότερα σημαντικά bits με Radix-2^K κωδικοποίηση. Έτσι η παράμετρος K είναι αυτή που διαμορφώνει την προσέγγιση που εισάγουμε. Στο Σχήμα 7 βλέπουμε τις αρχιτεκτονικές των προσεγγιστικών αριθμητικών κυκλωμάτων που υλοποιήθηκαν.

Πειραματική Αξιολόγηση Κυκλωμάτων

Η σύνθεση και υλοποίηση των κυκλωμάτων έγινε για αποδιαμόρφωση QAM64 σε πλατφόρμα Zynq Ultrascale+ MPSoC και για λόγους προτυποποίησης το εργαλείο σύνθεσης παραμετροποιήθηκε ώστε να μην χρησιμοποιήσει εξειδικευμένους πόρους για υψηλής απόδοσης πολλαπλασιασμούς και προσθέσεις ενώ η αξιολόγηση έγινε με βάση τις μετρικές Bit-Error Rate (BER), της κατανάλωσης πόρων του FPGA και το μέσο σχετικό σφάλμα των



(α) Approximate LLR (ALLR) **(β)** Piecewise LLR (PLLR) **(γ)** Approximate ML (AML)

Σχήμα 8: Αποτελέσματα BER για κυκλώματα QAM64. FLP = ακριβές κινητής υποδιαστολής, FLP-W4 = προσεγγιστική πρόσθεση κινητής υποδιαστολής [16], FXP = ακριβές σταθερής υποδιαστολής, FXP-TX = σταθερής υποδιαστολής με περικοπή X bit, FXP-K6 = προσεγγιστικός πολλαπλασιασμός σταθερής υποδιαστολής [17].

Πίνακας 1: Κατανάλωση πόρων κυκλωμάτων QAM64 στο Xilinx Zynq Ultrascale+ XCZU7EV[1] (ZCU106)

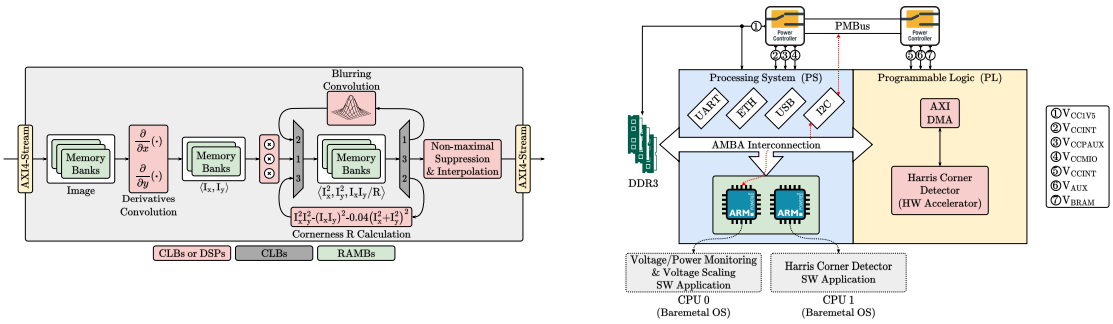
	Approximate LLR (ALLR)						**Piecewise LLR (PLLR)**					**Approximate ML (AML)**				
	FLP	FLP-W4	FXP	FXP-T8	FXP-T11	FXP-K6	FLP	FLP-W4	FXP	FXP-T8	FXP-T11	FLP	FLP-W4	FXP	FXP-T8	FXP-T11
LUT	106097	104881	56278	19957	10210	37653	2060	2020	160	76	59	3848	3504	139	91	44
DFF	42432	42432	21596	10684	9316	51572	886	886	215	126	96	1204	1204	44	28	18
MHz[2]	286	294	312	323	416	321	333	435	571	645	740	385	416	588	645	769
W[3]	5.0	4.9	5.5	3.0	2.5	3.6	0.8	0.8	1.1	1.0	0.9	0.6	0.7	1.0	0.7	0.6
MSps/W	57.2	60.0	56.7	107.7	166.4	89.2	416.3	543.8	519.1	645.0	822.2	641.7	594.3	588.0	921.4	1281.7
avg. MRE	–	86.00	2.22	20.59	193.14	1.74	–	177.05	1.28	23.35	165.55	–	8.36	0.02	4.65	22.98

[1] Διαθέσιμοι πόροι: 230K LUTs, 461K DFFs.
[2] Μέγιστη συχνότητα λειτουργίας. Ο ρυθμός μετάδοσης όλων των κυκλωμάτων είναι 1 δείγμα ανά κύκλο ρολογιού, επομένως MHz = Msamples/s.
[3] Κατανάλωση ισχύος στην μέγιστη συχνότητα λειτουργίας.

αποτελεσμάτων σε σύγκριση με τα αντίστοιχα αποτελέσματα των αρχιτεκτονικών κινητής υποδιαστολής χωρίς προσεγγιστικά αριθμητικά κυκλώματα. Στο Σχήμα 8 δίνονται τα αποτελέσματα του BER των αρχιτεκτονικών μας QAM64 για διαφορετικές τιμές SNR ανά μεταδιδόμενο σύμβολο QAM. Ενώ στον Πίνακα 1 παρουσιάζονται τα αποτελέσματα της υλοποίησης μετά το Place & Route των κυκλωμάτων.

Από τα αποτελέσματα παρατηρούμε ότι:

- **Approximate LLR**: τα κυκλώματα με αριθμητική FXP, είτε ακριβή είτε προσεγγιστικά, δείχνουν πολύ μικρό σχετικό σφάλμα στο BER, ενώ το προσεγγιστικό κύκλωμα FLP-W4 δεν μπορεί να ανταποκριθεί. Για την κατανάλωση πόρων βλέπουμε ότι το FLP-W4 δίνει πολύ μικρά κέρδη ($\sim 1\%$), και μόλις 3% αύξηση της συχνότητας λειτουργίας, ενώ τα FXP δίνουν κέρδη $50\% - 90\%$ στους πόρους και $9\% - 45\%$ αύξηση της συχνότητας λειτουργίας.
- **Piecewise LLR**: το κύκλωμα FXP-T8 έχει σχετικά μικρό σχετικό σφάλμα στο BER



(α) Εύρεση γωνιών Harris. **(β)** Τελικό σύστημα.

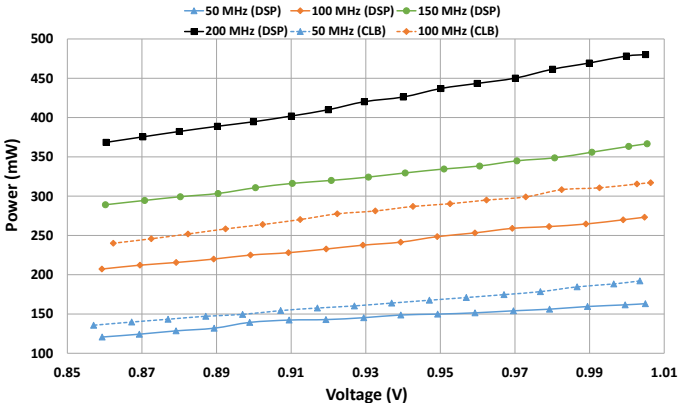
Σχήμα 9: Αρχιτεκτονικές εφαρμογής και τελικού συστήματος.

σε σχέση με το FLP, ενώ το προσεγγιστικό κύκλωμα FXP-T11 έχει χειρότερη απόδοση από ότι στην περίπτωση του ALLR, ενώ το FLP-W4 και πάλι δεν μπορεί να ανταποκριθεί. Για τους πόρους βλέπουμε ότι το FLP-W4 δίνει πάλι πολύ μικρά κέρδη στη κατανάλωση πόρων ($\sim 2\%$), αλλά σημαντική αύξηση της συχνότητας λειτουργίας (30%), ενώ τα FXP δίνουν κέρδη $75\% - 97\%$ στους πόρους και $70\% - 120\%$ αύξηση της συχνότητας λειτουργίας.

- **Approximate ML**: όλα τα κυκλώματα έχουν πολύ χειρότερη BER απόδοση από ότι τα ALLR και PLLR για ακριβώς την ίδια είσοδο και SNR/symbol, ενώ εάν επικεντρωθούμε μόνο σε αυτά τα κυκλώματα το FXP-T11 αποκλίνει σε BER απόδοση από το ακριβές FLP. Για την κατανάλωση πόρων βλέπουμε ότι το FLP-W4 δίνει κέρδη ($\sim 9\%$) και αύξηση της συχνότητας λειτουργίας (8%), ενώ όλα τα FXP δίνουν κέρδη $\sim 97\%$ στους πόρους και $53\% - 100\%$ αύξηση της συχνότητας λειτουργίας. Η πολύ χειρότερη BER απόδοση μεταξύ όλων των αλγορίθμων και αριθμητικών κάνει την χρήση του μη βιώσιμη λύση.

2.3 Βελτιστοποίηση της Κατανάλωσης Ισχύος σε SoC-FPGA

Στην 2η περίπτωση εφαρμόζουμε τα βήματα ① + ③ σε μία πολύ διαδομένη εφαρμογή επεξεργασίας εικόνας, την Εύρεση Γωνιών Harris. Ο συγκεκριμένος αλγόριθμος επεξεργασίας εικόνων, περιέχει απαιτητικούς υπολογισμούς, όπως συνελίξεις και πράξεις κινητής και σταθερής υποδιαστολής. Για να γίνει προσαρμογή των τάσεων τροφοδοσίας ο ένας πυρήνας του επεξεργαστή (CPU0) χρησιμοποιείτε για την διαχείριση της διαδικασίας αυτής, ενώ ο άλλος (CPU1) είναι υπεύθυνος για την εκτέλεση της εφαρμογής (Σχήμα 9).



Σχήμα 10: Αποτελέσματα κλιμάκωσης τάσης του VCCINT συναρτήσει της συχνότητα λειτουργίας του PL.

Πειραματική Διάταξη & Αξιολόγηση

Η εφαρμογή εκτελέστηκε σε ένα Zynq-7000 SoC-FPGA και μελετήθηκαν δύο σενάρια για την εφαρμογή: 1) μια καθαρά εφαρμογή λογισμικού και 2) μια εφαρμογή υλικού/λογισμικού με δύο παραλλαγές, i) Μία που κάνει χρήση των DSP μπλοκς του FPGA για τις αριθμητικές πράξεις και ii) μία που δεν τα χρησιμοποιεί. Επιπλέον εφαρμόστηκαν δύο στρατηγικές για την προσαρμογή των τάσεων τροφοδοσίας. Η πρώτη λαμβάνει υπόψη μόνο τις τάσεις τροφοδοσίας που σχετίζονται με το PS υποσύστημα ή μόνο με το PL υποσύστημα του SoC και η δεύτερη λαμβάνει υπόψη συνδυαστικά όλες μαζί.

Προχωρώντας στα αποτελέσματα για το PL μόνο αγνοώντας το PS βλέπουμε στο Σχήμα 10 πως κλιμακώνεται η κατανάλωση ισχύος για διαφορετικές τάσεις τροφοδοσίας για το VCCINT για διάφορες συχνότητες λειτουργίας και παραμετροποίηση χρήσης των DSP ή όχι. Ενώ στον Πίνακα 2 βλέπουμε τα μέγιστα κέρδη σε κατανάλωση ισχύος των δύο παραλλαγών στο PL για 100MHz. Αυτό που παρατηρούμε είναι ότι ανεξάρτητα της παραλλαγής τα κέρδη παραμένουν περίπου ίδια.

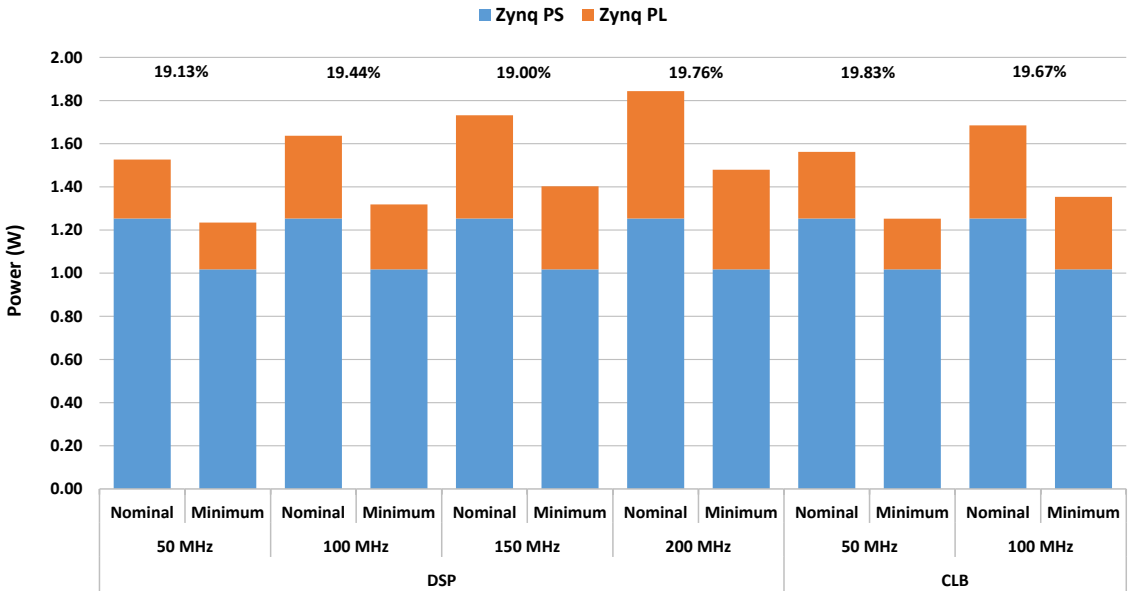
Στον Πίνακα 3 βλέπουμε τα αποτελέσματα της κατανάλωσης για όλο το SoC όταν κάνουμε συνδυαστική προσαρμογή των τάσεων στο PL και στο PS. Συγκεκριμένα βλέπουμε τη μέγιστη δυνατή μείωση για 100MHz συχνότητα λειτουργίας σε κάθε τάση τροφοδοσίας και το αντίστοιχο κέρδος σε κατανάλωση ισχύος, ενώ στο κάτω μέρος του πίνακα έχουμε τα συνολικά κέρδη σε κατανάλωση ισχύος ξεχωριστά για το PS, το PL-DSP και το PL-CLB. Τέλος στο Σχήμα 11 βλέπουμε το συνολικό κέρδος σε επίπεδο SoC για όλες τις συχνότητες λειτουργίας στο PL. Αυτό που παρατηρούμε είναι ότι σε όλες τις περιπτώσεις έχουμε μία μείωση κατά μέσο όρο 19.5% στην κατανάλωση ισχύος, ενώ το PS είναι αυτό που συμβάλλει

Πίνακας 2: Αποτελέσματα κατανάλωσης ισχύος για μεμονωμένη ρύθμιση των τάσεων τροφοδοσίας στο PL για συχνότητα λειτουργίας 100MHz.

	Rail	Voltage (V)			Power (W)		
		Nominal	Minimum	Difference	Nominal	Minimum	Difference
DSP	V_{CCINT}	1.00472	0.85925	14.5%	0.27314	0.20720	24.1%
	V_{CCAUX}	1.80717	1.54586	14.5%	0.06284	0.05353	14.7%
	V_{CCBRAM}	1.00917	0.85799	15%	0.04730	0.04020	15%
CLB	V_{CCINT}	1.00637	0.86232	14.3%	0.31704	0.24003	24.3%
	V_{CCAUX}	1.80744	1.54589	14.5%	0.06732	0.05576	17.2%
	V_{CCBRAM}	1.00754	0.85512	15.1%	0.04731	0.04020	15%

Πίνακας 3: Αποτελέσματα κατανάλωσης ισχύος για συνδυαστική ρύθμιση των των τάσεων τροφοδοσίας στα PS και PL για συχνότητα λειτουργίας 100MHz.

		Rail	Voltage (V)			Power (W)		
			Nominal	Minimum	Difference	Nominal	Minimum	Difference
PS		V_{CCPINT}	1.00513	0.86951	13.5%	0.45023	0.32843	27%
		V_{CCPAUX}	1.80374	1.66607	7.6%	0.14936	0.13116	12.2%
		V_{CCMIO}	1.80244	1.66554	7.6%	0.05634	0.05206	7.6%
		V_{CC1V5}	1.50346	1.36673	9%	0.59755	0.50603	15.3%
PL	DSP	V_{CCINT}	1.00500	0.85978	14.4%	0.27314	0.20720	24.1%
		V_{CCAUX}	1.80703	1.54422	14.5%	0.06284	0.05353	14.8%
		V_{CCBRAM}	1.00841	0.86394	14.3%	0.04730	0.04020	15%
	CLB	V_{CCINT}	1.00500	0.85978	14.4%	0.31704	0.24003	24.3%
		V_{CCAUX}	1.80703	1.54422	14.5%	0.06732	0.05576	17.2%
		V_{CCBRAM}	1.00841	0.86394	14.3%	0.04730	0.04020	15%
Total PS Power						1.25348	1.01768	18.8%
Total PL Power (DSP)						0.38328	0.30094	21.5%
Total PL Power (CLB)						0.43167	0.33599	22.1%

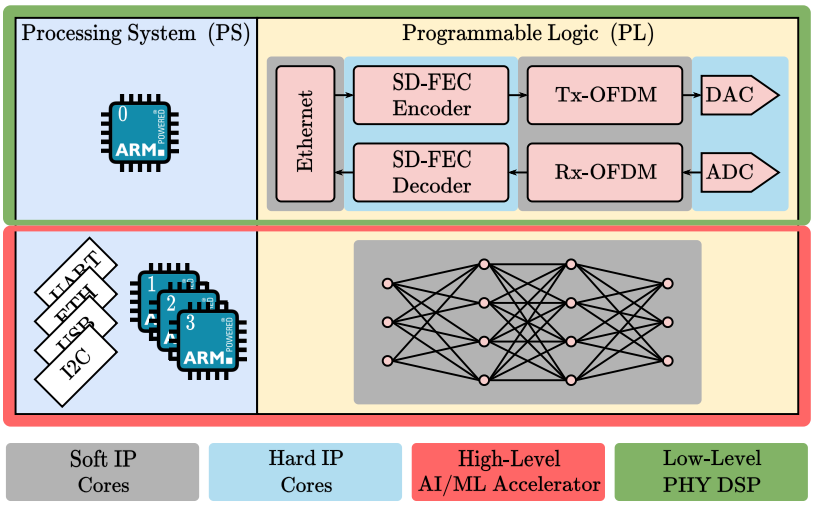


Σχήμα 11: Συνολική κατανάλωση ισχύος διαφόρων παραμετροποιήσεων του συστήματος PS-PL.

περισσότερο στη συνολική κατανάλωση.

2.4 Διαμοιρασμός SoC-FPGA για Υπολογιστικές Εφαρμογές Χαμηλού και Υψηλού Επιπέδου στο Edge

Ενώ στην τρίτη περίπτωση θεωρούμε ότι έχουμε δύο πολύ απαιτητικές εφαρμογές επεξεργασίας και εφαρμόζουμε και στις 2 εφαρμογές μόνο το βήμα ① και εξερευνάμε την ταυτόχρονη ενσωμάτωσή τους στην ίδια πλατφόρμα SoC-FPGA. Οι εφαρμογές που μελετάμε είναι: 1) μία τηλεπικοινωνιακή αλυσίδα και 2) εφαρμογές μηχανικής μάθησης. Ο λόγος που επιλέξαμε τις συγκεκριμένες εφαρμογές είναι ότι τα δίκτυα επόμενης γενιάς B5G αναμένεται να διανέμουν τους υπολογισμούς από Cloud υποδομές σε υπολογιστικούς κόμβους στο Edge, μερικοί από τους οποίους θα πρέπει να επεξεργάζονται τόσο χαμηλού επιπέδου τηλεπικοινωνιακές όσο και εργασίες υψηλού επιπέδου, όπως το AI/ML. Αυτοί οι υπολογιστικοί κόμβοι στο Edge θα απαιτούν υψηλή απόδοση, επαναπρογραμματισμό και χαμηλή κατανάλωση ισχύος, ειδικά όταν βρίσκονται στις απομακρυσμένες άκρες του δικτύου. Θεωρώντας ότι ένα SoC-FPGA θα χρησιμοποιηθεί σαν Edge συσκευή διερευνούμε την δυνατότητα λογικής διαίρεσης του SoC σε δύο μέρη (Σχήμα 12) ώστε να μπορεί να υλοποιήσει χαμηλού επιπέδου τηλεπικοινωνιακές λειτουργίες και υψηλού επιπέδου επιτάχυνση εφαρμογών AI/ML ταυτόχρονα. Ο λόγος κατανάλωσης πόρων του FPGA του τμήματος L προς το τμήμα H δεν θα είναι σταθερός αλλά θα εξαρτάται από τις εκάστοτε ανάγκες.



Σχήμα 12: Κατάτμηση SoC-FPGA για να φιλοξενήσει επεξεργασία χαμηλού επιπέδου και επιτάχυνση AI.

Πειραματική Διάταξη & Αξιολόγηση

Για την αξιολόγηση της προτεινόμενης αρχιτεκτονικής θεωρούμε διάφορους συνδυασμούς των δύο εφαρμογών που υλοποιούνται σε ένα RFSoC-FPGA. Για την τηλεπικοινωνιακή αλυσίδα θεωρούμε ότι έχουμε μετάδοση OFDM και 250MHz συχνότητα λειτουργίας, ενώ για το AI/ML μέρος χρησιμοποιήθηκε η ροή σχεδίασης Vitis AI της Xilinx για επιτάχυνση αντίστοιχων εφαρμογών με χρήση της DPU.

Στον Πίνακα 4 δίνεται η κατανάλωση πόρων των επιμέρους υποσυστημάτων της προτει-

Πίνακας 4: Κατανάλωση πόρων στο XCZU28DR[1] RFSoC

Name	FF	LUT	RAMB	DSP
B4096-DPU	105008	53540	257	562
Tx-Rx DSP chain	50K–100K	10K–20K	6–14	180–370
PL PCIe	20201	8183	22	0
10G Eth. Core	22004	51929	3	0
10G L2 Tx+Rx	912	1581	8	0
Misc. (AXI, etc.)	7156	9599	13	0

[1] Διαθέσιμοι πόροι: FF = 850K, LUT = 425K, BRAM = 1K, DSP = 4.2K

Πίνακας 5: Trade-off επιδόσεων-κατανάλωσης πόρων στο XCZU28DR

Configuration	Throughput		Resources		
	Gbps	TOP/s	LUT	BRAM	DSP
A+4·B+E	2.5	0.5	39%–48%	30%–33%	30%–48%
A+4·B+4·F+E	2.5	0.5	51%–61%	30%–33%	78%–96%
2·A+4·B+E	2.5	1.12	51%–61%	54%–57%	43%–61%
2·A+4·B+2·F+E	2.5	1.12	58%–67%	54%–57%	67%–85%
2·A+2·B+2·F+E	1.25	1.12	53%–58%	53%–55%	59%–68%
3·A+2·B+E	1.25	1.34	59%–64%	77%–78%	48%–57%
3·A+2·B+2·F+E	1.25	1.34	65%–70%	77%–78%	72%–81%

A=DPU, B=Tx+Rx DSP, F=FIR, E={10G Eth., L2 Tx+Rx, PL PCIe, Misc.}

νομένης αρχιτεκτονικής. Ενώ στον Πίνακα 5 βλέπουμε την κατανάλωση πόρων διαφόρων συνδυασμών των δύο κύριων λειτουργιών. Όσο περισσότερος τηλεπικοινωνιακός ρυθμός μετάδοσης απαιτείται, περισσότερη χαμηλού επίπεδου DSP λειτουργικότητα θα υλοποιηθεί και λιγότερο AI/ML. Από την άλλη αν χρειάζεται περισσότερη επιτάχυνση AI/ML, θα χρησιμοποιηθούν περισσότερες DPU και λιγότερο χαμηλού επιπέδου DSP. Σε κάθε περίπτωση, παρατηρούμε ότι πληρούνται οι περιορισμοί κατανάλωσης πόρων.

2.5 Συμπεράσματα

Σε αυτή την ενότητα, παρουσιάσαμε την προτεινόμενη σχεδιαστική προσέγγιση για ενσωματωμένους υπολογιστές σε πλατφόρμες SoC-FPGA. Για να αξιολογήσουμε την εφαρμοσιμότητά της, λάβαμε υπόψη μας συσκευές SoC-FPGA τελευταίας τεχνολογίας και εφαρμόσαμε διαφορετικούς συνδυασμούς της σχεδιαστικής προσέγγισης σε διαφορετικές εφαρμογές DSP. Συγκεκριμένα, υιοθετήσαμε μια ολοκληρωμένη σχεδιαστική προσέγγιση που περιελάμβανε εκτεταμένη εξερεύνηση του χώρου σχεδίασης (DSE) τόσο σε αλγοριθμικό όσο και σε αρχιτεκτονικό επίπεδο για τη βελτιστοποίηση των υλοποιήσεων των κυκλωμάτων μας. Επιπλέον, εφαρμόσαμε συστηματικά την κλιμάκωση τάσης για να μειώσουμε την κατανάλωση ισχύος σε επίπεδο συστήματος εντός του SoC-FPGA και πραγματοποιήσαμε μια αρχική διερεύνηση της ταυτόχρονης ανάπτυξης δύο απαιτητικών εφαρμογών στο ίδιο SoC-FPGA. Τα πειράματά μας οδήγησαν στα ακόλουθα αποτελέσματα:

1. Για τη βελτιστοποίηση του κυκλώματος, πετύχαμε έως και 98% μείωση στη χρήση των πόρων σε σύγκριση με την ακριβή αρχιτεκτονική κινητής υποδιαστολής του ίδιου αλγορίθμου, ενώ αυξήσαμε τη συχνότητα λειτουργίας του έως και 122% και την κα-

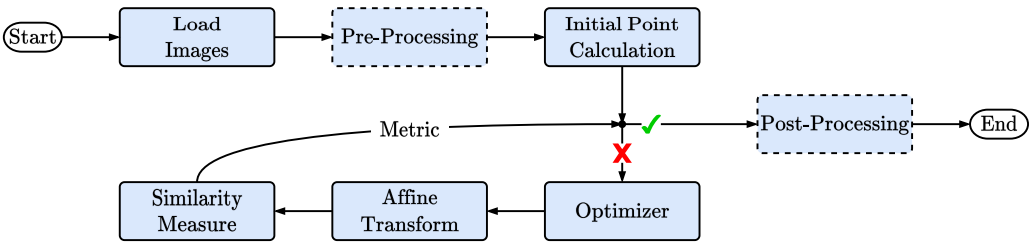
τανάλωση ενέργειας όχι περισσότερο από 1 Watt.

2. Με τη συστηματική προσέγγισή μας για την κλιμάκωση της τάσης και ανάλογα με τις αρχιτεκτονικές επιλογές για τον επιταχυντή σε μια HW/SW εφαρμογή που αναπτύχθηκε στη συσκευή SoC-FPGA, επιτύχαμε εξοικονόμηση ισχύος $20 - 30\%$ σε επίπεδο συστήματος, διατηρώντας παράλληλα τη λειτουργική ορθότητα και την ονομαστική απόδοση του συστήματος.
3. Παρουσιάσαμε με επιτυχία την πρακτικότητα του συνδυασμού των λειτουργιών DSP χαμηλού επιπέδου που απαιτούνται σε μια μονάδα βασικής ζώνης με την επιτάχυνση AI/ML σε μια συσκευή SoC-FPGA τελευταίας τεχνολογίας (RFSoC). Η αρχική μας αξιολόγηση επιβεβαιώνει ότι αυτή η ενσωμάτωση είναι εφικτή και παρέχει πολλά υποσχόμενα αποτελέσματα όσον αφορά την κατανάλωση πόρων και τις μετρήσεις επιδόσεων.

3 Αποδοτικές Αρχιτεκτονικές για Εφαρμογές Εικόνας/Βίντεο σε Σύγχρονες Συσκευές SoC-FPGA

3.1 Εισαγωγή

Αυτή η ενότητα παρουσιάζει εξειδικευμένους επιταχυντές υλικού που έχουν σχεδιαστεί για την επίτευξη υψηλών επιδόσεων σε εφαρμογές εικόνας και βίντεο. Η έμφαση δίνεται σε δύο συγκεκριμένους τομείς: Ιατρική απεικόνιση και πλοήγηση βασισμένη στην όραση Vision-Based Navigation - VBN στο διάστημα. Αυτοί οι τομείς έχουν μοναδικές απαιτήσεις, συμπεριλαμβανομένης της λειτουργίας σε πραγματικό χρόνο και της χαμηλής κατανάλωσης ισχύος/ενέργειας. Η υλοποίηση αυτών των επιταχυντών θέτει σημαντικές προκλήσεις λόγω της ευρείας χρήσης βαθιά ενσωματωμένων συσκευών. Στην περίπτωση της Ιατρικής Απεικόνισης, διερευνούμε μία εφαρμογή επεξεργασίας Image Registration, ενώ για την VBN, η εφαρμογή είναι η Παρακολούθηση της Πόζας (6D Pose Tracking - 6D PS) μη συνεργάσιμων διαστημικών αντικειμένων (π.χ. ανενεργών δορυφόρων). Για να αντιμετωπίσουμε αυτές τις προκλήσεις, προτείνουμε αποδοτικά συστήματα HW/SW για συσκευές SoC-FPGA και εισάγουμε βελτιστοποιήσεις σε επίπεδο υλικού για τη βελτίωση των επιδόσεων. Παρέχουμε ολοκληρωμένα αποτελέσματα επιδόσεων και συγκρίνουμε τις προσεγγίσεις μας με ενσωματωμένες συσκευές γενικής χρήσης, επιτρέποντας μια ενδελεχή αξιολόγηση των προτεινόμενων συστημάτων HW/SW.



Σχήμα 13: Ροή επεξεργασίας μίας γενικού σκοπού Image Registration εφαρμογής.

3.2 Βελτιστοποίηση Εφαρμογής Image Registration για Ενσωματωμένα Συστήματα με χρήση SoCs που βασίζονται σε FPGA

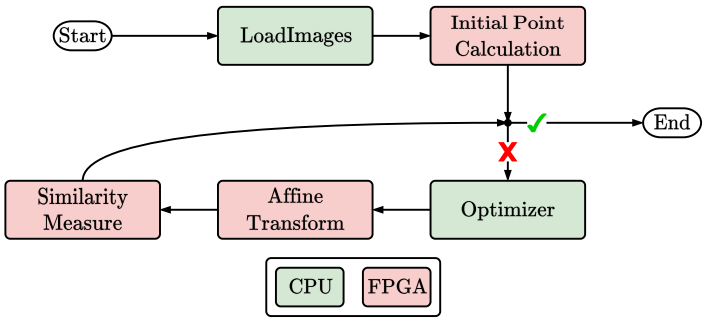
Εισαγωγή

Το Image Registration αναφέρεται στην αντιστοίχιση μεταξύ δύο εικόνων, τόσο χωρικά όσο και σε σχέση με την τιμή των εικονοστοιχείων. Χρησιμοποιείται για την αντιστοίχιση δύο ή περισσότερων εικόνων που λήφθηκαν σε διαφορετικό χρόνο ή από διαφορετικούς αισθητήρες/σημεία θέασης [78]. Μία από τις εικόνες είναι η σταθερή, ενώ οι άλλες είναι οι κινούμενες. Το πλαίσιο αναφοράς στη σταθερή εικόνα είναι σταθερό και η κινούμενη εικόνα μετασχηματίζεται χωρικά για να ταιριάζει με τον στόχο (Σχήμα 13).

Οι αλγόριθμοι Image Registration μπορούν να χωριστούν σε δύο κατηγορίες: 1) Με βάση την ένταση της εικόνας και 2) Με βάση τα χαρακτηριστικά της εικόνας [79]. Οι μέθοδοι που βασίζονται στην ένταση συγκρίνουν τα μοτίβα έντασης εντός των εικόνων μέσω μετρικών συσχέτισης ή σφάλματος. Πρέπει να καθοριστεί ένα ζεύγος εικόνων, μια μετρική ομοιότητας, ένας βελτιστοποιητής και ένας γεωμετρικός μετασχηματισμός. Οι μέθοδοι που βασίζονται σε χαρακτηριστικά βρίσκουν αντιστοιχίες μεταξύ χαρακτηριστικών της εικόνας (όπως σημεία, γραμμές και περιγράμματα) και τις χρησιμοποιούν για να εκτιμήσουν έναν συνολικό μετασχηματισμό, ο οποίος τελικά εφαρμόζεται στην κινούμενη εικόνα προκειμένου να την ευθυγραμμίσει. Γενικά, το Image Registration μπορεί να περιγραφεί ως μια επαναληπτική διαδικασία trial & error. Ένα μέτρο ομοιότητας ή απόστασης υπολογίζεται μεταξύ των εικόνων σε κάθε επανάληψη και χρησιμοποιείται για να διαπιστωθεί αν είναι επαρκώς ευθυγραμμισμένες. Η διαδικασία αυτή ελέγχεται από τον βελτιστοποιητή που ξεκινά από μια αρχική κατάσταση και καθορίζει τα επόμενα βήματα για την επίτευξη μιας βέλτιστης ευθυγράμμισης.

Πίνακας 6: Προφίλ εφαρμογής για δύο ζεύγη οφθαλμικών εικόνων.

	Total Execution Latency (sec)	Transform & Measure Execution Latency (sec)
Matching pair	39.923572	39.9216161
Differing pair	103.084555	103.080554

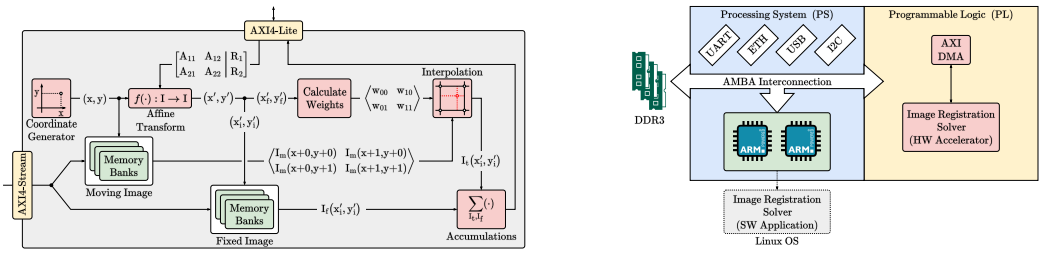


Σχήμα 14: Διαχωρισμός HW/SW της εφαρμογής Image Registration.

Προτεινόμενη HW/SW Αρχιτεκτονική Εφαρμογής Image Registration

Στον Πίνακα 6 παρουσιάζεται η μετρούμενη καθυστέρηση εκτέλεσης ενός Image Registration συστήματος σε SW, όπου υπάρχει ένα ζεύγος ταιριαστών εικόνων εισόδου (Matching pair) και ένα ζεύγος αταίριαστων εικόνων (Differing pair) του συνόλου δεδομένων εισόδου. Ο πίνακας περιλαμβάνει τη συνολική καθυστέρηση εκτέλεσης της εφαρμογής, καθώς και την καθυστέρηση εκτέλεσης της συνάρτησης Transform & Measure. Η καθυστέρηση αυτής της συνάρτησης απαιτεί σχεδόν το 99% της συνολικής καθυστέρησης και στις δύο περιπτώσεις. Κατά συνέπεια, αυτό είναι το αλγοριθμικό μέρος που θα αποτελέσει το επίκεντρο της επιταχυνόμενης σχεδίασης στο FPGA. Με βάση τα αποτελέσματα του πίνακα, μια υψηλού επιπέδου κατανομή της εφαρμογής Image Registration παρέχεται στο Σχήμα 14. Τα λιγότερο υπολογιστικά απαιτητικά τμήματα, όπως η φόρτωση των εικόνων, η ρύθμιση των παραμέτρων μετασχηματισμού και η αξιολόγηση των κριτηρίων τερματισμού παραμένουν ως στοιχεία λογισμικού, ενώ η εφαρμογή του υποψήφιου μετασχηματισμού στις συντεταγμένες εικονοστοιχείων και ο υπολογισμός του μέτρου ομοιότητας υλοποιούνται στην FPGA. Ο υλοποιημένος επιταχυντής υλικού στο FPGA αποτελείται από διάφορες υπομονάδες, όπως:

- Η μονάδα Transform, η οποία εφαρμόζει τον Affine μετασχηματισμό σε κάθε εικονοστοιχείο. Λαμβάνει τις συντεταγμένες εικονοστοιχείου και τις τρέχουσες παραμέτρους



(α) Αρχιτεκτονική επιταχυντή στο FPGA (β) Ενσωμάτωση HW/SW

Σχήμα 15: Τελική αρχιτεκτονική εφαρμογής Image Registration.

μετασχηματισμού ως εισόδους και παράγει τις μετασχηματισμένες συντεταγμένες, χωρισμένες στο δεκαδικό και το κλασματικό μέρος τους.

- Η μονάδα Interpolation Weights, υπεύθυνη για τη δημιουργία βαρών παρεμβολής που χρησιμοποιούνται για τον υπολογισμό της τελικής τιμής έντασης του μετασχηματισμένου εικονοστοιχείου. Λαμβάνει τα κλασματικά μέρη των μετασχηματισμένων συντεταγμένων από την μονάδα Transform και παράγει τα κατάλληλα βάρη παρεμβολής.
- Η μονάδα Interpolation, όπου υπολογίζεται η τελική ένταση του μετασχηματισμένου εικονοστοιχείου. Αυτή η μονάδα λαμβάνει ως είσοδο τα βάρη και τις τιμές έντασης των τριών εικονοστοιχείων που γειτνιάζουν με το μετασχηματισμένο εικονοστοιχείο, τα οποία παράγονται στην μονάδα Interpolation Weights, καθώς και το βάρος και την τιμή έντασης του ίδιου του μετασχηματισμένου εικονοστοιχείου.
- Η μονάδα Accumulations, η οποία περιλαμβάνει πολλαπλές μονάδες συσσώρευσης που υπολογίζουν τα απαραίτητα αθροίσματα για τον υπολογισμό της συσχέτισης μεταξύ των εικόνων. Λαμβάνει την τιμή έντασης του εικονοστοιχείου παρεμβολής από τη μονάδα υπολογισμού παρεμβολής και την αντίστοιχη τιμή έντασης του εικονοστοιχείου στην κινούμενη εικόνα

Επιπλέον, ο επιταχυντής υλικού ενσωματώνει δύο μονάδες μνήμης αφιερωμένες στην αποθήκευση εικόνων, μαζί με πολλαπλές μονάδες ελέγχου που συντονίζουν τις λειτουργίες του. Αυτοί οι ελεγκτές παράγουν συντεταγμένες εικονοστοιχείων, διαχειρίζονται τις προσβάσεις στη μνήμη, τη δέσμευση δεδομένων και τη ρύθμιση της επικοινωνίας με την εφαρμογή λογισμικού. Στο Σχήμα 15 δίνεται η τελική αρχιτεκτονική που υλοποιήθηκε.

Πίνακας 7: Τελικές παράμετροι affine μετασχηματισμού και μέτρο ομοιότητας στις τρεις διαφορετικές πλατφόρμες.

Transformation Parameters	**Intel i5 4200U**	**XC7X010**	
		PS	**PS+PL**
T1	1.002974	1.010045	0.990064
T2	-0.015803	-0.015795	-0.017116
T3	45.320175	45.801357	44.415871
T4	0.024869	0.027911	0.021519
T5	0.997309	1.003294	0.989229
T6	-16.065662	-15.859092	-15.909966
Measure of Match	0.792186	0.792874	0.780669

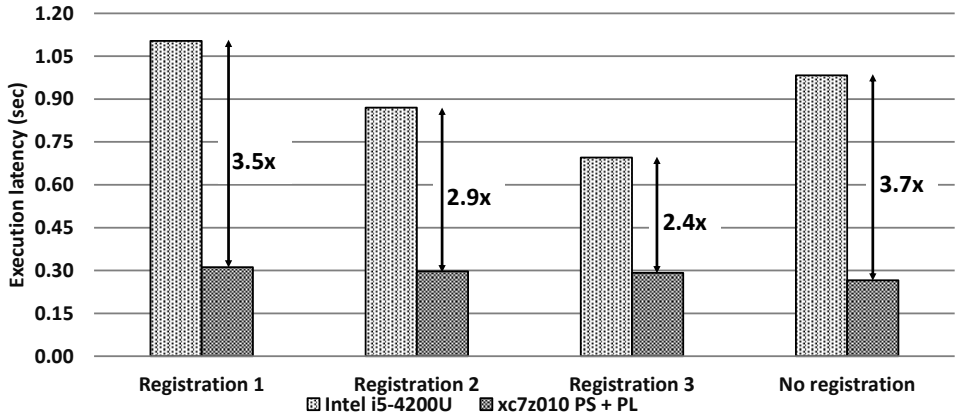
Πειραματική Αξιολόγηση

Η αξιολόγηση πραγματοποιήθηκε στην αναπτυξιακή πλακέτα Zybo, η οποία διαθέτει το XC7Z010 της οικογένειας συσκευών Zynq-7000 SoC. Η πλατφόρμα Zynq αποτελείται από το Processing System (PS), που φιλοξενεί έναν διπύρηνο επεξεργαστή ARM Cortex-A9, και το Programmable Logic (PL) που αναφέρεται στο FPGA της συσκευής. Η εφαρμογή HW/SW που σχεδιάστηκε συγκρίνεται με υλοποιήσεις λογισμικού που εκτελούνται σε έναν επεξεργαστή Intel i5 4200U και στον επεξεργαστή ARM του Zynq PS.

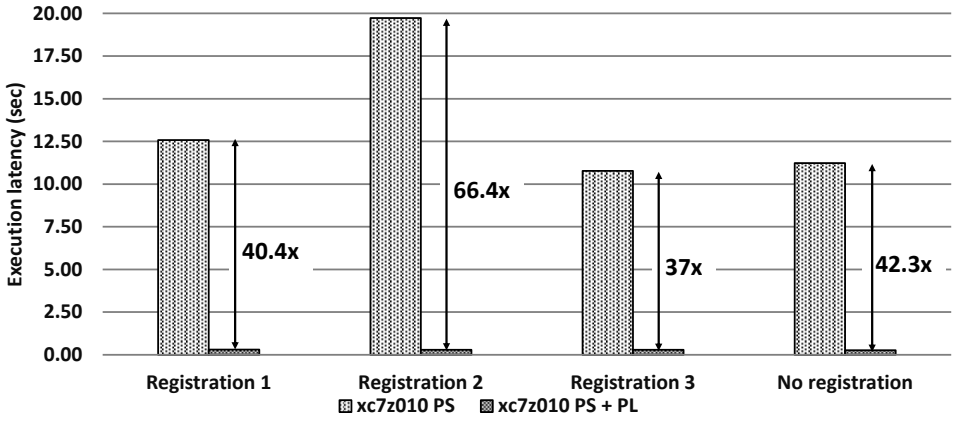
Τα πρώτα μας πειράματα αποσκοπούν στην παροχή ποσοτικοποιημένων αποτελεσμάτων ότι ο επιταχυντής υλικού παρέχει ακριβή αποτελέσματα σε σχέση με την εφαρμογή λογισμικού αναφοράς. Ως μετρικές ποιότητας χρησιμοποιούνται οι έξι παράμετροι του Affine μετασχηματισμού (2 για περιστροφή, 2 για κλιμάκωση και 2 για μετατόπιση) και το τελικό μέτρο ομοιότητας, με την εφαρμογή λογισμικού σε Intel i5 να χρησιμεύει ως αναφορά.

Στον Πίνακα 7 δίνονται οι τιμές των παραμέτρων του τελικού μετασχηματισμού που θα εφαρμοστεί στην κινούμενη εικόνα, καθώς και το τελικό μέτρο ομοιότητας που υπολογίστηκε στις τρεις διαφορετικές πλατφόρμες. Όλοι οι υπολογισμοί πραγματοποιήθηκαν με αριθμητική κινητής υποδιαστολής μονής ακρίβειας. Όπως φαίνεται, οι τιμές των παραμέτρων μετασχηματισμού διαφέρουν ελαφρώς, αλλά η διαφορά στο πλαίσιο της συγκεκριμένης δοκιμής είναι αποδεκτή.

Αφού επικυρώθηκε ότι οι υπολογισμένες παράμετροι μετασχηματισμού έχουν αποδεκτή α-



(α) Σύγκριση με Intel i5-4200U @1.6GHz



(β) Σύγκριση με ARM Cortex-A9 @650MHz

Σχήμα 16: Αξιολόγηση των επιδόσεων της προτεινόμενης HW/SW εφαρμογής.

πόκλιση από την βάση αναφορά και ότι το τελικό μέτρο ομοιότητας είναι αποδεκτά κοντά σε όλες τις πλατφόρμες, η καθυστέρηση εκτέλεσης του προτεινόμενου συστήματος HW/SW συγκρίνεται με τις υλοποιήσεις λογισμικού στον Intel i5 4200U και στον ARM Cortex-A9 (Zynq PS). Οι συγκρίσεις πραγματοποιούνται για single-threaded C κώδικα σε όλες τις πλατφόρμες. Οι συχνότητες λειτουργίας των επεξεργαστών διατηρήθηκαν στη βασική συχνότητα που υποστηρίζουν, δηλαδή 650MHz για τον ARM Cortex-A9 και 1.6GHz για τον Intel i5. Ο επιταχυντής FPGA λειτουργεί στα 100MHz, όπως αναφέρθηκε από το εργαλείο σύνθεσης.

Στο Σχήμα 16α παρουσιάζεται η μετρούμενη μέση καθυστέρηση εκτέλεσης για 100 εκτελέσεις της HW/SW εφαρμογής σε σύγκριση με την εφαρμογή λογισμικού σε Intel i5 για

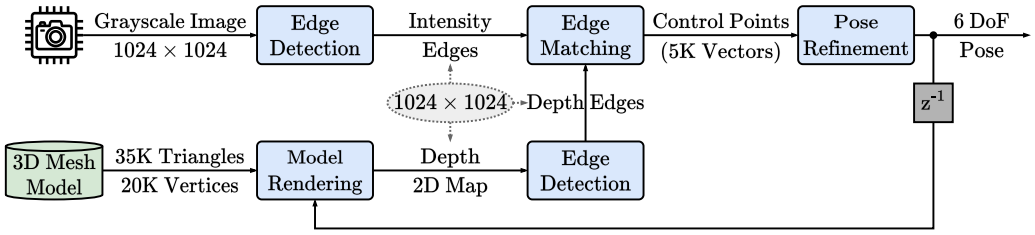
τέσσερα διαφορετικά σύνολα εικόνων ματιών. Τα τρία πρώτα από αυτά οδηγούν σε επιτυχές registration και το τελευταίο αποτυγχάνει. Στο σχήμα σημειώνεται επίσης η επιτευχθείσα επιτάχυνση εκτέλεσης. Σε όλες τις περιπτώσεις, παρατηρείται μείωση της καθυστέρησης εκτέλεσης και η μέση επιτευχθείσα επιτάχυνση για τις τέσσερις περιπτώσεις δοκιμής είναι 3×. Η μεταβλητή καθυστέρηση εκτέλεσης προκύπτει από το γεγονός ότι τα κριτήρια τερματισμού σχετίζονται άμεσα με (i) τον μέγιστο αριθμό επαναλήψεων που επιτρέπει ο βελτιστοποιητής, (ii) το όριο ανοχής του βελτιστοποιητή και (iii) τη φύση των εικόνων που χρησιμοποιούνται. Μικρές αποκλίσεις στις παραμέτρους που παράγονται στο τέλος μιας επανάληψης μπορεί να οδηγήσουν σε μια διαφορετική τροχιά σύγκλισης, η οποία προκαλείται από μια διαφορά στα βήματα βελτιστοποίησης.

Το πείραμα επαναλαμβάνεται σε ένα πλήρως ενσωματωμένο περιβάλλον εκτέλεσης και τα αποτελέσματα που προέκυψαν φαίνονται στο Σχήμα 16β, το οποίο δείχνει την καθυστέρηση εκτέλεσης της HW/SW εφαρμογής σε σύγκριση με την εφαρμογή λογισμικού που εκτελείται στον ARM Cortex-A9 (Zynq PS). Είναι σαφώς εμφανές ότι υπάρχει σημαντική μείωση της καθυστέρησης εκτέλεσης και στις τέσσερις περιπτώσεις και η μέση επιτάχυνση που επιτυγχάνεται είναι 46×. Και πάλι, η παρατηρούμενη διακύμανση στην καθυστέρηση εκτέλεσης οφείλεται στη μεταβλητή φύση των κριτηρίων τερματισμού του βελτιστοποιητή. Παρ΄ όλα αυτά, η εφαρμογή HW/SW που σχεδιάστηκε υπερτερεί έναντι των δύο υλοποιήσεων μόνο με λογισμικό, παρά τη χρήση υλοποίηση αριθμητικής κινητής υποδιαστολής μονής ακρίβειας.

Στον Πίνακα 8 παρουσιάζεται η κατανάλωση των πόρων της τελικής αρχιτεκτονικής HW. Παρόλο που η HW/SW εφαρμογή χρησιμοποιεί αριθμητική κινητής υποδιαστολής μονής ακρίβειας, μπορεί να χωρέσει στο FPGA, παρά την υψηλή χρήση των DSP (95%). Ωστόσο, η υψηλή χρήση των DSP αποτελεί σημαντικό σημείο συμφόρησης για την υλοποιημένη εφαρμογή, καθώς ο παραλληλισμός της εφαρμογής καθίσταται απαγορευτικός. Επιπλέον, η πλειονότητα των πόρων του FPGA έχει διατεθεί για υπολογισμούς και όχι για επικοινωνία. Όσον αφορά τις απαιτήσεις σε πόρους μνήμης, η αξιοποίηση των BRAM είναι περίπου 60%, γεγονός που επιτρέπει την προοπτική ανάλυσης εικόνων υψηλότερων διαστάσεων.

Πίνακας 8: Κατανάλωση πόρων στη συσκευή XC7Z010 Zynq.

XC7Z010 PL Resources	**Available**	**Used Resources**	
		Accelerator	**Communication**
LUTs	17600	11396 (65%)	2768 (16%)
DFFs	35200	19965 (57%)	2656 (8%)
RAMBs	60	34 (57%)	1 (2%)
DSPs	80	76 (95%)	0 (0%)



Σχήμα 17: Προτεινόμενος αλγόριθμος εντοπισμού πόζας: κύριες λειτουργίες & παράδειγμα ροής δεδομένων.

3.3 Υλοποίηση Ενσωματωμένης Εφαρμογής για Αυτόνομη Πλοήγηση στο Διάστημα με βάση την Όραση σε SoC-FPGA

Εισαγωγή

Οι διαδικασίες προσέγγισης στο διάστημα είναι ελιγμοί που πραγματοποιούνται από ένα διαστημικό σκάφος που βρίσκεται σε τροχιά με σκοπό την προσέγγιση και παραμονή κοντά σε ένα άλλο στόχο. Αλλά υπάρχουν περιπτώσεις όπου δεν γνωρίζουμε σε τι κατάσταση είναι αυτός ο στόχος κάνοντας πολύ δύσκολη την εξ΄ αποστάσεως διαχείριση της διαδικασίας από τη Γη [18–20]. Μία λύση την οποία και μελετάμε είναι η αυτόνομη πλοήγηση με βάση την όραση για την παρακολούθηση και συγχρονισμό με το στόχο [21]. Τέτοιες εφαρμογές βασίζονται σε αλγορίθμους εντοπισμού της θέσης με βάση το μοντέλο του στόχου. Οι αλγόριθμοι αυτοί έχουν πολύ υψηλές υπολογιστικές ανάγκες [21], καθώς θα πρέπει να πραγματοποιούν επεξεργασία εικόνων υψηλής ευκρίνειας σε πραγματικό χρόνο και να παρέχουν αυτόνομη λειτουργία χωρίς καμία παρέμβαση από το έδαφος.

Υλοποίηση Εφαρμογής Εύρεσης Πόζας

Τα βασικά βήματα της εφαρμογής για την ανίχνευση της πόζας του στόχου δίνονται στο Σχήμα 17. Η αρχή λειτουργίας βασίζεται στην επίλυση εξισώσεων για την εύρεση της τρισδιάστατης κίνησης του στόχου μέσω ενός συνόλου ακμών μεταξύ 2 εικόνων, οι οποίες ακμές ταιριάζουν μεταξύ τους. Το σύνολο των ακμών προκύπτει συγκρίνοντας το συνολικό αριθμό των ακμών που υπολογίζονται για τις 2 εικόνες, οι οποίες η μία λαμβάνεται σε πραγματικό χρόνο από μια κάμερα, ενώ η άλλη προκύπτει από το γνωστό μοντέλο που έχουμε για τον στόχο. Συγκεκριμένα η εφαρμογή που αναπτύχθηκε αποτελείται από τα παρακάτω υποσυστήματα:

- **Ανίχνευση ακμών (Edge Detection)**: Υπολογίζει τις ακμές στην εικόνα που λαμβάνεται από την κάμερα (Intesity Edges) και στην εικόνα που δημιουργείται με βάση το μοντέλο του δορυφόρου-στόχου (Depth Edges)
- **Απεικόνιση μοντέλου (Model Rendering)**: Υπολογίζει την προβολή του δορυφόρου-στόχου στο δισδιάστατο επίπεδο και δημιουργεί έναν χάρτη βάθους (Depth 2D Map)
- **Αντιστοίχιση άκρων (Edge Matching)**: Βρίσκει τη δισδιάστατη μετακίνηση μεταξύ των Intesity Edges και Depth Edges
- **Βελτίωση πόζας (Pose Refinement)**: Με βάση ένα σύνολο ταιριασμένων ακμών, γίνεται επίλυση συστήματος εξισώσεων και εύρεση της τρισδιάστατης κίνησης του δορυφόρου-στόχου

Για την υλοποίηση της πιο αποδοτικής αρχιτεκτονικής πρέπει να λάβουμε διάφορες παραμέτρους υπόψη, όπως ο χρόνος εκτέλεσης, η πολυπλοκότητα και το είδος των αριθμητικών πράξεων, κ.α. Για αυτό, πρώτο βήμα πού κάνουμε μία σκιαγράφηση του προφίλ της αλυσίδας επεξεργασίας σε επίπεδο SW για να δούμε τις απαιτήσεις κάθε συνάρτησης. Τα αποτελέσματα αυτά δίνονται στο Πίνακα 9.

Με βάση τα αποτελέσματα αποφασίζουμε πως θα γίνει ο διαμοιρασμός των επιμέρους λειτουργιών στα υποσυστήματα του SoC-FPGA. Στην συγκεκριμένη περίπτωση η επεξεργασία σε επίπεδο πιξελ και ακμών θα γίνει στο PL του SoC, ενώ η επίλυση των αλγεβρικών εξισώσεων στο PS του SoC (Σχήμα 18α). Ενώ στο Σχήμα 18β δίνεται μια πιο λεπτομερή αρχιτεκτονική των λειτουργιών που θα υλοποιηθούν στο PL. Επίσης στο ίδιο σχήμα βλέπουμε την αλληλεπίδραση μεταξύ των υποσυστημάτων στο PL, αλλά και μεταξύ του PL και PS, καθώς και το μέγεθος και είδος των δεδομένων που ανταλλάσσουν. Τέλος το Σχήμα 18γ παρουσιάζει τον προσαρμοσμένο χρονοπρογραμματισμό των επιμέρους λειτουργιών που δημιουργήθηκε για την συγκεκριμένη εφαρμογή και αρχιτεκτονική για την ομαλή λειτουργία ολόκληρου του συστήματος.

Αξιολόγηση Εφαρμογής

Η αξιολόγηση της εφαρμογής έγινε σε Zynq-7000 SoC-FPGA χρησιμοποιώντας ένα σύνολο από 700 συνθετικές εικόνες του δορυφόρου ENVISAT μεγέθους 1Megapixel η καθεμία σε απόσταση 50 και 30 μέτρων από την κάμερα και συγκρίνοντας τα σφάλματα Ευθυγράμμισης, Θέσης και Γωνίας έχοντας ως αναφορά τα αντίστοιχα αποτελέσματα του SW. Επίσης στόχος ήταν το σύστημα να μπορεί να υποστηρίξει τουλάχιστον 10 καρέ/δευτερόλεπτο ρυθμό επεξεργασίας.

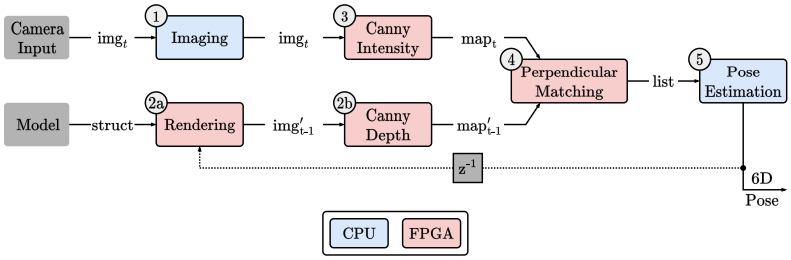
Προχωρώντας στα αποτελέσματα βλέπουμε ότι το σύστημα που σχεδιάστηκε πετυχαίνει

Πίνακας 9: Προφίλ εφαρμογής σε ARM A9 (Zynq) για εικόνες 1Mpixel σε $30m$ & $50m$.

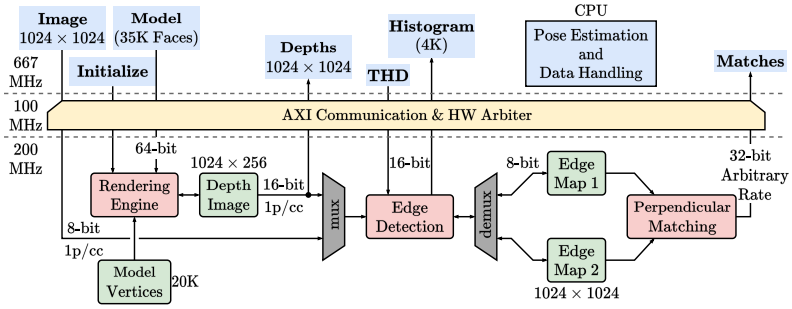
Algorithm −function	Time/Frame (cam@50m)	Time/Frame (cam@30m)	I/O (Mbit)	RAM (MB)	Other issue*
Imaging	82 msec	82 msec	16.8	1	
−fetch	56 msec	56 msec	16.8		FX, SP
−enhance	26 msec	26 msec	16.8		
Rendering	343 msec	869 msec	52	7	
−initialize	0.1 msec	0.1 msec	0.01		
−projtrian	7.5 msec	10 msec	125		FL+FX,
−gen+chk	206 msec	544 msec	4211		HP, EM
−pixdepth	51 msec	153 msec	659		
−other	78.4 msec	162 msec	≤ 1		
Edge Detect	303 msec	326 msec	12.6	18	
−gradient	210 msec	210 msec	41.9		HP+SP,
−suppress	44 msec	44 msec	67.1		FX
−hysterss	16 msec	29 msec	67.1		
−other	33 msec	43 msec	33.5		
Matching	38 msec	39 msec	17.6	2.2	SP, FX
Pose estim.	51 msec	99 msec	≤ 1.6	≤ 1	
−LQS-fit	16 msec	32 msec	≤ 1.6		PC, EM,
−LSinlier	33 msec	65 msec	≤ 1.6		FL
−other	2 msec	2 msec	≤ 1.6		
*Total**:*	1120 msec	1741 msec	8.4	44	–

*abbreviations: *FL*=floating-point arithmetic, *FX*=small fixed-point variables, *HP*=highly parallelizable, *SP*=streaming processing, *PC*=high programming complexity, *EM*=expensive math (e.g., divisions, trigonometric)

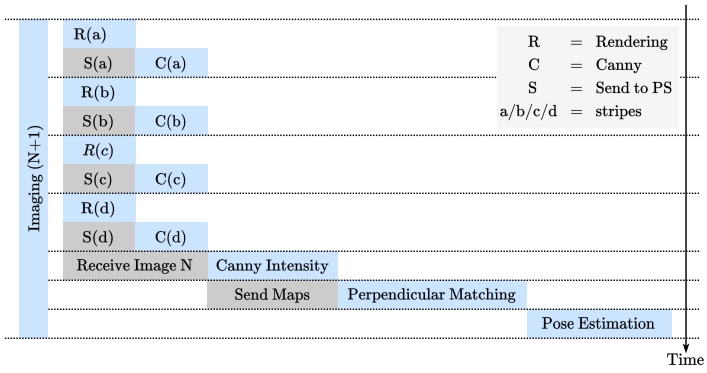
**total= imaging + rendering + 2×detection + matching + pose estimation



(α) Διαχωρισμός HW/SW.

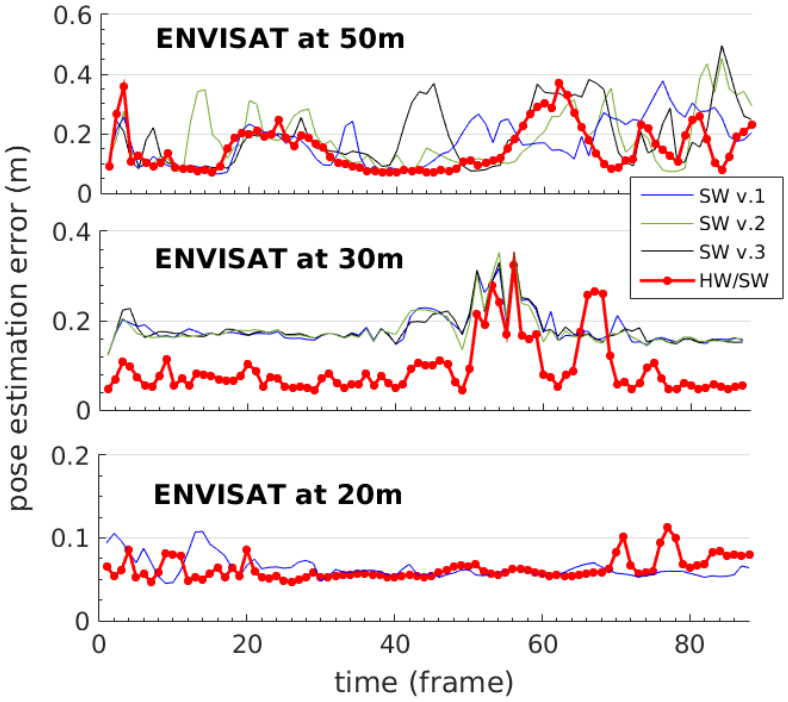


(β) Διασύνδεση/ροή δεδομένων μεταξύ SW & HW στην πλευρά PS & PL.



(γ) Προγραμματισμός εργασιών.

Σχήμα 18: Τελική αρχιτεκτονική HW σε επίπεδο λειτουργιών/δεδομένων.



Σχήμα 19: Εκτίμηση στάσης (σφάλμα ευθυγράμμισης) του συστήματος HW/SW για τρεις αποστάσεις ENVISAT. Και στις 3 περιπτώσεις το σφάλμα $< 40cm$.

λάθος στην ευθυγράμμιση $< 0.5\%$ που αντιστοιχεί σε 5-20cm ανάλογα την απόσταση του δορυφόρου από την κάμερα, το λάθος στον υπολογισμό της θέσης του δορυφόρου από το σύστημα δεν ξεπερνάει τα 30cm, ενώ το σφάλμα γωνίας είναι μεταξύ 1-3 μοιρών (Σχήμα 19).

Όσο αναφορά τις επιδόσεις του συνολικού συστήματος σε FPS, σε όλες τις περιπτώσεις επιτυγχάνεται πάνω από 10 FPS παρόλο που ο φόρτος εργασίας αυξάνεται έως και 87% καθώς πλησιάζουμε τον δορυφόρο. Τέλος, στον Πίνακα 10 δίνεται η κατανάλωση πόρων και οι επιδόσεις του τελικού συστήματος. Επισημαίνεται ότι πετύχαμε 19× επιτάχυνση της συνολικής εφαρμογής ενώ η κατανάλωση σε πόρους παραμένει χαμηλή σε LUT/FF/DSP ενώ σε BRAM είναι αυξημένη λόγω των απαιτήσεων του Rendering και του Edge Detection.

Πίνακας 10: Τελικό σύστημα: Κατανάλωση πόρων στο SoC-FPGA XC7X100IFFG900-2L

Component	**LUT**	**DFF**	**DSP**	**RAMB**	**Time (ms)**
Rendering	93383	148071	966	224	11±4
Edge Detection	2948	3174	4	346.5	6
Perp. Matching	298	389	-	5.5	5
Commun.+Misc.	2895	3777	-	6	3 Gbps
Pose Estimation		ARM core #1			55±10
Auxiliary SW		ARM core #1			0.5
Total	99524 (36%)	155411 (28%)	970 (48%)	582 (77%)	83±14 (∼12 fps)

3.4 Συμπεράσματα

Σε αυτή την ενότητα, παρουσιάσαμε τη συστηματική σχεδίαση δύο αποδοτικών ενσωματωμένων συστημάτων HW/SW που για εφαρμογές εικόνας/βίντεο και τις εκτελέσαμε σε συσκευές SoC-FPGA. Ο εφαρμογή επεξεργασίας Image Registration, απεικονίστηκε με επιτυχία σε μία από τις μικρότερες συσκευές Zynq-7000 SoC-FPGA, ενώ έγινε χρήση αριθμητική κινητής υποδιαστολής μονής ακρίβειας. Επιπλέον, η ποιότητα του προτεινόμενου συστήματος επαληθεύτηκε πειραματικά σε σχέση με υλοποιήσεις λογισμικού, παρουσιάζοντας υψηλά κέρδη απόδοσης και έως 67 φορές μεγαλύτερη ταχύτητα σε σχέση με ενσωματωμένες CPU, διατηρώντας παράλληλα την αλγοριθμική ακρίβεια. Στην περίπτωση της εφαρμογής επεξεργασίας πλοήγησης με βάση την όραση (VBN), που στοχεύει στις μελλοντικές απαιτήσεις αποστολών ADR, προτείναμε μια αρχιτεκτονική υψηλού επιπέδου που εκμεταλλεύεται τη δομή του Zynq-7000, βελτιώσαμε τον αλγόριθμο υπολογιστικής όρασης για την παρακολούθηση της πόζας του ENVISAT και επινοήσαμε μια μεθοδολογία υλοποίησης HW/SW για την προσαρμογή και την αποτελεσματική αντιστοίχιση όλων των λειτουργιών στο SoC-FPGA. Αντιμετωπίσαμε τις υπολογιστικές προκλήσεις μέσω ενός συνδυασμού τεχνικών σε επίπεδο αρχιτεκτονικής και VHDL. Το προκύπτον σύστημα VBN επιτυγχάνει ρυθμό επεξεργασίας 10-14 FPS για εικόνες 1Mpixel, με μόνο 4.3 Watt μέση ισχύ, παρακολουθώντας τον δορυφόρο-στόχο σε πραγματικό χρόνο και με μέσο σφάλμα 0.5% της απόστασης μεταξύ της κάμερας και του στόχου.

4 Επεξεργασία Βασικής Ζώνης σε RFSoC-FPGA για δίκτυα κινητών επικοινωνιών 5G/B5G

4.1 Εισαγωγή

Η μετάβαση από τα δίκτυα 4G στα 5G οφείλεται κυρίως στις απαιτήσεις διαφόρων αναδυόμενων τεχνολογιών, όπως η επαυξημένη πραγματικότητα, η μετάδοση υψηλής ευκρίνειας βίντεο, αλλά και οι ανάγκες για χαμηλή καθυστέρηση στην επικοινωνία, η αξιοπιστία και η ενεργειακή απόδοση. Αυτή η μετάβαση ήταν μια χρονοβόρα διαδικασία μέχρι να καθοριστούν οι τελικές προδιαγραφές της τεχνολογίας 5G. Για να επιτύχουν τους στόχους τους τα δίκτυα 5G αξιοποιούν νέες τεχνολογίες, όπως: 1) τα millimeter wave - (mmWave), το massive MIMO και το Beamforming. Επειδή όμως η ατμοσφαιρική απορρόφηση επηρεάζει σημαντικά τη μετάδοση με mmWave, οι οπτικές ίνες είναι ο κυρίος τρόπος διασύνδεσης μεταξύ των επεξεργαστών βασικής ζώνης (Basedand Unit - BBU) και των Remote Radio Head (RRH). Γενικά, η τεχνολογία Radio-over-Fiber (RoF) είναι η αιχμή του δόρατος των υλοποιήσεων για δίκτυα 5G, και ιδιαίτερα η αναλογική (Analog Radio-Over-Fiber - A-RoF)

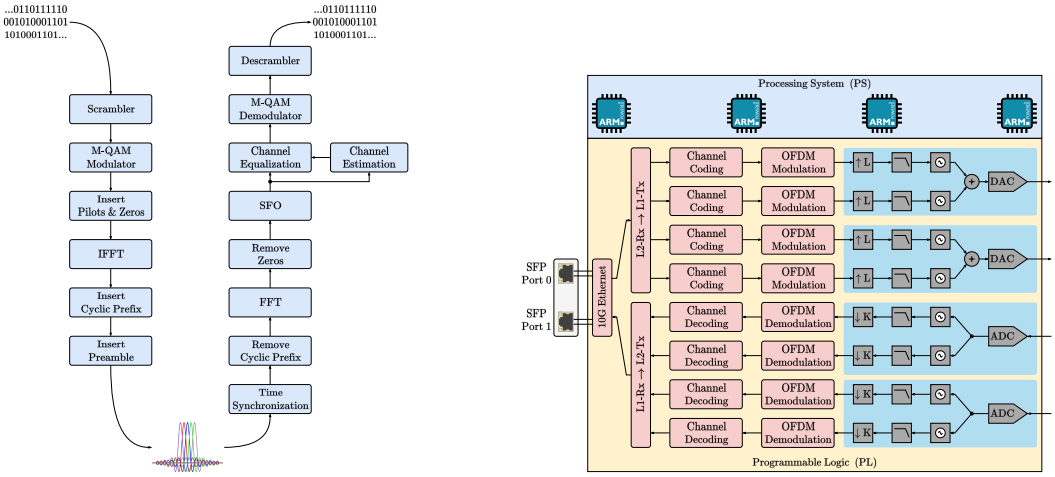
μετάδοση.

Οι BBUs απαιτούν επαναπρογραμματισιμότητα λόγω των εξελισσόμενων προτύπων επικοινωνίας. Αυτό είναι αποτέλεσμα της μετάβασης από αρχιτεκτονικές Distributed-RAN (D-RAN) σε αρχιτεκτονικές Cloud-RAN (C-RAN), με συνέπεια οι BBUs βασισμένες σε FPGA να διευκολύνουν τη μετάβαση σε τεχνολογίες fronthaul που θα μεταφέρουν κίνηση Ethernet με λογική RoF. Με στόχο την παροχή μιας αποτελεσματικής λύσης για την BBU, εστιάζουμε στη σχεδίαση μιας αρχιτεκτονικής βασισμένης σε SoC-FPGA για χειρισμό της ψηφιακής επεξεργασίας βασικής ζώνης και μετατροπής του σήματος από τη βασική ζώνη σε μία ενδιάμεση συχνότητα (Intermedeate Frequency - IF) για την μετάδοση μέσω οπτικών ινών. Για το σκοπό αυτό επιλέγουμε ως πλατφόρμα για την υλοποίηση της ψηφιακής επεξεργασίας βασική ζώνης το Zynq Ultrascale+ RFSoC-FPGA. Επιλέγουμε την συγκεκριμένη πλατφόρμα γιατί ενσωματώνει ετερογενής μονάδες επεξεργασίας μαζί με DAC και ADC στο ίδιο τσιπ. Η σημαντικά απλούστερη διασύνδεση του FPGA με τους DACs και ADCs μειωμένη την πολυπλοκότητα διασύνδεσή στους. Επίσης οι ενσωματωμένοι DACs και ADCs υποστηρίζουν φίλτρα μείωσης ή αύξησης της συχνότητας δειγματοληψίας, καθώς επίσης και ψηφιακούς μίκτες για την μετατροπή του σήματος από τη βασική ζώνη στην ενδιάμεση συχνότητα. Τέλος, τα RFSoC-FPGA υποστηρίζουν μέσω εξειδικευμένων μπλοκ στην υποδομή τους λειτουργίες διόρθωσης σφαλμάτων για τηλεπικοινωνιακές εφαρμογές και συνδεσιμότητα Ethernet 100Gbps.

4.2 Ψηφιακή Επεξεργασία Φυσικού Επιπέδου για Μετάδοση/Λήψη OFDM

Για τη μονάδα βασικής ζώνης χρησιμοποιούμε την πλακέτα ZCU111 που φιλοξενεί τη συσκευή Zynq Ultrascale+ RFSoC της Xilinx [22]. Η τηλεπικοινωνιακή αλυσίδα που υλοποιήθηκε βασίζεται σε OFDM μετάδοση υποστηρίζοντας διαμόρφωση QAM, FFT 256-σημείων με κυκλικό πρόθεμα (Cylcic-Prefix - CP) 64 δειγμάτων. Στο αναλογικό το εύρος μετάδοσης είναι 256MHz. Το τελικό σύστημα στο RFSoC-FPGA περιλαμβάνει:

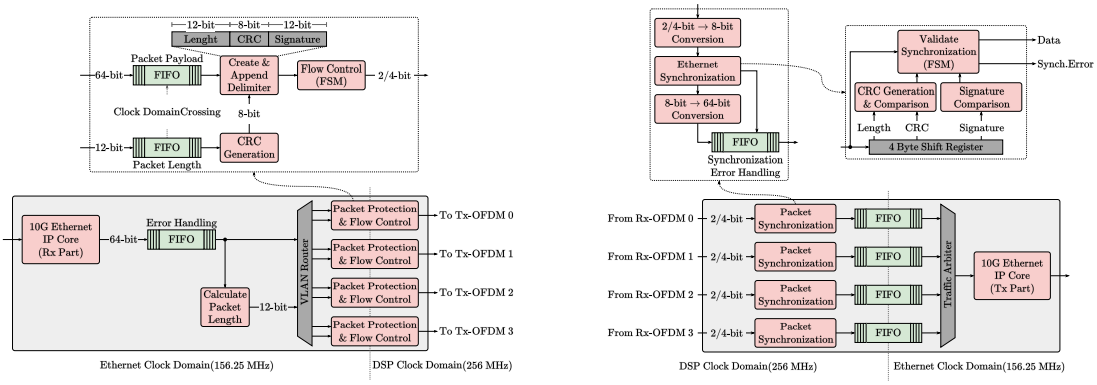
1. Αλυσίδα ψηφιακής επεξεργασίας για τον πομπό και τον δέκτη, 4 σε σύνολο για τον καθένα
2. 10Gbps Ethernet επικοινωνία μέσω SFP με το δίκτυο του παρόχου
3. Οι DACs και ADCs συνδέονται με το αναλογικό front-end με καλώδια SMA
4. Η (απο-)πολυπλεξία πολλαπλών OFDM σημάτων πραγματοποιείται στους DAC/ADC.



(α) Αλυσίδα επεξεργασίας OFDM (β) Τελικό σύστημα

Σχήμα 20: Αρχιτεκτονική επεξεργαστή βασικής ζώνης.

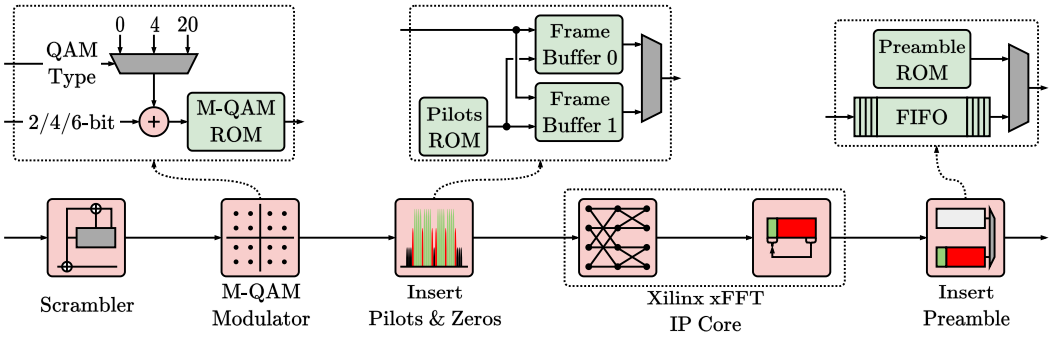
Στο Σχήμα 20α παρουσιάζεται η αλυσίδα επεξεργασίας DSP Tx και Rx που σχεδιάστηκε για να υποστηρίξει την κεντρικοποιημένη αρχιτεκτονική δικτύου για μετάδοση A-RoF. Ενώ το Σχήμα 20β απεικονίζει την αρχιτεκτονική υψηλού επιπέδου του επεξεργαστή βασικής ζώνης στη συσκευή RFSoC. Ο πομπός λαμβάνει δεδομένα από τον μεταγωγέα Ethernet και τα προωθεί σε τέσσερις ανεξάρτητες αλυσίδες επεξεργασίας που διαχειρίζονται μια ζώνη εύρους 256MHz η κάθε μία. Οι ζώνες επεξεργάζονται ανεξάρτητα (Σχήμα 20β πομπός) και τα δεδομένα τους προωθούνται σε τέσσερις ξεχωριστές μονάδες DAC που είναι διαμορφωμένες σε ζεύγη για να συγχωνεύουν τις αντίστοιχες ζώνες τους και να παράγουν δύο σήματα διπλής ζώνης που μετατρέπονται ψηφιακά σε IF στα 1.5GHz και έπειτα μετατρέπονται σε αναλογικό σήμα. Αυτά τα αναλογικά σήματα μεταδίδονται μέσω καλωδίων SMA σε έναν αναλογικό μίκτη συχνοτήτων, ικανό να μετατρέπει τα λαμβανόμενα σήματα στο αναλογικό τους IF των 5GHz και από εκεί να οδηγεί το επόμενο στάδιο, το οποίο αποτελείται από τα ηλεκτροπτικούς μετατροπείς. Στο δέκτη, τα οπτικά σήματα μετατρέπονται σε ηλεκτρικά από οπτοηλεκτρικούς μετατροπείς. Αυτά τα αναλογικά σήματα έχουν αναλογικό IF των 5GHz και μετατρέπονται σε αναλογικό IF των 1.5GHz από έναν αναλογικό μίκτη συχνοτήτων, ο οποίος τροφοδοτεί τους ADCs της συσκευής RFSoC. Ένας μοναδικός ADC ψηφιοποιεί το λαμβανόμενο σήμα και περαιτέρω μετατρέπει ψηφιακά ένα σήμα διπλής ζώνης διαχωρίζοντάς το σε δύο μεμονωμένες ζώνες. Η έξοδος ενός ADC επεξεργάζεται στη συνέχεια από τα DSP μπλοκ του υποσυστήματος του δέκτη (Σχήμα 20β δέκτης) και τα αποτελέσματα μεταδίδονται μέσω του Ethernet IP στον εξωτερικό μεταγωγέα Ethernet. Στην συνέχεια θα παρουσιάσουμε ορισμένα από τα βασικά μπλοκ επεξεργασίας του τελικού συστήματος.



(α) L2 Ethernet-Rx σε L1 Tx-OFDM **(β)** L1 Rx-OFDM σε L2 Ethernet-Tx

Σχήμα 21: L2 αρχιτεκτονική από/προς Ethernet προς/από Tx/Rx-DSP.

Για την χρήση του RFSoC ως BBU, η διασύνδεση Ethernet πραγματοποιείται με τη χρήση του υποσυστήματος 10G/25G High Speed Ethernet Subsystem Xilinx IP Core, διαμορφωμένου για 10Gbps. Προκειμένου να αντιστοιχίσουμε την εισερχόμενη κίνηση Ethernet στις αλυσίδες επεξεργασίας βασικής ζώνης και επίσης να ανακτήσουμε τα πακέτα Ethernet από τα αποδιαμορφωμένα σύμβολα OFDM, σχεδιάζουμε και αναπτύσσουμε προσαρμοσμένες λειτουργίες L2. Η αρχιτεκτονική των λειτουργιών L2 για την πλευρά λήψης Ethernet (ή εκπομπής RF) απεικονίζεται στο Σχήμα 21α. Στο συγκεκριμένο υποσύστημα πραγματοποιείται ο αρχικός έλεγχος σφαλμάτων των εισερχόμενων πακέτων Ethernet. Έπειτα με βάσει το VLAN του Ethernet πακέτου αυτό δρομολογείται σε μία από τις 4 αλυσίδες επεξεργασίας Tx-OFDM αφού πρώτα υπολογίσουμε μια λέξη 32bit η οποία προστίθεται ως κεφαλίδα στο Ethernet πακέτο για προστασία κατά την διάρκεια μετάδοσης μέσα από την οπτική ίνα και την κεραία. Η αρχιτεκτονική των λειτουργιών L2 για την πλευρά λήψης RF (ή εκπομπής Ethernet) παρουσιάζεται στο Σχήμα 21β. Στο συγκεκριμένο υποσύστημα πραγματοποιείται ο συγχρονισμός των πακέτων Ethernet που λαμβάνονται από τις 4 Rx-OFDM αλυσίδες επεξεργασίας. Αρχικά ελέγχεται η πρώτη 32bit λέξη, που αντιστοιχεί στην κεφαλίδα που προσθέσαμε στον Tx, και με βάση τις πληροφορίες που περιέχει και κάποιες που υπολογίζονται τοπικά αντιλαμβάνεται άμα υπήρξε φθορά των δεδομένων κατά την διάρκεια μετάδοσης. Εάν υπήρξε αγνοεί το συγκεκριμένο πακέτο, εάν όχι προωθεί το σωστό πακέτο προς το Ethernet για μετάδοση στο δίκτυο του παρόχου.



Σχήμα 22: Αλυσίδα επεξεργασίας Tx-OFDM.

Καθώς ο αγωγός επεξεργασίας Tx-OFDM είναι λιγότερο πολύπλοκος από τον αγωγό επεξεργασίας Rx-OFDM, θα περιγράψουμε τη λειτουργία του με ενιαίο τρόπο. Η αρχιτεκτονική υλικού του αγωγού επεξεργασίας Tx-OFDM που αναπτύχθηκε απεικονίζεται στο Σχήμα 22. Αρχικά, μια ροή bit δεδομένων τροφοδοτείται στον Scrambler, ο οποίος επεξεργάζεται τα δεδομένα και εξάγει μια νέα ροή δυαδικών δεδομένων. Στη συνέχεια, ο διαμορφωτής QAM δέχεται ομάδες bit και, με βάση ένα εξωτερικό σήμα ελέγχου, αντιστοιχίζει αυτή τη δυαδική ροή σε μιγαδικά σύμβολα I/Q χρησιμοποιώντας διαφορετικά σχήματα διαμόρφωσης. Τα σύμβολα QAM που παράγονται διαβιβάζονται στη συνέχεια στο μπλοκ Insert Pilots & Zeros, όπου συνδυάζονται με τα Pilots σύμβολα και διατάσσονται για να σχηματίσουν ένα σύμβολο OFDM. Επιπλέον, ένα μηδενικά δεδομένα προσαρτώνται στους φέροντες της ζώνης προστασίας κατά τη διάρκεια αυτής της διαδικασίας. Το σύμβολο OFDM επεξεργάζεται στη συνέχεια από έναν IFFT, ο οποίος μετατρέπει το σήμα OFDM από πεδίο της συχνότητας στο πεδίο του χρόνου. Ο IFFT προσθέτει επίσης αυτόματα το κυκλικό πρόθεμα (CP). Στη συνέχεια, το σήμα OFDM στο πεδίο του χρόνου, μαζί με το κυκλικό του πρόθεμα, αποθηκεύεται προσωρινά σε μία FIFO εντός της μονάδας Insert Preamble πριν προωθηθεί στον DAC. Η μονάδα Insert Preamble είναι υπεύθυνη για την προσθήκη ενός συμβόλου προοιμίου (Preamble) κάθε εννέα OFDM σύμβολα στο πεδίο του χρόνου (π.χ. P123456789P123456789...). Κάθε μπλοκ εντός τουTx-OFDM έχει σχεδιαστεί με πλήρη διοχέτευση και ελάχιστη προσωρινή αποθήκευση δεδομένων, εξασφαλίζοντας ρυθμό μετάδοσης ενός δεδομένου ανά κύκλο ρολογιού και διευκολύνοντας τη συνεχή ροή δεδομένων από το L2 στην είσοδο του DAC.

4.3 Αξιολόγηση Επεξεργαστή Βασικής Ζώνης

Κατανάλωση Πόρων

Η σύνθεση και υλοποίηση των κυκλωμάτων έγινε στην πλατφόρμα Zynq UltraScale+ RFSoC στην οποία γίνεται χρήση 2 DAC και ADC με συχνότητα δειγματοληψίας λίγο πάνω από τα 4Gsps, ενώ οι 2 μπάντες βασική ζώνης μεταφράζονταν στα 1.25GHz και 1.75GHz. Όπως φαίνεται στον Πίνακα 11, για το Tx-OFDM, η πλειονότητα των μπλοκ καταναλώνει αμελητέα ποσότητα πόρων. Το μπλοκ IFFT αποτελεί εξαίρεση και καταναλώνει $\sim 29\%$ περισσότερα LUTs και $\sim 11\%$ περισσότερα FFs από το μπλοκ Pilot & Zeros, το οποίο έχουν την υψηλότερη χρήση μεταξύ των άλλων μπλοκ. Επιπλέον, παρατηρούμε ότι το μπλοκ IFFT είναι το πιο υπολογιστικά απαιτητικό μπλοκ στον πομπό και, ως εκ τούτου, είναι το μόνο μπλοκ που χρησιμοποιεί DSPs. Από την άλλη πλευρά, στον δέκτη, παρατηρούμε μια πιο ομοιόμορφη κατανομή των πόρων γενικής χρήσης (LUTs, FFs) μεταξύ των διαφόρων μπλοκ. Λαμβάνοντας υπόψη ότι ο δέκτης είναι συνήθως το πιο υπολογιστικά απαιτητικό τμήμα μιας μονάδας επεξεργασίας βασικής ζώνης, παρατηρούμε αυξημένη χρήση πόρων σε σύγκριση με τον πομπό. Στον δέκτη, ο αποδιαμορφωτής QAM ξεχωρίζει με την υψηλότερη χρήση DSP, ξεπερνώντας ακόμη και το μπλοκ FFT. Αυτό οφείλεται στο γεγονός ότι η αρχιτεκτονική που υλοποιήθηκε έχει σχεδιαστεί για QAM64, ακόμη και όταν χρησιμοποιούνται χαμηλότερα σχήματα διαμόρφωσης, γεγονός που οδηγεί σε αυξημένες υπολογιστικούς πόρους. Ενώ κάθε μεμονωμένο μπλοκ μπορεί να επιτύχει συχνότητες λειτουργίας 400MHz και άνω, η τελική συχνότητα λειτουργίας του συνδυασμένου συστήματος είναι 256MHz περιορισμός που προέρχεται και από την παραμετροποίηση των DAC/ADC και το ρυθμό δειγματοληψίας τους. Τέλος βλέπουμε ότι η συνολική κατανάλωση σε πόρους του συστήματος είναι σημαντικά χαμηλή, δεδομένου ότι το πλήρες σύστημα χρησιμοποιεί 4×Tx-OFDM και 4×Rx-OFDM αλυσίδες επεξεργασίας.

Πειραματική Αξιολόγηση σε Πραγματικές Συνθήκες

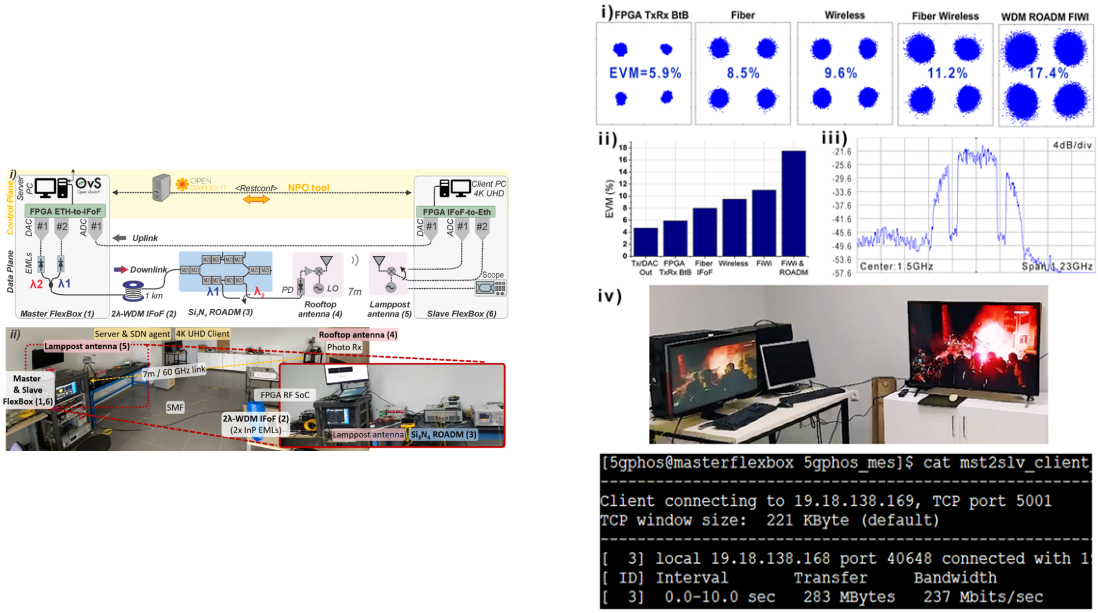
Συνεχίζοντας την αξιολόγησή μας, προχωρήσαμε ένα βήμα παραπέρα και ενσωματώσαμε την αναπτυχθείσα μονάδα βασικής ζώνης RFSoC σε δύο πειραματικές δοκιμές σε πραγματικό συνθήκες στο πλαίσιο του ευρωπαϊκού έργου 5G-PHOS [25].

Η πρώτη πειραματική διάταξη [23], η οποία έχει να κάνει με την παροχή υπηρεσιών σε πραγματικό χρόνο, και συγκεκριμένα 4K video streaming δίνεται διαγραμματικά στο Σχήμα 23α αριστερά πάνω ενώ η φυσική διάταξη παρουσιάζεται στο ίδιο σχήμα αριστερά κάτω. Στην συγκεκριμένη πειραματική διάταξη η RFSoC-based BBU που υλοποιήσαμε συνδέει δύο υπολογιστές για την μετάδοση από τον σερβερ στον πελάτη 4K βίντεο. Το downlink

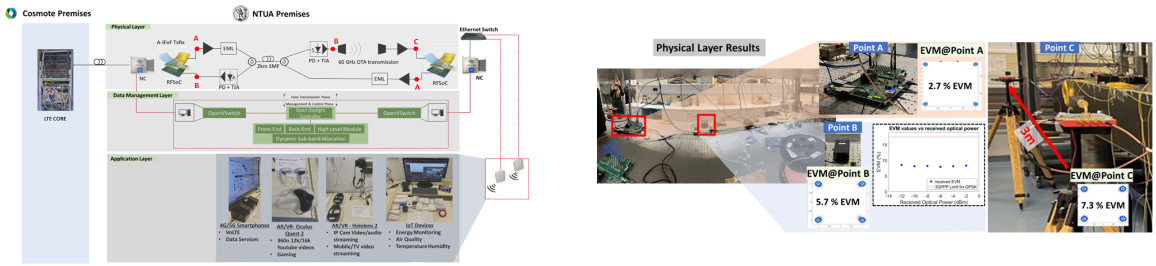
Πίνακας 11: Κατανάλωση πόρων στο XCZU28DR[1] RFSoC

Component	**LUT**	**DFF**	**DSP**	**RAMB**	**Freq. (MHz)**
QAM Mod.	103	176	0	0.5	500
Pilots & Zeros	430	329	0	0.5	500
IFFT	1488	2949	30	4	500
Preamble	170	117	0	1	500
Time Sync.	8799	55434	66	0	530
FFT	1214	3174	30	3.5	500
Zero Removal	100	91	0	1	500
SFO	3398	3583	15	0.5	500
Channel Est. & Eq.	1451	18	18	1.5	400
QAM demod	1554	2523	128	0	500
(De-)Scrambler	82	98	0	0	500
Total Tx+Rx	17971	70765	287	12.5	
L2 Tx+Rx	1170	1894	0	8	
10G Eth. (2 lanes)	23595	55301	0	5.5	
Misc. (AXI, RF, etc.)	6800	8487	0	5	
4x(Tx+Rx)+10G Eth.+(L2 Tx+Rx)	90055 (21%)	329506 (39%)	1106 (26%)	62 (6%)	256

[1] Total resources: FF = 850K, LUT = 425K, BRAM = 1K, DSP = 4.2K



(α) End-to-End Real-Time Service Provisioning [23]



(β) FPGA-based TxRx for Analog-IFoF/mmWave RAN [24]

Σχήμα 23: Πειραματικές διατάξεις αξιολόγησης & αποτελέσματα.

κανάλι που μεταφέρει το βίντεο από τον σερβερ στον πελάτη περιέχει οπτική και ασύρματη μετάδοση. Σχετικά με αποτελέσματα που αφορούν την απόδοση της BBU που υλοποιήσαμε, στο Σχήμα 23α δεξιά πάνω παρουσιάζουμε το μετρούμενο EVM σε διάφορα σημεία της διάταξης. Αυτό που παρατηρούμε είναι ότι το EVM για QAM4 μετάδοση σε όλες τις περιπτώσεις είναι κάτω από τις 3GPP προδιαγραφές (17.5%). Ενώ στο ίδιο σχήμα δεξιά κάτω βλέπουμε ότι μια αλυσίδα επεξεργασίας Tx/Rx μπορεί να υποστηρίξει Ethernet κίνηση ∼240Mbps.

Η δεύτερη πειραματική διάταξη [24] έχει να κάνει με την παροχή υπηρεσιών κινητής τηλεφωνίας σε πραγματικό χρόνο από κάποιον πάροχο τηλεπικοινωνιακών υπηρεσιών που στην συγκεκριμένη περίπτωση είναι η COSMOTE. Στο Σχήμα 23β αριστερά παρουσιάζεται η διάταξη, η οποία περιλαμβάνει το φυσικό επίπεδο, το επίπεδο διαχείρισης δεδομένων και το επίπεδο εφαρμογής. Στην συγκεκριμένη πειραματική διάταξη η RFSoC-based BBU που υλοποιήσαμε τοποθετείτε στο Data Transmission Plane και συνδέεται μεσω SFP με μία κάρτα δικτύου με την οποία ανταλλάσσουν Ethernet κίνηση. Προς το αναλογικό frontend τροφοδοτεί μέσω SMA καλωδίου OFDM σήματα στο οπτικό και ασύρματο λινκ. Σε αυτή τη διάταξη τα κανάλια downlink και uplink πολυπλέκονται ταυτόχρονα σε μία οπτική ίνα. Σχετικά με αποτελέσματα που αφορούν την απόδοση της BBU που υλοποιήσαμε, στο Σχήμα 23β δεξιά παρουσιάζουμε το μετρούμενο EVM σε διάφορα σημεία της διάταξης. Το σημείο A είναι ακριβώς στην έξοδο του DAC του RFSoC, το σημείο B στην έξοδο της φωτοδιόδου μετά από την οπτική μετάδοση και το σημείο C είναι στην έξοδο της κεραίας λήψης. Αυτό που παρατηρούμε είναι ότι το EVM για QAM4 μετάδοση σε όλες τις περιπτώσεις είναι πολύ κάτω από τις 3GPP προδιαγραφές (17.5%).

4.4 Συμπεράσματα

Στο κεφάλαιο αυτό παρουσιάστηκε ο σχεδιασμός μιας BBU με χρήση μετάδοσης OFDM, με έμφαση στα κινητά δίκτυα 5G/B5G. Το σύστημα υλοποιήθηκε με επιτυχία σε μια συσκευή RFSoC, η οποία διαθέτει 4 αλυσίδες επεξεργασίας Tx-OFDM και 4 αλυσίδες επεξεργασίας Rx-OFDM. Επιπλέον, παρουσιάστηκες μια επισκόπηση δύο πειραματικών διατάξεων όπου η BBU ενσωματώθηκε για να χειριστεί την επεξεργασία σημάτων σε χαμηλού επιπέδου. Και στις δύο δοκιμές, η μονάδα λειτούργησε απρόσκοπτα χωρίς να προκαλέσει καμία διακοπή στις υπηρεσίες που εκτελούνται πάνω από την υποδομή κινητής τηλεφωνίας.

5 Συμπεράσματα Διατριβής & Μελλοντικές Επεκτάσεις

5.1 Περίληψη Διατριβής

Οι επαναδιαμορφώσιμες συσκευές χρησιμεύουν ως γέφυρα μεταξύ των ASIC υψηλής απόδοσης και των δυνατοτήτων των συμβατικών πλατφορμών επεξεργαστών, ιδίως σε εφαρμογές εντατικών υπολογισμών όπως η ψηφιακή επεξεργασία σήματος (DSP). Οι συσκευές αυτές προσφέρουν ευελιξία και ταχύτερο χρόνο διάθεσης στην αγορά. Ωστόσο, η σχεδίαση με επαναδιαμορφώσιμες συσκευές μπορεί να αποτελέσει πρόκληση λόγω της ενσωμάτωσης πολλαπλών εργαλείων σχεδίασης και ειδικών αρχιτεκτονικών, γεγονός που δημιουργεί δυσκολίες για τους σχεδιαστές. Η σχεδίαση επαναδιαμορφώσιμου υλικού για εφαρμογές DSP έχει θεωρηθεί δύσκολη, οπότε για την αντιμετώπιση αυτών των προκλήσεων, οι ερευνητές αναζητούν αποτελεσματικές μεθόδους σχεδίασης που όχι μόνο λαμβάνουν υπόψη τις αρχιτεκτονικές FPGA αλλά και απλοποιούν τα βήματα σχεδίασης.

Τα FPGA προσφέρουν πλέον μια οικονομικά αποδοτική λύση για την υλοποίηση DSP σε διάφορες εφαρμογές, όπως η επεξεργασία εικόνας, οι ασύρματες επικοινωνίες, τα συστήματα πολυμέσων και τα ηλεκτρονικά είδη ευρείας κατανάλωσης. Οι κορυφαίοι κατασκευαστές FPGA δεν ενσωματώνουν στις συσκευές τους μόνο απλές λειτουργίες DSP, αλλά και προηγμένα εξειδικευμένα μπλοκ και CPU γενικού σκοπού. Αυτά τα χαρακτηριστικά, σε συνδυασμό με την on-chip μνήμη, αυξάνουν σημαντικά την απόδοση του συστήματος, ξεπερνώντας τις συμβατικές πλατφόρμες επεξεργασίας.

Η παρούσα διατριβή παρουσιάζει αποδοτικές αρχιτεκτονικές σε SoC-FPGAs για απαιτητικές εφαρμογές, που εκτείνονται από την επεξεργασία εικόνας/βίντεο στον τομέα της ιατρικής απεικόνισης έως την αυτόνομη πλοήγηση στο διάστημα. Επιπλέον, προτείνει μια προσέγγιση σχεδιασμού τριών πυλώνων για ενσωματωμένες εφαρμογές σε SoC-FPGA, με στόχο την επίτευξη τριών κύριων στόχων: 1) σημαντική αύξηση της απόδοσης του συστήματος, 2) μείωση της χρήσης των πόρων και της κατανάλωσης ενέργειας και 3) ταυτόχρονη εκτέλεση πολλαπλών εφαρμογών σε μία μόνο συσκευή. Πιο αναλυτικά:

- **Συνεργατική Προσέγγιση Σχεδίασης για Πλατφόρμες SoC-FPGA:** Παρουσιάσαμε μια σχεδιαστική προσέγγιση που περιελάμβανε εκτεταμένο DSE τόσο σε αλγοριθμικό όσο και σε αρχιτεκτονικό επίπεδο για να βελτιώσουμε τις υλοποιήσεις των κυκλωμάτων. Εφαρμόσαμε συστηματικά κλιμάκωση των τάσεων τροφοδοσίας για να μειώσουμε την κατανάλωση ισχύος σε επίπεδο συστήματος εντός του SoC-FPGA και πραγματοποιήσαμε μια αρχική διερεύνηση της συνεκμετάλλευσης ενός SoC-FPGA από δύο απαιτητικές εφαρμογών. Τα πειράματά μας απέδωσαν τα ακόλουθα αποτελέσματα: 1) Για τη βελτιστοποίηση του κυκλώματος (αποδιαμορφω-

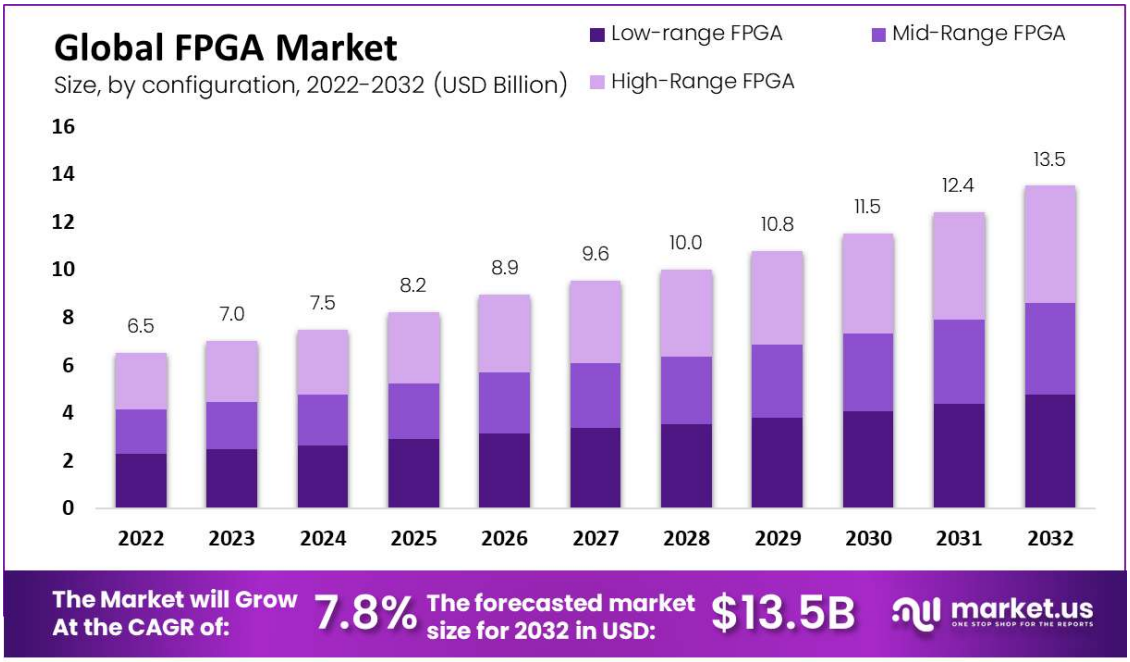
τής QAM), πετύχαμε έως και 98% μείωση στη χρήση των πόρων σε σύγκριση με την ακριβή αρχιτεκτονική κινητής υποδιαστολής του ίδιου αλγορίθμου, ενώ αυξήσαμε τη συχνότητα λειτουργίας του έως και 12%. 2) Ανάλογα με τις αρχιτεκτονικές επιλογές για τον επιταχυντή σε εφαρμογή HW/SW που εκτελείτε σε συσκευή SoC-FPGA, επιτύχαμε εξοικονόμηση ισχύος $20-30\%$ σε επίπεδο συστήματος, διατηρώντας παράλληλα τη λειτουργική ορθότητα και την ονομαστική χρονική απόδοση του συστήματος. 3) Καταδείξαμε τη δυνατότητα ενσωμάτωσης λειτουργιών DSP χαμηλού επιπέδου που απαιτούνται σε μια BBU με επιτάχυνση AI/ML σε μια συσκευή SoC-FPGA τελευταίας τεχνολογίας (RFSoC). Η αρχική μας αξιολόγηση δείχνει ότι αυτός ο συνδυασμός είναι βιώσιμος όσον αφορά την κατανάλωση πόρων και τις μετρήσεις επιδόσεων.

- **Αποδοτικές Αρχιτεκτονικές για Εφαρμογές Εικόνας/Βίντεο σε Σύγχρονες Συσκευές SoC-FPGA:** Αναπτύξαμε αποδοτικά ενσωματωμένα συστήματα HW/SW για εφαρμογές εικόνας/βίντεο και τα αναπτύξαμε σε σύγχρονες συσκευές SoC-FPGA. Η εφαρμογή Image Registration, όταν υλοποιήθηκε σε ένα σύγχρονο SoC-FPGA, επέδειξε ανώτερες επιδόσεις σε σύγκριση με την αντίστοιχη εφαρμπγή λογισμικού, επιτυγχάνοντας έως και $67\times$ επιτάχυνση σε σχέση με ενσωματωμένες CPUs, ενώ χρησιμοποιεί αριθμητική κινητής υποδιαστολής μονής ακρίβειας. Στην περίπτωση της εφαρμογή VBN, πετύχαμε επιτάχυνση $19\times$ με μέση κατανάλωση ισχύος μικρότερη από 5W και μέσο σφάλμα 0.5% της απόστασης μεταξύ της κάμερας και του δορυφόρου-στόχου.

- **Επεξεργασία Βασικής Ζώνης σε RFSoC-FPGA για δίκτυα κινητών επικοινωνιών 5G/B5G:** Σχεδιάσαμε αποδοτικές αρχιτεκτονικές υλικού για βασικά μπλοκ DSP που χρησιμοποιούνται στη μετάδοση και λήψη OFDM σε μια BBU. Με την ανάπτυξη πολλαπλών παράλληλων αλυσίδων επεξεργασίας Tx/Rx, επιτύχαμε υπηρεσίες κινητής τηλεφωνίας και ροή βίντεο υψηλής ευκρίνειας σε τελικούς χρήστες σε δύο πειραματικές δοκιμές. Η BBU μας επέδειξε εξαιρετική απόδοση, λειτουργώντας με EVM πολύ κάτω από τις προδιαγραφές 3GPP.

5.2 Μελλοντικές Επεκτάσεις

Με την πάροδο των ετών, η τεχνολογία FPGA έχει εξελιχθεί, προσφέροντας ταχύτερους χρόνους διεκπεραίωσης, μειωμένο κόστος και μικρότερη κατανάλωση ενέργειας σε σύγκριση με τα ASIC. Η ευελιξία των FPGA ξεχωρίζει, καθώς μπορούν να επαναπρογραμματιστούν ακόμη και μετά τον αρχικό σχεδιασμό και την αρχική ρύθμιση. Αυτή η προσαρμοστικότητα, η οποία επιτρέπει τροποποιήσεις στο σχεδιασμό ακόμη και μετά την εγκατάσταση ενός προϊόντος, αναμένεται να οδηγήσει στην ανάπτυξη της αγοράς, όπως φαίνεται στο Σχήμα 24.



Σχήμα 24: Προβλέψεις για την ανάπτυξη της αγοράς FPGA.

Αυτή η διδακτορική διατριβή έδειξε τις τεράστιες δυνατότητες των FPGA ως ψηφιακής πλατφόρμας που καλύπτει διάφορους τομείς εφαρμογών. Επιπλέον, με την ραγδαία εξέλιξη σε τομείς όπως το AI/ML και το 5G/B5G, η ενσωμάτωση των FPGA για αυτούς τους καινοτόμους τομείς φαίνεται μια λογική απόφαση. Παρόλα αυτά, υπάρχουν ανησυχίες, κυρίως όσον αφορά τις μεθοδολογίες ανάπτυξης συστημάτων βασισμένα σε FPGA και τον καθορισμό των εφαρμογών που πρέπει να αναπτυχθούν και να διαχειριστούν σε αυτές τις όλο και πιο ετερογενείς πλατφόρμες. Ως εκ τούτου, αυτή η έρευνα αυτής της διδακτορικής διατριβής μπορεί να επεκταθεί σε πολλά πιθανά μονοπάτια για βαθύτερη διερεύνηση, όπως περιγράφεται παρακάτω.

- **Αυτοματοποίηση της πρτοτεινόμενης προσέγγισης σχεδιασμού & ενσωμάτωση σε εμπορικά εργαλεία**

  Μια δυνατότητα είναι η αυτοματοποίηση της προτεινόμενης προσέγγισης σχεδιασμού, που παρουσιάστηκε στην Ενότητα 2, επιτρέποντας στους σχεδιαστές να διερευνούν αποτελεσματικότερα διάφορες αρχιτεκτονικές εναλλακτικές λύσεις αντί να εφαρμόζουν την προσέγγιση χειροκίνητα. Επιπλέον, μια σημαντική πρόοδος θα ήταν η ενσωμάτωση της προτεινόμενης προσέγγισης σχεδιασμού σε εμπορικά διαθέσιμα εργαλεία. Αυτό θα ενίσχυε την προσβασιμότητα και την υιοθέτηση της προσέγγισης σε σενάρια σχε-

διασμού συστημάτων στον πραγματικό κόσμο.

- **Εξερεύνηση προσεγγιστικών αριθμητικών κυκλωμάτων στις τηλεπικοινωνίες**

  Στην παρούσα διατριβή, επικεντρωθήκαμε στην ανάπτυξη αποδοτικών υλοποιήσεων για SoC-FPGAs και RFSoCs με χειροκίνητη λεπτομερή ρύθμιση των κυκλωμάτων τους. Μια πιθανή μελλοντική κατεύθυνση είναι η διερεύνηση της δυνατότητας εφαρμογής των τεχνικών προσεγγιστικών κυκλωμάτων σε ένα ευρύτερο φάσμα DSP μπλοκ, ιδίως στον τομέα των τηλεπικοινωνιών, όπου πολλά DSP μπλοκ μπορεί να παρουσιάζουν ανθεκτικότητα σε σφάλματα λόγω της φύσης των υπολογισμών τους. Αυτή η διερεύνηση θα μπορούσε να οδηγήσει σε βελτιστοποιημένα κυκλώματα για τηλεπικοινωνιακές εφαρμογές.

- **Συστηματική εφαργογή AI/ML στις τηλεπικοινωνίες**

  Καθώς οι εφαρμογές AI/ML αποκτούν μεγαλύτερη έλξη, μια πολλά υποσχόμενη ερευνητική κατεύθυνση είναι η διεξοδική διερεύνηση της ανάπτυξης αλγορίθμων μηχανικής μάθησης στον τομέα των τηλεπικοινωνιών. Αν και υπάρχει σημαντικό ενδιαφέρον μεταξύ των ερευνητών για την εφαρμογή τεχνικών AI/ML στην επεξεργασία βασικής ζώνης, η ανάπτυξη σε πραγματικό υλικό παραμένει περιορισμένη. Η διερεύνηση και η εφαρμογή αλγορίθμων AI/ML στο υλικό μπορεί να ανοίξει νέες δυνατότητες για τη βελτίωση της απόδοσης και της αποδοτικότητας στα τηλεπικοινωνιακά συστήματα.

- **Ετερογενείς πλατφόρμες υπολογισμού**

  Η αυξανόμενη διαθεσιμότητα διαφορετικών και εξαιρετικά ετερογενών υπολογιστικών πλατφορμών, όπως η Versal ACAP, παρουσιάζει νέες προκλήσεις στην αποτελεσματική αντιστοίχιση εφαρμογών σε αυτές τις πλατφόρμες. Μια πιθανή μελλοντική ερευνητική κατεύθυνση είναι η διερεύνηση μεθοδολογιών την αποτελεσματική απεικόνιση σύνθετων εφαρμογών DSP σε αυτές τις σύγχρονες πλατφόρμες χωρίς να απαιτείται βαθιά κατανόηση των υποκείμενων αρχιτεκτονικών τους. Επιπλέον, η διερεύνηση της ανάπτυξης αυτόματου προγραμματισμού και διαχείρισης για πολλαπλές εφαρμογές ως επεκτάσεις αυτών των μεθοδολογιών θα είχε μεγάλο ενδιαφέρον. Τέτοιες εξελίξεις θα μπορούσαν να βελτιώσουν την ανάπτυξη εφαρμογών σε αυτές τις πλατφόρμες αιχμής, διευκολύνοντας τους προγραμματιστές να αξιοποιήσουν πλήρως τις δυνατότητές τους, βελτιστοποιώντας παράλληλα τις επιδόσεις και τη χρήση πόρων.

Contents

Abstract v

Abstract in Greek vii

Acknowledgments ix

Brief Description of the Proposed Systems in Greek xi

List of Figures lix

List of Tables lxi

1 Introduction 1
- 1.1 Embedded Computing Systems . . . 1
- 1.2 Reconfigurable Devices - FPGAs . . . 3
- 1.3 Thesis Scope &Contributions . . . 4

2 A Synergistic Design Approach for SoC-FPGA Platforms 9
- 2.1 Introduction . . . 10
- 2.2 Exploring the Design Space of DSP Circuits: Algorithms, Arithmetic, and Approximation Techniques . . . 13
  - 2.2.1 Introduction . . . 13
  - 2.2.2 Related Work . . . 14
  - 2.2.3 Design of Approximate QAM Demodulation Circuits . . . 15
  - 2.2.4 Experimental Evaluation . . . 23
- 2.3 Power Consumption Optimization in SoC-FPGA Chips . . . 29
  - 2.3.1 Introduction . . . 29
  - 2.3.2 Background . . . 30
  - 2.3.3 System Description . . . 31
  - 2.3.4 System Evaluation . . . 34
- 2.4 Enabling SoC-FPGA Sharing: Bridging Low-Level and High-Level Computing at the Edge . . . 42
  - 2.4.1 Introduction . . . 42

2.4.2 Proposed Architecture . . . 43
2.4.3 Baseband Processing Unit (BBU) Partition . . . 44
2.4.4 AI/ML Acceleration Partition . . . 45
2.4.5 Preliminary Evaluation . . . 47
2.5 Conclusions . . . 49

3 Efficient Architectures for Image/Video Applications in Contemporary SoC-FPGA devices 51
3.1 Introduction . . . 52
3.2 Optimizing Image Registration for Embedded Systems using FPGA-based SoCs . . . 52
3.2.1 Introduction . . . 52
3.2.2 Background . . . 54
3.2.3 Application Profiling &HW/SW Partitioning . . . 55
3.2.4 Proposed HW/SW Image Registration Architecture . . . 58
3.2.5 Experimental Evaluation . . . 60
3.3 Embedded SoC-FPGA Implementation for Vision-Based Navigation in Space 65
3.3.1 Introduction . . . 65
3.3.2 Related work . . . 66
3.3.3 Exploration Methodology &System Description . . . 68
3.3.4 Algorithm Design . . . 71
3.3.5 Development on SoC-FPGA . . . 76
3.3.6 System Evaluation . . . 89
3.4 Conclusions . . . 94

4 Baseband Processing in RFSoC FPGA for 5G/B5G Mobile Networks 97
4.1 Introduction . . . 98
4.2 Rationale of selecting RFSoC FPGA . . . 99
4.3 Physical Layer DSP for OFDM Transmission . . . 101
4.3.1 Centralized DSP engine . . . 101
4.3.2 Baseband Unit Digital Architecture . . . 101
4.3.3 PHY L2 Layer . . . 105
4.3.4 PHY L1 Layer DSP . . . 108
4.4 Real-world experimental evaluation . . . 116
4.5 Conclusions . . . 121

5 Conclusions 123
5.1 Thesis Summary . . . 123
5.2 Future Work . . . 125

Abbreviations &Acronyms 127

Publications **133**

Bibliography **137**

List of Figures

1.1 Comparison between computing platforms. 2
1.2 Growth in processor performance over 40 years [6] 4
1.3 FPGA Evolution: From FPGA to SoC to ACAP [9] 5
1.4 Ph.D thesis scope overview. 6

2.1 Proposed design approach for efficient and flexible SoC-FPGA platforms. 11
2.2 M-QAM demodulation architecture of the exact log likelihood ratio (ELLR) algorithm. 16
2.3 M-QAM demodulation architecture of the approximate log likelihood ratio (ALLR) algorithm. 17
2.4 M-QAM demodulation architecture of the piecewise log likelihood ratio (PLLR) algorithm. 18
2.5 M-QAM demodulation architecture of approximate maximum likelihood (AML). 19
2.6 Approximate circuit of floating-point (FLP) adder. 20
2.7 Approximate fixed-point (FXP) multiplier. 21
2.8 BER scaling in our 64-QAM circuits. FLP = accurate floating-point, FLP-W4 = floating-point with approximate addition [16], FXP = accurate fixed-point, FXP-TX = fixed-point with bit truncation, FXP-K6 = fixed-point with approximate multiplication [17]. 24
2.9 Pareto analysis of the proposed 64-QAM circuits in terms of accuracy and hardware resources. 28
2.10 Hardware architecture of Harris IP core. 32
2.11 Finalized architecture used for voltage scaling and guard-band exploration. 35
2.12 Voltage scaling results of V_{CCINT} in function of PL frequency. 38
2.13 Total power consumption of various PS-PL system configurations. 40
2.14 SoC FPGA partitioning to accommodate low-level PHY processing and AI acceleration with HW/SW co-processing. 43
2.15 An abstract model of an FPGA implementation in circuit level derived from a neural network topology [26] 46

3.1 Registration example using the iris dataset 55
3.2 Processing flow of a general registration solver. 57
3.3 HW/SW partitioning of the Image Registration Solver. 59
3.4 Finalized image registration solver architecture. 61
3.5 Performance evaluation of the proposed HW/SW design 64
3.6 Proposed system on top of Zynq-7000 SoC-FPGA for VBN support. . . . 70
3.7 Proposed pose tracking algorithm: main functions &example dataflow . . 72
3.8 Mesh model used for tracking (left) and depth edges detected after rendering with the estimated pose, overlaid in red on an input frame (right). . . 74
3.9 Finalized HW architecture at function/data level. 82
3.10 Proposed HW architecture of Canny, with deep pipelining, on-the-fly processing, limited use of buffers, and recursive edge tracing with a small stack. 85
3.11 Proposed HW architecture of rendering, with 4x parallelization at pixel level, floating-point &deeply-pipelined components, and parallel memories. 87
3.12 Pose estimation (alignment error) of our HW/SW system for three ENVISAT distances. The HW/SW accuracy is practically equal to that of SW (cf. 3 similar versions plotted with thin lines). In all 3 cases, error $< 40cm$. 90

4.1 Evolution of FPGA-based BBUs. 99
4.2 Block diagrams of typical OFDM transmitter and receiver DSP chain . . 102
4.3 High-level architecture of DSP processing on Zynq Ultrascale+ RFSoC device. 103
4.4 L2 architecture for Ethernet to Tx-DSP mapping. 106
4.5 L2 architecture for Rx-DSP to Ethernet mapping. 107
4.6 Tx-OFDM processibg pipeline. 108
4.7 Time domain plot of preamble signal. 109
4.8 Time synchronization architecture. 110
4.9 Phase shift estimation and compensation architecture. 111
4.10 Channel estimation &equalization architecture. 112
4.11 Approximate LLR M-QAM demodulator architecture. 113
4.12 Expiremental setup for End-to-End Real-Time Service Provisioning [23]. . 116
4.13 Experimental results (EVM, RF spectrum, iperf tcp) [23]. 118
4.14 Experimental setup for Analog-IFoF/mmWave RAN in MNO's infrastructure [24]. 119
4.15 Experimental results (EVM) in different point of the network [24]. 121

5.1 FPGA market growth projection. 125

List of Tables

2.1 Indicative examples of various system deployments on a SoC-FPGA. . . . 12
2.2 Resources of our 64-QAM circuits on Xilinx Zynq Ultrascale+ XCZU7EV[1] (ZCU106) 25
2.3 Power consumption results for individual regulation of PS voltage rails. . 37
2.4 Power consumption results for combined regulation of PS voltage rails. . . 37
2.5 Power consumption results for individual regulation of PL voltage rails for 100MHz operating frequency. 39
2.6 Power consumption results for combined regulation of PS and PL voltage rails for 100MHz operating frequency. 40
2.7 Accuracy and performance benchmark accross different DNNs and datasets 48
2.8 Resource utilization on XCZU28DR[1] RFSoC 49
2.9 Performance-Utilization trade-off on XCZU28DR 50

3.1 Computed similarity measure for various maximum displacement. Scaling and rotation are fixed at 10% and 30° respectively. 56
3.2 Profiling of registration for two pairs of eye images. 58
3.3 Final affine transformation parameters and similarity measure on three different platforms. 62
3.4 Resource utilization on XC7Z010 Zynq device. 64
3.5 Profiling on ARM A9 (Zynq) for 1 Mpixel images at $30m$ &$50m$ 78
3.6 Final System: Resources on SoC-FPGA XC7X100IFFG900-2L 92

4.1 Resource utilization on XCZU28DR[1] RFSoC 115

Chapter 1

Introduction

1.1 Embedded Computing Systems

An embedded system is a dedicated computing system integrated into a larger product to perform specific tasks within a selected application domain. Reliability, efficiency in terms of size, cost, power consumption, and response times are crucial requirements for embedded systems. Additionally, they must exhibit a high level of dependability, meaning any malfunction is not tolerable. Over the year, as embedded systems achieved higher performance and efficiency, they are incorporating more sophisticated algorithms, evolving into complex systems, and expanding into diverse application domains, enabling the implementation of innovative technologies. As a result of these factors, it has become increasingly difficult to differentiate embedded computer systems from general-purpose ones.

The trend of growing complexity, broader application areas, and higher levels of sophistication, is accompanied by continuous advancements in semiconductor and information technology, making powerful embedded computing widely available at a relatively low cost. Thus, embedding computing systems have become pervasive, aiming to automate and optimize various tasks. The emergence of concepts like Internet of Things (IoT) and Edge computing has increased even more the computational and communication demands on embedded systems [1]. Concurrently, the significant surge of resource-intensive applications and advanced algorithms in areas like Digital Signal Processing (DSP) indicates a new era for the embedded computing systems. Even thought, conventional general-purpose processors, such as CPUs, lack the computational capacity to handle these demanding workloads, resulting in performance limitations [2], embedded systems have also limited memory capacity, which diminish the processing capabilities of such systems. Considering the constraints of low power consumption, a paradox arises: we aim for high-performance computing while using a low-power device. At the same time,

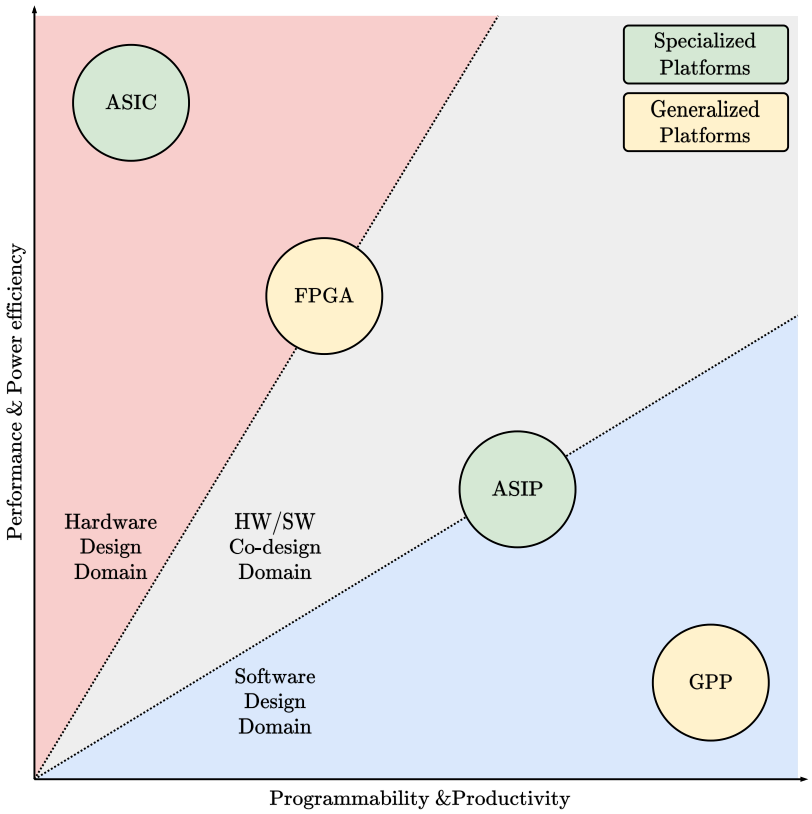


Figure 1.1: Comparison between computing platforms.

the tremendous amount of data created is overwhelming conventional computing systems, resulting in need to incorporate dedicated computing accelerators [3]. Such devices are the Field-Programmable Gate Arrays (FPGAs).

In comparison to other computational platforms, Figure 1.1 demonstrates that FPGAs occupy a position intermediate to the specialized hardware designs of Application-Specific Integrated Circuits (ASICs) and the more development-friendly yet typically slower software paradigms, such as Central Processing Units (CPUs) and Application-Specific Instruction set Processors (ASIPs). ASICs offer an optimised hardware solution tailored to either an application domain or a single application implementation. As a result, they provide better performance and lower power designs compared to FPGAs or software-based implementations. Alternatively, software solutions for generalized platforms are designed to run on general-purpose processors and so are much simpler and faster to develop. Software can easily be updated or improved should errors emerge in the design, or new application or implementation improvements be developed. Whether we view FPGAs

as either a more flexible, reconfigurable version of an ASIC, or as a higher-performing, lower-power platform for software-based designs, FPGAs bring their own benefits and challenges.

1.2 Reconfigurable Devices - FPGAs

In the early stages, FPGAs had limited logic resources, making them primarily used for connecting different logical modules together. They served as interfaces for various off-the-shelf Integrated Circuits (ICs) or self-contained electronic modules, allowing the construction of more complex digital applications. During this period, FPGAs mainly offered logic and memory resources. However, with advancements in semiconductor process technology, modern FPGAs have significantly evolved in terms of chip architecture, logic density, and on-chip hard-core functionalities, thus making them high-performance devices that are able to deliver exceptional performance per Watt [2].

Key characteristics of FPGAs include their inherent parallel architecture, facilitating the deployment of highly parallel designs with impressive throughput rates. Additionally, their low operation frequency contributes to reduced power consumption, while their reconfiguration/re-programmability provides enhanced flexibility and reusability across a broad spectrum of applications and operations. These features make FPGAs a promising solution to address the escalating demands for performance in data center and embedded applications [2, 4]. Despite being more power-demanding compared to ASICs [5], FPGAs are considered the optimal specialized accelerators for numerous use cases, outperforming general-purpose CPUs and GPUs [2]. In continuously expanding application domains, such as Artificial Intelligence (AI)/Machine Learning (ML) and mobile networks (5G/B5G), FPGAs stand out due to their reconfigurable nature. This allows them the flexibility to stay up-to-date with evolving architecture and feature integration in these domains, quickly adapting to changing processing demands, providing system-level solutions with improved performance and ensures rapid time-to-market.

As the demand for more processing performance increased, partly due to the slowdown in processor performance growth (Figure 1.2), in 2011, a new trend emerged on the FPGA market, with the introduction of System-on-Chip (SoC) FPGAs. SoC-FPGAs are heterogeneous platforms that combine traditional FPGA fabric with various processing units like CPUs, GPUs, etc. on a single chip. In addition to the mentioned processing units, FPGA chips can incorporate various other high-performance components, including high-speed

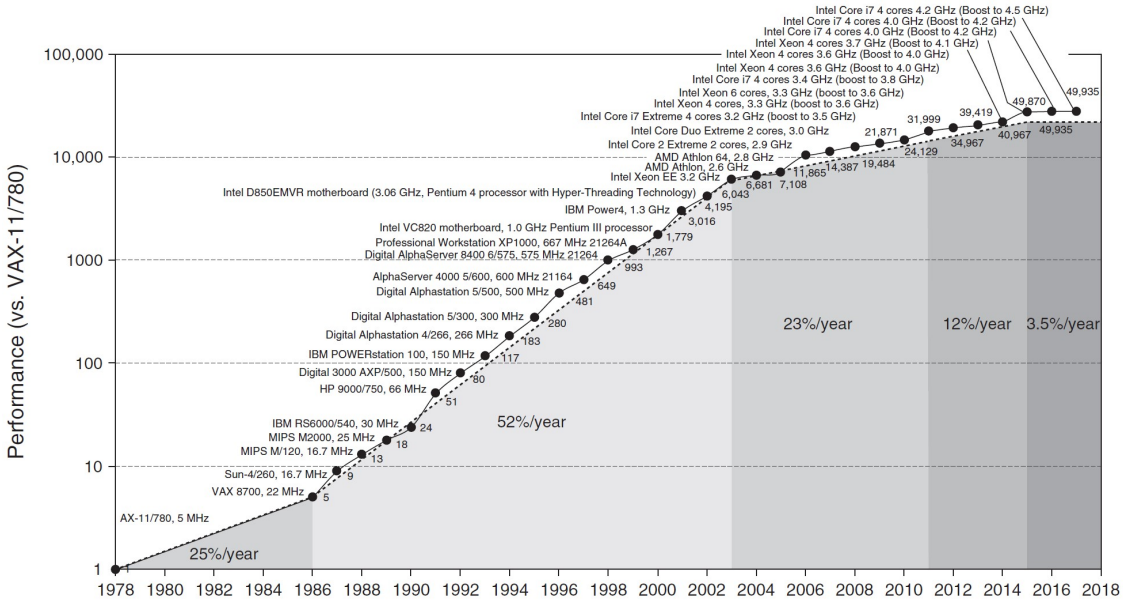


Figure 1.2: Growth in processor performance over 40 years [6]

serial transceivers (exceeding 10 Gbps per lane), customizable external memory interfaces, and integrated interfaces such as SPI, USB, Ethernet, and PCI-express. This integration offers several advantages, such as reduced power consumption, improved reliability, even faster time-to-market due to unified development and debugging, smaller size and weight as multiple components are integrated into a single chip, and most importantly, enhanced performance and adaptability through increased heterogeneity and efficient HW/SW co-design [7]. SoC-FPGAs have become increasingly popular in the embedded world due to their ability to greatly enhance the performance and energy efficiency of traditional CPU-only systems [8]. This trend of developing even more heterogeneous FPGA-based platforms continues even now, with Xilinx (now part of AMD) being the leader (Figure 1.3).

1.3 Thesis Scope & Contributions

In recent years, there has been a remarkable growth in embedded systems, particularly in the consumer electronics sector. The increasing demand for high-performance and low-power systems has driven researchers to develop innovative design techniques to meet these stringent requirements. Many of these systems involve streaming DSP that requires intensive mathematical computations, ranging from basic arithmetic operations

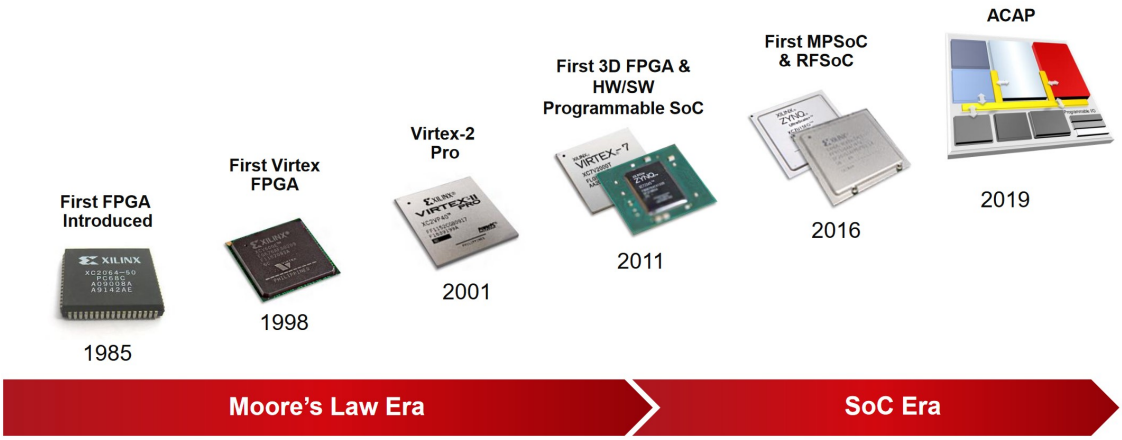


Figure 1.3: FPGA Evolution: From FPGA to SoC to ACAP [9]

to complex tasks like matrix inversion and filtering. As DSP becomes more prevalent in various devices, factors such as time-to-market and the ability to make late design changes have gained importance. Software offers flexibility in design and allows for late changes, but its performance lags behind hardware, which can execute in parallel. Conversely, creating a custom ASIC takes considerable time and is non-reconfigurable once completed.

This is where FPGAs emerge as an ideal solution, leveraging the strengths of both hardware and software. FPGAs combine the flexibility of software with the parallel processing capability of hardware, making them a compelling choice for various applications. However, achieving high performance and energy-efficient digital system implementations remains a persistent challenge, particularly in embedded devices. This necessitates optimization at different levels of design hierarchy. At the higher level of design, efficient algorithms are essential, and at the lower level, the use of suitable architectures can significantly reduce the overall power consumption of the system and increase performance.

In this PhD thesis, our focus lies in accelerating demanding and complex DSP applications on reconfigurable devices, particularly SoC-FPGAs. The applications addressed in this thesis comes from the image/video processing domain and the telecom domain. In the image/video processing domain, our specific focus is on accelerating a Medical Imaging application and a Vision-based navigation application for the space deployments. On the other hand, in the telecom domain, we concentrate on low-level physical layer processing, particularly targeting the next generation 5G/B5G mobile networks. All these applications have strict latency requirements and demand minimal power consumption.

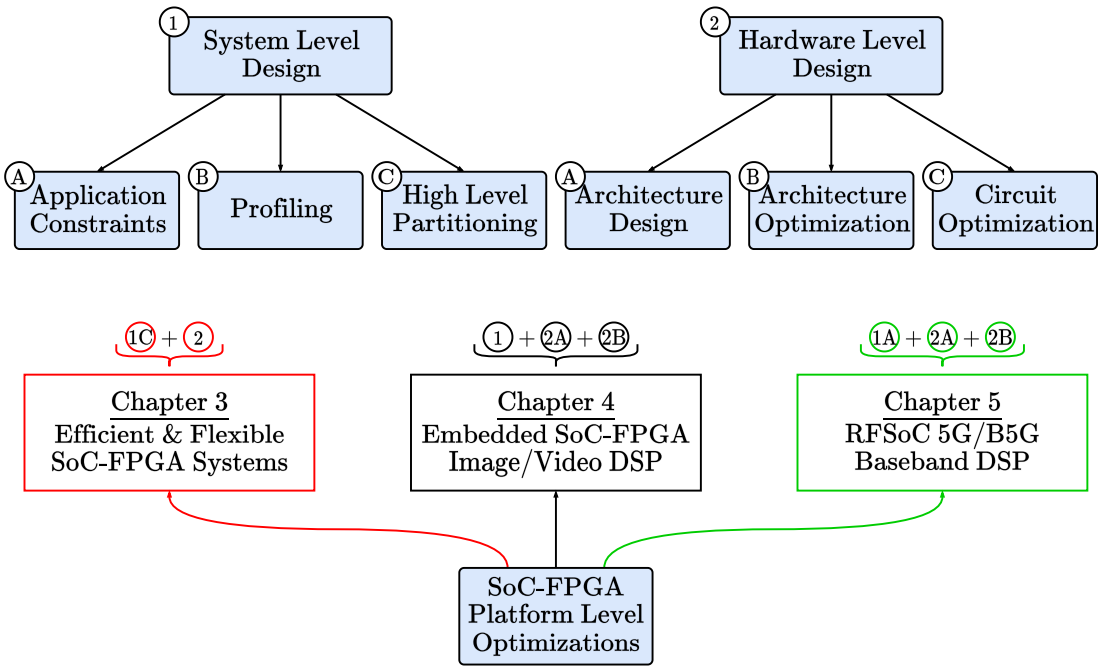


Figure 1.4: Ph.D thesis scope overview.

To reach our objectives, we conduct thorough Design Space Exploration (DSE) and implement advanced design techniques. Additionally, we consider the unique characteristics of the underlying SoC-FPGA device to further enhance our solutions. Figure 1.4 provides an overview of the research conducted in this thesis.

The main contributions of this thesis can be summarised as follows:

Chapter 3: In this chapter, we present a design approach for embedded DSP applications targeting SoC-FPGAs. The initial section focuses on optimizing the design process at different levels, starting from the algorithmic level and extending down to the circuit level. In the subsequent part, we illustrate a systematic procedure for determining the ideal supply voltages, ensuring compliance with the nominal operating conditions specified by the vendor's tools. Lastly, we propose a logical division of a single SoC-FPGA device, enabling the deployment of diverse applications within the same domain (e.g., computing@Edge). Our proposed design methodology allows for the independent application of each part or their complementary use. The ultimate objective is to enhance the efficiency of the developed applications and facilitate flexible deployments of multiple applications on a single SoC-FPGA device.

Chapter 4: This chapter presents high-performance hardware (HW) accelerators designed specifically for image and video applications. The focus is on two distinct domains: Medical Imaging (MI) and Vision-Based Navigation (VBN) in space. These domains impose stringent implementation constraints, requiring real-time operation and low power/energy consumption. Furthermore, the extensive utilization of deeply embedded devices in these domains introduces notable implementation challenges. In the case of Medical Imaging, we explore an Image Registration (IR) processing pipeline, while for Vision-Based Navigation, our target application is the 6D Pose Tracking (PS) of uncooperative objects. For both cases, we conduct an in-depth analysis of the target applications and describe efficient HW/SW partitioned systems for SoC-FPGA devices. Additionally, we present hardware-level optimizations to further enhance performance. Finally, comprehensive performance results are provided, accompanied by comparisons with general-purpose embedded CPUs, enabling a thorough evaluation of the proposed approaches.

Chapter 5: In this chapter, we present a proof-of-concept baseband Frequency-Division Multiplexing (OFDM) processing pipeline targeting 5G/B5G mobile networks based on Centralized-Radio Access Network (C-RAN) and Analog Radio-over-Fiber (A-RoF) concepts. The focus is on developing efficient hardware (HW) transmit (Tx) and receive (Rx) DSP datapaths utilizing OFDM transmission on a single SoC-FPGA device, the Xilinx RFSoC-FPGA. We discuss the hardware architecture of key components in both the Tx and Rx datapaths, highlighting the effective use of dedicated hard-blocks. Furthermore, we explore the potential for parallel deployment of multiple pairs of Tx/Rx processing datapaths to enable multi-band operation, thus improving bandwidth per user or serving multiple users simultaneously. Finally, performance and resource utilization metrics are provided, along with two indicative real-world test cases showcasing the successful deployment and achievement of objectives with the developed Tx/Rx-OFDM processing pipelines on the RFSoC platform.

Since the work conducted in this thesis is diverse in nature, we have chosen not to include a common discussion of related works. Instead, each Chapter has its own Introduction and Related Work sections for better comprehension. Finally, Chapter 5 concludes this thesis by summarizing the presented results and discusses the future extensions of this work.

Chapter 2

A Synergistic Design Approach for SoC-FPGA Platforms

This chapter focuses on enhancing efficiency and flexibility in FPGA-based systems-on-chip (SoC) platforms for digital systems. It tackles challenges related to power/energy consumption, performance optimization, and the integration of heterogeneous components. By employing approximation techniques and exploiting the error resiliency of DSP functions, the chapter showcases significant improvements in throughput and circuit complexity. Moreover, it explores customizing operating parameters in SoC-FPGA chips, allowing fine-tuning of voltage levels for different components. This optimization strategy, based on application requirements, results in improved energy efficiency while maintaining desired performance levels. Additionally, the chapter delves into the integration various digital system applications into sophisticated SoC-FPGAs, such as RFSoC or Versal ACAP devices. It illustrates how these devices serve as versatile accelerators for a wide range of DSP tasks, offering high-performance, re-programmability, and low-power consumption. The discussion covers various aspects, including high-throughput interfaces, HW/SW co-processing capabilities, accelerator deployment, and provides an initial estimation of performance and utilization. In summary, this chapter presents a cohesive design approach that enhances efficiency and flexibility in FPGA-based SoC platforms, contributing to the optimization and advancement of digital systems built upon SoC-FPGAs. This chapter is based on our publications in [27–30]

2.1 Introduction

This chapter presents a synergistic design approach to drive efficiency and flexibility in SoC-FPGA platforms. The primary focus is on enhancing digital signal processing (DSP) tasks within the SoC architecture, addressing challenges related to power/energy consumption, performance optimization, and the integration of heterogeneous components. By leveraging circuit approximation methods, notable improvements in throughput and circuit complexity are achieved. The chapter further explores the customization of operating parameters in SoC-FPGA chips, allowing for fine-tuning of voltage levels specific to each component. This customization strategy optimizes power efficiency while preserving the desired performance levels. Furthermore, the integration of advanced SoC-FPGAs, such as RFSoC [10] or Versal ACAP [11] devices, in various applications of digital systems is investigated. These highly sophisticated devices serve as versatile accelerators for a broad range of DSP tasks, providing high-performance capabilities, re-programmability, and low-power consumption. Key considerations including high throughput interfaces, HW/SW co-processing capabilities, and the strategic deployment of accelerators are discussed, offering insights into performance evaluation and resource utilization.

With these objectives in mind, we propose a multi-level design approach for DSP applications, primarily focusing on SoC-FPGA platforms, to maximize their efficiency. Figure 2.1 illustrates our proposed design approach, comprising three main steps that synergistically complement each other and are briefly described below:

- The first step (① in Figure 2.1), referred to as the *Conventional Design Process*, involves algorithmic development for the target application (e.g., HW/SW partitioning, functions implementation), exploring combinations of functional blocks, i.e. benchmarking and comparison, to meet application constraints, as well as arithmetic optimizations (e.g., floating-point vs fixed-point). The result of this step is the low-level architecture of the hardware and/or software to be deployed on the SoC-FPGA device.
- The next step (② in Figure 2.1), referred to as *Circuit Optimization*, concentrates on the hardware architecture from the previous step and seeks to attain further optimizations on the arithmetic circuits by employing approximation techniques. This leads to an even more optimized hardware architecture, that aims to reduce the power/energy consumption and resource utilization, as well as increase the throughput.
- The final step (③ in Figure 2.1), referred to as *Voltage Scaling*, is an online procedure, which means it is applied after deploying the HW/SW architecture to the target SoC-FPGA device. The objective is to regulate the supply voltages of the

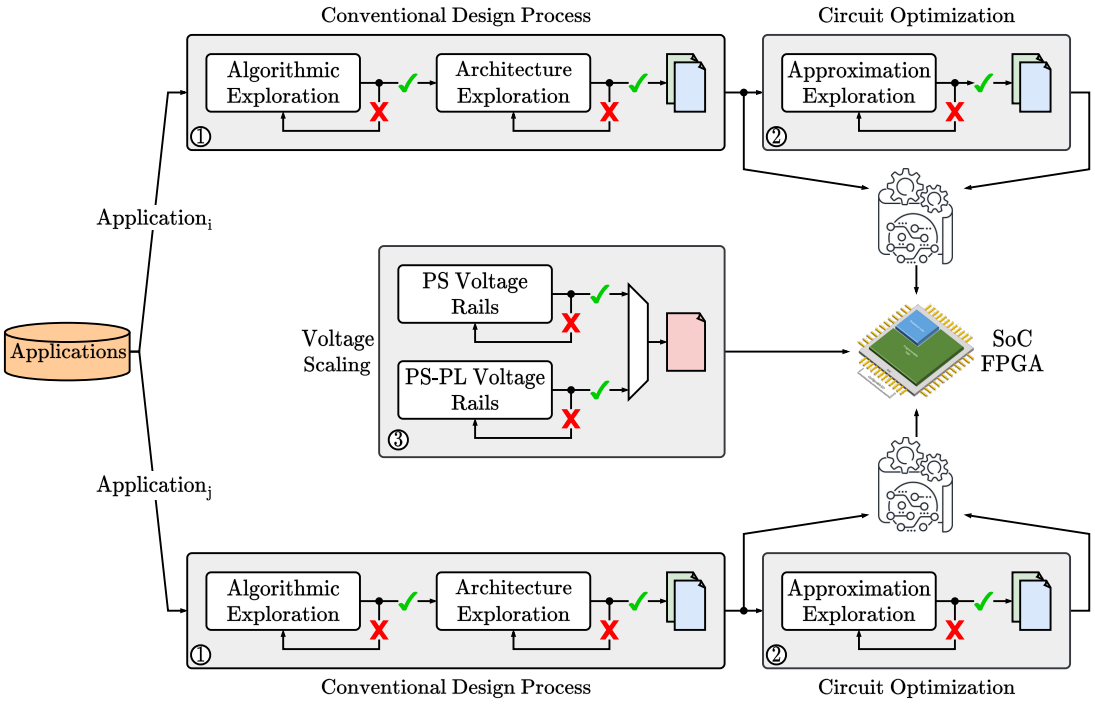


Figure 2.1: Proposed design approach for efficient and flexible SoC-FPGA platforms.

device and adjust them to the minimum allowable level, while ensuring the system operates normally.

Moving towards to a more advanced design approach, we also propose employing any combination of steps ①, ②, ③ to multiple applications that can coexist on the same SoC-FPGA device. In Table 2.1 we present various indicative scenarios of the proposed design approach, with some of them being detailed in the upcoming sections.

The rest of this chapter is structured as follows: In Section 2.2, we demonstrate the utilization of steps ① and ② (Scenario 1 in Table 2.1) on an M-QAM demodulator, a crucial DSP block of the receiver on a Baseband Processing Unit (BBU). We explore different algorithmic implementations, we design accurate floating-point and fixed-point hardware architectures based on the selected algorithms, and then apply approximation techniques. The implementations are compared we provide a detailed analysis of the results. In Section 2.3, we assess the application of steps ① and ③ (Scenario 2 in Table 2.1) on a computationally intensive image processing kernel, specifically the Harris Corner Detector. We present the kernel architecture, as well as its integration into a

Table 2.1: Indicative examples of various system deployments on a SoC-FPGA.

Deployment	Application 1		Application 2		③	Notes
Scenario	①	②	①	②		
1	✓	✓				Section 2.2
2	✓				✓	Section 2.3
3	✓		✓			Section 2.4
4	✓	✓	✓			-
5	✓	✓	✓	✓		-
6	✓	✓	✓	✓	✓	-

complete HW/SW image processing system in a low-budget Zynq-7000 SoC-FPGA. We show step by step the exploration performed to apply voltage scaling to the SoC-FPGA and we present the gains in power consumption achieved, while maintaining the nominal operating frequency of the kernel in the FPGA, and discus the results. In Section 2.4, we focus solely on step ① and apply it to two distinct yet equally computationally intensive application domains: AI/ML processing at the edge and Low-Level Physical processing (L1 Layer) of telecom infrastructure (Scenario 3 in Table 2.1). We provide a brief overview of each application and present evaluation results, including performance metrics and resource utilization. We consider the deployment of both applications on the same SoC-FPGA device located at the network edge, taking into account various scenarios based on different requirements. We note, that for each of these sections, we provide a brief introduction that outlines the motivations behind conducting these explorations. Finally, Section 2.5 summarizes the key findings of each Section and concludes this chapter.

2.2 Exploring the Design Space of DSP Circuits: Algorithms, Arithmetic, and Approximation Techniques

2.2.1 Introduction

Advanced technologies are utilized on every generation of mobile communication to provide improved network performance and consistent connectivity. Moreover, new and/or emerging applications, e.g., video streaming and augmented reality, promise to take advantage of these technologies to offer better and more consistent services to end-users. However, the rapid development of these applications introduces strict bandwidth constraints to maintain robust Quality of Service (QoS). In this context, the fifth generation (5G) technology standard promises high bandwidth and ultra-low latency for data transmission [31]. To tackle these strict constraints imposed to networks infrastructures, efficient solutions must be provided. One part of the network that handles large volumes of data is the Baseband Unit (BBU), which is responsible for modulating/demodulating the transmitting/receiving data and in order to meet these demands, processors such as Application-Specific Integrated Circuits (ASICs) and Field-Programmable Gate Arrays (FPGAs) [2,32,33] are considered for implementing components of the 5G infrastructure. On one hand, when the objective is to offer high computational efficiency then ASICs are preferred, on the other hand FPGAs are able to provide attractive throughput/power ratio in addition to their adaptability to new functionalities. In either case, for the research community, providing optimal circuits for the main computationally intensive tasks of a digital communication system is of high importance.

The work presented in this Section focuses on a Baseband unit implementation, and specifically, introduces efficient QAM demodulation architectures. Our aim is to expand the design space by exploring solutions, generic enough, for diverse scenarios (e.g., varying conditions in the channel) by examining various QAM demodulation and approximation techniques. Thus, we employ soft- and hard-decision algorithms relying on the calculation of the Log Likelihood Ratio (LLR) and the Maximum Likelihood (ML), respectively, and apply approximation techniques on both floating-point and fixed-point arithmetic. To achieve our goals we employ VHDL coding to implement four demodulation algorithms and then apply circuit approximation techniques. The circuits are customized and validated on an FPGA device, but we refrain from utilizing dedicated high performance FPGA resources (i.e., DSP blocks, RAM blocks), so as to be able to target also an ASIC implementation if needed. We focus on the most demanding part of their computation, i.e., the floating-point addition and the multiplication, and decrease the circuit complexity of the respective arithmetic units. More specifically, we develop approximate floating-point adders, generic hardware-friendly multipliers based on approximate

high-radix encoding and bit truncation. To verify our approximate QAM demodulation circuits, we also modeled them in MATLAB to examine the effects of approximations and tune the circuits. To summarize, the QAM demodulators are designed to perform their computations in parallel and in a pipelined fashion and we also carry out extensive Design Space Exploration (DSE) in terms of algorithm, arithmetic, and approximation levels, to determine the most suitable configuration.

2.2.2 Related Work

Telecommunication Functions on FPGAs

Communication standards/protocols change with fast pace, and as they evolve, new requirements arise that need to be integrated in the network infrastructure. One of the key components of a network infrastructure is the Baseband Processing Unit (BBU). The functionality of a BBU is to implement the low-level DSP function required for transmission at the physical layer (L1).

The role of the BBU is often undertaken by an FPGA device, due to its adaptability to new functionalities and/or features. There are a lot of works in literature that present OFDM-based telecommunication functions implemented on FPGAs. In [34] an FPGA-based OFDM transceiver that can support multiple IEEE 802.xx standards is presented. Dynamic partial-reconfiguration is used in [35] to adapt the processing datapath of a non-continuous OFDM FPGA-based baseband architecture, while based on the same principle in [36] an adaptable FFT core is given. [37] presents an SFO compensation method for OFDM. Finally, [38] presents the experimental results of transmitting/receiving an OFDM 5G NR signal using a real-time FPGA processing pipeline. All of these works consider a complete OFDM processing pipeline and/or target processing blocks that are more computational demanding (e.g., FFT/IFFT, SFO), while not considering any optimizations other than computation scheduling, parallel hardware architecture design and/or pipeline.

Circuit Approximation Techniques

In the field of approximate circuit design, approximation methods are applied at component level, e.g., adders and multipliers [16, 17, 39–46], as well as to bigger accelerators [47–50]. Most of these works focus on logic simplification techniques, namely, aim

at reducing the circuit complexity of the designs by pruning circuit nodes or using inexact building blocks. In addition to logic simplification, other state-of-the-art approaches involve voltage over-scaling [51] and over-clocking [52].

Regarding approximate adders and multipliers, which are also the main focus of our work, the EvoApprox8b library [41] provides numerous inexact designs of different accuracy and hardware efficiency. It is also worth mentioning that the integer arithmetic circuits [17,39,40,42,43,45,46] have received more research attention than their floating-point counterparts [16,44]. Another significant feature of several state-of-the-art works is the runtime accuracy/approximation configurability. In this context, the designs of [43] and [44] drive the operand bits to AND gates to tune the approximation strength by providing either the actual bit or '0' to the input of the multiplier. To provide dynamic configurability to adders, the designs of [45] and [46] share the addition into several sub-adders.

Regarding approximate accelerators, the main target is the implementation of kernels from the Digital Signal Processing (DSP) and Artificial Intelligence (AI) domains. In [47], a methodology for delivering approximate DSP and AI accelerators is proposed, involving design space exploration on algorithms, arithmetic, and approximate components. The authors of [48] propose an approximation framework that integrates approximate multipliers in deep neural networks. Similarly, in [50], the authors develop convolutional neural network architectures with various approximate fixed- and floating-point arithmetic.

2.2.3 Design of Approximate QAM Demodulation Circuits

To accommodate design space exploration, we design parallel M-QAM demodulation architectures in a modular approach. In this way, we are able to adopt alternative arithmetic and/or replace the arithmetic units with approximate ones.

We consider M-QAM signals (amplitude+phase) and their corresponding gray-coded constellation map, where each constellation point is represented by a vector b of $N = \log_2 M$ bits. Assuming that s is the transmitted symbol, each symbol is expressed as $z = s \cdot G_{ch} + w_0$, where G_{ch} is the channel frequency response and w_0 is the Additive White Gaussian Noise (AWGN) of variance σ_0^2. By performing zero-forcing frequency equalization and phase correction, the received symbol is equal to $r = s + w$, where w is the AWGN of variance $\sigma^2 = \sigma_0^2/|G_{ch}|^2$.

Subsequently, we present the QAM demodulation algorithms, their high-level block ar-

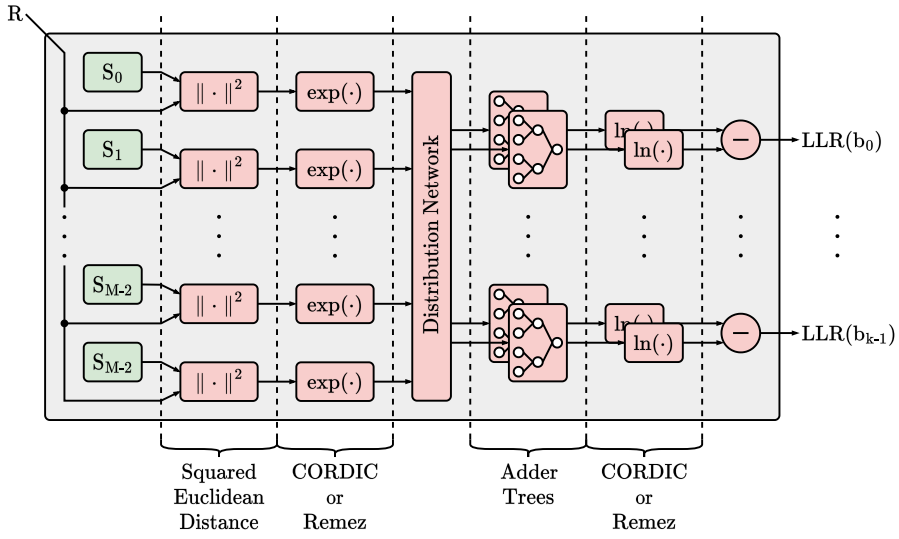


Figure 2.2: M-QAM demodulation architecture of the exact log likelihood ratio (ELLR) algorithm.

chitecture, and implementation details regarding their basic components and the applied approximations.

Exact LLR Architecture

The Exact LLR (ELLR) method predicts each bit of the received symbol based on probabilities. Assuming equal probabilities $P(b_i = 0) = P(b_i = 1) = 1/2$, the LLR for the i-th bit of the symbol is defined as [12]:

$$LLR(b_i) = \ln\left[\frac{\sum_{s:b_i=0} e^{-\frac{1}{\sigma^2}((r_x-s_x)^2+(r_y-s_y)^2)}}{\sum_{s:b_i=1} e^{-\frac{1}{\sigma^2}((r_x-s_x)^2+(r_y-s_y)^2)}}\right] \tag{2.1}$$

where (r_x, r_y) are the in-phase & quadrature coordinates of the received symbol r, and (s_x, s_y) are the coordinates of the constellation points s with 0 and 1 in the i-th bit.

In Figure 2.2, we present our ELLR architecture. As shown, all the LLRs are calculated in parallel, with the modules retaining the input throughput of 1 data per clock cycle (CC). Our distribution logic module forwards the square distances for the 0 and

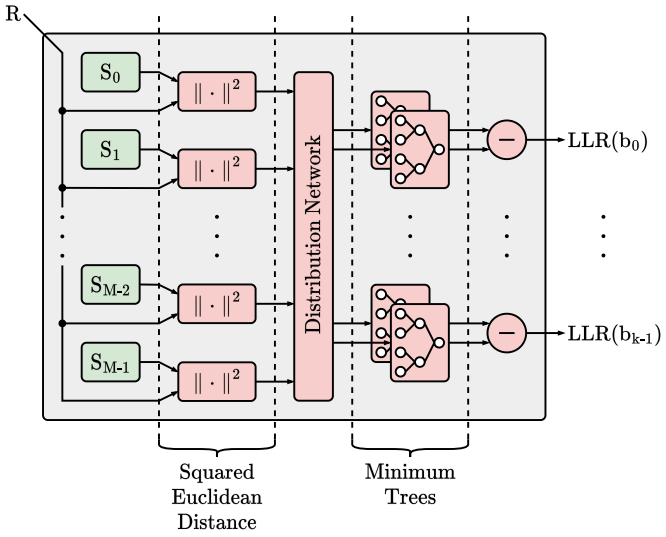


Figure 2.3: M-QAM demodulation architecture of the approximate log likelihood ratio (ALLR) algorithm.

1 constellation points in the corresponding adder tree, to perform the accumulations of Equation (2.1). The ELLR method requires the implementation of exponential and natural logarithmic functions, thus, we adopted two different approaches in VHDL, i.e., the hyperbolic Coordinate Rotation Digital Computer (CORDIC) function and a polynomial approximate function based on the Remez algorithm [53]. The CORDIC is designed in a generic way and is able to function with a configurable bit-width at compile time by the user in order to explore different resources–accuracy trade-offs. As for the polynomial function, we examine various polynomial degrees and evaluate their impact on resources and accuracy. Both functions are designed to be synchronous and fully pipelined, and impose only a small latency, e.g., for 64-QAM, the latency is 43 CCs and 14 CCs for the CORDIC and polynomial functions, respectively.

Approximate LLR Architecture

To avoid the exponent and logarithmic calculations as well as the additions [12], Eq. (2.1) can be simplified by using only the nearest constellation point with $b_i = 0$ and the nearest constellation point with $b_i = 1$. The final result is obtained by subtracting the minimum distances. This algorithmic approximation is called Approximate LLR (ALLR) and is

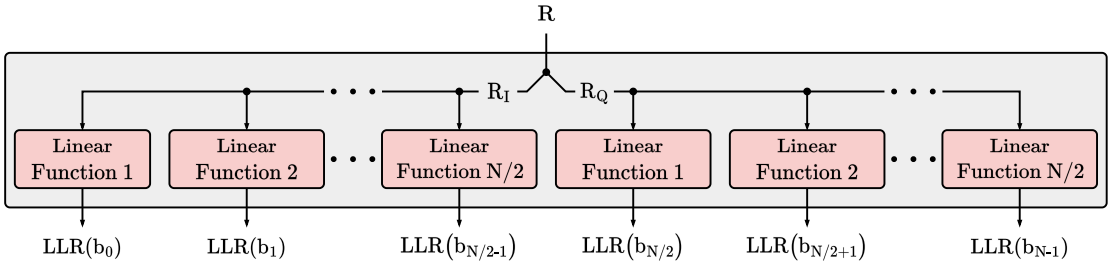


Figure 2.4: M-QAM demodulation architecture of the piecewise log likelihood ratio (PLLR) algorithm.

given by (2.2)

$$LLR(b_i) = -\frac{1}{\sigma^2}\left(\min_{s:b_i=0}\left((r_x-s_x)^2+(r_y-s_y)^2\right)-\min_{s:b_i=1}\left((r_x-s_x)^2+(r_y-s_y)^2\right)\right) \quad (2.2)$$

Figure 2.3 presents the ALLR architecture. The ALLR architecture is fully pipelined and operates on a continuous stream of data without stalling the processing. The minimum of the $M/2$ distances is computed by a fully pipelined comparator tree that is able to perform the comparison for every two inputs in $\log_2 M/2$ steps. The minimum value is stored in a register, to be used by the output subtractor. For 64-QAM the latency is 4 CCs, while for the 256-QAM 2 additional CCs are required. The extra CCs in the 256-QAM are added due to the need of a bigger tree to calculate the minimum distances.

Piecewise LLR Architecture

The Piecewise LLR (PLLR) algorithm approximates Equation (2.1) by using a linear function [14], This method considers $k = 1, 2, \ldots N$, as well as $d_{x,k}$ and $d_{y,k}$, which denote the half distance between the partitions boundaries of $b_{x,k}$ and $b_{y,k}$, respectively. Given that r is the received symbol with coordinates (r_x, r_y), the LLR is calculated as:

$$LLR(b_{x/y,k}) = |G_{ch}|^2 \cdot D_{x/y,k}, \quad \text{where} \quad D_{x/y,k} = \begin{cases} r_{x/y} & k = 1 \\ -|D_{x/y,k-1}| + d_{x/y,k} & k > 1 \end{cases} \quad (2.3)$$

Our PLLR architecture is presented in Figure 2.4, where N linear functions are calculated

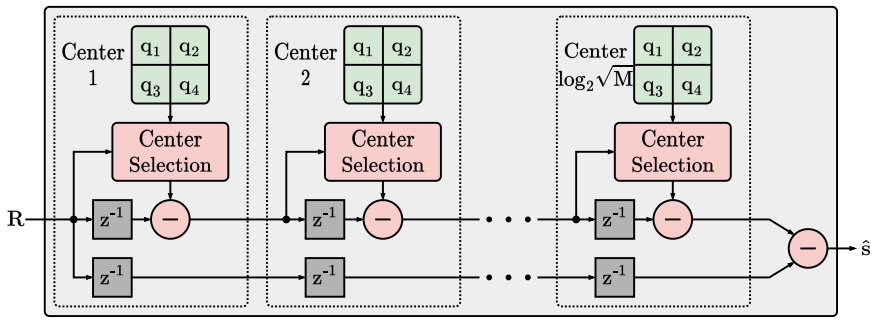


Figure 2.5: M-QAM demodulation architecture of approximate maximum likelihood (AML).

in parallel. The computation of $D_{x/y,k}$ involves only additions, followed by a multiplication with $|G_{ch}|^2$, and thus, the circuit complexity is significantly decreased compared to the previous demodulation methods. Similarly to ALLR, our PLLR design is fully pipelined, and performs the computations of the linear functions in streaming mode. The latency is 2 and 3 CCs for the 64-QAM and 256-QAM, respectively (256-QAM requires extra additions).

Approximate ML Architecture

Typical ML methods use non-linear functions to calculate the demodulation symbol. To decrease the conventional complexity, the Approximate ML (AML) method of [15] uses a closed function of the estimated channel and received signal to estimate the QAM symbol:

$$\hat{s} = \sum_{p=1}^{\log_2 \sqrt{M}} c_p, \quad \text{where} \quad c_p = \sqrt{\frac{3M}{M-1}} 2^{-p} e^{jg(r - \sum_{m=1}^{p-1} c_m)} \tag{2.4}$$

The proposed AML architecture is presented in Figure 2.5. Our design pre-calculates and stores in ROMs the $\log_2 \sqrt{M}$ centers c_p. Initially, the first center is identified by the quarter in which the input is located. Subsequently, each center is identified by the quarter that is derived when all the previous centers are subtracted from the input. With this approach, we avoid storing all the intermediate centers to accumulate them as indicated by Eq. (2.4). In contrast, the accumulation is performed as: $\hat{s} = r - (r - c_1 - c_2 - \cdots - c_{\log_2 \sqrt{M}}) = c_1 + c_2 + \cdots + c_{\log_2 \sqrt{M}}$. The AML architecture has a $\log_2 \sqrt{M}+1$ CCs latency, i.e., 4 and 5 CCs for 64-QAM and 256-QAM, respectively.

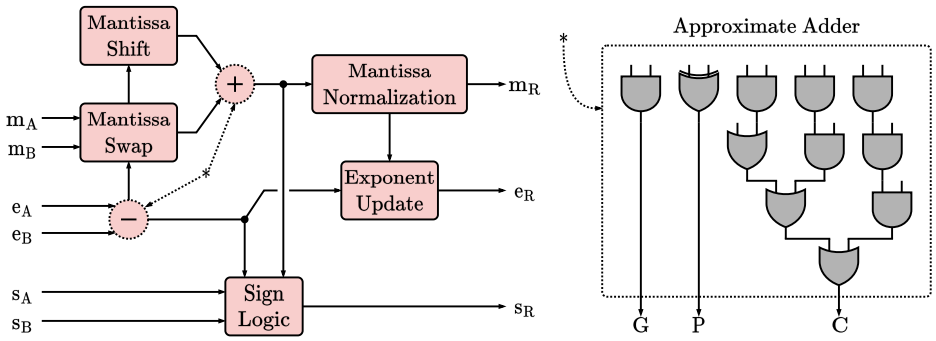


Figure 2.6: Approximate circuit of floating-point (FLP) adder.

Arithmetic Approximation Techniques

We adopt both the floating-point (FLP) and fixed-point (FXP) arithmetic formats in our QAM demodulation circuits, and perform approximate arithmetic operations. Next, we describe the approximate arithmetic units used in our QAM designs.

Floating-Point Approximations: Both the mantissa adder and exponent subtractor are replaced with an approximate carry look-ahead adder [16] that decreases the circuit complexity of the floating-point (FLP) addition. By splitting the calculations in two segments and omitting the least-significant one, an approximation is inserted in the generation of the carry output of the i-th stage. As a result, the calculation is performed approximately in a window of size W.

$$c_{(i+1)} = \sum_{j=i-W+1}^{i} G_j \Big(\prod_{m=j+1}^{i-1} Pm \Big) \tag{2.5}$$

where $P_i = A_i \oplus B_i$ and $G_i = A_i \cdot B_i$ are the propagate and generate signals of the i-th stage, respectively.

Figure 2.6 presents the high-level architecture of the approximate FLP adder, where the inexact mantissa addition and exponent subtraction are highlighted. The rest of the components remain the same as in the accurate FLP addition, except that we do not apply any rounding technique in the mantissa sum. In the right hand-side (Figure 2.6), we present the basic logic block of the approximate carry look-ahead adder, which approximately calculates the carries according to Equation (2.5), and stores them in DFFs. The total latency of the approximate FLP adder is 5 CCs.

Fixed-Point Approximations: In this work, we target multiplication architectures based

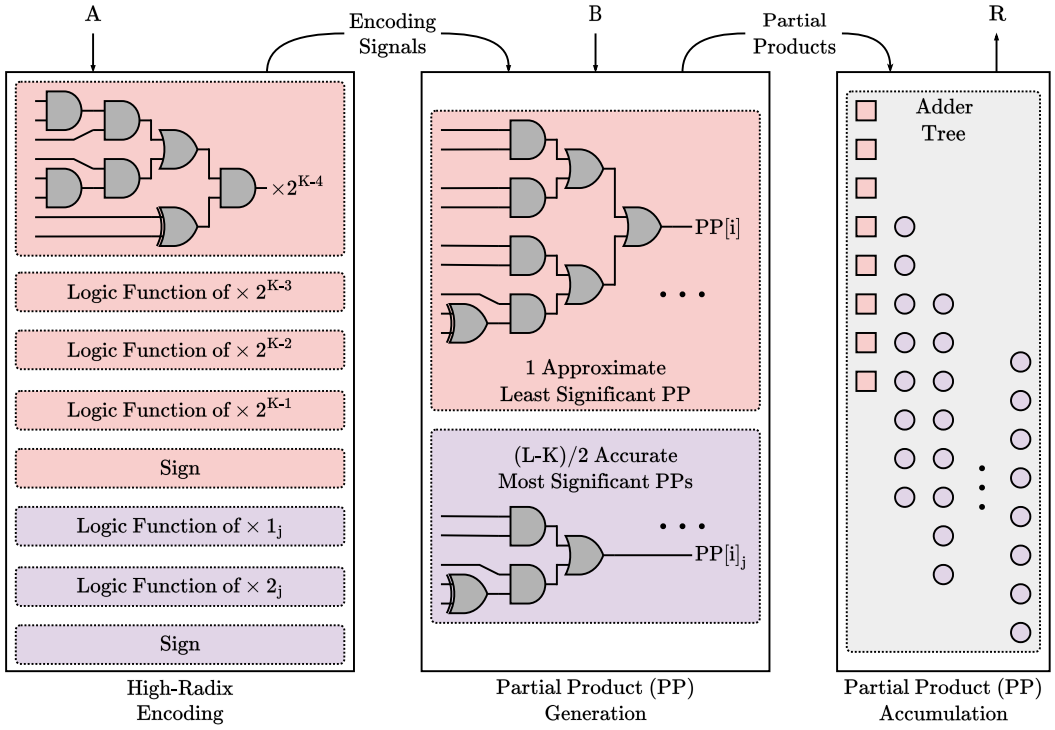


Figure 2.7: Approximate fixed-point (FXP) multiplier.

on radix encodings, due to their efficiency compared to other algorithms [54]. To decrease the circuit complexity of the fixed-point (FXP) multiplication, we perform inexact partial product (PP) generation based on the hybrid high-radix encoding [17]. This technique tunes the approximation levels with the configuration parameter $K = 2 \cdot l \geq 4$. In particular, the $L - K$ MSBs of one of the L-bit input operands (e.g., A) are accurately encoded with radix-4, while its K LSBs are approximately encoded with radix-2^K. The approximation is inserted by mapping the high-radix encoded values to the nearest of the four largest powers of 2 or 0 [17]:

$$A = \langle a_{L-1} \dots a_1 a_0 \rangle \quad \rightarrow \quad A = A_{MSB} + A_{LSB} = \sum_{\substack{j=K/2 \\ K \geq 4}}^{L/2-1} y_j^{R4} 4^j + y_0^{R2^K} \tag{2.6}$$

where

$$y_j^{R4} = -2a_{2j+1} + a_{2j} + a_{2j-1} \quad \Longrightarrow \quad y_j^{R4} \in \{0, \pm 1, \pm 2\} \tag{2.7}$$

$$y_0^{R2^K} = -2^{K-1} a_{K-1} + 2^{K-2} a_{K-2} + \dots + a_0 \implies y_0^{R2^K} \in \{0, \pm 1, \dots, \pm(2^{K-1} - 1), -2^{K-1}\} \tag{2.8}$$

To decrease the circuit complexity of the $y_0^{R2^K}$ encoding, which is due to values that are not a power of 2, all these values along with the K–4 smallest powers of 2 are mapped to the nearest of the four largest powers of 2 or 0, as shown in the approximate encoding table of [17]. As a result, the approximate version of the high-radix encoding uses $\hat{y}_0^{R2^K} \in \{0, \pm 2^{K-4}, \pm 2^{K-3}, \pm 2^{K-2}, \pm 2^{K-1}\}$ instead of $y_0^{R2^K}$, and the encoding circuit is similar to the classical y_j^{R4} encoder.

The approximate multiplier (Figure 2.7) generates (L–K)/2 accurate PPs based on the radix-4 encoding, i.e., y_j^{R4}, and 1 approximate PP based on the radix-2^K encoding, i.e., $\hat{y}_0^{R2^K}$, which practically substitutes the $K/2$ least-significant PPs of the accurate radix-4 multiplier. We implement such multipliers for various K values, following the architecture illustrated in Figure 2.7. The first stage is the approximate hybrid high-radix encoding, which involves circuits such as the one implementing $\times 2^{K-4}$, i.e., one of the five 1-bit signals that approximately encode the LSB segment of A. Similar small circuits are implemented for the accurate classical encoding of the MSB segment. Next, we forward all the encoding signals along with B in the PP generation module, and finally, we accumulate the PPs in an accurate adder tree. We note that the units for both encodings and i-th bit PP generators are fixed and independent of K. The configuration parameter affects only the bit-width of the approximate PP and the number of

accurate PPs, and thus, the number of 1-bit approximate PP generators and accurate encoders & PP generators, respectively. The total latency of our multiplication circuit is 5 CCs.

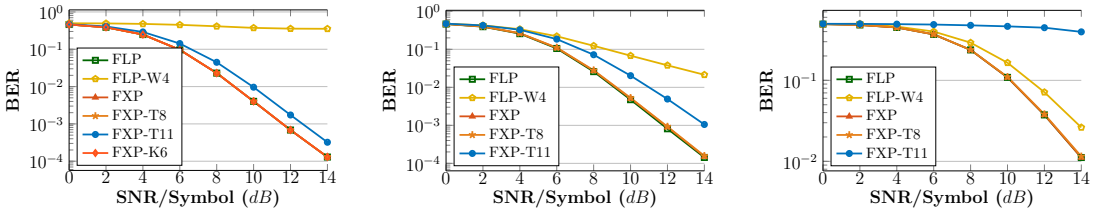
2.2.4 Experimental Evaluation

Experimental Setup

In this section, we present the experimental results for our QAM demodulation architectures in terms of accuracy, hardware resources and power consumption. We examine various arithmetic and approximation configurations, and we discuss the most interesting results from our design space exploration. For the evaluation, we assume a transmitter producing random bits and a convolutional encoder of 1/2 code rate with generator polynomial $(133, 171)$ and constraint length of 7. The QAM modulator considers normalized constellation diagram and the modulated signal is transmitted through an AWGN channel of variance $\sigma^2 = 10^{-(\mathrm{SNR}_{dB}/10)}$. Finally, the outputs of our LLR-based demodulation architectures are processed by a Viterbi decoder in unquantized mode.

To evaluate the hardware efficiency of the proposed approximate designs, we implement our QAM architectures on the Xilinx Zynq Ultrascale+ ZCU106 (XCZU7EV) FPGA, using parametric VHDL and design space exploration. Considering that the proposed architectures can be deployed on various technologies, e.g., ASICs, flash FPGAs, we assess the circuits' complexity and throughput without employing elaborate FPGA primitives such as the DSPs. Hence, in a prototyping fashion, we force the Xilinx Vivado 2019.2 tool to utilize only LUTs and DFFs. Moreover, for the designs without approximate arithmetic units, i.e., the accurate FLP and FXP designs as well as the FXP designs with bit truncation, we force the tool to employ the build-in libraries to map the arithmetic operations on the LUTs. The data bit-width of the FLP designs is 32 bits, while the bit-width of the accurate FXP design is 16 bits (with 14 fractional bits). Moreover, we present the power consumption, as reported by Vivado Design Suite, for the maximum achievable frequency. To assess the accuracy of our approximate demodulation designs, we use two metrics: (i) BER, which is equal to the number of bit errors divided by the total number of transmitted bits, and (ii) Mean Relative Error (MRE), i.e., the mean error of LLRs and symbol distances for the LLR and ML techniques, respectively (vs the "accurate" floating-point model).

We note that we do not present results for the ELLR, as it provides similar accuracy



(a) Approximate LLR (ALLR) **(b)** Piecewise LLR (PLLR) **(c)** Approximate ML (AML)

Figure 2.8: BER scaling in our 64-QAM circuits. FLP = accurate floating-point, FLP-W4 = floating-point with approximate addition [16], FXP = accurate fixed-point, FXP-TX = fixed-point with bit truncation, FXP-K6 = fixed-point with approximate multiplication [17].

with ALLR, at the expense of increased resource utilization. For example, for 64-QAM, the BER scaling of ELLR almost matches the scaling of ALLR (for small SNR, it is the same). In terms of hardware, when considering accurate FXP arithmetic, the CORDIC-based implementation utilizes 2.2× LUTs and 3.8× DFFs, while the polynomial-based implementation utilizes 2.6× LUTs and 8× DFFs. The resource utilization is increased for higher-order QAMs, e.g., for 256-QAM, the ELLR does not fit in such a big FPGA device, even when using approximation techniques.

Exploration Results

Figure 2.8 illustrates the BER scaling of our 64-QAM architectures for different SNR values per QAM symbol transmitted. The ALLR algorithm (Figure 2.8a) with fixed-point arithmetic in conjunction with approximations provides similar BER performance to the reference FLP implementation. Specifically, the accurate FXP and approximate FXP-T8 & FXP-K6 designs show maximum 1% relative error in BER values compared to FLP. On the other hand, FXP-T11 shows increased relative errors ($1\% - 150\%$) as SNR increases ($0 - 14$dB), however, these BER values are still small. As for the FLP-W4 implementation, its performance is the worst for all SNR values, thus, it is non-viable for deployment in real-world scenarios. The results suggest that when using relative small FXP approximations the computation error that propagates through the circuit is mitigated and the final LLR values are closer to the reference accurate FLP. However, this is not the case for the approximate FLP-W4 and FXP-T11 circuits.

PLLR (Figure 2.8b) presents the same BER performance as ALLR, with BER values ranging from 10^{-1} to 10^{-4}. The accurate FXP and approximate FXP-T8 implementations fol-

Table 2.2: Resources of our 64-QAM circuits on Xilinx Zynq Ultrascale+ XCZU7EV[1] (ZCU106)

	Approximate LLR (ALLR)						**Piecewise LLR (PLLR)**					**Approximate ML (AML)**				
	FLP	FLP-W4	FXP	FXP-T8	FXP-T11	FXP-K6	FLP	FLP-W4	FXP	FXP-T8	FXP-T11	FLP	FLP-W4	FXP	FXP-T8	FXP-T11
LUT	106097	104881	56278	19957	10210	37653	2060	2020	160	76	59	3848	3504	139	91	44
DFF	42432	42432	21596	10684	9316	51572	886	886	215	126	96	1204	1204	44	28	18
MHz[2]	286	294	312	323	416	321	333	435	571	645	740	385	416	588	645	769
W[3]	5.0	4.9	5.5	3.0	2.5	3.6	0.8	0.8	1.1	1.0	0.9	0.6	0.7	1.0	0.7	0.6
MSps/W	57.2	60.0	56.7	107.7	166.4	89.2	416.3	543.8	519.1	645.0	822.2	641.7	594.3	588.0	921.4	1281.7
avg. MRE	–	86.00	2.22	20.59	193.14	1.74	–	177.05	1.28	23.35	165.55	–	8.36	0.02	4.65	22.98

[1] Resources: 230K LUTs, 461K DFFs (DSPs are ignored).
[2] It is the maximum clock frequency. The throughput of all the circuits is 1 sample per CC, thus, MHz = Msamples/s.
[3] It is the power consumption measured at the maximum clock frequency of the circuits.

low closely the BER curve of the reference accurate FLP implementation, and show ∼0.5% to ∼14% relative error in BER values as SNR increases. The performance of FXP-T11 declines significantly compared to the ALLR counterpart, and exhibits ∼2% to ∼630% relative error in BER values for increased SNR compared to the FLP implementation. For the FLP-W4 implementation of PLLR, we note that it performs much better than the FLP-W4 of ALLR, but still not quite well to be considered for actual deployment on a system. Again, taking into account the approximation used and the operation performed, the approximate FXP circuits seem to be more robust.

The AML algorithm (Figure 2.8c) is orders of magnitude worse than the ALLR and PLLR algorithms for the same input dataset and SNR range (BER of 10^{-1} to 10^{-2}), regardless of the arithmetic and approximation used. Considering only the specific algorithm and its implementations, the FXP and FXP-T8 designs demonstrate the same BER performance as the reference FLP implementation with relative error in BER values near 0% for FXP and no more than 3% for FXP-T8 for all SNR values. On the other hand, FXP-T11 is inefficient, thus, aggressive approximations based on truncation seem unreasonable for AML. As for the FLP-W4 variant, it follows the same trend as the reference FLP, but presents 0% to 135% relative error in BER values as the SNR increases. In this algorithm, any significant error that appears in one stage will lead the comparison performed to select wrong reference centers, and as a result, larger error will be introduced in the next stage, leading to an increased accumulated error and thus very inefficient performance.

In Table 2.2, we present the implementation (place & route) results of our 64-QAM circuits on Xilinx Zynq Ultrascale+ XCZU7EV hosted on ZCU106 evaluation board. The table also reports the average MRE of LLR values for SNR $0 - 14$dB. As shown, ALLR is the more demanding one in terms of resource utilization for all arithmetic and approximations used. Comparing only the reference FLP implementations for each

algorithm, we observe a significant difference in LUT/DFF utilization. Specifically, PLLR and AML have $\sim$ 98% less resource utilization than ALLR, while also achieving higher operating frequency of 16% and 35%, respectively. When also considering the BER scaling, PLLR is the most efficient demodulator for floating-point arithmetic. Next, we examine the gains in resource utilization and operating frequency for each employed demodulation algorithm:

- *ALLR circuits*: The FLP-W4 implementation provides a negligible reduction of only 1% in LUTs and only 3% increase in frequency compared to FLP. On the other hand, the fixed-point ALLR circuits deliver increased gains. The accurate FXP circuit gains 50% in resources and has 9% higher frequency, while FXP-T8 and FXP-T11 have 81% and 90% reduction in LUTS, 75% and 78% reduction in DFFs, and 13% and 45% increase in frequency, respectively. FXP-K6 reduces its utilization by 64% in LUTs, 21% in DFFs and operates 12% faster. If we seek BER performance in-line with FLP, circuits with moderate truncation (FXP-T8) or approximate multipliers (FXP-K6) can be used. Also, considering the MRE metric, FXP-T8 is worse than FXP-K6 (20.59 vs 1.74).
- *PLLR circuits*: FLP-W4 offers only 2% reduction in LUTs, but gains 30% in frequency compared to the FLP circuit. Regarding fixed-point circuits, we observe 92%, 96%, 97% reduction in LUTs, 75%, 85%, 89% reduction in DFFs and 71%, 93%, 122% higher operating frequencies for the accurate FXP, approximate FXP-T8 and FXP-T11 circuits, respectively. As SNR increases (Figure 2.8b), the BER performance of FLP-W4 and FXP-T11 deviates significantly from FLP, thus, a fixed-point architecture with moderate truncation can be adopted.
- *AML circuits*: FLP-W4 achieves 9% LUT reduction and 8% frequency increase compared with the accurate FLP circuit. For the fixed-point circuits, we observe 96% LUT reduction, 98% DFF reduction either we have accurate circuits (FXP) or approximate ones (FXP-T8, FXP-T11). Regarding frequency, an increase by 53% for the accurate FXP, 68% for the FXP-T8 and almost 100% for the FXP-T11 is seen. However, we note that AML provides the worst BER performance among all algorithms and arithmetic, and thus, its deployment on a real-world system may be limited. Incorporating the BER results (Figure 2.8c), a fixed-point with moderate truncation (T8) circuit is preferable, as it follows the BER curve of FLP.

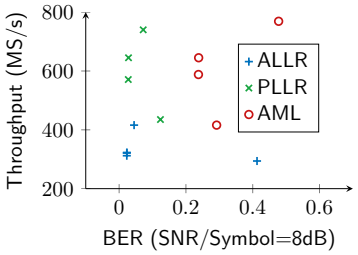
Considering the power consumption (Table 2.2) of the presented circuits, we observe that for each algorithm group, there is an increase in the power consumption of the accurate FXP implementation compared to the FLP implementations, either accurate or approximate. Moreover, the approximate FXP implementations consume less power than the accurate FXP, but in the same degree as the FLP ones. This behavior can be justified taking into account that the target platform for our evaluation is an FPGA device, which has static and dynamic power factors. For example, considering the PLLR

implementations, a significant reduction in resource utilization is observed moving from the accurate FLP to FXP-T11, thus the static power of the circuit, which relates to the number of used resources, is reduced. However, the operating frequency increases, which leads to higher switching activity of the utilized resources and thus, the dynamic power consumption of the circuits also increases and "compensates" for the lower static power consumed. Additionally, another factor that we must take into account is that the synthesis tool, after placement and routing may spread the utilized resources of a circuit across the FPGA fabric leading to increase usage of routing resources that also consume power. That power consumption (static & dynamic) is also included in the one reported by the tool.

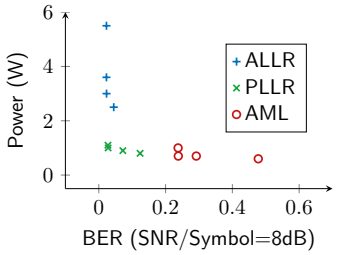
When considering higher-order QAMs, i.e., 256-QAM, as explained before, ELLR does not fit in the FPGA. Regarding ALLR, it is possible to fit only the FXP implementations, as the 256-QAM ALLR utilizes $\sim \times 4$ the resources of the 64-QAM demodulator. As a result, the LUT utilization reaches 97% of the total chip's resources for the accurate FXP implementation with datapath of 16 bits, however, when inserting approximations it is significantly decreased: the designs with bit truncation i.e., FXP-T8 and FXP-T11, utilize 38% and 21% of the LUT resources, while the use of the approximate multiplier (FXP-K6) reduces the utilization to 62%. Similar scaling is observed for the other two algorithms, which enable the implementation of even higher QAM orders, e.g., 1024, as they impose low computational complexity. Regarding the BER performance we have observed that the approximations introduced perform equally well for higher-order QAM modulation formats, e.g. 256-QAM. Finally, in real-world scenarios, the FPGA is used to implement additional computational-intensive functions of the baseband processing chain, thus, it is important not to preserve all the resources for the demodulation. In this direction, as shown, methodical approximations can provide significant resource gains at the expense of small accuracy loss. Also, when considering more platform-specific implementations, e.g., to exploit the hardwired DSP primitives of the FPGA, we can achieve additional resource savings as well as increased throughput.

Pareto Trade-Off Analysis: Hardware Resources vs. Accuracy

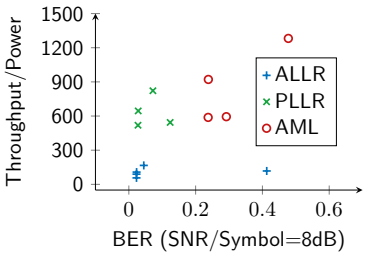
Figure 2.9 shows a comparison of all the circuits considering different metric combinations in a Pareto diagram. Specifically, in Figures 2.9a–2.9c and Figure 2.9g, we consider the BER performance versus throughput, power, throughput/power ratio and LUT utilization for a fixed SNR value (8dB). In these Figures, we do not observe much diversity regarding the circuits that form the Pareto fronts. Specifically, in all four diagrams the Pareto front consist of circuits from ALLR and PLLR algorithms. However, in Figure 2.9b and Figure 2.9g there are also circuits from the AML algorithm that form the Pareto



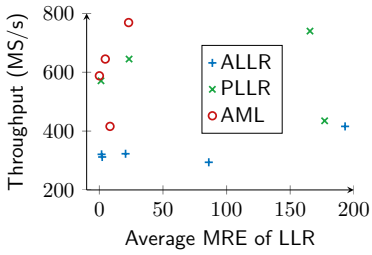
(a) BER–Throughput



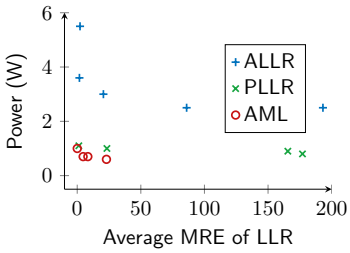
(b) BER–Power



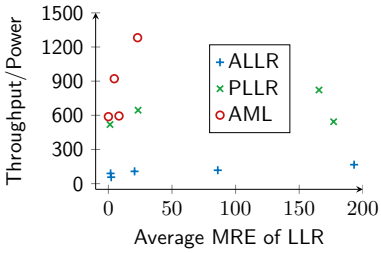
(c) BER–Throughput/Power



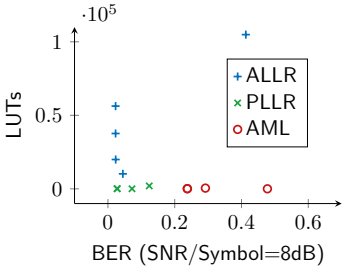
(d) MRE–Throughput



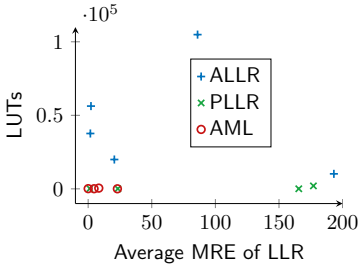
(e) MRE–Power



(f) MRE–Throughput/Power



(g) BER–LUT



(h) MRE–LUT

Figure 2.9: Pareto analysis of the proposed 64-QAM circuits in terms of accuracy and hardware resources.

front. However, considering the best trade-off of the examined metrics, the diagrams show that the optimal circuits belong to the PLLR algorithm. Next, in Figures 2.9d–2.9f and Figure 2.9h, we consider the average MRE instead of BER performance, while keeping the other metrics the same. In these test cases, the Pareto front consist of circuits from all algorithms, but the PLLR algorithm has less circuits. The most optimal circuits that give the best trade-off of the metrics used are from the AML algorithm, however, one should be cautious when considering the specific algorithm for deployment because it provides the worst BER performance compared to the others for the same SNR values.

Solely in terms of resources and throughput, our exploration indicates that, regardless of algorithm, fixed-point architectures are the most efficient and truncation is the preferable approximation technique. Taking into account the BER results, the ALLR and PLLR algorithms have better BER performance than AML, with ALLR been slightly better than PLLR.

2.3 Power Consumption Optimization in SoC-FPGA Chips

2.3.1 Introduction

The advancements in energy consumption consists a vital aspect of the embedded processing systems. The computational requirements at the edge are growing at an irreducible pace due to proliferation of more demanding services in the domains of automotive, space, Internet of Things (IoT), etc. The traditional trend in performance improvement based on transistor scaling has slowed down since the Dennard law breakdown, while the manufacturing of more advanced chips has become inefficient from an economic standpoint. As such, alternative strategies are followed to increase the efficiency of modern chips in terms of processing speed and power. One prominent strategy is the fabrication of System-on-Chip (SoC) devices, combining diverse processing units, e.g., CPUs, GPUs, DSPs, on the same chip. Such integration allows for better adaptation to the specifics of a given application and provides multi-fold benefits including lower power consumption, higher processing speed and increased reliability [7].

Among the various processing platforms, SoC FPGAs are considered a very promising solution for a wide range of applications, capable of proving tremendous performance per Watt figures. The industry has already acknowledged the benefits arising from the integration of FPGAs in their products, e.g., in the domains of space, automotive, datacenters, etc. [4, 55]. Even though FPGAs are considered more power demanding than

ASICs [56], they provide increased flexibility to potential algorithmic and system changes owing to their reconfiguration capabilities, thus improving time-to-market and reducing manufacturing cost.

The nominal performance characteristics of an FPGA chip are determined during the manufacturing stage. The vendors impose global and conservative operation guard-bands to huge sets of chips, with respect to extreme-case process corners scenarios referring to all process, voltage and temperature variations. Although this strategy ensures reliable operation even for the worst-case scenario of the production line, it is considered a pessimistic approach which does not adapt to the electrical characteristics of the individual chips, as well as to the specifics of the application running on the chip. As a result, significant performance is lost, either in terms of frequency or power. The determination of guard-bands becomes more complex in SoC FPGAs where multiple processing units with different characteristics co-operate on the same chip.

In the current Section, we focus on the exploration and customization of the guard-bands imposed in widely-used SoC-FPGAs as a means to provide more energy efficient solutions. Specifically, we develop an exploration methodology for the judiciously regulation of the voltage rails driving the different components of the chip including the programmable logic, the embedded RAM Blocks (RAMBs), the clock network, the I/O banks and the processing system hosting the CPU processors. The customization of guard-bands depends both on the actual silicon capabilities of the underlying FPGA, as well as to the design-specific operation, e.g., IR-drop. We conduct proof-of-concept experiments on a Zynq XC7Z020 SoC FPGA executing a real-world image processing algorithm, considering diverse HW/SW mapping and resource utilization scenarios.

2.3.2 Background

There are variety of methods targeting the reduction of power consumption in commercial FPGAs, ranging from low level, e.g., clock gating [57], to higher level, e.g., specific programming styles or power-aware implementation strategies provided by electronic design automation (EDA) tools [58]. Another more aggressive approach is based on the employment of voltage scaling techniques. Such approaches rely on the concept that available headroom exists for voltage customization, due to pessimistic guard-bands included in the solutions provided by the industrial EDA tools. Several works have been proposed towards this direction. In [59], the authors deploy additional shadow registers in place with the critical path registers latching the exact same input data. In this way, they measure the actual timing slack by sweeping the offset of the clock driving the shadow registers

until a mismatch is detected between the data of shadow and main registers. Based on the measured slacks, they build a discrepancy profile which is then used to perform dynamic voltage scaling and reduce the power. On the same principle, in [60] a library of soft-macro flip-flops augmented with detection logic is introduced to replace original critical flip-flops. Based on adaptive voltage scaling and timing violation detection they achieve significant power savings. In [61], the authors propose a multi-phase tool that initially generates a calibration bitstream to isolate the critical paths and identify their actual operation limits under various voltage and temperature conditions. The derived results are stored in a calibration table, which is then used in a second phase to perform dynamic voltage scaling and tune the FPGA operation when the original application bitstream is loaded on the FPGA. In [62], a custom HW/SW architecture is developed to automatically customize the supply voltage of the FPGA based on voltage scaling and verification of results with golden data.

The aforementioned works, however, target only the reduction of the power attributed to programmable logic. In our work, we propose an exploration methodology to comprehensively customize all the different voltage rails driving the various components of the SoC-FPGA, thus enabling further power gains at system level.

2.3.3 System Description

This subsection describes the system developed for the purposes of our exploration. We begin with the description of the employed SoC-FPGA and the selected power domains under examination. We continue with the description of the test-case application and its different implementation scenarios on the FPGA. Lastly, we present the integration of the final system.

Target Device, DSP Accelerator and Voltage Rails

We employ a Xilinx Zynq XC7Z020 SoC FPGA. This particular FPGA consists of two main sub-systems, the Processing System (PS) and the Programmable Logic (PL). The PS hosts a dual-core ARM Cortex-A9 processor as its main processing unit along with numerous peripheral interfaces, such as serial, Ethernet, I2C, DDR memory controller, etc. The PL refers to the conventional FPGA resources, including general purpose logic (CLBs), arithmetic (DSP) and memory (RAMB) blocks distributed across the fabric in a repeatable way. The PS and PL communicate with each other based on the AXI communication protocol [7]. Both in PS and PL, there are components operating at

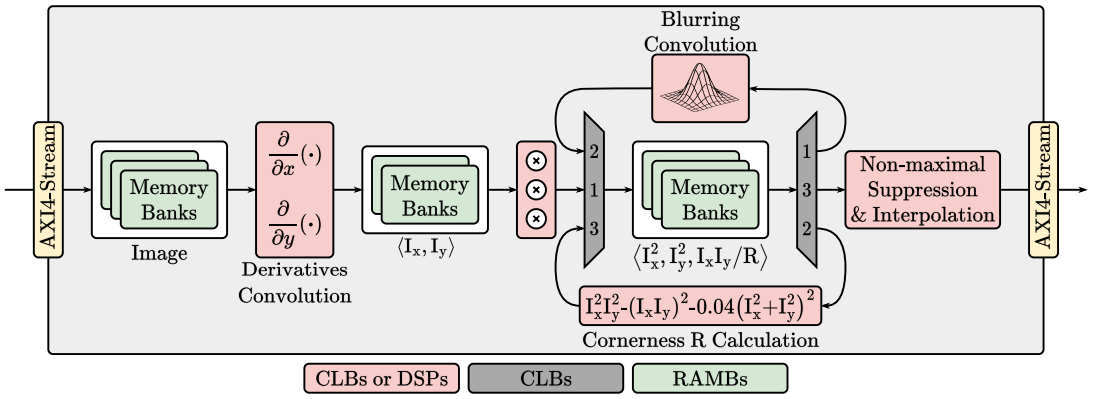


Figure 2.10: Hardware architecture of Harris IP core.

different power domains with varying supply voltage levels, thus, we specifically focus on the customization of the followings voltage rails, each one supplying a specific power domain:

- V_{CCPINT}: powers the internal logic circuits of PS
- V_{CCPAUX}: powers auxiliary circuits of PS
- V_{CCMIO}: powers the PS I/O banks and the reference clock of Zynq PS
- V_{CC1V5}: powers the PS memory sub-system and the external on-board DDR memories
- V_{CCINT}: powers the PL internal logic circuits, such as CLBs, DSPs, PL clock network, routing resources and clock manager blocks
- V_{CCAUX}: powers auxiliary circuits on PL, such as IODELAY blocks, output buffers, etc.
- V_{CCBRAM}: powers the RAMBs of PL

The XC7Z020 FPGA under investigation is hosted on a ZC702 development board. This particular board hosts an external power controller chip (UCD9248), which we use to regulate the aforementioned voltage rails. The communication with the power controller is realized by using the I2C interface of PS and the PMBus communication protocol.

Test Case: Harris Corner Detection Application

The application that we use as test case scenario to demonstrate our exploration methodology is Harris Corner Detection [63]. Harris is a widely used image processing algorithm for detecting points of interest (corners) on a captured image. We utilize Harris due to its diverse computational complexity, stemming from the usage of convolution-based processing tasks and fixed-/floating-point arithmetic formulas evaluation. Algorithmically, for each pixel Harris computes a cornerness strength according to the following formula:

$$R = I_x^2 I_y^2 - (I_x I_y)^2 - 0.04 \cdot (I_x^2 + I_y^2)^2 \tag{2.9}$$

The processing flow of Harris in hardware is as follows (Figure 2.10):

- **Step 1**: the pixels of the image are stored in a dedicated image memory on the FPGA.
- **Step 2**: the image memory feeds the module responsible for computing the image derivatives using a Sobel operator and then stores the results on the derivatives memory.
- **Step 3**: the derivatives of the image are fed to a multiplication module that computes I_x^2, I_y^2 and $I_x I_y$. These quantities are then stored as a triplet on the square derivatives memories.
- **Step 4**: the stored squared derivatives are smoothed with a Gaussian kernel and the results are stored back at the square derivatives memories overwriting the previously stored derivatives.
- **Step 5**: the smoothed squared derivatives are forwarded on the cornerness calculation module to compute the formula for cornerness R and the results are stored back again at the square derivatives memories.
- **Step 6**: finally the computed cornerness values are fed to a non-maximal suppression and interpolation unit before going back on the processing system.

For the purpose of our exploration, two different implementations of Harris are used. The first one is a HW/SW co-designed system as described above, utilizing both the PS and PL sub-systems. The second one is SW implementation where Harris is executed exclusively on a CPU processor. For the HW/SW implementation, a hardware description of Harris algorithm in VHDL is employed. Harris hardware IP can be configured to use either the logic resources (e.g. LUTs, carry chains), or the embedded DSP blocks for the mapping of its arithmetic operations. This configurability gives us the possibility to investigate and compare the power gains that arise from different resource utilization choices. Figure 2.10 illustrates the architecture of Harris IP core, as implemented on the FPGA. Notice

that the type of resources utilized for the implementation of each distinct component of Harris is given.

System Integration

We begin with the HW/SW implementation. From hardware perspective, Harris IP adopts the AXI4-Stream interface for efficient PS-PL communication, as shown in Figure 2.10. A Direct-Memory Access (DMA) core on PL is used to read image pixels directly from the external DDR memory and stream them to Harris IP, while at the same time accepts the derived results of Harris on a streaming fashion and stores them back to DDR for further processing in PS.

From software perspective, an Asymmetric Multiprocessing (AMP) scheme is adopted. On the one hand, CPU0 is responsible for the communication with the on-board power controller and the power management of the system; voltage monitoring/scaling and power monitoring. On the other hand, CPU1 is responsible for reading the input data, configuring the DMA transactions to/from Harris IP and verifying the derived results with apriori known golden data (correct results). Regarding the software-only implementation, only the PS part is used. Again, CPU0 is committed to power management tasks, while CPU1 executes all the tasks related to the execution of Harris and the verification of results.

Figure 2.11 illustrates the finalized system including the communication/interaction between the various on-chip and off-chip components. Note that, Harris on CPU1 refers to either SW or HW/SW implementation, while the entire PL part refers only to HW/SW implementation.

2.3.4 System Evaluation

Exploration Methodology

This subsection presents our exploration methodology relating to guard-bands customization of the various SoC components in terms of supply voltage. Different scenarios are investigated to obtain an in-depth understanding of the real limitations of the underlying hardware when voltage scaling is applied.

Our exploration relies on the concept of finding the minimum voltage level for each specific

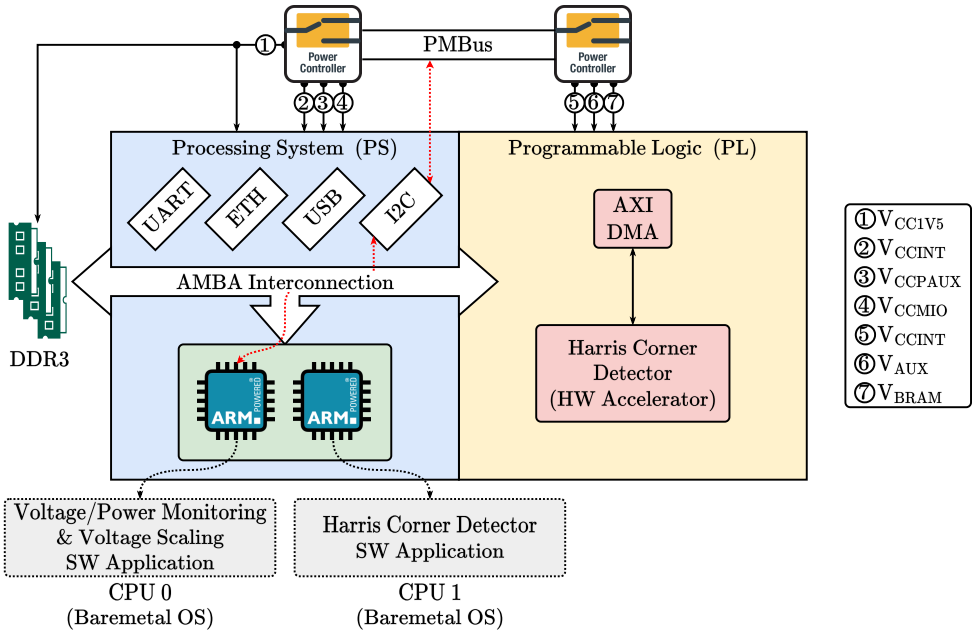


Figure 2.11: Finalized architecture used for voltage scaling and guard-band exploration.

power domain of the SoC, such as the highest power gain is achieved without hindering the functionality and timing performance of the application. We consider two different scenarios: i) the individual regulation of each voltage rail while keeping constant the remaining rails at their nominal values, ii) the combined tuning of all available voltage rails. In the first scenario, we investigate the real limitations of the components residing in a specific power domain when voltage is customized individually from the remaining system. In the second scenario, we investigate the combined tuning of all available rails in tandem with the specifics of the given application towards providing the optimal power gains. Specifically, the rails with the highest priority of tuning are the ones that impact the operation of the application the most. For instance, in the SW-only implementation that is V_{CCPINT} powering the CPUs performing the entire processing of Harris application. On the other side, in HW/SW implementation, the V_{CCINT} on PL is the most critical rail powering the CLB and DSP resources where all the logic and arithmetic operations of Harris take place. In general, the priority given to each voltage rail depends on the specifics of the target application. In case of a memory-intensive application, the rail supporting the memory blocks of the FPGA, e.g., V_{CCBRAM}, contributes more to power consumption than other rails, while in case of an I/O-intensive application, rails such as the V_{CCPMIO} becomes of increased importance for the reduction of power. Note that the pattern used to tune all the available rails may lead to varying power consumption

results, thus a careful strategy should be followed based on the characteristics of the target application. For Harris application, the tuning of the rails is performed according to the following order:

- PS-only implementation: V_{CCPINT}, V_{CC1V5}, V_{CCPAUX}, V_{CCMIO}
- PS-PL implementation: starting with PL we tune V_{CCINT}, V_{CCBRAM}, V_{CCAUX} and moving on PS V_{CCPINT}, V_{CC1V5}, V_{CCPAUX}, V_{CCMIO}

Based on our exploration, the above combinations lead to the highest power gains while preserving system functionality.

The aforementioned exploration is applied to both SW and HW/SW implementations of Harris. For each distinct implementation, we want to investigate the available headroom for guard–band customization of the various power domains, the strategy followed for the optimal tuning of voltage rails based on the different workload distribution, as well as the maximum achieved power savings. A step further, in HW/SW implementation, we include to our exploration/investigation the different utilization scenarios of hardware resources, i.e., CLBs versus DSPs.

Experimental Results

This subsection provides the results derived by our exploration methodology. Throughout all the experiments, the step used for the scaling of voltage is fixed at 10 mV. The minimum acceptable voltage level for each rail is considered the one that the application preserves its functional correctness. It is important to highlight that, the available headroom for voltage scaling is limited to ∼14% for the majority of voltage rails (despite V_{CCPAUX}) irrespectively of the design's workload; the specific FPGA board becomes non-functional under further scaling.

PS-only implementation: We begin with the illustration of power consumption results for each PS power domain separately. Each time we regulate the voltage of a distinct rail, the remaining rails are kept constant at their nominal voltage values. The derived results are listed in Table 2.3. We provide the nominal and minimum achieved voltage levels for each rail, as well as the power consumption for the corresponding power domain. Among all voltage rails, V_{CCPAUX} presents a significantly higher headroom for scaling, i.e., 31%, and hence the highest gain in power consumption, i.e., 48%. This can attributed to the fact that many components supplied by V_{CCPAUX} are not utilized during the execution of Harris application. Considering the other rails, even though they are scaled at the

Table 2.3: Power consumption results for individual regulation of PS voltage rails.

Rail	Voltage (V)			Power (W)		
	Nominal	**Minimum**	**Difference**	**Nominal**	**Minimum**	**Difference**
V_{CCPINT}	1.00513	0.85997	14.4%	0.45023	0.32152	28.6%
V_{CCPAUX}	1.80374	1.24577	30.9%	0.14936	0.07749	48.1%
V_{CCMIO}	1.80244	1.54625	14.2%	0.05634	0.04828	14.3%
V_{CC1V5}	1.50346	1.28641	14.4%	0.59755	0.45317	24.2%

Table 2.4: Power consumption results for combined regulation of PS voltage rails.

Rail	Voltage (V)			Power (W)		
	Nominal	**Minimum**	**Difference**	**Nominal**	**Minimum**	**Difference**
V_{CCPINT}	1.00513	0.88136	12.3%	0.45023	0.34172	24.1%
V_{CCPAUX}	1.80374	1.27507	29.3%	0.14936	0.07977	46.6%
V_{CCMIO}	1.80244	1.57443	12.7%	0.05634	0.04913	12.8%
V_{CC1V5}	1.50346	1.32228	12.1%	0.59755	0.42472	28.9%
Total PS Power				1.25348	0.89534	28.6%

same level, different power gains arise in each domain. The highest gain is achieved in V_{CCPINT} domain, i.e., 29%, as this is where the most activity of Harris execution take place. The minimum gain is measured in V_{CCMIO} domain, i.e. 14%, which is reasonable as the activity in the I/O interfaces of PS is rather low; only one serial controller is used.

Next, we continue with the evaluation of power consumption of the entire SoC in tandem with the combined regulation of all PS voltage rails. In Table 2.4, we present the tuning of all voltage rails and the resulting power consumption. We measure a significant reduction of total power, i.e., 29%, while the application remains functional without hindering its nominal timing performance. Notice that, the available headroom for voltage reduction is smaller compared to the case where each rail is regulated in a standalone fashion (Table 2.3).

PS-PL implementation: In this implementation, we begin first with the evaluation of the power domains related to PL sub-system. As described in Section 2.3.3, we employ

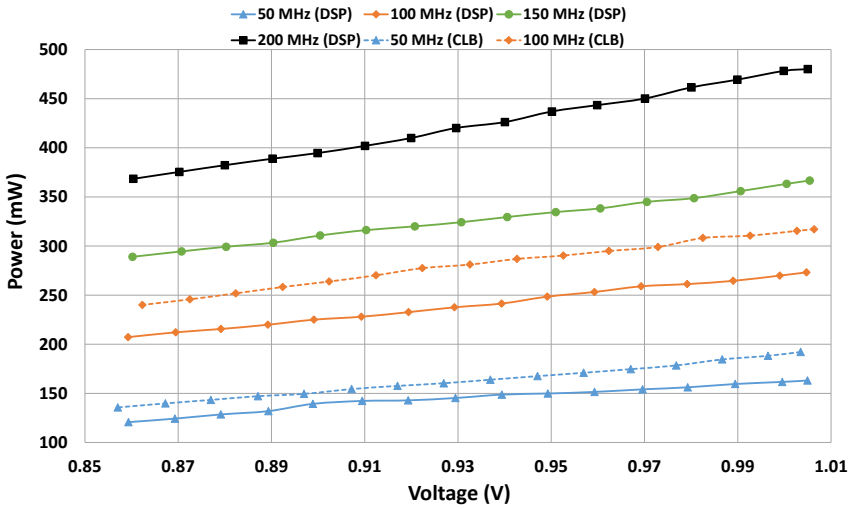


Figure 2.12: Voltage scaling results of V_{CCINT} in function of PL frequency.

two different variations of Harris hardware IP: i) utilizing only CLBs, ii) utilizing the embedded DSPs for the mapping of arithmetical operations. According to the static timing analysis (STA) tool, the DSP variant achieves double the performance, i.e., 200 MHz, compared to the CLB variant, i.e., 100 MHz. However, for our exploration we consider multiple operating frequencies for each variant; 50–200 MHz for DSP and 50–100 MHz for CLB.

We begin with individual assessment of the different PL voltage rails. In Figure 2.12, we illustrate the voltage scaling results for the V_{CCINT} rail, which is considered the most critical for the operation of Harris on PL. Considering the maximum achieved frequency, we measure on average 23% and 24% less power compared to nominal operation for DSP and CLB variants, respectively. Considering the exact same frequencies, the DSP variant consumes 1.5% and 2.6% less power than the CLB for 50 MHz and 100 MHz, respectively. These results are justified by the fact that, the implementation of the same arithmetic circuits (convolutions and multiplication in our case) requires higher utilization of logic and routing resources in case of CLBs, hence leading to higher power consumption.

In Table 2.5, we provide the voltage scaling results for all PL rails considering the case of 100 MHz operating frequency; the highest common frequency for both Harris variants. Initially, all voltage rails in both cases are scaled almost at the same level. In V_{CCINT} power domain, powering the processing components on PL, the DSP variant achieves slightly better power reduction compared to CLB variant. Regarding the V_{BRAM} power domain, the power reduction is identical in both cases which is reasonable as the utiliza-

Table 2.5: Power consumption results for individual regulation of PL voltage rails for 100MHz operating frequency.

	Rail	Voltage (V)			Power (W)		
		Nominal	**Minimum**	**Difference**	**Nominal**	**Minimum**	**Difference**
DSP	V_{CCINT}	1.00472	0.85925	14.5%	0.27314	0.20720	24.1%
	V_{CCAUX}	1.80717	1.54586	14.5%	0.06284	0.05353	14.7%
	V_{CCBRAM}	1.00917	0.85799	15%	0.04730	0.04020	15%
CLB	V_{CCINT}	1.00637	0.86232	14.3%	0.31704	0.24003	24.3%
	V_{CCAUX}	1.80744	1.54589	14.5%	0.06732	0.05576	17.2%
	V_{CCBRAM}	1.00754	0.85512	15.1%	0.04731	0.04020	15%

tion and activity of RAMBs remains the same. Regarding the V_{AUX} power domain, the CLB variant achieves noticeable better reduction of power. However, its power consumption is higher both on nominal and minimum voltage level.

Following, we evaluate the combined customization of all PL and PS rails concurrently. In Figure 2.13, we depict the total power consumption of the different HW/SW configurations distinguishing the power consumed by PS and PL sub-systems. As can been seen, the majority of power consumption is attributed to PS. Overall, for the entire SoC we measure on average $\sim 19.5\%$ total less power compared to nominal operating conditions. Specifically, in PS the power is reduced by $\sim 19\%$, while in PL by $\sim 21.5\%$ on average.

In Table 2.6, we present the combined voltage scaling results for PS and PL sub-systems for both variants of Harris (DSP and CLB), considering the common operating frequency of 100 MHz. Starting with the PS sub-system, the first thing to note is that the V_{CCPINT} domain shows slightly better headroom for voltage scaling than in the case software-only implementation (Table 2.4). This is reasonable, considering that the workload in V_{CCPINT} domain has been reduced; the main processing of Harris now is performed on PL. On the other hand, the other PS rails exhibit quite different behavior with smaller voltage scaling and reduced power gains compared to software-only implementation. This reflects that in the co-design implementation, even though the main processing is performed on PL, the overall system highly depends on the components residing on PS as well. For example, the DMA logic on PL depends on the memory sub-system of PS that is powered by V_{CC1V5}. Taking also into account that the processing speed of the co-design is significant faster than the software-only implementation, the workload imposed on this domain does not allow to achieve the same level of gains, i.e., only 9% voltage reduction with 15% power

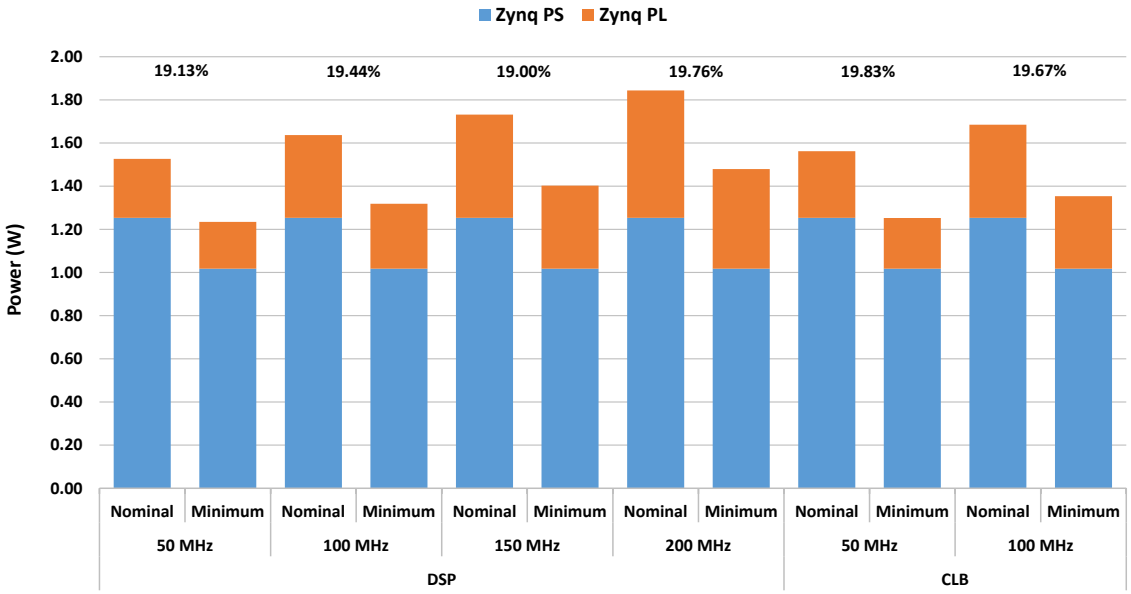


Figure 2.13: Total power consumption of various PS-PL system configurations.

Table 2.6: Power consumption results for combined regulation of PS and PL voltage rails for 100MHz operating frequency.

		Rail	Voltage (V)			Power (W)		
			Nominal	Minimum	Difference	Nominal	Minimum	Difference
PS		V_{CCPINT}	1.00513	0.86951	13.5%	0.45023	0.32843	27%
		V_{CCPAUX}	1.80374	1.66607	7.6%	0.14936	0.13116	12.2%
		V_{CCMIO}	1.80244	1.66554	7.6%	0.05634	0.05206	7.6%
		V_{CC1V5}	1.50346	1.36673	9%	0.59755	0.50603	15.3%
PL	DSP	V_{CCINT}	1.00500	0.85978	14.4%	0.27314	0.20720	24.1%
		V_{CCAUX}	1.80703	1.54422	14.5%	0.06284	0.05353	14.8%
		V_{CCBRAM}	1.00841	0.86394	14.3%	0.04730	0.04020	15%
	CLB	V_{CCINT}	1.00500	0.85978	14.4%	0.31704	0.24003	24.3%
		V_{CCAUX}	1.80703	1.54422	14.5%	0.06732	0.05576	17.2%
		V_{CCBRAM}	1.00841	0.86394	14.3%	0.04730	0.04020	15%
Total PS Power						1.25348	1.01768	18.8%
Total PL Power (DSP)						0.38328	0.30094	21.5%
Total PL Power (CLB)						0.43167	0.33599	22.1%

savings. The same applies for the V_{CCPAUX} and V_{CCMIO} rails, where the first powers the PS-PL AXI communication infrastructure that is heavily used on the PS-PL co-design and the later is utilized more often due to increase processing speed. Regarding the PL sub-system, the voltage reduction and power gains are the same with the minimums found during the exploration of the PL rails individually (Table 2.5). This shows that the PL has a more consistent behavior. The results in power consumption justifies that the most demanding domain in PL is the V_{CCINT} where the main processing of Harris is realized.

Discussion & Guidelines

The results presented so far exhibit that significant headroom exists for guard-bands customization in terms of supply voltage and that significant power gains can be achieved in practice. According to our exploration various conclusions are derived:

- First, despite the fact that the majority of voltage rails are able to be configured up to 14% lower than their nominal, the power gains in each power domain are not identical and depends on the specifics of the target application. For instance, the hardware implementation of Harris consumes 24% less power on V_{CCINT} domain and 15% on V_{CCBRAM} domain. Again, we highlight that the specific FPGA board allows for up to 14% scaling of voltage for the majority of voltage rails, irrespectively of the employed design. Thus, in a different situation/system higher gains would be expected.
- Second, another observation is that the reduction of voltage in each power domain is higher when regulating the corresponding voltage rail separately from the remaining system. When combining the regulation of all voltage rails, the available margins for voltage scaling are decreased for each power domain. For instance, when regulating only the V_{CCPAUX} rail alone, 31% lower voltage level is achieved, while when regulating all the rails in a combined fashion, the counterpart reduction is 8%.
- Third, we stress the importance of tuning each power domain to its optimal voltage level, which may not be the same with the other existing domains. Notice that, in case of a global mechanism for the tuning of all power domains simultaneously, at the same rate, would lead to significant power loss. For instance, in the hardware/software implementation of Harris, the global reduction of voltage in all domains, e.g., by 10%, results to 17% power savings, while according to our methodology we achieve 20% savings; which expected to be higher if there was the possibility of scaling the voltage of many rails over 14%.
- A fourth observation is related to the utilization of the FPGA resources. Our

exploration exhibits that the appropriate selection of the resources for the mapping of a given design impacts the final power figures.In Harris design, the usage of DSP blocks offers better performance/watt compared to CLB utilization; for the same operating frequency consumes up to 2.6% less power. Thus, depending on the kind of operations, e.g., big multiplications, the mapping on DPS blocks may be considered a preferable solution.

2.4 Enabling SoC-FPGA Sharing: Bridging Low-Level and High-Level Computing at the Edge

2.4.1 Introduction

The upcoming generations of mobile communications have to support an increased number of connected devices as well as novel and/or enhanced high demanding services, while keep the Quality-of-Service (QoS) high. Consequently, the emerging telecommunication technologies have to provide increased signal bandwidth and very low communication latency and to support the beyond 5G applications have to include edge compute nodes that are high-performance, low-power, and flexible/re-programmable, all at the same time. Moreover, these multi-layer network architectures and the remote edge computing nodes impose the need for processors that are powerful and efficient in executing both low- and high-level functions, e.g., telecommunication DSP pipelines and AI/ML.

In these novel network generations, a key element that has to meet all the aforementioned needs and requirements is the Baseband Unit (BBU) with a major role of (de)modulating the transmitting/receiving data. To tackle the challenge of designing a BBU meeting the above demands and specifications, this work proposes implementing concurrently such diverse functions on a single COTS device by exploiting the novel capabilities of very complex SoC-FPGA, i.e., RF-SoC and Versal ACAP. We explore the possibility of dividing logically the SoC in two parts: the low-level telecommunication implementation and the high-level AI/ML acceleration. The former will utilize hard-IPs for analogue and digital interfaces towards bridging the radio and core parts of the network, as well as CLBs for the radio signal processing. The latter will rely on CPU cores and the remaining CLBs to perform HW/SW co-processing of the AI/ML algorithms. Primarily, we evaluate the overhead of implementing the various interfaces and assess the FPGA space remaining for our high-performance processing; secondarily, we estimate the achievable throughput of real-time DSP telecommunication processing pipelines and representative

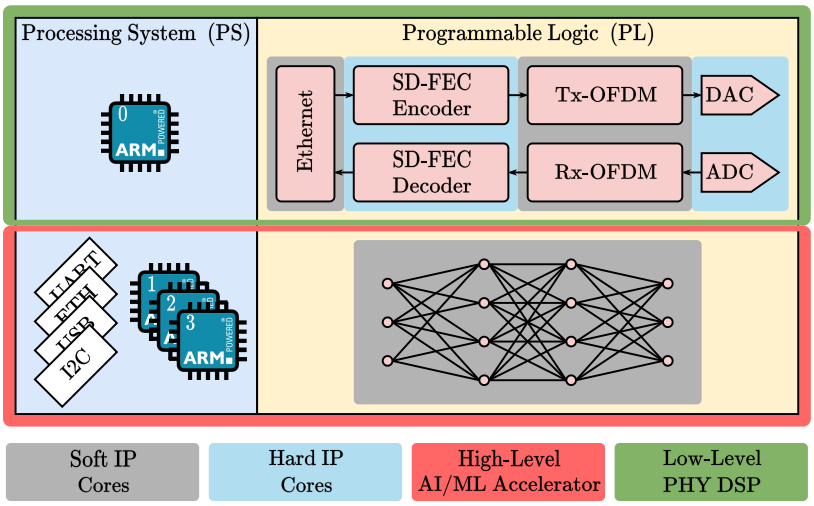


Figure 2.14: SoC FPGA partitioning to accommodate low-level PHY processing and AI acceleration with HW/SW co-processing.

AI accelerators when placed together in the FPGA. Overall, we explore the trade-offs and programming approaches while tuning function resources and changing architectural configurations.

2.4.2 Proposed Architecture

The implementation proposed in this section considers a single SoC-FPGA divided in two logical parts, one for low-level real-time DSP functions and one for off-line best-effort AI acceleration. Figure 2.14 shows an indicative partitioning of Xilinx RF-SoC. The hard-IPs for DAC/ADC and FEC are assigned to partition L, whereas the majority of CPU cores/peripherals are assigned to partition H (one core is still reserved for monitoring/controlling L and implementing auxiliary telecom functions, e.g., the EVM calculation). The FPGA's programmable logic (PL) is shared among L and H partitions, such that L can complete the signal modulation/demodulation in the given deadlines, while H can exploit the remaining PL to maximize the speedup of the given SW functions executing on PS. The L partition will utilize the PL to implement soft-IPs for digital interfaces (25G Ethernet, AXI stream, etc) and VHDL components for DSP (QAM, FFT, etc). The H partition will utilize the PL to implement AI functions (convolutions, activations, etc). The exact ratio of L to H resources is application dependent: higher telecommunication throughput specifications will lead to lower AI acceleration. For evaluation purposes in this exploration study, we assume telecommunications bi-

trates in the area of 5Gbps (e.g., up to 3Gbps useful data or 6Gbps OFDM data over the channel).

When assuming a device offering different hard-IPs, i.e., AI engines instead of FEC, as in the case of Xilinx Versal (ACAP), then the proposed partitioning changes to accommodate new soft-IPs on the PL side. The aforementioned AI functions will move from the PL to the hard-IP engines of the SoC, and hence, will leave space in the PL to implement soft-IPs for FEC. The trade-off between hard- and soft-IPs is also application dependent: lower RF specifications decrease the need to use hard-IPs for telecommunication and allow us to increase the performance of higher-level functions via hard-IPs for AI/ML and multi-Gbps Ethernet (to exchange more data with the core network instead of the RF domain).

The remainder of this section discusses those distinctive features of RF-SoC, which enable an efficient solution for the BBU implementation, together with AI/ML, at the far edge.

2.4.3 Baseband Processing Unit (BBU) Partition

RFSoC Analog-Mixed Signal (AMS) blocks consist of two main parts: a) the digital datapath and b) the actual RF-converters. The digital datapath is able to perform different DSP functions that used to be implemented on the FPGA fabric or by external components. First of all, in the boundary between the FPGA fabric and the AMS blocks exist configurable CDC FIFOs. These FIFOs regulate the data traffic and provide a flexible interface from/to custom DSP function on the FPGA fabric. The next sub-components, implemented on the digital path of AMS blocks, are the interpolation/decimation filters. The number of filters enabled is configurable and offer interpolation/decimation of signals up to x8 for the 1^{st} and 2^{nd} generation of RFSoCs, while the 3^{rd} generation further expands this capability with up to x40 factors. However, the filter coefficients are not programmable. Next, are the Digital Complex Mixers (DCM), which support real or complex output signals. The DCMs support three modes of operation: a) no mixing, b) coarse and c) fine. When configured to do coarse mixing only a few number of Intermediate Frequencies (IFs) are supported, which are sub-multiples of the sampling frequency. When doing fine mixing, the user is able to program the intended IF to the embedded Numeric Control Oscillator (NCO) of the DCM during the design phase or even create appropriate logic (hardware or software) to adapt the IF in real-time based on requirements at the moment. Another, common sub-component of the AMS blocks, is the Quadrature Modulator Correction (QMC) block. When the RFSoC are interfaced with external analog quadrature mixer devices, due to unpredictable events, imbalances to the RF signals

occur. These imbalances can lead to performance degradation of the BBU. The QMC blocks offer a convenient way to correct these imbalances, but the process of detecting the imbalances is still left to the designer, who must design the algorithm. One more feature of both DACs and ADCs in RFSoC is their capabilities on generating multi-band signal. Specifically, multiple digital datapaths can be combined and drive a single DAC for generating multi-band signals. Accordingly, the ADCs can distribute a multi-band signal to a different number of digital datapaths able to process the data of a single band. But, as with the previous blocks, there some limitations. When processing real signals up to four band can be combined, while for complex signals up to 2. The DSP functions on the FPGA process two data streams independently producing a complex signal and feed two distinct AMS blocks. After interpolation and up-conversion to their respective IF (Digital Up-Conversion - DUC), the two bands are combined and drive a single DAC. The DAC of the AMS block not used is powered down in this operation mode leading to reduce power consumption. Until now, the blocks described are common for both DACs and ADCs. However, there are some features distinct to DAC and ADC AMS blocks. The ADCs have built-in signal detection circuit that is able to report the strength of the received signal. The primary use of such feature is the realization of an Automatic Gain Control (AGC) mechanism. AGC is used so that the full dynamic input range of an ADC is used, by responding to varying signal amplitudes. The RFSoC devices provide only an indication of the signal strength and is left to the system designer to implement the appropriate algorithm to compute the required gain and apply the correction. For the DACs, there exists another filter before the digital to analog conversion implementing an Inverse Sinc operation. Thus, the DACs are able to have a flat response in a wider bandwidth.

2.4.4 AI/ML Acceleration Partition

Hardware devices such as FPGAs have been introduced to tackle the high computational demands in the edge domain where power efficiency and low latency poses a critical issue. Our representative AI solution relies both on the PS and PL side of the SoC. The FPGA communicates through PCIe to feed the data under a 16-lane endpoint configuration. This section will describe the varying levels of abstraction available for FPGA design regarding AI applications, from the hardware description languages (HDL) that operate on the circuit level and demand RTL expertise to the generalized computation engines that can be application agnostic and require little programming effort.

HDL design: Register-transfer-level (RTL) abstraction is used in HDL languages (i.e. VHDL or Verilog) to create low-level representations of an AI model, from which a mixture of registers and boolean equations is derived. Unlike in software compiler design,

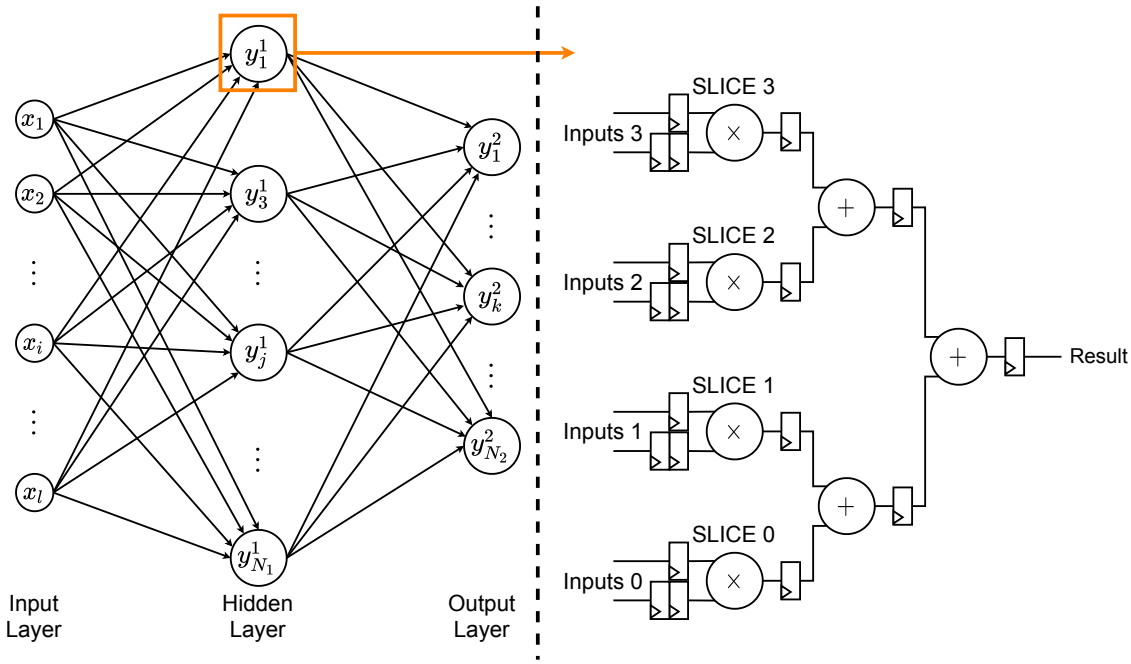


Figure 2.15: An abstract model of an FPGA implementation in circuit level derived from a neural network topology [26]

RTL takes as input the register transfer level representation and involves constructs such as cells, functions, and multi-bit registers. In order to accelerate an AI algorithm on hardware several Processing Elements (PEs) that perform multiply–accumulate operations are hand-written in Verilog or VHDL code that are designed and optimized based on the constraints of the target platform (i.e. the number of multipliers, etc). Mapping an AI model's computational operations to matrix-matrix or matrix-vector multiplication modules has been widely applied in prior studies [64–66]. Also, usually tiling and ping-pong double buffers techniques are then employed to improve the throughput along with other optimizations on the RTL level. Thus, this method can make the design of an AI application strenuous as opposed to software programming but the high reconfigurability of HDL can often be a key requirement especially in the AI market which is constantly evolving.

HLS design: Employing High Level Synthesis for FPGA design enables a fast development process and high flexibility. Although using HLS provides a software-like tool flow (i.e. Xilinx Vitis or Intel Quartus Prime), the developer must still learn hardware-centric concepts, such as pipelining and routing, that they may have not been exposed to in writing C-code for traditional processors. Also, studies have shown that HLS can simulate

faster the effects of data type precision and other hardware approximations which are often utilized on AI applications [67,68]. HLS automatically generates an RTL testbench which is driven by vectors generated by the original C++ code which is essential for simulating AI inference with bit-accurate precision. Also, in the case of Neural Networks, an HLS compiler can synthesize each layer as a separate module where every layer usually feeds its output to the next in a dataflow architecture with streaming transfers [69]. Additionally, each layer's precision can be controlled seamlessly by changing the weights or biases bitwidth using the HLS fixed-point arbitrary precision data types. Last, previous work has shown that HLS designs can exploit significant parallelism compared with RTL approaches and achieve similar latency improvements [70].

Dedicated DPU kernels: Responding to the extreme demand by the new emerging AI algorithms, HW companies introduced AI specific engines which have higher compute density and lower power requirements than processing on their traditional processing elements. For example, Xilinx has developed Vitis AI which is a complete development stack for AI inference targeting Xilinx hardware platforms, including both edge and cloud devices. Vitis AI uses an optimized IP named DPU (Deep Learning Processing Unit) which is a programmable engine optimized for convolutional neural networks. The Vitis AI tool converts the AI models into a graph-based intermediate representation which is then compiled for the DPU IP by generating the instruction stream along with coordinating data transfers.

2.4.5 Preliminary Evaluation

This section presents the software and hardware environments used to perform exploration and testing. FPGA resource utilization is estimated for both low-level PHY DSP and AI/ML accelerator, as well as their performance. Additionally, we performed an exploration of different AI/ML and low-level DSP configurations to determine the feasibility of our proposed architecture. Our target platform is ZCU111 evaluation board hosting XCZU28DR RF-SoC FPGA.

Software, Device setup, Benchmarks

The low-level PHY DSP evaluation we performed on Xilinx Vivado Design Suite with VHDL coding and IP cores, while for the AI/ML we focused on evaluating several DNNs using Xilinx Vitis AI development stack which is developed specifically for AI inference on Xilinx FPGAs. This framework consists of optimized IP, tools, libraries and exam-

Table 2.7: Accuracy and performance benchmark accross different DNNs and datasets

	Model specifications					FPGA results		
Category	***Model***	***GOPs***	***Img Size***	***Dataset***	***Float acc.***	***DPU acc.***	***Latency***	***FPS***
Image Classification	ResNet-50v.1	6.97	224x224	ImageNet	75.2%	74.8%	12ms	158
Image Classification	Mobilenet_v2	0.6	224x224	ImageNet	70.1%	67.7%	4ms	575.5
Object Detection	yolov3_voc	65.6	416x416	voc07_test	78.5%	77.4%	72.7ms	27.6
Semantic Segmentation	ERFNet	54	512x1024	Cityscapes	53%	51.7%	145ms	23.2

ple designs that supported many DNN topologies that we tested which were trained in popular Deep Learning frameworks such as Tensorflow or Pytorch. The models were also quantized into 8-bit precision using the Vitis AI quantizer and were compiled for the Xilinx DPU-B4096 architecture.

For AI/ML we utilized 1–3 B4096-DPU IPs for our exploration purposes. This configuration achieves 4096 operations/per clock for each DPU. The achieved frequency of the DPU IP was 300 MHz which translated to 2.56 TOPs of achievable performance. To use the DPU, we prepared the instructions and input image data in the specific memory address that DPU can access. The DPU operation also required the application processing unit (APU) to service interrupts to coordinate data transfer.

For the low-level PHY benchmarking, we assumed a Tx/Rx processing pipeline relying on OFDM and QAM. We estimated the FPGA resources by considering representative DSP function implementations and including margins based on various configuration/optimization levels (see Section 4 for details). Next to the DSP chain, we used the hard-IP FEC encoder/decoders. We assumed datapath pipelining and parallelization (multiple chains) to increase the overall telecommunication throughput. The operating frequencies of these configurations varied in the area of 250MHz.

Results

First, we measured the accuracy and performance of distinct neural networks for distinct application domains. Table 2.7 summarizes the results obtained with Vitis AI 1.3 using the B4096-DPU configuration along with INT8 bit quantization on DNNs trained on Tensorflow. The top–1 accuracy for float and quantized DPU models (or mAP for object detection models) is also reported in addition to the performance metrics of latency and Frames per Sec (FPS).

Table 2.8 gives the resource utilization of the basic parts of our proposed architecture. The AI/ML partition and part of the low-level PHY DSP (Ethernet core, L2 Tx/Rx)

Table 2.8: Resource utilization on XCZU28DR[1] RFSoC

Name	FF	LUT	RAMB	DSP
B4096-DPU	105008	53540	257	562
Tx-Rx DSP chain	50K–100K	10K–20K	6–14	180–370
PL PCIe	20201	8183	22	0
10G Eth. Core	22004	51929	3	0
10G L2 Tx+Rx	912	1581	8	0
Misc. (AXI, etc.)	7156	9599	13	0

[1] Total resources: FF = 850K, LUT = 425K, BRAM = 1K, DSP = 4.2K

partition and support logic for both partitions utilize around 19% FFs, 30% LUTs, 28% BRAMs and 13% DSPs. The resources used to implement one actual Tx-Rx DSP chain (row 3) vary due to our aforementioned estimation approach covering a range of possible implementations.

Lastly, we consider various combinations of the two main functionalities implemented on a single RFSoC device to explore the capabilities of our proposed architecture based on performance metrics and feasible utilization ratios for FFs/LUT/BRAM/DSP. Table 2.9 provides the utilization and the useful data rates of the most representative configurations. The more telecommunication throughput is required, the more low-level PHY DSP functionality will be implemented and the less AI/ML (eg. Table 2.9 rows 3, 4). On the other hand if more AI/ML acceleration is needed, more DPUs will be used and less low-level PHY DSP. Either case, from the table we observe that the resource consumption constrains are met. We note that the use of FIR filters for the low-level PHY DSP is application-specific and is not considered for all DSP chains in Table 2.9 (also, in certain cases/scenarios, they could be shared among chains). The most challenging configurations are those utilizing 3 DPUs and 4 low-level PHY DSP chains, together with the whole support logic and with/without the FIR filters, which start pushing our target device to its limits. Even so, with more aggressive optimizations during the design phase, all resource requirements of Table 2.9 can be met on RFSoC.

2.5 Conclusions

In this chapter, we presented our proposed design approach for embedded computing in SoC-FPGA platforms. To evaluate its applicability we took into consideration state-of-the-art SoC-FPGA devices and applied different combinations of the design ap-

Table 2.9: Performance-Utilization trade-off on XCZU28DR

Configuration	Throughput		Resources		
	Gbps	TOP/s	LUT	BRAM	DSP
A+4·B+E	2.5	0.5	39%–48%	30%–33%	30%–48%
A+4·B+4·F+E	2.5	0.5	51%–61%	30%–33%	78%–96%
2·A+4·B+E	2.5	1.12	51%–61%	54%–57%	43%–61%
2·A+4·B+2·F+E	2.5	1.12	58%–67%	54%–57%	67%–85%
2·A+2·B+2·F+E	1.25	1.12	53%–58%	53%–55%	59%–68%
3·A+2·B+E	1.25	1.34	59%–64%	77%–78%	48%–57%
3·A+2·B+2·F+E	1.25	1.34	65%–70%	77%–78%	72%–81%

A=DPU, B=Tx+Rx DSP, F=FIR, E={10G Eth., L2 Tx+Rx, PL PCIe, Misc.}

proach to distinct DSP applications. Specifically, we adopted a comprehensive design approach that involved extensive Design Space Exploration (DSE) at both the algorithmic and architectural levels to optimize our circuit implementations. Additionally, we systematically applied voltage scaling to reduce power consumption at the system level within the SoC-FPGA and conducted an initial exploration of co-deploying two demanding applications on the same SoC-FPGA. Our experiments produced the following outcomes:

1. For circuit optimization, we achieved up to 98% reduction in resource utilization compared to the accurate floating-point architecture of the same algorithm, while boosting its operating frequency by up to 122% and power consumption no more than 1Watt

2. With our systematic approach to voltage scaling and depending on the architectural choices for the accelerator in an HW/SW co-designed application deployed on the SoC-FPGA device, we achieved power savings of $20 - 30\%$ at the system level, while maintaining the functional correctness and nominal timing performance of the system.

3. We successfully showcased the practicality of combining low-level DSP functions required in a Baseband Unit with AI/ML acceleration on a cutting-edge SoC-FPGA device (RFSoC). Our initial evaluation confirms that this integration is feasible and delivers promising results in terms of resource consumption and performance metrics.

Chapter 3

Efficient Architectures for Image/Video Applications in Contemporary SoC-FPGA devices

This chapter introduces specialized hardware (HW) accelerators designed to achieve high-performance in image and video applications. The focus is on two specific domains: Medical Imaging (MI) and Vision-Based Navigation (VBN) in space. These domains have unique requirements, including real-time operation and low power/energy consumption. Implementing these accelerators poses significant challenges due to the widespread use of deeply embedded devices. This chapter is based on our publications in [71, 72]

3.1 Introduction

This chapter introduces specialized hardware (HW) accelerators designed to achieve high-performance in image and video applications. The focus is on two specific domains: Medical Imaging (MI) and Vision-Based Navigation (VBN) in space. These domains have unique requirements, including real-time operation and low power/energy consumption. Implementing these accelerators poses significant challenges due to the widespread use of deeply embedded devices. In the case of Medical Imaging, we explore an Image Registration (IR) processing pipeline, while for Vision-Based Navigation, our target application is the 6D Pose Tracking (PS) of uncooperative space objects (e.g. inactive satellites). To address these challenges, we propose efficient HW/SW partitioned systems for SoC-FPGA devices and introduce hardware-level optimizations to enhance performance. Throughout the chapter, we provide comprehensive performance results and compare our approaches to general-purpose embedded devices, enabling a thorough evaluation of the proposed HW/SW systems.

3.2 Optimizing Image Registration for Embedded Systems using FPGA-based SoCs

3.2.1 Introduction

Computing systems have invaded all aspects of human life in an effort to automatize and optimize important laborious tasks. Concepts and architectures such as Internet of Things and Edge computing have risen the bar in both computation and communication requirements of embedded systems [1]. In parallel, a flood of data is produced in unprecedented high rates leading to a saturation of the efficiency of general purpose computing systems and eventually paving the way for the integration of versatile and specialized computing accelerators [3].

A well established area of computing systems intervention is the medical domain, where electronic devices have revolutionized the way that patients are examined and treated [73]. While initially electronic devices were utilized mainly for medical data acquisition and visualization, there is a gradual wave of adoption of more sophisticated analysis mechanisms able to offload the critical tasks of a doctor. Towards this direction, the sampled patient data must be efficiently analyzed and correlated within short time frames, in order to assist diagnosis and reduce hospitalization. A significant amount of such

algorithmic techniques stems from the image processing domain, since medical imaging is a well-established and essential part of accurate diagnosis [74]. A key requirement in medical applications is the accuracy and dependability of the derived diagnosis, ergo sophisticated algorithms and high resolution image acquisition techniques are utilized.

The inherent trade-off of this design choice is the severe toll on the computation requirements of the developed applications in order to abide by the high standard requirements. Nevertheless, adequately fast response latency and cost effective designs are required to allow for the successful adoption of new, high-accuracy image processing algorithms. Consequently, the incorporation of accelerating computing sub-systems in medical devices is an appealing and effective design choice.

One of the most common acceleration techniques of modern computing systems is the integration of Field Programmable Gate Arrays (FPGAs), which bare the responsibility of executing the heavy tasks of the application. FPGAs can produce order of magnitude of acceleration, while preserving a low power consumption profile of the target computing systems. These features have established them as an acceleration alternative in the medical domain as presented in the works of [75–78].

This work targets image processing on FPGA-based systems, with emphasis on medical applications, utilizing computing platforms which integrate general purpose computing elements (CPUs) and FPGAs on the same System-on-Chip (SoC). This combination provides the flexibility for efficient HW/SW co-design, which if applied properly can get the most out of the FPGA sub-system. Most importantly, the design principles of this system are similar to the FPGA systems for large scale deployments (e.g. cloud based systems), thus enhancing the re-applicability and importance of our proposed design.

On the algorithmic side, we focus on image registration algorithms which is a fundamental task frequently encountered in image processing applications. It is used in computer vision [79], medical imaging [80, 81], remote sensing [82], military automatic target recognition [83] and satellite imaging [84].

Our methodology involves benchmarking and analysis of the sub-components of the image registration algorithmic pipeline in order (i) to optimize the algorithmic structure for our target dataset and (ii) to locate and accelerate the computational hotspots of the algorithm.

This approach enables us to develop an accurate and efficient accelerator.

3.2.2 Background

Image registration refers to the mapping between two images, both spatially and with respect to pixel intensity. It is used to match two or more pictures taken at a different time, or from different sensors/viewpoints [85]. One of the images is the fixed/source, while the others are the moving/targets. The reference frame in the fixed image is stationary, and the moving image is spatially transformed to match the target.

Image registration algorithms can be divided in two categories: 1) *Intensity based* and 2) *Feature based* [86]. Intensity-based methods compare intensity patterns within the images via correlation or error metrics. A pair of images, a similarity metric, an optimizer, and a geometrical transformation must be specified. Feature-based methods find correspondences between image features (such as points, lines and contours) and utilize them to estimate a global transformation which is finally applied to the moving image in order to align it. In general, image registration can be described as an iterative trial-and-error procedure. A measure of similarity or distance is computed between the images at each iteration and used to determine if they are sufficiently aligned. This process is controlled by the optimizer starting from an initial state and determining subsequent steps to reach an optimal alignment. A brief description of each of the three main steps is presented below.

- **Measure of match:** Image similarity metrics are methods used to calculate a quantitative evaluation of the similarity between two images. They act as a base in all registration applications, since they enable the optimizer to progress in the search of the optimum transformation. Commonly used measures include: *Sum of Absolute/Squared Differences, Correlation Coefficient, Matte's Mutual Information, Gradient Correlation* [87]. Two similarity metrics were examined in this study: the *Correlation Coefficient* and *Matte's Mutual Information*. The Correlation Coefficient provides a measure of the linear relationship between the two images. Matte's Mutual Information is a method that measures the amount of information that can be obtained for one of them by observing the other.
- **Transformer:** The transformer maps points of the moving image to new locations in the transformed image. According to the registration problem the transformer can be either collinear or deformable. Collinear transformations are the *rigid*, the *affine* and the *projection*. The rigid transformation includes translation and rotation only, whereas the affine transformation also includes scaling and shearing [88]. This work uses an affine transformation around the center of the image.
- **Optimizer:** The optimizer defines an efficient and often non-exhaustive strategy to search the allowed transformation space for the best match between the images. An optimizer can be categorized as gradient-based or gradient-free, global or lo-

Figure 3.1: Registration example using the iris dataset

cal [87]. In this study the *Downhill Simplex* and the *Powell's Direction Method*, both of which are classified as gradient-free and local [87] are examined. Other methods include: *Gradient-Descent*, *Soblex*, *Simulated Annealing*, etc. The Downhill Simplex utilizes the concept of the simplex, which consists of N+1 points in a N-dimensional space [89]. After initialization, the optimizer follows a series of steps that aim in moving the highest points of the simplex to lower ones. Convergence is usually achieved after multiple contractions of the simplex. The process terminates when the distance of the vector moved in two subsequent steps is below a threshold. Powell's is a direction method that produces N mutually conjugate directions. This method practically searches for a multidimensional function's minimum or maximum by moving along set directions.

3.2.3 Application Profiling & HW/SW Partitioning

To determine the most effective combination of optimizer and similarity measure, an exploration is performed. For this purpose, a dataset of gray-scale photographs of the iris of different eyes acquired by high resolution cameras (image size: 1000x1000). The registration of these eyes serves purposes in security and medical applications [81], emphasizing the need for thorough investigation. Figure 3.1 illustrates an image registration example on the utilized dataset. The first two images correspond to different photographs of the same eye. The left one is the moving image, while the second the fixed one. The registration result is depicted in the third image, where the moving image has been optimally transformed to be aligned to the fixed. An edge detection and an image fusion have also been applied for a visual representation.

Table 3.1: Computed similarity measure for various maximum displacement. Scaling and rotation are fixed at 10% and 30° respectively.

Maximum	Downhill Simplex		Powell's Direction	
Disp.	Correlation	Mutual	Correlation	Mutual
50	0.157844	0.261021	0.227008	0.285423
100	0.231213	0.374724	0.487449	0.385665
200	0.783975	0.788725	0.784077	0.788861
300	0.784035	0.789102	0.784041	0.789113

Evaluating the Optimizer and Similarity Measure Methods

As explained in Section 3.2.2, the *Downhill Simplex method* and *Powell's Direction method* are considered as the optimizing procedure, while for the similarity measure the *Correlation Coefficient* and *Matte's Mutual Information* are the methods under evaluation. Thus, four different combinations are derived and examined in order to choose the final configuration of the image registration application. The solution quality provided by these methods directly affects the registration's performance. For the dataset of the application, it was concluded that all four models had the same success rate. The affine transformation parameters and their maximum allowed values also influence performance. We experimentally concluded that typical values of 30° for rotation and 10% for scaling are sufficient for our target dataset. Conversely, the maximum value of displacement is what affects the alignment process the most.

Table 3.1 presents how the similarity measure is affected by the maximum displacement value and the different configurations. All methods achieve a higher similarity (and consequently a better alignment), when the maximum displacement is increased to a value that can sufficiently express the required geometric transformation. Therefore, both optimizers and both similarity metrics noted similar performance, provided that they were equally initialized. In total, the chosen solver for the hardware acceleration makes use of the *Downhill Simplex Optimizer*, the *Affine Transformation Model* and the *Correlation Similarity Metric*. The aforementioned components were preferred mainly due to their popularity and conceptual simplicity.

Figure 3.2 illustrates a high-level representation of the processing flow for a general image registration solver. The dashed boxes are optional steps, which were not necessary in this work, while the boxes with solid contours are the ones that are part of the Image Registration design. The flow of the application is as follows:

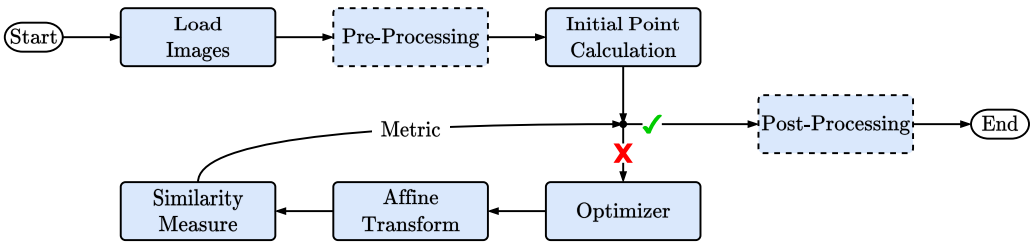


Figure 3.2: Processing flow of a general registration solver.

- **Step 1:** Images are loaded either from non-volatile memory or the moving image is dynamically captured from an imaging device and the fixed image is extracted from a database.
- **Step 2:** N+1 initial points are calculated. These points are required by the Downhill Simplex method and cover the whole range of the exploration space. For each of them an affine transformation is applied and the respective similarity measure between the fixed and the moving images is computed, thus creating the initial simplex in question.
- **Step 3:** Starting from the initialization of Step 2, the optimizer begins the evaluation process of the termination criteria. If they are satisfied then a valid transformation has been found and the application moves on to the post-processing stages. If the termination criteria are not satisfied, then Step 4 is executed.
- **Step 4:** The state of the transformation parameters are optimized and a new candidate transformation is determined. The corresponding similarity measure is computed and the optimizer again decides upon the completion or not of the registration procedure, by returning to Step 3.

Profiling of Image Registration Algorithm

The registration solver is implemented using the C programming language. Additionally, a further profiling procedure was performed to identify parts of the application that are computationally intensive, their memory requirements, as well as the type of operations involved. Besides its structure, the execution latency of the examined application is affected by the following parameters:

- *Image size*: As the image size becomes larger, the required latency to process it increases, becoming a serious bottleneck especially for image processing operations that are difficult to apply numerous pixels concurrently.
- *Tolerance threshold*: The threshold for determining a successful registration is a

Table 3.2: Profiling of registration for two pairs of eye images.

	Total Execution Latency (sec)	**Transform & Measure Execution Latency (sec)**
Matching pair	39.923572	39.9216161
Differing pair	103.084555	103.080554

trade-off between accuracy and the required computations of the application.

- *Iteration number*: In case the optimizer is unable to converge to a solution (e.g. very low threshold), after a predefined number of iterations the registration process is stopped. In this case, the results may not be optimal and a post-processing step must be taken. This value was set to maximum 500 allowed iterations.
- *Maximum displacement/rotation/scaling*: The initial transformation parameters are directly affected by the maximum allowed values of these parameters. Certain transformations may be rejected if the computed transformed coordinates lay out of the image boundaries. The maximum rotation is fixed at 30°, scaling at 10% and displacement at 300 pixels (Section 3.2.3).

The initial profiling was performed on the 1000x1000 images. Moreover, to decouple the measured profile of the application from the properties of the examined pairs, we examine both cases of successful and unsuccessful registrations. Table 3.2 presents the measured execution latency of such an example, where there is a pair of matching input images (Matching pair) and a pair of non matching images (Differing pair) of the input dataset. The Table includes the total execution latency of the application, as well as the execution latency of its Transform & Measure function. The latency of this function requires almost 99% of the total latency in both cases. Consequently, this is the algorithmic part that will be the focus of the accelerated design on the FPGA fabric.

Based on the profiling, a high-level HW/SW partitioning of the Image Registration solver is provided in Figure 3.3. The less computationally demanding parts, such as loading the images, tuning the transformation parameters and evaluating the termination criteria remain as software components, while the application of the candidate transformation to the pixel coordinates and the computation of the similarity measure are implemented on the FPGA.

3.2.4 Proposed HW/SW Image Registration Architecture

The implemented hardware accelerator consists of several sub-modules, including:

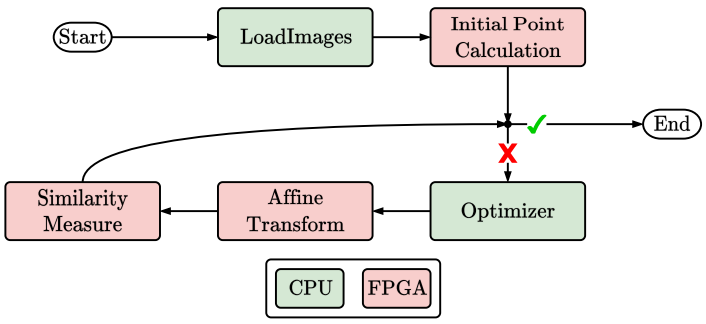


Figure 3.3: HW/SW partitioning of the Image Registration Solver.

(i) The *Transform* module, which applies the affine transformation to each pixel. It takes the pixel coordinates and current transformation parameters as inputs and generates the transformed coordinates, divided into their decimal and fractional parts

(ii) The *Interpolation Weights* module, responsible for generating interpolation weights used in computing the final intensity value of the transformed pixel. It takes the fractional parts of the transformed coordinates from the *Transform* module and generates the appropriate interpolation weights.

(iii) The *Interpolation Calculation* module,where the final intensity of the transformed pixel is computed. This module takes as inputs the weights and intensity values of the three pixels neighboring the transformed pixel, which are generated in the *Interpolation Weights* module, as well as the weight and intensity value of the transformed pixel itself.

(iv) The *Accumulations* module, comprising multiple accumulation units that compute the necessary sums for the correlation calculation between the images. It takes the intensity value of the interpolated pixel from the *Interpolation Calculation* module and the corresponding intensity value of the pixel on the moving image.

Moreover, the hardware accelerator integrates two memory modules dedicated to storing images, alongside multiple control units that coordinate its operations. These controllers generate pixel coordinates, manage memory accesses, buffering data, and regulating communication with the software application.

HW/SW Communication: Efficient communication between processing elements and memory is a critical design consideration in image processing applications, aiming to achieve high throughput and low latency data transfers. To address these requirements, a solution based on Direct-Memory Access (DMA) with streaming capabilities is employed. The streaming feature ensures compliance with throughput constraints, while

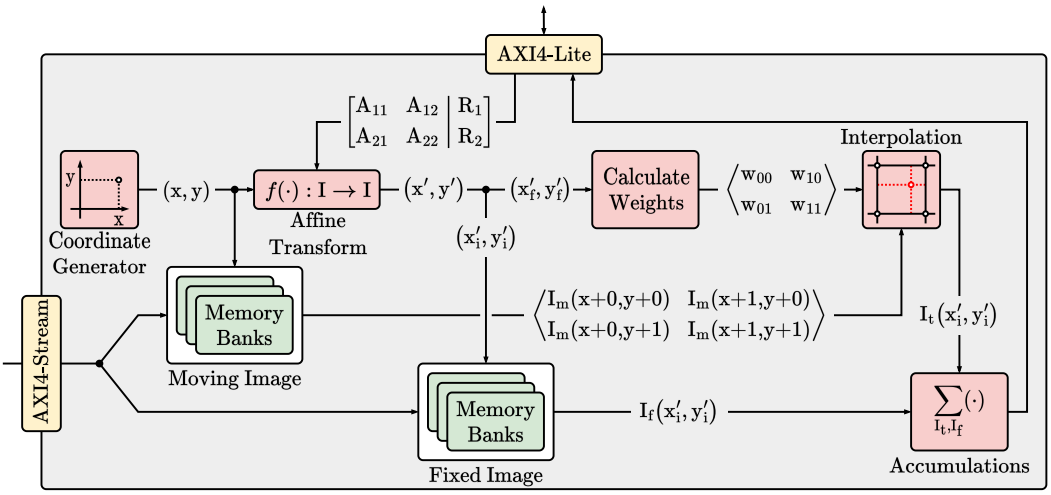
DMA enables direct memory access for the processing elements. The chosen platform offers these desired features by utilizing the AXI communication infrastructure. As a result, the hardware accelerator adopts the AXI4-Stream protocol for high-bandwidth image data transfers, leveraging its capacity for efficient transmission. For managing data transfers (i.e. control signals) with low-bandwidth communication requirements, the AXI4-Lite protocol is employed.

Memory Bottleneck: A common challenge in FPGA-based image processing applications is the limited availability of RAM Block (RAMB) resources for storing image pixels during processing. To address this issue, a widely adopted approach is the utilization of the sliding window technique [90]. The Image Registration application utilizes two images, the fixed image and the moving image. The transformation is applied to the moving image, resulting in new pixel coordinates that follow an irregular pattern. This irregularity creates difficulties in applying the sliding window technique or efficiently accessing the required pixel values. To overcome this obstacle, an offline preliminary step was introduced, downsizing the images from their original size of 1000x1000 to a reduced size of 256x256. Evaluation has confirmed that this downsizing strategy does not negatively affects the quality of the results By reducing the image size, it becomes possible to hold both images within the available RAMB resources of the FPGA, allowing for a straightforward memory-mapped access. The final HW/SW architecture of the developed Image Registration system is given in Figure 3.4.

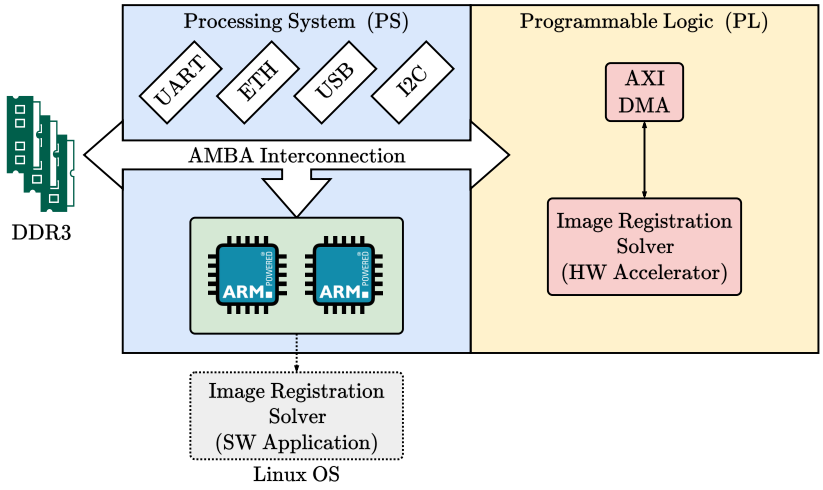
3.2.5 Experimental Evaluation

The evaluation targets the Zybo development board, featuring the XC7Z010 part of the Zynq-7000 SoC family devices. The Zynq platform consists of the Processing System (PS), hosting a dual-core ARM Cortex-A9 processor, and the Programmable Logic (PL) which refers to the FPGA fabric of the device. The HW/SW co-designed application is compared against software implementations executed on an Intel i5 4200U CPU and on the ARM CPU of the Zynq PS.

Each implementation is compared based on the quality of the results, which takes into consideration the calculated optimum transformation parameters and the similarity measure. Moreover, their execution latency is measured and the hardware cost in terms of FPGA resources is given. In order to have fair comparisons between the platforms all the measurements are performed after the dataset images are downscaled to 256x256 pixels in size in order to meet the memory constraint of the FPGA device (Subsection 3.2.4 - Memory Bottleneck).



(a) Image registration HW accelerator architecture.



(b) Image registration HW/SW integration

Figure 3.4: Finalized image registration solver architecture.

Table 3.3: Final affine transformation parameters and similarity measure on three different platforms.

Transformation Parameters	Intel i5 4200U	XC7X010 PS	XC7X010 PS+PL
T1	1.002974	1.010045	0.990064
T2	-0.015803	-0.015795	-0.017116
T3	45.320175	45.801357	44.415871
T4	0.024869	0.027911	0.021519
T5	0.997309	1.003294	0.989229
T6	-16.065662	-15.859092	-15.909966
Measure of Match	0.792186	0.792874	0.780669

Evaluation of Transformation Parameters

Our first experiments aim at providing quantified evidence that the hardware accelerator provides accurate results with respect to the reference software application. The six affine transformation parameters (2 for rotation, 2 for scaling and 2 for displacement) and the final similarity measure are used as quality metrics, with the software implementation on Intel i5 serving as reference.

Table 3.3 gives the parameter values of the final transformation to be applied on the moving image, as well as the final similarity measure computed on the three different platforms. All the computations were performed with single precision floating-point arithmetic in order to have an initial estimate of the migration of some software functions to the FPGA. As can be seen, the values of the transformation parameters differ slightly, but the difference in the context of the specific test case is acceptable.

Co-Designed System Execution Latency

Having validated that the computed transformation parameters have an acceptable deviation from the reference and that the final similarity measure is acceptably close in all platforms, the execution latency of the proposed HW/SW co-designed system is compared against the software only implementations on Intel i5 4200U and on ARM Cortex-A9 (Zynq PS). The comparisons are performed for a single-threaded C code ex-

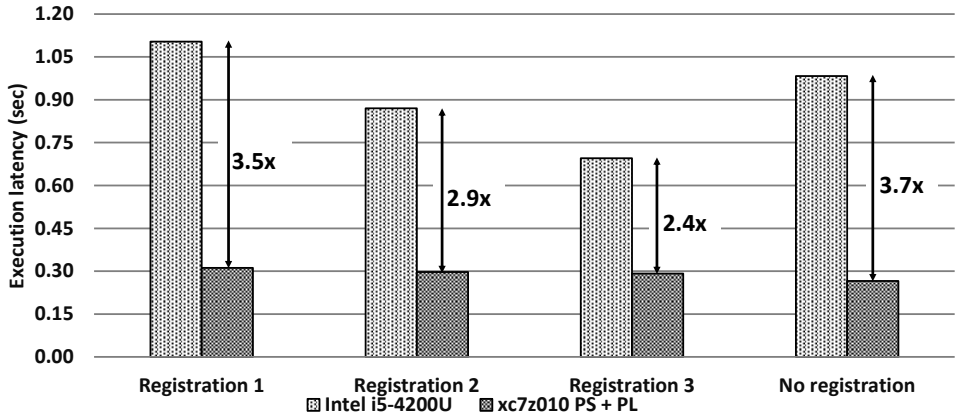
ecution on all platforms. The operating frequencies of CPUs were preserved at their base supported frequency, i.e. 650 MHz for the ARM Cortex-A9 CPU and 1.6 GHz for the Intel i5 CPU. The FPGA accelerator operated at 100MHz as reported by the synthesis tool. Also, note that for the following experiments the downsized images were used.

Figure 3.5a shows the measured average execution latency over 100 runs of the co-designed application versus the software only application on Intel i5 for four different sets of eye images. The first three out of which lead to successful registration and the last one fails. The achieved execution speedup is also annotated on the figure. In all cases, there is a decrease in execution latency and the average achieved speedup over the four test cases is $\sim$ 3x. The variable execution latency stems from the fact that the termination criteria is directly related to (i) the maximum number of iterations allowed on the optimizer, (ii) the tolerance threshold of the optimizer and (iii) on the nature of the images used. Small deviations in the parameters generated at the end of an iteration may lead to a varying convergence trajectory, which is caused by a difference in the optimizing steps.

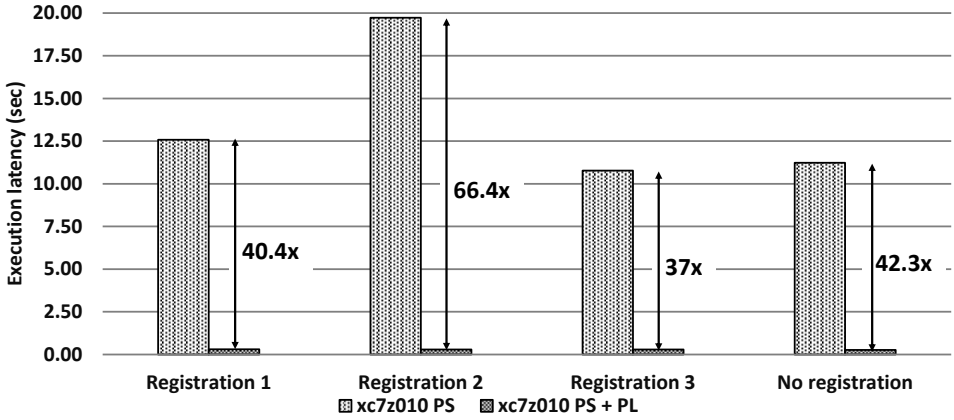
The experiment is repeated in a fully embedded execution environment and the obtained results can be seen in Figure 3.5b, which shows the execution latency of the presented design versus the software only implementation running on ARM Cortex-A9 (Zynq PS). It is clearly evident that there is a considerable decrease in execution latency in all four cases and the average speedup achieved is $\sim$ 46x. Again, observed fluctuation in the execution latency is due to the variable nature of the termination criteria of the optimizer. Nevertheless, the HW/SW co-designed application outperforms both software-only implementations, despite the use of single-precision floating-point implementation.

FPGA Resource Utilization

Table 3.4 presents the resource utilization of the final HW architecture. Even though the co-designed application uses single-precision floating-point arithmetic, it can still fit on the FPGA device, despite the high DSP utilization (95%). However, the high DSP utilization presents a major bottleneck for the implemented application, as the parallelization of the application in the FPGA becomes prohibitive. Moreover, the majority of the FPGA resources have been allocated for computation and not communication. Regarding memory resources requirements, the utilization of BRAMs is around 60%, which allows for the prospect on-chip analysis of images of higher dimensions.



(a) Comparison against Intel i5-4200U @ 1.6 GHz



(b) Comparison against ARM Cortex-A9 @ 650 MHz

Figure 3.5: Performance evaluation of the proposed HW/SW design

Table 3.4: Resource utilization on XC7Z010 Zynq device.

XC7Z010 PL Resources	**Available**	**Used Resources**	
		Accelerator	**Communication**
LUTs	17600	11396 (65%)	2768 (16%)
DFFs	35200	19965 (57%)	2656 (8%)
RAMBs	60	34 (57%)	1 (2%)
DSPs	80	76 (95%)	0 (0%)

3.3 Embedded SoC-FPGA Implementation for Vision-Based Navigation in Space

3.3.1 Introduction

Proximity operations in space are maneuvers conducted by an orbiting chaser spacecraft in order to approach and remain close to other space resident targets. Traditionally, proximity operations have been concerned with specially prepared, cooperative targets with controlled attitude. However, near future space-tug, on-orbit servicing and active debris removal missions [18–20], will also involve uncooperative/unprepared targets whose motion is not known accurately. As practical limitations preclude real-time teleoperation, to cope with any target state uncertainties while minimizing the risk of collision, the chaser should be maneuvered by a Guidance, Navigation, and Control (GNC) system with increased autonomy. Such a system should benefit from advanced Vision Based Navigation (VBN) [21]. However, advanced VBN demands significant on-board processing power, mandating the use of novel, high-performance avionics with one order of magnitude higher throughput than that of space-grade CPUs.

Advanced VBN involves intricate vision algorithms for estimating the relative pose between two spacecraft [91]. Relative pose refers to the position and orientation parameters defining the geometrical relationship between two objects. In lieu of visual markers and prior attitude stabilization, reliable pose estimation for uncooperative or unprepared targets will also require high-definition cameras and multiple frames per second (FPS) throughput. To execute complex algorithms on such high volumes of data, HW accelerators and HW/SW co-design are necessary. Field Programmable Gate Arrays (FPGAs) are the most promising platforms for this task, as they provide increased performance per Watt and high parallelization capabilities [2]. FPGAs are already being used in space and the market offers multiple space-qualified devices. Furthermore, following the latest trend of using Commercial Off The Shelf (COTS) components in certain commercial and/or cost-constrained missions, more powerful FPGAs should be considered by exploring COTS device families.

With these considerations and the support of the European Space Agency (ESA), we design vision algorithms and devise avionics solutions customized for active debris removal (ADR) via short-term, low Earth orbit missions [20]. We examine the approach with ENVISAT[1] and focus on VBN methods relying on passive cameras. In such settings, due to the partial shielding by Earth's magnetosphere and the short flight duration, radiation

[1] An inoperable Earth observation satellite measuring $26 \times 10 \times 5m^3$, a debris of uncertain structural integrity and tumbling rate ($\sim 2^\circ$/sec); shown in Figure 3.8.

tolerance becomes less critical, hence enabling COTS-based solutions and, in particular, modern high-density SRAM System on Chip (SoC) FPGAs. The requirements set by ESA are to define the most appropriate platform in terms of size/mass/power and develop a HW/SW system able to process at least 1 Mpixel images at 10 FPS, i.e., 5−10x faster than the expected GNC control rate.

In this paper, we explicitly address the HW/SW implementation of a vision system for tracking the relative pose of ENVISAT at distances $20-50m$ by processing monocular grayscale images. We co-design the pose estimation algorithm, the HW/SW system architecture and the VHDL accelerators, and demonstrate our proof-of-concept implementation on Xilinx Zynq-7000 SoC-FPGA. Assuming small motion between successive frames, our pose tracking algorithm evolves an estimate of target pose by first using it to render a target model and then updating it with the aid of matches established among the edges detected in the rendered depth map and the intensity edges extracted from each frame. The proposed architecture adapts to the various HW parts embedded in the SoC and exploits parallelization at multiple levels via parametric VHDL circuit design. The system is tested with a dataset comprised of synthetic videos of ENVISAT and its evaluation shows system acceleration in the area of 19x vs embedded processors, 10−14 FPS for 1024×1024 image resolution, 4.3 Watts mean power consumption on a 1U-size board, and average position estimation error in the range of 0.5% of the distance between ENVISAT and the camera.

The main contributions of the Section are a) to develop solutions for a novel application in space, b) to demonstrate the feasibility of a low-cost, low-power, low-size, high-rate, and high-definition VBN embedded system, c) to describe a custom methodology leading to efficient HW/SW implementations that exploit the capabilities of the SoC and improve the utilization of HW for use in avionics, d) to combine multiple techniques at scheduling and VHDL level (e.g., data partitioning, sliding windows, on-the-fly processing with minimal buffers, word-length optimization with fixed- and floating-point arithmetic, deep pipelining, parallelization at bit-/data-/instruction-/task-level) for efficient circuit design, e) to develop a pose estimation algorithm that can accommodate a complex object model without redesign/preprocessing and can robustly handle mismatches. Finally, we note that the proposed architecture/algorithm is adaptable and can support VBN for scenarios and targets other than ADR and ENVISAT.

3.3.2 Related work

Pose refers to the six position and orientation parameters defining the transformation between the coordinate system attached to a rigid object and a reference coordinate

system. Pose tracking exploits the temporal continuity of an object's motion to estimate its pose parameters over time. Using an object model to track pose is a widely studied topic [92]. A popular approach has been to employ an approximate object pose to predict the appearance of the model and then refine this pose using matches established between predicted and actually detected features. The most influential such tracker is RAPiD [93], one of the earliest to operate in real-time on common hardware. As a result, several of its attributes have been retained, e.g. in [94–99].

An important aspect in which techniques for pose tracking in space differ, concerns whether the tracked target is cooperative or not. A cooperative/prepared target spacecraft carries easily detectable special markers (e.g., light emitting diodes or retroreflectors), arranged in a special pattern for facilitating pose estimation by the chaser spacecraft [100]. Methods such as [101,102] that exploit structural features already present on some satellites are also classified as cooperative. Cooperative tracking is more accurate and robust but not universally applicable and might constrain the chaser to approach from angles ensuring the visibility of the employed pattern or feature.

When uncooperative pose determination is pursued, another important choice concerns the type of natural features employed for tracking. Sparse interest points extracted from local patches are very popular in terrestrial or planetary applications [103]. However, they are fairly sensitive to ambient light changes and lack of strong texture, which are particularly common occurrences in orbit environments. In contrast, image edges are defined by sharp intensity variations and are moderately robust against noise and illumination or viewpoint changes. They can also be accurately localized and are inferable even from weakly textured objects. Hence, edges are routinely preferred as the primary type of feature for tracking in space, e.g. [21, 97, 98, 104, 105]. A detailed overview of the state-of-the-art in spacecraft pose determination is in [106] and techniques for initial pose estimation are compared in [107].

Uncooperative spacecraft pose estimation errors reported in the literature differ considerably in the procedures, test data and metrics employed for evaluation. Yet, we briefly include here some figures to indicate the currently achievable performance. Positional errors under 2% of the camera-target range and rotational errors up to 3° are achieved in [21]. The GNFIR algorithm [104] yields 0.2-0.4m RMS position error for a target ranging 3-30m and attitude errors 2-4° in roll/pitch, and 0.5° in yaw. For a target ranging between 20-70m, [97] advertises around 0.5m error in position and 1° in attitude, whereas for a target range between 20-76m, [98] reports lateral (i.e., X, Y) position errors less than 1% of range, longitudinal errors around 4-5% and attitude errors between 1-5°.

Concerning the HW platforms suitable for our task, the market offers a number of space-

qualified components that we surveyed in [2]. However, the vast majority of flight heritage HW bases on radiation-hardened CPUs, such as PowerPC and LEON, which are 1–3 orders of magnitude slower than terrestrial CPU/GPUs. Hence, when targeting more than 10 FPS, complex vision algorithms for tracking the pose of satellites [97, 98], or generic objects [108], are usually implemented on GPUs. On high-end desktop CPUs, processing 1 Mpixel images for pose estimation can be as slow as 5 FPS [105]. Since desktop CPUs/GPUs are currently not a viable solution for space avionics, engineers consider DSP and FPGA platforms as accelerators [2]. Even COTS devices are being examined, especially FPGAs amenable to mitigation of the radiation effects via sophisticated design/programming. NASA supports such research, with certain teams proposing and prototyping platforms in Zynq-7000 [109, 110]. The delivered HW meets challenging constraints of spacecraft design, e.g., can even fit in CubeSat of form factor 1U ($=10\times10\times11.35\text{cm}^3$), or consume electrical power in the area of 2–5 Watts (comparable or even better than space-grade CPUs). However, these works focus more on the HW aspects and less on algorithms. Vision algorithms have been implemented on FPGAs achieving 1–20 FPS [103,111], but solve different problems (i.e., visual odometry for rover/UAV) rather than accelerating pose estimation. Note that, on rad-hard CPUs, even with the latest devices, time measurements in [2] for the algorithm of [103] show less than 1 FPS even for 0.2 Mpixel images. For the given problem and scenario, to the best of our knowledge, the current paper presents the first complete VBN solution on a compact SoC-FPGA with such high image resolutions and frame rates.

3.3.3 Exploration Methodology & System Description

The aforementioned publications and our own research indicate that conventional space-grade CPUs are not sufficiently powerful for the high-performance VBN processing required in future uncooperative/unprepared space proximity scenarios. Instead, alongside the algorithmic design, we must consider new platforms to facilitate one order of magnitude faster execution, e.g., via optimized HW accelerators. In this work, to tackle all aspects of the problem, we relied on the methodology summarized in the following interdependent steps:

1. Selection of Computational Platform
   a) high-level system architecture definition
   b) benchmarking and comparison
2. Design of Computer Vision Algorithm
   a) dataset and requirements definition
   b) algorithmic development on generic CPU

3. Development on Target HW
    a) profiling and low-level architecture design
    b) coding and optimization
    c) testing and tuning

The first step considers the plethora of factors related to the system-level design of a spacecraft's electronics: radiation-hardening, power consumption, mass/size, connectivity, dependability, market availability, re-programmability, and foremost, processing power. These factors are considered against the specifics of each space mission, prioritizing certain criteria depending on its nature/goals. For instance, for a short flight in low Earth orbit, radiation-hardening becomes less important than the real-time image processing needed for autonomous rendezvous operations. In this case, various possibilities open up for the use of COTS devices. Thus, the first step of our methodology is further divided in two parts: the high-level architectural definition of the platform and the benchmarking analysis for selecting the most suitable components according to the given criteria.

Regarding the high-level architecture, we examined multiple topologies involving single- or multi-core CPUs, interconnected with peripherals, GPU, FPGA, or DSP co-processors, altogether forming heterogeneous platforms consisting of one or more distinct chips. We performed a wide survey of the available devices in the market, both commercial and space-grade, and we reported the results in [2]. Overall, favoring miniaturization and fast data transfers among the compute nodes, we opt for single System-on-Chip solutions.

Going one step further, we performed a comparative study with extensive benchmarking to select the best type of co-processor for VBN [2]. Based on the preparatory work for step 2 of our methodology, we identified a variety of representative image processing kernels, which were implemented as common benchmarks on a number of diverse platforms, i.e., rad-hard CPU, embedded CPU, desktop CPU, mobile GPU, high-end DSP, FPGA, and desktop GPU. The execution results were used to compare, primarily, the throughput and power consumption of these platforms. The analysis gradually focused on 28nm technology nodes and is reported in detail in [2]. Overall, we concluded that assuming a budget of 10 Watts would allow for at least one order of magnitude faster execution than conventional space-grade CPUs by utilizing many-core DSPs, mobile GPUs, or FPGA devices. This gain suffices to meet the requirements of future VBN for increased image resolution and high frame-rate, e.g., 1 Mpixel at 10 FPS. Among the three device categories, FPGAs simultaneously provide the highest performance-per-Watt and the highest throughput (with the same device). More specifically, for high-definition images, HW/SW co-processing on Zynq7045 outperforms the Myriad2 DSP by $\sim$10x in speed, or

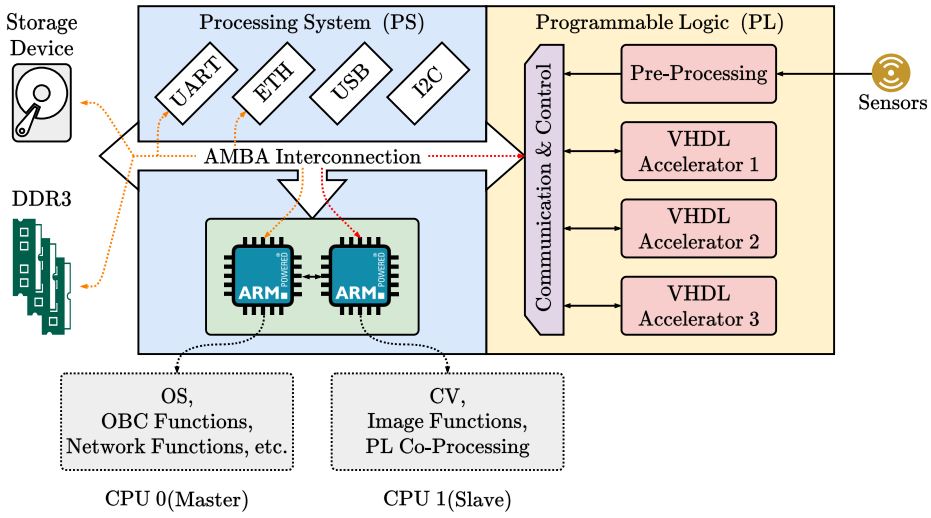


Figure 3.6: Proposed system on top of Zynq-7000 SoC-FPGA for VBN support.

the 66AK2H14 DSP by ∼10x in performance-per-Watt, or a high-end mobile GPU by at least 2x in speed and 2x in power consumption. By also considering its improved board connectivity and radiation mitigation amenability, we selected the Zynq-7000 FPGA SoC for developing our VBN system [2].

We complete here the first step of our methodology by proposing the high-level architecture depicted in Figure 3.6. Our aim is to efficiently match the various HW components already embedded in the SoC to the functionality required by VBN and/or the On Board Computer (OBC), while also exploiting the structure of Zynq-7000 to facilitate techniques for increasing the system's dependability. More specifically, following a master-slave approach, we propose isolating the 2 CPU cores and assigning almost distinct tasks and connections to each, i.e. the master core will execute generic OBC tasks to offload and communicate with the main processor (or even play the role of OBC in low-cost spacecraft), while the slave core will be dedicated to computer vision and image processing. During the GNC procedure, CPU0 will invoke CPU1 for executing the heavy task of pose estimation and will receive only a minimal set of results, e.g., those requested by the control algorithm. Furthermore, to offload the local processor (PS), CPU1 will be the only core linked via AXI4 to Zynq's PL (Figure 3.6), which will include VHDL accelerators for the most intensive kernels of pose estimation. Additionally, Zynq's PL will include a central VHDL arbiter for handling data transfers and scheduling the accelerators. The PS-PL communication will consist of two individual links, one for simple control commands (AXI4-lite) and one for transferring application data over DMA (AXI4-stream).

Regarding the off-chip connections, both cores will be able to access the external RAM, however, only the master will access the peripherals for communicating with other external components (e.g., network port, storage device). Such isolation can be imposed via the Xilinx tools at the initial system configuration, as follows: enable only a limited number of peripherals and route them explicitly to CPU0 through the 256KB-on-chip memory (not to be accessed by CPU1), allow the shared L2 to link with the RAM, disable the PS-PL interrupts from CPU0, enable the exception interrupts between CPU0 and CPU1. Finally, the camera will be connected directly to the PL pins to facilitate low-latency transfer and fast pre-processing of substantial amounts of pixel data without engaging the local CPU (PS). On top of this high-level architecture, depending on mission requirements, the developer can employ modular redundancy on the PL side, embedded watchdogs in Zynq, read-back checks, or any other reliability technique [109].

The second and third steps of our approach are explained separately in the next two sections, together with details of the methodology and actual development involved in each part.

3.3.4 Algorithm Design

The pose tracking algorithm developed in this work adopts a model-based approach to continuously update an estimate of an object's pose relative to the camera. It is inspired by the RAPiD tracker [93], which is discussed in more detail next. For each frame, RAPiD requires an object pose prediction and assumes that its difference from the true pose is small. Therefore, matches can be efficiently established using 1D local searches for image edges along directions perpendicular to predicted edges. To keep computational overhead low, matching is limited to a set of predetermined control points. Control points are sparsely selected on the tracked 3D object so that they are likely to project on high-contrast image edges. The predicted projections of control points are matched to edge pixels (edgels) and their displacements in directions perpendicular to edges are calculated. Each such displacement yields a linear constraint on the change in object pose parameters.

The standard RAPiD algorithm assumes that control points are manually sampled offline along the edges of a crude object model and in areas of rapid albedo change. Furthermore, it requires the visibility of control points to be managed externally and makes no provisions for robustness against mismatches and occlusions. To remedy these shortcomings, several improvements have been proposed, e.g. [94–96]. Nevertheless, all of them share the common drawback of requiring that the tracked object is modeled with a simple wireframe consisting of a small number of straight edges, whose projections are sampled to

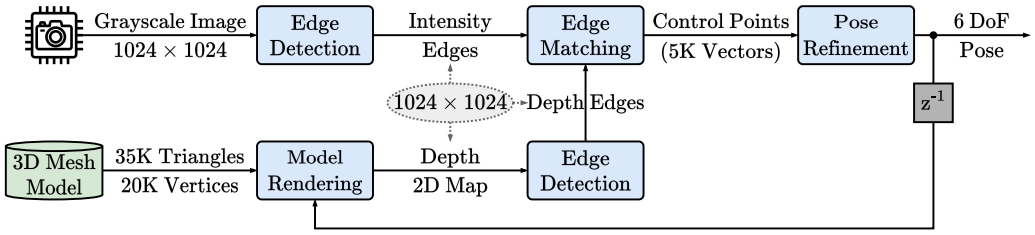


Figure 3.7: Proposed pose tracking algorithm: main functions & example dataflow

define control points. Such a choice ensures computational efficiency but imposes strong constraints on permissible object models. This is because detailed CAD object models that might be available need to be manually redesigned and only their most salient edges retained to be suitable for tracking.

Here, as shown in Figure 3.7, we propose to dynamically select control points by combining information from a depth map rendered from the object model and the edges of an input image (cf. Section 3.3.4-3.3.4). Irrespectively of the model's complexity, rendering automatically determines visibility and facilitates the generation of control points. This approach provides increased applicability and flexibility, as no constraints are imposed on the type of the employed 3D model and any tedious preprocessing for the control points is eliminated. We also ensure resilience to erroneous control points via a combination of robust regression techniques (cf. subsection 3.3.4). Our tracking algorithm assumes the initial object pose to be available and focuses on evolving it from frame to frame. The algorithm was entirely developed and refined in C. Figure 3.8 illustrates our target mesh model and visualizes sample output. The following subsections present the algorithm's key aspects, whilst a detailed description can be found in [105].

Edge detection

The Canny edge detector [112] is widely used for detecting edges in images. It combines the following sequence of steps: a) smoothing to reduce noise, b) differentiation to compute image gradients, c) non maximum suppression to remove spurious edges by retaining only maxima of the gradient magnitude in the gradient direction and d) hysteresis thresholding to track edges with a pair of thresholds on gradient magnitudes. Hysteresis identifies strong edges using a high threshold on gradient magnitude, suppresses all weak edges below a low threshold and retains the intermediate edges that are connected to strong edges. For customization to our HW implementation, we chose the 3x3 Sobel ker-

nels for smoothing and differentiation and set the hysteresis thresholds to 1.8μ and 1μ, where μ is the median of gradient magnitudes, computed from a corresponding histogram with 4K, 16-bit bins. Canny's algorithm is employed in our tracker to detect both intensity and depth edges, and hence reuse its resources in HW.

Depth map rendering

Supplied with a triangle mesh model and a relative pose, depth rendering produces an image whose pixel values are distances rather than intensities. Every pixel in the rendered depth image encodes the distance to the nearest point on the model's surface that projects on it. To save on-chip memory in our HW implementation, we represent depth images with 16 bit pixels, using an average 32-bit depth offset for the entire image and a 16 bit deviation for each depth pixel. Depth images are produced with rasterization rendering, a technique that determines which parts of a model are visible to the camera, excluding those outside the field of view or being self-occluded. Rasterization projects mesh triangles onto the image by projecting their vertices, using bounding box traversal to determine the image pixels that are inside the projected triangle and computing the distance of the 3D triangle from each such pixel. Multiple triangles projecting on the same pixel are dealt with Z-buffering which retains the projection of the triangle closest to the camera. Rasterization involves just geometry calculations (i.e., no texture mapping/shading) and uses [113] to calculate ray–triangle intersections. Automatic, off-line mesh decimation is applied to avoid very high triangle counts. Thus, for customization in HW, the triangles in our initial ENVISAT model were reduced from 120K to 35K without perceivable change in appearance and pose accuracy.

Edge matching

To obtain measurements from imagery, the object's mesh model is first rendered with the latest pose estimate to produce a depth map. Depth edges are then detected in this depth map and matched with the intensity edges found in an input image. Owing to the well-known aperture problem which states that the component of edge motion tangent to the edge itself is not observable through a finite-size aperture, the matching location of a depth edgel cannot be fully determined. Instead, only its perpendicular displacement from the corresponding depth edge is measurable. As this matching is local, it can cope with parts of the object being undetected or out of view.

The search for matching intensity edges is one-dimensional and confined in the vicinity of

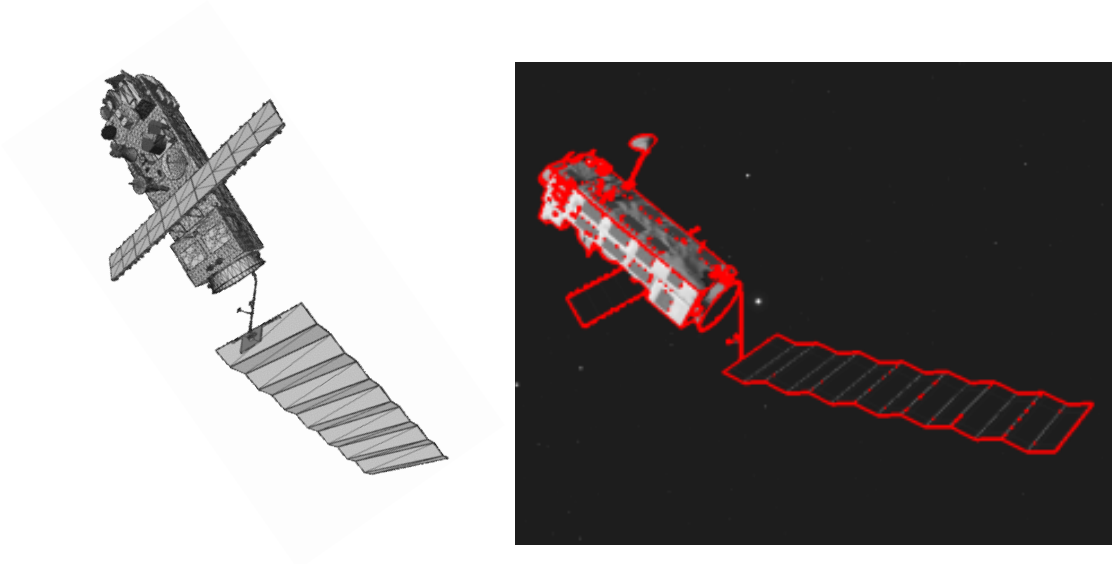


Figure 3.8: Mesh model used for tracking (left) and depth edges detected after rendering with the estimated pose, overlaid in red on an input frame (right).

the detected depth edges, along their gradient (i.e., perpendicular to edges). It proceeds until either an intensity edgel with the same orientation is found or a maximum distance has been traced. For customization, to keep complexity low, the search is performed up to a distance of 5 pixels in the horizontal, vertical, or diagonal direction that is closest to the actual depth gradient direction. Without compromising the accuracy of pose estimation, this 8-way search strategy decreases the complexity of edge matching, e.g., up to 4x compared to searching along the gradient vector, and simplifies implementation on HW. Matched edgels define the control points whose partial, normal displacements between predicted and actual edge locations are employed in Section 3.3.4 to refine object pose. The number of generated control points ranges between 500 to 5000, which represents an increase by one to two orders of magnitude compared to the typical number of control points in the original RAPiD [93].

Pose refinement

Let $\xi = (\omega_x, \omega_y, \omega_z, \Delta t_x, \Delta t_y, \Delta t_z)$ be a small rotation and translation motion applied to the tracked object. As shown in [105], the image projections $\mathbf{m}$ and $\mathbf{m}'$ of a point $\mathbf{P}$ on the object at two successive time instants are linearly related with $\mathbf{m}' = \mathbf{m} + \mathbf{A}\xi$,

where $\mathbf{A}$ is a 2×6 matrix whose elements are associated with the coordinates of $\mathbf{P}$. Combined with the perpendicular distance d between points $\mathbf{m}$ and $\mathbf{m}'$ along the unit gradient vector $\mathbf{n}$, this equation yields $\mathbf{n}^t\mathbf{A}\xi = d$. Thus, every control point provides one constraint on the infinitesimal incremental motion ξ, hence six such points suffice to estimate it. We accommodate more than six constraints in a least squares manner by minimizing

$$\hat{\xi} = \arg\min_{\xi} \sum_{i} (\mathbf{n_i}^t \mathbf{A}_i \xi - d_i)^2. \tag{3.1}$$

Solving Equation (3.1) with ordinary least squares is not recommended, as the latter is sensitive to outlying residuals caused by various types of error. To ensure that pose refinement is immune to outliers, estimation is carried out in a robust regression framework that allows invalid measurements to be identified and discarded, preventing them from corrupting the pose estimate. We achieve robustification by substituting the summation operation in Eq. (3.1) with the median:

$$\hat{\xi} = \arg\min_{\xi} \text{med}_i (\mathbf{n_i}^t \mathbf{A}_i \xi - d_i)^2. \tag{3.2}$$

This results in the Least Median of Squares (LMedS) estimator [114], which can tolerate up to 50% erroneous constraints. Unlike least squares, however, LMedS has no analytical solution. To overcome this, random sets of at least six constraints are repetitively sampled. Solving Eq. (3.1) for each such sample yields an estimate of ξ which is used to compute the median of the squared residuals. The estimate giving rise to the minimum median is retained as the minimizer of Eq. (3.2). To improve the precision of the LMedS estimate, the least squares estimate obtained from its corresponding inliers is computed. If fewer than 50% outliers are expected, LMedS can be generalized to the Least Quantile of Squares (LQS) estimator by employing a quantile other than the median; we used the 70th percentile.

For an extra level or robustness, we employ an M-estimator. M-estimators mitigate the effect of any remaining outliers by replacing the squared residuals with a symmetric function $\rho()$ that has a minimum at zero:

$$\hat{\xi} = \arg\min_{\xi} \sum_{i} \rho(\mathbf{n_i}^t \mathbf{A}_i \xi - d_i). \tag{3.3}$$

To down-weight excessively large residuals, function $\rho()$ is chosen to be increasing less steeply than quadratically. The particular M-estimator employed in this work is the L_p norm, i.e. $\rho(x) = |x|^p$ with $p = 1.5$. Eq. (3.3) is solved with the Iteratively Reweighted Least Squares (IRLS) algorithm, applied to the LMedS inliers of Eq. (3.2). The estimated

$\hat{\xi}$ is integrated with a linear Kalman filter which better predicts the pose estimate that drives rendering.

To measure pose error, we employed a single figure metric amounting to the average misalignment between the model's vertices at the true and estimated pose. More specifically, for the true $\{\mathbf{R}_g, \mathbf{t}_g\}$ and estimated $\{\mathbf{R}_e, \mathbf{t}_e\}$ pose, the alignment error is defined as

$$E = \frac{1}{N} \sum_{i=1}^{N} \|(\mathbf{R}_g x_i + \mathbf{t}_g) - (\mathbf{R}_e x_i + \mathbf{t}_e)\|, \tag{3.4}$$

where x_i denote the N vertices of the mesh model. Apart from evaluating performance, we also employed this metric to assess the effect of our customizations in SW and HW, paying attention so that they did not impact the baseline implementation error by more than 0.01% on average (wrt ENVISAT distance). We also report separate errors for the translation and rotation components. Specifically, position error is the norm of the difference between the translational components of the estimated and true pose; its ratio to the camera-target distance is the relative position error. Rotational error is the angle of the rotation that aligns the estimated with the true rotation.

3.3.5 Development on SoC-FPGA

To meet the high-performance requirements of autonomous GNC, the set of algorithms designed in the previous section must be accelerated while retaining the accuracy of pose estimation and adhering to the constraints of the underlying HW. We tackle this multi-faceted challenge with our custom HW/SW co-design methodology, which elaborates on the third step of the approach outlined in Section 3.3.3 as follows:

i) porting of SW on native CPU (Zynq's ARM)
ii) SW profiling and complexity analysis
iii) SIMD coding for native CPU (NEON)
iv) HW/SW partitioning and scheduling
v) parallel HW architecture design
vi) parametric VHDL development
vii) HW/SW integration and communication (AXI4)
viii) testing and tuning (return to step *iv* until specs are met)

The main purpose of the first two steps is to evaluate the capabilities of the embedded CPU in comparison to the computational demands of the specific algorithm. We perform a platform-dependent analysis to facilitate step *iv*, where we additionally consider the

application requirements to identify the kernels that must be accelerated on FPGA. Before HW/SW partitioning, we also explore the possibility of utilizing ARM's NEON engine for executing few lightweight functions instead of porting them on PL, i.e., we accelerate part of the SW code via SIMD instructions to exploit the full capabilities of the PS. After partitioning and scheduling of operations/communications, we design efficient parallel architectures for each HW component and use parametric VHDL to develop its circuits. Parameterization facilitates step *viii*, where we fine-tune the system to adapt specifically to the given problem, e.g., we refine datapath widths and/or parallelization factors to balance all speed−cost−accuracy trade-offs. Notice that we tune/test the fully integrated system and, moreover, we use custom datasets (described in Section 3.3.6) to emulate the given application scenario. In case of failure, i.e., violation of time or resources or accuracy constraints, our methodology tracks back to step *iv* for further adaptation and optimization.

Profiling, SIMD SW development and HW/SW Partitioning

Aiming at proper partitioning and efficient architectural design, we analyze the complexity of the algorithms with respect to a) execution time per function, b) number of generic calculations/operations (e.g., multiplications and divisions), c) type of operations and variables (e.g., float or integer values, dynamic range), d) memory footprint (including access patterns), e) communication requirements per function (I/O data), f) amenability to parallelization and/or streaming processing, and g) programming complexity (e.g., library dependencies, increased function calls). Our analysis involves both manual study and automated tools, such as the *Valgrind* profiler and the recording of CPU timestamps between major algorithmic tasks. Given the nature of our target SoC, the above criteria a+b+f point out strong candidates for FPGA implementation, criteria d+g point out strong candidates for CPU, whereas criteria c+e tip the scales when finalizing the partitions. In other words, functions with increased execution time and parallelization amenability are considered for FPGA acceleration, whereas functions with increased programming complexity and irregular accesses to large memories are better served by the flexibility of the CPU model. Functions that do not clearly fall in one of these categories are mapped to partitions according to secondary criteria, e.g., amount of data exchanged with other functions (to avoid creating PS-PL communication bottlenecks) or type of operations involved (to avoid excessive use of floating-point arithmetic in FPGA).

Table 3.5 reports succinctly the main results of our algorithmic analysis. Being representative of the target scenario, the results refer to the single-threaded C code ported on ARM Cortex-A9 @667MHz (compiled with gcc -O3) and configured with our final parameters

Table 3.5: Profiling on ARM A9 (Zynq) for 1 Mpixel images at $30m$ & $50m$

Algorithm −function	Time/Frame (cam@50m)	Time/Frame (cam@30m)	I/O (Mbit)	RAM (MB)	Other issue*
Imaging	82 msec	82 msec	16.8	1	
−fetch	56 msec	56 msec	16.8		FX, SP
−enhance	26 msec	26 msec	16.8		
Rendering	343 msec	869 msec	52	7	
−initialize	0.1 msec	0.1 msec	0.01		
−projtrian	7.5 msec	10 msec	125		FL+FX,
−gen+chk	206 msec	544 msec	4211		HP, EM
−pixdepth	51 msec	153 msec	659		
−other	78.4 msec	162 msec	≤ 1		
Edge Detect	303 msec	326 msec	12.6	18	
−gradient	210 msec	210 msec	41.9		HP+SP,
−suppress	44 msec	44 msec	67.1		FX
−hysterss	16 msec	29 msec	67.1		
−other	33 msec	43 msec	33.5		
Matching	38 msec	39 msec	17.6	2.2	SP, FX
Pose estim.	51 msec	99 msec	≤ 1.6	≤ 1	
−LQS-fit	16 msec	32 msec	≤ 1.6		PC, EM,
−LSinlier	33 msec	65 msec	≤ 1.6		FL
−other	2 msec	2 msec	≤ 1.6		
*Total**:*	1120 msec	1741 msec	8.4	44	–

*abbreviations: *FL*=floating-point arithmetic, *FX*=small fixed-point variables, *HP*=highly parallelizable, *SP*=streaming processing, *PC*=high programming complexity, *EM*=expensive math (e.g., divisions, trigonometric)

**total= imaging + rendering + 2×detection + matching + pose estimation

and processing sequences of synthetic images depicting different phases of the ENVISAT approach (at $30m$ and $50m$ away from the camera, with 1024x1024 8-bit pixels, cf. Figure 3.8 right). The 5 algorithms were analyzed to their key functions and the execution time is reported in an additive fashion. The I/O varies across rows in the table depending on the generation of internal data and the number of function calls. As shown in Table 3.5, the complexity is data-dependent for the majority of the functions and increases as the object approaches the camera. This is due to the increase in the object's apparent size, which begins by covering 6% of the image at $50m$ and grows to 20% at $30m$, i.e., the number of edgels grow from 6K to 11K to generate up to 5K matches between successive frames. To account for the worst case, we now focus on column 3 (columns 4 and 5 also refer to $30m$). Clearly, the most intensive algorithms are *rendering* and *edge detection*, which both consume almost 90% of the total time. This is mostly due to the repetitive *checking* of *generated* pixels inside the *projected triangles* of the model and the calculation of image *gradients* (including their 8-way quantized orientation). *Pose estimation* consumes up to 250msec when provided with up to 5K points (columns 2 and 3 are for 1K and 2K points, respectively). However, fine-tuning shows that $1000-1300$ is the optimal amount of control points with respect to output pose accuracy, and hence, the final cost of this function can be fixed at around $50-60$msec.

SIMD SW development: Our analysis advanced even further to examine the possibility of SIMD acceleration via the NEON engine of ARM. We focused on *imaging* and *matching*, which allow for streaming processing of small variables, but their execution time is relatively small for definitively mapping them on FPGA. We developed parallel SW code by following different approaches and we achieved an average speedup of up to 3x (vs the -O3 -ftree-vectorize executable, otherwise it appears as 8x). More specifically, we tested individually, a) NEON intrinsics, b) NEON inline assembly and prefetching mechanisms. The former approach provides a 2x speedup, whereas the latter provides up to 3x. However, due to the architectural peculiarities of NEON, more important than selecting the coding approach proved to be the parallelization technique, as explained next.

The *matching* function inputs a set of 'source' edgels and searches for each one's nearest correspondence within a set of 'target' edgels. We deduced that it is more efficient to process in parallel multiple 'source' edgels, rather than the 'target' edgels of one 'source'. That is, instead of creating a long instruction to examine in parallel all 9 candidate 'targets' of a single 'source' edgel, we create an instruction that handles 16 distinct 'source' edgels to examine one distinct 'target' for each. Hence, the entire search involves 9 such instructions per 16 'source' edgels. This technique is more efficient because, first, we utilize all 16 lanes of NEON's datapath instead of 9, second, we improve the data access patterns, and third, we manage to incorporate the sequential portion of the algorithm (determine nearest match by comparing distances) in the logic of the already 9 serialized

steps (instead of inserting extra instructions after the parallel examination of 9 candidates). The inefficient technique results in practically zero acceleration due to penalties such as ARM↔NEON register transferring, NEON pipeline hazards, or unconditional fetching of all 9 candidates without early termination. In contrast, our optimized technique decreases the execution time of *matching* to 13msec.

HW/SW Partitioning: Considering all of the above results on the PS of Zynq, we partition our algorithms by prioritizing the criterion of time: our constraint is 10 FPS or 100msec per frame. Since NEON provides acceleration in the area of 3x, *rendering* and *edge detection* remain one order of magnitude away from the requirements and must be accelerated by FPGA with a target speedup of 50−100x. Such acceleration is feasible due to the nature of both algorithms involving highly-parallel operations and/or streaming processing on pixel basis. One exception is the *hysteresis* function, which includes an inherently sequential tracing of edges; however, due to the low volume of weak edgels in our images (1−3K) and the possibility to relax the order of tracing, *hysteresis* can be pipelined without delaying the FPGA execution. Moreover, the arithmetic of *detection* is ideal for VHDL implementation as it relies on custom-length fixed-point variables and logical operations. In contrast, the arithmetic of *rendering* is very demanding and our word-length exploration showed that it is ineffective to revert from floating- to fixed-point operations in most of its functions (high dynamic range variables), except for *checking* pixel coordinates for inclusion in a triangle, where we managed to adapt to 17-bit words. As a result, *rendering* is expected to have significantly increased FPGA cost.

Further partitioning of these two algorithms is avoided due to the high I/O rates of their internal functions (Table 3.5). Isolating such a function would require PS-PL communication of several Gbps to complete on time, e.g., within a 10msec goal. The only exception is the *initialization* of rendering, which inputs limited information to calculate the camera matrix and optical center of a particular frame. This calculation bases on expensive and infrequently employed mathematical operations, which when implemented on HW increase the already huge cost of *rendering* by 33% (Section 3.3.6). Hence, in the second iteration of our methodology (Section 3.3.5, step *viii*), the *initialization* was irrevocably mapped on CPU. Lastly for *rendering* and *detection*, we note that their memory requirements are challenging the FPGA implementation and are tackled via certain optimization techniques (Section 3.3.5), such as word-length minimization of data stored on-chip, on-the-fly processing with minimal buffering, image splitting in stripes and sliding windows.

Despite the time decrease achieved via SIMD on NEON, *matching* was also mapped on FPGA in our final partitioning. The highly streaming nature of its process allows us to install it in a pipelined fashion after *detection* and practically diminish its time cost via

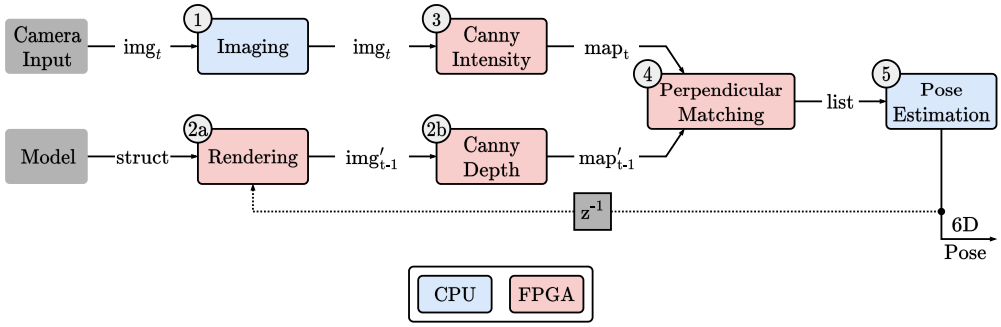
limited FPGA resources (we process the data on-the-fly while anyway being sent to the PS).

Finally, the *pose estimation* algorithm was mapped on CPU because, a) fine-tuning restricts execution to a manageable amount of time, b) it has increased programming complexity. The latter involves irregular paths in the computation graph with calls to specialized math libraries (BLAS/LAPACK) and expensive arithmetic operations (trigonometry, logarithms, etc). Such algorithms consume an excessive amount of HW resources with low HW utilization, i.e., they result in inefficient HW design as in the above case of rendering's *initialization*. We note that, in the context of this paper, we also implement *imaging* on CPU and we pipeline it with the processing of the previous image (feasible since it requires less than 100msec).

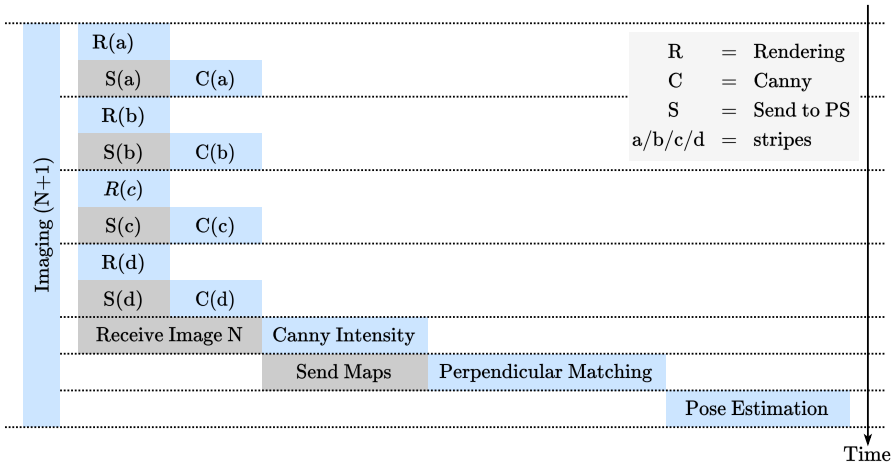
HW architecture and scheduling

Based on our analysis, we proceed to system-level HW/SW architecture design. Figure 3.9a summarizes the HW/SW partitioning, while Figure 3.9b the proposed scheduling of operations. First, each new frame is stored temporarily in the external memory of the system. Subsequently, we execute *rendering* on FPGA. To tackle the limited on-chip RAM challenge, we divide the depth image in 4 disjoint stripes of 1024×256 16-bit pixels, each, and we render the stripes sequentially by reusing the same FPGA resources. The rendering component utilizes a 1024×256 buffer to store the depths, which are updated by the algorithm in an unpredictable order preventing their continuous streaming to the remainder of the pipeline. Upon completion, the stripe is forwarded pixel-by-pixel to the *edge detection* component, which processes the depth image on-the-fly and decreases storage requirements from 16 to 5 bits per pixel. This decrease enables us to store on-chip an entire 1024×1024 edgel map and then trace its edges seamlessly via hysteresis. Notice that rendering and edge detection are executed pseudo-parallely, in 4 steps, with detection operating while the depths are being sent to the CPU (to extract information about the control points after *perpendicular matching*).

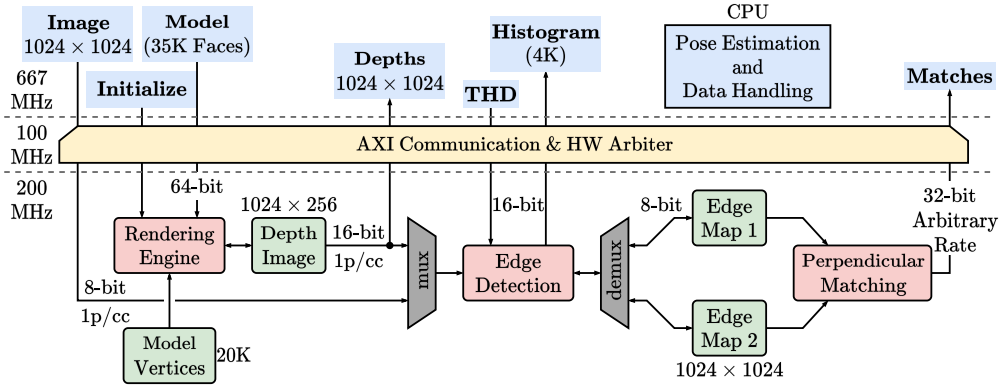
The next phase in scheduling concerns edge detection on the intensity image. The 1024×1024 8-bit pixels are sent from PS to PL, where they are forwarded directly to our *edge detection* component for on-the-fly processing. We reuse the same HW component, but switch to a distinct buffer to store the second edge map. Before and after detection, the HW component exchanges information with the CPU related to hysteresis (thresholds and histogram). Upon completion, the maps of the intensity and depth images are forwarded in raster-scan order, in parallel, to the *perpendicular matching* component, which finds their matching edgel (x, y) coordinates on-the-fly and sends them to



(a) HW/SW partition.



(b) Operations scheduling.



(c) Interconnection/dataflow of SW & HW components at PS & PL side.

Figure 3.9: Finalized HW architecture at function/data level.

the CPU. Note that, for all components, our architectural approach to perform pipelining and on-the-fly processing of the data while being transferred, with minimal buffering and balanced throughputs, leads to substantial masking of the communication time and improved HW utilization via parallelization at task-level.

The last part of scheduling concerns processing on CPU. Each (x, y) match is augmented with information regarding depth and derivatives (calculated by SW locally at (x, y)) to form a control point input to pose estimation. This augmentation on SW alleviates the need to store excessive amounts of depths and derivatives on HW until after *matching*, at the expense of a negligible time cost ('auxiliary SW', Table 3.6). In addition to pose estimation, CPU handles the Canny thresholds and the rendering initialization for the next image.

The architecture is depicted in Figure 3.9c, bottom. The buses/connections among the 3 HW accelerators facilitate the above schedule. The most notable on-chip memories are indicated by the 1 plus 3 green rounded boxes shown in Figure 3.9c. The latter 3 have similar size and store the two edge maps and the depth image stripe, as explained above. The former 1 stores the model's vertices. In particular, the model's triangles are stored at the PS and are streamed to the HW renderer upon every call, whereas the model's vertices are stored locally in ROM to support their random access pattern. The PS-PL communication is realized via the SoC's AXI interconnect. Overall, our HW-SW integration involves two clock domains on FPGA, one at 100MHz for PS-PL communication and another at 200MHz for the accelerators, a custom HW arbiter handling and interfacing the diverse ports of our HW components, and SW drivers for AXI. The following subsections describe the design and VHDL development of each HW component.

PS-PL Communication: To transfer data between the CPU and VHDL accelerators, we utilize the AXI4-lite and AXI4-stream of Zynq. In particular, we use AXI4-lite to control/synchronize our HW and SW components via small custom packets. Each packet holds a single 32-bit word and is sent through the hierarchy of AMBA interconnections of PS. The packets transfer reset & sync commands and Canny's threshold values. We prefer AXI4-lite for these small transfers, because it offers direct access to PL with limited cycles latency and an overhead of one handshake per word.

For the most important part of communication we use the AXI4-stream. Targeting high bandwidth, this link relies on Direct Memory Access (DMA) and direct connection to the memory controller of PS. Each stream consists of multiple consecutive 32-bit words sent at 100 MHz. The overhead of AXI4-stream is the additional configuration of DMA at the start of each transfer. Hence, we use AXI4-stream only for high volume data to avoid handshakes between words. At each iteration (Figure 3.9), we transfer successively,

a) 38 bytes to initialize rendering, b) 444 Kbytes for the satellite's model, c) the 2-Mbyte depth image, d) the 1-Mbyte intensity image, e) a histogram of 8 Kbytes, and f) matches of ∼10 Kbytes.

On top of the SoC parts embedded in Zynq, to facilitate the above PS-PL links, we install VHDL and SW drivers [115]. For crossing clock domains on PL and assembling the 32-bit words of AXI4, we employ 4 dual-clock FIFOs and a custom VHDL arbiter. We developed the arbiter as a Finite State Machine (FSM) to also incorporate the aforementioned schedule a–f and execute the algorithmic functions in sync with the CPU. Overall, we measured the communication at ∼3 Gbps. Such bandwidth implies 5.6 msec to transfer a 16-bit 1024×1024 depth image and is in balance with the processing throughput of our VHDL edge detection, i.e., our schedule is properly designed towards efficient masking/parallelization at task level.

Perpendicular Matching: The VHDL component is custom designed for pipelined processing of the data while being transferred to PS. It inputs the two maps in raster-scan order, 1+1 edgel per cycle, and outputs a 32-bit packet designating any occasionally detected match (ultimately, only the matches are sent to the CPU). Internally, it utilizes a serial-to-parallel (S2P) structure to extract the 9x9 region of each 'target' edgel entering the pipeline. The S2P consists of 9 FIFO RAMB18s and 9x9 registers connected linearly in a deep pipeline, as shown in the leftmost box of Figure 3.10 for the case of Canny. The 9x9 registers output a window of 81 edgels which, essentially, slides over the image map by one location per cycle. Each window is forwarded in parallel to a big multiplexer controlled by the orientation of the corresponding 'source' edgel entering the component. Notice that, for synchronization purposes, the 4-bit 'source' edgels enter in one RAMB36 FIFO properly configured by Xilinx tools for depth=5095. The multiplexer outputs 9 candidate edgels, which are compared in parallel to the 'source' edgel. The comparators are connected via a multiplexing structure giving precedence to the middle candidates. In case of equal orientation at $distance \in [-4, 4]$, the component outputs $\langle x, y, distance \rangle$, where x, y are being tracked via counters synchronized to the input rate. Therefore, with minimal on-chip memory and negligible logic resources (see Table 3.6 of the evaluation section), the proposed unit completes perpendicular matching, on-the-fly, with exactly the same accuracy as the CPU but eliminating the SW time cost.

VHDL development: Edge Detection

To detect edges in both the intensity and depth images, we design on HW the Canny algorithm customized in Section 3.3.4. We implement on FPGA one module config-

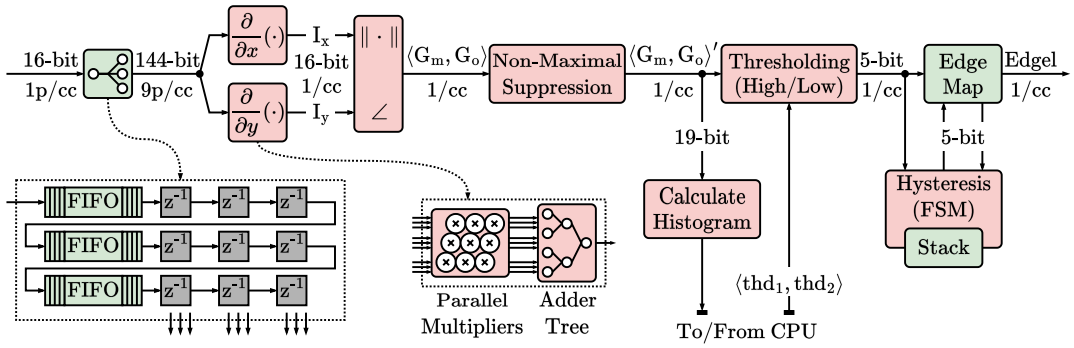


Figure 3.10: Proposed HW architecture of Canny, with deep pipelining, on-the-fly processing, limited use of buffers, and recursive edge tracing with a small stack.

ured for 16-bit 1024×1024 images. The module inputs either 16-bit depth values, or 8-bit pixels from the camera with zero-bit-padding. Processing is performed in an almost streaming fashion with a window sliding downwards the image and alleviating the need for more sophisticated data handling. In both cases, it outputs a 1024x1024 4-bit map, where each pixel denotes the existence of an edgel and its 8-way quantized orientation.

From a HW design perspective, the algorithm is divided in three parts: a) the fully streaming calculation of gradients and non-maximal suppression, b) the statistics on gradients for threshold calculation, which implies a second iteration/pass of the image, i.e., local storage that would cancel the benefits of the above streaming operation, and c) the hysteresis requiring a recursive computation on the initial edgel map. Each part is tackled individually as explained next.

First, during the image transfer to PL, the gradients are computed on-the-fly inside a deep pipeline with a throughput of one pixel/gradient per cycle. As shown in the leftmost part of Figure 3.10, the pixels enter in burst mode, in raster-scan order, and are forwarded to two distinct components for calculating in parallel the I_x and I_y partial image derivatives. Based on Sobel's 3×3 operator, these two components implement the same 2D convolution, but with their kernel rotated by 90°. To facilitate streaming processing and avoid a second pass of the image, we disregard the separability of the filter; instead, we employ a serial-to-parallel buffer to output one 3×3 pixel window per cycle (the S2P buffer is shared between I_x and I_y, Figure 3.10, left). With each new pixel entering Canny, the 3×3 window slides in raster-scan order and allows full parallelization of the 3×3 kernel multiplication. The 2D multiplication is implemented via hardwired, parallel, bit-shift operations and the results are accumulated by a 4-stage

adder tree to generate one $\langle I_x, I_y \rangle$ vector per cycle. This 32-bit vector is forwarded to the next component, which computes the 16-bit magnitude and 3-bit orientation of each gradient. More specifically, $G_m = \sqrt{I_x^2 + I_y^2}$ is calculated by a pair of multipliers and a square-root unit, altogether forming a 24-stage pipeline with continuous flow. In parallel, $G_o \in [0, 7]$ is calculated by comparing the signed tangent of the input vector to the tangent of $3\pi/8$ and $\pi/8$. For optimization purposes, we replace the I_y/I_x divider with constant multipliers (for $I_x \cdot \tan 3\pi/8$ and $I_x \cdot \tan \pi/8$) integrated in an 8-stage pipeline together with 6 comparators and extra registers synchronizing G_o to G_m. Subsequently, each G_m value is conditionally suppressed by examining its 2 neighbors in the direction of G_o. In particular, a second serial-to-parallel buffer extracts the 3×3 region of each gradient (1 per cycle). All 9 values are forwarded in parallel to a multiplexer controlled by G_o, which in turn feeds 3 comparators determining whether the current G_m is a local maximum. Afterwards, in the $\langle G_m, G_o \rangle'$ stream (Figure 3.10), we compare the unsuppressed G_m to two thresholds for distinguishing between 'weak' and 'strong' edgels. The thresholding operation also decreases storage requirements to 5 bits per pixel. Streaming terminates at the RAM storing the edge map, which is initially filled in 1024×1024 cycles.

The second part of Canny concerns the once-per-image updating of the aforementioned hysteresis thresholds. The calculation uses a histogram of gradients of the current image. To avoid storing all gradients on-chip and to decrease Canny's execution time in half, we exploit the temporal redundancy of successive frames: we calculate on-the-fly the threshold pair for the current image, but use it for the next image. Due to frame affinity, our tests show that successive threshold values differ by ∼10% and that our modification affects only 3% of each edgel set, whereas the overall result of pose estimation remains practically the same. As shown in Figure 3.10, a histogram component monitors the gradients while flowing in the main pipeline. By using a read-write loop over a small memory (2 RAMBs), including registers and adders, we group the 16-bit values in 4K bins and also calculate their median. Results are sent to the CPU, which refines and handles the thresholds throughout the sequence with practically zero cost.

The third part of Canny concerns edge tracing on the initial map (after thresholding, Figure 3.10). Our analysis of the ENVISAT images shows a manageable amount of initial 'strong' edgels, e.g., 10K, with limited fetches of adjacent edgels, e.g., 100K. Therefore, to avoid scanning the entire map, we prefer storing all initial strong edgels in a local LIFO memory and then, recursively, fetching their neighbors to update the connected weak edgels. The former operation is performed on-the-fly, during the storage of the initial map, by monitoring the main flow to the map's RAM. The latter is performed after the initial map is filled, as follows. An FSM continuously pops edgels from the local stack (1 per 8 cycles), fetches sequentially their 8 neighbors from the map's RAM, and checks

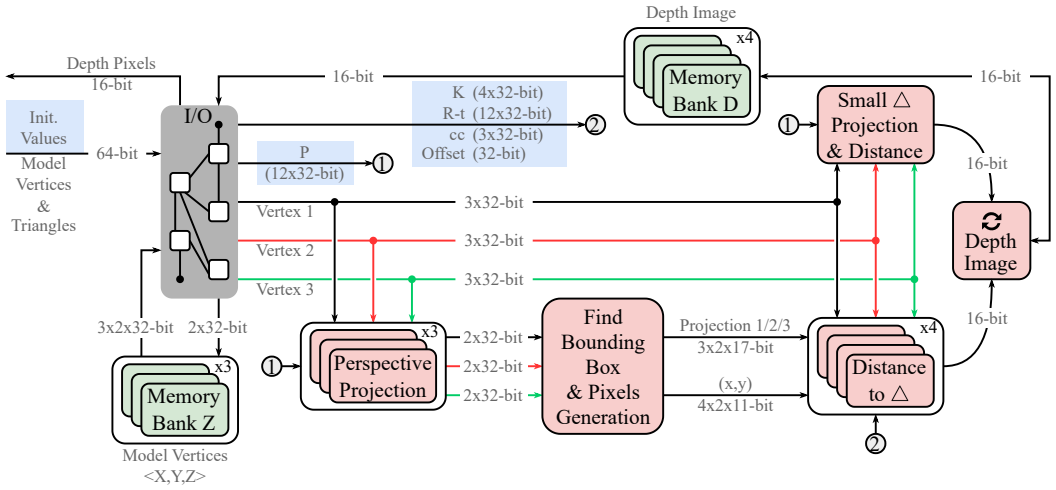


Figure 3.11: Proposed HW architecture of rendering, with 4x parallelization at pixel level, floating-point & deeply-pipelined components, and parallel memories.

their strength without stalling the pop procedure. When a 'weak' edgel is fetched, it is pushed to the LIFO and it is marked as 'strong' in the map's RAM. Notice that the push-pop order in the local stack is not strict (all adjacent edgels are eventually fetched) and allows continuous flow in a fully utilized pipeline. Edge tracing completes $10-20\times$ faster than the initial map creation.

VHDL development: Rendering

To generate depth images from a 3D model with the last estimated pose, we design via parametric VHDL the renderer customized in Section 3.3.4. For sufficient acceleration and in accordance to our analysis, we design our engine to project up to 1 triangle per cycle and render up to 4 pixels per cycle. To support such throughput, we rely on memory interleaving, deep pipelining at vertex/pixel level, and module parallelization. Moreover, due to the dynamic change of workload per model triangle, certain parts of the pipeline are designed to support arbitrary stalling. The proposed architecture (Figure 3.11) consists of five parts: a) the *I/O controller*, b) the *perspective projection* of a triangle, c) *generation of pixels* inside the box bounding the triangle, d) *depth computation* of each pixel in the triangle, and e) *updating* of the depth image.

The *I/O controller* receives from CPU and stores all model vertices at the beginning of

a sequence (the model can change during the chaser's approach). Each $\langle X, Y, Z \rangle$ vertex is stored in three distinct dual-port RAM memories allowing the parallel fetch and fast projection of a triangle. With each new image, the controller receives the initialization values (intrinsic calibration matrix and pose $\mathbf{R}$, $\mathbf{t}$). With each new stripe, the controller receives all model triangles in a stream, up to one triangle per cycle, and forwards them in the processing pipeline. Note that, a) it is impractical to distinguish and send only a subset of triangles per stripe, and b) the overhead of handling all triangles with each new stripe is negligible, because the majority of the complexity lies in the processing of pixels and triangles projecting outside the current stripe are filtered out early in our pipeline. The input stream is paused depending on each triangle's size (pixels to be rendered) to allow time for pixel *generation* to complete; stalling is achieved by using input buffers and carefully interconnected clock-enable signals throughout the pipeline.

The *perspective projection* of each triangle (Figure 3.11) utilizes single-precision floating-point units, i.e., 11 multipliers and 9 adders and 1 reciprocal unit, which are all integrated in a 67-stage pipeline for fine-grained register-transfer level operations. We employ three such components to project in parallel all 3 vertices of a triangle. Hence, the component can project up to one triangle per cycle. In addition, we develop a fourth pipeline to project the geometric center of a triangle when it is too small to be reliably intersected by the casted ray. In practice, when not stalled, due to the input rate from AXI4 and the access time of the model's RAM, the processing rate increases up to 1 triangle projection per 2 cycles.

The pixel *generation* component (Figure 3.11) inputs the projected triangle and outputs all pixels residing inside its bounding box. It is designed with 17-bit fixed-point arithmetic. First, with a small pipeline of comparators, it computes the projected triangle's minimum and maximum image coordinates to examine whether its bounding box resides partially in the current image stripe. Second, it generates 4 neighboring pixels in parallel, either in a 4×1 or in 2×2 vector of integer $\langle x, y \rangle$ pairs. We dynamically select between the row or square format depending on the size of the bounding box: if *width*>*height* and *pixels*>4, then we use the 4×1 rows, otherwise, we use 2×2 squares. Accordingly, by initializing two 11-bit counters, multiple quadruplets are generated continuously, one per cycle, until the entire box is scanned (the input of a new triangle pauses here, if necessary). Indicatively, for ENVISAT at $30m$, a bounding box can include up to 22K pixels, although half of the bounding boxes include up to 40 pixels.

The four generated pixels are forwarded to four modules computing their distance from the camera, in parallel (Figure 3.11). Each module utilizes 6 fixed point multipliers, 14 fixed point subtractors, 2+1 fixed-float converters, 42 floating point multipliers, 38 floating point adders/subtractors, 2 floating point dividers, 1 floating point square root

unit, and 1 floating point reciprocal unit. All these units are integrated in a 307-stage pipeline. Functionally, it involves a check on a pixel's validity (whether it is within the projected triangle) and a series of algebraic operations on vectors followed by an Euclidean distance calculation to estimate its depth [113]. Each of the four components outputs the depth by converting to a 16-bit fixed-point value, and hence, decreasing storage requirements in half. In addition to the four calculators (Figure 3.11), we employ a fifth component to handle small triangles that cannot be intersected by the casted ray. Similarly, this component utilizes 23 floating-point units integrated in a 124-stage pipeline.

The final module *updates* the depth image (Z-buffer) via 16-bit fixed-point arithmetic and a parallel memory organization that allows conflict-free access to any pixel quadruplet. The depth memory utilizes 4 banks $\{A, B, C, D\}$ interleaved in the x direction and shifted linearly by 2 in the y direction of the image (i.e., $ABCDABCD\ldots$ for the first image row, $CDABCDAB\ldots$ for the second row, etc.), so that any 4 neighboring pixels are mapped to distinct banks. Pixel located at (x, y) is stored in $\text{bank}(x, y) = ((y \bmod 2)\cdot 2 + x) \bmod 4$ in the $\text{address}(x, y) = y256{+}x \text{ div } 4$. At each cycle, we compare the 4 stored values to the 4 newly calculated distances, and we update the RAM contents with the smaller values. The *update* component is pipelined to operate seamlessly even for successive reads/writes to the same RAM address. Upon completion of the algorithm, the stripe is output in raster-scan order and the I/O controller clears the 4-bank memory.

3.3.6 System Evaluation

We implemented the entire vision-based navigation system on the Xilinx Zynq 7Z100 FPGA hosted on a Mini Module Plus board. We employ a pair of synthetic sequences which realistically simulate the motion of ENVISAT and the solar illumination. They have a length of 1000 frames each, a resolution of 1024x1024 8-bit pixels, 40° FoV and known ground truth pose. In the first sequence, ENVISAT rotates out of plane and remains approximately at about 50 meters from the camera. In the second, the camera is initially at $30m$ away from ENVISAT and gradually approaches to $20m$. We note that, since the camera specifics/interface are not the primary concern of this work, we assume pre-stored images being fetched and pipelined by CPU0 in less than 82msec each (Section 3.3.5). We also note that, since this work concerns primarily the high-performance potential of the proposed system, which is examined prior to verifying its mission-dependent reliability, we omit demonstrating here any additional radiation mitigation technique for Zynq [109].

Our evaluation begins with the accuracy of the HW/SW implementation. As expected,

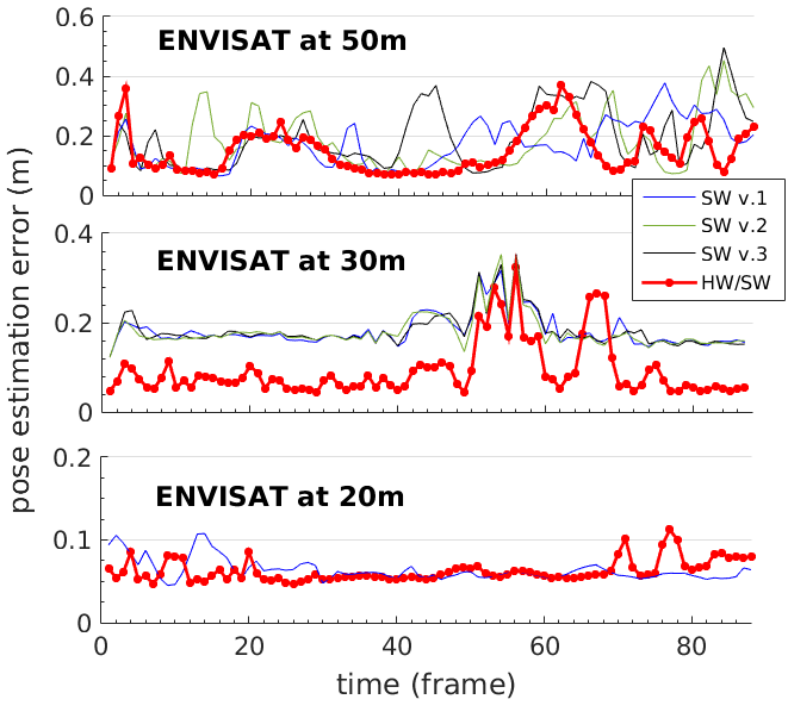


Figure 3.12: Pose estimation (alignment error) of our HW/SW system for three ENVISAT distances. The HW/SW accuracy is practically equal to that of SW (cf. 3 similar versions plotted with thin lines). In all 3 cases, error $< 40cm$.

due to HW optimizations such as floating- to fixed-point arithmetic modifications and word-length minimization, the HW/SW output deviates slightly from the initial all-SW output. In particular, Canny on HW generates 96−98% same sets of edgels with Canny on SW, with their equality increasing to 99% when we consider only the strongest edgels. Therefore, since usually the most prominent edgels are matched between successive frames, we get up to 99.9% same matches by the HW and SW versions (notice that the perpendicular matching function on HW is fully equivalent to SW). The HW rendering outputs a projected model, the shape of which is 99.7% equal to the SW output with 96% of the pixels having less than 1mm difference in depth (and 98% having less than 10mm). In total, when we combine all three HW modules, we derive sets of control points on ENVISAT, which are 90% identical to the sets derived by the SW version. Therefore, as Figure 3.12 shows qualitatively for three representative segments of the sequences, the pose estimates of the entire HW/SW system are practically equal to those of the original SW. Quantitatively, when taking into account our entire dataset, the all-SW and the HW/SW results have the same error in terms of *mean, std* and *range*. More specifically, when considering the *alignment* metric of Eq. (3.4), we obtain a mean error of $17cm$ when ENVISAT is $50m$ away from the camera, which decreases to $7cm$ at $20m$ distance. This alignment error comprises $2-50cm$ position error and $0.5^\circ - 3^\circ$ rotation error with the highest deviations presented at far distances. Overall, the alignment error of our HW/SW system is around 0.5% and less than the desired 1% of the camera–ENVISAT distance. With the additional metrics described in Section 3.3.4, our pose tracker has a mean relative position error of 0.53% and a mean rotation error of 0.51°. We note that at very short ranges (i.e., 10m or less), and due to most of ENVISAT and its edges being outside the camera field of view, the proposed algorithm might need to be replaced by one relying on more abundant visual features, possibly from a different sensor modality such as LIDAR.

Table 3.6 reports the FPGA resources required by each component and the utilization of Zynq XC7X100. In terms of logic, rendering proves to be the most expensive algorithm, by one order of magnitude, with 822 DSPs and 82K LUTs devoted to calculating the *distance-to-triangle* of the ENVISAT model. This is mainly due to the floating-point arithmetic and the 4x parallelization employed at pixel level. Edge detection with fixed-point arithmetic is far less demanding, with half of its logic resources devoted to the calculation of *gradients*. Memory is more reasonably distributed between rendering and edge detection, partly due to our image partitioning technique (Section 3.3.5), which also tackles the RAMB size problem and enables us to fit all functions in a single FPGA. Out of the 347 RAMBs in edge detection, 160 are devoted to the edge map of the intensity and 160 to the map of the depth image (both 1024×1024 5-bit), whereas 20 are for the stack memory of hysteresis. Out of the 224 RAMBs in rendering, 128 are devoted to storing a 1024×256 16-bit stripe of the depth image, whereas 96 are for storing the vertices of ENVISAT's model (20K vertices for 35K triangles). The remaining memory utilization

Table 3.6: Final System: Resources on SoC-FPGA XC7X100IFFG900-2L

Component	**LUT**	**DFF**	**DSP**	**RAMB**	**Time (ms)**
Rendering	93383	148071	966	224	11±4
Edge Detection	2948	3174	4	346.5	6
Perp. Matching	298	389	-	5.5	5
Commun.+Misc.	2895	3777	-	6	3 Gbps
Pose Estimation	ARM core #1				55±10
Auxiliary SW	ARM core #1				0.5
Total	99524 (36%)	155411 (28%)	970 (48%)	582 (77%)	83±14 (∼12 fps)

on FPGA was significantly decreased via on-the-fly processing of nodes/pixels/edgels and avoiding intermediate buffering. Perpendicular matching has negligible cost mainly due to the relatively small parallelization and straightforward operations involved. Communication is more costly due to the complexity of AXI4-stream (3−4x vs lite), e.g., for DMA and multiplexing of read/write channels. We mention here that in rendering, the frame initialization was implemented with 31K LUTs and 312 DSPs to gain only 0.1 msec, and hence, was omitted from the PL (Section 3.3.5). We also note that, when considering smaller FPGAs, i.e. the Zynq XC7Z045, the RAMBs and DSPs are over-utilized by ∼7%, however, the 55% free CLBs make plausible the tuning/rearrangement of resources in order to fit in a single XC7Z045 device.

To analyze execution time, the last column of Table 3.6 reports the milliseconds spent per function for a 1 Mpixel image. The FPGA components operate at 200MHz, communication at 100MHz, whereas ARM operates at 667MHz. Depending on the apparent size and the number of control points detected on ENVISAT, we get the reported fluctuation of time during rendering and pose estimation. In total, considering that edge detection is executed twice, the average time to complete an entire frame is 83msec. The average time increases from 71msec at $50m$ away from ENVISAT to 90msec at $20m$ (in terms of complexity, the amount of computations increases by up to 87% during the ENVISAT's approach). Therefore, our system achieves a processing rate of 10−14 FPS. We note that it is straightforward to increase this rate by another 2−3 FPS by decreasing SW time, i.e., by configuring the CPU to operate at 800MHz and/or by further limiting the number

of control points in pose estimation (now 1300 per frame, could be suppressed down to 1000). We also note that, when assuming higher resolution images input to our HW/SW system, the SW execution time would remain roughly the same (with the number of control points at 1300), whereas the HW time would increase almost proportionally to the number of pixels. Hence, our system would achieve a processing rate of more than 5 FPS even with 4 Mpixel images.

Further analysis for the 1 Mpixel images shows that, despite having mapped ∼96% of the computation on FPGA, due to the relatively slow speed of ARM, time is almost fairly distributed between PS and PL with a 2:1 ratio. The average pipeline utilization on FPGA is 2/3 when PL is engaged in processing. Owing to the efficient scheduling of FPGA operations and on-the-fly processing of data while being transferred from/to the CPU, we achieve 92% masking of the PS-PL communication time. As a result, we decrease the communication overhead to only 1msec, i.e., almost 1% of the total time, and we gain up to 11msec increasing our FPS rate by one extra frame. In total, when comparing to the all-SW execution on Zynq's ARM, we achieve a speedup factor of up to 19x for the entire HW/SW system. Individually for each kernel, we accelerated rendering by 50−62x and edge detection by 54x, or even 75x when assuming the maximum clock frequency of 278 MHz. Such acceleration factors are critical and, as shown here, can be achieved via parallelization on FPGA when targeting one order of magnitude faster execution than the 1 Mpixel frame-per-second capabilities of latest rad-hard CPUs [2].

To evaluate energy consumption, we combined the power estimator tool of Xilinx and real HW measurements via the UCD9248PFC controller chip hosted on a Zynq ZC702 board. When PS operates at 667MHz and PL at 200MHz, the ARM processor consumes 1.6W, whereas the static power on xc7z100's PL is 0.4W. The dynamic power on PL ranges between 6−7W. Therefore, given the PS-PL utilization time of our algorithms, the total power consumption ranges between 2−9W on MMP and is averaged over multiple frames at 4.3 Watts. Such power and size, i.e., only $5.7 \times 10.2\ cm^2$ and 65 gr, make the proposed embedded system highly suitable for many cost-effective LEO applications in space.

Comparison to previous works Compared to similar works in the literature, our VBN provides state-of-the-art HW *efficiency* (i.e., throughput per resources) at component level, high accuracy at algorithmic level, and improved miniaturization/readiness at system level. We resort here to this distinction due to the practical difficulties of performing fair/complete comparisons between diverse and scarce publications about the examined VBN scenario. First, at component level, even with recursive hysteresis, our 8-bit Canny utilizes half FPGA resources compared to [116] (it achieves 4x higher throughput, which however is not necessary in our case). Even in comparison with FPGA Canny com-

ponents specifically targeting low-cost/power implementation such as [117], we achieve almost the same HW efficiency (20% faster execution with 40% extra LUTs, when normalized to same frequency and image size and Xilinx technology). Similar conclusions can be drawn for our VHDL renderer, despite performing more coarse comparisons due to the differences in algorithms and object models. That is, by extrapolating to the same clock frequency and number of triangles/data, we get almost equal throughput per LUTs with [118] and [119]. Second, at algorithmic level, the results presented above show that our 0.53% relative position and 0.51° rotation errors are similar or lower than the 1%-5% and 1-5° errors reported in the literature (cf. Sectiopn 3.3.2). Moreover, our pose tracking algorithm was shown in [105] to perform much better compared to two state of the art trackers. Third, at system level, in contrast to [102], we estimate the full 6D pose of the satellite, we accelerate a much larger part of the computation on FPGA, and we deliver a self-contained embedded system on a compact board, which is more representative of space applications. This self-containment advantage is also evident when compared with the VIBANASS system [120], which requires an additional ground-segment equipment to perform image processing with slightly higher error (i.e., $\sim$1%). Likewise, compared to the ADR demonstration mission of [121] that proclaims an error of $\sim$0.4% at 25m, we achieve similar accuracy in 6D pose estimation with edge-based tracking, but we already accelerate the algorithm on FPGA and provide a high-performance, space-deployable VBN engine/solution on an embedded SoC ([121] does not perform image processing on-board the satellite).

3.4 Conclusions

In this chapter, we presented the systematic design of two efficient HW/SW co-designed embedded systems for image/video applications and deployed them on modern SoC-FPGA devices. The Image Registration processing pipeline, was successfully mapped to one of the smallest Zynq-7000 SoC-FPGA devices, while incorporating single-precision floating-point arithmetic. Moreover, the quality of the proposed design was experimentally verified against pure software implementations showing high performance gains up to 67x speedup against embedded CPUs, while preserving algorithmic accuracy. In the case of the Vision-Based Navigation (VBN) processing pipeline, targeting future ADR mission requirements, in a top-down co-design approach, we have proposed a high-level architecture exploiting the structure of Zynq-7000, we refined our computer vision algorithm for tracking the pose of ENVISAT, and we devised a HW/SW implementation methodology to customize and efficiently map all functions to the SoC FPGA. We tackled the computational challenges via a combination of techniques at the architecture and VHDL level. The resulting VBN system achieves a throughput of $10-14$ FPS for 1 Mpixel images,

with only 4.3 Watts mean power, tracking its target in real-time and with mean error 0.5% of the distance between the camera and the target.

Chapter 4

Baseband Processing in RFSoC FPGA for 5G/B5G Mobile Networks

The evolution of mobile networks (5G/B5G) relies on hybrid optical-wireless transport architectures, enabling flexible and scalable interconnection between Baseband Units (BBUs) and Remote Radio Heads (RRHs) with heterogeneous interfaces. These networks offer substantial improvements in data rates, device capabilities, and data volumes compared to previous generations. However, specific technology choices are still under consideration, with new suggestions continuously emerging. To effectively support 5G/B5G deployment, researchers and engineers are turning their attention to FPGA-based Baseband architectures. These cost-efficient and re-programmable solutions address emerging requirements and modifications. In this chapter we introduce a baseband DSP engine designed for converting on-the-fly real-time Ethernet traffic to OFDM waveform for Analog–Intermediate Frequency over Fiber (A-IFoF) transmission. We deploy the developed Tx/Rx-OFDM system on an Xilinx RFSoC ZCU111 board and evaluate its performance, resource utilization and efficiency by integrating it on real-world fiber-wireless infrastructure. This chapter is based on our publications in [23, 24, 30, 122–127] and is the collaborative effort of the Microprocessors and Digital Systems Lab and Photonics Communications Research Laboratory from National Technical University of Athens, Digital Systems Team of the Electronics Lab from National and Kapodistrian University of Athens and Wireless and Photonic Systems and Networks research group from Aristotle University of Thessaloniki. The author of this Ph.D. thesis developed hardware architectures for several of the Tx/Rx-OFDM blocks and performed the integration, testing, and debugging on the RFSoC FPGA device. Additionally, the author was part of the team responsible for integrating, testing, and debugging the RFSoC FPGA-based DSP engine with a fiber-wireless infrastructure to provide real-time mobile services.

4.1 Introduction

Even though the number of 5G technology demonstrations that are presented by the system vendors and telecom operators is increasing [128], the transition from 4G-Long Term Evolution (LTE) is long process, while the final 5G specifications are not defined yet. This transition is enforced by the network demands of increasingly popular emerging technologies such as virtual/augmented reality and 4K video, low latency, reliability and device/network energy efficiency, will be addressed by 5G networks by exploiting existing technologies that include millimeter waves (mmWave), massive multiple input – multiple output (MIMO), beamforming [129], [130] and multi-carrier modulation formats [131]. The prevailing network architectures are the centralized and ultra-dense network topologies at the mobile fronthaul (MFH), providing higher data rates in a new radio-band (30-300GHz) [132] through the use of mmWave (radio frequency) RF carriers. However, the high attenuation of mmWave transmission, due to atmospheric absorption, will anchor optical fiber as the main transmission link between the BaseBand Unit (BBU) and the Remote Radio Head (RRH) side, bringing Radio-over-Fiber (RoF) technology to the core of future 5G implementations [132].

The MFH with digital fiber-optic interfaces, such as the Common Public Radio Interface (CPRI), with high data rates cannot be supported by the CPRI-formatted frame specifications [133]. Moreover, the latency of RRH processing [134] makes Analog RoF (A-RoF) a strong 5G candidate, without digitization penalty but with high throughput eliminating the optical hardware demands. BBUs involve computationally demanding DSP functions and have to be re-programmable because the standards are still open. Moreover, the transition from Distributed-RAN (D-RAN) topologies towards Cloud-RAN (C-RAN) architectures, where a pool of BBUs accommodate the needs of several RRHs, will use re-configurable processors and FPGAs [135]. Besides the cloudification of the mobile fronthaul, fully elastic optical transceivers will operate in dual mode supporting both D-RoF and A-RoF transport, and allow for distance-adaptive modulation [136]. These elastic FPGA-based BBUs facilitate also the transition towards Ethernet-based fronthaul [137]. Through this Radio-over-Ethernet (RoE) approach, FPGAs can offer solutions that cope with the inherent jitters of the CPRI traffic over standard Ethernet [138].

In the current Section we present the design a proof-of-concept datapath to transmit and receive data according to the aforementioned ideas of C-RAN and A-RoF, with potential use even in future MEC scenarios. More specifically, we consider here data streams e.g., via Ethernet, as input to our system consisting of: a) CPU/smartNIC handling MAC layer tasks, b) FPGA handling the digital baseband processing, c) DAC/ADC for signal conversion to an IF, d) fiber optics for ARoF transmission/reception to the

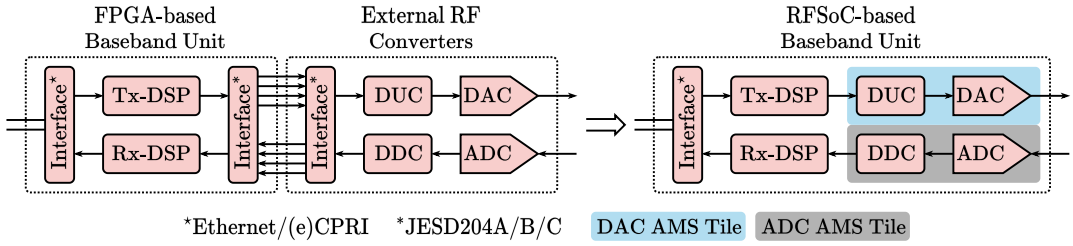


Figure 4.1: Evolution of FPGA-based BBUs.

RRH, and e) mmWave RRH for establishing new radio-band links. Aiming at providing an effective solution to the BBU, this paper focuses on the design of a SoC-FPGA based architecture realizing parts (b) and (c). To this end, we capitalize on state-of-the-art chips integrating processors, FPGA resources, DAC/ADC with digital-up/down-conversion and hard-IPs for the FEC functions, to provide a scalable embedded solution with low cost/power but real-time performance. We research for developing the BBU DSP functions and their fine tuning by the use of simulations and implement these on Xilinx RFSoC FPGA of the ZCU111 board [22]. The proposed custom VHDL circuits are based on pipelining and parallelization techniques to achieve the required real-time performance.

4.2 Rationale of selecting RFSoC FPGA

Until the widespread adoption of 5G standard, the devices must have backward compatibility with the already existing standards (e.g. 4G LTE). FPGAs are attractive for implementing BBUs that support a variety of standards. Until recently FPGAs were used only for basedband signal processing and communication with the core network. Xilinx introduced the Zynq Ultrascale+ RFSoC family of devices, that integrates on the same die an heterogeneous processing platform (CPUs, FPGA fabric, etc.) with direct RF-sampling converters (Figure 4.1). The integration of DACs/ADCs on the same die tackles many design challenges. The interface between the FPGA and the converters becomes significantly simpler. Previous systems had to implement specific protocols for communication (e.g JESD204), that led to significant portion of the FPGA resources used for this purpose. Moreover, the external I/O interfaces consumed significant power to operate. Now, RFSoC devices [139] offer 1) reduced power consumption, 2) reduced clock/data routing complexity on PCB 3) simplified synchronization procedures between FPGAs and RF-converters and 4) the full flexibility of traditional FPGAs.

Integrated DACs/ADCs

The RFSoC device integrates Analog-Mixed-Signal (AMS) components, i.e., on the same die, it hosts up to 16 Digital-to-Analog (DAC) and 16 Analog-to-Digital (ADC) converters depending on the device generation. The converters have analog bandwidth in the order of GHz. Moreover, each converter has each own dedicated digital datapath that implements some DSP functionalities previously implemented on the FPGA fabric. These functionalities include interpolation/decimation filters, digital mixers to up-/down-convert from/to baseband, signal compensation blocks, as well as signal detection capabilities. Moreover, the programmability of the AMS blocks offers 1) different number of parallel samples that ease the operating frequency on the FPGA, 2) ability to process real or complex signals and 3) multi-band operation.

Ethernet Connectivity

RFSoC has increased capabilities in terms of Gigabit Ethernet connectivity enabling the Ethernet-to-RF applications. On the PS side the designer can opt to use the equipment of the single Quad of PS-GTR transceivers as well as the four triple speed Ethernet MACs that provide support for IEEE Std 802.3 with 10/100/1000Mbps rates and a set of capabilities enabling the VLAN tagging, Precision Time Protocol (PTP) support, etc. More importantly, on the PL side, each of the GTY transceivers can support up to 25G Ethernet with the MAC and PCS implemented on the FPGA fabric. 100G Ethernet can also be supported with the Integrated Block for 100G Ethernet that includes both MAC and PCS logic. The above features/equipment/IPs can enable applications where connectivity to RRHs or network backhaul is required.

Embedded Components (Hard-IPs)

Next to the FPGA fabric, RFSoC integrates a quad-core Cortex-A53 APU and a dual-core Cortex-R5 RPU. Utilizing the embedded CPUs allow real-time monitoring and control of the BBU operation. Automatic calibration of the DSP functions can be accomplished through data monitoring and computation of link statistics (e.g EVM, noise variance, etc.). The user will also be able to connect remotely to the device to perform parameterization of the BBU, debugging in case of error reporting, perform maintenance and update supported functionalities. A single CPU core is sufficient to perform these tasks. RFSoC integrates Soft-Decision FEC (SD-FEC) hard-IP blocks. The FEC algorithms are very computationally intensive and would require huge amounts of resources (BRAMs, DSPs)

on the programmable FPGA fabric (PL), especially for decoding, as well as increased dynamic power consumption. Certain RFSoC devices integrate up to 8 such SD-FEC blocks with their own dedicated clock network (not limited by the remaining DSP chain). These blocks are configurable and can support different coding schemes, such as Turbo and LDPC. Pre-configured standards are supported (5G NR, LTE, 802.11, DOCSIS 3.1), as well as custom LDPC codes can be used.

4.3 Physical Layer DSP for OFDM Transmission

4.3.1 Centralized DSP engine

Figure 4.2 shows the single band Tx and Rx DSP block chain designed to support the centralized, Analog RoF based network architecture. Multiplexing of multiple bands will be performed in the DACs driving the centralized transceiver's unit optical interface. Considering 5G specifications [140], the OFDM waveform parameters are: 256MHz bandsize, 256-point FFT and 64-samples Cyclic Prefix (CP).

4.3.2 Baseband Unit Digital Architecture

For our Baseband Unit we use the ZCU111 board hosting a Xilinx's Zynq Ultrascale+ RFSoC device [22], which integrates a quad-core Cortex-A53 APU, a dual-core Cortex-R5 RPU, an FPGA fabric and dedicated ADCs/DACs. Moreover, each ADC/DAC component has its own digital datapath that offers real or complex signal processing, digital-up/down conversion units and mixing to digital IF frequencies in the order of GHz. Furthermore, specialized soft-decision FEC (SD-FEC) hard-blocks support the processing requirements.

Currently, similar deployed systems use an FPGA device along with external DACs/ADCs to implement the required functionality. These systems adopt the JESD204B standard as the link between converters and the receiver/transmitter. The downside of this approach is the development time needed to optimize the communication, as well as the power consumed. On the other hand, the RFSoC device offers an embedded hardware platform for the deployment of the proposed DSP engine, alleviating extra development or performance costs. In the presented setup, the RFSoC device includes the DSP blocks for the transmitter and the receiver. The board communicates with an ethernet switch

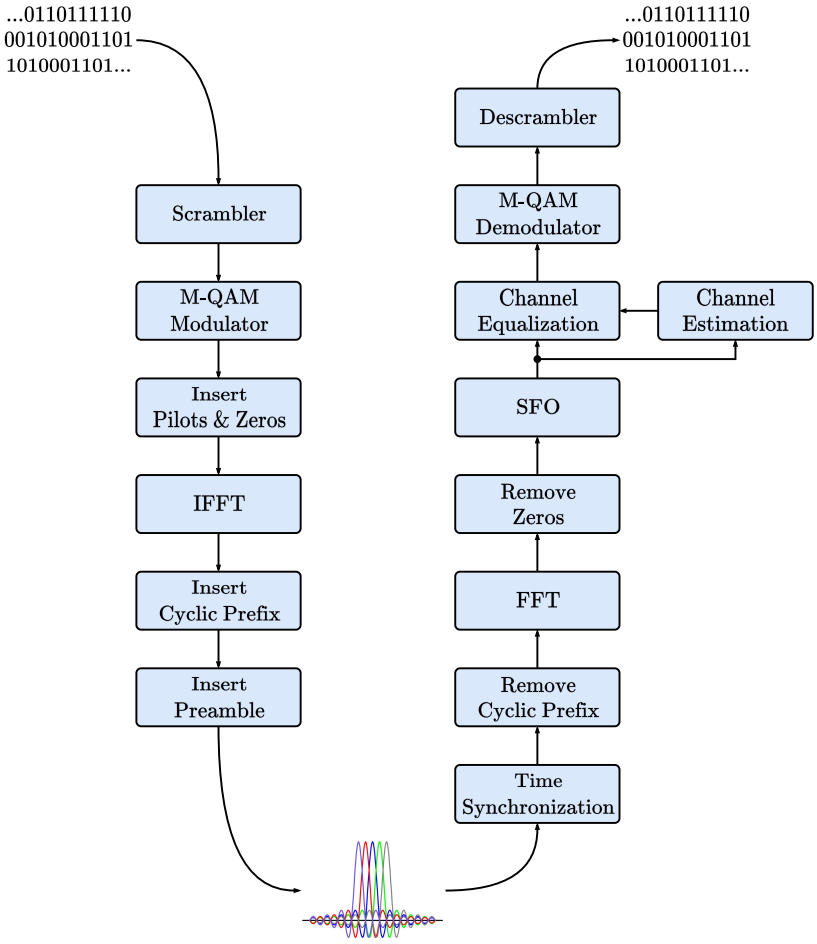


Figure 4.2: Block diagrams of typical OFDM transmitter and receiver DSP chain

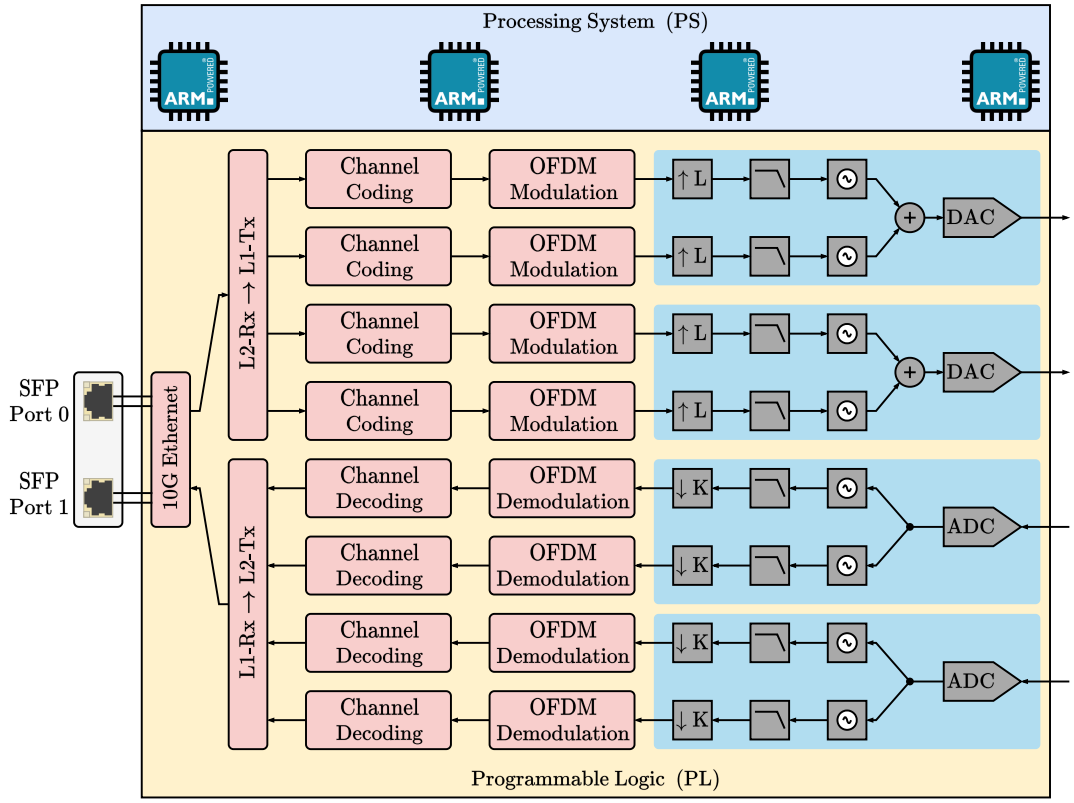


Figure 4.3: High-level architecture of DSP processing on Zynq Ultrascale+ RFSoC device.

utilizing the 10GBASE-SR standard through an SFP28 connector and by SMA cables it interfaces the ADCs/DACs with the analogue front-end.

The transmitter gets data from the ethernet switch and forwards these into four independent processing paths that handle a 256MHz band each. The bands are processed independently (Fig.4.2 transmitter) and their data are forwarded to four distinct DAC units configured in pairs to merge their respective bands and to produce two dual-band signals digitally up-converted to 1.5GHz digital IF and converted to analog. These analog signals are transmitted through SMA cables in an analog synthesizer, capable to up-convert the received signals to their analog IF of 5GHz and from there drive the next stage, which consists of the optoelectronics components. In the receiver, optical signals are converted to electrical on the optoelectronics components. These analog signals have an analog IF of 5GHz and are down-converted to 1.5GHz analog IF by the analog synthesizer, that feeds the ADCs of the RFSoC device. A single ADC digitizes the received signal and it further digitally down-converts a dual-band signal to baseband, separates it to two single bands and sets apart the I/Q parts of the signals. The output of an ADC is then processed by the FPGA DSP blocks of the receiver subsystem (Fig.4.2 receiver) and the results are transmitted through the Ethernet IP core to the external Ethernet switch. Figure 4.3 illustrates the high-level architecture of the DSP engine on RFSoC device. Note that the dual-band creation on the DACs is simplified for illustrative purposes.

We illustrate the design of the DSP algorithms of the RFSoC device by the description of the receiver processing chain for a single band signal. An ADC samples the analog waveform with 4.096GHz sampling rate and after Digital-Down Conversion (DDU) and decimation x8 provides to the FPGA a 256Msps single band signal separated to its I/Q parts. First, a timing synchronization algorithm on two streams of samples locates the frame start. The output rate will be 256Msps. Next, the samples of the CP are removed. The fourth step is the FFT decimation in frequency radix-2 FFT algorithm; it processes 256 complex points since every OFDM symbol consists of 256 I/Q samples (179 carriers, 25 pilots and 52 zeros) and produces results with ∼256Msps. The fifth step implements the SFO correction block based on the Viterbi Fourth Power Estimator algorithm. The block receives the complex output points from the FFT block and corrects the frequency offsets.

At this stage, after completing the correction of the possible errors introduced during the transmission of the signals through the channel, the DSP chain performs the channel estimation and equalization. This procedure takes as input the results calculated by the SFO correction block and uses the 25-pilots samples present in each OFDM symbol to perform its operation. The results of this block are forwarded to a soft-decision QAM demodulation block. The QAM demodulator can demodulate 4/16/64-QAM constella-

tions and implements the approximate log-likelihood ratio (ALLR) algorithm to give an estimate on the value of the data bits received. Finally, the ALLR values are given as input to the soft-decision FEC block. The FEC block creates a binary stream of data based on the ALLR values, and gives the final stream of binary data to Descrambler, which in turn forwards the binary stream of data is the input to the PHY L2 Layer block.

4.3.3 PHY L2 Layer

For the current utilization of the RFSoC as a BBU, the Ethernet interfacing is realized using the 10G/25G High Speed Ethernet Subsystem Xilinx IP Core, configured for 10GE. In order to map incoming Ethernet traffic to the DSP chains of the baseband processing and also recover Ethernet frames from demodulated OFDM symbols, we design and develop custom L2 functionalities in VHDL. Our VHDL includes basic error handling, flow control, frame encapsulation and allocation of DSP resources based on VLAN tags, all programmable at runtime between 4/16/64-QAM.

L2 Ethernet-Rx mapping to L1 Tx-OFDM

The architecture of the L2 functionalities for the Ethernet receive (or RF transmit) side is depicted in Figure 4.4. First, the incoming Ethernet packet is checked for errors and it will be dropped if the 10GE IP Core reports errors. Then the VLAN Route components routes the packet according to its VLAN tag to one of the 4 corresponding DSP chains that the architecture was designed to include (Figure 4.4). We note here that the VLAN tagging is part of the SDN functionality of bandwidth allocation and it is implemented by a network controller external to the FPGA. The length of the above incoming packet is calculated on the fly and both the packet and its length value are buffered to FIFOs performing CDC. The Delimiter FSM is responsible for appending the delimiter of Figure4.4 as a header to the Ethernet packet. The design choice for the delimiter is 32-bits wide to induce the minimum overhead possible to the air transmission while still maintaining an increased chance for the Ethernet packet to be correctly recovered at the RF receive side. It consists of a static 12-bit signature field with a hard coded value as well as a 12-bit length field indicating the Ethernet packet's length. The length value is protected by a CRC8 field in order to minimize the chances of erroneous detection of the delimiter at the opposite side. Finally, the Flow Control FSM converts the incoming 64-bit words into either 2/4/6-bit words based on the modulation order (4/16/64-QAM) and forwards them to the physical layer. In the absence of Ethernet traffic, the Flow Control FSM will

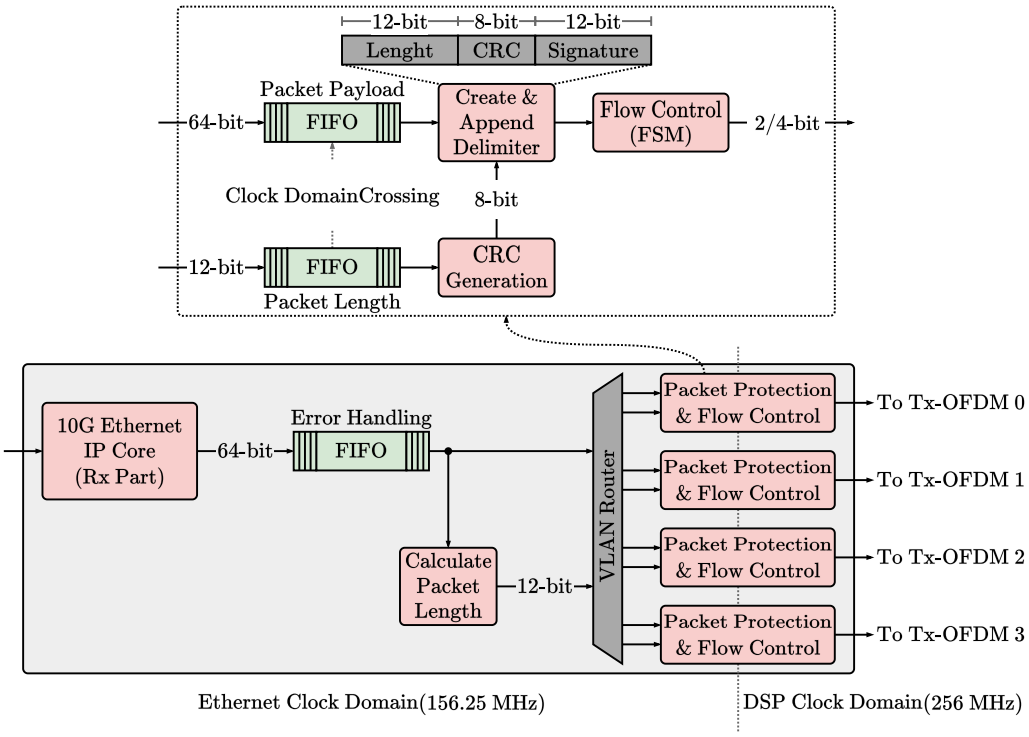


Figure 4.4: L2 architecture for Ethernet to Tx-DSP mapping.

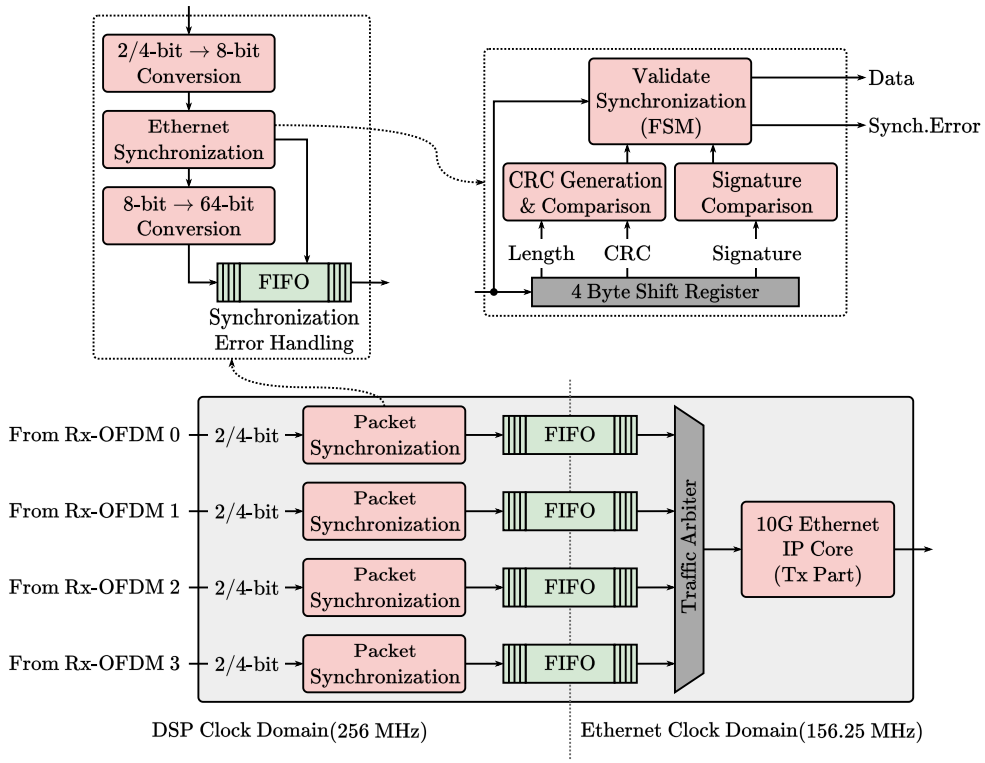


Figure 4.5: L2 architecture for Rx-DSP to Ethernet mapping.

indicate this input status to the L2 of the RF receive side by inserting a null delimiter with a length value of zero. In this case, idle symbols which are defined as x00, will be forwarded to the physical layer following the null delimiter until a new Ethernet packet shows up.

L1 Rx-OFDM mapping to L2 Ethernet-Tx

The L2 functionalities architecture for the RF receive (or Ethernet transmit) side is shown in Figure 4.5. Depending on the modulation order, 2/4/6-bit values are concatenated to bytes and forwarded to the *Ethernet Synchronization* component. This component is responsible for performing synchronization of Ethernet frames by correctly recovering them based on the received 32-bit delimiter. Its internal structure is also depicted in Figure 4.5. Whenever a new byte is received a new 32-bit word is formed by means of a 4-byte shifting register. The 32-bit word is then processed based on the expected delim-

iter structure, the CRC8 value of the length field is calculated and compared with the received CRC8 and the expected signature value is compared to the received corresponding field, all concurrently. The outputs of the comparison are forwarded to the *Validate Synchronization* FSM, which decides whether a null or actual delimiter is received at the expected time (after the end of the previous frame). If a delimiter is detected at an incorrect time, either while recovering a previous frame or not exactly after it has been recovered, the *Validate Synchronization* FSM will signal the *Synchronization Error Handling* FIFO to drop the frame at this FIFO's output. Such errors during recovery stop frames from being forwarded to the upper layers. The recovered packets are stored in a CDC FIFO ready to be transmitted via Ethernet. Finally, the *Traffic Arbiter* component aggregates traffic from the 4 independent DSP chains one-by-one and in a round robin fashion. Upon availability, traffic is forwarded as Ethernet packets to the 10GE IP Core.

4.3.4 PHY L1 Layer DSP

Tx-OFDM DSP

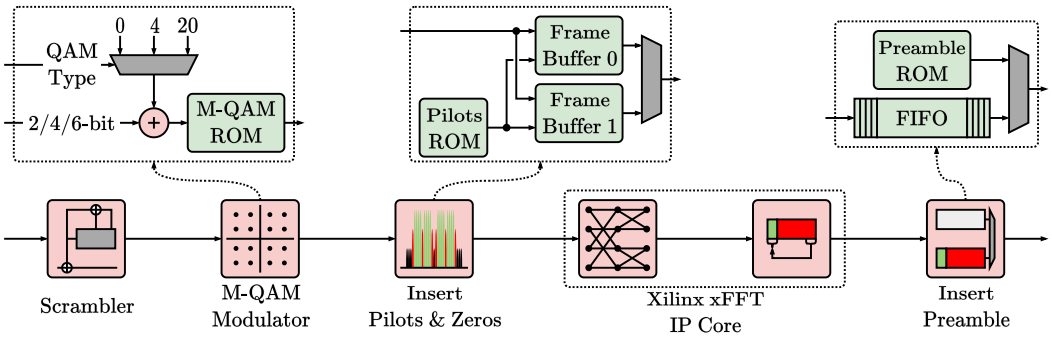


Figure 4.6: Tx-OFDM processibg pipeline.

As our Tx-OFDM processing pipeline is less complex than the Rx-OFDM processing pipeline, we will describe its functionality in a uniform manner. The hardware architecture of the developed Tx-OFDM processing pipeline is depicted in Figure 4.6. To begin, a stream of data bits is fed into the *Scrambler*, which processes the data and outputs a new stream of binary data. Afterward, the *QAM Modulator* takes in groups of bits and, based on an external control signal, maps this binary stream to complex I/Q symbols using different modulation schemes. The QAM symbols generated are subsequently passed to the *Insert Pilots & Zeros* module, where they are combined with the pilot symbols

and arranged to form the OFDM symbol frame. Additionally, a zero data symbol is appended to the guard-band carriers during this process. The OFDM symbol frame is then processed by a of-the-self IFFT-IP core, which transforms the frequency domain OFDM signal into the time domain. The IFFT core also automatically appends the cyclic prefix (CP). Afterward, the time-domain OFDM signal, along with its cyclic prefix, is temporarily stored in a FIFO buffer within the *Insert Preamble* module before being forwarded to the DAC. The *Insert Preamble* module is responsible for adding a time-domain Preamble symbol every nine time-domain OFDM symbols (e.g., P123456789P123456789...). Every hardware block within the Tx-OFDM is designed with full pipelining and minimal data buffering, ensuring a throughput of 1 data per clock cycle and facilitating a continuous data stream from the L2 layer to the DAC input.

Time Synchronization

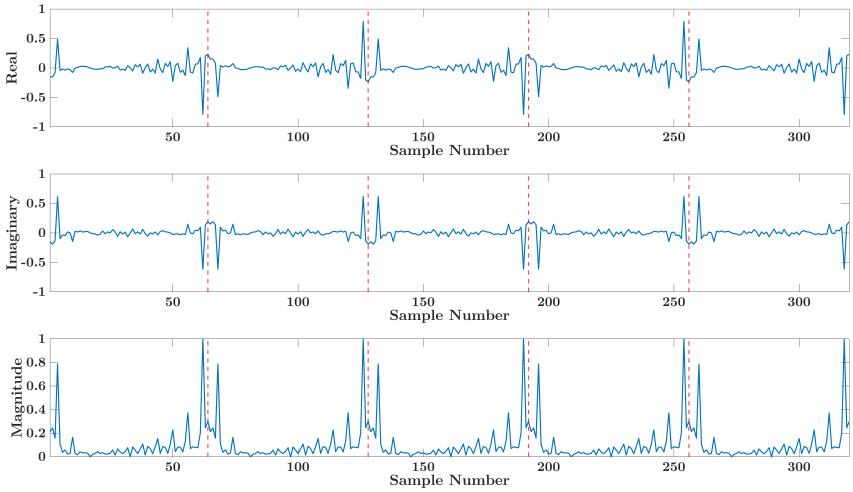


Figure 4.7: Time domain plot of preamble signal.

The Time Synchronization (FFT window alignment) is handled in the time domain, as illustrated in Figure 4.2. The basic principle relies on identifying an expected time domain sequence or some form of repetition in the received signal. To achieve this, a preamble is utilized, and a cross-correlation algorithm is executed at the receiver side to track the OFDM symbol boundaries. The structure of the preamble used is shown in Figure 4.7. Notably, the magnitude of the preamble exhibits periodicity, which we take into consideration when designing our time synchronization architecture. Additionally, to filter out possible false detections, we dynamically adjust the threshold used to find the peaks of correlation by monitoring the mean correlation value of the received signal with

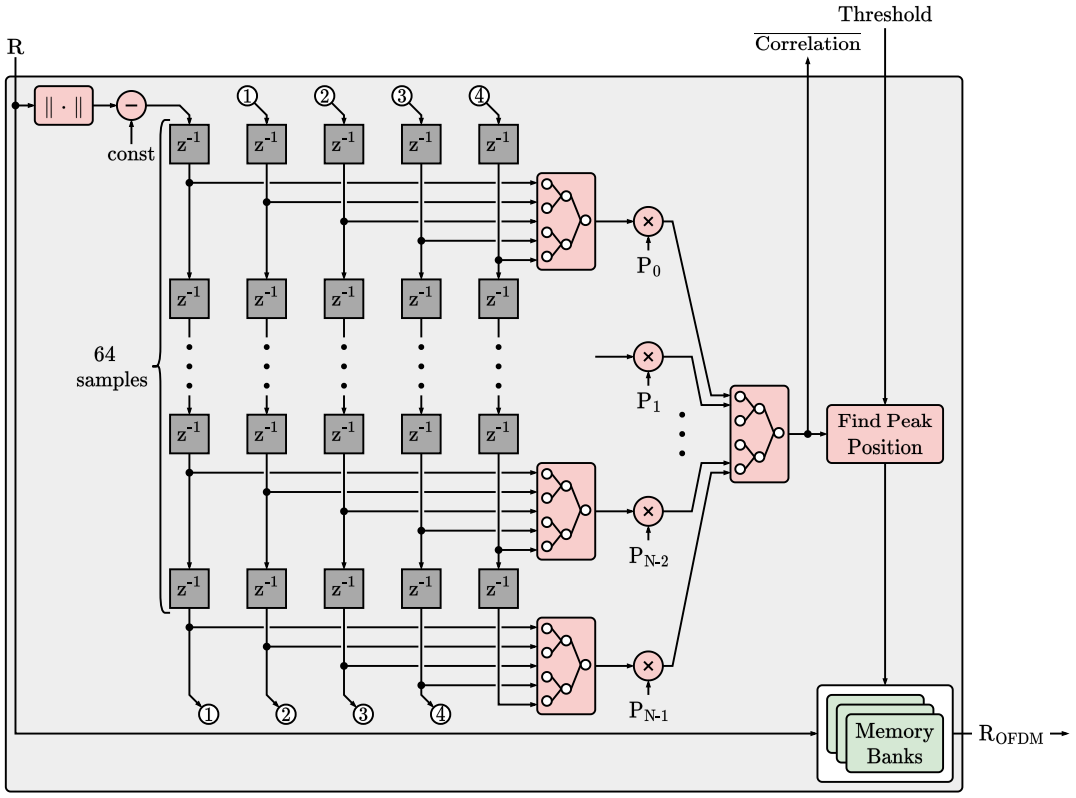


Figure 4.8: Time synchronization architecture.

the reference one stored in the receiver. Figure 4.8 illustrates the architecture developed for the Time Synchronization. Upon entry of the received signal samples into this block, their magnitudes are computed on the fly, and a constant value is subtracted from each magnitude and also stored in a local RAM buffer. Subsequently, these new magnitude values are shifted in a regular structure based on the mentioned periodicity. The samples with the same position inside each period of the preamble are then forwarded in parallel to an adder tree structure, where they are summed together. The subsequent stage involves computing the cross-correlation with the local copy of the preamble amplitude. Once a peak is detected (correlation metric > threshold), the *Find Peak Position* module performs a fine search in a window around the detected peak to confirm whether a true synchronization has occurred and determine the exact position, or to identify if the peak detection was false. When a correct detection occurs, the RAM buffer receives a signal to proceed and forwards the temporarily stored received OFDM symbol to the next component. All the processing stages is fully pipelined and support a throughput of 1 data per clock cycle

Phase Shift Estimation and Compensation (SFO)

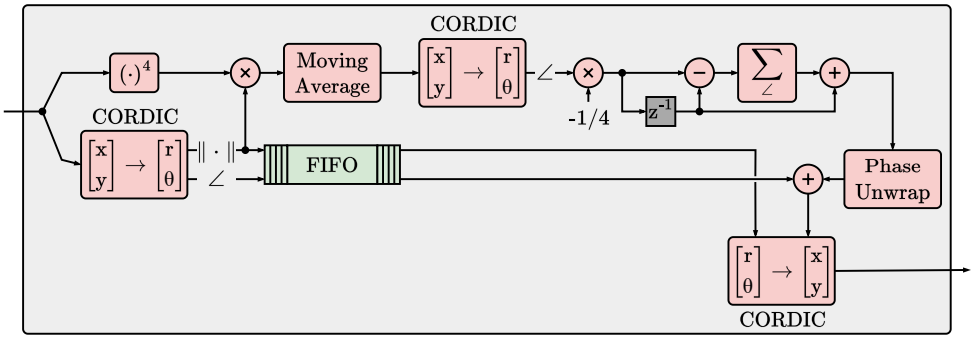


Figure 4.9: Phase shift estimation and compensation architecture.

The phase shift estimation and compensation module comprises seven main components: the fourth power calculator block, two Cartesian to Polar conversion blocks, a moving average filter, the delay block, the phase accumulation block, the Phase Unwrap, and the Polar to Cartesian conversion block (Figure 4.9). To determine the fourth power of the complex input, the fourth power calculator block employs two complex multipliers. The first multiplier calculates the square of the complex number, while the second one computes the fourth power. The Cartesian to Polar conversion block utilizes the CORDIC algorithm to extract the magnitude and phase of the Cartesian input. The moving average filter block is a transposed Finite Impulse Response (FIR) filter with L taps, where all the coefficients are equal to $1/L$. The delay block acts as a FIFO memory with a depth of $L/2$, resulting in a delay of $L/2$ cycles for the input. The phase accumulation block takes the phase of a complex number, represented as a real number, as input and continuously tracks the phase error from symbol to symbol. The phase unwrap operation reconstructs the signal's original phase by adding appropriate multiples of 2π to each phase input, ensuring that the values are between $-\pi$ and π. The corrected phase, obtained from the phase accumulation block, is added to the output of the delay block to perform the final phase correction. The output is the corrected phase, and the phase values are between $-\pi$ and π. Finally, the delayed magnitude of the input symbol, along with the corrected phase, is fed to the final CORDIC block, which translates the Polar representation of the signal to Cartesian format for further downstream processing.

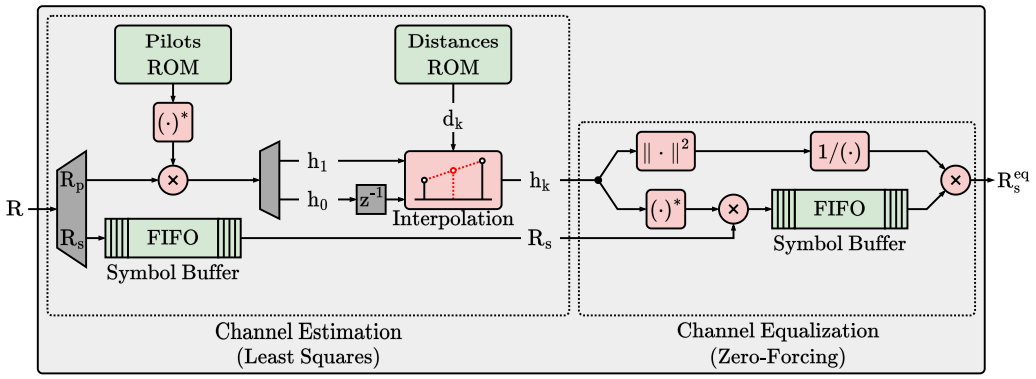


Figure 4.10: Channel estimation & equalization architecture.

Channel Estimation & Equalization

To achieve correct demodulation of the received signal, an estimation of the inverted channel response must be calculated and applied to the signal, minimizing the impact of the channel noise and frequency selectivity. The term "channel" refers to the whole transmission link that includes the length of the sSMF fiber, the wireless link, any electro/optical amplification and up/down-conversion required. The above processes is implemented at the receiver, by an equalization filter. For a C-RAN compatible DSP engine, the RFSoC-FPGA board will perform the equalization DSP functions, alleviating end-user hardware processing.

An advantage of multi-carrier waveforms is their robustness to chromatic dispersion and wireless link multipath effects. Thanks to the cyclic prefix insertion to the transmitted OFDM symbols, the time-domain signal presents periodicity which allows for a simpler channel estimation in frequency domain. Thus, the receiver equalization can be performed by a single-tap filter. The equalization schemes are selected by taking into account the wireless channel transmission effects, since the optical link is static and adds insignificant distortion to the signal. In our implementation we use the *Least Squares* for estimating the channel response and *Zero-Forcing* to equalize the received symbols. We have developed a single blocks that includes both estimation and equalization and its architecture is given in Figure 4.10.

Upon the arrival of the received QAM and Pilots symbols, we distinguish the Pilots symbols from the QAM data symbols. The QAM data symbols are then temporarily stored in a FIFO buffer, while the received Pilot symbols are utilized to compute an estimate of the channel response at these carrier positions. Once we obtain channel response estimations

for two consecutive Pilot symbols, we conduct a linear interpolation to determine the estimation of the channel response for the data carriers located between these pilots. Then, the channel estimation response of the data carrier along with the QAM symbol of these carrier are forwarded to the *Zero Forcing* sub-block, that computes the final equalized QAM symbol. The equalization is performed based on

$$R_k^{eq} = \frac{(h_k^* \times R_k)}{\|h_k\|^2} \tag{4.1}$$

where R_k is the unequalized received QAM symbol, h_k its channel estimation response and R_k^{eq} the final equalized QAM symbol. All the processing stages is fully pipelined and support a throughput of 1 data per clock cycle

M-QAM Demodulation

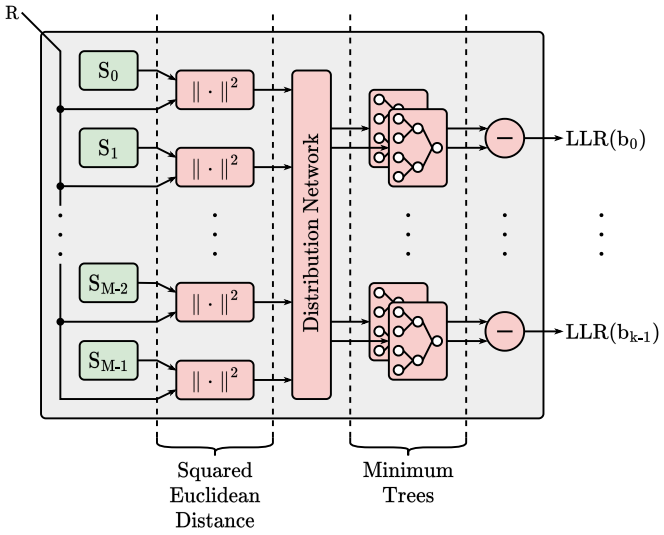


Figure 4.11: Approximate LLR M-QAM demodulator architecture.

To design an efficient QAM Demodulator architecture, the maximum supported constellation is taking into account, thus a single components is used to demodulate 4/16/64-QAM constellations. The demodulator implements the Approximate Log-Likelihood Ratio (ALLR) algorithm [13] to give an estimate on the value of the data bits received, and is defined by the following equation:

$$LLR(b_i) = -\frac{1}{\sigma^2}\left(\min_{s\in S_0}\left((r_x - s_x)^2 + (r_y - s_y)^2\right) - \min_{s\in S_1}\left((r_x - s_x)^2 + (r_y - s_y)^2\right)\right) \tag{4.2}$$

where:

- b the transmitted bit (one of the k bits on an M-QAM symbol)
- r_x, r_y the coordinates (I/Q) of the received QAM symbol
- S_0, S_1 the sets of reference constellation symbols with a value of 0 or 1 at the specified bit position b_i respectively
- s_x, s_y the coordinates (I/Q) of the reference symbol
- σ^2 the noise variance

Figure 4.11 presents the architecture of the QAM Demodulator, which is based on Equation (4.2). For every input QAM symbol and based on external control signal indicating the modulation scheme, initially, the squared euclidean distance is computed from all the QAM symbols of the reference constellation in parallel. Then a customized switch with hardwired connections between input and output ports is used to distribute the previously computed distances, thus creating the sets S_0 and S_1. These sets of are the forwarded in a parallel minimum trees that compute the minimum from each set S_0, S_1. After calculating the minimums, the final subtraction to obtain the ALLR value takes place. There are a total of six parallel paths for computing the ALLR values, corresponding to the number of bits required for the highest supported constellation, which is 64-QAM. For lower QAM constellations, the ALLR values for higher-order bits are invalid and ignored downstream.

Hardware Architecure Evaluation

We implemented the entire Baseband processing system on a Xilinx Zynq Ultrascale+ RFSoC-FPGA, specifically the XCZU28DR device, hosted on a ZCU111 evaluation board. In Table 4.1 we present the utilization report of the implemented DSP blocks. As can be seen from the results, for the Tx-OFDM processing pipeline, the majority of blocks consume negligible amount of resources. The IFFT blocks is the exception and consumes $\sim 29\%$ more LUTs and $\sim 11\%$ more DFFs than the Pilot & Zeros blocks, which has the highest utilization among the other blocks. Additionally, we observe that the IFFT block is the most computationally intensive component in the transmitter, and as a result, it is the only DSP block that utilizes DSPs. On the other hand, in the receiver, we notice a more even distribution of general purpose resources (LUTs, FFs) among the different blocks. Considering that the receiver is typically the most computationally intensive part of a Baseband unit, we can see increased resource utilization compared to the transmitter. In the receiver part, the QAM demodulator stands out with the highest DSP utilization, even surpassing the FFT block. This is because the implemented architecture is designed for 64-QAM, even when lower modulation schemes are used, leading to increased compu-

tational demands. Another interesting finding is the substantial DFFs utilization in the Time Synchronization block. This is a result of the long preamble period (64 samples) used in our architecture, which creates a lengthy chain of shift registers, significantly increasing the DFFs used for buffering. Additionally, Table 4.1 presents the resource utilization of other blocks in our system, such as the 10G eth and L2 components. The overall system performance is notable since the full-scale system (4Tx+4Rx) utilizes a relatively small number of resources. However, while each individual component can achieve operating frequencies above 400MHz, the combined system's operating frequency reduces to 256MHz.

Table 4.1: Resource utilization on XCZU28DR[1] RFSoC

Component	**LUT**	**DFF**	**DSP**	**RAMB**	**Freq. (MHz)**
QAM Mod.	103	176	0	0.5	500
Pilots & Zeros	430	329	0	0.5	500
IFFT	1488	2949	30	4	500
Preamble	170	117	0	1	500
Time Sync.	8799	55434	66	0	530
FFT	1214	3174	30	3.5	500
Zero Removal	100	91	0	1	500
SFO	3398	3583	15	0.5	500
Channel Est. & Eq.	1451	18	18	1.5	400
QAM demod	1554	2523	128	0	500
(De-)Scrambler	82	98	0	0	500
Total Tx+Rx	17971	70765	287	12.5	
L2 Tx+Rx	1170	1894	0	8	
10G Eth. (2 lanes)	23595	55301	0	5.5	
Misc. (AXI, RF, etc.)	6800	8487	0	5	
4x(Tx+Rx)+10G Eth.+(L2 Tx+Rx)	90055 (21%)	329506 (39%)	1106 (26%)	62 (6%)	256

[1] Total resources: FF = 850K, LUT = 425K, BRAM = 1K, DSP = 4.2K

4.4 Real-world experimental evaluation

Continuing with our evaluation, we took a step further and integrated the developed RFSoC Baseband unit into two real-world experimental trials as part of the 5G-PHOS project [25]. During these trials, we tested its performance in combination with various components and functionalities. In the following sections, we present these two experimental trials and share some indicative results, focusing primarily on our Baseband unit.

End-to-End Real-Time Service Provisioning [23]

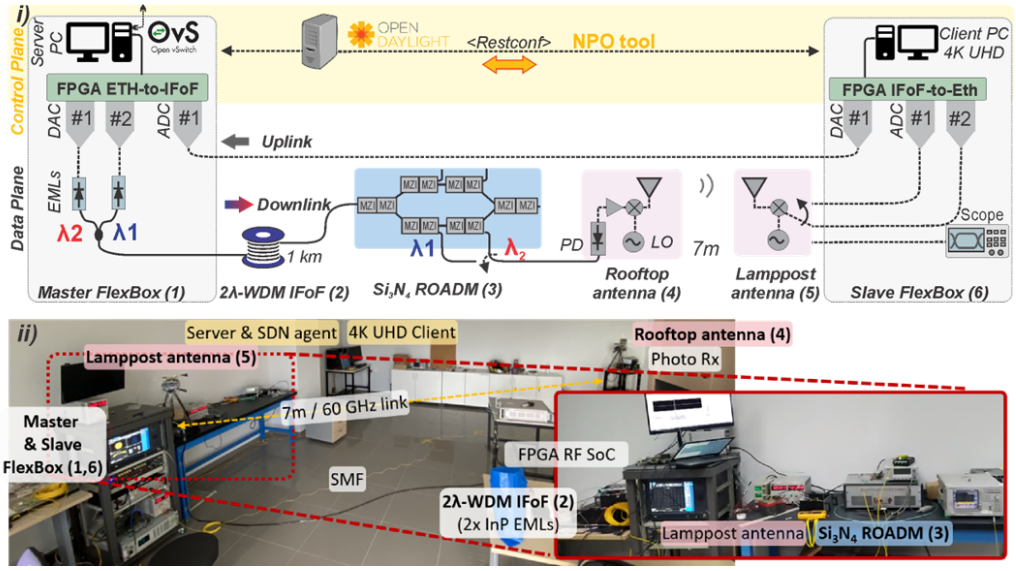


Figure 4.12: Expiremental setup for End-to-End Real-Time Service Provisioning [23].

Figure 4.12 (top) illustrates the experimental setup used to to provide a real-time trial of a 5G FiWi mmWave/IFoF X-haul network. In this setup, the Master Flexbox is responsible for transmitting a 2λ-WDM FiWi IFoF/60GHz data traffic through the Reconfigurable Optical Add Drop Multiplexer (ROADM) and a pair of mmWave antennas towards the slave Flexbox. The key components are numbered and colored to directly correspond with the employed devices in the actual experimental setup, as shown in the bottom of Figure 4.12.

At the Master Flexbox side of the setup, there is the Server PC that hosts the Mellanox NIC for implementing the network's Control Plane. This server also acts as the host for the streaming services. On the Slave Flexbox side, a second PC called the Client PC is present. The Client PC functions as the node requesting a set of services from the Server PC, including the streaming of a 4K-UHD video. In the downlink direction, the Server PC is optically interconnected to the master RFSoC FPGA board via Ethernet. This connection employs a 10 Gb/s SFP+ transceiver module to transmit the packets of the requested service, i.e., the video stream packets. At the Master FlexBox, the primary function of the NIC is to assign suitable 802.1Q VLAN tag values to each downlink ETH packet. These VLAN tags are then utilized by the FPGA RFSoC to determine the appropriate optical wavelength for modulating the optical downlink traffic. Upon receiving the VLAN tagged ETH packets, the FPGA RFSoC board performs a look-up in a pre-defined VLAN table. This mapping associates each VLAN tag value with a specific physical DAC output responsible for driving the modulation process of a dedicated EML using Intensity Modulation/Direct Detection. Additionally, the FPGA RF SoC board acts as a real-time analog Baseband Unit (BBU) that converts the received data traffic/ETH packets into an IFoF optical stream. Within the FPGA RFSoC board, there are two different DAC outputs, each connected to an InP EML with a distinct emission wavelength. These two outputs function as separate analog IFoF real-time interfaces, carrying the ETH traffic at two different wavelengths. This arrangement forms a 2λ-WDM X-haul downlink data traffic. In summary, the value of the 802.1Q VLAN tag mechanism dictates the forwarding of ETH traffic to the appropriate wavelength of the λ-WDM analog IFoF traffic. This enables SDN-controlled packet forwarding and wavelength routing operations.

After the DSP operation at the RFSoC processor, the resulting signal undergoes electro-optical (e-o) conversion using low-cost IM/DD (Intensity Modulation/Direct Detection) to generate the optical traffic flows that will traverse the FiWi X-haul network. To achieve this, two InP EMLs (Electro-absorption Modulated Lasers) are employed in the downlink direction. These EMLs emit modulated optical output power at wavelengths λl= 1538.7 nm (marked in blue) and λ2=1540.4 nm (marked in red), as shown in 4.12. The modulated EML outputs are then combined using a 2x1 power combiner, resulting in a 2λ-WDM (Wavelength Division Multiplexing) IFoF data stream carrying real-time traffic. This 2λ-WDM IFoF data stream is propagated through a standard SMF (Single-Mode Fiber) with a length of approximately 1 km before reaching the input port of the low-loss 1x4 Si3N4 (Silicon Nitride) ROADM (Reconfigurable Optical Add-Drop Multiplexer). The ROADM allows each output to be fiber-connected to the respective rooftop mmWave antennas situated around the stadium-hotspot area. In this experimental setup, two mmWave antennas are used, which corresponds to the number of wavelengths in the WDM optical data stream. Once the WDM fiber transmission and demultiplexing process is completed, the optical signal from each of the Drop outputs of the ROADM reaches

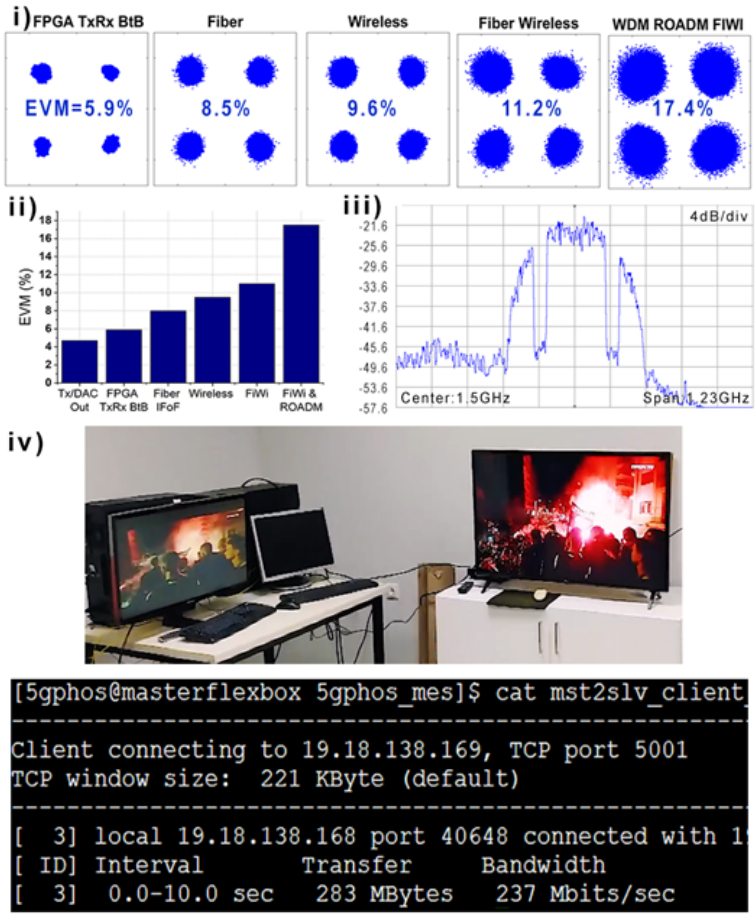


Figure 4.13: Experimental results (EVM, RF spectrum, iperf tcp) [23].

the respective antenna site. At each antenna site, the optical stream is connected to a 10 GHz Avalanche Photo Diode (APD) to perform opto-electronic (o-e) conversion, which generates the electrical Intermediate Frequency (IF) stream. This electrical IF stream then undergoes an electrical RF mixing stage, where it is up-converted to the 60GHz band and amplified. The resulting V-band signal is transmitted using a 60GHz horn antenna, forming a wireless link that extends over a 7m distance.

At the lamp-post antenna side, a similar 60 GHz horn antenna receives the signal. It is then amplified and down-converted by an integrated mixing stage, converting it from the 60GHz band down to the Intermediate Frequency (IF). The down-converted signal is then fed to the input of the Analog-to-Digital Converter (ADC) of the RF SoC. At

this stage, the IF signal is converted back to a digital stream and forwarded to the second Network Controller (NC) of the slave Flexbox, located at the remote unit. The second NIC removes the VLAN tag, completing the downlink transmission. A similar operation and signal flow are followed for the second wavelength dropped at another ROADM Drop output, and vice versa for the uplink transmission, achieving bi-directional communication. For the current demonstration and to simplify the setup, the real-time uplink transmission of the IF signal from the FPGA DAC RF output was directly wire-connected to the input of the ADC, without involving any optical IFoF path. Figure 4.13 shows some indicative performance results directly related to the RFSoC Baseband unit

SDN-reconfigurable, FPGA-based TxRx for Analog-IFoF/mmWave RAN, in MNO's infrastructure [24]

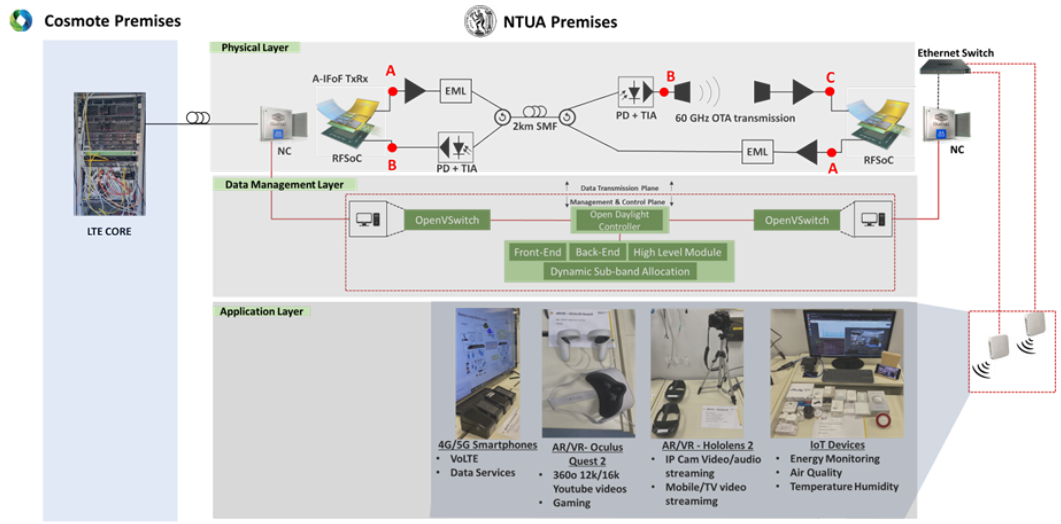


Figure 4.14: Experimental setup for Analog-IFoF/mmWave RAN in MNO's infrastructure [24].

This experimental trial showcased the integration of the developed A-IFoF TxRx RFSoC Baseband unit, enabling bidirectional FiWi connectivity, into standard mobile core infrastructure. The network infrastructure included a fully-fledged Long Term Evolution (LTE) Core network testbed with support from an Openstack multi-cloud infrastructure. Additionally, a NOKIA's Flexi Zone Multiband Indoor 4G small cell acted as the radio access node. To evaluate the performance of the deployed A-IFoF transceivers-assisted

network in delivering mobile services to end-users, the testbed was equipped with various components. These included a COSMOTE TV Set-Top-Box (STB) for live video content provisioning, High Definition (HD) / 4K Internet Protocol (IP) and High Definition Multimedia Interface (HDMI) cameras for live footage, AR/VR Glasses (OCULUS Quest-2, HOLOLENS-2), and Internet-of-Things (IoT) devices with sensors. Additionally, Android/iOS smartphones/tablets supporting carrier aggregation were utilized to boost traffic demand and evaluate the active capacity reconfiguration capabilities of the TxRx, as well as to measure the maximum achievable throughput. Figure 4.14 showcases the End-to-End (E2E) deployed setup, comprising the physical, data management, and application layers. In the Downlink (DL), mobile traffic was generated and transmitted as Ethernet packets from the EPC (located at COSMOTE premises). These packets were then sent via a dedicated dark fiber link to the NC, situated at the remote laboratory premises. The traffic originating from the mobile equipment was connected to the NC part of the developed A-IFoF TxRx using standard Small Form-factor Pluggable (SFP) connectivity. It was then forwarded to the FPGA fabric along with the A/D and e/o interfaces.

Specifically, the DAC output was amplified and directed to a (CANGLONG CEB510 series) Electro-absorption Modulated Laser (EML) emitting at 1560.42nm. The optical signal was transmitted over a 2 km fiber spool of Single Mode Fiber (SMF) and detected by a (Discovery Semiconductors DSC-R402) 10G linear InGaAs photoreceiver. The photoreceiver output was connected to the IF-to-V-band upconverter board (Gotmic gTCS020B). The up-converter output was then interfaced with a directional Tx-pyramidal gain horn V-band antenna, which featured 23 dBi gain and 10° beamwidth, using a WR15 waveguide. At a distance of 1m horizontally, an identical Rx-side radio unit (Gotmic gRCS016B) received the mmWave radio waveforms, and the down-converted radio waveforms were directed to the ADC unit. The received IF up converted waveforms were processed through the receiver-side DSP chain developed on the FPGA-based A-IFoF TxRx. The bitstreams were converted back to Ethernet traffic and forwarded to the NC, then to a small cell, providing access to mobile user equipment. For the Up-Link (UL) segment of this setup, a symmetrical optical link was employed. The EML used for the UL direction emitted at 1540nm, enabling the co-transmission of both directions in a single fiber. Multiplexing of the DL and UL optical paths in the same fiber core was achieved by using two low-loss optical circulators, as depicted in Figure 4.14.

In order to demonstrate the network's ability to handle varying traffic demands and surpass the capacity limitations of small cell connectivity, two extra pairs of the A-IFoF TxRx's DAC and ADC units were utilized. These additional units facilitated the parallel implementation of a second bidirectional link. This analog RoF link was identical to the first one and served to supply traffic to a second Small Cell. The SDN controller

dynamically activated this link when the traffic demand exceeded the capacity of a single Small Cell. Conversely, it deactivated the link when the demand reduced to optimize network resources accordingly. Figure 4.15 shows some indicative performance results directly related to the RFSoC Baseband unit

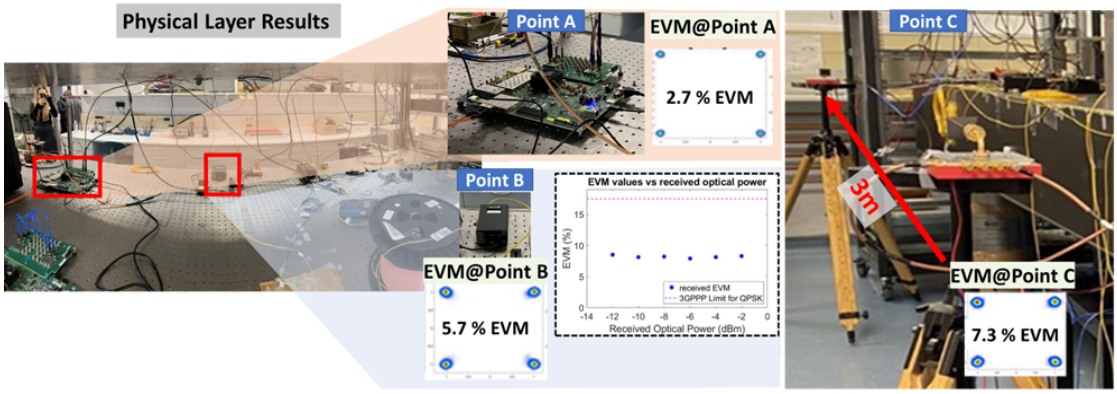


Figure 4.15: Experimental results (EVM) in different point of the network [24].

4.5 Conclusions

This chapter introduced the design of a BBU using OFDM transmission, with a focus on 5G/B5G mobile networks. The system was successfully implemented on a cutting-edge RFSoC device, featuring 4 Tx-OFDM and 4 Rx-OFDM processing pipelines. Additionally, we provided an overview of the two experimental trials where our BBU was integrated to handle the low-level physical layer processing. In both trials, the unit operated seamlessly without causing any disruption to the services running on the mobile infrastructure.

Chapter 5

Conclusions

This chapter briefly summarizes the presented work and results of our proposed implementations and discusses the conclusions derived from this thesis. We then discus possible extensions of the present work.

5.1 Thesis Summary

Re-configurable hardware serves as a bridge between high-performance ASICs and the capabilities of conventional processor platforms, particularly in computationally intensive applications like Digital Signal Processing (DSP). This hardware offers flexibility and faster time-to-market. However, designing with re-configurable hardware can be challenging due to the integration of multiple design tools and specific architectures, which poses difficulties for designers. Designing re-configurable hardware for DSP applications has been considered difficult, thus address these challenges, researchers have sought efficient design methods that not only consider FPGA architectures but also simplify the design flow.

FPGAs now offer a cost-effective solution for DSP implementation in various applications, such as image processing, wireless communications, multimedia systems, and consumer electronics. Leading FPGA manufacturers not only incorporate simple DSP features in their devices, but advanced specialized hard-IP blocks and general purpose CPUs. These features, coupled with on-chip memory, significantly increase system throughput, surpassing conventional processing platforms.

This thesis introduces efficient architectures on SoC-FPGAs for demanding applications, spanning from image/video processing in the Medical Imaging (MI) domain to autonomous navigation in space. Additionally, it proposes a three-pillar design approach

for embedded applications on SoC-FPGAs, aiming to achieve three main objectives: 1) significantly increase system performance, 2) reduce resource utilization and power consumption, and 3) accommodate multiple applications in a single device. In more detail

- **A Synergistic Design Approach for FPGA-based SoC Platforms:** We introduced a design approach that involved extensive DSE at both the algorithmic and architectural levels to enhance our circuit implementations. We systematically applied voltage scaling to decrease power consumption at the system level within the SoC-FPGA and conducted an initial exploration of co-deploying two demanding applications on the same SoC-FPGA. Our experiments yielded the following results: 1) For circuit optimization (QAM demodulator), we achieved up to 98% reduction in resource utilization compared to the accurate floating-point architecture of the same algorithm, while increasing its operating frequency by up to 12%. 2) Depending on the architectural choices for the accelerator in an HW/SW co-designed application deployed on the SoC-FPGA device, we achieved power savings of $20 - 30\%$ at the system level, while preserving the functional correctness and nominal timing performance of the system. 3) We demonstrated the feasibility of integrating low-level DSP functions required in a BBU with AI/ML acceleration on a state-of-the-art SoC-FPGA device (RFSoC). Our initial evaluation indicates that this combination is viable in terms of resource consumption and performance metrics.

- **Efficient Architectures for Image/Video Applications in Contemporary SoC-FPGA Devices:** We developed efficient HW/SW co-designed embedded systems for image/video applications and deployed them on modern SoC-FPGA devices. The Image Registration processing pipeline, when implemented on a state-of-the-art SoC-FPGA, demonstrated superior performance compared to software designs, achieving up to 67x speedup against embedded CPUs while using single-precision floating-point arithmetic. In the case of the VBN processing pipeline, we achieved a speedup of 19x with an average power consumption of less than 5W and a mean error of 0.5% of the distance between the camera and the target satellite.

- **Baseband Processing in RFSoC FPGA for 5G/B5G Mobile Networks:** We designed efficient hardware architectures for essential DSP blocks used in OFDM transmission and reception on a BBU. By deploying multiple parallel Tx/Rx processing pipelines, we successfully delivered mobile services and high-definition video streaming to end-users in two experimental trials. Our BBU demonstrated outstanding performance, operating with an EVM well below the 3GPP specifications.

5.2 Future Work

Over the years, FPGA technology has advanced, offering quicker turnaround times, reduced costs, and less power usage compared to ASICs. The flexibility of FPGAs stands out as they can be reprogrammed even after the initial design and setup. This adaptability, allowing for design modifications even after a product's installation, is anticipated to drive market growth as shown in Figure 5.1.

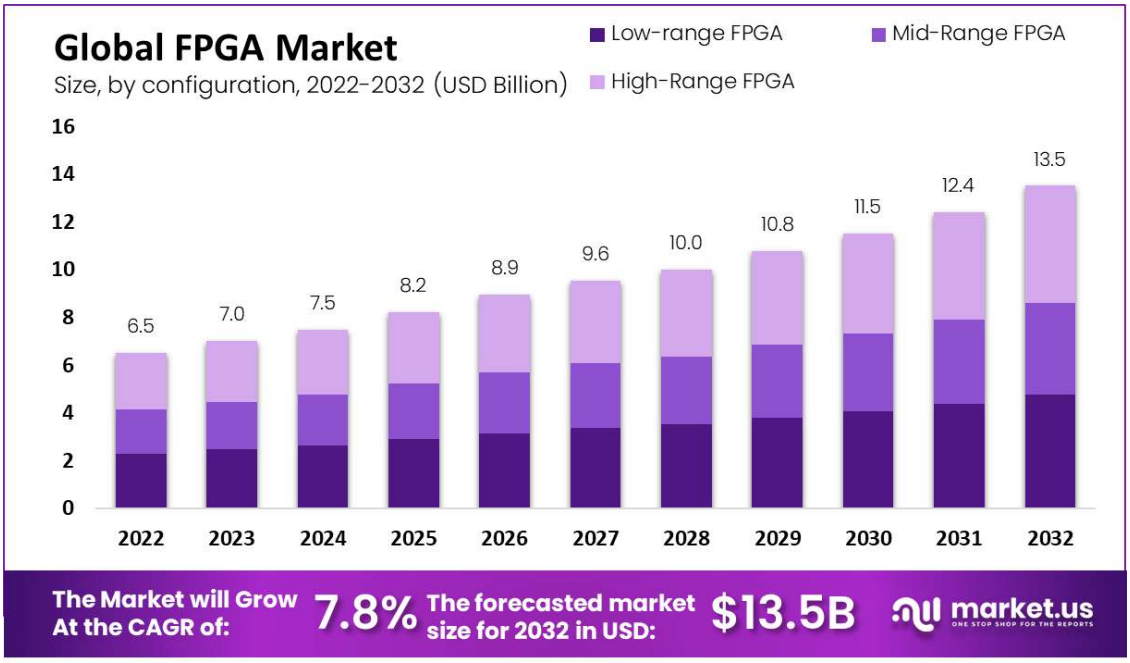


Figure 5.1: FPGA market growth projection.

This Ph.D. thesis has shown the immense potential of FPGAs as a digital platform spanning various application domains. Furthermore, with the rise in advancements like AI/ML and 5G/B5G, integrating FPGAs for these innovative sectors seems a logical decision. Still, there are existing concerns, primarily concerning FPGA development methodologies and determining which applications to deploy and manage on these increasingly heterogeneous platforms. As such, this Ph.D. thesis research can be extended in numerous potential paths for deeper investigation, as outlined below.

- **Automating Design Approach & Integration with Commercial Tools**

One possibility is to automate the proposed design approach, presented in Chapter 2, allowing designers to explore various architectural alternatives more efficiently instead of applying the approach manually. Furthermore, an important advancement would be integrating the proposed design approach into commercially available tools. This would enhance the accessibility and adoption of the approach in real-world design scenarios.

- **Exploration of Circuit Approximation in Telecommunications**

  In this thesis, we focused on developing efficient implementations for SoC-FPGAs and RFSoCs by manually fine-tuning their designs. A potential future direction is to explore the applicability of circuit approximation techniques to a wider range of DSP blocks, particularly in the telecom domain, where many DSP blocks may exhibit error resilience due to the nature of their computations. This investigation could lead to more optimized and robust designs for telecommunications applications.

- **AI/ML Applications in Telecommunications**

  As AI/ML applications gain more traction, a promising research direction is to thoroughly explore the deployment of machine learning algorithms in the telecom domain. Although there is considerable interest among researchers in applying AI/ML techniques to baseband processing, the actual deployment on real hardware remains limited. Investigating and implementing AI/ML algorithms on practical hardware can open up new possibilities for enhancing performance and efficiency in telecommunications systems.

- **Heterogeneous Computing Platforms and Mapping**

  The growing availability of diverse and highly heterogeneous computing platforms, such as Versal ACAP, presents new challenges in efficiently mapping applications to these platforms. One potential future research direction is to investigate methodologies and frameworks for effectively mapping complex DSP applications to these modern platforms without requiring a deep understanding of their underlying architectures. Furthermore, exploring the development of automatic scheduling and management features for multiple applications as extensions to these methodologies and frameworks would be of great interest. Such advancements could streamline the deployment of applications on these cutting-edge platforms, making it easier for developers to harness their full potential while optimizing performance and resource utilization.

Abbreviations & Acronyms

5G NR	5G New Radio
A-RoF	Analog Radio-over-Fiber
ACAP	Adaptive Compute Acceleration Platforms
ADC	Analog to Digital Converter
ADR	Active Debris Removal
AGC	Automatic Gain Control
AI	Artificial Intelligence
AMBA	Advanced Microcontroller Bus Architecture
AMS	Analog-Mixed Signal
ASIC	Application Specific Integrated Circuit
AWGN	Additive White Gaussian Noise
AXI	Advanced eXtensible Interface
BBU	Baseband Processing Uni
BER	Bit Error Rate
C-RAN	Centralized Radio Access Network
CAD	Computer-Aided Design

CDC	Clock Domain Crossing
CLB	Configurable Logic Block
CORDIC	Coordinate Rotation Digital Computer
COTS	Commercial Off-the-Shelf
CPRI	Common Public Radio Interface
CPU	Central Processing Unit
CV	Computer Vision
D-RoF	Digital Radio-over-Fiber
DAC	Digital to Analog Converter
DCM	Digital Complex Mixers
DMA	Direct-Memory Access
DNN	Deep Neural Networks
DPU	Deep Learning Processing Unit
DSE	Design Space Exploration
DSP	Digital Signal Processing
DUC/DDC	Digital Up/Down-Conversion
EDA	Electronic Automation Design
ESA	European Space Agency
EVM	Error Vector Magnitude
FEC	Forward Error Correction

FFT	(Inverse) Fast Fourier Transform
FIFO	First-In First-Out
FIR	Finite Impulse Response
FiWi	Fiber Wireless
FPGA	Field Programmable Gate Array
FPS	Frames Per Second
FSM	Finite State Machine
GNC	Guidance, Navigation, and Control
GPU	Graphic Processing Unit
HLS	High-Level Synthesis
I2C	Inter-Integrated Circuit
IC	Integrated Circuit
IF	Intermediate Frequencies
IoT	Internet of Things
IP	Intellectual Property
IR	Image Registration
LLR	Log-Likelihood-Ratio
LTE	Long Term Evolution
LUT	Look-Up Table
MAC	Medium Access Control

MFH	Mobile Fronthaul
MIMO	Multiple Input - Multiple Output
ML	Machine Learning
NCO	Numeric Control Oscillator
OBC	On Board Computer
OFDM	Orthogonal Frequency Division Multiplexing
PCS	Physical Coding Sublayer
PHY	Physical Layer
PMBus	Power Management Bus
PS	Pose Estimation
PTP	Precision Time Protocol
QAM	Quadrature Amplitude Modulation
QMC	Quadrature Modulator Correction
QoS	Quality of Service
RAMB	Random Access Memory Block
RF	Radio Frequency
ROADM	Reconfigurable Optical Add Drop Multiplexer
RoE	Radio-over-Ethernet
ROM	Read-Only Memory
RPU	Real-time Processing Unit

RRH	Remote Radio Head
RTL	Register-Transfer Level
SDN	Software Defined Network
SIMD	Single Instruction Multiple Data
SNR	Signal-to-Noise Ratio
SoC	System on Chip
VBN	Vision-based Navigation
VHDL	Very High-Speed Integrated Circuit Hardware Description Language
VLAN	Virtual Local Area Network
WDM	Wavelength-Division Multiplexing
Zynq PL	Zynq Programmable Logic
Zynq PS	Zynq Processing System

Publications

Journals

1. Lentaris, George; Maragos, Konstantinos; Stratakos, Ioannis; Papadopoulos, Lazaros; Papanikolaou, Odysseas; Soudris, Dimitrios; Lourakis, Manolis; Zabulis, Xenophon; Gonzalez-Arjona, David; Furano, Gianluca; ,High-performance embedded computing in space: Evaluation of platforms for vision-based navigation,Journal of Aerospace Information Systems,15,4,178-192,2018,American Institute of Aeronautics and Astronautics

2. Lentaris, George; Stratakos, Ioannis; Stamoulias, Ioannis; Soudris, Dimitrios; Lourakis, Manolis; Zabulis, Xenophon; ,High-performance vision-based navigation on SoC FPGA for spacecraft proximity operations,IEEE Transactions on Circuits and Systems for Video Technology,30,4,1188-1202,2019,IEEE

3. Kanta, K; Pagano, A; Ruggeri, E; Agus, M; Stratakos, I; Mercinelli, R; Vagionas, C; Toumasis, P; Kalfas, G; Giannoulis, G; ,Analog fiber wireless downlink transmission of IFoF/mmWave over in-field deployed legacy PON infrastructure for 5G fronthauling,Journal of Optical Communications and Networking,12,10,D57-D65,2020,Optical Society of America

4. Stratakos, Ioannis; Leon, Vasileios; Armeniakos, Giorgos; Lentaris, George; Soudris, Dimitrios;, Design Space Exploration on High-Order QAM Demodulation Circuits: Algorithms, Arithmetic and Approximation Techniques,Electronics,11,1,39,2021,MDPI

5. Kanta, Konstantina; Toumasis, Panagiotis; Tokas, Kostas; Stratakos, Ioannis; Papatheofanous, Elissaios Alexis; Giannoulis, Giannis; Mesogiti, Ioanna; Theodoropoulou, Eleni; Lyberopoulos, George; Lentaris, George; ,Demonstration of a Hybrid Analog–Digital Transport System Architecture for 5G and Beyond Networks,Applied Sciences,12,4,2122,2022,Multidisciplinary Digital Publishing Institute

6. Vagionas, C; Maximidis, R; Stratakos, I; Margaris, A; Mesodiakaki, A; Gatzianas,

M; Kanta, K; Toumasis, P; Giannoulis, G; Apostolopoulos, D; ,End-to-end real-time service provisioning over a SDN-controllable analog mmWave fiber-wireless 5G X-Haul network,Journal of Lightwave Technology,41,4,1104-1113,2023,IEEE

7. Kanta K, Toumasis P, Giannoulis G, Stratakos I, Lentaris G, Papatheofanous EA, Mesogiti I, Theodoropoulou E, Margaris A, Syrivelis D, Kyriazi E, Brestas G, Tokas K, Argyris N, Vagionas C, Maximidis R, Bakopoulos P, Mesodiakaki A, Gatzianas M, Kalfas G, Tsagkaris K, Pleros N, Reisis D, Lyberopoulos G, Apostolopoulos D, Soudris D, Avramopoulos H, "Live demonstration of an SDN-reconfigurable, FPGA-based TxRx for an analog-IFoF/mmWave radio access network in an MNO's infrastructure", Journal of Optical Communications and Networking, 15, 8, C299-C306, 2023, Optica Publishing Group

Conferences

1. Stratakos, Ioannis; Reisis, Dionysios; Lentaris, George; Maragos, Konstantinos; Soudris, Dimitrios;, A co-design approach for rapid prototyping of image processing on soc fpgas,Proceedings of the 20th Pan-Hellenic Conference on Informatics„,1-6,2016,

2. Maragos, Konstantinos; Lentaris, George; Stratakos, Ioannis; Soudris, Dimitrios;, A framework exploiting process variability to improve energy efficiency in FPGA applications,Proceedings of the 2018 on Great Lakes Symposium on VLSI„,87-92,2018,

3. Lentaris, George; Stratakos, Ioannis; Stamoulias, Ioannis; Maragos, Konstantinos; Soudris, Dimitrios; Lourakis, Manolis; Zabulis, Xenophon; Gonzalez-Arjona, David; ",Project HIPNOS: Case study of high performance avionics for active debris removal in space,2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)„,350-355,2017,IEEE

4. Stratakos, Ioannis; Gourounas, Dimitrios; Tsoutsouras, Vasileios; Economopoulos, Theodore; Matsopoulos, George; Soudris, Dimitrios; ",Hardware acceleration of image registration algorithm on fpga-based systems on chip,Proceedings of the International Conference on omni-layer intelligent systems„,92-97,2019,

5. Stratakos, Ioannis; Maragos, Konstantinos; Lentaris, George; ,Voltage Scaling and Guardband Customization of Multiple Constituent Components in SoC-FPGA,"2019 29th International Symposium on Power and Timing Modeling, Optimization and

Simulation (PATMOS)"„,75-80,2019,IEEE

6. Maragos, Konstantinos; Taka, Endri; Lentaris, George; Stratakos, Ioannis; Soudris, Dimitrios; ,Analysis of performance variation in 16nm finfet fpga devices,2019 29th International Conference on Field Programmable Logic and Applications (FPL)„,38-44,2019,IEEE

7. Kitsakis, Vasileios; Kanta, Konstantina; Stratakos, Ioannis; Giannoulis, Giannis; Apostolopoulos, Dimitrios; Lentaris, George; Avramopoulos, Hercules; Soudris, Dimitrios; Reisis, Dionysios I; ,Design of a Real-Time DSP Engine on RF-SoC FPGA for 5G Networks,"Optical Network Design and Modeling: 23rd IFIP WG 6.10 International Conference, ONDM 2019, Athens, Greece, May 13–16, 2019, Proceedings 23"„,540-551,2020,Springer International Publishing

8. Leon, Vasileios; Stratakos, Ioannis; Armeniakos, Giorgos; Lentaris, George; Soudris, Dimitrios; ",ApproxQAM: High-Order QAM Demodulation Circuits with Approximate Arithmetic,2021 10th International Conference on Modern Circuits and Systems Technologies (MOCAST)„,1-5,2021,IEEE

9. Toumasis, P; Kanta, K; Tokas, K; Stratakos, I; Papatheofanous, EA; Giannoulis, G; Mesogiti, I; Theodoropoulou, E; Lyberopoulos, G; Lentaris, G; ",Demonstration of FPGA-based A-IFoF/mmWave transceiver integration in mobile infrastructure for beyond 5G transport,2021 European Conference on Optical Communication (ECOC)„,1-4,2021,IEEE

10. Stratakos, Ioannis; Papatheofanous, Elissaios Alexios; Danopoulos, Dimitrios; Lentaris, George; Reisis, Dionysios; Soudris, Dimitrios; ",Towards sharing one FPGA SoC for both low-level PHY and high-level AI/ML computing at the edge,2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)„,76-81,2021,IEEE

11. Vagionas, Christos; Maximidis, R; Stratakos, Ioannis; Margaris, Aristotelis; Mesodiakaki, Agapi; Gatzianas, Marios; Kanta, K; Toumasis, Panagiotis; Giannoulis, Giannis; Apostolopoulos, Dimitrios; ",End-to-End Real-Time Service Provisioning over a SDN-controllable 60 GHz analog FiWi X-haul for 5G Hot-Spot Networks,Optical Fiber Communication Conference,Th4A. 7,2022,Optica Publishing Group

12. Kanta, K; Toumasis, P; Giannoulis, G; Stratakos, I; Lentaris, G; Papatheofanous, EA; Mesogiti, I; Theodoropoulou, E; Margaris, A; Syrivelis, D;, "End-to-End Demonstration of an SDN-reconfigurable, FPGA-based TxRx Interface for Analog-IFoF/mmWave

X-haul", European Conference and Exhibition on Optical Communication, We2F. 6, 2022, Optica Publishing Group

Book Chapter

1. Stratakos, Ioannis; Maragos, Konstantinos; Lentaris, George; Soudris, Dimitrios; Siozios, Kostas;, Aging Evaluation and Mitigation Techniques Targeting FPGA Devices,"Low-Power Circuits for Emerging Applications in Communications, Computing, and Sensing",131-149,2018,CRC Press

Bibliography

[1] J. A. Stankovic, "Research directions for the internet of things," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 3–9, 2014.

[2] G. Lentaris *et al.*, "High-performance embedded computing in space: Evaluation of platforms for vision-based navigation," *J. Aerosp. Inf. Syst.*, vol. 15, no. 4, pp. 178–192, 2018.

[3] C. Kachris and D. Soudris, "A survey on reconfigurable accelerators for cloud computing," in *Field Programmable Logic and Applications (FPL), 2016 26th International Conference on*, pp. 1–10, IEEE, 2016.

[4] C. Kachris, B. Falsafi, and D. Soudris, *Hardware Accelerators in Data Centers*. Springer, 2019.

[5] I. Kuon and J. Rose, "Measuring the gap between fpgas and asics," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 2, pp. 203–215, 2007.

[6] J. L. Hennessy and D. A. Patterson, *Computer Architecture, Sixth Edition: A Quantitative Approach*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 6th ed., 2017.

[7] L. H. Crockett, R. A. Elliot, M. A. Enderwitz, and R. W. Stewart, *The Zynq book: embedded processing with the ARM Cortex-A9 on the Xilinx Zynq-7000 all programmable SoC*. Strathclyde Academic Media, 2014.

[8] I. Stratakos, D. Reisis, G. Lentaris, K. Maragos, and D. Soudris, "A co-design approach for rapid prototyping of image processing on soc fpgas," in *Proceedings of the 20th Pan-Hellenic Conference on Informatics*, pp. 1–6, 2016.

[9] Xilinx, "How fpgas have evolved to address compute acceleration and interconnects." Available at https://www.xilinx.com/publications/events/OFC/how_fpgfs_have_evolved_ofc_03042019.pdf, 2019.

[10] Xilinx, "Zynq ultrascale+ rfsoc." Available at https://www.xilinx.com/products/silicon-devices/soc/rfsoc.html.

[11] Xilinx, "What's a versal adaptive soc." Available at https://www.xilinx.com/products/silicon-devices/acap/versal.html.

[12] J. Hamkins, "Performance of low-density parity-check coded modulation," in *2010 IEEE Aerospace Conference*, pp. 1–14, 2010.

[13] A. Viterbi, "An intuitive justification and a simplified implementation of the map decoder for convolutional codes," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 2, pp. 260–264, 1998.

[14] F. Tosato and P. Bisaglia, "Simplified soft-output demapper for binary interleaved cofdm with application to hiperlan/2," in *2002 IEEE International Conference on Communications. Conference Proceedings. ICC 2002 (Cat. No.02CH37333)*, vol. 2, pp. 664–668 vol.2, 2002.

[15] E. Yoon, "Maximum likelihood detection with a closed-form solution for the square qam constellation," *IEEE Communications Letters*, vol. 21, no. 4, pp. 829–832, 2017.

[16] R. Omidi and S. Sharifzadeh, "Design of low power approximate floating-point adders," *International Journal of Circuit Theory and Applications*, vol. 49, no. 1, pp. 185–195, 2021.

[17] V. Leon, G. Zervakis, D. Soudris, and K. Pekmestzi, "Approximate hybrid high radix encoding for energy-efficient inexact multipliers," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 3, pp. 421–430, 2018.

[18] A. Long, M. Richards, and D. E. Hastings, "On-orbit servicing: A new value proposition for satellite design and operation," *J. Spacecr. Rockets*, vol. 44, pp. 964–976, 2007.

[19] A. Flores-Abad *et al.*, "A review of space robotics technologies for on-orbit servicing," *Prog. Aerosp. Sci.*, vol. 68, pp. 1–26, Jul. 2014.

[20] R. Biesbroek, L. Innocenti, A. Wolahan, and S. M. Serrano, "e.Deorbit - ESA's active debris removal mission," in *Eur. Conf. Space Debris*, 2017.

[21] J. Kelsey *et al.*, "Vision-based relative pose estimation for autonomous rendezvous and docking," in *IEEE Aerosp. Conf.*, pp. 1–20, 2006.

[22] "https://www.xilinx.com/products/boards-and-kits/zcu111.html."

[23] C. Vagionas, R. Maximidis, I. Stratakos, A. Margaris, A. Mesodiakaki, M. Gatzianas, K. Kanta, P. Toumasis, G. Giannoulis, D. Apostolopoulos, *et al.*, "End-to-end real-time service provisioning over a sdn-controllable analog mmwave fiber-wireless 5g x-haul network," *Journal of Lightwave Technology*, vol. 41, no. 4, pp. 1104–1113, 2023.

[24] K. Kanta, P. Toumasis, G. Giannoulis, I. Stratakos, G. Lentaris, E. A. Papatheofanous, I. Mesogiti, E. Theodoropoulou, A. Margaris, D. Syrivelis, E. Kyriazi, G. Brestas, K. Tokas, N. Argyris, C. Vagionas, R. Maximidis, P. Bakopoulos, A. Mesodiakaki, M. Gatzianas, G. Kalfas, K. Tsagkaris, N. Pleros, D. Reisis, G. Lyberopoulos, D. Apostolopoulos, D. Soudris, and H. Avramopoulos, "Live demonstration of an sdn-reconfigurable, fpga-based txrx for an analog-ifof/mmwave radio access network in an mno's infrastructure," *J. Opt. Commun. Netw.*, vol. 15, pp. C299–C306, Aug 2023.

[25] "http://www.5g-phos.eu/."

[26] A. R. Omondi and J. C. Rajapakse, *FPGA Implementations of Neural Networks.*

Berlin, Heidelberg: Springer-Verlag, 2006.

[27] I. Stratakos, V. Leon, G. Armeniakos, G. Lentaris, and D. Soudris, "Design space exploration on high-order qam demodulation circuits: Algorithms, arithmetic and approximation techniques," *Electronics*, vol. 11, no. 1, p. 39, 2021.

[28] V. Leon, I. Stratakos, G. Armeniakos, G. Lentaris, and D. Soudris, "Approxqam: High-order qam demodulation circuits with approximate arithmetic," in *2021 10th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, pp. 1–5, IEEE, 2021.

[29] I. Stratakos, K. Maragos, and G. Lentaris, "Voltage scaling and guardband customization of multiple constituent components in soc-fpga," in *2019 29th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pp. 75–80, IEEE, 2019.

[30] I. Stratakos, E. A. Papatheofanous, D. Danopoulos, G. Lentaris, D. Reisis, and D. Soudris, "Towards sharing one fpga soc for both low-level phy and high-level ai/ml computing at the edge," in *2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*, pp. 76–81, IEEE, 2021.

[31] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. K. Soong, and J. C. Zhang, "What will 5g be?," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 6, pp. 1065–1082, 2014.

[32] I. Kuon, R. Tessier, and J. Rose, "Fpga architecture: Survey and challenges," *Foundations and Trends® in Electronic Design Automation*, vol. 2, no. 2, pp. 135–253, 2008.

[33] V. Leon, I. Stamoulias, G. Lentaris, D. Soudris, D. Gonzalez-Arjona, R. Domingo, D. M. Codinachs, and I. Conway, "Development and testing on the european space-grade brave fpgas: Evaluation of ng-large using high-performance dsp benchmarks," *IEEE Access*, vol. 9, pp. 131877–131892, 2021.

[34] T. H. Pham, S. A. Fahmy, and I. V. McLoughlin, "An end-to-end multi-standard ofdm transceiver architecture using fpga partial reconfiguration," *IEEE Access*, vol. 5, pp. 21002–21015, 2017.

[35] M. L. Ferreira and J. C. Ferreira, "Reconfigurable nc-ofdm processor for 5g communications," in *2015 IEEE 13th International Conference on Embedded and Ubiquitous Computing*, pp. 199–204, 2015.

[36] M. L. Ferreira, A. Barahimi, and J. C. Ferreira, "Reconfigurable fpga-based fft processor for cognitive radio applications," in *Applied Reconfigurable Computing* (V. Bonato, C. Bouganis, and M. Gorgon, eds.), (Cham), pp. 223–232, Springer International Publishing, 2016.

[37] R. Deng, J. He, M. Chen, and L. Chen, "Sfo compensation by pilot-aided channel estimation for real-time ddo-ofdm system," *Optics Communications*, vol. 355, pp. 172–176, 2015.

[38] S. Rommel, E. Grivas, B. Cimoli, D. Dodane, A. Morales, E. Pikasis, J. Bourderionnet, G. Feugnet, J. Barros Carvalho, M. Katsikis, K. Ntontin, D. Kritharidis,

I. Spaleniak, P. Mitchell, M. Dubov, and I. Tafur Monroy, "Real-time high-bandwidth mm-wave 5g nr signal transmission with analog radio-over-fiber fronthaul over multi-core fiber," *EURASIP Journal on Wireless Communications and Networking*, vol. 2021, p. 43, Feb 2021.

[39] H. Jiang, J. Han, F. Qiao, and F. Lombardi, "Approximate radix-8 booth multipliers for low-power and high-performance operation," *IEEE Transactions on Computers*, vol. 65, no. 8, pp. 2638–2644, 2016.

[40] W. Liu, L. Qian, C. Wang, H. Jiang, J. Han, and F. Lombardi, "Design of approximate radix-4 booth multipliers for error-tolerant computing," *IEEE Transactions on Computers*, vol. 66, no. 8, pp. 1435–1441, 2017.

[41] V. Mrazek, R. Hrbacek, Z. Vasicek, and L. Sekanina, "Evoapprox8b: Library of approximate adders and multipliers for circuit design and benchmarking of approximation methods," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pp. 258–261, 2017.

[42] V. Leon, K. Asimakopoulos, S. Xydis, D. Soudris, and K. Pekmestzi, "Cooperative arithmetic-aware approximation techniques for energy-efficient multipliers," in *Proceedings of the 56th Annual Design Automation Conference 2019*, DAC '19, (New York, NY, USA), Association for Computing Machinery, 2019.

[43] V. Leon, G. Zervakis, S. Xydis, D. Soudris, and K. Pekmestzi, "Walking through the energy-error pareto frontier of approximate multipliers," *IEEE Micro*, vol. 38, no. 4, pp. 40–49, 2018.

[44] V. Leon, T. Paparouni, E. Petrongonas, D. Soudris, and K. Pekmestzi, "Improving power of dsp and cnn hardware accelerators using approximate floating-point multipliers," *ACM Trans. Embed. Comput. Syst.*, vol. 20, jul 2021.

[45] A. B. Kahng and S. Kang, "Accuracy-configurable adder for approximate arithmetic designs," in *Proceedings of the 49th Annual Design Automation Conference*, DAC '12, (New York, NY, USA), p. 820–825, Association for Computing Machinery, 2012.

[46] M. Shafique, W. Ahmad, R. Hafiz, and J. Henkel, "A low latency generic accuracy configurable adder," in *Proceedings of the 52nd Annual Design Automation Conference*, DAC '15, (New York, NY, USA), Association for Computing Machinery, 2015.

[47] V. Leon, K. Pekmestzi, and D. Soudris, "Exploiting the potential of approximate arithmetic in dsp & ai hardware accelerators," in *2021 31st International Conference on Field-Programmable Logic and Applications (FPL)*, pp. 263–264, 2021.

[48] V. Mrazek, Z. Vasicek, L. Sekanina, M. A. Hanif, and M. Shafique, "Alwann: Automatic layer-wise approximation of deep neural network accelerators without retraining," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–8, 2019.

[49] V. Leon, I. Stratakos, G. Armeniakos, G. Lentaris, and D. Soudris, "Approxqam: High-order qam demodulation circuits with approximate arithmetic," in *2021 10th*

International Conference on Modern Circuits and Systems Technologies (MOCAST), pp. 1–5, 2021.

[50] G. Lentaris, G. Chatzitsompanis, V. Leon, K. Pekmestzi, and D. Soudris, "Combining arithmetic approximation techniques for improved cnn circuit design," in *2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pp. 1–4, 2020.

[51] R. Ragavan, B. Barrois, C. Killian, and O. Sentieys, "Pushing the limits of voltage over-scaling for error-resilient applications," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pp. 476–481, 2017.

[52] X. Jiao, Y. Jiang, A. Rahimi, and R. K. Gupta, "Slot: A supervised learning model to predict dynamic timing errors of functional units," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pp. 1183–1188, 2017.

[53] A. TASISSA, "Function approximation and the remez algorithm," 2019.

[54] V. Leon, S. Xydis, D. Soudris, and K. Pekmestzi, "Energy-efficient vlsi implementation of multipliers with double lsb operands," *IET Circuits, Devices & Systems*, vol. 13, no. 6, pp. 816–821, 2019.

[55] G. Lentaris, K. Maragos, I. Stratakos, L. Papadopoulos, O. Papanikolaou, D. Soudris, M. Lourakis, X. Zabulis, D. Gonzalez-Arjona, and G. Furano, "High-performance embedded computing in space: Evaluation of platforms for vision-based navigation," *Journal of Aerospace Information Systems*, vol. 15, no. 4, pp. 178–192, 2018.

[56] A. Amara, F. Amiel, and T. Ea, "FPGA vs. ASIC for low power applications," *Microelectronics Journal*, vol. 37, no. 8, pp. 669–677, 2006.

[57] E. Bezati, S. Casale-Brunet, M. Mattavelli, and J. W. Janneck, "Clock-gating of streaming applications for energy efficient implementations on FPGAs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 4, pp. 699–703, 2017.

[58] A. K. Sultania, C. Zhang, D. K. Gandhi, and F. Zhang, "Power analysis and optimization," in *Designing with Xilinx® FPGAs*, pp. 177–187, Springer, 2017.

[59] J. M. Levine, E. Stott, and P. Y. Cheung, "Dynamic voltage & frequency scaling with online slack measurement," in *International Symposium on Field-programmable Gate Arrays FPGA*, pp. 65–74, ACM, 2014.

[60] J. L. Nunez-Yanez, M. Hosseinabady, and A. Beldachi, "Energy optimization in commercial FPGAs with voltage, frequency and logic scaling," *IEEE Transactions on Computers*, vol. 65, no. 5, pp. 1484–1493, 2016.

[61] I. Ahmed, S. Zhao, O. Trescases, and V. Betz, "Measure twice and cut once: Robust dynamic voltage scaling for FPGAs," in *International Conference on Field Programmable Logic and Applications (FPL)*, pp. 1–11, IEEE, 2016.

[62] K. Maragos, G. Lentaris, I. Stratakos, and D. Soudris, "A framework exploiting process variability to improve energy efficiency in FPGA applications," in *Great Lakes Symposium on VLSI (GLSVLSI)*, pp. 87–92, ACM, 2018.

[63] C. G. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the Alvey Vision Conference, AVC 1988, Manchester, UK, September, 1988* (C. J. Taylor, ed.), pp. 1–6, Alvey Vision Club, 1988.

[64] H. Sharma, J. Park, N. Suda, L. Lai, B. Chau, J. K. Kim, V. Chandra, and H. Esmaeilzadeh, "Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network," *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pp. 764–775, 2018.

[65] Y. Guan, H. Liang, N. Xu, W. Wang, S. Shi, X. Chen, G. Sun, W. Zhang, and J. Cong, "Fp-dnn: An automated framework for mapping deep neural networks onto fpgas with rtl-hls hybrid templates," in *2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pp. 152–159, 2017.

[66] K. Guo, L. Sui, J. Qiu, S. Yao, S. Han, Y. Wang, and H. Yang, "Angel-eye: A complete design flow for mapping cnn onto customized hardware," in *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 24–29, 2016.

[67] A. Shawahna, S. M. Sait, and A. El-Maleh, "Fpga-based accelerators of deep learning networks for learning and classification: A review," *IEEE Access*, vol. 7, pp. 7823–7859, 2019.

[68] M. Wess, S. M. P D, and A. Jantsch, "Neural network based ecg anomaly detection on fpga and trade-off analysis," pp. 1–4, 05 2017.

[69] D. Danopoulos, C. Kachris, and D. Soudris, "Utilizing cloud fpgas towards the open neural network standard," *Sustainable Computing: Informatics and Systems*, vol. 30, p. 100520, 2021.

[70] R. Nane, V. Sima, C. Pilato, J. Choi, B. Fort, A. Canis, Y. Chen, H. Hsiao, S. Brown, F. Ferrandi, J. Anderson, and K. Bertels, "A survey and evaluation of fpga high-level synthesis tools," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, pp. 1–1, 12 2015.

[71] I. Stratakos, D. Gourounas, V. Tsoutsouras, T. Economopoulos, G. Matsopoulos, and D. Soudris, "Hardware acceleration of image registration algorithm on fpga-based systems on chip," in *Proceedings of the International Conference on omni-layer intelligent systems*, pp. 92–97, 2019.

[72] G. Lentaris, I. Stratakos, I. Stamoulias, D. Soudris, M. Lourakis, and X. Zabulis, "High-performance vision-based navigation on soc fpga for spacecraft proximity operations," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 4, pp. 1188–1202, 2019.

[73] E. J. Topol and D. Hill, *The creative destruction of medicine: How the digital revolution will create better health care.*
Basic Books New York, 2012.

[74] T. M. Deserno, *Biomedical image processing.*
Springer Science & Business Media, 2011.

[75] J. J. Koo, A. C. Evans, and W. J. Gross, "3-d brain mri tissue classification on

fpgas," *IEEE Transactions on Image Processing*, vol. 18, no. 12, pp. 2735–2746, 2009.

[76] M. F. B. Othman, N. Abdullah, and N. A. B. A. Rusli, "An overview of mri brain classification using fpga implementation," in *Industrial Electronics & Applications (ISIEA), 2010 IEEE Symposium on*, pp. 623–628, IEEE, 2010.

[77] C. R. Castro-Pareja, J. M. Jagadeesh, and R. Shekhar, "Fair: a hardware architecture for real-time 3-d image registration," *IEEE Transactions on Information Technology in Biomedicine*, vol. 7, no. 4, pp. 426–434, 2003.

[78] J. Chen, A. C. Yu, and H. K.-H. So, "Design considerations of real-time adaptive beamformer for medical ultrasound research using fpga and gpu," in *Field-Programmable Technology (FPT), 2012 International Conference on*, pp. 198–205, IEEE, 2012.

[79] N. Nkanza, *Image registration and its application to computer vision: mosaicing and independant motion detection.*
PhD thesis, University of Cape Town, 2005.

[80] H. Peng, P. Chung, F. Long, L. Qu, A. Jenett, A. M. Seeds, E. W. Myers, and J. H. Simpson, "Brainaligner: 3d registration atlases of drosophila brains," *Nature methods*, vol. 8, no. 6, p. 493, 2011.

[81] G. K. Matsopoulos, N. A. Mouravliansky, K. K. Delibasis, and K. S. Nikita, "Automatic retinal image registration scheme using global optimization techniques," *IEEE Transactions on Information Technology in Biomedicine*, vol. 3, no. 1, pp. 47–60, 1999.

[82] Y. Bentoutou, N. Taleb, K. Kpalma, and J. Ronsin, "An automatic image registration for applications in remote sensing," *IEEE transactions on geoscience and remote sensing*, vol. 43, no. 9, pp. 2127–2137, 2005.

[83] I. V. Maslov, "Automatic image registration and target recognition with multiresolution hybrid evolutionary algorithm," in *Signal Processing, Sensor Fusion, and Target Recognition XIII*, vol. 5429, pp. 180–188, International Society for Optics and Photonics, 2004.

[84] U. S. Shah and D. Mistry, "Survey of image registration techniques for satellite images," *International Journal for Scientific Research & Development*, vol. 1, no. 11, pp. 2321–0613, 2014.

[85] L. G. Brown, "A survey of image registration techniques," *ACM computing surveys (CSUR)*, vol. 24, no. 4, pp. 325–376, 1992.

[86] A. A. Goshtasby, *2-D and 3-D image registration: for medical, remote sensing, and industrial applications.*
John Wiley & Sons, 2005.

[87] R. Shams, P. Sadeghi, R. A. Kennedy, and R. I. Hartley, "A survey of medical image registration on multicore and the gpu," *IEEE signal processing magazine*, vol. 27, no. 2, pp. 50–60, 2010.

[88] J. F. Hughes, A. Van Dam, J. D. Foley, M. McGuire, S. K. Feiner, D. F. Sklar, and

K. Akeley, *Computer graphics: principles and practice.*
Pearson Education, 2014.

[89] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical recipes 3rd edition: The art of scientific computing.*
Cambridge university press, 2007.

[90] D. G. Bailey, *Design for Embedded Image Processing on FPGAs.*
Wiley Publishing, 1st ed., 2011.

[91] C. English *et al.*, "Real-time dynamic pose estimation systems in space: Lessons learned for system design and performance evaluation," *Intl. J. Intell. Contr. Syst.*, vol. 16, no. 2, pp. 79–96, 2011.

[92] V. Lepetit and P. Fua, "Monocular model-based 3D tracking of rigid objects: A survey," *Found. & Trends. Comp. Graph. Vision*, vol. 1, 2005.

[93] C. Harris, "Tracking with rigid objects," in *Active Vision* (A. Blake and A. Yuille, eds.), pp. 59–73, MIT Press, 1992.

[94] M. Armstrong and A. Zisserman, "Robust object tracking," in *Asian Conf. on Computer Vision*, vol. I, pp. 58–61, 1995.

[95] T. Drummond and R. Cipolla, "Real-time visual tracking of complex structures," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 932–946, 2002.

[96] A. I. Comport *et al.*, "Real-time markerless tracking for augmented reality: The virtual visual servoing framework," *IEEE Trans. Vis. Comput. Graph.*, vol. 12, no. 4, pp. 615–628, 2006.

[97] K. Kanani *et al.*, "Vision based navigation for debris removal missions," in *63rd Intl. Astronaut. Congr.*, 2012.
IAC-12.

[98] A. Petit, *Robust visual detection and tracking of complex objects: applications to space autonomous rendez-vous and proximity operations.*
PhD thesis, Université Rennes 1, Dec. 2013.

[99] Y. Zhang *et al.*, "Comparative study of visual tracking method: A probabilistic approach for pose estimation using lines," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 6, pp. 1222–1234, 2017.

[100] G. Zhang *et al.*, "Cooperative relative navigation for space rendezvous and proximity operations using controlled active vision," *J. Field Robot.*, vol. 33, no. 2, pp. 205–228, 2016.

[101] X. Du *et al.*, "Pose measurement of large non-cooperative satellite based on collaborative cameras," *Acta Astronaut.*, vol. 68, no. 11, pp. 2047–2065, 2011.

[102] J. Peng, W. Xu, and H. Yuan, "An efficient pose measurement method of a space non-cooperative target based on stereo vision," *IEEE Access*, vol. 5, pp. 22344–22362, 2017.

[103] G. Lentaris *et al.*, "HW/SW Co-design and FPGA Acceleration of Visual Odometry Algorithms for Rover Navigation on Mars," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 8, pp. 1563–1577, 2016.

[104] B. Naasz *et al.*, "The HST SM4 relative navigation sensor system: Overview and preliminary testing results from the flight robotics lab," *J. Astronaut. Sci.*, vol. 57, pp. 457–483, Jan 2009.

[105] M. Lourakis and X. Zabulis, "Model-based visual tracking of orbiting satellites using edges," in *Proc. IEEE/RSJ Intl. Conf. on Intell. Robot. Syst.*, pp. 3791–3796, 2017.

[106] R. Opromolla *et al.*, "A review of cooperative and uncooperative spacecraft pose determination techniques for close-proximity operations," *P. Aerosp. Sci.*, vol. 93, pp. 53–72, 2017.

[107] S. Sharma and S. D'Amico, "Comparative assessment of techniques for initial pose estimation using monocular vision," *Acta Astronaut.*, vol. 123, pp. 435–445, 2016.

[108] L. Zhong, M. Lu, and L. Zhang, "A direct 3D object tracking method based on dynamic textured model rendering and extended dense feature fields," *IEEE Trans. Circuits Syst. Video Technol.*, 2017.

[109] X. Iturbe *et al.*, "An integrated SoC for science data processing in next-generation space flight instruments avionics," in *2015 IFIP/IEEE Intl. Conf. on Very Large Scale Integration (VLSI-SoC)*, pp. 134–141, 2015.

[110] D. Rudolph *et al.*, "CSP: A multifaceted hybrid architecture for space computing," in *SSC14-III-3, 28th Annual AIAA/USU Conference on Small Satellites*, 2014.

[111] Z. Zhang *et al.*, "Visual-inertial odometry on chip: An algorithm-and-hardware co-design approach," in *Robotics: Science and Systems*, 2017.

[112] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, pp. 679–698, Nov. 1986.

[113] T. Möller and B. Trumbore, "Fast, minimum storage ray-triangle intersection," *J. Graph. Tools*, vol. 2, no. 1, pp. 21–28, 1997.

[114] P. J. Rousseeuw, "Least median of squares regression," *J. Amer. Stat. Assoc.*, vol. 79, no. 388, pp. 871–880, 1984.

[115] Xillybus, "Xillybus FPGA designer's guide." Available at http://xillybus.com/downloads/doc/.

[116] C.-L. Sotiropoulou *et al.*, "Real-time machine vision FPGA implementation for microfluidic monitoring on lab-on-chips," *IEEE Trans. Biomed. Circuits Syst.*, vol. 8, no. 2, pp. 268–277, 2014.

[117] J. Lee, H. Tang, and J. Park, "Energy efficient Canny edge detector for advanced mobile vision applications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 4, pp. 1037–1046, 2018.

[118] X. Wang, F. Guo, and M. Zhu, "A more efficient triangle rasterization algorithm implemented in FPGA," in *Intl. Conf. Audio, Language and Image Processing (ICALIP)*, pp. 1108–1113, IEEE, 2012.

[119] Z. Safarzik, E. Gervais, and M. Vucic, "Implementation of division-free perspective-correct rendering optimized for FPGA devices," in *Intl. Convention MIPRO*, pp. 177–182, 2010.

[120] G. Hausmann *et al.*, "VIBANASS (vision based navigation sensor system) system

test results," in *Eur. Conf. Space Debris*, vol. 723, 2013.
[121] T. Chabot *et al.*, "Vision-based navigation experiment onboard the RemoveDebris mission," in *Intl. ESA Conf. on GNC Systems*, pp. 1–23, 2017.
[122] K. Kanta, P. Toumasis, K. Tokas, I. Stratakos, E. A. Papatheofanous, G. Giannoulis, I. Mesogiti, E. Theodoropoulou, G. Lyberopoulos, G. Lentaris, *et al.*, "Demonstration of a hybrid analog–digital transport system architecture for 5g and beyond networks," *Applied Sciences*, vol. 12, no. 4, p. 2122, 2022.
[123] C. Vagionas, R. Maximidis, I. Stratakos, A. Margaris, A. Mesodiakaki, M. Gatzianas, K. Kanta, P. Toumasis, G. Giannoulis, D. Apostolopoulos, *et al.*, "End-to-end real-time service provisioning over a sdn-controllable 60 ghz analog fiwi x-haul for 5g hot-spot networks," in *Optical Fiber Communication Conference*, pp. Th4A–7, Optica Publishing Group, 2022.
[124] K. Kanta, P. Toumasis, G. Giannoulis, I. Stratakos, G. Lentaris, E. Papatheofanous, I. Mesogiti, E. Theodoropoulou, A. Margaris, D. Syrivelis, *et al.*, "End-to-end demonstration of an sdn-reconfigurable, fpga-based txrx interface for analog-ifof/mmwave x-haul," in *European Conference and Exhibition on Optical Communication*, pp. We2F–6, Optica Publishing Group, 2022.
[125] P. Toumasis, K. Kanta, K. Tokas, I. Stratakos, E. Papatheofanous, G. Giannoulis, I. Mesogiti, E. Theodoropoulou, G. Lyberopoulos, G. Lentaris, *et al.*, "Demonstration of fpga-based a-ifof/mmwave transceiver integration in mobile infrastructure for beyond 5g transport," in *2021 European Conference on Optical Communication (ECOC)*, pp. 1–4, IEEE, 2021.
[126] K. Kanta, A. Pagano, E. Ruggeri, M. Agus, I. Stratakos, R. Mercinelli, C. Vagionas, P. Toumasis, G. Kalfas, G. Giannoulis, *et al.*, "Analog fiber-wireless downlink transmission of ifof/mmwave over in-field deployed legacy pon infrastructure for 5g fronthauling," *Journal of Optical Communications and Networking*, vol. 12, no. 10, pp. D57–D65, 2020.
[127] V. Kitsakis, K. Kanta, I. Stratakos, G. Giannoulis, D. Apostolopoulos, G. Lentaris, H. Avramopoulos, D. Soudris, and D. I. Reisis, "Design of a real-time dsp engine on rf-soc fpga for 5g networks," in *Optical Network Design and Modeling: 23rd IFIP WG 6.10 International Conference, ONDM 2019, Athens, Greece, May 13–16, 2019, Proceedings 23*, pp. 540–551, Springer International Publishing, 2020.
[128] "https://alertify.eu/154-operators-in-66-countries-have-demonstrated-testing-or-trialling-5g-technologies/."
[129] T. E. Bogale and L. B. Le, "Massive mimo and mmwave for 5g wireless hetnet: potential benefits and challenges," *IEEE Veh. Technol. Mag.*, vol. 11, 2016.
[130] "Keysight technologies: Mimo and beamforming in the 5g context, sbr t (2017)."
[131] "Schaich, f., wild, t.: Waveform contenders for 5g – ofdm vs. fbmc vs. ufmc. in: 2014 6th international symposium on communications, control and signal processing (isccsp), athens, 2014, pp. 457–460 (2014)."
[132] L. Giorgi, "Subcarrier multiplexing rf plans for analog radio over fiber in heteroge-

neous networks," *J. Lightw. Technol.*, vol. 34, 2016.

[133] H. Kim, "Rof-based optical fronthaul technology for 5g and beyond," in *2018 Optical Fiber Communications Conference and Exposition (OFC)*, pp. 1–3, 2018.

[134] "Chen, d.: Fronthaulbandwidth analysis and latency constraint considerations. ieee 802.1, san diego, july 2016."

[135] "Argyris, n., et al.: Dsp enabled fiber-wireless ifof/mmwave link for 5g analog mobile fronthaul. in: Proceedings of 2018 ieee 5g world forum, 9–11 july 2018, santa clara, ca, usa (2018)."

[136] "Osorio, c., et al.: Dual-mode distance-adaptive transceiver architecture for 5g optical fiber fronthaul. in: Proceedings of advances in wireless and optical communications 2018 (rtuwo 2018), 15-16 november 2018, riga, latvia (2018)."

[137] D. Chitimalla, "5g fronthaul-latency and jitter studies of cpri over ethernet," *IEEE/OSA J. Opt. Commun. Networking*, vol. 9, 2017.

[138] "Paulo, j., et al.: Fpga-based testbed for synchronization on ethernet fronthaul with phase noise measurements. in: Proceedings of 1st international symposium on instrumentation systems, circuits and transducers (inscit 2016), 29 august–03 september 2016, belo horizonte, brazil (2016)."

[139] Xilinx, "White Paper:An Adaptable Direct RF-Sampling Solution," tech. rep., February 20 2019.

[140] "3gpp, ts 38.104 v15.0.0, table 9.6.2.3-1, december 2017."